NAACL HLT 2019

The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies

Proceedings of the Conference Vol. 1
(Long and Short Papers)

June 2 - June 7, 2019

NAACL organizers gratefully acknowledge the generous support from the following sponsors.

Diamond

Bloomberg Engineering
Google
facebook
amazon
ASAPP

Platinum

2σ TWO SIGMA
Capital One
THOMSON REUTERS
DeepMind

Gold

Megagon Labs
ByteDance 字节跳动
Vanguard

Silver

duolingo
SAP
ebay

grammarly
Microsoft Research
cisco
interactions

Bronze

ETS
Measuring the Power of Learning.®

Raytheon
BBN Technologies

USC Viterbi
School of Engineering
Information Sciences Institute

clinc

Diversity & Inclusion: Champion

DeepMind

Diversity & Inclusion: Ally

Google

Social Event Entertainment Sponsor

CAPITAL
GROUP℠

In-Kind Donations

grammarly

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-950737-13-0 (Volume 1)

Message from the General Chair

Welcome to Minneapolis! NAACL 2019 promises to further build our growing and increasingly diverse community through substantive presentations, new diversity and inclusion initiatives, and a culturally-rich social event!

Christy Doran, Thamar Solorio, and Ted Pedersen, our 2019 Program Co-Chairs have gracefully managed the largest number of submissions at any NAACL to date – with submissions in 2019 almost doubling from the previous year. They demonstrated unrelenting dedication throughout the conference planning process to ensure preparation of a balanced program with thoughtfully-crafted and fair reviewing processes, on-time notifications, and a final careful selection of papers from a wide range of topics of interest represented in the main conference program. They also introduced a number of innovations in processes, including submission of abstracts before final submissions with the intention of getting a head start on the reviewing process and securing sufficient reviewers by topic; video lightning talks for posters to promote greater attendance and a preview of poster content; and, five categories of Best Papers: 1) Best Long Paper, 2) Best Short Paper, 3) Best Theme, 4) Best Resource, and 5) Best Explainable, to highlight a range of community values. The PCs introduced a conference theme this year that reflects a concern around ethics in our research community. Specifically, the theme examines the tension between data privacy and model bias in NLP. Our three invited talks, Arvind Narayanan, Rada Mihalcea, and Kieran Snyder will all present keynotes related to the conference theme.

In 2019, back by popular demand, there is an Industry Track, co-chaired by Rohit Kumar, Anastassia Loukina, and Michelle Morales. It will address practical issues in real-world deployment of Natural Language Processing and Speech Processing technologies that describe key lessons learned and new challenges posed by real-world implementations. There was an increase in the number of industry track submissions from 2018 (when the track was started) which suggests a continued interest in this track. A highlight this year is a Careers in NLP panel taking into consideration our more junior community members and their mentors.

As our international community grows, we can expect increasing diversity. Consistent with an awareness about diversity in the Natural Language Processing and Computational Linguistics community, at NAACL HLT 2019, we have introduced the Diversity and Inclusion (D&I) committee, co-chaired by Jason Eisner and Nathalie Schluter, and the Remote Presentation (RP) committee, co-chaired by Abhinav Misra and Meg Mitchell. The D&I committee was intended to support community concerns including, more diverse attendance through feasible childcare support efforts, community building through mentoring and social networking through the conference app, and comfort of all attendees through pronoun choice on badges and gender-neutral bathrooms. The RP committee responded to a concern to provide all members of our community with greater access to conferences, especially with regard to U.S. visa issues given the current political constraints. It is our hope that these new initiatives enrich the conference experience by further promoting greater access, and in turn, community-building.

On behalf of the Natural Language Processing and Computational Linguistics community, I would like to thank all of the organizers for their dedication, creativity, and lively communication that lead to a successful program and set of events: Christy Doran, Thamar Solorio and Ted Pedersen (program chairs); Rohit Kumar, Anastassia Loukina, and Michelle Morales (industrial track chairs); Nitin Madnani (website and app chair); Smaranda Muresan, Swapna Somasundaran, and Elena Volodina (workshop co-chairs); Anoop Sarkar and Michael Strube (tutorial co-chairs); Waleed Ammar, Annie Louis, and Nasrin Mostafazahdeh, (demo co-chairs); Jason Eisner and Nathalie Schulter (Diversity and Inclusion Co-Chairs) Stephanie Lukin and Alla Roskovskaya (publication co-chairs); Steve DeNeefe (handbook chair); Laura Burdick, Sudipta Kar, and Farah Nadeem (student co-chairs) along with Greg Durrett and Na-Rae-Han (Faculty Advisors) for the student research workshop; Lu Wang (student volunteer coordinator); Jason Baldridge and Alexis Palmer (the Americas International Sponsorship Team). Chris Callison-Burch and Tonya Custis (local sponsorships co-chairs); Yuval Pinter and Rachael Tatman (publicity and social media chairs); Abhinav Misra and Meg Mitchell (Remote Presentation co-chairs); Spencer Whitehead (video chair). Many thanks to Rich Gerber at SoftConf for on-going and rapid support. Many thanks to Julia Hockenmaier and the NAACL Executive Board for their on-going consultation, and Barbara Di Eugenio, Marti Hearst and David Yarowsky in their roles as ACL Conference Officer, ACL President, and ACL Treasurer, respectively. We also thank the Organizers of ACL 2019 and EMNLP 2019 for support in coordinating the programs, workshops and tutorials. We have twenty workshops plus the student research workshop. As we do every year, we owe many, many thanks to Priscilla Rasmussen for her guidance and moral support in addition to the mind-boggling task list associated with large-scale event planning, including managing exhibitors and our large sponsors. Thanks to Priscilla's efforts we have a great social event planned at the Minneapolis Institute of Art. In the spirit of community diversity, the museum offers internationally-themed galleries, and the food will reflect the themes.

We are immensely grateful to our sponsors for their generous contributions to NAACL 2019. Diamond sponsors are Amazon, ASAPP, Bloomberg Engineering, Facebook, and Google. Platinum sponsors are Capital One, DeepMind Google, Thomson Reuters, and Two Sigma. Our Gold sponsors are ByteDance and Megagon. Silver sponsors are Cisco, Duolingo, eBay, Grammarly, Microsoft Research, and SAP. Bronze sponsors are Clinc, ETS, Raytheon BBN Technology, and USC Viterbi School of Engineering/Information Sciences Institute. Additionally, Google is supporting our Diversity & Inclusion initiative and Grammarly also made an in-kind donation of Grammarly codes to help with proofreading. And, many of these same sponsors also generously support some of the workshops that make such a great finale to our conference. We are also pleased to welcome many companies who will participate in the Recruitment Lunch. There are many more people who through their hard work and dedication have contributed to make this conference a success: the area chairs and reviewers, tutorial presenters, workshop organizers, those who participated in D&I efforts, including student mentorship and the ACL Office staff. Many thanks to all of the presenters and conference attendees for your participation.

NAACL HLT 2019, General Chair
Jill Burstein, Educational Testing Service

Message from the Program Chairs

Welcome to Minneapolis and NAACL-HLT 2019! This conference is the largest by submission and acceptance volume of any NAACL to date, and it was through the fantastic hard work of the organizing committees, 94 Area Chairs, and 1321 reviewers that we were able to put together such a strong and varied program from a large pool of submissions. Similar to what other PCs have done in the past, we distributed a wide call for volunteers to recruit the Area Chairs and Reviewers–we seeded the areas with volunteers who responded, and then Area Chairs filled out the remainder of their respective committees. Our goal was to ensure greater diversity by including in each area some participants who may not have been previously involved, and therefore would not have been invited if the committees were built from lists of previous reviewers.

This year we followed a two-stage submission process, in which abstracts were due one week before full papers. Our goal was to get a head start on assigning papers to areas, and recruiting additional area chairs where submissions exceeded our predicted volume. Relative to the projected numbers from NAACL-HLT 2018, several areas received a higher-than-predicted number of submissions: Biomedical/Clinical, Dialogue and Vision. Text Mining ended up with the overall largest number of submissions. We used a hybrid reviewing form, combining elements of the EMNLP 2018, NAACL-HLT 2018 and ACL 2018, with a 6-point overall rating scale so there was no “easy out” mid-point, distinct sections of summary, strengths and weaknesses to make easy to scan and compare relevant sections, and the minimum length feature of START enabled to elicit more consistently substantive content for the authors.

Authors were permitted to switch format (long/short) when they submitted the full papers, so the total in the chart below uses 2271 as the total number of submissions, discounting the 103 that never submitted a full paper in the second phase. Seventy nine papers were desk-rejected due to anonymity, formatting, or dual-submission violations; 456 papers withdrawn prior to acceptance decisions being sent, although some were withdrawn part way through the review process; and an additional 11 papers were withdrawn after acceptance notifications had been sent. Keeping the acceptance rate consistent with past years meant we needed 5 parallel tracks to fit more papers into 3 days–as the conference grows, decisions will have to be made about continuing to add more tracks, adding more days to the main conference, or lowering the acceptance rate. The overall technical program consists of 423 main conference papers, plus 9 TACL papers, 23 SRW papers, 28 Industry papers, and 24 demos. The TACL and SRW papers are integrated into the program, and are marked SRW or TACL accordingly.

Acceptance break-down:

	Long	**Short**	**Total**	**TACL**
Reviewed	1067	666	1733	
Accepted as talk	140	72	212	4
Accepted as poster	141	70	211	5
Total Accepted	281 (26.3%)	142 (21.3%)	423 (24.4%)	9

A select group of Area Chairs was identified to make the Best Paper decisions, with independent teams assigned to select Best Long and Short Paper, Best Thematic Paper, Best Explainable NLP and Best Resource paper. The candidate papers were nominated by reviewers and/or Area Chairs.

It really takes a lot of volunteers' hard work to organize a NAACL conference. Our hearty thanks go out to:

- Jill Burstein, our fearless leader and General Conference Chair
- Priscilla Rasmussen, who knows everything and keeps the machine running
- Rich Gerber and his team at Softconf for having illuminated the secret corners of START and added several new features for us
- Recent past chairs–Amanda Stent, Heng Ji, Julia Hockenmaier, Emily Bender, Leon Derczynski, Iryna Gurevych and Yusuke Miyao–for answering piles of questions and generously sharing their documentation and resources
- All of the NAACL Organizing Committees
- The 43 session chairs (Alessandro Moschitti, Ani Nenkova, Anna Rumshisky, Bridget McInnes, Byron C. Wallace, Chenhao Tan, Daisuke Kawahara, Diyi Yang, Eduardo Blanco, Ekaterina Shutova, Emily Prud'hommeaux, Fei Liu, Gerard de Melo, Grzegorz Kondrak, Heng Ji, Ion Androutsopoulos, Kai-Wei Chang, Kevin Gimpel, Matt Gardner, Michael J. Paul, Mo Yu, Preslav Nakov, Roi Reichart, Ryan Cotterell, Saif Mohammad, Samuel Bowman, Sara Rosenthal, Serguei Pakhomov, Steven Bethard, Sujith Ravi, T. J. Hazen, Timothy Miller, Valia Kordoni, Vincent Ng, Wei Xu, William Yang Wang, Xiaodan Zhu, Yang Liu, Zornitsa Kozareva, Ellen Riloff, Colin Cherry, Joel Tetreault and Marine Carpuat)
- The special projects team: John Henderson for help detecting duplicate submissions and building us a clustering model to help with session creation, Sudipta Kar for additional help in detecting duplicate submissions, Cash de Leon and Jalen Tran for their help in putting together the slides for the poster highlights, Mahsa Shafaei for helping us screen volunteers, and Ted Pedersen for help getting this whole effort off the ground.

NAACL-HLT 2019 Program Co-Chairs
Christy Doran, Interactions, USA
Thamar Solorio, University of Houston, USA

Organizing Committee:

General Chair
Jill Burstein, ETS

Program Co-Chairs
Christy Doran, Interactions
Thamar Solorio, University of Houston

Industry Track Co-chairs
Rohit Kumar
Anastassia Loukina, Educational Testing Service, USA
Michelle Morales, IBM, USA

Workshop Co-Chairs
Smaranda Muresan, Columbia University, USA
Swapna Somasundaran, Educational Testing Service, USA
Elena Volodina, University of Gothenburg, Sweden

Tutorial Co-Chairs
Anoop Sarkar, Simon Fraser University, Canada
Michael Strube, Heidelberg Institute for Theoretical Studies, Germany

System Demonstration Co-Chairs
Waleed Ammar, Allen Institute for AI, USA
Annie Louis, University of Edinburgh, Scotland
Nasrin Mostafazadeh, Elemental Cognition, USA

Publication Co-Chairs
Stephanie Lukin, U.S. Army Research Laboratory
Alla Roskovskaya, City University of New York, USA

Handbook Chair
Steve DeNeefe, SDL, USA

Student Research Workshop Co-Chairs & Faculty Advisors
Sudipta Kar, University of Houston, USA
Farah Nadeem, University of Washington, USA
Laura Wendlandt, University of Michigan, USA
Greg Durrett, University of Texas at Austin, USA
Na-Rae Han, University of Pittsburgh, USA

Diversity & Inclusion Co-Chairs
Jason Eisner, Johns Hopkins University, USA
Natalie Schluter, IT University, Copenhagen, Denmark

Publicity & Social Media Co-Chairs
Yuval Pinter, Georgia Institute of Technology, USA
Rachael Tatman, Kaggle, USA

Website & Conference App Chair
Nitin Madnani, Educational Testing Service, USA

Student Volunteer Coordinator
Lu Wang, Northeastern University, USA

Video Chair
Spencer Whitehead, Rensselaer Polytechnic Institute, USA

Remote Presentation Co-Chairs
Meg Mitchell, Google, USA
Abhinav Misra, Educational Testing Service, USA

Local Sponsorship Co-Chairs
Chris Callison-Burch, University of Pennsylvania, USA
Tonya Custis, Thomson Reuters, USA

Local Organization
Priscilla Rasmussen, ACL

Area Chairs:

Biomedical NLP & Clinical Text Processing
Bridget McInnes, Virginia Commonwealth University, USA
Byron C. Wallace, Northeastern University, USA

Cognitive Modeling – Psycholinguistics
Serguei Pakhomov, University of Minnesota, USA
Emily Prud'hommeaux, Boston College, USA

Dialog and Interactive systems
Nobuhiro Kaji, Yahoo Japan Corporation, Japan
Zornitsa Kozareva, Google, USA
Sujith Ravi, Google, USA
Michael White, Ohio State University, USA

Discourse and Pragmatics
Ruihong Huang, Texas A&M University, USA
Vincent Ng, University of Texas at Dallas, USA

Ethics, Bias and Fairness
Saif Mohammad, National Research Council Canada, Canada
Mark Yatskar, University of Washington, USA

Generation
He He, Amazon Web Services, USA
Wei Xu, Ohio State University, USA
Yue Zhang, Westlake University, China

Information Extraction
Heng Ji, Rensselaer Polytechnic Institute, USA
David McClosky, Google, USA
Gerard de Melo, Rutgers University, USA
Timothy Miller, Boston Children's Hospital, USA
Mo Yu, IBM Research, USA

Information Retrieval
Sumit Bhatia, IBM's India Research Laboratory, India
Dina Demner-Fushman, US National Library of Medicine, USA

Machine Learning for NLP
Ryan Cotterell, Johns Hopkins University, USA
Daichi Mochihashi, The Institute of Statistical Mathematics, Japan
Marie-Francine Moens, KU Leuven, Belgium
Vikram Ramanarayanan, Educational Testing Service, USA
Anna Rumshisky, University of Massachusetts Lowell, USA
Natalie Schluter, IT University of Copenhagen, Denmark

Machine Translation
Rafael E. Banchs, Human Language Technology Institute for Infocomm Research A*Star, Singapore
Daniel Cer, Google Research, USA
Haitao Mi, Ant Financial US, USA
Preslav Nakov, Qatar Computing Research Institute, Qatar
Zhaopeng Tu, Tencent, China

Mixed Topics
Ion Androutsopoulos, Athens University of Economics and Business, Greece
Steven Bethard, University of Arizona, USA

Multilingualism, Cross lingual resources
Željko Agić, IT University of Copenhagen, Denmark
Ekaterina Shutova, University of Amsterdam, Netherlands
Yulia Tsvetkov, Carnegie Mellon University, USA
Ivan Vulic, Cambridge University, UK

NLP Applications
T. J. Hazen, Microsoft, USA
Alessandro Moschitti, Amazon, USA
Shimei Pan, University of Maryland Baltimore County, USA
Wenpeng Yin, University of Pennsylvania, USA
Su-Youn Yoon, Educational Testing Service, USA

Phonology, Morphology and Word Segmentation
Ramy Eskander, Columbia University, USA
Grzegorz Kondrak, University of Alberta, Canada

Question Answering
Eduardo Blanco, University of North Texas, USA
Christos Christodoulopoulos, Amazon, USA
Asif Ekbal, Indian Institute of Technology Patna, India
Yansong Feng, Peking University, China
Tim Rocktäschel, Facebook, USA
Avi Sil, IBM Research, USA

Resources and Evaluation
Torsten Zesch, University of Duisburg-Essen, Germany
Tristan Miller, Technische Universität Darmstadt, Germany

Semantics
Ebrahim Bagheri, Ryerson University, Canada
Samuel Bowman, New York University, USA
Matt Gardner, Allen Institute for Artificial Intelligence, USA
Kevin Gimpel, Toyota Technological Institute at Chicago, USA
Daisuke Kawahara, Kyoto University, Japan
Carlos Ramisch, Aix Marseille University, France

Sentiment Analysis
Isabelle Augenstein, University of Copenhagen, Denmark
Wai Lam, The Chinese University of Hong Kong, Hong Kong
Soujanya Poria, Nanyang Technological University, Singapore
Ivan Vladimir Meza Ruiz, Universidad Nacional Autónoma de México, Mexico

Social Media
Dan Goldwasser, Purdue University, USA
Michael J. Paul, University of Colorado Boulder, USA
Sara Rosenthal, IBM Research, USA
Paolo Rosso, Universitat Politècnica de València, Spain
Chenhao Tan, University of Colorado Boulder, USA
Xiaodan Zhu, Queen's University, Canada

Speech
Keelan Evanini, Educational Testing Service, USA
Yang Liu, LAIX Inc, USA

Style
Beata Beigman Klebanov, Educational Testing Service, USA
Manuel Montes y Gómez, Instituto Nacional de Astrofísica, Óptica y Eletrónica, Mexico
Joel Tetreault, Grammarly, USA

Summarization
Mohit Bansal, University of North Carolina Chapel Hill, USA
Fei Liu, University of Central Florida, USA
Ani Nenkova, University of Pennsylvania, USA

Tagging, Chunking, Syntax and Parsing
Adam Lopez, University of Edinburgh, Scotland
Roi Reichart, Technion – Israel Institute of Technology, Israel
Agata Savary, University of Tours, France
Guillaume Wisniewski, Université Paris Sud, France

Text Mining
Kai-Wei Chang, University of California Los Angeles, USA
Anna Feldman, Montlcair State University, USA
Shervin Malmasi, Harvard Medical School, USA
Verónica Pérez-Rosas, University of Michigan, USA
Kevin Small, Amazon, USA
Diyi Yang, Carnegie Mellon University, USA

Theory and Formalisms
Valia Kordoni, Humboldt University Berlin, Germany
Andreas Maletti, University of Stuttgart, Germany

Vision, Robotics and other grounding
Francis Ferraro, University of Maryland Baltimore County, USA
Vicente Ordóñez, University of Virginia, USA
William Yang Wang, University of California Santa Barbara, USA

Reviewers:

We would like to recognize the following reviewers as outstanding reviewers nominated by the Area Chairs:
Heike Adel, Miguel A. Alonso, Reinald Kim Amplayo, Caitrin Armstrong, Yoav Artzi, Timothy Baldwin, Jeremy Barnes, Gemma Boleda, Benjamin Börschinger, Cristina Bosco, Antoine Bosselut, Florian Boudin, Chris Brew, Jose Camacho-Collados, Ricardo Campos, Marie Candito, Tanmoy Chakraborty, Khyathi Chandu, Angel Chang, Francine Chen, Lin Chen, John Conroy, Chris Culy, Anna Currey, Rajarshi Das, Pradeep Dasigi, David DeVault, Bhuwan Dhingra, Georgiana Dinu, Mark Dredze, Ondřej Dušek, Hady Elsahar, Tobias Falke, Olivier Ferret, Michael Flor, Daniel Fried, Rashmi Gangadharaiah, Julio Gonzalo, Chulaka Gunasekara, Nitish Gupta, Hannaneh Hajishirzi, Hua He, Jack Hessel, Zhichao Hu, Peter Jansen, Kristiina Jokinen, David Jurgens, Chris Kedzie, Eliyahu Kiperwasser, Rebecca Knowles, Thomas Kober, Ari Kobren, Ekaterina Kochmar, Emiel Krahmer, Jayant Krishnamurthy, Vivek Kulkarni, Sachin Kumar, Junyi Jessy Li, Wenjie Li, Timm Lichte, Daniel Loureiro, Daniel Marcu, Scott Martin, Yann Mathet,

Yevgen Matusevych, Stephen Mayhew, Ryan McDonald, Nurit Melnik, Éva Mújdricza-Maydt, Yugo Murawaki, Sudip Kumar Naskar, Vivi Nastase, Graham Neubig, Mark Neumann, Vlad Niculae, Debora Nozza, Lilja Øvrelid, Gustavo Henrique Paetzold, Nikolaos Pappas, Natalie Parde, Ramakanth Pasunuru, Panupong Pasupat, Slav Petrov, Maxime Peyrard, Scott Piao, Heather Pon-Barry, Edoardo Maria Ponti, Vinodkumar Prabhakaran, Alexandre Rademaker, Marek Rei, Laura Rimell, Yu-Ping Ruan, Manfred Sailer, Sylvain Salvati, Nathan Schneider, Ethan Selfridge, Pararth Shah, Ian Stewart, Kristina Striegnitz, Alane Suhr, Xu Sun, Erik Tjong Kim Sang, Kentaro Torisawa, Sowmya Vajjala, Martin Villalba, Svitlana Volkova, Henning Wachsmuth, Joachim Wagner, Jakub Waszczuk, Ingmar Weber, Spencer Whitehead, Michael Wiegand, Adina Williams, Jiacheng Xu, Kevin Yancey, Zhilin Yang, Jin-ge Yao, Dian Yu, François Yvon, Justine Zhang, Wei Zhao, Victor Zhong, Li Zhou, Arkaitz Zubiaga

Many thanks to the almost 2,000 other reviewers as well!
Omri Abend, Sallam Abualhaija, Oliver Adams, Stergos Afantenos, Željko Agić, Wasi Ahmad, Mohammad Akbari, Alan Akbik, Md Shad Akhtar, Syed Sarfaraz Akhtar, Khalid Al Khatib, Nora Al-Twairesh, Firoj Alam, Nikolaos Aletras, Jan Alexandersson, Enrique Alfonseca, Dimitris Alikaniotis, Miltiadis Allamanis, Cissi Ovesdotter Alm, Hadi Amiri, Antonios Anastasopoulos, Ion Androutsopoulos, Anietie Andy, Krasimir Angelov, Mohammed Ansari, Marianna Apidianaki, Jun Araki, Timofey Arkhangelskiy, Ehsaneddin Asgari, Elliott Ash, Duygu Ataman, Kartik Audhkhasi, Joe Austerweil, Eleftherios Avramidis, Amittai Axelrod, Mahmoud Azab, Wilker Aziz, Joan Bachenko, AmirAli Bagher Zadeh, Sanaz Bahargam, Anusha Balakrishnan, Niranjan Balasubramanian, Miguel Ballesteros, David Bamman, Trapit Bansal, Francesco Barbieri, Verginica Barbu Mititelu, Gianni Barlacchi, Loïc Barrault, Alberto Barrón-Cedeño, Valerio Basile, Riza Batista-Navarro, Timo Baumann, Yonatan Belinkov, Iz Beltagy, Anja Belz, Yassine Benajiba, Farah Benamara, Emily M. Bender, Adrian Benton, Jonathan Berant, Dario Bertero, Nicola Bertoldi, Gayatri Bhat, Suma Bhat, Archna Bhatia, parminder bhatia, Arnab Bhattacharya, Pushpak Bhattacharyya, Sudha Bhingardive, Timothy Bickmore, Or Biran, Yonatan Bisk, Johannes Bjerva, Johanna Björklund, Philippe Blache, Alan W Black, Graeme Blackwood, Su Lin Blodgett, Michael Bloodgood, Théodore Bluche, Victoria Bobicev, Danushka Bollegala, Marcel Bollmann, Daniele Bonadiman, Georgeta Bordea, Gerlof Bouma, Samuel R. Bowman, S.R.K. Branavan, Chloé Braud, Felipe Bravo-Marquez, Chris Brockett, Julian Brooke, Thomas Brovelli (Meyer), Christopher Bryant, Paweł Budzianowski, Razvan Bunescu, Laura Burdick, Kaylee Burns, Jan Buys, Bill Byrne, José G. C. de Souza, Leticia Cagnina, Renqin Cai, Ruket Cakici, Iacer Calixto, Hiram Calvo, Nicoletta Calzolari, Erik Cambria, Burcu Can, Yixin Cao, Spencer Caplan, Annalina Caputo, Cornelia Caragea, Doina Caragea, Dallas Card, Marine Carpuat, Xavier Carreras, Lucien Carroll, Vittorio Castelli, Thiago Castro Ferreira, Asli Celikyilmaz, Fabio Celli, Daniel Cer, Dustin Chacón, Joyce Chai, Nathanael Chambers, Arjun Chandrasekaran, Muthu Kumar Chandrasekaran, Shiyu Chang, Wei-Lun Chao, Geeticka Chauhan, Wanxiang Che, Ciprian Chelba, Boxing Chen, Chen Chen, Danlu Chen, Danqi Chen, Kehai Chen, Lei Chen, Liwei Chen, Muhao Chen, Nancy Chen, Qian Chen, Tongfei Chen, Wenhu Chen, Xilun Chen, Xinchi Chen, Yue Chen, Yun-Nung Chen, Fei Cheng, Hao Cheng, Weiwei Cheng, Artem Chernodub, Ekaterina Chernyak, Colin Cherry, Emmanuele Chersoni, Jackie Chi Kit Cheung, Niyati Chhaya, David Chiang, Manoj Chinnakotla, Luis Chiruzzo, Eunsol Choi, Shamil Chollampatt, Leshem Choshen, Prafulla Kumar Choubey, Monojit Choudhury, Shammur Absar Chowdhury, Thomas Christie, Chenhui Chu, Kenneth Church, Philipp Cimiano, Volkan Cirik, Seamus Clancy, Elizabeth Clark, Stephen Clark, Ann Clifton, Martin Cmejrek, Maximin Coavoux, Arman Cohan, Trevor Cohen, Çağrı Çöltekin, Mathieu Constant, Danish Contractor, Paul Cook, João Cordeiro, Silvio Cordeiro, Caio Corro, Marta R. Costa-jussà, Benoit Crabbé, Josep Crego, Mathias Creutz, Paul Crook, Xiaodong Cui, Yiming Cui, Aron Culotta, Luis Fernando D'Haro, Rossana da Cunha Silva, Daniel Dahlmeier,

Zeyu Dai, Bhavana Dalvi, Amitava Das, Dipanjan Das, Vidas Daudaravicius, Elnaz Davoodi, Johannes Daxenberger, Adrià de Gispert, Éric de la Clergerie, Miryam de Lhoneux, Maarten de Rijke, Thierry Declerck, Luciano Del Corro, Louise Deléger, Vera Demberg, Thomas Demeester, Berkan Demirel, Lingjia Deng, Yuntian Deng, Nina Dethlefs, Daniel Deutsch, Barry Devereux, Kuntal Dey, Paramveer Dhillon, Maria Pia di Buono, Haibo Ding, Shuoyang Ding, Dmitriy Dligach, Li Dong, Doug Downey, Gabriel Doyle, Mark Dras, Markus Dreyer, Rotem Dror, Lan Du, Xinya Du, Nan Duan, Stefan Dück, Philipp Dufter, Kevin Duh, Ewan Dunbar, Jonathan Dunn, Nadir Durrani, Greg Durrett, Tomasz Dwojak, Melody Dye, Chris Dyer, Steffen Eger, Markus Egg, Vladimir Eidelman, Jacob Eisenstein, Layla El Asri, Heba Elfardy, Ahmed Elgohary, Michael Elhadad, Micha Elsner, Erkut Erdem, Akiko Eriguchi, Katrin Erk, Marcelo Errecalde, Hugo Jair Escalante, Maxine Eskenazi, Allyson Ettinger, Kilian Evang, Marzieh Fadaee, Cédrick Fairon, Agnieszka Falenska, Ingrid Falk, James Fan, Hui Fang, Meng Fang, Hossein Fani, Manaal Faruqui, Benoit Favre, Marcello Federico, Mariano Felice, Song Feng, Xiaocheng Feng, Elisabetta Fersini, Besnik Fetahu, Simone Filice, Katja Filippova, Tim Finin, Mark Finlayson, Gregory Finley, Orhan Firat, Nicholas FitzGerald, Eileen Fitzpatrick, Jeffrey Flanigan, Margaret Fleck, Lucie Flekova, Antske Fokkens, José A. R. Fonollosa, Tommaso Fornaciari, Karën Fort, Meaghan Fowlie, Kathleen C. Fraser, Markus Freitag, André Freitas, Lea Frermann, Annemarie Friedrich, Jason Fries, Lisheng Fu, Michel Galley, Michael Gamon, Kuzman Ganchev, Ashwinkumar Ganesan, Debasis Ganguly, Qin Gao, Wei Gao, Jorge Garcia Flores, Claire Gardent, Matt Gardner, Dan Garrette, Milica Gasic, Albert Gatt, Niyu Ge, Sebastian Gehrmann, Spandana Gella, Kallirroi Georgila, Reza Ghaeini, Shabnam Ghaffarzadegan, Marjan Ghazvininejad, Debanjan Ghosh, Kevin Gimpel, Filip Ginter, Rahul Goel, Darina Gold, Yoav Goldberg, Sujatha Gollapalli, Helena Gomez, Carlos Gómez-Rodríguez, Teresa Gonçalves, Fabio Gonzalez, Jonathan Gordon, Anuj Goyal, Pawan Goyal, Mario Graff, Yvette Graham, David Grangier, Christan Grant, Yulia Grishina, Alvin Grissom II, Kristina Gulordava, Jiang Guo, Weiwei Guo, Yufan Guo, Deepak Gupta, Vivek Gupta, E. Dario Gutierrez, Francisco Guzmán, Jeremy Gwinnup, Nizar Habash, Ivan Habernal, Gholamreza Haffari, Udo Hahn, John Hale, Oren Halvani, Michael Hammond, Lifeng Han, Na-Rae Han, Xu Han, Abram Handler, Thomas Hanneforth, Sanda Harabagiu, Tatsuya Harada, Randy Harris, Mohammed Hasanuzzaman, Chikara Hashimoto, Eva Hasler, Hany Hassan, Bradley Hauer, Claudia Hauff, Catherine Havasi, Serhii Havrylov, Katsuhiko Hayashi, Devamanyu Hazarika, Hangfeng He, He He, Ji He, Kun He, Ruidan He, Shilin He, Shizhu He, Yulan He, Kenneth Heafield, Peter Heeman, Jindřich Helcl, Christian Hempelmann, John Henderson, Sam Henry, Aurélie Herbelot, Delia Irazú Hernández Farías, Daniel Hershcovich, Jonathan Herzig, Derrick Higgins, Tsutomu Hirao, Graeme Hirst, Beth Ann Hockey, Eric Holgate, Juneki Hong, Enamul Hoque, Tobias Horsmann, Nabil Hossain, Yufang Hou, Estevam Hruschka, Baotian Hu, Hexiang Hu, Junjie Hu, Zhiting Hu, Xinyu Hua, Yiqing Hua, Binxuan Huang, Hen-Hsen Huang, Lifu Huang, Qiuyuan Huang, Xiaolei Huang, Xuanjing Huang, Zhongqiang Huang, Mans Hulden, Gonzalo Iglesias, Ryu Iida, Shajith Ikbal, Kenji Imamura, Oana Inel, Diana Inkpen, Dan Iter, Srinivasan Iyer, Mohit Iyyer, Cassandra L. Jacobs, Aaron Jaech, Sarthak Jain, Abhik Jana, Hyeju Jang, Adam Jardine, Adam Jatowt, Rahul Jha, Harsh Jhamtani, Yangfeng Ji, Robin Jia, Zhen Jia, Hui Jiang, Jyun-Yu Jiang, Yong Jiang, Zhanming Jie, Antonio Jimeno Yepes, Lifeng Jin, Yohan Jo, Anders Johannsen, Michael Johnston, Gareth Jones, Aditya Joshi, Vidur Joshi, Prathusha K Sarma, Jad Kabbara, Sushant Kafle, Nobuhiro Kaji, Tomoyuki Kajiwara, Jaap Kamps, Min-Yen Kan, Dongyeop Kang, Katharina Kann, Sarvnaz Karimi, Dimitri Kartsaklis, Ramakanth Kavuluru, Anna Kazantseva, Katherine Keith, Frank Keller, Yova Kementchedjhieva, Kian Kenyon-Dean, Nitish Shirish Keskar, Salam Khalifa, Maxim Khalilov, Daniel Khashabi, Huda Khayrallah, Douwe Kiela, Mert Kilickaya, Halil Kilicoglu, Been Kim, Gunhee Kim, Joo-Kyung Kim, Sun Kim, Yoon Kim, Young-Bum Kim, Yunsu Kim, David King, Svetlana Kiritchenko, Andreas Søeborg Kirkedal, Christo Kirov, Julia Kiseleva, Nikita Kitaev, Roman Klinger, Hayato Kobayashi, Sosuke Kobayashi, Gregory Kobele, Philipp Koehn, Arne Köhn, Tom Kollar, Mamoru Komachi, Kazunori Komatani, Rik Koncel-Kedziorski, Ravi Kondadadi, Stasinos Konstantopoulos, Ioannis Konstas, Parisa Kordjamshidi, Leila Kosseim, Brigitte Krenn, Ralf Krestel, Julia Kreutzer, Udo Kruschwitz, Da Kuang, Sandra Kübler, Marco Kuhlmann, Anjishnu Kumar, Sanjeev Kumar Karn, Anil Kumar Singh, Jonathan K. Kummerfeld, Anoop Kunchukuttan, Gourab Kundu, Mikko Kurimo, Shuhei Kurita, Sadao Kurohashi, Ilia Kuznetsov, Ophélie Lacroix, Faisal Ladhak, Surafel Melaku Lakew, Sobha Lalitha Devi, John P. Lalor, Mathias Lambert, Patrik Lambert, Vasileios Lampos, Gerasimos Lampouras, Ni Lao, Mirella Lapata, Mark Last, Alon Lavie, Carolin Lawrence, Joseph Le Roux, Chong Min Lee, Kathy Lee, Kenton Lee, Lung-Hao Lee, Moontae Lee, Artuur Leeuwenberg, Els Lefever, Alessandro Lenci, Chee Wee (Ben) Leong, Sarah Ita Levitan, Omer Levy, Mike Lewis, Chen Li, Chenliang Li, Chunyuan Li, Jian Li, Jing Li, Piji Li, Sheng Li, Sujian Li, Wei Li, Xiang Li, Xiujun Li, Yitong Li, Zhongliang Li, Paul Pu Liang, Jindřich Libovický, Anne-Laure Ligozat, Chu-Cheng Lin, Jimmy Lin, Xi Victoria Lin, Yankai Lin, Ying Lin, Zhouhan Lin, Tom Lippincott, Zachary Lipton, Pierre Lison, Bing Liu, Bing Liu, Chenxi Liu, Fei Liu, Hongfang Liu, Jiangming Liu, Jingjing Liu, Liyuan Liu, Mengwen Liu, Qun Liu, Shujie Liu, Tianyu Liu, Xueqing Liu, Yang Liu, Ye Liu, Yijia Liu, Zhengzhong Liu, Zhiyuan Liu, Nikola Ljubešić, Sharid Loáiciga, Adam Lopez, Adrián Pastor López Monroy, Juan Antonio Lossio-Ventura, Annie Louis, Pablo Loyola, Wei Lu, Yi Luan, Oswaldo Ludwig, Stephanie M. Lukin, Ruotian Luo, Anh Tuan Luu, Verena Lyding, Teresa Lynn, Danni Ma, Mingbo Ma, Tengfei Ma, Yukun Ma, Mounica Maddela, Pranava Madhyastha, Andrea Madotto, Debanjan Mahata, Darshini Mahendran, Jean Maillard, Anutosh Maitra, Navonil Majumder, Peter Makarov, Prodromos Malakasiotis, Chaitanya Malaviya, Andreas Maletti, Fragkiskos Malliaros, Rob Malouf, Suresh Manandhar, Michael Mandel, Lydia Manikonda, Gideon Mann, Ramesh Manuvinakurike, Diego Marcheggiani, Michał Marcińczuk, David Mareček, Marco Marelli, Benjamin Marie, Katja Markert, Ilia Markov, Edison Marrese-Taylor, Iain Marshall, Héctor Martínez Alonso, André F. T. Martins, David Martins de Matos, Yuji Matsumoto, Takuya Matsuzaki, Cynthia Matuszek, Jonathan May, Diana Maynard, Bryan McCann, Arya D. McCarthy, Diana McCarthy, John Philip McCrae, Bill McDowell, Bridget McInnes, Kathy McKeown, Oren Melamud, Arul Menezes, Rui Meng, Yuanliang Meng, Elizabeth Merkhofer, Christian M. Meyer, Julian Michael, Paul Michel, Lesly Miculicich, Sebastian J. Mielke, Margot Mieskes, Claudiu Mihăilă, Todor Mihaylov, Elena Mikhalkova, Simon Mille, Pasquale Minervini, Sabino Miranda-Jiménez, Paramita Mirza, Abhinav Misra, Amita Misra, Dipendra Misra, Jeff Mitchell, Margaret Mitchell, Bhaskar Mitra, Arpit Mittal, Makoto Miwa, Yusuke Miyao, Ashutosh Modi, Aditya Mogadala, Mahmoud Mohammadi, Mrinal Mohit, Diego Molla, Nicholas Monath, Will Monroe, Andres Montoyo, Seungwhan Moon, Nafise Sadat Moosavi, Mohamed Morchid, Isabel Moreno, Masud Moshtaghi, Nasrin Mostafazadeh, Diego Moussallem, Kadri Muischnek, Animesh Mukherjee, Hala Mulki, Philippe Muller, Smaranda Muresan, Kenton Murray, Rudra Murthy, Kaili Müürisep, Moin Nabi, Aakanksha Naik, Maryam Najafian, Mikio Nakano, Vinay Namboodiri, Courtney Napoles, Diane Napolitano, Jason Naradowsky, Shashi Narayan, Setareh Nasihati Gilani, Alexis Nasr, Tristan Naumann, Roberto Navigli, Guenter Neumann, Aurélie Névéol, Vincent Ng, Dat Quoc Nguyen, Dong Nguyen, Patrick Nguyen, Thien Huu Nguyen, Jianmo Ni, Zhaoheng Ni, Garrett Nicolai, Jian-Yun NIE, Thomas Niebler, Vassilina Nikoulina, Qiang Ning, Malvina Nissim, Xing Niu, Zheng-Yu Niu, Joakim Nivre, Jekaterina Novikova, Benjamin Nye, Diarmuid Ó Séaghdha, Alexander O'Connor, Brendan O'Connor, Daniela Alejandra Ochoa, Nir Ofek, Kemal Oflazer, Alice Oh, Jong-Hoon Oh, Emily Öhman, Kiyonori Ohtake, Tsuyoshi Okita, Manabu Okumura, Amy Olex, Shereen Oraby, Constantin Orasan, Petya Osenova, Mari Ostendorf, Naoki Otani, Arzucan Özgür, Deepak P, Sebastian Padó, Serguei Pakhomov, Shruti Palaskar, Martha Palmer, Girish Palshikar, Alexander Panchenko, Haris Papageorgiou, Emerson Paraiso, Pulkit Parikh, Ji Ho Park, Joonsuk Park, Yannick Parmentier, Carla Parra Escartín, Md Rizwan Parvez, Katerina Pastra, Roma Patel, Olga Patterson, Viviana Patti, Siddharth Patwardhan, Romain Paulus, Umashanthi Pavalanathan, Ellie Pavlick, Sachin Pawar, Haiyun Peng, Haoruo Peng, Jing Peng, Nanyun Peng, Gerald Penn, Laura Perez-Beltrachini, Matthew Peters, Karl Pichotta, Ildiko Pilan, Mohammad Taher Pilehvar, Tiago Pimentel, Mārcis Pinnis, Yuval Pinter, Florina Piroi, Emily

Pitler, Lidia Pivovarova, Barbara Plank, Lahari Poddar, Massimo Poesio, Adam Poliak, Simone Paolo Ponzetto, Hoifung Poon, Andrei Popescu-Belis, Maja Popović, Alexandros Potamianos, Peter Potash, Martin Potthast, Shrimai Prabhumoye, Sameer Pradhan, Judita Preiss, Daniel Preoţiuc-Pietro, Emily Prud'hommeaux, Danish Pruthi, Reid Pryzant, Adam Przepiórkowski, Ratish Puduppully, Robert Pugh, Rajkumar Pujari, Avinesh PVS, Ashequl Qadir, Peng Qi, Yao Qian, Yujie Qian, Likun Qiu, Xipeng Qiu, Lizhen Qu, Chris Quirk, Ella Rabinovich, Will Radford, Vipul Raheja, Afshin Rahimi, Dheeraj Rajagopal, Nazneen Fatema Rajani, Taraka Rama, Anil Ramakrishna, Gabriela Ramirez-de-la-Rosa, Nitin Ramrakhiyani, Francisco Manuel Rangel Pardo, Aarne Ranta, Sudha Rao, Farzana Rashid, Hannah Rashkin, Mohammad Sadegh Rasooli, Abhilasha Ravichander, Vinit Ravishankar, Julia Rayz, Simon Razniewski, Livy Real, Marta Recasens, Lena Reed, Georg Rehm, Ehud Reiter, Steffen Remus, Shuo Ren, Xiang Ren, Rezvaneh Rezapour, Frank Richter, Mark Riedl, Ruty Rinott, Alan Ritter, Kirk Roberts, Gil Rocha, Melissa Roemmele, Anna Rogers, Anna Rohrbach, Lina M. Rojas Barahona, Oleg Rokhlenko, Stephen Roller, Alexey Romanov, Salvatore Romeo, Marc-Antoine Rondeau, Amirmohammad Rooshenas, Rudolf Rosa, Andrew Rosenberg, Benjamin Roth, Michael Roth, Sascha Rothe, Masoud Rouhizadeh, Salim Roukos, Bryan Routledge, Dwaipayan Roy, Alla Rozovskaya, Andreas Rücklé, Sebastian Ruder, Rachel Rudinger, Frank Rudzicz, Attapol Rutherford, Derek Ruths, Neville Ryant, Devendra Sachan, Mrinmaya Sachan, Fatiha Sadat, Marzieh Saeidi, Saeid Safavi, Kenji Sagae, Horacio Saggion, Benoît Sagot, Rishiraj Saha Roy, Navanath Saharia, Sunil Kumar Sahu, Keisuke Sakaguchi, Iman Saleh, Bahar Salehi, Ramon Sanabria, Ivan Sanchez, Karthik Sankaranarayanan, Enrico Santus, Maarten Sap, Elvis Saravia, Ruhi Sarikaya, Himangshu Sarma, Hassan Sawaf, Carolina Scarton, Jonathan Schler, Helmut Schmid, Martin Schmitt, Sylvain Schmitz, Sabine Schulte im Walde, Anne-Kathrin Schumann, Lane Schwartz, Roy Schwartz, Holger Schwenk, Djamé Seddah, Satoshi Sekine, Shubhashis Sengupta, Rico Sennrich, Minjoon Seo, Fei Sha, Samira Shaikh, Jingbo Shang, Pamela Shapiro, Raksha Sharma, Shikhar Sharma, Dinghan Shen, Yilin Shen, Wei Shi, Xing Shi, Tomohide Shibata, Manish Shrivastava, Vered Shwartz, Gerardo Sierra Martínez, Miikka Silfverberg, Fabrizio Silvestri, Khalil Simaan, Patrick Simianer, Kiril Simov, Abhishek Singh, Gaurav Singh, Kairit Sirts, Gabriel Skantze, Kevin Small, Parinaz Sobhani, Linfeng Song, Sandeep Soni, Radu Soricut, Irena Spasic, Lucia Specia, Matthias Sperber, Damiano Spina, Caroline Sporleder, Vivek Srikumar, Felix Stahlberg, Sanja Štajner, Efstathios Stamatatos, Miloš Stanojević, Gabriel Stanovsky, Manfred Stede, Mark Steedman, Amanda Stent, Mark Stevenson, Svetlana Stoyanchev, Veselin Stoyanov, Carlo Strapparava, Amber Stubbs, Hang Su, Yu Su, Shivashankar Subramanian, David Suendermann-Oeft, Maria Sukhareva, Md Arafat Sultan, Huan Sun, Kai Sun, Weiwei Sun, Weiyi Sun, Yibo Sun, Mihai Surdeanu, Simon Suster, Jun Suzuki, Swabha Swayamdipta, György Szarvas, Stan Szpakowicz, Alon Talmor, Hao Tan, Duyu Tang, Jiliang Tang, Jiliang Tang, Yi Tay, Christoph Teichmann, Eric Tellez, Nedelina Teneva, Nithum Thain, Patricia Thaine, Mariët Theune, Jesse Thomason, Sam Thomson, James Thorne, Ran Tian, Jörg Tiedemann, Christoph Tillmann, Sara Tonelli, Mariya Toneva, Fatemeh Torabi Asr, Manabu Torii, Samia Touileb, Julien Tourille, Trang Tran, Diana Trandabat, Chen-Tse Tsai, Reut Tsarfaty, Michael Tschuggnall, Yoshimasa Tsuruoka, Kewei Tu, Lifu Tu, Gokhan Tur, Rutuja Ubale, Nicola Ueffing, Dmitry Ustalov, Ozlem Uzuner, Naushad UzZaman, Preethi Vaidyanathan, Svitlana Vakulenko, Andreas van Cranenburgh, Esther van den Berg, Rob van der Goot, Benjamin Van Durme, Emiel van Miltenburg, Menno van Zaanen, Clara Vania, Aparna Varde, Vasudeva Varma, Alakananda Vempala, Suzan Verberne, Patrick Verga, Manisha Verma, Rakesh Verma, David Vilar, David Vilares, Esau Villatoro-Tello, Aline Villavicencio, Veronika Vincze, Harit Vishwakarma, Andreas Vlachos, Rob Voigt, Nikos Voskarides, Ngoc Thang Vu, Thuy Vu, Ivan Vulić, Yogarshi Vyas, Ekaterina Vylomova, Xiaojun Wan, Cheng Wang, Di Wang, Dingquan Wang, Jinpeng Wang, Longyue Wang, Lu Wang, Pidong Wang, Qingyun Wang, Rui Wang, Shuai Wang, Shuohang Wang, Tianlu Wang, Tong Wang, Wei Wang, Wen Wang, Wenlin Wang, Wenya Wang, Xin Wang, Xing Wang, Yequan Wang, Yong Wang, Zhongqing Wang, Ziqi Wang, Leo Wanner, Taro Watanabe, Bonnie Webber, Furu Wei, Johannes Welbl, Michael White, John Wieting, Derry Tanti Wijaya, Graham Wilcock, Colin Wilson, Shomir Wilson, Steven Wilson, Shuly Wintner, Sam Wiseman, Guillaume Wisniewski, Michael Wojatzki, Marcin Woliński, Kam-Fai Wong, Alina Wróblewska, Chien-Sheng Wu, Fangzhao Wu, Jiawei Wu, Lingfei Wu, Qi Wu, Shijie Wu, Stephen Wu, Yuexin Wu, Joern Wuebker, Dirk Wulff, Rui Xia, Tong Xiao, Qizhe Xie, Ruobing Xie, Shasha Xie, Caiming Xiong, Wenhan Xiong, Chenliang Xu, Hu Xu, Huijuan Xu, Kun Xu, Ruochen Xu, Tan Xu, Shweta Yadav, Semih Yagcioglu, Yadollah Yaghoobzadeh, Mohamed Yahya, Rui Yan, Zhao Yan, Victoria Yaneva, Baosong Yang, Bishan Yang, Chao Yang, Jie Yang, Qian Yang, Weiwei Yang, Yi Yang, Zhaojun Yang, Zi Yang, Helen Yannakoudakis, Wenlin Yao, Mahsa Yarmohammadi, Andrew Yates, Ainur Yessenalina, Seid Muhie Yimam, Pengcheng Yin, Seunghyun Yoon, Naoki Yoshinaga, Jianfei Yu, Liang-Chih Yu, Tao Yu, Xingdi Yuan, Zheng Yuan, Hyokun Yun, Frances Yung, Zdeněk Žabokrtský, Hamed Zamani, Marcos Zampieri, Fabio Massimo Zanzotto, Guido Zarrella, Sina Zarrieß, Fattane Zarrinkalam, Rabih Zbib, Rowan Zellers, Luke Zettlemoyer, Boliang Zhang, Bowen Zhang, Jiajun Zhang, Meishan Zhang, Qi Zhang, Rui Zhang, Rui Zhang, Sheng Zhang, Shuo Zhang, Tongtao Zhang, Xingxing Zhang, Ye Zhang, Yi Zhang, Yizhe Zhang, Yu Zhang, Yuhao Zhang, Zhe Zhang, Zhirui Zhang, Jieyu Zhao, Tiancheng Zhao, Guoqing Zheng, Kai Zhong, Dong Zhou, Guangyou Zhou, Ming Zhou, Qingyu Zhou, Hao Zhu, Ayah Zirikly, Markus Zopf

Keynote Speaker: Rada Mihalcea, University of Michigan

When the Computers Spot the Lie (and People Don't)

Abstract: Whether we like it or not, deception occurs everyday and everywhere: thousands of trials take place daily around the world; little white lies: "I'm busy that day!" even if your calendar is blank; news "with a twist" (a.k.a. fake news) meant to attract the readers attention or influence people in their future undertakings; misinformation in health social media posts; portrayed identities, on dating sites and elsewhere. Can a computer automatically detect deception in written accounts or in video recordings? In this talk, I will overview a decade of research in building linguistic and multimodal resources and algorithms for deception detection, targeting deceptive statements, trial videos, fake news, identity deception, and health misinformation. I will also show how these algorithms can provide insights into what makes a good lie - and thus teach us how we can spot a liar. As it turns out, computers can be trained to identify lies in many different contexts, and they can often do it better than humans do.

Bio: Rada Mihalcea is a Professor of Computer Science and Engineering at the University of Michigan and the Director of the Michigan Artificial Intelligence Lab. Her research interests are in lexical semantics, multilingual NLP, and computational social sciences. She serves or has served on the editorial boards of the Journals of Computational Linguistics, Language Resources and Evaluations, Natural Language Engineering, Journal of Artificial Intelligence Research, IEEE Transactions on Affective Computing, and Transactions of the Association for Computational Linguistics. She was a program co-chair for EMNLP 2009 and ACL 2011, and a general chair for NAACL 2015 and *SEM 2019. She currently serves as the ACL Vice-President Elect. She is the recipient of an NSF CAREER award (2008) and a Presidential Early Career Award for Scientists and Engineers awarded by President Obama (2009). In 2013, she was made an honorary citizen of her hometown of Cluj-Napoca, Romania.

Keynote Speaker: Kieran Snyder, Textio

Leaving the Lab: Building NLP Applications that Real People can Use

Abstract: There is a chasm between an NLP technology that works well in the research lab and something that works for applications that real people use. Research conditions are often theoretical or idealized. The first time they contribute to industry projects, many theoretical researchers are surprised to discover how much goes into building outside the lab, and how hard it is to build data products for real people ethically and transparently. This talk explores my NLP journey in three stages: working as an academic NLP researcher, learning to be a practical creator of NLP products in industry, and becoming the founding CEO of an NLP business. While each role has used my background in computational linguistics in essential ways, every step has also required me to learn and unlearn new things along the way. The further I have gone in my industry career, the more critical it has become to define and work within a well-established set of principles for data ethics. This talk is for academic researchers considering industry careers or collaborations, for people in industry who started out in academia, and for anyone on either side of the divide who wants to make NLP products that real people can use.

Bio: Kieran Snyder is the CEO and Co-Founder of Textio, the augmented writing platform. For anything you write, Textio tells you ahead of time who's going to respond based on the language you've used. Textio's augmented writing engine is designed to attach to any large text corpus with outcomes to find the patterns that work. Prior to founding Textio, Kieran held product leadership roles at Microsoft and Amazon. Kieran has a PhD in linguistics from the University of Pennsylvania. Her work has appeared in Fortune, Re/code, Slate, and the Washington Post.

Keynote Speaker: Arvind Narayanan, Princeton

Data as a Mirror of Society: Lessons from the Emerging Science of Fairness in Machine Learning

Abstract: Language corpora reflect human society, including cultural stereotypes, prejudices, and historical patterns. By default, statistical language models will absorb these stereotypes. As a result, NLP systems for word analogy generation, toxicity detection, and many other tasks have been found to reflect racial and gender biases. Based on this observation, I will discuss two emerging research directions. First, a deeper understanding of human culture can help identify possible harmful stereotypes in algorithmic systems. The second research direction is the converse of the first: if data is a mirror of society, machine learning can be used as a magnifying lens to study human culture.

Bio: Arvind Narayanan is an Associate Professor of Computer Science at Princeton. His research has shown how state-of-the-art word embeddings reflect racial, gender, and other cultural stereotypes. He leads the Princeton Web Transparency and Accountability Project to uncover how companies collect and use our personal information. His doctoral research showed the fundamental limits of de-identification, for which he received the Privacy Enhancing Technologies Award. Narayanan also co-created a Massive Open Online Course as well as a textbook on Bitcoin and cryptocurrency technologies.

Table of Contents

Conference Program

Sunday, June 2, 2019

18:00–20:00 *Welcome Reception (Nicollet Grand Ballroom)*

Monday, June 3, 2019

9:00–9:30 *Land Acknowledgments, Opening Remarks and Janyce Wiebe and Richard Kittredge Remembrances* (Nicollet Grand Ballroom)

Session 1

9:30–10:30 *Keynote 1: Arvind Narayanan "Data as a Mirror of Society: Lessons from the Emerging Science of Fairness in Machine Learning"* (Nicollet Grand Ballroom)

10:30–11:00 *Coffee Break*

11:00–12:30 ***Oral Sessions (long papers) and Posters (long and short papers)***

Session 1A: Psycholinguistics & Cognitive Modelling

Room: Nicollet B+C, Chair: **Serguei Pakhomov**

11:00–11:18 *Entity Recognition at First Sight: Improving NER with Eye Movement Information*
Nora Hollenstein and Ce Zhang

11:18–11:36 *The emergence of number and syntax units in LSTM language models*
Yair Lakretz, Germán Kruszewski, Théo Desbordes, Dieuwke Hupkes, Stanislas Dehaene and Marco Baroni

11:36–11:54 *Neural Self-Training through Spaced Repetition*
Hadi Amiri

11:54–12:12 *Neural language models as psycholinguistic subjects: Representations of syntactic state*
Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros and Roger Levy

12:12–12:30 *Understanding language-elicited EEG data by predicting it from a fine-tuned language model*
Dan Schwartz and Tom Mitchell

Monday, June 3, 2019 (continued)

Session 1B: Speech
Room: Nicollet A, Chair: **Yang Liu**

11:00–11:18 *Pre-training on high-resource speech recognition improves low-resource speech-to-text translation*
Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez and Sharon Goldwater

11:18–11:36 *Measuring the perceptual availability of phonological features during language acquisition using unsupervised binary stochastic autoencoders*
Cory Shain and Micha Elsner

11:36–11:54 *Giving Attention to the Unexpected: Using Prosody Innovations in Disfluency Detection*
Vicky Zayats and Mari Ostendorf

11:54–12:12 *Massively Multilingual Adversarial Speech Recognition*
Oliver Adams, Matthew Wiesner, Shinji Watanabe and David Yarowsky

12:12–12:30 *Lost in Interpretation: Predicting Untranslated Terminology in Simultaneous Interpretation*
Nikolai Vogler, Craig Stewart and Graham Neubig

Session 1C: Generation
Room: Northstar Ballroom A, Chair: **Wei Xu**

11:00–11:18 *AudioCaps: Generating Captions for Audios in The Wild*
Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee and Gunhee Kim

11:18–11:36 *"President Vows to Cut <Taxes> Hair": Dataset and Analysis of Creative Text Editing for Humorous Headlines*
Nabil Hossain, John Krumm and Michael Gamon

11:36–11:54 *Answer-based Adversarial Training for Generating Clarification Questions*
Sudha Rao and Hal Daumé III

11:54–12:12 *Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data*
Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia and Jingming Liu

12:12–12:30 *Topic-Guided Variational Auto-Encoder for Text Generation*
Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen and Lawrence Carin

Session 1D: Tagging, Chunking, Syntax & Parsing
Room: Greenway, Chair: **Roi Reichart**

11:00–11:18 *Implementation of a Chomsky-Schützenberger n-best parser for weighted multiple context-free grammars*
Thomas Ruprecht and Tobias Denkinger

11:18–11:36 *Phylogenic Multi-Lingual Dependency Parsing*
Mathieu Dehouck and Pascal Denis

11:36–11:54 *Discontinuous Constituency Parsing with a Stack-Free Transition System and a Dynamic Oracle*
Maximin Coavoux and Shay B. Cohen

11:54–12:12 *How Bad are PoS Tagger in Cross-Corpora Settings? Evaluating Annotation Divergence in the UD Project.*
Guillaume Wisniewski and François Yvon

12:12–12:30 *CCG Parsing Algorithm with Incremental Tree Rotation*
Miloš Stanojević and Mark Steedman

Session 1E: Theory & Formalisms
Room: Nicollet D, Chair: **Ryan Cotterell**

11:00–11:18 *Cyclical Annealing Schedule: A Simple Approach to Mitigate KL Vanishing*
Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz and Lawrence Carin

11:18–11:36 *Recurrent models and lower bounds for projective syntactic decoding*
Natalie Schluter

11:36–11:54 *Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings*
Gijs Wijnholds and Mehrnoosh Sadrzadeh

11:54–12:12 *Neural Finite-State Transducers: Beyond Rational Relations*
Chu-Cheng Lin, Hao Zhu, Matthew R. Gormley and Jason Eisner

12:12–12:30 *Riemannian Normalizing Flow on Variational Wasserstein Autoencoder for Text Modeling*
Prince Zizhuang Wang and William Yang Wang

Session 1F: Question Answering, Sentiment, Machine Translation, Resources & Evaluation (Posters)

Room: Hyatt Exhibit Hall

Question Answering

A Study of Incorrect Paraphrases in Crowdsourced User Utterances
Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah, Moshe Chai Barukh and Shayan Zamanirad

ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters
Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya and Gerhard Weikum

FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase
Kelvin Jiang, Dekun Wu and Hui Jiang

Simple Question Answering with Subgraph Ranking and Joint-Scoring
Wenbo Zhao, Tagyoung Chung, Anuj Goyal and Angeliki Metallinou

Learning to Attend On Essential Terms: An Enhanced Retriever-Reader Model for Open-domain Question Answering
Jianmo Ni, Chenguang Zhu, Weizhu Chen and Julian McAuley

UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering
Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak and Lun-Wei Ku

BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering
Yu Cao, Meng Fang and Dacheng Tao

[SRW] *Is It Dish Washer Safe? Automatically Answering "Yes-No" Questions Using Customer Reviews*
Daria Dzendzik, Carl Vogel and Jennifer Foster

Sentiment

Vector of Locally-Aggregated Word Embeddings (VLAWE): A Novel Document-level Representation
Radu Tudor Ionescu and Andrei Butnaru

Multi-task Learning for Multi-modal Emotion Recognition and Sentiment Analysis
Md Shad Akhtar, Dushyant Chauhan, Deepanway Ghosal, Soujanya Poria, Asif Ekbal and Pushpak Bhattacharyya

Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence
Chi Sun, Luyao Huang and Xipeng Qiu

A Variational Approach to Weakly Supervised Document-Level Multi-Aspect Sentiment Classification
Ziqian Zeng, Wenxuan Zhou, Xin Liu and Yangqiu Song

HiGRU: Hierarchical Gated Recurrent Units for Utterance-Level Emotion Recognition
Wenxiang Jiao, Haiqin Yang, Irwin King and Michael R. Lyu

Learning Interpretable Negation Rules via Weak Supervision at Document Level: A Reinforcement Learning Approach
Nicolas Pröllochs, Stefan Feuerriegel and Dirk Neumann

Simplified Neural Unsupervised Domain Adaptation
Timothy Miller

Learning Bilingual Sentiment-Specific Word Embeddings without Cross-lingual Supervision
Yanlin Feng and Xiaojun Wan

Machine Translation
ReWE: Regressing Word Embeddings for Regularization of Neural Machine Translation Systems
Inigo Jauregi Unanue, Ehsan Zare Borzeshi, Nazanin Esmaili and Massimo Piccardi

Lost in Machine Translation: A Method to Reduce Meaning Loss
Reuben Cohn-Gordon and Noah Goodman

Bi-Directional Differentiable Input Reconstruction for Low-Resource Neural Machine Translation
Xing Niu, Weijia Xu and Marine Carpuat

Code-Switching for Enhancing NMT with Pre-Specified Translation
Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang and Min Zhang

Aligning Vector-spaces with Noisy Supervised Lexicon
Noa Yehezkel Lubin, Jacob Goldberger and Yoav Goldberg

Understanding and Improving Hidden Representations for Neural Machine Translation
Guanlin Li, Lemao Liu, Xintong Li, Conghui Zhu, Tiejun Zhao and Shuming Shi

Monday, June 3, 2019 (continued)

Resources & Evaluation

Content Differences in Syntactic and Semantic Representation
Daniel Hershcovich, Omri Abend and Ari Rappoport

Attentive Mimicking: Better Word Embeddings by Attending to Informative Contexts
Timo Schick and Hinrich Schütze

Evaluating Style Transfer for Text
Remi Mir, Bjarke Felbo, Nick Obradovich and Iyad Rahwan

Big BiRD: A Large, Fine-Grained, Bigram Relatedness Dataset for Examining Semantic Composition
Shima Asaadi, Saif Mohammad and Svetlana Kiritchenko

Outlier Detection for Improved Data Quality and Diversity in Dialog Systems
Stefan Larson, Anish Mahendran, Andrew Lee, Jonathan K. Kummerfeld, Parker Hill, Michael A. Laurenzano, Johann Hauswald, Lingjia Tang and Jason Mars

Asking the Right Question: Inferring Advice-Seeking Intentions from Personal Narratives
Liye Fu, Jonathan P. Chang and Cristian Danescu-Niculescu-Mizil

Seeing Things from a Different Angle:Discovering Diverse Perspectives about Claims
Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch and Dan Roth

12:30–13:00 *Grab your lunch break*

13:00–14:30 *Careers in NLP Panel (Nicollet Grand Ballroom)*

14:30–15:00 *Coffee Break*

15:00–16:30 ***Oral sessions (short papers), Posters (long and short papers) & Demos***

Session 2A: Dialogue & Discourse
Room: Northstar A, Chair: **Ellen Riloff**

15:00–15:15 *IMHO Fine-Tuning Improves Claim Detection*
Tuhin Chakrabarty, Christopher Hidey and Kathy McKeown

15:15–15:30 *Joint Multiple Intent Detection and Slot Labeling for Goal-Oriented Dialog*
Rashmi Gangadharaiah and Balakrishnan Narayanaswamy

15:30–15:45 *CITE: A Corpus of Image-Text Discourse Relations*
Malihe Alikhani, Sreyasi Nag Chowdhury, Gerard de Melo and Matthew Stone

15:45–16:00 *Improving Dialogue State Tracking by Discerning the Relevant Context*
Sanuj Sharma, Prafulla Kumar Choubey and Ruihong Huang

16:00–16:15 *CLEVR-Dialog: A Diagnostic Dataset for Multi-Round Reasoning in Visual Dialog*
Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra and Marcus Rohrbach

16:15–16:30 *Learning Outside the Box: Discourse-level Features Improve Metaphor Identification*
Jesse Mu, Helen Yannakoudakis and Ekaterina Shutova

Session 2B: Ethics, Bias & Fairness

Room: Nicollet B+C, Chair: **Preslav Nakov**

15:00–15:15 *Detection of Abusive Language: the Problem of Biased Datasets*
Michael Wiegand, Josef Ruppenhofer and Thomas Kleinbauer

15:15–15:30 *Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them*
Hila Gonen and Yoav Goldberg

15:30–15:45 *Black is to Criminal as Caucasian is to Police: Detecting and Removing Multiclass Bias in Word Embeddings*
Thomas Manzini, Lim Yao Chong, Alan W. Black and Yulia Tsvetkov

15:45–16:00 *On Measuring Social Biases in Sentence Encoders*
Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman and Rachel Rudinger

16:00–16:15 *Gender Bias in Contextualized Word Embeddings*
Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez and Kai-Wei Chang

16:15–16:30 [SRW] *Identifying and Reducing Gender Bias in Word-Level Language Models Status*
Shikha Bordia and Samuel R. Bowman

Session 2C: Style & Sentiment
Room: Nicollet D, Chair: **Diyi Yang**

15:00–15:15 *Combining Sentiment Lexica with a Multi-View Variational Autoencoder*
Alexander Miserlis Hoyle, Lawrence Wolf-Sonkin, Hanna Wallach, Ryan Cotterell and Isabelle Augenstein

15:15–15:30 *Enhancing Opinion Role Labeling with Semantic-Aware Word Representations from Semantic Role Labeling*
Meishan Zhang, Peili Liang and Guohong Fu

15:30–15:45 *Frowning Frodo, Wincing Leia, and a Seriously Great Friendship: Learning to Classify Emotional Relationships of Fictional Characters*
Evgeny Kim and Roman Klinger

15:45–16:00 *Generalizing Unmasking for Short Texts*
Janek Bevendorff, Benno Stein, Matthias Hagen and Martin Potthast

16:00–16:15 *Adversarial Training for Satire Detection: Controlling for Confounding Variables*
Robert McHardy, Heike Adel and Roman Klinger

16:15–16:30 [SRW] *Emotion impacts Speech Recognition Performance*
Rushab Munot and Ani Nenkova

Session 2D: Summarization & Information Retrieval

Room: Nicollet A, Chair: **Michael J. Paul**

15:00–15:15 *Keyphrase Generation: A Text Summarization Struggle*
Erion Çano and Ondřej Bojar

15:15–15:30 *SEQˆ3: Differentiable Sequence-to-Sequence-to-Sequence Autoencoder for Unsupervised Abstractive Sentence Compression*
Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas and Alexandros Potamianos

15:30–15:45 *Crowdsourcing Lightweight Pyramids for Manual Summary Evaluation*
Ori Shapira, David Gabay, Yang Gao, Hadar Ronen, Ramakanth Pasunuru, Mohit Bansal, Yael Amsterdamer and Ido Dagan

15:45–16:00 *Serial Recall Effects in Neural Language Modeling*
Hassan Hajipoor, Hadi Amiri, Maseud Rahgozar and Farhad Oroumchian

16:00–16:15 *Fast Concept Mention Grouping for Concept Map-based Multi-Document Summarization*
Tobias Falke and Iryna Gurevych

16:15–16:30 [SRW] *The Strength of the Weakest Supervision: Topic Classification Using Class Labels*
Jiatong Li, Kai Zheng, Hua Xu, Qiaozhu Mei and Yue Wang

Session 2E: Tagging, Chunking, Syntax & Parsing

Room: Greenway, Chairs: **Joel Tetreault**

15:00–15:15 *Syntax-aware Neural Semantic Role Labeling with Supertags*
Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev and Owen Rambow

15:15–15:30 *Left-to-Right Dependency Parsing with Pointer Networks*
Daniel Fernández-González and Carlos Gómez-Rodríguez

15:30–15:45 *Viable Dependency Parsing as Sequence Labeling*
Michalina Strzyz, David Vilares and Carlos Gómez-Rodríguez

15:45–16:00 *Pooled Contextualized Embeddings for Named Entity Recognition*
Alan Akbik, Tanja Bergmann and Roland Vollgraf

16:00–16:15 *Better Modeling of Incomplete Annotations for Named Entity Recognition*
Zhanming Jie, Pengjun Xie, Wei Lu, Ruixue Ding and Linlin Li

16:16–16:30 [SRW] *Handling Noisy Labels for Robustly Learning from Self-Training Data for Low-Resource Sequence Labeling*
Debjit Paul, Mittul Singh, Michael A. Hedderich and Dietrich Klakow

Session 2F: Information Extraction, Generation & Semantics (Posters & Demos)
Room: Hyatt Exhibit Hall

Information Extraction
Event Detection without Triggers
Shulin Liu, Yang Li, Feng Zhang, Tao Yang and Xinpeng Zhou

Sub-event detection from twitter streams as a sequence labeling problem
Giannis Bekoulis, Johannes Deleu, Thomas Demeester and Chris Develder

GraphIE: A Graph-Based Framework for Information Extraction
Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo and Regina Barzilay

OpenKI: Integrating Open Information Extraction and Knowledge Bases with Relation Inference
Dongxu Zhang, Subhabrata Mukherjee, Colin Lockard, Luna Dong and Andrew McCallum

Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing
Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo and William Yang Wang

Improving Event Coreference Resolution by Learning Argument Compatibility from Unlabeled Data
Yin Jou Huang, Jing Lu, Sadao Kurohashi and Vincent Ng

Sentence Embedding Alignment for Lifelong Relation Extraction
Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang and William Yang Wang

Description-Based Zero-shot Fine-Grained Entity Typing
Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi and Prasad Tadepalli

[SRW] *Opinion Mining with Deep Contextualized Embeddings*
Wen-Bin Han and Noriko Kando

[SRW] *A Bag-of-concepts Model Improves Relation Extraction in a Narrow Knowledge Domain with Limited Data*
Jiyu Chen, Karin Verspoor and Zenan Zhai

Generation
Adversarial Decomposition of Text Representation
Alexey Romanov, Anna Rumshisky, Anna Rogers and David Donahue

PoMo: Generating Entity-Specific Post-Modifiers in Context
Jun Seok Kang, Robert Logan, Zewei Chu, Yang Chen, Dheeru Dua, Kevin Gimpel, Sameer Singh and Niranjan Balasubramanian

Improved Lexically Constrained Decoding for Translation and Monolingual Rewriting
J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post and Benjamin Van Durme

Courteously Yours: Inducing courteous behavior in Customer Care responses using Reinforced Pointer Generator Network
Hitesh Golchha, Mauajama Firdaus, Asif Ekbal and Pushpak Bhattacharyya

How to Avoid Sentences Spelling Boring? Towards a Neural Approach to Unsupervised Metaphor Generation
Zhiwei Yu and Xiaojun Wan

[SRW] *Generating Text through Adversarial Training Using Skip-Thought Vectors*
Afroz Ahamad

[SRW] *A Partially Rule-Based Approach to AMR Generation*
Emma Manning

Semantics

Incorporating Context and External Knowledge for Pronoun Coreference Resolution
Hongming Zhang, Yan Song and Yangqiu Song

Unsupervised Deep Structured Semantic Models for Commonsense Reasoning
Shuohang Wang, Sheng Zhang, Yelong Shen, Xiaodong Liu, Jingjing Liu, Jianfeng Gao and Jing Jiang

Recovering dropped pronouns in Chinese conversations via modeling their referents
Jingxuan Yang, Jianzhuo Tong, Si Li, Sheng Gao, Jun Guo and Nianwen Xue

The problem with probabilistic DAG automata for semantic graphs
Ieva Vasiljeva, Sorcha Gilroy and Adam Lopez

A Systematic Study of Leveraging Subword Information for Learning Word Representations
Yi Zhu, Ivan Vulić and Anna Korhonen

Better Word Embeddings by Disentangling Contextual n-Gram Information
Prakhar Gupta, Matteo Pagliardini and Martin Jaggi

Integration of Knowledge Graph Embedding Into Topic Modeling with Hierarchical Dirichlet Process
Dingcheng Li, Siamak Zamani, Jingyuan Zhang and Ping Li

Correlation Coefficients and Semantic Textual Similarity
Vitalii Zhelezniak, Aleksandar Savkov, April Shen and Nils Hammerla

Generating Token-Level Explanations for Natural Language Inference
James Thorne, Andreas Vlachos, Christos Christodoulopoulos and Arpit Mittal

Strong Baselines for Complex Word Identification across Multiple Languages
Pierre Finnimore, Elisabeth Fritzsch, Daniel King, Alison Sneyd, Aneeq Ur Rehman, Fernando Alva-Manchego and Andreas Vlachos

[SRW] *Computational Investigations of Pragmatic Effects in Natural Language*
Jad Kabbara

16:30–17:00 *Coffee Break*

17:00–18:30 ***Oral sessions (long papers) and Posters (long and short papers)***

Session 3A: Information Extraction & Retrieval
Room: Nicollet A, Chair: **Gerard de Melo**

17:00–17:18 *Adaptive Convolution for Multi-Relational Learning*
Xiaotian Jiang, Quan Wang and Bin Wang

17:18–17:36 *Graph Pattern Entity Ranking Model for Knowledge Graph Completion*
Takuma Ebisu and Ryutaro Ichise

17:36–17:54 *Adversarial Training for Weakly Supervised Event Detection*
Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun and Peng Li

17:54–18:12 *A Submodular Feature-Aware Framework for Label Subset Selection in Extreme Classification Problems*
Elham J. Barezi, Ian D. Wood, Pascale Fung and Hamid R. Rabiee

18:12–18:30 *Relation Extraction with Temporal Reasoning Based on Memory Augmented Distant Supervision*
Jianhao Yan, Lin He, Ruqin Huang, Jian Li and Ying Liu

Session 3B: Semantics
Room: Nicollet D, Chair: **Kevin Gimpel**

17:00–17:18 *Integrating Semantic Knowledge to Tackle Zero-shot Text Classification*
Jingqing Zhang, Piyawat Lertvittayakumjorn and Yike Guo

17:18–17:36 *WordNode2Vec: Graph-based Embedding of Words*
Procheta Sen, Debasis Ganguly and Gareth Jones

17:36–17:54 *Cross-topic distributional semantic representations via unsupervised mappings*
Eleftheria Briakou, Nikos Athanasiou and Alexandros Potamianos

17:54–18:12 *What just happened? Evaluating retrofitted distributional word vectors*
Dmetri Hayes

18:12–18:30 *Linguistic Knowledge and Transferability of Contextual Representations*
Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters and Noah A. Smith

Session 3C: Parsing & Modelling
Room: Greenway, Chair: **Kai-Wei Chang**

17:00–17:18 *Mutual Information Maximization for Simple and Accurate Part-Of-Speech Induction*
Karl Stratos

17:18–17:36 *Unsupervised Recurrent Neural Network Grammars*
Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer and Gábor Melis

17:36–17:54 *Cooperative Learning of Disjoint Syntax and Semantics*
Serhii Havrylov, Germán Kruszewski and Armand Joulin

17:54–18:12 *Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Auto-Encoders*
Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer and Andrew McCallum

18:12–18:30 *Knowledge-Augmented Language Model and Its Application to Unsupervised Named-Entity Recognition*
Angli Liu, Jingfei Du and Veselin Stoyanov

Session 3D: Machine Translation
Room: Nicollet B+C, Chair: **Marine Carpuat**

17:00–17:18 *Syntax-Enhanced Neural Machine Translation with Syntax-Aware Word Representations*
Meishan Zhang, Zhenghua Li, Guohong Fu and Min Zhang

17:18–17:36 *Competence-based Curriculum Learning for Neural Machine Translation*
Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos and Tom Mitchell

17:36–17:54 *Extract and Edit: An Alternative to Back-Translation for Unsupervised Neural Machine Translation*
Jiawei Wu, Xin Wang and William Yang Wang

17:54–18:12 *Consistency by Agreement in Zero-Shot Neural Machine Translation*
Maruan Al-Shedivat and Ankur Parikh

18:12–18:30 *Modeling Recurrence for Transformer*
Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang and Zhaopeng Tu

Session 3E: Dialogue
Room: Northstar, Chair: **Sujith Ravi**

17:00–17:18 *Rethinking Action Spaces for Reinforcement Learning in End-to-end Dialog Agents with Latent Variable Models*
Tiancheng Zhao, Kaige Xie and Maxine Eskenazi

17:18–17:36 *Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory*
Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam and Shuming Shi

17:36–17:54 *Jointly Optimizing Diversity and Relevance in Neural Response Generation*
Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao and Bill Dolan

17:54–18:12 *Disentangling Language and Knowledge in Task-Oriented Dialogs*
Dinesh Raghu, Nikhil Gupta and Mausam

18:12–18:30 [TACL] *DREAM: A Challenge Dataset and Models for Dialogue-Based Reading Comprehension*
Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi and Claire Cardie

Session 3F: Applications, Social Media, Biomedical NLP & Clinical Text Processing (Posters)
NLP Applications

Tensorized Self-Attention: Efficiently Modeling Pairwise and Global Dependencies Together
Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang and Chengqi Zhang

WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations
Mohammad Taher Pilehvar and Jose Camacho-Collados

Does My Rebuttal Matter? Insights from a Major NLP Conference
Yang Gao, Steffen Eger, Ilia Kuznetsov, Iryna Gurevych and Yusuke Miyao

Casting Light on Invisible Cities: Computationally Engaging with Literary Criticism
Shufan Wang and Mohit Iyyer

PAWS: Paraphrase Adversaries from Word Scrambling
Yuan Zhang, Jason Baldridge and Luheng He

Cross-Corpora Evaluation and Analysis of Grammatical Error Correction Models — Is Single-Corpus Evaluation Enough?
Masato Mita, Tomoya Mizumoto, Masahiro Kaneko, Ryo Nagata and Kentaro Inui

Star-Transformer
Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue and Zheng Zhang

[SRW] *SEDTWik: Segmentation-based Event Detection from Tweets Using Wikipedia*
Keval Morabia, Neti Lalita Bhanu Murthy, Aruna Malapati and Surender Samant

Social Media
Adaptation of Hierarchical Structured Models for Speech Act Recognition in Asynchronous Conversation
Tasnim Mohiuddin, Thanh-Tung Nguyen and Shafiq Joty

From legal to technical concept: Towards an automated classification of German political Twitter postings as criminal offenses
Frederike Zufall, Tobias Horsmann and Torsten Zesch

Joint Multi-Label Attention Networks for Social Text Annotation
Hang Dong, Wei Wang, Kaizhu Huang and Frans Coenen

Multi-Channel Convolutional Neural Network for Twitter Emotion and Sentiment Recognition
Jumayel Islam, Robert E. Mercer and Lu Xiao

Detecting Cybersecurity Events from Noisy Short Text
Semih Yagcioglu, Mehmet saygin Seyfioglu, Begum Citamak, Batuhan Bardak, Seren Guldamlasioglu, Azmi Yuksel and Emin Islam Tatli

White-to-Black: Efficient Distillation of Black-Box Adversarial Attacks
Yotam Gil, Yoav Chai, Or Gorodissky and Jonathan Berant

Analyzing the Perceived Severity of Cybersecurity Threats Reported on Social Media
Shi Zong, Alan Ritter, Graham Mueller and Evan Wright

Fake News Detection using Deep Markov Random Fields
Duc Minh Nguyen, Tien Huu Do, Robert Calderbank and Nikos Deligiannis

Issue Framing in Online Discussion Fora
Mareike Hartmann, Tallulah Jansen, Isabelle Augenstein and Anders Søgaard

Vector of Locally Aggregated Embeddings for Text Representation
Hadi Amiri and Mitra Mohtarami

Predicting the Type and Target of Offensive Posts in Social Media
Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra and Ritesh Kumar

Biomedical NLP & Clinical Text Processing
Biomedical Event Extraction based on Knowledge-driven Tree-LSTM
Diya Li, Lifu Huang, Heng Ji and Jiawei Han

Detecting cognitive impairments by agreeing on interpretations of linguistic features
Zining Zhu, Jekaterina Novikova and Frank Rudzicz

Relation Extraction using Explicit Context Conditioning
Gaurav Singh and Parminder Bhatia

Conversation Model Fine-Tuning for Classifying Client Utterances in Counseling Dialogues
Sungjoon Park, Donghyun Kim and Alice Oh

Using Similarity Measures to Select Pretraining Data for NER
Xiang Dai, Sarvnaz Karimi, Ben Hachey and Cecile Paris

Predicting Annotation Difficulty to Improve Task Routing and Model Performance for Biomedical Information Extraction
Yinfei Yang, Oshin Agarwal, Chris Tar, Byron C. Wallace and Ani Nenkova

Detecting Depression in Social Media using Fine-Grained Emotions
Mario Ezra Aragon, Adrian Pastor Lopez Monroy, Luis Carlos Gonzalez Gurrola and Manuel Montes-y-Gomez

Tuesday, June 4, 2019

[SRW] *Kickstarting NLP for the Whole-person Function Domain with Representation Learning and Data Analysis*
Denis Newman-Griffis

A Silver Standard Corpus of Human Phenotype-Gene Relations
Diana Sousa, Andre Lamurias and Francisco M Couto

9:00–10:30 ***Oral sessions (long papers) and Posters (long and short papers)***

Session 4A: Phonology & Morphology
Room: Nicollet A, Chair: **Greg Kondrak**

9:00–9:18 *Improving Lemmatization of Non-Standard Languages with Joint Learning*
Enrique Manjavacas, Ákos Kádár and Mike Kestemont

9:18–9:36 *One Size Does Not Fit All: Comparing NMT Representations of Different Granularities*
Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov and Preslav Nakov

9:36–9:54 *A Simple Joint Model for Improved Contextual Neural Lemmatization*
Chaitanya Malaviya, Shijie Wu and Ryan Cotterell

9:54–10:12 *A Probabilistic Generative Model of Linguistic Typology*
Johannes Bjerva, Yova Kementchedjhieva, Ryan Cotterell and Isabelle Augenstein

10:12–10:30 *Quantifying the morphosyntactic content of Brown Clusters*
Manuel Ciosici, Leon Derczynski and Ira Assent

Session 4B: Multilingual NLP
Room: Nicollet D, Chair: **Ekaterina Shutova**

9:00–9:18 *Analyzing Bayesian Crosslingual Transfer in Topic Models*
Shudong Hao and Michael J. Paul

9:18–9:36 *Recursive Subtree Composition in LSTM-Based Dependency Parsing*
Miryam de Lhoneux, Miguel Ballesteros and Joakim Nivre

9:36–9:54 *Cross-lingual CCG Induction*
Kilian Evang

9:54–10:12 *Density Matching for Bilingual Word Embedding*
Chunting Zhou, Xuezhe Ma, Di Wang and Graham Neubig

10:12–10:30 *Cross-Lingual Alignment of Contextual Word Embeddings, with Applications to Zero-shot Dependency Parsing*
Tal Schuster, Ori Ram, Regina Barzilay and Amir Globerson

Tuesday, June 4, 2019 (continued)

Session 4C: Social Media
Room: Nicollet B+C, Chair: **Xiaodan Zhu**

9:00–9:18 *Early Rumour Detection*
Kaimin Zhou, Chang Shu, Binyang Li and Jey Han Lau

9:18–9:36 *Microblog Hashtag Generation via Encoding Conversation Contexts*
Yue Wang, Jing Li, Irwin King, Michael R. Lyu and Shuming Shi

9:36–9:54 *Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems*
Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson and Iryna Gurevych

9:54–10:12 *Something's Brewing! Early Prediction of Controversy-causing Posts from Discussion Features*
Jack Hessel and Lillian Lee

10:12–10:30 *No Permanent Friends or Enemies: Tracking Relationships between Nations from News*
Xiaochuang Han, Eunsol Choi and Chenhao Tan

Session 4D: Generation
Room: Northstar A, Chair: Ion Androutsopoulos

9:00–9:18 *Improving Human Text Comprehension through Semi-Markov CRF-based Neural Section Title Generation*
Sebastian Gehrmann, Steven Layne and Franck Dernoncourt

9:18–9:36 *Unifying Human and Statistical Evaluation for Natural Language Generation*
Tatsunori Hashimoto, Hugh Zhang and Percy Liang

9:36–9:54 *What makes a good conversation? How controllable attributes affect human judgments*
Abigail See, Stephen Roller, Douwe Kiela and Jason Weston

9:54–10:12 *An Empirical Investigation of Global and Local Normalization for Recurrent Neural Sequence Models Using a Continuous Relaxation to Beam Search*
Kartik Goyal, Chris Dyer and Taylor Berg-Kirkpatrick

10:12–10:30 *Pun Generation with Surprise*
He He, Nanyun Peng and Percy Liang

Session 4E: Industry Session: Real World Challenges
Room: Greenway

Session 4F: Discourse, Information Retrieval, Machine Translation, Vision & Robotics (Posters)

Discourse
Single Document Summarization as Tree Induction
Yang Liu, Ivan Titov and Mirella Lapata

Fixed That for You: Generating Contrastive Claims with Semantic Edits
Christopher Hidey and Kathy McKeown

Box of Lies: Multimodal Deception Detection in Dialogues
Felix Soldner, Verónica Pérez-Rosas and Rada Mihalcea

A Crowdsourced Corpus of Multiple Judgments and Disagreement on Anaphoric Interpretation
Massimo Poesio, Jon Chamberlain, Silviu Paun, Juntao Yu, Alexandra Uma and Udo Kruschwitz

A Streamlined Method for Sourcing Discourse-level Argumentation Annotations from the Crowd
Tristan Miller, Maria Sukhareva and Iryna Gurevych

Unsupervised Dialog Structure Learning
Weiyan Shi, Tiancheng Zhao and Zhou Yu

Modeling Document-level Causal Structures for Event Causal Relation Identification
Lei Gao, Prafulla Kumar Choubey and Ruihong Huang

[TACL] *Planning, Inference, and Pragmatics in Sequential Language Games*
Fereshte Khani, Noah Goodman and Percy Liang

Information Retrieval
Hierarchical User and Item Representation with Three-Tier Attention for Recommendation
Chuhan Wu, Fangzhao Wu, Junxin Liu and Yongfeng Huang

Text Similarity Estimation Based on Word Embeddings and Matrix Norms for Targeted Marketing
Tim vor der Brück and Marc Pouly

Glocal: Incorporating Global Information in Local Convolution for Keyphrase Extraction
Animesh Prasad and Min-Yen Kan

A Study of Latent Structured Prediction Approaches to Passage Reranking
Iryna Haponchyk and Alessandro Moschitti

Combining Distant and Direct Supervision for Neural Relation Extraction
Iz Beltagy, Kyle Lo and Waleed Ammar

Tweet Stance Detection Using an Attention based Neural Ensemble Model
Umme Aymun Siddiqua, Abu Nowshed Chy and Masaki Aono

Machine Translation
Word Embedding-Based Automatic MT Evaluation Metric using Word Position Information
Hiroshi Echizen'ya, Kenji Araki and Eduard Hovy

Learning to Stop in Structured Prediction for Neural Machine Translation
Mingbo Ma, Renjie Zheng and Liang Huang

Learning Unsupervised Multilingual Word Embeddings with Incremental Multilingual Hubs
Geert Heyman, Bregt Verreet, Ivan Vulić and Marie-Francine Moens

Curriculum Learning for Domain Adaptation in Neural Machine Translation
Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat and Kevin Duh

Improving Robustness of Machine Translation with Synthetic Noise
Vaibhav Vaibhav, Sumeet Singh, Craig Stewart and Graham Neubig

Non-Parametric Adaptation for Neural Machine Translation
Ankur Bapna and Orhan Firat

Online Distilling from Checkpoints for Neural Machine Translation
Hao-Ran Wei, Shujian Huang, Ran Wang, Xin-Yu Dai and Jiajun Chen

Vision & Robotics

Value-based Search in Execution Space for Mapping Instructions to Programs
Dor Muhlgay, Jonathan Herzig and Jonathan Berant

VQD: Visual Query Detection In Natural Scenes
Manoj Acharya, Karan Jariwala and Christopher Kanan

Improving Natural Language Interaction with Robots Using Advice
Nikhil Mehta and Dan Goldwasser

Generating Knowledge Graph Paths from Textual Definitions using Sequence-to-Sequence Models
Victor Prokhorov, Mohammad Taher Pilehvar and Nigel Collier

Shifting the Baseline: Single Modality Performance on Visual Navigation & QA
Jesse Thomason, Daniel Gordon and Yonatan Bisk

ExCL: Extractive Clip Localization Using Natural Language Descriptions
Soham Ghosh, Anuva Agarwal, Zarana Parekh and Alexander Hauptmann

10:30–11:00 *Coffee Break*

11:00–12:30 ***Oral Sessions (short papers), Posters (long and short papers) & Demos***

Session 5A: Multilingual NLP
Room: Nicollet D, Chair: **Valia Kordoni**

11:00–11:15 *Detecting dementia in Mandarin Chinese using transfer learning from a parallel corpus*
Bai Li, Yi-Te Hsu and Frank Rudzicz

11:15–11:30 *Cross-lingual Visual Verb Sense Disambiguation*
Spandana Gella, Desmond Elliott and Frank Keller

11:30–11:45 *Subword-Level Language Identification for Intra-Word Code-Switching*
Manuel Mager, Özlem Çetinoğlu and Katharina Kann

11:45–12:00 *MuST-C: a Multilingual Speech Translation Corpus*
Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri and Marco Turchi

12:00–12:15 *Contextualization of Morphological Inflection*
Ekaterina Vylomova, Ryan Cotterell, Trevor Cohn, Timothy Baldwin and Jason Eisner

12:15–12:30 *A Robust Abstractive System for Cross-Lingual Summarization*
Jessica Ouyang, Boya Song and Kathy McKeown

Session 5B: Machine Translation

Room: Nicollet B+C, Chair: **Daisuke Kawahara**

11:00–11:15 *Improving Neural Machine Translation with Neural Syntactic Distance*
Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita and Tiejun Zhao

11:15–11:30 *Measuring Immediate Adaptation Performance for Neural Machine Translation*
Patrick Simianer, Joern Wuebker and John DeNero

11:30–11:45 *Differentiable Sampling with Flexible Reference Word Order for Neural Machine Translation*
Weijia Xu, Xing Niu and Marine Carpuat

11:45–12:00 *Reinforcement Learning based Curriculum Optimization for Neural Machine Translation*
Gaurav Kumar, George Foster, Colin Cherry and Maxim Krikun

12:00–12:15 *Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation*
Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh and Philipp Koehn

12:15–12:30 [SRW] *Multimodal Machine Translation with Embedding Prediction*
Tosho Hirasawa, Hayahide Yamagishi, Yukio Matsumura and Mamoru Komachi

Session 5C: Social Media
Room: Greenway, Chair: **Chenhao Tan**

11:00–11:15 *Short-Term Meaning Shift: A Distributional Exploration*
Marco Del Tredici, Raquel Fernández and Gemma Boleda

11:15–11:30 *Detecting Derogatory Compounds – An Unsupervised Approach*
Michael Wiegand, Maximilian Wolf and Josef Ruppenhofer

11:30–11:45 *Personalized Neural Embeddings for Collaborative Filtering with Text*
Guangneng Hu

11:45–12:00 *An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models*
Alexandra Chronopoulou, Christos Baziotis and Alexandros Potamianos

12:00–12:15 *Incorporating Emoji Descriptions Improves Tweet Classification*
Abhishek Singh, Eduardo Blanco and Wei Jin

12:15–12:30 *Modeling Personal Biases in Language Use by Inducing Personalized Word Embeddings*
Daisuke Oba, Naoki Yoshinaga, Shoetsu Sato, Satoshi Akasaki and Masashi Toyoda

Session 5D: Text Analysis
Room: Northstar A, Chair: **Saif Mohammad**

11:00–11:15 *Multi-Task Ordinal Regression for Jointly Predicting the Trustworthiness and the Leading Political Ideology of News Media*
Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass and Preslav Nakov

11:15–11:30 *Joint Detection and Location of English Puns*
Yanyan Zou and Wei Lu

11:30–11:45 *Harry Potter and the Action Prediction Challenge from Natural Language*
David Vilares and Carlos Gómez-Rodríguez

11:45–12:00 *Argument Mining for Understanding Peer Reviews*
Xinyu Hua, Mitko Nikolov, Nikhil Badugu and Lu Wang

12:00–12:15 *An annotated dataset of literary entities*
David Bamman, Sejal Popat and Sheng Shen

12:15–12:30 *Abusive Language Detection with Graph Convolutional Networks*
Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis and Ekaterina Shutova

Session 5E: Semantics
Room: Nicollet A, Chair: **Samuel Bowman**

11:00–11:15 *On the Importance of Distinguishing Word Meaning Representations: A Case Study on Reverse Dictionary Mapping*
Mohammad Taher Pilehvar

11:15–11:30 *Factorising AMR generation through syntax*
Kris Cao and Stephen Clark

11:30–11:45 *A Crowdsourced Frame Disambiguation Corpus with Ambiguity*
Anca Dumitrache, Lora Aroyo and Chris Welty

11:45–12:00 *Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets*
Nelson F. Liu, Roy Schwartz and Noah A. Smith

12:00–12:15 [SRW] *Word Polysemy Aware Document Vector Estimation*
Vivek Gupta, Ankit Saw, Harshit Gupta, Pegah Nokhiz and Partha Talukdar

12:15–12:30 [SRW] *EQUATE: A Benchmark Evaluation Framework for Quantitative Reasoning in Natural Language Inference*
Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose and Eduard Hovy

Session 5F: Information Retrieval, Question Answering, Generation & Syntax (Posters & Demos)

Information Retrieval
A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization
Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen and Dinh Phung

Partial Or Complete, That's The Question
Qiang Ning, Hangfeng He, Chuchu Fan and Dan Roth

Sequential Attention with Keyword Mask Model for Community-based Question Answering
Jianxin Yang, Wenge Rong, Libin Shi and Zhang Xiong

Simple Attention-Based Representation Learning for Ranking Short Social Media Posts
Peng Shi, Jinfeng Rao and Jimmy Lin

AttentiveChecker: A Bi-Directional Attention Flow Mechanism for Fact Verification
Santosh Tokala, Vishal G, Avirup Saha and Niloy Ganguly

Practical, Efficient, and Customizable Active Learning for Named Entity Recognition in the Digital Humanities
Alexander Erdmann, David Joseph Wrisley, Benjamin Allen, Christopher Brown, Sophie Cohen-Bodénès, Micha Elsner, Yukun Feng, Brian Joseph, Béatrice Joyeux-Prunel and Marie-Catherine de Marneffe

Doc2hash: Learning Discrete Latent variables for Documents Retrieval
Yifei Zhang and Hao Zhu

Generation

Evaluating Text GANs as Language Models
Guy Tevet, Gavriel Habib, Vered Shwartz and Jonathan Berant

Latent Code and Text-based Generative Adversarial Networks for Soft-text Generation
Md Akmal Haidar, Mehdi Rezagholizadeh, Alan Do Omri and Ahmad Rashid

Neural Text Generation from Rich Semantic Representations
Valerie Hajdik, Jan Buys, Michael Wayne Goodman and Emily M. Bender

Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation
Amit Moryossef, Yoav Goldberg and Ido Dagan

Evaluating Rewards for Question Generation Models
Tom Hosking and Sebastian Riedel

Text Generation from Knowledge Graphs with Graph Transformers
Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata and Hannaneh Hajishirzi

Question Answering

Open Information Extraction from Question-Answer Pairs
Nikita Bhutani, Yoshihiko Suhara, Wang-Chiew Tan, Alon Halevy and H. V. Jagadish

Question Answering by Reasoning Across Documents with Graph Convolutional Networks
Nicola De Cao, Wilker Aziz and Ivan Titov

A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC
Mark Yatskar

BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis
Hu Xu, Bing Liu, Lei Shu and Philip Yu

Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text
Ahmad Sakor, Isaiah Onando Mulang', Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann and Sören Auer

Be Consistent! Improving Procedural Text Comprehension using Label Consistency
Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark and Claire Cardie

MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms
Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi and Hannaneh Hajishirzi

DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs
Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh and Matt Gardner

Syntax
An Encoding Strategy Based Word-Character LSTM for Chinese NER
Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song and Yueran Zu

Highly Effective Arabic Diacritization using Sequence to Sequence Modeling
Hamdy Mubarak, Ahmed Abdelali, Hassan Sajjad, Younes Samih and Kareem Darwish

SC-LSTM: Learning Task-Specific Representations in Multi-Task Learning for Sequence Labeling
Peng Lu, Ting Bai and Philippe Langlais

Learning to Denoise Distantly-Labeled Data for Entity Typing
Yasumasa Onoe and Greg Durrett

A Simple and Robust Approach to Detecting Subject-Verb Agreement Errors
Simon Flachs, Ophélie Lacroix, Marek Rei, Helen Yannakoudakis and Anders Søgaard

A Grounded Unsupervised Universal Part-of-Speech Tagger for Low-Resource Languages
Ronald Cardenas, Ying Lin, Heng Ji and Jonathan May

On Difficulties of Cross-Lingual Transfer with Order Differences: A Case Study on Dependency Parsing
Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang and Nanyun Peng

A Multi-Task Approach for Disentangling Syntax and Semantics in Sentence Representations
Mingda Chen, Qingming Tang, Sam Wiseman and Kevin Gimpel

12:30–14:00 *Lunch Break*

14:00–15:00 *Keynote 2: Rada Mihalcea "When the Computers Spot the Lie (and People Don't)" (Nicollet Grand Ballroom)*

15:00–15:30 *Coffee Break*

15:30–17:00 ***Oral sessions (long papers), Posters (long and short papers) & Demos***

Session 6A: Sentiment Analysis
Room: Northstar A, Chair: **Sara Rosenthal**

15:30–15:48 *Self-Discriminative Learning for Unsupervised Document Embedding*
Hong-You Chen, Chin-Hua Hu, Leila Wehbe and Shou-de Lin

15:48–16:06 *Adaptive Convolution for Text Classification*
Byung-Ju Choi, Jun-Hyung Park and SangKeun Lee

16:06–16:24 *Zero-Shot Cross-Lingual Opinion Target Extraction*
Soufian Jebbara and Philipp Cimiano

16:24–16:42 *Adversarial Category Alignment Network for Cross-domain Sentiment Classification*
Xiaoye Qu, Zhikang Zou, Yu Cheng, Yang Yang and Pan Zhou

16:42–17:00 *Target-oriented Opinion Words Extraction with Target-fused Neural Sequence Labeling*
Zhifang Fan, Zhen Wu, Xin-Yu Dai, Shujian Huang and Jiajun Chen

Session 6B: Summarization

Room: Greenway, Chair: **Ani Nenkova**

15:30–15:48 *Abstractive Summarization of Reddit Posts with Multi-level Memory Networks*
Byeongchang Kim, Hyunwoo Kim and Gunhee Kim

15:48–16:06 *Automatic learner summary assessment for reading comprehension*
Menglin Xia, Ekaterina Kochmar and Ted Briscoe

16:06–16:24 *Data-efficient Neural Text Compression with Interactive Learning*
Avinesh P.V.S and Christian M. Meyer

16:24–16:42 *Text Generation with Exemplar-based Adaptive Decoding*
Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra and Dipanjan Das

16:42–17:00 *Guiding Extractive Summarization with Question-Answering Rewards*
Kristjan Arumae and Fei Liu

Session 6C: Vision & Robotics

Room: Nicollet A, Chair: **William Yang Wang**

15:30–15:48 *Beyond task success: A closer look at jointly learning to see, ask, and GuessWhat*
Ravi Shekhar, Aashish Venkatesh, Tim Baumgärtner, Elia Bruni, Barbara Plank, Raffaella Bernardi and Raquel Fernández

15:48–16:06 *The World in My Mind: Visual Dialog with Adversarial Multi-modal Feature Encoding*
Yiqun Yao, Jiaming Xu and Bo Xu

16:06–16:24 *Strong and Simple Baselines for Multimodal Utterance Embeddings*
Paul Pu Liang, Yao Chong Lim, Yao-Hung Hubert Tsai, Ruslan Salakhutdinov and Louis-Philippe Morency

16:24–16:42 *Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout*
Hao Tan, Licheng Yu and Mohit Bansal

16:42–17:00 *Towards Content Transfer through Grounded Text Generation*
Shrimai Prabhumoye, Chris Quirk and Michel Galley

Session 6D: Question Answering
Room: Nicollet B+C, Chair: **Eduardo Blanco**

15:30–15:48 *Improving Machine Reading Comprehension with General Reading Strategies*
Kai Sun, Dian Yu, Dong Yu and Claire Cardie

15:48–16:06 *Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension*
Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu and Jianfeng Gao

16:06–16:24 *Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems*
Ting-Rui Chiang and Yun-Nung Chen

16:24–16:42 *Iterative Search for Weakly Supervised Semantic Parsing*
Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer and Eduard Hovy

16:42–17:00 *Alignment over Heterogeneous Embeddings for Question Answering*
Vikas Yadav, Steven Bethard and Mihai Surdeanu

Session 6E: Industry Session: Deployed Systems
Room: Nicollet D

Session 6F: Phonology, Speech and Text Mining (Posters & Demos)

Phonology
Bridging the Gap: Attending to Discontinuity in Identification of Multiword Expressions
Omid Rohanian, Shiva Taslimipoor, Samaneh Kouchaki, Le An Ha and Ruslan Mitkov

Incorporating Word Attention into Character-Based Word Segmentation
Shohei Higashiyama, Masao Utiyama, Eiichiro Sumita, Masao Ideuchi, Yoshiaki Oida, Yohei Sakamoto and Isaac Okada

VCWE: Visual Character-Enhanced Word Embeddings
Chi Sun, Xipeng Qiu and Xuanjing Huang

Subword Encoding in Lattice LSTM for Chinese Word Segmentation
Jie Yang, Yue Zhang and Shuailong Liang

Improving Cross-Domain Chinese Word Segmentation with Word Embeddings
Yuxiao Ye, Weikang Li, Yue Zhang, Likun Qiu and Jian Sun

Neural Semi-Markov Conditional Random Fields for Robust Character-Based Part-of-Speech Tagging
Apostolos Kemos, Heike Adel and Hinrich Schütze

Shrinking Japanese Morphological Analyzers With Neural Networks and Semi-supervised Learning
Arseny Tolmachev, Daisuke Kawahara and Sadao Kurohashi

[TACL] *Grammar Error Correction in Morphologically-Rich Languages: The Case of Russian*
Alla Rozovskaya and Dan Roth

[SRW] *Deep Learning and Sociophonetics: Automatic Coding of Rhoticity Using Neural Networks*
Sarah Gupta and Anthony DiPadova

[SRW] *Learn Languages First and Then Convert: towards Effective Simplified to Traditional Chinese Conversion*
Pranav A, S.F. Hui, I-Tsun Cheng, Ishaan Batra and Chiu Yik Hei

Speech
Neural Constituency Parsing of Speech Transcripts
Paria Jamshid Lou, Yufei Wang and Mark Johnson

Acoustic-to-Word Models with Conversational Context Information
Suyoun Kim and Florian Metze

A Dynamic Speaker Model for Conversational Interactions
Hao Cheng, Hao Fang and Mari Ostendorf

Fluent Translations from Disfluent Speech in End-to-End Speech Translation
Elizabeth Salesky, Matthias Sperber and Alexander Waibel

[SRW] *Data Augmentation by Data Noising for Open-vocabulary Slots in Spoken Language Understanding*
Hwa-Yeon Kim, Yoon-Hyung Roh and Young-Kil Kim

[SRW] *Expectation and Locality Effects in the Prediction of Disfluent Fillers and Repairs in English Speech*
Samvit Dammalapati, Rajakrishnan Rajkumar and Sumeet Agarwal

Text Mining

Relation Classification Using Segment-Level Attention-based CNN and Dependency-based RNN
Van-Hien Tran, Van-Thuy Phi, Hiroyuki Shindo and Yuji Matsumoto

Document-Level Event Factuality Identification via Adversarial Neural Network
Zhong Qian, Peifeng Li, Qiaoming Zhu and Guodong Zhou

Distant Supervision Relation Extraction with Intra-Bag and Inter-Bag Attentions
Zhi-Xiu Ye and Zhen-Hua Ling

Ranking-Based Autoencoder for Extreme Multi-label Classification
Bingyu Wang, Li Chen, Wei Sun, Kechen Qin, Kefeng Li and Hui Zhou

Posterior-regularized REINFORCE for Instance Selection in Distant Supervision
Qi Zhang, Siliang Tang, Xiang Ren, Fei Wu, Shiliang Pu and Yueting Zhuang

Scalable Collapsed Inference for High-Dimensional Topic Models
Rashidul Islam and James Foulds

An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction
Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing and Irwin King

Predicting Malware Attributes from Cybersecurity Texts
Arpita Roy, Youngja Park and Shimei Pan

Improving Distantly-supervised Entity Typing with Compact Latent Space Clustering
Bo Chen, Xiaotao Gu, Yufeng Hu, Siliang Tang, Guoping Hu, Yueting Zhuang and Xiang Ren

Modelling Instance-Level Annotator Reliability for Natural Language Labelling Tasks
Maolin Li, Arvid Fahlström Myrman, Tingting Mu and Sophia Ananiadou

Review-Driven Multi-Label Music Style Classification by Exploiting Style Correlations
Guangxiang Zhao, Jingjing Xu, Qi Zeng, Xuancheng Ren and Xu Sun

Fact Discovery from Knowledge Base via Facet Decomposition
Zihao Fu, Yankai Lin, Zhiyuan Liu and Wai Lam

A Richer-but-Smarter Shortest Dependency Path with Attentive Augmentation for Relation Extraction
Duy-Cat Can, Hoang-Quynh Le, Quang-Thuy Ha and Nigel Collier

Wednesday, June 5, 2019

9:00–10:00 *Keynote 3: Kieran Snyder "Leaving the Lab: Building NLP Applications that Real People can Use" (Nicollet Grand Ballroom)*

10:00–10:30 *Coffee Break*

10:30–12:00 ***Oral sessions (long papers) and Posters (long and short papers)***

Session 7A: Question Answering
Room: Greenway, Chair: **Alessandro Moschitti**

10:30–10:48 *Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases*
Yu Chen, Lingfei Wu and Mohammed J Zaki

10:48–11:06 *BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions*
Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins and Kristina Toutanova

11:06–11:24 *Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering*
Kun Xu, Yuxuan Lai, Yansong Feng and Zhiguo Wang

11:24–11:42 *Repurposing Entailment for Multi-Hop Question Answering Tasks*
Harsh Trivedi, Heeyoung Kwon, Tushar Khot, Ashish Sabharwal and Niranjan Balasubramanian

11:42–12:00 [TACL] *CoQA: A Conversational Question Answering Challenge*
Siva Reddy, Danqi Chen and Christopher D. Manning

Session 7B: Ethics, Bias & Fairness
Room: Nicollet A, Chair: **Emily Prud'hommeaux**

10:30–10:48 [TACL] *Mind the GAP: A Balanced Corpus of Gendered Ambiguous Pronouns*
Kellie Webster, Marta Recasens, Vera Axelrod and Jason Baldridge

10:48–11:06 *GenderQuant: Quantifying Mention-Level Genderedness*
Ananya, Nitya Parthasarthi and Sameer Singh

Wednesday, June 5, 2019 (continued)

11:06–11:24 *Analyzing Polarization in Social Media: Method and Application to Tweets on 21 Mass Shootings*
Dorottya Demszky, Nikhil Garg, Rob Voigt, James Zou, Jesse Shapiro, Matthew Gentzkow and Dan Jurafsky

11:24–11:42 *Learning to Decipher Hate Symbols*
Jing Qian, Mai ElSherief, Elizabeth Belding and William Yang Wang

11:42–12:00 [TACL] *Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science*
Emily M. Bender and Batya Friedman

Session 7C: Information Extraction
Room: Nicollet D, Chair: **Heng Ji**

10:30–10:48 *Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks*
Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang and Huajun Chen

10:48–11:06 *GAN Driven Semi-distant Supervision for Relation Extraction*
Pengshuai Li, Xinsong Zhang, Weijia Jia and Hai Zhao

11:06–11:24 *A general framework for information extraction using dynamic span graphs*
Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf and Hannaneh Hajishirzi

11:24–11:42 *OpenCeres: When Open Information Extraction Meets the Semi-Structured Web*
Colin Lockard, Prashant Shiralkar and Xin Luna Dong

11:42–12:00 *Structured Minimally Supervised Learning for Neural Relation Extraction*
Fan Bai and Alan Ritter

Session 7D: Machine Translation
Room: Northstar A, Chair: **Colin Cherry**

10:30–10:48 *Neural Machine Translation of Text from Non-Native Speakers*
Antonios Anastasopoulos, Alison Lui, Toan Q. Nguyen and David Chiang

10:48–11:06 *Improving Domain Adaptation Translation with Domain Invariant and Specific Information*
Shuhao Gu, Yang Feng and Qun Liu

Wednesday, June 5, 2019 (continued)

11:06–11:24 *Selective Attention for Context-aware Neural Machine Translation*
Sameen Maruf, André F. T. Martins and Gholamreza Haffari

11:24–11:42 *On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models*
Paul Michel, Xian Li, Graham Neubig and Juan Pino

11:42–12:00 *Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction*
Kazuma Hashimoto and Yoshimasa Tsuruoka

Session 7E: Text Analysis
Room: Nicollet B+C, Chair: **Steven Bethard**

10:30–10:48 *Mitigating Uncertainty in Document Classification*
Xuchao Zhang, Fanglan Chen, ChangTien Lu and Naren Ramakrishnan

10:48–11:06 *Complexity-Weighted Loss and Diverse Reranking for Sentence Simplification*
Reno Kriz, Joao Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki and Chris Callison-Burch

11:06–11:24 *Predicting Helpful Posts in Open-Ended Discussion Forums: A Neural Architecture*
Kishaloy Halder, Min-Yen Kan and Kazunari Sugiyama

11:24–11:42 *Text Classification with Few Examples using Controlled Generalization*
Abhijit Mahabal, Jason Baldridge, Burcu Karagol Ayan, Vincent Perot and Dan Roth

11:42–12:00 *Reinforcement Learning Based Text Style Transfer without Parallel Training Corpus*
Hongyu Gong, Suma Bhat, Lingfei Wu, JinJun Xiong and Wen-mei Hwu

Session 7F: Machine Learning, Tagging, Chunking, Syntax & Parsing (Posters)

Machine Learning

Adapting RNN Sequence Prediction Model to Multi-label Set Prediction
Kechen Qin, Cheng Li, Virgil Pavlu and Javed Aslam

Customizing Grapheme-to-Phoneme System for Non-Trivial Transcription Problems in Bangla Language
Sudipta Saha Shubha, Nafis Sadeq, Shafayat Ahmed, Md. Nahidul Islam, Muhammad Abdullah Adnan, Md. Yasin Ali Khan and Mohammad Zuberul Islam

Connecting Language and Knowledge with Heterogeneous Representations for Neural Relation Extraction
Peng Xu and Denilson Barbosa

Segmentation-free compositional n-gram embedding
Geewook Kim, Kazuki Fukui and Hidetoshi Shimodaira

Exploiting Noisy Data in Distant Supervision Relation Classification
Kaijia Yang, Liang He, Xin-Yu Dai, Shujian Huang and Jiajun Chen

Misspelling Oblivious Word Embeddings
Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira and Fabrizio Silvestri

Learning Relational Representations by Analogy using Hierarchical Siamese Networks
Gaetano Rossiello, Alfio Gliozzo, Robert Farrell, Nicolas Fauceglia and Michael Glass

An Effective Label Noise Model for DNN Text Classification
Ishan Jindal, Daniel Pressel, Brian Lester and Matthew Nokleby

Understanding Learning Dynamics Of Language Models with SVCCA
Naomi Saphra and Adam Lopez

Using Large Corpus N-gram Statistics to Improve Recurrent Neural Language Models
Yiben Yang, Ji-Ping Wang and Doug Downey

Continual Learning for Sentence Representations Using Conceptors
Tianlin Liu, Lyle Ungar and Joao Sedoc

Relation Discovery with Out-of-Relation Knowledge Base as Supervision
Yan Liang, Xin Liu, Jianwen Zhang and Yangqiu Song

Corpora Generation for Grammatical Error Correction
Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar and Simon Tong

Structural Supervision Improves Learning of Non-Local Grammatical Dependencies
Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros and Roger Levy

Benchmarking Approximate Inference Methods for Neural Structured Prediction
Lifu Tu and Kevin Gimpel

Evaluating and Enhancing the Robustness of Dialogue Systems: A Case Study on a Negotiation Agent
Minhao Cheng, Wei Wei and Cho-Jui Hsieh

Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications
Pouya Pezeshkpour, Yifan Tian and Sameer Singh

Analysis Methods in Neural Language Processing: A Survey
Yonatan Belinkov and James Glass

[TACL] *Attentive Convolution: Equipping CNNs with RNN-style Attention Mechanisms*
Wenpeng Yin and Hinrich Schütze

[TACL] *Rotational Unit of Memory: A Novel Representation Unit for RNNs with Scalable Applications*
Rumen Dangovski, Li Jing, Preslav Nakov, Mićo Tatalović and Marin Soljačić

Transferable Neural Projection Representations
Chinnadhurai Sankar, Sujith Ravi and Zornitsa Kozareva

[SRW] *Gating Mechanisms for Combining Character and Word-level Word Representations: an Empirical Study*
Jorge Balazs and Yutaka Matsuo

Tagging, Chunking, Syntax & Parsing
Semantic Role Labeling with Associated Memory Network
Chaoyu Guan, Yuhao Cheng and Hai Zhao

Better, Faster, Stronger Sequence Tagging Constituent Parsers
David Vilares, Mostafa Abdou and Anders Søgaard

CAN-NER: Convolutional Attention Network for Chinese Named Entity Recognition
Yuying Zhu and Guoxin Wang

Decomposed Local Models for Coordinate Structure Parsing
Hiroki Teranishi, Hiroyuki Shindo and Yuji Matsumoto

Multi-Task Learning for Japanese Predicate Argument Structure Analysis
Hikaru Omori and Mamoru Komachi

Domain adaptation for part-of-speech tagging of noisy user-generated text
Luisa März, Dietrich Trautmann and Benjamin Roth

Neural Chinese Address Parsing
Hao Li, Wei Lu, Pengjun Xie and Linlin Li

[SRW] *A Pregroup Representation of Word Order Alternation Using Hindi Syntax*
Alok Debnath and Manish Shrivastava

12:00–12:30 *Grab your lunch break*

12:30–13:30 *NAACL Business Meeting* (Nicollet B+C)

13:30–15:00 ***Oral Sessions (long papers) and Posters (long and short papers)***

Session 8A: Discourse
Room: Northstar A, Chair: **Vincent Ng**

13:30–13:48 *Learning Hierarchical Discourse-level Structure for Fake News Detection*
Hamid Karimi and Jiliang Tang

13:48–14:06 *DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion*
Mor Geva, Eric Malmi, Idan Szpektor and Jonathan Berant

14:06–14:24 *Linguistically-Informed Specificity and Semantic Plausibility for Dialogue Generation*
Wei-Jen Ko, Greg Durrett and Junyi Jessy Li

14:24–14:42 *Learning to Describe Unknown Phrases with Local and Global Contexts*
Shonosuke Ishiwatari, Hiroaki Hayashi, Naoki Yoshinaga, Graham Neubig, Shoetsu Sato, Masashi Toyoda and Masaru Kitsuregawa

14:42–15:00 *Mining Discourse Markers for Unsupervised Sentence Representation Learning*
Damien Sileo, Tim Van de Cruys, Camille Pradel and Philippe Muller

Session 8B: Machine Learning
Room: Nicollet B+C, Chair: **Anna Rumshisky**

13:30–13:48 *How Large a Vocabulary Does Text Classification Need? A Variational Approach to Vocabulary Selection*
Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan and William Yang Wang

13:48–14:06 *Subword-based Compact Reconstruction of Word Embeddings*
Shota Sasaki, Jun Suzuki and Kentaro Inui

14:06–14:24 *Bayesian Learning for Neural Dependency Parsing*
Ehsan Shareghi, Yingzhen Li, Yi Zhu, Roi Reichart and Anna Korhonen

14:24–14:42 *AutoSeM: Automatic Task Selection and Mixing in Multi-Task Learning*
Han Guo, Ramakanth Pasunuru and Mohit Bansal

14:42–15:00 *Studying the Inductive Biases of RNNs with Synthetic Variations of Natural Languages*
Shauli Ravfogel, Yoav Goldberg and Tal Linzen

Session 8C: Applications
Room: Nicollet A, Chair: **T. J. Hazen**

13:30–13:48 *Attention is not Explanation*
Sarthak Jain and Byron C. Wallace

13:48–14:06 *Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning*
Prithviraj Ammanabrolu and Mark Riedl

14:06–14:24 *Information Aggregation for Multi-Head Attention with Routing-by-Agreement*
Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R. Lyu and Zhaopeng Tu

14:24–14:42 *Context Dependent Semantic Parsing over Temporally Structured Data*
Charles Chen and Razvan Bunescu

14:42–15:00 *Structural Scaffolds for Citation Intent Classification in Scientific Publications*
Arman Cohan, Waleed Ammar, Madeleine van Zuylen and Field Cady

Session 8D: Semantics

Room: Greenway, Chair: **Matt Gardner**

13:30–13:48 *pair2vec: Compositional Word-Pair Embeddings for Cross-Sentence Inference*
Mandar Joshi, Eunsol Choi, Omer Levy, Daniel Weld and Luke Zettlemoyer

13:48–14:06 *Submodular Optimization-based Diverse Paraphrasing and its Effectiveness in Data Augmentation*
Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari and Partha Talukdar

14:06–14:24 *Let's Make Your Request More Persuasive: Modeling Persuasive Strategies via Semi-Supervised Neural Nets on Crowdfunding Platforms*
Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky and Eduard Hovy

14:24–14:42 *Recursive Routing Networks: Learning to Compose Modules for Language Understanding*
Ignacio Cases, Clemens Rosenbaum, Matthew Riemer, Atticus Geiger, Tim Klinger, Alex Tamkin, Olivia Li, Sandhini Agarwal, Joshua D. Greene, Dan Jurafsky, Christopher Potts and Lauri Karttunen

14:42–15:00 *Structural Neural Encoders for AMR-to-text Generation*
Marco Damonte and Shay B. Cohen

Session 8E: Biomedical NLP & Clinical Text Processing

Room: Nicollet D, Chair: **Timothy Miller**

13:30–13:48 *Multilingual prediction of Alzheimer's disease through domain adaptation and concept-based language modelling*
Kathleen C. Fraser, Nicklas Linz, Bai Li, Kristina Lundholm Fors, Frank Rudzicz, Alexandra Konig, Jan Alexandersson, Philippe Robert and Dimitrios Kokkinakis

13:48–14:06 *Ranking and Selecting Multi-Hop Knowledge Paths to Better Predict Human Needs*
Debjit Paul and Anette Frank

14:06–14:24 *NLP Whack-A-Mole: Challenges in Cross-Domain Temporal Expression Extraction*
Amy Olex, Luke Maffey and Bridget McInnes

14:24–14:42 *Document-Level N-ary Relation Extraction with Multiscale Representation Learning*
Robin Jia, Cliff Wong and Hoifung Poon

14:42–15:00 *Inferring Which Medical Treatments Work from Reports of Clinical Trials*
Eric Lehman, Jay DeYoung, Regina Barzilay and Byron C. Wallace

Session 8F: Dialogue, Multilingual NLP & Summarization (Posters)

Dialogue

Decay-Function-Free Time-Aware Attention to Context and Speaker Indicator for Spoken Language Understanding
Jonggu Kim and Jong-Hyeok Lee

Dialogue Act Classification with Context-Aware Self-Attention
Vipul Raheja and Joel Tetreault

Affect-Driven Dialog Generation
Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy and Mubbasir Kapadia

Multi-Level Memory for Task Oriented Dialogs
Revanth Gangi Reddy, Danish Contractor, Dinesh Raghu and Sachindra Joshi

Topic Spotting using Hierarchical Networks with Self Attention
Pooja Chitkara, Ashutosh Modi, Pravalika Avvaru, Sepehr Janghorbani and Mubbasir Kapadia

Top-Down Structurally-Constrained Neural Response Generation with Lexicalized Probabilistic Context-Free Grammar
Wenchao Du and Alan W. Black

What do Entity-Centric Models Learn? Insights from Entity Linking in Multi-Party Dialogue
Laura Aina, Carina Silberer, Ionut-Teodor Sorodoc, Matthijs Westera and Gemma Boleda

Continuous Learning for Large-scale Personalized Domain Classification
Han Li, Jihwan Lee, Sidharth Mudgal, Ruhi Sarikaya and Young-Bum Kim

Multilingual NLP

Cross-lingual Transfer Learning for Multilingual Task Oriented Dialog
Sebastian Schuster, Sonal Gupta, Rushin Shah and Mike Lewis

Evaluating Coherence in Dialogue Systems using Entailment
Nouha Dziri, Ehsan Kamalloo, Kory Mathewson and Osmar Zaiane

On Knowledge distillation from complex networks for response prediction
Siddhartha Arora, Mitesh M. Khapra and Harish G. Ramaswamy

Cross-lingual Multi-Level Adversarial Transfer to Enhance Low-Resource Name Tagging
Lifu Huang, Heng Ji and Jonathan May

Unsupervised Extraction of Partial Translations for Neural Machine Translation
Benjamin Marie and Atsushi Fujita

Low-Resource Syntactic Transfer with Unsupervised Source Reordering
Mohammad Sadegh Rasooli and Michael Collins

Revisiting Adversarial Autoencoder for Unsupervised Word Translation with Cycle Consistency and Improved Training
Tasnim Mohiuddin and Shafiq Joty

Addressing word-order Divergence in Multilingual Neural Machine Translation for extremely Low Resource Languages
Rudra Murthy, Anoop Kunchukuttan and Pushpak Bhattacharyya

Massively Multilingual Neural Machine Translation
Roee Aharoni, Melvin Johnson and Orhan Firat

A Large-Scale Comparison of Historical Text Normalization Systems
Marcel Bollmann

Combining Discourse Markers and Cross-lingual Embeddings for Synonym–Antonym Classification
Michael Roth and Shyam Upadhyay

Context-Aware Cross-Lingual Mapping
Hanan Aldarmaki and Mona Diab

Polyglot Contextual Representations Improve Crosslingual Transfer
Phoebe Mulcaire, Jungo Kasai and Noah A. Smith

Typological Features for Multilingual Delexicalised Dependency Parsing
Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre and Carlos Ramisch

Wednesday, June 5, 2019 (continued)

Summarization

Recommendations for Datasets for Source Code Summarization
Alexander LeClair and Collin McMillan

Question Answering as an Automatic Evaluation Metric for News Article Summarization
Matan Eyal, Tal Baumel and Michael Elhadad

Understanding the Behaviour of Neural Abstractive Summarizers using Contrastive Examples
Krtin Kumar and Jackie Chi Kit Cheung

Jointly Extracting and Compressing Documents with Summary State Representations
Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins and Shay B. Cohen

News Article Teaser Tweets and How to Generate Them
Sanjeev Kumar Karn, Mark Buckley, Ulli Waltinger and Hinrich Schütze

Cross-referencing Using Fine-grained Topic Modeling
Jeffrey Lund, Piper Armstrong, Wilson Fearn, Stephen Cowley, Emily Hales and Kevin Seppi

Conversation Initiation by Diverse News Contents Introduction
Satoshi Akasaki and Nobuhiro Kaji

Positional Encoding to Control Output Sequence Length
Sho Takase and Naoaki Okazaki

15:00–15:30 *Coffee Break*

15:30–16:30 ***Oral Sessions (short papers) and Posters (Industry track)***

Session 9A: Question Answering
Room: Greenway, Chair: **Mo Yu**

15:30–15:45 *The Lower The Simpler: Simplifying Hierarchical Recurrent Models*
Chao Wang and Hui Jiang

15:45–16:00 *Using Natural Language Relations between Answer Choices for Machine Comprehension*
Rajkumar Pujari and Dan Goldwasser

16:00–16:15 *Saliency Learning: Teaching the Model Where to Pay Attention*
Reza Ghaeini, Xiaoli Fern, Hamed Shahbazi and Prasad Tadepalli

16:15–16:30 *Understanding Dataset Design Choices for Multi-hop Reasoning*
Jifan Chen and Greg Durrett

Session 9B: Applications
Room: Nicollet A, Chair: **Zornitsa Kozareva**

15:30–15:45 *Neural Grammatical Error Correction with Finite State Transducers*
Felix Stahlberg, Christopher Bryant and Bill Byrne

15:45–16:00 *Convolutional Self-Attention Networks*
Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao and Zhaopeng Tu

16:00–16:15 *Rethinking Complex Neural Network Architectures for Document Classification*
Ashutosh Adhikari, Achyudh Ram, Raphael Tang and Jimmy Lin

16:15–16:30 [SRW] *Speak Up, Fight Back! Detection of Social Media Disclosures of Sexual Harassment*
Arijit Ghosh Chowdhury, Ramit Sawhney, Puneet Mathur, Debanjan Mahata and Rajiv Ratn Shah

Session 9C: Generation
Room: Northstar A, Chair: **Fei Liu**

15:30–15:45 *Pre-trained language model representations for language generation*
Sergey Edunov, Alexei Baevski and Michael Auli

15:45–16:00 *Pragmatically Informative Text Generation*
Sheng Shen, Daniel Fried, Jacob Andreas and Dan Klein

16:00–16:15 *Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation*
Hareesh Bahuleyan, Lili Mou, Hao Zhou and Olga Vechtomova

16:15–16:30 *Benchmarking Hierarchical Script Knowledge*
Yonatan Bisk, Jan Buys, Karl Pichotta and Yejin Choi

Session 9D: Cognitive Modeling & Psycholinguistics
Room: Nicollet D, Chair: **Bridget McInnes**

15:30–15:45 [SRW] *SNAP-BATNET: Cascading Author Profiling and Social Network Graphs for Suicide Ideation Detection on Social Media*
Rohan Mishra, Pradyumn Prakhar Sinha, Ramit Sawhney, Debanjan Mahata, Puneet Mathur and Rajiv Ratn Shah

15:45–16:00 *A large-scale study of the effects of word frequency and predictability in naturalistic reading*
Cory Shain

16:00–16:15 *Augmenting word2vec with latent Dirichlet allocation within a clinical application*
Akshay Budhkar and Frank Rudzicz

16:15–16:30 *On the Idiosyncrasies of the Mandarin Chinese Classifier System*
Shijia Liu, Hongyuan Mei, Adina Williams and Ryan Cotterell

Session 9E: Machine Learning
Room: Nicollet B+C, Chair: **Byron C. Wallace**

15:30–15:45 *Joint Learning of Pre-Trained and Random Units for Domain Adaptation in Part-of-Speech Tagging*
Sara Meftah, Youssef Tamaazousti, Nasredine Semmar, Hassane Essafi and Fatiha Sadat

15:45–16:00 *Show Some Love to Your n-grams: A Bit of Progress and Stronger n-gram Language Modeling Baselines*
Ehsan Shareghi, Daniela Gerz, Ivan Vulić and Anna Korhonen

16:00–16:15 *Training Data Augmentation for Context-Sensitive Neural Lemmatizer Using Inflection Tables and Raw Text*
Toms Bergmanis and Sharon Goldwater

16:15–16:30 *A Structural Probe for Finding Syntax in Word Representations*
John Hewitt and Christopher D. Manning

Session 9F: Industry Session (posters)
Room: Hyatt Exhibit Hall

16:30–16:45 *Short Break*

Best Paper Session
Room: (Nicollet Grand Ballroom)

16:45–17:03 *CNM: An Interpretable Complex-valued Network for Matching*
Qiuchi Li, Benyou Wang and Massimo Melucci

17:03–17:21 *CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge*
Alon Talmor, Jonathan Herzig, Nicholas Lourie and Jonathan Berant

17:21–17:39 *Probing the Need for Visual Context in Multimodal Machine Translation*
Ozan Caglayan, Pranava Madhyastha, Lucia Specia and Loïc Barrault

17:39–17:57 *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova

17:57–18:15 *What's in a Name? Reducing Bias in Bios without Access to Protected Attributes*
Alexey Romanov, Maria De-Arteaga, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, Anna Rumshisky and Adam Kalai

18:15–18:30 *Closing Remarks*

Entity Recognition at First Sight: Improving NER with Eye Movement Information

Nora Hollenstein
ETH Zurich
`noraho@ethz.ch`

Ce Zhang
ETH Zurich
`ce.zhang@inf.ethz.ch`

Abstract

Previous research shows that eye-tracking data contains information about the lexical and syntactic properties of text, which can be used to improve natural language processing models. In this work, we leverage eye movement features from three corpora with recorded gaze information to augment a state-of-the-art neural model for named entity recognition (NER) with gaze embeddings. These corpora were manually annotated with named entity labels. Moreover, we show how gaze features, generalized on word type level, eliminate the need for recorded eye-tracking data at test time. The gaze-augmented models for NER using token-level and type-level features outperform the baselines. We present the benefits of eye-tracking features by evaluating the NER models on both individual datasets as well as in cross-domain settings.

1 Introduction

The field of natural language processing includes studies of tasks of different granularity and depths of semantics: from lower level tasks such as tokenization and part-of-speech tagging up to higher level tasks of information extraction such as named entity recognition, relation extraction, and semantic role labeling (Collobert et al., 2011). As NLP systems become increasingly prevalent in society, how to take advantage of information passively collected from human readers, e.g. eye movement signals, is becoming more interesting to researchers. Previous research in this area has shown promising results: Eye-tracking data has been used to improve tasks such as part-of-speech tagging (Barrett et al., 2016), sentiment analysis (Mishra et al., 2017), prediction of multiword expressions (Rohanian et al., 2017), and word embedding evaluation (Søgaard, 2016).

However, most of these studies focus on either relatively lower-level tasks (e.g. part-of-speech tagging and multiword expressions) or relatively global properties in the text (e.g. sentiment analysis). In this paper, we test a hypothesis on a different level: *Can eye movement signals also help improve higher-level semantic tasks such as extracting information from text?*

The answer to this question is not obvious. On one hand, the quality improvement attributed to eye movement signals on lower-level tasks implies that such signals do contain linguistic information. On the other hand, it is not clear whether these signals can also provide significant improvement for tasks dealing with higher-level semantics. Moreover, even if eye movement patterns contain signals related to higher-level tasks, as implied by a recent psycholinguistic study (Tokunaga et al., 2017), noisy as these signals are, it is not straightforward whether they would help, if not hurt, the quality of the models.

In this paper, we provide the first study of the impact of gaze features to automatic named entity recognition from text. We test the hypothesis that eye-tracking data is beneficial for entity recognition in a state-of-the-art neural named entity tagger augmented with embedding layers of gaze features. Our contributions in the current work can be summarized as follows:

1. First, we manually annotate three eye-tracking corpora with named entity labels to train a neural NER system with gaze features. This collection of corpora facilitates future research in related topics. The annotations are publicly available.

2. Beyond that, we present a neural architecture for NER, which in addition to textual information, incorporates embedding layers to encode eye movement information.

3. Finally, we show how gaze features generalized to word types eliminate the need for recorded eye-tracking data at test time. This

Proceedings of NAACL-HLT 2019, pages 1–10
Minneapolis, Minnesota, June 2 - June 7, 2019.

makes the use of eye-tracking data in NLP applications more feasible since recorded eye-tracking data for each token in context is not required anymore at prediction time. Moreover, type-aggregated features appear to be particularly useful for cross-domain systems.

Our hypotheses are evaluated not only on the available eye-tracking corpora, but also on an external benchmark dataset, for which gaze information does not exist.

2 Related Work

The benefits of eye movement data for machine learning have been assessed in various domains, including NLP and computer vision. Eye-trackers provide millisecond-accurate records on where humans look when they are reading, and they are becoming cheaper and more easily available by the day (San Agustin et al., 2009; Sewell and Komogortsev, 2010). Although eye-tracking data is still being recorded in controlled experiment environments, this will likely change in the near future. Recent approaches have shown substantial improvements in recording gaze data while reading by using cameras of mobile devices (Gómez-Poveda and Gaudioso, 2016; Papoutsaki et al., 2016). Hence, eye-tracking data will probably be more accessible and available in much larger volumes in due time, which will facilitate the creation of sizable datasets enormously.

Tokunaga et al. (2017) recently analyzed eye-tracking signals during the annotation of named entities to find effective features for NER. Their work proves that humans take into account a broad context to identify named entities, including predicate-argument structure. This further strengthens our intuition to use eye movement information to improve existing NER systems. And going even a step further, it opens the possibility for real-time entity annotation based on the reader's eye movements.

The benefit of eye movement data is backed up by extensive psycholinguistic studies. For example, when humans read a text they do not focus on every single word. The number of fixations and the fixation duration on a word depends on a number of linguistic factors (Clifton et al., 2007; Demberg and Keller, 2008). First, readers are more likely to fixate on open-class words that are not predictable from context (Rayner, 1998). Reading patterns are a reliable indicator of syntactical categories (Barrett and Søgaard, 2015a). Second, word frequency and word familiarity influence how long readers look at a word. The frequency effect was first noted by Rayner (1977) and has been reported in various studies since, e.g. Just and Carpenter (1980) and Cop et al. (2017). Moreover, although two words may have the same frequency value, they may differ in familiarity (especially for infrequent words). Effects of word familiarity on fixation time have also been demonstrated in a number of recent studies (Juhasz and Rayner, 2003; Williams and Morris, 2004). Additionally, the positive effect of fixation information in various NLP tasks has recently been shown by Barrett et al. (2018), where an attention mechanism is trained on fixation duration.

State-of-the-art NER Non-linear neural networks with distributed word representations as input have become increasingly successful for any sequence labeling task in NLP (Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016). The same applies to named entity recognition: State-of-the-art systems are combinations of neural networks such as LSTMs or CNNs and conditional random fields (CRFs) (Strauss et al., 2016). Lample et al. (2016) developed such a neural architecture for NER, which we employ in this work and enhance with eye movement features. Their model successfully combines word-level and character-level embeddings, which we augment with embedding layers for eye-tracking features.

3 Eye-tracking corpora

For our experiments, we resort to three eye-tracking data resources: the *Dundee corpus* (Kennedy et al., 2003), the *GECO corpus* (Cop et al., 2017) and the *ZuCo corpus* (Hollenstein et al., 2018). For the purpose of information extraction, it is important that the readers process longer fragments of text, i.e. complete sentences instead of single words, which is the case in all three datasets.

Table 1 shows an overview of the domain and size of these datasets. In total, they comprise 142,441 tokens with gaze information. Table 1 also shows the differences in mean fixation times between the datasets (i.e. fixation duration (the average duration of a single fixation on a word in

	Dundee	**GECO**	**ZuCo**	**Total**
domain(s)	news articles	literature	movie reviews, Wikipedia articles	-
number of sentences	2367	5424	700	8491
mean sentence length	24.75	12.65	22.12	19.84
number of words	58598	68606	15237	142441
unique word types	9131	5283	4408	13937
mean word length	4.29	3.76	4.44	4.16
fixation duration (ms)	202	214	226	214
gaze duration (ms)	237	232	265	244.7

Table 1: Descriptive statistics of the eye-tracking corpora, including domain, size and mean fixation and gaze duration per token.

	Dundee		**GECO**		**ZuCo**		**Total**	
	all	unique	all	unique	all	unique	all	unique
PERSON	732	415	1870	108	657	446	3259	955
ORGANIZATION	475	261	26	12	156	95	657	364
LOCATION	431	177	101	23	366	155	898	1646
total	**1638**	**853**	**1997**	**143**	**1179**	**696**	**4814**	**1646**
		52%		7%		59%		34%

Table 2: Number and distribution of named entity annotations in all three eye-tracking corpora.

milliseconds) and gaze duration (the average duration of all fixations on a word)).

Dundee Corpus The gaze data of the Dundee corpus (Kennedy et al., 2003) was recorded with a *Dr. Bouis Oculometer Eyetracker*. The English section of this corpus comprises 58,598 tokens in 2,367 sentences. It contains eye movement information of ten native English speakers as they read the same 20 newspaper articles from *The Independent*. The text was presented to the readers on a screen five lines at a time. This data has been widely used in psycholinguistic research to analyze the reading behavior of subjects while reading sentences in context under relatively naturalistic conditions.

GECO Corpus The Ghent Eye-Tracking Corpus (Cop et al., 2017) is a more recent dataset, which was created for the analysis of eye movements of monolingual and bilingual subjects during reading. The data was recorded with an *EyeLink 1000* system. The text was presented one paragraph at a time. The subjects read the entire novel *The Mysterious Affair at Styles* by Agatha Christie (1920) containing 68,606 tokens in 5,424 sentences. We use only the monolingual data recorded from the 14 native English speakers for this work to maintain consistency across corpora.

ZuCo Corpus The Zurich Cognitive Language Processing Corpus (Hollenstein et al., 2018) is a combined eye-tracking and EEG dataset. The gaze data was also recorded with an *EyeLink 1000* system. The full corpus contains 1,100 English sentences read by 12 adult native speakers. The sentences were presented at the same position on the screen one at a time. For the present work, we only use the eye movement data of the first two reading tasks of this corpus (700 sentences, 15,237 tokens), since these tasks encouraged natural reading. The reading material included sentences from movie reviews from the Stanford Sentiment Treebank (Socher et al., 2013) and the Wikipedia dataset by Culotta et al. (2006).

For the purposes of this work, all datasets were manually annotated with named entity labels for three categories: PERSON, ORGANIZATION and LOCATION. The annotations are available at `https://github.com/DS3Lab/ner-at-first-sight`.

The datasets were annotated by two NLP experts. The IOB tagging scheme was used for the labeling. We followed the ACE Annotation Guidelines (Linguistic Data Consortium, 2005). All conflicts in labelling were resolved by adjudication between both annotators. An inter-

Basic	
n fixations fixation probability mean fixation duration	total number of fixations on a word *w* the probability that a word *w* will be fixated mean of all fixation durations for a word *w*
Early	
first fixation duration first pass duration	duration of the first fixation on a word *w* sum of all fixation durations during the first pass
Late	
total fixation duration *n* re-fixations re-read probability	sum of all fixation durations for a word *w* number of times a word *w* is fixated (after the first fixation) the probability that a word *w* will be read more than once
Context	
total regression-from duration *w-2* fixation probability *w-1* fixation probability *w+1* fixation probability *w+2* fixation probability *w-2* fixation duration *w-1* fixation duration *w+1* fixation duration *w+2* fixation duration	combined duration of the regressions that began at word *w* fixation probability of the word before the previous word fixation probability of the previous word fixation probability of the next word fixation probability of the word after the next word fixation duration of the word before the previous word fixation duration of the previous word fixation duration of the next word fixation duration of the word after the next word

Table 3: Gaze features extracted from the Dundee, GECO and ZuCo corpora.

annotator reliability analysis on 10,000 tokens (511 sentences) sampled from all three datasets yielded an agreement of 83.5% on the entity labels ($\kappa = 0.68$).

Table 2 shows the number of annotated entities in each dataset. The distribution of entities between the corpora is highly unbalanced: Dundee and ZuCo, the datasets containing more heterogeneous texts and thus, have a higher ratio of unique entity occurrences, versus GECO, a homogeneous corpus consisting of a single novel, where the named entities are very repetitive.

4 Eye-tracking features

The gaze data of all three corpora was recorded for multiple readers by conducting experiments in a controlled environment using specialized equipment. It is important to consider that, while we extract the same features for all corpora, there are certainly practical aspects that differ across the datasets. The following factors are expected to influence reading: experiment procedures; text presentation; recording hardware, software and quality; sampling rates; initial calibration and filtering, as well as *human factors* such as head movements and lack of attention. Therefore, separate normalization for each dataset should better preserve the signal within each corpus and for the same reason the type-aggregation was computed on the normalized feature values. This is especially relevant for the type-aggregated features and the cross-corpus experiments described below.

In order to add gaze information to the neural network, we have selected as many features as available from those present in all three corpora. Previous research shows benefits in combining multiple eye-tracking features of different stages of the human reading process (Barrett et al., 2016; Tokunaga et al., 2017).

The features extracted follow closely on Barrett et al. (2016). As described above, psycholinguistic research has shown how fixation duration and probability differ between word classes and syntactic comprehension processes. Thus, the features focus on representing these nuances as broadly as possible, covering the complete reading time of a word at different stages. Table 3 shows the eye movement features incorporated into the experiments. We split the 17 features into 4 distinct groups (analogous to Barrett et al. (2016)), which define the different stages of the reading process:

1. *BASIC* eye-tracking features capture characteristics on word-level, e.g. the number of all

fixations on a word or the probability that a word will be fixated (namely, the number of subjects who fixated the word divided by the total number of subjects).

2. *EARLY* gaze measures capture lexical access and early syntactic processing and are based on the first time a word is fixated.

3. *LATE* measures reflect the late syntactic processing and general disambiguation. These features are significant for words which were fixated more than once.

4. *CONTEXT* features capture the gaze measures of the surrounding tokens. These features consider the fixation probability and duration up to two tokens to the left and right of the current token. Additionally, regressions starting at the current word are also considered to be meaningful for the syntactic processing of full sentences.

The eye movement measurements were averaged over all native-speaking readers of each dataset to obtain more robust estimates. The small size of eye-tracking datasets often limits the potential for training data-intensive algorithms and causes overfitting in benchmark evaluation (Xu et al., 2015). It also leads to sparse samples of gaze measurements. Hence, given the limited number of observations available, we normalize the data by splitting the feature values into quantiles to avoid sparsity issues. The best results were achieved with 24 bins. This normalization is conducted separately for each corpus.

Moreover, special care had to be taken regarding tokenization, since the recorded eye-tracking data considers only whitespace separation. For example, the string *John's* would constitute a single token for eye-tracking feature extraction, but would be split into *John* and *'s* for NER, with the former token holding the label PERSON and the latter no label at all. Our strategy to address this issue was to assign the same values of the gaze features of the originating token to split tokens.

4.1 Type aggregation

Barrett and Søgaard (2015b) showed that type-level aggregation of gaze features results in larger improvements for part-of-speech tagging. Following their line of work, we also conducted experiments with type aggregation for NER. This implies that the eye-tracking feature values were averaged for each word type over all occurrences in the training data. For instance, the sum of the features of all n occurrences of the token "island" are averaged over the number of occurrences n. As a result, for each corpus as well as for the aggregated corpora, a lexicon of lower-cased word types with their averaged eye-tracking feature values was compiled. Thus, as input for the network, either the type-level aggregates for each individual corpus can be used or the values from the combined lexicon, which increases the number of word types with known gaze feature values.

The goal of type aggregation is twofold. First, it eliminates the requirement of eye-tracking features when applying the models at test time, since the larger the lexicon, the more tokens in the unseen data receive type-aggregated eye-tracking feature values. For those tokens not in the lexicon, we assign a placeholder for unknown feature values. Second, type-aggregated features can be used on any dataset and show that improvements can be achieved with aggregated gaze data without requiring large quantities of recorded data.

5 Model

The experiments in this work were executed using an enhanced version of the system presented by Lample et al. (2016). This hybrid approach is based on bidirectional LSTMs and conditional random fields and relies mainly on two sources of information: character-level and word-level representations.

For the experiments, the originally proposed values for all parameters were maintained. Specifically, the bidirectional LSTMs for character-based embeddings are trained on the corpus at hand with dimensions set to 25. The lookup table tor the word embeddings was initialized with the pre-trained GloVe vectors of 100 dimensions (Pennington et al., 2014). The model uses a single layer for the forward and backward LSTMs. All models were trained with a dropout rate at 0.5. Moreover, all digits were replaced with zeros.

The original model[1] was modified to include the gaze features as additional embedding layers to the network. The character-level representation, i.e. the output of a bidirectional LSTM, is concatenated with the word-level representation from

---

[1] https://github.com/glample/tagger

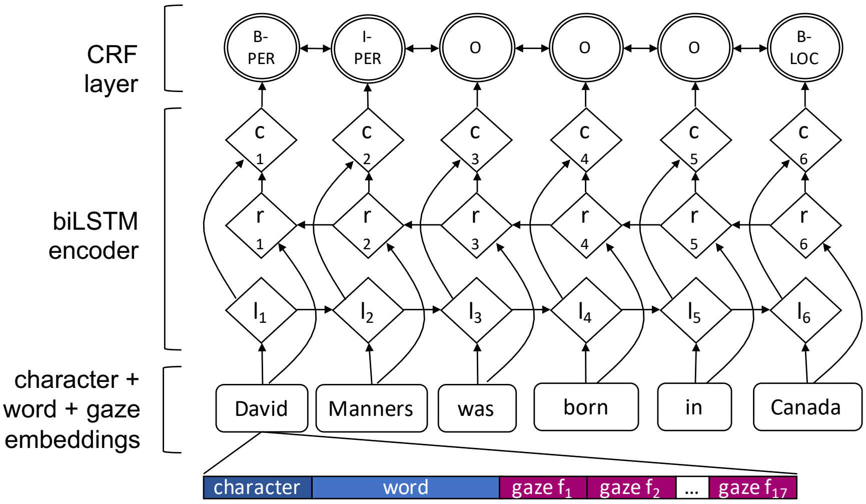


Figure 1: Main architecture of the network. Character and word embeddings concatenated with gaze features are given to a bidirectional LSTM. l_i represents the word *i* and its left context, r_i represents the word *i* and its right context. Concatenating these two vectors yields a representation of the word *i* in its context, c_i.

a word lookup table. In the augmented model with eye-tracking information, the embedding for each discrete gaze feature is also concatenated to the input. The dimension of the gaze feature embeddings is equal to the number of quantiles. This architecture is shown in Figure 1. Word length and word frequency are known to correlate and interact with gaze features (Tomanek et al., 2010), which is why we selected a base model that allows us to combine the eye-tracking features with word- and character-level information.

6 Results

Our main finding is that our models enhanced with gaze features consistently outperform the baseline. As our baseline, we trained and evaluated the original models with the neural architecture and parameters proposed by Lample et al. (2016) on the GECO, Dundee, and ZuCo corpora and compared it to the models that were enriched with eye-tracking measures. The best improvements on F_1-score over the baseline models are significant under one-sided t-tests ($p<0.05$).

All models were trained with 10-fold cross validation (80% training set, 10% development set, 10% test set) and early stopping was performed after 20 epochs of no improvement on the development set to reduce training time.

First, the performance on the individual datasets is tested, together with the performance of one combined dataset consisting of all three corpora (consisting of 142,441 tokens). In addition, we evaluate the effects of the type-aggregated features using individual type lexicons for each datasets, and combining the three type lexicons of each corpus. Finally, we experiment with cross-corpus scenarios to evaluate the potential of eye-tracking features in NER for domain adaptation. Both settings were also tested on an external corpus without eye-tracking features, namely the CoNLL-2003 dataset (Sang and De Meulder, 2003).

6.1 Individual dataset evaluation

First, we analyzed how augmenting the named entity recognition system with eye-tracking features affects the results on the individual datasets. Table 4 shows the improvements achieved by adding all 17 gaze features to the neural architecture, and training models on all three corpora, and on the combined dataset containing *all* sentences from the Dundee, GECO and ZuCo corpora. Noticeably, adding token-level gaze features improves the results on all datasets individually *and* combined, even on the GECO corpus, which yields a high baseline due to the homogeneity of the contained named entities (see Table 2).

Furthermore, Table 4 also presents the results of the NER models making use of the type-aggregated features instead of token-level gaze features. There are two different experiments for these type-level features: Using the features of the word types occurring in the corpus only, or using the aggregated features of all word types in the three corpora (as describe above). As can be seen, the performance of the different gaze fea-

	P	R	F
Dundee			
baseline	79.29	78.56	78.86
with gaze	79.55	79.27	79.35
type individual	**81.05**	**79.37**	**80.17***
type combined	80.27	79.26	79.67
Geco			
baseline	96.68	97.24	96.95
with gaze	**98.08**	**97.94**	**98.01***
type individual	97.72	97.42	97.57*
type combined	97.76	97.16	97.46*
ZuCo			
baseline	84.52	81.66	82.92
with gaze	**86.19**	**84.28**	**85.12***
type individual	84.21	82.61	83.30
type combined	83.26	83.37	83.31
All			
baseline	86.92	86.58	86.72
with gaze	88.72	89.39	89.03*
type combined	**89.04**	**89.52**	**89.26***

Table 4: Precision (P), recall (R) and F_1-score (F) for all models trained on individual datasets (best results in bold; * indicates statistically significant improvements on F_1-score). *With gaze* are models trained on the original eye-tracking features on token-level, *type individual* are the models trained on type-aggregated gaze features of this corpus only, while *type combined* are the models trained with type-aggregated features computed on all datasets.

ture levels varies between datasets, but both the original token-level features as well as the individual and combined type-level features achieve improvements over the baselines of all datasets.

To sum up, the largest improvement with eye-tracking features is achieved when combining all corpora into one larger dataset, where an additional 4% is gained in F_1-score by using type-aggregated features. Evidently, a larger mixed-domain dataset benefits from the type aggregation, while the original token-level gaze features achieve the best results on the individual datasets. Moreover, the additional gain when training on all datasets is due to the higher signal-to-noise ratio of type-aggregated features from multiple datasets.

Evaluation on CoNLL-2003 Going on step further, we evaluate the type-aggregated gaze features on an external corpus with no eye movement information available. The CoNLL-2003 corpus (Sang and De Meulder, 2003) has been widely used as a benchmark dataset for NER in different shared tasks. The English part of this corpus consists of Reuters news stories and contains 302,811 tokens in 22,137 sentences. We use this dataset as an additional corpus without gaze information. Only the type-aggregated features (based on the combined eye-tracking corpora) are added to each word. Merely 76% of the tokens in the CoNLL-2003 corpus also appear in the eye-tracking corpora described above and thus receive type-aggregated feature values. The rest of the tokens without aggregated gaze information available receive a placeholder for the unknown feature values.

CoNLL-2003	**P**	**R**	**F**
baseline	93.89	94.16	94.03
type combined	**94.38**	**94.32**	**94.35***

Table 5: Precision (P), recall (R) and F_1-score (F) for using type-aggregated gaze features on the CoNLL-2003 dataset (* marks statistically significant improvement).

Note that to avoid overfitting we do not train on the official train/test split of the CoNLL-2003 dataset, but perform 10-fold cross validation. Applying the same experiment setting, we train the augmented NER model with gaze features on the CoNLL-2003 data and compare it to a baseline model without any eye-tracking features. We achieve a minor, but nonetheless significant improvement (shown in Table 5), which strongly supports the generalizability effect of the type-aggregated features on unseen data.

6.2 Cross-dataset evaluation

In a second evaluation scenario, we test the potential of eye-tracking features for NER across corpora. The goal is to leverage eye-tracking features for domain adaptation. To show the robustness of our approach across domains, we train the models with token-level and type-level features on 100% of corpus A and a development set of 20% of corpus B and test on the remaining 80% of the corpus B, alternating only the development and the test set for each fold.

Table 6 shows the results of this cross-corpus evaluation. The impact of the eye-tracking features varies between the different combinations of datasets. However, the inclusion of eye-tracking features improves the results for all combinations, except for the models trained on the ZuCo corpus

		Dundee			GECO			ZuCo		
		P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Dundee	baseline				74.20	70.71	72.40	75.36	75.62	75.44
	token				75.68	71.54	73.55*	**78.85**	74.51	77.02
	type				**76.44**	**77.09**	**76.75***	78.33	**76.49**	**77.35**
GECO	baseline	58.91	34.91	43.80				68.88	42.49	52.38
	token	**59.61**	35.62	**44.53**				**69.18**	**44.22**	**53.81**
	type	58.39	**35.99**	44.44				67.69	42.36	52.01
ZuCo	baseline	65.85	**54.01**	59.34	83.00	**78.11**	**80.48**			
	token	**72.62**	50.76	59.70	82.92	75.35	78.91			
	type	69.21	53.05	**59.95**	**83.68**	74.57	78.85			

Table 6: Cross-corpus results: Precision (P), recall (R) and F_1-score (F) for all models trained on one dataset and tested on another (rows = training dataset; columns = test dataset; best results in bold; * indicates statistically significant improvements). The baseline models are trained without eye-tracking features, *token* models on the original eye-tracking features, and *type* are the models trained with type-aggregated features computed on all datasets.

and tested on the GECO corpus. Presumably, this is due to the combination of the small training data size of the ZuCo corpus and the homogeneity of the named entities in the GECO corpus.

CoNLL-2003	**P**	**R**	**F**
baseline	72.80	56.97	63.92
type combined	**74.56**	**60.20**	**66.61***

Table 7: Precision (P), recall (R) and F_1-score (F) for using type-aggregated gaze features trained on all three eye-tracking datasets and tested on the CoNLL-2003 dataset (* marks statistically significant improvement).

Evaluation on CoNLL-2003 Analogous to the individual dataset evaluation, we also test the potential of eye-tracking features in a cross-dataset scenario on an external benchmark dataset. Again, we use the CoNLL-2003 corpus for this purpose. We train a model on the Dundee, GECO *and* ZuCo corpora using type-aggregated eye-tracking features and test this model on the ConLL-2003 data. Table 7 shows that compared to a baseline without gaze features, the results improve by 3% F_1-score. These results underpin our hypothesis of the possibility of generalizing eye-tracking features on word type level, such that no recorded gaze data is required at test time.

7 Discussion

The models evaluated in the previous section show that eye-tracking data contain valuable semantic information that can be leveraged effectively by NER systems. While the individual datasets are still limited in size, the largest improvement is observed in the models making use of *all* the available data.

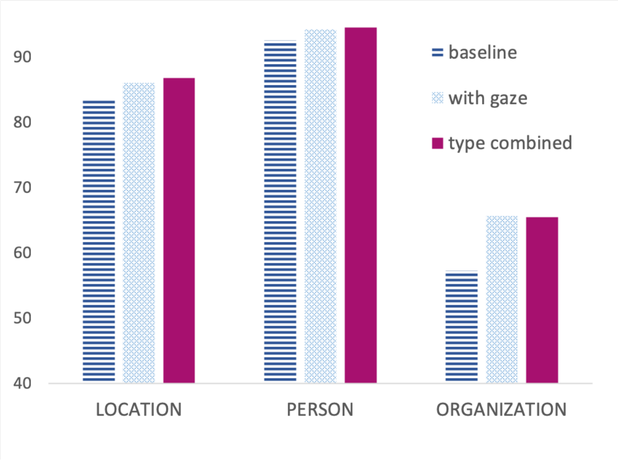


Figure 2: Results per class for the models trained on all gaze datasets combined.

At a closer look, the model leveraging gaze data yield a considerably higher increase in recall when comparing to the baselines. In addition, a class-wise analysis shows that the entity type benefiting the most from the gaze features over all models is ORGANIZATION, which is the most difficult class to predict. Figure 2 illustrates this with the results per class of the models trained on all three gaze corpora jointly.

In the individual dataset evaluation setting, the combined type-level feature aggregation from all datasets does not yield the best results, since each sentence in these corpora already has accurate eye-tracking features on toke-level. Thus, it is understandable that in this scenario the original gaze features and the gaze features aggregated only on the individual datasets result in better models. However, when evaluating the NER models in a cross-corpus scenario, the type-aggregated features lead to significant improvements.

Type aggregation evidently reduces the fine-grained nuances contained in eye-tracking information and eliminates the possibility of disambiguation between homographic tokens. Nevertheless, this type of disambiguation is not crucial for named entities, which mainly consist of proper nouns and the same entities tend to appear in the same context. Especially noteworthy is the gain in the models tested on the CoNLL-2003 benchmark corpus, which shows that aggregated eye-tracking features from other datasets can be applied to any unseen sentence and show improvements, even though more than 20% of the tokens have unknown gaze feature values. While the high number of unknown values is certainly a limitation of our approach, it shows at once the possibility of not requiring original gaze features at prediction time. Thus, the trained NER models can be applied robustly on unseen data.

8 Conclusion

We presented the first study of augmenting a NER system with eye-tracking information. Our results highlight the benefits of leveraging cognitive cues such as eye movements to improve entity recognition models. The manually annotated named entity labels for the three eye-tracking corpora are freely available. We augmented a neural NER architecture with gaze features. Experiments were performed using a wide range of features relevant to the human reading process and the results show significant improvements over the baseline for all corpora individually.

In addition, the type-aggregated gaze features are effective in cross-domain settings, even on an external benchmark corpus. The results of these type-aggregated features are a step towards leveraging eye-tracking data for information extraction at training time, without requiring real-time recorded eye-tracking data at prediction time.

References

Maria Barrett, Joachim Bingel, Nora Hollenstein, Marek Rei, and Anders Søgaard. 2018. Sequence classification with human attention. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 302–312.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 579–584.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. In *Proceedings of the 19th Conference on Computational Natural Language Learning*, pages 345–349.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*, pages 1–5.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Agatha Christie. 1920. *The Mysterious Affair at Styles*. Retrieved from Project Gutenberg, *www.gutenberg.org*.

Charles Clifton, Adrian Staub, and Keith Rayner. 2007. Eye movements in reading words and sentences. In *Eye Movements*, pages 341–371. Elsevier.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2017. Presenting GECO: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods*, 49(2):602–615.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 296–303.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Jose Gómez-Poveda and Elena Gaudioso. 2016. Evaluation of temporal stability of eye tracking algorithms using webcams. *Expert Systems with Applications*, 64:69–83.

Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. ZuCo, a simultaneous EEG and eye-tracking resource for natural sentence reading. *Scientific Data*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Barbara J Juhasz and Keith Rayner. 2003. Investigating the effects of a set of intercorrelated variables on eye fixation durations in reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(6):1312.

Marcel A Just and Patricia A Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The Dundee corpus. In *Proceedings of the 12th European Conference on Eye Movement*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Linguistic Data Consortium. 2005. ACE (Automatic Content Extraction) English annotation guidelines for entities. *Version*, 5(6):2005–08.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1:1064–1074.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Leveraging cognitive features for sentiment analysis. *Proceedings of The 20th Conference on Computational Natural Language Learning*, pages 156–166.

Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. 2016. WebGazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Keith Rayner. 1977. Visual attention in reading: Eye movements reflect cognitive processes. *Memory & Cognition*, 5(4):443–448.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372.

Omid Rohanian, Shiva Taslimipoor, Victoria Yaneva, and Le An Ha. 2017. Using gaze data to predict multiword expressions. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 601–609.

Javier San Agustin, Henrik Skovsgaard, John Paulin Hansen, and Dan Witzner Hansen. 2009. Low-cost gaze interaction: ready to deliver the promises. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4453–4458.

Erik F Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, volume 4, pages 142–147.

Weston Sewell and Oleg Komogortsev. 2010. Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3739–3744.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121.

Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine de Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 138–144.

Takenobu Tokunaga, Hitoshi Nishikawa, and Tomoya Iwakura. 2017. An eye-tracking study of named entity annotation. *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 758–764.

Katrin Tomanek, Udo Hahn, Steffen Lohmann, and Jürgen Ziegler. 2010. A cognitive cost model of annotations based on eye-tracking data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1158–1167.

Rihana Williams and Robin Morris. 2004. Eye movements, word familiarity, and vocabulary acquisition. *European Journal of Cognitive Psychology*, 16(1-2):312–339.

Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. 2015. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*.

The Emergence of Number and Syntax Units in LSTM Language Models

Yair Lakretz
Cognitive Neuroimaging Unit
NeuroSpin center
91191, Gif-sur-Yvette, France
`yair.lakretz@gmail.com`

German Kruszewski
Facebook AI Research
Paris, France
`germank@gmail.com`

Theo Desbordes
Facebook AI Research
Paris, France
`tdesbordes@fb.com`

Dieuwke Hupkes
ILLC, University of Amsterdam
Amsterdam, Netherlands
`d.hupkes@uva.nl`

Stanislas Dehaene
Cognitive Neuroimaging Unit
NeuroSpin center
91191, Gif-sur-Yvette, France
`stanislas.dehaene@gmail.com`

Marco Baroni
Facebook AI Research
Paris, France
`mbaroni@fb.com`

Abstract

Recent work has shown that LSTMs trained on a generic language modeling objective capture syntax-sensitive generalizations such as long-distance number agreement. We have however no mechanistic understanding of how they accomplish this remarkable feat. Some have conjectured it depends on heuristics that do not truly take hierarchical structure into account. We present here a detailed study of the inner mechanics of number tracking in LSTMs at the single neuron level. We discover that long-distance number information is largely managed by two "number units". Importantly, the behaviour of these units is partially controlled by other units independently shown to track syntactic structure. We conclude that LSTMs are, to some extent, implementing genuinely syntactic processing mechanisms, paving the way to a more general understanding of grammatical encoding in LSTMs.

1 Introduction

In the last years, recurrent neural networks (RNNs), and particularly long-short-term-memory (LSTM) architectures (Hochreiter and Schmidhuber, 1997), have been successfully applied to a variety of NLP tasks. This has spurred interest in whether these generic sequence-processing devices are discovering genuine structural properties of language in their training data, or whether their success can be explained by opportunistic surface-pattern-based heuristics.

Until now, this debate has mostly relied on "behavioural" evidence: The LSTM had been treated as a black box, and its capacities had been indirectly inferred by its performance on linguistic tasks. In this study, we took a complementary approach inspired by neuroscience: We thoroughly investigated the inner dynamics of an LSTM language model performing a number agreement task, striving to achieve a mechanistic understanding of how it accomplishes it. We found that the LSTM had specialized two "grandmother" cells (Bowers, 2009) to carry number features from the subject to the verb across the intervening material.[1] Interestingly, the LSTM also

---

[1] In the neuroscientific literature, "grandmother" cells are (sets of) neurons coding for specific information, e.g., about your grandmother, in a non-distributed manner.

Proceedings of NAACL-HLT 2019, pages 11–20
Minneapolis, Minnesota, June 2 - June 7, 2019.

possesses a more distributed mechanism to predict number when subject and verb are close, with the grandmother number cells only playing a crucial role in more difficult long-distance cases. Crucially, we independently identified a set of cells tracking syntactic structure, and found that one of them encodes the presence of an embedded phrase separating the main subject-verb dependency, and has strong efferent connections to the long-distance number cells, suggesting that the network relies on genuine syntactic information to regulate agreement-feature percolation.

Our analysis thus provides direct evidence for the claim that LSTMs trained on unannotated corpus data, despite lacking significant linguistic priors, learn to perform structure-dependent linguistic operations. In turn, this suggests that raw linguistic input and generic memory mechanisms, such as those implemented in LSTMs, may suffice to trigger the induction of non-trivial grammatical rules.

2 Related Work

Starting with the seminal work of Linzen et al. (2016), a long-distance number agreement task has emerged as a standard way to probe the syntactic capabilities of neural language models. In the number agreement task, a model is asked to predict the verb in a sentence where the subject and main verb are separated by one or more intervening nouns ("the **boy** near the *cars* **greets**...") and evaluated based on how often it predicts the right verb form.

Following mixed initial results by Linzen and colleagues and Bernardy and Lappin (2017), Gulordava et al. (2018) and Kuncoro et al. (2018b) have robustly established that LSTM language models achieve near-human performance on the agreement task. While Gulordava and colleagues provided some evidence that the LSTMs are relying on genuine syntactic generalizations, Kuncoro et al. (2018a) and Linzen and Leonard (2018) suggested that the LSTM achievements can, at least in part, be accounted by superficial heuristics (e.g., "percolate the number of the first noun in a sentence"). Other recent work has extended syntax probing to other phenomena such as negative polarity items and island constraints (Chowdhury and Zamparelli, 2018; Jumelet and Hupkes, 2018; Marvin and Linzen, 2018; Wilcox et al., 2018).

While Linzen et al. (2016) presented intriguing qualitative data showing cells that track grammatical number in a network directly trained on the agreement task, most of the following work focused on testing the network output behaviour, rather than on understanding how the latter follows from the inner representations of the network. Another research line studied linguistic processing in neural networks through 'diagnostic classifiers', that is, classifiers trained to predict a certain property from network activations (e.g., Gelderloos and Chrupała, 2016; Adi et al., 2017; Alain and Bengio, 2017; Hupkes et al., 2018). This approach may give insight into which information is encoded by the network in different layers or at different time points, but it only provides indirect evidence about the specific mechanics of linguistic processing in the network.

Other studies are closer to our approach in that they attempt to attribute function to specific network cells, often by means of visualization (Karpathy et al., 2016; Li et al., 2016; Tang et al., 2017). Radford et al. (2017), for example, detected a "sentiment" grandmother cell in a language-model-trained network. Kementchedjhieva and Lopez (2018) recently found a character-level RNN to track morpheme boundaries in a single cell. We are however not aware of others studies systematically characterizing the processing of a linguistic phenomenon at the level of RNN cell dynamics, as is the attempt in the study hereby presented.

3 Setup

Language Model We study the pretrained LSTM language model made available by Gulordava et al. (2018). This model is composed of a 650-dimensional embedding layer, two 650-dimensional hidden layers, and an output layer with vocabulary size 50,000. The model was trained on Wikipedia data, without fine-tuning for number agreement, and obtained perplexity close to state of the art in the experiments of Gulordava et al.[2]

Number-Agreement Tasks We complement analysis of the naturalistic, corpus-derived number-agreement test set of Linzen et al. (2016), in the version made available by Gulordava et al. (2018), with synthetically generated data-sets.

---

[2] Key findings reported below were also replicated with the same model trained with different initialization seeds and variations with different hyper-parameters.

Simple	the **boy** **greets** the guy
Adv	the **boy** probably **greets** the guy
2Adv	the **boy** most probably **greets** the guy
CoAdv	the **boy** openly and deliberately **greets** the guy
NamePP	the **boy** near Pat **greets** the guy
NounPP	the **boy** near the car **greets** the guy
NounPPAdv	the **boy** near the car kindly **greets** the guy

Table 1: NA tasks illustrated by representative singular sentences.

Each synthetic number-agreement task (NA-task) instantiates a fixed syntactic structure with varied lexical material, in order to probe subject-verb number agreement in controlled and increasingly challenging setups.[3] The different structures are illustrated in Table 1, where all forms are in the singular. Distinct sentences were randomly generated by selecting words from pools of 20 subject/object nouns, 15 verbs, 10 adverbs, 5 prepositions, 10 proper nouns and 10 location nouns. The items were selected so that their combination would not lead to semantic anomalies. For each NA-task, we generated singular and plural versions of each sentence. We refer to each such version as a *condition*. For NA-tasks that have other nouns occurring between subject and main verb, we also systematically vary their number, resulting in two *congruent* and two *incongruent* conditions. For example, the NounPP sentence in the table illustrates the congruent SS (singular-singular) condition and the corresponding sentence in the incongruent PS (plural-singular) condition is: "the *boys* near the *car greet* the guy". For all NA-tasks, each condition consisted of 600 sentences

Syntactic Depth Data-Set We probed the implicit syntax-parsing abilities of the model by testing whether its representations predict the syntactic depth of the words they process. Following Nelson et al. (2017), this was operationalized as predicting the number of open syntactic nodes at each word, given the canonical syntactic parse of a sentence. We generated a data-set of sentences with unambiguous but varied syntactic structures and annotated them with the number of open nodes at each word. For example: "Ten$_1$ really$_2$ ecstatic$_3$ cousins$_3$ of$_4$ four$_5$ teachers$_6$ are$_2$ quickly$_3$ laughing$_4$", where indexes show the corresponding number of open nodes. Since syntactic depth is naturally correlated with the position of a word in a sentence, we used a data-point sampling strategy to de-correlate these factors. For each length between 2 and 25 words, we randomly generated 300 sentences. From this set, we randomly picked examples uniformly covering all possible position-depth combinations within the 7-12 position and 3-8 depth ranges. The final data-set contains 4,033 positions from 1,303 sentences.[4]

4 Experiments

To successfully perform the NA-task, the LSTM should: (1) encode and store the grammatical number of the subject; and (2) track the main subject-verb syntactic dependency. The latter information is important for identifying the time period during which subject number should be stored, output and then updated by the network. This section describes the 'neural circuit' that encodes and processes this information in the LSTM.

4.1 Long-Range Number Units

We first tested the performance of the LSTM on the Linzen's data and on the NA-tasks in Table 1. Following Linzen et al. (2016) and later work, we computed the likelihood that the LSTM assigns to the main verb of each sentence given the preceding context and compared it to the likelihood it assigns to the wrong verb inflection. Accuracy in a given condition was measured as the proportion of sentences in this condition for which the model assigned a higher likelihood to the correct verb form than to the wrong one.

Network performance is reported in Table 2 (right column – 'Full'). We first note that our results on the Linzen NA-task confirm those reported in Gulordava et al. (2018). For the other NA-tasks, results show that some tasks and conditions are more difficult than others. For example, performance on the Simple (0-distance) NA-task is better than that on the Co-Adv NA-task, which in turn is better than that of the nounPP tasks. Second, as expected, incongruent conditions (the number-mismatch conditions of namePP, nounPP and nounPPAdv) reduce network performance.

---

[3] We exclude, for the time being, agreement across a relative clause, as it comes with the further complication of accounting for the extra agreement process taking place inside the relative clause.

[4] All our data-sets are available at: `https://github.com/FAIRNS/Number_and_syntax_units_in_LSTM_LMs`.

NA task	**C**	**Ablated**		**Full**
		776	**988**	
Simple	S	-	-	100
Adv	S	-	-	100
2Adv	S	-	-	99.9
CoAdv	S	-	82	98.7
namePP	SS	-	-	99.3
nounPP	SS	-	-	99.2
nounPP	SP	-	54.2	87.2
nounPPAdv	SS	-	-	99.5
nounPPAdv	SP	-	54.0	91.2
Simple	P	-	-	100
Adv	P	-	-	99.6
2Adv	P	-	-	99.3
CoAdv	P	79.2	-	99.3
namePP	PS	39.9	-	68.9
nounPP	PS	48.0	-	92.0
nounPP	PP	78.3	-	99.0
nounPPAdv	PS	63.7	-	99.2
nounPPAdv	PP	-	-	99.8
Linzen	**-**	75.3	-	93.9

Table 2: Ablation-experiments results: Percentage accuracy in all NA-tasks. Full: non-ablated model, C: condition, S: singular, P: plural. Pink (dark lines in B&W printing): plural subject, Light blue: singular subject. Performance reduction less than 10% is denoted by '-'.

Third, for long-range dependencies, reliably encoding singular subject across an interfering noun is more difficult than a plural subject: for both nounPP and nounPPAdv, PS is easier than SP. A possible explanation for this finding is that in English the plural form is almost always more frequent than the singular one, as the latter only marks third person singular, whereas the former is identical to the infinitive and other forms. Thus, if the network reverts to unigram probabilities, it will tend to prefer the plural.

Looking for Number Units Through Ablation Number information may be stored in the network in either a local, sparse, or a distributed way, depending on the fraction of active units that carry it. We hypothesized that if the network uses a local or sparse coding, meaning that there's a small set of units that encode number information, then ablating these units would lead to a drastic decrease in performance in the NA-tasks. To test this, we ablated each unit of the network, one at a time, by fixing its activation to zero, and tested on the NA-tasks.

Two units were found to have exceptional effect on network performance (Table 2, 776 and 988 columns).[5] Ablating them reduced network performance by more than 10% across various conditions, and, importantly, they were the only units whose ablation consistently brought network performance to around chance level in the more difficult incongruent conditions of the namePP, nounPP and nounPPAdv tasks.

Moreover, the ablation effect depended on the grammatical number of the subject: ablating 776 significantly reduced network performance only if the subject was plural (P, PS or PP conditions) and 988 only if the subject was singular (S, SP or SS conditions). In what follows, we will therefore refer to these units as the 'plural' and 'singular' units, respectively, or long-range (LR) number units when referring to both. Finally, we note that although the Linzen NA-task contained mixed stimuli from many types of conditions, the plural unit was found to have a substantial effect on average on network performance. The singular unit didn't show a similar effect in this case, which highlights the importance of using carefully crafted stimuli, as in the nounPP and nounPPAdv tasks, for understanding network dynamics. Taken together, these results suggest a highly local coding scheme of grammatical number when processing long-range dependencies.

Visualizing Gate and Cell-State Dynamics To understand the functioning of the number units, we now look into their gate and state dynamics during sentence processing. We focus on the nounPP NA-task, which is the simplest NA-task that includes a long-range dependency with an interfering noun, in both SP and PS conditions.

Recall the standard LSTM memory update and output rules (Hochreiter and Schmidhuber, 1997):

$$C_t = f_t \circ C_{t-1} + i_t \circ \widetilde{C}_t \qquad (1)$$

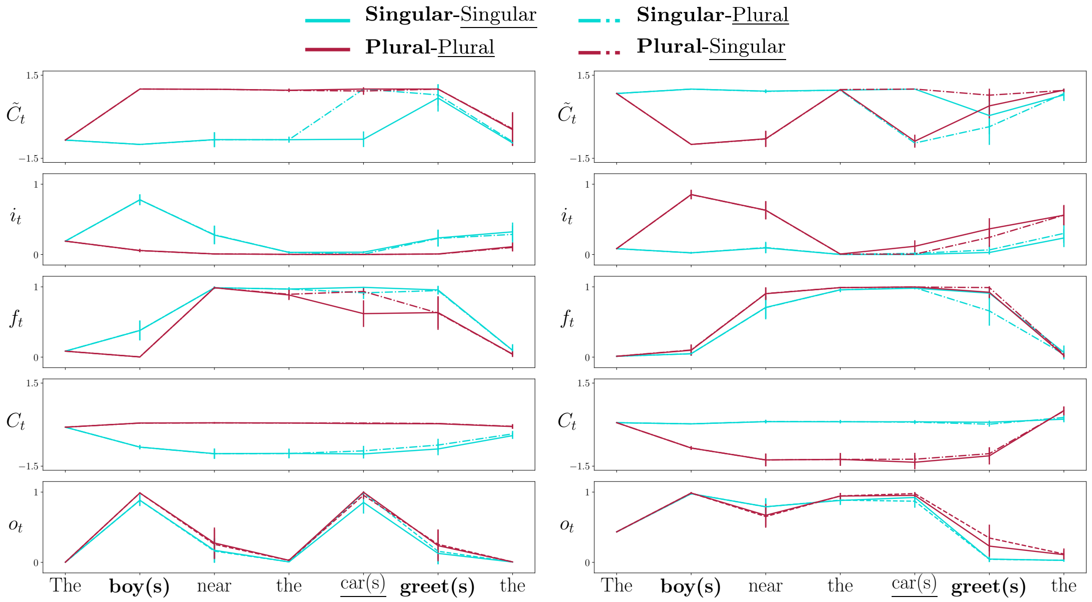
$$h_t = o_t \circ \tanh(C_t), \qquad (2)$$

where $f_t, i_t, o_t \in (0,1)$ are gating scalars computed by the network, and $\widetilde{C}_t \in (-1,1)$ is an update candidate for cell value.

Consider now how a number unit may reliably encode and store subject number across interfering nouns. Figure 1c exemplifies this for a singular

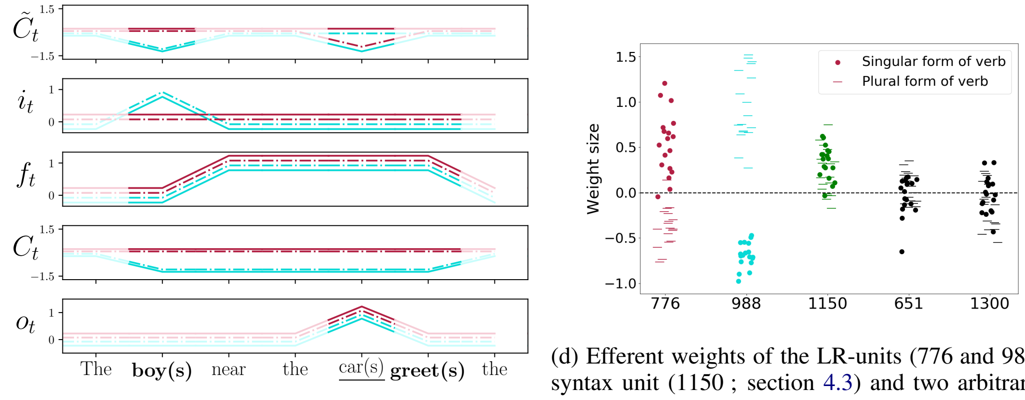
---

[5] Units 1-650 belong to the first layer, 651-1300 to the second. All units detected by our analyses come from the latter.



(a) 988 (singular)

(b) 776 (plural)



(c) Prediction (singular)

(d) Efferent weights of the LR-units (776 and 988), the syntax unit (1150 ; section 4.3) and two arbitrary units (651 and 1300).

Figure 1: (a) to (c) – Cell and gate activations during processing of sentences with a prepositional phrase between subject and verb. Values in (a) and (b) are averaged across all condition sentences, with error bars showing standard deviations. (d) – Efferent weights of specific units at the output layer to singular and plural verb forms.

unit, showing the desired gate and cell dynamics. The four conditions are represented with separated curves - pink for plural subject, light blue for singular, and dashed lines for incongruent conditions. Gate and cell activity at time points unrelated to solving the NA-task are masked with white, as we do not make precise predictions for them.

The update rule of the LSTM cell has two terms (Eq. 1).[6] In the first, $f_t \circ C_{t-1}$, the forget gate controls whether to keep the previous cell content ($f_t = 1$: perfect remembering) or forget it ($f_t = 0$: complete forgetting). In the second, $i_t \circ \tilde{C}_t$, the input gate controls whether the information currently presented to the network, as encoded by $\tilde{C}_t$, should be written onto the cell ($i_t = 1$: full access) or not ($i_t = 0$). The singular unit can thus use these gates to reliably store number information across long-range dependencies. Specifically, the unit can (enumeration follows the same order as the panels in Figure 1c): (1) encode subject number via $\tilde{C}_{t_{subject}}$ with different values for singular and plural; (2) open the input gate *only* when a singular subject is presented ($i_{t_{subject}} = 1$ in light-blue curves *only*) and protect it from interfering nouns ($i_t = 0, t_{subject} < t < t_{verb}$); (3) at the same time, clear the cell from previously stored information ($f_{t_{subject}} = 0$) and then

---

[6] We abuse notation here, using the symbols denoting whole layers in equations (1) and (2) to denote the components of single cells.

store subject number across the entire dependency ($f_t = 1, t_{subject} < t < t_{verb}$); (4) this will result in stable encoding of subject number in the cell C_t throughout the dependency; (5) finally, output subject number at the right moment, when predicting the verb form ($o_{t_{verb}-1} = 1$) (Eq. 2).

Figures 1a and 1b present the actual gate and cell dynamics of the singular and plural units. Both units follow the general solution for reliable number storage described above. Note that for $\tilde{C}_t$ and i_t, and as a result also for C_t, the plural unit 'mirrors' the singular unit with respect to subject number (pink curves of PP and PS vs. Light blue curves of SS and SP). This is in accordance with the results of the ablation experiments, which showed that ablating these units had an effect that depended on the grammatical number of the subject (Table 2). This provides complementary support for the identification of these units as 'singular' and 'plural'.

A single divergence between the solution depicted in Figure 1c and the actual dynamics of the number units is that input gate activity is smaller, but not zero, at the time step immediately following the subject. One speculative explanation is that this might be useful to process compound nouns. In these cases, subject number information is stored with the second noun, whereas in the case of simple nouns there is no 'risk' of encountering an interfering noun immediately after the subject, making the delay in closing the gate safe.

The singular and plural units had emerged at the second layer of the network. This seems appropriate since number information needs to be directly projected to the output layer for correct verb-form prediction. Moreover, number-unit output should be projected differently to singular and plural verb forms in the output layer, only increasing activity in output units representing the suitable form. For example, for the singular unit, since singular subjects are encoded with a negative value ($C_{t_{verb}-1} < -1$ in figure 1a), the more negative its efferent weights to singular verb forms in the output layer, the higher the probabilities of these verb forms would be. Figure 1d shows the efferent weights of the LR-number units to all verbs in our data-sets. We found that, indeed, the efferent weights to the singular and plural verb forms are segregated from each other, with weight signs that correspond to the negative encoding of subject number used by both singular and plural units.

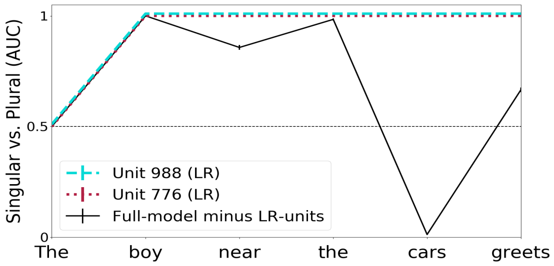


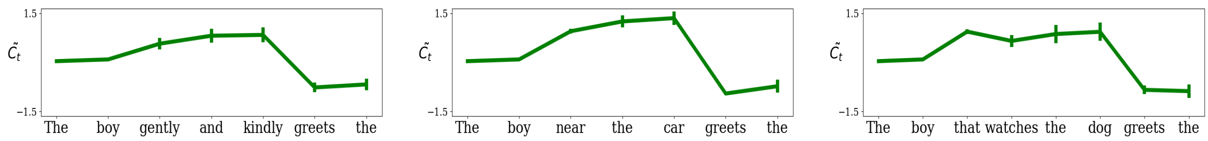
Figure 2: Generalization across time of subject-number prediction. Error bars represent standard deviations across cross-validation splits.

Two other arbitrary units, 651 and 1300 , and the syntax unit 1150 to be described below (Section 4.3) do not have segregated efferent weights to verb forms, as expected.

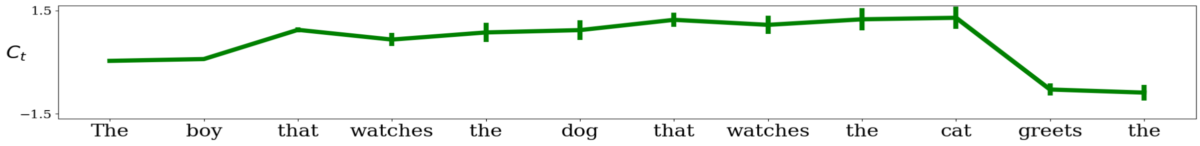
4.2 Short-Range Number Information

Performance on the easier NA-tasks (Simple, Adv, 2Adv) was not impaired by single-unit ablations. This suggests that number may be encoded also elsewhere in the network, perhaps via a more distributed code. To verify this, we tested whether subject number can be decoded from the whole pattern of activities in the network (excluding the two LR-number units) and whether this decoding is stable across time (see Giulianelli et al., 2018, for similar observations and related methods). We expected this distributed activity to track number in a small time window after the subject, but, unlike the LR-number units, to be affected by incongruent intervening nouns.

We trained a linear model to predict the grammatical number of the subject from network activity in response to the presentation of the subject, and tested its prediction on test sets from all time points (King and Dehaene, 2014), in incongruent conditions only of the nounPP task. We used Area under of Curve (AUC) to evaluate model performance. Figure 2 shows decoding across time of subject number from cell activity of each number unit separately and from cell activity of the entire network without these two units ('Full model minus LR-units'). Results show that number information can be efficiently decoded from other units in the network, and that this information can be carried for several time steps (relatively high AUC up to the second determiner). However, the way in which these units encode number is sensitive to the last encountered noun, with AUC decreasing



(a) 2Adv (b) nounPP (c) subject relative



(d) Two embeddings with subject relatives

Figure 3: Cell activity of syntax unit 1150 while processing various syntactic structures. Values averaged across all stimuli in an NA-task, with error bars representing standard deviations. Relative clause NA-task stimuli were specifically generated for this visualization.

to zero around the second noun (‘cars’), whereas test performance of the models trained on cell activity of the LR-number units is consistently high. This confirms that number prediction is supported both by the LR-number units, and by distributed activation patterns of other short-range (SR) number units. The latter, however, are not syntax-sensitive, and simply encode the number of the last noun encountered.

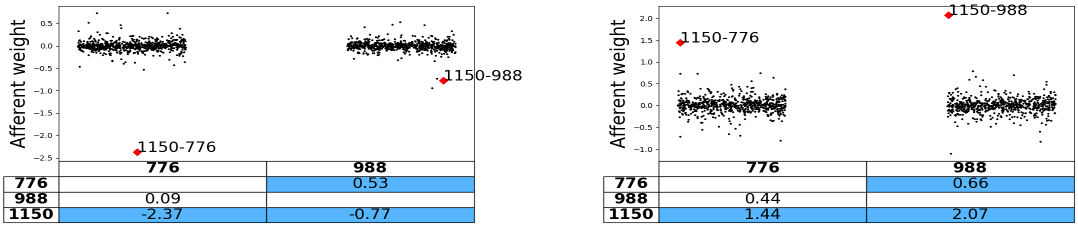
A full description of the SR-number units is beyond our scope. However, we note that 10 SR-number units in the second layer of the network were identified, which had efferent weights with a similar segregated structure as that of the LR units (Figure 1d). These units were indeed sensitive to the last encountered noun: subject number could be decoded from single-unit cell activity during its presentation (AUC$>$ 0.9), but activity ‘swaps’ once an interfering noun appears (i.e., AUC decreases to zero in a generalization-across-time analysis). Finally, to validate the role of SR-number units in encoding number for easier NA-tasks, we ablated both SR and LR number units (12 in total) or SR units only (10 in total) and evaluated network performance on these NA-tasks. Both experiments resulted in a significant reduction in task performance compared to 1,000 random equi-size ablations ($p < 0.01$ in all ‘easier’ tasks).

Intriguingly, we observed qualitatively that LR units are almost always making the right prediction, even when the network predicts the wrong number. The wrong outcome, in such cases, might be due to interference from the syntax-insensitive SR units. We leave the study of LR-SR unit interplay to future work.

4.3 Syntax Units

We saw how the input and forget gates of the LR-number units control the flow of subject-number information. It remains unclear, however, how the dynamics of these gates are controlled by the network. We hypothesized that other units in the network may encode information about the syntactic structure of the sentence, and thus about the subject-verb dependency. These units could then control and coordinate the opening and closing of the input and forget gates of the number units.

To identify such ’syntax’ units, we tested from which units syntactic information can be efficiently decoded. We used depth of the syntactic tree as a proxy for syntactic structure (Nelson et al., 2017) and trained an L2-regularized regression model to predict syntactic tree-depth from the hidden-state activity of all units. In all experiments, we used the data presented in Section 3 above and performed a nested 5-fold cross-validation procedure. Word frequency, which was added as a covariate to the model, had a negligible effect on the results. Syntactic tree-depth was found to be efficiently decodable from network activity ($R^2_{test-set} = 0.85 \pm 0.009$; covariate-corrected). A small subset of ‘syntax’ units had relatively high weights in the regression model (mean weight = 7.6×10^{-4}, SD=7.86×10^{-2}; cutoff for outlier weights was set to three SDs). Since the interpretation of the regression weights may depend on possible correlations among the features, we also tested the causal effect of these units on NA-task performance. Ablating the syntax units together resulted in significant performance reduction in NA-tasks that have an interfering noun: Linzen NA-task: $p = 0.024$, nounPPAdv-



(a) Input gate

(b) Forget gate

Figure 4: Connectivity among the syntax unit 1150 and LR-number units 776 and 988 . Projecting units are on the table rows. Blue background highlights outlier values ($|z - score| > 3$). Weights from the syntax unit are marked with large diamond markers and are explicitly labeled in the plots.

SP: $p = 0.011$, nounPPAdv-PS: $p = 0.034$, nounPP-SP: $p < 0.001$ and marginally significant in nounPP-PS: $p = 0.052$ (compared to 1000 random ablations of subsets of units of the same size).

To gain further insight regarding the functioning of the syntax units, we next visualized their gate and cell dynamics during sentence processing. We found that cell activity of unit 1150 , which also had one of the highest weights in the regression model, was remarkably structured. The activity of this unit increases across the entire subject-verb dependency and drops abruptly right after. Figures 3a and 3b show cell activity of this unit during the processing of stimuli from the 2Adv and nounPP tasks. We found the same dynamics in cases where another verb occurs between subject and main verb, as in subject relatives (Figure 3c), and in exceptionally long-distance dependencies with two interfering nouns and verbs (Figure 3d). Taken together, these results suggest that unit 1150 consistently encodes subject-verb dependencies in a syntax-sensitive manner. Other syntax units did not show an easily interpretable dynamics and had no clear interactions with the number units in the analysis discussed next. This suggests that they perform different syntactic, or possibly other, functions.

4.4 Syntax-Number Units Connections

We finally look at the connections that were learned by the LSTM between syntax unit 1150 , which appears to be more closely involved in tracking subject-verb agreement, and the LR number units, as well as at the connections between the LR-number units themselves. For each unit pair, there are 4 connection types, one for each component of the target cell (to the 3 gates and to the update candidate). We focus on input and forget gates, as they control the flow and storage of number information.

Figures 4a and 4b show the distributions of all afferent recurrent weights to the input and forget gates of the LR-number units, scaled by the maximal activity h_t of the pre-synaptic units during the nounPP task (this scaling evaluates the *effective* input to the units and did not change the conclusions described below). We found that the weights from the syntax unit to the forget gate of both 776 and 988 are exceptionally high in the positive direction compared to all other afferent connections in the network ($z - score = 8.1, 11.2$, respectively) and those to their input gates exceptionally negative ($z - score = -16.2, -7.2$). Since the cell activity of syntax unit 1150 is positive across the entire subject-verb dependency (e.g., Figure 3d), the connectivity from the syntax unit drives the number unit forget gates towards one ($W^f_{776,1150}h^{1150} \gg 0$ and $W^f_{988,1150}h^{1150} \gg 0$; $t_{subject} < t < t_{verb}$) and their input gates towards zero ($W^i_{776,1150}h^{1150} \ll 0$ and $W^i_{988,1150}h^{1150} \ll 0$). Looking at the right-hand-side of Eq. (1), this means that the first term becomes dominant and the second vanishes, suggesting that, across the entire dependency, the syntax unit conveys a 'remember flag' to the number units. Similarly, when the activity of the syntax unit becomes negative at the end of the dependency, it conveys an 'update flag'.

Last, we note that the reciprocal connectivity between the two LR-number units is always positive, to both input and forget gates (with $|z - score| > 3$ for the 776 -to-988 direction). Since their activity is negative throughout the subject-verb dependency (Figures 1a and 1b), this means that they are *mutually inhibiting*, thus steering towards an unequivocal signal about the grammatical number of the subject to the output layer.

5 Summary and Discussion

We provided the first detailed description of the underlying mechanism by which an LSTM language-model performs long-distance number agreement. Strikingly, simply training an LSTM on a language-model objective on raw corpus data brought about single units carrying exceptionally specific linguistic information. Three of these units were found to form a highly interactive local network, which makes up the central part of a 'neural' circuit performing long-distance number agreement.

One of these units encodes and stores grammatical number information when the main subject of a sentence is singular, and it successfully carries this information across long-range dependencies. Another unit similarly encodes plurality. These number units show that a highly local encoding of linguistic features can emerge in LSTMs during language-model training, as was previously suggested by theoretical studies of artificial neural networks (e.g., Bowers, 2009) and in neuroscience (e.g., Kutter et al., 2018).

Our analysis also identified units whose activity correlates with syntactic complexity. These units, as a whole, affect performance on the agreement tasks. We further found that one of them encodes the main subject-verb dependency across various syntactic constructions. Moreover, the highest afferent weights to the forget and input gates of both LR-number units were from this unit. A natural interpretation is that this unit propagates syntax-based remember and update flags that control when the number units store and release information.

Finally, number is also redundantly encoded in a more distributed way, but the latter mechanism is unable to carry information across embedded syntactic structures. The computational burden of tracking number information thus gave rise to two types of units in the network, encoding similar information with distinct properties and dynamics.

The relationship we uncovered and characterized between syntax and number units suggests that agreement in an LSTM language-model cannot be entirely explained away by superficial heuristics, and the networks have, to some extent, learned to build and exploit structure-based syntactic representations, akin to those conjectured to support human-sentence processing.

In future work, we intend to explore how the encoding pattern we found varies across network architectures and hyperparameters, as well as across languages and domains. We also would like to investigate the timecourse of emergence of the found behaviour over training time.

More generally, we hope that our study will inspire more analyses of the inner dynamics of LSTMs and other sequence-processing networks, complementing the currently popular "black-box probing" approach. Besides bringing about a mechanistic understanding of language processing in artificial models, this could inform work on human-sentence processing. Indeed, our study yields particular testable predictions on brain dynamics, given that the computational burden of long-distance agreement remains the same for artificial and biological neural network, despite implementation differences and different data sizes required for language acquisition. We conjecture a similar distinction between SR and LR units to be found in the human brain, as well as an interaction between syntax-processing and feature-carrying units such as the LR units, and plan to test these in future work.

Acknowledgments

We would like to thank Kristina Gulordava, Jean-Remi King, Tal Linzen, Gabriella Vigliocco and Christophe Pallier for helpful feedback and comments on the work.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of ICLR Conference Track*, Toulon, France. Published online: `https://openreview.net/group?id=ICLR.cc/2017/conference`.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *Proceedings of ICLR Conference Track*, Toulon, France.

Jean-Philippe Bernardy and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *Linguistic Issues in Language Technology*, 15(2):1–15.

Jeffrey Bowers. 2009. On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. *Psychological Review*, 116(1):220–251.

Shammur Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of COLING*, pages 133–144, Santa Fe, NM.

Lieke Gelderloos and Grzegorz Chrupała. 2016. From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1309–1319.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the EMNLP BlackboxNLP Workshop*, pages 240–248, Brussels, Belgium.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL*, pages 1195–1205, New Orleans, LA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? on the ability of lstms to understand negative polarity items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *Proceedings of ICLR Workshop Track*, San Juan, Puerto Rico. Published online: `https://openreview.net/group?id=ICLR.cc/2016/workshop`.

Yova Kementchedjhieva and Adam Lopez. 2018. 'Indicatements' that character language models learn English morpho-syntactic units and regularities. In *Proceedings of the EMNLP Workshop on analyzing and interpreting neural networks for NLP*, Brussels, Belgium. In press.

Jean-Rémi King and Stanislas Dehaene. 2014. Characterizing the dynamics of mental representations: The temporal generalization method. *Trends in Cognitive Sciences*, 18(4):203–210.

Adhiguna Kuncoro, Chris Dyer, John Hale, and Phil Blunsom. 2018a. The perils of natural behavioral tests for unnatural models: The case of number agreement. Poster presented at the Learning Language in Humans and in Machines conference, online at: `https://osf.io/view/L2HM/`.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018b. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of ACL*, pages 1426–1436, Melbourne, Australia.

Esther Kutter, Jan Bostroem, Christian Elger, Florian Mormann, and Andreas Nieder. 2018. Single neurons in the human brain encode numbers. *Neuron*, 100(3):753–761.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 681–691.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Tal Linzen and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *Proceedings of CogSci*, pages 692–697, Austin, TX.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Matthew Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney Cash, Lionel Naccache, John Hale, Christophe Pallier, and Stanislas Dehaene. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences*, 114(18):E3669–E3678.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. `https://arxiv.org/abs/1704.01444`.

Zhiyuan Tang, Ying Shi, Dong Wang, Yang Feng, and Shiyue Zhang. 2017. Memory visualization for gated recurrent neural networks in speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2736–2740. IEEE.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221.

Neural Self-Training through Spaced Repetition

Hadi Amiri
Department of Biomedical Informatics, Harvard
10 Shattuck Street, Boston, MA 02115
`hadi@hms.harvard.edu`

Abstract

Self-training is a semi-supervised learning approach for utilizing unlabeled data to create better learners. The efficacy of self-training algorithms depends on their data sampling techniques. The majority of current sampling techniques are based on predetermined policies which may not effectively explore the data space or improve model generalizability. In this work, we tackle the above challenges by introducing a new data sampling technique based on *spaced repetition* that dynamically samples informative and diverse unlabeled instances with respect to individual learner and instance characteristics. The proposed model is specifically effective in the context of neural models which can suffer from overfitting and high-variance gradients when trained with small amount of labeled data. Our model outperforms current semi-supervised learning approaches developed for neural networks on publicly-available datasets.

1 Introduction

It is often expensive or time-consuming to obtain labeled data for Natural Language Processing tasks. In addition, manually-labeled datasets may not contain enough samples for downstream data analysis or novelty detection (Wang and Hebert, 2016). To tackle these issues, semi-supervised learning (Zhu, 2006; Chapelle et al., 2009) has become an important topic when one has access to small amount of labeled data and large amount of unlabeled data.

Self-training is a type of semi-supervised learning in which a downstream learner (e.g. a classifier) is first trained with labeled data, then the trained model is applied to unlabeled data to generate more labeled instances. A select sample of these instances together with their *pseudo* (predicted) labels are added to the labeled data and the learner is re-trained using the new labeled dataset. This process repeats until there is no more unlabeled data left or no improvement is observed in model performance on validation data (Zhu, 2006; Zhu and Goldberg, 2009).

Conventional self-training methods often rely on prediction confidence of their learners to sample unlabeled data. Typically the most confident unlabeled instances are selected (HEARST, 1991; Yarowsky, 1995; Riloff and Jones, 1999; Zhou et al., 2012). This strategy often causes only those unlabeled instances that match well with the current model being selected during self-training, therefore, the model may fail to best generalize to complete sample space (Zhang and Rudnicky, 2006; Wu et al., 2018). Ideally, a self-training algorithm should explore the space thoroughly for better generalization and higher performance. Recently Wu et al. (2018) developed an effective data sampling technique for "co-training" (Blum and Mitchell, 1998) methods which require two distinct views of data. Although effective, this model can't be readily applied to some text datasets due to the two distinct view requirement.

In the context of neural networks, *pretraining* is an effective semi-supervised approach in which layers of a network are first pretrained by learning to reconstruct their inputs, and then network parameters are optimized by supervised fine-tuning on a target task (Hinton and Salakhutdinov, 2006; Bengio et al., 2007; Erhan et al., 2010). While pretraining has been effective in neural language modeling and document classification (Dai and Le, 2015; Miyato et al., 2016), it has an inherent limitation: the *same* neural model or parts thereof must be used in both pretraining and fine-tuning steps. This poses a major limitation on the design choices as some pretraining tasks may need to exploit several data types (e.g., speech and text), or might require deeper network architectures.

Proceedings of NAACL-HLT 2019, pages 21–31
Minneapolis, Minnesota, June 2 - June 7, 2019.

The above challenges and intuitions inspire our work on developing a novel approach for neural self-training. The core part of our approach is a data sampling policy which is inspired by findings in cognitive psychology about spaced repetition (Dempster, 1989; Cepeda et al., 2006; Averell and Heathcote, 2011); the phenomenon in which a learner (often a human) can learn efficiently and effectively by accurately scheduling reviews of learning materials. In contrast to previous self-training approaches, our spaced repetition-based data sampling policy is not predetermined, explores the entire data space, and dynamically selects unlabeled instances with respect to the "strength" of a downstream learner on a target task, and "easiness" of unlabeled instances. In addition, our model relaxes the "same model" constraint of pretraining-based approaches by naturally decoupling pretraining and fine-tuning models through spaced repetition.

The contributions of this paper are (a): we propose an effective formulation of spaced repetition for self-training methods; to the best of our knowledge, this is the first work that investigates spaced repetition for semi-supervised learning, (b): our approach dynamically samples data, is not limited to predetermined sampling strategies, and naturally decouples pretraining and fine-tuning models, and (c): it outperforms current state-of-the-art baselines on large-scale datasets.

Our best model outperforms standard and current state-of-the-art semi-supervised learning methods by 6.5 and 4.1 points improvement in macro-F1 on sentiment classification task, and 3.6 and 2.2 points on churn classification task. Further analyses show that the performance gain is due to our model's ability in sampling *diverse* and *informative* unlabeled instances (those that are different from training data and can improve model generalizability).

2 Method

Conventional self-training methods employ the following steps to utilize unlabeled data for semi-supervised learning: (1) train a learner, e.g. a classifier, using labeled data, (2) iteratively select unlabeled instances based on a data sampling technique, and add the sampled instances (together with their predicted *pseudo* labels) to the labeled data, and (3) iteratively update the learner using the new labeled dataset.

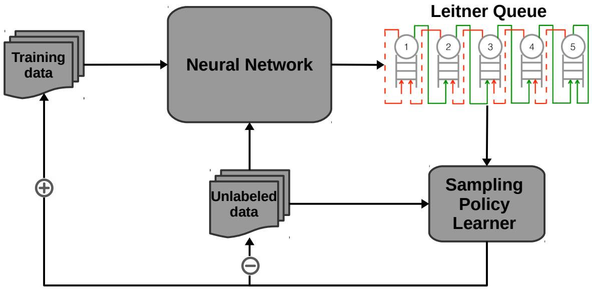


Figure 1: Neural Self-training Framework: at every self-training episode, the network uses current labeled data to iteratively optimize its parameters against a target task, and dynamically explores unlabeled data space through spaced repetition (specifically Leitner queue) to inform a data sampler that selects unlabeled data for the next self-training episode. Dashed/Red and solid/green arrows in Leitner queue indicate instance movements among queues.

The core difference between self-training algorithms is in the second step: data sampling policy. In this paper, we develop a new data sampling technique based on "spaced repetition" which dynamically explores the data space and takes into account instance and learner characteristics (such as easiness of instances or learner strength on target task) to sample unlabeled data for effective self-training.

Figure 1 illustrates our proposed neural self-training framework. We assume the downstream learner is a neural network that, at every self-training episode, (a): takes *current* labeled and unlabeled data as input, (b): uses labeled data to iteratively optimize its parameters with respect to a target task, and (c): dynamically explores unlabeled data space through spaced repetition to inform a data sampler that selects unlabeled data for the next self-training episode.

2.1 Spaced Repetition

Spaced repetition (Dempster, 1989; Cepeda et al., 2006; Averell and Heathcote, 2011) was presented in psychology and forms the building block of many educational devices, including flashcards, in which small pieces of information are repeatedly presented to a learner on a schedule determined by a spaced repetition algorithm. Such algorithms show that humans and machines can better learn by scheduling reviews of materials so that more time is spent on difficult concepts and less time on easier ones (Dempster, 1989; Novikoff et al., 2012; Amiri et al., 2017).

In this paper, we focus on a specific spaced repetition framework called Leitner system (Leitner, 1974). Suppose we have n queues $\{q_0, \ldots, q_{n-1}\}$. In general, Leitner system initially places all instances in the first queue, q_0. During training, if an instance from q_i is correctly classified by the learner, it will be "promoted" to q_{i+1} (solid/green arrows in Figure 1), otherwise it will be "demoted" to the previous queue, q_{i-1} (dashed/red arrows in Figure 1). Therefore, as the learner trains through time, higher queues will accumulate instances that are easier for the learner, while lower queues will accumulate harder instances.

To use Leitner system for neural self-training, we assume our learner is a neural network, place all *unlabeled* instances in the first queue of Leitner system (line 2 in Algorithm 1), and gradually populate them to other queues while training the network. Our Leitner system uses iteration-specific network predictions on unlabeled instances and current pseudo labels of these instances to move them between queues (see line 4-5 in Algorithm 1); pseudo labels can be obtained through posterior predictions generated by any trained downstream learner (see Section 2.2). Instances with similar class predictions and pseudo labels will be promoted to their next queues, and those with opposite predictions and labels will be demoted to lower queues. We note that, errors (e.g. inaccurate pseudo labels or network predictions) can inversely affect instance movements among queues. However, our sampling technique (see below) alleviates this issue because such misleading instances, if sampled, can't improve the generalizability of downstream learners. Details of our Leitner system is shown in Table 1.

Algorithm 1. Leitner system

```
Input:
  $\mathbf{L}, \mathbf{U}, \mathbf{V}$ : labeled, unlabeled, and validation data
  $\mathbf{y}$ : pseudo labels for $\mathbf{U}$
  $k$ : number of training epochs
  $n$ : number of queues
Output:
  $Q$: Leitner queue populated with $\mathbf{U}$

1  $Q = [q_0, q_1, \ldots, q_{n-1}]$
2  $q_0 = [\mathbf{U}], q_i = []$ for $i \in [1, n-1]$
3  for $epoch = 1$ to $k$:
4    $model = \texttt{epoch\_train}(\mathbf{L}, \mathbf{V})$
5    $promos, demos = \texttt{eval}(Q, \mathbf{y}, model)$
6    $Q = \texttt{schedule}(Q, promos, demos)$
7  end for
8  return Q
```

Table 1: Leitner system for neural self-training. All *unlabeled* instances are initially placed in the first queue and then populated to other queues depending on their easiness and learner (network) performance. `epoch_train(.)` uses training data to train the network for a single epoch and returns a trained model, `eval(.)` applies the current model on unlabeled instances in all queues and, based on given pseudo labels (treated as gold labels), returns lists of correctly and incorrectly classified instances, $promos$ and $demos$ respectively, and `schedule(.)` moves $promos$ and $demos$ instances to their next and previous queues respectively, and returns the updated queue.

2.2 Self-Training with Leitner Queues

We formulate the data sampling process as a decision-making problem where, at every self-training episode, the decision is to select a subset of unlabeled instances for self-training using information from Leitner queues. A simple, yet effective, approach to utilize such information is a greedy one in which instances of the queue that most improves the performance of the current model on validation data will be selected. We refer to this queue as *designated* queue:

Algorithm 2 shows details of our self-training approach. At every episode, we use current labeled data to train a task-specific neural network (line 2). Here, we weight the loss function using class size to deal with imbalanced data, and weight pseudo-labeled instances (as a function of episodes) to alleviate the effect of potentially wrong pseudo labels while training the network. We then use the trained network to generate pseudo labels for current unlabeled instances (line 3). These instances are then populated in Leitner queues as described before (line 4). Given the populated Leitner queues, the sample for current self-training episode is then created using instances of the designated queue, the queue that most improves the performance of the current network on validation data (lines 5-8). Instances of the designated queue will be removed from unlabeled data and added to labeled data with their pseudo labels treated as gold labels (lines 9-10).

We note that finding designated queues (lines 5-8 in Algorithm 2) imposes computational complexity on our model. However, in practice, we observe that designated queues are almost always among middle or higher queues in Leitner system, i.e. $q_i, \forall i \in [\lfloor n/2 \rfloor, n-1]$ where n in the number of queues. This can help accelerating the search

Algorithm 2. Neural Self-training

```
Input:
  $\mathbf{L}, \mathbf{U}, \mathbf{V}$ : labeled, unlabeled, and validation data
  $K$ : number of self-training episodes
Output:
  $M$: classification model

for $episode = 1$ to $K$:
  $M_L = \texttt{train}(\mathbf{L}, \mathbf{V})$
  $\mathbf{y} = \texttt{label}(M_L, \mathbf{U})$
  $Q = \texttt{Leitner\_system}(\mathbf{L}, \mathbf{U}, \mathbf{V}, \mathbf{y})$ \\Alg. 1
  for $q$ in $Q$:
    $M_q = \texttt{train}(\mathbf{L} + [q, \mathbf{y}[q]], \mathbf{V})$
  end for
  $M, q_{desig} = \texttt{get\_best}(M_Q, M_L)$
  $\mathbf{L} = \mathbf{L} + [q_{desig}, \mathbf{y}[q_{desig}]]$
  $\mathbf{U} = \mathbf{U} - q_{desig}$
end for
return $M$
```

Table 2: Proposed neural self-training framework. `train`(.) uses current labeled data to train the network and returns a trained model, `label`(.) generates pseudo labels for unlabeled instances using the trained model, `Leitner_system`(.) populates current unlabeled instances in Leitner queue, and `get_best`(.) compares performance of given models on validation data and returns the best model in conjunction with the queue that leads to the best performance, if any. Instances of the designated queue will be removed from unlabeled data and added to labeled data with their pseudo labels treated as gold labels.

process. In addition, learning a data sampling policy from movement patterns of instances among queues may help alleviating/eliminating the need for such an iterative search; see Section 4.4.

Finally, at test time, we apply the resulting self-trained network to test data and use the result for model comparison.

3 Experiments

We compare different self-training approaches in two settings where learners (neural networks) have low or high performance on original labeled data. This consideration helps investigating sensitivity of different self-training algorithms to the initial performance of learners.

3.1 Datasets and Evaluation Metric

As datasets, we use movie reviews from IMDb and short microblog posts from Twitter. These datasets and their corresponding tasks are described below and their statistics are provided in Table 3. In terms of preprocessing, we change all texts to lowercase, and remove stop words, user names, and URLs from texts in these datasets:

	Train	**Val.**	**Test**	**Unlabeled**
IMDb	5×1K	5×1K	5×48K	50K
Churn	5×1K	5×1K	5×3K	100K

Table 3: Statistics of dataset used in experiments.

IMDb: The IMDb dataset was developed by Maas et al. (2011)[1] for sentiment classification where systems should classify the polarity of a given movie review as positive or negative. The dataset contains 50K labeled movie reviews. For the purpose of our experiments, we randomly sample 1K, 1K, and 48K instances from this data (with balanced distribution over classes) and treat them as labeled (training), validation, and test data respectively. We create five such datasets for robustness against different seeding or data partitions. This dataset also provides 50K unlabeled reviews.

Churn: This dataset contains more than 5K tweets about three telecommunication brands and was developed by Amiri and Daumé III (2015)[2] for the task of *churn* prediction[3] where systems should predict if a twitter post indicates user intention about leaving a brand - classifying tweets as *churny* or *non-churny* with respect to brands. We replace all target brand names with the keyword BRAND and other non-target brands with BRAND-OTHER for the purpose of our experiments. Similar to IMDb, we create five datasets for experiments. We also crawl an additional 100K tweets about the target brands and treat them as unlabeled data.

We evaluate models in terms of macro-F1 score, i.e. the mean of F1-scores across classes.

3.2 Downstream Learner and Settings

As downstream neural networks (referred to as base classifiers), we consider current state-of-the-art deep averaging networks (DANs) (Shen et al., 2018; Iyyer et al., 2015; Joulin et al., 2017; Arora et al., 2017) for IMDb, and a basic CNN model for Churn dataset with parameters set from the work presented in (Gridach et al., 2017) except for pretrained embeddings. In terms of DANs, we use `FastText` (Joulin et al., 2017) for its high per-

---

[1] `http://ai.stanford.edu/~amaas/data/sentiment/`

[2] `https://scholar.harvard.edu/hadi/chData`

[3] Churn is a term relevant to customer retention in marketing discourse; examples of churny tweets are "my days with BRAND are numbered," "debating if I should stay with BRAND," and "leaving BRAND in two days."

formance and simplicity. `FastText` is a feedforward neural network that consists of an embedding layer that maps vocabulary indices to embeddings, an averaging layer that averages word embeddings of inputs, and several hidden layers (we use two layers of size 256) followed by a prediction layer with `sigmoid` activation.

We use 300-dimensional word embeddings provided by Google's `word2vec` toolkit (Mikolov et al., 2013). In Algorithm 1, we set the number of training epochs to $k = 32$, and stop training when F1 performance on validation data stops improving with patience of three continuous iterations, i.e. after three continuous epochs with no improvement, training will be stopped. In addition, we set the number of training episodes to $K = 20$ and stop training when this number of episodes is reached or there is no unlabeled data left for sampling; the latter case is often the reason for stopping in our self-training method. In addition, we experiment with different number of Leitner queues chosen from $n = \{3, 5, 7, 9, 11\}$.

3.3 Baselines

We consider the following baselines:

- **Standard self-training**: This approach iteratively trains a network on current labeled data and applies it to current unlabeled data; it uses a prediction confidence threshold to sample unlabeled instances (Zhu, 2006). We set the best confidence threshold from {.80,.85,.90,.95} using validation data.
- **Autoencoder self-training** (Dai and Le, 2015): This approach first pretrains a network using unlabeled data (through a layer-wise training approach to optimally reconstruct the inputs), and then fine-tunes it using labeled data with respect to the target task.
- **Adversarial self-training** (Miyato et al., 2016): This model utilizes pretraining as described above, but also applies adversarial perturbations to word embeddings for more effective learning (perturbation is applied to embeddings instead of word inputs because words or their one-hot vectors do not admit infinitesimal perturbation; the network is trained to be robust to the worst perturbation).
- **Knowledge Transfer self-training** (Noroozi et al., 2018): This model uses a clustering approach (e.g. k-means) to create clusters of unlabeled instances that have similar representations, where representations are derived from standard pretraining as described above. The model then pretrains a network by learning to *classify* unlabeled instances to their corresponding clusters. The resulting pretrained network is then fine-tuned with respect to the target task using labeled data (with slight modification at prediction layer which makes the network suitable for target task). We set the best number of clusters from $\{10, 20, \ldots, 100\}$ based on model performance on validation data.

	IMDb	**Churn**
Base Classifier	73.02	65.77
SST (Standard ST)	74.43	65.77
PST (Pretraining ST)	76.36	67.27
AST (Adversarial ST)	76.09	67.70
KST (Knowledge Transfer ST)	77.11	67.06
LST (Leitner ST)	**78.27***	**69.90***

Table 4: Macro-F1 performance of models across datasets; Note that Standard ST (SST) samples only 1.4K and 0 instances from IMDb and Churn datasets respectively; sampling more data decreases SST's performance down to 66.94 and 57.04 perhaps due to ineffective exploring of data space. Our model achieves its best performance on IMDb and Churn datasets with $n = 5$ and $n = 7$ Leitner queues respectively.

3.4 Model Performance

Table 4 reports Macro-F1 performance of different models; we report average performance across five random test sets for each task (see Section 3.1 and Table 3). The performance of base classifiers in supervised settings, where the networks are only trained on original labeled datasets, is reasonably high on IMDb (73.02) and low on Churn (65.77). Standard ST (SST) improves performance on IMDb but not on Churn dataset. SST achieves its best performance (on validation data) in the first few episodes when, on average, 1.4K and 0 instances are sampled for IMDb and Churn datasets respectively. Beyond that, the performance considerably decreases down to 66.94 (IMDb) and 57.04 (Churn) respectively. This is perhaps due to imbalanced class size in Churn dataset, failure of SST to explore the data space, or classification mistakes that reinforce each other. Several previous works also observed no improvement with SST (Gollapalli et al., 2013; Zhu and Goldberg, 2009; Zhang and Rudnicky, 2006); but some successful applications have been reported (Wu et al., 2018; Zhou et al., 2012; Riloff and Jones, 1999; Yarowsky, 1995; HEARST, 1991).

The result also show that pretraining and adversarial-based training, PST and AST in Table 4 respectively, improve the performance of base classifiers by 3.34 and 3.37 points in macro-F1 on IMDb, and by 1.5 and 1.93 points on Churn dataset. In addition, since PST and AST show comparable performance, we conjecture that when original labeled data has a small size, adversarial-based self-training do not considerably improve pretraining. But, considerable improvement can be achieved with larger amount of labeled data, see (Miyato et al., 2016) for detailed comparison on pretraining and adversarial-based training. The results also show that knowledge transfer (KST) outperforms PST and AST on IMDb - indicating that good initial labels derived through clustering information could help semi-supervised learning, even with small amount of seed labeled data.

Table 4 also shows the result of our model, Leitner ST (LST). The best performance of LST is obtained using $n = 5$ and $n = 7$ queues for IMDb and Churn datasets respectively. Considering these queue lengths, our model outperforms base classifiers by 5.25 and 4.13 points in Macro-F1 on IMDb and Churn datasets respectively; similar to PST and AST, our model results in a greater gain when the learner has higher initial performance. It also improves the best self-training baseline, KST for IMDb and AST for Churn, by 1.16 and 2.2 points in macro-F1 on IMDb and Churn datasets respectively where both differences are significant (average ρ-values based on t-test are .004 and .015 respectively).

4 Model Introspection

We investigate several questions about our model to shed light on its improved performance. One partial explanation is that by differentiating instances and augmenting the informative ones, we are creating a more powerful model that better explores the space of unlabeled data. In this section, we elaborate on the behavior of our model by conducting finer-grained analysis at queue-level and investigating the following questions in the context of challenges of semi-supervised learning. Due to space limit, we mainly report results on IMDb and discuss corresponding behaviors on Churn dataset in the text.

4.1 Queue-level Performance

We analyze queue level performance to understand how instances of different queues contribute in creating better models during the self-training process. For this experiment, we train networks using our Leitner self-training framework as normal (where, at every iteration, only instances of the designated queue are added to training data), and report the average macro-F1 performance of the network–on validation data–if it is trained with instances of each queue. Concretely, we report average macro-F1 performance of models learned at line 6 of Algorithm 2 (see M_qs in Table 2).

Figures 2(a) and 2(b) show the results on IMDb and Churn datasets for $n = 5$ and $n = 7$ queues respectively. Note that the last queue for Churn dataset, q_6, has never been reached by any instance. This is perhaps because of the difficulty of this task[4] and low initial performance of the network on Churn dataset. q_2 on IMDb and q_4 on Churn dataset result in the best average performance across training episodes, both queues are close to the middle. In addition, the result show that the highest queues (q_4 for IMDb and q_5 for Churn) are often not the best queues. This result can justify the lower performance of Standard ST (SST) as instances in these queues are the easiest (and perhaps most confident ones) for the network; we further analyze these queues in Section 4.2.[5]

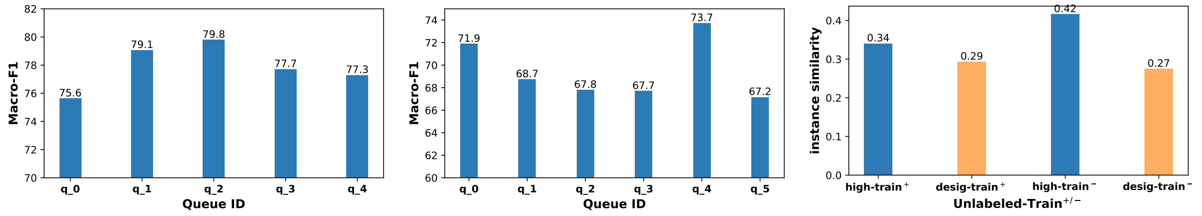
4.2 What's the Issue with Highest Queues?

As we discussed before, instances in the highest queues, although easy to learn for the classifier, are not informative and do not contribute to training an improved model; therefore, highest queues are often not selected by our model. To understand the reason, we try to quantify how well instances of these queues *match* with training data. For this purpose, we compute `cosine` similarity between representations of training instances (see below) and those in the highest and designated queues

---

[4] Churn prediction is a target-dependent task, largely affected by negation and function words, e.g. compare "*switching from*" and "*switching to*," and language complexity, e.g. the tweets "*hate that I may end up leaving BRAND cause they have the best service*" is a positive yet churny tweet.

[5] Note that the performance on lower queues (e.g. q_1 for IMDb and q_0 for Churn) are higher than expected. This is because, at the end of each iteration, instances of designated (best-performing) queues–but not lower queues–are added to training data; instances of designated queues help creating better and more robust models which still perform well even if instances of lower queues are added.



(a) Queue level performance (IMDb) (b) Queue level performance (Churn) (c) Highest vs. designated queue (IMDb)

Figure 2: (a) and (b): Average macro-F1 performance computed over individual queues using validation dataset across training episodes (average performance of M_qs at line 6 of Algorithm 2). (a): Performance on IMDb with optimal queue length of $n = 5$, and (b): performance on Churn with optimal queue length of $n = 7$: note that none of unlabeled instances has made it to the last queue. (c): Comparison of highest and designated queues in terms of instance similarity to training data; `high_train` indicates similarity between (representations of) instances in the highest queue and training instances, and `desig_train` shows the corresponding values for instances in the designated queue. $+$ and $-$ signs indicate positive and negative pseudo/gold labels for unlabeled/training instances.

during self-training as follows:

$$\frac{1}{K}\sum_{e=1}^{K}\texttt{cosine}(\mathbf{T}^e, \mathbf{Q}^e)$$

where $\mathbf{T}^e \in \mathbb{R}^{m^e \times d}$ and $\mathbf{Q}^e \in \mathbb{R}^{p^e \times d}$ indicate representations of training instances and those of a given target queue respectively (where d indicates the dimension of representations, and m^e and p^e indicate number of instances in training data and target queue at episode e respectively), and `cosine(.,.)` computes L2-normalized dot product of its input matrices. To obtain the above representations for instances, we compute the output of the last hidden layer (the layer below prediction layer) of the *trained* network at each episode. These outputs can be considered as feature representations for inputs. For finer-grained comparison, we compute similarities with respect to positive and negative classes.

As the results in Figure 2(c) show, instances in the highest queue match well with current training data (and hence the current model), and, therefore, are less informative. On the other hand, instances in the designated queues show considerably smaller similarity with training instances in both positive and negative classes, and, therefore, do not match well with training data. These instances are more informative, and help the network to better explore the space of unlabeled data and optimize for the target task.

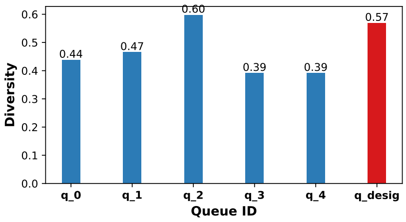


Figure 3: The amount of diversity that instances of each queue introduce if added to training data (on IMDb).

4.3 Does Diversity Matter?

We analyze different queues to measure the extent of diversity that each queue introduces to training data during our normal self-training process where, at every iteration, only instances of the designated queue are added to training data. Specifically, we compute the extent of diversity that each given queue introduces as follows:

$$\frac{1}{K}\sum_{e=1}^{K} 1 - \texttt{cosine}(\mathbf{T}^e, \texttt{concat}(\mathbf{T}^e, \mathbf{Q}^e))$$

where, as before, $\mathbf{T}^e$ and $\mathbf{Q}^e$ indicate the representations of training and queue instances at episode e respectively, and `concat(.,.)` is a function that creates a new dataset by vertically concatenating $\mathbf{T}^e$ and $\mathbf{Q}^e$.

Figure 3 shows the results. On IMDb, q_2 and designated queues show greater diversity to training data compared to other queues. We note that q_0 carries a greater diversity than q_3 and q_4, but, as we observed in Figure 2, instances of q_0 do not improve performance of the model, perhaps due to their difficulty or wrong pseudo labels. We observe similar behavior in case of Churn dataset where q_4 introduces the highest diversity. From this analysis, we conclude that Leitner self-training enables sampling diverse sets of instances that contributes to training an improved model.

	Leitner ST				
	$n = 3$	$n = 5$	$n = 7$	$n = 9$	$n = 11$
IMDb	76.83	76.77	78.03	75.34	**80.71**
Churn	65.74	64.87	67.06	**68.56**	65.80

Table 5: Macro-F1 performance of diverse queues across datasets. Compare these results with those obtained by designated queues in Table 4.

4.3.1 Diverse Queue

Given the above results on diversity, we investigate whether greater diversity can further improve the performance of our model. For this analysis, we create a considerably more "diverse" queue at every self-training episode and treat it as the designated queue. We create the diverse queue by sampling instances with *high* prediction confidence from *all* queues. In particular, at every episode, we rank instances of each queue based on their prediction confidence and create a diverse queue by combining top $r\%$ instances of each queue, where r indicates the rate of adding new instances and set to $r = 10\%$. We note that a smaller rate is better for adding instances because it allows the model to gradually consume unlabeled instances with high prediction confidence.

Table 5 shows the effect of diverse queues on the performance of our model on both IMDb and Churn datasets. The results show that diverse queues improve the performance of our Leitner self-training model from 78.27 (reported in Table 4) to 80.71 on IMDb, i.e. 2.44 points improvement in macro-F1. However, the corresponding performance on Churn dataset decreases from 69.90 to 68.56, i.e. 1.34 points decrease in macro-F1. The inverse effect of diverse queues in case of Churn dataset is because diverse queues suffer from the issue of considerable class imbalance more than designated queues. This is because highly confident instances which accumulate in higher queues are often negative instances in case of Churn prediction. Although we tackle this issue by weighting the loss function during training, diverse positive instances which are different from their training counterparts are still needed for performance improvement.

4.4 Do We Need Better Sampling Policies?

We investigate the challenges associated with our data sampling policy by conducting finer-grained analysis on instance movement patterns among queues. To illustrate, assume that we have a Leitner queue of size $n = 3$ and the following movement patterns for four individual instances that ultimately home in on q_0 (recall that correct prediction promotes an instance to a higher queue, while wrong prediction demotes it to a lower queue):

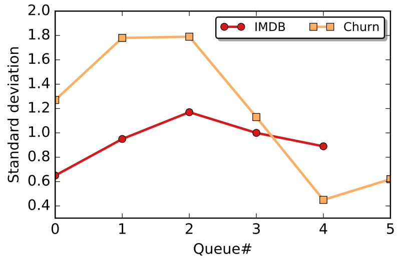


Figure 4: Deviation in instance movements for each queue (in terms of average standard deviation over all training episodes). At every episode, we keep track of instance movements among queues and measure movement variation among instances that ultimately home in on the same queue.

$q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$: always in q_0
$q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$: mainly in q_0
$q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0$: partially in q_0
$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_1 \rightarrow q_0$: partially in q_0 & q_1.

Although all these instances ultimately home in on the same queue, they may have different contributions to the training of a model because there is a considerable difference in the ability of the downstream network in learning their labels. Therefore, if there is a large deviation among movement patterns of instances of the same queue, better data sampling policies could be developed, perhaps through finer-grained queue-level sampling.

For this analyses, we keep track of instance movements among queues and measure standard deviation among movement patterns of instances of the same queue at every self-training episode, and report the average of these deviations.

Figure 4 shows the results. On both datasets, there is considerably greater deviation in movements for middle queues than lower/higher queues. This is meaningful because Leitner system (and other spaced repetition schedulers) are expected to keep easy and hard instances at higher and lower queues respectively. Since such instances mainly stay at lower or higher queues, we observe smaller deviation in their movements. On the other hand, the corresponding values for middle queues indicate that movements in these queues are spread out over a larger range of queues. From these results, we conjecture that a data sampling policy that conducts finer-grained analysis at queue-level (e.g. by taking into account queue movement patterns) could create better data samples. Verifying this hypothesis will be the subject for future work.

5 Related Work

Semi-supervised learning (Zhu, 2006; Chapelle et al., 2009) is a type of machine learning where one has access to a small amount of labeled data and a large amount of unlabeled data. Self-training is a type of semi-supervised learning to boost the performance of downstream learners (e.g. classifiers) through data sampling from unlabeled data. Most data sampling policies rely on prediction confidence of the downstream learner for sampling unlabeled data (Zhu and Goldberg, 2009). Self-training has been successfully applied to various tasks and domains including word sense disambiguation (HEARST, 1991; Yarowsky, 1995), information extraction (Riloff and Jones, 1999), and object recognition (Zhou et al., 2012).

In addition, co-training (Blum and Mitchell, 1998; Zhang and Rudnicky, 2006; Wu et al., 2018) is another type of semi-supervised learning. It assumes that each instance can be described using two *distinct* feature sets that provide different and complementary information about the instance. Ideally, the two views should be conditionally independent, i.e., the two feature sets of each instance are conditionally independent given the class, and each view should be sufficient, i.e., the class of an instance can be accurately predicted from each view alone. Co-training first learns separate downstream learners for each view using a small set of labeled data. The most confident predictions of each learner on the unlabeled data are then used to iteratively construct additional labeled training data. Recently Wu et al. (2018) developed an effective model based on reinforcement learning (specifically, a joint formulation of a Q-learning agent and two co-training classifiers) to learn data sampling policies and utilize unlabeled data space in the context of co-training methods.

Effective semi-supervised learning algorithms based on pretraining techniques (Hinton and Salakhutdinov, 2006; Bengio et al., 2007; Erhan et al., 2010) have been developed for text classification, deep belief networks (Hinton and Salakhutdinov, 2006), and stacked autoencoders (Vincent et al., 2010; Bengio et al., 2007). In particular, Dai and Le (2015) developed an autoencoder for the later supervised learning process. Miyato et al. (2016) applied perturbations to word embeddings and used pretraining technique and adversarial training for effective semi-supervised learning. These models although effective have not been well studied in the context of semi-supervised learning where models may have low initial performance or limited amount of labeled data. In addition, pretraining is limited by the same architecture requirement in both pretraining and fine-tuning steps.

In this work, we extend previous work in self-training by developing a new and effective data sampling policy based on spaced repetition (Dempster, 1989; Cepeda et al., 2006; Averell and Heathcote, 2011) which addresses some of the above challenges. In particular, our model's data sampling policy is not predetermined, it explores the entire data space and dynamically selects unlabeled instances with respect to the strength of a learner on a target task and easiness of unlabeled instances, and it relaxes the same model constraint of pretraining-based approaches by decoupling pretraining and fine-tuning steps.

6 Conclusion and Future Work

We propose a novel method based on spaced repetition to self-train neural networks using small amount of labeled and large amount of unlabeled data. Our model can select high-quality unlabeled data samples for self-training and outperforms current state-of-the-art semi-supervised baselines on two text classification problems. We analyze our model from various perspectives to explain its improvement gain with respect to challenges of semi-supervised learning. There are several venues for future work including (a): finer-grained data sampling at queue level, (b): extending our model to other machine learning algorithms that employ iterative training, such as boosting approaches, and (c): applying this model to areas where neural networks have not been investigated, e.g. due to limited availability of labeled data.

Acknowledgments

I sincerely thank Mitra Mohtarami and anonymous reviewers for their insightful comments and constructive feedback.

References

Hadi Amiri and Hal Daumé III. 2015. Target-dependent churn classification in microblogs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2361–2367. AAAI Press.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2017. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2410.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations (ICLR)*.

Lee Averell and Andrew Heathcote. 2011. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132(3):354.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Frank N Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C Lee Giles. 2013. Researcher homepage classification using unlabeled data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 471–482. ACM.

Mourad Gridach, Hatem Haddad, and Hala Mulki. 2017. Churn identification in microblogs using convolutional neural networks with structured logical knowledge. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 21–30.

M HEARST. 1991. Noun homograph disambiguation using local context in large corpora. In *Proc. 7th Annual Conference of the Centre for the New OED and Text Research: Using Corpora*, pages 1–22.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Sebastian Leitner. 1974. *So lernt man lernen*. Herder.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *International conference on learning representations (ICLR)*.

Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. 2018. Boosting self-supervised learning via knowledge transfer. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9359–9367.

Timothy P Novikoff, Jon M Kleinberg, and Steven H Strogatz. 2012. Education of a model student. *Proceedings of the National Academy of Sciences*, 109(6):1868–1873.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the sixteenth national conference on Artificial intelligence*, pages 474–479. American Association for Artificial Intelligence.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

Yu-Xiong Wang and Martial Hebert. 2016. Learning from small sample sets by combining unsupervised meta-training with cnns. In *Advances in Neural Information Processing Systems*, pages 244–252.

Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1252–1262.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

Rong Zhang and Alexander I Rudnicky. 2006. A new data selection approach for semi-supervised acoustic modeling. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham. 2012. Self-training with selection-by-rejection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 795–803. IEEE.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Technical Report 1530, Computer Science, University of Wisconsin-Madison*, 2(3):4.

Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Neural Language Models as Psycholinguistic Subjects: Representations of Syntactic State

Richard Futrell[1], **Ethan Wilcox**[2], **Takashi Morita**[3,4], **Peng Qian**[5], **Miguel Ballesteros**[6], and **Roger Levy**[5]

[1]Department of Language Science, UC Irvine, `rfutrell@uci.edu`
[2]Department of Linguistics, Harvard University, `wilcoxeg@g.harvard.edu`
[3]Primate Research Institute, Kyoto University, `tmorita@alum.mit.edu`
[4]Department of Linguistics and Philosophy, MIT
[5]Department of Brain and Cognitive Sciences, MIT, {`pqian`,`rplevy`}`@mit.edu`
[6]IBM Research, MIT-IBM Watson AI Lab, `miguel.ballesteros@ibm.com`

Abstract

We investigate the extent to which the behavior of neural network language models reflects incremental representations of syntactic state. To do so, we employ experimental methodologies which were originally developed in the field of psycholinguistics to study syntactic representation in the human mind. We examine neural network model behavior on sets of artificial sentences containing a variety of syntactically complex structures. These sentences not only test whether the networks have a representation of syntactic state, they also reveal the specific lexical cues that networks use to update these states. We test four models: two publicly available LSTM sequence models of English (Jozefowicz et al., 2016; Gulordava et al., 2018) trained on large datasets; an RNN Grammar (Dyer et al., 2016) trained on a small, parsed dataset; and an LSTM trained on the same small corpus as the RNNG. We find evidence for basic syntactic state representations in all models, but only the models trained on large datasets are sensitive to subtle lexical cues signalling changes in syntactic state.

1 Introduction

It is now standard practice in NLP to derive sentence representations using neural sequence models of various kinds (Elman, 1990; Sutskever et al., 2014; Goldberg, 2017; Peters et al., 2018; Devlin et al., 2018). However, we do not yet have a firm understanding of the precise content of these representations, which poses problems for interpretability, accountability, and controllability of NLP systems. More specifically, the success of neural sequence models has raised the question of whether and how these networks learn robust syntactic generalizations about natural language, which would enable robust performance even on data that differs from the peculiarities of the training set.

Here we build upon recent work studying neural language models using experimental techniques that were originally developed in the field of psycholinguistics to study language processing in the human mind. The basic idea is to examine language models' behavior on targeted sentences chosen to probe particular aspects of the learned representations. This approach was introduced by Linzen et al. (2016), followed more recently by others (Bernardy and Lappin, 2017; Enguehard et al., 2017; Gulordava et al., 2018), who used an agreement prediction task (Bock and Miller, 1991) to study whether RNNs learn a hierarchical morphosyntactic dependency: for example, that *The key to the cabinets...* can grammatically continue with *was* but not with *were*. This dependency turns out to be learnable from a language modeling objective (Gulordava et al., 2018). Subsequent work has extended this approach to other grammatical phenomena, with positive results for filler–gap dependencies (Chowdhury and Zamparelli, 2018; Wilcox et al., 2018) and negative results for anaphoric dependencies (Marvin and Linzen, 2018).

In this work, we consider syntactic representations of a different kind. Previous studies have focused on relationships of **dependency**: one word licenses another word, which is tested by asking whether a language model favors one (grammatically licensed) form over another in a particular context. Here we focus instead on whether neural language models show evidence for incremental **syntactic state** representations: whether behavior of neural language models reflects the kind of generalizations that would be captured using a stack-based incremental parse state in a symbolic grammar-based model. For example, during the underlined portion of Example (1), an incremental language model should represent and maintain the knowledge that it is currently inside a subordinate clause, implying (among other things) that a full main clause must follow.

Proceedings of NAACL-HLT 2019, pages 32–42
Minneapolis, Minnesota, June 2 - June 7, 2019.

(1) As the doctor studied the textbook, the nurse walked into the office.

In this work, we use a targeted evaluation approach (Marvin and Linzen, 2018) to elicit evidence for syntactic state representations from language models. That is, we examine language model behavior on artificially constructed sentences designed to expose behavior that is crucially dependent on syntactic state representations. In particular, we study complex subordinate clauses and garden path effects (based on main-verb/reduced-relative ambiguities and NP/Z ambiguities). We ask three general questions: (1) Is there basic evidence for the representation of syntactic state? (2) What textual cues does a neural language model use to infer changes to syntactic state? (3) Do the networks maintain knowledge about syntactic state over long spans of complex text, or do the syntactic state representations degrade?

Among neural language models, we study both generic sequence models (LSTMs), which have no explicit representation of syntactic structure, and an RNN Grammar (RNNG) (Dyer et al., 2016), which explicitly calculates Penn Treebank-style context-free syntactic representations as part of the process of assigning probabilities to words. This comparison allows us to evaluate the extent to which explicit representation of syntactic structure makes models more or less sensitive to syntactic state. RNNGs have been found to outperform LSTMs not only in overall test-set perplexity (Dyer et al., 2016), but also in modeling long-distance number agreement in Kuncoro et al. (2018) for certain model configurations; our work extends this comparison to a variety of syntactic state phenomena.

2 General methods

We investigate neural language model behavior primarily by studying the **surprisal**, or log inverse probability, that a language model assigns to each word in a sentence:

$$S(x_i) = -\log_2 p(x_i|h_{i-1}),$$

where x_i is the current word or character, h_{i-1} is the model's hidden state before consuming x_i, the probability is calculated from the network's softmax activation, and the logarithm is taken in base 2, so that surprisal is measured in bits. Surprisal is equivalent to the pointwise contribution to the language modeling loss function due to a word.

In psycholinguistics, the common practice is to study reaction times per word (for example, reading time as measured by an eyetracker), as a measure of the word-by-word difficulty of online language processing. These reading times are often taken to reflect the extent to which humans expect certain words in context, and may be generally proportional to surprisal given the comprehender's probabilistic language model (Hale, 2001; Levy, 2008; Smith and Levy, 2013; Futrell and Levy, 2017). In this study, we take language model surprisal as the analogue of human reading time, using it to probe the neural networks' expectations about what words will follow in certain contexts. There is a long tradition linking RNN performance to human language processing (Elman, 1990; Christiansen and Chater, 1999; MacDonald and Christiansen, 2002) and grammaticality judgments (Lau et al., 2017), and RNN surprisals are a strong predictor of human reading times (Frank and Bod, 2011; Goodkind and Bicknell, 2018). RNNGs have also been used as models of human online language processing (Hale et al., 2018).

2.1 Experimental methodology

In each experiment presented below, we design a set of sentences such that the word-by-word surprisal values will show evidence for syntactic state representations. The idea is that certain words will be surprising to a language model only if the model has a representation of a certain syntactic state going into the word. We analyze word-by-word surprisal profiles for these sentences using regression analysis. Except where otherwise noted, all statistics are derived from linear mixed-effects models (Baayen et al., 2008) with sum-coded fixed-effect predictors and maximal random slope structure (Barr et al., 2013). This method lets us factor out by-item variation in surprisal and focus on the contrasts between conditions.

2.2 Models tested

We study the behavior of four models of English: two LSTMs trained on large data, an an RNNG and an LSTM trained on matched, smaller data (the Penn Treebank). The models are summarized in Table 1. All models are trained on a language modeling objective.

Our first LTSM is the model presented in Jozefowicz et al. (2016) as "BIG LSTM+CNN Inputs", which we call "JRNN", which was trained on the One Billion Word Benchmark (Chelba et al., 2013) with two hidden layers of 8196 units each

Model	Architecture	Training data	Data size (tokens)	Reference
JRNN	LSTM	One Billion Word	~ 800 million	Jozefowicz et al. (2016)
GRNN	LSTM	Wikipedia	~ 90 million	Gulordava et al. (2018)
RNNG	RNN Grammar	Penn Treebank	~ 1 million	Dyer et al. (2016)
TinyLSTM	LSTM	Penn Treebank	~ 1 million	—

Table 1: Models tested, by architecture, training data, and training data size.

and CNN character embeddings as input. The second large LSTM is the model described in the supplementary materials of Gulordava et al. (2018), which we call “GRNN”, trained on 90 million tokens of English Wikipedia with two hidden layers of 650 hidden units each.

Our RNNG is trained on syntactically labeled Penn Treebank data (Marcus et al., 1993), using 256-dimensional word embeddings for the input layer and 256-dimensional hidden layers, and dropout probability 0.3. Next-word predictions are obtained through hierarchical softmax with 140 clusters, obtained with the greedy agglomerative clustering algorithm of Brown et al. (1992). We estimate word surprisals using word-synchronous beam search (Stern et al., 2017; Hale et al., 2018): at each word w_i a beam of incremental parses is filled, the summed forward probabilities (Stolcke, 1995) of all candidates on the beam is taken as a lower bound on the prefix probability: $P_{\min}(w_{1\ldots i})$, and the surprisal of the i-th word in the sentence is estimated as $\log \frac{P_{\min}(w_{1\ldots i})}{P_{\min}(w_{1\ldots i-1})}$. Our action beam is size 100, and our word beam is size 10. Finally, to disentangle effects of training set from model architecture, we use an LSTM trained on string data from the Penn Treebank training set, which we call TinyLSTM. For TinyLSTM we use 256-dimensional word-embedding inputs and hidden layers and dropout probability 0.3, just as with the RNNG.

3 Subordinate clauses

We begin by studying subordinate clauses, a key example of a construction requiring stack-like representation of syntactic state. In such constructions, as shown in Example (1), a **subordinator** such as “as” or “when” serves as a cue that the following clause is a subordinate clause, meaning that it must be followed by some main (matrix) clause. In an incremental language model, this knowledge must be maintained and carried forward while processing the words inside subordinate clause. A grammar-based symbolic language model (e.g., Stolcke, 1995; Manning and Carpenter, 2000) would maintain this knowledge by keeping track of syntactic rules representing the incomplete subordinate clause and the upcoming main clause in a stack data structure. Psycholinguistic research has clearly demonstrated that humans maintain representations of this kind in syntactic processing (Staub and Clifton, 2006; Lau et al., 2006; Levy et al., 2012). Here we ask whether the string completion probabilities produced by neural language models show evidence of the same knowledge.

We can detect the knowledge of syntactic state in this case by examining whether the network licenses and requires a matrix clause following the subordinate clause. These expectations can be detected by examining surprisal differences between sentences of the form in Example (2):

(2) a. As the doctor studied the textbook, the nurse walked into the office. [SUBordinator, MATRIX]
b. *As the doctor studied the textbook. [SUB, NO-MATRIX]
c. ?The doctor studied the textbook, the nurse walked into the office. [NO-SUBordinator, MATRIX]
d. The doctor studied the textbook. [NO-SUB, NO-MATRIX]

If the network *licenses* a matrix clause following the subordinate clause—and maintains knowledge of that licensing relationship throughout the clause, from the subordinator to the comma—then this should be manifested as lower surprisal at the matrix clause in (2-a) as compared to (2-c). We call this the **matrix licensing effect**: the surprisal of the condition [SUB, MATRIX] minus [NOSUB, MATRIX], which will be negative if there is a licensing effect. If the network *requires* a following matrix clause, then this will be manifested as higher surprisal at the matrix clause for (2-b) compared with (2-d). We call this the **no-matrix penalty effect**: the surprisal of [SUB,NOMATRIX] minus [NOSUB, NOMATRIX], which will be positive if there is a penalty.

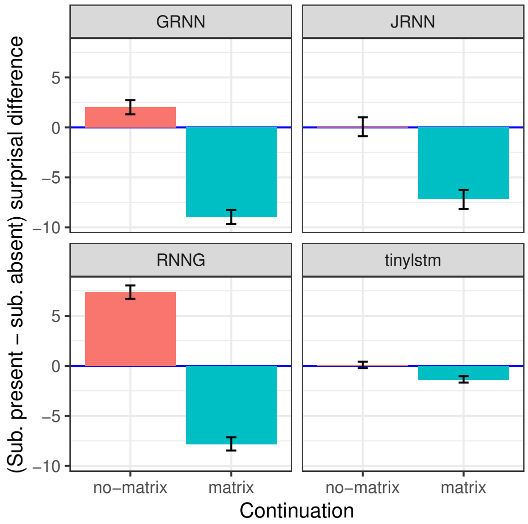


Figure 1: Effect of subordinator absence/presence on surprisal of continuations. Red: no-matrix penalty effect. Blue: matrix licensing effect. In this and all other figures, unless otherwise noted, **error bars** represent 95% confidence intervals of the *contrasts between conditions shown*, computed from the standard error of the by-item and by-condition mean surprisals after subtracting out the by-item means (Masson and Loftus, 2003).

We designed 23 experimental items on the pattern of (2) and calculated difference in the sum surprisal of the words in the matrix clause.[1] Figure 3 shows the matrix licensing effect (in blue) and the no-matrix penalty effect (in red), averaged across items. For all models, we see a facilitative matrix licensing effect ($p < .001$ for all models), smallest in TinyLSTM. However, we only find a significant no-matrix penalty for GRNN and the RNNG ($p < .001$ in both): the other models do not significantly penalize an ungrammatical continuation ($p = .9$ for JRNN; $p = .5$ for TinyLSTM). That is, JRNN and TinyLSTM give no indication that (2-b) is less probable than (2-c).

We found that all models at least partially represent the licensing relationship between a subordinate and matrix clause. However, in order to fully represent the syntactic requirements induced by a subordinator, it seems that a model needs either large amounts of data (as in GRNN) or explicit representation of syntax (as in the RNNG, as opposed to TinyLSTM).

3.1 Maintenance and degradation of syntactic state

The foregoing results show that neural language models use the presence of a subordinator as a cue to the onset of a subordinate clause, and that they maintain knowledge that they are in a subordinate clause throughout the intervening material up to the comma. Now we probe the ability of models to maintain this knowledge over long spans of complex intervening material. To do so, we use sentences on the template of (2) and add intervening material modifying the NPs in the subordinate clause. To both of these NPs (in subject and object position), we add modifiers of increasing syntactic complexity: PPs, subject-extracted relative clauses (SRCs), and object-extracted relative clauses (ORCs), as shown in Figure 2. We study the extent to which these modifiers weaken the language models' expectations about the upcoming matrix clause.

As a summary measure of the strength of language models' expectations about an upcoming matrix clause, we collapse the two measures of the previous section into one: the **matrix licensing interaction**, consisting of the difference between the no-matrix penalty effect and the matrix licensing effect (the two bars in Figure 1). A similar measure was used to detect filler–gap dependencies by Wilcox et al. (2018).

Figure 3 shows the strength of the matrix licensing interaction given sentences with various modifiers inserted. For the large LSTMs, GRNN exhibits a strong interaction when the intervening material is short and syntactically simple, and the interaction gets progressively weaker as the intervening material becomes progressively longer and more complex ($p < 0.001$ for subject postmodifiers and $p < 0.01$ object postmodifiers). The other models show less interpretable behavior.

Our results indicate that at least some large LSTMs, along with the RNNG, are capable of maintaining a representation of syntactic state over spans of complex intervening material. Quantified as a licensing interaction, this representation of syntactic state exhibits the most clearly understandable behavior in GRNN, which shows a graceful degradation of syntactic expectations as the complexity of intervening material increases. The representation is maintained most strongly in the RNNG, except for one particular construction (object-position SRCs).

---

[1] Note that it would not be sufficient to look at surprisal only at the punctuation token, because the comma could indicate the beginning of a conjoined NP.

As the $\underbrace{\text{doctor}}$ studied the $\underbrace{\text{textbook}}$ . . .

in a white lab coat (PP)
who was wearing a white lab coat (SRC)
who the administrator had recently hired (ORC)
(Subject interveners)

about several recent advances in cancer therapy (PP)
that described several recent advances in cancer therapy (SRC)
that colleagues had written on cancer therapy (ORC)
(Object interveners)

Figure 2: Scheme for lengthening the subordinate clause in Section 3.1.

4 Garden path effects

The major phenomenon that has been used to probe incremental syntactic representations in humans is **garden path effects**. Garden path effects arise from local ambiguities, where a context leads a comprehender to believe one parse is likely, but then a disambiguating word forces her to drastically revise her beliefs, resulting in high surprisal/reading time at the disambiguating word. In effect, the comprehender is "led down the garden path" by a locally likely but ultimately incorrect parse (Bever, 1970). Garden-pathing in LSTMs has recently been demonstrated by van Schijndel and Linzen (2018a,b) in the context of modeling human reading times.

Garden path effects allow us to detect representations of syntactic state because if a person or language model shows a garden path effect at a word, that means that the person or model had some belief about syntactic state which was disconfirmed by that word. In psycholinguistics, these effects have been used to study the question of what information determines people's beliefs about likely parses given locally ambiguous contexts: for example, whether factors such as world knowledge play a role (Ferreira and Clifton, 1986; Trueswell et al., 1994).

Here we study two major kinds of local ambiguities inducing garden path effects. For each ambiguity, we ask two main questions. First, whether the network shows the basic garden path effect, which would indicate that it had a syntactic state representation that made a disambiguating word surprising. Second, whether the network is sensitive to subtle lexical cues to syntactic structure which may modulate the size of the garden path effect: this question allows us to determine what information the network uses to determine the beginnings and endings of certain syntactic states.

4.1 NP/Z Ambiguity

The **NP/Z ambiguity**[2] refers to a local ambiguity in sentences of the form given in Example (3).

(3)a. When the dog scratched the vet with his new assistant **took off** the muzzle. [TRANSITIVE, NOCOMMA]
b. When the dog scratched, the vet with his new assistant **took off** the muzzle. [TRANSITIVE, COMMA]
c. When the dog struggled the vet with his new assistant **took off** the muzzle. [INTRANSITIVE, NOCOMMA]
d. When the dog struggled, the vet with his new assistant **took off** the muzzle. [INTRANSITIVE, COMMA]

When a comprehender reads the underlined phrase "the vet with his new assistant" in (3-a), she may at first believe that this phrase is the direct object of the verb "scratched" inside the subordinate clause. However, upon reaching the verb "took off", she realizes that the underlined phrase was not in fact an object of the verb "scratched", rather it was the subject of a new clause, and the subordinate clause in fact ended after the verb "scratched". The key region of the sentence where the garden path disambiguation happens—called the **disambiguator**—is the phrase "took off", marked in bold.

While a garden path should obtain in (3-a), no such garden path should exist for (3-b), because a comma clearly demarcates the end of the subordinate clause. Therefore a basic garden path effect would be indicated by the difference in surprisal at the disambiguator for (3-a) minus (3-b). Furthermore, if a comprehender is sensitive to the relationship between verb argument structure and clause boundaries, then there should be no garden path in (3-c), because the verb "struggled" is INTRANSITIVE: it cannot take an object in English, so an incremental parser should never be misled into believing that "the vet..." is its object. This lexical information about syntactic structure is subtle enough that there has been controversy about whether even humans are sensitive to it in online processing (Staub, 2007).

4.1.1 NP/Z Garden Path Effect

We tested whether neural language models would show the basic garden path effect and if this ef-

---

[2] For Noun Phrase/Zero ambiguity. At first the embedded verb appears to take an NP object, but later it turns out that it was a zero (null) object.

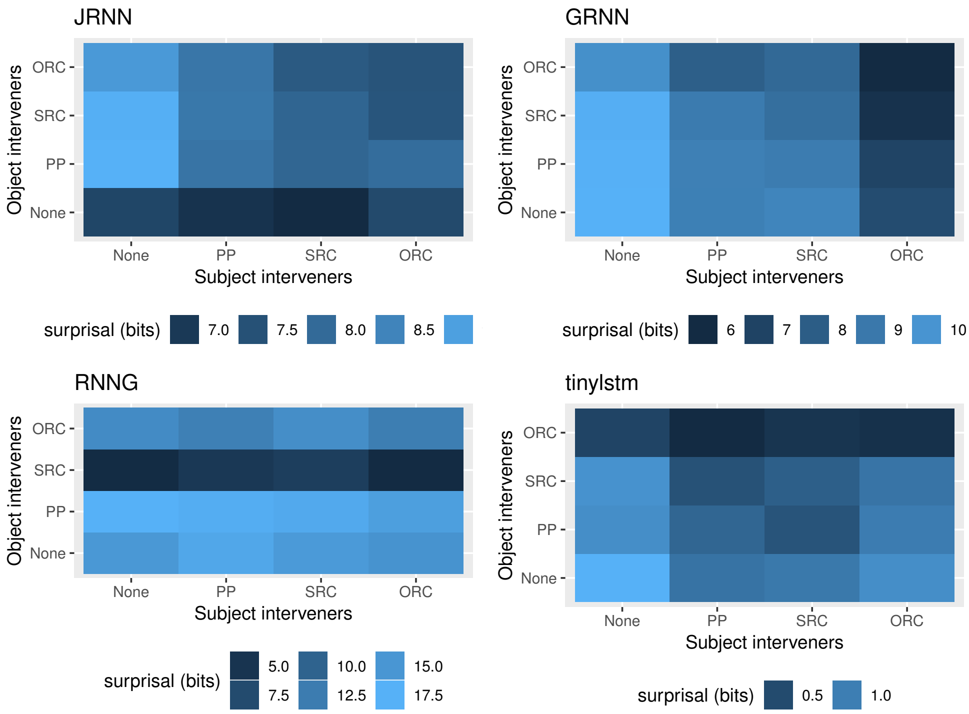


Figure 3: Size of matrix clause licensing interaction (see text) given various intervening elements in the subordinate clause. Note that the heatmaps are on different scales across models.

fect would be modulated by verb transitivity. We constructed 32 items based of the same structure as (3), based on materials from Staub (2007), manipulating the transitivity of the embedded verb ("scratched" vs. "struggled"), and the presence of a disambiguating comma at the end of the subordinate clause. An NP/Z garden path effect would show up as increased surprisal at the main verb "took off" in the absence of a comma. If the networks use the transitivity of the embedded verb as a cue to clause structure, and maintain that information over the span of six words between the embedded verb and the main verb, then there should be a garden path effect for the transitive verb, but not for the intransitive verb. More generally we would expect a *stronger* garden path given the transitive verb than given the intransitive verb.

Figure 4 shows the mean surprisals at the disambiguator for all four models, for both transitive and intransitive embedded verbs. The overall per-region surprisals, averaged over words in each region, are shown in Figure 5. We see that a garden path effect exists in all models (though very small in TinyLSTM): all models show significantly higher surprisal at the main verb when the disambiguating comma is absent ($p < .001$ for all models). However, only the large LSTMs appear to be sensitive to the transitivity of the embedded verb, showing a smaller garden path effect for intransitive verbs. Statistically, there is a significant interaction of comma presence and verb transitivity only in GRNN and JRNN (GRNN: $p < .01$; JRNN: $p < .001$; RNNG: $p = .3$, TinyLSTM: $p = .3$).

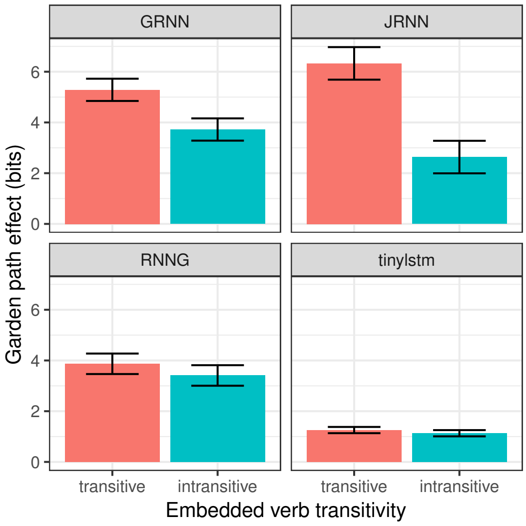


Figure 4: Average garden path effect (surprisal at disambiguator in NO-COMMA condition minus COMMA condition) by model and embedded verb transitivity.

All models show NP/Z garden path effects, indicating that they are sensitive to some cues indicat-

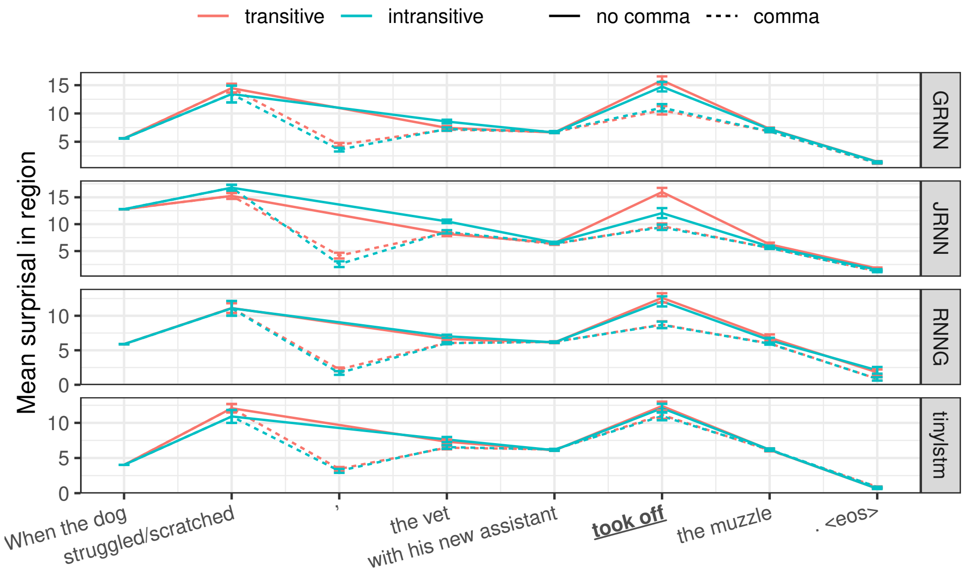


Figure 5: Region-by-region surprisal values for NP/Z garden path materials. Surprisal values are averaged across items and across words in regions. The critical region where the garden path effect is visible is the verb "took off".

ing end-of-clause boundaries. However, only the large LSTMs appear to use verb argument structure information as a cue to these boundaries. The results suggest that very large amounts of data may be necessary for current neural models to discover such fine-grained dependencies between syntactic properties of verbs and sentence structure.

4.1.2 Maintenance and degradation of state

We can probe the maintenance and degradation of syntactic state information by manipulating the length of the intervening material between the onset of the local ambiguity and the disambiguator in examples such as (3). The question is whether the networks maintain the knowledge, while processing the intervening material, that the intervening noun phrase is probably the object of the embedded verb inside a subordinate clause, or whether they gradually lose track of this information. To study this question we used materials on the pattern of (4): these materials manipulate the length of the intervening material (underlined) while holding constant the distance between the subordinator ("As") and the disambiguator (**grew**).

(4)a. As the author studying Babylon in ancient times wrote the book **grew**. [SHORT, NO-COMMA]

b. As the author studying Babylon in ancient times wrote, the book **grew**. [SHORT, COMMA]

c. As the author wrote the book describing Babylon in ancient times **grew**. [LONG, NO-COMMA]

d. As the author wrote, the book describing Babylon in ancient times **grew**. [LONG, COMMA]

If neural language models show degradation of syntactic state, then the garden path effect (measured as the difference in surprisal between the COMMA and NO-COMMA conditions at the disambiguator) will be smaller for the LONG conditions. We tested 32 sentences of the form in (4), based on materials from Tabor and Hutchins (2004). The garden path effect sizes are shown in Figure 6.

We find a significant garden effect in all models in the SHORT condition ($p < .001$ in JRNN and GRNN; $p < .01$ in the RNNG and $p = .03$ in TinyLSTM). In the long condition, we find the garden path effect in all models except TinyLSTM: ($p < .001$ in JRNN; $p < .01$ in GRNN; $p = .02$ in the RNNG; and $p = .2$ in TinyLSTM). The crucial interaction between length and comma presence (indicating that syntactic state degrades) is significant in GRNN ($p < .01$) and TinyLSTM ($p < .001$) but not JRNN ($p = .7$) nor the RNNG ($p = .6$). The pattern is reminiscent of the results on degradation of state information about subordinate clauses in Section 3, where GRNN and TinyLSTM showed the clearest evidence of degradation.

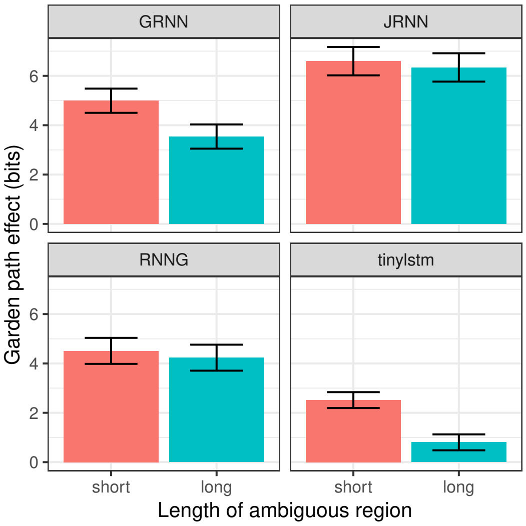


Figure 6: Average garden path effect by model and length of ambiguous region.

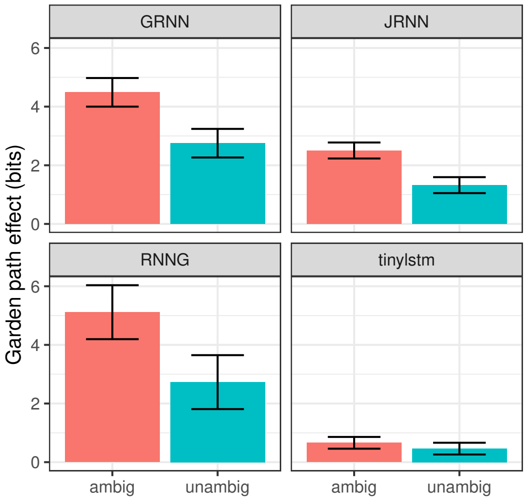


Figure 7: Garden path effect size for MV/RR ambiguity by model and verb-form ambiguity.

Note that the pattern found here is the opposite of the pattern of human reading times. Humans appear to show "digging-in" effects: the longer the span of time between the introduction of a local ambiguity and its resolution, the larger the garden path effect (Tabor and Hutchins, 2004; Levy et al., 2009).

4.2 Main Verb/Reduced Relative Ambiguity

Next we turn to garden path effects induced by the classic **Main Verb/Reduced Relative (MV/RR) ambiguity**, in which a word is locally ambiguous between being the main verb of a sentence or introducing a **reduced relative clause** (reduced RC: a relative clause with no explicit complementizer, headed by a passive-participle verb). That ambiguity can be maintained over a long stretch of material:

(5)a. The woman brought the sandwich from the kitchen **tripped** on the carpet. [REDUCED, AMBIGuous]
b. The woman who was brought the sandwich from the kitchen **tripped** on the carpet. [UNREDUCED, AMBIG]
c. The woman given the sandwich from the kitchen **tripped** on the carpet. [REDUCED, UNAMBIGuous]
d. The woman who was given the sandwich from the kitchen **tripped** on the carpet. [UNREDUCED, UNAMBIG]

In Example (5-a), the verb "brought" is initially analyzed as a main verb phrase, but upon reaching the verb "tripped"—the disambiguator in this case—the reader must re-analyze it as an RC. The garden path should be eliminated in sentences such as (5-b), the UNREDUCED condition, where the words "who was" clarify that the verb "brought" is part of an RC, rather than the main verb of the sentence. Therefore we quantify the garden path effect as the surprisal at the disambiguator for the REDUCED minus UNREDUCED conditions.

There is another possible cue that the initial verb is the head of an RC: the morphological form of the verb. In examples such as (5-c), the the verb "given" is unambiguously in its past-participle form, indicating that it cannot be the main verb of the sentence. If a language model is sensitive to morphological cues to syntactic structure, then it should either not show a garden path effect in this UNAMBIGuous condition, or it should show a reduced garden path effect.

We constructed 29 experimental items following the template of (5). Figure 7 shows the garden path effect sizes by model and verb-form ambiguity. All networks show the basic garden path effect ($p < .001$ in JRNN, GRNN, and RNNG; $p < 0.01$ in TinyLSTM). However, the garden path effect in TinyLSTM is much smaller than the other models: RC reduction causes an additional .3 bits of surprisal at the disambiguating verb, as compared to 2.8 bits in the RNNG, 1.9 in JRNN, and 3.6 in GRNN (TinyLSTM's garden path effect is significantly smaller than each other model at $p < 0.001$).

If the network is using the morphological form

Phenomenon	GRNN	JRNN	RNNG	TinyLSTM
Subordination	✓✓	✓✗	✓✓	✓✗
NP/Z Garden Path	✓✓	✓✓	✓✗	✓✗
MV/RR Garden Path	✓✓	✓✓	✓✓	✓✗

Table 2: Summary of results by model and phenomenon. The first check mark indicates basic evidence of syntactic state representation. The second check mark indicates the ability to capture more fine-grained phenomena: for subordination, the no-matrix penalty effect; for the NP/Z garden path, the effect of verb transitivity; and for the MV/RR garden path, the effect of verb morphology.

of the verb as a cue to syntactic structure, then it should show the garden path effect more strongly in the AMBIG condition than the UNAMBIG condition. The large language models and the RNNG do show this pattern: at the critical main-clause verb, surprisal is superadditively highest in the reduced ambiguous condition (the dotted blue line; a positive interaction between the reduced and ambiguous conditions is significant in the three models at $p < 0.001$). However, TinyLSTM does not show evidence for superadditive surprisal for the ambiguous verbform and the reduced RC ($p = .45$).

The three large LSTMs and the RNNG replicate the key human-like garden-path disambiguation effect due to to ambiguity in verb form. But strikingly, even when the participial verbform is unambiguous, there is still a significant garden path effect in all models ($p < 0.01$ in all models except TinyLSTM, where $p = .08$). Apparently, these networks treat an unambiguous passive-participial verb as only a *noisy cue* to the presence of an RC.

5 General Discussion and Conclusion

In all models studied, we found clear evidence of basic incremental state syntactic representation. However, models varied in how well they fully captured the effects of such state and the potentially subtle lexical cues indicating the beginnings and endings of such states: only the large LSTMs could sometimes reliably infer clause boundaries from verb argument structure (Section 4.1) and morphological verb-form (Section 4.2), and only GRNN and the RNNG fully captured the proper behavior of subordinate clauses. The results are summarized in Table 2. We suggest that representation of course-grained syntactic structure requires either syntactic supervision or large data, while exploiting fine-grained lexical cues to structure requires large data.

More generally, we believe that the psycholinguistic methodology employed in this paper provides a valuable lens on the internal representations of black-box systems, and can form the basis for more systematic tests of the linguistic competence of NLP systems. We make all experimental items, results, and analysis scripts available online at `github.com/langprocgroup/nn_syntactic_state`.

References

R. Haraald Baayen, D.J. Davidson, and D.M. Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412.

Dale J. Barr, Roger P. Levy, Christoph Scheepers, and Harry J Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

Jean-Philippe Bernardy and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *Linguistic Issues in Language Technology*, 15:1–15.

Thomas G. Bever. 1970. The cognitive basis for linguistic structures. In J. R. Hayes, editor, *Cognition and the Development of Language*. Wiley, New York.

Kathryn Bock and Carol A Miller. 1991. Broken agreement. *Cognitive Psychology*, 23(1):45–93.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144.

Morten H. Christiansen and Nick Chater. 1999. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23(2):157–205.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

J.L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Emile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. *arXiv preprint arXiv:1706.03542*.

Fernanda Ferreira and Charles Clifton. 1986. The independence of syntactic processing. *Journal of Memory and Language*, 25(3):348–368.

Stefan L. Frank and Rens Bod. 2011. Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science*, 22(6):829–834.

Richard Futrell and Roger Levy. 2017. Noisy-context surprisal as a human sentence processing cost model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 688–698, Valencia, Spain.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18, Salt Lake City, UT. Association for Computational Linguistics.

K. Gulordava, P. Bojanowski, E. Grave, T. Linzen, and M. Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL*.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Melbourne, Australia.

John T. Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics and Language Technologies*, pages 1–8.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv*, 1602.02410.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Ellen Lau, Clare Stroud, Silke Plesch, and Colin Phillips. 2006. The role of structural prediction in rapid syntactic analysis. *Brain & Language*, 98:74–88.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: a probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1241.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Roger Levy, Evelina Fedorenko, Mara Breen, and Ted Gibson. 2012. The processing of extraposed structures in English. *Cognition*, 122(1):12–36.

Roger P Levy, Florencia Reali, and Thomas L Griffiths. 2009. Modeling the effects of memory on human online sentence processing with particle filters. In *Advances in neural information processing systems*, pages 937–944.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Maryellen C. MacDonald and Morten H. Christiansen. 2002. Reassessing working memory: Comment on Just and Carpenter (1992) and Waters and Caplan (1996). *Psychological Review*, 109(1):35–54.

Christopher D Manning and Bob Carpenter. 2000. Probabilistic parsing using left corner language models. In *Advances in probabilistic and other parsing technologies*, pages 105–124. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Michael EJ Masson and Geoffrey R Loftus. 2003. Using confidence intervals for graphically based data interpretation. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 57(3):203.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Marten van Schijndel and Tal Linzen. 2018a. Modeling garden path effects without explicit hierarchical syntax. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*.

Marten van Schijndel and Tal Linzen. 2018b. A neural model of adaptation in reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Nathaniel J. Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.

Adrian Staub. 2007. The parser doesn't ignore intransitivity, after all. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(3):550.

Adrian Staub and Charles Clifton. 2006. Syntactic prediction in language comprehension: Evidence from *either . . . or*. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 32(2):425–436.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700. Association for Computational Linguistics.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Whitney Tabor and Sean Hutchins. 2004. Evidence for self-organized sentence processing: Digging-in effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(2):431.

John C. Trueswell, Michael K. Tanenhaus, and S. Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language*, 33:285–318.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do rnn language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221. Association for Computational Linguistics.

Understanding language-elicited EEG data by predicting it from a fine-tuned language model

Dan Schwartz
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
drschwar@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
tom.mitchell@cs.cmu.edu

Abstract

Electroencephalography (EEG) recordings of brain activity taken while participants read or listen to language are widely used within the cognitive neuroscience and psycholinguistics communities as a tool to study language comprehension. Several time-locked stereotyped EEG responses to word-presentations – known collectively as event-related potentials (ERPs) – are thought to be markers for semantic or syntactic processes that take place during comprehension. However, the characterization of each individual ERP in terms of what features of a stream of language trigger the response remains controversial. Improving this characterization would make ERPs a more useful tool for studying language comprehension. We take a step towards better understanding the ERPs by fine-tuning a language model to predict them. This new approach to analysis shows for the first time that all of the ERPs are predictable from embeddings of a stream of language. Prior work has only found two of the ERPs to be predictable. In addition to this analysis, we examine which ERPs benefit from sharing parameters during joint training. We find that two pairs of ERPs previously identified in the literature as being related to each other benefit from joint training, while several other pairs of ERPs that benefit from joint training are suggestive of potential relationships. Extensions of this analysis that further examine what kinds of information in the model embeddings relate to each ERP have the potential to elucidate the processes involved in human language comprehension.

1 Introduction

The cognitive processes involved in human language comprehension are complex and only partially identified. According to the dual-stream model of speech comprehension (Hickok and Poeppel, 2007), sound waves are first converted to phoneme-like features and further processed by a ventral stream that maps those features onto words and semantic structures, and a dorsal stream that (among other things) supports audio-short term memory. The mapping of words onto meaning is thought to be subserved by widely distributed regions of the brain that specialize in particular modalities — for example visual aspects of the

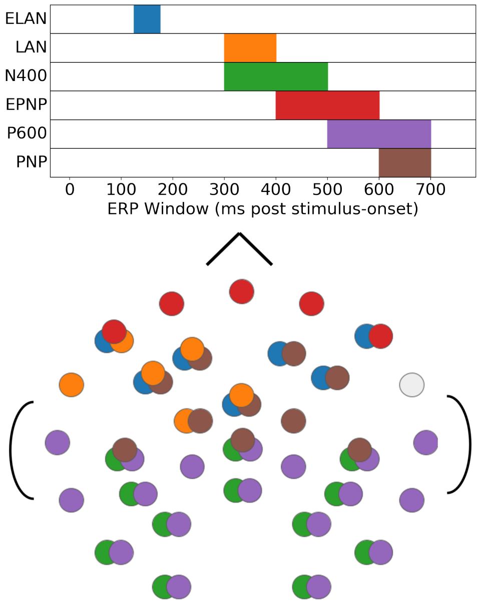


Figure 1: The electrodes from which each event-related potential was recorded in the data from Frank et al. (2015) (after figure 3 in (Frank et al., 2015)). The bottom portion of the figure shows a top-down schematic of the electrode locations with the nose facing towards the top of the page. Each ERP is the mean potential from all of the indicated electrodes during a specific time-window, creating a single scalar value per ERP per word. Overlapping circles indicate multiple ERPs recorded from the same electrode. The ELAN is measured from 125-175ms after stimulus onset, the LAN from 300-400ms, the N400 from 300ms-500ms, the EPNP from 400-600ms, the P600 from 500-700ms, and the PNP from 600-700ms.

Proceedings of NAACL-HLT 2019, pages 43–57
Minneapolis, Minnesota, June 2 - June 7, 2019.

word *banana* reside in the occipital lobe of the brain and are activated when the word *banana* is heard (Kemmerer, 2014) — and the different representation modalities are thought to be integrated into a single coherent latent representation in the anterior temporal lobe (Ralph et al., 2010). While this part of meaning representation in human language comprehension is somewhat understood, much less is known about how the meanings of words are integrated together to form the meaning of sentences and discourses. One tool researchers use to study the integration of meaning across words is electroencephelography (EEG), which measures the electrical activity of large numbers of neurons acting in concert. EEG has the temporal resolution necessary to study the processes involved in meaning integration, and certain stereotyped electrical responses to word presentations, known as event-related potentials (ERPs), have been identified with some of the processes thought to contribute to comprehension.

In this work, we consider six ERP components that have been associated in the cognitive neuroscience and psycholinguistics literature with language processing and which we analyze in the data from Frank et al. (2015) (see Figure 1 for spatial and temporal definitions of these ERP components). Three of these — the N400, EPNP, and PNP responses — are primarily considered markers for semantic processing, while the other three — the P600, ELAN, and LAN responses — are primarily considered markers for syntactic processing. However, the neat division of the ERP responses into either semantic or syntactic categories is controversial. The N400 response has been very well studied (for an overview see (Kutas and Federmeier, 2011)) and it is well established that it is associated with semantic complexity, but the features of language that trigger the other ERP responses we consider here are poorly understood. We propose to use a neural network pretrained as a language model to probe what features of language drive these ERP responses, and in turn to probe what features of language mediate the cognitive processes that underlie human language comprehension, and especially the integration of meaning across words.

2 Background

While a full discussion of each ERP component and the features of language thought to trigger each are beyond the scope of this document (for reviews see e.g. Frank et al. (2015), Kemmerer (2014), Kutas and Federmeier (2011), Kuperberg et al. (2003), and Van Petten and Luka (2012)), we introduce some basic features of ERP components to help in the discussion later. ERP components are electrical potential responses measured with respect to a baseline that are triggered by an event (in our case the presentation of a new word to a participant in an experiment). The name of each ERP component reflects whether the potential is positive or negative relative to the baseline. The N400 is so-named because it is **N**egative relative to a baseline (the baseline is typically recorded just before a word is presented at an electrode that is not affected by the ERP response) and because it peaks in magnitude at about **400**ms after a word is presented to a participant in an experiment. The P600 is **P**ositive relative to a baseline and peaks around **600**ms after a word is presented to a participant (though its overall duration is much longer and less specific in time than the N400). The post-N400 positivity is so-named because it is part of a biphasic response; it is a positivity that occurs after the negativity associated with the N400. The early post-N400 positivity (EPNP) is also part of a biphasic response, but the positivity has an eariler onset than the standard PNP. Finally, the LAN and ELAN are the left-anterior negativity and early left-anterior negativity respectively. These are named for their timing, spatial distribution on the scalp, and direction of difference from the baseline. It is important to note that ERP components can potentially cancel and mask each other, and that it is difficult to precisely localize the neural activity that causes the changes in electrical potential at the electrodes where those changes are measured.

3 Related Work

This work is most closely related to the paper from which we get the ERP data: Frank et al. (2015). In that work, the authors relate the surprisal of a word, i.e. the (negative log) probability of the word appearing in its context, to each of the ERP signals we consider here. The authors do not directly train a model to predict ERPs. Instead, models of the probability distribution of each word in context are used to compute a surprisal for each word, which is input into a mixed effects regression along with word frequency, word length, word position in the sentence, and sentence position in the experiment. The effect of the surprisal is assessed using a likelihood-ratio test. In Hale et al. (2018), the authors take an approach similar to Frank et al. (2015). The authors compare the explanatory power of surprisal (as computed by an LSTM or a Recurrent Neural Network Grammar (RNNG) language model) to a measure of syntactic complexity they call "distance" that counts the number of parser actions in the RNNG language model. The authors find that surprisal (as predicted by the RNNG) and distance are both significant factors in a mixed effects regression which predicts the P600, while the surprisal as computed by an LSTM is not. Unlike Frank et al. (2015) and Hale et al. (2018), we do not use a linking function (e.g. surprisal) to relate a language model to ERPs. We thus lose the interpretability provided by the linking function, but we are able to predict a significant proportion of the variance for all of the ERP components, where prior work could not. We interpret our results through characterization of the ERPs in terms of how they relate to each other and to eye-tracking data rather than through a linking function. The authors in Wehbe et al. (2014) also use a recurrent neural network to predict neural activity directly. In that work the authors predict magnetoencephalography (MEG) activity, a close cousin to EEG, recorded while participants read a chapter of *Harry Potter and the Sorcerers Stone* (Rowling, 1999). Their approach to characterization of processing at each MEG sensor location is to determine whether it is best predicted by the context vector of the recurrent network (prior to the current word being processed), the embedding of the current word, or the probability of the current word given the context. In future work we also intend to add these types of studies to the ERP predictions.

4 Method

Data. We use two sources of data for this analysis. The primary dataset we use is the ERP data collected and computed by Frank et al. (2015), and we also use behavioral data (eye-tracking data and self-paced reading times) from Frank et al. (2013) which were collected on the same set of 205 sentences. In brief, the sentences were selected from sources using British English with a criterion that they be understandable out of context. We use the ERP component values as computed by Frank et al. (2015) which have been high-pass filtered at 0.5 Hz to reduce correlation between ERP components and modulus transformed (John and Draper, 1980) to make the distribution of component values more normal. We do not use the 100ms pre-trial baseline which is made available by Frank et al. (2015) and which they use as a separate input to the mixed effects regression. For more information about the ERP datasets and data collection procedures we refer the reader to the original papers. For the behavioral data, we use self-paced reading times and four eye-tracking measures. Self-paced reading time is considered a signal of integration difficulty (i.e. as it becomes more difficult to integrate the meaning of the current word into the context, the amount of time a reader spends on the current word increases). The eye-tracking measures are intended to capture both early effects (effects modulated primarily by properties of the word independent of its context, such as word frequency and word length) and late effects (effects modulated by the context in which the word is found, i.e. comprehension difficulty) in word processing (Rayner and Pollatsek, 2006). In both cases, the eye-tracking measures provide a signal of overt visual attention, which is thought to strongly correlate with covert perceptual attention in normal reading (Rayner, 2009). We log-transform the self-paced reading time and the eye-tracking measures.

Model. To predict the ERP signals in the data, we start with a 3-layer bidirectional LSTM-based language model encoder using the architecture found in Merity et al. (2017) and pretrained on the WikiText-103 dataset (Merity et al., 2016) (we use the pretrained model from Howard and Ruder (2018)). The pretraining objective is to minimize the negative log-likelihood of the next word for the forward LSTM and the previous word for the reverse LSTM. The word-embeddings (input embeddings) in the encoder have 400 components, the hidden layer outputs have 1150 components each, and the context-embeddings output from the encoder have 400 components. The forward-encoder and backward-encoder are independently fine-tuned on the baby version of the British National Corpus (Consortium, 2005) to help with prediction of British English (both the ERP data and eye-tracking data use British English). During task training the two encoders' output embeddings are concatenated together and fed into a causal-convolution layer which combines each pair of adjacent timepoints into a single pair-embedding with 10 components. The causal-convolution (i.e. convolution which is left padded) ensures that the pair-embeddings are aligned so that the prediction targets correspond to the later word in the pair. In other words the pair can be thought of as representing the ‘current’ and ‘previous’ words together. A ReLU is applied to the pair-embedding before it, along with the word length and the log probability of the word, is fed into a linear output layer to predict each ERP and behavioral measure (see Figure 2). The convolution and linear layers are initialized using the default PyTorch (Paszke et al., 2017) initialization, i.e. the initialization proposed in He et al. (2015). The encoder portion of the model includes dropout as applied in Merity et al. (2017), but we use different dropout probabilities when we fit the neural and behavioral data (the dropout probability on the input embeddings was 0.05, 0.4 on the input to the LSTM, 0.4 on LSTM hidden layers, 0.5 on the output of the LSTM, and 0.5 on the recurrent weights). We did not find dropout in the decoder to be helpful. We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.95, \beta_2 = 0.999$ for training and we use mean squared error as the loss.

Procedure. We begin our training procedure by fine-tuning the forward- and backward-encoders independently on the baby version of the British National Corpus (Consortium, 2005). This corpus has British English that may help in modeling the University College London corpus, while not overlapping with it.

After the model fine-tuning, we estimate how well the model predicts each of the ERP signals and eye-tracking measures by training the model 100 times with different train/test splits and decoder parameter initializations. We use 10% of the data for testing and the remainder for training. The sentences in the ERP data are split at random. After we split the data, we compute the mean and standard deviation of each ERP signal (and each eye-tracking measure and the self-paced reading time) within participant on the training data. We use these values to standardize the training data within participant, and then average the data from all of the participants together. After we average, we again compute the mean and standard deviation to standardize the average. We follow a similar procedure for the test data, but we use the mean and standard deviation from the training data when standardizing. Note that we use the log of the behavior measures, and the log is taken before the data-standardization.

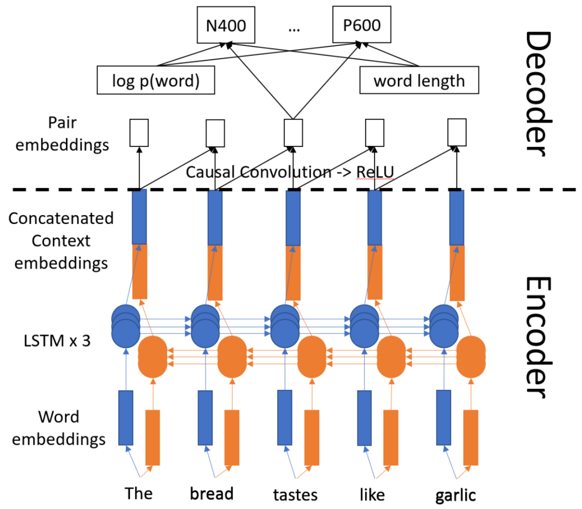


Figure 2: The model uses an encoder based on the architecture and regularization in Merity et al. (2017) and pretrained by Howard and Ruder (2018). Within this architecture 2 independent 3-layer LSTM models encode a sentence. The context-embeddings output from each encoder are then concatenated together to give a single representation to each word in the sentence. These concatenated context-embeddings are fed into a causal-convolution, which learns a function to combine each pair of context-representations into a pair-embedding. A rectified linear unit (ReLU) non-linearity is applied to the pair-embedding, after which independent linear layers map the pair-embedding along with the log-probability of a word and the word-length to a prediction of each ERP or behavioral signal.

In the loss function (and when we evaluate model performance) we only consider content words. We mark as a content word any word that is an adjective, adverb, auxiliary verb, noun, pronoun, proper noun, or verb (including to-be verbs). All other words are considered function words.

During the first 20 epochs of training, only the parameters of the decoder are modified. Following this, we train the model for an additional 15 epochs during which the parameters of the decoder and the final layer of the encoder (the final LSTM layer in both the forward and backward encoder) can be modified. We also experimented with additional training epochs and allowing all parameters of the model to be modified, but we found that this caused overfitting.

Comparing models trained with different loss functions. To better understand the relationship between ERP signals, and between ERP signals and behavioral data, we train the model with different loss functions that include mean squared error terms corresponding to various combinations of the ERP signals and behavioral data. For example, one of the training variations includes a mean squared error term for the P600 and a mean squared error term for the N400 in the loss, but does not use the other signals during training. In this variation, for a mini-batch of size B, where example b has T_b content tokens and the superscripts p and a denote the predicted and actual values for a measure respectively, the loss function can be written as:

$$\frac{1}{\sum_{b=1}^{B} T_b} \sum_{b=1}^{B} \sum_{t=1}^{T_b} (\mathrm{P600}_{b,t}^{p} - \mathrm{P600}_{b,t}^{a})^2 + (\mathrm{N400}_{b,t}^{p} - \mathrm{N400}_{b,t}^{a})^2 \tag{1}$$

For each of the training variations, we repeat the training procedure described above (but fine-tuning the language model on the British National Corpus is done only once). We use a consistent train/test split procedure, such that the split for the ith run of the 100 runs is the same across all training variations, but the split changes between run i and run j. This enables us to use paired statistical testing when we test for significance.

We test for whether the proportion of variance explained (computed as $1 - \frac{\text{MSE}}{\text{variance}}$ on the validation set) on each ERP and behavioral measure is significantly different from 0 using the single sample t-test controlled for false discovery rate using the Benjamini-Hochberg-Yekutieli procedure (Benjamini and Yekutieli, 2001) with a false discovery rate of 0.01.

To test whether the proportion of variance explained is different between different training variations (for example training with just the N400 signal included in the loss vs. training with both the N400 and the LAN included in the loss), we use a paired t-test. We then adjust for the false discovery rate again with a rate of 0.01.

5 Results

All ERP components are predictable. In the original study on this dataset, the investigators found that when surprisal was used as a linking function between the language model and the mixed effects regression, the only ERP for which the surprisal showed a significant effect in the regression was the N400 (Frank et al., 2015). In contrast, we find that when we directly predict the ERP signals we are able to predict a significant proportion of the variance for all of them (see Table 1).

Joint training benefits ERP component prediction. To explore the relationship between ERP components, we train $63 = \binom{6}{1} + \binom{6}{2} + \cdots + \binom{6}{6}$ different models using all of the possible combinations of which of the six ERP signals are included in the loss function during training. For each of the six ERP components, we look for the best performing models (see Table 1). The N400 is best predicted when the model is trained on that component independently, but every other ERP component prediction can be improved by including a second ERP component in the training. Thus multitask learning has a clear benefit when applied to the ERP data and some information is shared between ERP component predictions via the model parameters. We also note that it is not the case that training with more ERP components is always better, or that the signals which are most correlated benefit each other most (see Appendix A). The relationship between components clearly impacts whether the prediction of one ERP component benefits from the inclusion of others in model training. The results suggest that 8 pairs of ERP signals are related to each other: the LAN is paired with the P600, EPNP, and PNP, the ELAN with the N400, EPNP, PNP, and P600, and the EPNP is paired with the P600. We discuss these relationships in the Discussion section.

In an additional analysis, we modified our training procedure slightly to probe how jointly training on multiple ERP components compares to training individually on each ERP component. In this analysis we compare only training on each ERP component individually to training on all six ERP components together. We also train for a total of 60 epochs (rather than the 35 epochs used elsewhere). During the first 20 epochs we allow only the parameters of the decoder to be modified. During the next 20 epochs, we allow the parameters of the decoder and the final layer of the encoder (i.e. the final recurrent layer) to be modified. During the last 20 epochs, we allow all of the parameters of the model to be modified. The mean squared error for each of the ERP components from this anal-

Target	Additional	POVE	Target	Additional	POVE	Target	Additional	POVE
ELAN		0.20	LAN		0.30	N400		0.26
ELAN	+ EPNP	0.22	LAN	+ EPNP	0.31			
ELAN	+ N400	0.22	LAN	+ PNP	0.32			
ELAN	+ PNP	0.22	LAN	+ P600	0.32			
ELAN	+ P600	0.22	LAN	+ PNP, N400	0.33			
EPNP		0.34	P600		0.27	PNP		0.33
EPNP	+ LAN	0.35	P600	+ EPNP	0.30	PNP	+ LAN	0.36
EPNP	+ GROUP A	0.36	P600	+ LAN	0.30	PNP	+ GROUP B	0.36

Table 1: Proportion of variance explained (POVE) for each of the ERP components (mean of 100 training runs). The second column in each cell shows which ERP components in addition to the target ERP component were included in training. All combinations of training signals were explored. Shown is the best combination for each ERP target as well as every combination which is (i) significantly different from training on the target component alone, (ii) not significantly different from the best training combination, and (iii) uses no more than the number of signals used by the best combination. The N400 is predicted best when only the N400 signal is included in training. All values are significantly different from 0. GROUP A refers to (PNP, ELAN, LAN, P600) and GROUP B refers to (EPNP, ELAN, LAN, P600).

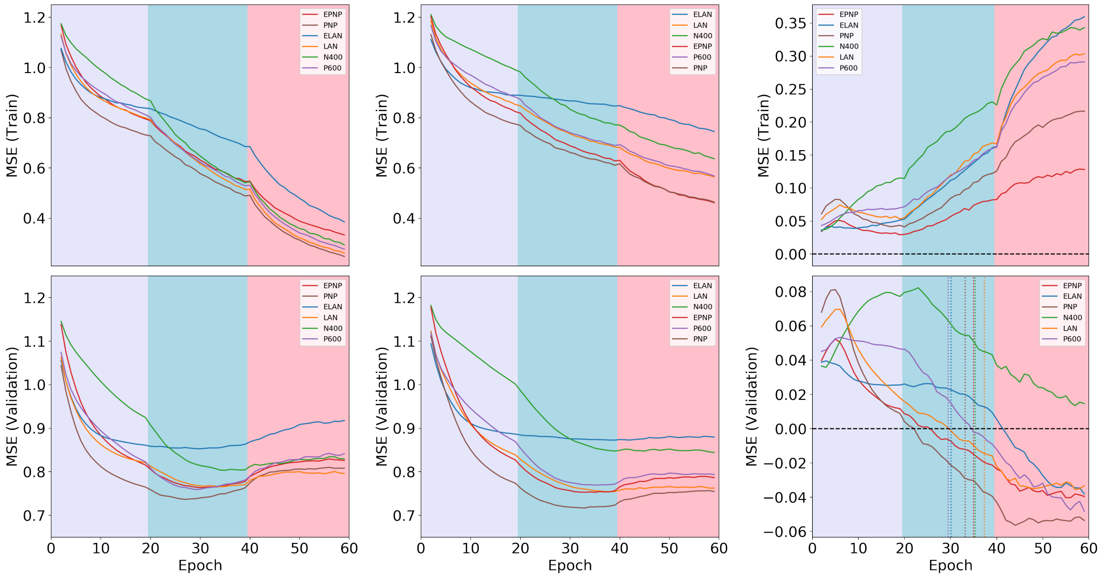
ysis is shown for each epoch in Figure 3. From the loss curves, we make a few observations. First, we see inflection points at epochs 20 and 40, when we allow more parameters of the model to be modified. The first inflection point indicates that allowing the recurrent layer to be modified benefits the prediction, while the second inflection point shows that overfitting becomes more severe if we allow all parameters of the model to be modified. We also see from these curves that part of the benefit of joint training is that it helps reduce overfitting – we see less of a climb in the validation loss after the minimum point in the joint training. Beyond this reduction in overfitting severity, we note that for some of the ERP components (the LAN, EPNP and PNP components) joint training actually gives a better overall minimum in prediction error.

Behavioral data benefits the prediction of ERP components. We are also interested in whether behavioral data can be used to improve ERP prediction since it should signal both the amount of overt attention required at various points in a sentence as well as integration difficulty. To study this question, we again train models using different combinations of training signals that include or do not include the behavioral data predictions in the loss function (see Table 2). We see that self-paced reading time indeed can improve prediction of a target ERP component relative to training on the target ERP component alone by about the same amount as the best combination of ERP components for all but the N400. Eye-tracking data can also improve the prediction accuracy of the ELAN, P600, and PNP components.

Insensitivity to choice of architecture. One potential concern about our results is the degree to which the relationships we see between ERP components and between ERP components and behavioral data is an artefact of our rather arbitrary choice of network architecture. We partially address this by running the same analysis using (i) only the forward direction of the encoder, and (ii) only the word-embeddings (the input embeddings) and not the context-embeddings (the output embeddings) of the encoder. The proportion of variance explained for each ERP component is lower using these variants of the analysis than using the bidirectional variant (see Appendix A), but qualitatively the relationships are similar. We leave further analysis of the sensitivity of our qualitative results to choice of architecture for future work.

6 Discussion

In this work we find that all six of the ERP components from Frank et al. (2015) can be predicted above chance by a model which has been pretrained using a language modeling objective and then directly trained to predict the components. This is in contrast to prior work which has successfully linked language models to the N400 (Frank et al., 2015) and P600 (Hale et al., 2018) but not the other ERP components. We also note that con-



(a) Independently trained (b) Jointly trained (c) Joint - Indep.

Figure 3: The mean squared error (MSE) for prediction of each of the ERP signals during each epoch of training (mean of 100 training runs). The first 2 epochs have been omitted for clarity. During the first 20 epochs (lavender background), only the decoder parameters are modified. During the next 20 epochs (light blue background), the parameters in the final layer of the encoder are also modified. During the last 20 epochs (pink background), all of the parameters are modified. Note that in this model architecture, information can be shared between ERP signals even when only the decoder is modified. The figure shows the MSE when separate models are trained for each ERP independently (a), the MSE when a single model is trained on all ERPs jointly (b), and the difference between these two scenarios (c). The top row in each column shows the MSE on the training data while the bottom row shows the MSE on the validation data. In the bottom row right, the dotted vertical lines indicate the epoch at which the minimum MSE is reached in the lower of the independent or joint training. The LAN, EPNP, and PNP all show modest benefits from joint training before overfitting sets in (the minimum value occurs in the joint training scenario), while all ERP signals other than the N400 show reduced overfitting in joint training.

Target	Additional	POVE	Target	Additional	POVE	Target	Additional	POVE
ELAN		0.20	LAN		0.30	N400		0.26
ELAN	+ ERP	**0.22**	LAN	+ ERP	**0.33**	N400	+ ERP	0.26
ELAN	+ READ	**0.22**	LAN	+ READ	**0.31**	N400	+ READ	0.27
ELAN	+ EYE	**0.22**	LAN	+ EYE	0.30	N400	+ EYE	0.25
EPNP		0.34	P600		0.27	PNP		0.33
EPNP	+ ERP	**0.36**	P600	+ ERP	**0.30**	PNP	+ ERP	**0.36**
EPNP	+ READ	**0.35**	P600	+ READ	**0.29**	PNP	+ READ	**0.34**
EPNP	+ EYE	0.34	P600	+ EYE	**0.29**	PNP	+ EYE	**0.34**

Table 2: Proportion of variance explained (POVE) for each of the ERP components (mean of 100 training runs). +ERP indicates the best combination of ERP training signals for the target ERP component, + READ indicates the inclusion of self-paced reading times, +EYE indicates the inclusion of eye-tracking data, and bold font indicates a significant difference from training on the target component alone.

trary to Hale et al. (2018), we find that an LSTM does contain information that can be used to predict EEG data, and in particular that it can predict the P600. We speculate that the analysis used in Hale et al. (2018) did not find reliable effects because the language models were related to the EEG data through functions chosen a priori (the surprisal, and the 'distance' metric). These functions, though interpretable, might be interpretable at the cost of losing much of the information in the representations learned by the network.

In addition, we show through our multitask learning analysis that information is shared between ERP components, and between ERP components and behavioral data. Although these relationships must be viewed with caution until they can be verified across multiple datasets and with more variation in neural network architectures, here we consider some potential reasons for our findings. The broad point we wish to make is that by better understanding which ERP components share information with each other and with behavioral data through the type of analysis we present here (multitask learning) or other means, we can better understand what drives each ERP component and in turn the processes involved in human language comprehension.

Relationship between ERPs. Our findings that the LAN and P600 are related, and that the ELAN and P600 are related are expected from both a theoretical perspective and from previous work examining the interactions of ERP components (Gunter et al., 1997; Hagoort et al., 2003a; Hahne and Friederici, 1999; Kutas et al., 2006; Palolahti et al., 2005). Since the ELAN and LAN have been theorized by some to mark word-category (i.e. part-of-speech) or morpho-syntactic (e.g. subject-verb number agreement) violations (Friederici, 2011; Hahne and Friederici, 2002; Hagoort et al., 2003b) and the P600 is considered a marker for syntactic effort (Coulson et al., 1998; Huettig, 2015; Kemmerer, 2014; Kuperberg, 2007; Kuperberg et al., 2003; Van Petten and Luka, 2012), these signals would naturally be related to each other.

The other relationships we find are more surprising. Some researchers have speculated that the LAN and ELAN are markers for working memory demands (King and Kutas, 1995; Kutas et al., 2006), and that indeed these might be part of sustained negativities that are frequently masked by the P600 (Kemmerer, 2014). If we take this view, then we would expect to find them in the presence of semantic and syntactic complexity, and this might explain why they seem to benefit from joint training with the other ERP component signals (and benefit prediction of other ERP signals with which they are trained). However, it is notable that predictions of the LAN and ELAN do not benefit each other in our analysis, and that the N400 (a marker for semantic complexity) is not benefited by the prediction of any other ERP component. This absence is by no means definitive, but it undermines the argument that all of these relationships can be explained by complexity and working memory demands alone.

The relative isolation of the N400 from other ERP components in our analysis is interesting. If the N400 is a marker for semantic memory retrieval (Kutas and Federmeier, 2011), then it might be expected to be somewhat isolated from the other components, which may involve syntactic processing or later integration effects.

Alternatively, the relationships we find in our analysis might be an artefact of the way the ERPs are operationalized in Frank et al. (2015). Several of the pairings we find overlap spatially and are near to each other in time, so the ERP components might spill over into each other. Further work is required to disambiguate between these possibilities.

Relationship between behavioral data and ERPs. It is reassuring to see that jointly training models to predict behavioral data along with a target ERP component benefits the prediction of the ERP component compared to training on the target ERP component alone. The benefit to prediction in this case cannot be explained as an artefact of how the ERP components are operationalized in the datasetes we use for analysis.

Self-paced reading times widely benefit ERP prediction, while eye-tracking data seems to have more limited benefit to just the ELAN, LAN, and PNP ERP components. It's difficult to know why this might be the case, but perhaps it is not a coincidence that these three ERP components also show up frequently in the pairs of components that benefit from joint training. If indeed the PNP marks semantic role irregularities (Van Petten and Luka, 2012) and the ELAN and LAN mark working memory or look-forward or look-back operations (Kutas et al., 2006), then its possible that eye-movements might be more related to these types of operations than to general semantic and syntactic complexities marked by other ERP components. Self-paced reading might better capture these generic difficulties. This explanation is highly speculative, and further work is required to determine whether the relationships between the ERP components and behavioral data are consistent across datasets, and if so, what the explanation is for these relationships.

Choice of bidirectional architecture. We emphasize that the neural network architecture we chose for these analyses was motivated primarily by its success on downstream NLP tasks, public availability of pre-trained models and code, and prior work studying how best to fine-tune the model (Howard and Ruder, 2018; Merity et al., 2017). We do not claim that this architecture reflects human processing. We experimented with a forward-only model variant of our analysis, and found that the bidirectional model predicts brain activity better than the forward-only version (see Appendix A). Although the bidirectional model has access to 'future' language input, it does not have access to future brain-activity, so the bidirectional model is not 'cheating' when it makes predictions. There are at least three possible explanations for why the bidirectional model performs better than the forward-only model. First, it is possible that when a human reads a sentence, he or she predicts the upcoming language input. Under this hypothesis, a model with access to the future language input can do a better job of predicting the current brain activity because the future language is reflected in that brain activity. Second, it is possible that a bidirectional model is simply able to produce better embeddings for each word in the input because it has more context than a forward-only model. For example, the bidirectional model might be (implicitly) better at anaphora resolution given more context. Under this hypothesis, the additional context given to the model partially compensates for its relative deficit of real-world knowledge compared to a human. Where a human can in many cases solve the anaphora resolution problem by using background knowledge and does not need to see the future language input, a model benefits from additional context. Finally, in our setup, the bidirectional model has more parameters than the forward-only model, and the additional degrees of freedom might give the model an advantage in predicting brain activity. Exploration of why the bidirectional model is better than the forward-only model is an interesting question, but it is left to future work. Additionally, as we noted earlier, the qualitative results of our analysis (e.g. how ERP components relate to each other) should be viewed with caution until they are replicated across multiple choices of architecture.

7 Conclusion

We have shown that ERP components can be predicted from neural networks pretrained as language models and fine-tuned to directly predict those components. To the best of our knowledge, prior work has not successfully used statistical models to predict all of these components. Furthermore, we have shown that multitask learning benefits the prediction of ERP components and can suggest how components relate to each other. At present, these joint-training benefit relationships are only suggestive, but if these relationships ultimately lead to insights about what drives each ERP component, then the components become more useful tools for studying human language comprehension. By using multitask learning as a method of characterization, we have found some expected relationships (LAN+P600 and ELAN+P600) and several more surprising relationships. We believe that this is exactly the kind of finding that makes multitask learning an interesting exploratory technique in this area. Additionally, we have shown that information can be shared between heterogeneous types of data (eye-tracking, self-paced reading, and ERP components) in the domain of human language processing prediction, and in particular between behavioral and neural data. Given the small datasets associated with human language processing, using heterogeneous data is a potentially major advantage of a multitask approach. In future work, we will further explore what information is encoded into the model representations when neural and behavioral data are used to train neural networks, and how these representations differ from the representations in a model trained on language alone.

Acknowledgments

We thank our reviewers for their valuable feedback. This work is supported in part by National Institutes of Health grant number U01NS098969.

References

Yoav Benjamini and Daniel Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188.

British National Consortium. 2005. The BNC Baby. `http://www.natcorp.ox.ac.uk/`.

Seana Coulson, Jonathan W King, and Marta Kutas. 1998. Expect the unexpected: Event-related brain response to morphosyntactic violations. *Language and cognitive processes*, 13(1):21–58.

Stefan L Frank, Irene Fernandez Monsalve, Robin L Thompson, and Gabriella Vigliocco. 2013. Reading time data for evaluating broad-coverage models of english sentence processing. *Behavior Research Methods*, 45(4):1182–1190.

Stefan L Frank, Leun J Otten, Giulia Galli, and Gabriella Vigliocco. 2015. The ERP response to the amount of information conveyed by words in sentences. *Brain and language*, 140:1–11.

Angela D Friederici. 2011. The brain basis of language processing: from structure to function. *Physiological reviews*, 91(4):1357–1392.

Thomas C Gunter, Laurie A Stowe, and Gusbertus Mulder. 1997. When syntax meets semantics. *Psychophysiology*, 34(6):660–676.

Peter Hagoort, Marlies Wassenaar, and Colin Brown. 2003a. Real-time semantic compensation in patients with agrammatic comprehension: Electrophysiological evidence for multiple-route plasticity. *Proceedings of the National Academy of Sciences*, 100(7):4340–4345.

Peter Hagoort, Marlies Wassenaar, and Colin M Brown. 2003b. Syntax-related erp-effects in dutch. *Cognitive Brain Research*, 16(1):38–50.

Anja Hahne and Angela D Friederici. 1999. Electrophysiological evidence for two steps in syntactic analysis: Early automatic and late controlled processes. *Journal of cognitive neuroscience*, 11(2):194–205.

Anja Hahne and Angela D Friederici. 2002. Differential task effects on semantic and syntactic processes as revealed by erps. *Cognitive Brain Research*, 13(3):339–356.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R Brennan. 2018. Finding syntax in human encephalography with beam search. *arXiv preprint arXiv:1806.04127*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Gregory Hickok and David Poeppel. 2007. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *arXiv preprint arXiv:1801.06146*.

Falk Huettig. 2015. Four central questions about prediction in language processing. *Brain Research*, 1626:118–135.

JA John and Norman R Draper. 1980. An alternative family of transformations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(2):190–197.

David Kemmerer. 2014. *Cognitive neuroscience of language*. Psychology Press.

Jonathan W King and Marta Kutas. 1995. Who did what and when? using word-and clause-level erps to monitor working memory usage in reading. *Journal of cognitive neuroscience*, 7(3):376–395.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gina R Kuperberg. 2007. Neural mechanisms of language comprehension: Challenges to syntax. *Brain research*, 1146:23–49.

Gina R Kuperberg, Tatiana Sitnikova, David Caplan, and Phillip J Holcomb. 2003. Electrophysiological distinctions in processing conceptual relationships within simple sentences. *Cognitive Brain Research*, 17(1):117–129.

Marta Kutas and Kara D Federmeier. 2011. Thirty years and counting: finding meaning in the n400 component of the event-related brain potential (ERP). *Annual review of psychology*, 62:621–647.

Marta Kutas, Cyma K Van Petten, and Robert Kluender. 2006. Psycholinguistics electrified ii (1994–2005). In *Handbook of Psycholinguistics (Second Edition)*, pages 659–724. Elsevier.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Maria Palolahti, Sakari Leino, Markus Jokela, Kreetta Kopra, and Petri Paavilainen. 2005. Event-related potentials suggest early interaction between syntax and semantics during on-line sentence comprehension. *Neuroscience Letters*, 384(3):222–227.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Matthew A Lambon Ralph, Karen Sage, Roy W Jones, and Emily J Mayberry. 2010. Coherent concepts are computed in the anterior temporal lobes. *Proceedings of the National Academy of Sciences*, 107(6):2717–2722.

Keith Rayner. 2009. Eye movements and attention in reading, scene perception, and visual search. *The quarterly journal of experimental psychology*, 62(8):1457–1506.

Keith Rayner and Alexander Pollatsek. 2006. Eye-movement control in reading. In *Handbook of Psycholinguistics (Second Edition)*, pages 613–657. Elsevier.

J. K. Rowling. 1999. *Harry Potter and the Sorcerer's Stone*, 1 edition, volume 1. Scholastic, New York.

Cyma Van Petten and Barbara J Luka. 2012. Prediction during language comprehension: Benefits, costs, and erp components. *International Journal of Psychophysiology*, 83(2):176–190.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243, Doha, Qatar. Association for Computational Linguistics.

A Appendix

Here we present a visualization (Figure 4) of the results presented in Table 1 of the main paper, and a visualization (Figure 5) of a more complete set of results from which the information in Table 2 of the main paper is drawn. We also show supplemental results for variants of our primary analysis on multitask learning with eye-tracking, self-paced reading time and ERP data. In the variants we modify the input representation to our decoder network to see whether the relationships between the behavioral data and neural activity appear to be consistent with different choices of encoder architectures. Additional (and more varied) choices or architectures are left to future work. The results in Table 3 reflect using only the forward-encoder (rather than the bi-LSTM) in the encoder network, while the results in Table 4 reflect using only the word embeddings (i.e. bypassing the LSTM entirely). While the results are clearly worse for each of these choices of architecture than for using a bi-LSTM encoder, the relationships between the behavioral data and the ERP signals is qualitatively similar. Finally, 5 shows the Pearson correlation coefficient between different measures. We note that the patterns of correlation are different than the patterns of which measures benefit from joint training with each other.

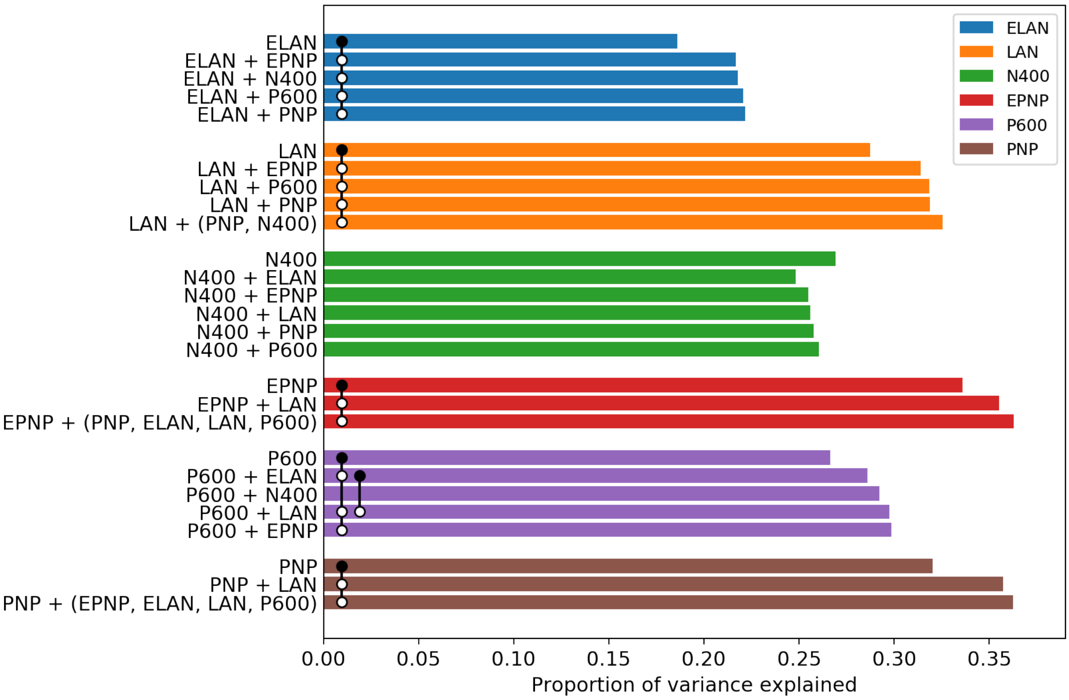


Figure 4: The proportion of variance explained for prediction of each of the ERP signals (mean of 100 training runs). The target ERP is indicated by color; each group of bars shows performance for a different target ERP. The top bar in each group shows the proportion of variance explained when the model is trained using only the target ERP. The bottom bar in each group shows the maximum proportion of variance explained over all combinations of training ERPs (or in the case of the N400, the second best). Also shown in each group are any training combinations that (i) used no more than the number of ERP signals used by the combination that achieved the maximum, and (ii) which were not significantly different from the maximum. Bars are statistically different from each other if a black dot on one bar is connected by a contiguous vertical line to a white dot on the other bar. The bars in the N400 group are not significantly different from each other. The N400 signal is best predicted when the model is trained on just that signal. In every other group, there is at least one ERP that, when combined with the target ERP during training, improves the prediction of the target ERP. The results suggest that these pairs are related: (LAN, P600), (LAN, EPNP), (LAN, PNP), (ELAN, N400), (ELAN, EPNP), (ELAN, PNP), (ELAN, P600), (EPNP, P600).

Target	Additional	POVE	Target	Additional	POVE	Target	Additional	POVE
ELAN		0.20	LAN		0.23	N400		0.20
ELAN	+ERP	**0.22**	LAN	+ ERP	**0.26**	N400	+ ERP	0.20
ELAN	+READ	**0.22**	LAN	+ READ	**0.25**	N400	+ READ	0.20
ELAN	+EYE	0.21	LAN	+ EYE	**0.24**	N400	+ EYE	0.18
EPNP		0.28	P600		0.24	PNP		0.28
EPNP	+ ERP	0.28	P600	+ ERP	**0.25**	PNP	+ ERP	**0.31**
EPNP	+ READ	**0.29**	P600	+ READ	**0.25**	PNP	+ READ	**0.30**
EPNP	+ EYE	0.29	P600	+ EYE	0.24	PNP	+ EYE	0.29

Table 3: Proportion of variance explained for each of the ERP components when using only the forward direction of the encoder (mean of 100 training runs). +ERP indicates the best combination of ERP training signals for the target ERP component, + READ indicates the inclusion of self-paced reading times, +EYE indicates the inclusion of eye-tracking data, and bold font indicates a significant difference from training on the target component alone.

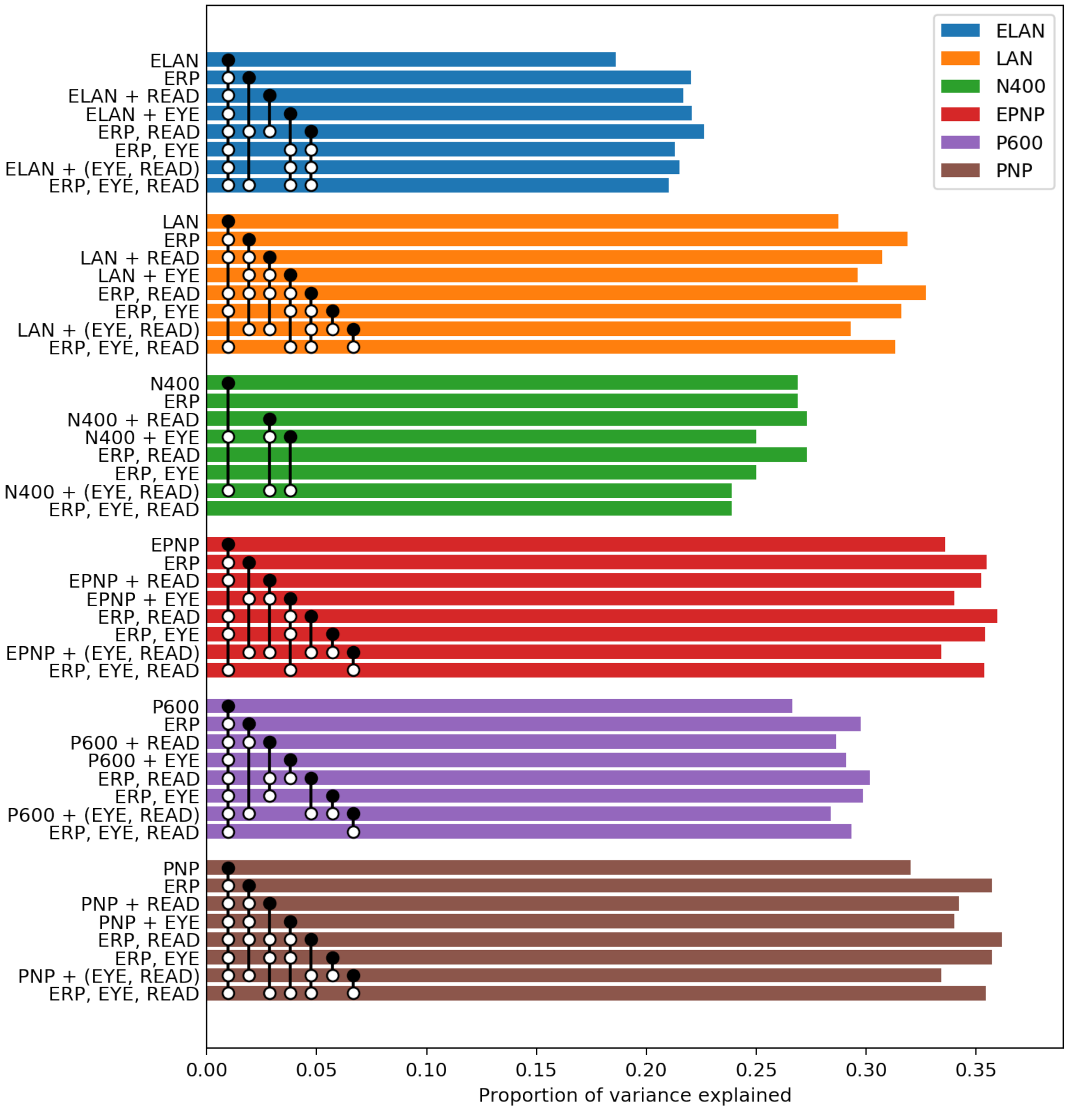


Figure 5: The proportion of variance explained for prediction of each of the ERP signals (mean of 100 training runs). The target ERP is indicated by color; each group of bars shows performance for a different target ERP. The top bar in each group shows the proportion of variance explained when the model is trained using only the target ERP. Moving down, the next bar in each group, labeled *ERP* shows the proportion of variance explained by the best combination of ERP signals for the target ERP. The other bars in each group moving from top to bottom show training variations that use behavioral data with either just the target ERP, or with the best combination of ERP signals. *READ* denotes self-paced reading data, and *EYE* denotes all four eye-tracking measures (in this analysis we use right-bounded pass time, gaze duration, go-past time, and first-fixation duration). Pairs of bars are significantly different from each other (paired t-test, false discovery rate ¡ 0.01) if a black dot on one bar is connected to a white dot on the other bar by a contiguous vertical line. Self-paced reading time benefits prediction of all target ERP components except the N400. In the case of the ELAN, LAN, and PNP, self-paced reading time also has marginal benefit compared to the best combination of ERP training signals. Eye-tracking data benefits prediction of the ELAN, P600, and PNP components.

Target	Additional	POVE	Target	Additional	POVE	Target	Additional	POVE
ELAN		0.15	LAN		0.17	N400		0.05
ELAN	+ ERP	**0.18**	LAN	+ ERP	**0.19**	N400	+ ERP	0.05
ELAN	+ READ	**0.18**	LAN	+ READ	**0.19**	N400	+ READ	0.08
ELAN	+ EYE	**0.19**	LAN	+ EYE	**0.19**	N400	+ EYE	0.10
EPNP		0.18	P600		0.10	PNP		0.20
EPNP	+ ERP	**0.20**	P600	+ ERP	**0.13**	PNP	+ ERP	**0.23**
EPNP	+ READ	**0.20**	P600	+ READ	**0.13**	PNP	+ READ	**0.22**
EPNP	+ EYE	**0.21**	P600	+ EYE	**0.14**	PNP	+ EYE	**0.23**

Table 4: Proportion of variance explained for each of the ERP components when using only the word embeddings as input to the decoder and bypassing the LSTM entirely (mean of 100 training runs). +ERP indicates the best combination of ERP training signals for the target ERP component, + READ indicates the inclusion of self-paced reading times, +EYE indicates the inclusion of eye-tracking data, and bold font indicates a significant difference from training on the target component alone.

Signal	ELAN	EPNP	LAN	N400	P600	PNP	FIX	PASS	GO	RIGHT	READ
ELAN	1.00	0.27	0.32	0.11	0.10	0.24	0.27	0.26	0.27	0.26	-0.04
EPNP	0.27	1.00	0.66	0.41	0.50	0.83	0.17	0.17	0.19	0.17	0.02
LAN	0.32	0.66	1.00	0.58	0.33	0.47	0.12	0.11	0.13	0.12	0.01
N400	0.11	0.41	0.58	1.00	0.47	0.33	-0.04	-0.04	-0.02	-0.03	0.11
P600	0.10	0.50	0.33	0.47	1.00	0.69	0.14	0.14	0.16	0.14	0.10
PNP	0.24	0.83	0.47	0.33	0.69	1.00	0.25	0.24	0.26	0.25	0.03
FIX	0.27	0.17	0.12	-0.04	0.14	0.25	1.00	1.00	1.00	1.00	0.04
PASS	0.26	0.17	0.11	-0.04	0.14	0.24	1.00	1.00	1.00	1.00	0.04
GO	0.27	0.19	0.13	-0.02	0.16	0.26	1.00	1.00	1.00	1.00	0.05
RIGHT	0.26	0.17	0.12	-0.03	0.14	0.25	1.00	1.00	1.00	1.00	0.04
READ	-0.04	0.02	0.01	0.11	0.10	0.03	0.04	0.04	0.05	0.04	1.00

Table 5: Raw Pearson's correlation coefficients (computed on content words after the standardization and participant-averaging) between each neural and behavioral measure and each other measure. FIX indicates first-fixation time, PASS indicates first-pass time, GO indicates go-past time, RIGHT indicates right-bounded reading time, and READ indicates self-paced reading. Many of the measures are highly correlated, but the pattern of correlations is different from the pattern of benefits that we find during joint-training. In particular we note that the N400 is correlated with the other ERP signals, and yet we do not see benefit in prediction of the N400 when jointly training a model to predict it and other signals.

Pre-training on High-Resource Speech Recognition Improves Low-Resource Speech-to-Text Translation

Sameer Bansal[1] **Herman Kamper**[2] **Karen Livescu**[3] **Adam Lopez**[1] **Sharon Goldwater**[1]
[1]School of Informatics, University of Edinburgh, UK
[2]E&E Engineering, Stellenbosch University, South Africa
[3]Toyota Technological Institute at Chicago, USA
`{sameer.bansal, sgwater, alopez}@inf.ed.ac.uk`
`kamperh@sun.ac.za`
`klivescu@ttic.edu`

Abstract

We present a simple approach to improve direct speech-to-text translation (ST) when the source language is low-resource: we pre-train the model on a high-resource automatic speech recognition (ASR) task, and then fine-tune its parameters for ST. We demonstrate that our approach is effective by pre-training on 300 hours of English ASR data to improve Spanish-English ST from 10.8 to 20.2 BLEU when only 20 hours of Spanish-English ST training data are available. Through an ablation study, we find that the pre-trained encoder (acoustic model) accounts for most of the improvement, despite the fact that the shared language in these tasks is the target language text, not the source language audio. Applying this insight, we show that pre-training on ASR helps ST even when the ASR language differs from both source and target ST languages: pre-training on French ASR also improves Spanish-English ST. Finally, we show that the approach improves performance on a true low-resource task: pre-training on a combination of English ASR and French ASR improves Mboshi-French ST, where only 4 hours of data are available, from 3.5 to 7.1 BLEU.

1 Introduction

Speech-to-text Translation (ST) has many potential applications for low-resource languages: for example in language documentation, where the source language is often unwritten or endangered (Besacier et al., 2006; Martin et al., 2015; Adams et al., 2016a,b; Anastasopoulos and Chiang, 2017); or in crisis relief, where emergency workers might need to respond to calls or requests in a foreign language (Munro, 2010). Traditional ST is a pipeline of automatic speech recognition (ASR) and machine translation (MT), and thus requires transcribed source audio to train ASR and parallel text to train MT. These resources are often unavailable for low-resource languages, but for our potential applications, there may be some source language audio paired with target language text translations. In these scenarios, end-to-end ST is appealing.

Recently, Weiss et al. (2017) showed that end-to-end ST can be very effective, achieving an impressive BLEU score of 47.3 on Spanish-English ST. But this result required over 150 hours of translated audio for training, still a substantial resource requirement. By comparison, a similar system trained on only 20 hours of data for the same task achieved a BLEU score of 5.3 (Bansal et al., 2018). Other low-resource systems have similarly low accuracies (Anastasopoulos and Chiang, 2018; Bérard et al., 2018).

To improve end-to-end ST in low-resource settings, we can try to leverage other data resources. For example, if we have transcribed audio in the source language, we can use multi-task learning to improve ST (Anastasopoulos and Chiang, 2018; Weiss et al., 2017; Bérard et al., 2018). But source language transcriptions are unlikely to be available in our scenarios of interest.

Could we improve low-resource ST by leveraging data from a high-resource language? For ASR, training a single model on multiple languages can be effective for all of them (Toshniwal et al., 2018b; Deng et al., 2013). For MT, *transfer learning* (Thrun, 1995) has been very effective: pre-training a model for a high-resource language pair and transferring its parameters to a low-resource language pair when the target language is shared (Zoph et al., 2016; Johnson et al., 2017). Inspired by these successes, we show that low-resource ST can leverage transcribed audio in a high-resource target language, or even a different language altogether, simply by pre-training a model for the high-resource ASR task, and then transferring and fine-tuning some or all of the model's parameters for low-resource ST.

Proceedings of NAACL-HLT 2019, pages 58–68
Minneapolis, Minnesota, June 2 - June 7, 2019.

We first test our approach using Spanish as the source language and English as the target. After training an ASR system on 300 hours of English, fine-tuning on 20 hours of Spanish-English yields a BLEU score of 20.2, compared to only 10.8 for an ST model without ASR pre-training. Analyzing this result, we discover that the main benefit of pre-training arises from the transfer of the *encoder* parameters, which model the input acoustic signal. In fact, this effect is so strong that we also obtain improvements by pre-training on a language that differs from both the source and the target: pre-training on French and fine-tuning on Spanish-English. We hypothesize that pre-training the encoder parameters, even on a different language, allows the model to better learn about linguistically meaningful phonetic variation while normalizing over acoustic variability such as speaker and channel differences. We conclude that the acoustic-phonetic learning problem, rather than translation itself, is one of the main difficulties in low-resource ST. A final set of experiments confirm that ASR pre-training also helps on another language pair where the input is truly low-resource: Mboshi-French.

2 Method

For both ASR and ST, we use an encoder-decoder model with attention adapted from Weiss et al. (2017), Bérard et al. (2018) and Bansal et al. (2018), as shown in Figure 1. We use the same model architecture for all our models, allowing us to conveniently transfer parameters between them. We also constrain the hyper-parameter search to fit a model into a single Titan X GPU, allowing us to maximize available compute resources.

We use a pre-trained English ASR model to initialize training of Spanish-English ST models, and a pre-trained French ASR model to initialize training of Mboshi-French ST models. During ST training, all model parameters are updated. In these configurations, the decoder shares the same vocabulary across the ASR and ST tasks. This is practical for settings where the target text language is high-resource with ASR data available.

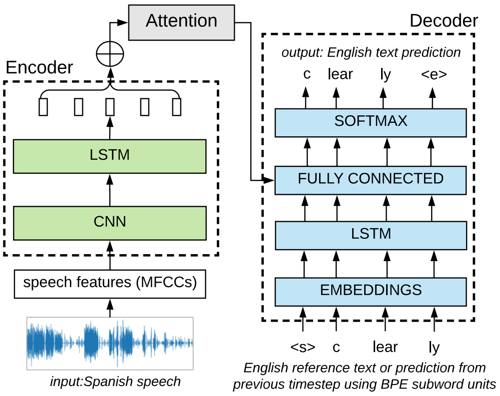


Figure 1: Encoder-decoder with attention model architecture for both ASR and ST. The encoder input is the Spanish speech utterance *claro*, translated as *clearly*, represented as BPE (subword) units.

In settings where both ST languages are low-resource, ASR data may only be available in a third language. To test whether transfer learning will help in this setting, we use a pre-trained French ASR model to train Spanish-English ST models; and English ASR for Mboshi-French models. In these cases, the ST languages are different from the ASR language, so we can only transfer the encoder parameters of the ASR model, since the dimensions of the decoder's output softmax layer are indexed by the vocabulary, which is not shared.[1] Sharing only the speech encoder parameters is much easier, since the speech input can be preprocessed in the same manner for all languages. This form of transfer learning is more flexible, as there are no constraints on the ASR language used.

3 Experimental Setup

3.1 Data sets

English ASR. We use the Switchboard Telephone speech corpus (Godfrey and Holliman, 1993), which consists of around 300 hours of English speech and transcripts, split into 260k utterances. The development set consists of 5 hours that we removed from the training set, split into 4k utterances.

French ASR. We use the French speech corpus from the GlobalPhone collection (Schultz, 2002), which consists of around 20 hours of high quality read speech and transcripts, split into 9k utterances. The development set consists of 2 hours, split into 800 utterances.

Spanish-English ST. We use the Fisher Spanish speech corpus (Graff et al., 2010), which consists of 160 hours of telephone speech in a variety of Spanish dialects, split into 140K utterances. To simulate low-resource conditions, we construct smaller train-

---

[1] Using a shared vocabulary of characters or subwords is an interesting direction for future work, but not explored here.

ing corpora consisting of 50, 20, 10, 5, or 2.5 hours of data, selected at random from the full training data. The development and test sets each consist of around 4.5 hours of speech, split into 4K utterances. We do not use the corresponding Spanish transcripts; our target text consists of English translations that were collected through crowdsourcing (Post et al., 2013, 2014).

Mboshi-French ST. Mboshi is a Bantu language spoken in the Republic of Congo, with around 160,000 speakers.[2] We use the Mboshi-French parallel corpus (Godard et al., 2018), which consists of around 4 hours of Mboshi speech, split into a training set of 5K utterances and a development set of 500 utterances. Since this corpus does not include a designated test set, we randomly sampled and removed 200 utterances from training to use as a development set, and use the designated development data as a test set.

3.2 Preprocessing

Speech. We convert raw speech input to 13-dimensional MFCCs using Kaldi (Povey et al., 2011).[3] We also perform speaker-level mean and variance normalization.

Text. The target text of the Spanish-English data set contains 1.5M word tokens and 17K word types. If we model text as sequences of words, our model cannot produce any of the unseen word types in the test data and is penalized for this, but it can be trained very quickly (Bansal et al., 2018). If we instead model text as sequences of characters as done by Weiss et al. (2017), we would have 7M tokens and 100 types, resulting in a model that is open-vocabulary, but very slow to train (Bansal et al., 2018). As an effective middle ground, we use byte pair encoding (BPE; Sennrich et al., 2016) to segment each word into subwords, each of which is a character or a high-frequency sequence of characters—we use 1000 of these high-frequency sequences. Since the set of subwords includes the full set of characters, the model is still open vocabulary; but it results in a text with only 1.9M tokens and just over 1K types, which can be trained almost as fast as the word-level model.

The vocabulary for BPE depends on the frequency of character sequences, so it must be computed with respect to a specific corpus. For English, we use the full 160-hour Spanish-English ST target training text. For French, we use the Mboshi-French ST target training text.

3.3 Model architecture for ASR and ST

Speech encoder. As shown schematically in Figure 1, MFCC feature vectors, extracted using a window size of 25 ms and a step size of 10ms, are fed into a stack of two CNN layers, with 128 and 512 filters with a filter width of 9 frames each. In each CNN layer we stride with a factor of 2 along time, apply a ReLU activation (Nair and Hinton, 2010), and apply batch normalization (Ioffe and Szegedy, 2015). The output of the CNN layers is fed into a three-layer bi-directional long short term memory network (LSTM; Hochreiter and Schmidhuber, 1997); each hidden layer has 512 dimensions.

Text decoder. At each time step, the decoder chooses the most probable token from the output of a softmax layer produced by a fully-connected layer, which in turn receives the current state of a recurrent layer computed from previous time steps and an attention vector computed over the input. Attention is computed using the *global attentional model* with *general* score function and *input-feeding*, as described in Luong et al. (2015). The predicted token is then fed into a 128-dimensional embedding layer followed by a three-layer LSTM to update the recurrent state; each hidden state has 256 dimensions. While training, we use the predicted token 20% of the time as input to the next decoder step and the training token for the remaining 80% of the time (Williams and Zipser, 1989). At test time we use beam decoding with a beam size of 5 and length normalization (Wu et al., 2016) with a weight of 0.6.

Training and implementation. Parameters for the CNN and RNN layers are initialized using the scheme from (He et al., 2015). For the embedding and fully-connected layers, we use Chainer's (Tokui et al., 2015) default initialition.

We regularize using dropout (Srivastava et al., 2014), with a ratio of 0.3 over the embedding and LSTM layers (Gal, 2016), and a weight decay rate of 0.0001. The parameters are optimized using Adam (Kingma and Ba, 2015), with a starting alpha of 0.001.

---

[2] `ethnologue.com/language/mdw`

[3] In preliminary experiments, we did not find much difference between between MFCCs and more raw spectral representations like Mel filterbank features.

Following some preliminary experimentation on our development set, we add Gaussian noise with standard deviation of 0.25 to the MFCC features during training, and drop frames with a probability of 0.10. After 20 epochs, we corrupt the true decoder labels by sampling a random output label with a probability of 0.3.

Our code is implemented in Chainer (Tokui et al., 2015) and is freely available.[4]

3.4 Evaluation

Metrics. We report BLEU (Papineni et al., 2002) for all our models.[5] In low-resource settings, BLEU scores tend to be low, difficult to interpret, and poorly correlated with model performance. This is because BLEU requires exact four-gram matches only, but low four-gram accuracy may obscure a high unigram accuracy and inexact translations that partially capture the semantics of an utterance, and these can still be very useful in situations like language documentation and crisis response. Therefore, we also report word-level unigram precision and recall, taking into account *stem*, *synonym*, and *paraphrase* matches. To compute these scores, we use METEOR (Lavie and Agarwal, 2007) with default settings for English and French.[6] For example, METEOR assigns "eat" a recall of 1 against reference "eat" and a recall of 0.8 against reference "feed", which it considers a synonym match.

Naive baselines. We also include evaluation scores for a naive baseline model that predicts the *K* most frequent words of the training set as a bag of words for each test utterance. We set *K* to be the value at which precision/recall are most similar, which is always between 5 and 20 words. This provides an empirical lower bound on precision and recall, since we would expect any usable model to outperform a system that does not even depend on the input utterance. We do not compute BLEU for these baselines, since they do not predict sequences, only bags of words.

4 ASR results

Using the experimental setup of Section 3, we pretrained ASR models in English and French, and report their word error rates (WER) on development data in Table 1.[7] We denote each ASR model by *L-Nh*, where *L* is a language code and *N* is the size of the training set in hours. For example, *en-300h* denotes an English ASR model trained on 300 hours of data.

	en-100h	**en-300h**	**fr-20h**
WER	35.4	27.3	29.6

Table 1: Word Error Rate (WER, in %) for the ASR models used as pretraining, computed on Switchboard *train-dev* for English and Globalphone dev for French.

Training ASR models for state-of-the-art performance requires substantial hyper-parameter tuning and long training times. Since our goal is simply to see whether pre-training is useful, we stopped pretraining our models after around 30 epochs (3 days) to focus on transfer experiments. As a consequence, our ASR results are far from state-of-the-art: current end-to-end Kaldi systems obtain 16% WER on Switchboard *train-dev*, and 22.7% WER on the French Globalphone dev set.[8] We believe that better ASR pre-training may produce better ST results, but we leave this for future work.

5 Spanish-English ST

In the following, we denote an ST model by *S-T-Nh*, where *S* and *T* are source and target language codes, and *N* is the size of the training set in hours. For example, *sp-en-20h* denotes a Spanish-English ST model trained using 20 hours of data. We use the code *mb* for Mboshi and *fr* for French.

5.1 Using English ASR to improve ST

Figure 2 shows the BLEU and unigram precision/recall scores on the development set for baseline Spanish-English ST models and those trained after initializing with the *en-300h* model. Corresponding results on the test set (Table 2) reveal very similar patterns. The remainder of our analysis is confined to the development set. The naive baseline, which predicts the 15 most frequent English words in the training set, achieves a precision/recall of around 20%, setting a performance lower bound.

Low-resource: 20-50 hours of ST training data. Our baseline ST models substantially improve over

---

[4] `github.com/0xSameer/ast`

[5] We compute BLEU with `multi-bleu.pl` from the Moses toolkit (Koehn et al., 2007).

[6] `cs.cmu.edu/~alavie/METEOR`

[7] We computed WER with the NIST `sclite` script.

[8] These WER results taken from respective Kaldi recipes on GitHub, and may not represent the very best results on these data sets.

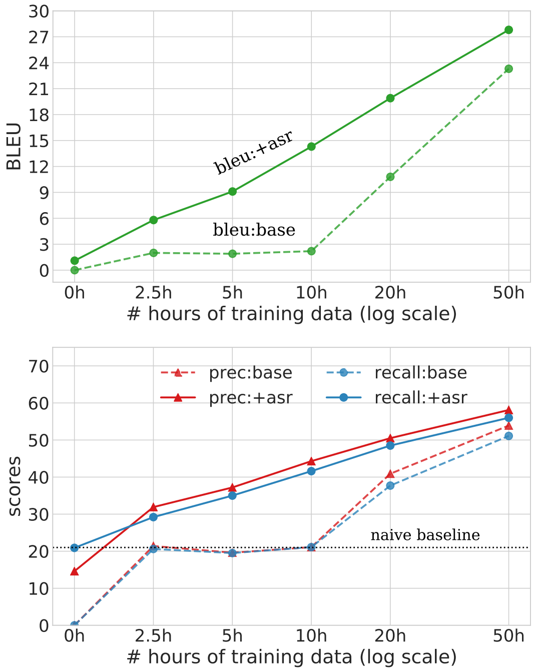


Figure 2: (top) BLEU and (bottom) Unigram precision/recall for Spanish-English ST models computed on Fisher dev set. base indicates no transfer learning; +asr are models trained by fine-tuning *en-300h* model parameters. *naive baseline* indicates the score when we predict the 15 most frequent English words in the training set.

$N =$	0	2.5	5	10	20	50
base	0	2.1	1.8	2.1	10.8	22.7
+asr	0.5	5.7	9.1	14.5	20.2	28.2

Table 2: BLEU scores for Spanish-English ST on the Fisher test set, using N hours of training data. base: no transfer learning. +asr: using model parameters from English ASR (en-300h).

Spanish	super caliente pero muy bonito
English	super hot but very nice
20h	you support it but it was very nice
20h+asr	you can get alright but it's very nice
50h	super expensive but very nice
50h+asr	super hot but it's very nice
Spanish	sí y usted hace mucho tiempo que que vive aquí
English	yes and have you been living here a long time
20h	yes i've been a long time what did you come here
20h+asr	yes and you have a long time that you live here
50h	yes you are a long time that you live here
50h+asr	yes and have you been here long

Table 3: Example translations on selected sentences from the Fisher development set, with stem-level n-gram matches to the reference sentence underlined. 20h and 50h are Spanish-English models without pretraining; 20h+asr and 50h+asr are pre-trained on 300 hours of English ASR.

previous results (Bansal et al., 2018) using the same train/test splits, primarily due to better regularization and modeling of subwords rather than words. Yet transfer learning still substantially improves over these strong baselines. For *sp-en-20h*, transfer learning improves dev set BLEU from 10.8 to 19.9, precision from 41% to 51%, and recall from 38% to 49%. For *sp-en-50h*, transfer learning improves BLEU from 23.3 to 27.8, precision from 54% to 58%, and recall from 51% to 56%.

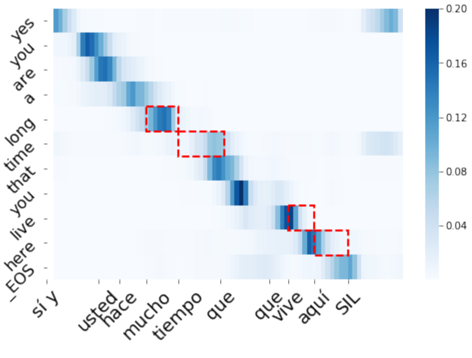
Very low-resource: 10 hours or less of ST training data. Figure 2 shows that without transfer learning, ST models trained on less than 10 hours of data struggle to learn, with precision/recall scores close to or below that of the naive baseline. But with transfer learning, we see gains in precision and recall of between 10 and 20 points.

We also see that with transfer learning, a model trained on only 5 hours of ST data achieves a BLEU of 9.1, nearly as good as the 10.8 of a model trained on 20 hours of ST data without transfer learning. In other words, fine-tuning an English ASR model—which is relatively easy to obtain—produces similar results to training an ST model on four times as much data, which may be difficult to obtain.

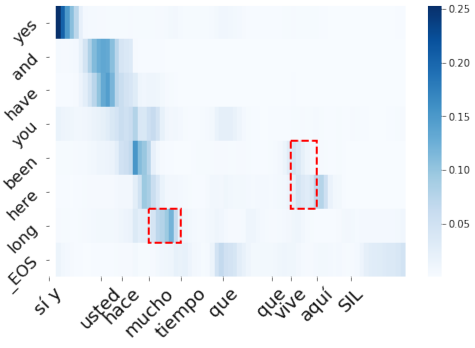
We even find that in the very low-resource setting of just 2.5 hours of ST data, with transfer learning the model achieves a precision/recall of around 30% and improves by more than 10 points over the naive baseline. In very low-resource scenarios with time constraints—such as in disaster relief—it is possible that even this level of performance may be useful, since it can be used to spot keywords in speech and can be trained in just three hours.

Sample translations. Table 3 shows example translations for models *sp-en-20h* and *sp-en-50h* with and without transfer learning using *en-300h*.

Figure 3 shows the attention weights for the last sample utterance in Table 3. For this utterance, the Spanish and English text have a different word order: *mucho tiempo* occurs in the middle of the speech utterance, and its translation, *long time*, is at the end of the English reference. Similarly, *vive aquí* occurs at the end of the speech utterance, while the translation, *living here*, is in the middle of the English reference. The baseline *sp-en-50h* model translates the words correctly but doesn't get



(a) **50h:baseline**



(b) **50h:asr**

Figure 3: Attention plots for the final example in Table 3, using 50h models with and without pre-training. The x-axis shows the reference Spanish word positions in the input; the y-axis shows the predicted English subwords. In the reference, *mucho tiempo* is translated to *long time*, and *vive aquí* to *living here*, but their order is reversed, and this is reflected in (b).

the English word order right. With transfer learning, the model produces a shorter but still accurate translation in the correct word order.

5.2 Analysis

To understand the source of these improvements, we carried out a set of ablation experiments. For most of these experiments, we focus on Spanish-English ST with 20 hours of training data, with and without transfer learning.

Transfer learning with selected parameters. In our first set of experiments, we transferred all parameters of the *en-300h* model, including the speech encoder CNN and LSTM; the text decoder embedding, LSTM and output layer parameters; and attention parameters. To see which set of parameters has the most impact, we train the *sp-en-20h* model by transferring only selected parameters from *en-300h*, and randomly initializing the rest.

The results (Figure 4) show that transferring all parameters is most effective, and that the speech encoder parameters account for most of the gains. We hypothesize that the encoder learns transferable low-level acoustic features that normalize across variability like speaker and channel differences to better capture meaningful phonetic differences, and that much of this learning is language-independent. This hypothesis is supported by other work showing the benefits of cross-lingual and multilingual training for speech technology in low-resource target languages (Carlin et al., 2011; Jansen et al., 2010; Deng et al., 2013; Vu et al., 2012; Thomas et al., 2012; Cui et al., 2015; Alumäe et al., 2016; Yuan et al., 2016; Renshaw et al., 2015; Hermann and Goldwater, 2018).

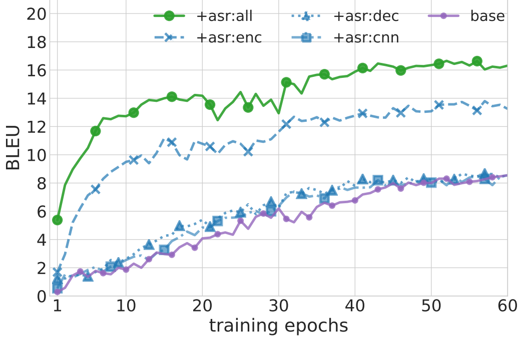


Figure 4: Fisher development set training curves (reported using BLEU) for *sp-en-20h* using selected parameters from *en-300h*: none (base); encoder CNN only (+asr:cnn); encoder CNN and LSTM only (+asr:enc); decoder only (+asr:dec); and all: encoder, attention, and decoder (+asr:all). These scores do not use beam search and are therefore lower than the best scores reported in Figure 2.

By contrast, transferring only decoder parameters does not improve accuracy. Since decoder parameters help when used in tandem with encoder parameters, we suspect that the dependency in parameter training order might explain this: the transferred decoder parameters have been trained to expect particular input representations from the encoder, so transferring only the decoder parameters without the encoder might not be useful.

Figure 4 also suggests that models make strong gains early on in the training when using transfer learning. The *sp-en-20h* model initialized with all model parameters (*+asr:all*) from *en-300h* reaches a higher BLEU score after just 5 epochs (2 hours) of training than the model without transfer learning trained for 60 epochs/20 hours. This again can be useful in disaster-recovery scenarios, where the

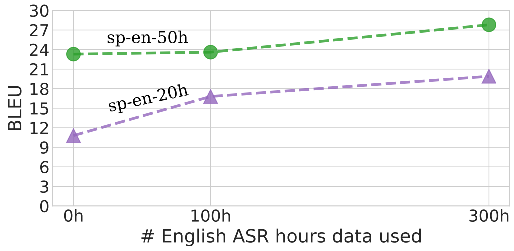


Figure 5: Spanish-to-English BLEU scores on Fisher dev set, with 0h (no transfer learning), 100h and 300h of English ASR data used.

time to deploy a working system must be minimized.

Amount of ASR data required. Figure 5 shows the impact of increasing the amount of English ASR data used on Spanish-English ST performance for two models: *sp-en-20h* and *sp-en-50h*.

For *sp-en-20h*, we see that using *en-100h* improves performance by almost 6 BLEU points. By using more English ASR training data (*en-300h*) model, the BLEU score increases by almost 9 points. However, for *sp-en-50h*, we only see improvements when using *en-300h*. This implies that transfer learning is most useful when only a few tens of hours of training data are available for ST. As the amount of ST training data increases, the benefits of transfer learning tail off, although it's possible that using even more monolingual data, or improving the training at the ASR step, could extend the benefits to larger ST data sets.

Impact of code-switching. We also tried using the *en-300h* ASR model without any fine-tuning to translate Spanish audio to English text. This model achieved a BLEU score of 1.1, with a precision of 15 and recall of 21. The non-zero BLEU score indicates that the model is matching *some* 4-grams in the reference. This seems to be due to code-switching in the Fisher-Spanish speech data set. Looking at the dev set utterances, we find several examples where the Spanish transcriptions match the English translations, indicating that the speaker switched into English. For example, there is an utterance whose Spanish transcription and English translation are both "right yeah", and this English expression is indeed present in the source audio. The English ASR model correctly translates this utterance, which is unsurprising since the phrase "right yeah" occurs nearly 500 times in Switchboard.

Overall, we find that in nearly 500 of the 4,000 development set utterances (14%), the Spanish transcription and English translations share more than half of their tokens, indicating likely code-switching. This suggests that transfer learning from English ASR models might help more than from other languages. To isolate this effect from transfer learning of language-independent speech features, we carried out a further experiment.

5.3 Using French ASR to improve Spanish-English ST

In this experiment, we pre-train using French ASR data for a Spanish-English translation task. Here, we can only transfer the speech encoder parameters, and there should be little if any benefit due to code-switching.

Because our French data set (20 hours) is much smaller than our English one (300 hours), for a fair comparison we used a 20 hour subset of the English data for pre-training in this experiment. For both the English and French models, we transferred only the encoder parameters.

Table 4 shows that both the English and French 20-hour pre-trained models improve performance on Spanish-English ST. The English model works slightly better, as would be predicted given our discussion of code-switching, but the French model is also useful, improving BLEU from 10.8 to 12.5. This result strengthens the claim that ASR pre-training on a completely distinct third language can help low-resource ST. Presumably benefits would be much greater if we used a larger ASR data set, as we did with English above.

In this experiment, the French pre-trained model used a French BPE output vocabulary, distinct from the English BPE vocabulary used in the ST system. In the future it would be interesting to try combining the French and English text to create a combined output vocabulary, which would allow transferring both the encoder and decoder parameters, and may be useful for translating names or cognates. More generally, it would also be possible to pre-train on multiple languages simultaneously using a shared BPE vocabulary. There is evidence that speech features trained on multiple languages transfer better than those trained on the same amount of data from a single language (Hermann and Goldwater, 2018), so multilingual pretraining for ST could improve results.

	baseline	**+fr-20h**	**+en-20h**
sp-en-20h	10.8	12.5	13.2

Table 4: Fisher dev set BLEU scores for *sp-en-20h*. baseline: model without transfer learning. Last two columns: Using encoder parameters from French ASR (+fr-20h), and English ASR (+en-20h).

model	**pretrain**	**BLEU**	**Pr.**	**Rec.**
fr-top-8w	–	0	23.5	22.2
fr-top-10w	–	0	20.6	24.5
en-300h	–	0	0.2	5.7
fr-20h	–	0	4.1	3.2
mb-fr-4h	–	3.5	18.6	19.4
	fr-20h	5.9	23.6	20.9
	en-300h	5.3	23.5	22.6
	en + fr	**7.1**	**26.7**	**23.1**

Table 5: Mboshi-to-French translation scores, with and without ASR pre-training. Pr. is the precision, and Rec. the recall score. fr-top-8w and fr-top-10w are *naive baselines* that, respectively, predict the 8 or 10 most frequent training words. For en + fr, we use encoder parameters from *en-300h* and attention+decoder parameters from *fr-20h*

6 Mboshi-French ST

Our final set of experiments test our transfer method on ST for the low-resource language Mboshi, where we have only 4 hours of ST training data: Mboshi speech input paired with French text output.

Table 5 shows the ST model scores for Mboshi-French with and without using transfer learning. The first two rows *fr-top-8w*, *fr-top-10w*, show precision and recall scores for the *naive baselines* where we predict the top 8 or 10 most frequent French words in the Mboshi-French training set. These show that a precision/recall in the low 20s is easy to achieve, although with no n-gram matches (0 BLEU). The pre-trained ASR models by themselves (next two lines) are much worse.

The baseline model trained only on ST data actually has lower precision/recall than the naive baseline, although its non-zero BLEU score indicates that it is able to correctly predict some n-grams. We see comparable precision/recall to the naive baseline with improvements in BLEU by transferring either French ASR parameters (both encoder and decoder, *fr-20h*) or English ASR parameters (encoder only, *en-300h*).

Finally, to achieve the benefits of both the larger training set size for the encoder and the matching language of the decoder, we tried transferring the encoding parameters from the *en-300h* model and the decoding parameters from the *fr-20h* model. This configuration (*en*+*fr*) gives us the best evaluation scores on all metrics, and highlights the flexibility of our framework. Nevertheless, the 4-hour scenario is clearly a very challenging one.

7 Conclusion

This paper introduced the idea of pre-training an end-to-end speech translation system involving a low-resource language using ASR training data from a higher-resource language. We showed that large gains are possible: for example, we achieved an improvement of 9 BLEU points for a Spanish-English ST model with 20 hours of parallel data and 300 hours of English ASR data. Moreover, the pre-trained model trains faster than the baseline, achieving higher BLEU in only a couple of hours, while the baseline trains for more than a day.

We also showed that these methods can be used effectively on a real low-resource language, Mboshi, with only 4 hours of parallel data. The very small size of the data set makes the task challenging, but by combining parameters from an English encoder and French decoder, we outperformed baseline models to obtain a BLEU score of 7.1 and precision/recall of about 25%. We believe ours is the first paper to report word-level BLEU scores on this data set.

Our analysis indicates that, other things being equal, transferring both encoder and decoder parameters works better than just transferring one or the other. However, transferring the encoder parameters is where most of the benefit comes from. Pre-training using a large ASR corpus from a mismatched language will therefore probably work better than using a smaller ASR corpus that matches the output language.

Our analysis suggests several avenues for further exploration. On the speech side, it might be even more effective to use multilingual training; or to replace the MFCC input features with pre-trained multilingual features, or features that are targeted to low-resource multispeaker settings (Kamper et al., 2015, 2017; Thomas et al., 2012; Cui et al., 2015; Yuan et al., 2016; Renshaw et al., 2015). On the language modeling side, simply transferring decoder parameters from an ASR model did not work; it might work better to use pre-trained decoder parameters from a language model, as proposed by Ramachandran et al. (2017), or *shallow fusion* (Gülçehre et al., 2015; Toshniwal et al., 2018a), which interpolates a pre-trained language model during beam search. In these methods, the decoder parameters are independent, and can therefore be used on their own. We plan to explore these strategies in future work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. This work was supported in part by a James S McDonnell Foundation Scholar Award, a Google faculty research award, and NSF grant 1816627. We thank Ida Szubert and Clara Vania for helpful comments on previous drafts of this paper and Antonios Anastasopoulos for tips on experimental setup.

References

Oliver Adams, Graham Neubig, Trevor Cohn, and Steven Bird. 2016a. Learning a translation model from word lattices. In *Proc. Interspeech*.

Oliver Adams, Graham Neubig, Trevor Cohn, Steven Bird, Quoc Truong Do, and Satoshi Nakamura. 2016b. Learning a lexicon and translation model from phoneme lattices. In *Proc. EMNLP*.

Tanel Alumäe, Stavros Tsakalidis, and Richard M Schwartz. 2016. Improved multilingual training of stacked neural network acoustic models for low resource languages. In *Proc. Interspeech*.

Antonios Anastasopoulos and David Chiang. 2017. A case study on using speech-to-translation alignments for language documentation. In *Proc. ACL*.

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proc. NAACL HLT*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In *Proc. Interspeech*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *Proc. ICASSP*.

Laurent Besacier, Bowen Zhou, and Yuqing Gao. 2006. Towards speech translation of non written languages. In *Proc. SLT*.

Michael A Carlin, Samuel Thomas, Aren Jansen, and Hynek Hermansky. 2011. Rapid evaluation of speech representations for spoken term discovery. In *Proc. Interspeech*.

Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, Abhinav Sethy, Kartik Audhkhasi, Xiaodong Cui, Ellen Kislal, Lidia Mangu, Markus Nussbaum-Thom, Michael Picheny, et al. 2015. Multilingual representations for low resource speech recognition and keyword search. In *Proc. ASRU*.

Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Mike Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, Yifan Gong, and Alex Acero. 2013. Recent advances in deep learning for speech research at Microsoft. In *Proc. ICASSP*.

Yarin Gal. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. NIPS*.

Pierre Godard, Gilles Adda, Martine Adda-Decker, Juan Benjumea, Laurent Besacier, Jamison Cooper-Leavitt, Guy-Noël Kouarata, Lori Lamel, Hélène Maynard, Markus Müller, Annie Rialland, Sebastian Stüker, François Yvon, and Marcely Zanon Boito. 2018. A very low resource language speech corpus for computational language documentation experiments. In *Proc. LREC*.

John Godfrey and Edward Holliman. 1993. Switchboard-1 Release 2 (LDC97S62). `https://catalog.ldc.upenn.edu/ldc97s62`.

David Graff, Shudong Huang, Ingrid Cartagena, Kevin Walker, and Christopher Cieri. 2010. Fisher Spanish Speech (LDC2010S01). `https://catalog.ldc.upenn.edu/ldc2010s01`.

Caglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loıc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. ICCV*.

Enno Hermann and Sharon Goldwater. 2018. Multilingual bottleneck features for subword modeling in zero-resource languages. In *Proc. Interspeech*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*.

Aren Jansen, Kenneth Church, and Hynek Hermansky. 2010. Towards spoken term discovery at scale with zero resources. In *Proc. Interspeech*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Trans. ACL*, 5:339–351.

Herman Kamper, Micha Elsner, Aren Jansen, and Sharon Goldwater. 2015. Unsupervised neural network based feature extraction using weak top-down constraints. In *Proc. ICASSP*.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2017. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Comput. Speech Lang.*, 46:154–174.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proc. WMT*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*.

Lara J Martin, Andrew Wilkinson, Sai Sumanth Miryala, Vivian Robison, and Alan W Black. 2015. Utterance classification in speech-to-speech translation for zero-resource languages in the hospital administration domain. In *Proc. ASRU*.

Robert Munro. 2010. Crowdsourced translation for emergency response in Haiti: The global collaboration of local knowledge. In *AMTA Workshop Collaborative Crowdsourcing Transl.*

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proc. ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the Fisher and Callhome Spanish-English speech translation corpus. In *Proc. IWSLT*.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2014. Fisher and CALLHOME Spanish–English Speech Translation. `https://catalog.ldc.upenn.edu/ldc2014t23`.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proc. EMNLP*.

Daniel Renshaw, Herman Kamper, Aren Jansen, and Sharon Goldwater. 2015. A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge. In *Proc. Interspeech*.

Tanja Schultz. 2002. Globalphone: a multilingual speech and text database developed at karlsruhe university. In *Seventh International Conference on Spoken Language Processing*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*

Samuel Thomas, Sriram Ganapathy, and Hynek Hermansky. 2012. Multilingual mlp features for low-resource LVCSR systems. In *Proc. ICASSP*.

Sebastian Thrun. 1995. Is learning the n-th thing any easier than learning the first? In *Proc. NIPS*.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: A next-generation open source framework for deep learning. In *Proc. LearningSys*.

Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, and Karen Livescu. 2018a. A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition. In *Proc. SLT*.

Shubham Toshniwal, Tara N. Sainath, Ron J. Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. 2018b. Multilingual Speech Recognition with A Single End-To-End Model. In *Proc. ICASSP*.

Ngoc Thang Vu, Wojtek Breiter, Florian Metze, and Tanja Schultz. 2012. An investigation on initialization schemes for multilayer perceptron training using multilingual data and their effect on ASR performance. In *Proc. Interspeech*.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. In *Proc. Interspeech*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yougen Yuan, Cheung-Chi Leung, Lei Xie, Bin Ma, and Haizhou Li. 2016. Learning neural network representations using cross-lingual bottleneck features with word-pair information. In *Proc. Interspeech*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proc. EMNLP*.

Measuring the Perceptual Availability of Phonological Features During Language Acquisition Using Unsupervised Binary Stochastic Autoencoders

Cory Shain
Department of Linguistics
The Ohio State University
`shain.3@osu.edu`

Micha Elsner
Department of Linguistics
The Ohio State University
`melsner@ling.osu.edu`

Abstract

In this paper, we deploy binary stochastic neural autoencoder networks as models of infant language learning in two typologically unrelated languages (Xitsonga and English). We show that the drive to model auditory percepts leads to latent clusters that partially align with theory-driven phonemic categories. We further evaluate the degree to which theory-driven phonological features are encoded in the latent bit patterns, finding that some (e.g. [±approximant]), are well represented by the network in both languages, while others (e.g. [±spread glottis]) are less so. Together, these findings suggest that many reliable cues to phonemic structure are immediately available to infants from bottom-up perceptual characteristics alone, but that these cues must eventually be supplemented by top-down lexical and phonotactic information to achieve adult-like phone discrimination. Our results also suggest differences in degree of perceptual availability between features, yielding testable predictions as to which features might depend more or less heavily on top-down cues during child language acquisition.

1 Introduction

Distinctive features like [±voice] and [±sonorant] have been a core construct of phonological theory for many decades (Trubetskoy, 1939; Jakobson et al., 1951; Chomsky and Halle, 1968; Clements, 1985). They have been used in automatic speech recognition (Livescu and Glass, 2004), and psycholinguistic evidence suggests that they are cognitively available during language acquisition (Kuhl, 1980; White and Morgan, 2008). Nonetheless, distinctive features are not directly observed by humans; they are abstractions that must be inferred from dense perceptual information (sound waves) during language acquisition and comprehension, which raises questions about how they are learned and recognized. In adults, phonological comprehension is aided by top-down lexical and phonotactic (i.e. sound sequencing) constraints. For example, the classic *phonemic restoration effect* (Warren, 1970) shows that adults infer missing phonemes from context with such ease that they often fail to notice when acoustic cues to phone identity are erased. However, infants first learning their phonemic categories have not yet acquired reliable top-down lexical and phonotactic models and must rely more heavily on bottom-up perceptual information. To a learner faced with the immense challenge of discovering structure in dense perceptual input, do theory-driven phonological features “stand out” or are they swamped by noise? In this paper, we address this question using an unsupervised computational acquisition model.

Previous models of phonological category induction have emphasized the importance of top-down information (information about the contexts in which phonemes occur) (Peperkamp et al., 2006; Swingley, 2009; Feldman et al., 2009a, 2013a,b; Moreton and Pater, 2012a,b; Martin et al., 2013; Pater and Moreton, 2014; Frank et al., 2014; Doyle et al., 2014; Doyle and Levy, 2016). But to prevent the acquisition process from being circular, the learner cannot operate solely on top-down information — the acoustic signal must provide some evidence for the phonemic categories. We hypothesize that the same must be true for at least some phonological features (e.g. [±nasal], [±lateral]), but previous work on unsupervised speech processing has inferred phonological structure from spoken utterances using either (1) discrete transition-based architectures (Varadarajan et al., 2008; Jansen and Church, 2011; Lee and Glass, 2012), which do attempt to discover featurally-related natural classes, or (2) continuous deep neural (Kamper et al., 2015,

Proceedings of NAACL-HLT 2019, pages 69–85
Minneapolis, Minnesota, June 2 - June 7, 2019.

2017a; Renshaw et al., 2015) architectures, whose internal representations are difficult to interpret. Furthermore, these approaches do not separate the contributions of top-down sequential information from bottom-up acoustic properties of segments, making it difficult to assess the relative importance of these information sources throughout the acquisition process.

By contrast, our model attends exclusively to phone-internal acoustic patterns using a deep neural autoencoder with a discrete embedding space composed of binary stochastic neurons (BSNs) (Rosenblatt, 1958; Hinton, 2012; Bengio et al., 2013; Courbariaux et al., 2016). BSNs allow us to exploit (1) the interpretability of discrete representations, (2) the decomposability of phone segments into phonological features, and (3) and the power of deep neural function approximators to relate percepts and their representations. Since every token is labeled with a binary latent code, it is possible to evaluate the model's recovery not only of phonological categories but also of phonological features. Featural representations can encode distributional facts about which processes apply to which classes of sounds in ways that cross-cut the phonological space, rather than simply grouping each segment with a set of similar neighbors (LeCun et al., 2015). By focusing on the acoustic properties of sounds themselves rather than their sequencing in context, our model enables exploration of two questions about the data available to young learners whose training signal must primarily be extracted from bottom-up perceptual information: (1) to what extent can phoneme categories emerge from a drive to model auditory percepts, and (2) how perceptually available are theory-driven phonological features (that is, how easily can they be extracted directly from low-level acoustic percepts)?

Our results show (a) that phonemic categories emerge naturally but imperfectly from perceptual reconstruction and (b) that theory-driven features differ in their degree of perceptual availability. Together, these findings suggest that many reliable cues to phonemic structure are immediately available to infants from bottom-up perceptual characteristics alone, but that these cues may eventually need to be supplemented by top-down lexical and phonotactic information to achieve adult-like phone discrimination (Feldman et al., 2013a; Pater and Moreton, 2014). Our findings also suggest hypotheses as to precisely which kinds of phonological features are more or less perceptually available and therefore might depend more or less heavily on top-down cues for acquisition. Such differences might suggest relative timelines at which different features might be appropriated in support of phonemic, phonotactic, and lexical generalization, providing a rich set of testable hypotheses about child language acquisition.

2 Background and Related Work

2.1 Unsupervised Speech Processing

The present paper has a strong connection to recent work on unsupervised speech processing, especially the Zerospeech 2015 (Versteegh et al., 2015) and 2017 (Dunbar et al., 2017) shared tasks. Participating systems (Badino et al., 2015; Renshaw et al., 2015; Agenbag and Niesler, 2015; Chen et al., 2015; Baljekar et al., 2015; Räsänen et al., 2015; Lyzinski et al., 2015; Zeghidour et al., 2016; Heck et al., 2016; Srivastava and Shrivastava, 2016; Kamper et al., 2017b; Chen et al., 2017; Yuan et al., 2017; Heck et al., 2017; Shibata et al., 2017; Ansari et al., 2017a,b) perform unsupervised ABX discrimination and/or spoken term discovery on the basis of unlabeled speech alone. The design and evaluation of these and related systems (Kamper et al., 2015, 2017a; Elsner and Shain, 2017; Räsänen et al., 2018) are oriented toward word-level modeling. As such, our focus on the perceptual availability of phonological features is orthogonal to but complementary with this line of research. Since distinctive features are important for indexing lexical contrasts, especially between highly confusable words (e.g. onset voicing alone distinguishes *sap* and *zap* in English), studying the perceptual availability of distinctive features to an unsupervised learner may help improve the design and analysis of low-resource speech processing systems.

To our knowledge, the task most closely related to the current paper is unsupervised phone discovery. Some studies in this tradition segment speech into phone-like units without clustering them (Dusan and Rabiner, 2006; Qiao et al., 2008), while others cluster small subsets of pre-segmented sounds (usually vowels) using parametric models (mixture-of-Gaussians) (Vallabha et al., 2007; Feldman et al., 2013a; Antetomaso et al., 2017). Further work combines these tasks and extends the approach to cover the entire acoustic space (Lee and Glass, 2012). However, for a variety of reasons, the Lee and Glass (2012) model does not straightforwardly support evaluation of the perceptual availability of phonological features. First, they do not quantitatively evaluate the discovered phoneme clusters. Second, the model incorporates phonotactics through transition probabilities, making it difficult to disentangle the contributions of top-down and bottom-up information to the learning process. Third, the clustering model is not feature-based, but instead consists of atomic categories, each defining a distinct generative process for acoustics. This design is at odds with the widely held view in linguistic theory that phonemes are not inscrutable atoms of the phonological grammar, but instead labels for bundles of features that define natural classes (Clements, 1985). Our approach is therefore more appropriate to the question at hand.

2.2 Distinctive Features and Phonology Acquisition

There is a great deal of evidence that many phonological contrasts are perceptually available from a very early stage (Eimas et al., 1971; Moffitt, 1971; Trehub, 1973; Jusczyk and Derrah, 1987; Eimas et al., 1987). However, studies of infant phone discrimination typically use carefully-enunciated laboratory stimuli, which have been shown to be substantially easier to discriminate than phones in naturalistic utterances (Feldman et al., 2013a; Antetomaso et al., 2017). It is thus likely that inferring phone categories from acoustic evidence is a persistently challenging task, and studies have found language-specific tuning of the speech perception system from fetal stages (Moon et al., 2013) through the first year (Kuhl et al., 1992; Werker and Tees, 1984) and even all the way into the preteen years (Hazan and Barrett, 2000).

Experiments show that these contrasts are expressed, not simply as oppositions between particular categories, but as a featural system, even in early infancy. Evidence of featural effects has been found in the phone discrimination patterns of both adults (Chládková et al., 2015) and infants (Kuhl, 1980; Hillenbrand, 1985; White and Morgan, 2008). Studies have also shown that infants generalize new distinctions along featural dimensions (Maye et al., 2008b; Cristià et al., 2011). Given infants' early detection and use of some featural contrasts, we hypothesize that there is strong evidence in the acoustic signal for these distinctions, which may then bootstrap the acquisition of phonotactic and lexical patterns (Beckman and Edwards, 2000).

Experiments also suggest asymmetries in the perceptual availability of features. For example, a consonant-vowel distinction appears to be an important early foothold in phonology acquisition: vowel/consonant discrimination emerges early in infant speech processing (Dehaene-Lambertz and Dehaene, 1994), language-specificity in perception follows different timecourses for consonants (Werker and Tees, 1984) and vowels (Kuhl et al., 1992), and vowels and consonants play distinct roles in lexical access vs. rule discovery in children (Nazzi, 2005; Pons and Toro, 2010; Hochmann et al., 2011). Young infants have also been shown to be sensitive to voicing contrasts (Lasky et al., 1975; Aslin et al., 1981; Maye et al., 2008b). Features that distinguish consonant-like from vowel-like segments or voiced from unvoiced segments may thus be highly available to young learners. Infants struggle by comparison with other kinds of phone discrimination tasks, including certain stop-fricative contrasts (Polka et al., 2001) and certain place distinctions within nasal (Narayan et al., 2010) and sibilant (Nittrouer, 2001; Cristià et al., 2011) segments. Even adults struggle with fricative place discrimination from strictly acoustic cues (McGuire and Babel, 2012). Similar asymmetries emerge from our unsupervised learner, as shown in Section 4.2.

Our computational acquisition model complements this experimental research in several ways. First, its internal representations, unlike those of human infants, are open to detailed analysis, even when exposed to naturalistic language stimuli. Second, we can perform cross-linguistic comparisons using readily available corpora without requiring access to a pool of human subjects in each language community. Third, our model provides global and graded quantification of the perceptual availability of distinctive features in natural speech, permitting us to explore relationships between features in a way that is difficult to do through experiments on infants, which are generally constrained to same-different contrasts over a small set of manipulations.

2.3 Cognition and the BSN Autoencoder

The reconstruction objective used here is not merely a convenient supervision signal. There is reason to believe that people actively model their perceptual worlds (Mamassian et al., 2002; Feldman, 2012; Singer et al., 2018; Yan et al., 2018), and autoassociative structures have been found in several brain areas (Treves and Rolls, 1991; Rolls and Treves, 1998). There is also evidence that phonetic comprehension and production can be acquired symbiotically through a sensorimotor loop relating acoustic perception and articulator movements (Houde and Jordan, 1998; Fadiga et al., 2002; Watkins et al., 2003; Wilson et al., 2004; Pulvermüller et al., 2006; Kröger et al., 2009; Bolhuis et al., 2010; Kröger and Cao, 2015; Bekolay, 2016). Finally, evidence suggests that working memory limitations impose compression pressures on the perceptual system that favor sparse representations of dense acoustic percepts (Baddeley and Hitch, 1974) and may guide infant language acquisition (Baddeley et al., 1998; Elsner and Shain, 2017). It is thus reasonable to suppose that perceptual reconstruction — such as that implemented by an autoencoder architecture — is immediately available as a learning signal to infants who still lack reliable guidance from phonotactics or the lexicon.

Our use of BSNs follows the spirit of the earliest work on artificial neural networks (Rosenblatt, 1958). Rosenblatt's perceptron was designed to study learning and decision-making in the brain and therefore used binary neurons to model the discrete firing behavior of their biological counterparts. This tradition has been replaced in deep learning research with differentiable activation functions that support supervised learning through backpropagation of error but are less biologically plausible. Our work takes advantage of the development of effective estimators for the gradients of discrete neurons (Williams, 1992; Hinton, 2012; Bengio et al., 2013; Courbariaux et al., 2016; Chung et al., 2017) to wed these two traditions, exploiting BSNs to encode the learner's latent representation of auditory percepts and deep networks to map between percepts and their latent representations. In addition to the greater similarity of BSNs to biological neurons, the use of discrete featural representations is motivated by experimental evidence that human phone perception (including that of infants) is both featural (White and Morgan, 2008; Chládková et al., 2015) and categorical (Liberman et al., 1961; Eimas et al., 1987; Harnad, 2003; Feldman et al., 2009b).

Experiments reported here use an 8-bit binary segment encoding. Eight bits is the the lower bound on binary encodings that are sufficiently expressive to capture all segmental contrasts in any known language (Mielke, 2009). Although theory-driven taxonomies generally contain more than eight distinctive features, these taxonomies are known to be highly redundant (Cherry et al., 1953). For example, the phonological featurization of the Xitsonga segments analyzed in our experiments contains 26 theory-driven features (Hayes, 2011; Hall et al., 2016), yielding up to $2^{26} = 67108864$ distinct segment categories, far more than the number of known segment types in Xitsonga or even the number of training instances in our data. By entailment, any representation that can identify all segment types in a language can also identify all featural contrasts that discriminate those types, regardless of how the feature space is factored. For this reason, we consider a phonological feature to be represented if it can be detected by an arbitrary function of the latent bits (Section 4.2), without assuming that the true and discovered feature spaces will factor identically.

2.4 Supervised Acoustic Feature Learning

Our study shares an interest in phonological features with previous work in automatic speech recognition attempting to discover mappings between acoustics and hand-labeled featural representations (Liu, 1996; Bitar and Espy-Wilson, 1996; Frankel and King, 2001; Kirchhoff et al., 2002; Livescu and Glass, 2004; Mitra et al., 2011, *inter alia*). While these results provide evidence that such a mapping is indeed learnable in an oracle setting, they rely on a supervision signal (direct annotation of the target representations) to which children do not have access. Our unsupervised approach measures perceptual availability of features in a more realistic learning scenario.

3 Experimental Setup

3.1 Model

The simulated learner used in this study is a deep neural autoencoder with an 8-bit layer of BSNs as its principle information bottleneck, depicted in Figure 1. The model processes a given phone segment by encoding the segment's acoustic informa-

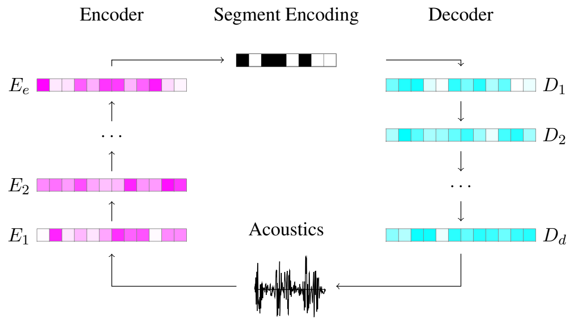


Figure 1: The binary stochastic neural autoencoder architecture with encoder layers $E_{1,\ldots,e}$ and decoder layers $D_{1,\ldots,d}$. For expository purposes, acoustics are represented as pressure waves. In reality, the system uses frames of Mel frequency cepstral coefficients.

tion into a bit pattern and then reconstructing the acoustic information from the encoded bit pattern. It is thus incentivized to use the latent bits in a systematic featural manner, encoding similar segments in similar ways.

The encoder and decoder are both deep feedfoward residual networks (He et al., 2016).[1] To enable feedforward autoencoding of sequential data, phone segments are clipped at 50 timesteps (500ms), providing complete coverage of over 99% of the phone segments in each corpus. Given F-dimensional input acoustic frames and a maximum input length of M timesteps, the weight matrix of each encoder layer is $\in \mathbb{R}^{FM \times FM}$ except the final layer ($\in \mathbb{R}^{FM \times 8}$). Given R-dimensional reconstructed acoustic frames and a maximum output length of N timesteps, the weight matrix of each decoder layer is $\in \mathbb{R}^{RN \times RN}$ except the first layer ($\in \mathbb{R}^{8 \times RN}$). Both the encoder and decoder contain initial and final dense transformation layers, with three residual layers in between. Each residual layer contains two dense layers. All internal layers use tanh activations and are batch-normalized with a decay rate of 0.9 (Ioffe and Szegedy, 2015).

Given that the capacity for *speaker adaptation* — short-term accommodation of idiosyncrasies in individuals' productions — has been shown for both adults (Clarke and Garrett, 2004; Maye et al., 2008a) and children (Kuhl, 1979; van Heugten and Johnson, 2014), we equip the models with a 16-dimensional speaker embedding, which is concatenated both to the acoustic input frames and to the latent bit vector.

Each BSN of the latent encoding is associated with a firing probability $\in [0, 1]$ parameterized by the encoder network. The neural activation can be discretized either deterministically or by sampling. The use of BSNs to encode segments is a problem for gradient-based optimization since it introduces a non-differentiable discrete decision into the network's latent structure. We overcome this problem by approximating missing gradients using the straight-through estimator (Hinton, 2012; Bengio et al., 2013; Courbariaux et al., 2016) with slope-annealing (Chung et al., 2017). Slope annealing multiplies the pre-activations a by a monotonically increasing function of the training iteration t, incrementally decreasing the bias of the straight-through estimator. We use the following annealing function:

$$a \leftarrow a(1 + 0.1t)$$

We discretize the latent dimensions using Bernoulli sampling during training and thresholding at 0.5 during evaluation.

The models are implemented in Tensorflow (Abadi et al., 2015) and optimized using Adam (Kingma and Ba, 2014) for 150 training epochs with a constant learning rate of 0.001. The source code is available at `https://github.com/coryshain/dnnseg`.

3.2 Data

We apply our model to the Xitsonga and English speech data from the Zerospeech 2015 shared task. The Xitsonga data are drawn from the NCHLT corpus (De Vries et al., 2014) and contain 2h29m07s of read speech from 24 speakers. The English data are drawn from the Buckeye Corpus (Pitt et al., 2005) and contain 4h59m05s of conversational speech from 12 speakers. While neither of these corpora represent child-directed speech, they both consist of fluently produced word tokens in context, rather than isolated productions as in many previous laboratory studies with infants (Eimas et al., 1971; Werker and Tees, 1984; Kuhl et al., 1992, *inter alia*). We pre-segment the audio files using time-aligned phone transcriptions pro-

---

[1] Feedforward networks are used both for computational reasons and because they dramatically outperformed recurrent networks in initial experiments, especially when RNN's were used for decoding. We hypothesize that this is due to the lack of direct access to the encoder timesteps, such as that permitted by sequence to sequence models with attention (Bahdanau et al., 2015). Attention is not viable for our goals because it defeats the purposes of an autoencoder by allowing the decoder to bypass the encoder's latent representation.

	Xitsonga			English		
Model	H	C	V	H	C	V
Baseline	0.023	0.013	0.016	0.006	0.004	0.005
Sigmoid	0.281	0.191	0.227	0.246	0.166	0.198
Sigmoid+Speaker	0.302	0.185	0.230	0.205	0.180	0.192
BSN	0.360	0.206	0.262	0.240	0.161	0.193
Our model (BSN+Speaker)	**0.462**	**0.268**	**0.339**	**0.270**	**0.180**	**0.216**

Table 1: Phone clustering scores. Homogeneity (H), completeness (C) and V-measure (V) across the Zerospeech 2015 Xitsonga and English challenge datasets.

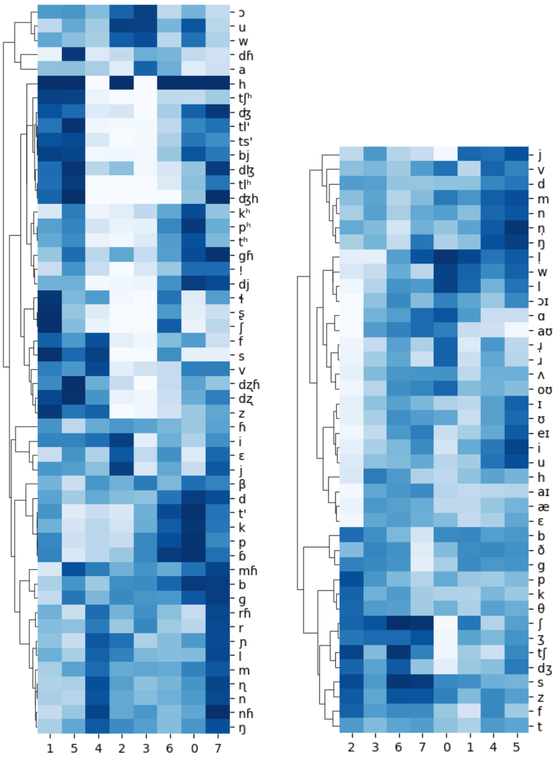
vided in the challenge repository. The gold segment labels are used in clustering evaluation metrics, but the unsupervised learner never has access to them. Data selection criteria and annotation procedures are are described in more detail in Versteegh et al. (2015).

Prior to fitting, we apply a standard spectral preprocessing pipeline from automatic speech recognition: raw acoustic signals are converted into 13-dimensional vectors of Mel frequency cepstral coefficients (MFCCs) (Mermelstein, 1976) with first and second order deltas, yielding 39-dimensional frames sequenced in time. Each frame covers 25ms of speech, and frames are spaced 10ms apart. The deltas are used by the encoder but stripped from the reconstruction targets. Following preceding work showing improved unsupervised clustering when segments are given fixed-dimensional acoustic representations, thus abstracting away from the variable temporal dilation in natural speech (Kamper et al., 2017a,b), we resample all reconstruction targets to a length of 25 frames.

This pipeline instantiates some standard assumptions about the perceptual representations underlying human speech processing. Alternative representations — for instance, articulatory representations (Liu, 1996; Frankel and King, 2001; Kirchhoff et al., 2002; Livescu and Glass, 2004) or other spectral transforms (Zwicker, 1961; Makhoul, 1975; Hermansky, 1990; Hermansky et al., 1991; Coifman and Wickerhauser, 1992; Shao et al., 2009) — have been proposed as alternatives to MFCCs. Our results concerning perceptual availability are of course tied to our input representation, since phenomena that are poorly distinguished by MFCCs have less effect on our autoencoder loss function. Nonetheless, MFCCs are known to produce high-quality supervised speech recognizers (Zheng et al., 2001; Hinton et al., 2012), and we therefore leave optimization of the representation of speech features to future work.

4 Results and Discussion

4.1 Phonemic Categories Partially Emerge from Modeling Auditory Percepts



(a) Xitsonga (b) English

Figure 2: Mean activation pattern by gold segment label from the BSN model with speaker embeddings, with darker color indexing higher average activation.

The first research question posed in the introduction was to what extent theory-driven phoneme categories emerge from a drive to model auditory percepts. We explore this question by evaluating the degree of correspondence between the autoencoder hidden states and the gold phone labels. Table 1 reports learning outcomes using the information theoretic measures *homogeneity* (H), *completeness* (C), and *V-measure* (V) for unsupervised cluster evaluation (Rosenberg and Hirschberg, 2007). All three metrics range over the interval $[0, 1]$, with 1 indexing perfect performance. As shown in the table, our model yields dramatically better clustering performance than a random baseline that uniformly draws cluster IDs from a pool of 256 categories: we obtain 2118% and 4500% relative V-measure improvements in Xitsonga and English, respectively. At the same time, clustering performance is far from perfect. This result indicates that perceptual modeling — an immediately-available learning signal in infant language acquisition — both (1) drives the learner a long way toward phoneme acquisition, and (2) is insufficient to fully identify phone categories in our learners. One likely explanation for the latter is evidence from cognitive science that phonotactic and lexical information (to which our learners do not have access) supplement perception as the acquisition process unfolds (Feldman et al., 2013a; Pater and Moreton, 2014).

The middle rows of Table 1 show ablation results from using non-discrete sigmoid neurons rather than BSNs in the encoding layer (*Sigmoid* vs. *BSN*)[2] and/or removing the speaker adaptation feature (i.e. removing speaker embeddings). As shown, the classification performance of our model benefits substantially from the use of BSN encodings with speaker adaptation, especially on Xitsonga. Note that the reconstruction losses of the sigmoid encoders are better than those of the BSN encoders despite their degraded classification performance. This is to be expected: sigmoid neurons have greater representational capacity than binary neurons, since they can encode information through continuous gradations. They are therefore more capable of memorizing idiosyncratic properties of the input and are less incentivized to discover generalizable latent classes. The ablation results thus suggest that speaker adaptation and categorical perception support the discovery of linguistically relevant abstractions.

4.2 Distinctive Features Differ in Perceptual Availability

The second research question posed in the introduction was to what extent distinctive features differ in perceptual availability. We explore this question in two ways.

First, we qualitatively assess the linguistic plausibility of the natural clustering in the latent bits. Figure 2 visualizes this clustering based on correlations between the average of the bit patterns across all instances of each gold phone type for both datasets. If the unsupervised classifier ignored phonological structure altogether, the plots would be roughly uniform in color, and if the unsupervised classifier perfectly identified phonemes, the plots would consist entirely of fully light or fully dark cells, with unique bit patterns associated with each phone type. As shown, the reality falls in between: while the visualized classifications are far from perfect, they nonetheless contain a great deal of structure and suggest the presence of rough natural classes in both languages, especially of affricates, nasals, sibilants, and approximants. Our learners also replicate infants' difficulty in discriminating some nasal and fricative place features (Polka et al., 2001; Nittrouer, 2001; Narayan et al., 2010), assigning highly similar representations to many subtypes of nasals and fricatives across places of articulation (see e.g. similar mean bit patterns of /n/ vs. /ŋ/ and /s/ vs. /ʃ/ in both languages).

Second, we quantitatively evaluate the degree to which theory-driven features like [±voice] are recoverable from the network's latent representations. To do so, we map gold phone labels into binary distinctive feature clusters from Hayes (2011) using Phonological CorpusTools (Hall et al., 2016). One possible form of analysis would be to search for individual correspondences between distinctive features and the model's latent dimensions. However, this is likely to underestimate the degree of feature learning because the deep decoder can learn arbitrary logics on the latent bit patterns, a necessary property for fitting complex non-linear mappings from latent features to acoustics. We instead evaluate distinctive feature discovery by fitting random forest classifiers that predict theory-driven features using the latent bit patterns as inputs. We can then use classifier performance to assess the degree to which a given distinctive feature can be recovered by a logical statement on the network's latent bits. The classifiers were fitted using 5-fold cross-validation in Scikit-learn (Pedregosa et al., 2011) with 100 estimators, balanced class weighting, and an entropy-based split criterion.

Results are given in Tables 2 and 3. As shown, (1) there are large differences in perceptual availability between features, and (2) relative avail-

---

[2] To obtain class labels from the sigmoid encoder, we rounded the activations. Rounding was only used for evaluation and had no impact on the fitting procedure.

Feature	P	R	F
voice	0.9767	0.9033	0.9386
sonorant	0.9249	0.9085	0.9166
continuant	0.9492	0.7936	0.8645
consonantal	0.8314	0.8915	0.8604
approximant	0.8998	0.8192	0.8576
syllabic	0.8278	0.8523	0.8398
dorsal	0.8935	0.7703	0.8273
strident	0.6991	0.9594	0.8089
low	0.7175	0.8978	0.7976
front	0.6590	0.8101	0.7268
high	0.5875	0.7882	0.6732
back	0.5352	0.8527	0.6577
round	0.5332	0.8551	0.6568
labial	0.5669	0.7725	0.6539
coronal	0.5382	0.8301	0.6530
tense	0.5208	0.8115	0.6344
delayed release	0.5468	0.7226	0.6225
anterior	0.4078	0.8355	0.5481
nasal	0.3635	0.8796	0.5144
distributed	0.2459	0.8537	0.3819
constricted glottis	0.1762	0.9007	0.2948
lateral	0.1536	0.8062	0.2581
labiodental	0.0934	0.7980	0.1672
trill	0.0809	0.7401	0.1458
spread glottis	0.0671	0.5856	0.1204
implosive	0.0041	0.4041	0.0081

Table 2: Perceptual availability by feature in Xitsonga

Feature	P	R	F
voice	0.9244	0.8567	0.8893
sonorant	0.8544	0.8862	0.8700
approximant	0.8005	0.8370	0.8183
continuant	0.8577	0.7669	0.8098
consonantal	0.8249	0.7357	0.7777
syllabic	0.6624	0.8426	0.7417
dorsal	0.7046	0.7114	0.7080
strident	0.5505	0.9027	0.6839
coronal	0.5758	0.7066	0.6345
anterior	0.5251	0.7280	0.6101
delayed release	0.4413	0.7374	0.5521
front	0.4322	0.7407	0.5459
high	0.3841	0.6931	0.4943
tense	0.3275	0.7101	0.4483
back	0.3128	0.7504	0.4416
nasal	0.2796	0.7544	0.4080
labial	0.2541	0.7077	0.3739
low	0.2410	0.7787	0.3680
distributed	0.2203	0.6881	0.3337
diphthong	0.2039	0.8051	0.3254
round	0.1665	0.7012	0.2692
lateral	0.1484	0.8333	0.2519
labiodental	0.0787	0.6756	0.1410
spread glottis	0.0377	0.6683	0.0714

Table 3: Perceptual availability by feature in English

ability of features is remarkably consistent between these unrelated languages, suggesting that the models are tapping into generalized perceptual patterns. The best-learned feature in both languages is [±voice], which is consistent with early evidence of voicing sensitivity in infants (see Section 2.2). Below this, the features [±sonorant], [±continuant], [±consonantal], [±approximant], and [±syllabic] are faithfully recovered in both languages. All of these features distinguish prototypical consonants from prototypical vowels but differ in their treatment of edge cases like nasals, liquids, and glides. Thus, similarly to the infant subjects discussed in Section 2.2, the model finds the consonant-vowel contrast to be highly available. Like human infants, our computational learner finds certain consonantal place and manner features relatively more difficult, although the features [±dorsal], [±coronal], [±strident] and [±delayed release] are also fairly well recovered in both languages. By contrast, both models poorly capture features like [±lateral], [±labiodental], [±distributed], [±nasal], [±constricted glottis], [±spread glottis], and [±implosive],[3] suggesting that these features are more difficult to discover bottom-up and may therefore be more dependent on phonotactic and lexical constraints for acquisition.[4] This finding aligns with the acquisition literature in suggesting that there may be substantial differences in perceptual availability between different place and manner features (see Section 2.2).

In addition to these cross-linguistic similarities, the models also reveal important differences between Xitsonga and English. For example, the two languages differ in the relative availability of features that distinguish vowels vs. features that distinguish consonants. In English, vowel features like [±front], [±high], and [±back] are substantially less well learned than consonant features like [±coronal], [±anterior], and [±delayed release], while the opposite holds in Xitsonga. We hypothesize that this is due to the fact that there are more vowels and fewer consonants in English than in Xitsonga: having fewer distinctions might reduce the degree of "crowding" in the articulatory space, increasing perceptual contrast between phone types (Liljencrants and Lindblom, 1972).

---

[3] Delayed release: affricates, constricted glottis: ejectives; spread glottis: glottal frication (e.g. aspirated stops).

[4] Note that we are not suggesting that e.g. [±spread glottis] cannot be detected in speech. Our claim is rather that acoustic cues to [±spread glottis] are less pronounced and/or less reliable than cues to e.g. [±voice] and therefore perhaps more difficult to exploit in early infancy, since our autoencoder model does not find them particularly useful for perceptual reconstruction.

Finally, note that the cluster maps in Figure 2 and the feature recovery data in Tables 2 and 3 provide complementary perspectives on the learned representations. For example, it may at first seem surprising that the feature [±nasal] is recovered relatively poorly in both languages, given that nasals are well clustered in Figure 2. This discrepancy indicates that nasal segments are represented similarly to each other but also similarly enough to other segments that they are not reliably differentiated as a class. Conversely, the voicing feature is well recovered in both languages despite the lack of a visible cluster of voiced segments. This indicates that voicing is reliably encoded in the latent bits, even if the representation as a whole is dominated by other kinds of information.

5 Conclusion

In this paper, we used binary stochastic neural autoencoders to explore the perceptual availability of (1) theory-driven phonemic categories and (2) theory-driven phonological features, based only on the acoustic properties of segments. We found that phonemic categories exert substantial influence on a learner driven to model its auditory percepts, but that additional information — especially phonotactic and lexical (Feldman et al., 2013a) — is likely necessary for full adult-like phone discrimination. We also found asymmetries in the perceptual availability of phonological features like [±voice] and [±nasal] and showed that these asymmetries reflect attested patterns of infant phone discrimination. Our model both replicates broad trends in the child acquisition literature (successful consonant-vowel and voicing discrimination, relatively less successful discrimination of various place and manner features) and sheds new light on potential relationships between auditory perception and language acquisition: the overall cline of perceptual availability revealed by the model in Tables 2 and 3 suggests a range of testable hypotheses about the role of perception in infant speech processing.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by National Science Foundation grant #1422987 to ME. All views expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

Wiehan Agenbag and Thomas Niesler. 2015. Automatic segmentation and clustering of speech using sparse coding and metaheuristic search. In *Sixteenth Annual Conference of the International Speech Communication Association*.

T K Ansari, Rajath Kumar, Sonali Singh, and Sriram Ganapathy. 2017a. Deep learning methods for unsupervised acoustic modelingLeap submission to ZeroSpeech challenge 2017. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 754–761. IEEE.

T K Ansari, Rajath Kumar, Sonali Singh, Sriram Ganapathy, and Susheela Devi. 2017b. Unsupervised HMM posteriograms for language independent acoustic modeling in zero resource conditions. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 762–768. IEEE.

Stephanie Antetomaso, Kouki Miyazawa, Naomi Feldman, Micha Elsner, Kasia Hitczenko, and Reiko Mazuka. 2017. Modeling phonetic category learning from natural acoustic data. In *Proceedings of the annual Boston University Conference on Language Development*.

Richard N Aslin, David B Pisoni, Beth L Hennessy, and Alan J Perey. 1981. Discrimination of voice onset time by human infants: New findings and implications for the effects of early experience. *Child development*, 52(4):1135.

Alan Baddeley, Susan Gathercole, and Costanza Papagno. 1998. The Phonological Loop as a Language Learning Device. *Psychological Review*, 105(1):158–173.

Alan D Baddeley and Graham Hitch. 1974. *Working Memory*. University of Stirling, Stirling, Scotland.

Leonardo Badino, Alessio Mereta, and Lorenzo Rosasco. 2015. Discovering discrete subword units with binarized autoencoders and hidden-markov-model encoders. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations 2015*.

Pallavi Baljekar, Sunayana Sitaram, Prasanna Kumar Muthukumar, and Alan W Black. 2015. Using articulatory features and inferred phonological segments in zero resource speech processing. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Mary E Beckman and Jan Edwards. 2000. The ontogeny of phonological categories and the primacy of lexical learning in linguistic development. *Child development*, 71(1):240–249.

Trevor Bekolay. 2016. *Biologically inspired methods in speech recognition and synthesis: Closing the loop*. Ph.D. thesis, University of Waterloo.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Nabil N Bitar and Carol Y Espy-Wilson. 1996. Knowledge-based parameters for HMM speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 29–32. IEEE.

Johan J Bolhuis, Kazuo Okanoya, and Constance Scharff. 2010. Twitter evolution: converging mechanisms in birdsong and human speech. *Nature Reviews Neuroscience*, 11(11):747.

Hongjie Chen, Cheung-Chi Leung, Lei Xie, Bin Ma, and Haizhou Li. 2015. Parallel inference of Dirichlet process Gaussian mixture models for unsupervised acoustic modeling: A feasibility study. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Hongjie Chen, Cheung-Chi Leung, Lei Xie, Bin Ma, and Haizhou Li. 2017. Multilingual bottle-neck feature learning from untranscribed speech. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 727–733. IEEE.

E Colin Cherry, Morris Halle, and Roman Jakobson. 1953. Toward the logical description of languages in their phonemic aspect. *Language*, pages 34–46.

Katerina Chládková, Paul Boersma, Titia Benders, and others. 2015. The perceptual basis of the feature vowel height. In *ICPhS*.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper \& Row.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. In *International Conference on Learning Representations 2017*.

Constance M Clarke and Merrill F Garrett. 2004. Rapid adaptation to foreign-accented English. *The Journal of the Acoustical Society of America*, 116(6):3647–3658.

George N Clements. 1985. The geometry of phonological features. *Phonology*, 2(1):225–252.

Ronald R Coifman and M Victor Wickerhauser. 1992. Entropy-based algorithms for best basis selection. *IEEE Transactions on information theory*, 38(2):713–718.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Alejandrina Cristià, Grant L McGuire, Amanda Seidl, and Alexander L Francis. 2011. Effects of the distribution of acoustic cues on infants' perception of sibilants. *Journal of phonetics*, 39(3):388–402.

Nic J De Vries, Marelie H Davel, Jaco Badenhorst, Willem D Basson, Febe De Wet, Etienne Barnard, and Alta De Waal. 2014. A smartphone-based ASR data collection tool for under-resourced languages. *Speech communication*, 56:119–131.

Ghislaine Dehaene-Lambertz and Stanislas Dehaene. 1994. Speed and cerebral correlates of syllable discrimination in infants. *Nature*, 370(6487):292.

Gabriel Doyle, Klinton Bicknell, and Roger Levy. 2014. Nonparametric learning of phonological constraints in optimality theory. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1094–1103.

Gabriel Doyle and Roger Levy. 2016. Data-driven learning of symbolic constraints for a log-linear model in a phonological setting. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2217–2226.

Ewan Dunbar, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux. 2017. The zero resource speech challenge 2017. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 323–330. IEEE.

Sorin Dusan and Lawrence Rabiner. 2006. On the relation between maximum spectral transition positions and phone boundaries. In *Ninth International Conference on Spoken Language Processing*.

Peter D. Eimas, Joanne L. Miller, and Peter W. Jusczyk. 1987. On infant speech perception and the acquisition of language. In Stevan Harnad, editor, *Categorical perception: The groundwork of cognition*, pages 161–195. Cambridge University Press, New York.

Peter D Eimas, Einar R Siqueland, Peter Jusczyk, and James Vigorito. 1971. Speech perception in infants. *Science*, 171(3968):303–306.

Micha Elsner and Cory Shain. 2017. Speech segmentation with a neural encoder model of working memory. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1080.

Luciano Fadiga, Laila Craighero, Giovanni Buccino, and Giacomo Rizzolatti. 2002. Speech listening specifically modulates the excitability of tongue muscles: a TMS study. *European journal of Neuroscience*, 15(2):399–402.

Jacob Feldman. 2012. Symbolic representation of probabilistic worlds. *Cognition*, 123(1):61–83.

Naomi Feldman, Thomas Griffiths, and James Morgan. 2009a. Learning phonetic categories by learning a lexicon. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 31.

Naomi H Feldman, Thomas L Griffiths, Sharon Goldwater, and James L Morgan. 2013a. A role for the developing lexicon in phonetic category acquisition. *Psychological review*, 120(4):751.

Naomi H Feldman, Thomas L Griffiths, and James L Morgan. 2009b. The influence of categories on perception: Explaining the perceptual magnet effect as optimal statistical inference. *Psychological review*, 116(4):752.

Naomi H Feldman, Emily B Myers, Katherine S White, Thomas L Griffiths, and James L Morgan. 2013b. Word-level information influences phonetic learning in adults and infants. *Cognition*, 127(3):427–438.

Stella Frank, Naomi H Feldman, and Sharon Goldwater. 2014. Weak semantic context helps phonetic learning in a model of infant language acquisition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Joe Frankel and Simon King. 2001. ASR-articulatory speech recognition. In *Seventh European Conference on Speech Communication and Technology*.

Kathleen Currie Hall, Blake Allen, Michael Fry, Scott Mackie, and Michael McAuliffe. 2016. Phonological CorpusTools: A free, open-source tool for phonological analysis. In *14th Conference for Laboratory Phonology*, volume 543.

Stevan Harnad. 2003. Categorical Perception. In *Encyclopedia of Cognitive Science*, volume 67. MacMillan: Nature Publishing Group.

Bruce Hayes. 2011. *Introductory phonology*, volume 32. John Wiley \& Sons, Hoboken.

Valerie Hazan and Sarah Barrett. 2000. The development of phonemic categorization in children aged 6–12. *Journal of phonetics*, 28(4):377–396.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Michael Heck, Sakriani Sakti, and Satoshi Nakamura. 2016. Unsupervised linear discriminant analysis for supporting DPGMM clustering in the zero resource scenario. *Procedia Computer Science*, 81:73–79.

Michael Heck, Sakriani Sakti, and Satoshi Nakamura. 2017. Feature optimized dpgmm clustering for unsupervised subword modeling: A contribution to zerospeech 2017. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 740–746. IEEE.

Hynek Hermansky. 1990. Perceptual linear predictive (PLP) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hynek Hermansky, Nelson Morgan, Aruna Bayya, and Phil Kohn. 1991. Compensation for the effect of the communication channel in auditory-like analysis of speech (RASTA-PLP). In *Second European Conference on Speech Communication and Technology*.

Marieke van Heugten and Elizabeth K Johnson. 2014. Learning to contend with accents in infancy: Benefits of brief speaker exposure. *Journal of Experimental Psychology: General*, 143(1):340.

James Hillenbrand. 1985. Perception of feature similarities by infants. *Journal of Speech & Hearing Research*, 28(2):317–318.

Geoffrey Hinton. 2012. Neural Networks for Machine Learning. *Coursera, video lectures*.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, and others. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

Jean-Rémy Hochmann, Silvia Benavides-Varela, Marina Nespor, and Jacques Mehler. 2011. Consonants and vowels: different roles in early language acquisition. *Developmental science*, 14(6):1445–1458.

John F Houde and Michael I Jordan. 1998. Sensorimotor adaptation in speech production. *Science*, 279(5354):1213–1216.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456.

Roman Jakobson, C Gunnar Fant, and Morris Halle. 1951. *Preliminaries to speech analysis: The distinctive features and their correlates*. MIT press.

Aren Jansen and Kenneth Church. 2011. Towards unsupervised training of speaker independent acoustic models. In *Twelfth Annual Conference of the International Speech Communication Association*.

Peter W Jusczyk and Carolyn Derrah. 1987. Representation of speech sounds by young infants. *Developmental Psychology*, 23(5):648.

Herman Kamper, Micha Elsner, Aren Jansen, and Sharon Goldwater. 2015. Unsupervised neural network based feature extraction using weak top-down constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5818–5822. IEEE.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2017a. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Computer Speech & Language*, 46:154–174.

Herman Kamper, Karen Livescu, and Sharon Goldwater. 2017b. An embedded segmental k-means model for unsupervised segmentation and clustering of speech. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 719–726. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6.

Katrin Kirchhoff, Gernot A Fink, and Gerhard Sagerer. 2002. Combining acoustic and articulatory feature information for robust speech recognition. *Speech Communication*, 37(3-4):303–319.

Bernd J Kröger and Mengxue Cao. 2015. The emergence of phonetic–phonological features in a biologically inspired model of speech processing. *Journal of Phonetics*, 53:88–100.

Bernd J Kröger, Jim Kannampuzha, and Christiane Neuschaefer-Rube. 2009. Towards a neurocomputational model of speech production and perception. *Speech Communication*, 51(9):793–809.

Patricia K Kuhl. 1979. Speech perception in early infancy: Perceptual constancy for spectrally dissimilar vowel categories. *The Journal of the Acoustical Society of America*, 66(6):1668–1679.

Patricia K Kuhl. 1980. Perceptual constancy for speech-sound categories in early infancy. *Child phonology*, 2:41–66.

Patricia K Kuhl, Karen A Williams, Francisco Lacerda, Kenneth N Stevens, and Bjrn Lindblom. 1992. Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, 255(5044):606–608.

Robert E Lasky, Ann Syrdal-Lasky, and Robert E Klein. 1975. VOT discrimination by four to six and a half month old infants from Spanish environments. *Journal of Experimental Child Psychology*, 20(2):215–225.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Chia-ying Lee and James Glass. 2012. A Nonparametric {Bayesian} Approach to Acoustic Model Discovery. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 40–49.

Alvin Liberman, Katherine Safford Harris, Peter Eimas, Leigh Lisker, and Jarvis Bastian. 1961. An effect of learning on speech perception: The discrimination of durations of silence with and without phonemic significance. *Language and Speech*, 4(4):175–195.

Johan Liljencrants and Bjorn Lindblom. 1972. Numerical Simulation of Vowel Quality Systems: The Role of Perceptual Contrast. *Language*, 48(4):839–862.

Sharlene A Liu. 1996. Landmark detection for distinctive feature-based speech recognition. *The Journal of the Acoustical Society of America*, 100(5):3417–3430.

Karen Livescu and James Glass. 2004. Feature-based pronunciation modeling for speech recognition. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 81–84. Association for Computational Linguistics.

Vince Lyzinski, Gregory Sell, and Aren Jansen. 2015. An evaluation of graph clustering methods for unsupervised term discovery. In *Sixteenth Annual Conference of the International Speech Communication Association*.

John Makhoul. 1975. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Pascal Mamassian, Michael Landy, and Laurence T Maloney. 2002. Bayesian modelling of visual perception. In Rajesh P N Rao, Bruno A Olshausen, Michael S Lewicki, Michael I Jordan, and Thomas G Dietterich, editors, *Probabilistic models of the brain: Perception and neural function*, pages 13–36. The MIT Press, Cambridge, MA.

Andrew Martin, Sharon Peperkamp, and Emmanuel Dupoux. 2013. Learning phonemes with a protolexicon. *Cognitive Science*, 37(1):103–124.

Jessica Maye, Richard N Aslin, and Michael K Tanenhaus. 2008a. The weckud wetch of the wast: Lexical adaptation to a novel accent. *Cognitive Science*, 32(3):543–562.

Jessica Maye, Daniel J Weiss, and Richard N Aslin. 2008b. Statistical phonetic learning in infants: Facilitation and feature generalization. *Developmental science*, 11(1):122–134.

Grant McGuire and Molly Babel. 2012. A crossmodal account for synchronic and diachronic patterns of/f/and/$\theta$$/in English. *Laboratory Phonology*, 3(2):251–272.

Paul Mermelstein. 1976. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388.

Jeff Mielke. 2009. Segment inventories. *Language and linguistics compass*, 3(2):700–718.

Vikramjit Mitra, Hosung Nam, and Carol Espy-Wilson. 2011. Robust speech recognition using articulatory gestures in a Dynamic Bayesian Network framework. *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*.

Alan R Moffitt. 1971. Consonant cue perception by twenty-to twenty-four-week-old infants. *Child development*, pages 717–731.

Christine Moon, Hugo Lagercrantz, and Patricia K Kuhl. 2013. Language experienced in utero affects vowel perception after birth: A two-country study. *Acta Paediatrica*, 102(2):156–160.

Elliott Moreton and Joe Pater. 2012a. Structure and substance in artificial-phonology learning, part I: Structure. *Language and linguistics compass*, 6(11):686–701.

Elliott Moreton and Joe Pater. 2012b. Structure and substance in artificial-phonology learning, part II: Substance. *Language and linguistics compass*, 6(11):702–718.

Chandan R Narayan, Janet F Werker, and Patrice Speeter Beddor. 2010. The interaction between acoustic salience and language experience in developmental speech perception: Evidence from nasal place discrimination. *Developmental Science*, 13(3):407–420.

Thierry Nazzi. 2005. Use of phonetic specificity during the acquisition of new words: Differences between consonants and vowels. *Cognition*, 98(1):13–30.

Susan Nittrouer. 2001. Challenging the notion of innate phonetic boundaries. *The Journal of the Acoustical Society of America*, 110(3):1598–1605.

Joe Pater and Elliott Moreton. 2014. Structurally biased phonology: complexity in learning and typology. *The EFL Journal*, 3(2).

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.

Sharon Peperkamp, Rozenn Le Calvez, Jean-Pierre Nadal, and Emmanuel Dupoux. 2006. The acquisition of allophonic rules: Statistical learning with linguistic constraints. *Cognition*, 101(3):B31–B41.

Mark A Pitt, Keith Johnson, Elizabeth Hume, Scott Kiesling, and William Raymond. 2005. The Buckeye corpus of conversational speech: labeling conventions and a test of transcriber reliability. *Speech Communication*, 45(1):89–95.

Linda Polka, Connie Colantonio, and Megha Sundara. 2001. A cross-language comparison of/d/–/{\dh}/perception: evidence for a new developmental pattern. *The Journal of the Acoustical Society of America*, 109(5):2190–2201.

Ferran Pons and Juan M Toro. 2010. Structural generalizations over consonants and vowels in 11-month-old infants. *Cognition*, 116(3):361–367.

Friedemann Pulvermüller, Martina Huss, Ferath Kherif, Fermin Moscoso del Prado Martin, Olaf Hauk, and Yury Shtyrov. 2006. Motor cortex maps articulatory features of speech sounds. *Proceedings of the National Academy of Sciences*, 103(20):7865–7870.

Yu Qiao, Naoya Shimomura, and Nobuaki Minematsu. 2008. Unsupervised optimal phoneme segmentation: Objectives, algorithm and comparisons. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3989–3992. IEEE.

Okko Räsänen, Gabriel Doyle, and Michael C Frank. 2015. Unsupervised word discovery from speech using automatic segmentation into syllable-like units. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Okko Johannes Räsänen, Gabriel Doyle, and Michael C Frank. 2018. Pre-linguistic segmentation of speech into syllable-like units. *Cognition*, 171:130–150.

Daniel Renshaw, Herman Kamper, Aren Jansen, and Sharon Goldwater. 2015. A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Edmund T Rolls and Alessandro Treves. 1998. *Neural networks and brain function*, volume 572. Oxford University Press, Oxford.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Yang Shao, Zhaozhang Jin, DeLiang Wang, and Soundararajan Srinivasan. 2009. An auditory-based feature for robust speech recognition. In *2009 IEEE*

International Conference on Acoustics, Speech and Signal Processing, pages 4625–4628. IEEE.

Hayato Shibata, Taku Kato, Takahiro Shinozaki, and Shinji Watanabet. 2017. Composite embedding systems for ZeroSpeech2017 Track1. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 747–753. IEEE.

Yosef Singer, Yayoi Teramoto, Ben D B Willmore, Jan W H Schnupp, Andrew J King, and Nicol S Harper. 2018. Sensory cortex is optimized for prediction of future input. *eLife*, 7:e31557.

Brij Mohan Lal Srivastava and Manish Shrivastava. 2016. Articulatory gesture rich representation learning of phonological units in low resource settings. In *International Conference on Statistical Language and Speech Processing*, pages 80–95. Springer.

Daniel Swingley. 2009. Contributions of infant word learning to language development. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1536):3617–3632.

Sandra E Trehub. 1973. Infants' sensitivity to vowel and tonal contrasts. *Developmental Psychology*, 9(1):91.

Alessandro Treves and Edmund T Rolls. 1991. What determines the capacity of autoassociative memories in the brain? *Network: Computation in Neural Systems*, 2(4):371–397.

Nikolaï Sergeyevich Trubetskoy. 1939. Grundzüge der phonologie. In *Travaux du Cercle Linguistique de Prague*, volume 7. Van den Hoeck & Ruprecht.

Gautam K Vallabha, James L McClelland, Ferran Pons, Janet F Werker, and Shigeaki Amano. 2007. Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences*, 104(33):13273–13278.

Balakrishnan Varadarajan, Sanjeev Khudanpur, and Emmanuel Dupoux. 2008. Unsupervised learning of acoustic sub-word units. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 165–168. Association for Computational Linguistics.

Maarten Versteegh, Roland Thiollière, Thomas Schatz, Xuan Nga Cao, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux. 2015. The zero resource speech challenge 2015. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Richard M Warren. 1970. Perceptual restoration of missing speech sounds. *Science*, 167(3917):392–393.

Kate E Watkins, Antonio P Strafella, and Tomáš Paus. 2003. Seeing and hearing speech excites the motor system involved in speech production. *Neuropsychologia*, 41(8):989–994.

Janet F Werker and Richard C Tees. 1984. Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. *Infant behavior and development*, 7(1):49–63.

Katherine S White and James L Morgan. 2008. Sub-segmental detail in early lexical representations. *Journal of Memory and Language*, 59(1):114–132.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Stephen M Wilson, Ayşe Pinar Saygin, Martin I Sereno, and Marco Iacoboni. 2004. Listening to speech activates motor areas involved in speech production. *Nature Neuroscience*, 7(7):701.

Shaorong Yan, Francis Mollica, and Michael K Tanenhaus. 2018. A context constructivist account of contextual diversity. In *Proceedings of the 40th Annual Cognitive Science Society Meeting*.

Yougen Yuan, Cheung-Chi Leung, Lei Xie, Hongjie Chen, Bin Ma, and Haizhou Li. 2017. Extracting bottleneck features and word-like pairs from untranscribed speech for feature representation. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 734–739. IEEE.

Neil Zeghidour, Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. 2016. A deep scattering spectrumdeep siamese network pipeline for unsupervised acoustic modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4965–4969. IEEE.

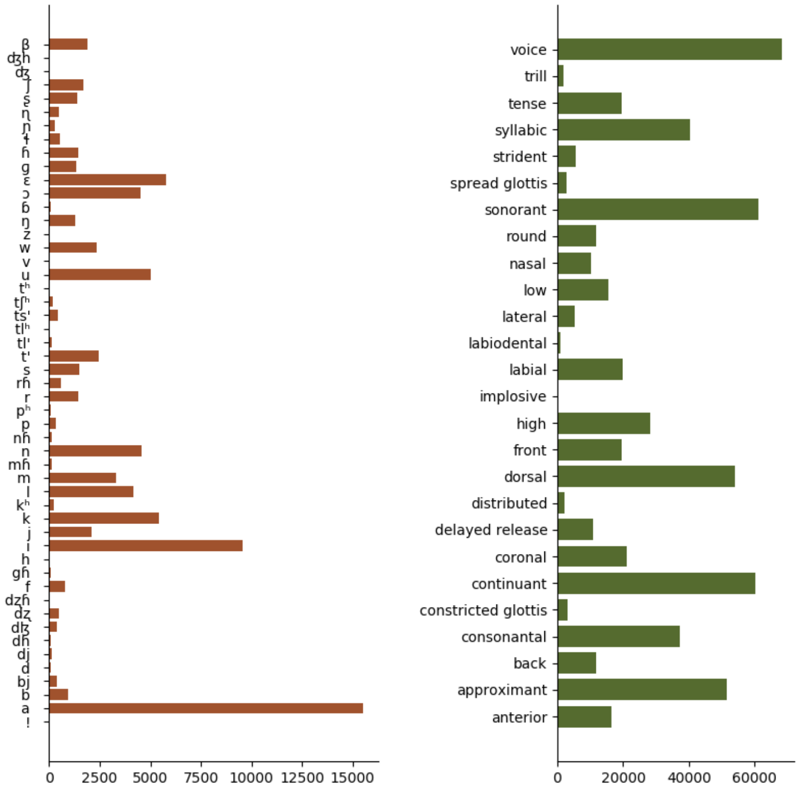
Fang Zheng, Guoliang Zhang, and Zhanjiang Song. 2001. Comparison of different implementations of MFCC. *Journal of Computer science and Technology*, 16(6):582–589.

Eberhard Zwicker. 1961. Subdivision of the audible frequency range into critical bands (Frequenzgruppen). *The Journal of the Acoustical Society of America*, 33(2):248.

A Phonological feature definitions

We adopt the phonological feature definitions presented in Hayes (2011). For full exposition of the features and their motivations, we refer readers to the source. However, for convenience, we provide the following brief (and in some cases oversimplified) definitions based on Hayes (2011):

- **syllabic:** Vowels are [+syllabic], others are [-syllabic]
- **consonantal:** Vowels and glides are [-consonantal], others are [+consonantal]
- **approximant:** Vowels, liquids, and glides are [+approximant], others are [-approximant]

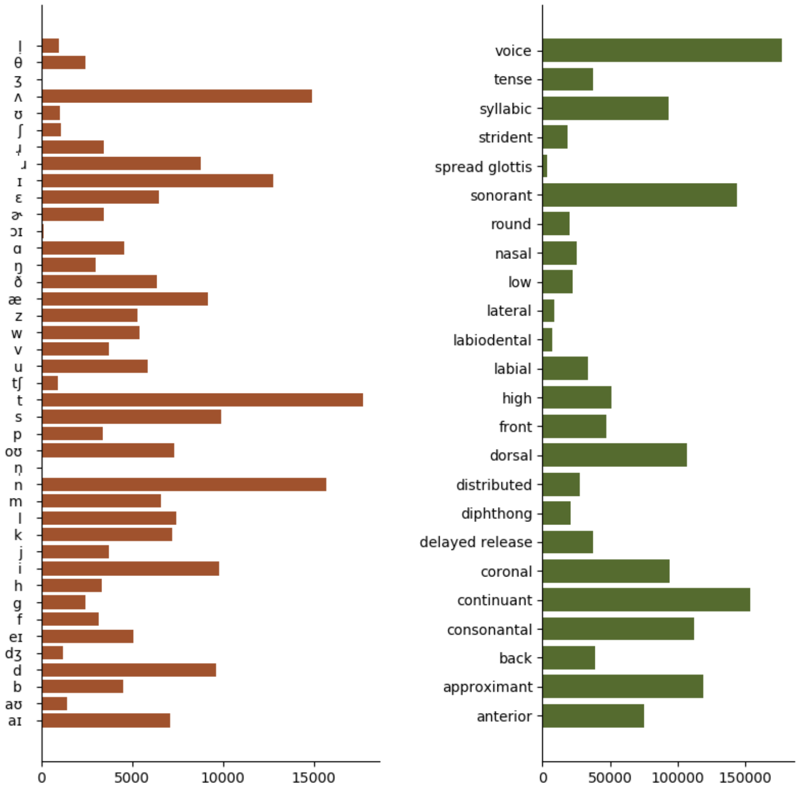


(a) Phonemes

(b) Features

Figure 3: Xitsonga phoneme and feature distributions.

- **sonorant:** Vowels, liquids, glides, and nasals are [+sonorant], others are [-sonorant]
- **continuant:** Stops and affricates are [-continuant], others are [+continuant]
- **delayed release:** Affricates and fricatives are [+delayed release], others are [-delayed release]
- **trill:** Trills are [+trill], others are [-trill]
- **front:** Front vowels and fronted velars are [+front], others are [-front]
- **back:** Back vowels and back velars are [+back], others are [-back]
- **high:** High vowels and velars are [+high], others are [-high]
- **low:** Low vowels and pharyngeals are [+low], others are [-low]
- **tense:** Tense vowels are [+tense], others are [-tense]
- **round:** Rounded vowels and rounded labial consonants are [+round], others are [-round]
- **nasal:** Nasal consonants and (contrastively) nasalized vowels are [+nasal], others are [-nasal]
- **labial:** Sounds articulated with the lips are [+labial], others are [-labial]
- **coronal:** Sounds articulated with the tongue blade/tip are [+coronal], others are [-coronal]
- **dorsal:** Sounds articulated with the tongue body are [+dorsal], others are [-dorsal]
- **anterior:** Coronals articulated at the alveolar ridge or forward are [+anterior], others are [-anterior]
- **distributed:** Coronals articulated with the tongue blade are [+distributed], others are [-distributed]
- **strident:** Sibilants (i.e. coronal fricatives and affricates) are [+strident], others are [-strident]
- **lateral:** Sounds with lateral oral closure (open at edges, like [l]) are [+lateral], others are [-lateral]
- **labiodental:** Sounds that are articulated by touching the lower lip to the upper teeth are [+labiodental], others are [-labiodental]



(a) Phonemes

(b) Features

Figure 4: English phoneme and feature distributions.

- **voice:** Voiced sounds are [+voice], others are [-voice]
- **spread glottis:** [h], [ɦ], and (contrastively) aspirated consonants are [+spread glottis], others are [-spread glottis]
- **constricted glottis:** Ejectives and glottal stops are [+constricted glottis], others are [-constricted glottis]
- **implosive:** Implosives are [+implosive], others are [-implosive]

B Xitsonga Phoneme Featurization

To the best of our knowledge, the gold Xitsonga phone transcriptions provided by the Zerospeech 2015 dataset use a non-standard pronunciation alphabet that is undocumented but isomorphic to the NCHLT transcription convention. In order to extract distinctive features for the Xitsonga phone labels, we hand-mapped the Zerospeech labels onto NCHLT labels by cross-referencing the Zerospeech phone sequences, the Zerospeech orthographic word sequences, and the NCHLT pronunciation dictionary, searching for systematic correspondences between Zerospeech and NCHLT transcription practices. Once the Zerospeech-to-NCHLT mapping was obtained, we used the International Phonetic Alphabet (IPA) phone labels provided by NCHLT to look up distinctive features in the Phonological CorpusTools (PCT) feature maps (Hall et al., 2016). Some IPA labels from NCHLT were not found in the PCT database, and for those we used the following featurization rules:

- **Consonants with palatal offglides:** We used the features associated with the non-offglide consonant and switched on the *approximant*, *dorsal*, *high*, *front*, and *tense* features.
- **Aspirated consonants:** We used the features associated with the non-aspirated consonant and switched on the *spread glottis* feature.
- **Ejective consonants:** We used the features associated with the non-ejective consonant and switched on the *constricted glottis* feature.
- **Voiceless alveolar lateral stops:** We used the features associated with voiceless alveolar lateral affricates and switched off the *de-*

layed release feature.

Our hand-made symbol correspondences and featurizations are distributed with this project's code repository.

C Phoneme and feature distributions

For reference, counts of phonemes and features by corpus are plotted in Figures 3 and 4. Note that the feature counts are generally larger because multiple features can be true of any one segment.

Giving Attention to the Unexpected: Using Prosody Innovations in Disfluency Detection

Vicky Zayats and Mari Ostendorf
Electrical Engineering Department
University of Washington
`[vzayats, ostendor]@uw.edu`

Abstract

Disfluencies in spontaneous speech are known to be associated with prosodic disruptions. However, most algorithms for disfluency detection use only word transcripts. Integrating prosodic cues has proved difficult because of the many sources of variability affecting the acoustic correlates. This paper introduces a new approach to extracting acoustic-prosodic cues using text-based distributional prediction of acoustic cues to derive vector z-score features (innovations). We explore both early and late fusion techniques for integrating text and prosody, showing gains over a high-accuracy text-only model.

1 Introduction

Speech disfluencies are frequent events in spontaneous speech. The rate of disfluencies varies with the speaker and context; one study observed disfluencies once in every 20 words, affecting up to one third of utterances (Shriberg, 1994). Disfluencies are important to account for, both because of the challenge that the disrupted grammatical flow poses for natural language processing of spoken transcripts and because of the information that they provide about the speaker.

Most work on disfluency detection builds on the framework that annotates a disfluency in terms of a reparandum followed by an interruption point (+), an optional interregnum ({ }), and then the repair, if any. A few simple examples are given below:

```
[ it's + {uh} it's] almost...
[ was it, + {I mean} , did you ] put...
[I just + I] enjoy working...
[By + ] it was attached to...
```

Based on the similarity/differences between the reparandum and the repair, disfluencies are often categorized into three types: repetition (the first example), rephrase (the next example), and restart (the last example).

The interruption point is associated with a disruption in the realization of a prosodic phrase, which could involve cutting words off or elongation associated with hesitation, followed by a prosodic reset at the start of the repair. There may also be emphasis in the repair to highlight the correction.

Researchers have been working on automatic disfluency detection for many years (Lickley, 1994; Shriberg et al., 1997; Charniak and Johnson, 2001; Johnson and Charniak, 2004; Lease et al., 2006; Qian and Liu, 2013; Zayats et al., 2016), motivated in part by early work on parsing speech that assumed reliable detection of the interruption point (Nakatani and Hirschberg, 1994; Shriberg and Stolcke, 1997; Liu et al., 2006). The first efforts to integrate prosody with word cues for disfluency detection (Baron et al., 2002; Snover et al., 2004) found gains from using prosody, but word cues played the primary role. In subsequent work (Qian and Liu, 2013; Honnibal and Johnson, 2014; Wang et al., 2017), more effective models of word transcripts have been the main source of performance gains. The success of recent neural network systems raises the question of what the role is for prosody in future work. In the next section, we hypothesize where prosody might help and look at the relative frequency of these cases and the performance of a high accuracy disfluency detection algorithm in these contexts.

With the premise that there is a potential for prosody to benefit disfluency detection, we then propose a new approach to extracting prosodic features. A major challenge for all efforts to incorporate prosodic cues in spoken language understanding is the substantial variability in the acoustic correlates of prosody. For example, duration cues are expected to be useful – disfluencies are often associated with duration lengthening related to hesitation. However, duration varies with

Proceedings of NAACL-HLT 2019, pages 86–95
Minneapolis, Minnesota, June 2 - June 7, 2019.

phonetic context, word function, prosodic phrase structure, speaking rate, etc. To account for some of this variability, various feature normalization techniques are used, but typically these account for only limited contexts, e.g. phonetic context for duration or speaker pitch range for fundamental frequency. In our work, we introduce a mechanism for normalization using the full sentence context. We train a sequential neural prediction model to estimate distributions of acoustic features for each word, given the word sequence of a sentence. Then, the actual observed acoustic feature is used to find the prediction error, normalized by the estimated variance. We refer to the resulting features as innovations, which can be thought of as a non-linear version of the innovations in a Kalman filter. The innovations will be large when the acoustic cues do not reflect the expected prosodic structure, such as during hesitations, disfluencies, and contrastive or emphatic stress. The idea is to provide prosodic cues that are less redundant with the textual cues. We assess the new prosodic features in experiments on disfluency detection using the Switchboard corpus, exploring both early and late fusion techniques to integrate innovations with text features. Our analysis shows that prosody does help with detecting some of the more difficult types of disfluencies.

This paper has three main contributions. First, our analysis of a high performance disfluency detection algorithm confirms hypotheses about contexts where text-only models have high error rates. Second, we introduce a novel representation of prosodic cues, i.e. the innovation vector resulting from predicting prosodic cues given the whole sentence context. Analyses of the innovation distributions show expected patterns of prosodic cues at interruption points. Finally, we demonstrate improved disfluency detection performance on Switchboard by integrating prosody and text-based features in a neural network architecture, while comparing early and late fusion approaches.

2 How Might Prosody Help?

Disfluency detection algorithms based on text alone rely on the fact that disfluencies often involve parallel syntactic structure in the reparandum and the repair, as illustrated in the previous examples. In these cases, pattern match provides a strong cue to the disfluency. In addition, ungrammatical function word sequences are frequently associated with disfluencies, and these are relatively easy for a text-based model to learn. In some cases, an interregnum word (or words) provides a word cue to the interruption point. In the Switchboard corpus, only 15% of interruption points are followed by an interregnum, but it can provide a good cue when present. Prosody mainly serves to help identify the interruption point. Thus, for these types of disfluencies, it makes sense that prosodic cues would not really be needed.

	Reparandum Length				**% in**
Type	**1-2**	**3-5**	**6-8**	**8+**	**type**
repetition	1894	419	12	1	46%
rephrase	794	585	66	–	28%
restart	196	14	–	–	4%
nested*	149	262	158	118	13%

Table 1: Total word counts associated with reparanda of different lengths and types of disfluencies. *Counts for nested disfluencies exclude repetition tokens.

	Reparandum Length				
Type	**1-2**	**3-5**	**6-8**	**8+**	**overall**
repetition	0.99	0.99	1	1	0.99
rephrase	0.75	0.66	0.44	–	0.70
restart	0.41	0	–	–	0.39
nested*	0.79	0.66	0.62	0.21	0.62

Table 2: Percent of reparandum tokens that were correctly predicted as disfluent. *Statistics for nested disfluencies exclude repetition tokens.

Because disfluencies with a parallel syntactic structure do represent a substantial fraction of disfluencies in spontaneous speech, text-based algorithms have been relatively effective. The best models achieve F-scores of 86-91%[1] (Lou and Johnson, 2017; Zayats and Ostendorf, 2018; Wang et al., 2017, 2018). We hypothesize that many er-

---

[1] It is difficult to directly compare published results, because there are different approaches to tokenization that have a non-trivial impact on performance but are not well documented in the literature. Those differences include handling of fragment words, turn boundaries, and tokenization. For example, some studies use fragment features explicitly, while others omit them because speech recognition systems often miss them. Turn boundaries that do not end with a slash unit pose an ambiguity during speaker overlap: cross-turn 'sentences' can either be combined into a longer sentence or separated based on the turn boundary, which impacts what can be detected. Lastly, there are differences in whether contractions and possessives are split into two tokens, and whether conversational terms such as "you know" are combined into a single token.

	Reparandum Length	
Type	**1-2**	**3-5**
content-content	0.61 (30%)	0.58 (52%)
content-function	0.77 (20%)	0.66 (17%)
function-function	0.83 (50%)	0.80 (32%)

Table 3: Relative frequency of rephrases correctly predicted as disfluent for disfluencies that contain a content word in both the reparandum and repair (content-content), either the reparandum or repair (content-function) or in neither. Percentages in parentheses show the fraction of tokens belong to each category.

rors are associated with contexts where we expect that prosodic cues are useful, specifically the five cases below, with examples from the development set.

Restarts: Some disfluencies have no repair; the speaker simply restarts the sentence with no obvious parallel phrase.

`[ it would be + ] I think it's clear...`
`well [the +] uh i think what changed...`

Long disfluencies: These include distant pattern match or substantial rephrasing.

`[there is + for people who don't want to do the military service it would be neat if there were]`
`[what they're basically trying to do + i don't know up here in massachusetts anyhow what they're basically trying to do]`

Complex (nested) disfluencies: Disfluencies can occur within other disfluencies.

`[really + [[i + i] + we were really]...`
`[[to + to try to] + for two people who don't really have a budget to] ]...`

Non-trivial rephrasing: Rephrasing does not always involve a simple "rough copy" of a repair.

`[can + still has the option of]...`
`to keep them [in + uh quiet ]...`

Fluent repetitions: Contexts with fluent repetitions often include expressing a strong stance.

`a long long time ago...`
`she has very very black and white...`

In order to confirm that there is potential for prosody to help in these contexts, we first categorize the disfluencies. To avoid hand-labeling of categories, we distinguished disfluencies based on surface forms (repetition, rephrase, restart) and length of the disfluency reparandum. Word counts for the different categories are given in Table 1.

Conditioning on the different contexts, we analyze errors in the development set made by the high accuracy text-based disfluency detection system that is the baseline for this study (Zayats and Ostendorf, 2018). For this model, trained on Switchboard, the performance is 87.4 F-score (P=93.3, R=82.2) on the development set and 87.5 (P=93.1, R=82.5) on the test set. For each class, we measured the disfluency detection recall (relative frequency of reparandum tokens that were predicted correctly), as well as the percentage of tokens associated with each class. The results in Table 2 confirm that error rates are higher for restarts, longer rephrasings, and complex disfluencies.

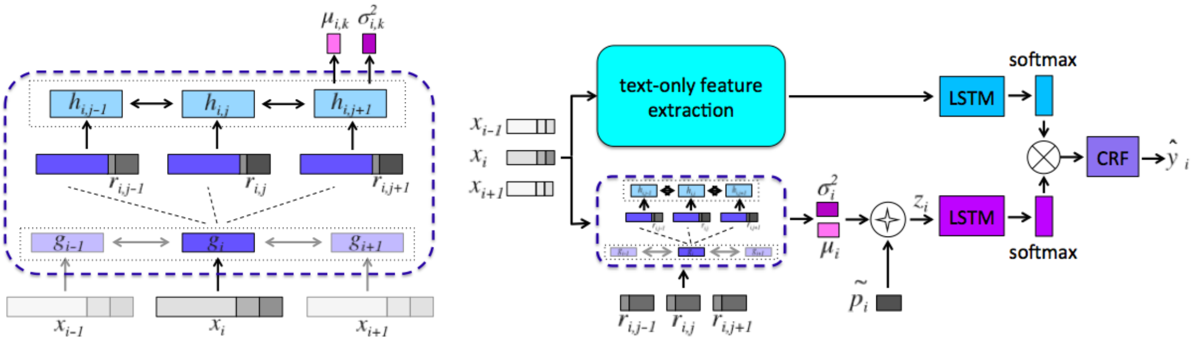
Rephrase disfluencies include both short lexical access errors, as well as non-trivial rewordings, which tend to be longer and involve content words. Table 3 breaks down performance for different lengths and word class to explore this difference. We found that rephrase disfluencies that contain content words are harder for the model to detect, compared to rephrases with function words only, and error increases for longer disfluencies.

Finally, the relative frequency of false positives in fluent repetitions is 0.35. Since fluent repetitions account for only 4% of all repetitions, the impact on overall performance is small.

The ultimate goal of a disfluency detection system is to perform well in domains other than Switchboard. Other datasets are likely to have different distributions of disfluencies, often with a higher frequency of those that are hard to detect, such as restarts and repairs (Zayats et al., 2014). In addition, due to the differences in vocabulary, disfluencies with content words are more likely to get misdetected if there is a domain mismatch. Thus, we hypothesize that prosody features can have a greater impact in a domain transfer scenario.

3 Method

Integrating prosodic cues has proved difficult because of the many sources of variability affecting the acoustic correlates, while systems that only use text achieve high performance. In this work, we propose a new approach that operates on differences in information found in text and prosody. In order to calculate such differences, we introduce innovation features, similar to the concept of innovations in Kalman filters. The key idea is to predict prosodic features based on text information, and then use the difference between the predicted and observed prosodic signal (innovations)



(a) Prosody prediction model

(b) Late fusion model

Figure 1: Prosody prediction (**left**) and late fusion (**right**) models. x_i is a contcatenation of token, POS and identity features embeddings at time i; r_i, j is a concatenation of stress and phone embeddings for phone j in token i; $\widetilde{p_i}$ is a vector of prosodic cues; g_i and h_i are hidden states of token level and phone level LSTMs, correspondingly.

as a new feature that is additionally used to predict disfluencies.

Let a prosody cue, p_i at time i be an observation associated with a sentence transcript containing n word tokens, $x_0 \dots x_n$. This observation can be modeled as a function of the sentence context $H(x_0 \dots x_n)$ perturbed with Gaussian noise $v_i \sim \mathcal{N}(0,\, \sigma_i^2)$:

$$p_i = H(x_0 \dots x_n) + v_i \qquad (1)$$

v_i can be viewed as a difference in information found between text and prosody. This difference can be measured using a z-score, which is a measure of how many standard deviations below or above the population mean an observation is. This framework can be viewed as a non-linear extension of a Kalman filter, where both H and σ_i^2 are parametrized using a neural network. Since disfluencies are irregularities in spoken language, they can be considered anomalies to fluent speech flow. A prosody flow that is unusual for a given word sequence, such as one that happens at interruption points, will likely have higher deviation from the predicted distribution. This anomaly in speech flow provides a strong signal when extracted using innovations, which is complementary to the text cues. In the next sections we give more details about the neural network architecture for text encoding, prosodic cues and innovation features, as well as an overview of the whole system.

3.1 Text Encoding for Prosody Prediction

We use both context around a word as well as subword information in text encoding for prosody prediction. Our text encoding consists of two bidirectional LSTMs: one on the token level and another on the phone level. First, we use pretrained word embeddings (Levy and Goldberg, 2014), part-of-speech tags embeddings, and identity features (whether the word is a filled pause, discourse marker, or incomplete) as inputs to a word-level bidirectional LSTM. Then, for each phone in a word we concatenate the phone embedding, its stress embedding, and the hidden state of the word-level LSTM for the corresponding token. The resulting phone feature vector is used as input to the second bidirectional LSTM. The last hidden state h_i of this second LSTM for token i summarizes the phone, stress and context information of that token, which we use to predict word-level prosodic cues. We use 3 categories of stress features in our experiments: primary, secondary and a non-stress phone.

3.2 Prosodic Cues

Our prosodic cues include:

Pause. Given a pause before a word, r_i, our pause cues are scaled as follows:

$$\widetilde{r_i} = \min(1, \ln(1 + r_i)) \qquad (2)$$

Pause information is extracted on a word-level using Mississippi State (MsState) time alignments (more details on data preprocessing in Section 4.1.) We use scaled real-valued pause information. Scaling pause lengths this way, including the threshold for pauses longer than 1 sec (which are rare), makes the pause distribution less skewed.

Word Duration. Similar to pause information, we extract word duration information using MsState time alignments. We do not need to do the standard word-based duration normalization, since the idea behind the innovation model is to normalize prosodic features using a richer context representation.

Fundamental frequency (F0) and Energy (E). Similar to Tran et al. (2018), we use three F0 features and three energy features. The three F0 features include normalized cross correlation function (NCCF), log-pitch weighted by probability of voicing (POV), and the estimated delta of log pitch. The three energy features include the log of total energy, the log of total energy from lower 20 mel-frequency bands and the log of total energy from higher 20 mel-frequency bands. The contour features are extracted from 25-ms frames with 10-ms hops using Kaldi (Povey et al., 2011). Our model is trained to predict the mean of these features across the frames in a word.

MFCCs. In addition to features used in Tran et al. (2018), we also use 13 mel-frequency cepstral coefficients, averaged at the word level, similar to F0 and energy features as described above.

3.3 Prosody Innovation Cues

Given a word-level text encoding h_i, for each token in a sentence we predict each of the k prosodic cues $\widetilde{p_i}^k$ listed above. We assume that the predicted prosody cues conditioned on text have a Gaussian distribution:

$$\begin{aligned}\widetilde{p_i}^k | h_i &\sim \mathcal{N}(\mu_{i,k}, \sigma^2_{i,k}) \\ \mu_{i,k} &= f(W_1^k h_i + b_1^k) \\ \sigma^2_{i,k} &= softplus(W_2^k h_i + b_2^k)\end{aligned} \tag{3}$$

W_1^k, b_1^k, W_2^k, b_2^k are learnable parameters; the activation function

$$softplus(x) = \log(1 + \exp(x))$$

ensures that the variance is always positive; f is an activation function, which is $softplus$ for pauses and durations, and $tanh$ for the rest of the prosodic cues. The objective function is a sum of the negative log-likelihood of prosodic cues $\widetilde{p_i}^k$ given text encoding. Then, given the predicted $\mu_{i,k}$, $\sigma^2_{i,k}$ and true values of prosodic cues $\widetilde{p_i}^k$, we calculate z-scores for each of the cues, which should have high absolute value for tokens with unusual prosodic behaviour:

$$z_i^k = \frac{\widetilde{p_i}^k - \mu_{i,k}}{\sigma_{i,k}} \tag{4}$$

The prosody prediction module is illustrated in Figure 1a.

These z-scores, or *innovations*, are used as additional features in our disfluency detection model. We train the prosody prediction model only on sentences that do not contain any disfluencies. Any unusual behaviours in disfluency regions, therefore, should have large innovation values predicted by our model.

3.4 Disfluency Detection System

Following (Zayats and Ostendorf, 2018), we use a bidirectional LSTM-CRF model as our disfluency detection framework. This framework uses a BIO tagging approach, where we predict whether each token is a part of a reparandum, repair or both. Following previous studies, the overall performance is measured in F-score of correctly predicted disfluencies in the reparandum. Previous work used textual features only. Here, we evaluate the importance of innovation cues with two types of multimodal fusion - early and late fusion. In early fusion, we concatenate innovations and/or prosody features with the rest of the textual features used in the framework at the input to LSTM layer. In late fusion, we create two separate models - one with only textual features and another with innovations and/or prosody features. Then we do a linear interpolation of the states of two models just before feeding the result to the CRF layer:

$$u_i^{shared} = \alpha u_i^{prosody} + (1 - \alpha) u_i^{text} \tag{5}$$

We tune the interpolation weight α and report the best in our experiments section. We train our model jointly, optimizing both prosodic cues prediction and disfluency detection. The schematic view of the late fusion system is presented in Figure 1b.

4 Experiments

In our experiments we evaluate the usefulness of innovation features, and compare it to baselines with text-only or raw prosodic cues. For each model configuration, we run 10 experiments with different random seeds. This alleviates the potential of making wrong conclusions due to "lucky/unlucky" random seeds. We report both the mean and best scores among the 10 runs.

4.1 Data Preprocessing

Switchboard (Godfrey et al., 1992) is a collection of telephone conversations between strangers,

	Model	**dev**		**test**		α
		mean	best	mean	best	
single	text	86.54	86.80	86.47	86.96	–
	raw	35.00	37.33	35.78	37.70	–
	innovations	80.86	81.51	80.28	82.15	–
early	text + raw	86.46	86.65	86.24	86.53	–
	text + innovations	86.53	86.77	86.54	87.00	–
	text + raw + innovations	86.35	86.69	86.55	86.44	–
late	text + raw	86.71	87.05	86.35	86.71	0.2
	text + innovations	**86.98**	**87.48**	**86.68**	**87.02**	0.5
	text + raw + innovations	86.95	87.30	86.60	86.87	0.5

Table 4: F1 scores on disfluency detection when using a single set of features (text-only, raw prosody features or innovation features), with early fusion and late fusion. “Raw” indicates the usage of original prosodic features (Section 3.2), while “innovations” indicate the usage of innovation features (Section 3.3).

but it ’s just you know leak leak leak everywhere
people should know that that ’s an option
and i think you do accomplish more after that
i mean [it was + it]
interesting thing [about gas is when + i mean about battery powered cars is]

Table 5: Examples of sentences where prosody innovations help. Words in red are correctly labeled when using prosody but not with text only. The first three show fluent phrases; the last two have disfluencies that are missed without prosody.

containing 1126 files hand-annotated with disfluencies. Because human transcribers are imperfect, the original transcripts contained errors. MsState researchers ran a clean-up project which hand-corrected the transcripts and word alignments (Deshmukh et al., 1998). In this work, we use the MsState version of the word alignments, which allows us to extract more reliable prosodic features. Since the corrected version of Switchboard does not contain updated disfluency annotations, we corrected the annotations using a semi-automated approach: we used a text-based disfluency detection algorithm to re-annotate tokens that were corrected by MsState, while keeping the rest of the original disfluency annotations. The result is referred to as a silver annotation. Most of the corrected tokens are repetitions and restarts. To assess the quality of the automatic mapping of disfluencies, we hand-annotated a subset (6.6k tokens, 453 sentences) of the test data and evaluated the performance of the silver annotation against the gold annotation, which has an F1 score of 90.1 (Prec 90.1, Rec 90.1). Comparing the performance estimates from gold and silver annotations on this subset, we find that the silver annotations give somewhat lower F1 scores (2-3% absolute), both due to lower precision and recall scores.

4.2 Results

Our experiments evaluate the use of innovations with two popular multimodal fusion approaches: early fusion and late fusion. Our baselines include models with text-only, prosody cues only (raw), and innovation features only as inputs. Since innovations require both text and raw prosodic cues, this baseline is multimodal. In addition, for the late fusion experiments, we show the optimal value of α, the interpolation weight from Equation 5. All experiment results are presented in Table 4.

We found that innovations are helpful in both early and late fusion frameworks, while late fusion performs better on average. The interpolation weight α for the late fusion experiments is high when innovations are used, which further indicates that innovation features are useful in overall prediction. Interestingly, innovation features alone perform surprisingly well. We also take a closer look at the importance of joint training of the disfluency detection system with prosody prediction. To do this, we pretrain the prosody pre-

i like to run [about + oh about] [two + two and a half] miles
the old-timers even the people who are technologists do n’t know how to operate
i do n’t know whether that ’s because they you know sort of give up hope
it must be really challenging to um try to juggle a job

Table 6: Examples of the sentences where prosody innovations hurt. Words in red are incorrectly labeled when using prosody but not with text only. The first shows a disfluency missed when using prosody; the other three are fluent regions with false detections.

diction part of the model first. Then, we train the full model with innovation inputs while freezing the part of the network responsible for predicting prosodic cues. The mean F-score of this disjointly trained model is 49.27% on the dev set, compared to 80.86% for the jointly trained model. This result suggests that training the system end-to-end in a multitask setup is very important.

5 Analysis

5.1 Error analysis

In order to better understand the impact of the prosody innovations, we perform an error analysis where we compare the predictions of two models: a late fusion model that uses both text and innovation features, and a baseline model that uses text only. All of the analysis is done on the dev set with the model that has the median performance out of 10 that were trained.

First, we extract all the sentences where the number of disfluency detection errors using the innovation model is lower than when using the text-only model (168 sentences). Examples of such sentences are presented in Table 5. By looking at the sentences where the model with innovations performs better, we see fluent repetitions and other ambiguous cases where audio is useful for correctly identifying disfluencies.

On the other hand, in Table 6, we have examples of sentences that have a higher number of errors when prosody is used (143 sentences). In the first example, the labeling of “two” as fluent by the model with prosody is arguably correct, with the repetition indicating a range rather than a correction. The next involves a parenthetical phrase, the start of which may be confused with an interruption point. In the last two cases, there is a prosodic disruption and an interegnum, but no correction.

In order to understand whether incorporating prosody through our model supports the hypotheses in Section 2, we compare the performance of two models for different categories of disfluencies. We found that using prosody innovations improves detection of: non-repetition disfluencies (from 68.2% to 73.7%), particularly for disfluencies with content words (65.2% to 71.0%); long repairs (64.0% to 72.7% and 40.0% to 64.6% for disfluencies with length of repair greater than 3 and 5 correspondingly); and restarts (from 36.0% to 37.4%). Prosodic innovations also help decrease the rate of false positives for fluent repetitions: the false positives rate decreased from 46.5% to 38.4%. However, the prosody model increases the false positives in other contexts, such as in the examples in Table 6.

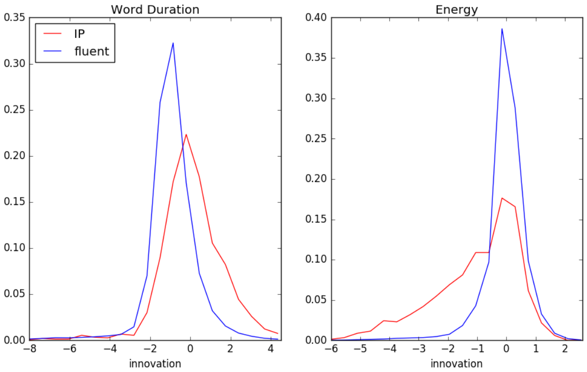


Figure 2: Histogram of innovations for word duration and energy features for words preceding an interruption point vs. fluent words.

5.2 Innovation Predictors

In order to understand what the model actually learns with respect to innovations, we look at innovation distributions for words preceding interruption points compared to fluent words. The histograms are presented in Figure 2. As expected, we see that words preceding interruption points have atypically longer duration and lower energy. The intonation features did not show substantial distribution differences, probably due to the overly simplistic word-level averaging strategy.

6 Related Work

Most work on disfluency detection falls into three main categories: sequence tagging, noisy-channel and parsing-based approaches. Sequence tagging approaches rely on BIO tagging with recurrent neural networks (Hough and Schlangen, 2015; Zayats et al., 2016; Wang et al., 2016; Zayats and Ostendorf, 2018; Lou et al., 2018). Noisy channel models operate on a relationship between the reparandum and repair for identifying disfluencies (Charniak and Johnson, 2001; Zwarts et al., 2010). Lou and Johnson (2017) used a neural language model to rerank sentences using the noisy channel model. Another line of work combined parsing and disfluency removal tasks (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Tran et al., 2018). Recently a transition-based neural model architecture was proposed for disfluency detection (Wang et al., 2017). The current state of the art in disfluency detection (Wang et al., 2018) uses a neural machine translation framework with a transformer architecture and additional simulated data. All of the models mentioned above rely heavily on pattern match features, hand-crafted or automatically extracted, that help to identify repetitions and disfluencies with parallel syntactic structure.

While prosodic features are useful for detecting interruption points (Nakatani and Hirschberg, 1994; Shriberg and Stolcke, 1997; Shriberg, 1999; Liu et al., 2006), recent methods on disfluency detection predominantly rely on lexical information exclusively. An exception is (Ferguson et al., 2015), which showed some gains using a simple concatenation of pause and word duration features. Similar to disfluency detection, parsing has seen little use of prosody in recent studies. However, Tran et al. (2018) recently demonstrated that that a neural model using pause, word and rhyme duration, f0 and energy helps in spoken language parsing, specifically in the regions that contain disfluencies.

Early fusion and late fusion are the two most popular types of modality fusion techniques. In recent years, more interesting modality fusion approaches were introduced, most of them where the fusion happens inside the model (Zadeh et al., 2017; Chen et al., 2017; Zadeh et al., 2018). Those methods usually require the model to learn interactions between modalities implicitly, by backpropagating the errors based on the main objective function with respect to the task. Other multimodal representation learning approaches learn a shared representation between multiple modalities (Andrew et al., 2013; Ryan Kiros, 2014; Xu et al., 2015; Suzuki et al., 2016), often targeting unsupervised translation from one modality to the other. In our work we use innovations as a novel representation learning approach, where our emphasis is on looking into complementary cues rather than similarity between multiple modalities.

7 Conclusions

In this paper, we introduce a novel approach to extracting acoustic-prosodic cues with the goal of improving disfluency detection, but also with the intention of impacting spoken language processing more generally. Our initial analysis of a text-only disfluency detection system shows that despite high performance of such models, there exists a big gap in the performance of text-based approaches for some types of disfluencies, such as restarts and non-trivial or long rephrases. Thus, prosody cues, which can be indicative of interruption points, have a potential to contribute towards detection of more difficult types of disfluencies. Since the acoustic-prosodic cues carry information related to multiple phenomena, it can be difficult to isolate the cues that are relevant to specific events, such as interruption points. In this work, we introduce a novel approach where we extract relevant acoustic-prosodic information using text-based distributional prediction of acoustic cues to derive vector z-score features, or innovations. The innovations point to irregularities in prosody flow that are not predicted by the text, helping to better isolate signals relevant to disfluency detection that are not simply redundant with textual cues. We explore both early and late fusion approaches to combine innovations with text-based features. Our experiments show that innovation features are better predictors of disfluencies compared to the original acoustic cues.

Our analysis of the errors and of the innovation features point to a limitation of the current work, which is in the modeling of F0 features. The current model obtains word-based F0 (and energy) features by simply averaging the values over the duration of the word, which loses any distinctions between rising and falling F0. By leveraging polynomial contour models, we expect to improve both intonation and energy features, which we hope will reduce some of the false detections associated with emphasis and unexpected fluent phrase boundaries.

An important next step is to test the system using ASR rather than hand transcripts. It is possible that errors in the transcripts could hurt the residual prediction, but if prosody is used to refine the recognition hypothesis, this could actually lead to improved recognition. Finally, we expect that the innovation model of prosody can benefit other NLP tasks, such as sarcasm and intent detection, as well as detecting paralinguist information.

References

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255.

Don Baron, Elizabeth Shriberg, and Andreas Stolcke. 2002. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Seventh International Conference on Spoken Language Processing*.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proc. NAACL*, pages 118–126.

Minghai Chen, Sen Wang, Paul Pu Liang, Tadas Baltrušaitis, Amir Zadeh, and Louis-Philippe Morency. 2017. Multimodal sentiment analysis with word-level fusion and reinforcement learning. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 163–171. ACM.

Neeraj Deshmukh, Aravind Ganapathiraju, Andi Gleeson, Jonathan Hamaker, and Joseph Picone. 1998. Resegmentation of switchboard. In *Fifth international conference on spoken language processing*.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proc. NAACL HLT*.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. ICASSP*, volume I, pages 517–520.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2(1):131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Proc. Interspeech*.

M. Johnson and E. Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proc. ACL*.

Matthew Lease, Mark Johnson, and Eugene Charniak. 2006. Recognizing disfluencies in conversational speech. *IEEE Trans. Audio, Speech, and Language Processing*, 14(5):169–177.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Robin J Lickley. 1994. *Detecting disfluency in spontaneous speech*. Ph.D. thesis, University of Edinburgh.

Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Trans. Audio, Speech and Language Processing*, 14:1526–1540.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency detection using auto-correlational neural networks. *arXiv preprint arXiv:1808.09092*.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency detection using a noisy channel model and a deep neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 547–553.

C. Nakatani and J. Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustical Society of America*, pages 1603–1616.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*.

Xian Qian and Yang Liu. 2013. Disuency detection using multi-step stacked learning. In *Proc. NAACL HLT*.

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proc. EMNLP*, pages 124–129.

Richard S. Zemel Ryan Kiros, Ruslan Salakhutdinov. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *https://arxiv.org/abs/1411.2539*.

E. Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Department of Psychology, University of California, Berkeley, CA.

E. Shriberg. 1999. Phonetic consequences of speech disfluency. In *Proc. International conference of Phonetics Sciences*, pages 619–622.

E. Shriberg and A. Stolcke. 1997. A prosody-only decision-tree model for disfluency detection. In *Proc. Eurospeech*, pages 2383–2386.

Elizabeth Shriberg, Rebecca Bates, and Andreas Stolcke. 1997. A prosody only decision-tree model for disfluency detection. In *Fifth European Conference on Speech Communication and Technology*.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2004. A lexically-driven algorithm for disfluency detection. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 157–160. Association for Computational Linguistics.

Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. 2016. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*.

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2018. Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information. In *Proc. NAACL*, pages 69–81.

Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. 2018. Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3529–3538.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2785–2794.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. 2018. Multi-attention recurrent network for human communication comprehension. *arXiv preprint arXiv:1802.00923*.

Vicky Zayats and Mari Ostendorf. 2018. Robust cross-domain disfluency detection with pattern match networks. *arXiv preprint arXiv:1811.07236*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional LSTM. In *Proc. Interspeech*.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2014. Multidomain disfluency and repair detection. In *Proc. Interspeech*.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proc. Coling*, pages 1371–1378.

Massively Multilingual Adversarial Speech Recognition

Oliver Adams, Matthew Wiesner, Shinji Watanabe, and David Yarowsky
Department of Computer Science
Johns Hopkins University, Baltimore MD, USA
`{oadams1,wiesner,shinjiw,yarowsky}@jhu.edu`

Abstract

We report on adaptation of multilingual end-to-end speech recognition models trained on as many as 100 languages. Our findings shed light on the relative importance of similarity between the target and pretraining languages along the dimensions of phonetics, phonology, language family, geographical location, and orthography. In this context, experiments demonstrate the effectiveness of two additional pretraining objectives in encouraging language-independent encoder representations: a context-independent phoneme objective paired with a language-adversarial classification objective.

1 Introduction

The main difficulty in creating automatic speech recognition (ASR) systems for a large number of the world's 7,000 languages is a lack of training data. Such data comes in the form of speech paired with transcriptions, a pronunciation lexicon, and text for language model training. A common technique in data-constrained settings is to learn language-independent representations of speech via multilingual training. Popular approaches include the use of multilingual bottleneck features (Vesely et al., 2012) as well as multilingual model training before fine-tuning to a *target* language (Scanzio et al., 2008; Vu et al., 2012).

Prior work in multilingual and cross-lingual speech recognition has been restricted to a small handful of the world's most-spoken languages, relying on multilingual corpora such as GlobalPhone (Schultz, 2002), the IARPA Babel corpora (Gales et al., 2014), or the VoxForge[1] corpora. Most work typically only reports on models trained on a subset of these languages.

In this paper we explore pretraining multilingual ASR models using speech from as many as 100 languages from the CMU Wilderness Multilingual Speech Dataset (Black, 2019).[2] To the best of our knowledge, this is the greatest number of languages that has been used in multilingual ASR model training to date. We perform experiments to guide the choice of languages used when pretraining the model and assess the relative importance of similarity between the pretraining languages and target language in terms of geographic location, phonology, phonetic inventory, language family and orthography.

We examine these variables in the context of two experimental setups: one where models are adapted to target language and target speakers, and one where models are adapted to target language but non-target speakers. The first task is relevant to language documentation contexts, which often involves transcribing speech of specific speakers for which there already exists some transcribed speech as training data (Michaud et al., 2018). The second case is relevant to incident response as modelled by LORELEI (Strassel and Tracey, 2016), where there may only be a single target-language consultant available for which transcribed speech can be elicited, but the goal is to have an ASR model that generalizes to multiple speakers.

Multilingual ASR training on such a scale presents challenges because of this language diversity. In order to guide the model to learn language-independent representations that are more amenable to adaptation, we experiment with two auxiliary training tasks. The first is context-independent phoneme sequence prediction to help bridge orthographic inconsistencies between languages. The second is a domain-adversarial classification objective (Ganin et al., 2016) over languages to encourage invariance

---

[1] `voxforge.org`

[2] `festvox.org/cmu_wilderness/index.html`

Proceedings of NAACL-HLT 2019, pages 96–108
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

of the model with respect to language-specific phenomena. The hierarchical combination of grapheme and phoneme objectives has only been used in monolingual end-to-end frameworks (Krishna et al., 2018; Rao and Sak, 2017). Language-adversarial training in ASR (Yi et al., 2018) has not been done at this scale before, nor in an end-to-end framework.

Our experiments are designed to answer the following questions:

1. Is there benefit in scaling multilingual model training to a large number of languages?
2. In what circumstances, if any, does the addition of a phoneme and/or language-adversarial objective improve multilingual models?
3. How should we choose languages with which to pretrain a multilingual model?
4. Do the answers to the above questions change when adapting to target versus non-target speakers in the target language?

We find that using the auxiliary objectives in pretraining facilitates model transfer to unseen languages, especially when the pretraining languages are very dissimilar (Section 6). When the target speakers are seen in adaptation (Section 7), similarity of the pretraining languages and the target language is more important than quantity of pretraining languages. Choosing as pretraining languages geographically proximal languages tends to help more than phonetically and phonologically similar but otherwise distant languages. However, when adapting to a handful of non-target speakers of the target language (Section 8), the domain mismatch caused by the unseen speaker, language, or recording environment degrades performance. Exposing the model to as many pretraining languages as possible becomes vital to minimize this mismatch. Results on this task demonstrate that a massively multilingual seed model substantially outperforms other seed models trained on languages similar to the target. We have provided an ESPnet recipe to train and test our models.[3]

2 Related Work

This paper builds on work on multilingual ASR, end-to-end ASR, and adversarial learning.

Multilingual transfer in ASR often relies on using bottle-neck features (Vesely et al., 2012; Vu et al., 2012; Karafiát et al., 2018) and adapting an acoustic model trained on one language to effectively recognize the sounds of other languages (Schultz and Waibel, 2001; Le and Besacier, 2005; Stolcke et al., 2006; Tóth et al., 2008; Plahl et al., 2011; Thomas et al., 2012; Imseng et al., 2014; Do et al., 2014; Heigold et al., 2013; Scharenborg et al., 2017). However, while most work uses less than 10 languages for model training, we include up to 100 languages in training.

End-to-end ASR has recently become popular, with approaches such as attention-based encoder-decoder models (Chorowski et al., 2015; Chan et al., 2015), the connectionist temporal classification (CTC) objective of Graves et al. (2006, 2013), or a combination of both (Kim et al., 2016; Hori et al., 2017). These approaches have also been deployed in multilingual settings (Toshniwal et al., 2017; Chiu et al., 2018; Müller et al., 2017; Dalmia et al., 2018; Watanabe et al., 2017a). Our baseline approach to multilingual knowledge transfer is most similar to Inaguma et al. (2018), and involves training a hybrid CTC-attention seed model.

Hierarchical and multi-task approaches including combining grapheme and phoneme prediction in monolingual contexts (Rao and Sak, 2017; Krishna et al., 2018) at different levels of the network, or using sub-word units of varying granularity (Sanabria and Metze, 2018), have been shown to improve ASR performance. In this paper we extend the approach of hierarchical placement of additional objectives in order to enforce language independent, transferable models.

Domain-adversarial training is one such method for encouraging the model to learn language independent representations. A key contribution of this paper is the use of a domain-adversarial classification objective (Ganin et al., 2016) over many languages in order to encourage the model to learn representations that are invariant to language. Domain-adversarial training incorporates an auxiliary domain classification task, but negates gradients for encoder weights before the parameter update in order to guide the encoder to produce hidden representations that fool the classifier: i.e. they minimize information about the language while still facilitating the

---

[3] `https://github.com/espnet/espnet/tree/master/egs/cmu_wilderness/asr1`

primary task of speech recognition.

Domain-adversarial training has been used in speech recognition to learn features invariant to noise conditions (Shinohara, 2016), accents (Sun, 2018), and sex (Tripathi et al., 2018). Most closely related to our work is that of Yi et al. (2018), who use a language-adversarial objective when preparing multilingual bottleneck features from four languages for a hidden Markov model (HMM) ASR pipeline. In contrast, our work uses an adversarial objective across many languages, pairing it with a context-independent phoneme objective in an end-to-end framework.

3 Data

We scraped the data that forms the CMU Wilderness dataset, using a freely available script.[4] This dataset consists of dramatized readings of the Bible in hundreds of languages. Each reading is ascribed a rating based on alignment quality which fits into one of these classes: `very good`, `good`, `okay`, and `not okay`.

The script used to preprocess the data uses a universal pronunciation module in Festival (Taylor et al., 1998)[5] to produce pronunciation lexicons using an approach based on that of UniTran (Yoon et al., 2007), which we use to create phonemic transcriptions.

3.1 Characteristics of the Speech

The dataset consists of readings of the Bible, with readings typically of just a few speakers, mostly male. These are often dramatized, with sound effects and background music. For many purposes this could be considered a limitation of the data. Although the characteristics of the speech are unique, it allows us to investigate multilingual models over many languages without the confounds of an overly noisy environment. It is not unreasonable to expect our findings to generalize to other speech recognition domains.

3.2 Evaluation Languages

While the dataset includes only a single reading of the Bible for most languages, there are a number with two or more. We evaluate on languages for which we can find two or more readings. This is so that we can compare adaptation to a target language but not the speakers of the target reading (we refer to this task as *language adaptation*, as explored in Section 8) with adaptation to the target language as well as the target reading (we refer to this task as *reading adaptation*). We additionally restricted the evaluation languages to those that have at least one `good` or `very good` reading in terms of alignment quality. Table 1 presents the evaluation languages and readings grouped by family or geographic location, along with their durations.

	Hours:minutes/quality per reading		
Aymara (ayr)	16:19/G	18:37/G	-
SB Quechua (quh)	27:41/G	20:02/G	-
Kekchi (kek)	19:32/G	18:30/G	-
Ixil (ixl)	35:06/VG	25:35/G	18:29/G
Malagasy (mlg)	12:29/NO	15:52/O	15:59/G
Indonesian (ind)	19:01/G	21:20/G	30:34/G
Garap (kia)	15:34/G	12:17/VG	-
Swedish (swe)	15:55/G	16:46/VG	-
Spanish (spn)	16:35/G	15:19/G	-

Table 1: The duration of each reading in the evaluation languages (ISO 639-3 language codes in parentheses), before our preprocessing. Alignment quality categories are `very good` (VG), `good` (G), `okay` (O), `not okay` (NO). *SB Quechua* denotes South Bolivian Quechua.

4 Auxiliary Training Objectives

In addition to scaling ASR training to 100 languages, a key contribution of our work is the use of a context-independent phoneme objective paired with a language-adversarial classification objective in a end-to-end grapheme-based neural network, as illustrated in Figure 1.

4.1 Baseline Model

Our experiments are conducted within the framework of a hybrid CTC-attention end-to-end neural model using ESPnet (Watanabe et al., 2017b), which uses an encoder-decoder architecture implemented in PyTorch (Paszke et al., 2017). The encoder we use consists of VGG-like convolution layers (Simonyan and Zisserman, 2014; Sercu et al., 2016) followed by a multilayer bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). The decoder uses location-based attention (Chorowski et al., 2015) and an LSTM. In addition to the attention, the decoder also incorporates CTC

---

[4] https://github.com/festvox/datasets-CMU_Wilderness

[5] http://www.cstr.ed.ac.uk/projects/festival/

probabilities over graphemes to encourage monotonicity in decoding.

4.2 Phoneme Objective

The end-to-end neural model performs direct grapheme prediction without recourse to a pronunciation lexicon as traditional hybrid HMM-DNN models do. Since different orthographies may be mutually disjoint or only weakly related to the phonetic content of the input speech, we use a context-independent phoneme CTC objective to encourage learning of representations independent of such orthographic idiosyncrasies.

We performed limited preliminary experiments to determine how best to use the phoneme objective, which corroborated recent work in hierarchical training objectives that supports inserting the phoneme objective in the layers below the final layer (Krishna et al., 2018). We also found that using the phoneme objective during adaptation was harmful and therefore in all reported experiments we use it only during multilingual pretraining.

4.3 Language-Adversarial Pretraining

For language-adversarial training we used a log-linear classifier over all languages seen in pretraining. An utterance-level mean of the penultimate encoder layer states is fed into the classifier. For each batch in training we update the network using the interpolated grapheme and phoneme objectives before a separate update step using the adversarial objective.

We follow the learning rate scheduling of Ganin et al. (2016), where the weight of the adversarial objective relative to the speech recognition tasks follows $\lambda(p) = \frac{2}{1+\exp(-10p)} - 1$ over the course of training, where $p \in [0, 1]$ is a measure of training progress. We drop the adversarial objective during target language adaptation.

5 Experimental Setup

5.1 Language Versus Reading Adaptation

We chose as target adaptation languages those languages for which we have multiple readings of the Bible. This allows us to assess adaptation of the pretrained multilingual model in two scenarios: *language adaptation* and *reading adaptation*. In *reading adaptation*, it is adapted to data from each reading of the target language, including the reading from which we select held-out evaluation utterances. In *language adaptation* it is adapted only to readings that are not represented in the evaluation set. This last case, of adapting to just one or several speakers of a new language (in order to ultimately have a system that generalizes beyond those speakers in the language) is not common in speech recognition experimentation. Results and findings for language adaptation will be presented in Section 8.

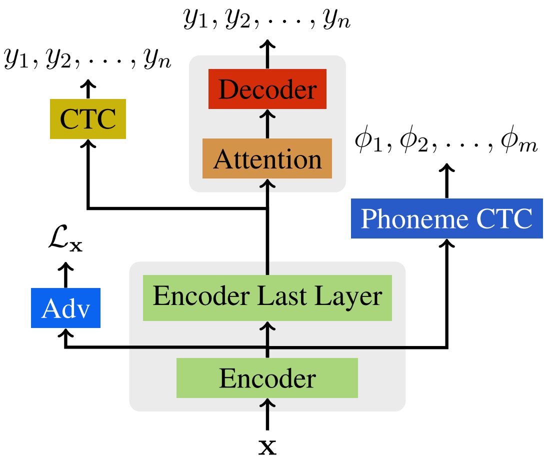


Figure 1: The end-to-end architecture used during pretraining. $\mathbf{x}$ is the input speech features, $y_1, y_2, \ldots, y_n$ is a character sequence the model is trained to output (eg. “knife”). $\phi_1, \phi_2, \ldots, \phi_m$ is a phoneme sequence the model is trained to output (eg. /naɪf/), and $\mathcal{L}_\mathbf{x}$ is the language identity of the input speech $\mathbf{x}$.

5.2 Training Settings

We established training, validation and test sets for each reading using a random 80/10/10 split. When pretraining or adapting the multilingual systems, we used the combined training sets of the constituent readings.

We used 80-dimensional log Mel filterbank features with 3-dimensional pitch features. We tuned hyperparameters for these models using one Aymara reading.[6] We found that a 4 layer encoder, 1 layer decoder with 768 for the encoder hidden size and projections, decoder hidden size, and attention hidden size yielded equal-best results with deeper models. These settings were then used for training the models used in our experiments.

For the training objective, we linearly interpolated the the attentional decoder cross-entropy loss with the grapheme CTC and phoneme CTC objectives. Equal weight was given to all three since we found that to be effective in preliminary experiments. Note however, that the effective weight of

---

[6] CMU Wilderness reading ID: AYMSBU.

Target	Mono	Que		Cyr		Que+Cyr			
		-	`+phn+adv`	-	`+phn+adv`	-	`+phn`	`+adv`	`+phn+adv`
Aymara	40.6	34.3	34.5 (+0.6%)	37.9	35.9 (-5.3%)	34.6	**34.2**	34.8	**34.2** (-1.2%)
SB Quechua	14.8	**13.8**	14.0 (+1.4%)	16.3	17.0 (+4.3%)	14.9	14.2	14.0	13.9 (-6.7%)
Indonesian	14.9	15.1	15.3 (+1.3%)	16.1	17.9 (+11.2%)	15.8	15.6	15.5	**14.7** (-7.0%)
		Avg. rel. Δ:	(+1.1%)	Avg. rel. Δ:	(+3.4%)			Avg. rel. Δ:	(-4.9%)

Table 2: Word error rate (%) comparison of multilingual models adapted to target languages, with and without auxiliary training objectives (relative change in parentheses). Additionally including Cyrillic-script languages in pretraining (Cyr) doesn't consistently improve over a model pretrained on Quechuan languages (Que) unless additional phoneme and language-adversarial objectives (`+phn` and `+adv`) are used in combination (`+phn+adv`). The auxiliary objectives help when pretraining languages are varied, but hinder when they are very similar. The final four columns suggest that the objectives are complementary. Average relative word error rate change for each pretraining set when adding in the auxiliary objectives (versus no aditional objectives) is indicated by *Avg. rel.* Δ.

the adversarial objective effectively changes over the course of training because of the learning rate scheduling mentioned in §4.3. We trained for 15 epochs in all cases except where otherwise noted.

Note that during adaptation we initialize the model using both the multilingual encoder and decoder. We found this to work best in preliminary experimentation on a Spanish reading.

6 Preliminary Investigation of the Auxiliary Objectives

In this section we evaluate the use of the auxiliary phoneme and language-adversarial objectives described in Section 4 on two divergent groups of languages that are distinct along a number of dimensions, including orthography, language family and phonology, in order to assess the auxiliary objectives' capacity to bridge the divide between these languages during pretraining. This serves as an initial exploration before further experiments in Section 7 and Section 8, where we choose from a broader set of pretraining languages.

Pretraining languages We pretrained models on two groups of languages separately and together. The first consists of six languages from the Quechuan language family, including subvarieties of Quechua I and II (qub, quf, qvs, qvw, qwh and qvh). We henceforth refer to this group as Que. The second consists of six languages that use the Cyrillic script and we refer to this group as Cyr. These languages include Nogai (nog), Bashkir (bak), Gagauz (gag), Khakas (kjh), Crimean Tatar (crh), and Russian (rus). With the exception of Russian, these languages are all Turkic. The character sets do not overlap between Que and Cyr and this was a deliberate choice in this preliminary experiment to maximize the differences between the two groups.

Evaluation languages To test the pretrained models in varied contexts, we evaluate our models on three languages: Central Aymara (ayr), South Bolivian Quechua (SB Quechua; quh), and Indonesian (ind). These languages vary in a number of dimensions: SB Quechua is very closely related to Que, while Indonesian is distant; Aymara is phonologically very similar to Quechuan languages, but is considered to be from a different family; Aymara had a high monolingual baseline error rate, while the others are lower; and Indonesian has three readings while the others have two. However, all evaluation languages use the Latin script. Note that in this section we assess performance in the reading adaptation case, while Section 8 presents results on the held-out reading case.

Experiments Table 2 compares the performance of monolingual target-language models to models adapted to the target language after being pretrained on Que, Cyr and their combination, Que+Cyr. Cyr pretraining underperforms pretraining with Que for all evaluation languages likely due to the orthographic mismatch with all of the evaluation languages. The model pretrained on Que+Cyr also underperforms Que. Introducing the auxiliary phoneme and language-adversarial objectives helps to overcome this performance loss, making the Que+Cyr-pretrained model the best for adaptation to Aymara and Indonesian. Que remained the best pretraining set for adaptation to SB Quechua, which is unsurprising given how well represented SB Quechua is by the languages included in the Quechuan language group. This suggests that when a substantial

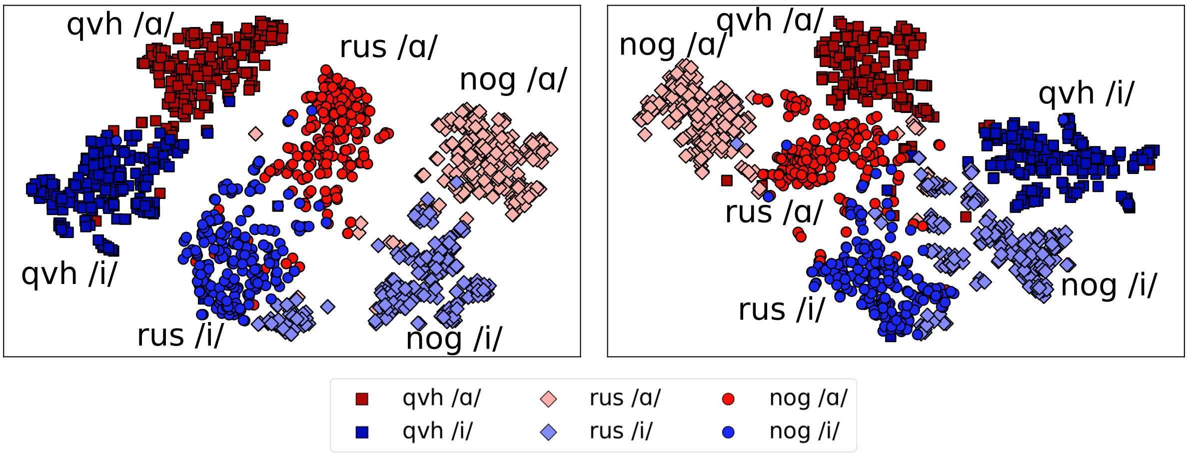


Figure 2: t-SNE representation of encoder states corresponding to /ɑ/ and /i/ across Quechua (Huamalies Dos de Mayo; qvh), Russian (rus), and Nogai (nog). *Left*: the model without the phoneme and adversarial objective. *Right*: the phoneme and language-adversarial objectives are added in, causing phoneme clusters between languages to gather closer together, and language to become less relevant in cluster placement.

amount of data in very closely related languages is available (in this case, close to 100 hours of QUE data), then there is little to be gained from highly unrelated languages.

When pretraining on QUE and CYR separately, the auxiliary objectives underperformed baseline multilingual pretraining on average. The variation in languages within these groups is far less than the variation between groups. Given that the phoneme and adversarial objectives are intended to overcome variation between pretraining languages, this result indicates that there must be a sufficient level of diversity in the pretraining languages before the auxiliary objectives are of benefit when adapting to certain target languages.

Results from pretraining on QUE+CYR showed either objective to help on average, and that the effects are complementary. Because of this, we opted to include them together in subsequent experimentation. We evaluated this best performing model on the larger set of other evaluation languages. Results in Table 3 show that in all cases multilingual pretraining of QUE+CYR with the auxiliary objectives outperformed its counterpart without the objectives (which frequently undeperformed the monolingual model), and in all but one case this led to an improvement over the monolingual baseline.[7]

To gain insight into how the auxiliary objectives change the representation of speech learnt by the models, we applied 2D t-SNE dimensionality reduction (Van Der Maaten and Hinton, 2008). Figure 2 plots the representations of two phonemes in three languages learnt by the encoder[8] in the case without and with the auxiliary objectives. In the multilingual pretraining baseline, six clusters are represented for each language–phoneme combination. These appear stratified by language, with different phoneme clusters within languages close to one another. With the auxiliary objectives, phoneme clusters between languages move closer to one another, while language identity becomes less relevant in determining which phoneme clusters neighbour one another. In the latter plot, the Nogai phonemes become separated by a Russian /ɑ/. This is particularly salient since the Nogai speaker was female, while the Russian speaker had a deep male voice.

7 Reading Adaptation

In the previous section we explored the use of two dissimilar groups of languages in a multilingual setup. Multilingual pretraining of languages from a different language family and script benefitted from an explicit phoneme objective and adversarial objective when there was sufficient diversity in the pretraining languages. However, a change in orthography was conflated with a change in language family, geographic location, and phono-

---

[7] However, this doesn't hold in the *language adaptation* scenario, where the auxiliary objectives help QUE+CYR only slightly; see Section 8.

[8] We established the correspondence between encoder states and phonemes by using forced alignment with Kaldi (Povey et al., 2011), taking the encoder state at the mid-point of the duration on the phonemes.

	Mono	Que+Cyr		Phonology		Geo		100-lang	
		-	+phn+adv	-	+phn+adv	-	+phn+adv	-	+phn+adv
ayr	40.6	34.6	34.2 (-1.2%)	**33.9**	34.5 (+1.8%)	35.4	34.9 (-1.4%)	34.2	34.5 (+0.9%)
quh	14.8	14.9	**13.9** (-6.7%)	14.4	14.5 (+0.7%)	15.5	14.8 (-4.5%)	15.1	14.7 (-2.6%)
kek	23.9	24.8	23.7 (-4.4%)	24.8	24.5 (-1.2%)	23.0	**22.9** (-0.4%)	24.9	24.4 (-2.0%)
ixl	20.7	21.2	20.1 (-5.2%)	-	-	**19.7**	20.1 (+2.0%)	20.8	20.6 (-1.0%)
mlg	45.2	43.5	**41.4** (-4.8%)	43.2	41.7 (-3.5%)	43.3	42.2 (-2.5%)	44.4	42.2 (-5.0%)
ind	14.9	15.8	14.7 (-7.0%)	**13.7**	14.3 (+4.4%)	14.0	**13.7** (-2.1%)	14.7	14.2 (-3.4%)
kia	14.6	14.6	13.2 (-9.6%)	-	-	**12.1**	**12.1** (-0.0%)	14.4	13.0 (-9.7%)
swe	**20.5**	22.7	21.6 (-4.9%)	26.4	24.2 (-8.3%)	22.0	21.2 (-3.6%)	23.9	24.6 (+2.9%)
spn	14.5	19.7	14.4 (-26.9%)	13.9	13.8 (-0.7%)	13.1	**12.1** (-7.6%)	15.8	14.8 (-6.3%)
		Avg. rel. Δ:	(-7.8%)	Avg. rel. Δ:	(-1.0%)	Avg. rel. Δ:	(-2.3%)	Avg. rel. Δ:	(-2.9%)

Table 3: Word error rate (%) comparison of adaptation of models pretrained on: Quechuan and Cyrillic-script languages (Que+Cyr), languages phonologically and phonetically similar to the target (Phon/Inv), geographically proximate languages (Geo), and a massively multilingual set of languages (100-lang). In each case we compared the average relative WER change when adding auxiliary phoneme and language-adversarial objectives (`+phn+adv`). Dashed entries had phonology and phonetic inventories that weren't well attested in URIEL, so were not assessed.

logical/phonetic characteristics. In this section, we investigate which factors are most important in choosing languages for multilingual pretraining and how useful it is to scale up model pretraining to many languages. This exploration is conducted in the reading adaptation scenario; language adaptation with unseen target speakers is addressed in Section 8. Beyond answering these questions, this investigation reveals more information about the utility of the proposed auxiliary objectives in different scenarios.

Phonology & Geography We test across a number of evaluation languages (c.f. Table 1) by determining, for each evaluation language, groups of pretraining languages that are similar to the evaluation languages in different ways. In order to determine language similarity in a principled way we used URIEL and *lang2vec* (Littell et al., 2017) to produce feature vectors for each language based on information from several linguistic resources before calculating their cosine similarity. For each language we used two feature vectors. The first is a concatenation of the lang2vec `phonology_average` and `inventory_average` vectors, characterizing phonological properties and phonetic inventory. The second represents geographic location. We denote these two groups Phon/Inv and Geo respectively.[9] Geographic proximity may serve as a proxy for other similarities not captured in Phon/Inv, including language family, orthographic similarity, and the likelihood of exchanged loan words.

We filtered for languages in the dataset with `good` or `very good` alignments before ranking them by cosine similarity with the evaluation languages in terms of phonological and phonetic similarity as well as geographical proximity. To create each of the pretraining sets, we took between 7 and 14 of the top languages, matching approximately the total duration of the phonetically/phonologically similar groups with the geographically proximate language groups.[10] For most languages, there is no overlap between the Geo and Phon/Inv sets.

Massively multilingual model As a further point of comparison, we pretrain a model on around 100 languages (denoted 100-lang), for approximately 1650 training hours in total.[11]

7.1 Auxiliary Objectives Findings

The results in Table 3 extend on our findings in Section 6, continuing to support the benefit of the use of the auxiliary objectives while shedding more light on the type of language variability the objectives help to overcome. Geo and 100-lang

---

[9] We didn't create Phon/Inv sets for Ixil and Garap because their phonological features and phonetic inventories were not well attested, and we didn't use the lang2vec language family vectors since most of the Quechuan languages were not captured as being highly similar to SB Quechua.

[10] An exhaustive list of the CMU Wilderness language codes for each pretraining group can be found in Appendix A, along with durations of each pretraining set.

[11] These models were pretrained for 6 epochs.

benefitted comparably from the objectives on average, while Phon/Inv did less so. Que+Cyr benefitted the most. This suggests that the objectives may help more when pretraining languages are orthographically, phonetically and phonologically diverse.

Unlike the other languages, the Swedish Phon/Inv vectors were not well attested. As a result the Swedish Phon/Inv group has languages with a similar phonetic inventory that were also unattested phonologically. This model underperformed the monolingual model by a large margin, suggesting that similarity of phonetic inventory alone may not be so useful alone without similarity of phonological features. Models pretrained on this set also benefitted the most from the auxiliary objectives. It may be the case that the auxiliary objectives push together representations of allophones within languages, and pronunciation variations of the same phonemes between languages. When Swedish is discounted, the average relative improvement when adding auxiliary objectives for Phon/Inv becomes negligable.

The Phon/Inv configurations are hurt by the auxiliary objectives for SB Quechua and Aymara and Indonesian. The Phon/Inv sets for the first two of these languages emphasized Quechuan languages, and this corroborates the indication in Section 6 that the auxiliary objectives may not help so much when pretraining languages are similar. On the other hand, the Indonesian Phon/Inv included Afro-Asiatic and Niger-Congo languages, as well an Indo-European language and Huave, a language isolate from Mexico, yet it was not improved by auxiliary objectives.

7.2 Choice of Pretraining Languages

The average relative word error rate (WER) change for Geo against Phon/Inv was -2.2% without auxiliary objectives, and -4.4% with them,[12] suggesting that features correlated with geography are useful for guiding pretraining language selection. Counter-examples were Aymara, SB Quechua and Malagasy, which performed worse when pretrained on Geo. In the case of SB Quechua, only one Quechuan language was represented in Geo (Inga), while Phon/Inv had three (qub, qvh, quf). Madagascar is far removed from where most Austronesian languages are spoken, so Malagasy's Geo set were almost all Niger-Congo languages, while the Phon/Inv had a diverse array of Austronesian, Indo European, Afro-Asiatic, Sino-Tibetan and Mayan languages. However, on average, these results suggest that geographical proximity is a decent guide to pretraining language selection. Another advantage is that it requires no explicit phonological features, making it applicable to a much larger number of languages.

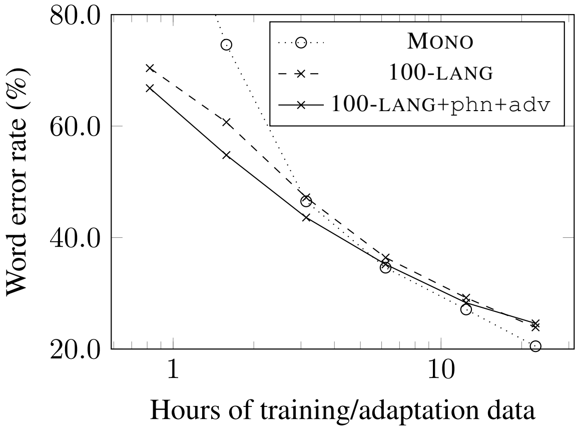


Figure 3: Scaling training/adaptation data for Swedish. Adapting to the full dataset, the auxiliary objectives underperformed both the monolingual and baselines, but yields an advantage when the model is adapted to less target language data.

The average relative WER change of 100-lang against Mono was +1.3%, indicating that massively multilingual pretraining by itself not useful if the target speakers are seen in training. Using the auxiliary objectives overcame the difference, resulting in a -1.6% average relative WER change. However, pretraining with Geo+phn+adv yielded an average relative delta of -7.4% over the monolingual model. Though more languages help, they are not necessarily better than geographically proximal languages (however, results are very different when not adapting to target speakers: see Section 8).

In two cases pretraining with 100-lang was hindered by the auxiliary objective. In one of these cases, Swedish, both 100-lang variations substantially underperformed the monolingual baseline. One possible reason is that there is enough target language and speaker data that the multilingual pretraining and auxiliary objectives offer no benefit. We scaled training/adaptation data for Swedish from under 1 hour. Figure 3 indicates that in this case the auxiliary objectives do lead to better initialization, with gains being lost only when around 5 hours of target language and reading data

---

[12] Discounting Swedish, this becomes +0.2% and -3.1%.

are seen.

8 Language Adaptation

Previous sections have addressed the reading adaptation scenario, where the ASR model is adapted to speech from the target reading (ie. where target speakers have been heard in adaptation). In this section we evaluate in a language adaptation scenario, adapting to readings in the target language, but not the target reading. The question of how well a multilingual model can be adapted to a language on the basis of recordings from a small number of target-language speakers is relevant to incident response situations such as those modelled by LORELEI (Strassel and Tracey, 2016), where a single language consultant is available for which recorded speech can be made. We performed experiments analogous to those of the previous sections where the evaluation reading was not seen in training or adaptation. This is a challenging task as the model must generalize to multiple speakers of a language on the basis of seeing only several in training. Most of the findings corroborate what was found in the previous sections. Here we highlight differences.

Massively multilingual pretraining led to substantially better performance than other methods, unlike in the reading adaptation task. For each evaluation language, the 100-Lang model outperformed the next best method, with one exception: Indonesian. In that case Geo set performed the best, as the languages were not only geographically proximate, but also consisted entirely of other Austronesian languages. The takeaway (c.f. Table 4) is that you should always use more pretraining languages unless you know your target speakers, as in the reading adaptation scenario.

Auxiliary objectives remained useful on the whole. However, while the difference in WER achieved when adding the auxiliary objectives was similar to those reported in Section 7 for Phon/Inv and 100-Lang, Geo and Que+Cyr no longer achieved improvements. Que+Cyr notably only achieved a -0.2% average relative WER change when adding the auxiliary objectives, while achieving -7.8% in the reading adaptation case. While the auxiliary objectives remained useful on the whole, their effect was dwarfed by the value of adding more languages.

	Mono	Que +Cyr	Phon +Inv	Geo	100-Lang	
ayr	91.4	86.3	86.7	87.2	**79.2**	(-8.2%)
quh	62.3	35.8	35.5	42.8	**30.1**	(-15.2%)
kek	75.6	74.3	73.8	74.4	**73.5**	(-0.4%)
ixl	81.8	79.8	-	78.4	**74.3**	(-6.9%)
mlg	103.6	68.3	64.0	63.7	**62.2**	(-2.4%)
ind	24.6	23.5	22.1	**21.1**	21.6	(+2.4%)
kia	57.2	51.5	-	49.9	**48.2**	(-6.4%)
swe	72.9	64.4	75.4	62.5	**55.1**	(-11.8%)
spn	44.8	33.8	33.4	32.7	**29.9**	(-8.6%)
Avg. rel. Δ of 100-Lang wrt. next best method:						(-6.0%)

Table 4: Adaptation to the non-target reading in the target language. All language sets use the auxiliary training objectives, which again exhibited an relative gain over the corresponding model without. The relative deltas of 100-Lang are with respect to the next closest model on a language-by-language basis.

Phonology versus Geography Geo sets with or without auxiliary objectives lost their edge over Phon/Inv, with high variance in scores. The amount of training data becomes the dominating variable affecting WER.

9 Conclusions

We have explored the utility of pretraining multilingual models on a variety of language sets, scaling to as as many as 100 languages. Our experiments have demonstrated the value of auxiliary phoneme and language-adversarial pretraining objectives in a multilingual end-to-end ASR framework, particularly when the pretraining languages are diverse. Our results suggest how to pick pretraining languages when target speakers are seen in the adaptation data: find geographically proximal languages. When adapting to just several non-target speakers, exposure to more speech in pretraining is the most important thing for model generality, even if from a wide range of dissimilar languages.

Acknowledgments

We would like to thank Tim Baldwin for an off-hand comment that planted the language-adversarial idea in the first author's head, and to Trevor Cohn for some related discussion. Thanks also go to Alexis Michaud and the reviewers for comments.

References

Alan W Black. 2019. CMU Wilderness Multilingual Speech Dataset. In *ICASSP*.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. In *ICASSP*.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *ICASSP*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *Advances in Neural Information Processing Systems 28*, pages 577–585.

Siddharth Dalmia, Ramon Sanabria, Florian Metze, and Alan W Black. 2018. Sequence-based multilingual low resource speech recognition. In *ICASSP*.

Van Hai Do, Xiong Xiao, Eng Siong Chng, and Haizhou Li. 2014. Cross-lingual phone mapping for large vocabulary speech recognition of under-resourced languages. *IEICE Transactions on Information and Systems*, E97-D(2).

Mark J F Gales, Kate M Knill, Anton Ragni, and Shakti P Rath. 2014. Speech recognition and keyword spotting for low-resource languages: BABEL project research at CUED. In *Spoken Language Technologies for Under-Resourced Languages*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1).

A Graves, A.-R. Mohamed, and G Hinton. 2013. Speech recognition with deep recurrent neural networks. *ICASSP*.

Alex Graves, Santiago Fernandez, Faustino Gomez, and Jurgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *ICML*.

Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. 2013. Multilingual acoustic models using distributed deep neural networks. In *ICASSP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. 2017. Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *INTERSPEECH*.

David Imseng, Petr Motlicek, Hervé Bourlard, and Philip N Garner. 2014. Using out-of-language data to improve an under-resourced speech recognizer. *Speech Communication*, 56.

Hirofumi Inaguma, Jaejin Cho, Murali Karthick Baskar, Tatsuya Kawahara, and Shinji Watanabe. 2018. Transfer learning of language-independent end-to-end ASR with language model fusion. *arXiv:1811.02134*.

Martin Karafiát, Murali Karthick Baskar, Shinji Watanabe, Takaaki Hori, Matthew Wiesner, and Jan “Honza” Černocký. 2018. Analysis of multilingual sequence-to-sequence speech recognition systems. *arXiv:1811.03451*.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2016. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *ICASSP*.

Kalpesh Krishna, Shubham Toshniwal, and Karen Livescu. 2018. Hierarchical multitask learning for CTC-based speech recognition. *arXiv:1807.06234*.

Viet Bac Le and Laurent Besacier. 2005. First steps in fast acoustic modeling for a new target language: application to Vietnamese. In *ICASSP*.

Patrick Littell, David Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *EACL*.

Alexis Michaud, Oliver Adams, Trevor Anthony Cohn, Graham Neubig, and Séverine Guillaume. 2018. Integrating automatic transcription into the language documentation workflow: Experiments with Na data and the Persephone toolkit. *Language Documentation & Conservation*, 12.

Markus Müller, Sebastian Stüker, and Alex Waibel. 2017. Phonemic and Graphemic Multilingual CTC Based Speech Recognition. *arXiv:1711.04564*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NeurIPS*.

Christian Plahl, Ralf Schlüter, and Hermann Ney. 2011. Cross-lingual portability of Chinese and English neural network features for French and German LVCSR. In *Automatic Speech Recognition and Understanding (ASRU), IEEE Workshop on*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, and Others. 2011. The Kaldi speech recognition toolkit. In *Automatic Speech Recognition and Understanding (ASRU), IEEE workshop on*.

Kanishka Rao and Haim Sak. 2017. Multi-accent speech recognition with hierarchical grapheme based models. *ICASSP*.

Ramon Sanabria and Florian Metze. 2018. Hierarchical multi task learning with CTC. In *IEEE Spoken Language Technology Workshop (SLT)*.

Stefano Scanzio, Pietro Laface, Luciano Fissore, Roberto Gemello, and Franco Mana. 2008. On the use of a multilingual neural network front-end. In *INTERSPEECH*.

Odette Scharenborg, Francesco Ciannella, Shruti Palaskar, Alan Black, Florian Metze, Lucas Ondel, and Mark Hasegawa-Johnson. 2017. Building an ASR system for a low-research language through the adaptation of a high-resource language ASR system: preliminary results. In *International Conference on Natural Language, Signal and Speech Processing (ICNLSSP)*.

Tanja Schultz. 2002. GlobalPhone: a multilingual speech and text database developed at Karlsruhe University. In *Seventh International Conference on Spoken Language Processing*.

Tanja Schultz and Alex Waibel. 2001. Experiments on cross-language acoustic Modeling. *EUROSPEECH'01*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11).

Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yorktown Heights. 2016. Very deep multilingual convolutional neural networks for LVCSR. In *ICASSP*.

Yusuke Shinohara. 2016. Adversarial multi-task learning of deep neural networks for robust speech recognition. In *INTERSPEECH*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

A. Stolcke, F. Grezl, Mei-Yuh Hwang, Xin Lei, N. Morgan, and D. Vergyri. 2006. Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. In *ICASSP*.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: data, tools, and resources for technology development in low resource languages. *LREC*.

Sining Sun. 2018. Domain adversarial training for accented speech recognition. In *ICASSP*.

Paul Taylor, Alan W Black, and Richard Caley. 1998. The architecture of the Festival speech synthesis system. In *ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*.

Samuel Thomas, Sriram Ganapathy, Hynek Hermansky, and Speech Processing. 2012. Multilingual MLP features for low-resource LVCSR systems. In *ICASSP*.

Shubham Toshniwal, Tara N. Sainath, Ron J. Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. 2017. Multilingual speech recognition with a single end-to-end model. In *ICASSP*.

László Tóth, Joe Frankel, Gábor Gosztolya, and Simon King. 2008. Cross-lingual portability of MLP-based tandem features - a case study for English and Hungarian. *INTERSPEECH*.

Aditay Tripathi, Aanchan Mohan, Saket Anand, and Maneesh Singh. 2018. Adversarial learning of raw speech features for domain invariant speech recognition. In *ICASSP*.

Laurens Van Der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Technical report.

Karel Vesely, Martin Karafiát, Frantisek Grezl, Marcel Janda, and Ekaterina Egorova. 2012. The language-independent bottleneck features. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*.

Ngoc Thang Vu, Florian Metze, and Tanja Schultz. 2012. Multilingual bottle-neck features and its application for under-resourced languages. In *The third International Workshop on Spoken Language Technologies for Under-resourced languages*.

Shinji Watanabe, Takaaki Hori, and John R Hershey. 2017a. Language independent end-to-end architecture for joint language identification and speech recognition. In *Automatic Speech Recognition and Understanding Workshop (ASRU), IEEE Workshop on*.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. 2017b. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal on Selected Topics in Signal Processing*.

Jiangyan Yi, Jianhua Tao, Zhengqi Wen, and Ye Bai. 2018. Adversarial multilingual training for low-resource speech recognition. *ICASSP*.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *ACL*.

A List of readings in each language set

Below is a collection of lists of the CMU Wilderness reading codes that comprise different groupings. This includes the target language readings; the Quechuan group; the Cyrillic-script group; the phonologically similar and geographically similar sets for each target language; and the massively multilingual set.

Target language readings MLGEIV, MLGRCV, MLGRPV, IX1WBT, IXIWBT, IXLWBT, INZNTV, INZSHL, INZTSI, QUHRBV, QUHSBB, QEJLLB, QUBPBS, QUFLLB, QVSTBL, QVWTBL, QWHLLB, SPNBDA, SPNWTC, KIABSC, KIAWBT, KEKIBS, KEKSBG, SWESFB, SWESFV, AYMSBU, AYMBSB.

Evaluation readings AYMSBU, MLGRPV, IXIWBT, INZSHL, QUHRBV, SPNBDA, KIAWBT, KEKIBS, SWESFV.

QUE (97.6 training hours) QEJLLB, QUBPBS, QUFLLB, QVSTBL, QVWTBL, QWHLLB.

CYR (59.6 training hours) NOGIBT, BAKIBT, GAGIB1, KJHIBT, RUSS76, CRHIBT.

AYR-PHON/INV (145.3 training hours) QUBPBS, TOBBSA, QUFLLB, QVSTBL, INBWBT, QEJLLB, JICWBT, QU1LSM, QUTIBS.

AYR-GEO (146.2 training hours) IGNSBB, TNATBL, GNWNTM, ESENTM, MCBTBL, GYRSBB, CBSBSP

QUH-PHON/INV (177.9 training hours) TOBBSA, DUGBTL, QUBPBS, TZHSBM, HUSLLB, NYFBTL, NCUWBT, QEJLLB, QUFLLB, HAGGIL, NZIBSG, MNBTBL.

QUH-GEO (178.5 training hours) GNWNTM, IGNSBB, TOBBSA, ENXBSP, GYRSBB, CAXSBB, CEGNTP, TNATBL, ESENTM, TERTBL.

KEK-PHON+INV (142.1 training hours) QU1LSM, QUTIBS, TZTWBT, TUFWYI, QWHLLB, PAGPBS, UDUSIM, YUASBM.

KEK-GEO (137.0 training hours) MOPWBT, POHBSG, CA1WBT, CKIWBT, TZTWBT, QU1LSM, QUTIBS, BZJBSW.

MLG-PHON/INV (198.2 training hours) RONBSR, TGLPBS, KVNWBT, HUVTBL, KBRSIM, TPMWBT, BTXLAI, KACUBS, WMWWYI, IGNSBB, HAEBSE, IBATIV, HILHPV, TZBSBM.

MLG-GEO (205.38 training hours) WMWWYI, VMWBSM, MFEBSM, SEHBSM, TOHSBM, CCESBM, KDCPBT, CWEPBT, KKIBST, NYYBST, KSBBST, KDNBSZ, DUGBTL, GOGBST.

IND-PHON/INV (193.1 training hours) IBATIV, TGLPBS, HAEBSE, KERABT, KACUBS, NYFBTL, RONBSR, CWTATB, HUVTBL, BTXLAI, IGNSBB, JAVNRF, DUGBTL, MNKBSG.

IND-GEO (191.5 training hours) SUNIBS, NIJLAI, JAVNRF, PSELAI, IBATIV, PTULAI, MVPLAI, PPKLAI, BEPLAI, NPYLAI, LEWLAI, MWVLAI.

SWE-PHON/INV (122.4 training hours) KDJBSU, NZIBSG, ANVWBT, DGABSG, SHKBSS, SLDTBL, KUSTBL, MUYWBT, NCUWBT, LIABSL, CKOGIL.

SWE-GEO (122.4 training hours) RMCWFW, EN1NIV, RMORAM, RONBSR, GAGIB1, GAGIBT, CRHIBT, KPVIBT, LTNNVV, ALSBSA, UDMIBT, XALIBT, BAKIBT.

SPN-PHON/INV (123.7 training hours) KVNWBT, HAEBSE, HUVTBL, GUGRPV, HUSLLB, GUMTBL, NYFBTL, KWIWBT.

SPN-GEO (129.5 training hours) PORARA, LTNNVV, EN1NIV, RMORAM, ALSBSA, RMCWFW, RONBSR, GAGIB1, GAGIBT, CRHIBT, TAQWBT, FUQWBT, MYKWBT.

100-LANG (1646.8 training hours) OBOWBT, ACUTBL, SEYWBT, HAUCLV, BZHPNG, AMKWBT, GAGIB1, GNWNTM, URBWBT, RUGWBT, PAUUBS, SEHBSM, SNNWBT, KQETBL, TGOTBL, NOGIBT, XTMTBL, OJ1CBS, TNATBL, AIAWYI, PABTBL, MEJTBL, TWBOMF, HUSLLB, ESENTM, BAKIBT, HNNOMF, IFAWBT, ENXBSP, ALJOMF, PXMBSM, JAISBG, PIRWBT, DOMBEC, NINWYI, BEPLAI, JAMBSW, TERTBL, LAWNTM, URATBL, AGNWPS, TPIPNG, TTCWBT, HUUTBL, NPYLAI, KJHIBT, AZZTBL, COKWBT, KWIWBT, SABWBT, PADTBL, GUMTBL, CRHIBT, QXRBSE, RMORAM, NHYTBL, TPPTBL, TUFWYI, ZLMAVB, PRFWBT, TWULAI, GAGIBT, FARWBT, OM1TBL, RUSS76, PTULAI, MIFWBT, MIYWYI, MRWNVS, KNETBL, PBCBSS, MYYWBT, ACHBSU, ACNBSM, ADETBL, AHKTBS, AK1BSG, ALPWBT, ALSBSA, ALTIBT, ANVWBT, ATGWYI, AVNWBT, AVUWBT, AYMBSB, AYMSBU, AZEBSA,

BEXWBT, BQJATB, BTXLAI, BZJBSW, CA1WBT, CARBSS, CAXSBB, CBSBSP, CMRWBT, CNLTBL, CNMRGB, CRNWBT.

Lost in Interpretation:
Predicting Untranslated Terminology in Simultaneous Interpretation

Nikolai Vogler and **Craig Stewart** and **Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
{nikolaiv,cas1,gneubig}@cs.cmu.edu

Abstract

Simultaneous interpretation, the translation of speech from one language to another in real-time, is an inherently difficult and strenuous task. One of the greatest challenges faced by interpreters is the accurate translation of difficult terminology like proper names, numbers, or other entities. Intelligent computer-assisted interpreting (CAI) tools that could analyze the spoken word and detect terms likely to be untranslated by an interpreter could reduce translation error and improve interpreter performance. In this paper, we propose a task of predicting which terminology simultaneous interpreters will leave untranslated, and examine methods that perform this task using supervised sequence taggers. We describe a number of task-specific features explicitly designed to indicate when an interpreter may struggle with translating a word. Experimental results on a newly-annotated version of the NAIST Simultaneous Translation Corpus (Shimizu et al., 2014) indicate the promise of our proposed method.[1]

1 Introduction

Simultaneous interpretation (SI) is the act of translating speech in real-time with minimal delay, and is crucial in facilitating international commerce, government meetings, or judicial settings involving non-native language speakers (Bendazzoli and Sandrelli, 2005; Hewitt et al., 1998). However, SI is a cognitively demanding task that requires both active listening to the speaker and careful monitoring of the interpreter's own output. Even accomplished interpreters with years of training can struggle with unfamiliar concepts, fast-paced speakers, or memory constraints (Lambert and Moser-Mercer, 1994; Liu et al., 2004). Human short-term memory is particularly at odds with the simultaneous interpreter as he or she must consistently recall and translate specific terminology uttered by the speaker (Lederer, 1978; Darò and Fabbro, 1994). Despite psychological findings that rare words have long access times (Balota and Chumbley, 1985; Jescheniak and Levelt, 1994; Griffin and Bock, 1998), listeners expect interpreters to quickly understand the source words and generate accurate translations. Therefore, professional simultaneous interpreters often work in pairs (Millán and Bartrina, 2012); while one interpreter performs, the other notes certain challenging items, such as dates, lists, names, or numbers (Jones, 2002).

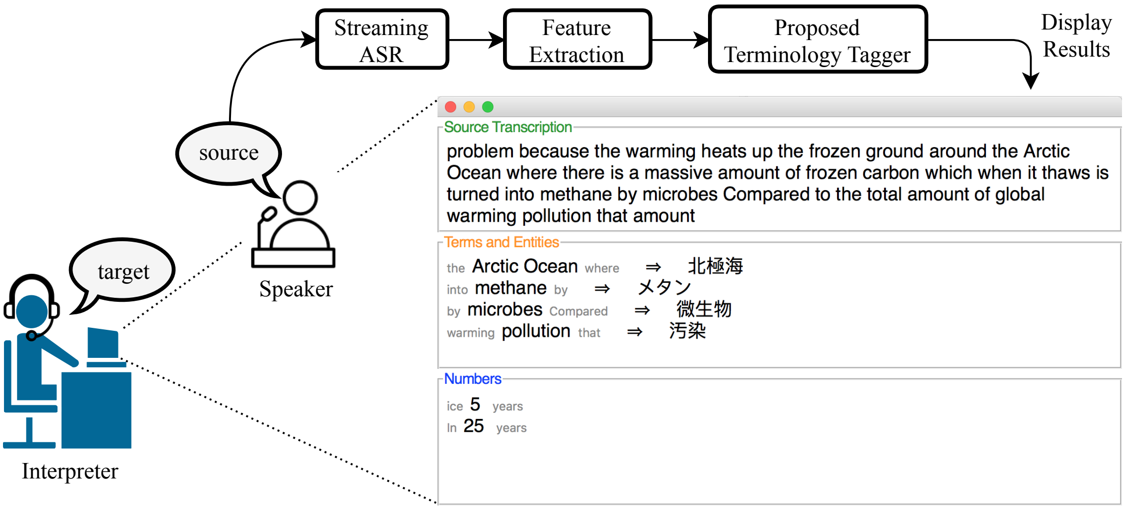
Computers are ideally suited to the task of recalling items given their ability to store large amounts of information, which can be accessed almost instantaneously. As a result, there has been recent interest in developing computer-assisted interpretation (CAI; Plancqueel and Werner; Fantinuoli (2016, 2017b)) tools that have the ability to display glossary terms mentioned by a speaker, such as names, numbers, and entities, to an interpreter in a real-time setting. Such systems have the potential to reduce cognitive load on interpreters by allowing them to concentrate on fluent and accurate production of the target message.

These tools rely on automatic speech recognition (ASR) to transcribe the source speech, and display terms occurring in a prepared glossary. While displaying all terminology in a glossary achieves *high recall* of terms, it suffers from *low precision*. This could potentially have the unwanted effect of cognitively overwhelming the interpreter with too many term suggestions (Stewart et al., 2018). Thus, an important desideratum of this technology is to only provide terminology

---

[1] Code is available at `https://github.com/nvog/lost-in-interpretation`. Term annotations for the NAIST Simultaneous Translation Corpus will be provided upon request after confirmation that you have access to the corpus, available at `https://ahcweb01.naist.jp/resource/stc/`.

Proceedings of NAACL-HLT 2019, pages 109–118
Minneapolis, Minnesota, June 2 - June 7, 2019.



problem because the warming heats up the frozen ground around the Arctic Ocean where there is a massive amount of frozen carbon which when it thaws is turned into methane by microbes Compared to the total amount of global warming pollution that amount

the Arctic Ocean where ⇒ 北極海
into methane by ⇒ メタン
by microbes Compared ⇒ 微生物
warming pollution that ⇒ 汚染

ice 5 years
In 25 years

Figure 1: The simultaneous interpretation process, which could be augmented by our proposed terminology tagger embedded in a computer-assisted interpreting interface on the interpreter's computer. In this system, automatic speech recognition transcribes the source speech, from which features are extracted, input into the tagger, and term predictions are displayed on the interface in real-time. Finally, machine translations of the terms can be suggested.

assistance when the interpreter requires it. For instance, an NLP tool that learns to predict only terms an interpreter is likely to miss could be integrated into a CAI system, as suggested in Fig. 1.

In this paper, we introduce the task of predicting the terminology that simultaneous interpreters are likely to leave untranslated using *only* information about the source speech and text. We approach the task by implementing a supervised, sliding window, SVM-based tagger imbued with delexicalized features designed to capture whether words are likely to be missed by an interpreter. We additionally contribute new manual annotations for untranslated terminology on a seven talk subset of an existing interpreted TED talk corpus (Shimizu et al., 2014). In experiments on the newly-annotated data, we find that intelligent term prediction can increase average precision over the heuristic baseline by up to 30%.

2 Untranslated Terminology in SI

Before we describe our supervised model to predict untranslated terminology in SI, we first define the task and describe how to create annotated data for model training.

2.1 Defining Untranslated Terminology

Formally, we define untranslated terminology with respect to a source sentence S, sentence created by a translator R, and sentence created by an interpreter I. Specifically, we define any consecutive sequence of words $s_{i:j}$, where $0 \leq i \leq N-1$ (inclusive) and $i < j \leq N$ (exclusive), in source sentence $S_{0:N}$ that satisfies the following criteria to be an untranslated term:

- **Termhood:** It consists of only numbers or nouns. We specifically focus on numbers or nouns for two reasons: (1) based on the interpretation literature, these categories contain items that are most consistently difficult to recall (Jones, 2002; Gile, 2009), and (2) these words tend to have less ambiguity in their translations than other types of words, making it easier to have confidence in the translations proposed to interpreters.

- **Relevance:** A translation of $s_{i:j}$, we denote t, occurs in a sentence-aligned reference translation R produced by a translator in an offline setting. This indicates that in a time-unconstrained scenario, the term *should* be translated.

- **Interpreter Coverage:** It is not translated, literally or non-literally, by the interpreter in interpreter output I. This may reasonably allow us to conclude that translation thereof may have presented a challenge, resulting in the content not being conveyed.

Importantly, we note that the phrase *untranslated* terminology entails words that are either dropped mistakenly, intentionally due to the interpreter deciding they are unnecessary to carry across the meaning, or mistranslated. We contrast this with *literal* and *non-literal* term coverage, which encompasses words translated in a verbatim and a paraphrastic way, respectively.

2.2 Creating Term Annotations

To obtain data with labels that satisfy the previous definition of untranslated terminology, we can leverage existing corpora containing sentence-aligned source, translation, and simultaneous interpretation data. Several of these resources exist, such as the NAIST Simultaneous Translation Corpus (STC) (Shimizu et al., 2014) and the European Parliament Translation and Interpreting Corpus (EPTIC) (Bernardini et al., 2016). Next, we process the source sentences, identifying terms that satisfy the termhood, relevance, and interpreter coverage criteria listed previously.

- **Termhood Tests:** To check termhood for each source word in the input, we first part-of-speech (POS) tag the input, then check the tag of the word and discard any that are not nouns or numbers.

- **Relevance and Interpreter Coverage Tests:** Next, we need to measure relevancy (whether a corresponding target-language term appears in translated output), and interpreter coverage (whether a corresponding term *does not* appear in interpreted output). An approximation to this is whether one of the translations listed in a bilingual dictionary appears in the translated or interpreted outputs respectively, and as a first pass we identify all source terms with the corresponding target-language translations. However, we found that this automatic method did not suffice to identify many terms due to lack of dictionary coverage and also to non-literal translations. To further improve the accuracy of the annotations, we commissioned human translators to annotate whether a particular source term is translated literally, non-literally, or untranslated by the translator or interpreters (details given in §4).

Once these inclusion criteria are calculated, we can convert all untranslated terms into an appropriate format conducive to training supervised taggers. In this case, we use an IO tagging scheme (Ramshaw and Marcus, 1999) where all words corresponding to untranslated terms are assigned the label I, and all others are assigned a label O, as shown in Fig. 2.

Src	In California, there has been a [40] percent O O O O O O I O decline in the [Sierra snowpack]. O O O I I
Interp	カリフォルニアでは、 4 パーセント California 4 percent 少なくなってしまいました。 decline

Figure 2: A source sentence and its corresponding interpretation. Untranslated terms are surrounded by brackets and each word in the term is labeled with an I-tag. The interpreter mistakes the term *40* for *4*, and omits *Sierra snowpack*.

3 Predicting Untranslated Terminology

With supervised training data in hand, we can create a model for predicting untranslated terminology that could potentially be used to provide interpreters with real-time assistance. In this section, we outline a couple baseline models, and then describe an SVM-based tagging model, which we specifically tailor to untranslated terminology prediction for SI by introducing a number of hand-crafted features.

3.1 Heuristic Baselines

In order to compare with current methods for term suggestion in CAI, such as Fantinuoli (2017a), we first introduce a couple of heuristic baselines.

- **Select noun/# POS tag:** Our first baseline recalls all words that meet the termhood requirement from §2. Thus, it will achieve perfect recall at the cost of precision, which will equal the percentage of I-tags in the data.

- **Optimal frequency threshold:** To increase precision over this naive baseline, we also experiment with a baseline that has a frequency threshold, and only output words that are rarer than this frequency threshold in a large web corpus, with the motivation that rarer words are more likely to be difficult for translators to recall and be left untranslated.

3.2 SVM-based Tagging Model

While these baselines are simple and intuitive, we argue that there are a large number of other features that indicate whether an interpreter is

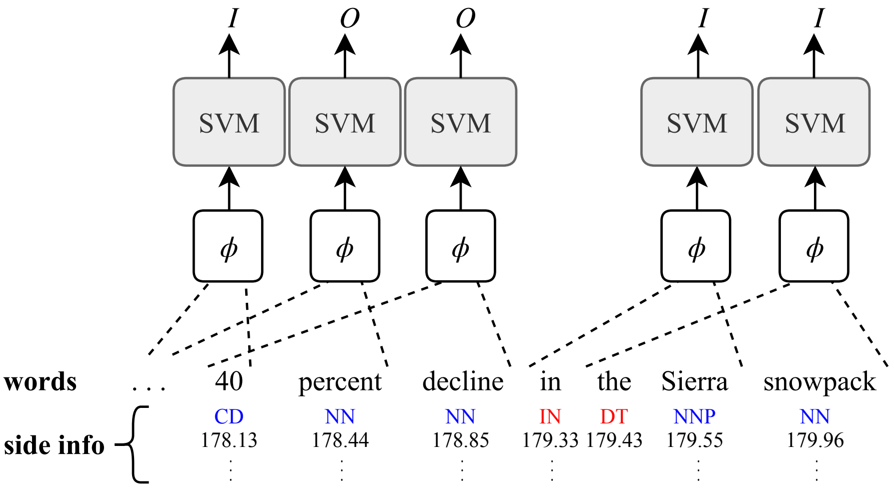


Figure 3: Our tagging model at prediction time. A sliding window SVM, informed by a task-specific feature function ϕ with access to the POS tags, source speech timing (in seconds), and other information, predicts whether or not words matching the termhood constraint (in blue) are likely to be left untranslated in SI.

likely to leave a term untranslated. We thus define these features, and resort to machine-learned classifiers to integrate them and improve performance. State-of-the-art sequence tagging models process sequences in both directions prior to making a globally normalized prediction for each item in the sequence (Huang et al., 2015; Ma and Hovy, 2016). However, the streaming, real-time nature of simultaneous interpretation constrains our model to sequentially process data from left-to-right and make local, monotonic predictions (as noted in Oda et al. (2014); Grissom II et al. (2014), among others). Therefore, we use a sliding-window, linear support vector machine (SVM) classifier (Cortes and Vapnik, 1995; Joachims, 1998) that uses only local features of the history to make independent predictions, as depicted in Fig. 3.[2] Formally, given a sequence of source words with their side information (such as timings or POS tags) $S = s_{0:N}$, we slide a window W of size k incrementally across S, extracting features $\phi(s_{i-k+1:i+1})$ from s_i and its $k-1$ predecessors.

Since our definition of terminology only allows for nouns and numbers, we restrict prediction to words of the corresponding POS tags $Q = \{\text{CD, NN, NNS, NNP, NNPS}\}$ using the Stanford POS tagger (Toutanova et al., 2003). That is, we assign a POS tag p_i to each word from s_i and only extract features/predict using the classifier if $p_i \in Q$; otherwise we always assign the Outside tag. This disallows words that are of other POS tags from being classified as untranslated terminology and greatly reduces the class imbalance issue when training the classifier.[3]

3.3 Task-specific Features

Due to the fact that only a small amount of human-interpreted human-annotated data can be created for this task, it is imperative that we give the model the precise information it needs to generalize well. To this end, we propose multiple task-specific, non-lexical features to inform the classifier about certain patterns that may indicate terminology likely to be left untranslated.

- **Elapsed time:** As discussed in §1, SI is a cognitively demanding task. Interpreters often work in pairs and usually swap between active duty and notetaking roles every 15-20 minutes (Lambert and Moser-Mercer, 1994). Towards the end of talks or long sentences, an interpreter may become fatigued or face working memory issues—especially if working alone. Thus, we monitor the number of minutes elapsed in the talk and the index of the word in the talk/current sentence to inform the classifier.

- **Word timing:** We intuit that a presenter's quick speaking rate can cause the simultaneous interpreter to potentially drop some terminology. We obtain word timing informa-

---

[2] We also experimented with a unidirectional LSTM tagger (Hochreiter and Schmidhuber, 1997; Graves, 2012), but found it ineffective on our small amount of annotated data.

[3] We note that a streaming POS tagger would have to be used in a real-time setting, as in (Oda et al., 2015).

tion from the source speech via forced alignment tools (Ochshorn and Hawkins, 2016; Povey et al., 2011). The feature function extracts both the number of words in the past m seconds and the time deltas between the current word and previous words in the window.

- **Word frequency:** We anticipate that interpreters often leave rarer source words untranslated because they are probably more difficult to recall from memory. On the other hand, we would expect loan words, words adopted from a foreign language with little or no modification, to be easier to recognize and translate for an interpreter. We extract the binned unigram frequency of the current source word from the large monolingual Google Web 1T Ngrams corpus (Brants and Franz, 2006). We define a loan word as an English word with a Katakana translation in the bilingual dictionaries (eij; Breen, 2004).
- **Word characteristics and syntactic features:** We extract the number of characters and number of syllables in the word, as determined by lookup in the CMU Pronunciation dictionary (Weide, 1998). Numbers are converted to their word form prior to dictionary lookup. Generally, we expect longer words, both by character and syllable count, to represent more technical or marked vocabulary, which may be challenging to translate. Additionally, we syntactically inform the model with POS tags and regular expression patterns for numerals.

These features are extracted via sliding a window over the sentence, as displayed in Fig. 3 and discussed in §3.2. Thus, we also utilize previous information from the window when predicting for the current word. This previous information includes past predictions, word characteristics and syntax, and source speech timing.

4 Experimental Annotation and Analysis

In this section, we detail our application of the term annotation procedure in §2 to an SI corpus and analyze our results.

4.1 Annotation of NAIST STC

For SI data, we use a seven-talk, manually-aligned subset of the English-to-Japanese NAIST STC (Shimizu et al., 2014), which consists of source subtitle transcripts, En→Ja offline translations, and interpretations of English TED talk videos from professional simultaneous interpreters with 1, 4, and 15 years of experience, who are dubbed B-rank, A-rank, and S-rank[4]. TED talks offer a unique and challenging format for simultaneous interpreters because the speakers typically talk in-depth about a single topic, and such there are many new terms that are difficult for an interpreter to process consistently and reliably. The prevalence of this difficult terminology presents an interesting testbed for our proposed method.

First, we use the Stanford POS Tagger (Toutanova et al., 2003) on the source subtitle transcripts to identify word chunks with a POS tag in {CD, NN, NNS, NNP, NNPS}, discarding words with other tags. After performing word segmentation on the Japanese data using KyTea (Neubig et al., 2011), we automatically detect for translation coverage between the source subtitles, SI, and translator transcripts with a string-matching program, according to the relevance and coverage tests from §2. The En↔Ja EIJIRO (2.1m entries) (eij) and EDICT (393k entries) (Breen, 2004) bilingual dictionaries are combined to provide term translations. Additionally, we construct individual dictionaries for each TED talk with key acronyms, proper names, and other exclusive terms (e.g., *UNESCO*, *CO2*, *conflict-free*, *Pareto-improving*) to increase this automatic coverage. Nouns are lemmatized prior to lookup in the bilingual dictionary, and we discard any remaining closed-class function words.

While this automatic process is satisfactory for identifying if a translated term occurs in the translator's or interpreters' transcripts (relevancy), it is inadequate for verifying the terms that occur in the translator's transcript, but *not* the interpreters' outputs (interpreter coverage). Therefore, we commissioned seven professional translators to review and annotate those source terms that could not be marked as translated by the automatic process as either *translated*, *untranslated*, or *non-literally translated* in each target sentence. Lastly, we add I-tags to each word in the untranslated terms and O-tags to the words in literally and non-literally translated terms.

---

[4] {B, A, S}-rank is the Japanese equivalent to {C, B, A}-rank on the international scale.

	trans.		non-lit.		raw untrans.	
T/I	**#**	**%**	**#**	**%**	**#**	**%**
T	2,213	80	158	6	401	14
B	1,134	41	92	3	1,546	56
A	1,151	42	114	4	1,507	54
S	1,531	55	170	6	1,071	39

Table 1: Translated, non-literally translated, and raw untranslated term annotations obtained in the annotation process using the NAIST STC for (T)ranslator, and {B,A,S}-rank SI. Note that these *raw* untranslated term figures are directly from the annotation process, prior to filtering based off of the term relevancy constraint from §2.

4.2 Annotation Analysis

Table 1 displays the term coverage annotation statistics for the translators and interpreters. Since translators performed in an offline setting without time constraints, they were able to translate the largest number of source terms into the target language, with 80% being literally translated, and 6% being non-literally translated. On the other hand, interpreters tend to leave many source terms uncovered in their translations. The A-rank and B-rank interpreters achieve roughly the same level of term coverage, with the A-rank being only slightly more effective than B-rank at translating terms literally and non-literally. This is in contrast with Shimizu et al. (2014)'s automatic analysis of translation quality on a three-talk subset, in which A-rank has slightly higher translation error rate and lower BLEU score (Papineni et al., 2002) than the B-rank interpreter. The most experienced S-rank interpreter leaves 17% fewer terms than B-rank uncovered in the translations. More interestingly, the number of non-literally translated terms also correlates with experience-level. In fact, the S-rank interpreter actually exceeds the translator in the number of non-literal translations produced. Non-literal translations can occur when the interpreter fully comprehended the source expression, but chose to generate it in a way that better fit the translation in terms of fluency.

In Table 2, we show the number of terms left untranslated by each interpreter rank after processing our annotations for the relevancy constraint of §2. Since the number of per-word I-tags is only slightly higher than the number of untranslated terms, most such terms consist of only a single word of about 6.5 average characters for all ranks. Capitalized terms (i.e., named entities/locations) constitute about 14% of B-rank, 13% of A-rank, and 15% of S-rank terms. Numbers represent about 5% of untranslated terms for each rank.

		% I-tag of	
SI	**# untrans. terms**	**all**	**noun/#**
B-rank	1,256	10.8	45.4
A-rank	1,206	10.4	43.6
S-rank	812	7.0	29.6

Table 2: Final untranslated term count and number of I-tags after filtering based off of the *relevancy* constraint (§2). That is, only the raw untranslated source terms that appear in the translator's transcript are truly considered untranslated.

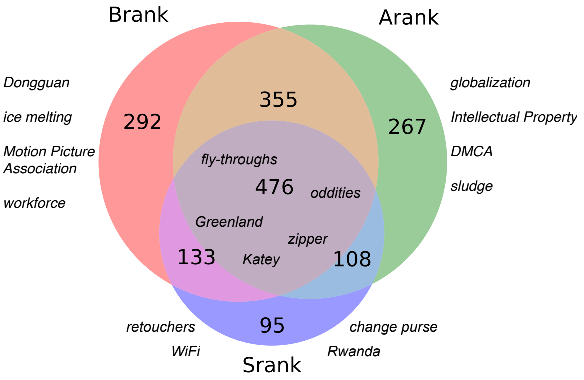


Figure 4: Untranslated term overlap between interpreters.

The untranslated term overlap between interpreters is visualized in Fig. 4. Most difficult terms are shared amongst interpreter ranks as only 23.2% (B), 22.1% (A), and 11.7% (S) of terms are unique for each interpreter. We show a sampling of some unique noun terms on the outside of the Venn diagram, along with the untranslated terms shared among all ranks in the center. Among these unique terms, capitalized terms make up 19% of B-rank/S-rank, but only 13% of A-rank. 7.4% of S-rank's unique terms are numbers compared with about 5% for the other two ranks.

5 Term Prediction Experiments

5.1 Experimental Setting

We design our experiments to evaluate both the effectiveness of a system to predict untranslated terminology in simultaneous interpretation and the usefulness of our features given the small amount

	AP		
Method	**B**	**A**	**S**
Select noun/# POS tag	45.4	43.6	29.6
Optimal freq threshold	49.7	48.1	32.9
SVM (all features)	58.9	53.5	39.1
– elapsed time	58.8	53.0	38.8
– word timing	58.2	53.2	38.5
– word freq	**59.4**	52.5	39.1
– characteristic/syntax	59.3	**55.1**	**42.5**

Table 3: Average precision score cross-validation results with feature ablation for the untranslated term class on test data. Optimal word frequency threshold is determined on dev set of each fold. Evaluation performed on a word-level. Highest numbers per column are bolded. Each setting is statistically significant at $p < 0.05$ by paired bootstrap (Koehn, 2004).

Select POS	in the last 5 years we 've added 70000000 tons of co2 every 24 hours 25000000 tons every day to the oceans
Optimal freq	in the last 5 years we 've added 70000000 tons of co2 every 24 hours 25000000 tons every day to the oceans
SVM	in the last 5 years we 've added 70000000 tons of co2 every 24 hours 25000000 tons every day to the oceans

Table 4: B-rank output from our model contrasted with baselines. Type I errors are in red, type II errors in orange, and correctly tagged untranslated terminology in blue.

of aligned and labeled training data we possess.

We perform leave-one-out cross-validation using five of the seven TED talks as the training set, one as the development set, and one as the test set. Hyperparameters (SVM's penalty term, the number of bins for the word frequency feature=9, and sliding window size=8) are tuned on the dev. fold and the best model, determined by average precision score, is used for the test fold predictions. Both training and predictions are performed on a sentence-level. During training, we weight the two classes inversely proportional to their frequencies in the training data to ensure that the majority O-tag does not dominate the I-tag.

5.2 Results and Analysis

Since we are ultimately interested in the precision and recall trade-off among the methods, we evaluate our results using precision-recall curves in Fig. 5 and the average precision (AP) scores in Table 3. AP[5] summarizes the precision-recall curve by calculating the weighted mean of the precisions at each threshold, where the weights are equal to the increase in recall from the previous threshold. If the method is embedded in a CAI system, then the user could theoretically adjust the precision-recall threshold to balance helpful term suggestions with cognitive load.

Overall, we tend to see that all methods perform best when tested on data from the B-rank interpreter, and observe a decline in performance across all methods with an increase in interpreter experience. We believe that this is due to a decrease in the number of untranslated terminology as experience increases (i.e., class imbalance) coupled with the difficulty of predicting such exclusive word occurrences from only source speech and textual cues. Ablation results in Table 3 show that not all of the features are able to improve classifier performance for all interpreters. While the elapsed time and word timing features tend to cause a degradation in performance when removed, ablating the word frequency and characteristic/syntax features can actually improve average precision score. Word frequency, which is a recall-based feature, seems to be more helpful for B- and S-rank interpreters because it is challenging to recall the smaller number of untranslated terms from the data. Although the characteristic/syntax features are also recall-based, we see a decline in performance for them across all interpreter ranks because they are simply too noisy. When ablating the uninformative features for each rank, the SVM is able to increase AP vs. the optimal word frequency baseline by about 20%, 15%, and 30% for the B, A, and S-rank interpreters, respectively.

In Table 4, we show an example taken from the first test fold with results from each of the three methods. The SVM's increased precision is able to greatly reduce the number of false positives, which we argue could overwhelm the interpreter if left unfiltered and shown on a CAI system. Nevertheless, one of the most apparent false

---

[5] We compute AP using the scikit-learn implementation (Pedregosa et al., 2011).

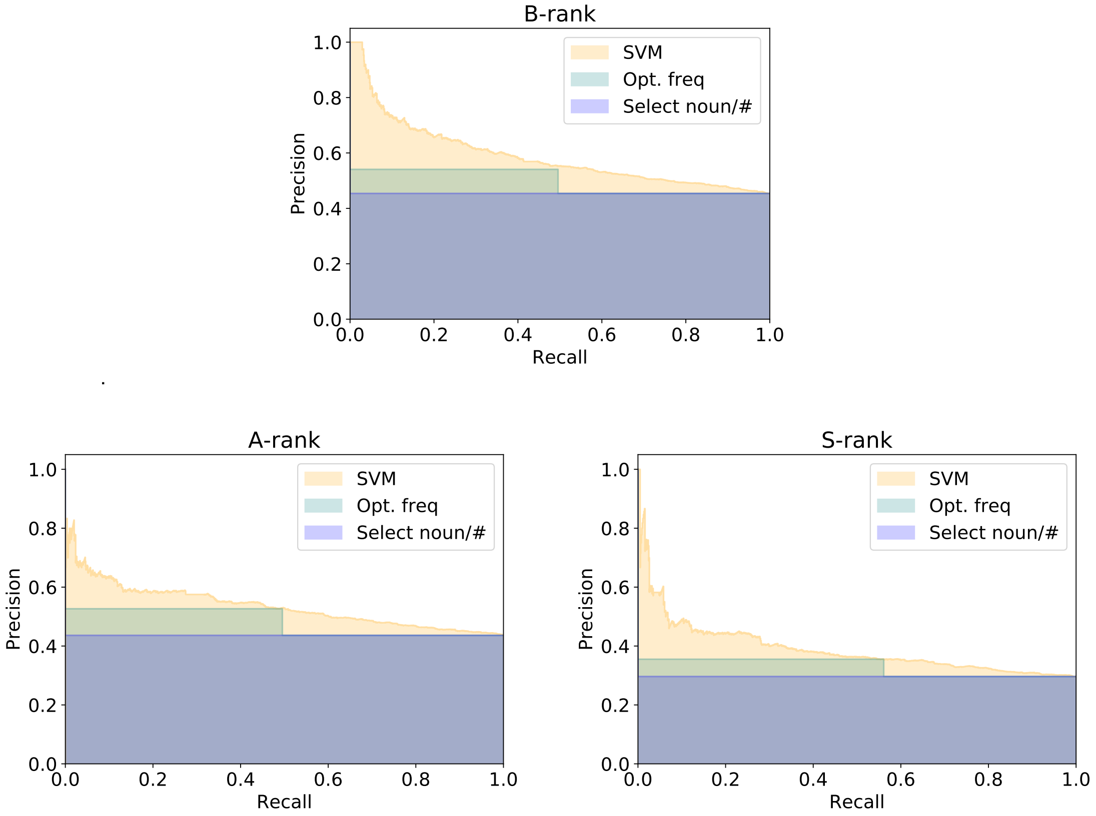


Figure 5: Precision-recall curves for each interpreter rank.

positive errors that still occurs with our method is on units following numbers, such as the word *tons* in the example. Also, because our model prioritizes avoiding this type I error, it is more susceptible to type II errors, such as ignoring untranslated terms *24* and *day*. A user study with our method embedded in a CAI would reveal the true costs of these different errors, but we leave this to future work.

6 Conclusion and Future Work

In this paper, we introduce the task of automatically predicting terminology likely to be left untranslated in simultaneous interpretation, create annotated data from the NAIST ST corpus, and propose a sliding window, SVM-based tagger with task-specific features to perform predictions.

We plan to assess the effectiveness of our approach in the near future by integrating it in a heads-up display CAI system and performing a user study. In this study, we hope to discover the ideal precision and recall tradeoff point regarding cognitive load in CAI terminology assistance and use this feedback to adjust the model.

Other future work could examine the effectiveness of the approach in the opposite direction (Japanese to English) or on other language pairs. Additionally, speech features could be extracted from the source or interpreter audio to reduce the dependence on a strong ASR system.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE1745016 and National Science Foundation EAGER under Grant No. 1748642. We would like to thank Jordan Boyd-Graber, Hal Daumé III and Leah Findlater for helpful discussions, Arnav Kumar for assistance with the term annotation interface, and the anonymous reviewers for their useful feedback.

References

Eijiro electronic dictionary. https://eijiro.jp. Accessed: 2018-12-06.

David A Balota and James I Chumbley. 1985. The locus of word-frequency effects in the pronunciation task: Lexical access and/or production? *Journal of Memory and Language*, 24(1):89–106.

Claudio Bendazzoli and Annalisa Sandrelli. 2005. An approach to corpus-based interpreting studies: developing EPIC (European Parliament Interpreting Corpus). In *Proceedings of the EU-High-Level Scientific Conference Series MuTra 2005—Challenges of Multidimensional Translation*.

Silvia Bernardini, Adriano Ferraresi, and Maja Miličević. 2016. From EPIC to EPTIC — exploring simplification in interpreting and translation from an intermodal perspective. *Target. International Journal of Translation Studies*, 28(1):61–86.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1.

James Breen. 2004. Jmdict: a japanese-multilingual dictionary. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, pages 71–79. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Valeria Darò and Franco Fabbro. 1994. Verbal memory during simultaneous interpretation: Effects of phonological interference. *Applied Linguistics*, 15(4):365–381.

Claudio Fantinuoli. 2016. Interpretbank. redefining computer-assisted interpreting tools. In *Proceedings of the Translating and the Computer 38 Conference in London*, pages 42–52.

Claudio Fantinuoli. 2017a. Computer-assisted interpreting: challenges and future perspectives. *Trends in E-Tools and Resources for Translators and Interpreters*, page 153.

Claudio Fantinuoli. 2017b. Speech recognition in the interpreter workstation. *Translating and the Computer 39*, page 25.

Daniel Gile. 2009. *Basic concepts and models for interpreter and translator training*, volume 8. John Benjamins Publishing.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Zenzi M Griffin and Kathryn Bock. 1998. Constraint, word frequency, and the relationship between lexical processing levels in spoken word production. *Journal of Memory and Language*, 38(3):313–338.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. pages 1342–1352.

W Hewitt, Paula Hannaford, Catherine Gill, and Melissa Cantrell. 1998. Court interpreting services in state and federal courts: Reasons and options for inter-court coordination. *National Center for State Courts*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jörg D Jescheniak and Willem JM Levelt. 1994. Word frequency effects in speech production: Retrieval of syntactic information and of phonological form. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(4):824.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98*.

Roderick Jones. 2002. *Conference interpreting explained*, volume 6. St Jerome Pub.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. pages 388–395.

Sylvie Lambert and Barbara Moser-Mercer. 1994. *Bridging the gap: Empirical research in simultaneous interpretation*, volume 3. John Benjamins Publishing.

Marianne Lederer. 1978. Simultaneous interpretation—units of meaning and other features. In *Language interpretation and communication*, pages 323–332. Springer.

Minhua Liu, Diane L Schallert, and Patrick J Carroll. 2004. Working memory and expertise in simultaneous interpreting. *Interpreting*, 6(1):19–42.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Carmen Millán and Francesca Bartrina. 2012. *The Routledge handbook of translation studies*. Routledge.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 529–533. Association for Computational Linguistics.

Robert Ochshorn and Max Hawkins. 2016. Gentle: A forced aligner.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. pages 551–556.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. pages 198–207.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. pages 311–318.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Yann Plancqueel and Benoît Werner. Interpreters' Help. `https://interpretershelp.com`. Online. Accessed: 2018-08-25.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi speech recognition toolkit. pages 1–4.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a simultaneous translation corpus for comparative analysis. In *LREC*, pages 670–673. Citeseer.

Craig Stewart, Nikolai Vogler, Junjie Hu, Jordan Boyd-Graber, and Graham Neubig. 2018. Automatic Estimation of Simultaneous Interpreter Performance. *arXiv preprint arXiv:1805.04016*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Robert Weide. 1998. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*.

AudioCaps: Generating Captions for Audios in The Wild

Chris Dongjoo Kim **Byeongchang Kim** **Hyunmin Lee** **Gunhee Kim**
Department of Computer Science and Engineering & Center for Superintelligence
Seoul National University, Seoul, Korea
`{cdjkim,byeongchang.kim}@vision.snu.ac.kr` `{lhm1442,gunhee}@snu.ac.kr`

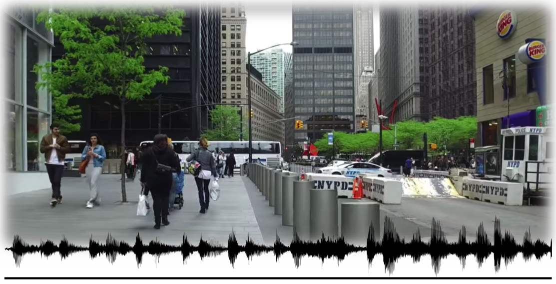
Abstract

We explore the problem of *audio captioning*[1]: generating natural language description for any kind of audio in the wild, which has been surprisingly unexplored in previous research. We contribute a large-scale dataset of 46K audio clips with human-written text pairs collected via crowdsourcing on the AudioSet dataset (Gemmeke et al., 2017). Our thorough empirical studies not only show that our collected captions are indeed loyal to the audio inputs but also discover what forms of audio representation and captioning models are effective for audio captioning. From extensive experiments, we also propose two novel components that are integrable with any attention-based captioning model to help improve audio captioning performance: the top-down multi-scale encoder and aligned semantic attention.

1 Introduction

Captioning, the task of translating a multimedia input source into natural language, has been substantially studied over the past few years. The vast majority of the journey has been through the visual senses ranging from static images to videos. Yet, the exploration into the auditory sense has been circumscribed to human speech transcription (Panayotov et al., 2015; Nagrani et al., 2017), leaving the basic natural form of sound in an uncharted territory of the captioning research.

Recently, sound event detection has gained much attention such as DCASE challenges (Mesaros et al., 2017) along with the release of a large scale *AudioSet* dataset (Gemmeke et al., 2017). However, sound classification (*e.g.* predicting multiple labels for a given sound) and event detection (*e.g.* localizing the sound of interest in a clip) may not be sufficient for a full understanding of the sound. Instead, a natural sentence offers a greater freedom to express a sound, because it allows to characterize objects along with their states, properties, actions and interactions. For example, suppose that suddenly sirens are ringing in the downtown area. As a natural reaction, people may notice the presence of an emergency vehicle, even though they are unable to see any flashing lights nor feel the rush of wind from a passing vehicle. Instead of simply tagging this sound as *ambulance* or *siren*, it is more informative to describe which direction the sound is coming from or whether the source of the sound is moving closer or further away, as shown in Figure 1.



[Audio Classification] rumble | vehicle | speech | car | outside

[Video Captioning] A bus passing by with some people walking by in the afternoon.

[Audio Captioning] A muffled rumble with man and woman talking in the background while a siren blares in the distance.

Figure 1: Comparison of audio captioning with audio classification and video captioning tasks.

To that end, we address the *audio captioning* problem for audios in the wild, which has not been studied yet, to the best of our knowledge. This work focuses on one of the most important bases toward this research direction, *contributing a large-scale dataset*. The overarching sources of in-the-wild sounds are grounded on the *AudioSet* (Gemmeke et al., 2017), so far the largest collection of sound events collected from Youtube

---

[1] For a live demo and details, https://audiocaps.github.io

Proceedings of NAACL-HLT 2019, pages 119–132
Minneapolis, Minnesota, June 2 - June 7, 2019.

videos. We newly collect human-written sentences for a subset of AudioSet audio clips via crowdsourcing on Amazon Mechanical Turk (section 3). We also develop two simple yet effective techniques to generate captions through the joint use of multi-level pretrained features and better attention mechanism named aligned-semantic attention (section 4). Lastly, we perform experiments contrasting between video-based captions and audio-focused captions by employing a variety of features and captioning models (section 5).

The contributions of this work are as follows.

1. To the best of our knowledge, this work is the first attempt to address the audio captioning task for sound in the wild. We contribute its first large-scale dataset named *AudioCaps*, which consists of 46K pairs of audio clips and text description.

2. We perform thorough empirical studies not only to show that our collected captions are indeed true to the audio inputs and but also to discover what forms of audio representations and captioning models are effective. For example, we observe that the embeddings from large-scale pretrained VGGish (Hershey et al., 2017) are powerful in describing the audio input, and both temporal and semantic attention are helpful to enhance captioning performance.

3. From extensive experiments, we propose two simple yet effective technical components that further improve audio captioning performance: the *top-down multi-scale encoder* that enables the joint use of multi-level features and *aligned semantic attention* that advances the consistency between semantic attention and spatial/temporal attention.

2 Related Work

Speech recognition and separation. One of the most eminent tasks for audio understanding may be speech recognition, the task of recognizing and translating human spoken language into text with less emphasis on background sound that may coexist. A multitude of datasets exist for such task *e.g.* Speech Commands dataset (Warden, 2018), Common Voice dataset (Mozilla, 2017), Librispeech (Panayotov et al., 2015), LS Speech (Ito, 2017). As one of similar lineage, automatic speech separation forks an input audio signal into several individual speech sources (Hershey et al., 2016; Ephrat et al., 2018). To most of these tasks, in the wild sound is deemed as background noise to be removed as an obstructer of speech recognition. On the other hand, our work puts the spotlight on these neglected sounds and express them through natural language.

Audio classification and sound event detection. This line of tasks emphasizes categorizing a sound into a set of predefined classes. There exist a number of datasets to aid in achieving this goal, including DCASE series (Stowell et al., 2015; Mesaros et al., 2016, 2017), UrbanSound8k (Salamon et al., 2014), ESC (Piczak, 2015). AudioSet (Gemmeke et al., 2017) is an audio event dataset collected from Youtube that is unsurpassed in terms of coverage and size, structured with an ontology containing 527 classes. Another predominant large-scale dataset is Freesound (Fonseca et al., 2017). It consists of audio samples from *freesound.org* recordings based on the preceding AudioSet ontology. In contrast to audio classification, which uniquely map the audio to a set of labels, our task generates a descriptive sentence. Hence, it needs to not only detect salient sounds of classes but also explores their states, properties, actions or interactions.

Captioning tasks and datasets. The vast majority of captioning tasks and datasets focus on the visual domain. Image captioning generates text description of an image, and numerous datasets are proposed, such as Flickr 8k (Rashtchian et al., 2010), Flickr 30k (Young et al., 2014), MS COCO (Lin et al., 2014), DenseCap (Johnson et al., 2016) and Conceptual Captions (Sharma et al., 2018). Akin to the image captioning is video captioning, for which there are many datasets too, including MSVD (Guadarrama et al., 2013), MSR-VTT (Xu et al., 2016), LSMDC (Rohrbach et al., 2017) and ActivityNet Captions (Krishna et al., 2017).Compared to previous captioning tasks and datasets, our work confines the problem by focusing on in the wild audio inputs.

Recently, there have been some efforts to solve video captioning with audio input (Hori et al., 2017, 2018; Wang et al., 2018). However, the audio input merely serves as auxiliary features for video captioning, and as a result, it only marginally improves the performance (*e.g.* BLEU-4 score: 39.6 (video only) vs. 40.3 (video + MFCC) (Wang et al., 2018)). These results are partly culpable to dataset collection, where the annotators mostly rely on the video input. On the contrary, our collection induces the annotators to mainly abide to audio, hence, increasing the dependency of written text on the audio input as can be shown in our survey analysis in Figure 5.

3 The Audio Captioning Dataset

Our *AudioCaps* dataset entails 46K audio caption pairs. Table 1 outlines its key statistics. The audio sources are rooted in *AudioSet* (Gemmeke et al., 2017), a large-scale audio event dataset, from which we draft the *AudioCaps*, as discussed below. We present more details of data collection and statistics in the Appendix.

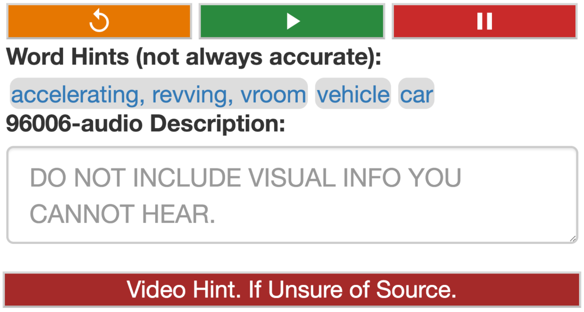
3.1 AudioSet Tailoring

It is important to select qualified audio clips as the first step of dataset collection. The chosen categories of clips must be well-rounded in coverage of naturally occurring audios, be relevant to practical applications and appear with high frequency. To that end, we tailor the AudioSet dataset (Gemmeke et al., 2017) that comprises 1,789,621 human-labeled 10 second YouTube excerpts with an ontology of 527 audio event categories. However, an immediate collection of captions from these audios pose several difficulties: (i) too many audio clips, (ii) inconsistent level of abstraction among the classes, (iii) distribution bias of some labels and (iv) noisy labels that are only noticeable from visual cues. We circumvent these issues through a controlled sampling process as described below.

Among 527 audio event categories of AudioSet, we first exclude all the labels whose number of clips are less than 1,000 to promote a balanced distribution within the dataset. We also remove all 151 labels in the *music* super-category, because they are often indiscernible even for a human. For example, a human with no expertise can hardly discriminate the sound of *Guitar* from *Banjo*. Thus, we set aside the musical territory for future exploration. We further discard categories if they do not satisfy the following two constraints. The word labels should be identifiable solely from sound (i) without requiring visuals (*e.g.* remove the category *inside small room*) and (ii) without requiring any expertise (*e.g.* remove *power windows* and *electric windows* because their distinction may be possible only for car experts). Finally, we select 75 word labels derived from 7 augmented super-categories as avoiding the sharp skewness in the word labels (*e.g.* 48.5% clips include *speech* label). We limit the number of instances per category to 2,000 by sampling with preference to audio clips associated with more word labels to prioritize the audios with diverse content. The final number of audio clips is about 115K, from which we obtain captions for 46K as the first version.

Split	# clips	# captions	# words/caption	# labels/clip
Train	38,118	38,118	8.79 (8)	4.25 (4)
Val	500	2,500	10.12 (9)	4.06 (3)
Test	979	4,895	10.43 (9)	4.03 (3)
Total	39,597	45,513	9.03 (9)	4.22 (4)

Table 1: Some statistics of *AudioCaps* dataset. We also show average and median (in parentheses) values. labels refer to the semantic attributes.



Word Hints (not always accurate):

accelerating, revving, vroom vehicle car

96006-audio Description:

DO NOT INCLUDE VISUAL INFO YOU CANNOT HEAR.

Video Hint. If Unsure of Source.

Figure 2: The AMT interface for audio annotation.

3.2 Audio Annotation

The collected captions should be precise, specific, diverse, expressive, large-scale and correlated with the paired audios with minimal visual presumptions. Such complex nature of our requirements necessitates employing crowdworkers through Amazon Mechanical Turk (AMT). Some qualification measures are set for the crowdworkers, such as they should hold a +95% HIT approval rate and the total number of approved HITs that are greater than 1,000 and be located at one of [AU, CA, GB, NZ, US]. In total, 108 caption writing workers and 3 caption reviewing workers participate and are compensated at 10 cents per clip.

Annotation Interface. Figure 2 shows our annotation interface, which is designed to minimize the visual presumption while maintaining diversity. Each task page consists of an audio clip of about 10 seconds, word hints and video hints.

The *word hints* are the word labels that are provided by AudioSet for the clip and are employed

Below officers **creep toward** the entrance the door and **points** a gun

(a) LSMDC

A **black** and **white** video of about actors

(b) MSR-VTT

A train is approaching with a low **rumble** and **rhythmic click** and **squeal**

(c) AudioCaps

Figure 3: Comparison between two video captioning datasets and AudioCaps. The text from (a) LSMDC (Rohrbach et al., 2017) and (b) MSR-VTT (Xu et al., 2016) includes multiple visually grounded vocabularies (indicated in blue), whereas the text from (c) AudioCaps contains vocabularies relying on auditory cues (in red).

as hints to the crowdworkers. Even to humans, recognizing the true identity of a sound can be ambiguous, and thus the word hints act as a precursor to *accurately* guide the crowdworkers during the description process, while staying aloof from visual bias. Another benefit is that the diversity of the word labels may also enrich the expressiveness of the description. Also derived from AudioSet, the *video hints* are provided as a stronger hint for sounds that are too difficult even to the human ear or for clips associated with some erroneous or missing word hints (weak labels). We advise the workers to use them as a last resort measure.

Some instructions[2] are also provided to demarcate crowdworkers' descriptions as follows. (i) Do not include the words for visuals in the video that are not present in the sound. (ii) Ignore speech semantics. (iii) When applicable, be detailed and expressive. (iv) Do not be imaginative and be literal and present with the descriptions.

Quality Control. We use a qualification test to discern many crowdworkers who frequently violate the given instructions (*e.g.* transcribing instead of describing, just enumerating provided word hints or writing visual captions). Interested crowdworkers must participate in the test and submit a response, which the authors manually check and approve if they are eligible. We employ three additional workers to verify the data in accordance to our guidelines. In order to maintain high approval rates, we periodically blacklist malicious crowdworkers while granting reasonable incentives to benevolent workers.

3.3 Post-processing

We exclude the period symbol from all the captions, convert numbers to words using `num2words`[3] and correct grammar errors by `languagetool`[4]. We then tokenize words with `spacy` [5]. Finally, we build a dictionary $\mathcal{V}$ with a size of 4506 by choosing all the unique tokens.

3.4 Comparison with Other Datasets

Figure 3 qualitatively compares some caption examples between our *AudioCaps* and two captioning datasets with audio: LSMDC (Rohrbach et al., 2017) and MSR-VTT (Xu et al., 2016). Since both LSMDC and MSR-VTT focus more on describing videos than audios, their captions are characterized by visually grounded vocabularies (blue). On the other hand, the captions of *AudioCaps* accompany sound-based vocabularies (red).

4 Approach

We present a hierarchical captioning model that can attend to the fine details of the audio. The backbone of our model is an LSTM (Hochreiter and Schmidhuber, 1997) that we fortify with two novel components which are easily integrable with any attention-based captioning model. The *top-down multi-scale encoder* enables the contextual use of multi-level features, and the *aligned semantic attention* enhances the consistency between semantic attention and temporal attention (see Figure 4). Our experiments in section 5.3 show that these two techniques lead to non-trivial performance improvement.

---

[2] `https://audiocaps.github.io/instruction_only.html`.

[3] `https://github.com/savoirfairelinux/num2words`.

[4] `https://github.com/languagetool-org/languagetool`.

[5] `https://spacy.io`.

The input to our model are mel-frequency cepstral coefficient (MFCC) audio features (Davis and Mermelstein, 1980) and the output is a sequence of words $\{y_m\}_{m=1}^{M}$, each of which is a symbol from the dictionary. For text representation, we use `fastText` (Bojanowski et al., 2016) trained on the Common Crawl corpus to initialize the word embedding matrix $\mathbf{W}_{emb}$, which is fine-tuned with the model during training. We represent word sequences (*e.g.* attribute words for semantic attention and output words for answer captions) in a distributional space as $\{\mathbf{d}_n\}_{n=1}^{N}$ with $\mathbf{d}_n = \mathbf{W}_{emb}\mathbf{w}_n$ where $\mathbf{w}_n$ is a one-hot vector for n-th word in the word sequence and $\mathbf{d}_n \in \mathbb{R}^{300}$.

4.1 Top-down Multi-scale Encoder

Unlike speech data, sound in the wild is not always continuous. It can be often brief, noisy, occluded, in-the-distance and randomly sparsed throughout the audio. Hence, the lower-level features can be useful to capture such characteristics of natural sound, although they may lack the semantics of the higher-level features. Thus, the joint use of these two levels of features can be mutually beneficial.

The top-down multi-scale encoder takes as input the two-level audio embedding $\{\mathbf{f}_t\}_{t=1}^{T}$, $\{\mathbf{c}_t\}_{t=1}^{T}$ and generates the fused encoding vector, where T is the sequence length of the audio. For input, we use the features from the two layers of the pretrained VGGish network (Hershey et al., 2017): the `fc2` vector $\{\mathbf{f}_t\}_{t=1}^{T}$ as a high-level semantic feature, and the `conv4` vector $\{\mathbf{c}_t\}_{t=1}^{T}$ as a mid-level feature.

The first level of hierarchy encodes high-level features $\{\mathbf{f}_t\}_{t=1}^{T}$ using a bi-directional LSTM. We regard the last hidden state as the global audio embedding $\mathbf{h}^{ctxt} \in \mathbb{R}^{I}$:

$$\overleftrightarrow{\mathbf{h}}^{a1}_{t} = \text{biLSTM}(\mathbf{f}_t, \overrightarrow{\mathbf{h}}^{a1}_{t-1}, \overleftarrow{\mathbf{h}}^{a1}_{t+1}), \qquad (1)$$

$$\mathbf{h}^{ctxt} = \mathbf{W}_c[\overrightarrow{\mathbf{h}}^{a1}_{T}; \overleftarrow{\mathbf{h}}^{a1}_{1}] + \mathbf{b}_c, \qquad (2)$$

where $\mathbf{W}_c \in \mathbb{R}^{I \times D^1}$ and $\mathbf{b}_c \in \mathbb{R}^{I}$ are parameters, I is the dimension of input to the next layer and D^1 is the dimension of the first layer hidden states.

We then reshape and encode mid-level features $\{\mathbf{c}_t\}_{t=1}^{T} \in \mathbb{R}^{512}$ using another bi-directional LSTM. In order to inject the global semantics, we perform an element-wise addition of $\mathbf{h}^{ctxt}$ to the mid-level feature along the time axis, and feed them into the bi-directional LSTM one at a time, producing a hidden state $\overleftrightarrow{\mathbf{h}}^{a2}_{t} \in \mathbb{R}^{D^2}$ at each step:

$$\overleftrightarrow{\mathbf{h}}^{a2}_{t} = \text{biLSTM}(\mathbf{c}_t + \mathbf{h}^{ctxt}, \overrightarrow{\mathbf{h}}^{a2}_{t-1}, \overleftarrow{\mathbf{h}}^{a2}_{t+1}). \qquad (3)$$

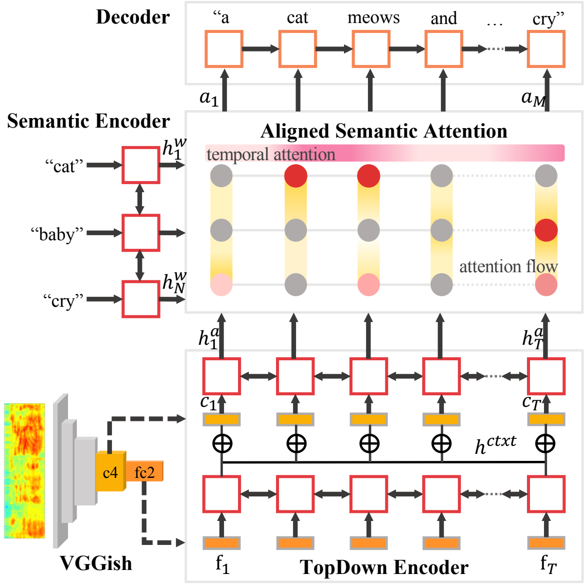


Figure 4: The audio captioning model with *top-down multi-scale encoder* and *aligned semantic attention*.

4.2 Aligned Semantic Attention

In many captioning models (You et al., 2016; Yu et al., 2017; Laokulrat et al., 2018; Long et al., 2018), semantic attention has been independently used from temporal/spatial attention. However, it can be troublesome because there may exist some discrepancies between the two attentions *i.e.* they do not attend to the same part of the input. For instance, given an audio of a cat meowing and a baby crying, temporal attention may attend to the *crying baby* while semantic attention attends to the word *cat*. We propose a simple yet effective approach that implicitly forces both semantic and temporal/spatial attention to be correctly aligned to one another to maximize the mutual consistency.

For semantic attention, we extract a set of N attribute words for each audio: following You et al. (2016), we retrieve the nearest training audio from the subset of AudioSet and transfer its labels as attribute words. We encode each attribute word vector using a bi-directional LSTM (named *semantic encoder*):

$$\overleftrightarrow{\mathbf{h}}^{w}_{n} = \text{biLSTM}(\mathbf{d}_n, \overrightarrow{\mathbf{h}}^{w}_{n-1}, \overleftarrow{\mathbf{h}}^{w}_{n+1}), \qquad (4)$$

where $\mathbf{d}_n$ is the input text representation of the attribute word sequence. We then align these semantic word features $\overleftrightarrow{\mathbf{h}}^{w}_{n}$ to the temporal axis of the audio features $\overleftrightarrow{\mathbf{h}}_t^{a2}$ via the attention flow layer (Seo et al., 2017). For notational simplicity, we omit the bidirectional arrow in the following.

Attention flow layer. We first compute the similarity matrix, $\mathbf{S} \in \mathbb{R}^{T \times N}$ between each pair of audio and word features using the score function $\alpha(\mathbf{h}_t^{a2}, \mathbf{h}_n^w) \in \mathbb{R}$:

$$\alpha(\mathbf{h}_t^{a2}, \mathbf{h}_n^w) = \mathbf{W}_\alpha[\mathbf{h}_t^{a2}; \mathbf{h}_n^w; \mathbf{h}_t^{a2} \circ \mathbf{h}_n^w], \quad (5)$$
$$\mathbf{S}_{tn} = \alpha(\mathbf{h}_t^{a2}, \mathbf{h}_n^w), \quad (6)$$

where $\circ$ is element-wise multiplication.

We then use $\mathbf{S}$ to obtain the attentions and the attended vectors in two directions: word-to-audio $\{\tilde{\mathbf{h}}_t^w\}_{t=1}^T \in \mathbb{R}^{D^2}$ and audio-to-word $\tilde{\mathbf{h}}^{a2} \in \mathbb{R}^{D^2}$:

$$\mathbf{a}_t = \text{softmax}(\mathbf{S}_{t:}), \qquad \tilde{\mathbf{h}}_t^w = \sum_n \mathbf{a}_{tn}\mathbf{h}_n^w, \quad (7)$$
$$\mathbf{b} = \text{softmax}(\max_{row}(\mathbf{S})), \quad \tilde{\mathbf{h}}^{a2} = \sum_t \mathbf{b}_t \mathbf{h}_t^{a2}, \quad (8)$$

where $\mathbf{a}_t \in \mathbb{R}^N$, $\mathbf{b} \in \mathbb{R}^T$.

Lastly, we concatenate them into $\{\mathbf{h}_t^{flow}\}_{t=1}^T \in \mathbb{R}^{4D^2}$, while keeping the temporal axis intact:

$$\mathbf{h}_t^{flow} = [\mathbf{h}_t^{a2}; \tilde{\mathbf{h}}_t^w; \mathbf{h}_t^{a2} \circ \tilde{\mathbf{h}}_t^w; \mathbf{h}_t^{a2} \circ \tilde{\mathbf{h}}^{a2}]. \quad (9)$$

Temporal attention over attention flow. We now have an embedding that aligns the semantic features of words with the time steps of audio features. Subsequently, we apply temporal attention over it; the attention weight is calculated as in Luong et al. (2015). Specifically, we use the global method for each t in $\{\mathbf{h}_t^{flow}\}_{t=1}^T$:

$$\alpha_m = \text{align}(\mathbf{h}_m^{dec}, \mathbf{h}_t^{flow}), \quad (10)$$
$$\mathbf{c}_m = \sum_t \alpha_{mt}\mathbf{h}_t^{flow}, \quad (11)$$
$$\mathbf{a}_m = \tanh(\mathbf{W}_{dec}[\mathbf{c}_m; \mathbf{h}_m^{dec}]), \quad (12)$$

where $\mathbf{h}_m^{dec} \in \mathbb{R}^{D^o}$ is the state of the decoder LSTM, $\mathbf{c}_m \in \mathbb{R}^{4D^2}$ is the context vector, $\alpha_m \in \mathbb{R}^T$ is the attention mask, and $\mathbf{W}_{dec} \in \mathbb{R}^{D^o \times (4D^2 + D^o)}$ is a parameter.

Next, we obtain the output word probability:

$$\mathbf{s}_m = \text{softmax}(\mathbf{W}_o \mathbf{a}_m) \quad (13)$$

where $\mathbf{W}_o \in \mathbb{R}^{V \times D^o}$. Finally, we select the output word as $y_{m+1} = \text{argmax}_{s \in \mathcal{V}}(\mathbf{s}_m)$. We repeat this process until y_{m+1} reaches an EOS token.

The model is trained to maximize the log-likelihood assigned to the target labels via the softmax as done in most captioning models.

5 Evaluation

We perform several quantitative evaluations to provide more insights about our *AudioCaps* dataset. Specifically, our experiments are designed to answer the following questions:

1. Are the collected captions indeed faithful to the audio inputs?
2. Which audio features are useful for audio captioning on our dataset?
3. What techniques can improve the performance of audio captioning?

We present further implementation details and more experimental results in the Appendix. Some resulting audio-caption pairs can be found at https://audiocaps.github.io/supp.

Before presenting the results of our experiments on these three questions, we first explain the experimental setting and baseline models.

5.1 Experimental Setting

Evaluation metrics. Audio captioning can be quantitatively evaluated by the language similarity between the predicted sentences and the ground-truths (GTs) such as BLEU (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004) and SPICE (Anderson et al., 2016). In all metrics, higher scores indicate better performance.

Audio features. Audios are resampled to 16kHz, and stereo is converted into mono by averaging both channels. We zero-pad clips that are shorter than 10 seconds and extract three levels of audio features. For the low-level audio feature, the lengthy raw audios are average-pooled by the WaveNet encoder as in Engel et al. (2017). For the mid-level feature, mel-frequency cepstral coefficients (MFCC) (Davis and Mermelstein, 1980) are extracted using `librosa` (McFee et al., 2015) with a window size of 1024, an overlap of 360 and the number of frames at 240, and encoded further with a bi-directional LSTM followed by a gated convolutional encoder (Xu et al., 2018). Lastly, we use two high-level features: the 24th output layer of SoundNet[6] (Aytar et al., 2016) with a (10×1024) dimension and the final output embedding of VGGish[7] (Hershey et al., 2017) with a (10×128) dimension of (time × embedding).

---

[6] `https://github.com/cvondrick/soundnet`.

[7] `https://github.com/tensorflow/models/tree/master/research/audioset`.

Video features. To contrast with video captioning datasets, we also extract video features at the frame-level and at the sequence-level from YouTube clips. For frame features, we use VGG16 (Simonyan and Zisserman, 2015) pretrained on the ILSVRC-2014 dataset (Russakovsky et al., 2015). For sequence features, we use C3D[8] (Tran et al., 2015) pretrained on the Sport1M dataset (Karpathy et al., 2014). We extract subsequent frames with 50% overlap centered at each time step on the input clips for AudioSet videos, while proceeding with no overlap for MSR-VTT clips as in the original paper. We sample videos at 25fps.

5.2 Baselines

Retrieval methods. As straightforward baselines, we test the 1-nearest search with audio features, denoted by `1NN-MFCC`, `1NN-SoundNet` and `1NN-VGGish`. For a query audio, we find its closest training audio using the ℓ_2 distance on the features and return its text as a prediction. We mean-pool all the audio features over time, because it empirically leads to a strong performance.

LSTM methods. As simple generative baselines, we test with the LSTM decoder, denoted by `-LSTM` postfix, where the encoded audio feature is set as the initial state of the LSTM. For instance, `WaveNet-LSTM` is the model with the WaveNet encoder and the LSTM decoder. We use a single-layer LSTM with dropout (Srivastava et al., 2014) and layer normalization (Ba et al., 2016).

Attention models. We test two popular attention models developed in video captioning research: (i) `TempAtt` (Luong et al., 2015; Yao et al., 2016) generates captions by selectively attending to audio features over time, and (ii) `SemAtt` (You et al., 2016) creates text attending to attribute words as secondary information.

Our models. We denote our top-down multi-scale encoder as the prefix `TopDown-` and aligned semantic attention as `AlignedAtt-`.

Upper-bounds. Given that each test data has five human-generated captions, we perform cross validation on the five GT captions as an upper-bound of performance denoted as `Human`. We regard one of five human annotations as model prediction and compute the performance metric with the other four as ground-truths. After doing this on each of five, we then average the scores.

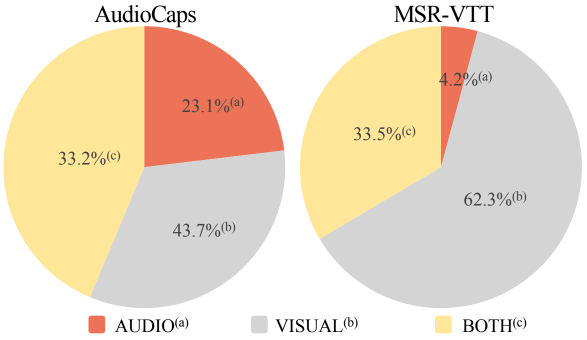


Figure 5: Comparison of vocabulary tag distribution between AudioCaps and MSR-VTT.

5.3 Results

We discuss experimental results in response to the three questions regarding the AudioCaps dataset.

5.3.1 Audio *vs* Video Captioning

We first evaluate whether the collected audio-based captions are indeed loyal to the audio clips. As one possible method to validate it, we perform comparative experiments with the video-oriented MSR-VTT dataset (Xu et al., 2016). Note that MSR-VTT and AudioCaps both provide pairs of audio clips and its corresponding videos, allowing us to perform this comparative study. We hypothesize that the captions from MSR-VTT would not coherently map to audio features, because they are written mainly based on the visual information. In contrast, AudioCaps captions would be better aligned to audio features than visual features.

The results in Table 4 support our hypothesis. In MSR-VTT, the video-based captioning model `C3D-LSTM` attains better scores than the preceding three audio-captioning models `⋆-LSTM`, while in AudioCaps the video-based model performs far worse than the audio models. This may be due to our collection method of AudioCaps, which encourages turkers to submit the descriptions based on the audio rather than the visual.

Vocabulary comparison. We also make comparisons between AudioCaps and MSR-VTT in terms of vocabulary usage in the captions. We select the 1,800 most frequent vocabularies of verbs, adjectives and adverbs from each dataset, and run a user study in which three different workers are asked to categorize each sampled word into one of (*Audio, Visual, Both, Not Applicable*). The category label per word is decided by a majority vote of three workers' opinions. We use AMT once more to collect the unbiased opinions. In or-

---

[8] https://github.com/facebook/C3D.

Methods	B-1	B-2	B-3	B-4	METEOR	CIDEr	ROUGE-L	SPICE
`1NN-MFCC`	34.1	17.8	10.0	5.3	9.9	8.7	23.4	4.7
`1NN-SoundNet` (Aytar et al., 2016)	39.1	22.0	12.9	7.6	12.0	16.4	27.2	6.9
`1NN-VGGish` (Hershey et al., 2017)	44.2	26.5	15.8	9.0	15.1	25.2	31.2	9.2
`WaveNet-LSTM` (Engel et al., 2017)	48.9	31.5	20.2	13.0	13.8	29.6	35.5	9.0
`MFCC-LSTM` (Xu et al., 2018)	57.3	40.0	26.8	16.4	18.4	44.8	41.1	11.5
`SoundNet-LSTM` (Aytar et al., 2016)	54.0	38.0	26.4	17.6	16.5	43.2	39.2	10.8
`VGGish-LSTM` (Hershey et al., 2017)	58.7	42.3	29.8	20.4	18.7	50.4	42.6	13.0
`TempAtt-WaveNet-LSTM` (Luong et al., 2015)	50.7	34.3	22.9	14.8	14.8	28.2	36.4	8.6
`TempAtt-MFCC-LSTM` (Luong et al., 2015)	57.7	40.7	27.6	17.9	18.2	49.3	41.8	12.4
`TempAtt-SoundNet-LSTM` (Luong et al., 2015)	55.5	37.4	24.8	15.8	17.0	43.4	40.0	11.6
`TempAtt-VGGish(FC2)-LSTM` (Luong et al., 2015)	61.3	43.2	29.6	19.5	19.3	50.9	43.5	13.5
`TempAtt-VGGish(C4)-LSTM` (Luong et al., 2015)	61.8	44.5	30.7	20.4	19.4	55.3	44.0	13.2
`TempAtt-VGGish(C3)-LSTM` (Luong et al., 2015)	61.2	44.1	30.3	20.9	19.0	52.3	43.7	13.0
`TopDown-VGGish(FC2,C4)-LSTM`	62.9	45.1	31.5	21.4	19.9	57.7	44.8	14.3
`TopDown-VGGish(FC2,C4,C3)-LSTM`	60.9	43.7	30.7	20.8	20.0	55.8	43.7	13.6
`TopDown-SemTempAtt(1NN)` (You et al., 2016)	62.2	44.9	31.3	20.9	20.2	58.1	44.9	13.6
`TopDown-AlignedAtt(1NN)`	61.4	44.6	**31.7**	**21.9**	**20.3**	**59.3**	**45.0**	**14.4**
`Human`	65.4	48.9	37.3	29.1	28.8	91.3	49.6	21.6

Table 2: Captioning results of different methods on *AudioCaps* measured by language similarity metrics.

Methods	B-1	B-2	B-3	B-4	METEOR	CIDEr	ROUGE-L	SPICE
`SemTempAtt(1NN)-VGGish-LSTM` (You et al., 2016)	62.2	44.5	31.0	20.5	19.3	52.5	44.0	13.7
`AlignedAtt(1NN)-VGGish-LSTM`	62.0	45.1	32.0	21.6	19.6	56.1	44.4	13.5
`SemTempAtt(GT)-VGGish-LSTM` (You et al., 2016)	67.0	50.3	36.4	24.8	22.5	72.0	48.3	16.3
`AlignedAtt(GT)-VGGish-LSTM`	69.1	52.3	38.0	26.1	23.6	77.7	49.6	17.2

Table 3: Upper-bound of *aligned semantic attention* by language similarity metrics.

	MSR-VTT		**AudioCaps**	
Methods	METEOR	CIDEr	METEOR	CIDEr
`MFCC-LSTM`	21.4	19.2	18.2	49.3
`SoundNet-LSTM`	20.0	14.7	17.0	43.4
`VGGish-LSTM`	22.8	26.1	19.3	50.9
`C3D-LSTM`	24.8	36.8	15.9	42.7
Gap (Audio - Video)	-2.0	-10.7	**+3.4**	**+8.2**

Table 4: Comparison of captioning results between video-based and audio-based datasets. The first three methods perform captioning using only audios while the last method `C3D-LSTM`, only use videos. The gaps empirically show how much *AudioCaps* is audio-oriented in contrast to MSR-VTT.

der to guarantee thoughtful submissions, we ask the workers to provide a description using the word. We compensate $0.05 per word to English-speaking workers with a 95% approval rate.

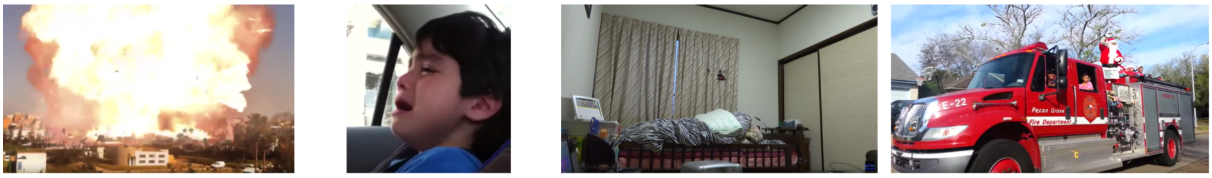
Figure 5 shows that AudioCaps has more vocabularies tagged as *Audio* (*e.g. neighs, rustling*) by 18.9%p more than MSR-VTT. Furthermore, 56.3% of the total vocabularies in AudioCaps are categorized as audio-related, that is, labeled as *Audio* or *Both* (*e.g. vibrating, applauds*). Hence, this vocabulary comparison result reassures that AudioCaps is more audio-oriented than MSR-VTT.

5.3.2 Comparison of Audio Features

The methods in the second group of Table 2 are compared to investigate which audio features are more suitable for captioning on AudioCaps. The best results are obtained by `VGGish-LSTM`. This may be because VGGish is pretrained on YouTube audio clips, similar to AudioCaps. Although the topics of YouTube are extremely diverse, the domain proximity may help VGGish learn more utilizable features for AudioCaps. `SoundNet-LSTM` shows inferior performance compared to `VGGish-LSTM`, one possible reason being because it is pretrained with Flickr videos, which are rather distant in domain from the source of our dataset, in terms of topic diversity and the amount of possible noise. `MFCC-LSTM` does not perform as well as `VGGish-LSTM`, even with the similar convolutional recurrent encoder. This result hints that pretraining with a proper dataset is essential for audio captioning. A comparison between `MFCC-LSTM` and `WaveNet-LSTM` reveals that using MFCC is better than directly taking raw waveform as input. The raw waveform is relatively long (>500× longer than MFCC); hence, it may pose a difficulty for RNN-based encoders to precisely represent the whole audio context.

5.3.3 Comparison of Models

Temporal attention consistently boosts the captioning performance of the LSTM decoder in all audio features, as shown in the models with `TempAtt-` prefix in Table 2. No-



(Ours) a large explosion followed by a loud pop
(1NN-VGGish) a man speaking followed by a loud bang
(SemTempAtt) a large explosion followed by a loud splash and thunder
(GT) a whooshing noise followed by an explosion

(Ours) a man and woman talking, then a baby crying
(1NN-VGGish) an engine runs, and people speak
(SemTempAtt) a dog barks and a woman laughs
(GT) a child cries followed by a man and woman speaking and then the child resumes sobbing

(Ours) a small motor is running, whirring occurs, and a high-pitched whine is present
(1NN-VGGish) a drill is operated, then a man speaks and restarts the drill
(SemTempAtt) a small motor running and a man speaking
(GT) a drone whirring followed by a crashing sound

(Ours) a truck engine is running, a siren is occurring, and an adult male speaks
(1NN-VGGish) a loud fire engine is followed by an emergency siren
(SemTempAtt) emergency sirens and a siren blaring
(GT) a large engine passes as people speak followed by a siren

Figure 6: Four examples of audio captioning with captured video frames, grouthtruths (GT), and generated captions by our method (Ours) and baselines. They can be heard at https://audiocaps.github.io/supp.

tably, a large performance gain is observed for `TempAtt-MFCC-LSTM`. This may be because MFCC features are transformed to temporally longer features than SoundNet and VGGish features ($240 > 10$), and thus allow temporal attention to better aid the model and bypass the vanishing gradient problem.

The semantic attention is also favorable for captioning performance, as `SemTempAtt(1NN)-VGGish-LSTM` in Table 3 slightly outperforms `TempAtt-VGGish(FC2)-LSTM` in Table 2. That is, the additional use of semantic attention enhances the temporal attention model. Obviously, when using GT labels instead of 1NN retrieved labels as attribute words, the performance increases much, hinting that better semantic attributes are more synergetic with the aligned attention.

The comparison between different layers (C4, C3, FC2) confirms the effectiveness of jointly using multi-level features. The fused features by the top-down multi-scale encoder (*i.e.* `TopDown-`) prove the most beneficial as they outperform their counterparts in Table 2. However, a stack of (FC2,C4) layers performs the best, while the three layer stack is slightly inferior, presumably due to overfitting and weak information flow between the upper and lower levels of the stacks. Finally, our best performing model is `TopDown-AlignedAtt` where both the top-down multi-scale encoder and aligned semantic attention are jointly used. We postulate that the two techniques synergize well thanks to rich information provided by `TopDown` allowing for better attention alignment.

5.3.4 Captioning Examples

Figure 6 shows selected examples of audio captioning. In each set, we show a video frame, GT and text descriptions generated by our method and baselines. Many audio clips consist of sounds with multiple sources in sequence, for which baselines often omit some details or mistakenly order the event sequence, whereas our model is better at capturing the details in the correct order.

6 Conclusion

We addressed a new problem of audio captioning for sound in the wild. Via Amazon Mechanical Turk, we contributed a large-scale dataset named *AudioCaps*, consisting of 46K pairs of audio clips and human-written text. In our experiments, we showed that the collected captions were indeed faithful to the audio inputs as well as improve the captions by two newly proposed components: the top-down multi-scale encoder and aligned semantic attention.

There are several possible directions beyond this work. First, we can further expand the scope of *AudioCaps*. Second, our model is integrable with speech counterparts to achieve more complete auditory captioning tasks.

Acknowledgments

We would like to thank SNU Vision & Learning Lab members and Yunseok Jang for the helpful comments and discussions. This work is supported by Kakao and Kakao Brain corporations and the international cooperation program by the NRF of Korea (NRF-2018K2A9A2A11080927). Gunhee Kim is the corresponding author.

References

Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. 2018. The Conversation: Deep Audio-Visual Speech Enhancement. In *Interspeech*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic Propositional Image Caption Evaluation. In *ECCV*.

Relja Arandjelovic and Andrew Zisserman. 2018. Objects that Sound. In *ECCV*.

Yusuf Aytar, Carl Vondrick, and Antonio Torralba. 2016. SoundNet: Learning Sound Representations from Unlabeled Video. In *NIPS*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. In *Stat*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *ACL Workshop MTSumm*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. In *TACL*.

Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. 2018. VoxCeleb2: Deep Speaker Recognition. In *Interspeech*.

Steven B Davis and Paul Mermelstein. 1980. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In *TASSP*.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *ICML*.

Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. 2018. Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. In *SIGGRAPH*.

Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. 2017. Freesound Datasets: A Platform for the Creation of Open Audio Datasets. In *ISMIR*.

Ruohan Gao, Rogerio Feris, and Kristen Grauman. 2018. Learning to Separate Object Sounds by Watching Unlabeled Video. In *ECCV*.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio Set: An Ontology and Human-labeled Dataset for Audio Events. In *ICASSP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *AISTATS*.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-Shot Recognition. In *ICCV*.

John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. 2016. Deep Clustering: Discriminative Embeddings for Segmentation and Separation. In *ICASSP*.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. CNN Architectures for Large-Scale Audio Classification. In *ICASSP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*.

Chiori Hori, Huda Alamri, Jue Wang, Gordon Wichern, Takaaki Hori, Anoop Cherian, Tim K. Marks, Vincent Cartillier, Raphael Gontijo Lopes, Abhishek Das, Irfan Essa, Dhruv Batra, and Devi Parikh. 2018. End-to-End Audio Visual Scene-Aware Dialog using Multimodal Attention-based Video Features. In *arXiv:1806.08409*.

Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John R Hershey, Tim K Marks, and Kazuhiko Sumi. 2017. Attention-based Multimodal Fusion for Video Description. In *ICCV*.

Keith Ito. 2017. The LJ Speech Dataset. `https://keithito.com/LJ-Speech-Dataset/`.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. In *CVPR*.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *CVPR*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-Captioning Events in Videos. In *ICCV*.

Natsuda Laokulrat, Naoaki Okazaki, and Hideki Nakayama. 2018. Incorporating Semantic Attention in Video Description Generation. In *LREC*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *TSBO*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in Context. In *ECCV*.

Xiang Long, Chuang Gan, and Gerard de Melo. 2018. Video Captioning with Multi-Faceted Attention. *TACL*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and Music Signal Analysis in Python. In *SCIPY*.

Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, and Tuomas Virtanen. 2017. DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System. In *DCASE*.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. 2016. TUT Database for Acoustic Scene Classification and Sound Event Detection. In *EUSIPCO*.

Mozilla. 2017. Mozilla Common Voice. `https://voice.mozilla.org/`.

Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. 2017. VoxCeleb: A Large-Scale Speaker Identification Dataset. In *Interspeech*.

Andrew Owens and Alexei A. Efros. 2018. Audio-Visual Scene Analysis with Self-Supervised Multisensory Features. In *ECCV*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR Corpus Based on Public Domain Audio Books. In *ICASSP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL*.

Karol J Piczak. 2015. ESC: Dataset for Environmental Sound Classification. In *ACM MM*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting Image Annotations Using Amazon's Mechanical Turk. In *NAACL-HLT*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2017. Movie Description. In *IJCV*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. 2014. A Dataset and Taxonomy for Urban Sound Research. In *ACM MM*.

Arda Senocak, Tae-Hyun Oh, Junsik Kim, Ming-Hsuan Yang, and In So Kweon. 2018. Learning to Localize Sound Source in Visual Scenes. In *CVPR*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *ICLR*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning. In *ACL*.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *JMLR*.

Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D Plumbley. 2015. Detection and Classification of Acoustic Scenes and Events. In *IEEE Transactions on Multimedia*.

Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based Image Description Evaluation. In *CVPR*.

Xin Wang, Yuan-Fang Wang, and William Yang Wang. 2018. Watch, Listen, and Describe: Globally and Locally Aligned Cross-Modal Attentions for Video Captioning. In *NAACL-HLT*.

Pete Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. In *arXiv:1804.03209*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A Large Video Description Dataset for Bridging Video and Language. In *CVPR*.

Yong Xu, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. 2018. Large-Scale Weakly Supervised Audio Classification using Gated Convolutional Neural Network. In *ICASSP*.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2016. Describing Videos by Exploiting Temporal Structure. In *ICCV*.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image Captioning with Semantic Attention. In *CVPR*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions. In *TACL*.

Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. 2017. End-to-End Concept Word Detection for Video Captioning, Retrieval, and Question Answering. In *CVPR*.

Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. 2018. The Sound of Pixels. In *ECCV*.

Appendix

In the supplemental material, we enlist the following which may shed further insights:

- Additional related work [section A]
- Additional dataset analysis [section B]
- Training Details [section C]

A Related Work

Audio-Visual correspondence. Over the past year, a great interest has been shone to the interconnection of auditory and visual senses. The task of localizing the sound source within the visual input have been actively explored (Nagrani et al., 2017; Chung et al., 2018; Senocak et al., 2018; Afouras et al., 2018; Gao et al., 2018; Arandjelovic and Zisserman, 2018; Zhao et al., 2018), along with blind source separation aided by visual features (Ephrat et al., 2018) and learning of audio-visual multisensory representation (Owens and Efros, 2018). These previous studies compensate the lack of information in the auditory input with visual information, whereas this work focuses solely on the auditory input to generate informative descriptions.

B Dataset

The full ontology of selected labels is outlined in Figure 7.

Figure 8 shows the number of clips per word label. The original AudioSet has an extreme label bias. For instance, a difference of 660,282 between the average of top 3 most common and average of top 3 most uncommon classes. Whereas our dataset at the moment has a difference of 971. Notice the label bias is significantly reduced in comparison to the original AudioSet. We plan to reduce this further in the upcoming releases.

Table 5 compares our audio captioning dataset with some representative benchmarks of video captioning: MSR-VTT (Xu et al., 2016) and LSMDC (Rohrbach et al., 2017). One interesting property of our dataset is that the portion of verbs in the vocabularies are larger than the others. This may imply that the captions describe what is *happening* rather than what *is* in the content.

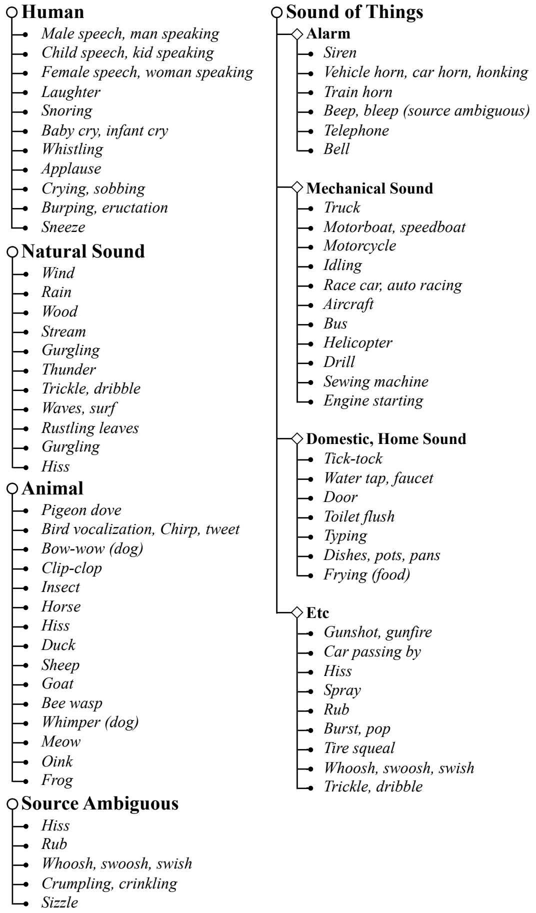


Figure 7: The curated ontology for *AudioCaps* on the basis of *AudioSet*.

C Training Details

All the parameters are initialized with Xavier method (Glorot and Bengio, 2010). We apply the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e - 8$.

Dataset	Clips	Sentences	Unique clips	Tokens	Vocabs	Nouns	Verbs	Adjectives	Adverbs	Duration(h)
`MSR-VTT`	10,000	200,000	7,180	1,856,523	29,316	16,437	6,379	3,761	872	41.2
`LSMDC`	128,085	128,118	200	1,157,155	22,500	12,181	3,394	5,633	1,292	147
`AudioCaps`	39,106	43,022	39,106	567,927	4,506	2,747	1,825	766	353	108.6

Table 5: Comparison of *AudioCaps* with MSR-VTT (Xu et al., 2016), LSMDC (Rohrbach et al., 2017).

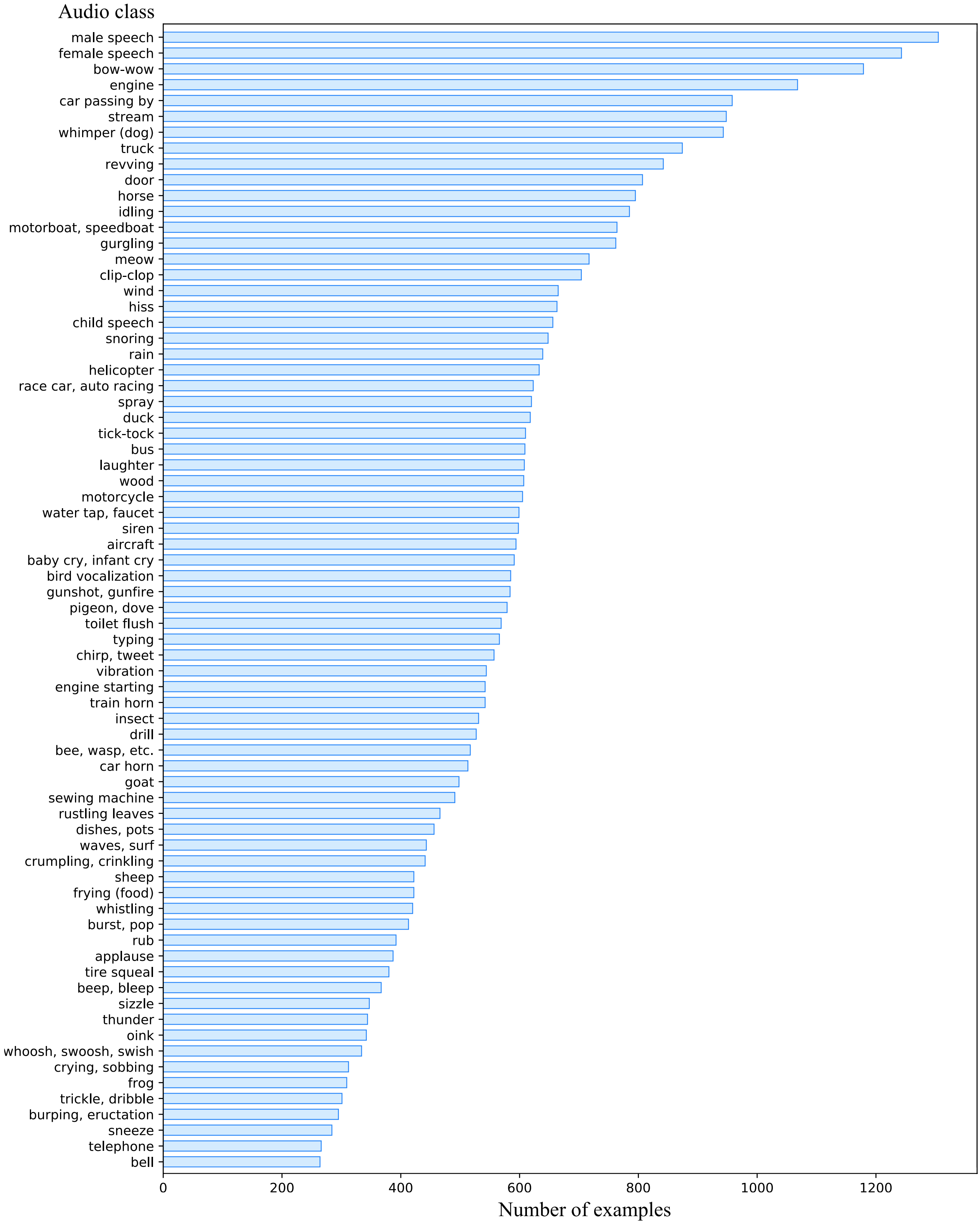
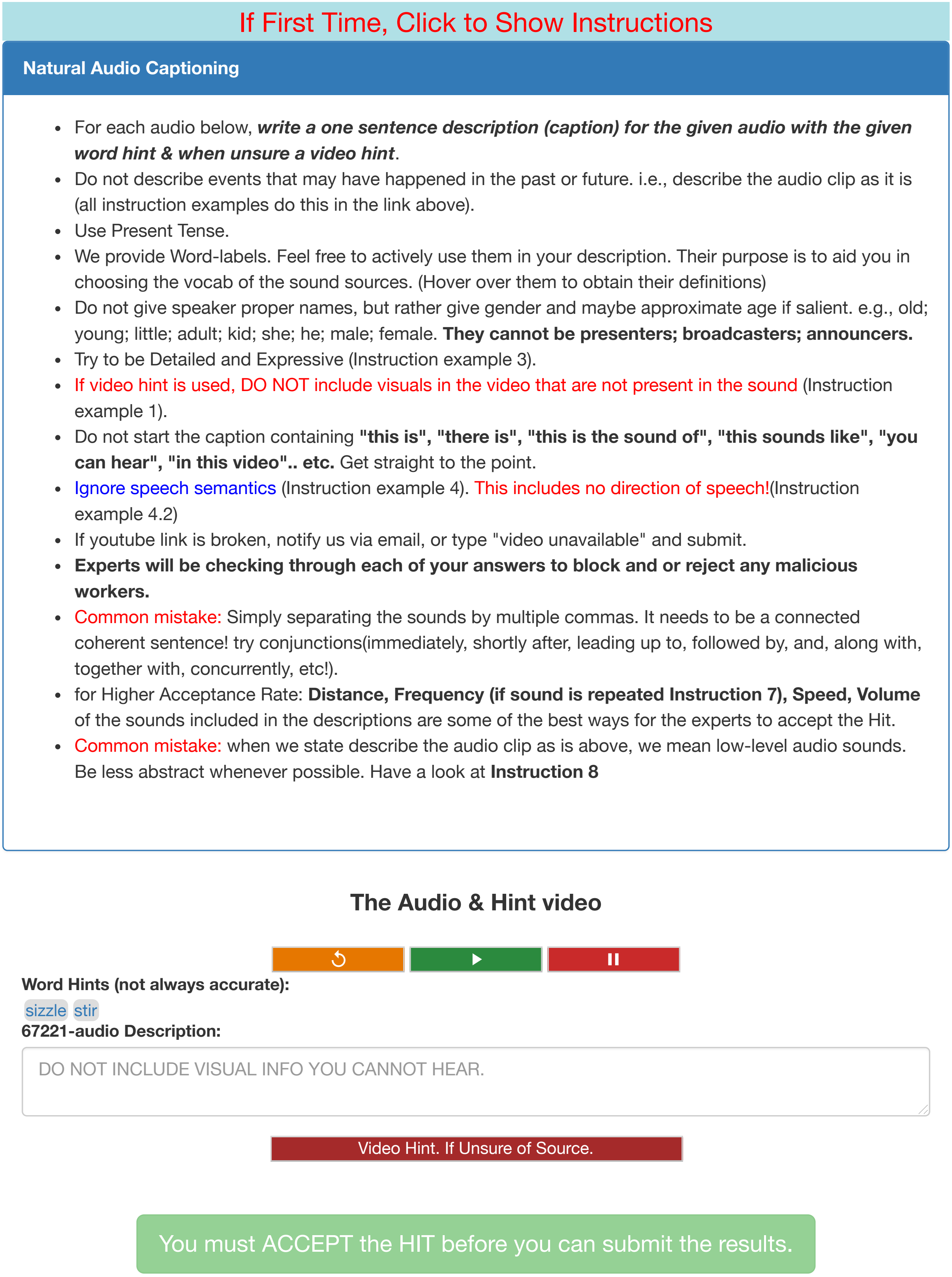


Figure 8: The frequencies of annotated instances per category (*i.e.* word labels) for *AudioCaps*.



If First Time, Click to Show Instructions

Natural Audio Captioning

- For each audio below, ***write a one sentence description (caption) for the given audio with the given word hint & when unsure a video hint***.
- Do not describe events that may have happened in the past or future. i.e., describe the audio clip as it is (all instruction examples do this in the link above).
- Use Present Tense.
- We provide Word-labels. Feel free to actively use them in your description. Their purpose is to aid you in choosing the vocab of the sound sources. (Hover over them to obtain their definitions)
- Do not give speaker proper names, but rather give gender and maybe approximate age if salient. e.g., old; young; little; adult; kid; she; he; male; female. **They cannot be presenters; broadcasters; announcers.**
- Try to be Detailed and Expressive (Instruction example 3).
- If video hint is used, DO NOT include visuals in the video that are not present in the sound (Instruction example 1).
- Do not start the caption containing **"this is", "there is", "this is the sound of", "this sounds like", "you can hear", "in this video".. etc.** Get straight to the point.
- Ignore speech semantics (Instruction example 4). This includes no direction of speech!(Instruction example 4.2)
- If youtube link is broken, notify us via email, or type "video unavailable" and submit.
- **Experts will be checking through each of your answers to block and or reject any malicious workers.**
- Common mistake: Simply separating the sounds by multiple commas. It needs to be a connected coherent sentence! try conjunctions(immediately, shortly after, leading up to, followed by, and, along with, together with, concurrently, etc!).
- for Higher Acceptance Rate: **Distance, Frequency (if sound is repeated Instruction 7), Speed, Volume** of the sounds included in the descriptions are some of the best ways for the experts to accept the Hit.
- Common mistake: when we state describe the audio clip as is above, we mean low-level audio sounds. Be less abstract whenever possible. Have a look at **Instruction 8**

The Audio & Hint video

Word Hints (not always accurate):

sizzle stir

67221-audio Description:

DO NOT INCLUDE VISUAL INFO YOU CANNOT HEAR.

Figure 9: The AMT interface for sentence annotation with instructions.

"President Vows to Cut ~~Taxes~~ Hair": Dataset and Analysis of Creative Text Editing for Humorous Headlines

Nabil Hossain
Dept. Computer Science
University of Rochester
Rochester, NY
`nhossain@cs.rochester.edu`

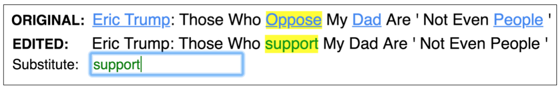
John Krumm and **Michael Gamon**
Microsoft Research AI
Microsoft Corporation
Redmond, WA
`{jckrumm,mgamon}@microsoft.com`

Abstract

We introduce, release, and analyze a new dataset, called Humicroedit, for research in computational humor. Our publicly available data consists of regular English news headlines paired with versions of the same headlines that contain simple replacement edits designed to make them funny. We carefully curated crowdsourced editors to create funny headlines and judges to score a to a total of 15,095 edited headlines, with five judges per headline. The simple edits, usually just a single word replacement, mean we can apply straightforward analysis techniques to determine what makes our edited headlines humorous. We show how the data support classic theories of humor, such as incongruity, superiority, and setup/punchline. Finally, we develop baseline classifiers that can predict whether or not an edited headline is funny, which is a first step toward automatically generating humorous headlines as an approach to creating topical humor.

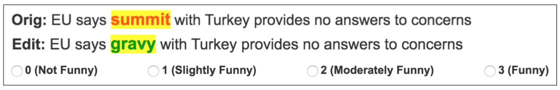
1 Introduction

Humor detection and generation continue to be challenging AI problems. While there have been some advances in automatic humor recognition (Khodak et al., 2017; Davidov et al., 2010; Barbieri and Saggion, 2014; Reyes et al., 2012; Cattle and Ma, 2018; Bertero and Fung, 2016; Yang et al., 2015), computerized humor generation has seen less progress (Binsted et al., 1997; Stock and Strapparava, 2003; Petrović and Matthews, 2013). This is not surprising, given that humor involves in-depth world-knowledge, common sense, and the ability to perceive relationships across entities and objects at various levels of understanding. Even humans often fail at being funny or recognizing humor.



ORIGINAL: Eric Trump: Those Who Oppose My Dad Are ' Not Even People '
EDITED: Eric Trump: Those Who support My Dad Are ' Not Even People '
Substitute: support

(a) The Headline Editing Task.



Orig: EU says summit with Turkey provides no answers to concerns
Edit: EU says gravy with Turkey provides no answers to concerns
○ 0 (Not Funny) ○ 1 (Slightly Funny) ○ 2 (Moderately Funny) ○ 3 (Funny)

(b) The Headline Grading Task.

Figure 1: Snapshots of the headline editing and grading interfaces. Only the underlined tokens are replaceable.

A big hindrance to progress on humor research is the scarcity of public datasets. Furthermore, the existing datasets address specific humor templates, such as funny one-liners (Mihalcea and Strapparava, 2006) and filling in Mad Libs® (Hossain et al., 2017). Creating a humor corpus is non-trivial, however, because it requires (i) human annotation, and (ii) a clear definition of humor to achieve good inter-annotator agreement.

We introduce **Humicroedit**, a novel dataset for research in computational humor. First, we collect original news headlines from news media posted on Reddit (`reddit.com`). Then, we *qualify* expert annotators from Amazon Mechanical Turk (`mturk.com`) to (i) generate humor by applying small edits to these headlines, and to (ii) judge the humor in these edits. Our resulting dataset contains 15,095 edited news headlines and their numerically assessed humor. Screenshots of our two annotation tasks are shown in Figure 1, and Table 1 shows some of these annotated headlines.

This new dataset enables various humor tasks, such as: (i) understanding what makes an edited headline funny, (ii) predicting whether an edited headline is funny, (iii) ranking multiple edits of the same headline on a funniness scale, (iv) generating humorous news headlines, and (v) recommending funny headlines personalized to a reader.

Our dataset presents several opportunities for computational humor research since:

Proceedings of NAACL-HLT 2019, pages 133–142
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

- Headlines do not have specific templates.
- Headlines contain very few words, but convey a lot of information.
- A deeper understanding of world-knowledge and common-sense is needed to completely understand what makes a headline funny.
- Humorous headlines are often generated using several layers of cognition and reasoning.
- Despite us carefully qualifying annotators, their knowledge, preferences, bias and stance towards information presented in headlines influence whether they perceive a potentially funny headline as humorous, offensive, confusing, etc.

The presence of these factors suggests that thorough humor comprehension in our dataset requires the development of NLP tools that are not only robust at pattern recognition but also capable of deeper semantic understanding and reasoning. As an initial exploration of this proposition, we perform various data analysis against the background of humor theories, and we train and examine classifiers to detect humorous edited headlines in our data.

2 The Humor Dataset

In this section, we describe how we gathered our set of original headlines, directed editors to make them funny, employed graders to assess the level of humor in the modified headlines, and created the Humicroedit dataset.

2.1 Task Description

Our goal is to study how humor is generated by applying short edits to headlines. News headlines are ripe for humor, since they convey rich information using only a few words. While the short form may seem to limit context, readers have rich background information in the form of their existing world knowledge, which helps them understand the headline. Allowing only short edits means we can apply focused analysis on the tipping point between regular and funny.

Therefore, our task is to edit a headline to make it funny, where an edit is defined as the insertion of a single-word noun or verb to replace an existing entity or single-word noun or verb. Note that our rules do not allow:

- Addition/removal of a whole noun/verb phrase, except removal of noun phrases that are entities (e.g., One World, Virtual Reality).
- Removal of sub-tokens within entities (e.g., replacing only “States” in “United States”).

The decision to strictly avoid edits of other parts-of-speech (POS) words was motivated by the observation in our pilot experiments that those edits did not provide enough variety of humor. For example, when substituting adjectives and adverbs, our editors mostly used antonyms or superlatives. Switching nouns and verbs, on the other hand, enables the introduction of diverse novel connections between entities and actions.

To identify the replaceable entities, we apply named entity recognition (NER) and POS tagging using the Stanford CoreNLP toolkit (Manning et al., 2014). We allow for replacement of only those entities that are well-known, according to the Microsoft Knowledge Base[1]. This improves the likelihood that the terms are familiar to both headline editors and humor judges. We allow a noun (or verb) to be replaced if it is an unambiguous noun (or verb) in WordNet (Fellbaum, 1998) (i.e., has a single WordNet POS). Editors are only allowed to replace one of the selected replaceable words/entities in the headline.

We refer to a single-term substitution of this type as a “**micro-edit**”, and we will use this term interchangeably with “edit” in the remainder of this paper. Micro-edits approach the smallest change that can induce humor in text, letting us focus intently on what causes humor.

2.2 Collecting Headlines

We build our dataset from popular news headlines posted on the social media site Reddit. This strategy steers us towards a set of headlines that is part of general discourse, rather than being only of specialized interest, which would make editing them for humor difficult.

We obtain all Reddit posts from the popular subreddits `r/worldnews` and `r/politics` from January 2017 to May 2018 using Google BigQuery[2]. Each of these posts is a headline from a news source. We remove duplicate headlines and headlines that have fewer than 4 words or more than 20 words. Finally, we keep only the headlines from the 25 English news sources that contribute the most headlines in the Reddit data, resulting in a total of 287,076 news headlines.

---

[1] http://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/

[2] `https://cloud.google.com/bigquery`

ID	Original Headline	Substitute	Grade	Prediction
1	Kushner to visit **Mexico** following latest Trump tirades	therapist	2.8	✘
2	Trump wants you to take his **tweets** seriously. His aides don't	hair	2.8	✔
3	Essential Politics: California's hottest congressional **races**, ranked	mistresses	2.8	✘
4	Hillary Clinton Staffers Considered Campaign Slogan 'Because It's Her **Turn**'	fault	2.8	✘
5	Trump Vows North Korea Could be Met With 'Fire and **Fury**'	marshmallows	2.6	✔
6	Here's how **Wall** Street is reacting to Trump's tax plan	sesame	2.4	✔
7	Swedish prosecutor says **truck** attack suspect has not spoken	mime	2.4	✘
8	**Steve Bannon** questioned by special counsel	kindergarteners	2.4	✘
9	New survey shows majority of US troops has 'unfavorable' view of Obama's **years**	ears	2.2	✘
10	The Latest: BBC cuts **ties** with Myanmar TV station	pies	1.8	N/A (train)
11	Bill Maher: "I **doubt** that Trump will be president the full term"	hope	0.2	✘
12	Malawi arrests 140 in clampdown after 'vampirism' **killings**	rumors	0.2	✘
13	**Rising** Dem star announces engagement to same-sex partner	gay	0.0	✘
14	**Taylor Swift** claims Denver DJ sexually assaulted her back in 2013	hen	0.0	✘
15	4 **soldiers** killed in Nagorno-Karabakh fighting: Officials	rabbits	0.0	✔

Table 1: Some headlines in our dataset and their edits, mean funniness grades, and accuracy of funniness prediction by LSTM. We acknowledge that some of these are offensive, but we use them for analysis in Section 4.1.

2.3 Annotation

For our data annotation tasks, we use Mechanical Turk workers who (i) are located in the U.S., (ii) have a HIT approval rate greater than 97%, and (iii) have more than 10,000 HITs approved. To ensure high data quality, we further qualify distinct sets of (i) turker judges for recognizing humor in an edited headline, and (ii) editors adept at editing headlines to generate humor.

2.3.1 Qualifying Humor Judges

We manually collected a set of 20 original news headlines and edited each of them such that some edits are funny and some are not. We asked several members of our research group to assess the funniness of each edited headline using the following integer scale developed by Hossain et al. (2017):

0 - Not funny
1 - Slightly funny
2 - Moderately funny
3 - Funny

We instructed internal and turker judges (i) to grade objectively regardless of their own stance towards issues, entities and information expressed in the headline, and (ii) to grade an edited headline as funny if they believed it would be funny to a large audience. Further, we instructed judges to grade an edited headline as funny if either the headline was funny by itself regardless of the original headline, or the headline was only funny when considering how the original headline was changed.

We labeled the ground truth funniness of each of these 20 edited qualifier headlines as its mean internal judge grade. For the qualification task, we classified as funny any edited headline with a mean grade of 1.0 or above.

Next, we launched the same task on Mechanical Turk until we found 150 qualified judges (60% of the candidates). Turkers were qualified if (i) they had 3 or fewer classification errors according to our 1.0 threshold, and (ii) on average, their grades were within 0.6 of the mean internal judge grades.

We calculated the inter-annotator agreement for assigning headline funniness grades using the Krippendorff's α interval metric (Krippendorff, 1970) — a real number in the range $[-1, 1]$, with -1, 0 and 1, respectively, implying complete disagreement, no consensus and full agreement. The α for the internal judges and qualified turker judges were, respectively, 0.57 and 0.64.

2.3.2 Qualifying Humor Editors

For editor qualification, we randomly sampled 60 headlines, split into 6 separate Mechanical Turk tasks of 10 headlines each. Candidate editors were asked to complete one of these tasks, which was to make each headline as funny as possible to a general audience using a micro-edit. Task participants were instructed not to apply the following edits:

- Cheap humor generation techniques: add profanity, slang, bathroom/potty humor, *crude* sexual references or informal language.
- Squeeze multiple words into one (e.g., Housecat, JumpedOverWall).

Next, we used 7 qualified judges to assess the funniness of each edited headline of each candidate. We qualified all candidates whose mean funniness of edited headlines was above 0.8 or the task's average headline's funniness grade, whichever was higher. In total, we obtained 100 qualified editors (57.5% of the candidates) who met our expectations in their ability to create funny headlines.

2.4 Data Collection and Quality Control

For our final dataset, we randomly sampled a total of 5,170 news headlines from our Reddit dataset, obtaining roughly an equal number of headlines from each news source. We asked 3 editors to edit each headline and 5 judges to grade each edited headline. Multiple micro-edits of the same headline allow us to compare different edits in terms of their effectiveness for generating humor, which we leave for future work.

To avoid turker exhaustion and decision fatigue, we performed the annotation task over a series of mini-batches launched at least 24 hours apart. After each round of editing, we applied tools to (i) check the edits for spelling mistakes which we manually corrected, and (ii) to find and eliminate inserted tokens that were a concatenation of two or more words (e.g., selftanner). To allow diversity in annotations, we applied a maximum HIT limit for annotators per batch. After each batch was completed, we temporarily suspended those editors and judges who had done significantly more HITs than the rest, until the others caught up.

Lastly, as we obtained more and more annotated data, the editors started employing the same humor generation strategies (e.g., inserting words from a small vocabulary). Consequently, judges saw repeated, identical edits, so the element of surprise was gone, and the judges were grading fewer humorous edited headlines as funny. We addressed this by randomly sampling a set of editors and judges for each batch, obtaining new editors and judges over time, and removing those editors who had done a majority of the HITs but whose edits' average funniness grade fell below a threshold (=0.7) after they participated in a batch. We also removed judges who repeatedly assigned very low funniness grades compared to the 4 other judges for the same edit. The judges' agreement score based on α was 0.20, showing modest agreement considering the factors above and others such as judges' personal preferences, bias, political stance, etc. which make consensus difficult.

Our Humicroedit dataset includes 15,095 unique edited headlines graded for funniness. For annotating a single headline, we paid 10 US cents to editors and 2.5 US cents to judges. There were also small costs for qualification. Our total cost for obtaining the dataset is about USD 4,500[3].

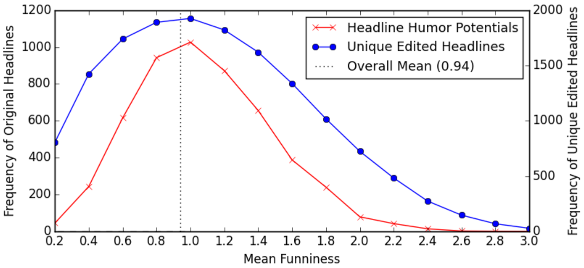


Figure 2: Histograms of humor potentials of headlines and mean funniness of unique edited headlines. The humor potential of a headline is the mean funniness score over all edits, shown as the red curve. The blue curve shows the histogram of the mean score of each distinct edited headline.

3 Humor Analysis

In this section, we analyze what types of micro-edits are effective at creating humor in our dataset, and we discuss our findings against the background of humor theories.

Figure 2 shows the histogram of the mean rating of each edited headline. While the majority of the headlines achieve slight to moderate levels of humor, some of them appear inherently difficult to make humorous by micro-editing. We noticed that editors encountered difficulty making headlines funny when the headlines had very negative themes, such as shootings, death, etc., and when they focused on information less likely to be known by a general audience (e.g., relatively unknown person, an insignificant political issue).

3.1 Humor Generation Strategies

By manual inspection, we can gain insights into humor generation strategies employed by our editors, which we discuss with references to Table 1:

1. Using a word that forms a meaningful n-gram with the adjacent words (e.g., ID 5: Fire and ~~Fury~~ marshmallows; ID 6: ~~Wall~~ sesame street).
2. Connection between replaced word and the replacement: replacements that are semantically distant from (e.g., ID 1: ~~Mexico~~ therapist) or similar in pronunciation to (e.g., ID 10: ~~ties~~ pies) to the replaced word.
3. Using a word that makes a strong connection with an entity in the headline (e.g., ID 2: Trump and hair; ID 9: Obama and ears).
4. Creating sarcasm (ID: 11).
5. Belittling an entity or noun in the headline (e.g., ID 4: Hillary Clinton's ~~turn~~ fault; ID 9: Obama's ~~years~~ ears).

---

[3] Dataset: cs.rochester.edu/u/nhossain/humicroedit.html. Total cost is USD 4,500, not USD $4,500^{3}$ (joke!)

6. Tension suppression[4]: making a serious headline silly (e.g., IDs 5 and 9).
7. Inserting words that generate incongruity (common among most examples in Table 1).
8. Setup and punchline: let the headline build up towards an expected ending, and then change words towards the end to produce a coherent but surprising ending (e.g., IDs 3, 4 and 5).

3.2 Clusters of Replacement Words

Each micro-edit used a new replacement word to change the headline. We clustered these replacement words using their GloVe word vectors (Pennington et al., 2014) and k-means clustering, with $k = 20$. Our manually-generated cluster names are shown in Table 2, where the clusters are ordered by the mean funniness score of the edited headlines whose replacement word is in the cluster. For each cluster, we show the frequency with which the cluster was used for replacement words and frequent sample words from the cluster.

We can compare our automatically generated clusters with those of Westbury and Hollis (2018). They manually created six clusters from the 200 funniest, single words in Engelthaler and Hills (2018) and then they added more words algorithmically. Four of their six manually curated classes have direct correspondences to our automatically curated classes: *sex*, *insults*, *bodily functions*, and *animals*. We did not find an equivalent to their *profanity* class, because we instructed our editors to avoid profanity. There is also a *party* class that we do not have. Overall, though, we find good agreement between their manually curated classes and some of our automatically generated clusters, leading us to believe that our clusters are meaningfully representative of humor generation strategies for our task.

3.3 Support for Theories of Humor

Our rated headlines give us an opportunity to explore theories of humor in a systematic way. We find, in general, that these theories are supported by our data.

3.3.1 Length of Joke

Although some linguists argue that jokes should make economical use of words (Tomoioagă, 2015), Ritchie (2004) argues that jokes often have extra information, which can make a joke funnier. While humorous headlines form a special niche of jokes, we observed that longer headlines generally had higher humor potential.

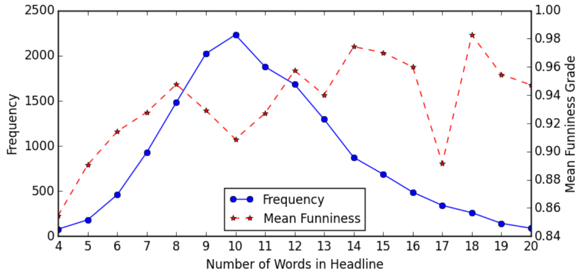


Figure 3: Short headlines did not lend themselves to high humor scores, while longer headlines generally had more potential for humor. The blue line shows the raw distribution of headline lengths in our data, and the red line shows the mean funniness score over different lengths.

Figure 3 shows that the population of our collected headlines from Reddit has a length distribution with a peak at 10 words and a long tail to the right. The least funny edited headlines are the shortest, and the most funny are the longest. This makes sense since very short headlines (4-5 words long) barely have enough contextual information to exploit to make a humorous edit, whereas headlines that have very rich contexts generally allow editors more flexibility to generate humor. We note that Dunbar et al. (2016) also found that longer jokes are funnier, but that some jokes could be too complicated to be funny.

We can also examine the number and proportion of *replaceable* words and how these numbers affect funniness. In our dataset, the number of replaceable words ranged between 1 and 12, and funniness grades of micro-edits were significantly lower at the two extremes. Editors apparently had difficulty generating humor when they were severely constrained in choosing a word to replace, or when they had too many choices for replacement. However, edited headlines with a higher proportion of replaceable words were generally funnier, as shown in Table 3. This suggests that allowing editors more freedom in choosing words from the headline to edit results in better humor, or that high proportion of nouns, entities and verbs in the headline increases the chance of successful humor generation.

3.3.2 Incongruity for Humor

We see evidence for the incongruity theory of humor (Morreall, 2016). Jokes that use incongruity

---

[4] This is also known as the relief theory of humor.

Class Label	Funniness	Frequency %	5 Frequent Sample Words
clothes and fashion	1.213	5.77	hair, pants, haircut, fashion, underwear
sex	1.209	3.52	orgy, spank, mistress, porn, striptease
food (savory)	1.113	3.89	cheese, sandwich, chicken, potato, tacos
eating	1.048	4.18	food, pizza, dinner, restaurant, eat
fantasy characters	1.036	4.94	clown, aliens, penguin, robot, ghost
music and shows	1.023	3.85	dance, circus, music, sings, mime
bodily functions	1.021	3.11	diet, brain, odor, dandruff, pimple
food (snacks)	1.017	5.36	pumpkin, cake, vodka, cookies, candy
animals	1.000	9.22	dog, monkey, puppy, cats, duck
various nouns (1)	0.935	5.82	toupee, hoedown, jaywalking, barbers, seance
jobs and roles	0.890	4.95	children, wife, baby, mother, barber
insults	0.876	3.36	clowns, tantrum, trolls, racist, whining
emotional	0.868	7.74	love, hug, jokes, fights, cry
leisure	0.862	6.01	vacation, shopping, tanning, hotel, pool
sports	0.856	3.03	horse, game, bowling, golf, wrestling
various nouns (2)	0.850	4.90	water, nose, balloon, gas, smoke
human deficiencies	0.803	4.21	lies, ignorance, humor, boredom, stupidity
media	0.787	4.20	tweet, movie, book, video, television
aspirations	0.726	7.28	party, date, people, money, model
corrupted	0.712	4.48	president, bribes, politicians, destroy, prison

Table 2: Twenty clusters of replacement words with manually determined cluster labels.

Repl. Words Prop.	0.2	0.4	0.6	0.8	1.0
Mean Funniness	0.87	0.92	0.95	0.94	1.0

Table 3: Edited headlines were judged funnier when they had a larger proportion of replaceable words.

aim to violate an expectation, with the expectation normally set up by the joke itself. We test incongruity by examining the relationship between the replacement words chosen by our editors and words in the original headline using cosine distances between their GloVe vectors. If incongruity is important, we expect the replacement word to be distant from the headline's original words.

The results of this analysis are shown in Figure 4. Our approach involved computing the correlations between mean funniness scores of edited headlines and different GloVe distances between their replacement words and the other words in the headline serving as context. In order to sharpen the analysis, we looked at subsets of headlines with extreme funniness scores. For instance, the left-most data points in Figure 4 pertain only to those edited headlines that are in the top and bottom 5% of mean scores, which filters out headlines whose scores are in the middle.

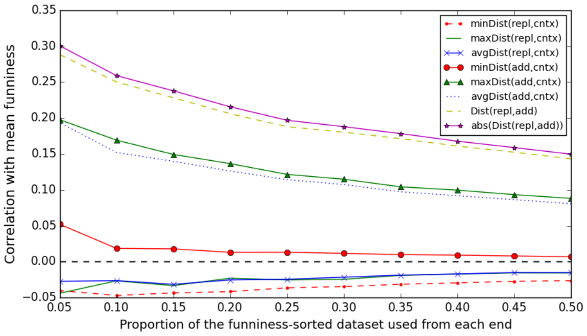


Figure 4: Correlations of word vector based cosine distances with mean funniness at various dataset sizes. We measure distance (and its absolute value) between the replaced word(s) and the added word, and also their minimum, maximum and average distances with the set of other words in the headline.

The four curves higher on the plot show a relatively high correlation between humor scores and the cosine distance between the added word ("add" in legend) and the replaced word ("repl" in legend) or the other words in the headline ("cntx" in legend). This suggests that incongruity leads to humor. The three lower curves show there is not a strong correlation between humor and the distance between the original, replaced word the other words in the headline. Finally, smaller, less humor-ambiguous data leads to stronger positive correlations, which suggests that higher incongruity leads to more quality humor.

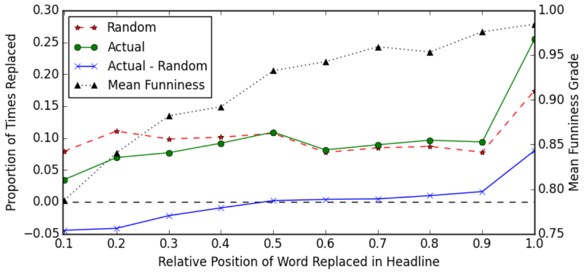


Figure 5: The joke profile graph for setup and punchline humor, showing that word substitutions toward the end of the headline normally lead to better humor.

3.3.3 Setup and Punchline

We specifically studied whether the "setup and punchline" (Rochmawati, 2017) approach is used in funny headline generation, where the humor comes toward the end of the joke after a setup at the beginning. This has been verified numerically for funny cartoon captions by Shahaf et al. (2015). For our analysis, we construct the **joke profile graph**, shown in Figure 5. It shows the proportion of time the editors substituted a word at each relative word position bin compared to if they randomly chose a word to substitute. Specifically, the red curve in the plot shows the proportion of replacement word locations if they were chosen randomly from those available in the editing task. The green curve shows the proportion of word locations actually chosen by our editors, and the blue curve shows the difference. We see that the blue line rises monotonically toward the end of the headline, meaning that editors tend to prefer replacing words later in the headline. The plot also shows the average funniness grade as a dotted line as a function of the position of the replacement word. It rises dramatically toward the end, showing that the funniest headlines were generally those with a replacement word toward the end.

3.3.4 Superiority for Humor

Jokes often express our feelings of superiority over someone else (Morreall, 2016). This can lead to frequent use of negative sentiment in jokes, as found by Mihalcea and Pulman (2007) in their analysis of humorous texts. We find similar support in our clusters of replacement words in Table 2, where the clusters labeled *insults*, *human deficiencies*, and *corrupted* are all comprised of words that tend to denigrate other people, accounting for about 12% of the substitute words inserted by our editors.

4 Humorous Headline Detection

In this section, we develop baseline classifiers to infer whether an edited headline is funny. Given our dataset, there are three possible combinations of information that we can use to detect humor:

1. Using only the edited headline to predict whether it is funny or not.
2. Using only the original headline to predict its potential for funniness.
3. Using both original and edited headlines to jointly predict resulting funniness.

We address the first of these scenarios. A classifier of this type could be used in a generate-and-test setting to create humorous headlines by trying different micro-edits.

To map the range of observed funniness grades (see Figure 2) to the funny/not-funny classes, we sort our full dataset in decreasing order of mean funniness[5] scores, and we take the top $X\%$ of the data from each end, at size intervals of 10%. Note that each train/test split has an equal number of funny and not-funny headlines, establishing a 50% majority class baseline for accuracy.

We first trained a number of non-neural classifiers (logistic regression, random forest, SVM), using two feature sets: n-gram features (1, 2 and 3-grams combined) and features based on GloVe embeddings[6] as shown in Figure 4. We use 80% of the data for training and 20% for testing. We optimized hyperparameters for accuracy on 10-fold cross validation on the training set. The random forest classifier consistently performed best, so we only report its test set performance.

We also applied a neural baseline model using a single-layer bi-directional LSTM (Hochreiter and Schmidhuber, 1997) with 16 hidden units, a dropout of 0.5, and GloVe pre-trained embedding of the sequence of words in the edited headline. The training set was further split into 80%-20% splits for training and validation. We used a mini-batch size of 32 with up to 25 epochs to train our model, optimizing for cross-entropy.

Table 5 shows the results obtained with our classifiers. LSTM performs better than random forest with either n-gram (Rf-ngram) or GloVe features (Rf-Glv), achieving our best accuracy of 68.54%

---

[5] We then sort by increasing standard deviations of grades to further rank headlines which tie on funniness, as lower standard deviation indicates stronger judge agreement.

[6] This is our only feature set that uses the replaced word.

Bin	0-.4	.4-.8	.8-1.2	1.2-1.6	1.6-2.0	2.0-2.4	2.4-3.0
Acc.	71.5	61.1	52.0	61.0	68.6	68.3	76.2

Table 4: LSTM accuracy for distinct grade bins (upper-bounds are inclusive) on the test set for $X = 40$.

using X = 10. We suspect that the reason for the LSTM's superior performance is that it learns predictive interactions between the semantics of the headline's words (via the GloVe embeddings) that trigger the humor[7].

Table 5 also shows that accuracy generally decreases as X increases, which is expected since higher X implies a smaller separation between funny and not-funny classes, making classification harder. This is further corroborated by the observation that annotator-agreement scores (also shown in Table 5) decrease similarly with increasing X, indicating that funny and not-funny classes are easier to distinguish at the extreme ends of the dataset for both humans and machines alike.

4.1 LSTM Classification Analysis

We now investigate the test-set performance of the LSTM trained on the dataset obtained using X = 40, the largest of our experimental datasets for which the class boundaries are distinct.

To analyze how well the classifier predicts the extremes in the test set, we obtained classification accuracy on distinct mean grade bins, presented in Table 4. The LSTM is able to distinguish the far extremes ($\leq$0.4 and $>$1.6) of the test set much more convincingly than the headlines with mean grades in the interval (0.4,1.6]. We found a slightly negative correlation between classification accuracy and standard deviation of grades. Using additional judges for headlines with high standard deviation of grades would possibly improve annotator agreement and classification accuracy.

The LSTM achieved a significantly lower accuracy when an entity (61.8%) was replaced by the micro-edit compared to when a noun (64.5%) or a verb (65.5%) was replaced. For 5 of the 7 bins in Table 4, the entity-replaced headline classification accuracy was lower than when the other two types were replaced, with the LSTM only achieving an accuracy of 47.9% on the (0.8,1.2] bin for entity-replaced headlines. Although the classifier is never shown what has been replaced, it is better at assessing humor when the replaced word is not an entity. Our judges did have access to the replaced word, so we speculate this knowledge is important when the replaced word is an entity, especially when the entity triggers the judge's recollection of their world knowledge surrounding the entity, which the LSTM does not have. Another potential reason is that the pretrained GloVe vectors are trained on web data (840 billion tokens obtained from Common Crawl) no more recent than 2014, which may not appropriately represent common entities in our 2017-2018 headline data.

Next, we qualitatively analyzed the LSTM's classification accuracy towards the two extremes of the dataset, some of which are shown in Table 1. Overall, the LSTM seems to suffer from a relatively high level of brittleness (possibly arising from the unusual writing style in headlines), where correct predictions could be obtained by very little modification to the text. For example, changing "Trump → Trump's" in ID 1 and deleting "Essential Politics:" in ID 3 fix their classification errors. Quotes in headlines also confused the LSTM (e.g., ID 4) since it is sometimes non-trivial to discern the speaker of the quote in a headline.

The classifier often had difficulty figuring out humorous replacements that involve common-sense knowledge (e.g., IDs 7 and 8). Not surprisingly, it also failed to detect offensive replacements as in IDs 13 and 14, where the model probably recognized the incongruity and marked these as funny. World knowledge and cultural references were other challenges (e.g., IDs 4, 9 and 14).

The LSTM was able to figure out some of the obvious negative sentiments which were common in unfunny headlines (e.g., ID 15), and it detected some humor patterns resulting from using words that form a common (but funny in the context) n-gram with the adjacent words (e.g., IDs 5 and 6).

Overall, our results show that there is a discernible signal separating funny and and not-funny headlines, even when using relatively shallow features that only take the content of the headline into account (modulo GloVe embeddings which are pretrained and hence contain semantic information gleaned from a larger corpus). We expect that further work, which could examine deeper relationships to current events, historical context, and common sense knowledge, will improve the ability to distinguish funny from not-funny beyond the baselines provided here.

---

[7] Using only words in the original headline produced accuracy in the mid-50% range, suggesting that the LSTM captures some humor impact of the replacement word as input and that some headlines have high potential for funniness.

X	MaxUF	MinF	α	Rf-Glv	Rf-ngram	Lstm-Glv
10	0.2	1.8	0.66	60.27	65.56	68.54
20	0.4	1.4	0.49	61.67	63.41	67.21
30	0.6	1.2	0.37	59.43	62.96	66.11
40	0.8	1.0	0.27	57.45	59.35	64.07
50	0.8	0.8	0.20	55.63	56.52	60.63

Table 5: Classification accuracy for various funniness-sorted dataset proportions and classifiers/feature sets. MaxUF is the highest score for the not-funny class, MinF is the lowest score for the funny class, and we also provide Krippendorff's α for judge agreement.

5 Related Work

Previous research on automated humor can be divided into work on datasets, analysis, detection, and generation. We will give examples of each.

Datasets are important for automated understanding of humor and for training models. Starting at the simplest linguistic level, Engelthaler and Hills (2018) gathered almost 5,000 English words with funniness ratings for each one. Filatova (2012) found 1,905 Amazon product reviews classified as either regular or ironic/sarcastic, and Khodak et al. (2017) collected 1.3 million sarcastic statements from Reddit and a much larger set of non-sarcastic statements. Mihalcea and Strapparava (2005) collected about 24,000 one-liner jokes, Potash et al. (2017) shared a dataset to rank funny tweets for certain hashtags, and Miller et al. (2017) created a task for pun detection.

Humor analysis, as we have done, is aimed at understanding what makes something funny. Building on the word-level corpus of Engelthaler and Hills (2018), Westbury and Hollis (2018) developed models to predict the funniness of 4,997 words. Looking at multi-word, but still short text, Shahaf et al. (2015) analyzed cartoon captions in order to understand what made some funnier than others. The work that is most similar to ours is from West and Horvitz (2019), who looked at pairs of funny and normal headlines. While we employed editors to create funny headlines from serious ones, they went the other way using a Web-based game, producing and analyzing 2,801 modified versions of 1,191 satirical headlines.

Humor detection is characterized by determining if a given text is funny or not. Examples include Khodak et al. (2017), detecting sarcasm in Reddit and Davidov et al. (2010) detecting sarcasm in Amazon product reviews and Twitter. Barbieri and Saggion (2014) and Reyes et al. (2012) showed how to detect humorous tweets, and Kiddon and Brun (2011) detected double entendres.

Generating humor is a difficult problem. Past work includes Binsted et al. (1997) producing punning riddles, funny acronyms from Stock and Strapparava (2003), jokes of the type "I like my coffee like I like my war, cold" by Petrović and Matthews (2013), and filling in Mad Libs® by Hossain et al. (2017). Our headline work has the potential to help in humor generation, moving away from jokes with a strong template to more free form.

6 Conclusion and Future Work

We have developed and released Humicroedit, a carefully curated dataset of 15,095 headlines with simple edits designed to make them funny. The dataset specifies the edits and also comes with five funniness scores for each edited headline. The simple replacement edits facilitate focused analysis on what causes the humor. We showed how our data supports, in a quantitative way, humor theories about length of joke, incongruity, superiority, and setup/punchline. Finally, we developed baseline classifiers that show how well we can distinguish funny edits from non-funny edits using simple linguistic features.

We expect our dataset will facilitate research in humor and natural language processing. Headlines present unique challenges and opportunities, because their humor is largely topical, depending on a knowledge of current events and prominent people and entities.

Future work with this data could include deeper features for assessing humor. We expect that humor detection would likely improve using features that incorporate world knowledge and common sense. Likewise, there may be something to learn by analyzing topical jokes from professional comedians. With our single-word edits, this analysis becomes easier, because we are looking at the minimal change in a headline to make it funny. Additionally, if we can better understand what makes a headline funny, we may be able to automatically generate funny headlines and even personalize them to particular readers.

Acknowledgments

We thank Daniel Gildea for carefully reviewing our paper and for his advice on the machine learning experiments. We also thank the NAACL reviewers for their various helpful suggestions.

References

Francesco Barbieri and Horacio Saggion. 2014. Automatic detection of irony and humour in twitter. In *ICCC*.

Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *NAACL-HLT*.

Kim Binsted, Helen Pain, and Graeme D Ritchie. 1997. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition*, 5(2):305–54.

Andrew Cattle and Xiaojuan Ma. 2018. Recognizing humour using word associations and humour anchor extraction. In *COLING*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *CoNLL*.

R.I.M Dunbar, Jacques Launay, and Oliver Curry. 2016. The complexity of jokes is limited by cognitive constraints on mentalizing. *Human Nature*, 27(2):130–140.

Tomas Engelthaler and Thomas T Hills. 2018. Humor norms for 4,997 english words. *Behavior research methods*, 50(3):1116–1124.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *LREC*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nabil Hossain, John Krumm, Lucy Vanderwende, Eric Horvitz, and Henry Kautz. 2017. Filling the blanks (hint: plural noun) for Mad Libs humor. In *EMNLP*.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *ACL*.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

Rada Mihalcea and Stephen Pulman. 2007. Characterizing humour: An exploration of features in humorous texts. In *CICLing*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *EMNLP*.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *SemEval*.

John Morreall. 2016. Philosophy of humor. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, winter 2016 edition. Metaphysics Research Lab, Stanford University.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *ACL*.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Semeval-2017 task 6: #hashtagwars: Learning a sense of humor. In *SemEval*.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Graeme Ritchie. 2004. *The linguistic analysis of jokes*. Routledge.

Dyah Rochmawati. 2017. Pragmatic and rhetorical strategies in the english-written jokes. *Indonesian Journal of Applied Linguistics*, 7(1):149–159.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *KDD*.

Oliviero Stock and Carlo Strapparava. 2003. Hahacronym: Humorous agents for humorous acronyms. *Humor*, 16(3):297–314.

Anca Tomoioagă. 2015. A linguistic analysis of jokes. *Discourse As A Form Of Multiculturalism In Literature And Communication, Section: Language And Discourse Arhipelag Xxi Press, TîrguMure*.

Robert West and Eric Horvitz. 2019. Reverse-engineering satire, or "paper on computational humor accepted despite making serious advances". In *AAAI*.

Chris Westbury and Geoff Hollis. 2018. Wriggly, squiffy, lummox, and boobs: What makes some words funny? *Journal of experimental psychology. General*.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *EMNLP*.

Answer-based Adversarial Training for Generating Clarification Questions

Sudha Rao[*]
Microsoft Research, Redmond
`Sudha.Rao@microsoft.com`

Hal Daumé III
University of Maryland, College Park
Microsoft Research, New York City
`me@hal3.name`

Abstract

We present an approach for generating clarification questions with the goal of eliciting new information that would make the given textual context more complete. We propose that modeling hypothetical answers (to clarification questions) as latent variables can guide our approach into generating more useful clarification questions. We develop a Generative Adversarial Network (GAN) where the generator is a sequence-to-sequence model and the discriminator is a utility function that models the value of updating the context with the answer to the clarification question. We evaluate on two datasets, using both automatic metrics and human judgments of usefulness, specificity and relevance, showing that our approach outperforms both a retrieval-based model and ablations that exclude the utility model and the adversarial training.

1 Introduction

A goal of natural language processing is to develop techniques that enable machines to process naturally occurring language. However, not all language is clear and, as humans, we may not always understand each other (Grice, 1975); in cases of gaps or mismatches in knowledge, we tend to ask questions (Graesser et al., 2008). In this work, we focus on the task of automatically generating clarification questions: questions that ask for information that is *missing* from a given linguistic context. Our clarification question generation model builds on the sequence-to-sequence approach that has proven effective for several language generation tasks (Sutskever et al., 2014; Serban et al., 2016; Yin et al., 2016; Du et al., 2017). Unfortunately, training a sequence-to-sequence model directly on (context, question) pairs yields questions that are highly generic[1], corroborating a common finding in dialog systems (Li et al., 2016b). Our goal is to be able to generate clarification questions that are useful *and* specific.

To achieve this, we begin with a recent observation of Rao and Daumé III (2018), who consider the task of question reranking: a good clarification question is the one whose answer has a high *utility*, which they define as the likelihood that this question would lead to an answer that will make the context more complete (§2.3). Inspired by this, we construct a model that first generates a question given a context, and then generates a hypothetical answer to that question. Given this (context, question, answer) triple, we train a utility calculator to estimate the usefulness of this question. We then show that this utility calculator can be generalized using ideas for generative adversarial networks (Goodfellow et al., 2014) for text (Yu et al., 2017), wherein the utility calculator plays the role of the "discriminator" and the question generator is the "generator" (§2.2), which we train using the MIXER algorithm (Ranzato et al., 2015). We evaluate our approach on two datasets: Amazon product descriptions (Figure 1) and Stack Exchange posts (Figure 2). Our two main contributions are:

1. An adversarial training approach for generating clarification questions that models the utility of updating a context with an answer to the clarification question. [2]
2. An empirical evaluation using both automatic metrics and human judgments to show that our adversarially trained model generates questions that are more *useful* and *specific to the context* than all the baseline models.

---

[*] This research performed when the author was still at University of Maryland, College Park.

[1] For instance, under home appliances, frequently asking "Is it made in China?" or "What are the dimensions?"

[2] Code and data: `https://github.com/raosudha89/clarification_question_generation_pytorch`

Proceedings of NAACL-HLT 2019, pages 143–155
Minneapolis, Minnesota, June 2 - June 7, 2019.

Product title	T-fal Nonstick Cookware Set, 18 pieces, Red
Product description	Easy non-stick 18pc set includes every piece for your everyday meals. Exceptionally durable dishwasher safe cookware for easy clean up. Durable non-stick interior. Oven safe up to 350.F/177.C
Question	Are they induction compatible?
Answer	They are aluminium so the answer is NO.

Figure 1: Sample product description from Amazon paired with a clarification question and answer.

Title	Wifi keeps dropping on 5Ghz network
Post	Recently my wireless has been iffy at my university. I notice I am connected to a 5Ghz network, while I am connected to a 2.4Ghz everywhere else (where things work fine). Sometimes it reconnects, but I have to run 'sudo service network-manager restart'. Is it possible a kernel update caused this?
Question	what is the make of your wifi card ?
Answer	intel corporation wireless 7260 (rev 73)

Figure 2: Sample post from stackexchange.com paired with a clarification question and answer.

2 Training a Clarification Question Generator

Our goal is to build a model that, given a context, can generate an appropriate clarification question. Our dataset consists of (*context*, *question*, *answer*) triples where the *context* is an initial textual context, *question* is the clarification question that asks about some missing information in the context and *answer* is the answer to the clarification question (details in § 3.1). Representationally, our question generator is a standard sequence-to-sequence model with attention (§2.1). The learning problem is: how to train the sequence-to-sequence model to generate good clarification questions.

An overview of our training setup is shown in Figure 3. Given a context, our question generator, which is a sequence-to-sequence model, outputs a question. In order to evaluate the usefulness of this question, we then have a second sequence-to-sequence model called the "answer generator" that generates a hypothetical answer based on the context and the question (§ 2.5). This (context, generated question and generated answer) triple is fed into a UTILITY calculator, whose initial goal is to estimate the probability that this (question, answer) pair is useful in this context (§ 2.3). This UTILITY is treated as a reward, which is used to update the question generator using the MIXER (Ranzato et al., 2015) algorithm (§ 2.2). Finally, we reinterpret the answer-generator-plus-utility-calculator component as a *discriminator* for differentiating between (context, true question, generated answer) triples and (context, generated question, generated answer) triples , and optimize the generator for this adversarial objective using MIXER (§2.4).

2.1 Sequence-to-sequence Model for Question Generation

We use a standard attention based sequence-to-sequence model (Luong et al., 2015) for our question generator. Given an input sequence (context) $c = (c_1, c_2, ..., c_N)$, this model generates an output sequence (question) $q = (q_1, q_2, ..., q_T)$. The architecture of this model is an encoder-decoder with attention. The encoder is a recurrent neural network (RNN) operating over the input word embeddings to compute a source context representation $\tilde{c}$. The decoder uses this source representation to generate the target sequence one word at a time:

$$\begin{aligned} p(q|\tilde{c}) &= \prod_{t=1}^{T} p(q_t|q_1, q_2, ..., q_{t-1}, \tilde{c}_t) \\ &= \prod_{t=1}^{T} \mathit{softmax}(W_s\tilde{h}_t) \quad ; \\ \text{where } \tilde{h}_t &= \tanh(W_c[\tilde{c}_t; h_t]) \end{aligned} \tag{1}$$

In Eq 1, $\tilde{h}_t$ is the attentional hidden state of the RNN at time t and W_s and W_c are parameters of the model.[3] The predicted token q_t is the token in the vocabulary that is assigned the highest probability using the softmax function. The standard training objective for sequence-to-sequence model is to maximize the log-likelihood of all (c, q) pairs in the training data D which is equivalent to minimizing the following loss,

$$L_{\text{mle}}(D) = -\sum_{(c,q)\in D} \sum_{t=1}^{T} \log p(q_t|q_1, ..., q_{t-1}, \tilde{c}_t) \tag{2}$$

---

[3] Details are in Appendix A.

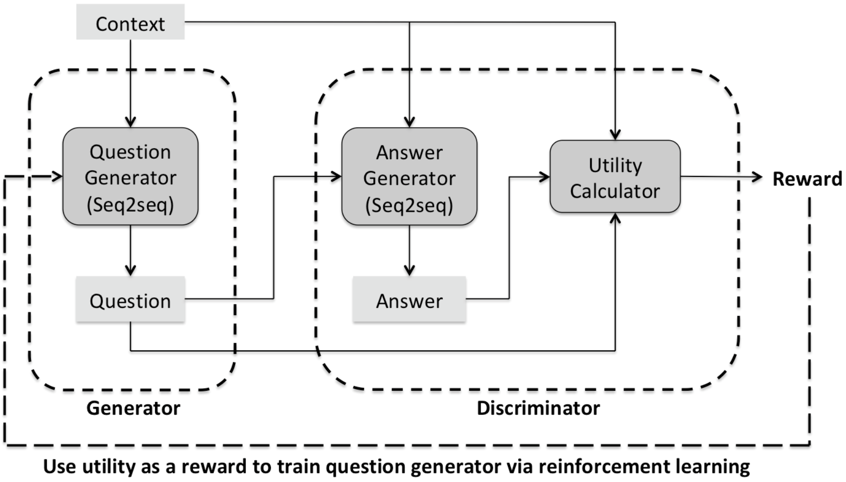


Figure 3: Overview of our GAN-based clarification question generation model (refer preamble of §2)

2.2 Training the Generator to Optimize UTILITY

Training sequence-to-sequence models for the task of clarification question generation (with context as input and question as output) using maximum likelihood objective unfortunately leads to the generation of highly generic questions, such as *"What are the dimensions?"* when asking questions about home appliances. Recently, Rao and Daumé III (2018) observed that the usefulness of a question can be better measured as the *utility* that would be obtained if the context were updated with the answer to the proposed question. Following this observation, we first use a pretrained answer generator (§2.5) to generate an answer given a context and a question. We then use a pretrained UTILITY calculator (§ 2.3) to predict the likelihood that the generated answer would increase the utility of the context by adding useful information to it. Finally, we train our question generator to optimize this UTILITY based reward.

Similar to optimizing metrics like BLEU and ROUGE, this UTILITY calculator also operates on discrete text outputs, which makes optimization difficult due to non-differentiability. A successful recent approach dealing with the non-differentiability while also retaining some advantages of maximum likelihood training is the Mixed Incremental Cross-Entropy Reinforce (Ranzato et al., 2015) algorithm (MIXER). In MIXER, the overall loss L is differentiated as in REINFORCE (Williams, 1992):

$$\begin{aligned} L(\theta) &= -\mathbb{E}_{q^s \sim p_\theta} r(q^s) \quad ; \\ \nabla_\theta L(\theta) &= -\mathbb{E}_{q^s \sim p_\theta} r(q^s) \nabla_\theta \log p_\theta(q^s) \end{aligned} \tag{3}$$

where q^s is a random output sample according to the model p_θ and θ are the parameters of the network. The expected gradient is then approximated using a single sample $q^s = (q_1^s, q_2^s, ..., q_T^s)$ from the model distribution (p_θ). In REINFORCE, the policy is initialized randomly, which can cause long convergence times. To solve this, MIXER starts by optimizing maximum likelihood for the initial Δ time steps, and slowly shifts to optimizing the expected reward from Eq 3 for the remaining $(T - \Delta)$ time steps.

In our model, for the initial Δ time steps, we minimize L_{mle} and for the remaining steps, we minimize the following UTILITY-based loss:

$$L_{\text{max-utility}} = -(r(q^p) - r(q^b)) \sum_{t=1}^{T} \log p(q_t | q_1, ..., q_{t-1}, \tilde{c}_t) \tag{4}$$

where $r(q^p)$ is the UTILITY based reward on the predicted question and $r(q^b)$ is a baseline reward introduced to reduce the high variance otherwise observed when using REINFORCE. To estimate this baseline reward, we take the idea from the self-critical training approach Rennie et al. (2017) where the baseline is estimated using the reward obtained by the current model under greedy decoding during test time. We find that this approach for baseline estimation stabilizes our model better than the approach used in MIXER.

2.3 Estimating UTILITY from Data

Given a (context, question, answer) triple, Rao and Daumé III (2018) introduce a utility calculator UTILITY(c, q, a) to calculate the value of updating a context c with the answer a to a clarification question q. They use the utility calculator to estimate the probability that an *answer* would be a meaningful addition to a context. They treat this as a binary classification problem where the positive instances are the true (context, question, answer) triples in the dataset whereas the negative instances are contexts paired with a random (question, answer) from the dataset. Following Rao and Daumé III (2018), we model our UTILITY calculator by first embedding the words in c and then using an LSTM (long-short term memory) (Hochreiter and Schmidhuber, 1997) to generate a neural representation $\bar{c}$ of the context by averaging the output of each of the hidden states. Similarly, we obtain neural representations $\bar{q}$ and $\bar{a}$ of q and a respectively using a question and an answer LSTM models. Finally, we use a feed forward neural network $F_{\text{UTILITY}}(\bar{c}, \bar{q}, \bar{a})$ to predict the usefulness of the question.

2.4 UTILITY GAN for Clarification Question Generation

The UTILITY calculator trained on true vs random samples from real data (as described in the previous section) can be a weak reward signal for questions generated by a model due to the large discrepancy between the true data and the model's outputs. In order to strengthen the reward signal, we reinterpret the UTILITY calculator (coupled with the answer generator) as a discriminator in an adversarial learning setting. That is, instead of taking the UTILITY calculator to be a fixed model that outputs the expected quality of a (question, answer) pair, we additionally optimize it to distinguish between true (question, answer) pairs and model-generated ones. This reinterpretation turns our model into a form of a generative adversarial network (GAN) (Goodfellow et al., 2014).

GAN is a training procedure for "generative" models that can be interpreted as a game between a generator and a discriminator. The generator is a model $g \in \mathcal{G}$ that produces outputs (in our case, questions). The discriminator is another model $d \in \mathcal{D}$ that attempts to classify between true outputs and model-generated outputs. The goal of the generator is to generate data such that it can fool the discriminator; the goal of the discriminator is to be able to successfully distinguish between real and generated data. In the process of trying to fool the discriminator, the generator produces data that is as close as possible to the real data distribution. Generically, the GAN objective is:

$$L_{\text{GAN}}(\mathcal{D}, \mathcal{G}) = \max_{d \in \mathcal{D}} \min_{g \in \mathcal{G}} \mathbb{E}_{x \sim \hat{p}} \log d(x) + \mathbb{E}_{z \sim p_z} \log(1 - d(g(z))) \tag{5}$$

where x is sampled from the true data distribution $\hat{p}$, and z is sampled from a prior defined on input noise variables p_z.

Although GANs have been successfully used for image tasks, training GANs for text generation is challenging due to the discrete nature of outputs in text. The discrete outputs from the generator make it difficult to pass the gradient update from the discriminator to the generator. Recently, Yu et al. (2017) proposed a sequence GAN model for text generation to overcome this issue. They treat their generator as an agent and use the discriminator as a reward function to update the generative model using reinforcement learning techniques. Our GAN-based approach is inspired by this sequence GAN model with two main modifications: a) We use MIXER algorithm as our generator (§ 2.2) instead of a purely policy gradient approach; and b) We use UTILITY calculator (§ 2.3) as our discriminator instead of a convolutional neural network (CNN).

Theoretically, the discriminator should be trained using (context, true question, true answer) triples as positive instances and (context, generated question, generated answer) triples as the negative instances. However, we find that training a discriminator using such positive instances makes it very strong since the generator would have to not only generate real looking questions but also generate real looking answers to fool the discriminator. Since our main goal is question generation and since we use answers only as latent variables, we instead use (context, true question, *generated answer*) as our positive instances where we use the pretrained answer generator to get the *generated answer* for the true question. Formally, our objective function is:

$$L_{\text{GAN-U}}(\mathcal{U}, \mathcal{M}) = \max_{u \in \mathcal{U}} \min_{m \in \mathcal{M}} \mathbb{E}_{q \sim \hat{p}} \log u(c, q, \mathcal{A}(c, q)) + \mathbb{E}_{c \sim \hat{p}} \log(1 - u(c, m(c), \mathcal{A}(c, m(c)))) \tag{6}$$

where $\mathcal{U}$ is the UTILITY discriminator, $\mathcal{M}$ is the MIXER generator, $\hat{p}$ is our data of (context, question, answer) triples and $\mathcal{A}$ is the answer generator.

2.5 Pretraining

Question Generator. We pretrain our question generator using the sequence-to-sequence model

(§2.1) to maximize the log-likelihood of all (context, question) pairs in the training data. Parameters of this model are updated during adversarial training.

Answer Generator. We pretrain our answer generator using the sequence-to-sequence model (§2.1) to maximize the log-likelihood of all ([context+question], answer) pairs in the training data. Parameters of this model are kept fixed during the adversarial training.[4]

Discriminator. In our UTILITY GAN model (§ 2.4), the discriminator is trained to differentiate between true and generated questions. However, since we want to guide our UTILITY based discriminator to also differentiate between true ("good") and random ("bad") questions, we pretrain our discriminator in the same way we trained our UTILITY calculator. For positive instances, we use a context and its true question, answer from the training data and for negative instances, we use the same context but randomly sample a question from the training data (and use the answer paired with that random question).

3 Experimental Results

We base our experimental design on the following research questions:

1. Do generation models outperform simpler retrieval baselines?
2. Does optimizing the UTILITY reward improve over maximum likelihood training?
3. Does using adversarial training improve over optimizing the pretrained UTILITY?
4. How do the models perform when evaluated for nuances such as specificity & usefulness?

3.1 Datasets

We evaluate our model on two datasets.

Amazon. In this dataset, *context* is a product description on amazon.com combined with the product title, *question* is a clarification question asked to the product and *answer* is the seller's (or other users') reply to the question. To obtain these data triples, we combine the Amazon question-answering dataset (McAuley and Yang, 2016) with the Amazon reviews dataset (McAuley et al., 2015). We show results on the `Home & Kitchen` category of this dataset since it contains a large number of questions and is relatively easier for human-based evaluation. It consists of $19,119$ training, $2,435$ tune and $2,305$ test examples (product descriptions), with 3 to 10 questions (average: 7) per description.

Stack Exchange. In this dataset, *context* is a post on stackexchange.com combined with the title, *question* is a clarification question asked in the comments section of the post and *answer* is either the update made to the post in response to the question or the author's reply to the question in the comments section. Rao and Daumé III (2018) curated a dataset of $61,681$ training, $7,710$ tune and $7,709$ test such triples from three related subdomains on stackexchage.com (askubuntu, unix and superuser). Additionally, for 500 instances each from the tune and the test set, their dataset includes 1 to 6 other questions identified as valid questions by expert human annotators from a pool of candidate questions.

3.2 Baselines and Ablated Models

We compare three variants (ablations) of our proposed approach, together with an information retrieval baseline:

GAN-Utility is our full model which is a UTILITY calculator based GAN training (§2.4) including the UTILITY discriminator and the MIXER question generator.[5]

Max-Utility is our reinforcement learning baseline where the pretrained question generator model is further trained to optimize the UTILITY reward (§2.2) without the adversarial training.

MLE is the question generator model pretrained on context, question pairs using maximum likelihood objective (§2.1).

Lucene[6] is our information retrieval baseline similar to the Lucene baseline described in Rao and Daumé III (2018). Given a context in the test set, we use Lucene, which is a TF-IDF based document ranker, to retrieve top 10 contexts that are most similar to the given context in the train set. We randomly choose a question from the human written questions paired with these 10 contexts in the train set to construct our Lucene baseline[7].

---

[4] We leave the experimentation of updating parameters of answer generator during adversarial training to future work.

[5] Experimental details are in Appendix B.

[6] `https://lucene.apache.org/`

[7] For the Amazon dataset, we ignore questions asked to products of the same brand as the given product since Amazon replicates questions across same brand allowing the true question to be included in that set.

3.3 Evaluation Metrics

We evaluate initially with automated evaluation metrics, and then more substantially with crowdsourced human judgments.

3.3.1 Automatic Metrics

Diversity, which calculates the proportion of unique trigrams in the output to measure the diversity as commonly used to evaluate dialogue generation (Li et al., 2016b).
BLEU (Papineni et al., 2002) [8], which evaluates n-gram precision between the output and the references.
METEOR (Banerjee and Lavie, 2005), which is similar to BLEU but includes stemmed and synonym matches to measure similarity between the output and the references.

3.3.2 Human Judgements

We use Figure-Eight[9], a crowdsourcing platform, to collect human judgements. Each judgement[10] consists of showing the crowdworker a context and a generated question and asking them to evaluate the question along following axes:
Relevance: We ask *"Is the question on topic?"* and let workers choose from: Yes (1) and No (0)
Grammaticality: We ask *"Is the question grammatical?"* and let workers choose from: Yes (1) and No (0)
Seeking new information: We ask *"Does the question ask for new information currently not included in the description?"* and let workers choose from: Yes (1) and No (0)
Specificity: We ask *"How specific is the question?"* and let workers choose from:

4: Specific pretty much only to this product (or same product from different manufacturer)
3: Specific to this and other very similar products
2: Generic enough to be applicable to many other products of this type
1: Generic enough to be applicable to any product under Home and Kitchen
0: N/A (Not applicable) i.e. Question is not on topic OR is incomprehensible

Usefulness: We ask *"How useful is the question to a potential buyer (or a current user) of the product?"* and let workers choose from:

4: Useful enough to be included in the product description
3: Useful to a large number of potential buyers (or current users)
2: Useful to a small number of potential buyers (or current users)
1: Useful only to the person asking the question
0: N/A (Not applicable) i.e. Question is not on topic OR is incomprehensible OR is not seeking new information

Criteria	**Agreement**
Relevance	0.92
Grammaticality	0.92
Seeking new information	0.84
Usefulness	0.65
Specificity	0.72

Table 1: Inter-annotator agreement on the five criteria used in human-based evaluation.

3.3.3 Inter-annotator Agreement

Table 1 shows the inter-annotator agreement (reported by Figure-Eight as confidence[11]) on each of the above five criteria. Agreement on *Relevance*, *Grammaticality* and *Seeking new information* is high. This is not surprising given that these criteria are not very subjective. On the other hand, the agreement on usefulness and specificity is quite moderate since these judgments can be very subjective.

Since the inter-annotator agreement on the usefulness criteria was particularly low, in order to reduce the subjectivity involved in the fine grained annotation, we convert the range [0-4] to a more coarse binary range [0-1] by mapping the scores 4 and 3 to **1** and the scores 2, 1 and 0 to **0**.

3.4 Automatic Metric Results

Table 2 shows the results on the two datasets when evaluated according to automatic metrics.

In the Amazon dataset, GAN-Utility outperforms all ablations on DIVERSITY, suggesting that it produces more diverse outputs. Lucene, on the other hand, has the highest DIVERSITY since it consists of human written questions, which tend to be more diverse because they are much longer compared to model generated questions. This comes at the cost of lower match with the reference as visible in the BLEU and METEOR scores.

---

[8] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[9] `https://www.figure-eight.com`

[10] We paid crowdworkers 5 cents per judgment and collected five judgments per question.

[11] `https://success.figure-eight.com/hc/en-us/articles/201855939-How-to-Calculate-a-Confidence-Score`

	`Amazon`			`StackExchange`		
Model	DIVERSITY	BLEU	METEOR	DIVERSITY	BLEU	METEOR
Reference	0.6934	—	—	0.7509	—	—
Lucene	0.6289	4.26	10.85	0.7453	1.63	7.96
MLE	0.1059	**17.02**	12.72	0.2183	3.49	8.49
Max-Utility	0.1214	16.77	12.69	**0.2508**	3.89	8.79
GAN-Utility	**0.1296**	15.20	**12.82**	0.2256	**4.26**	**8.99**

Table 2: DIVERSITY as measured by the proportion of unique trigrams in model outputs. Bigrams and unigrams follow similar trends. BLEU and METEOR scores using up to 10 references for the Amazon dataset and up to six references for the StackExchange dataset. Numbers in bold are the highest among the models. All results for Amazon are on the entire test set whereas for StackExchange they are on the 500 instances of the test set that have multiple references.

In terms of BLEU and METEOR, there is inconsistency. Although GAN-Utility outperforms all baselines according to METEOR, the fully ablated MLE model has a higher BLEU score. This is because BLEU score looks for exact n-gram matches and since MLE produces more generic outputs, it is much more likely that it will match one of 10 references compared to the specific/diverse outputs of GAN-Utility, since one of those ten is highly likely to itself be generic.

In the StackExchange dataset GAN-Utility outperforms all ablations on both BLEU and METEOR. Unlike in the Amazon dataset, MLE does not outperform GAN-Utility in BLEU. This is because the MLE outputs in this dataset are not as generic as in the amazon dataset due to the highly technical nature of contexts in StackExchange. As in the Amazon dataset, GAN-Utility outperforms MLE on DIVERSITY. Interestingly, the Max-Utility ablation achieves a higher DIVERSITY score than GAN-Utility. On manual analysis we find that Max-Utility produces longer outputs compared to GAN-Utility but at the cost of being less grammatical.

3.5 Human Judgements Analysis

Table 3 shows the numeric results of human-based evaluation performed on the reference and the system outputs on 300 random samples from the test set of the Amazon dataset.[12] All approaches produce relevant and grammatical questions. All models are all equally good at seeking new information, but are weaker than Lucene, which performs better at seeking new information but at the cost of much lower specificity and lower usefulness.

Our full model, GAN-Utility, performs significantly better at the usefulness criteria showing that the adversarial training approach generates more useful questions. Interestingly, all our models produce questions that are more useful than Lucene and Reference, largely because Lucene and Reference tend to ask questions that are more often useful only to the person asking the question, making them less useful for potential other buyers (see Figure 4). GAN-Utility also performs significantly better at generating questions that are more specific to the product (see details in Figure 5), which aligns with the higher DIVERSITY score obtained by GAN-Utility under automatic metric evaluation.

Table 5 contains example outputs from different models along with their usefulness and specificity scores. MLE generates questions such as *"is it waterproof?"* and *"what is the wattage?"*, which are applicable to many other products. Whereas our GAN-Utility model generates more specific question such as *"is this shower curtain mildew resistant?"*. Appendix C includes further analysis of system outputs on both Amazon and Stack Exchange datasets.

4 Related Work

Question Generation. Most previous work on question generation has been on generating reading comprehension style questions i.e. questions that ask about information present in a given text (Heilman, 2011; Rus et al., 2010, 2011; Duan et al., 2017). Our goal, on the other hand, is to generate questions whose answer cannot be found

---

[12] We could not ask crowdworkers evaluate the StackExchange data due to its highly technical nature.

Model	Relevant [0-1]	Grammatical [0-1]	New Info [0-1]	Useful [0-1]	Specific [0-4]
Reference	0.96	0.99	0.93	0.72	3.38
Lucene	**0.90**	**0.99**	**0.95**	0.68	2.87
MLE	**0.92**	**0.96**	0.85	0.91	3.05
Max-Utility	**0.93**	**0.96**	0.88	0.91	3.29
GAN-Utility	**0.94**	**0.96**	0.87	**0.96**	**3.52**

Table 3: Results of human judgments on model generated questions on 300 sample Home & Kitchen product descriptions. Numeric range corresponds to the options described in §3.3. The difference between the bold and the non-bold numbers is statistically significant with p <0.05. Reference is excluded in the significance calculation.

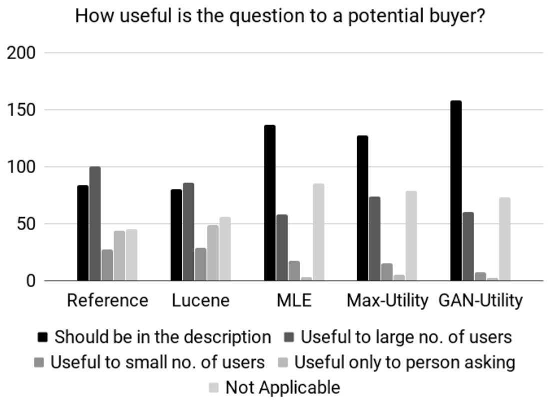


Figure 4: Human judgements on the usefulness criteria.

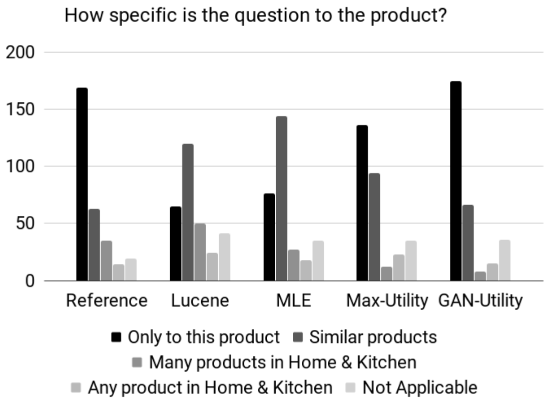


Figure 5: Human judgements on the specificity criteria.

in the given text. Outside reading comprehension questions, Liu et al. (2010) use templated questions to help authors write better related work sections whereas we generate questions to fill information gaps. Labutov et al. (2015) use crowdsourcing to generate question templates whereas we learn from naturally occurring questions. Mostafazadeh et al. (2016, 2017) generate natural and engaging questions, given an image (and some initial text). Whereas, we generate questions specifically for identifying missing information. Stoyanchev et al. (2014) generate clarification questions to resolve ambiguity caused by speech recognition failures during dialog, whereas we generate clarification questions to resolve ambiguity caused by missing information. The recent work most relevant to our work is by Rao and Daumé III (2018). They build a model which given a context and a set of candidate clarification questions, ranks them in a way that more useful clarification questions would be higher up in the ranking. In our work, we build on their ideas to propose a model that generates (instead of ranking) clarification questions given a context.

Neural Models and Adversarial Training for Text Generation. Neural network based models have had significant success at a variety of text generation tasks, including machine translation (Bahdanau et al., 2015; Luong et al., 2015), summarization (Nallapati et al., 2016), dialog (Bordes et al., 2016; Li et al., 2016a; Serban et al., 2017), textual style transfer (Jhamtani et al., 2017; Rao and Tetreault, 2018) and question answering (Yin et al., 2016; Serban et al., 2016). Our task is most similar to dialog, in which a wide variety of possible outputs are acceptable, and where lack of specificity in generated outputs is common. We addresses this challenge using an adversarial network approach (Goodfellow et al., 2014), a training procedure that can generate natural-looking outputs, which have been effective for natural image generation (Denton et al., 2015). Due to the challenges in optimizing over discrete output spaces like text, Yu et al. (2017) introduced a Seq(uence)GAN approach where they overcome this issue by using REINFORCE to optimize. Our GAN-Utility model is inspired by the SeqGAN model where we replace their policy gra-

Title	Raining Cats and Dogs Vinyl Bathroom **Shower Curtain**		
Product Description	This adorable shower curtain measures 70 by 72 inches and is sure to make a great gift!		
		Usefulness [0-4]	Specificity [0-4]
Reference	does the vinyl smells?	3	4
Lucene	other than home sweet home, what other sayings on the curtain?	2	4
MLE	is it waterproof ?	4	2
Max-Utility	is this shower curtain mildew ?	0	0
GAN-Utility	is this shower curtain mildew resistant ?	4	4
Title	PURSONIC HF200 Pedestal **Bladeless Fan & Humidifier** All-in-one		
Product Description	The first bladeless fan to incoporate a humidifier! This product operates solely as a fan, a humidifier or both simultaneously. Atomizing function via ultrasonic. 5.5L tank lasts up to 12 hours.		
		Usefulness [0-4]	Specificity [0-4]
Reference	i can not get the humidifier to work	1	2
Lucene	does it come with the vent kit	3	3
MLE	what is the wattage of this fan ?	4	2
Max-Utility	is this battery operated ?	3	2
GAN-Utility	does this fan have an automatic shut off ?	4	4

Table 4: Example outputs from each of the systems for two product descriptions along with the usefulness and the specificity score given by human annotators.

dient based generator with a MIXER model and their CNN based discriminator with our UTILITY calculator. Li et al. (2017) train an adversarial model similar to SeqGAN for generating next utterance in a dialog given a context. However, unlike our work, their discriminator is a binary classifier trained only to distinguish between human and machine generated utterances.

5 Conclusion

In this work, we describe a novel approach to the problem of clarification question generation. We use the observation of Rao and Daumé III (2018) that the usefulness of a clarification question can be measured by the value of updating a context with an answer to the question. We use a sequence-to-sequence model to generate a question given a context and a second sequence-to-sequence model to generate an answer given the context and the question. Given the (context, generated question, generated answer) triple, we calculate the utility of this triple and use it as a reward to retrain the question generator using reinforcement learning based MIXER model. Further, to improve upon the utility calculator, we reinterpret it as a discriminator in an adversarial setting and train both the utility calculator and the MIXER model in a minimax fashion. We find that our adversarial training approach produces more useful and specific questions compared to both a model trained using maximum likelihood objective and a model trained using utility reward based reinforcement learning.

There are several avenues of future work. Following Mostafazadeh et al. (2016), we could combine text input with image input in the Amazon dataset (McAuley and Yang, 2016) to generate more relevant and useful questions. One significant research challenge in the space of free text generation problems when the set of possible outputs is large, is that of automatic evaluation (Lowe et al., 2016): in our results we saw some correlation between human judgments and automatic metrics, but not enough to trust the automatic metrics completely. Lastly, we hope to integrate such a question generation model into a real world platform like StackExchange or Amazon to understand the real utility of such models and to unearth additional research questions.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments and suggestions. We also thank the members of the Computational Linguistics and Information Processing (CLIP) lab at University of Maryland for helpful discussions. This work was supported by NSF grant IIS-1618193. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Art Graesser, Vasile Rus, and Zhiqiang Cai. 2008. Question classification schemes. In *Proc. of the Workshop on Question Generation*.

H Paul Grice. 1975. Logic and conversation. *1975*, pages 41–58.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 889–898.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sėbastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 366–376. Association for Computational Linguistics.

Ryan Lowe, Iulian V. Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. In *SIGDIAL*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 462–472.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1802–1813.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Sudha Rao and Joel Tetreault. 2018. Dear Sir or Madam, May I introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer. In *HLT-NAACL*. The Association for Computational Linguistics.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge: Status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 318–320. Association for Computational Linguistics.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg. 2014. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*, volume 20.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2972–2978. AAAI Press.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *arxiv*.

A Sequence-to-sequence model details

In this section, we describe some of the details of the attention based sequence-to-sequence model introduced in Section 2.1 of the main paper. In equation 1, $\tilde{h}_t$ is the attentional hidden state of the RNN at time t obtained by concatenating the target hidden state h_t and the source-side context vector $\tilde{c}_t$, and W_s is a linear transformation that maps h_t to an output vocabulary-sized vector. Each attentional hidden state $\tilde{h}_t$ depends on a distinct input context vector $\tilde{c}_t$ computed using a global attention mechanism over the input hidden states as:

$$\tilde{c}_t = \sum_{n=1}^{N} a_{nt} h_n \tag{7}$$

$$a_{nt} = \text{align}(h_n, h_t) \tag{8}$$

$$= \exp\left[h_t^T W_a h_n\right] \Big/ \sum_{n'} \exp\left[h_t^T W_a h_{n'}\right] \tag{9}$$

The attention weights a_{nt} is calculated based on the alignment score between the source hidden state h_n and the current target hidden state h_t.

B Experimental Details

In this section, we describe the details of our experimental setup.

We tokenize and lowercase all inputs (context, question and answers). We set the max length of context to be 100, question to be 20 and answer to be 20. We find that increasing the length of contexts (to 150 or 200) of question/ answer (to 40) yields similar results according to automatic metrics with increased experimentation time.

Our sequence-to-sequence model (Section 2.1) operates on word embeddings which are pretrained on in domain data using Glove (Pennington et al., 2014). As frequently used in previous work on neural network modeling, we use an embeddings of size 200 and a vocabulary with cut off frequency set to 10. During train time, we use teacher forcing (Williams and Zipser, 1989). During test time, we use beam search decoding with beam size 5.

We use a hidden layer of size two for both the encoder and decoder recurrent neural network models with size of hidden unit set to 100. We use a dropout of 0.5 and learning ratio of 0.0001. In the MIXER model, we start with $\Delta = T$ and decrease it by 2 for every epoch (we found decreasing Δ to 0 is ineffective for our task, hence we stop at 2).

C Analysis of System Outputs

C.1 Amazon Dataset

First half of Table 5 shows the system generated questions for three product descriptions in the Amazon dataset.

In the first example, the product is a shower curtain. The Reference question is specific and highly useful. Lucene, on the other hand, picks a moderately specific ("how to clean it?") but useful question. MLE model generates a generic but useful "is it waterproof?". Max-Utility generates comparatively a much longer question but in doing so loses out on relevance. This behavior of generating two unrelated sentences is observed quite a few times in both Max-Utility and GAN-Utility models. This suggests that these models, in trying to be very specific, end up losing out on relevance. In the same example, GAN-Utility also generates a fairly long question which, although awkwardly phrase, is quite specific and useful.

In the second example, the product is a Duvet Cover Set. Both Reference and Lucene questions here are examples of questions that are pretty much useful only to the person asking the question. We find many such questions in both Reference and Lucene outputs which is the main reason for the comparatively lower usefulness scores for their outputs. All three of our models generate irrelevant questions since the product description explicitly says that the set is full size.

In the last example, the product is a set of mopping clothes. Reference question is quite specific but has low usefulness. Lucene picks an irrelevant question. MLE and Max-Utility generate highly specific and useful questions. GAN-Utility generates an ungrammatical question by repeating the last word many times. We observe this behavior quite a few times in the outputs of both Max-Utility and GAN-Utility models suggesting that our sequence-to-sequence models are not very good at maintaining long range dependencies.

C.2 Stack Exchange Dataset

Second half of Table 5 shows the system generated questions for three posts from the Stack Exchange dataset.

The first example is of a post where someone describes their issue of not being able to recover from their boot. Reference and Lucene questions are useful. MLE generates a generic question that is not very useful. Max-Utility generates a useful question but has slight ungrammaticality in it. GAN-Utility, on the other hand, generates a specific and an useful question.

In the second example, again Reference and Lucene questions are useful. MLE generates a generic question. Max-Utility and GAN-Utility both generate fairly specific question but contain unknown tokens. The Stack Exchange dataset contains several technical terms leading to a long tail in the vocabulary. Owing to this, we find that both Max-Utility and GAN-Utility models generate many instances of questions with unknown tokens.

In the third example, the Reference question is very generic. Lucene asks a relevant question. MLE again generates a generic question. Both Max-Utility and GAN-Utility generate specific and relevant questions.

	Amazon Dataset		
Title	Extra Wide **Shower Curtain** for a Clawfoot Tub/opaque with Magnets		
Description	Approximately 180” Wide 70” Long. Extra Wide to keep the water and heat in. Opaque in color to let the light in. Can be used as a liner or the curtain itself.		
		Usefulness [0-4]	Specificity [0-4]
Reference	does this curtain come with hooks?	4	4
Lucene	how to clean it ? i had it over 10 months now . and some mildew grow on it	3	3
MLE	is it waterproof ?	1	2
Max-Utility	what is the diameter of the bottom of the shower ? i have a kohler shower curtain .	0	0
GAN-Utility	does this curtain have suction feet on the bottom of the shower curtain ?	3	4
Title	Chezmoi Collection 7-piece Chic Ruched White **Duvet Cover Set**, Full Size (with Throw Pillows)		
Description	Create a world of romance with the elegantly-styled, ruching,and luxurious all white duvet cover set.		
		Usefulness [0-4]	Specificity [0-4]
Reference	How long will it take to ship this to my pin code?	1	1
Lucene	and can you use the duvet as is ? if not what shall i purchase to put in it for winter or fall ?	1	4
MLE	what are the dimensions of the king size ?	0	0
Max-Utility	what are the dimensions of the king size ?	0	0
GAN-Utility	does the king size come with a duvet cover or do you have to buy a king duvet ?	0	0
Title	Microfiber 3-Pack, **Pro-Clean Mopping Cloths** for Braava Floor Mopping Robot		
Description	Braavas textured Pro-Clean microfiber mopping cloths remove dirt and hair from your floors. The cloths can be washed and used hundreds of times. They are compatible with all Braava models, including the Pro-Clean Reservoir Pad. Each cloth is easy to attach and remove from the magnetic cleaning pad.		
		Usefulness [0-4]	Specificity [0-4]
Reference	do i have to use a new cloth every time i want to clean my floor? $5/$6 seems expensive per clean	2	4
Lucene	do they remove pet odor ?	0	0
MLE	will these work with the scooba ?	3	3
Max-Utility	do these cloths work on hardwood floors ?	3	4
GAN-Utility	will this work with the scooba mop mop mop mop mop mop mop	0	0
	StackExchange Dataset		
Title	how can i recover my boot ?		
Post	since last week i am trying to recover my boot . after the last update for ubuntu 12.04 i lost it and nobody could help me , i used boot-repair but there are problems with dependences , which can’t be fix . i will be very grateful if somebody could help me .		
Reference	what happens when you try to boot ?		
Lucene	can you get into bios ?		
MLE	how much ram do you have ?		
Max-Utility	do you have a swap partition partition ?		
GAN-Utility	what happens when you try to boot into safe mode ?		
Title	packages have unmet dependencies when trying to install anything		
Post	i ’m running ubuntu 14.04 lts . ive recently run into this problem with several applications , it seems to happen to anything i need to update or install . ive tried a lot of solutions and nothing seems to work.		
Reference	sudo dpkg -reconfigure all ? ?		
Lucene	if you use the graphical package manager , does n’t add the required packages automatically ?		
MLE	how long did you wait ?		
Max-Utility	can you post the output of ‘apt-cache policy UNK ?		
GAN-Utility	can you post a screenshot of the output of ‘sudo apt-get install UNK		
Title	full lubuntu installation on usb (uefi capable)		
Post	i want to do a full lubuntu installation on a usb stick that can be booted in uefi mode. i do not want persistent live usb but a full lubuntu installation and that can boot fromanyuefi-capable computer ...		
Reference	hello and welcome on askubuntu . could you please clarify what you want ?		
Lucene	so , ubuntu was installed to the pen drive ?		
MLE	which version of ubuntu ?		
Max-Utility	do you have a live cd or usb stick ?		
GAN-Utility	what is the model of the usb stick ?		

Table 5: Example outputs from each of the systems for three product descriptions from the Home & Kitchen category of the Amazon dataset and for three posts of StackExchange dataset.

Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, Jingming Liu
Yuanfudao Research / Beijing, China
{zhaowei01,wangliang01,shenkw,jiary,liujm}@fenbi.com

Abstract

Neural machine translation systems have become state-of-the-art approaches for Grammatical Error Correction (GEC) task. In this paper, we propose a copy-augmented architecture for the GEC task by copying the unchanged words from the source sentence to the target sentence. Since the GEC suffers from not having enough labeled training data to achieve high accuracy. We pre-train the copy-augmented architecture with a denoising auto-encoder using the unlabeled One Billion Benchmark and make comparisons between the fully pre-trained model and a partially pre-trained model. It is the first time copying words from the source context and fully pre-training a sequence to sequence model are experimented on the GEC task. Moreover, We add token-level and sentence-level multi-task learning for the GEC task. The evaluation results on the CoNLL-2014 test set show that our approach outperforms all recently published state-of-the-art results by a large margin. The code and pre-trained models are released at https://github.com/zhawe01/fairseq-gec.

1 Introduction

Grammatical Error Correction (GEC) is a task of detecting and correcting grammatical errors in text. Due to the growing number of language learners of English, there has been increasing attention to the English GEC, in the past decade.

The following sentence is an example of the GEC task, where the word in bold needs to be corrected to its adverb form.

*Nothing is [**absolute** → absolutely] right or wrong.*

Although machine translation systems have become state-of-the-art approaches for GEC, GEC is different from translation since it only changes several words of the source sentence. In Table 1, we list the ratio of unchanged words of the target sentence to the source sentence in three different datasets. We can observe that more than 80% of the words can be copied from the source sentence.

Corpus	Sent.	Tok.	Same %
CoNLL-2013	1,381	28,944	96.50%
JFELG	754	14,240	84.23%
Lang-8	4,936	73,705	83.22%

Table 1: The ratio of unchanged words in the target sentence to the source sentence. “Sent.” means the sentence number. “Tok.” means the token number of the target sentence. “Same %” means the same word percentage.

Considering the percentage of unchanged words is high in the GEC task, a more proper neural architecture is needed for it. We enhance the current neural architecture by enabling it to copy the unchanged words and the out-of-vocabulary words directly from the source sentence, just as what humans do when they correct sentences. To our knowledge, this is the first time that neural copying mechanism is used on GEC.

Progresses have been made thanks to large-scale training corpus, including NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) and the large-scale Lang-8 corpus(Tajiri et al., 2012). However, even with millions of labeled sentences, automatic GEC is challenging due to the lack of enough labeled training data to achieve high accuracy.

To alleviate the problem of insufficient labeled data, we propose a method to leverage the unlabeled data. The concrete way is to pre-train our copy-augmented model with the unlabeled One Billion Benchmark (Chelba et al., 2013) by leveraging denoising auto-encoders.

We also add two multi-tasks for the copy-augmented architecture, including a token-level

Proceedings of NAACL-HLT 2019, pages 156–165
Minneapolis, Minnesota, June 2 - June 7, 2019.

labeling task and a sentence-level copying task, to further improve the performance of the GEC task.

The copying mechanism is for the first time used on the GEC task, which was used on text summarization tasks. On the GEC task, copying mechanism enables training a model with a small vocabulary since it can straightly copy the unchanged and out-of-vocabulary words from the source input tokens. Besides, by separating the constant part of the work from the GEC task, copying makes the generating portion of the architecture more powerful. In the experiment section of this paper, we show that copying does more than just solving the "UNK problem", and it can also recall more edits for the GEC problem.

The copy-augmented architecture outperforms all the other architectures on the GEC task, by achieving a 56.42 $F_{0.5}$ score on the CoNLL 2014 test data set. Combined with denoising auto-encoders and multi-tasks, our architecture achieves 61.15 $F_{0.5}$ on the CoNLL-2014 test data set, improving +4.9 $F_{0.5}$ score than state-of-the-art systems.

In summary, our main contributions are as follows. (1) We propose a more proper neural architecture for the GEC problem, which enables copying the unchanged words and out-of-vocabulary words directly from the source input tokens. (2) We pre-train the copy-augmented model with large-scale unlabeled data using denoising auto-encoders, alleviating the problem of the insufficient labeled training corpus. (3) We evaluate the architecture on the CoNLL-2014 test set, which shows that our approach outperforms all recently published state-of-the-art approaches by a large margin.

2 Our Approach

2.1 Base Architecture

Neural machine translation systems have become the state-of-the-art approaches for Grammatical Error Correction (GEC), by treating the sentence written by the second language learners as the source sentence and the grammatically corrected one as the target sentence. Translation models learn the mapping from the source sentence to the target sentence.

We use the attention based Transformer (Vaswani et al., 2017) architecture as our baseline. The Transformer encodes the source sentence with a stack of L identical blocks, and each of them applies a multi-head self-attention over the source tokens followed by position-wise feedforward layers to produce its context-aware hidden state. The decoder has the same architecture as the encoder, stacking L identical blocks of multi-head attention with feed-forward networks for the target hidden states. However, the decoder block has an extra attention layer over the encoder's hidden states.

The goal is to predict the next word indexed by t in a sequence of word tokens $(y_1, ..., y_T)$, given the source word tokens $(x_1, ..., x_N)$, as follows:

$$h^{src}_{1...N} = encoder(L^{src} x_{1...N}) \tag{1}$$

$$h_t = decoder(L^{trg} y_{t-1...1}, h^{src}_{1...N}) \tag{2}$$

$$P_t(w) = softmax(L^{trg} h_t) \tag{3}$$

The matrix $L \in R^{d_x \times |V|}$ is the word embedding matrix, where d_x is the word embedding dimension and $|V|$ is the size of the vocabulary. $h^{src}_{1...N}$ is the encoder's hidden states and h_t is the target hidden state for the next word. Applying softmax operation on the inner product between the target hidden state and the embedding matrix, we get the generation probability distribution of the next word.

$$l_{ce} = -\sum_{t=1}^{T} log(p_t(y_t)) \tag{4}$$

The loss l_{ce} of each training example is an accumulation of the cross-entropy loss of each position during decoding.

2.2 Copying Mechanism

Copying mechanism was proved effective on text summarization tasks (See et al., 2017; Gu et al., 2016) and semantic parsing tasks (Jia and Liang, 2016). In this paper, we apply the copying mechanism on GEC task, for the first time, enabling the model to copy tokens from the source sentence.

As illustrated in Figure 1, besides generating words from a fixed vocabulary, our copy-augmented network allows copying words from the source input tokens. Defined in Equation 5, the final probability distribution P_t is a mix of the generation distribution P^{gen}_t and the copy distribution P^{copy}_t. As a result, the fixed vocabulary is extended by all the words appearing in the source sentence. The balance between the copying

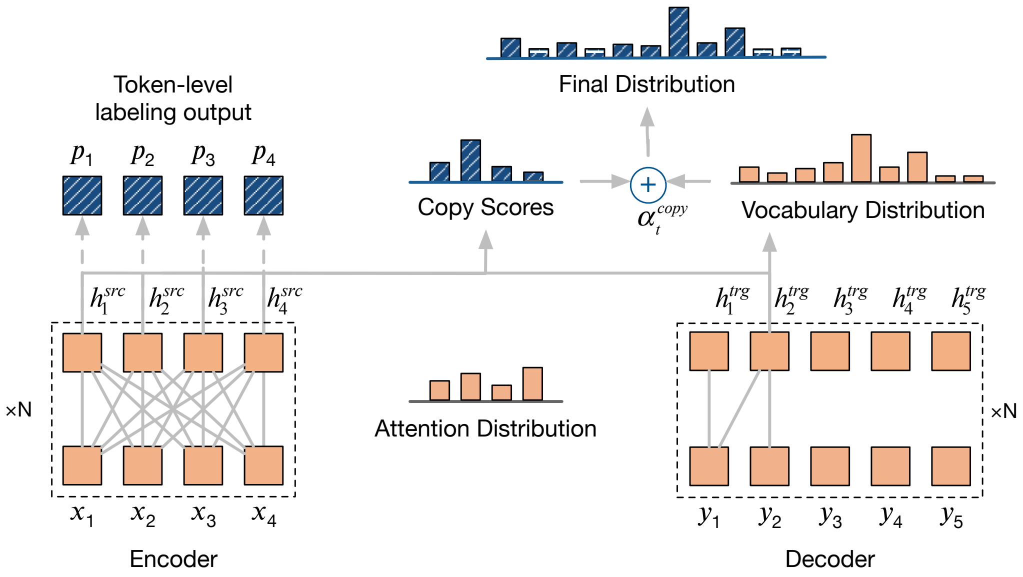


Figure 1: Copy-Augmented Architecture.

and generating is controlled by a balancing factor $\alpha_t^{copy} \in [0, 1]$ at each time step t.

$$p_t(w) = (1-\alpha_t^{copy}) * p_t^{gen}(w) + (\alpha_t^{copy}) * p_t^{copy}(w) \quad (5)$$

The new architecture outputs the generation probability distribution as the base model, by generating the target hidden state. The copying score over the source input tokens is calculated with a new attention distribution between the decoder's current hidden state h^{trg} and the encoder's hidden states H^{src} (same as $h^{src}_{1...N}$). The copy attention is calculated the same as the encoder-decoder attentions, listed in Equation 6, 7, 8 :

$$q_t, K, V = h_t^{trg} W_q^T, H^{src} W_k^T, H^{src} W_v^T \quad (6)$$

$$A_t = q_t^T K \quad (7)$$

$$P_t^{copy}(w) = softmax(A_t) \quad (8)$$

The q_t, K and V are the query, key, and value that needed to calculate the attention distribution and the copy hidden state. We use the normalized attention distribution as the copy scores and use the copy hidden states to estimate the balancing factor α_t^{copy}.

$$\alpha_t^{copy} = sigmoid(W^T \sum (A_t^T \cdot V)) \quad (9)$$

The loss function is as described in Equation 4, but with respect to our mixed probability distribution y_t given in Equation 5.

3 Pre-training

Pre-training is shown to be useful in many tasks when lacking vast amounts of training data. In this section, we propose denoising auto-encoders, which enables pre-training our models with large-scale unlabeled corpus. We also introduce a partially pre-training method to make a comparison with the denoising auto-encoder.

3.1 Denoising Auto-encoder

Denoising auto-encoders (Vincent et al., 2008) are commonly used for model initialization to extract and select features from inputs. BERT (Devlin et al., 2018) used a pre-trained bi-directional transformer model and outperformed existing systems by a wide margin on many NLP tasks. In contrast to denoising auto-encoders, BERT only predicts the 15% masked words rather than reconstructing the entire input. BERT denoise the 15% of the tokens at random by replacing 80% of them with [MASK], 10% of them with a random word and 10% of them unchanged.

Inspired by BERT and denoising auto-encoders, we pre-traine our copy-augmented sequence to sequence model by noising the One Billion Word Benchmark (Chelba et al., 2013), which is a large sentence-level English corpus. In our experiments, the corrupted sentence pairs are generated by the following procedures.

- Delete a token with a probability of 10%.
- Add a token with a probability of 10%.
- Replace a word with a randomly picked word from the vocabulary with a probability of 10%.
- Shuffle the words by adding a normal distribution bias to the positions of the words and re-sort the words by the rectified positions with a standard deviation 0.5.

With a large amount of the artificial training data, the sequence to sequence model learns to reconstruct the input sentence, by trusting most of the input tokens but not always. A sentence pair generated by the corruption process is a GEC sentence pair to some degree, since both of them are translating a not "perfect" sentence to a "perfect" sentence by deleting, adding, replacing or shuffling some tokens.

3.2 Pre-training Decoder

In nature language processing (NLP), pre-training part of the model also improves many tasks' performance. Word2Vec and GloVe (Pennington et al., 2014; Mikolov et al., 2013) pre-trained word embeddings. CoVe (McCann et al., 2017) pre-trained a encoder. ELMo (Peters et al., 2018) pre-trained a deep bidirectional architecture, and etc. All of them are shown to be effective in many NLP tasks.

Following (Ramachandran et al., 2016; Junczys-Dowmunt et al., 2018), we experiment with pre-training the decoder of the copy-augmented sequence-to-sequence architecture as a typical language model. We initialize the decoder of the GEC model with the pre-trained parameters, while initializing the other parameters randomly. Since we use the tied word embeddings between encoder and decoder, most parameters of the model are pre-trained, except for those of the encoder, the encoder-decoder's attention and the copy attention.

4 Multi-Task Learning

The Multi-Task Learning (MTL) solves problems by jointly training multiple related tasks, and has shown its advantages in many tasks, ranging from computer vision (Zhang et al., 2014; Dai et al., 2016) to NLP (Collobert and Weston, 2008; Søgaard and Goldberg, 2016). In this paper, we explore two different tasks for GEC to improve the performance.

4.1 Token-level Labeling Task

We propose a token-level labeling task for the source sentence, and assign each token in the source sentence a label indicating whether this token is right/wrong.

Assuming that each source token x_i can be aligned with a target token y_j, we define that the source token is right if $x_i = y_j$, and wrong otherwise. Each token's label is predicted by passing the final state h_i^{src} of the encoder through a softmax after an affine transformation, as shown in Equation 10.

$$p(label_i|x_{1...N}) = softmax(W^T h_i^{src}) \quad (10)$$

This token-level labeling task explicitly augment the input tokens' correctness to the encoder, which can later be used by the decoder.

4.2 Sentence-level Copying Task

The primary motivation behind the sentence-level copying task is to make the model do more copying when the input sentence looks entirely correct.

During training, we send equal number of sampled correct sentence pairs and the edited sentence pairs to the model. When inputting the right sentences, we remove the decoder's attention over the outputs of the encoder. Without the encoder-decoder attention, the generating work gets hard. As a result, the copying part of the model will be boosted for the correct sentences.

5 Evaluations

5.1 Datasets

As previous studies, we use the public NUCLE (Dahlmeier et al., 2013), Lang-8 (Tajiri et al., 2012) and FCE (Yannakoudakis et al., 2011) corpus as our parrallel training data. The unlabeled dataset we use is the well-known One Billion Word Benchmark (Chelba et al., 2013). We choose the test set of CoNLL-2014 shared task as our test set and CoNLL-2013 test data set (Dahlmeier et al., 2013) as our development benchmark. For the CoNLL data sets, the Max-Match (M^2) scores (Dahlmeier and Ng, 2012) were reported, and for the JFLEG (Napoles et al.,

Corpus	**Sent.**	**Public**	**Type**
Lang-8	1,097,274	Yes	Labeled
NUCLE	57,119	Yes	Labeled
FCE	32,073	Yes	Labeled
One-Billion	30,178,573	Yes	Unlabeled

Table 2: Training Corpus

Corpus	**Sent.**	**Annot.**	**Metric**
CoNLL-2013	1,381	1	M^2
CoNLL-2014	1,312	2	M^2
JFLEG	747	4	GLEU

Table 3: Evaluation Corpus

2017) test set, the GLEU metric (Sakaguchi et al., 2016) were reported.

To make our results comparable to state-of-the-art results in the field of GEC, we limit our training data strictly to public resources. Table 2 and Table 3 list all the data sets that we use in this paper.

We build a statistical-based spell error correction system and correct the spell errors in our training data. Following (Ge et al., 2018; Junczys-Dowmunt et al., 2018; Chollampatt and Ng, 2018) and etc., we apply spell correction before evaluation for our dev/test datasets. A 50,000-word dictionary is extracted from the spell-corrected Lang-8 data corpus. Like previous works, we remove the unchanged sentence pairs in the Lang-8 corpus before training.

5.2 Model and Training Settings

In this paper, we use the Transformer implementation in the public FAIR Sequence-to-Sequence Toolkit [1] (Gehring et al., 2017) codebase.

For the transformer model, we use token embeddings and hidden size of dimension 512, and the encoder and decoder have 6 layers and 8 attention heads. For the inner layer in the position-wise feed-forward network, we use 4096. Similar to previous models we set the dropout to 0.2. A 50,000 vocabulary for the input and output tokens are collected from the training data. In total, this model has 97M parameters.

Models are optimized with Nesterovs Accelerated Gradient (Nesterov, 1983). We set the learning rate with 0.002, the weight decay 0.5, the patience 0, the momentum 0.99 and minimum learning rate 10-4. During training, we evaluate the performance on the development set for every epoch.

We also use edit-weighted MLE objective as (Junczys-Dowmunt et al., 2018), by scaling the loss of the changed words with a balancing factor Λ.

Almost the same architecture and hyper-parameters are used when pre-training using unlabeled data, except the Λ parameter for edit-weighted loss. We set $\Lambda = 3$ when we train the denoising auto-encoder, and set $\Lambda \in [1, 1.8]$ when we train GEC models.

During decoding, we use a beam-size of 12 and normalize model scores by length. We do not use reranking when evaluating the CoNLL-2014 data sets. But we rerank the top 12 hypothesizes using the language model trained on Common Crawl (Junczys-Dowmunt and Grundkiewicz, 2016) for the JFLEG test sets.

5.3 Experimental Results

We compare our results with the well-known GEC systems, as shown in Table 4. Rule, classification, statistical machine translation (SMT), and neural machine translation (NMT) based systems were built for the GEC task. We list the well-known models on the top section of Table 4 and our results in the middle. Almost all the previous systems reranked their top 12 results using a big language model and some of them used partially pre-trained parameters, which improve their results by 1.5 to 5 $F_{0.5}$ score. Our copy-augmented architecture achieve a 56.42 $F_{0.5}$ score on the CoNLL-2014 dataset and outperforms all the previous architectures even without reranking or pre-training.

Combined with denoising auto-encoders and multi-tasks, our model achieve a 61.15 $F_{0.5}$ score on the CoNLL-2014 data set. This result exceeds the previous state-of-the-art system +4.9 $F_{0.5}$ points.

In the bottom section of Table 4, we list the results of (Ge et al., 2018). No direct comparison can be made between us, because they used the non-public Cambridge Learner Corpus (CLC) (Nicholls, 2003) and their own collected non-public Lang-8 corpus, making their labeled training data set 3.6 times larger than ours. Even so, our results on the CoNLL 2014 test data set and JFLEG test data set are very close to theirs.

In Table 4, "SMT (with LM)" refers to (Junczys-Dowmunt and Grundkiewicz, 2014);

---

[1] https://github.com/pytorch/fairseq

Model	**Year**	**CoNLL-14**			**JFELEG**	**Dict**
		Pre.	**Rec.**	$F_{0.5}$	**GLEU**	
SMT (with LM)	2014	41.72	22.00	35.38	-	word
SMT Rule-Based Hybird (with LM)	2014	39.71	30.10	37.33	-	word
SMT Classification Hybird (with LM)	2016	60.17	25.64	47.40	-	word
Neural Hybird MT (with LM)	2017	-	-	45.15	53.41	char/word
CNN + EO (4 ens. with LM)	2018	65.49	33.14	54.79	57.47	bpe
Transformer + MIMs (4 ens. with LM)	2018	63.00	38.90	56.10	59.90	bpe
NMT SMT Hybrid (4 ens. with LM)	2018	66.77	34.49	56.25	61.50	bpe
Our Model						
Copy-augmented Model (4 ens.)	-	68.48	33.10	**56.42**	59.48*	word
+ DA, Multi-tasks (4 ens.)	-	71.57	38.65	**61.15**	61.00*	word
Model Trained with Large Non-public Training Data						
CNN + FB Learning (4 ens. with LM)	2018	74.12	36.30	61.34	61.41	bpe

Table 4: Comparison of GEC systems on CoNLL-2014 and JFLEG test set. The M^2 score for CoNLL-2014 test dataset and the GLEU for the JFLEG test set are reported. DA refers to the "Denoising Auto-encoder". (with LM) refers to the usage of an extra language model. (4 ens.) refers to the ensemble decoding of 4 independently trained models. We re-rank the results of the top 12 hypothesizes for the JFLEG test set with an extra language model and marked them with *.

"SMT Rule-Based Hybird" refers to (Felice et al., 2014); "SMT Classification Hybird" refers to (Rozovskaya and Roth, 2016); "Neural Hybird MT" refers to (Ji et al., 2017); "CNN + EO" refers to (Chollampatt and Ng, 2018) and "EO" means rerank with edit-operation features; "Transformer + MIMs" refers to (Junczys-Dowmunt et al., 2018) and "MIMs" means model indepent methods; "NMT SMT Hybrid" refers to (Grundkiewicz and Junczys-Dowmunt, 2018); "CNN + FB Learning" refers to (Ge et al., 2018).

5.4 Ablation Study

5.4.1 Copying Ablation Results

In this section, we compare the Transformer architecture's results with and without copying mechanism on the GEC task. As illustrated in Table 5, copy-augmented model increases the $F_{0.5}$ score from 48.07 to 54.67, with a +6.6 absolute increase. Most of the improvements come from the words that are out of the fixed vocabulary, which will be predicted as a UNK word in the base model but will be copied as the word itself in the copy-augmented model.

Copying is generally known as good at handling the UNK words. To verify if copying is more than copying UNK words, we do experiments by ignoring all UNK edits. From Table 5, we can see that even ignoring the UNK benefits, the copy-augmented model is still 1.62 $F_{0.5}$ points higher than the baseline model, and most of the benefit comes from the increased recall.

5.4.2 Pre-training Ablation Results

From Table 5, we can observe that by partially pre-training the decoder, the $F_{0.5}$ score is improved from 54.67 to 57.21 (+2.54). It is an evident improvment compared to the un-pre-trained ones. However, the denoising auto-encoder improves the single model from 54.67 to 58.8 (+4.13). We can also see that both the precision and recall are improved after pre-training.

To further investigate how good the pre-trained parameters are, we show the results of the early stage with and without the denoising auto-encoder's pre-trained parameters in Table 6. The results show, if we finetune the model for 1 epoch with the labeled training data, the pre-trained model beats the un-pretrained one with a big gap (48.89 vs 17.19). Even without finetune, the pre-trained model can get a $F_{0.5}$ score of 31.33. This proves that pre-training gives the models much better initial parameters than the randomly picked ones.

5.4.3 Sentence-level Copying Task Ablation Results

We add the sentence-level copying task to encourage the model outputs no edits when we input a correct sentence. To verify this, we create a correct sentence set by sampling 500 sentences from

Model	**Pre.**	**Rec.**	$F_{0.5}$	**Imp.**
Transformer	55.96	30.73	48.07	-
+ Copying	65.23	33.18	**54.67**	+6.60
Ignoring UNK words as edits				
Transformer	65.26	30.63	53.23	-
+ Copying	65.54	33.18	54.85	+1.62
+ Pre-training				
Copy-Augmented Transformer	65.23	33.18	54.67	-
+ Pre-training Decoder (partially pre-trained)	68.02	34.98	57.21	+2.54
+ Denosing Auto-encoder (fully pre-trained)	68.97	36.98	**58.80**	+4.13
+ Multi-tasks				
Copy-Augmented Transformer	67.74	40.62	**59.76**	-

Table 5: Single Model Ablation Study on CoNLL 2014 Test Data Set.

Finetune	**Pre.**	**Rec.**	$F_{0.5}$
with the denoising auto-encoder			
no finetune	36.61	19.87	31.33
finetune 1 epoch	68.58	22.76	48.89
without the denoising auto-encoder			
finetune 1 epoch	32.55	05.96	17.19

Table 6: Denoising Auto-encoder's Results on CoNLL-2014 Test Data Set.

Wikipedia. Also, we generate an error sentence set by sampling 500 sentences from CoNLL-2013 test data set, which is an error-annotated dataset. Then we calculate the average value of the balance factor α^{copy} of the two sets.

Before we add the sentence-level copying task, the α^{copy} is 0.44/0.45 for the correct and error sentence sets. After adding the sentence-level copying task, the value changed to 0.81/0.57. This means that 81% of the final score comes from copying on the correct sentence set, while only 57% on the error sentence set. By adding the sentence-level copying task, models learn to distinguish correct sentences and error sentences.

5.5 Attention Visualization

To analyze how copying and generating divide their work. We visualized the copying attention alignment and the encoder-decoder attention alignment in Figure 2. In Figure 2(a), copying focus their weights on the next word in good order, while in Figure 2(b), generating moves its attention more on the other words, e.g., the nearby words, and the end of the sentence. As explained in (Raganato et al., 2018), this means that the generating part tries to find long dependencies and attend more on global information.

By separating the copying work from the generation work, the generation part of the model can focus more on the "creative" works.

Error Type	**%**	**Recall**
Article Or Determiner	14.31%	44.54%
Wrong Collocation/Idiom	12.75%	**10.38%**
Spelling, Punctuation, etc.	12.47%	45.66%
Preposition	10.38%	49.03%
Noun number	9.38%	**72.65%**
Verb Tense	5.41%	28.15%
Subject-Verb Agreement	4.93%	**61.79%**
Verb form	4.69%	57.26%
Redundancy	4.65%	25.86%
Others	20.99%	23.28%

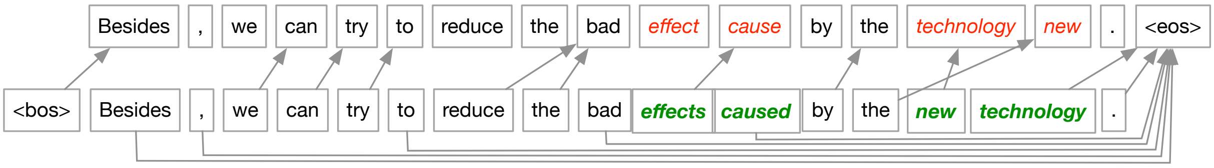
Table 7: Recall on Different Error Types. % is the percentage of this error type in the test data set. Recall is the percentage of the fixed errors in each error type.

6 Discussion

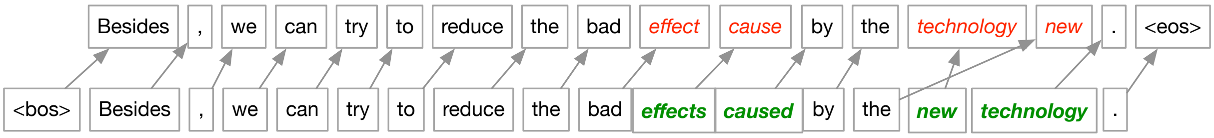
6.1 Recall on Different Error Types

Automatic grammatical error correction is a complicated task since there are different kinds of errors and various correction ways. In this section, we analyze our systems' performance on different grammatical error types. (Ng et al., 2014) labeled CoNLL-2014 test set with 28 error types, and we list the recall percentage on the top 9 error types. We summarize the other 19 types in the last line of the table.

Our approach recalls 72.65% errors on the "Noun number" type and 61.79% on the "Subject-



(a) Copy Alignment



(b) Encoder-Decoder Atttention Alignment

Figure 2: An example of the different behaviors between the copy and encoder-decoder attention. In each figure, the above line is the source sentence, where the error words are in italic. The bottom line is the corrected sentence, where the corrected words are in bold italic. The arrow means which source token the copy and encoder-decoder attention mainly focus on, when predicting the current word. "$\langle bos \rangle$" refers to the begin of the sentence and "$\langle eos \rangle$" refers to the end of the sentence.

Verb Agreement" type. However, only 10.38% errors are recalled on the "Wrong Collocation/Idiom" type.

Computers are good at the definite and mechanical errors, but still have a big gap with humans on the error types that are subjective and with cultural characteristics.

7 Related Work

Early published works in GEC develop specific classifiers for different error types and then use them to build hybrid systems. Later, leveraging the progress of statistical machine translation(SMT) and large-scale error corrected data, GEC systems are further improved treated as a translation problem. SMT systems can remember phrase-based correction pairs, but they are hard to generalize beyond what was seen in training. The CoNLL-14 shared task overview paper (Ng et al., 2014) provides a comparative evaluation of approaches. (Rozovskaya and Roth, 2016) detailed classification and machine translation approaches to grammatical error correction problems, and combined the strengths for both methods.

Recently, neural machine translation approaches have been shown to be very powerful. (Yannakoudakis et al., 2017) developed a neural sequence-labeling model for error detection to calculate the probability of each token in a sentence as being correct or incorrect, and then use the error detecting model's result as a feature to re-rank the N best hypotheses. (Ji et al., 2017) proposed a hybrid neural model incorporating both the word and character-level information. (Chollampatt and Ng, 2018) used a multilayer convolutional encoder-decoder neural network and outperforms all prior neural and statistical based systems on this task. (Junczys-Dowmunt et al., 2018) tried deep RNN (Barone et al., 2017) and transformer (Vaswani et al., 2017) encoder-decoder models and got a higher result by using transformer and a set of model-independent methods for neural GEC.

The state-of-the-art system on GEC task is achieved by (Ge et al., 2018), which are based on the sequence-to-sequence framework and fluency boost learning and inference mechanism. However, the usage of the non-public CLC corpus (Nicholls, 2003) and self-collected non-public error-corrected sentence pairs from Lang-8 made their training data 3.6 times larger than the others and their results hard to compare.

8 Conclusions

We present a copy-augmented architecture for GEC, by considering the characteristics of this problem. Firstly, we propose an enhanced copy-augmented architecture, which improves the sequence-to-sequence model's ability by directly copying the unchanged words and out-of-vocabulary words from the source input tokens. Secondly, we fully pre-train the copy-augmented architecture using large-scale unlabeled data, leveraging denoising auto-encoders.

Thirdly, we introduce two auxiliary tasks for multi-task learning. Finally, we outperform the state-of-the-art automatic grammatical error correction system by a large margin. However, due to the complexity of the GEC problem, there is still a long way to go to make the automatic GEC systems as reliable as humans.

References

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. *arXiv preprint arXiv:1707.07631*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 22–31.

Jifeng Dai, Kaiming He, and Jian Sun. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv preprint arXiv:1804.05945*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. *arXiv preprint arXiv:1707.02026*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv preprint arXiv:1804.05940*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

Yurii E Nesterov. 1983. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alessandro Raganato, Jörg Tiedemann, et al. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2205–2215.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association of Computational Linguistics*, 4(1):169–182.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 198–202. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Helen Yannakoudakis, Marek Rei, Øistein E Andersen, and Zheng Yuan. 2017. Neural sequence-labelling models for grammatical error correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2795–2806.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer.

Topic-Guided Variational Autoencoders for Text Generation

Wenlin Wang[1], **Zhe Gan**[2], **Hongteng Xu**[1,3], **Ruiyi Zhang**[1], **Guoyin Wang**[1],
Dinghan Shen[1], **Changyou Chen**[4], **Lawrence Carin**[1]
[1]Duke University, [2]Microsoft Dynamics 365 AI Research,
[3]Infinia ML, Inc, [4]University at Buffalo
`{wenlin.wang, ruiyi.zhang, gw60, ds448, lcarin}@duke.edu`
`zhe.gan@microsoft.com, hongteng.xu@infiniaml.com`
`changyou@buffalo.edu`

Abstract

We propose a topic-guided variational autoencoder (TGVAE) model for text generation. Distinct from existing variational autoencoder (VAE) based approaches, which assume a simple Gaussian prior for the latent code, our model specifies the prior as a Gaussian mixture model (GMM) parametrized by a neural topic module. Each mixture component corresponds to a latent topic, which provides guidance to generate sentences under the topic. The neural topic module and the VAE-based neural sequence module in our model are learned jointly. In particular, a sequence of invertible Householder transformations is applied to endow the approximate posterior of the latent code with high flexibility during model inference. Experimental results show that our TGVAE outperforms alternative approaches on both unconditional and conditional text generation, which can generate semantically-meaningful sentences with various topics.

1 Introduction

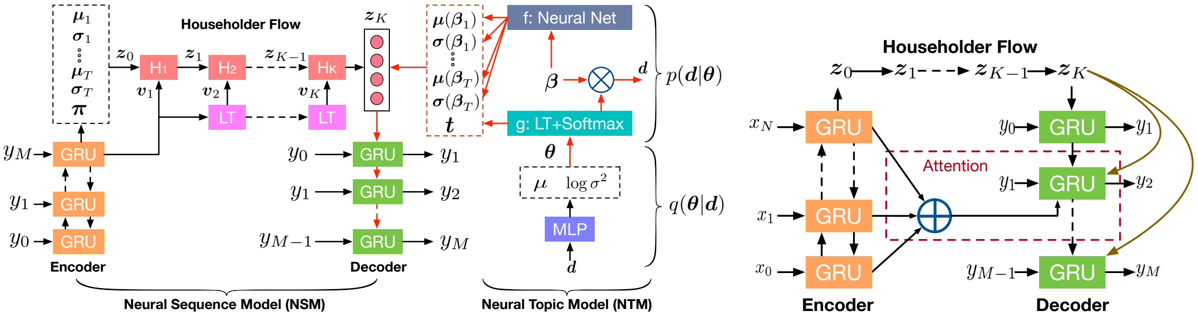
Text generation plays an important role in various natural language processing (NLP) applications, such as machine translation (Cho et al., 2014; Sutskever et al., 2014), dialogue generation (Li et al., 2017a), and text summarization (Nallapati et al., 2016; Rush et al., 2015). As a competitive solution to this task, the variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014) has been widely used in text-generation systems (Bowman et al., 2015; Hu et al., 2017; Serban et al., 2017). In particular, VAE defines a generative model that propagates latent codes drawn from a simple prior through a decoder to manifest data samples. The generative model is further augmented with an inference network, that feeds observed data samples through an encoder to yield a distribution on the corresponding latent codes.

Compared with other potential methods, *e.g.*, those based on generative adversarial networks (GANs) (Yu et al., 2017; Guo et al., 2017; Zhang et al., 2017b, 2018; Chen et al., 2018), VAE is of particular interest when one desires not only text generation, but also the capacity to infer meaningful latent codes from text. Ideally, semantically-meaningful latent codes can provide high-level guidance while generating sentences. For example, when generating text, the vocabulary could potentially be narrowed down if the input latent code corresponds to a certain topic (*e.g.*, the word "military" is unlikely to appear in a sports-related document).

However, in practice this desirable property is not fully achieved by existing VAE-based text generative models, because of the following two challenges. First, the sentences in documents may associate with different semantic information (*e.g.*, topic, sentiment, etc.) while the latent codes of existing VAE-based text generative models often employ a simple Gaussian prior, which cannot indicate the semantic structure among sentences and may reduce the generative power of the decoder. Although some variants of VAE try to impose some structure on the latent codes (Jiang et al., 2016; Dilokthanakul et al., 2016), they are often designed with pre-defined parameter settings without incorporating semantic meanings into the latent codes, which may lead to over-regularization (Dilokthanakul et al., 2016).

The second issue associated with VAE-based text generation is "posterior collapse," first identified in Bowman et al. (2015). With a strong auto-regressive decoder network (*e.g.*, LSTM), the model tends to ignore the information from the latent code and merely depends on previous generated tokens for prediction. Several strategies are proposed to mitigate this problem, including making the decoder network less auto-regressive (*i.e.*,

Proceedings of NAACL-HLT 2019, pages 166–177
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) The scheme of our method.

(b) The extension to text summarization.

Figure 1: Illustration of the proposed Topic-Guided Variational Autoencoder (TGVAE) for text generation. (a) For generation (the red arrows), the topics inferred from a neural topic model are used to guide a Gaussian mixture prior of the latent code, which is further fed into the decoder to generate a sentence. For inference (the black arrows), the sentence is encoded into a vector and then propagated through the Householder flow to obtain the approximate posterior. (b) An attention module is further added for text summarization. The same neural topic model is also applied, but omitted here for simplicity of illustration. "LT" denotes a linear transformation.

using less conditional information while generating each word) (Yang et al., 2017; Shen et al., 2017a), or bridging the amortization gap (between the log-likelihood and the ELBO) using semi-amortized inference networks (Kim et al., 2018). However, these methods mitigate the issue by weakening the conditional dependency on the decoder, which may fail to generate high-quality continuous sentences.

To overcome the two problems mentioned above, we propose a topic-guided variational autoencoder (TGVAE) model, permitting text generation with designated topic guidance. As illustrated in Figure 1(a), TGVAE specifies a Gaussian mixture model (GMM) as the prior of the latent code, where each mixture component corresponds to a topic. The GMM is learnable based on a neural topic model — the mean and diagonal covariance of each mixture component is parameterized by the corresponding topic. Accordingly, the degree to which each component of the GMM is used to generate the latent code and the corresponding sentence is tied to the usage of the topics. In the inference phase, we initialize the latent code from a GMM generated via the encoder, and apply the invertiable Householder transformation (Bischof and Sun, 1994; Sun and Bischof, 1995) to derive the latent code with high flexibility and low complexity.

As shown in Figure 1(b), besides unconditional text generation, the proposed model can be extended for conditional text generation, *i.e.*, abstractive text summarization (Nallapati et al., 2016) with an attention module. By injecting the topics learned by our model (semantic information), we are able to make better use of the source document and improve a sequence-to-sequence summarization model (Sutskever et al., 2014).

We highlight the contributions of our model as follows: (*i*) A new Topic-Guided VAE (TGVAE) model is proposed for text generation with designated topic guidance. (*ii*) For the model inference, Householder flow is introduced to transform a relatively simple mixture distribution into an arbitrarily flexible approximate posterior, achieving powerful approximate posterior inference. (*iii*) Experiments for both unconditional and conditional text generation demonstrate the effectiveness of the proposed approach.

2 Model

The proposed TGVAE, as illustrated in Figure 1(a), consists of two modules: a neural topic model (NTM) and a neural sequence model (NSM). The NTM aims to capture long-range semantic meaning across the document, while the NSM is designed to generate a sentence with designated topic guidance.

2.1 Neural Topic Model

Let $\boldsymbol{d} \in \mathbb{Z}_+^D$ denote the bag-of-words representation of a document, with $\mathbb{Z}_+$ denoting non-negative integers. D is the vocabulary size, and each element of $\boldsymbol{d}$ reflects a count of the number of times the corresponding word occurs in the document. Let a_n represent the topic assignment for word w_n. Following Miao et al. (2017), a Gaussian random vector is passed through a softmax function to parameterize the multinomial document topic distributions. Specifically, the generative process of the NTM is

$$\boldsymbol{\theta} \sim \mathcal{N}(0, \mathbf{I}), \quad \boldsymbol{t} = g(\boldsymbol{\theta}), \tag{1}$$
$$a_n \sim \text{Discrete}(\boldsymbol{t}), \quad w_n \sim \text{Discrete}(\boldsymbol{\beta}_{a_n}),$$

where $\mathcal{N}(0, \mathbf{I})$ is an isotropic Gaussian distribution, $g(\cdot)$ is a transformation function that maps sample $\boldsymbol{\theta}$ to the topic embedding $\boldsymbol{t}$, defined here as $g(\boldsymbol{\theta}) = \text{softmax}(\hat{\mathbf{W}}\boldsymbol{\theta} + \hat{\boldsymbol{b}})$, where $\hat{\mathbf{W}}$ and $\hat{\boldsymbol{b}}$ are trainable parameters; $\boldsymbol{\beta}_{a_n}$ represents the distribution over words for topic a_n; $n \in [1, N_d]$, and N_d is the number of words in the document. The marginal likelihood for document $\boldsymbol{d}$ is:

$$\begin{aligned} p(\boldsymbol{d}|\boldsymbol{\beta}) &= \int_{\boldsymbol{t}} p(\boldsymbol{t}) \prod\nolimits_n \sum\nolimits_{a_n} p(w_n|\boldsymbol{\beta}_{a_n}) p(a_n|\boldsymbol{t}) d\boldsymbol{t} \quad (2) \\ &= \int_{\boldsymbol{t}} p(\boldsymbol{t}) \prod\nolimits_n p(w_n|\boldsymbol{\beta}, \boldsymbol{t}) d\boldsymbol{t} \\ &= \int_{\boldsymbol{t}} p(\boldsymbol{t}) p(\boldsymbol{d}|\boldsymbol{\beta}, \boldsymbol{t}) d\boldsymbol{t} = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{d}|\boldsymbol{\beta}, \boldsymbol{\theta}) d\boldsymbol{\theta}. \end{aligned}$$

The second equation in (2) holds because we can marginalize out the sampled topic words a_n by

$$p(w_n|\boldsymbol{\beta}, \boldsymbol{t}) = \sum\nolimits_{a_n} p(w_n|\boldsymbol{\beta}_{a_n}) p(a_n|\boldsymbol{t}) = \boldsymbol{\beta}\boldsymbol{t}, \tag{3}$$

where $\boldsymbol{\beta} = \{\boldsymbol{\beta}_i\}_{i=1}^T$ are trainable parameters of the decoder; T is the number of topics and each $\boldsymbol{\beta}_i \in \mathbb{R}^D$ is a topic distribution over words (all elements of $\boldsymbol{\beta}_i$ are nonnegative, and sum to one).

2.2 Neural Sequence Model

Our neural sequence model for text generation is built upon the VAE proposed in Bowman et al. (2015). Specifically, a continuous latent code $\boldsymbol{z}$ is first generated from some prior distribution $p(\boldsymbol{z})$, based on which the text sequence $\boldsymbol{y}$ is then generated from a conditional distribution $p(\boldsymbol{y}|\boldsymbol{z})$ parameterized by a neural network (often called the decoder). Since the model incorporates a latent variable $\boldsymbol{z}$ that modulates the entire generation of the sentence, it should be able to capture the high-level source of variation in the data.

Topic-Guided Gaussian Mixture Prior The aforementioned intuition is hard to be captured by a standard VAE, simply imposing a Gaussian prior on top of $\boldsymbol{z}$, since the semantic information associated with a document intrinsically contains different subgroups (such as topics, sentiment, etc.). In our model, we consider incorporating the topic information into latent variables. Our model assumes each $\boldsymbol{z}$ is drawn from a *topic-dependent* GMM, that is,

$$\begin{aligned} p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{t}) &= \sum\nolimits_{i=1}^T t_i \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\beta}_i), \boldsymbol{\sigma}^2(\boldsymbol{\beta}_i)) \\ \boldsymbol{\mu}(\boldsymbol{\beta}_i) &= f_\mu(\boldsymbol{\beta}_i) \\ \boldsymbol{\sigma}^2(\boldsymbol{\beta}_i) &= \text{diag}(\exp(f_\sigma(\boldsymbol{\beta}_i))), \end{aligned} \tag{4}$$

where t_i is the usage of topic i in a document and $\boldsymbol{\beta}_i$ is the i-th topic distribution over words. Both of them are inherited from the NTM discussed above. Both $f_\mu(\cdot)$ and $f_\sigma(\cdot)$ are implemented as feedforward neural networks, with trainable parameters $\mathbf{W}_\mu$ and $\mathbf{W}_\sigma$, respectively. Compared with a normal GMM prior that sets each mixture component to be $\mathcal{N}(0, \mathbf{I})$, the proposed topic guided GMM prior provides semantic meaning for each mixture component, and hence makes the model more interpretable and controllable for text generation.

Decoder The likelihood of a word sequence $\boldsymbol{y} = \{y_m\}_{m=1}^M$ conditioned on the latent code $\boldsymbol{z}$ is defined as:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{z}) &= p(y_1|\boldsymbol{z}) \prod\nolimits_{m=2}^M p(y_m|y_{1:m-1}, \boldsymbol{z}) \\ &= p(y_1|\boldsymbol{z}) \prod\nolimits_{m=2}^M p(y_m|\boldsymbol{h}_m), \end{aligned} \tag{5}$$

where the conditional probability of each word y_m given all the previous words $y_{1:m-1}$ and the latent code $\boldsymbol{z}$ is defined through the hidden state $\boldsymbol{h}_m$: $\boldsymbol{h}_m = f(\boldsymbol{h}_{m-1}, y_{m-1}, \boldsymbol{z})$, where the function $f(\cdot)$ is implemented as a Gated Recurrent Unit (GRU) cell (Cho et al., 2014) in our experiments.

3 Inference

The proposed model (see Figure 1(a)) takes the bag-of-words as input and embeds a document into a topic vector. The topic vector is then used to reconstruct the bag-of-words input, and the learned topic distribution over words is used to model a topic-dependent prior to generate a sentence in the VAE setup. Specifically, the joint marginal likelihood can be written as:

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{d}|\boldsymbol{\beta}) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} & p(\boldsymbol{\theta}) p(\boldsymbol{d}|\boldsymbol{\beta}, \boldsymbol{\theta}) \\ & \cdot p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\theta}) p(\boldsymbol{y}|\boldsymbol{z}) \, d\boldsymbol{\theta} d\boldsymbol{z}. \end{aligned} \tag{6}$$

Since direct optimization of (6) is intractable, auto-encoding variational Bayes is employed (Kingma and Welling, 2013). Denote $q(\boldsymbol{\theta}|\boldsymbol{d})$ and $q(\boldsymbol{z}|\boldsymbol{y})$ as the variational distributions for $\boldsymbol{\theta}$ and $\boldsymbol{z}$, respectively. The variational objective function, also called the evidence lower bound (ELBO), is constructed as

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})}\left[\log p(\boldsymbol{d}|\boldsymbol{\beta},\boldsymbol{\theta})\right] - \mathrm{KL}\left(q(\boldsymbol{\theta}|\boldsymbol{d})||p(\boldsymbol{\theta})\right)}_{\text{neural topic model},\mathcal{L}_t} + \quad (7)$$

$$\underbrace{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{y})}\left[\log p(\boldsymbol{y}|\boldsymbol{z})\right] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})}\left[\mathrm{KL}\left(q(\boldsymbol{z}|\boldsymbol{y})||p(\boldsymbol{z}|\boldsymbol{\beta},\boldsymbol{\theta})\right)\right]}_{\text{neural sequence model},\mathcal{L}_s}.$$

By assuming

$$q(\boldsymbol{\theta}|\boldsymbol{d}) = \mathcal{N}(\boldsymbol{\theta}|g_\mu(\boldsymbol{d}), \mathrm{diag}(\exp\left(g_\sigma(\boldsymbol{d})\right))),$$

where both $g_\mu(\cdot)$ and $g_\sigma(\cdot)$ are implemented as feed-forward neural networks, the reparameterization trick (Kingma and Welling, 2013) can be applied directly to build an unbiased and low-variance gradient estimator for the $\mathcal{L}_t$ term in (7). Below, we discuss in detail how to approximate the $\mathcal{L}_s$ term in (7) and infer an arbitrarily complex posterior for $\boldsymbol{z}$. Note that $\boldsymbol{z}$ is henceforth represented as $\boldsymbol{z}_K$ in preparation for the introduction of Householder flows.

3.1 Householder Flow for Approximate Posterior

Householder flow (Zhang et al., 2017a; Tomczak and Welling, 2016) is a volume-preserving normalizing flow (Rezende and Mohamed, 2015), capable of constructing an arbitrarily complex posterior $q_K(\boldsymbol{z}_K|\boldsymbol{y})$ from an initial random variable $\boldsymbol{z}_0$ with distribution q_0, by composing a sequence of invertible mappings, *i.e.*, $\boldsymbol{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\boldsymbol{z}_0)$. By repeatedly applying the chain rule and using the property of Jacobians of invertible functions, $q_K(\boldsymbol{z}_K|\boldsymbol{y})$ is expressed as:

$$\log q_K(\boldsymbol{z}_K|\boldsymbol{y}) = \log q_0(\boldsymbol{z}_0|\boldsymbol{y}) - \sum\nolimits_{k=1}^{K} \log\left|\det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}}\right|, \quad (8)$$

where $|\det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}}|$ is the absolute value of the Jacobian determinant. Therefore, the $\mathcal{L}_s$ term in (7) may be rewritten as

$$\mathbb{E}_{q_0(\boldsymbol{z}_0|\boldsymbol{y})}[\log p(\boldsymbol{y}|\boldsymbol{z}_K)] + \sum\nolimits_{k=1}^{K} \log\left|\det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}}\right|$$
$$-\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})}\left[\mathrm{KL}(q_0(\boldsymbol{z}_0|\boldsymbol{y})||p(\boldsymbol{z}_K|\boldsymbol{\beta},\boldsymbol{\theta}))\right]. \quad (9)$$

Here $q_0(\boldsymbol{z}_0|\boldsymbol{y})$ is also specified as a GMM, *i.e.*, $q_0(\boldsymbol{z}_0|\boldsymbol{y}) = \sum_{i=1}^{T} \pi_i(\boldsymbol{y})\mathcal{N}(\boldsymbol{\mu}_i(\boldsymbol{y}), \boldsymbol{\sigma}_i^2(\boldsymbol{y}))$. As illustrated in Figure 1(a), $\boldsymbol{y}$ is first represented as a hidden vector $\boldsymbol{h}$, by encoding the text sequence with an RNN. Based on this, the mixture probabilities $\boldsymbol{\pi}$, the means and diagonal covariances of all the mixture components are all produced by an encoder network, which is a linear layer with the input $\boldsymbol{h}$. In (9), the first term can be considered as the reconstruction error, while the remaining two terms act as regularizers, the tractability of which is important for the whole framework.

KL Divergence between two GMMs Since both the prior $p(\boldsymbol{z}_K|\boldsymbol{\beta},\boldsymbol{\theta})$ and the initial density $q_0(\boldsymbol{z}_0|\boldsymbol{y})$ for the posterior are GMMs, the calculation of the third term in (9) requires the KL divergence between two GMMs. Though no closed-form solutions exist, the KL divergence has an explicit upper bound (Dilokthanakul et al., 2016), shown in Proposition 1.

Proposition 1. *For any two mixture densities $p = \sum_{i=1}^{n} \pi_i g_i$ and $\hat{p} = \sum_{i=1}^{n} \hat{\pi}_i \hat{g}_i$, their KL divergence is upper-bounded by*

$$\mathrm{KL}\left(p||\hat{p}\right) \leq \mathrm{KL}\left(\pi||\hat{\pi}\right) + \sum\nolimits_{i=1}^{n} \pi_i \mathrm{KL}\left(g_i||\hat{g}_i\right), \quad (10)$$

where equality holds if and only if $\frac{\pi_i g_i}{\sum_{i=1}^{n} \pi_i g_i} = \frac{\hat{\pi} \hat{g}_i}{\sum_{i=1}^{n} \hat{\pi} \hat{g}_i}$.

Proof. With the log-sum inequality

$$\begin{aligned}\mathrm{KL}\left(p||\hat{p}\right) &= \int \left(\sum_i \pi_i g_i\right) \log \frac{\sum_i \pi_i g_i}{\sum_i \hat{\pi} \hat{g}_i} \\ &\leq \int \sum_i \pi_i g_i \log \frac{\pi_i g_i}{\hat{\pi} \hat{g}_i} \\ &= \sum_i \pi_i \log \frac{\pi_i}{\hat{\pi}} + \sum_i \pi_i \int g_i \log \frac{g_i}{\hat{g}_i} \\ &= \mathrm{KL}(\pi||\hat{\pi}) + \sum_i \pi_i \mathrm{KL}(g_i||\hat{g}_i). \end{aligned} \quad (11)$$

Since the KL divergence between two Gaussian distributions has a closed-form expression, the upper bound of the KL divergence between two GMMs can be readily calculated. Accordingly, the third term in (9) is upper bounded as

$$\mathcal{U}_{KL} = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})}\Big[\mathrm{KL}\left(\boldsymbol{\pi}(\boldsymbol{y})||\boldsymbol{t}\right) \quad (12)$$
$$+ \sum\nolimits_{i=1}^{T} \pi_i(\boldsymbol{y})\mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}_i(\boldsymbol{y}), \boldsymbol{\sigma}_i^2(\boldsymbol{y})||\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\beta}_i), \boldsymbol{\sigma}^2(\boldsymbol{\beta}_i)))\right)\Big],$$

where the expectation $\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})}[\cdot]$ can be approximated by a sample from $q(\boldsymbol{\theta}|\boldsymbol{d})$.

Householder Flow Householder flow (Tomczak and Welling, 2016) is a series of Householder transformations, defined as follows. For a given vector $\boldsymbol{z}_{k-1}$, the reflection hyperplane can be defined by a Householder vector $\boldsymbol{v}_t$ that is orthogonal to the hyperplane. The reflection of this point about the hyperplane is

$$\boldsymbol{z}_k = \left(\mathbf{I} - 2\frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{||\boldsymbol{v}_k||^2}\right) \boldsymbol{z}_{k-1} = \mathbf{H}_k \boldsymbol{z}_{k-1}, \quad (13)$$

where $\mathbf{H}_k = \mathbf{I} - 2\frac{\boldsymbol{v}_k\boldsymbol{v}_k^T}{||\boldsymbol{v}_k||^2}$ is called the *Householder matrix*. An important property of the *Householder matrix* is that the absolute value of the Jacobian determinant is equal to 1, therefore $\sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \boldsymbol{z}_{k-1}} \right| = \sum_{k=1}^{K} \log |\det \mathbf{H}_k| = 0$, significantly simplifying the computation of the lower bound in (9). For $k = 1, \ldots, K$, the vector $\boldsymbol{v}_k$ is produced by a linear layer with the input $\boldsymbol{v}_{k-1}$, where $\boldsymbol{v}_0 = \boldsymbol{h}$ is the last hidden vector of the encoder RNN that encodes the sentence $\boldsymbol{y}$.

Finally, by combining (7), (9) and (12), the ELBO can be rewritten as

$$\mathcal{L} \geq \mathcal{L}_t + \mathbb{E}_{q_0(\boldsymbol{z}_0|\boldsymbol{y})}[\log p(\boldsymbol{y}|\boldsymbol{z}_K)] - \mathcal{U}_{KL} \, . \quad (14)$$

3.2 Extension to text summarization

When extending our model to text summarization, we are interested in modeling $p(\boldsymbol{y}, \boldsymbol{d}|\boldsymbol{x})$, where $(\boldsymbol{x}, \boldsymbol{y})$ denotes the document-summary pair, and $\boldsymbol{d}$ denotes the bag-of-words of the input document. The marginal likelihood can be written as $p(\boldsymbol{y}, \boldsymbol{d}|\boldsymbol{x}) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} p(\boldsymbol{\theta}) p(\boldsymbol{d}|\boldsymbol{\beta}, \boldsymbol{\theta}) p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\theta}) p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) \, d\boldsymbol{\theta} d\boldsymbol{z}$. Assume the approximate posterior of $\boldsymbol{z}$ is only dependent on $\boldsymbol{x}$, *i.e.*, $q(\boldsymbol{z}|\boldsymbol{x})$ is proposed as the variational distribution for $\boldsymbol{z}$. The ELBO is then constructed as

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_t + \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})} \left[\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) \right] \\ & - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{d})} \left[\mathrm{KL} \left(q(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\theta}) \right) \right] \, , \quad (15) \end{aligned}$$

where $\mathcal{L}_t$ is the same as used in (7). The main difference when compared with unconditional text generation lies in the usage of $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ and $q(\boldsymbol{z}|\boldsymbol{x})$, illustrated in Figure 1(b). The generation of $\boldsymbol{y}$ given $\boldsymbol{x}$ is not only dependent on a standard Seq2Seq model with attention (Nallapati et al., 2016), but also affected by $\boldsymbol{z}$ (*i.e.*, $\boldsymbol{z}_K$), which provides the high-level topic guidance.

3.3 Diversity Regularizer for NTM

Redundancy in inferred topics is a common issue existing in general topic models. In order to address this, it is straightforward to regularize the row-wise distance between paired topics to diversify the topics. Following Xie et al. (2015); Miao et al. (2017), we apply a topic diversity regularization while carrying out the inference.

Specifically, the distance between a pair of topics is measured by their cosine distance $a(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j) = \arccos\left(\frac{|\boldsymbol{\beta}_i \cdot \boldsymbol{\beta}_j|}{|\boldsymbol{\beta}_i|_2 |\boldsymbol{\beta}_j|_2}\right)$. The mean angle of all pairs of T topics is $\phi = \frac{1}{T^2} \sum_i \sum_j a(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j)$, and the variance is $\nu = \frac{1}{T^2} \sum_i \sum_j (a(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j) - \phi)^2$. Finally, the topic-diversity regularization is defined as $R = \phi - \nu$.

4 Related Work

The VAE was proposed by Kingma and Welling (2013), and since then, it has been applied successfully in a variety of applications (Gregor et al., 2015; Kingma et al., 2014; Chen et al., 2017; Wang et al., 2018b; Shen et al., 2018). Focusing on text generation, the methods in Miao et al. (2017, 2016); Srivastava and Sutton (2017) represent texts as bag-of-words, and Bowman et al. (2015) proposed the usage of an RNN as the encoder and decoder, and found some negative results. In order to improve the performance, different convolutional designs (Semeniuta et al., 2017; Shen et al., 2017a; Yang et al., 2017) have been proposed. A VAE variant was further developed in Hu et al. (2017) to control the sentiment and tense of generated sentences. Additionally, the VAE has also been considered for conditional text generation tasks, including machine translation (Zhang et al., 2016), image captioning (Pu et al., 2016), dialogue generation (Serban et al., 2017; Shen et al., 2017b; Zhao et al., 2017) and text summarization (Li et al., 2017b; Miao and Blunsom, 2016). In particular, distinct from the above works, we propose the usage of a topic-dependent prior to explicitly incorporate topic guidance into the text-generation framework.

The idea of using learned topics to improve NLP tasks has been explored previously, including methods combining topic and neural language models (Ahn et al., 2016; Dieng et al., 2016; Lau et al., 2017; Mikolov and Zweig, 2012; Wang et al., 2017), as well as leveraging topic and word embeddings (Liu et al., 2015; Xu et al., 2018). Distinct from them, we propose the use of topics to guide the prior of a VAE, rather than only the language model (*i.e.*, the decoder in a VAE setup). This provides more flexibility in text modeling and also the ability to infer the posterior on latent codes, which could be useful for visualization and downstream tasks.

Neural abstractive summarization was pioneered in Rush et al. (2015), and it was followed and extended by Chopra et al. (2016). Currently the RNN-based encoder-decoder framework with attention (Nallapati et al., 2016; See et al., 2017) remains popular in this area. Attention models typically work as a keyword detector, which is similar to topic modeling in spirit. This fact motivated us to extend our topic-guided VAE model to text summarization.

5 Experiments

We evaluate our TGVAE on text generation and text summarization, and interpret its improvements both quantitatively and qualitatively.

5.1 Text Generation

Dataset We conduct experiments on three publicly available corpora: APNEWS, IMDB and BNC.[1] APNEWS[2] is a collection of Associated Press news articles from 2009 to 2016. IMDB is a set of movie reviews collected by Maas et al. (2011), and BNC (BNC Consortium, 2007) is the written portion of the British National Corpus, which contains excerpts from journals, books, letters, essays, memoranda, news and other types of text. For the three corpora, we tokenize the words and sentences, lowercase all word tokens, and filter out word tokens that occur less than 10 times. For the topic model, we remove stop words in the documents and exclude the top 0.1% most frequent words and also words that appear less than 100 documents. A summary statistics is provided in Table 1.

Evaluation We first compare the perplexity of our neural sequence model with a variety of baselines. Further, we evaluate BLEU scores on the generated sentences, noted as *test*-BLEU and *self*-BLEU. *test*-BLEU (higher is better) evaluates the quality of generated sentences using a group of real test-set sentences as the reference, and *self*-BLEU (lower is better) mainly measures the diversity of generated samples (Zhu et al., 2018).

Setup For the neural topic model (NTM), we consider a 2-layer feed-forward neural network to model $q(\boldsymbol{\theta}|\boldsymbol{d})$, with 256 hidden units in each layer; ReLU is used as the activation function. The hyper-parameter λ for the neural topic model diversity regularizer is fixed to 0.1 across all the experiments. All the sentences in the paragraph are used to obtain the bag-of-words presentation $\boldsymbol{d}$. The maximum number of words in a paragraph is set to 300. For the neural sequence model (NSM), we use bidirectional-GRU as the encoder and a standard GRU as the decoder. The hidden state of our GRU is fixed to 600 across all the three corpora. For the input sequence, we fix the sequence length to 30. In order to avoid overfitting, dropout with a rate of 0.4 is used in each GRU layer.

Baseline We test the proposed method with different numbers of topics (components in GMM) and different numbers of Householder flows ($i.e.$, K), and compare it with six baselines: (*i*) a standard language model (LM); (*ii*) a standard variational RNN auto-encoder (VAE); (*iii*) a Gaussian prior-based VAE with Householder Flow (VAE+HF); (*iv*) a standard LSTM language model with LDA as additional feature (LDA+LSTM); (*v*) Topic-RNN (Dieng et al., 2016), a joint learning framework which learns a topic model and a language model simultaneously; (*vi*) TDLM (Lau et al., 2017), a joint learning framework which learns a convolutional based topic model and a language model simultaneously.

Results The results in Table 3 show that the models trained with a VAE and its Householder extension does not outperform a well-optimized language model, and the KL term tends to be annealed with the increase of K. In comparison, our TGVAE achieves a lower perplexity upper bound, with a relative larger $\mathcal{U}_{KL}$. We attribute the improvements to our topic guided GMM model design, which provides additional topical clustering information in the latent space; the Householder flow also boosts the posterior inference for our TGVAE. We also observe consistent improvements with the number of topics, which demonstrates the efficiency of our TGVAE.

To verify the generative power of our TGVAE, we generate samples from our *topic-dependent* prior and compare various methods on the BLEU scores in Table 2. With the increase of topic numbers, our TGVAE yields consistently better *self*-BLEU and a boost over *test*-BLEU relative to standard VAE models. We also show a group of sampled sentences drawn from a portion of topics in Table 5. Our TGVAE is able to generate diverse sentences under topic guidance. When generating sentences under a mixture of topics, we draw multiple samples from the GMM and take $\boldsymbol{z}$ as the averaged sample.

Though this paper focuses on generating coherent topic-specific sentences rather than the learned topics themselves, we also evaluate the topic coherence (Lau et al., 2017) to show the rationality of our joint learning framework. We compute topic coher-

---

[1] These three datasets can be downloaded from https://github.com/jhlau/topically-driven-language-model.

[2] https://www.ap.org/en-gb/

Dataset	Vocabulary		Training			Development			Testing		
	LM	TM	# Docs	# Sents	# Tokens	# Docs	# Sents	# Tokens	# Docs	# Sents	# Tokens
APNEWS	$32,400$	$7,790$	$50K$	$0.7M$	$15M$	$2K$	$27.4K$	$0.6M$	$2K$	$26.3K$	$0.6M$
IMDB	$34,256$	$8,713$	$75K$	$0.9M$	$20M$	$12.5K$	$0.2M$	$0.3M$	$12.5K$	$0.2M$	$0.3M$
BNC	$41,370$	$9,741$	$15K$	$0.8M$	$18M$	$1K$	$44K$	$1M$	$1K$	$52K$	$1M$

Table 1: Summary statistics for APNEWS, IMDB and BNC.

Metric	Methods	APNEWS				IMDB				BNC			
		B-2	B-3	B-4	B-5	B-2	B-3	B-4	B-5	B-2	B-3	B-4	B-5
test-BLEU	VAE	0.564	0.278	0.192	0.122	0.597	0.315	0.219	0.147	0.479	0.266	0.169	0.117
	VAE+HF (K=1)	0.566	0.280	0.193	0.124	0.593	0.317	0.218	0.148	0.475	0.268	0.165	0.112
	VAE+HF (K=10)	0.570	0.279	0.195	0.123	0.610	0.322	0.221	0.147	0.483	0.270	0.169	0.110
	TGVAE (K=0, T=10)	0.582	0.320	0.203	0.125	0.627	0.362	0.223	0.159	0.517	0.282	0.181	0.115
	TGVAE (K=1, T=10)	0.581	0.326	0.202	0.124	0.623	0.358	0.224	0.160	0.519	0.282	0.182	0.118
	TGVAE (K=10, T=10)	0.584	0.327	0.202	0.126	0.621	0.357	0.223	0.159	0.518	0.283	0.173	0.119
	TGVAE (K=10, T=30)	0.627	0.335	0.207	0.131	**0.655**	0.369	**0.243**	**0.165**	0.528	**0.291**	0.182	0.119
	TGVAE (K=10, T=50)	**0.629**	**0.340**	**0.210**	**0.132**	0.652	**0.372**	0.239	0.160	**0.535**	0.290	**0.188**	**0.120**
self-BLEU	VAE	0.866	0.531	0.233	-	0.891	0.632	0.275	-	0.851	0.51	0.163	-
	VAE+HF (K=1)	0.865	0.533	0.241	-	0.899	0.641	0.278	-	0.854	0.515	0.163	-
	VAE+HF (K=10)	0.873	0.552	0.219	-	0.902	0.648	0.262	-	0.854	0.520	0.168	-
	TGVAE (K=0, T=10)	0.847	0.499	0.161	-	0.878	0.572	0.234	-	0.832	0.488	0.160	-
	TGVAE (K=1, T=10)	0.847	0.495	0.160	-	0.871	0.571	0.233	-	0.828	0.483	0.150	-
	TGVAE (K=10, T=10)	0.839	0.512	0.172	-	0.889	0.577	0.242	-	0.829	0.488	0.151	-
	TGVAE (K=10, T=30)	0.811	0.478	0.157	-	0.850	0.560	0.231	-	0.806	0.473	0.150	-
	TGVAE (K=10, T=50)	**0.808**	**0.476**	**0.150**	-	**0.842**	**0.559**	**0.227**	-	**0.793**	**0.469**	**0.150**	-

Table 2: *test*-BLEU (higher is better) and *self*-BLEU (lower is better) scores over three corpora.

Methods	APNEWS		IMDB		BNC	
	PPL	KL	PPL	KL	PPL	KL
LM	62.79	–	70.38	–	100.07	–
LDA+LSTM	57.05	–	69.58	–	96.42	–
Topic-RNN	56.77	–	68.74	–	94.66	–
TDLM	53.00	–	63.67	–	91.42	–
VAE	$\leq$75.89	1.78	86.16	2.78	$\leq$105.10	0.13
VAE+HF (K=1)	$\leq$72.99	1.32	$\leq$84.06	1.83	$\leq$105.13	0.31
VAE+HF (K=10)	$\leq$71.60	0.83	$\leq$83.67	1.51	$\leq$104.82	0.17
TGVAE (K=0, T=10)	$\leq$56.12	2.73	$\leq$62.99	3.99	$\leq$92.32	3.40
TGVAE (K=1, T=10)	$\leq$56.08	2.70	$\leq$62.12	3.86	$\leq$91.17	3.12
TGVAE (K=10, T=10)	$\leq$55.77	2.69	$\leq$62.22	3.94	$\leq$91.19	2.99
TGVAE (K=10, T=30)	$\leq$51.27	3.62	$\leq$59.45	4.62	$\leq$88.34	3.82
TGVAE (K=10, T=50)	$\leq$**48.73**	3.55	$\leq$**57.11**	5.02	$\leq$**87.86**	4.57

Table 3: Perplexity and averaged KL scores over three corpora. KL in our TGVAE denotes $\mathcal{U}_{KL}$ in Eqn. (12).

Methods	APNEWS	IMDB	BNC
	T=50	T=50	T=50
LDA (Blei et al., 2003)	0.125	0.084	0.106
TDLM (Lau et al., 2017)	0.149	0.104	0.102
Topic-RNN (Dieng et al., 2016)	0.134	0.103	0.102
TGVAE	**0.157**	**0.105**	**0.113**

Table 4: Topic coherence over APNEWS, IMDB and BNC.

ence using normalized PMI (NPMI). In practice, we average topic coherence over the top 5/10/15/20 topic words. To aggregate topic coherence score, we further average the coherence scores over topics. Results are summarized in Table 4.

5.2 Text Summarization

Dataset We further test our model for text summarization on two popular datasets. First, we follow the same setup as in Rush et al. (2015) and consider the GIGAWORDS corpus[3], which consists of 3.8M training pair samples, 190K validation samples and 1,951 test samples for evaluation. An input-summary pair consists of the first sentence and the headline of the source articles. We also evaluate various models on the DUC-2004 test set[4], which has 500 news articles. Different from GIGAWORDS, each article in DUC-2004 is paired with four expert-generated reference summaries. The length of each summary is limited to 75 bytes.

Evaluation We evaluate the performance of our model with the ROUGE score (Lin, 2004), which counts the number of overlapping content between the generated summaries and the reference summaries, *e.g.*, overlapped n-grams. Following practice, we use F-measures of ROUGE-1 (RF-1), ROUGE-2 (RF-2) and ROUGE-L (RF-L) for GIGAWORDS and Recall measures of ROUGE-1 (RR-1), ROUGE-2 (RR-2) and ROUGE-L (RR-L) for DUC-2004.

Setup We have a similar data tokenization as we

---

[3] https://catalog.ldc.upenn.edu/ldc2012t21

[4] http://duc.nist.gov/duc2004

Data	**Topic**	**Sentences**
APNEWS	education	• the commission has approved a bill that would make state funding available for the city 's new school .
	animal	•the feline did n't survive fence hangars at the lake .
	crime	• the jury found the defense was not a <unk> , <unk> 's ruling and that the state 's highest court has been convicted of first-degree murder .
	weather	• forecasters say they 're still trying to see the national weather service watch for the latest forecast for friday evening .
	lottery	• she hopes the jackpot now exceeds $ 9 million .
	education+law	• an alabama law professor thomas said monday that the state's open court claims it takes an emotional matter about issuing child molesters based on religion.
	animal+medicine	• the study says the animal welfare department and others are not sure to make similar cases to the virus in the zoo.
IMDB	war	• after watching the movie , there is a great documentary about the war in the years of the israeli war .
	children	• the entire animation was great at times as to the readings of disney favorites .
	epsiode	• the show would have warranted for 25 episodes and it does help immediately .
	name	• she steals the other part where norma 's <unk> husband (crawford) (as at his part , sh*t for the road) .
	detective	• holmes shouted just to be as much as the movie 's last scene where there were <unk> pills to nab the <unk> .
	horror + negative	• the movie about a zombie is the worst movie i have ever seen.
	detective + children	• my favorite childhood is that rochester takes the character in jane's way, playing the one with hamlet.
BNC	medical	• here mistaking ' causes ' drugs as the problem although both economically ill patients arising from a local job will be in traumatic dangers .
	education	• he says the sale is given to five students ' award off : out at a laboratory after the three watts of the hours travelling in and chairman store the bank of the <unk> sutcliffe .
	religion	• schoolchildren will either go or back to church in his place every year in the savoy .
	entertainment	• 100 company and special lace with <unk> garland for tea our garden was filmed after a ceremony
	IT	• ibm also has shut all the big macs in the 60mhz ncube , represent on the acquisition and mips unix .
	environment + crime	• the earth's environmental protection agency said that the government was still being shut down by the police.
	education+entertainment	• the school is 55 and hosts one of a musician's theme charities festival.

Table 5: Generated sentences from given topics.

Sample of Summaries
D: a court here thursday sentenced a ##-year-old man to ## years in jail after he admitted pummelling his baby son to death to silence him while watching television . **R**: man who killed baby to hear television better gets ## years. **Seq2Seq**: man sentenced to ## years after the son 's death **Ours**: a court sentenced a man ## years in jail
D: european stock markets advanced strongly thursday on some bargain-hunting and gains by wall street and japanese shares ahead of an expected hike in us interest rates , dealers said **R**: european stocks bounce back UNK UNK with closing levels **Seq2Seq**: european stocks advance ahead of us interest rate hike **Ours**: european stocks rise on bargain-hunting, dealer said friday
D: the democratic people 's republic of korea whitewashed south korea in the women 's team semi-finals at the world table tennis championships here on sunday **R**: dpr korea sails into women 's team final **Seq2Seq**: dpr korea whitewash south korea in women 's team final **Ours**: dpr korea beat south korea in table tennis worlds

Table 6: Example generated summaries on GIGAWORDS. D is the source article, R means the reference summary, Seq2seq represents the summary generated from the Seq2Seq model.

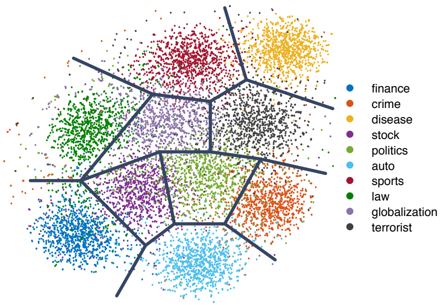


Figure 2: The t-SNE visualization of $1,000$ samples drawn from the learned topic-guided Gaussian mixture prior and they can be best viewed in color.

Methods	GIGAWORDS			DUC-2004		
	RF-1	**RF-2**	**RF-L**	**RR-1**	**RR-2**	**RR-L**
ABS	29.55	11.32	26.42	26.55	7.06	22.05
ABS+	29.78	11.89	26.97	28.18	8.49	23.81
RAS-LSTM	32.55	14.70	30.03	28.97	8.26	24.06
RAS-Elman	33.78	15.97	31.15	27.41	7.69	23.06
lvt2k-lsent	32.67	15.59	30.64	28.35	9.46	24.59
lvt5k-lsent	35.30	16.64	32.62	28.61	9.42	25.24
ASC+FSC	34.17	15.94	31.92	26.73	8.39	23.88
Seq2Seq	34.03	15.93	31.68	28.39	9.26	24.83
Var-Seq2Seq	34.00	15.97	31.85	28.11	9.24	24.86
Var-Seq2Seq-HF (K=1)	34.04	15.98	31.84	28.18	9.27	24.84
Var-Seq2Seq-HF (K=10)	34.22	16.10	32.13	28.78	9.11	24.96
TGVAE (K=0, T=10)	35.34	16.68	32.69	28.99	9.21	24.89
TGVAE (K=1, T=10)	35.35	16.70	32.64	29.02	9.24	24.93
TGVAE (K=10, T=10)	35.40	16.77	32.71	29.07	9.32	25.17
TGVAE (K=10, T=30)	35.59	17.18	32.89	29.38	**9.60**	25.22
TGVAE (K=10, T=50)	**35.63**	**17.27**	**33.02**	**29.65**	9.55	**25.38**

Table 7: Results on Gigawords and DUC-2004.

have in text generation. Additionally, for the vocabulary, we count the frequency of words in both the source article the target summary, and maintain the top 30,000 tokens as the source article and target summary vocabulary. For the NTM, we further remove top 0.3% words and infrequent words to get a topic model vocabulary in size of 8000. For the NTM, we follow the same setup as our text generation. In the NSM, we keep using bidirectional-GRU as the encoder and a standard GRU as the decoder. The hidden state is fixed to 400. An attention mechanism (Bahdanau et al., 2015) is applied in our sequence-to-sequence model.

Baseline We compare our method with the following alternatives: (*i*) a standard sequence-to-sequence model with attention (Bahdanau et al., 2015) (Seq2Seq); (*ii*) a model similar to our TGVAE, but without the usage of the topic-dependent prior and Householder flow (Var-Seq2Seq); and (*iii*) a model similar to our TGVAE, but without the usage of the topic dependent prior (Var-Seq2Seq-

Dataset	**education**	**animal**	**crime**	**weather**	**lottery**	**terrorism**	**law**	**art**	**transportation**	**market**
APNEWS	students	animals	murder	weather	mega	syria	lawsuit	album	airlines	zacks
	education	dogs	first-degree	corecasters	lottery	iran	appeals	music	rail	cents
	schools	zoo	shooting	winds	powerball	militants	justices	film	transit	earnings
	math	bear	sentenced	rain	gambling	afgan	constitutional	songs	bridge	revenue
	teachers	wildlife	gunshot	snow	jackpot	korea	judge	comedy	airport	income
IMDB	**war**	**children**	**epsiode**	**name**	**detective**	**ethic**	**action**	**horror**	**negative**	**japanese**
	aircraft	cinderella	season	crawford	holmes	porn	batman	horror	stupid	miike
	president	musical	episode	stanwyck	poirot	unfunny	king	zombie	horrible	kurosawa
	war	beatles	sandler	gable	christie	sex	chan	werewolf	sucks	sadako
	military	musicals	cartoons	powell	book	gay	li	candyman	waste	anime
	soldiers	disney	jokes	harlow	agatha	erotic	ninja	dracula	scary	takashi
BNC	**medical**	**education**	**religion**	**entertainment**	**IT**	**Law**	**facilities**	**crime**	**sports**	**environment**
	patients	award	church	film	unix	tax	bedrooms	police	cup	nuclear
	gastric	discipline	god	video	corp	coun	hotel	killed	league	emission
	cells	economic	art	album	software	lamont	restaurant	arrested	striker	dioxide
	oesophageal	research	theological	comedy	server	council	rooms	soldiers	season	pollution
	mucosa	institution	religious	movie	ibm	pensioners	situated	murder	goal	warming
GIGAWORDS	**terrorist**	**crime**	**finance**	**sports**	**law**	**stock**	**auto**	**disease**	**globalization**	**politics**
	palestinian	wounding	tael	scored	sentenced	seng	motor	flu	nuclear	republican
	arafat	killed	hk	rebounds	guilty	index	automaker	desease	eu	mccain
	yasser	roadside	gold	points	crimes	prices	toyota	virus	dpark	democrats
	abbas	injuring	cppcc	champion	court	taies	auto	bird	nato	barack
	israeli	crashed	cpc	beats	convicted	stock	ford	health	bilateral	presidential

Table 8: 10 topics learned from our model on APNEWS, IMDB, BNC and Gigawords.

HF).

Results The results in Table 7 show that our TGVAE achieves better performance than a variety of strong baseline methods on both GIGAWORDS and DUC-2004, demonstrating the practical value of our model. It is worthwhile to note that recently several much more complex CNN/RNN architectures have been proposed for abstract text summarization, such as SEASS (Zhou et al., 2017), ConvS2S (Gehring et al., 2017), and Reinforced-ConvS2S (Wang et al., 2018a). In this work, we focus on a relatively simple RNN architecture for fair comparison. In such a way, we are able to conclude that the improvements on the results are mainly from our topic-guided text generation strategy. As can be seen, though the Var-Seq2Seq model achieves comparable performance with the standard Seq2Seq model, the usage of Householder flow for more flexible posterior inference boosts the performance. Additionally, by combining the proposed topic-dependent prior and Householder flow, we yield further performance improvements, demonstrating the importance of topic guidance for text summarization.

To demonstrate the readability and diversity of the generated summaries, we present typical examples in Table 6. The words in blue are the topic words that appear in the source article but do not exist in the reference, while the words in red are neither in the reference nor in the source article. When the topic information is provided, our model is able to generate semantically-meaningful words which may not even exist in the reference summaries and the source articles. Additionally, with our topic-guided model, we can always generate a summary with meaningful initial words. These phenomena imply that our model supplies more insightful semantic information to improve the quality of generated summaries.

Finally, to demonstrate that our TGVAE learns interpretable topic-dependent GMM priors, we draw multiple samples from each mixture component and visualize them with t-SNE (Maaten and Hinton, 2008). As can be seen from Figure 2, we have learned a group of separable *topic-dependent* components. Each component is clustered and also maintains semantic meaning in the latent space, *e.g.*, the clusters corresponding to the topic "stock" and "finance" are close to each other, while the clusters for "finance" and "disease" are far away from each other. Additionally, to understand the topic model we have learned, we provide the top 5 words for 10 randomly chosen topics on each dataset (the boldface word is the topic name summarized by us), as shown in Table 8.

6 Conclusion

A novel text generator is developed, combining a VAE-based neural sequence model with a neural topic model. The model is an extension of conditional VAEs in the framework of unsupervised learning, in which the topics are extracted from the data with clustering structure rather than predefined labels. An effective inference method based on Householder flow is designed to encourage the complexity and the diversity of the learned topics. Experimental results are encouraging, across multiple NLP tasks.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Christian H Bischof and Xiaobai Sun. 1994. On orthogonal block elimination. *Preprint MCS-P450-0794, Mathematics and Computer Science Division, Argonne National Laboratory*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR*.

BNC BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition). *Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium.*

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. 2017. Continuous-time flows for efficient inference and density estimation. *arXiv preprint arXiv:1709.01179*.

Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. 2018. Adversarial text generation via feature-mover's distance. In *NeurIPS*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2016. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-Amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NIPS*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. *arXiv preprint arXiv:1704.08012*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017b. Deep recurrent generative decoder for abstractive text summarization. *arXiv preprint arXiv:1708.00625*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. *arXiv preprint arXiv:1706.00359*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *SLT*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv preprint arXiv:1602.06023*.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018. Nash: Toward end-to-end neural architecture for generative semantic hashing. *arXiv preprint arXiv:1805.05361*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017a. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109*.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017b. A conditional variational framework for dialog generation. *arXiv preprint arXiv:1705.00316*.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Xiaobai Sun and Christian Bischof. 1995. A basis-kernel representation of orthogonal matrices. *SIAM journal on matrix analysis and applications*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Jakub M Tomczak and Max Welling. 2016. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018a. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. *arXiv preprint arXiv:1805.03616*.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2017. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*.

Wenlin Wang, Yunchen Pu, Vinay Kumar Verma, Kai Fan, Yizhe Zhang, Changyou Chen, Piyush Rai, and Lawrence Carin. 2018b. Zero-shot learning via class-conditioned deep generative models. In *AAAI*.

Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. Diversifying restricted Boltzmann machine for document modeling. In *KDD*.

Hongteng Xu, Wenlin Wang, Wei Liu, and Lawrence Carin. 2018. Distilled wasserstein learning for word embedding and topic modeling. In *NIPS*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*.

Ruiyi Zhang, Changyou Chen, Zhe Gan, Wenlin Wang, Liqun Chen, Dinghan Shen, Guoyin Wang, and Lawrence Carin. 2018. Sequence generation with guider network. *arXiv preprint arXiv:1811.00696*.

Ruiyi Zhang, Chunyuan Li, Changyou Chen, and Lawrence Carin. 2017a. Learning structural weight uncertainty for sequential decision-making. *arXiv preprint arXiv:1801.00085*.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017b. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv preprint arXiv:1704.07073*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*.

Implementation of a Chomsky-Schützenberger n-Best Parser for Weighted Multiple Context-Free Grammars

Thomas Ruprecht and **Tobias Denkinger**
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
{thomas.ruprecht,tobias.denkinger}@tu-dresden.de

Abstract

Constituent parsing has been studied extensively in the last decades. Chomsky-Schützenberger parsing as an approach to constituent parsing has only been investigated theoretically, yet. It uses the decomposition of a language into a regular language, a homomorphism, and a bracket language to divide the parsing problem into simpler subproblems. We provide the first implementation of Chomsky-Schützenberger parsing. It employs multiple context-free grammars and incorporates many refinements to achieve feasibility. We compare its performance to state-of-the-art grammar-based parsers.

1 Introduction

The description of the syntax of natural languages (such as Danish, English, and German) with the help of formal grammars has been studied since Chomsky (1956). With a formal grammar, computers can calculate a syntactic representation (called *parse*) of a sentence in a natural language. Of the grammar classes in the Chomsky hierarchy (Chomsky, 1959), context-free grammars (short: CFGs) lack the expressive power necessary to model natural languages (Shieber, 1985) and parsing with context-sensitive grammars cannot be done efficiently (i.e. in polynomial time). This led to the introduction of a series of classes of *mildly context-sensitive grammars* (Joshi, 1985) that allow parsing in polynomial time but also capture an increasing amount of phenomena present in natural languages. Tree adjoining grammars (Joshi et al., 1975), linear context-free string-rewriting systems (short: LCFRSs, Vijay-Shanker et al., 1987), and multiple CFGs (short: MCFGs, Seki et al., 1991) are among those classes.

Chomsky-Schützenberger (short: CS) parsing was introduced by Hulden (2011) for CFGs and extended to MCFGs by Denkinger (2017). It uses a classical theorem by Chomsky and Schützenberger (1963, or the generalisation by Yoshinaka et al., 2010), which states that the language $\mathcal{L}(G)$ of a CFG (or an MCFG) G can be represented by a regular language R, a homomorphism h, and a Dyck language (resp. multiple Dyck language) D such that $\mathcal{L}(G) = h(R \cap D)$. The elements of $R \cap D$ correspond to parses in G. For a sentence w, a CS parser calculates the elements of $h^{-1}(w) \cap R \cap D$ and transforms them into parses. CS parsing can be viewed as a coarse-to-fine mechanism where R corresponds to the coarse grammar and $R \cap D$ to the fine grammar. The respective coarse-to-fine pipeline consists of (conceptually) simple operations such as h^{-1} or the intersection with R, which provides great flexibility. The flexibility is used to provide a fallback mechanism in case a finer stage of the pipeline rejects all proposals of a coarser stage. It also permits CS parsing in a broader setting than usual (for parsing) with minimal modification (see sec. 6).

We suspected that the coarse-to-fine view on CS parsing leads to an efficient implementation. Since initial tests revealed that the original algorithm for MCFGs (Denkinger, 2017, alg. 3, recalled in sec. 2) is not feasible in practice, we explore numerous optimisations (sec. 4), one of which is the use of a context-free approximation of the multiple Dyck language D. We introduce *component-wise derivations* (sec. 3) to relate this context-free approximation to D. Employing the optimisations, we provide the first implementation of a CS parser. In sec. 5, we compare our parser's performance to Grammatical Framework (Angelov and Ljunglöf, 2014), rparse (Kallmeyer and Maier, 2013), and disco-dop (van Cranenburgh et al., 2016). We restrict our comparison to (discontinuous) grammar-based parsers (excluding e.g. transition systems, Maier, 2015, Coavoux and Crabbé, 2017) since the principle of CS parsing requires a grammar.

Proceedings of NAACL-HLT 2019, pages 178–191
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Preliminaries

The sets of *non-negative integers* and *positive integers* are denoted by $\mathbb{N}$ and $\mathbb{N}_+$, respectively. We abbreviate $\{1, \dots, n\}$ by $[n]$ for each $n \in \mathbb{N}$.

Let A and B be sets. The *powerset of* A and the *set of (finite) strings over* A are denoted by $\mathcal{P}(A)$ and A^*, respectively. The *set of possibly infinite sequences of elements of* A is denoted by A^ω. A *partition of* A is a set $\mathfrak{P} \subseteq \mathcal{P}(A)$ whose elements (called *cells*) are non-empty, pairwise disjoint, and cover A (i.e. $\bigcup_{\mathfrak{p} \in \mathfrak{P}} \mathfrak{p} = A$). For each $a \in A$ and each equivalence relation $\approx$ on A, we denoted the *equivalence class of* a *w.r.t.* $\approx$ by $[a]_\approx$. The set of *functions from* A *to* B is denoted by $A \to B$. Note that $(A \to B) \subseteq (A \times B)$. The *composition* of two binary relations R_1 and R_2 is $R_2 \circ R_1 = \{(a, c) \mid \exists b\colon (a, b) \in R_1, (b, c) \in R_2\}$.

Finite state automata. We assume that the reader is familiar with finite state automata. For details, we refer to Hopcroft and Ullman (1979).

A *finite state automaton* (short: FSA) is a tuple $\mathcal{A} = (Q, \Delta, q_\mathrm{i}, q_\mathrm{f}, T)$ where Q and Δ are finite sets (*states* and *terminals*, respectively), $q_\mathrm{i}, q_\mathrm{f} \in Q$ (*initial* and *final state*, respectively), and $T \subseteq Q \times \Delta^* \times Q$ is finite (*transitions*). We call q the *source* and q' the *target* of a transition (q, u, q'). A *run* is a string θ of transitions such that the target of a transition is the source of the next transition in θ. The *language of* $\mathcal{A}$ is denoted by $\mathcal{L}(\mathcal{A})$.

Sorts. Sorts are a widespread concept in computer science: one can think of sorts as data types in a programming language. Let S be a set (of *sorts*). An S*-sorted set* is a tuple (Ω, sort) where Ω is a set and $\mathit{sort}\colon \Omega \to S$. We abbreviate (Ω, sort) by Ω and $\mathit{sort}^{-1}(s)$ by Ω_s for $s \in S$.

Now let Ω be an $(S^* \times S)$-sorted set. The *set of trees over* Ω is the S-sorted set T_Ω where $(\mathrm{T}_\Omega)_s = \{\omega(t_1, \dots, t_k) \mid s_1, \dots, s_k \in S, \omega \in \Omega_{(s_1 \cdots s_k, s)}, t_1 \in (\mathrm{T}_\Omega)_{s_1}, \dots, t_k \in (\mathrm{T}_\Omega)_{s_k}\}$ for each $s \in S$.

Multiple context-free grammars. A rule of a context-free grammar has the ability to concatenate the strings generated by its right-hand side non-terminals. *Multiple* context-free grammars extend this ability to concatenating string-*tuples*. This is done with the help of composition functions. Let Σ be a finite set. A *composition function w.r.t.* Σ is a function c that takes tuples of strings over Σ as arguments and returns a tuple of strings over Σ (i.e. there are $k \in \mathbb{N}$ and $s_1, \dots, s_k, s \in \mathbb{N}_+$ such that $c\colon (\Sigma^*)^{s_1} \times \dots \times (\Sigma^*)^{s_k} \to (\Sigma^*)^s$), and is defined by an equation $c((x_1^1, \dots, x_1^{s_1}), \dots, (x_k^1, \dots, x_k^{s_k})) = (u_1, \dots, u_s)$ where $u_1, \dots, u_s$ are strings of x_i^j's and symbols from Σ. We call c *linear* if each x_i^j occurs at most once in $u_1 \cdots u_s$. We sometimes write $[u_1, \dots, u_s]$ instead of c. Furthermore, setting $\mathit{sort}(c) = (s_1 \cdots s_k, s)$, the composition functions w.r.t. Σ form a sorted set.

The following example shows how linear composition functions are used in the rules of a multiple context-free grammar.

Example 1. Consider $G = (N, \Sigma, S, P)$ where $N = \{S, A, B\}$ and $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$ are finite sets (*non-terminals* and *terminals*, respectively), $S \in N$ (*initial non-terminal*) and P is a finite set (*rules*) that contains the following five objects:

$$\begin{aligned} &\rho_1 = S \to [x_1^1 x_2^1 x_1^2 x_2^2](A, B) \\ &\rho_2 = A \to [\mathrm{a}x_1^1, \mathrm{c}x_1^2](A) \qquad \rho_4 = A \to [\varepsilon, \varepsilon]() \\ &\rho_3 = B \to [\mathrm{b}x_1^1, \mathrm{d}x_1^2](B) \qquad \rho_5 = B \to [\varepsilon, \varepsilon](). \end{aligned}$$

We call G a multiple context-free grammar. Consider the rule ρ_1. Similar to a rule of a context-free grammar, ρ_1 rule has one left-hand side non-terminal (S) and zero or more right-hand side non-terminals (A and B). A *derivation of* G can be build by combining rules in P to form a tree according to their left- and right-hand side non-terminals. If a derivation starts with the initial non-terminal (here S), then it is called *complete*. Hence, each complete derivation in G has the form $d_{m,n} = \rho_1(\rho_2^m(\rho_4), \rho_3^n(\rho_5))$ for some $m, n \in \mathbb{N}$. If we replace each rule in a derivation by its composition function, we obtain a term of composition functions which can be evaluated. We call the resulting value the *yield* of a derivation. A derivation $d_{m,n}$ has yield $\mathrm{yd}(d_{m,n}) = \mathrm{a}^m \mathrm{b}^n \mathrm{c}^m \mathrm{d}^n$. The set of yields of all complete derivations is the *language of* G: $\mathcal{L}(G) = \{\mathrm{a}^m \mathrm{b}^n \mathrm{c}^m \mathrm{d}^n \mid m, n \in \mathbb{N}\}$. ☐

The following definition formalises the notions of ex. 1 and introduces some additional concepts.

Definition 2 (Seki et al., 1991)**.** A *multiple context-free grammar* (short: MCFG) is a tuple $G = (N, \Sigma, S, P)$ where N is a finite $\mathbb{N}_+$-sorted set (*non-terminals*), Σ is a finite set (*terminals*), $S \in N_1$ (*initial non-terminal*), P is a finite $(N^* \times N)$-sorted set of strings ρ of the form $A \to c(B_1, \dots, B_k)$ such that $A, B_1, \dots, B_k \in N$, c is a linear composition function, and $\mathit{sort}(c) =$

$(sort(B_1)\cdots sort(B_k), sort(A))$. The sort of ρ is $(B_1\cdots B_k, A)$. The *left-hand side* (short: lhs) of ρ is A. The *fanout of* ρ is $\mathrm{fanout}(\rho) = sort(A)$ and the *rank of* ρ is $rank(\rho) = k$. The elements of P are called *rules*.

The *set of derivations (resp. complete derivations) of* G is $\mathrm{D}_G = \mathrm{T}_P$ (resp. $\mathrm{D}^{\mathrm{c}}_G = (\mathrm{T}_P)_S$). Let $w \in \Sigma^*$ and $d = \rho(d_1,\dots,d_k) \in \mathrm{D}_G$ with $\rho = A \to c(B_1,\dots,B_k)$. The *yield of* d is $\mathrm{yd}(d) = c(\mathrm{yd}(d_1),\dots,\mathrm{yd}(d_k))$. The *set of derivations of* w *in* G is $\mathrm{D}^{\mathrm{c}}_G(w) = \mathrm{yd}^{-1}(w)\cap\mathrm{D}^{\mathrm{c}}_G$. The *language of* A *in* G is $\mathcal{L}(G,A) = \{\mathrm{yd}(d) \mid d \in (\mathrm{T}_P)_A\}$. The *language of* G is $\mathcal{L}(G) = \mathcal{L}(G,S)$. Any language generated by an MCFG is called *multiple context-free* (short: mcf). ☐

A *context-free grammar* (short: CFG) is an MCFG where each non-terminal has sort 1. Each rule of a CFG has the form $A \to [u_0 x^1_{i(1)} u_1 \cdots x^1_{i(n)} u_n](B_1,\dots,B_k)$. We abbreviate this rule by $A \to u_0 B_{i(1)} u_1 \cdots B_{i(n)} u_n$.

Weighted multiple context-free grammars. A weighted MCFG is obtained by assigning a weight to each rule of an (unweighted) MCFG. In this paper, the weights will be taken from a **p**artially **o**rdered **c**ommutative **mo**noid with **z**ero (short: POCMOZ). A *POCMOZ* is an algebra $(M,\odot,\mathbb{1},\mathbb{0},\trianglelefteq)$ where $\trianglelefteq$ is a partial order on M; $\odot$ is associative, commutative, decreasing (i.e. $m\odot m' \trianglelefteq m$), and monotonous (i.e. $m_1 \trianglelefteq m_2$ implies $m_1\odot m \trianglelefteq m_2\odot m$); $\mathbb{1}$ is neutral w.r.t. $\odot$; and $\mathbb{0}$ is absorbing w.r.t. $\odot$. We call M *factorisable* if for each $m \in M\setminus\{\mathbb{1}\}$, there are $m_1, m_2 \in M\setminus\{\mathbb{1}\}$ with $m = m_1\odot m_2$. The *probability algebra* $\mathrm{Pr} = ([0,1],\cdot,1,0,\leq)$ is a factorisable POCMOZ where $r = \sqrt{r}\cdot\sqrt{r}$ for each $r \in [0,1)$.

Example 3 (continues ex. 1). Consider the tuple (G,μ) where $\mu\colon P \to \mathrm{Pr}$ is a function where $\mu(\rho_1) = 1$, $\mu(\rho_2) = \mu(\rho_4) = 1/2$, $\mu(\rho_3) = 1/3$, and $\mu(\rho_5) = 2/3$. We call (G,μ) a weighted MCFG. The *weight* of the a derivation $d_{m,n}$ is obtained by multiplying the weights of all rule occurrences in it: $\mathrm{wt}(d_{m,n}) = 1/2^{m+1}\cdot 2/3^{n+1}$. ☐

Definition 4. A *weighted MCFG* (short: wMCFG) is a tuple (G,μ) where $G = (N,\Sigma,S,P)$ is an MCFG (*underlying MCFG*), $\mu\colon P \to M\setminus\{\mathbb{0}\}$ (*weight assignment*), and $(M,\odot,\mathbb{1},\mathbb{0},\trianglelefteq)$ is a factorisable POCMOZ.

(G,μ) inherits all objects associated with G. Let $d = \rho(d_1,\dots,d_k) \in \mathrm{D}_G$. The *weight of* d is $\mathrm{wt}(d) = \mu(\rho)\odot\bigodot_{i=1}^k \mathrm{wt}(d_i)$. ☐

For the rest of this paper, we fix a wMCFG (G,μ) *with underlying MCFG* $G = (N,\Sigma,S,P)$ *and weight assignment* $\mu\colon P \to M$.

Chomsky-Schützenberger theorem. In the Chomsky-Schützenberger theorem for CFGs (cf. sec. 1), D contains strings of brackets where each opening bracket is matched by the corresponding closing bracket. This property can be described with an equivalence relation. Let Δ be a set (of opening brackets) and $\bar\Delta$ be the set (of closing brackets) that contains $\bar\delta$ for each $\delta \in \Delta$. We define $\equiv_\Delta$ as the smallest equivalence relation where $u\,\delta\bar\delta\,v \equiv_\Delta uv$ for each $\delta \in \Delta$ and $u, v \in \Delta^*$. The *Dyck language w.r.t.* Δ is $\mathrm{D}_\Delta = [\varepsilon]_{\equiv_\Delta}$.

In the Chomsky-Schützenberger representation for MCFGs, the brackets fulfil three functions: (i) *terminal brackets* $[\![_\sigma\,]\!]_\sigma$ stand for a terminal symbol σ, (ii) *component brackets* $[\![^\ell_\rho$ and $]\!]^\ell_\rho$ denote beginning and end of substrings produced by the ℓ-th component of a rule ρ, and (iii) *variable brackets* $[\![^j_{\rho,i}$ and $]\!]^j_{\rho,i}$ denote beginning and end of substrings produced by variable x^j_i in a rule ρ. As for CFGs, each opening bracket must be matched by the corresponding closing bracket. Furthermore, because applying a rule of an MCFG produces multiple strings simultaneously, we need to ensure that the brackets corresponding to the same application of a rule occur simultaneously. This is described with another equivalence relation. Let $\mathfrak{P}$ be a partition of Δ. Intuitively, each cell of $\mathfrak{P}$ is a set of (opening) brackets that occur simultaneously. We define $\equiv_{\mathfrak{P}}$ as the smallest equivalence relation on $\mathcal{P}\big((\Delta\cup\bar\Delta)^*\big)$ where for each $\{\delta_1,\dots,\delta_s\} \in \mathfrak{P}$ with $|\{\delta_1,\dots,\delta_s\}| = s$, $u_0,\dots,u_s,v_1,\dots,v_s \in \mathrm{D}_\Delta$, and $L \subseteq (\Delta\cup\bar\Delta)^*$:

$$\begin{aligned} &\{u_0\;\delta_1 v_1\bar\delta_1\;u_1\cdots\;\delta_s v_s\bar\delta_s\;u_s\}\cup L\\ \equiv_{\mathfrak{P}}\;&\{u_0\cdots u_s,\;\;v_1\cdots v_s\}\qquad\;\cup L.\end{aligned}$$

The *multiple Dyck language w.r.t.* $\mathfrak{P}$ is $\mathrm{mD}_{\mathfrak{P}} = \bigcup(L \mid L \in [\{\varepsilon\}]_{\equiv_{\mathfrak{P}}})$. Note that $\mathrm{mD}_{\mathfrak{P}} \subseteq \mathrm{D}_\Delta$.

Theorem 5 provides a representation of each mcf language by a multiple Dyck language (see above), a recognisable language (to ensure local consistency), and a homomorphism (to decode the bracket sequences into terminal strings). The corresponding construction is recalled in def. 6.

Theorem 5 (cf. Yoshinaka et al., 2010, thm. 3). For every mcf language $L \subseteq \Sigma^*$ there are a homomorphism $h\colon(\Delta\cup\bar\Delta)^* \to \Sigma^*$, a regular language $R \subseteq (\Delta\cup\bar\Delta)^*$, and a multiple Dyck language $mD \subseteq (\Delta\cup\bar\Delta)^*$ such that $L = h(R\cap mD)$. ■

Definition 6 (Denkinger, 2017, def. 3.6, 4.9, 5.15)**.** The *multiple Dyck language w.r.t. G* is $\mathrm{mD}_G = \mathrm{mD}_{\mathfrak{P}_G}$ where $\mathfrak{P}_G$ is the smallest set that contains the cell $\{\llbracket_\sigma\}$ for each $\sigma \in \Sigma$ and the cells $\{\llbracket_\rho^\ell \mid \ell \in [sort(A)]\}$ and $\{\llbracket_{\rho,i}^j \mid j \in [sort(B_i)]\}$ for each $\rho = A \to c(B_1, \ldots, B_k) \in P$ and $i \in [k]$. Let $\Delta_G = \bigcup_{\mathfrak{p} \in \mathfrak{P}_G} \mathfrak{p}$. We denote the elements of $\overline{\Delta_G}$ by closing brackets, e.g. $\overline{\llbracket_\sigma} = \rrbracket_\sigma$, and let $\Omega_G = \Delta_G \cup \overline{\Delta_G}$.

The *homomorphism w.r.t. G*, denoted by hom_G, is the unique extension of $h\colon \Omega_G \to \Sigma \cup \{\varepsilon\}$ to strings where $h(\delta) = \sigma$ if δ is of the form $\llbracket_\sigma$ and $h(\delta) = \varepsilon$ otherwise.

The *automaton w.r.t. G*, denoted by $\mathcal{A}_G$, is the FSA $(Q, \Omega_G, S^1, \overline{S^1}, T)$ where $Q = \{A^\ell, \overline{A^\ell} \mid A \in N, \ell \in [sort(A)]\}$ and T is the smallest set such that for each rule $\rho \in P$ of the form

$$A \to [u_{1,0}y_{1,1}u_{1,1} \cdots y_{1,n_1}u_{1,n_1},\ \ldots,\\ u_{s,0}y_{s,1}u_{s,1} \cdots y_{s,n_s}u_{s,n_s}](B_1, \ldots, B_k)$$

where the ys are elements of X and the us are elements of Σ^*, we have (abbreviating $\llbracket_{\sigma_1}\rrbracket_{\sigma_2} \cdots \llbracket_{\sigma_k}\rrbracket_{\sigma_k}$ by $\widetilde{\sigma_1 \cdots \sigma_k}$) the following transitions in T:

(i) $\big(A^\ell,\ \llbracket_\rho^\ell \widetilde{u_{\ell,0}} \rrbracket_\rho^\ell,\ \bar{A}^\ell\big) \in T$ for every $\ell \in [s]$ with $n_\ell = 0$,

(ii) $\big(A^\ell,\ \llbracket_\rho^\ell \widetilde{u_{\ell,0}} \llbracket_{\rho,i}^j,\ B_i^j\big) \in T$ for every $\ell \in [s]$ where $n_\ell \neq 0$ and $y_{\ell,1}$ is of the form x_i^j,

(iii) $\big(\bar{B}_i^j,\ \rrbracket_{\rho,i}^j \widetilde{u_{\ell,\kappa}} \llbracket_{\rho,i'}^{j'},\ B_{i'}^{j'}\big) \in T$ for every $\ell \in [s]$ and $\kappa \in [n_\ell - 1]$ where $y_{\ell,\kappa}$ is of the form x_i^j and $y_{\ell,\kappa+1}$ is of the form $x_{i'}^{j'}$, and

(iv) $\big(\bar{B}_i^j,\ \rrbracket_{\rho,i}^j \widetilde{u_{\ell,n_\ell}} \rrbracket_\rho^\ell,\ \bar{A}^\ell\big) \in T$ for every $\ell \in [s]$ where $n_\ell \neq 0$ and y_{ℓ,n_ℓ} is of the form x_i^j.

We abbreviate $\mathcal{L}(\mathcal{A}_G)$ by $\mathcal{R}_G$. ☐

Example 7 (continues ex. 1)**.** The automaton w.r.t. G is shown in fig. 1. An illustration of the application of $\equiv_{\mathfrak{P}_G}$ is given in the appendix (p. 11). ☐

The vanilla parser. The *vanilla parser* (i.e. alg. 3 from Denkinger, 2017), is shown in fig. 2 (top). Similar to the parser proposed by Hulden (2011), we divide it in three essential phases: (i) FSA constructions for the intersection of $\mathrm{hom}_G^{-1}(w)$ and $\mathcal{R}_G$, (ii) an extraction of (in our case *multiple*) Dyck words from the intersection, and (iii) the conversion of words into derivations.

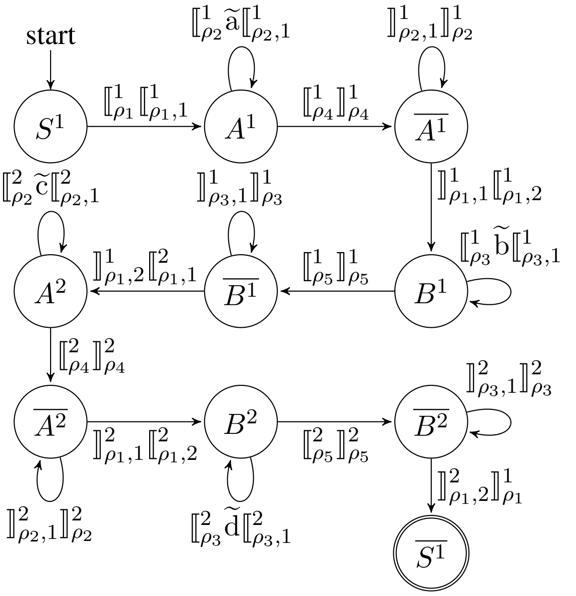


Figure 1: Automaton w.r.t. G, cf. ex. 7.

Formally, the vanilla parser is the function $V\colon \Sigma^* \to (\mathrm{D}_G^{\mathrm{c}})^\omega$ defined as

$$V = \textsc{map}(\textsc{toDeriv}) \circ \textsc{filter}(\mathrm{mD}_G)\\ \circ \textsc{sort}(\mu') \circ (\cap\, \mathcal{R}_G) \circ \mathrm{hom}_G^{-1}$$

where $\mathrm{hom}_G^{-1}(w) \cap \mathcal{R}_G$ is represented by an FSA for each $w \in \Sigma^*$ (phase (i)). $\mu'(u)$ is the product of the weights of each occurrence of a bracket of the form $\llbracket_\rho^\ell$ or $\rrbracket_\rho^\ell$ in u. These weights are fixed such that $\mu'\big(\llbracket_\rho^1\rrbracket_\rho^1 \cdots \llbracket_\rho^\ell\rrbracket_\rho^\ell\big) = \mu(\rho)$ for each $\rho \in P$ with fanout ℓ. $\textsc{sort}(\mu')$ brings the elements of its argument, which is a subset of $\Omega_G{}^*$, in some descending order w.r.t. μ' and $\trianglelefteq$, returning a (possibly infinite) sequence of elements of $\Omega_G{}^*$, which we call *candidates*. Sequences are implemented as iterators. $\textsc{filter}(\mathrm{mD}_G)$ removes the candidates from its argument sequence that are not in mD_G while preserving the order (cf. Denkinger, 2017, alg. 2). (Both steps, $\textsc{sort}(\mu')$ and $\textsc{filter}(\mathrm{mD}_G)$, are phase (ii).) $\textsc{toDeriv}$ returns the derivation in G that corresponds to its argument (which is from the set $\mathcal{R}_G \cap \mathrm{mD}_G$), cf. Denkinger (2017, function fromBrackets, p. 20). $\textsc{map}(\textsc{toDeriv})$ applies $\textsc{toDeriv}$ to each candidate in its argument while preserving the order (phase (iii)).

Denkinger (2017, thm. 5.22) showed that $\textsc{take}(n) \circ V$ solves the n-best parsing problem.[1] We omit the additional restrictions that he imposed on the given wMCFG because they are only necessary to show the termination of his algorithm.

---

[1] In the following, we will gloss over the distinction between derivations and parses.

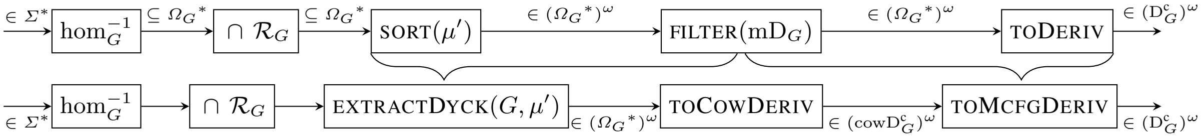


Figure 2: Visualisation of the vanilla parser (top) and the parser with the optimisations from sec. 4 (bottom).

3 Component-Wise Derivations

In sec. 4, we will outline modifications to the vanilla parser that make the extraction of the elements of mD_G from $\mathrm{hom}_G^{-1}(w) \cap \mathcal{R}_G$ efficient (items 2–4). To facilitate this, we first decompose FILTER(mD_G) into FILTER(mD_G) $\circ$ FILTER(D_{Δ_G}), which is possible because $\mathrm{D}_{\Delta_G} \supseteq \mathrm{mD}_G$. Secondly, we implement FILTER(D_{Δ_G}) $\circ$ SORT(μ') with a dynamic programming algorithm (cf. Hulden, 2011, alg. 1, similar to Bar-Hillel et al., 1961, sec. 8). And lastly, we replace FILTER(mD_G) by steps that exploit the well-bracketing of the elements of D_{Δ_G}.

The elements of $\mathcal{R}_G \cap \mathrm{D}_{\Delta_G}$ can be represented as trees over rules of G.[2] We label the edges of those trees to allow us to check if vertices that correspond to the same application of a rule of the MCFG G match. The resulting objects are called *component-wise derivations*. The set $\mathcal{R}_G \cap \mathrm{D}_{\Delta_G}$ is characterised in terms a CFG G_{cf}.

Definition 8. Let $\rho \in P$ be a rule of the form $A \to [u_1, \ldots, u_s](B_1, \ldots, B_k)$, $\ell \in [s]$, and u_ℓ be of the form $w_0 x_{i(1)}^{j(1)} w_1 \cdots x_{i(n)}^{j(n)} w_n$ for some $w_0, \ldots, w_n \in \Sigma^*$. We define the rule

$$\rho^{(\ell)} = A^\ell \to \llbracket^\ell_\rho \widetilde{w_0} v_1 \widetilde{w_1} \cdots v_n \widetilde{w_n} \rrbracket^\ell_\rho$$

where each $v_\kappa = \llbracket^{j(\kappa)}_{\rho,i(\kappa)} B^{j(\kappa)}_{i(\kappa)} \rrbracket^{j(\kappa)}_{\rho,i(\kappa)}$. The *context-free CS approximation of G* (short: *CFA*), denoted by G_{cf}, is the CFG $(N_{\mathrm{cf}}, \Omega_G, S^1, P_{\mathrm{cf}})$ where $N_{\mathrm{cf}} = \{A^\ell \mid A \in N, \ell \in [\mathit{sort}(A)]\}$ and $P_{\mathrm{cf}} = \{\rho^{(\ell)} \mid \rho \in P, \ell \in [\mathrm{fanout}(\rho)]\}$. ☐

Observation 9. $\mathrm{D}_{\Delta_G} \cap \mathcal{R}_G = \mathcal{L}(G_{\mathrm{cf}})$. ■

We introduce *component-wise derivations* to relate the derivations of G_{cf} with those of G.

Definition 10. Let $\ell \in \mathbb{N}_+$ and t be a tree whose vertices are labelled with elements of P and whose edges are labelled with elements of $\mathbb{N}_+ \times \mathbb{N}_+$. The *label at the root of t* is denoted by $\mathrm{root}(t)$. The *set of labels of the outgoing edges from the root of t* is denoted by $\mathrm{out}(t)$. A (i,j)-*subtree of t*, is a sub-graph of t consisting of all the vertices (and their edges) reachable from some target vertex of the outgoing edge from the root that is labelled with (i,j). If there is a unique (i,j)-subtree of t, then we denote it by $\mathrm{sub}_{(i,j)}(t)$. Now let $\mathrm{root}(t) = A \to [u_1, \ldots, u_s](B_1, \ldots, B_k)$. We call t an *(ℓ-)component-wise derivation, short: (ℓ-)cow derivation, of G* if the following four requirements are met: (i) $\mathrm{out}(t)$ contains exactly the pairs (i,j) such that x_i^j occurs in u_ℓ, (ii) a unique (i,j)-subtree of t exists, (iii) $\mathrm{root}(\mathrm{sub}_{(i,j)}(t))$ has lhs B_i, and (iv) $\mathrm{sub}_{(i,j)}(t)$ is a j-cow derivation for each $(i,j) \in \mathrm{out}(t)$. We denote the set of cow derivations of G whose root's lhs is S by $\mathrm{cowD}^{\mathrm{c}}_G$. The set of ℓ-cow derivations whose root's label has lhs A is denoted by $\ell\text{-}\mathrm{cowD}^{A}_G$. ☐

An example of a cow derivation is shown in fig. 3a. The root is the top-most vertex.

Definition 11. Let $\rho = A \to c(B_1, \ldots, B_k) \in P$, $\ell \in [\mathrm{fanout}(\rho)]$, and the ℓ-th component of c be $u_0 x_{i(1)}^{j(1)} u_1 \cdots x_{i(n)}^{j(n)} u_n$ with $u_1, \ldots, u_n \in \Sigma^*$. Furthermore, for each $\kappa \in [n]$, let $t_\kappa \in j(\kappa)\text{-}\mathrm{cowD}^{B_{i(\kappa)}}_G$. By $\rho\langle (i(\kappa), j(\kappa))/t_\kappa \mid \kappa \in [n] \rangle$, we denote the cow derivation t such that $\mathrm{root}(t) = \rho$, $\mathrm{out}(t) = \{(i(\kappa), j(\kappa)) \mid \kappa \in [n]\}$, and for each $\kappa \in [n]$: $\mathrm{sub}_{(i(\kappa), j(\kappa))}(t) = t_\kappa$. ☐

Lemma 12. There is a bijection toCowD between $\mathcal{L}(G_{\mathrm{cf}})$ and $\mathrm{cowD}^{\mathrm{c}}_G$. ■

Proof sketch. We define the partial function toCowD from ${\Omega_G}^*$ to cow derivations of G as follows: $\mathrm{toCowD}(u) = \rho\langle (i(\kappa), j(\kappa))/\mathrm{toCowD}(v_\kappa) \mid \kappa \in [n] \rangle$ if u is of the form

$$\llbracket^\ell_\rho \widetilde{u_0} \llbracket^{j(1)}_{\rho,i(1)} v_1 \rrbracket^{j(1)}_{\rho,i(1)} \widetilde{u_1} \ldots \llbracket^{j(n)}_{\rho,i(n)} v_n \rrbracket^{j(n)}_{\rho,i(n)} \widetilde{u_n} \rrbracket^\ell_\rho$$

for some rule $\rho = A \to c(B_1, \ldots, B_k)$ where the ℓ-th component of c is $u_0 x_{i(1)}^{j(1)} u_1 \cdots x_{i(n)}^{j(n)} u_n$ with $u_1, \ldots, u_n \in \Sigma^*$; otherwise, $\mathrm{toCowD}(u)$ is undefined. The partial function toCowD is a bijection between $\mathcal{L}(G_{\mathrm{cf}})$ and $\mathrm{cowD}^{\mathrm{c}}_G$ (proven in appendix A.2). ■

---

[2] Those trees correspond to the derivations of the *guiding grammar* in the coarse-to-fine parsing approach of Barthélemy et al. (2001, sec. 3).

Example 13 (continues ex. 1)**.** We construct $G_{\text{cf}} = (\{S^1, A^1, A^2, B^1, B^2\}, \varOmega_G, S^1, P_{\text{cf}})$ where P_{cf} contains, among others, the following rules:

$$\begin{aligned}
\rho_1^{(1)} &= S^1 \to \llbracket^1_{\rho_1} \llbracket^1_{\rho_1,1} A^1 \rrbracket^1_{\rho_1,1} \llbracket^1_{\rho_1,2} B^1 \rrbracket^1_{\rho_1,2} \\
&\qquad\qquad \llbracket^2_{\rho_1,1} A^2 \rrbracket^2_{\rho_1,1} \llbracket^2_{\rho_1,2} B^2 \rrbracket^2_{\rho_1,2} \rrbracket^1_{\rho_1}, \\
\rho_3^{(1)} &= B^1 \to \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_3,1} B^1 \rrbracket^1_{\rho_3,1} \rrbracket^1_{\rho_3}, \\
\rho_4^{(1)} &= A^1 \to \llbracket^1_{\rho_4} \rrbracket^1_{\rho_4}, \quad \rho_4^{(2)} = A^2 \to \llbracket^2_{\rho_4} \rrbracket^2_{\rho_4}, \\
\rho_5^{(1)} &= B^1 \to \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5}, \quad \ldots .
\end{aligned}$$

Figure 3a shows the image of the word

$$\begin{aligned}
&\llbracket^1_{\rho_1} \llbracket^1_{\rho_1,1} \llbracket^1_{\rho_4} \rrbracket^1_{\rho_4} \rrbracket^1_{\rho_1,1} \\
&\quad \llbracket^1_{\rho_1,2} \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_3,1} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_3,1} \rrbracket^1_{\rho_3} \rrbracket^1_{\rho_1,2} \\
&\quad \llbracket^2_{\rho_1,1} \llbracket^2_{\rho_4} \rrbracket^2_{\rho_4} \rrbracket^2_{\rho_1,1} \quad \llbracket^2_{\rho_1,2} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_1,2} \rrbracket^1_{\rho_1}
\end{aligned}$$

in $\mathcal{L}(G_{\text{cf}})$ under toCowD. □

In the following, we define a property called *consistency* to discern those cow derivations that correspond to derivations of the MCFG G.

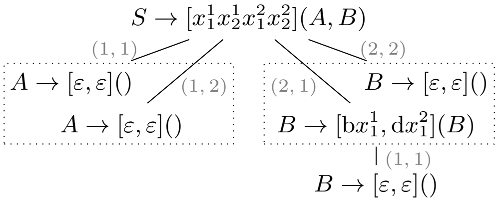
Definition 14. Let $s \in \mathbb{N}_+$ and $t_1, \ldots, t_s$ be cow derivations of G. We call the set $\{t_1, \ldots, t_s\}$ *consistent* if there is a rule $\rho = A \to c(B_1, \ldots, B_k) \in P$ such that $\mathrm{root}(t_1) = \ldots = \mathrm{root}(t_s) = \rho$, $s = \mathit{sort}(A)$, and for each $i \in [k]$: the set $\{\mathrm{sub}_{(i,j)}(t_\ell) \mid \ell \in [s], j \in [\mathit{sort}(B_i)]\colon (i,j) \in \mathrm{out}(t_\ell)\}$ is consistent. If $s = 1$, then we also call t_1 *consistent*. □

The cow derivation shown in fig. 3a is *not* consistent. If we consider the set of nodes that is reachable from the root via edges labelled with a tuple whose first component is 2 (the right dotted box), then it is easy to see that the rules at these nodes are not equal. A consistent cow derivation is shown in the appendix (fig. 6).

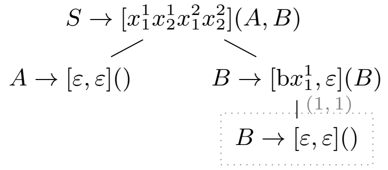
Proposition 15. $\textsc{toDeriv} \circ \mathrm{toCowD}^{-1}$ is a bijection between the consistent cow derivations in $\mathrm{cowD}^{\mathrm{c}}_G$ and $\mathrm{D}^{\mathrm{c}}_G$. ■

4 Optimisations

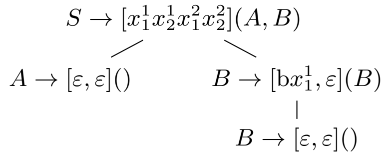
In this section, we describe several improvements to the vanilla parser (cf. end of sec. 2). Since the definitions of $\mathcal{A}_G$, hom_G, and mD_G do not depend on the word w, we may compute appropriate representations for these objects before the beginning of the parsing process, and store them persistently.



(a) A cow derivation. The dotted boxes show clusters of nodes that are reachable from the root via edges labelled with matching first components.



(b) Construction of new rules for each cluster in fig. 3a. If there were any unused nonterminals in these constructed rules, they are removed and the indices of variables changed accordingly. For each cluster, all reachable nodes are clustered via the first component of the labels as in fig. 3a.



(c) Construction of new rules from the clusters in fig. 3b.

Figure 3: A strategy to convert a non-consistent cow derivation into a complete derivation of an MCFG.

In the following, we briefly describe each improvement that we applied to the vanilla parser:

1. Let us call a rule ρ in G *w-consistent* if each string of terminals that occurs in (the composition function of) ρ is a substring of w. A rule is called *useful w.r.t. w* if it occurs in some complete derivation of G in which each rule is w-consistent. In the construction of the FSA for $\mathcal{R}_G \cap \mathrm{hom}_G^{-1}(w)$, we only calculate the transitions that relate to rules of G that are useful w.r.t. w.

2. The function $\textsc{filter}(\mathrm{mD}_G)$ is decomposed into $\textsc{filter}(\mathrm{mD}_G) \circ \textsc{filter}(\mathrm{D}_{\Delta_G})$ in preparation for the next two items.

3. $\textsc{filter}(\mathrm{D}_{\Delta_G}) \circ \textsc{sort}(\mu')$ is implemented with the algorithm $\textsc{extractDyck}(G, \mu')$ that uses dynamic programming to extract Dyck words from the language of the given FSA more efficiently. For this, we extend alg. 1 by Hulden (2011) to use weights such that it returns the elements in descending order w.r.t. μ' and $\trianglelefteq$ (see appendix A.3, alg. 3). In our implementation, we change this al-

Algorithm 1 reads off cow derivations from words of the CFA of G.

Input: $v \in \mathcal{L}(G_{\mathrm{cf}})$

Output: $\mathrm{toCowD}(v)$

```
function TOCOWDERIV($v$)
    $t \leftarrow$ empty graph
    $v \leftarrow$ new vertex
    $pd \leftarrow \varepsilon$                    ▷ pushdown of vertices
    for all $\sigma$ in $v$ from left to right do
        if $\sigma$ is of the form $[\![^{\ell}_{\rho}$ then
            add vertex $v$ to $t$ with label $\rho$
            push vertex $v$ on top of $pd$
        else if $\sigma$ is of the form $[\![^{j}_{\rho,i}$ then
            $v' \leftarrow$ top-most vertex on $pd$
            $v \leftarrow$ new vertex
            add edge $(v' \to v)$ to $t$ with label $(i, j)$
        else if $\sigma$ is of the form $]\!]^{j}_{\rho,i}$ then
            remove top-most vertex from $pd$
    return $t$
```

gorithm even further such that items are explored in a similar fashion as in the CKY-algorithm (Kasami, 1966; Younger, 1967; Cocke and Schwartz, 1970).

4. For FILTER(mD_G), instead of *isMember*$'$ by Denkinger (2017, p. 28–30), which runs in quadratic time, we use the composition of two algorithms that run in linear time:
    - alg. 1, which reads a cow derivation off a given word in $\mathcal{R}_G \cap \mathrm{D}_{\Delta_G}$, and
    - an algorithm that checks a given cow derivation for consistency. (This is similar to alg. 2; but instead of derivations, we return Boolean values. The algorithm is given explicitly in sec. A.3.)

5. Algorithm 2 computes the bijection between $\mathrm{cowD}^{\mathrm{c}}_G$ and $\mathrm{D}^{\mathrm{c}}_G$ (see prop. 15). Analogously to def. 14, the function TOMCFGDERIV' checks a set of cow derivations for equivalence of the root symbol and the function COLLECTCHILDREN groups the subtrees via the first component of the successor labels. It is easy to see that TOMCFGDERIV(t) is only defined if the cow derivation t is consistent (cf. item 4). Thus, we use TOMCFGDERIV in combination with TOCOWDERIV to replace MAP(TODERIV) $\circ$ FILTER(mD_G).

Algorithm 2 converts a consistent element of $\mathrm{cowD}^{\mathrm{c}}_G$ into a complete derivation of G.

Input: $t \in \mathrm{cowD}^{\mathrm{c}}_G$

Output:
$$\begin{cases} \text{TODERIV}\big(\mathrm{toCowD}^{-1}(t)\big) & \text{if } t \text{ is consistent} \\ \text{undefined} & \text{otherwise} \end{cases}$$

```
function TOMCFGDERIV($t$)
    return TOMCFGDERIV'($\{t\}$)

function TOMCFGDERIV'($T$)
    if not $\bigwedge_{t,t' \in T} \big(\mathrm{root}(t) = \mathrm{root}(t')\big)$ then
        return undefined
    $(T_1, \ldots, T_k) \leftarrow$ COLLECTCHILDREN($T$)
    for $i \in [k]$ do
        $t_i \leftarrow$ TOMCFGDERIV'($T_i$)
        if $t_i =$ undefined then
            return undefined
    $\{\sigma\} \leftarrow \big\{\mathrm{root}(t) \mid t \in T\big\}$
    return $\sigma(t_1, \ldots, t_k)$

function COLLECTCHILDREN($T$)
    $\{k\} \leftarrow \big\{rank(\mathrm{root}(t)) \mid t \in T\big\}$
    for $i \in [k]$ do
        $T_i \leftarrow \big\{\mathrm{sub}_{(i,j)}(t) \mid t \in T, (i, j) \in \mathrm{out}(t)\big\}$
    return $(T_1, \ldots, T_k)$
```

The time complexity of alg. 2 is linear in the number of vertices of the given cow derivation. This number, in turn, is linear in the length of the processed candidate.

The parser obtained by applying items 1 to 5 to the vanilla parser is visualised in fig. 2 (bottom). It is sound and complete.[3] The following two modifications (items 6 and 7) destroy both soundness and completeness. Item 6 allows only the best intermediate results to be processed further and limits the results to a subset of those of the vanilla parser. In item 7, we compensate this by an approximation we consider useful in practise.

6. EXTRACTDYCK is extended with an optional implementation of beam search by limiting the amount of items for certain groups of state spans to a specific number (*beam width*), cf. Collins (1999). In our implementation, we chose these groups of state spans such that they correspond to equal states in

---

[3] A parser is *complete* if it (eventually) computes all complete derivations of the given word in the given grammar. A parser is called *sound* if all computed parses are complete derivations of the given word in the given grammar.

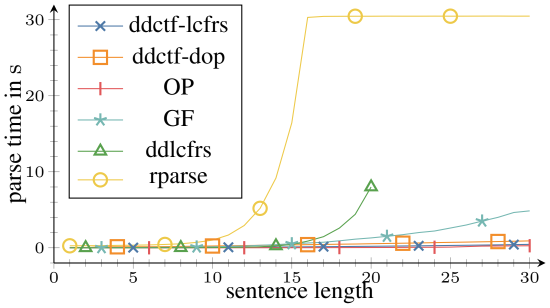
the automaton for $\hom_G^{-1}(w)$. Moreover, we introduce a variable that limits the number of candidates that are yielded by Algorithm 3 (*candidate count*). Both variables are the *meta-parameters* of our parser.

7. We introduce a fallback mechanism for the case that $\text{FILTER}(\text{mD}_G)$ has input candidates but an empty output. Usually, in that case, we would suggest there is no derivation for w in G, yet for robustness, it is preferable to output some parse. Figure 3 illustrates a strategy to construct a complete derivation from any complete *cow* derivation with an example.

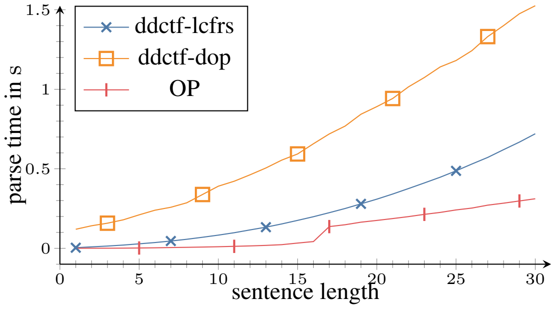
5 Evaluation and Conclusion

We implemented the parser with the modifications sketched in sec. 4 for ε-free and simple wMCFGs,[4] but no problems should arise generalising this implementation to arbitrary wMCFGs. The implementation is available as a part of Rustomata,[5] a framework for weighted automata with storage written in the programming language Rust. We used the *NeGra corpus* (German newspaper articles, 20,602 sentences, 355,096 tokens; Skut et al., 1998) to compare our parser to *Grammatical Framework* (Angelov and Ljunglöf, 2014), *rparse* (Kallmeyer and Maier, 2013), and *disco-dop* (van Cranenburgh et al., 2016) with respect to parse time and accuracy.[6] Our experiments were conducted on defoliated trees, i.e. we removed the leaves from each tree in the corpus. Parsing was performed on gold part-of-speech tags.

We performed a variant of *ten-fold cross validation* (short: TFCV; cf. Mosteller and Tukey, 1968), i.e. we split the corpus into ten consecutive parts; each part becomes the validation set in one iteration while the others serve as training set. We used the first iteration to select suitable values for our meta-parameters and the remaining nine for validation. In case of Rustomata, a binarised and markovized grammar was induced with discodop (head-outward binarisation, $v = 1$, $h = 2$, cf. Klein and Manning, 2003) in each iteration. For all other parsers, we induced a probabilistic LCFRS with the respective default configurations (for details, cf. the evaluation scripts). After that, we ran our parser on each sentence of the validation set and recorded the parse time and the computed 1-best parse. The computed parses were evaluated against the gold parses of the validation set w.r.t. precision, recall, and f_1-score (according to the labelled parseval measures, cf. Black et al., 1991; Collins, 1997, we used the implementation by van Cranenburgh et al., 2016).



(a) NeGra corpus, $|w| \leq 30$ (for ddlcfrs: $|w| \leq 20$)



(b) Lassy corpus, $|w| \leq 30$

Figure 4: Median parse times

Previous experiments with an implementation of the vanilla parser already struggled with small subsets (we used grammars extracted from 250–1500 parse trees) of the NeGra corpus. Therefore, we omit evaluation of the vanilla parser.

Meta-parameters. A grid search for meta-parameters was performed on sentences of up to 20 tokens (see the appendix, tab. 2, for a detailed listing). The results suggested to set the beam width to 200 and the candidate count to 10,000.

Comparison to other parsers. The experiments were performed on sentences with up to 30 tokens. We instructed rparse, Grammatical Framework (short: GF) and Rustomata (short: OP) to stop parsing each sentence after 30 seconds (*timeout*). Disco-dop did not permit passing a timeout. In the case of disco-dop's LCFRS parser (short: ddlcfrs), we limited the validation set to sentences

---

[4] A wMCFG G is called ε-free and simple if each composition function that occurs in the rules of G is either of the form $[u_1, \ldots, u_s]$ for some non-empty strings of variables $u_1, \ldots, u_s$, or of the form $[t]$ for some terminal symbol t.

[5] available on `https://github.com/tud-fop/rustomata`. We used commit `867a451` for evaluation.

[6] The evaluation scripts are available on `https://github.com/truprecht/rustomata-eval`.

parser	precision	recall	f_1-score	coverage
NeGra corpus, $\lvert w\rvert \leq 20$				
ddctf-dop	81.34	81.44	81.39	100%
ddlcfrs	74.85	73.99	74.42	100%
ddctf-lcfrs	74.78	73.89	74.33	100%
OP	74.38	74.22	74.30	99.20%
GF	70.98	72.17	71.57	98.15%
rparse	71.15	41.67	52.56	76.31%
NeGra corpus, $\lvert w\rvert \leq 30$				
ddctf-dop	77.91	78.22	78.07	100%
ddctf-lcfrs	70.32	69.70	70.01	100%
OP	68.81	69.27	69.04	99.22%
GF	66.56	68.16	67.35	98.19%
rparse	71.19	23.73	35.59	57.33%
Lassy corpus, $\lvert w\rvert \leq 30$				
ddctf-dop	73.78	73.37	73.57	100%
ddctf-lcfrs	61.23	59.73	60.52	100%
OP	50.92	51.42	51.17	100%

Table 1: Precision, recall, f_1-score, and coverage

of at most 20 tokens, since ddlcfrs frequently exceeded 30 seconds of parse time for longer sentences in preliminary tests. Disco-dop's coarse-to-fine data-oriented parser (short: ddctf-dop) and disco-dop's coarse-to-fine LCFRS parser (short: ddctf-lcfrs) rarely exceeded 30 seconds of parse time in preliminary tests and we let them run on sentences of up to 30 tokens without the timeout.

Figure 4a shows the parse times for each sentence length and parser. The parsers ddctf-dop, ddctf-lcfrs, GF, and OP perform similar for sentences of up to 20 tokens. The parse times of rparse and ddlcfrs grow rapidly after 10 and 16 tokens, respectively. Rparse even exceeds the timeout for more than half of the test sentences that are longer than 15 tokens. For sentences with up to 30 tokens, the parse times of ddctf-dop, ddctf-lcfrs and OP seem to remain almost constant.

Table 1 shows the accuracy (i.e. precision, recall, and f_1-score) and the coverage (i.e. the percentage of sentences that could be parsed) for each parser on the validation set. We report these scores to assert a correct implementation of our parser and to compare the different approximation strategies (and our fallback mechanism) implemented in the parsers. The low coverage of rparse stems from the frequent occurrences of timeouts. They also depress the recall for rparse. For sentences with at most 20 tokens, ddlcfrs, ddctf-lcfrs and OP perform very similar. These three parsers are outperformed by ddctf-dop in all aspects. For sentences of up to 30 tokens, the scores of all tested parsers drop similarly. However, ddctf-dop's scores drop the least amount.

We repeated a part of the experiments with the Lassy corpus (Lassy Small, various kinds of written Dutch, 65,200 sentences, 975,055 tokens; van Noord et al., 2013). Since it is considerably larger than the NeGra corpus, we limited the experiments to one iteration of TFCV, and we only investigate OP, ddctf-lcfrs, and ddctf-dop. The results are shown in fig. 4b (parse time) and at the bottom of tab. 1 (accuracy). Figure 4b shows the difference of ddctf-lcfrs, ddctf-dop and OP in terms of parse times (which is not discernible in fig. 4a). This plot shows that OP maintains very small parse times – even for large copora – compared to the state-of-the-art parser disco-dop.

All in all, our parser performs comparable to state-of-the-art MCFG parsers (GF, rparse, ddlcfrs, ddctf-lcfrs) and, using the NeGra corpus, it shows excellent results in parse time and good results in accuracy. Moreover, our parser can deal with any ε-free and simple MCFG provided by an external tool, making it more flexible than disco-dop and rparse. However, we are not able to compete with ddctf-dop in terms of accuracy, since discontinuous data-oriented parsing is a more accurate formalism (van Cranenburgh and Bod, 2013).

6 Future Work

We see potential to improve the fallback mechanism explained in sec. 4. For now, we only considered reporting the first cow derivation. By introducing some *degree of consistency* of cow derivations, we could select a cow derivation that is closer to a derivation of G.

Since recognisable languages are closed under inverse homomorphisms, we can use any recognisable language as input for $\hom_G^{-1}$ (cf. fig. 2) without changing the rest of the pipeline. This is useful when the input of the parsing task is ambiguous, as in lattice-based parsing (e.g. Goldberg and Tsarfaty, 2008).

Moreover, since weighted recognisable languages are closed under inverse homomorphisms and scalar product, we can even use a weighted recognisable language as input for $\hom_G^{-1}$, as in the setting of Rastogi et al. (2016).

Acknowledgements

We thank our colleague Kilian Gebhardt as well as the anonymous reviewers for their insightful comments on drafts of this paper.

References

Krasimir Angelov and Peter Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 368–376.

Yehoshua Bar-Hillel, Micha Asher Perles, and Eli Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172.

François Barthélemy, Pierre Boullier, Philippe Deschamp, and Éric Villemonte de la Clergerie. 2001. Guided parsing of range concatenation languages. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, and Tomek Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the Workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics.

Noam Chomsky. 1956. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Noam Chomsky and Marcel Paul Schützenberger. 1963. The algebraic theory of context-free languages. *Computer Programming and Formal Systems, Studies in Logic*, pages 118–161.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270. Association for Computational Linguistics.

John Cocke and J. T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. techreport, Courant Institute of Mathematical Sciences, New York University.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate dop model. In *Proceedings of the 13th International Conference on Parsing Technologies*.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57.

Tobias Denkinger. 2017. Chomsky-Schützenberger parsing for weighted multiple context-free languages. *Journal of Language Modelling*, 5(1):3.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. *Proceedings of ACL-08: HLT*, pages 371–379.

John Edward Hopcroft and Jeffrey David Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*, 1st edition.

Mans Hulden. 2011. Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation. In *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*, pages 151–160.

Aravind K. Joshi. 1985. *Tree adjoining grammars: How much context-sensitivity is needed for characterizing structural descriptions?*, chapter 6. Cambdrige University Press.

Aravind Krishna Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

T. Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. techreport R-257, AFCRL.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Frederick Mosteller and John Wilder Tukey. 1968. Data analysis, including statistics. In G. Lindzey and E. Aronson, editors, *Handbook of Social Psychology*, volume 2. Addison-Wesley.

Gertjan van Noord, Gosse Bouma, Frank Van Eynde, Daniël de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. 2013. *Large Scale Syntactic Annotation of Written Dutch: Lassy*, pages 147–164. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California. Association for Computational Linguistics.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the 10th European Summer School in Logic, Language and Information. Workshop on Recent Advances in Corpus Annotation*.

Krishnamurti Vijay-Shanker, David Jeremy Weir, and Aravind Krishna Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, pages 104–111.

Ryo Yoshinaka, Yuichi Kaji, and Hiroyuki Seki. 2010. Chomsky-Schützenberger-type characterization of multiple context-free languages. In Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications*, pages 596–607.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

A Appendices

A.1 Additional Examples

Example 1 (continuing from p. 2)**.** Figure 5 shows graphical representations of a derivation and the corresponding term over composition functions. □

Example 7 (continuing from p. 4)**.** The following calculation reduces a word of $\mathcal{A}_G$ to ε using the equivalence relation $\equiv_{\mathfrak{P}_G}$ (abbreviated by $\equiv$) and thereby proves that it is an element of mD_G. In each step, we point out, which cell of $\mathfrak{P}_G$ was/were used. Note that the set obtained after two applications of $\equiv$ has two elements.

$$\begin{aligned}
&\Big\{ \llbracket^1_{\rho_1} \llbracket^1_{\rho_{1,1}} \llbracket^1_{\rho_4} \rrbracket^1_{\rho_4} \rrbracket^1_{\rho_{1,1}} \llbracket^1_{\rho_{1,2}} \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \rrbracket^1_{\rho_3} \rrbracket^1_{\rho_{1,2}} \\
&\llbracket^2_{\rho_{1,1}} \llbracket^2_{\rho_4} \rrbracket^2_{\rho_4} \rrbracket^2_{\rho_{1,1}} \llbracket^2_{\rho_{1,2}} \llbracket^2_{\rho_3} \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \rrbracket^2_{\rho_3} \rrbracket^2_{\rho_{1,2}} \rrbracket^1_{\rho_1} \Big\} \\
&\equiv \Big\{ \llbracket^1_{\rho_{1,1}} \llbracket^1_{\rho_4} \rrbracket^1_{\rho_4} \rrbracket^1_{\rho_{1,1}} \llbracket^1_{\rho_{1,2}} \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \rrbracket^1_{\rho_3} \rrbracket^1_{\rho_{1,2}} \\
&\qquad \llbracket^2_{\rho_{1,1}} \llbracket^2_{\rho_4} \rrbracket^2_{\rho_4} \rrbracket^2_{\rho_{1,1}} \llbracket^2_{\rho_{1,2}} \llbracket^2_{\rho_3} \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \rrbracket^2_{\rho_3} \rrbracket^2_{\rho_{1,2}} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_1}\} \in \mathfrak{P}_G) \\
&\equiv \Big\{ \llbracket^1_{\rho_4} \rrbracket^1_{\rho_4} \llbracket^2_{\rho_4} \rrbracket^2_{\rho_4}, \\
&\qquad\qquad \llbracket^1_{\rho_{1,2}} \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \rrbracket^1_{\rho_3} \rrbracket^1_{\rho_{1,2}} \\
&\qquad\qquad \llbracket^2_{\rho_{1,2}} \llbracket^2_{\rho_3} \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \rrbracket^2_{\rho_3} \rrbracket^2_{\rho_{1,2}} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_{1,1}}, \llbracket^2_{\rho_{1,1}}\} \in \mathfrak{P}_G) \\
&\equiv \Big\{ \varepsilon, \qquad \llbracket^1_{\rho_{1,2}} \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \rrbracket^1_{\rho_3} \rrbracket^1_{\rho_{1,2}} \\
&\qquad\qquad \llbracket^2_{\rho_{1,2}} \llbracket^2_{\rho_3} \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \rrbracket^2_{\rho_3} \rrbracket^2_{\rho_{1,2}} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_4}, \llbracket^2_{\rho_4}\} \in \mathfrak{P}_G) \\
&\equiv \Big\{ \varepsilon,\ \llbracket^1_{\rho_3} \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \rrbracket^1_{\rho_3} \llbracket^2_{\rho_3} \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \rrbracket^2_{\rho_3} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_{1,2}}, \llbracket^2_{\rho_{1,2}}\} \in \mathfrak{P}_G) \\
&\equiv \Big\{ \varepsilon, \quad \widetilde{\mathrm{b}} \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \quad \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_3}, \llbracket^2_{\rho_3}\} \in \mathfrak{P}_G) \\
&\equiv \Big\{ \varepsilon, \quad \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \quad \widetilde{\mathrm{d}} \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \Big\} \\
&\hfill \text{(because } \{\llbracket_{\mathrm{b}}\} \in \mathfrak{P}_G \text{ and } \widetilde{\mathrm{b}} = \llbracket_{\mathrm{b}} \rrbracket_{\mathrm{b}}) \\
&\equiv \Big\{ \varepsilon, \quad \llbracket^1_{\rho_{3,1}} \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \rrbracket^1_{\rho_{3,1}} \quad \llbracket^2_{\rho_{3,1}} \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \rrbracket^2_{\rho_{3,1}} \Big\} \\
&\hfill \text{(because } \{\llbracket_{\mathrm{d}}\} \in \mathfrak{P}_G \text{ and } \widetilde{\mathrm{d}} = \llbracket_{\mathrm{d}} \rrbracket_{\mathrm{d}}) \\
&\equiv \Big\{ \varepsilon, \qquad \llbracket^1_{\rho_5} \rrbracket^1_{\rho_5} \qquad \llbracket^2_{\rho_5} \rrbracket^2_{\rho_5} \Big\} \\
&\hfill \text{(because } \{\llbracket^1_{\rho_{3,1}}, \llbracket^2_{\rho_{3,1}}\} \in \mathfrak{P}_G) \\
&\equiv \{\varepsilon\} \qquad\qquad \text{(because } \{\llbracket^1_{\rho_5}, \llbracket^2_{\rho_5}\} \in \mathfrak{P}_G) \quad \square
\end{aligned}$$

A.2 Additional Proofs

Lemma 12. There is a bijection toCowD between $\mathcal{L}(G_{\mathrm{cf}})$ and $\mathrm{cowD}^{\mathrm{c}}_G$.

Proof. For each $\ell \in \mathbb{N}_+$, we define the partial function f_ℓ from $(\varDelta_G \cup \overline{\varDelta_G})^*$ to ℓ-cow derivations of G as follows: $f_\ell(u) = \rho\big\langle (i(\kappa), j(\kappa))/f_{j(\kappa)}(v_\kappa) \mid \kappa \in [n]\big\rangle$ if u is of the

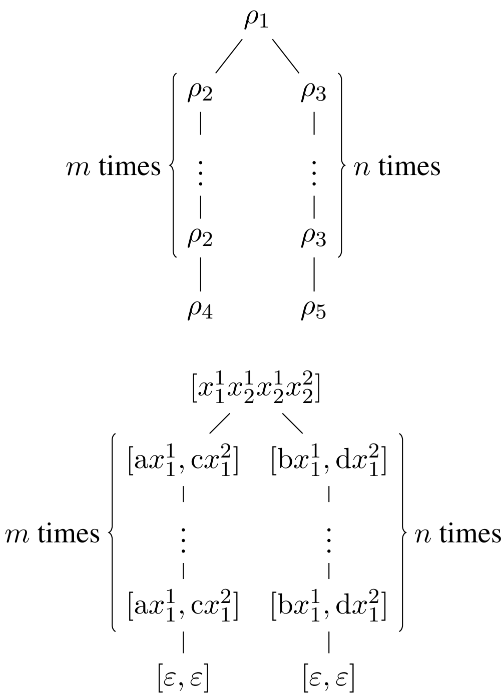


Figure 5: A derivation $d_{m,n}$ (top) together with the corresponding term over composition functions (bottom), cf. ex. 1.

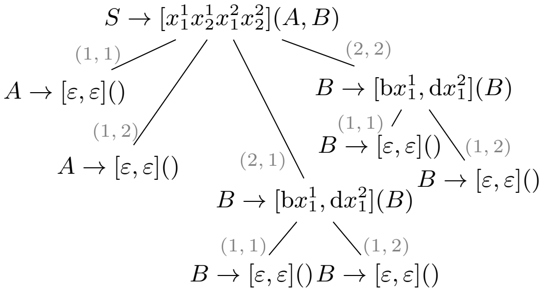


Figure 6: A consistent cow derivation.

form

$$\llbracket^{\ell}_{\rho}\ \widetilde{u_0}\llbracket^{j(1)}_{\rho,i(1)} v_1\rrbracket^{j(1)}_{\rho,i(1)}\widetilde{u_1}\cdots\llbracket^{j(n)}_{\rho,i(n)} v_n\rrbracket^{j(n)}_{\rho,i(n)}\widetilde{u_n}\ \rrbracket^{\ell}_{\rho}$$

for some rule $\rho = A \to c(B_1, \ldots, B_k)$ where the ℓ-th component of c is $u_0 x^{j(1)}_{i(1)} u_1 \cdots x^{j(n)}_{i(n)} u_n$ with $u_1, \ldots, u_n \in \Sigma^*$; otherwise, $f_\ell(u)$ is undefined. Note that $f_1, f_2, \ldots$ are pairwise disjoint (in the set-theoretic sense).

To prove that the function f_ℓ is bijective, we show that it is injective and surjective.

(Injectivity) We show, for each $\ell \in \mathbb{N}_+$, by induction on the structure of cow derivations that $f_\ell(v) = f_\ell(v')$ implies $v = v'$ for each v, v' in the domain of f_ℓ (i.e. $f_\ell(v)$ and $f_\ell(v')$ are both defined).

Let $v, v' \in (\Delta_G \cup \overline{\Delta_G})^*$ be in the domain of f_ℓ and let $t = f_\ell(v) = f_\ell(v')$. Furthermore, let $\rho = A \to c(B_1, \ldots, B_k) = \mathrm{root}(t)$, $t_{(i,j)} = \mathrm{sub}_{(i,j)}(t)$ for each $(i,j) \in \mathrm{out}(t)$, and let the ℓ-th component of c be of the form $u_0 x^{j(1)}_{i(1)} u_1 \cdots x^{j(n)}_{i(n)} u_n$ with $u_1, \ldots, u_n \in \Sigma^*$. By definition of f_ℓ, we know that u is the string

$$\llbracket^{\ell}_{\rho}\ \widetilde{u_0}\llbracket^{j(1)}_{\rho,i(1)} v_1\rrbracket^{j(1)}_{\rho,i(1)}\widetilde{u_1}\cdots\llbracket^{j(n)}_{\rho,i(n)} v_n\rrbracket^{j(n)}_{\rho,i(n)}\widetilde{u_n}\ \rrbracket^{\ell}_{\rho}$$

for some $v_1, \ldots, v_n \in (\Delta_G \cup \overline{\Delta_G})^*$, u' is the string

$$\llbracket^{\ell}_{\rho}\ \widetilde{u_0}\llbracket^{j(1)}_{\rho,i(1)} v'_1\rrbracket^{j(1)}_{\rho,i(1)}\widetilde{u_1}\cdots\llbracket^{j(n)}_{\rho,i(n)} v'_n\rrbracket^{j(n)}_{\rho,i(n)}\widetilde{u_n}\ \rrbracket^{\ell}_{\rho}$$

for some $v'_1, \ldots, v'_n \in (\Delta_G \cup \overline{\Delta_G})^*$, and $f_{j(\kappa)}(v_\kappa) = f_{j(\kappa)}(v'_\kappa) = t_{(i(\kappa),j(\kappa))}$ for each $\kappa \in [n]$. By principle of induction, we get $v_\kappa = v'_\kappa$ for each $\kappa \in [n]$. Hence $v = v'$.

(Surjectivity) We show by induction on the structure of cow derivations that for each $A \in N$, $\ell \in \mathit{sort}(A)$, and $t \in \ell\text{-cowD}^A_G$, there is a string $v \in \mathcal{L}(G_{\mathrm{cf}}, A^\ell)$ such that $f_\ell(v) = t$.

Let $A \in N$, $\ell \in \mathit{sort}(A)$, and $t \in \ell\text{-cowD}^A_G$. Furthermore, let $\rho = A \to c(B_1, \ldots, B_k) = \mathrm{root}(t)$, $t_{(i,j)} = \mathrm{sub}_{(i,j)}(t)$ for each $(i,j) \in \mathrm{out}(t)$, and let the ℓ-th component of c be of the form $u_0 x^{j(1)}_{i(1)} u_1 \cdots x^{j(n)}_{i(n)} u_n$ with $u_1, \ldots, u_n \in \Sigma^*$. By principle of induction, we know that there are $v_\kappa \in \mathcal{L}(G_{\mathrm{cf}}, B^{j(\kappa)}_{i(\kappa)})$ with $f_{j(\kappa)}(v_\kappa) = t_{(i(\kappa),j(\kappa))}$ for each $\kappa \in [n]$. Now let v be the string

$$\llbracket^{\ell}_{\rho}\ \widetilde{u_0}\llbracket^{j(1)}_{\rho,i(1)} v_1\rrbracket^{j(1)}_{\rho,i(1)}\widetilde{u_1}\cdots\llbracket^{j(n)}_{\rho,i(n)} v_n\rrbracket^{j(n)}_{\rho,i(n)}\widetilde{u_n}\ \rrbracket^{\ell}_{\rho}.$$

By definition of the rule $\rho^{(\ell)}$ (def. 8), we know that $v \in \mathcal{L}(G_{\mathrm{cf}}, A^\ell)$. By definition of f_ℓ, we know that $f_\ell(v) = t$. Hence, f_ℓ is surjective.

It is easy to see that $\mathrm{toCowD} = \bigcup_{\ell \in \mathbb{N}_+} f_\ell$. If we restrict the domain of toCowD to $\mathcal{L}(G_{\mathrm{cf}})$, then (since each element of $\mathcal{L}(G_{\mathrm{cf}})$ starts with a bracket of the form $\llbracket^1_\rho$ for some $\rho \in P$) the resulting function is a subset of f_1. Since f_1 is bijective, we know that toCowD is a bijection between $\mathcal{L}(G_{\mathrm{cf}})$ and $\mathrm{cowD}^{\mathrm{c}}_G$. ■

Lemma 16. Let $v \in \mathcal{L}(G_{\mathrm{cf}})$. Then

$$v \in \mathrm{mD}_G \iff \mathrm{toCowD}(v) \text{ is consistent.}$$

Proof. Let $s \in \mathbb{N}_+$ and let $t^1, \ldots, t^s$ be cow derivations in G. We abbreviate the following property by $\mathfrak{C}(t^1, \ldots, t^s)$: (i) $\{t^1, \ldots, t^s\}$ is consistent, (ii) $s = \mathrm{fanout}(\mathrm{root}(t^1))$, and (iii) t^ℓ is an ℓ-cow derivation for each $\ell \in [s]$. Now, we show by structural induction that $\mathfrak{C}(\mathrm{toCowD}(v^1), \ldots, \mathrm{toCowD}(v^s))$ holds iff each permutation of $v^1, \ldots, v^s$ is in mD_G. Let $\rho = A \to [c_1, \ldots, c_s](B_1, \ldots, B_k)$ be a rule in P. Let $v_i^j \in \mathcal{L}(G, B_i^j)$ for each $i \in [k]$ and $j \in [sort(B_i)]$ such that

$$\begin{aligned} &\mathrm{Per}(v_i^1, \ldots, v_i^{sort(B_i)}) \subseteq \mathrm{mD}_G \\ &\quad \iff \mathfrak{C}\big(t_i^1, \ldots, t_i^{sort(B_i)}\big) \end{aligned} \tag{IH}$$

for each $i \in [k]$ where t_i^j abbreviates $\mathrm{toCowD}(v_i^j)$ and $\mathrm{Per}(v_i^1, \ldots, v_i^{sort(B_i)})$ is the set of permutations of $v_i^1, \ldots, v_i^{sort(B_i)}$. For each $\ell \in [s]$, let v^ℓ be obtained from the right-hand side of $\rho^{(\ell)}$ by replacing each non-terminal of the form B_i^j by v_i^j. Clearly, for each $\ell \in [s]$, the sequence v^ℓ is an element of $\mathcal{L}(G, A^\ell)$. From now on, we will abbreviate $\mathrm{toCowD}(v^\ell)$ by t^ℓ for each $\ell \in [s]$. Now, let $B_{i(\ell,1)}^{j(\ell,1)}, \ldots, B_{i(\ell,n_\ell)}^{j(\ell,n_\ell)}$ be the nonterminals on the rhs of $\rho^{(\ell)}$, then t^ℓ is the cow derivation

$$\rho\big\langle (i(\ell,\kappa), j(\ell,\kappa))/t_{i(\ell,\kappa)}^{j(\ell,\kappa)} \quad \big| \quad \kappa \in [n_\ell]\big\rangle.$$

Note that ρ is the root symbol of each $t^1, \ldots, t^s$ and the set of indices $\{(i(\ell,\kappa), j(\ell,\kappa)) \mid \ell \in [s], \kappa \in [n_\ell]\}$ is the set $\{(i,j) \mid i \in [k], j \in [sort(B_i)]\}$. We derive

$$\begin{aligned} &\mathfrak{C}(t^1, \ldots, t^s) \\ &\iff \{t^1, \ldots, t^s\} \text{ is consistent} && \text{(by def. 10)} \\ &\iff \forall i \in [k]\colon \{t_i^1, \ldots, t_i^{sort(B_i)}\} \text{ is consistent} && \text{(by def. 14)} \\ &\iff \forall i \in [k]\colon \mathrm{Per}(v_i^1, \ldots, v_i^{sort(B_i)}) \subseteq \mathrm{mD}_G && \text{(by eq. (IH))} \\ &\iff \mathrm{Per}(v^1, \ldots, v^s) \subseteq \mathrm{mD}_G && \text{(by defs. 6 and 8)} \end{aligned}$$

Since the v_i^js and ρ were selected arbitrarily, we can obtain any element of $\mathcal{L}(G, A^\ell)$ in that manner. In particular, for each $v^1 \in \mathcal{L}(G_{\mathrm{cf}}, S^1) = \mathcal{L}(G_{\mathrm{cf}})$, the cow derivation $\mathrm{toCowD}(v^1)$ is consistent iff v^1 is in mD_G. ■

Proposition 15. TODERIV $\circ\, \mathrm{toCowD}^{-1}$ is a bijection between the consistent cow derivations in $\mathrm{cowD}_G^{\mathrm{c}}$ and $\mathrm{D}_G^{\mathrm{c}}$.

Proof. By lems. 12 and 16, there is a bijection between the consistent cow derivations in $(1\text{-cowD}_G)_S$ and $\mathcal{R}_G \cap \mathrm{mD}_G$, and there is a bijection between $\mathcal{R}_G \cap \mathrm{mD}_G$ and $\mathrm{D}_G^{\mathrm{c}}$ (Denkinger, 2017, cor. 3.9). ■

A.3 Additional Algorithms

Algorithm 3 is a modification of the algorithm given by Hulden (2011, alg. 1). The changes involve an introduction of weights in the algorithm; elements of A are drawn by maximum weight instead of being drawn randomly. In our implementation, we defined the weight of an item (p, v, q) as the weight $\mu'(v)$ defined in def. 6.

Algorithm 3 extracts Dyck words from an FSA.

Input: a weight assignment $wt\colon Q \times (\Delta \cup \overline{\Delta})^* \times Q \to \mathcal{M}$, and an automaton $\mathcal{A} = (Q, \Delta \cup \overline{\Delta}, q_{\mathrm{init}}, q_{\mathrm{final}}, T)$

Output: a sequence $v_1, v_2, \ldots$ of elements in $\mathcal{L}(\mathcal{A}) \cap \mathrm{D}_\Delta$ such that $wt(q_{\mathrm{init}}, v_1, q_{\mathrm{final}}) \trianglerighteq wt(q_{\mathrm{init}}, v_2, q_{\mathrm{final}}) \trianglerighteq \ldots$

```
procedure EXTRACTDYCK($wt$)($\mathcal{A}$)
  $(A, C) \leftarrow (\varnothing, \varnothing)$
  for $(p, \delta, q), (q, \overline{\delta}, r) \in T$ do
    $A \leftarrow A \cup \{(p, \delta\overline{\delta}, r)\}$
  for $(p, v, q) \in \mathrm{argmax}_{\trianglelefteq, wt} A$ do
    if $(p, q) = (q_{\mathrm{init}}, q_{\mathrm{final}})$ then yield $v$
    $A \leftarrow A \setminus \{(p, v, q)\}$
    $C \leftarrow C \cup \{(p, v, q)\}$
    for $(r, \delta, p), (q, \overline{\delta}, s) \in T$ do
      $A \leftarrow A \cup \{(r, \delta v \overline{\delta}, s)\}$
    for $(q, w, r) \in C$ do
      $A \leftarrow A \cup \{(p, vw, r)\}$
    for $(o, u, p) \in C$ do
      $A \leftarrow A \cup \{(o, uv, q)\}$
```

Since the function given in alg. 4 is very similar to def. 14, we omit further discussions.

Algorithm 4 checks consistency of a set of cow derivations.

Input: a set T of cow derivations

Output: $\begin{cases} \text{true} & \text{if } T \text{ is consistent} \\ \text{false} & \text{otherwise} \end{cases}$

```
function ISCONSISTENT($T$)
  if not $\bigwedge_{t,t' \in T} \big(\mathrm{root}(t) = \mathrm{root}(t')\big)$ then
    return false
  $(T_1, \ldots, T_k) \leftarrow$ COLLECTCHILDREN($T$)
  return $\bigwedge_{i=1}^{k}$ ISCONSISTENT'($T_i$)
```

			candidate count			
			10^2	10^3	10^4	10^5
beam width	20	median parse time	8.04 ms	8.55 ms	8.52 ms	9.61 ms
		mean parse time	8.18 ms	11.48 ms	34.71 ms	256.08 ms
		f_1-score	67.83 %	68.44 %	68.45 %	68.80 %
		coverage (no fallback)	81.20 %	87.49 %	90.40 %	92.10 %
		coverage (with fallback)	97.31 %	97.31 %	97.31 %	97.31 %
	50	median parse time	13.23 ms	15.26 ms	15.44 ms	15.77 ms
		mean parse time	13.11 ms	17.99 ms	53.51 ms	299.63 ms
		f_1-score	70.54 %	71.82 %	72.17 %	72.46 %
		coverage (no fallback)	78.36 %	85.88 %	90.25 %	93.47 %
		coverage (with fallback)	98.31 %	98.31 %	98.31 %	98.31 %
	100	median parse time	19.68 ms	23.18 ms	24.43 ms	24.67 ms
		mean parse time	18.81 ms	25.11 ms	70.12 ms	374.90 ms
		f_1-score	70.27 %	71.43 %	71.99 %	72.86 %
		coverage (no fallback)	76.82 %	84.11 %	88.72 %	92.93 %
		coverage (with fallback)	98.93 %	98.93 %	98.93 %	98.93 %
	200	median parse time	31.33 ms	36.04 ms	**39.75 ms**	40.05 ms
		mean parse time	30.10 ms	35.80 ms	**85.94 ms**	412.16 ms
		f_1-score	70.50 %	71.50 %	**72.13 %**	72.98 %
		coverage (no fallback)	76.66 %	83.57 %	**88.33 %**	92.79 %
		coverage (with fallback)	99.00 %	99.00 %	**99.00 %**	99.00 %
	500	median parse time	53.16 ms	58.07 ms	68.22 ms	68.86 ms
		mean parse time	50.21 ms	56.70 sm	105.94 ms	431.90 ms
		f_1-score	70.51 %	71.45 %	72.08 %	72.90 %
		coverage (no fallback)	76.66 %	83.50 %	88.26 %	92.79 %
		coverage (with fallback)	99.00 %	99.00 %	99.00 %	99.00 %
	1000	median parse time	53.10 ms	58.59 ms	76.07 ms	77.37 ms
		mean parse time	53.53 ms	59.26 ms	109.46 ms	443.56 ms
		f_1-score	70.51 %	71.45 %	72.08 %	72.90 %
		coverage (no fallback)	76.66 %	83.50 %	88.26 %	92.79 %
		coverage (with fallback)	99.00 %	99.00 %	99.00 %	99.00 %

Table 2: Results of the grid search for meta-parameters.

A.4 Results of Grid Search for Meta-Parameters

Table 2 shows the results of the grid search for the two introduced meta-parameters. For each combination of *beam width* and *candidate count*, we list the *median* and *mean parse time*s (since medians hide outliers, those two may differ drastically) for all sentences of length 20 and f_1-*score* over all test sentences. Moreover, we show the percentage of sentences (*coverage*) that we were able to parse with and without the fallback mechanism. The results for the combination of meta-parameters that was selected for later experiments (i.e. a beam width of 200 and a candidate count of 10^4) are written in bold.

Phylogenetic Multi-Lingual Dependency Parsing

Mathieu Dehouck
Univ. Lille, CNRS, UMR 9189 - CRIStAL
Magnet Team, Inria Lille
59650 Villeneuve d'Ascq, France
mathieu.dehouck@inria.fr

Pascal Denis
Magnet Team, Inria Lille
59650 Villeneuve d'Ascq, France
pascal.denis@inria.fr

Abstract

Languages evolve and diverge over time. Their evolutionary history is often depicted in the shape of a phylogenetic tree. Assuming parsing models are representations of their languages grammars, their evolution should follow a structure similar to that of the phylogenetic tree. In this paper, drawing inspiration from multi-task learning, we make use of the phylogenetic tree to guide the learning of multi-lingual dependency parsers leveraging languages structural similarities. Experiments on data from the Universal Dependency project show that phylogenetic training is beneficial to low resourced languages and to well furnished languages families. As a side product of phylogenetic training, our model is able to perform zero-shot parsing of previously unseen languages.

1 Introduction

Languages change and evolve over time. A community that spoke once a single language can be split geographically or politically, and if the separation is long enough their language will diverge in direction different enough so that at some point they might not be intelligible to each other. The most striking differences between related languages are often of lexical and phonological order but grammars also change over time.

Those divergent histories are often depicted in the shape of a tree in which related languages whose common history stopped earlier branch off higher than languages that have shared a longer common trajectory (Jäger, 2015). We hypothesize that building on this shared history is beneficial when learning dependency parsing models. We thus propose to use the phylogenetic structure to guide the training of multi-lingual graph-based neural dependency parsers that will tie parameters between languages according to their common history.

As our phylogenetic learning induces parsing models for every inner node in the phylogenetic tree, it can also perform zero-shot dependency parsing of unseen languages. Indeed, one can use the model of the lowest ancestor (in the tree) of a new language as an approximation of that language grammar.

We assess the potential of phylogenetic training with experiments on data from the Universal Dependencies project version 2.2. Our results show that parsers indeed benefit from this multi-lingual training regime as models trained with the phylogenetic tree outperform independently learned models. The results on zero-shot parsing show that a number of factors such as the genre of the data and the writing system have a significant impact on the quality of the analysis of an unseen language, with morphological analysis being of great help.

The remaining of this paper is organized as follows. Section 2 presents both the neural parsing model as well as the phylogenetic training procedure. Section 3 presents some experiments over data from UD 2.2. Section 4 presents some related works on multi-task learning and multi-lingual parsing. Finally, Section 5 closes the paper and gives some future perspectives.

2 Model

We propose a multi-task learning framework that shares information between tasks using a tree structure. The tree structure allows us to both share model parameters and training samples between related tasks. We instantiate it with a graph-based neural parser and use the language phylogenetic tree to guide the learning process, but it can in principle be used with any tree that encodes tasks relateness and any learning algorithm that supports fine-tuning.

In this section we first describe the intuition be-

Proceedings of NAACL-HLT 2019, pages 192–203
Minneapolis, Minnesota, June 2 - June 7, 2019.

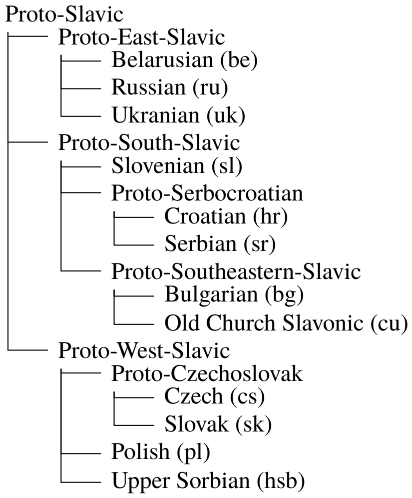


Figure 1: A possible phylogenetic tree for languages in the Slavic family.

hind phylogenetic training, then the neural parser and then the phylogenetic training itself.

2.1 Phylogenetic Hypothesis

Languages evolve from earlier stages and sometimes a language will change differently in different places leading to different languages with a common ancestor. This evolution process is often depicted in the shape of a tree in which leaves are actual languages and inner nodes can be either attested ancestral languages or their idealized reconstruction. Figure 1 gives an example of such a tree for a subset of the Slavic family of Indo-European languages (Simons and Fennig, 2018).

Just as languages evolve and diverge, so do their grammars. Assuming a parsing model is a parameterized representation of a grammar, then we can expect those models to evolve in a similar way. We thus take a multi-task approach to the problem of multi-lingual dependency parsing. What was once a single problem (e.g. parsing sentences in Proto-West-Slavic) becomes a set of distinct but related problems (parsing sentences in Czech, Polish, Slovak and Sorbian) as Proto-West-Slavic was evolving into its modern descendants.

We assume that the grammar of the last common ancestor is a good approximation of those languages grammars. Thus it should be easier to learn a language's grammar starting from its ancestor grammar than from scratch. There are however some issues with this assumption. First, a language grammar can be very different from its ancestor one from two millennia earlier. Consider the difference between modern French and early Classical Latin for example, in two millennia Latin has witnessed the loss of its case system and a complete refoundation of its verbal system. And the ”last common” ancestors can have very different age depending on the languages we consider. We expect the common ancestor of Tagalog and Indonesian to be much much older than the common ancestor of Portuguese and Galician. Second, a lot of languages have only started to be recorded very recently thus lacking historical data all together. And when historical records are available, much work still needs to be done to render those data usable by parsers. For example the Universal Dependencies Project (Nivre et al., 2018) only has annotated corpora for Latin, old Greek, old Church Slavonic and Sanskrit. And even for those classical languages, it is not clear to which extent their modern counterparts really descend from them. Thus we need to find another way to access the ancestor language grammar than using historical data.

We propose to use all the data from descendent languages to represent an ancestor language. In principle, one could give more weight to older languages or to languages that are known to be more conservative, but this knowledge is not available for all languages families. Thus we resort to using all the available data from descendent languages without distinction.

Another problem is that the tree view is too simple to represent the complete range of phenomena involved in language evolution, such as language contacts. Furthermore, languages do not evolve completely randomly, but follow some linguistic universals and have to keep a balance between speakability, learnability and understandability. Thus, languages can share grammatical features without necessarily being genetically related, either by contact or by mere chance. However, the tree model is still a good starting point in practice and language families align well with grammatical similarity as recent works on typological analysis of UD treebanks have shown (Chen and Gerdes, 2017; Schluter and Agić, 2017). We thus make the simplifying assumption that a language grammar evolves only from an older stage and can be approximated by that previous stage.

2.2 Neural Model

Our scoring model is an edge factored graph-based neural model in the vein of recent works by Dozat et al. (Dozat et al., 2017). There are two major differences here compared to the parser of Dozat

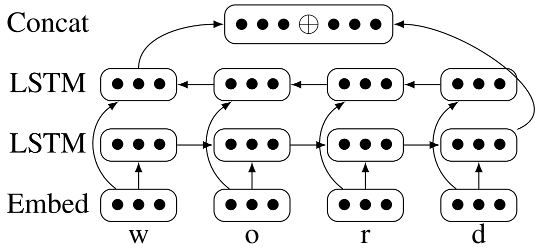


Figure 2: Bi-LSTM architecture for character based word representation. The final representation is the concatenation of the final cells of each layer.

et al. The first difference is in individual word representation, for which we use only the UPOS[1] tag, morphological information provided by UD treebanks and a character based word representation, whilst Dozat et al. use also the XPOS[2] tag, holistic word vectors (from Word2Vec (Mikolov et al., 2013) and their own) and they do not use morphological information beside what might already be given by the XPOS. The second difference is the scoring function proper. While they use biaffine scoring functions and decouple edge scoring from label scoring, we use a simple multi-layer perceptron to compute label scores and pick the max over the label as the edge score.

Let $x = (w_1 w_2 ... w_l)$ be a sentence of length l. Each word w_i is represented as the concatenation of 3 subvectors, one for its part-of-speech, one for its morphological attributes and one for its form:

$$\mathbf{w}_i = \mathbf{pos}_i \oplus \mathbf{morph}_i \oplus \mathbf{char}_i.$$

The part-of-speech vector ($\mathbf{pos}_i$) is from a look up table. The morphological vector ($\mathbf{morph}_i$) is the sum of the representation $\mathbf{m}_m$ of each morphological attribute m of the word given by the treebanks:

$$\mathbf{morph}_i = \sum_{m \in morph_i} \mathbf{m}_m.$$

We add a special dummy attribute representing the absence of morphological attributes.

The form vector ($\mathbf{char}_i$) is computed by a character BiLSTM (Hochreiter and Schmidhuber, 1997). Characters are fed one by one to the recurrent neural network in each direction. The actual form vector is then the concatenation of the outputs of the forward character LSTM and of the backward character LSTM as depicted in Figure 2.

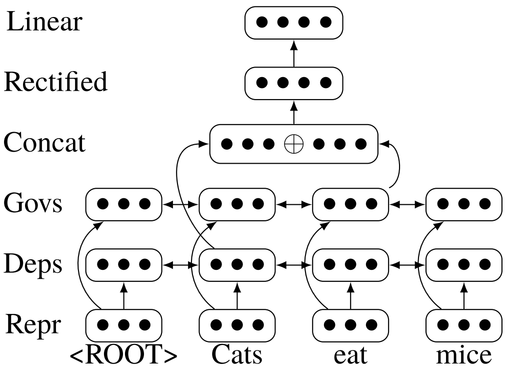


Figure 3: Neural network architecture for edge scoring. The contextualised representation of the governor (eat) and the dependent (Cats) are concatenated and passed through a rectified linear layer and a final plain linear layer to get a vector of label scores.

Once, each word has been given a representation in isolation, those representations are passed to two other BiLSTMs. Each word is then represented as the concatenation of its contextualised vector from the forward and backward layers:

$$\mathbf{c}_i = forward(\mathbf{w}_1, ..., \mathbf{w}_i) \oplus backward(\mathbf{w}_i, ..., \mathbf{w}_l).$$

We actually train two different BiLSTMs, one representing words as dependents ($\mathbf{c}$) and one words as governors ($\hat{\mathbf{c}}$). An edge score is then computed as follows. Its governor word vector $\hat{\mathbf{c}}_i$ and its dependent word vector $\mathbf{c}_j$ are concatenated and fed to a two layer perceptron (whose weights are $\mathbf{L}_1$ and $\mathbf{L}_2$) with a rectifier (noted $[...]^+$) after the first layer in order to compute the score s_{ijl} of the edge for every possible relation label l:

$$s_{ij} = \max_l s_{ijl} = \max_l (\mathbf{L}_2 \cdot [\mathbf{L}_1 \cdot (\hat{\mathbf{c}}_i \oplus \mathbf{c}_j)]^+)_l.$$

All the neural model parameters $\boldsymbol{\theta}$ (part-of-speech, character and morphological embeddings, character, dependant and governor BiLSTMs and the two layer perceptron weights) are trained end to end via back propagation one sentence at a time. Given a sentence x, we note j the index of the governor of w_i and l the relation label of w_i, the loss function is:

$$loss(x) = \sum_{w_i} \Big[\sum_{\substack{j' \neq j \\ j' \neq i}} \max(0, s_{ij'} - s_{ij} + 1)^2 + \sum_{l' \neq l} \max(0, s_{ijl'} - s_{ijl} + 1)^2 \Big]$$

---

[1] Universal part-of-speech for a set of 17 tags. Does not encode morphology.

[2] Language specific part-of-speech. Might include morphological information, but is not available for all languages.

For each word, there are two terms. The first term enforces that for all potential governors that are neither the word itself nor its actual governor, their highest score (irrespective of the relation label) should be smaller than the score of the actual governor and actual label by a margin of 1. The second term is similar and enforces that for the actual governor, any label that is not the true label should have a score smaller than the score of the actual label again by a margin of 1.

2.3 Phylogenetic Training

Let $\mathcal{L} = \{l_1, l_2, ..., l_{n_l}\}$ be a set of n_l languages and let $\mathcal{P} = \{p_1, p_2, ..., p_{n_p}\}$ be a set of n_p proto-languages (hypothesized ancestors of languages in $\mathcal{L}$). Let $\mathcal{T}$ be a tree over $\mathcal{L}^* = \mathcal{L} \cup \mathcal{P}$ such that languages of $\mathcal{L}$ are leaves and proto-languages of $\mathcal{P}$ are inner nodes. This means that we assume no two languages in $\mathcal{L}$ share a direct parenthood relation, but they at best descend both from a hypothesized parent. We could in principle have data appearing only in inner nodes. Tree $\mathcal{T}$ has a single root, a proto-language from $\mathcal{P}$ that all grammars descend from. This ancestor of all languages shall model linguistic universals[3] and ensure we deal with a well formed tree. We use the notation $p > l$ for the fact that language/node l descends from language/node p.

For each language $l \in \mathcal{L}$, we assume access to a set of n annotated examples $\mathcal{D}_l$. For each proto-language $p \in \mathcal{P}$, we create an annotated set $\mathcal{D}_p = \bigcup_{p>l} \mathcal{D}_l$ as the union of its descendent sets.

For each language $l \in \mathcal{L}^*$, we want to learn a parsing model $\boldsymbol{\theta}_l$.

2.3.1 Model Evolution

The main idea behind phylogenetic training is to initialize a new model with the model of its parent, thus effectively sharing information between languages and letting models diverged and specialize over time. The training pocedure is summarized in Algorithm 1.

At the beginning, we initialize a new blank/random model that will be the basic parsing model for all the world languages. Then, we sample sentences (we will discuss sampling issues in next section) randomly from all the available languages, parse them, compute the loss and update the model accordingly. Since the training sentences are sampled from all the available languages, the model will learn to be as good as possible for all the languages at the same time.

```
Data: a train set $\mathcal{D}_l$ and a dev set $\mathcal{D}'_l$ per
      language, a tree $\mathcal{T}$, two sampling
      sizes $k$, $k'$ and a maximum number
      of reboot $r$
Result: a model $\boldsymbol{\theta}$ per node in $\mathcal{T}$
begin
    Instantiate empty queue $Q$
    $Q.push(\mathcal{T}.root)$
    while $Q$ is not empty do
        $l = Q.pop()$
        if $l = \mathcal{T}.root$ then
            initialize $\boldsymbol{\theta}^0_{\mathcal{T}.root}$ randomly
        else
            $\boldsymbol{\theta}^0_l = \boldsymbol{\theta}_{l.parent}$
        $reboot = 0, i = 1, a_0 = 0$
        while $reboot < r$ do
            $\boldsymbol{\theta}^i_l = train(\boldsymbol{\theta}^{i-1}_l, \mathcal{D}_l, k)$
            $a_i = test(\boldsymbol{\theta}^i_l, \mathcal{D}'_l, k')$
            if $a_i \leq a_{i-1}$ then
                $reboot \mathrel{+}= 1$
            else
                $reboot = 0, i \mathrel{+}= 1$
        $\boldsymbol{\theta}_l = \boldsymbol{\theta}^i_l$
        for $c$ in l.children do
            $Q.push(c)$
```

Algorithm 1: Phylogenetic training procedure.

When the model $\boldsymbol{\theta}_p$ has reached an optimum (that we defined hereafter), we pass a copy of it to each of its children. Thus, for each child c of p, we initialize $\boldsymbol{\theta}^0_c = \boldsymbol{\theta}_p$ to its parent (p) final state. Each model $\boldsymbol{\theta}_c$ is then refined on its own data set $\mathcal{D}_c$ which is a subset of $\mathcal{D}_p$, until it reaches its own optimum state and is passed down to its own children. This process is repeated until the model reaches a leaf language, where the model $\boldsymbol{\theta}_c$ is eventually refined over its mono-lingual data set $\mathcal{D}_c$.

By passing down optimal models from older/larger languages sets to newer/smaller ones, models get the chance to learn relevant information from many different languages while specializing as time goes by.

The question now is to find when to pass down a model to its children. In other words, at which stage has a model learned the most it could from its data and should start to diverge to improve again?

Following the principle of cross-validation, we

---

[3] It does not imply anything about our belief or not in the monoglotto genesis hypothesis.

propose to let held-out data decide when is the right time to pass the model down. Let $\mathcal{D}'_p$ be a set of held-out sentences from the same languages as $\mathcal{D}_p$. Then, after every epoch i of k training examples, we freeze the model $\boldsymbol{\theta}_p^i$, and test it on k' sentences from $\mathcal{D}'_p$. This gives a score a_i (UAS/LAS) to the current model. If the score is higher than the score of the previous model $\boldsymbol{\theta}_p^{i-1}$ then training goes on, otherwise we discard it and retrain $\boldsymbol{\theta}_p^{i-1}$ for another k sentences. If after having discarded r epochs in a raw we have not yet found a better one, then we assume we have reached an optimal model $\boldsymbol{\theta}_p^{i-1}$ and pass it on to its children (unless it is a leaf, in which training is over for that language).

2.3.2 Sentence Sampling

There are a few things we should consider when drawing examples from a proto-language distribution. Beside the question of whether some languages are more conservative than others with respect to their ancestor, which we have decided to simplify saying that all languages are as representative of their ancestors, there is the problem of data unbalance and tree unbalance.

Sampling sentences uniformly across languages is not a viable option for the size of datasets varies a lot across languages and that they do not correlate with how close a language is to its ancestor. For example, there are 260 Belarusian training sentences against 48814 Russian ones. The basic question is thus whether one should draw examples from languages or branches. Basic linguistic intuition tells us that drawing should be performed on branches. Modern languages distribution has no reason to reflect their proximity to their ancestor language. Amongst Indo-European languages, there are one or two Armenian languages as well as one or two Albanian languages (depending on the criteria for being a language), while there are tens of Slavic languages and Romance languages. However, there is no reason to believe that Slavic or Romance languages are better witnesses of proto-Indo-European than Armenian or Albanian.

Drawing examples from languages would bias the intermediate models toward families that have more languages (or more treebanks). It might be a good bias depending on the way one compute the overall accuracy of the system. If one uses the macro-average of the individual language parsers, then biasing models toward families with many members should improve the accuracy overall.

In this work, at a given inner node, we decided to sample uniformly at random over branches spanning from this node, then uniformly at random over languages and then uniformly at random over sentences. It boils down to flattening the subtree below an inner node to have a maximum depth of 2. For example in Figure 1, at the root (Proto-Slavic) we pick a branch at random (e.g. Proto-South-Slavic), then a language at random (e.g. Croatian) then a sentence at random. Given that we have picked the Proto-South-Slavic branch, all South-Slavic languages are then as likely to be chosen. This biases slightly the model toward bigger sub-families. In our example, Croatian and Serbian have the same chances to be sampled than Slovenian, therefore their family, Proto-Serbocroatian is twice as likely to be chosen as Slovenian is, while being at the same depth in the tree.

We could otherwise sample over branches, then over sub-branches again and again until we reach a leaf and only then pick a sentence. In this case, Proto-Serbocroatian and Slovenian would have the same probability to be chosen. This would give much more weight to languages high in the tree than languages low in the tree. While this would give more balance to the actual model, it could be detrimental to the averaged results since the data distribution is itself unbalanced. It would of course be possible to try any variation between those two, picking sub-branches according to a probability that would depend on the number of languages in that family for example, therefore mitigating the unbalance problem.

2.4 Zero-Shot Parsing

An interesting property of the phylogenetic training procedure is that it provides a model for each inner node of the tree and thus each intermediary grammar. If one were to bring a new language with its position in the tree, then we can use the pre-trained model of its direct ancestor as an initialization instead of learning a new model from scratch. Similarly, one can use this ancestor model directly to parse the new language, effectively performing zero-shot dependency parsing. We investigate this possibility in the experiment section.

3 Experiments

To assess the potential of phylogenetic training both in terms of multi-task learning and zero-shot parsing capabilities, we experimented with data from the Universal Dependencies project version

2.2 (Nivre et al., 2018). When several corpora are available for a language, we chose one to keep a good balance between morphological annotation and number of sentences. For example, the Portuguese GSD treebank has slightly more sentences than the Bosque treebank but it is not well morphologically annotated. The zero-shot parsing models have been directly tested on languages that lack of training set. The treebanks names are given in the tree 4 and the result table 1.

3.1 Setting

As some languages have no training data and unique writing systems making the character model inefficient for them, we resorted to use gold parts-of-speech and morphological attributes rather than predicted ones. For example, Thai has no training data, no language relative and a unique script, which altogether make it really hard to parse (from a phylogenetic perspective).

The phylogenetic tree used for the experiment is adapted from the Ethnologue (2018). For space reasons, it is reported in the appendix in Figures 4 and 5. We tried to have a tree as consensual as possible, but there are still a few disputable choices, mostly about granularity and consistency. Sanskrit could have its own branch in the Indic family just as Latin in the Romance family, but because Sanskrit has no training data, that would not actually change the results. Likewise, as Czechoslovak and Dutch-Afrikaans have their own branches, Scandinavian languages could also distributed between east and west Scandinavian. As an English based Creole, Naija could as well be put in the Germanic family, but we kept it as a separate (Creole) family.

Regarding model training proper, we used $k = 500$ training sentences per iteration, $k' = 500$ held-out sentences from the developpement set to compute running LAS and a maximum number of reboot $r = 5$. Following Dozat et al (2017), we use Eisner algorithm (Eisner, 1996) at test time to ensure outputs are well formed trees. The neural model is implemented in Dynet (Neubig et al., 2017) and we use Adadelta with default parameters as our trainer. We averaged the results over 5 random initializations. Independent models are trained in the same manner but with mono-lingual data only. We report both labeled and unlabeled edge prediction accuracy (UAS/LAS). In the appendix we also report results averaged per family.

3.2 Multi-Task Learning

Table 1 reports parsing results for languages that have a training set. Note that a few languages do not have a separate developpement set, then we used the training set for both training and validation. The training set size of those languages is reported in square brackets. This has low to no impact on other languages results but it can be problematic for the language itself as it can over-fit its training data especially when they are very few as is the case of Buryat for example. To be fair, we report two different averages. Avg is the average over languages that have a separate developpement set, and Avg No Dev is the average over languages that do not have a separate developpement set. For each language, the best UAS/LAS are reported in bold.

On average, phylogenetic training improves parsing accuracy, both labeled and unlabeled. This is especially true for languages that have very small training sets (50 sentences or less) and lack of developpement set. Those languages show an averaged 7 points improvement and up to 15 points (hsb, kmr). Since independent mono-lingual models follow the exact same training procedure but without phylogenetic initialization and that every sentence will be seen several times both at training and validation, the sampling method cannot explain such a difference. This shows that the ancestor's model is a good initialization and acts as a form of regularization, slowing down over-fitting.

Phylogenetic training is also beneficial as one gains information from related languages. Indo-European languages gain from sharing information. This is especially true for Balto-Slavic (sk +5.82, lt +5.07 UAS) and Indo-Iranian languages (mr +2.05 UAS). It is less consistent for Romance and Germanic languages. This might be due to the tree not representing well typology for those families. Typically, English tends to group syntactically with Scandinavian languages more than with West-Germanic. Turkic and Uralic languages show the same benefits overall (ug +2.67, fi +3.39 UAS).

Dravidian and Afro-Asiatic languages are not as consistent. While Telugu seems to gain from Tamil data, the reverse is not true. Result variation for Arabic, Hebrew and Coptic are marginal. This is likely due to the fact that we only have three quite different languages from that family and that they all have their own script.

Similarly, phylogenetic training is not consistently useful for languages that do not have rela-

	Phylogenetic		Independent	
	UAS	LAS	UAS	LAS
ar nuyad	74.81	70.32	**75.07**	**71.08**
cop	85.51	79.28	**86.03**	**80.15**
he	**81.89**	75.36	81.59	**75.57**
bxr [19]	**48.72**	**30.68**	37.88	18.09
eu	76.81	69.51	**78.61**	**72.76**
af	85.15	80.94	**85.44**	**81.66**
da	78.50	72.50	**79.16**	**74.13**
de gsd	**80.37**	**73.54**	79.48	72.37
en ewt	79.25	74.34	**79.27**	**74.66**
got	77.83	71.54	**79.91**	**74.33**
nb	**84.62**	**78.78**	83.82	78.09
nl alpino	**77.19**	**68.55**	76.52	68.40
nn nynorsk	82.39	76.44	**82.58**	**77.32**
sv talbanken	80.46	74.62	**81.17**	**75.47**
be	**80.18**	**74.11**	78.09	72.76
bg	86.01	79.16	**86.40**	**79.79**
cs pdt	**79.78**	**71.71**	77.45	69.88
cu	82.98	77.19	**83.31**	**78.32**
hr	**81.70**	**74.73**	81.05	73.95
hsb [23]	**74.24**	**66.01**	58.59	50.37
lt	**61.42**	**50.88**	56.35	46.14
lv	**78.39**	**70.14**	76.69	68.89
pl lfg	**92.88**	**88.53**	91.07	86.49
ru syntagrus	**77.91**	72.72	77.33	**72.85**
sk	**84.91**	**79.17**	79.09	73.20
sl ssj	87.15	83.43	**88.39**	**85.21**
sr	85.85	79.86	**86.17**	**80.47**
uk	**78.16**	**73.50**	74.96	70.91
ca	84.67	78.81	**85.69**	**80.11**
es ancora	85.11	79.52	**85.61**	**80.18**
fr gsd	**84.35**	77.59	84.21	**77.94**
fro	**82.32**	**74.24**	78.91	69.95
gl [600] treegal	**83.80**	**78.06**	83.60	77.63
it isdt	87.03	81.67	**87.10**	**82.27**
la proiel	**66.25**	**58.88**	65.07	57.80
pt bosque	**84.93**	79.37	84.90	**79.83**
ro rrt	79.83	70.46	**79.93**	**70.88**
fa	78.76	72.95	**79.93**	**74.07**
hi hdtb	**89.32**	**82.89**	88.75	82.60
kmr [20]	**69.08**	**59.64**	54.77	45.07
mr	**78.65**	**68.97**	76.60	64.04
ur	84.32	77.02	**84.82**	**78.19**
el	86.44	83.30	**86.88**	**83.96**
grc proiel	**73.82**	**67.88**	71.68	66.05
ga [566]	75.91	67.54	**76.20**	**67.72**
hy [50]	**65.03**	**51.76**	59.27	46.67
id gsd	**81.08**	**74.97**	80.83	74.69
ja gsd	91.22	87.31	**91.40**	**87.37**
ko kaist	73.38	68.35	**74.23**	**69.81**
kk [31]	**70.82**	**55.42**	62.81	44.59
tr imst	**59.64**	**50.66**	59.00	50.54
ug	**66.33**	**48.20**	63.66	46.07
et	**75.32**	**68.13**	73.91	66.96
fi ftb	**78.05**	**72.20**	74.66	68.22
hu	79.51	72.88	**80.15**	**74.31**
sme [2257]	**80.13**	**76.40**	78.34	74.25
ta	75.05	66.94	**76.19**	**67.93**
te	**88.88**	**74.24**	87.01	72.05
vi	65.59	61.15	**66.02**	**61.74**
zh	**80.36**	**74.79**	80.14	74.52
Avg	80.05	73.35	79.47	73.02
Avg No Dev	70.97	60.69	63.93	53.05

Table 1: Parsing results for languages with a training set for phylogenetic models and independent models. The training set size of languages without a developpement set are reported in brackets.

Lang	Model	UAS	LAS
am	Semitic	57.27	26.25
br	Celtic*	61.36	43.89
fo	North-Germanic	52.40	46.52
sa	Indic	56.18	40.46
kpv lattice	Finno-Permiac*	65.16	52.11
pcm	World	60.43	43.80
th	World	29.14	17.61
tl	Austronesian*	70.89	50.38
wbp	World	87.67	65.66
yo	World	56.16	37.51
yue	Sino-Tibetan*	41.68	25.02
Avg		58.04	40.83

Table 2: Accuracy of languages without a training set.

tives. While Buryat (bxr) that has a very small training set benefits from universal linguistic information and gain almost 11 points UAS, Basque (eu) that has a very different grammatical structure than other languages and enough training data (5396 sentences) looses 3.25 LAS. Gains and losses are marginal for the other five languages (id, ja, ko, vi, zh).

Overall results are a bit below the state of the art, but the model is very simple and relies on gold morphology, so it is not really comparable.

3.3 Zero-Shot Parsing

Table 2 reports parsing results for languages that do not have a training set. Because of phylogenetic training and the tree structure that guides it, it can happen that a language ancestor's model is in fact trained on data only accounting for a narrow range of later stages. For example, while Faroese uses the North-Germanic model refined on both Norwegians, Swedish and Danish data, Tagalog uses the Austronesian model only refined with Indonesian data thus making it more an Indonesian model than an actual Austronesian model. Those cases are marked by an asterisk in the table. Komi (kpv) model is refined on Finno-Samic data, Breton (br) model on Irish data, Cantonese (yue) model on Mandarin data.

Looking at Table 2, we make the following observations. As expected scores are on average lower than for languages with training data, however the UAS/LAS gap is substantially bigger from 6.781 to 17.08 points. It is hard to compare to other works on zero-shot parsing since they use different data and scores span a big range, but our results are comparable to those of Aufrant et al. (2016) and Naseem et al. (2012), while our zero-shot models are given for free by the phylogenetic training method.

On a language per language basis, we see that there are a few important factors, the most striking being genre. Tagalog (tl) and more surprisingly Warlpiri (wbp) have relatively high parsing accuracy despite being either completely isolated or having only one relative (Indonesian). This is likely because their data are well annotated stereotypical sentences extracted from grammars, thus making them easy to parse.

Then we see that Naija (pcm) and Yoruba (yo) are about 25 points higher than Thai (th) despite them three having low morphology (in the treebanks). As they have different genres (spoken, bible, news and wiki), without a deeper look at the trees themselves, our best guess is that this is due to Thai having a different script. Naija and Yoruba both use the Latin alphabet, and as such they can rely to some extent on the character model to share information with other languages, to at least organise the character space. This analysis would also carry for Cantonese (yue). It is a morphologically simple language, and despite having a relative (Mandarin), its score is rather low. The genre alone (spoken) would not explain everything as Naija has also a spoken treebank and a higher score. The writing system might be at blame once again. Indeed, Chinese characters are very different from alphabetic characters and are much harder to use in character models because of sparsity. Comparing Mandarin and Cantonese test sets with Mandarin train set, the amount of out-of-vocabulary words is 32.47% of types (11.90% of tokens) for Mandarin and 54.88% of types (56.50% of tokens) for Cantonese. The results for out-of-vocabulary characters are even more striking with 3.73% of types (0.49% of tokens) for Mandarin and 12.97% of types (34.29% of tokens) for Cantonese. This shows that not only there are a lot of OOV in Cantonese test set, but that those words/characters are common ones as 12.97% of character types missing make up for more than a third of all character tokens missing, where on the contrary Mandarin OOV are seldom and account for less tokens percentage than types. This is one more argument supporting the importance of the character vector.

Other important factors are typology and morphology. Amharic (am) despite its unique script has a higher score than Cantonese that actually shares its scripts (to some extent as we have seen) with Mandarin. The key point for Amharic score, is that all its relatives (Hebrew, Arabic and Coptic) have their own scripts and are morphologically rich, thus the model learns to use morphological information. The analysis is similar for Komi which on top of sharing morphology with its relatives also share the writing system which provides it an extra gain. However, this might word in the opposite direction as well, as we can see with Faroese, Breton and Sanskrit. Faroese (fo) is morphologically rich and that should help, however its North-Germanic relatives are morphologically much simpler. Thus the model does not learn to rely on morphological attributes nor on word endings for the character model as much. The same is true for Sanskrit (sa), which is morphologically richer than its modern Indic relatives, with an extra layer of specific writing systems. Eventually, Breton model (br) is refined over Irish data only and while Irish is a typological outlier amongst Indo-European languages because of its Verb-Subject-Object word order, Breton has the standard Subject-Verb-Object, thus using Irish data might actually be detrimental.

These arguments show the respective importance of the writing system, the genre of the data, the morphological analysis and the typology in phylogenetic zero-shot dependency parsing. Those factors can either work together positively (Komi) or negatively (Cantonese) or cancel each other out (Amharic, Faroese).

4 Related Work

The goal of multi-task learning is to learn related tasks (either sharing their input and/or output space of participating of the same pipeline) jointly in order to improve their models over independently learned one (Caruana, 1997). In Søgaard et al. (2016), task hierarchy is directly encoded in the neural model allowing tasks with different output space to share parts of their parameters (POS tagging comes at a lower level than CCG parsing and only back-propagates to lower layers). Likewise, in Johnson et al. (2017), the encoder/decoder architecture allows to learn encoders that target several output languages and decoders than handle data from various input languages. However, in multi-task learning literature, task relationships are often fixed. In Cavallanti et al. (2010) tasks with the same output spaces share parameter updates through a fixed similarity graph. In this work, changing level in the tree can be seen as splitting the similarity graph into disjoint sub graphs. It is a way to have tasks relationships evolving during training and to encode information about task evolution that lacks in other multi-task methods.

In multi-lingual parsing, Ammar et al. (2016) propose to train a single model to parse many languages using both typological information, cross-lingual word representations and language specific information. While their model gives good results, they only apply it to 7 Germanic and Romance languages. It would be worth doing the experiment with 50+ languages and see how the results would change. However, because of language specific information their model would probably become very big. In this work, language specific information is not added on the top of the model, but is just language generic information that refines over time.

Che et al. (2017; 2018) and Stymne et al. (2018) propose to train parsers on several concatenated treebanks either from the same language or from related languages and to fine-tune the parsers on individual treebanks afterward to fit specific languages/domains. The main difference with our method, is that instead of one step of fine-tuning, we perform as many fine-tuning as there are ancestors in the tree, each time targeting more and more specific data. This in turn requires that we handle data imbalance therefore using sampling rather than plain concatenation.

Aufrant et al. (2016) propose to tackle zero-shot parsing by rewriting source treebanks to better fit target language typology. Assuming that typology is homogeneous in a language family, the phylogeny should drive models to be typologically aware. However, as we have seen for Breton and Irish, that assumption might not always hold.

Eventually, the closest work from our in spirit is the one of Berg-Kirkpatrick et al. (2010). They use a phylogenetic tree to guide the training of unsupervised dependency parsing models of several languages, using ancestor models to tie descendent ones. The main difference here beside supervision, is that we do not use ancestor models as biases but rather as initialization of descendent models.

5 Conclusion

We have presented a multi-task learning framework that allows one to train models for several tasks that have diverged over time. Leveraging their common evolutionary history through a phylogenetic tree, models share parameters and training samples until they need to diverge. As a by product of this phylogenetic training, we are provided with intermediary models that can be used to zero-shot a new related task, given its position in the evolutionary history.

We have applied this framework to dependency parsing using a graph-based neural parser and the phylogenetic tree of the languages from UD 2.2 to guide the training process. Our results show that phylogenetic training is beneficial for well populated families such as Indo-European and Uralic. It also helps generalization and prevents over-fitting when very few data are available. For zero-shot parsing, genre, writing system and morphology are crucial factors for the quality of parse predictions.

Some works have been done on automatically learning task relationship in multi-task setting. It would be interesting to see how the algorithm could figure out when and how to cluster languages automatically as phylogenetic trees do not directly depict grammar evolution.

Our model does not know that Latin came before Old French and before modern French, or that despite being Germanic, English underwent a heavy Romance influence. It would be worth investigating softening the tree constraints and instigating more evolutionary information in the structure.

Another important point is that we use gold part-of-speech and morphological information which is unlikely to be available in real scenarios. However, our new training procedure can be applied to any task, so a future work would be to use it to perform phylogenetic POS tagging.

Other directions for the future are designing better sampling methods as well as better ways to measure training convergence at each level.

Acknowledgement

This work was supported by ANR Grant GRASP No. ANR-16-CE33-0011-01 and Grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. We also thank the reviewers for their valuable feedback.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge. In *COLING 2016, the 26th International Conference on Computational Linguistics*, pages 119–130, Osaka, Japan. The COLING 2016 Organizing Committee.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1288–1297.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. 2010. Linear algorithms for online multitask classification. *J. Mach. Learn. Res.*, 11:2901–2934.

Wanxiang Che, Jiang Guo, Yuxuan Wang, Bo Zheng, Huaipeng Zhao, Yang Liu, Dechuan Teng, and Ting Liu. 2017. The hit-scir system for end-to-end parsing of universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 52–62, Vancouver, Canada. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64. Association for Computational Linguistics.

Xinying Chen and Kim Gerdes. 2017. Classifying languages by dependency structure. typologies of delexicalized universal dependency treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017), September 18-20, 2017, Università di Pisa, Italy*, 139, pages 54–63. Linköping University Electronic Press, Linköpings universitet.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Gerhard Jäger. 2015. Support for linguistic macro-families from weighted sequence alignment. *Proceedings of the National Academy of Sciences*, 112(41):12752–12757.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roşca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Natalie Schluter and Željko Agić. 2017. Empirically sampling universal dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 117–122.

Gary F. Simons and Charles D. Fennig, editors. 2018. *Ethnologue: Languages of the World, Twenty-first edition.* SIL International, Dallas, TX, USA.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia. Association for Computational Linguistics.

A Appendix

	Phylogenetic		Independent	
	UAS	LAS	UAS	LAS
Afro-Asiatic	80.73	74.99	**80.90**	**75.60**
Indo-European	**80.41**	**73.73**	78.93	72.45
Germanic	80.64	74.58	**80.82**	**75.16**
Slavic	**80.83**	**74.37**	78.21	72.09
Romance	**82.03**	**75.40**	81.67	75.18
Indo-Iranian	**80.03**	**72.29**	76.97	68.79
Greek	**80.13**	**75.59**	79.28	75.01
Turkic	**65.60**	**51.43**	61.82	47.07
Uralic	**78.25**	**72.40**	76.76	70.93
Dravidian	**81.97**	**70.59**	81.60	69.99
Avg	80.05	73.35	79.47	73.02
Avg No Dev	70.97	60.69	63.93	53.05

Table 3: Parsing results for phylogenetic and independent neural models averaged by language family. Families are sorted in the same order as they appear in Table 1. Indo-European averages include Armenian (hy) and Irish (ga). Global averages are repeated for completeness. Best results are reported in bold.

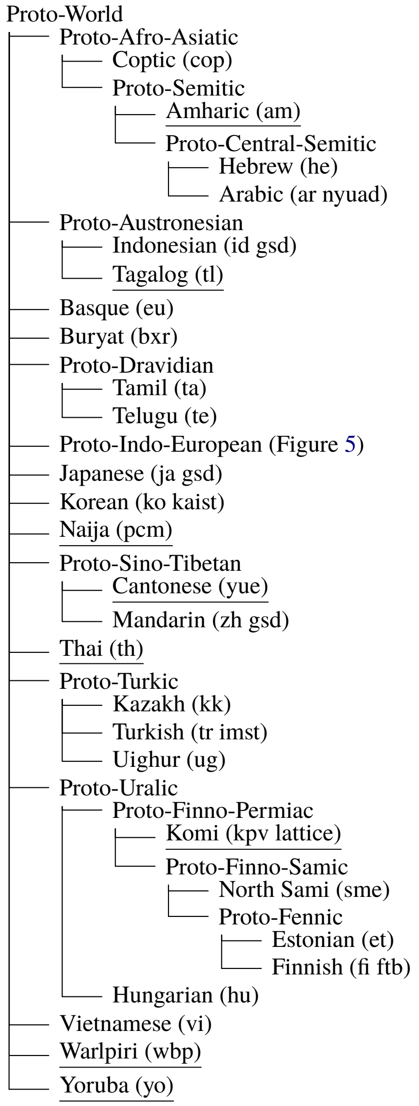


Figure 4: Phylogenetic tree used to guide the training process of the multi-lingual parser. Underlined languages are those that do not have a training set. The code of the language and if necessary the name of the treebank are given in parentheses. The Indo-European sub-tree is depicted on the right.

Figures 4 and 5 represent the phylogenetic tree used for guiding the training process. As we only use data from the UD project 2.2, we collapse unique child so that Vietnamese is not an Austro-Asiatic language, it is just Vietnamese. We also only use well attested families, thus Buryat, a Mongolic language, is alone and not linked to Turkic languages. Maybe, the most disputable choice is to put Naija in its own Creole family instead of the Germanic family.

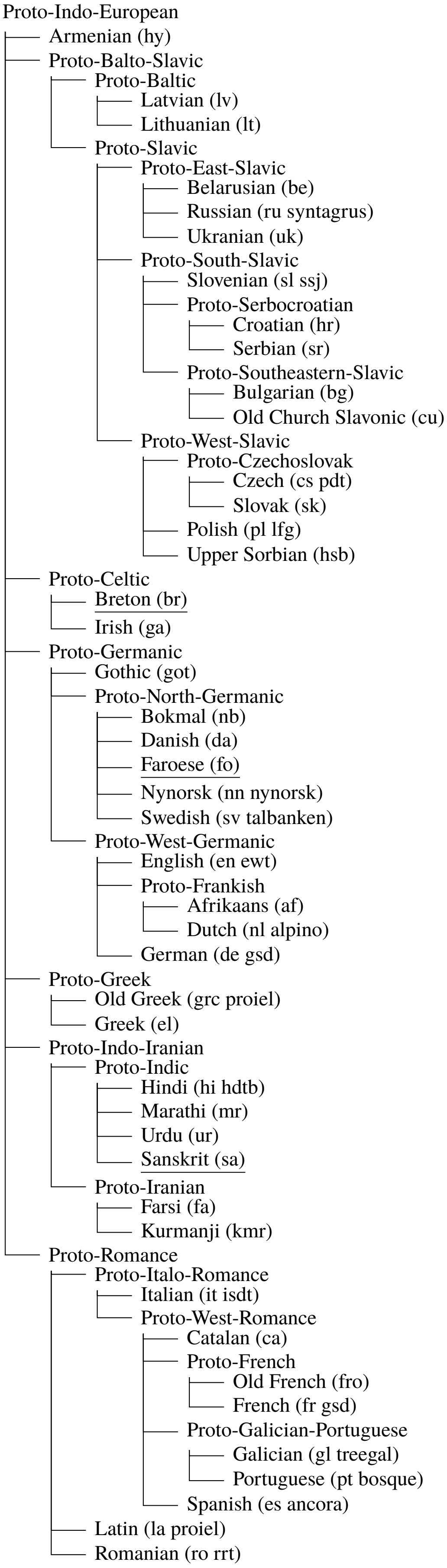


Figure 5: The Indo-European phylogenetic tree.

Discontinuous Constituency Parsing with a Stack-Free Transition System and a Dynamic Oracle

Maximin Coavoux[*]
Naver Labs Europe
`maximin.coavoux@naverlabs.com`

Shay B. Cohen
ILCC, School of Informatics
University of Edinburgh
`scohen@inf.ed.ac.uk`

Abstract

We introduce a novel transition system for discontinuous constituency parsing. Instead of storing subtrees in a stack –i.e. a data structure with linear-time *sequential access*– the proposed system uses a *set* of parsing items, with constant-time *random access*. This change makes it possible to construct any discontinuous constituency tree in exactly $4n-2$ transitions for a sentence of length n, whereas existing systems need a quadratic number of transitions to derive some structures. At each parsing step, the parser considers every item in the set to be combined with a *focus* item and to construct a new constituent in a bottom-up fashion. The parsing strategy is based on the assumption that most syntactic structures can be parsed incrementally and that the set –the memory of the parser– remains reasonably small on average. Moreover, we introduce a dynamic oracle for the new transition system, and present the first experiments in discontinuous constituency parsing using a dynamic oracle. Our parser obtains state-of-the-art results on three English and German discontinuous treebanks.

1 Introduction

Discontinuous constituency trees extend standard constituency trees by allowing crossing branches to represent long distance dependencies, such as the *wh*-extraction in Figure 1. Discontinuous constituency trees can be seen as derivations of Linear Context-Free Rewriting Systems (LCFRS, Vijay-Shanker et al., 1987), a class of formal grammars more expressive than context-free grammars, which makes them much harder to parse. In particular, exact CKY-style LCFRS parsing has an $\mathcal{O}(n^{3f})$ time complexity where f is the fan-out of the grammar (Kallmeyer, 2010).

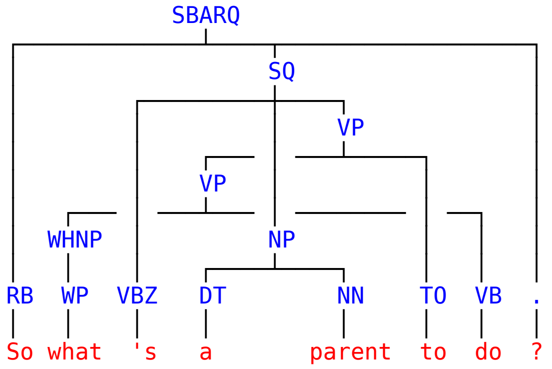


Figure 1: Discontinuous constituency tree from the Discontinuous Penn treebank.

A natural alternative to grammar-based chart parsing is transition-based parsing, that usually relies on fast approximate decoding methods such as greedy search or beam search. Transition-based discontinuous parsers construct discontinuous constituents by reordering terminals with the SWAP action (Versley, 2014a,b; Maier, 2015; Maier and Lichte, 2016; Stanojević and Garrido Alhama, 2017), or by using a split stack and the GAP action to combine two non-adjacent constituents (Coavoux and Crabbé, 2017a; Coavoux et al., 2019). These proposals represent the memory of the parser (i.e. the tree fragments being constructed) with data structures with **linear-time sequential access** (either a stack, or a stack coupled with a double-ended queue). As a result, these systems need to perform at least n actions to construct a new constituent from two subtrees separated by n intervening subtrees. Our proposal aims at avoiding this cost when constructing discontinuous constituents.

We design a novel transition system in which a discontinuous constituent is constructed in a single step, without the use of reordering actions such as SWAP. The main innovation is that the memory of the parser is not represented by a stack,

---

[*] Work done at the University of Edinburgh.

Proceedings of NAACL-HLT 2019, pages 204–217
Minneapolis, Minnesota, June 2 - June 7, 2019.

Initial configuration	$(\emptyset,\ \texttt{null},\ 0,\ \emptyset) : 0$		
Goal configuration	$(\emptyset,\ \{0, 1, \ldots, n-1\},\ n,\ C) : 4n-2$		
Structural actions	Input	Output	Precondition
SHIFT	$(S,\ s_f,\ i,\ C) : j$	$\Rightarrow (S \cup \{s_f\},\ \{i\},\ i+1,\ C) : j+1$	$i < n$, j is even
COMBINE-s	$(S,\ s_f,\ i,\ C) : j$	$\Rightarrow (S - s,\ s_f \cup s,\ i,\ C) : j+1$	$s \in S$, j is even
Labelling actions			
LABEL-X	$(S,\ s_f,\ i,\ C) : j$	$\Rightarrow (S,\ s_f,\ i,\ C \cup \{(X,\ s_f)\}) : j+1$	j is odd
NO-LABEL	$(S,\ s_f,\ i,\ C) : j$	$\Rightarrow (S,\ s_f,\ i,\ C) : j+1$	$i \neq n$ or $S \neq \emptyset$, j is odd

Table 1: Set-based transition system description. Variable j is the number of steps performed since the start of the derivation.

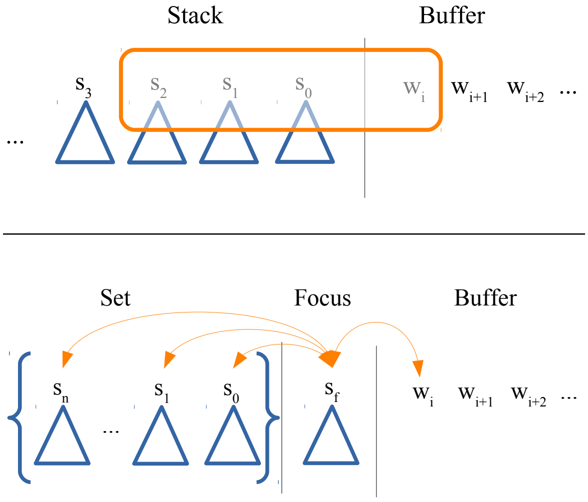


Figure 2: In a stack-based system like shift-reduce-swap (upper part), the parser extracts features from a *local* region of the configuration (orange part), to predict the next action such as: construct a new tree with label X and children s_0 and s_1 (REDUCE-X). In our proposed set-based system (lower part), we consider every item in the set to be combined bottom-up with the focus item s_f and score independently each possible transition.

as is usual in shift-reduce systems, but by an unordered **random-access** set. The parser considers every constituent in the current memory to construct a new constituent in a bottom-up fashion, and thus instantly models interactions between parsing items that are not adjacent. As such, we describe a left-to-right parsing model that deviates from the standard stack-buffer setting, a legacy from pushdown automata and classical parsing algorithms for context-free grammars.

Our contributions are summarized as follows:

- We design a novel transition system for discontinuous constituency parsing, based on a memory represented by a *set* of items, and that derives any tree in exactly $4n - 2$ steps for a sentence of length n;
- we introduce the first dynamic oracle for discontinuous constituency parsing;
- we present an empirical evaluation of the transition system and dynamic oracle on two German and one English discontinuous treebanks.

The code of our parser is released as an open-source project at `https://gitlab.com/mcoavoux/discoparset`.

2 Set-based Transition System

System overview We propose to represent the memory of the parser by (i) a set of parsing items and (ii) a single *focus item*. Figure 2 (lower part) illustrates a configuration in our system. The parser constructs a tree with two main actions: shift the next token to make it the new focus item (SHIFT), or combine any item in the set with the focus item to make a new constituent bottom-up (COMBINE action).

Since the memory is not an ordered data structure, the parser considers equally every pending parsing item, and thus constructs a discontinuous constituent in a single step, thereby making it able to construct any discontinuous tree in $\mathcal{O}(n)$ transitions.

The use of an unordered random-access data structure to represent the memory of the parser also leads to a major change for the scoring system (Figure 2). Stack-based systems use a *local view* of a parsing configuration to extract features and score actions: features only rely on the few topmost elements on the stack and buffer. The score of each transition depends on the totality of this local view. In constrast, we consider equally every item in the set, and therefore rely on a *global*

Even action	Set (S)	Focus (s_f)	Buffer	Odd action
	$\{\}$	none	So what 's a parent to do ?	
⇒SH⇒	$\{\}$	$\{\text{So}\}$	what 's a parent to do ?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So}\}_0\}$	$\{\text{what}\}$	's a parent to do ?	⇒LABEL-WHNP
⇒SH⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1\}$	$\{\text{'s}\}$	a parent to do ?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1, \{\text{'s}\}_2\}$	$\{\text{a}\}$	parent to do ?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1, \{\text{'s}\}_2, \{\text{a}\}_3\}$	$\{\text{parent}\}$	to do ?	⇒NO-LABEL
⇒COMB-3⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1, \{\text{'s}\}_2\}$	$\{\text{a parent}\}$	to do ?	⇒LABEL-NP
⇒COMB-2⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1\}$	$\{\text{'s a parent}\}$	to do ?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1, \{\text{'s a parent}\}_2\}$	$\{\text{to}\}$	do ?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So}\}_0, \{\text{what}\}_1, \{\text{'s a parent}\}_2, \{\text{to}\}_5\}$	$\{\text{do}\}$	?	⇒NO-LABEL
⇒COMB-1⇒	$\{\{\text{So}\}_0, \{\text{'s a parent}\}_2, \{\text{to}\}_5\}$	$\{\text{what do}\}$	?	⇒LABEL-VP
⇒COMB-5⇒	$\{\{\text{So}\}_0, \{\text{'s a parent}\}_2\}$	$\{\text{what to do}\}$	?	⇒LABEL-VP
⇒COMB-2⇒	$\{\{\text{So}\}_0\}$	$\{\text{what 's a parent to do}\}$	?	⇒LABEL-SQ
⇒COMB-0⇒	$\{\}$	$\{\text{so what 's a parent to do}\}$	?	⇒NO-LABEL
⇒SH⇒	$\{\{\text{So what 's a parent to do}\}_0\}$	$\{?\}$		⇒NO-LABEL
⇒COMB-0⇒	$\{\}$	$\{\text{So what 's a parent to do ?}\}$		⇒LABEL-SBARQ

Table 2: Full derivation for the sentence in Figure 1. As a convention, we index elements in the set with their left-index and use COMB-i to denote COMB-s_i. We also use tokens instead of their indexes for better legibility.

view of the memory (Section 3). However, we score each possible combinations independently: the score of a single combination only depends on the two constituents that are combined, regardless of the rest of the set.

2.1 System Description

Definitions We first define an *instantiated (discontinuous) constituent* (X, s) as a nonterminal label X associated with a set of token indexes s. We call $\min(s)$ the *left-index* of c and $\max(s)$ its *right-index*. For example in Figure 1, the two VPs are respectively represented by (VP, $\{1, 6\}$) and (VP, $\{1, 5, 6\}$), and they have the same right index (6) and left index (1).

A *parsing configuration* is a quadruple (S, s_f, i, C) where:

- S is a set of sets of indexes and represents the memory of the parser;
- s_f is a set of indexes called the *focus item*, and satisfies $\max(s_f) = i - 1$;
- i is the index of the next token in the buffer;
- C is a set of instantiated constituents.

Each new constituent is constructed bottom-up from the *focus* item and another item in the set S.

Transition set Our proposed transition system is based on the following types of actions:

- SHIFT constructs a singleton containing the next token in the buffer and assigns it as the new focus item. The former focus item is added to S.
- COMBINE-s computes the union between the focus item s_f and another item s from the set S, to form the new focus item $s \cup s_f$.
- LABEL-X instantiates a new constituent (X, s_f) whose yield is the set of indexes in the focus item s_f.
- NO-LABEL has no effect; its semantics is that the current focus set is not a constituent.

Following Cross and Huang (2016b), transitions are divided into **structural** actions (SHIFT, COMBINE-s) and **labelling** actions (LABEL-X, NO-LABEL). The parser may only perform a structural action on an even step and a labelling action on an odd step. For our system, this distinction has the crucial advantage of keeping the number of possible actions low at each parsing step, compared to a system that would perform a COMBINE action and a labelling action in a single REDUCE-s-X action.[1]

Table 1 presents each action as a deduction rule associated with preconditions. In Table 2, we describe how to derive the tree from Figure 1.

2.2 Oracles

Training a transition-based parser requires an oracle, i.e. a function that determines what the best action is in a specific parsing configuration to serve as a training signal. We first describe a static oracle that provides a canonical derivation for a given gold tree. We then introduce a dynamic oracle that determines what the best action is in any parsing configuration.

---

[1] In such a case, we would need to score $|S| \times |N| + 1$ actions, where N is the set of nonterminals, instead of $|S| + 1$ actions for our system.

2.2.1 Static Oracle

Our transition system exhibits a fair amount of *spurious ambiguity*, the ambiguity exhibited by the existence of many possible derivations for a single tree. Indeed, since we use an unordered memory, an n-ary constituent (and more generally a tree) can be constructed by many different transition sequences. For example, the set $\{0, 1, 2\}$ might be constructed by combining

- $\{0\}$ and $\{1\}$ first, and the result with $\{2\}$; or
- $\{1\}$ and $\{2\}$ first, and the result with $\{0\}$; or
- $\{0\}$ and $\{2\}$ first, and the result with $\{1\}$.

Following Cohen et al. (2012), we eliminate spurious ambiguity by selecting a canonical derivation for a gold tree. In particular, we design the static oracle (i) to apply COMBINE as soon as possible in order to minimize the size of the memory (ii) to combine preferably with the most recent set in the memory when several combinations are possible. The first choice is motivated by properties of our system: when the memory is smaller, there are fewer choices, therefore decisions are simpler and less expensive to score.

2.2.2 Dynamic Oracle

Parsers are usually trained to predict the gold sequence of actions, using a static oracle. The limitation of this method is that the parser only sees a tiny portion of the search space at train time and only trains on gold input (i.e. configurations obtained after performing gold actions). At test time, it is in a different situation due to error propagation: it must predict what the best actions are in configurations from which the gold tree is probably no longer reachable.

To alleviate this limitation, Goldberg and Nivre (2012) proposed to train a parser with a *dynamic oracle*, an oracle that is defined for any parsing configuration and outputs the set of best actions to perform. In contrast, a *static oracle* is deterministic and is only defined for gold configurations.

Dynamic oracles were proposed for a wide range of dependency parsing transition systems (Goldberg and Nivre, 2013; Gómez-Rodríguez et al., 2014; Gómez-Rodríguez and Fernández-González, 2015), and later adapted to constituency parsing (Coavoux and Crabbé, 2016; Cross and Huang, 2016b; Fernández-González and Gómez-Rodríguez, 2018b,a).

In the remainder of this section, we introduce a dynamic oracle for our proposed transition system. It can be seen as an extension of the oracle of Cross and Huang (2016b) to the case of discontinuous parsing.

Preliminary definitions For a parsing configuration c, the relation $c \vdash c'$ holds iff c' can be derived from c by a single transition. We note $\vdash^*$ the reflexive and transitive closure of $\vdash$. An instantiated constituent (X, s) is **reachable** from a configuration $c = (S, s_f, i, C)$ iff there exists $c' = (S', s'_f, i', C')$ such that $(X, s) \in C'$ and $c \vdash^* c'$. Similarly, a set of constituents t (possibly a full discontinuous constituency tree) is reachable iff there exists a configuration $c' = (S', s'_f, i', C')$ such that $t \subseteq C'$ and $c \vdash^* c'$. We note $\mathsf{reach}(c, t^*)$ the set of constituents that are (i) in the gold set of constituents t^* (ii) reachable from c.

We define a total order $\preceq$ on index sets:

$$s \preceq s' \Leftrightarrow \begin{cases} \max(s) < \max(s'), \\ \textbf{or} \\ \max(s) = \max(s') \\ \text{and } s \subseteq s'. \end{cases}$$

This order naturally extends to the constituents of a tree: $(X, s) \preceq (X', s')$ iff $s \preceq s'$. If (X, s) precedes (X', s'), then (X, s) must be constructed before (X', s'). Indeed, since the right-index of the focus item is non-decreasing during a derivation (as per the transition definitions), constituents are constructed in the order of their right-index (first condition). Moreover, since the algorithm is bottom-up, a constituent must be constructed before its parent (second condition).

From a configuration $c = (S, s_f, i, C)$ at an odd step, a constituent $(X, s_g) \notin C$ is reachable iff both the following properties hold:

1. $\max(s_f) \leq \max(s_g)$;
2. $\forall s \in S \cup \{s_f\}, (s \subseteq s_g)$ or $(s \cap s_g = \emptyset)$.

Condition 1 is necessary because the parser can only construct new constituents (X, s) such that $s_f \preceq s$. Condition 2 makes sure that s_g can be constructed from a union of elements from $S \cup \{s_f\}$, potentially augmented with terminals from the bufffer: $\{i, i+1, \dots, \max(s_g)\}$.

Following Cross and Huang (2016b), we define $\mathsf{next}(c, t^*)$ as the smallest reachable gold constituent from a configuration c. Formally:

$$\mathsf{next}(c, t^*) = \operatorname*{argmin}_{\preceq} \mathsf{reach}(c, t^*).$$

Oracle algorithm We first define the oracle o for the odd step of a configuration $c = (S, s_f, i, C)$:

$$\mathsf{o}_{\mathsf{odd}}(c, t^*) = \begin{cases} \{\text{LABEL-X}\} & \text{if } \exists(X, s_f) \in t^*, \\ \{\text{NO-LABEL}\} & \text{otherwise.} \end{cases}$$

For even steps, assuming $\mathsf{next}(c, t^*) = (X, s_g)$, we define the oracle as follows:

$$\mathsf{o}_{\mathsf{even}}(c, t^*) = \begin{cases} \{\text{COMB-}s | (s_f \cup s) \subseteq s_g\} \\ \qquad \text{if } \max(s_g) = \max(s_f), \\ \{\text{COMB-}s | (s_f \cup s) \subseteq s_g\} \cup \{\text{SH}\} \\ \qquad \text{if } \max(s_g) > \max(s_f). \end{cases}$$

We provide a proof of the correctness of the oracle in Appendix A.

3 A Neural Network based on Constituent Boundaries

We first present an encoder that computes context-aware representations of tokens (Section 3.1). We then discuss how to compute the representation of a set of tokens (Section 3.2). We describe the action scorer (Section 3.3), the POS tagging component (Section 3.4), and the objective function (Section 3.5).

3.1 Token Representations

As in recent proposals in dependency and constituency parsing (Cross and Huang, 2016a; Kiperwasser and Goldberg, 2016), our scoring system is based on a sentence transducer that constructs a context-aware representation for each token.

Given a sequence of tokens $x_1^n = (x_1, \dots, x_n)$, we first run a single-layer character bi-LSTM encoder $\mathbf{c}$ to obtain a character-aware embedding $\mathbf{c}(x_i)$ for each token. We represent a token x_i as the concatenation of a standard word embedding $\mathbf{e}(x_i)$ and the character-aware embedding: $\mathbf{w}_{x_i} = [\mathbf{c}(x_i); \mathbf{e}(x_i)]$.

Then, we run a 2-layer bi-LSTM transducer over the sequence of token representations:

$$(\mathbf{h}_1^{(1)}, \dots, \mathbf{h}_n^{(1)}) = \text{bi-LSTM}(\mathbf{w}_{x_1}, \dots, \mathbf{w}_{x_n}),$$
$$(\mathbf{h}_1^{(2)}, \dots, \mathbf{h}_n^{(2)}) = \text{bi-LSTM}(\mathbf{h}_1^{(1)}, \dots, \mathbf{h}_n^{(1)}).$$

The parser uses the context-aware token representations $\mathbf{h}_i^{(2)}$ to construct vector representations of sets or constituents.

3.2 Set Representations

An open issue in neural discontinuous parsing is the representation of discontinuous constituents. In projective constituency parsing, it has become standard to use the boundaries of constituents (Hall et al., 2014; Crabbé, 2015; Durrett and Klein, 2015), an approach that proved very successful with bi-LSTM token representations (Cross and Huang, 2016b; Stern et al., 2017).

Although constituent boundary features improves discontinuous parsing (Coavoux and Crabbé, 2017a), relying only on the left-index and the right-index of a constituent has the limitation of ignoring gaps inside a constituent. For example, since the two VPs in Figure 1 have the same right-index and left-index, they would have the same representations. It may also happen that constituents with identical right-index and left-index do not have the same labels.

We represent a (possibly partial) constituent with the yield s, by computing 4 indexes from s: $(\min(s), \max(s), \min(\overline{s}), \max(\overline{s}))$. The set $\overline{s}$ represents the gap in s, i.e. the tokens between $\min(s)$ and $\max(s)$ that are not in the yield of s:

$$\overline{s} = \{i | \min(s) < i < \max(s) \text{ and } i \notin s\}.$$

Finally, we extract the corresponding token representations of the 4 indexes and concatenate them to form the vector representation $\mathbf{r}(s)$ of s:

$$\mathbf{r}(s) = [\mathbf{h}^{(2)}_{\min(s)}; \mathbf{h}^{(2)}_{\max(s)}; \mathbf{h}^{(2)}_{\min(\overline{s})}; \mathbf{h}^{(2)}_{\max(\overline{s})}].$$

For an index set that does not contain a gap, we have $\overline{s} = \emptyset$. To handle this case, we use a parameter vector $\mathbf{h}_{\mathsf{nil}}$, randomly initialized and learned jointly with the network, to embed $\max(\emptyset) = \min(\emptyset) = \mathsf{nil}$.

For example, the constituents (VP, $\{1, 6\}$) and (VP, $\{1, 5, 6\}$) will be respectively vectorized as:

$$\mathbf{r}(\{1,6\}) = [\mathbf{h}_1^{(2)}; \mathbf{h}_6^{(2)}; \mathbf{h}_2^{(2)}; \mathbf{h}_5^{(2)}],$$
$$\mathbf{r}(\{1,5,6\}) = [\mathbf{h}_1^{(2)}; \mathbf{h}_6^{(2)}; \mathbf{h}_2^{(2)}; \mathbf{h}_4^{(2)}].$$

This representation method makes sure that two distinct index sets have distinct representations, as long as they have at most one gap each. This property no longer holds if one index sets has more than one gap.

3.3 Action Scorer

For each type of action –structural or labelling– we use a feedforward network with two hidden layers.

Structural actions At structural steps, for a configuration $c = (S, s_f, i, C)$, we need to compute the score of $|S|$ COMBINE actions and possibly a SHIFT action. In our approach, the score of a combine-s action only depends on s and s_f and is independent of the rest of the configuration (i.e. other items in the set). We first construct input matrix M as follows:

$$M = \begin{pmatrix} \mathbf{r}(s_1) & \cdots & \mathbf{r}(s_n) & \mathbf{r}(\{i\}) \\ \mathbf{r}(s_f) & \cdots & \mathbf{r}(s_f) & \mathbf{r}(s_f) \end{pmatrix}.$$

Each of the first n columns of matrix M represents the input for a COMBINE action, whereas the last column is the input for the SHIFT action. We then compute the score of each structural action:

$$P(\cdot|c) = \text{Softmax}(\text{FF}_s(M)),$$

where FF_s is a feedforward network with two hidden layers, a $\tanh$ activation and a single output unit. In other words, it outputs a single scalar for each column vector of matrix M. This part of the network can be seen as an attention mechanism, where the focus item is the query, and the context is formed by the items in the set and the first element in the buffer.

Labelling actions We compute the probabilities of labelling actions as follows:

$$P(\cdot|s_f) = \text{Softmax}(\text{FF}_l(\mathbf{r}(s_f))),$$

where FF_l is a feedforward network with two hidden layers activated with the $\tanh$ function, and $|N| + 1$ output units, where N is the set of nonterminals.

3.4 POS Tagger

Following Coavoux and Crabbé (2017b), we use the first layer of the bi-LSTM transducer as input to a Part-of-Speech (POS) tagger that is learned jointly with the parser. For a sentence x_1^n, we compute the probability of a sequence of POS tags $t_1^n = (t_1, \ldots, t_n)$ as follows:

$$P(t_1^n|x_1^n) = \prod_{i=1}^{n} \text{Softmax}(\mathbf{W}^{(t)} \cdot \mathbf{h}_i^{(1)} + \mathbf{b}^{(t)})_{t_i},$$

where $\mathbf{W}^{(t)}$ and $\mathbf{b}^{(t)}$ are parameters.

3.5 Objective Function

In the static oracle setting, for a single sentence x_1^n, we optimize the sum of the log-likelihood of gold POS-tags t_1^n and the log-likelihood of gold parsing actions a_1^n:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_t + \mathcal{L}_p, \\ \mathcal{L}_t &= -\sum_{i=1}^{n} \log P(t_i|x_1^n), \\ \mathcal{L}_p &= -\sum_{i=1}^{4n-2} \log P(a_i|a_1^{i-1}, x_1^n). \end{aligned}$$

We optimize this objective by alternating a stochastic step for the tagging objective and a stochastic step for the parsing objective, as is standard in multitask learning (Caruana, 1997).

In the dynamic oracle setting, instead of optimizing the likelihood of the gold actions (assuming all previous actions were gold), we optimize the likelihood of the best actions, as computed by the dynamic oracle, from a configuration sampled from the space of all possible configurations. In practice, before each epoch, we sample each sentence from the training corpus with probability p and we use the current (non-averaged) parameters to parse the sentence and generate a sequence of configurations. Instead of selecting the highest-scoring action at each parsing step, as in a normal inference step, we sample an action using the softmax distribution computed by the parser, as done by Ballesteros et al. (2016). Then, we use the dynamic oracle to calculate the best action from each of these configurations. In case there are several best actions, we deterministically choose a single action by favoring a COMBINE over a SHIFT (to bias the model towards a small memory), and to COMBINE with the item with the highest right-index (to avoid spurious discontinuity in partial constituents). We train the parser on these sequences of potentially non-gold configuration-action pairs.

4 Experiments

We carried out experiments to assess the adequacy of our system and the effect of training with the dynamic oracle. We present the three discontinuous constituency treebanks that we used (Section 4.1), our experimental protocol (Section 4.2), then we discuss the results (Section 4.3) and the efficiency of the parser (Section 4.4).

	DPTB			Tiger			Negra		
	F1	Disc. F1	POS	F1	Disc. F1	POS	F1	Disc. F1	POS
static	91.1	68.2	97.2	87.4	61.7	98.3	83.6	51.3	97.9
dynamic	91.4	70.9	97.2	87.6	62.5	98.4	84.0	54.0	98.0

Table 3: Results on development corpora. F1 is the Fscore on all constituents, Disc. F1 is an Fscore computed only on discontinuous constituents, POS is the accuracy on part-of-speech tags. Detailed results (including precision and recall) are given in Table 7 of Appendix C.

	English (DPTB)		German (Tiger)		German (Negra)	
Model	F	Disc. F	F	Disc. F	F	Disc. F
Predicted POS tags or own tagging						
This work, dynamic oracle	**90.9**	67.3	82.5	**55.9**	**83.2**	**56.3**
Coavoux et al. (2019),* GAP, bi-LSTM	**91.0**	**71.3**	**82.7**	**55.9**	**83.2**	54.6
Stanojević and Garrido Alhama (2017),* SWAP, stack/tree-LSTM			77.0			
Coavoux and Crabbé (2017a), SR-GAP, perceptron			79.3			
Versley (2016), pseudo-projective, chart-based			79.5			
Corro et al. (2017),* bi-LSTM, Maximum Spanning Arborescence	89.2					
van Cranenburgh et al. (2016), DOP, ≤ 40	87.0				74.8	
Fernández-González and Martins (2015), dependency-based			77.3			
Gebhardt (2018), LCFRS with latent annotations			75.1			
Gold POS tags						
Stanojević and Garrido Alhama (2017),* SWAP, stack/tree-LSTM			81.6		82.9	
Coavoux and Crabbé (2017a), SR-GAP, perceptron			81.6	49.2	82.2	50.0
Maier (2015), SWAP, perceptron			74.7	18.8	77.0	19.8
Corro et al. (2017),* bi-LSTM, Maximum Spanning Arborescence	90.1		81.6			
Evang and Kallmeyer (2011), PLCFRS, < 25	$79^{\dagger}$					

Table 4: Discontinuous parsing results on the test sets. *Neural scoring system. †Does not discount root symbols and punctuation.

4.1 Datasets

We perform experiments on three discontinuous constituency corpora. The discontinuous Penn Treebank was introduced by Evang and Kallmeyer (2011) who converted the long distance dependencies encoded by indexed traces in the original Penn treebank (Marcus et al., 1993) to discontinuous constituents. We used the standard split (sections 2-21 for training, 22 for development and 23 for test). The Tiger corpus (Brants et al., 2004) and the Negra corpus (Skut et al., 1997) are both German treebanks natively annotated with discontinuous constituents. We used the SPMRL split for the Tiger corpus (Seddah et al., 2013), and the split of Dubey and Keller (2003) for the Negra corpus.

4.2 Implementation and Protocol

We implemented our parser in Python using the Pytorch library (Paszke et al., 2017). We trained each model with the ASGD algorithm (Polyak and Juditsky, 1992) for 100 epochs. Training a single model takes approximately a week with a GPU. We evaluate a model every 4 epochs on the validation set and select the best performing model according to the validation F-score. We refer the reader to Table 5 of Appendix B for the full list of hyperparameters.

We evaluate models with the dedicated module of `discodop`[2] (van Cranenburgh et al., 2016). We use the standard evaluation parameters (`proper.prm`), that ignore punctuations and root symbols. We report two evaluation metrics: a standard Fscore (F) and an Fscore computed only on discontinuous constituents (Disc. F), which provides a more qualitative evaluation of the ability of the parser to recover long distance dependencies.

---

[2] `https://github.com/andreasvc/disco-dop`

4.3 Results

Effect of Dynamic Oracle We present parsing results on the development sets of each corpus in Table 3. The effect of the oracle is in line with other published results in projective constituency parsing (Coavoux and Crabbé, 2016; Cross and Huang, 2016b) and dependency parsing (Goldberg and Nivre, 2012; Gómez-Rodríguez et al., 2014): the dynamic oracle improves the generalization capability of the parser.

External comparisons In Table 4, we compare our parser to other transition-based parsers (Maier, 2015; Coavoux and Crabbé, 2017a; Stanojević and Garrido Alhama, 2017; Coavoux et al., 2019), the pseudo-projective parser of Versley (2016), grammar-based chart parsers (Evang and Kallmeyer, 2011; van Cranenburgh et al., 2016; Gebhardt, 2018) and parsers based on dependency parsing (Fernández-González and Martins, 2015; Corro et al., 2017). Note that some of them only report results in a gold POS tag setting (the parser has access to gold POS tags and use them as features), a setting that is much easier than ours.

Our parser matches the state of the art of Coavoux et al. (2019). This promising result shows that it is feasible to design accurate transition systems without an ordered memory.

4.4 Efficiency

Our transition system derives a tree for a sentence of n words in exactly $4n-2$ transitions. Indeed, there must be n SHIFT actions, and $n-1$ COMBINE actions. Each of these $2n-1$ transitions must be followed by a single labelling action.

The statistical model responsible for choosing which action to perform at each parsing step needs to score $|S|+1$ actions for a structural step and $|N|+1$ actions for a labelling step (where N is the set of possible nonterminals). Since in the worst case, $|S|$ contains $n-1$ singletons, the parser has an $\mathcal{O}(n(|N|+n))$ time complexity.

In practice, the memory of the parser S remains relatively small on average. We report in Figure 3 the distribution of the size of S across configurations when parsing the development sets of three corpora. For the German treebanks, the memory contains 7 or fewer elements for more than 99 percents of configurations. For the Penn treebank, the memory is slighlty larger, with 98 percents of configuration with 11 or fewer items.

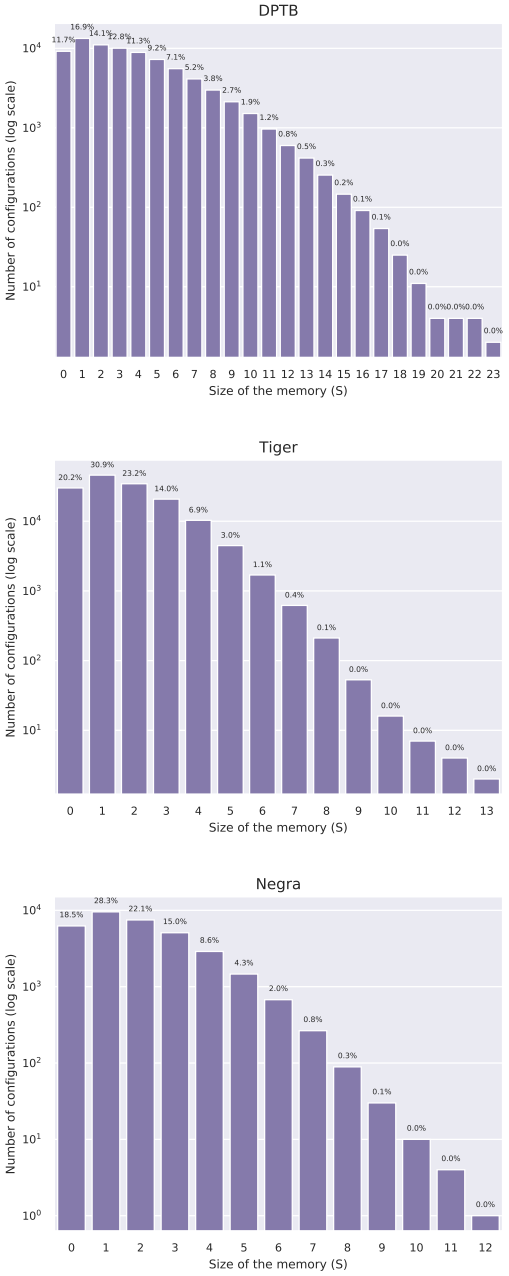


Figure 3: Distribution of the size of the memory of the parser S across configurations derived when parsing the development set of the three corpora. In practice, we observe that the memory remains small.

We report empirical runtimes in Table 6 of Appendix C. Our parser compares decently with other transition-based parsers, despite being written in Python.

5 Related Work

Existing transition systems for discontinuous constituency parsing rely on three main strategies for constructing discontinuous constituents: a swap-based strategy, a split-stack strategy, and the use of non-local transitions.

Swap-based systems Swap-based transition systems are based on the idea that any discontinuous constituency tree can be transformed into a projective tree by reordering terminals. They reorder terminals by swapping them with a dedicated action (SWAP), commonly used in dependency parsing (Nivre, 2009). The first proposals in transition-based discontinuous constituency parsing used the SWAP action on top of an easy-first parser (Versley, 2014a,b). Subsequent proposals relied on a shift-reduce system (Maier, 2015; Maier and Lichte, 2016) or a shift-promote-adjoin system (Stanojević and Garrido Alhama, 2017).

The main limitation of swap-based system is that they tend to require a large number of transitions to derive certain trees. The choice of an oracle that minimizes derivation lengths has a substantially positive effect on parsing (Maier and Lichte, 2016; Stanojević and Garrido Alhama, 2017).

Split-stack systems The second parsing strategy constructs discontinuous constituents by allowing the parser to reduce pairs of items that are not adjacent in the stack. In practice, Coavoux and Crabbé (2017a) split the usual stack of shift-reduce parsers into two data structures (a stack and a double-ended queue), in order to give the parser access to two focus items: the respective tops of the stack and the dequeue, that may or may not be adjacent. A dedicated action, GAP, pushes the top of the stack onto the bottom of the queue to make the next item in the stack available for a reduction.

The split stack associated with the GAP action can be interpreted as a linear-access memory: it is possible to access the i^{th} element in the stack, but it requires i operations.

Non-local transitions Non-local transitions generalize standard parsing actions to non-adjacent elements in the parsing configurations. Maier and Lichte (2016) introduced a non-local transition SKIPSHIFT-i which applies SHIFT to the i^{th} element in the buffer. Non-local transitions are also widely used in non-projective dependency parsing (Attardi, 2006; Qi and Manning, 2017; Fernández-González and Gómez-Rodríguez, 2018).

The key difference between these systems and ours is that we use an unordered memory. As a result, the semantics of the COMBINE-s action we introduce in Section 2 is independent from a specific position in the stack or the buffer. A system with an action such as SKIPSHIFT-i needs to learn parameters with every possible i, and will only learn parameters with the SKIPSHIFT-i actions that are required to derive the training set. In contrast, we use the same parameters to score each possible COMBINE-s action.

6 Conclusion

We have presented a novel transition system that dispenses with the use of a stack, i.e. a memory with linear sequential access. Instead, the memory of the parser is represented by an unordered data structure with random-access: a set. We have designed a dynamic oracle for the resulting system and shown their empirical potential with state-of-the-art results on discontinuous constituency parsing of one English and two German treebanks. Finally, we plan to adapt our system to non-projective dependency parsing and semantic graph parsing.

Acknowledgments

We thank Caio Corro, Giorgio Satta, Marco Damonte, as well as NAACL anonymous reviewers for feedback and suggestions. We gratefully acknowledge the support of Huawei Technologies.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 166–170. Association for Computational Linguistics.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010, Austin, Texas. Association for Computational Linguistics.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France. Springer.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017a. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270, Valencia, Spain. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017b. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 331–336, Valencia, Spain. Association for Computational Linguistics.

Maximin Coavoux, Benoît Crabbé, and Shay B. Cohen. 2019. Unlexicalized transition-based discontinuous constituency parsing. *CoRR*, abs/1902.08912v1.

Shay B. Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2012. Elimination of spurious ambiguity in transition-based dependency parsing. *CoRR*, abs/1206.6735.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1645–1655, Copenhagen, Denmark. Association for Computational Linguistics.

Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1856, Lisbon, Portugal. Association for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for german using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China. Association for Computational Linguistics.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Non-projective dependency parsing with non-local transitions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 693–700. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018a. Dynamic oracles for top-down and in-order shift-reduce constituent parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1303–1313, Brussels, Belgium. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018b. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. *CoRR*, abs/1804.07961.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3063, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. The COLING 2012 Organizing Committee.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–261, Beijing, China. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927, Doha, Qatar. Association for Computational Linguistics.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*, 1st edition. Springer Publishing Company, Incorporated.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics, Volume 19, Number 2, June 1993, Special Issue on Using Large Corpora: II*.

Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *CoRR*, abs/1511.06807.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Boris T. Polyak and Anatoli B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington, DC, USA. Association for Computational Linguistics.

Miloš Stanojević and Raquel Garrido Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1667–1677, Copenhagen, Denmark. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Yannick Versley. 2014a. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland. Dublin City University.

Yannick Versley. 2014b. Incorporating semi-supervised features into discontinuous easy-first constituent parsing. *CoRR*, abs/1409.3813.

Yannick Versley. 2016. Discontinuity re^2-visited: A minimalist approach to pseudoprojective constituent parsing. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 58–69, San Diego, California. Association for Computational Linguistics.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*.

A Oracle Correctness

The oracle o leads to the reachable tree with the highest F-score with respect to the gold tree. The F-score of a predicted tree $\hat{t}$ (represented as a set of instantiated constituents) with respect to a gold tree t^* is defined as:

$$\mathsf{precision}(\hat{t}, t^*) = p = \frac{|\hat{t} \cap t^*|}{|\hat{t}|},$$

$$\mathsf{recall}(\hat{t}, t^*) = r = \frac{|\hat{t} \cap t^*|}{|t^*|},$$

$$F_1(\hat{t}, t^*) = \frac{2pr}{p + r}.$$

By definition, $\mathsf{o}_{\mathsf{odd}}$ is optimal for precision because it constructs a constituent only if it is gold, and optimal for recall because it will construct a gold constituent if it is possible to do so.

Moreover, $\mathsf{o}_{\mathsf{even}}$ is optimal for recall because any gold constituent reachable from c will still be reachable after any transition in $\mathsf{o}_{\mathsf{even}}(c, t^*)$. Assuming a configuration $c = (S, s_f, i, C)$ and $\mathsf{next}(c, t^*) = s_g$, we consider separately the SHIFT case and the COMBINE-s case:

- SHIFT case ($\max(s_g) > \max(s_f)$): constituents (X, s) reachable from c and not reachable from SHIFT(c) satisfy $\max(s) = i$. If a gold constituent satisfies this property, we have $s \preceq s_g$, which contradicts the assumption that $s_g = \mathsf{next}(c, t^*)$ (see definition of oracle in Section 2.2.2).
- COMBINE-s case: Let (X, s') be a reachable gold constituent. Since it is compatible with s_g, there are three possible cases:
    - if (X, s') is an ascendant of s_g, then $(s \cup s_f) \subseteq s_g \subset s'$, therefore (X, s') is still reachable from COMBINE-$s(c)$.
    - if (X, s') is a descendant of s_g then $s' \preceq s_g$, which contradicts the definition of s_g.
    - if s' and s_g are completely disjoint, we have $s' \cap s = s' \cap s_f = \emptyset$, therefore $s' \cap (s \cup s_f) = \emptyset$, and s' is still reachable from COMBINE-$s(c)$.

Finally, since $\mathsf{o}_{\mathsf{even}}$ does not construct new constituents (it is the role of labelling actions), it is optimal for precision.

B Hyperparameters

The list of hyperparameters is presented in Table 5.

- We use learning rate warm-up (linear increase from 0 to t_{1000} during the first 1000 steps).
- Before the t^{th} update, we add Gaussian noise to the gradient of every parameter with mean 0 and variance $\frac{0.01}{(1+t)^{0.55}}$ (Neelakantan et al., 2015).
- All experiments use greedy search decoding (we did not experiment with beam search).

Architecture hyperparameters	
Dimension of word embeddings	32
Dimension of character embeddings	100
Dimension of character bi-LSTM state	50 for each direction
Dimension of sentence-level bi-LSTM	200 for each direction
Dimension of hidden layers for the action scorer	200
Activation functions	$\tanh$ for all hidden layers
Optimization hyperparameters	
Initial learning rate	$l_0 = 0.01$
Learning rate decay	$l_t = \dfrac{l_0}{1 + t \cdot 10^{-7}}$ for step number t
Dropout for tagger input	0.5
Dropout for parser input	0.2
Training epochs	100
Batch size	1 sentence
Optimization algorithm	Averaged SGD (Polyak and Juditsky, 1992; Bottou, 2010)
Word and character embedding initialization	$\mathcal{U}([-0.1, 0.1])$
Other parameters initialization (including LSTMs)	Xavier (Glorot and Bengio, 2010)
Gradient clipping (norm)	100
Dynamic oracle p	0.15

Table 5: Hyperparameters of the model.

Parser	Setting	Tiger		DPTB	
		tok/s	sent/s	tok/s	sent/s
This work	Python, neural, greedy, CPU	978	64	910	38
MTG (Coavoux et al., 2019)	C++, neural, greedy, CPU	1934	126	1887	80
MTG (Coavoux and Crabbé, 2017a)	C++, perceptron, beam=4, CPU	4700	260		
rparse (Maier, 2015)	Java, perceptron, beam=8, CPU		80		
rparse (Maier, 2015)	Java, perceptron, beam=1, CPU		640		
Corro et al. (2017)	C++, neural, CPU				≈ 7.3

Table 6: Parsing times on development sets in tokens per second (tok/s) and sentences per second (sent/s). The parsing times are presented as reported by authors, they are not comparable across parsers (since the experiments were run on different hardware). Our parser is run on a single core of an Intel i7 CPU.

- Before each training step, we replace a word embedding by an 'UNK' pseudo-word embedding with probability 0.3. We only do this replacement for the least frequent word-types ($\frac{2}{3}$ least frequent word-types). The 'UNK' embedding is then used to represent unknown words.
- We apply dropout at the input of the tagger and the input of action scorers: each single prediction has its own dropout mask.

C Detailed Results

We report indicative parsing times in Table 6. Detailed parsing results —including precision, recall, F-score and part-of-speech tag accuracy— are presented in Table 7.

		All const.			Disc. const.			POS
Development sets		F	P	R	F	P	R	Acc.
English (DPTB)	static	91.1	91.1	91.2	68.2	75.3	62.3	97.2
	dynamic	91.4	91.5	91.3	70.9	76.1	66.4	97.2
German (Tiger)	static	87.4	87.8	87.0	61.7	64.4	59.2	98.3
	dynamic	87.6	88.2	87.0	62.5	68.6	57.3	98.4
German (Negra)	static	83.6	83.8	83.4	51.3	53.3	49.5	97.9
	dynamic	84.0	84.7	83.4	54.0	58.1	50.5	98.0
Test sets		F	P	R	F	P	R	Acc.
English (DPTB)	dynamic	90.9	91.3	90.6	67.3	73.3	62.1	97.6
German (Tiger)	dynamic	82.5	83.5	81.5	55.9	62.4	50.6	98.0
German (Negra)	dynamic	83.2	83.8	82.6	56.3	64.9	49.8	98.0

Table 7: Detailed results. Overall, the positive effect of the dynamic oracle on F-score is explained by its effect on precision.

How Bad are PoS Taggers in Cross-Corpora Settings? Evaluating Annotation Divergence in the UD Project

Guillaume Wisniewski and **François Yvon**
LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, F-91405 Orsay, France
`{guillaume.wisniewski,francois.yvon}@limsi.fr`

Abstract

The performance of Part-of-Speech tagging varies significantly across the treebanks of the Universal Dependencies project. This work points out that these variations may result from divergences between the annotation of train and test sets. We show how the annotation variation principle, introduced by Dickinson and Meurers (2003) to automatically detect errors in gold standard, can be used to identify inconsistencies between annotations ; we also evaluate their impact on prediction performance.

1 Introduction

The performance of Part-of-Speech (PoS) taggers significantly degrades when they are applied to test sentences that depart from training data. To illustrate this claim, Table 1 reports the error rate achieved by our in-house PoS tagger on the different combinations of train and test sets of the French treebanks of the Universal Dependencies (UD) project (Nivre et al., 2018). [1] It shows that depending on the train and test sets considered, the performance can vary by a factor of more than 25.

Many studies (Foster, 2010; Plank et al., 2014) attribute this drop in accuracy to *covariate shift* (Shimodaira, 2000), characterizing the differences between domains by a change in the marginal distribution $p(\mathbf{x})$ of the input (e.g. increase of out-of-vocabulary words, missing capitalization, different usage of punctuation, etc), while assuming that the conditional label distribution remains unaffected.

This work adopts a different point of view : we believe that the variation in tagging performance is due to a *dataset shift* (Candela et al., 2009), i.e. a change in the joint distribution of the features and labels. We assume that this change mainly results from incoherences in the annotations between corpora or even within the same corpus. Indeed, ensuring inter-annotator agreement in PoS tagging is known to be a difficult task as annotation guidelines are not always interpreted in a consistent manner (Marcus et al., 1993). For instance, Manning (2011) shows that many errors in the WSJ corpus are just mistakes rather than uncertainties or difficulties in the task ; Table 2 reports some of these *annotation divergences* that can be found in UD project. The situation is naturally worse in cross-corpora settings, in which treebanks are annotated by different laboratories or groups.

The contribution of this paper is threefold :

— we show that, as already pointed out by de Marneffe et al. (2017), the variation principle of Boyd et al. (2008) can be used to flag potential annotation discrepancies in the UD project. Building on this principle, we introduce, to evaluate the annotation consistency of a corpus, several methods and metrics that can be used, during the annotation to improve the quality of the corpus.

— we generalize the conclusions of Manning (2011), highlighting how error rates in PoS tagging are stemming from the poor quality of annotations and inconsistencies in the resources ; we also systematically quantify the impact of annotation variation on PoS tagging performance for a large number of languages and corpora.

— we show that the evaluation of PoS taggers in cross-corpora settings (typically in domain adaptation experiments) is hindered by systematic annotation discrepancies between the corpora and quantify the impact of this divergence on PoS tagger evaluation. Our observations stress the fact that comparing in- and out-domain scores as many

---

[1] See Section 2 for details regarding our experimental setting

Proceedings of NAACL-HLT 2019, pages 218–227
Minneapolis, Minnesota, June 2 - June 7, 2019.

test → ↓ train	FTB	GSD	ParTUT	SRCMF	Sequoia	Spoken	PUD
FTB	**2.8%**	7.0%	6.5%	45.4%	5.4%	18.7%	12.9%
GSD	6.7%	**3.7%**	7.2%	45.5%	5.4%	16.3%	10.2%
ParTUT	11.2%	10.9%	**5.9%**	55.7%	11.3%	22.9%	15.8%
SRCMF	38.8%	37.8%	36.2%	**7.5%**	37.4%	34.7%	36.1%
Sequoia	7.5%	7.5%	8.4%	48.0%	**4.0%**	19.3%	13.6%
Spoken	32.1%	30.3%	25.7%	51.8%	29.5%	**7.9%**	30.1%

Table 1: Error rate (%) achieved by a PoS tagger trained and tested on all possible combinations of the French train and test sets of the UD project. To mitigate the variability of our learning algorithm, all scores are averaged over 10 training sessions.

works do (e.g. to evaluate the quality of a domain adaptation method or the measure the difficulty of the domain adaptation task) can be flawed and that this metrics has to be corrected to take into account the annotation divergences that exists between corpora.

The rest of this paper is organized as follows. We first present the corpora and the tools used in our experiments (§ 2). We then describe the annotation variation principle of Dickinson and Meurers (2003) (§ 3) and its application to the treebanks of the Universal Dependencies project (§ 4). We eventually assess the impact of annotation variations on prediction performance (§ 5 and § 6).

The code and annotations of all experiments are available on the first author website.[2] For the sake of clarity, we have only reported our observations for the English treebanks of the UD project and, sometimes, for the French treebanks (because it has seven treebanks). Similar results have however been observed for other languages and corpora.

2 Experimental Setting

Data All experiments presented in this work use the Universal Dependencies (UD) 2.3 dataset (Nivre et al., 2018) that aims at developing cross-linguistically consistent treebank annotations for a wide array of languages.

This version of the UD project contains 129 treebanks covering 76 languages. Among those, 97 treebanks define a train set that contains between 19 sentences and 68,495 sentences and a test set that contains between 34 and 10,148 sentences. For 21 languages, several test sets are available : there are, for instance, 7 test sets for French, 6 for English, 5 for Czech and 4 for Swedish, Chinese, Japanese, Russian and Italian. Overall, it is possible to train and test 290 taggers (i.e. there are 290 possible combinations of a train and a test set of the same language), 191 of these *conditions* (i.e. pairs of a train set and a test set) correspond to a cross-corpus setting and can be considered for domain adaptation experiments.

Many of these corpora[3] result from an automatic transformation (with, for some of them, manual corrections) from existing dependency or constituent treebanks (Bosco et al., 2013; Lipenkova and Souček, 2014). Because most treebanks have been annotated and/or converted independently by different groups,[4] the risk of inconsistencies and errors in the application of annotation guidelines is increased. There may indeed be several sources of inconsistencies in the gold annotations : in addition to the divergences in the theoretical linguistic principles that governed the design of the original annotation guidelines, inconsistencies may also result from automatic (pre-)processing, human post-editing, or human annotation. Actually, several studies have recently pointed out that treebanks for the same language are not consistently annotated (Vilares and Gómez-Rodríguez, 2017; Aufrant et al., 2017). In a closely related context, Wisniewski et al. (2014) have also shown that, in spite of common annotation guidelines, one of the main bottleneck in cross-lingual transfer between UD corpora is the difference in the annotation conventions across treebanks and languages.

---

[2] `https://perso.limsi.fr/wisniews/recherche/#coherence`

[3] For PoS, only 23 treebanks have been manually annotated natively with the Universal PoS tagset.

[4] almost 65% of the UD contributors have participated in the annotation of only one corpus; for more than 15% of the treebanks all contributors have annotated a single corpus.

①	◇ With regard to the effect of the programme **on the convergence of high level**$_{\text{ADJ}}$ **training for trainers ,** it was not possible to make an assessment as there was not sufficient information on the link between national strategies and the activities under Pericles . ◇ With a view to enabling the assessment of the effect of the programme , among others **on the convergence of high level**$_{\text{NOUN}}$ **training for trainers ,** the evaluator recommends the preparation of a strategy document , to be finalised before the new Pericles enters into effect .
②	◇ **Notice**$_{\text{NOUN}}$ **Regarding Privacy and Confidentiality : PaineWebber reserves the right to monitor and review the content of all e-mail communications sent and or received by its employees .** ◇ **Notice**$_{\text{PROPN}}$ **Regarding Privacy and Confidentiality : PaineWebber reserves the right to monitor and review the content of all e-mail communications sent and or received by its employees .**
③	◇ The above applies to the Work as incorporated in a Collective Work , but this does not require the Collective Work apart from the **Work itself to be made subject**$_{\text{ADJ}}$ **to the terms of this License.** ◇ The above applies to the Derivative Work as incorporated in a Collective Work , but this does not require the Collective Work apart from the Derivative **Work itself to be made subject**$_{\text{NOUN}}$ **to the terms of this License .**

Table 2: Examples of annotation divergences in the English Web Treebank (EWT) corpus : these sentences share some common words (in bold) that do not have the same annotation. Only the labels that differ are represented.

PoS tagger In all our experiments, we use a history-based model (Black et al., 1992) with a LaSO-like training method (Daumé III and Marcu, 2005). This model reduces PoS tagging to a sequence of multi-class classification problems : the PoS of the words in the sentence are predicted one after the other using an averaged perceptron. We consider the standard feature set for PoS tagging (Zhang and Nivre, 2011) : current word, two previous and following words, the previous two predicted labels, etc. This 'standard' feature set has been designed for English and has not been adapted to the other languages considered in our experiments.

Our PoS tagger achieves an average precision of 91.10% over all UD treebanks, a result comparable to the performance of `UDPipe` 1.2 (Straka and Straková, 2017), the baseline of CoNLL'17 Shared Task '*Multilingual Parsing from Raw Text to Universal Dependencies*' that achieves an average precision of 91.22%. When not otherwise specified, all PoS tagging scores reported below are averaged over 10 runs (i.e. independent training of a model and evaluation of the test performance).

3 Annotation variation principle

The *annotation variation principle* (Boyd et al., 2008) states that if two identical sequences appear with different annotations, one of these two label sequences may be inconsistently annotated. Our work relies on this principle to identify discrepancies in the PoS annotation of treebanks.

We call *repeat* a sequence of words that appears in, at least, two sentences and *suspicious repeat* a repeat that is annotated in at least two different ways. Identifying suspicious repeats requires, first, to find all sequences of words that appear in two different sentences ; this is an instance of the *maximal repeat problem* : a *maximal repeat*, is a substring that occurs at least in two different sentences and cannot be extended to the left or to right to a longer common substring. Extracting maximal repeats allows us to find all sequence of words common to at least two sentences without extracting all their substrings. This problem can be solved efficiently using Generalized Suffix Tree (GST) (Gusfield, 1997) : if the corpus contains n words, extracting all the maximal repeats takes $\mathcal{O}(n)$ to build the GST and $\mathcal{O}(n)$ to list all the repeats. PoS annotations for these repeats can then be easily extracted and the ones that are identical can be filtered out to gather all suspicious repeats in a set of corpora. A detailed description of our implementation can be found in (Wisniewski, 2018).

Filtering heuristics Suspicious repeats can of course correspond to words or structures that are

ambiguity	◇ The early voting suggests that this time the Latin Americans **will come out to**$_{\text{PART}}$ vote in greater numbers , but it is unclear whether the increase will have an impact . ◇ Keep his cage open and go on your computer , or read a book , etc and maybe he **will come out to**$_{\text{ADP}}$ you .
inconsistency	◇ Trudeau will extend that invitation to the 45th **president**$_{\text{NOUN}}$ **of the United**$_{\text{ADJ}}$ **States**$_{\text{NOUN}}$, whoever he or she may be . ◇ I am GEORGE WALKER BUSH , son of the former **president**$_{\text{PROPN}}$ **of the United**$_{\text{PROPN}}$ **States**$_{\text{PROPN}}$ of America George Herbert Walker Bush , and currently serving as President of the United States of America .

Table 3: Example of an actual ambiguity and of an annotation inconsistency between the English EWT and PUD corpora. Repeated words are in bold and words with different PoS in red.

truly ambiguous. We consider two heuristics to filter out suspicious repeats. First with the *size heuristic*, we assume that longer suspicious repeats are more likely to result from annotation errors than shorter ones. For instance, Table 2 displays suspicious repeats with at least 10 words that all stem from an annotation error.

Second, with the *disjoint heuristic*, we assume that actual ambiguities will be reflected in intra-corpus suspicious repeats, whereas errors will likely correspond to cases where differences in labelings are observed in different corpora. Formally, the *disjoint heuristic* flags repeats m occurring in at least two corpora A and B, and such that the set of labelings of m observed in A are disjoint from the set of labelings observed in B.

For instance, in French, *“la porte”* can either be a determiner and a noun (e.g. in the sentence *“la porte est fermée”* — the door is closed) or a pronoun followed by a verb (e.g. in the sentence *“je la porte”* — I carry her). Observing these two possible labelings in at least two corpora is a good sign of an actual ambiguity. The disjoint heuristic allows us to detect that this suspicious repeat is an actual ambiguity. To reiterate, the intuition beyond the disjoint heuristic is that for ambiguities, the two possible annotations will appear in, at least, one of the two corpora.

Conversely, systematic divergences in labeling observed across corpora are likely to be errors : for instance, in English, depending on the treebank, cardinal points are labeled as either proper nouns or as nouns. In this case, the set of labelings of the repeats in the first corpus is disjoint from the set of labeling in the second corpus and the disjoint heuristic captures the annotation inconsistency.

Analyzing filtering heuristics To further analyze these two heuristics, we have manually annotated the suspicious repeats between the train set of the English EWT corpus and the test set of the English PUD corpus. For each suspicious repeat, we record whether it is an annotation error or an actual ambiguity. Examples of annotations are given in Table 3.

Results are in Table 4. It appears that, for the heuristics considered, a large part of the suspicious repeats correspond to annotation discrepancies rather than ambiguities. In many cases, these discrepancies result from systematic divergences in the interpretation of the UD guidelines. [5] For instance, the contraction “`n't`” is always labeled as a particle in the train set of the EWT corpus, but either as particle or an adverb in the PUD corpus. Most of these systematic differences involve distinction between nouns and proper nouns, auxiliaries and verbs and adjectives and verbs (for past participles).

4 Quantifying Annotation Divergence in the UD Corpora

4.1 Annotation Variations in the UD

We will first show how the annotation variation principle allows us to characterize the noise and/or the difficulty of PoS tagging. Table 5 reports the number of repeats and suspicious repeats in the English corpora of the UD project. These numbers have been calculated by applying the method described in the previous section to the concatenation of train, development and test sets of each treebanks. To calibrate these measures, we conducted

---

[5] Discrepancies are not only due to improper interpretations of the guidelines, but also sometimes to actual ambiguities in the annotation rules.

heuristic	# susp. repeats	# inconsistencies
size=4	28	22 78.6%
size=3	214	153 71.5%
disjoint	580	407 70.3%
none	2507	—

Table 4: Percentage of suspicious repeats between the EWT and PUD corpora that contain an annotation inconsistency according to a human annotator either when the disjoint heuristic is used or when only suspicious repeats with at least *n* words are considered.

the same experiments with the Wall Street Journal (Marcus et al., 1993),[6] the iconic corpus of PoS tagging for which a thorough manual analysis of the annotation quality is described in (Manning, 2011).

The observations reported in Table 5 show that the number of repeats varies greatly from one corpus to another, which is not surprising considering the wide array of genres covered by the treebanks that includes sentences written by journalists or learner of English (the genres with the largest number of repeats) or sentences generated by users on social media (that contain far less repeated parts). These observations also show that the percentage of repeats that are not consistently annotated is slightly larger in the UD treebanks than in the WSJ, a corpus in which a manual inspection of the corpus reveals that many variations are 'mistakes' rather than representing uncertainties or difficulties in the PoS prediction (Manning, 2011).

More interestingly, Table 6 shows the percentage of repeats that are not consistently annotated for all possible combinations of a train and a test sets (ignoring sequences of words that do not appear at least once in both corpora). It appears that in all cases there are (sometimes significantly) more variations in annotations in cross-treebank settings than in situations where the train and the test sets belong to the same treebank. This observation suggests that there may be systematic differences in the annotations of different treebanks which could make the domain adaptation setting artificially more difficult.

Treebank	# sent.	% sent. repeat	% var.
ESL	5,124	79.0	10.4
EWT	16,622	13.1	9.0
GUM	4,399	10.5	8.5
LinES	4,564	10.9	11.8
PUD	1,000	2.7	8.7
ParTUT	2,090	18.8	9.0
WSJ	21,928	66.1	8.4

Table 5: Percentage of sentences with a repeat of at least three words in the English treebanks (`% sent. repeat`) and percentage of these repeats that are not labeled consistently (`% var.`).

4.2 How do treebanks differ ?

To characterize the difference between two treebanks, we measure the error rate of a binary classifier deciding from which corpus an annotated sentence is coming from.[7] Intuitively, the higher this error rate, the more difficult it is to distinguish sentences of the two corpora and the more similar the treebanks are. More formally, it can be shown (Ben-David et al., 2010) that this error rate is an estimation of the $\mathcal{H}$-divergence (Kifer et al., 2004), a metric introduced in machine learning theory to quantify the impact of a change in domains by measuring the divergence between the distributions of examples sampled from two datasets.

In our experiments, we use a Naive Bayes classifier[8] and three sets of features to describe a sentence pair and their annotation : *words*, in which each example is represented by the bag of its 1-gram and 2-gram of words ; *labels*, in which examples are represented in the same way, but this time, considering PoS ; and *combi* which uses the same representation after the words of all the treebanks have been concatenated with their PoS. The first set aims at capturing a potential covariate shift, the last two target divergence in annotations. To reduce the impact of the strong between-class imbalance,[9] in all our experiments we sub-sample the largest set to ensure that the two datasets we try to distinguish always have the same number of examples. All scores in this experiment are averaged over 20 train-test splits.

---

[6] The Penn Treebank tagset has been manually converted to the Universal PoS tagset using the mapping of (Petrov et al., 2012) generalized to the extended UD PoS tagset.

[7] More precisely, the classifier analyses pairs of sentences and predicts whether they belong to th same corpus or not.

[8] We used the implementation provided by (Pedregosa et al., 2011) without tuning any hyper-parameters. Experiments with a logistic regression show similar results.

[9] The ratio between the number of examples in the two corpora can be as large as 88.

↓ train / test →	ESL	EWT	GUM	LinES	ParTUT	WSJ	PUD
ESL	**10.0%**	11.7%	10.6%	11.1%	10.0%	12.9%	10.9%
EWT	11.8%	**8.7%**	9.1%	10.1%	8.9%	18.8%	9.2%
GUM	14.3%	9.0%	**8.2%**	11.6%	8.5%	15.8%	11.1%
LinES	16.9%	12.8%	12.6%	**12.4%**	12.5%	16.6%	14.2%
ParTUT	13.8%	10.5%	9.9%	12.0%	**9.0%**	14.9%	12.5%
WSJ	9.0%	9.9%	9.0%	9.5%	8.5%	**8.2%**	9.6%

Table 6: Percentage of repeats between a train and a test sets that are not annotated consistently. In-domain settings (i.e. when the train and test sets come from the same treebank) are reported in bold; for each train set, the most consistent setting is underlined.

Table 7 reports the results achieved with the different features sets averaged over all combinations of a train and a test set of the same language and gives the percentage of conditions for which each feature set achieved the best results; Figure 1 details these results for the English and French treebanks. Results for other languages show similar patterns. These results suggest that, in many cases, it is possible to accurately identify from which treebank a sentence and its annotation are coming, although these raw numbers are difficult to interpret as prediction performances are averaged over many different experimental conditions. In more than 50% of the cases, combining words to their PoS results in the best performance, which is consistent to the qualitative study reported in Section 3 : some words appear in two corpora with different PoS allowing to distinguish these corpora. This observation strongly suggests that divergence in annotations across corpora are often genuine.

5 Impact of annotation variation on prediction performance

To study annotation divergence in the UD project, we propose to analyze suspicious repeats (i.e. sequence of repeated words with different annotations). We start by extracting all the suspicious repeats that can be found when considering all the possible combinations of a train set and a test or development set of a given language. These matches are then filtered using the heuristics described in §3. There are, overall, 357,301 matches in the UD project, 69,157 of which involve 3 words or more and 14,142 5 words or more; the disjoint heuristic selects 122,634 of these matches (see Table 8 in §A).

features	median	% best
words	78.2	31.0
labels	70.9	13.5
combi	78.8	55.5

Table 7: Precision (%) achieved over all cross-treebank conditions by a classifier identifying to which treebank a sentence belongs to.

To highlight the connection between prediction errors and annotation divergence, we compute, for each possible combination of a train and a test set (considering all languages in the UD project), the correlation between the error rate achieved on a corpus `B` when training our PoS on a corpus `A` and the number of suspicious repeats between `A` and `B` normalized by the number of tokens in `A` and `B`. The Spearman correlation coefficient between these two values is 0.72 indicating a correlation generally qualified as ‘strong’ following the interpretation proposed by (Cohen, 1988) : the more there are sequences of words with different annotations in the train and test sets, the worse the tagging performance, which shows that annotation inconsistencies play an important role in explaining the poor performance of PoS tagger on some conditions.

For a more precise picture, we also estimate the number of suspicious repeats that contain a prediction error. Using the disjoint heuristics to filter suspicious repeats, it appears that 70.2% (resp. 73.0%) of the suspicious repeats for English (resp. French) contain a prediction error. As expected, these numbers fall to 51.7% (resp. 49.9%) when the suspicious repeats are not filtered and therefore contain more ambiguous words. Figure 2 displays a similar trend when the suspicious repeats are filtered by their length; similar results are observed for all other languages.

These observations suggest that annotation variations often results in prediction errors, espe-

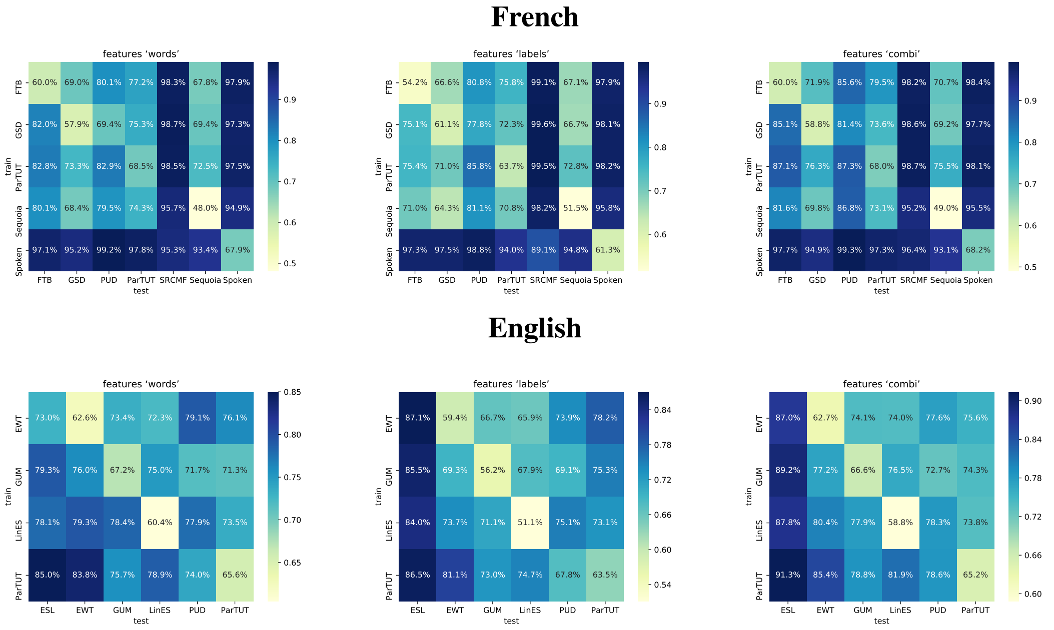


Figure 1: Precision of a classifier identifying to which French (top) or English (bottom) treebank a sentence belongs to. Train corpora are on the y-axis and test corpora on the x-axis.

cially when there are good reasons to assume that the variation actually stems from an inconsistency.

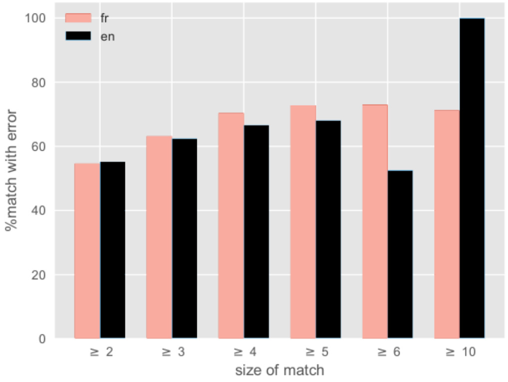


Figure 2: Percentage of suspicious repeats that contain at least one prediction error in function of their size.

6 Re-Assessing the Performance of PoS Tagger in Cross-Corpus Setting

To evaluate the impact of annotation errors on prediction performance, we propose, for each combination of a train and a test set, to train a PoS tagger and compare $\varepsilon_{\text{full}}$, the error rate achieved on the full test set to $\varepsilon_{\text{ignoring}}$ the error rate achieved ignoring errors that occur in a suspicious repeat. More precisely, $\varepsilon_{\text{ignoring}}$ is defined as :

$$\varepsilon_{\text{ignoring}} = \frac{\#\{\text{err}\} - \#\{\text{err in suspicious repeats}\}}{\#\{\text{words}\}} \quad (1)$$

where $\#\{\text{err in suspicious repeats}\}$ in the number of errors in the suspicious repeats that have survived filtering. Intuitively $\varepsilon_{\text{ignoring}}$ can be seen as an 'oracle' score corresponding to a tagger that would always predict the labels of suspicious repeat correctly. In the following, We will consider three different filters : the disjoint heuristic, keeping only suspicious repeats with more than three words and keeping all of them.

Figure 3 reports these errors rates for French and English. Results for other languages show similar results. As expected, ignoring errors in suspicious repeats significantly improve prediction performance. It even appears that $\varepsilon_{\text{ignoring}}$ is often on par with the score achieved on in-domain sets. Overall, in more than 43% (resp. 25%) of all the conditions the error rate ignoring errors in suspicious repeats filtered with the disjoint heuristic (resp. minimum heuristic) is lower than the error rate achieved on in-domain data. These values are naturally over-estimated as, in these experiments, we remove all potential annotation errors as well as words and structures that are ambiguous and therefore are more difficult to label. They can however be considered as lower-bound on the predic-

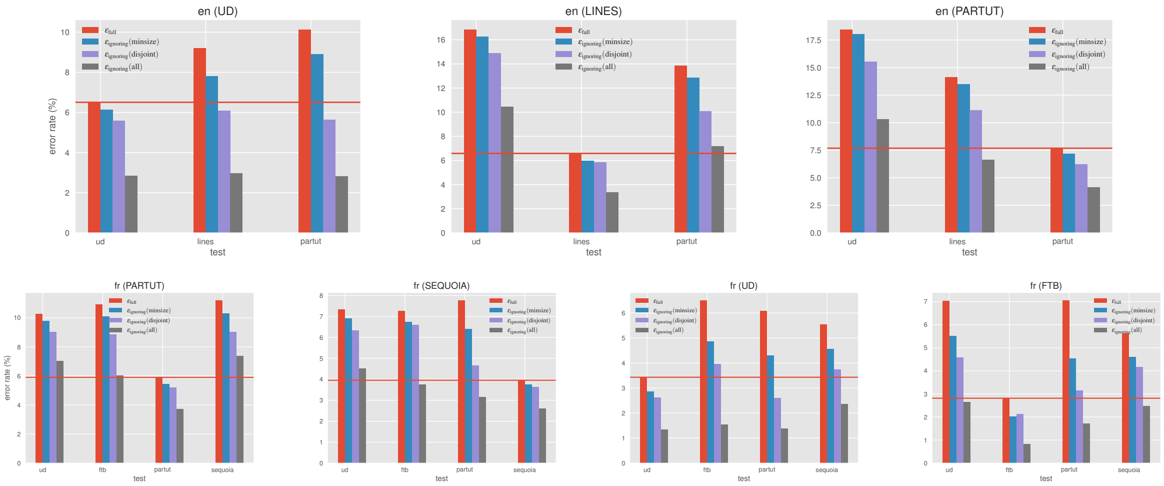


Figure 3: Error rate achieved by a PoS tagger on the different English treebanks of the UD project when errors in suspicious repeats are ignored. The red line indicates the error rate on in-domain data.

tion quality.

To assess their quality, we have manually checked all the suspicious repeats between the train set of French UD and the test set of the French FTB correcting inconsistencies and errors (almost 2,000 PoS were modified). [10] When trained on the original UD corpus, the PoS tagger achieved an error rate of 6.78% on the FTB corpus (4.51% on in-domain data). After correcting inconsistencies, the out-domain error rate falls down to 5.11%. This value is close to the error rate ignoring suspicious repeats containing three and more words, showing the validity of the heuristics we have considered.

7 Conclusion

In this work, we have shown that, for PoS tagging, many prediction errors in cross-corpora settings (which is a typical domain adaptation scenario) stem from divergence between annotations. We have also described a method to quantify this divergence. We have only considered here corpora from the UD project and PoS annotation, but we consider that our method is very generic and can be easily applied to other corpora or tasks (e.g. tokenization, dependency parsing, etc.) that we will address in future work. We also plan to see how the different experiments we have made to identify annotation errors and inconsistencies can be used during the annotation process to reduce the workload of annotators and help them creating high-quality corpora.

Acknowledgements

This work has been partly funded by the French *Agence Nationale de la Recherche* under ParSiTi (ANR-16-CE33-0021) and MultiSem projects (ANR-16-CE33-0013).

References

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2017. LIMSI@CoNLL'17 : UD shared task. In *Proceedings of the CoNLL 2017 Shared Task : Multilingual Parsing from Raw Text to Universal Dependencies*, pages 163–173, Vancouver, Canada. Association for Computational Linguistics.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2) :151–175.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars : Using richer models for probabilistic parsing. In *Proceedings of the Workshop on Speech and Natural Language*, HLT'91, pages 134–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cristina Bosco, Simonetta Montemagni, and Maria Simi. 2013. Converting italian treebanks : Towards an Italian stanford dependency treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria. Association for Computational Linguistics.

Adriane Boyd, Markus Dickinson, and W. Detmar Meurers. 2008. On detecting errors in dependency

---

[10] The 'corrected' corpora will be made available upon publication. In this experiment, the impact of annotation errors is under-estimated as we have only corrected errors that appear in a suspicious repeat without trying to 'generalize' these corrections to words that appear only in one corpus.

treebanks. *Research on Language and Computation*, 6(2) :113–137.

Joaquin Q. Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. 2009. *Dataset Shift in Machine Learning*. The MIT Press.

Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Routledge.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization : Approximate large margin methods for structured prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML'05, pages 169–176, New York, NY, USA. ACM.

Markus Dickinson and W. Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 107–114, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jennifer Foster. 2010. "cba to check the spelling" : Investigating parser performance on discussion forum posts. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384, Los Angeles, California. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 180–191. VLDB Endowment.

Janna Lipenkova and Milan Souček. 2014. Converting Russian dependency treebank to Stanford typed dependencies representation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2 : Short Papers*, pages 143–147, Gothenburg, Sweden. Association for Computational Linguistics.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100% : Is it time for some linguistics ? In *Proceedings of the Computational Linguistics and Intelligent Text Processing, 12th International Conference*, CICLing 2011, pages 171–189. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english : The penn treebank. *Comput. Linguist.*, 19(2) :313–330.

Marie-Catherine de Marneffe, Matias Grioni, Jenna Kanerva, and Filip Ginter. 2017. Assessing the annotation consistency of the universal dependencies corpora. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 108–115. Linköping University Electronic Press.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, and other. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Barbara Plank, Anders Johannsen, and Anders Søgaard. 2014. Importance weighting and unsupervised domain adaptation of POS taggers : a negative result. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–973, Doha, Qatar. Association for Computational Linguistics.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2) :227 – 244.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task : Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

David Vilares and Carlos Gómez-Rodríguez. 2017. A non-projective greedy dependency parser with bidirectional LSTMs. In *Proceedings of the CoNLL 2017 Shared Task : Multilingual Parsing from Raw Text to Universal Dependencies*, pages 152–162, Vancouver, Canada. Association for Computational Linguistics.

Guillaume Wisniewski. 2018. Errator : a tool to help detect annotation errors in the universal dependencies project. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1779–1785, Doha, Qatar. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of ACL 2011, the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

A Appendices

# repeated words	# repeats	# suspicious
2	4,366,885	146,516 3.36%
3	1,977,969	44,800 2.26%
4	622,192	9,684 1.56%
5	183,680	1,998 1.09%
6	60,869	509 0.84%
7	25,697	158 0.61%
8	13,132	123 0.94%
9	7,572	61 0.81%
≥ 10	24,629	264 1.07%

Table 8: Number of repeated sequence of words across the different combinations of a train set and a test set (‘repeats’ column) and number of these sequences that are annotated differently (‘suspicious repeats’ column) when no filtering is applied.

CCG Parsing Algorithm with Incremental Tree Rotation

Miloš Stanojević
School of Informatics
University of Edinburgh
m.stanojevic@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
steedman@inf.ed.ac.uk

Abstract

The main obstacle to incremental sentence processing arises from right-branching constituent structures, which are present in the majority of English sentences, as well as from optional constituents that adjoin on the right, such as right adjuncts and right conjuncts. In CCG, many right-branching derivations can be replaced by semantically equivalent left-branching incremental derivations.

The problem of right-adjunction is more resistant to solution, and has been tackled in the past using *revealing*-based approaches that often rely either on the higher-order unification over lambda terms (Pareschi and Steedman, 1987) or heuristics over dependency representations that do not cover the whole CCGbank (Ambati et al., 2015).

We propose a new incremental parsing algorithm for CCG following the same *revealing* tradition of work but having a purely syntactic approach that does not depend on access to a distinct level of semantic representation. This algorithm can cover the whole CCGbank, with greater incrementality and accuracy than previous proposals.

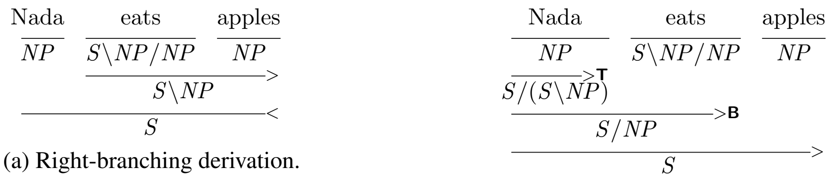
1 Introduction

Combinatory Categorial Grammar (CCG) (Ades and Steedman, 1982; Steedman, 2000) is a mildly context sensitive grammar formalism that is attractive both from a cognitive and an engineering perspective. Compared to other grammar formalisms, the aspect in which CCG excels is incremental sentence processing. CCG has a very flexible notion of constituent structure which allows (mostly) left-branching derivation trees that are easier to process incrementally. Take for instance the derivation tree in Figure 1a. If we use a non-incremental shift-reduce parser (as done in the majority of transition-based parsers for CCG (Zhang and Clark, 2011; Xu et al., 2014; Xu, 2016)) we will be able to establish the semantic connection between the subject "Nada" and the verb "eats" only when we reach the end of the sentence. This is undesirable for several reasons. First, human sentence processing is much more incremental, so that the meaning of the prefix "Nada eats" is available as soon as it is read (Marslen-Wilson, 1973). Second, if we want a predictive model—either for better parsing or language modelling—it is crucial to establish relations between the words in the prefix as early as possible.

To address this problem, a syntactic theory needs to be able to represent partial constituents like "Nada eats" and have mechanisms to build them just by observing the prefix. In CCG solutions for these problems come out of the theory naturally. CCG categories can represent partial structures and these partial structures can combine into bigger (partial) structures using CCG combinators recursively. Figure 1b shows how CCG can incrementally process the example sentence via a different derivation tree that generates the same semantics more incrementally by being left-branching.

This way of doing incremental processing seems straightforward except for one obstacle: optional constituents that attach from the right, i.e. right adjuncts. Because they are optional, it is impossible to predict them with certainty. This forces an eager incremental processor to make an uninformed decision very early and, if later that decision turns out to be wrong, to backtrack to repair the mistake. This behaviour would imply that human processors have difficulty in processing right adjuncts, but that does not seem to be the case. For instance, let's say that after incrementally processing "Nada eats apples" we encounter right adjunct "regularly" as in Figure 2a. The parser will be stuck at this point because there is no way to at-

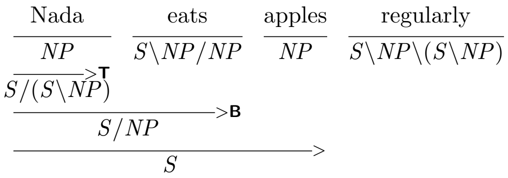
Proceedings of NAACL-HLT 2019, pages 228–239
Minneapolis, Minnesota, June 2 - June 7, 2019.



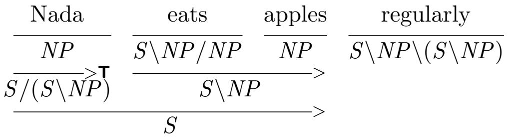
(a) Right-branching derivation.

(b) Left-branching derivation.

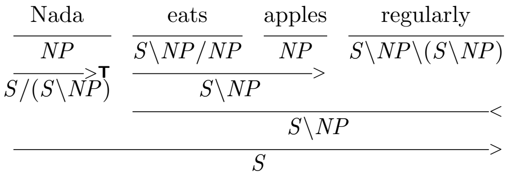
Figure 1: Semantically equivalent CCG derivations.



(a) Problem – $S\backslash NP$ that needs to be modified was never built.



(b) Incremental tree rotation *reveals* $S\backslash NP$.



(c) Right adjunct is attached to the revealed node.

Figure 2: Right adjunction.

tach the right adjunct of a verb phrase to a sentence constituent. A simple solution would be some sort of limited back-tracking where we would look if we could extract the verb-phrase, attach its right adjunct, and then put the derivation back together. But how do we do the extraction of the verb-phrase "eats apples" when that constituent was never built during the incremental left-branching derivation?

Pareschi and Steedman (1987) proposed to *reveal* the constituent that is needed, the verb-phrase in our example, by having an elegant way of reanalysing the derivation. This reanalysis does not repeat parsing from scratch but instead runs a single CCG combinatory rule backwards. In the example at hand, first we recognise that right adjunction needs to take place because we have a category of shape $X\backslash X$ (concretely $(S\backslash NP)\backslash(S\backslash NP)$ but in the present CCG notation slashes "associate to the left", so we drop the first pair of brackets). Thanks to the type of the adjunct we know that the constituent that needs to be revealed is of type X, in our case $S\backslash NP$. Now, we take the constituent on the left of the right adjunct, in our example constituent S, and look for CCG category Y and combinatory rule C that satisfies the following relation: $C(Y, S\backslash NP) = S$. The solution to this type equation is $Y{=}NP$ and $C{=}{<}$.

To confine revealing to delivering constituents that the parser *could have* built if it had been less greedy for incrementality, and exclude revelation of unsupported types, such as PP in Figure 2a, the process must be constrained by the actual derivation. Pareschi and Steedman proposed to do so by accessing the semantic representation in parallel, using higher-order unification, which is in general undecidable and may be unsound unless defined over a specific semantic representation.

Ambati et al. (2015) propose an alternative method for revealing where dependencies are used as a semantic representation (instead of first-order logic) and special heuristics are used for revealing (instead of higher order unification). This is computationally a much more efficient approach and appears sound, but requires distinct revealing rules for each constituent type and has specific difficulties with punctuation.

In this paper we propose a method of revealing that does not depend on any specific choice of semantic representation, can discover multiple possible revealing options if they are available, is sound and complete and computationally efficient, and gives state-of-the-art parsing results. The algorithm works by building left-branching derivations incrementally, but, following Niv (1993, 1994), as soon as a left branching derivation is built, its derivation tree is rebalanced to be right-branching. When all such constituents' derivation trees are right-branching, revealing becomes a trivial operation where we just traverse the right spine looking for the constituent(s) of the right type to be modified by the right adjunct.

We call this rebalancing operation *tree rota-*

tion since it is a technical term established in the field of data structures for similar operation of balanced binary search trees (Adelson-Velskii and Landis, 1962; Guibas and Sedgewick, 1978; Okasaki, 1999; Cormen et al., 2009). Figure 2b shows the right rotated derivation "Nada eats apples" next to the adjunct. Here we can just look up the required $S\backslash NP$ and attach the right adjunct to it as in Figure 2c.

2 Combinatory Categorial Grammar

CCG is a lexicalized grammar formalism where each lexical item in a derivation has a category assigned to it which expresses the ways in which the lexical item can be used in the derivation. These categories are put together using combinatory rules.

The binary combinatory rules we use are:

$$\begin{array}{llcll}
X/Y & Y & \Rightarrow & X & (>) \\
Y & X\backslash Y & \Rightarrow & X & (<) \\
X/Y & Y/Z & \Rightarrow & X/Z & (>\mathbf{B}) \\
Y\backslash Z & X\backslash Y & \Rightarrow & X\backslash Z & (<\mathbf{B}) \\
Y/Z & X\backslash Y & \Rightarrow & X/Z & (<\mathbf{B}_{\times}) \\
Y/Z|W & X\backslash Y & \Rightarrow & X/Z|W & (<\mathbf{B}^2_{\times}) \\
X/Y & Y/Z|W & \Rightarrow & X/Z|W & (>\mathbf{B}^2)
\end{array}$$

Each binary combinatory rule has one primary and one secondary category as its inputs. The primary functor is the one that selects; while the secondary category is the one that is selected. In forward combinatory rules the primary functor is always the left argument, while in the backward combinatory rules it is always the right.

It is useful to look at the mentioned combinatory rules in a generalised way. For instance, if we look at forward combinatory rules we can see that they all follow the same pattern of combining X/Y with a category that starts with Y. The only difference among them is how many subcategories follow Y in the secondary category. In case of forward function application there will be nothing following Y so we can treat forward function application as a generalised forward composition combinator of the zeroth order $>\mathbf{B0}$. Standard forward function composition $>\mathbf{B}$ will be a generalised composition of first order $>\mathbf{B1}$ while $>\mathbf{B}^2$ will be $>\mathbf{B2}$. Same generalisation can be applied to backward combinators. There is a low bound on the order of combinatory rules, around 2 or 3.

Following Hockenmaier and Steedman (2007), the proclitic character of conjunctions is captured in a syncategorematic rule combining them with the right conjunct, with the result later combining with the left conjunct [1] :

$$\begin{array}{llcll}
conj & X & \Rightarrow & X[conj] & (>\mathbf{\Phi}) \\
X & X[conj] & \Rightarrow & X & (<\mathbf{\Phi})
\end{array}$$

Some additional unary and binary type-changing rules are also needed to process the derivations in CCGbank (Hockenmaier and Steedman, 2007). We use the same type-changing rules as those described in (Clark and Curran, 2007).

Among the unary combinatory rules the most important one is type-raising. The first reason for that is that it allows CCG to handle constructions like argument cluster coordination in a straightforward way. Second, it allows CCG to be much more incremental as seen from the example in Figure 1b. Type-raising rules are expressed in the following way:

$$\begin{array}{lcll}
X & \Rightarrow & Y/(Y\backslash X) & (>\mathbf{T}) \\
X & \Rightarrow & Y\backslash(Y/X) & (<\mathbf{T})
\end{array}$$

Type-raising, is strictly limited to applying to category types that are *arguments*, such as NP, PP, etc., making it analogous to grammatical *case* in languages like Latin and Japanese, in spite of the lack of morphological case in English.

3 Parsing

CCG derivations can be parsed with the same shift-reduce mechanism used for CFG parsing (Steedman, 2000). In the context of CFG parsing, the shift-reduce algorithm is not incremental, because CFG structures are mostly right-branching, but in CCG by changing the derivation via the combinatory rules we also change the level of incrementality of the algorithm.

As usual, the shift-reduce algorithm consists of a stack of the constituents built so far and a buffer with words that are yet to be processed. Parsing starts with the stack empty and the buffer containing the whole sentence. The end state is a stack with only one element and an empty buffer. Transitions between parser states are:

- *shift(X)* – moves the first word from the buffer to the stack and labels it with category *X*,
- *reduceUnary(C)* – applies a unary combinatory rule *C* to the topmost constituent on the stack,
- *reduceBinary(C)* – applies a binary combinatory rule *C* to the two topmost constituents on

---

[1] This notation differs unimportantly from Steedman (2000) who uses a ternary coordination rule, and more recent work in which conjunctions are $X\backslash X/X$.

the stack.

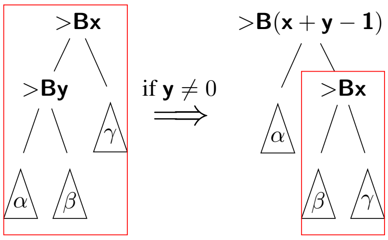
CCG shift-reduce parsers are often built over right-branching derivations that obey Eisner normal form (Eisner, 1996). Processing left-branching derivations is not any different except that it requires an opposite normal form.

Our revealing algorithm adds a couple of modifications to this default shift-reduce algorithm. First, it guarantees that all the trees stored on the stack are right-branching – this still allows left-branching parsing and only adds the requirement of adjusting newly reduced trees on the stack to be right leaning. Second, it adds *revealing* transitions that exploit the right-branching guarantee to apply right adjunction. Both tree rotation and revealing are performed efficiently as described in the following subsections.

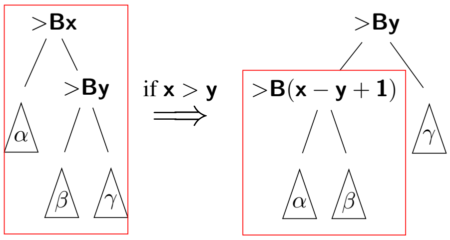
3.1 Tree rotation

A naïve way of enforcing right-branching guarantee is to do a complete transformation of the subtree on the stack into a right-branching one. However, that would be unnecessarily expensive. Instead we do *incremental tree rotation* to right. If we assume that all the elements on the stack are respecting this right-branching form (our inductive case), this state can be disturbed only by *reduceBinary* transition (*shift* just adds a single word which is trivially right-branching and *reduceUnary* does not influence the direction of branching). The *reduceBinary* transition will take two topmost elements on the stack that are already right-branching and put them as children of some new binary node. We need to repair that potential "imperfection" on top of the tree. This is done by recursively rotating the nodes as in Figure 3a.[2]

This figure shows one of the sources of CCG's spurious ambiguity: parent-child relation of the combinatory rules with the same directionality. Here we concentrate on forward combinators because they are the most frequent in our data—most backward combinators disappear with the addition of forward type-raising and the addition of special right adjunct transitions—but the same method can be applied to backward combinatory rules as a mirror image. Having two combinatory rules of the same directionality is necessary but not sufficient condition for spurious ambiguity. As visible on the Figure 3a side condition, the lower combinator must not be >**B0**. The tree rotation function assumes that both of the children are "perfect"—meaning right-branching[3]— and that the only imperfection is on the root node. The method repairs this imperfection on the root by applying the tree rotation transformation, but it also creates a new node as a right child and that node might be imperfect. That is why the method goes down the right node recursively until all the imperfections are removed and the whole tree becomes fully right-branching. In the worst case the method will reach the bottom of the tree, but often only 3 or 4 nodes need to be transformed to make the tree perfectly the right branching The worst case complexity of repairing the imperfection is $O(n)$ which makes the complexity of the whole parsing algorithm $O(n^2)$ for building a single derivation.



(a) Rotate to right.



(b) Rotate to left.

Figure 3: Tree rotation operations. The red square signifies recursion. Variables x and y represent the orders of the combinatory rules.

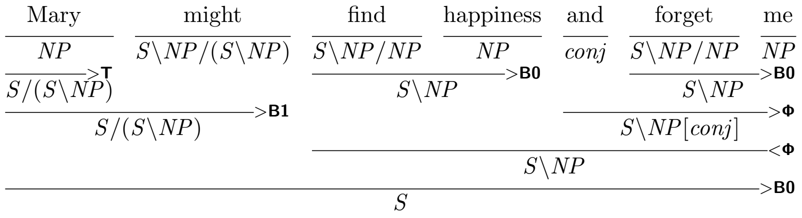
As a running example we will use a derivation tree in Figure 4a for which a transition sequence is given in Figure 4b. Here tree rotation is used in transitions 6 and 8 that introduce imperfections. In transition 6 a single tree rotation at the top was enough to correct the imperfection, while in transition 8 recursive tree rotation function went to depth two.

If the upper and lower combinators are both >**B2** the topmost combinator on the right will be-

---

[2] Although we do not discuss the operations on the semantic predicate-argument structure that correspond to tree-rotation, the combinatory semantics of the rules themselves guarantees that such operations can be done uniformly and in parallel.

[3] By right branching we mean as right branching as it is allowed by CCG formalism and predicate-argument structure.



(a) Derivation tree.

	transition		stack
1	shift	⟹	■ Mary
2	reduceUnary(>**T**)	⟹	■ >**T** Mary
3	shift	⟹	■ >**T** Mary ■ might
4	reduceBinary(>**B1**)	⟹	■ >**B1** (>**T** Mary) might
5	shift	⟹	■ >**B1** (>**T** Mary) might ■ find
6	reduceBinary(>**B1**)	⟹	■ >**B1** (>**B1** (>**T** Mary) might) find
	rotate to right	⟹	■ >**B1** (>**T** Mary) (>**B1** might find)
7	shift	⟹	■ >**B1** (>**T** Mary) (>**B1** might find) ■ happiness
8	reduceBinary(>**B1**)	⟹	■ >**B0** (>**B1** (>**T** Mary) (>**B1** might find)) happiness
	rotate to right	⟹	■ >**B0** (>**T** Mary) (>**B0** (>**B1** might find) happiness)
	rotate to right	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness))
9	shift	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness)) ■ and
10	shift	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness)) ■ and ■ forget
11	shift	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness)) ■ and ■ forget ■ me
12	reduceBinary(>**B0**)	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness)) ■ and ■ >**B0** forget me
13	reduceBinary(>**Φ**)	⟹	■ >**B0** (>**T** Mary) (>**B0** might (>**B0** find happiness)) ■ >**Φ** and (>**B0** forget me)
14	reveal	⟹	■ two options for right adjunction and both with <**Φ** combinator:
			option 1: >**B0** might (>**B0** find happiness)
			option 2: >**B0** find happiness
15	pick option 2	⟹	■ >**B0** (>**T** Mary) (>**B0** might (<**Φ** (>**B0** find happiness) (>**Φ** and (>**B0** forget me))))

(b) Transition sequence for the derivation tree.

Figure 4: Example of the algorithm run over a sentence with tensed VP coordination.

come >**B3**, a combinatory rule that may be unnecessary for defining the competence grammar of human languages, but which is required if parsing performance is to be as incremental as possible. Fortunately, the configuration with two connected >**B2** combinatory rules appears very rarely in CCGbank.

Many papers have been published on using left-branching CCG derivations but, to the best of our knowledge, none of them explains how are they constructed from right-branching CCGbank trees. A very simple algorithm for that can be made using our tree rotation function. Here we use rotation in the opposite direction i.e. rotation to left (Figure 3b). We cannot apply this operation from the top node of the CCGbank tree because that tree does not satisfy the assumption of the algorithm: immediate children are not "perfect" (here perfect means being left-branching). That is why we start from the bottom of the tree with terminal nodes that are trivially "perfect" and apply tree transformation to each node in post-order traversal.

This incremental tree rotation algorithm is inspired by the AVL self-balancing binary search trees (Adelson-Velskii and Landis, 1962) and Red-Black trees (Guibas and Sedgewick, 1978; Okasaki, 1999). The main difference is that here we are trying to do the opposite of AVLs: instead of making the tree perfectly balanced we are trying to make it perfectly unbalanced, i.e. leaning to the right (or left). Also, our imperfections start at the top and are pushed to the bottom of the tree which is in contrast to AVLs trees where imperfections start at the bottom and get pushed to the top.

The last important point about tree rotation concerns punctuation rules. All punctuation is attached to the left of the highest possible node in case of left-branching derivations (Hockenmaier and Bisk, 2010), while in the right-branching derivations we lower the punctuation to the bottom left neighbouring node. Punctuation has no influence on the predicate-argument structure so it is safe to apply this transformation.

3.2 Revealing transitions

If the topmost element on the stack is of the form $X\backslash X$ and the second topmost element on the stack has on its right edge one or more constituents of a type $X|\$$ we allow *reveal* transition.[4] This is a more general way of revealing than approaches of Pareschi and Steedman (1987) and Ambati et al. (2015) who attempt to reveal only constituents of type X while we reveal any type that has X as its prime element (that is the meaning of $X|\$$ notation).

We also treat X[conj] as right adjuncts of the left conjunct. Similarly to the previous case, if the topmost element on the stack is X[conj] and the right edge of the second topmost element on the stack has constituent(s) of type X, they are revealed for possible combination via $<\mathbf{\Phi}$ combinator.

If *reveal* transition is selected, as in transition 14 in Figure 4b, the parser enters into a mode of choosing among different constituents labelled $X|\$$ that could be modified by the right adjunct $X\backslash X$. After particular $X|\$$ node is chosen $X\backslash X$ is combined with it and the rest of the tree above X node is rebuilt in the same way. This rebuild is fully deterministic and is done quickly even though in principle it could take $O(n)$ to compute. Even in the worst case scenario, it does not make the complexity of the algorithm go higher than $O(n^2)$.

The ability of our algorithm to choose among different possible revealing options is unique among all the proposals for revealing. For transition 15 in Figure 4b the parser can choose whether to adjoin (coordinate) to a verb phrase that already contains a left modifier or without. This is similar to *Selective Modifier Placement* strategy from older Augmented Transition Network (ATN) systems (Woods, 1973) which finds all the attachment options that are syntactically legal and then allows the parser to choose among those using some criteria. Woods (1973) suggests using lexical semantic information for this selection, but in his ATN system only handwritten semantic selection rules were used. Here we will also use selection based on the lexical content but it will be broad coverage and learned from the data. This ability to semantically select the modifier's attachment point is essential for good parsing results as will be shown.

4 Neural Model

The neural probabilistic model that chooses which transition should be taken next conditions on the whole state of the configuration in a similar way to RNNG parser (Dyer et al., 2016). The words in the sentence are first embedded using the concatenation of top layers of ELMo embeddings (Peters et al., 2018) that are normalised to L2 norm and then refined with two layers of bi-LSTM (Graves et al., 2005). The neural representation of the terminal is composed of concatenated ELMo embedding and supertag embedding.

The representation of a subtree combines:

- span representation – we subtract representation of the leftmost terminal from the representation of the rightmost terminal as done in LSTM-Minus architecture (Wang and Chang, 2016),
- combinator and category embeddings,
- head words encoding – because each constituent can have a set of heads, for instance arising from coordination, we model representation of heads with DeepSet architecture (Zaheer et al., 2017) over representations of head terminals.

We do not use recursive neural networks like Tree-LSTM (Tai et al., 2015) to encode subtrees because of the frequency of tree rotation. These operations are fast, but they would trigger frequent recomputation of the neural tree representation, so we opted for a mechanism that is invariant to re-branching.

The stack representation is encoded using Stack-LSTM (Dyer et al., 2015). The configuration representation is the concatenation of the stack representation and the representation of the rightmost terminal in the stack. The next *non-revealing* transition is chosen by a two-layer feed-forward network.

If the *reveal* transition is triggered, the system needs to choose which among the candidate nodes $X|\$$ to adjoin the right modifier $X\backslash X$ to. The number of these modifiers can vary so we cannot use a simple feed-forward network to choose among them. Instead, we use the mechanism of *Pointer networks* (Vinyals et al., 2015), which works in a similar way to attention (Bahdanau et al., 2014) except that attention weights are interpreted as probabilities of selecting any particular node. Attention is computed over representations of each candidate node. Because we expect that there

---

[4] The "\$ notation" is from (Steedman, 2000) where \$ is used as a (potentially empty) placeholder variable ranging over multiple arguments.

	Waiting time	Connect-edness
Right-branching	4.29	5.01
Left-branching	2.32	3.15
Ambati et al. (2015)*	0.69	2.15
Revealing our	**0.46**	**1.72**

Table 1: Train set measure of incrementality. *: taken from (Ambati et al., 2015)

could be some preference for attaching adjuncts high or low in the tree we add to the context representation of each node two position embeddings (Vaswani et al., 2017) that encode the candidate node's height and depth in the current tree.

We optimize for maximum log-likelihood on the training set, using only the most frequent supertags and the most important combinators. To avoid discarding sentences with rare supertags and type-changing rules we use all supertags and combinatory rules during training but do not add their probability to the loss function. The number of supertags used is 425, as in the EasyCCG parser, and the combinatory rules that are used are the same as in C&C parser. The loss is minimised for 15 epochs on the training portion of CCGbank (Hockenmaier and Steedman, 2007) using Adam with learning rate 0.001. Dimensionality is set to 128 in all cases, except for ELMo set at 300. Dropout is applied only to the ELMo input with a rate of 0.2. The parser is implemented in Scala using the DyNet toolkit (Neubig et al., 2017) and is available at https://github.com/stanojevic/Rotating-CCG.

5 Experiments

5.1 How incremental is the Revealing algorithm?

To measure the incrementality of the proposed algorithm we use two evaluation metrics: *waiting time* and *connectedness*. Waiting time is the average number of nodes that need to be shifted before the dependency between two nodes is established. The minimal value for a fully incremental algorithm is 0 (the single shift that is always necessary is not counted). Connectedness is defined as the average stack size before a shift operation is performed (the initial two shifts are forced so they are not taken in the average). The minimal value for connectedness is 1. We have computed these measures on the training portion of the CCGbank for standard non-incremental right-branching derivations, the more incremental left-branching derivations and our revealing derivations. We also put in the results numbers for the previous proposal of revealing by Ambati et al. (2015) taken from their paper but these numbers should be taken with caution, because it is not clear from the paper whether the authors computed them in the same way and on the same portion of the dataset as we did. Table 1 results shows that our revealing derivations are significantly more incremental even in comparison to previous revealing proposals, and barely use more than the minimal amount of stack memory.

	heads	SMP	LF	UF	Sup. Tag
Left	yes	—	89.2	95.1	95.0
Right	yes	—	89.1	95.0	**95.1**
Revealing	no	yes	89.3	95.2	94.9
Revealing	yes	no	88.8	94.9	94.9
Revealing	yes	yes	**89.5**	**95.4**	**95.1**

Table 2: Development set F1 results with greedy decoding for CCG dependencies.

5.2 Which algorithm gives the best parsing results?

We have tested on the development set which of the parsing algorithms gives best parsing accuracy. All the algorithms use the same neural architecture and training method except for the revealing operations that require additional mechanisms to choose the node for revealing. This allows us to isolate machine learning factors and see which of the parsing strategies works the best.

There are two methods that are often used for evaluating CCG parsers. They are both based on "deep" dependencies extracted from the derivation trees. The first is from (Clark et al., 2002) and is closer to categorial grammar view of dependencies. The second is from (Clark and Curran, 2007) and is meant to be more formalism independent and closer to standard dependencies (Caroll et al., 1998). We opt for the first option for development as we find it more robust and reliable but we report both types on the test set.

Table 2 shows the results on development set. The *heads* column shows if the head words representation is used for computing the representation of the nodes in the tree. The *SMP* column shows if Selective Modifier Placement is used: whether we choose where to attach right adjunct based only

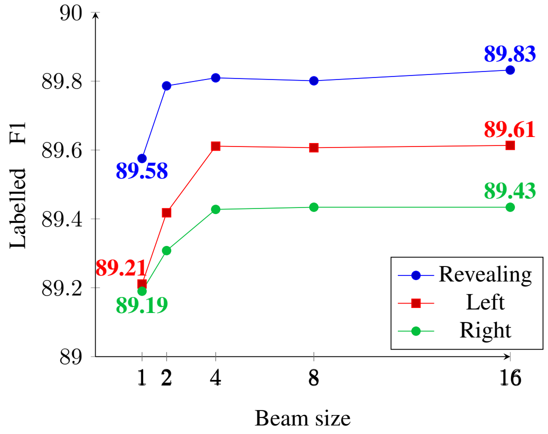


Figure 5: Influence of beam size on the dev results.

on the position embeddings or also on the node's lexical content. First we can see that Revealing approach that uses head representation and does selective modifier placement outperforms all the models both on labelled and unlabelled dependencies. Ablation experiments show that SMP was the crucial component: without it the Revealing model is much worse. This is a clear evidence that attachment heuristics are not enough and also that previous approaches that extract only single revealing option are sub-optimal.

A possible reason why Revealing model works better than Left and Right branching models is that Left and Right models need to commit early on whether there will be a right adjunct in the future or not. If they make a mistake during greedy decoding there will be no way to repair that mistake. This is not an issue for the Revealing model because it can attach right adjuncts at any point and does not need to forecast them. A natural question then is if these improvements of Revealing model will stay if we use a bigger beam. Figure 5 shows exactly that experiment. We see that the model that gains the most from the biggest beam is for the Left-branching condition, which is expected since that is the model that commits to its predictions the most — it commits with type-raising, unlike Right model, and it commits with predicting right adjunction, unlike Revealing model. With an increased beam Left model equals the Revealing greedy model. But if all the models use the same beam the Revealing model remains the best. An interesting result is that the small beam of size 4 is enough to get the maximal improvement. This probably reflects the low degree of lexical ambiguity that is unresolved at each point during parsing.

	Tag	UF	LF
Lewis and Steedman (2014)	93.0	88.6	81.3
Ambati et al. (2015)	91.2	89.0	81.4
Hockenmaier (2003)	92.2	92.0	84.4
Zhang and Clark (2011)	93.1	—	85.5
Clark and Curran (2007)	94.3	93.0	87.6
Revealing (beam=1)	**95.2**	**95.5**	**89.8**
Revealing (beam=4)	**95.4**	**95.8**	**90.2**

Table 3: Test set F1 results for prediction of supertags (Tag), unlabelled (UF) and labelled (LF) CCG dependencies extracted using scripts from Hockenmaier (2003) parser.

		Dev LF	Test LF
Clark and Curran (2007)		83.8	85.2
Lewis and Steedman (2014)		—	86.1
Yoshikawa et al. (2017)		86.8	87.7
Xu et al. (2016)		87.5	87.8
Lewis et al. (2016)	tri-train	87.5	88.1
Vaswani et al. (2016)		87.8	88.3
Lee et al. (2016)	tri-train	88.4	88.7
Yoshikawa et al. (2017)	tri-train	87.7	88.8
Revealing (beam=1)		**90.8**	**90.5**

Table 4: F1 results for labelled dependencies extracted with *generate* program of C&C parser (Clark and Curran, 2007).

5.3 Comparison to other published models

We compute test set results for our Revealing model and compare it to most of the previous results on CCGbank using both types of dependencies. Table 3 shows results with (Clark et al., 2002) style dependencies. Here we get state-of-the-art results by a large margin, probably mostly thanks to the machine learning component of our parser. An interesting comparison to be made is against EasyCCG parser of Lewis and Steedman (2014). This parser uses a neural supertagger of accuracy that is not too far from ours, but the dependencies extracted by our parser are much more accurate. This shows that a richer probabilistic model that we use contributes more to the good results than the exact $A^{\star}$ search that EasyCCG does with a more simplistic model. Another comparison of relevance would be with the revealing model of Ambati et al. (2015) but the comparison of the algorithms is difficult since the machine learning component is very different: Ambati uses a structured perceptron while our model is a heavily parametrized neural network.

In Table 4 we show results with the second type of dependencies used for CCG evaluation. All the models, except Clark and Curran (2007), are neural and use external embeddings. From the presented models only Revealing and Xu et al. (2016) are transition based. All other models have a global search either via CKY or A* search. Our revealing-based parser that does only greedy search is outperforming all of them including those trained on large amounts of unlabelled data using semi-supervised techniques like tri-training (Lewis et al., 2016; Lee et al., 2016; Yoshikawa et al., 2017).

In some sense, all the neural models in Table 4 are implicitly trained in semi-supervised way because they use pretrained embeddings that are estimated on unlabelled data. The quality of ELMo embeddings is probably one of the reasons why our parser achieves such good results. However, another semi-supervised training method, namely tri-training, is particularly attractive because, unlike ELMo, it is trained on a CCG parsing objective which is more closely aligned to what we want to do. All tri-training models are trained on much larger dataset that in addition to CCGbank also includes 43 million word corpus automatically annotated with silver CCG derivations by Lewis et al. (2016). It is likely that incorporating tri-training into our training setup will further increase the improvement over other models.

6 Other relevant work

Recurrent Neural Network Grammar (RNNG) (Dyer et al., 2016) is a fully incremental top-down parsing model. Because it is top-down it has no issues with right branching structures, but right adjuncts would still make parsing more difficult for RNNG because they will have to be predicted even earlier than in Left- and Right- branching derivations in CCG.

Left-corner parsers (which can be seen as a more constrained version of CCG Left-branching parsing strategy) seem more psychologically realistic than top-down parsers (Abney and Johnson, 1991; Resnik, 1992; Stanojević and Stabler, 2018). Some proposals about handling right adjunction in left-corner parsing are based on extension to generalized left-corner parsers (Demers, 1977; Hale, 2014) that can force some grammar rules (in particular right-adjunction rules) to be less incremental. Our approach does not decrease incrementality of the parser in this way. On the contrary, having a special mechanism for right adjunction makes parser both more incremental and more accurate.

Revealing based on higher order unification by Pareschi and Steedman (1987) was also proposed by Steedman (1990) as the basis for CCG explanation of gapping. The present derivation-based mechanism for revealing does not extend to gapping, and is targeting to model only derivations that could be explained with a standard CCG grammar derived from CCGbank. While that guarantees that we stay in the safe zone of sound and complete "standard" CCG derivations, it would be good as a future work to extend support for gapping and other types of derivations not present in CCGbank.

Niv (1993, 1994) proposed an alternative to the unification-based account of Pareschi and Steedman similar to our proposal for online tree rotation. Niv's parser is mostly a formal treatment of left-to-right rotations evaluated against psycholinguistic garden paths, but lacks the wide coverage implementation and statistical parsing model as a basis for resolving attachment ambiguities.

7 Conclusion

We have presented a revealing-based incremental parsing algorithm that has special transitions for handling right-adjunction. The parser is neutral with regard to the particular semantic representation used. It is computationally efficient, and can reveal all possible constituents types. It is the most incremental CCG parser yet proposed, and has state-of-the-art results against all published parsers trained on the CCGbank under both dependency recovery measures that are in use for the purpose.

Acknowledgments

This work was supported by ERC H2020 Advanced Fellowship GA 742137 SEMANTAX grant.

References

Steven P. Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20:233–249.

G M Adelson-Velskii and E M Landis. 1962. An algorithm for the organization of information. *Soviet Mathematics Doklady*, 3(2):263–266.

Anthony Ades and Mark Steedman. 1982. On the order of words. *Linguistics and Philosophy*, 4:517–558.

Bharat Ram Ambati, Tejaswini Deoskar, Mark Johnson, and Mark Steedman. 2015. An Incremental Algorithm for Transition-based CCG Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 53–63. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Caroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *First International Conference on language resources & evaluation: Granada, Spain, 28-30 May 1998*, pages 447–456. European Language Resources Association.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 327–334. Association for Computational Linguistics.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition*, 3rd edition. The MIT Press.

Alan J. Demers. 1977. Generalized left corner parsing. In *4th Annual ACM Symposium on Principles of Programming Languages*, pages 170–181.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 79–86. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg. Springer-Verlag.

Leo J. Guibas and Robert Sedgewick. 1978. A dichromatic framework for balanced trees. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, pages 8–21, Washington, DC, USA. IEEE Computer Society.

John T. Hale. 2014. *Automaton Theories of Human Sentence Comprehension*. CSLI, Stanford.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form Parsing for Combinatory Categorial Grammars with Generalized Composition and Type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 465–473, Stroudsburg, PA, USA. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global Neural CCG Parsing with Optimality Guarantees. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2366–2376. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000.

William Marslen-Wilson. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–523.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Michael Niv. 1993. *A Computational Model of Syntactic Processing: Ambiguity Resolution from Interpretation*. Ph.D. thesis, University of Pennsylvania. IRCS Report 93-27.

Michael Niv. 1994. A psycholinguistically motivated parser for CCG. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 125–132. Association for Computational Linguistics.

Chris Okasaki. 1999. Red-black trees in a functional setting. *Journal of functional programming*, 9(4):471–477.

Remo Pareschi and Mark Steedman. 1987. A Lazy Way to Chart-parse with Categorial Grammars. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, ACL '87, pages 81–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *Proceedings of the 14th International Conference on Computational Linguistics, COLING 92*, pages 191–197.

Miloš Stanojević and Edward Stabler. 2018. A sound and complete left-corner parsing for minimalist grammars. In *Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 65–74. Association for Computational Linguistics.

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and philosophy*, 13(2):207–263.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging With LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Wenhui Wang and Baobao Chang. 2016. Graph-based Dependency Parsing with Bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315. Association for Computational Linguistics.

William Woods. 1973. An experimental parsing system for Transition Network Grammars. In Randall Rustin, editor, *Natural Language Processing*, pages 111–154. Algorithmics Press, New York.

Wenduan Xu. 2016. LSTM shift-reduce CCG parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1764.

Wenduan Xu, Michael Auli, and Stephen Christopher Clark. 2016. Expected f-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-reduce CCG parsing with a dependency model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 218–227.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* ccg parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287. Association for Computational Linguistics.

M. Zaheer, S. Kottur, M. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. 2017. Deep sets. In *NIPS*. (Accepted for oral presentation, 1.23

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 683–692. Association for Computational Linguistics.

Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing

Hao Fu[1†], **Chunyuan Li**[2†*], **Xiaodong Liu**[2]
Jianfeng Gao[2], **Asli Celikyilmaz**[2], **Lawrence Carin**[1]
[1]Duke University [2]Microsoft Research, Redmond
{chunyl, xiaodl, jfgao, aslicel}@microsoft.com
{hao.fu, lcarin}@duke.edu

Abstract

Variational autoencoders (VAEs) with an auto-regressive decoder have been applied for many natural language processing (NLP) tasks. The VAE objective consists of two terms, (i) reconstruction and (ii) KL regularization, balanced by a weighting hyper-parameter β. One notorious training difficulty is that the KL term tends to vanish. In this paper we study scheduling schemes for β, and show that KL vanishing is caused by the lack of good latent codes in training the decoder at the beginning of optimization. To remedy this, we propose a cyclical annealing schedule, which repeats the process of increasing β multiple times. This new procedure allows the progressive learning of more meaningful latent codes, by leveraging the informative representations of previous cycles as warm re-starts. The effectiveness of cyclical annealing is validated on a broad range of NLP tasks, including language modeling, dialog response generation and unsupervised language pre-training.

1 Introduction

Variational autoencoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) have been applied in many NLP tasks, including language modeling (Bowman et al., 2015; Miao et al., 2016), dialog response generation (Zhao et al., 2017; Wen et al., 2017), semi-supervised text classification (Xu et al., 2017), controllable text generation (Hu et al., 2017), and text compression (Miao and Blunsom, 2016). A prominent component of a VAE is the distribution-based latent representation for text sequence observations. This flexible representation allows the VAE to explicitly model holistic properties of sentences, such as style, topic, and high-level linguistic and semantic features. Samples from the prior latent distribution can produce diverse and well-formed sentences through simple deterministic decoding (Bowman et al., 2015).

Due to the sequential nature of text, an auto-regressive decoder is typically employed in the VAE. This is often implemented with a recurrent neural network (RNN); the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) RNN is used widely. This introduces one notorious issue when a VAE is trained using traditional methods: the decoder ignores the latent variable, yielding what is termed the *KL vanishing* problem.

Several attempts have been made to ameliorate this issue (Yang et al., 2017; Dieng et al., 2018; Zhao et al., 2017; Kim et al., 2018). Among them, perhaps the simplest solution is monotonic KL annealing, where the weight of the KL penalty term is scheduled to gradually increase during training (Bowman et al., 2015). While these techniques can effectively alleviate the KL-vanishing issue, a proper unified theoretical interpretation is still lacking, even for the simple annealing scheme.

In this paper, we analyze the variable dependency in a VAE, and point out that the auto-regressive decoder has two paths (formally defined in Section 3.1) that work together to generate text sequences. One path is conditioned on the latent codes, and the other path is conditioned on previously generated words. KL vanishing happens because (i) the first path can easily get blocked, due to the lack of good latent codes at the beginning of decoder training; (ii) the easiest solution that an expressive decoder can learn is to ignore the latent code, and relies on the other path only for decoding. To remedy this issue, a promising approach is to remove the blockage in the first path, and feed meaningful latent codes in training the decoder, so that the decoder can easily adopt them to generate controllable observations (Bowman et al., 2015).

This paper makes the following contributions: (i) We provide a novel explanation for the KL-

---

[*] Corresponding author [†] Equal Contribution

Proceedings of NAACL-HLT 2019, pages 240–250
Minneapolis, Minnesota, June 2 - June 7, 2019.

vanishing issue, and develop an understanding of the strengths and weaknesses of existing scheduling methods (*e.g.*, constant or monotonic annealing schedules). (*ii*) Based on our explanation, we propose a cyclical annealing schedule. It repeats the annealing process multiple times, and can be considered as an inexpensive approach to leveraging good latent codes learned in the previous cycle, as a warm restart, to train the decoder in the next cycle. (*iii*) We demonstrate that the proposed cyclical annealing schedule for VAE training improves performance on a large range of tasks (with negligible extra computational cost), including text modeling, dialog response generation, and unsupervised language pre-training.

2 Preliminaries

2.1 The VAE model

To generate a text sequence of length T, $\boldsymbol{x} = [x_1, \cdots, x_T]$, neural language models (Mikolov et al., 2010) generate every token x_t conditioned on the previously generated tokens:

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p(x_t | x_{<t}),$$

where $x_{<t}$ indicates all tokens before t.

The VAE model for text consists of two parts, generation and inference (Kingma and Welling, 2013; Rezende et al., 2014; Bowman et al., 2015). The *generative model* (*decoder*) draws a continuous latent vector $\boldsymbol{z}$ from prior $p(\boldsymbol{z})$, and generates the text sequence $\boldsymbol{x}$ from a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$; $p(\boldsymbol{z})$ is typically assumed a multivariate Gaussian, and $\boldsymbol{\theta}$ represents the neural network parameters. The following auto-regressive decoding process is usually used:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t | x_{<t}, \boldsymbol{z}). \tag{1}$$

Parameters $\boldsymbol{\theta}$ are typically learned by maximizing the marginal log likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$. However, this marginal term is intractable to compute for many decoder choices. Thus, variational inference is considered, and the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \propto p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ is approximated via the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is (often known as the *inference model* or *encoder*), implemented via a $\boldsymbol{\phi}$-parameterized neural network. It yields the *evidence lower bound* (ELBO) as an objective:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_{\text{ELBO}} = \\ \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \big] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z})) \end{aligned} \tag{2}$$

Typically, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is modeled as a Gaussian distribution, and the re-parametrization trick is used for efficient learning (Kingma and Welling, 2013).

2.2 Training Schedules and KL Vanishing

There is an alternative interpretation of the ELBO: the VAE objective can be viewed as a regularized version of the autoencoder (AE) (Goodfellow et al., 2016). It is thus natural to extend the negative of $\mathcal{L}_{\text{ELBO}}$ in (2) by introducing a hyperparameter β to control the strength of regularization:

$$\mathcal{L}_{\beta} = \mathcal{L}_E + \beta \mathcal{L}_R, \quad \text{with} \tag{3}$$

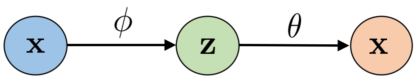
$$\mathcal{L}_E = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \big] \tag{4}$$

$$\mathcal{L}_R = \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z})) \tag{5}$$

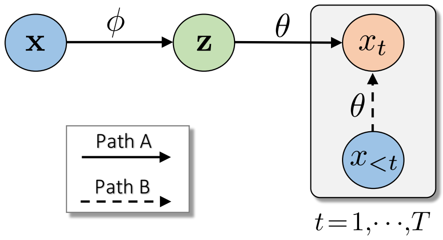
where $\mathcal{L}_E$ is the reconstruction error (or negative log-likelihood (NLL)), and $\mathcal{L}_R$ is a KL regularizer.

The cost function $\mathcal{L}_{\beta}$ provides a unified perspective for understanding various autoencoder variants and training methods. When $\beta = 1$, we recover the VAE in (2). When $\beta = 0$, and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is a delta distribution, we recover the AE. In other words, the AE does not regularize the variational distribution toward a prior distribution, and there is only a point-estimate to represent the text sequence's latent feature. In practice, it has been found that learning with an AE is prone to overfitting (Bowman et al., 2015), or generating plain dialog responses (Zhao et al., 2017). Hence, it is desirable to retain meaningful posteriors in real applications. Two different schedules for β have been commonly used for a text VAE.

Constant Schedule The standard approach is to keep $\beta = 1$ fixed during the entire training procedure, as it corresponds to optimizing the true VAE objective. Unfortunately, instability on text analysis has been witnessed, in that the KL term $\mathcal{L}_R$ becomes vanishingly small during training (Bowman et al., 2015). This issue causes two undesirable outcomes: (*i*) an encoder that produces posteriors almost identical to the Gaussian prior, for all observations (rather than a more interesting posterior); and (*ii*) a decoder that completely ignores the latent variable $\boldsymbol{z}$, and a learned model that reduces to a simpler language model. This is known as the *KL vanishing* issue in text VAEs.



(a) Traditional VAE



(b) VAE with an auto-regressive decoder

Figure 1: Illustration of learning parameters $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ in the two different paradigms. Starting from the observation $\boldsymbol{x}$ in blue circle, a VAE infers its latent code $\boldsymbol{z}$ in the green circle, and further generates its reconstruction in the red circle. (a) Standard VAE learning, with only one path via $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ from $\boldsymbol{x}$ to its reconstruction; (b) VAE learning with an auto-regressive decoder. Two paths are considered from $\boldsymbol{x}$ to its reconstruction: Path A via $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ and Path B via $\boldsymbol{\theta}$.

Monotonic Annealing Schedule. A simple remedy has been proposed in (Bowman et al., 2015) to alleviate KL collapse. It sets $\beta = 0$ at the beginning of training, and gradually increases β until $\beta = 1$ is reached. In this setting, we do not optimize the proper lower bound in (2) during the early stages of training, but nonetheless improvements on the value of that bound are observed at convergence in previous work (Bowman et al., 2015; Zhao et al., 2017).

The monotonic annealing schedule has become the *de facto* standard in training text VAEs, and has been widely adopted in many NLP tasks. Though simple and often effective, this heuristic still lacks a proper justification. Further, how to best schedule β is largely unexplored.

3 Cyclical Annealing Schedule

3.1 Identifying Sources of KL Vanishing

In the traditional VAE (Kingma and Welling, 2013), $\boldsymbol{z}$ generates $\boldsymbol{x}$ directly, and the reconstruction depends only on one path of $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ passing through $\boldsymbol{z}$, as shown in Figure 1(a). Hence, $\boldsymbol{z}$ can largely determine the reconstructed $\boldsymbol{x}$. In contrast, when an auto-regressive decoder is used in a text VAE (Bowman et al., 2015), there are two paths from $\boldsymbol{x}$ to its reconstruction, as shown in Figure 1(b). Path A is the same as that in the standard VAE, where $\boldsymbol{z}$ is the global representation that controls the generation of $\boldsymbol{x}$; Path B leaks the partial ground-truth information of $\boldsymbol{x}$ at every time step of the sequential decoding. It generates x_t conditioned on $x_{<t}$. Therefore, Path B can potentially bypass Path A to generate $\boldsymbol{x}$, leading to KL vanishing.

From this perspective, we hypothesize that the model-collapse problem is related to the low quality of $\boldsymbol{z}$ at the beginning phase of decoder training. A lower quality $\boldsymbol{z}$ introduces more difficulties in reconstructing $\boldsymbol{x}$ via Path A. As a result, the model is forced to learn an easier solution to decoding: generating $\boldsymbol{x}$ via Path B only.

We argue that this phenomenon can be easily observed due to the powerful representation capability of the auto-regressive decoder. It has been shown empirically that auto-regressive decoders are able to capture highly-complex distributions, such as natural language sentences (Mikolov et al., 2010). This means that Path B alone has enough capacity to model $\boldsymbol{x}$, even though the decoder takes $\{x_{<t}, \boldsymbol{z}\}$ as input to produce x_t. Zhang et al. (2017a) has shown that flexible deep neural networks can easily fit randomly labeled training data, and here the decoder can learn to rely solely on $x_{<t}$ for generation, when $\boldsymbol{z}$ is of low quality.

We use our hypothesis to explain the learning behavior of different scheduling schemes for β as follows.

Constant Schedule The two loss terms in (2) are weighted equally in the constant schedule. At the early stage of optimization, $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ are randomly initialized and the latent codes $\boldsymbol{z}$ are of low quality. The KL term $\mathcal{L}_R$ pushes $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ close to an uninformative prior $p(\boldsymbol{z})$: the posterior becomes more like an isotropic Gaussian noise, and less representative of their corresponding observations. In other words, $\mathcal{L}_R$ blocks Path A, and thus $\boldsymbol{z}$ remains uninformative during the entire training process: it starts with random initialization and then is regularized towards a random noise. Although the reconstruction term $\mathcal{L}_E$ can be satisfied via two paths, since $\boldsymbol{z}$ is noisy, the decoder learns to discard Path A (*i.e.*, ignores $\boldsymbol{z}$), and chooses Path B to generate the sentence word-by-word.

Monotonic Annealing Schedule The monotonic schedule sets β close to 0 in the early stage of training, which effectively removes the blockage $\mathcal{L}_R$ on Path A, and the model reduces to a denoising autoencoder [1]. $\mathcal{L}_E$ becomes the only objective,

---

[1] The Gaussian sampling remains for $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$

which can be reached by both paths. Though randomly initialized, $\boldsymbol{z}$ is learned to capture useful information for reconstruction of $\boldsymbol{x}$ during training. At the time when the full VAE objective is considered ($\beta = 1$), $\boldsymbol{z}$ learned earlier can be viewed as the VAE initialization; such latent variables are much more informative than random, and thus are ready for the decoder to use.

To mitigate the KL-vanishing issue, it is key to have meaningful latent codes $\boldsymbol{z}$ at the beginning of training the decoder, so that $\boldsymbol{z}$ can be utilized. The monotonic schedule under-weights the prior regularization, and the learned $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ tends to collapse into a point estimate (*i.e.,* the VAE reduces to an AE). This underestimate can result in sub-optimal decoder learning. A natural question concerns how one can get a better distribution estimate for $\boldsymbol{z}$ as initialization, while retaining low computational cost.

3.2 Cyclical Annealing Schedule

Our proposal is to use $\boldsymbol{z} \sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, which has been trained under the full VAE objective, as initialization. To learn to progressively improve latent representation $\boldsymbol{z}$, we propose a cyclic annealing schedule. We start with $\beta = 0$, increase β at a fast pace, and then stay at $\beta = 1$ for subsequent learning iterations. This encourages the model to converge towards the VAE objective, and infers its first raw full latent distribution.

Unfortunately, Path A is blocked at $\beta = 1$. The optimization is then continued at $\beta = 0$ again, which perturbs the VAE objective, dislodges it from the convergence, and reopens Path A. Importantly, the decoder is now trained with the latent code from a full distribution $\boldsymbol{z} \sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, and both paths are considered. We repeat this process several times to achieve better convergences.

Formally, β has the form:

$$\beta_t = \begin{cases} f(\tau), & \tau \leq R \\ 1, & \tau > R \end{cases} \quad \text{with} \tag{6}$$

$$\tau = \frac{\operatorname{mod}(t-1, \lceil T/M \rceil)}{T/M}\,, \tag{7}$$

where t is the iteration number, T is the total number of training iterations, f is a monotonically increasing function, and we introduce two new hyper-parameters associated with the cyclical annealing schedule:

- M: number of cycles (default $M = 4$);
- R: proportion used to increase β within a cycle (default $R = 0.5$).

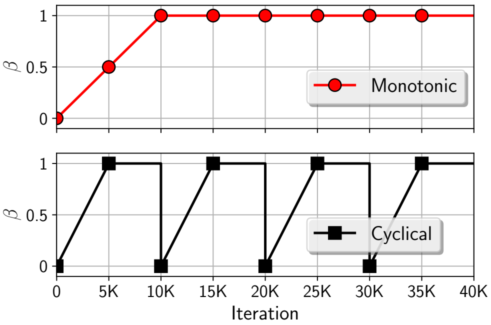


Figure 2: Comparison between (a) traditional monotonic and (b) proposed cyclical annealing schedules.In this figure, $M = 4$ cycles are illustrated, $R = 0.5$ is used for increasing within each cycle.

In other words, we split the training process into M cycles, each starting with $\beta = 0$ and ending with $\beta = 1$. We provide an example of a cyclical schedule in Figure 2(b), compared with the monotonic schedule in Figure 2(a). Within one cycle, there are two consecutive stages (divided by R):

- **Annealing**. β is annealed from 0 to 1 in the first $R\lceil T/M \rceil$ training steps over the course of a cycle. For example, the steps $[1, 5K]$ in the Figure 2(b). $\beta = f(0) = 0$ forces the model to learn representative $\boldsymbol{z}$ to reconstruct $\boldsymbol{x}$. As depicted in Figure 1(b), there is no interruption from the prior on Path A, $\boldsymbol{z}$ is forced to learn the global representation of $\boldsymbol{x}$. By gradually increasing β towards $f(R) = 1$, $q(\boldsymbol{z}|\boldsymbol{x})$ is regularized to transit from a point estimate to a distribution estimate, spreading out to match the prior.
- **Fixing**. As our ultimate goal is to learn a VAE model, we fix $\beta = 1$ for the rest of training steps within one cycle, *e.g.,* the steps $[5K, 10K]$ in Figure 2(b). This drives the model to optimize the full VAE objective until convergence.

As illustrated in Figure 2, the monotonic schedule increasingly anneals β from 0 to 1 once, and fixes $\beta = 1$ during the rest of training. The cyclical schedules alternatively repeats the annealing and fixing stages multiple times.

A Practical Recipe The existing schedules can be viewed as special cases of the proposed cyclical schedule. The cyclical schedule reduces to the constant schedule when $R = 0$, and it reduces to an monotonic schedule when $M = 1$ and R is relatively small [2]. In theory, any monotonically increasing function f can be adopted for the cyclical schedule, as long as $f(0) = 0$ and $f(R) = 1$. In practice, we suggest to build the cyclical schedule upon the success of monotonic schedules: we adopt the same f, and modify it by setting M and R (as default). Three widely used increasing functions for f are linear (Fraccaro et al., 2016; Goyal et al., 2017), Sigmoid (Bowman et al., 2015) and Consine (Lai et al., 2018). We present the comparative results using the linear function $f(\tau) = \tau/R$ in Figure 2, and show the complete comparison for other functions in Figure 7 of the Supplementary Material (SM).

3.3 On the impact of β

This section derives a bound for the training objective to rigorously study the impact of β; the proof details are included in SM. For notational convenience, we identify each data sample with a unique integer index $n \sim q(n)$, drawn from a uniform random variable on $\{1, 2, \cdots, N\}$. Further we define $q(\boldsymbol{z}|n) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_n)$ and $q(\boldsymbol{z}, n) = q(\boldsymbol{z}|n)q(n) = q(\boldsymbol{z}|n)\frac{1}{N}$. Following (Makhzani et al., 2016), we refer to $q(\boldsymbol{z}) = \sum_{n=1}^{N} q(\boldsymbol{z}|n)q(n)$ as the aggregated posterior. This marginal distribution captures the aggregated $\boldsymbol{z}$ over the entire dataset. The KL term in (5) can be decomposed into two refined terms (Chen et al., 2018; Hoffman and Johnson, 2016):

$$\begin{aligned}\mathcal{F}_R &= \mathbb{E}_{q(n)}[\mathrm{KL}(q(\boldsymbol{z}|n)||p(\boldsymbol{z}))] \\ &= \underbrace{I_q(\boldsymbol{z}, n)}_{\mathcal{F}_1\text{: Mutual Info.}} + \underbrace{\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))}_{\mathcal{F}_2\text{: Marginal KL}} \qquad (8)\end{aligned}$$

where $\mathcal{F}_1$ is the mutual information (MI) measured by q. Higher MI can lead to a higher correlation between the latent variable and data variable, and encourages a reduction in the degree of KL vanishing. The marginal KL is represented by $\mathcal{F}_2$, and it measures the fitness of the aggregated posterior to the prior distribution.

The reconstruction term in (5) provides a lower bound for MI measured by q, based on Corollary 3 in (Li et al., 2017):

$$\begin{aligned}\mathcal{F}_E &= E_{q(n),\boldsymbol{z}\sim q(\boldsymbol{z}|n)}(\log p(n|\boldsymbol{z}))] + H_q(n) \\ &\leq I_q(\boldsymbol{z}, n) \qquad (9)\end{aligned}$$

where $H(n)$ is a constant.

Analysis of β When scheduled with β, the training objective over the dataset can be written as:

$$\begin{aligned}\mathcal{F} &= -\mathcal{F}_E + \beta\mathcal{F}_R \qquad (10) \\ &\geq (\beta - 1)I_q(\boldsymbol{z}, n) + \beta\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z})) \qquad (11)\end{aligned}$$

To reduce KL vanishing, we desire an increase in the MI term $I(\boldsymbol{z}, n)$, which appears in both $\mathcal{F}_E$ and $\mathcal{F}_R$, modulated by β. It shows that reducing KL vanishing is inversely proportional with β. When $\beta = 0$, the model fully focuses on maximizing the MI. As β increases, the model gradually transits towards fitting the aggregated latent codes to the given prior. When $\beta = 1$, the implementation of MI becomes implicit in $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))$. It is determined by the amortized inference regularization (implied by the encoder's expressivity) (Shu et al., 2018), which further affects the performance of the generative density estimator.

4 Visualization of Latent Space

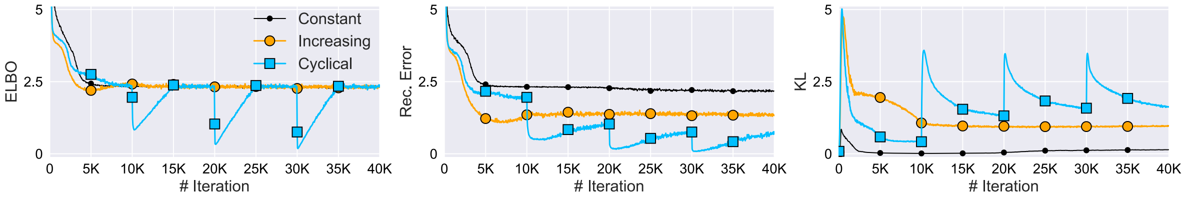
We compare different schedule methods by visualizing the learning processes on an illustrative problem. Consider a dataset consisting of 10 sequences, each of which is a 10-dimensional one-hot vector with the value 1 appearing in different positions. A 2-dimensional latent space is used for the convenience of visualization. Both the encoder and decoder are implemented using a 2-layer LSTM with 64 hidden units each. We use $T = 40$K total iterations, and the scheduling schemes in Figure 2.

The learning curves for the ELBO, reconstruction error, and KL term are shown in Figure 3. The three schedules share very similar values. However, the cyclical schedule provides substantially lower reconstruction error and higher KL divergence. Interestingly, the cyclical schedule improves the performance progressively: it becomes better than the previous cycle, and there are clear periodic patterns across different cycles. This suggests that the cyclical schedule allows the model to use the previously learned results as a warm-restart to achieve further improvement.

We visualize the resulting division of the latent space for different training steps in Figure 4, where each color corresponds to $\boldsymbol{z} \sim q(\boldsymbol{z}|n)$, for $n = 1, \cdots, 10$. We observe that the constant schedule produces heavily mixed latent codes $\boldsymbol{z}$ for different sequences throughout the entire training process. The monotonic schedule starts with a mixed $\boldsymbol{z}$, but soon divides the space into a mixture

---

[2] In practice, the monotonic schedule usually anneals in a very fast pace, thus R is small compared with the entire training procedure.



(a) ELBO (b) Reconstruction Error (c) KL term

Figure 3: Comparison of the learning curves for the three schedules on an illustrative problem.

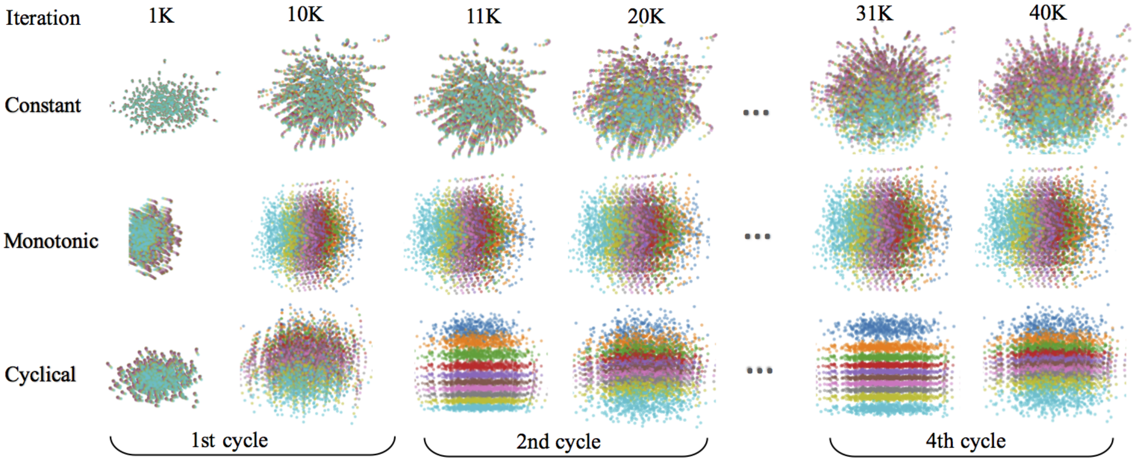


Figure 4: Visualization of the latent space along the learning dynamics on an illustrative problem.

of 10 cluttered Gaussians in the annealing process (the division remains cluttered in the rest of training). The cyclical schedule behaves similarly to the monotonic schedule in the first 10K steps (the first cycle). But, starting from the 2nd cycle, much more divided clusters are shown when learning on top of the 1st cycle results. However, $\beta < 1$ leads to some holes between different clusters, making $q(\boldsymbol{z})$ violate the constraint of $p(\boldsymbol{z})$. This is alleviated at the end of the 2nd cycle, as the model is trained with $\beta = 1$. As the process repeats, we see clearer patterns in the 4th cycle than the 2nd cycle for both $\beta < 0$ and $\beta = 1$. It shows that more structured information is captured in $\boldsymbol{z}$ using the cyclical schedule, which is beneficial in downstream applications as shown in the experiments.

5 Related Work

Solutions to KL vanishing Several techniques have been proposed to mitigate the KL vanishing issue. The proposed method is most closely related to the monotonic KL annealing technique in (Bowman et al., 2015). In addition to introducing a specific algorithm, we have comprehensively studied the impact of β and its scheduling schemes. Our explanations can be used to interpret other techniques, which can be broadly categorized into two classes.

The first category attempts to weaken Path B, and force the decoder to use Path A. Word drop decoding (Bowman et al., 2015) sets a certain percentage of the target words to zero. It has shown that it may degrade the performance when the drop rate is too high. The dilated CNN was considered in (Yang et al., 2017) as a new type of decoder to replace the LSTM. By changing the decoder's dilation architecture, one can control Path B: the effective context from $x_{<t}$.

The second category of techniques improves the dependency in Path A, so that the decoder uses latent codes more easily. Skip connections were developed in (Dieng et al., 2018) to shorten the paths from $\boldsymbol{z}$ to $\boldsymbol{x}$ in the decoder. Zhao et al. (2017) introduced an auxiliary loss that requires the decoder to predict the bag-of-words in the dialog response (Zhao et al., 2017). The decoder is thus forced to capture global information about the target response. Zhao et al. (2019) enhanced Path A via mutual information. Concurrent with our work, He et al. (2019) proposed to update encoder multiple times to achieve better latent code before updating decoder. Semi-amortized training (Kim et al., 2018) was proposed to perform stochastic variational inference (SVI) (Hoffman et al., 2013) on top of the amortized inference in VAE. It shares a similar motivation with the proposed approach, in that better latent codes can reduce KL vanishing. However, the computational cost to run SVI is high, while our monotonic schedule does not require any additional compute overhead. The KL scheduling methods are complementary to these techniques. As shown in experiments, the proposed cyclical schedule can further improve them.

β-VAE The VAE has been extended to β-regularized versions in a growing body of work (Higgins et al., 2017; Alemi et al., 2018). Perhaps the seminal work is β-VAE (Higgins et al., 2017), which was extended in (Kim and Mnih, 2018; Chen et al., 2018) to consider β on the refined terms in the KL decomposition. Their primary goal is to learn disentangled latent representations to explain the data, by setting $\beta > 1$. From an information-theoretic point of view, (Alemi et al., 2018) suggests a simple method to set $\beta < 1$ to ensure that latent-variable models with powerful stochastic decoders do not ignore their latent code. However, $\beta \neq 1$ results in an improper statistical model. Further, β is static in their work; we consider dynamically scheduled β and find it more effective.

Cyclical schedules Warm-restart techniques are common in optimization to deal with multimodal functions. The cyclical schedule has been used to train deep neural networks (Smith, 2017), warm restart stochastic gradient descent (Loshchilov and Hutter, 2017), improve convergence rates (Smith and Topin, 2017), obtain model ensembles (Huang et al., 2017) and explore multimodal distributions in MCMC sampling (Zhang et al., 2019). All these works applied cyclical schedules to the learning rate. In contrast, this paper represents the first to consider the cyclical schedule for β in VAE. Though the techniques seem simple and similar, our motivation is different: we use the cyclical schedule to re-open Path A in Figure 1(b) and provide the opportunity to train the decoder with high-quality $\boldsymbol{z}$.

6 Experiments

The source code to reproduce the experimental results will be made publicly available on GitHub[3]. For a fair comparison, we follow the practical recipe described in Section 3.2, where the monotonic schedule is treated as a special case of cyclical schedule (while keeping all other settings the same). The default hyper-parameters of the cyclical schedule are used in all cases unless stated otherwise. We study the impact of hyper-parameters in the SM, and show that larger M can provide higher performance for various R. We show the major results in this section, and put more details in the SM. The monotonic and cyclical schedules are denoted as **M** and **C**, respectively.

Schedule		**Rec**	**KL**	**ELBO**	**PPL**
VAE	**M**	101.73	0.907	-102.63	108.09
	C	**100.51**	**1.955**	-102.46	107.25
SA-VAE	**M***	100.75	1.796	-102.54	107.64
	M	101.83	1.053	-102.89	109.33
	C	**100.50**	**2.261**	-102.76	108.71
$\beta_m = 0.5$	**M**	97.36	9.605	-106.96	132.57
	C	95.22	9.484	-104.70	118.78

Table 1: Comparison of language modeling on Penn Tree Bank (PTB). The last iteration results are reported. Since SA-VAE tends to overfit, we report its best results in row M*.

6.1 Language Modeling

We first consider language modeling on the Penn Tree Bank (PTB) dataset (Marcus et al., 1993). Language modeling with VAEs has been a challenging problem, and few approaches have been shown to produce rich generative models that do not collapse to standard language models. Ideally a deep generative model trained with variational inference would pursue higher ELBO, making use of the latent space (*i.e.,* maintain a nonzero KL term) while accurately modeling the underlying distribution (*i.e.,* lower reconstruction errors). We implemented different schedules based on the code[4] published by Kim et al. (2018).

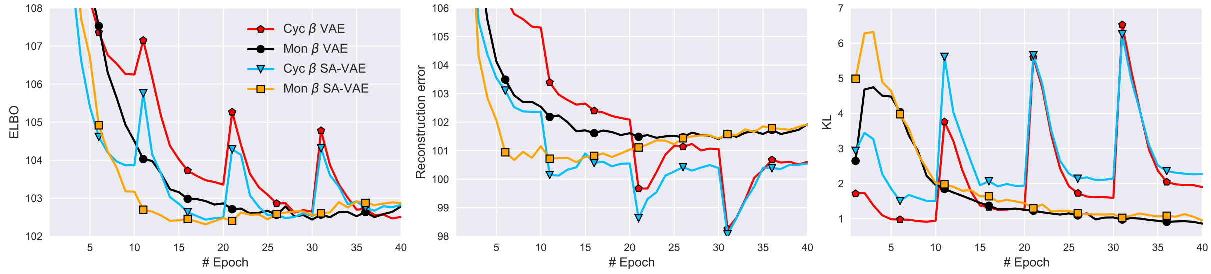
The latent variable is 32-dimensional, and 40 epochs are used. We compare the proposed cyclical annealing schedule with the monotonic schedule baseline that, following (Bowman et al., 2015), anneals linearly from 0 to 1.0 over 10 epochs. We also compare with semi-amortized (SA) training (Kim et al., 2018), which is considered as the state-of-the-art technique in preventing KL vanishing. We set SVI steps to 10.

Results are shown in Table 1. The perplexity is reported in column PPL. The cyclical schedule outperforms the monotonic schedule for both standard VAE and SA-VAE training. SA-VAE training

---

[3] `https://github.com/haofuml/cyclical_annealing`

[4] `https://github.com/harvardnlp/sa-vae`



(a) ELBO (b) Reconstruction error (c) KL

Figure 5: Learning curves of VAE and SA-VAE on PTB. Under similar ELBO, the cyclical schedule provides lower reconstruction errors and higher KL values than the monotonic schedule.

Context `Alice`: yeah you know its interesting especially when my experience has always been at a public university.			
Topic: Choose a College	**Target** `Bob` (statement): yeah that's right		
C	1. yes 2. oh really 3. and there's a lot of <unk> there's a lot of people 4. yeah 5. and i think that's probably the biggest problem i've ever seen in the past	**M**	1. i'm not sure 2. and i'm not sure 3. and i'm not sure 4. i'm not sure 5. i'm not sure

Table 2: Generated dialog responses from the cyclical and monotonic schedules.

can effectively reduce KL vanishing, it takes 472s per epoch. However, this is significantly more expensive than the standard VAE training which takes 30s per epoch. The proposed cyclical schedule adds almost zero cost.

We show the learning curves for VAE and SA-VAE in Figure 5. Interestingly, the cyclical schedule exhibits periodical learning behaviours. The performance of the cyclical schedule gets better progressively, after each cycle. While ELBO and PPL ar similar, the cyclical schedule improves the reconstruction ability and KL values for both VAE and SA-VAE. We observe clear over-fitting issues for the SA-VAE with the monotonic schedule, while this issue is less severe for SA-VAE with the cyclical schedule.

Finally, we further investigate whether our improvements are from simply having a lower β, rather than from the cyclical schedule re-opening Path A for better learning. To test this, we use a monotonic schedule with maximum $\beta = 0.5$. We observe that the reconstruction and KL terms perform better individually, but the ELBO is substantially worse than $\beta = 1$, because $\beta = 0.5$ yields an improper model. Even so, the cyclical schedule improves its performance.

6.2 Conditional VAE for Dialog

Model	CVAE		CVAE+BoW	
Schedule	**M**	**C**	**M**	**C**
Rec-P ↓	36.16	**29.77**	18.44	**16.74**
KL Loss ↑	0.265	**4.104**	14.06	**15.55**
B4 prec	0.185	**0.234**	0.211	**0.219**
B4 recall	0.122	**0.220**	0.210	**0.219**
A-bow prec	0.957	**0.961**	0.958	**0.961**
A-bow recall	0.911	**0.941**	0.938	**0.940**
E-bow prec	0.867	0.833	0.830	0.828
E-bow recall	0.784	**0.808**	0.808	0.805

Table 3: Comparison on dialog response generation. Reconstruction perplexity (Rec-P) and BLEU (B) scores are used for evaluation.

We use a cyclical schedule to improve the latent codes in (Zhao et al., 2017), which are key to diverse dialog-response generation. Following (Zhao et al., 2017), Switchboard (SW) Corpus (Godfrey and Holliman, 1997) is used, which has 2400 two-sided telephone conversations.

Two latent variable models are considered. The first one is the Conditional VAE (CVAE), which has been shown better than the encoder-decoder neural dialog (Serban et al., 2016). The second is to augment VAE with a bag-of-word (BoW) loss to tackle the KL vanishing problem, as proposed in (Zhao et al., 2017).

Table 2 shows the sample outputs generated from the two schedules using CVAE. Caller `Alice` begins with an open-ended statement on choosing a college, and the model learns to generate responses from Caller `Bob`. The cyclical schedule generated highly diverse answers that cover multiple plausible dialog acts. On the contrary, the responses from the monotonic schedule are limited to repeat plain responses, *i.e.,* "*i'm not sure*".

Quantitative results are shown in Table 3, using the evaluation metrics from (Zhao et al., 2017). (i) Smoothed Sentence-level BLEU (Chen and Cherry, 2014): BLEU is a popular metric that measures the geometric mean of modified n-gram precision with a length penalty. We use BLEU-1 to 4 as our lexical similarity metric and normalize the score to 0 to 1 scale. (ii) Cosine Distance of Bag-of-word Embedding (Liu et al., 2016): a simple method to obtain sentence embeddings is to take the average or extreme of all the word embeddings in the sentences. We used Glove embedding and denote the average method as $A-bow$ and extreme method as $E-bow$. The score is normalized to $[0, 1]$. Higher values indicate more plausible responses.

The BoW indeed reduces the KL vanishing issue, as indicated by the increased KL and decreased reconstruction perplexity. When applying the proposed cyclical schedule to CVAE, we also see a reduced KL vanishing issue. Interestingly, it also yields the highest BLEU scores. This suggests that the cyclical schedule can generate dialog responses of higher fidelity with lower cost, as the auxiliary BoW loss is not necessary. Further, BoW can be improved when integrated with the cyclical schedule, as shown in the last column of Table 3.

6.3 Unsupervised Language Pre-training

We consider the Yelp dataset, as pre-processed in (Shen et al., 2017) for unsupervised language pre-training. Text features are extracted as the latent codes $\boldsymbol{z}$ of VAE models, pre-trained with monotonic and cyclical schedules. The AE is used as the baseline. A good VAE can learn to cluster data into meaningful groups (Kingma and Welling, 2013), indicating that well-structured $\boldsymbol{z}$ are highly informative features, which usually leads to higher classification performance. To clearly compare the quality of $\boldsymbol{z}$, we build a simple one-layer classifier on $\boldsymbol{z}$, and fine-tune the model on different proportions of labelled data (Zhang et al., 2017b).

The results are shown in Figure 6. The cyclical schedule consistently yields the highest accuracy relative to other methods. We visualize the tSNE embeddings (Maaten and Hinton, 2008) of $\boldsymbol{z}$ in Figure 9 of the SM, and observe that the cyclical schedule exhibits clearer clustered patterns.

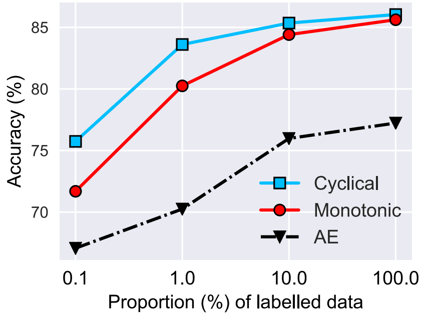


Figure 6: Accuracy of fine-tuning on the unsupervised pre-trained models on the Yelp dataset.

Schedule	Rec	KL	ELBO
Cyc β + Const η	**101.30**	**1.457**	-102.76
Mon β + Const η	101.93	0.858	-102.78
Cyc β + Cyc η	**100.61**	**1.897**	-102.51
Mon β + Cyc η	101.74	0.748	-102.49

Table 4: Comparison of cyclical schedules on β and η, tested with language modeling on PTB.

6.4 Ablation Study

To enhance the performance, we propose to apply the cyclical schedule to the learning rate η on real tasks. It ensures that the optimizer has the same length of optimization trajectory for each β cycle (so that each cycle can fully converge). To investigate the impact of cyclical on η, we perform two more ablation experiments: (i) We make only β cyclical, keep η constant. (ii) We make only η cyclical, keep β monotonic. The last epoch numbers are shown in Table 4, and the learning curves on shown in Figure 10 in SM. Compared with the baseline, we see that it is the cyclical β rather than cyclical η that contributes to the improved performance.

7 Conclusions

We provide a novel two-path interpretation to explain the KL vanishing issue, and identify its source as a lack of good latent codes at the beginning of decoder training. This provides an understanding of various β scheduling schemes, and motivates the proposed cyclical schedule. By reopening the path at $\beta = 0$, the cyclical schedule can progressively improve the performance, by leveraging good latent codes learned in the previous cycles as warm re-starts. We demonstrate the effectiveness of the proposed approach on three NLP tasks, and show that it is superior to or complementary to other techniques.

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a broken ELBO. In *ICML*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. 2018. Isolating sources of disentanglement in VAEs. *NIPS*.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*.

J Godfrey and E Holliman. 1997. Switchboard-1 release 2: Linguistic data consortium. *SWITCHBOARD: A User's Manual*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *NIPS*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *ICLR*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*.

Matthew D Hoffman and Matthew J Johnson. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *ICML*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *ICLR*.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. *ICML*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-amortized variational autoencoders. *ICML*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *ICLR*.

Guokun Lai, Bohan Li, Guoqing Zheng, and Yiming Yang. 2018. Stochastic wavenet: A generative latent variable model for sequential data. *ICML workshop*.

Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. 2017. ALICE: Towards understanding adversarial learning for joint distribution matching. In *NIPS*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. *ICLR workshop*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *EMNLP*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *ICML*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. 2018. Amortized inference regularization. *NIPS*.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *WACV*. IEEE.

Leslie N Smith and Nicholay Topin. 2017. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint arXiv:1708.07120*.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. *ICML*.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *AAAI*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017a. Understanding deep learning requires rethinking generalization. *ICLR*.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. 2019. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017b. Deconvolutional paragraph representation learning. In *NIPS*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. InfoVAE: Information maximizing variational autoencoders. *AAAI*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL*.

Recurrent Models and Lower Bounds for Projective Syntactic Decoding

Natalie Schluter
IT University of Copenhagen
Copenhagen, Denmark
natschluter@itu.dk

Abstract

The current state-of-the-art in neural graph-based parsing uses only approximate decoding at the training phase. In this paper aim to understand this result better. We show how recurrent models can carry out projective maximum spanning tree decoding. This result holds for both current state-of-the-art models for shift-reduce and graph-based parsers, projective or not. We also provide the first proof on the lower bounds of projective maximum spanning tree, DAG, and digraph decoding.

1 Introduction

For several years, the NLP field has seen widespread investigation into the application of Neural Networks to NLP tasks, and with this, much, rather inexplicable progress. A string of very recent work (for example, Chen et al. (2018); Weiss et al. (2018); Peng et al. (2018)), has attempted to delve into the formal properties of neural network topology choices, in attempts to both motivate, predict, and explain associated research in the field. This paper aims to further contribute along this line of research.

We present the results of our study into the ability of state-of-the-art first-order neural graph-based parsers, with seemingly simple architectures, to explicitly forego structured learning and prediction.[1] In particular, this is not due to a significantly faster, simpler, algorithm for projective maximum spanning tree (MST) decoding than Eisner (1996)'s algorithm, which we formally prove to be impossible, given the Exponential Time Hypothesis. But rather, this is due to the capacity of recurrent components of these architectures to implicitly discover a projective MST. We prove this formally by showing how these recurrent components can intrinsically simulate exact projective decoding.

The context. The current state-of-the-art for graph-based syntactic dependency parsing is a seemingly basic neural model by Dozat and Manning (2017). The parser's performance is an improvement on the first, even simpler, rather engineering-free, neural graph-based parser by Kiperwasser and Goldberg (2016). This latter parser updates with respect to an output structure: projective decoding over a matrix of arc scores coupled with hinge loss between predicted and gold arcs, reporting parser performance of, for example, 93.32% UAS and 91.2% LAS on the converted Penn Treebank.[2] Remarkably, the former parser by Dozat and Manning (2017) forgoes entirely any structural learning, employing simple cross-entropy at training time, and saving (unconstrained) maximum spanning tree decoding for test time.

We further optimised Kiperwasser and Goldberg (2016)'s parser (Varab and Schluter, 2018) and extended it for cross-entropy learning, as is done by Dozat and Manning (2017). At test time, instead of any explicit decoding algorithm over the arc score matrix, we simply take the maximum weighted incoming arc for each word; that is, the parser is highly streamlined, without any heavy neural network engineering, but now also without any structured learning, nor without any structural decoding at test time. The resulting neural parser still achieves an impressively competitive UAS of 92.61% evaluated on the converted Penn Treebank data, without recourse to any pre-trained embeddings, unlike the systems by Kiperwasser and Goldberg (2016) and Dozat and

---

[1] For the remainder of this paper, all decoding algorithms discussed are first-order.

[2] Training is on Sections 2-21, development on Section 22 and testing on Section 23), converted to dependency format following the default configuration of the Stanford Dependency Converter (version $\geq$ 3.5.2).

Proceedings of NAACL-HLT 2019, pages 251–260
Minneapolis, Minnesota, June 2 - June 7, 2019.

Manning (2017). Using GloVe 100-dimensional Wikipedia and Gigaword corpus (6 billion tokens) pretrained embeddings, without updates, but linearly projected through a single linear dense layer to the same dimension, the structure-less parser achieves 93.18% UAS.[3] With this paper, we shed light on these surprising results from seemingly simple architectures. The insights we present here apply to any neural architecture that first encodes input words of a sentence using some type of recurrent neural network–i.e., all current state-of-the-art graph-based or shift reduce neural parsers.

Our contributions. This paper presents results for understanding the surprisingly superior performance of structure-free learning and prediction in syntactic (tree) dependency parsing.

1. We provide a formal proof that there will never be an algorithm that carries out projective MST decoding in sub-cubic time, unless a widely believed assumption in computational complexity theory, the *Exponential Time Hypothesis* (ETH), is false.

   Hence, computationally, we provide convincing evidence that these neural parsing architectures cannot be as simple as they appear. These results are then extended to projective maximum spanning DAG and digraph decoding.
2. In particular, we then show how to simulate Eisner's algorithm using a single recurrent neural network. This shows how, in particular, the LSTM stacked architectures for graph-based parsing by Dozat and Manning (2017), Cheng et al. (2016), Hashimoto et al. (2017), Zhang et al. (2017), and Kiperwasser and Goldberg (2016), are capable of intrinsically decoding over arc scores.

   This therefore provides one practical application where RNNs do not need supplementary approximation considerations (Chen et al., 2018).

2 Preliminaries

The Exponential Time Hypothesis (ETH) and k-Clique. The Exponential Time Hypothesis is a widely held though unproven computational hardness assumption stating that 3-SAT (or any of the several related NP-complete problems) cannot be solved in sub-exponential time in the worst case (Impagliazzo and Paturi, 1999). According to ETH, if 3-SAT were solvable in sub-exponential time, then also $P = NP$. But the ETH assumption is stronger than the assumption that $P \neq NP$, so the converse is not necessarily true. ETH can be used to show that many computational problems are equivalent in complexity, in the sense that if one of them has a subexponential time algorithm then they all do.

The k-Clique problem is the parameterised version of the NP-hard Max-Clique problem. This canonical intractable problem in parameterised complexity asks, given an input graph, whether there exists a clique of size k. A naïve algorithm for this problem running in $O(n^k)$ time checks all n^k combinations of nodes and verifies each combination in $O(k^2)$ time to see if they form a clique. However, Chen et al. (2006) showed that the problem has no $n^{o(k)}$ time algorithm–that is, the problem has no algorithm that runs in time subexponential in the exponent k assuming ETH.[4].

Recurrent neural networks. Recurrent neural networks (Rumelhart et al., 1986), as we generally use them in practise in NLP, take as input a matrix $\mathbf{x}$, containing a sequence of n vectors $\mathbf{x} = \mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$, and apply the following set of equations recursively, with h_0 the initial state:

$$\mathbf{h_t} = g(\mathbf{b} + \mathbf{W}\mathbf{h_{(t-1)}} + \mathbf{U}\mathbf{x_t})$$

Here, g is the activation function. Typically this activation function is $\tanh$, however the computational power of the model is theoretically maintained with any so-called "squashing" function (Siegelmann, 1996).

The choice of g, on the other hand, has been shown to affect the power of the recurrent model in general, depending on the restrictions involved in the formal investigation. For the purposes of this paper, the activation function is a rectified linear unit, or `ReLU`. The general computational power of such RNNs has recently been formally explored by Chen et al. (2018) (given infinite precision) and Weiss et al. (2018) (given finite precision), and empirically investigated for practical considerations of convergence under training by Le et al. (2015).

---

[3] Our structure-less but optimised implementation of the (Kiperwasser and Goldberg, 2016) graph-based parser, with 100 dimensional generated word embeddings, 50 dimensional generated POS-tag embeddings, a stack of 3 BiLSTMs with an output dimension of 225 each (total 450 concatenated), no dropout, MLP mappings for arc nodes of 400 dimension, and for labels of 100 dimensions. We use DyNet 2.1 (Neubig et al., 2017), and the parser code is freely available at `https://github.com/natschluter/MaxDecodeParser`.

[4] The problem is said to be $W[1]$-complete (Flum and Grohe, 2006)

LSTMs (Hochreiter and Schmidhuber, 1997) are RNNs with weighted self-loops (so-called gates). The recurrence equations take the form:

$$
\begin{aligned}
\mathbf{f_t} &= g_1(\mathbf{b_f} + \mathbf{W_f}\mathbf{h_{(t-1)}} + \mathbf{U_f}\mathbf{x_t}) \\
\mathbf{i_t} &= g_1(\mathbf{b_i} + \mathbf{W_i}\mathbf{h_{(t-1)}} + \mathbf{U_i}\mathbf{x_t}) \\
\mathbf{o_t} &= g_1(\mathbf{b_o} + \mathbf{W_o}\mathbf{h_{(t-1)}} + \mathbf{U_o}\mathbf{x_t}) \\
\mathbf{c_t} &= f_t \circ c_{t-1} + i_t \circ g_1(\mathbf{b_c} + \mathbf{W_c}\mathbf{h_{(t-1)}} + \mathbf{U_c}\mathbf{x_t}) \\
\mathbf{h_t} &= \mathbf{o_t} \circ g_2(c_t)
\end{aligned}
$$

where g_1, g_2 are activation functions. Setting all $\mathbf{W_f}, \mathbf{W_i}, \mathbf{W_c}, \mathbf{U_f}, \mathbf{U_i}, \mathbf{U_c}$ to be zero matrices, b_f to be a 0 vector, b_i, b_c to be 1 vectors, and the activation function g_2 to be `ReLU` we see that, in terms of hidden states, the LSTM model includes that of the RNN. In this paper, all activation functions are `ReLU`s.

3 Related Work

State-of-the-art in neural syntactic dependency parsing. The graph-based neural architectures we refer to here have important commonalities. We focus our discussion on the key contributions by Kiperwasser and Goldberg (2016) (the simplest architecture, and the first), and by Dozat and Manning (2017) (the state-of-the-art).

The architectures can be partitioned into three general components:

1. **Word representation generation**: Both architectures generate word embeddings and POS-tag embeddings. Pretrained embeddings, if they are being used, are added to the trained embeddings, and concatenated to corresponding POS-tag embedding. The embeddings are sent through a stacked BiLSTM. Output embeddings are projected to two further vector representations: as head node or as dependent node (specialised representations).
2. **Arc scoring**: All (head, dependent) combinations are scored.
3. **Decoding**: By some decoding process, the arc score matrix yields a (possibly disconnected) graph representation of the input sentence: (n-1) arcs, where no word has more than one head, as well as their probabilities.

We show in this paper how the second and third components can be carried out implicitly within the BiLSTM layers of the first component. Since currently state-of-the-art shift-reduce parsers also encode input words of a sentence using some type of recurrent neural network, this insight also applies to these non-graph-based models.

Related computational hardness results. To date there is no known truly sub-cubic algorithm for Boolean Matrix Multiplication (BMM), nor for Context-Free Grammar (CFG) parsing. Adapting Satta (1994)'s lower bound proof for Tree Adjoining Grammar parsing, Lee (1997) proved that BMM can be reduced to finding a valid derivation a string of length $O(n^{\frac{1}{3}})$ with respect to a CFG of size $\Theta(n^2)$. Lee (1997)'s reduction shows that there can be a no $O(|G|n^{3-\epsilon})$ for some constant $\epsilon > 0$ (sub-cubic-time) algorithm for CFG-parsing without implying a significant breakthrough in BMM, which is widely believed not to be possible. However, the construction required the grammar size $|G| = \Theta(n^6)$ to be dependent on the the input size n, which, as Lee (1997) points out, is unrealistic in most applications.

Abboud et al. (2015), on the other hand, present a proof of the unlikelihood of a sub-cubic algorithm for CFG-parsing using ETH and specifically the k-Clique problem. Given an instance of the $3k$-Clique problem (i.e., an undirected graph and the parameter $3k$), they construct a string w of length n^k and a CFG, G of constant size (for any $3k$) such that if G derives w in sub-cubic time, then there is an algorithm running in time $n^{o(3k)}$ for the $3k$-Clique problem, which, as we explained in Section 2, is impossible, assuming ETH.

To date, no truly sub-cubic algorithm for projective maximum spanning tree decoding is known. In the next section, we present a proof similar in spirit to Abboud et al. (2015)'s that also shows that such an algorithm most likely cannot be found.

4 Lower Bounds for First-Order Projective Dependency Decoding

Current state-of-the-art neural graph-based parsers forego structural learning and do not even seem to require structured prediction. In this section, we provide evidence that this is indeed not because the parsers are so seemingly simple. Computationally it is unlikely that some simpler and faster decoding method alone is achieving such a competitive performance. We show this with the following theorem.

Theorem 1. *Under the assumption of ETH, there is no algorithm that carries out projective MST decoding in time significantly faster than* $O(n^3)$*; that is, there is no sub-cubic (*$O(n^{3-\epsilon})$ *for some constant* $\epsilon > 0$*) time algorithm for finding the maximally weighted projective spanning tree,* T^**, over a weighted digraph input.*

Notation and special remarks. We denote by $[n]$ the set $\{1, \ldots, n\}$. For lack of a better symbol, we use $\odot$ here to signify iterative string concatenation, which otherwise is signified by just writing symbols beside each other, or by the symbol $\cdot$. Rather than working over words of a sentence, given the formal nature of the proof, the projective MST algorithm must work over symbols of the input word w. Hence the input is a weighted digraph over the symbols of w and the output is a projective MST, T^*, over these symbols. The reduction, makes use of the weight of T^*.

Proof (of Theorem 1). Let $G = (V, E)$ be an arbitrary simple undirected graph. We place an arbitrary order on the nodes from V and fix it, so $V := \{v_1, \ldots, v_n\}$.

As in Abboud et al. (2015)'s reduction from $3k$-clique to CFG-parsing, we first generate a string of length $O(n^k)$ to represent the graph for the task at hand; we do so in $O(n^k)$ time. The string contains a representation of all of the possible k-cliques in the graph. We can create a listing of all of these k-cliques using exhaustive search in at most $O(n^k)$ time and space. Let $K := \{\{v_{i_1}, \ldots, v_{i_k}\}$ a k-clique in $G \mid i_j \in [n], v_{i_j} \in V\}$ correspond to the set of k-cliques from G, and place an arbitrary order on $K := \{k_1, \ldots, k_{|K|}\}$. So, $|K| \in O(n^k)$. We define $6 \cdot k \cdot |K|$ sets of symbols with respect to V, each with $n (= |V|)$ elements:

- **Unmarked symbols**: $A_{i,t} := \{a_{i,j,t} \mid j \in [n]\}$ for $i \in [k], t \in [|K|]$ where $a_{i,j,t}$ corresponds to node $v_j \in V$.
  Similarly for the sets $B_{i,t}$ and $C_{i,t}$.
- **Marked symbols**: $\overline{A}_{i,t} := \{\overline{a}_{i,j,t} \mid a_{i,j,t} \in A_i\}$. Similarly for the sets $\overline{B}_{i,t}$ and $\overline{C}_{i,t}$.

We let $A = \cup_{i \in [k], t \in [|K|]} (A_{i,t} \cup \overline{A}_{i,t})$ and similarly for B and C. Then the vocabulary for constructing our input word is $U := A \cup B \cup C$.

Constructing the input word w. We now construct a word w over the vocabulary U such that if the projective maximum spanning tree has weight $|w| + 2k - 2 + |K|$, then the graph G has a $3k$-clique. We do this by defining the weights of possible arcs between carefully selected pairs of symbols from the vocabulary. The entire construction of the word w takes time $O(n^k)$ (coinciding with the upper bound on the word's length).

The input word is made up of a series of gadgets. For each k-clique, we have three types of gadgets: A-, B-, and C-gadgets. A- and C-gadgets each correspond both to a particular k-clique in G, as well as all k-cliques in G. B-gadgets, on the other hand, only correspond to particular k-cliques in G. Let $k_t = \{v_{(t,1)}, \ldots, v_{(t,k)}\} \in K$ be the tth k-clique. Even if each $v_{(t,q)}$ is a node in $V(G)$ the notation for indices is useful to refer to the qth node of the tth k-clique. Also, in what follows, we use the middle index of symbols to simultaneously refer to the k-clique membership: $j_{(t,q)} \in [n]$, and simultaneously allows us to refer to the qth node in the tth k-clique from K, for $q \in [k], t \in [|K|]$.

A-gadgets:

$$\begin{aligned} A(t) :=& \odot_{i \in [k]} (\overline{a}_{i,j_{(t,1)},t} \overline{a}_{i,j_{(t,2)},t} \cdots \overline{a}_{i,j_{(t,k)},t}) \cdot \\ & \odot_{i \in [k]} (a_{i,j_{(t,1)},t} \cdot a_{i,j_{(t,2)},t} \cdots a_{i,j_{(t,k)},t}) \end{aligned}$$

C-gadgets:

$$\begin{aligned} C(t) :=& (\odot_{i \in [k]} c_{i,j_{(t,1)},t} c_{i,j_{(t,2)},t} \cdots c_{i,j_{(t,k)},t}) \\ & \cdot \overline{c}_{k,j_{(t,k)},t} \cdot \overline{c}_{k-1,j_{(t,k-1)},t} \cdots \overline{c}_{1,j_{(t,1)},t} \end{aligned}$$

and **B-gadgets:**

$$\begin{aligned} B(t) :=& L_t b_{k,j_{(t,k)},t} \cdots b_{2,j_{(t,2)},t} b_{1,j_{(t,1)},t} H_t \overline{b}_{k,j_{(t,k)},t} \\ & \cdot \overline{b}_{k-1,j_{(t,k-1)},t} \cdots \overline{b}_{2,j_{(t,2)},t} \overline{b}_{1,j_{(t,1)},t} R_t, \end{aligned}$$

We call the symbol H_t the *head of the gadget* $B(t)$, and L_t and R_t the gadget's *left and right boundary symbols* respectively.

We then set the word w to be

$$\big(\odot_{t \in [|K|]} A(t)\big) \cdot \big(\odot_{t \in [|K|]} B(t)\big) \cdot \big(\odot_{t \in [|K|]} C(t)\big)$$

consisting of an *A-gadget region* followed by a *B-gadget region*, and then a *C-gadget region*.

Idea of the proof. The idea of the proof is to allow an optimal projective MST, T^*, to be built that matches up one distinct (with respect to the k-clique) gadget from each region, each representing different k-cliques whenever there is a $3k$-clique in G. We will deduce the existence of such a clique by the weight of T^*. Essentially, a projective spanning tree of weight $|w| - 1$ will always be present, but T^* having weight superior to this will indicate a matching up of gadgets. Now, suppose we have a sub-cubic projective MST algorithm A. By our construction, if A returns a T^* with weight $|w| + 2k - 2 + |K|$, then there is a $3k$-clique. Otherwise, there is no $3k$-clique. On input of length n^k, the sub-cubic time algorithm runs in time $O((n^k)^{3-\epsilon}) = O(n^{3k-k\epsilon}) \in n^{o(3k)}$ for some constant $\epsilon > 0$. Thus A will have solved $3k$-clique in time $n^{o(3k)}$, which is impossible under the ETH assumption.

Note that by the definition of a $3k$-clique, a $3k$-clique can be partitioned arbitrarily into 3 equal sized sub-graphs over k nodes that must each form a k-clique. So, if $|K| < k$, then there trivially cannot be any $3k$-clique in G. We therefore only consider without loss of generality the argumentation for the case where $|K| \geq k$, since our algorithm can simply return a negative answer about the existence of a $3k$-clique in G after enumerating the set K and before computing any projective MST.

The projective MST algorithm takes as input the description of a weighted digraph, D, whose nodes are defined by symbols of the input word w. The digraph need not be explicitly constructed, since the algorithm can simply use the description of the digraph that follows instead to check for the existence of arcs between symbols. This description has constant length.

A description of the input weighted graph D over w. For the input digraph, arcs can (1) be missing from the fully complete digraph, (2) have weight 1, or (3) have weight 2. To construct D, weights are assigned to arcs by the following rules.

Weight 1 arcs. The following arcs of our input graph have weight 1.

1. **Region connectivity arcs.** These arcs ensure connectivity is possible within respective gadget regions.
    (a) All arcs $(\overline{a}_{1,j',t}, a_{i,j,t-1})$ and $(\overline{a}_{1,j',t}, \overline{a}_{i,j,t-1})$, i.e., the first symbol of the tth A-gadget attaches to all symbols of the previous ($t - 1$th) A-gadget.
    (b) All arcs $(\overline{c}_{1,j',t}, c_{i,j,t+1})$ and $(\overline{c}_{1,j',t}, \overline{c}_{i,j,t+1})$, i.e., the last symbol of the tth C-gadget attaches to all symbols of the next ($t + 1$th) C-gadget gadget.
    (c) All arcs $(b_{i,j,t}, b_{i+1,j',t})$ and $(\bar{b}_{i+1,j,t}, \bar{b}_{i,j',t})$ for $i \in [k-1]$.
    (d) All arcs $(b_{k,j,t}, L_t), (\bar{b}_{1,j',t}, R_t)$.
    (e) All arcs $(H_t, b_{1,j,t})$ and $(H_t, \bar{b}_{k,j',t})$ making H_t a possible head of the respective B-gadget (*B-gadget heads*) for any MST.
    (f) All arcs $(L_{t+1}, H_t), (R_t, H_{t+1})$ for all $t \in [|K| - 1]$.
    (g) All arcs $(c_{k,j_{(t,k)},t}, \overline{c}_{k,j,t})$, i.e., arcs from the last nonmarked symbol to the first marked symbol, in every C-gadget. Also, all arcs $(\overline{c}_{i+1,j,t}, \overline{c}_{i,j,t})$ for $i \in [k-1]$, i.e., together forming a path of marked symbols within each C-gadget.

        The following arcs are the reversals of (1c) through (1e).
    (h) All arcs $(b_{i+1,j,t}, b_{i,j',t})$ and $(\bar{b}_{i,j,t}, \bar{b}_{i+1,j',t})$ for $i \in [k-1]$.
    (i) All arcs $(L_t, b_{k,j,t}), (R_t, \bar{b}_{1,j',t})$.
    (j) All arcs $(b_{1,j,t}, H_t)$ and $(\bar{b}_{k,j',t}, H_t)$ making H_t the head of the respective B-gadget (*B-gadget heads*) for any MST.
2. **Boundary connectivity arcs.** These arcs ensure that the boundaries of regions are connected.
    (a) The arcs $(L_1, a_{i,j,|K|})$ and $(L_1, \overline{a}_{i,j,|K|})$, i.e., all symbols from the last of the A-gadgets attach to the first symbol of the B-gadget region.
    (b) The arcs $(R_{|K|}, c_{i,j,1})$ and $(R_{|K|}, \overline{c}_{i,j,1})$, i.e., all symbols from the first of the C-gadgets attach to the last symbol of the B-gadget region.
3. **G-induced arcs.** These arcs reflect the connections of the original graph G, and ultimately the existence of a $3k$-clique.
    (a) All arcs $(b_{i,j,t}, a_{i,j',t'})$, for each $i \in [k-1], t \neq t'$, if $v_j v_{j'} \in E(G)$ (i.e., not for $i = k$, which has a weight of 2 rather).
    (b) All arcs $(\bar{b}_{i,j,t}, c_{i,j',t'})$, for each $i \in \{2, \dots, k\}, t \neq t'$, if $v_j v_{j'} \in E(G)$ (i.e., not for $i = 1$, which has a weight of 2 rather).
    (c) All arcs $(\overline{c}_{i,j,t}, \overline{a}_{i,j',t'})$ for all $i \in [k], t \neq t'$, if $v_j v_{j'} \in E(G)$ (i.e., this time also for $i = 1$).

As we show in Lemma 1.1, with the region connectivity arcs (1a-1g) and boundary connectivity arcs (2), we ensure that the algorithm can always return a projective MST with weight at least $|w| - 1$. The G-induced arcs and region connectivity arcs (1h-1j, 4) on the other hand will be triggered to use by the algorithm's prioritisation of the following arcs.

Weight 2 arcs. We have the following arcs of weight 2.

4. **Region connectivity arcs.** (L_{t+1}, L_t) and (R_t, R_{t+1}) for $t \in [|K| - 1]$.
5. **G-induced arcs.**
    (a) All arcs $(b_{k,j,t}, a_{k,j',t'})$, for each $t \neq t'$, if $v_j v_{j'} \in E(G)$
    (b) All arcs $(\bar{b}_{1,j,t}, c_{1,j',t'})$, for each $t \neq t'$, if $v_j v_{j'} \in E(G)$

There are no other arcs in the input digraph D.

Lemma 1.1. *There always exists a projective MST in D of weight $|w| - 1$.*

Proof. The A-region, together with the symbol L_1 from the B-region can form a tree rooted in L_1

using region connectivity arcs (1a) with boundary connectivity arcs (2a)–all weight 1 arcs. Similarly for the C-region with the symbol $R_{|K|}$ from the B-region (arcs (1b) and (2b)). Moreover, these regional sub-trees are trivially projective. If we construct a projective subtree out of the B-region, in which L_1 and $R_{|K|}$ are leaf nodes, then we have the result.

The combination of weight-1 arcs from (1c), (1d), and (1e) results in each B-gadget B(t) being a projective subtree headed by its head node H_t made up of a combination of two paths $H_t, b_{1,j_1,t}, \ldots, b_{k,j_k,t}, L_t$ and $H_t, b_{k,j_k,t}, \ldots, \overline{b_{1,j_1,t}}, R_t$. To make a projective subtree out of the entire B-region, we choose some arbitrary H_t node as the root and take further weight-1 arcs described in (1f): (L_p, H_{p-1}) if $p \leq i$ and $(R_p, H_p + 1)$ otherwise, for $i \in [2, |K| - 1]$. In all these possible B-regional projective subtrees, both L_1 and $R_{|K|}$ are leaf nodes, which gives the result. □

Lemma 1.2. *Let T^* be a projective MST over D. There are at most $2k+(|K|-1)$ arcs of weight 2 in T^*: k from the B- to the A-region, k from the B- to the C-region, and the rest internal to the B-region.*

The number of arcs of weight 2, internal to or originating from the B-region, will be maximised if arcs exiting the B-region all originate from the same B-gadget (instead of 2+ distinct ones).

Moreover, suppose distinct $t_1, t_2, t_3 \in [|K|]$. If T^* includes an arc of weight 2 from gadget $B(t_2)$ to gadget $A(t_1)$ and from gadget $B(t_2)$ to gadget $C(t_3)$, then T^* must also include arcs characterised by the following

1. *all non-marked nodes in $A(t_1)$ have non-marked heads in $B(t_2)$,*
2. *all non-marked nodes in $C(t_3)$ have marked heads in $B(t_2)$, and*
3. *all marked nodes in $A(t_1)$ have marked heads in $C(t_3)$.*

Proof. There are only arcs of weight 2 in D from the B-region to both the A- and the C-regions, and internally in the B-region. We show that there are at most k weight 2 arcs connecting the A- and B-regions and C- and B-regions. Then we show that the maximal number of weight 2 edges internal to the B-region is $(|K| - 1)$.

Suppose there are more than k arcs of weight 2 from the B-region to the A-region in T^*. Then there are at least two of these arcs entering different A-gadgets: $(b_{1,j',t'}, a_{1,j,t})$ and $(b_{1,i',p'}, a_{1,i,p})$, with $p < t$. Consider the barred symbols in the tth A-gadget. There are only two possible heads: (1) the symbol following the gadget (region connectivity arcs (1a) or boundary connectivity arcs (2a)), which by projectivity is excluded because these arcs would cross $(b_{1,j',t'}, a_{1,j,t})$, or (2) symbols from the C-region, which by projectivity is also excluded because they would cross the arc $(b_{1,i',p'}, a_{1,i,p})$.

The proof that there are at most k arcs of weight 2 from the B-region to the C-region in T^* is analogous.

For the maximal number of arcs of weight 2, internal to the B-region, we first consider the maximum number of weight 2 region connectivity arcs (4). By projectivity, a B-gadget with arcs entering an A- or C-gadget cannot have any entering weight 2 region connectivity arc. Also, by projectivity, a single B-gadget can have at most 1 weight 2 region connectivity arc. Thus, the number of weight 2 arcs would be maximised by ensuring arcs exiting the B-region originate from the same B-gadget, so only one B-gadget does not have weight 2 entering arcs. Since there are $|K|$ B-gadgets in total, this means there are at most $|K| - 1$ weight 2 B-region internal arcs.

The rest of the proof follows by the similar projectivity arguments. □

Lemma 1.3. *T^* has weight $|w|+2k-1+(|K|-1)$ if and only if there is a $3k$-clique in G.*

Proof. ($\Leftarrow$) Suppose there is a $3k$-clique in G consisting of the three k-cliques k_1, k_2, and k_3, and such that $k_1 \cup k_2 \cup k_3$ is a $3k$-clique. In w, there must be corresponding gadgets in each of its gadget regions. We consider $A(1)$ the gadget for k_1 in A, by $B(2)$ the gadget for k_2 in B, and by $C(3)$ the gadget for k_3 in C. We build up a set S of arcs based on these three gadgets. The set S consists of all the possible G-induced (weight 1 and 2) arcs between these three regions–a disconnected set where no two arcs cross, by Lemma 1.2. By the same lemma, S includes exactly $2k + (|K| - 1)$ arcs of weight 2. We will add arcs to S to connect the rest of the symbols in w until we form a tree, and by Lemma 1.2 again we cannot add any further weight 2 arcs.

We must now supplement S to make a tree. We first connect the B-region. For $t < 2$, we connect B-gadgets internally by making the path from the R_t to L_t, using weight 1 region connectivity arcs. We make paths in the opposite direction, from L_t

to R_t for $t > 2$. We then add all possible weight 2 region connectivity arcs. This makes A(1) and B-region connected.

All other A-gadgets are connected as in the proof of Lemma 1.1. Similarly for the C-gadgets before and after $C(3)$.

The only nodes that still lack a head node are the marked nodes from $C(3)$. We connect these using region connectivity arcs from (1g).

We have now constructed a projective tree of weight $|w| - 1 + 2k + (|K| - 1)$. We cannot have a higher weighted projective tree by Lemma 1.2. Hence tree is an optimal T^*.

($\Rightarrow$) Suppose T^* has weight $|w| - 1 + 2k + (|K| - 1)$. By Lemma 1.2, T^* has exactly k arcs of weight 2 from the B-region to the A-region, and k from the B-region to the C-region, and that in this case all possible G-induced arcs between the three corresponding gadgets are in T^*. Moreover, internally to the B-region, there are $|K| - 1$ weight 2 edges.

Let the gadgets be w.l.o.g., A(1), B(2), and C(3). Each unmarked b symbol in B(2) corresponds to a node in V, and is the head of an unmarked symbol from A(1) corresponding to every node in k-clique k_1. This means that in G, all possible connections between nodes in k_1 and k_2 exist. The same holds for B(2) with C(3) and C(3) with A(1). Hence there is a $3k$-clique in G. □

□

With Theorem 1, we have shown that the non-structural graph-based neural parsing systems cannot be carrying out explicit exact decoding in with a significantly simpler algorithm. As we show in the next section, in fact, the LSTM stacks of these systems alone are powerful enough to simulate all components.

In our proof, the algorithm consistently makes a choice between edges of weight 1 and edges of weight 2 for the result to preserve projectivity. Possibly more edges of weight 1 may end up in a maximum spanning projective DAG or digraph, so we cannot necessarily use the weight in the same way to deduce the result. The number of edges in D is less than n^2. Hence if we replace the weights of weight 1 arcs in D by weight $1/(n^2)$, then an output maximum spanning projective digraph or DAG with weight superior to $2k + (|K| - 1)$ would indicate a $3k$-clique. By the algorithms to do this from (Schluter, 2015) in cubic time, we therefore have the same lower bound for finding a maximum spanning projective DAG or digraph.

Corollary 1.1. *Under the assumption of ETH, there is no algorithm that carries out projective maximum spanning DAG or digraph decoding in sub-cubic time.*

5 RNN Simulation of Eisner's Algorithm

Eisner (1996)'s algorithm on an input sentence of length n uses an $n \times n$ table M and dynamic programming to compute for table cell $M_{i,j}$ the highest weighted sub-trees over the span (i, j) of the input sentence. The algorithm iterates over spans of increasing length. For $M_{i,j}$, the weights of all possible combinations of sub-spans are considered as candidate sub-trees over the span, and the maximum of these is retained in $M_{i,j}$.

For our purposes, the problem with this version of the algorithm is that the RNN cannot compute the maximum of the corresponding $O(n)$ values in either constant space nor in one time-step, and the corresponding sub-tree weight is required in the computation of maximum sub-trees over the span $j - i + 1$ at the next recursive step.

In Algorithm 1, we precompute enough of the comparisons required for finding the maximum spanning sub-tree combination before the algorithm arrives in that table cell (from line 5). Thus, instead of taking the maximum across $k \in O(n)$ values, we only ever take the maximum across 2 values at a time. We now explain this algorithm.

A sub-tree over the span (i, j) is said to be *complete* if it includes some arc between i and j. Otherwise the sub-tree is called *incomplete*. We use seven weight matrices (which we extend to 25 matrices later):

- S an $n \times n$ matrix of arc scores, where $S[i, j]$ is the score of the (i, j) arc.
- I an $n \times n$ matrix of incomplete sub-tree scores, where $I[i, j, h]$ is the incomplete sub-tree for the span (i, j) with head $h \in 0, 1$. If $h = 0$, then i is the root of the sub-tree, and if $h = 1$, then j is the root.
- C is defined in the same way as I but for complete sub-trees.
- $I_r[i, j, h]$ (resp. C_r) stores the current "row"-maximum value for $I[i, j, h]$ across the span combinations $(i, k), (k + 1, j)$ for $k - i > (k + 1) - j$ (resp. $(i, k), (k, j)$ for $k - i > k - j$). These are the cases where the span (i, k) is the largest of the two sub-spans (i, j). These table values are adjusted while the algorithm visits cells (i, k).

- $I_c[i,j,h]$ (resp. C_c) stores the current "column"-maximum value for $I[i,j,h]$ across the span combinations $(i,k),(k+1,j)$ for $k-i \leq k+1-j$ (resp. $(i,k),(k,j)$ for $k-i > k-j$). These are the cases where the span $(k+1,j)$ (resp. (k,j)) is larger or equal to the other sub-span of the partitioned span (i,j). These table values are adjusted while the algorithm visits cells $(k+1,j)$ (resp. (k,j)).

The pseudocode for this algorithm, which we refer to as `streaming-max-eisner` is presented in Algorithm 1. The main difference with the original version is that the internal loop partitioning of a span is separated in Algorithm 1 over several previous iterations of the loop, so that once the algorithm visits cell (i,j), all that needs to be computed is the maximum of the two row- and column-maximum values, from I_r and I_c, or from C_r and C_c.

It is straightforward to show the correctness of this algorithm, which we state as Theorem 2. We omit the proof due to space constraints. The algorithm can also be easily adapted for backtracking.

Theorem 2. *Algorithm 1 returns the weight of* T^*.

We make a final adjustment to the algorithm before stating the simulation construction. For the simulation, we only have RNN operations at our disposal: linear combinations and a `ReLU` activation function, but no explicit max operation. In order to use only RNN operations, we replace the explicit max function.

Replacing the explicit max function. We note that to find the maximum of the two positive numbers a and b, we can use the `ReLU` function. Without loss of generality, suppose that $a > b$, then

$$\begin{aligned}\frac{(\texttt{ReLU}(a-b)+\texttt{ReLU}(b-a)+a+b)}{2} \\ = \frac{(a-b+a+b)}{2} = \frac{2a}{2} = a \\ = \max(a,b). \quad (1)\end{aligned}$$

In fact, since all weights are assumed positive, Equation 1 can be rewritten as

$$\begin{aligned}\max(a,b) = \frac{1}{2}(\texttt{ReLU}(a-b)+\texttt{ReLU}(b-a) \\ +\texttt{ReLU}(a)+\texttt{ReLU}(b)). \quad (2)\end{aligned}$$

We therefore make a final adjustment to the original Eisner algorithm, over the version Algorithm 1, replacing all max functions using Equation 2. Instead of storing only one value for each matrix I_r, I_c, C_r, C_c, I, C, we store four, denoted by the fields a, b, ab, ba corresponding the four values we need to store: $\texttt{ReLU}(a), \texttt{ReLU}(b), \texttt{ReLU}(a-b)$, and $\texttt{ReLU}(b-a)$ respectively. For instance, for the matrix I, we have I_a, I_b, I_{ab}, I_{ba}. Then, for example, line 6 becomes

$$\begin{aligned}a \leftarrow \texttt{ReLU}(\frac{1}{2} * (I_r[i,j,0].a + I_r[i,j,0].b \\ + I_r[i,j,0].ab + I_r[i,j,0].ba)) \quad (3) \\ b \leftarrow \texttt{ReLU}(\frac{1}{2} * (I_c[i,j,0].a + I_c[i,j,0].b \\ + I_c[i,j,0].ab + I_c[i,j,0].ba)) \quad (4) \\ I[i,j,0].a \leftarrow \texttt{ReLU}(a) \\ I[i,j,0].b \leftarrow \texttt{ReLU}(b) \\ I[i,j,0].ab \leftarrow \texttt{ReLU}(a-b) \\ I[i,j,0].ba \leftarrow \texttt{ReLU}(b-a)\end{aligned}$$

where Equations 3 and 4 are wrapped in an extra `ReLU` operation which yields no difference to the parameter, but which will be convenient for our simulation in Section 5.

Lines 11-14 and 16-19 are adapted in the same way. We provide the adaption of line 11 to make this precise:

$$\begin{aligned}a \leftarrow \texttt{ReLU}(\frac{1}{2} * (I[i,p,1].a + I[i,p,1].b \\ + I[i,p,1].ab + I[i,p,1].ba)) \\ b \leftarrow \texttt{ReLU}(\frac{1}{2} * (C[i,j,0].a + C[i,j,0].b \\ + C[i,j,0].ab + C[i,j,0].ba \\ + C[j+1,p,1].a + C[j+1,p,1].b \\ + C[j+1,p,1].ab + C[j+1,p,1].ba \\ + S[p,i])) \\ I_r[i,p,1].a \leftarrow \texttt{ReLU}(a) \\ I_r[i,p,1].b \leftarrow \texttt{ReLU}(b) \\ I_r[i,p,1].ab \leftarrow \texttt{ReLU}(a-b) \\ I_r[i,p,1].ba \leftarrow \texttt{ReLU}(b-a).\end{aligned}$$

Algorithm 1 therefore uses 25 matrices on input of length n–hence, still $O(n^2)$ space.

Simulating Algorithm 1. The projective dependency parsing architecture M' to be simulated first sends word embeddings $x_i, i \in [n]$ through a forward (and backward) LSTM with output word representations $\overrightarrow{o}'_i$ (and $\overleftarrow{o}'_i$) of dimension d. The concatenated result $[\overrightarrow{o}'_i; \overleftarrow{o}'_i]$ is further specialised through two unrelated nonlinear dense layers: one for dependents and one for heads. Then all resulting pairs (dependent,head) of word representations are sent through a scoring function to generate a score matrix as input to projective MST decoding (Kiperwasser and Goldberg, 2016).

Algorithm 1 Projective MST algorithm computing the maximum over at most 2 arguments.

```
1:  procedure STREAMING-MAX-EISNER
2:      $S, I, C, I_r, I_c, C_r, C_c$ all initialised to 0 matrices
3:      for $t \leftarrow 1$ to n-1 do                                        ▷ span for-loop
4:          for $i \leftarrow 1$ to $n - t$ do                                 ▷ diagonal for-loop
5:              $j \leftarrow i + t$                                           ▷ the algorithm is visiting cells $(i, j)$
6:              $I[i, j, 0] \leftarrow \max(I_r[i, j, 0], I_c[i, j, 0])$
7:              $I[i, j, 1] \leftarrow \max(I_r[i, j, 0], I_c[i, j, 1])$
8:              $C[i, j, 0] \leftarrow \max(C_r[i, j, 0], C_c[i, j, 0])$
9:              $C[i, j, 1] \leftarrow \max(C_r[i, j, 0], C_c[i, j, 1])$
10:             for $p \leftarrow j + 1$ up to $\min(j + t + 1, n)$ do     ▷ streaming-row for-loop, i.e., while $p - (j + 1) \leq t$
11:                 $I_r[i, p, 1] = \max(I[i, p, 1], C[i, j, 0] + C[j + 1, p, 1] + S[p, i])$
12:                 $I_r[i, p, 0] = \max(I[i, p, 0], C[i, j, 0] + C[j + 1, p, 1] + S[i, p])$
13:                 $C_r[i, p, 1] = \max(C[i, p, 1], C[i, j, 1] + I[j + 1, p, 1])$
14:                 $C_r[i, p, 0] = \max(C[i, p, 0], I[i, j, 0] + C[j + 1, p, 0])$
15:             for $p \leftarrow i - 1$ down to $\max(i - 1 - t, 1)$ do   ▷ streaming-column for-loop, i.e., while $(i - 1) - p \leq t$
16:                 $I_c[p, j, 1] = \max(I[p, j, 1], C[p, i - 1, 0] + C[i - 1, j, 1] + S[p, i])$
17:                 $I_c[i, p, 0] = \max(I[i, p, 0], C[i, j, 0] + C[j + 1, p, 1] + S[i, p])$
18:                 $C_c[i, p, 1] = \max(C[i, p, 1], C[i, j, 1] + I[j + 1, p, 1])$
19:                 $C_c[i, p, 0] = \max(C[i, p, 0], I[i, j, 0] + C[j + 1, p, 0])$
20:     Return $I, C$
```

The architecture M to simulate M' consists of two components, each being a recurrent layer: a BiLSTM (for contextual word representations and word specialisations) and an RNN (for scoring and to simulate Algorithm 1).

M starts by feeding word embeddings x_i into its first component, the BiLSTM. In the forward direction, at the tth time step, the contextual representation $\overrightarrow{o}'_t$ is generated, $\overrightarrow{o}'_{t-1}$ is specialised to $\overrightarrow{o}^h_{t-1}$ (head) and $\overrightarrow{o}^d_{t-1}$ (dependent), and the previously specialised word representations in $\overrightarrow{o}_t$ (i.e., corresponding to $\overrightarrow{o}'_1, \ldots, \overrightarrow{o}'_{t-2}$) are copied over. We add a single extra $(n + 1)$th time step to each direction, so M can finish specialising contextualised word representations within this first component. Similarly for the backward direction.

There is one single input to M's second component, an RNN, which also works in $n + 1$ time steps. We refer to the inputs for this component as $z_1, \ldots, z_{(n+1)}$, where $z_2 \ldots z_{(n+1)}$ are all dummy inputs. z_1 is the concatenation of the final output vectors from each direction of M's BiLSTM. In the first time step of this component, M computes the score matrix and stores it in the hidden state h_1. The hidden state has a dimension large enough to house the 25 tables ($O(n^2)$ space) required by Algorithm 1 for subtree score bookkeeping and computing the maximum of two values using linear combinations and a `ReLU`.

The outer loop (the *span for-loop* with variable t) of the algorithm corresponds to each time-step t of the RNN. For the first internal for-loop (the *diagonal for-loop* with variable i), we note that, in lines 6-9, no cells $(i, i + t)$ whose values are being computed require information from each other at this time-step t.

The *streaming-row* and *streaming-column* for loops (lines 11-14, 16-19) on the other hand sometimes requires maximal values $(i, i + t)$ from lines 6-9 to be computed. This problem is simply solved by replacing the corresponding expressions appearing as left-hand sides in lines 6-9 by the right-hand sides.

The output h_{n+1} contains the desired maximum value.

6 Concluding Remarks

Recent state-of-the art neural graph-based parsers comprising, among other components, a short stack of BiLSTMs, seem to obviate any explicit structural learning or prediction. In this paper, under the assumption of ETH, we showed that this is not due to any possible indirect discovery of a faster algorithm for finding a projective maximum spanning tree and extended the result to projective maximimum spanning DAGs and digraphs. We further showed how these architectures allow for simulating decoding, implying that they are indeed carrying out implicit structured learning and prediction.

Acknowledgments

We gratefully acknowledge the careful remarks of the anonymous NAACL reviewers.

References

Amir Abboud, Arturs Backurs, and Vanessa Vassilevska Williams. 2015. If the current clique algorithms are optimal, so is Valiant's parser. In *Proceedings of FOCS*.

Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. 2006. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 8:1346–1367.

Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May, and Kevin Knight. 2018. Recurrent neural networks as weighted language recognizers. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2261–2271, New Orleans, Louisiana. Association for Computational Linguistics.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas. Association for Computational Linguistics.

Timothy Dozat and Christopher M. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer-Verlag.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Russell Impagliazzo and Ramamohan Paturi. 1999. The complexity of k-SAT. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, pages 237–240.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the ACL*, 4:313–327.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Lillian Lee. 1997. Fast context-free parsing requires fast boolean matrix multiplication. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 9–15, Madrid, Spain. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1214, Brussels, Belgium. Association for Computational Linguistics.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

Giorgio Satta. 1994. Tree-adjoining grammar parsing and Boolean matrix multiplication. *Computational Linguistics*, 20:173–191.

Natalie Schluter. 2015. The complexity of finding the maximum spanning DAG and other restrictions for DAG parsing of natural language. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 259–268, Denver, Colorado. Association for Computational Linguistics.

Hava T. Siegelmann. 1996. Recurrent neural networks and finite automata. *Computational Intelligence*, 12:567574.

Daniel Varab and Natalie Schluter. 2018. Uniparse: A universal graph-based parsing toolkit. *CoRR*, abs/1807.04053.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision rnns for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings

Gijs Wijnholds
Queen Mary University of London
`g.j.wijnholds@qmul.ac.uk`

Mehrnoosh Sadrzadeh
Queen Mary University of London
`mehrnoosh.sadrzadeh@qmul.ac.uk`

Abstract

Ellipsis is a natural language phenomenon where part of a sentence is missing and its information must be recovered from its surrounding context, as in “Cats chase dogs and so do foxes.”. Formal semantics has different methods for resolving ellipsis and recovering the missing information, but the problem has not been considered for distributional semantics, where words have vector embeddings and combinations thereof provide embeddings for sentences. In elliptical sentences these combinations go beyond linear as copying of elided information is necessary. In this paper, we develop different models for embedding VP-elliptical sentences. We extend existing verb disambiguation and sentence similarity datasets to ones containing elliptical phrases and evaluate our models on these datasets for a variety of non-linear combinations and their linear counterparts. We compare results of these compositional models to state of the art holistic sentence encoders. Our results show that non-linear addition and a non-linear tensor-based composition outperform the naive non-compositional baselines and the linear models, and that sentence encoders perform well on sentence similarity, but not on verb disambiguation.

1 Introduction

Compositional distributional semantics has so far relied on a tight connection between syntactic and semantic resources. Based on the assembly principle of compositionality, these models assign a sentence vector by applying a linear map to the individual word embeddings therein. The meaning of “cats chase dogs” is as follows in (1) additive, (2) multiplicative, and (3) tensor-based models:

(1) $\overrightarrow{cats} + \overrightarrow{chase} + \overrightarrow{dogs}$
(2) $\overrightarrow{cats} \odot \overrightarrow{chase} \odot \overrightarrow{dogs}$
(3) $\overrightarrow{cats}^{\top} \times (\overline{chase} \times \overrightarrow{dogs})$

Some linguistic phenomena, however, rely on copying resources while computing meaning; canonical examples thereof are anaphora and ellipsis, exemplified below:

(a) Cats clean themselves.
(b) Cats chase dogs, children do too.

More complex examples involve a structural ambiguity such as the following:

(c) Cats chase their tail, dogs too.

These lend themselves to a strict (dogs chase the cat’s tail) and a sloppy reading (dogs chase their own tail). In these examples, the meaning of at least one part of the sentence is used twice, e.g. the subject in a, the verb phrase “chase dogs” in b. Such cases can often be extended to a situation in which a meaning is used more than twice, e.g. in “Cats chase their tail, dogs too, and so do foxes”.

In order to develop distributional semantics for such sentences while respecting the principle of compositionality, one has a choice between a linear or a non-linear composition of resources. In the linear case, no information is copied, resulting in vector embeddings such as the following one (when only considering content words):

$$\overrightarrow{cats} + \overrightarrow{chase} + \overrightarrow{dogs} + \overrightarrow{children}$$

In the non-linear case, the necessary resources are copied to resolve the ellipsis, resulting in vectors embeddings such as:

$$\overrightarrow{cats} + \overrightarrow{chase} + \overrightarrow{dogs} + \overrightarrow{children} + \overrightarrow{chase} + \overrightarrow{dogs}$$

One has the same choice when dealing with multiplicative and tensor-based models. The question is which of these composition frameworks, i.e. linear versus non-linear, provides a better choice for embedding elliptical sentences. To our knowledge, this has remained an open question: although some theoretical work has been done to

Proceedings of NAACL-HLT 2019, pages 261–271
Minneapolis, Minnesota, June 2 - June 7, 2019.

model verb phrase ellipsis in compositional distributional semantics (Wijnholds and Sadrzadeh, 2018), none of the existing datasets or evaluation methods for distributional semantics focus on elliptical phenomena.

In this paper, we provide some answers. Our starting point is the lambda logical forms of sentences, e.g. those produced by the approach of Dalrymple et al. (1991), which uses a higher order unification algorithm to resolve ellipsis. We apply to these the lambdas-to-vectors mapping of Muskens and Sadrzadeh (2016, 2017) to homomorphically map the lambda terms into concrete vector embeddings resulting from a multitude of composition operators, such as addition, multiplication, and tensor-based. We work with four vector spaces (count-based, Word2Vec, GloVe, FastText) and three different verb embeddings, and contrast our compositional models with state of the art holistic sentence encoders.

We evaluate the sentence embeddings by using them in a verb disambiguation and in a sentence similarity task, created by extending previous SVO tasks from Grefenstette and Sadrzadeh (2011a) and Kartsaklis and Sadrzadeh (2013) to an elliptical setting, and obtaining new human judgements using the Amazon Mechanical Turk crowdsourcing tool. Our experiments show that in both tasks, the models that use a non-linear form of composition perform better than the models whose composition framework is linear, suggesting that resolving ellipsis contributes to the quality of the sentence embedding.

2 Background

Single-Word Embeddings: Distributional semantics on the word level relies on the embedding of word meaning in a vectorial form: by taking context words as the basis of a vector space one computes the vector components of each word by considering its distribution among corpus data. Then a similarity measure is defined on the vector space via the cosine similarity. In a count-based model, the context is taken to be a linear window and the corpus is traversed to collect raw co-occurrence counts. Then, a weighting scheme is applied to smooth the raw frequencies in the meaning representation. More discussion on count-based vector space models can be found in (Turney and Pantel, 2010), and a systematic study of the parameters of count-based word embeddings is given by (Kiela and Clark, 2014).

With the rise of deep learning techniques, much attention has been given to neural word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017), which try to predict rather than observe, the context of a word by optimising an objective function based on the probability of observing a context.

Compositional Models: The key idea of compositional models is that the meaning of elementary constituents can be combined in a structured way to obtain a representation for larger phrases. In a distributional setting, having a compositional operator is imperative: a data-driven model would not be adequate given the sparsity of full sentences in a corpus. Moreover, it is not clear that sentences follow the distributional hypothesis.

Concrete composition operators can roughly be classified as simple and tensor-based. Simple models add or multiply the word vectors to obtain a sentence vector. The work of Mitchell and Lapata (2010) experiments with these models. Tensor-based models differ in that they represent complex words as vectors of a higher order: Baroni and Zamparelli (2010) represents adjectives as matrices which, applied to a word vector produce a vector representation of the compound adjective-noun combination. The account of (Coecke et al., 2010, 2013; Clark, 2015) generalises this to higher-order tensors, e.g. cubes for transitive verbs and hypercubes for ditransitive verbs. The benefit of a type-driven approach over the simple models is that they respect the grammatical structure of sentences: the meaning of “man bites dog” is distinct from that of “dog bites man” whereas in an additive/multiplicative model they would be identical. The trade-off is that the tensors themselves have to be learnt; where Baroni and Zamparelli (2010) apply regression learning to learn the content of adjective matrices, for transitive verbs there have been several approaches using multistep regression learning (Grefenstette et al., 2013), relational learning (Grefenstette and Sadrzadeh, 2011a), or a combination of co-occurrence information with machine learning techniques (Polajnar et al., 2014a,b; Fried et al., 2015). A comparative study between count-based and neural embeddings in a compositional setting was carried out by (Milajevs et al., 2014).

Neural composition turns the problem of compositionality around by learning the composition operator instead of predicting the result. Examples are Skip-Thought Vectors (Kiros et al., 2015), the Distributed Bag of Words model (Le and Mikolov, 2014), InferSent (Conneau et al., 2017), and Universal Sentence Encoder (Cer et al., 2018).

Ellipsis, Formally: There exists many formal approaches to ellipsis and anaphora in the literature. These have generally taken either a syntactic or a semantic form[1]. Examples of the syntactic approaches are in the work of Hendriks and Dekker (1995); Morrill and Valentín (2015); Jäger (2006); Kubota and Levine (2017); these use directional extensions of categorial grammars that allow for the syntactic types at the site of ellipsis be unified with copies of the types at the antecedent of the elliptical phrase. Another approach deletes the syntactic structure at the ellipsis site and reconstruct it by copying across the antecedent structure (Fiengo and May, 1994; Merchant, 2004).

Semantic approaches (Dalrymple et al., 1991; Szabolcsi, 1987; Pulman, 1997) assume that ellipsis involves underspecification of content and resolve this by producing a predicate via a suitable abstraction from the antecedent. For instance, the elliptical phrase (b) "Cats chase dogs, children do too", will take an initial logical form (b_1); a resolution step (b_2) provides it with the lambda term in (b_3), which constitutes its final semantic form:

$$(b_1) \quad chase(cats, dogs) \wedge P(children)$$

$$(b_2) \quad P = \lambda x.chase(x, dogs)$$

$$(b_3) \quad (b_1) \rightsquigarrow_\beta chase(cats, dogs) \\ \wedge\, chase(children, dogs)$$

The ambiguous example (d) "Cats chase their tails, dogs too" is treated similarly, but can now obtain its respective strict and sloppy readings by producing predicates (d_1) and (d_2) below:

$$(d_2) \quad P = \lambda x.chase(x, tail(cats))$$

$$(d_3) \quad P = \lambda x.chase(x, tail(x))$$

Mixed syntactic/semantic approaches have also been proposed to cover wider ranges of phenomena; see Kempson et al. (2015) for an overview.

The only existing work attempting to join ellipsis analysis with vector embeddings is the proposal of (Kartsaklis et al., 2016), which is preliminary work and gives unwanted results[2]. Below, we develop a new such approach.

3 Embeddings for Elliptical Phrases

Vectors and their basic operations can be *emulated* using a lambda calculus with constants for the relevant operations, as shown in (Muskens and Sadrzadeh, 2016). They assume a type I (a finite index set) and R (modelling the real numbers) and model any vector as a term of type $V := IR$; that is, as a function from indices to real numbers. Matrices can then be represented by types $M := IIR$ and in general a tensor of rank n will have type $T^n := I_1 \ldots I_n R$. The standard operations like scalar multiplication, addition, element wise multiplication and tensor contraction can be modelled with lambda terms as follows:

$$\begin{array}{ccll}
\cdot & := & \lambda rvi.r \cdot v_i & : RVV \\
+ & := & \lambda vwi.v_i + w_i & : VVV \\
\odot & := & \lambda vwi.v_i \cdot w_i & : VVV \\
\times_1 & := & \lambda mvij.\sum\limits_j m_{ij} \cdot v_j & : MVV \\
\times_2 & := & \lambda cvijk.\sum_k c_{ijk} \cdot v_k & : T^3VM
\end{array}$$

The first three definitions above extend the arithmetic operations of addition and multiplication on real numbers in R to lists of numbers in IR and define corresponding definitions on vectors, and so $\odot$ defines the pointwise multiplication of two vectors. The operation $\times_1$ defines matrix multiplication; $\times_2$ defines the tensor contraction between a cube c (in I^3R) and a list of numbers v.

c	$\mathcal{H}(c)$	$\mathcal{T}(c)$
`cn`	**cn**	V
`adj`	$\lambda v.(\mathbf{adj} \times_1 v)$	VV
`adv`	$\lambda v.(\mathbf{adv} \times_1 v)$	VV
`itv`	$\lambda v.(\mathbf{itv} \times_1 v)$	VV
`tv`	$\lambda uv.(\mathbf{tv} \times_2 v) \times_1 u$	VVV
`coord`	$\lambda P.\lambda Q.P \nabla Q$	VVV
`quant`	$\lambda vZ.Z(\mathbf{quant} \times_1 v)$	$V(VV)V$

Table 1: Lambda Vector look up table for a tensor-based composition model. `cn`: a common noun, `adj`: adjective, `adv`: adverb, `itv`: intransitive verb, `tv`: transitive verb, `coord`: coordinator, `quant`: generalised quantifier; P, Q are variables of type V, and so are v, u, Z is a variable of type VV, ∇ is either $\odot$ or $+$.

The vector semantics of a lambda term m is computed by taking a homomorphic image over the set of its constants c. This image is computed compositionally from the vector or tensor embeddings of the constants c of m via their homomorphic images $\mathcal{H}(c)$, whose types are denoted by $\mathcal{T}(c)$.

Examples of these are given in Table 1 for a

---

[1] Although pragmatics approaches exist (Merchant, 2010), we focus here on syntactic and semantic approaches.

[2] The meaning of "Bill brought apples and John pears" coincides with that of "Bill and John brought apples and pears".

tensor-based composition model, where the boldface **c** denotes the vector/tensor embedding of c.

Using this table, we obtain homomorphic images of any lambda term over the constants. For instance, the lambda term of our exemplary resolved ellipsis phrase (b_3) $\mathit{chase}(\mathit{cats}, \mathit{dogs}) \wedge \mathit{chase}(\mathit{children}, \mathit{dogs})$ is given the following semantic, obtained by computing $\mathcal{H}(b_3)$:

$$((\mathbf{chase} \times_2 \mathbf{dogs}) \times_1 \mathbf{cats})\nabla$$
$$((\mathbf{chase} \times_2 \mathbf{dogs}) \times_1 \mathbf{children})$$

The constituents of the $\mathcal{H}(c)$ entries of Table 1 are only exemplary. Many other interpretations are possible. For instance, taking vector embeddings for all words and replacing all tensor contractions and ∇ by $+$ defines a purely additive model. The concrete models for transitive sentences that were evaluated by Milajevs et al. (2014) can all be derived by varying the $H(c)$ entries. Below are the sentences obtained by using the Copy Object (CO), Frobenius Additive (FA), Frobenius Multiplicative (FM) and Frobenius Outer (FO) instantiations of the verb, respectively:

$$\begin{aligned}
\mathbf{CO}: &\quad \lambda os.o \odot (\mathbf{verb} \times^{\top} s)\\
\mathbf{FA}: &\quad \lambda os.s \odot (\mathbf{verb} \times o) + o \odot (\mathbf{verb} \times^{\top} s)\\
\mathbf{FM}: &\quad \lambda os.s \odot (\mathbf{verb} \times o) \odot o \odot (\mathbf{verb} \times^{\top} s)\\
\mathbf{FO}: &\quad \lambda os.s \odot (\mathbf{verb} \times o) \otimes o \odot (\mathbf{verb} \times^{\top} s)
\end{aligned}$$

The vector semantics of the extensions of transitive sentences with VP elliptical phrases are obtained by taking each of the above as the semantics of each conjunct of the lambda logical form and interpreting the conjunction operation of $\wedge$ as either sum or multiplication.

4 Experimental Evaluation

For the evaluation of the model(s) in the previous section, we built two new datasets and experimented with count based and neural vector spaces, and sentence encoders.[3].

4.1 Building new datasets

In order to experiment with ellipsis, we extended the verb disambiguation dataset of Grefenstette and Sadrzadeh (2011a) and the transitive sentence similarity dataset of Kartsaklis and Sadrzadeh (2013), henceforth GS2011 and KS2013.

4.1.1 GS2011

The GS2011 verb disambiguation dataset contains 10 verbs, each with two possible interpretations. For each verb v and its two interpretations v_1 and v_2, the dataset contains human similarity judgments for 10 subject-object combinations. For instance, for the verb *meet* – ambiguous between *visit* and *satisfy* – the dataset contains the pairs ⟨*system meet requirements, system satisfy requirements*⟩ and ⟨*system meet requirements, system visit requirements*⟩. The more likely interpretation is marked as HIGH whereas the unlikely interpretation is marked LOW.

We extended this dataset as follows: for each combination of a verb triple (v, v_1, v_2) and a subject-object pair (s, o), where $\langle s\ v\ o,\ s\ v_1\ o\rangle$ is expected to have LOW similarity in the dataset and $\langle s\ v\ o,\ s\ v_2\ o\rangle$ is thus expected to have HIGH similarity, we selected a new subject s^* from the list of most frequent subjects for the verb v_2 such that it was significantly more frequent for v_2 than for v_1[4]. By doing so we strengthened the disambiguating effect of the context for each verb. The subject was selected such that the resulting elliptical phrase pairs made sense. For each combination and new subject considered, we added the two sentence pairs in the elliptical form

$$\langle s\ v\ o \text{ and } s^* \text{ does too},\ s\ v_1\ o \text{ and } s^* \text{ does too}\rangle$$
$$\langle s\ v\ o \text{ and } s^* \text{ does too},\ s\ v_2\ o \text{ and } s^* \text{ does too}\rangle$$

For example, for the verb triple (*draw, depict, attract*), and original sentence pairs

⟨*man draw sword, man depict sword* ⟩
⟨*man draw sword, man attract sword* ⟩

we selected the new subject *artist* and added two pairs, comparing *man draw sword and artist does too* with

man depict sword and artist does too
man attract sword and artist does too

We selected two new subjects for each combination, and in this way we obtained a dataset of roughly 400 entries. New human judgments were collected through Amazon Mechanical Turk, by prepending *the* to each noun and putting the phrase in the past tense. As with the original dataset, participants were asked to judge the similarity between sentence pairs using a discrete number between 1 and 7; 1 for highly dissimilar, 7 for highly similar. By inserting gold standard pairs of identical sentences we checked if participants were

---

[3] All models, the new datasets, and evaluation code are available at `github.com/gijswijnholds/compdisteval-ellipsis`

[4] As found in the combined ukWaC+WackyPedia corpus.

trustworthy. We collected 25 judgments per sentence pair but excluded participants that annotated less than 20 entries of the total dataset. We ended up with 55 different participants who ranked more than 20 entries of the total dataset, to give a final amount of ca. 9200 annotations. As an example, the verb *show* was a very hard case to disambiguate in the GS2011 dataset: *child show sign* had an average score of 2.5 with both *child picture sign* and *child express sign*. In the new dataset, with the extra subject *patient*, it got much clearer that the verb had to be interpreted as *express* with an average score of 5.869, versus 4.875 for *picture*.

4.1.2 KS2013

The KS2013 sentence similarity dataset contains 108 transitive sentence pairs annotated with human similarity judgments. As opposed to the GS2011 dataset, subjects and objects of each sentence pair are not the same, so several different contexts get compared to one another. In this sense, the KS2013 dataset aims to investigate the role of content of individual words versus the role of composition, as the similarity of sentences might be predictable from the contribution of individual words rather than the specific way of composing them.

We extend this dataset to cover VP ellipsis by following a similar procedure as for GS2011. For each transitive sentence of the form $s\ v\ o$ in the dataset, we selected a new subject s^* from a list of most frequent subjects of the verb[5] and built elliptical entries $s\ v\ o$ *and* s^* *does too* in such a way that the meaning of the original transitive sentence got changed as little as possible and that the resulting elliptical phrase made sense. We then considered every transitive sentence pair in the dataset and added the new respective subjects to both sentences. For example, for the pair

$\langle$*school encourage child, employee leave company*$\rangle$

we selected *parent* and *student* to get the new pair

$\langle$*school encourage child and parent does too,*
employee leave company and student does too$\rangle$

We chose two subjects for every original sentence, generating four possibilities for each sentence pair, and a new dataset of 432 entries. This dataset was also annotated using Amazon Mechanical Turk, after putting each verb in the past tense and prepending *the* to each noun in the dataset. Gold standard pairs of identical sentences were inserted to validate trustworthiness of participants. The final dataset contains ca. 9800 annotations by 42 different participants.

4.2 Vector Spaces

To provide a comprehensive study with robust results, we used four vector spaces: a count based vector space, and newly trained Word2Vec, GloVe, and FastText spaces, as detailed below.

Count-Based: We used the combined ukWaC and Wackypedia corpora[6] to extract raw co-occurrence counts, using as a basis the 2000 most frequently occurring tokens (after excluding the 50 most frequent ones). When extracting counts, we disregarded a list of stopwords that do not contribute to the content of the vectors. We used a context window of 5 around the focus word, and PPMI as weighting scheme. These settings were use in the original KS2013 dataset (Kartsaklis and Sadrzadeh, 2013).

Word2Vec: The Word2Vec embeddings we used were trained with the continuous bag of words model of (Mikolov et al., 2013) (CBOW). We trained this model on the combined and lemmatised ukWaC and Wackypedia corpora, using the implementation for Python available in the `gensim` package[7], with a minimum word frequency of 50, a window of 5, dimensionality 300, and 5 training iterations.

GloVe: The GloVe model (Pennington et al., 2014) considers the *ratio* of co-occurrence probabilities by minimising the least-squares objective between the dot product of two word embeddings and the log-probability of the words' co-occurrence. We trained a GloVe space on the combined and lemmatised ukWaC and Wackypedia corpora, using the code provided by the original authors[8]. Similar to the Word2Vec settings above, we trained 300 dimensional vectors with a minimum word frequency of 50 and a window of 5, but we trained with 15 iterations.

FastText: The FastText vectors are like Word2Vec, except the word vector takes into account subword information: words are represented as n-grams, for which vectors are trained. The final word vector will then be the sum of its constituent n-gram vectors (Bojanowski et al., 2017). We trained a FastText space with the same settings

---

[5] Again taken from the ukWaC+Wackypedia corpus.

[6] `wacky.sslmit.unibo.it`

[7] `radimrehurek.com/gensim`

[8] `nlp.stanford.edu/projects/glove`

as the Word2Vec space (CBOW, minimum word frequency 50, dimensions 300, window 5, with 5 iterations), again using `gensim`.

4.2.1 Verb Matrices

In order to work with tensor-based models we had to represent verbs as matrices rather than as vectors. We generated verb tensors using two methods that have been used previously in the literature (Grefenstette and Sadrzadeh, 2011a; Kartsaklis and Sadrzadeh, 2014).

Relational: For each verb, its corresponding matrix is obtained by summing over the tensor product of the respective subject and object vectors of the verb (subjects and objects collected from the corpus):

$$\overline{verb} = \sum_i subj_i \otimes obj_i$$

Kronecker: For each verb, its corresponding matrix is obtained by taking the tensor product of the verb vector with itself:

$$\widetilde{verb} = \overrightarrow{verb} \otimes \overrightarrow{verb}$$

In the case of the count-based space, we trained verb matrices of dimensions 2000 × 2000, for the neural word embeddings the matrices had dimensions 300 × 300. We also experimented with the skip-gram extension of Maillard and Clark (2015) and the plausibility model of Polajnar et al. (2014a) but excluded the results because the obtained verb matrices were far below par.

4.3 Concrete Models

For the experiments, we had two main goals in mind: primarily, we wanted to verify that resolving ellipsis contributes to the performance of a compositional model. For this purpose we experimented with non-linear models, i.e. models that resolve the ellipsis (and thus use the verb and object resources twice) versus linear models, which do not resolve the ellipsis (and thus only use the verb and object once). Our second goal was to investigate whether amongst the models that resolve the ellipsis, the ones that did so in a tensor-based way, i.e. using tensors instead of vectors to represent the verbs, performed better than additive and multiplicative models, and how these compare to holistic sentence encoders. Hence, we considered three classes of models: linear vector models, non-linear vector models and tensor-based models.

Linear Vector Models: These models use every resource exactly once, following the pattern $\overrightarrow{w_1} \star \overrightarrow{w_2} ... \star \overrightarrow{w_n}$ for any sequence of words $w_1 w_2 ... w_n$. For an elliptical phrase "*subj verb obj and subj* does too*" it will compute the vector

$$\overrightarrow{subj} \star \overrightarrow{verb} \star \overrightarrow{obj} \star \overrightarrow{and} \star \overrightarrow{subj^*} \star \overrightarrow{does} \star \overrightarrow{too}$$

where $\star$ denotes either addition or multiplication.

Non-Linear Vector Models: Here, the assumption is that ellipsis is resolved but models do not respect word order. The meaning of "*subj verb obj and subj* does too*" now is

$$\overrightarrow{subj} \star \overrightarrow{verb} \star \overrightarrow{obj} \star \overrightarrow{subj^*} \star \overrightarrow{verb} \star \overrightarrow{obj}$$

Tensor-Based Models: These models all are assumed to resolve ellipsis and are based on various previous models (Grefenstette and Sadrzadeh, 2011b,a; Kartsaklis et al., 2012; Kartsaklis and Sadrzadeh, 2014). Essentially, the tensor-based meaning of "*subj verb obj and subj* does too*" is

$$T(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \star T(\overrightarrow{subj^*}, \overline{verb}, \overrightarrow{obj})$$

where T is a transitive model from (Milajevs et al., 2014) and $\star$ interprets the conjunction of the two subclauses. For the verb matrix we used either the relational verb or the Kronecker verb, and for $\star$ we tried both addition and multiplication. We did consider a model which simply adds or multiplies the second subject without duplicating the verb phrase, but it performed worse than non-linear addition and multiplication so we did not include it in this paper.

Sentence Encoders: To compare the mentioned compositional models with state of the art neural baselines, we carried out our experiments with a four types of holistic sentence encoders, that take arbitrary text as input and produce an embedding. To properly compare with the compositional models above, we gave three different inputs to the encoders: a baseline encoding (Base), a resolved encoding (Res), and an encoding without functional words (Abl), all as below:

Base: "*subj verb obj and subj* does too*"
Res: "*subj verb obj and subj* verb obj*"
Abl: "*subj verb obj subj**"

We used six concrete pretrained encoders, available online: 4800-dimensional embeddings from the Skip-Thought model[9], 300-dimensional embeddings from two Doc2Vec implementations

---

[9] `github.com/ryankiros/skip-thoughts`

(Lau and Baldwin, 2016)[10], 4096-dimensional embeddings from two InferSent encoders[11], and 512-dimensional embeddings from Universal Sentence Encoder[12].

5 Results

To validate the quality of the trained word spaces, we evaluate on several standard word similarity tasks: we used Rubenstein & Goodenough (RG, 1965), WordSim353 (WS353, 2001), Miller & Charles (MC, 1991), SimLex-999 (SL999, 2015), and the MEN dataset (Bruni et al., 2012). The results are displayed in Table 2, for the spaces described in the previous section.

	RG	WS353	MC	SL999	MEN
Count	.6081	.3583	.5455	.2593	.5527
Word2Vec	.8227	**.6983**	**.7682**	**.4026**	**.7810**
GloVe	**.8312**	.6180	.7377	.3902	.7727
FastText	.7724	.5461	.6961	.4021	.7683

Table 2: Spearman ρ scores on word similarity tasks.

Verb Disambiguation: Table 3 shows the results of the linear, non-linear and tensor-based models for this task, compared against a baseline in which only the verb vector or verb matrix is compared.

Our first observation is that generally, the highest performing models were tensor-based. The highest found correlation score was 0.5385 in the count based space for a tensor-based model (**CO** model above, Kronecker matrix, $\nabla = +$), with the Frobenius Additive model giving the second best result of 0.5263 (**FA** model above, Kronecker matrix, $\nabla = +$). For the neural spaces, the highest performing models were mostly tensor-based as well; they were always the Frobenius Additive (**FA**) model and the Frobenius Outer (**FO**) model, using the relational tensor and addition for the coordinator, except in the case of GloVe, where the Copy Object (**CO**) model was the second best. The only exception to this observation is the GloVe space, for which the baseline Vector Only model in fact has a higher correlation than any other model on that space.

	CB	W2V	GloVe	FT
Verb Only Vector	.4363	.2406	.4451	.2290
Verb Only Tensor	.3295	.4376	.3942	.3876
Add. Linear	.4416	.2728	.3046	.1409
Mult. Linear	3250	-.0123	.1821	.2928
Add. Non-Linear	.4448	.3275	.3262	.1399
Mult. Non-Linear	.5029	.2087	.2446	.0440
Best Tensor	**.5385**	.4621	.3688	.4937
2nd Best Tensor	.5263	.4544	.3581	.4652

Table 3: Spearman ρ scores for the ellipsis disambiguation experiment. CB: count based, W2V: Word2Vec, FT: FastText.

	D2V1	D2V2	ST	IS1	IS2	USE
Base	.1448	.2432	-.1932	.3471	**.3841**	.2693
Res	.2340	.2980	-.1720	.3436	.3373	.2770
Abl	.1899	.2423	-.1297	.3525	.3571	.2402

Table 4: Spearman ρ scores for the ellipsis disambiguation experiment. D2V1: Doc2Vec1, D2V2: Doc2Vec 2, ST: Skip-Thought, IS1: InferSent 1, IS2: InferSent 2, USE: Universal Sentence Encoder.

Our second observation is that the non-linear variants of the additive and multiplicative models (which resolve ellipsis but in a naive way) show an increased performance over the linear models (which do not resolve ellipsis). All of this holds for all the four vector spaces, except for the FastText space where the linear multiplicative model achieves significantly higher correlation (0.2928) than its non-linear counterpart (0.0440).

Overall, these results suggests that a logical resolving of ellipsis and further grammatical sensitivity benefits the performance of composition.

One interesting fact about our results is that the best compositional methods across the board were those that interpret the coordinator 'and' as addition; in set-theoretic semantics one interprets this coordinator as set intersection, which corresponds to multiplication rather than addition in a vectorial setting. We suggest that the feature intersection approach using multiplication leads to sparsity in the resulting vectorial representation, which then has a negative effect on the overall result. This would explain the case of FastText, since those vectors take into account subword information one would expect them to be more fine-grained and therefore conflate more of their features under multiplication.

The choice of verb matrix was mixed: for the count-based models the Kronecker matrix worked best, for the neural embeddings it was best to use

---

[10] `github.com/jhlau/doc2vec`

[11] `github.com/facebookresearch/InferSent`

[12] `tfhub.dev/google/universal-sentence-encoder`

the relational matrix.

In comparison, the sentence encoder results of Table 4 show the same trend that suggests that resolving ellipsis improves the quality of the embeddings: with the exception of the two InferSent encoders, the resolved models gave a higher correlation than their linear baseline. However, none of the encoder models come near the results achieved using the compositional models. Since the verb disambiguation dataset contains pairs of sentence that only differ in the verb, the task becomes very much grammar-oriented, and so we argue that the tensor-based models work better since they explicitly emphasise syntactic structure.

Sentence Similarity: For the extension of the KS2013 sentence similarity dataset, the results are shown in Table 5. We again wanted to see if resolving ellipsis benefits the compositional process. This was in general true, although we observed a different pattern to the previous experiment.

In all cases, except for the FastText space, we saw that non-linear models in fact perform better than their linear counterparts. But this time the best tensor-based models only outperformed addition for the count-based space: the best models scored 0.7410 and 0.7370 (respectively for the **FO** and **FA** models above, Kronecker matrix, $\nabla = \odot$). Both Word2Vec and GloVe worked best with a non-linear additive model, with Word2Vec achieving the overall highest correlation score of 0.7617, and GloVe achieving 0.7103. For FastText, the highest score of 0.7408 was achieved by linear addition. What is more, the multiplicative model did not benefit from a non-linear approach in the case of GloVe (from 0.3666 to 0.2439), and the additive model had a similar decline in performance for the count-based space (from 0.7000 to 0.6808) and FastText (0.7408 to 0.7387). We can see that for the neural word embeddings the additive models work best, with all of them seeing a drop in performance for the tensor-based models.

Again, the best count-based models use the Kronecker matrix whereas the neural models benefit the most from using the relational matrix. However, this time the best count-based models used multiplication for coordination, the neural models preferring addition.

The sentence encoders worked a lot better in the similarity task, with all non-linear resolved models outperforming the baseline model, and the InferSent model even outperforming non-linear addition on a Word2Vec space. We argue this is the case for two reasons: first, the similarity dataset is more diffuse than the verb disambiguation dataset since sentence pairs now differ for every word in the sentence, giving more opportunity to exploit semantic similarity rather than syntactic similarity. Second, the embeddings from the sentence encoder are larger (4096), allowing them to effectively store more information to benefit the similarity score.

	CB	**W2V**	**GloVe**	**FT**
Verb Only Vector	.4562	.5833	.4348	.6513
Verb Only Tensor	.3946	.5664	.4426	.5337
Add. Linear	.7000	.7258	.6964	.7408
Mult. Linear	.6330	.1302	.3666	.1995
Add. Non-Linear	.6808	**.7617**	.7103	.7387
Mult. Non-Linear	.7237	.3550	.2439	.4500
Best Tensor	.7410	.7061	.4907	.6989
2nd Best Tensor	.7370	.6713	.4819	.6871

Table 5: Spearman ρ scores for the ellipsis similarity experiment.

	D2V1	**D2V2**	**ST**	**IS1**	**IS2**	**USE**
Base	.5901	.6188	.5851	.7785	.7009	.6463
Res	.6878	.6875	.6039	**.8022**	.7486	.6791
Abl	.1840	.6599	.4715	.7815	.7301	.6397

Table 6: Spearman ρ scores for the ellipsis similarity experiment. D2V1: Doc2Vec1, D2V2: Doc2Vec 2, ST: Skip-Thought, IS1: InferSent 1, IS2: InferSent 2, USE: Universal Sentence Encoder.

Overall we conclude again that resolving ellipsis improves the performance of composition, but this time the InferSent sentence encoder seems to work best, followed by the non-linear additive compositional model on Word2Vec, with tensor-based models only performing well in a count-based space.

6 Conclusion

In this paper we experimented with vector space semantics for VP ellipsis, working with a large variety of compositional models. We created two new datasets and compared the performance of several compositional methods, both linear and non-linear, across four vector spaces, and against state of the art holistic sentence encoders.

Our main conclusion is that resolving ellipsis improves performance: non-linear models almost always performed better than linear ones in both a verb disambiguation and a sentence similarity task. The highest performance on the verb disambiguation task was given by a grammar-driven, tensor-based model in a count-based vector space, whereas for the similarity task, the highest performance was achieved by the InferSent sentence encoder, followed by a non-linear additive model on a Word2Vec space. Although the neural word embeddings and sentence encoders were largely outperformed on the disambiguation dataset that places more emphasis on syntactic structure than on semantic similarity, they generally performed better in the sentence similarity case, where the distinction between syntactic and semantic similarity is more diffuse.

Acknowledgments

The authors gratefully acknowledge the three anonymous reviewers for their valuable comments. Mehrnoosh Sadrzadeh is grateful to the Royal Society for an International Exchange Award *IE161631 - Dynamic Vector Semantics for Lambda Calculus Models of Natural Language* and discussion with Reinhard Muskens in this context. Gijs Wijnholds would like to express gratitude for support by a Queen Mary Principal Studentship, and the Theory group of the School of Electronic Engineering and Computer Science at Queen Mary University of London. Both authors would like to thank Ruth Kempson and Matthew Purver for many helpful discussions.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Stephen Clark. 2015. *Vector space models of lexical meaning*, chapter 16. John Wiley & Sons, Ltd.

Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. 2013. Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus. *Annals of Pure and Applied Logic*, 164(11):1079–1100.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Mary Dalrymple, Stuart M Shieber, and Fernando CN Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Robert Fiengo and Robert May. 1994. *Indices and identity*. MIT Press, Cambridge, Mass.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414. ACM.

Daniel Fried, Tamara Polajnar, and Stephen Clark. 2015. Low-Rank tensors for verbs in compositional distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 731–736.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-Step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language*

Semantics, pages 62–66. Association for Computational Linguistics.

Herman Hendriks and Paul Dekker. 1995. Links without locations. In *Proceedings of the Tenth Amsterdam Colloquium*, pages 339–358. Citeseer.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Gerhard Jäger. 2006. *Anaphora and type logical grammar*, volume 24. Trends in Logic. Springer Science & Business Media.

Dimitri Kartsaklis, Matthew Purver, and Mehrnoosh Sadrzadeh. 2016. Verb phrase ellipsis using Frobenius algebras in categorical compositional distributional semantics. *DSALT Workshop, European Summer School on Logic, Language and Information*.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*. Kyoto Japan.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of 24th International Conference on Computational Linguistics (COLING): Posters*. Mumbai, India.

Ruth Kempson, Ronnie Cann, Arash Eshghi, Eleni Gregoromichelaki, and Matthew Purver. 2015. Ellipsis. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*, 2nd edition, chapter 4. Wiley.

Douwe Kiela and Stephen Clark. 2014. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Yusuke Kubota and Robert Levine. 2017. Pseudogapping as pseudo-VP-ellipsis. *Linguistic Inquiry*, 48(2):213–257.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Jean Maillard and Stephen Clark. 2015. Learning adjective meanings with a tensor-based skip-gram model. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 327–331.

Jason Merchant. 2004. Fragments and ellipsis. *Linguistics and Philosophy*, 27:661–738.

Jason Merchant. 2010. Three types of ellipsis. *Context-Dependence, Perspective and Relativity*, 6:141.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Glyn Morrill and Oriol Valentín. 2015. Computational coverage of TLG: Nonlinearity. In *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*, volume 32, pages 51–63. EasyChair Publications.

Reinhard Muskens and Mehrnoosh Sadrzadeh. 2016. Context update for lambdas and vectors. In *International Conference on Logical Aspects of Computational Linguistics*, pages 247–254. Springer.

Reinhard Muskens and Mehrnoosh Sadrzadeh. 2017. Lambdas, vectors, and word meaning in context. In *Proceedings of the 21st Amsterdam Colloquium*, pages 65–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tamara Polajnar, Luana Fagarasan, and Stephen Clark. 2014a. Reducing dimensions of tensors in type-driven distributional semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1036–1046.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2014b. Using sentence plausibility to learn the semantics of transitive verbs. *arXiv preprint arXiv:1411.7942*.

Stephen G. Pulman. 1997. Higher order unification and the interpretation of focus. *Linguistics and Philosophy*, 20(1):73–115.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Anna Szabolcsi. 1987. Bound variables in syntax (are there any?). *Sixth Amsterdam Colloquium Proceedings*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2018. Classical copying versus quantum entanglement in natural language: The case of VP-ellipsis. In *EPTCS Proceedings of the Workshop on Compositional Approaches for Physics, NLP, and Social Sciences*, volume 283, pages 103–119.

Neural Finite State Transducers: Beyond Rational Relations

Chu-Cheng Lin
Computer Science Department
Johns Hopkins University
Baltimore, MD 21218, USA
`kitsing@cs.jhu.edu`

Hao Zhu
Dept. of Computer Science and Technology
Tsinghua University
Beijing, China
`zhuhao15@mails.tsinghua.edu.cn`

Matthew R. Gormley
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
`mgormley@cs.cmu.edu`

Jason Eisner
Computer Science Department
Johns Hopkins University
Baltimore, MD 21218, USA
`jason@cs.jhu.edu`

Abstract

We introduce neural finite state transducers (NFSTs), a family of string transduction models defining joint and conditional probability distributions over pairs of strings. The probability of a string pair is obtained by marginalizing over all its accepting paths in a finite state transducer. In contrast to ordinary weighted FSTs, however, each path is scored using an arbitrary function such as a recurrent neural network, which breaks the usual conditional independence assumption (Markov property). NFSTs are more powerful than previous finite-state models with neural features (Rastogi et al., 2016). We present training and inference algorithms for locally and globally normalized variants of NFSTs. In experiments on different transduction tasks, they compete favorably against seq2seq models while offering interpretable paths that correspond to hard monotonic alignments.

1 Introduction

Weighted finite state transducers (WFSTs) have been used for decades to analyze, align, and transduce strings in language and speech processing (Roche and Schabes, 1997; Mohri et al., 2008). They form a family of efficient, interpretable models with well-studied theory. A WFST describes a function that maps each string pair $(\mathbf{x}, \mathbf{y})$ to a weight—often a real number representing $p(\mathbf{x}, \mathbf{y})$ or $p(\mathbf{y} \mid \mathbf{x})$. The WFST is a labeled graph, in which each path $\mathbf{a}$ represents a sequence of operations that describes how some $\mathbf{x}$ and some $\mathbf{y}$ could be jointly generated, or how $\mathbf{x}$ could be edited into $\mathbf{y}$. Multiple paths for the same $(\mathbf{x}, \mathbf{y})$ pair correspond to different analyses (labeled alignments) of that pair.

However, WFSTs can only model certain functions, known as the rational relations (Berstel and Reutenauer, 1988).The weight of a path is simply the product of the weights on its arcs. This means that in a random path of the form $a \rightsquigarrow b \rightsquigarrow c$, the two subpaths $\rightsquigarrow$ are conditionally independent given their common state b: a Markov property.

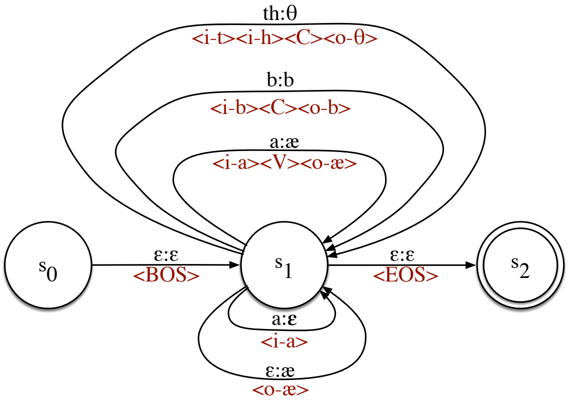


Figure 1: A marked finite-state transducer $\mathcal{T}$. Each arc in $\mathcal{T}$ is associated with input and output substrings, listed above the arcs in the figure. The arcs are not labeled with weights as in WFSTs. Rather, each arc is labeled with a sequence of *marks* (shown in brown) that featurize its qualities. The neural scoring model scores a path by scoring each mark in the context of all marks on the entire path. The example shown here is from the G2P application of §4.1; for space, only a few arcs are shown. ε represents the empty string.

In this paper, we propose neural finite state transducers (NFSTs), in which the weight of each path is instead given by some sort of neural network, such as an RNN. Thus, the weight of an arc can depend on the context in which the arc is used. By abandoning the Markov property, we lose exact dynamic programming algorithms, but we gain expressivity: the neural network can capture dependencies among the operations along a path. For example, the RNN might give higher weight to a path if it is "internally consistent": it might thus prefer to transcribe a speaker's utterance with a path that maps similar sounds in similar contexts to similar phonemes, thereby adapting to the speaker's accent.

Proceedings of NAACL-HLT 2019, pages 272–283
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

Consider a finite-state transducer $\mathcal{T}$ as in Figure 1 (see Appendix A for background). Using the composition operator $\circ$, we can obtain a new FST, $\mathbf{x} \circ \mathcal{T}$, whose accepting paths correspond to the accepting paths of $\mathcal{T}$ that have input string $\mathbf{x}$. Similarly, the accepting paths of $\mathcal{T} \circ \mathbf{y}$ correspond to the accepting paths of $\mathcal{T}$ that have output string $\mathbf{y}$. Finally, $\mathbf{x} \circ T \circ \mathbf{y}$ extracts the paths that have both properties. We define a joint probability distribution over $(\mathbf{x}, \mathbf{y})$ pairs by marginalizing over those paths:

$$p(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{a} \in \mathbf{x} \circ \mathcal{T} \circ \mathbf{y}} p(\mathbf{a}) = \frac{1}{Z(\mathcal{T})} \sum_{\mathbf{a} \in \mathbf{x} \circ \mathcal{T} \circ \mathbf{y}} \tilde{p}(\mathbf{a}) \quad (1)$$

where $\tilde{p}(\mathbf{a})$ is the weight of path $\mathbf{a}$ and $Z(\mathcal{T}) = \sum_{\mathbf{a} \in \mathcal{T}} \tilde{p}(\mathbf{a})$ is a normalization constant.

We define $\tilde{p}(\mathbf{a}) \triangleq \exp G_{\boldsymbol{\theta}}(\mathbf{a})$ with $G_{\boldsymbol{\theta}}(\mathbf{a})$ being some parametric scoring function. In our experiments, we will adopt a fairly simple left-to-right RNN architecture (§2.2), but one could easily substitute fancier architectures. We will also consider defining $G_{\boldsymbol{\theta}}$ by a *locally normalized* RNN that ensures $Z(\mathcal{T}) = 1$.

In short, we use the finite-state transducer $\mathcal{T}$ to compactly define a set of *possible* paths $\mathbf{a}$. The number of paths may be exponential in the size of $\mathcal{T}$, or infinite if $\mathcal{T}$ is cyclic. However, in contrast to WFSTs, we abandon this combinatorial structure in favor of neural nets when defining the probability distribution over $\mathbf{a}$. In the resulting marginal distribution $p(\mathbf{x}, \mathbf{y})$ given in equation (1), the path $\mathbf{a}$ that *aligns* $\mathbf{x}$ and $\mathbf{y}$ is a latent variable. This is also true of the resulting conditional distribution $p(\mathbf{y} \mid \mathbf{x})$.

We explore training and inference algorithms for various classes of NFST models (§3). Classical WFSTs (Mohri et al., 2008) and BiRNN-WFSTs (Rastogi et al., 2016) use restricted scoring functions and so admit exact dynamic programming algorithms. For general NFSTs, however, we must resort to approximate computation of the model's training gradient, marginal probabilities, and predictions. In this paper, we will use sequential importance sampling methods (Lin and Eisner, 2018), leaving variational approximation methods to future work.

Defining models using FSTs has several benefits:

Output-sensitive encoding Currently popular models of $p(\mathbf{y} \mid \mathbf{x})$ used in machine translation and morphology include seq2seq (Sutskever et al., 2014), seq2seq with attention (Bahdanau et al., 2015; Luong et al., 2015), the Transformer (Vaswani et al., 2017). These models first encode $\mathbf{x}$ as a vector or sequence of vectors, and then condition the generation of $\mathbf{y}$ on this encoding. The vector is determined from $\mathbf{x}$ only. This is also the case in the BiRNN-WFST (Rastogi et al., 2016), a previous finite-state model to which we compare. By contrast, in our NFST, the state of the RNN as it reads and transduces the second half of $\mathbf{x}$ is influenced by the first halves of both $\mathbf{x}$ *and* $\mathbf{y}$ and their alignment.

Inductive bias Typically, a FST is constructed with domain knowledge (possibly by compiling a regular expression), so that its states reflect interpretable properties such as syllable boundaries or linguistic features. Indeed, we will show below how to make these properties explicit by "marking" the FST arcs. The NFST's path scoring function then sees these marks and can learn to take them into account. The NFST also inherits any hard constraints from the FST: if the FST omits all $(\mathbf{x}, \mathbf{y})$ paths for some "illegal" $\mathbf{x}, \mathbf{y}$, then $p(\mathbf{x}, \mathbf{y}) = 0$ for any parameter vector $\boldsymbol{\theta}$ (a "structural zero").

Interpretability Like a WFST, an NFST can "explain" why it mapped $\mathbf{x}$ to $\mathbf{y}$ in terms of a latent path $\mathbf{a}$, which specifies a hard monotonic labeled alignment. The posterior distribution $p(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$ specifies which paths $\mathbf{a}$ are the best explanations (e.g., Table 5).

We conduct experiments on three tasks: grapheme-to-phoneme, phoneme-to-grapheme, and action-to-command (Bastings et al., 2018). Our results on these datasets show that our best models can improve over neural seq2seq and previously proposed hard alignment models.

2 Neuralizing Finite-State Transducers

2.1 Neuralized FSTs

An NFST is a pair $(\mathcal{T}, G_{\boldsymbol{\theta}})$, where $\mathcal{T}$ is an unweighted FST with accepting paths $\mathcal{A}$ and $G_{\boldsymbol{\theta}} : \mathcal{A} \to \mathbb{R}$ is a function that **scores** these paths. As explained earlier, we then refer to $\tilde{p}(\mathbf{a}) = \exp G_{\boldsymbol{\theta}}(\mathbf{a})$ as the **weight** of path $\mathbf{a} \in \mathcal{A}$. A **weighted relation** between input and output strings is given by $\tilde{p}(\mathbf{x}, \mathbf{y})$, which is defined to be the total weight of all paths with input string $\mathbf{x} \in \Sigma^*$ and output string $\mathbf{y} \in \Delta^*$, where where Σ and Δ are the input and output alphabets of $\mathcal{T}$. The real parameter vector $\boldsymbol{\theta}$ can be adjusted to obtain different weighted relations. We

Model	Training Algorithms	Long-Term Output-Output Dependency	Left-to-Right Factorization
WFSTs	Dynamic Programming	✗	✓
BiRNN-WFSTs	Dynamic Programming	✗	✓
Local NFSTs	Importance Sampling	✓	✓
Global NFSTs	Importance Sampling	✓	✗

Table 1: Comparison between WFSTs, BiRNN-WFSTs (Rastogi et al., 2016), and NFSTs.

can normalize $\tilde{p}$ to get a probability distribution as shown in equation (1).

2.2 A basic scoring architecture

Weighted FST. A WFST over the $(+, \times)$ semiring can be regarded as the special case in which $G_{\boldsymbol{\theta}}(\mathbf{a}) \triangleq \sum_{t=1}^{|\mathbf{a}|} g_{\boldsymbol{\theta}}(a_t)$. This is a sum of scores assigned to the arcs in $\mathbf{a} = a_1 a_2 \cdots$.

Marked FST. Our innovation is to allow the arcs' scores to depend on their context in the path. Now $\boldsymbol{\theta}$ no longer associates a fixed score with each arc. Rather, we assume that each arc a in the FST comes labeled with a sequence of **marks** from a **mark alphabet** Ω, as illustrated in Figure 1. The marks reflect the FST constructor's domain knowledge about what arc a does (see §4.2 below). We now define $G_{\boldsymbol{\theta}}(\mathbf{a}) = G_{\boldsymbol{\theta}}(\boldsymbol{\omega}(\mathbf{a}))$, where $\boldsymbol{\omega}(\mathbf{a}) = \boldsymbol{\omega}(a_1)\boldsymbol{\omega}(a_2) \cdots \in \Omega^*$ is the concatenated sequence of marks from the arcs along path $\mathbf{a}$.

It is sometimes helpful to divide marks into different classes. An arc can be regarded as a possible "edit" that aligns an input substring with an output substring in the context of transitioning from one FST state to another. The arc's **input marks** describe its input substring, its **output marks** describe its output substring, and the remaining marks may describe other properties of the arc's aligned input-output pair or the states that it connects.

Recall that an FST encodes domain knowledge. Its paths represent alignments between input and output strings, where each alignment specifies a segmentation of $\mathbf{x}$ and $\mathbf{y}$ into substrings labeled with FST states. Decorating the arcs with marks furnishes the path scoring model with domain-specific information about the alignments.

RNN scoring. If $\boldsymbol{\theta}$ merely associated a fixed score with each mark, then the marked FST would be no more powerful than the WFST. To obtain contextual mark scores as desired, one simple architecture is a recurrent neural network:

$$G_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \triangleq \sum_{t=1}^{|\boldsymbol{\omega}|} g_{\boldsymbol{\theta}}(\mathbf{s}_{t-1}, \omega_t) \tag{2}$$

$$\mathbf{s}_t = f_{\boldsymbol{\theta}}(\mathbf{s}_{t-1}, \omega_t), \text{ with } \mathbf{s}_0 = \mathbf{0} \tag{3}$$

where $\mathbf{s}_{t-1} \in \mathbb{R}^d$ is the hidden state vector of the network after reading $\omega_1 \cdots \omega_{t-1}$. The $g_{\boldsymbol{\theta}}$ function defines the score of reading ω_t in this left context, and $f_{\boldsymbol{\theta}}$ defines how doing so updates the state.

In our experiments, we chose $f_{\boldsymbol{\theta}}$ to be the GRU state update function (Cho et al., 2014). We defined $g_{\boldsymbol{\theta}}(\mathbf{s}, \omega_t) \triangleq (\mathbf{W}\mathbf{s} + \mathbf{b})^\top \mathrm{emb}(\omega_t)$. The parameter vector $\boldsymbol{\theta}$ specifies the GRU parameters, $\mathbf{W}$, $\mathbf{b}$, and the mark embeddings $\mathrm{emb}(\omega)$.

One could easily substitute much fancier architectures, such as a stacked BiLSTM with attention (Tilk and Alumäe, 2016), or a Transformer (Vaswani et al., 2017).

2.3 Partitioned hidden vectors

In hopes of improving the inductive bias of the learner, we partitioned the hidden state vector into three sub-vectors: $\mathbf{s}_t = [\mathbf{s}_t^{\mathrm{a}}; \mathbf{s}_t^{\mathrm{x}}; \mathbf{s}_t^{\mathrm{y}}]$. The mark scoring function $f_{\boldsymbol{\theta}}(\mathbf{s}_{t-1}, \omega_t)$ was as before, but we restricted the form of $g_{\boldsymbol{\theta}}$, the state update function. $\mathbf{s}_t^{\mathrm{a}}$ encodes *all* past marks and depends on the *full* hidden state so far: $\mathbf{s}_t^{\mathrm{a}} = g_{\boldsymbol{\theta}}^{\mathrm{a}}(\mathbf{s}_{t-1}, \omega_t)$. However, we make $\mathbf{s}_t^{\mathrm{x}}$ encode only the sequence of past *input* marks, ignoring all others. Thus, $\mathbf{s}_t^{\mathrm{x}} = g_{\boldsymbol{\theta}}^{\mathrm{x}}(\mathbf{s}_{t-1}^{\mathrm{x}}, \omega_t)$ if ω_t is an input mark, and $\mathbf{s}_t^{\mathrm{x}} = \mathbf{s}_{t-1}^{\mathrm{x}}$ otherwise. Symmetrically, $\mathbf{s}_t^{\mathrm{y}}$ encodes only the sequence of past *output* marks. This architecture is somewhat like Dyer et al. (2016), which also uses different sub-vectors to keep track of different aspects of the history.

2.4 Local normalization

A difficulty with the general model form in equation (1) is that the normalizing constant $Z(\mathcal{T}) = \sum_{\mathbf{a} \in \mathcal{T}} \tilde{p}(\mathbf{a})$ must sum over a large set of paths—in fact, an infinite set if $\mathcal{T}$ is cyclic. This sum may diverge for some values of the parameter vector $\boldsymbol{\theta}$, which complicates training of the model (Dreyer, 2011). Even if the sum is known to converge, it is in general intractable to compute it exactly. Thus, estimating the gradient of $Z(\mathcal{T})$ during training involves approximate sampling from the typically high-entropy distribution $p(\mathbf{a})$. The resulting estimates are error-prone because the sample size tends to be too small and the approximate sampler is biased.

A standard solution in the WFST setting (e.g. Cotterell et al., 2014) is to use a locally normalized model, in which $Z(\mathcal{T})$ is guaranteed to be 1.[1] The big summation over all paths $\mathbf{a}$ is replaced by small summations—which can be computed explicitly—over just the outgoing edges from a given state.

Formally, we define the unnormalized score of arc a_i in the context of path $\mathbf{a}$ in the obvious way, by summing over the contextual scores of its marks:

$$\tilde{g}_{\boldsymbol{\theta}}(a_i) \triangleq \sum_{t=j+1}^{k} g_{\boldsymbol{\theta}}(\mathbf{s}_{t-1}, \omega_t) \tag{4}$$

where $j = |\boldsymbol{\omega}(a_1) \cdots \boldsymbol{\omega}(a_{i-1})|$ and $k = |\boldsymbol{\omega}(a_1) \cdots \boldsymbol{\omega}(a_i)|$. Its normalized score is then

$$g_{\boldsymbol{\theta},\mathcal{T}}(a_i) \triangleq \log\left(\exp \tilde{g}_{\boldsymbol{\theta}}(a_i) / \sum\nolimits_{a'} \exp \tilde{g}_{\boldsymbol{\theta}}(a')\right)$$

where a' ranges over all arcs in $\mathcal{T}$ (including a_i itself) that emerge from the same state as a_i does. We can now score the paths in $\mathcal{T}$ using

$$G_{\boldsymbol{\theta},\mathcal{T}}(\mathbf{a}) = \sum_{i=1}^{|\mathbf{a}|} g_{\boldsymbol{\theta},\mathcal{T}}(a_i) \tag{5}$$

This gives rise to a proper probability distribution $p(\mathbf{a}) \triangleq \tilde{p}(\mathbf{a}) = \exp G_{\boldsymbol{\theta},\mathcal{T}}(\mathbf{a})$ over the paths of $\mathcal{T}$. No global normalization constant is necessary. However, note that the scoring function now requires $\mathcal{T}$ as an extra subscript, because it is necessary when scoring $\mathbf{a}$ to identify the competitors in $\mathcal{T}$ of each arc a_i. Thus, when $p(\mathbf{x}, \mathbf{y})$ is found as usual by summing up the probabilities of all paths in $\mathbf{x} \circ \mathcal{T} \circ \mathbf{y}$, each path is still scored using its arcs' competitors from $\mathcal{T}$. This means that each state in $\mathbf{x} \circ \mathcal{T} \circ \mathbf{y}$ must record the state in $\mathcal{T}$ from which it was derived.

3 Sampling, Training, and Decoding

3.1 Sampling from conditioned distributions with amortized inference

Many algorithms for working with probability distributions—including our training and decoding algorithms below—rely on conditional sampling. In general, we would like to sample a path of $\mathcal{T}$ given the knowledge that its input and output strings fall into sets X and Y respectively.[2] If X and Y are regular languages, this is equivalent to defining $\mathcal{T}' = X \circ \mathcal{T} \circ Y$ and sampling from

$$p(\mathbf{a} \mid \mathcal{T}') \triangleq \frac{\tilde{p}(\mathbf{a})}{\sum_{\mathbf{a}' \in \mathcal{T}'} \tilde{p}(\mathbf{a}')}, \tag{6}$$

Due to the nonlinearity of $G_{\boldsymbol{\theta}}$, the denominator of equation (6) is generally intractable. If $\mathcal{T}'$ is cyclic, it cannot even be computed by brute-force enumeration. Thus, we fall back on normalized importance sampling, directly adopting the ideas of Lin and Eisner (2018) in our more general FST setting. We employ a proposal distribution q:

$$\begin{aligned} p(\mathbf{a} \mid \mathcal{T}') &= \mathbb{E}_{\mathbf{a} \sim q}\left[\frac{p(\mathbf{a} \mid \mathcal{T}')}{q(\mathbf{a})}\right], \qquad (7) \\ &\approx \sum_{m=1}^{M} \frac{\tilde{p}(\mathbf{a}^{(m)})}{q(\mathbf{a}^{(m)}) \cdot \hat{Z}} \cdot \mathbb{I}(\mathbf{a} = \mathbf{a}^{(m)}) \\ &= \hat{p}(\mathbf{a} \mid \mathcal{T}'), \end{aligned}$$

where $\hat{Z} = \sum_{m'=1}^{M} \frac{\tilde{p}(\mathbf{a}^{(m')})}{q(\mathbf{a}^{(m')})}$, and q is a locally normalized distribution over paths $\mathbf{a} \in \mathcal{T}'$. In this paper we further parametrize q as

$$q_{\boldsymbol{\phi}}(\mathbf{a}; \mathcal{T}') = \prod_{t=1}^{T} q_t(a_t \mid \mathbf{a}_{1 \ldots t-1}; \boldsymbol{\phi}, \mathcal{T}'), \tag{8}$$

$$q_t(a \mid \mathbf{a}_{:t-1}; \boldsymbol{\phi}, \mathcal{T}') \propto \exp(g(\mathbf{s}_{t-1}, a_t; \boldsymbol{\theta}, \mathcal{T}) + C_{\boldsymbol{\phi}}),$$

where $C_{\boldsymbol{\phi}} \triangleq C(\mathbf{s}'_t, X, Y, \boldsymbol{\phi}) \in \mathbb{R}$, $\mathbf{s}'_t \triangleq f(\mathbf{s}_{t-1}, \boldsymbol{\omega}(a))$ is a *compatibility function* that is typically modeled using a neural network. In this paper, one the following three cases are encountered:

- $X = \mathbf{x}$, is a string, and $Y = \Delta^*$: in this case $\mathcal{T}' = \mathbf{x} \circ \mathcal{T}$. We let $C_{\boldsymbol{\phi}} = C^{\mathrm{x}}(\mathbf{s}'_t, \mathrm{RNN}^{\mathrm{x}}(\mathbf{x}, i, \boldsymbol{\phi}); \boldsymbol{\phi})$, where i is the length of the input prefix in $\mathbf{a}_{1 \ldots t} . a$, $\mathrm{RNN}^{\mathrm{x}}(\mathbf{x}, i, \boldsymbol{\phi})$ is the hidden state of the i-th position after reading $\mathbf{x}$ (not $\mathbf{a}$ nor $\boldsymbol{\omega}$) backwards, and $C^{\mathrm{x}}(\cdot, \cdot)$ is a feed-forward network that takes the concatenated vector of all arguments, and outputs a real scalar. We describe the parametrization of C^{x} in Appendix C.1.

---

[1] Provided that every state in $\mathcal{T}$ is co-accessible, i.e., has a path to a final state.

[2] When X or Y is larger than a single string, it is commonly all of Σ^* or Δ^* respectively, in which case conditioning on it gives no information.

- $X = \Sigma^*$, and $Y = \mathbf{y}$ is a string: in this case $\mathcal{T}' = \mathcal{T} \circ \mathbf{y}$. We let $C_{\boldsymbol{\phi}} = C^{\mathrm{y}}(\mathbf{s}'_t, \mathrm{RNN}^{\mathrm{y}}(\mathbf{y}, j, \boldsymbol{\phi}); \boldsymbol{\phi})$, where j is the length of the output prefix in $\mathbf{a}_{1\ldots t}.a$, and $\mathrm{RNN}^{\mathrm{y}}, C^{\mathrm{y}}$ are similarly defined as in $\mathrm{RNN}^{\mathrm{x}}$ and C^{x}.
- X and Y are both strings — $X = \mathbf{x}, Y = \mathbf{y}$: in this case we let $C_{\boldsymbol{\phi}} = C^{\mathrm{xy}}(\mathbf{s}'_t, \mathrm{RNN}^{\mathrm{x}}(\mathbf{x}, i, \boldsymbol{\phi}), \mathrm{RNN}^{\mathrm{y}}(\mathbf{y}, j, \boldsymbol{\phi}); \boldsymbol{\phi})$.

Given a path prefix $\mathbf{a}_{:t-1}$, $q_t(a \mid \mathbf{a}_{:t-1}; \boldsymbol{\phi}, \mathcal{T}')$ is defined over arcs a such that $\mathbf{a}_{:t-1}.a$ is a valid path prefix in $\mathcal{T}'$. To optimize $\boldsymbol{\phi}$ with regard to $q_{\boldsymbol{\phi}}$, we follow (Lin and Eisner, 2018) and seek to find $\boldsymbol{\phi}^* = \mathrm{argmin}_{\boldsymbol{\phi}} \mathrm{KL}[\hat{p} || q_{\boldsymbol{\phi}}]$, where $\hat{p}$ is the approximate distribution defined in equation (7), which is equivalent to maximizing the log-likelihood of $q_{\boldsymbol{\phi}}(\mathbf{a})$ when $\mathbf{a}$ is distributed according to the approximation $\hat{p}$.

3.2 Training

In this paper, we consider joint training. The loss function of our model is defined as the negative log joint probability of string pair $(\mathbf{x}, \mathbf{y})$:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\log p(\mathbf{x}, \mathbf{y}) = -\log \sum_{\mathbf{a} \in \mathbf{x} \circ \mathcal{T} \circ \mathbf{y}} p(\mathbf{a}). \tag{9}$$

Since p is an exponential family distribution, the gradients of $\mathcal{L}$ can be written as (Bishop, 2006)

$$\nabla \mathcal{L}(\mathbf{x}, \mathbf{y}) = -\mathbb{E}_{\mathbf{a} \sim p(\cdot \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})}[\nabla \log p(\mathbf{a})], \tag{10}$$

where $p(\cdot \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})$ is a conditioned distribution over paths. Computing equation (10) requires sampling from $p(\cdot \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})$, which, as we discuss in §3.1, is often impractical. We therefore approximate it with

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}, \mathbf{y}) &= -\mathbb{E}_{\mathbf{a} \sim p(\cdot \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})}[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{a})] \\ &\approx -\mathbb{E}_{\mathbf{a} \sim \hat{p}(\cdot \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})}[\nabla_{\boldsymbol{\theta}} \log p(\mathbf{a})] \qquad (11) \\ &= -\sum_{m=1}^{M} w^{(m)} \nabla_{\boldsymbol{\theta}} G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)}), \qquad (12) \end{aligned}$$

where q is a proposal distribution parametrized as in equation (8) (discussed in §3.1,) $\mathbf{a}^{(1)} \ldots \mathbf{a}^{(M)} \sim q$ are i.i.d. samples of paths in $\mathbf{x} \circ \mathcal{T} \circ \mathbf{y}$, and $w^{(m)}$ is the **importance weight** of the m-th sample satisfying $w^{(m)} \propto \frac{\exp G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)})}{q(\mathbf{a}^{(m)})}, \sum_{m=1}^{M} w^{(m)} = 1$. Pseudocode for calculating equation (12) is listed in Algorithm 1.

Algorithm 1 Compute approximate gradient for updating $G_{\boldsymbol{\theta}}$

```
Require: $G_{\boldsymbol{\theta}} : \mathcal{A} \to \mathbb{R}$ is an NFST scoring function, $q$ is a distribution over paths, $M \in \mathbb{N}$ is the sample size
function GET-GRADIENT($G_{\boldsymbol{\theta}}$, $M$, $q$)
    for $m$ in $1 \ldots M$ do
        $\mathbf{a}^{(m)} \sim q$
        $\tilde{w}^{(m)} \leftarrow \frac{\exp G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)})}{q(\mathbf{a})}$
    end for
    $\hat{Z} \leftarrow \sum_{m=1}^{M} \tilde{w}^{(m)}$
    for $m$ in $1 \ldots M$ do
        $w^{(m)} \leftarrow \frac{\tilde{w}^{(m)}}{\hat{Z}}$
    end for
    return $-\sum_{m=1}^{M} w^{(m)} \nabla_{\boldsymbol{\theta}} G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)})$
end function
```

3.3 Decoding most probable strings

Besides finding good *paths* in a conditioned distribution as we discuss in §3.1, we are also often interested in finding good *output strings*, which is conventionally referred to as the *decoding* problem, which we define to be finding the best output string $\mathbf{y}^* \triangleq \mathrm{argmax}_{\mathbf{y} \in L(Y)} p_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}')$, where

$$p_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}') \triangleq \frac{\sum_{\mathbf{a} \in \mathcal{T}' \circ \mathbf{y}} \tilde{p}(\mathbf{a})}{\sum_{\mathbf{a}' \in \mathcal{T}'} \tilde{p}(\mathbf{a}')}. \tag{13}$$

$\hat{\mathbf{y}}^* \triangleq \mathrm{argmax}_{\mathbf{y}} \hat{P}_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}')$ is a consistent estimator of $\mathbf{y}^*$, which can directly be used to find the best string. However, making this estimate accurate might be expensive: it requires sampling many paths in the machine $\mathcal{T}'$, which is usually cyclic, and therefore has infinitely many more paths, than $\mathcal{T}' \circ \mathbf{y}_k$, which has finitely many paths when A is acyclic. On the other hand, for the task of finding the best string among a pool candidates, we do not need to compute (or approximate) the denominator in equation (13), since

$$\mathbf{y}^* = \underset{\mathbf{y} \in L(Y)}{\mathrm{argmax}} \sum_{\mathbf{a} \in \mathcal{T}' \circ \mathbf{y}} \tilde{p}(\mathbf{a}). \tag{14}$$

As in the case for paths, the language $L(Y)$ is usually infinitely large. However given an output candidate $\mathbf{y}_k \in L' \subseteq L(Y)$, we can approximate the summation in equation (14) using importance sampling:

$$\sum_{\mathbf{a} \in \mathcal{T}' \circ \mathbf{y}_k} \tilde{p}(\mathbf{a}) = \mathbb{E}_{\mathbf{a} \sim q(\cdot \mid \mathcal{T}' \circ \mathbf{y}_k)}\left[\frac{\tilde{p}(\mathbf{a})}{q(\mathbf{a} \mid \mathcal{T}' \circ \mathbf{y}_k)}\right], \tag{15}$$

Algorithm 2 Training procedure for $G_{\boldsymbol{\theta}}$. See Appendix C.2 for implementation details.

```
Require: $(\mathcal{T}, G_{\boldsymbol{\theta}})$ is an NFST, $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1) \ldots (\mathbf{x}_{|\mathcal{D}|}, \mathbf{y}_{|\mathcal{D}|})\}$ is the training dataset, LR : $\mathbb{N} \to \mathbb{R}$ is a
    learning rate scheduler, $\boldsymbol{\theta}_0$ are the initial parameters of $G_{\boldsymbol{\theta}}$, $M$ is a given sample size, maxEpoch $\in \mathbb{N}$
    is the number of epochs to train for
procedure TRAIN($\mathcal{T}$, $G_{\boldsymbol{\theta}}$, $\mathcal{D}$, LR, $\boldsymbol{\theta}_0$, $M$, maxEpochs)
    for epoch $\in [1 \ldots \mathrm{maxEpochs}]$ do
        for $(\mathbf{x}_i, \mathbf{y}_i) \in \mathrm{shuffle}(\mathcal{D})$ do
            $\mathcal{T}' \leftarrow \mathbf{x}_i \circ \mathcal{T} \circ \mathbf{y}_i$
            Construct distribution $q(\cdot \mid \mathcal{T}')$ according to equation (8)
            $\mathbf{u} \leftarrow$ GET-GRADIENT$(G_{\boldsymbol{\theta}}, M, q)$ (listed in Algorithm 1)
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mathrm{LR}(\mathrm{epoch}) \times \mathbf{u}$
            (Optional) update the parameters of $q(\cdot \mid \mathcal{T}')$.
        end for
    end for
end procedure
```

where $q(\cdot \mid \mathcal{T}' \circ \mathbf{y}_k)$ is a proposal distribution over paths in $\mathcal{T}' \circ \mathbf{y}_k$. In this paper we parametrize $q(\cdot \mid \mathcal{T}' \circ \mathbf{y}_k)$ following the definition in equation (8). When L' is finitely large, we reduce the decoding task into a reranking task.

To populate L', one possibility is to marginalize over paths in the approximate distribution $\hat{p}(\mathbf{a} \mid \mathcal{T}')$ discussed in §3.1 to obtain an estimate $\hat{p}_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}')$, and use its support as L'. Note that it's possible to populate the candidate pool in other ways, each with its advantages and drawbacks: for example, one can use a top-k path set from a weighted (Markovian) FST. This approach guarantees exact computation, and the pool quality would no longer depend on the qualities of the smoothing distribution $q_{\boldsymbol{\phi}}$. However it is also a considerably much weaker model and may yield uninspiring candidates. In the common case where the conditioned machine $\mathcal{T}' = X \circ \mathcal{T} \circ Y$ has $X = \mathbf{x} \in \Sigma^*$ as the input string, and Y is the universal acceptor that accepts Δ^*, one can obtain a candidate pool from seq2seq models: seq2seq models can capture long distance dependencies between input and output strings, and are typically fast to train and decode from. However they are not applicable in the case where $L(Y) \neq \Delta^*$. Experimental details of decoding are further discussed in §4.3.

4 Experiments

Our experiments mainly aim to: (1) show the effectiveness of NFSTs on transduction tasks; (2) illustrate that how prior knowledge can be introduced into NFSTs and improve the performance; (3) demonstrate the interpretability of our model. Throughout, we experiment on three tasks: (i) grapheme-to-phoneme, (ii) phoneme-to-grapheme, and (iii) actions-to-commands. We compare with competitive string transduction baseline models in these tasks.

4.1 Tasks and datasets

We carry out experiments on three string transduction tasks:

Grapheme-to-phoneme and phoneme-to-grapheme (G2P/P2G) refer to the transduction between words' spelling and phonemic transcription. English has a highly irregular orthography (Venezky, 2011), which necessitates the use of rich models for this task. We use a portion of the standard CMUDict dataset: the Sphinx-compatible version of CMUDict (Weide, 2005). As for metrics, we choose widely used exact match accuracy and edit distance.

Action-to-command (A2C) refers to the transduction between an action sequence and imperative commands. We use NACS (Bastings et al., 2018) in our experiment. As for metrics, we use exact match accuracy (EM). Note that the in A2C setting, a given input can yield different outputs, e.g. `I_JUMP I_WALK I_WALK` corresponds to both "jump and walk twice" and "walk twice after jump". NACS is a finite set of action-command pairs; we consider a predicted command to be correct if it is in the finite set and its corresponding actions is exactly the input. We evaluate on the *length* setting proposed by Bastings et al. (2018), where we train on shorter sequences and evaluate on longer sequences.

4.2 FST designs

NFSTs require an unweighted FST $\mathcal{T}$ which defines a scaffold for the relation it recognizes. In this paper we experiment with two versions of $\mathcal{T}$: the first is a simple 'general' design $\mathcal{T}_0$, which contains only three states $s_{\{0,1,2\}}$, where the only arc between q_0 and q_1 consumes the mark `<BOS>`; and the only arc between q_1 and q_2 consumes the mark `<EOS>`. $\mathcal{T}_0$ has exactly one accepting state, which is q_2. To ensure that $\mathcal{T}_0$ defines relation for all possible string pairs $(\mathbf{x}, \mathbf{y}) \in \Sigma^* \times \Delta^*$, we add all arcs of the form $a = (s_1, s_1, \boldsymbol{\omega}, \sigma, \delta), \forall(\sigma, \delta) \in \Sigma \times \Delta$ to $\mathcal{T}$.

To recognize transduction rules defined in the Wikipedia English IPA Help page, we define $\mathcal{T}_{\text{IPA}}$, which has all states and arcs of $\mathcal{T}_0$, and additional states and arcs to handle multi-grapheme and multi-phoneme transductions defined in the IPA Help:[3] for example, the transduction `th` → θ is encoded as two arcs $(s_1, s_3, \boldsymbol{\omega}, \mathtt{t}, \theta)$ and $(s_3, s_1, \boldsymbol{\omega}, \mathtt{h}, \varepsilon)$. Because of the lack of good prior knowledge that can be added to A2C experiments, we only use general FSTs in those experiments for such experiments. Nor do we encode special marks that we are going to introduce below.[4]

4.2.1 Design of mark sequences

As with regular WFSTs, the arcs can often be hand-engineered to incorporate prior knowledge. Recall that as we describe in §2.2, each arc is associated with a mark sequence. In this paper, we will always derive the mark sequence on an arc $a = (s', s, \boldsymbol{\omega}', \sigma, \delta)$ of the transducer $\mathcal{T}$ as $\boldsymbol{\omega} = [\sigma, \boldsymbol{\omega}', \delta, s]$, where $\boldsymbol{\omega}' \in \Omega^*$ can be engineered to reflect FST- and application-specific properties of a path, such as the IPA Help list we mentioned earlier. One way to encode such knowledge into mark sequences is to have special mark symbols in mark sequences for particular transductions. In this paper we experiment with two schemes of marks:

- **IPA Help (IPA).** We define the IPA mark $\omega_{\text{IPA}} = \{\mathtt{C} \mid \mathtt{V}\}$, where the symbol `C` indicates that this arc is part of a transduction rule listed in the consonant section of the Wikipedia English IPA Help page. Similarly, the mark `V` indicates that the transduction rule is listed in the vowel section.
- **Phoneme Classes (PHONE).** We define PHONE marks $\omega_{\text{PHONE}} = \Phi(\delta)$, where Φ is a lookup function that returns the phoneme class of δ defined by the CMUDict dataset.[5]

In this paper we experiment with the following three FST and mark configurations for G2P/P2G experiments:

- **-IPA-PHONE** in which case $\boldsymbol{\omega}' = \varnothing$ for all arcs. $\mathcal{T} = \mathcal{T}_0$.
- **+IPA-PHONE** in which case $\boldsymbol{\omega}' = [\omega_{\text{IPA}}]$ when the transduction rule is found in the IPA Help list, otherwise $\boldsymbol{\omega}' = \varnothing$. $\mathcal{T} = \mathcal{T}_{\text{IPA}}$.
- **+IPA+PHONE** in which case $\boldsymbol{\omega}' = [\omega_{\text{IPA}}\omega_{\text{PHONE}}]$ when the transduction rule is found in the IPA Help list, otherwise $\boldsymbol{\omega}' = [\omega_{\text{PHONE}}]$. $\mathcal{T} = \mathcal{T}_{\text{IPA}}$.

As we said earlier, we only use $\mathcal{T} = \mathcal{T}_0$ with no special marks for A2C experiments. Experimental results on these different configurations are in §5.3.

4.3 Decoding methods

We experiment with the following methods to decode the most probable strings:

- **Approximate Posterior (AP).** We approximate the posterior distribution over output strings $\hat{p}_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}')$, and pick $\hat{\mathbf{y}}^* = \operatorname{argmax}_{\mathbf{y}} \hat{p}_{\mathbf{Y}}(\mathbf{y} \mid \mathcal{T}')$ as the output.
- **Reranking AP.** As we discuss in §3.3, improving $\hat{\mathbf{y}}^*$ by taking more path samples in $\mathcal{T}'$ may be expensive. The reranking method uses the support of $\hat{p}_{\mathbf{Y}}$ as a candidate pool L', and for each $\mathbf{y}_k \in L'$ we estimate equation (15) using path samples in $\mathcal{T}' \circ \mathbf{y}_k$.
- **Reranking External.** This decoding method uses k-best lists from external models. In this paper, we make use of sequence-to-sequence baseline models as the candidate pool L'.
- **Reranking AP + External.** This decoding method uses the union of the support of $\hat{p}_{\mathbf{Y}}$ and k-best lists from the sequence-to-sequence baseline models as the candidate pool L'.

In this paper, we take 128 path samples per candidate for all **Reranking** methods.

---

[3] `https://en.wikipedia.org/wiki/Help:IPA/English`

[4] The NACS dataset was actually generated from a regular transducer, which we could in principle use, but doing so would make the transduction fully deterministic and probably not interesting/hard enough.

[5] `https://github.com/cmusphinx/cmudict/blob/master/cmudict.phones`

5 Results

5.1 Baselines

We compare NFSTs against the following baselines:

BiRNN-WFSTs proposed by Rastogi et al. (2016), were weighted finite-state transducers whose weights encode input string features by the use of recurrent neural networks. As we note in Table 1, they can be seen as a special case of NFSTs, where the Markov property is kept, but where exact inference is still possible.

Seq2seq models are the standard toolkit for transduction tasks. We make use of the attention mechanism proposed by Luong et al. (2015), which accomplishes 'soft alignments' that do not enforce a monotonic alignment constraint.

Neuralized IBM Model 1 is a character transduction model recently proposed by Wu et al. (2018), which marginalizes over non-monotonic hard alignments between input and output strings. Like (Luong et al., 2015), they did not enforce monotonic alignment constraints; but unlike them, they did not make use of the **input feeding** mechanism,[6] where past alignment information is fed back into the RNN decoder. This particular omission allows (Wu et al., 2018) to do exact inference with a dynamic programming algorithm.

All baseline systems are tuned on the validation sets. The seq2seq models employ GRUs, with word and RNN embedding size = 500 and a dropout rate of 0.3. They are trained with the Adam optimizer (Kingma and Ba, 2014) over 50 epochs. The Neuralized IBM Model 1 models are tuned as described in (Wu et al., 2018).

5.2 The effectiveness of NFSTs

5.2.1 Does losing the Markov property help?

Table 2 indicates that BiRNN-WFST models (Rastogi et al., 2016) perform worse than other models. Their Markovian assumption helps enable dynamic programming, but restricts their expressive power, which greatly hampers the BiRNN-WFST's performance on the P2G/G2P task. The NACS task also relies highly on output-output interactions, and BiRNN-WFST performs very poorly there.

	G2P / P2G				NACS
	EM Accuracy		Edit Distance		EM Accuracy
	Dev	Test	Dev	Test	Test
BiRNN-WFST	16.9	15.9	1.532	1.645	5.6
Seq2seq	30.7	28.9	1.373	1.426	9.0
Neuralized IBM Model 1	31.6	30.2	1.366	1.398	—
Local NFSTs	32.7	31.8	1.319	1.332	15.64

Table 2: Average exact match accuracy (%, higher the better) and edit distance (lower the better) on G2P and P2G as well as exact match accuracy on NACS. Comparison between our models with baselines. For NFST models, we make use of the **Reranking AP** decoding method described in §4.2.

5.2.2 Effectiveness of proposed decoding methods

Table 3 shows results from different decoding methods on the G2P/P2G tasks, configuration **+IPA+PHONE**. **AP** performs significantly worse than **Reranking AP**, suggesting that the estimate $\hat{\mathbf{y}}^*$ suffers from the variance problem. Interestingly, of decoding methods that employ external models, **Reranking External** performs better than **Reranking AP + External**, despite having a smaller candidate pool. We think there is some product-of-experts effect in **Reranking External** since the external model may not be biased in the same way as our model is. But such benefits vanish when candidates from **AP** are also in the pool — our learned approximation learns the bias in the model — and hence the worse performance in **Reranking AP + External**. This suggests an interesting regularization trick in practice: populating the candidate pool using external models to hide our model bias. However when we compare our method against non-NFST baseline methods we do not make use of such tricks, to ensure a more fair comparison.

	EM Accuracy		Edit Distance	
	Dev	Test	Dev	Test
AP	28.2	28.2	1.513	1.467
Reranking AP	32.7	31.8	1.319	1.332
Reranking External	33.3	32.7	1.297	1.298
Reranking AP + External	32.9	32.0	1.309	1.303

Table 3: Average exact match accuracy (%, higher the better) and edit distance (lower the better) on G2P and P2G. The effectiveness of different decoding methods.

5.3 Prior knowledge: does it help?

In Table 4 we see that combining both +IPA and +PHONE improves model generalizability over the general FST (-IPA -PHONE). We also note that using only the IPA marks leads to degraded performance

---

[6] We discuss this further in Appendix B.1.

	EM Accuracy		Edit Distance	
	Dev	Test	Dev	Test
-IPA -PHONE	31.8	29.3	1.38	1.373
+IPA -PHONE	31.3	29.2	1.367	1.431
+IPA +PHONE	32.7	31.8	1.319	1.332

Table 4: Average exact match accuracy (%, higher the better) and edit distance (lower the better) on G2P and P2G. The effectiveness of different FST designs.

compared to the general FST baseline. This is a surprising result — one explanation is the IPA marks are not defined on all paths that transduce the intended input-output pairs: NFSTs are capable of recognizing phoneme-grapheme alignments in different paths,[7] but only one such path is marked by +IPA. But we leave a more thorough analysis to future work.

6 Related Work

Recently, there has been work relating finite-state methods and neural architectures. For example, Schwartz et al. (2018) and Peng et al. (2018) have shown the equivalence between some neural models and WFSAs. The most important differences of our work is that in addition to classifying strings, NFSTs can also transduce strings. Moreover, NFSTs also allow free topology of FST design, and breaks the Markovian assumption. In addition to models we compare against in §4, we note that (Aharoni and Goldberg, 2017; Deng et al., 2018) are also similar to our work; in that they also marginalize over latent alignments, although they do not enforce the monotonicity constraint. Work that discusses globally normalized sequence models are relevant to our work. In this paper, we discuss a training strategy that bounds the partition function; other ways to train a globally normalized model (not necessarily probabilistic) include (Wiseman and Rush, 2016; Andor et al., 2016). On the other hand, our locally normalized FSTs bear resemblance to (Dyer et al., 2016), which was also locally normalized, and also employed importance sampling for training.

7 Conclusions and Future Work

Neural finite state transducers (NFSTs) are able to model string pairs, considering their monotonic alignment but also enjoying RNNs' power to handle non-finite-state phenomena. They compete favorably with state-of-the-art neural models on transduction tasks. At the same time, it is easy to inject domain knowledge into NFSTs for inductive bias, and they offer interpretable paths.

In this paper, we have used rather simple architectures for our RNNs; one could experiment with multiple layers and attention. One could also experiment with associating marks differently with arcs—the marks are able to convey useful domain information to the RNNs. For example, in a P2G or G2P task, all arcs that cross a syllable boundary might update the RNN state using a `syllable` mark. We envision using regular expressions to build the NFSTs, and embedding marks in the regular expressions as a way of sending useful features to the RNNs to help them evaluate paths.

In this paper, we have studied NFSTs as standalone systems. But as probabilistic models, they can be readily embedded in a bigger picture: it should be directly feasible to incorporate a globally/locally normalized NFST in a larger probabilistic model (Finkel and Manning, 2009; Chiang et al., 2010).

The path weights of NFSTs could be interpreted simply as scores, rather than log-probabilities. One would then decode by seeking the 1-best path with input $\mathtt{x}$, e.g., via beam search or Monte Carlo Tree Search. In this setting, one might attempt to train the NFST using methods similar to the max-violation structured perceptron or the structured SVM.

Acknowledgments

This work has been generously supported by a Google Faculty Research Award and by Grant No. 1718846 from the National Science Foundation, both to the last author. Hao Zhu is supported by Tsinghua University Initiative Scientific Research Program. We thank Shijie Wu for providing us IBM Neuralized Model 1 experiment results.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Association for Computational Linguistics*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

---

[7] This is discussed further in Appendix B.2.

Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: Scan both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55.

Jean Berstel, Jr. and Christophe Reutenauer. 1988. *Rational Series and Their Languages*. Springer-Verlag, Berlin, Heidelberg.

Christopher M Bishop. 2006. Pattern recognition and machine learning.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. 2010. Bayesian inference for finite-state transducers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 447–455. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. ACL.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 625–630, Baltimore.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9735–9747. Curran Associates, Inc.

Markus Dreyer. 2011. *A Non-Parametric Model for the Discovery of Inflectional Paradigms from Plain Text Using Graphical Models over Strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *HLT-NAACL*.

Jenny Rose Finkel and Christopher D Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Chu-Cheng Lin and Jason Eisner. 2018. Neural particle smoothing for sampling from conditional sequence models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 929–941, New Orleans.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *EMNLP*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 623–633, San Diego. 11 pages. Supplementary material (1 page) also available.

Emmanuel Roche and Yves Schabes, editors. 1997. *Finite-State Language Processing*. MIT Press.

Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Sopa: Bridging cnns, rnns, and weighted finite-state machines. *CoRR*, abs/1805.06061.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *INTERSPEECH*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Richard L Venezky. 2011. *The structure of English orthography*, volume 82. Walter de Gruyter.

Robert Weide. 2005. The carnegie mellon pronouncing dictionary [cmudict. 0.6].

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438.

A Finite-state transducers

A.1 Rational Relations

A **relation** is a set of pairs—in this paper, a subset of $\Sigma^* \times \Delta^*$, so it relates strings over an "input" alphabet Σ to strings over an "output" alphabet Δ.

A **weighted relation** is a function R that maps any string pair $(\mathbf{x}, \mathbf{y})$ to a **weight** in $\mathbb{R}_{\geq 0}$.

We say that the relation R is **rational** if R can be defined by some weighted finite-state transducer (FST) $\mathcal{T}$. As formalized in Appendix A.3, this means that $R(\mathbf{x}, \mathbf{y})$ is the total weight of all accepting paths in $\mathcal{T}$ that are labeled with $(\mathbf{x}, \mathbf{y})$ (which is 0 if there are no such accepting paths). The weight of each accepting path in $\mathcal{T}$ is given by the product of its arc weights, which fall in $\mathbb{R}_{>0}$.

The set of pairs $\text{support}(R) \triangleq \{(\mathbf{x}, \mathbf{y}) : R(\mathbf{x}, \mathbf{y}) > 0\}$ is then said to be a **regular relation** because it is recognized by the unweighted FST obtained by dropping the weights from $\mathcal{T}$. In this paper, we are interested in defining *non-rational* weighting functions R with this same regular support set.

A.2 Finite-state transducers

We briefly review finite-state transducers (FSTs). Formally, an FST is a tuple $\mathcal{T}_0 = (\Sigma, \Delta, Q, A_0, I, F)$ where

- Σ is a finite input alphabet
- Δ is a finite output alphabet
- Q is a finite set of states
- $A_0 \subseteq Q \times Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\})$ is the set of weighted arcs
- $I \subseteq Q$ is the set of initial states (conventionally $|I| = 1$)
- $F \subseteq Q$ is the set of final states

Let $\mathbf{a} = a_1 \ldots a_T$ (for $T \geq 0$) be an accepting path in $\mathcal{T}_0$, that is, each $a_i = (q_{i-1}, q_i, \sigma_i, \delta_i) \in A_0$ and $q_0 \in I, q_T \in F$. We say that the input and output strings of $\mathbf{a}$ are $\sigma_1 \cdots \sigma_T$ and $\delta_1 \cdots \delta_T$.

A.3 Real-valued weighted FSTs

Weighted FSTs (WFSTs) are defined very similarly to FSTs. A WFST is formally defined as a 6-tuple, just like an (unweighted) FST: $\mathcal{T} = (\Sigma, \Delta, Q, A, I, F)$, with arcs carrying weights: $A \subseteq Q \times Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R}$. Compared to FST arcs in Appendix A.2, a WFST arc each $a_i = (q_{i-1}, q_i, \sigma_i, \delta_i, \kappa_i) \in A$ has weight κ_i. We also define the weight of $\mathbf{a}$ to be $w(\mathbf{a}) \triangleq \bigotimes_{i=1}^{T} \kappa_i \in \mathbb{R}$.

The weight of the entire WFST $\mathcal{T}$ is defined as the total weight (under $\oplus$) of all accepting paths:

$$\mathcal{T}[\,] \triangleq \bigoplus_{\mathbf{a}} w(\mathbf{a}) \in \mathbb{R} \tag{16}$$

More interestingly, the weight $\mathcal{T}[\mathbf{x}, \mathbf{y}]$ of a string pair $\mathbf{x} \in \Sigma^*, \mathbf{y} \in \Delta^*$ is given by similarly summing $w(\mathbf{a})$ over just the accepting paths $\mathbf{a}$ whose input string is $\mathbf{x}$ and output string is $\mathbf{y}$.

B More analysis on the effectiveness of NFSTs

B.1 Does feeding alignments into the decoder help?

In particular, we attribute our models' outperforming Neuralized IBM Model 1 to the fact that a complete history of past alignments is remembered in the RNN state. (Wu et al., 2018) noted that in character transduction tasks, past alignment information seemed to barely affect decoding decisions made afterwards. However, we empirically find that there is performance gain by explicitly modeling past alignments. This also shows up in our preliminary experiments with non-input-feeding seq2seq models, which resulted in about 1% of lowered accuracy and about 0.1 longer edit distance.

B.2 Interpretability of learned paths

The model is not required to learn transduction rules that conform to our linguistic knowledge. However, we expect that a well-performing one would tend to pick up rules that resemble what we know. To verify this, we obtain samples (listed in Table 4) from $\hat{p}(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$ using the importance sampling algorithm described in §3.3. We find that our NFST model has learned to align phonemes and graphemes, generating them alternately. It has no problem picking up obvious pairs in the English orthography (e.g. (ʃ, `c h`), and (ŋ, `n g`)). We also find evidence that the model has picked up how context affects alignment: for example, the model has learned that the bigram 'gh' is pronounced differently in different contexts: in 'onslau**ght**,' it is aligned with ɔ in the sequence 'augh;' in 'Willin**gh**am,' it spans over two phonemes ŋ h; and in '**gh**ezzi,' it is aligned with the phoneme ɡ. We also find that our NFST has no problem learning phoneme-grapheme alignments that span over two

Input / Output	Paths	$\hat{P}(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$
/mɑɹʃ/ marche	ϵ:m m:ɑ a:ɹ r:ʃ c:ϵ h:ϵ e:ϵ	96.5%
	ϵ:m m:ɑ a:ɹ r:ϵ ϵ:ʃ c:ϵ h:ϵ e:ϵ	2.5%
	ϵ:m m:ɑ a:ϵ ϵ:ɹ r:ʃ c:ϵ h:ϵ e:ϵ	1.0%
/ɔnslɔt/ onslaught	ϵ:ɔ o:n n:ϵ ϵ:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:t t:ϵ	76.3%
	ϵ:ɔ o:n n:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:t t:ϵ	21.4%
	ϵ:ɔ o:n n:ϵ ϵ:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:ϵ ϵ:t t:ϵ	1.5%
/wɪlɪŋhəm/ Willingham	ϵ:w W:ɪ i:l l:ϵ l:ϵ ϵ:ɪ i:ŋ n:ϵ g:ϵ ϵ:h h:ə a:ϵ ϵ:m m:ϵ	40.1%
	ϵ:w W:ɪ i:l l:ϵ l:ɪ i:ŋ n:ϵ g:ϵ ϵ:h h:ə a:ϵ ϵ:m m:ϵ	36.6%
	ϵ:w W:ɪ i:l l:ϵ l:ɪ i:ŋ n:ϵ g:h h:ə a:ϵ ϵ:m m:ϵ	7.4%
/gezɪ/ ghezzi	ϵ:g g:ϵ h:e e:z z:ϵ ɪ:z i:ϵ	98.8%
	ϵ:g g:e h:ϵ e:z z:ɪ z:ϵ i:ϵ	1.2%

Table 5: Most probable paths from $\mathbf{x} \circ \mathcal{T} \circ \mathbf{y}$ under the approximate posterior distribution.

arcs, which is beyond the capability of of ordinary WFSTs.

C Implementation Details

C.1 Model parametrization details

As mentioned before, the type of RNN that we use is GRU. The GRU parameterizing $G_{\boldsymbol{\theta}}$ has 500 hidden states. The embedding sizes of tokens, including the input symbol, output symbol and states, and marks are all 500. During inference we make use of proposal distributions $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathcal{T}')$, where $\mathcal{T}' \in \{\mathbf{x} \circ \mathcal{T}, \mathcal{T} \circ \mathbf{y}, \mathbf{x} \circ \mathcal{T} \circ \mathbf{y}\}$. All RNNs used to parametrize $q_{\boldsymbol{\phi}}$ are also GRUs, with 125 hidden states. $q_{\boldsymbol{\phi}}$ makes use of input/output embeddings independent from $G_{\boldsymbol{\theta}}$, which also have size 125 in this paper. The feed-forward networks $C^{\mathrm{x},\mathrm{y},\mathrm{xy}}$ are parametrized by 3-layer networks, with ReLU as the activation function of the first two layers. The output dimension sizes of the first and second layers are $\lfloor {}^{D}/_{2} \rfloor$ and $\lfloor {}^{D}/_{4} \rfloor$, where D is the input vector dimension size.

C.2 Training procedure details

We use stochastic gradient descent (SGD) to train $G_{\boldsymbol{\theta}}$. For each example, we compute the gradient using normalized importance sampling over an ensemble of 512 particles (paths), the maximum that we could compute in parallel. By using a large ensemble, we reduce both the bias (from normalized importance sampling) and the variance of the gradient estimate; we found that smaller ensembles did not work as well. Thus, we used only one example per minibatch.

We train the 'clamped' proposal distribution $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})$ differently from the 'free' ones $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{x} \circ \mathcal{T})$ and $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathcal{T} \circ \mathbf{y})$. The clamped distribution is trained alternately with $G_{\boldsymbol{\theta}}$, as listed in Algorithm 2. We evaluate on the development dataset at the end of each epoch using the **Reranking External** method described in §4.3. When the EM accuracy stops improving, we fix the parameters of $G_{\boldsymbol{\theta}}$ and start training $q_{\boldsymbol{\phi}}(\mathbf{x} \circ \mathcal{T})$ and $q_{\boldsymbol{\phi}}(\mathcal{T} \circ \mathbf{y})$ on the inclusive KL divergence objective function, using methods described in (Lin and Eisner, 2018). We then initialize the free distributions' RNNs using those of the clamped distributions. We train the free proposal distributions for 30 epochs, and evaluate on the development dataset at the end of each epoch. Results from the best epochs are reported in this paper.

Riemannian Normalizing Flow on Variational Wasserstein Autoencoder for Text Modeling

Prince Zizhuang Wang
Department of Computer Science
University of California, Santa Barbara
`zizhuang_wang@ucsb.edu`

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
`william@cs.ucsb.edu`

Abstract

Recurrent Variational Autoencoder has been widely used for language modeling and text generation tasks. These models often face a difficult optimization problem, also known as the Kullback-Leibler (KL) term vanishing issue, where the posterior easily collapses to the prior, and the model will ignore latent codes in generative tasks. To address this problem, we introduce an improved Wasserstein Variational Autoencoder (WAE) with Riemannian Normalizing Flow (RNF) for text modeling. The RNF transforms a latent variable into a space that respects the geometric characteristics of input space, which makes posterior impossible to collapse to the non-informative prior. The Wasserstein objective minimizes the distance between the marginal distribution and the prior directly, and therefore does not force the posterior to match the prior. Empirical experiments show that our model avoids KL vanishing over a range of datasets and has better performances in tasks such as language modeling, likelihood approximation, and text generation. Through a series of experiments and analysis over latent space, we show that our model learns latent distributions that respect latent space geometry and is able to generate sentences that are more diverse. [1]

1 Introduction

Variational Autocoder (VAE) (Kingma and Welling, 2013; Rezende and Mohamed, 2015) is a probabilistic generative model shown to be successful over a wide range of tasks such as image generation (Gregor et al., 2015; Yan et al., 2016), dialogue generation (Zhao et al., 2017b), transfer learning (Shen et al., 2017), and classification (Jang et al., 2017). The encoder-decoder architecture of VAE allows it to learn a continuous space of latent representations from high-dimensional data input and makes sampling procedure from such latent space very straightforward. Recent studies also show that VAE learns meaningful representations that encode non-trivial information from input (Gao et al., 2018; Zhao et al., 2017a).

Applications of VAE in tasks of Natural Language Processing (Bowman et al., 2015; Zhao et al., 2017b, 2018; Miao et al., 2016) is not as successful as those in Computer Vision. With *long-short-term-memory* network (LSTM) (Hochreiter and Schmidhuber, 1997) used as encoder-decoder model, the recurrent variational autoencoder (Bowman et al., 2015) is the first approach that applies VAE to language modeling tasks. They observe that LSTM decoder in VAE often generates texts without making use of latent representations, rendering the learned codes as useless. This phenomenon is caused by an optimization problem called KL-divergence vanishing when training VAE for text data, where the KL-divergence term in VAE objective collapses to zero. This makes the learned representations meaningless as zero KL-divergence indicates that the latent codes are independent of input texts.

Many recent studies are proposed to address this key issue. Yang et al. (2017); Semeniuta et al. (2017) use *convolutional neural network* as decoder architecture to limit the expressiveness of decoder model. Xu and Durrett (2018); Zhao et al. (2017a,b, 2018) seek to learn different latent space and modify the learning objective. And, even though not designed to tackle KL vanishing at the beginning, recent studies on *Normalizing Flows* (Rezende and Mohamed, 2015; van den Berg et al., 2018) learn meaningful latent space as it helps to transform an over-simplified latent distribution into more flexible distributions.

In this paper, we propose a new type of flow,

---

[1] Code could be found at `https://github.com/kingofspace0wzz/wae-rnf-lm`

Proceedings of NAACL-HLT 2019, pages 284–294
Minneapolis, Minnesota, June 2 - June 7, 2019.

called Riemannian Normalizing Flow (RNF), together with the recently developed Wasserstein objective (Tolstikhin et al., 2018; Arjovsky et al., 2017), to ensure VAE models more robust against the KL vanishing problem. As further explained in later sections, the Wasserstein objective helps to alleviate KL vanishing as it only minimizes the distance between latent marginal distribution and the prior. Moreover, we suspect that the problem also comes from the over-simplified prior assumption about latent space. In most cases, the prior is assumed to be a standard Gaussian, and the posterior is assumed to be a diagonal Gaussian for computational efficiency. These assumptions, however, are not suitable to encode intrinsic characteristics of input into latent codes as in reality the latent space is likely to be far more complex than a diagonal Gaussian.

The RNF model we proposed in this paper thus helps the situation by encouraging the model to learn a latent space that encodes some geometric properties of input space with a well-defined geometric metric called Riemannian metric tensor. This renders the KL vanishing problem as impossible since a latent distribution that respects input space geometry would only collapse to a standard Gaussian when the input also follows a standard Gaussian, which is never the case for texts and sentences datasets. We then empirically evaluate our RNF Variational Wasserstein Autoencoder on standard language modeling datasets and show that our model has achieved state-of-the-art performances. Our major contributions can be summarized as the following:

- We propose Riemannian Normalizing Flow, a new type of flow that uses the Riemannian metric to encourage latent codes to respect geometric characteristics of input space.
- We introduce a new Wasserstein objective for text modeling, which alleviates KL divergence term vanishing issue, and makes the computation of normalizing flow easier.
- Empirical studies show that our model produces state-of-the-art results in language modeling and is able to generate meaningful text sentences that are more diverse.

2 Related Work

2.1 Variational Autoencoder

Given a set of data $\mathbf{x} = (x_1, x_2, ..., x_n)$, a Variational Autoencoder (Kingma and Welling, 2013) aims at learning a continuous latent variable that maximizes the log-likelihood $\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. Since this marginal is often intractable, a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is used to approximate the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. VAE tries to maximize the following lower bound of likelihood,

$$\mathcal{L}(\theta;\phi;\mathbf{x}) = \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \quad (1)$$

$$- KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] \quad (2)$$

where $q(\mathbf{x})$ is the empirical distribution of input, and the prior $p(\mathbf{z})$ is often assumed to be a standard Gaussian for simplicity. The first term in the objective is the reconstruction error, and the second one is KL divergence. For modeling text sentences, Bowman et al. (2015) parameterizes both the inference model $q(\mathbf{z}|\mathbf{x})$ and the generative model $p(\mathbf{x}|\mathbf{z})$ as LSTMs. The reparameterization trick proposed by Rezende and Mohamed (2015) is used to train these two models jointly.

2.2 KL Divergence Term Vanishing

Since the generative model is often an LSTM that has strong expressiveness, the reconstruction term in the objective will dominate KL divergence term. In this case, the model is able to generate texts without making effective use of latent codes as the latent variable $\mathbf{z}$ becomes independent from input when KL divergence term collapses to zero.

There are two main approaches to address this issue. One is to explore different choices of the decoder model to control the expressiveness of LSTM. Yang et al. (2017) and Semeniuta et al. (2017) use CNN as an alternative to LSTM. The dilation technique used by Yang et al. (2017) also helps to control the trade-off between decoder capacity and KL vanishing. The other approach is to change the form of latent distribution and to modify the training objective. Xu and Durrett (2018) proposes to use hyperspherical distribution and shows that the KL vanishing problem does not happen in hypersphere. The infoVAE (Zhao et al., 2017a) argues that VAE objective is anti-informatics which encourages KL divergence to be zero. They, therefore, add a mutual information term $I(\mathbf{z};\mathbf{x})$ explicitly to ensure that latent variable $\mathbf{z}$ encodes non-trivial information about $\mathbf{x}$, in which case the KL would be greater than zero as $\mathbf{z}$ is no longer independent from $\mathbf{x}$. In a similar manner, Xiao et al. (2018) introduces a Dirichlet latent variable to force latent codes to learn useful topic information given input documents. He et al. (2019) and Kim et al. (2018) achieve the current state-of-the-art in terms of sample perplexity. Moreover, the bag-of-words loss used by Zhao et al. (2017b) also dramatically alleviates KL vanishing while not sacrificing sample quality. In Section 3 and Section 4, we introduce RNF with Wasserstein objective. Our proposed model also lies in the direction which seeks to learn more flexible latent distribution, as the main advantage of RNF is to ensure a flexible latent space able to capture geometric characteristics of input space.

3 Background

In this section, we review the basic concepts of Riemannian geometry, normalizing flow, and Wasserstein Autoencoder. We then introduce our new Riemannian normalizing flow in Section 4.

3.1 Riemannian Geometry Review

Consider an input space $\mathcal{X} \subset R^D$, a d-dimensional $(d < D)$ manifold is a smooth surface of points embedded in $\mathcal{X}$. Given a manifold $\mathcal{M}$, a Riemannian manifold is a metric space $(\mathcal{M}, G)$, where G is the Riemannian metric tensor that assigns an inner product to every point on the manifold. More formally, a Riemannian metric $G : \mathcal{Z} \to R^{d\times d}$ is defined as a smooth function such that for any two vectors u, v in the tangent space $T_z M$ of each point $z \in \mathcal{M}$, it assigns the following inner product for u and v,

$$< u, v >_G = u^T G(z) v \tag{3}$$

The Riemannian metric helps us to characterize many intrinsic properties of a manifold. Consider an arbitrary smooth curve $\gamma(t) : [a, b] \to \mathcal{M}$ on a given manifold $\mathcal{M}$ with a Riemannian metric tensor G, the length of this curve is given by

$$\begin{aligned}\mathcal{L}(\gamma) &= \int_a^b ||\gamma'(t)||dt \\ &= \int_a^b \sqrt{< \gamma'_t, \gamma'_t >_G}dt \\ &= \int_a^b \sqrt{\gamma'^T_t G(\gamma_t)\gamma'_t}dt \end{aligned}\tag{4}$$

where γ'_t is the curve velocity and lies in the tangent space $T_{\gamma_t}M$ at point $\gamma(t)$. When the metric tensor G is equal to 1 everywhere on the curve, it becomes a metric tensor on Euclidean space, where the length of curve is defined as the integral of the velocity function, $\mathcal{L}(\gamma) = \int_a^b \sqrt{\gamma'^T_t \gamma'_t}dt = \int_a^b \gamma'_t dt$. Given the definition of curve length, the geodesic path between any two points can be defined as the curve that minimizes the curve length γ. Namely, if γ_t is the geodesic curve connecting $\gamma(a)$ and $\gamma(b)$, then

$$\gamma_t = \underset{\gamma}{\operatorname{argmin}}\, \mathcal{L}(\gamma) \tag{5}$$

Practically, a geodesic line is often found by optimizing the following energy function,

$$\begin{aligned} E(\gamma) &= \frac{1}{2}\int_a^b \gamma'^T_t G(\gamma_t)\gamma'_t dt \\ \gamma_t &= \underset{\gamma}{\operatorname{argmin}}\, E(\gamma) \end{aligned}\tag{6}$$

Note that the Euclidean metric is a special case of Riemannian metric. The more general metric tensor G gives us a sense of how much Riemannian geometry deviates from Euclidean geometry.

3.2 Review: Normalizing Flow

The powerful inference model of VAE can approximate the true posterior distribution through variational inference. The choice of this approximated posterior is one of the major problems. For computational efficiency, a diagonal Gaussian distribution is often chosen as the form of the posterior. As the covariance matrix is always assumed to be diagonal, the posterior fails to capture dependencies among individual dimensions of latent codes. This poses a difficult problem in variational inference. As it is unlikely that the true posterior has a diagonal form, the approximated diagonal distribution is not flexible enough to match the true posterior even in asymptotic time.

A normalizing flow, developed by (Rezende and Mohamed, 2015), is then introduced to transform a simple posterior to a more flexible distribution. Formally, a series of normalizing flows is a set of invertible, smooth transformations $f_t : R^d \to R^d$, for $t = 1, ..., T$, such that given a random variable z_0 with distribution $q(z_0)$, the resulting random variable $z^T = (f_T \circ f_{T-1} \circ ... \circ f_1)(z_0)$ has the following density function,

$$q(z_T) = q(z_0)\prod_{t=1}^{T} |det \frac{\partial f_t^{-1}}{\partial z_{t-1}}| \tag{7}$$

Since each transformation f_i for $i = 1, ..., T$ is invertible, its Jacobian determinant exists and can be computed. By optimizing the modified *evidence lower bound* objective,

$$\ln p(x) \geq \mathbb{E}_{q(z_0|x)} \left[\ln p(x|z_T) + \sum_{t=1}^{T} \ln |det \frac{\partial f_t}{\partial z_{t-1}}| \right] - KL(q(z_0|x)||p(z_T)) \quad (8)$$

the resulting latent codes z_T will have a more flexible distribution.

Based on how the Jacobian-determinant is computed, there are two main families of normalizing flow (Tomczak and Welling, 2016; Berg et al., 2018): *general normalizing flow* and *volume preserving flow*. While they both search for flexible transformation that has easy-to-compute Jacobian-determinant, the volume-preserving flow aims at finding a specific flow whose Jacobian-determinant equals 1, which simplifies the optimization problem in equation (6). Since we want a normalizing flow that not only gives flexible posterior but also able to uncover the true geometric properties of latent space, we only consider general normalizing flow whose Jacobian-determinant is not a constant as we need it to model the Riemannian metric introduced earlier.

3.3 Wasserstein Autoencoder

Wasserstien distance has been brought to generative models and is shown to be successful in many image generation tasks (Tolstikhin et al., 2018; Arjovsky et al., 2017; Bousquet et al., 2017). Instead of maximizing the *evidence lower bound* as VAE does, the Wasserstein Autoencoder (Tolstikhin et al., 2018) optimizes the optimal transport cost (Villani, 2008) between the true data distribution $P_X(x)$ and the generative distribution $P_G(x)$. This leads to the Wasserstein objective,

$$D(P_X, P_G) = \inf_{Q(Z|X) \in \mathcal{Q}} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)}[c(X, G(Z))] + \lambda D_Z(Q_Z, P_Z) \quad (9)$$

where $c(\cdot)$ is the optimal transport cost, $G : \mathcal{Z} \rightarrow \mathcal{X}$ is any generative function, and the coefficient λ controls the strength of regularization term D_Z. Given a positive-definite reproducing kernel $k : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathcal{R}$, the regularization term D_Z can be approximated by the *Maximum Mean Discrepancy* (**MMD**) (Gretton et al., 2012) between the prior P_Z and the aggregate posterior $Q_Z(z) = \int q(z|x)p(x)dx$,

$$MMD_k(P_Z, Q_Z) = || \int_{\mathcal{Z}} k(z, \cdot)dP_Z - \int_{\mathcal{Z}} k(z, \cdot)dQ_Z|| \quad (10)$$

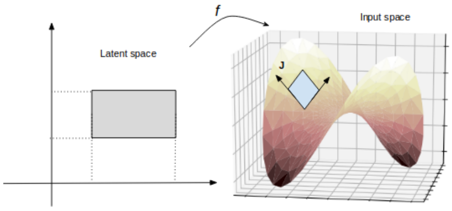


Figure 1: Parameterization of the input manifold by latent space and generative function f. J is the Jacobian tangent space at a point on the manifold, which reflects how curved the neighborhood is around that point.

4 Our Approach

In this section we propose our *Riemannian Normalizing Flow* (**RNF**). RNF is a new type of flow that makes use of the Riemannian metric tensor introduced earlier in Section 3. This metric enforces stochastic encoder to learn a richer class of approximated posterior distribution in order to follow the true geometry of latent space, which helps to avoid the local optimum in which posterior collapses to a standard prior. We then combine this with WAE and we will explain why and how WAE should be used to train with RNF.

4.1 Riemannian Normalizing Flow

In the context of VAE, learning a latent space that is homeomorphic to input space is often very challenging. Consider a manifold $\mathcal{M} \subset R^D$, a generator model $x = f(z) : \mathcal{Z} \rightarrow R^D$ serves as a low-dimensional parameterization of manifold $\mathcal{M}$ with respect to $z \in \mathcal{Z}$. For most cases, latent space $\mathcal{Z}$ is unlikely to be homeomorphic to $\mathcal{M}$, which means that there is no invertible mapping between $\mathcal{M}$ and $\mathcal{Z}$. And, since the inference model $h : \mathcal{M} \rightarrow \mathcal{Z}$ is nonlinear, the learned latent space often gives a distorted view of input space. Consider the case in Figure 2, where the leftmost graph is the input manifold, and the rightmost graph is the corresponding latent space with curvature reflected by brightness. Let us take two arbitrary points on the manifold and search for the geodesic path connecting these two points. If we consider the distorted latent space as Euclidean, then the geodesic path in latent space does not reflect the true shortest distance between these two points on the manifold, as a straight line in the latent space would

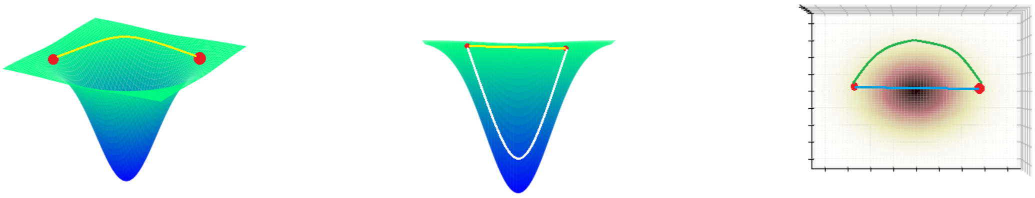


Figure 2: An example when latent space does not reflect input space. *Left:* a manifold that is highly curved in the central region. The yellow line is the geodesic (shortest) path connecting two sample points shown on the manifold. *Right:* The projection of manifold into 2D latent space, where the color brightness indicates curvature with respect to the manifold. The green line is the geodesic path if taking the curvature into account, while the blue line is the geodesic path if we regard latent space as Euclidean. *Middle:* The corresponding geodesic paths projected back from latent space to manifold. The white line corresponds to the straight geodesic path in Euclidean space. It is far longer than the true geodesic on manifold since it does not take the curvature into account in latent space.

cross the whole manifold, while the true geodesic path should circumvent this hole. This distortion is caused by the non-constant curvature of latent space. Hence, the latent space should be considered as a curved space with curvature reflected by the Riemannian metric defined locally around each point. As indicated by the brightness, we see that the central area of latent space is highly curved, and thus has higher energy. The geodesic path connecting the two latent codes minimizes the energy function $E(\gamma) = \frac{1}{2}\int \gamma'^{T}_{t} G(\gamma_t)\gamma'_t dt$, indicating that it should avoid those regions with high curvature G.

The question now becomes how to impose this intrinsic metric and curvature into latent space. In this paper, we propose a new form of normalizing flow to incorporate with this geometric characteristic. First, consider a normalizing flow $f : \mathcal{Z} \rightarrow \mathcal{Z}'$, we can compute length of a curve in the transformed latent space $\mathcal{Z}'$,

$$\begin{aligned}\mathcal{L}(f(\gamma_t)) &= \int_a^b ||\mathbf{J}_{\gamma_t}\gamma'_t||dt \\ &= \int_a^b \sqrt{\gamma'_t \mathbf{J}^T_{\gamma_t}\mathbf{J}_{\gamma_t}\gamma'_t}dt \\ &= \int_a^b \sqrt{\gamma'_t G(\gamma_t)\gamma'_t}dt \qquad (11)\end{aligned}$$

where

$$\begin{aligned}\gamma_t : [a, b] &\rightarrow \mathcal{Z}', \qquad a, b \in \mathcal{Z} \\ \mathbf{J}_{\gamma_t} &= \frac{\partial f}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\gamma_t} \\ G(\gamma_t) &= \mathbf{J}^T_{\gamma_t}\mathbf{J}_{\gamma_t} \\ &= (\frac{\partial f}{\partial \mathbf{z}})^T(\frac{\partial f}{\partial \mathbf{z}})\Big|_{\mathbf{z}=\gamma_t} \qquad (12)\end{aligned}$$

$\mathbf{J}_{\gamma_\mathbf{t}}$ is the Jacobian matrix defined at γ_t. In our case, the Riemannian metric tensor G is the inner product of Jacobian $\mathbf{J}_{\gamma_t}$ and is therefore symmetric positive definite. It reflects input space curvature in low-dimensional parameterization $\mathcal{Z}'$. In a highly curved region, the metric tensor $G = \mathbf{J}^T\mathbf{J}$ is larger than those in other areas, indicating that the latent representation of input manifold has lower curvature, or area of low energy, as any geodesic connecting each pair of points on the manifold favors lower energy path. This implies that those regions outside of data manifold should have high curvature reflected in their low-dimensional parameterization $\mathcal{Z}'$.

In this paper, we introduce *Riemannian normalizing flow* (**RNF**) to model curvature. For simplicity, we build our model based on *planar flow* (Rezende and Mohamed, 2015). A planar flow is an invertible transformation that retracts and extends the support of original latent space with respect to a plane. Mathematically, a planar flow $f : \mathcal{Z} \rightarrow \mathcal{Z}'$ has the form,

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w^T z} + b) \qquad (13)$$

where $h : R^d \rightarrow R^d$ is any smooth non-linear function and is often chosen as $tanh(\cdot)$. The invertibiliy condition is satisfied as long as $\mathbf{u^T w} \geq -1$. Its Jacobian-determinant with respect to latent codes $\mathbf{z}$ is very each to compute,

$$\begin{aligned}|det\frac{\partial f}{\partial \mathbf{z}}| &= |1 + \mathbf{u^T}\phi(\mathbf{z})\mathbf{w}| \\ \phi(\mathbf{z}) &= h'(\mathbf{w^T}z + b) \qquad (14)\end{aligned}$$

With the Jacobian-determinant of planar flow, it is straightforward to compute the determinant of metric tensor G. To see that, note that since $\frac{\partial f}{\partial \mathbf{z}}$:

$R^d \rightarrow R^d$ is a square matrix with full column rank due to invertibility of f, we have

$$|det\ G| = \left|\frac{\partial f}{\partial \mathbf{z}}^T\right| \left|\frac{\partial f}{\partial \mathbf{z}}\right| = \left|\frac{\partial f}{\partial \mathbf{z}}\right|^2 \tag{15}$$

To ensure well-behaved geometry in a transformed latent space, we need the Jacobian-determinant $|\frac{\partial f}{\partial \mathbf{z}}|$ to be large in region with high curvature $|det\ G|$. Hence, we propose to model the metric tensor with the inverse multiquadratics kernel function $\mathcal{K}$ used by (Tolstikhin et al., 2018) and a Gaussian kernel, that is,

$$\begin{aligned} \mathcal{K}_m(\mathbf{z}, \mathbf{c_k}) &= C/(C + ||\mathbf{z} - \mathbf{c_k}||_2^2) \\ \mathbf{k} &= \text{argmin}\, ||\mathbf{z} - \mathbf{c_k}||_2^2 \\ \mathcal{K}_g(\mathbf{z}, \mathbf{c_k}) &= \exp\left(-\beta_k ||\mathbf{z} - \mathbf{z_k}||_2^2\right) \end{aligned}$$

where $\mathbf{c_k}, k = 1, 2, 3, ..., K$ are clusters of latent codes, and β_k is the bandwidth. We observe that the inverse multiquadratics kernel $\mathcal{K}_m$ generally performs better. We use the above kernels as constraints over the Jacobian-determinant, so that,

$$|det \frac{\partial f'}{\partial \mathbf{z}}| = |1 + \mathbf{u^T}(\mathbf{z})\phi(\mathbf{z})\mathbf{w}| \cdot \mathcal{K}(\mathbf{z}, \mathbf{c_k}) \tag{16}$$

As we explained earlier in this section, latent representation of region outside of input manifold should have high curvature in latent space. During training, we seek to maximize this regularized Jacobian $|det \frac{\partial f'}{\partial \mathbf{z}}|$ rather than the original one. This ensures that those latent codes within latent clusters, and therefore very likely to be on or near input manifold in input space, have much smaller curvature $|det\ G| = |\frac{\partial f}{\partial \mathbf{z}}|^2$ than those outside of latent clusters, as those outside of manifold would seek larger Jacobian in order to counter-effect the regularization term $\mathcal{K}$. The latent space $\mathcal{Z}'$ transformed by normalizing flow f is thus curved with respect to input manifold. This type of normalizing flow thus learns a latent space to respect geometric characteristics of input space. The KL vanishing problem is then unlikely to happen with a curved latent space. This is because most high-dimensional data in real life forms a curved manifold which is unlikely sampled from a multivariate standard Gaussian. Then, if the latent space reflects curvature of a curved manifold, the support of latent codes certainly does not follow a standard Gaussian either. This helps to push the posterior $q(\mathbf{z}|\mathbf{x})$ away from the standard Gaussian and never collapse to a non-informative prior.

4.2 RNF Wasserstein Autoencoder

Here we consider using the Wasserstein objective to model the latent marginal distribution of a curved latent space learned by an RNF. The Wasserstein objective with MMD is appealing in our case for two main reasons.

First, instead of minimizing the KL-divergence $KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$, it minimizes distance between $q(\mathbf{z}) = \int q(\mathbf{z}|\mathbf{x})p(\mathbf{x})dx$ and $p(\mathbf{z})$, which encourages the marginal distribution of latent space to be as close as the prior while not affecting individual posterior distribution $q(\mathbf{z}|\mathbf{x})$ conditioned on each input. This makes the KL-divergence between posterior and prior, and equivalently the mutual information between latent codes and input sentences $I(\mathbf{z}, \mathbf{x}) = KL(q(\mathbf{z}, \mathbf{x})||q(\mathbf{z})p(\mathbf{x})) = \mathbb{E}_{p(\mathbf{x})}[KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] - KL(q(\mathbf{z})||p(\mathbf{z}))$ impossible to be vanished as the objective does not require it to be small. Since the learned latent codes and input sentences have non-zero mutual information, the generative model will not ignore latent codes when generating texts.

Second, the MMD regularization in WAE makes it possible to optimize normalizing flow without computing the Jacobian-determinant explicitly. The use of MMD is necessary as getting a closed form KL divergence is no-longer possible after we apply RNF to the posterior. And, since the generative function G in the reconstruction term of Wasserstein objective can be any function or composition of functions (Tolstikhin et al., 2018), we can easily compose an RNF function into G such that the reconstructed texts are $\bar{X} = G(f(Z)) = G(Z'), Z' \in \mathcal{Z}'$.

Now, given a series of RNF $F = f_K \circ ... \circ f_1$, and let $\mathcal{Z}_K$ be the **curved** latent space after applying K flows over the original latent space $\mathcal{Z}$, we optimize the following RNF-Wasserstein objective,

$$\begin{aligned} D(P_X, P_G) = &\inf_{Q(Z|X) \in \mathcal{Q}} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)}[c(X, G(Z'))] \\ &+ \lambda MMD(Q_{Z'}, P_{Z'}) \\ &+ \alpha (KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - \sum \log |det \frac{\partial f'}{\partial \mathbf{z}}|) \end{aligned} \tag{17}$$

where $Z \sim \mathcal{Z}$, $Z' \sim \mathcal{Z}'$, and $Z' = F(Z)$. We approximate MMD term with the Gaussian kernel $k(\mathbf{z}, \mathbf{z}') = e^{-||\mathbf{z}-\mathbf{z}'||^2}$, that is, $MMD(p, q) = \mathbb{E}_{p(\mathbf{z}), p(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')] + \mathbb{E}_{q(\mathbf{z}), q(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')] - 2\, \mathbb{E}_{p(\mathbf{z}), q(\mathbf{z}')}[k(\mathbf{z}, \mathbf{z}')]$.

Here we choose to minimize the MMD distance

Model	**PTB**		**YAHOO**		**YELP**	
	NLL(KL)	PPL	NLL(KL)	PPL	NLL(KL)	PPL
LSTM-LM **	116.2 (-)	104.2	334.9 (-)	66.2	- -	
VAE	105.2 (1.74)	121.1	339.2 (0.01)	69.9	198.6 (0.01)	55.0
VAE-NF	96.8 (0.87)	82.9	353.8 (0.10)	83.0	200.4 (0.10)	62.5
lagging-VAE **	- -		**326.6 (6.70)**	**64.9**	–	
vmf-VAE	96.0 (5.70)	79.6	359.3 (17.9)	89.9	198.0 (6.40)	54.0
WAE-RNF	**91.9 (15.4)**	**66.1**	**339.0 (3.00)**	**71.6**	**183.9 (12.7)**	**41.1**

Table 1: Language Modeling results on PTB, YAHOO and YELP 13 Reviews. ** are results gathered from (Yang et al., 2017; Xiao et al., 2018; He et al., 2019). Negative log-likelihood (NLL) is approximated by its lower bound, where the number in parentheses indicates KL-divergence. NF stands for the standard planar normalizing flow without Riemannian curvature.

Data	PTB				Yelp			
Model	**NLL**	**Re-KL**	**Log-J**	**PPL**	**NLL**	**Re-KL**	**Log-J**	**PPL**
WAE	104.9	–(1.9)	–	131.	198.5	–(1.9)	–	55
WAE-NF	92.3	14.3	14.3	67.3	184.3	13.9	14.2	41.4
WAW-RNF	**91.9**	**15.4**	**15.2**	**66.1**	**183.9**	**12.7**	**12.1**	**41.1**

Table 2: Language Modeling using WAE-RNF. We report NLL, PPL, Sum of Log Jacobian, and KL divergence between $q(\mathbf{z}'|\mathbf{x})$ and $p(\mathbf{z}')$.

between the prior P_Z and the marginal of non-curved latent space. This makes sampling procedure for generation tasks much easier, as it is easy to sample a latent code from a non-informative prior. We can get a sample $\mathbf{z}'$ from $\mathcal{Z}'$ indirectly by sampling: $\mathbf{z} \sim P_Z(\mathbf{z})$ and $\mathbf{z}' \sim F(\mathbf{z})$. On the other hand, it would be much more difficult to sample from a curved latent space $\mathcal{Z}'$ directly as the only prior knowledge we have about $\mathcal{Z}'$ is the curvature reflected by RNF implicitly and hence we do not know the support of $Q(Z')$.

5 Experimental Results

In this section, we investigate WAE's performance with Riemannian Normalizing Flow over language and text modeling.

5.1 Datasets

We use Penn Treebank (Marcus et al., 1993), Yelp 13 reviews (Xu et al., 2016), as in (Xu and Durrett, 2018; Bowman et al., 2015), and Yahoo Answers used in (Xu and Durrett, 2018; Yang et al., 2017) to follow and compare with prior studies. We limit the maximum length of a sample from all datasets to 200 words. The datasets statistics is shown in Table 3.

5.2 Experimental Setup

For each model, we set the maximum vocabulary size to 20K and the maximum length of input to 200 across all data sets. Following Bowman et al. (2015), we use one-layer undirectional LSTM for both encoder-decoder models with hidden size 200. Latent codes dimension is set to 32 for all models. We share Word Embeddings of size 200. For stochastic encoders, both MLP_μ and MLP_σ are two layer fully-connected networks with hidden size 200 and a batch normalizing output layer (Ioffe and Szegedy, 2015).

Data	**Train**	**Dev**	**Test**	**Vocab**
PTB	42068	3370	3761	10K
Yelp13	62522	7773	8671	15K
Yahoo	100K	10K	10K	20K

Table 3: Datasets statistics; The numbers reflect size of each dataset. Vocab is the vocabulary size.

We use Adam (Kingma and Ba, 2015) with learning rate set to 10^{-3} to train all models. Dropout is used and is set to 0.2. We train all models for 48 epochs, each of which consists of 2K steps. For models other than WAE, KL-annealing is applied and is scheduled from 0 to 1 at the 21st epoch.

For vmf-VAE (Xu and Durrett, 2018), we set the word embedding dimension to be 512 and the hidden units to 1024 for Yahoo, and set both of them to 200 for PTB and Yelp. The temperature κ is set to 80 and is kept constant during training.

For all WAE models, we add a small KL divergence term to control the posterior distribution. We found that if we only use RNF with MMD as the distance metric, then the posterior may diverge from the prior such that no reasonable samples can be generated from a standard Gaussian variable.

Hence, for all data sets, we schedule the KL divergence weight α from 0 to 0.8, and the weight of the MMD term is set as $\lambda = 10 - \alpha$. β_k of RBF is set to 10 for all models. For RNF, we use pretrained standard VAE models to gather the clusters $\mathbf{c_k}$, $k = 1, ..., K$, of latent codes, where we set the number of clusters to be 20. We use three normalizing flow for all experiments.

Hyperparameter of $\mathbf{K}_m$ When using the inverse multiquadratics kernel $\mathcal{K}_m(\mathbf{z}, \mathbf{c}_k) = C/(C + ||\mathbf{z} - \mathbf{c}_k||_2^2)$ for RNF, we follow the choice of hyperparameter in (Tolstikhin et al., 2018). We set $C = 2 \cdot d \cdot s$, where d is the dimensionality of latent codes $\mathbf{z}$, and s is ranged in $(0.1, 0.2, 0.5, 1, 2, 5, 10)$. The final kernel is computed by $\mathcal{K}_m(\mathbf{z}, \mathbf{c}_k) = \sum_s 2ds/(2ds + ||\mathbf{z} - \mathbf{c}_k||_2^2)$. As explained by (Tolstikhin et al., 2018), this strategy allows us to explore a wider range of hyperparameter in one setting.

5.3 Language Modeling Results

We show the language modeling results for PTB, Yahoo and Yelp in Table 1. We compare negative log-likelihood (NLL), KL divergence, and perplexity (PPL) with all other existing methods. The negative log-likelihood is approximated by its lower bound.

We use the negative of ELBO to approximate NLL for all VAE models. For those with normalizing flows, we use the modified ELBO, which is $\mathcal{L} = \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}^{(T)}) - \log q(\mathbf{z}^{(0)}|\mathbf{x}) + \log p(\mathbf{z}^{(T)})] + \mathbb{E}_{q(\mathbf{z}^{(T)})}[\sum_{t=1}^{T} log|\frac{\partial f^{(t)}}{\partial \mathbf{z}^{(t-1)}}|]$.

The numbers show that KL-annealing and dropout used by Bowman et al. (2015) are helpful for PTB, but for complex datasets such as Yahoo and Yelp, the KL divergence still drops to zero due to the over-expressiveness of LSTM. This phenomenon is not alleviated by applying normalizing flow to make the posterior more flexible, as shown in the third row. Part of the reason may be that a simple NF such as a planar flow is not flexible enough and is still dominated by a powerful LSTM decoder.

We find that the KL vanishing is alleviated a little bit if using WAE, which should be the case as WAE objective does not require small KL. We also find that simply applying a planar flow over WAE does not improve the performance that much. On the other hand, using RNF to train WAE dramatically helps the situation which achieves the lowest text perplexity on most conditions except for YAHOO Answers, where (He et al., 2019; Yang et al., 2017; Kim et al., 2018) have the current state-of-the-art results. We want to emphasize that CNN-VAE (Yang et al., 2017) and SA-VAE (Kim et al., 2018) are not directly comparable with other current approaches. Here, we compare with models that use LSTM as encoder-decoder and have similar time complexity, while the use of CNN as decoder in CNN-VAE would dramatically change the model expressiveness, and it is known that SA-VAE's time complexity (Kim et al., 2018; He et al., 2019) is much higher than all other existing approaches.

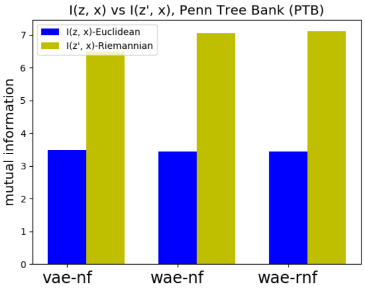


Figure 3: PTB. Comparison between the amount of mutual information stored in latent codes for different models.

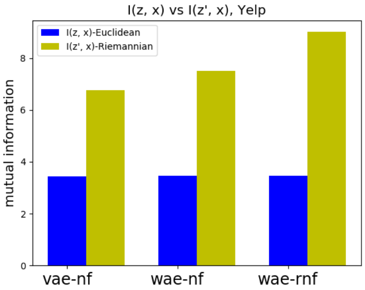


Figure 4: Yelp. Comparison between the amount of mutual information stored in latent codes for different models.

5.4 How Good is Riemannian Latent Representation?

Mutual information between $\mathcal{Z}'$ and $\mathcal{X}$ One important question is how useful are latent variables. Since no metrics are perfect (Wang et al., 2018), we should not just look at sample perplexity to judge how good a latent code is. Hence, we also investigate how much information can be encoded into latent codes. We believe that the mutual information term $I(\mathbf{z}; \mathbf{x})$ is a better metric regarding the usefulness of latent codes than sample

the company said it will be sold to the company 's promotional programs and _UNK
the company also said it will sell \$ n million of soap eggs turning millions of dollars
the company said it will be _UNK by the company 's _UNK division n
the company said it would n't comment on the suit and its reorganization plan
this is a reflection of socialism and capitalism
the company also said it will sell its _UNK division of the company 's _UNK
earlier this year the company said it will sell \$ n billion of assets and _UNK to the u.s
last year he said the company 's earnings were n't disclosed
one of my favorite places to eat at the biltmore . the food is good . and the food is good.
very good food . the food was very good . the service was great and the food is very good.
one of my favorite places to eat and a great breakfast spot . the food is great . the staff is friendly.
took a few friends to join me to the _UNK . i was n't sure what to expect.
one of my favorite places to eat at the biltmore . the food is good , the service was great .
i love the fact that they have a lot took a few friends to join me to the _UNK .
i have been to this location a few times , but i ' ve never been disappointed
let me start by saying that i love the idea of how to describe it .

Table 4: Qualitative comparison between VAE and our proposed approach. First row: PTB samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and WAE-RNF (*lower half*). Second row: Yelp samples generated from prior $p(\mathbf{z})$ by VAE (*upper half*) and WAE-RNF (*lower half*).

perplexity, as it tells us directly how much information we can infer from $\mathbf{x}$ by looking $\mathbf{z}$.

We use Monte Carlo method (Metropolis and Ulam, 1949) to get an approximation of $I(\mathbf{z}, \mathbf{x}) = KL(q(\mathbf{z}, \mathbf{x})||q(\mathbf{z})p(\mathbf{x})) = \mathbb{E}_{p(\mathbf{x})}[KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))] - KL(q(\mathbf{z})||p(\mathbf{z}))$. We compared mutual information between input $\mathbf{x}$ and latent codes $\mathbf{z}$ sampled from Euclidean latent space $\mathcal{Z}$ and Riemannian latent space $\mathcal{Z}'$ respectively. We see that even though NF does not necessarily help WAE to achieve the lowest perplexity, it does make latent codes to preserve more information about the input. For WAE trained with RNF, sample perplexity and mutual information metric are both good. It is cleary that $I(\mathbf{z}', \mathbf{x}) > I(\mathbf{z}, \mathbf{x})$, where $\mathbf{z}'$ is sampled from the curved space, and $\mathbf{z}$ is the sample transformed by the normal planar flow. This further strengthens our confidence over the usefulness of the curved latent space $\mathcal{Z}'$.

Generating Texts from latent spaces Another way to explore latent space is to look at the quality of generated texts. Here we compare sentences generated from methods that do not use Wasserstein objective and RNF with those generated from curved latent space $\mathcal{Z}'$ learned by WAE.

We observe that texts generated from flat Euclidean space are not as diverse as the ones generated from curved space learned by WAE-RNF. This is largely related to the nature of Wasserstein objective. In WAE, the KL-divergence $KL(q||p))$ between the prior and the posterior $q(\mathbf{z}|\mathbf{x})$ conditioned on each input $\mathbf{x}$ does not need to be small to optimize the Wasserstein objective. This indicates that the marginal $q(\mathbf{z})$ is able to match to the prior $p(\mathbf{z})$ while allowing each posterior $q(\mathbf{z}|\mathbf{x})$ to have a much more diverse support than that of a standard Gaussian $p(\mathbf{z})$. Therefore, if we randomly generate samples from curved latent space $\mathcal{Z}'$ many times, we are likely to get samples scattered in different support of distinct posterior conditioned on different input $\mathbf{x}$. Hence, the reconstructed sentences will have a much more diverse meaning or structure.

6 Conclusion

In this paper, we introduced Riemannian Normalizing Flow to train Wasserstein Autoencoder for text modeling. This new model encourages learned latent representation of texts to respect geometric characteristics of input sentences space. Our results show that RNF WAE does significantly improve the language modeling results by modeling the Riemannian geometric space via normalizing flow.

Acknowledgement

We want to thank College of Creative Studies and Gene & Lucas Undergraduate Research Fund for providing scholarships and research opportunities for Prince Zizhuang Wang. We also want to thank Yanxin Feng from Wuhan University for helpful discussion about Riemannian Geometry, and Yunxian He (CMU), Wenhu Chen (UCSB), and Yijun Xiao (UCSB) for their comments which helped us improve our paper and experiments.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *UAI*.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. 2017. From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoNLL*.

Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. 2018. Auto-encoding total correlation explanation. *arXiv preprint arXiv:1802.05822*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. *ICML*.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *JMLR*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. iclr 2015. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Nicholas Metropolis and Stanislaw Ulam. 1949. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICML*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *ICML*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *EMNLP*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2018. Wasserstein auto-encoders. *ICLR*.

Jakub M Tomczak and Max Welling. 2016. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*.

Cédric Villani. 2008. *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

Yijun Xiao, Tiancheng Zhao, and William Yang Wang. 2018. Dirichlet variational autoencoder for text modeling. *arXiv preprint arXiv:1811.00135*.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *EMNLP*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. *EMNLP*.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *ECCV*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017a. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *ACL.*

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017b. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *ACL.*

A Study of Incorrect Paraphrases in Crowdsourced User Utterances

Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah,
Moshe Chai Barukh and **Shayan Zamanirad**
University of New South Wales, Sydney
{m.yaghoubzadehfard,b.benatallah,mosheb,shayanz}@unsw.edu.au

Abstract

Developing bots demands high quality training samples, typically in the form of user utterances and their associated intents. Given the fuzzy nature of human language, such datasets ideally must cover all possible utterances of each single intent. Crowdsourcing has widely been used to collect such inclusive datasets by paraphrasing an initial utterance. However, the quality of this approach often suffers from various issues, particularly language errors produced by unqualified crowd workers. More so, since workers are tasked to write open-ended text, it is very challenging to automatically asses the quality of paraphrased utterances. In this paper, we investigate common crowdsourced paraphrasing issues, and propose an annotated dataset called *Para-Quality*, for detecting the quality issues. We also investigate existing tools and services to provide baselines for detecting each category of issues. In all, this work presents a data-driven view of incorrect paraphrases during the bot development process, and we pave the way towards automatic detection of unqualified paraphrases.

1 Introduction

With the increasing advances in deep learning as well as natural language processing, a new generation of conversational agents is attracting significant attention (Dale, 2016). Also known as *dialogue systems*, *virtual assistants*, *chatbots* or simply *bots* (Campagna et al., 2017; Su et al., 2017), some advanced bots are now designed to perform complex tasks (e.g., *flight booking*), many of which are built using machine learning techniques.

At the heart of building such *task-oriented bots* lies the challenge of accurately capturing the user's intent (e.g., *find cafes in Chicago*), and then extracting its entities to service the request (e.g *term= "cafes", location="Chicago"*). However, its success relies heavily on obtaining both, large and high quality corpora of training samples showing mappings between sample utterances and intents. This is necessary given the ambiguous nature of the human language (Wasow et al., 2005) and large variations of expressions (Wang et al., 2012; Zamanirad et al., 2017).

A lack of variations in training samples can result in incorrect intent detection and consequently execution of undesirable tasks (e.g., booking an expensive hotel instead of a cheap room) (Henderson et al., 2018). Likewise, quality issues in the training samples can lead to unmitigated disasters (Neff and Nagy, 2016) as it happened to Microsoft's *Tay* by making a huge number of offensive commentaries due to biases in the training data (Henderson et al., 2018). It is therefore not surprising that research and development into training data acquisition for bots has received significant consideration (Campagna et al., 2017; Kang et al., 2018).

Collecting training samples usually involves two primary steps: (i) firstly, obtaining an initial utterance for a given user intent (e.g., *find a cafe in Chicago*); and (ii) secondly, paraphrasing this initial expression into multiple variations (Su et al., 2017; Campagna et al., 2017). Paraphrasing is thus vital to cover the variety of ways an expression can be specified (Yang et al., 2018a). As summarized in (McCarthy et al., 2009), a quality paraphrases has three components: semantic completeness, lexical difference, and syntactic difference. To obtain lexically and syntactically diverse paraphrase, crowdsourcing paraphrases has gained popularity in recent years. However, crowdsourced paraphrases need to be checked for quality, given that they are produced by unknown workers with varied skills and motivations (Campagna et al., 2017; Daniel et al., 2018). For example, spammers, malicious and even inexperi-

Proceedings of NAACL-HLT 2019, pages 295–306
Minneapolis, Minnesota, June 2 - June 7, 2019.

enced crowd-workers may provide misleading, erroneous, and semantically invalid paraphrases (Li et al., 2016; Campagna et al., 2017). Quality issues may also stem from misunderstanding the intent or not covering important information such as values of the intent parameters (Su et al., 2017).

The common practice for quality assessment of crowdsourced paraphrases is to design another crowdsourcing task in which workers validate the output from others. However, this approach is costly having to pay for the task twice, making domain-independent automated techniques a very appealing alternative. Moreover, quality control is especially desirable if done before workers submit their paraphrases, since low quality workers can be removed early on without any payment. This can also allow crowdsourcing tasks to provide feedback to users in order to assist them in generating high quality paraphrases (Nilforoshan et al., 2017; Nilforoshan and Wu, 2018). To achieve this, it is therefore necessary to automatically recognize quality issues in crowdsourced paraphrases during the process of bot development.

In this paper, we investigate common paraphrasing errors when using crowdsourcing, and we propose an annotated dataset called *Para-Quality* in which each paraphrase is labelled with the error categories. Accordingly, this work presents a quantitative data-driven study of incorrect paraphrases in bot development process and paves the way towards *enhanced* automated detection of unqualified paraphrased utterances. More specifically, our contributions are two-folded:

- We obtained a sample set of 6000 paraphrases using crowdsourcing. To aim for a broad diversity of samples, the initial expressions were sourced from 40 expressions of highly popular APIs from various domains. Next, we examined and analyzed these samples in order to identify a taxonomy of common paraphrase errors errors (e.g., cheating, misspelling, linguistic errors). Accordingly, we constructed an annotated dataset called *Para-Quality* (using both crowdsourcing and manual verification), in which the paraphrases were labeled with a range of different categorized errors.

- We investigated existing tools and services (e.g., spell and grammar checkers, language identifiers) to detect potential errors. We formulated baselines for each category of errors to determine if they were capable to automatically detect such issues. Our experiments indicate that existing tools often have low precision and recall, and hence our results advocates the need for new approaches in effective detection of paraphrasing issues.

2 Paraphrase Dataset Collection

Various types of paraphrasing issues have been reported in the literature, namely: spelling errors (Braunger et al., 2018), grammatical errors (Jiang et al., 2017; Negri et al., 2012), and missing slot-value (happens when a worker forget to include an entity in paraphrases) (Su et al., 2017). We collected paraphrases for two main reasons: (i) to have a hands-on experience on how incorrect paraphrases are generated, and (ii) to annotate the dataset for building and evaluating paraphrasing quality control systems.

Methodology. We obtained 40 expressions from various domains (i.e. *Yelp, Skyscanner, Spotify, Scopus, Expedia, Open Weather, Amazon AWS, Gamil, Facebook, Bing Image Search*) indexed in ThingPedia (Campagna et al., 2017) and API-KG[1]. We then launched a paraphrasing task on Figure-Eight[2]. Workers were asked to provide three paraphrases for a given expression (Jiang et al., 2017), which is common practice in crowdsourced paraphrasing to reduce repetitive results (Campagna et al., 2017; Jiang et al., 2017). In the provided expression, parameter values were highlighted and crowd-workers were asked to preserve them. Each worker's paraphrases for an initial utterance are normalized by lowercasing and removing punctuation. Next, the initial utterance and the paraphrases are compared to forbid submitting empty strings or repeated paraphrases, and checked if they contain highlighted parameter values (which is also a common practice to avoid missing parameter values) (Mitchell et al., 2014). We collected paraphrases from workers in English speaking countries, and created a dataset containing 6000 paraphrases (2000 triple-paraphrases) in total[3].

3 Common Paraphrasing Issues

To characterize the types of paraphrasing issues, two authors of this paper investigated the crowd-

---

[1] http://apikg.ngrok.io

[2] https://www.figure-eight.com

[3] https://github.com/mysilver/ParaQuality

sourced paraphrases, and recognized 5 primary categories of paraphrasing issues. However, we only considered paraphrase-level issues related to the validity of a paraphrase without considering dataset-level quality issues such as lexical diversity (Negri et al., 2012) and bias (Henderson et al., 2018).

3.1 Spelling Errors

Misspelling has been reported as one of most common mistakes in paraphrasing (Inaba et al., 2015; Wang et al., 2012; Chklovski, 2005; Braunger et al., 2018). In our sample set, we also noticed misspellings were generated both intentionally (as an act of cheating to quickly generate a paraphrase such as Example 2 in Table 1) and unintentionally (due to a lack of knowledge or a simple mistake such as Example 3 in Table 1).

3.2 Linguistic Errors

Linguistic errors are also common in crowdsourced natural language collections (Jiang et al., 2017; Negri et al., 2012). Verb errors, preposition errors, vocabulary errors (improper word substitutions), and incorrect singular/plural nouns, just to name a few. Moreover, capitalization and article errors seems abundant (e.g., Example 5 in Table 1). Given that real bot users also make such errors, it is important to have linguistically incorrect utterances in the training samples (Bapat et al., 2018). However, at a very least, detecting linguistic errors can contribute to quality-aware selection of crowd workers.

3.3 Semantic Errors

This occurs when a paraphrase deviates from the meaning of the initial utterance (e.g., *find cafes in Chicago*). As reported in various studies, workers may forget to mention parameter values (also known as missing slot)(e.g., *find cafes*)[4] (Crossley et al., 2016; Su et al., 2017; Ravichander et al., 2017; Wang et al., 2012; Braunger et al., 2018), provide wrong values (e.g., *find cafes in Paris*) (Su et al., 2017; Ravichander et al., 2017; Wang et al., 2012; Negri et al., 2012; Braunger et al., 2018), or add unmentioned parameter values(Wang et al., 2012) (e.g., *find two cafes in Chicago*). Workers may also incorrectly use a singular noun instead of its plural form, and vice versa. For instance, in Example 6 of Table 1, the paraphrase only asks for the status of one specific *burglar alarm* while the expression asks for the status of all *burglar alarms*. Making mistakes in paraphrasing complementary forms of words also exists in the crowdsourced dataset. For instance, in Example 7 of Table 1, assuming that the bot answers the question only by saying "YES" or "NO", the answer for the paraphrase differs from that of the expression. However, it will make no difference if the bot's response is more descriptive (e.g., "it's working", "it isn't working".) Finally, some paraphrases significantly diverge from expressions. For instance, in Example 8 of Table 1, the intent of paraphrase is to turn off the TV; however, that of initial utterance is to query about the TV status.

3.4 Task Misunderstanding

In some cases, workers misunderstood the task and provided translations in their own native languages (referred to as *Translation* issues) (Crossley et al., 2016; Braunger et al., 2018; Bapat et al., 2018), and some mistakenly thought they should provide answers for expressions phrased as questions (referred to as *Answering* issues) such as Example 9 in Table 1. This occurred even though workers were provided with comprehensive instructions and examples. We infer that some workers did not read the instructions, ignoring the possibility of cheating.

3.5 Cheating

In crowdsourced tasks, collecting paraphrases is not immune to unqualified workers, cheaters, or spammers (Daniel et al., 2018; Crossley et al., 2016; Chklovski, 2005).

Detecting malicious behaviour is vital because even constructive feedback may not guarantee quality improvements as workers act carelessly on purpose. *Cheating* is thus considered a special case of *Semantic Error* which is done intentionally. It is difficult even for experts to detect if someone is cheating or unintentionally making mistakes. However, it becomes easier when we consider all three paraphrases written by a worker for a given expression at once. For example, in Example 10 of Table 1, the malicious worker removes words one by one to generate new paraphrases. In this example, we also notice that it is still possible that a cheater produces a valid paraphrase accidentally such as the first paraphrase in Example 10. Workers may also start providing

---

[4] In our task design, this type of error cannot happen since parameter values are checked using regular expressions before submission

#	Label		Sample
1	Correct	**Expression**	*Create a public playlist named new_playlist*
		Paraphrase	▷ Make a public playlist named new_playlist
2	Spelling Errors	**Expression**	*Estimate the taxi fare from the airport to home*
		Paraphrase	▷ Estimate the taxi fare from the airport to hom
3	Spelling Errors	**Expression**	*Estimate the taxi fare from the airport to home*
		Paraphrase	▷ Tell me about the far from airport to home
4	Spelling Errors	**Expression**	*Where should I try coffee near Newtown?*
		Paraphrase	▷ Find cafes near Newtown
5	Linguistic Errors	**Expression**	*Estimate the taxi fare from the airport to home*
		Paraphrase	▷ How much for taxi from airport to home
6	Semantic Errors	**Expression**	*Are the burglar alarms in the office malfunctioning?*
		Paraphrase	▷ Is the burglar alarm faulty in our work place?
7	Semantic Errors	**Expression**	*Are the burglar alarms in the office malfunctioning?*
		Paraphrase	▷ Are the office alarms working?
8	Semantic Errors	**Expression**	*Is the TV in the house off?*
		Paraphrase	▷ Can you turn off the TV in the house if it's on?
9	Task Misunderstanding	**Expression**	*Estimate the taxi fare from the airport to home?*
		Paraphrase	▷ Airport to home is $50
10	Cheating	**Expression**	*Request a taxi from airport to home*
		Paraphrases	▷ A taxi from airport to home ▷ Taxi from airport to home ▷ From airport to home
11	Cheating	**Expression**	*I want reviews for McDonald at Kensington st.*
		Paraphrases	▷ I want reviews g for McDonald at Kensington st. ▷ I want for reviews for McDonald at Kensington st. ▷ I want reviegws for McDonald at Kensington st.
12	Cheating	**Expression**	*I want reviews for McDonald at Kensington st.*
		Paraphrases	▷ I want to do reviews for McDonald's in Kensington st. ▷ I would like to do reviews for McDonald's in Kensington st. ▷ I could do reviews for McDonald's in Kensington st.
13	Cheating	**Expression**	*Create a public playlist named NewPlaylist*
		Paraphrases	▷ That song hits the public NewPlaylist this year ▷ Public really loved that NewPlaylist played on the event ▷ Public saw many NewPlaylist this year
14	Cheating	**Expression**	*Estimate the taxi fare from the airport to home*
		Paraphrases	▷ What is the fare of taxi from airport to home ▷ Tell me about fare from airport to home ▷ You have high taxi fare airport to home

Table 1: Paraphrase Samples

faulty paraphrases after generating some correct paraphrases as shown in Example 14 of Table 1. Based on our observations, the simplest mode of cheating is to add a few random characters to the source sentence as shown in Example 11. Next is adding a few words to the source sentence without much editing as shown in Example 12. Finally, there are cheaters who rewrite and change the sentences substantially in a very random way such as Example 13.

4 Dataset Annotation

Next, we designed another crowdsourcing task to annotate the collected paraphrases according to the category of issues devised above. Namely, using following labels: *Correct*, *Semantic Error*, *Misspelling*, *Linguistic Error*, *Translation*, *Answering*, and *Cheating*. We split the category of misunderstanding issues into *Translation* and *Answering* because they require different methods to detect.

Methodology. In the annotation task, crowd workers were instructed to label each paraphrase with the paraphrasing issues. Next, to further increase the quality of annotations [5], two authors of this paper manually re-annotated the paraphrases to resolve disagreements between crowd annotators. Moreover, contradictory labels (e.g., a paraphrase cannot be labeled both *Correct* and *Misspelling* simultaneously) were checked to ensure consistency. The overall Kappa test showed a high agreement coefficient between the annotators (McHugh, 2012) by Kappa being 0.85. Table 2 also shows the pair-wise inter-annotator agree-

---

[5] because of weak agreement between crowd workers

Label	**Kappa**
Correct	0.900
Misspelling	0.972
Linguistic Errors	0.879
Translation	1.000
Answering	0.855
Cheating	0.936
Semantic Errors	0.833

Table 2: Pairwise Inter-Annotator Agreement

ment (Cohen, 1960). Next, the authors discussed and revised the re-annotated labels to further increase the quality of annotations by discussing and resolving disagreements.

Statistics. Figure 1 shows the frequencies of each label in the crowdsourced paraphrases as well as their co-occurrences in an UpSet plot (Lex et al., 2014) using Intervene (Khan and Mathelier, 2017). Accordingly we infer that only 61% of paraphrases are labeled *Correct*. This plot also shows how many times two labels co-occurred. For example, all paraphrases which are labeled *Translation* (24 times), are also labeled *Cheating*[6].

5 Automatic Error Detection

Automatically detecting paraphrasing issues, especially when done during the crowd task, can minimize the cost of crowdsourcing by eliminating malicious workers, reducing the number of erroneous paraphrases, and eliminating the need for launching another crowdsourced validation task. Moreover, by detecting *Misspelling* and *Linguistic Errors*, users can be provided with proper feedback to help them improve the quality of paraphrasing by showing the source of error and suggestions to address the error (e.g., "*Spelling error detected: articl → article*"). Detecting *Semantic Errors*, such as missing parameter values, can also help crowd workers to generate high quality correct paraphrases. Automated methods can also be used to identify low quality workers, and particularly cheaters who may generate potentially large amount of invalid paraphrases intentionally. Moreover, providing suggestions to cheaters will not help and therefore early detection is of paramount.

Spell Checker	**Precision**	**Recall**	**F1**
Aspell[8]	0.249	0.618	0.354
Hunspell[9]	0.249	0.619	0.355
MySpell[10]	0.249	0.619	0.355
Norvig[11]	0.488	0.655	0.559
Ginger[12]	0.540	0.719	0.616
Yandex[13]	0.571	0.752	0.650
Bing Spell Check[14]	0.612	**0.737**	0.669
LanguageTool[15]	**0.630**	0.727	**0.674**

Table 3: Comparison of Spell Checkers

In a pre-hoc quality control approach for crowdsourced paraphrases, the most important metric seems to be the *precision* of detecting invalid paraphrases (Nilforoshan et al., 2017). That is because the main aim of using such a quality control approach is rejecting invalid paraphrases without rejecting correct ones (Burrows et al., 2013). This is essential because rejecting correct paraphrases would be unfair and unproductive. For instance, sincere and trustful crowd workers might not get paid as a result of false-positives (incorrectly detected errors). On the other hand, having a high *recall* in detecting invalid paraphrases is important to eliminate faulty paraphrases and consequently obtain robust training samples.

Moreover, such a quality control technique should ideally be domain-independent, accessible, and easily-operated to minimize the cost of customization for a special domain and requiring paid experts (e.g., an open source pre-built machine learning model). In the rest of this section, we examine current tools and approaches and discuss their effectiveness in assessing the paraphrasing issues.

5.1 Spelling Errors

We employed several spell checkers as listed in Table 3 to examine if they are effective in recognizing spelling errors. We looked up Wikipedia, Github, and ProgrammableWeb[7] to find available tools and APIs for this purpose.

---

[6] We used *Google Translate* to check whether they were proper translations or just random sentences in other languages

[7] https://www.programmableweb.com

[8] http://aspell.net/

[9] http://hunspell.github.io/

[10] http://www.openoffice.org/lingucomponent/dictionary.html

[11] https://github.com/barrust/pyspellchecker

[12] https://www.gingersoftware.com/ grammarcheck

[13] https://tech.yandex.ru/speller/

[14] https://azure.microsoft.com/en-us/services/cognitive-services/spell-check/

[15] https://languagetool.org

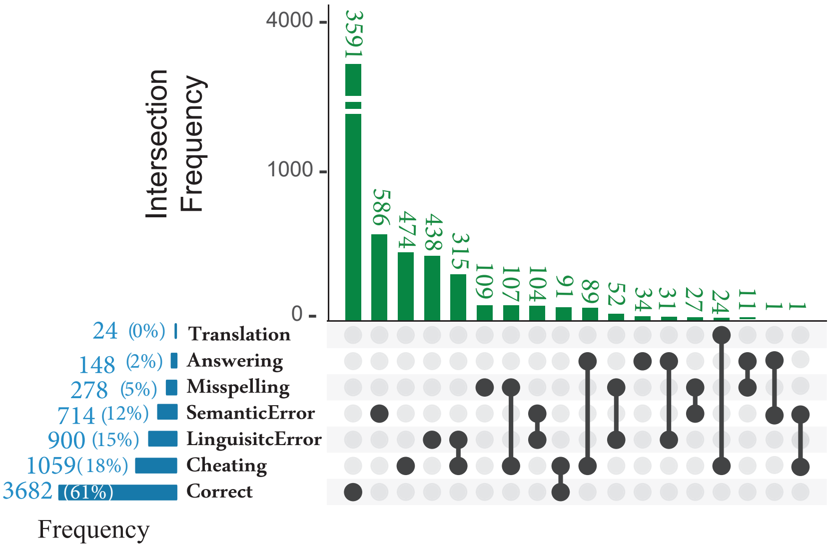


Figure 1: Dataset Label Statistics

Even though detecting misspelled words seems easy with existing automatic spellcheckers, they fall short in a few cases. This can be also concluded from Table 3 by considering the precision and recall of each spell checker in detecting only paraphrases with misspellings. For instance, spell checkers are often unable to identify homonyms (Perelman, 2016), incorrectly mark proper nouns and unusual words (Bernstein et al., 2015), and sometimes do not identify wrong words that are properly spelled (Chisholm and Henry, 2005). For instance, in Example 1 of Table 1, the "*new_playlist*" is incorrectly detected as a misspelled word by LanguageTool (the best performer as listed in Table 3). In Example 3, the word "*far*" is not detected even though the worker has misspelled the word "*fare*". In Example 4, the word "Newtown" (a suburb in Sydney) is mistakenly detected as a misspelling error. Some of these deficiencies can be addressed. For instance, in the case of spelling errors, assuming that the initial expressions given to the crowd are free of typos, we can ignore false-positives like the "Newtown" and "new_playlist".

5.2 Linguistic Errors

We investigated how well grammar checkers perform in detecting linguistic errors. We employed several grammar checkers as listed in Table 4.

Our experiments shows that spell checkers have both low precision and recall. Perelman (Perelman, 2016) also conducted several experiments with major commercial and non-commercial grammar checkers, and identified that grammar checkers are unreliable. Based on our observations, grammar checkers often fail in detecting linguistic errors as shown in Table 4. Examples include improper use of words (e.g., *"Who is the latest scientific article of machine learning?"*), random sequence of words generated by cheaters (e.g., *"Come the next sing"*), and missing articles[16] (e.g., *"I'm looking for flight for Tehran to Sydney"*). Given these examples, we believe that language models can be used to measure the likelihood of a sequence of words to detect if it is linguistically acceptable.

Grammar Checker	**Precision**	**Recall**	**F1**
AfterDeadline[17]	0.228	0.069	0.106
Ginger	0.322	**0.256**	**0.285**
GrammarBot[18]	0.356	0.139	0.200
LanguageTool	**0.388**	0.098	0.156

Table 4: Comparison of Grammar Checkers

5.3 Translation

We also investigated several language detectors to evaluate how well they perform when crowd workers use another language instead of English. The results of experiment in Table 5 indicate that these tools detect almost all sentences in other languages. But they produce lots of false-positives including for correct English sentences (e.g., *"play next song"*). As a result, the tools in our experi-

---

[16] Missing articles in expressions similar to newspaper headlines are not considered error in the dataset (e.g., *"Hotel near Disneyland"*)

[17] https://www.afterthedeadline.com

[18] https://www.grammarbot.io

Language Detector	Precision	Recall	F1
FastText[19]	0.072	**1.000**	0.135
LangDetect[20]	0.080	0.917	0.147
LanguageIdentifier[21]	0.080	0.917	0.147
IBM Watson[22]	0.170	0.958	0.289
DetectLanguage[23]	0.344	0.917	0.500
DetectLanguage+	**0.909**	**1.000**	**0.952**

Table 5: Language Detection

ment have low precision in detecting languages as shown in Table 5. Most of the false-positives are caused by sentences that contain unusual words such as misspellings and named entities in the sentence (e.g., Email *Phil* saying "I got you").

One possible approach to improve the precision of such tools and APIs is to check if a given paraphrase has spelling errors prior to using language detection tools. We therefore extended the *DetectLanguage* (the best performing tool) by adding a constraint: a sentence is not written in another language unless it has at least two spelling errors. This constraint is based on the assumption that spell checkers treat foreign words as spelling errors and a sentence has at least two words to be called a sentence. This approach (*DetectLanguage*+ in Table 5) significantly reduced the number of false-positives and thus improved precision.

5.4 Answering

Dialog Acts (DAs) (Jurafsky and Martin, 2018), also known as speech acts, represent general intents of an utterance. DA tagging systems label utterances with a predefined set of utterance types (*Directive, Commissive, Informative, etc* (Mezza et al., 2018).) Based on the fact that DAs must remain consistent during paraphrasing, we employed a state-of-art, domain-independent, pre-trained DA tagger proposed in (Mezza et al., 2018). For example, if an initial utterance is a *question* (e.g., *are there any cafes nearby?*) it is acceptable to paraphrase it into a *directive* sentence (e.g., *find cafes nearby.*), but its speech act cannot be *informative* (e.g., there is a cafe on the corner.). Overall, due to the lack of any other domain-independent DA tagger for the English language, we only investigated this tagger. We found that it has a precision of 2% with recall of 63%. This shows that detecting speech acts is a very challenging task especially for domain-independent environments.

Advances in speech act detection and availability of public speech act datasets can assist in detecting this category of the paraphrasing issues. Moreover, it is feasible to automatically generate pairs of questions and answers by mining datasets in the fields of Question Answering and dialog systems. Automatically building such pairs can help building a dataset which is diverse enough to be used in practice. Such a dataset can be fed into deep learning algorithms to yield better performance in detecting *Answering* issues.

5.5 Semantic Errors & Cheating

To the best our knowledge, there is not yet an approach to distinguish between categories of semantically invalid paraphrases. Paraphrase detection and textual semantic similarity (STS) methods are designed to measure how two pieces of text are semantically similar. However, they do not differentiate between different types of errors (e.g., *Cheating*, *Answering*, *Semantic Errors*) in our settings. As such, these techniques are not directly applicable. In the rest of this section, we focus on building machine learning models to detect the paraphrasing errors.

For this purpose, we used 38 established features from the literature as summarized in Table 6. Using these features and Weka (Hall et al., 2009), we built various classifiers to detect the following paraphrasing issues: *Answering*, *Semantic Errors*, and *Cheating*. We chose to test the five classification algorithms applied in paraphrasing literature as mentioned in (Burrows et al., 2013): C4.5 Decision Tree, K-Nearest Neighbor (K=50), Maximum Entropy, Naive Bayes, and Support Vector Machines (SVM) using default Weka 3.6.13 parameters for each of the classification algorithms. We also experimented with Random Forest algorithm since it is a widely-used classifier. We did not apply deep learning based classifiers directly due to the lack of expressions in the collected dataset which seems essential for developing domain independent classifiers. While our dataset is reasonably large, it contains only 40 expressions (each having 150 paraphrases). Given that deep learn-

---

[19] https://fasttext.cc/blog/2017/10/02/blog-post.html(Joulin et al., 2017)

[20] https://pypi.org/project/langdetect/

[21] https://github.com/saffsd/langid.py

[22] https://console.bluemix.net/apidocs/language-translator#identify-language

[23] https://ws.detectlanguage.com

Category	**#**	**Description**
N-gram Features	12	N-gram overlap, exclusive longest common prefix n-gram overlap, and SUMO all proposed in (Joao et al., 2007), as well as Gaussian, Parabolic, and Trigonometric proposed in (Cordeiro et al., 2007), Paraphrase In N-gram Changes (PINC) (Chen and Dolan, 2011), Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002), Google's BLEU (GLEU) (Wu et al., 2016), NIST (Doddington, 2002), Character n-gram F-score (CHRF) (Popović, 2016), and the length of the longest common subsequence.
Semantic Similarity	15	Semantic Textual Similarity (Fakouri-Kapourchali et al., 2018), Word Mover's Distance (Kusner et al., 2015) between words embeddings of expression and paraphrase, cosine similarity and euclidean distance between vectors of expression and paraphrase generated by Sent2Vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), Universal Sentence Encoder (Cer et al., 2018), Concatenated Power Mean Embeddings (Rücklé et al., 2018), tenses of sentences, pronoun used in the paraphrase, and miss-matched named entities
Others	11	Number of spelling and grammatical errors detected by LanguageTool, task completion time, edit distance, normalized edit distance, word-level edit distance (Fakouri-Kapourchali et al., 2018), length difference between expression and paraphrase (in characters and words), and simple functions to detect questions, imperative sentences, and answering.

Table 6: Summary of Feature Library

Classifier	**Precision**	**Recall**	**F1**
Random Forest	**0.947**	0.129	0.226
Maximum Entropy	0.564	0.157	0.246
Decision Tree	0.527	**0.350**	**0.421**

Table 7: Automatic *Answering* Detection

Classifier	**Precision**	**Recall**	**F1**
Random Forest	**0.798**	0.120	0.209
Decision Tree	0.377	0.276	**0.319**
Naive Bayes	0.171	**0.783**	0.280

Table 8: Automatic *Semantic Error* Detection

Classifier	**Precision**	**Recall**	**F1**
SVM	**0.878**	0.223	0.356
K-Nearest Neighbor	0.871	0.248	0.386
Random Forest	0.843	0.546	**0.663**
Maximum Entropy	0.756	0.440	0.557
Decision Tree	0.632	**0.566**	0.597
Naive Bayes	0.473	0.426	0.449

Table 9: Automatic *Cheating* Detection

ing techniques are data thirsty (Goodfellow et al., 2016; Yang et al., 2018a), to use these kinds of models and eliminate the burden of manual feature engineering, much more expressions are needed. Instead, we benefited from the state-of-art sentence encoders via Transfer Learning as listed in Table 6.

Table 7, 8, and 9 demonstrate the performance of various classifiers (excluding classifiers with F1 being less than 0.2) for each of paraphrasing issues using 10-fold cross validation. To keep the classifiers domain-independent, we split the dataset based on the expressions without sharing any paraphrases of a single expression between the test and train samples. It can be seen that automatically detecting these quality issues is very challenging; even the best performing classifier has a very low F1 score especially for detecting *Answering* and *Semantic Error* issues. Based on manual exploration, we also found that the classifiers fail to recognize complex cheating behaviours such as Example 13 in Table 1 as discussed in Section 3. Therefore, new approaches are required to accurately detect paraphrasing issues. Based on our explorations and a prior work (McCarthy et al., 2009), we postulate that accurately detecting linguistic errors such as grammatically incorrect paraphrases can play indispensable role in detecting cheating behaviours. Moreover, advances in measuring semantic similarity between sentences can help differentiate between semantically invalid paraphrases and correct ones.

5.6 Incorrect Paraphrases Detection

We also assessed the performance of detecting incorrect paraphrases regardless of their categories. In this setting, we labeled all incorrect sentences with a single label (*"Incorrect"*) regardless of their categories. Table 10 demonstrates the performance of various classifiers. Detecting incorrect paraphrases is useful for post-hoc quality control to remove incorrect paraphrases after crowdsourcing paraphrases and consequently eliminate the need for crowdsourced validation task.

6 Related Work

To the best of our knowledge, that our work is the first to categorize paraphrasing issues and propose

Classifier	Precision	Recall	F1
K-Nearest Neighbor	**0.799**	0.341	0.478
Random Forest	0.781	**0.551**	**0.646**
Maximum Entropy	0.721	0.489	0.583
SVM	0.709	0.289	0.411
Decision Tree	0.633	0.585	0.608
Naive Bayes	0.574	0.557	0.565

Table 10: Automatic Incorrect Paraphrase Detection

an annotated dataset for assessing quality issues of paraphrased user expressions. Nevertheless, our work is related to the areas of (i) quality control in crowdsourced natural language datasets; and (ii) semantic similarity.

Quality Control. Quality can be assessed after or before data acquisition. While *post-hoc* methods evaluate quality when all paraphrases are collected, *pre-hoc* methods can prevent submission of low quality paraphrases during crowdsourcing. The most prevalent *post-hoc* approach is launching a verification task to evaluate crowdsourced paraphrases (Negri et al., 2012; Tschirsich and Hintz, 2013). However, automatically removing misspelled paraphrases (Wang et al., 2012) and discarding submissions from workers with low/high task completion time (Ma et al., 2017) are also applied in literature. Machine learning models have also been explored in plagiarism detection systems to assure quality of crowdsourced paraphrases (Crossley et al., 2016; Burrows et al., 2013).

Pre-hoc methods, on the other hand, rely on online approaches to asses the quality of the data provided during crowdsourcing (Nilforoshan et al., 2017). Sophisticated techniques are required to avoid generation of erroneous paraphrases (e.g., automatic feedback generation was used to assist crowd workers in generating high quality paraphrases). Precog (Nilforoshan et al., 2017) is an example of such tools which is based on a supervised method for generating automatic writing feedback for multi-paragraph text– designed mostly for crowdsourced product reviews (Nilforoshan et al., 2017; Nilforoshan and Wu, 2018). This paper aims for paving the way for building automatic *pre-hoc* approaches, and providing appropriate online feedback to users to assist them in generating appropriate paraphrases. However, the provided dataset can also be used for building *post-hoc* methods to automatically omit faulty paraphrases.

Semantic Similarity. Measuring similarity between units of text plays an important role in Natural Language Processing (NLP). Several NLP tasks have been designed to cover various aspects and usages of textual similarity. Examples include textual entailment, semantic textual similarity (Yang et al., 2018b; Fakouri-Kapourchali et al., 2018), paraphrase detection (Agarwal et al., 2018; Issa et al., 2018), duplicate question detection (Mannarswamy and Chidambaram, 2018) tasks which are studied well in NLP. Moreover, recent success in sentence encoders (e.g., Sent2Vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), Universal Sentence Encoder (Cer et al., 2018), and Concatenated Power Mean Embeddings (Rücklé et al., 2018)) can be exploited to detect paraphrasing issues with more accuracy. These techniques can be borrowed with some domain specific considerations to build automatic quality control systems for detecting low quality paraphrases.

7 Conclusion

In this paper, we employed a data-driven approach to investigate and quantitatively study various crowdsourced paraphrasing issues. We discussed how automatic techniques for detecting various quality issues can assist the manual process of crowdsourced paraphrasing. We collected an annotated dataset of crowdsourced paraphrasing in which each paraphrase is labeled with associated paraphrasing issues. We used this dataset to assess existing tools and techniques and to determine whether they are sufficient for automatically detecting such issues. Our experiments revealed that automated detection of errors in paraphrases is a challenging task. As a future work, we will be working on devising automated-assisted methods for detection of paraphrasing issues. This will be based on a two-way feedback mechanism: generating feedback for workers, while at the same time the system learns from the (data of) users to improve its machine intelligence. In time, we envision increasingly less dependence on users.

acknowledgements

This research was supported fully by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP1601104515).

References

Basant Agarwal, Heri Ramampiaro, Helge Langseth, and Massimiliano Ruocco. 2018. A deep network model for paraphrase detection in short text messages. *Information Processing & Management*, 54(6):922–937.

Rucha Bapat, Pavel Kucherbaev, and Alessandro Bozzon. 2018. Effective crowdsourced generation of training data for chatbots natural language understanding. In *Web Engineering*, Cham. Springer International Publishing.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2015. Soylent: A word processor with a crowd inside. volume 58, pages 85–94, New York, NY, USA. ACM.

Patricia Braunger, Wolfgang Maier, Jan Wessling, and Maria Schmidt. 2018. Towards an automatic assessment of crowdsourced data for nlu. In *LREC*.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. volume 4, pages 43:1–43:21, New York, NY, USA. ACM.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S. Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 341–350, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 190–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wendy A. Chisholm and Shawn Lawton Henry. 2005. Interdependent components of web accessibility. In *Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, W4A '05, pages 31–37, New York, NY, USA. ACM.

Timothy Chklovski. 2005. Collecting paraphrase corpora from volunteer contributors. In *Proceedings of the 3rd International Conference on Knowledge Capture*, K-CAP '05, pages 115–120, New York, NY, USA. ACM.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Joao Cordeiro, Gael Dias, and Pavel Brazdil. 2007. A metric for paraphrase detection. In *Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on*, pages 7–7. IEEE.

Scott Crossley, Luc Paquette, Mihai Dascalu, Danielle S. McNamara, and Ryan S. Baker. 2016. Combining click-stream data with nlp tools to better understand mooc completion. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, LAK '16, pages 6–14, New York, NY, USA. ACM.

Robert Dale. 2016. The return of the chatbots. *Natural Language Engineering*, 22(5):811–817.

Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. 2018. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. volume 51, pages 7:1–7:40, New York, NY, USA. ACM.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Roghayeh Fakouri-Kapourchali, Mohammad-Ali Yaghoub-Zadeh-Fard, and Mehdi Khalili. 2018. Semantic textual similarity as a service. In *Service Research and Innovation*, pages 203–215, Cham. Springer International Publishing.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. volume 11, pages 10–18, New York, NY, USA. ACM.

Peter Henderson, Koustuv Sinha, Nicolas Angelard-Gontier, Nan Rosemary Ke, Genevieve Fried, Ryan Lowe, and Joelle Pineau. 2018. Ethical challenges in data-driven dialogue systems. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and*

Society, AIES '18, pages 123–129, New York, NY, USA. ACM.

Michimasa Inaba, Naoyuki Iwata, Fujio Toriumi, Takatsugu Hirayama, Yu Enokibori, Kenichi Takahashi, and Kenji Mase. 2015. Statistical response method and learning data acquisition using gamified crowdsourcing for a non-task-oriented dialogue agent. In *Revised Selected Papers of the 6th International Conference on Agents and Artificial Intelligence - Volume 8946*, ICAART 2014, pages 119–136, Berlin, Heidelberg. Springer-Verlag.

Fuad Issa, Marco Damonte, Shay B Cohen, Xiaohui Yan, and Yi Chang. 2018. Abstract meaning representation for paraphrase detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 442–452.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding task design trade-offs in crowdsourced paraphrase collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 103–109. Association for Computational Linguistics.

Cordeiro Joao, Dias Gaël, and Brazdil Pavel. 2007. New functions for unsupervised asymmetrical paraphrase detection. *Journal of Software*, 2(4):12–23.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

D Jurafsky and JH Martin. 2018. Dialog systems and chatbots. *Speech and language processing*.

Yiping Kang, Yunqi Zhang, Jonathan K Kummerfeld, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, volume 3, pages 33–40.

Aziz Khan and Anthony Mathelier. 2017. Intervene: a tool for intersection and visualization of multiple gene or genomic region sets. *BMC bioinformatics*, 18(1):287.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. 2014. Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992.

Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2296–2319.

Xiao Ma, Trishala Neeraj, and Mor Naaman. 2017. A computational approach to perceived trustworthiness of airbnb host profiles.

Sandya Mannarswamy and Saravanan Chidambaram. 2018. Geminio: Finding duplicates in a question haystack. In *Advances in Knowledge Discovery and Data Mining*, pages 104–114, Cham. Springer International Publishing.

Philip M. McCarthy, Rebekah H. Guess, and Danielle S. McNamara. 2009. The components of paraphrase evaluations. *Behavior Research Methods*, 41(3):682–690.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282.

Stefano Mezza, Alessandra Cervone, Evgeny Stepanov, Giuliano Tortoreto, and Giuseppe Riccardi. 2018. Iso-standard domain-independent dialogue act tagging for conversational agents. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3539–3551. Association for Computational Linguistics.

Margaret Mitchell, Dan Bohus, and Ece Kamar. 2014. Crowdsourcing language generation templates for dialogue systems. *Proceedings of the INLG and SIGDIAL 2014 Joint Session*, pages 172–180.

Gina Neff and Peter Nagy. 2016. Automation, algorithms, and politics— talking to bots: symbiotic agency and the case of tay. *International Journal of Communication*, 10:17.

Matteo Negri, Yashar Mehdad, Alessandro Marchetti, Danilo Giampiccolo, and Luisa Bentivogli. 2012. Chinese whispers: Cooperative paraphrase acquisition. In *LREC*, pages 2659–2665.

Hamed Nilforoshan, Jiannan Wang, and Eugene Wu. 2017. Precog: Improving crowdsourced data quality before acquisition. *CoRR*, abs/1704.02384.

Hamed Nilforoshan and Eugene Wu. 2018. Leveraging quality prediction models for automatic writing feedback. In *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM 2018, Stanford, California, USA, June 25-28, 2018.*, pages 211–220.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Les Perelman. 2016. Grammar checkers do not work. *WLN: A Journal of Writing Center Scholarship*, 40(7-8):11–20.

Maja Popović. 2016. chrf deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504. Association for Computational Linguistics.

Abhilasha Ravichander, Thomas Manzini, Matthias Grabmair, Graham Neubig, Jonathan Francis, and Eric Nyberg. 2017. How would you say it? eliciting lexically diverse dialogue for supervised semantic parsing. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 374–383.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *CoRR*, abs/1803.01400.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. 2017. Building natural language interfaces to web apis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 177–186, New York, NY, USA. ACM.

Martin Tschirsich and Gerold Hintz. 2013. Leveraging crowdsourcing for paraphrase recognition. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 205–213.

William Yang Wang, Dan Bohus, Ece Kamar, and Eric Horvitz. 2012. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 73–78. IEEE.

Thomas Wasow, Amy Perfors, and David Beaver. 2005. The puzzle of ambiguity. *Morphology and the web of grammar: Essays in memory of Steven G. Lapointe*, pages 265–282.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jie Yang, Thomas Drake, Andreas Damianou, and Yoelle Maarek. 2018a. Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 23–32, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2018b. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174. Association for Computational Linguistics.

Shayan Zamanirad, Boualem Benatallah, Moshe Chai Barukh, Fabio Casati, and Carlos Rodriguez. 2017. Programming bots by synthesizing natural language expressions into api invocations. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 832–837. IEEE Press.

ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters

Abdalghani Abujabal[1], Rishiraj Saha Roy[2], Mohamed Yahya[3] and Gerhard Weikum[2]
[1]Amazon Alexa, Aachen, Germany
`abujabaa@amazon.de`
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
`{rishiraj, weikum}@mpi-inf.mpg.de`
[3]Bloomberg L.P., London, United Kingdom
`yahya.mohamed@gmail.com`

Abstract

To bridge the gap between the capabilities of the state-of-the-art in factoid question answering (QA) and what users ask, we need large datasets of real user questions that capture the various question phenomena users are interested in, and the diverse ways in which these questions are formulated. We introduce *ComQA*, a *large* dataset of *real* user questions that exhibit different *challenging aspects* such as *compositionality*, *temporal reasoning*, and *comparisons*. ComQA questions come from the WikiAnswers community QA platform, which typically contains questions that are not satisfactorily answerable by existing search engine technology. Through a large crowdsourcing effort, we clean the question dataset, group questions into *paraphrase clusters*, and annotate clusters with their answers. ComQA contains $11,214$ questions grouped into 4,834 paraphrase clusters. We detail the process of constructing ComQA, including the measures taken to ensure its high quality while making effective use of crowdsourcing. We also present an extensive analysis of the dataset and the results achieved by state-of-the-art systems on ComQA, demonstrating that our dataset can be a driver of future research on QA.

Cluster 1
temporal
Q: ***"Who was the Britain's leader during WW1?"***
Q: ***"Who ran Britain during WW1?"***
Q: ***"Who was the leader of Britain during World War One?"***
A: [https://en.wikipedia.org/wiki/h._h._asquith, https://en.wikipedia.org/wiki/david_lloyd_george]

Cluster 2
comparison
Q: ***"largest city located along the Nile river?"***
Q: ***"largest city by the Nile river?"***
Q: ***"What is the largest city in Africa that is on the banks of the Nile river?"*** A: [https://en.wikipedia.org/wiki/cairo]

Cluster 3
compositional
Q: ***"John Travolta and Jamie Lee Curtis acted in this film?"***
Q: ***"Jamie Lee Curtis and John Travolta played together in this movie?"***
Q: ***"John Travolta and Jamie Lee Curtis were actors in this film?"*** A: [https://en.wikipedia.org/wiki/perfect_(film)]

Cluster 4
empty answer
Q: ***"Who is the first human landed in Mars?"***
Q: ***"Who was the first human being on Mars?"***
Q: ***"first human in Mars?"***
A: []

Figure 1: ComQA paraphrase clusters covering a range of question aspects e.g., temporal and compositional questions, with lexical and syntactic diversity.

1 Introduction

Factoid QA is the task of answering natural language questions whose answer is one or a small number of entities (Voorhees and Tice, 2000). To advance research in QA in a manner consistent with the needs of end users, it is important to have access to datasets that reflect *real* user information needs by covering various *question phenomena* and the wide *lexical and syntactic variety* in expressing these information needs. The benchmarks should be large enough to facilitate the use of data-hungry machine learning methods. In this paper, we present ComQA, a large dataset of 11,214 real user questions collected from the WikiAnswers community QA website. As shown in Figure 1, the dataset contains various question phenomena. ComQA questions are grouped into 4,834 paraphrase clusters through a large-scale crowdsourcing effort, which capture lexical and syntactic variety. Crowdsourcing is also used to pair paraphrase clusters with answers to serve as a supervision signal for training and as a basis for evaluation.

Table 1 contrasts ComQA with publicly available QA datasets. The foremost issue that ComQA tackles is ensuring research is driven by information needs formulated by real users. Most large-scale datasets resort to highly-templatic synthetically generated natural language questions (Bordes et al., 2015; Cai and Yates, 2013; Su et al.,

---

[1] The main part of this work was carried out when the author was at the Max Planck Institute for Informatics.

Proceedings of NAACL-HLT 2019, pages 307–317
Minneapolis, Minnesota, June 2 - June 7, 2019.

Dataset	Large scale (> 5K)	Real Information Needs	Complex Questions	Question Paraphrases
ComQA (This paper)	✓	✓	✓	✓
Free917 (Cai and Yates, 2013)	✗	✗	✗	✗
WebQuestions (Berant et al., 2013)	✓	✓	✗	✗
SimpleQuestions (Bordes et al., 2015)	✓	✗	✗	✗
QALD (Usbeck et al., 2017)	✗	✗	✓	✗
LC-QuAD (Trivedi et al., 2017)	✓	✗	✓	✗
ComplexQuestions (Bao et al., 2016)	✗	✓	✓	✗
GraphQuestions (Su et al., 2016)	✓	✗	✓	✓
ComplexWebQuestions (Talmor and Berant, 2018)	✓	✗	✓	✗
TREC (Voorhees and Tice, 2000)	✗	✓	✓	✗

Table 1: Comparison of ComQA with existing QA datasets over various dimensions.

2016; Talmor and Berant, 2018; Trivedi et al., 2017). Other datasets utilize search engine logs to collect their questions (Berant et al., 2013), which creates a bias towards simpler questions that search engines can already answer reasonably well. In contrast, ComQA questions come from WikiAnswers, a community QA website where users pose questions to be answered by other users. This is often a reflection of the fact that such questions are beyond the capabilities of commercial search engines and QA systems. Questions in our dataset exhibit a wide range of interesting aspects such as the need for temporal reasoning (Figure 1, cluster 1), comparison (Figure 1, cluster 2), compositionality (multiple subquestions with multiple entities and relations) (Figure 1, cluster 3), and unanswerable questions (Figure 1, cluster 4).

ComQA is the result of a carefully designed large-scale crowdsourcing effort to group questions into paraphrase clusters and pair them with answers. Past work has demonstrated the benefits of paraphrasing for QA (Abujabal et al., 2018; Berant and Liang, 2014; Dong et al., 2017; Fader et al., 2013). Motivated by this, we judiciously use crowdsourcing to obtain clean paraphrase clusters from WikiAnswers' noisy ones, resulting in ones like those shown in Figure 1, with both lexical and syntactic variations. The only other dataset to provide such clusters is that of Su et al. (2016), but that is based on synthetic information needs.

For answering, recent research has shown that combining various resources for answering significantly improves performance (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016). Therefore, we do not pair ComQA with a specific knowledge base (KB) or text corpus for answering. We call on the research community to innovate in combining different answering sources to tackle ComQA and advance research in QA. We use crowdsourcing to pair paraphrase clusters with answers. ComQA answers are primarily Wikipedia entity URLs. This has two motivations: (i) it builds on the example of search engines that use Wikipedia entities as answers for entity-centric queries (e.g., through knowledge cards), and (ii) most modern KBs ground their entities in Wikipedia. Wherever the answers are temporal or measurable quantities, we use TIMEX3[1] and the International System of Units[2] for normalization. Providing canonical answers allows for better comparison of different systems.

We present an extensive analysis of ComQA, where we introduce the various question aspects of the dataset. We also analyze the results of running state-of-the-art QA systems on ComQA. ComQA exposes major shortcomings in these systems, mainly related to their inability to handle compositionality, time, and comparison. Our detailed error analysis provides inspiration for avenues of future work to ensure that QA systems meet the expectations of real users. To summarize, in this paper we make the following contributions:

- We present a dataset of 11,214 real user questions collected from a community QA website. The questions exhibit a range of aspects that are important for users and challenging for existing QA systems. Using crowdsourcing, questions are grouped into 4,834 paraphrase clusters that are annotated with answers. ComQA is available at: `http://qa.mpi-inf.mpg.de/comqa`.
- We present an extensive analysis and quantify the various difficulties in ComQA. We also present the results of state-of-the art QA systems on ComQA, and a detailed error analysis.

---

[1] `http://www.timeml.org`

[2] `https://en.wikipedia.org/wiki/SI`

2 Related Work

There are two main variants of the factoid QA task, with the distinction tied to the underlying answering resources and the nature of answers. Traditionally, QA has been explored over large textual corpora (Cui et al., 2005; Harabagiu et al., 2001, 2003; Ravichandran and Hovy, 2002; Saquete et al., 2009) with answers being textual phrases. Recently, it has been explored over large structured resources such as KBs (Berant et al., 2013; Unger et al., 2012), with answers being semantic entities. Recent work demonstrated that the two variants are complementary, and a combination of the two results in the best performance (Sun et al., 2018; Xu et al., 2016).

QA over textual corpora. QA has a long tradition in IR and NLP, including benchmarking tasks in TREC (Voorhees and Tice, 2000; Dietz and Gamari, 2017) and CLEF (Magnini et al., 2004; Herrera et al., 2004). This has predominantly focused on retrieving answers from textual sources (Ferrucci, 2012; Harabagiu et al., 2006; Prager et al., 2004; Saquete et al., 2004; Yin et al., 2015). In IBM Watson (Ferrucci, 2012), structured data played a role, but text was the main source for answers. The TREC QA evaluation series provide hundreds of questions to be answered over documents, which have become widely adopted benchmarks for answer sentence selection (Wang and Nyberg, 2015). ComQA is orders of magnitude larger than TREC QA.

Reading comprehension (RC) is a recently introduced task, where the goal is to answer a question from a *given* textual paragraph (Kociský et al., 2017; Lai et al., 2017; Rajpurkar et al., 2016; Trischler et al., 2017; Yang et al., 2015). This setting is different from factoid QA, where the goal is to answer questions from a large repository of data (be it textual or structured), and not a single paragraph. A recent direction in RC is dealing with unanswerable questions from the underlying data (Rajpurkar et al., 2018). ComQA includes such questions to allow tackling the same problem in the context of factoid QA.

QA over KBs. Recent efforts have focused on natural language questions as an interface for KBs, where questions are translated to structured queries via semantic parsing (Bao et al., 2016; Bast and Haussmann, 2015; Fader et al., 2013; Mohammed et al., 2018; Reddy et al., 2014; Yang et al., 2014; Yao and Durme, 2014; Yahya et al., 2013). Over the past five years, many datasets were introduced for this setting. However, as Table 1 shows, they are either small in size (Free917, and ComplexQuestions), composed of synthetically generated questions (SimpleQuestions, GraphQuestions, LC-QuAD and ComplexWebQuestions), or are structurally simple (WebQuestions). ComQA addresses these shortcomings. Returning semantic entities as answers allows users to further explore these entities in various resources such as their Wikipedia pages, Freebase entries, etc. It also allows QA systems to tap into various interlinked resources for improvement (e.g., to obtain better lexicons, or train better NER systems). Because of this, ComQA provides semantically grounded reference answers in Wikipedia (without committing to Wikipedia as an answering resource). For numerical quantities and dates, ComQA adopts the International System of Units and TIMEX3 standards, respectively.

3 Overview

In this work, a factoid question is a question whose answer is one or a small number of entities or literal values (Voorhees and Tice, 2000) e.g., *"Who were the secretaries of state under Barack Obama?"* and *"When was Germany's first post-war chancellor born?"*.

3.1 Questions in ComQA

A question in our dataset can exhibit *one or more* of the following phenomena:

- **Simple:** questions about a single property of an entity (e.g., *"Where was Einstein born?"*)
- **Compositional:** A question is compositional if answering it requires answering more primitive questions and combining these. These can be *intersection* or *nested* questions. Intersection questions are ones where two or more subquestions can be answered independently, and their answers intersected (e.g., *"Which films featuring Tom Hanks did Spielberg direct?"*). In nested questions, the answer to one subquestion is necessary to answer another (*"Who were the parents of the thirteenth president of the US?"*).
- **Temporal:** These are questions that require temporal reasoning for deriving the answer, be it explicit (e.g., *'in 1998'*), implicit (e.g., *'during the WWI'*), relative (e.g., *'current'*), or latent (e.g. *'Who is the US president?'*). Temporal questions also include those *whose answer*

is an explicit temporal expression (*"When did Trenton become New Jersey's capital?"*).

- **Comparison:** We consider three types of comparison questions: *comparatives* (*"Which rivers in Europe are longer than the Rhine?"*), *superlatives* (*"What is the population of the largest city in Egypt?"*), and *ordinal* questions (*"What was the name of Elvis's first movie?"*).
- **Telegraphic** (Joshi et al., 2014): These are short questions formulated in an informal manner similar to keyword queries (*"First president India?"*). Systems that rely on linguistic analysis often fail on such questions.
- **Answer tuple**: Where an answer is a tuple of connected entities as opposed to a single entity (*"When and where did George H. Bush go to college, and what did he study?"*).

3.2 Answers in ComQA

Recent work has shown that the choice of answering resource, or the combination of resources significantly affects answering performance (Savenkov and Agichtein, 2016; Sun et al., 2018; Xu et al., 2016). Inspired by this, ComQA is not tied to a specific resource for answering. To this end, answers in ComQA are primarily Wikipedia URLs. This enables QA systems to combine different answering resources which are linked to Wikipedia (e.g., DBpedia, Freebase, YAGO, Wikidata, etc). This also allows seamless comparison across these QA systems. An answer in ComQA can be:

- **Entity:** ComQA entities are grounded in Wikipedia. However, Wikipedia is inevitably incomplete, so answers that cannot be grounded in Wikipedia are represented as plain text. For example, the answer for *"What is the name of Kristen Stewart adopted brother?"* is {Taylor Stewart, Dana Stewart}.
- **Literal value:** Temporal answers follow the TIMEX3 standard. For measurable quantities, we follow the International System of Units.
- **Empty:** In the factoid setting, some questions can be based on a false premise, and hence, are unanswerable e.g., *"Who was the first human being on Mars?"* (no human has been on Mars, yet). The correct answer to such questions is the empty set. Such questions allow systems to cope with these cases. Recent work has started looking at this problem (Rajpurkar et al., 2018).

4 Dataset Construction

Our goal is to collect factoid questions that represent real information needs and cover a range of question aspects. Moreover, we want to have different paraphrases for each question. To this end, we tap into the potential of community QA platforms. Questions posed there represent real information needs. Moreover, users of those platforms provide (noisy) annotations around questions e.g., paraphrase clusters. In this work, we exploit the annotations where users mark questions as duplicates as a basis for paraphrase clusters, and clean those. Concretely, we started with the WikiAnswers crawl by Fader et al. (2014). We obtained ComQA from this crawl primarily through a large-scale crowdsourcing effort, which we describe in what follows.

The original resource curated by Fader et al. contains 763M questions. Questions in the crawl are grouped into 30M paraphrase clusters based on feedback from WikiAnswers users. This clustering has a low accuracy (Fader et al., 2014). Extracting factoid questions and cleaning the clusters are thus essential for a high-quality dataset.

4.1 Preprocessing of WikiAnswers

To remove non-factoid questions, we filtered out questions that (i) start with *'why'*, or (ii) contain words like *(dis)similarities, differences, (dis)advantages*, etc. Questions matching these filters are out of scope as they require a narrative answer. We also removed questions with less than three or more than twenty words, as we found these to be typically noisy or non-factoid questions. This left us with about 21M questions belonging to 6.1M clusters.

To further focus on factoid questions, we automatically classified questions into one or more of the following *four classes*: (1) temporal, (2) comparison, (3) single entity, and (4) multi-entity questions. We used SUTime (Chang and Manning, 2012) to identify temporal questions and the Stanford named entity recognizer (Finkel et al., 2005) to detect named entities. We used part-of-speech patterns to identify comparatives, superlatives, and ordinals. Clusters which did not have questions belonging to any of the above classes were discarded from further consideration. Although these clusters contain false negatives e.g., *"What official position did Mendeleev hold until his death?"* due to errors by the tagging tools,

"Who was henry VII son?"
"Who was henry's vii sons?"

"Henry VII of England second son?"

"When did henry 7th oldest son die?"

"Who was Henry vii's oldest son?"
"Who is king henry VII eldest son?"
"What was the name of Henry VII first son?"
"Who was henry vII eldest son?"
"What was henry's vii oldest son?"
"Who was the oldest son of Henry VII?"

Figure 2: A WikiAnswers cluster split into four clusters by AMT Turkers.

most discarded questions are out-of-scope.

Manual inspection. We next applied the first stage of human curation to the dataset. Each WikiAnswers cluster was assigned to one of the four classes above based on the majority label of the questions within. We then randomly sampled 15K clusters from each of the four classes (60K clusters in total with 482K questions) and sampled a representative question from each of these clusters at random (60K questions). We relied on the assumption that questions within the same cluster are semantically equivalent. These 60K questions were manually examined by the authors and those with unclear or non-factoid intent were removed along with the cluster that contains them. We thus ended up with 2.1K clusters with 13.7K questions.

4.2 Curating Paraphrase Clusters

We inspected a random subset of the 2.1K WikiAnswers clusters and found that questions in the same cluster are semantically *related but not necessarily equivalent*, which is in line with observations in previous work (Fader et al., 2014). Dong et al. (2017) reported that 45% of question pairs were related rather than genuine paraphrases. For example, Figure 2 shows 10 questions in the same WikiAnswers cluster. Obtaining accurate paraphrase clusters is crucial to any systems that want to utilize them (Abujabal et al., 2018; Berant and Liang, 2014; Dong et al., 2017). We therefore utilized crowdsourcing to clean the Wikianswers paraphrase clusters. We used Amazon Mechanical Turk (AMT) to identify semantically equivalent questions within a WikiAnswers cluster, thereby obtaining cleaner clusters for ComQA. Once we had the clean clusters, we set up a second AMT task to collect answers for each ComQA cluster.

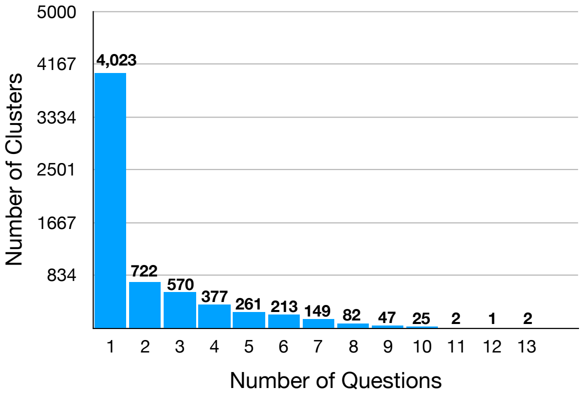


Figure 3: The distribution of questions in clusters.

Task design. We had to ensure the simplicity of the task to obtain high quality results. Therefore, rather than giving workers a WikiAnswers cluster and asking them to partition it into clusters of paraphrases, we showed them pairs of questions from a cluster and asked them to make the binary decision of whether the two questions are paraphrases. To reduce potentially redundant annotations, we utilized the transitivity of the paraphrase relationship. Given a WikiAnswers cluster $Q = \{q_1, ..., q_n\}$, we proceed in rounds to form ComQA clusters. In the first round, we collect annotations for each pair (q_i, q_{i+1}). The majority annotation among five annotators is taken. An initial clustering is formed accordingly, with clusters sharing the same question merged together (to account for transitivity). This process continues iteratively until no new clusters can be formed from Q.

Task statistics. We obtained annotations for 18,890 question pairs from 175 different workers. Each pair was shown to five different workers, with 65.7% of the pairs receiving unanimous agreement, 21.4% receiving four agreements and 12.9% receiving three agreements. By design, with five judges and binary annotations, no pair can have less three agreements. This resulted in questions being placed in paraphrase clusters, and no questions were discarded at this stage. At the end of this step, the original 2.1K WikiAnswers clusters became 6.4K ComQA clusters with a total of 13.7K questions. Figure 3 shows the distribution of questions in clusters.

To test whether relying on the transitivity of the

Property	**Example**	**Percentage%**
Compositional questions		
Conjunction	*“What is the capital of the country whose northern border is Poland and Germany?”*	17.67
Nested	*“When is Will Smith’s oldest son’s birthday?”*	14.33
Temporal questions		
Explicit time	*“Who was the winner of the World Series **in 1994?**”*	4.00
Implicit time	*“Who was Britain’s leader **during WW1?**”*	4.00
Temporal answer	*“**When** did Trenton become New Jersey’s capital?”*	15.67
Comparison questions		
Comparative	*“Who was the first US president to serve **2 terms**?”*	1.00
Superlative	*“What ocean does the **longest** river in the world flow into?”*	14.33
Ordinal	*“When was Thomas Edisons **first** wife born?”*	14.00
Question formulation		
Telegraphic	*“Neyo first album?”*	8.00
Entity distribution in questions		
Zero entity	*“What public company has the most employees in the world?”*	2.67
Single entity	*“Who is **Brad Walst**’s wife?”*	75.67
Multi-entity	*“What country in **South America** lies between **Brazil** and **Argentina**?”*	21.67
Other features		
Answer tuple	*“**Where** was Peyton Manning born and **what year was he born**?”*	2.00
Empty answer	*“Who was Calgary’s first woman mayor?”*	3.67

Table 2: Results of the manual analysis of 300 questions. Note that properties are not mutually exclusive.

paraphrase relationship is suitable to reduce the annotation effort, we asked annotators to annotate 1,100 random pairs (q_1, q_3), where we had already received positive annotations for the pairs (q_1, q_2) and (q_2, q_3) being paraphrases of each other. In 93.5% of the cases there was agreement. Additionally, as experts on the task, the authors manually assessed 600 pairs of questions, which serve as honeypots. There was 96.6% agreement with our annotations. An example result of this task is shown in Figure 2, where Turkers split the original WikiAnswers cluster into the four clusters shown.

4.3 Answering Questions

We were now in a position to obtain an answer annotation for each of the 6.4K clean clusters.

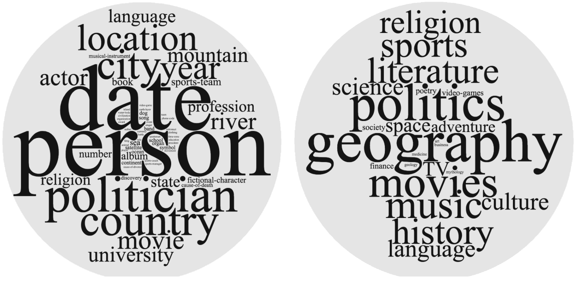
Task design. To collect answers, we designed another AMT task, where workers were shown a representative question randomly drawn from a cluster. Workers were asked to use the Web to find answers and to provide the corresponding URLs of Wikipedia entities. Due to the inevitable incompleteness of Wikipedia, workers were asked to provide the surface form of an answer entity if it does not have a Wikipedia page. If the answer is a full date, workers were asked to follow dd-mmm-yyyy format. For measurable quantities, workers were asked to provide units. We use TIMEX3 and the international system of units for normalizing temporal answers and measurable quantities e.g., *‘12th century’* to `11XX`. If no answer is found, workers were asked to type in *‘no answer’*.

Task statistics. Each representative question was shown to three different workers. An answer is deemed correct if it is common between at least two workers. This resulted in 1.6K clusters (containing 2.4K questions) with no agreed-upon answers, which were dropped. For example, *“Who was the first democratically elected president of Mexico?”* is subjective. Other questions received related answers e.g., *“Who do the people in Iraq worship?”* with `Allah`, `Islam` and `Mohamed` as answers from the three annotators. Other questions were underspecified e.g., *“Who was elected the vice president in 1796?”*. At the end of the task, we ended up with 4,834 clusters with 11,214 question-answer pairs, which form ComQA.

5 Dataset Analysis

In this section, we present a manual analysis of 300 questions sampled at random from the ComQA dataset. This analysis helps understand the different aspects of our dataset. A summary of the analysis is presented in Table 2.

Question categories. We categorized each question as either simple or complex. A question is complex if it belongs to one or more of the compositional, temporal, or comparison classes. 56.33% of the questions were complex; 32% compositional, 23.67% temporal, and 29.33% contain



(a) Answer types (b) Question topics

Figure 4: Answer types and question topics on 300 annotated examples as word clouds.

comparison conditions. A question may contain multiple conditions (*"What country has the **highest** population in the year **2008**?"* with comparison and temporal conditions).

We also identified questions of telegraphic nature e.g., *"Julia Alvarez's parents?"*, with 8% of our questions being telegraphic. Such questions pose a challenge for systems that rely on linguistic analysis of questions (Joshi et al., 2014).

We counted the number of named entities in questions: 23.67% contain two or more entities, reflecting their compositional nature, and 2.67% contain no entities e.g., *"What public company has the most employees in the world?"*. Such questions can be hard as many methods assume the existence of a pivot entity in a question.

Finally, 3.67% of the questions are unanswerable, e.g., *"Who was the first human being on Mars?"*. Such questions incentivise QA systems to return non-empty answers only when suitable. In Table 3 we compare ComQA with other current datasets based on real user information needs over different question categories.

Answer types. We annotated each question with the most fine-grained context-specific answer type (Ziegler et al., 2017). Answers in ComQA belong to a diverse set of types that range from coarse (e.g., `person`) to fine (e.g., `sports manager`). Types also include literals such as `number` and `date`. Figure 4(a) shows answer types of the 300 annotated examples as a word cloud.

Question topics. We annotated questions with topics to which they belong (e.g., geography, movies, sports). These are shown in Figure 4(b), and demonstrate the topical diversity of ComQA.

Question length. Questions in ComQA are fairly long, with a mean length of 7.73 words, indicating the compositional nature of questions.

6 Experiments

In this section we present experimental results for running ComQA through state-of-the-art QA systems. Our experiments show that these systems achieve humble performance on ComQA. Through a detailed analysis, this performance can be attributed to systematic shortcomings in handling various question aspects in ComQA.

6.1 Experimental Setup

Splits. We partition ComQA into a random train/dev/test split of 70/10/20% with 7,850, 1,121 and 2,243 questions, respectively.

Metrics. We follow the community's standard evaluation metrics: we compute average precision, recall, and F1 scores across all test questions. For unanswerable questions whose correct answer is the empty set, we define precision and recall to be 1 for a system that returns an empty set, and 0 otherwise (Rajpurkar et al., 2018).

6.2 Baselines

We evaluated two categories of QA systems that differ in the underlying answering resource: either KBs or textual extractions. We ran the following systems: (i) Abujabal et al. (2017), which automatically generates templates using question-answer pairs; (ii) Bast and Haussmann (2015), which instantiates hand-crafted query templates followed by query ranking; (iii) Berant and Liang (2015), which relies on agenda-based parsing and imitation learning; (iv) Berant et al. (2013), which uses rules to build queries from questions; and (v) Fader et al. (2013), which maps questions to queries over open vocabulary facts extracted from Web documents. Note that our intention is not to assess the quality of these systems, but to assess how challenging ComQA is.

The systems were trained with ComQA data. All systems were run over the data sources for which they were designed. The first four baselines are over Freebase. We therefore mapped ComQA answers (Wikipedia entities) to the corresponding Freebase names using the information stored with entities in Freebase. We observe that the Wikipedia answer entities have no counterpart in Freebase for 7% of the ComQA questions. This suggests an oracle F1 score of 93.0. For Fader et al. (2013), which is over web extractions, we mapped Wikipedia URLs to their titles.

Dataset	**Size**	**Compositional**	**Temporal**	**Comparison**	**Telegraphic**	**Empty Answer**
ComQA	11,214	32%	24%	30%	8%	4%
WebQuestions (Berant et al., 2013)	5,810	2%	7%	2%	0%	0%
ComplexQuestions (Bao et al., 2016)	2,100	39%	34%	9%	0%	0%

Table 3: Comparison of ComQA with existing datasets over various phenomena. We manually annotated 100 random questions from each dataset.

	Avg. Prec	**Avg. Rec**	**Avg. F1**
Abujabal et al. (2017)	21.2	38.4	22.4
Bast and Haussmann (2015)	20.7	37.6	21.6
Berant and Liang (2015)	10.7	15.4	10.6
Berant et al. (2013)	13.7	20.1	12.0
Fader et al. (2013)	7.22	6.59	6.73

Table 4: Results of baselines on ComQA test set.

	WebQuestions F1	**Free917 Accuracy**	**ComQA F1**
Abujabal et al. (2017)	51.0	78.6	22.4
Bast and Haussmann (2015)	49.4	76.4	21.6
Berant and Liang (2015)	49.7	–	10.6
Berant et al. (2013)	35.7	62.0	12.0

Table 5: Results of baselines on different datasets.

6.3 Results

Table 4 shows the performance of the baselines on the ComQA test set. Overall, the systems achieved poor performance, suggesting that current methods cannot handle the complexity of our dataset, and that new models for QA are needed. Table 5 compares the performance of the systems on different datasets (Free917 uses accuracy as a quality metric). For example, while Abujabal et al. (2017) achieved an F1 score of 51.0 on WebQuestions, it achieved 22.4 on ComQA.

The performance of Fader et al. (2013) is worse than the others due to the incompleteness of its underlying extractions and the complexity of ComQA questions that require higher-order relations and reasoning. However, the system answered some complex questions, which KB-QA systems failed to answer. For example, it answered *"What is the highest mountain in the state of Washington?"*. The answer to such a question is more readily available in Web text, compared to a KB, where more sophisticated reasoning is required to handle the superlative. However, a slightly modified question such as *"What is the* ***fourth*** *highest mountain in the state of Washington?"* is unlikely to be found in text, but be answered using KBs with the appropriate reasoning. Both examples above demonstrate the benefits of combining text and structured resources.

6.4 Error Analysis

For the two best performing systems on ComQA, QUINT (Abujabal et al., 2017) and AQQU (Bast and Haussmann, 2015), we manually inspected 100 questions on which they failed. We classified failure sources into four categories: compositionality, temporal, comparison or NER. Table 6 shows the distribution of these failure sources.

Compositionality. Neither system could handle the compositional nature of questions. For example, they returned the father of Julius Caesar as an answer for *"What did Julius Caesar's father work as?"*, while, the question requires another KB predicate that connects the father to his profession. For *"John Travolta and Jamie Lee Curtis starred in this movie?"*, both systems returned movies with Jamie Lee Curtis, ignoring the constraint that John Travolta should also appear in them. Properly answering multi-relation questions over KBs remains an open problem.

Temporal. Our analysis reveals that both systems fail to capture temporal constraints in questions, be it explicit or implicit. For *"Who won the Oscar for Best Actress in 1986?"*, they returned all winners and ignored the temporal restriction from *'in 1986'*. Implicit temporal constraints like named events (e.g., *'Vietnam war'* in *"Who was the president of the US during Vietnam war?"*) pose a challenge to current methods. Such constraints need to be detected first and normalized to a canonical time interval (November 1st, 1955 to April 30th, 1975, for the Vietnam war). Then, systems need to compare the terms of the US presidents with the above interval to account for the temporal relation of *'during'*. While detecting explicit time expressions can be done reasonably well using existing time taggers (Chang and Manning, 2012), identifying implicit ones is difficult. Furthermore, retrieving the correct temporal scopes of entities in questions (e.g., the terms of the US presidents) is hard due to the large number of temporal KB predicates associated with entities.

Comparison. Both systems perform poorly on comparison questions, which is expected since they were not designed to address those. To the best of our knowledge, no existing KB-QA system can handle comparison questions. Note that our goal is not to assess the quality the of current methods, but to highlight that these methods miss categories of questions that are important to real users. For *"What is the first film Julie Andrews made?"* and *"What is the largest city in the state of Washington?"*, both systems returned the list of Julie Andrews's films and the list of Washington's cities, for the first and the second questions, respectively. While the first question requires the attribute of `filmReleasedIn` to order by, the second needs the attribute of `hasArea`. Identifying the correct attribute to order by as well as determining the order direction (ascending for the first and descending for the second) is challenging and out of scope for current methods.

NER. NER errors come from false negatives, where entities are not detected. For example, in *"On what date did the Mexican Revolution end?"* QUINT identified *'Mexican'* rather than *'Mexican Revolution'* as an entity. For *"What is the first real movie that was produced in 1903?"*, which does not ask about a specific entity, QUINT could not generate SPARQL queries. Existing QA methods expect a pivotal entity in a question, which is not always the case.

Note that while baseline systems achieved low precision, they achieved higher recall (21.2 vs 38.4 for QUINT, respectively) (Table 4). This reflects the fact that these systems often cannot cope with the full complexity of ComQA questions, and instead end up evaluating underconstrained interpretations of the question.

To conclude, current methods can handle simple questions very well, but struggle with complex questions that involve multiple conditions on different entities or need to join the results from sub-questions. Handling such complex questions, however, is important if we are to satisfy information needs expressed by real users.

7 Conclusion

We presented ComQA, a dataset for QA that harnesses a community QA platform, reflecting questions asked by real users. ComQA contains 11,214 question-answer pairs, with questions grouped into paraphrase clusters through crowdsourcing. Questions exhibit different aspects that current QA systems struggle with. ComQA is a challenging dataset that is aimed at driving future research on QA, to match the needs of real users.

Category	**QUINT**	**AQQU**
Compositionality error	39%	43%
Missing comparison	31%	26%
Missing temporal constraint	19%	22%
NER error	11%	9%

Table 6: Distribution of failure sources on ComQA questions on which QUINT and AQQU failed.

Acknowledgments

We would like to thank Tommaso Pasini for his helpful feedback.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. In *WWW*, pages 1053–1062.

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*, pages 1191–1200.

Jun-Wei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *COLING*, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *CIKM*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*, pages 1415–1425.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *TACL*, 3:545–558.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433.

Angel X. Chang and Christopher D. Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *SIGIR*, pages 400–407.

Laura Dietz and Ben Gamari. 2017. TREC CAR: A data set for complex answer retrieval. version 1.4.(2017).

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *EMNLP*, pages 875–886.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*, pages 1156–1165.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618.

David A. Ferrucci. 2012. This is watson. *IBM Journal of Research and Development*, 56(3):1.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370.

Sanda M. Harabagiu, V. Finley Lacatusu, and Andrew Hickl. 2006. Answering complex questions with random walk models. In *SIGIR*, pages 220–227.

Sanda M. Harabagiu, Steven J. Maiorano, and Marius Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Sanda M. Harabagiu, Dan I. Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C. Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *ACL*, pages 274–281.

Jesús Herrera, Anselmo Peñas, and Felisa Verdejo. 2004. Question answering pilot task at CLEF 2004. In *CLEF*, pages 581–590.

Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP*, pages 1104–1114.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *CoRR*, abs/1712.07040.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *EMNLP*, pages 785–794.

Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Peñas, Maarten de Rijke, Paulo Rocha, Kiril Ivanov Simov, and Richard F. E. Sutcliffe. 2004. Overview of the CLEF 2004 multilingual question answering track. In *CLEF*, pages 371–391.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *NAACL-HLT*, pages 291–296.

John M. Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. 2004. Question answering using constraint satisfaction: Qa-by-dossier-with-contraints. In *ACL*, pages 574–581.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Deepak Ravichandran and Eduard H. Hovy. 2002. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, pages 377–392.

Estela Saquete, José Luis Vicedo González, Patricio Martínez-Barco, Rafael Muñoz, and Hector Llorens. 2009. Enhancing QA systems with complex temporal question processing capabilities. *J. Artif. Intell. Res.*

Estela Saquete, Patricio Martínez-Barco, Rafael Muñoz, and José Luis Vicedo González. 2004. Splitting complex temporal questions for question answering systems. In *ACL*, pages 566–573.

Denis Savenkov and Eugene Agichtein. 2016. When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data. In *SIGIR*, pages 235–244.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *EMNLP*, pages 562–572.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *EMNLP*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*, pages 641–651.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Rep4NLP@ACL*, pages 191–200.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC*, pages 210–218.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *WWW*, pages 639–648.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th Open Challenge on Question Answering over Linked Data (QALD-7). In *SemWebEval*.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *SIGIR*, pages 200–207.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, pages 707–712.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *ACL*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM*, pages 1107–1116.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018.

Xuchen Yao and Benjamin Van Durme. 2014. Information Extraction over Structured Data: Question Answering with Freebase. In *ACL*, pages 956–966.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *CIKM*, pages 1301–1310.

David Ziegler, Abdalghani Abujabal, Rishiraj Saha Roy, and Gerhard Weikum. 2017. Efficiency-aware answering of compositional questions using answer type prediction. In *IJCNLP*, pages 222–227.

FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase

Kelvin Jiang[*] [†] and **Dekun Wu**[*] and **Hui Jiang**
Department of Electrical Engineering and Computer Science
York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
`{kelvin,jackwu,hj}@eecs.yorku.ca`

Abstract

In this paper, we present a new data set, named *FreebaseQA*, for open-domain factoid question answering (QA) tasks over structured knowledge bases, like Freebase. The data set is generated by matching trivia-type question-answer pairs with subject-predicate-object triples in Freebase. For each collected question-answer pair, we first tag all entities in each question and search for relevant predicates that bridge a tagged entity with the answer in Freebase. Finally, human annotation is used to remove false positives in these matched triples. Using this method, we are able to efficiently generate over 54K matches from about 28K unique questions with minimal cost. Our analysis shows that this data set is suitable for model training in factoid QA tasks since FreebaseQA provides more linguistically sophisticated questions than other existing data sets. The data set is available for free download at `http://github.com/infinitecold/FreebaseQA`.

1 Introduction

Within the field of natural language processing (NLP), there has been an increase in developments towards various real-world applications, such as factoid question answering (QA): the process of obtaining the answer(s) to a factual question - similar to trivia game settings. For this task to be successfully completed, there are several steps that need to occur. Notably, we need to interpret and parse the question, determine the domain of relevance, eliminate ambiguities if existent, and pinpoint the exact answer to the question asked. Fortunately, this task has been simplified with the emergence of large knowledge graphs, including Freebase (Bollacker et al., 2008), from where we can retrieve information. Knowledge graphs are colossal networks of data that describe concepts, entities, and their relations. In fact, Freebase is the largest publicly-available knowledge graph, consisting of 4 million nodes and approximately 3 billion edges (Google, 2017). Each node represents an entity existing in the physical world, such as a person, a location, or an organization. Each edge represents a relation between two entities, in a directed manner from a *subject* node to an *object* node. In Freebase, these edges are referred to as *predicates*, and a collection of a subject-predicate-object is referred to as a *triple*. An example triple in Freebase is the subject `Clarissa`, predicate `book.written_work.author` and object `Samuel Richardson`, explaining that the book Clarissa is written by author Samuel Richardson. Specifically, we take advantage of these relations between entities, which describe facts, to help with factoid question answering. We believe open-domain factoid QA over structured knowledge graphs like Freebase is a very interesting NLP task since it opens up many interesting real-world applications, such as natural language based query and search. Finally, once questions are formulated using a variety of rich and sophisticated representations in natural languages, such factoid QA tasks may serve as an excellent testbed to study many natural language understanding problems, e.g., examine the recently emerging research efforts to combine neural models with the traditional symbolic processing methods (Liang et al., 2017; Mou et al., 2017).

On the other hand, machine learning approaches for NLP are data hungry since they require large amounts of real-world data to train the models for the best possible performance. Existing data sets for the factoid QA task over structured knowledge bases are either too small in scale to train neural networks effectively, or contain questions

---

[*] Equal contribution.

[†] Currently at University of Waterloo. Work was done at York University.

Proceedings of NAACL-HLT 2019, pages 318–323
Minneapolis, Minnesota, June 2 - June 7, 2019.

that are too simple in linguistic structure to amply cover real-world scenarios. In this paper, we introduce a new data set for open-domain QA over Freebase, called *FreebaseQA*, which is created by matching trivia-type question-answer pairs with Freebase triples that reflect the semantic meaning of the questions. FreebaseQA contains over 54K matches from about 28K unique questions that can be used to train machine learning (ML) models and help the development of factoid QA systems for more realistic applications. Particularly, these matches may be used to train ML models to align natural language questions with Freebase predicates to search for the correct answers in Freebase. Our analysis shows that FreebaseQA provides an advantage over all pre-existing data sets with similar objectives, which are either too small or only contain questions that are too simple in linguistic structure. These results will be explained in detail in Section 4.

2 Related Work

Factoid QA data sets involving question-answer pairs as well as their corresponding Freebase matches have been created in the past. In (Berant et al., 2013), factoid QA over knowledge graphs are formulated as semantic parsing problems, where each natural language question is first converted into a logic form to retrieve the answer with traditional symbolic approaches. In (Berant et al., 2013), a small-scale data set of several thousands of question-answer pairs, called WebQuestions, is created by human annotators. In (Yih et al., 2016), the WebQuestions set is further refined by providing human-annotated semantic parses for some questions that are answerable using Freebase, which is called WebQuestionsSP (WebQSP). Recently, deep learning approaches have become popular in the field of NLP. Neural networks require far more training data than a small data set of several thousands of samples. In (Bordes et al., 2015), a much larger QA data set of about 100K question/answer pairs, called SimpleQuestions, is created. In this work, some randomly chosen Freebase triples are shown to human annotators. For each given triple, an annotator is asked to manually compose a question to reflect the relation in the triple. The issues with SimpleQuestions lie in that most constructed questions are quite simple in linguistic structure and many questions even directly use the keywords in the Freebase predicates since human annotators may be greatly limited in composition when a particular triple is shown. According to (Petrochuk and Zettlemoyer, 2018), SimpleQuestions is nearly solved with only standard neural network methods if its linguistic ambiguity is taken into account. In (Vlad Serban et al., 2016), a large QA data set is automatically generated by neural networks but it obviously lacks rich linguistic variations. Additionally, many similar factoid QA data sets are also released for other non-English languages, e.g. WebQA in (Li et al., 2016). Meanwhile, another direction of data collection efforts involve QA in various reading comprehension tasks, e.g. SQuAD in (Rajpurkar et al., 2016), MS-MARCO in (Nguyen et al., 2016), TriviaQA in (Joshi et al., 2017). However, we believe question answering over structured knowledge graphs remains a viable NLP task for the promising research direction to combine neural computing methods with the traditional symbolic processing approaches.

3 Constructing the FreebaseQA Data Set

In this section, we outline the construction procedure of the FreebaseQA data set, which consists of about 54K matches in the form of two examples shown in Table 1.

3.1 Preparation of Question-Answer Pairs

In FreebaseQA, we have not generated any new question-answer pairs but we have instead collected pre-composed trivia-type factoid questions from a number of sources. Unlike SimpleQuestions, these questions are independently composed for human contestants in various trivia-like competitions. As a result, these questions show much richer linguistic variation and complexity than almost all existing KB-QA data sets. In particular, we use the TriviaQA (Joshi et al., 2017) data set as the primary source of our QA pairs, while also including questions scraped from the trivia websites, KnowQuiz (`http://www.knowquiz.com`), QuizBalls (`http://www.quizballs.com`), and QuizZone (`https://www.quiz-zone.co.uk`). We remove duplicate entries and the remaining pairs are consolidated into a single source.

Each question is then run through two named entity recognition (NER) systems: TAGME (Ferragina and Scaiella, 2010) and FOFE NER (Xu et al., 2017). By combining the results of both

Components	Example 1	Example 2
Question [Answer]	`Which 18th century author wrote Clarissa (or The History of a Young Lady), said to be the longest novel in the English language? [Samuel Richardson]`	`What is the correct name of the character voiced by Angela Lansbury in Beauty and The Beast? [Mrs Potts]`
Subject (Freebase ID)	`Clarissa (m.05s1st)`	`Angela Lansbury (m.0161h5)`
Predicate	`book.written-work.author`	`film.actor.dubbing-performances`
Secondary Predicate	`-`	`film.dubbing-performance.character`
Object/Answer (ID)	`Samuel Richardson (m.0hb27)`	`Mrs Potts (m.02vw823)`

Table 1: Two typical examples to illustrate the data format of all matches in FreebaseQA.

systems, we create a list of possible subjects for each question. We use confidence thresholds of 0.2 and 0.9 for the respective systems to ensure that an adequate amount of entities are produced while avoiding the production of irrelevant results.

3.2 Freebase Matching

The matching starts by searching for all Freebase nodes with a name or alias matching each subject name. For each matched Freebase node (called a subject node), we search through all object nodes that are directly linked with the subject node. Then for each object node, we search through all of its names and aliases to see if one matches the answer to the question. Once a match is found, the subject node's Freebase ID, the predicate name, and the object node's Freebase ID are saved as a triple representing the question-answer pair. Note that one question-answer pair may generate several matched triples when multiple related predicates are found since each question may contain multiple entities and each subject node may link to an object node through different predicates.

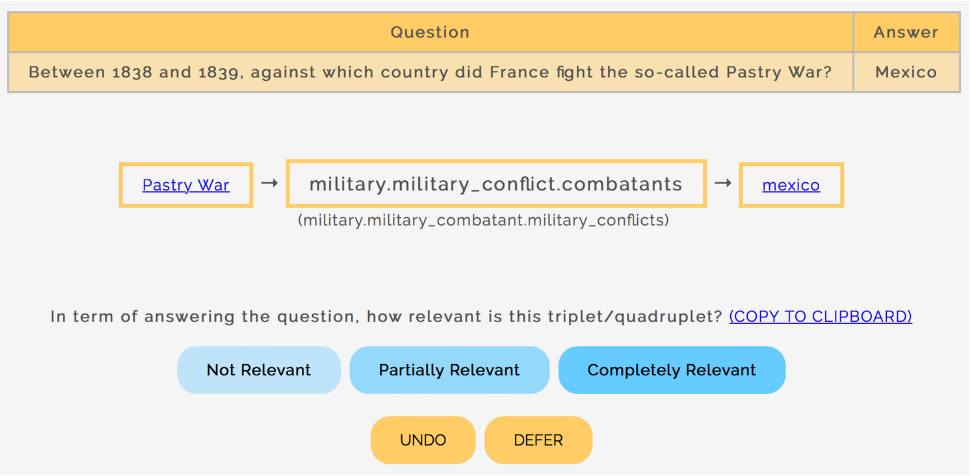
However, this procedure becomes inefficient since there is an enormous number of object nodes to process for some popular subject nodes, such as `United States (m.09c7w0)`, leading to a tremendous number of Freebase queries. Since we know the end point of the search, the answer to the question, this procedure is optimized by also starting from the answer and searching for all object nodes with a name or alias matching it. Then, the search concludes when the same object node is found from both starting points of the search. By using this two-way search method, we have accelerated the Freebase matching algorithm more than ten-fold.

3.3 Mediator Nodes

Freebase has been constructed with some special nodes called *mediator* nodes. A mediator is an intermediate node that connects a subject node with an object node. Since it itself is also considered a node, there are predicates from the subject to the mediator and from the mediator to the object. These mediator nodes are special as they do not have a name or alias associated with it, and only occurring in Freebase when there are multiple subjects and objects that are related through the mediator. When constructing the FreebaseQA data set, mediators are also accounted for. If the above search procedure reaches a mediator, a 2-hop matching strategy is conducted to search all nodes linked to this mediator. This captures a secondary predicate that bridges the subject to the answer through a mediator node. An example involving a mediator is described as Example 2 in Table 1.

3.4 Human Annotation

Since the matches found through the previously-explained algorithm are not guaranteed to be completely relevant to the question, human verification of the produced results is required to remove all possible false postitive matches. A group of 10 native English speakers are hired to label all of the collected matches. Each match is rated by the individuals as either "Completely Relevant", "Somewhat Relevant", or "Not Relevant". The choice of rating is dependent on the relevancy of the predicate to the question. If the predicate completely reflects the main idea asked by the question, the match is rated "Completely Relevant". If the predicate reflects part of the main idea of the question or is only somewhat related to it, the match is rated "Partially Relevant". Otherwise, the match is rated "Not Relevant". Compared with other



Question	Answer
Between 1838 and 1839, against which country did France fight the so-called Pastry War?	Mexico

Pastry War → military.military_conflict.combatants → mexico

(military.military_combatant.military_conflicts)

In term of answering the question, how relevant is this triplet/quadruplet? (COPY TO CLIPBOARD)

Figure 1: Human annotators use this website interface to label all automatically-generated matches, rating either Completely Relevant, Partially Relevant, or Not Relevant.

QA data collection tasks, human involvement in FreebaseQA is relatively light since each person only needs to make a one-out-of-three choice instead of composing a question or sentence from scratch. Therefore, using this method, we may significantly reduce the cost of QA data collection. As an illustration, the user interface for this data annotation procedure is shown in Figure 1.

In order to facilitate model training, the matches rated "Completely Relevant" are randomly chosen to populate the training, evaluation, and development sets of FreebaseQA. These sets are separated so that if there are multiple matches for a single question-answer pair, all of those matches will exist in only one of the sets. Moreover, the matches rated "Partially Relevant" are provided as a separate set, which may be useful for model training as well. The FreebaseQA data set is available for public use at `http://github.com/infinitecold/FreebaseQA`.

4 Results

We report the preliminary results of our statistical analysis on the collected FreebaseQA data set.

4.1 Collected Raw Question-Answer Pairs

The statistics of the originally collected question-answer pairs and the corresponding Freebase matches are summarized in Table 2. We see that with the exception of KnowQuiz, the number of matches in Freebase roughly equate the number of questions in each source. Among all the generated matches, 54,611 matches in total are kept as true positives by human annotators.

Source	Questions	Matches
TriviaQA	98,973	99,523
KnowQuiz	9,996	2,389
QuizBalls	15,370	17,856
QuizZone	7,686	7,289
Total	132,025	127,057

Table 2: A summary of the number of question-answer pairs from each source along with the number of matches generated from the above Freebase matching procedure.

4.2 FreebaseQA Statistics

The size of the FreebaseQA data set is compared to two similar QA data sets, WebQuestionsSP (WebQSP) (Yih et al., 2016) and SimpleQuestions (Bordes et al., 2015), in Table 3.

Data Set	`train`	`dev`	`eval`	Total
FreebaseQA	20,358	3,994	3,996	28,348
SimpleQuestions	75,910	10,845	21,687	108,442
WebQSP	3,098	-	1,639	4,737

Table 3: Total numbers of unique questions found in the subsets of each data set.

We see that FreebaseQA has a significantly larger size than WebQSP in number of unique questions, but it is about one quarter of SimpleQuestions in number of unique questions. Among these matches, FreebaseQA contains 28,348 unique questions in total, with 20,358, 3,994 and 3,996 in the train, dev and eval sets respectively.

However, another important factor to consider

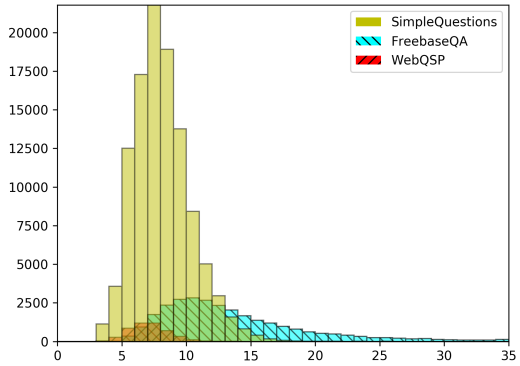


Figure 2: A histogram showing the spread of the length of the questions in each data set.

is the linguistic sophistication of the data. The sophistication of the linguistic structure of the questions in the FreebaseQA data set is compared to other similar data sets based on the average length, in number of words, of the questions. The histogram of question lengths of three data sets is shown in Figure 2. From the histogram, we see that the length of the questions in FreebaseQA extend much longer than the questions in SimpleQuestions or WebQSP (Yih et al., 2016). In fact, SimpleQuestions has an average length of 7.65 words per question and WebQSP has an average length of 6.62 words per question, while FreebaseQA has an average length of 13.35 words per question: approximately double the length of either data set.

4.3 Baseline Performance on FreebaseQA

Finally, we use FOFE-net (Zhang et al., 2015; Xu et al., 2017) to build a baseline KBQA system on FreebaseQA, which consists of subject detection, entity linking and relation detection in the pipeline. Our FOFE-net models are first compared with the popular hierarchical residual BiLSTM in (Yu et al., 2017) on two public data sets, such as SimpleQuestions and WebQSP. See (Wu et al., 2019) for more details on experimental settings and results. The comparison results are listed in Table 4.

As shown in Table 4, our baseline has achieved strong performance on the two public data sets but its final question answering accuracy has dropped significantly down to 37.0% on FreebaseQA. Obviously, FreebaseQA is a much more challenging KBQA task than both SimpleQuestions and WebQSP due to the fact that the questions in FreebaseQA are more complex in linguistic structure. Therefore, FreebaseQA may serve as an excellent testbed for more advanced KBQA techniques.

Data Set	BiLSTM (Yu et al., 2017)	FOFE-net (this work)
SimpleQuestions	77.0%	77.3%
WebQSP	63.0%	67.6%
FreebaseQA	-	**37.0%**

Table 4: Comparison of end-to-end QA accuracies on several KBQA data sets.

To facilitate the evaluation of the end-to-end question-answering pipeline on FreebaseQA, we have extracted a subset of Freebase, which contains all nodes and their corresponding predicates matching any entities in the FreebaseQA data set. This Freebase subset, also available at `http://github.com/infinitecold/FreebaseQA`, may be used to conduct end-to-end QA experiments to compare with our performance results in Table 4.

5 Conclusion

This paper presents a new data set, *FreebaseQA*, for open-domain factoid QA over structured knowledge bases. FreebaseQA has a size of over 54K matches, significantly larger than WebQSP and linguistically more sophisticated than SimpleQuestions. Our baseline QA results have also shown that FreebaseQA is a much more difficult KBQA task than either WebQSP or SimpleQuestions. Therefore, FreebaseQA may be an invaluable asset to the investigation of more advanced machine learning methods for factoid KBQA problems. Furthermore, the use of this data set is not only limited to factoid question answering, but several other applications can also be approached with this data set, including reading comprehension, natural language-based search, and the quantification of natural language understanding.

Acknowledgments

This work is partially supported by a research donation from iFLYTEK Co., Ltd., Hefei, China, and a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Weston Jason. 2015. Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1625–1628. Association for Computing Machinery.

Google. 2017. Freebase data dumps. `https://developers.google.com/freebase/data`.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611. Association for Computational Linguistics.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. arXiv preprint arXiv:1607.06275.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, pages 2518–2526.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading COmprehension dataset. arXiv preprint arXiv:1611.09268.

Michael Petrochuk and Luke Zettlemoyer. 2018. SimpleQuestions nearly solved: A new upperbound and baseline approach. arXiv preprint arXiv:1804.08798.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto Garca-Durn, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. arXiv preprint arXiv:1603.06807.

Dekun Wu, Nana Nosirova, Hui Jiang, and Mingbin Xu. 2019. A general FOFE-net framework for simple and effective question answering over knowledge bases. arXiv preprint arXiv:1903.12356.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1237–1247. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206. Association for Computational Linguistics.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 571–581. Association for Computational Linguistics.

ShiLiang Zhang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 495–500. Association for Computational Linguistics.

Simple Question Answering with Subgraph Ranking and Joint-Scoring

Wenbo Zhao[*]
Carnegie Mellon University
wzhao1@andrew.cmu.edu

Tagyoung Chung, Anuj Goyal, and **Angeliki Metallinou**
Amazon Alexa AI
{tagyoung, anujgoya, ametalli}@amazon.com

Abstract

Knowledge graph based simple question answering (KBSQA) is a major area of research within question answering. Although only dealing with *simple* questions, i.e., questions that can be answered through a single knowledge base (KB) fact, this task is neither simple nor close to being solved. Targeting on the two main steps, subgraph selection and fact selection, the research community has developed sophisticated approaches. However, the importance of subgraph ranking and leveraging the subject–relation dependency of a KB fact have not been sufficiently explored. Motivated by this, we present a unified framework to describe and analyze existing approaches. Using this framework as a starting point, we focus on two aspects: improving subgraph selection through a novel ranking method and leveraging the subject–relation dependency by proposing a joint scoring CNN model with a novel loss function that enforces the well-order of scores. Our methods achieve a new state of the art (85.44% in accuracy) on the SimpleQuestions dataset.

1 Introduction

Knowledge graph based simple question answering (KBSQA) is an important area of research within question answering, which is one of the core areas of interest in natural language processing (Yao and Van Durme, 2014; Yih et al., 2015; Dong et al., 2015; Khashabi et al., 2016; Zhang et al., 2018; Hu et al., 2018). It can be used for many applications such as virtual home assistants, customer service, and chat-bots. A *knowledge graph* is a multi-entity and multi-relation directed graph containing the information needed to answer the questions. The graph can be represented as collection of triples {(subject, relation, object)}. Each triple is called a *fact*, where a directed relational arrow points from subject node to object node. A *simple* question means that the question can be answered by extracting a single fact from the knowledge graph, i.e., the question has a single subject and a single relation, hence a single answer. For example, the question "Which Harry Potter series did Rufus Scrimgeour appear in?" can be answered by a single fact (Rufus Scrimgeour, book.book-characters.appears-in-book, Harry Potter and the Deathly Hallows). Given the simplicity of the questions, one would think this task is trivial. Yet it is far from being easy or close to being solved. The complexity lies in two aspects. One is the massive size of the knowledge graph, usually in the order of billions of facts. The other is the variability of the questions in natural language. Based on this anatomy of the problem, the solutions also consist of two steps: (1) selecting a relatively small subgraph from the knowledge graph given a question and (2) selecting the correct fact from the subgraph.

Different approaches have been studied to tackle the KBSQA problems. The common solution for the first step, subgraph selection (which is also known as entity linking), is to label the question with subject part (*mention*) and non-subject part (*pattern*) and then use the mention to retrieve related facts from the knowledge graph, constituting the subgraph. Sequence labeling models, such as a BiLSTM-CRF tagger (Huang et al., 2015), are commonly employed to label the mention and the pattern. To retrieve the subgraph, it is common to search all possible n-grams of the mention against the knowledge graph and collect the facts with matched subjects as the subgraph. The candidate facts in the subgraph may contain incorrect subjects and relations. In our running example, we first identify the

---

[*] Work conducted during an internship at Alexa AI, CA.

Proceedings of NAACL-HLT 2019, pages 324–334
Minneapolis, Minnesota, June 2 - June 7, 2019.

mention in the question, i.e.,"Rufus Scrimgeour", and then retrieve the subgraph which could contain the following facts: {(Rufus Scrimgeour, book.book-characters.appears-in-book, Harry Potter and the Deathly Hallows), (Rufus Wainwright, music.singer.singer-of, I Don't Know What That Is)}.

For the second step, fact selection, a common approach is to construct models to match the mention with candidate subjects and match the pattern with candidate relations in the subgraph from the first step. For example, the correct fact is identified by matching the mention "Rufus Scrimgeour" with candidate subjects {Rufus Scrimgeour, Rufus Wainwright} and matching the pattern "Which Harry Potter series did m appear in" with candidate relations {book.book-characters.appears-in-book, music.singer.singer-of}. Different neural network models can be employed (Bordes et al., 2015; Dai et al., 2016; Yin et al., 2016; Yu et al., 2017; Petrochuk and Zettlemoyer, 2018).

Effective as these existing approaches are, there are three major drawbacks. (1) First, in subgraph selection, there is no effective way to deal with inexact matches and the facts in subgraph are not ranked by relevance to the mention; however, we will later show that effective ranking can substantially improve the subgraph recall. (2) Second, the existing approaches do not leverage the dependency between mention–subjects and pattern–relations; however, mismatches of mention–subject can lead to incorrect relations and hence incorrect answers. We will later show that leveraging such dependency contributes to the overall accuracy. (3) Third, the existing approaches minimize the ranking loss (Yin et al., 2016; Lukovnikov et al., 2017; Qu et al., 2018); however, we will later show that the ranking loss is suboptimal.

Addressing these points, the contributions of this paper are three-fold: (1) We propose a subgraph ranking method with combined literal and semantic score to improve the recall of the subgraph selection. It can deal with inexact match, and achieves better performance compared to the previous state of the art. (2) We propose a low-complexity joint-scoring CNN model and a well-order loss to improve fact selection. It couples the subject matching and the relation matching by learning order-preserving scores and dynamically adjusting the weights of scores. (3) We achieve better performance (85.44% in accuracy) than the previous state of the art on the SimpleQuestions dataset, surpassing the best baseline by a large margin[1].

2 Related Work

The methods for subgraph selection fall in two schools: parsing methods (Berant et al., 2013; Yih et al., 2015; Zheng et al., 2018) and sequence tagging methods (Yin et al., 2016). The latter proves to be simpler yet effective, with the most effective model being BiLSTM-CRF (Yin et al., 2016; Dai et al., 2016; Petrochuk and Zettlemoyer, 2018).

The two categories of methods for fact selection are match-scoring models and classification models. The match-scoring models employ neural networks to score the similarity between the question and the candidate facts in the subgraph and then find the best match. For instance, Bordes et al. (2015) use a memory network to encode the questions and the facts to the same representation space and score their similarities. Yin et al. (2016) use two independent models, a character-level CNN and a word-level CNN with attentive max-pooling. Dai et al. (2016) formulate a two-step conditional probability estimation problem and use BiGRU networks. Yu et al. (2017) use two separate hierarchical residual BiLSTMs to represent questions and relations at different abstractions and granularities. Qu et al. (2018) propose an attentive recurrent neural network with similarity matrix based convolutional neural network (AR-SMCNN) to capture the semantic-level and literal-level similarities. In the classification models, Ture and Jojic (2017) employ a two-layer BiGRU model. Petrochuk and Zettlemoyer (2018) employ a BiLSTM to classify the relations and achieve the state-of-the-art performance. In addition, Mohammed et al. (2018) evaluate various strong baselines with simple neural networks (LSTMs and GRUs) or non-neural network models (CRF). Lukovnikov et al. (2017) propose an end-to-end word/character-level encoding network to rank subject–relation pairs and retrieve relevant facts.

However, the multitude of methods yield progressively smaller gains with increasing model complexity (Mohammed et al., 2018; Gupta et al.,

---

[1] Ture and Jojic (2017) reported better performance than us but neither Petrochuk and Zettlemoyer (2018) nor Mohammed et al. (2018) could replicate their result.

2018). Most approaches focus on fact matching and relation classification while assigning less emphasis to subgraph selection. They also do not sufficiently leverage the important signature of the knowledge graph—the subject–relation dependency, namely, incorrect subject matching can lead to incorrect relations. Our approach is similar to (Yin et al., 2016), but we take a different path by focusing on accurate subgraph selection and utilizing the subject–relation dependency.

3 Question Answering with Subgraph Ranking and Joint-Scoring

3.1 Unified Framework

We provide a unified description of the KBSQA framework. First, we define

Definition 1. Answerable Question *A question is answerable if and only if one of its facts is in the knowledge graph.*

Let $\mathcal{Q} := \{q \mid q \text{ is anwerable}\}$ be the set of answerable questions, and $\mathcal{G} := \{(s, r, o) \mid s \in \mathcal{S}, r \in \mathcal{R}, o \in \mathcal{O}\}$ be the knowledge graph, where $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{O}$ are the set of subjects, relations and objects, respectively. The triple (s, r, o) is a *fact*. By the definition of answerable questions, the key to solving the KBSQA problem is to *find the fact in knowledge graph corresponding to the question*, i.e., we want a map $\Phi : \mathcal{Q} \to \mathcal{G}$. Ideally, we would like this map to be injective such that for each question, the corresponding fact can be uniquely determined (more precisely, the injection maps from the *equivalent class* of $\mathcal{Q}$ to $\mathcal{G}$ since similar questions may have the same answer, but we neglect such difference here for simplicity). However, in general, it is hard to find such map directly because of (1) the massive knowledge graph and (2) natural language variations in questions. Therefore, end-to-end approaches such as parsing to structured query and encoding-decoding models are difficult to achieve (Yih et al., 2015; Sukhbaatar et al., 2015; Kumar et al., 2016; He and Golub, 2016; Hao et al., 2017). Instead, related works and this work mitigate the difficulties by breaking down the problem into the aforementioned two steps, as illustrated below:

(1) Subgraph Selection:

$$\text{question} \longrightarrow \{\text{mention, pattern}\}, \quad \text{mention} \longrightarrow \text{subgraph}$$

(2) Fact Selection:

$$\text{match} \begin{cases} \text{mention} \leftrightarrow \text{subject} \\ \text{pattern} \leftrightarrow \text{relation} \end{cases} \forall (\text{subject, relation}) \in \text{subgraph}$$

$$\Rightarrow (\text{subject*, relation*}) \longrightarrow \text{object* (answer*)}$$

In the first step, the size of the knowledge graph is significantly reduced. In the second step, the variations of questions are confined to mention–subject variation and pattern–relation variation.

Formally, we denote the questions as the union of mentions and patterns $\mathcal{Q} = \mathcal{M} \bigcup \mathcal{P}$ and the knowledge graph as the subset of the Cartesian product of subjects, relations and objects $\mathcal{G} \subseteq \mathcal{S} \times \mathcal{R} \times \mathcal{O}$. In the first step, given a question $q \in \mathcal{Q}$, we find the mention via a sequence tagger $\tau : \mathcal{Q} \to \mathcal{M}$, $q \mapsto m_q$. The tagged mention consists of a sequence of words $m_q = \{w_1, \ldots, w_n\}$ and the pattern is the question excluding the mention $p_q = q \backslash m_q$. We denote the set of n-grams of m_q as $\mathcal{W}_n(m_q)$ and use $\mathcal{W}_n(m_q)$ to retrieve the subgraph as $\mathcal{S}_q \times \mathcal{R}_q \times \mathcal{O}_q \supseteq \mathcal{G}_q := \{(s, r, o) \in \mathcal{G} \mid \mathcal{W}_n(s) \bigcap \mathcal{W}_n(m_q) \neq \varnothing, n = 1, \ldots, |m_q|\}$.

Next, to select the correct fact (the answer) in the subgraph, we match the mention m_q with candidate subjects in $\mathcal{S}_q$, and match the pattern p_q with candidate relations in $\mathcal{R}_q$. Specifically, we want to maximize the log-likelihood

$$\begin{cases} \max_{s \in \mathcal{S}_q} \log \mathbb{P}(s \mid m_q) \\ \max_{r \in \mathcal{R}_q} \log \mathbb{P}(r \mid p_q). \end{cases} \tag{1}$$

The probabilities in (1) are modeled by

$$\mathbb{P}(s \mid m_q) = \frac{e^{h(f(m_q), f(s))}}{\sum_{s' \in \mathcal{S}_q} e^{h(f(m_q), f(s'))}} \tag{2}$$

$$\mathbb{P}(r \mid p_q) = \frac{e^{h(g(p_q), g(r))}}{\sum_{r' \in \mathcal{R}_q} e^{h(g(p_q), g(r'))}}, \tag{3}$$

where $f : \mathcal{M} \bigcup \mathcal{S} \to \mathbb{R}^d$ maps the mention and the subject onto a d-dimensional differentiable manifold embedded in the Hilbert space and similarly, $g : \mathcal{P} \bigcup \mathcal{R} \to \mathbb{R}^d$. Both f and g are in the form of neural networks. The map $h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a metric that measures the similarity of the vector representations (e.g., the cosine similarity). Practically, directly optimizing (1) is difficult because the subgraph $\mathcal{G}_q$ is large and computing the partition functions in (2) and (3) can be intractable. Alternatively, a surrogate objective, the ranking loss (or hinge loss with negative samples) (Collobert and Weston, 2008; Dai et al., 2016) is minimized

$$\mathcal{L}_{\text{rank}} = \sum_{q \in \mathcal{Q}} \left(\sum_{s \in \mathcal{S}_q} \left[h_f(m_q, s^-) - h_f(m_q, s^+) + \lambda \right]_+ + \sum_{r \in \mathcal{R}_q} \left[h_g(p_q, r^-) - h_g(p_q, r^+) + \lambda \right]_+ \right), \quad (4)$$

where $h_f(\cdot, \cdot) = h(f(\cdot), f(\cdot))$, $h_g(\cdot, \cdot) = h(g(\cdot), g(\cdot))$; the sign $+$ and $-$ indicate correct candidate and incorrect candidate, $[\cdot]_+ = \max(\cdot, 0)$, and $\lambda > 0$ is a margin term. Other variants of the ranking loss are also studied (Cao et al., 2006; Zhao et al., 2015; Vu et al., 2016).

3.2 Subgraph Ranking

To retrieve the subgraph of candidate facts using n-gram matching (Bordes et al., 2015), one first constructs the map from n-grams $\mathcal{W}_n(s)$ to subject s for all subjects in the knowledge graph, yielding $\{\mathcal{W}_n(s) \to s \mid s \in \mathcal{S}, n = 1, \dots, |s|\}$. Next, one uses the n-grams of mention $\mathcal{W}_n(m)$ to match the n-grams of subjects $\mathcal{W}_n(s)$ and fetches those matched facts to compose the subgraph $\{(s, r, o) \in \mathcal{G} \mid \mathcal{W}_n(s) \bigcap \mathcal{W}_n(m) \neq \varnothing, n = 1, \dots |m|\}$. In our running example, for the mention "Rufus Scrimgeour", we collect the subgraph of facts with the bigrams and unigrams of subjects matching the bigram {"Rufus Scrimgeour"} and unigrams {"Rufus", "Scrimgeour"}.

One problem with this approach is that the retrieved subgraph can be fairly large. Therefore, it is desirable to rank the subgraph by relevance to the mention and only preserve the most relevant facts. To this end, different ranking methods are used, such as surface-level matching score with added heuristics (Yin et al., 2016), relation detection network (Yu et al., 2017; Hao et al., 2018), term frequency-inverse document frequency (TF-IDF) score (Ture and Jojic, 2017; Mohammed et al., 2018). However, these ranking methods only consider matching surface forms and cannot handle inexact matches, synonyms, or polysemy ("New York" , "the New York City", "Big Apple").

This motivates us to rank the subgraph not only by literal relevance but also semantic relevance. Hence, we propose a ranking score with literal closeness and semantic closeness. Specifically, the literal closeness is measured by the length of the longest common subsequence $|\sigma|(s, m)$ between a subject s and a mention m. The semantic closeness is measured by the co-occurrence probability of the subject s and the mention m

$$\begin{aligned}
\mathbb{P}(s, m) &= \mathbb{P}(s|m)\mathbb{P}(m) \\
&= \mathbb{P}(w_1, \dots w_n | \widetilde{w}_1, \dots \widetilde{w}_m) \mathbb{P}(\widetilde{w}_1, \dots \widetilde{w}_m) && (5) \\
&= \prod_{i=1}^{n} \mathbb{P}(w_i | \widetilde{w}_1, \dots \widetilde{w}_m) \mathbb{P}(\widetilde{w}_1, \dots \widetilde{w}_m) && (6) \\
&= \prod_{i=1}^{n} \left(\prod_{k=1}^{m} \mathbb{P}(w_i | \widetilde{w}_k) \right) \mathbb{P}(\widetilde{w}_1, \dots \widetilde{w}_m) && (7) \\
&= \prod_{i=1}^{n} \left(\prod_{k=1}^{m} \mathbb{P}(w_i | \widetilde{w}_k) \right) \prod_{j=1}^{m-1} \mathbb{P}(\widetilde{w}_{j+1} | \widetilde{w}_j) \mathbb{P}(\widetilde{w}_1), && (8)
\end{aligned}$$

where from (5) to (6) we assume conditional independence of the words in subject and the words in mention; from (6) to (7) and from (7) to (8) we factorize the factors using the chain rule with conditional independence assumption. The marginal term $\mathbb{P}(\widetilde{w}_1)$ is calculated by the word occurrence frequency. Each conditional term is approximated by $\mathbb{P}(w_i | w_j) \approx \exp\{\hat{w}_i^T \hat{w}_j\}$ where $\hat{w}_i$s are pretrained GloVe vectors (Pennington et al., 2014). These vectors are obtained by taking into account the word co-occurrence probability of surrounding context. Hence, the GloVe vector space encodes the semantic closeness. In practice we use the log-likelihood as the semantic score to convert multiplication in (8) to summation and normalize the GloVe embeddings into a unit ball. Then, the score for ranking the subgraph is the weighted sum of the literal score and the semantic score

$$\text{score}(s, m) = \tau |\sigma|(s, m) + (1 - \tau) \log \mathbb{P}(s, m), \quad (9)$$

where τ is a hyper-parameter whose value need to be tuned on the validation set. Consequently, for each question q, we can get the top-n ranked subgraph $\mathcal{G}_{q\downarrow}^n$ as well as the corresponding top-n ranked candidate subjects $\mathcal{S}_{q\downarrow}^n$ and relations $\mathcal{R}_{q\downarrow}^n$.

3.3 Joint-Scoring Model with Well-Order Loss

Once we have the ranked subgraph, next we need to identify the correct fact in the subgraph. One school of conventional methods (Bordes et al., 2014, 2015; Yin et al., 2016; Dai et al., 2016) is minimizing the surrogate ranking loss (4) where neural networks are used to transform the (subject, mention) and (relation, pattern) pairs into a Hilbert space and score them with inner product.

One problem with this approach is that it matches mention–subject and pattern–relation

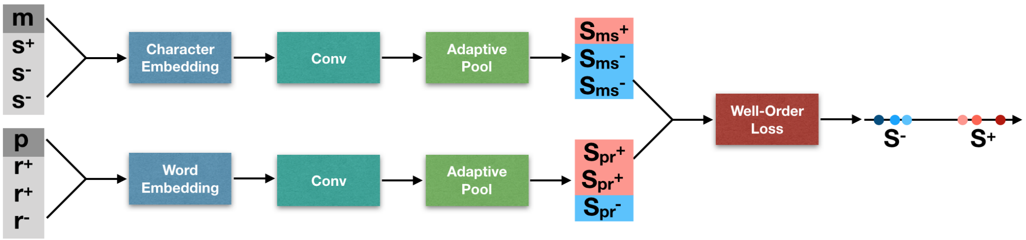


Figure 1: Model Diagram (Section 3.3) The model takes input pairs (**m**ention, **s**ubject) and (**p**attern, **r**elation) to produce the similarity scores. The loss dynamically adjusts the weights and enforces the order of positive and negative scores.

separately, neglecting the difference of their contributions to fact matching. Given that the number of subjects (order of millions) are much larger than the number of relations (order of thousands), incorrect subject matching can lead to larger error than incorrect relation matching. Therefore, matching the subjects correctly should be given more importance than matching the relations. Further, the ranking loss is suboptimal, as it does not preserve the relative order of the matching scores. We empirically find that the ranking loss tends to bring the matching scores to the neighborhood of zero (during the training the scores shrink to very small numbers), which is not functioning as intended.

To address these points, we propose a joint-scoring model with well-order loss (Figure 1). Together they learn to map from joint-input pairs to order-preserving scores supervised by a well-order loss, hence the name. The joint-scoring model takes joint-input pairs, (subject, mention) or (relation, pattern), to produce the similarity scores directly. The well-order loss then enforces the well-order in scores.

A well-order, first of all, is a total order—a binary relation on a set which is antisymmetric, transitive, and connex. In our case it is just the "$\leq$" relation. In addition, the well-order is a total order with the property that every non-empty set has a least element. The well-order restricts that the scores of correct matches are always larger or equal to the scores of incorrect matches, i.e., $\forall i : \forall j : S_i^+ \geq S_j^-$ where S_i^+ and S_i^- indicate the score of correct match and the score of incorrect match.

We derive the well-order loss in the following way. Let $S = \{S_1, \ldots, S_n\} = S^+ \bigcup S^-$ be the set of scores where S^+ and S^- are the set of scores with correct and incorrect matches. Let $I = I^+ \bigcup I^-$ be the index set of S, $|I^+| = n_1$, $|I^-| = n_2$, $n = n_1 + n_2$. Following the well-order relation

$$\begin{aligned}
&\inf S^+ \geq \sup S^- \\
\Leftrightarrow\ &\forall i^+ \in I^+ : \forall i^- \in I^- : S_{i^+}^+ - S_{i^-}^- \geq 0 \\
\Leftrightarrow\ &\sum_{i^+ \in I^+} \sum_{i^- \in I^-} (S_{i^+}^+ - S_{i^-}^-) \geq 0 \qquad (10) \\
\Leftrightarrow\ &n_2 \sum_{i^+ \in I^+} S_{i^+}^+ - n_1 \sum_{i^- \in I^-} S_{i^-}^- \geq 0, \qquad (11)
\end{aligned}$$

where from (10) to (11) we expand the sums and reorder the terms. Consequently, we obtain the well-order loss

$$\begin{aligned}
&\mathcal{L}_{\text{well-order}}(S_{ms}, S_{pr}) = \\
&\left[|I^+| \sum_{i^-} S_{ms}^{i^-} - |I^-| \sum_{i^+} S_{ms}^{i^+} + |I^+||I^-|\lambda \right]_+ + \\
&\left[|J^+| \sum_{j^-} S_{pr}^{j^-} - |J^-| \sum_{j^+} S_{pr}^{j^+} + |J^+||J^-|\lambda \right]_+, \qquad (12)
\end{aligned}$$

where S_{ms}, S_{pr} are the scores for (mention, subject), (pattern, relation) pairs for a question, I, J are the index sets for candidate subjects, relations in the ranked subgraph, $+$, $-$ indicate the correct candidate and incorrect candidate, $[\cdot]_+ = \max(\cdot, 0)$, and $\lambda > 0$ is a margin term. Then, the objective (1) becomes

$$\begin{aligned}
\min_{q \in \mathcal{Q}, (s,r) \in \mathcal{S}_{q\downarrow}^n \times \mathcal{R}_{q\downarrow}^n} &\Bigg[|I^+| \sum_{i^-} h_f(m_q, s^{i^-}) - \\
&|I^-| \sum_{i^+} h_f(m_q, s^{i^+}) + |I^+||I^-|\lambda \Bigg]_+ \\
+ &\Bigg[|J^+| \sum_{j^-} h_g(p_q, r^{j^-}) - \\
&|J^-| \sum_{j^+} h_g(p_q, r^{j^+}) + |J^+||J^-|\lambda \Bigg]_+ . \qquad (13)
\end{aligned}$$

This new objective with well-order loss differs from the ranking loss (4) in two ways, and plays a vital role in the optimization. First, instead of considering the match of mention–subjects and pattern–relations separately, (13) *jointly considers both input pairs and their dependency*. Specifically, (13) incorporates such dependency as the weight factors $|I|$ (for subjects) and $|J|$ (for relations). These factors are the controlling factors and are automatically and dynamically adjusted as they are the sizes of candidate subjects and relations. Further, the match of subjects, weighted by (I^+, I^-), will control the match of relations, weighted by (J^+, J^-). To see this, for a question and a fixed number of candidate facts in subgraph, $|I| = |J|$, the incorrect number of subjects $|I^-|$ is usually larger than the incorrect number of relations $|J^-|$, which causes larger loss for mismatching subjects. As a result, the model is forced to match subjects more correctly, and in turn, prune the relations with incorrect subjects and reduce the size of J^-, leading to smaller loss. Second, the well-order loss enforces the well-order relation of scores while the ranking loss does not have such constraint.

4 Experiments

Here, we evaluate our proposed approach for the KBSQA problem on the SimpleQuestions benchmark dataset and compare with baseline approaches.

4.1 Data

The SimpleQuestions (Bordes et al., 2015) dataset is released by the Facebook AI Research. It is the standard dataset on which almost all previous state-of-the-art literature reported their numbers (Gupta et al., 2018; Hao et al., 2018). It also represents the largest publicly available dataset for KBSQA with its size several orders of magnitude larger than other available datasets. It has $108,442$ simple questions with the corresponding facts from subsets of the Freebase (FB2M and FB5M). There are $1,837$ unique relations. We use the default train, validation and test partitions (Bordes et al., 2015) with $75,910$, $10,845$ and $21,687$ questions, respectively. We use FB2M with $2,150,604$ entities, $6,701$ relations and $14,180,937$ facts, respectively.

Vocab. size	151,718
Embedding dim	300
LSTM hidden dim	256
# of LSTM layers	2
LSTM dropout	0.5
# of CRF states	4 (incl. start & end)

Table 1: Sequence Tagger Configurations

Config.	CharCNN	WordCNN
Alphabet / Vocab. size	69	151,718
Embedding dim	60	300
CNN layer 1	(300, 3, 1, 1)	(1500, 3, 1, 1)
Activation	ReLU	ReLU
CNN layer 2	(60, 3, 1, 1)	(300, 3, 1, 1)
AdaptiveMaxPool dim	1	1

Table 2: Matching Model Configurations

4.2 Models

For sequence tagging, we use the same BiLSTM-CRF model as the baseline (Dai et al., 2016) to label each word in the question as either subject or non-subject. The configurations of the model (Table 1) basically follow the baseline (Dai et al., 2016).

For subgraph selection, we use only unigrams of the tagged mention to retrieve the candidate facts (see Section 3.2) and rank them by the proposed relevance score (9) with the tuned weight $\tau = 0.9$ (hence more emphasizing on literal matching). We select the facts with top-n scores as the subgraphs and compare the corresponding recalls with the baseline method (Yin et al., 2016).

For fact selection, we employ a character-based CNN (CharCNN) model to score (mention, subject) pairs and a word-based CNN (WordCNN) model to score (pattern, relation) pairs (with model configurations shown in Table 2), which is similar to one of the state-of-the-art baselines AMPCNN (Yin et al., 2016). In fact, we first replicated the AMPCNN model and achieved comparable results, and then modified the AMPCNN model to take joint inputs and output scores directly (see Section 3.3 and Figure 1). Our CNN models have only two convolutional layers (versus six convolutional layers in the baseline) and have no attention mechanism, bearing much lower complexity than the baseline. The CharCNN and WordCNN differ only in the embedding layer, the former using character embeddings and the latter using word embeddings.

Config.	Sequence Tagging	Matching
Optimizer	Adam	Adam
Learning rate	0.001	0.01
Batch size	64	32
# of epochs	50	20

Table 3: Learning Configurations

Rank Method	Top-N	Recall
Literal: $\|\sigma\|$ + heuristics (Yin et al., 2016)	1	0.736
	5	0.850
	10	0.874
	20	0.888
	50	0.904
	100	0.916
Semantic: $\log \mathbb{P}$	1	0.482
	10	0.753
	20	0.854
	50	0.921
	100	0.848
Joint: $0.9\|\sigma\| + 0.1 \log \mathbb{P}$	1	0.855
	5	0.904
	10	0.920
	20	0.927
	50	0.945
	100	0.928

Table 4: Subgraph Selection Results

The optimizer used for training the models is Adam (Kingma and Ba, 2014). The learning configurations are shown in Table 3.

For the hyper-parameters shown in Table 1, 2 and 3, we basically follow the settings in baseline literature (Yin et al., 2016; Dai et al., 2016) to promote a fair comparison. Other hyper-parameters, such as the τ in the relevance score (9), are tuned on the validation set.

Our proposed approach and the baseline approaches are evaluated in terms of (1) the top-n subgraph selection recall (the percentage of questions that have the correct subjects in the top-n candidates) and (2) the fact selection accuracy (i.e., the overall question answering accuracy).

4.3 Results

Subgraph selection The subgraph selection results for our approach and one of the state-of-the-art baselines (Yin et al., 2016) are summarized in Table 4. Both the baseline and our approach use unigrams to retrieve candidates. The baseline ranks the candidates by the length of the longest common subsequence with heuristics while we rank the candidates by the joint relevance score defined in (9). We see that the literal score used in the baseline performs well and using the semantic score (the log-likelihood) (8) only does not outperform the baseline (except for the top-50 case). This is due to the nature of how the questions in the SimpleQuestions dataset are generated—the majority of the questions only contain mentions matching the subjects in the Freebase in the lexical level, making the literal score sufficiently effective. However, we see that combining the literal score and semantic score outperforms the baseline by a large margin. For top-1, 5, 10, 20, 50 recall our ranking approach surpasses the baseline by 11.9%, 5.4%, 4.6%, 3.9%, 4.1%, respectively. Our approach also surpasses other baselines (Lukovnikov et al., 2017; Yu et al., 2017; Qu et al., 2018; Gupta et al., 2018) under the same settings. We note that the recall is not monotonically increasing with the top-n. The reason is that, as opposed to conventional methods which rank the entire subgraph returned from unigram matching to select the top-n candidates, we choose only the first 200 candidates from the subgraph and then rank them with our proposed ranking score. This is more efficient, but at the price of potentially dropping the correct facts. One could trade efficiency for accuracy by ranking all the candidates in the subgraph.

Fact selection The fact selection results for our approach and baselines are shown in Table 5. The object accuracy is the same as the overall question answer accuracy. Recall that in Section 3.3 we explained that the weight components in the well-order loss (13) are adjusted dynamically in the training to impose a larger penalty for mention–subject mismatches and hence enforce correct matches. This can be observed by looking at the different loss components and weights as well the subject and relation matching accuracies during the training. As weights for mention–subject matches increase, the losses for mention–subject matches also increase, while both the errors for mention–subject matches and pattern–relation matches are high. To reduce the errors, the model is forced to match mention–subject more correctly. As a result, the corresponding weights and losses decrease, and both mention–subject and pattern–relation match accuracies increase.

Effectiveness of well-order loss and joint-

	Approach	Obj. (= **Overall Acc.**)	Sub.	Rel.
1	AMPCNN (Yin et al., 2016)	76.4		
2	BiLSTM (Petrochuk and Zettlemoyer, 2018)	78.1		
3	AMPCNN + wo-loss	77.69		
4	JS + wo-loss	81.10	87.44	69.22
5	JS + wo-loss + sub50	**85.44**	91.47	76.98
6	JS + wo-loss + sub1	79.34	87.97	84.12

Table 5: Fact Selection Accuracy (%). The object accuracy is the end-to-end question answer accuracy, while subject and relation accuracies refer to separately computed subject accuracy and relation accuracy.

scoring model The first and second row of Table 5 are taken from the baseline AMPCNN (Yin et al., 2016) and BiLSTM (Petrochuk and Zettlemoyer, 2018) (the state of the art prior to our work[2]). The third row shows the accuracy of the baseline with our proposed well-order loss and we see a 1.3% improvement, demonstrating the effectiveness of the well-order loss. Further, the fourth row shows the accuracy of our joint-scoring (JS) model with well-order loss and we see a 3% improvement over the best baseline[3], demonstrating the effectiveness of the joint-scoring model.

Effectiveness of subgraph ranking The fifth row of Table 5 shows the accuracy of our joint-scoring model with well-order loss and top-50 ranked subgraph and we see a further 4.3% improvement over our model without subgraph ranking (the fourth row), and a 7.3% improvement over the best baseline. In addition, the subject accuracy increases by 4.0%, which is due to the subgraph ranking. Interestingly, the relation accuracy increases by 7.8%, which supports our claim that improving subject matching can improve relation matching. This demonstrates the effectiveness of our subgraph ranking and joint-scoring approach. The sixth row shows the accuracy of our joint-scoring model with well-order loss and only the top-1 subject. In this case, the subject accuracy is limited by the top-1 recall which is 85.5%. Despite that, our approach outperforms the best baseline by 1.2%. Further, the relation accuracy increases by 7.1% over the fifth row, because restricting the subject substantially confines the choice of relations. This shows that a sufficiently high top-1 subgraph recall reduces the need for subject matching.

Incorrect Sub. only	8.67
Incorrect Rel. only	16.26
Incorrect Sub. & Rel.	34.50
Other	40.57

Table 6: Error Decomposition (%). Percentages for total of 3157 errors.

4.4 Error Analysis

In order to analyze what constitutes the errors of our approach, we select the questions in the test set for which our best model has predicted wrong answers, and analyze the source of errors (see Table 6). We observe that the errors can be categorized as follows: (1) Incorrect subject prediction; however, some subjects are actually correct, e.g., the prediction "New York" v.s. "New York City." (2) Incorrect relation prediction; however, some relations are actually correct, e.g., the prediction "fictional-universe.fictional-character.character-created-by" v.s. "book.written-work.author" in the question "Who was the writer of Dark Sun?" and "music.album.genre" v.s. "music.artist.genre." (3) Incorrect prediction of both.

However, these three reasons only make up 59.43% of the errors. The other 40.57% errors are due to: (4) Ambiguous questions, which take up the majority of the errors, e.g., "Name a species of fish." or "What movie is a short film?" These questions are too general and can have multiple correct answers. Such issues in the SimpleQuestions dataset are analyzed by Petrochuk and Zettlemoyer (2018) (see further discussion on this at the end of this Section). (5) Non-simple questions,

---

[2] As noted, Ture and Jojic (2017) reported better performance than us but neither Petrochuk and Zettlemoyer (2018) nor Mohammed et al. (2018) could replicate their result.

[3] At the time of submission we also found that Hao et al. (2018) reported 80.2% accuracy.

e.g., "Which drama film was released in 1922?" This question requires two KB facts instead of one to answer correctly. (6) Wrong fact questions where the reference fact is non-relevant, e.g., "What is an active ingredient in Pacific?" is labeled with "Triclosan 0.15 soap". (7) Out of scope questions, which have entities or relations out the scope of FB2M. (8) Spelling inconsistencies, e.g., the predicted answer "Operation Shylock: A Confession" v.s. the reference answer "Operation Shylock", and the predicted answer "Tom and Jerry: Robin Hood and His Merry Mouse" v.s. the reference answer "Tom and Jerry". For these cases, even when the models predict the subjects and relations correctly, these questions are fundamentally *unanswerable*.

Although these issues are inherited from the dataset itself, given the large size of the dataset and the small proportion of the problematic questions, it is sufficient to validate the reliability and significance of our performance improvement and conclusions.

Answerable Questions Redefined Petrochuk and Zettlemoyer (2018) set an upper bound of 83.4% for the accuracy on the SimpleQuestions dataset. However, our models are able to do better than the upper bound. Are we doing something wrong? Petrochuk and Zettlemoyer (2018) claim that a question is unanswerable if there exist multiple valid subject–relation pairs in the knowledge graph, but we claim that a question is unanswerable if and only if there is no valid fact in the knowledge graph. There is a subtle difference between these two claims.

Based on different definitions of answerable questions, we further claim that incorrect subject or incorrect relation can still lead to a correct answer. For example, for the question "What is a song from Hier Komt De Storm?" with the fact (Hier Komt De Storm: 1980-1990 live, music.release.track-list, Stephanie), our predicted subject "Hier Komt De Storm: 1980-1990 live" does not match the reference subject "Hier Komt De Storm", but our model predicts the correct answer "Stephanie" because it can deal with inexact match of the subjects. In the second example, for the question "Arkham House is the publisher behind what novel?", our predicted relation "book.book-edition.publisher" does not match the reference relation "book.publishing-company.books-published", but our model predicts the correct answer "Watchers at the Strait Gate" because it can deal with paraphrases of relations. In the third example, for the question "Who was the king of Lydia and Croesus's father?", the correct subject "Croesus" ranks second in our subject predictions and the correct relation "people.person.parents" ranks fourth in our relation predictions, but our model predicts the correct answer "Alyattes of Lydia" because it reweighs the scores with respect to the subject–relation dependency and the combined score of subject and relation ranks first.

To summarize, the reason that we are able to redefine answerable questions and achieve significant performance gain is that we take advantage of the subgraph ranking and the subject–relation dependency.

5 Conclusions

In this work, we propose a subgraph ranking method and joint-scoring approach to improve the performance of KBSQA. The ranking method combines literal and semantic scores to deal with inexact match and achieves better subgraph selection results than the state of the art. The joint-scoring model with well-order loss couples the dependency of subject matching and relation matching and enforces the order of scores. Our proposed approach achieves a new state of the art on the SimpleQuestions dataset, surpassing the best baseline by a large margin.

In the future work, one could further improve the performance on simple question answering tasks by exploring relation ranking, different embedding strategies and network structures, dealing with open questions and out-of-scope questions. One could also consider extending our approach to complex questions, e.g., multi-hop questions where more than one supporting facts is required. Potential directions may include ranking the subgraph by assigning each edge (relation) a closeness score and evaluating the length of the shortest path between any two path-connected entity nodes.

Acknowledgments

The authors would like to thank anonymous reviewers. The authors would also like to thank Nikko Ström and other Alexa AI team members for their feedback.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193. ACM.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 800–810, Berlin, Germany. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1: Long Papers, pages 260–269.

Vishal Gupta, Manoj Chinnakotla, and Manish Shrivastava. 2018. Retrieve and re-rank: A simple and effective ir approach to simple question answering over knowledge graphs. In *Proceedings of the First Workshop on Fact Extraction and Verification (FEVER)*, pages 22–27.

Yanchao Hao, Hao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Pattern-revising enhanced simple question answering over knowledge bases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3272–3282.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 221–231.

Xiaodong He and David Golub. 2016. Character-level question answering with attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1598–1607.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1145–1152. AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2: Short Papers, pages 291–296.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Michael Petrochuk and Luke Zettlemoyer. 2018. SimpleQuestions nearly solved: A new upper-bound and baseline approach. *arXiv preprint arXiv:1804.08798*.

Yingqi Qu, Jie Liu, Liangyi Kang, Qinfeng Shi, and Dan Ye. 2018. Question answering over Freebase via attentive RNN with similarity matrix based CNN. *arXiv preprint arXiv:1804.03317*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

Ferhan Ture and Oliver Jojic. 2017. No need to pay attention: Simple recurrent neural networks work! In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2866–2872, Copenhagen, Denmark. Association for Computational Linguistics.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6060–6064. IEEE.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 956–966.

Wentau Yih, Minwei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1: Long Papers, pages 1321–1331.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756, Osaka, Japan. The COLING 2016 Organizing Committee.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 571–581.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1556–1564.

Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. 2018. Question answering over knowledge graphs: Question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11).

Learning to Attend On Essential Terms: An Enhanced Retriever-Reader Model for Open-domain Question Answering

Jianmo Ni[1][*], **Chenguang Zhu**[2], **Weizhu Chen**[3], **Julian McAuley**[1]
[1] University of California, San Diego
[2] Microsoft Speech and Dialogue Research Group
[3] Microsoft Dynamics 365 AI
{jin018,jmcauley}@ucsd.edu, {chezhu,wzchen}@microsoft.com

Abstract

Open-domain question answering remains a challenging task as it requires models that are capable of understanding questions and answers, collecting useful information, and reasoning over evidence. Previous work typically formulates this task as a reading comprehension or entailment problem given evidence retrieved from search engines. However, existing techniques struggle to retrieve indirectly related evidence when no directly related evidence is provided, especially for complex questions where it is hard to parse precisely what the question asks. In this paper we propose a retriever-reader model that learns to attend on essential terms during the question answering process. We build (1) an essential term selector which first identifies the most important words in a question, then reformulates the query and searches for related evidence; and (2) an enhanced reader that distinguishes between essential terms and distracting words to predict the answer. We evaluate our model on multiple open-domain multiple-choice QA datasets, notably performing at the level of the state-of-the-art on the AI2 Reasoning Challenge (ARC) dataset.

1 Introduction

Open-domain question answering (QA) has been extensively studied in recent years. Many existing works have followed the 'search-and-answer' strategy and achieved strong performance (Chen et al., 2017; Kwon et al., 2018; Wang et al., 2018b) spanning multiple QA datasets such as TriviaQA (Joshi et al., 2017), SQuAD (Rajpurkar et al., 2016), MS-Macro (Nguyen et al., 2016), ARC (Clark et al., 2018) among others.

However, open-domain QA tasks become inherently more difficult when (1) dealing with questions with little available evidence; (2) solving questions where the answer type is free-form text (e.g. multiple-choice) rather than a span among existing passages (i.e., 'answer span'); or when (3) the need arises to understand long and complex questions and reason over multiple passages, rather than simple text matching. As a result, it is essential to incorporate commonsense knowledge or to improve retrieval capability to better capture partially related evidence (Chen et al., 2017).

As shown in Table 1, the TriviaQA, SQuAD, and MS-Macro datasets all provide passages within which the correct answer is guaranteed to exist. However, this assumption ignores the difficulty of retrieving question-related evidence from a large volume of open-domain resources, especially when considering complex questions which require reasoning or commonsense knowledge. On the other hand, ARC does not provide passages known to contain the correct answer. Instead, the task of identifying relevant passages is left to the solver. However, questions in ARC have multiple answer choices that provide indirect information that can help solve the question. As such an effective model needs to account for relations among passages, questions, and answer choices. Real-world datasets such as Amazon-QA (a corpus of user queries from Amazon) (McAuley and Yang, 2016) also exhibit the same challenge, i.e., the need to surface related evidence from which to extract or summarize an answer.

Figure 1 shows an example of a question in the ARC dataset and demonstrates the difficulties in retrieval and reading comprehension. As shown for Choice 1 (C1), a simple concatenation of the

---

[*] Most of the work was done during internship at Microsoft, Redmond.

[1] For SQuAD and TriviaQA, since the questions are paired with span-type answers, it is convenient to obtain ranking supervision where retrieved passages are relevant via distant supervision; however free-form questions in ARC and Amazon-QA result in a lack of supervision which makes the problem more difficult. For MS-Macro, the dataset is designed to annotate relevant passages though it has free-form answers.

Proceedings of NAACL-HLT 2019, pages 335–344
Minneapolis, Minnesota, June 2 - June 7, 2019.

Dataset	# of questions	Open-domain	Multiple choice	Passage retrieval	No ranking supervision[1]
ARC (Clark et al., 2018)	≈ 7K	✓	✓	✓	✓
Amazon-QA (McAuley and Yang, 2016)	≈ 1.48M	✓			✓
SQuAD (Rajpurkar et al., 2016)	≈ 100K	✓			
TriviaQA (Joshi et al., 2017)	≈ 650K	✓			
MS-Macro (Nguyen et al., 2016)	≈ 1M	✓			

Table 1: Differences among popular QA datasets.

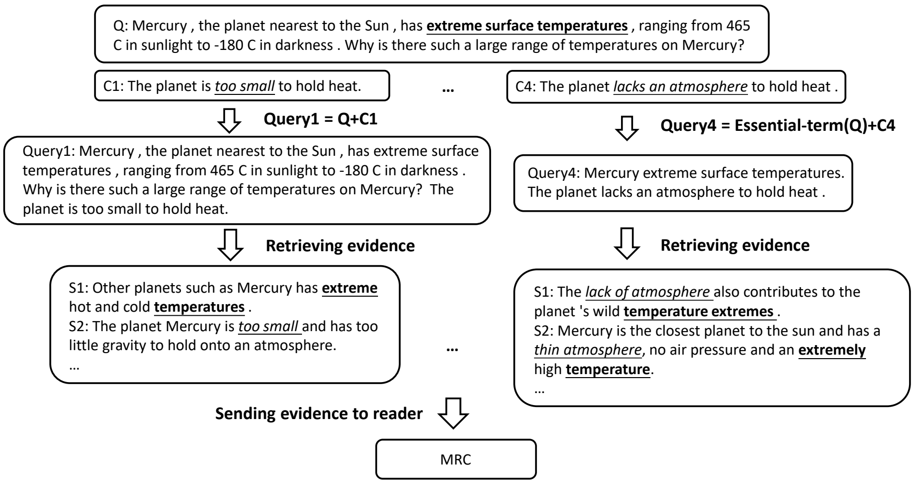


Figure 1: Example of the retrieve-and-read process to solve open-domain questions. Words related with the question are in **bold**; and words related with C1 and C4 are in *italics*.

question and the answer choice is not a reliable query and is of little help when trying to find supporting evidence to answer the question (e.g. we might retrieve sentences similar to the question or the answer choice, but would struggle to find evidence explaining *why* the answer choice is correct). On the other hand, a reformulated query consisting of essential terms in the question and Choice 4 can help retrieve evidence explaining why Choice 4 is a correct answer. To achieve this, the model needs to (1) ensure that the retrieved evidence supports the fact mentioned in both the question and the answer choices and (2) capture this information and predict the correct answer.

To address these difficulties, we propose an essential-term-aware Retriever-Reader (ET-RR) model that learns to attend on essential terms during retrieval and reading. Specifically, we develop a two-stage method with an essential term selector followed by an attention-enhanced reader.

Essential term selector. ET-Net is a recurrent neural network that seeks to understand the question and select essential terms, i.e., key words, from the question. We frame this problem as a classification task for each word in the question. These essential terms are then concatenated with each answer choice and fed into a retrieval engine to obtain related evidence.

Attention-Enhanced Reader. Our neural reader takes the triples (question, answer choice, retrieved passage) as input. The reader consists of a sequence of language understanding layers: an input layer, attention layer, sequence modeling layer, fusion layer, and an output layer. The attention and fusion layers help the model to obtain a refined representation of one text sequence based on the understanding of another, e.g. a passage representation based on an understanding of the question. We further add a choice-interaction module to handle the semantic relations and differences between answer choices. Experiments show that this can further improve the model's accuracy.

We evaluate our model on the ARC Challenge dataset, where our model achieves an accuracy of 36.61% on the test set, and outperformed all leaderboard solutions at the time of writing (Sep. 2018). To compare with other benchmark datasets, we adapt RACE (Lai et al., 2017) and MCScript (Ostermann et al., 2018) to the open domain setting by removing their supervision in the form of relevant passages. We also consider a large-scale real-world open-domain dataset, Amazon-QA, to evaluate our model's scalability and to compare against standard benchmarks designed for the open-domain setting. Experiments on these three datasets show that ET-RR outperforms baseline models by a large margin. We conduct multiple ablation studies to show the effectiveness of each component of our model. Finally, we perform in-depth error analysis to explore the model's limitations.

2 Related Work

There has recently been growing interest in building better retrievers for open-domain QA. Wang et al. (2018b) proposed a Reinforced Ranker-Reader model that ranks retrieved evidence and assigns different weights to evidence prior to processing by the reader. Min et al. (2018) demonstrated that for several popular MRC datasets (e.g. SQuAD, TriviaQA) most questions can be answered using only a few sentences rather than the entire document. Motivated by this observation, they built a sentence selector to gather this potential evidence for use by the reader model. Nishida et al. (2018) developed a multi-task learning (MTL) method for a retriever and reader in order to obtain a strong retriever that considers certain passages including the answer text as positive samples during training. The proposed MTL framework is still limited to scenarios where it is feasible to discover whether the passages contain the answer span. Although these works have achieved progress on open-domain QA by improving the ranking or selection of given evidence, few have focused on the scenario where the model needs to start by searching for the evidence itself.

Scientific Question Answering (SQA) is a representative open-domain task that requires capability in both retrieval and reading comprehension. In this paper, we study question answering on the AI2 Reasoning Challenge (ARC) scientific QA dataset (Clark et al., 2018). This dataset contains multiple-choice scientific questions from 3rd to 9th grade standardized tests and a large corpus of relevant information gathered from search engines. The dataset is partitioned into "Challenge" and "Easy" sets. The challenge set consists of questions that cannot be answered correctly by either of the solvers based on Pointwise Mutual Information (PMI) or Information Retrieval (IR). Existing models tend to achieve only slightly better and sometimes even worse performance than random guessing, which shows that existing models are not well suited to this kind of QA task.

Jansen et al. (2017) first developed a rule-based focus word extractor to identify essential terms in the question and answer candidates. The extracted terms are used to aggregate a list of potential answer justifications for each answer candidate. Experiments shown that focus words are beneficial for SQA on a subset of the ARC dataset. Khashabi et al. (2017) also worked on the problem of finding essential terms in a question for solving SQA problems. They published a dataset containing over 2,200 science questions annotated with essential terms and train multiple classifiers on it. Similarly, we leverage this dataset to build an essential term selector using a neural network-based algorithm. More recently, Boratko et al. (2018) developed a labeling interface to obtain high quality labels for the ARC dataset. One finding is that human annotators tend to retrieve better evidence after they reformulate the search queries which are originally constructed by a simple concatenation of question and answer choice. By feeding the evidence obtained by human-reformulated queries into a pre-trained MRC model (i.e., DrQA (Chen et al., 2017)) they achieved an accuracy increase of 42% on a subset of 47 questions. This shows the potential for a "human-like" retriever to boost performance on this task.

Query reformulation has been shown to be effective in information retrieval (Lavrenko and Croft, 2001). Nogueira and Cho (2017) modeled the query reformulation task as a binary term selection problem (i.e., whether to choose the term in the original query and the documents retrieved using the original query). The selected terms are then concatenated to form the new query. Instead of choosing relevant words, Buck et al. (2018) proposed a sequence-to-sequence model to generate new queries. Das et al. (2019) proposed Multi-step Retriever-Reader which explores an iterative retrieve-and-read strategy for open-domain question answering. It formulates the query reformulation problem in the embedding space where the vector representation of the question is changed to improve the performance. Since there is no supervision for training the query reformulator, all these methods using reinforcement learning to maximize the task-specific metrics (e.g. Recall for paragraph ranking, F1 and Exact Matching for span-based MRC). Different from these works, we train the query reformulator using an annotated dataset as supervision and then apply the output to a separate reader model. We leave the exploration of training our model end-to-end using reinforcement learning as future work.

3 Approach

In this section, we introduce the essential-term-aware retriever-reader model (ET-RR). As shown in Figure 2, we build a term selector to discover which terms are essential in a question. The selected terms are then used to formulate a more efficient query enabling the retriever to obtain related evidence. The retrieved evidence is then fed to the reader to predict the final answer.

For a question with q words $\mathbf{Q} = \{w_t^Q\}_{t=1}^q$ along with its N answer choices $\mathbf{C} = \{\mathbf{C}_n\}_{n=1}^N$ where $\mathbf{C}_n = \{w_t^C\}_{t=1}^c$, the essential-term selector chooses a subset of essential terms $\mathbf{E} \subset \mathbf{Q}$, which are then concatenated with each $\mathbf{C}_n$ to formulate a query. The query for each answer choice, $\mathbf{E} + \mathbf{C}_n$, is sent to the retriever (e.g. Elastic Search[2]), and the top K retrieved sentences based on the scores returned by the retriever are then concatenated into the evidence passage $\mathbf{P}_n = \{w_t^P\}_{t=1}^K$.

Next, given these text sequences $\mathbf{Q}$, $\mathbf{C}$, and $\mathbf{P} = \{\mathbf{P}_n\}_{n=1}^N$, the reader will determine a matching score for each triple $\{\mathbf{Q}, \mathbf{C}_n, \mathbf{P}_n\}$. The answer choice $\mathbf{C}_{n^*}$ with the highest score is selected.

We first introduce the reader model in Section 3.1 and then the essential term selector in Section 3.2.

3.1 Reader Model

3.1.1 Input Layer

To simplify notation, we ignore the subscript n denoting the answer choice until the final output layer. In the input layer, all text inputs—the question, answer choices, and passages, i.e., retrieved evidence—are converted into embedded representations. Similar to Wang (2018), we consider the following components for each word:

Word Embedding. Pre-trained GloVe word embedding with dimensionality $d_w = 300$.

Part-of-Speech Embedding and Named-Entity Embedding. The part-of-speech tags and named entities for each word are mapped to embeddings with dimension 16.

Relation Embedding. A relation between each word in $\mathbf{P}$ and any word in $\mathbf{Q}$ or $\mathbf{C}$ is mapped to an embedding with dimension 10. In the case that multiple relations exist, we select one uniformly at random. The relation is obtained by querying ConceptNet (Speer et al., 2017).

Feature Embeddings. Three handcrafted features are used to enhance the word representations: (1) Word Match; if a word or its lemma of $\mathbf{P}$ exists in $\mathbf{Q}$ or $\mathbf{C}$, then this feature is 1 (0 otherwise). (2) Word Frequency; a logarithmic term frequency is calculated for each word. (3) Essential Term; for the i-th word in $\mathbf{Q}$, this feature, denoted as w_{e_i}, is 1 if the word is an essential term (0 otherwise). Let $\mathbf{w}_e = [w_{e_1}, w_{e_2}, \ldots, w_{e_q}]$ denote the essential term vector.

For $\mathbf{Q}, \mathbf{C}, \mathbf{P}$, all of these components are concatenated to obtain the final word representations $\mathbf{W}_Q \in \mathbb{R}^{q \times d_Q}, \mathbf{W}_C \in \mathbb{R}^{c \times d_C}, \mathbf{W}_P \in \mathbb{R}^{p \times d_P}$, where d_Q, d_C, d_P are the final word dimensions of $\mathbf{Q}$, $\mathbf{C}$, and $\mathbf{P}$.

3.1.2 Attention Layer

As shown in Figure 2, after obtaining word-level embeddings, attention is added to enhance word representations. Given two word embedding sequences $\mathbf{W}_U, \mathbf{W}_V$, word-level attention is calculated as:

$$\begin{aligned} \mathbf{M}'_{UV} &= \mathbf{W}_U \mathbf{U} \cdot (\mathbf{W}_V \mathbf{V})^\top \\ \mathbf{M}_{UV} &= \mathrm{softmax}(\mathbf{M}'_{UV}) \\ \mathbf{W}_U^V &= \mathbf{M}_{UV} \cdot (\mathbf{W}_V \mathbf{V}), \end{aligned} \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{d_U \times d_w}$ and $\mathbf{V} \in \mathbb{R}^{d_V \times d_w}$ are two matrices that convert word embedding sequences to dimension d_w, and $\mathbf{M}'_{UV}$ contains dot products between each word in $\mathbf{W}_U$ and $\mathbf{W}_V$, and softmax is applied on $\mathbf{M}'_{UV}$ row-wise. Three types of attention are calculated using Equation (1): (1) question-aware passage representation $\mathbf{W}_P^Q \in \mathbb{R}^{p \times d_w}$; (2) question-aware choice representation $\mathbf{W}_C^Q \in \mathbb{R}^{c \times d_w}$; and (3) passage-aware choice representation $\mathbf{W}_C^P \in \mathbb{R}^{c \times d_w}$.

3.1.3 Sequence Modeling Layer

To model the contextual dependency of each text sequence, we use BiLSTMs to process the word representations obtained from the input layer and attention layer:

$$\begin{aligned} \mathbf{H}^q &= \mathbf{BiLSTM}[\mathbf{W}_Q] \\ \mathbf{H}^c &= \mathbf{BiLSTM}[\mathbf{W}_C; \mathbf{W}_C^P; \mathbf{W}_C^Q] \\ \mathbf{H}^p &= \mathbf{BiLSTM}[\mathbf{W}_P; \mathbf{W}_P^Q], \end{aligned} \tag{2}$$

---

[2] https://www.elastic.co/products/elasticsearch

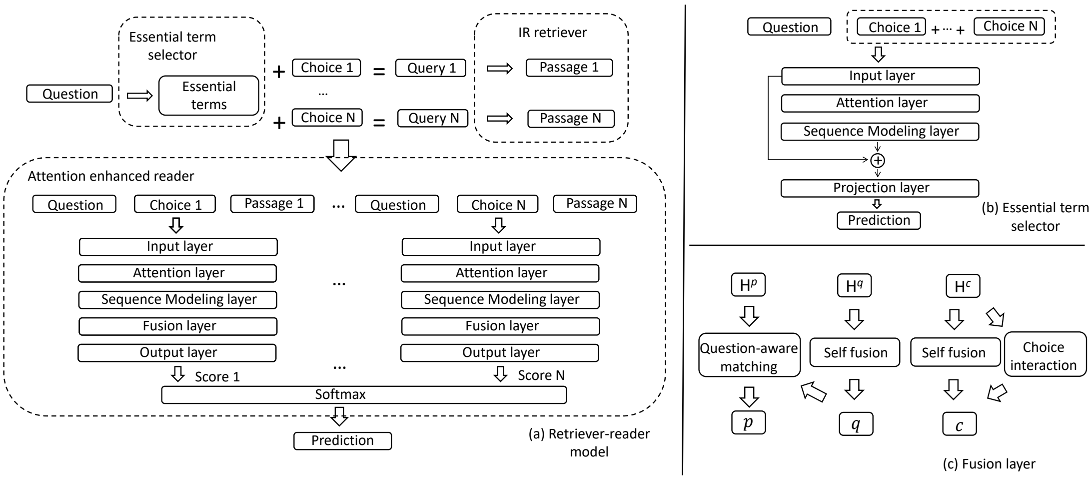


Figure 2: Model structure for our essential-term-aware retriever-reader model.

where $\mathbf{H}^q \in \mathbb{R}^{q\times l}, \mathbf{H}^c \in \mathbb{R}^{c\times l}$, and $\mathbf{H}^p \in \mathbb{R}^{p\times l}$ are the hidden states of the BiLSTMs, ';' is feature-wise concatenation, and l is the size of the hidden states.

3.1.4 Fusion Layer

We further convert each question and answer choice into a single vector: $\mathbf{q} \in \mathbb{R}^l$ and $\mathbf{c} \in \mathbb{R}^l$:

$$\begin{aligned}\boldsymbol{\alpha}_q &= \text{softmax}([\mathbf{H}^q; \mathbf{w}_e] \cdot \mathbf{w}_{sq}^\top),\ \mathbf{q} = \mathbf{H}^{q\top}\boldsymbol{\alpha}_q\\ \boldsymbol{\alpha}_c &= \text{softmax}(\mathbf{H}^c \cdot \mathbf{w}_{sc}^\top),\ \mathbf{c} = \mathbf{H}^{c\top}\boldsymbol{\alpha}_c,\end{aligned} \tag{3}$$

where the essential-term feature $\mathbf{w}_e$ from Section 3.1.1 is concatenated with $\mathbf{H}^q$, and $\mathbf{w}_{sq}$ and $\mathbf{w}_{sc}$ are learned parameters.

Finally, a bilinear sequence matching is calculated between $\mathbf{H}^p$ and $\mathbf{q}$ to obtain a question-aware passage representation, which is used as the final passage representation:

$$\boldsymbol{\alpha}_p = \text{softmax}(\mathbf{H}^p \cdot \mathbf{q}); \quad \mathbf{p} = \mathbf{H}^{p\top}\boldsymbol{\alpha}_p. \tag{4}$$

3.1.5 Choice Interaction

When a QA task provides multiple choices for selection, the relationship between the choices can provide useful information to answer the question. Therefore, we integrate a choice interaction layer to handle the semantic correlation between multiple answer choices. Given the hidden state $\mathbf{H}^{c_n}$ of choice c_n and $\mathbf{H}^{c_i}$ of other choices $c_i, \forall i \neq n$, we calculate the differences between the hidden states and apply max-pooling over the differences:

$$\mathbf{c}_{inter} = \text{Maxpool}(\mathbf{H}^{c_n} - \frac{1}{N-1}\sum_{i\neq n}\mathbf{H}^{c_i}), \tag{5}$$

where N is the total number of answer choices. Here, $\mathbf{c}_{inter}$ characterizes the differences between an answer choice c_n and other answer choices. The final representation of an answer choice is updated by concatenating the self-attentive answer choice vector and inter-choice representation as $\mathbf{c}^{final} = [\mathbf{c}; \mathbf{c}_{inter}]$.

3.1.6 Output Layer

For each tuple $\{\mathbf{q}, \mathbf{p}_n, \mathbf{c}_n\}_{n=1}^N$, two scores are calculated by matching (1) the passage and answer choice and (2) question and answer choice. We use a bilinear form for both matchings. Finally, a softmax function is applied over N answer choices to determine the best answer choice:

$$\begin{aligned} s_n^{pc} = \mathbf{p}_n\mathbf{W}^{pc}\mathbf{c}_n^{final}; \quad s_n^{qc} = \mathbf{q}\mathbf{W}^{qc}\mathbf{c}_n^{final}\\ \mathbf{s} = \text{softmax}(\mathbf{s}^{pc}) + \text{softmax}(\mathbf{s}^{qc}),\end{aligned} \tag{6}$$

where s_n^{pc}, s_n^{qc} are the scores for answer choice $1 \leq n \leq N$; $\mathbf{s}^{pc}, \mathbf{s}^{qc}$ are score vectors for all N choices; and $\mathbf{s}$ contains the final scores for each answer choice. During training, we use a cross-entropy loss.

3.2 Essential Term Selector

Essential terms are key words in a question that are crucial in helping a retriever obtain related evidence. Given a question $\mathbf{Q}$ and N answer choices $\mathbf{C}_1, \ldots, \mathbf{C}_N$, the goal is to predict a binary variable y_i for each word Q_i in the question $\mathbf{Q}$, where $y_i = 1$ if Q_i is an essential term and 0 otherwise. To address this problem, we build a neural model, ET-Net, which has the same design as the reader model for the input layer, attention layer, and sequence modeling layer to obtain the hidden state $\mathbf{H}^q$ for question $\mathbf{Q}$.

In detail, we take the question $\mathbf{Q}$ and the concatenation $\mathbf{C}$ of all N answer choices as input to

Question	If an object is attracted to a magnet, the object is most likely made of (A) wood (B) plastic (C) cardboard (D) metal
# annotators	5
Annotation	If,0; an,0; object,3; is,0; attracted,5; to,0; a,0; magnet,,5; the,0; object,1; is,0; most,0; likely,0; made,2; of,0

Table 2: Example of essential term data.

ET-Net. $\mathbf{Q}$ and $\mathbf{C}$ first go through an input layer to convert to the embedded word representation, and then word-level attention is calculated to obtain a choice-aware question representation $\mathbf{W}_Q^C$ as in Equation (1). We concatenate the word representation and word-level attention representation of the question and feed it into the sequence modeling layer:

$$\mathbf{H}^q = \mathbf{BiLSTM}[\mathbf{W}_Q; \mathbf{W}_Q^C]. \tag{7}$$

As shown in Figure 2, the hidden states obtained from the attention layer are then concatenated with the embedded representations of $\mathbf{Q}$ and fed into a projection layer to obtain the prediction vector $\mathbf{y} \in \mathbb{R}^q$ for all words in the question:

$$\mathbf{y} = [\mathbf{H}^q; \mathbf{W}_Q^f] \cdot \mathbf{w}^s, \tag{8}$$

where $\mathbf{w}^s$ contains the learned parameters, and $\mathbf{W}_Q^f$ is the concatenation of the POS embedding, NER embedding, relation embedding, and feature embedding from Section 3.1.1.

After obtaining the prediction for each word, we use a binary cross-entropy loss to train the model. During evaluation, we take words with y_i greater than 0.5 as essential terms.

4 Experiments

In this section, we first discuss the performance of the essential term selector, ET-Net, on a public dataset. We then discuss the performance of the whole retriever-reader pipeline, ET-RR, on multiple open-domain datasets.For both the ET-Net and ET-RR models, we use 96-dimensional hidden states and 1-layer BiLSTMs in the sequence modeling layer. A dropout rate of 0.4 is applied for the embedding layer and the BiLSTMs' output layer. We use adamax (Kingma and Ba, 2014) with a learning rate of 0.02 and batch size of 32. The model is trained for 100 epochs. Our code is released at `https://github.com/nijianmo/arc-etrr-code`.

Model	Precision	Recall	F1
MaxPMI	0.88	0.65	0.75
SumPMI	0.88	0.65	0.75
PropSurf	0.68	0.64	0.66
PropLem	0.76	0.64	0.69
ET Classifier	0.91	0.71	0.80
ET-Net	0.74	**0.90**	**0.81**

Table 3: Performance of different selectors.

4.1 Performance on Essential Term Selection

We use the public dataset from Khashabi et al. (2017) which contains 2,223 annotated questions, each accompanied by four answer choices. Table 2 gives an example of an annotated question. As shown, the dataset is annotated for binary classification. For each word in the question, the data measures whether the word is an "essential" term according to 5 annotators. We then split the dataset into training, development, and test sets using an 8:1:1 ratio and select the model that performs best on the development set.

Table 3 shows the performance of our essential term selector and baseline models from Khashabi et al. (2017). The second best model (ET Classifier) is an SVM-based model from Khashabi et al. (2017) requiring over 100 handcrafted features. As shown, our ET-Net achieves a comparable result with ET Classifier in terms of the F1 Score.

Table 4 shows example predictions made by ET-Net. As shown, ET-Net is capable of selecting most ground-truth essential terms. It rejects certain words such as "organisms" which have a high TF-IDF in the corpus but are not relevant to answering a particular question. This shows its ability to discover essential terms according to the context of the question.

4.2 Performance on Open-domain Multiple-choice QA

We train and evaluate our proposed pipeline method ET-RR on four open-domain multiple-choice QA datasets. All datasets are associated with a sentence-level corpus. Detailed statistics are shown in Table 5.

- ARC (Clark et al., 2018): We consider the 'Challenge' set in the ARC dataset and use the provided corpus during retrieval.
- RACE-Open: We adapted the RACE dataset (Lai et al., 2017) to the open-domain setting. Originally, each question in RACE comes

Example questions
Which **unit** of **measurement** can be used to describe the **length** of a **desk** ?
One way **animal** usually **respond** to a sudden **drop** in **temperature** is by
Organisms require energy to survive. Which of the following **processes provides energy** to the body ?

Table 4: Examples of essential term prediction (in questions) by ET-Net. True positives are marked bold and underlined while false positives are only underlined.

Dataset	Train	Dev	Test	Corpus
ARC	1,119	299	1,172	1.46M
RACE-Open	9,531	473	528	0.52M
MCScript-Open	1,036	156	319	24.2K
Amazon-Patio	36,587	4,531	4,515	2.55M
Amazon-Auto	49,643	6,205	6,206	7.32M
Amazon-Cell	40,842	5,105	5,106	1.86M

Table 5: Statistics on ARC, RACE-Open, MCScript-Open and Amazon-QA. Corpus size is the number of sentences.

Dataset	Example questions
ARC	The best way to separate salt from water is with the use of Which geologic process most likely caused the formation of the Mount St. Helens Volcano?
RACE-Open	According to the article, what does the band Four Square hope to do in the future? According to the article we know it is _ to prevent the forests from slowly disappearing.
Amazon-QA	For anyone with small ears, do these fit comfortably or do they feel like they are always going to fall out, not in correctly, etc. Does it remove easily and does it leave any sticky residue behind? thanks in advance.

Table 6: Example of predictions on ARC, RACE-Open and Amazon-QA. Predicted terms are underlined.

with a specific passage. To enable passage retrieval, we concatenate all passages into a corpus with sentence deduplication.[3]

- MCScript-Open: The MCScript (Ostermann et al., 2018) dataset is also adapted to the open-domain setting. Again we concatenate all passages to build the corpus.[4]
- Amazon-QA: The Amazon-QA dataset (McAuley and Yang, 2016) is an open-domain QA dataset covering over one million questions across multiple product categories. Each question is associated with a free-form answer. We adapt it into a 2-way multiple-choice setting by randomly sampling an answer from other questions as an answer distractor. We split all product reviews at the sentence-level to build the corpus. We consider three categories from the complete dataset in our experiments.

In the experiments, ET-RR uses ET-Net to choose essential terms in the question. Table 6 shows example predictions on these target datasets. Then it generates a query for each of the N answer choices by concatenating essential terms and the answer choice. For each query, ET-RR obtains the top K sentences returned by the retriever and considers these sentences as a passage for the reader. We set $K = 10$ for all experiments.

We compare ET-RR with existing retrieve-and-read methods on both datasets. As shown in Table 7, on the ARC dataset, ET-RR outperforms all previous models without using pre-trained models and achieves a relative 8.1% improvement over the second best BiLSTM Max-out method (Mihaylov et al., 2018). Recently, finetuning on pre-trained models has shown great improvement over a wide range of NLP tasks. Sun et al. (2019) proposed a 'Reading Strategies' method to finetune the pre-trained model OpenAI GPT, a language model trained on the BookCorpus dataset (Radford, 2018). They trained Reading Strategies on the RACE dataset to obtain more auxiliary knowledge and then finetune that model on the ARC corpus. Table 8 demonstrates the performance comparison of ET-RR and Reading Strategies on ARC. As shown, though Reading Strategies trained on both ARC and RACE dataset outperforms ET-RR, ET-RR outperforms Reading Strategies using only the ARC dataset at training time.

On the RACE-Open and MCScript-Open datasets, ET-RR achieves a relative improvement of 24.6% and 10.5% on the test set compared with the second best method IR solver respectively. We also evaluate on multiple categories of

---

[3] As short questions might not contain any words which can relate the question to any specific passage or sentence, we only keep questions with more than 15 words.

[4] We keep questions with more than 10 words rather than 15 words to ensure that there is sufficient data.

Model	ARC Test	RACE-Open Test	MCScript-Open Test
IR solver	20.26	30.70	60.46
Random	25.02	25.01	50.02
BiDAF	26.54	26.89	50.81
BiLSTM Max-out	33.87	/	/
ET-RR (Concat)	35.33	36.87	66.46
ET-RR	**36.61**	**38.61**	**67.71**

Table 7: Accuracy on multiple-choice selection on ARC, RACE-Open and MCScript-Open.

Training Corpus	model	ARC
ARC	Reading Strategies	35.0
	ET-RR	36.6
ARC+RACE	Reading Strategies	40.7

Table 8: Comparisons of ET-RR and Reading Strategies on ARC.

Model	Amazon -Patio	Amazon -Auto	Aamzon -Cell
IR solver	72.80	73.60	70.50
Moqa	84.80	86.30	88.60
ET-RR (Concat)	96.19	95.21	93.26
ET-RR	**96.61**	**95.96**	**93.81**

Table 9: Accuracy on multiple-choice selection on three product categoris of Amazon-QA.

Pre-trained	model	RACE
✓	Reading Strategies	63.8
✓	OpenAI GPT	59.0
	ET-RR (reader)	**52.3**
	Bi-attn (MRU)	50.4
	Hier. Co-Matching	50.4

Table 10: Experimental results for reader on RACE.

Model	Test
ET-RR	**36.61**
– inter-choice	36.36
– passage-choice	35.41
– question-choice	34.47
– passage-question	34.05

Table 11: Ablation test on attention components of ET-RR on ARC. '–' denotes the ablated feature.

the Amazon-QA dataset. As shown in Table 9, ET-RR increases the accuracy by 10.33% on average compared to the state-of-the-art model Moqa (McAuley and Yang, 2016). We also compare ET-RR with ET-RR (Concat), which is a variant of our proposed model that concatenates the question and choice as a query for each choice. Among all datasets, ET-RR outperforms ET-RR (concat) consistently which shows the effectiveness of our essential-term-aware retriever.

4.3 Ablation study

We investigate how each component contributes to model performance.

Performance of reader. Our reader alone can be applied on MRC tasks using the given passages. Here, we evaluate our reader on the original RACE dataset to compare with other MRC models as shown in Table 10. As shown, the recently proposed Reading Strategies and OpenAI GPT models, that finetune generative pre-trained models achieve the highest scores. Among non-pre-trained models, our reader outperforms other baselines: Bi-attn (MRU) (Tay et al., 2018) and Hierarchical Co-Matching (Wang et al., 2018a) by a relative improvement of 3.8%.

Attention components. Table 11 demonstrates how the attention components contribute to the performance of ET-RR. As shown, ET-RR with all attention components performs the best on the ARC test set. The performance of ET-RR without passage-question attention drops the most significantly out of all the components. It is worth noting that the choice interaction layer gives a further 0.24% boost on test accuracy.

Essential term selection. To understand the contribution of our essential-term selector, we compare ET-RR with two variants: (1) ET-RR (Concat) and (2) ET-RR (TF-IDF). For ET-RR (TF-IDF), we calculate the TF-IDF scores and take words with the top 30% of TF-IDF scores in the question to concatenate with each answer choice as a query.[5]

Table 12 shows an ablation study comparing different query formulation methods and amounts of retrieved evidence K. As shown, with the essential term selector ET-Net, the model consistently outperforms other baselines, given different numbers of retrievals K. Performance for all models is best when $K = 10$. Furthermore, only using TF-IDF to select essential terms in a question is not effective. When $K = 10$, the ET-RR (TF-IDF)

---

[5] According to the annotated dataset, around 30% of the terms in each question are labelled as essential.

Model	ET-RR (Concat)		ET-RR (TF-IDF)		ET-RR	
Top K	Dev	Test	Dev	Test	Dev	Test
5	39.26	33.36	39.93	34.73	39.93	35.59
10	38.93	35.33	39.43	35.24	**43.96**	**36.61**
20	41.28	34.56	38.59	33.88	42.28	35.67

Table 12: Comparison of query formulation methods and amounts of retrieved evidence (i.e., top K) on the ARC dataset, in terms of percentage accuracy.

method performs even worse than ET-RR (Concat). This illustrates the challenges in understanding what is essential in a question.

Though ET-RR consistently outperforms ET-RR (TF-IDF), the improvement is relatively modest on the Test set (around 1.4%). A similar outcome has been reported in Jansen et al. (2017); Khashabi et al. (2017) where essential term extraction methods have shown around 2%-4% gain compared with TF-IDF models and struggle to obtain further improvement on SQA tasks. This consensus might show the discrepancy of essential terms between human and machine (i.e., the essential terms obtained using a human annotated dataset might not be helpful in a machine inference model). Another reason might be the current retrieval method does not effectively use these essential terms and the performance highly depends on the dataset. Note that the ET-RR outperforms ET-RR (TF-IDF) by around 4% on the Dev set. Therefore, how to develop well-formed single or even multi-hop queries using these terms are worth studying in the future.

4.4 Error Analysis

Table 13 shows two major types of error, where the correct answer choice is in **bold** and the predicted answer choice is in *italics*.

Retrieved supporting evidence but failed to reason over it. For the first question, there exists evidence that can justify the answer candidate (C). However, the model chooses (D) which has more words overlapping with its evidence. This shows that the model still lacks the reasoning capability to solve complex questions.

Failed to retrieve supporting evidence. For the second question, the retrieved evidence of both the correct answer (D) and the prediction (B) is not helpful to solve the question. Queries such as 'what determines the year of a planet' are needed to acquire the knowledge for solving this question. This poses further challenges to design a retriever that can rewrite such queries.

The elements carbon, hydrogen, and oxygen are parts of many different compounds. Which explains why these three elements can make so many different compounds?
(A) They can be solids, liquids, or gases. (B) They come in different sizes and shapes. **(C) They combine in different numbers and ratios.** * There are many different types of compounds because atoms of elements combine in many different ways (and in different whole number ratios) to form different compounds. *(D) They can be a proton, a neutron, or an electron.* * Atoms of different elements have a different number of protons, neutrons, and electrons.
Which planet in the solar system has the longest year?
(A) The planet closest to the Sun. *(B) The planet with the longest day.* * The planet with the longest day is Venus; a day on Venus takes 243 Earth days. (C) The planet with the most moons. **(D) The planet farthest from the Sun.** * The last planet discovered in our solar system is farthest away from the sun.

Table 13: Examples where ET-RR fails on ARC. The retrieved evidence for each answer candidate is marked by *.

5 Conclusion

We present a new retriever-reader model (ET-RR) for open-domain QA. Our pipeline has the following contributions: (1) we built an essential term selector (ET-Net) which helps the model understand which words are essential in a question leading to more effective search queries when retrieving related evidence; (2) we developed an attention-enhanced reader with attention and fusion among passages, questions, and candidate answers. Experimental results show that ET-RR outperforms existing QA models on open-domain multiple-choice datasets as ARC Challenge, RACE-Open, MCScript-Open and Amazon-QA. We also perform in-depth error analysis to show the limitations of current work. For future work, we plan to explore the directions of (1) constructing multi-hop query and (2) developing end-to-end retriever-reader model via reinforcement learning.

Acknowledgments

We thank Jade Huang for proofreading the paper, Liang Wang and Daniel Khashabi for sharing code and the annotated dataset with us. We thank all the reviewers for their constructive suggestions.

References

Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock. 2018. A systematic classification of knowledge, reasoning, and context within the arc dataset. In *1st Workshop on Machine Reading for Question Answering*.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *CoRR*, abs/1803.05457.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *ICLR*.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics*, 43:407–449.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke S. Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *CoNLL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Heeyoung Kwon, Harsh Trivedi, Peter Jansen, Mihai Surdeanu, and Niranjan Balasubramanian. 2018. Controlling information aggregation for complex question answering. In *ECIR*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. *SIGIR Forum*, 51:260–267.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *WWW*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *ACL*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *CIKM*.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *EMNLP*.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Semeval-2018 task 11: Machine comprehension using commonsense knowledge. In *SemEval@NAACL-HLT*.

Alec Radford. 2018. Improving language understanding by generative pre-training. In *Technical report*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *NAACL*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *CoRR*, abs/1803.09074.

Liang Wang. 2018. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *SemEval@NAACL-HLT*.

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018a. A co-matching model for multi-choice reading comprehension. In *ACL*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018b. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering

Zi-Yuan Chen[1], **Chih-Hung Chang**[1], **Yi-Pei Chen**[2], **Jijnasa Nayak**[3], **Lun-Wei Ku**[1,4]
Institute of Information Science, Academia Sinica[1]
University of Massachusetts Amherst[2]
National Institute of Technology, Rourkela[3]
Most Joint Research Center for AI Technology and All Vista Healthcare[4]
{zychen,lance5,lwku}@iis.sinica.edu.tw
yipeichen@cs.umass.edu, jijnasa23@gmail.com

Abstract

In relation extraction for knowledge-based question answering, searching from one entity to another entity via a single relation is called "one hop". In related work, an exhaustive search from all one-hop relations, two-hop relations, and so on to the max-hop relations in the knowledge graph is necessary but expensive. Therefore, the number of hops is generally restricted to two or three. In this paper, we propose UHop, an unrestricted-hop framework which relaxes this restriction by use of a transition-based search framework to replace the relation-chain-based search one. We conduct experiments on conventional 1- and 2-hop questions as well as lengthy questions, including datasets such as WebQSP, PathQuestion, and Grid World. Results show that the proposed framework enables the ability to halt, works well with state-of-the-art models, achieves competitive performance without exhaustive searches, and opens the performance gap for long relation paths.

1 Introduction

A knowledge graph (KG) is a powerful graph structure that encodes knowledge to save and organize it, and to provide users with direct access to this knowledge via various applications, one of which is question answering, or knowledge-based question answering (KBQA). In the knowledge graph, beliefs are commonly represented by triples showing relations between two entities, such as LocatedIn(NewOrleans, Louisiana), where the two entities are nodes and their relation is the edge connecting them in the knowledge graph. Given a natural language question, a KBQA system returns its answer if it is included in the knowledge graph; the process of answering a question can be transformed into a traversal that starts from the question (topic) entity and searches for the appropriate path to the answer entity.

In the literature (Yu et al., 2017; Yin et al., 2016; Yih et al., 2015) KBQA is decomposed into topic entity linking, which determines the starting entity corresponding to the question, and relation extraction, which finds the path to the answer node(s). Theoretically, relation extraction finds paths of any length, that is, paths that contain any number of relation links, or hops (between two nodes), as long as it reaches the answer node. In previous work, models consider all relation paths starting from the topic entity (Yu et al., 2017; Yin et al., 2016; Yih et al., 2015); we call these relation-chain-based methods. Two main difficulties for these methods are that processing through all relations in a KG is not practical as the combination of these relations is nearly infinite, and that the number of candidate paths grows exponentially with the path length and quickly becomes intractable for large knowledge graphs. As a result, current relation-chain-based methods set the maximum length of candidate paths to 1, 2 or 3. However, under this framework we cannot find answer entities for indirect or complicated questions.

Most importantly, even given a larger maximum length, it is unrealistic to expect to know in advance the maximum number of hops for real-world applications. Thus even with exhaustive searches, if the answer entity is still too distant or lies outside of the search space, it is not reachable or answerable. In addition, setting a large maximum number of hops necessitates lengthy training instances, which is especially difficult.

In this paper, we propose UHop, an unrestricted-hop relation extraction framework to relax restrictions on candidate path length. We decompose the task of relation extraction in the knowledge graph into two subtasks: knowing where to go, and knowing when to stop (or to halt). That is, single-hop relation extraction and termination decision. Our contribution is three-

Proceedings of NAACL-HLT 2019, pages 345–356
Minneapolis, Minnesota, June 2 - June 7, 2019.

fold: (1) No predefined maximum hop number is required in UHop, as it enables models within the framework to halt; (2) UHop reduces the search space complexity from exponential to polynomial while maintaining comparable results; (3) UHop facilitates the use of different models, including state-of-the-art models.

2 Related Work

State-of-the-art KBQA methods are in general based on either semantic parsing, or on embedding (Zhou et al., 2018). Semantic parsing methods learn semantic parsers which parse natural language input queries into logical forms, and then use the logical forms to query the KG for answers (Berant et al., 2013; Yih et al., 2015, 2016; Krishnamurthy et al., 2017; Iyyer et al., 2017; Peng et al., 2017; Sorokin and Gurevych, 2018). These systems are effective and provide deep interpretation of the question, but require expensive data annotation, or require training using reinforcement learning.

Embedding-based methods first allocate candidates from the knowledge graph, represent these candidates as distributed embedding vectors, and choose or rank these vectors. Here the candidates can be either entities or relations. Some use embedding-based models to predict answers directly (Dong et al., 2015; Bast and Haussmann, 2015; Hao et al., 2017; Zhou et al., 2018; Lukovnikov et al., 2017), whereas others focus on extracting relation paths and require further procedures to select the answer entity (Bordes et al., 2015; Xu et al., 2016; Yin et al., 2016; Yu et al., 2017; Zhang et al., 2018a; Yu et al., 2018; Chen et al., 2018a; Shen et al., 2018). Our work follows the latter methods in focusing on predicting relation paths, but we seek to eliminate the need to assume in advance a maximum number of hops.

For the solution, we turn to the field of multi-hop knowledge based reasoning. Early methods include the Path-Ranking Algorithm and its variants. (Lao et al., 2011; Gardner et al., 2014, 2013; Toutanova et al., 2015) The drawback of these methods is that they use random walks independent of the type of input. DeepPath (Xiong et al., 2017) and MINERVA (Das et al., 2017) tackle this issue by framing the multi-hop reasoning problem as a Markov decision process, efficiently searching for paths using reinforcement learning; others propose an algorithm (Yang et al., 2017) for learning logical rules, a variational auto-encoder view of the knowledge graph (Chen et al., 2018b; Zhang et al., 2018b), and reward shaping technique (Lin et al., 2018) for further improvement. The major difference between UHop and these methods is that they do not utilize annotated relations and hence require REINFORCE training (Williams, 1992) for optimization. As some datasets are already annotated with relations and paths, direct learning using an intermediate reward is more reasonable. Hence UHop adopts a novel comparative termination decision module to control the search process of the relation path.

The most related approach is the IRN model (Zhou et al., 2018), composed of an input module, a memory-based reasoning module, and an answer module. At each hop, it predicts a relation path using the reasoning module, and also optimizes it using intermediate results. However, UHop has demonstrated the ability to process large-scale knowledge graphs in experiments conducted on Freebase (Bordes et al., 2015). In contrast, IRN consumes memory linearly to the size of the knowledge graph, resulting in a limited workspace, e.g., they use a subset of Freebase in their experiments. Also, IRN still uses a constraint for the number of maximum hops in the experiments, while UHop needs no such limit. Most importantly, as UHop is a framework which facilitates the use of different models, we can expect the performance of UHop to remain competitive with the state of the art over time.

3 UHop Relation Extraction

With UHop, we aim to handle unrestricted relation hops and to be compatible with existing relation extraction models. UHop breaks down unrestricted-hop relation extraction into two major subtasks: single-hop relation extraction and comparative termination decision.

Algorithm 1 illustrates how we perform these two tasks in the UHop framework. Given a question Q and the topic entity e extracted by an existing entity linking method such as S-MART (Yang and Chang, 2015), we first query the knowledge graph for the candidate outbound relations R that are connected to e. For all relations R, we extract single-hop relations in order to choose one relation to transit to the next entity e'. After transition ($e \leftarrow e'$), we decide whether to terminate, that is, we determine whether the process should proceed

Algorithm 1: Unrestricted-hop relation extraction. e denotes the extracted topic entity, ':' is the concatenation operation, and the termination decision returns *True* if the framework decides to stop.

```
Given KB, $Q$, $e$
$stop \leftarrow False$;
$P \leftarrow NULL$;
$R \leftarrow$ outbound relations of $e$;
while $stop = False$ do
    $\hat{r} \leftarrow$ single hop relation extraction;
    $P \leftarrow P : \hat{r}$;
    $e' \leftarrow$ traverse from $e$ through $\hat{r}$;
    $e \leftarrow e'$;
    $R \leftarrow$ outbound relations of $e$;
    $stop \leftarrow$ termination decision;
end
Result: $P$
```

through another iteration to extract the next relation in the relation path. If the decision to terminate is false, we search the KB again for outbound relations of the new e, after which the search process starts again. Note that starting from the second iteration, candidate relations are concatenated with the previously selected relations to remember the history and consider them as a whole. We continue this loop until the process decides to terminate. The termination decision thus enables UHop to learn when to stop searching for relations to extract: it determines the number of hops needed to reach the correct target entity. Upon termination, UHop returns the extracted relation(s).

In the UHop framework, the model is trained to favor the correct relation over incorrect relations. That is, to select the correct outbound single-hop relations from current entity e, the model prefers the correct $\hat{r}$ over the other relations $R - \hat{r}$ of e; to terminate at the current entity e, the model favors the correct relation $\hat{r}$ linked to the current entity e over the outbound R relations from e. To continue the iteration, it proceeds likewise. In UHop, we successfully utilize this preference over relations to train the same model to perform both single-hop relation extraction and termination decision. Figure 1 shows the difference between previous work and our model in the scenario of multi-hop KBQA task with an simplified knowledge graph and the question "Who published the novel adapted into A Study in Pink ?" as example.

3.1 Single Hop Relation Extraction

Single-hop relation extraction can be modeled as pairwise classification of the set of candidate relations. Given a question Q, the candidate relation set R, and a pairwise classification model F, single-hop relation extraction is illustrated as

$$r = \arg\max_{r \in R} F(Q, r). \tag{1}$$

Hinge loss, used for optimization, is defined as

$$\mathcal{L}_{RE} = \frac{\sum\limits_{r \in R - \hat{r}} \max(0, -(s^{\hat{r}} - s^{r}) + M)}{|R - \hat{r}|}, \tag{2}$$

where $s^{\hat{r}}$, s^{r} are scores of the true relation and the candidate relations respectively. The margin, M, is an arbitrary value in the range $(0, 1]$, where the goal of the loss function is to maximize the margin between the scores of the correct and the incorrect predictions. Note that this relation extraction process and those proposed in related work are compatible, which facilitates the installation of state-of-the-art models in the UHop framework.

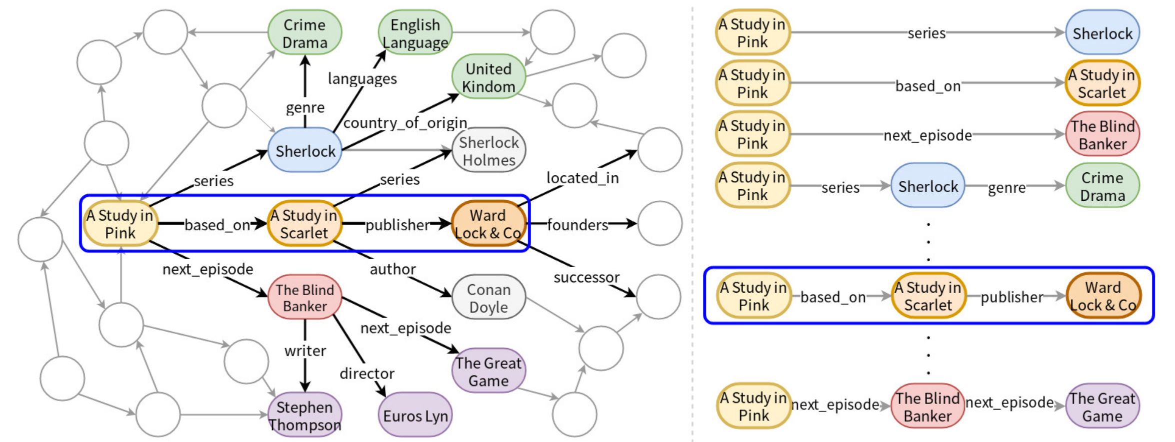
3.2 Comparative Termination Decision

In the UHop framework, as we hope to easily replace the used model by state-of-the-art models, we make the termination decision using the same model for single-hop relation extraction so that no additional model is needed. Therefore, we propose a progressive method which treats the termination decision as a comparison. That is, the model stops when it cannot extract any relation better than that from its previous hop.

What is different here is R, the relations to be compared against $\hat{r}$, are the concatenation of extracted relation and all the relation starting from the new current entity e; recall that we update $e \leftarrow e'$ before we step into termination decision. If the score $s^{\hat{r}}$ is higher than all the compared relations, the searching process terminates; otherwise, it continues.

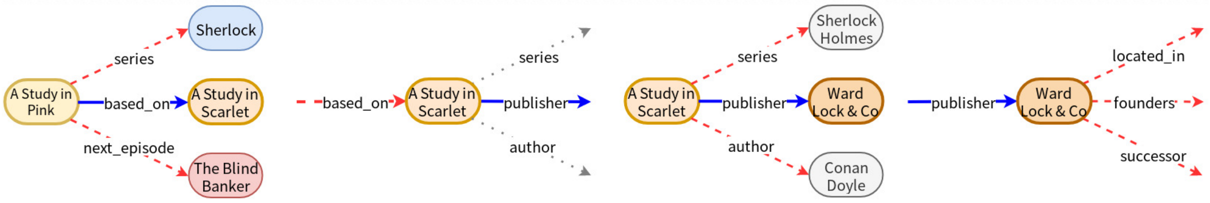
Given a question Q, an extracted relation $\hat{r}$ from the previous entity, the candidate relation set R from the new current entity e, and the same model F as in the single hop relation extraction, the procedure can be formulated as

$$stop = \begin{cases} True, & F(Q, \hat{r}) > F(Q, r) \ \forall r \in R \\ False, & F(Q, \hat{r}) < F(Q, r) \ \exists r \in R \end{cases} \tag{3}$$



(a)

(b) without UHop



(1) first hop relation extraction

(2) termination decision (stop_flag=false)

(3) second-hop relation extraction

(4) termination decision (stop_flag=true)

(c) UHop

Figure 1: (a) A snippet of knowledge graph. (b) without UHop, all the paths less than two hop are considered as candidates; (c) under UHop, the next part of relation is extracted hop by hop (step 1 and 3), then we compare the chosen relation and its outbound relations to decide to terminate or to extract the next relation: if the extracted relation has the highest score than all the outbound relations then the process is terminated (step 4), otherwise, continued (step 2). Here we use solid arrows and dash arrows to respectively represent positive/negative candidates.

Loss is defined depending on the flag *stop*. If the process should continue, i.e., *stop* is false, loss is defined as

$$\mathcal{L}_{TD} = \max(0, -(s^{r'} - s^{\hat{r}}) + \text{margin}), \quad (4)$$

where score $s^{r'}$ is the score of the question paired with the gold relation r' in the next hop and $s^{\hat{r}}$ is the score of the question paired with the extracted relation $\hat{r}$. In contrast, if the process should terminate, we optimize the model by

$$\mathcal{L}_{TD} = \frac{\sum_{r \in R} \max(0, -(s^{\hat{r}} - s^{r}) + M)}{|R|}. \quad (5)$$

The model thus learns to infer $s^{\hat{r}}$ is greater than s^{r}, resulting in the termination of relation extraction.

3.3 Dynamic Question Representation

While UHop inferences hop by hop, it is straightforward to enforce the focus at different aspects of the question. For this purpose, we update the question representation for each hop by defining a dynamic question representation generation function G. Given the previously selected relation path P and the original question Q, G generates the new question representation as $Q' = G(Q, P)$. Our assumption is that since the current relation has been selected, its related information in the question loses importance when extracting the next relation.

Inspired by both supervised attention (Mi et al., 2016; Liu et al., 2016; Kamigaito et al., 2017), which is lacking in our datasets, and the coverage loss design for summarization (See et al., 2017), we de-focus the selected relation by manipulating weights in the question representation. We propose two ways of updating the question representation, taking into account the existence of the attention layer in the model's architecture. For attentive models, we directly utilize the attention weight as part of our dynamic question representation generation function by

$$G(Q, P) = W(Q - \mathit{attention}(Q, P)) + B. \quad (6)$$

For non-attentive models, we apply a linear transformation function as G on the concatenation of the previously selected relation and the question representation to yield the new representation:

$$G(Q, P) = W[Q : P] + B, \tag{7}$$

where W and B are weight matrices to be optimized during training.

3.4 Jointly Trained Subtasks

In training, we jointly optimize the two subtasks of UHop. For each question and its candidates, the loss function is defined as

$$\mathcal{L} = \sum_{i}^{H}(\mathcal{L}_{RE}^{(i)} + \mathcal{L}_{TD}^{(i)}), \tag{8}$$

where H is the number of hops in the gold relation path; $\mathcal{L}_{RE}^{(i)}$ and $\mathcal{L}_{TD}^{(i)}$ are the loss of the two subtasks at the i-th hop respectively.

4 Experiments

In this section, we illustrate the performance UHop achieves while reducing the search space, and its relation inference power for multi-hop questions. Performances of the state of the art models are listed as the upper-bound.

4.1 Datasets

For our benchmarking evaluation materials, we selected WebQSP (WQ) (Yih et al., 2016), as it is used in most related work. WebQSP is the annotated version of WebQuestions (Berant et al., 2013), which contains questions that require a 1- or 2-hop relation path to arrive at the answer entity. More specifically, about 40% of the questions require a 2-hop relation to reach the answer. This dataset is based on the Freebase knowledge graph (Bordes et al., 2015). For questions with multiple answers, we use each answer to construct a question-answer pair. Every question is annotated with its inferential relation chain (i.e., a relation), topic entity, and answer entity. The statistics for these two datasets are shown in Table 1.

As WQ contains only questions with 1- and 2-hop answers that are still short, we also conduct experiments for path length related analysis on the PathQuestion dataset (Zhou et al., 2018), which includes questions requiring 3-hop answers. To the best of our knowledge, this is the only available general-KB dataset containing 3-hop questions. PathQuestion provides two datasets: PathQuestion (PQ) and PathQuestion-Large (PQL). These both contain 2-hop (PQ2/PQL2) and 3-hop (PQ3/PQL3) questions respectively, and both use a subset of Freebase as their knowledge graph. Note that for both PQ and PQL, questions are generated using templates, paraphrasing, and synonyms. PQL is more challenging than PQ because it utilizes a larger subset of Freebase, and provides fewer training instances. Table 1 shows statistics of these datasets.

	hops	Train	Valid	Test
WQ	1	2113	-	1,144
	2	1,285	-	647
PQ2	2	1,526	191	191
PQ3	3	4,158	520	520
PQL2	2	1,275	159	160
PQL3	3	1,649	206	207
Grid 2–4	2–4	68,046	9,742	19,298
Grid 4–6	4–6	73,092	10,362	21,037
Grid 6–8	6–8	41,473	5,844	11,789
Grid 8–10	8–10	18,386	2,667	5,326

Table 1: Number of questions in experimental datasets

The above datasets serve to show that the UHop framework yields performance competitive with state-of-the-art KBRE models. Further, we seek to demonstrate that UHop reduces the search space when required reasoning paths are even longer, i.e., longer than 3 hops, and that UHop works for different kinds of relations. For this we use Grid World (Yang et al., 2017), a synthetic dataset with questions requiring lengthy – up to 10 hops – relation paths to answer. We select it to demonstrate that UHop works for long as well as task-specific relations. In Grid World, the input is the starting node, a sequence of navigation instructions, and a 16-by-16 fully connected grid. The model must follow the instructions to arrive at the destination node. Specifically, the task is to navigate to an answer cell (answer entity) starting from a random cell (topic entity) given a sequence of instructions (questions). The KB consists of triples such as ((4, 1), South, (5, 1)), which indicates that the entity (5, 1) is south of the entity (4, 1); questions are sequences of directions such as (North, NorthEast, South). Samples in Grid World are classified into 4 buckets – [2–4], [4–6], [6–8], and [8–10] – according to their reasoning path length. Unlike relations included in general knowledge bases like Freebase, relations in Grid World are the relative directions of two nodes.

MetaQA (Zhang et al., 2018b) and sequence QA are two other multi-hop knowledge-based question-answering datasets which we do not use for experiments in this paper. MetaQA is a multi-hop dataset for end-to-end KBQA based on a movie knowledge graph with 43k entities. However, it is too simple for discussions as it contains only 6 relations and on average the number of the outbound relations for each node is 3. The Complex Sequential QA dataset (Saha et al., 2018) improves on overly simplistic KBQA datasets. Nevertheless, instead of questions requiring multi-hop relation paths, it provides a sequence of questions, each of which requires a single-hop relation to answer, resulting a different setting. Hence these two datasets are beyond the scope of this paper.

4.2 Benchmark: WQ Experiments

4.2.1 Baseline and Settings

We used two state of the art models, HR-BiLSTM (Yu et al., 2017) and ABWIM (Zhang et al., 2018a), as the models for use within the UHop framework. Another state of the art model, MVM (Yu et al., 2018), is not selected here as it requires additional information: the tail entity type. In MVM, to consider each n-th-hop relation, the model searches all related $(n+1)$-th-hop relations to collect enough information; thus further queries are necessary in MVM. This property of MVM causes the UHop to degrade to a relation-chain based model, which we are trying to avoid.

We report the results of these two models working within and independent of the UHop framework to evaluate whether relaxing the constraint on the number of hops has any impact on their performance. For comparison, we select BiCNN as baselines and list their results. As there is no predefined validation set in WQ, we randomly select 10% of the training data as the validation set. The best parameters for different models and datasets were set empirically.

In all cases we used 300-dimensional pretrained GloVe (Pennington et al., 2014) word embeddings and RMSprop optimization. In ABWIM, following the setting of (Zhang et al., 2018a), we respectively chose 1, 3, 5 as kernel sizes and 150 as the number of filters for its three CNN layers. We tune the following hyperparameters with grid search : (1) the hidden size for all LSTM ([100, 150, 256]); (2) dropout rate ([0, 0.2, 0.4]); (3) margin for Hinge loss ([0.1, 0.3, 0.5, 0.7, 1.0]); (4) learning rate ([0.01, 0.001, 0.0001]).

Method	Accuracy
BiCNN (Yih et al., 2015)	77.74
HR-BiLSTM (Yu et al., 2017)	82.53
ABWIM (Zhang et al., 2018a)	83.26[1]
HR-BiLSTM with UHop	82.60
ABWIM with UHop	82.27

Table 2: Results adopting state-of-the-art models in UHop framework vs standalone versions

4.2.2 Results and Discussion

The experimental results are shown in Table 2. As expected, the performance of models within the UHop framework is comparable to those independent of it, with the additional advantage of the unrestricted number of relation hops and a greatly reduced search space.

Table 3 lists the average number of candidates the experimental models consider for each question when working within and independent of UHop. For a dataset based on a KB with an average of n relations connected to each entity, the approximate search space without UHop is $n(n-1)^{(L-1)}$, where L is the predefined maximum hop number; with UHop the approximate search space is reduced to $n(L+1)$. The specific number depends on the actual number of outbound relations connected to the entities. Table 3 shows that UHop reduces the search space by 30% for WQ, which translates to lower processing time, less memory consumption, and sometimes slightly improved performance.

	Train	Test
Without UHop	97.2	98.8
With UHop	66.7	65.6

Table 3: Number of relation candidates in WQ

4.3 More Hops: PQ/PQL Experiments

4.3.1 Baseline and Settings

Following the original paper (Zhou et al., 2018), PQ and PQL are both partitioned into training/validation/testing sets at a ratio of 8:1:1. In addition to the original PQ/PQL dataset, we merge

---

[1] Note that the original paper reported 85.32, but we failed to reproduce such performance. Hence we report our reproduced performance which is the same model adapted in our proposed framework.

Method	PQ2	PQ3	PQ+	PQL2	PQL3	PQL+
IRN (Zhou et al., 2018)	96	87.7	53.6	72.5	71	52.9
HR-BiLSTM (Yu et al., 2017)	100	99.62	99.72	97.5	87.92	92.92
HR-BiLSTM with UHop	99.48	99.23	99.72	91.25	88.41	91.01
HR-BiLSTM with UHop + DQ	100	99.62	99.58	95	89.37	91.83
ABWIM (Zhang et al., 2018a)	98.95	99.81	99.72	94.37	89.37	92.64
ABWIM with UHop	97.38	99.62	99.02	91.25	88.89	91.01
ABWIM with UHop + DQ	100	99.62	99.44	97.5	89.37	92.37

Table 4: Accuracy on PathQuestion. PQ+ is mix of PQ2 and PQ3, and PQL+ contains PQL2 and PQL3. We use the accuracy reported in (Zhou et al., 2018) directly for PQ2, PQ3, PQ2L and PQ3L; for PQ+ and PQL+ we use the model released with the dataset. DQ stands for dynamic question representation.

PQ2 and PQ3, and then PQL2 and PQL3, to create the mixed datasets PQ+ and PQL+ to evaluate if the model terminates correctly instead of always stopping on the majority of the training data length. Again we adopt HR-BiLSTM and ABWIM in this experiment. In addition, the IRN model[2] proposed together with the PQ/PQL dataset was selected as one of the baselines for comparison. For this dataset containing questions of long relation paths, we also applied the dynamic question representations (DQ) in UHop.

4.3.2 Results and Discussion

Results[3] are shown in Table 4. Both HR-BiLSTM and ABWIM either within or independent of UHop outperform IRN and perform nearly perfectly in all datasets, which confirms that UHop is competitive even with longer relation paths. However, as shown in Table 5, the search space reduction for PQ and PQL is not obvious. We find that the knowledge graph used in PQ/PQL (a subset of Freebase) is much smaller and less complicated than the original Freebase used in WQ, i.e., the outbound degree of nodes is relatively small. Nevertheless, UHop still performs comparably with previous work. This indicates that it also works well in small and simple KBs.

As all PQ/PQL questions are multi-hop questions, we used dynamic question representations to better reflect transitions in the relation extraction process. Table 4 shows that updating the question representation dynamically (+DQ) in each iteration benefits relation extraction in most cases.

		2-hop			3-hop		
		Train	Valid	Test	Train	Valid	Test
PQ	W/o	3.53	3.65	3.90	9.77	9.63	9.61
	With	3.68	3.81	3.85	9.14	9.27	9.48
PQL	W/o	3.71	2.64	3.91	24.52	12.96	8.45
	With	3.94	3.23	4.29	10.11	9.28	6.98

Table 5: Candidates of PQ/PQL within and independent of the UHop framework

4.4 Very Long Paths: Grid World

4.4.1 Baseline and Settings

In the Grid World experiments, we used MINERVA (Das et al., 2017) and Neural LP (Yang et al., 2017) as baselines. As understanding questions is not an issue here, we randomly initialized the word embeddings and optimized them during the training process. We set the learning rate to 0.001, the hidden size to 256, the embedding size to 300, and optimized the model using the RMSprop (Hinton et al., 2014) Algorithm. In this experiment, the search space has gone too large to afford for HR-BiLSTM and ABWIM without the assistance of UHop.

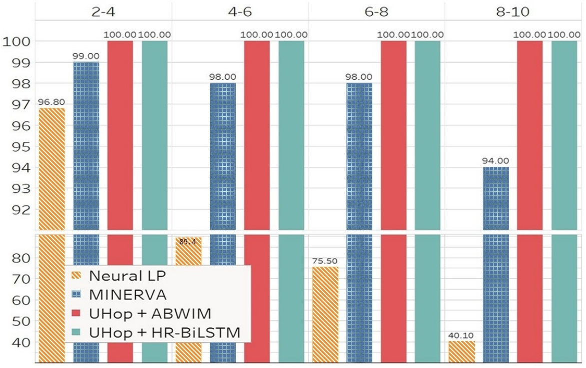


Figure 2: Grid World results of state-of-the-art knowledge-based relation extraction models

4.4.2 Results and Discussion

The results in Figure 2 show that together with the relation extraction model, UHop perfectly solves

---

[2] https://github.com/zmtkeke/IRN. We consulted the authors of the repository, who stated that this version is not the one in their paper, which they did not release publicly.

[3] Note that IRN's performance was evaluated using final answer prediction, which is slightly different from relation path prediction. However, finding the correct relation path should imply finding the correct answer entity.

this problem. In the first place, compared to Neural LP and MINERVA, UHop benefits from the more powerful natural language understanding models – HR_BiLSTM and ABWIM – equipped with sophisticated LSTM models, whereas Neural LP and MINERVA only use multi-layer neural networks as the policy network. This demonstrates UHop's merit of facilitating the use of novel models. In the second place, Figure 2 shows that error propagation leading to poor performance for long-path questions in Neural LP and MINERVA is mitigated by the relation inference power of UHop: it performs well for all four buckets of questions. Also, as Grid World includes paths of up to 10 hops, conducting experiments purely by relation-chain based models themselves like HR-BiLSTM or ABWIM independent of UHop is not feasible: the number of candidate relations in the exhaustive search space grows exponentially. In Grid World, there are 8 directions (relations), and models are allowed to go back and forth. Hence given the path length k, the approximate search space for the models working independently is 8^k, while for models working within UHop is $8 \times k$. We observe that without UHop, the required search space would preclude experiments even on the set containing the shortest paths (Grid World [2–4]), much less the longer ones.

5 Further Discussion

5.1 Dataset Characteristics

In this section we further compare the experimental multi-hop KBQA datasets WQ, PQ, and Grid World. Grid World contains questions that require the longest reasoning paths. However, they are synthetic, the relations are simply direction tokens, and the questions are just sequences of direction instructions. Therefore in this paper, it is only used to test the model's ability of making long sequential decisions instead of understanding questions. From experiments we have seen that delicate models like HR-BiLSTM and ABWIM cannot work on it without UHop, and other models such as Neural LP and MINERVA perform worse as they are rewarded only by question.

On the other hand, in WQ, questions are written in natural language and can be answered by 1-hop or 2-hop reasoning. However, for real-world questions, 2-hop reasoning is still overly simplistic. For example, although WQ questions such as "What is the name of Justin Bieber's brother?" are challenging for models, humans can easily answer these with a simple Internet search.

Noting this problem, the authors of IRN (Zhou et al., 2018) propose PQ and PQL, for which questions require at least 2-hop at most 3-hop relation paths. However, PQ/PQL also has its limitations. First, the KB used in PQ/PQL is smaller than that in WQ, and its relations are repetitive and show little variety. Figure 3 illustrates the relation distributions. Second, PQ/PQL questions are generated by extracting relation paths and filling templates, which can lead to questions with obvious, learnable patterns. This can be observed by comparing results in Tables 2 and 4. However, repeated relations could also help the model to learn better dynamic question representations with respect to these relations. Table 4 shows that updating question representations dynamically (DQ) does improve PQ/PQL performance.

Figure 3: A visualization of the KB relations that cover the dataset. The bubble's size is proportional to the relation's frequency.

5.2 Trained on 3-hop, Tested on 2-hop

To evaluate if the model halts in the search process, we conducted an experiment using PQL3 as the training/validation set and PQL2 as the testing set. The results are shown in Table 6. Within the UHop framework, both models outperform their original version by more than 7%. However, with zero 2-hop samples, it still overfits on the 3-hop length in training data, resulting in accuracies lower than 50%.

	HR-BiLSTM	ABWIM
Without UHop	32.18	32.94
With UHop	39.65	49.94

Table 6: Accuracies of models trained on PQL3 and tested on PQL2. The maximum length of relation paths for models without UHop is set to 3.

5.3 Error Analysis

The interpretability of UHop, i.e., the possibility to analyze each hop, facilitates the analysis of error distributions. We list the percentage of questions for which UHop fails to extract the correct relations by the number of hops for different

Dataset/model			1-hop		2-hop		3-hop	
			RE	TD	RE	TD	RE	TD
WQ	1-hop	H	17.46	0	-	-	-	-
		A	20.95	0.1	-	-	-	-
	2-hop	H	16.15	1.03	1.2	0	-	-
		A	18.38	0.86	2.06	0.17	-	-
PQ2		H	0.52	0	0	0	-	-
		H*	0	0	0	0	-	-
		A	2.09	0	0.52	0	-	-
		A*	0	0	0	0	-	-
PQ+	2-hop	H	0	0	0	0	-	-
		H*	0	0	0	0	-	-
		A	0	0	0.52	0	-	-
		A*	0	0	0	0	-	-
	3-hop	H	0	0	0	0	0.38	0
		H*	0	0	0	0	0.58	0
		A	0	0	0	0	1.15	0
		A*	0	0	0	0	0.77	0
PQ3		H	0	0	0.19	0	0.58	0
		H*	0	0	0	0	0.38	0
		A	0	0	0	0	0.38	0
		A*	0	0	0	0	0.38	0
PQL2		H	5.62	0	3.12	0	-	-
		H*	2.5	0	2.5	0	-	-
		A	5.0	0	3.75	0	-	-
		A*	0	0	2.5	0	-	-
PQL+	2-hop	H	0	0	3.12	0	-	-
		H*	0	0	3.75	0	-	-
		A	0	0	3.75	0	-	-
		A*	0	0	2.5	0	-	-
	3-hop	H	0.48	0	5.31	0	7.73	0
		H*	0	0	2.9	0	8.7	0
		A	0	0	3.86	1.93	7.25	0
		A*	0	0	2.9	0.97	7.73	0
PQL3		H	0	0	3.86	0	7.73	0
		H*	0	0	2.9	0	7.73	0
		A	0	0	2.9	0.97	7.25	0
		A*	0	0	2.9	0	7.73	0
Train3, Test2		H	1.13	2.95	2.7	53.58	-	-
		A	0.25	2.38	7.4	40.03	-	-

Table 7: Distribution of error types under UHop framework (in percentage). H stands for HR-BiLSTM, A for ABWIM, RE for 'Relation Extraction', and TD for 'Termination Decision'. * denotes the +DQ setting.

datasets. The results of HR_BiLSTM and ABWIM within the UHop framework are reported in Table 7. Our observations are offered below.

First, whether for 1-hop or 2-hop WQ questions, both models suffer in relation extraction in the first hop, whereas there are fewer errors in the second hop and for the termination decision.

Second, for the PQ/PQL datasets, as with the WQ dataset, incorrect relation extraction is the major error, and surprisingly there were no errors for termination decision except for a few on PQL3 with ABWIM. After comparing the 2-hop testing data from PQ2/PQL2 and PQ+/PQL+, we also observe that long questions help the learning of short questions. The model predicts better on 2-hop data when trained on both 2-hop and 3-hop data than when trained on 2-hop data only. Here the improvement in relation extraction in the first hop is the main contributor to this improved performance. In contrast, the performance on 3-hop data suffers when trained on 2-hop data.

Third, dynamic question representations (noted by *) significantly benefit the relation extraction (RE) for the first hop. As UHop utilizes the same model for relation selection and termination decision, relieving the attention to the previous relation in the later selection process in the training phase decreases the ambiguity in the earlier selection process in the testing phase.

Finally, in the experiments trained on 3-hop and tested on 2-hop, the model does not terminate correctly on more than 40% of the PQL2 data even though the relation extraction for 1-hop and 2-hop are both correct. We conclude that having no samples of the predicted length for training still hurts performance. In addition, there are also a few early terminations after the first relation extraction. Due to the different generation processes with different templates for the 2-hop and 3-hop questions in PQL, learning from one may not apply to the other.

6 Conclusion

In this paper, we propose the UHop framework to allow an unrestricted number of hops in knowledge-based relation extraction and to reduce the search space. Results show that running the same model in the UHop framework achieves comparable results in a reduced search space. Moreover, experiments show UHop works well for lengthy relation extraction and can be applied to small, simple KBs with task-specific relations. UHop even facilitates the use of most state-of-the-art models, and its transition-based design naturally supports the dynamic question representation for better performance. These results attest its strong power for knowledge-based relation extraction. The current framework uses a greedy search for each single hop. We expect in the future that incorporating a beam search may further improve performance.

Acknowledgements

This research is partially supported by Ministry of Science and Technology, Taiwan under Grant no. MOST108-2634-F-002-008-, and the Academia Sinica Thematic Project under Grant no. 233b-1070100.

References

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on Freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Hung-Chen Chen, Zi-Yuan Chen, Sin-Yi Huang, Lun-Wei Ku, Yu-Shian Chiu, and Wei-Jen Yang. 2018a. Relation extraction in knowledge base question answering: From general-domain to the catering industry. In *HCI in Business, Government, and Organizations*, pages 26–41, Cham. Springer International Publishing.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018b. Variational knowledge graph reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1823–1832.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2014. Notes from Lecture 6a of Neural Networks for Machine Learning: Overview of mini-batch gradient descent.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 7–12.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 31-November 4, 2018*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2283–2288.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017. Maximum margin reward networks for learning from explicit and implicit supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2368–2378.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368.*

Cun Shen, Tinglei Huang, Xiao Liang, Feng Li, and Kun Fu. 2018. Chinese knowledge base question answering by attention-based multi-granularity model. *Information*, 9(4):98.

Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690.*

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2326–2336.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328.

Yi Yang and Ming-Wei Chang. 2015. S-MART: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 504–513.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–206.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 571–581.

Yang Yu, Kazi Saidul Hasan, Mo Yu, Wei Zhang, and Zhiguo Wang. 2018. Knowledge base relation detection via multi-view matching. *CoRR*, abs/1803.00612.

Hongzhi Zhang, Guandong Xu, Xiao Liang, Tinglei Huang, and Kun Fu. 2018a. An attention-based word-level interaction model: Relation detection for knowledge base question answering. *CoRR*, abs/1801.09893.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018b. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of* *the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2010–2022.

BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering

Yu Cao[1] **Meng Fang**[2] **Dacheng Tao**[1]
[1]UBTECH Sydney AI Center, School of Computer Science, FEIT, The University of Sydney
[2]Tencent Robotics X, China
ycao8647@uni.sydney.edu.au, mfang@tencent.com,
dacheng.tao@sydney.edu.au

Abstract

Multi-hop reasoning question answering requires deep comprehension of relationships between various documents and queries. We propose a Bi-directional Attention Entity Graph Convolutional Network (BAG), leveraging relationships between nodes in an entity graph and attention information between a query and the entity graph, to solve this task. Graph convolutional networks are used to obtain a relation-aware representation of nodes for entity graphs built from documents with multi-level features. Bidirectional attention is then applied on graphs and queries to generate a query-aware nodes representation, which will be used for the final prediction. Experimental evaluation shows BAG achieves state-of-the-art accuracy performance on the QAngaroo WIKIHOP dataset.

1 Introduction

Question Answering (QA) and Machine Comprehension (MC) tasks have drawn significant attention during the past years. The proposal of large-scale single-document-based QA/MC datasets, such as SQuAD (Rajpurkar et al., 2016), CNN/Daily mail (Hermann et al., 2015), makes training available for end-to-end deep neural models, such as BiDAF (Seo et al., 2016), DCN (Xiong et al., 2016) and SAN (Liu et al., 2017). However, gaps still exist between these datasets and real-world applications. For example, reasoning is constrained to a single paragraph, or even part of it. Extended work was done to meet practical demand, such as DrQA (Chen et al., 2017) answering a SQuAD question based on the whole Wikipedia instead of single paragraph. Besides, latest large-scale datasets, e.g. TriviaQA (Joshi et al., 2017) and NarrativeQA (Kočiskỳ et al., 2018), address this limitation by introducing multiple documents, ensuring reasoning cannot be done within local information. Although those datasets are fairly challenging, reasoning are within one document.

In many scenarios, we need to comprehend the relationships of entities across documents before answering questions. Therefore, reading comprehension tasks with multiple hops were proposed to make it available for machine to tackle such problems, e.g. QAngaroo task (Welbl et al., 2018). Each sample in QAngaroo contains multiple supporting documents, and the goal is selecting the correct answer from a set of candidates for a query. Most queries cannot be answered depending on a single document, and multi-step reasoning chains across documents are needed. Therefore, it is possible that understanding a part of paragraphs loses effectiveness for multi-hop inference, which posts a huge challenge for previous models. Some baseline models, e.g. BiDAF (Seo et al., 2016) and FastQA (Weissenborn et al., 2017), which are popular for single-document QA, suffer dramatical accuracy decline in this task.

In this paper, we propose a new graph-based QA model, named Bi-directional Attention Entity Graph convolutional network (BAG). Documents are transformed into a graph in which nodes are entities and edges are relationships between them. The graph is then imported into graph convolutional networks (GCNs) to learn relation-aware representation of nodes. Furthermore, we introduce a new bi-directional attention between the graph and a query with multi-level features to derive the mutual information for final prediction.

Experimental results demonstrate that BAG achieves state-of-the-art performance on the WIKIHOP dataset. Ablation test also shows BAG benefits from the bi-directional attention, multi-level features and graph convolutional networks.

Our contributions can be summarized as:

- Applying a bi-directional attention between graphs and queries to learn query-aware representation for reading comprehension.
- Multi-level features are involved to gain comprehensive relationship representation for graph nodes during processing of GCNs.

Proceedings of NAACL-HLT 2019, pages 357–362
Minneapolis, Minnesota, June 2 - June 7, 2019.

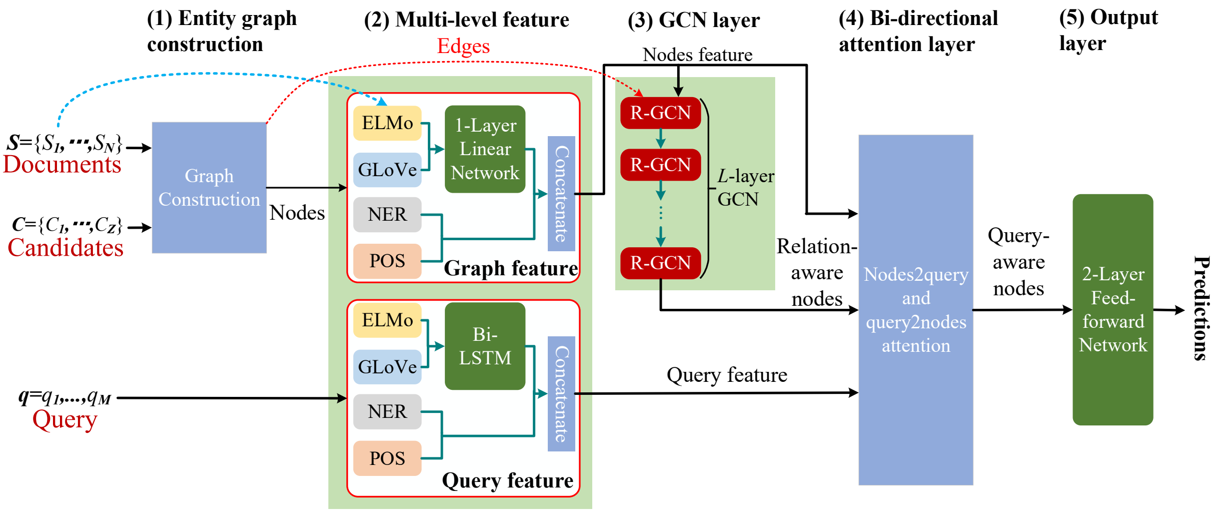


Figure 1: Framework of BAG model.

2 Related Work

Recently coreference and graph-based models are studied for multi-hop QA (Dhingra et al., 2018; Santoro et al., 2017). Coref-GRU (Dhingra et al., 2018) uses coreferences among tokens in documents. However, it is still limited by the long-distance relation propagation capability of RNNs. Besides, graph is proved to be an efficient way to represent complex relationships among objects and derive relational information (Santoro et al., 2017). MHQA-GRN (Song et al., 2018) and Entity-GCN (De Cao et al., 2018) construct entity graphs based on documents to learn more compact representation for multi-hop reasoning and derive answers from graph networks. However, both of them care less about input features and the attention between queries and graph nodes.

Attention has been proven to be an essential mechanism to promote the performance of NLP tasks in previous work (Bahdanau et al., 2014; Sukhbaatar et al., 2015). In addition, bi-directional attention (Seo et al., 2016) shows its superiority to vanilla mutual attention because it provides complementary information to each other for both contexts and queries. However, little work exploits the attention between graphs and queries.

3 BAG Model

We first formally define the multiple-hop QA task, taking QAngaroo (Welbl et al., 2018) WIKIHOP data as an example, There is a set $\boldsymbol{S}$ containing N supporting documents, a query $\boldsymbol{q}$ with M tokens and a set of answer candidates $\boldsymbol{C}$. Our goal is to find the correct answer index a. Giving a triple-style query $\boldsymbol{q} = (\textit{country, kepahiang})$, it means *which country does kepahiang belongs to*. Then answer candidates are provided, e.g. $C = \{Indonesia, Malaysia\}$. There are multiple supporting documents but not all of them are related to reasoning, e.g. *Kephiang is a regency in Bengkulu*, *Bengkulu is one of provinces of Indonesia*, *Jambi is a province of Indonesia*. We can derive the correct candidate is *Indonesia*, i.e. $a = 0$, based on reasoning hops in former two documents.

We show the proposed BAG model in Figure 1. It contains five modules: (1) entity graph construction, (2) multi-level feature layer, (3) GCN layer, (4) bi-directional attention and (5) output layer.

3.1 Entity Graph Construction

We construct an entity graph based on Entity-GCN (De Cao et al., 2018), which means all mentions of candidates found out in documents are used as nodes in the graph. Undirected edges are defined according to positional properties of every node pair. There are two kinds of edges included: 1) cross-document edge, for every node pair with the same entity string located in different documents; 2) within-document edge, for every node pair located in the same document.

Nodes in an entity graph can be found out via simple string matching. This approach can simplify calculation as well as make sure all relevant entities are included in the graph. Picked out along possible reasoning chains during dataset generating (Welbl et al., 2018), answer candidates have contained all related entities for answering. Finally, We can obtain a set of T nodes $\{n_i\}, 1 \leq i \leq T$ and corresponding edges among these nodes via above procedures.

3.2 Multi-level Features

We represent both nodes and queries using multi-level features as shown in Figure 1(2). We first use pretrained word embeddings to represent tokens, such as GLoVe (Pennington et al., 2014) because nodes and queries are composed of tokens. Then contextual-level feature is used to offset the deficiency of GLoVe. Note that only part of tokens are remained during graph construction because we only extract entities as nodes. Thus contextual information around these entities in original document becomes essential for indicating relations between tokens and we use higher-level information for nodes except for token-level feature.

We use ELMo (Peters et al., 2018) as contextualized word representations, modeling both complex word characteristics and contextual linguistic conditions. It should be noted that ELMo features for nodes are calculated based on original documents, then truncated according to the position indices of nodes. Token-level and context-level features will be concatenated and encoded to make a further comprehension. Since a node may contain more than one token, we average features among tokens to generate a feature vector for each node before encoding it. It will be transformed into the encoded node feature via a 1-layer linear network.

Different from nodes, we represent a query by directly using a bidirectional LSTM (Bi-LSTM) whose output in each step is used as encoded query features. And both linear network and LSTM have the same output dimension $\hat{d}$.

In addition, we add two manual features to reflect the semantic properties of tokens, which are named-entity recognition (NER) and part-of-speech (POS). The complete feature $f_n \in \mathbb{R}^{T \times d}$, $f_q \in \mathbb{R}^{M \times d}$ for both nodes and queries will be the concatenation of corresponding encoded features, NER embedding and POS embedding, where $d = \hat{d} + d_{POS} + d_{NER}$.

3.3 GCN Layer

In order to realize multi-hop reasoning, we use a Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018) that can propagate message across different entity nodes in graphs and generate transformed representation of original ones. The R-GCN is employed to handle high-relational data characteristics and make use of different edge types. At lth layer, given the hidden state $h_i^l \in \mathbb{R}^d$ of node i, the hidden states $h_j^l \in \mathbb{R}^d, j \in \{N_i\}$ and relations R_{N_i} of all its neighbors (d is the hidden state dimension), the hidden state in the next layer can be obtained via

$$h_i^{l+1} = \sigma\Big(\sum_{r \in R_{N_i}} \sum_{j \in N_i} \frac{1}{c_{i,r}} W_r^l h_j^l + W_0^l h_i^l \Big), \quad (1)$$

where $c_{i,r}$ is a normalization constant $|N_i|$, $W_r^l \in \mathbb{R}^{d \times d}$ stands a relation-specific weight matrix and $W_0^l \in \mathbb{R}^{d \times d}$ stands a general weight.

Similar to Entity-GCN (De Cao et al., 2018), we apply a gate on update vector u_i^l and hidden state h_i^l of current node by a linear transformation f_s,

$$w_i^l = \sigma(f_s(\text{concat}(u_i^l, h_i^l))), \quad (2)$$

in which u_i^l can be obtained via (1) without sigmoid function. Then it will be used for updating weights for the hidden state h_i^{l+1} of the same node in next layer,

$$h_i^{l+1} = w_i^l \odot \tanh(u_i^l) + (1 - w_i^l) \odot h_i^l. \quad (3)$$

We stack such networks for L layers in which all parameters are shared. The information of each node will be propagated up to L-node distance away, generating L-hop-reasoning relation-aware representation of nodes. The initial input will be mutli-level nodes features $\mathbf{f_n} = \{f_{n_i}\}, 0 \leq i \leq T$ and edges $\mathbf{e} = \{e_{ij}\}$ in the graph.

3.4 Bi-directional Attention Between a Graph and a Query

Bi-directional attention is responsible for generating the mutual information between a graph and a query. In BiDAF (Seo et al., 2016), attention is applied to sequence data in QA tasks such as supporting texts. However, we also find it works well between graph nodes and queries. It generates query-aware node representations that can provide more reasoning information for prediction.

What differs in BAG is that attention is applied for graphs as shown in Figure 1(4). The similarity matrix $\mathbf{S} \in \mathbb{R}^{\mathbf{T} \times \mathbf{M}}$ is calculated via

$$\mathbf{S} = \text{avg}_{-1} f_a(\text{concat}(h_n, f_q, h_n \circ f_q)), \quad (4)$$

in which $h_n \in \mathbb{R}^{T \times d}$ is all node representations obtained from the last GCN layer, $f_q \in \mathbb{R}^{M \times d}$ is the query feature matrix after encoding, d is the dimension for both query feature and transformed node representation, f_a is a linear transformation, avg_{-1} stands for the average operation in last dimension, and $\circ$ is element-wise multiplication.

Unlike the context-to-query attention in BiDAF, we introduce a node-to-query attention $\widetilde{a}_{n2q} \in \mathbb{R}^{T \times d}$, which signifies the query tokens that have the highest relevancy for each node using

$$\widetilde{a}_{n2q} = \mathrm{softmax}_{\mathrm{col}}(\mathbf{S}) \cdot f_q, \tag{5}$$

where $\mathrm{softmax}_{\mathrm{col}}$ means performing softmax function across the column, and $\cdot$ stands for matrix multiplication.

At the same time, we also design query-to-node attention $\widetilde{a}_{q2n} \in \mathbb{R}^{M \times d}$ which signifies the nodes that are most related to each token in the query via

$$\widetilde{a}_{q2n} = \mathrm{dup}(\mathrm{softmax}(\max_{\mathrm{col}}(\mathbf{S})))^{\top} \cdot f_n\,, \tag{6}$$

in which $\max_{\mathrm{col}}$ is the maximum function applied on across column of a matrix, which will transform $\mathbf{S}$ into $\mathbb{R}^{1 \times M}$. Then function dup will duplicate it for T times into shape $\mathbb{R}^{T \times M}$. $f_n \in \mathbb{R}^{T \times d}$ is the original node feature before GCN layer.

Our bi-directional attention layer is the concatenation of the original nodes feature, nodes-to-query attention, the element-wise multiplication of nodes feature and nodes-to-query attention, and multiplication of nodes feature and query-to-nodes attention. It should be noted that the relation-aware nodes representation from GCN layer is just used to calculate the similarity matrix, and original node feature is used in rest calculation to obtain more general complementary information between graph and query. Edges are not taken in account because they are discrete and combined with nodes in GCN layer. The output is defined as

$$\widetilde{a} = \mathrm{concat}(f_n, \widetilde{a}_{n2q}, f_n \circ \widetilde{a}_{n2q}, f_n \circ \widetilde{a}_{q2n}). \tag{7}$$

3.5 Output layer

A 2-layer fully connect feed-forward network is employed to generate the final prediction, with $\tanh$ as the activation function in each layer. Softmax will be applied among the output. It uses query-aware representation of nodes from the attention layer as input, and its output is regarded as the probability of each node becoming answer. Since each candidate may appear several times in the graph, the probability of each candidate is the sum of all corresponding nodes. The loss function is defined as the cross entropy between the gold answer and its predicted probability.

4 Experiment

We used both unmasked and masked versions of the QAngaroo WIKIHOP dataset (Welbl et al., 2018) and followed its basic setting, in which masked version used specific tokens such as *__MASK1__* to replace original candidates tokens in documents. There are 43,738, 5,129 and 2,451 examples in the training set, the development set and the test set respectively, and test set is not public.

In the implementation[1], we used standard ELMo with a 1024 dimension representation. Besides, 300-dimension GLoVe pre-trained embeddings from 840B Web crawl data were used as token-level features. We used spaCy to provide additional 8-dimension NER and POS features. The dimension of the 1-layer linear network for nodes in multi-level feature module was 512 with $\tanh$ as activation function. A 2-layer Bi-LSTM was employed for queries whose hidden state size is 256. Then the feature dimension is $d = 512 + 8 + 8 = 528$. The GCN layer number L was set as 5. And the unit number of intermediate layers in output layer was 256.

In addition, the number of nodes and the query length were truncated as 500 and 25 respectively for normalized computation. Dropout with rate 0.2 was applied before GCN layer. Adam optimizer is employed with initial learning rate 2×10^{-4}, which will be halved for every 5 epochs, With batch size 32. It took about 14 hours for 50-epoch training on two GTX1080Ti GPUs using pre-built and pre-processed graph data generated from original corpus, which can significantly decrease the training time.

We consider the following baseline models: FastQA (Weissenborn et al., 2017), BiDAF (Seo et al., 2016), Coref-GRU (Dhingra et al., 2018), MHQA-GRN (Song et al., 2018), Entity-GCN (De Cao et al., 2018). Former three models are RNN-based models, while coreference relationship is involved in Coref-GRU. The last two models are graph-based models specially designed for multi-hop QA tasks.

As shown in Table 1, we collected three kinds of results. The dev and test results stand for the original validation and test sets respectively, noting that the test set is not public. In addition, we divide the original validation set of masked version into two parts evenly, one as a split validation set for tuning model and the other one as a split test set. The test[1] results are for the split test set.

Our BAG model achieves state-of-the art per-

---

[1] Source code is available on `https://github.com/caoyu1991/BAG`.

formance on both unmasked and masked data[2], with accuracy 69.0% on the test set, which is 1.4% higher in value than previous best model Entity-GCN. It is significant superior than FastQA and BiDAF due to leveraging of relationship information given by the graph and abandoning some distracting context in multiple documents. Although Coref-GRU extends GRU with coreference relationships, it is still not enough for multi-hop because hop relationships are not limited to coreference, entities with the same strings also existed across documents which can be used for reasoning. Both MHQA-GRN and Entity-GCN utilize graph networks to resolve relations among entities in documents. However, the lack of attention and complementary features limits their performance. Therefore our BAG model achieves the best performance under all data configurations. It is noticed that BAG only gets a small promotion on masked data. We argue that the reason is the attention between masks and queries generating less useful information compared to unmasked ones.

Moreover, ablation experimental results on unmasked version of the WIKIHOP dev set are given in Table 2. Once we remove the bi-directional attention and put the concatenation of nodes and queries directly into the output layer, it shows significant performance drop with more than 3%, proving the necessity of attention for reasoning in multi-hop QA. If we use linear-transformation-based single attention $a = h_n \mathbf{W_a} f_q$ given in (Luong et al., 2015) instead of our bi-directional attention, the accuracy drops with 2%, which means attention bi-directionality also contributes to the performance improvement. The similar condition will appear if we remove GCN, but use raw nodes as input for the attention layer.

In addition, if edge types are no longer considered, which makes R-GCN degraded to vanilla GCN, noticeable accuracy loss about 2% appears. The absence of multi-level features will also cause degradation. The removal of semantic-level features causes slight decline on the performance, including NER and POS features. Further removal of ELMo feature will causes a dramatical drop, which reflects the insufficiency of only using word embeddings as features for nodes and that contextual information is very important.

Models	**Unmasked**		**Masked**	
	dev	test	dev	test[1]
FastQA	27.2*	25.7	38.0*	48.3
BiDAF	49.7*	42.9	59.8*	57.5
Coref-GRU†	56.0*	59.3	-	-
MHQA-GRN‡	62.8*	65.4	-	-
Entity-GCN	64.8*	67.6	70.5*	68.1
BAG	**66.5**	**69.0**	**70.9**	**68.9**

Table 1: The performance of different models on both masked and unmasked version of WIKIHOP dataset. ([*] Results reported in original papers, others are obtained by official code. [†] Masked data is not suitable for coreference parsing. [‡] Some results are missing due to unavailability of source code.)

Models	**Unmasked**
Without Attention	63.1
Using Single Attention	64.5
Without GCN	63.3
Without edge type	63.9
Without NER, POS	66.0
+Without ELMo	60.5
Full Model	**66.5**

Table 2: Ablation test results of BAG model on the unmasked validation set of the WIKIHOP dataset.

5 Conclusion

We propose a Bi-directional Attention entity Graph convolutional network (BAG) for multi-hop reasoning QA tasks. Regarding task characteristics, graph convolutional networks (GCNs) are efficient to handle relationships among entities in documents. We demonstrate that both bi-directional attention between nodes and queries and multi-level features are necessary for such tasks. The former one aims to obtain query-aware node representation for answering, while the latter one provides contextual comprehension of isolated nodes in graphs. Our experimental results not only demonstrate the effectiveness of two proposed modules, but also show BAG achieves state-of-the-art performance on the WIKIHOP dataset.

Our future work will be making use of more complex relations between entities and building graphs in more general way without candidates.

Acknowledgements

This work was supported by Australian Research Council Projects under grants FL-170100117, DP-280103424.

---

[2] The paper was written on early Dec. 2018, during that time Entity-GCN is the best public model, and only one anonymous model is better than it.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. *arXiv preprint arXiv:1804.05922*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *CoRR, abs/1703.04816*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Vector of Locally-Aggregated Word Embeddings (VLAWE): A Novel Document-level Representation

Radu Tudor Ionescu and **Andrei M. Butnaru**
University of Bucharest
Department of Computer Science
14 Academiei, Bucharest, Romania
raducu.ionescu@gmail.com
butnaruandreimadalin@gmail.com

Abstract

In this paper, we propose a novel representation for text documents based on aggregating word embedding vectors into document embeddings. Our approach is inspired by the Vector of Locally-Aggregated Descriptors used for image representation, and it works as follows. First, the word embeddings gathered from a collection of documents are clustered by k-means in order to learn a codebook of semnatically-related word embeddings. Each word embedding is then associated to its nearest cluster centroid (codeword). The Vector of Locally-Aggregated Word Embeddings (VLAWE) representation of a document is then computed by accumulating the differences between each codeword vector and each word vector (from the document) associated to the respective codeword. We plug the VLAWE representation, which is learned in an unsupervised manner, into a classifier and show that it is useful for a diverse set of text classification tasks. We compare our approach with a broad range of recent state-of-the-art methods, demonstrating the effectiveness of our approach. Furthermore, we obtain a considerable improvement on the Movie Review data set, reporting an accuracy of 93.3%, which represents an absolute gain of 10% over the state-of-the-art approach. Our code is available at https://github.com/raduionescu/vlawe-boswe/.

1 Introduction

In recent years, word embeddings (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014) have had a huge impact in natural language processing (NLP) and related fields, being used in many tasks including sentiment analysis (Dos Santos and Gatti, 2014; Fu et al., 2018), information retrieval (Clinchant and Perronnin, 2013; Ye et al., 2016) and word sense disambiguation (Bhingardive et al., 2015; Butnaru et al., 2017; Chen et al., 2014; Iacobacci et al., 2016), among many others. Starting from word embeddings, researchers proposed various ways of aggregating word embedding vectors to obtain efficient sentence-level or document-level representations (Butnaru and Ionescu, 2017; Cheng et al., 2018; Clinchant and Perronnin, 2013; Conneau et al., 2017; Cozma et al., 2018; Fu et al., 2018; Hill et al., 2016; Kiros et al., 2015; Kusner et al., 2015; Le and Mikolov, 2014; Shen et al., 2018; Torki, 2018; Zhao et al., 2015; Zhou et al., 2016, 2018). Although the mean (or sum) of word vectors is commonly adopted because of its simplicity (Mitchell and Lapata, 2010), it seems that more complex approaches usually yield better performance (Cheng et al., 2018; Conneau et al., 2017; Cozma et al., 2018; Fu et al., 2018; Hill et al., 2016; Kiros et al., 2015; Torki, 2018; Zhao et al., 2015; Zhou et al., 2016, 2018). To this end, we propose a simple yet effective approach for aggregating word embeddings into document embeddings. Our approach is inspired by the Vector of Locally-Aggregated Descriptors (VLAD) (Jégou et al., 2010, 2012) used in computer vision to efficiently represent images for various image classification and retrieval tasks. To our knowledge, we are the first to adapt and use VLAD in the text domain.

Our document-level representation is constructed as follows. First, we apply a pre-trained word embedding model, such as *GloVe* (Pennington et al., 2014), on all the words from a set of training documents in order to obtain a set of training word vectors. The word vectors are clustered by k-means in order to learn a codebook of semnatically-related word embeddings. Each word embedding is then associated to its nearest cluster centroid (codeword). The Vector of Locally-Aggregated Word Embeddings (VLAWE) representation of a text document is then com-

Proceedings of NAACL-HLT 2019, pages 363–369
Minneapolis, Minnesota, June 2 - June 7, 2019.

puted by accumulating the differences between each codeword vector and each word vector that is both present in the document and associated to the respective codeword. Since our approach considers cluster centroids as reference for building the representation, it can easily accommodate new words, not seen during k-means training, simply by associating them to the nearest cluster centroids. Thus, VLAWE is robust to vocabulary distribution gaps between training and test, which can appear when the training set is particularly smaller or from a different domain. Certainly, the robustness holds as long as the word embeddings are pretrained on a very large set of documents, e.g. the entire Wikipedia.

We plug the VLAWE representation, which is learned in an unsupervised manner, into a classifier, namely Support Vector Machines (SVM), and show that it is useful for a diverse set of text classification tasks. We consider five benchmark data sets: Reuters-21578 (Lewis, 1997), RT-2k (Pang and Lee, 2004), MR (Pang and Lee, 2005), TREC (Li and Roth, 2002) and Subj (Pang and Lee, 2004). We compare VLAWE with recent state-of-the-art methods (Butnaru and Ionescu, 2017; Cheng et al., 2018; Fu et al., 2018; Hill et al., 2016; Iyyer et al., 2015; Kim, 2014; Kiros et al., 2015; Le and Mikolov, 2014; Liu et al., 2017; Shen et al., 2018; Torki, 2018; Xue and Zhou, 2009; Zhao et al., 2015; Zhou et al., 2016, 2018), demonstrating the effectiveness of our approach. Furthermore, we obtain a considerable improvement on the Movie Review (MR) data set, surpassing the state-of-the-art approach of Cheng et al. (2018) by almost 10%.

The rest of the paper is organized as follows. We present related works on learning document-level representations in Section 2. We describe the Vector of Locally-Aggregated Word Embeddings in Section 3. We present experiments and results on various text classification tasks in Section 4. Finally, we draw our conclusion in Section 5.

2 Related Work

There are various works (Butnaru and Ionescu, 2017; Cheng et al., 2018; Conneau et al., 2017; Fu et al., 2018; Hill et al., 2016; Iyyer et al., 2015; Kim, 2014; Kiros et al., 2015; Kusner et al., 2015; Le and Mikolov, 2014; Clinchant and Perronnin, 2013; Shen et al., 2018; Torki, 2018; Zhao et al., 2015; Zhou et al., 2018) that propose to build effective sentence-level or document-level representations based on word embeddings. While most of these approaches are based on deep learning (Cheng et al., 2018; Conneau et al., 2017; Hill et al., 2016; Iyyer et al., 2015; Kim, 2014; Kiros et al., 2015; Le and Mikolov, 2014; Zhao et al., 2015; Zhou et al., 2018), there have been some approaches that are inspired by computer vision research, namely by the bag-of-visual-words (Butnaru and Ionescu, 2017) and by Fisher Vectors (Clinchant and Perronnin, 2013). The relationship between the bag-of-visual-words, Fisher Vectors and VLAD is discussed in (Jégou et al., 2012). The discussion can be transferred to describe the relantionship of our work and the closely-related works of Butnaru and Ionescu (2017) and Clinchant and Perronnin (2013).

3 Method

The Vector of Locally-Aggregated Descriptors (VLAD) (Jégou et al., 2010, 2012) was introduced in computer vision to efficiently represent images for various image classification and retrieval tasks. We propose to adapt the VLAD representation in order to represent text documents instead of images. Our adaptation consists of replacing the Scale-Invariant Feature Transform (SIFT) image descriptors (Lowe, 2004) useful for recognizing object patterns in images with word embeddings (Mikolov et al., 2013; Pennington et al., 2014) useful for recognizing semantic patterns in text documents. We coin the term *Vector of Locally-Aggregated Word Embeddings (VLAWE)* for the resulting document representation.

The VLAWE representation is derived as follows. First, each word in the collection of training documents is represented as a word vector using a pre-trained word embeddings model. The result is a set $X = \{x_1, x_2, ..., x_n\}$ of n word vectors. As for the VLAD model, the next step is to learn a codebook $\{\mu_1, \mu_2, ..., \mu_k\}$ of representative meta-word vectors (codewords) using k-means. Each codeword μ_i is the centroid of the cluster $C_i \subset X$:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_t \in C_i} x_t, \forall i \in \{1, 2, ..., k\}, \qquad (1)$$

where $|C_i|$ is the number of word vectors assigned to cluster C_i and k is the number of clusters. Since word embeddings carry semantic information by projecting semantically-related words in the same region of the embedding space, it means that the resulting clusters contain semantically-related words. The formed centroids are stored in a randomized forest of k-d trees to reduce search cost, as described in (Philbin et al., 2007; Ionescu et al., 2013; Ionescu and Popescu, 2014, 2015a). Each word embedding x_t is associated to a single cluster C_i, such that the Euclidean distance between x_t and the corresponding codeword μ_i is minimum, for all $i \in \{1, 2, ..., k\}$. For each document D and each codeword μ_i, the differences $x_t - \mu_i$ of the vectors $x_t \in C_i \cap D$ and the codeword μ_i are accumulated into column vectors:

$$v_{i,D} = \sum_{x_t \in C_i \cap D} x_t - \mu_i, \qquad (2)$$

where $D \subset X$ is the set of word embeddings in a given text document. The final VLAWE embedding for a given document D is obtained by stacking together the d-dimensional residual vectors $v_{i,D}$, where d is equal to the dimension of the word embeddings:

$$\phi_D = \begin{bmatrix} v_{1,D} \\ v_{2,D} \\ \vdots \\ v_{k,D} \end{bmatrix}. \qquad (3)$$

Therefore, the VLAWE document embedding is has $k \cdot d$ components.

The VLAWE vector ϕ_D undergoes two normalization steps. First, a power normalization is performed by applying the following operator independently on each component (element):

$$f(z) = sign(z) \cdot |z|^{\alpha}, \qquad (4)$$

where $0 \leq \alpha \leq 1$ and $|z|$ is the absolute value of z. Since words in natural language follow the Zipf's law (Powers, 1998), it seems natural to apply the power normalization in order to reduce the influence of highly frequent words, e.g. common words or stopwords, which can corrupt the representation. As Jégou et al. (2012), we empirically observed that this step consistently improves the quality of the representation. The power normalized document embeddings are then L_2-normalized. After obtaining the normalized VLAWE representations, we employ a classification method to learn a discriminative model for each specific text classification task.

4 Experiments

4.1 Data Sets

We exhibit the performance of VLAWE on five public data sets: Reuters-21578 (Lewis, 1997), RT-2k (Pang and Lee, 2004), MR (Pang and Lee, 2005), TREC (Li and Roth, 2002) and Subj (Pang and Lee, 2004).

The Reuters-21578 data set (Lewis, 1997) contains articles collected from Reuters newswire. Following Joachims (1998) and Yang and Liu (1999), we select the categories (topics) that have at least one document in the training set and one in the test set, leading to a total of 90 categories. We use the ModeApte evaluation (Xue and Zhou, 2009), in which unlabeled documents are eliminated, leaving a total of 10787 documents. The collection is already divided into 7768 documents for training and 3019 documents for testing.

The RT-2k data set (Pang and Lee, 2004) consists of 2000 movie reviews taken from the IMDB movie review archives. There are 1000 positive reviews rated with four or five stars, and 1000 negative reviews rated with one or two stars. The task is to discriminate between positive and negative reviews.

The Movie Review (MR) data set (Pang and Lee, 2005) consists of 5331 positive and 5331 negative sentences. Each sentence is selected from one movie review. The task is to discriminate between positive and negative sentiment.

TREC (Li and Roth, 2002) is a question type classification data set, where questions are divided into 6 classes. The collection is already divided into 5452 questions for training and 500 questions for testing.

The Subjectivity (Subj) (Pang and Lee, 2004) data set contains 5000 objective and 5000 subjective sentences. The task is to classify a sentence as being either subjective or objective.

4.2 Evaluation and Implementation Details

In the experiments, we used the pre-trained word embeddings computed with the *GloVe* toolkit provided by Pennington et al. (2014). The pre-trained GloVe model contains 300-dimensional vectors for 2.2 million words and phrases. Most of the steps required for building the VLAWE representation, such as the k-means clustering and the randomized forest of k-d trees, are implemented using the VLFeat library (Vedaldi and Fulkerson, 2008). We set the number of clusters (size of the codebook) to $k = 10$, leading to a VLAWE representation of $k \cdot d = 10 \cdot 300 = 3000$ components. Similar to Jégou et al. (2012), we set $\alpha = 0.5$ for the power normalization step in Equation (4), which consistently leads to near-optimal results on all data sets. In the learning stage, we employ the

Method	Reuters-21578	RT-2k	MR	TREC	Subj
Average of word embeddings (baseline)	85.3	84.7	77.4	80.0	89.5
BOW (baseline)	86.5	84.1	77.1	89.3	89.3
TF + FA + CP + SVM (Xue and Zhou, 2009)	87.0	-	-	-	-
Paragraph vectors (Le and Mikolov, 2014)	-	-	74.8	91.8	90.5
CNN (Kim, 2014)	-	83.5	81.5	93.6	93.4
DAN (Iyyer et al., 2015)	-	-	80.1	-	-
Combine-skip (Kiros et al., 2015)	-	-	76.5	92.2	93.6
Combine-skip + NB (Kiros et al., 2015)	-	-	80.4	-	93.6
AdaSent (Zhao et al., 2015)	-	-	83.1	92.4	95.5
SAE + embs. (Hill et al., 2016)	-	-	73.2	80.4	89.8
SDAE + embs. (Hill et al., 2016)	-	-	74.6	78.4	90.8
FastSent + AE (Hill et al., 2016)	-	-	71.8	80.4	88.8
BLSTM (Zhou et al., 2016)	-	-	80.0	93.0	92.1
BLSTM-Att (Zhou et al., 2016)	-	-	81.0	93.8	93.5
BLSTM-2DCNN (Zhou et al., 2016)	-	-	82.3	96.1	94.0
DC-TreeLSTM (Liu et al., 2017)	-	-	81.7	93.8	93.7
BOSWE (Butnaru and Ionescu, 2017)	87.2	89.7	-	-	-
TreeNet (Cheng et al., 2018)	-	-	79.8	91.6	92.0
TreeNet-GloVe (Cheng et al., 2018)	-	-	83.6	96.1	95.9
BOMV (Fu et al., 2018)	-	90.2	-	-	90.9
SWEM-average (Shen et al., 2018)	-	-	77.6	92.2	92.5
SWEM-concat (Shen et al., 2018)	-	-	78.2	91.8	93.0
COV + Mean (Torki, 2018)	-	-	80.2	90.3	93.1
COV + BOW (Torki, 2018)	-	-	80.7	91.8	93.3
COV + Mean + BOW (Torki, 2018)	-	-	81.1	91.6	93.2
DARLM (Zhou et al., 2018)	-	-	83.2	96.0	94.1
VLAWE (ours)	89.3	94.1	93.3	94.2	95.0

Table 1: Performance results (in %) of our approach (VLAWE) versus several state-of-the-art methods (Butnaru and Ionescu, 2017; Cheng et al., 2018; Fu et al., 2018; Hill et al., 2016; Iyyer et al., 2015; Kim, 2014; Kiros et al., 2015; Le and Mikolov, 2014; Liu et al., 2017; Shen et al., 2018; Torki, 2018; Xue and Zhou, 2009; Zhao et al., 2015; Zhou et al., 2016, 2018) on the Reuters-21578, RT-2k, MR, TREC and Subj data sets.

Support Vector Machines (SVM) implementation provided by LibSVM (Chang and Lin, 2011). We set the SVM regularization parameter to $C = 1$ in all our experiments. In the SVM, we use the linear kernel. For optimal results, the VLAWE representation is combined with the BOSWE representation (Butnaru and Ionescu, 2017), which is based on the PQ kernel (Ionescu and Popescu, 2013, 2015b).

We follow the same evaluation procedure as Kiros et al. (2015) and Hill et al. (2016), using 10-fold cross-validation when a train and test split is not pre-defined for a given data set. As evaluation metrics, we employ the micro-averaged F_1 measure for the Reuters-21578 data set and the standard classification accuracy for the RT-2k, the MR, the TREC and the Subj data sets, in order to fairly compare with the related art.

4.3 Results

We compare VLAWE with several state-of-the-art methods (Butnaru and Ionescu, 2017; Cheng et al., 2018; Fu et al., 2018; Hill et al., 2016; Iyyer et al., 2015; Kim, 2014; Kiros et al., 2015; Le and Mikolov, 2014; Liu et al., 2017; Shen et al., 2018; Torki, 2018; Xue and Zhou, 2009; Zhao et al., 2015; Zhou et al., 2016, 2018) as well as two baseline methods, namely the average of word embeddings and the standard bag-of-words (BOW). The corresponding results are presented in Table 1.

First, we notice that our approach outperforms both baselines on all data sets, unlike other related methods (Le and Mikolov, 2014; Hill et al., 2016). In most cases, our improvements over the baselines are higher than 5%. On the Reuters-21578 data set, we surpass the closely-related approach of Butnaru and Ionescu (2017) by around 2%. On

Method	MR
VLAWE ($k = 2$)	93.0
VALWE (PCA)	93.2
VLAWE (full, $k = 10$)	93.3

Table 2: Performance results (in %) of the full VLAWE representation (with $k = 10$) versus two compact versions of VLAWE, obtained either by setting $k = 2$ or by applying PCA.

the RT-2k data set, we surpass the related works of Fu et al. (2018) and Butnaru and Ionescu (2017) by around 4%. To our knowledge, our accuracy of 94.1% on RT-2k (Pang and Lee, 2004) surpasses all previous results reported in literature. On the MR data set, we surpass most related works by more than 10%. To our knowledge, the best accuracy on MR reported in previous literature is 83.6%, and it is obtained by Cheng et al. (2018). We surpass the accuracy of Cheng et al. (2018) by almost 10%, reaching an accuracy of 93.3% using VLAWE. On the TREC data set, we reach the third best performance, after methods such as (Cheng et al., 2018; Zhou et al., 2016, 2018). Our performance on TREC is about 2% lower than the state-of-the-art accuracy of 96.1%. On the Subj data set, we obtain an accuracy of 95.0%. There are two state-of-the-art methods (Cheng et al., 2018; Zhao et al., 2015) reporting better performance on Subj. Compared to the best one of them (Cheng et al., 2018), our accuracy is 1% lower. Overall, we consider that our results are noteworthy.

4.4 Discussion

The k-means clustering algorithm and, on some data sets, the cross-validation procedure can induce accuracy variations due to the random choices involved. We have conducted experiments to determine how large are the accuracy variations. We observed that the accuracy can decrease by up to 1%, which does not bring any significant differences to the results reported in Table 1.

Even for a small number of clusters, e.g. $k = 10$, the VLAWE document representation can grow up to thousands of features, as the number of features is $k \cdot d$, where $d = 300$ is the dimensionality of commonly used word embeddings. However, there are several document-level representations that usually have a dimensionality much smaller than $k \cdot d$. Therefore, it is desirable to obtain a more compact VLAWE representation. We hereby propose two approaches that lead to more compact representations. The first one is simply based on reducing the number of clusters. By setting $k = 2$ for instance, we obtain a 600-dimensional representation. The second one is based on applying Principal Component Analysis (PCA), to reduce the dimension of the feature vectors. Using PCA, we propose to reduce the size of the VLAWE representation to 300 components. In Table 2, the resulting compact representations are compared against the full VLAWE representation on the MR data set. Although the compact VLAWE representations provide slightly lower results compared to the VLAWE representation based on 3000 components, we note that the differences are insignificant. Furthermore, both compact VLAWE representations are far above the state-of-the-art method (Cheng et al., 2018).

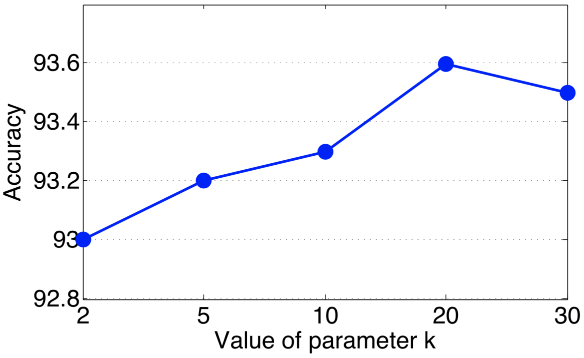


Figure 1: Accuracy on MR for different numbers of k-means clusters.

In Figure 1, we illustrate the performance variation on MR, when using different values for k. We notice that the accuracy tends to increase slightly, as we increase the number of clusters from 2 to 30. Overall, the VLAWE representation seems to be robust to the choice of k, always surpassing the state-of-the-art approach (Cheng et al., 2018).

5 Conclusion

We proposed a novel representation for text documents which is based on aggregating word embeddings using k-means and on computing the residuals between each word embedding allocated to a given cluster and the corresponding cluster centroid. Our experiments on five benchmark data sets prove that our approach yields competitive results with respect to the state-of-the-art methods.

Acknowledgments

We thank the reviewers for their useful comments. This research is supported by University of Bucharest, Faculty of Mathematics and Computer Science, through the 2019 Mobility Fund.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Sudha Bhingardive, Dhirendra Singh, Rudramurthy V, Hanumant Harichandra Redkar, and Pushpak Bhattacharyya. 2015. Unsupervised Most Frequent Sense Detection using Word Embeddings. In *Proceedings of NAACL*, pages 1238–1243.

Andrei Butnaru and Radu Tudor Ionescu. 2017. From Image to Text Classification: A Novel Approach based on Clustering Word Embeddings. In *Proceedings of KES*, pages 1784–1793.

Andrei Butnaru, Radu Tudor Ionescu, and Florentina Hristea. 2017. ShotgunWSD: An unsupervised algorithm for global word sense disambiguation inspired by DNA sequencing. In *Proceedings of EACL*, pages 916–926.

Chih-Chung Chang and Chih-Jen Lin. 2011. LibSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A Unified Model for Word Sense Representation and Disambiguation. In *Proceedings of EMNLP*, pages 1025–1035.

Zhou Cheng, Chun Yuan, Jiancheng Li, and Haiqin Yang. 2018. TreeNet: Learning Sentence Representations with Unconstrained Tree Structure. In *Proceedings of IJCAI*, pages 4005–4011.

Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. In *Proceedings of CVSC Workshop*, pages 100–109.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of ICML*, pages 160–167.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of EMNLP*, pages 670–680.

Mădălina Cozma, Andrei Butnaru, and Radu Tudor Ionescu. 2018. Automated essay scoring with string kernels and word embeddings. In *Proceedings of ACL*, pages 503–509.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING*, pages 69–78.

Mingsheng Fu, Hong Qu, Li Huang, and Li Lu. 2018. Bag of meta-words: A novel method to represent document for the sentiment classification. *Expert Systems with Applications*, 113:33–43.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of NAACL*, pages 1367–1377.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of ACL*, pages 897–907.

Radu Tudor Ionescu and Marius Popescu. 2013. Kernels for Visual Words Histograms. In *Proceedings of ICIAP*, pages 81–90.

Radu Tudor Ionescu and Marius Popescu. 2014. Objectness to improve the bag of visual words model. In *Proceedings of ICIP*, pages 3238–3242.

Radu Tudor Ionescu and Marius Popescu. 2015a. Have a SNAK. Encoding Spatial Information with the Spatial Non-alignment Kernel. In *Proceedings of ICIAP*, pages 97–108.

Radu Tudor Ionescu and Marius Popescu. 2015b. PQ kernel: a rank correlation kernel for visual word histograms. *Pattern Recognition Letters*, 55:51–57.

Radu Tudor Ionescu, Marius Popescu, and Cristian Grozea. 2013. Local Learning to Improve Bag of Visual Words Model for Facial Expression Recognition. In *Proceedings of WREPL*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of ACL*, pages 1681–1691.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating local descriptors into a compact image representation. In *Proceedings of CVPR*, pages 3304–3311.

Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. 2012. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716.

Thorsten Joachims. 1998. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In *Proceedings of ECML*, pages 137–142, London, UK, UK. Springer-Verlag.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Proceedings of NIPS*, pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of ICML*, pages 957–966.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of ICML*, pages 1188–1196.

David Lewis. 1997. The Reuters-21578 text categorization test collection. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*, pages 1–7.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. In *Proceedings of IJCAI*, pages 4054–4060.

David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships For Sentiment Categorization With Respect To Rating Scales. In *Proceedings of ACL*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*, pages 1532–1543.

James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of CVPR*, pages 1–8.

David Powers. 1998. Applications and explanations of Zipf's law. In *Proceedings of NeMLaP/CoNLL*, pages 151–160.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of ACL*, pages 440–450.

Marwan Torki. 2018. A Document Descriptor using Covariance of Word Vectors. In *Proceedings of ACL*, pages 527–532.

Andrea Vedaldi and B. Fulkerson. 2008. VLFeat: An Open and Portable Library of Computer Vision Algorithms. `http://www.vlfeat.org/`.

Xiao-Bing Xue and Zhi-Hua Zhou. 2009. Distributional features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 21(3):428–442.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR*, pages 42–49.

Xin Ye, Hui Shen, Xiao Ma, Răzvan Bunescu, and Chang Liu. 2016. From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of ICSE*, pages 404–415.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-Adaptive Hierarchical Sentence Model. In *Proceedings of IJCAI*, pages 4069–4076.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In *Proceedings of COLING*, pages 3485–3495.

Qianrong Zhou, Xiaojie Wang, and Xuan Dong. 2018. Differentiated attentive representation learning for sentence classification. In *Proceedings of IJCAI*, pages 4630–4636.

Multi-task Learning for Multi-modal Emotion Recognition and Sentiment Analysis

Md Shad Akhtar[†], **Dushyant Singh Chauhan**[†], **Deepanway Ghosal**[†], **Soujanya Poria**[+],
Asif Ekbal[†] **and Pushpak Bhattacharyya**[†]
[†] Department of Computer Science & Engineering
Indian Institute of Technology Patna, India
`{shad.pcs15, 1821CS17, deepanway.ee14, asif, pb}@iitp.ac.in`
[+] School of Computer Science and Engineering, Nanyang Technological University, Singapore
`sporia@ntu.edu.sg`

Abstract

Related tasks often have inter-dependence on each other and perform better when solved in a joint framework. In this paper, we present a deep multi-task learning framework that jointly performs sentiment and emotion analysis both. The multi-modal inputs (i.e., *text*, *acoustic* and *visual frames*) of a video convey diverse and distinctive information, and usually do not have equal contribution in the decision making. We propose a context-level inter-modal attention framework for simultaneously predicting the sentiment and expressed emotions of an utterance. We evaluate our proposed approach on CMU-MOSEI dataset for multi-modal sentiment and emotion analysis. Evaluation results suggest that multi-task learning framework offers improvement over the single-task framework. The proposed approach reports new state-of-the-art performance for both sentiment analysis and emotion analysis.

1 Introduction

With the rapid growth of social media video platforms such as Youtube, Vimeo, users now tend to upload videos on these platforms. Such video platforms offer users an opportunity to express their opinions on any topic. Videos usually consist of *audio* and *visual* modalities, and thus can be considered as a source of multi-modal information. Although videos contain more information than text, fusing multiple modalities is a major challenge. A common practice in sentiment analysis and emotion recognition or affective computing, in general, is to analyze textual opinions. However, in recent days multi-modal affect analysis has gained a major attention (Poria et al., 2017b, 2016). In these works, in addition to the *visual frames*, other sources of information such as *acoustic* and *textual* (transcript) representation of the spoken languages are also incorporated in the analysis. Multi-modal analysis (e.g. sentiment analysis Zadeh et al. 2018c, emotion recognition Poria et al. 2016, question-answering Teney et al. 2017 etc.) is an emerging field of study, that utilizes multiple information sources for solving a problem. These sources (e.g., text, visual, acoustic, etc.) offer a diverse and often distinct piece of information that a system can leverage on. For example, *'text'* carries semantic information of the spoken sentence, whereas '*acoustic*' information reveals the emphasis (pitch, voice quality) on each word. In contrast, the '*visual*' information (image or video frame) extracts the gesture and posture of the speaker.

Traditionally, '*text*' has been the key factor in any Natural Language Processing (NLP) tasks including sentiment and emotion analysis. However, with the recent emergence of social media platforms and their available multi-modal contents, an interdisciplinary study involving *text*, *acoustic* and *visual* features have drawn significant interest among the research community. Effectively fusing this diverse information is non-trivial and poses several challenges to the underlying problem.

In our current work, we propose a multi-task model to extract both sentiment (i.e. ***positive*** or ***negative***) and emotion (i.e. ***anger***, ***disgust***, ***fear***, ***happy***, ***sad*** or ***surprise***) of a speaker in a video. In multi-task framework, we aim to leverage the inter-dependence of these two tasks to increase the confidence of individual task in prediction. For e.g., information about *anger* emotion can help in prediction of *negative* sentiment and vice-versa.

A speaker can utter multiple utterances (a unit of speech bounded by breathes or pauses) in a single video and these utterances can have different sentiments and emotions. We hypothesize that the sentiment (or, emotion) of an utterance often has inter-dependence on other contextual utterances i.e. the knowledge of sentiment (or, emo-

Proceedings of NAACL-HLT 2019, pages 370–379
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

tion) for an utterance can assist in classifying its neighbor utterances. We utilize all three modalities (i.e. *text*, *acoustic* and *visual*) for the analysis. Although all these sources of information are crucial, they are not equally beneficial for each individual instance. Few examples are presented in Table 1. In the first example, *visual frames* provide important clues than *textual* information for finding the sentiment of a sarcastic sentence "*Thanks for putting me on hold! I've all the time in the world.*". Similarly, the *textual* representation of second example "*I'm fine.*' does not reveal the exact emotion of a *sad* person. For this particular case, *acoustic* or *visual* information such as low tone voice, facial expression etc. have bigger role to play for the classification.

Utterance	Feeling	**T**	**A**	**V**
Thanks for putting me on hold! I've all the time in the world.	Sentiment (Negative)	-	-	✓
I'm fine.	Emotion (Sad)	-	✓	✓

Table 1: Contributing modalities for different scenario. Tick represents the most contributing information.

2 Problem Definition

Multi-task learning paradigm provides an efficient platform for achieving generalization. Multiple tasks can exploit the inter-relatedness for improving individual performance through a shared representation. Overall, it provides three basic advantages over the single-task learning paradigm a). it helps in achieving generalization for multiple tasks; b). each task improves its performance in association with the other participating tasks; and c). offers reduced complexity because a single system can handle multiple problems/tasks at the same time.

Sentiments (Pang et al., 2005) and emotions (Ekman, 1999) are closely related. Most of the emotional states have clear distinction of being a positive or negative situation. Emotional states e.g. '*anger*', '*fear*', '*disgust*', '*sad*' etc. belong to negative situations, whereas '*happy*' and '*surprise*' reflect the positive situations. Motivated by the association of sentiment & emotion and the advantages of the multi-task learning paradigm, we present a multi-task framework that jointly learns and classifies the sentiments and emotions in a video. As stated earlier, contextual-utterances and/or multi-modal information provide important cues for the classification. Our proposed approach applies attention over both of these sources of information simultaneously (i.e., contextual utterance and inter-modal information), and aims to reveal the most contributing features for the classification. We hypothesize that applying attention to contributing neighboring utterances and/or multi-modal representations may assist the network to learn in a better way.

Our proposed architecture employs a recurrent neural network based contextual inter-modal attention framework. In our case, unlike the previous approaches, that simply apply attention over the contextual utterance for classification, we take a different approach. Specifically, we attend over the contextual utterances by computing correlations among the modalities of the target utterance and the context utterances. This particularly helps us to distinguish which modalities of the relevant contextual utterances are more important for the classification of the target utterance. The model facilitates this modality selection process by attending over the contextual utterances and thus generates better multi-modal feature representation when these modalities from the context are combined with the modalities of the target utterance. We evaluate our proposed approach on the recent benchmark dataset of CMU-MOSEI (Zadeh et al., 2018c). It is the largest available dataset (approx. 23K utterances) for multi-modal sentiment and emotion analysis (c.f. Dataset Section). The evaluation shows that contextual inter-modal attention framework attains better performance than the state-of-the-art systems for various combinations of input modalities.

The main contributions of our proposed work are three-fold: **a)** *we leverage the interdependence of two related tasks (i.e. sentiment and emotion) in improving each others performance using an effective multi-modal framework*; **b)** *we propose contextual inter-modal attention mechanism that facilitates the model to assign weightage to the contributing contextual utterances and/or to different modalities simultaneously*. Suppose, to classify an utterance '*u1*' of 5 utterances video, visual features of '*u2*' & '*u4*', acoustic features of '*u3*' and textual features of '*u1*', '*u3*' & '*u5*' are more important than others. Our attention model is capable of highlighting such diverse contributing features; and **c)** *we present the state-of-the-arts for both sentiment and emotion predictions*.

3 Related Work

A survey of the literature suggests that multi-modal sentiment prediction is a relatively new area as compared to textual based sentiment prediction (Morency et al., 2011; Poria et al., 2017b; Zadeh et al., 2018a). A good review covering the literature from uni-modal analysis to multi-modal analysis is presented in (Poria et al., 2017a).

Zadeh et al. (2016) introduced the multi-modal dictionary to understand the interaction between facial gestures and spoken words better when expressing sentiment. In another work, Zadeh et al. (2017) proposed a Tensor Fusion Network (TFN) model to learn the intra-modality and inter-modality dynamics of the three modalities (i.e., text, visual and acoustic). Authors reported improved accuracy using multi-modality on the CMU-MOSI dataset. These works did not take contextual information into account. Poria et al. (2017b) proposed a Long Short Term Memory (LSTM) based framework for sentiment classification that leverages the contextual information to capture the inter-dependencies between the utterances. Zadeh et al. (2018a) proposed multi-attention blocks (MAB) to capture the information across the three modalities (text, visual and acoustic) for predicting the sentiments. Authors evaluated their approach on the different datasets and reported improved accuracies in the range of 2-3% over the state-of-the-art models. Blanchard et al. (2018) proposed a multi-modal fusion model that exclusively uses high-level visual and acoustic features for sentiment classification.

An application of multi-kernel learning based fusion technique was proposed in (Poria et al., 2016), where the authors employed deep convolutional neural network (CNN) for extracting the textual features and fused it with other modalities (*visual* & *acoustic*) for emotion prediction. Ranganathan et al. (2016) proposed a convolutional deep belief network (CDBN) models for multi-modal emotion recognition. The author used CDBN to learn salient multi-modal (acoustic and visual) features of low-intensity expressions of emotions. Hazarika et al. (2018) introduced a self- attention mechanism for multi-modal emotion detection by feature level fusion of text and speech. Recently, Zadeh et al. (2018c) introduced the CMU-MOSEI dataset for multi-modal sentiment analysis and emotion recognition. They effectively fused the tri-modal inputs through a dynamic fusion graph and also reported competitive performance *w.r.t.* various state-of-the-arts on MOSEI dataset for both sentiment and emotion classification.

The main difference between the proposed and existing methods is contextual inter-modal attention. Systems (Poria et al., 2016; Zadeh et al., 2016, 2017; Blanchard et al., 2018) do not consider context for the prediction. System (Poria et al., 2017b) uses contextual information for the prediction but without any attention mechanism. In contrast, (Zadeh et al., 2018a) uses multi-attention blocks but did not account for contextual information. Our proposed model is novel in the sense that our approach applies attention over multi-modal information of the contextual utterances in a single step. Thus, it ensures to reveal the contributing features across *multiple modalities* and *contextual utterances* simultaneously for sentiment and emotion analysis. Further, to the best of our knowledge, this is the first attempt at solving the problems of multi-modal sentiment and emotion analysis together in a multi-task framework.

The contextual inter-modal attention mechanism is not much explored in NLP domains as such. We found one work that accounts for bi-modal attention for visual question-answering (VQA) (Teney et al., 2017). However, its attention mechanism differs from our proposed approach in the following manner: a) VQA proposed question guided image-attention, but our attention mechanism attends multi-modalities; b) attention is applied over different positions of the image, whereas our proposed approach applies attention over multiple utterances and two-modalities at a time; c). our proposed attention mechanism attends a sequence of utterances (text, acoustic or visual), whereas VQA applies attention in the spatial domain. In another work, Ghosal et al. (2018) proposed an inter-modal attention framework for the multi-modal sentiment analysis. However, the key differences with our current work are as follows: a) Ghosal et al. (2018) addressed only sentiment analysis, whereas, in our current work, we address both the sentiment and emotion analysis; b) Ghosal et al. (2018) handles only sentiment analysis in single task learning framework, whereas our proposed approach is based on multi-task learning framework, where we solve two tasks, i.e., sentiment analysis and emotion analysis, together in a single network; c) we perform detailed comparative analysis over the single-task vs. multi-task learning; and d) we present state-of-the-art for both sentiment and emotion analysis.

4 Multi-task Multi-modal Emotion Recognition and Sentiment Analysis

In our proposed framework, we aim to leverage multi-modal and contextual information for predicting sentiment and emotion of an utterance simultaneously in a multi-task learning framework. As stated earlier, a video consists of a sequence of utterances and their semantics often have inter-dependencies on each other. We employ three bi-directional Gated Recurrent Unit (biGRU) network for capturing the contextual information (i.e., one for each modality). Subsequently, we introduce pair-wise inter-modal attention mechanism (i.e. *visual-text*, *text-acoustic* and *acoustic-visual*) to learn the joint-association between the multiple modalities & utterances. The objective is to emphasize on the contributing features by putting more attention to the respective utterance and neighboring utterances. Motivated by the residual skip connection (He et al., 2016) the outputs of pair-wise attentions along with the representations of individual modalities are concatenated. Finally, the concatenated representation is shared across the two branches of our proposed network- corresponding to two tasks, i.e., sentiment and emotion classification for prediction (one for each task in the multi-task framework). Sentiment classification branch contains a *softmax* layer for final classification (i.e. *positive* & *negative*), whereas for emotion classification we use sigmoid layer. The shared representation will receive gradients of error from both the branches (sentiment & emotion) and accordingly adjust the weights of the models. Thus, the shared representations will not be biased to any particular task, and it will assist the model in achieving generalization for the multiple tasks. Empirical evidences support our hypothesis (c.f. Table 4).

4.1 Contextual Inter-modal (CIM) Attention Framework

Our contextual inter-modal attention framework works on a pair of modalities. At first, we capture the cross-modality information by computing a pair of matching matrices $M_1, M_2 \in \mathbb{R}^{u \times u}$, where '$u$' is the number of utterances in the video. Further, to capture the contextual dependencies, we compute the probability distribution scores ($N_1, N_2 \in \mathbb{R}^{u \times u}$) over each utterance of cross-modality matrices M_1, M_2 using a softmax function. This essentially computes the attention weights for contextual utterances. Subsequently, we apply soft attention over the contextual inter-modal matrices to compute the modalitiy-wise attentive representations ($O_1 \& O_2$). Finally, a multiplicative gating mechanism (Dhingra et al., 2016) ($A_1 \& A_2$) is introduced to attend the important components of multiple modalities and utterances. The concatenated attention matrix of $A_1 \& A_2$ then acts as the output of our contextual inter-modal attention framework. The entire process is repeated for each pair-wise modalities i.e. *text-visual*, *acoustic-visual* and *text-acoustic*. We illustrate and summarize the proposed methodology in Figure 1 and Algorithm 1, respectively.

Algorithm 1 Multi-task Multi-modal Emotion and Sentiment (MTMM-ES)

```
procedure MTMM-ES($t, v, a$)
    $d \leftarrow 100$                                   ▷ GRU dimension
    $T \leftarrow biGRU_T(t, d)$
    $V \leftarrow biGRU_V(v, d)$
    $A \leftarrow biGRU_A(a, d)$
    $Atn_{TV} \leftarrow CIM\text{-}Attention(T, V)$
    $Atn_{AV} \leftarrow CIM\text{-}Attention(A, V)$
    $Atn_{TA} \leftarrow CIM\text{-}Attention(T, A)$
    $Rep \leftarrow [Atn_{TV}, Atn_{AV}, Atn_{TA}, T, V, A]$
    $polarity \leftarrow Sentiment(Rep)$
    $emotion \leftarrow Emotion(Rep)$
    return $polarity, emotion$

procedure CIM-ATTENTION($X, Y$)
    /*Cross-modality information*/
    $M_1 \leftarrow X.Y^T$
    $M_2 \leftarrow Y.X^T$
    /*Contextual Inter-modal attention*/
    for $i, j \in 1, ..., u$ do                          ▷ $u = \#utterances$
        $N_1(i,j) \leftarrow \frac{e^{M_1(i,j)}}{\sum_{k=1}^{u} e^{M_1(i,k)}}$
        $N_2(i,j) \leftarrow \frac{e^{M_2(i,j)}}{\sum_{k=1}^{u} e^{M_2(i,k)}}$
    $O_1 \leftarrow N_1.Y$
    $O_2 \leftarrow N_2.X$
    /*Multiplicative gating*/
    $A_1 \leftarrow O_1 \odot X$                          ▷ Element-wise mult.
    $A_2 \leftarrow O_2 \odot Y$
    return $[A_1, A_2]$
```

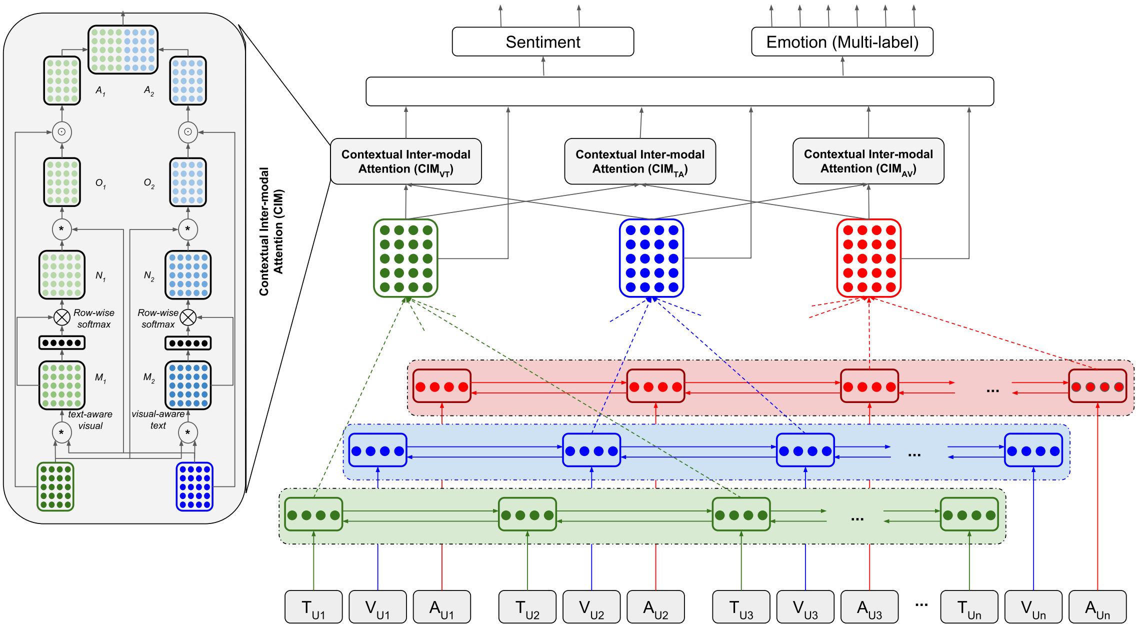


Figure 1: Overall architecture of the proposed framework. Contextual inter-modal (CIM) attention computation between *visual* and *text* modality.

5 Datasets, Experiments, and Analysis

In this section, we describe the datasets used for our experiments and report the results along with necessary analysis.

5.1 Datasets

We evaluate our proposed approach on the benchmark datasets of sentiment and emotion analysis, namely CMU Multi-modal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) dataset (Zadeh et al., 2018c). CMU-MOSEI dataset consists of 3,229 videos spanning over 23,000 utterances from more than 1,000 online YouTube speakers. The training, validation & test set comprises of 16216, 1835 & 4625 utterances, respectively.

Each utterance has six emotion values associated with it, representing the degree of emotion for *anger*, *disgust*, *fear*, *happy*, *sad* and *surprise*. Emotion labels for an utterance are identified as all non-zero intensity values, i.e. if an utterance has three emotions with non-zero values, we take all three emotions as multi-labels. Further, an utterance that has no emotion label represents the absence of emotion. For experiments, we adopt 7-classes (6 *emotions* + 1 *no emotion*) and pose it as multi-label classification problem, where we try to optimize the binary-cross entropy for each of the class. A brief statistics for multi-label emotions is presented in Table 3. In contrast, the sentiment values for each utterance are disjoint, i.e. *value* < 0 and *value* ≥ 0 represent the negative and positive sentiments, respectively. A detailed statistics of the CMU-MOSEI dataset is shown in Table 2.

Statistics	Train	Dev	Test
#Videos	2250	300	679
#Utterance	16216	1835	4625
#Positive	11499	1333	3281
#Negative	4717	502	1344
#Anger	3506	334	1063
#Disgust	2946	280	802
#Fear	1306	163	381
#Happy	8673	978	2484
#Sad	4233	511	1112
#Surprise	1631	194	437
#Speakers		1000	

Table 2: Dataset statistics for CMU-MOSEI. Each utterance contains multi-modal information.

Single emotion	11050	Two emotions	5526
Three emotions	2084	Four emotions	553
Five emotions	84	Six emotions	8
No emotion	3372		

Table 3: Statistics of multi-label emotions.

Tasks		F1-score							Acc (Sentiment) & Weighted-Acc (Emotion)						
		T	A	V	T+V	T+A	A+V	T+A+V	T	A	V	T+V	T+A	A+V	T+A+V
Sent	STL	75.1	67.9	66.3	77.0	76.5	69.6	77.6	78.2	74.8	75.8	79.4	79.7	76.6	79.8
	MTL	77.5	72.1	69.1	78.7	78.6	75.8	78.8	79.7	75.7	76.5	80.4	80.2	77.4	80.5
Emo	STL	75.9	72.3	73.6	77.5	76.8	76.0	77.7	58.0	56.7	53.7	60.1	59.6	58.0	60.8
	MTL	76.9	74.6	75.4	78.5	77.6	77.0	78.6	60.2	56.2	57.5	62.5	60.5	59.3	62.8

Table 4: Single-task learning (STL) and Multi-task (MTL) learning frameworks for the proposed approach. T: Text, V: Visual, A: *Acoustic*. Weighted accuracy as a metric is chosen due to unbalanced samples across various emotions and it is also in line with the other existing works (Zadeh et al., 2018c).

5.2 Feature extraction

We use the CMU-Multi-modal Data SDK[1] for downloading and feature extraction. The dataset was pre-tokenized and a feature vector was provided for each word in an utterance. The *textual*, *visual* and *acoustic* features were extracted by *GloVe* (Pennington et al., 2014), *Facets*[2] & *CovaRep* (Degottex et al., 2014), respectively. Thereafter, we compute the average of *word-level* features to obtain the *utterance-level* features.

Parameters	Values
Bi-GRU	2×200 *neurons*, *dropout=0.3*
Dense layer	100 *neurons*, *dropout=0.3*
Activations	*ReLu*
Optimizer	*Adam* (*lr=0.001*)
Output	*Softmax* (Sent) & *Sigmoid* (Emo)
Loss	*Categorical cross-entropy* (Sent) *Binary cross-entropy* (Emo)
Threshold	0.4 (F1) & 0.2 (W-Acc) for multi-label
Batch	*16*
Epochs	*50*

Table 5: Model configurations

5.3 Experiments

We evaluate our proposed approach on the datasets of CMU-MOSEI. We use the Python based Keras library for the implementation. We compute *F1-score* and *accuracy values* for sentiment classification and *F1-score* and *weighted accuracy* (Tong et al., 2017) for emotion classification. Weighted accuracy as a metric is chosen due to unbalanced samples across various emotions and it is also in line with the other existing works (Zadeh et al., 2018c). To obtain multi-labels for emotion classification, we use 7 sigmoid neurons (corresponds to 6 emotions + 1 no-emotion) with binary cross-entropy loss function. Finally, we take all the emotions whose respective values are above a $threshold$. We optimize and cross-validate both the evaluation metrics (i.e. F1- score and weighted accuracy) and set the *threshold* as 0.4 & 0.2 for F1-score and weighted accuracy, respectively. We show our model configurations in Table 5.

As stated earlier, our proposed approach requires at least two modalities to compute bi-modal attention. Hence, we experiment with bi-modal and tri-modal input combinations for the proposed approach i.e. taking *text-visual*, *text-acoustic*, *acoustic-visual* and *text-visual-acoustic* at a time. For completeness (i.e., uni-modal inputs), we also experiment with a variant of the proposed approach where we apply self-attention on the utterances of each modality separately. The self-attention unit utilizes the contextual information of the utterances (i.e., it receives $u \times d$ hidden representations), applies attention and forward it to the output layer for classification. We report the experimental results of both single-task (STL) and multi-task (MTL) learning framework in Table 4. In the single-task framework, we build separate systems for sentiment and emotion analysis, whereas in multi-task framework a joint-model is learned for both of these problems. For sentiment classification, our single-task framework reports an F1-score of 77.67% and accuracy value of 79.8% for the tri-modal inputs. Similarly, we obtain 77.71% F1-score and 60.88% weighted accuracy for emotion classification.

Comparatively, when both the problems are learned and evaluated in a multi-task learning framework, we observe performance enhancement for both sentiments as well as emotion classification. MTL reports 78.86% F1-score and 80.47% accuracy value in comparison to 77.67% and 79.8% of STL with tri-modal inputs, respectively. For emotion classification, we also observe an improved F-score (78.6 (MTL) vs. 77.7 (STL)) and weighted accuracy (62.8 (MTL) vs. 60.8 (STL))

---

[1] `https://github.com/A2Zadeh/CMU-MultimodalDataSDK`

[2] `https://pair-code.github.io/facets/`

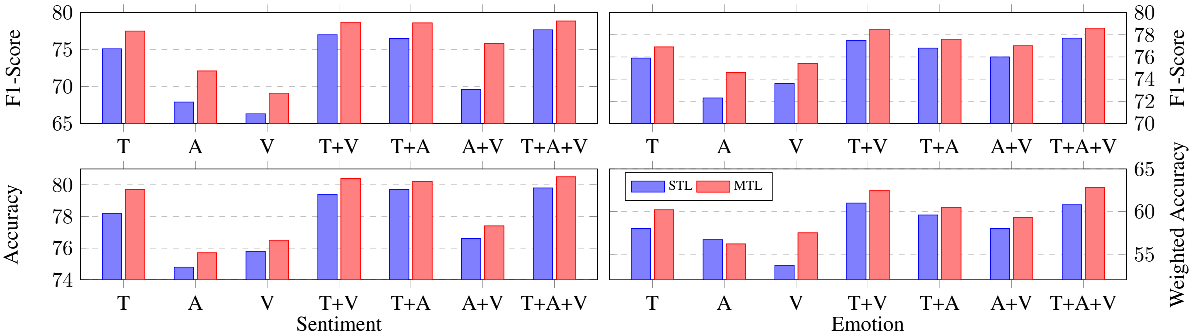
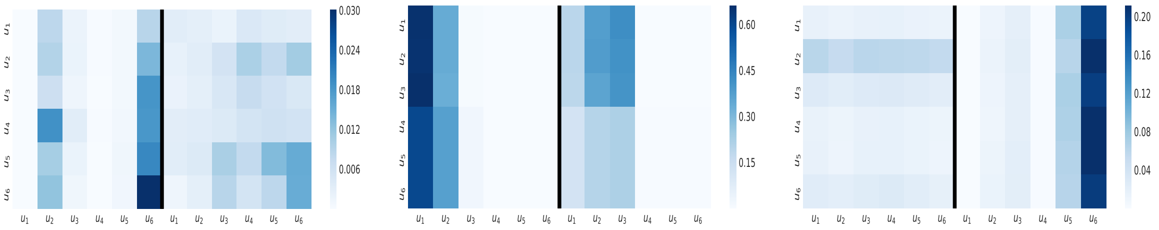


Figure 2: Single-task learning (STL) and Multi-task (MTL) learning frameworks for the proposed approach.

		Sentiment		Emotion	
	Utterances	**Actual**	**MTL**	**Actual**	**MTL**
1	*line hello my name is sarah and i will be doing my video opinion on the movie shall we dance uhh starring jennifer*	Pos	Pos	Anger	Anger
2	*richard gere and susan umm you i really didn't enjoy this movie at all it kinda boring for*	Neg	Neg	Anger, Disgust	Anger, Disgust, Happy, Sad
3	*for umm it kinda felt as if there were parts in there they*	Pos	Neg	No class	Anger, Disgust, Happy, Sad
4	*they just put in there to kinda pass the time on basically the movie is about umm richard character and him being a*	Pos	Pos	Happy	Anger, Disgust, Happy, Sad
5	*umm looking for some some stutter extra sizzle to add into his life he meets up with a dance instructor who is played by jennifer lopez and basically she convinces him to sign up for some ballroom he gets into it he enjoys it a lot but still a secret from his family*	Pos	Pos	Anger	Anger Disgust, Happy, Sad
6	*family he is trying to cope with having this this stutter*	Neg	Neg	Anger, Disgust	Anger, Disgust, Happy, Sad

Table 6: Example video for heatmap analysis of the contextual inter-modal (CIM) attention mechanism of the proposed MTMM-ES framework. Figure 3 depicts the heatmaps for the above video.



(a) *Softmax attention weights N_1 & N_2 for TV.* (b) *Softmax attention weights N_1 & N_2 for AV.* (c) *Softmax attention weights N_1 & N_2 for TA.*

Figure 3: (a), (b) & (c): Pair-wise softmax attention weights N_1 & N_2 of *visual-text*, *acoustic-visual* & *text-acoustic* for multi-task learning framework. Solid line at the center represents boundary of N_1 & N_2. The heatmaps represent attention weights of a particular utterance with respect to other utterances in N_1 & N_2. Each cell (i, j) of the heatmap signifies the weights of utterance 'j' for the classification of utterance 'i' of the pair-wise modality matrices, hence, assists in predicting the labels concisely by incorporating contextual inter-modal information.

in the multi-task framework. It is evident from Figure 2 that multi-task learning framework successfully leverages the inter-dependence of both the tasks in improving the overall performance in comparison to single-task learning. The improvements of MTL over STL framework is also statistically significant with *p*-value < 0.05 (c.f. Table 7).

We also present attention heatmaps of the multi-task learning framework in Figure 3. For illustration, we take the video of the first utterance of Table 6. It has total six utterances. We depict three pair-wise attention matrices of $2 \times (6 \times 6)$ dimension-one each for *text-visual*, *text-acoustics* and *acoustics-visual*. Solid lines in between represent the boundary of the two modalities, e.g. left

	Emotion														***Sentiment***	
	Anger		***Disgust***		***Fear***		***Happy***		***Sad***		***Surprise***		***Average***†			
System	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*W-Acc*	*F1*	*Acc*
Blanchard et al. (2018)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*63.2*	*60.0*
Zadeh et al. (2018b)*	-	-	71.4	65.2	**89.9**	-	-	-	60.8	-	85.4	53.3	-	-	76.0	76.0
Nojavanasghari et al. (2016)*	71.4	-	-	67.0	-	-	-	-	-	-	-	-	-	-	-	-
Rajagopalan et al. (2016)*	-	56.0	-	-	-	-	-	-	-	-	-	-	-	-	76.4	76.4
EF-LSTM (Zadeh et al., 2018c)*	-	-	-	-	-	56.7	-	57.8	-	59.2	-	-	-	-	-	-
TFN (Zadeh et al., 2017)*	-	60.5	-	-	-	-	66.6	66.5	-	58.9	-	52.2	-	-	-	-
Random Forest (Breiman, 2001)*	72.0	-	73.2	-	**89.9**	-	-	-	61.8	-	85.4	-	-	-	-	-
SVM (Zadeh et al., 2016)*	-	-	-	-	-	60.0	-	-	-	-	-	-	-	-	-	-
Zadeh et al. (2018a)*	-	-	-	-	-	-	**71.0**	-	-	-	-	-	-	-	-	-
Zadeh et al. (2018c)	72.8	62.6	76.6	69.1	**89.9**	62.0	66.3	66.3	66.9	60.4	85.5	53.7	76.3	62.3	77.0	76.9
Proposed (Single-task learning)	75.6	64.5	81.0	72.2	87.7	51.5	59.3	**61.6**	67.3	**65.4**	**86.5**	53.0	76.2	61.3	77.6	79.8
Proposed (Multi-task learning)	**75.9**	**66.8**	**81.9**	**72.7**	87.9	**62.2**	67.0	53.6	**72.4**	61.4	86.0	**60.6**	**78.6**	**62.8**	**78.8**	**80.5**
Significance T-test *w.r.t.* SOTA	-	-	-	-	-	-	-	-	-	-	-	-	***0.0240***	***0.0420***	***0.0012***	***0.0046***
Significance T-test *w.r.t.* STL	-	-	-	-	-	-	-	-	-	-	-	-	***0.0171***	***0.0312***	***0.0015***	***0.0278***

Table 7: Comparative results: Proposed multi-task framework attains better performance as compared to the state-of-the-art (SOTA) systems in both the tasks i.e. emotion recognition (average) and sentiment analysis. *Values are taken from Zadeh et al. (2018c). †Six-class average. Significance T-test (< 0.05). STL: Single-task learning.

side of Figure 3a represents *text* modality and right side represents the *visual* modality. The heatmaps represent the contributing features for the classification of utterances. Each cell (*i,j*) of Figure 3 signifies the weights of utterance '*j*' for the classification of utterance '*i*' of the pair-wise modality matrices. For example, for the classification of utterance '*u4*' in Figure 3a, model puts more focus on the textual features of '*u2*' and '*u6*' than others and more-or-less equal focus on the visual features of all the utterances.

5.4 Comparative Analysis

We compare our proposed approach against various existing systems (Nojavanasghari et al., 2016; Rajagopalan et al., 2016; Zadeh et al., 2017, 2018a,b,c; Blanchard et al., 2018) that made use of the same datasets. A comparative study is shown in Table 7. We report the results of the top three existing systems (as reported in Zadeh et al. 2018c) for each case. In emotion classification, the proposed multi-task learning framework reports the best F1-score of 78.6% as compared to the 76.3% and Weighted Accuracy of 62.8% as compared to the 62.3% of the state-of-the-art. Similarly, for sentiment classification, the state-of-the-art system reports 77.0% F1-score and 76.9% accuracy value in the multi-task framework. In comparison, we obtain the best F1-score and accuracy value of 78.8% and 80.4%, respectively, i.e., an improvement of 1.8% and 3.5% over the state-of-the-art systems.

During analysis, we make an important observation. Small improvements in performance do not reveal the exact improvement in the number of instances. Since there are more than 4.6K test samples, even the improvement by one point reflects that the system improves its predictions for 46 samples.

We also perform test-of-significance (T-test) and observe that the obtained results are statistically significant *w.r.t.* the state-of-the-art and proposed single-task results with p-values< 0.05.

5.5 STL v/s MTL framework

In this section, we present our analysis *w.r.t.* single-task and multi-task frameworks. Table 8 lists a few example cases where the proposed multi-task learning framework shows how it yields better performance for both sentiment and emotion, while the single-task framework finds it non-trivial for the classification. For example, first utterance has gold sentiment label as negative which was misclassified by STL framework. However, the MTL framework improves this by correctly predicting '*positive*'. Similarly, in emotion analysis STL predicts three emotions i.e. *disgust*, *happy* and *sad*, out of which only one emotion (*disgust*) matches the gold emotions of *anger* and *disgust*. In comparison, MTL predicts four emotions (i.e. *anger*, *disgust*, *happy* and *sad*) for the same utterance. The precision (2/4) and recall (2/2) for MTL framework is better than the precision (1/3) and recall (1/2) for the STL framework. These analyses suggest that the MTL framework, indeed, captures better evidences than the STL framework.

In the second example, knowledge of sentiment helps in identifying the correct emotion label in the MTL framework. For the gold sentiment (*positive*) and emotion (*happy* and *sad*) labels, STL correctly classifies one emotion (i.e. *sad*), but fails to predict the other emotion (i.e. *happy*). In addition, it misclassifies another emotion (i.e. *anger*). Since, gold label *happy* corresponds to the *posi-*

	Utterances	**Sentiment**			**Emotion**		
		Actual	**STL**	**MTL**	**Actual**	**STL**	**MTL**
1	*richard gere and susan umm you i really didn't enjoy this movie at all it kinda boring for*	Neg	Pos	Neg	Anger, Disgust	Disgust, Happy, Sad	Anger, Disgust, Happy, Sad
2	*we look forward to cooperating with the new government as it works to make progress on a wide range of issues including further democratic reforms promotion of human rights economic development and national reconciliation*	Pos	Pos	Pos	Happy, Sad	Anger, Sad	Happy, Sad
3	*laughter and applause still there though..*	Pos	Neg	Pos	Happy	Happy, Surprise	Happy
4	*is in love with some other person so you know the story*	Neg	Pos	Neg	Anger, Disgust, Sad	Disgust, Happy, Sad	Anger, Disgust, Sad
5	*i can say unfortunately i don't think it's a serious program*	Neg	Pos	Neg	Disgust, Sad, Surprise	Anger, Happy, Sad	Anger, Disgust, Happy, Sad
6	*the last administration bought into just as much as this one does unfortunately*	Neg	Pos	Neg	Anger, Disgust, Sad	Anger, Happy	Anger, Disgust, Happy, Sad
7	*it's just too great of a risk and it is socially unacceptable*	Neg	Pos	Neg	Anger, Disgust, Happy	Anger, Happy	Anger, Disgust, Happy
8	*had a robot here at hopkins since the year longer than most institutions in this country and around the world we*	Pos	Pos	Pos	Happy	Happy, Sad, No class	Happy
9	*in total we spent hundreds of hours on the ground on site watching these leaders in action*	Pos	Pos	Pos	No class	Happy	No class

Table 8: Comparison with multi-task learning and single-task learning frameworks. Few error cases where multi-task learning framework performs better than the single-task framework. *First utterance*: Improved MTL (*Pre: 0.5, Rec: 1.0*) performance over STL (*Pre: 0.3, Rec: 0.5*). *Second utterance:* Sentiment (i.e. *Pos*) assists in emotion classification (i.e. *Happy*). Red color represents error in classification.

tive scenario and predicted label *anger* is related to negative, knowledge of sentiment is a crucial piece of information. Our MTL framework identifies this relation and leverage the predicted sentiment for the classification of emotion i.e. *positive* sentiment assists in predicting *happy* emotion. This is an example of inter-dependence between the two related tasks and the MTL framework successfully exploits it for the performance improvement.

We also observe that the system puts comparatively more focus on some classes in MTL framework than the STL framework. As an instance, MTL predicts '*anger*' class for 1173 utterances, whereas STL predicts it for 951 utterances (1063 anger utterances in the gold dataset). Further, we observe contrasting behavior for the '*sad*' class, where MTL predicts 1618 utterances as '*sad*' compared to the 2126 utterances of STL. For '*disgust*' and '*happy*' classes, both STL and MTL frameworks predict the approximately equal number of utterances.

Further, we observe that MTL performs poorly for the '*fear*' and '*surprise*' classes, where it could not predict a significant number of utterances. A possible reason would be the under-representation of these instances in the given dataset.

6 Conclusion

In this paper, we have proposed a deep multi-task framework that aims to leverage the inter-dependence of two related tasks, i.e., multi-modal sentiment and emotion analysis. Our proposed approach learns a joint-representation for both the tasks as an application of GRU based inter-modal attention framework. We have evaluated our proposed approach on the recently released benchmark dataset on multi-modal sentiment and emotion analysis (MOSEI). Experimental results suggest that sentiment and emotion assist each other when learned in a multitask framework. We have compared our proposed approach against the various existing systems and observed that multi-task framework attains higher performance for all the cases.

In the future, we would like to explore the other dimensions to our multi-task framework, e.g., Sentiment classification & intensity prediction, Emotion classification & intensity prediction and all the four tasks together.

Acknowledgement

Asif Ekbal acknowledges the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya Ph.D. scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Nathaniel Blanchard, Daniel Moreira, Aparna Bharati, and Walter Scheirer. 2018. Getting the subtext without the text: Scalable multimodal sentiment classification from visual and acoustic modalities. In *Proceedings of Grand Challenge and Workshop on Human Multimodal Language*, pages 1–10, Melbourne, Australia.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

G. Degottex, J. Kane, T. Drugman, T. Raitio, and

S. Scherer. 2014. Covarep - a collaborative voice analysis repository for speech technologies. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 960–964.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Paul Ekman. 1999. *Basic Emotions*. The handbook of cognition and emotion.

Deepanway Ghosal, Md Shad Akhtar, Dushyant Chauhan, Soujanya Poria, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Contextual inter-modal attention for multi-modal sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3454–3466, Brussels, Belgium.

D. Hazarika, S. Gorantla, S. Poria, and R. Zimmermann. 2018. Self-attentive feature-level fusion for multimodal emotion detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval*, pages 196–201.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI-2011*$, pages 169 - -176, Alicante, Spain.$

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 284–288, New York, NY, USA.

Bo Pang, , and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37:98–125.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017b. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 873–883.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrusaitis, and Roland Goecke. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision (ECCV-2016)*, pages 338–353.

Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. 2016. Multimodal emotion recognition using deep learning architectures. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2017. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *CoRR*, abs/1708.02711.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating Human Trafficking with Multimodal Deep Models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1547–1556, Vancouver, Canada.

A Zadeh, PP Liang, S Poria, P Vij, E Cambria, and LP Morency. 2018a. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5642 – 5649, New Orleans, LA, USA.

A. Zadeh, R. Zellers, E. Pincus, and L. P. Morency. 2016. Multimodal Sentiment Intensity Analysis in Videos: Facial Gestures and Verbal Messages. *IEEE Intelligent Systems*, 31(6):82–88.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018b. Memory fusion network for multiview sequential learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA.

Amir Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018c. Multimodal Language Analysis in the Wild: CMU-MOSEI Dataset and Interpretable Dynamic Fusion Graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2236–2246, Melbourne, Australia.

Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence

Chi Sun, Luyao Huang, Xipeng Qiu[*]
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{sunc17,lyhuang18,xpqiu}@fudan.edu.cn

Abstract

Aspect-based sentiment analysis (ABSA), which aims to identify fine-grained opinion polarity towards a specific aspect, is a challenging subtask of sentiment analysis (SA). In this paper, we construct an auxiliary sentence from the aspect and convert ABSA to a sentence-pair classification task, such as question answering (QA) and natural language inference (NLI). We fine-tune the pre-trained model from BERT and achieve new state-of-the-art results on SentiHood and SemEval-2014 Task 4 datasets. The source codes are available at https://github.com/HSLCY/ABSA-BERT-pair.

1 Introduction

Sentiment analysis (SA) is an important task in natural language processing. It solves the computational processing of opinions, emotions, and subjectivity - sentiment is collected, analyzed and summarized. It has received much attention not only in academia but also in industry, providing real-time feedback through online reviews on websites such as Amazon, which can take advantage of customers' opinions on specific products or services. The underlying assumption of this task is that the entire text has an overall polarity.

However, the users' comments may contain different aspects, such as: "This book is a hardcover version, but the price is a bit high." The polarity in 'appearance' is positive, and the polarity regarding 'price' is negative. Aspect-based sentiment analysis (ABSA) (Jo and Oh, 2011; Pontiki et al., 2014, 2015, 2016) aims to identify fine-grained polarity towards a specific aspect. This task allows users to evaluate aggregated sentiments for each aspect of a given product or service and gain a more granular understanding of their quality.

Both SA and ABSA are sentence-level or document-level tasks, but one comment may refer to more than one object, and sentence-level tasks cannot handle sentences with multiple targets. Therefore, Saeidi et al. (2016) introduce the task of targeted aspect-based sentiment analysis (TABSA), which aims to identify fine-grained opinion polarity towards a specific aspect associated with a given target. The task can be divided into two steps: (1) the first step is to determine the aspects associated with each target; (2) the second step is to resolve the polarity of aspects to a given target.

The earliest work on (T)ABSA relied heavily on feature engineering (Wagner et al., 2014; Kiritchenko et al., 2014), and subsequent neural network-based methods (Nguyen and Shirai, 2015; Wang et al., 2016; Tang et al., 2015, 2016; Wang et al., 2017) achieved higher accuracy. Recently, Ma et al. (2018) incorporate useful commonsense knowledge into a deep neural network to further enhance the result of the model. Liu et al. (2018) optimize the memory network and apply it to their model to better capture linguistic structure.

More recently, the pre-trained language models, such as ELMo (Peters et al., 2018), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2018), have shown their effectiveness to alleviate the effort of feature engineering. Especially, BERT has achieved excellent results in QA and NLI. However, there is not much improvement in (T)ABSA task with the direct use of the pretrained BERT model (see Table 3). We think this is due to the inappropriate use of the pre-trained BERT model.

Since the input representation of BERT can represent both a single text sentence and a pair of text sentences, we can convert (T)ABSA into a sentence-pair classification task and fine-tune the

---

[*] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 380–385
Minneapolis, Minnesota, June 2 - June 7, 2019.

pre-trained BERT.

In this paper, we investigate several methods of constructing an auxiliary sentence and transform (T)ABSA into a sentence-pair classification task. We fine-tune the pre-trained model from BERT and achieve new state-of-the-art results on (T)ABSA task. We also conduct a comparative experiment to verify that the classification based on a sentence-pair is better than the single-sentence classification with fine-tuned BERT, which means that the improvement is not only from BERT but also from our method. In particular, our contribution is two-fold:

1. We propose a new solution of (T)ABSA by converting it to a sentence-pair classification task.
2. We fine-tune the pre-trained BERT model and achieve new state-of-the-art results on SentiHood and SemEval-2014 Task 4 datasets.

2 Methodology

In this section, we describe our method in detail.

2.1 Task description

TABSA In TABSA, a sentence s usually consists of a series of words: $\{w_1, \cdots, w_m\}$, and some of the words $\{w_{i_1}, \cdots, w_{i_k}\}$ are pre-identified targets $\{t_1, \cdots, t_k\}$, following Saeidi et al. (2016), we set the task as a 3-class classification problem: given the sentence s, a set of target entities T and a fixed aspect set $A = \{general, price, transit\text{-}location, safety\}$, predict the sentiment polarity $y \in \{positive, negative, none\}$ over the full set of the target-aspect pairs $\{(t, a) : t \in T, a \in A\}$. As we can see in Table 1, the gold standard polarity of (LOCATION2, price) is negative, while the polarity of (LOCATION1, price) is none.

ABSA In ABSA, the target-aspect pairs $\{(t, a)\}$ become only aspects a. This setting is equivalent to learning subtask 3 (Aspect Category Detection) and subtask 4 (Aspect Category Polarity) of SemEval-2014 Task 4[1] at the same time.

2.2 Construction of the auxiliary sentence

For simplicity, we mainly describe our method with TABSA as an example.

We consider the following four methods to convert the TABSA task into a sentence pair classification task:

Example:
LOCATION2 is central London so extremely expensive, LOCATION1 is often considered the coolest area of London.

Target	Aspect	Sentiment
LOC1	general	Positive
LOC1	price	None
LOC1	safety	None
LOC1	transit-location	None
LOC2	general	None
LOC2	price	Negative
LOC2	safety	None
LOC2	transit-location	Positive

Table 1: An example of SentiHood dataset.

Methods	Output	Auxiliary Sentence
QA-M	S.P.	Question w/o S.P.
NLI-M	S.P.	Pseudo w/o S.P.
QA-B	{yes,no}	Question w/ S.P.
NLI-B	{yes,no}	Pseudo w/ S.P.

Table 2: The construction methods. Due to limited space, we use the following abbreviations: *S.P.* for *sentiment polarity*, *w/o* for *without*, and *w/* for *with*.

Sentences for QA-M The sentence we want to generate from the target-aspect pair is a question, and the format needs to be the same. For example, for the set of a target-aspect pair (LOCATION1, safety), the sentence we generate is “what do you think of the safety of location - 1 ?”

Sentences for NLI-M For the NLI task, the conditions we set when generating sentences are less strict, and the form is much simpler. The sentence created at this time is not a standard sentence, but a simple pseudo-sentence, with (LOCATION1, safety) pair as an example: the auxiliary sentence is: “location - 1 - safety”.

Sentences for QA-B For QA-B, we add the label information and temporarily convert TABSA into a binary classification problem ($label \in \{yes, no\}$) to obtain the probability distribution. At this time, each target-aspect pair will generate three sequences such as “the polarity of the aspect safety of location - 1 is positive”, “the polarity of the aspect safety of location - 1 is negative”, “the polarity of the aspect safety of location - 1 is none”. We use the probabil-

---

[1] http://alt.qcri.org/semeval2014/task4/

ity value of yes as the matching score. For a target-aspect pair which generates three sequences ($positive, negative, none$), we take the class of the sequence with the highest matching score for the predicted category.

Sentences for NLI-B The difference between NLI-B and QA-B is that the auxiliary sentence changes from a question to a pseudo-sentence. The auxiliary sentences are: "location - 1 - safety - positive", "location - 1 - safety - negative", and "location - 1 - safety - none".

After we construct the auxiliary sentence, we can transform the TABSA task from a single sentence classification task to a sentence pair classification task. As shown in Table 3, this is a necessary operation that can significantly improve the experimental results of the TABSA task.

2.3 Fine-tuning pre-trained BERT

BERT (Devlin et al., 2018) is a new language representation model, which uses bidirectional transformers to pre-train a large corpus, and fine-tunes the pre-trained model on other tasks. We fine-tune the pre-trained BERT model on TABSA task. Let's take a brief look at the input representation and the fine-tuning procedure.

2.3.1 Input representation

The input representation of the BERT can explicitly represent a pair of text sentences in a sequence of tokens. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. For classification tasks, the first word of each sequence is a unique classification embedding ([CLS]).

2.3.2 Fine-tuning procedure

BERT fine-tuning is straightforward. To obtain a fixed-dimensional pooled representation of the input sequence, we use the final hidden state (i.e., the output of the transformer) of the first token as the input. We denote the vector as $C \in \mathbb{R}^H$. Then we add a classification layer whose parameter matrix is $W \in \mathbb{R}^{K \times H}$, where K is the number of categories. Finally, the probability of each category P is calculated by the softmax function $P = \mathbf{softmax}(CW^{\mathrm{T}})$.

2.3.3 BERT-single and BERT-pair

BERT-single for (T)ABSA BERT for single sentence classification tasks. Suppose the number of target categories are n_t and aspect categories are n_a. We consider TABSA as a combination of $n_t \cdot n_a$ target-aspect-related sentiment classification problems, first classifying each sentiment classification problem, and then summarizing the results obtained. For ABSA, We fine-tune pre-trained BERT model to train n_a classifiers for all aspects and then summarize the results.

BERT-pair for (T)ABSA BERT for sentence pair classification tasks. Based on the auxiliary sentence constructed in Section 2.2, we use the sentence-pair classification approach to solve (T)ABSA. Corresponding to the four ways of constructing sentences, we name the models: BERT-pair-QA-M, BERT-pair-NLI-M, BERT-pair-QA-B, and BERT-pair-NLI-B.

3 Experiments

3.1 Datasets

We evaluate our method on the SentiHood (Saeidi et al., 2016) dataset[2], which consists of 5,215 sentences, 3,862 of which contain a single target, and the remainder multiple targets. Each sentence contains a list of target-aspect pairs $\{t, a\}$ with the sentiment polarity y. Ultimately, given a sentence s and the target t in the sentence, we need to:

(1) detect the mention of an aspect a for the target t;

(2) determine the positive or negative sentiment polarity y for detected target-aspect pairs.

We also evaluate our method on SemEval-2014 Task 4 (Pontiki et al., 2014) dataset[3] for aspect-based sentiment analysis. The only difference from the SentiHood is that the target-aspect pairs $\{t, a\}$ become only aspects a. This setting allows us to jointly evaluate subtask 3 (Aspect Category Detection) and subtask 4 (Aspect Category Polarity).

3.2 Hyperparameters

We use the pre-trained uncased BERT-base model[4] for fine-tuning. The number of Transformer blocks is 12, the hidden layer size is 768, the number of self-attention heads is 12, and the total number of parameters for the pre-trained model is 110M. When fine-tuning, we keep the dropout probability at 0.1, set the number of

---

[2] Dataset mirror: https://github.com/uclmr/jack/tree/master/data/sentihood

[3] http://alt.qcri.org/semeval2014/task4/

[4] https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

Model	Aspect			Sentiment	
	Acc.	F_1	AUC	*Acc.*	AUC
LR (Saeidi et al., 2016)	-	39.3	92.4	87.5	90.5
LSTM-Final (Saeidi et al., 2016)	-	68.9	89.8	82.0	85.4
LSTM-Loc (Saeidi et al., 2016)	-	69.3	89.7	81.9	83.9
LSTM+TA+SA (Ma et al., 2018)	66.4	76.7	-	86.8	-
SenticLSTM (Ma et al., 2018)	67.4	78.2	-	89.3	-
Dmu-Entnet (Liu et al., 2018)	73.5	78.5	94.4	91.0	94.8
BERT-single	73.7	81.0	96.4	85.5	84.2
BERT-pair-QA-M	79.4	86.4	97.0	**93.6**	96.4
BERT-pair-NLI-M	78.3	87.0	**97.5**	92.1	96.5
BERT-pair-QA-B	79.2	**87.9**	97.1	93.3	**97.0**
BERT-pair-NLI-B	**79.8**	87.5	96.6	92.8	96.9

Table 3: Performance on SentiHood dataset. We boldface the score with the best performance across all models. We use the results reported in Saeidi et al. (2016), Ma et al. (2018) and Liu et al. (2018). "-" means not reported.

epochs to 4. The initial learning rate is 2e-5, and the batch size is 24.

3.3 Exp-I: TABSA

We compare our model with the following models:

- LR (Saeidi et al., 2016): a logistic regression classifier with n-gram and pos-tag features.
- LSTM-Final (Saeidi et al., 2016): a biLSTM model with the final state as a representation.
- LSTM-Loc (Saeidi et al., 2016): a biLSTM model with the state associated with the target position as a representation.
- LSTM+TA+SA (Ma et al., 2018): a biLSTM model which introduces complex target-level and sentence-level attention mechanisms.
- SenticLSTM (Ma et al., 2018): an upgraded version of the LSTM+TA+SA model which introduces external information from SenticNet (Cambria et al., 2016).
- Dmu-Entnet (Liu et al., 2018): a bidirectional EntNet (Henaff et al., 2016) with external "memory chains" with a delayed memory update mechanism to track entities.

During the evaluation of SentiHood, following Saeidi et al. (2016), we only consider the four most frequently seen aspects (general, price, transit-location, safety). When evaluating the aspect detection, following Ma et al. (2018), we use strict accuracy and Macro-F1, and we also report AUC. In sentiment classification, we use accuracy and macro-average AUC as the evaluation indices.

3.3.1 Results

Results on SentiHood are presented in Table 3. The results of the BERT-single model on aspect detection are better than Dmu-Entnet, but the accuracy of sentiment classification is much lower than that of both SenticLstm and Dmu-Entnet, with a difference of 3.8 and 5.5 respectively.

However, BERT-pair outperforms other models on aspect detection and sentiment analysis by a substantial margin, obtaining 9.4 macro-average F1 and 2.6 accuracies improvement over Dmu-Entnet. Overall, the performance of the four BERT-pair models is close. It is worth noting that BERT-pair-NLI models perform relatively better on aspect detection, while BERT-pair-QA models perform better on sentiment classification. Also, the BERT-pair-QA-B and BERT-pair-NLI-B models can achieve better AUC values on sentiment classification than the other models.

3.4 Exp-II: ABSA

The benchmarks for SemEval-2014 Task 4 are the two best performing systems in Pontiki et al. (2014) and ATAE-LSTM (Wang et al., 2016). When evaluating SemEval-2014 Task 4 subtask 3 and subtask 4, following Pontiki et al. (2014), we use Micro-F1 and accuracy respectively.

3.4.1 Results

Results on SemEval-2014 are presented in Table 4 and Table 5. We find that BERT-single

Models	P	R	F1
XRCE	83.23	81.37	82.29
NRC-Canada	91.04	86.24	88.58
BERT-single	92.78	89.07	90.89
BERT-pair-QA-M	92.87	90.24	91.54
BERT-pair-NLI-M	93.15	90.24	91.67
BERT-pair-QA-B	93.04	89.95	91.47
BERT-pair-NLI-B	93.57	90.83	**92.18**

Table 4: Test set results for Semeval-2014 task 4 Subtask 3: Aspect Category Detection. We use the results reported in XRCE (Brun et al., 2014) and NRC-Canada (Kiritchenko et al., 2014).

Models	4-way	3-way	Binary
XRCE	78.1	-	-
NRC-Canada	82.9	-	-
LSTM	-	82.0	88.3
ATAE-LSTM	-	84.0	89.9
BERT-single	83.7	86.9	93.3
BERT-pair-QA-M	85.2	89.3	95.4
BERT-pair-NLI-M	85.1	88.7	94.4
BERT-pair-QA-B	**85.9**	**89.9**	**95.6**
BERT-pair-NLI-B	84.6	88.7	95.1

Table 5: Test set accuracy (%) for Semeval-2014 task 4 Subtask 4: Aspect Category Polarity. We use the results reported in XRCE (Brun et al., 2014), NRC-Canada (Kiritchenko et al., 2014) and ATAE-LSTM (Wang et al., 2016). "-" means not reported.

has achieved better results on these two subtasks, and BERT-pair has achieved further improvements over BERT-single. The BERT-pair-NLI-B model achieves the best performance for aspect category detection. For aspect category polarity, BERT-pair-QA-B performs best on all 4-way, 3-way, and binary settings.

4 Discussion

Why is the experimental result of the BERT-pair model so much better? On the one hand, we convert the target and aspect information into an auxiliary sentence, which is equivalent to exponentially expanding the corpus. A sentence s_i in the original data set will be expanded into $(s_i, t_1, a_1), \cdots, (s_i, t_1, a_{n_a}), \cdots, (s_i, t_{n_t}, a_{n_a})$ in the sentence pair classification task. On the other hand, it can be seen from the amazing improvement of the BERT model on the QA and NLI tasks (Devlin et al., 2018) that the BERT model has an advantage in dealing with sentence pair classification tasks. This advantage comes from both unsupervised masked language model and next sentence prediction tasks.

TABSA is more complicated than SA due to additional target and aspect information. Directly fine-tuning the pre-trained BERT on TABSA does not achieve performance growth. However, when we separate the target and the aspect to form an auxiliary sentence and transform the TABSA into a sentence pair classification task, the scenario is similar to QA and NLI, and then the advantage of the pre-trained BERT model can be fully utilized. Our approach is not limited to TABSA, and this construction method can be used for other similar tasks. For ABSA, we can use the same approach to construct the auxiliary sentence with only aspects.

In BERT-pair models, BERT-pair-QA-B and BERT-pair-NLI-B achieve better AUC values on sentiment classification, probably because of the modeling of label information.

5 Conclusion

In this paper, we constructed an auxiliary sentence to transform (T)ABSA from a single sentence classification task to a sentence pair classification task. We fine-tuned the pre-trained BERT model on the sentence pair classification task and obtained the new state-of-the-art results. We compared the experimental results of single sentence classification and sentence pair classification based on BERT fine-tuning, analyzed the advantages of sentence pair classification, and verified the validity of our conversion method. In the future, we will apply this conversion method to other similar tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by Shanghai Municipal Science and Technology Commission (No. 16JC1420401 and 17JC1404100), National Key Research and Development Program of China (No. 2017YFB1002104), and National Natural Science Foundation of China (No. 61672162 and 61751201).

References

Caroline Brun, Diana Nicoleta Popa, and Claude Roux. 2014. Xrce: Hybrid classification for aspect-based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 838–842.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2666–2677.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. *arXiv preprint arXiv:1804.11019*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Proceedings of AAAI*.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2509–2514.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pretraining. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Marzieh Saeidi, Guillaume Bouchard, Maria Liakata, and Sebastian Riedel. 2016. Sentihood: targeted aspect based sentiment analysis dataset for urban neighbourhoods. *arXiv preprint arXiv:1610.03771*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. Dcu: Aspect-based polarity classification for semeval task 4. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 223–229.

Bo Wang, Maria Liakata, Arkaitz Zubiaga, and Rob Procter. 2017. Tdparse: Multi-target-specific sentiment recognition on twitter. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 483–493.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

A Variational Approach to Weakly Supervised Document-Level Multi-Aspect Sentiment Classification

Ziqian Zeng[1], **Wenxuan Zhou**[2], **Xin Liu**[1], **and Yangqiu Song**[1]
[1]Department of CSE, Hong Kong University of Science and Technology, HK
[2]Department of CS, University of Southern California, CA, USA
[1]{zzengae, xliucr, yqsong}@cse.ust.hk
[2]{zhouwenx}@usc.edu

Abstract

In this paper, we propose a variational approach to weakly supervised document-level multi-aspect sentiment classification. Instead of using user-generated ratings or annotations provided by domain experts, we use target-opinion word pairs as "supervision." These word pairs can be extracted by using dependency parsers and simple rules. Our objective is to predict an opinion word given a target word while our ultimate goal is to learn a sentiment polarity classifier to predict the sentiment polarity of each aspect given a document. By introducing a latent variable, i.e., the sentiment polarity, to the objective function, we can inject the sentiment polarity classifier to the objective via the variational lower bound. We can learn a sentiment polarity classifier by optimizing the lower bound. We show that our method can outperform weakly supervised baselines on TripAdvisor and BeerAdvocate datasets and can be comparable to the state-of-the-art supervised method with hundreds of labels per aspect.

1 Introduction

Document-level multi-aspect sentiment classification (DMSC) aims to predict the sentiment polarity of each aspect given a document which consists of several sentences describing one or more aspects (Wang et al., 2010, 2011; Yin et al., 2017). Solving the DMSC task is useful for providing both recommendations for users and suggestions for business owners on customer review platforms.

Aspect based sentiment classification (Tang et al., 2016a,b; Wang et al., 2016b; Chen et al., 2017; Ma et al., 2017; Wang et al., 2018) was usually done by supervised learning, where aspect-level annotations should be provided. Aspect-level annotations are not easy to obtain. Even when the platform provides the function to rate for different aspects, users are less likely to submit all of them. For example, about 37% of the aspect ratings are missing on TripAdvisor. If we can solve DMSC task without using aspect-level annotations, it can save human effort to annotate data or collect user-generated annotations on the platform.

Existing weakly supervised approaches (Wang et al., 2010, 2011) use overall polarities instead of aspect polarities as "supervision." Compared with the polarity of each aspect, it is relatively easy to obtain overall polarities. Specifically, they minimize the square loss between the overall polarity and the weighted sum of all aspect polarities. However, when users only care about a particular rare aspect, e.g., childcare services, these approaches cannot estimate parameters of the rare aspect incrementally. They have to re-collect documents which mentioned this rare aspect and estimate parameters of all aspects based on the new corpus. In addition, these approaches assume the document is a bag-of-words, which neglects the order of the words and fails to capture the similarity between words.

In this paper, we propose to use target-opinion word pairs as "supervision." Target-opinion word pairs can be helpful with our ultimate goal which is to learn a classifier to predict the sentiment polarity of each aspect given a document. For example, in a document "The bedroom is very spacious," if we can extract the target-opinion pair "bedroom-spacious," the sentiment polarity of the aspect *room* is likely to be *positive*. Hence, we propose to achieve the polarity classification goal by accomplishing another relevant objective: to predict an opinion word given a target word.

We can decompose the opinion word prediction objective into two sub-tasks. The first sub-task is to predict the sentiment polarity based on a document. For example, given a document "The bedroom is very spacious," it predicts the sentiment polarity of the aspect *room* to be *positive*. The sec-

Proceedings of NAACL-HLT 2019, pages 386–396
Minneapolis, Minnesota, June 2 - June 7, 2019.

ond sub-task is to predict the opinion word given a target word and a sentiment polarity predicted by the first sub-task. For example, knowing the fact that the sentiment polarity of the aspect *room* is *positive*, it predicts the opinion word associated with the target word "room" to be "spacious." By introducing a latent variable, i.e., the sentiment polarity of an aspect, to the opinion word prediction objective, we can inject the polarity classification goal (the first sub-task) into the objective via the variational lower bound which also incorporates the second sub-task. In this sense, our training objective is only based on the target-opinion word pairs which can be extracted by using dependency parsers and some manually designed rules. We consider our approach as weakly supervised learning because there is no direct supervision from polarity of each aspect.

In other words, our model includes two classifiers: a sentiment polarity classifier and an opinion word classifier. In the sentiment polarity classifier, it predicts the sentiment polarity given a document. In the opinion word classifier, it predicts an opinion word based on a target word and a sentiment polarity. Compared with previous approaches (Wang et al., 2010, 2011), our approach can get rid of the assumption that the overall polarity should be observed and it is a weighted sum of all aspect polarities. Moreover, our approach can estimate parameters of a new aspect incrementally. In addition, our sentiment polarity classifier can be more flexible to capture dependencies among words beyond the bag-of-words representation if we use a deep neural network architecture to extract features to represent a document. We conducted experiments on two datasets, TripAdvisor (Wang et al., 2010) and BeerAdvocate (McAuley et al., 2012), to illustrate the effectiveness of our approach.

Our contributions are summarized as follows,

- We propose to solve DMSC task in a nearly unsupervised way.
- We propose to learn a classifier by injecting it into another relevant objective via the variational lower bound. This framework is flexible to incorporate different kinds of document representations and relevant objectives.
- We show promising results on two real datasets and we can produce comparable results to the supervised method with hundreds of labels per aspect.

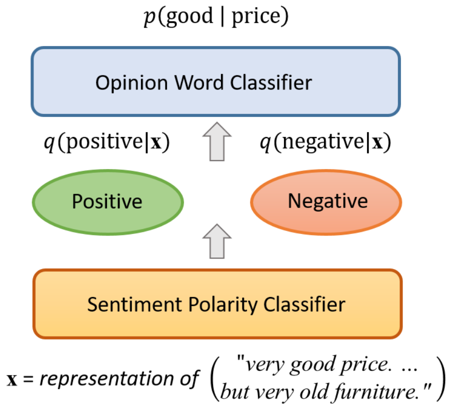


Figure 1: A sentiment polarity classifier and an opinion word classifier associated with the aspect *price*.

Code and data for this paper are available on https://github.com/HKUST-KnowComp/VWS-DMSC.

2 VWS-DMSC Approach

In this section, we describe our variational approach to weakly supervised DMSC (VWS-DMSC). In the next section, we present how we obtain target-opinion word pairs by using a rule-based extraction approach.

2.1 Overview

Our model consists of a sentiment polarity classifier and an opinion word classifier. Our task is document-level multi-aspect sentiment classification. For each aspect, we train a sentiment polarity classifier and an opinion word classifier. The input of the sentiment polarity classifier of each aspect is the same, i.e., a representation of a document. The target-opinion word pairs used in opinion word classifiers are different for different aspects.

Figure 1 shows the relation between two classifiers (on the aspect *price*). The input $\mathbf{x}$ of the sentiment polarity classifier is a representation of a document, e.g., bag-of-words or a representation learned by recurrent neural networks. The sentiment polarity classifier takes $\mathbf{x}$ as input and produces a distribution of sentiment polarity R_a of an aspect a, denoted as $q(R_a|\mathbf{x})$. If R_a only has two possible values, i.e., positive and negative, then outputs of the classifier are $q(positive|\mathbf{x})$ and $q(negative|\mathbf{x})$ respectively. The opinion word classifier takes a target word ("price") and a possible value of the sentiment polarity r_a as input, and estimates $p(\text{“good”}|r_a, \text{“price”})$. Our training objective is to maximize the log-likelihood of an opinion word given a target word, e.g., p(“good”|“price”). The likelihood is estimated based on the sentiment polarity classifier and the opinion word classifier.

2.2 Sentiment Polarity Classifier

The sentiment polarity classifier aims to estimate a distribution of sentiment polarity $q(R_a|\mathbf{x})$, where R_a is a discrete random variable representing the sentiment polarity and $\mathbf{x}$ is a feature representation of a document. We use a simple Softmax classifier here. We denote r_a as a possible value of the random variable R_a, representing a possible sentiment polarity. The model estimates the probability of class r_a as

$$q(R_a = r_a|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_{r_a}^T \mathbf{x}\right)}{\sum_{r'_a} \exp\left(\mathbf{w}_{r'_a}^T \mathbf{x}\right)} , \quad (1)$$

where $\mathbf{w}_{r_a}$ is a vector associated with sentiment class r_a for aspect a.

Document Representation The representation of a document $\mathbf{x}$ can be different using different feature extraction approaches. Traditional document representations of sentiment classification would be bag-of-words, n-gram, or averaged word embeddings. Recently, end-to-end recurrent neural network based models demonstrate a powerful capacity to extract features of a document. The state-of-the-art model in DMSC task is (Yin et al., 2017). We use it as the document representation in our model.

2.3 Opinion Word Classifier

The opinion word classifier aims to estimate the probability of an opinion word w_o given a target word w_t and a sentiment polarity r_a:

$$p(w_o|r_a, w_t) = \frac{\exp\left(\varphi(w_o, w_t, r_a)\right)}{\sum_{w'_o} \exp\left(\varphi(w'_o, w_t, r_a)\right)} , \quad (2)$$

where φ is a scoring function related to opinion word w_o, target word w_t, and sentiment polarity r_a. Here we use the dot product as the scoring function:

$$\varphi(w_o, w_t, r_a) = I\big((w_t, w_o) \in \mathcal{P}, w_t \in \mathcal{K}_a\big) \cdot \mathbf{c}_{r_a}^T \mathbf{w}_o , \quad (3)$$

where $\mathbf{w}_o$ is the word embedding of opinion word w_o, $\mathbf{c}_{r_a}$ is a vector associated with r_a, $\mathcal{P}$ is the set of pairs extracted from the document, $\mathcal{K}_a$ is the set of target words associated with aspect a, and $I(\cdot)$ is an indicator function where $I(true) = 1$ and $I(false) = 0$.

Given a target word w_t and a sentiment polarity r_a, we aim to maximize the probability of opinion words highly related to them. For example, opinion word “good” is usually related to target word “price” for aspect *value* with sentiment polarity *positive*, and opinion word “terrible” is usually related to target word “traffic” for aspect *location* with sentiment polarity *negative*.

2.4 Training Objective

The objective function is to maximize the log-likelihood of an opinion word w_o given a target word w_t. As we mentioned before, the objective function can be decomposed into two sub-tasks. The first one corresponds to the sentiment polarity classifier. The second one corresponds to the opinion word classifier. After introducing a latent variable, i.e., the sentiment polarity, to the objective function, we can derive a variational lower bound of the log-likelihood which can incorporate two classifiers:

$$\begin{aligned}
\mathcal{L} &= \log p(w_o|w_t) \\
&= \log \sum_{r_a} p(w_o, r_a|w_t) \\
&= \log \sum_{r_a} q(r_a|\mathbf{x}) \left[\frac{p(w_o, r_a|w_t)}{q(r_a|\mathbf{x})}\right] \\
&\geq \sum_{r_a} q(r_a|\mathbf{x}) \left[\log \frac{p(w_o, r_a|w_t)}{q(r_a|\mathbf{x})}\right] \\
&= \mathbb{E}_{q(R_a|\mathbf{x})} \left[\log p(w_o|r_a, w_t) p(r_a|w_t)\right] \\
&\quad + H(q(R_a|\mathbf{x})) \\
&= \mathbb{E}_{q(R_a|\mathbf{x})} \left[\log p(w_o|r_a, w_t) p(r_a)\right] \\
&\quad + H(q(R_a|\mathbf{x})) , \qquad (4)
\end{aligned}$$

where $H(\cdot)$ refers to the Shannon entropy. By applying Jensen’s inequality, the log-likelihood is lower-bounded by Eq. (4). The equality holds if and only if the KL-divergence of two distributions, $q(R_a|\mathbf{x})$ and $p(R_a|w_t, w_o)$, equals to zero. Maximizing the variational lower bound is equivalent to minimizing the KL-divergence. Hence, we can learn a sentiment polarity classifier which can produce a similar distribution to the true posterior $p(R_a|w_t, w_o)$. Compared with $p(R_a|w_t, w_o)$, $q(R_a|\mathbf{x})$ is more flexible since it can take any kind of feature representations as input. We assume that a target word w_t and a sentiment polarity r_a are independent since the polarity assignment is not influenced by the target word. We also assume that the sentiment polarity R_a follows a uniform distribution, which means $p(r_a)$ is a constant. We remove it in Eq. (4) to get a new objective function as follows:

$$\mathbb{E}_{q(R_a|\mathbf{x})}\left[\log p(w_o|r_a, w_t)\right] + H(q(R_a|\mathbf{x})) \,. \quad (5)$$

2.4.1 Approximation

The partition function of Eq. (2) requires the summation over all opinion words in the vocabulary. However, the size of opinion word vocabulary is large, so we use the negative sampling technique (Mikolov et al., 2013) to approximate Eq. (2). Specifically, we substitute $\log p(w_o|r_a, w_t)$ in the objective (5) with the following objective function:

$$\log\sigma\big(\varphi(w_o, w_t, r_a)\big) + \sum_{w_o' \in \mathcal{N}} \log\sigma\big(-\varphi(w_o', w_t, r_a)\big) \,, \quad (6)$$

where w_o' is a negative sample of opinion words in the vocabulary, $\mathcal{N}$ is the set of negative samples and σ is the sigmoid function. Then our final objective function is rewritten as:

$$\mathbb{E}_{q(R_a|\mathbf{x})}\Big[\log\sigma\big(\varphi(w_o, w_t, r_a)\big) + \sum_{w_o' \in \mathcal{N}} \log\sigma\big(-\varphi(w_o', w_t, r_a)\big)\Big] + \alpha H(q(R_a|\mathbf{x})) \,, \quad (7)$$

where α is a hyper-parameter which can adjust the expectation and entropy terms into the same scale (Marcheggiani and Titov, 2016).

3 Target Opinion Word Pairs Extraction

Target-opinion word pairs extraction is a well studied problem (Hu and Liu, 2004; Popescu and Etzioni, 2005; Bloom et al., 2007; Qiu et al., 2011). We designed five rules to extract potential target-opinion word pairs. Our method relies on Stanford Dependency Parser (Chen and Manning, 2014). We describe our rules as follows.

Rule 1: We extract pairs satisfying the grammatical relation *amod* (adjectival modifier) (De Marneffe and Manning, 2008). For example, in phrase "very good price," we extract "price" and "good" as a target-opinion pair.

Rule 2: We extract pairs satisfying the grammatical relation *nsubj* (nominal subject), and the head word is an adjective and the tail word is a noun. For example, in a sentence "The room is small," we can extract "room" and "small" as a target-opinion pair.

Dataset	TripAdvisor	BeerAdvocate
# docs	28,543	27,583
# target words	3,737	3,088
# opinion words	12,406	9,166
# pairs from R1	208,676	249,264
# pairs from R2	82,944	28,505
# pairs from R3	2,241	1,092
# pairs from R4	2,699	6,812
# pairs from R5	16,537	55,825

Table 1: Statistics of extracted target-opinion pairs .

Rule 3: Some verbs are also opinion words and they are informative. We extract pairs satisfying the grammatical relation *dobj* (direct object) when the head word is one of the following four words: "like", "dislike", "love", and "hate". For example, in the sentence "I like the smell," we can extract "smell" and "like" as a target-opinion pair.

Rule 4: We extract pairs satisfying the grammatical relation *xcomp* (open clausal complement), and the head word is one of the following word: "seem","look", "feel", "smell", and "taste". For example, in the sentence "This beer tastes spicy," we can extract "taste" and "spicy" as a target-opinion pair.

Rule 5: If the sentence contains some adjectives that can implicitly indicate aspects, we manually assign them to the corresponding aspects. According to (Lakkaraju et al., 2014), some adjectives serve both as target words and opinion words. For example, in the sentence "very tasty, and drinkable," the previous rules fail to extract any pair. But we know it contains a target-opinion pair, i.e., "taste-tasty." Most of these adjectives have the same root form with the aspects they indicated, e.g., "clean" (cleanliness), and "overpriced" (price). This kind of adjective can be extracted first and then we can obtain more similar adjectives using word similarities. For example, given "tasty," we could get "flavorful" by retrieving similar words.

Table 1 shows the statistics of the rule-based extraction on our two datasets. The first four rules can be applied to any dataset while the last one is domain dependent which requires human effort to identify these special adjectives. In practice, rule

5 can be removed to save human effort. The effect of removing rule 5 is shown in experiments.

After extracting potential target-opinion word pairs, we need to assign them to different aspects as supervision signals. We select some seed words to describe each aspect, and then calculate similarities between the extracted target (or opinion) word and seed words, and assign the pair to the aspect where one of its seed words has the highest similarity. The similarity we used is the cosine similarity between two word embeddings trained by word2vec (Mikolov et al., 2013). For example, suppose seed words {"room", "bed"} and {"business", "Internet"} are used to describe the aspect *room* and *business* respectively, and the candidate pair "pillow - soft" will be assigned to the aspect *room* if the similarity between "pillow" and "bed" is highest among all combinations.

4 Experiment

In this section, we report average sentiment classification accuracies over all aspects on binary DMSC task.

4.1 Datasets

We evaluate our model on TripAdvisor (Wang et al., 2010) and BeerAdvocate (McAuley et al., 2012; Lei et al., 2016; Yin et al., 2017) datasets, which contain seven aspects (value, room, location, cleanliness, check in/front desk, service, and business) and four aspects (feel, look, smell, and taste) respectively. We run the same preprocessing steps as (Yin et al., 2017). Both datasets are split into train/development/test sets with proportions 8:1:1. All methods can use development set to tune their hyper-parameters. Ratings of TripAdvisor and BeerAdvocate datasets are on scales of 1 to 5 and 0 to 5 respectively. But in BeerAdvocate, 0 star is rare, so we treat the scale as 1 to 5. We convert original scales to binary scales as follows: 1 and 2 stars are treated as negative, 3 is ignored, and 4 and 5 stars are treated as positive. In BeerAdvocate, most reviews have positive polarities, so to avoid the unbalanced issue, we perform data selection according to overall polarities. After data selection, the number of reviews with negative overall polarities and that with positive overall polarities are equal.

4.2 Compared Methods

To demonstrate the effectiveness of our method, we compare our model with following baselines:

Majority uses the majority of sentiment polarities in training sets as predictions.

Lexicon means using an opinion lexicon to assign sentiment polarity to an aspect (Read and Carroll, 2009; Pablos et al., 2015). We combine two popular opinion lexicons used by Hu and Liu (2004) and Wilson et al. (2005) to get a new one. If an opinion word from extracted pairs is in positive (negative) lexicon, it votes for positive (negative). When the opinion word is with a negation word, its polarity will be flipped. Then, the polarity of an aspect is determined by using majority voting among all opinion words associated with the aspect. When the number of positive and negative words is equal, we adopt two different ways to resolve it. For **Lexicon-R**, it randomly assigns a polarity. For **Lexicon-O**, it uses the overall polarity as the prediction. Since overall polarities can also be missing, for both Lexicon-R and Lexicon-O, we randomly assign a polarity in uncertain cases and report both mean and std based on five trials of random assignments.

Assign-O means directly using the overall polarity of a review in the development/test sets as the prediction for each aspect.

LRR assumes the overall polarity is a weighted sum of the polarity of each aspect (Wang et al., 2010). LRR can be regarded as the only existing weakly supervised baseline where both algorithm and source code are available.

BoW-DMSC-A is a simple softmax classifier using all annotated training data where the input is a bag-of-words feature vector of a document.

N-DMSC-A is the state-of-the-art neural network based model (Yin et al., 2017) (**N-DMSC**) in DMSC task using all annotated training data, which serves an upper bound to our method.

N-DMSC-O is to use overall polarities as "supervision" to train an N-DMSC and apply it to the classification task of each aspect at the test time.

N-DMSC-{50,100,200,500,1000} is the N-DMSC algorithm using partial data. In order to see our method is comparable to supervised methods using how many labeled data, we use $\{50, 100, 200, 500, 1000\}$ annotations of each aspect to train N-DMSC and compare them to our method. In addition to annotated data for training, there are extra 20% annotated data for validation.

Dataset	**TripAdvisor**				**BeerAdvocate**			
	DEV		TEST		DEV		TEST	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Majority	0.6286	–	0.6242	–	0.6739	–	0.6726	–
Lexicon-R	0.5914	0.0021	0.5973	0.0018	0.5895	0.0020	0.5881	0.0025
Lexicon-O	0.7153	0.0012	0.7153	0.0015	0.6510	0.0023	0.6510	0.0021
Assign-O	0.7135	0.0016	0.7043	0.0020	0.6652	0.0028	0.6570	0.0034
N-DMSC-O	0.7091	–	0.7064	–	0.6386	–	0.6493	–
LRR	0.6915	0.0045	0.6947	0.0024	0.5976	0.0110	0.5941	0.0113
VWS-DMSC (Our)	0.7577	0.0016	0.7561	0.0012	0.7502	0.0058	0.7538	0.0066
N-DMSC-50	0.7255	0.0231	0.7270	0.0204	0.7381	0.0143	0.7442	0.0157
N-DMSC-100	0.7482	0.0083	0.7487	0.0069	0.7443	0.0126	0.7493	0.0145
N-DMSC-200	0.7531	0.0040	0.7550	0.0043	0.7555	0.0096	0.7596	0.0092
N-DMSC-500	0.7604	0.0028	0.7616	0.0040	0.7657	0.0066	0.7713	0.0070
N-DMSC-1000	0.7631	0.0054	0.7638	0.0042	0.7708	0.0066	0.7787	0.0053
N-DMSC-A	0.8281	–	0.8334	–	0.8576	–	0.8635	–
BoW-DMSC-A	0.8027	–	0.8029	–	0.8069	–	0.8089	–

Table 2: Averaged accuracies on DMSC of unsupervised, weakly supervised, and supervised methods on TripAdvisor and BeerAdvocate. Methods involve randomness also report standard deviation.

Since the sampled labeled data may vary for different trials, we perform five trials of random sampling and report both mean and std of the results.

For our method, denoted as **VWS-DMSC**, the document representation we used is obtained from N-DMSC (Yin et al., 2017). They proposed a novel hierarchical iterative attention model in which documents and pseudo aspect related questions are interleaved at both word and sentence-level to learn an aspect-aware document representation. The pseudo aspect related questions are represented by aspect related keywords. In order to benefit from their aspect-aware representation scheme, we train an N-DMSC to extract the document representation using only overall polarities. In the iterative attention module, we use the pseudo aspect related keywords of all aspects released by Yin et al. (2017). One can also use document-to-document autoencoders (Li et al., 2015) to generate the document representation. In this way, our method can get rid of using overall polarities to generate the document representation. Hence, unlike LRR, it is not necessary for our method to use overall polarities. Here, to have a fair comparison with LRR, we use the overall polarities to generate document representation. For our method, we do not know which state is positive and which one is negative at training time, so the Hungarian algorithm (Kuhn, 1955) is used to resolve the assignment problem at the test time.

4.3 Results and Analysis

We show all results in Table 2, which consists of three blocks, namely, unsupervised, weakly supervised, and supervised methods.

For unsupervised methods, our method can outperform majority on both datasets consistently. But other weakly supervised methods cannot outperform majority on BeerAdvocate dataset, which shows these baselines cannot handle unbalanced data well since BeerAdvocate is more unbalanced than TripAdvisor. Our method outperforms Lexicon-R and Lexicon-O, which shows that predicting an opinion word based on a target word may be a better way to use target-opinion pairs, compared with performing a lexicon lookup using opinion words from extract pairs. Good performance of Lexicon-O and Assign-O demonstrates the usefulness of overall polarities in development/test sets. N-DMSC-O trained with the overall polarities cannot outperform Assign-O since N-DMSC-O can only see overall polarities in training set while Assign-O can see overall polarities for both development and test sets and does not involve learning and generalization.

For weakly supervised methods, LRR is the only open-source baseline in the literature on weakly supervised DMSC, and our method outperforms LRR by **6%** and **16%** on TripAdvisor and BeerAdvocate datasets. N-DMSC-O can also be considered as a weakly supervised method be-

Dataset	TripAdvisor		BeerAdvocate	
Rule	DEV	TEST	DEV	TEST
R1	0.7215	0.7174	0.7220	0.7216
R2	0.7172	0.7180	0.6864	0.6936
R3	0.6263	0.6187	0.6731	0.6725
R4	0.6248	0.6279	0.6724	0.6717
R5	0.5902	0.5856	0.7095	0.7066
- R1	0.7538	0.7481	0.7458	0.7474
- R2	0.7342	0.7368	0.7504	0.7529
- R3	0.7418	0.7397	0.7565	0.7558
- R4	0.7424	0.7368	0.7518	0.7507
- R5	0.7448	0.7440	0.7550	0.7548
All	0.7577	0.7561	0.7502	0.7538

Table 3: Averaged accuracies on DMSC. "R1 – R5" means only using a rule while "-R1 – -R5" means removing a rule from all the rules.

cause it only uses overall polarities as "supervision," and we still outperform it significantly. It is interesting that LRR is worse than N-DMSC-O. We guess that assuming that the overall polarity is a weighted sum of all aspect polarities may not be a good strategy to train each aspect's polarity or the document representation learned by N-DMSC is better than the bag-of-words representation.

For supervised block methods, BoW-DMSC-A and N-DMSC-A are both supervised methods using all annotated data, which can be seen as the upper bound of our algorithm. N-DMSC-A outperforms BoW-DMSC-A, which shows that the document representation based on neural network is better than the bag-of-words representation. Hence, we use the neural networks based document representation as input of the sentiment polarity classifier. Our results are comparable to N-DMSC-200 on TripAdvisor and N-DMSC-100 on BeerAdvocate.

4.4 Ablation Study

To evaluate effects of extracted rules, we performed an ablation study. We run our algorithm VWS-DMS with each rule kept or removed over two datasets. If no pairs extracted for one aspect in training set, the accuracy of this aspect will be 0.5, which is a random guess. From the Table 3 we can see that, the rule R1 is the most effective rule for both datasets. Rules R3/R4/R5 are less effective on their own. However, as a whole, they can still improve the overall performance. When considering removing each of rules, we found that our algorithm is quite robust, which indicates missing one of the rules may not hurt the performance much. Hence, if human labor is a major concern, rule 5 can be discarded. We found that sometimes removing one rule may even result in better accuracy (e.g., "-R3" for BeerAdvocate dataset). This means this rule may introduce some noises into the objective function. However, "-R3" can result in worse accuracy for TripAdvisor, which means it is still complementary to the other rules for this dataset.

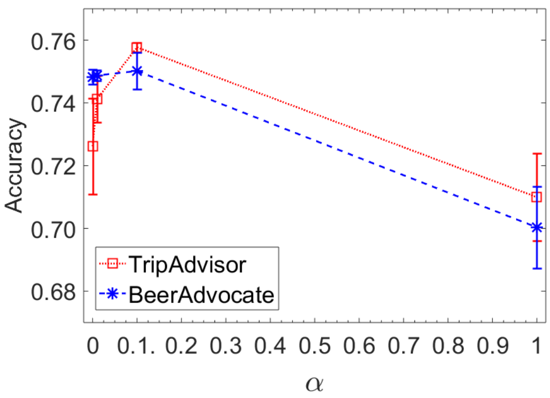


Figure 2: Parameter sensitivity analysis.

4.5 Parameter Sensitivity

We also conduct parameter sensitivity analysis of our approach. The parameter α in Equation (7) adjusts the expectation and entropy terms on the same scale. We test $\alpha = \{0, 0.01, 0.1, 1\}$ for both of the datasets. As we can see from Figure 2, $\alpha = 0.1$ is a good choice for both datasets.

4.6 Implementation Details

We implemented our models using TensorFlow (Abadi et al., 2016). For N-DMSC and LRR, we used code released by Yin et al. (2017) and Wang et al. (2010) respectively and followed their preprocessing steps and optimal settings.

Parameters are updated by using ADADELTA (Zeiler, 2012), an adaptive learning rate method. To avoid overfitting, we impose weight decay and drop out on both classifiers. The regularization coefficient and drop out rate are set to 10^{-3} and 0.3 respectively. The number of negative samples and α in our model are set to 10 and 0.1 respectively. For each document and each aspect, multiple target-opinion pairs are extracted. The opinion word classifier associated with an aspect will predict five target-opinion pairs at a time. These five target-opinion pairs are selected with bias. The probability of a pair being selected is proportional to the frequency of the opinion word to the power of -0.25. In this way, opinion words with low frequency are more likely to be selected compared to the uniform sampling. In order to initialize both classifiers better, the word embeddings are retrofitted (Faruqui et al., 2015) using PPDB (Ganitkevitch et al., 2013) semantic lexicons.

5 Related Work

In this section, we review the related work on document-level multi-aspect sentiment classification, target-opinion word pairs extraction, and variational methods.

Document-level Multi-Aspect Sentiment Classification. Wang et al. (2010) proposed a LRR model to solve this problem. LRR assumes the overall polarity is a weighted sum of all aspect polarities which are represented by word frequency features. LRR needs to use aspect keywords to perform sentence segmentation to generate the representation of each aspect. To address the limitation of using aspect keywords, LARAM (Wang et al., 2011) assumes that the text content describing a particular aspect is generated by sampling words from a topic model corresponding to the latent aspect. Both LRR and LARAM can only access to overall polarities in the training data, but not gold standards of aspect polarities. Meng et al. (2018) proposed a weakly supervised text classification method which can take label surface names, class-related keywords, or a few labeled documents as supervision. Ramesh et al. (2015) developed a weakly supervised joint model to identify aspects and the corresponding sentiment polarities in online courses. They treat aspect (sentiment) related seed words as weak supervision. In the DMSC task which is a fine-grained text classification task, the label surface names or keywords for some aspects would be very similar. Given that the inputs are the same and the supervisions are similar, weakly supervised models cannot distinguish them. So we do not consider them as our baselines. Yin et al. (2017) modeled this problem as a machine comprehension problem under a multi-task learning framework. It also needs aspect keywords to generate aspect-aware document representations. Moreover, it can access gold standards of aspect polarities and achieved state-of-the-art performance on this task. Hence, it can serve as an upper bound. Some sentence-level aspect based sentiment classification methods (Wang et al., 2016b, 2018) can be directly applied to the DMSC task, because they can solve aspect category sentiment classification task. For example, given a sentence "the restaurant is expensive," the aspect category sentiment classification task aims to classify the polarity of the aspect category "price" to be *negative*. The aspect categories are predefined which are the same as the DMSC task. Some of them (Tang et al., 2016a,b; Chen et al., 2017; Ma et al., 2017) cannot because they are originally designed for aspect term sentiment classification task. For example, given a sentence "I loved their fajitas," the aspect term sentiment classification task aims to classify the polarity of the aspect term "fajitas" to be *positive*. The aspect terms appearing in the sentence should be provided as inputs.

Target Opinion Word Pairs Extraction. There are two kinds of methods, namely, rule based methods and learning based methods to solve this task. Rule based methods extract target-opinion word pairs by mining the dependency tree paths between target words and opinion words. Learning based methods treat this task as a sequence labeling problem, mapping each word to one of the following categories: target, opinion, and other.

(Hu and Liu, 2004) is one of earliest rule based methods to extract target-opinion pairs. An opinion word is restricted to be an adjective. Target words are extracted first, and then an opinion word is linked to its nearest target word to form a pair. Popescu and Etzioni (2005) and Bloom et al. (2007) manually designed dependency tree path templates to extract target-opinion pairs. If the path between a target word candidate and an opinion word candidate belongs to the set of path templates, the pair will be extracted. Qiu et al. (2011) identified dependency paths that link opinion words and targets via a bootstrapping process. This method only needs an initial opinion lexicon to start the bootstrapping process. Zhuang et al. (2006) adopted a supervised learning algorithm to learn valid dependency tree path templates, but it requires target-opinion pairs annotations.

Learning based methods require lots of target-opinion pairs annotations. They trained conditional random fields (CRF) (Lafferty et al., 2001) based models (Jakob and Gurevych, 2010; Yang and Cardie, 2012; Wang et al., 2016a) or deep neural networks (Liu et al., 2015; Wang et al., 2017; Li and Lam, 2017) to predict the label (target, opinion or other) of each word. Jakob and Gurevych (2010) and Li et al. (2012) extracted target-opinion pairs without using using any labeled data in the domain of interest, but it needs lots of labeled data in another related domain.

In this paper, we only use very simple rules to extract target-opinion pairs to validate the effectiveness of our approach. If better pairs can be extracted, we can further improve our results.

Variational Methods. Variational autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) (VAEs) use a neural network to parameterize a probability distribution. VAEs consists of an encoder which parameterizes posterior probabilities and a decoder which parameterizes the reconstruction likelihood given a latent variable. VAEs inspire many interesting works (Titov and Khoddam, 2015; Marcheggiani and Titov, 2016; Šuster et al., 2016; Zhang et al., 2018; Chen et al., 2018) which are slightly different from VAEs. Their encoders produce a discrete distribution while the encoder in VAEs yields a continuous latent variable. Titov and Khoddam (2015) aimed to solve semantic role labeling problem. The encoder is essentially a semantic role labeling model which predicts roles given a rich set of syntactic and lexical features. The decoder reconstructs argument fillers given predicted roles. Marcheggiani and Titov (2016) aimed to solve unsupervised open domain relation discovery. The encoder is a feature-rich relation extractor, which predicts a semantic relation between two entities. The decoder reconstructs entities relying on the predicted relation. Šuster et al. (2016) tried to learn multi-sense word embeddings. The encoder uses bilingual context to choose a sense for a given word. The decoder predicts context words based on the chosen sense and the given word. Zhang et al. (2018) aimed to solve knowledge graph powered question answering. Three neural networks are used to parameterize probabilities of a topic entity given a query and an answer, an answer based on a query and a predicted topic, and the topic given the query. Chen et al. (2018) aimed to infer missing links in a knowledge graph. Three neural networks are used to parameterize probabilities of a latent path given two entities and a relation, a relation based on two entities and the chosen latent path, and the relation given the latent path. Our method also uses neural networks to parameterize two discrete distributions but aims to solve the DMSC task.

6 Conclusion

In this paper, we propose a variational approach to weakly supervised DMSC. We extract many target-opinion word pairs from dependency parsers using simple rules. These pairs can be "supervision" signals to predict sentiment polarity. Our objective function is to predict an opinion word given a target word. After introducing the sentiment polarity as a latent variable, we can learn a sentiment polarity classifier by optimizing the variational lower bound. We show that we can outperform weakly supervised baselines by a large margin and achieve comparable results to the supervised method with hundreds of labels per aspect, which can reduce a lot of labor work in practice. In the future, we plan to explore better target-opinion word extraction approaches to find better "supervision" signals.

Acknowledgments

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong. Ziqian Zeng has been supported by the Hong Kong Ph.D. Fellowship. We thank Intel Corporation for supporting our deep learning related research. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of OSDI*, pages 265–283.

Kenneth Bloom, Navendu Garg, Shlomo Argamon, et al. 2007. Extracting appraisal expressions. In *Proceedings of NAACL-HLT*, pages 308–315.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*, pages 452–461.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. In *Proceedings of NAACL-HLT*, pages 1823–1832.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *Proceedings of EMNLP*, pages 1035–1045.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *ICLR*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics*, 2(1-2):83–97.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *Proceedings of NIPS workshop on Deep Learning and Representation Learning*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*, pages 107–117.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of ACL*, pages 410–419.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*, pages 1106–1115.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of EMNLP*, pages 2886–2892.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of EMNLP*, pages 1433–1443.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of IJCAI*, pages 4068–4074.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of ICDM*, pages 1020–1025.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 983–992. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Aitor García Pablos, Montse Cuadros, and German Rigau. 2015. V3: Unsupervised aspect based sentiment analysis for semeval2015 task 12. In *Proceedings of SemEval*, pages 714–718.

Ana-Maria Popescu and Orena Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of EMNLP-HLT*, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37:9–27.

Arti Ramesh, Shachi H Kumar, James Foulds, and Lise Getoor. 2015. Weakly supervised models of aspect-sentiment for online course discussion forums. In *Proceedings of ACL*, pages 74–83.

Jonathon Read and John Carroll. 2009. Weakly supervised techniques for domain-independent sentiment classification. In *Proceedings of CIKM workshop on Topic-sentiment Analysis for Mass Opinion*, pages 45–52. ACM.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML*, pages 1278–1286.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of NAACL-HLT*, pages 1346–1356.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP*, pages 214–224.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of NAACL-HLT*, pages 1–10.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*, pages 783–792.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of KDD*, pages 618–626.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018. Aspect sentiment classification with both word-level and clause-level attention networks. In *IJCAI*, pages 4439–4445.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 616–626.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Proceedings of AAAI*, pages 3316–3322.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016b. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of EMNLP*, pages 606–615.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of EMNLP-HLT*, pages 347–354.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of EMNLP-CoNLL*, pages 1335–1345.

Yichun Yin, Yangqiu Song, and Ming Zhang. 2017. Document-level multi-aspect sentiment classification as machine comprehension. In *Proceedings of EMNLP*, pages 2034–2044.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of AAAI*, pages 6069–6076.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of CIKM*, pages 43–50.

HiGRU: Hierarchical Gated Recurrent Units for Utterance-level Emotion Recognition

Wenxiang Jiao[1], **Haiqin Yang**[2,3], **Irwin King**[1], and **Michael R. Lyu**[1]
[1] Department of Computer Science and Engineering,
The Chinese University of Hong Kong, HKSAR, China
[2] Meitu and [3] The Hang Seng University of Hong Kong, HKSAR, China
{wxjiao,king,lyu}@cse.cuhk.edu.hk, hqyang@ieee.org

Abstract

In this paper, we address three challenges in utterance-level emotion recognition in dialogue systems: (1) the same word can deliver different emotions in different contexts; (2) some emotions are rarely seen in general dialogues; (3) long-range contextual information is hard to be effectively captured. We therefore propose a hierarchical Gated Recurrent Unit (HiGRU) framework with a lower-level GRU to model the word-level inputs and an upper-level GRU to capture the contexts of utterance-level embeddings. Moreover, we promote the framework to two variants, HiGRU with individual features fusion (HiGRU-f) and HiGRU with self-attention and features fusion (HiGRU-sf), so that the word/utterance-level individual inputs and the long-range contextual information can be sufficiently utilized. Experiments on three dialogue emotion datasets, IEMOCAP, Friends, and EmotionPush demonstrate that our proposed HiGRU models attain at least 8.7%, 7.5%, 6.0% improvement over the state-of-the-art methods on each dataset, respectively. Particularly, by utilizing only the textual feature in IEMOCAP, our HiGRU models gain at least 3.8% improvement over the state-of-the-art conversational memory network (CMN) with the trimodal features of text, video, and audio.

1 Introduction

Emotion recognition is a significant artificial intelligence research topic due to the promising potential of developing empathetic machines for people. Emotion is a universal phenomena across different cultures and mainly consists of six basic types: anger, disgust, fear, happiness, sadness, and surprise (Ekman, 1971, 1992).

In this paper, we focus on textual dialogue systems because textual feature dominates the performance over audio and video features (Poria et al., 2015, 2017). In utterance-level emotion recognition, an utterance (Olson, 1977) is a unit of speech bounded by breathes or pauses and its goal is to tag each utterance in a dialogue with the indicated emotion.

Role	Utterance	Emotion
Rachel	Oh okay, I'll fix that to. What's her e-mail address?	Neutral
Ross	Rachel!	Anger
Rachel	All right, I promise. I'll fix this. I swear. I'll-I'll- I'll-I'll talk to her.	Non-neutral
Ross	***Okay!***	***Anger***
Rachel	***Okay.***	***Neutral***
Nurse	This room's available.	Neutral
Rachel	***Okay!***	***Joy***
Rachel	Okay wait!	Non-neutral
Rachel	You listen to me!	Anger

Figure 1: The word "okay" exhibits different emotions in the American television sitcom, Friends.

In this task, we address three challenges: First, the same word can deliver different emotions in different contexts. For example, in Figure 1, the word "okay" can deliver three different emotions, anger, neutral, and joy, respectively. Strong emotions like joy and anger may be indicated by the symbols "!" or "?" along the word. To identify a speaker's emotion precisely, we need to explore the dialogue context sufficiently. Second, some emotions are rarely seen in general dialogues. For example, people are usually calm and present a neutral emotion while only in some particular situations, they express strong emotions, like anger or fear. Thus we need to be sensitive to the minority emotions while relieving the effect of the majority emotions. Third, the long-range contextual information is hard to be effectively captured in an utterance/dialogue, especially when the length of an utterance/dialogue in the testing set is longer than those in the training set.

To tackle these challenges, we propose a hierarchical Gated Recurrent Unit (HiGRU) framework for the utterance-level emotion recognition

Proceedings of NAACL-HLT 2019, pages 397–406
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

in dialogue systems. More specifically, HiGRU is composed by two levels of bidirectional GRUs, a lower-level GRU to model the word sequences of each utterance to produce individual utterance embeddings, and an upper-level GRU to capture the sequential and contextual relationship of utterances. We further promote the proposed HiGRU to two variants, HiGRU with individual features fusion (HiGRU-f), and HiGRU with self-attention and features fusion (HiGRU-sf). In HiGRU-f, the individual inputs, i.e., the word embeddings in the lower-level GRU and the individual utterance embeddings in the upper-level GRU, are concatenated with the hidden states to generate the contextual word/utterance embeddings, respectively. In HiGRU-sf, a self-attention layer is placed on the hidden states from the GRU to learn long-range contextual embeddings, which are concatenated with the original individual embeddings and the hidden states to generate the contextual word/utterance embeddings. Finally, the contextual utterance embedding is sent to a fully-connected (FC) layer to determine the corresponding emotion. To alleviate the effect of data imbalance issue, we follow (Khosla, 2018) to train our models by minimizing a weighted categorical cross-entropy.

We summarize our contributions as follows:

- We propose a HiGRU framework to better learn both the individual utterance embeddings and the contextual information of utterances, so as to recognize the emotions more precisely.
- We propose two progressive HiGRU variants, HiGRU-f and HiGRU-sf, to sufficiently incorporate the individual word/utterance-level information and the long-range contextual information respectively.
- We conduct extensive experiments on three textual dialogue emotion datasets, IEMOCAP, Friends, and EmotionPush. The results demonstrate that our proposed HiGRU models achieve at least 8.7%, 7.5%, 6.0% improvement over state-of-the-art methods on each dataset, respectively. Particularly, by utilizing only the textual feature in IEMOCAP, our proposed HiGRU models gain at least 3.8% improvement over the existing best model, conversational memory network (CMN) with not only the text feature, but also the visual, and audio features.

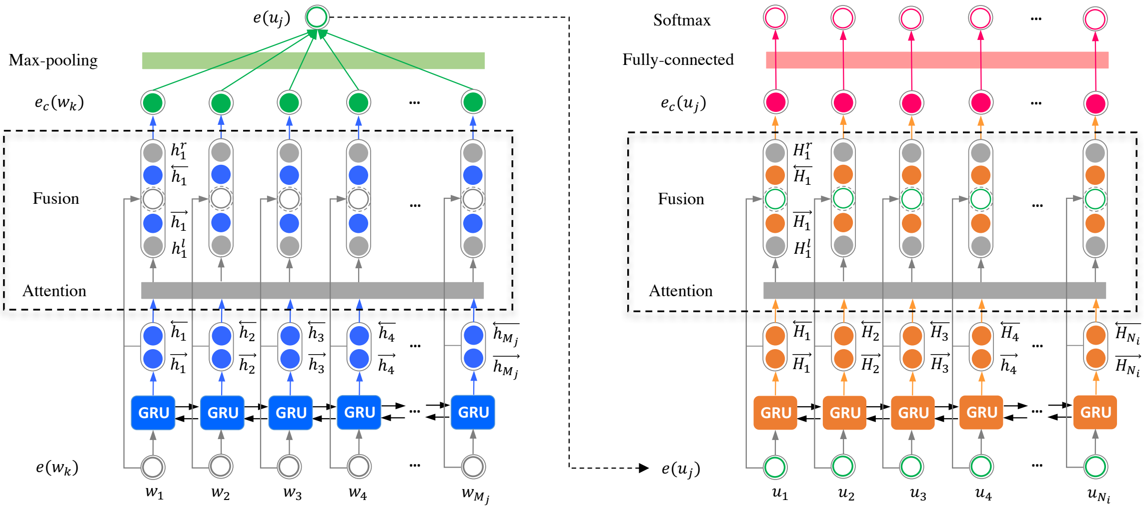
2 Related Work

Text-based emotion recognition is a long-standing research topic (Wilson et al., 2004; Yang et al., 2007; Medhat et al., 2014). Nowadays, deep learning technologies have become dominant methods due to the outstanding performance. Some prominent models include recursive autoencoders (RAEs) (Socher et al., 2011), convolutional neural networks (CNNs) (Kim, 2014), and recurrent neural networks (RNNs) (Abdul-Mageed and Ungar, 2017). However, these models treat texts independently thus cannot capture the inter-dependence of utterances in dialogues (Kim, 2014; Lai et al., 2015; Grave et al., 2017; Chen et al., 2016; Yang et al., 2016). To exploit the contextual information of utterances, researchers mainly explore in two directions: (1) extracting contextual information among utterances, or (2) enriching the information embedded in the representations of words and utterances.

Contextual Information Extraction. The RNN architecture is a standard way to capture the sequential relationship of data. Poria et al. propose a bidirectional contextual long short-term memory (LSTM) network, termed bcLSTM, to model the context of textual features extracted by CNNs. Hazarika et al. improve bcLSTM by a conversational memory network (CMN) to capture the self and inter-speaker emotional influence, where GRU is utilized to model the self-influence and the attention mechanism is employed to excavate the inter-speaker emotional influence. Though CMN is reported to attain better performance than bcLSTM on IEMOCAP (Hazarika et al., 2018), the memory network is too complicated for small-size dialogue datasets.

Representation Enrichment. Multimodal features have been utilized to enrich the representation of utterances (Poria et al., 2015, 2017). Previous work indicate that textual features dominate the performance of recognizing emotions in contrast to visual or audio features (Poria et al., 2015, 2017). Recently, the textual features are mainly extracted by CNNs to learn individual utterance embeddings (Poria et al., 2015, 2017; Zahiri and Choi, 2018; Hazarika et al., 2018). However, CNNs do not capture the contextual information within each utterance well.

On the other hand, hierarchical RNNs have been proposed and demonstrated good performance in



Individual Utterance Embedding **Contextual Utterance Embedding**

Figure 2: The architecture of our proposed HiGRU-sf. "Attention" denotes self-attention. By removing the "Attention" layer, we attain HiGRU-f, and by further removing the "Fusion" layer, we can recover the vanilla HiGRU.

conventional text classification task (Tang et al., 2015), dialogue act classification (Liu et al., 2017; Kumar et al., 2018), and speaker change detection (Meng et al., 2017). But they are not well explored in the task of utterance-level emotion recognition in dialogue systems.

3 Approach

The task of utterance-level emotion recognition is defined as follows:

Definition 1 (Utterance-level Emotion Recognition). *Suppose we are given a set of dialogues,* $\mathcal{D} = \{D_i\}_{i=1}^{L}$, *where* L *is the number of dialogues. In each dialogue,* $D_i = \{(u_j, s_j, c_j)\}_{j=1}^{N_i}$, *is a sequence of* N_i *utterances, where the utterance* u_j *is spoken by the speaker* $s_j \in \mathcal{S}$ *with a certain emotion* $c_j \in \mathcal{C}$. *All speakers compose the set* $\mathcal{S}$ *and the set* $\mathcal{C}$ *consists of all emotions, such as anger, joy, sadness, and neutral. Our goal is to train a model* $\mathcal{M}$ *to tag each new utterance with an emotion label from* $\mathcal{C}$ *as accurately as possible.*

To solve this task, we propose a hierarchical Gated Recurrent Units (HiGRU) framework and extend two progressive variants, HiGRU with individual features fusion (HiGRU-f) and HiGRU with self-attention and features fusion (HiGRU-sf) (illustrated in Figure 2).

3.1 HiGRU: Hierarchical GRU

The vanilla HiGRU consists of two-level GRUs: the lower-level bidirectional GRU is to learn the individual utterance embedding by modeling the word sequence within an utterance and the upper-level bidirectional GRU is to learn the contextual utterance embedding by modeling the utterance sequence within a dialogue.

Individual Utterance Embedding. For the j^{th} utterance in D_i, $u_j = \{w_k\}_{k=1}^{M_j}$, where M_j is the number of words in the utterance u_j. The corresponding sequence of individual word embeddings $\{e(w_k)\}_{k=1}^{M_j}$ are fed into the lower-level bidirectional GRU (Cho et al., 2014) to learn the individual utterance embedding in two opposite directions:

$$\overrightarrow{h_k} = \mathrm{GRU}(e(w_k), \overrightarrow{h_{k-1}}), \quad (1)$$

$$\overleftarrow{h_k} = \mathrm{GRU}(e(w_k), \overleftarrow{h_{k+1}}). \quad (2)$$

The two hidden states $\overrightarrow{h_k}$ and $\overleftarrow{h_k}$ are concatenated into $hs = [\overrightarrow{h_k}; \overleftarrow{h_k}]$ to produce the *contextual word embedding* for w_k via the tanh activation function on a linear transformation:

$$e_c(w_k) = \tanh(W_w \cdot hs + b_w), \quad (3)$$

where $W_w \in \mathbb{R}^{d_1 \times 2d_1}$ and $b_w \in \mathbb{R}^{d_1}$ are the model parameters, d_0 and d_1 are the dimensions of word embeddings and the hidden states of the lower-level GRU, respectively.

The individual utterance embedding is then obtained by max-pooling on the contextual word embeddings within the utterance:

$$e(u_j) = \mathrm{maxpool}\left(\{e_c(w_k)\}_{k=1}^{M_j}\right). \quad (4)$$

Contextual Utterance Embedding. For the i^{th} dialogue, $D_i = \{(u_j, s_j, c_j)\}_{j=1}^{N_i}$, the learned individual utterance embeddings, $\{e(u_j)\}_{j=1}^{N_i}$, are fed into the upper-level bidirectional GRU to capture the sequential and contextual relationship of utterances in a dialogue:

$$\overrightarrow{H_j} = \text{GRU}(e(u_j), \overrightarrow{H_{j-1}}), \tag{5}$$

$$\overleftarrow{H_j} = \text{GRU}(e(u_j), \overleftarrow{H_{j+1}}). \tag{6}$$

Here, the hidden states of the upper-level GRU are represented by $H_j \in \mathbb{R}^{d_2}$, to distinguish from those learned in the lower-level GRU denoted by h_k. Accordingly, we can obtain the *contextual utterance embedding* by

$$e_c(u_j) = \tanh(W_u \cdot Hs + b_u), \tag{7}$$

where $Hs = [\overrightarrow{H_j}; \overleftarrow{H_j}]$, $W_u \in \mathbb{R}^{d_2 \times 2d_2}$ and $b_u \in \mathbb{R}^{d_2}$ are the model parameters, d_2 is the dimension of the hidden states in the upper-level GRU. Since the emotions are recognized at utterance-level, the learned contextual utterance embedding $e_c(u_j)$ is directly fed to a FC layer followed by a softmax function to determine the corresponding emotion label:

$$\hat{y}_j = \text{softmax}(W_{fc} \cdot e_c(u_j) + b_{fc}), \tag{8}$$

where $\hat{y}_j$ is the predicted vector over all emotions, and $W_{fc} \in \mathbb{R}^{|\mathcal{C}| \times d_2}$, $b_{fc} \in \mathbb{R}^{|\mathcal{C}|}$.

3.2 HiGRU-f: HiGRU + Individual Features Fusion

The vanilla HiGRU contains two main issues: (1) the individual word/utterance embeddings are diluted with the stacking of layers; (2) the upper-level GRU tends to gather more contextual information from the majority emotions, which deteriorates the overall model performance.

To resolve these two problems, we propose to fuse individual word/utterance embeddings with the hidden states from GRUs so as to strengthen the information of each word/utterance in its contextual embedding. This variant is named as HiGRU-f, representing HiGRU with individual features fusion. Hence, the lower-level GRU can maintain individual word embeddings and the upper-level GRU can relieve the effect of majority emotions and attain a more precise utterance representation for different emotions. Specifically, the contextual embeddings are updated as:

$$e_c(w_k) = \tanh(W_w \cdot hs^f + b_w), \tag{9}$$

$$e_c(u_j) = \tanh(W_u \cdot Hs^f + b_u), \tag{10}$$

where $W_w \in \mathbb{R}^{d_1 \times (d_0 + 2d_1)}$, $W_u \in \mathbb{R}^{d_2 \times (d_1 + 2d_2)}$, $hs^f = [\overrightarrow{h_k}; e(w_k); \overleftarrow{h_k}]$, and $Hs^f = [\overrightarrow{H_j}; e(u_j); \overleftarrow{H_j}]$.

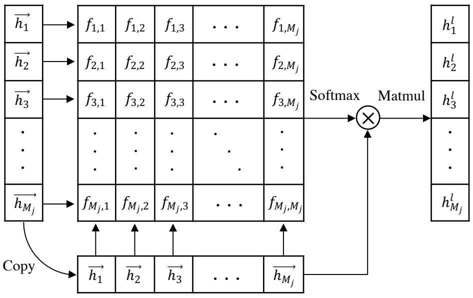


Figure 3: Self-attention over the forward hidden states of GRU.

3.3 HiGRU-sf: HiGRU + Self-Attention and Feature Fusion

Another challenging issue is to extract the contextual information of long sequences, especially the sequences in the testing set that are longer than those in the training set (Bahdanau et al., 2014). To fully utilize the global contextual information, we place a self-attention layer upon the hidden states of HiGRU and fuse the attention outputs with the individual word/utterance embeddings and the hidden states to learn the contextual word/utterance embeddings. Hence, this variant is termed HiGRU-sf, representing HiGRU with self-attention and features fusion.

Particularly, we apply self-attention upon the forward and backward hidden states separately to produce the left context embedding, h_k^l (H_j^l), and the right context embedding, h_k^r (H_j^r), respectively. This allows us to gather the unique global contextual information at the current step in two opposite directions and yield the corresponding contextual embeddings computed as follows:

$$e_c(w_k) = \tanh(W_w \cdot hs^{sf} + b_w), \tag{11}$$

$$e_c(u_j) = \tanh(W_u \cdot Hs^{sf} + b_u), \tag{12}$$

where $W_w \in \mathbb{R}^{d_1 \times (d_0 + 4d_1)}$, $W_u \in \mathbb{R}^{d_2 \times (d_1 + 4d_2)}$, $hs^{sf} = [h_k^l; \overrightarrow{h_k}; e(w_k); \overleftarrow{h_k}; h_k^r]$, and $Hs^{sf} = [H_j^l; \overrightarrow{H_j}; e(u_j); \overleftarrow{H_j}; H_j^r]$.

Self-Attention (SA). The self-attention mechanism is an effective non-recurrent architecture to compute the relation between one input to all other inputs and has been successfully applied in various natural language processing applications such as reading comprehension (Hu et al., 2018), and neural machine translation (Vaswani et al., 2017). Figure 3 shows the dot-product SA over the forward hidden states of GRU to learn the left context h_k^l. Each element in the attention matrix is computed by

$$f(\overrightarrow{h_k}, \overrightarrow{h_p}) = \begin{cases} \overrightarrow{h_k}^\top \overrightarrow{h_p}, & \text{if } k, p \le M_j, \\ -\infty, & \text{otherwise.} \end{cases} \tag{13}$$

An attention mask is then applied to waive the inner attention between the sequence inputs and paddings. At each step, the corresponding left context h_k^l is then computed by the weighted sum of all the forward hidden states:

$$h_k^l = \sum_{p=1}^{M_j} a_{kp} \overrightarrow{h_p}, \; a_{kp} = \frac{\exp(f(\overrightarrow{h_k}, \overrightarrow{h_p}))}{\sum_{p'=1}^{M_j} \exp\big(f(\overrightarrow{h_k}, \overrightarrow{h_{p'}})\big)}, \tag{14}$$

where a_{kp} is the weight of $\overrightarrow{h_p}$ to be included in h_k^l. The right context h_k^r can be computed similarly.

3.4 Model Training

Following (Khosla, 2018) which attains the best performance in the EmotionX shared task (Hsu and Ku, 2018), we minimize a weighted categorical cross-entropy on each utterance of all dialogues to optimize the model parameters:

$$loss = -\frac{1}{\sum_{i=1}^{L} N_i} \sum_{i=1}^{L} \sum_{j=1}^{N_i} \omega(c_j) \sum_{c=1}^{|\mathcal{C}|} y_j^c \log_2(\hat{y}_j^c), \tag{15}$$

where y_j is the original one-hot vector of the emotion labels, and y_j^c and $\hat{y}_j^c$ are the elements of y_j and $\hat{y}_j$ corresponding to the class c.

Similar to (Khosla, 2018), we assign the loss weight $\omega(c_j)$ inversely proportional to the number of training utterances in the class c_j, denoted by I_c, i.e., assigning larger loss weights for the minority classes to relieve the data imbalance issue. The difference is that we add a constant α to adjust the smoothness of the distribution. Then, we have:

$$\frac{1}{\omega(c)} = \frac{I_c^\alpha}{\sum_{c'=1}^{|\mathcal{C}|} I_{c'}^\alpha}. \tag{16}$$

4 Experiments

We conduct systematical experiments to demonstrate the advantages of our proposed HiGRU models.

4.1 Datasets

The experiments are carried out on three textual dialogue emotion datasets (see the statistics in Table 1):

IEMOCAP[1]**.** It contains approximately 12 hours of audiovisual data, including video, speech, motion capture of face, text transcriptions. Following (Poria et al., 2017; Hazarika et al., 2018): (1) We apply the first four sessions for training and the last session for test; (2) The validation set is extracted from the shuffled training set with the ratio of 80:20; (3) We only evaluate the performance on four emotions: anger, happiness, sadness, neutral, and remove the rest utterances.

Friends[2]**.** The dataset is annotated from the Friends TV Scripts (Hsu and Ku, 2018), where each dialogue in the dataset consists of a scene of multiple speakers. Totally, there are 1,000 dialogues, which are split into 720, 80, and 200 dialogues for training, validation, and testing, respectively. Each utterance in a dialogue is labeled by one of the eight emotions: anger, joy, sadness, neutral, surprise, disgust, fear, and non-neutral.

EmotionPush[3]**.** The dataset consists of private conversations between friends on the Facebook messenger collected by an App called EmotionPush, which is released for the EmotionX shared task (Hsu and Ku, 2018). Totally, there are 1,000 dialogues, which are split into 720, 80, 200 dialogue for training, validation, and testing, respectively. All the utterances are categorized into one of the eight emotions as in the Friends dataset.

Following the setup of (Hsu and Ku, 2018), in Friends and EmotionPush, we only evaluate the model performance on four emotions: anger, joy, sadness, and neutral, and we exclude the contribution of the rest emotion classes during training by setting their loss weights to zero.

Data Preprocessing. We preprocess the datasets by the following steps: (1) The utterances are split into tokens with each word being made into the lowercase; (2) All non-alphanumerics except "?" and "!" are removed because these two symbols usually exhibit strong emotions, such as surprise,

---

[1] `https://sail.usc.edu/iemocap/`

[2] `http://doraemon.iis.sinica.edu.tw/emotionlines`

[3] `http://doraemon.iis.sinica.edu.tw/emotionlines`

Dataset	#Dialogue (#Utterance)			#Emotion				
	Train	Val	Test	Ang	Hap/Joy	Sad	Neu	Others
IEMOCAP	96 (3,569)	24 (721)	31 (1,208)	1,090	1,627	1,077	1,704	0
Friends	720 (10,561)	80 (1,178)	200 (2,764)	759	1,710	498	6,530	5,006
EmotionPush	720 (10,733)	80 (1,202)	200 (2,807)	140	2,100	514	9,855	2,133

Table 1: Statistics of the textual dialogue datasets.

joy and anger; (3) We build a dictionary based on the words and symbols extracted, and follow (Poria et al., 2017) to represent the tokens by the publicly available 300-dimensional word2vec[4] vectors trained on 100 billion words from Google News. The tokens not included in the word2vec dictionary are initialized by randomly-generated vectors.

4.2 Evaluation Metrics

To conduct fair comparison, we adopt two metrics as (Hsu and Ku, 2018), the weighted accuracy (WA) and unweighted accuracy (UWA):

$$\mathrm{WA} = \sum_{c=1}^{|\mathcal{C}|} p_c \cdot a_c, \quad \mathrm{UWA} = \frac{1}{|\mathcal{C}|} \sum_{c=1}^{|\mathcal{C}|} a_c, \quad (17)$$

where p_c is the percentage of the class c in the testing set, and a_c is the corresponding accuracy.

Generally, recognizing strong emotions may provide more value than detecting the neutral emotion (Hsu and Ku, 2018). Thus, in Friends and EmotionPush, UWA is a more favorite evaluation metric because WA is heavily compromised with the large proportion of the neutral emotion.

4.3 Compared Methods

Our proposed vanilla HiGRU, HiGRU-f, and HiGRU-sf[5] are compared with the following state-of-the-art baselines:

bcLSTM (Poria et al., 2017): a bidirectional contextual LSTM with multimodal features extracted by CNNs;

CMN (Hazarika et al., 2018): a conversational memory network with multimodal features extracted by CNNs;

SA-BiLSTM (Luo et al., 2018): a self-attentive bidirectional LSTM model, a neat model achieving the second place of EmotionX Challenge (Hsu and Ku, 2018);

CNN-DCNN (Khosla, 2018): a convolutional-deconvolutional autoencoder with more handmade features, the winner of EmotionX Challenge (Hsu and Ku, 2018);

bcLSTM$_*$ and **bcGRU**: our implemented bcLSTM and bcGRU with the weighted loss on the textual feature extracted from CNNs.

4.4 Training Procedure

All our implementations are coded on the Pytorch framework. To prevent the models fitting the order of data, we randomly shuffle the training set at the beginning of every epoch.

Parameters. For **bcLSTM$_*$** and **bcGRU**, the CNN layer follows the setup of (Kim, 2014), i.e., consisting of the kernels of 3, 4, and 5 with 100 feature maps each. The convolution results of each kernel are fed to a max-over-time pooling operation. The dimension of the hidden states of the upper-level bidirectional LSTM or GRU is set to 300. For HiGRU, HiGRU-f, and HiGRU-sf, the dimensions of hidden states are set to 300 for both levels. The final FC layer contains two sub-layers with 100 neurons each.

Training. We adopt Adam (Kingma and Ba, 2014) as the optimizer and set an initial learning rate, 1×10^{-4} for IEMOCAP and 2.5×10^{-4} for Friends and EmotionPush, respectively. An annealing strategy is utilized by decaying the learning rate by half every 20 epochs. Early stopping with a patience of 10 is adopted to terminate training based on the accuracy of the validation set. Specifically, following the best models on each dataset, the parameters are tuned to optimize WA on the validation set of IEMOCAP and to optimize UWA on the validation set of Friends and EmotionPush, respectively. Gradient clipping with a norm of 5 is applied to model parameters. To prevent overfitting, dropout with a rate of 0.5 is applied after the contextual word/utterance embeddings, and the FC layer.

Loss weights. For Friends and EmotionPush, as mentioned in Section 4.1, the loss weights are set to zero except the four considered emotions, to ignore the others during training. Besides, the power rate α of loss weights is tested from 0 to 1.5 with

---

[4] `https://code.google.com/archive/p/word2vec/`

[5] `https://github.com/wxjiao/HiGRUs`

Model (Feat)	Ang	Hap	Sad	Neu	WA	UWA
bcLSTM (T)	76.07	78.97	76.23	67.44	73.6	74.6
(T+V+A)	77.98	79.31	78.30	69.92	76.1	76.3
CMN (T)	-	-	-	-	74.1	-
(T+V+A)	**89.88**	81.75	77.73	67.32	77.6	79.1
bcLSTM$_*$ (T)	75.29	79.40	78.07	76.53	77.7$_{(1.1)}$	77.3$_{(1.4)}$
bcGRU (T)	77.20	80.99	76.26	72.50	76.9$_{(1.6)}$	76.7$_{(1.3)}$
HiGRU (T)	75.41	**91.64**	79.79	70.74	80.6$_{(0.5)}$	79.4$_{(0.5)}$
HiGRU-f (T)	76.69	88.91	80.25	75.92	81.5$_{(0.7)}$	80.4$_{(0.5)}$
HiGRU-sf (T)	74.78	89.65	**80.50**	**77.58**	**82.1**$_{(0.4)}$	**80.6**$_{(0.2)}$

Table 2: Experimental results on IEMOCAP. "(Feat)" represents the features used in the models, where T, V, and A denote the textual, visual, and audio features, respectively. The results of bcLSTM and CMN are from (Poria et al., 2017) and (Hazarika et al., 2018), respectively. The underlined results are derived by us accordingly, while "-" means the results are unavailable from the original paper.

a step of 0.25, and we use the best one for each model and dataset.

4.5 Main Results

Table 2 and Table 3 report the average results of 10 trials each on the three datasets, where the standard deviations of WA and UWA are recorded by the subscripts in round brackets. The results of bcLSTM, CMN, SA-BiLSTM, and CNN-DCNN are copied directly from the original papers for a fair comparison because we follow the same configuration for the corresponding datasets. From the results, we have the following observations:

(1) Baselines. Our implemented bcLSTM$_*$ and bcGRU, attain comparable performance with the state-of-the-art methods on all three datasets.

From the results on IEMOCAP in Table 2, we observe that: **(a)** By utilizing the textual feature only, bcGRU outperforms bcLSTM and CMN trained on the textual feature significantly, attaining +3.3 and +2.8 gain in terms of WA, respectively. bcLSTM$_*$ performs better than bcGRU, and even beats bcLSTM and CMN with the trimodal features in terms of WA. In terms of UWA, CMN performs better than bcLSTM$_*$ only when it is equipped with multimodal features. **(b)** By examining the detailed accuracy in each emotion, bcLSTM$_*$ and bcGRU with the textual feature attain much higher accuracy on the neutral emotion than bcLSTM with the only textual feature while maintaining good performance on the other three emotions. The results show that the weighted loss function benefits the training of models.

From the results on Friends and EmotionPush in Table 3, we observe that bcLSTM$_*$ and bcGRU trained on the same dataset (F+E) of CNN-DCNN perform better than CNN-DCNN on EmotionPush while attaining comparable performance with CNN-DCNN on Friends. The results show that by utilizing the contextual information with the weighted loss function, bcLSTM$_*$ and bcGRU can beat the state-of-the-art method.

(2) HiGRUs vs. Baselines. Our proposed HiGRUs outperform the state-of-the-art methods with significant margins on all the datasets.

From Table 2, we observe that: **(a)** CMN with the trimodal features attains the best performance on the anger emotion while our vanilla HiGRU achieves the best performance on the happiness emotion and gains further improvement on sadness and neutral emotions over CMN. Overall, the vanilla HiGRU achieves at least 8.7% and 3.8% improvement over CMN with the textual feature and the trimodal features in terms of WA, respectively. The results, including those of bcLSTM$_*$ and bcGRU, indicate that GRU learns better representations of utterances than CNN in this task. **(b)** The two variants, HiGRU-f and HiGRU-sf, can further attain +0.9 and +1.5 improvement over HiGRU in terms of WA and +1.0 and +1.2 improvement over HiGRU in terms of UWA, respectively. The results demonstrate that the included individual word/utterance-level features and long-range contextual information in HiGRU-f and HiGRU-sf, are indeed capable of boosting the performance of the vanilla HiGRU.

From Table 3, we can see that: **(a)** In terms of UWA, HiGRU trained and tested on individual sets of Friends and EmotionPush gains at least 7.5% and 6.0% improvement over CNN-DCNN, respectively. Overall, our proposed HiGRU achieves well-balanced performance for the four tested emotions, especially attaining significant better performance on the minority emotions of anger and sadness. **(b)** Moreover, HiGRU-f and HiGRU-sf further improve HiGRU +1.2 accuracy and +1.7 accuracy on Friends and +0.6 accuracy and +1.8 accuracy on EmotionPush in terms of UWA, respectively. The results again demonstrate the superior power of HiGRU-f and HiGRU-sf.

(3) Mixing Training Sets. By examining the results from the last ten rows in Table 3, we conclude that it does not necessarily improve the performance by mixing the two sets of training data.

Though the best performance of SA-BiLSTM

Model	Train	Friends (F)						EmotionPush (E)					
		Ang	**Joy**	**Sad**	**Neu**	**WA**	**UWA**	**Ang**	**Joy**	**Sad**	**Neu**	**WA**	**UWA**
SA-BiLSTM	F+E	49.1	68.8	30.6	**90.1**	-	59.6	24.3	70.5	31.0	**94.2**	-	55.0
CNN-DCNN	F+E	55.3	71.1	55.3	68.3	-	62.5	45.9	76.0	51.7	76.3	-	62.5
bcLSTM$_*$	F(E)	64.7	69.6	48.0	75.6	72.4(4.2)	64.4(1.6)	32.9	69.9	47.1	78.0	74.7(4.4)	57.0(2.1)
bcGRU	F(E)	69.5	65.4	52.9	74.7	71.7(4.7)	65.6(1.2)	33.7	71.1	57.2	76.1	73.9(2.9)	59.5(1.8)
bcLSTM$_*$	F+E	54.5	75.6	43.4	73.0	70.5(4.5)	61.6(1.6)	52.4	79.1	54.7	73.3	73.4(3.8)	64.9(2.1)
bcGRU	F+E	59.0	78.6	42.3	71.4	70.2(5.1)	62.8(1.4)	49.4	74.8	61.9	72.4	72.1(4.3)	64.6(1.8)
HiGRU	F(E)	66.9	73.0	51.8	77.2	**74.4**(1.7)	67.2(0.6)	55.6	78.1	57.4	73.8	73.8(2.0)	66.3(1.7)
HiGRU-f	F(E)	69.1	72.1	**60.4**	72.1	71.3(2.9)	68.4(1.0)	55.9	78.9	60.4	72.4	73.0(2.2)	66.9(1.2)
HiGRU-sf	F(E)	**70.7**	70.9	57.7	76.2	74.0(1.4)	**68.9**(1.5)	57.5	78.4	64.1	72.5	73.0(1.6)	68.1(1.2)
HiGRU	F+E	55.4	81.2	51.4	64.4	65.8(4.2)	63.1(1.5)	50.8	76.9	69.0	75.7	75.3(1.7)	68.1(1.2)
HiGRU-f	F+E	54.9	78.3	55.5	68.7	68.5(3.0)	64.3(1.2)	**58.3**	79.1	**69.6**	70.0	71.5(2.5)	69.2(0.9)
HiGRU-sf	F+E	56.8	**81.4**	52.2	68.7	69.0(2.0)	64.8(1.3)	57.8	**79.3**	66.3	77.4	**77.1**(1.0)	**70.2**(1.1)

Table 3: Experimental results on Friends and EmotionPush. In the Train column, F(E) denotes the model is trained on only one training set, Friends or EmotionPush. F+E means the model is trained on the mixed training set while validated and tested individually. The results of SA-BiLSTM and CNN-DCNN are from (Luo et al., 2018) and (Khosla, 2018), respectively.

d_1	bcGRU	HiGRU	HiGRU-f	HiGRU-sf
-	65.6(1.2)	-	-	-
300	-	67.2(0.6)	68.4(1.0)	68.9(1.5)
200	-	67.6(2.0)	**68.9**(0.9)	69.1(1.3)
150	-	**67.6**(1.5)	68.5(1.3)	68.9(1.2)
100	-	67.5(1.7)	68.4(1.3)	**69.6**(1.0)

Table 4: Experimental results of UWA on Friends by our proposed models with different scales of utterance encoder.

and CNN-DCNN is obtained by training on the mixed dataset, the testing results show that our implemented bcLSTM$_*$, bcGRU and our proposed HiGRU models can attain better performance on EmotionPush but yield worse performance on Friends in terms of UWA.

By examining the detailed emotions, we speculate that: EmotionPush is a highly imbalanced dataset with over 60% of utterances in the neutral emotion. Introducing EmotionPush into a more balanced dataset, Friends, is equivalent to down-sampling the minority emotions in Friends. This hurts the performance on the minority emotions, anger and sadness. Meanwhile, introducing Friends into EmotionPush corresponds to up-sampling the minority emotions in EmotionPush. The performance of the sadness emotion is significantly boosted and that on the anger emotion is at least unaffected.

4.6 Discussions

Model Size. We study how the scale of the utterance encoder affects the performance of our proposed models, especially when our models contain a similar number of parameters as the baseline, say bcGRU. Such a fair condition can be made between our HiGRU-sf and bcGRU if we set d_1 to 150. From the testing results on Friends in Table 4, we can observe that: (1) Under the fair condition, the performance of our HiGRU-sf is not degraded compared to that when $d_1 = 300$. HiGRU-sf still outperforms bcGRU by a significant margin. (2) Overall, no matter d_1 is larger or smaller than 150, HiGRU-sf maintains consistently good performance and the difference between HiGRU-sf and HiGRU-f or HiGRU keeps noticeable. These results further demonstrate the superiority of our proposed models over the baseline bcGRU and the motivation of developing the two variants based on the vanilla HiGRU.

Successful Cases. We investigate three scenes related to the word "okay" that expresses three distinct emotions. The first two scenes come from the testing set of Friends and the third one from that of IEMOCAP. We report the predictions made by bcGUR and our HiGRU-sf, respectively, in Table 5. In **Scene-1**, "okay" with period usually exhibits little emotion and both bcGRU and HiGRU-sf correctly classify it as "Neu". In **Scene-2**, "okay" with "!" expresses strong emotion. However, bcGRU misclassifies it to "Ang" while HiGRU-sf successfully recognizes it as "Joy". Actually, the mistake can be traced back to the first utterance of this scene which is also misclassified as "Ang". This indicates that bcGRU tends to capture the wrong atmosphere within the dialogue. As for **Scene-3**, "okay" with period now indicates "Sad" and is correctly recognized by HiGRU-sf but misclassified as "Neu" by bcGRU. Note that HiGRU-sf also classifies the third utterance in **Scene-3** as "Sad" which seems to be conflicting

Role	Utterance	Truth	bcGRU	HiGRU-sf
Scene-1				
Phoebe	Okay. Oh but don't tell them Monica's pregnant because they frown on that.	Neu	Neu	Neu
Rachel	Okay.	Neu	Neu	Neu
Phoebe	Okay.	Neu	Neu	Neu
Scene-2				
Phoebe	Yeah! Sure! Yep! Oh, y'know what? If I heard a shot right now, I'd throw my body on you.	Joy	Ang	Joy
Gary	Oh yeah? Well maybe you and I should take a walk through a bad neighborhood.	Other	/	/
Phoebe	Okay!	Joy	Ang	Joy
Gary	All right.	Neu	Neu	Neu
Scene-3				
Female	Can I send you, like videos and stuff? What about when they start walking.	Other	/	/
Male	Yeah yeah yeah.	Sad	Hap	Sad
Male	You you record every second. You record every second because I want to see it all. Okay?	Hap	Hap	Sad
Male	If I don't get to see it now, I get to see it later at least, you know? You've got to keep it all for me; all right?	Other	/	/
Female	Okay.	Sad	Neu	Sad

Table 5: "Okay" expresses distinct emotions in three different scenes.

Role	Utterance	Truth	bcGRU	HiGRU-sf
Scene-4				
Ross	Hi.	Neu	Neu	Neu
Rachel	Hi.	Neu	Neu	Neu
Ross	Guess what?	Neu	Neu	Neu
Rachel	What?	Neu	Neu	Neu
Ross	They published my paper.	Joy	Sad	Neu
Rachel	Oh, really, let me see, let me see.	Joy	Neu	Neu
Phoebe	Rach, look! Oh, hi! Where is my strong Ross Skywalker to come rescue me. There he is.	Other	/	/
Scene-5				
Speaker-1	Sorry for keeping you up	Sad	Sad	Sad
Speaker-2	Lol don't be	Joy	Joy	Joy
Speaker-2	I didn't have to get up today	Neu	Sad	Sad
Speaker-1	:p	Joy	Joy	Joy
Speaker-2	It's actually been a really lax day	Joy	Neu	Sad

Table 6: Wrong predictions made by both bcGRU and our HiGRU-sf in two scenes.

to the ground truth. In fact, our HiGRU-sf captures the blues of this parting situation, where the true label "Hap" may not be that suitable. These results show that our HiGRU-sf learns from both each utterance and the context, and can make correct predictions of the emotion of each utterance.

Failed Cases. At last, we show some examples that both bcGRU and our HiGRU-sf fail in recognizing the right emotions in Table 6, i.e., **Scene-4** from Friends and **Scene-5** from EmotionPush. In **Scene-4**, both bcGRU and HiGRU-sf make wrong predictions for the fifth and the sixth utterances. It should be good news that Ross has his paper published and Rachel is glad to see related reports about it. However, the transcripts do not reveal very strong emotions compared to what the characters might act in the TV show. This kind of scenes may be addressed by incorporating some other features like audio and video. As for **Scene-5**, the third and the fifth utterances are classified into wrong emotions. Notice that the emotions indicated from the two utterances are very subtle even for humans. The Speaker-2 did not plan to get up today, but Speaker-1 kept him/her up and it ended up with a really lax day. So, the Speaker-2 feels joyful now. This indicates that even taking into the context into account, the models' capability of understanding subtle emotions is still limited and more exploration is required.

5 Conclusion

We propose a hierarchical Gated Recurrent Unit (HiGRU) framework to tackle the utterance-level emotion recognition in dialogue systems, where the individual utterance embeddings are learned by the lower-level GRU and the contexts of utterances are captured by the upper-level GRU. We promote the HiGRU framework to two variants, HiGRU-f, and HiGRU-sf, and effectively capture the word/utterance-level inputs and the long-range contextual information, respectively. Experimental results demonstrate that our proposed HiGRU models can well handle the data imbalance issue and sufficiently capture the available text information, yielding significant performance boosting on all three tested datasets. In the future, we plan to explore semi-supervised learning methods to address the problem of data scarcity in this task.

Acknowledgments

This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund, and Project No. UGC/IDS14/16), and Meitu (No. 7010445). We thank the three anonymous reviewers for the insightful suggestions on various aspects of this work.

References

Muhammad Abdul-Mageed and Lyle H. Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *ACL*, pages 718–728.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *EMNLP*, pages 1650–1659.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Paul Ekman. 1971. *Universal and cultural differences in facial expressions of emotion*. Lincoln: University of Nebraska Press.

Paul Ekman. 1992. Are there basic emotions? *Psychological Review*, 99(3):550–553.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *EACL*, pages 427–431.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. In *NAACL-HLT*, pages 2122–2132.

Chao-Chun Hsu and Lun-Wei Ku. 2018. Socialnlp 2018 emotionx challenge overview: Recognizing emotions in dialogues. In *SocialNLP@ACL'18*, pages 27–31.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*, pages 4099–4106.

Sopan Khosla. 2018. Emotionx-ar: CNN-DCNN autoencoder based emotion classifier. In *SocialNLP@ACL'18*, pages 37–44.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, and Sachindra Joshi. 2018. Dialogue act sequence labeling using hierarchical encoder with CRF. In *AAAI*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using context information for dialog act classification in DNN framework. In *EMNLP*, pages 2170–2178.

Linkai Luo, Haiqing Yang, and Francis Y. L. Chin. 2018. Emotionx-dlc: Self-attentive BiLSTM for detecting sequential emotions in dialogues. In *SocialNLP@ACL'18*, pages 32–36.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113.

Zhao Meng, Lili Mou, and Zhi Jin. 2017. Hierarchical RNN with static sentence-level attention for text-based speaker change detection. In *CIKM*, pages 2203–2206.

David Olson. 1977. From utterance to text: The bias of language in speech and writing. *Harvard educational review*, 47(3):257–281.

Soujanya Poria, Erik Cambria, and Alexander F. Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP*, pages 2539–2544.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. 2017. Context-dependent sentiment analysis in user-generated videos. In *ACL*, pages 873–883.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? finding strong and weak opinion clauses. In *AAAI*, pages 761–769.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *WI*, pages 275–278.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT*, pages 1480–1489.

Sayyed M. Zahiri and Jinho D. Choi. 2018. Emotion detection on TV show transcripts with sequence-based convolutional neural networks. In *AAAI*, pages 44–52.

Learning Interpretable Negation Rules via Weak Supervision at Document Level: A Reinforcement Learning Approach

Nicolas Pröllochs
Oxford-Man Institute
University of Oxford
nicolas.prollochs@eng.ox.ac.uk

Stefan Feuerriegel
ETH Zurich
sfeuerriegel@ethz.ch

Dirk Neumann
University of Freiburg
dirk.neumann@is.uni-freiburg.de

Abstract

Negation scope detection is widely performed as a supervised learning task which relies upon negation labels at word level. This suffers from two key drawbacks: (1) such granular annotations are costly and (2) highly subjective, since, due to the absence of explicit linguistic resolution rules, human annotators often disagree in the perceived negation scopes. To the best of our knowledge, our work presents the first approach that eliminates the need for word-level *negation* labels, replacing it instead with document-level *sentiment* annotations. For this, we present a novel strategy for learning fully interpretable negation rules via weak supervision: we apply reinforcement learning to find a policy that reconstructs negation rules from sentiment predictions at document level. Our experiments demonstrate that our approach for weak supervision can effectively learn negation rules. Furthermore, an out-of-sample evaluation via sentiment analysis reveals consistent improvements (of up to 4.66 %) over both a sentiment analysis with (i) no negation handling and (ii) the use of word-level annotations from humans. Moreover, the inferred negation rules are fully interpretable.

1 Introduction

Negations are a frequently utilized linguistic tool for expressing disapproval or framing negative content with positive words. Neglecting negations can lead to false attributions (Morante et al., 2008) and, moreover, impair accuracy when analyzing natural language; e. g., in information retrieval (Cruz Díaz et al., 2012; Rokach et al., 2008) and especially in sentiment analysis (Cruz et al., 2015; Wiegand et al., 2010). Hence, even simple heuristics for identifying negation scopes can yield substantial improvements in such cases (Jia et al., 2009).

Negation scope detection is sometimes implemented as unsupervised learning (e. g., Pröllochs et al., 2016), while a better performance is commonly achieved via supervised learning (see our supplements for a detailed overview): the resulting models thus learn to identify negation scopes from word-level annotations (e. g., Li and Lu, 2018; Reitan et al., 2015). We argue that this approach suffers from inherent drawbacks. (1) Such granular annotations are costly and, especially at word level, a considerable number of them is needed. (2) Negation scope detection is highly subjective (Councill et al., 2010). Due to the absence of explicit linguistic rules for resolutions, existing corpora often come with annotation guidelines (Morante and Blanco, 2012; Morante and Daelemans, 2012). Yet there are considerable differences: some corpora were labeled in a way that negation scopes consist of single text spans, while others allowed disjoint spans (Fancellu et al., 2017). More importantly, given the absence of universal rules, human annotators largely disagree in their perception of what words should be labeled as negated.

Motivational experiment. Since prevalent corpora were labeled only by a single rater, we now establish the severity of between-rater discrepancies. For this, we carried out an initial analysis of 500 sentences from movie reviews.[1] Each sentence contained at least one explicit negation phrase from the list of Jia et al. (2009), such as *"not"* or *"no."* Two human raters were then asked

---

[1] Details are reported in our supplementary materials.

Proceedings of NAACL-HLT 2019, pages 407–413
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

to annotate negation scopes. They could choose an arbitrary selection of words and were not restricted to a single subspan, as recommended by Fancellu et al. (2017). The annotations resulted in large differences: only 50.20 % of the words were simultaneously labeled as “negated” by both raters. Based on this experimental evidence, we showcase there is no universal definition of negation scopes (rather, human annotations are likely to be noisy or even error-prone) and thus highlight the need for further research.

Contributions. To the best of our knowledge, our work presents the first approach that eliminates the need for *word-level* annotations of *negation labels*. Instead, we perform negation scope detection merely by utilizing shallow annotations at *document level* in the form of *sentiment labels* (e. g., from user reviews). Our novel strategy learns interpretable negation rules via weak supervision: we apply reinforcement learning to find a policy that reconstructs negation rules based on sentiment prediction at document level (as opposed to conventional word-level annotations).

In our approach, a single document d comes with a sentiment label y_d. The document consists of N_d words, $w_{d,1}, \ldots, w_{d,N_d}$, where the number of words can easily surpass several hundreds. Based on the sentiment value, we then need to make a decision (especially out-of-sample) for each of the N_d words, whether or not it should be negated. In this case, a *single* sentiment value is outnumbered by potentially *hundreds* of negation decisions, thus pinpointing to the difficulty of this task. Formally, the goal is to learn individual labels $a_{d,i} \in \{\text{Negated}, \neg\text{Negated}\}$ for each word $w_{d,i}$. Rewards are the errors in sentiment prediction at document level.

Strengths. Our approach exhibits several favorable features that overcome shortcomings found in prior works. Among them, it eliminates the need for manual word-level labels. It thus avoids the detrimental influence of subjectivity and misinterpretation. Instead, our model is solely trained on a document-level variable and can thus learn domain-specific particularities of the given prose. The inferred negation rules are fully interpretable while documents can contain multiple instances of negations with arbitrary complexity, sometimes nested or consisting out of disjoint text spans. Despite facing several times more negation decisions than sentiment labels, our experiments demonstrate that this problem can be effectively learned through reinforcement learning.

Evaluation. Given the considerable inconsistencies in human annotations of negation scopes and the lack of universal rules, we regard the “true” negation scopes as unobservable. Hence, we later compare the identified negation scopes with those from rater 1 and 2 only as a sensitivity check because of the fact that both raters have only 50.2 % overlap. Instead, we choose the following evaluation strategy. We concentrate on the performance of negation scope detection as a supporting tool in natural language processing where its primary role is to facilitate more complex learning tasks such as sentiment analysis. Therefore, we report the performance improvements in sentiment analysis resulting from our approach. For a fair comparison, we use baselines that only rely upon the same information as our weak supervision (and thus have no access to word-level negation labels). Our performance is even on par with a supervised classifier that can exploit richer labels during training.

2 Learning Negation Scope Detection via Weak Supervision

Intuition. The choice of reinforcement learning for weak supervision might not be obvious at first, but, in fact, it is informed by theory: it imitates the human reading process as stipulated by cognitive reading theory (Just and Carpenter, 1980), where readers iteratively process information word-by-word.

States and actions. In each learning iteration, the reinforcement learning agent observes the current state $s_i = (w_i, a_{i-1})$ that we engineer as the combination of the i-th word w_i in a document and the previous action a_{i-1}. This specification establishes a *recurrent* architecture whereby the previous negation can pass on to the next word. At the same time, this allows for nested negations, as a word can first introduce a negation scope and another subsequent negation can potentially revert it.

After observing the current state, the agent chooses an action a_t from of two possibilities: (1) it can set the current word to *negated* or (2) it can mark it as *not negated*. Hence, we obtain the following set of possible actions $A = \{\text{Negated}, \neg\text{Negated}\}$. Based on the selected action, the agent receives a reward, r_i which updates the knowledge in the state-action function

$Q(s_i, a_i)$. This state-action function is then used to infer the best possible action a_i in each state s_i, i. e., the optimal policy $\pi^*(s_i, a_i)$.

Reward function. The reward r_i depends upon the correlation between a given a document-level label (e. g., a rating in movie reviews) and the sentiment of a document. We predict the sentiment S_d of document d using a widely-used sentiment routine based on the occurrences of positively- and negatively-opinionated terms (see Taboada et al., 2011). If a term is negated by the policy, the polarity of the corresponding term is inverted, i. e., positively opinionated terms are counted as negative and vice versa. In the following, S_d^0 denotes the document sentiment without considering negations; S_d^π the sentiment when incorporating negations based on policy π.

When processing a document, we cannot actually compute the reward until we have processed all words. Therefore, we set the reward before the last word to $c \approx 0$, i. e., $r_i = c$ for $i = 1, \ldots, N_d - 1$. For the final word, the agent compares its performance in predicting the document label based on sentiment without considering negations S_d^0 to the sentiment when incorporating negations based on the current policy π^*. The former is defined by the absolute difference between the document label y_d and the predicted sentiment without negations S_d^0, whereas the latter is defined by the absolute difference between y_d and the adjusted sentiment using the current policy S_d^π. Then the difference between these values returns the terminal reward r_{N_d}. Thus the reward is

$$r_i = \begin{cases} 0, & \text{if } a_i = \text{Neg and } i < N_d, \\ c, & \text{if } a_i = \neg\text{Neg and } i < N_d, \\ \left|y_d - S_d^0\right| - \left|y_d - S_d^\pi\right|, & \text{if } i = N_d, \end{cases}$$

with a constant c (we use $c = 0.005$) that adds a small reward for default (i. e., non-negating) actions to avoid overfitting.

***Q*-learning.** During the learning process[2], the agent then successively observes a sequence of words in which it can select between exploring new actions or taking the current optimal one. This choice is made by ε-greedy selection according to which the agent explores the environment by selecting a random action with probability ε or, alternatively, exploits the current knowledge with probability $1 - \varepsilon$.

3 Experiments

Datasets. We use the following benchmark datasets with document-level annotations from the literature (cf. Hogenboom et al., 2011; Pröllochs et al., 2016; Wiegand et al., 2010):

IMDb: movie reviews from the Internet Movie Database archive, each annotated with an overall rating at document level (Pang and Lee, 2005).

Airport: user reviews of airports from Skytrax, each annotated with an overall rating at document level (Pérezgonzález and Gilbey, 2011).

Ad hoc: financial announcements with complex, domain-specific language (Pröllochs et al., 2016), labeled with the daily abnormal return of the corresponding stock.

Learning parameters. We perform 4000 learning iterations with a higher exploration rate as given by the following parameters[3]: exploration $\varepsilon = 0.001$, discount factor $\gamma = 0$ and learning rate $\alpha = 0.005$. In a second phase, we run 1000 iterations for fine-tuning with exploration $\varepsilon = 0.0001$, discount factor $\gamma = 0$ and learning rate $\alpha = 0.001$.

Policy learning. For each dataset, the reinforcement learning process converges to a stationary policy that shows reward fluctuations below 0.05 %. As part of a benchmark, we study the mean squared error (MSE) between y_d and the predicted sentiment S_d^0 when leaving negations untreated as our benchmark. For all datasets, the in-sample MSE decreases substantially (see Figure 1), demonstrating the effectiveness of our learning approach. The reductions number to 14.93 % (IMDb), 16.77 % (airport), and 0.91 % (ad hoc). The latter is a result of the considerably more complex language in financial statements.

Performance in Sentiment Analysis. We use 10-fold cross validation to compare the out-of-sample performance in sentiment analysis of reinforcement learning to benchmarks without word-level labels from previous works. The benchmarks consists of rules (Hogenboom et al., 2011; Taboada et al., 2011), which search for the occurrence of specific cues based on pre-defined lists and then invert the meaning of a fixed number of surrounding words. Table 1 compares the out-of-sample MSE between predicted sentiment and the

---

[2] We use Watkin's $Q(\lambda)$ with eligibility traces; see Sutton and Barto (1998) for details. At the beginning, we initialize the action-value function $Q(s, a)$ to zero for all states and actions. This also controls our default action when encountering unknown states or out-of-vocabulary (OOV) words. In such cases, the non-negated action is preferred.

[3] Further details regarding the learning parameters are provided in the supplementary materials.

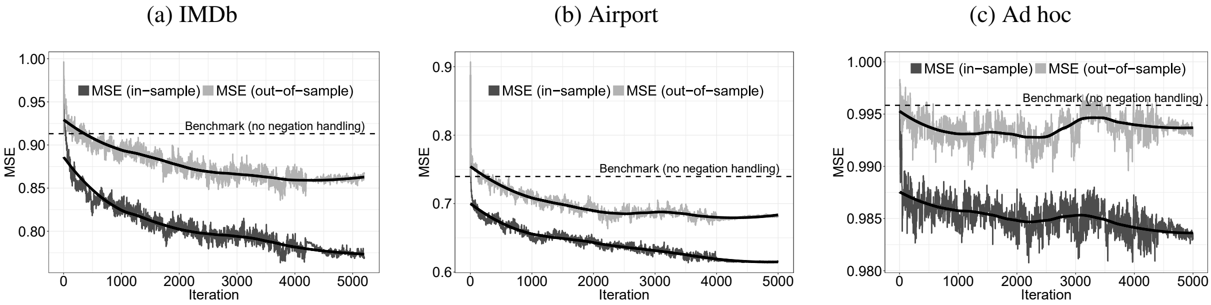

(a) IMDb (b) Airport (c) Ad hoc

Figure 1: MSE between the document label and predicted sentiment across different learning iterations using 10-fold cross validation. Additional lines in black from smoothing.

document-level label:[4]

IMDb: Negating a fixed window of the next 4 words achieves the lowest error among all rules, similar to Dadvar et al. (2011). This rule reduces the MSE of the benchmark with no negation handling by 1.05 %. Our approach works even more accurately, and dominates all of the rules, reducing the out-of-sample MSE by at least 4.60 %.

Airport: Our method decreases the MSE by 4.66 % compared to the best-performing rule (negating a fixed window of the next 4 words).

Ad hoc: Even for complex financial language, reinforcement learning exceeds this benchmark method by 0.19 % in terms of out-of-sample MSE.

Altogether, our weak supervision improves sentiment analysis consistently across all datasets.[5]

Approach	IMDb	Airport	Ad hoc
Benchmark: no negation handling	0.9133	0.7415	0.9958
Negating all subsequent words	0.9160	0.7312	0.9949
Negating the whole sentence	0.9339	0.7811	0.9961
Fixed window of 1 word	0.9082	0.7212	0.9950
Fixed window of 2 words	0.9052	0.7130	0.9943
Fixed window of 3 words	0.9047	0.7146	0.9943
Fixed window of 4 words	0.9038	0.7134	0.9942
Fixed window of 5 words	0.9039	0.7136	0.9942
Proposed reinforcement learning for weak supervision	**0.8622**	**0.6798**	**0.9923**

Table 1: Out-of-sample MSE between sentiment S_d^π and the document label y_d. Lowest error in bold.

Comparison to human raters. For reasons of completeness, our supplements report the overlap with both human raters from our motivational experiment, which is in the range of 18.8 % to 25.2 %. However, these numbers should be treated with caution, as we remind that there is no universal definition of negation scopes and even the two human annotations reveal on 50.2 %. Moreover, our approach was not learned towards reconstructing these human annotations, since we focused on rules that achieve the greatest benefit in sentiment analysis.

Comparison to word-level classifiers. We also compared weak supervision against a supervised HMM classifier from Pröllochs et al. (2016) that draws upon granular word-level negation labels. Here we report the sentiment analysis on IMDb in order to be able to use the domain-specific negation labels from IMDb text snippets of our initial experiment. In comparison to our reinforcement learning, the supervised classifier results in a 5.79 % higher (and thus inferior) MSE. Yet our weak supervision circumvents costly word-level annotations.

Interpretability. Our method yields negation rules that are fully interpretable: one simply has to assess the state-action function $Q(s_i, a_i)$. Table 2 provides an example excerpt for the document *"this beautiful movie isn't good but fantastic."* The agent the starts by observing the first state given by the combination of the first word w_1 and the previous action a_0, i. e. $s_1 = (\textit{this}, \neg\text{Negated})$. According to the state-action table, the best action for the agent is to set this state to not negated ($a_1 = \neg\text{Negated}$). This pattern continues until it observes state $s_4 = (\textit{isn't}, \neg\text{Negated})$ in which the policy implies to initiate a negation scope ($a_4 = \text{Negated}$). Subsequently, the negation scope is forwarded until the agent observes $s_6 = (\textit{but}, \text{Negated})$ in which it terminates the negation scope ($a_6 = \neg\text{Negated}$). Finally, the agent observes $s_7 = (\textit{fantastic}, \neg\text{Negated})$ in which it takes action $a_7 = \neg\text{Negated}$.

---

[4] We also experimented with performance comparisons in a classification task, yet our approach also yields consistent improvements in this evaluation.

[5] We also investigated the relationship between prediction performance and text length, finding only minor effects.

State=(w_i, a_{i-1})	Negated	¬Negated	$\pi^*(s_i, a_i)$
(*this*, ¬Negated)	0.0114	0.0502	¬Negated
(*beautiful*, ¬Negated)	0.0081	0.0779	¬Negated
(*movie*, ¬Negated)	0.0039	0.0506	¬Negated
(*isn't*, ¬Negated)	0.0700	0.0456	Negated
(*good*, Negated)	0.0578	0.0322	Negated
(*but*, Negated)	0.0120	0.0365	¬Negated
(*fantastic*, ¬Negated)	–0.0181	0.1708	¬Negated

Table 2: Excerpt of state-action function $Q(s_i, a_i)$ actions $A = \{\text{Negated}, \neg\text{Negated}\}$ and the learned policy π^* for IMDb reviews.

4 Related Work

State-of-the-art methods for detecting, handling and interpreting negations can be grouped into different categories (cf. Pröllochs et al., 2015, 2016; Rokach et al., 2008).

Rule-based approaches are among the most common due to their ease of implementation and solid out-of-the-box performance. These usually suppose a forward influence of negation cues based on which they invert the meaning of the whole sentence or a fixed number of subsequent words (Hogenboom et al., 2011). Furthermore, they can also incorporate syntactic information in order to imitate subject and object (Padmaja et al., 2014; Chowdhury and Lavelli, 2013). Negation rules have been found to work effectively across different domains and rarely need fine-tuning (Taboada et al., 2011). However, rule-based approaches entail several drawbacks, as the list of negations must be pre-defined and the selection criterion according to which rule a rule is chosen is usually random or determined via cross validation. In addition, rules cannot effectively cope with implicit expressions or particular, domain-specific characteristics.

Generative probabilistic models (e. g., hidden Markov models or conditional random fields) can partially overcome these shortcomings (Li and Lu, 2018; Reitan et al., 2015; Rokach et al., 2008), such as the difficulty of recognizing implicit negations. These process narrative language word-by-word and move between hidden states representing negated and non-negated parts. Such models can adapt to domain-specific language, but require more computational resources and rely upon ex ante transition probabilities. Although variants based on unsupervised learning avoid the need for any labels, practical applications reveal inferior performance compared to supervised approaches (Pröllochs et al., 2015). The latter usually depend on manual labels at a granular level, which are not only costly but suffer from subjective interpretations (Fancellu et al., 2017).

A third category of methods links the polarity shift effect of negations more closely to sentiment analysis tasks at sentence or document level. For example, text parts can be classified into a polarity-unshifted part and a polarity-shifted part according to certain rules (Li and Huang, 2009). Sentiment classification models are then trained using both parts (Li et al., 2010). Alternatively, rule-based algorithms can extract sentences with inconsistent sentiment and omit them from standard sentiment analysis procedures (Orimaye et al., 2012). Reversely, antonym dictionaries have been used to generate sentiment-inverted texts to classify polarity in pairs (Xia et al., 2016). Although such data expansion techniques usually enhance the performance of sentiment analysis, they require either complex linguistic knowledge or extra human annotations (Xia et al., 2015).

Research gap. In contrast to these methods, we propose a novel strategy for learning negation rules via weak supervision. Our model uses reinforcement learning to reconstruct negation rules based on an document-level variable and does not require any kind of manual word-level labeling or precoded linguistic patterns. It is able to recognize explicit as well as implicit negations, while avoiding the influence of subjective interpretations.

5 Conclusion

This paper proposes the first approach for negation scope detection based on weak supervision. Our proposed reinforcement learning strategy circumvents the need for word-level annotations with negation scopes, as it reconstructs negation rules based on a document-level sentiment labels. Our experiments show that our weak supervision is effective in negation scope detection; it yields consistent improvements (of up to 4.66 %) over a sentiment analysis without negation handling.

Our works suggests important implications. We are in line with growing literature (e. g., Fancellu et al., 2017) that reports challenges in resolving negation scopes through humans. Beyond prior works, our experiment reveals between-rater inconsistencies. While negation scope detection is widely studied as an isolated task, it could be beneficial when linking its evaluation more closely to context-specific uses such as sentiment analysis.

References

Md Faisal Mahbub Chowdhury and Alberto Lavelli. 2013. Exploiting the scope of negations and heterogeneous features for relation extraction: A case study for drug-drug interaction extraction. In *NAACL-HLT*, pages 765–771.

Isaac G. Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. In *Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59. ACL.

Noa P. Cruz, Maite Taboada, and Ruslan Mitkov. 2015. A machine–learning approach to negation and speculation detection for sentiment analysis. *Journal of the Association for Information Science and Technology*, 67(9):2118–2136.

Noa P. Cruz Díaz, Maña López, Manuel J., Jacinto Mata Vázquez, and Victoria Pachón Álvarez. 2012. A machine-learning approach to negation and speculation detection in clinical texts. *Journal of the American Society for Information Science and Technology*, 63(7):1398–1410.

Maral Dadvar, Claudia Hauff, and Franciska de Jong. 2011. Scope of negation detection in sentiment analysis. In *Dutch-Belgian Information Retrieval Workshop*, pages 16–20.

Federico Fancellu, Adam Lopez, Bonnie Webber, and Hangfeng He. 2017. Detecting negation scope is easy, except when it isn't. In *Conference of the European Chapter of the ACL*, pages 58–63. ACL.

Alexander Hogenboom, Paul van Iterson, Bas Heerschop, Flavius Frasincar, and Uzay Kaymak. 2011. Determining negation scope and strength in sentiment analysis. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2589–2594.

Lifeng Jia, Clement Yu, and Weiyi Meng. 2009. The effect of negation on sentiment analysis and retrieval effectiveness. In *CIKM*, pages 1827–1830.

Marcel A Just and Patricia A Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329.

Hao Li and Wei Lu. 2018. Learning with structured representations for negation scope extraction. In *Proceedings of the ACL*, pages 533–539.

Shoushan Li and Chu-Ren Huang. 2009. Sentiment classification considering negation and contrast transition. In *Pacific Asia Conference on Language, Information and Computation*, pages 297–306. ACL.

Shoushan Li, Sophia Yat Mei Lee, Ying Chen, Chu-Ren Huang, and Guodong Zhou. 2010. Sentiment classification and polarity shifting. In *International Conference on Computational Linguistics*, pages 635–643. ACL.

Roser Morante and Eduardo Blanco. 2012. Sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (SemEval '12)*, pages 265–274.

Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul.*

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the scope of negation in biomedical texts. In *EMNLP*, pages 715–724.

Sylvester Olubolu Orimaye, Saadat M Alhashmi, and Eu-Gene Siew. 2012. Buy it-don't buy it: sentiment classification on amazon reviews using sentence polarity shift. In *Pacific Rim International Conference on Artificial Intelligence*, pages 386–399. Springer.

S. Padmaja, Sameen Fatima, and Sasidhar Bandu. 2014. Evaluating sentiment analysis methods and identifying scope of negation in newspaper articles. *International Journal of Advanced Research in Artificial Intelligence*, 3(11):1–6.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124.

Jose D. Pérezgonzález and Andrew Gilbey. 2011. Predicting skytrax airport rankings from customer reviews. *Journal of Airport Management*, 5(4):335–339.

Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. 2015. Enhancing sentiment analysis of financial news by detecting negation scopes. In *HICSS*, pages 959–968.

Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. 2016. Negation scope detection in sentiment analysis: Decision support for news-driven trading. *Decision Support Systems*, 88:67–75.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108.

Lior Rokach, Roni Romano, and Oded Maimon. 2008. Negation recognition in medical narrative reports. *Information Retrieval*, 11(6):499–538.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68. ACL.

Rui Xia, Feng Xu, Jianfei Yu, Yong Qi, and Erik Cambria. 2016. Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis. *Information Processing & Management*, 52(1):36–45.

Rui Xia, Feng Xu, Chengqing Zong, Qianmu Li, Yong Qi, and Tao Li. 2015. Dual sentiment analysis: Considering two sides of one review. *Transactions on Knowledge and Data Engineering*, 27(8):2120–2133.

Simplified Neural Unsupervised Domain Adaptation

Timothy Miller
Computational Health Informatics Program
Boston Children's Hospital
Harvard Medical School
`timothy.miller@childrens.harvard.edu`

Abstract

Unsupervised domain adaptation (UDA) is the task of modifying a statistical model trained on labeled data from a source domain to achieve better performance on data from a target domain, with access to only unlabeled data in the target domain. Existing state-of-the-art UDA approaches use neural networks to learn representations that can predict the values of subset of important features called "pivot features." In this work, we show that it is possible to improve on these methods by jointly training the representation learner with the task learner, and examine the importance of existing pivot selection methods.

1 Introduction

Unsupervised domain adaptation (UDA) is the task of modifying a statistical model trained on labeled data from a source domain to achieve better performance on data from a target domain, without access to any labeled data in the target domain. Supervised domain adaptation methods can obtain excellent performance from a small number of labeled examples in the target domain (Daumé III, 2007), but UDA is attractive in cases where annotation requires specialized expertise or the number of meaningfully different sub-domains is large (e.g., both are true for clinical NLP).

Structural correspondence learning (Blitzer et al., 2006) (SCL) is one widely-used method for UDA in natural language processing. The key idea in SCL is that a subset of features, believed to be predictive across domains, are selected as *pivot features*. For each selected pivot feature, SCL creates an auxiliary classification task of predicting the value of that feature in an instance, given the values of all the non-pivot features for that instance. The auxiliary classifiers therefore learn important cross-domain information about the structure of the feature space, which the SCL algorithm uses to create an augmented representation that aligns features from different domains (further details in Section 2).

Meanwhile, recent advances in neural network learning have shown that training regimens that jointly consider evidence from multiple sources can improve performance – both multi-task learning (Søgaard and Goldberg, 2016) and fine tuning (Howard and Ruder, 2018; Devlin et al., 2018). However, existing SCL-based methods treat the representation learning and task learning as separate tasks, so the parameters of the representation learning machinery are fixed before training for the downstream task. Jointly learning the representation- and task-specific parameters can potentially allow a learning algorithm to find representations that are better suited for the task.

In this work, we describe a new UDA algorithm that is trained to jointly maximize two objectives: the primary supervised task in the source domain, and a pivot feature reconstruction task that can be trained on unlabeled data. We also explore the importance of pivot feature selection to this algorithm, in experiments that quantitatively and qualitatively examine the quality of existing pivot selection methods. We find that our joint neural approach to SCL improves unsupervised domain adaptation substantially on a standard sentiment classification task. Our results also show that while existing pivot selection methods perform well, they are below an oracle-provided ceiling for many source-target pairs for the sentiment classification task we examine.

2 Background

This work builds off of existing work in unsupervised domain adaptation, starting with Blitzer's work on structural correspondence learning (SCL) (Blitzer et al., 2006, 2007). In the UDA

Proceedings of NAACL-HLT 2019, pages 414–419
Minneapolis, Minnesota, June 2 - June 7, 2019.

task setup, one is given two datasets, the source $D_S = \{X_s, y_s\}$, with labels for each instance, and the target $D_T = \{X_t\}$, with unlabeled instances only. The goal of UDA is to learn a function $f_u(X_s, y_s, X_t)$ that improves on the classification performance over a function $f_l(X_s, y_s)$ when applied to new data drawn from the target distribution.

SCL is essentially a representation learning algorithm that works by creating a number of auxiliary classification tasks from the unlabeled source and target training instances (inspired by Ando and Zhang 2005). First, a set of p *pivot features* are selected, intended (in Blitzer's words) to be "features which behave in the same way for discriminative learning in both domains." Then, SCL creates p auxiliary tasks of predicting the value of pivot features in an instance given the non-pivot features in the instance. The weights of these linear classifiers are then aligned as columns in a matrix W, and the k left singular vectors are chosen from the singular value decomposition $W = U\Sigma V^\top$ to reduce its dimensionality, leading to a projection matrix $\theta \in R^{n\times k}$ that maps instances from the original feature space into the learned space. Most practical implementations find the best performance of SCL is obtained when projected features are concatenated with the original feature space; for some tasks and datasets other combinations have been tested and proved superior (Sapkota et al., 2016).

Recently, neural-network-based domain adaptation algorithms have been successful, including domain adversarial methods (Ganin et al., 2016) and auto-encoder-based methods (Glorot et al., 2011; Chen et al., 2014). However, a neural version of SCL still obtains near state-of-the-art performance (Ziser and Reichart, 2017). In that work, the AE-SCL system uses a multi-layer perceptron to replace the SVD for learning the feature projection. This network takes non-pivot features as input, has one hidden layer, and predicts the value of the pivot features at the output layer. Since it obtains supervision from the values of features, it can be trained on unlabeled instances from the source and target domains. To train for the downstream sentiment classification task, the source instances are first passed into the trained representation learning network, and the values of the hidden layer are considered an additional set of features. These features are combined with all the original features, and the authors use a logistic regression classifier for the final sentiment classifier.

One standard corpus used to develop new domain adaptation algorithms is the Amazon sentiment analysis dataset.[1] This corpus was created by Blitzer et al. (2007), but we use the version included in the software release from Ziser and Reichart (2017)[2], along with their pre-processing steps, for ease of comparison with their results. This dataset contains reviews from four product categories on Amazon.com – books, DVDs, electronics, and kitchen appliances. Reviews are mapped to binary categories: *positive* if the review assigns the product >3 stars (out of 5) and *negative* if it assigns the product <3 stars. This dataset also contains additional unlabeled instances for each category, used for training the pivot predictor.

3 Methods

The current work has two motivating factors. First, we would like to improve the performance of SCL using joint training. Existing SCL-based methods are successful in treating pivot prediction as a pre-training phase, but joint training may improve UDA by allowing the network to find representations that are equally good at pivot reconstruction but better for downstream task performance. Second, we would like to evaluate the quality of pivot selection methods and explore whether this step might be eliminated to simplify SCL.

We focus on feature-based UDA methods, as opposed to approaches that rely on embeddings (e.g., Barnes et al., 2018; Ziser and Reichart, 2018), since our primary interest is in improving existing models developed with a feature engineering approach. Such methods allow us to quickly adapt a number of different models to new datasets (e.g., for already-existing NLP pipeline software), rather than engineering new neural models from scratch for each of the pipeline tasks. For that reason, we compare to the AE-SCL model of Ziser and Reichart, rather than their subsequent models that take embeddings as input. In any case, we show that with some tuning the AE-SCL model can obtain state-of-the-art performance for many pairs.

---

[1] `https://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

[2] `https://github.com/yftah89/Neural-SCL-Domain-Adaptation`

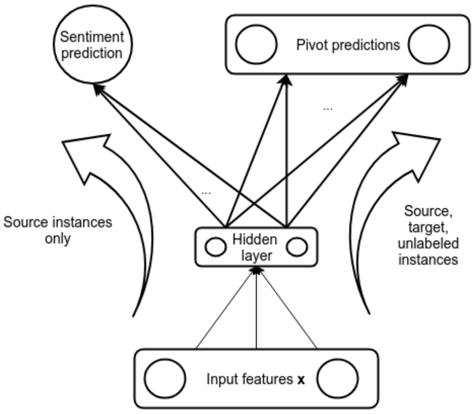


Figure 1: Network architecture for unsupervised domain adaptation.

3.1 Joint Neural Structural Correspondence Learning

Figure 1 graphically depicts our proposed joint model. The input to the model $\mathbf{x} \in \mathcal{R}^n$ is the set of all features extracted from the text – to compare with Ziser and Reichart (2017) we use unigrams and bigrams, extracted using scikit-learn (Pedregosa et al., 2011). We experimented with a few different hidden layer sizes, and settled on $d = 2000$ – this balances the need of the representation to predict more output variables than the AE-SCL method with run-time constraints. The representation is generated with $h(\mathbf{x}) = ReLU(W_h\mathbf{x})$, for $W_h \in \mathcal{R}^{d\times n}$. The task prediction is $f_{task}(\mathbf{x}) = Sigmoid(W_t h(\mathbf{x}))$ ($W_t \in \mathcal{R}^{1\times d}$) and the pivot prediction is $f_{pivot}(\mathbf{x}) = Sigmoid(W_p h(\mathbf{x}))$ ($W_p \in \mathcal{R}^{p\times d}$).

The joint loss function for labeled source data D_l, all data D_a, and model parameters θ is:

$$\begin{aligned}\mathcal{L}(D;\theta) = & \sum_{\mathbf{x},y\in D_l} BCE(f_{task}(\mathbf{x}), y) + \\ & \lambda \sum_{\mathbf{x}\in D_a} BCE(f_{pivot}(\mathbf{x}), pivots(\mathbf{x})) + \\ & \rho\mathcal{R}(\theta)\end{aligned} \tag{1}$$

where BCE is the binary cross-entropy loss, λ controls the weight of pivot prediction loss, $pivots$ is a function that selects the indices from an instance that are the pivot features to be predicted, and ρ is the weight of the regularization term $\mathcal{R}$.

To train this model, we alternate passing labeled source data and unlabeled data from the source and target domains into the network. For the labeled data, the error term is the sum of task- and pivot-prediction tasks, while for the unlabeled data only the pivot-prediction loss is computed.

Training proceeds for 30 epochs, with mini-batch size of 50 instances, using the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. For the loss function weight, we used $\rho = 0.1$ and $\lambda = 100$. We used held-out source data to compute validation loss after each epoch and selected the trained model with the lowest validation loss.

3.2 Pivot Selection for Neural Systems

One standard way of choosing pivot features is by calculating mutual information (MI) between the source features and labels, and selecting the features with the highest MI. It is far from clear, however, that this technique is always optimal. Earlier experiments with the POS tagging task (Blitzer et al., 2006) used feature frequency instead, and the extent of the correlation between frequency and MI for that task is not established.

Here, we attempt to provide some evidence about the quality of MI for the task of sentiment classification, using the classification pair of books to electronics. First, we wanted to rule out the null hypothesis that prediction of MI pivots is essentially a generic representation learning algorithm – in other words, that a network learning structure between any sets of sufficiently common features may improve adaptation performance. We modified Ziser et al.'s code to simply select random feature indices from the subset of those that occurred frequently enough to be pivot candidates. With this setup, adaptation performance averaged 0.724 across ten runs, well below their reported 0.744, casting doubt on the null hypothesis.

Next, we want to examine the contention that features with high MI relative to source labels are general. To do this, we simply compare the list of MI features used when the source is *books* to those when the source is *electronics*. We find that, out of the 100 pivot features selected by MI in either cases, there is overlap of 26 features, some examples of which are shown in Table 1 (left). Table 1 also shows a number of MI-selected pivots from the *books* domain that are not general (middle), and then a set of features MI-correlated with the target domain that seem general (right). These latter two columns are essentially precision and recall errors of the MI pivot selection algorithm.

Shared features	Book (spec)	Elec (gen)
an excellent	writing	and great
best	care about	is perfect
bad	flat	never buy
highly recommend	pages	not good
waste your	finish	poor

Table 1: Example features selected as pivots using MI. *Shared features* indicates those selected for both domains. *Book (spec)* highlights pivots selected in the books domain that appear domain-specific. *Elec (gen)* indicates high-MI pivots in the electronics domain that appear domain-general that are not selected when *books* is the source domain.

S→T	AE-SCL	AE-SCL$_R$	$Joint_{MI}$	$Joint_O$
B→D	0.794	**0.812**	0.808	0.84
B→E	0.744	0.761	**0.786**	0.818
B→K	0.795	**0.814**	0.809	0.831
D→B	0.758	0.779	**0.807**	0.808
D→E	0.763	0.763	**0.803**	0.843
D→K	0.8	0.821	**0.83**	0.832
E→B	0.701	0.701	**0.738**	0.749
E→D	0.732	0.748	**0.766**	0.824
E→K	0.848	0.848	**0.875**	0.869
K→B	0.742	0.731	**0.744**	0.767
K→D	0.743	0.754	0.752	0.828
K→E	0.828	0.841	**0.856**	0.851
Ave.	0.771	0.781	**0.798**	0.821

Table 2: Summary of results. B=Books, D=Dvd, E=Electronics,K=Kitchen. AE-SCL=Reported results from Ziser and Reichart (2017). AE-SCL$_R$=Replicated results from the same. $Joint_{MI}$=results from the joint model in Section 3.1. $Joint_O$=results from the joint model using oracle MI pivot selection. Bold indicates a significant difference ($p < 0.05$) between AE-SCL$_R$ and $Joint_{MI}$ using Welch's one-tailed t-test.

Finally, we perform an oracle-based adaptation experiment, where we select the pivot feature indices using MI against the gold labels of the *target* domain, but then proceed with training looking only at source labels, with results in Table 2 discussed below.

4 Evaluation

We follow the standard setup for the Amazon sentiment task, splitting each source dataset into 1600 training and 400 validation instances, and evaluating on the entire labeled target dataset for each pair. We compare against two baselines: First, the reported results of Ziser and Reichart, and second, our replication of their results using their code. Our replication changed their code by replacing the stochastic gradient descent optimizer with Adam (Kingma and Ba, 2014), and increasing the training batch size from 1 to 50. These changes were made to speed training runs during development; we found they produced better-than-reported results and include these superior results as an even stronger baseline. We report results of two configurations of our joint learner. The first configuration ($Joint_{MI}$) uses the MI between source labels and features to select 100 pivot features. The second configuration ($Joint_{Oracle}$) is an oracle-informed system where we use the MI between *target* labels and features to select pivot features, but only use source labels while training the network. Both the AE-SCL$_R$ model and our $Joint_{MI}$ model were run for 10 iterations to minimize differences due to random initialization and to calculate significance statistics.

Table 2 shows the results of our experiments. First, we note that our replication of AE-SCL improves upon their reported results in 8 of 12 pairs, often by substantial margins, and is only worse in one pair (Kitchen→Books). Our $Joint_{MI}$ method is superior to the reported AE-SCL results in all pairs, 1.7 points (absolute) on average, and significantly better than the AE-SCL$_R$ in 9 of 12 pairs, using Welch's one-tailed t-test. This is, to our knowledge, the best result on this task using a feature-based approach (i.e., excluding systems that use embeddings). Despite constraining our system to adapting feature-based models, this result is competitive with the best-known result using a pure neural approach with embeddings as input, as Ziser and Reichart (2018) report an average accuracy of 0.804. The $Joint_{Oracle}$ configuration shows that, despite the large gains of joint training, there is still significant improvement available with better pivot selection.

5 Discussion and Conclusion

Our results show that by jointly learning representations and task networks, UDA can be greatly improved over existing neural UDA methods. We note that there are existing domain adaptation methods that use joint training with auxiliary tasks. Yu and Jiang (2016) use an auxiliary task of predicting whether a masked pivot word in a sentence is positive or negative sentiment, where they introduce a new technique to select pivots that still is based on correlations with source labels. Our work is unique in showing that the standard task of mutual-information-selected pivot prediction is a high quality auxiliary task, though future work should explore whether their pivot selection algorithm is superior to MI in our joint model. We also showed that existing neural UDA methods can be improved significantly with minor changes to the training regimen. Finally, we show that mutual information pivot selection is quite far from the performance ceiling provided by oracle-based pivot selection.

This work evaluated on the widely-used Amazon sentiment dataset from Blitzer et al. (2007). However, we believe that future work on domain adaptation should phase out the use of this dataset.[3] The test set for this setup is flawed in two important ways: first, it is artificially balanced with positive and negative reviews, when the problem is not actually balanced; it also has 3-star reviews removed, which is not a realistic test set setup without looking at labels. For these reasons, we recommend that future work use different domain adaptation datasets.

The Pytorch implementation used to produce these results is publicly available. [4]

Acknowledgments

Research reported in this publication was supported by the National Library Of Medicine of the National Institutes of Health under Award Number R01LM012918. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Jeremy Barnes, Roman Klinger, and Sabine im Walde. 2018. Projecting Embeddings for Domain Adaptation: Joint Modeling of Sentiment in Diverse Domains. In *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics*, Santa Fe, USA. Association for Computational Linguistics.

J Blitzer, M Dredze, and F Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *ACL 2007*, page 440.

J Blitzer, R McDonald, and F Pereira. 2006. Domain adaptation with structural correspondence learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006).*

Minmin Chen, Kilian Weinberger, Fei Sha, and Yoshua Bengio. 2014. Marginalized denoising autoencoders for nonlinear representations. In *International Conference on Machine Learning*, pages 1476–1484.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.

U Sapkota, T Solorio, M Montes-y Gómez, and S Bethard. 2016. Domain Adaptation for Authorship Attribution: Improved Structural Correspondence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2226–2235.

---

[3] Thanks to the anonymous reviewer who made this argument which we found convincing. We credit them with these points while accepting the blame for any poor communication of these points.

[4] `https://github.com/tmills/Neural-SCL-Domain-Adaptation`

A Søgaard and Y Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 236–246.

Yftah Ziser and Roi Reichart. 2017. Neural Structural Correspondence Learning for Domain Adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 400–410. Association for Computational Linguistics.

Yftah Ziser and Roi Reichart. 2018. Pivot Based Language Modeling for Improved Neural Domain Adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1241–1251. Association for Computational Linguistics.

Learning Bilingual Sentiment-Specific Word Embeddings without Cross-lingual Supervision

Yanlin Feng and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{fengyanlin,wanxiaojun}@pku.edu.cn

Abstract

Word embeddings learned in two languages can be mapped to a common space to produce Bilingual Word Embeddings (BWE). Unsupervised BWE methods learn such a mapping without any parallel data. However, these methods are mainly evaluated on tasks of word translation or word similarity. We show that these methods fail to capture the sentiment information and do not perform well enough on cross-lingual sentiment analysis. In this work, we propose UBiSE (Unsupervised Bilingual Sentiment Embeddings), which learns sentiment-specific word representations for two languages in a common space without any cross-lingual supervision. Our method only requires a sentiment corpus in the source language and pretrained monolingual embeddings of both languages. We evaluate our method on three language pairs for cross-lingual sentiment analysis. Experimental results show that our method outperforms previous unsupervised BWE methods and even supervised BWE methods. Our method succeeds for a distant language pair English-Basque.

1 Introduction

Lack of annotated corpora degrades the quality of sentiment analysis in low-resource languages. Cross-lingual sentiment analysis tackles this problem by adapting the sentiment resource in a resource-rich language (the source language) to a resource-poor language (the target language).

Bilingual Word Embeddings (BWE) provide a way to transfer the sentiment information from the source language to the target language. There has been an increasing interest in BWE methods in recent years, including both supervised methods and unsupervised methods. Supervised BWE methods map the word vectors of the two languages in a common space by exploiting either a bilingual seed dictionary or other parallel data, while unsupervised BWE methods do not utilize any form of bilingual supervision. Yet, these methods are mostly evaluated on tasks of word translation or word similarity, and do not perform well enough on cross-lingual sentiment analysis as shown in Section 4.

Consider the case where we want to perform sentiment analysis on the target language with merely an annotated sentiment corpus in the source language. We assume pretrained monolingual embeddings of both languages are available to us. One solution is to first align the embeddings of both languages in a common space using unsupervised BWE methods, then train a classifier based on the source sentiment corpus. In this solution, no sentiment information is utilized to learn the alignment.

In this paper, we propose to exploit the sentiment information and learn sentiment-specific alignment. The sentiment information is gradually incorporated into the BWE through an iterative constraint relaxation procedure. Unlike previous work which performed alignment in a single direction by linearly mapping the source vectors to the target vector space, we propose an alignment model that maps the vectors of the two languages to a new shared space with two non-linear transformations. Our model is able to separate positive vectors from negative vectors in the bilingual space and allow such sentiment information to be transferred to the target language. Our main contributions are as follows:

1. We propose a novel approach to learn bilingual sentiment-specific word embeddings without any cross-lingual supervision and perform cross-lingual sentiment analysis with minimum resource requirement. We propose an iterative constraint relaxation pro-

Proceedings of NAACL-HLT 2019, pages 420–429
Minneapolis, Minnesota, June 2 - June 7, 2019.

cedure that gradually incorporates the sentiment information into the BWE. Our proposed approach achieves state-of-the-art results.

2. We introduce a novel sentiment-specific objective without having to explicitly build a classifier. Our approach is more explainable and better balances sentimental similarity and semantic similarity compared to previous approaches.

3. We introduce an alignment-specific objective and a simple re-normalization trick. Unlike previous BWE methods that learn orthogonal mappings, we introduce non-orthogonal mappings which enable the transfer of sentiment information from the source language to the target language.

2 Related Work

Cross-Lingual Sentiment Analysis Existing approaches for cross-lingual sentiment analysis can be mainly divided into two categories: (i) approaches that rely on machines translation (MT) systems (ii) approaches that rely on cross-lingual word embeddings.

Standard MT-based approaches perform cross-lingual sentiment analysis by translating the sentiment data into a selected language (e.g. English). More sophisticated algorithms including co-training (Wan, 2009; Demirtas and Pechenizkiy, 2013) and multi-view learning (Xiao and Guo, 2012) have been shown to improve performance.

Zhou et al. (2015, 2016b,a) performed cross-lingual sentiment analysis by learning bilingual document representations. These methods translate each document into the other language and enforce a bilingual constraint between the original document and the translated version.

Bilingual Word Embeddings Word embeddings trained separately on two languages can be aligned in a shared space to produce Bilingual Word Embeddings (BWE), which support many NLP tasks including machine translation (Lample et al., 2017), cross-lingual sentiment analysis (Barnes et al., 2018; Zhou et al., 2015) and cross-lingual dependency parsing (Guo et al., 2015). BWE can be obtained in a supervised way using a seed dictionary (Joulin et al., 2018; Artetxe et al., 2016), or in an unsupervised way without any bilingual data. Adversarial training was the first successful attempt to learn unsupervised BWE (Zhang et al., 2017; Conneau et al., 2017). Self-learning was proposed by (Artetxe et al., 2017)to learn BWE with minimum bilingual resources, which was later extended into a fully unsupervised framework by adding an unsupervised dictionary initialization step (Artetxe et al., 2018).

Multilingual Word Embeddings BWE methods can be extended to the case of multiple languages by simply mapping all the languages to the vector space of a selected language. However, directly learning multilingual word embeddings (MWE) in a shared space has been shown to improve performance (Ammar et al., 2016; Duong et al., 2017; Chen and Cardie, 2018; Alaux et al., 2018). Yet, all these approaches are mainly evaluated on word translation and their effectiveness on cross-lingual sentiment analysis have not been empirically compared.

Sentimental Embeddings Continuous word representations encode the syntactic context of a word but often ignore the information of sentiment polarity. This drawback makes them hard to distinguish words with similar syntactic context but opposite sentiment polarity (e.g. *good* and *bad*), resulting in unsatisfactory performance on sentiment analysis. Tang et al. (2014) learned word representations that encode both syntactic context and sentiment polarity by adding an objective to classify the polarity of an n-gram. This method can be generalized to the cross-lingual setting by training monolingual sentimental embeddings on both languages then aligning them in a common space. However, it requires sentiment resources in the target language thus is impractical for low-resource languages.

There are also approaches to learn sentimental embeddings in the bilingual space without any sentiment resources in the target language. Barnes et al. (2018) jointly minimized an alignment objective based on a seed dictionary, and a classification objective based on the sentiment corpus. Its performance is compared to our method in Section 4. Xu and Wan (2017) learned multilingual sentimental embeddings by extending the BiSkip model (Luong et al., 2015). However, their method does not apply to pretrained embeddings and requires large-scale parallel corpora thus is not included in our experiments.

3 Proposed Method

3.1 The Overall Framework

This subsection first introduces the proposed mappings for aligning the monolingual embeddings in the bilingual space, then describes the general self-learning algorithm used to learn these bilingual mappings. The details of our algorithm are explained in Section 3.2 - Section 3.6.

3.1.1 The Alignment Model

We assume we have normalized monolingual embeddings $\mathbf{S} \in \mathbb{R}^{v \times d}$ and $\mathbf{T} \in \mathbb{R}^{v \times d}$, where the i-th row of $\mathbf{S}$ is the vector representation of word i in the source language. The normalization procedure is as follows: (i) l_2-normalize each vector (ii) center the vectors (iii) l_2-normalize each vector again (Artetxe et al., 2018).

Given these monolingual embeddings, existing BWE methods typically learn a projection matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ from the source vector space to the target vector space. However, these methods are unsuitable in our setting for two reasons: (i) most methods constrain $\mathbf{W}$ to be orthogonal or near-orthogonal, thus preserving distances between word vectors; (ii) word vectors in the target language space remain unchanged. These two properties prevent us from separating words with opposite sentiment polarity in the bilingual space. In this work, we propose to align the monolingual embeddings with two non-linear mappings:

$$f_s(\mathbf{x}) = \frac{\mathbf{W}_s\mathbf{x}}{\|\mathbf{W}_s\mathbf{x}\|} \qquad f_t(\mathbf{x}) = \frac{\mathbf{W}_t\mathbf{x}}{\|\mathbf{W}_t\mathbf{x}\|}$$

where $\|\cdot\|$ denotes the l_2-norm, $\mathbf{W}_s$ ($\mathbf{W}_t$) is the projection matrix for the source (target) embeddings, and $\mathbf{x}$ is a d-dimension word vector. Each mapping can be seen as a linear projection followed by a re-normalization step.

We propose the following convex domain $\mathcal{D} = \{\mathbf{W} \in \mathbb{R}^{d \times d} \mid \|\mathbf{W}\|_2 \leq r\}$ as an alternative for the orthogonal constraint, where $\|\cdot\|_2$ denotes the spectral norm and r is a hyperparameter that determines to what extent we want to preserve word distances. This is inspired by the unit spectral norm constraint proposed by (Joulin et al., 2018).

3.1.2 The Self-learning Procedure

Given a bilingual seed dictionary, we can learn the projection matrices $\mathbf{W}_s$ and $\mathbf{W}_t$ by forcing the word pairs in the dictionary to have similar representations in the bilingual space. In the unsupervised case, such a dictionary can be induced from the monolingual embeddings $\mathbf{S}$ and $\mathbf{T}$ (Artetxe et al., 2018). However, the quality of this dictionary is usually not good, which in turn degrades the quality of the projection matrices learned from this dictionary. Previous work (Artetxe et al., 2017, 2018) showed that an iterative self-learning procedure can induce a good bilingual dictionary and hence good projection matrices. Given an initial dictionary D_{bi}, this procedure iterates over two steps: (i) it aligns the monolingual embeddings in a common space based on D_{bi}, yielding $\mathbf{S}'$ and $\mathbf{T}'$; (ii) it computes a new dictionary D_{bi} using nearest neighbour retrieval over the approximately aligned embeddings $\mathbf{S}'$ and $\mathbf{T}'$.

In our method, there are three objects $\mathbf{W}_s$, $\mathbf{W}_t$ and D_{bi} to update through the self-learning procedure. Thus we iterates over the following three steps:

1. Solve $\mathbf{W}_s$ by minimizing a sentiment-specific objective $\mathcal{L}_s$ over $\mathcal{D}$, as described in Section 3.3;
2. Solve $\mathbf{W}_t$ by minimizing an alignment-specific objective $\mathcal{L}_t$ over $\mathcal{D}$, as described in Section 3.4;
3. Derive a new bilingual dictionary D_{bi} based on $\mathbf{S}' = \mathbf{S}\mathbf{W}_s^\top$ and $\mathbf{T}' = \mathbf{T}\mathbf{W}_t^\top$, as described in Section 3.5.

Re-normalization is applied as a final step after we have obtained $\mathbf{W}_s$ and $\mathbf{W}_t$.

3.2 Preliminaries

3.2.1 Unsupervised Bilingual Dictionary Initialization

The normalized embeddings $\mathbf{S}$ and $\mathbf{T}$ are not aligned along the first axis, i.e., the i-th row of $\mathbf{S}$ does not correspond to the i-th row of $\mathbf{T}$. Therefore, an initial bilingual dictionary is required in order to access the correspondence between the two languages. Following (Artetxe et al., 2018), we first compute the similarity matrices $\mathbf{M}_s = \sqrt{\mathbf{S}\mathbf{S}^\top}$ and $\mathbf{M}_t = \sqrt{\mathbf{T}\mathbf{T}^\top}$, sort them along the second axis and normalize the rows, yielding $\mathbf{M}'_s$ and $\mathbf{M}'_t$. For each row in $\mathbf{M}'_s$, we apply nearest neighbour retrieval over the rows of $\mathbf{M}'_t$ to find its corresponding translation, yielding a dictionary $D_{s \to t} =$

$\{(1, T_{s\to t}(1)), (2, T_{s\to t}(2)), \ldots, (v, T_{s\to t}(v))\}$, where $T_{s\to t}(i)$ is the translation of the source word i. The same procedure is repeated in the other direction, yielding $D_{t\to s}$. The two dictionaries are then concatenated to produce the initial bilingual dictionary $D_{bi} = D_{s\to t} \cup D_{t\to s}$.

3.2.2 Learning Sentiment-Specific Vectors

In order to incorporate the sentiment information into the bilingual word embeddings, we need a set of d-dimension vectors with known sentiment polarity. We propose a neural network based approach to learn these sentiment-specific vectors. Let the training corpus in the source language be $\mathcal{C} = \{(z_1, y_1), (z_2, y_2), \ldots, (z_{|\mathcal{C}|}, y_{|\mathcal{C}|})\}$, where z_i is a text and y_i is its corresponding label. A d-dimension vector with sentiment polarity y_i can be obtained by calculating the weighted average of the word vectors in z_i:

$$\mathbf{h}_i = \frac{\sum_{j\in z_i} \exp(\alpha_j)\mathbf{S}_{j\cdot}}{\sum_{j\in z_i} \exp(\alpha_j)} \tag{1}$$

where $\mathbf{S}_{j\cdot}$ is the vector representation of the word j in the source language (corresponding to the j-th row of $\mathbf{S}$) and α_j is a scalar that scores the importance of word j on the sentiment polarity. α_j is computed by $\alpha_j = \max(\mathbf{A}\mathbf{S}_{j\cdot} + \mathbf{b})$, where $\mathbf{A} \in \mathbb{R}^{h\times d}$ and $\mathbf{b} \in \mathbb{R}^h$ are the parameters to learn. This function can be seen as a convolution layer with h filters followed by a max pooling layer. The number of filters h is set to 4. Each $\mathbf{h}_i$ is then forwarded to a linear classifier to predict the sentiment label y_i.

Once we have trained the model by minimizing the cross-entropy loss, we re-compute $\mathbf{h}_i$ for each training example z_i. We denote the set of vectors (i.e., $\mathbf{h}_i$) with positive labels as $\mathcal{P} = \{\mathbf{h}^p_1, \mathbf{h}^p_2, \ldots, \mathbf{h}^p_{|\mathcal{P}|}\}$ and the set of vectors with negative labels as $\mathcal{N} = \{\mathbf{h}^n_1, \mathbf{h}^n_2, \ldots, \mathbf{h}^n_{|\mathcal{N}|}\}$. In the 4-class setup, we have four sets: $\mathcal{P}$, $\mathcal{N}$, $\mathcal{SP}$ (the set of strongly positive vectors), $\mathcal{SN}$ (the set of strongly negative vectors).

3.3 Solving $\mathbf{W}_s$

Given a set of positive d-dimension vectors $\mathcal{P} = \{\mathbf{h}^p_1, \mathbf{h}^p_2, \ldots, \mathbf{h}^p_{|\mathcal{P}|}\}$ and a set of negative d-dimension vectors $\mathcal{N} = \{\mathbf{h}^n_1, \mathbf{h}^n_2, \ldots, \mathbf{h}^n_{|\mathcal{N}|}\}$ (or four sets in the 4-class setup), our goal is to distinguish the positive vectors from the negative vectors in the bilingual space, i.e., to separate $\mathbf{W}_s\mathbf{h}^p_i$ from $\mathbf{W}_s\mathbf{h}^n_j$ for any pair of i, j.

We introduce a new d-dimension vector $\mathbf{a}^p \in \mathcal{O} = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\| \leq 1\}$ to represent the "positive direction", which is to be learned. In order to separate positive vectors from negative vectors in the bilingual space, we try to make $\mathbf{W}_s\mathbf{h}^p_i$ $(i = 1, \ldots, |\mathcal{P}|)$ to be close to $\mathbf{a}^p$ and $\mathbf{W}_s\mathbf{h}^n_i$ $(i = 1, \ldots, |\mathcal{N}|)$ to be distant from $\mathbf{a}^p$.

For a given $\mathbf{a}^p$, we first compute ${\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^p_i$ for $i = 1, 2, \ldots, |\mathcal{P}|$ and denote the set of i with $\lambda|\mathcal{P}|$ smallest values as $\mathcal{Q}^p_+$, where $\lambda \in [0, 1]$ is a hyperparameter[1]. These $\mathbf{W}_s\mathbf{h}^p_i$ are least similar with $\mathbf{a}^p$ (dot product is used as the similarity metric), hence we maximize the average of ${\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^p_i$ over $\mathcal{Q}^p_+$. Likewise, we denote the set of $i \in \{1, 2, \ldots, |\mathcal{N}|\}$ with $\lambda|\mathcal{N}|$ largest values of ${\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^n_i$ as $\mathcal{Q}^p_-$. These $\mathbf{W}_s\mathbf{h}^n_i$ are most similar to $\mathbf{a}^p$, hence we minimize the average of ${\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^n_i$ over $\mathcal{Q}^p_-$. The overall objective is as follows:

$$\begin{aligned}\min_{\substack{\mathbf{W}_s\in\mathcal{D}\\ \mathbf{a}^p\in\mathcal{O}}} \mathcal{L}_s(\mathbf{W}_s, \mathbf{a}^p) &= \mathcal{L}'(\mathbf{W}_s, \mathbf{a}^p, \mathcal{P}, \mathcal{N})\\ &\triangleq -\frac{1}{\lambda|\mathcal{P}|}\sum_{i\in\mathcal{Q}^p_+} {\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^p_i\\ &\quad+\frac{1}{\lambda|\mathcal{N}|}\sum_{i\in\mathcal{Q}^p_-} {\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^n_i\end{aligned} \tag{2}$$

where $\mathcal{D}$ is the convex set defined in Section 3.1.1. The rationale for this objective is that, instead of forcing every $\mathbf{W}_s\mathbf{h}^p_i$ to be close to $\mathbf{a}^p$, we only focus on a fraction of positive vectors that are most distant from $\mathbf{a}^p$, and vice versa for those negative vectors. We observe that this objective can be rewritten as:

$$\begin{aligned}\min_{\substack{\mathbf{W}_s\in\mathcal{D}\\ \mathbf{a}^p\in\mathcal{O}}} &\mathcal{L}_s(\mathbf{W}_s, \mathbf{a}^p)\\ &= \frac{1}{\lambda|\mathcal{P}|}\max_{\mathcal{Q}\in\mathcal{S}_{\lambda|\mathcal{P}|}(|\mathcal{P}|)} -\sum_{i\in\mathcal{Q}} {\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^p_i\\ &\quad+\frac{1}{\lambda|\mathcal{N}|}\max_{\mathcal{Q}\in\mathcal{S}_{\lambda|\mathcal{N}|}(|\mathcal{N}|)}\sum_{i\in\mathcal{Q}} {\mathbf{a}^p}^\top\mathbf{W}_s\mathbf{h}^n_i\end{aligned} \tag{3}$$

where $\mathcal{S}_{\lambda|\mathcal{P}|}(|\mathcal{P}|)$ represents all subsets of $\{1, 2, \ldots, |\mathcal{P}|\}$ of size $\lambda|\mathcal{P}|$, and $\mathcal{S}_{\lambda|\mathcal{N}|}(|\mathcal{N}|)$ is similarly defined. [2] This formulation shows that

---

[1] For simplicity, we assume $\lambda|\mathcal{P}|$ is already rounded to an integer.

[2] There is no need to introduce a new vector $\mathbf{a}^n$ to represent the "negative direction" and introduce a new objective, since the new objective is exactly the same after replacing $\mathbf{a}^p$ with $-\mathbf{a}^n$.

both terms of this objective can be seen as a maximum of linear functions of either $\mathbf{W}_s$ or $\mathbf{a}^p$. Therefore, our objective is convex with respect to either $\mathbf{W}_s$ or $\mathbf{a}^p$, thus can be efficiently minimized by using the projected gradient descent algorithm. We first minimize this objective with respect to $\mathbf{a}^p$ over $\mathcal{O}$, then minimize it with respect to $\mathbf{W}_s$ over $\mathcal{D}$.

While this objective is useful in the binary setup, it does not separate a strongly positive vector in $\mathcal{SP}$ from a weakly positive vector in $\mathcal{P}$ (similarly for $\mathcal{SN}$ and $\mathcal{N}$). In order to achieve better performance in the 4-class setup, we adopt the one-versus-rest strategy to write $\mathcal{L}_s$ as the sum of four terms:

$$\begin{aligned}
&\min_{\substack{\mathbf{W}_s \in \mathcal{D} \\ \mathbf{a}^p \in \mathcal{O} \\ \mathbf{a}^{sp} \in \mathcal{O} \\ \mathbf{a}^n \in \mathcal{O} \\ \mathbf{a}^{sn} \in \mathcal{O}}} \mathcal{L}_s(\mathbf{W}_s, \mathbf{a}^p, \mathbf{a}^{sp}, \mathbf{a}^n, \mathbf{a}^{sn}) \\
&\quad = \mathcal{L}'(\mathbf{W}_s, \mathbf{a}^p, \mathcal{P}, \mathcal{N} \cup \mathcal{SP} \cup \mathcal{SN}) \\
&\quad + \mathcal{L}'(\mathbf{W}_s, \mathbf{a}^{sp}, \mathcal{SP}, \mathcal{P} \cup \mathcal{N} \cup \mathcal{SN}) \\
&\quad + \mathcal{L}'(\mathbf{W}_s, \mathbf{a}^n, \mathcal{N}, \mathcal{P} \cup \mathcal{SP} \cup \mathcal{SN}) \\
&\quad + \mathcal{L}'(\mathbf{W}_s, \mathbf{a}^{sn}, \mathcal{SN}, \mathcal{P} \cup \mathcal{SP} \cup \mathcal{N}) \quad (4)
\end{aligned}$$

where $\mathcal{L}'$ is defined in Eq.(2) and $\mathbf{a}^c$ is a d-dimension vector representing the "direction" of class c.

3.4 Solving $\mathbf{W}_t$

Based on the current bilingual dictionary D_{bi}, we construct two sets of vectors $\{\mathbf{x}_1^s, \mathbf{x}_2^s, \ldots, \mathbf{x}_{2v}^s\}$ and $\{\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_{2v}^t\}$, where $\mathbf{x}_i^s$ and $\mathbf{x}_i^t$ are the vector representations of the i-th word pair in D_{bi}. With $\mathbf{W}_s$ fixed, we can solve $\mathbf{W}_t$ by minimizing:

$$\min_{\mathbf{W}_t \in \mathcal{D}} \mathcal{L}_t(\mathbf{W}_t) = \sum_{i=1}^{2v} \|\mathbf{W}_s \mathbf{x}_i^s - \mathbf{W}_t \mathbf{x}_i^t\|^2 \quad (5)$$

where $\mathcal{D}$ is the convex set defined in Section 3.1.1. This objective is convex with respect to $\mathbf{W}_t$, thus can be minimized efficiently by using the projected gradient descent algorithm.

3.5 Bilingual Dictionary Induction

Once we have computed $\mathbf{W}_s$ and $\mathbf{W}_t$, we can obtain the aligned embeddings $\mathbf{S}' = \mathbf{S}\mathbf{W}_s^\top$ and $\mathbf{T}' = \mathbf{T}\mathbf{W}_t^\top$. Then we induce a new dictionary D_{bi} using nearest neighbour retrieval over the rows of $\mathbf{S}'$ and $\mathbf{T}'$. We perform the induction in two directions to produce $D_{s \to t}$ and $D_{t \to s}$, then concatenate them to produce D_{bi}.

In this work, we propose a modified version of CSLS(Conneau et al., 2017) to be used as the similarity metric to preform nearest neighbour retrieval:

$$\begin{aligned}
&\mathrm{CSLS}'(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} \\
&\quad - \frac{1}{k} \sum_{\mathbf{y}' \in \mathcal{N}_Y(\mathbf{x})} \mathbf{x}^\top \mathbf{y}' - \frac{1}{k} \sum_{\mathbf{x}' \in \mathcal{N}_X(\mathbf{y})} \mathbf{x}'^\top \mathbf{y} \quad (6)
\end{aligned}$$

where $\mathcal{N}_Y(\mathbf{x})$ is the set of k nearest neighbours of $\mathbf{x}$ in the set of vectors Y (in our case Y is the set of rows of $\mathbf{T}'$). We set k to 10 following the original paper.

3.6 Iterative Constraint Relaxation

As mentioned in Section 3.1.1, we introduce a hyperparameter r to define the convex domain $\mathcal{D}$. There is a trade-off to make for r: a large r better incorporates sentimental similarity but significantly harms the quality of the alignment, while a small r constrains $\mathbf{W}_s$ to be near-orthogonal thus prevents it to capture the sentimental similarity.

In order to address this problem, we propose to first set r to 1, letting the the monolingual embeddings to be properly aligned. Then r is iteratively increased by Δr, causing the positive vectors in the bilingual space to be gradually moved further away from the negative vectors. The training process stops when r reaches a maximum value r_{max}, where r_{max} is a hyperparameter [3].

The pseudo code of UBiSE in the binary setup is shown in Algorithm 1. For the 4-class UBiSE, lines 3,6,7 are replaced by their counterparts in the 4-class setup.

4 Experiments

4.1 Datasets

We use the multilingual sentiment dataset provided by (Barnes et al., 2018). It contains annotated hotel reviews in English (EN), Spanish (ES), Catalan (CA) and Basque (EU). In our experiment, we use EN as the source language and ES, CA, EU as the target languages. For each target language, the dataset is divided into a target development set and a target test set. We also combine the strong and weak labels to produce a binary setup.

---

[3] Although having the number of iterations be implicitly defined by r_{max} and Δr makes choosing a small r_{max} impractical, it allows us to tune r_{max} in a single training process.

Algorithm 1 binary UBiSE

```
Input: $\lambda, r_{max}, \Delta r, \mathbf{S}, \mathbf{T}, \mathcal{C}$
Output: $\mathbf{S}', \mathbf{T}', \mathbf{W}_s, \mathbf{W}_t$

$r \leftarrow 1$
Initialize $\mathbf{W}_s$ and $\mathbf{W}_t$ to identity matrices
Learn $\mathcal{P}, \mathcal{N}$ from $\mathbf{S}$ and $\mathcal{C}$, according to Section 3.2.2
Compute the initial bilingual dictionary $D_{bi}$ from $\mathbf{S}$ and $\mathbf{T}$, according to Section 3.2.1
while $r \leq r_{max}$ do
    $\mathbf{a}^p \leftarrow \mathrm{argmin}_{\mathbf{a}^p \in \mathcal{O}} \mathcal{L}_s(\mathbf{a}^p, \mathbf{W}_s)$
    $\mathbf{W}_s \leftarrow \mathrm{argmin}_{\mathbf{W}_s \in \mathcal{D}} \mathcal{L}_s(\mathbf{a}^p, \mathbf{W}_s)$
    $\mathbf{W}_t \leftarrow \mathrm{argmin}_{\mathbf{W}_t \in \mathcal{D}} \mathcal{L}_t(\mathbf{W}_t)$
    $\mathbf{S}' \leftarrow \mathbf{S}\mathbf{W}_s^\top$
    $\mathbf{T}' \leftarrow \mathbf{T}\mathbf{W}_t^\top$
    Derive a new bilingual dictionary $D_{bi}$ from $\mathbf{S}'$ and $\mathbf{T}'$, according to Section 3.5
    $r \leftarrow r + \Delta r$
end while
Normalize the rows of $\mathbf{S}'$, $\mathbf{T}'$ to unit length
return $\mathbf{S}', \mathbf{T}', \mathbf{W}_s, \mathbf{W}_t$
```

The normalized 300-dimension fastText vectors (Bojanowski et al., 2017) are used by all methods. The MUSE dataset(Conneau et al., 2017) is used by approaches that require bilingual supervision [4]. Each dictionary contains 5000 unique source words.

4.2 Implementation details

We empirically set $\Delta r = 0.01$ and $v = 10000$. The vocabulary of each language is limited to the v most frequent words so that the embedding matrix has shape $v \times d$. Hyper parameters λ and r_{max} are tuned on the target development set via a grid search. We apply stochastic dictionary induction by randomly setting the elements of the similarity matrix used for nearest neighbour retrieval to zero with probability $1 - p$, as described in (Artetxe et al., 2018). p is initialized to 0.1 and increased by 0.005 at each iteration. We empirically stop updating the dictionary when r exceeds 3.

4.3 Baselines

We compare our method with the following baselines, including state-of-the-art BWE methods that are originally evaluated on the word translation task, as well as bilingual sentimental embeddings methods that are optimized for cross-lingual sentiment analysis. The bilingual word embeddings learned by each method are later evaluated on cross-lingual sentiment analysis using the same classifier for fairness.

4.3.1 Unsupervised BWE Methods

ADVERSARIAL Conneau et al. (2017) proposed an unsupervised BWE method based on adversarial training. After a near-orthogonal projection matrix is learned through adversarial training, a refinement procedure is applied to improve the quality of the alignment.

VECMAP Artetxe et al. (2018) proposed an unsupervised BWE learning framework. It consists of an unsupervised dictionary initialization step and the self-learning procedure mentioned in Section 3.1.2.

4.3.2 Supervised BWE Methods

PROCRUSTES Artetxe et al. (2016) proposed a simple and effective supervised BWE method that requires a seed dictionary. It computes the optimal projection matrix by taking singular value decomposition (SVD).

RCSLS Joulin et al. (2018) proposed an supervised BWE method that also requires a seed dictionary. They proposed a training objective that is consistent with the retieval criterion that can be minimized by using gradient descent. It achieves state-of-the-art results on the word translation task.

4.3.3 Bilingual Sentimental Embedding Methods

BLSE Barnes et al. (2018) exploited both bilingual supervision and the sentiment corpus to learn bilingual sentimental embeddings. They jointly minimize an alignment-specific objective and a classification objective to learn the projection matrices. The trade-off between the two objectives is controlled by a hyperparameter $\alpha \in [0, 1]$. We tune α on the target development set as described in the original paper. Once the projection matrices have been learned, the classifier in this model is abandoned. The quality of the resulting BWE is evaluated using the classifier mentioned in Section 4.4.

4.4 Evaluation

We use DAN (Iyyer et al., 2015) as the classifier to preform cross-lingual sentiment analysis. The loss

---

[4] This dataset does not contain a dictionary for EN-EU, thus we translate the EN-ES dictionary into EN-EU

of each instance is weighted by its inverse class frequency to address the class imbalance problem. For each method, the dropout rate is fixed at 0.3 and the l_2-regularization strength is tuned on the target development set[5]. We train five classifiers for each method and report the average macro-F1 on the target test set.

4.5 Results and Analysis

Table 1 presents the results of different BWE methods. UBISE outperforms all unsupervised methods on all six tasks and outperforms all baselines on four out of six tasks.

All methods, especially unsupervised methods, suffer from distant language pairs, which is consistent with the observation of (Søgaard et al., 2018). VECMAP and ADVERSARIAL perform significantly worse on EN-EU compared to supervised methods. Yet, UBISE outperforms the strongest baseline by 2.1% on EN-EU, indicating that incorporating sentiment is vital for cross-lingual sentiment analysis on distant languages.

Despite the similar performance across different BWE methods in the binary setup, UBISE outperform all baselines in the 4-class setup by a large margin (average of $+2.2\%$). This may indicate that the original monolingual embeddings are able to distinguish positive words from negative words(e.g., *good* and *bad*), but bad at distinguishing strongly positive words from weakly positive words (e.g., *good* and *perfect*).

The performance of BLSE is merely comparative with other baselines.[6] We suspect that this is due to the classifier we use to perform cross-lingual sentiment analysis. The original paper used SVM or logistic regression to perform classification, in which case BLSE achieved better performance due to the utilization of sentiment information. But if we use a deeper neural network to perform cross-lingual sentiment analysis, preserving the original semantic similarity is more important. A qualitative comparison between BLSE and UBISE is presented in Section 4.8.

4.6 Effect of the Sentiment Information

We perform an ablation test to demonstrate the effect of the sentiment information provided by $\mathcal{L}_s$. We create a new model UBISE_MIN that does not utilize the sentiment information by eliminating lines 6,7,12 in Algorithm 1. UBISE_MIN runs 500 iterations for every language pairs.

The comparative results in Table 2 show that utilizing the sentiment information leads to an average improvement of $+3.1\%$ in the binary setup or $+4.1\%$ in the 4-class setup.

4.7 Effect of Re-Normalization

Re-normalization is useful in the sense that it leads to better alignment by constraining all the bilingual vectors to be on the unit sphere. While this property does not matter for word translation as long as cosine-similarity is used as the retrieval criterion, it matters for cross-lingual sentiment analysis. Another effect of re-normalization is that it introduces non-linearity between the linear projection and the classifier, which is vital for separating words with opposite sentiment polarity. Without non-linearity the linear projection and the first layer of the classifier would collapse into a single linear projection, thus eliminating the effect of $\mathbf{W}_s$. Figure 2 illustrates how this non-linearity helps separating positive words from negative words in the bilingual space. This effect is demonstrated in Section 4.8.

4.8 Visualization of the Bilingual Space

To illustrate how UBISE transfers sentiment information from the source language to the target language, we visualize six categories of words in the bilingual space of UBISE and BLSE using t-SNE (Maaten and Hinton, 2008). As shown in Figure 1, both methods manage to separate positive words from negative words without any annotated data in Spanish. However, Barnes et al. (2018) abandon the original semantic similarity, which degrades its performance as shown in Section 4.5. In contrast, our method preserves semantic similarity by limiting the largest singular values of $\mathbf{W}_s$ and $\mathbf{W}_t$ to be smaller than r_{max}. The trade-off between semantic similarity and sentimental similarity is made by choosing an appropriate r_{max}.

5 Conclusion

This paper presents a method to learn bilingual sentiment-specific word embeddings without any cross-lingual supervision. We propose a novel sentiment-specific objective that separates words

---

[5] The optimal regularization strength depends on the BWE method. Stronger regularization is favourable to BLSE and UBISE.

[6] We already obtain significantly better results after replacing the original classifier with DAN, compared with the original reported results.

		Binary			4-class		
Bilingual Supervision	**Method**	**ES**	**CA**	**EU**	**ES**	**CA**	**EU**
$5k$ dict.	Procrustes	80.4	**83.1**	74.1	49.1	50.9	43.0
	RCSLS	**80.7**	81.4	73.4	50.3	47.8	41.3
	BLSE	80.2	82.2	73.5	50.0	47.0	35.1
None	VecMap	80.0	80.2	69.2	51.2	52.2	38.8
	Adversarial	79.8	79.9	60.3	45.8	47.9	31.0
None	UBiSE	80.5	80.4	**76.7**	**54.4**	**54.1**	**44.6**

Table 1: Macro F1 of different BWE approaches. The best score for each language pair is shown in **bold**. The best score among unsupervised BWE methods for each language pair is underlined.

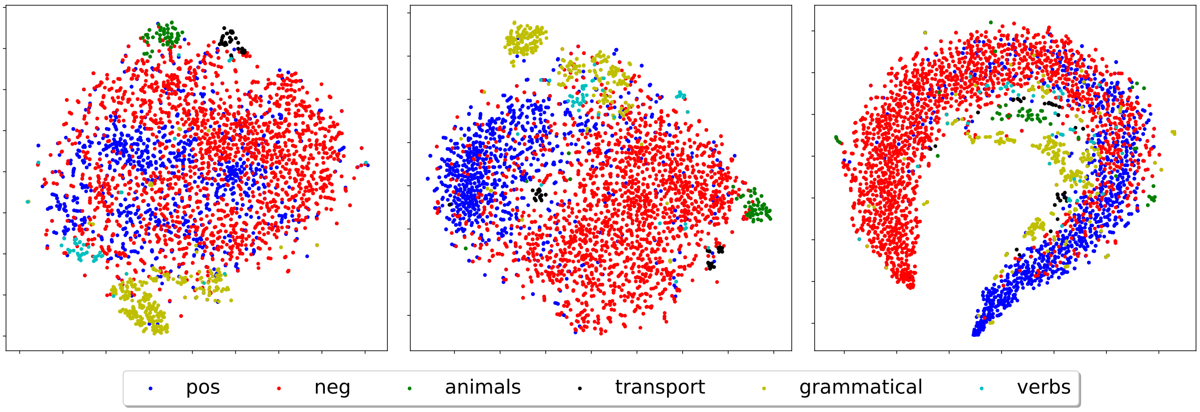
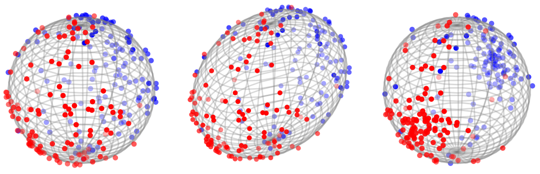


Figure 1: t-SNE Visualization of the Spanish word vectors. Grammatical words include pronouns, prepositions, articles, conjunctions, etc. *left*: original normalized vectors; *middle*: the bilingual space of binary UBiSE with $\lambda = 0.5$ and $r_{max} = 5.5$; *right*: the bilingual space of BLSE.

Setup	**Method**	**ES**	**CA**	**EU**
Binary	UBiSE_Min	77.4	80.2	70.8
	UBiSE	80.5	80.4	76.7
4-class	UBiSE_Min	51.7	49.3	39.9
	UBiSE	54.4	54.1	44.6

Table 2: Comparison between UBiSE and UBiSE_Min

.



(a) (b) (c)

Figure 2: Illustration of the effect of re-normalization. (a) the original normalized embeddings (b) embeddings after linear projection (c) embeddings after re-normalization

with opposite sentiment polarity in the bilingual space, and an alignment objective that enables the transfer of sentiment information from the source language to the target language. An iterative constraint relaxation procedure is applied to gradually incorporate the sentiment information into the bilingual word embeddings. We empirically evaluate our method on three language pairs for cross-lingual sentiment analysis and demonstrate its effectiveness. Experimental results show that incorporating sentiment information significantly improves the performance on fine-grained cross-lingual sentiment analysis.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2018. Unsupervised hyperalignment for multilingual word embeddings. *arXiv preprint arXiv:1811.01124*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2018. Bilingual sentiment embeddings: Joint projection of sentiment across languages. *arXiv preprint arXiv:1805.09016*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. *arXiv preprint arXiv:1808.08933*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Erkin Demirtas and Mykola Pechenizkiy. 2013. Cross-lingual polarity detection with machine translation. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 9. ACM.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual training of crosslingual word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 894–904.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*, pages 235–243. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2012. Multi-view adaboost for multilingual subjectivity analysis. *Proceedings of COLING 2012*, pages 2851–2866.

Kui Xu and Xiaojun Wan. 2017. Towards a universal sentiment classifier in multiple languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 511–520.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 430–440.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016a. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1403–1412.

ReWE: Regressing Word Embeddings for Regularization of Neural Machine Translation Systems

Inigo Jauregi Unanue[1,2], **Ehsan Zare Borzeshi**[*,2], **Nazanin Esmaili**[2], **Massimo Piccardi**[1]
[1] University of Technology Sydney, Sydney, Australia
[2] Capital Markets Cooperative Research Centre, Sydney, Australia

`{ijauregi,ezborzeshi,nesmaili}@cmcrc.com`
`massimo.piccardi@uts.edu.au`

Abstract

Regularization of neural machine translation is still a significant problem, especially in low-resource settings. To mollify this problem, we propose regressing word embeddings (ReWE) as a new regularization technique in a system that is jointly trained to predict the next word in the translation (categorical value) and its word embedding (continuous value). Such a joint training allows the proposed system to learn the distributional properties represented by the word embeddings, empirically improving the generalization to unseen sentences. Experiments over three translation datasets have showed a consistent improvement over a strong baseline, ranging between 0.91 and 2.54 BLEU points, and also a marked improvement over a state-of-the-art system.

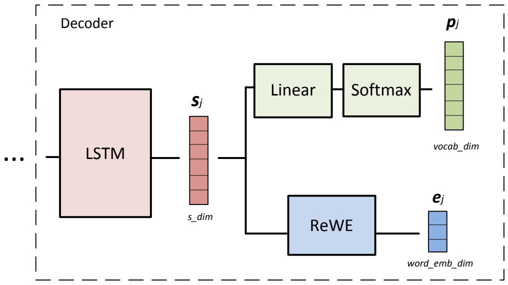


Figure 1: The proposed regularizer: the hidden vector in the decoder, s_j, transits through two paths: 1) a linear and a softmax layers that output vector v_j (vocab_dim) which is used for predicting the target word as usual, and 2) a two-layer network (ReWE) that outputs a vector, e_j, of word embedding size (word_emb_dim). During training, e_j is used in a regressive loss with the ground-truth embedding.

1 Introduction

The last few years have witnessed remarkable improvements in the performance of machine translation (MT) systems. These improvements are strongly linked to the development of neural machine translation (NMT): based on encoder-decoder architectures (also known as seq2seq), NMT can use recurrent neural networks (RNNs) (Sutskever et al., 2014; Cho et al., 2014; Wu et al., 2016), convolutional neural networks (CNNs) (Gehring et al., 2017) or transformers (Vaswani et al., 2017) to learn how to map a sentence from the source language to an adequate translation in the target language. In addition, attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015) help soft-align the encoded source words with the predictions, further improving the translation.

NMT systems are usually trained via maximum likelihood estimation (MLE). However, as pointed out by (Elbayad et al., 2018), MLE suffers from two obvious limitations: the first is that it treats all the predictions other than the ground truth as equally incorrect. As a consequence, synonyms and semantically-similar words — which are often regarded as highly interchangeable with the ground truth — are completely ignored during training. The second limitation is that MLE-trained systems suffer from "exposure bias" (Bengio et al., 2015; Ranzato et al., 2015) and do not generalize well over the large output space of translations. Owing to these limitations, NMT systems still struggle to outperform other traditional MT approaches when the amount of supervised data is limited (Koehn and Knowles, 2017).

In this paper, we propose a novel regularization technique for NMT aimed to influence model learning with contextual properties. The technique — nicknamed ReWE from "regressing word embedding" — consists of modifying a conventional seq2seq decoder to jointly learn to a) predict the next word in the translation (categorical value), as

---

[*] The author has changed affiliation to Microsoft after the completion of this work. His new email is: `Ehsan.ZareBorzeshi@microsoft.com`

Proceedings of NAACL-HLT 2019, pages 430–436
Minneapolis, Minnesota, June 2 - June 7, 2019.

usual, and b) regress its word embedding (numerical value). Figure 1 shows the modified decoder. Both predictions are incorporated in the training objective, combining standard MLE with a continuous loss function based on word embeddings. The rationale is to encourage the system to learn to co-predict the next word together with its context (by means of the word embedding representation), in the hope of achieving improved generalization. At inference time, the system operates as a standard NMT system, retaining the categorical prediction and ignoring the predicted embedding. We qualify our proposal as a regularization technique since, like any other regularizers, it only aims to influence the model's training, while leaving the inference unchanged. We have evaluated the proposed system over three translation datasets of different size, namely English-French (en-fr), Czech-English (cs-en), and Basque-English (eu-en). In each case, ReWE has significantly outperformed its baseline, with a marked improvement of up to 2.54 BLEU points for eu-en, and consistently outperformed a state-of-the-art system (Denkowski and Neubig, 2017).

2 Related work

A substantial literature has been devoted to improving the generalization of NMT systems. Fadaee et al. (2017) have proposed a data augmentation approach for low-resource settings that generates synthetic sentence pairs by replacing words in the original training sentences with rare words. Kudo (2018) has trained an NMT model with different subword segmentations to enhance its robustness, achieving consistent improvements over low-resource and out-of-domain settings. Zhang et al. (2018) have presented a novel regularization method that encourages target-bidirectional agreement. Other work has proposed improvements over the use of a single ground truth for training: Ma et al. (2018) have augmented the conventional seq2seq model with a bag-of-words loss under the assumption that the space of correct translations share similar bag-of-words vectors, achieving promising results on a Chinese-English translation dataset; Elbayad et al. (2018) have used sentence-level and token-level reward distributions to "smooth" the single ground truth. Chousa et al. (2018) have similarly leveraged a token-level smoother.

In a recent paper, Denkowski and Neubig (2017) have achieved state-of-the-art translation accuracy by leveraging a variety of techniques which include: dropout (Srivastava et al., 2014), lexicon bias (Arthur et al., 2016), pre-translation (Niehues et al., 2016), data bootstrapping (Chen et al., 2016), byte-pair encoding (Sennrich et al., 2016) and ensembles of independent models (Rokach, 2010).

However, to our knowledge none of the mentioned approaches have explicitly attempted to leverage the embeddings of the ground-truth tokens as targets. For this reason, in this paper we explore regressing toward pre-trained word embeddings as an attempt to capture contextual properties and achieve improved model regularization.

3 Model

3.1 Seq2seq baseline

The model is a standard NMT model with attention in which we use RNNs for the encoder and decoder. Following the notation of (Bahdanau et al., 2015), the RNN in the decoder generates a sequence of hidden vectors, $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$, given the context vector, the previous hidden state $\mathbf{s}_{j-1}$ and the previous predicted word $\mathbf{y}_{j-1}$:

$$\mathbf{s}_j = dec_{rnn}(\mathbf{s}_{j-1}, \mathbf{y}_{j-1}, \mathbf{c}_j) \quad j = 1, \ldots, m \quad (1)$$

where y_0 and s_0 are initializations for the state and label chains. Each hidden vector $\mathbf{s}_j$ (of parameter size S) is then linearly transformed into a vector of vocabulary size, V, and a softmax layer converts it into a vector of probabilities (Eq. 2), where W (a matrix of size $V \times S$) and b (a vector of size $V \times 1$) are learnable parameters. The predicted conditional probability distribution over the words in the target vocabulary, $\mathbf{p}_j$, is given as:

$$\mathbf{p}_j = softmax(\mathbf{W}\mathbf{s}_j + \mathbf{b}) \quad (2)$$

As usual, training attempts to minimize the negative log-likelihood (NLL), defined as:

$$NLL_{loss} = -\sum_{j=1}^{m} \log(\mathbf{p}_j(\mathbf{y}_j)) \quad (3)$$

where $\mathbf{p}_j(\mathbf{y}_j)$ notes the probability of ground-truth word $\mathbf{y}_j$. The NLL loss is minimized when the probability of the ground truth is one and that of all other words is zero, treating all predictions different from the ground truth as equally incorrect.

3.2 ReWE

Pre-trained word embeddings (Pennington et al., 2014; Bojanowski et al., 2017; Mikolov et al., 2013) capture the contextual similarities of words, typically by maximizing the probability of word w_{t+k} to occur in the context of center word w_t. This probability can be expressed as:

$$\begin{gathered} p(w_{t+k}|w_t), \quad -c \leq k \leq c, k \neq 0 \\ t = 1, \ldots, T \end{gathered} \tag{4}$$

where c is the size of the context and T is the total number of words in the training set. Traditionally, word embeddings have only been used as input representations. In this paper, we instead propose using them in output as part of the training objective, in the hope of achieving regularization and improving prediction accuracy. Building upon the baseline model presented in Section 3.1, we have designed a new "joint learning" setting: our decoder still predicts the probability distribution over the vocabulary, $\mathbf{p}_j$ (Eq. 2), while simultaneously regressing the same shared $\mathbf{s}_j$ to the ground-truth word embedding, $e(\mathbf{y}_j)$. The ReWE module consists of two linear layers with a Rectified Linear Unit (ReLU) in between, outputting a vector $\mathbf{e}_j$ of word embedding size (Eq. 5). Please note that adding this extra module adds negligible computational costs and training time. Full details of this module are given in the supplementary material.

$$\begin{aligned} \mathbf{e}_j &= ReWE(\mathbf{s}_j) \\ &= \mathbf{W}_2(ReLU(\mathbf{W}_1\mathbf{s}_j + \mathbf{b}_1)) + \mathbf{b}_2 \end{aligned} \tag{5}$$

The training objective is a numerical loss, l (Eq. 6), computed between the output vector, $\mathbf{e}_j$, and the ground-truth embedding, $e(\mathbf{y}_j)$:

$$ReWE_{loss} = l(\mathbf{e}_j, e(\mathbf{y}_j)) \tag{6}$$

In the experiment, we have explored two cases for the $ReWE_{loss}$: the minimum square error (MSE)[1] and the cosine embedding loss (CEL)[2]. Finally, the NLL_{loss} and the $ReWE_{loss}$ are combined to form the training objective using a positive trade-off coefficient, λ:

$$Loss = NLL_{loss} + \lambda ReWE_{loss} \tag{7}$$

As mentioned in the Introduction, at inference time we ignore the ReWE output, $\mathbf{e}_j$, and the model operates as a standard NMT system.

Dataset	Size	Sources
IWSLT16 en-fr	219,777	TED talks
IWSLT16 cs-en	114,243	TED talks
WMT16 eu-en	89,413	IT-domain data

Dataset	Validation set	Test set
en-fr	TED test 2013+2014	TED test 2015+2016
cs-en	TED test 2012+2013	TED test 2015+2016
eu-en	Sub-sample of PaCo	IT-domain test

Table 1: Top: parallel training data. Bottom: validation and test sets.

4 Experiments

We have developed our models building upon the OpenNMT toolkit (Klein et al., 2017)[3]. For training, we have used the same settings as (Denkowski and Neubig, 2017). We have also explored the use of sub-word units learned with byte pair encoding (BPE) (Sennrich et al., 2016). All the preprocessing steps, hyperparameter values and training parameters are described in detail in the supplementary material to ease reproducibility of our results.

We have evaluated these systems over three publicly-available datasets from the 2016 ACL Conference on Machine Translation (WMT16)[4] and the 2016 International Workshop on Spoken Language Translation (IWSLT16)[5]. Table 1 lists the datasets and their main features. Despite having nearly 90,000 parallel sentences, the eu-en dataset only contains 2,000 human-translated sentences; the others are translations of Wikipedia page titles and localization files. Therefore, we regard the eu-en dataset as very low-resource.

In addition to the seq2seq baseline, we have compared our results with those recently reported by Denkowski and Neubig for non-ensemble models (2017). For all models, we report the BLEU scores (Papineni et al., 2002), with the addition of selected comparative examples. Two contrastive experiments are also added in supplementary notes.

4.1 Results

As a preliminary experiment, we have carried out a sensitivity analysis to determine the optimal value of the trade-off coefficient, λ (Eq. 6), using the en-fr validation set. The results are shown in Figure 2, where each point is the average of three runs trained with different seeds. The figure shows that

---

[1] https://pytorch.org/docs/stable/nn.html#torch.nn.MSELoss

[2] https://pytorch.org/docs/stable/nn.html#torch.nn.CosineEmbeddingLoss

[3] Our code can be found at: https://github.com/ijauregiCMCRC/ReWE_NMT

[4] WMT16: http://www.statmt.org/wmt16/

[5] IWSLT16: https://workshop2016.iwslt.org/

Models	**en-fr**		**cs-en**		**eu-en**	
	Word	BPE	Word	BPE	Word	BPE
(Denkowski and Neubig, 2017)	33.60	34.50	21.00	22.60		
(Denkowski and Neubig, 2017) + Dropout	34.5	34.70	21.4	23.60		
(Denkowski and Neubig, 2017) + Lexicon	33.9	34.80	20.6	22.70		
(Denkowski and Neubig, 2017) + Pre-translation	N/A	34.90	N/A	**23.80**		
(Denkowski and Neubig, 2017) + Bootstrapping	34.40	35.20	21.60	23.60		
Our baseline	34.16	34.09	20.57	22.69	12.14	17.17
Our baseline + ReWE (CEL) ($\lambda = 20$)	**35.52**	**35.22**	**21.83**	23.60	**13.73**	**19.71**

Table 2: BLEU scores over the test sets. Average of 10 models independently trained with different seeds.

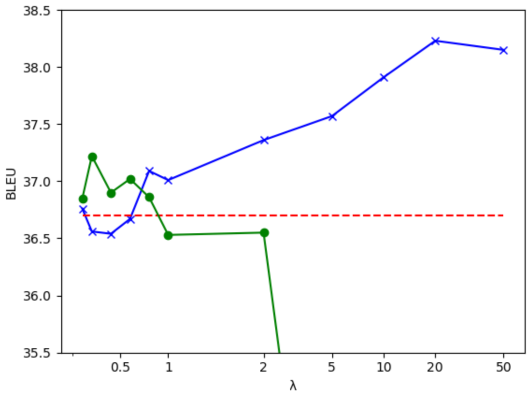


Figure 2: BLEU scores of three models over the en-fr validation set for different λ values: baseline (red, dashed), baseline + ReWE (MSE) (green, ●), baseline + ReWE (CEL) (blue, $\times$). Each point in the graph is an average of 3 independently trained models.

Src:	Hautatu Kontrol panela → Programa lehenetsiak , eta aldatu bertan .
Ref:	Go to Control Panel → Default programs , and change it there .
Baseline:	Select the Control Panel → program , and change .
Baseline + ReWE:	Select the Control Panel → Default Program , and change it .

Table 3: Translation example from the eu-en test set.

the MSE loss has outperformed slightly the baseline for small values of λ (< 1), but the BLEU score has dropped drastically for larger values. Conversely, the CEL loss has increased steadily with λ, reaching 38.23 BLEU points for $\lambda = 20$, with a marked improvement of 1.53 points over the baseline. This result has been encouraging and therefore for the rest of the experiments we have used CEL as the $ReWE_{loss}$ and kept the value of λ to 20. In Section 4.3, we further discuss the behavior of CEL and MSE.

Table 2 reports the results of the main experiment for all datasets. The values of our experiments are for blind runs over the test sets, averaged over 10 independent runs with different seeds. The results show that adding ReWE has significantly improved the baseline in all cases, with an average of 1.46 BLEU points. In the case of the eu-en dataset, the improvement has reached 2.54 BLEU points. We have also run unpaired t-tests between our baseline and ReWE, and the differences have proved statistically significant (p-values < 0.05) in all cases. Using BPE has proved beneficial for the cs-en and eu-en pairs, but not for the en-fr pair. We speculate that English and French may be closer to each other at word level and, therefore, less likely to benefit from the use of sub-word units. Conversely, Czech and Basque are morphologically very rich, justifying the improvements with BPE.

Table 2 also shows that our model has outperformed almost all the state-of-the-art results reported in (Denkowski and Neubig, 2017) (dropout, lexicon bias, pre-translation, and bootstrapping), with the only exception of the pre-translation case for the cs-en pair with BPE. This shows that the proposed model is competitive with contemporary NMT techniques.

4.2 Qualitative comparison

To further explore the improvements obtained with ReWE, we have qualitatively compared several translations provided by the baseline and the baseline + ReWE (CEL), trained with identical seeds. Overall, we have noted a number of instances where ReWE has provided translations with more information from the source (higher adequacy). For reasons of space, we report only one example in Table 3, but more examples are available in the supplementary material. In the example, the baseline has chosen a generic word, “program”, while ReWE has been capable of correctly predicting “Default Program” and being specific about the object, “it”.

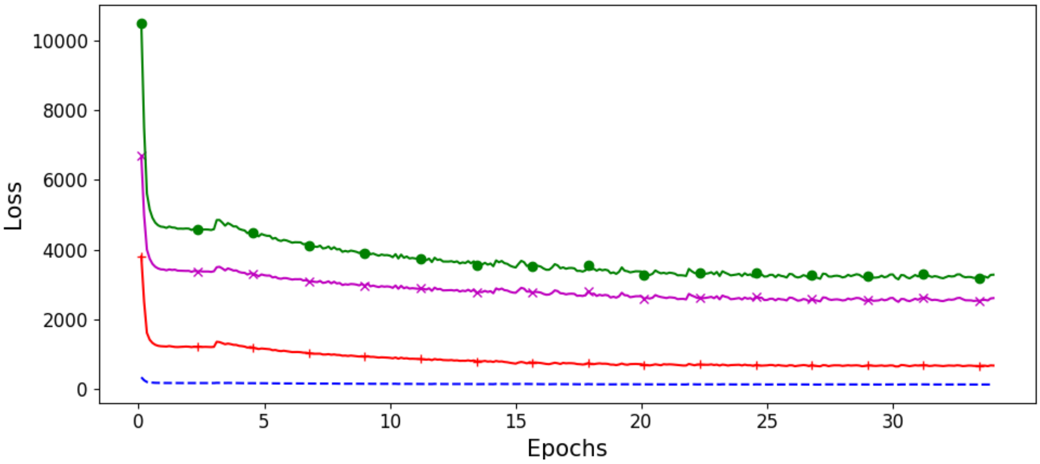


Figure 3: Plot of the values of various loss functions during training of our model over the en-fr training set: green, ●: training loss (NLL + ($\lambda = 20$) ReWE (CEL); Eq.7); red, +: NLL loss; blue, dashed: ReWE (CEL) loss; magenta, ×: ReWE (CEL) loss scaled by $\lambda = 20$. Each point in the graph is an average value of the corresponding loss over 25,000 sentences.

4.3 Discussion

To further explore the behaviour of the ReWE loss, Figure 3 plots the values of the NLL and ReWE (CEL) losses during training of our model over the en-fr training set. The natural values of the ReWE (CEL) loss (blue, dashed) are much lower than those of the NLL loss (red, +), and thus its contribution to the gradient is likely to be limited. However, when scaled up by a factor of $\lambda = 20$ (magenta, ×), its influence on the gradient becomes more marked. Empirically, both the NLL and ReWE (CEL) losses decrease as the training progresses and the total loss (green, ●) decreases. As shown in the results, this combined training objective has been able to lead to improved translation results.

Conversely, the MSE loss has not exhibited a similarly smooth behaviour (supplementary material). Even when brought to scale with the NLL loss, it shows much larger fluctuations as the training progresses. In particular, it shows major increases at the re-starts of the optimizer for the simulated annealing that are not compensated for by the rest of the training. It is easy to speculate that the MSE loss is much more sensitive than the cosine distance to the changes in the weights caused by dropout and the re-starts. As such, it seems less suited for use as training objective.

5 Conclusion

In this paper, we have proposed a new regularization technique for NMT (ReWE) based on a joint learning setting in which a seq2seq model simultaneously learns to a) predict the next word in the translation and b) regress toward its word embedding. The results over three parallel corpora have shown that ReWE has consistently improved over both its baseline and recent state-of-the-art results from the literature. As future work, we plan to extend our experiments to better understand the potential of the proposed regularizer, in particular for unsupervised NMT (Artetxe et al., 2018; Lample et al., 2018).

Acknowledgments

We would like to acknowledge the financial support received from the Capital Markets Cooperative Research Centre (CMCRC), an industry-led research initiative of the Australian Government. We would also like to thank Ben Hachey, Michael Nolan and Nadia Shnier for their careful reading of our paper and their insightful comments. Finally, we are grateful to the anonymous reviewers for all their comments and suggestions.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Empirical Meth-*

ods in Natural Language Processing, pages 1557–1567.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, pages 135–146.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*, pages 1724–1734.

Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. 2018. Training neural machine translation using word embedding-based loss. *arXiv preprint arXiv:1807.11219*.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27. Empirical Methods in Natural Language Processing.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2018. Token-level and sequence-level loss smoothing for rnn language models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2094–2103.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 567–573.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 123–135.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *Empirical Methods in Natural Language Processing*, pages 5039–5049.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 1412–1421.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 332–338.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. *arXiv preprint arXiv:1610.05243*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, pages 1532–1543.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Regularizing neural machine translation by target-bidirectional agreement. *arXiv preprint arXiv:1808.04064*.

Lost in Machine Translation: A Method to Reduce Meaning Loss

Reuben Cohn-Gordon
Stanford
reubencg@stanford.edu

Noah D. Goodman
Stanford
ngoodman@stanford.edu

Abstract

A desideratum of high-quality translation systems is that they preserve meaning, in the sense that two sentences with different meanings should not translate to one and the same sentence in another language. However, state-of-the-art systems often fail in this regard, particularly in cases where the source and target languages partition the "meaning space" in different ways. For instance, "I cut my finger." and "I cut my finger off." describe different states of the world but are translated to French (by both *Fairseq* and *Google Translate*) as "Je me suis coupé le doigt.", which is ambiguous as to whether the finger is detached. More generally, translation systems are typically many-to-one (non-injective) functions from source to target language, which in many cases results in important distinctions in meaning being lost in translation. Building on Bayesian models of informative utterance production, we present a method to define a less ambiguous translation system in terms of an underlying pre-trained neural sequence-to-sequence model. This method increases injectivity, resulting in greater preservation of meaning as measured by improvement in cycle-consistency, without impeding translation quality (measured by BLEU score).

1 Many-to-One Translations

Languages differ in what meaning distinctions they must mark explicitly. As such, translations risk mapping from a form in one language to a more ambiguous form in another. For example, the definite (1) and indefinite (2) both translate (under *Fairseq* and *Google Translate*) to (3) in French, which is ambiguous in definiteness.

The animals run fast. (1)

Animals run fast. (2)

Les animaux courent vite (3)

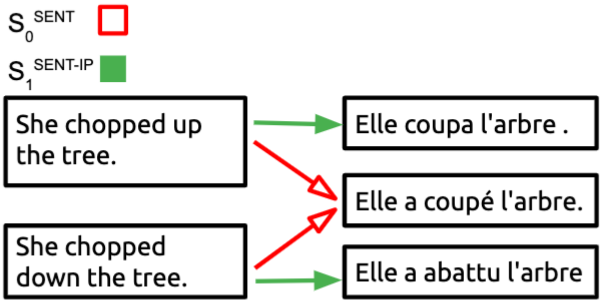


Figure 1: State-of-the-art neural image captioner S_0^{SNT} loses a meaning distinction which *informative* model $S_1^{\text{SNT-IP}}$ preserves.

Survey To evaluate the nature of this problem, we explored a corpus[1] of 500 pairs of distinct English sentences which map to a single German one (the evaluation language in section 2.3). We identify a number of common causes for the many-to-one maps. Two frequent types of verbal distinction lost when translating to German are tense (54 pairs, e.g. "...others {were, have been} introduced .") and modality (16 pairs, e.g. "...prospects for this year {*could*, *might*} be better."), where German "können" can express both epistemic and ability modality, distinguished in English with "might" and "could" respectively. Owing to English's large vocabulary, lexical difference in verb (31 pairs, e.g. "arise" vs. "emerge"), noun (56 pairs, e.g. "mystery" vs. "secret"), adjective (47 pairs, e.g. "unaffected" vs. "untouched") or deictic/pronoun (32 pairs, usually "this" vs "that") are also common. A large number of the pairs differ instead either orthographically, or in other ways that do not correspond to a clear semantic distinc-

---

[1] Generated by selecting short sentences from the Brown corpus (Kučera and Francis, 1967), translating them to German, and taking the best two candidate translations back into English, if these two themselves translate to a single German sentence. Translation in both directions was done with Fairseq.

Proceedings of NAACL-HLT 2019, pages 437–441
Minneapolis, Minnesota, June 2 - June 7, 2019.

A	He is wearing glasses.
B	He wears glasses.
$S_0^{\text{SNT}}(A)$	Er trägt eine Brille.
$S_0^{\text{SNT}}(B)$	Er trägt eine Brille .
$S_1^{\text{SNT-IP}}(A)$	Er trägt jetzt eine Brille.
$S_1^{\text{SNT-IP}}(B)$	Er hat eine Brille.

Figure 2: Similar to Figure 1, S_0^{SNT} collapses two English sentences into a single German one, whereas $S_1^{\text{SNT-IP}}$ distinguishes the two in German.

tion (e.g. "She had {*taken*, *made*} a decision.").

Our approach While languages differ in what distinctions they are *required* to express, all are usually capable of expressing any given distinction when desired. As such, meaning loss of the kind discussed above is, in theory, avoidable. To this end, we propose a method to reduce meaning loss by applying the Rational Speech Acts (RSA) model of an informative speaker to translation. RSA has been used to model natural language *pragmatics* (Goodman and Frank, 2016), and recent work has shown its applicability to image captioning (Andreas and Klein, 2016; Vedantam et al., 2017; Mao et al., 2016), another sequence-generation NLP task. Here we use RSA to define a translator which reduces many-to-one mappings and consequently meaning loss, in terms of a pretrained neural translation model. We introduce a strategy for performing inference efficiently with this model in the setting of translation, and show gains in *cycle-consistency*[2] as a result. Moreover, we obtain improvements in translation quality (BLEU score), demonstrating that the goal of meaning preservation directly yields improved translations.

2 Meaning Preservation as Informativity

In the RSA framework, speakers and listeners, modeled as Bayesian agents, reason about each other in a nested fashion. We refer to listeners and speakers which do not reason about another agent as L_0 and S_0 respectively, and an agent which reasons about another agent as L_1 or S_1. For instance, an informative speaker model S_1 is given a state $w \in W$, and chooses an utterance $u \in U$ to convey w to S_1's model of a listener. By contrast, S_0 chooses utterances without a listener model in mind — its behavior might be determined by a semantics, or in our case, by a pretrained neural model.

For translation, the state space W is a set of source language sentences (sequences of words in the language), while U is a set of target language sentences. S_1's goal is to choose a translation u which allows a listener to pick out the source sentence w from among the set of distractors. This informative behavior discourages many-to-one maps that a non-informative translation model S_0 might allow.

$\mathbf{S_0}$ Model BiLSTMs with attention (Bahdanau et al., 2014), and more recently CNNs (Gehring et al., 2016) and entirely attention based models (Vaswani et al., 2017) constitute the state-of-the-art architectures in neural machine translation . All of these systems, once trained end-to-end on aligned data, can be viewed as a conditional distribution[3] $S_0^{\text{WD}}(wd|w, c)$, for a word *wd* in the target language, a source language sentence w, and a partial sentence c in the target language. S_0^{WD} yields a distribution S_0^{SNT} over full sentences[4]:

$$S_0^{\text{SNT}}(u|w, c) = \prod_t S_0^{\text{WD}}(u[t]|w, c + u[: t]) \quad (4)$$

S_0^{SNT} returns a distribution over continuations of c into full target language sentences[5]. To obtain a sentence from S_0^{SNT} given a source language sentence s, one can greedily choose the highest probability word from S_0^{WD} at each timestep, or explore a beam of possible candidates. We implement S_0^{WD} (in terms of which all our other models are defined) using *Fairseq*'s publicly available[6] pretrained Transformer models for English-German, and for German-English train a CNN using *Fairseq*.

2.1 Explicit Distractors

We first describe a sentence level, *globally pragmatic* model $S_1^{\text{SNT-GP}}$ for the simple case where

---

[2] Formally, say that a pair of functions $f : A \to B$, $g : B \to A$ is cycle-consistent if $g \cdot f = id$, the identity function. If f is *not* one-to-one, then (f, g) is not cycle-consistent. (Note however that when A and B are infinite, the converse does not hold: even if f and g are both one-to-one, (f, g) need not be cycle-consistent, since many-to-one maps between infinite sets are not necessarily bijective.)

[3] We use $S_{0/1}^{WD}$ and $S_{0/1}^{SNT}$ respectively to distinguish word and sentence level speaker models

[4] Python list indexing conventions are used, "+" means concatenation of list to element or list

[5] In what follows, we omit c when it is empty, so that $S_0^{\text{SNT}}(u|w)$ is the probability of sentence u given w

[6] `https://github.com/pytorch/fairseq`

a source language sentence needs to be distinguished from a presupplied distractor[7] (as in the pairs shown in figures (2) and (1)). We use this model as a stepping stone to one which requires an input sentence in the source language only, and no distractors. We begin by defining a listener L_1^{SNT}, which receives a target language sentence u and infers which sentence $w \in W$ (a presupplied set such as the pair (1) and (2)) would have resulted in the pretrained neural model S_0^{SNT} producing u:

$$L_1^{\text{SNT}}(w|u) \propto \frac{S_0^{\text{SNT}}(u|w)}{\sum_{w' \in W} S_0^{\text{SNT}}(u|w')} \tag{5}$$

This allows $S_1^{\text{SNT-GP}}$ to be defined in terms of L_1^{SNT}, where U is the set of all possible target language sentences[8]:

$$S_1^{\text{SNT-GP}}(u|w) = \frac{S_0^{\text{SNT}}(u|w) L_1^{\text{SNT}}(w|u)^{\alpha}}{\sum_{u' \in U} S_0^{\text{SNT}}(u'|w) L_1^{\text{SNT}}(w|u')^{\alpha}} \tag{6}$$

The key property of this model is that, for $W = \{A, B\}$, when translating A, $S_1^{\text{SNT-GP}}$ prefers translations of A that are unlikely to be good translations of B. So for pairs like (1) and (2), $S_1^{\text{SNT-GP}}$ is compelled to produce a translation for the former that reflects its difference from the latter, and vice versa.

Inference Since U is an infinite set, exactly computing the most probable utterance under $S_1^{\text{SNT-GP}}(\cdot|w)$ is intractable. Andreas and Klein (2016) and Mao et al. (2016) perform approximate inference by sampling the subset of U produced by a beam search from S_0^{SNT}. Vedantam et al. (2017) and Cohn-Gordon et al. (2018) employ a different method, using an *incremental* model $S_1^{\text{SNT-IP}}$ as an approximation of $S_1^{\text{SNT-GP}}$ on which inference can be tractably performed.

$S_1^{\text{SNT-IP}}$ considers informativity not over the whole set of utterances, but instead at each decision of the next word in the target language sentence. For this reason, the incremental method avoids the problem of lack of beam diversity encountered when subsampling from S_0^{SNT}, which becomes especially bad when producing long sequences, as is often the case in translation. $S_1^{\text{SNT-IP}}$ is defined as the product of informative decisions, specified by S_1^{WD} (itself defined in terms of L_1^{WD}), which are defined analogously to (6) and (5).

$$L_1^{\text{WD}}(w|wd, c) \propto S_0^{\text{WD}}(wd|w, c) \tag{7}$$

$$S_1^{\text{WD}}(wd|w, c) \propto S_0^{\text{WD}}(wd|w, c) * L_1^{\text{WD}}(w|wd, c)^{\alpha} \tag{8}$$

$$S_1^{\text{SNT-IP}}(u|w, c) = \prod_t S_1^{\text{WD}}(u[t]|w, c + u[:t]) \tag{9}$$

Examples $S_1^{\text{SNT-IP}}$ is able to avoid many-to-one mappings by choosing more informative translations. For instance, its translation of (1) is "Ces animaux courent vite" (*These* animals run fast.). See figures (1) and (2) for other examples of many-to-one mappings under S_0^{SNT} avoided by $S_1^{\text{SNT-IP}}$.

2.2 Avoiding Explicit Distractors

While $S_1^{\text{SNT-IP}}$ can disambiguate between pairs of sentences, it has two shortcomings. First, it requires one (or more) distractors to be provided, so translation is no longer fully automatic. Second, because the distractor set W consists of only a pair (or finite set) of sentences, $S_1^{\text{SNT-IP}}$ only cares about being informative up to the goal of distinguishing between these sentences. Intuitively, total meaning preservation is achieved by a translation which distinguishes the source sentence w from every other sentence in the source language which differs in meaning.

Both of these problems can be addressed by introducing a new "cyclic" globally pragmatic model $S_1^{\text{SNT-CGP}}$ which reasons not about L_1^{SNT} but about a pretrained translation model from target language to source language, which we term L_0^{SNT}.

$$S_1^{\text{SNT-CGP}}(u|w) \propto S_0^{\text{SNT}}(u|w) L_0^{\text{SNT}}(w|u)^{\alpha} \tag{10}$$

$S_1^{\text{SNT-CGP}}$ is like $S_1^{\text{SNT-GP}}$, but its goal is to produce a translation which allows a listener model (now L_0^{SNT}) to infer the original sentence, not among a small set of presupplied possibilities, but among *all source language sentences*. As such, an optimal translation u of w under $S_1^{\text{SNT-CGP}}$ has high probability of being generated by S_0^{SNT} and

---

[7] Implementations for all models are available to `https://github.com/reubenharry/pragmatic-translation`

[8] α is a hyperparameter of $S_1^{\text{SNT-GP}}$; as it increases, the model cares more about being informative and less about producing a reasonable translation.

high probability of being back-translated to w by L_0^{SNT}. $S_1^{\text{SNT-CGP}}$ is very closely related to reconstruction methods, e.g. (Tu et al., 2017).

Incremental Model Exact inference is again intractable, though as with $S_1^{\text{SNT-GP}}$, it is possible to approximate by subsampling from S_0^{SNT}. This is very close to the approach taken by (Li et al., 2016), who find that reranking a set of outputs by probability of recovering input "dramatically decreases the rate of dull and generic responses." in a question-answering task. However, because the subsample is small relative to U, they use this method in conjunction with a diversity increasing decoding algorithm.

As in the case with explicit distractors, we instead opt for an incremental model, now $S_1^{\text{SNT-CIP}}$ which approximates $S_1^{\text{SNT-CGP}}$. The definition of $S_1^{\text{SNT-CIP}}$ (12) is more complex than the incremental model with explicit distractors ($S_1^{\text{SNT-IP}}$) since L_0^{WD} must receive complete sentences, rather than partial ones like L_1^{WD}. As such, we need to marginalize over continuations k of partial sentences in the target language:

$$S_1^{\text{WD-C}}(wd|w,c) \propto S_0^{\text{WD}}(wd|w,c)* \\ \sum_k (L_0^{\text{SNT}}(w|c+wd+k) S_0^{\text{SNT}}(k|w,c+wd)) \tag{11}$$

$$S_1^{\text{SNT-CIP}}(u|w,c) = \prod_t S_1^{\text{WD-C}}(u[t]|w,c+u[:t]) \tag{12}$$

Since the sum over continuations of c in (11) is intractable to compute exactly, we approximate it by a single continuation, obtained by greedily unrolling S_0^{SNT}. The whole process of generating a new word wd of the translation from a sequence c and a source language sentence w is as follows: first use S_0^{WD} to generate a set of candidates for the next word (in practice, we only consider two, for efficiency). For each of these, use S_0^{SNT} to greedily unroll a full target language sentence from $c+wd$, namely $c+wd+k$, and rank each wd by the probability $L_0^{\text{SNT}}(w|c+wd+k)$.

2.3 Evaluating the Informative Translator

Our objective is to improve meaning preservation without detracting from translation quality in other regards (e.g. grammaticality). We conduct our evaluations on English to German translation, making use of publicly available pre-trained English-German and German-English Fairseq models. The pragmatic model we evaluate is $S_1^{\text{SNT-CIP}}$ since, unlike $S_1^{\text{SNT-IP}}$, it is not necessary to hand-supply a distractor set of source language sentences.

An example of the behavior of $S_1^{\text{SNT-CIP}}$ and S_0^{SNT} on of our test sentences is shown below; S_0^{SNT} is able to preserve the phrase "world's eyes", which S_0^{SNT} translates merely as "world":

- Source sentence: Isolation keeps the world's eyes off Papua.
- Reference translation: Isolation hält die Augen der Welt fern von Papua.
- S_0^{SNT}: Die Isolation hält die Welt von Papua fern.
- $S_1^{\text{SNT-CIP}}$: Die Isolation hält die Augen der Welt von Papua fern.

We use cycle-consistency as a measure of meaning preservation, since the ability to recover the original sentence requires meaning distinctions not to be collapsed. In evaluating cycle-consistency, it is important to use a separate target-source translation mechanism than the one used to define $S_1^{\text{SNT-CIP}}$. Otherwise, the system has access to the model which evaluates it and may improve cycle-consistency without producing meaningful target language sentences. For this reason, we translate German sentences (produced by S_0^{SNT} or $S_1^{\text{SNT-CIP}}$) back to English with *Google Translate*. To measure cycle-consistency, we use the BLEU metric (implemented with sacreBLEU (Post, 2018)), with the original sentence as the reference.

However, this improvement of cycle consistency, especially with a high value of α, may come at the cost of translation quality. Moreover, it is unclear whether BLEU serves as a good metric for evaluating sentences of a single language. To further ensure that translation quality is not compromised by $S_1^{\text{SNT-CIP}}$, we evaluate BLEU scores of the German sentences it produces. This requires evaluation on a corpus of aligned sentences, unlike the sentences collected from the Brown corpus in section 1[9].

---

[9] While we find that $S_1^{\text{SNT-CIP}}$ improves cycle-consistency for the Brown corpus over S_0^{SNT}, we have no way to establish whether this comes at the cost of translation quality.

Model	Cycle	Translate
S_0^{SNT}	43.35	37.42
$S_1^{\text{SNT-CIP}}$	**47.34**	**38.29**

Table 1: BLEU score on cycle-consistency and translation for WMT, across baseline and informative models. Greedy unrolling and $\alpha = 0.1$

We perform both evaluations (cycle-consistency and translation) on 750 sentences[10] of the 2018 English-German WMT News test-set.[11] We use greedy unrolling in all models (using beam search is a goal for future work). For α (which represents the trade-off between informativity and translation quality) we use 0.1, obtained by tuning on validation data.

Results As shown in table (1), $S_1^{\text{SNT-CIP}}$ improves over S_0^{SNT} not only in cycle-consistency, but in translation quality as well. This suggests that the goal of preserving information, in the sense defined by $S_1^{\text{SNT-CGP}}$ and approximated by $S_1^{\text{SNT-CIP}}$, is important for translation quality.

3 Conclusions

We identify a shortcoming of state-of-the-art translation systems and show that a version of the RSA framework's informative speaker S_1, adapted to the domain of translation, alleviates this problem in a way which improves not only cycle-consistency but translation quality as well. The success of $S_1^{\text{SNT-CIP}}$ on two fairly similar languages raises the question of whether improvements will increase for more distant language pairs, in which larger scale differences exist in what information is obligatorily represented - this is a direction for future work.

Acknowledgements

Thanks to the reviewers for their substantive comments, and to Daniel Fried and Jacob Andreas for many helpful discussions during the development of this project.

---

[10] Our implementation of $S_1^{\text{SNT-CIP}}$ was not efficient, and we could not evaluate on more sentences for reasons of time.

[11] `http://www.statmt.org/wmt18/translation-task.html`

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2018. Pragmatically informative image captioning with character-level inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 439–443. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Noah D Goodman and Michael C Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829.

Henry Kučera and Winthrop Nelson Francis. 1967. *Computational analysis of present-day American English*. Dartmouth Publishing Group.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Computer Vision and Pattern Recognition (CVPR)*, volume 3.

Bi-Directional Differentiable Input Reconstruction for Low-Resource Neural Machine Translation

Xing Niu
University of Maryland
xingniu@cs.umd.edu

Weijia Xu
University of Maryland
weijia@cs.umd.edu

Marine Carpuat
University of Maryland
marine@cs.umd.edu

Abstract

We aim to better exploit the limited amounts of parallel text available in low-resource settings by introducing a differentiable reconstruction loss for neural machine translation (NMT). This loss compares original inputs to reconstructed inputs, obtained by back-translating translation hypotheses into the input language. We leverage differentiable sampling and bi-directional NMT to train models end-to-end, without introducing additional parameters. This approach achieves small but consistent BLEU improvements on four language pairs in both translation directions, and outperforms an alternative differentiable reconstruction strategy based on hidden states.

1 Introduction

Neural Machine Translation (NMT) performance degrades sharply when parallel training data is limited (Koehn and Knowles, 2017). Past work has addressed this problem by leveraging monolingual data (Sennrich et al., 2016a; Ramachandran et al., 2017) or multilingual parallel data (Zoph et al., 2016; Johnson et al., 2017; Gu et al., 2018a). We hypothesize that the traditional training can be complemented by better leveraging limited training data. To this end, we propose a new training objective for this model by augmenting the standard translation cross-entropy loss with a **differentiable input reconstruction loss** to further exploit the source side of parallel samples.

Input reconstruction is motivated by the idea of round-trip translation. Suppose sentence $\boldsymbol{f}$ is translated forward to $\boldsymbol{e}$ using model $\boldsymbol{\theta}_{fe}$ and then translated back to $\hat{\boldsymbol{f}}$ using model $\boldsymbol{\theta}_{ef}$, then $\boldsymbol{e}$ is more likely to be a good translation if the distance between $\hat{\boldsymbol{f}}$ and $\boldsymbol{f}$ is small (Brislin, 1970). Prior work applied round-trip translation to monolingual examples and sampled the intermediate translation $\boldsymbol{e}$ from a K-best list generated by model $\boldsymbol{\theta}_{fe}$ using beam search (Cheng et al., 2016; He et al., 2016). However, beam search is not differentiable which prevents back-propagating reconstruction errors to $\boldsymbol{\theta}_{fe}$. As a result, reinforcement learning algorithms, or independent updates to $\boldsymbol{\theta}_{fe}$ and $\boldsymbol{\theta}_{ef}$ were required.

In this paper, we focus on the problem of making input reconstruction differentiable to simplify training. In past work, Tu et al. (2017) addressed this issue by reconstructing source sentences from the decoder's hidden states. However, this reconstruction task can be artificially easy if hidden states over-memorize the input. This approach also requires a separate auxiliary reconstructor, which introduces additional parameters.

We propose instead to combine benefits from differentiable sampling and bi-directional NMT to obtain a compact model that can be trained end-to-end with back-propagation. Specifically,

- Translations are sampled using the Straight-Through Gumbel Softmax (STGS) estimator (Jang et al., 2017; Bengio et al., 2013), which allows back-propagating reconstruction errors.

- Our approach builds on the bi-directional NMT model (Niu et al., 2018; Johnson et al., 2017), which improves low-resource translation by jointly modeling translation in both directions (e.g., Swahili ↔ English). A single bi-directional model is used as a translator and a reconstructor (i.e. $\boldsymbol{\theta}_{ef} = \boldsymbol{\theta}_{fe}$) without introducing more parameters.

Experiments show that our approach outperforms reconstruction from hidden states. It achieves consistent improvements across various low-resource language pairs and directions, showing its effectiveness in making better use of limited parallel data.

Proceedings of NAACL-HLT 2019, pages 442–448
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Background

Using round-trip translations ($\boldsymbol{f} \to \boldsymbol{e} \to \hat{\boldsymbol{f}}$) as a training signal for NMT usually requires auxiliary models to perform back-translation and cannot be trained end-to-end without reinforcement learning. For instance, Cheng et al. (2016) added a reconstruction loss for monolingual examples to the training objective. He et al. (2016) evaluated the quality of $\boldsymbol{e}$ by a language model and $\hat{\boldsymbol{f}}$ by a reconstruction likelihood. Both approaches have symmetric forward and backward translation models which are updated alternatively. This require policy gradient algorithms for training, which are not always stable.

Back-translation (Sennrich et al., 2016a) performs half of the reconstruction process, by generating a synthetic source side for monolingual target language examples: $\boldsymbol{e} \to \hat{\boldsymbol{f}}$. It uses an auxiliary backward model to generate the synthetic data but only updates the parameters of the primary forward model. Iteratively updating forward and backward models (Zhang et al., 2018; Niu et al., 2018) is an expensive solution as back-translations are regenerated at each iteration.

Prior work has sought to simplify the optimization of reconstruction losses by side-stepping beam search. Tu et al. (2017) first proposed to reconstruct NMT input from the decoder's hidden states while Wang et al. (2018a,b) suggested to use both encoder and decoder hidden states to improve translation of dropped pronouns. However, these models might achieve low reconstruction errors by learning to copy the input to hidden states. To avoid copying the input, Artetxe et al. (2018) and Lample et al. (2018) used denoising autoencoders (Vincent et al., 2008) in unsupervised NMT.

Our approach is based instead on the Gumbel Softmax (Jang et al., 2017; Maddison et al., 2017), which facilitates differentiable sampling of sequences of discrete tokens. It has been successfully applied in many sequence generation tasks, including artificial language emergence for multi-agent communication (Havrylov and Titov, 2017), composing tree structures from text (Choi et al., 2018), and tasks under the umbrella of generative adversarial networks (Goodfellow et al., 2014) such as generating the context-free grammar (Kusner and Hernández-Lobato, 2016), machine comprehension (Wang et al., 2017) and machine translation (Gu et al., 2018b).

3 Approach

NMT is framed as a conditional language model, where the probability of predicting target token e_t at step t is conditioned on the previously generated sequence of tokens $\boldsymbol{e}_{<t}$ and the source sequence $\boldsymbol{f}$ given the model parameter $\boldsymbol{\theta}$. Suppose each token is indexed and represented as a one-hot vector, its probability is realized as a softmax function over a linear transformation $a(\boldsymbol{h}_t)$ where $\boldsymbol{h}_t$ is the decoder's hidden state at step t:

$$P(e_t|\boldsymbol{e}_{<t}, \boldsymbol{f}; \boldsymbol{\theta}) = \mathrm{softmax}(a(\boldsymbol{h}_t))^\top e_t. \quad (1)$$

The hidden state is calculated by a neural network g given the embeddings of the previous target tokens $\boldsymbol{e}_{<t}$ in the embedding matrix $\boldsymbol{E}(\boldsymbol{e}_{<t})$ and the context $\boldsymbol{c}_t$ coming from the source:

$$\boldsymbol{h}_t = g(\boldsymbol{E}(\boldsymbol{e}_{<t}), \boldsymbol{c}_t). \quad (2)$$

In our bi-directional model, the source sentence can be either $\boldsymbol{f}$ or $\boldsymbol{e}$ and is respectively translated to $\boldsymbol{e}$ or $\boldsymbol{f}$. The language is marked by a tag (e.g., `<en>`) at the beginning of each source sentence (Johnson et al., 2017; Niu et al., 2018). To facilitate symmetric reconstruction, we also add language tags to target sentences. The training data corpus is then built by swapping the source and target sentences of a parallel corpus and appending the swapped version to the original.

3.1 Bi-Directional Reconstruction

Our bi-directional model performs both forward translation and backward reconstruction. By contrast, uni-directional models require an auxiliary reconstruction module, which introduces additional parameters. This module can be either a decoder-based reconstructor (Tu et al., 2017; Wang et al., 2018a,b) or a reversed dual NMT model (Cheng et al., 2016; He et al., 2016; Wang et al., 2018c; Zhang et al., 2018).

Here the reconstructor, which shares the same parameter with the translator $T(\cdot)$, can also be trained end-to-end by maximizing the log-likelihood of reconstructing $\boldsymbol{f}$:

$$\mathcal{L}_R = \sum_{\boldsymbol{f}} \log P(\boldsymbol{f} \,|\, T(\boldsymbol{f}; \boldsymbol{\theta}); \boldsymbol{\theta}), \quad (3)$$

Combining with the forward translation likelihood

$$\mathcal{L}_T = \sum_{(\boldsymbol{f} \| \boldsymbol{e})} \log P(\boldsymbol{e} \,|\, \boldsymbol{f}; \boldsymbol{\theta}), \quad (4)$$

we use $\mathcal{L} = \mathcal{L}_T + \mathcal{L}_R$ as the final training objective for $\boldsymbol{f} \rightarrow \boldsymbol{e}$. The dual $\boldsymbol{e} \rightarrow \boldsymbol{f}$ model is trained simultaneously by swapping the language direction in bi-directional NMT.

Reconstruction is reliable only with a model that produces reasonable base translations. Following prior work (Tu et al., 2017; He et al., 2016; Cheng et al., 2016), we pre-train a base model with $\mathcal{L}_T$ and fine-tune it with $\mathcal{L}_T + \mathcal{L}_R$.

3.2 Differentiable Sampling

We use differentiable sampling to side-step beam search and back-propagate error signals. We use the Gumbel-Max reparameterization trick (Maddison et al., 2014) to sample a translation token at each time step from the softmax distribution in Equation 1:

$$e_t = \text{one-hot}\left(\arg\max_k \left(a(\boldsymbol{h}_t)_k + G_k\right)\right) \quad (5)$$

where G_k is i.i.d. and drawn from $\text{Gumbel}(0,1)$[1]. We use scaled Gumbel with parameter β, i.e. $\text{Gumbel}(0,\beta)$, to control the randomness. The sampling becomes deterministic (which is equivalent to greedy search) as β approaches 0.

Since $\arg\max$ is not a differentiable operation, we approximate its gradient with the Straight-Through Gumbel Softmax (STGS) (Jang et al., 2017; Bengio et al., 2013): $\nabla_{\boldsymbol{\theta}} e_t \approx \nabla_{\boldsymbol{\theta}} \tilde{e}_t$, where

$$\tilde{e}_t = \text{softmax}\left((a(\boldsymbol{h}_t) + G)/\tau\right) \quad (6)$$

As τ approaches 0, softmax is closer to $\arg\max$ but training might be more unstable. While the STGS estimator is biased when τ is large, it performs well in practice (Gu et al., 2018b; Choi et al., 2018) and is sometimes faster and more effective than reinforcement learning (Havrylov and Titov, 2017).

To generate coherent intermediate translations, the decoder used for sampling only consumes its previously predicted $\hat{\boldsymbol{e}}_{<t}$. This contrasts with the usual *teacher forcing* strategy (Williams and Zipser, 1989), which always feeds in the ground-truth previous tokens $\boldsymbol{e}_{<t}$ when predicting the current token $\hat{e}_t$. With teacher forcing, the sequence concatenation $[\boldsymbol{e}_{<t}; \hat{e}_t]$ is probably coherent at each time step, but the actual predicted sequence $[\hat{\boldsymbol{e}}_{<t}; \hat{e}_t]$ would break the continuity.[2]

# sent.	Training	Dev.	Test
`SW↔EN`	60,570	500	3,000
`TL↔EN`	70,703	704	3,000
`SO↔EN`	68,550	844	3,000
`TR↔EN`	207,021	1,001	3,007

Table 1: Experiments are conducted on four low-resource language pairs, in both translation directions.

4 Experiments

4.1 Tasks and Data

We evaluate our approach on four low-resource language pairs. Parallel data for Swahili↔English (`SW↔EN`), Tagalog↔English (`TL↔EN`) and Somali↔English (`SO↔EN`) contains a mixture of domains such as news and weblogs and is collected from the IARPA MATERIAL program[3], the Global Voices parallel corpus[4], Common Crawl (Smith et al., 2013), and the LORELEI Somali representative language pack (LDC2018T11). The test samples are extracted from the held-out ANALYSIS set of MATERIAL. Parallel Turkish↔English (`TR↔EN`) data is provided by the WMT news translation task (Bojar et al., 2018). We use pre-processed "corpus", "newsdev2016", "newstest2017" as training, development and test sets.[5]

We apply normalization, tokenization, truecasing, joint source-target BPE with 32,000 operations (Sennrich et al., 2016b) and sentence-filtering (length 80 cutoff) to parallel data. Itemized data statistics after preprocessing can be found in Table 1. We report case-insensitive BLEU with the WMT standard '13a' tokenization using SacreBLEU (Post, 2018).

4.2 Model Configuration and Baseline

We build NMT models upon the attentional RNN encoder-decoder architecture (Bahdanau et al., 2015) implemented in the Sockeye toolkit (Hieber et al., 2017). Our translation model uses a bi-directional encoder with a single LSTM layer of size 512, multilayer perceptron attention with a layer size of 512, and word representations of size 512. We apply layer normalization (Ba et al.,

---

[1] i.e. $G_k = -\log(-\log(u_k))$ and $u_k \sim \text{Uniform}(0,1)$.

[2] Sampling with teacher forcing yielded consistently worse BLEU than baselines in preliminary experiments.

[3] `https://www.iarpa.gov/index.php/research-programs/material`

[4] `http://casmacat.eu/corpus/global-voices.html`

[5] `http://data.statmt.org/wmt18/translation-task/preprocessed/`

Model	EN→SW	SW→EN	EN→TL	TL→EN	EN→SO	SO→EN	EN→TR	TR→EN
Baseline	33.60 ± 0.14	30.70 ± 0.19	27.23 ± 0.11	32.15 ± 0.21	12.25 ± 0.08	20.80 ± 0.12	12.90 ± 0.04	15.32 ± 0.11
HIDDEN	33.41 ± 0.15	30.91 ± 0.19	27.43 ± 0.14	32.20 ± 0.35	12.30 ± 0.11	20.72 ± 0.16	12.77 ± 0.11	15.34 ± 0.10
Δ	-0.19 ± 0.24	**0.21 ± 0.14**	**0.19 ± 0.13**	0.04 ± 0.17	0.05 ± 0.11	-0.08 ± 0.12	-0.13 ± 0.13	0.01 ± 0.07
$\beta = 0$	33.92 ± 0.10	31.37 ± 0.18	27.65 ± 0.09	32.75 ± 0.32	12.47 ± 0.08	21.14 ± 0.19	13.26 ± 0.07	15.60 ± 0.19
Δ	**0.32 ± 0.12**	**0.66 ± 0.11**	**0.42 ± 0.16**	**0.59 ± 0.13**	**0.22 ± 0.04**	**0.35 ± 0.15**	**0.36 ± 0.09**	**0.28 ± 0.11**
$\beta = 0.5$	33.97 ± 0.08	31.39 ± 0.09	27.65 ± 0.10	32.65 ± 0.24	12.48 ± 0.09	21.20 ± 0.14	13.16 ± 0.08	15.52 ± 0.07
Δ	**0.37 ± 0.09**	**0.69 ± 0.11**	**0.42 ± 0.11**	**0.50 ± 0.08**	**0.23 ± 0.03**	**0.41 ± 0.13**	**0.25 ± 0.09**	**0.19 ± 0.05**

Table 2: BLEU scores on eight translation directions. The numbers before and after '±' are the mean and standard deviation over five randomly seeded models. Our proposed methods ($\beta = 0/0.5$) achieve small but consistent improvements. ΔBLEU scores are in bold if mean−std is above zero while in red if the mean is below zero.

2016) and add dropout to embeddings and RNNs (Gal and Ghahramani, 2016) with probability 0.2. We train using the Adam optimizer (Kingma and Ba, 2015) with a batch size of 48 sentences and we checkpoint the model every 1000 updates. The learning rate for baseline models is initialized to 0.001 and reduced by 30% after 4 checkpoints without improvement of perplexity on the development set. Training stops after 10 checkpoints without improvement.

The bi-directional NMT model ties source and target embeddings to yield a bilingual vector space. It also ties the output layer's weights and embeddings to achieve better performance in low-resource scenarios (Press and Wolf, 2017; Nguyen and Chiang, 2018).

We train five randomly seeded bi-directional baseline models by optimizing the forward translation objective $\mathcal{L}_T$ and report the mean and standard deviation of test BLEU. We fine-tune baseline models with objective $\mathcal{L}_T + \mathcal{L}_R$, inheriting all settings except the learning rate which is re-initialized to 0.0001. Each randomly seeded model is fine-tuned independently, so we are able to report the standard deviation of ΔBLEU.

4.3 Contrastive Reconstruction Model

We compare our approach with reconstruction from hidden states (HIDDEN). Following the best practice of Wang et al. (2018a), two reconstructors are used to take hidden states from both the encoder and the decoder. The corresponding two reconstruction losses and the canonical translation loss were originally uniformly weighted (i.e. $1, 1, 1$), but we found that balancing the reconstruction and translation losses yields better results (i.e. $0.5, 0.5, 1$) in preliminary experiments.[6]

We use the reconstructor exclusively to compute the reconstruction training loss. It has also been used to re-rank translation hypotheses in prior work, but Tu et al. (2017) showed in ablation studies that the gains from re-ranking are small compared to those from training.

4.4 Results

Table 2 shows that our reconstruction approach achieves small but consistent BLEU improvements over the baseline on all eight tasks.[7]

We evaluate the impact of the Gumbel Softmax hyperparameters on the development set. We select $\tau = 2$ and $\beta = 0/0.5$ based on training stability and BLEU. Greedy search (i.e. $\beta = 0$) performs similarly as sampling with increased Gumbel noise (i.e. more random translation selection when $\beta = 0.5$): increased randomness in sampling does not have a strong impact on BLEU, even though random sampling may approximate the data distribution better (Ott et al., 2018). We hypothesize that more random translation selection introduces lower quality samples and therefore noisier training signals. This is consistent with the observation that random sampling is less effective for back-translation in low-resource settings (Edunov et al., 2018).

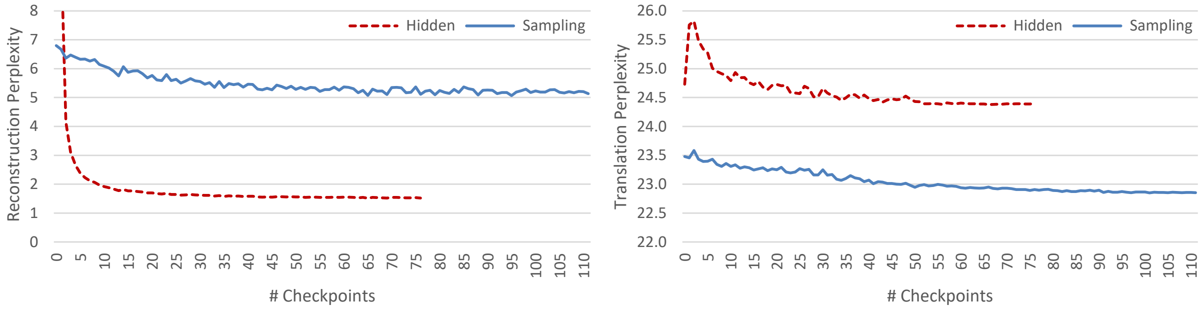
Sampling-based reconstruction is effective even if there is moderate domain mismatch between the training and the test data, such as in the case that the word type out-of-vocabulary (OOV) rate of TR→EN is larger than 20%. Larger improvements can be achieved when the test data is closer to training examples. For example, the OOV rate of SW→EN is much smaller than the OOV rate of TR→EN and the former obtains higher ΔBLEU.

Our approach yields more consistent results than reconstructing from hidden states. The latter fails to improve BLEU in more difficult cases, such as TR↔EN with high OOV rates. We observe extremely low training perplexity for HID-

---

[6] We observed around 0.2 BLEU gains for TR↔EN tasks.

[7] The improvements are significant with $p < 0.01$.



(a) training set (b) development set

Figure 1: Training curves of perplexity on the training and the development sets for TR↔EN. Reconstructing from hidden states (HIDDEN) and reconstructing from sampled translations ($\beta = 0$) are compared. HIDDEN achieves extremely low training perplexity and suffers from unstable training during the early stage.

DEN compared with our proposed approach (Figure 1a). This suggests that HIDDEN yields representations that memorize the input rather than improve output representations.

Another advantage of our approach is that all parameters were jointly pre-trained, which results in more stable training behavior. By contrast, reconstructing from hidden states requires to initialize the reconstructors independently and suffers from unstable early training behavior (Figure 1).

5 Conclusion

We studied reconstructing the input of NMT from its intermediate translations to better exploit training samples in low-resource settings. We used a bi-directional NMT model and the Straight-Through Gumbel Softmax to build a fully differentiable reconstruction model that does not require any additional parameters. We empirically demonstrated that our approach is effective in low-resource scenarios. In future work, we will investigate the use of differentiable reconstruction from sampled sequences in unsupervised and semi-supervised sequence generation tasks. In particular, we will exploit monolingual corpora in addition to parallel corpora for NMT.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments and suggestions. We also thank the members of the Computational Linguistics and Information Processing (CLIP) lab at the University of Maryland for helpful discussions.

This research is based upon work supported in part by an Amazon Web Services Machine Learning Research Award, and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations*.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3th International Conference on Learning Representations*.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Ondrej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation*, pages 272–303. Association for Computational Linguistics.

Richard W. Brislin. 1970. Back-translation for cross-cultural research. *Journal of Cross-Cultural Psychology*, 1(3):185–216.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1965–1974. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5094–5101. AAAI Press.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O. K. Li. 2018a. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 344–354. Association for Computational Linguistics.

Jiatao Gu, Daniel Jiwoong Im, and Victor O. K. Li. 2018b. Neural machine translation with gumbel-greedy decoding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5125–5132. AAAI Press.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems 30*, pages 2146–2156. Curran Associates, Inc.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29*, pages 820–828. Curran Associates, Inc.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. GANS for sequences of discrete elements with the gumbel-softmax distribution. In *Proceedings of the NIPS 2016 Workshop on Adversarial Training*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the 6th International Conference on Learning Representations*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the 5th International Conference on Learning Representations*.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems 27*, pages 3086–3094. Curran Associates, Inc.

Toan Q. Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 334–343. Association for Computational Linguistics.

Xing Niu, Michael Denkowski, and Marine Carpuat. 2018. Bi-directional neural machine translation with synthetic parallel data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 84–91. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3953–3962. PMLR.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation*, pages 186–191. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Computational*, pages 157–163. Association for Computational Linguistics.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1374–1383. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3097–3103. AAAI Press.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM.

Bingning Wang, Kang Liu, and Jun Zhao. 2017. Conditional generative adversarial networks for commonsense machine comprehension. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4123–4129.

Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018a. Translating pro-drop languages with reconstruction models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4937–4945. AAAI Press.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2018b. Learning to jointly translate and predict dropped pronouns with a shared reconstruction mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2997–3002. Association for Computational Linguistics.

Yijun Wang, Yingce Xia, Li Zhao, Jiang Bian, Tao Qin, Guiquan Liu, and Tie-Yan Liu. 2018c. Dual transfer learning for neural machine translation with marginal distribution regularization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5553–5560. AAAI Press.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 555–562. AAAI Press.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575. Association for Computational Linguistics.

Code-Switching for Enhancing NMT with Pre-Specified Translation

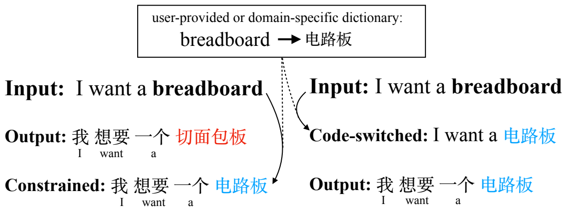
Kai Song[1,2], **Yue Zhang**[3], **Heng Yu**[2], **Weihua Luo**[2], **Kun Wang**[1], **Min Zhang**[1]
[1] Soochow University, Suzhou, China
[2] Alibaba DAMO Academy, Hangzhou, China
[3] School of Engineering, Westlake University, Hangzhou, China
`songkai.sk@alibaba-inc.com`, `zhangyue@wias.org.cn`
`{yuheng.yh,weihua.luowh}@alibaba-inc.com`
`kwang1994@stu.suda.edu.cn`, `minzhang@suda.edu.cn`

Abstract

Leveraging user-provided translation to constrain NMT has practical significance. Existing methods can be classified into two main categories, namely the use of placeholder tags for lexicon words and the use of hard constraints during decoding. Both methods can hurt translation fidelity for various reasons. We investigate a data augmentation method, making code-switched training data by replacing source phrases with their target translations. Our method does not change the NMT model or decoding algorithm, allowing the model to learn lexicon translations by copying source-side target words. Extensive experiments show that our method achieves consistent improvements over existing approaches, improving translation of constrained words without hurting unconstrained words.

1 Introduction

One important research question in domain-specific machine translation (Luong and Manning, 2015) is how to impose translation constraints (Crego et al., 2016; Hokamp and Liu, 2017; Post and Vilar, 2018). As shown in Figure 1 (a), the word “breadboard” can be translated into “切面包板 (a wooden board that is used to cut bread on)” in the food domain, but “电路板 (a construction base for prototyping of electronics)” in the electronic domain. To enhance translation quality, a lexicon can be leveraged for domain-specific or user-provided words (Arthur et al., 2016; Hasler et al., 2018). We investigate the method of leveraging pre-specified translation for NMT using such a lexicon.



(a) Constrained NMT (b) Constrained NMT: Our Method

Figure 1: Constrained NMT

For leveraging pre-specified translation, one existing approach uses *placeholder tags* to substitute named entities (Crego et al., 2016; Li et al., 2016; Wang et al., 2017b) or rare words (Luong et al., 2014) on both the source and target sides during training, so that a model can translate such words by learning to translate placeholder tags. For example, the i-th named entity in the source sentence is replaced with “tag_i”, as well as its corresponding translation in the target side. Placeholder tags in the output are replaced with pre-specified translation as a post-processing step. One disadvantage of this approach, however, is that the meaning of the original words in the pre-specified translation is not fully retained, which can be harmful to both adequacy and fluency of the output.

Another approach (Hokamp and Liu, 2017; Post and Vilar, 2018) imposes pre-specified translation via *lexical constraints*, making sure such constraints are satisfied by modifying NMT decoding. This method ensures that pre-specified translations appear in the output. A problem of this method is that it does not explicitly explore the correlation between pre-specified translations and their corresponding source words during decoding, and thus can hurt translation fidelity (Hasler et al., 2018). There is not a mechanism that allows the model to *learn* constraint translations during training, which the placeholder method allows.

We investigate a novel method based on *data augmentation*, which combines the advantages of both methods above. The idea is to construct synthetic parallel sentences from the original parallel training data. The synthetic sentence pairs resemble code-switched source sentences and their translations, where certain source words are replaced with their corresponding target translations. The motivation is to make the model learn to "translate" embedded pre-specified translations by copying them from the modified source. During decoding, the source is similarly modified as a preprocessing step. As shown in Figure 1 (b), translation is executed over the code-switched source, without further constraints or post-processing.

In contrast to the placeholder method, our method keeps lexical semantic information (i.e. target words v.s. placeholder tags) in the source, which can lead to more adequate translations. Compared with the lexical constraint method, pre-specified translation is *learned* because such information is available both in training and decoding. As a data augmentation method, it can be used on any NMT architecture. In addition, our method enables the model to translate code-switched source sentences, and preserve its strength in translating un-replaced sentences.

To further strengthen copying, we propose two model-level adjustments: First, we share target-side embeddings with source-side target words, so that target vocabulary words have a unique embedding in the NMT system. Second, we integrate pointer network (Vinyals et al., 2015; Gulcehre et al., 2016; Gu et al., 2016; See et al., 2017) into the decoder. The copy mechanism was firstly proposed to copy source words. In our method, it is further used to copy source-side *target* words.

Results on large scale English-to-Russian (En-Ru) and Chinese-to-English (Ch-En) tasks show that our method outperforms both placeholder and lexical constraint methods over a state-of-the-art Transformer (Vaswani et al., 2017) model on various test sets across different domains. We also show that shared embedding and pointer network can lead to more successful applications of the copying mechanism. We release four high-quality En-Ru e-commerce test sets translated by Russian language experts, totalling 7169 sentences with an average length of 21[1].

2 Related Work

Using placeholders. Luong et al. (2014) use *annotated unk tags* to present the *unk* symbols in training corpora, where the correspondence between source and target *unk* symbols are obtained from word alignment (Brown et al., 1993). Output *unk tags* are replaced through a post-processing stage by looking up a pre-specified dictionary or copying the corresponding source word. Crego et al. (2016) extended *unk tags* symbol to specific symbols that can present name entities. Wang et al. (2017b) and Li et al. (2016) use a similar method. This method is limited when constrain NMT with pre-specified translations consisting of more general words, due to the loss of word meaning when representing them with placeholder tags. In contrast to their work, word meaning is fully kept in modified source in our work.

Lexical constraints. Hokamp and Liu (2017) propose an altered beam search algorithm, namely grid beam search, which takes target-side pre-specified translations as lexical constraints during beam search. A potential problem of this method is that translation fidelity is not specifically considered, since there is no indication of a matching source of each pre-specific translation. In addition, decoding speed is significantly reduced (Post and Vilar, 2018). Hasler et al. (2018) use alignment to gain target-side constraints' corresponding source words, simultaneously use finite-state machines and multi-stack (Anderson et al., 2016) decoding to guide beam search. Post and Vilar (2018) give a fast version of Hokamp and Liu (2017), which limits the decoding complexity linearly by altering the beam search algorithm through dynamic beam allocation.

In contrast to their methods, our method does not make changes to the decoder, and therefore decoding speed remains unchanged. Translation fidelity of pre-specified source words is achieved through a combination of training and decoding procedure, where replaced source-side words still contain their target-side meaning. As a soft method of inserting pre-specified translation, our method does not guarantee that all lexical constraints are satisfied during decoding, but has better overall translation quality compared to their method.

Using probabilistic lexicons. Aiming at making use of one-to-many phrasal translations, the following work is remotely related to our work. Tang et al. (2016) use a phrase memory to provide extra information for their NMT encoder, dynamically switching between word generation and

---

[1] To best of our knowledge, this is the first public e-commerce test set.

phrase generation during decoding. Wang et al. (2017a) use SMT to recommend prediction for NMT, which contains not only translation operations of a SMT phrase table, but also alignment information and coverage information. Arthur et al. (2016) incorporate discrete lexicons by converting lexicon probabilities into predictive probabilities and linearly interpolating them with NMT probability distributions.

Our method is similar in the sense that external translations of source phrases are leveraged. However, their tasks are different. In particular, these methods regard one-to-many translation lexicons as a *suggestion*. In contrast, our task aims to *constrain* NMT translation through **one-to-one** pre-specified translations. Lexical translations can be used to generate code-switched source sentences during training, but we do not modify NMT models by integrating translation lexicons. In addition, our data augmentation method is more flexible, because it is model-free.

Alkhouli et al. (2018) simulate a dictionary-guided translation task to evaluate NMT's alignment extraction. A one-to-one word translation dictionary is used to guide NMT decoding. In their method, a dictionary entry is limited to only one word on both the source and target sides. In addition, a pre-specified translation can come into effect only if the corresponding source-side word is successfully aligned during decoding.

On translating named entities, Currey et al. (2017) augment the training data by copying target-side sentences to the source-side, resulting in augmented training corpora where the source and the target sides contain identical sentences. The augmented data is shown to improve translation performance, especially for proper nouns and other words that are identical in the source and target languages.

3 Data augmentation

Our method is based on *data augmentation*. During training, augmented data are generated by replacing source words or phrases directly with their corresponding target translations. The motivation is to sample as many code-switched translation pairs as possible. During decoding, given pre-specified translations, the source sentence is modified by replacing phrases with their pre-specified translations, so that the trained model can directly copy embedded target translations in the output.

3.1 Training

Given a bilingual training corpus, we sample augmented sentence pairs by leveraging a SMT phrase table, which can be trained over the same bilingual corpus or a different large corpus. We extract source-target phrase pairs[2] from the phrase table, replacing source-side phrases of source sentences using the following sampling steps:

1. Indexing between source-target phrase pairs and training sentences: (a) For each source-target phrase pair, we record all the matching bilingual sentences that contain both the source and target. Word alignment can be used to ensure the phrase pairs that are mutual translation. (b) We also sample bilingual sentences that match two source-target phrase pairs. In particular, given a combination of two phrase pairs, we index bilingual sentences that match both simultaneously.

2. Sampling: (a) For each source-target phrase pair, we keep at most k_1 randomly selected matching sentences. The source-side phrase is replaced with its target-side translation. (b) For each combination of two source-target phrase pairs, we randomly sample at most k_2 matching sentences. Both source-side matching phrases are replaced with their target translations.[3]

The sampled training data is added to the original training data to form a final set of training sentences.

3.2 Decoding

We impose target-side pre-specified translations to the source by replacing source phrases with their translations. Lexicons are defined in the form of one-to-one source-target phrase pairs. Different from training, the number of replaced phrases in a source sentence is not necessarily restricted to one or two, which will be discussed in Section 5.5. In practice, pre-specified translations can be provided by customers or through user feedback, which contains one identified translation for specified source segment.

4 Model

Transformer (Vaswani et al., 2017) uses self-attention network for both encoding and decod-

---

[2] Source-side phrase is at most trigram.

[3] We set $k_1 = 100, k_2 = 30$ empirically.

ing. The encoder is composed of n stacked neural layers. For time step i in layer j, the hidden state $h_{i,j}$ is calculated by employing self-attention over the hidden states in layer $j-1$, which are $\{h_{1,j-1}, h_{2,j-1}, ..., h_{m,j-1}\}$, where m is the number of source-side words.

In particular, $h_{i,j}$ is calculated as follows: First, a self-attention sub-layer is employed to encode the context. Then *attention weights* are computed as scaled dot product between the current query $h_{i,j-1}$ and all keys $\{h_{1,j-1}, h_{2,j-1}, ..., h_{m,j-1}\}$, normalized with a softmax function. After that, the context vector is represented as *weighted sum* of the values projected from hidden states in the previous layer, which are $\{h_{1,j-1}, h_{2,j-1}, ..., h_{m,j-1}\}$. The hidden state in the previous layer and the context vector are then connected by residual connection, followed by a layer normalization function (Ba et al., 2016), to produce a candidate hidden state $h'_{i,j}$. Finally, another sub-layer including a feed-forward network (FFN) layer, followed by another residual connection and layer normalization, are used to obtain the hidden state $h_{i,j}$.

In consideration of translation quality, multi-head attention is used instead of single-head attention as mentioned above, positional encoding is also used to compensate the missing of position information in this model.

The decoder is also composed of n stacked layers. For time step t in layer j, a self-attention sub-layer of hidden state $s_{t,j}$ is calculated by employing self-attention mechanism over hidden states in previous target layer, which are $\{s_{1,j-1}, s_{2,j-1}, ..., s_{t-1,j-1}\}$, resulting in candidate hidden state $s'_{t,j}$. Then, a second target-to-source sub-layer of hidden state $s_{t,j}$ is inserted above the target self-attention sub-layer. In particular, the queries(Q) are projected from $s'_{t,j}$, and the keys(K) and values(V) are projected from the source hidden states in the last layer of encoder, which are $\{h_{1,n}, h_{2,n}, ..., h_{m,n}\}$. The output state is another candidate hidden state $s''_{t,j}$. Finally, a last feed-forward sub-layer of hidden state $s_{t,j}$ is calculated by employing self-attention over $s''_{t,j}$.

A softmax layer based on decoder's last layer $s_{t,n}$ is used to gain a probability distribution $P_{predict}$ over target-side vocabulary.

$$p(y_t|y_1, ..., y_{t-1}, \mathbf{x}) = \text{softmax}(s_{t,n} * \mathbf{W}), \quad (1)$$

where $\mathbf{W}$ is the weight matrix which is learned, $\mathbf{x}$ represent the source sentence, $\{y_1, y_2, ..., y_t\}$ represent target words.

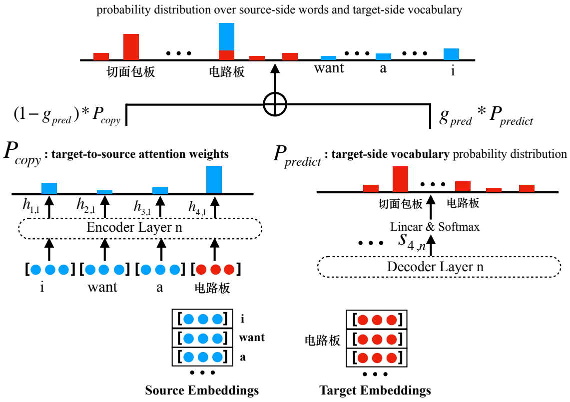


Figure 2: Shared embeddings and pointer network

4.1 Shared Target Embeddings

Shared target embeddings enforces the correspondence between source-side and target-side expressions on the embedding level. As shown in Figure 2, during encoding, source-side target word embeddings are identical to their embeddings in the target-side vocabulary embedding matrix. This makes it easier for the model to copy source-side target words to the output.

4.2 Pointer Network

To strengthen copying through locating source-side target words, we integrate *pointer network* (Gulcehre et al., 2016) into the decoder, as shown in Figure 2. At each decoding time step t, the target-to-source attention weights $\alpha_{t,1}, ..., \alpha_{t,m}$ are utilized as a probability distribution P_{copy}, which models the probability of *copying* a word from the i-th source-side position. The i-th source-side position may represent a source-side word or a source-side target word. P_{copy} is added to $P_{predict}$, the probability distribution over target-side vocabulary, to gain a new distribution over both the source and the target side vocabulary[4]:

$$P = (1 - g_{pred}) * P_{copy} + g_{pred} * P_{predict}, \quad (2)$$

where g_{pred} is used to control the contribution of two probability distributions. For time step t, g_{pred} is calculated from the context vector c_t and the current hidden state of the decoder's last layer $s_{t,n}$:

---

[4] For the words which belong to the source-side vocabulary but are not appeared in the source-side sentence, the probabilities are set to 0.

$$g_{pred} = \sigma(c_t * W_p + s_{t,n} * W_q + b_r), \quad (3)$$

where W_p, W_q, and b_r are parameters trained and σ is the sigmoid function. In addition, the context vector c_t is calculated as $c_t = \sum_{i=1}^{m} \alpha_{t,i} * h_{i,n}$, where $\alpha_{t,i}$ is attention weight mentioned earlier. $\{h_{1,n}, h_{2,n}, ..., h_{m,n}\}$ are the source-side hidden states of the encoder's last layer.

5 Experiments

We compare our method with strong baselines on large-scale En-Ru and Ch-En tasks on various test sets across different domains, using a strongly optimized Transformer (Vaswani et al., 2017). BLEU (Papineni et al., 2002) is used for evaluation.

5.1 Data

Our training corpora are taken from the WMT2018 news translation task.

En-Ru. We use 13.88M sentences as baseline training data, containing both a real bilingual corpus and a synthetic back-translation corpus (Sennrich et al., 2015a). The synthetic corpus is translated from "NewsCommonCrawl", which can be obtained from the WMT task. The news domain contains four different test sets published by WMT2018 over the recent years, namely "news2015", "news2016", "news2017", and "news2018", respectively, each having one reference. The e-commerce domain contains four files totalling 7169 sentences, namely "subject17", "desc17", "subject18", and "desc18", respectively, each having one reference. The sentences are extracted from e-commerce websites, in which "subject"s are the goods names shown on a listing page. "desc"s refer to information in a commodity's description page. "subject17" and "desc17" are released[5]. Our development set is "news2015".

Ch-En. We use 7.42M sentences as our baseline training data, containing both real bilingual corpus and synthetic back-translation corpus (Sennrich et al., 2015a). We use seven public development and test data sets, four in the news domain, namely "NIST02", "NIST03", "NIST04", "NIST05", respectively, each with four references, and three in the spoken language domain, namely "CSTAR03", "IWSLT2004", "IWLST2005", respectively, each with 16 references. "NIST03" is used for development.

5.2 Experimental Settings

We use six self-attention layers for both the encoder and the decoder. The embedding size and the hidden size are set to 512. Eight heads are used for self-attention. A feed-forward layer with 2048 cells and Swish (Ramachandran et al., 2018) is used as the activation function. Adam (Kingma and Ba, 2014) is used for training; warmup step is 16000; the learning rate is 0.0003. We use label smoothing (Junczys-Dowmunt et al., 2016) with a confidence score of 0.9, and all the drop-out (Gal and Ghahramani, 2016) probabilities are set to 0.1.

We extract a SMT phrase table on the bilingual training corpus by using moses (Koehn et al., 2007) with default setting, which is used for matching sentence pairs to generate augmented training data. We apply count-based pruning (Zens et al., 2012) to the phrase table, the threshold is set to 10.

During **decoding**, similar to Hasler et al. (2018), Alkhouli et al. (2018) and Post and Vilar (2018), we make use of references to obtain gold constraints. Following previous work, pre-specified translations for each source sentence are sampled from references and used by all systems for fair comparison.

In all the baseline systems, the vocabulary size is set to 50K on both sides. For "Data augmentation", to allow the source-side dictionary to cover target-side words, the target- and source-side vocabularies are merged for a new source vocabulary. For "Shared embeddings", the source vocabulary remains the same as the baselines, where the source-side target words use embeddings from target-side vocabulary.

5.3 System Configurations

We use an in-house reimplementation of Transformer, similar to Google's Tensor2Tensor. For the baselines, we reimplement Crego et al. (2016), as well as Post and Vilar (2018). BPE (Sennrich et al., 2015b) is used for all experiments, the operation is set to 50K. Our test sets cover news and e-commerce domains on En-Ru, and news and spoken language domains on Ch-En.

Baseline 1: Using Placeholder. We combine Luong et al. (2014) and Crego et al. (2016). For

---

[5] `https://github.com/batman2013/e-commerce_test_sets`

	news15	news16	news17	news18	△	subject17	desc17	subject18	desc18	△
Marian	33.27	31.91	36.18	32.11	-0.15	8.03	23.21	11.02	27.94	-0.46
Transformer	33.29	31.95	36.57	32.27	-	8.56	23.53	11.95	27.90	-
+ Placeholder	33.14	32.07	36.24	32.03	-0.15	9.81	24.04	13.84	29.34	+1.27
+ Lexi. Cons.	33.50	32.62	36.65	32.88	+0.39	9.24	23.67	13.1	29.83	+0.98
Data Aug.	34.71	33.69	38.43	33.51	+1.57	10.63	25.56	14.26	30.92	+2.36
+ Share	35.28	34.37	39.02	34.44	+2.26	10.82	25.84	15.20	30.97	+2.72
+ Share&Point	**36.44**	**35.31**	**40.23**	**35.43**	**+3.33**	**11.58**	**26.53**	**16.08**	**32.17**	**+3.61**

Table 1: Results on En-Ru, one or two source phrases of each sentence have pre-specified translation. "Transformer" is our in-house vanilla Transformer baseline. "Marian" is the implementation of Transformer by Junczys-Dowmunt et al. (2018), which is used as a reference of our Transformer implementation.

	CSTAR03	IWSLT04	IWSLT05	△	NIST02	NIST03	NIST04	NIST05	△
Transformer	53.03	56.52	64.72	-	40.52	37.85	40.12	39.26	-
+ Placeholder	52.51	56.15	64.44	-0.39	40.01	37.16	39.96	38.87	-0.44
+ Lexi. Cons.	53.30	56.95	65.63	+0.54	40.36	38.02	40.44	39.72	+0.20
Data Aug.	53.82	57.28	65.54	+0.79	40.85	38.41	40.81	40.29	+0.65
+Share	**53.90**	**57.67**	65.59	**+0.96**	41.06	38.57	41.22	40.38	+0.87
+Share&Point	53.79	57.29	**65.65**	+0.82	**41.11**	**38.7**	**41.3**	**40.4**	**+0.94**

Table 2: Results on Ch-En, one or two source phrases of each sentence have pre-specified translation.

generating placeholder tags during training, following Crego et al. (2016), we use a named entity translation dictionary which is extracted from Wikidata[6]. The dictionary is released together with e-commerce test sets, which is mentioned before. For Ch-En, the dictionary contains 285K person names, 746K location names and 1.6K organization names. For En-Ru, the dictionary contains 471K person names, 254K location names and 1.5K organization names. Additionally, we manually corrected a dictionary which contains 142K brand names and product names translation for En-Ru. By further leveraging word alignment in the same way as Luong et al. (2014), the placeholder tags are annotated with indices. We use FastAlign (Dyer et al., 2013) to generate word alignment. The amount of sentences containing placeholder tags is controlled to a ratio of 5% of the corpus. During decoding, pre-specified translations described in Section 5.2 are used.

Baseline 2: Lexical Constraints. We re-implement Post and Vilar (2018), integrating their algorithm into our Transformer. Target-side words or phrases of pre-specified translations mentioned in Section 5.2 are used as lexical constraints.

Our System. During training, we use the method described in Section 3.1 to obtain the augmented training data. The SMT phrase table mentioned in Section 5.2 is used for "Indexing" and "Sampling". During decoding, pre-specified translations mentioned in Section 5.2 are used. The augmented data contain sampled sentences with one or two replacements on the source side. By applying the two sampling steps described in Section 3.1, about 10M and 6M augmented Ch-En and En-Ru sentences are generated, respectively. The final training corpora consists of both the augmented training data and the original training data.

5.4 Results

Comparison with Baselines. Our Transformer implementation can give comparable performance with state-of-the-art NMT (Junczys-Dowmunt et al., 2018), see "Transformer" and "Marian" in Table 1, which also shows a comparison of different methods on En-Ru. The *lexical constraint* method gives improvements on both the news and the e-commerce domains, compared with the Transformer baseline. The *placeholder* method also gives an improvement on the e-commerce

---

[6] `https://www.wikidata.org`

Pre-Specified Translation: **计划生育 —> planned parenthood**

Source: 上 个 月 ， 民主 党 否决 了 共和 党 提出 的 一 项 提议 ， 该 提议 要求 拨款 11 亿 美元 抗击 寨 卡 但 包含 禁止 资助 计划 生育 协会 提供 与 病毒 相关 的 避孕 条件 的 条款 ， 而 寨 卡 病毒 可以 通过 性 途径 传播 。

Transformer: last month , the democratic party of the republic rejected a proposal by the republican party to grant $ 1.1 billion to fight against the zika but contained provisions prohibiting the **family planning** association from providing virus - related contraceptive conditions and the zika virus could be transmitted sexually .

Baseline1(Placeholder): last month , the party rejected a proposal from the republican party that would require $ 1.1 billion to fight zika , but would include provisions that prohibit **program** **planned parenthood** from providing virus - related contraceptive conditions , and zika could be transmitted sexually .

Baseline2(Lexi. Cons.): last month , the democratic party of the republic rejected a proposal by the republican party to grant $ 1.1 billion to fight against the zika but contained provisions that prohibit the family planning association from providing contraceptive conditions related to the virus and the zika virus could be transmitted through a sexual route **, the u . s . national security council has been banned from funding for the** **planned parenthood** .

Ours: last month , the democratic party rejected a republican proposal calling for $ 1.1 billion to fight zika , which contains provisions that prohibit the association of **planned parenthood** from providing virus - related contraceptive conditions , and the zika virus could be transmitted sexually .

Reference: last month , democrats blocked consideration of a republican measure that would have allocated $ 1.1 billion to fight zika but included provisions that would have banned funding for **planned parenthood** to provide contraception related to the virus , which can be sexually transmitted .

Figure 3: Sample outputs.

domain. The average improvement is calculated over all the test set results in each domain. In the news domain, the average improvement of *our method* is 3.48 BLEU higher compared with *placeholder*, and 2.94 over *lexical constraints*. In the e-commerce domain, the average improvement of *our method* is 1.34 BLEU compared with *placeholder*, and 2.63 with *lexical constraints*. Both shared embedding and pointer network are effective. Table 2 shows the same comparison on Ch-En. In the spoken language domain, the average improvement is 1.35 BLEU compared with *placeholder*, and 0.42 with *lexical constraints*. In the news domain, the average improvement is 1.38 BLEU compared with *placeholder*, and 0.74 with *lexical constraints*.

We find that the *placeholder* method can only bring improvements on the En-Ru e-commerce test sets, since the pre-specified translations of the four e-commerce test sets are mostly *entities*, such as brand names or product names. Using placeholder tags to represent these entities leads to relatively little loss of word meaning. But on many of the other test sets, pre-specified translations are mostly vocabulary words. The placeholder tags fail to keep their word meaning during translation, leading to lower results.

The speed contrast between unconstrained NMT, lexical constraint and our method is shown in Table 3. The decoding speed of our method is equal to unconstrained NMT, and faster than the lexical constraint method, which confirms our intuition introduced earlier.

Beam Size	5	10	20	30
Unconstrained & Ours	416	312	199	146
Lexical Constraint	102	108	74	50

Table 3: Decoding speed (words/sec), Ch-En dev set.

Sample Outputs. Figure 3 gives a comparison of different system's translations. Given a Chinese source sentence, the baseline system fails to translate "计划生育" adequately, as "family planning" is not a correct translation of "计划生育". In the pre-specified methods, the correct translation ("计划生育" to "planned parenthood") is achieved through different ways.

For the *placeholder* method, the source phrase "计划生育" is replaced with the placeholder tag "tag_1" during pre-processing. After translation, output "tag_1" is replaced with "planned parenthood" as a post-processing step. However, the underlined word "program" is generated before "planned parenthood", which has no relationship with any source-side word. The source-side word "协会", which means "association", is omitted in translation. Through deeper analysis, the specific phrase "program tag_1" occurs frequently in the training data. During decoding, using the hard tag leads to the loss of the source phrase's original meaning. As a result, the word "program" is incorrectly generated along with "tag_1".

The *lexical constraints* method regards the tar-

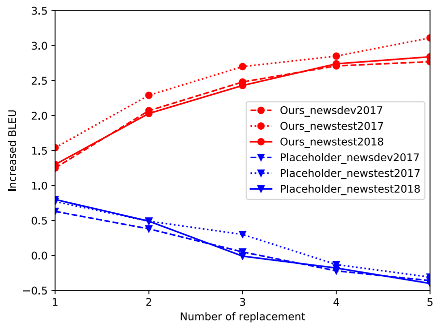


Figure 4: Increased BLEU on Ch-En test sets.

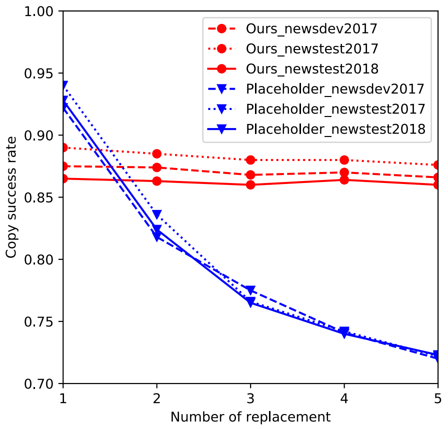


Figure 5: Copy success rate on Ch-En test sets.

get side of the pre-specified translation as a lexical constraint. Here the altered beam search algorithm fails to predict the constraint "planned parenthood" during previous decoding steps. Although the constraint finally comes into effect, over translation occurs, which is highlighted by the underlined words. This is because the method enforces hard constraints, preventing decoding to stop until all constraints are met.

Our method makes use of pre-specified translation by replacing the source-side phrase "计划生育" with the target-side translation "planned parenthood", copying the desired phrase to the output along with the decoding procedure. The translation "association of planned parenthood from providing" is the exact translation of the source-side phrase "计划(planned) 生育(parenthood) 协会(association) 提供(providing)", and agrees with the reference, "planned parenthood to provide".

5.5 Analysis

Effect of Using More Pre-specified Translations. Even though the augmented training data have only one or two replacements on the source side, the model can translate a source sentence with up to five replacements. Figure 4 shows that compared with unconstrained Transformer, the translation quality of our method keeps increasing when the number of replacements increases, since more pre-specified translations are used.

We additionally measure the effect on the Ch-En WMT test sets, namely "newsdev2017", "newstest2017", "newstest2018", respectively, each having only one reference instead of four. The baseline BLEU scores on these three test sets are 18.49, 20.01 and 19.05, respectively. Our method gives BLEU scores of 20.56, 22.3, 21.08, respectively, when using one or two pre-specified translations for each sentence. The increased BLEU when utilizing different number of pre-specified translations is shown in Figure 4. We found that the improvements on WMT test sets are more significant than on NIST, since pre-specified translations are sampled from one reference only, enforcing the output to match this reference. The placeholder method does not give consistent improvements on news test sets, due to the same reason as mentioned earlier.

As shown in Figure 5, the copy success rate of our method does not decrease significantly when the number of replacements grows. Here, a *copy success* refers a pre-specified target translation that can occur in the output. The placeholder method achieves a higher copy success rate than ours when the number of replacements is 1, but the copy success rate decreases when using more pre-specified translations. The copy success rate of the lexical constraint method is always 100%, since it imposes hard constraints rather than soft constraints. However, as discussed earlier, overall translation quality can be harmed as a cost of satisfying decoding constraints by their method.

In the presented experiment results, the highest copy success rate of our method is 90.54%, which means a number of source-side target words or phrases are not successfully copied to the translation output. This may be caused by the lack of training samples for certain target-side words or phrases. In En-Ru, we additionally train a model with augmented data that is obtained by matching

	NIST02	NIST03	NIST04	NIST05
Data Aug.	83.89%	85.71%	86.71%	87.45%
+Share&Point	87.72%	88.31%	89.18%	90.54%

Table 4: Copy success rate on Ch-En test sets.

	news15	news16	news17	news18
Baseline	33.29	31.95	36.57	32.27
Ours	33.53	32.29	36.54	32.47

Table 5: BLEU scores of non code-switched (original) input on En-Ru test sets.

an SMT phrase table without any pruning strategy. The copy success rate can reach 98%, even without using "shared embedding" and "pointer network" methods.

Effect of Shared Embeddings and Pointer Network. The gains of *shared embeddings* and *pointer network* are reflected in both the copy success rate and translation quality. As shown in Table 4, when using one pre-specified translation for each source sentence, the copy success rate improves on various test sets by integrating shared embeddings and pointer network, demonstrating that more pre-specified translations come into effect. Table 1 and Table 2 earlier show the improvement of translation quality.

Translating non Code-Switched Sentences. Our method preserves its strength on translating non code-switched sentences. As shown in Table 5, the model trained on the augmented corpus has comparable strength on translating unreplaced sentences as the model trained on the original corpus. In addition, on some test sets, our method is slightly better than the baseline when translating non code-switched source sentences. This can be explained from two aspects: First, the augmented data make the model more robust to perturbed inputs; Second, the pointer network makes the model better by copying certain source-side words (Gulcehre et al., 2016), such as non-transliterated named entities.

6 Conclusion

We investigated a data augmentation method for constraining NMT with pre-specified translations, utilizing code-switched source sentences and their translations as augmented training data. Our method allows the model to learn to translate source-side target phrases by "copying" them to the output, achieving consistent improvements over previous lexical constraint methods on large NMT test sets. To the best of our knowledge, we are the first to leverage code switching for NMT with pre-specified translations.

7 Future Work

In the future, we will study how the copy success rate and the BLEU scores interact when different sampling strategies are taken to obtain augmented training corpus and when the amount of augmented data grows. Another direction is to validate the performance when applying this approach to language pairs that contain a number of identical letters in their alphabets, such as English to French and English to Italian.

Acknowledgments

We thank the anonymous reviewers for their detailed and constructed comments. Yue Zhang is the corresponding author. The research work is supported by the National Natural Science Foundation of China (61525205). Thanks for Shaohui Kuang, Qian Cao, Zhongqiang Huang and Fei Huang for their useful discussion.

References

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. *arXiv preprint arXiv:1809.03985*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Guided open vocabulary image captioning with constrained beam search. *CoRR*, abs/1612.00576.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. *arXiv preprint arXiv:1606.02006*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Eva Hasler, Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. *arXiv preprint arXiv:1805.03750*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *arXiv preprint arXiv:1704.07138*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The amu-uedin submission to the wmt16 news translation task: Attention-based nmt models as feature functions in phrase-based smt. *arXiv preprint arXiv:1605.04809*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Neural name translation improves neural machine translation. *arXiv preprint arXiv:1607.01856*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. *arXiv preprint arXiv:1804.06609*.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2018. Searching for activation functions.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *Computer Science*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip LH Yu. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017a. Neural machine translation advised by statistical machine translation. In *AAAI*, pages 3330–3336.

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017b. Sogou neural machine translation systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 410–415.

Richard Zens, Daisy Stanton, and Peng Xu. 2012. A systematic comparison of phrase table pruning techniques. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 972–983. Association for Computational Linguistics.

Aligning Vector-spaces with Noisy Supervised Lexicons

Noa Yehezkel Lubin[†1] **Jacob Goldberger**[‡2] **Yoav Goldberg**[†*3]

[†]Computer Science Department, Bar Ilan University, Ramat Gan, Israel
[‡]Electrical Engineering Department, Bar Ilan University, Ramat Gan, Israel
[*]Allen Institute for Artificial Intelligence

[1]`noa.kel@gmail.com`
[2]`jacob.goldberger@biu.ac.il`
[3]`yoav.goldberg@gmail.com`

Abstract

The problem of learning to translate between two vector spaces given a set of aligned points arises in several application areas of NLP. Current solutions assume that the lexicon which defines the alignment pairs is noise-free. We consider the case where the set of aligned points is allowed to contain an amount of noise, in the form of incorrect lexicon pairs and show that this arises in practice by analyzing the edited dictionaries after the cleaning process. We demonstrate that such noise substantially degrades the accuracy of the learned translation when using current methods. We propose a model that accounts for noisy pairs. This is achieved by introducing a generative model with a compatible iterative EM algorithm. The algorithm jointly learns the noise level in the lexicon, finds the set of noisy pairs, and learns the mapping between the spaces. We demonstrate the effectiveness of our proposed algorithm on two alignment problems: bilingual word embedding translation, and mapping between diachronic embedding spaces for recovering the semantic shifts of words across time periods.

1 Introduction

We consider the problem of mapping between points in different vector spaces. This problem has prominent applications in natural language processing (NLP). Some examples are creating bilingual word lexicons (Mikolov et al., 2013), machine translation (Artetxe et al., 2016, 2017a,b, 2018a,b; Conneau et al., 2017), hypernym generation (Yamane et al., 2016), diachronic embeddings alignment (Hamilton et al., 2016) and domain adaptation (Barnes et al., 2018). In all these examples one is given word embeddings in two different vector spaces, and needs to learn a mapping from one to the other.

The problem is traditionally posed as a supervised learning problem, in which we are given two sets of vectors (e.g.: word-vectors in Italian and in English) and a *lexicon* mapping the points between the two sets (known word-translation pairs). Our goal is to learn a mapping that will correctly map the vectors in one space (e.g.: English word embeddings) to their known corresponding vectors in the other (e.g.: Italian word embeddings). The mapping will then be used to translate vectors for which the correspondence is unknown. This setup was popularized by Mikolov et al. (2013).

The supervised setup assumes a perfect lexicon. Here, we consider what happens in the presence of *training noise*, where some of the lexicon's entries are incorrect in the sense that they don't reflect an optimal correspondence between the word vectors.

2 Background

2.1 The Supervised Translation Problem

We are given two datasets, $X = x_1, ..., x_m$ and $Y = y_1, ..., y_n$, coming from d-dimensional spaces $\mathcal{X}$ and $\mathcal{Y}$. We assume that the spaces are related, in the sense that there is a function $f(x)$ mapping points in space $\mathcal{X}$ to points in space $\mathcal{Y}$. In this work, we focus on linear mappings, i.e. a $d \times d$ matrix Q mapping points via $y_i = Qx_i$. The goal of the learning is to find the translation matrix Q. In the supervised setting, $m = n$ and we assume that $\forall i \ f(x_i) \approx y_i$. We refer to the sets X and Y as the *supervision*. The goal is to learn a matrix $\hat{Q}$ such the Frobenius norm is minimized:

$$\hat{Q} = \arg\min_{Q} \|QX - Y\|_F^2. \qquad (1)$$

2.2 Existing Solution Methods

Gradient-based The objective in (1) is convex, and can be solved via least-squares method or via stochastic gradient optimization iterating over the

Proceedings of NAACL-HLT 2019, pages 460–465
Minneapolis, Minnesota, June 2 - June 7, 2019.

pairs (x_i, y_i), as done by Mikolov et al. (2013) and Dinu and Baroni (2014).

Orthogonal Procrustes (OP) Artetxe et al. (2016) and Smith et al. (2017) argued and proved that a linear mapping between sub-spaces must be orthogonal. This leads to the modified objective:

$$\hat{Q} = \underset{Q, s.t: Q^TQ=I}{\arg\min} \|QX - Y\|_F^2 \tag{2}$$

Objective (2) is known as the *Orthogonal Procrustes Problem*. It can be solved algebraically by using a singular value decomposition (SVD). Schnemann (1966) proved that the solution to 2 is: $\hat{Q} = UV^T$ s.t. $U\Sigma V^T$ is the SVD of YX^T.

The OP method is used in Xing et al. (2015); Artetxe et al. (2016, 2017a,b, 2018a,b); Hamilton et al. (2016); Conneau et al. (2017); Ruder et al. (2018).

2.3 The Unsupervised Translation Problem

The supervised alignment problem can be expended to the semi-supervised (Artetxe et al., 2017b; Lample et al., 2017; Ruder et al., 2018) or unsupervised (Zhang et al., 2017; Conneau et al., 2017; Artetxe et al., 2018b; Xu et al., 2018; Alvarez-Melis and Jaakkola, 2018) case, where a very small lexicon or none at all is given. In iterative methods, the lexicon is expended and used to learn the alignment, later the alignment is used to predict the lexicon for the next iteration and so on. In adversarial methods, a final iterative step is used after the lexicon is built to refine the result. We will focus on the supervised stage in the unsupervised setting, meaning estimating the alignment once a lexicon is induced.

3 The Effect of Noise

The previous methods assume the supervision set X, Y is perfectly correct. However, this is often not the case in practice. We consider the case where a percentage p of the pairs in the supervision set are "noisy": applying the gold transformation to a noisy point x_j will not result in a vector close to y_j. The importance of the quality of word-pairs selection was previously analyzed by Vulić and Korhonen (2016). Here, we equate "bad pairs" to noise, and explore the performance in the presence of noise by conducting a series of synthetic experiments. We take a set of points X, a random transformation Q and a gold set $Y = QX$. We define *error* as $\|Y - \hat{Y}\|_F^2$ where $\hat{Y} = \hat{Q}X$ is the prediction according to the learned transform $\hat{Q}$. Following the claim that linear transformations between word vector spaces are orthogonal, we focus here on orthogonal transformations.

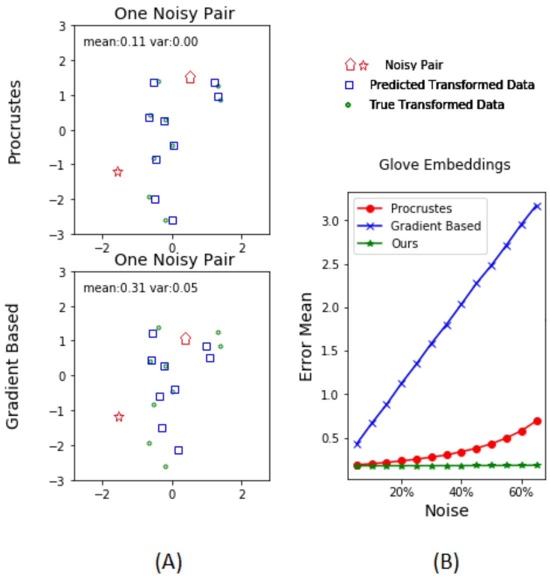


Figure 1: Noise influence. (A): the effect of a noisy pair on 2D alignment. (B) mean error over non-noisy pairs as a function of noise level.

Low Dimensional Synthetic Data We begin by inspecting a case of few 2-dimensional points, which can be easily visualized. We compare a noise-free training to the case of a single noisy point. We construct X by sampling $n = 10$ points of dimension $d = 2$ from a normal distribution. We take nine points and transformed them via an orthogonal random transform Q. We then add a single noisy pair which is generated by sampling two normally distributed random points and treating them as a pair. The error is measured only on the nine aligned pairs.

When no noise is applied, both Gradient-based and Procrustes methods are aligned with 0 error mean and variance. Once the noisy condition is applied this is no longer the case. Figure 1(A) shows the noisy condition. Here, the red point (true) and box (prediction) represent the noisy point. Green dots are the true locations after transformation, and the blue boxes are the predicted ones after transformation. Both methods are affected by the noisy sample: all ten points fall away from their true location. The effect is especially severe for the gradient-based methods.

High Dimensional Embeddings The experiment setup is as before, but instead of a normal distribution we use (6B, 300d) English Glove Embeddings (Pennington et al., 2014) with lexicon of size $n = 5000$. We report the mean error for various noise levels on an unseen aligned test set of size 1500.

In Figure 1(B) we can see that both methods are effected by noise. As expected, as the amount of noise increases the error on the test set increases. We can again see that the effect is worse with gradient-based methods.

4 Noise-aware Model

Having verified that noise in the supervision severely influences the solution of both methods, we turn to proposing a noise-aware model.

The proposed model jointly identifies noisy pairs in the supervision set and learns a translation which ignores the noisy points. Identifying the point helps to clean the underlying lexicon (dictionary) that created the supervision. In addition, by removing those points our model learns a better translation matrix.

Generative Model We are given $x \in \mathbb{R}^d$ and we sample a corresponding $y \in \mathbb{R}^d$ by first sampling a Bernoulli random variable with probability α:

$$z \sim \text{Bernoulli}(\alpha)$$

$$y \sim \begin{cases} N(\mu_y, \sigma_y^2 I) & z = 0\,(\text{‘noise’}) \\ N(Qx, \sigma^2 I) & z = 1\,(\text{‘aligned’}) \end{cases}$$

The density function y is a mixture of two Gaussians:

$$f(y|x) = (1-\alpha)N(\mu_y, {\sigma_y}^2 I) + \alpha N(Qx, \sigma^2 I).$$

The likelihood function is:

$$L(Q, \sigma, \mu_y, \sigma_y) = \sum_t \log f(y_t|x_t)$$

EM Algorithm We apply the EM algorithm (Dempster et al., 1977) to maximize the objective in the presence of latent variables. The algorithm has both soft and hard decision variants. We used the hard decision one which we find more natural, and note that the posterior probability of z_t was close to 0 or 1 also in the soft-decision case.

It is important to properly initialize the EM algorithm to avoid convergence to a local optima. We initialize Q by applying OP on the entire lexicon (not just the clean pairs). We initialize the variance, σ, by calculating $\sigma^2 = \frac{1}{n \cdot d} \sum_{t=1} \|Qx_t - y_t\|^2$. We initialize, μ_y, σ_y by taking the mean and variance of the entire dataset. Finally, we initialize α to 0.5.

The (hard version) EM algorithm is shown in Algorithm box 1. The runtime of each iteration is dominated by the OP algorithm (matrix multiplication and SVD on a $d \times d$ matrix). Each iteration contains an additional matrix multiplication and few simple vector operations. Figure 1(B) shows it obtains perfect results on the simulated noisy data.

Algorithm 1 Noise-aware Alignment

```
Data: List of paired vectors: $(x_1, y_1), ..., (x_n, y_n)$
Result: $Q$, $\sigma$, $\mu_y$, $\sigma_y$
while $|\alpha_{curr} - \alpha_{prev}| > \epsilon$ do
    E step:
        $w_t = p(z_t=1|x_t, y_t) = \frac{\alpha N(Qx_t, \sigma^2 I)}{f(y_t|x_t)}$
        $h_t = 1(w_t > 0.5)$
        $n_1 = \sum_t h_t$
    M step:
        Apply OP on the subset $\{t|h_t = 1\}$ to find Q.
        $\sigma^2 = \frac{1}{d \cdot n_1} \sum_{t|h_t=1} \|Qx_t - y_t\|^2$
        $\mu_y = \frac{1}{(n-n_1)} \sum_{t|h_t=0} y_t$
        $\sigma_y^2 = \frac{1}{d(n-n_1)} \sum_{t|h_t=0} \|\mu_y - y_t\|^2$
        $\alpha_{prev} = \alpha_{curr}$
        $\alpha_{curr} = \frac{n_1}{n}$
end
```

5 Experiments

5.1 Bilingual Word Embedding

Experiment Setup This experiment tests the noise-aware solution on an unsupervised translation problem. The goal is to learn the “translation matrix”, which is a transformation matrix between two languages by building a dictionary. We can treat the unsupervised setup after retrieving a lexicon as an iterative supervised setup where some of the lexicon pairs are noisy. We assumes the unsupervised setting will contain higher amount of noise than the supervised one, especially in the first iterations. We follow the experiment setup in Artetxe et al. (2018b). But instead of using OP for learning the translation matrix, we used our Noise-Aware Alignment (NAA), meaning we jointly learn to align and to ignore the noisy pairs. We used the En-It dataset provided by Dinu and Baroni (2014) and the extensions: En-De, En-Fi and En-Es of Artetxe et al. (2018a, 2017b).

Method	En→It			En→De			En→Fi			En→Es		
	best	avg	iters	best	avg	iters	best	avg	iters	best	avg	iters
Artetxe et al., 2018b	**48.53**	48.13	573	48.47	48.19	773	33.50	32.63	988	37.60	37.33	808
Noise-aware Alignment	**48.53**	**48.20**	471	**49.67**	**48.89**	568	**33.98**	**33.68**	502	**38.40**	**37.79**	551

Table 1: Bilingual Experiment P@1. Numbers are based on 10 runs of each method. The En→De, En→Fi and En→Es improvements are significant at $p < 0.05$ according to ANOVA on the different runs.

Experiment Results In Table 1 we report the best and average precision@1 scores and the average number of iterations among 10 experiments, for different language translations. Our model improves the results in the translation tasks. In most setups our average case is better than the former best case. In addition, the noise-aware model is more stable and therefore requires fewer iterations to converge. The accuracy improvements are small but consistent, and we note that we consider them as a lower-bound on the actual improvements as the current test set comes from the same distribution of the training set, and also contains similarly noisy pairs. Using the soft-EM version results in similar results, but takes roughly 15% more iterations to converge.

Table 2 lists examples of pairs that were kept and discarded in En-It dictionary. The algorithm learned the pair (dog → dog) is an error. Another example is the translation (good → santo) which is a less-popular word-sense than (good → buon / buona). When analyzing the En-It cleaned dictionary we see the percentage of potentially misleading pairs (same string, numbers and special characters) is reduced from 12.1% to 4.6%.

English	Italian	Latent Variable
dog	cane	Aligned
dog	cani	Aligned
dog	dog	Noise
good	buon	Aligned
good	buona	Aligned
good	santo	Noise
new	new	Noise
new	york	Noise
new	nuove	Aligned

Table 2: A sample of decisions from the noise-aware alignment on the English → Italian dataset.

5.2 Diachronic (Historical) Word Embedding

Experiment Setup The goal is to align English word-embedding derived from texts from different time periods, in order to identify which words changed meaning over time. The assumption is that most words remained stable, and hence the supervision is derived by aligning each word to itself. This problem contains noise in the lexicon by definition. We follow the exact setup fully described in Hamilton et al. (2016), but replace the OP algorithm with our Noise-aware version [1]. We project 1900s embeddings to 1990s embeddings vector-space. The top 10 distant word embeddings after alignment are analyzed by linguistic experts for semantic shift.

Experiment Results 45.5% of the input pairs were identified as noise. After the post processing of removing the non-frequent words as described in the experiment setup we end up with 121 noisy words. Our algorithm successfully identifies **all** the top-changing words in Hamilton et al. (2016) as noise, and learns to ignore them in the alignment. In addition, we argue our method provides better alignment. Table 3 shows the Nearest Neighbor (NN) of a 1990s word, in the 1900s vector-space after projection. We look at the top 10 changed words in Hamilton et al. (2016) and 3 unchanged words. We compare the alignment of the OP projection to the Noise-aware Alignment (NAA). For example, with our solution the word *actually* whose meaning shifted from ”in fact” to express emphasize or surprise, is correctly mapped to *really* instead of *believed*. The word *gay* shifted from *cheerful* to *homosexual*, yet is still mapped to *gay* with NAA. This happens because the related embeddings (*homosexual*, *lesbian* and so on) are empty embeddings in 1900s, leaving *gay* as the next-best candidate, which we argue is better than OP's *society*. The words *car, driver, eve* whose meaning didn't change, were incorrectly aligned with OP to *cab, stepped, anniversary* instead of to themselves.

---

[1] Pre-possessing: removing proper nouns, stop words and empty embeddings. Post-processing: removing words whose frequency is below 10^{-5} in either years.

1990s Word	1900s NN aligned with OP	1900s NN aligned with NAA	Latent Variable
wanting	need	wishing	Noise
gay	society	gay	Noise
check	give	send	Noise
starting	begin	beginning	Noise
major	general	successful	Noise
actually	believed	really	Noise
touching	touched	touching	Noise
harry	hello	john	Noise
headed	halfway	toward	Noise
romance	artists	romance	Noise
car	cab	car	Aligned
driver	stepped	driver	Aligned
eve	anniversary	eve	Aligned

Table 3: Diachronic Semantic Change Experiment. Upper-part: noisy pairs. Bold: real semantic shifts. Underlined: global genre/discourse shifts. Unmarked: corpus artifacts. Bottom-part: clean pairs: Italics: unchanged words, no semantic shift.

6 Conclusion

We introduced the problem of embedding space projection with *noisy* lexicons, and showed that existing projection methods are sensitive in the presence of noise. We proposed an EM algorithm that jointly learns the projection and identifies the noisy pairs. The algorithm can be used as a drop-in replacement for the OP algorithm, and was demonstrated to improve results on two NLP tasks. We provide code at https://github.com/NoaKel/Noise-Aware-Alignment.

Acknowledgments

The work was supported by The Israeli Science Foundation (grant number 1555/15), and by the Israeli ministry of Science, Technology and Space through the Israeli-French Maimonide Cooperation program. We also, thank Roee Aharoni for helpful discussions and suggestions.

References

David Alvarez-Melis and Tommi S Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. *arXiv preprint arXiv:1809.00013*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2018. Projecting embeddings for domain adaption: Joint modeling of sentiment analysis in diverse domains. *arXiv preprint arXiv:1806.04381*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Georgiana Dinu and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *CoRR*, abs/1412.6568.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A discriminative latent-variable model for bilingual lexicon induction. *arXiv preprint arXiv:1808.09334*.

Peter Schnemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR*, abs/1702.03859.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 247–257.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised cross-lingual transfer of word embedding spaces. *arXiv preprint arXiv:1809.03633*.

Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1871–1879.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Understanding and Improving Hidden Representation for Neural Machine Translation

Guanlin Li[ϵ], Lemao Liu[λ], Xintong Li[∞], Conghui Zhu[ϵ] [*], Tiejun Zhao[ϵ], Shuming Shi[λ]
[ϵ]Harbin Institute of Technology, [λ] Tencent AI Lab
[∞] The Chinese University of Hong Kong
{epsilonlee.green, znculee}@gmail.com, {chzhu, tjzhao}@hit.edu.cn,
{redmondliu, shumingshi}@tencent.com

Abstract

Multilayer architectures are currently the gold standard for large-scale neural machine translation. Existing works have explored some methods for understanding the hidden representations, however, they have not sought to improve the translation quality rationally according to their understanding. Towards understanding for performance improvement, we first artificially construct a sequence of nested relative tasks and measure the feature generalization ability of the learned hidden representation over these tasks. Based on our understanding, we then propose to regularize the layer-wise representations with all tree-induced tasks. To overcome the computational bottleneck resulting from the large number of regularization terms, we design efficient approximation methods by selecting a few coarse-to-fine tasks for regularization. Extensive experiments on two widely-used datasets demonstrate the proposed methods only lead to small extra overheads in training but no additional overheads in testing, and achieve consistent improvements (up to +1.3 BLEU) compared to the state-of-the-art translation model.

1 Introduction

Neural machine translation (NMT) has witnessed great successes in recent years (Bahdanau et al., 2014; Wu et al., 2016). Current state-of-the-art (SOTA) NMT models are mainly constructed by a stacked neural architecture consisting of multiple hidden layers from bottom-up, where a classifier is built upon the topmost layer to solve the target task of translation (Gehring et al., 2017; Vaswani et al., 2017). Most works tend to focus on the translation performance of the classifier defined on the topmost layer, however, they do not deeply understand the learned representations of hidden layers. Shi et al. (2016) and Belinkov et al. (2017) attempt to understand the hidden representations through the lens of a few linguistic tasks, while Ding et al. (2017) and Strobelt et al. (2018) propose appealing visualization approaches to understand NMT models including the representation of hidden layers. However, employing the analyses to motivate new methods for better translation, the ultimate goal of understanding NMT, is not achieved in these works.

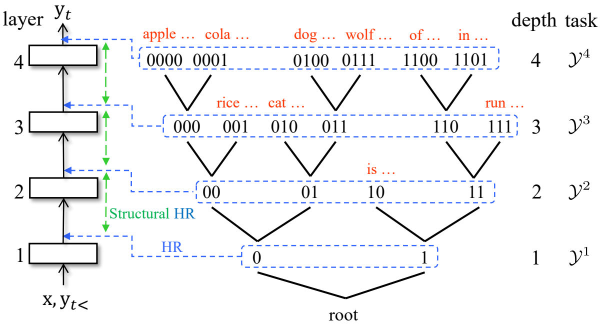


Figure 1: The structural hierarchical regularization framework. On the left is a 4-layer NMT decoder; on the right is a hierarchical clustering tree and the tree-induced relative tasks at every tree depth.

In our paper, we aim at understanding the hidden representation of NMT from an alternative viewpoint, and particularly we propose simple yet effective methods to improve the translation performance based on our understanding. We start from a fundamental question: what are the characteristics of the hidden representation for better translation modeling? Inspired by the lessons from transfer learning (Yosinski et al., 2014), we propose to empirically verify the argument: good hidden representation for a target task should be able to generalize well across *any* similar tasks. Unlike Shi et al. (2016) and Belinkov et al. (2017) who employ one or two linguistic tasks involving human annotated data to evaluate the feature generalization ability of the hidden representation,

---

[*] Conghui Zhu is the corresponding author.

Proceedings of NAACL-HLT 2019, pages 466–477
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

which might make understanding bias to a specific task, we instead construct a nested sequence of *many* relative tasks with entailment structure induced by a hierarchical clustering tree over the output label space (target vocabulary). Each task is defined as predicting the cluster of the next token according to a given source sentence and its translation prefix. Similar to Yu et al. (2018), Zamir et al. (2018) and Belinkov et al. (2017), we measure the feature generalization ability of the hidden representation regarding each task. Our observations are (§2):

- The hidden representations learned by NMT indeed has decent feature generalization ability for the tree-induced relative tasks compared to the randomly initialized NMT model and a strong baseline with lexical features.
- The hidden representations from the higher layers generalize better across tasks than those from the lower layers. And more similar tasks have closer performances.

Based on the above findings, we decide to regularize and improve the hidden representations of NMT for better predictive performances regarding those relative tasks, in hope of achieving improved performance in terms of the target translation task. One natural solution is to feed all relative tasks to every hidden layer of the NMT decoder under the framework of multi-task learning. This may make the full coverage of the potential regularization effect. Unfortunately, this vanilla method is inefficient in training because there are more than one hundred task-layer combinations. [1] Based on the second finding, to approximate the vanilla method, we instead feed a single relative task to each hidden layer as a regularization auxiliary in a coarse-to-fine manner (§3.1). Furthermore, we design another regularization criterion to encourage predictive decision consistency between a pair of adjacent hidden layers, which leads to better approximated regularization effect (§3.2). Our method is simple to implement and efficient for training and testing. Figure 1 illustrates the representation regularization framework.

To summarize, our contributions are as follows:

- We propose an approach to understand hidden representation of multilayer NMT by measuring their feature generalization ability across relative tasks constructed by a hierarchical clustering tree.
- We propose two simple yet effective methods to regularize the hidden representation. These two methods serve as trade-offs between regularization coverage and efficiency with respect to the tree-induced tasks.
- We conduct experiments on two widely used datasets and obtain consistent improvements (up to +1.3 BLEU) over the current SOTA Transformer (Vaswani et al., 2017) model.

2 Understanding Hidden Representation

In this section, we first introduce some background knowledge and notations of the multilayer NMT model. Then, we present a simple approach to better understand hidden representation through transfer learning. By analyzing the feature generalization ability, we draw some constructive conclusions which are used for designing regularization methods in Section 3.

2.1 Background and Notations

Suppose $\mathrm{x} = \langle \mathrm{x}_1, \cdots, \mathrm{x}_{|\mathrm{x}|}\rangle$ is a source sentence, i.e. a sequence of source tokens, and a target sentence $\mathrm{y} = \langle \mathrm{y}_1, \cdots, \mathrm{y}_{|\mathrm{y}|}\rangle$ is a translation of x, where each y_t in y belongs to $\mathcal{Y}$, the target vocabulary. A translation model minimizes the following chain-rule factorized negative log-likelihood loss:

$$\begin{aligned}\ell_{mle} &= -\log P(\mathrm{y} \mid \mathrm{x}; \theta)\\ &= -\sum_t \log P(\mathrm{y}_t \mid \mathrm{x}, \mathrm{y}_{<t}; \theta),\end{aligned} \tag{1}$$

where θ denotes the overall parameter of the translation model. According to Eq.(1), an alternative view of the translation problem can be cast to token-level stepwise **classification** (Daumé et al., 2009): predict the target token y_t given a context consisting of x and $\mathrm{y}_{<t} = \langle \mathrm{y}_1, \cdots, \mathrm{y}_{t-1}\rangle$ corresponding to each factor $P(\mathrm{y}_t \mid \mathrm{x}, \mathrm{y}_{<t}; \theta)$.

The SOTA multilayer NMT models parameterize $P(\mathrm{y}_t \mid \mathrm{x}, \mathrm{y}_{<t}; \theta)$ via powerful multilayer encoder and stacked layers of feature transformations $\langle h^1, \cdots, h^L\rangle$ at the decoder side:

$$P(\mathrm{y}_t \mid \mathrm{x}, \mathrm{y}_{<t}; \theta) = P(\mathrm{y}_t \mid \mathrm{x}, \mathrm{y}_{<t}, h^L; \theta), \tag{2}$$

where $h^l(\mathrm{x}, \mathrm{y}_{<t}) = \phi^l\big(\mathrm{x}, \mathrm{y}_{<t}; h^{l-1}; \theta\big)$ is the l^{th} hidden layer recursively defined by ϕ^l on h^{l-1}.

---

[1] There are about 22 tasks that we have constructed and 6 layers in SOTA NMT models (Vaswani et al., 2017).

We also use $h^l(\mathrm{x}, \mathrm{y}_{<t})$ to represent the output hidden representation of layer l for a specific context. Note that, ϕ^l bears several types of instantiation and is an active area of research (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017).

2.2 Feature Generalization Ability of Hidden Representations

Inspired by feature transfer learning (Yosinski et al., 2014), we attempt to understand hidden representations of NMT by evaluating their generalization abilities across any tasks that are related to translation. There are some researchers who study hidden representations of NMT by using linguistic tasks such as morphology, named entity, part-or-speech or syntax (Shi et al., 2016; Belinkov et al., 2017, 2018). They typically rely on human annotated resources to train a model for each linguistic task, so their methods can not be used for languages which lack human annotations. Moreover, their considered tasks are too few to have a good coverage over task space for measuring transferability (Yu et al., 2018; Zamir et al., 2018), and their understanding results may bias to a specific task. As a result, to evaluate the feature generalization ability of hidden representation, we artificially construct plenty of relative tasks which do not employ any human annotation. This makes our evaluation approach more general.

Definition of the relative tasks Suppose $\mathcal{Y}^k$ denotes any partition (or clustering) regarding the output label space (target vocabulary) $\mathcal{Y}$. That is, $\mathcal{Y}^k$ is a set of subsets $\mathcal{Y}_i^k \subset \mathcal{Y}$ where $i = 1 ... |\mathcal{Y}^k|$, such that $\forall i, j, \mathcal{Y}_i^k \cap \mathcal{Y}_j^k = \emptyset$ and $\cup_i \mathcal{Y}_i^k = \mathcal{Y}$. We define the following relative task: given a context $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$, predict the subset or the cluster to which the t^{th} token y_t belongs in $\mathcal{Y}^k$, denoted as $\mathcal{Y}^k(\mathrm{y}_t)$. To simplify notation, we regard $\mathcal{Y}^k$ both as a relative task and as a partition.

It is clear that the above type of tasks are similar to the task of translation according to the description in Section §2.1. Furthermore, different k represents different relative task and thus we actually obtain a great many relative tasks in total. However, it is impossible to evaluate the hidden representation on all those tasks; moreover, due to relationship between tokens (Hu et al., 2016) in $\mathcal{Y}$, not all partitions are reasonable. As a consequence, motivated by the analysis of VC Dimension (Vapnik, 1995), we construct a sequence of nested partitions with an entailment structure: [2] $\mathcal{Y}^1 \preceq \cdots \preceq \mathcal{Y}^K$. The benefit is that a spectrum of task hardness can be constructed due to the increased partition or task cardinalities.

As a matter of fact, we instantiate the above nested partitions through brown clustering (Brown et al., 1992; Stratos et al., 2014) over $\mathcal{Y}$ to get a hierarchical clustering tree and then consider each tree depth along the tree as a partition representing a relative task $\mathcal{Y}^k$ (as shown in Figure 1). In the following experiments, we run brown clustering algorithm over a Ch⇒En dataset (§4) and construct a tree of English with depth 21. Without loss of generality, we regard the task $\mathcal{Y}^{22}$ at a virtual 22 depth of the tree as equivalent to the translation task $\mathcal{Y}$. Actually, $\mathcal{Y}$ and $\mathcal{Y}^{22}$ have the same cardinality but are different in definition. [3]

Evaluating generalization We use multi-class logistic regression to fit the layer-wise hidden representation learned by a well-trained 6-layer Transformer (Vaswani et al., 2017) over each training instance $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$. Specifically, given a context $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$, for each task $\mathcal{Y}^k$ and a hidden representation $h^l(\mathrm{x}, \mathrm{y}_{<t})$ of this context, which is fixed as constant, we predict the cluster $\mathcal{Y}^k(\mathrm{y}_t)$ according to the following probability:

$$P\big(\mathcal{Y}^k(\mathrm{y}_t) \mid h^l(\mathrm{x}, \mathrm{y}_{<t}); \theta^l_{\mathcal{Y}^k}\big), \tag{3}$$

where $\theta^l_{\mathcal{Y}^k}$ is the parameter of the logistic regression model for task $\mathcal{Y}^k$ at l^{th} layer. The difference between Eq.(3) and Eq.(2) is that the former is the linear model parameterized by $\theta^l_{\mathcal{Y}^k}$ while the latter is the NMT model parameterized by θ.

Since there are $L = 6$ layers in Transformer's decoder and $K = 22$ relative tasks, we have more than one hundred such linear models defined with Eq. (3) in total. Therefore, it is costly to train them independently. Since the loss for each linear model is convex, joint training leads to exactly the same optimum as independent training and thus we employ mini-batch stochastic gradient descent to minimize the joint loss as follows:

$$-\sum_k \sum_l \sum_t \log P\big(\mathcal{Y}^k(\mathrm{y}_t) \mid h^l(\mathrm{x}, \mathrm{y}_{<t}); \theta^l_{\mathcal{Y}^k}\big). \tag{4}$$

---

[2] Here what we mean entailment relation ($\preceq$) between two partitions $\mathcal{Y}^k$ and $\mathcal{Y}^{k+1}$ is: $\forall \mathcal{Y}_i^{k+1}, \exists! \mathcal{Y}_j^k, s.t. \mathcal{Y}_i^{k+1} \subseteq \mathcal{Y}_j^k$.

[3] Please refer to Appendix A for detailed preprocessing of the tree to get nested partitions.

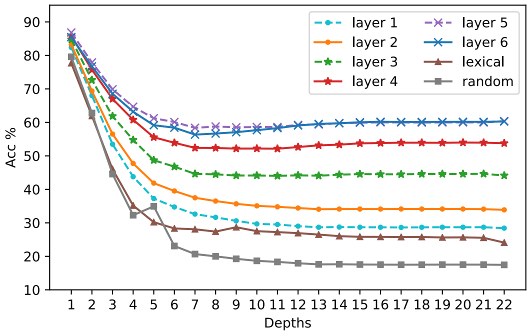


Figure 2: The transfer learning performances.

After training, we fix each $\theta^l_{\mathcal{Y}^k}$ and then measure the feature generalization ability of each h^l by validating on the task $\mathcal{Y}^k$ regarding a heldout dataset, following Yu et al. (2018). For validation, we report accuracy on a heldout dataset through the strategy of maximum a posteriori (MAP). [4]

Analysis To figure out how good the learned hidden representations are, we consider two **baselines** to extract features regarding each context $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$ to train logistic regression models for comparison. For the first baseline, the features of the context are the hidden representations from the last layer of a randomly initialized Transformer; for the second, the features are derived by lexical feature templates, which include the source-side bag-of-words (BOW) features and target-side BOW features indexed by relative positions of y_t's previous with up to m (Markov length) tokens. [5]

As shown in Figure 2, the lexical baseline delivers comparable accuracies for fine-grained tasks with respect to well learned Transformer's first layer, thanks to its discriminant ability with abundant lexical features. For example, its accuracy reaches about 26% for the task with cardinality $|\mathcal{Y}^{21}|$. The random baseline performs worse for tasks with cardinality $|\mathcal{Y}^{8}|$, which indicates that random representations in NMT have limited generalization abilities to fine-grained tasks as expected. The well-trained low-layer hidden representations yield much higher accuracies than the random baseline and are even better than the lexical baseline. This shows that the hidden representations from a well-trained NMT have good generalization abilities across relative tasks. In addition, as the layer goes up, the performance of hidden representations increase significantly over different relative tasks, which clearly demonstrates that more complex neural architecture leads to stronger expressibility. This provides a quantitative evidence to support the statement in Bengio et al. (2009), Goodfellow et al. (2016).

3 Structural Hierarchical Regularization

In this section, we propose two simple methods, which respect the above findings, to enhance the hidden representations in NMT such that they generalize well across those relative tasks.

3.1 Hierarchical Regularization

A natural method to improve feature generalization of hidden representation is to jointly train the target task with all relative tasks for all hidden layers, which we call full-coverage method. As mentioned in Section §2.2, this method will lead to training more than one hundred tasks ($K \times L$) in total, where K denotes the depth of the hierarchical clustering tree (aka. the number of tasks) and L the number of hidden layers. Unfortunately, since each task involves a softmax operation which may be the computation bottleneck for the task $\mathcal{Y}^k$ with large cardinality, this method is inefficient for training.

As a solution to approximate the potential regularization effect of the full-coverage method, we confine each hidden layer to engage in a single relative task. Motivated by the observation that representations from higher layers have better expressibility than lower layers, as claimed in §2.2, we instead employ a coarse-to-fine strategy to select one task for each layer: finer-grained tasks for higher layers while coarser-grained task for lower layers. Specifically, suppose $1 \leq s(l) \leq K$ is the selected index regarding task $\mathcal{Y}^{s(l)}$ for the l^{th} layer, then it subjects to $s(l) < s(l+1)$ for each l. In addition, to encourage the diversity among the selected L tasks, we require $s(l+1) - s(l)$ to be large enough for all l. Formally, the loss of the hierarchical regularization (HR) method is:

$$\ell_{hr} = -\sum_{l} \sum_{t} \log P(\mathcal{Y}^{s(l)}(\mathrm{y}_t) \mid \mathrm{x}, \mathrm{y}_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}}), \quad (5)$$

where $P(\mathcal{Y}^{s(l)}(\mathrm{y}_t) \mid \mathrm{x}^j, \mathrm{y}^j_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}})$ is similar to Eq. 3 except that it treats the parameters θ in NMT as parameters besides $\theta^l_{\mathcal{Y}^{s(l)}}$. Compared to Eq. 4, it includes fewer terms for summation.

---

[4] The accuracy is measured by whether $\arg\max_{z \in \mathcal{Y}^k} P(z \mid h^l(\mathrm{x}, \mathrm{y}_{<t}); \theta^l_{\mathcal{Y}^k})) = \mathcal{Y}^k(\mathrm{y}_t)$.

[5] Please refer to Appendix B for more details are.

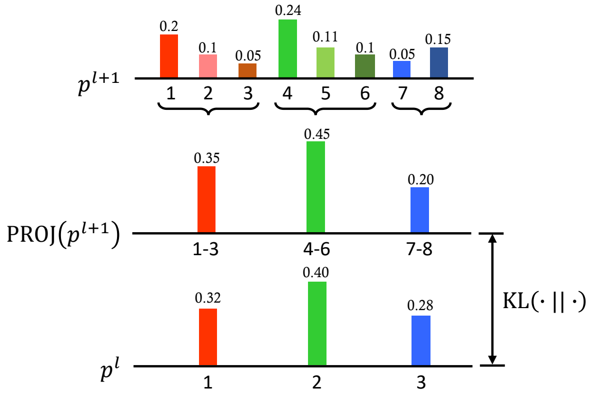


Figure 3: The conceptual graph of the KL consistency loss in SHR. Here, the multinomial probability vector p^l is calculated through $P(\cdot \mid \mathrm{x}, \mathrm{y}_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}})$.

3.2 Structural Hierarchical Regularization

The HR method is very simple and computationally efficient, however, using one task to regularize a layer may not be a good approximation of the full-coverage method, since HR method might lead to inconsistent decisions for two different layers, which is formalized through the following entailment structure as introduced in Section 2.2:

$$\arg\max_{\mathrm{z}\in\mathcal{Y}^{s(l_1)}} P(\mathrm{z} \mid \mathrm{x}, \mathrm{y}_{<t}, h^{l_1}; \theta, \theta^{l_1}_{\mathcal{Y}^{s(l_1)}}) \supset \arg\max_{\mathrm{z}\in\mathcal{Y}^{s(l_2)}} P(\mathrm{z} \mid \mathrm{x}, \mathrm{y}_{<t}, h^{l_2}; \theta, \theta^{l_2}_{\mathcal{Y}^{s(l_2)}}), \quad (6)$$

where $s(l)$ is the selected task for the l^{th} layer by HR, $1 \leq l_1 < l_2 \leq L$ and $P(\mathrm{z}|\mathrm{x}, \mathrm{y}_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}})$ is similar to Eq. (3) for the task $\mathcal{Y}^{s(l)}$ and l^{th} layer except that it does not treat the NMT parameters θ as constant. However, it always occurs on the training data that $\mathcal{Y}^{s(l_1)}(\mathrm{y}_t) \subset \mathcal{Y}^{s(l_2)}(\mathrm{y}_t)$.

To alleviate this inconsistency issue for better approximating the full-coverage method, we leverage the above structural property by adding another regularization term. Firstly, we project the distribution $P(\cdot|\mathrm{x}, \mathrm{y}_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}})$ into the domain of $\mathcal{Y}^{s(l-1)}$. Then we calculate KL divergence between the projected distribution and $P(\cdot|\mathrm{x}, \mathrm{y}_{<t}, h^{l-1}; \theta, \theta^{l-1}_{\mathcal{Y}^{s(l-1)}})$. Figure 3 illustrates the idea. Since it is inefficient to consider all pairs of l_1 and l_2, so we instead consider the consistency between all adjacent layers. Formally, we obtain the following loss function:

$$\ell_{shr} = \ell_{hr} + \frac{1}{L-1}\sum_{l} \mathrm{KL}\Big(P(\cdot|\mathrm{x}, \mathrm{y}_{<t}, h^l; \theta, \theta^l_{\mathcal{Y}^{s(l)}}) \mid\mid \mathrm{PROJ}\big[P(\cdot|\mathrm{x}, \mathrm{y}_{<t}, h^{l+1}; \theta, \theta^{l+1}_{\mathcal{Y}^{s(l+1)}})\big]\Big), \quad (7)$$

where PROJ is the projection defined in Figure 3, and other notations are defined as before.

Method	# Param.		Speed	
	Zh⇒En	**En⇒De**	**Train**	**Test**
Baseline	90.2M	88.0M	1.82	1.33
FHR	+60.9M	+63.2M	0.72	
HR	+3.2M	+3.1M	1.65	
SHR	+3.2M	+3.1M	1.40	

Table 1: Train and test efficiency comparison, measured by steps/s (second) and sentences/s respectively.

We call the above regularization as structural hierarchical regularization (SHR) since it takes advantage of the structure of the tree. In our experiments, we add HR (Eq.(5)) and SHR (Eq.(7)) losses respectively into the negative log-likelihood regarding Eq. (1) for training all parameters θ and $\theta^l_{\mathcal{Y}^{s(l)}}$. One of our advantage is that we only use θ for testing and thus our testing is as efficient as that for the baseline NMT model.

4 Experiments

4.1 Setup

We conduct experiments on two widely-used corpora. We choose from the LDC corpora about 1.8M sentence pairs for Zh⇒En translation with word-level vocabulary of 30k for both languages. We use the WMT14 En⇒De task which consists 4.5M sentence pairs and the vocabulary is built by joint BPE with 32k merging operations. Besides the baseline, we also conduct experiments on 3 regularization variants:

- **Baseline**: the Transformer base model proposed in Vaswani et al. (2017).
- **FHR**: fine-grained HR based Transformer, which adopts the original label space as task for all selected layers for regularization. This variant is used to demonstrate that low layers which are weak in expressibility can mess up hard tasks which are unsuitable to learn.
- **HR** and **SHR**: as proposed in Section 3.

Method	MT02	MT03	MT04	MT05	MT06	MT08	Avg.
Zhao et al. (2018)	N/A	44.98	45.51	43.93	43.95	33.33	42.34
Baseline	46.08	44.09	46.50	44.45	45.26	37.10	43.48
FHR	45.46	43.56	47.51	44.00	45.45	37.22	43.58
HR	46.28	44.04	47.80	44.56	45.56	38.17	44.08
SHR	**47.05**	**44.80**	**48.15**	**45.55**	**46.30**	**39.02**	**44.78**

Table 2: BLEU comparison on the LDC dataset.

Choice of relative tasks Based on the heuristics in Section 3.1, we first choose the task with the largest cardinality from the hierarchical clustering tree without the virtual depth, because this task is most related to translation (close cardinalities). Then we balance task diversity through a 5 times cardinality difference between tasks from the previous chosen task. As a result, we can obtain 4 tasks with $s(l) = 5, 8, 11, 20$ for the Zh⇒En task and $s(l) = 5, 7, 10, 21$ for the En⇒De task, where $l = 2, 3, 4, 5$ of the 6-layer decoder. [6]

4.2 Efficiency Comparison

Table 1 summarizes the total number of parameters for the baseline and 3 regularization variants. As in Eq. (5), HR introduces extra parameters compared with the baseline. Besides, calculating the second term in Eq. (7) requires modest overheads. Therefore, training our SHR is slower than training the baseline. Although the proposed HR and SHR introduce extra parameters during training, they do not involve them during testing and thus testing is as efficient as the baseline.

4.3 Translation Quality on Zh⇒En Dataset

Table 2 shows the evaluation results of the baseline and 3 regularization variants on the Zh⇒En dataset. Since there are no recent work reporting Transformer's performance on this dataset, we choose a recurrent SOTA model to show that our baseline is already better than it, which is a common knowledge that Transformer can outperform recurrent NMT models. Our HR method surpasses the baseline 0.6 BLEU point, while the SHR method can improve upon HR by about a further 0.8 point, namely about 1.4 points over the baseline. Interestingly, the FHR method only performs on par with baseline, which indicates that forcing low layers to learn fine-grained tasks will not lead to beneficial intermediate representations since they struggle to learn a well-structured representation space. This matches the finding in Section 2: low layers may not be expressible enough to perform well on tasks with large cardinalities.

4.4 Analyses on Zh⇒En Dataset

In the following, we conduct several quantitative experiments to demonstrate the advantages of our proposed two regularization methods over the baseline. Note that, since we need to guarantee that the decoded sequence has the same length with the reference for one-by-one token comparison, the following experiments are all conducted with teacher forcing and greedy decoding.

4.4.1 Better Feature Generalization Ability

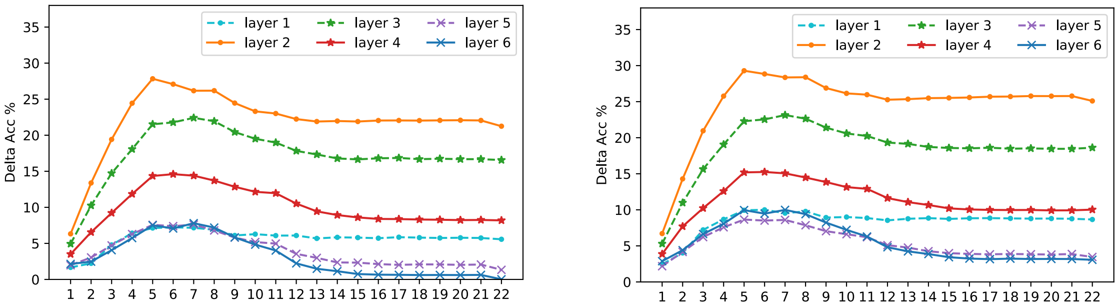
In the same manner as Section 2, we learn softmax weights for all relative tasks by fixing model weights learned by HR and SHR methods. Figure 4(a), (b) show the Δ feature generalization ability (absolute accuracy difference) of HR and SHR over baseline. Since layer 1 is not selected as the regularized layer, no significant gap is observed. However, since layer 1 is close to the loss directly imposed on layer 2, improvements about 5% and 8% are obtained. Since in the baseline, layer 5, 6 are already close or with the ultimate fine-grained loss, HR method shows very small gain. But our SHR method can still improve about 4% absolute points. Except for layer 1, it is also evident to see larger gaps (more than 20%) at lower layers than higher layers due to the fact that lower layers, which are distant from the topmost loss in the baseline, require more supervision signals to shape their latent representation space.

4.4.2 Improved Decision Consistency

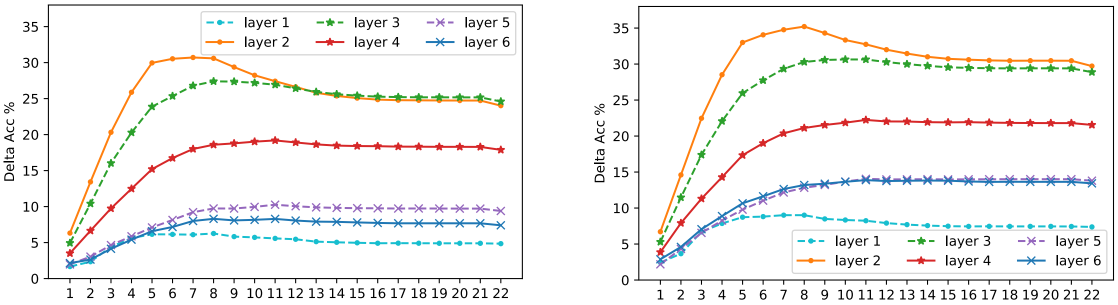
We measure decision consistency for a specific layer and decision consistency between a pair of layers using two metrics. The first metric is measured by conditional accuracy, which is the possibilities of the classifier parameterized by $\theta^l_{\mathcal{Y}^k}$ correctly predicting $\mathcal{Y}^k(\mathrm{y}_t)$ if the classifier parameterized by $\theta^l_{\mathcal{Y}^{k'}}$ correctly predicts $\mathcal{Y}^{k'}(\mathrm{y}_t)$ for any

---

[6] Please refer to Appendix C for detailed information.



(a) HR vs Baseline (b) SHR vs Baseline

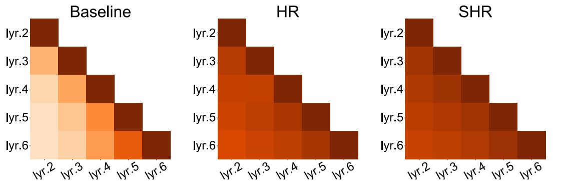


(c) HR vs Baseline (d) SHR vs Baseline

Figure 4: (a), (b) are the Δ feature generalization ability (absolute accuracy difference) of HR and SHR method compared to baseline; (c), (d) are the conditional absolute accuracy difference of HR and SHR over baseline.

$k' < k$. The second metric is measured by the counts of consistent decision pairs between any pair of regularized layers as defined in Eq. (6).

Figure 4(c), (d) shows the absolute conditional accuracy difference of our HR and SHR over baseline. In accordance with the observations in previous subsection, except for layer 1, other layers show significant gains (HR more than 7%, SHR more than 10%) over baseline. Decision consistency for each layer proves the well-shaped layer-wise representation and potentially paves the way for better inter-layer decision consistency.



(a) (b) (c)

Figure 5: The consistency correlation between different regularized layers (lyr.2 to lyr.6) of the baseline and our two regularization methods.

Figure 5 illustrates the consistency counts between any regularized layer pairs, including those without KL-based regularization. Deeper color represents more consistency counts. It is evident that the baseline has a very poor consistency between any layers. Our HR method is almost 2 times better, and the SHR obtains further improvement. A better decision consistency can couple the decision between relative tasks, so that by reaching a high accuracy on easier tasks can benefit the harder ones. Another interesting observation is that non-adjacent layers without KL loss also obtain significant improvements on decision consistency, because the KL term is actually transitive between layers where the predictive distributions are in accordance with the tree structure.

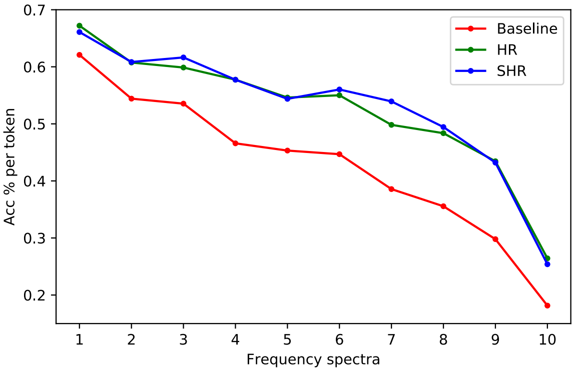


Figure 6: Development set accuracy among the baseline and our proposed regularization variants over different word frequency bins.

4.4.3 Promoted Low-Frequency Word Performance

In this subsection, we clarify that the coarse-to-fine regularized representations can also benefit low-frequency words. We divide the vocabulary into ten equally-sized bins, and summarize token accuracy for each bin over the development set. As shown in Figure 6, the x-axis represents the frequency spectra, that is, we sort the bins by word frequency from rank 1 (the most frequent words) to 10 (the rare words). We can see that both HR and SHR methods demonstrate a gradually increased gap over the baseline as the word frequency decreases, which means our methods become better for less frequent word bins. However the gap shrinks at the 10^{th} bin. This may be the fact that for those words that appears with less than 50 counts, both methods are helpless.

For baseline, it is hard to train well-shaped hidden representations for low-frequent words; in addition, due to the distance between the loss and the low layers, it is also hard to train weights due to the unstable gradient signal. By adding our regularization terms, every level of the multilayer decoder will receive supervision signals directly and lower layers will receive coarser grained thus higher frequency signals to shape their representations.

4.5 Translation Quality on En⇒De Dataset

Table 3 shows the evaluation results of the baseline and the 3 regularization variants on the En⇒De dataset. Notice that we use the base model while Chen et al. (2018) and Ott et al. (2018) use big models. The FHR method still does not show significant improvement over the baseline (less than 0.2 BLEU point), which verifies the hypothesis that we make by analyzing the Zh⇒En results. Our HR method is already stronger than Chen et al. (2018) which uses a multilayer RNN as decoder. Compared to the current state-of-the-art in Ott et al. (2018) who utilize huge batch size and over 100 GPUs on the Transformer big model, our SHR method can be on par with them. This comparison indicates that better regularized hidden representations can be potentially powerful than increasing model capacity when using the same optimization method.

5 Related Work

Since the dawn of NMT, many works have proposed for understanding what has been encoded in the learned hidden representations. Shi et al. (2016) are the first to investigate source syntax encoded in source hidden representations. Similarly, Belinkov et al. (2017) and Belinkov et al. (2018) give detailed analyses of both encoder and decoder's learned knowledge about part-of-speech and semantic tags at different layers. Unlike those works that employ one or two linguistic tasks, we instead construct plenty of artificial tasks without any human annotations to analyze the hidden representations. This makes our approach more general and may potentially lead to less biased conclusions.

Method	MT13	MT14
Chen et al. (2018)	N/A	28.49
Ott et al. (2018)	**26.70**	**29.30**
Baseline	25.99	27.75
FHR	26.10	27.91
HR	26.48	28.87
SHR	**26.64**	**29.18**

Table 3: BLEU comparison on the WMT14 dataset. Here MT13 and MT14 denote newstest2013 and newstest2014, which are used as development and test set respectively.

Based on our understanding of the hidden representations, we further develop simple methods to improve NMT through representation regularization. Many works regularize NMT with lexical knowledge such as BOW (Weng et al., 2017) and morphology (Niehues and Cho, 2017; Zaremoodi et al., 2018), or syntactic knowledge (Kiperwasser and Ballesteros, 2018; Eriguchi et al., 2017). One significant difference is that we take into account the structure among plenty of artificial tasks and design a well motivated regularization term to encourage the structural consistency of tasks, which further improves NMT performance. In addition, our coarse-to-fine way to select tasks for regularization is also inspired by recent works using a coarse-to-fine mechanism for learning better word embeddings in NMT (Zhang et al., 2018) and predicting intermediate solutions for semantic parsing (Dong and Lapata, 2018).

6 Conclusion

In this work, we present a simple approach for better understanding NMT learned layer-wise representations with transfer learning over plenty of artificially constructed relative tasks. This approach is general as it requires no human annotated data, only demanding target monolingual corpus. Based on our understanding, we propose two efficient yet effective methods for representation regularization which further pushes forward the SOTA NMT performances. In the future, we want to dig deeply into the subspace regularities of the learned representations for more fine-grained understanding.

Acknowledgements

The authors would like to first thank all the anonymous reviewers for their critical suggestion and valuable experimental advice. The authors would also like to thank Yong Jiang for discussion; Shangchen Zhou for better figure design; Haozhe Xie, Chaoqun Duan, Xin Li, Ziyi Dou and Mengzhou Xia for proof reading. Tiejun Zhao is supported by National Key RD Program of China Project 2017YFB1002102.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Yoshua Bengio et al. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Hexiang Hu, Guang-Tong Zhou, Zhiwei Deng, Zicheng Liao, and Greg Mori. 2016. Learning structured inference neural networks with label relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2960–2968.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. 2017. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *arXiv preprint arXiv:1804.08915*.

Marc T Law, Jake Snell, Amir-massoud Farahmand, Raquel Urtasun, and Richard S Zemel. 2018. Dimensionality reduction for representing the knowledge of probabilistic models.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. *arXiv preprint arXiv:1708.00993*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation*, pages 1–9, Belgium, Brussels. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel J Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. In *UAI*, pages 762–771. Citeseer.

H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. 2018. Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models. *ArXiv e-prints*.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xinyu Dai, and Jiajun Chen. 2017. Neural machine translation with word predictions. *arXiv preprint arXiv:1708.01771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215. Association for Computational Linguistics.

Amir R Zamir, Alexander Sax, , William B Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 656–661.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Coarse-to-fine learning for neural machine translation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 316–328. Springer.

Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. 2018. Addressing troublesome words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400. Association for Computational Linguistics.

A Hierarchical Clustering Tree Induced Relative Tasks

In this appendix, we demonstrate a more detailed introduction to the partition of $\mathcal{Y}$, the hierarchical clustering tree constructed by $\mathcal{Y}$ and the tree-induced relative tasks, which are introduced in Section 2.2, through an example.

Suppose our vocabulary $\mathcal{Y}$ only consists of a few words, that is, $\mathcal{Y} = \{\text{cat, dog, run, jump, is}\}$. Any partition of $\mathcal{Y}$ denoted as $\mathcal{Y}^k$ is a set of subsets of $\mathcal{Y}$. As shown in Figure 7, this partition is actually { { cat, dog }, { jump, run }, { is } }.

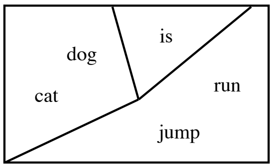


Figure 7: One example partition of the vocabulary $\mathcal{Y}$.

Then suppose the constructed hierarchical clustering tree looks like the tree in Figure 8(a). In this tree, not all the leaves are at the same depth, so we left-branches the leaves with {cat, dog} at depth 1 to stretch to depth 2. Then by adding a virtual level at depth 3, we can construct a relative task with the same cardinality as the fine-grained translation task $\mathcal{Y}$, as shown in Figure 9(b).

B Lexical Feature-based Baseline

In this appendix, we describe the lexical feature-based baseline that we use in Section 2.2 to compare with the layer-wise representations learned by Transformer. There are two types of feature template: a) the source-side bag-of-word (BOW) features; and b) the target-side order-aware BOW features. Specifically, given a context $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$, we extract features according to the above templates:

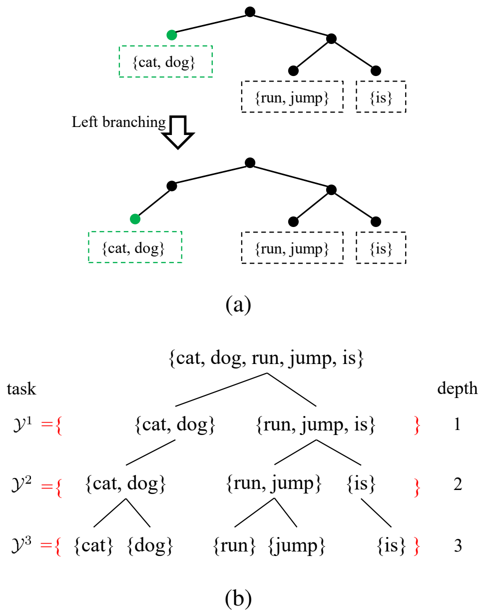
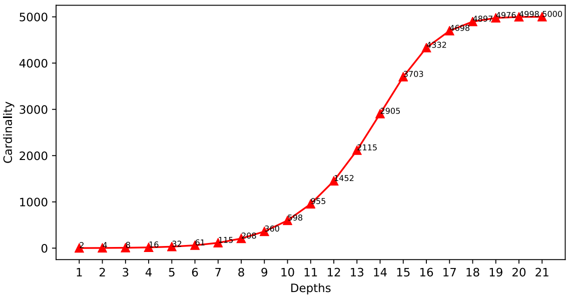
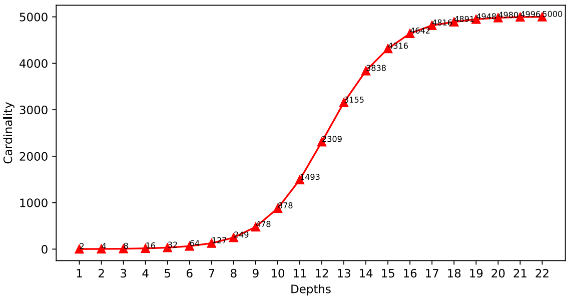


Figure 8: (a) The original hierarchical clustering tree, with left branching to have all leaves at the same tree depth; (b) The hierarchical clustering tree with depth 2, and a virtual task is constructed at depth 3.



(a) Task cardinalities for English (Zh⇒En).



(b) Task cardinalities for German (En⇒De).

Figure 9: Task cardinalities.

- the BOW representation of the source sentence x: that is, if the source vocabulary is $\mathcal{X}$, the source BOW feature vector has length of $|\mathcal{X}|$, with each entry the appearances of that token in x.

- the order-aware BOW representation within k-Markov dependency chain: that is, we extract feature from $\mathrm{y}_{t-k:t-1}$ by considering both the token identity and the relative distance of that token to the predicted one y_t. This feature template constructs a feature vector of $k \times |\mathcal{Y}|$ entries ($\mathcal{Y}$ the target vocabulary), with each entry set to 1 when appears in the chain, otherwise 0.

- the order-unaware BOW representation outside the k-Markov dependency chain, that is, we extract feature from the $\mathrm{y}_{0:t-k-1}$ with the same philosophy of the source-side BOW feature and obtain a feature vector of size $\mathcal{Y}$.

So in total, the feature vector extracted from the context $\langle \mathrm{x}, \mathrm{y}_{<t} \rangle$ has a size $(|\mathcal{X}| + k \times |\mathcal{Y}|)$, which will be around 300k if the vocabulary size is around 30k and the Markov order $k = 10$.

C Detailed Experiment Information

In this appendix, we describe the detailed experiment information: the construction of the hierarchical clustering tree (Brown et al., 1992; Stratos et al., 2014), the model configuration, and the training details.

We construct two hierarchical clustering trees for the two target languages, English (Zh⇒En) and German (En⇒De) in our experiments, with Percy Liang's Brown Clustering algorithm implementation. [7] In both languages, we set the number of clusters to 5000 (a hyperparameter in the algorithm, $c = 5000$), that is, we will obtain trees with 5000 leaves. We set c to 5000 since the total vocabulary of the two languages are around 30k, and 5000 clusters will get a token coverage of 6 (30k/5000=6) for each leave if the clusters are balanced. By using left-branching introduced in Appendix A, we can finally get two trees with every depths as a relative task. The statistics of each relative task's cardinality for each tree are shown in Figure 9(a) and (b) respectively. For selecting the depths in a coarse-to-fine manner, we follow the heuristics mentioned in Section 3.1, that is, we select depths which are diverse enough so as to have better coverage of all the relative tasks. Specifically, we follow a quotient between two adjacent selected tasks of 5, and select from the task which has the cardinality of 5000, then we select tasks of cardinalities around 1000, 200, 40 respectively.

---

[7] https://github.com/percyliang/brown-cluster.

Method	**MT02**	**MT03**	**MT04**	**MT05**	**MT06**	**MT08**	**Avg.**
Baseline	46.08	44.09	46.50	44.45	45.26	37.10	43.48
L2	46.67	44.50	47.35	45.02	46.20	38.43	44.30
L3	46.40	44.65	46.90	45.02	45.95	37.92	44.08
L4	46.35	44.30	46.97	45.10	46.06	37.31	43.95
L5	46.29	44.57	46.97	44.75	45.45	37.74	43.89
HR	46.28	44.04	47.80	44.56	45.56	38.17	44.08
SHR	**47.05**	**44.80**	**48.15**	**45.55**	**46.30**	**39.02**	**44.78**

Table 4: BLEU comparison on the LDC dataset with independently regularized layers.

For the Zh⇒En dataset, we select tasks at depths 5, 8, 11, 20 with cardinalities 32, 208, 955, 5000; for the En⇒De dataset, we select tasks at depths 5, 7, 10, 21 with cardinalities 32, 127, 878, 5000.

The model configuration strictly follows that of the base model in Vaswani et al. (2017) with word embedding size of 512, hidden size of 512, feed-forward projection size of 2048, layer and head number of 6 and 8. The dropout rates of the embedding, attention and residual block are all set to 0.1. All architectural settings are in accordance with the base model of Vaswani et al. (2017).

We use Adam (Kingma and Ba, 2014) as the default optimizer with an initial learning rate of 0.0001. Training batch size is set to 8192 × 4 tokens per batch, that is, we use data parallelism with 4 P40 or M40 GPUs and 8192 tokens per GPU. We train the Zh⇒En models from scratch for about 240k iterations about 2 weeks on 4 M40 GPUs for both the baseline and the 3 regularization variants. For the En⇒De models, we first attempt to train all the methods from scratch up to 200k iterations, but do not see significant improvement on BLEU score (around 0.6 points on test). So we use the pretrained baseline to initialize our proposed methods, and further train them for about 200k iterations, which results in the reported improvement in Section 4.5.

D Experiments of Independent Layer Regularization on Zh⇒En

As suggested by the reviewers, we conduct independent layer regularization by imposing a relative task with cardinality of 5000 on layer 2 to 5 with the six layer Transformer. The performances are demonstrated in Table 4. It seems that independent layer regularization can as well brings about descent improvements over the baseline. And regularizing layer 2 of the decoder performs better than our HR method. It is surprising that lower layers are more urgent to be regularized than higher layers. This phenomenon may raise the question that how on earth the intermediate layer representations help with final prediction. One hypothesis may be drawn from our paper is that: in baseline, the coherence among the layer-wise representations may be weak so that some non-linear transformations from lower layer may not lead to essential predictive power of the final layer representation. And by using KL divergence to externally constrain their decision consistency may take better advantage of lower layers. As one of the reviewer pointed out, Universal Transformer (Dehghani et al., 2018) with base model's hyperparameters achieves 28.90 on WMT14 En⇒De newstest14, with 1/6 enc-dec parameters (not considering embeddings). Its architectural inductive bias is motivated from iterative refinement of the layer-wise representation so the decoder at each time step builds an RNN like reasoning process to refine upon previous layer's representation. We think this might be a more effective inductive bias in ResNet architecture which is adopted by Transformer, since Jastrzebski et al. (2017) provides evidence that ResNet does iterative representation inference (residual as refined quantity). Future directions may include relating the dimension reduced representations (Law et al., 2018) to the coarse-to-fine structural bias, or experiments on Universal Transformer architecture to probe its learned representations.

Content Differences in Syntactic and Semantic Representations

Daniel Hershcovich[1,2] **Omri Abend**[2] **Ari Rappoport**[2]
[1]The Edmond and Lily Safra Center for Brain Sciences
[2]School of Computer Science and Engineering
Hebrew University of Jerusalem
{danielh,oabend,arir}@cs.huji.ac.il

Abstract

Syntactic analysis plays an important role in semantic parsing, but the nature of this role remains a topic of ongoing debate. The debate has been constrained by the scarcity of empirical comparative studies between syntactic and semantic schemes, which hinders the development of parsing methods informed by the details of target schemes and constructions. We target this gap, and take Universal Dependencies (UD) and UCCA as a test case. After abstracting away from differences of convention or formalism, we find that most content divergences can be ascribed to: (1) UCCA's distinction between a Scene and a non-Scene; (2) UCCA's distinction between primary relations, secondary ones and participants; (3) different treatment of multi-word expressions, and (4) different treatment of inter-clause linkage. We further discuss the long tail of cases where the two schemes take markedly different approaches. Finally, we show that the proposed comparison methodology can be used for fine-grained evaluation of UCCA parsing, highlighting both challenges and potential sources for improvement. The substantial differences between the schemes suggest that semantic parsers are likely to benefit downstream text understanding applications beyond their syntactic counterparts.

1 Introduction

Semantic representations hold promise due to their ability to transparently reflect distinctions relevant for text understanding applications. For example, syntactic representations are usually sensitive to distinctions based on POS (part of speech), such as between compounds and possessives. Semantic schemes are less likely to make this distinction since a possessive can often be paraphrased as a compound and vice versa (e.g., "US president"/"president of the US"), but may distinguish different senses of possessives (e.g., "some of the presidents" and "inauguration of the presidents").

Nevertheless, little empirical study has been done on what distinguishes semantic schemes from syntactic ones, which are still in many cases the backbone of text understanding systems. Such studies are essential for (1) determining whether and to what extent semantic methods should be adopted for text understanding applications; (2) defining better inductive biases for semantic parsers, and allowing better use of information encoded in syntax; (3) pointing at semantic distinctions unlikely to be resolved by syntax.

The importance of such an empirical study is emphasized by the ongoing discussion as to what role syntax should play in semantic parsing, if any (Swayamdipta et al., 2018; Strubell et al., 2018; He et al., 2018; Cai et al., 2018). See §8.

This paper aims to address this gap, focusing on *content* differences. As a test case, we compare relatively similar schemes (§2): the syntactic Universal Dependencies (UD; Nivre et al., 2016), and the semantic Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013).

We UCCA-annotate the entire web reviews section of the UD EWT corpus (§3), and develop a converter to assimilate UD and UCCA, which use formally different graphs (§4). We then align their nodes, and identify which UCCA categories match which UD relations, and which are unmatched.

Most content differences are due to (§5):

1. UCCA's distinction between words and phrases that evoke Scenes (events) and ones that do not. For example, eventive and non-eventive nouns are treated differently in UCCA, but similarly in UD.
2. UCCA's distinction between primary relations, secondary relations and Participants, in contrast to UD's core/non-core distinction.

Proceedings of NAACL-HLT 2019, pages 478–488
Minneapolis, Minnesota, June 2 - June 7, 2019.

3. Different treatment of multi-word expressions (MWEs), where UCCA has a stronger tendency to explicitly mark them.

4. UCCA's conflation of several syntactic realizations of inter-clause linkage, and disambiguation of other cases that UD treats similarly.

We show that the differences between the schemes are substantial, and suggest that UCCA parsing in particular and semantic parsing in general are likely to benefit downstream text understanding applications. For example, only 72.9% of UCCA Participants are UD syntactic arguments, i.e., many semantic participants cannot be recovered from UD.[1] Our findings are relevant to other semantic representations, given their significant overlap in content (Abend and Rappoport, 2017).

A methodology for comparing syntactic and semantic treebanks can also support fine-grained error analysis of semantic parsers, as illustrated by Szubert et al. (2018) for AMR (Banarescu et al., 2013). To demonstrate the utility of our comparison methodology, we perform fine-grained error analysis on UCCA parsing, according to UD relations (§6). Results highlight challenges for current parsing technology, and expose cases where UCCA parsers may benefit from modeling syntactic structure more directly.[2]

2 Representations

The conceptual and formal similarity between UD and UCCA can be traced back to their shared design principles: both are designed to be applicable across languages and domains, to enable rapid annotation and to support text understanding applications. This section provides a brief introduction to each of the schemes, whereas the next sections discuss their content in further detail.[3]

UCCA is a semantic annotation scheme rooted in typological and cognitive linguistic theory. It aims to represent the main semantic phenomena in text, abstracting away from syntactic forms. Shown to be preserved remarkably well across translations (Sulem et al., 2015), it has been applied to improve text simplification (Sulem et al., 2018b), and text-to-text generation evaluation (Birch et al., 2016; Choshen and Abend, 2018; Sulem et al., 2018a).

Participant	A	Linker	L
Center	C	Connector	N
Adverbial	D	Process	P
Elaborator	E	Quantifier	Q
Function	F	Relator	R
Ground	G	State	S
Parallel Scene	H	Time	T

Table 1: Legend of UCCA categories (edge labels).

Formally, UCCA structures are directed acyclic graphs (DAGs) whose nodes (or *units*) correspond either to words, or to elements viewed as a single entity according to some semantic or cognitive consideration. Edges are labeled, indicating the role of a child in the relation the parent represents. Figure 1 shows a legend of UCCA abbreviations. A *Scene* is UCCA's notion of an event or a frame, and is a description of a movement, an action or a state which persists in time. Every Scene contains one primary relation, which can be either a Process or a State. Scenes may contain any number of Participants, a category which also includes abstract participants and locations. They may also contain temporal relations (Time), and secondary relations (Adverbials), which cover semantic distinctions such as manner, modality and aspect.[4]

Scenes may be *linked* to one another in several ways. First, a Scene can provide information about some entity, in which case it is marked as an Elaborator. This often occurs in the case of participles or relative clauses. For example, "(child) who went to school" is an Elaborator Scene in "The child who went to school is John". A Scene may also be a Participant in another Scene. For example, "John went to school" in the sentence: "He said John went to school". In other cases, Scenes are annotated as Parallel Scenes (H), which are flat structures and may include a Linker (L), as in: "When$_L$ [he arrives]$_H$, [he will call them]$_H$".

Non-Scene units are headed by units of the category Center, denoting the type of entity or thing described by the whole unit. Elements in non-Scene units include Quantifiers (such as "*dozens* of people") and Connectors (mostly coordinating conjunctions). Other modifiers to the Center are marked as Elaborators.

UCCA distinguishes *primary* edges, corresponding to explicit relations, from *remote* edges,

---

[1] This excludes cases of shared argumenthood, which are partially covered by *enhanced UD*. See §4.1.

[2] Our conversion and analysis code is public available at `https://github.com/danielhers/synsem`.

[3] See Supplementary Material for a definition of each category in both schemes, and their abbreviations.

[4] Despite the similar terminology, UCCA Adverbials are not necessarily adverbs syntactically.

which allow for a unit to participate in several super-ordinate relations. See example in Figure 1. Primary edges form a tree, whereas remote edges (dashed) enable reentrancy, forming a DAG.

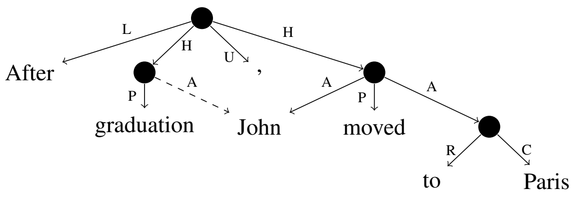


Figure 1: UCCA graph. Dashed: remote edge.

UD is a syntactic dependency scheme used in many languages, aiming for cross-linguistically consistent and coarse-grained treebank annotation. Formally, UD uses bi-lexical trees, with edge labels representing syntactic relations.

One aspect of UD similar to UCCA is its preference of lexical (rather than functional) heads. For example, in auxiliary verb constructions (e.g., “is eating”), UD marks the lexical verb (*eating*) as the head, while other dependency schemes may select the auxiliary *is* instead. While the approaches are largely inter-translatable (Schwartz et al., 2012), lexical head schemes are more similar in form to semantic schemes, such as UCCA and semantic dependencies (Oepen et al., 2016).

Being a dependency representation, UD is structurally underspecified in an important way: it is not possible in UD to mark the distinction between an element modifying the head of the phrase and the same element modifying the whole phrase (de Marneffe and Nivre, 2019).

An example UD tree is given in Figure 2. UD relations will be written in `typewriter` font.

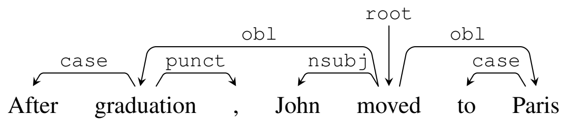


Figure 2: UD tree.

3 Shared Gold-standard Corpus

We annotate 723 English passages (3,813 sentences; 52,721 tokens), comprising the web reviews section of the English Web Treebank (EWT; Bies et al., 2012). Text is annotated by two UCCA annotators according to v2.0 of the UCCA guidelines[5] and cross-reviewed. As these sentences are included in the UD English_EWT treebank, this is a *shared* gold-standard UCCA and UD annotated corpus.[6] We use the standard train/development/test split, shown in Table 2.

	Train	**Dev**	**Test**
# Passages	347	192	184
# Sentences	2,723	554	535
# Tokens	44,804	5,394	5,381

Table 2: Data split for the shared gold-standard corpus.

4 Comparison Methodology

To facilitate comparison between UCCA and UD, we first assimilate the graphs by abstracting away from formalism differences, obtaining a similar graph format for both schemes. We then match pairs of nodes in the converted UD and UCCA trees if they share all terminals in their yields.

UD annotates bi-lexical dependency trees, while UCCA graphs contain non-terminal nodes. In §4.1, we outline the unified DAG converter by Hershcovich et al. (2018a,b),[7] which we use to reach a common format. In §4.2, we describe a number of extensions to the converter, which abstract away from further non-content differences.

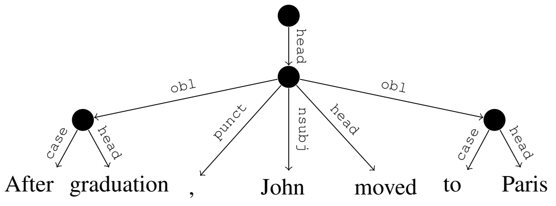


Figure 3: Converted UD tree. Non-terminals and *head* edges are introduced by the unified DAG converter.

4.1 Basic Conversion

Figure 3 presents the same tree from Figure 2 after conversion. The converter adds one pre-terminal per token, and attaches them according to the original dependency tree: traversing it from the root, for each head it creates a non-terminal parent with the edge label *head*, and adds the dependents as children of the created non-terminal. Relation subtypes are stripped, leaving only universal relations. For example, the language-specific definite article label `det:def` is replaced by the universal `det`.

---

[5] `http://bit.ly/ucca_guidelines_v2`

[6] Our data is available at `https://github.com/UniversalConceptualCognitiveAnnotation/UCCA_English-EWT`.

[7] `https://github.com/huji-nlp/semstr`

Reentrancies. Remote edges in UCCA enable reentrancy, forming a DAG together with primary edges. UD allows reentrancy when including *enhanced dependencies* (Schuster and Manning, 2016),[8] which form (bi-lexical) graphs, representing phenomena such as predicate ellipsis (e.g., gapping), and shared arguments due to coordination, control, raising and relative clauses.

UCCA is more inclusive in its use of remote edges, and accounts for the entire class of implicit arguments termed *Constructional Null Instantiation* in FrameNet (Ruppenhofer et al., 2016). For example, in "The Pentagon is bypassing official US intelligence channels [...] in order to create strife" (from EWT), remote edges mark *Pentagon* as a shared argument of *bypassing* and *create*. Another example is "if you call for an appointment [...] so you can then make one", where a remote edge in UCCA indicates that *one* refers to *appointment*. Neither is covered by enhanced UD.

In order to facilitate comparison, we remove remote edges and enhanced dependencies in the conversion process. We thus compare basic UD and UCCA trees, deferring a comparison of UCCA and enhanced UD to future work.

4.2 Extensions to the Converter

We extend the unified DAG converter to remove further non-content differences.

Unanalyzable units. An unanalyzable phrase is represented in UCCA as a single unit covering multiple terminals. In multi-word expressions (MWEs) in UD, each word after the first is attached to the previous word, with the `flat`, `fixed` or `goeswith` relations (depending on whether the expression is grammaticalized, or split by error). We remove edges of these relations and join the corresponding pre-terminals to one unit.

Promotion of conjunctions. The basic conversion generally preserves terminal yields: the set of terminals spanned by a non-terminal is the same as the original dependency yield of its head terminal (e.g., in Figure 3, the yield of the non-terminal headed by *graduation* is "After graduation", the same as that of "graduation" in Figure 2).

Since UD attaches subordinating and coordinating conjunctions to the subsequent conjunct, this results in them being positioned in the same conjunct they relate (e.g., *After* will be included in the first conjunct in "After arriving home, John went to sleep"; *and* will be included in the second conjunct in "John and Mary"). In contrast, UCCA places conjunctions as siblings to their conjuncts (e.g., "[After] [arriving home], [John went to sleep]" and "[John] [and] [Mary]").

To abstract away from these convention differences, we place coordinating and subordinating conjunctions (i.e., `cc`-labeled units, and `mark`-labeled units with an `advcl` head such as *when*, *if*, *after*) as siblings of their conjuncts.

5 Analysis of Divergences

Using the shared format, we turn to analyzing the content differences between UCCA and UD.[9]

5.1 Confusion Matrix

Table 3 presents the confusion matrix of categories between the converted UD and UCCA, calculated over all sentences in the training and development sets of the shared EWT reviews corpus. We leave the test set out of this evaluation to avoid contamination for future parsing experiments.

In case of multiple UCCA units with the same terminal yield (i.e., units with a single non-remote child), we take the top category only, to avoid double-counting. Excluding punctuation, this results in 60,434 yields in UCCA and 58,992 in UD. Of these, 52,280 are common, meaning that a UCCA "parser" developed this way would get a very high F1 score of 87.6%, if it is provided with the gold UCCA label for every converted edge.

Some yields still have more than one UCCA category associated with them, due to edges with multiple categories (**A**|**P** and **A**|**S**). For presentation reasons, 0.15% of the UCCA units in the data are not presented here, as they belong to rare ($<0.1\%$) multiple-category combinations.

Only 82.6% of UD's syntactic arguments (`ccomp`, `csubj`, `iobj`, `nsubj`, `obj`, `obl` and `xcomp`) are UCCA Participants, and only 72.9% of the Participants are syntactic arguments—a difference stemming from the Scene/non-Scene (§5.2) and argument/adjunct (§5.3) distinctions. Moreover, if we identify predicates as words having at least one argument and Scenes as units with at least one Participant, then only 92.1% of UD's predicates correspond to Scenes (many are secondary relations within one scene), and only 80%

---

[8] `https://universaldependencies.org/u/overview/enhanced-syntax.html`

[9] See `http://bit.ly/uccaud` for a detailed explanation of each example in this section.

	A	A|P	A|S	C	D	E	F	G	H	L	N	P	Q	R	S	T	No Match
`acl`	58			1	4	249	1		48			6			1	1	409
`advcl`	14			12	2	2		6	512	4		11					423
`advmod`	225		1	69	1778	332	27	135	14	258	2	2	15	44	9	368	273
`amod`	25			134	647	837		1	28			7	130	3	269	25	176
`appos`	21			39	2	34			18						8		33
`aux`					384	2	1335			2		1		1			17
`case`	11			31	27	25	123			213	26	11	1	2629	154	1	262
`cc`				8	4	1	4	1	1	1567	381		6	12			52
`ccomp`	345			1		1			36			2			1	1	166
`compound`	225			116	67	586	21		2			32	19	1	12	24	683
`conj`	10			449	4	5		1	1262	1		6	2		10		497
`cop`				1			1312			1		9		10	178		7
`csubj`	13								3								46
`det`	10			17	119	440	2963				1		129	16	1		124
`discourse`	1			2	1		25	29	27	16					5		19
`expl`	21			1			98								17		3
`iobj`	131			1			1										10
`list`	3			7	2	1			27						1		6
`mark`				9	7	1	531	1		654				407	1	5	143
`nmod`	844	1	1	20	9	786	8	4	12	1	1	20	2	2	11	27	488
`nsubj`	4296	7	21	25	3	2	55	1	5	61		58	1	80	14	4	247
`nummod`	2			33	12	17		4		4			334				64
`obj`	1845		1	54	21	6	11	1	4	23		52	1	23	3	11	583
`obl`	1195			19	115	41	1	17	39	34		6	6	26	7	302	611
`parataxis`	6		1	5		4		6	285						3		180
`vocative`	17							8									
`xcomp`	121			4	25				8			38			38		526
head	445	48	159	6388	717	142	564	83	2462	42	1	4163	120	52	1547	32	2235
No Match	1421	37	58	640	417	291	14	33	2291	146	6	802	94	52	369	96	

Table 3: UD-UCCA confusion matrix calculated based on EWT gold-standard annotations from the training and development sets (§3), after applying our extended converter to UD (§4), by matching UD vertices and UCCA units with the same terminal yield. The last column (row), labeled No Match, shows the number of edges of each UD (UCCA) category that do not match any UCCA (UD) unit. Zero counts are omitted.

of Scenes correspond to predicates (e.g., eventive nouns, which are not syntactic predicates).

Examining the *head* row in Table 3 allows us to contrast the schemes' notions of a head. *head*-labeled units have at least one dependent in UD, or are single-clause sentences (technically, they are non-terminals added by the converter). Of them, 75.7% correspond to Processes, States, Parallel Scenes or Centers, which are UCCA's notions of semantic heads, and 11.6% are left unmatched, mostly due to MWEs analyzed in UD but not in UCCA (§5.4). Another source of unmatched units is inter-Scene linkage, which tends to be flatter in UCCA (§5.5). The rest are mostly due to head swap (e.g., "*all* of Dallas", where *all* is a Quantifier of *Dallas* in UCCA, but the head in UD).

In the following subsections, we review the main content differences between the schemes, as reflected in the confusion matrix, and categorize them according to the UD relations involved.

5.2 Scenes vs. Non-Scenes

UCCA distinguishes between Scenes and non-Scenes. This distinction crosses UD categories, as a Scene can be evoked by a verb, an eventive or stative noun (*negotiation*, *fatigue*), an adjective or even a preposition ("this is *for* John").

Core syntactic arguments. Subjects and objects are usually Participants (e.g., "*wine* was excellent"). However, when describing a Scene, the subject may be a Process/State (e.g., "but *service* is very poor"). Some wh-pronouns are the subjects or objects of a relative clause, but are Linkers or Relators, depending on whether they link Scenes or non-Scenes, respectively. For example, "who" in "overall, Joe is a happy camper *who* has found a great spot" is an `nsubj`, but a Linker. Other arguments are Adverbials or Time (see §5.3), and some do not match any UCCA unit, especially when they are parts of MWEs (see §5.4).

Adjectival modifiers are Adverbials when modifying Scenes ("*romantic* dinner"), States when describing non-Scenes ("*beautiful* hotel") or when semantically predicative ("such a *convenient* location"), or Elaborators where defining inherent properties of non-Scenes ("*medical* school").

Nominal and clausal modifiers. Most are Participants or Elaborators, depending on whether they modify a Scene (e.g., "discount *on services*" and "our decision *to buy when we did*" are Participants, but "*my car's* gears and brakes" and "Some of the younger kids *that work there*" are Elaborators). Unmatched `acl` are often free relative clauses (e.g., in "the prices were worth what *I got*", *what* is the `obj` of *worth* but a Participant of *I got*).

Case markers. While mostly Relators modifying non-Scenes (e.g., "the team *at* Bradley

Chevron”), some case markers are Linkers linking Scenes together (e.g., “very informative website *with* a lot of good work”). Others are Elaborators (e.g., “*over* a year”) or States when used as the main relation in verbless or copula clauses (e.g., “it is right *on* Wisconsin Ave”).

Coordination. Coordinating conjunctions (`cc`) are Connectors where they coordinate non-Scenes (e.g., “Mercedes *and* Dan”) or Linkers where they coordinate Scenes (e.g., “outdated *but* not bad”). Similarly, conjuncts and list elements (`conj`, `list`) may be Parallel Scenes (H), or Centers when they are non-Scenes.[10]

Determiners. Articles are Functions, but determiners modifying non-Scenes are Elaborators (e.g., “I will never recommend this gym to *any* woman”). Where modifying Scenes (mostly negation) they are marked as Adverbials. For example, “*no* feathers in stock”, “*what* a mistake”, and “the rear window had *some* leakage” are all Adverbials.

5.3 Primary and Secondary Relations

UD distinguishes core arguments, adverb modifiers, and obliques (in English UD, the latter mostly correspond to prepositional dependents of verbs). UCCA distinguishes Participants, including locations and abstract entities, from secondary relations (Adverbials), which cover manner, aspect and modality. Adverbials can be verbs (e.g., *begin*, *fail*), prepositional phrases (*with disrespect*), as well as modals, adjectives and adverbs.

Adverbs and obliques. Most UD adverb modifiers are Adverbials (e.g., “I *sometimes* go”), but they may be Participants, mostly in the case of semantic arguments describing location (e.g., *here*). Obliques may be Participants (e.g., “wait *for Nick*”), Time (e.g., “for *over 7 years*”) or Adverbials—mostly manner adjuncts (*by far*).

Clausal arguments are Participant Scenes (e.g., “it was great *that they did not charge a service fee*”, “did not really know *what I wanted*” or “I asked them *to change it*”). However, when serving as complements to a secondary verb, they will not match any unit in UCCA, as it places secondary verbs on the same level as their primary relation. For example, *to pay* is an `xcomp` in “they have to pay”, while the UCCA structure is flat: *have to* is an Adverbial and *pay* is a Process. Single-worded clausal arguments may correspond to a Process/State, as in “this seems *great*”.

Auxiliary verbs are Functions (e.g., “*do* not forget”), or Adverbials when they are modals (e.g., “you *can* graduate”). Semi-modals in UD are treated as clausal heads, which take a clausal complement. For example, in “able to do well”, UD treats *able* as the head, which takes *do well* as an `xcomp`. UCCA, on the other hand, treats it as an Adverbial, creating a mismatch for `xcomp`.

5.4 Multi-Word Expressions

UD and UCCA treat MWEs differently. In UD they include names, compounds and grammaticalized fixed expressions. UCCA treats names and grammaticalized MWEs as unanalyzable units, but also a range of semantically opaque constructions (e.g., light verbs and idioms). On the other hand, compounds are not necessarily unanalyzable in UCCA, especially if compositional.

Compounds. English compounds are mostly nominal, and are a very heterogeneous category. Most compounds correspond to Elaborators (e.g., “*industry* standard”), or Elaborator Scenes (e.g., “*out-of-place* flat-screen TV”), and many are unanalyzable expressions (e.g., “*mark* up”). Where the head noun evokes a Scene, the dependent is often a Participant (e.g., “*food* craving”), but can also be an Adverbial (e.g., “*first time* buyers”) depending on its semantic category. Other compounds in UD are phrasal verbs (e.g., “figure *out*”, “cleaned *up*”), which UCCA treats as unanalyzable (leading to unmatched units).

Core arguments. A significant number of subjects and objects are left unmatched as they form parts of MWEs marked in UCCA as unanalyzable. UD annotates MWEs involving a verb and its argument(s) just like any other clause, and therefore lacks this semantic content. Examples include light verbs (e.g., “give *a try*”), idioms (“bites *the dust*”), and figures of speech (e.g., “when *it* comes to”, “offer *a taste* (of)”), all are UCCA units.

Complex prepositions. Some complex prepositions (e.g., *according to* or *on top of*), not encoded as MWEs in UD, are unanalyzable in UCCA.

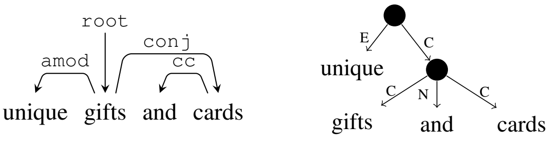
5.5 Linkage

Head selection. UCCA tends to flatten linkage, where UD, as a dependency scheme, selects a head

---

[10] While in UD the conjunction `cc` is attached to the following conjunct, in UCCA coordination is a flat structure. This is a convention difference that we normalize (§4.2).

and dependent per relation. This yields scope ambiguities for coordination, an inherently flat structure. For instance, "unique gifts and cards" is ambiguous in UD as to whether *unique* applies only to *gifts* or to the whole phrase—both annotated as in Figure 4a. UCCA, allowing non-terminal nodes, disambiguates this case (Figure 4b).



(a) UD (b) UCCA

Figure 4: Coordination in UD and UCCA.

Clausal dependents. UD categorizes clause linkage into coordination, subordination, argumenthood (complementation), and parataxis. UCCA distinguishes argumenthood but conflates the others into the Parallel Scene category. For example, "We called few companies before *we decided to hire them*" and "Check out The Willow Lounge, *you'll be happy*" are Parallel Scenes.

Note that while in UD, `mark` (e.g., *before*) is attached to the dependent adverbial clause, a UCCA Linker lies outside the linked Scenes. To reduce unmatched `advcl` instances, this convention difference is fixed by the converter (§4.2). Many remaining unmatched units are due to conjunctions we could not reliably raise. For instance, the marker *to* introducing an `xcomp` is ambiguous between Linker (purposive *to*) and Function (infinitive marker). Similarly, wh-pronouns may be Linkers ("he was willing to budge a little on the price *which* means a lot to me"), but have other uses in questions and free relative clauses. Other mismatches result from the long tail of differences in how UD and UCCA construe linkage. Consider the sentence in Figure 5. While *moment* is an oblique argument of *know* in UD, *From the moment* is analyzed as a Linker in UCCA.

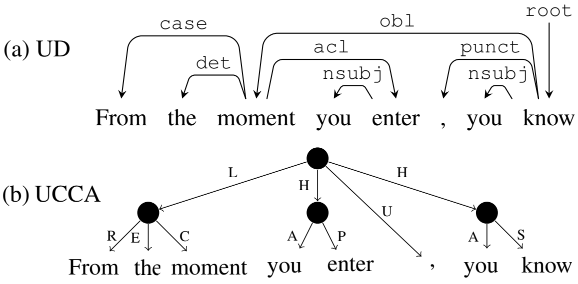


Figure 5: Clause linkage in UD and UCCA.

5.6 Other Differences

Appositions in UD always follow the modified noun, but named entities in them are UCCA Centers, regardless of position (e.g., in "its sister store Peking Garden", the UD head *its sister store* is an Elaborator, while *Peking Garden* is the Center).

Copulas. UCCA distinguishes copular constructions expressing identity (e.g., "This *is* the original Ham's restaurant") where the copula is annotated as State, and cases of attribution (e.g., "Mercedes and Dan *are* very thorough") or location (e.g., "Excellent chefs *are* in the kitchen"), where the copula is a Function.

Discourse markers and interjections. Units relating a Scene to the speech event or to the speaker's opinion are Ground (e.g., "*no*, Warwick in New Jersey" and "*Please* visit my website"). On the other hand, discourse elements that relate one Scene to another are Linkers (e.g., *anyway*).

Vocatives are both Ground and Participants if they participate in the Scene *and* are the party addressed. For example, *Mark* in "Thanks *Mark*" is both the person addressed and the one thanked.[11]

Expletives and subjects. Expletives are generally Functions, but some instances of *it* and *that* are analyzed as `nsubj` in UD and as Function in UCCA (e.g., "*it*'s like driving a new car").

Excluded relations. We exclude the following UD labels, as they are irrelevant to our evaluation: `root` (always matches the entire sentence); `punct` (punctuation is ignored in UCCA evaluation); `dep` (unspecified dependency), `orphan` (used for gapping, which is represented using remote edges in UCCA—see §4.1); `fixed`, `flat` and `goeswith` (correspond to parts of unanalyzable units in UCCA, and so do not represent units on their own—see §4.2); `reparandum` and `dislocated` (too rare in EWT).

6 Fine-Grained UCCA Parsing Evaluation

In §5 we used our comparison methodology, consisting of the conversion to a shared format and matching units by terminal yield, to compare gold-standard UD and UCCA. In this section we ap-

---

[11] The **A**|**G** column is omitted from Table 3 as this category combination occurs in only 0.02% of edges in the corpus.

ply the same methodology to parser outputs, using gold-standard UD for fine-grained evaluation.

6.1 Experimental Setup

Data. In addition to the UCCA EWT data (§3), we use the reviews section of the UD v2.3 English_EWT treebank (Nivre et al., 2018),[12] annotated over the exact same sentences. We additionally use UDPipe v1.2 (Straka et al., 2016; Straka and Straková, 2017), trained on English_EWT,[13] for feature extraction. We apply the extended converter to UD as before (§4.2).

Parser. We train TUPA v1.3 (Hershcovich et al., 2017, 2018a) on the UCCA EWT data, with the standard train/development/test split. TUPA uses POS tags and syntactic dependencies as features. We experiment both with using gold UD for feature extraction, and with using UDPipe outputs.

Evaluation by gold-standard UD. UCCA evaluation is generally carried out by considering a predicted unit as correct if there is a gold unit that matches it in terminal yield and labels. Precision, Recall and F-score (F1) are computed accordingly. For the fine-grained analysis, we split the gold-standard, predicted and matched UCCA units according to the labels of the UD relations whose dependents have the same terminal yield (if any).

6.2 Results

Table 4 presents TUPA's scores on the UCCA EWT development and test sets. Surprisingly, using UDPipe for feature extraction results in better scores than gold syntactic tags and dependencies.

	Primary			Remote		
Features	**LP**	**LR**	**LF**	**LP**	**LR**	**LF**
Development						
Gold UD	72.1	71.2	71.7	61.2	38.1	47.0
UDPipe	73.0	72.1	72.5	53.7	40.8	46.4
Test						
Gold UD	72.2	71.2	71.7	60.9	36.8	45.9
UDPipe	72.4	71.7	72.1	60.3	38.5	47.0

Table 4: Labeled precision, recall and F1 (in %) for primary and remote edges output by TUPA on the UCCA EWT development (top) and test (bottom) sets, using either gold-standard UD or UDPipe for TUPA features.

Table 5 shows fine-grained evaluation by UD relations. TUPA does best on auxiliaries and determiners, despite the heterogeneity of corresponding UCCA categories (see Table 3), possibly by making lexical distinctions (e.g., modals and auxiliary verbs are both UD auxiliaries, but are annotated as Adverbials and Functions, respectively).

Copulas and coordinating conjunctions pose a more difficult distinction, since the same lexical items may have different categories depending on the context: State/Function for copulas, due to the distinction between identity and attribution, and Connector/Linker for conjunctions, due to the distinction between Scenes and non-Scenes. However, the reviews domain imposes a strong prior for both (Function and Linker, respectively), which TUPA learns successfully.

Inter-clause linkage (`conj`, `advcl`, `xcomp`, `ccomp`, `parataxis`, `acl` and `csubj`) is a common source of error for TUPA. Although the match between UCCA and UD is not perfect in these cases, it is overall better than TUPA's unlabeled performance, despite using gold-standard syntactic features. Our results thus suggest that encoding syntax more directly, perhaps using syntactic scaffolding (Swayamdipta et al., 2018) or guided attention (Strubell et al., 2018), may assist in predicting unit boundaries. However, TUPA often succeeds at making distinctions that are not even encoded in UD. For example, it does reasonably well (71%) on distinguishing between noun modifiers of Scene-evoking nouns (Participants) and modifiers of other nouns (Elaborators), surpassing a majority baseline based on the UD relation (51%). Lexical resources that distinguish eventive and relational nouns from concrete nouns may allow improving it even further. In the similar case of compounds, lexical resources for light verbs and idioms may increase performance.

7 Discussion

NLP tasks often require semantic distinctions that are difficult to extract from syntactic representations. Consider the example "after graduation, John moved to Paris" again. While *graduation* evokes a Scene (Figure 1), in UD it is an oblique modifier of *moved*, just like *Paris* is (Figure 2). The Scene/non-Scene distinction (§5.2) would assist structural text simplification systems in paraphrasing this sentence to two sentences, each one containing one Scene (Sulem et al., 2018a).

Another example is machine translation—translating the same sentence into Hebrew, which does not have a word for *graduation*, would re-

---

[12] `https://hdl.handle.net/11234/1-2895`

[13] `https://hdl.handle.net/11234/1-2898`

		`aux`	`det`	`cop`	`cc`	`expl`	`iobj`	`nsubj`	`case`	`list`	`advmod`	`amod`	`nummod`	`mark`	`vocative`	`compound`	`obj`	`nmod`	`conj`	`advcl`	`obl`	`xcomp`	`discourse`	`ccomp`	`parataxis`	`appos`	`acl`	`csubj`
(a)	**Labeled F1 %**	94	93	89	86	83	83	80	76	76	72	71	71	70	62	59	57	55	50	49	48	41	38	29	23	21	20	0
	Unlabeled F1 %	99	99	100	99	100	83	84	95	76	95	95	86	97	92	84	65	77	61	51	61	63	95	29	36	48	37	33
(b)	**Total in UD #**	156	392	187	212	12	8	463	335	15	378	374	38	116	1	219	222	231	244	52	208	1	16	29	52	22	81	5
	Match Gold #	156	385	187	206	12	6	468	305	12	359	361	33	111	7	146	187	198	210	40	162	28	10	20	48	17	56	4
	Match Predicted #	154	388	187	203	12	6	446	313	9	345	339	32	113	6	136	163	183	177	30	147	26	11	15	30	12	36	2
	Labeled Correct #	145	361	166	175	10	5	365	236	8	253	248	23	78	4	83	99	104	96	17	74	11	4	5	9	3	9	0
	Unlabeled Correct #	154	381	187	203	12	5	386	293	8	336	334	28	109	6	118	113	147	119	18	94	17	10	5	14	7	17	1
(c)	**Labeled/Unlabeled %**	94	95	89	86	83	100	95	81	100	75	74	82	72	67	70	88	71	81	94	79	65	40	100	64	43	53	0
	Mode/Match Gold %	79	82	86	75	58	100	91	79	83	51	35	85	45	71	54	91	51	70	92	68	44	30	94	98	41	72	100
(d)	**Average Words #**	1.0	1.0	1.0	1.0	1.0	1.1	1.6	1.0	2.2	1.2	1.2	1.1	1.0	1.6	1.2	3.0	2.4	5.8	6.6	3.8	6.0	1.1	9.0	6.7	4.0	5.6	7.5

Table 5: Fine-grained evaluation of TUPA (with gold-standard UD features) on the EWT development set. (a) Columns are sorted by labeled F1, measuring performance on each subset of edges. Unlabeled F1 ignores edge categories, evaluating unit boundaries only. (b) Total number of instances of each UD relation; of them, matching UCCA units in gold-standard and in TUPA's predictions; their intersection, with/without regard to categories. (c) Percentage of correctly categorized edges; for comparison, percentage of most frequent category (see Table 3). (d) Average number of words in corresponding terminal yields.

quire a clause to convey the same meaning. The mapping would therefore be more direct using a semantic representation, and we would benefit from breaking the utterance into two Scenes.

8 Related Work

The use of syntactic parsing as a proxy for semantic structure has a long tradition in NLP. Indeed, semantic parsers have leveraged syntax for output space pruning (Xue and Palmer, 2004), syntactic features (Gildea and Jurafsky, 2002; Hershcovich et al., 2017), joint modeling (Surdeanu et al., 2008; Hajič et al., 2009), and multi-task learning (Swayamdipta et al., 2016, 2018; Hershcovich et al., 2018a). Empirical comparison between syntactic and semantic schemes, however, is still scarce. Rudinger and Van Durme (2014) mapped Stanford Dependencies (precursor to UD) to Hobbsian Logical Form, identifying semantic gaps in the former. PredPatt (White et al., 2016), a framework for extracting predicate-argument structures from UD, was evaluated by Zhang et al. (2017) on a large set of converted PropBank annotations. Szubert et al. (2018) proposed a method for aligning AMR and UD subgraphs, finding that 97% of AMR edges are evoked by one or more words or syntactic relations. Damonte et al. (2017) refined AMR evaluation by UD labels, similar to our fine-grained evaluation of UCCA parsing.

Some syntactic representation approaches, notably CCG (Steedman, 2000), directly reflect the underlying semantics, and have been used to transduce semantic forms using rule-based systems (Basile et al., 2012). A related line of work tackles the transduction of syntactic structures into semantic ones. Reddy et al. (2016) proposed a rule-based method for converting UD to logical forms. Stanovsky et al. (2016) converted Stanford dependency trees into proposition structures (PROPS), abstracting away from some syntactic detail.

9 Conclusion

We evaluated the similarities and divergences in the content encoded by UD and UCCA. We annotated the reviews section of the English Web Treebank with UCCA, and used an automated methodology to evaluate how well the two schemes align, abstracting away from differences of mere convention. We provided a detailed picture of the content differences between the schemes. Notably, we quantified the differences between the notions of syntactic and semantic heads and arguments, finding substantial divergence between them. Our findings highlight the potential utility of using semantic parsers for text understanding applications (over their syntactic counterparts), but also expose challenges semantic parsers must address, and potential approaches for addressing them.

Acknowledgments

This work was supported by the Israel Science Foundation (grant No. 929/17), and by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. We thank Jakob Prange, Nathan Schneider and the anonymous reviewers for their helpful comments.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*, pages 77–89.

Joakim Nivre et al. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the Linguistic Annotation Workshop*.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proc. of LREC*, pages 3196–3200.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank. *Linguistic Data Consortium, Philadelphia, PA*.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP*, pages 1264–1274.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proc. of COLING*, pages 2753–2765.

Leshem Choshen and Omri Abend. 2018. Reference-less measure of faithfulness for grammatical error correction. In *Proc. of NAACL-HLT*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proc. of EACL*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL*, pages 1–18.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*, pages 2061–2071.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018a. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018b. Universal dependency parsing with a general transition-based DAG parser. In *Proc. of CoNLL UD Shared Task*, pages 103–112.

Marie-Catherine de Marneffe and Joakim Nivre. 2019. Dependency grammar. *Annual Review of Linguistics*, 5(1):197–218.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *TACL*, 4:127–141.

Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2015. Because syntax does matter: Improving predicate-argument structures parsing with syntactic features. In *Proc. of NAACL-HLT*, pages 64–74.

Rachel Rudinger and Benjamin Van Durme. 2014. Is the Stanford dependency representation semantic? In *Proc. of EVENTS*, pages 54–58.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proc. of LREC*. ELRA.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *Proc. of COLING*, pages 2405–2422.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with PropS. *arXiv preprint arXiv:1603.01648*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proc. of LREC*, Portoro, Slovenia. European Language Resources Association.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proc. of CoNLL UD Shared Task*, pages 88–99, Vancouver, Canada.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*, pages 5027–5038.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *Proc. of S2MT*, pages 11–22.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. Semantic structural annotation for text simplification. In *Proc. of NAACL*.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Proc. of ACL*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*, pages 159–177.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*, pages 187–197.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*, pages 3772–3782.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for English-AMR alignment. In *Proc. of NAACL-HLT*, pages 1169–1180.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proc. of EMNLP*, pages 1713–1723.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An evaluation of PredPatt and open IE via stage 1 semantic role labeling. In *Proc. of IWCS*.

Attentive Mimicking:
Better Word Embeddings by Attending to Informative Contexts

Timo Schick
Sulzer GmbH
Munich, Germany
timo.schick@sulzer.de

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Learning high-quality embeddings for rare words is a hard problem because of sparse context information. Mimicking (Pinter et al., 2017) has been proposed as a solution: given embeddings learned by a standard algorithm, a model is first trained to reproduce embeddings of frequent words from their surface form and then used to compute embeddings for rare words. In this paper, we introduce *attentive mimicking*: the mimicking model is given access not only to a word's surface form, but also to all available contexts and learns to attend to the most informative and reliable contexts for computing an embedding. In an evaluation on four tasks, we show that attentive mimicking outperforms previous work for both rare and medium-frequency words. Thus, compared to previous work, attentive mimicking improves embeddings for a much larger part of the vocabulary, including the medium-frequency range.

1 Introduction

Word embeddings have led to large performance gains in natural language processing (NLP). However, embedding methods generally need many observations of a word to learn a good representation for it.

One way to overcome this limitation and improve embeddings of infrequent words is to incorporate surface-form information into learning. This can either be done directly (Wieting et al., 2016; Bojanowski et al., 2017; Salle and Villavicencio, 2018), or a two-step process is employed: first, an embedding model is trained on the word level and then, surface-form information is used either to fine-tune embeddings (Cotterell et al., 2016; Vulić et al., 2017) or to completely recompute them. The latter can be achieved using a model trained to reproduce (or *mimic*) the original embeddings (Pinter et al., 2017). However, these methods only work if a word's meaning can at least partially be predicted from its form.

A closely related line of research is embedding learning for *novel words*, where the goal is to obtain embeddings for previously unseen words from at most a handful of observations. While most contemporary approaches exclusively use context information for this task (e.g. Herbelot and Baroni, 2017; Khodak et al., 2018), Schick and Schütze (2019) recently introduced the *form-context model* and showed that joint learning from both surface form and context leads to better performance.

The problem we address in this paper is that often, only few of a word's contexts provide valuable information about its meaning. Nonetheless, the current state of the art treats all contexts the same. We address this issue by introducing a more intelligent mechanism of incorporating context into mimicking: instead of using all contexts, we learn – by way of self-attention – to pick a subset of especially informative and reliable contexts. This mechanism is based on the observation that in many cases, reliable contexts for a given word tend to resemble each other. We call our proposed architecture *attentive mimicking* (AM).

Our contributions are as follows: (i) We introduce the attentive mimicking model. It produces high-quality embeddings for rare and medium-frequency words by attending to the most informative contexts. (ii) We propose a novel evaluation method based on VecMap (Artetxe et al., 2018) that allows us to easily evaluate the embedding quality of low- and medium-frequency words. (iii) We show that attentive mimicking improves word embeddings on various datasets.

2 Related Work

Methods to train surface-form models to mimic word embeddings include those of Luong et al.

Proceedings of NAACL-HLT 2019, pages 489–494
Minneapolis, Minnesota, June 2 - June 7, 2019.

(2013) (morpheme-based) and Pinter et al. (2017) (character-level). In the area of fine-tuning methods, Cotterell et al. (2016) introduce a Gaussian graphical model that incorporates morphological information into word embeddings. Vulić et al. (2017) retrofit embeddings using a set of language-specific rules. Models that directly incorporate surface-form information into embedding learning include fastText (Bojanowski et al., 2017), LexVec (Salle and Villavicencio, 2018) and Charagram (Wieting et al., 2016).

While many approaches to learning embeddings for novel words exclusively make use of context information (Lazaridou et al., 2017; Herbelot and Baroni, 2017; Khodak et al., 2018), Schick and Schütze (2019)'s form-context model combines surface-form and context information.

Ling et al. (2015) also use attention in embedding learning, but their attention is *within* a context (picking words), not *across* contexts (picking contexts). Also, their attention is based only on word type and distance, not on the more complex factors available in our attentive mimicking model, e.g., the interaction with the word's surface form.

3 Attentive Mimicking

3.1 Form-Context Model

We briefly review the architecture of the form-context model (FCM), see Schick and Schütze (2019) for more details.

FCM requires an embedding space of dimensionality d that assigns high-quality embeddings $v \in \mathbb{R}^d$ to frequent words. Given an infrequent or novel word w and a set of contexts $\mathcal{C}$ in which it occurs, FCM can then be used to infer an embedding $v_{(w,\mathcal{C})}$ for w that is appropriate for the given embedding space. This is achieved by first computing two distinct embeddings, one of which exclusively uses surface-form information and the other context information. The surface-form embedding, denoted $v^{\text{form}}_{(w,\mathcal{C})}$, is obtained from averaging over a set of n-gram embeddings learned by the model; the context embedding $v^{\text{context}}_{(w,\mathcal{C})}$ is obtained from averaging over all embeddings of context words in $\mathcal{C}$.

The two embeddings are then combined using a weighting coefficient α and a $d \times d$ matrix A, resulting in the form-context embedding

$$v_{(w,\mathcal{C})} = \alpha \cdot A v^{\text{context}}_{(w,\mathcal{C})} + (1-\alpha) \cdot v^{\text{form}}_{(w,\mathcal{C})} \,.$$

The weighing coefficient α is a function of both embeddings, modeled as

$$\alpha = \sigma(u^\top [v^{\text{context}}_{(w,\mathcal{C})}; v^{\text{form}}_{(w,\mathcal{C})}] + b)$$

with $u \in \mathbb{R}^{2d}$, $b \in \mathbb{R}$ being learnable parameters and σ denoting the sigmoid function.

3.2 Context Attention

FCM pays equal attention to all contexts of a word but often, only few contexts are actually suitable for inferring the word's meaning. We introduce *attentive mimicking* (AM) to address this problem: we allow our model to assign different weights to contexts based on some measure of their "reliability". To this end, let $\mathcal{C} = \{C_1, \ldots, C_m\}$ where each C_i is a multiset of words. We replace the context-embedding of FCM with a weighted embedding

$$v^{\text{context}}_{(w,\mathcal{C})} = \sum_{i=1}^{m} \rho(C_i, \mathcal{C}) \cdot v_{C_i}$$

where v_{C_i} is the average of the embeddings of words in C_i and ρ measures context reliability.

To obtain a meaningful measure of reliability, our key observation is that reliable contexts typically agree with many other contexts. Consider a word w for which six out of ten contexts contain words referring to sports. Due to this high inter-context agreement, it is then reasonable to assume that w is from the same domain and, consequently, that the four contexts not related to sports are less informative. To formalize this idea, we first define the similarity between two contexts as

$$s(C_1, C_2) = \frac{(M v_{C_1}) \cdot (M v_{C_2})^\top}{\sqrt{d}}$$

with $M \in \mathbb{R}^{d \times d}$ a learnable parameter, inspired by Vaswani et al. (2017)'s scaled dot-product attention. We then define the reliability of a context as

$$\rho(C, \mathcal{C}) = \frac{1}{Z} \sum_{i=1}^{m} s(C, C_i)$$

where $Z = \sum_{i=1}^{m} \sum_{j=1}^{m} s(C_i, C_j)$ is a normalization constant, ensuring that all weights sum to one.

The model is trained by randomly sampling words w and contexts $\mathcal{C}$ from a large corpus and mimicking the original embedding of w, i.e., minimizing the squared distance between the original embedding and $v_{(w,\mathcal{C})}$.

4 Experiments

For our experiments, we follow the setup of Schick and Schütze (2019) and use the Westbury Wikipedia Corpus (WWC) (Shaoul and Westbury, 2010) for training of all embedding models. To obtain training instances $(w, \mathcal{C})$ for both FCM and AM, we sample words and contexts from the WWC based on their frequency, using only words that occur at least 100 times. We always train FCM and AM on skipgram embeddings (Mikolov et al., 2013) obtained using Gensim (Řehůřek and Sojka, 2010).

Our experimental setup differs from that of Schick and Schütze (2019) in two respects: (i) Instead of using a fixed number of contexts for $\mathcal{C}$, we randomly sample between 1 and 64 contexts and (ii) we fix the number of training epochs to 5. The rationale behind our first modification is that we want our model to produce high-quality embeddings both when we only have a few contexts available and when there is a large number of contexts to pick from. We fix the number of epochs simply because our evaluation tasks come without development sets on which it may be optimized.

To evaluate our model, we apply a novel, intrinsic evaluation method that compares embedding spaces by transforming them into a common space (§4.1). We also test our model on three word-level downstream tasks (§4.2, §4.3, §4.4) to demonstrate its versatile applicability.

4.1 VecMap

We introduce a novel evaluation method that explicitly evaluates embeddings for rare and medium-frequency words by downsampling frequent words from the WWC to a fixed number of occurrences.[1] We then compare "gold" skipgram embeddings obtained from the original corpus with embeddings learned by some model trained on the downsampled corpus. To this end, we transform the two embedding spaces into a common space using VecMap (Artetxe et al., 2018), where we provide all but the downsampled words as a mapping dictionary. Intuitively, the better a model is at inferring an embedding from few observations, the more similar its embeddings must be to the gold embeddings in this common space. We thus measure the quality of a model by computing the average cosine similarity between its embeddings and the gold embeddings.

model	**number of occurrences** 1	2	4	8	16	32	64	128
skipgram	8.7	18.2	30.9	42.3	52.3	59.5	66.7	**71.2**
fastText	**45.4**	44.3	45.7	50.0	55.9	56.7	62.6	67.7
Mimick	10.7	11.7	12.1	11.0	12.5	11.0	10.6	9.2
FCM	37.9	**45.3**	49.1	53.4	**58.3**	55.4	59.9	58.8
AM	38.0	45.1	**49.6**	**53.7**	**58.3**	55.6	60.2	58.9
FCM†	32.3	36.9	41.9	49.1	57.4	59.9	67.3	70.1
AM†	32.8	37.8	42.8	49.8	57.7	**60.5**	**67.6**	70.4

Table 1: Average cosine similarities for the VecMap evaluation, scaled by a factor of 100. †: Downsampled words were included in the training set.

model	**maximum word frequency** 10	50	100	500	1000
skipgram	–0.16	0.21	0.33	0.55	**0.66**
fastText	–0.20	0.10	0.23	0.50	0.61
Mimick	0.00	0.01	–0.03	0.40	0.56
FCM	0.21	0.37	0.37	0.55	0.63
AM	**0.27**	**0.39**	**0.40**	**0.56**	0.64

Table 2: Spearman's ρ for various approaches on SemEval2015 Task 10E

As baselines, we train skipgram and fastText on the downsampled corpus. We then train Mimick (Pinter et al., 2017) as well as both FCM and AM on the skipgram embeddings. We also try a variant where the downsampled words are included in the training set (i.e., the mimicking models explicitly learn to reproduce their skipgram embeddings). This allows the model to learn representations of those words not completely from scratch, but to also make use of their original embeddings. Accordingly, we expect this variant to only be helpful if a word is not too rare, i.e. its original embedding is already of decent quality. Table 1 shows that for words with a frequency below 32, FCM and AM infer much better embeddings than all baselines. The comparably poor performance of Mimick is consistent with the observation of Pinter et al. (2017) that this method captures mostly syntactic information. Given four or more contexts, AM leads to consistent improvements over FCM. The variants that include downsampled words during training (†) still outperform skipgram for 32 and more observations, but perform worse than the default models for less frequent words.

4.2 Sentiment Dictionary

We follow the experimental setup of Rothe et al. (2016) and fuse Opinion lexicon (Hu and Liu,

---

[1] The VecMap dataset is publicly available at `https://github.com/timoschick/form-context-model`

	$f=1$		$f\in[2,4)$		$f\in[4,8)$		$f\in[8,16)$		$f\in[16,32)$		$f\in[32,64)$		$f\in[1,100]$	
model	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1
skipgram	0.0	2.6	2.2	7.8	11.5	30.7	44.7	64.5	37.8	59.4	**35.0**	**59.7**	33.5	58.3
fastText	44.6	51.1	50.5	65.1	48.4	62.9	44.3	59.6	34.1	53.5	29.8	55.7	31.4	56.4
Mimick	0.0	0.0	0.0	0.0	0.0	0.0	1.0	4.0	1.0	1.0	3.9	14.4	4.2	14.8
FCM	86.5	88.9	76.9	85.1	**72.0**	**81.8**	57.7	68.5	36.0	54.2	27.7	52.5	30.1	53.4
AM	**87.8**	**90.7**	**79.1**	**86.5**	**72.0**	80.9	59.5	**70.9**	37.8	56.1	28.9	53.4	31.1	54.5
AM+skip	**87.8**	**90.7**	**79.1**	**86.5**	**72.0**	81.6	**60.1**	**70.9**	**40.7**	**59.9**	**35.0**	**59.7**	**36.8**	**60.5**

Table 3: Results on the Name Typing dataset for various word frequencies f. The model that uses a linear combination of AM embeddings with skipgram is denoted AM+skip.

2004) and the NRC Emotion lexicons (Mohammad and Turney, 2013) to obtain a training set of words with binary sentiment labels. On that data, we train a logistic regression model to classify words based on their embeddings. For our evaluation, we then use SemEval2015 Task 10E where words are assigned a sentiment rating between 0 (completely negative) and 1 (completely positive) and use Spearman's ρ as a measure of similarity between gold and predicted ratings.

We train logistic regression models on both skipgram and fastText embeddings and, for testing, replace skipgram embeddings by embeddings inferred from the mimicking models. Table 2 shows that for rare and medium-frequency words, AM again outperforms all other models.

4.3 Name Typing

We use Yaghoobzadeh et al. (2018)'s name typing dataset for the task of predicting the fine-grained named entity types of a word, e.g., PRESIDENT and LOCATION for "Washington". We train a logistic regression model using the same setup as in §4.2 and evaluate on all words from the test set that occur ≤100 times in WWC. Based on results in §4.1, where AM only improved representations for words occurring fewer than 32 times, we also try the variant *AM+skip* that, in testing, replaces $v_{(w,\mathcal{C})}$ with the linear combination

$$\hat{v}_w = \beta(f_w) \cdot v_{(w,\mathcal{C})} + (1 - \beta(f_w)) \cdot v_w$$

where v_w is the skipgram embedding of w, f_w is the frequency of w and $\beta(f_w)$ scales linearly from 1 for $f_w = 0$ to 0 for $f_w = 32$.

Table 3 gives accuracy and micro F1 for several word frequency ranges. In accordance with results from previous experiments, AM performs drastically better than the baselines for up to 16 occurrences. Notably, the linear combination of skipgram and AM achieves by far the best overall results.

4.4 Chimeras

The Chimeras (CHIMERA) dataset (Lazaridou et al., 2017) consists of similarity scores for pairs of made-up words and regular words. CHIMERA provides only six contexts for each made-up word, so it is not ideal for evaluating our model. Nonetheless, we can still use it to analyze the difference of FCM (no attention) and AM (using attention). As the surface-form of the made-up words was constructed randomly and thus carries no meaning at all, we restrict ourselves to the context parts of FCM and AM (referred to as FCM-ctx and AM-ctx). We use the test set of Herbelot and Baroni (2017) and compare the given similarity scores with the cosine similarities of the corresponding word embeddings, using FCM-ctx and AM-ctx to obtain embeddings for the made-up words. Table 4 gives Spearman's ρ for our model and various baselines; baseline results are adopted from Khodak et al. (2018). We do not report results for Mimick as its representations for novel words are entirely based on their surface form. While AM performs worse than previous methods for 2–4 sentences, it drastically improves over the best result currently published for 6 sentences. Again, context attention consistently improves results: AM-ctx performs better than FCM-ctx, regardless of the number of contexts. Since A La Carte (Khodak et al., 2018), the method performing best for 2–4 contexts, is conceptually similar to FCM, it most likely would similarly benefit from context attention.

While the effect of context attention is more pronounced when there are many contexts available, we still perform a quantitative analysis of one exemplary instance of CHIMERA to better understand what AM learns; we consider the made-up word "petfel", a combination of "saxophone" and "harmonica", whose occurrences are shown in Table 5. The model attends most to sentences

model	2 sent.	4 sent.	6 sent.
skipgram	0.146	0.246	0.250
additive	**0.363**	0.370	0.360
additive – sw	0.338	0.362	0.408
Nonce2Vec	0.332	0.367	0.389
A La Carte	**0.363**	**0.384**	0.394
FCM-ctx	0.337	0.359	0.422
AM-ctx	0.342	0.376	**0.436**

Table 4: Spearman's ρ for the Chimeras task given 2, 4 and 6 context sentences for the made-up word

sentence	ρ
• i doubt if we ll ever hear a man play a **petfel** like that again	0.19
• also there were some other assorted instruments including a **petfel** and some wind chimes	0.31
• they finished with new moon city a song about a suburb of drem which featured beautifully controlled **petfel** playing from callum	0.23
• a programme of jazz and classical music showing the **petfel** as an instrument of both musical genres	0.27

Table 5: Context sentences and corresponding attention weights for the made-up word "petfel"

(2) and (4); consistently, the embeddings obtained from those sentences are very similar. Furthermore, of all four sentences, these two are the ones best suited for a simple averaging model as they contain informative, frequent words like "instrument", "chimes" and "music".

5 Conclusion

We have introduced attentive mimicking (AM) and showed that attending to informative and reliable contexts improves representations of rare and medium-frequency words for a diverse set of evaluations.

In future work, one might investigate whether attention mechanisms on the word level (cf. Ling et al., 2015) can further improve the model's performance. Furthermore, it would be interesting to investigate whether the proposed architecture is also beneficial for languages typologically different from English, e.g., morphologically rich languages.

Acknowledgments

This work was funded by the European Research Council (ERC #740516). We would like to thank the anonymous reviewers for their helpful comments.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660. Association for Computational Linguistics.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*, 41:677–705.

Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso, and Chu-Cheng Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1367–1372.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777. Association for Computational Linguistics.

Alexandre Salle and Aline Villavicencio. 2018. Incorporating subword information into matrix factorization word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 66–71. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2019. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Cyrus Shaoul and Chris Westbury. 2010. The westbury lab wikipedia corpus.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–68. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *CoRR*, abs/1607.02789.

Yadollah Yaghoobzadeh, Katharina Kann, and Hinrich Schütze. 2018. Evaluating word embeddings in multi-label classification using fine-grained name typing. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 101–106. Association for Computational Linguistics.

Evaluating Style Transfer for Text

Remi Mir[1], **Bjarke Felbo**[2], **Nick Obradovich**[2], **Iyad Rahwan**[2]
[1]Department of EECS, Massachusetts Institute of Technology
[2]Media Lab, Massachusetts Institute of Technology
`{rmir, bfelbo, nobradov, irahwan}@mit.edu`

Abstract

Research in the area of style transfer for text is currently bottlenecked by a lack of standard evaluation practices. This paper aims to alleviate this issue by experimentally identifying best practices with a Yelp sentiment dataset. We specify three aspects of interest (style transfer intensity, content preservation, and naturalness) and show how to obtain more reliable measures of them from human evaluation than in previous work. We propose a set of metrics for automated evaluation and demonstrate that they are more strongly correlated and in agreement with human judgment: direction-corrected Earth Mover's Distance, Word Mover's Distance on style-masked texts, and adversarial classification for the respective aspects. We also show that the three examined models exhibit tradeoffs between aspects of interest, demonstrating the importance of evaluating style transfer models at specific points of their tradeoff plots. We release software with our evaluation metrics to facilitate research.

1 Introduction

Style transfer in text is the task of changing an attribute (style) of an input, while retaining non-attribute related content (referred to simply as *content* for brevity in this paper).[1] For instance, previous work has modified text to make it more positive (Shen et al., 2017), romantic (Li et al., 2018), or politically slanted (Prabhumoye et al., 2018).

Some style transfer models enable modifications by manipulating latent representations of the text (Shen et al., 2017; Zhao et al., 2018; Fu et al., 2018), while others identify and replace style-related words directly (Li et al., 2018). Regardless of approach, they are hard to compare as there is currently neither a standard set of evaluation practices, nor a clear definition of which exact aspects to evaluate. In Section 2, we define three key aspects to consider. In Section 3, we summarize issues with previously used metrics. Many rely on human ratings, which can be expensive and time-consuming to obtain.

To address these issues, in Section 4, we consider how to obtain more reliable measures of human judgment for aspects of interest, and automated methods more strongly correlated with human judgment than previously used methods. Lastly, in Section 5, we show that the three examined models exhibit aspect tradeoffs, highlighting the importance of evaluating style transfer models at specific points of their tradeoff plots. We release software with our evaluation metrics at `https://github.com/passeul/style-transfer-model-evaluation`.

2 Aspects of Evaluation

We consider three aspects of interest on which to evaluate output text x' of a style transfer model, potentially with respect to input text x:

1. *style transfer intensity* $STI(SC(x), SC(x'))$ quantifies the difference in style, where $SC(\cdot)$ maps an input to a style distribution
2. *content preservation* $CP(x, x')$ quantifies the similarity in content between the input and the output
3. *naturalness* $NT(x')$ quantifies the degree to which the output appears as if it could have been written by humans

Style transfer models should be compared across all three aspects to properly characterize differences. For instance, if a model transfers from negative to positive sentiment, but alters content such as place names, it preserves content poorly.

---

[1] This definition of style transfer makes a simplifying assumption that "style" words can be distinguished from "content" words, or words carrying relatively less or no stylistic weight, such as "cafè" in "What a nice cafè." The definition is motivated by penalizing unnecessary changes to content words, e.g. "What a nice cafè" to "This is an awful cafè."

Proceedings of NAACL-HLT 2019, pages 495–504
Minneapolis, Minnesota, June 2 - June 7, 2019.

	Style Transfer			Content Preservation			Naturalness	
	HRC(x')	HRD(x')	SC(x')	HRC(x,x')	HRR(x,$\{x'\}$)	BLEU(x,x')	HRC(x')	PPL(x')
CAAE		x	x		x		x^{F}	
ARAE		x	x	x		x	x	x^{F}
DAR	x		x	x		x	x^{G}	

Table 1: Summary of past evaluation techniques. HRC is human rating on a continuous scale (e.g. 1 to 5). HRD is on discrete options (e.g. positive/negative). HRR is human ranking (most to least similar) of outputs, with respect to given input x. $\{x'\}$ is the set of x' from models trained on different parameters. SC is a style classifier. PPL is perplexity. Superscripts denote that evaluation is done for fluency (F) or grammar (G), which we consider subsets of naturalness. Readers can see the original papers for details on methods falling under these techniques.

If it preserves content well, but sequentially repeats words such as "the", the output is unnatural. Conversely, a model that overemphasizes text reconstruction would yield high content preservation and possibly high naturalness, but little to no style transfer. All three aspects are thus critical to analyze in a system of style transfer evaluation.

3 Related Work

We review previously used approaches for evaluating the outputs of style transfer models. Due to the high costs related to obtaining human evaluations, we focus on three models: the cross-aligned autoencoder (CAAE), adversarially regularized autoencoder (ARAE), and delete-and-retrieve (DAR) models (Shen et al., 2017; Zhao et al., 2018; Li et al., 2018). Table 1 illustrates the spread of evaluation practices in these papers using our notation from Section 2, showing that they all rely on a different combination of human and automated evaluation. For human evaluation, the papers use different instruction sets and scales, making it difficult to compare scores. Below we describe the automated metrics used for each aspect. Some rely on training external models on the corpus of input texts, X, and/or the corpus of output texts, X'. We encourage readers seeking details on how to compute the metrics to reference the algorithms in the original papers.

Style Transfer Previous work has trained classifiers on X and corresponding style labels, and measured the number of outputs classified as having a target style (Shen et al., 2017; Zhao et al., 2018; Li et al., 2018). Results from this *target style scoring* approach may not be directly comparable across papers due to different classifiers used in evaluations.

Content Preservation To evaluate content preservation between x and x', previous work has used BLEU (Zhao et al., 2018; Li et al., 2018), an n-gram based metric originally designed to evaluate machine translation models (Papineni et al., 2002). BLEU does not take into account the aim of style transfer models, which is to alter style by necessarily changing words. Intended differences between x and x' are thus penalized.

Naturalness Past evaluations of naturalness have relied largely on human ratings on a variety of scales under different names: grammaticality, fluency/readability, and naturalness itself (Table 1). An issue with measuring grammaticality is that text with proper syntax can still be semantically nonsensical, e.g. "Colorless green ideas sleep furiously" (Chomsky, 1957). Furthermore, input texts may not demonstrate perfect grammaticality or readability, despite being written by humans and thus being natural by definition (Section 2). This undermines the effectiveness of measures for such specific qualities of output texts.

Zhao et al. (2018) used perplexity to evaluate fluency, which, like grammaticality, we consider a subset of naturalness itself. Low perplexity signifies less uncertainty over which words can be used to continue a sequence, quantifying the ability of a language model to predict gold or reference texts (Brown et al., 1992; Young et al., 2006). However, style transfer outputs are not necessarily gold standard, and the correlation between perplexity and human judgments of those outputs is unknown in the style transfer setting.

4 Methods

We describe how to construct a style lexicon for use in human and automated evaluations. We also describe best practices that we recommend for obtaining scores of those evaluations, as well as how they can be used for evaluating other datasets. Please refer to Section 5 for experimental results.

Negative Sentiment	Positive Sentiment
ruined	mouthwatering
worst	delightfully
failure	wonderfully
lackluster	marvelous
horrible	refreshing

Table 2: Sample of words in a sentiment style lexicon.

4.1 Construction of Style Lexicon

Because the process of style transfer may result in the substitution or removal of more stylistically weighted words, it is ideal to have a lexicon of style-related words to reference. Words in x and/or x' that also appear in the lexicon can be ignored in evaluations of content preservation.

While building a new style lexicon or an extension of existing ones like WordNet-Affect (Strapparava and Valitutti, 2004) may be feasible with binary sentiment as the style, it may not be scalable to manually do so for various other types of styles. Static lexica also might not take context into account. This is an issue for text with words or phrases that are ambiguous in terms of stylistic weight, e.g. “dog” in “That is a man with a dog” vs. “That man is a dog.”

It is more appropriate to automate the construction of a style lexicon per dataset of interest. While multiple options may exist for doing so, we emphasize the simplicity and replicability of training a logistic regression classifier on X and corresponding style labels. We populate the lexicon with features having the highest absolute weights, as those have the most impact on the outcome of the style labels. (Table 2 shows sample words in the lexicon constructed for the dataset used in our experiments.) While sentiment datasets have been widely used in the literature (Shen et al., 2017; Zhao et al., 2018; Li et al., 2018), a lexicon can be constructed for other datasets in the same manner, as long as the dataset has style labels.

Given existing NLP techniques, it may not be possible to correctly identify all style-related words in a text. Consequently, there is a tradeoff between identifying more style-related words and incorrectly marking some other (content) words as style-related. We opt for higher precision and lower recall to minimize the risk of removing content words, which are essential to evaluations of content preservation. This issue is not critical because researchers can compare their style transfer methods using our lexicon.

4.2 Human Evaluation

As seen in Table 1, past evaluations of both style transfer and naturalness consider only output text x'. Existing work from other fields have, however, shown that asking human raters to evaluate two relative comparisons provides more accurate scores than asking them to provide a numerical score for a single observation (Stewart et al., 2005; Bijmolt and Wedel, 1995). With this knowledge, we construct more reliable ways of obtaining human evaluations via *relative scoring* instead of *absolute scoring*.

Style Transfer Intensity Past evaluations have raters mark the degree to which x' exhibits a target style (Li et al., 2018). We instead ask raters to score the difference in style between x and x', on a scale of 1 (identical styles) to 5 (completely different styles). This approach can also used for non-binary cases. Consider text modeled as a distribution over multiple emotions (e.g. happy, sad, scared, etc.), where each emotion can be thought of as a style. One task could be to make a scared text more happy. Presented with x and x', raters would still rate the degree to which they differ in style.

Content Preservation We consider the difficulty of asking raters to ignore style-related words as done in (Shen et al., 2017). Because not all raters may identify the same words as stylistic, their evaluations may vary substantially from one another. To account for this, we ask raters to evaluate content preservation on the same texts, but where we have masked style words using our style lexicon. Under this new “masking” approach, raters have a simpler task, as they are no longer responsible for taking style into account when they rate the similarity of two texts on a scale of 1 to 5.

Naturalness We ask raters to determine whether x or x' (they are not told which is which) is more natural. An x' marked as more natural indicates some success on the part of the style transfer model, as it is able to fool the rater. This is in contrast to previous work, where raters score the naturalness of x' on a continuous scale without taking x into account at all, even though x serves as the basis for comparison of what is considered natural.

4.3 Automated Evaluation

In this section, we describe our approaches to automating the evaluation of each aspect of interest.

No modification Input: the girls up front *incompetent* . Output: the girls up front are *amazing* .
Style removal Input: the girls up front . Output: the girls up front are .
Style masking Input: the girls up front $\langle customstyle \rangle$. Output: the girls up front are $\langle customstyle \rangle$.

Table 3: Text under different settings of style-based modification, as used in evaluations of content preservation. The sample output is from ARAE ($\lambda = 1$).

Style Transfer Intensity Rather than count how many output texts achieve a target style, we can capture more nuanced differences between the style distributions of x and x', using Earth Mover's Distance (Rubner et al., 1998; Pele and Werman, 2009). $EMD(SC(x), SC(x'))$ is the minimum "cost" to turn one distribution into the other, or how "intense" the transfer is. Distributions can have any number of values (styles), so EMD handles binary and non-binary datasets.

Note that even if $argmax(SC(x'))$ is not the target style class, EMD still acknowledges movement towards the target style with respect to $SC(x)$. However, we penalize (negate) the score if $SC(x')$ displays a relative change of style in the wrong direction, away from the target style.

Depending on x, not a lot of rewriting may be necessary to achieve a different style. This is not an issue, as STI relies on a style classifier to quantify not the difference between the content of x and x', but their style distributions. For the style classifier, we experiment with textcnn (Kim, 2014; Lee, 2018) and fastText (Joulin et al., 2017).

Content Preservation We first subject texts to different settings of modification: style removal and style masking. This is to address undesired penalization of metrics on texts expected to demonstrate changes after style transfer (Section 3). For style removal, we remove style words from x and x' using the style lexicon. For masking, we replace those words with a $\langle customstyle \rangle$ placeholder. Table 3 exemplifies these modifications.

For measuring the degree of content preservation, in addition to the widely used BLEU, we consider METEOR and embedding-based metrics. METEOR is an n-gram based metric like BLEU, but handles sentence-level scoring more robustly, allowing it to be both a sentence-level and corpus-level metric (Banerjee and Lavie, 2005).

For the embedding-based metrics, word embeddings can be obtained with methods like Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). Sentence-level embeddings can be comprised of the most extreme values of word embeddings per dimension (*vector extrema*) (Forgues et al., 2014), or word *embedding averages* (Sharma et al., 2017). *Word Mover's Distance* (WMD), based on EMD, calculates the minimum "distance" between word embeddings of x and of x', where smaller distances signify higher similarity (Kusner et al., 2015). *Greedy matching* greedily matches words in x and x' based on their embeddings, calculates their similarity (e.g. cosine similarity), and averages all the similarities. It repeats the process in the reverse direction and takes the average of those two scores (Rus and Lintean, 2012).

We evaluate with all these metrics to identify the one most strongly correlated with human judgment of content preservation.

Naturalness For a baseline understanding of what is considered "natural," any method used for automated evaluation of naturalness requires the human-sourced input texts. We train unigram and neural logistic regression classifiers (Bowman et al., 2016) on samples of X and X' for each transfer model. Via adversarial evaluation, these classifiers must distinguish human-generated inputs from machine-generated outputs. The more natural an output is, the likelier it is to fool a classifier (Jurafsky and Martin, 2018). We calculate agreement between each type of human evaluation (Section 4.2) and each classifier AC. Agreement is the ratio of instances where humans and AC rate a text as more natural than the other.

We also train LSTM language models (Hochreiter and Schmidhuber, 1997) on X and compute sentence-level perplexities for each text in X' in order to determine the relative effectiveness of adversarial classification as a metric.

5 Experiments and Results

Due to high costs of human evaluation, we focus on CAAE, ARAE, and DAR models with transfer tasks based on samples from the Yelp binary sentiment dataset (Shen et al., 2017).[2] Below we detail

---

[2] Like most literature, including the papers on CAAE, ARAE and DAR, we focus on the binary case. Creating a high-quality, multi-label style transfer dataset for evaluation is a demanding task, which is out of scope for this paper.

Input	would *n't recommend* until management works on friendliness and communication with residents .
ARAE ($\lambda = 1$)	highly *recommend* this place while living in tempe and management .
CAAE ($\rho = 0.5$)	would highly *recommend* management on duty and staff on business .
DAR ($\gamma = 500$)	until management works on friendliness and is a *great* place for communication with residents .

Table 4: Sample outputs of a negative to positive sentiment style transfer task. Italicized words are style-related, according to a style lexicon. They can be masked or removed in evaluations of content preservation (Section 4.3).

the range of parameters each model is trained on in order to compare evaluation practices and generate aspect tradeoff plots. Each of three Amazon Turk raters evaluated 244 texts per aspect, per model. Of those texts, half are originally of positive sentiment transferred to negative, and vice versa.

For brevity, we reference average scores (correlation, kappa, and agreement, each of which is described below) from across all models in our analysis of results. For detailed scores per model, please refer to the corresponding tables.

5.1 Style Transfer Models

For each style transfer model, we choose a wide range of training parameters to allow for variation of content preservation, and indirectly, of style transfer intensity, in X. We show sample outputs from the models for a given input text in Table 4.

CAAE uses autoencoders (Vincent et al., 2008) that are cross-aligned, assuming that texts already share a latent content distribution (Shen et al., 2017). It uses latent states of the RNN and multiple discriminators to align distributions of texts in X' exhibiting one style with distributions of texts in X exhibiting another. Adversarial components help separate style information from the latent space where inputs are represented. We train CAAE on various values (0.01, 0.1, 0.5, 1, 5) of ρ, a weight on the adversarial loss.

CAAE is a baseline for other style transfer models, such as ARAE, which trains a separate decoder per style class (Zhao et al., 2018). We train ARAE on various values (1, 5, 10) of λ, which is also a weight on adversarial loss.

The third model that we evaluate, which also uses CAAE as a baseline, avoids adversarial methods in an approach called Delete-and-Retrieve (DAR) (Li et al., 2018). It identifies and removes style words from texts, searches for related words pertaining to a new target style, and combines the de-stylized text with the search results using a neural model. We train DAR on $\gamma = 15$, where γ is a threshold parameter for the maximum number of style words that can be removed from texts, with respect to the size of the corpus vocabulary. For this single training value, we experiment with a range of γ values (0.1, 1, 15, 500) during test time because, by design, the model does not need to be retrained (Li et al., 2018).

Model	Text Modification Setting	
	Unmasked	Style Masked
CAAE	0.158	**0.289**
ARAE	0.201	**0.321**
DAR	0.161	**0.281**
Average	0.173	**0.297**

Table 5: Fleiss' kappas for human judgments of content preservation of unmasked and style-masked texts.

Model	Absolute		Relative
	$\tau = 3$	$\tau = 2$	
CAAE	0.193	0.321	**0.579**
ARAE	0.215	0.415	**0.741**
DAR	0.103	0.201	**0.259**
Average	0.170	0.312	**0.526**

Table 6: Fleiss' kappas for human judgments of absolute naturalness and relative naturalness of texts.

5.2 Human Evaluation

We use Fleiss' kappa κ of inter-rater reliability (see formula in L. Fleiss and Cohen, 1973) to identify the more effective human scoring task for different aspects of interest. The kappa metric is often levied in a relative fashion, as there are no universally accepted thresholds for agreements that are slight, fair, moderate, etc. For comprehensive experimentation, we compare kappas over the outputs of each style transfer model. The kappa score for ratings of content preservation based on style-masked texts is 0.297. Given the kappa score of 0.173 for unmasked texts, style masking is a more reliable approach towards human evaluation for content preservation (Table 5).

For style transfer intensity, kappas for relative scoring do not show improvement over the previously used approach of absolute scoring of x'. However, we observe the opposite for the aspect of naturalness. Kappas for relative naturalness scoring tasks exceed those of the absolute scoring ones (Table 6). Despite the two types of tasks having

Model	fastText		textcnn	
	Target Style Scores	Earth Mover's Distance	Target Style Scores	Earth Mover's Distance
CAAE	0.566 ± 0.038	**0.573 ± 0.038**	0.587 ± 0.037	**0.589 ± 0.037**
ARAE	0.513 ± 0.053	**0.516 ± 0.053**	0.515 ± 0.053	**0.519 ± 0.053**
DAR	0.470 ± 0.049	**0.539 ± 0.045**	0.508 ± 0.047	**0.566 ± 0.043**
Average	0.516 ± 0.047	**0.543 ± 0.045**	0.537 ± 0.046	**0.558 ± 0.044**

Table 7: Correlations of automated style transfer intensity metrics with human scores.

Model	BLEU	METEOR	Embed Average	Greedy Match	Vector Extrema	WMD
CAAE	0.458 ± 0.044	**0.498 ± 0.042**	0.370 ± 0.048	0.489 ± 0.043	0.496 ± 0.042	0.496 ± 0.042
ARAE	0.337 ± 0.064	0.387 ± 0.062	0.313 ± 0.065	0.419 ± 0.060	0.423 ± 0.060	**0.445 ± 0.058**
DAR	0.440 ± 0.051	0.455 ± 0.050	0.379 ± 0.054	0.472 ± 0.049	0.472 ± 0.049	**0.484 ± 0.048**
Average	0.412 ± 0.053	0.447 ± 0.051	0.354 ± 0.056	0.460 ± 0.051	0.464 ± 0.050	**0.475 ± 0.049**

Table 8: Absolute correlations of content preservation metrics with human scores on texts with style removal.

Model	BLEU	METEOR	Embed Average	Greedy Match	Vector Extrema	WMD
CAAE	0.488 ± 0.043	0.517 ± 0.041	0.356 ± 0.049	0.490 ± 0.043	0.496 ± 0.042	**0.517 ± 0.041**
ARAE	0.356 ± 0.063	0.374 ± 0.062	0.302 ± 0.066	0.405 ± 0.061	0.422 ± 0.060	**0.457 ± 0.057**
DAR	0.444 ± 0.050	0.454 ± 0.050	0.370 ± 0.054	0.450 ± 0.050	0.473 ± 0.049	**0.475 ± 0.049**
Average	0.429 ± 0.052	0.448 ± 0.051	0.343 ± 0.056	0.448 ± 0.051	0.464 ± 0.050	**0.483 ± 0.049**

Table 9: Absolute correlations of content preservation metrics with human scores on texts with style masking.

Model	Unigram Adv. Clf.		Neural Adv. Clf.	
	Absolute	Relative	Absolute	Relative
CAAE	56.07	**64.51**	57.38	**67.87**
ARAE	49.45	**66.67**	50.68	**67.90**
DAR	65.16	**65.57**	61.07	**62.30**
Average	56.89	**65.58**	56.38	**66.02**

Table 10: Percent agreement between adversarial classifiers and human scores on the naturalness of texts.

different numbers of categories (2 vs 5), we can compare them by using a threshold τ to bin the absolute score for each text into a "natural" group (x' is considered to be more natural than x) or "unnatural" one (vice versa), like in relative scoring. For example, $\tau = 2$ places texts with absolute scores greater than or equal to 2 into the natural group. Judgments for relative tasks yield greater inter-rater reliability than those of absolute tasks across multiple thresholds ($\tau \in \{2, 3\}$). This suggests that the relative scoring paradigm is preferable in human evaluations of naturalness.

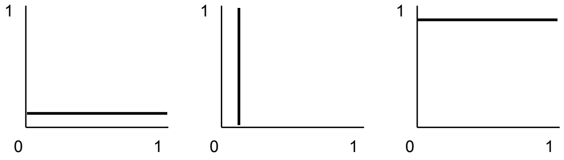
5.3 Automated Evaluation

Per aspect of interest, we compute Pearson correlations between scores from the existing metric and human judgments. (As there were three raters for any given scoring task, we take the average of their scores.) We do the same for our proposed metrics to identify which metric is more reliable for automated evaluation of a given aspect.

For style transfer intensity, across both the fastText and textcnn classifiers, our proposed direction-corrected Earth Mover's Distance metric has higher correlation with human scores than the past approach of target style scoring (Table 7).

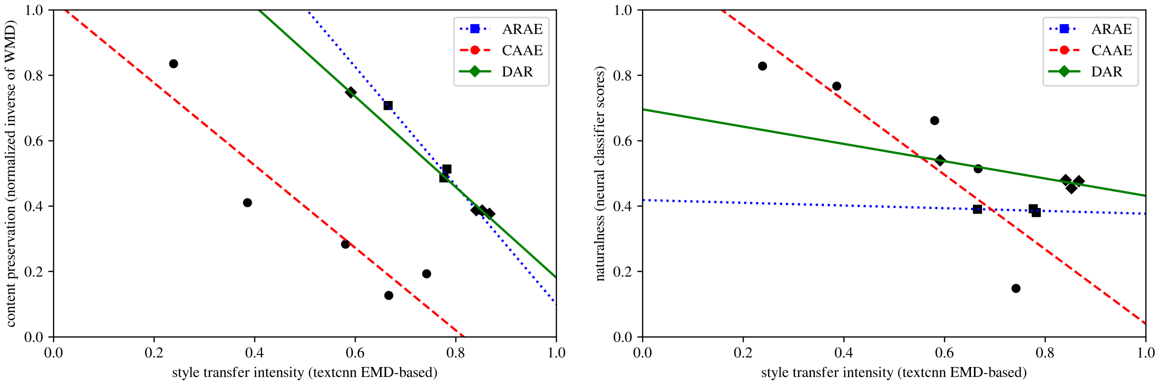
For content preservation, METEOR, shown to have higher correlation with human judgments than BLEU for machine translation (Banerjee and Lavie, 2005), shows the same relationship for style transfer. However, across various text modification settings, WMD generally shows the strongest correlation with human scores (Tables 8 and 9). Because WMD is lower when texts are more similar, it is anti-correlated with human scores. We take absolute correlations to facilitate comparison with other content preservation metrics. With respect to text modification, style masking may be more suitable as it, on average for WMD, exhibits a higher correlation with human judgments.

For naturalness, both unigram and neural classifiers exhibit greater agreement on which texts are considered more natural with the humans given relative scoring tasks than with those given absolute scoring tasks (Table 10), although the neural classifier achieves higher agreements on average. We also confirm that sentence-level perplexity is



(a) (b) (c)

Figure 1: Extreme tradeoff plots, with style transfer intensity on the x-axis and content preservation on the y-axis.



(a) Content vs. Style Tradeoffs

(b) Naturalness vs. Style Tradeoffs

Figure 2: Tradeoffs between aspects of evaluation, using metrics most strongly correlated with human scores.

not an appropriate metric. It exhibits no significant correlation with human scores ($\alpha = 0.05$). These results suggest that adversarial classifiers can be useful for automating measurement of naturalness.

5.4 Aspect Tradeoffs

Previous work has compared models with respect to a single aspect of interest at a time, but has only, to a limited degree, considered how relationships between multiple aspects influence these comparisons. In particular, concurrent work by (Li et al., 2018) examines tradeoff plots, but focuses primarily on variants of its own model, while including only a single point on the plots of style transfer models from other papers. For a comprehensive comparison, it is ideal to have plots for all models.

It is helpful to first understand the tradeoff space. For example, we define extreme cases for style transfer intensity and content preservation, where we assume measurement of the latter ignores stylistic content. Consider two classes of suboptimal models. One class produces outputs with a wide range of style transfer intensity, but poor content preservation (Figure 1a). The other class of models produces outputs with low style transfer intensity, but a wide range of content preservation (Figure 1b).

This is in contrast to a model that yields a wide range of style transfer intensity and consistently high content preservation (Figure 1c). If we take that to be an ideal model for a sentiment dataset, we can interpret models with better performance to be the ones whose tradeoff plots are closer to that of the ideal model and farther from those of the suboptimal ones. The plot for an ideal model will likely vary by dataset, especially because the tradeoff between content preservation and style transfer intensity depends on the level of distinction between style words and content words of the dataset.

With this interpretation of the tradeoff space, we construct a plot for each style transfer model (Figure 2), where each point represents a different hyperparameter setting for training (Section 5.1). We collect scores based on the automated metrics most strongly correlated with human judgment: direction-corrected EMD for style transfer intensity, WMD for content preservation, and percent of output texts marked by an adversarial classifier as more natural than input texts. Because WMD scores are lower when texts are more similar, we instead take the normalized inverses of the scores to represent the degree of content preservation.

Across all models, there is a trend of reduction in content preservation and naturalness as style transfer intensity increases. Without the plots, one might conclude that ARAE and DAR perform substantially differently, especially if hyperparameters are chosen such that ARAE achieves the left-most point on its plot and DAR achieves the right-most point on its plot. With the plots, at least for the set of hyperparameters considered, it is evident that they perform comparably (Figure 2a) and do not exhibit the same level of decrease in naturalness as CAAE (Figure 2b).

6 Discussion

Previous work on style transfer models used a variety of evaluation methods (Table 1), making it difficult to meaningfully compare results across papers. Moreover, it is not clear from existing research how exactly to define particular aspects of interest, or which methods (whether human or automated) are most suitable for evaluating and comparing different style transfer models.

To address these issues, we specified key aspects of interest (style transfer intensity, content preservation, and naturalness) and showed how to obtain more reliable measures of them from human evaluation than in previous work. Our proposed automated metrics (direction-corrected EMD, WMD on style-masked texts, and adversarial classification) exhibited stronger correlations with human scores than existing automated metrics on a binary sentiment dataset. While human evaluation may still be useful in future research, automation facilitates evaluation when it is infeasible to collect human scores due to prohibitive cost or limited time.

6.1 Human Evaluation

For style transfer intensity, the relative scoring task (rating the degree of stylistic difference between x and x') did not have greater rater reliability than the previously used task of rating output texts on an absolute scale. This is likely due to task complexity or rater uncertainty, which motivates the need for further exploration of task design for this particular aspect of interest.

For content preservation, our form of human evaluation operates on texts whose style words are masked out, unlike the previous approach (no masking). Our approach addresses the unintentional variable of rater-dependent style identification that could lead to noisy, less reliable ratings.

Identification and masking of words was made possible with a style lexicon. We automatically constructed the lexicon in a way that can be done for any style dataset, as long as style labels are available (Section 4.1). We acknowledge a trade-off between filling the lexicon with more style words and being conservative in order to avoid capturing content words. We justify taking a more conservative approach as content words are naturally critical to evaluations of content preservation.

For naturalness, we introduced a paradigm of relative scoring that uses both the output and input texts. This achieved a higher inter-rater reliability than did absolute scoring, the previous approach.

6.2 Automated Evaluation

For style transfer intensity, we proposed using a metric with EMD as the basis to acknowledge the spectrum of styles that can appear in outputs and to handle both binary and non-binary datasets. The metric also accounts for direction by penalizing scores in the cases where the style distribution of the output text explicitly moves away from the target style. Previous work used external classifiers, whose style distributions for x and x' can be used to calculate direction-corrected EMD, making it a simple addition to the evaluation workflow.

For content preservation, WMD (based on EMD) works in a similar fashion, but with word embeddings of x and of x'. BLEU, used widely in previous work, may yield weaker correlations with human judgment in comparison as it was designed to have multiple reference texts per candidate text (Papineni et al., 2002). Several reference texts, which are more common in machine translation tasks, increase the chance of n-gram (such as $n \geq 3$) overlap with the candidate. In the style transfer setting, however, the only reference text for x' is x. Having a single reference text reduces the likelihood of overlap and the overall effectiveness of BLEU.

For naturalness, strong agreement of adversarial classifiers with relative scores assigned by humans suggest that classifiers are suitable for automated evaluation. One might assume input texts would almost always be rated as more natural by both humans and classifiers, biasing the agreement. This is not the case, as we justify our rating scheme with evidence of outputs being rated as more natural across several models (Figure 2b). Output texts classified as more natural indicate some success for a style transfer model, as it can produce texts with a quality like that of human-generated inputs, which are, by definition, natural.

Finally, with aspect tradeoff plots constructed using scores from the automated metrics, we can directly compare models with respect to multiple aspects simultaneously. Points of intersection, or near intersection, for different models signify that they, at the hyperparameters that yielded those points, can achieve similar results for various aspects. These parameters can be useful for understanding the impact of decisions made during model design and optimization phases.

6.3 Future Research

As we confirmed, sentence-level perplexity of output x' is not meaningful by itself for the automated evaluation of naturalness. The idea of using both x and x', akin to how we train automated classifiers of naturalness (Section 4.3), can be extended to construct a perplexity-based metric that also takes into account the perplexity of input x.

Another avenue for future work could be evaluating on datasets with a different style or number of style classes. It is worth studying the distinction between style words and content words in the vocabulary of each such dataset. Given the definition of style transfer and its simplifying assumption in Section 1, it would be reasonable to expect naturally low content preservation scores for any given style transfer model operating on datasets with less distinction, such as those of formality. This is not so much an issue as it is a dataset-specific trend that can be visualized in corresponding tradeoff plots, which would provide a holistic evaluation of model performance. In any case, results from inter-rater reliability and correlation testing on these additional datasets would overall enable more consistent evaluation practices and further progress in style transfer research.

Acknowledgments

We would like to thank Juncen Li, Tianxiao Shen, and Junbo (Jake) Zhao for guidance in the use of their respective style transfer models. These models serve as markers of major progress in the area of style transfer research, without which this work would not have been possible.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.

Tammo Bijmolt and Michel Wedel. 1995. The effects of alternative methods of collecting similarity data for multidimensional scaling. *International Journal of Research in Marketing*, 12(4):363–371.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21. Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1).

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Dan Jurafsky and James H Martin. 2018. *Speech and Language Processing*, volume 3.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, pages 957–966.

Joseph L. Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement*, 33:613–619.

Dongjun Lee. 2018. text-cnn. `https://github.com/DongjunLee/text-cnn-tensorflow`.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Ofir Pele and Michael Werman. 2009. Fast and robust earth mover's distances. pages 460–467. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876. Association for Computational Linguistics.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision*, pages 59–66. IEEE.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems 30*, pages 6830–6841.

Neil Stewart, Gordon DA Brown, and Nick Chater. 2005. Absolute identification by relative judgment. *Psychological review*, 112(4):881–911.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet affect: an affective extension of wordnet. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103. ACM.

Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. 2006. *The HTK Book Version 3.4*. Cambridge University Press.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911.

Big BiRD: A Large, Fine-Grained, Bigram Relatedness Dataset for Examining Semantic Composition

Shima Asaadi
Technische Universität Dresden, Germany
shima.asaadi@tu-dresden.de

Saif M. Mohammad
National Research Council Canada
saif.mohammad@nrc-cnrc.gc.ca

Svetlana Kiritchenko
National Research Council Canada
svetlana.kiritchenko@nrc-cnrc.gc.ca

Abstract

Bigrams (two-word sequences) hold a special place in semantic composition research since they are the smallest unit formed by composing words. A semantic relatedness dataset that includes bigrams will thus be useful in the development of automatic methods of semantic composition. However, existing relatedness datasets only include pairs of unigrams (single words). Further, existing datasets were created using rating scales and thus suffer from limitations such as inconsistent annotations and scale region bias. In this paper, we describe how we created a large, fine-grained, bigram relatedness dataset (*BiRD*), using a comparative annotation technique called *Best–Worst Scaling*. Each of BiRD's 3,345 English term pairs involves at least one bigram. We show that the relatedness scores obtained are highly reliable (split-half reliability $r = 0.937$). We analyze the data to obtain insights into bigram semantic relatedness. Finally, we present benchmark experiments on using the relatedness dataset as a testbed to evaluate simple unsupervised measures of semantic composition. BiRD is made freely available to foster further research on how meaning can be represented and how meaning can be composed.

1 Introduction

The term *semantic relatedness* refers to the extent to which two concepts are close in meaning. The ability to assess semantic relatedness is central to the use and understanding of language (Hutchison, 2003; Mohammad and Hirst, 2005; Huth et al., 2016). Manual ratings of semantic relatedness are useful for: (a) obtaining insights into how humans perceive and use language; and (b) developing and evaluating automatic natural language systems.

Existing datasets of semantic relatedness, such as the one by Finkelstein et al. (2002), only focus on pairs of unigrams (single words). However, the concept of semantic relatedness applies more generally to any unit of text. Work in semantic representation explores how best to represent the meanings of words, phrases, and sentences. Bigrams (two-word sequences) are especially important there since they are the smallest unit formed by composing words. Thus it would be useful to have large semantic relatedness datasets involving bigrams.

Existing datasets also suffer from shortcomings due to the annotation schemes employed. Except in the case of a few small but influential datasets, such as those by Miller and Charles (1991) and Rubenstein and Goodenough (1965), annotations were obtained using rating scales. (Annotators were asked to give scores for each pair; usually on a discrete 0 to 5 scale.) Rating scales suffer from significant known limitations, including: inconsistencies in annotations by different annotators, inconsistencies in annotations by the same annotator, scale region bias (annotators often have a bias towards a portion of the scale), and problems associated with a fixed granularity (Presser and Schuman, 1996).

Best–Worst Scaling (BWS) is an annotation scheme that addresses these limitations by employing comparative annotations (Louviere, 1991; Cohen, 2003; Louviere et al., 2015; Kiritchenko and Mohammad, 2017). Annotators are given n items at a time (an n-tuple, where $n > 1$ and commonly $n = 4$). They are asked which item is the *best* (highest in terms of the property of interest) and which is the *worst* (least in terms of the property of interest).[1] When

---

[1] At its limit, when $n = 2$, BWS becomes a *paired comparison* (Thurstone, 1927; David, 1963), but then a much larger set of tuples need to be annotated (closer to N^2).

Proceedings of NAACL-HLT 2019, pages 505–516
Minneapolis, Minnesota, June 2 - June 7, 2019.

working on 4-tuples, best–worst annotations are particularly efficient because each best and worst annotation will reveal the order of five of the six items (i.e., for a 4-tuple with items A, B, C, and D, if A is the best, and D is the worst, then A > B, A > C, A > D, B > D, and C > D). It has been empirically shown that annotating $2N$ 4-tuples is sufficient for obtaining reliable scores (where N is the number of items) (Louviere, 1991; Kiritchenko and Mohammad, 2016). Kiritchenko and Mohammad (2017) showed through empirical experiments that BWS produces more reliable and more discriminating scores than those obtained using rating scales.[2]

In this paper, we describe how we obtained fine-grained human ratings of semantic relatedness for English term pairs involving at least one bigram.[3] The other term in the pair is either another bigram or a unigram. We first selected a set of target bigrams AB (A represents the first word in the bigram and B represents the second word). For each AB, we created several pairs of the form AB–X, where X is a unigram or bigram. As X's we chose terms from a diverse set of language resources:

- terms that are transpose bigrams BA—where the first word is B and the second word is A (taken from occurrences in Wikipedia);
- terms that are related to AB by traditional semantic relations such as hypernymy, hyponymy, holonymy, meronymy, and synonymy (taken from WordNet); and
- terms that are co-aligned with AB in a parallel corpus (taken from a machine translation phrase table).

The dataset includes 3,345 term pairs corresponding to 410 ABs. We refer to this dataset as the *Bigram Relatedness Dataset* (or, *BiRD*).

We use BWS to obtain semantic relatedness by: (1) creating items that are pairs of terms, and (2) prompting four items (pairs) at a time and asking annotators to mark the pair that is most related and the pair that is least related. Once the annotations are complete, we obtain real-valued scores of semantic relatedness for each pair using simple arithmetic on the counts of how often an item is chosen best and worst (Orme, 2009; Flynn and Marley, 2014). (Details in Section 3.) To evaluate the quality of BiRD we determine the consistency of the BWS annotations. A commonly used approach to determine consistency in dimensional annotations is to calculate split-half reliability (Cronbach, 1951). We show that our semantic relatedness annotations have a split-half reliability score of $r = 0.937$, indicating high reliability, that is, if the annotations were repeated then similar scores and rankings would be obtained. (Details in Section 4.)

We use BiRD to (a) obtain insights into bigram semantic relatedness, and (b) to evaluate automatic semantic composition methods.

Examining Bigram Semantic Relatedness: Since very little work exists on the semantic relatedness of bigrams, several research questions remain unanswered, including: What is the distribution and mean of the semantic relatedness between a bigram and its transpose?; What is the average semantic relatedness between a bigram and its hypernym?; Are co-aligned terms from a phrase table a good source of term pairs to be included in a semantic relatedness dataset (specifically, do they cover a wide range of semantic relatedness values)?; etc. In Section 5, we present an analysis of BiRD to obtain insights into these questions.

Evaluating Semantic Composition: A common approach to evaluate different methods of representing words via vectors is through their ability to rank pairs of words by closeness in meaning (Pennington et al., 2014; Levy and Goldberg, 2014; Faruqui and Dyer, 2014). BiRD allows for the evaluation of semantic composition methods through their ability to rank pairs involving bigrams, by semantic relatedness. In Section 6, we present benchmark experiments on using BiRD as a testbed to evaluate various common semantic composition methods using various pre-trained word representations. Specifically, we conduct experiments to gain insights into research questions such as: Which common mathematical operations for vector composition (e.g., vector addition, vector multiplication, etc.) capture the semantics of a bigram more accurately?; Which of the two words in a noun phrase bigram (the head noun or

---

[2] See Kiritchenko and Mohammad (2016, 2017) for further details on BWS and its use in NLP applications.

[3] In a separate project, the second author is developing a semantic relatedness dataset for unigrams using BWS (an order of magnitude larger than existing ones). Project page: http://saifmohammad.com/WebPages/Relatedness.html

the modifier) has greater influence on the semantics of the bigram?; etc.

Contributions: The contributions of this work can be summarized as follows:

- We obtain fine-grained human ratings of semantic relatedness for 3,345 term pairs, each of which includes at least one bigram. The other term in the pair is either another bigram or a unigram.
- We use the comparative annotation technique Best–Worst Scaling, which addresses the limitations of traditional rating scales. This is the first time BWS has been used to create a dataset for semantic relatedness. We show that the ratings obtained are highly reliable.
- We analyse BiRD to obtain insights into semantic relatedness when it involves bigrams. We also develop interactive visualizations that allow for easy exploration of the data. (Available on the project webpage.)
- We present benchmark experiments on using BiRD as a testbed to evaluate methods of semantic composition.

The Bigram Relatedness Dataset, visualizations of the data, and the annotation questionnaire are made freely available through the project's webpage.[4] We hope that the new dataset will foster further research on how meaning is composed in bigrams, on semantic representation in general, and on the understanding of bigram semantic relatedness.

The annotation task described in this paper was approved by the National Research Council Canada's Research Ethics Board (protocol number 2018-72). The board examines the proposed methods to ensure that they adhere to the required ethical standards. Special attention was paid to obtaining informed consent and protecting participant anonymity.

2 Background and Related Work

Semantic Relatedness and Semantic Similarity Closeness of meaning can be of two kinds: semantic similarity and semantic relatedness. Two terms are considered to be semantically similar if there is a taxonomic relationship between them such as hyponymy (hypernymy), or troponymy. Two terms are considered to be semantically related if there is any lexical semantic relation between them—taxonomic or non-taxonomic. Semantically similar items tend to share a number of properties. For example, *apples* and *bananas* (co-hyponyms of fruit) are both edible, they grow on trees, they have seeds, etc. On the other hand, semantically related concepts may not have many properties in common, but there exists some relationship between them which lends them the property of being semantically close. For example, *surgeon* and *scalpel* are semantically related as the former uses the latter for their work.

We focus on semantic relatedness in this work, not only because it is the broader class subsuming semantic similarity, but also because many psychology and neuro-linguistic studies have demonstrated the importance of semantic relatedness. Notable among these are studies on semantic priming and fMRI studies that show that the human brain stores information in a thematic manner (based on relatedness) rather than based on similarity (Hutchison, 2003; Huth et al., 2016).

Word-Pair Datasets: Several semantic similarity and relatedness datasets involving unigram pairs (word pairs) exist. Rubenstein and Goodenough (1965) and Miller and Charles (1991) provided influential but small English word–pair datasets with fine–grained semantic similarity scores. More recent larger datasets including hundreds of pairs were provided by Finkelstein et al. (2002) (for relatedness) and Hill et al. (2015) (for similarity). Similar datasets exist in some other languages as well, such as the one by Gurevych (2006) and Panchenko et al. (2016) for relatedness. However, none of these datasets include items that are bigrams.

Bigram Semantic Similarity Datasets: Mitchell and Lapata (2010) created a semantic similarity dataset for 324 bigram pairs. The terms include adjective–noun, noun–noun, and verb–object bigrams. Annotators were asked to choose an integer between one and seven, indicating a coarse semantic similarity rating. Turney (2012) compiled a dataset of 2,180 bigram–unigram synonym pairs from WordNet synsets. (The bigrams are either noun–noun or adjective–noun phrases.) Other pairs were created taking bigrams and words that do not exist in the same synsets.

---

[4] http://saifmohammad.com/WebPages/BiRD.html

He thus created a dataset of synonyms and non-synonyms. In contrast to these datasets, BiRD has fine-grained relatedness scores.

Other Similarity Datasets: There exist datasets on the semantic similarity between sentences and between documents (Marelli et al., 2014; Agirre et al., 2014; Cera et al., 2017). Those are outside the scope of this work.

Other Natural Language Datasets Created Using BWS: BWS has been used for creating datasets for relational similarity (Jurgens et al., 2012), word-sense disambiguation (Jurgens, 2013), word–sentiment intensity (Kiritchenko and Mohammad, 2016), word–emotion intensity (Mohammad, 2018b), and tweet–emotion intensity (Mohammad and Kiritchenko, 2018). The largest BWS dataset is the NRC Valence, Arousal, and Dominance Lexicon, which has valence, arousal, and dominance scores for over 20,000 English words (Mohammad, 2018a).

3 English Bigram Relatedness Dataset

We first describe how we selected the term pairs to include in the bigram relatedness dataset, followed by how they were annotated using BWS.

3.1 Term Pair Selection

Randomly selecting term pairs will result in most pairs being unrelated. This is sub-optimal in terms of the human annotation effort that is to follow. Further, since our goal is to create a gold standard relatedness dataset, we wanted it to include term pairs across the whole range of semantic relatedness: from maximally unrelated to maximally related. Thus, a key challenge in term-pair selection is obtaining pairs with a wide range of semantic relatedness scores, without knowing their true semantic relatedness in advance. In addition, we also wanted the dataset to satisfy the following criteria:

- For each target bigram AB we wanted to include several pairs of the form AB–X, where X is a unigram or bigram.
  Motivation: Applications of semantic relatedness, such as real-word spelling correction and textual entailment, often require judgments of the form '*is AB–X_1 more related or less related than AB–X_2*'.
- There should exist some pairs AB–X, such that X is BA and a common English bigram.
  Motivation: This is useful for testing sensitivity of semantic composition models to word order.
- The unigrams and bigrams should be commonly used English terms.
  Motivation: Data annotation of common terms is expected to be more reliable. Also, common terms are more likely to occur in application datasets.
- There should exist pairs that are taxonomically related (i.e., semantically similar), for example, hypernyms, hyponyms, holonyms, etc.; and there should exist pairs that are not taxonomically related but semantically related nonetheless.
  Motivation: This increases dataset diversity.
- We focus on noun phrases (adjective–noun and noun–noun bigrams).
  Motivation: Noun phrases are the most frequent phrases.

To pursue these criteria, we compiled a set of term pairs from three diverse sources (Wikipedia, WordNet, and a machine translation phrase table) as described below.

Wikipedia: We chose to collect our target bigrams from the English Wikipedia dump (2018).[5] The corpus was tagged with parts of speech (POS) using the NLTK toolbox.[6] For each of the adjective–noun and noun–noun bigrams AB in the corpus, we checked to see if the bigram BA (its transpose) also exists in the corpus. We will refer to such pairs of bigrams as *transpose bigrams*. Only those transpose bigrams (AB and BA) were selected that were both noun phrases and where both AB and BA occur in the corpus with frequencies greater than a pre-chosen threshold t (we chose $t = 30$). For a pair of transpose bigrams, the bigram with the higher frequency was chosen as AB and the bigram with the lower frequency was chosen as the corresponding BA. The above process resulted in 4,095 transpose pairs (AB–BA).

WordNet: Among the 4,095 ABs, 330 exist in WordNet version 3.0 (Fellbaum, 1998).[7] For each of these, we selected (when available) synonyms (at most five), a hypernym, a hyponym, a holonym, and a meronym from WordNet.

---

[5] https://dumps.wikimedia.org/

[6] https://www.nltk.org/

[7] https://wordnet.princeton.edu/download/current-version

Translation Phrase Table: Word-aligned parallel corpora map words in text of one language to those in text of another language. Often this can lead to more than one word/phrase in one language being mapped to a common word/phrase in the other language. We will refer to such terms as being *co-aligned*. Due to the nature of languages and the various forms that the same text can be translated to, co-aligned terms tend to include not just synonyms but also other semantically related terms, and sometimes even unrelated terms. Thus, we hypothesize that it is beneficial to include pairs of co-aligned terms in a semantic relatedness dataset as they pertain to varying degrees of semantic relatedness.

We used an English–French phrase table from the Portage Machine Translation Toolkit (Larkin et al., 2010) to determine additional pairs AB–X.[8] Specifically, for each AB–F entry in the phrase table (where F is a French term) we keep the five most frequent English unigrams and the five most frequent English bigrams (other than AB) that are also aligned to F. Among the 4,095 ABs, 454 occurred in the phrase table. This resulted in 3,255 AB–X pairs in total (1,897 where X is a unigram, and 1,358 where X is a bigram).

Finally, we chose to filter the term pairs, keeping only those ABs that occurred in at least three unique pairs. (So for a given AB, apart from the AB–BA entry, there should be at least two other entries of the form AB–X, generated using WordNet or the phrase table.) We also manually examined the remaining entries and removed those with obscure terms. The final master term pairs list consists of 3,345 AB–X pairs in total (1,718 where X is a unigram, and 1,627 where X is a bigram), corresponding to 410 ABs. Thus on average, each AB occurred in about 8 distinct pairs. This is yet another aspect that makes BiRD unique, as existing datasets were not designed to include terms in multiple pairs. Table 1 shows the number of adjective–noun pairs, the number of noun–noun pairs, and the total number of pairs in BiRD. (We grouped the hypernym and hyponym pairs into a common class, which we will refer to as the *is-a* pairs. Similarly we group the meronym and holonym pairs into a common class, which we will refer to as the *part-whole* pairs.)

Source	# a–n	# n–n	# both
Wikipedia_transpose	80	330	410
WordNet_synonym	18	70	88
WordNet_is-a	49	220	269
WordNet_part-whole	7	30	37
PhraseTable_co-aligned	440	2,101	2,541
All	**594**	**2,751**	**3,345**

Table 1: Number of pairs from different sources.

3.2 Annotating For Semantic Relatedness

As mentioned in the introduction, we use the comparative annotation method Best–Worst Scaling (BWS) to obtain the annotations. From the list of $N = 3,345$ term pairs, we generated $2N = 6,690$ distinct 4-tuples (each 4-tuple is a set of four term pairs) such that each term pair appears in roughly equal distinct tuples, and no term pair appears more than once in a tuple.[9] (Recall that past research has shown that generating 2N 4-tuples in this manner is sufficient for obtaining fairly reliable scores (Louviere, 1991; Kiritchenko and Mohammad, 2017; Mohammad, 2018a).) The annotators were presented with one tuple at a time and were asked to specify which of the four pairs is most close in meaning (or most related) and which term is the least close (or least related).

Detailed annotation instructions (with examples of appropriate and inappropriate responses) were provided. Notably, we made it clear that if terms in the pair have several meanings, then the annotators should consider the meanings that are closest to each other. We also asked the annotators to be mindful of word order (i.e., the meaning of a bigram AB may be different from the meaning of its transpose BA).

We set up the annotation task on the crowdsourcing platform, Figure Eight.[10] We did not collect personally identifiable information from the annotators. The compensation that the annotators would receive was clearly stated. We selected a pool of annotators fluent in English and with a history of high-quality annotations. Annotators were told that they could annotate as many instances as they wished. As mentioned in the Introduction, prior to the annotation, the planned procedure was approved by the National Research Council Canada's Research Ethics Board (protocol number 2018-72).

---

[8] French was chosen as it is close to English and there exist English–French parallel corpora of sufficient size.

[9] If 2N 4-tuples are generated from N items, and each item is to occur in an equal number of tuples, then each item will occur in eight tuples.

[10] https://www.figure-eight.com/

# Term Pairs	# Tuples	# Annotations per Tuple	# Annotations	# Annotators	SHR
3,345	6,690	8 (for most tuples), >8 (for some)	57,482	427	0.9374

Table 2: BiRD annotation statistics. SHR = split-half reliability (as measured by Pearson correlation).

About 2% of the data was annotated beforehand by the authors. These questions are referred to as gold questions. Figure Eight interspersed the gold questions with the other questions. If a crowd worker answered a gold question incorrectly, then they were immediately notified. This served as an additional way to guide the annotators. If an annotator's accuracy on the gold questions fell below 70%, then they were refused further annotation, and all of their annotations were discarded. This served as a mechanism to avoid malicious annotations.

In the task settings for Figure Eight, we specified that we needed annotations from eight people for each 4-tuple.[11] In all, 57,482 pairs of best and worst responses were obtained from 427 annotators.[12]

Annotation Aggregation: The final semantic relatedness scores were calculated from the BWS responses using a simple counting procedure (Orme, 2009; Flynn and Marley, 2014): For each term pair, the semantic relatedness score is the proportion of times the term pair was chosen as the best minus the proportion of times the term pair was chosen as the worst.[13] The scores were linearly transformed to the interval: 0 (lowest semantic relatedness) to 1 (highest semantic relatedness). We refer to the final list of 3,345 English term pairs along with their scores for semantic relatedness as the *Bigram Relatedness Dataset (BiRD)*. Table 2 summarizes key annotation statistics.

4 Reliability of Data Annotations

A commonly used measure of quality in dimensional annotation tasks is the reproducibility of the final scores—the extent to which repeated independent manual annotations produce similar results. To assess this reproducibility, we calculate average *split-half reliability (SHR)* (Cronbach, 1951) as follows: The annotations for each 4-tuple are randomly split into two halves. One set is put in bin 1 and another set in bin 2. Next, two sets of semantic relatedness scores are produced independently from the two bins, 1 and 2, respectively. Then the Pearson correlation between the two sets of scores is calculated. If the annotations are of good quality, then the correlation between the two sets of relatedness scores will be high (closer to 1).[14] This process is repeated 100 times, and the correlations are averaged. The last column in Table 2 shows the result. An SHR of $r = 0.9374$ indicates high reliability.

5 Studying Bigram Semantic Relatedness

Since very little prior work exists on the semantic relatedness of bigrams, several research questions remain unanswered, including:

- If both AB and BA are common English bigrams, then what is the average semantic relatedness between AB and BA?
- What is the range of semantic relatedness between a bigram and its hypernym or hyponym? What is the average semantic relatedness of such pairs? How do these averages and standard deviations vary with respect to the different semantic relations?
- What is the distribution of semantic relatedness values for co-aligned terms?

We now present analyses of the relatedness dataset to obtain insights into these questions.

Figure 1 shows example adjective–noun and noun–noun entries from BiRD. Observe that for the term *ageing population*, the most related term is *ageing society*—a co-aligned term in the phrase table. (Other co-aligned terms have lower relatedness scores.) The transpose bigram *population ageing* is also marked as highly related. WordNet does not provide a synonym for *ageing population*. For the term *adult female*, the WordNet synonym and the transposed bigram (BA) are marked as being most related. Note that the WordNet-provided hyponym *amazon* is marked as less related (probably because that sense of *amazon* is rare). BiRD can be examined

---

[11] Note that since each term pair occurs in eight different 4-tuples, it is involved in $8 \times 8 = 64$ best–worst judgments.

[12] Gold questions were annotated more than eight times.

[13] More complex optimization algorithms exist, such as those described in (Hollis, 2018); however, our past experiments showed that the simple counting procedure obtained the most reliable results.

[14] Scores close to 0 indicate no correlation.

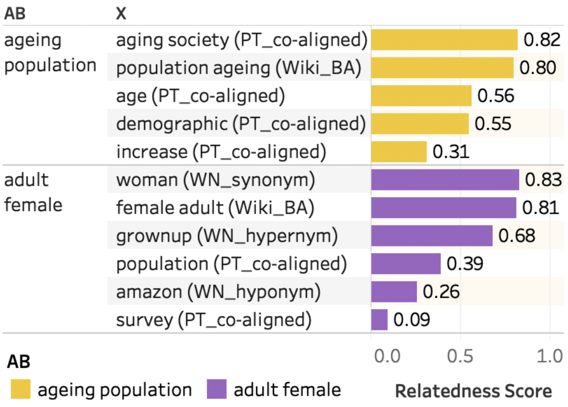


Figure 1: Example entries from BiRD.

for each individual relation and sorted by relatedness scores to determine other example pairs that seemingly should be closely related, but are not highly semantically related in the perception of the average English speaker. These include pairs such as *subject area–discipline* (WordNet synonym) and *frying pan–spider* (WordNet hyponym). The AB–BA pairs with low relatedness, such as *law school–school law, home run–run home,* and *traffic light–light traffic* are especially useful in testing whether measures of semantic composition generate suitably different representations for the terms in such pairs.

Table 3 shows the average semantic relatedness scores as well as standard deviations for the term pairs from various sources.[15] Observe that, on average, the AB–BA pairs and the AB–WordNet synonym pairs are found to be the most related. On average, the AB–WordNet part-whole pairs and the AB–phrase table co-aligned pairs have the lowest semantic relatedness scores. The high average relatedness and low standard deviation (σ) for the transpose bigrams, indicate that these pairs tend to be closely related to each other. The standard deviation is markedly higher for the other sources of word pairs. Manual examination of such pairs (especially those involving WordNet synonyms) revealed that this is often because one of the terms might be related to the other in a rare sense (such as in the *amazon* example). The high standard deviations for hypernyms, hyponyms, meronyms, and holonyms, indicate that pairs connected by this relation in WordNet can still exhibit a wide range of semantic relatedness.

The standard deviations also indicate that 95% of the co-aligned pairs have semantic relatedness between 0.09 and 0.83 (a wide interval). Manual examination revealed that the lowest score pairs were unrelated and the highest score terms were often synonymous. Thus co-aligned pairs from phrase tables are indeed a good source of term pairs for a semantic relatedness dataset, since they include pairs with a wide variety of relatedness values.

Source	**avg. rel.**	σ
Wikipedia_transpose	0.669	0.118
WordNet_synonym	0.640	0.194
WordNet_is-a	0.550	0.177
WordNet_part-whole	0.453	0.193
PhraseTable_co-aligned	0.463	0.189

Table 3: Average relatedness and standard deviation (σ) scores for term pairs from the various sources.

6 Evaluating Methods of Semantic Composition on BiRD

A popular approach to represent word meaning in natural language systems is through vectors that capture the contexts in which the word occurs. An area of active research is how these word vectors can be composed to create representations for larger units of text such as phrases and sentences (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012; Tai et al., 2015). Even though there is a large body of work on how to represent the meanings of sentences (Le and Mikolov, 2014; Kiros et al., 2015; Lin et al., 2017), there is relatively less work on how best to compose the meanings of two words to represent the meaning of a bigram. One reason for this is a lack of suitable evaluation resources. A common approach to evaluate representations of unigrams is through their ability to rank pairs of words by closeness in meaning (Pennington et al., 2014; Levy and Goldberg, 2014; Faruqui and Dyer, 2014). BiRD allows for the evaluation of semantic composition methods through their ability to rank pairs involving bigrams, by semantic relatedness.

Here, we present benchmark experiments on commonly used semantic composition methods by measuring their ability to rank the term pairs in BiRD by relatedness scores. The underlying assumption is that the more accurately a method of semantic composition can determine the representation of a bigram, the more accurately systems can determine the relatedness of that bigram with other terms.

---

[15] The scores for just the adjective–noun pairs and just the noun—noun pairs are similar.

We focus on unsupervised approaches as we wanted to identify how well basic composition operations perform. The applicability of BiRD is much broader though, and it can be used: (1) for evaluating the large number of proposed supervised methods of semantic composition; (2) for evaluating the large number of measures of semantic relatedness; (3) to study the mechanisms underpinning semantic composition; etc. We leave those for future work.

We test three vector space models to obtain word representations: GloVe (Pennington et al., 2014), fastText (Grave et al., 2018), and a traditional model based on matrix factorization of a word–context co-occurrence matrix (Turney et al., 2011). We test four mathematical composition operations: (1) vector addition, (2) element-wise vector multiplication, (3) tensor product with circular convolution (Widdows, 2008), and (4) dilation (Mitchell and Lapata, 2010). In adjective–noun and noun–noun bigrams, the second word usually plays a role of a head noun, and the first word is a modifier. We test the performance of two baseline methods that do not employ vector composition: one that represents a bigram with the vector for the first word and one that represents a bigram with the vector for the second word.

Word representations: We use GloVe word embeddings pre-trained on 840B-token CommonCrawl corpus[16] and fastText word embeddings pre-trained on Common Crawl and Wikipedia using CBOW.[17] For the traditional model, we use the exact word–context co-occurrence matrix described in Turney et al. (2011).[18] They created the matrix from a corpus of 5 × 10^{10} tokens gathered from university websites. The rows correspond to terms (single words from WordNet) and the columns correspond to contexts (single words from WordNet appearing to the left or to the right of the term). Each cell of the matrix is the positive pointwise mutual information between the term and the context. The matrix is decomposed to $\mathbf{U_d \Sigma_d V_d^\top}$ (d denotes dimensionality) via truncated singular value decomposition. Word vectors are obtained from the matrix $\mathbf{U_d \Sigma_d^p}$, where rows correspond to the d-dimensional word vectors and p is the weight factor for singular values in $\mathbf{\Sigma_d}$. We set parameter p to 0.5, and the dimensionality of word vectors to $d = 300$ for all three vector space models.

Unsupervised Compositional Models: For a bigram $w_1 w_2$, let $u \in \mathbb{R}^{1\times d}$ and $v \in \mathbb{R}^{1\times d}$ denote the vectors for words w_1 and w_2, respectively. Each of the methods below applies a different composition function f on the word vectors u and v to obtain the vector representation p for the bigram $w_1 w_2$: $p = f(u, v)$:

- *Addition* (Salton and McGill, 1986): add the two word vectors ($p = u + v$).
- *Multiplication* (Mitchell and Lapata, 2010): element-wise multiplication of the two vectors ($p = u \odot v$, where $p_i = u_i \cdot v_i$).
- *Tensor product with convolution* (Widdows, 2008): outer product of two vectors resulting in matrix Q ($q_{ij} = u_i v_j$). Then, circular convolution is applied to map Q to vector p. This is equivalent to: $p_i = \sum_j u_j \cdot v_{i-j}$.
- *Dilation* (Mitchell and Lapata, 2010): decompose v to parallel and orthogonal components to u, and then stretch the parallel component along u ($p_i = v_i \sum_j u_j u_j + (\lambda - 1) u_i \sum_j u_j v_j$, where λ is the dilation factor). We set $\lambda = 2$.

For the two baseline experiments that do not employ vector composition, *head only*: $p = v$ and *modifier only*: $p = u$.

Semantic Relatedness: The relatedness score for a term pair AB–X in the Bigram Relatedness Dataset (BiRD) is computed by taking the cosine between the vectors representing AB and X, where X can be a unigram or a bigram.

Evaluation: As evaluation metric, we use the Pearson correlation of the relatedness scores predicted by a method with the gold relatedness scores in BiRD. Some words in BiRD do not occur in some of the corpora used to create the word vectors. Thus we conduct experiments on a subset of BiRD (3,159 pairs) for which word vectors exist for all models under consideration. To determine if the differences between the correlation scores are statistically significant, we perform Steiger's Z significance test (Steiger, 1980).

---

[16] https://nlp.stanford.edu/projects/glove/

[17] https://fasttext.cc/docs/en/crawl-vectors.html

[18] We thank Peter Turney for providing the data.

Method	**GloVe**	**fastText**	**Matrix Factor.**
Baselines			
head only	0.342	0.403	0.339
modifier only	0.438	0.495	0.425
Composition methods			
addition	**0.564**	**0.601**	**0.582**
multiplication	0.182	0.328	0.244
tensor product	0.374	0.382	0.451
dilation	0.523	0.495	0.496

Table 4: Pearson correlations of model predictions with BiRD relatedness ratings. Highest scores are in bold.

Results: Table 4 shows the results. Observe that among the three methods of word vector representations, the best results are obtained using fastText (word–context matrix factorization model being a close second). Among the methods of semantic composition, the additive models perform best (for all three ways of representing word vectors). The scores are statistically significantly higher than those of the second best (dilation). The element-wise vector multiplication and tensor product with convolution perform poorly (even worse than the baseline methods). These results differ substantially from the observations by Mitchell and Lapata (2010). In particular, in their work the multiplication model showed the best results, markedly outperforming the addition model. Our results are consistent with the findings of Turney (2012), where too the addition model performed better than the multiplication model. It should be noted though that unlike BiRD which has scores for semantic relatedness, the Mitchell and Lapata (2010) and Turney (2012) datasets have scores for semantic similarity. Further work is required to determine whether certain composition models are better suited for estimating one or the other.

Surprisingly, the baseline model that uses the vector for the modifier word obtains better results than the one that uses the vector for the head noun. (The difference is statistically significant.) To better understand this, we compute relatedness correlations using the *weighted addition* of the two word vectors ($p = \alpha u + (1 - \alpha)v$), where α is a parameter that we vary between 0 and 1, in steps of 0.1. Figure 2 shows the results. Observe that giving more weight (but not too much weight) to the modifier word than the head word is beneficial. $\alpha = 0.7$ and $\alpha = 0.8$ produce the highest correlations. These results raise further questions under what conditions is the role of the modifier particularly prominent, and why. We leave that for future work.

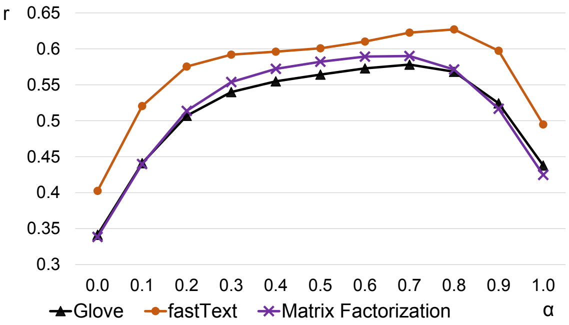


Figure 2: Pearson correlation coefficient (r) of the model predictions using weighted addition with BiRD relatedness ratings. α is varied from 0 to 1 in steps of 0.1. $\alpha = 0.7$ and $\alpha = 0.8$ produce the highest scores.

7 Conclusions

We created a dataset with fine-grained human ratings of semantic relatedness for term pairs involving bigrams. We used the comparative annotation technique Best–Worst Scaling, which addresses the limitations of traditional rating scales. We showed that the ratings obtained are highly reliable (high SHR, $r = 0.937$). We analyzed the dataset to obtain insights into the distributions of semantic relatedness values for pairs associated through various relations such as WordNet assigned lexical semantic relations, transposed bigrams, and co-aligned terms in a parallel corpus. We show that co-aligned terms can be related to varying degrees (from unrelated to synonymous), thereby making them a useful source of term pairs to include in relatedness datasets. Finally, we presented benchmark experiments on using BiRD as a testbed to evaluate various unsupervised methods of semantic composition. We found that the additive models performed best and that giving more weight to the modifier word can improve results further. We make BiRD freely available to foster further research. In the short term, it will be interesting to explore the use of supervised semantic composition methods, including resources and models such as BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018), to determine bigram relatedness.

Acknowledgments

We thank Peter Turney, Michel Simard, and Tara Small for helpful discussions. This work is partially supported by the German Research Foundation (DFG) within the Research Training Group QuantLA (GRK 1763).

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193.

Daniel Cera, Mona Diab, Eneko Agirrec, Inigo Lopez-Gazpioc, Lucia Speciad, and Basque Country Donostia. 2017. SemEval-2017 Task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 1–14.

Steven H. Cohen. 2003. Maximum difference scaling: Improved measures of importance and preference for segmentation. Sawtooth Software, Inc.

Lee J. Cronbach. 1951. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334.

Herbert Aron David. 1963. *The method of paired comparisons*. Hafner Publishing Company, New York.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

T. N. Flynn and A. A. J. Marley. 2014. Best-worst scaling: theory and methods. In Stephane Hess and Andrew Daly, editors, *Handbook of Choice Modelling*, pages 178–201. Edward Elgar Publishing.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).*

Iryna Gurevych. 2006. Thinking beyond the nouns-computing semantic relatedness across parts of speech. Technical report, Darmstadt University of Technology, Germany, Department of Computer Science, Telecooperation.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Geoff Hollis. 2018. Scoring best-worst data in unbalanced many-item designs, with applications to crowdsourcing semantic judgments. *Behavior Research Methods*, 50(2):711–729.

Keith A Hutchison. 2003. Is semantic priming due to association strength or feature overlap? a microanalytic review. *Psychonomic Bulletin & Review*, 10(4):785–813.

Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Atlanta, GA, USA.

David Jurgens, Saif M. Mohammad, Peter Turney, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval)*, pages 356–364, Montréal, Canada.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best–worst scaling. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Svetlana Kiritchenko and Saif M. Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of The Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 3294–3302.

Samuel Larkin, Boxing Chen, George Foster, Uli Germann, Eric Joanis, J. Howard Johnson, and Roland Kuhn. 2010. Lessons from NRC's Portage System at WMT 2010. In *Proceedings of the 5th Workshop on Statistical Machine Translation (WMT-2010)*, pages 127–132.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*, pages 1188–1196.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 2177–2185.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the International Conference on Learning Representations*.

Jordan J. Louviere. 1991. Best-worst scaling: A model for the largest difference judgments. Working Paper.

Jordan J. Louviere, Terry N. Flynn, and A. A. J. Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Saif Mohammad and Graeme Hirst. 2005. Distributional measures as proxies for semantic relatedness. *arXiv:1203.1858. 2005.*

Saif Mohammad and Svetlana Kiritchenko. 2018. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Saif M. Mohammad. 2018a. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of The Annual Conference of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Saif M. Mohammad. 2018b. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and HB.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. 2016. Human and machine judgements for Russian semantic relatedness. In *Proceedings of the International Conference on Analysis of Images, Social Networks and Texts*, pages 221–235.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Stanley Presser and Howard Schuman. 1996. *Questions and Answers in Attitude Surveys: Experiments on Question Form, Wording, and Context.* SAGE Publications, Inc.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Gerard Salton and Michael J. McGill. 1986. *Introduction to modern information retrieval.* McGraw-Hill, Inc.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

James H. Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Louis L. Thurstone. 1927. A law of comparative judgment. *Psychological Review*, 34(4):273.

Peter D Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, volume 26.

Outlier Detection for Improved Data Quality and Diversity in Dialog Systems

Stefan Larson **Anish Mahendran** **Andrew Lee** **Jonathan K. Kummerfeld**
Parker Hill **Michael A. Laurenzano** **Johann Hauswald** **Lingjia Tang** **Jason Mars**
Clinc, Inc.
Ann Arbor, MI, USA
`{stefan,anish,andrew,jkk}@clinc.com`
`{parkerhh,mike,johann,lingjia,jason}@clinc.com`

Abstract

In a corpus of data, outliers are either *errors*: mistakes in the data that are counterproductive, or are *unique*: informative samples that improve model robustness. Identifying outliers can lead to better datasets by (1) removing noise in datasets and (2) guiding collection of additional data to fill gaps. However, the problem of detecting both outlier types has received relatively little attention in NLP, particularly for dialog systems. We introduce a simple and effective technique for detecting both erroneous and unique samples in a corpus of short texts using neural sentence embeddings combined with distance-based outlier detection. We also present a novel data collection pipeline built atop our detection technique to automatically and iteratively mine unique data samples while discarding erroneous samples. Experiments show that our outlier detection technique is effective at finding errors while our data collection pipeline yields highly diverse corpora that in turn produce more robust intent classification and slot-filling models.

1 Introduction

High-quality annotated data is one of the fundamental drivers of progress in Natural Language Processing (e.g. Marcus et al., 1993; Koehn, 2005). In order to be effective at producing an accurate and robust model, a dataset needs to be correct while also diverse enough to cover the full range of ways in which the phenomena it targets occur. Substantial research effort has considered dataset correctness (Eskin, 2000; Dickinson and Meurers, 2003; Rehbein and Ruppenhofer, 2017), particularly for crowdsourcing (Snow et al., 2008; Jiang et al., 2017), but addressing diversity in data has received less attention, with the exception of using data from diverse domains (Hovy et al., 2006). Outlier detection, the task of finding examples in a dataset that are atypical, provides a means of approaching the questions of correctness and diversity, but has mainly been studied at the document level (Guthrie et al., 2008; Zhuang et al., 2017), whereas texts in dialog systems are often no more than a few sentences in length.

We propose a novel approach that uses sentence embeddings to detect outliers in a corpus of short texts. We rank samples based on their distance from the mean embedding of the corpus and consider samples farthest from the mean outliers. Outliers come in two varieties: (1) *errors*, sentences that have been mislabeled whose inclusion in the dataset would be detrimental to model performance, and (2) *unique* samples, sentences that differ in structure or content from most in the data and whose inclusion would be helpful for model robustness. Building upon this approach, we propose a novel crowdsourcing pipeline that distinguishes errors from unique samples and uses the unique samples to guide workers to give more diverse samples.

Experimentally, we find that our outlier detection technique leads to efficient detection of both artificial and real errors in our datasets. We also use the proposed crowdsourcing pipeline to collect new datasets and build models for the dialog system tasks of intent classification and slot-filling. We find that the proposed pipeline produces more diverse data, which in turn results in models that are more robust.

2 Background and Related Work

2.1 Outlier Detection

Outlier detection (Rousseeuw and Leroy, 1987), also called outlier analysis (Aggarwal, 2015) or anomaly detection (Chandola et al., 2009), is the task of identifying examples in a dataset that dif-

Proceedings of NAACL-HLT 2019, pages 517–527
Minneapolis, Minnesota, June 2 - June 7, 2019.

fer substantially from the rest of the data.

For almost two decades, a body of work in NLP has investigated applying these ideas to data in order to identify annotation errors (Abney et al., 1999). Approaches have included the use of scores from trained models for POS tagging (Abney et al., 1999; Eskin, 2000; van Halteren, 2000; Dligach and Palmer, 2011; Fukumoto and Suzuki, 2004), count-based methods that compare examples from across the corpus (Nakagawa and Matsumoto, 2002; Hollenstein et al., 2016), characterizing data based on feature vectors projected down into a low-dimensional space (Guthrie et al., 2008), and tracking the difficulty of learning each example during training (Amiri et al., 2018). One particularly effective approach has been to find *n*-grams that match but have different labels, as shown for annotations including POS tags (Dickinson and Meurers, 2003), syntactic parses (Dickinson and Meurers, 2005; Dickinson, 2010; Dickinson and Smith, 2011), and predicate-argument relations (Dickinson and Lee, 2008). Our work instead uses continuous representations of text derived from neural networks.

While finding errors is an extremely useful application of outlier detection, also of interest are examples that are correct even though they are outliers, as these can be the most interesting and informative examples in a dataset. We term these examples *unique*. The problems of detecting and leveraging the unique examples in a dataset has received less attention, and the work that does exist focuses on identifying complete documents or segments of documents that are outliers out of a broader set of documents: Guthrie et al. (2007) used manually defined feature vectors to identify segments of documents with anomalous style, topic, or tone, and Kumaraswamy et al. (2015) and Zhuang et al. (2017) construct statistical models, identifying complete documents that are outliers within a set based on semantic variation.

Finally, a related but distinct topic is novelty detection (Soboroff and Harman, 2005; Lee, 2015; Ghosal et al., 2018), in which two sets of documents are provided, one that is assumed to be known, and one that may contain new content. The task is to identify novel content in the second set. While outlier detection methods are often applied to this problem, the inclusion of the known document set makes the task fundamentally different from the problem we consider in this work.

2.2 Data Collection

We build on prior work employing online crowd workers to create data by paraphrasing. In particular, we refine the idea of iteratively asking for paraphrases, where each round prompts workers with sentences from the previous round, leading to more diverse data (Negri et al., 2012; Jiang et al., 2017; Kang et al., 2018). We also apply the idea of a multi-stage process, in which a second set of workers check paraphrases to ensure they are correct (Buzek et al., 2010; Burrows et al., 2013; Coucke et al., 2018). Most notably, by incorporating our outlier detection method, we are able to automate detecting detrimental data points while also prompting workers in subsequent rounds to paraphrase more unique examples.

3 Outlier Detection

We propose a new outlier detection approach using continuous representations of sentences. Using that approach, we explored two applications: (1) identifying errors in crowdsourced data, and (2) guiding data collection in an iterative pipeline.

3.1 Method

We detect outliers in a dataset as follows:

1. Generate a vector representation of each instance.
2. Average vectors to get a mean representation.
3. Calculate the distance of each instance from the mean.
4. Rank by distance in ascending order.
5. (Cut off the list, keeping only the top $k\%$ as outliers.)

The final step is parenthesized as in practice we use a dynamic threshold approach, allowing the user to go through as much or as little of the list as they like.

The intuition behind this approach is that we expect our representations to capture the semantic structure of the space for each class. An example that is far away from other examples in the set is therefore less semantically similar in some sense, making it an outlier. Importantly, it may be an outlier for two distinct reasons: (1) it is not a valid instance of this class (i.e., an *error*), or (2) it is an unusual example of the class (i.e., *unique*).

This approach is applied independently to each class of data. As example applications we consider two dialog system tasks: intent classification and slot-filling. For classification, data for each possible intent label is considered separately, meaning we find outliers in the data by considering one intent class at a time. For slot-filling, we group the data into classes based on combinations of slots.

This outlier detection method is rather simple as it relies only on a sentence embedding method, a distance metric, and a threshold k; no other hyperparameters are involved. Moreover, the method requires no training. We shall see in Section 4 that this method performs well compared to baseline methods, no matter what sentence embedding method is used. We use Euclidean distance as our distance metric.[1]

3.1.1 Sentence Representations

Vector representation of sentences is an active area of research and we leverage the following approaches, each of which has been shown to have state of the art results in different use cases:

Universal Sentence Encoder (USE; Cer et al., 2018) A Deep Averaging Network method, which averages word embeddings and passes the result through a feedforward network. The USE is trained using a range of supervised and unsupervised tasks.

Smooth Inverse Frequency (SIF; Arora et al., 2017) A weighted average of word embeddings, with weights determined by word frequency within a corpus. We consider word embeddings from GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018).

Average An unweighted average of word embeddings. While simple, this approach has been shown to be effective for classification (Zhang and Wallace, 2017) and other downstream tasks (Zhu et al., 2018). Again, we consider GloVe and ELMo word embeddings as inputs.

3.1.2 Model Combination

In addition to ranked lists produced by using these core sentence embeddings, we also investigated aggregating the ranked lists using the Borda count, a rank aggregation technique that has previously been used for combining web search results (Dwork et al., 2001).

The Borda count aggregates multiple ranked lists of the same set of items into a single ranked list. First, points are assigned to each item in each list, with an item in position i in a ranked list of length N receiving $N - i$ points. Next, points for items are summed across all of the lists. Finally, the items are ranked by their total number of points, producing a final ranking.

3.2 Application: Error Detection

Our proposed use of outlier detection to identify errors requires no further processing. When used in practice, a user looks through the sorted list of examples, either stopping at a given fraction, or when errors become infrequent enough.

3.3 Application: Uniqueness-driven Data Collection

One core insight in this work is that outlier detection can be used for more than just finding errors. The outliers that are not errors are likely to be the most interesting and informative examples in our dataset. We propose to use these examples to guide data collection in an iterative process, with the goal of yielding more diverse data.

To demonstrate this idea, we developed a novel crowdsourcing pipeline for data collection. Following prior work in crowdsourcing for dialog (Kang et al., 2018; Jiang et al., 2017), we ask crowd workers to write paraphrases of seed sentences with known intents and slot values. This provides linguistic diversity in our data in a way that is easily explained to workers. For instance, given the seed sentence "*What is my savings account balance?*" a worker might write "*How much money do I have in my savings account?*".

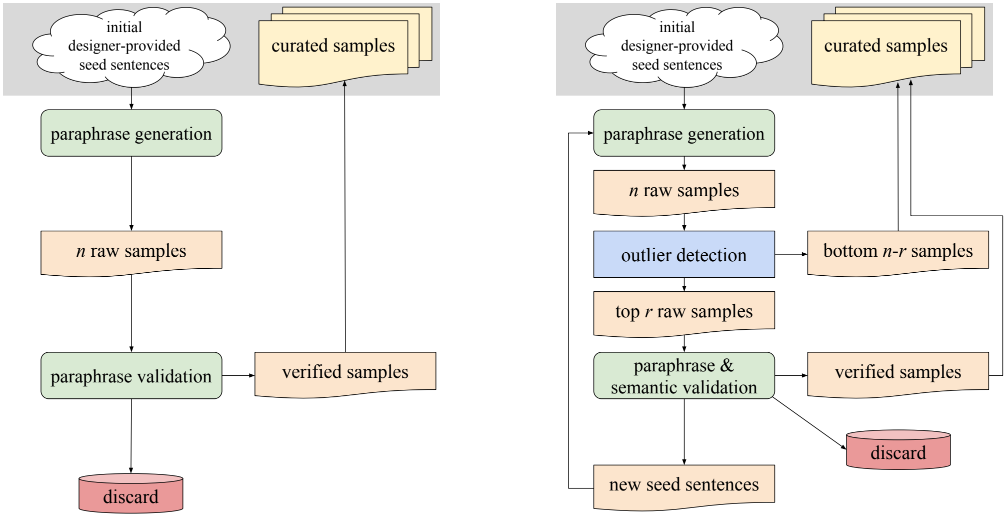
Figure 1a shows a common crowdsourcing pipeline. The task designer writes seed sentences that target an intent (for classification) or a slot (for slot-filling). Crowd workers read the seed and write paraphrases. These paraphrases are then passed to another set of workers who validate if they are in fact accurate paraphrases.

There are two major downsides to this standard pipeline. First, the validation step increases the cost-per-example. Second, the diversity of paraphrases depends on the given seed sentences (Jiang et al., 2017), creating a challenge for the task designer to think creatively.

We introduce a new pipeline, shown in Figure 1b that uses outlier detection to (1) reduce the number of sentences being checked, and (2) collect more diverse examples. Our new approach

---

[1] We found similar results in experiments with a density-based metric, Local Outlier Factor (Breunig et al., 2000).



(a) Standard Pipeline (b) Our Uniqueness-driven Pipeline

Figure 1: Data collection pipelines. Our outlier detection method is incorporated into the uniqueness-driven data collection pipeline to guide crowd workers to write more diverse paraphrases. Boxes with rounded corners are manual processes performed by crowd workers, boxes with curved bases are data, and the box with square corners is our outlier detection method. In (b), r is the number of outliers detected from n samples.

uses outlier detection to select only a subset of sentences to be checked: namely, the ones ranked as most likely to be outliers. This reduces effort by focusing on the sentences most likely to be incorrect.

To try to increase diversity, we also introduce a process with several *rounds* of data collection. Outlier paraphrases collected in one round are used to seed the next round of data collection. We could directly use the sentences labeled as correct in the validation step, but while these sentences are correct, they may have diverged from the desired semantics (e.g. diverged from the desired intent class). To avoid confusion in the next round, we add a step in which workers are shown the most similar sentence from another intent (based on sentence embedding distance) and asked if the new seed is more similar to its intended intent or the alternative example. Only seeds judged as closer to their intended intent are retained.

This iterative process is intended to collect more diverse data by priming workers to think about ways of phrasing the intent that are not well covered in the current data. At the same time, we avoid the correctness issues Jiang et al. (2017) observed by incorporating the validation step.

4 Experiments

We perform two sets of experiments to probe the effectiveness of our outlier detection method. First, we consider error detection, comparing various ranking methods in artificial and real data scenarios. Second, we use our uniqueness-driven pipeline to collect data, measuring the impact on data diversity and model robustness. All experiments were conducted on English language data.

4.1 Error Detection

We measure error detection effectiveness in two settings, one artificial and the other more realistic.

Artificial First, following prior work, we consider an artificial dataset in which we inject noise by mixing data from different intents (Guthrie et al., 2007, 2008; Zhuang et al., 2017; Amiri et al., 2018). This provides an easy way to control the amount and type of anomalous data, but does lead to an easier task as the incorrect examples are generally more different than naturally collected errors would be. The specific data we consider is a set of 20 intents from an in-production dialog system. To generate outliers for a given intent class X_i, we randomly sample $p \cdot |X_i|$ sentences from other intents (e.g. $p = 0.04$, or 4%).

Real We collected a new set of sentences for ten intents. Workers were given three seed sentences per intent and asked to write five paraphrases per seed.[2] Each seed was given to 15 crowd workers, leading to 2250 samples overall, after which

---

[2] Crowd workers were paid 20¢ per HIT.

Method		MAP					Recall@k=10%				
		1%	2%	4%	8%	Real	1%	2%	4%	8%	Real
Baseline	Random	0.02	0.04	0.07	0.09	0.06	0.06	0.09	0.12	0.09	0.04
	Short	0.02	0.05	0.07	0.11	0.27	0.12	0.13	0.12	0.15	0.27
	Long	0.08	0.10	0.08	0.17	0.04	0.25	0.17	0.11	0.12	0.04
	BoW	0.14	0.14	0.14	0.24	0.10	0.41	0.24	0.24	0.22	0.10
Neural	Average GloVe	0.16	0.17	0.22	0.26	0.33	0.55	0.41	0.49	0.39	0.33
	Average ELMo	0.20	0.19	0.23	0.32	0.32	0.48	0.48	0.45	0.42	0.32
	SIF GloVe (SG)	0.35	0.29	0.41	0.45	0.21	0.75	0.64	0.62	0.54	0.20
	SIF ELMo (SE)	0.30	0.37	0.48	0.52	0.20	0.87	0.74	0.70	0.65	0.19
	USE (U)	0.46	0.50	0.63	0.69	**0.37**	0.84	0.83	0.80	0.74	**0.38**
Combined	U+SG	0.51	0.54	0.66	0.68	0.36	0.84	0.81	0.83	0.76	0.33
	U+SE	**0.58**	**0.62**	**0.68**	**0.72**	0.34	**0.89**	**0.87**	**0.86**	**0.81**	**0.38**
	U+SE+SG	0.51	0.57	0.67	0.69	0.36	0.87	0.84	0.84	0.78	0.34

Table 1: Outlier detection effectiveness for error detection in an artificial setting (constructed by randomly adding content from other intents) and a real setting (manually checked utterances from a crowdsourced set). The artificial results are represented by different values of p (1%, 2%, 4%, and 8%), where p represents different amounts of errors injected into each intent class. Our proposed neural methods are consistently more effective, reducing the manual effort required to identify errors.

Intent	Label	Example
withdrawal	*seed*	what is my withdrawal limit?
	inlier	how high is my withdrawal ceiling?
	error	How much money do I have available
balance	*seed*	what's my balance?
	inlier	Let me know how much money I have.
	error	What can I afford?
phone	*seed*	what's my bank's phone number
	inlier	I need to call my bank
	error	information on my bank
checks	*seed*	i need to order more checks
	inlier	I need to stock up on more checks
	error	No checkbox, more?

Table 2: Examples from the Real dataset. The "How much money do I have available" example was labeled an error since it is too similar to the balance intent. The "What can I afford?", "information on my bank", and "No checkbox, more?" examples are labeled as errors since they are too vague and ambiguous.

duplicates were discarded. To identify errors, the authors independently checked each sentence and discussed any disagreements to determine a consensus label (either *inlier* or *error*). Examples of seed sentences, along with inliers and errors is shown in Table 2.

4.1.1 Evaluation Metrics

Since our core outlier detection method produces a ranked list, we are interested in evaluating how effective it is at ranking errors near the top. We use Mean Average Precision (MAP) as an overall measure of list quality. MAP is the mean over intents of:

$$\frac{1}{|\text{errors for intent}|}\sum_{e\,\in\,\text{errors}}\frac{|\text{errors at or above } e|}{e}$$

where e is the position of an error in the list.

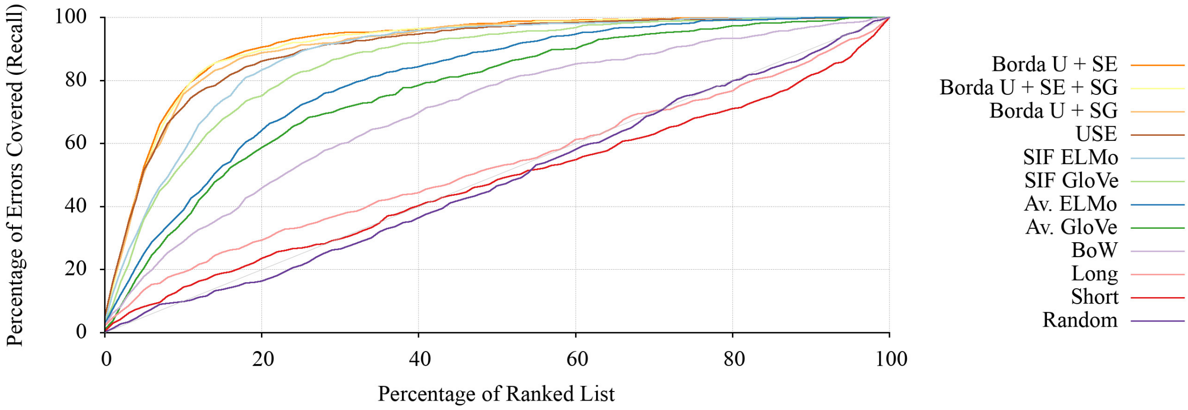
While this gives an overall qualitative measure for comparison, we are also interested in understanding the precision–recall tradeoff when choosing a threshold k on the ranked lists. We consider defining the cutoff as a percentage k of the list and measure the percentage of errors that are covered for each possible cutoff. This measure is equivalent to Recall@k, that is,

$$\text{Recall@}k = \frac{|\text{ errors above } k|}{|\text{errors}|}.$$

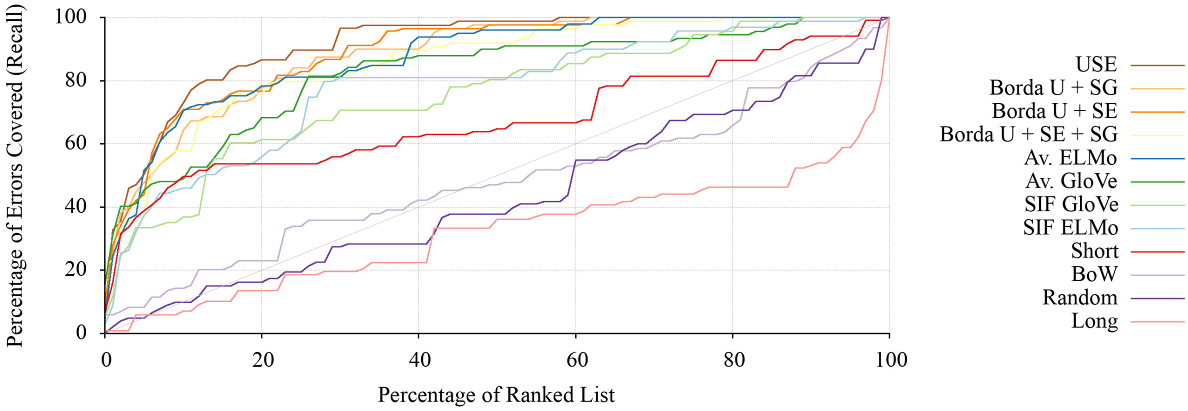
We average these values across intents to get an overall value for each cutoff percentage k.

4.1.2 Baselines

For comparison, we consider four simple baselines: randomly ordering the samples (Random), sorting from shortest to longest (Short), sorting from longest to shortest (Long), and calculating distances in the vector space defined by a bag of words (BoW).



(a) Artificial data.



(b) Real data.

Figure 2: Cumulative distribution of errors in ranked lists: a higher line indicates that the method places more errors earlier in the list. Results are less smooth for the real data as there are only 51 errors in the set. For each plot, the legend is in the same order as the lines at 20% (i.e., in (a) the top line is Borda U+SE, while in (b) it is USE). Neural ranking methods are consistently more effective, with USE covering over 80% of errors in the first 20% of the list.

4.1.3 Results

Table 1 presents MAP and Recall@k for error detection in the two settings (Artificial and Real). The neural methods outperform the baselines in both settings, demonstrating the effectiveness of our proposed approach. However, the relative performance of the neural methods differs substantially between the two settings. Specifically, (1) SIF performs better than an unweighted average on artificial data, but on real data we see the opposite trend, (2) combining rankings with Borda appears to help on the artificial data, but not on the real data, (3) ranking by length is surprisingly effective on the real data, and (4) results tend to be lower on the real data than the artificial (even at lower values of p). This last point suggests that the commonly used artificial setting does not perfectly capture the types of errors that occur in practice.

show average exchange rate from ten usd to cad last year

Figure 3: Example annotated sentence for the slot-filling task. The slot names are (in order of appearance) metric, amount, currency, and date.

For (3), we note that the Short baseline method performs particularly well vis-à-vis other baselines on the real data, but not comparatively well on the artificial data. This can be explained by observing that the length of the average error in the real data is roughly 6 tokens, while the average inlier length is 8 tokens. Lengths of errors and inliers are roughly the same (roughly 8 tokens) in the artificial dataset, due to the outlier selection scheme.

While the values in Table 1 allow an overall comparison of the methods, they do not provide a clear qualitative sense of the distribution of errors in the lists. Figure 2 shows the distribution for each method in the two settings. The effective-

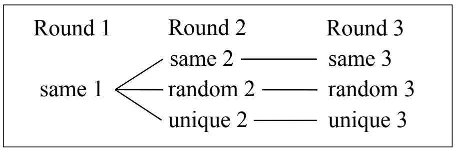


Figure 4: Data collection rounds. The final datasets combine data from all three rounds along each path.

ness of the neural methods, and USE in particular, is again clear. In the real data, when considering just the first 20% of the list, USE covers over 85% of the errors on average. One easy example was "*No checkbox, more?*" when the intent was to order more checks. This is clearly an error, which would at the very least need to have *checkbox* replaced by *checkbook*. In contrast, one hard example for USE was "*How much money do my banks*" when the intent was to request the user's balance. Until the last word, this example looks like it will be a valid balance request. These examples show that the system is qualitatively fitting our expectations for error detection.

4.2 Uniqueness-driven Data Collection

The second set of experiments evaluates our proposed uniqueness-driven data collection pipeline. We consider collecting data for two tasks used by dialog systems: intent classification and slot-filling. In each case, we calculate intrinsic measures of data diversity and the robustness of models trained on the data.

Tasks We consider intent classification with 10 intents related to banking, and slot-filling for foreign exchange rate requests with four slots: *amount*, *currency*, *date*, and *metric*. Figure 3 shows an example query with annotated slots.

Approaches As well as our proposed data collection pipeline (`unique`), we consider a variant where the next seed is chosen randomly (`random`), and one where the seeds are the same in every round (`same`). The third case is equivalent to the standard pipeline from Figure 1a. All three pipelines start from the same first round and then vary in the subsequent rounds, as depicted in Figure 4. Each pipeline collected data for three rounds. The final dataset for each approach combines data collected from all three rounds.

In both tasks, we asked workers to rephrase each seed sentence 5 times and showed each seed sentence to 15 workers. For classification there were 3 seed sentences per intent. For slot-filling we defined 4 example scenarios, each corresponding to a specific combination of slots. We used Borda USE+SG with k set to 10% for the outlier detection model.

$$D(a,b) = 1 - \frac{1}{N}\sum_{n=1}^{N}\frac{|n\text{-grams}_a \cap n\text{-grams}_b|}{|n\text{-grams}_a \cup n\text{-grams}_b|}$$

$$\text{Diversity}(X) = \frac{1}{|I|}\sum_{i=1}^{|I|}\frac{1}{|X_i|^2}\left[\sum_{a\in X_i}\sum_{b\in X_i} D(a,b)\right]$$

$$\text{Coverage}(X,Y) = \frac{1}{|I|}\sum_{i=1}^{|I|}\frac{1}{|Y_i|}\sum_{b\in Y_i}\max_{a\in X_i}(1 - D(a,b))$$

Figure 5: Metrics for diversity and coverage from Kang et al. (2018). X and Y are sets of utterances labeled with intents from I, and X_i is the data in X for intent i. The distance metric for comparing a pair of utterances is based on the Jaccard Index over n-grams. We follow the work from Kang et al. (2018) and set N, the max n-gram length, to 3.

4.2.1 Evaluation Metrics

We consider several different metrics to probe how effectively our proposed pipeline improves data quality. In all cases, higher values are better.

Intrinsic We measure the diversity and coverage of each dataset using the metrics introduced in (Kang et al., 2018) and shown in Figure 5.

Extrinsic The main reason to increase dataset diversity is to construct more robust models. To directly evaluate that objective, we randomly divided the datasets collected by each pipeline into training and test sets (85-15 split). Our intuition is that a robust model should perform fairly well across all test sets. Training on a dataset that is not diverse will lead to a brittle model that only does well on data collected with the same seed sentences. For intent classification, we measure accuracy of two models: an SVM (Cortes and Vapnik, 1995) using bag of words feature representation, and FastText (Joulin et al., 2017), a neural network that averages across sentence embeddings and passes the result through feedforward layers. For slot-filling, we measure the F_1-score of a bi-directional LSTM with word vectors that are trained, but initialized with GloVe 300-dimensional embeddings. For all models, we average results across 10 runs.

	Data Collection Round			
	1	2	3	All
Diversity				
`same`	0.911	0.917	0.908	0.916
`random`	–	0.912	0.817	0.920
`unique`	–	**0.935**	**0.944**	**0.947**
Samples				
`same`	2040	2025	2024	5648
`random`	–	2010	2007	5747
`unique`	–	2083	1954	5922

Table 3: Classification: Diversity scores for data collected in each round (top), and the number of samples collected (bottom). The data for the All column combines the previous two sets in the row and the data from (`same` round 1). The `unique` approach produces data that is considerably higher diversity.

		Test Set		
Metric	Training	`same`	`random`	`unique`
SVM Accuracy	`same`	**0.99**	0.97	0.81
	`random`	0.98	**0.98**	0.81
	`unique`	0.99	0.97	**0.98**
FastText Accuracy	`same`	**0.98**	0.97	0.80
	`random`	0.98	**0.99**	0.83
	`unique`	0.98	0.98	**0.98**
Coverage	`same`	**0.68**	0.64	0.45
	`random`	0.67	**0.66**	0.44
	`unique`	0.64	0.58	**0.56**

Table 4: Classifier accuracy when training on one dataset and testing on another (top and middle), and coverage of the test set for each training set (bottom). As expected, the highest scores are when we train and test on the same data, but off the diagonal the `unique` test set (gray column) is considerably harder for models trained on other data while a model trained on `unique` performs consistently well. This accuracy trend is matched in coverage.

	Data Collection Round			
	1	2	3	All
Diversity				
`same`	0.916	0.911	0.893	0.909
`random`	–	0.913	0.910	0.915
`unique`	–	**0.926**	**0.935**	**0.930**
Samples				
`same`	994	911	952	2717
`random`	–	941	923	2808
`unique`	–	977	988	2914

Table 5: Slot-filling: Diversity scores for data collected in each round (top), and the number of samples collected (bottom). The data for the All column combines the previous two sets in the row and the data from (`same` Round 1). As seen for intent classification, the `unique` approach produces data that is of considerably higher diversity.

		Test Set		
Metric	Training	`same`	`random`	`unique`
Slot F_1	`same`	96.4	96.0	93.1
	`random`	96.4	**96.8**	93.6
	`unique`	**96.7**	96.5	**94.9**
Coverage	`same`	**0.812**	0.788	0.726
	`random`	0.736	**0.764**	0.660
	`unique`	0.761	0.752	**0.774**

Table 6: F_1-scores and coverage scores for each train-test pair for the slot-filling experiment. Training on the `unique` data produces a more robust model, with consistently high performance across test sets.

4.2.2 Results

Classification Table 3 presents the number of examples and diversity of data collected in each round with each approach. Diversity is consistently higher with seeds chosen using our proposed `unique` approach. Dataset sizes vary because of the removal of duplicates. The `unique` approach produces a larger final set as there is less duplication across rounds.

Table 4 displays accuracy scores and coverage for each combination of train and test sets. As expected, the highest scores are on the diagonal—training and testing on the same source data. More importantly however, training on the `unique` data produces a model that is robust, performing well across all three test sets. In contrast, training on the `same` or `random` data produces models that perform substantially worse on the `unique` test set. This trend is also present in the coverage scores in the bottom section of the table.

Table 7 shows some of the seed sentences produced by the `unique` and `random` approaches. These examples illustrate the trends in our metrics, with the seeds for the `random` approach often being very similar. Meanwhile, the `unique` approach produces seeds with grammatical variation and the introduction of quite different expressions, such as “ABA” instead of “routing number”.

Intent	Pipeline	Round 2	Round 3
routing	`random`	Where can I find the routing number for my bank? Do you have my bank's routing number? show me my banks routing number	what is the best routing number for my bank can you help find my bank routing number? I need to see the routing number for my bank
	`unique`	acquire my banks routing number how does a person find their correct routing number? I'm looking for the router number for my bank.	what is the correct ABA? How do I find the ABA? Whats the router number?
hours	`random`	Can you tell me when my bank is open? When does my bank open? What time is the bank open	How long is my bank open? What are the hours for my bank? What time will the bank be open
	`unique`	display when the bank closes What is the earliest you are open? look up the hours of operation for my bank	What hours do you carry what are your operating hours? What is the latest I can come in to a physical branch?
checks	`random`	i require ordering more checks. Get me more checks. can you explain to me how to order additional checks	I'd like additional checks Can you get me more checks? Please order more checks for me.
	`unique`	I need to stock up on more checks in what manner would i get more checks Could you rush me some more checks? I'm nearly out.	what is check ordering procedure? what is the fastest method to order checks? Teach me how to get more checks.

Table 7: Seed sentences for selected intents for the classification task. The `unique` approach leads to changes like the use of *ABA* for *routing* (top), and grammatical variations in the sentences for requesting checks (bottom). Examples of seeds for Round 1 include "what is my bank's routing number?", "when does the bank close?", and "how do i order more checks?".

Slot-filling Table 5 shows the number of samples collected per round for each of the data collection pipelines and the diversity of the sets. As in the classifier experiment, we observe that data produced by the `unique` pipeline is of higher diversity than the other two pipelines.

Table 6 displays F_1-scores and coverage for each train–test combination. Again, we see the same trends, with training on `same` or `random` leading to low results on the `unique` dataset, but not the reverse, and similarly for coverage, though the gaps are smaller than for classification.

5 Conclusion

Outliers are often the most interesting parts of our data, but outlier detection has received relatively little attention in NLP beyond its application to finding annotation errors. This paper introduces the first neural outlier detection method for short text and demonstrates its effectiveness across multiple metrics in multiple experiments.

We also propose a way to integrate outlier detection into data collection, developing and evaluating a novel crowdsourcing pipeline. This pipeline supports the creation of higher quality datasets to yield higher quality models by both reducing the number of errors and increasing the diversity of collected data. While the experiments discussed herein are concerned with components of dialog systems, we believe that similar data collection strategies could yield benefits to other areas of NLP as well.

Acknowledgments

The authors thank Yiping Kang, Yunqi Zhang, Joseph Peper, and the anonymous reviewers for their helpful comments and feedback.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and pp attachment. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Charu C. Aggarwal. 2015. *Outlier Analysis*, pages 237–263. Springer International Publishing, Cham.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Trans. Intell. Syst. Technol.*, 4(3):43:1–43:21.

Olivia Buzek, Philip Resnik, and Ben Bederson. 2010. Error driven paraphrase annotation using mechanical turk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Computing Research Repository (CoRR)*, abs/1805.10190.

Markus Dickinson. 2010. Detecting errors in automatically-parsed dependency relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Markus Dickinson and Chong Min Lee. 2008. Detecting errors in semantic annotation. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*.

Markus Dickinson and W. Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Markus Dickinson and W. Detmar Meurers. 2005. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Markus Dickinson and Amber Smith. 2011. Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT)*.

Dmitriy Dligach and Martha Palmer. 2011. Reducing the need for double annotation. In *Proceedings of the 5th Linguistic Annotation Workshop (LAW)*.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Fumiyo Fukumoto and Yoshimi Suzuki. 2004. Correcting category errors in text classification. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*.

Tirthankar Ghosal, Amitra Salam, Swati Tiwary, Asif Ekbal, and Pushpak Bhattacharyya. 2018. TAP-DLND 1.0 : A Corpus for Document Level Novelty Detection. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*.

David Guthrie, Louise Guthrie, Ben Allison, and Yorick Wilks. 2007. Unsupervised anomaly detection. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*.

David Guthrie, Louise Guthrie, and Yorick Wilks. 2008. An unsupervised probabilistic approach for the detection of outliers in corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*.

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora*.

Nora Hollenstein, Nathan Schneider, and Bonnie Webber. 2016. Inconsistency detection in semantic annotation. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding task design trade-offs in crowdsourced paraphrase collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Yiping Kang, Yunqi Zhang, Jonathan K. Kummerfeld, Parker Hill, Johann Hauswald, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective.

In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit*.

Raksha Kumaraswamy, Anurag Wazalwar, Tushar Khot, Jude Shavlik, and Sriraam Natarajan. 2015. Anomaly detection in text: The value of domain knowledge. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Sungjin Lee. 2015. Online sentence novelty scoring for topical document streams. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*.

Matteo Negri, Yashar Mehdad, Alessandro Marchetti, Danilo Giampiccolo, and Luisa Bentivogli. 2012. Chinese whispers: Cooperative paraphrase acquisition. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

P. J. Rousseeuw and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ian Soboroff and Donna Harman. 2005. Novelty detection: The trec experience. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ye Zhang and Byron C. Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Xunjie Zhu, Tingfeng Li, and Gerard de Melo. 2018. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Honglei Zhuang, Chi Wang, Fangbo Tao, Lance Kaplan, and Jiawei Han. 2017. Identifying semantically deviating outlier documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Asking the Right Question: Inferring Advice-Seeking Intentions from Personal Narratives

Liye Fu
Cornell University
liye@cs.cornell.edu

Jonathan P. Chang
Cornell University
jpc362@cornell.edu

Cristian Danescu-Niculescu-Mizil
Cornell University
cristian@cs.cornell.edu

Abstract

People often share personal narratives in order to seek advice from others. To properly infer the narrator's intention, one needs to apply a certain degree of common sense and social intuition. To test the capabilities of NLP systems to recover such intuition, we introduce the new task of inferring what is the advice-seeking goal behind a personal narrative. We formulate this as a cloze test, where the goal is to identify which of two advice-seeking questions was removed from a given narrative.

The main challenge in constructing this task is finding pairs of semantically plausible advice-seeking questions for given narratives. To address this challenge, we devise a method that exploits commonalities in experiences people share online to automatically extract pairs of questions that are appropriate candidates for the cloze task. This results in a dataset of over 20,000 personal narratives, each matched with a pair of related advice-seeking questions: one actually intended by the narrator, and the other one not. The dataset covers a very broad array of human experiences, from dating, to career options, to stolen iPads. We use human annotation to determine the degree to which the task relies on common sense and social intuition in addition to a semantic understanding of the narrative. By introducing several baselines for this new task we demonstrate its feasibility and identify avenues for better modeling the intention of the narrator.

1 Introduction

> "Computers are useless.
> They can only give you answers." - Pablo Picasso

People often share their personal experiences to elicit advice from others. These personal narratives provide the necessary context for properly understanding the informational goals of the narrators. Endowing automated systems with the capability to infer these advice-seeking intentions could support personalized assistance and more empathetic human-computer interaction.

Personal narrative: I am generally a person who needs a lot of sleep, but today I was not able to sleep more than 6 hours and I am extremely tired. My eyes hurt and two hours later I have programming [lesson] so I have to be alert. I've already drunk a cup of coffee and although I rarely drink coffee, it had no effect on me. I am not at home so I have limited possibilities as for food. I don't want to do anything too unhealthy such as drinking 10 cups of coffee, tho I may consider drinking another one.

Which advice-seeking question is more likely to have been asked by the narrator:
Q1: Is it even possible to be addicted to coffee?
Q2: How can I energize myself?

Figure 1: An abbreviated instance from the `ASQ` dataset. A personal narrative is matched with two plausible advice-seeking questions, only one of which was actually asked by the narrator when sharing the story.

As humans, to properly distill the narrator's intention from the events and situations they describe, we need to apply a certain degree of social intuition (Conzelmann, 2012; Conzelmann et al., 2013; Baumgarten et al., 2015; Kehler and Rohde, 2017). As an example, consider the goals of a narrator sharing the personal story in Figure 1. We are presented with a wealth of information about the narrator's general sleep patterns, about a particular sleep deprivation situation and its physiological effects, about an upcoming lesson, about coffee intake, its effects, and potential health impacts, and about the current location of the narrator and its impact on food supply. Taking these facts separately, we can imagine providing advice on how to get more sleep, on whether to postpone the lesson,

Proceedings of NAACL-HLT 2019, pages 528–541
Minneapolis, Minnesota, June 2 - June 7, 2019.

	Task	**Desired output**
A	Question generation Reading comprehension	What do I need to do in 2 hours?
	Summarization	I must go for a lesson after getting little sleep.
B	Ending generation Narrative chains, story cloze	Lastly, I tried an energizing drink.
C	Event2Mind Desire fulfillment	to learn to code, to be educated
D	**Our task**	How can I energize myself?

Table 1: Contrast with desired outputs in related narrative understanding tasks that focus on the within-story (intradiegetic) aspects of narrative understanding. Tasks are grouped according to the categories discussed in Section 2.1 (which also includes corresponding references). We assumed the second sentence ("My eyes hurt and two hours later I have programming lesson so I have to be alert.") to be the answer span for question generation, and the input for Event2Mind (which operates at sentence level).

on how to get food delivered, or on the risks of caffeine intake. However, given how the narrative is constructed, we can intuit that the more likely goal of the narrator is to get advice on how to overcome the effects of sleep deprivation so that they can be alert for the upcoming programming lesson.

Importantly, the primary goal of our proposed task is not to understand details about the narrator's actions in the story ("Why is the narrator tired?", "When do they need to go to the lesson?"), but to infer the reason why the narrator is sharing this story (i.e., "To get advice on how to stay alert in the next few hours."). That is, we are not concerned with the *intradiegetic* aspects of the narrative, but with the *extradiegetic* intention of the narrator in sharing the story. While an understanding of the former is likely necessary for the latter, it is often not sufficient.

In this work, we introduce a task and a large dataset to evaluate the capabilities of automated systems to infer the narrator's (extradiegetic) intention in constructing and sharing an advice-seeking personal story. This complements existing narrative understanding tasks which focus on testing semantic understanding of events, actors and their (intradiegetic) intentions within the narrative itself. Table 1 contrasts the goals of these existing narrative understanding tasks with that of inferring a narrator's advice-seeking intention, in the context of our introductory example.

Formally, we implement the task as a binary choice cloze test, where the goal is to identify which of two candidate advice-seeking questions was actually asked by the narrator of a given personal narrative. Beyond collecting a large and diverse set of realistic personal stories that contain an advice-seeking question, the main challenge in constructing this task is finding a plausible alternative advice-seeking question for each given narrative. To address this challenge, we develop a methodology for identifying such questions by exploiting both the commonalities in experiences people share online and the diversity of possible advice-seeking intentions that can be tied to similar experiences.

By applying our methodology to a large collection of online personal narratives, we construct a dataset of over 20,000 cloze test instances, covering a very broad spectrum of realistic advice-seeking situations.[1] Each instance contains a narrative that is matched with two advice-seeking questions, one of which is actually asked by the narrator (Q2 in our introductory example), and the other semantically related to the narrative (Q1).

We use human annotations to judge the relative difficulty of different subsets of the test instances and the type of reasoning necessary to solve them. We find that more than half of the instances contain pairs of questions that are not only semantically related to the narratives but also do not contain any explicit factual mismatches with the stories. These are thus unsolvable by pure logical reasoning and require some degree of common sense or social intuition. And indeed, simple

---

[1] The dataset is available at `https://github.com/CornellNLP/ASQ`.

baseline approaches perform worse on these types of instances, highlighting the need for more direct modeling of the intention of the narrator.

To summarize, in this work we:

- formulate the task of inferring advice-seeking intents from personal narratives (Section 2);
- develop a methodology to construct a large dataset of personal narratives matched with plausible options for **a**dvice-**s**eeking **q**uestions (the `ASQ` dataset) to be used for this task (Section 3);
- show the task is viable and evaluate the relative difficulty of its items (Sections 4 & 5).

We end by discussing the practical implications of endowing systems with the capability to infer advice-seeking intentions and use our results to identify avenues for developing better models.

2 Task formulation

To evaluate the capability of automated systems to infer advice-seeking intentions, we formulate a cloze-style binary choice test where the system is presented with a personal narrative and is required to choose between two plausible candidate questions: one actually asked by the narrator and the other one not (as exemplified in Figure 1).

We motivate the task by contrasting it with other (narrative) understanding tasks (Section 2.1), and provide the rationale for this particular formulation by discussing its advantages (Section 2.2).

2.1 Related narrative understanding tasks

There are many tasks involving reading comprehension in general, and story understanding in particular. Given a narrative, there are a few broad categories of questions that may be asked to test different types and degrees of understanding. Table 1 follows directly from the discussion below, by contrasting the goal of our task with those of (intradiegetic) narrative understanding tasks in the context of our introductory example.

A: What happened in the story? The most direct approach to test story understanding is to check whether the reader could comprehend the events and actions that occur within the story. This requires semantic understanding, but nothing more. This type of task can be set up in various forms, as the system can be asked to summarize the story (summarization, see Nenkova (2011); Allahyari et al. (2017) for surveys), generate a question that is answerable from the text (question generation (Du et al., 2017)), or answer a question for which the information can be retrieved or reasoned directly from the story (reading comprehension, see Chen (2018) for a survey; notable datasets include MCTest (Richardson, 2013) and NarrativeQA (Kočiský et al., 2018)).

B: What might happen next? While reading the story, people not only grasp and process the events that already occurred but also have some intuition of its likely trajectory. Related tasks include the narrative cloze task (Chambers and Jurafsky, 2008), the story cloze test (Mostafazadeh et al., 2016; Chaturvedi et al., 2016), and its generative versions (Guan et al., 2019). These tasks might require some common sense reasoning on top of semantic understanding; the fact that they aim to predict the future might require a deeper level of understanding than the previous tasks.

C: What can we infer about the characters? When people read a narrative, they not only grasp the facts explicitly stated in the story, but also make inferences about the actors' mental states, such as their attitudes and desires, as the story unfolds.[2] Oftentimes, such an understanding requires inference, either logical or based on common sense reasoning. Such tasks can aim to generate the likely intents and reactions from the actors involved in the events (Rashkin et al., 2018a,b), or to determine whether a given desire of the protagonist was fulfilled (Rahimtoroghi et al., 2017).

D: What is the intention of the narrator in sharing their story? While these prior tasks cover a wide range of angles to narrative understanding, they take an intradiegetic view by focusing on understanding the story itself. We propose another dimension to this line of work by taking an outside-the-story (extradiegetic) perspective[3] and aiming to understand *why* the story is shared by the narrator, potentially inferred from *how* the narrator decides to construct it. In particular, the task introduced here is to infer the *advice-seeking* intention of the narrator.[4]

---

[2] See Bratman (1987) for an account of the Belief-Desire-Intention model of human practical reasoning.

[3] Recognizing the importance of these two different perspectives for story understanding, Swanson et al. (2017) attempted to classify narrative clauses into intradiegetic vs. extradiegetic levels.

[4] Sharing personal stories can have other goals, e.g., therapeutic (Pennebaker, 1997; Pennebaker and Seagal, 1999).

We argue that solving this task requires not only the semantic understanding and common sense reasoning involved in prior tasks but also a certain degree of social intuition. To uncover the goals of the narrator, one needs to find cues in the narrative construction—what has been selectively included or emphasized, and what might have been purposefully omitted (Labov, 1972). In fact, such intention-understanding tasks are often included in "social intelligence" tests (Conzelmann et al., 2013; Baumgarten et al., 2015).

2.2 Advantages of cloze test formulation

To evaluate the capacity of NLP systems to solve this task, we consider a binary choice cloze test formulation for two main reasons. First, it allows natural ground-truth labels: often, when people share their personal experiences to seek advice, they add explicit requests for the information they are seeking. After removing these requests from the narratives, we can use them as proxies for the narrators' intentions. Second, the binary choice operationalization also has the advantage of non-ambiguity in evaluations and ease of comparisons between systems (as opposed to a generation task).

It is worth noting that our dataset is constructed in a way that allows easy modifications into other task formats if so desired. For instance, the methodology of identifying a plausible false choice for a given narrative could be applied multiple times to extend the task to a more difficult multiple-choice version. Similarly, by ignoring the incorrect question in each instance, our dataset can be used as a source for a new generation task, i.e., generating the advice-seeking question from the given narrative.

3 Task implementation

For a meaningful implementation of the proposed task, the collection of test instances must conform to several expectations, in terms of both the narratives and their (actual) advice-seeking questions. In what follows we outline these desiderata and our method for collecting instances that meet them (Section 3.1).

Furthermore, as with any multiple-choice cloze test formulation, the difficulty of each test instance largely depends on how plausible the alternative answers are. Yet, finding plausible (but not actually correct) alternatives automatically is challenging. Not surprisingly, many of the cloze-style multiple-choice datasets use humans to write these alternatives (Mostafazadeh et al., 2016; Xie et al., 2018), limiting their scalability.

We tackle this challenge by developing a methodology that exploits both the commonalities in human experiences shared online and the diversity in the types of advice needed for similar situations under different circumstances (Section 3.2).

3.1 Collection of candidate instances

Narratives desiderata. As a pre-requisite, we need to start from personal narratives containing advice-seeking needs that are explicitly expressed (as questions), and that can be removed to form the cloze test instances.[5] Ideally, these narratives would cover a broad range of topics, in order to be able to test how well a system can generalize to a diverse range of real-life scenarios, rather than apply only to restricted and artificial settings.

Question desiderata. Not all questions contained within an advice-seeking narrative are suitable for our task. Some of the questions might be too general, while others might be rhetorical. For instance, *Any advice?* holds no particular connection with the context of the narrative in which it appears. To contribute to meaningful test instances, questions need to meet a level of relevance and specificity such that (at least) humans could match them with the narratives from which they are extracted.

Data source. We start from a dataset of over 415,000 advice-seeking posts collected from the subreddit r/Advice, which self-defines as "a place where anyone can seek advice on any subject".[6] We only use publicly available data and will honor the authors' rights to remove their posts.

Applying cloze. For each post, we strip off all questions that appear in any position of the post, including the post title.[7]

We keep the remaining narratives as the cloze texts.[8] Figure 2 shows how the cloze transforma-

---

[5] An interesting future work avenue could be considering narratives that only have implicit advice-seeking intentions.

[6] We start from an existing collection of Reddit posts (Tan and Lee, 2015) which we supplement with The Baumgartner Reddit Corpus retrieved via Pushshift API on Nov. 21, 2018.

[7] To identify questions, we use the simple heuristic of looking for sentences that end with '?' or start with *why, how, am, is, are, do, does, did, can, could, should, would.*

[8] To ensure that the cloze text can provide sufficient context, yet are not overly verbose, we only consider cloze texts that are 50-300 tokens long. This is a choice we made prior to any experiments, and we do not claim it is the optimal range to set up the task.

Selected topics	Question keywords	Example questions
Housing	move live house city apartment roommate	What is it like living with **roommates**? Should I **move** to the **city**?
School	college school class degree study	Should I drop out of **college**? What's the best way for me to **study** for my biology tests?
Work	job boss quit work interview employer	Can I somehow ask to **work** from home? How do I explain during an **interview** why I left a **job**?
Relationships	girl date text tell guy think crush	Does it sound like this **girl** may like me? How can I think of a better greeting for online **dating**?
Personal finances	money car pay rent loan insurance	How do I afford a **car** in my situation? Am I stupid for wanting a student **loan**?
Family	parent convince let mom dad sister	How do I **convince** my **parents** to believe me? How can I try and make a better relationship with my **sister**?

Table 2: Selected narrative topics and example question keywords associated with each topic.

Title: How can I energize myself?

I am generally a person who needs a lot of sleep [...] I don't want to do anything too unhealthy such as drinking 10 cups of coffee, tho I may consider drinking another one. Help? What has worked for you?

Figure 2: Cloze application to the post from which we obtain the introductory test instance. After filtering out questions that are too general, only the title question remains as a candidate for representing the actual advice-seeking intention of the narrator.

tion is applied to the post containing our introductory example.

Selecting ground-truth test answers.[9] We select candidate ground-truth answers for the cloze test as the ?-ending sentences removed from narratives. In order to keep only well-formed information-seeking questions, we filter the candidate questions by keeping only those that start with interrogatives[10] or *any, anyone, help, advice, thoughts*. To further discard questions that are too general, we compute a simple specificity score $S(q)$ of a question q containing the set of words $\{w_1, w_2, \ldots, w_N\}$ as its maximum inverse document frequency (idf):

$$S(q) = S(\{w_1, w_2, \ldots, w_N\}) = \max_{i \in N} \text{idf}(w_i),$$

and filter out questions for which $S(q) < 5$ or questions that have less than 5 words. At the end of this selection process, from the example post in Figure 2, *Help?* and *What has worked for you?* are discarded and the title question is kept as the ground-truth answer to this cloze instance. If multiple questions survive the filtering process, we select one at random.

Diversity evaluation. To verify that the resulting data has broad topical diversity in both narratives and questions, we perform a two-step clustering analysis. First, we use singular value decomposition on tf-idf transformed narratives to obtain their vector representations, we then cluster similar narratives using k-means to surface underlying topics. Next, for each topic, we extract nouns and verbs from the questions attached to each narrative in the topic, and surface common question keywords as those with high document frequency within the topic, correcting for their global document frequency (via subtraction).

To provide a qualitative feel of the diversity of the data, Table 2 shows a selection of the resulting narrative topics and question keywords, together with example questions (corresponding narratives can be found in the data release). We find a

---

[9] As it happens, test answers are actually questions.

[10] We consider the following set of words as interrogatives: *what, when, why, where, which, who, whom, whose, how, am, is, are, was, were, do, does, did, has, have, had, can, could, shall, should, will, would, may, might, must*.

wide range of experiences represented in the narratives, from relationships to student life to apartment rentals. Furthermore, within each narrative topic, there is a variety of question types; for instance, questions related to housing could be about dealing with roommates, paying rent, or choosing a city to live in.

3.2 Finding alternative test answers

To find plausible alternative answer options for each candidate cloze test instance, one direct approach could be to find questions that are semantically related to the ground-truth question. However, there are two underlying problems with this approach. First, the task of finding semantically similar questions is itself very challenging (Haponchyk et al., 2018), given their terseness and lack of context. Second, semantic similarity is arguably a different concept from *plausibility* with respect to a narrative. For example, the two questions in the introductory example are semantically distant, but they are both plausible in the context of the narrative.

Our main intuition in solving this problem is that individuals who are in similar situations tend to have advice-seeking intentions that are related. For each candidate cloze test narrative instance, we can thus search for a similar narrative first (by exploiting commonalities in experiences people share online) and then select an advice-seeking question from that narrative as the *alternative* answer for the test.

Narrative pairing. To operationalize this intuition, we first find pairs of similar narratives based on the cosine similarity of their tf-idf representations.[11] A greedy search based on this similarity metric results in a set of pairs of related narratives (N_1, N_2) with their respective advice-seeking questions (qn_1, qn_2) identified in the previous step.

Narrative masking. At this point, the pair of advice-seeking questions could be used with either narrative to form a test instance. For example, Figure 3 shows the other possible cloze instance corresponding to the introductory example if we were to use the other narrative in the narrative pair. This, however, would arguably be a poor test instance since Q2 is hardly applicable to this other narrative. More generally, we want to ensure that our choice of which narrative (N_i) to include in the cloze test optimizes the plausibility of the question pair (qn_1, qn_2).

Masked narrative: I've noticed something, over the past few years I've gained a habit of drinking coffee. The average day is about six cups, but it can exceed that sometimes (8 or so). The only reason I question my habit is cause I'm up at 4AM right now cause I couldn't fall asleep. I honestly have a headache in the morning until I drink a cup of coffee. I'll have some for essentially no reason, I'll just make some out of a urge almost.

Q1: Is it even possible to be addicted to coffee?
Q2: How can I energize myself?

Figure 3: Alternative cloze test instance corresponding to the introductory example.

To achieve this, we compute the similarity between each narrative in the pair and each of the two respective questions,[12] and select the narrative that maximizes the minimum question-narrative similarity. Formally,

$$N_i = \arg\max_i \textsc{min}\{sim(N_i, qn_1), sim(N_i, qn_2)\}.$$

Importantly, this selection criterion is purposely symmetric with respect to the two questions in order to avoid introducing any unnatural preference between the two that a classifier (with no access to the masked narrative) could exploit.

As a final check, we ensure that in each cloze instance the two questions are neither too similar to each other (and thus indistinguishable) nor too dissimilar (which may indicate unsatisfactory narrative pairings). To this end, we discard instances in which the questions have extremely high or low surface similarity according to their InferSent (Conneau et al., 2017) sentence embeddings.[13]

This process leaves us with a total of 21,865 instances. A detailed account of the number of instances filtered at different stages of the construction process can be found in the Appendix.

---

[11] We consider both unigrams and bigrams, and set a minimum document frequency of 50. We also remove likely duplicates (cosine > 0.8) and cases for which the similarity between narratives is too low (cosine < 0.1). We have also experimented with embedding-based representations to compute cosine similarities from, but they do not seem to produce qualitatively better pairings upon inspection.

[12] To account for the terseness of the questions, we represent both narratives and questions with tf-idf weighted GloVe embeddings (Pennington et al., 2014) and compute the cosine similarity between them.

[13] We set a lower bound of 0.8 and an upper bound of 0.95. We choose this representation because questions are short and thus we anticipate tf-idf representation to be less informative.

4 Human performance

To understand the feasibility of the task, as well as the relative difficulty of the items in the dataset, eight non-author annotators labeled a random sample of 200 instances.[14] Each annotator is asked to choose first, out of the two candidate questions, which they consider to be *more likely* to have been asked by the narrator. Overall, human annotators achieve an accuracy of 90% (Cohen's κ = 0.79),[15] showing that humans can indeed recover the advice-seeking intentions of the narrators, and thus validating the feasibility of the task.[16]

We are also interested in understanding the types of skills needed to solve the task. In particular, we want to estimate the proportion of the task instances that can not be solved by mere factual reasoning. To this end, we ask humans to identify candidate questions that contain a factual mismatch with the narrative, making them **E**xplicitly incompatible; 57% of the annotated instances do not contain any such mismatches in any of the questions. Similarly, we want to estimate how many instances require common sense expectations about the behavior of the protagonist (within the story). So we ask annotators to mark questions as being **I**mplicitly incompatible if they do not contain any factual mismatches, but they are incompatible with what can be inferred implicitly about events and characters in the story.

The questions that are neither explicitly nor implicitly incompatible would be labeled as being **C**ompatible, and as either **L**ikely or **U**nlikely to represent the narrators' intentions. Test items in our data forcing a choice between **C**ompatible questions are expected to be the hardest to solve, as they might require a certain degree of social intuition in addition to factual and common sense reasoning. Table 3 provides an example narrative and one representative question from each of the above-mentioned categories.[17]

Table 4 shows a human performance breakdown according to some of the most common types of instances in our data.[18] As expected, instances involving only compatible questions (C + C) are harder to solve,[19] as they might require some social intuition, whereas when explicit contradictions exist (C + E), they are perfectly solvable. We also note that humans can perfectly solve the subset of task instances (L + {U, I}) that exhibit perceived qualitative differences between the actual and the alternative questions, but nevertheless, require more than semantic understanding (and sometimes require social intuition).

Narrative: I asked a girl that I really like if she would like to get coffee sometime. She said she's really busy but that we'll see. I can't get her off my mind and I spend all day waiting for her to tell me she's free.

Explicitly incompatible (E):
How to deal with my roommate?

Implicitly incompatible (I):
What to do if I asked a girl out and now regret it?

Compatible (C) but unlikely (U):
Which coffee place would you recommend?

Compatible (C) and likely (L):
Would it seem desperate if I asked her again in a more direct way a week later?

Table 3: Example questions in each plausibility category for an example narrative.

Pair type	SIM	FT-LM	HUMAN	% in data
C + E	86%	88%	100%	38%
C + {C, I}	68%	74%	89%	46%
C + C	66%	73%	84%	32%
L + {U, I}	75%	75%	100%	30%
OVERALL	76%	80%	90%	

Table 4: Breakdown of performances on selected ACTUAL + ALTERNATIVE question pair types. For instance, the pair type C + E corresponds to instances where the ACTUAL question asked by the narrator is compatible and the ALTERNATIVE question is explicitly incompatible.

---

[14] See the Appendix for detailed annotation instructions.

[15] We obtained a second round of annotations on a subset of 75 task instances to compute agreement statistics.

[16] By construction, random accuracy is 50%.

[17] The example is adapted from our instructions to annotators, which includes further explanations for these categories. See the Appendix for details.

[18] See the Appendix for some representative examples for selected question pair types in our data.

[19] We also concede that some of the instances in this category may be unsolvable, e.g., when the wrong question fits the narrative just as well.

Model	Accuracy (held-out)
NARRATIVE-QN-SIM	73.4%
FINETUNED LM	**78.7%**

Table 5: Performance of different baselines.

5 Baseline systems performance

We divide our data into a 8,865-2,500 train-test split and have reserved 10,000 instances as a held-out set.[20] In Table 5 we report accuracy for the best-performing model on the (never-before-seen) held-out for a simple similarity-based method and for a deep learning method.

Narrative-question similarity. We expect that questions would show greater similarity to narratives they are removed from. We thus establish a narrative-question similarity baseline by considering features based on cosine similarities between narrative and questions, with text represented as tf-idf vectors, tf-idf weighted GloVe embeddings, averaged GloVe embeddings, as well as word overlap between content words, all combined in a logistic regression model.

Finetuned transformer LM. We also use a Finetuned Transformer LM model (Radford et al., 2018), which was shown to perform competitively on a diverse set of NLP tasks, achieving state-of-the-art results on the story cloze test.[21]

5.1 Error analysis

Required skills. As shown in Table 4, systems perform worst on items that do not exhibit any (implicit or explicit) mismatches (C + C), and thus might require some social intuition. Importantly, the largest gap between baseline and human performance (25%) is on the subset of items that can not be solved based solely on a semantic understanding (L + {U, I}). These results underline the need for models that can combine common sense reasoning about the events within the story with an intuition about the intention of the narrator.

Question concreteness. Questions may also differ in how concrete they are. In a preliminary analysis aimed at understanding how this property affects performance, we compare words used in ground-truth questions that the best-performing model predicts correctly with those used in questions that are classified incorrectly. We observe that questions that are predicted correctly have significantly higher average inverse document frequencies (t-test $p < 0.01$). Intuitively, these more specific questions may be more concrete in nature, making them easier to connect to the narratives to which they belong. We also find that some common interrogatives have skewed distributions. For instance, questions starting with *Is* are less likely to be classified correctly than those starting with *How*. A cursory manual investigation suggests that this can also be tied by concreteness, with the latter type of questions appearing to be more concrete than the former.

6 Further related work

One broad motivation behind our work is to eventually help better support personalized informational needs (Teevan et al., 2007). This connects to several related lines of work that were not previously discussed.

Query/question intents. Datasets and models are proposed for understanding user intents behind search queries (Radlinski et al., 2010; Fariha et al., 2018), or even more generally, user questions (Haponchyk et al., 2018). To complement this line of work that looks at user intents behind the explicit request, our task aims to uncover user intents when they are implied in personal narratives (without access to the explicit question).

Conversational search/QA. One way to better satisfy user intents is by making such processes collaborative (Morris and Horvitz, 2007; Morris, 2013), or conversational (Radlinski and Craswell, 2017). Conversational QA datasets (Choi et al., 2018; Reddy et al., 2019) have been introduced to help develop systems with such capability.

Social QA. Some questions posed by users are inherently more social in nature, and require more nuanced contextual understanding (Harabagiu, 2008). The social nature may affect how people ask questions (Dahiya and Talukdar, 2016; Rao and Daumé III, 2018), and pose challenges for identifying appropriate answers (Shtok et al., 2012; Zhang et al., 2017).

---

[20] The set annotated by humans is disjoint.

[21] We fine-tune with our training set on top of the pre-trained transformer language model, using the implementation from `https://github.com/huggingface/pytorch-openai-transformer-lm`.

7 Discussion

In this work, we introduce the new task of inferring advice-seeking intentions from personal narratives, a methodology for creating appropriate test instances for this task and the `ASQ` dataset. This task complements existing (intradiegetic) narrative understanding tasks by focusing on extradiegetic aspects of the narrative: in order to understand "Why is the narrator sharing this?", we often need to apply a certain degree of common sense and social intuition.

From a practical perspective, this extradiegetic capability is a prerequisite to properly address personalized information needs that are constrained by personal circumstances described as free-form personal stories. Currently, to address these types of information needs, people seek (or even hire) other individuals with relevant experience or expertise. As with conversational search (Radlinski and Craswell, 2017), we can envision systems that can more directly address complex information needs by better understanding the circumstances and intentions of the user.

Our analysis of the human and baseline performance on different types of test instances points to interesting avenues for future work, both in terms of designing better-performing systems and in terms of constructing better test data. We envision that (intradiegetic) narrative understanding could help identify the components of the narrative that are most relevant to the advice-seeking goal. For example, identifying the narrator's intentions and desires within the story (Rashkin et al., 2018b), and whether these desires are fulfilled (Rahimtoroghi et al., 2017) could help focus the attention of the model, especially when dealing with less concrete questions. Furthermore, a better representation of the structure of the narrative (Ouyang and McKeown, 2014), in terms of discourse acts (Elson, 2012) and sentiment flow (Ouyang and McKeown, 2015), could also help distinguish between spurious and essential circumstances of the narratives.

In terms of improving the task itself and the methodology for creating testing instances that better approximate the inferential task, we note a few possible directions. Firstly, better narrative modeling could lead to higher quality matching. Similarly, better representation of the questions can help select more appropriate candidate options (e.g., currently 6% of the questions are deemed by the annotators to be too general). In addition, the generative version of the task, when appropriately evaluated, could be a closer operationalization for intention inference, and also offer more potential for practical uses.

Finally, future work could expand on our methodology to formulate other more general tasks aiming to understand the reasons why a person is sharing a personal story. While we have focused on narratives shared with the intention of seeking advice, people may also share stories to express emotions, to entertain or educate others. A better understanding of these different (explicit or implicit) intentions could lead to more personalized and empathetic human-computer interaction.

Acknowledgments. The authors thank Tom Davidson, Tsung-Yu Hou, Qian Huang, Hajin Lim, Laure Thompson, Andrew Wang, Xiaozhi Wang and Justine Zhang for helping with the annotations. We are grateful to Thorsten Joachims, Avery Quinn Smith and Todd Cullen for helping us when our server crashed on the day of the deadline while testing the model on the held-out set, to Lillian Lee, Andrew Wang, Justine Zhang and the anonymous reviewers for their helpful comments, and to Fernando Pereira for the early discussions that inspired this research direction. This work is supported in part by NSF CAREER award IIS-1750615 and NSF Grant SES-1741441.

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. *arXiv:1707.02268v3*.

Melanie Baumgarten, Heinz-Martin Süß, and Susanne Weis. 2015. The Cue Is the Key: The Relevance of Cues and Contextual Information in the Social Understanding Tasks of the Magdeburg Test of Social Intelligence. *European Journal of Psychological Assessment*, 31(1).

Michael Bratman. 1987. *Intention, Plans, and Practical Reason*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL*.

Snigdha Chaturvedi, Dan Goldwasser, and Hal Daume III. 2016. Ask, and Shall You Receive? Understanding Desire Fulfillment in Natural Language Text. In *Proceedings of AAAI*.

Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis, Stanford University.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question Answering in Context. In *Proceedings of EMNLP*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of EMNLP*.

Kristin Conzelmann. 2012. *Social Intelligence and Auditory Intelligence – Useful Constructs?* Ph.D. thesis, The Otto von Guericke University Magdeburg.

Kristin Conzelmann, Susanne Weis, and Heinz-Martin Süß. 2013. New Findings About Social Intelligence: Development and Application of the Magdeburg Test of Social Intelligence (MTSI). *Journal of Individual Differences*, 34(3).

Yogesh Dahiya and Partha Talukdar. 2016. Discovering Response-Eliciting Factors in Social Question Answering: A Reddit Inspired Study. *Proceedings of ICWSM*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. In *Proceedings of ACL*.

David K. Elson. 2012. *Modeling Narrative Discourse*. Ph.D. thesis, Columbia University.

Anna Fariha, Sheikh M. Sarwar, and Alexandra Meliou. 2018. SQuID: Semantic Similarity-Aware Query Intent Discovery. In *Proceedings of SIGMOD*.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story Ending Generation with Incremental Encoding and Commonsense Knowledge. *Proceedings of AAAI*.

Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina, and Alessandro Moschitti. 2018. Supervised Clustering of Questions into Intents for Dialog System Applications. In *Proceedings of EMNLP*.

Sanda M. Harabagiu. 2008. Questions and Intentions. In *Advances in Open Domain Question Answering*.

Andrew Kehler and Hannah Rohde. 2017. Evaluating an Expectation-Driven Question-Under-Discussion Model of Discourse Interpretation. *Discourse Processes*, 54(3).

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl M. Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA Reading Comprehension Challenge. *TACL*.

William Labov. 1972. The Transformation of Experience in Narrative Syntax. *Language in the Inner City*.

Meredith R. Morris. 2013. Collaborative Search Revisited. In *Proceedings of CSCW*.

Meredith R. Morris and Eric Horvitz. 2007. SearchTogether: An Interface for Collaborative Web Search. In *Proceedings of UIST*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of NAACL*.

Ani Nenkova. 2011. Automatic Summarization. *Foundations and Trends in Information Retrieval*, 5(2).

Jessica Ouyang and Kathleen McKeown. 2014. Towards Automatic Detection of Narrative Structure. In *Proceedings the LREC*.

Jessica Ouyang and Kathleen McKeown. 2015. Modeling Reportable Events as Turning Points in Narrative. In *Proceedings of EMNLP*.

James W. Pennebaker. 1997. Writing About Emotional Experiences as a Therapeutic Process. *Psychological Science*, 8(3).

James W. Pennebaker and Janel D. Seagal. 1999. Forming a Story: The Health Benefits of Narrative. *Journal of Clinical Psychology*, 55(10).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-training. *Preprint*.

Filip Radlinski and Nick Craswell. 2017. A Theoretical Framework for Conversational Search. In *Proceedings of CHIIR*.

Filip Radlinski, Martin Szummer, and Nick Craswell. 2010. Inferring Query Intent from Reformulations and Clicks. In *Proceedings of WWW*.

Elahe Rahimtoroghi, Jiaqi Wu, Ruimin Wang, Pranav Anand, and Marilyn Walker. 2017. Modelling Protagonist Goals and Desires in First-Person Narrative. In *Proceedings of SIGDIAL*.

Sudha Rao and Hal Daumé III. 2018. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. In *Proceedings of ACL*.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018a. Modeling Naive Psychology of Characters in Simple Commonsense Stories. *Proceedings of ACL*.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018b. Event2Mind: Commonsense Inference on Events, Intents, and Reactions. In *Proceedings of ACL*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *TACL*.

Matthew Richardson. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of EMNLP*.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the Past: Answering New Questions with Past Answers. In *Proceedings of WWW*.

Reid W. Swanson, Andrew S. Gordon, Peter Khooshabeh, Kenji Sagae, Richard Huskey, Michael Mangus, Ori Amir, and Rene Weber. 2017. An Empirical Analysis of Subjectivity and Narrative Levels in Weblog Storytelling Across Cultures. *Dialogue & Discourse*, 8(2).

Chenhao Tan and Lillian Lee. 2015. All Who Wander: On the Prevalence and Characteristics of Multi-community Engagement. In *Proceedings of WWW*.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2007. Characterizing the Value of Personalizing Search. In *Proceedings of SIGIR*.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2018. Large-Scale Cloze Test Dataset Created by Teachers. In *Proceedings of EMNLP*.

Wei E. Zhang, Quan Z. Sheng, Jey Han Lau, and Ermyas Abebe. 2017. Detecting Duplicate Posts in Programming QA Communities via Latent Semantics and Association Rules. In *Proceedings of WWW*.

Appendix

A1 Instructions to human annotators

As described in Section 4 of the main paper, we obtained human annotations on a small subset of our data for validation purposes. Annotators were shown instructions which included the definitions for different question-narrative plausibility types, together with two examples to help further clarify the task and the definitions. The exact instructions are reproduced in Table A8, while the examples provided are shown in Table A9.

A2 Distribution of plausibility categories

Table A6 shows the distribution of plausibility categories given out by our human annotators, for both the actual questions which belong to the given narrative (Column 2) and the paired alternative questions (Column 3).

Question type	**% in actual**	**% in altern.**
Compatible and **L**ikely	81%	15%
Compatible but **U**nlikely	10%	21%
Incompatible (**I**mplicit)	5%	16%
Incompatible (**E**xplicit)	2%	41%
Very **G**eneral	3%	8%

Table A6: Data distribution estimated from the human annotated subset. For more than half of the cases, the wrong answer (i.e., the alternative question) could not be simply discarded based on factual mismatches, and the task instance would require additional common sense or social intuition to solve.

# of unique post ids	415,693
# of posts with narrative bodies	339,815
# of narratives with questions	262,721
# of narratives after filtering for length	151,418
# of narratives with specific questions	89,527
# of narratives paired	43,730

Table A7: Counts for narrative instances at different stages of dataset construction.

A3 Further processing details

Table A7 provides the number of instances remaining at each of the processing steps. After masking one narrative from each narrative pair, we have a total of 21,865 narratives, each successfully paired with two plausible candidate questions.

A4 Examples of pair types

In our error analysis, we find that the performance of both our baselines, as well as that of our human annotators, vary depending on question pair type, where pair type is defined as the human-judged plausibility of the ground-truth question and that of the alternative question. To give a better sense of what these pair types look like in practice, Table A10 shows example instances for a few selected pair types.

You will be presented with one narrative and two advice-seeking questions (qn1 and qn2).

Firstly, you will need to indicate which of these questions is **more likely** to be asked by the narrator in the context of the narrative (Column D, in yellow). Use the dropdown menu to select the more likely question among the two. (You must pick one.)

In addition, for **each** question separately, you will need to provide a rating on how plausible the question is in the context of the narrative by choosing one of the following options (dropdown menu in Columns E and F, in green):

1. **Very general**: i.e., this question could follow most narratives, and it's not in any way specific to this narrative.
2. **Compatible and likely**: i.e., the question follows naturally from this narrative.
3. **Compatible but unlikely**: i.e., while there is no direct contradiction (either explicit or implicit) with the narrative, it seems unlikely that the narrator's intention was to ask this question.
4. **Incompatible (explicit)**: i.e., the question is incompatible due to clear factual mismatches with the information explicitly contained in the narrative, or it is completely irrelevant.
5. **Incompatible (implicit)**: i.e., the question is incompatible due to mismatches with something that you can indirectly infer from the narrative.

You should judge each question **separately** when selecting a category. It is possible for both questions to fall into the same category.

Now you need to read the example narratives, questions and explanations in the adjacent cells (B2 and C2) to get a feel of each of the categories, after which you could proceed to the sub-sheet **Items to annotate (see tab at the bottom of this page)** to complete the annotation task.

Optionally, you can also provide comments for each item (scroll to the right to see the comment column): Did you find an item particularly challenging or interesting? Was one of the questions not really asking for an advice? Do share your thoughts with us.

Table A8: Instruction text shown to the annotators.

Example 1	**Example 2**
Narrative: “I am a freshman in college and I have a group of friends that I have been hanging out with for the past couple months. I feel like we have a good time when we hang out, but a lot of the time, the rest of the group will go out and do stuff together, but I won’t be included.” Example questions in each category [and explanations where appropriate]: a) **Very general**: Any advice? b) **Compatible and likely**: Can I ask to be included? c) **Compatible but unlikely**: How to make new friends in college? [explanation: it is more likely that the narrator is trying to be more included in the current group of friends, rather than giving up entirely on them and look for replacement.] d) **Incompatible (explicit)**: Any advice for finding new friends for a senior in college? [Explanation: The narrator is a freshman in college, not a senior. This constitutes a clear factual mismatch between the question and the narrative.] e) **Incompatible (implicit)**: What are some excuses to not hang out with them? [Explanation: We can imply from the narrative that the narrator wants to hang out with the group. This is incompatible with a question asking how NOT to do that.]	Narrative: “I asked a girl that I really like if she would like to get coffee sometime. She said she’s really busy but that we’ll see. I can’t get her off my mind and I spend all day waiting for her to tell me she’s free.” Example questions in each category [and explanations where appropriate]: a) **Very general**: What should I do? b) **Compatible and likely**: Would it seem desperate if I asked her again in a more direct way a week later ? c) **Compatible but unlikely**: Which coffee place would you recommend? [Explanation: While the narrator is trying to invite the girl for coffee, the main concern seems to be whether the attempt would be successful rather the choices between coffee places.] d) **Incompatible (explicit)**: How to deal with my roommate? [Explanation: This question is completely irrelevant to the narrative (no roommate is mentioned).] e) **Incompatible (implicit)**: What to do if I asked a girl out and now regret it? [Explanation: We can infer that the narrator is looking forward to the potential date, which contradicts with the feeling of regret in the question.]

Table A9: Example narratives and questions shown to the annotators.

Pair type	Narrative	Actual question	Alternative question
L + L	Hey everyone I have a bit of a dilemma. It's the first week of school and I am talking three advanced classes, AP world history II, English honors II and Chemistry honors. I am pretty sure that I can handle it but; I am falling behind in chemistry honors and it is the first week. I don't have the mathematical background as the other students. They have taken physics and geometry. I am in a special Algebra class which means I am a year behind in math and science.	Should I drop chemistry honors?	How much do honors courses matter?
L + U	My college roommate/one of my best friends is getting married Saturday. I'm a groomsman, as is our third roommate. Our third roommate gave he and his betrothed their wedding gifts early today: an Xbox One and a crystal decorative bowl from Tiffany. I'm an assistant manager at a sporting goods store making \$8.50 an hour, and between rent, utilities, groceries, gas, and my student loan payments, I usually either barely break even every month or have to borrow money from my parents until my next paycheck. I've checked their registry, and even the less expensive gifts are outside what I can afford (\$30 can make or break me right now).	What to do about a wedding gift if I'm broke?	Where can I buy food that's cheap, and it'll last me until then?
C + I	I just recently switched schools this school year. I'm pretty okay with how it's going so far academics wise but I have no idea how to put myself out there. Everyone has seemed to have made friend groups already or they already know everyone from previous years. I used to be in a private school so no one really knows me from this school except for my close friends that I've known for a long time.	Is there any way that I could gain any popularity before it's too late?	Is it too early to tell if I want to drop out?
L + E	So there is a dream job which is PERFECT for me and of course I really want it. I called the employer last week and she said she was going to call candidates for interviews that week. Then I called this week and she said she was going to call for interviews this week. And please, no advice telling me 'don't call'. I have nothing to lose, so I'm going to call, I would just really appreciate some advice as to how to ask for an interview appropriately - Thank you all!	How do I call an employer asking for an interview?	When I went for the interview she did seem busy so maybe she was too busy to call?
L + G	I'm in a relationship with an amazing girl and feel very happy with her. Recently though I've been having intrusive thoughts about her ex-boyfriends (her having sex with them, etc) which are leading to feelings of jealousy and it's really disrupting my ability to enjoy my time with her.	How to deal with feelings of jealousy?	Is this "normal" – in the sense of, do other people experience this?

Table A10: Example task instances for different ACTUAL + ALTERNATIVE question pair types.

Seeing Things from a Different Angle:
Discovering Diverse Perspectives about Claims

Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch, Dan Roth
Department of Computer and Information Science, University of Pennsylvania
`{sihaoc,danielkh,wenpeng,ccb,danroth}@cis.upenn.edu`

Abstract

One key consequence of the information revolution is a significant increase and a contamination of our information supply. The practice of fact-checking won't suffice to eliminate the biases in text data we observe, as the degree of factuality alone does not determine whether biases exist in the spectrum of opinions visible to us. To better understand controversial issues, one needs to view them from a diverse yet comprehensive set of *perspectives*.

For example, there are many ways to respond to a *claim* such as *"animals should have lawful rights"*, and these responses form a spectrum of perspectives, each with a *stance* relative to this claim and, ideally, with evidence supporting it. Inherently, this is a natural language understanding task, and we propose to address it as such. Specifically, we propose the task of *substantiated perspective discovery* where, given a *claim*, a system is expected to discover a *diverse* set of *well-corroborated perspectives* that take a *stance* with respect to the claim. Each perspective should be substantiated by *evidence* paragraphs which summarize pertinent results and facts.

We construct PERSPECTRUM, a dataset of claims, perspectives and evidence, making use of online debate websites to create the initial data collection, and augmenting it using search engines in order to expand and diversify our dataset. We use crowdsourcing to filter out noise and ensure high-quality data. Our dataset contains $1k$ claims, accompanied by pools of $10k$ and $8k$ perspective sentences and evidence paragraphs, respectively. We provide a thorough analysis of the dataset to highlight key underlying language understanding challenges, and show that human baselines across multiple subtasks far outperform machine baselines built upon state-of-the-art NLP techniques. This poses a challenge and an opportunity for the NLP community to address.

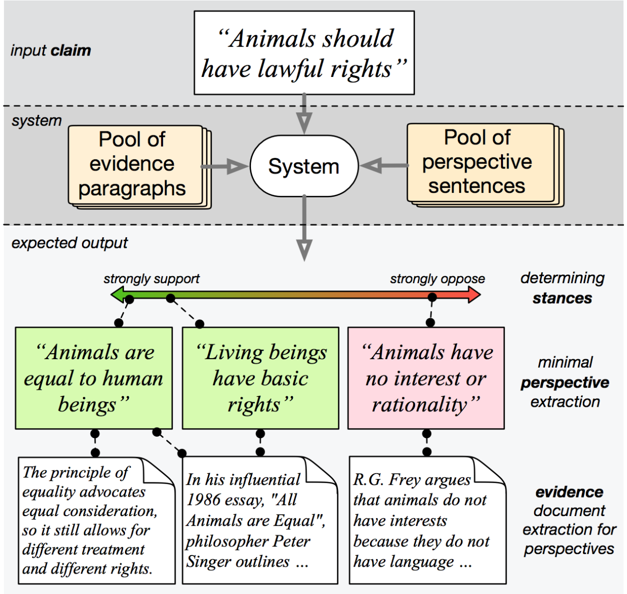


Figure 1: Given a *claim*, a hypothetical system is expected to discover various *perspectives* that are substantiated with *evidence* and their *stance* with respect to the claim.

1 Introduction

Understanding most nontrivial *claims* requires insights from various *perspectives*. Today, we make use of search engines or recommendation systems to retrieve information relevant to a claim, but this process carries multiple forms of *bias*. In particular, they are optimized relative to the claim (query) presented, and the popularity of the relevant documents returned, rather than with respect to the diversity of the *perspectives* presented in them or whether they are supported by evidence.

In this paper, we explore an approach to mitigating this *selection bias* (Heckman, 1979) when studying (disputed) claims. Consider the *claim* shown in Figure 1: *"animals should have lawful rights."* One might compare the biological similarities/differences between humans and other an-

Proceedings of NAACL-HLT 2019, pages 542–557
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

imals to support/oppose the claim. Alternatively, one can base an argument on morality and rationality of animals, or lack thereof. Each of these arguments, which we refer to as *perspectives* throughout the paper, is an opinion, possibly conditional, in support of a given *claim* or against it. A *perspective* thus constitutes a particular attitude towards a given *claim*.

Natural language understanding is at the heart of developing an ability to identify diverse perspectives for claims. In this work, we propose and study a setting that would facilitate discovering *diverse perspectives* and their supporting evidence with respect to a given *claim*. Our goal is to identify and formulate the key NLP challenges underlying this task, and develop a dataset that would allow a systematic study of these challenges. For example, for the claim in Figure 1, multiple (non-redundant) perspectives should be retrieved from a pool of perspectives; one of them is *"animals have no interest or rationality"*, a *perspective* that should be identified as taking an *opposing* stance with respect to the *claim*. Each *perspective* should also be well-supported by *evidence* found in a pool of potential pieces of evidence. While it might be impractical to provide an exhaustive spectrum of ideas with respect to a *claim*, presenting a small but diverse set of *perspectives* could be an important step towards addressing the *selection bias* problem. Moreover, it would be impractical to develop an exhaustive pool of evidence for all perspectives, from a diverse set of credible sources. We are not attempting to do that. We aim at formulating the core NLP problems, and developing a dataset that will facilitate studying these problems from the NLP angle, realizing that using the outcomes of this research in practice requires addressing issues such as trustworthiness (Pasternack and Roth, 2010, 2013) and possibly others. Inherently, our objective requires understanding the relations between *perspective*s and *claim*s, the nuances in the meaning of various *perspectives* in the context of *claim*s, and relations between perspectives and evidence. This, we argue, can be done with a diverse enough, but not exhaustive, dataset. And it can be done without attending to the legitimacy and credibility of sources contributing evidence, an important problem but orthogonal to the one studied here.

To facilitate the research towards developing solutions to such challenging issues, we propose PERSPECTRUM, a dataset of *claims*, *perspectives* and *evidence* paragraphs. For a given *claim* and pools of *perspectives* and *evidence paragraphs*, a hypothetical system is expected to select the relevant perspectives and their supporting paragraphs.

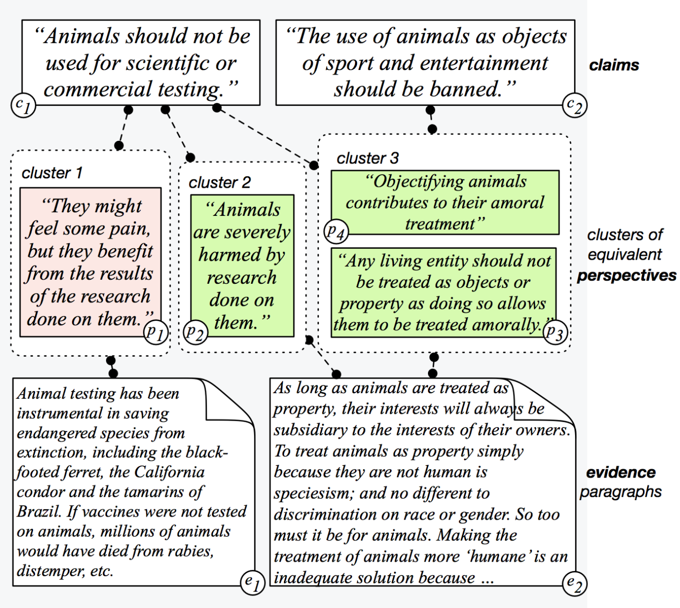


Figure 2: Depiction of a few claims, their *perspectives* and evidences from PERSPECTRUM. The *supporting* and *opposing* perspectives are indicated with green and red colors, respectively.

Our dataset contains 907 claims, 11,164 perspectives and 8,092 evidence paragraphs. In constructing it, we use online debate websites as our initial seed data, and augment it with search data and paraphrases to make it richer and more challenging. We make extensive use of crowdsourcing to increase the quality of the data and clean it from annotation noise.

The contributions of this paper are as follows:

- To facilitate making progress towards the problem of *substantiated perspective discovery*, we create a high-quality dataset for this task.[1]
- We identify and formulate multiple NLP tasks that are at the core of addressing the *substantiated perspective discovery* problem. We show that humans can achieve high scores on these tasks.
- We develop competitive baseline systems for each sub-task, using state-of-the-art techniques.

---

[1] https://github.com/CogComp/perspectrum

2 Design Principles and Challenges

In this section we provide a closer look into the challenge and propose a collection of tasks that move us closer to *substantiated perspective discovery*. To clarify our description we use to following notation. Let c indicate a target claim of interest (for example, the claims c_1 and c_2 in Figure 2). Each claim c is addressed by a collection of perspectives $\{p\}$ that are grouped into clusters of *equivalent* perspectives. Additionally, each perspective p is supported, relative to c, by at least one evidence paragraph e, denoted $e \vDash p|c$.

Creating systems that would address our challenge in its full glory requires solving the following interdependent tasks:

Determination of argue-worthy claims: not every claim requires an in-depth discussion of perspectives. For a system to be practical, it needs to be equipped with understanding argumentative structures (Palau and Moens, 2009) in order to discern disputed claims from those with straightforward responses. We set aside this problem in this work and assume that all the inputs to the systems are discussion-worthy claims.

Discovery of pertinent perspectives: a system is expected to recognize argumentative sentences (Cabrio and Villata, 2012) that directly address the points raised in the disputed claim. For example, while the perspectives in Figure 2 are topically related to the claims, p_1, p_2 do not directly address the focus of claim c_2 (i.e., *"use of animals" in "entertainment"*).

Perspective equivalence: a system is expected to extract a *minimal* and *diverse* set of perspectives. This requires the ability to discover equivalent perspectives p, p', with respect to a claim c: $p|c \approx p^{'}|c$. For instance, p_3 and p_4 are equivalent in the context of c_2; however, they might not be equivalent with respect to any other claim. The conditional nature of perspective equivalence differentiates it from the *paraphrasing* task (Bannard and Callison-Burch, 2005).

Stance classification of perspectives: a system is supposed to assess the stances of the perspectives with respect to the given claim (supporting, opposing, etc.) (Hasan and Ng, 2014).

Substantiating the perspectives: a system is expected to find valid evidence paragraph(s) in support of each perspective. Conceptually, this is similar to the well-studied problem of textual entailment (Dagan et al., 2013) except that here the entailment decisions depend on the choice of claims.

3 Related Work

Claim verification. The task of *fact verification* or *fact-checking* focuses on the assessment of the truthfulness of a claim, given evidence (Vlachos and Riedel, 2014; Mitra and Gilbert, 2015; Samadi et al., 2016; Wang, 2017; Nakov et al., 2018; Hanselowski et al., 2018; Karimi et al., 2018; Alhindi et al., 2018). These tasks are highly related to the task of textual-entailment that has been extensively studied in the field (Bentivogli et al., 2008; Dagan et al., 2013; Khot et al., 2018). Some recent work study jointly the problem of identifying evidence and verifying that it supports the claim (Yin and Roth, 2018).

Our problem structure encompasses the *fact verification* problem (as verification of *perspectives* from *evidence*; Figure 1).

Stance classification. Stance classification aims at detecting phrases that *support* or *oppose* a given claim. The problem has gained significant attention in the recent years; to note a few important ones, Hasan and Ng (2014) create a dataset of dataset text snippets, annotated with "reasons" (similar to *perspectives* in this work) and stances (whether they support or oppose the claim). Unlike this work, our pool of the relevant "reasons" is not restricted. Ferreira and Vlachos (2016) create a dataset of rumors (claims) coupled with news headlines and their stances. There are a few other works that fall in this category (Boltužić and Šnajder, 2014; Park and Cardie, 2014; Rinott et al., 2015; Swanson et al., 2015; Mohammad et al., 2016; Sobhani et al., 2017; Bar-Haim et al., 2017). Our approach here is closely related to existing work in this direction, as stance classification is part of the problem studied here.

Argumentation. There is a rich literature on *formalizing* argumentative structures from free text. There are a few theoretical works that lay the ground work to characterizing units of arguments and argument-inducing inference (Teufel et al., 1999; Toulmin, 2003; Freeman, 2011).

Others have studied the problem of extracting argumentative structures from free-form text; for example, Palau and Moens (2009); Khatib et al. (2016); Ajjour et al. (2017) studied elements of arguments and the internal relations between them.

Dataset	Stance Classification	Evidence Verification	Human Verified	Open Domain
PERSPECTRUM (this work)	✓	✓	✓	✓
FEVER (Thorne et al., 2018)	✗	✓	✓	✓
(Wachsmuth et al., 2017)	✓	✓	✗	✓
LIAR (Wang, 2017)	✗	✓	✓	✓
(Vlachos and Riedel, 2014)	✗	✓	✓	✓
(Hasan and Ng, 2014)	✓	✗	✓	✗

Table 1: Comparison of PERSPECTRUM to a few notable datasets in the field.

Feng and Hirst (2011) classified an input into one of the argument schemes. Habernal and Gurevych (2017) provided a large corpus annotated with argument units. Cabrio and Villata (2018) provide a thorough survey the recent work in this direction. A few other works studied other aspects of argumentative structures (Cabrio and Villata, 2012; Khatib et al., 2016; Lippi and Torroni, 2016; Zhang et al., 2017; Stab and Gurevych, 2017).

A few recent works use a similar conceptual design that involves a *claim*, *perspectives* and *evidence*.These works are either too small due to the high cost of construction (Aharoni et al., 2014) or too noisy because of the way they are crawled from online resources (Wachsmuth et al., 2017; Hua and Wang, 2017). Our work makes use of both online content and of crowdsourcing, in order to construct a sizable and high-quality dataset.

4 The PERSPECTRUM Dataset

4.1 Dataset construction

In this section we describe a multi-step process, constructed with detailed analysis, substantial refinements and multiple pilots studies.

We use crowdsourcing to annotate different aspects of the dataset. We used Amazon Mechanical Turk (AMT) for our annotations, restricting the task to workers in five English-speaking countries (USA, UK, Canada, New Zealand, and Australia), more than 1000 finished HITs and at least a 95% acceptance rate. To ensure the diversity of responses, we do not require additional qualifications or demographic information from our annotators.

For any of the annotations steps described below, the users are guided to an external platform where they first read the instructions and try a verification step to make sure they have understood the instructions. Only after successful completion are they allowed to start the annotation tasks.

Throughout our annotations, it is our aim to make sure that the workers are responding objectively to the tasks (as opposed to using their personal opinions or preferences). The screen-shots of the annotation interfaces for each step are included in the Appendix (Section A.3).

In the steps outlined below, we filter out a subset of the data with low rater–rater agreement ρ (see Appendix A.2). In certain steps, we use an information retrieval (IR) system[2] to generate the best candidates for the task at hand.

Step 1: The initial data collection. We start by crawling the content of a few notable debating websites: `idebate.com`, `debatewise.org`, `procon.org`. This yields $\sim 1k$ claims, $\sim 8k$ perspectives and $\sim 8k$ evidence paragraphs (for complete statistics, see Table 4 in the Appendix). This data is significantly noisy and lacks the structure we would like. In the following steps we explain how we denoise it and augment it with additional data.

Step 2a: Perspective verification. For each perspective we verify that it is a complete English sentence, with a clear stance with respect to the given claim. For a fixed pair of *claim* and *perspective*, we ask the crowd-workers to label the perspective with one of the five categories of *support*, *oppose*, *mildly-support*, *mildly-oppose*, or *not a valid perspective*. The reason that we ask for two levels of intensity is to distinguish *mild* or *conditional* arguments from those that express *stronger* positions.

Every 10 claims (and their relevant perspectives) are bundled to form a HIT. Three independent annotators solve a HIT, and each gets paid \$1.5-2 per HIT. To get rid of the ambiguous/noisy perspectives we measure rater-rater agreement on the resulting data and retain only the subset which has a significant agreement of $\rho \geq 0.5$. To account for minor disagreements in the intensity of

---

[2] `www.elastic.co`

perspective stances, before measuring any notion of agreement, we collapse the five labels into three labels, by collapsing *mildly-support* and *mildly-oppose* into *support* and *oppose*, respectively.

To assess the quality of these annotations, two of the authors independently annotate a random subset of instances in the previous step (328 perspectives for 10 claims). Afterwards, the differences were adjudicated. We measure the accuracy adjudicated results with AMT annotations to estimate the quality of our annotation. This results in an accuracy of 94%, which shows high-agreement with the crowdsourced annotations.

Step 2b: Perspective paraphrases. To enrich the ways the perspectives are phrased, we crowdsource paraphrases of our perspectives. We ask annotators to generate two paraphrases for each of the 15 perspectives in each HIT, for a reward of $1.50.

Subsequently, we perform another round of crowdsourcing to verify the generated paraphrases. We create HITs of 24 candidate paraphrases to be verified, with a reward of $1. Overall, this process gives us $\sim$ 4.5 paraphrased perspectives. The collected paraphrases form clusters of equivalent perspectives, which we refine further in the later steps.

Step 2c: Web perspectives. In order to ensure that our dataset contains more realistic sentences, we use web search to augment our pool of perspectives with additional sentences that are topically related to what we already have. Specifically, we use Bing search to extract sentences that are similar to our current pool of perspectives, by querying "claim+perspective". We create a pool of relevant web sentences and use an IR system (introduced earlier) to retrieve the 10 most similar sentences. These candidate perspectives are annotated using (similar to step 2a) and only those that were agreed upon are retained.

Step 2d: Final perspective trimming. In a final round of annotation for perspectives, an expert annotator went over all the claims in order to verify that all the equivalent perspectives are clustered together. Subsequently, the expert annotator went over the most similar claim-pairs (and their perspectives), in order to annotate the missing perspectives shared between the two claims. To cut the space of claim pairs, the annotation was done on the top 350 most similar claim pairs retrieved by the IR system.

Category	Statistic	Value
Claims	# of claims (step 1)	907
	avg. claim length (tokens)	8.9
	median claims length (tokens)	8
	max claim length (tokens)	30
	min claim length (tokens)	3
Perspectives	# of perspectives	11,164
	Debate websites (step 1)	4,230
	Perspective paraphrase (step 2b)	4,507
	Web (step 2c)	2,427
	# of perspectives with stances	5,095
	# of "support" perspectives	2,627
	# of "opposing" perspectives	2,468
	avg size of perspective clusters	2.3
	avg length of perspectives (tokens)	11.9
Evidences	# of total evidences (step 1)	8,092
	avg length of evidences (tokens)	168

Table 2: A summary of PERSPECTRUM statistics

Step 3: Evidence verification. The goal of this step is to decide whether a given evidence paragraph provides enough substantiations for a perspective or not. Performing these annotations exhaustively for any perspective-evidence pair is not possible. Instead, we make use of a retrieval system to annotate only the relevant pairs. In particular, we create an index of all the perspectives retained from *step 2a*. For a given evidence paragraph, we retrieve the top relevant perspectives. We ask the annotators to note whether a given evidence paragraph *supports* a given perspective or not. Each HIT contains a 20 evidence paragraphs and their top 8 relevant candidate perspectives. Each HIT is paid $1 and annotated by at least 4 independent annotators.

In order to assess the quality of our annotations, a random subset of instances (4 evidence-perspective pairs) are annotated by two independent authors and the differences are adjudicated. We measure the accuracy of our adjudicated labels versus AMT labels, resulting in 87.7%. This indicates the high quality of the crowdsourced data.

4.2 Statistics on the dataset

We now provide a brief summary of PERSPECTRUM. The dataset contains about $1k$ claims with a significant length diversity (Table 2). Additionally, the dataset comes with $\sim$ $12k$ perspectives, most of which were generated through paraphrasing (step 2b). The perspectives which convey the same point with respect to a claim are grouped into clusters. On average, each cluster has a size of 2.3 which shows that, on average, many perspectives have

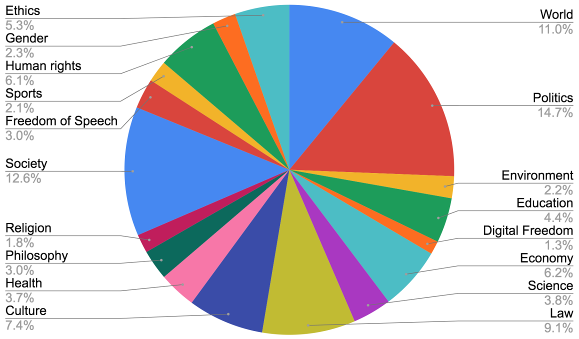


Figure 3: Distribution of claim topics.

equivalents. More granular details are available in Table 2.

To better understand the topical breakdown of claims in the dataset, we crowdsource the set of "topics" associated with each *claim* (e.g., *Law, Ethics, etc.*) We observe that, as expected, the three topics of *Politics, World, and Society* have the biggest portions (Figure 3). Additionally, the included claims touch upon 10+ different topics. Figure 4 depicts a few popular categories and sampled questions from each.

4.3 Required skills

We perform a closer investigation of the abilities required to solve the stance classification task. One of the authors went through a random subset of claim-perspectives pairs and annotated each with the abilities required in determining their stances labels. We follow the common definitions used in prior work (Sugawara et al., 2017; Khashabi et al., 2018). The result of this annotation is depicted in Figure 5. As can be seen, the problem requires understanding of *common-sense*, i.e., an understanding that is commonly shared among humans and rarely gets explicitly mentioned in the text. Additionally, the task requires various types of *coreference* understanding, such as *event coreference* and *entity coreference*.

5 Empirical Analysis

In this section we provide empirical analysis to address the tasks. We create a split of 60%/15%/25% of the data train/dev/test. In order to make sure our baselines are not overfitting to the keywords of each topic (the "topic" annotation from Section 4.2), we make sure to have claims with the same topic fall into the same split.

For simplicity, we define a notation which we will extensively use for the rest of this paper. The clusters of equivalent perspectives are denoted as $[\![p]\!]$, given a representative member p. Let $P(c)$ denote the collection of relevant perspectives to a claim c, which is the union of all the equivalent perspectives participating in the claim: $\{[\![p_i]\!]\}_i$. Let $E([\![p]\!]) = E(p) = \bigcup_i e_i$ denote the set of evidence documents lending support to a perspective p. Additionally, denote the two pools of perspectives and evidence with $\mathcal{U}^p$ and $\mathcal{U}^e$, respectively.

5.1 Systems

We make use of the following systems in our evaluation:

IR (Information Retrieval). This baseline has been successfully used for related tasks like Question Answering (Clark et al., 2016). We create two versions of this baseline: one with the pool of perspectives $\mathcal{U}^p$ and one with the pool of evidences $\mathcal{U}^e$. We use this system to retrieve a ranked list of best matching perspective/evidence from the corresponding index.

BERT (Contextual representations). A recent state-of-the-art contextualized representation (Devlin et al., 2018). This system has been shown to be effective on a broad range of natural language understanding tasks.

Human Performance. Human performance provides us with an estimate of the best achievable results on datasets. We use human annotators to measure human performance for each task. We randomly sample 10 claims from the test set, and instruct two expert annotators to solve each of T1 to T4.

5.2 Evaluation metrics.

We perform evaluations on four different subtasks in our dataset. In all of the following evaluations, the systems are given the two pools of perspectives $\mathcal{U}^p$ and evidences $\mathcal{U}^e$.

T1: Perspective extraction. A system is expected to return the collection of mutually disjoint perspectives with respect to a given claim. Let $\hat{P}(c)$ be the set of output perspectives. Define the precision and recall as $\text{Pre}(c) = \frac{\sum_{\hat{p} \in \hat{P}(c)} \mathbf{1}\{\exists p, s.t. \hat{p} \in [\![p]\!]\}}{|\hat{P}(c)|}$ and $\text{Rec}(c) = \frac{\sum_{\hat{p} \in \hat{P}(c)} \mathbf{1}\{\exists p, s.t. \hat{p} \in [\![p]\!]\}}{|P(c)|}$ respectively. To calculate dataset metrics, the aforementioned per-claim metrics are averaged across all the claims in the test set.

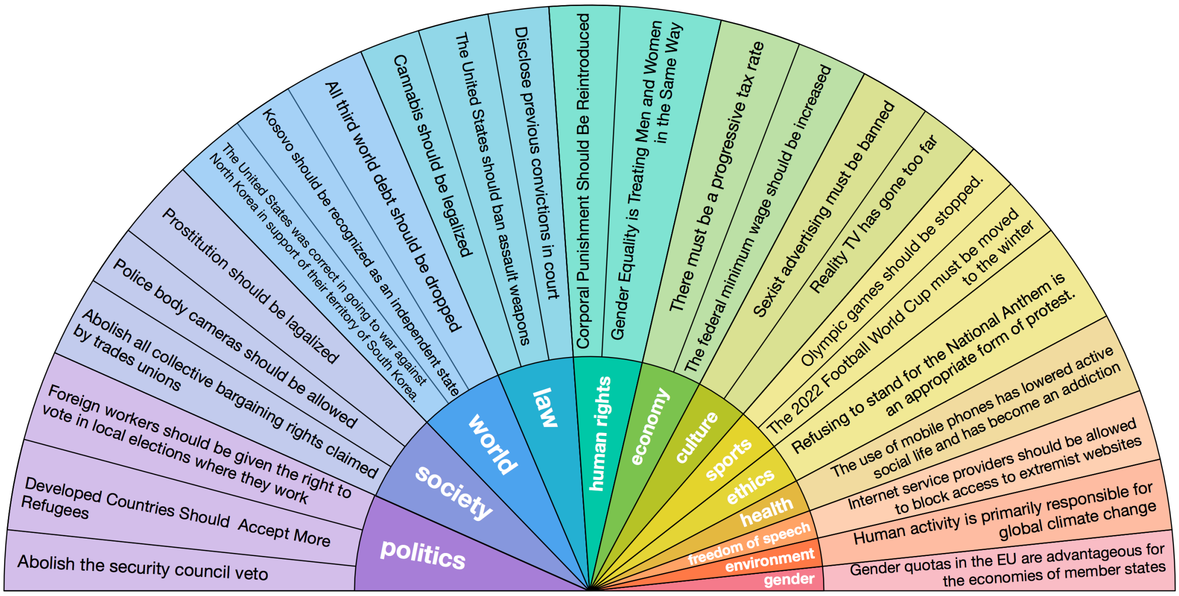


Figure 4: Visualization of the major topics and sample claims in each category.

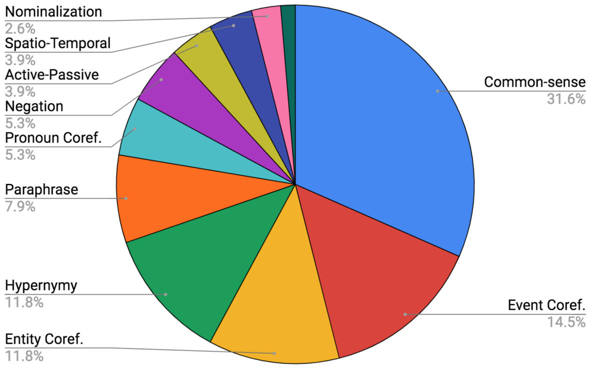


Figure 5: The set of reasoning abilities required to solve the stance classification task.

T2: Perspective stance classification. Given a claim, a system is expected to label every perspective in $P(c)$ with one of two labels *support* or *oppose*. We use the well-established definitions of precision-recall for this binary classification task.

T3: Perspective equivalence. A system is expected to decide whether two given perspectives are equivalent or not, with respect to a given claim. We evaluate this task in a way similar to a clustering problem. For a pair of perspectives $p_1, p_2 \in P(c)$, a system predicts whether the two are in the same cluster or not. The ground-truth is whether there is a cluster which contains both of the perspectives or not: $\exists \tilde{p} \ s.t. \ \tilde{p} \in P(c) \wedge p_1, p_2 \in \llbracket \tilde{p} \rrbracket$. We use this pairwise definition for all the pairs in $P(c) \times P(c)$, for any claim c in the test set.

T4: Extraction of supporting evidences. Given a perspective p, we expect a system to return all the evidence $\{e_i\}$ from the pool of evidence $\mathcal{U}^e$. Let $\hat{E}(p)$ and $E(p)$ be the predicted and gold evidence for a perspective p. Define macro-precision and macro-recall as $\text{Pre}(p) = \frac{|\hat{E}(p) \cap E(p)|}{|\hat{E}(p)|}$ and $\text{Rec}(p) = \frac{|\hat{E}(p) \cap E(p)|}{|E(p)|}$, respectively. The metrics are averaged across all the perspectives p participating in the test set.

T5: Overall performance. The goal is to get estimates of the overall performance of the systems. Instead of creating a complex measure that would take all the aspects into account, we approximate the overall performance by multiplying the disjoint measures in $T1$, $T2$ and $T4$. While this gives an estimate on the overall quality, it ignores the pipeline structure of the task (e.g., the propagation of the errors throughout the pipeline). We note that the task of $T3$ (perspective equivalence) is indirectly being measured within $T1$. Furthermore, since we do not report an IR performance on $T2$, we use the "always supp" baseline instead to estimate an overall performance for IR.

5.3 Results

5.3.1 Minimal perspective extraction (T1)

Table 3 shows a summary of the experimental results. To measure the performance of the IR system, we use the index containing $\mathcal{U}^p$. Given each claim, we query the top k perspectives, ranked according to their retrieval scores. We tune k on our development set and report the results on the test section according to the tuned parameter. We use IR results as candidates for other solvers (including humans). For this task, IR with top-15 candidates yields $>90\%$ recall (for the PR-curve, see Figure 6 in the Appendix). In order to train BERT on this task, we use the IR candidates as the training instances. We then tune a threshold on the dev data to select the top relevant perspectives. In order to measure human performance, we create an interface where two human annotators see IR top-k and select a *minimal* set of perspectives (i.e., no two equivalent perspectives).

5.3.2 Perspective stance classification (T2)

We measure the quality of perspective stance classification, where the input is a claim-perspective pair, mapped to {support, oppose}. The candidate inputs are generated on the collection of perspectives $P(c)$ relevant to a claim c. To have an understanding of a lower bound for the metric, we measure the quality of an always-support baseline. We measure the performance of BERT on this task as well, which is about 20% below human performance. This might be because this task requires a deep understanding of *commonsense* knowledge/reasoning (as indicated earlier in Section 5). Since a retrieval system is unlikely to distinguish perspectives with different stances, we do not report the IR performance for this task.

5.3.3 Perspective equivalence (T3)

We create instances in the form of (p_1, p_2, c) where $p_1, p_2 \in P(c)$. The expected label is whether the two perspectives belong to the same equivalence class or not. In the experiments, we observe that BERT has a significant performance gain of $\sim 36\%$ over the IR baseline. Meanwhile, this system is behind human performance by a margin of $\sim 20\%$.

5.3.4 Extraction of supporting evidence (T4)

We evaluate the systems on the extraction of items from the pool of evidences $\mathcal{U}^e$, given a *claim-perspective* pair. To measure the performance of the IR system working with the index containing $\mathcal{U}^e$ we issue a query containing the concatenation of a perspective-claim pair. Given the sorted results (according to their retrieval confidence score), we select the top candidates using a threshold parameter tuned on the dev set. We also use the IR system's candidates (top-60) for other baselines. This set of candidates yields a $>85\%$ recall (for the PR-curve, see Figure 6 in the Appendix). We train BERT system to map each (gold) *claim-perspective* pair to its corresponding *evidence* paragraph(s). Since each evidence paragraph could be long (hence hard to feed into BERT), we split each evidence paragraph into sliding windows of 3 sentences. For each *claim-perspective* pair, we use all 3-sentences windows of gold evidence paragraphs as positive examples, and rest of the IR candidates as negative examples. In the run-time, if a certain percentage (tuned on the dev set) of the sentences from a given evidence paragraph are predicted as positive by BERT, we consider the whole evidence as positive (i.e. it supports a given *perspective*).

Setting	Target set	System	*Pre.*	*Rec.*	$F1$
T1: Perspective relevance	$\mathcal{U}^p$	IR	46.8	34.9	40.0
		IR + BERT	47.3	54.8	**50.8**
		IR + Human	63.8	83.8	72.5
T2: Perspective stance	$P(c)$	Always "supp."	51.6	100.0	68.0
		BERT	70.5	71.1	**70.8**
		Human	91.3	90.6	90.9
T3: Perspective equivalence	$P(c)^2$	Always "¬equiv."	100.0	11.9	21.3
		Always "equiv."	20.3	100.0	33.7
		IR	36.5	36.5	36.5
		BERT	85.3	50.8	**63.7**
		Human	87.5	80.2	83.7
T4: Evidence extraction	$\mathcal{U}^e$	IR	42.2	52.5	46.8
		IR + BERT	69.7	46.3	**55.7**
		IR + Human	70.8	53.1	60.7
T5: Overall	$\mathcal{U}^p$,$\mathcal{U}^e$	IR	-	-	12.8
		IR + BERT	-	-	**17.5**
		IR + Human	-	-	40.0

Table 3: Quality of different baselines on different subtasks (Section 5). All the numbers are in percentage. Top machine baselines are in **bold**.

Overall, the performances on this task are lower, which could probably be expected, considering the length of the evidence paragraphs. Similar to the previous scenarios, the BERT solver has a significant gain over a trivial baseline, while standing behind human with a significant margin.

6 Discussion

As one of the key consequences of the information revolution, *information pollution* and *over-personalization* have already had detrimental effects on our life. In this work, we attempt to facilitate the development of systems that aid in better organization and access to information, with the hope that the access to more diverse information can address over-personalization too (Vydiswaran et al., 2014).

The dataset presented here is not intended to be *exhaustive*, nor does it attempt to reflect a true distribution of the important claims and perspectives in the world, or to associate any of the perspective and identified evidence with levels of expertise and trustworthiness. Moreover, it is important to note that when we ask crowd-workers to evaluate the validity of perspectives and evidence, their judgement process can potentially be influenced by their prior beliefs (Markovits and Nantel, 1989). To avoid additional biases introduced in the process of dataset construction, we try to take the least restrictive approach in filtering dataset content beyond the necessary quality assurances. For this reason, we choose not to explicitly ask annotators to filter contents based on the intention of their creators (e.g. offensive content).

A few algorithmic components were not addressed in this work, although they are important to the complete *perspective discovery and presentation* pipeline. For instance, one has to first verify that the input to the system is a reasonably well-phrased and an argue-worthy claim. And, to construct the pool of perspectives, one has to extract relevant arguments (Levy et al., 2014). In a similar vein, since our main focus is the study of the relations between *claim*s, *perspective*s, and *evidence*, we leave out important issues such as their degree of factuality (Vlachos and Riedel, 2014) or trustworthiness (Pasternack and Roth, 2014, 2010) as separate aspects of problem.

We hope that some of these challenges and limitations will be addressed in future work.

7 Conclusion

The importance of this work is three-fold; we define the problem of *substantiated perspective discovery* and characterize language understanding tasks necessary to address this problem. We combine online resources, web data and crowdsourcing and create a high-quality dataset, in order to drive research on this problem. Finally, we build and evaluate strong baseline supervised systems for this problem. Our hope is that this dataset would bring more attention to this important problem and would speed up the progress in this direction.

There are two aspects that we defer to future work. First, the systems designed here assumed that the input are valid claim sentences. To make use of such systems, one needs to develop mechanisms to recognize valid argumentative structures. In addition, we ignore trustworthiness and credibility issues, important research issues that are addressed in other works.

Acknowledgments

The authors would like to thank Jennifer Sheffield, Stephen Mayhew, Shyam Upadhyay, Nitish Gupta and the anonymous reviewers for insightful comments and suggestions. This work was supported in part by a gift from Google and by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68.

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit Segmentation of Argumentative Texts. In *Proceedings of the Fourth Workshop on Argument Mining (ArgMining 2017)*, pages 118–128.

Tariq Alhindi, Savvas Petridis, Smar Muresan, and a. 2018. Where is your Evidence: Improving Fact-checking by Justification Modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 85–90.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 251–261.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2008. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 208–212. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2018. Five Years of Argument Mining: a Data-driven Analysis. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5427–5433.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzoto. 2013. Recognizing textual entailment: Models and applications.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 987–996. Association for Computational Linguistics.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

James B Freeman. 2011. *Argument Structure:: Representation and Theory*, volume 18. Springer Science & Business Media.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. 2018. A Retrospective Analysis of the Fake News Challenge Stance Detection Task. *arXiv preprint arXiv:1806.05180*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the ACL (Volume 1: Long Papers)*.

James J Heckman. 1979. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society*, pages 153–161.

Xinyu Hua and Lu Wang. 2017. Understanding and Detecting Supporting Arguments of Diverse Types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 203–208.

Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. 2018. Multi-Source Multi-Class Fake News Detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Conference of the North American Chapter of the Association for Computational Linguistics (Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL))*.

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500.

Marco Lippi and Paolo Torroni. 2016. Argument Mining from Speech: Detecting Claims in Political Debates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 2979–2985.

Henry Markovits and Guilaine Nantel. 1989. The belief-bias effect in the production and evaluation of logical conclusions. *Memory and Cognition*, 17(1):11–17.

Tanushree Mitra and Eric Gilbert. 2015. CREDBANK: A Large-Scale Social Media Corpus With Associated Credibility Annotations. In *ICWSM*, pages 258–267. AAAI Press.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 372–387. Springer.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38.

Jeff Pasternack and Dan Roth. 2010. Knowing what to believe (when you already know something). In *Proc. of the International Conference on Computational Linguistics (COLING)*, Beijing, China.

Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *Proc. of the International World Wide Web Conference (WWW)*.

Jeff Pasternack and Dan Roth. 2014. Judging the veracity of claims and reliability of sources with fact-finders. In Anwitaman Datta Xin Liu and Ee-Peng Lim, editors, *Computational Trust Models and Machine Learning*, pages 39–72. Chapman and Hall/CRC.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450.

Mehdi Samadi, Partha Pratim Talukdar, Manuela M Veloso, and Manuel Blum. 2016. ClaimEval: Integrated and Flexible Framework for Claim Evaluation Using Credibility of Sources. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 222–228.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 551–557.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. 2017. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 217–226.

Simone Teufel et al. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, Citeseer.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 809–819.

Stephen E Toulmin. 2003. *The uses of argument*. Cambridge university press.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

V.G.Vinod Vydiswaran, Chengxiang Zhai, Dan Roth, and Peter Pirolli. 2014. Overcoming bias to learn about controversial topics. *Journal of the American Society for Information Science and Technology (JASIST)*.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59.

William Yang Wang. 2017. " Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 422–426.

Wenpeng Yin and Dan Roth. 2018. Twowingos: A two-wing optimization strategy for evidential claim verification. In *EMNLP*.

Fan Zhang, Homa B Hashemi, Rebecca Hwa, and Diane Litman. 2017. A corpus of annotated revisions for studying argumentative writing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1568–1578.

A Supplemental Material

A.1 Statistics

We provide brief statistics on the sources of different content in our dataset in Table 4. In particular, this table shows:

1. the size of the data collected from online debate websites (step 1).
2. the size of the data filtered out (step 2a).
3. the size of the perspectives added by paraphrases (step 2b).
4. the size of the perspective candidates added by web (step 2c).

	Website	# of claims	# of perspectives	# of evidences
after step 1	`idebate`	561	4136	4133
	`procon`	50	960	953
	`debatewise`	395	3039	3036
	total	1006	8135	8122
after step 2a	`idebate`	537	2571	–
	`procon`	49	619	–
	`debatewise`	361	1462	–
	total	947	4652	–
step 2b	paraphrases	–	4507	–
step 2c	web perspectives	–	2427	–

Table 4: The dataset statistics (See section 4.1).

A.2 Measure of agreement

We use the following definition formula in calculation of our measure of agreement. For a fixed subject (problem instance), let n_j represent the number of raters who assigned the given subject to the j-th category. The measure of agreement is defined as

$$\rho \triangleq \frac{1}{n(n-1)} \sum_{j=1}^{k} n_j(n_j - 1)$$

where for $n = \sum_{j=1}^{k} n_j$. Intuitively, this function measure concentration of values the vector $(n_1, ..., n_k)$. Take the edge cases:

- Values concentrated: $\exists j, n_j = n$ (in other words $\forall i \neq j, n_i = 0$) $\Rightarrow P = 1.0$.
- Least concentration (uniformly distribution): $n_1 = n_2 = ... = n_k \Rightarrow \rho = 0.0$.

This definition is used in calculation of more extensive agreement measures (e.g, Fleiss' kappa (Fleiss and Cohen, 1973)). There multiple ways of interpreting this formula:

- It indicates how many rater–rater pairs are in agreement, relative to the number of all possible rater–rater pairs.
- One can interpret this measure by a simple combinatorial notions. Suppose we have sets $A_1, ...A_k$ which are pairwise disjunct and for each j let $n_j = |A_j|$. We choose randomly two elements from $A = A_1 \cup A_2 \cup ... \cup A_k$. Then the probability that they are from the same set is the expressed by ρ.
- We can write ρ in terms of $\sum_{i=1}^{k}(n_i - n/k)^2/(n/k)$ which is the conventional *Chi-Square statistic* for testing if the vector of n_i values comes from the all-categories-equally-likely flat multinomial model.

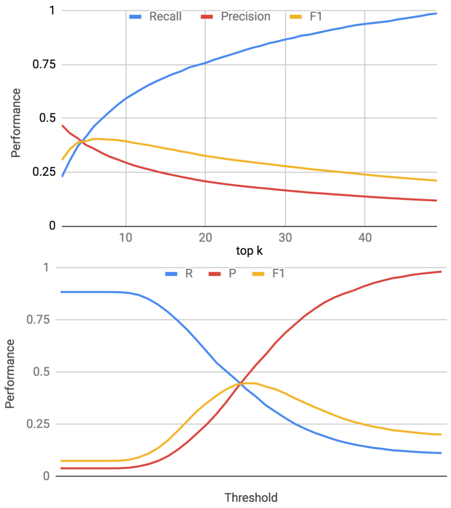


Figure 6: Candidates retrieved from IR baselines vs Precision, Recall, F1, for T1 and T4 respectively.

A.3 crowdsourcing interfaces

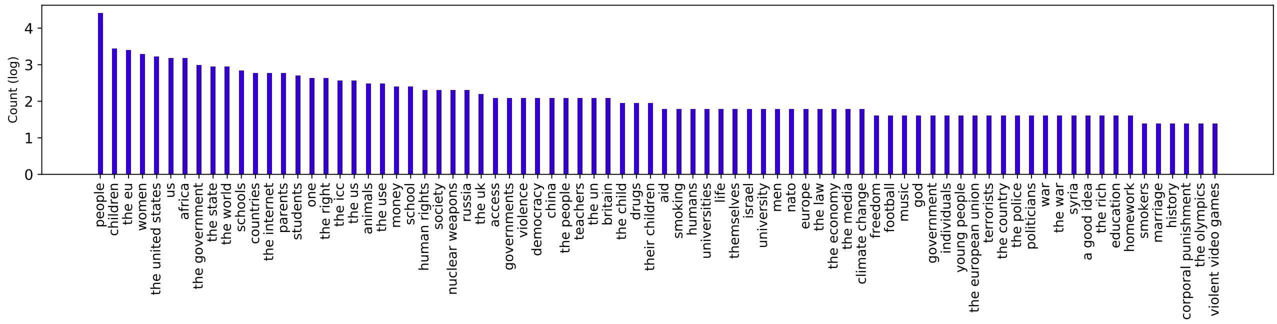


Figure 7: Histogram of popular noun-phrases in our dataset. The y-axis shows count in logarithmic scale.

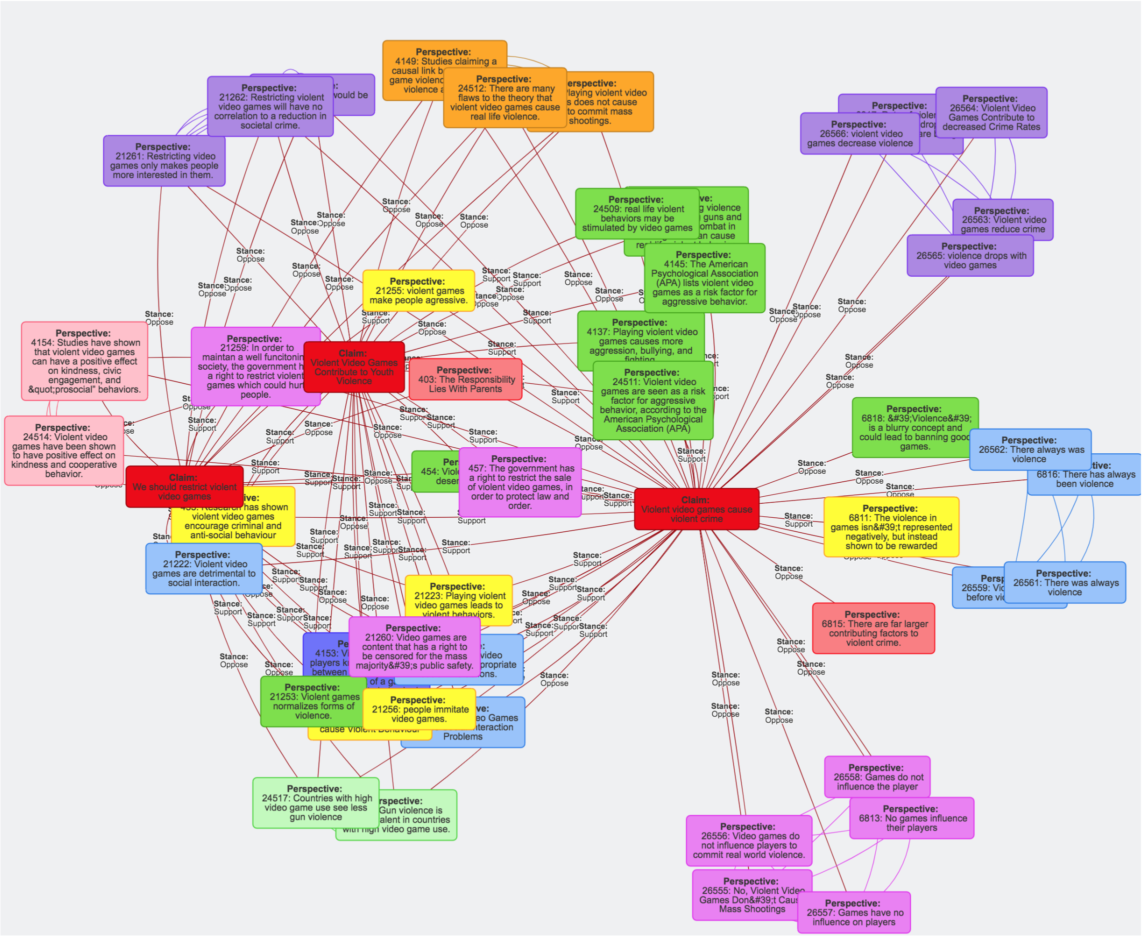
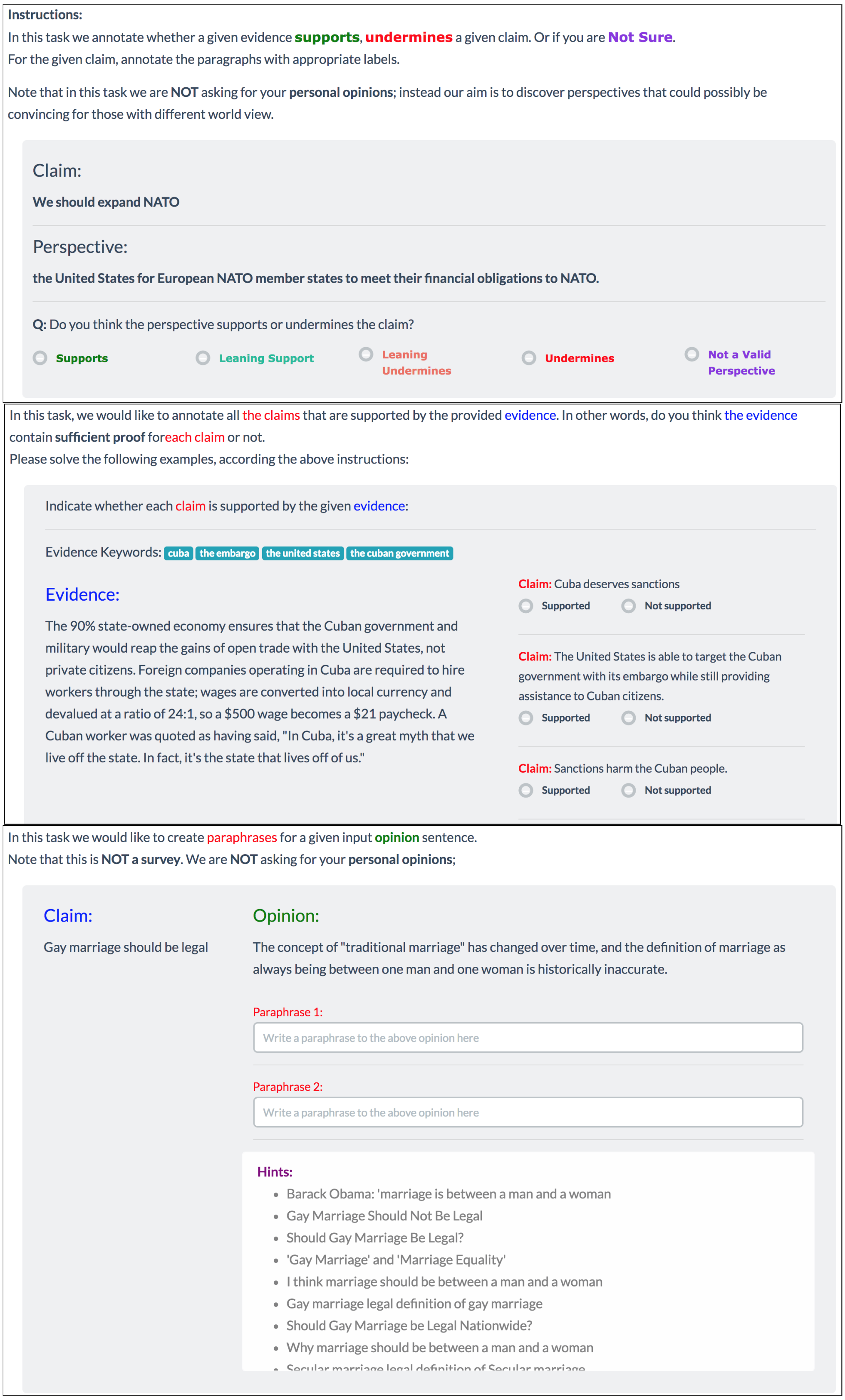


Figure 8: Graph visualization of three related example *claims* (colored in red) in our dataset with their *perspectives*. Each edge indicates a supporting/opposing relation between a perspective and a claim.



Instructions:
In this task we annotate whether a given evidence **supports**, **undermines** a given claim. Or if you are **Not Sure**.
For the given claim, annotate the paragraphs with appropriate labels.

Note that in this task we are **NOT** asking for your **personal opinions**; instead our aim is to discover perspectives that could possibly be convincing for those with different world view.

Claim:

We should expand NATO

Perspective:

the United States for European NATO member states to meet their financial obligations to NATO.

Q: Do you think the perspective supports or undermines the claim?

○ **Supports** ○ **Leaning Support** ○ **Leaning Undermines** ○ **Undermines** ○ **Not a Valid Perspective**

In this task, we would like to annotate all the claims that are supported by the provided evidence. In other words, do you think the evidence contain **sufficient proof** foreach claim or not.
Please solve the following examples, according the above instructions:

Indicate whether each claim is supported by the given evidence:

Evidence Keywords: cuba the embargo the united states the cuban government

Evidence:

The 90% state-owned economy ensures that the Cuban government and military would reap the gains of open trade with the United States, not private citizens. Foreign companies operating in Cuba are required to hire workers through the state; wages are converted into local currency and devalued at a ratio of 24:1, so a $500 wage becomes a $21 paycheck. A Cuban worker was quoted as having said, "In Cuba, it's a great myth that we live off the state. In fact, it's the state that lives off of us."

Claim: Cuba deserves sanctions
○ **Supported** ○ **Not supported**

Claim: The United States is able to target the Cuban government with its embargo while still providing assistance to Cuban citizens.
○ **Supported** ○ **Not supported**

Claim: Sanctions harm the Cuban people.
○ **Supported** ○ **Not supported**

In this task we would like to create paraphrases for a given input opinion sentence.
Note that this is **NOT a survey**. We are **NOT** asking for your **personal opinions**;

Claim:

Gay marriage should be legal

Opinion:

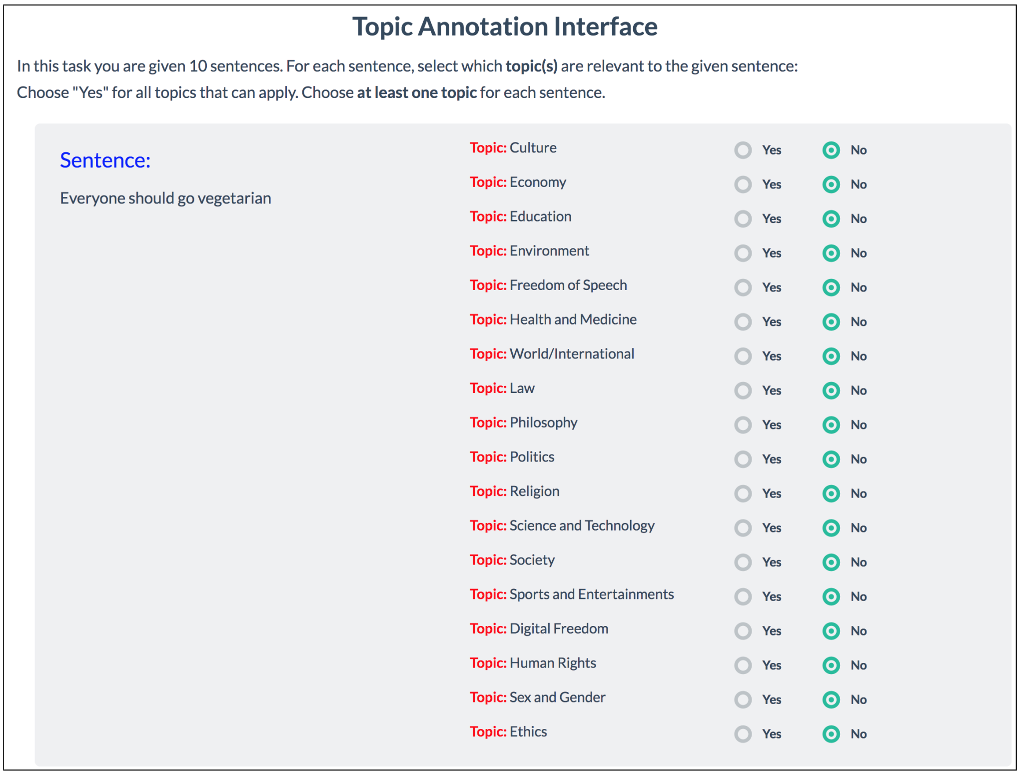
The concept of "traditional marriage" has changed over time, and the definition of marriage as always being between one man and one woman is historically inaccurate.

Paraphrase 1:
Write a paraphrase to the above opinion here

Paraphrase 2:
Write a paraphrase to the above opinion here

Hints:
- Barack Obama: 'marriage is between a man and a woman
- Gay Marriage Should Not Be Legal
- Should Gay Marriage Be Legal?
- 'Gay Marriage' and 'Marriage Equality'
- I think marriage should be between a man and a woman
- Gay marriage legal definition of gay marriage
- Should Gay Marriage be Legal Nationwide?
- Why marriage should be between a man and a woman
- Secular marriage legal definition of Secular marriage

Figure 9: Interfaces shown to the human annotators. Top: the interface for verification of perspectives (step 2a). Middle: the interface for annotation of evidences (step 3a). Bottom: the interface for generation of perspective paraphrases (step 2b).



Topic Annotation Interface

In this task you are given 10 sentences. For each sentence, select which **topic(s)** are relevant to the given sentence:
Choose "Yes" for all topics that can apply. Choose **at least one topic** for each sentence.

Sentence:

Everyone should go vegetarian

Topic: Culture ○ Yes ● No
Topic: Economy ○ Yes ● No
Topic: Education ○ Yes ● No
Topic: Environment ○ Yes ● No
Topic: Freedom of Speech ○ Yes ● No
Topic: Health and Medicine ○ Yes ● No
Topic: World/International ○ Yes ● No
Topic: Law ○ Yes ● No
Topic: Philosophy ○ Yes ● No
Topic: Politics ○ Yes ● No
Topic: Religion ○ Yes ● No
Topic: Science and Technology ○ Yes ● No
Topic: Society ○ Yes ● No
Topic: Sports and Entertainments ○ Yes ● No
Topic: Digital Freedom ○ Yes ● No
Topic: Human Rights ○ Yes ● No
Topic: Sex and Gender ○ Yes ● No
Topic: Ethics ○ Yes ● No

Figure 10: Annotation interface used for topic of claims (Section 4.2)

IMHO Fine-Tuning Improves Claim Detection

Tuhin Chakrabarty
Dept. Of Computer Science
Columbia University
tc2896@columbia.edu

Christopher Hidey
Dept. Of Computer Science
Columbia University
ch3085@columbia.edu

Kathleen McKeown
Dept. Of Computer Science
Columbia University
kathy@cs.columbia.edu

Abstract

Claims are the central component of an argument. Detecting claims across different domains or data sets can often be challenging due to their varying conceptualization. We propose to alleviate this problem by fine tuning a language model using a Reddit corpus of 5.5 million opinionated claims. These claims are self-labeled by their authors using the internet acronyms IMO/IMHO (in my (humble) opinion). Empirical results show that using this approach improves the state of art performance across four benchmark argumentation data sets by an average of 4 absolute F1 points in claim detection. As these data sets include diverse domains such as social media and student essays this improvement demonstrates the robustness of fine-tuning on this novel corpus.

1 Introduction

Toulmin's influential work on argumentation (2003) introduced a claim as an *assertion that deserves our attention*. More recent work describes a claim as *a statement that is in dispute and that we are trying to support with reasons* (Govier, 2010). While some traits of claims are defined by their context, such as that claims usually need some support to make up a 'complete' argument (e.g., premises, evidence, or justifications), the exact definition of a claim may vary depending on the domain, register, or task. Daxenberger et al. (2017) try to solve the problem of claim conceptualization by training models across one data set and testing on others, but their cross-domain claim detection experiments mostly led to decreased results over in-domain experiments.

To demonstrate that some properties of claims are shared across domains, we create a diverse and rich corpus mined from Reddit and evaluate on held out datasets from different sources. We use Universal Language Model Fine-Tuning (ULMFiT) (Howard and Ruder, 2018), which pre-trains a language model (LM) on a large general-domain corpus and fine-tunes it on our Reddit corpus before training a final classifier to identify claims on various data sets.

We make the following contributions:

- We release a dataset of 5.5 million opinionated claims from Reddit,[1] which we hope will be useful for computational argumentation.
- We show transfer learning helps in the detection of claims with varying definitions and conceptualizations across data sets from diverse domains such as social media and student essays.
- Empirical results show that using the Reddit corpus for language model fine-tuning improves the state-of-the-art performance across four benchmark argumentation data sets by an average of 4 absolute F1 points in claim detection.

2 Related Work

Argumentation mining (AM) is a research field within the growing area of computational argumentation. The tasks pursued within this field are highly challenging and include segmenting argumentative and non-argumentative text units, parsing argument structures, and recognizing argumentative components such as claims- the main focus of this work. On the modeling side, Stab and Gurevych (2017) and Persing and Ng (2016) used pipeline approaches for AM, combining parts of the pipeline using integer linear programming (ILP). Eger et al. (2017) proposed state-of-art sequence tagging neural end-to-end models for AM. Schulz et al. (2018) used multi-task learning (MTL) to identify argumentative components,

---

[1] https://bitbucket.org/tuhinch/imho-naacl2019

Proceedings of NAACL-HLT 2019, pages 558–563
Minneapolis, Minnesota, June 2 - June 7, 2019.

challenging assumptions that conceptualizations across AM data sets are divergent and that MTL is difficult for semantic or higher-level tasks.

Rosenthal and McKeown (2012) were among the first to conduct cross-domain experiments for claim detection. However they focused on relatively similar data sets like blog articles from LiveJournal and Wikipedia discussions. Al-Khatib et al. (2016), on the other hand, wanted to identify argumentative sentences through cross-domain experiments. Their goal was, however, to improve argumentation mining via distant supervision rather than detecting differences in the notions of a claim. Daxenberger et al. (2017) showed that while the divergent conceptualization of claims in different data sets is indeed harmful to cross-domain classification, there are shared properties on the lexical level as well as system configurations that can help to overcome these gaps. To this end they carried out experiments using models with engineered features and deep learning to identify claims in a cross-domain fashion.

Pre-trained language models have been recently used to achieve state-of-the-art results on a wide range of NLP tasks (e.g., sequence labeling and sentence classification). Some of the recent works that have employed pre-trained language models include ULMFiT (Howard and Ruder, 2018), ELMo (Peters et al., 2018), GLoMo (Yang et al., 2018), BERT (Devlin et al., 2019) and OpenAI transformer (Radford et al., 2018). While these models have demonstrated success on a variety of tasks, they have yet to be widely used in argumentation mining.

3 Data

As the goal of our experiments is to develop models that generalize across domains, we collect a large, diverse dataset from social media and fine-tune and evaluate on held out data sets.

3.1 Self-labeled Opinion Data Collection

In order to obtain a data set representative of claims, we need a method of automatic data collection that introduces minimal linguistic bias. We thus mine comments containing the acronyms IMO (in my opinion) or IMHO (in my humble opinion) from the social media site Reddit. IM(H)O is a commonly used acronym[2] with the only purpose of identifying one's own comment as a personal opinion. We provide some examples[3] below:

> That's virtually the same as neglect right there **IMHO**.
>
> **IMO**, Lakers are in big trouble next couple years

To use these examples for pre-training, we need only to remove the acronym (and any resulting unnecessary punctuation).

We collect Reddit comments from December 2008 to August 2017 through the pushshift.io API, resulting in 5,569,962 comments. We perform sentence and word tokenization using Spacy. We then extract only the sentence containing IMO or IMHO and discarded the surrounding text. We refer to the resulting collection of comments as the **IMHO** dataset.

3.2 Labeled Claim Data

The IMHO dataset contains no negative examples, only labeled opinions. Furthermore, opinions in this dataset may be only a claim or both a claim and a premise. As our goal is to identify claims, we thus consider four data sets from argumentation mining. As argumentation appears in both monologue and dialogue data, we choose two datasets created from student essays and two from social media. Peldszus and Stede (2016) created a corpus of German microtexts (**MT**) of controlled linguistic and rhetorical complexity. Each document includes a single argument and does not exceed five argumentative components. This corpus was translated to English, which we use for our experiments. The persuasive essay (**PE**) corpus (Stab and Gurevych, 2017) includes 402 student essays. The scheme comprises major claims, claims, and premises at the clause level. This corpus has been used extensively in the argumentation mining community. The corpus from Habernal and Gurevych (2017) includes user-generated web discourse (**WD**) such as blog posts, or user comments annotated with claims and premises as well as backings, rebuttals and refutations. Finally, Hidey et al. (2017) propose a two-tiered annotation scheme to label claims and premises and their semantic types in an online persuasive forum (**CMV**) using a sample of 78 threads from the sub-reddit Change My View, with the long-term goal

---

[2] https://reddit.zendesk.com/hc/en-us/articles/205173295-What-do-all-these-acronyms-mean-

[3] Examples have been modified to protect user privacy

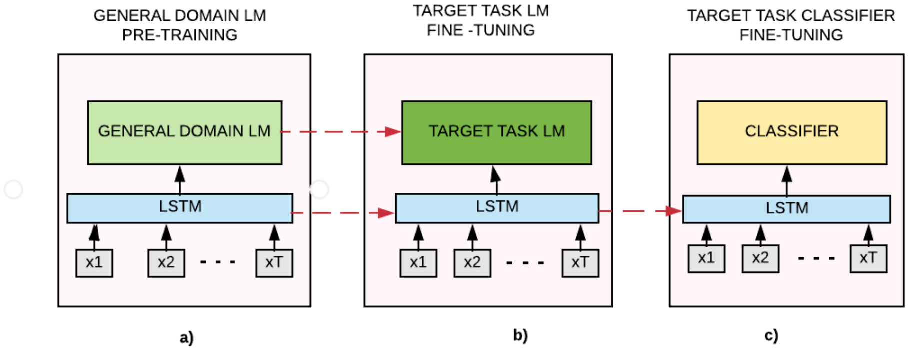


Figure 1: Schematic of ULMFiT, showing three stages. The dashed arrows indicate that the parameters from the previous stage were used to initialize the next stage.

	#Claims	#Sentences	%Claims
MT	112	449	24.94
PE	2108	7116	29.62
WD	211	3899	5.41
CMV	1206	3541	34.0

Table 1: Table showing number of claims and total number of sentences in the data sets along with the percentage of claims in them

of understanding what makes a message persuasive. As with Daxenberger et al. (2017), we model claim detection at the sentence level, as this is the only way to make all data sets compatible to each other. Table 1 gives an overview of the data.

4 Model

As the IMHO dataset is only self-labeled with claim data but does not contain non-claims, we need a method of incorporating this dataset into a claim detection model. We thus use a language model fine-tuning approach, which requires only data similar to the task of interest.

The Universal Language Model Fine-Tuning method (ULMFiT) (Howard and Ruder, 2018) consists of the following steps: a) General-domain LM pre-training b) Task-specific LM fine-tuning and c) Task-specific classifier fine-tuning. In step (a), the language model is trained on Wikitext-103 (Merity et al., 2017) consisting of 28,595 preprocessed Wikipedia articles and 103 million words capturing general properties of language. Step (b) fine-tunes the LM on task-specific data, as no matter how diverse the general-domain data used for pre-training is, the data of the target task will likely come from a different distribution. In step (c), a classifier is then trained on the target task, fine-tuning the pre-trained LM but with an additional layer for class prediction. The models all use a stacked LSTM to represent each sentence. For stages (a) and (b), the output of the LSTM is used to make a prediction of the next token and the parameters from stage (a) are used to initialize stage (b). For stage (c), the model is initialized with the same LSTM but with a new classifier layer given the output of the LSTM.

This process is illustrated in Figure 1. We refer the reader to Howard and Ruder (2018) for further details.

In our work, we maintain steps (a) and (c) but modify step (b) so that we fine-tune the language model on our IMHO dataset rather than the task-specific data. The goal of ULMFiT is to allow training on small datasets of only a few hundred examples, but our experiments will show that fine-tuning the language model on opinionated claims improves over only task-specific LM fine-tuning.

5 Results and Experiments

Table 2 show the results on the four data sets. We compare to two baselines. The numbers in the **CNN** column are taken directly from the results of the deep learning experiments mentioned in the work of Daxenberger et al. (2017). Their deep learning experiments consisted of 4 different models: a) bidirectional LSTM b) LSTM c) CNN initialized with random word embeddings and d) CNN initialized with word2vec. In their experiments for MT and PE, a CNN initialized with random word embeddings gave the best results and for WD a CNN with word2vec gave the best results. As CMV is a new data set we experimented with all four models and obtained the best result using a CNN with random initialization. The **Task-Specific LM Fine-Tuning** column

	Metric	CNN		Task-Specific LM Fine-Tuning		IMHO LM Fine-Tuning	
		Claim	Macro	Claim	Macro	Claim	Macro
WD	P	50.0	72.5	50.0	72.5	**54.0**	**75.9**
	R	20.4	59.2	20.0	59.8	**24.0**	**61.7**
	F	28.9	62.6	28.5	62.7	**33.3**	**65.2**
MT	P	66.5	79.0	66.2	78.5	**71.0**	**80.9**
	R	68.2	78.5	68.0	77.8	**71.8**	**81.4**
	F	67.3	78.6	67.0	78.1	**71.2**	**81.1**
PE	P	60.9	73.2	62.3	73.2	**62.6**	**74.4**
	R	61.2	74.0	65.8	75.1	**66.0**	**75.0**
	F	61.1	73.6	64.0	74.1	**64.3**	**74.8**
CMV	P	54.0	65.1	55.0	68.0	**55.7**	**69.5**
	R	53.0	62.5	59.0	65.0	**60.0**	**65.3**
	F	53.5	63.8	57.0	66.4	**57.8**	**67.3**

Table 2: Table showing the results on four data sets. Each cell contains the Precision (P), Recall (R) and F-score (F) for Claims as well as the Macro Precision, Recall and F-score for the binary classification.

contains the results obtained by fine-tuning the language model on each respective dataset while the **IMHO LM Fine-Tuning** column contains the results from fine-tuning the language model on IMHO. As in previous work, we report both Claim F1 and Macro F1.

The experiments were carried out in a 10-fold cross-validation setup with fixed splits into training and test data and the F1 scores are averaged over each of the folds. Each model was run 10 times to account for variance and the results reported in the table are an average of 10 runs. We use the same hyper-parameters as Howard and Ruder (2018) except for a batch size of 32 for MT and 64 for the remaining data sets. The learning rate for classifier fine-tuning is set to 0.0001. We train our classifier for 5 epochs on each data set.

We obtain statistically significant results ($p < 0.05$ using Chi Squared Test) over all CNN models trained only on the task-specific datasets. We also find that for all models, IMHO LM Fine-Tuning even performs better than Task-Specific LM Fine-Tuning, and is significantly better for the MT and WD datasets (which both contain very few claims). For the MT and WD datasets, Task-Specific LM Fine-Tuning actually performs worse than the CNN models.

6 Qualitative Analysis

To understand how using the IMHO dataset improved over the CNN and Task-Specific Fine-Tuning settings, we show examples that were incorrectly classified by the two baseline models but correctly classified by the IMHO Fine-Tuning. We retrieve the most similar example in the IMHO dataset to these misclassified samples according to TF-IDF over unigrams and bigrams. Table 3 presents the examples labeled by their dataset and the corresponding IMHO example. We find that the IMHO dataset contains n-grams indicative of claims, e.g. *can be very rewarding*, *should be taken off the market*, and *should intervene*, demonstrating that the IMHO LM Fine-Tuning learns representations of claims based on discriminatory phrases. In fact, the CMV example is almost an exact paraphrase of the IMHO example, differing only in the phrase *anecdotal evidence* compared to *my anecdotal experience*. At the same time, we find that many of the topics in these datasets occur in the IMHO dataset as well, such as *public schooling* and *licence fees*, suggesting that the language model learns a bias towards topics as well.

While empirical results indicate that IMHO Fine-Tuning helps in claim detection, we also investigated whether the language model introduces any bias towards types of claims. To this end, we also evaluated examples classified incorrectly by the model. Table 4 shows sentences that are predicted to be opinionated claims by our model but are actually non-claims. We note that a portion of these misclassified examples were premises used to back a claim which could be classified correctly given additional context. For instance, the second example from the MT data set in the table backs

Dataset	Sentence
MT	If there must be rent increases , there should also be a cap to avoid nasty surprises
MT	Video games namely FIFA in my case , can fascinate young people for hours more intensively and emotionally than any sport in the world !
PE	Last but not the least , using public transportation is much safer than using private transportation
PE	In a positive point of view , when people without jobs have hand phones that have access to the internet , they will be able to browse the net for more job opportunities
CMV	Cheating is evidence , that *something* must be wrong

Table 4: Sentences which are actually non-claims but predicted as claims by IMHO Fine-Tuning

Dataset	Sentence
WD	I send my daughter to public school but if I could afford to I would definitely send her to a nearby private school and not have to deal with lots of the problems in public schools.
IMHO	There is no telling that a private school will be better than public, that 's a parents choice, I pulled my kid from private school and went to public school that choice was made because the school we had access to was new and he excellent ratings and it was superior to the private school.
MT	That's why they should be taken off the market, unless they're unbreakable .
IMHO	Should be taken off the market.
MT	The Tv/Radio licence fee can only be required of all citizens/households equally.
IMHO	Radio 4 and Radio 6 music are pretty much worth the licence fee.
MT	Since, however, in Russia besides gas and oil only propaganda and corruption rule, the EU should intervene right away.
IMHO	Neither Russia or the EU should intervene in this case
CMV	Other than anecdotal evidence, I haven't seen anything to support this claim.
IMHO	I have personally seen no evidence to support this claim, but that's just my anecdotal experience .
PE	However, flourishing tourism in a place can be very rewarding in terms of local economy.
IMHO	It can be very rewarding.

Table 3: Sentences from each dataset and their nearest neighbor in the IMHO dataset

the claim *It would be fair to make them into an Olympic event* while the first example from the PE data set backs the claim *There is no reason that governments should hesitate to invest in public transportation, a healthy, safe and economical way of transporting*. While discriminatory phrases like *should* or *must be* and comparative statements like *much safer than* or *more ... than any* are often indicative of claims, the lack of context may lead to incorrect classifications. Language modeling with additional context sentences or jointly modeling context (e.g. by predicting relations between claims and premises) may address these errors.

7 Conclusion

We have collected a large dataset of over 5 million self-labeled opinionated claims and validated their utility on a variety of claim detection domains. Second, we demonstrate that by fine-tuning the language model on our IMHO dataset rather than each individual claim dataset, we obtain statistically significant improvement over previous state-of-the-art performance on each of these datasets on claim detection. Finally, our empirical results and error analysis show that there are features indicative of claims that transfer across data-sets.

In the future, we plan to expand this work beyond single sentences as the data-set for LM Fine-Tuning used in our experiments consists of sentences containing IM(H)O without additional context. We plan to experiment with modeling the context sentences from Reddit as well by using models such as BERT (Devlin et al., 2019), which perform well on pair classification tasks, as the fine-tuning step rather than ULMFiT. As BERT pre-training includes a next-sentence prediction task, we expect this model to be effective for modeling argumentative context and to be beneficial for predicting premise or justifications for these claims and the relations between these argumentative components.

Acknowledgements

The authors thank Johannes Daxenberger for sharing the sentence level data from their previous experiments and Smaranda Muresan for her insightful discussions on improving the contributions of the paper as well as thoughtful ideas on how to conduct a good error analysis. The authors also thank Olivia Winn for suggesting the paper title and the anonymous reviewers for helpful comments.

References

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Kohler, and Benno Stein. 2016. Domain mining of argumentative text through distant supervision. In *15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066.

Jacob Devlin, Ming-Wei Chang, Lee Kenton, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 17th Annual Meeting of the North American Association for Computational Linguistics*.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.*, pages 11–22.

Trudy Govier. 2010. *A Practical Study of Argument*. 7th edition Cengage Learning, Wadsworth.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. In *Computational Linguistics, 43(1)*, pages 125–179.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathleen McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *In Proceedings of the 4th Workshop on Argument Mining. EMNLP*, pages 11–21.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 328–339.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. In proceedings of the international conference on learning representations.

Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In *In Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation*, pages 810–815.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *In Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*, pages 1384–1394.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Sara Rosenthal and Kathy McKeown. 2012. Detecting opinionated claims in online discussion. In *Inproceedings of the Sixth IEEE International Conference on Semantic Computing (IEEE ICSC2012) Special Session on Semantics and Sociolinguistics in Social Medi*.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multitask learning for argumentation mining in low-resource settings. In *Proceedings of NAACL-HLT 2018*, pages 35–41.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. In *Computational Linguistics, pages in press, preprint available at arXiv:1604.07370.*

Stephen E. Toulmin. 2003. *The Uses of Argument*. Cambridge University Press, New York.

Zhilin Yang, Jake Zhao, Bhuwan Dhingra, Kaiming He, William W. Cohen, Ruslan Salakhutdinov, and Yann LeCun. 2018. Glomo: Unsupervisedly learned relational graphs as transferable. In *arXiv:1806.05662*.

Joint Multiple Intent Detection and Slot Labeling for Goal-Oriented Dialog

Rashmi Gangadharaiah
AWS AI, Amazon
`rgangad@amazon.com`

Balakrishnan Narayanaswamy
AWS AI, Amazon
`muralibn@amazon.com`

Abstract

Neural network models have recently gained traction for sentence-level intent classification and token-based slot-label identification. In many real-world scenarios, users have multiple intents in the same utterance, and a token-level slot label can belong to more than one intent. We investigate an attention-based neural network model that performs multi-label classification for identifying multiple intents and produces labels for both intents and slot-labels at the token-level. We show state-of-the-art performance for both intent detection and slot-label identification by comparing against strong, recently proposed models. Our model provides a small but statistically significant improvement of 0.2% on the predominantly single-intent ATIS public data set, and 55% intent accuracy improvement on an internal multi-intent dataset.

1 Introduction

In dialog systems, the natural language understanding component (NLU) is responsible for identifying the user's request and creating a semantic frame that succinctly summarizes the user's needs. These semantic frames are typically constructed using *intents* and *slot-labels* (Tür et al., 2010). As the names imply, an intent captures the intention of the user and slot-labels capture any additional information or constraints the user provides. These constraints must be satisfied in order to fulfill the user's request. The example below shows a user's request, *"how is the weather in Dallas ?"*. We need to identify the intent (*"GetWeatherInfo"*) as well as the values for the slot-labels (SL), here, *"City"* (value=*"Dallas"*). It is crucial that intents and slot-labels are identified with high accuracy as an error made by the NLU component propagates through downstream components such as the dialog state tracker, the dialog policy and the natural language generator components, leading to a substantial degradation of the performance of the entire dialog system.

NLU Semantic Frame

	how	is	the	weather	in	Dallas	?
	↕	↕	↕	↕	↕	↕	↕
SL:	O	O	O	O	O	City	O

Intent: GetWeatherInfo

Intent detection has been modeled as a sentence classification task where an intent (y^I) is assigned to the user's utterance. Slot labeling is typically modeled as a sequential labeling problem, where a user's sentence, $x_1, x_2, ...x_N$, is labeled with $y_1^S, y_2^S, ..y_N^S$, and y_i^S is the slot label assigned to the token at position i (x_i). In the example above, the sequence of slot labels would be, *"O O O O O City O"*, where, *"O"* stands for *"Other"*.

Sequential models such as Maximum Entropy Markov models (Toutanova and Manning, 2000; McCallum et al., 2000; Berger et al., 1996) and Conditional Random Fields, CRFs (Lafferty et al., 2001; Jeong and Geunbae Lee, 2008) are popular approaches for slot-labeling while intent prediction is often performed using standard classification approaches such as Support Vector Machines (Cortes and Vapnik, 1995) or logistic regression (Bishop, 2006). More recently, neural network-based models (Mesnil et al., 2015; Kurata et al., 2016; Goo et al., 2018; Liu and Lane, 2016) have been shown to significantly outperform previous approaches. These models are also appealing since a single model is trained end-to-end to perform both intent detection and slot label identification. Jointly modeling intent and slot label identification (Liu and Lane, 2016; Goo et al., 2018) has been shown to significantly outperform other neural network-based approaches. This is intuitive since slot labels could depend on the intent.

Proceedings of NAACL-HLT 2019, pages 564–569
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

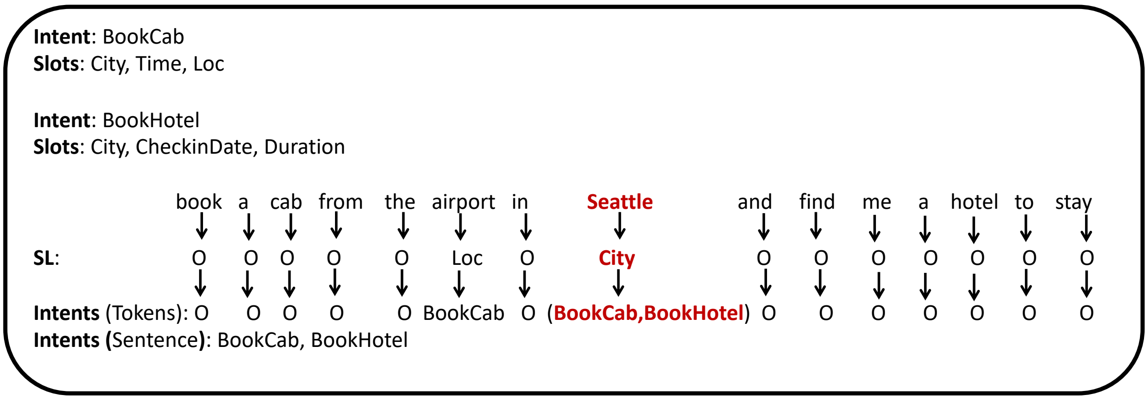


Figure 1: An example showing slot values belonging to multiple intents. Here, *Seattle* belongs to two of the intents in the user's utterance, *BookHotel* and *BookCab*.

Most neural network-based approaches (Mesnil et al., 2015; Kurata et al., 2016; Goo et al., 2018; Liu and Lane, 2016), with the exception of (Xu and Sarikaya, 2013a), predict a single intent for a user's utterance. In real-world scenarios, users indicate multiple intents in the same utterance. For example, a user's utterance such as, *"show me flights from Dallas to New York and the cost"*, clearly has two intents, one for obtaining the price of the flights (*"GetFlightCost"*) and another for the flight information. It is critical to understand and model such scenarios to allow more natural interaction with users. In this paper, we treat the intent detection task as a multi-label classification problem and suggest various neural network models to obtain multiple intents.

Our work is related to Xu et al.,(2013b) and Kim et al.,(2017), where multiple intents are assigned to a user's utterance. Xu et al., (2013b) use log-linear models to achieve this, while we use neural network models. Both Xu et al., (2013b) and Kim et al., (2017) only consider intents and do not handle slot labels. In this paper, we jointly perform multi-label intent classification and slot-label identification.

In contrast with all prior work, we investigate and study the problem of assigning slot labels (or constraints) provided by a user to multiple intents. Consider the example in Figure 1 with two intents in the same domain, *"BookCab"* and *"BookHotel"*. Suppose *"BookCab"* has three possible slot labels, *"City"*, *"Time"* and pick up location (*"Loc"*), and suppose that *"BookHotel"* has slot labels *"City"*, *"CheckinDate"*, and *"Duration"*. Consider a user's utterance, *"book a cab from the airport in Seattle and find me a hotel to stay"*. Here, the user wants to book a cab (*"BookCab"* intent) as well book a hotel (*"BookHotel"*). The slot label *"Seattle"* should be assigned to both intents to accurately capture the user's request. Hence, we study a model that predicts multiple intents both at the token level as well as at the sentence-level.

We model token-level multi-intent classification using Long Short Term Memory (LSTMs) units to capture dependencies that may exist between intents. For example, a user who wants to book a cab is also likely to make a request for a hotel in the same utterance but probably not order food i.e., intents such as *"BookCab"* and *"BookHotel"* are more likely to occur together when compared to *"BookCab"* and *"OrderFood"*. To summarize, the contributions of this paper are:

- We investigate approaches to the problem of multi-intent classification. We perform joint multi-intent classification both at sentence-level and at token-level. We see that,
  - the token-level multi-intents help assign user constraints to the intents.
  - sentence-level multi-intent classification captures dependencies between intents.
- We compare the performance of the approach with recently proposed state-of-the-art approaches and show significant improvement.

The paper is organized as follows. Section 2 describes the proposed approach. Section 3 describes the experimental setup, including, data sets and metrics used to evaluate the approaches followed by the results in Section 3.2. Finally, we conclude and suggest possible future directions and extensions in Section 4.

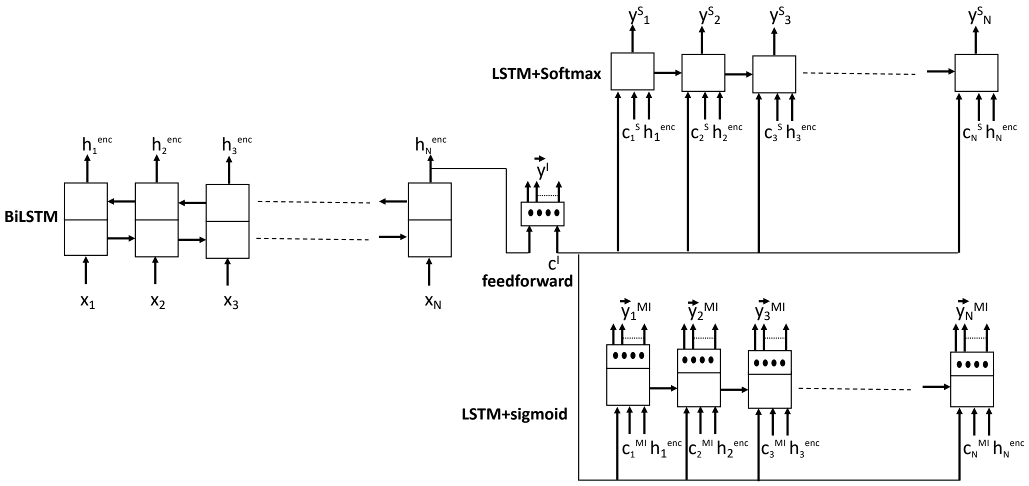


Figure 2: Proposed Approach. A bidirectional LSTM is used for the encoder layer. Multiple intents are predicted both at the sentence level (y^I) and at the token level (y^{MI}). y^I uses a feedforward network. Slot labels (y^S) and token level intent prediction (y^{MI}) both use LSTM layers, which have skip connections to the encoder states.

2 Proposed Approaches

LSTM-based RNN models have become popular for sequential labeling, especially in natural language processing tasks, due to their ability to model long-term dependencies. We extend encoder-decoder architectures from Liu et al., (2016) and Gangadharaiah et al., (2018), which showed superior performance when compared to Convolutional neural network based CRFs (Xu and Sarikaya, 2013a) and other RNN-based architectures (Mesnil et al., 2015; Kurata et al., 2016) for intent detect and slot label identification.

We use a bidirectional LSTM encoder to encode the input word sequence. The encoder hidden state, h_i^{enc}, at each word position is a concatenation of the forward state (fh_i) and backward state (bh_i), $h_i^{enc} = [fh_i, bh_i]$.

For intent detection at the sentence-level, a context vector c^I is computed using the encoder's final hidden state. The vectors, c^I and the final encoder's hidden state vector are sent to a dense layer of sigmoid units to predict the probabilities for every intent. This produces multiple intents ($\vec{y}^I$) as opposed to previous approaches that produce a single intent.

For slot labeling, the decoder also uses LSTMs. At each decoding step i, the decoder state ($h_i^{S,dec}$) is a function of the previous decoder state ($h_{i-1}^{S,dec}$), the previously emitted label (y_{i-1}^S), the encoder's state ($h_i^{S,enc}$), the context vectors, (c_i^S) and c^I, as shown in Figure 2. The context vector c_i^S is a weighted combination of the encoder's states ($h_1^{enc}, h_2^{enc}, ...h_N^{enc}$) with weights, $\alpha_{i,j}^S$, as shown in eqn. 1. g is a feed forward network.

$$c_i^S = \sum_{j=1}^{N} \alpha_{i,j}^S h_j^{enc} \tag{1}$$

$$\alpha_{i,j}^S = \frac{exp(e_{i,j})}{\sum_{k=1}^{N} exp(e_{i,k})}$$

$$e_{i,k} = g(h_{i-1}^{S,dec}, h_k^{enc})$$

The output of the LSTM layer is then sent to a softmax layer to predict the slot labels. We also experimented with a CRF layer as the decoder. In our preliminary experiments, the LSTM decoder was faster to train and also showed better performance when compared to the CRF layer and hence we use LSTMs in the experiments below. We also apply a slot-gated mechanism similar to Goo et al., (2018). The idea is to leverage the intent's context vector for modeling slot-intent relationships, thereby improving the performance of slot labeling. The slot gate is computed as a function of both the slot context vector (c_i^S) and the intent context vector (c^I), where, v and W are both trainable. In Goo et al., (2018), a similar model showed at-par or better performance over Liu et al. (2016) and Tür et. al. (2016). The slot gate gS is defined as,

$$gS = \sum v \cdot tanh(c_i^S + W \cdot c^I) \tag{2}$$

where, gS is used to weight h_i^{enc} and c_i^S to obtain y_i^S, i.e., $h_i^{enc} + c_i^S \cdot gS$ is sent to the feed forward network to compute y_i^S.

Since a slot label can belong to multiple intents, we also perform multi-label intent detection at the token level. We again use an LSTM decoder, where each decoder state, $h_i^{MI,dec}$, is a function of c^I, previous decoder state ($h_{i-1}^{MI,dec}$), the encoder's state (h_i^{enc}) and the context vector (c_i^{MI}), as shown in Figure 2. c_i^{MI} is computed in the same manner as c_i^S. The output of the decoder is sent to a dense layer with sigmoid units. Thus at each word position, we predict multiple intents.

3 Experiments

In all our experiments, we set the hidden vectors to a dimension of 64 and use the adam optimizer with an early stopping strategy. We use a drop-out rate of 0.5 for regularization and the maximum norm for gradient clipping is set to 5. The results are obtained by averaging the performance of the models over 10 runs. To do a fair comparison against existing models, we do not pre-train our word embeddings (Devlin et al., 2018; Pennington et al., 2014; Mikolov et al., 2013), instead we use an embedding layer in the model which is trained along with the rest of the model's parameters.

As done in the NLU community, we report F1 scores for slot labeling. We use F1 scores for intent detection at the token-level and accuracy for sentence-level intent detection.

3.1 Datasets

We use two widely used public datasets, ATIS (Airline Travel Information System) (Tür et al., 2010) and SNIPS [1]. The ATIS dataset contains audio recordings of people requesting flight reservations, with 21 intent types and 120 slot labels. There are 4,478 utterances in the training set, 893 in the test set and 500 utterances in the development set. The SNIPS data was collected from the SNIPS personal voice assistant, with 7 intent types and 72 slot labels. The training set contains 13,084 utterances, the test set contains 700 utterances and the development set also contains 700 utterances. The ATIS dataset contains utterances with multi intents, while the SNIPS is only single intent. In order to demonstrate that our approach does not degrade performance on single intent datasets, we also perform evaluations on the SNIPS dataset.

We also test the performance of the models on an internal dataset. In this dataset, about 52% of examples are multi-intent compared to ATIS which has ≈2% of test examples with multi-intents. The average number of intents per utterance in the internal dataset is 1.6.

3.2 Results

We compare our approach against two of the state of the art approaches that have shown the best performance in previous work. We will use **Model 1** to refer to the model proposed by Liu et al., (2016). **Model 2** refers to the more recent model proposed by Goo et al., (2018). Table **??** shows results obtained by the model investigated in this paper when compared with Model 1 and Model 2.

As mentioned earlier, both Models 1 and 2 only handle single intents per user utterance. For these two models, we insert a # between the multiple intents and treat it as one single intent, i.e., when an example such as, *"please give me a list of all the flights between dallas and baltimore and their cost"*, contains multiple intents, *"atis_flight"* and *"atis_airfare"*, we use *"atis_flight#atis_airfare"* instead. When evaluating the baselines, the ordering of intents does not matter, and so we replace the # with spaces once we have the predictions.

To allow comparison across approaches, both ATIS and SNIPS were modified to include token-level intents as follows. For utterances that had only a single intent, we assigned this intent to all tokens that had a slot label (i.e., to slot labels that do not correspond to O). For utterances that had more than one intent, we assigned all intents to all tokens that had slot labels. After this process, if an utterance had two intents, $intent_1$ and $intent_2$, and if a token i had a slot label, the token would end up with targets of the form,

$$(slot^i, intent_1^i, intent_2^i)$$

The proposed model shows a statistically significant improvement in sentence-level intent prediction (S-level) on ATIS when compared to the two baselines. Any improvement in slot labeling is unclear, since this could be attributed to the architecture changes which involved additional penalty terms on the intent (since we use both token-level and sentence-level intent loss). We also notice that the performance on SNIPS (a single intent dataset) does not degrade. We see a larger performance boost in both token-level (T-level) and sentence-level (S-level) intent detection on the internal dataset due to the large percentage of examples with multi-intents. Wilcoxon signed-rank test

---

[1] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines

Model	ATIS			SNIPS			Internal Dataset		
	Slot (F1)	Intent (Acc) S-level	Intent (F1) T-level	Slot (F1)	Intent (Acc) S-level	Intent (F1) T-level	Slot F1	Intent (Acc) S-level	Intent (F1) T-level
Model 1	90.16	93.84	N/A	87.24	97.14	N/A	89.28	57.27	N/A
Model 2	93.37	95.18	N/A	**88.23**	96.85	N/A	89.64	57.47	N/A
Proposed approach	**94.22**	**95.39**	95.82	88.03	**97.23**	97.89	**90.94**	**89.41**	**94.54**

Table 1: Performance of the model against Model 1 and Model 2. We report F1 scores for slot labeling. For intent detection, we use F1 scores for intent detection at the token-level (T-level) and accuracies (acc) for sentence-level (S-level) intent detection. N/A: as Models 1 and 2 perform single intent detection only at S-level.

(Wilcoxon, 1945) was used to find statistical significance.

4 Conclusion and Future Work

The paper investigated an approach for multi-intent classification. We perform multi-intent classification both at sentence-level and at token-level. The token-level multi-label classification helped assign common constraints (or slot labels) to multiple intents, improving accuracy. The sentence-level multi-intent classification captured dependencies between intents. We compared the performance of our approach with previously proposed state-of-the-art approaches for single intent classification and showed significant improvements in performance on all the datasets.

As future work, we would like to explore other architectures to directly model dependencies between slot labels and intents. This is useful since only a subset of slot labels occur with certain intents. We will also test the proposed approaches against real-world scenarios to understand their generality across various domains.

References

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Rashmi Gangadharaiah, Balakrishnan Narayanaswamy, and Charles Elkan. 2018. What we need to learn if we want to do and not just talk. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, pages 25–32.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757. Association for Computational Linguistics.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*, INTERSPEECH '16. ISCA.

Minwoo Jeong and G. Geunbae Lee. 2008. Triangular-Chain Conditional Random Fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302.

Byeongchang Kim, Seonghan Ryu, and Gary Geunbae Lee. 2017. Two-stage Multi-intent Detection for Spoken Language Understanding. *Multimedia Tools Appl.*, 76(9):11377–11390.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2083. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bing Liu and Ian Lane. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. *CoRR*, abs/1609.01454.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *Trans. Audio, Speech and Lang. Proc.*, 23(3):530–539.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gökhan Tür, Dilek Hakkani-Tür, and Larry P. Heck. 2010. What is left to be understood in ATIS? In *SLT*, pages 19–24. IEEE.

Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

Puyang Xu and Ruhi Sarikaya. 2013a. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *ASRU*, pages 78–83. IEEE.

Puyang Xu and Ruhi Sarikaya. 2013b. Exploiting Shared Information for Multi-intent Natural Language Sentence Classification. In *Interspeech*.

CITE: A Corpus of Image–Text Discourse Relations

Malihe Alikhani
Computer Science
Rutgers University
malihe.alikhani@rutgers.edu

Sreyasi Nag Chowdhury
Max Planck Institute
for Informatics
sreyasi@mpi-inf.mpg.de

Gerard de Melo
Computer Science
Rutgers University
gerard.demelo@rutgers.edu

Matthew Stone
Computer Science
Rutgers University
matthew.stone@rutgers.edu

Abstract

This paper presents a novel crowd-sourced resource for multimodal discourse: our resource characterizes inferences in image–text contexts in the domain of cooking recipes in the form of coherence relations. Like previous corpora annotating discourse structure between text arguments, such as the Penn Discourse Treebank, our new corpus aids in establishing a better understanding of natural communication and common-sense reasoning, while our findings have implications for a wide range of applications, such as understanding and generation of multimodal documents.

1 Introduction

"*Sometimes a picture is worth the proverbial thousand words; sometimes a few well-chosen words are far more effective than a picture*" – Feiner and McKeown (1991). Modeling how visual and linguistic information can jointly contribute to coherent and effective communication is a longstanding open problem with implications across cognitive science. As Feiner and McKeown (1991) already observe, it is particularly important for automating the understanding and generation of text–image presentations. Theoretical models have suggested that images and text fit together into integrated presentations via *coherence relations* that are analogous to those that connect text spans in discourse; see Alikhani and Stone (2018a) and Section 2. This paper follows up this theoretical perspective through systematic corpus investigation.

We are inspired by research on text discourse, which has led to large-scale corpora with information about discourse structure and discourse semantics. The Penn Discourse Treebank (PDTB) is one of the most well-known examples (Miltsakaki et al., 2004; Prasad et al., 2008). However, although multimodal corpora increasingly include discourse relations between linguistic and nonlinguistic contributions, particularly for utterances and other events in dialogue (Cuayáhuitl et al., 2015; Hunter et al., 2015), to date there has existed no dataset describing the coherence of text–image presentations. In this paper, we describe the construction of an annotated corpus that fills this gap, and report initial analyses of the communicative inferences that connect text and accompanying images in this corpus.

As we describe in Section 2, our approach asks annotators to identify the presence of specific inferences linking text and images, rather than to use a taxonomy of coherence relations. This enables us to deal with the distinctive discourse contributions of photographic imagery. We describe our data collection process in Section 3, showing that our annotation scheme allows us to get reliable labels by crowdsourcing. We present analyses in Section 4 that show that our annotation highlights a range of cases where text and images work together in distinctive and theoretically challenging ways, and discuss the implications of our work for the understanding and generation of multimodal documents. We conclude in Section 5 with a number of problems for future research.

2 Discourse Coherence and Text–Image Presentations

We begin with an example to motivate our approach and clarify its relationship to previous work. Figure 1 shows two steps in an online recipe for a ravioli casserole from the RecipeQA data set (Yagcioglu et al., 2018). The image of Figure 1a shows a moment towards the end of carrying out the "covering" action of the accompanying text; that of Figure 1b shows one instance of the result of the "spooning" actions of the text. Cognitive scientists have argued that such images are

Proceedings of NAACL-HLT 2019, pages 570–575
Minneapolis, Minnesota, June 2 - June 7, 2019.

(a) TEXT: Cover with a single layer of ravioli.

(b) TEXT: Let cool 5 minutes before spooning onto individual plates.

Figure 1: Two steps in a recipe from Yagcioglu et al. (2018) illustrating diverse inferential relationships between text and accompanying imagery in instructions. The recipe is from Autodesk Inc. www.instructables.com and is contributed by www.RealSimple.com.

much like text contributions in the way their interpretation connects to the broader discourse. In particular, inferences analogous to those used to interpret text seem to be necessary with such images to recognize their spatio-temporal perspective (Cumming et al., 2017), the objects they depict (Abusch, 2013), and their place in the arc of narrative progression (McCloud, 1993; Cohn, 2013). In fact, such inferences seem to be a general feature of multimodal communication, applying also in the coherent relationships of utterance to co-speech gesture (Lascarides and Stone, 2009) or the coherent relationships of elements in diagrams (Alikhani and Stone, 2018b; Hiippala and Orekhova, 2018).

In empirical analyses of text corpora, researchers in projects such as the Penn Discourse Treebank (Miltsakaki et al., 2004; Prasad et al., 2008) have been successful at documenting such effects by annotating discourse structure and discourse semantics via coherence relations. We would like to apply a similar strategy to text–image documents like that shown in Figure 1. However, existing discourse annotation guidelines depend on the distinctive ways that coherence is signaled in text. In text, we find syntactic devices such as structural parallelism, semantic devices such as negation, and pragmatic elements such as discourse connectives, all of which can help annotators to recognize coherence relations in text. Images lack such features. At the same time, characterizing the communicative role of imagery, particularly photographic imagery, involves a special problem: distinguishing the content that the author specifically aimed to depict from merely incidental details that happen to appear in the scene (Stone and Stojnic, 2015).

Thus, rather than start from a taxonomy of discourse relations like that used in PDTB, we characterize the different kinds of inferential relationships involved in interpreting imagery separately.

- To characterize temporal relationships between imagery and text, we ask if the image gives information about the preparation, execution or results of the accompanying step.
- To characterize the logical relationship of imagery to text, we ask if the image shows one of several actions described in the text, and if it depicts an action that needs to be repeated.
- To characterize the significance of incidental detail, we ask a range of further questions (some relevant specifically to our domain of instructions), asking about what the image depicts from the text, what it leaves out from the text, and what it adds to the text.

Our approach is designed to elicit judgments that crowd workers can provide quickly and reliably.

This approach allows us to highlight a number of common patterns that we can think of as prototypical coherence relations between images and text. Figure 1a, for example, instantiates a natural **Depiction** relation: the image shows the action described in the text in progress; the mechanics of the action are fully visible in the image, but the significant details in the imagery are all reported in the text as well. Our approach also lets us recognize more sophisticated inferential relationships, like the fact that Figure 1b shows an **Example:Result** of the accompanying instruction. Many of the relationships that emerge from our annotation effort involve newly-identified features of text–image presentations that deserve further investigation: particularly, the use of loosely-related imagery to provide background and motivation for a multimodal presentation as a whole, and depictions of action that seem simultaneously to give key information about the context, manner and result of an action.

3 Annotation Effort[1]

Work on text has found that text genre heavily influences both the kinds of discourse relations one finds in a corpus and the way those relations are signalled (Webber, 2009). Since our focus is on developing methodology for consistent annotation, we therefore choose to work within a single genre. We selected instructional text because of its concrete, practical subject matter and because of its step-by-step organization, which makes it possible to automatically group together short segments of related text and imagery.

Text–Image Pairs. We base our data collection on an existing instructional dataset, RecipeQA (Yagcioglu et al., 2018). This is the only publicly available large-scale dataset of multimodal instructions. It consists of multimodal recipes—textual instructions accompanied by one or more images.

We excluded documents that either have multiple steps without images or that have multiple images per set. This was so that we could more easily study the direct relationship between an image and the associated text. There are 1,690 documents with this characteristic in the RecipeQA train set. To avoid overwhelming crowd workers, we further filtered those to retain only recipes with 70 or fewer words per step, for a final count of 516 documents (2,047 image–text pairs).

Protocol. We recruit participants through Amazon Mechanical Turk. All subjects were US citizens, agreed to a consent form approved by Rutgers's institutional review board, and were compensated at an estimated rate of USD 15 an hour.

Experiment Interface. Given an image and the corresponding textual instruction from the dataset, participants were requested to answer the following 10 questions.

For Question 1, participants were asked to highlight the relevant part of the text. For the others, we solicited True/False responses.

1. Highlight the part of the text that is most related to the image.
2. The image gives visual information about the step described in the text.
3. You need to see the image in order to be able to carry out the step properly.
4. The text provides specific quantities (amounts, measurements, etc.) that you would not know just by looking at the picture.
5. The image shows a tool used in the step but *not* mentioned in the text.
6. The image shows how to prepare before carrying out the step.
7. The image shows the results of the action that is described in the text.
8. The image depicts an action in progress that is described in the text.
9. The text describes several different actions but the image only depicts one.
10. One would have to repeat the action shown in the image many times in order to complete this step.

The interface is designed such that if the answer to Question 8 is TRUE, the subject will be prompted with Question 9 and 10. Otherwise, Question 8 is the last question in the list.

Agreement. To assess the inter-rater agreement, we determine Cohen's κ and Fleiss's κ values. For Cohen's κ, we randomly selected 150 image–text pairs and assigned each to two participants, obtaining a Cohen's κ of 0.844, which indicates almost perfect agreement. For Fleiss's κ (Fleiss and Cohen, 1973; Cocos et al., 2015; Banerjee et al., 1999), we randomly selected 50 text–image pairs, assigned them to five subjects, and computed the average κ. We obtain a score of 0.736, which indicates substantial agreement (Viera et al., 2005).

4 Analysis

Overall Statistics. Table 1 shows the rates of true answers for questions Q2–Q10. Subjects reported that in 17% of cases the images did not give any information about the step described in the accompanying text. Such images deserve further investigation to characterize their interpretive relationship to the document as a whole. Our anecdotal experience is that such images sometimes provide context for the recipe, which may suggest that imagery, like real-world events (Hunter et al., 2015), creates more flexible discourse structures than linguistic segments on their own.

Subjects reported that the image was required in order to carry out the instruction only for 6%

---

[1] The dataset and the code for the machine learning experiments are available at https://github.com/malihealikhani/CITE

	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
True	0.829	0.058	0.211	0.131	0.056	0.491	0.209	0.289	0.133

Table 1: Rate of true answers for annotation questions Q2–Q10 across the corpus.

	Q1	Q2**	Q3**	Q4**	Q5	Q6**	Q7**	Q8**	Q9*	Q10**
F1	0.74	0.86	0.76	0.85	0.88	0.92	0.64	0.83	0.77	0.92

Table 2: SVM classification accuracy: bag-of-words features; 80-20 train-test split; 5-fold cross validation. For the first question, this distinguishes highlighted text vs. its complement (excluded vs. included). For the rest of the questions, this distinguishes text of true instances from text of false instances, and is different from majority class baseline * at $p < 0.04, t = -3.5$ and ** at $p < 0.01, t > |2.49|$.

of cases. This suggests that subjects construe imagery as backgrounded or peripheral to the document, much as speakers regard co-speech iconic gesture as peripheral to speech (Schlenker and Chemla, 2017). Note, by contrast, that subjects characterized 12.7% of images as introducing a new tool: this includes many cases where the same subjects say the image is not required. In other words, subjects' intuitions suggest that coherent imagery typically does not contribute instruction content, but rather serves as a visual signal that facilitates inferences that have to be made to carry out the instruction regardless. Our annotated examples, where imagery is linked to specific kinds of inferences, provide materials to test this idea.

TEXT: Top with another layer of ravioli and the remaining sauce not all the ravioli may be needed. Sprinkle with the Parmesan.

Figure 2: The image depicts both the action and the result of the action. The recipe is from Autodesk Inc. www.instructables.com and was contributed by www.RealSimple.com.

The Complex Coherence of Imagery. Our annotation reveals cases where a single image does include more information than could be packaged into a single textual discourse unit (the proverbial thousand words). In particular, such imagery participates in more complex coherence relationships than we find between text segments. Multiple temporal relationships show this most clearly: 12% of images that have any temporal relation have more than one. For example, many images depict the action that is described in the text, while also showing preparations that have already been made by displaying the scene in which the action is performed. Figure 2 depicts the action and the result of the action. It also shows how to prepare before carrying out the action. Other images show an action in progress but nearing completion and thereby depict the result. For instance, the image that accompanies "mix well until blended" can show both late-stage mixing and the blended result. Looking at a few such cases closely, the circumstances and composition of the photos seem staged to invite such overlapping inferences.

Such cases testify to the richness of multimodal discourse, and help to justify our research methodology. The True/False questions characterize the relevant features of interpretation without necessarily mapping to single discourse relations. For instance, Q4 and Q5 indicate inferences in line with an **Elaboration** relation; Q9 and Q10 indicate inferences in line with an **Exemplification** relation, as information presented in images show just one case of a generalization presented in accompanying text. However, our data shows that these inferences can be combined in productive ways, in keeping with the potentially complex relevant content of images.

Information across modalities. We carried out machine learning experiments to assess what information images provide and what textual cues can guide image interpretation. We use SVM classifiers for performance, and Multinomial Naive Bayes classifiers to explain classifier decision making, both with bag-of-words features.

Table 2 reports the F1 measure for instance classification with SVMs (with 5-fold cross validation). In many cases, machine learning is able to find cues that reliably help guess inferential pat-

Q4. Text has quantities not in image			
True		False	
1	-4.1	add	-4.5
cup	-4.4	place	-4.9
minutes	-4.7	put	-5.0
2	-4.7	make	-5.1
1/2	-4.9	mix	-5.1

Q8. Image depicts action in progress			
True		False	
add	-5.0	1	-4.6
mix	-5.2	cup	-4.7
place	-5.3	minutes	-4.9
bread	-5.5	160	-5.1
make	-5.6	put	-5.2

Table 3: Top five features of Multimodal Naive Bayes classifier for two classification problems and their corresponding log–probability estimates.

terns. Table 3 looks at two effective Naive Bayes classifiers, for Q4 (text has quantities) and Q8 (image depicts action in progress). It shows the features most correlated with the classification decision and their log probability estimates. For Q4, not surprisingly, numbers and units are positive instances.

More interestingly, verbs of movement and combination are negative instances, perhaps because such steps normally involve material that has already been measured. For Q8, a range of physical action verbs are associated with actions in progress; negative features correlate with steps involved in actions that don't require ongoing attention (e.g., baking). Table 4 reports top SVM with NB (NBSVM) (Wang and Manning, 2012) features for Q1 that asks subjects to highlight the part of the text that is most related to the image. Action verbs are part of highlighted text, whereas adverbs and quantitative information that cannot be easily depicted in images are part of the remaining segments of the text. Such correlations set a direction for designing or learning strategies to select when to include imagery.

5 Conclusions

In this paper, we have presented the first dataset describing discourse relations across text and imagery. This data affords theoretical insights into the connection between images and instructional text, and can be used to train classifiers to support automated discourse analysis. Another important contribution of this study is that it presents a discourse annotation scheme for cross-modal data, and establishes that annotations for this scheme can be procured from non-expert contributors via crowd-sourcing.

Q1. Information in text		
1	do it clearly	on which
2	let cool for	favorite toppings
3	recipe with directions	after an
4	how slowly the	lightly season
5	7 minutes on	the 2

Q1. Information in images		
1	added a beautiful	cover with
2	put as much	scrapping the
3	skin off of	finally fold
4	cut side toward	after an
5	blend and blend	add a

Table 4: Top five bigram and trigram features of NB-SVM for the first question. The highlighted text that is most relevant to the image describes depicted actions, while the complement descriptions describe quantities or modifications of the actions that are described in the highlighted segments.

Our paper sets the agenda for a range of future research. One obvious example is to extend the approach to other genres of communication with other coherence relations, such as the distinctive coherence of images and caption text (Alikhani and Stone, 2019). Another is to link coherence relations to the structure of multimodal discourse. For example, our methods have not yet addressed whether image–text relations have the same kinds of subordinating or coordinating roles that comparable relations have in structuring text discourse (Asher and Lascarides, 2003). Ultimately, of course, we hope to leverage such corpora to build and apply better models of multimodal communication.

Acknowledgments

The research presented here is supported by NSF Award IIS-1526723 and through a fellowship from the Rutgers Discovery Informatics Institute. Thanks to Gabriel Greenberg, Hristiyan Kourtev and the anonymous reviewers for helpful comments. We would also like to thank the Mechanical Turk annotators for their contributions.

References

Dorit Abusch. 2013. Applying discourse semantics and pragmatics to co-reference in picture sequences. In *Proceedings of Sinn und Bedeutung 17*, pages 9–25, Paris.

M. Alikhani and M. Stone. 2018a. Exploring coherence in visual explanations. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 272–277.

Malihe Alikhani and Matthew Stone. 2018b. Arrows are the verbs of diagrams. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3552–3563.

Malihe Alikhani and Matthew Stone. 2019. "caption" as a coherence relation: Evidence and implications. In *Second Workshop on Shortcomings in Vision and Language (SiVL)*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Mousumi Banerjee, Michelle Capozzoli, Laura McSweeney, and Debajyoti Sinha. 1999. Beyond kappa: A review of interrater agreement measures. *Canadian journal of statistics*, 27(1):3–23.

Anne Cocos, Aaron Masino, Ting Qian, Ellie Pavlick, and Chris Callison-Burch. 2015. Effectively crowdsourcing radiology report annotations. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 109–114.

Neil Cohn. 2013. Visual narrative structure. *Cognitive science*, 37(3):413–452.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. In *NIPS Workshop on Deep Reinforcement Learning*. ArXiv:1511.08099.

Samuel Cumming, Gabriel Greenberg, and Rory Kelly. 2017. Conventions of viewpoint coherence in film. *Philosophers' Imprint*, 17(1):1–29.

Steven K Feiner and Kathleen R McKeown. 1991. Automating the generation of coordinated multimedia explanations. *Computer*, 24(10):33–41.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Tuomo Hiippala and Serafina Orekhova. 2018. Enhancing the ai2 diagrams dataset using rhetorical structure theory.

J. Hunter, N. Asher, and A. Lascarides. 2015. Integrating non-linguistic events into discourse structure. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*, pages 184–194, London.

Alex Lascarides and Matthew Stone. 2009. Discourse coherence and gesture interpretation. *Gesture*, 9(2):147–180.

Scott McCloud. 1993. *Understanding comics: The invisible art*. William Morrow.

Eleni Miltsakaki, Rashmi Prasad, Aravind K. Joshi, and Bonnie L. Webber. 2004. The Penn Discourse Treebank. In *LREC*. European Language Resources Association.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K. Joshi, and Bonnie L. Webber. 2008. The Penn Discourse TreeBank 2.0. In *LREC*. European Language Resources Association.

Philippe Schlenker and Emmanuel Chemla. 2017. Gestural agreement. *Natural Language & Linguistic Theory*, pages 1–39.

Matthew Stone and Una Stojnic. 2015. Meaning and demonstration. *Review of Philosophy and Psychology*, 6(1):69–97.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Bonnie Webber. 2009. Genre distinctions for discourse in the penn treebank. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 674–682. Association for Computational Linguistics.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *EMNLP*, pages 1358–1368. Association for Computational Linguistics.

Improving Dialogue State Tracking by Discerning the Relevant Context

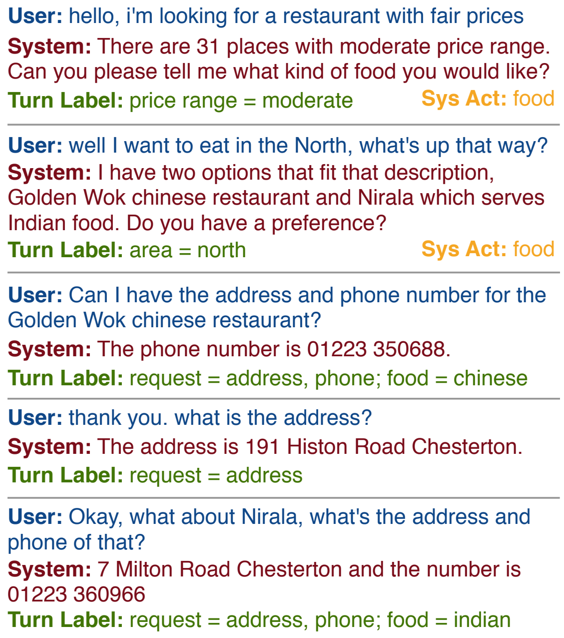
Sanuj Sharma, Prafulla Kumar Choubey, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
`(sanuj, prafulla.choubey, huangrh)@tamu.edu`

Abstract

A typical conversation comprises of multiple turns between participants where they go back-and-forth between different topics. At each user turn, dialogue state tracking (DST) aims to estimate user's goal by processing the current utterance. However, in many turns, users implicitly refer to the previous goal, necessitating the use of relevant dialogue history. Nonetheless, distinguishing relevant history is challenging and a popular method of using dialogue recency for that is inefficient. We, therefore, propose a novel framework for DST that identifies relevant historical context by referring to the past utterances where a particular slot-value changes and uses that together with weighted system utterance to identify the relevant context. Specifically, we use the current user utterance and the most recent system utterance to determine the relevance of a system utterance. Empirical analyses show that our method improves joint goal accuracy by 2.75% and 2.36% on WoZ 2.0 and MultiWoZ 2.0 restaurant domain datasets respectively over the previous state-of-the-art GLAD model.

1 Introduction

Dialog state tracking (DST) is a vital component in the task-oriented dialog systems which is used to estimate user's goals and requests in order to plan next action and respond accordingly. At each turn, DST aims to identify the set of goals that a user aims to achieve and requests that are represented as slot-value pairs. Typically, this decision is made by considering user utterance in the current turn or system actions in the previous turn. However, in many cases, the considered user utterance or system actions do not present enough information and refers to a previous utterance.



User: hello, i'm looking for a restaurant with fair prices
System: There are 31 places with moderate price range. Can you please tell me what kind of food you would like?
Turn Label: price range = moderate **Sys Act:** food

User: well I want to eat in the North, what's up that way?
System: I have two options that fit that description, Golden Wok chinese restaurant and Nirala which serves Indian food. Do you have a preference?
Turn Label: area = north **Sys Act:** food

User: Can I have the address and phone number for the Golden Wok chinese restaurant?
System: The phone number is 01223 350688.
Turn Label: request = address, phone; food = chinese

User: thank you. what is the address?
System: The address is 191 Histon Road Chesterton.
Turn Label: request = address

User: Okay, what about Nirala, what's the address and phone of that?
System: 7 Milton Road Chesterton and the number is 01223 360966
Turn Label: request = address, phone; food = indian

Figure 1: An example dialog from WoZ 2.0 dataset. A turn contains user utterance (blue), system utterance (red), system actions (yellow) and turn label (green). Each turn is separated by a line.

As shown through an example in Figure 1, while exploring different available options, user can go back-and-forth between the currently and previously discussed facts. For instance, when offered with two different restaurant options namely *Nirala* `(food=indian)` and *Golden Wok* `(food=chinese)` in the second turn, user first inquires about the details of *Golden Wok*. And after getting relevant details about the *Golden Wok* in the following two turns, user refers back to the second option provided in second turn and asks about *Nirala* restaurant. To predict the correct slot-value pair `food=indian` in the dialog state of the fifth turn, the system is required to refer back to the second turn again to find information about *Nirala*, as the context obtained from the current dialog turn is insufficient.

Identifying such implicitly referenced historical turns is challenging since implicit references are

Proceedings of NAACL-HLT 2019, pages 576–581
Minneapolis, Minnesota, June 2 - June 7, 2019.

not local and most recent turns are often not informative. Therefore, the traditional approach of modeling dialogue recency (El Asri et al., 2017) may not suffice. Instead, we propose to model implicit references by storing links to the past turn where each of the slots was modified. Then at each turn, we look up though the stored links to find the previous turn which may provide additional cues for predicting the appropriate slot-value.

Moreover, the dialogue system often asks polar questions with yes-no answers. For instance, the DST system should update the dialogue state with `food=indian` when a user replies *Yes* to a system utterance *Do you want Indian food?*. In such cases, neither the user utterance nor system acts (*food* in this example) contain any information about the actual slot-value. This makes utilization of both system and user utterance eminent for dialog state tracking. However, utilizing the previous system utterance together with the current user utterance always at each turn may add noise. Therefore, we use a gating mechanism based on both utterances to determine the relevance of the previous system utterance in the current turn.

The evaluation shows that identifying the relevant context is essential for dialogue state tracking. Our novel model that discerns important details in non-adjacent dialogue turns and the previous system utterance from a dialog history is able to improve the previous state-of-the-art GLAD (Zhong et al., 2018) model on all evaluation metrics for both WoZ and MultiWoZ (restaurant) datasets. Furthermore, we empirically show that a simple self-attention based biLSTM model, using only one-third of the number of parameters as GLAD, outperforms GLAD by identifying and incorporating the relevant context.

2 Related Work

Early work for DST relied on separate Spoken Language Understanding (SLU) module (Henderson et al., 2012) to extract relevant information from user utterances in a pipelined approach. Such systems are prone to error accumulation from a separate SLU module, in absence of necessary dialog context required to interpret the user utterance. Thus, later work on DST moved away from separate SLU modules and inferred the dialog state directly from user utterance and dialog history (Henderson et al., 2014b,c; Zilka and Jurcicek, 2015). These models depend on delexicalization, using generic tags to replace specific slot types and values, and handcrafted semantic dictionaries. In practice, it is difficult to scale these models for every slot type and recent state-of-the-art models for DST use deep learning based methods to learn general representations for user and system utterances and previous system actions, and predict the turn state (Henderson et al., 2013, 2014b; Mrkšić et al., 2015, 2017; Hori et al., 2016; Liu and Lane, 2017; Dernoncourt et al., 2017; Chen et al., 2016). However, these systems are found to perform poorly on rare and unknown slot-value pairs which was recently addressed through local slot-specific encoders (Zhong et al., 2018) and pointer network (Xu and Hu, 2018).

A crucial limitation to all these approaches lies in the modeling of appropriate historical context, which is simply ignored in most of the works. Since user's goal may change back-and-forth between previous values, incorporating relevant historical context is useful in monitoring implicit goal references. In a recent work, El Asri et al. (2017) discussed on similar limitations of current DST task and introduced a new task of frame tracking that explicitly tracks every slot-values that were introduced during the dialogue. However, that significantly complicates the task by maintaining multiple redundant frames that are often left unreferenced. Our proposed model, that explicitly track relevant historical user and system utterances, can be easily incorporated into any known DST or frame tracking systems such as Schulz et al. (2017) to replace the recency encoding.

3 Discerning Relevant Context for DST

Similar to previous works, we decompose the multi-label classification problem to binary classification where we score each slot-value pair and select the ones that receive a score above a threshold to be included in the current dialog state. To predict the score for a candidate slot-value pair, the model uses the relevant past user utterance (referential utterance), a fused utterance composed using the current user utterance and the system utterance of the previous turn, as well as previous system actions as evidence. Shown in Figure 2, our model comprises of:

Lookup module: retrieves a link to the turn where each of the slots changes. At each step, our system refers to the lookup module that returns the past user utterance (the "*antecedent user utterance*")

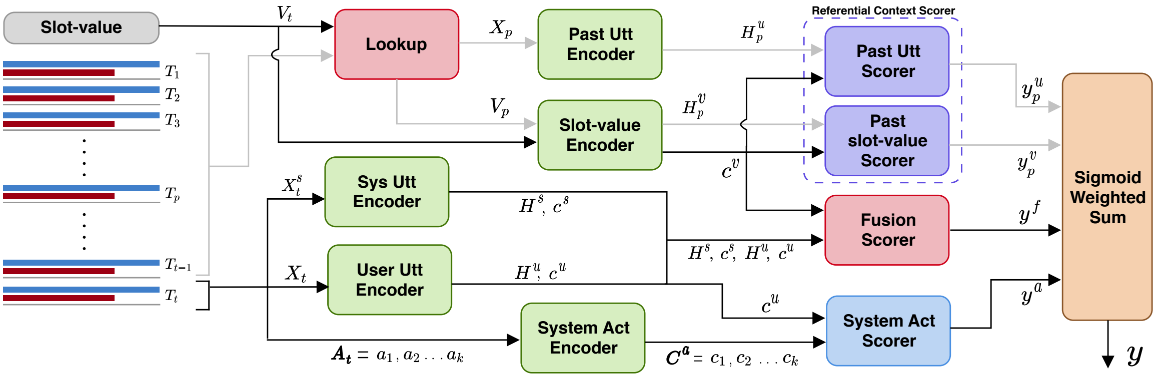


Figure 2: The Architecture of Context Aware Dialogue State Tracker.

where the candidate slot-type was modified as well as outputs the previous slot-value.

GLE modules: Each of the five green modules in Figure 2 is a global-locally self-attentive encoder (GLE module) (Zhong et al., 2018) that encodes each type of evidence into a vector representation (c). Each input is represented as a sequence of words which is encoded to a vector representation via global-local self-attentive encoder (GLE) module (Zhong et al., 2018). Specifically, GLE employs local slot-specific bidirectional LSTMs and a global bidirectional LSTM (Hochreiter and Schmidhuber, 1997) that is shared across all slots for encoding the input sequence into a sequence of hidden states (H), followed by a self-attention layer (Lin et al., 2016) to obtain a fixed dimension vector representation (c).

The GLE modules are used to encode the antecedent user utterance (H_p^u, c_p^u), the current user utterance (H^u, c^u), the previous system utterance (H^s, c^s), each of the system acts (H^{a_i}, c^{a_i}), as well as the previous slot-value (H_p^v, c_p^v) and the candidate (H^v, c^v) slot-value.

Referential Context Scorer: uses the candidate slot value (c^v), the antecedent user utterance as well as the previous slot-value to determine if the candidate slot value was referenced in the antecedent utterance. Specifically, the scorer uses the representation of the candidate slot value c^v to attend over hidden states of the antecedent user utterance and the previous slot-value, H_p^u and H_p^v, and then computes attention weights for each of the hidden states. Next, the scorer sums up the hidden states weighed with the calculated attentions to get the summary context (Equation 1). Finally, the scorer applies a linear neural layer to calculate the scores y_p^v and y_p^u representing the likelihoods that the candidate slot-value is different from the previous slot-value and the candidate slot-value was unreferenced in the antecedent utterance (Equation 2).

$$Q(H,c): \ a_j = (H_j)^\top c \, ; \ p = softmax(a) \qquad Q(H,c) = \sum_i p_i H_i \tag{1}$$

$$y_p^u = W_p^u \, Q(H_p^u, c^v) + b_p^u \qquad y_p^v = W_p^v \, Q(H_p^v, c^v) + b_p^v \tag{2}$$

Fusion Scorer: leverages necessary details in the previous system utterance to enrich the current user utterance. First, we use a gating mechanism based on c^s and c^u that determines the relevance of the previous system utterance in the current turn. We concatenate c^s and c^u and use a linear layer with sigmoid activation to calculate the score α (Equation 3). Then, we use attention from c^v over H^s and H^u to calculate context summaries (l^s, l^u), and combine the summary vectors by taking their normalized weighted sum based on α. We finally apply a single linear layer to calculate the score y^f that determines the likelihood of the candidate slot-value based on both the current user utterance and the previous system utterance (Equation 4).

$$f_c = W_{fc}(c^s \oplus c^u) + b_{fc} \qquad \alpha = \sigma(W_\alpha tanh(f_c) + b_\alpha) \tag{3}$$

$$l^s = Q(H^s, c^v) \, ; \ l^u = Q(H^u, c^v) \qquad l^f = \alpha l^s + (1-\alpha) l^u \, ; \ y^f = W_{lf} l^f + b_{lf} \tag{4}$$

System Act Scorer: is the same as the action scorer proposed by (Zhong et al., 2018). Specifically, The scorer uses attention from c^u over C^a to calculate action summary followed by a linear layer with sigmoid activation to calculate the score y^a that determines the relevance of the candidate slot-value based on the previous system actions (Equation 5).

$$l^a = Q(C^a, c^u)\ ; \quad y^a = (l^a)^\top c^v \qquad (5)$$

It then calculates the final score of the candidate slot-value by taking weighted sum of the four scores (y_p^u, y_p^v, y^f, y^a) followed by a sigmoid layer, where weights are learned in the network.

4 Evaluations

4.1 Experimental Setup

We primarily use WoZ 2.0 (Wen et al., 2017) restaurant reservation task dataset that consists of 1200 dialogues for training and evaluation. Each dialogue has an average of eight turns, where each turn contains *system utterance transcript*, *user utterance transcript*, *turn label* and *belief state*. All the dialogue states and actions are based on a task ontology that supports three different informable slot-types namely *price range* with 4 values, *food* with 72 values, *area* with 7 values, and *requests* of 7 different types like *address* and *phone*. Following the standard settings, we use 600 dialogues for training, 200 for validation and the remaining 400 for testing.

We also use dialogues from restaurant domain in MultiWoZ 2.0 dataset (Budzianowski et al., 2018) for secondary evaluation. It banks on a significantly complex ontology covering seven informable slot types with 276 different values (*food, price range, restaurant name, area, book time, book day* and *book people* with 97, 6, 105, 8, 43, 8 and 9 values respectively). We use standard training, validation and test splits of 1199, 50 and 61 dialogues respectively.

All the models on WoZ 2.0 are evaluated on the two standard metrics introduced in Henderson et al. (2014a). First, **Joint Goal Accuracy** is the percentage of turns in a dialogue where the user's informed joint goals are identified correctly. Joint goals are accumulated turn goals up to the current dialog turn. Second, **Turn Request Accuracy** calculates the percentage of turns in a dialogue where the user's requests were correctly identified. Models on MultiWoZ 2.0 dataset are evaluated using joint goal and turn inform accuracies, as used by Nouri and Hosseini-Asl (2018).

4.2 Implementation Details

We use pretrained GloVe word embeddings (Pennington et al., 2014) concatenated with character n-gram embeddings (Hashimoto et al., 2017) which are kept fixed during the training. Each of bi-LSTMs use 200 hidden dimensions. All the models are trained using ADAM optimizer (Kingma and Ba, 2014) with the initial learning rate of 0.001. Dropout rate (Srivastava et al., 2014) is set to 0.2 for all biLSTM modules and the embedding layer. The models are trained for a maximum of 100 epochs with a batch size of 50. The validation data was used for early stopping and hyperparameter tuning.

Model	WoZ 2.0 Joint Goal	 Turn Request
Delexalisation-Based Model + SD	83.7%	87.6%
NBT - DNN	84.4%	91.2%
NBT - CNN	84.2%	91.6%
GLAD †	**86.4%**	**97.1%**
Global biLSTM based GLE	85.0%	96.8%
Global biLSTM based GLE + RC	87.4%	**97.0%**
Global biLSTM based GLE + RC + FS	**88.4%**	**97.0%**
GLAD + RC + FS	**89.2%**	**97.4%**

Table 1: Test accuracy of baselines and proposed models on WoZ 2.0 restaurant reservation dataset. †Retrained using docker container provided by the authors with exactly same hyper-parameters. We also experimented with different versions of PyTorch and cuDNN and found that results had high variance. Therefore, we report the average performance over 5 runs with different initializations for GLAD and all our models.

Model	MultiWoZ 2.0 (Restaurant) Joint Goal	 Turn Inform
GLAD	43.95%	76.99%
GLAD + RC	**45.72%**	**77.87%**
GLAD + RC + FS	**46.31%**	**78.76%**

Table 2: Test accuracy of GLAD and proposed models on MultiWoZ 2.0 restaurant domain dataset. Note that we considered all 276 slot-values for evaluating models. Budzianowski et al. (2018) reported joint goal accuracy of 80.9 on MultiWoZ 2.0 (restaurant) dataset. We believe they didn't include *restaurant name* slot in their evaluation and only considered presence of three slot-types—*book time, book day* and *book people*—and not their values.

4.3 Results

Table 1 compares the performance of our proposed models with different baselines, including **delexalisation-based model + SD** (Wen et al., 2017), DNN and CNN variants of **neural belief tracker** (Mrkšić et al., 2017) and the previous state-of-the-art GLAD systems (Zhong et al., 2018) on WoZ 2.0 dataset. We also implement a simplified variant of GLAD, **Global BiLSTM**

Model	Approx. # of parameters
Global biLSTM based GLE	1.2 million
Global biLSTM based GLE + RC + FS	**6 million**
GLAD	17 million
GLAD + RC + FS	28 million

Table 3: Number of learnable parameters for different models on WoZ 2.0 dataset

based GLE, by removing slot-specific local biLSTMs from the GLE encoder. We then successively combine it with referential context (**Global biLSTM based GLE + RC**) and the fused previous system utterance (**Global biLSTM based GLE + RC + FS**). Finally, we directly incorporate the referential context and gate selected system utterance into the GLAD system (**GLAD + RC + FS**).

Irrespective of the underlying system, utilizing appropriate context from the previous turns improves the overall performance of a dialogue state tracker on both joint goal and turn request accuracies on WoZ 2.0 dataset. First, incorporating relevant referential utterances to identify implicitly mentioned slot-value improves the accuracy of global biLSTM based GLE model on joint goal task by 2.4%. Then, gating based mechanism to augment user utterance with relevant information from the previous system utterance further improves the joint goal accuracy by 1.0%. Together, they improve joint goal and request accuracy of the global biLSTM based GLE model by 3.4% and 0.2% respectively. Furthermore, as evident from the results in Table 2, both referential context and fused system utterance proportionally improve performance on MultiWoZ 2.0 dataset as well with overall improvement of 2.36% and 1.77% on joint goal and turn inform accuracies respectively. Performances of all models on MultiWoZ 2.0 are significantly inferior compared to WoZ 2.0 owing to higher complexity, with richer and longer utterances and considerably more slot-values in the former dataset.

5 Analysis

The utilization of relevant context results in significant reduction in the number of learnable parameters in the model as shown in Table 3. Relevant context with the baseline model is able to outperform GLAD while using only one third of the number of learnable parameters. The parameters added due to using relevant context are the parameters for encoding the antecedent referential user utterance and the previous system utterance as well as the past utterance and past slot-value scorers. However, we also observe high variance in the joint goal accuracy. Since joint goal is calculated by accumulating turn goals, an error in predicting a turn goal is propagated to all the downstream turns.

6 Conclusion

We have presented a novel method for identifying the relevant historical user utterance as well as determining the relevance of the system utterance from the last turn to enrich the current user utterance and improve goal tracking in dialogue systems. The experimental results show that discerning relevant context from the dialog history is crucial for tracking dialog states.

Acknowledgments

We want to thank our anonymous reviewers for providing insightful review comments.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249.

Franck Dernoncourt, Ji Young Lee, Trung H Bui, and Hung H Bui. 2017. Robust dialog state tracking for large ontologies. In *Dialogues with Social Robots*, pages 475–485. Springer.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219. Association for Computational Linguistics.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933. Association for Computational Linguistics.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 176–181. IEEE.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 324–329. IEEE.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Takaaki Hori, Hai Wang, Chiori Hori, Shinji Watanabe, Bret Harsham, Jonathan Le Roux, John R Hershey, Yusuke Koji, Yi Jing, Zhaocheng Zhu, et al. 2016. Dialog state tracking with attention-based sequence-to-sequence learning. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 552–558. IEEE.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2016. A structured self-attentive sentence embedding.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *arXiv preprint arXiv:1708.05956*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 794–799. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788. Association for Computational Linguistics.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Hannes Schulz, Jeremie Zumer, Layla El Asri, and Shikhar Sharma. 2017. A frame tracking model for memory-enhanced dialogue systems. *arXiv preprint arXiv:1706.01690*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467. Association for Computational Linguistics.

Lukas Zilka and Filip Jurcicek. 2015. Incremental lstm-based dialog state tracker. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 757–762. IEEE.

CLEVR-Dialog:
A Diagnostic Dataset for Multi-Round Reasoning in Visual Dialog

Satwik Kottur[1], **José M.F. Moura**[1], **Devi Parikh**[2,3], **Dhruv Batra**[2,3], **Marcus Rohrbach**[2]
[1]Carnegie Mellon University, [2]Facebook AI Research, [3]Georgia Institute of Technology
{skottur,moura}@andrew.cmu.edu,{parikh,dbatra}@gatech.edu,mrf@fb.com

Abstract

Visual Dialog is a multimodal task of answering a sequence of questions grounded in an image (using the conversation history as context). It entails challenges in vision, language, reasoning, and grounding. However, studying these subtasks in isolation on large, real datasets is infeasible as it requires prohibitively-expensive complete annotation of the 'state' of all images and dialogs.

We develop CLEVR-Dialog, a large diagnostic dataset for studying multi-round reasoning in visual dialog. Specifically, we construct a *dialog grammar* that is grounded in the scene graphs of the images from the CLEVR dataset. This combination results in a dataset where all aspects of the visual dialog are fully annotated. In total, CLEVR-Dialog contains 5 instances of 10-round dialogs for about $85k$ CLEVR images, totaling to $4.25M$ question-answer pairs.

We use CLEVR-Dialog to benchmark performance of standard visual dialog models; in particular, on *visual coreference resolution* (as a function of the coreference distance). This is the first analysis of its kind for visual dialog models that was not possible without this dataset. We hope the findings from CLEVR-Dialog will help inform the development of future models for visual dialog. Our code and dataset are publicly available[1].

1 Introduction

The focus of this work is on intelligent systems that can *see* (perceive their surroundings through vision), *talk* (hold a visually grounded dialog), and *reason* (store entities in memory as a dialog progresses, refer back to them as appropriate, count, compare, *etc*.). Recent works have begun studying such systems under the umbrella of *Visual Dialog* (Das et al., 2017a; de Vries et al., 2017), where an agent must answer a *sequence* of questions grounded in an image. As seen in Fig. 1, this entails challenges in – vision (*e.g*., identifying objects and their attributes in the image), language/reasoning (*e.g*., keeping track of and referencing previous conversation via memory), and grounding (*e.g*., grounding textual entities in the image).

In order to train and evaluate agents for Visual Dialog, Das et al. (2017a) collected a large dataset of human-human dialog on real images collected between pairs of workers on Amazon Mechanical Turk (AMT). While such large-scale realistic datasets enable new lines of research, it is difficult to study the different challenges (vision, language, reasoning, grounding) in isolation or to break down the performance of systems over different challenges to identify bottlenecks, because that would require prohibitively-expensive complete annotation of the 'state' of all images and dialogs (all entities, coreferences, *etc*.).

In this work, we draw inspiration from Johnson et al. (2017), and develop a large diagnostic dataset—CLEVR-Dialog—for studying and benchmarking multi-round reasoning in visually-grounded dialog. Each CLEVR image is synthetically rendered by a particular scene graph (Johnson et al., 2017) and thus, is by construction exhaustively annotated. We construct a *dialog grammar* that is grounded in these scene graphs. Specifically, similar to Das et al. (2017b), we view dialog generation as communication between an Answerer (A-er) who can 'see' the image and has the complete scene graph (say S_a), and a Questioner (Q-er), who does not 'see' the image and is trying to reconstruct the scene graph over rounds of dialog (say S_q^t). As illustrated in Fig. 1, the dialog begins by A-er providing a grounded caption for the image, which conveys some but not all information about S_a. The Q-er builds a partial scene graph S_q^0 based on the caption, and follows up by asking questions

---

[1] https://github.com/satwikkottur/clevr-dialog

Proceedings of NAACL-HLT 2019, pages 582–595
Minneapolis, Minnesota, June 2 - June 7, 2019.

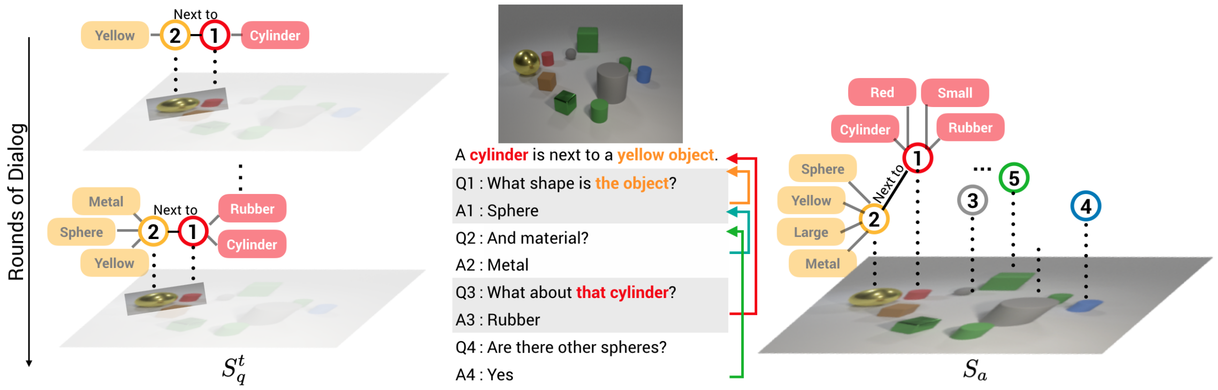


Figure 1: CLEVR-Dialog: we view dialog generation as communication between an Answerer (A-er) who can 'see' the image I and has the complete scene graph S_a (far right), and a Questioner (Q-er), who does not 'see' the image. A-er begins the dialog with a grounded caption (*'A cylinder is next to a yellow object'*). The Q-er converts this caption into a partial scene graph S_q^0 (far left, top), follows up with a question grounded in S_q^0 (*'What shape is the object?'*), which the A-er answers, and the dialog progresses. Questions at round t are generated based solely on S_q^t, *i.e.*, without looking at I or S_a, which mimics real-life scenarios of visual dialog. Note that while studying visual dialog on CLEVR-Dialog, models are forced to answer questions with just the image and dialog history as additional inputs, and do not have access to S_a.

grounded in S_q^0, which the A-er answers, and the dialog progresses. Our dialog grammar defines rules and templates for constructing this grounded dialog. Note that A-er with access to S_a (perfect vision) exists **only** during dialog generation to obtain ground truth answers. While studying visual dialog on CLEVR-Dialog, models are forced to answer questions with just the image and dialog history as additional inputs.

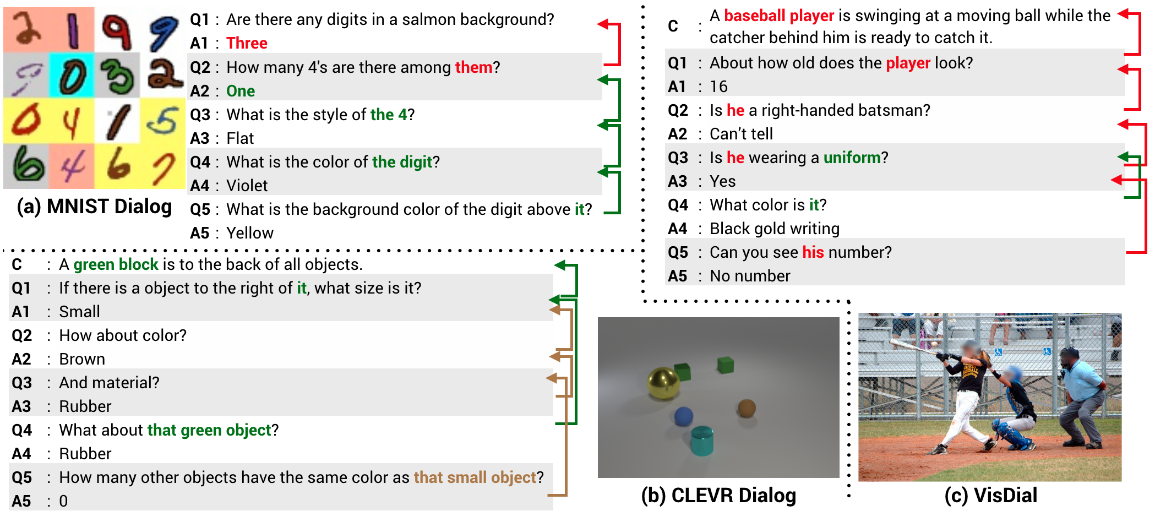
In total, CLEVR-Dialog contains 5 instances of 10-round dialogs for each of 70k (train) and 15k (val) CLEVR images, totaling to 3.5M (train) and 0.75M (val) question-answer pairs. We benchmark several visual dialog models on CLEVR-Dialog as strong baselines for future work.

The combination of CLEVR images (with full scene graph annotations) and our dialog grammar results in a dataset where all aspects of the visual dialog are fully annotated. We use this to study one particularly difficult challenge in multi-dialog visual reasoning – of *visual coreference resolution*. A coreference arises when two or more phrases (*coreferring phrases*) in the conversation refer to the same entity (*referent*) in the image. For instance, in the question *'What about that cylinder?'* (Q3) from Fig. 1, the referent for the phrase *'that cylinder'* can be inferred only after resolving the phrase correctly based on the dialog history, as there are multiple cylinders in the image. We use CLEVR-Dialog to diagnose performance of different methods as a function of the history dependency (*e.g.*, coreference distance—the number of rounds between successive mentions of the same object) and find that the performance of a state-of-art model (CorefNMN) is at least 30 points inferior for questions involving coreference resolution compared to those which do not (Fig. 5), highlighting the challenging nature of our dataset. This is the first analysis of its kind for visual dialog that was simply not possible without this dataset. We hope the findings from CLEVR-Dialog will help inform the development of future models for visual dialog.

2 Related Work

Coreference Resolution is a well studied problem in the NLP community (Ng, 2010; Wiseman et al., 2016; Lee et al., 2017; Clark and Manning, 2016a,b). Our work focuses on *visual* coreference resolution – the referent is now a visual entity to be grounded in visual data. Several works have tackled visual coreference resolution in videos (Ramanathan et al., 2014; Rohrbach et al., 2017) and 3D data (Kong et al., 2014), and have introduced real image datasets for the same (Hodosh et al., 2014).

Visual Dialog and Synthetic Datasets. We contrast CLEVR-Dialog against four existing datasets: (1) **CLEVR** (Johnson et al., 2017) is a diagnostic dataset for visual question answering (VQA) (Antol et al., 2015) on rendered images that contain objects like cylinders, cubes, *etc.*, against a plain background (Fig. 1). While CLEVR-Dialog uses the same set of images, the key difference is that of focus and emphasis – the objective of CLEVR-



Q1 : Are there any digits in a salmon background?
A1 : Three
Q2 : How many 4's are there among them?
A2 : One
Q3 : What is the style of the 4?
A3 : Flat
Q4 : What is the color of the digit?
A4 : Violet
Q5 : What is the background color of the digit above it?
A5 : Yellow

(a) MNIST Dialog

C : A baseball player is swinging at a moving ball while the catcher behind him is ready to catch it.
Q1 : About how old does the player look?
A1 : 16
Q2 : Is he a right-handed batsman?
A2 : Can't tell
Q3 : Is he wearing a uniform?
A3 : Yes
Q4 : What color is it?
A4 : Black gold writing
Q5 : Can you see his number?
A5 : No number

C : A green block is to the back of all objects.
Q1 : If there is a object to the right of it, what size is it?
A1 : Small
Q2 : How about color?
A2 : Brown
Q3 : And material?
A3 : Rubber
Q4 : What about that green object?
A4 : Rubber
Q5 : How many other objects have the same color as that small object?
A5 : 0

(b) CLEVR Dialog

(c) VisDial

Figure 2: Example dialogs from MNIST Dialog, CLEVR-Dialog, and VisDial, with coreference chains manually marked for VisDial and automatically extracted for MNIST Dialog and CLEVR-Dialog.

VQA questions is to stress-test spatial reasoning in independent single-shot question answering; the objective of CLEVR-Dialog is to stress-test temporal or multi-round reasoning over the dialog history. (2) **CLEVR-Ref+** (Liu et al., 2019) is a diagnostic dataset based on CLEVR images for visual reasoning in referring expressions. CLEVR-Dialog goes beyond CLEVR-Ref+, which focuses on grounding objects given a natural language expression, and deals with additional visual and linguistic challenges that require multi-round reasoning in visual dialog. (3) **MNIST-Dialog** (Seo et al., 2017) is a synthetic dialog dataset on a grid of 4×4 stylized MNIST digits (Fig. 2). While MNIST-Dialog is similar in spirit to CLEVR-Dialog, key difference is complexity – the distance between a coreferring phrase and its antecedent is always 1 in MNIST-Dialog; in contrast, CLEVR-Dialog has a distribution ranging from 1 to 10. (4) **VisDial** (Das et al., 2017a) is a large scale visual dialog dataset collected by pairing two human annotators (a Q-er and an A-er) on AMT, built on COCO (Lin et al., 2014) images. VisDial being a large open-ended real dataset encompasses all the challenges of visual dialog, making it difficult to study and benchmark progress on individual challenges in isolation. Fig. 2 qualitatively compares MNIST-Dialog, CLEVR-Dialog, and VisDial, and shows coreference chains (manually annotated for this VisDial example, and automatically computed for MNIST-Dialog and CLEVR-Dialog). We can see that the chains in MNIST-Dialog are the simplest (distance always 1). While coreferences in VisDial can be on a similar level of difficulty than CLEVR-Dialog, the difficult cases are rarer in VisDial.

Name	CLEVR Dialog (ours)	MNIST Dialog	VisDial
# Images	85k	50k	123k
# Dialogs	425k	150k	123k
# Questions	4.25M	1.5M	1.2M
# Unique Q	73k	355	380k
# Unique A	29	38	340k
Vocab. Size	125	54	7.k
Mean Q Len.	10.6	8.9	5.1
Mean Coref Dist.	3.2	1.0	-

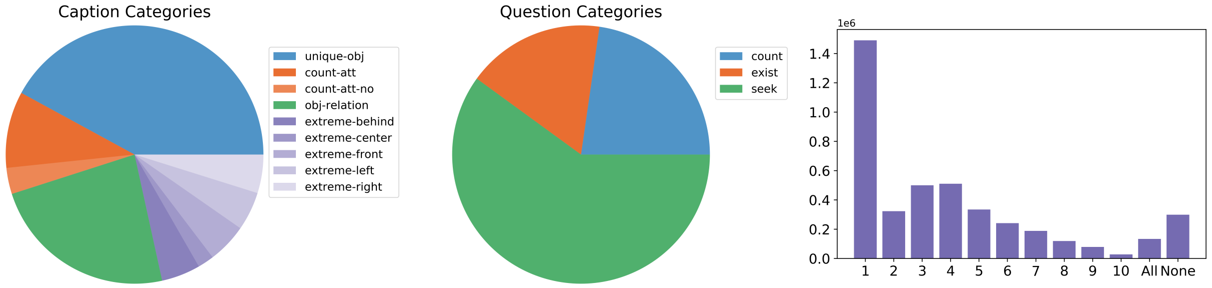
Table 1: Dataset statistics comparing CLEVR-Dialog to MNIST Dialog (Seo et al., 2017). Our dataset has 3× the questions (larger), 206× the unique number of questions (more diverse), 3.2× the mean coreference distance (more complex), and longer question lengths. Similar stats for VisDial are also shown. Coreference distance can not be computed for VisDial due to lack of annotations.

3 CLEVR-Dialog Dataset

In this section, we describe the existing annotation for CLEVR images, then detail the generation process for CLEVR-Dialog, and present the dataset statistics in comparison to existing datasets.

Setup. Every CLEVR image I has a full scene graph annotation, S_a. This contains information about all the objects in the scene, including four major attributes $\{$*color, shape, material, size*$\}$, 2D image and 3D world positions, and relationships $\{$*front, back, right, left*$\}$ between these objects. We only use objects, attributes, and relationships.

Dialog Grammar. An important characteristic of visual dialog that makes it suitable for practical applications is that the questioner does not 'see'



(a) Distribution of caption (left) and question (right) categories.

(b) Distribution of coreference distances.

Figure 3: Distribution of caption and question categories, and history dependency in CLEVR-Dialog dataset.

the image (because if it did, it would not *need* to ask questions). To mimic this setup, we condition our question generation at round t only on the partial scene graph S_q^t that accumulates information received so far from the dialog history (and not on S_a). Specifically, we use a set of caption $\{T_i^C\}$ and question $\{T_i^Q\}$ templates, which serve as the structural units of our dialog grammar. The role of the caption is to *seed* the dialog and initialize S_q^0. Each of the question templates is accompanied by a set of constraints on S_q^t, which decide if a particular template can be selected at the current round. For instance, a question *'What shape is the blue object?'* can be only be asked (generated) if the dialog so far has already *mentioned* a 'blue object', *i.e.*, only if S_q^t contains a (unique) 'blue object'. The nature and difficulty of the dataset is highly dependent on these templates, thus making their selection crucial.

To this end, we carefully design four categories of caption templates: (a) `Obj-unique` mentions an object with unique set of attributes in the image, (b) `Obj-count` specifies the presence of a group of objects with common attributes, (c) `Obj-extreme` describes an object at one of the positional extremes of the image (right, left, fore, rear, center), (d) `Obj-relation` talks about the relationship between two objects along with their attributes in a way that allows them to be uniquely identified in the complete scene graph S_a.

For the questions, we experiment with three different categories: (a) **Count** questions ask for a count of objects in the image satisfying specific conditions, *e.g.*, *'How many objects share the same color as this one?'*, (b) **Existence** questions are yes/no binary questions that verify conditions in the image, *e.g.*, *'Are there any other cubes?'*, and (c) **Seek** questions query attributes of objects, *e.g.*, *'What color is that cylinder?'*. Note that CLEVR-Dialog represents not just a static dataset but also a recipe for constructing increasingly challenging grounded dialog by expanding this grammar. Refer to the appendix for further details.

Dialog Generation. At a high level, dialog generation now 'simply' involves selecting a sequence of templates such that the accompanying constraints are satisfied by S_q^t at all t. As a tractable approximation to this exponentially-large constraint satisfaction problem, we use beam search that finds a valid solution *and* enforces additional conditions to make the dialog *interesting* (see Fig. 4). At every round of the dialog (after 3 rounds), we ensure that each of the question template types—count, existence, and seek—falls within a range ($10\% - 20\%$ for count/existence each, and $30\% - 60\%$ for seek). In addition, we identify *independent* questions that do not need history to answer them, *e.g.*, *'How many objects are present in the image?'*, and limit their number to under 10%. We found this to be effective both in terms of speed and dialog diversity. Fig. 4 illustrates the diverse set of candidate questions generated at each round for a given image.

Dataset Statistics. We compare CLEVR-Dialog to MNIST-Dialog and VisDial in Tab. 1, but the key measure of coreference distance cannot be reported for VisDial as it is not annotated. Overall, CLEVR-Dialog has 3× the questions and a striking 206× the unique number of questions than MNIST-Dialog , indicating higher linguistic diversity. CLEVR-Dialog questions are longer with a mean length of 10.6 compared to 8.9 for MNIST-Dialog. Crucially, supporting our motivation, the mean distance (in terms of rounds) between the coreferring expressions in CLEVR-Dialog is 3.2× compared to 1.0 in MNIST-Dialog. Moreover, the distances (see Fig. 3b) in CLEVR-Dialog vary (min of 1, max of 10), while it is constant (at 1) in MNIST-Dialog, making it easy for models to pick

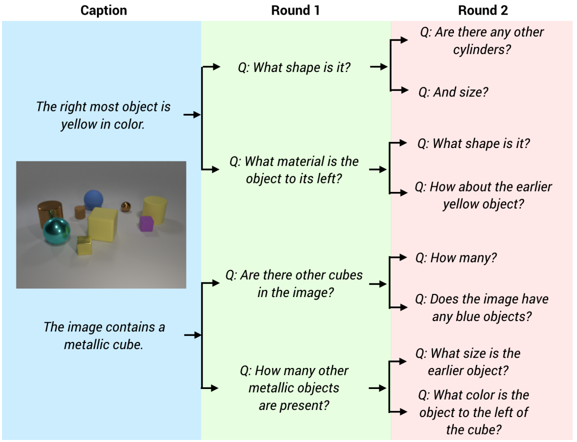


Figure 4: Dialog generation in CLEVR-Dialog. At each round, all valid question templates are used to generate candidates for the next question. However, only a few *interesting* candidates (beams) are retained for further generation, thus avoiding an exploding number of possibilities as rounds of dialog progress.

Model	**Acc.**
Random	3.4
Random-Q	33.4
LF-Q	40.3
LF-QI	50.4
LF-QH	44.1
LF-QIH	55.9
HRE-QH	45.9
HRE-QIH	63.3
MN-QH	44.2
MN-QIH	59.6
NMN	56.6
CorefNMN	**68.0**

Table 2: Accuracy (%) on CLEVR-Dialog (higher is better). See text for details.

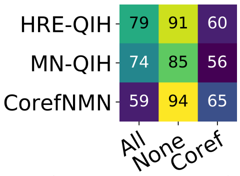


Figure 5: Breakdown of performance by questions that depend on entire history (*All*), require coreference resolution (*Coref*), and are history-independent (*None*).

up on this bias. The distribution of caption and question templates is given in Fig. 3a. See appendix for further analysis.

4 Experiments

Baselines. To benchmark performance, we evaluate several models on CLEVR-Dialog. **Random** picks an answer at random. **Random-Q** picks an answer at random among valid answers for a given question type (*e.g.*, name of a color for color questions). Further, we adapt the discriminative visual dialog models from Das et al. (2017a): (a) **L**ate **F**usion (**LF**) that models separately encode each of question (Q), history (H), and image (I); and then fuse them by concatenation. (b) **H**ierarchical **R**ecurrent **E**ncoder (**HRE**) that models dialog via both dialog-level and sentence-level recurrent neural networks. (c) **M**emory **N**etwork (**MN**) that stores history as memory units and retrieves them based on the current question. We also consider neural modular architectures: (a) **CorefNMN** (Kottur et al., 2018) that explicitly models coreferences in visual dialog by identifying the reference in the question (textual grounding) and then localizing the referent in the image (visual grounding), (b) **NMN** (Hu et al., 2017), which is a history-agnostic ablation of CorefNMN.

Results. We use multi-class classification accuracy for evaluation since CLEVR-Dialog has one-word answers. Tab. 2 shows the performance of different models. The key observations are: (a) Neural models outperform random baselines by a large margin. The best performing model, CorefNMN, outperforms Random-Q by 35%. (b) History-agnostic models (LF-Q, LF-QI, NMN) also suffer in performance, highlighting the importance of history. (c) Finally, we break down the performance of top-3 models on questions which depend on entire history (*All*), require coreference resolution (*Coref*), and are history-independent (*None*), in Fig. 5. We find that CorefNMN is 30% worse on *Coref* than *None* questions, signifying the complexity of CLEVR-Dialog as the former are qualitatively harder to answer than the latter. (d) More interestingly, HRE-QIH, though inferior to CorefNMN on *Coref*, outperforms the latter on *All* questions (*'How many other objects?'*) by around 20%. A possible explanation is that the former, owing to its dialog-level RNN, captures global summaries more efficiently than the latter. This is the first analysis of its kind for visual dialog that was simply not possible without this dataset. Appendix provides a further analysis of model performances.

Conclusion. We proposed a large, synthetic dataset called CLEVR-Dialog, to study multi-round reasoning in visual dialog, and in particular the challenge of visual coreference resolution. We benchmarked several qualitatively different models from prior work on this dataset, which act as baselines for future work. Our dataset opens the door to evaluate how well models do on visual coreference resolution, without the need to collect expensive annotations on real datasets.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653. Association for Computational Linguistics.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *CVPR*.

Abhishek Das, Satwik Kottur, Jos M. F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Peter Hodosh, Alice Young, Micah Lai, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics (TACL)*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Satwik Kottur, Jose M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *The European Conference on Computer Vision (ECCV)*.

Ranjay Krishna, Ines Chami, Michael Bernstein, and Li Fei-Fei. 2018. Referring relationships. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Runtao Liu, Chenxi Liu, Yutong Bai, and Alan Yuille. 2019. Clevr-ref+: Diagnosing visual reasoning with referring expressions. *arXiv preprint arXiv:1901.00850*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1396–1411, Stroudsburg, PA, USA. Association for Computational Linguistics.

V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei. 2014. Linking people with "their" names using coreference resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Anna Rohrbach, Marcus Rohrbach, Siyu Tang, Seong Joon Oh, and Bernt Schiele. 2017. Generating descriptions with grounded and co-referenced people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. 2017. Visual reference resolution using attention memory for visual dialog. In *Advances in Neural Information Processing Systems (NIPS)*.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004. Association for Computational Linguistics.

Appendix

The appendix is organized as follows:

- We begin with the description of CLEVR images in Sec. A,
- Sec. B describes further details of the dialog generation,
- Sec. C provides additional statistical analysis for CLEVR-Dialog,
- Diagnostic model performance analysis in given in Sec. D, and finally
- Implementation details can be found in Sec. E.

A CLEVR Images

First introduced by Johnson et al. (2017), CLEVR images are synthetically rendered, and contain several objects spatial located on a plain background. These objects have four different attributes: (a) *Shape*—cylinder, cube, sphere; (b) *Color*—blue, brown, cyan, gray, green, purple, red, yellow; (c) *Size*—large and small; and finally (d) *Material*—metal and rubber.

B Generating CLEVR-Dialog Dataset

As noted in the main paper, an important characteristic of visual dialog that makes it suitable for practical applications is that the questioner does not 'see' the image (because if it did, it would not *need* to ask questions). To mimic this setup, we condition our question generation at round t only on the partial scene graph S_q^t that accumulates information received so far from the dialog history (and not on S_a). Specifically, we use a set of caption $\{T_i^C\}$ and question $\{T_i^Q\}$ templates (enumerated in Tab. 3), which serve as the basis for our dialog generation. Each of these templates in turn consists of primitives, composed together according to a generation grammar. In what follows, we will first describe these primitives, discuss how they are used to generate a caption or a question at each round, and tie everything together to explain dialog generation in CLEVR-Dialog.

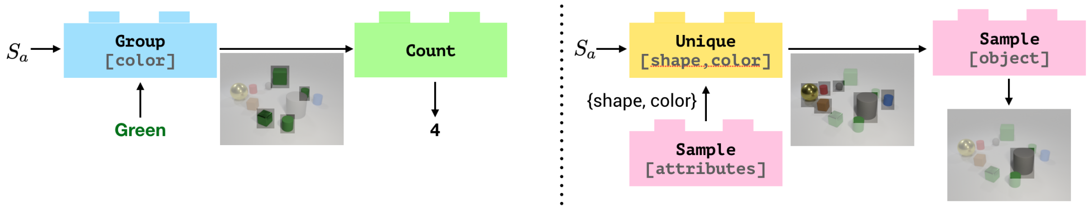
Grammar Primitives. The templates used to generate captions and questions are composed of intuitive and atomic operations called primitives. Each of these primitives can have different instantiations depending on a parameter, and also take input arguments. For example, all `Filter` primitives filter out objects from an input set of objects according to certain constraints. In particular, `Filter[color](blue)` filters out blue objects from a given set of objects, while `Filter[shape](sphere)` filters out all spheres. In our work, we use the following primitives:

- `Sample`: sample an object/attribute,
- `Unique`: identify unique objects/attributes,
- `Count`: count the number of input objects,
- `Group`: group objects based on attribute(s),
- `Filter`: filter inputs according to a constraint,
- `Exist`: check for existence of objects,
- `Relate`: apply a relation (*e.g.*, *right of*).

Note that each of these primitives inherently denotes a set of constraints, which when failed leads to a reset of the generation process for the current caption/question in the dialog. For example, if the output of `Filter[color](blue)` is empty due to an absence of blue objects in the input, we abort generation for the current template and move on to the next template.

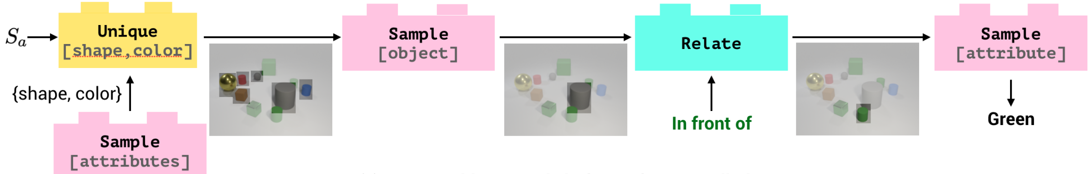
Caption Generation. The role of the caption is to *seed* the dialog and initialize S_q^0. In other words, caption gives Q-er partial information about the image so that asking follow-up questions is possible. Because A-er generates the caption, it uses the full scene graph S_a. Fig. 6 shows the caption grammar in action, producing three different captions for a given image. Consider the grammar for Fig. 6(c). First, `Sample[attribute]` produces {*shape, color*} used by `Unique` to select objects from S_a with unique shape and color attributes. An object (gray cylinder) is then sampled from these using `Sample[object]`. Next, a relation (*in front of*) is enforced via a `Relate` primitive leading to the green cylinder in front of the gray cylinder. Finally, `Sample[attribute]` samples one of the attributes to give us the caption, *'A green object stands in front of a gray cylinder.'*

Question Generation. Unlike the caption, the questions are generated by the Q-er, having access only to a partial scene graph S_q^t at round t. This S_q^t is an assimilation of information from the previous rounds of the dialog. The primitives in the question template therefore take S_q^t as the input scene graph, and the generation proceeds in a manner similar to that of the caption explained above. As the dialog is driven by Q-er based on partial scene information, only a few questions are non-redundant (or even plausible) at a given round of the dialog. To this



(a) There are four green objects in the scene.

(b) The image contains a gray cylinder.



(c) A green object stands in front of a gray cylinder.

Figure 6: Usage of dialog grammar in caption generation. See text for details.

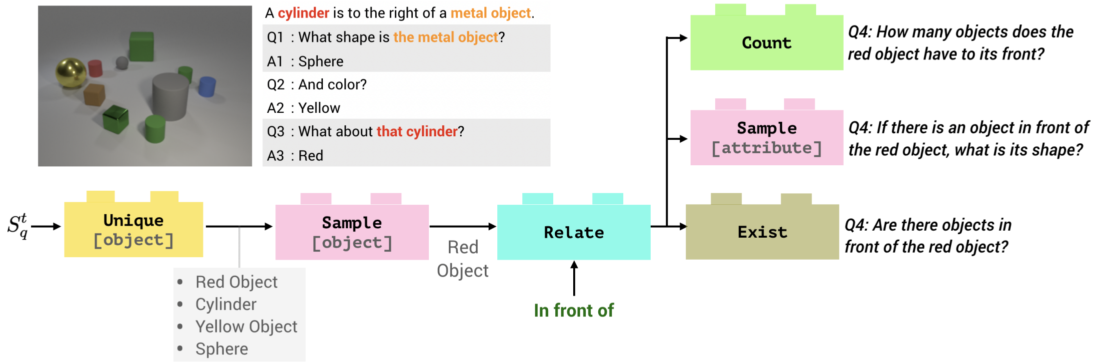


Figure 7: Usage of dialog grammar in question generation. See text for details.

end, the inherent constraints associated with the primitives now play a bigger role in the template selection.

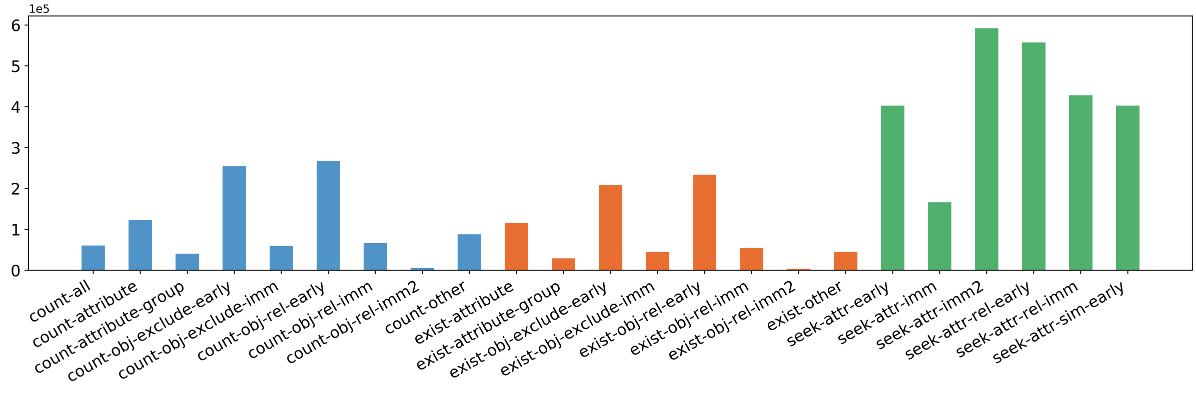
Consider Fig. 7 that shows how the current question is generated using the primitives and grammar, given the caption and dialog history (question-answer pair for the first three rounds). For the current round, the question *'What material is the green object at the back?'* is clearly implausible (Q-er is unaware of the existence of a green object), while the question *'What shape is the red object?'* is redundant. For the templates visualized, `Unique[object]` returns a list of unique known object-attribute pairs (using S_q^t). A candidate is sampled by `Sample[object]` and a relation is applied through `Relate(in front of)`. There are multiple choices at this junction: (a) The use of `Count` leads to a counting question (`count-obj-rel-early`), (b) Invoking `Sample[attribute]` results in a seek question (`seek-attr-rel-early`), and finally, (c) `Exist` primitive generates an exist question of type `exist-obj-rel-early`.

Dialog Generation. As specified in the main paper, we use beam search as a more tractable alternative to search through the exponential space of possible dialogs, by using additional constrains to retain only *interesting* dialogs. At every round of the dialog (after 3 rounds), we ensure that each of the question template types—count, existence, and seek—falls within a range (10% – 20% for count/existence each, and 30% – 60% for seek). In addition, we identify *independent* questions that do not need history to answer them, *e.g.*, *'How many objects are present in the image?'*, and limit their number to under 10%. Finally, to encourage questions that require reasoning over the history, *e.g.*, `seek-attr-sim-early` and `count-obj-excl-imm`, we tailor our beam search objective so that dialogs containing such questions have a higher value. We use a beam search with 100 beams for each dialog. Fig. 4 illustrates the diverse set of candidate questions generated at each round for a given image.

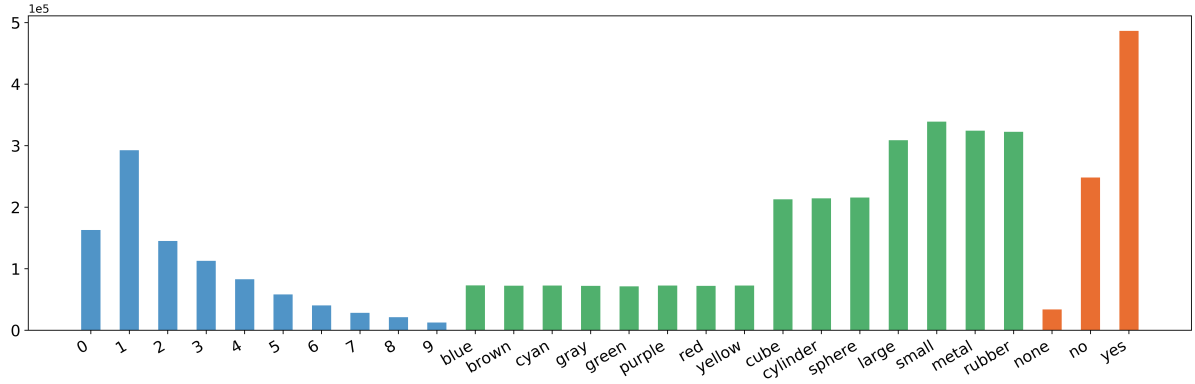
To summarize, the usage of primitives and a dialog grammar makes our generation procedure: (a) modular: each primitive has an intuitive meaning,

Captions	
`obj-relation`	*‘A [Z] [C] [M] [S] stands [R] a [Z1] [C1] [M1] [S1].’* *‘A gray sphere stands to the right of a red object.’*
`obj-unique`	*’A [Z] [C] [M] [S] is present in the image.’* *‘A red object is present in the image’*
`obj-extreme`	*‘The rightmost thing in the view is a [Z] [C] [M] [S].’* *‘The rightmost thing in the view is a cylinder.’*
`obj-count`	*‘The image has [X] [Z] [C] [M] [S].’* *‘The image has four cylinders.’*
Count/Exist Question Type	
`count-all`	*‘How many objects in the image?’*
`count/exist-excl`	*‘[How many | Are there] other [Z] [C] [M] [S] in the picture?’* *‘[How many | Are there] other cubes in the picture?’*
`count/exist-attr`	*‘[If present, how many | Are there] [Z] [C] [M] [S] objects?’* *‘[If present, how many | Are there] metallic objects?’*
`count/exist-attr-group`	*‘[How many | Are there] [Z] [C] [M] [S] among them?’* *‘[How many | Are there] blue cylinders among them?’*
`count/exist-obj-rel-imm`	*‘[How many | Are there] things to its [R]?’* *‘[How many | Are there] things to its right?’*
`count/exist-obj-rel-imm2`	*‘How about to its [R]?’* *‘How about to its left?’*
`count/exist-obj-rel-early`	*‘[How many | Are there] things [R] that [Z] [C] [M] [S]?’* *‘[How many | Are there] things in front of that shiny object?’*
`count/exist-obj-excl-imm`	*‘[How many | Are there] things that share its [A]?’* *‘[How many | Are there] things that share its color?’*
`count/exist-obj-excl-early`	*‘[How many | Are there] things that are the same [A] as that [Z] [C] [M] [S]?’* *‘[How many | Are there] things that are the same size as that round object?’*
Seek Question Type	
`seek-attr-imm`	*‘What is its [A]?’* *‘What is its shape?’*
`seek-attr-imm2`	*‘How about [A]?’* *‘How about color?’*
`seek-attr-early`	*‘What is the [A] of that [Z] [C] [M] [S]?’* *‘What is the shape of that shiny thing?’*
`seek-attr-sim-early`	*‘What about the earlier [Z] [C] [M] [S]?’* *‘What about the earlier box?’*
`seek-attr-rel-imm`	*‘If there is a thing to its [R], what [A] is it?’* *‘If there is a thing to its right, what color is it?’*
`seek-attr-rel-early`	*‘If there is a thing [R] that [Z] [C] [M] [S], what [A] is it made of?’* *‘If there is a thing in front of that shiny object, what material is it made of?’*

Table 3: Example templates for all the caption and question types used to generate CLEVR-Dialog dataset. For each type, we show both: (a) a sample template with placeholders (Z=size, C=color, M=material, S=shape, A=attribute, X=count, R=relation), and (b) a realization with placeholders filled with random values.



(a) Distribution of questions according to the template labels.



(b) Distribution of answers.

Figure 8: Visualization of distributions for question types and answers in our CLEVR-Dialog dataset. See Sec. C for more details.

(b) expressive: complex templates can be broken down into these primitives, (c) computationally efficient: outputs can reused for templates sharing similar primitive structures (as seen in Fig. 7), thus allowing an easy extension to new primitives and templates. We believe that CLEVR-Dialog represents not just a static dataset but also a recipe for constructing increasingly challenging grounded dialog by expanding this grammar.

C Additional Datasets Analysis

Fig. 8 visualizes the distribution of caption templates, question templates, answers, and the history dependency of questions in CLEVR-Dialog.

Caption Categories. As the dialog between Q-er and A-er is initiated by the caption, care must be taken to ensure it is *interesting enough* to spawn clarifying questions from the Q-er. To this end, we carefully design four different categories of caption templates (Fig. 3a): (a) `Obj-unique` mentions an object with unique set of attributes in the image, (b) `Obj-count` specifies the presence of a group of objects with common attributes, (c) `Obj-extreme` describes an object at one of the positional extremes of the image (right, left, fore, rear, center), (d) `Obj-relation` talks about the relationship between two objects along with their attributes in a way that allows them to be uniquely identified in the complete scene graph S_a. Example captions are given in Tab. 3. In contrast, MNIST Dialog does not have captions.

Question Categories and Types. CLEVR-Dialog contains three broad question categories—count, exist, and seek—with each further containing variants totaling up to 23 different types of questions. In comparison, MNIST-Dialog only has 5 types of questions and is less diverse. The distributions for the question categories and question types are shown in Fig. 3a and Fig. 8a, respectively. Our questions are 60% seek as they open up more interesting follow-up questions, 23% count, and 17% exist.

History Dependency. Recall that our motivation for CLEVR-Dialog to create a diagnostic dataset

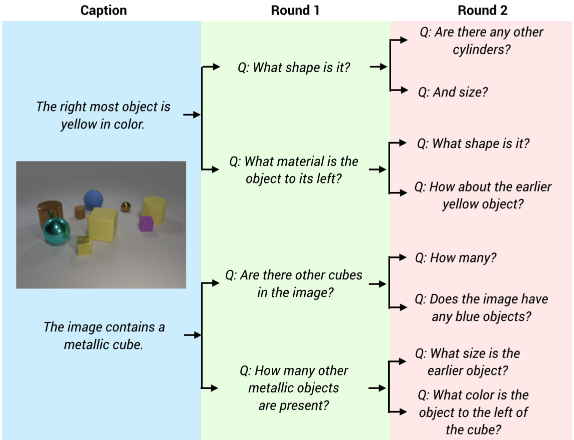
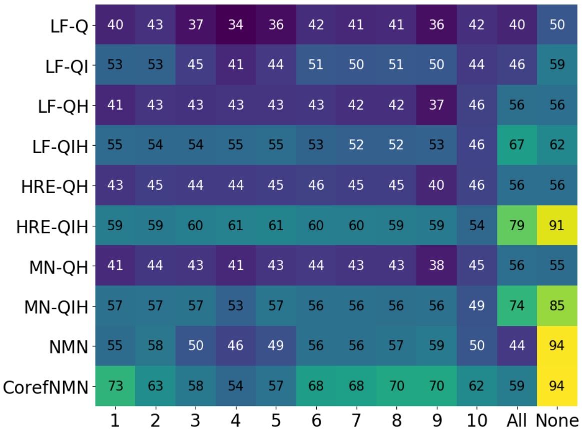


Figure 9: Dialog generation in CLEVR-Dialog. At each round, all valid question templates are used to generate candidates for the next question. However, only a few *interesting* candidates (beams) are retained for further generation, thus avoiding an exploding number of possibilities as rounds of dialog progress.



	1	2	3	4	5	6	7	8	9	10	All	None
LF-Q	40	43	37	34	36	42	41	41	36	42	40	50
LF-QI	53	53	45	41	44	51	50	51	50	44	46	59
LF-QH	41	43	43	43	43	43	42	42	37	46	56	56
LF-QIH	55	54	54	55	55	53	52	52	53	46	67	62
HRE-QH	43	45	44	44	45	46	45	45	40	46	56	56
HRE-QIH	59	59	60	61	61	60	60	59	59	54	79	91
MN-QH	41	44	43	41	43	44	43	43	38	45	56	55
MN-QIH	57	57	57	53	57	56	56	56	56	49	74	85
NMN	55	58	50	46	49	56	56	57	59	50	44	94
CorefNMN	73	63	58	54	57	68	68	70	70	62	59	94

Figure 10: Accuracy breakdown of models according to the history dependency type. While CorefNMN outperforms all methods on questions (average) containing references ($1-10$), it performs poorly on questions that depend on the entire history ('All').

for multi-round reasoning in visual dialog. As a result, a majority of questions in our dataset depend on the dialog history. We identify three major kinds of history dependency for the questions: (a) **Coreference** occurs when a phrase within the current question refers to a earlier mentioned object (referent). We characterize coreferences by measuring the distance between the current and the earlier mention, in terms of dialog rounds. This can range from 1 (*e.g.*, *'What is its color?'*) to 10 (a question in round 10 referring to an entity in the caption). (b) **All**: When the question depends on the entire dialog history, *e.g.*, *'How many other objects are present in the image?'*, (c) **None**: When the question is stand-alone and does not depend on the history, *e.g.*, *'How many spheres does the scene have?'* The distribution of questions characterized according to the history dependency is shown in Fig. 3b. Unlike MNIST Dialog, CLEVR-Dialog contains a good distribution of reference distances beyond just 1, leading to a mean distance of 3.2. Thus, the models will need to reason through different rounds of dialog history in order to succeed.

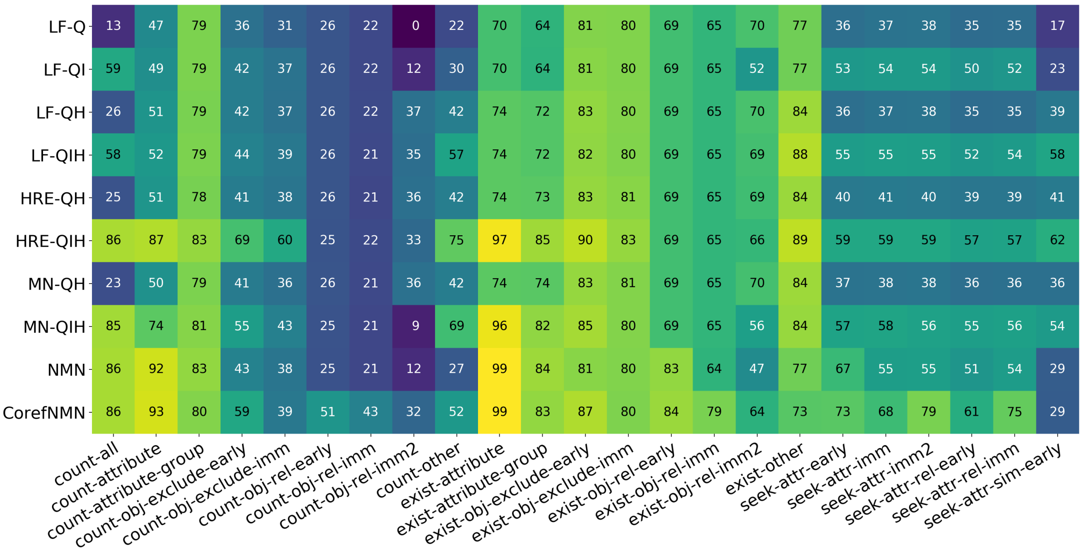
D Additional Model Analysis

In this section, we diagnose performance of all the models by breaking it down according to question type and history dependency. We then focus on the best performing model, CorefNMN (Kottur et al., 2018), which explicitly models coreferences in visual dialog by identifying the *reference* in the question (textual grounding) and then localizing the *referent* in the image (visual grounding). We study the behavior of CorefNMN on CLEVR-Dialog both qualitatively and quantitatively. Specifically, we visualize qualitative examples and develop metrics to quantitatively evaluate the textual and visual grounding. Note that such a diagnostic analysis is first of its kind which would not be possible without our CLEVR-Dialog.

D.1 Accuracy vs History Dependency

The breakdown of model performances based on this history dependency is presented in Fig. 10. The following are the key observations:

- The best performing model, CorefNMN, has a superior performance (on an average) on all question with coreference ($1-10$) compared to all other models. As CorefNMN is designed specifically to handle coreferences in visual dialog, this is not surprising.
- Interestingly, the second best model HRE-QIH has the best accuracy on 'All' questions, even beating CorefNMN by a margin of 20%. In other words, HRE-QIH (and even MN-QIH) is able to answer 'All' questions significantly better than CorefNMN perhaps due to the ability of its dialog-level RNN to summarize information as the dialog progresses.
- Both NMN and CorefNMN perform similarly on the 'None' questions. This observation is intuitive as NMN is a history-agnostic version of CorefNMN by construction. However, the difference becomes evident as CorefNMN outperforms NMN by about 12% overall.



	count-all	count-attribute	count-attribute-group	count-obj-exclude-early	count-obj-exclude-imm	count-obj-rel-early	count-obj-rel-imm	count-obj-rel-imm2	count-other	exist-attribute	exist-attribute-group	exist-obj-exclude-early	exist-obj-exclude-imm	exist-obj-rel-early	exist-obj-rel-imm	exist-obj-rel-imm2	exist-other	seek-attr-early	seek-attr-imm	seek-attr-imm2	seek-attr-rel-early	seek-attr-rel-imm	seek-attr-sim-early
LF-Q	13	47	79	36	31	26	22	0	22	70	64	81	80	69	65	70	77	36	37	38	35	35	17
LF-QI	59	49	79	42	37	26	22	12	30	70	64	81	80	69	65	52	77	53	54	54	50	52	23
LF-QH	26	51	79	42	37	26	22	37	42	74	72	83	80	69	65	70	84	36	37	38	35	35	39
LF-QIH	58	52	79	44	39	26	21	35	57	74	72	82	80	69	65	69	88	55	55	55	52	54	58
HRE-QH	25	51	78	41	38	26	21	36	42	74	73	83	81	69	65	69	84	40	41	40	39	39	41
HRE-QIH	86	87	83	69	60	25	22	33	75	97	85	90	83	69	65	66	89	59	59	59	57	57	62
MN-QH	23	50	79	41	36	26	21	36	42	74	74	83	81	69	65	70	84	37	38	38	36	36	36
MN-QIH	85	74	81	55	43	25	21	9	69	96	82	85	80	69	65	56	84	57	58	56	55	56	54
NMN	86	92	83	43	38	25	21	12	27	99	84	81	80	83	64	47	77	67	55	55	51	54	29
CorefNMN	86	93	80	59	39	51	43	32	52	99	83	87	80	84	79	64	73	73	68	79	61	75	29

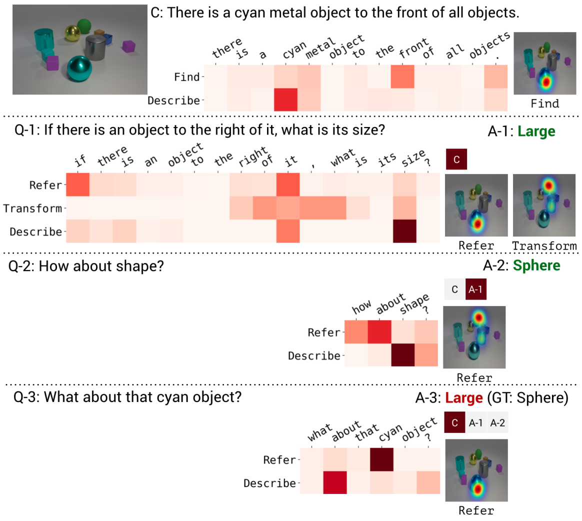
Figure 11: Accuracy breakdown of models according to the question type.

D.2 Accuracy vs Question Type

Fig. 11 breaks down the performance of all the models according to the question types. An obvious observation is that performance on counting and seek questions is worse than that on exist questions. While this is in part because of the binary nature of exist questions, they are also easier to answer than counting or extracting attributes that need complicated visual understanding.

D.3 Qualitative Anaylsis for CorefNMN

We now qualitatively visualize (Fig. 12) the best performing model, CorefNMN. In the example shown, CorefNMN first parses the caption *'There is a cyan metal object to the front of all the objects.'* and localizes the right cyan object. While answering Q-1, CorefNMN rightly instantiates the `Refer` module and applies the desired transformation (see module outputs on the right). For Q-2, it accurately identifies the object as the previous one, and extracts the attributes. Finally, the question *'What about that cyan object?'* cannot be answered in isolation as: (a) there are multiple cyan objects, (b) the meaning of the question is incomplete without Q-2. It is interesting to note that even though CorefNMN overcomes (a) by correctly resolving the reference *that cyan object* (in the image), it is unable to circumvent (b) due to its specialization in visual coreferences.



C: There is a cyan metal object to the front of all objects.

Q-1: If there is an object to the right of it, what is its size?

A-1: Large

Q-2: How about shape?

A-2: Sphere

Q-3: What about that cyan object?

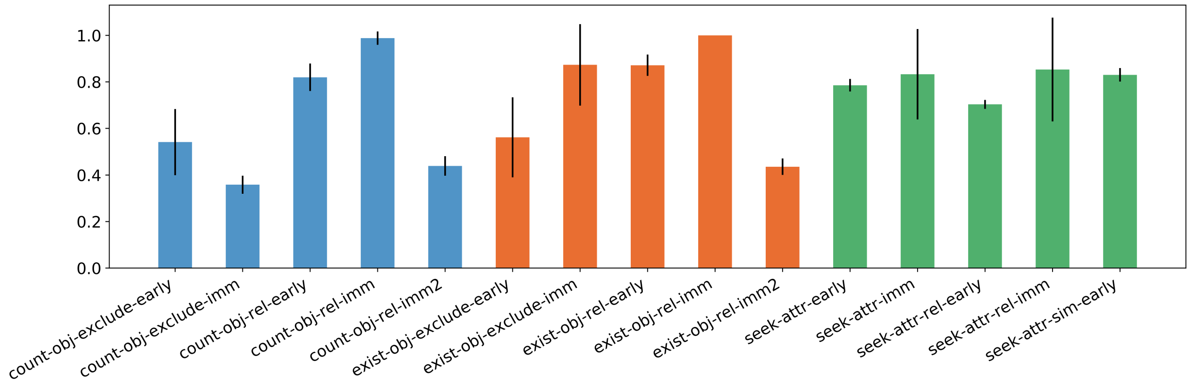
A-3: Large (GT: Sphere)

Figure 12: Qualitative visualization of CorefNMN on CLEVR-Dialog.

D.4 Grounding Analysis for CorefNMN

As shown in Fig. 12, CorefNMN identifies a reference phrase in the current question and proceeds to visually ground the corresponding referent in the image. Such explicit textual and visual grounding at each round allows for an interesting quantitative analysis for CorefNMN, with the help of annotations in our CLEVR-Dialog. To elaborate, CLEVR-Dialog provides coreference annotations for each question, if any, in the form of a reference phrase and its bounding box localization in the image. By comparing these grounding annotations with the output from the model, we can quantitatively assess grounding (both textual and visual) by CorefNMN. In what follows, we first describe the ground annotations, detail the evaluation procedure and then present our observations.

(a) NDCG value for text grounding for various question types.



(b) NDCG value for visual grounding for various question types.

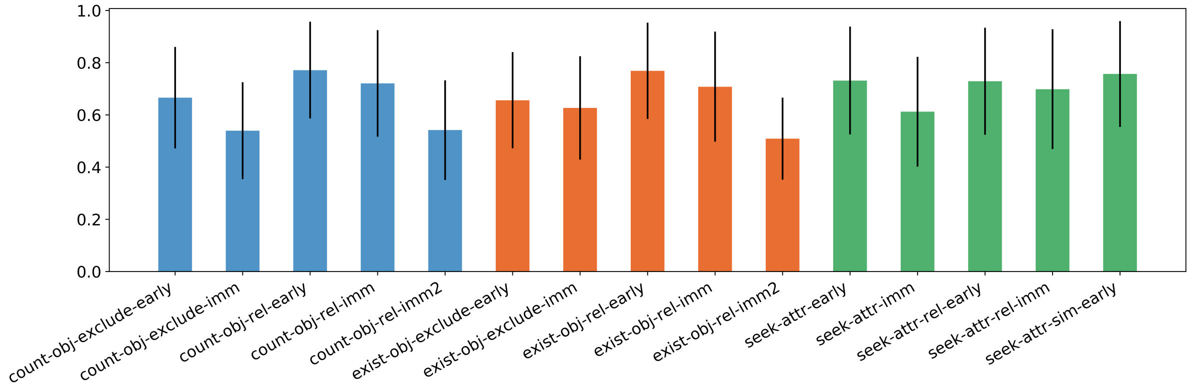


Figure 13: Evaluating the textual (above) and visual (below) grounding of CorefNMN on CLEVR-Dialog, using Normalized Discounted Cumulative Gain (NDCG) for various question types. Higher is better.

Annotations. While the original CLEVR dataset (Johnson et al., 2017) does not contain bounding box annotations for the objects in the scene, Krishna et al. (2018) later added these in their work on referring expressions. We leverage these annotations to obtain the ground truth visual groundings (A_V) for the referents in our questions. On the other hand, each of the caption and question templates has referring phrase annotations in them, thus giving the ground truth textual groundings (A_T). We use the above two groundings for evaluation.

Evaluation. For every coreference resolution, CorefNMN produces an visual attention map of size 14×14 ($\widehat{A}_V$) and a textual attention over the question words ($\widehat{A}_T$). We rank all the $14^2 = 196$ cells in $\widehat{A}_V$ according to their attention values. Next, we obtain the relevant cells among them from an appropriately scaled down (14×14) version of A_V and Next, we appropriately scaled down A_V (14×14) and consider the cells spanning the bounding box as relevant. To evaluate grounding, we measure the retrieval performance of the relevant cells in the sorted $\widehat{A}_V$ through the widely used Normalized Discounted Cumulative Gain (NDCG)[2]. It is a measure of how highly the relevant cells were ranked in the sorted $\widehat{A}_V$, with a logarithmic weighting scheme to higher ranks, thus higher is better. For the textual grounding, we perform a similar computation between $\widehat{A}_T$ and A_T and report NDCG.

Observations. The NDCG values to evalute both textual and visual groundings for CorefNMN are shown in Fig. 13. An important takeaway being that the model is able to accurately ground the references in the question (Fig. 13a) consistently for several question types, as reflected in an higher average NDCG. On the other hand, the visual grounding in Fig. 13b is inferior compared to the ground truth annotations with a mean of around 0.3 and a high variance. This trend remains the same across all the question types. A possible hypothesis is that while the model is able to identify the references

---

[2] `https://en.wikipedia.org/wiki/Discounted_cumulative_gain`

in text, it is unable to resolve and ground the referent accurately in the image–an area of potential improvement.

E Implementation details

Dataset generation was done entirely in Python, without any significant additional package dependencies. To evaluate the models from Das et al. (2017a), we use their open source implementation[3] based on Lua Torch[4]. For the neural module architectures (Hu et al., 2017; Kottur et al., 2018), we use the authors' Python-based, publicly available implementations—NMN[5] and CorefNMN[6]. Questions are encoded by first learning a 128-dimensional embedding for the words, which are then fed into a single layer LSTM of hidden size 128. We use a pretrained convolution neural network, ResNet-101 (He et al., 2016), to extract features for the images. Adam (Kingma and Ba, 2014) steps with a learning rate of 0.0001 are employed to maximize the loglikelihood of the ground truth answer, while training. A small portion (500 images) from the training set is set aside to pick the best performing model via early stopping. Our code and dataset are publicly available[7].

---

[3] `https://github.com/batra-mlp-lab/visdial`

[4] `http://torch.ch/`

[5] `https://github.com/ronghanghu/n2nmn`

[6] `https://github.com/facebookresearch/corefnmn`

[7] `https://github.com/satwikkottur/clevr-dialog`

Learning Outside the Box:
Discourse-level Features Improve Metaphor Identification

Jesse Mu[1,3], **Helen Yannakoudakis**[2], **Ekaterina Shutova**[4]
[1]Computer Science Department, Stanford University, USA
[2]The ALTA Institute, [3]Dept. of CS & Technology, University of Cambridge, UK
[4]ILLC, University of Amsterdam, The Netherlands
`muj@stanford.edu, hy260@cl.cam.ac.uk, e.shutova@uva.nl`

Abstract

Most current approaches to metaphor identification use restricted linguistic contexts, e.g. by considering only a verb's arguments or the sentence containing a phrase. Inspired by pragmatic accounts of metaphor, we argue that broader discourse features are crucial for better metaphor identification. We train simple gradient boosting classifiers on representations of an utterance and its surrounding discourse learned with a variety of document embedding methods, obtaining near state-of-the-art results on the 2018 VU Amsterdam metaphor identification task without the complex metaphor-specific features or deep neural architectures employed by other systems. A qualitative analysis further confirms the need for broader context in metaphor processing.

1 Introduction

From *bottled up anger* to *the world is your oyster*, metaphor is a defining component of language, adding poetry and humor to communication (Glucksberg and McGlone, 2001) and serving as a tool for reasoning about relations between concepts (Lakoff and Johnson, 1980). Designing metaphor processing systems has thus seen significant interest in the NLP community, with applications from information retrieval (Korkontzelos et al., 2013) to machine translation (Saygin, 2001).

An important first step in any metaphor processing pipeline is metaphor *identification*. To date, most approaches to its identification operate in restricted contexts, for instance, by only considering isolated verb–argument pairs (e.g. *deflate economy*) (Rei et al., 2017) or the sentence containing an utterance (Gao et al., 2018). However, wider context is crucial for understanding metaphor: for instance, the phrase *drowning students* can be interpreted as literal (in the context of *water*) or metaphorical (in the context of *homework*). Often the context required extends beyond the immediate sentence; in Table 1, coreferences (*them*) must be resolved to understand the arguments of a verb, and a *game* is metaphorical in a political context. Indeed, a rich linguistic tradition (Grice, 1975; Searle, 1979; Sperber and Wilson, 1986) explains metaphor as arising from violations of expectations in a conversational context.

"You can't steal their ideas." "No, idiot—not so I can **steal** them."
Britain still can't decide when to **play** the mandarinate game of of silence [. . .] interests and concern of the Chinese government.

Table 1: Metaphorical examples from the VUA dataset in context. Verb is bolded, arguments underlined. Immediate sentence in black, with further context in gray.

Following these theories, in this paper we argue that metaphor processing models should expand beyond restricted contexts to use representations of wider discourse. We support this claim with two contributions: (1) we develop metaphor identification models which take as input an utterance, its immediate lexico–syntactic context, and broader discourse representations, and demonstrate that incorporating discourse features improves performance; (2) we perform a qualitative analysis and show that broader context is often required to correctly interpret metaphors. To the best of our knowledge, this is the first work to investigate the effects of broader discourse on metaphor identification.[1]

2 Related work

Metaphor identification is typically framed as a binary classification task, either with (1) word tu-

---

[1] Code and data available at `https://github.com/jayelm/broader-metaphor`.

Proceedings of NAACL-HLT 2019, pages 596–601
Minneapolis, Minnesota, June 2 - June 7, 2019.

ples such as SVO triples (car *drinks* gasoline) or (2) whole sentences as input, where the goal is to predict the metaphoricity of a token in the sentence. Recent work has used a variety of features extracted from these two types of contexts, including selectional preferences (Shutova, 2013; Beigman Klebanov et al., 2016), concreteness/imageability (Turney et al., 2011; Tsvetkov et al., 2014), multi-modal (Tekiroglu et al., 2015; Shutova et al., 2016) and neural features (Do Dinh and Gurevych, 2016; Rei et al., 2017).

At the recent VU Amsterdam (VUA) metaphor identification shared task (Leong et al., 2018), neural approaches dominated, with most teams using LSTMs trained on word embeddings and additional linguistic features, such as semantic classes and part of speech tags (Wu et al., 2018; Stemle and Onysko, 2018; Mykowiecka et al., 2018; Swarnkar and Singh, 2018). Most recently, Gao et al. (2018) revisited this task, reporting state-of-the-art results with BiLSTMs and contextualized word embeddings (Peters et al., 2018). To the best of our knowledge, none of the existing approaches have utilized information from wider discourse context in metaphor identification, nor investigated its effects.

3 Data

Following past work, we use the *Verbs* subset of the VUA metaphor corpus (Steen et al., 2010) used in the above shared task. The data consists of 17240 training and 5873 test examples, equally distributed across 4 genres of the British National Corpus: Academic, Conversation, News, and Fiction. All verbs are annotated as metaphorical or literal in these texts. We sample 500 examples randomly from the training set as a development set.

4 Models

For each utterance, our models learn generic representations of a *verb lemma*,[2] its syntactic *arguments*, and its broader discourse *context*. We concatenate these features into a single feature vector and feed them into a gradient boosting decision tree classifier (Chen and Guestrin, 2016).[3] By observing performance differences when using the lemma only (L), lemma + arguments (LA), or lemma + arguments + context (LAC), we can investigate the effects of including broader context.

To obtain arguments for verbs, we extract subjects and direct objects with Stanford CoreNLP (Manning et al., 2014). 67.4% of verb usages in the dataset have at least one argument; absent arguments are represented as zero vectors. To obtain the broader *context* of a verb, we take its surrounding paragraph as defined by the BNC; the average number of tokens in a context is 97.3. Figure 1 depicts the feature extraction and classification pipeline of our approach.

To learn representations, we use several widely-used embedding methods:[4]

GloVe We use 300-dimensional pre-trained GloVe embeddings (Pennington et al., 2014) trained on the Common Crawl corpus as representations of a lemma and its arguments. To learn a context embedding, we simply average the vectors of the tokens in the context. Out-of-vocabulary words are represented as a mean across all vectors.

doc2vec We use pretrained 300-dimensional paragraph vectors learned with the distributed bag-of-words method of Le and Mikolov (2014) (colloquially, doc2vec), trained on Wikipedia (Lau and Baldwin, 2016). Here, paragraph vectors are learned to predict randomly sampled words from the paragraph, ignoring word order. To extract representations for verbs and arguments, we embed one-word "documents" consisting of only the word itself.[5] We use a learning rate $\alpha = 0.01$ and 1000 epochs to infer vectors.

Skip-thought We use pretrained skip-thought vectors (Kiros et al., 2015) learned from training an encoder–decoder model to reconstruct the surrounding sentences of an input sentence from the Toronto BooksCorpus (Zhu et al., 2015). From this model, we extract 4800-dimensional representations for verb lemma, arguments, and contexts.

ELMo Finally, we use ELMo, a model of deep contextualized word embeddings (Peters et al., 2018). We extract 1024-dimensional representations from the last layer of a stacked BiLSTM

---

[2] The lemmatized form of the verb has improved generalization in other systems (Beigman Klebanov et al., 2016).

[3] We use the default parameters of the XGBoost package: a maximum tree depth of 3, 100 trees, and $\eta = 0.1$.

[4] These methods differ significantly in dimensionality and training data. Our intent is not to exhaustively compare these methods, but rather claim generally that many embeddings give good performance on this task.

[5] Since some methods provide only document embeddings and not word embeddings, for consistency, in all methods we use the same embedding process even for single-word verbs and arguments.

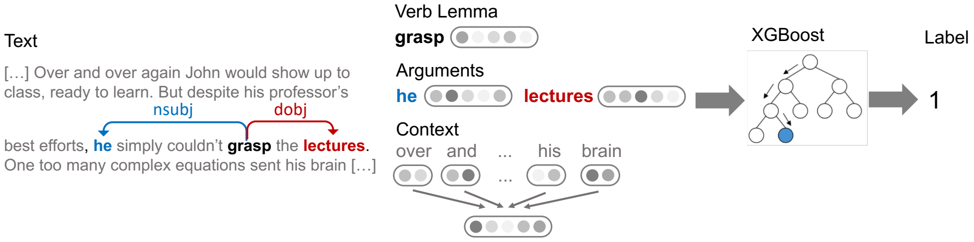


Figure 1: The general feature extraction and classification pipeline of our approach.

trained on Wikipedia and monolingual news data from WMT 2008–2012. To learn embeddings for verbs and arguments, we extract representations for sentences containing only the word itself. To learn context embeddings, we again average the constituent word embeddings.

5 Evaluation

For each embedding method, we evaluate the three configurations of features—L, LA, and LAC—on the VUA shared task train/test split, reporting precision, recall and F1 score. Since we are interested in whether incorporating broader context significantly improves identification performance, we compare successive model predictions (LAC vs. LA; LA vs. L) using the *mid-p* variant of McNemar's test for paired binary data (Fagerland et al., 2013).

5.1 Comparison Systems

We first compare our models to the baselines of the VUA shared task (Leong et al., 2018): *Baseline 1*, a logistic regression classifier trained only on one-hot encodings of verb lemmas; and *Baseline 2*, the same classifier with additional WordNet class and concreteness features. We also compare to the best systems submitted to the VUA shared task: Wu et al. (2018), an ensemble of 20 CNN-BiLSTMs trained on word2vec embeddings, part-of-speech tags, and word embedding clusters; and Stemle and Onysko (2018), a BiLSTM trained on embeddings from English language learner corpora.

5.2 Results

Results for our models are presented in Table 2. Interestingly, most of the simple lemma models (L) already perform at Baseline 2 level, obtaining F1 scores in the range 60–62. This is likely due to the generalization made possible by dense representations of lemmas (vs. one-hot encodings) and the more powerful statistical classifier used. As expected, the addition of argument information consistently enhances performance.

Model		P	R	F1
Baseline 1 (lemma)		51.0	65.4	57.3
Baseline 2 (+WN, concrete)		52.7	69.8	60.0
Stemle and Onysko (2018)		54.7	**77.9**	64.2
Wu et al. (2018)		**60.0**	76.3	**67.2**
GloVe	L (lemma)	51.6	74.1	60.8
	LA (+ args)	54.0	74.4	62.6^{***}
	LAC (+ ctx)	56.7	76.8	65.2^{***}
doc2vec	L	48.8	72.1	58.2
	LA	50.5	71.4	59.1^{**}
	LAC	52.7	72.2	60.9^{***}
skip-thought	L	53.5	76.1	62.8
	LA	57.0	74.0	64.3^{***}
	LAC	59.5	75.4	66.5^{***}
ELMo	L	51.3	74.9	60.9
	LA	56.0	73.5	63.6^{***}
	LAC	58.9	77.1	66.8^{***}

**,*** Significant improvement over previous model ($p < 0.01, 0.001$).

Table 2: Metaphor identification results.

Crucially, the addition of broader discourse context improves performance for all embedding methods. In general, we observe consistent, statistically significant increases of 2-3 F1 points for incorporating discourse. Overall, all LAC models except doc2vec exhibit high performance, and would have achieved second place in the VUA shared task. These results show a clear trend: the incorporation of discourse information leads to improvement of metaphor identification performance across models.

Table 3 displays the performance breakdown by genre in the VUA test set for our best performing model (ELMo LAC) and selected comparison systems. Echoing Leong et al. (2018), we observe that the *Conversation* and *Fiction* genres are consistently more difficult than the *Academic* and *News* genres across all models. This is partially because in this dataset, metaphors in these genres are rarer, occuring 35% of the time in *Academic* and 43% in *News*, but only 15% in *Conversation* and 24% in *Fiction*. In addition, for our model specifically, Conversation genre contexts are much

Genre	Model	P	R	F1
Academic	Baseline 2	70.7	83.6	**76.6**
	Wu et al. (2018)	**74.6**	76.3	75.5
	ELMo LAC	65.4	**86.8**	74.6
Conversation	Baseline 2	30.1	82.1	44.1
	Wu et al. (2018)	40.3	65.6	**50.3**
	ELMo LAC	**42.6**	56.0	48.4
Fiction	Baseline 2	40.7	66.7	50.6
	Wu et al. (2018)	**54.5**	**78.4**	**57.6**
	ELMo LAC	48.2	63.0	54.6
News	Baseline 2	67.7	68.9	68.3
	Wu et al. (2018)	**69.4**	74.4	71.8
	ELMo LAC	65.2	**80.0**	**71.8**

Table 3: Performance breakdown by genre for ELMo LAC model and comparison systems.

	Args	Sentence	Paragraph
Overall	40	49	11
Model errors			
ELMo L	37	50	13
ELMo LA	36	49	15
ELMo LAC	39	53	8

Table 4: Types of context required to interpret metaphors in the development set, both overall (first row) and for model errors. Each row is a separate (but overlapping) sample from the development set.

shorter on average (23.8 vs. 97.3).

Our best performing model (ELMo LAC) is within 0.4 F1 score of the first-place model in the VUA shared task (Wu et al., 2018). The GloVe LAC model would also have obtained second place at 65.2 F1, yet is considerably simpler than the systems used in the shared task, which employed ensembles of deep neural architectures and hand-engineered, metaphor-specific features.

6 Qualitative analysis

To better understand the ways in which discourse information plays a role in metaphor processing, we randomly sample 100 examples from our development set and manually categorize them by the amount of context required for their interpretation. For instance, a verb may be interpretable when given just its arguments (direct subject/object), it may require context from the enclosing sentence, or it may require paragraph-level context (or beyond). We also similarly analyze 100 sampled errors made on the development set by the ELMo L, LA, and LAC models, to examine whether error types vary between models.

Our analysis in Table 4 shows that 11% of examples in the development set require paragraph-level context for correct interpretation. Indeed, while such examples are frequently misclassified by the L and LA models (13%, 15%), the error rate is halved when context is included (8%).

Table 5 further presents examples requiring at least paragraph-level context, along with gold label and model predictions. Out of the 31 unique such examples identified in the above analyses, we found 11 (35%) requiring explicit coreference resolution of a pronoun or otherwise underspecified noun (e.g. Table 5 row 1) and 5 (16%) which reference an entity or event implicitly (*ellipsis*; e.g. Table 5 row 2). However, we also observed 4 errors (13%) due to examples with non-verbs and incomplete sentences and 11 examples (35%) where not even paragraph-level context was sufficient for interpretation, mostly in the *Conversation* genre, demonstrating the subjective and borderline nature of many of the annotations.

This analysis shows *a priori* the need for broader context beyond sentence-level for robust metaphor processing. Yet this is not an upper bound on performance gains; the general improvement of the LAC models over LA shows that even when context is not strictly necessary, it can still be a useful signal for identification.

7 Conclusion

We presented the first models which leverage representations of discourse for metaphor identification. The performance gains of these models demonstrate that incorporating broader discourse information is a powerful feature for metaphor identification systems, aligning with our qualitative analysis and the theoretical and empirical evidence suggesting metaphor comprehension is heavily influenced by wider context.

Given the simplicity of our representations of context in these models, we are interested in future models which (1) use discourse in more sophisticated ways, e.g. by modeling discourse relations or dialog state tracking (Henderson, 2015), and (2) leverage more sophisticated neural architectures (Gao et al., 2018).

Acknowledgments

We thank anonymous reviewers for their insightful comments, Noah Goodman, and Ben Leong for assistance with the 2018 VUA shared task data.

Sentence	Gold label	LA	LAC
A major complication [. . .] is that the environment can rarely be treated as in a laboratory experiment. **Given** this, determining the nature of the interactions between the variables becomes a matter of major difficulty.	0	1	0
For example, on high policy common opinion said that there was nothing for it but to stay in the ERM. He **stayed** in, and the recession worsened.	1	0	1

Table 5: Examples where context helps, along with gold label (0 – literal; 1 – metaphor) and model predictions (LA, LAC). Verb is bolded, arguments underlined. Additional context (needed for interpretation) in gray.

We thank the Department of Computer Science and Technology and Churchill College, University of Cambridge for travel funding. Jesse Mu is supported by a Churchill Scholarship and an NSF Graduate Research Fellowship. Helen Yannakoudakis was supported by Cambridge Assessment, University of Cambridge. We thank the NVIDIA Corporation for the donation of the Titan GPU used in this research.

References

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutierrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 101–106.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33.

Morten W Fagerland, Stian Lydersen, and Petter Laake. 2013. The McNemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional. *BMC Medical Research Methodology*, 13(1):91.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613.

Sam Glucksberg and Matthew S McGlone. 2001. *Understanding figurative language: From metaphor to idioms*. Oxford University Press, Oxford.

Herbert P Grice. 1975. Logic and conversation. In Peter Cole and Jerry L Morgan, editors, *Syntax and Semantics*, volume 3, pages 41–58. Academic Press, New York.

Matthew Henderson. 2015. Machine learning for dialog state tracking: A review. In *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. 2013. SemEval-2013 task 5: Evaluating phrasal semantics. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 39–47.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196.

Chee Wee (Ben) Leong, Beata Beigman Klebanov, and Ekaterina Shutova. 2018. A report on the 2018 VUA metaphor detection shared task. In *Proceedings of the Workshop on Figurative Language Processing*, pages 56–66.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Agnieszka Mykowiecka, Aleksander Wawer, and Malgorzata Marciniak. 2018. Detecting figurative word occurrences using recurrent neural networks. In *Proceedings of the Workshop on Figurative Language Processing*, pages 124–127.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1537–1546.

Ayse Pinar Saygin. 2001. Processing figurative language in a multi-lingual task: Translation, transfer and metaphor. In *Proceedings of the Workshop on Corpus-based and Processing Approaches to Figurative Language*.

John Searle. 1979. Metaphor. In *Expression and Meaning: Studies in the Theory of Speech Acts*, pages 76–116. Cambridge University Press, Cambridge and New York.

Ekaterina Shutova. 2013. Metaphor identification as interpretation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 276–285.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 160–170.

Dan Sperber and Deirdre Wilson. 1986. *Relevance: Communication and cognition*. Harvard University Press, Cambridge, MA.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*. John Benjamins Publishing Company, Amsterdam.

Egon Stemle and Alexander Onysko. 2018. Using language learner data for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing*, pages 133–138.

Krishnkant Swarnkar and Anil Kumar Singh. 2018. Di-LSTM contrast : A deep neural network for metaphor detection. In *Proceedings of the Workshop on Figurative Language Processing*, pages 115–120.

Serra Sinem Tekiroglu, Gözde Özbal, and Carlo Strapparava. 2015. Exploring sensorial features for metaphor identification. In *Proceedings of the Third Workshop on Metaphor in NLP*, pages 31–39.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 248–258.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. Neural metaphor detecting with CNN-LSTM model. In *Proceedings of the Workshop on Figurative Language Processing*, pages 110–114.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

Detection of Abusive Language: the Problem of Biased Datasets

Michael Wiegand[*0], **Josef Ruppenhofer**[†‡], **Thomas Kleinbauer**[*]
[*]Spoken Language Systems, Saarland University, Saarbrücken, Germany
[†]Leibniz ScienceCampus, Heidelberg/Mannheim, Germany
[‡]Institute for German Language, Mannheim, Germany
`michael.wiegand@lsv.uni-saarland.de`
`ruppenhofer@ids-mannheim.de`
`thomas.kleinbauer@lsv.uni-saarland.de`

Abstract

We discuss the impact of data bias on abusive language detection. We show that classification scores on popular datasets reported in previous work are much lower under realistic settings in which this bias is reduced. Such biases are most notably observed on datasets that are created by focused sampling instead of random sampling. Datasets with a higher proportion of implicit abuse are more affected than datasets with a lower proportion.

1 Introduction

Abusive or offensive language is commonly defined as hurtful, derogatory or obscene utterances made by one person to another person.[1] Examples are (1)-(3). In the literature, closely related terms include *hate speech* (Waseem and Hovy, 2016) or *cyber bullying* (Zhong et al., 2016). While there may be nuanced differences in meaning, they are all compatible with the general definition above.

(1) stop editing this, you **dumbass**.
(2) Just want to slap the **stupid** out of these **bimbos**!!!
(3) Go lick a pig you arab muslim piece of **scum**.

Due to the rise of user-generated web content, in particular on social media networks, the amount of abusive language is also steadily growing. NLP methods are required to focus human review efforts towards the most relevant microposts.

In this paper, we examine the issue of data bias. For the creation of manually annotated datasets, randomly sampling microposts from large social media platforms typically results in a too small proportion of abusive comments (Wulczyn et al., 2017; Founta et al., 2018). Therefore, more focused sampling strategies have to be applied which cause biases in the resulting datasets. We show what implications this has on classifiers trained on these datasets: Previous evaluations reported high classification performance on datasets with difficult cases of abusive language, e.g. implicit abuse (§2). Contrarily, we find that the high classification scores are likely to be the result of modeling the bias in those datasets.

Although we will explicitly name shortcomings of existing individual datasets, our paper is not intended as a reproach of those who created them. On the contrary, we acknowledge the great efforts the researchers have taken to provide these resources. Without them, much existing research would not have been possible. However, we also noticed a lack of awareness of the special properties of those datasets among researchers using them. As we will illustrate with specific examples, this may result in unforeseen results of particular classification approaches.

2 Explicit and Implicit Abuse

One major distinction that has been proposed in the literature is the division into **explicitly** and **implicitly** abusive language (Waseem et al., 2017). The former are microposts that employ some abusive words (1)-(3) (e.g. *dumbass* or *scum*), while the latter represents the more difficult case in which the abusive nature is conveyed by other means, such as sarcasm, jokes, and particularly the usage of negative stereotypes etc. (4)-(5).

(4) i havent had an intelligent conversation with a woman.
(5) Jews don't marry children. Muslims do. All the time.

To determine which of the datasets that we consider in this work contain which type of abusive language, we proceeded as follows. On the set of abusive microposts of each dataset, we computed the proportion of microposts that include at least

---

[0] Present affiliation: Leibniz ScienceCampus, Heidelberg/Mannheim, Germany

[1] http://thelawdictionary.org/

Proceedings of NAACL-HLT 2019, pages 602–608
Minneapolis, Minnesota, June 2 - June 7, 2019.

one abusive word according to the lexicon of abusive words from Wiegand et al. (2018a). Datasets with a high proportion of abusive words typically contain a high amount of explicitly abusive microposts, whereas datasets with a low proportion contain a higher amount of implicitly abusive language. The resulting figures, of course, are only a lower bound estimate for explicit language abuse. There will also be microposts containing abusive words that are missing from the lexicon. However, after manual inspection of a sample of microposts, we are fairly confident that this does not significantly change the *relative* order of datasets when ranked according to their degree of explicit language abuse.

3 Datasets and Their Properties

Due to the limited space of this paper, we restrict our discussion to frequently cited (publicly available) datasets and datasets from shared tasks. Substantial interannotation agreement has also been reported with these datasets.

As we focus on the detection of abusive language in general, for those datasets containing more fine-grained class inventories describing subtypes of abusive language[2], we conflate the categories to one general category. As a result, there are always only two categories: *abuse* and *no-abuse*. This merging removes differences between the individual annotation schemes that would otherwise impede a meaningful comparison.

Table 1 shows a brief summary of the different datasets. Among the properties, we list the performance of a text classifier in the right-most column. Since in previous work performance on the different datasets was reported on the basis of different types of classifiers and also varying evaluation metrics, we ran the same classifier on all datasets in order to ensure a meaningful comparison. We chose FastText, which is an efficient supervised classifier known to produce state-of-the-art performance on many text classification tasks[3] (Joulin et al., 2017) and whose results are easy to reproduce. Performance is evaluated in a 10-fold crossvalidation setting using the macro-average F1-score.

Table 1 also describes the way the datasets were sampled. Not a single dataset has been produced by *pure* random sampling. This would always result in tiny proportions of abusive language. For example, Founta et al. (2018) estimate that on Twitter, there are only between 0.1% up to at most 3% abusive tweets. What comes closest to random sampling is the procedure followed by Founta et al. (2018), Razavi et al. (2010) and the Kaggle-challenge.[4] They took a random sample and applied some heuristics in order to boost the proportion of abusive microposts. For instance, in the Kaggle-challenge, further microposts from users were added who had been blocked due to being reported to post personal attacks.

The procedures applied by other researchers are more drastic because, as we show in §4 and §5, they affect more heavily the topic distribution of the dataset. These approaches do not even start with a random sample. The topic distribution is mostly determined by the creators of the dataset themselves. For example, Waseem and Hovy (2016) extract tweets matching query words likely to co-occur with abusive content. Kumar et al. (2018) choose Facebook-pages covering topics that similarly coincide with abusive language. The resulting datasets are far from representing a natural sample of the underlying social-media sites.

Table 1 shows that datasets that apply biased sampling (*Warner*, *Waseem*, *Kumar*) contain a high degree of implicit abuse. Boosted random sampling, which provides a more realistic cross section of microposts, on the other hand, captures a larger amount of explicit abuse. Future work should explore whether this is due to the predominance of explicit abuse on social media or some other reason, for example, the fact that human annotators more readily detect explicit abuse.

Intuitively, one would expect that the lower the proportion of explicit abuse is on the set of abusive microposts of a dataset, the lower the F1-score becomes because implicit abuse is not conveyed by lexical cues that are easy to learn. Table 1 confirms this notion, yet *Waseem* is the notable exception. We need to find an explanation for this deviation since *Waseem* is by far the most frequently used dataset for detecting abusive language (Badjatiya et al., 2017; Bourgonje et al., 2017; Pitsilis et al., 2018; Agrawal and Awekar, 2018; Karan and Snajder, 2018; Kshirsagar et al.,

---

[2] For example, Waseem and Hovy (2016) distinguish between *sexism* and *racism*.

[3] More involved classifiers achieve better performance, however, the relative differences between the datasets remain.

[4] www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

name	publication	source	microposts	%abusive	sampling	%explicit*	F1
Kaggle†	(Wulczyn et al., 2017)	Wikipedia	312,737	9.6	boosted random sampling	76.9	88.2
Founta	(Founta et al., 2018)	Twitter	59,357	14.1	boosted random sampling	75.9	87.3
Razavi	(Razavi et al., 2010)	diverse	1,525	31.9	boosted random sampling	64.7	83.3
Warner	(Warner and Hirschberg, 2012)	diverse	3,438	14.3	biased sampling	51.3	71.8
Waseem	(Waseem and Hovy, 2016)	Twitter	16,165	35.3	biased sampling	44.4	**80.5**
Kumar	(Kumar et al., 2018)	Facebook	15,000	58.1	biased sampling	32.7	70.4

Table 1: Properties of the different datasets. (**: proportion of explicitly abusive microposts among abusive microposts.* †*: This is an extension of the dataset presented in* Wulczyn et al. (2017)*. Details on the corpus creation about* Kaggle *can therefore be found in that publication.*)

2018; Mishra et al., 2018a,b; Park et al., 2018; Qian et al., 2018; Sahlgren et al., 2018; Sharifirad et al., 2018; Unsvåg and Gambäck, 2018; Wiegand et al., 2018a). This investigation is only possible since, fortunately, *Waseem* is one of the datasets whose creation process has been meticulously documented.

4 Topic Bias

The *Waseem*-dataset has been sampled in such a way that it contains a high proportion of microposts discussing the role of women in sports, particularly their suitability as football commentators. Such microposts also very often coincide with sexist remarks. However, the authors did not make any attempt to *debias* their dataset. As a consequence, domain-specific expressions such as *announcer*, *commentator*, *football* or *sport* occur very frequently and almost exclusively in abusive tweets. Yet intuitively these words should not be representative of abusive language. There are many texts on the web including Twitter that contain mentions of these expressions but that are not abusive. The current dataset, however, does not reflect that.

Table 2 illustrates this bias by listing the words with the highest Pointwise Mutual Information (PMI) towards abusive microposts. It compares the *Founta*-dataset (a dataset representing almost random sampling) with the *Waseem*-dataset (a dataset produced by biased sampling). We deliberately chose two datasets sampled from the *same* social-media site, namely Twitter, as otherwise the difference we report could be ascribed to differences in the underlying text sources. Table 2 shows that on the *Founta*-dataset, abusive words occupy the high ranks. Most of the highly ranked words of the *Waseem*-dataset, however, are not abusive. Similar observations can be made on the other datasets produced by biased sampling (i.e. *Warner* and *Kumar*). In the *Warner*-dataset, the words *CBS* and *Hollywood* are two of the most predictive words. They refer to the anti-semitic prejudice that Jews are supposed to control most of the US media. On that dataset, the bias of identity terms is also extreme: Almost 80% of the 256 mentions of the identity term *Jew* occur in abusive microposts. On the *Kumar*-dataset, even common Arabic person names, such as *Azan* or *Nahid*, strongly correlate with abusive language.

rank	Founta	Waseem
1	bitch	**commentator**
2	niggas	comedian
3	motherfucker	**football**
4	fucking	**announcer**
5	nigga	pedophile
6	idiot	mankind
7	asshole	sexist
8	fuck	**sport**
9	fuckin	outlaw
10	pussy	driver

Table 2: Top 10 words having strongest correlation with abusive microposts according to PMI on *Founta* (dataset representing almost random sample) and *Waseem* (dataset produced by biased sampling).

Feature Set	Prec	Rec	F1
all words	80.91	80.08	80.49
(ii) query words removed	76.65	76.02	76.33
(i) topic words removed	75.07	74.41	74.72

Table 3: Impact of removing specific words from classifier trained and tested on *Waseem*.

In order to demonstrate the detrimental effects such biases have, we now report the performance of further classifiers trained on the *Wassem*-dataset. Similar results could be obtained on the *Warner*- and *Kumar*-dataset. Yet they are most pronounced on the *Waseem*-dataset, which is also the dataset on which unexpectedly high classification performance has been observed in Table 1. Presumably, it is also the most biased dataset.

In our first experiment, we tested a FastText-classifier (§3) trained on the *Waseem*-dataset on a random sample of 500 additional tweets that include mentions of the topic words *football* and

racist		sexist	
author name	**freq**	**author name**	**freq**
Vile_Islam	1915	Yes You're Sexist	1320
Yes You're Sexist	8	Male Tears #4648	948
Standing Up 4 Trump	5	Vile_Islam	50
YESMarriageEquality	1	LilBeasy91	10
LilBeasy91	1	N!ck	9

Table 4: The 5 most sexist and racist authors on the *Waseem*-dataset and the number of their microposts.

sport. One would expect a low proportion of these particular tweets to be predicted as abusive. However, due to the fact that the abusive training data have such a large topic bias towards sports, the proportion of tweets predicted to be abusive is unreasonably high (i.e. 70%). Manual inspection confirmed that only a small proportion (up to 5%) was actually abusive. This result shows us that classifiers trained on the *Waseem*-dataset hardly generalize to the concept of abusive language. Difficult tweets on that dataset, e.g. instances of implicit abuse, may be classified correctly just because biased words such as *football* or *sport* occur in them.

In our second experiment, we train and test a classifier on the original *Waseem*-dataset in 10-fold crossvalidation. However, we remove either of the two types of biased words from the dataset:

(i) We remove 25 topic words from the 100 most correlating words that we thought bear no relation towards abusive language (e.g. *announcer*, *commentator*, *football* or *sport*).

(ii) We remove the 17 words that were used as a query by Waseem and Hovy (2016) to produce the dataset.

With (i) we want to show how good classifiers are that do not have access to biased words. This would be a realistic setting since words, such as *football* or *sport*, only have this bias towards abusive language on the *Waseem*-dataset. Such removal is also necessary since otherwise these biased words cause a huge amount of false positives when testing on other datasets (as shown above).

With (ii) we want to show that query words themselves are biased, too. For example, we observed that the query word *WomenAgainstFeminism* correlates with abusive tweets while *gamergate* correlates with non-abusive tweets. The purpose of query words is to retrieve tweets that address specific topics. The fact that they correlate with the classes of the dataset further proves that the focused sampling process introduces data bias.

The results of these two configurations are displayed in Table 3. It shows that the removal of a very few words (i.e. 0.2% of the overall vocabulary) already causes the classification performance to drop notably. Please note that these experiments do not capture the full impact of the bias in this dataset. That is, there will be more biased words beyond the 25 words we identified on the list of top 100 words ranked according to PMI since the cut-off value of 100 was arbitrarily chosen.

5 Author Bias

Datasets may also be affected by author bias. By that, we mean that information relating to the author of a micropost may artificially boost classification performance. Author information can be explicitly derived from *meta-information* of a micropost, for example, a feature that encodes the user name of a particular tweet that is to be classified. However, even if we do not explicitly encode such information, a (lexical) supervised classifier, such as the FastText-classifier from Table 1, may indirectly be affected by author biases. If the set of tweets belonging to a certain class predominantly comes from the same author, then a supervised classifier may largely pick up the writing style or the topics addressed by that author. Whenever the writing style or those topics are recognized, abusive language is predicted. This may work on a biased dataset but not beyond it.

We found that the distribution of abusive tweets on the *Waseem*-dataset is highly skewed towards 3 different authors as shown in Table 4. More than 70% of the sexist tweets originate from the two authors *Male Tears #4648* and *Yes, They're Sexist*. 99% of the racist tweets originate from a single author (i.e. *Vile_Islam*). If virtually all racist tweets originate from the same author, a classifier just needs to consider tweets from that author and can predict tweets from every other author as non-racist. On this particular dataset, such a strategy leads to good results: Both Qian et al. (2018) and Mishra et al. (2018a) proposed classification approaches that add author information to common text-level features. These approaches were solely evaluated on the *Waseem*-dataset. However, the author distribution on the *Waseem*-dataset does not reflect reality where abusive tweets originate from far more than a very few authors. In reality, we therefore should expect author information to be less predictive.

6 How to Avoid a Biased Evaluation

A possible way to prevent classification scores from looking unreasonably well is by applying cross-domain classification, i.e. testing a classifier on a dataset different from the one it was trained on. The specific biases we pointed out should be primarily restricted to individual datasets and not carry over to other ones. This is illustrated by Table 5. Compared to in-domain classification (Table 1), all classifiers perform worse. So all classifiers seem to be affected by data bias to some degree. Datasets with explicit abuse and less biased sampling (*Kaggle*, *Founta*, *Razavi*) still perform reasonably when trained among each other, i.e. they are not heavily affected, whereas datasets with implicit abuse and biased sampling (*Warner*, *Waseem*, *Kumar*) perform poorly. This time this also includes *Waseem* which implies that the good performance in in-domain classification (Table 1) was indeed caused by data bias.

Of course, cross-domain classification may not always be practical, particularly if a specific subtype of language abuse is studied for which there is only one dataset available. However, even then, simple methods such as computing the words that highly correlate with the different classes on that dataset, similar to what we did in Table 2, may already indicate that there are biases hidden in the dataset. If only a very small amount of biased words is identified, then usually it suffices to manually *debias* the dataset. By that, one understands sampling additional microposts containing the words manually detected to be biased (Dixon et al., 2017; Wiegand et al., 2018b). For example, in the case of the *Waseem*-dataset, randomly sampling additional tweets matching the words *announcer*, *commentator*, *football* or *sport*, would reduce the sexism bias we reported in this paper (simply because random tweets are unlikely to contain sexist remarks unlike the existing tweets from the *Waseem*-dataset).[5] In order to avoid author bias to interfere with classification, one could restrict the number of microposts per author. This would result in a more balanced distribution of microposts per author.

	test					
train.	**Kaggle**	**Founta**	**Razavi**	**Warner**	**Waseem**	**Kumar**
Kag.	N/A	**85.83**	**76.15**	63.91	60.32	62.48
Fou.	**84.70**	N/A	**70.11**	66.12	64.80	61.25
Raz.	**72.15**	**73.34**	N/A	60.22	61.76	60.61
War.	54.79	55.66	49.32	N/A	61.78	52.94
Was.	61.23	60.88	53.00	61.66	N/A	55.20
Kum.	69.31	65.93	62.98	55.74	59.20	N/A

Table 5: Cross-domain classification (*eval.: F1*).

7 Related Work

Previous work already established that identity terms (e.g. *gay*, *Jew* or *woman*) have a bias to co-occur with abusive language (Dixon et al., 2017; Park et al., 2018). In this work, we showed that this problem is not restricted to the small set of identity terms. Most biases are introduced by the sampling method used on a dataset and they have a huge impact on classification performance.

8 Conclusion

We examined the impact of data bias on abusive language detection and showed that this problem is closely related to how data have been sampled. On the popular *Waseem*-dataset, we illustrated that under more realistic settings, where such biases would be less prominent, classification performance is much lower than reported in research publications. Currently, datasets with a higher degree of implicit abuse are more affected by data bias. Such bias often goes unnoticed in in-domain classification which is why we recommend cross-domain classification. Our finding that under a realistic evaluation classification performance is actually quite poor particularly on implicit abuse, is also in line with assessments from industry on the quality of the state of the art[6] which suggests that there is still a long way to go.

Acknowledgements

The authors were partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

References

Sweeta Agrawal and Amit Awekar. 2018. Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms. In *Proceedings of the European Conference in Information Retrieval (ECIR)*. Grenoble, France, pages 141–153.

---

[5] Please note, however, that in the case of the *Waseem*-dataset, this form of debiasing would not completely solve the data bias since this dataset contains biased words beyond the four words mentioned above.

[6] https://www.businessinsider.de/facebook-ceo-zuckerberg-says-hate-speech-stumps-ai-2018-4

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the International Conference on World Wide Web (WWW)*. Perth, Australia, pages 759–760.

Peter Bourgonje, Julian Moreno-Schneider, Ankit Srivastava, and Georg Rehm. 2017. Automatic Classification of Abusive Language and Personal Attacks in Various Forms of Online Communication. In *Proceedings of the German Society for Computational Linguistics and Language Technology (GSCL)*. Springer, Berlin, Germany, LNAI, pages 180–191.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2017. Measuring and Mitigating Unintended Bias in Text Classification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. San Francisco, CA, USA.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behaviour. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*. Stanford, CA, USA.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 427–431.

Mladen Karan and Jan Snajder. 2018. Cross-Domain Detection of Abusive Language Online. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 132–137.

Rohan Kshirsagar, Tyrus Cukuvac, Kathleeen McKeown, and Susan McGregor. 2018. Predictive Embeddings for Hate Speech Detection on Twitter. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 26–32.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the Workshop on Trolling, Aggression and Cyberbullying (TRAC)*. Santa Fe, NM, USA, pages 1–11.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018a. Author Profiling for Abuse Detection. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. Santa Fe, NM, USA, pages 1088–1098.

Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2018b. Neural Character-based Composition Models for Abuse Detectiion. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 1–10.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing Gender Bias in Abusive Language Detection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Brussels, Belgium, pages 2799–2804.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. In *arXiv preprint arXiv: 1801.04433*.

Jing Qian, Mai ElSherief, Elizabeth M. Belding, and William Yang Wang. 2018. Leveraging intra-user and inter-user representation learning for automated hate speech detection. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. New Orleans, LA, USA, pages 118–123.

Amir Hossein Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive Language Detection Using Multi-level Classification. In *Proceedings of the Canadian Conference on Artificial Intelligence*. Ottawa, Canada, pages 16–27.

Magnus Sahlgren, Tim Isbister, and Fredrik Olsson. 2018. Learning Representations for Detecting Abusive Language. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 115–123.

Sima Sharifirad, Borna Jafarpour, and Stan Matwin. 2018. Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 107–101.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The Effects of User Features on Twitter Hate Speech Detection. In *Proceedings of the Workshop on Abusive Language Online (ALW)*. Brussels, Belgium, pages 75–85.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Workshop on Language in Social Media (LSM)*. Montréal, Canada, pages 19–26.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the ACL-Workshop on Abusive Language Online*. Vancouver, BC, Canada, pages 78–84.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL – Student Research Workshop)*. San Diego, CA, USA, pages 88–93.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018a. Inducing a Lexicon of Abusive Words – A Feature-Based Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*. New Orleans, LA, USA, pages 1046–1056.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018b. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval Workshop*. Vienna, Austria, pages 1–10.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the International Conference on World Wide Web (WWW)*. Perth, Australia, pages 1391–1399.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J. Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. New York City, NY, USA, pages 3952–3958.

Lipstick on a Pig:
Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them

Hila Gonen[1] and **Yoav Goldberg**[1,2]
[1]Department of Computer Science, Bar-Ilan University
[2]Allen Institute for Artificial Intelligence
`{hilagnn,yoav.goldberg}@gmail.com`

Abstract

Word embeddings are widely used in NLP for a vast range of tasks. It was shown that word embeddings derived from text corpora reflect gender biases in society. This phenomenon is pervasive and consistent across different word embedding models, causing serious concern. Several recent works tackle this problem, and propose methods for significantly reducing this gender bias in word embeddings, demonstrating convincing results. However, we argue that this removal is superficial. While the bias is indeed substantially reduced according to the provided bias definition, the actual effect is mostly hiding the bias, not removing it. The gender bias information is still reflected in the distances between "gender-neutralized" words in the debiased embeddings, and can be recovered from them. We present a series of experiments to support this claim, for two debiasing methods. We conclude that existing bias removal techniques are insufficient, and should not be trusted for providing gender-neutral modeling.

1 Introduction

Word embeddings have become an important component in many NLP models and are widely used for a vast range of downstream tasks. However, these word representations have been proven to reflect social biases (e.g. race and gender) that naturally occur in the data used to train them (Caliskan et al., 2017; Garg et al., 2018).

In this paper we focus on gender bias. Gender bias was demonstrated to be consistent and pervasive across different word embeddings. Bolukbasi et al. (2016b) show that using word embeddings for simple analogies surfaces many gender stereotypes. For example, the word embedding they use (word2vec embedding trained on the Google News dataset[1] (Mikolov et al., 2013)) answer the analogy "man is to computer programmer as woman is to x" with "x = homemaker". Caliskan et al. (2017) further demonstrate association between female/male names and groups of words stereotypically assigned to females/males (e.g. arts vs. science). In addition, they demonstrate that word embeddings reflect actual gender gaps in reality by showing the correlation between the gender association of occupation words and labor-force participation data.

Recently, some work has been done to reduce the gender bias in word embeddings, both as a post-processing step (Bolukbasi et al., 2016b) and as part of the training procedure (Zhao et al., 2018). Both works substantially reduce the bias with respect to the same definition: the projection on the gender direction (i.e. $\overrightarrow{he} - \overrightarrow{she}$), introduced in the former. They also show that performance on word similarity tasks is not hurt.

We argue that current debiasing methods, which lean on the above definition for gender bias and directly target it, are mostly hiding the bias rather than removing it. We show that even when drastically reducing the gender bias according to this definition, it is still reflected in the geometry of the representation of "gender-neutral" words, and a lot of the bias information can be recovered.[2]

2 Gender Bias in Word Embeddings

In what follows we refer to words and their vectors interchangeably.

Definition and Existing Debiasing Methods Bolukbasi et al. (2016b) define the gender bias of a word w by its projection on the "gender direction": $\overrightarrow{w} \cdot (\overrightarrow{he} - \overrightarrow{she})$, assuming all vectors are normalized. The larger a word's projection is on

---

[1] `https://code.google.com/archive/p/word2vec/`

[2] The code for our experiments is available at `https://github.com/gonenhila/gender_bias_lipstick`.

Proceedings of NAACL-HLT 2019, pages 609–614
Minneapolis, Minnesota, June 2 - June 7, 2019.

$\overrightarrow{he} - \overrightarrow{she}$, the more biased it is. They also quantify the bias in word embeddings using this definition and show it aligns well with social stereotypes.

Both Bolukbasi et al. (2016b) and Zhao et al. (2018) propose methods for debiasing word embeddings, substantially reducing the bias according to the suggested definition.[3]

In a seminal work, Bolukbasi et al. (2016b) use a post-processing debiasing method. Given a word embedding matrix, they make changes to the word vectors in order to reduce the gender bias as much as possible for all words that are not inherently gendered (e.g. mother, brother, queen). They do that by zeroing the gender projection of each word on a predefined gender direction.[4] In addition, they also take dozens of inherently gendered word pairs and explicitly make sure that all neutral words (those that are not predefined as inherently gendered) are equally close to each of the two words. This extensive, thoughtful, rigorous and well executed work surfaced the problem of bias in embeddings to the ML and NLP communities, defined the concept of debiasing word embeddings, and established the defacto metric of measuring this bias (the gender direction). It also provides a perfect solution to the problem of removing the gender direction from non-gendered words. However, as we show in this work, while the gender-direction is a great indicator of bias, it is only an indicator and not the complete manifestation of this bias.

Zhao et al. (2018) take a different approach and suggest to train debiased word embeddings from scratch. Instead of debiasing existing word vectors, they alter the loss of the GloVe model (Pennington et al., 2014), aiming to concentrate most of the gender information in the last coordinate of each vector. This way, one can later use the word representations excluding the gender coordinate. They do that by using two groups of male/female seed words, and encouraging words that belong to different groups to differ in their last coordinate. In addition, they encourage the representation of neutral-gender words (excluding the last coordinate) to be orthogonal to the gender direction.[5] This work did a step forward by trying to remove the bias during training rather than in post-processing, which we believe to be the right approach. Unfortunately, it relies on the same definition that we show is insufficient.

These works implicitly define what is good gender debiasing: according to Bolukbasi et al. (2016b), there is no gender bias if each non-explicitly gendered word in the vocabulary is in equal distance to both elements of all explicitly gendered pairs. In other words, if one cannot determine the gender association of a word by looking at its projection on any gendered pair. In Zhao et al. (2018) the definition is similar, but restricted to projections on the gender-direction.

Remaining bias after using debiasing methods Both works provide very compelling results as evidence of reducing the bias without hurting the performance of the embeddings for standard tasks.

However, both methods and their results rely on the specific bias definition. We claim that the bias is much more profound and systematic, and that simply reducing the projection of words on a gender direction is insufficient: it merely hides the bias, which is still reflected in similarities between "gender-neutral" words (i.e., words such as "math" or "delicate" are in principle gender-neutral, but in practice have strong stereotypical gender associations, which reflect on, and are reflected by, neighbouring words).

Our key observation is that, almost by definition, most word pairs maintain their previous similarity, despite their change in relation to the gender direction. The implication of this is that most words that had a specific bias before are still grouped together, and apart from changes with respect to specific gendered words, the word embeddings' spatial geometry stays largely the same.[6] In what follows, we provide a series of experiments that demonstrate the remaining bias in the debiased embeddings.

---

[3] Another work in this spirit is that of Zhang et al. (2018), which uses an adversarial network to debias word embeddings. There, the authors rely on the same definition of gender bias that considers the projection on the gender direction. We expect similar results for this method as well, however, we did not verify that.

[4] The gender direction is chosen to be the top principal component (PC) of ten gender pair difference vectors.

[5] The gender direction is estimated during training by averaging the differences between female words and their male counterparts in a predefined set.

[6] We note that in the extended arxiv version, Bolukbasi et al. (2016a) do mention this phenomenon and refer to it as "indirect bias". However, they do not quantify its extensiveness before and after debiasing, treat it mostly as a nuance, and do not provide any methods to deal with it.

3 Experimental Setup

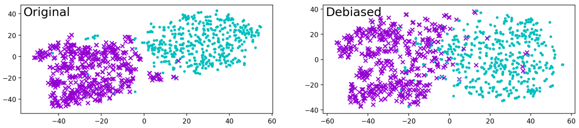
We refer to the word embeddings of the previous works as Hard-Debiased (Bolukbasi et al., 2016b) and GN-GloVe (gender-neutral GloVe) (Zhao et al., 2018). For each debiased word embedding we quantify the hidden bias with respect to the biased version. For Hard-Debiased we compare to the embeddings before applying the debiasing procedure. For GN-GloVe we compare to embedding trained with standard GloVe on the same corpus.[7]

Unless otherwise specified, we follow Bolukbasi et al. (2016b) and use a reduced version of the vocabulary for both word embeddings: we take the most frequent 50,000 words and phrases and remove words with upper-case letters, digits, or punctuation, and words longer than 20 characters. In addition, to avoid quantifying the bias of words that are inherently gendered (e.g. mother, father, queen), we remove from each vocabulary the respective set of gendered words as pre-defined in each work.[8] This yeilds a vocabulary of 26,189 words for Hard-Debiased and of 47,698 words for GN-GloVe.

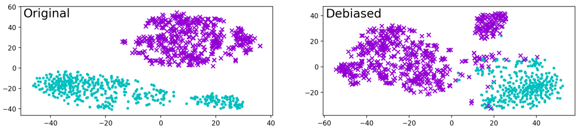
As explained in Section 2 and according to the definition in previous works, we compute the bias of a word by taking its projection on the gender direction: $\overrightarrow{he} - \overrightarrow{she}$.

In order to quantify the association between sets of words, we follow Caliskan et al. (2017) and use their Word Embedding Association Test (WEAT): consider two sets of target words (e.g., male and female professions) and two sets of attribute words (e.g., male and female names). A permutation test estimates the probability that a random permutation of the target words would produce equal or greater similarities to the attribute sets.

4 Experiments and Results



(a) Clustering for Hard-Debiased embedding, before (left hand-side) and after (right hand-side) debiasing.



(b) Clustering for GN-GloVe embedding, before (left hand-side) and after (right hand-side) debiasing.

Figure 1: Clustering the 1,000 most biased words, before and after debiasing, for both models.

Male- and female-biased words cluster together We take the most biased words in the vocabulary according to the original bias (500 male-biased and 500 female-biased[9]), and cluster them into two clusters using k-means. For the Hard-Debiased embedding, the clusters align with gender with an accuracy of 92.5% (according to the original bias of each word), compared to an accuracy of 99.9% with the original biased version. For the GN-GloVe embedding, we get an accuracy of 85.6%, compared to an accuracy of 100% with the biased version. These results suggest that indeed much of the bias information is still embedded in the representation after debiasing. Figure 1 shows the tSNE (Maaten and Hinton, 2008) projection of the vectors before and after debiasing, for both models.

Bias-by-projection correlates to bias-by-neighbours This clustering of gendered words indicates that while we cannot directly "observe" the bias (i.e. the word "nurse" will no longer be closer to explicitly marked feminine words) the bias is still manifested by the word being close to *socially-marked* feminine words, for example "nurse" being close to "receptionist", "caregiver" and "teacher". This suggests a new mechanism for measuring bias: the percentage of male/female socially-biased words among the k nearest neighbors of the target word.[10]

We measure the correlation of this new bias

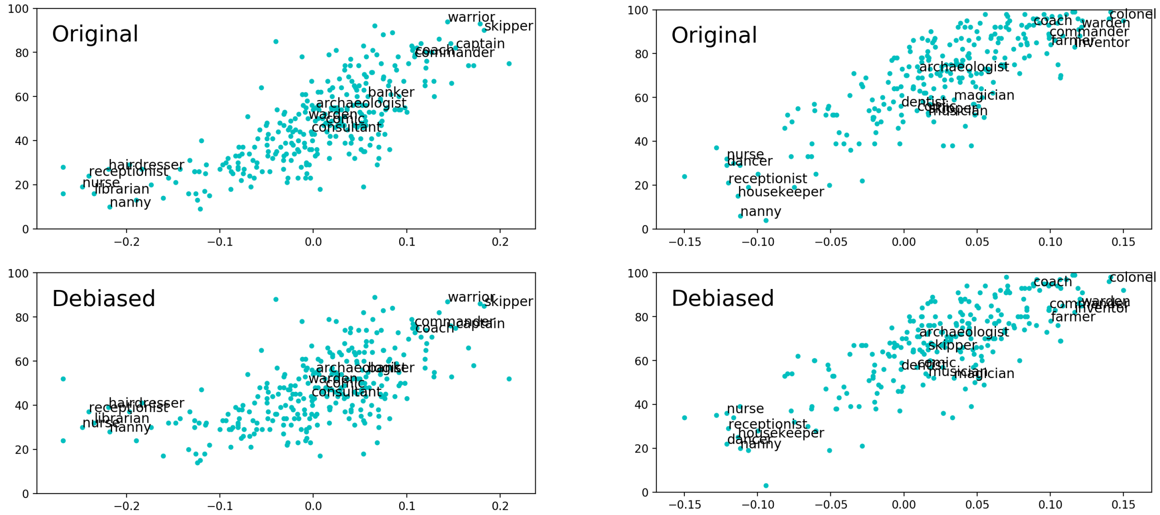
---

[7] We use the embeddings provided by Bolukbasi et al. (2016b) in https://github.com/tolga-b/debiaswe and by Zhao et al. (2018) in https://github.com/uclanlp/gn_glove.

[8] For Hard-Debiased we use first three lists from: https://github.com/tolga-b/debiaswe/tree/master/data and for GN-GloVe we use the two lists from: https://github.com/uclanlp/gn_glove/tree/master/wordlist

[9] highest on the two lists for Hard-Debiased are 'petite', 'mums', 'bra', 'breastfeeding' and 'sassy' for female and 'rookie', 'burly', 'hero', 'training_camp' and 'journeyman' for male. Lowest on the two lists are 'watchdogs', 'watercolors', 'sew', 'burqa', 'diets' for female and 'teammates', 'playable', 'grinning', 'knee_surgery', 'impersonation' for male.

[10] While the social bias associated with a word cannot be observed directly in the new embeddings, we can approximate it using the gender-direction in non-debiased embeddings.



(a) The plots for HARD-DEBIASED embedding, before (top) and after (bottom) debiasing.

(b) The plots for GN-GLOVE embedding, before (top) and after (bottom) debiasing.

Figure 2: The number of male neighbors for each profession as a function of its original bias, before and after debiasing. We show only a limited number of professions on the plot to make it readable.

measure with the original bias measure. For the HARD-DEBIASED embedding we get a Pearson correlation of 0.686 (compared to a correlation of 0.741 when checking neighbors according to the biased version). For the GN-GLOVE embedding we get a Pearson correlation of 0.736 (compared to 0.773). All these correlations are statistically significant with p-values of 0.

Professions We consider the list of professions used in Bolukbasi et al. (2016b) and Zhao et al. (2018)[11] in light of the neighbours-based bias definition. Figure 2 plots the professions, with axis X being the original bias and axis Y being the number of male neighbors, before and after debiasing. For both methods, there is a clear correlation between the two variables.

We observe a Pearson correlation of 0.606 (compared to a correlation of 0.747 when checking neighbors according to the biased version) for HARD-DEBIASED and 0.792 (compared to 0.820) for GN-GLOVE. All these correlations are significant with p-values $< 1 \times 10^{-30}$.

Association between female/male and female/male-stereotyped words We replicate the three gender-related association experiments from Caliskan et al. (2017). For these experiments we use the full vocabulary since some of the words are not included in the reduced one.

The first experiment evaluates the association between female/male names and family and career words. The second one evaluates the association between female/male concepts and arts and mathematics words. Since the inherently gendered words (e.g. girl, her, brother) in the second experiment are handled well by the debiasing models we opt to use female and male names instead. The third one evaluates the association between female/male concepts and arts and science words. Again, we use female and male names instead.[12]

For the HARD-DEBIASED embedding, we get a p-value of 0 for the first experiment, 0.00016 for the second one, and 0.0467 for the third. For the GN-GLOVE embedding, we get p-values of 7.7×10^{-5}, 0.00031 and 0.0064 for the first, second and third experiments, respectively.

Classifying previously female- and male-biased words Can a classifier learn to generalize from some gendered words to others based only on their

---

[11] `https://github.com/tolga-b/debiaswe/tree/master/data/professions.json`

[12] All word lists are taken from Caliskan et al. (2017): **First experiment:** *Female names*: Amy, Joan, Lisa, Sarah, Diana, Kate, Ann, Donna. *Male names*: John, Paul, Mike, Kevin, Steve, Greg, Jeff, Bill. *Family words*: home, parents, children, family, cousins, marriage, wedding, relatives. *Career words*: executive, management, professional, corporation, salary, office, business, career. **Second experiment:** *Arts Words*: poetry, art, dance, literature, novel, symphony, drama, sculpture. *Math words*: math, algebra, geometry, calculus, equations, computation, numbers, addition. **Third experiment:** *Arts words*: poetry, art, Shakespeare, dance, literature, novel, symphony, drama. *Science words*: science, technology, physics, chemistry, Einstein, NASA, experiment, astronomy.

representations? We consider the 5,000 most biased words according to the original bias (2,500 from each gender), train an RBF-kernel SVM classifier on a random sample of 1,000 of them (500 from each gender) to predict the gender, and evaluate its generalization on the remaining 4,000. For the HARD-DEBIASED embedding, we get an accuracy of 88.88%, compared to an accuracy of 98.25% with the non-debiased version. For the GN-GLOVE embedding, we get an accuracy of 96.53%, compared to an accuracy of 98.65% with the non-debiased version.

5 Discussion and Conclusion

The experiments described in the previous section reveal a systematic bias found in the embeddings, which is independent of the gender direction. We observe that semantically related words still maintain gender bias both in their similarities, and in their representation. Concretely, we find that:

1. Words with strong previous gender bias (with the same direction) are easy to cluster together.

2. Words that receive implicit gender from social stereotypes (e.g. receptionist, hairdresser, captain) still tend to group with other implicit-gender words of the same gender, similar as for non-debiased word embeddings.

3. The implicit gender of words with prevalent previous bias is easy to predict based on their vectors alone.

The implications are alarming: while suggested debiasing methods work well at removing the gender direction, the debiasing is mostly superficial. The bias stemming from world stereotypes and learned from the corpus is ingrained much more deeply in the embeddings space.

We note that the real concern from biased representations is not the association of a concept with words such as “he”, “she”, “boy”, “girl” nor being able to perform gender-stereotypical word analogies. While these are nice “party tricks”, algorithmic discrimination is more likely to happen by associating one implicitly gendered term with other implicitly gendered terms, or picking up on gender-specific regularities in the corpus by learning to condition on gender-biased words, and generalizing to other gender-biased words (i.e., a resume classifier that will learn to favor male over female candidates based on stereotypical cues in an existing—and biased—resume dataset, despite of being “oblivious” to gender). Our experiments show that such classifiers would have ample opportunities to pick up on such cues also after debiasing w.r.t the gender-direction.

The crux of the issue is that the gender-direction provides a way to *measure* the gender-association of a word, but *does not determine* it. Debiasing methods which directly target the gender-direction are for the most part merely hiding the gender bias and not removing it. The popular definitions used for quantifying and removing bias are insufficient, and other aspects of the bias should be taken into consideration as well.

Acknowledgments

This work is supported by the Israeli Science Foundation (grant number 1555/15), and by the Israeli ministry of Science, Technology and Space through the Israeli-French Maimonide Cooperation program.

References

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016a. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *arXiv:1607.06520*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016b. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. In *Proceedings of EMNLP*, pages 4847–4853.

Black is to Criminal as Caucasian is to Police: Detecting and Removing Multiclass Bias in Word Embeddings

Thomas Manzini[†]* , **Yao Chong Lim**[‡]* , **Yulia Tsvetkov**[‡], **Alan W Black**[‡]
Microsoft AI Development Acceleration Program[†], Carnegie Mellon University[‡]
`Thomas.Manzini@microsoft.com, {yaochonl,ytsvetko,awb}@cs.cmu.edu`

Abstract

Online texts—across genres, registers, domains, and styles—are riddled with human stereotypes, expressed in overt or subtle ways. Word embeddings, trained on these texts, perpetuate and amplify these stereotypes, and propagate biases to machine learning models that use word embeddings as features. In this work, we propose a method to debias word embeddings in multiclass settings such as race and religion, extending the work of (Bolukbasi et al., 2016) from the binary setting, such as binary gender. Next, we propose a novel methodology for the evaluation of multiclass debiasing. We demonstrate that our multiclass debiasing is robust and maintains the efficacy in standard NLP tasks.

Gender Biased Analogies	
man → doctor woman → receptionist woman → secretary	woman → nurse man → supervisor man → principal
Racially Biased Analogies	
black → criminal asian → doctor caucasian → leader	caucasian → police caucasian → dad black → led
Religiously Biased Analogies	
muslim → terrorist jewish → philanthropist christian → unemployed	christian → civilians christian → stooge jewish → pensioners

Table 1: Examples of gender, racial, and religious biases in analogies generated from word embeddings trained on the Reddit data from users from the USA.

1 Introduction

In addition to possessing informative features useful for a variety of NLP tasks, word embeddings reflect and propagate social biases present in training corpora (Caliskan et al., 2017; Garg et al., 2018). Machine learning systems that use embeddings can further amplify biases (Barocas and Selbst, 2016; Zhao et al., 2017), discriminating against users, particularly those from disadvantaged social groups.

(Bolukbasi et al., 2016) introduced a method to *debias* embeddings by removing components that lie in stereotype-related embedding subspaces. They demonstrate the effectiveness of the approach by removing gender bias from word2vec embeddings (Mikolov et al., 2013), preserving the utility of embeddings and potentially alleviating biases in downstream tasks. However, this method was only for *binary* labels (e.g., male/female), whereas most real-world demographic attributes, including gender, race, religion, are not binary but continuous or categorical, with more than two categories.

In this work, we show a generalization of Bolukbasi et al.'s (2016) which enables *multiclass* debiasing, while preserving utility of embeddings (§3). We train word2vec embeddings using the Reddit L2 corpus (Rabinovich et al., 2018) and apply multiclass debiasing using lexicons from studies on bias in NLP and social science (§4.2). We introduce a novel metric for evaluation of bias in collections of word embeddings (§5). Finally, we validate that the utility of debiased embeddings in the tasks of part-of-speech (POS) tagging, named entity recognition (NER), and POS chunking is on par with off-the-shelf embeddings.

2 Background

As defined by (Bolukbasi et al., 2016), debiasing word embeddings in a binary setting requires identifying the bias subspace of the embeddings. Components lying in that subspace are then removed

---

* Equal contributions

† Work done while at CMU and The Microsoft AI Development Acceleration Program

Proceedings of NAACL-HLT 2019, pages 615–621
Minneapolis, Minnesota, June 2 - June 7, 2019.

from each embedding.

2.1 Identifying the bias subspace

(Bolukbasi et al., 2016) define the gender subspace using *defining sets* of words, where the words in each set represent different ends of the bias. For example, in the case of gender, one defining set might be the gendered pronouns {*he*, *she*} and another set might be the gendered nouns {*man*, *woman*}. The gender subspace is then computed from these defining sets by 1) computing the vector differences of the word embeddings of words in each set from the set's mean, and 2) taking the most significant components of these vectors.

2.2 Removing bias components

Following the identification of the gender subspace, one can apply hard or soft debiasing (Bolukbasi et al., 2016) to completely or partially remove the subspace components from the embeddings.

Hard debiasing

Hard debiasing (also called "Neutralize and Equalize") involves two steps. First, bias components are removed from words that are not gendered and should not contain gender bias (e.g., *doctor*, *nurse*), and second, gendered word embeddings are centered and their bias components are equalized. For example, in the binary case, *man* and *woman* should have bias components in opposite directions, but of the same magnitude. Intuitively, this then ensures that any neutral words are equidistant to any biased words with respect to the bias subspace.

More formally, to *neutralize*, given a bias subspace $\mathcal{B}$ spanned by the vectors $\{\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_k}\}$, we compute the component of each embedding in this subspace:

$$\mathbf{w}_{\mathcal{B}} = \sum_{i=1}^{k} \langle \mathbf{w}, \boldsymbol{b_i} \rangle \boldsymbol{b_i} \tag{1}$$

We then remove this component from words that should be bias-neutral and normalize to get the debiased embedding:

$$\mathbf{w}' = \frac{\mathbf{w} - \mathbf{w}_{\mathcal{B}}}{\|\mathbf{w} - \mathbf{w}_{\mathcal{B}}\|} \tag{2}$$

To *equalize* the embeddings of words in an equality set E, let $\boldsymbol{\mu} = \frac{1}{|E|} \sum_{\mathbf{w} \in E} \mathbf{w}$ be the mean embedding of the words in the set and $\boldsymbol{\mu}_{\mathcal{B}}$ be its component in the bias subspace as calculated in Equation 1. Then, for $\mathbf{w} \in E$,

$$\mathbf{w}' = (\boldsymbol{\mu} - \boldsymbol{\mu}_{\mathcal{B}}) + \sqrt{1 - \|\boldsymbol{\mu} - \boldsymbol{\mu}_{\mathcal{B}}\|^2} \frac{\mathbf{w}_{\mathcal{B}} - \boldsymbol{\mu}_{\mathcal{B}}}{\|\mathbf{w}_{\mathcal{B}} - \boldsymbol{\mu}_{\mathcal{B}}\|} \tag{3}$$

Note that in both Equations 2 and 3, the new embedding has unit length.

Soft debiasing

Soft debiasing involves learning a projection of the embedding matrix that preserves the inner product between biased and debiased embeddings while minimizing the projection onto the bias subspace of embeddings that should be neutral.

Given embeddings $\mathbf{W}$ and $\mathbf{N}$ which are embeddings for the whole vocabulary and the subset of bias-neutral words respectively, and the bias subspace $\mathcal{B}$ obtained in Section 2.1, soft debiasing seeks for a linear transformation A that minimizes the following objective:

$$\begin{aligned} &\|(\mathbf{AW})^{\intercal}(\mathbf{AW}) - \mathbf{W}^{\intercal}\mathbf{W}\|_F^2 \\ &\quad + \lambda \|(\mathbf{AN})^{\intercal}(\mathbf{A}\mathcal{B})\|_F^2 \end{aligned} \tag{4}$$

Minimizing the first term preserves the inner product after the linear transformation $\mathbf{A}$, and minimizing the second term minimizes the projection onto the bias subspace $\mathcal{B}$ of embeddings. $\lambda \in \mathbb{R}$ is a tunable parameter that balances the two objectives.

3 Methodology

We now discuss our proposed extension of word embedding debiasing to the multiclass setting.

3.1 Debiasing

As in the binary setting, debiasing consists of two steps: identifying the "bias subspace" and removing this component from the set of embeddings.

Identifying the bias subspace

The core contribution of our work is in *identifying* the "bias subspace" in a multiclass setting; if we can identify the bias subspace then prior work can be used for multiclass debiasing.

Past work has shown that it is possible to linearly separate multiple social classes based on components of word embeddings (Garg et al., 2018). Based on this we hypothesize that there exists some component of these embeddings which can capture multiclass bias. While a multiclass

Gender Debiasing	MAC	P-Value
Biased	0.623	N/A
Hard Debiased	1.000	1.582e-14
Soft Debiased ($\lambda = 0.2$)	0.747	1.711e-12
Race Debiasing	MAC	P-Value
Biased	0.892	N/A
Hard Debiased	1.009	7.235e-04
Soft Debiased ($\lambda = 0.2$)	0.985	6.217e-05
Religion Debiasing	MAC	P-Value
Biased	0.859	N/A
Hard Debiased	1.004	3.006e-07
Soft Debiased ($\lambda = 0.2$)	0.894	0.007

Table 2: The associated mean average cosine similarity (MAC) (defined in Section 3.2) and P-Values for debiasing methods for gender, race, and religious bias.

problem is inherently not a linearly separable problem, a one versus rest classifier is. Following from this, the computation of a multiclass bias subspace does not have any linear constraints, though it does come with a loss of resolution. As a result we can compute the principal components required to compute the "bias subspace" by simply adding an additional term for each additional bias class to each defining set.

Formally, given defining sets of word embeddings $D_1, D_2, ..., D_n$, let the mean of the defining set i be $\boldsymbol{\mu}_i = \frac{1}{|D_i|}\sum_{\mathbf{w}\in D_i}\mathbf{w}$, where $\mathbf{w}$ is the word embedding of w. Then the bias subspace $\mathcal{B}$ is given by the first k components of the following principal component analysis (PCA) evaluation:

$$\mathbf{PCA}\left(\bigcup_{i=1}^{n}\bigcup_{\mathbf{w}\in D_i}\mathbf{w}-\boldsymbol{\mu}_i\right) \tag{5}$$

The number of components k can be empirically determined by inspecting the eigenvalues of the PCA, or using a threshold. Also, note that the defining sets do not have to be the same size. We discuss the robustness of this method later.

Removing Bias Components

Following the identification of the bias subspace, we apply the hard Neutralize and Equalize debiasing and soft debiasing method presented in (Bolukbasi et al., 2016) and discussed in Section 2.2 to completely or partially remove the subspace components from the embeddings.

For equalization, we take the defining sets to be the equality sets as well.

3.2 Quantifying Bias Removal

We propose a new metric for the evaluation of bias in collections of words which is simply the mean average cosine similarity (MAC). This approach is motivated by the WEAT evaluation method proposed by (Caliskan et al., 2017) but modified for a multiclass setting. To compute this metric the following data is required: a set of target word embeddings T containing terms that inherently contain some form of social bias (e.g. {*church*, *synagogue*, *mosque*}), and a set A which contains sets of attributes $A_1, A_2, ..., A_N$ containing word embeddings that should not be associated with any word embeddings contained in the set T (e.g. {*violent*, *liberal*, *conservative* }).

We define a function S that computes the mean cosine distance between a particular target T_i and all terms in a particular attribute set A_j:

$$S(\mathbf{t}, A_j) = \frac{1}{N}\sum_{\mathbf{a}\in A_j} cos(\mathbf{t}, \mathbf{a}), \tag{6}$$

where the cosine distance is:

$$cos(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|_2 \cdot \|\mathbf{v}\|_2}. \tag{7}$$

Finally, we define MAC as:

$$\text{MAC}(T, A) = \frac{1}{|T||A|}\sum_{T_i\in T}\sum_{A_j\in A} S(T_i, A_j) \tag{8}$$

We also perform a paired t-test on the distribution of average cosines used to calculate the MAC. Thus we can quantify the effect of debiasing on word embeddings in T and sets in A.

3.3 Measuring Downstream Utility

To measure the utility of the debiased word embeddings, we use the tasks of NER, POS tagging, and POS chunking. This is to ensure that the debiasing procedure has not destroyed the utility of the word embeddings. We evaluate test sentences that contain at least one word affected by debiasing. Additionally, we measure the change in performance after replacing the biased embedding matrix by a debiased one, and retraining the model on debiased embeddings.

4 Data

In this section we discuss the different data sources we used for our initial word embeddings, the social bias data used for evaluating bias, and the linguistic data used for evaluating the debiasing process.

Embedding Matrix Replacement									
	Hard Gender Debiasing			Hard Racial Debiasing			Hard Religious Debiasing		
	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking
Biased F1	0.9954	0.9657	0.9958	0.9948	0.9668	0.9958	0.9971	0.9665	0.9968
Δ F1	+0.0045	-0.0098	+0.0041	+0.0051	-0.0117	+0.0041	+0.0103	-0.0345	+0.0120
Δ Precision	0.0	-0.0177	0.0	0.0	-0.0208	0.0	0.0	-0.0337	0.0
Δ Recall	+0.0165	-0.0208	+0.0156	+0.0186	-0.0250	+0.0155	+0.00286	-0.0174	+0.0031
	Soft Gender Debiasing			Soft Racial Debiasing			Soft Religious Debiasing		
	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking
Biased F1	0.9952	0.9614	0.9950	0.9946	0.9612	0.9946	0.9964	0.9616	0.9961
Δ F1	+0.0047	-0.0102	+0.0049	+0.0053	-0.0107	+0.0053	+0.0128	-0.0242	+0.0148
Δ Precision	0.0	-0.0202	0.0	0.0	-0.0223	0.0	0.0	-0.0199	0.0
Δ Recall	+0.0169	-0.0198	+0.0187	+0.0193	-0.0197	+0.0202	+0.0035	-0.0112	+0.0038
Model Retraining									
	Hard Gender Debiasing			Hard Racial Debiasing			Hard Religious Debiasing		
	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking
Biased F1	0.9954	0.9657	0.9958	0.9948	0.9668	0.9958	0.9971	0.9665	0.9968
Δ F1	+0.0045	-0.0137	+0.0041	+0.0051	-0.0165	+0.0041	+0.0103	-0.0344	+0.0120
Δ Precision	0.0	-0.0259	0.0	0.0	-0.0339	0.0	0.0	-0.0287	0.0
Δ Recall	+0.0165	-0.0278	+0.0156	+0.0186	-0.0306	+0.0156	+0.00286	-0.0161	+0.0031
	Soft Gender Debiasing			Soft Racial Debiasing			Soft Religious Debiasing		
	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking	NER Tagging	POS Tagging	POS Chunking
Biased F1	0.9952	0.9614	0.9950	0.9946	0.9612	0.9946	0.9964	0.9616	0.9961
Δ F1	+0.0047	+0.00178	+0.0049	+0.0053	-0.00119	+0.0053	+0.0128	-0.0098	+0.0148
Δ Precision	0.0	+0.0048	0.0	0.0	-0.00187	0.0	0.0	-0.0125	0.0
Δ Recall	+0.0169	+0.00206	+0.0187	+0.0193	-0.00264	+0.0202	+0.0035	-0.0057	+0.0038

Table 3: The performance of embeddings the downstream tasks of NER, POS Tagging, and POS Chunking.

4.1 Embedding Language Corpus

We used the L2-Reddit corpus (Rabinovich et al., 2018), a collection of Reddit posts and comments by both native and non-native English speakers. The native countries of post authors are determined based on their posts in country- and region-specific subreddits (such as r/Europe and r/UnitedKingdom), and other metadata such as user flairs, which serve as self-identification of the user's country of origin.

In this work, we exclusively explore data collected from the United States. This was done to leverage extensive studies of social bias done in the United States. To obtain the initial biased word embeddings, we trained word2vec embeddings (Mikolov et al., 2013) using approximately 56 million sentences.

4.2 Social Bias Data

We used the following vocabularies and studies to compile lexicons for bias detection and removal.[1]

For gender, we used vocabularies created by (Bolukbasi et al., 2016) and (Caliskan et al., 2017).

For race we consulted a number of different sources for each race: Caucasians (Chung-Herrera and Lankau, 2005; Goad, 1998); African Americans (Punyanunt-Carter, 2008; Brown Givens and Monahan, 2005; Chung-Herrera and Lankau, 2005; Hakanen, 1995; Welch, 2007; Kawai, 2005); and Asian Americans (Leong and Hayes, 1990; Lin et al., 2005; Chung-Herrera and Lankau, 2005; Osajima, 2005; Garg et al., 2018).

Finally, for religion we used the following sources and labels: Christians (Rios et al., 2015; Zuckerman, 2009; Unnever et al., 2005); Jews (Dundes, 1971; Fetzer, 2000); and Muslims (Shryock, 2010; Alsultany, 2012; Shaheen, 1997).

4.3 Downstream Tasks

We evaluate biased and debiased word embeddings on several downstream tasks. Specifically, the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003) which provides evaluation data for NER, POS tagging, and POS chunking.

5 Results and Discussion

In this section we review the results of our experiments and discuss what those results mean in the context of this work.

5.1 Observations of Bias

We can observe bias in word embeddings in many different ways. However, for the purposes of demonstrating that bias exists in these word embeddings we use the analogy task that was used to demonstrate bias in (Bolukbasi et al., 2016). We observe that bias is present in generated analogies by viewing them directly. A small subset of these analogies are in Table 1 to highlight our findings.

---

[1] The lexicon used can be found here: `https://docs.google.com/spreadsheets/d/1BQBFLUvB9bnuifxikjrcJNqLA0dx9ZeAWk4kdu_OCUM`.

5.2 Removal of Bias

We perform our debiasing in the same manner as described in Section 3.1 and calculate the MAC scores and p-values to measure the effects of debiasing. Results are presented in Table 2.

Does multiclass debiasing decrease bias? We see that this debiasing procedure categorically moves MAC scores closer to 1.0. This indicates an increase in cosine distance. Further, the associated P-values indicate these changes are statistically significant. This demonstrates that our approach for multiclass debiasing decreases bias.

5.3 Downstream Effects of Bias Removal

The effects of debiasing on downstream tasks are shown in Table 3. Debiasing can either help or harm performance. For POS tagging there is almost always a decrease in performance. However, for NER and POS chunking, there is a consistent increase. We conclude that these models have learned to depend on some bias subspaces differently. Note that many performance changes are of questionable statistical significance.

Does multiclass debiasing preserve semantic utility? We argue the minor changes in Table 3 support the preservation of semantic utility in the multiclass setting, especially compared to gender debiasing which is known to preserve utility (Bolukbasi et al., 2016).

Is the calculated bias subspace robust? The bias subspace is at least robust enough to support the above debiasing operations. This is shown by statistically significant changes in MAC scores.

6 Limitations & Future Work

Calculating multiclass bias subspace using our proposed approach has drawbacks. For example, in the binary gender case, the extremes of bias subspace reflect extreme male and female terms. However, this is not possible when projecting multiple classes into a linear space. Thus, while we can calculate the magnitude of the bias components, we cannot measure extremes of each class.

Additionally, the methods presented here rely on words that represent biases (defining sets) and words that should or should not contain biases (equality sets). These lists are based on data collected specifically from the US. Thus, they may not translate to other countries or cultures. Further, some of these vocabulary terms, while peer reviewed, may be subjective and may not fully capture the bias subspace.

Recent work by Gonen and Goldberg (2019) suggests that debiasing methods based on bias component removal are insufficient to completely remove bias in the embeddings, since embeddings with similar biases are still clustered together after bias component removal. Following Gonen and Goldberg's (2019) procedure, we plot the number of neighbors of a particular bias class as a function of the original bias, before and after debiasing in Figure 1 and 2 in the Appendix. In line with Gonen and Goldberg's (2019) findings, simply removing the bias component is insufficient to remove multiclass "cluster bias". However, increasing the size of the bias subspace reduces the correlation of the two variables (Table 4 in the Appendix).

7 Conclusion

We showed that word embeddings trained on `www.reddit.com` data contain multiclass biases. We presented a novel metric for evaluating debiasing procedures for word embeddings. We robustly removed multiclass bias using a generalization of existing techniques. Finally, we showed that this multiclass generalization preserves the utility of embeddings for different NLP tasks.

Acknowledgments

This research was supported by Grant No. IIS1812327 from the United States National Science Foundation (NSF). We also acknowledge several people who contributed to this work: Benjamin Pall for his valuable early support of this work; Elise Romberger who helped edit this work prior to its final submission. Finally, we are greatly appreciative of the anonymous reviewers for their time and constructive comments.

References

Evelyn Alsultany. 2012. *Arabs and Muslims in the Media: Race and Representation after 9/11*. nyu Press.

Solon Barocas and Andrew D Selbst. 2016. Big data's disparate impact. *Calif. L. Rev.*, 104:671.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Proc. of NIPS*, pages 4349–4357.

Sonja M Brown Givens and Jennifer L Monahan. 2005. Priming mammies, jezebels, and other controlling images: An examination of the influence of mediated stereotypes on perceptions of an african american woman. *Media Psychology*, 7(1):87–106.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Beth G Chung-Herrera and Melenie J Lankau. 2005. Are we there yet? an assessment of fit between stereotypes of minority managers and the successful-manager prototype. *Journal of Applied Social Psychology*, 35(10):2029–2056.

Alan Dundes. 1971. A study of ethnic slurs: The jew and the polack in the united states. *The Journal of American Folklore*, 84(332):186–203.

Joel S Fetzer. 2000. *Public attitudes toward immigration in the United States, France, and Germany*. Cambridge University Press.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proc. of the National Academy of Sciences*, 115(16).

Jim Goad. 1998. *The Redneck Manifesto: How Hillbillies Hicks and White Trash Becames America's Scapegoats*. Simon and Schuster.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them.

Ernest A Hakanen. 1995. Emotional use of music by african american adolescents. *Howard Journal of Communications*, 5(3):214–222.

Yuko Kawai. 2005. Stereotyping asian americans: The dialectic of the model minority and the yellow peril. *The Howard Journal of Communications*, 16(2):109–130.

Frederick TL Leong and Thomas J Hayes. 1990. Occupational stereotyping of asian americans. *The Career Development Quarterly*, 39(2):143–154.

Monica H Lin, Virginia SY Kwan, Anna Cheung, and Susan T Fiske. 2005. Stereotype content model explains prejudice for an envied outgroup: Scale of anti-asian american stereotypes. *Personality and Social Psychology Bulletin*, 31(1):34–47.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Keith Osajima. 2005. Asian americans as the model minority: An analysis of the popular press image in the 1960s and 1980s. *A companion to Asian American studies*, pages 215–225.

Narissra M Punyanunt-Carter. 2008. The perceived realism of african american portrayals on television. *The Howard Journal of Communications*, 19(3):241–257.

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. In *Proc. of the Transactions of Association for Computational Linguistics*.

Kimberly Rios, Zhen Hadassah Cheng, Rebecca R Totton, and Azim F Shariff. 2015. Negative stereotypes cause christians to underperform in and disidentify with science. *Social Psychological and Personality Science*, 6(8):959–967.

Jack G Shaheen. 1997. *Arab and Muslim stereotyping in American popular culture*. Center for Muslim-Christian Understanding, History and International Affairs .

Andrew Shryock. 2010. *Islamophobia/Islamophilia: Beyond the politics of enemy and friend*. Indiana University Press.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

James D Unnever, Francis T Cullen, and Brandon K Applegate. 2005. Turning the other cheek: Reassessing the impact of religion on punitive ideology. *Justice Quarterly*, 22(3):304–339.

Kelly Welch. 2007. Black criminal stereotypes and racial profiling. *Journal of Contemporary Criminal Justice*, 23(3):276–288.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proc. of EMNLP*, pages 2979–2989.

Phil Zuckerman. 2009. Atheism, secularity, and well-being: How the findings of social science counter negative stereotypes and assumptions. *Sociology Compass*, 3(6):949–971.

A Addressing Cluster Bias

To visualize the degree of cluster bias before and after our debiasing procedure, we follow a similar procedure to Gonen and Goldberg (2019). For a defining set D for the target task (e.g. religion, race, gender), we compute the mean embedding $\boldsymbol{\mu} = \frac{1}{|D|}\sum_{\mathbf{c}\in D}\mathbf{c}$. Then, for each class $\mathbf{c}$ in the defining set, we define the bias direction as

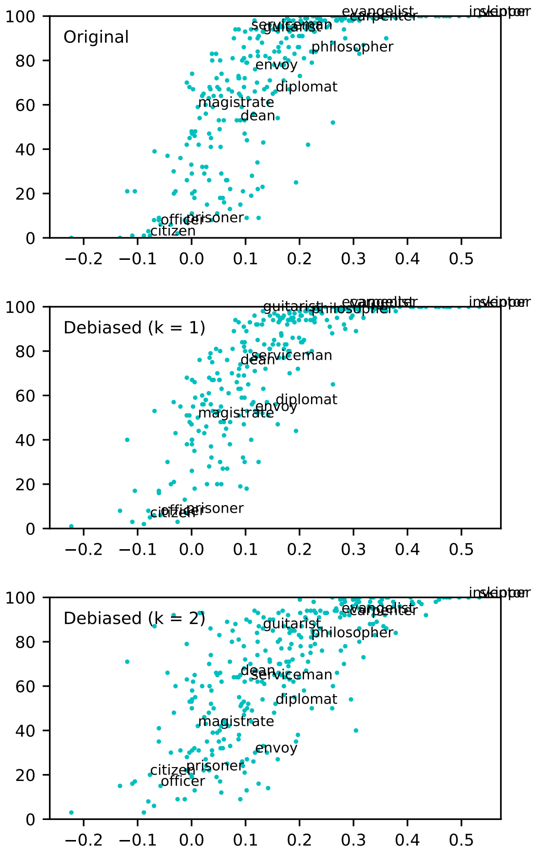


Figure 1: Plots of number of neighbors to *jew* for each profession as a function of its original bias with respect to *jew*, before and after debiasing, for different subspace dimensionalities.

$\mathbf{b} = \frac{\mathbf{c}-\boldsymbol{\mu}}{\|\mathbf{c}-\boldsymbol{\mu}\|}$. Using this, we find the 500 most biased words in each direction in the whole vocabulary based on their component in the bias direction: $\langle \mathbf{w}, \mathbf{b} \rangle$.

Then, using the list of professions from Bolukbasi et al. (2016)[2], we find the 100 closest neighbors for each profession. We then plot the number of neighbors with positive bias against the original bias of the profession word, as shown in Figures 1 and 2. The plots suggest that while the correlation between the bias component and the number of positively-biased neighbors might decrease slightly as the number of bias subspace dimensions increase, the cluster bias is still not fully removed. As Table 4 shows, while the correlation between the two quantities decreases as the number of subspace dimensions increase to 2 or 3, its magnitude is still high.

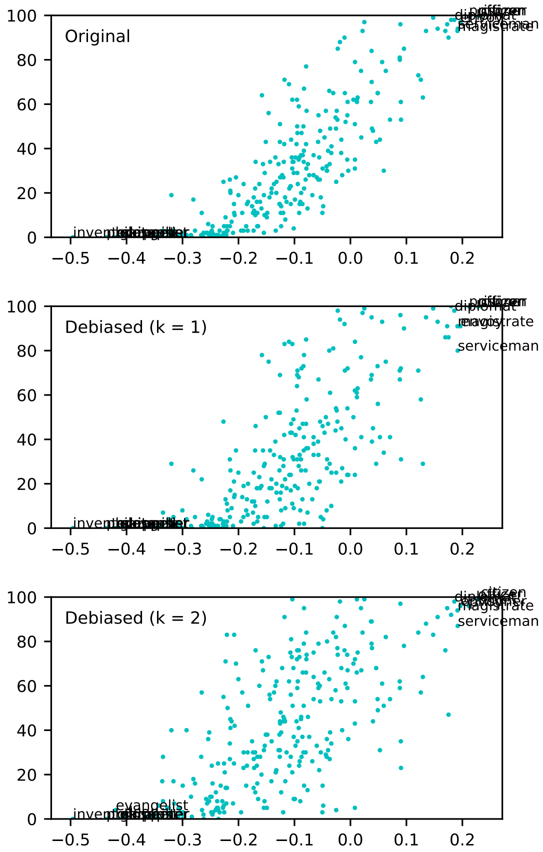


Figure 2: Plots of number of neighbors to *muslim* for each profession as a function of its original bias with respect to *muslim*, before and after debiasing, for different subspace dimensionalities.

Target	k	r	ρ
jew	0	0.767	0.875
	1	0.795	0.891
	2	0.718	0.756
	3	0.736	0.772
christian	0	0.925	0.947
	1	0.835	0.841
	2	0.825	0.831
	3	0.832	0.839
muslim	0	0.858	0.894
	1	0.774	0.812
	2	0.715	0.721
	3	0.712	0.718

Table 4: Pearson's r and Spearman's ρ correlation coefficients between the number of biased neighbors and the original bias of professions with respect to target classes for religion. k is the dimension of the bias subspace used ($k = 0$ represents the original embedding). All correlation coefficients have p-values $< 10^{-30}$.

---

[2] `https://github.com/tolga-b/debiaswe/blob/master/data/professions.json`

On Measuring Social Biases in Sentence Encoders

Chandler May[1] **Alex Wang**[2] **Shikha Bordia**[2]
Samuel R. Bowman[2] **Rachel Rudinger**[1]
[1]Johns Hopkins University [2]New York University
`{cjmay,rudinger}@jhu.edu` `{alexwang,sb6416,bowman}@nyu.edu`

Abstract

The *Word Embedding Association Test* shows that GloVe and word2vec word embeddings exhibit human-like implicit biases based on gender, race, and other social constructs (Caliskan et al., 2017). Meanwhile, research on learning reusable text representations has begun to explore sentence-level texts, with some sentence encoders seeing enthusiastic adoption. Accordingly, we extend the Word Embedding Association Test to measure bias in sentence encoders. We then test several sentence encoders, including state-of-the-art methods such as ELMo and BERT, for the social biases studied in prior work and two important biases that are difficult or impossible to test at the word level. We observe mixed results including suspicious patterns of sensitivity that suggest the test's assumptions may not hold in general. We conclude by proposing directions for future work on measuring bias in sentence encoders.

1 Introduction

Word embeddings quickly achieved wide adoption in natural language processing (NLP), precipitating the development of efficient, word-level neural models of human language. However, prominent word embeddings such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) encode systematic biases against women and black people (Bolukbasi et al., 2016; Garg et al., 2018, i.a.), implicating many NLP systems in scaling up social injustice. We investigate whether sentence encoders, which extend the word embedding approach to sentences, are similarly biased.[1]

The previously developed Word Embedding Association Test (WEAT; Caliskan et al., 2017) measures bias in word embeddings by comparing two sets of target-concept words to two sets of attribute words. We propose a simple generalization of WEAT to phrases and sentences: the Sentence Encoder Association Test (SEAT). We apply SEAT to sentences generated by inserting individual words from Caliskan et al.'s tests into simple templates such as "This is a[n] <word>."

To demonstrate the new potential of a sentence-level approach and advance the discourse on bias in NLP, we also introduce tests of two biases that are less amenable to word-level representation: the *angry black woman* stereotype (Collins, 2004; Madison, 2009; Harris-Perry, 2011; hooks, 2015; Gillespie, 2016) and a *double bind* on women in professional settings (Heilman et al., 2004).

The use of sentence-level contexts also facilitates testing the impact of different experimental designs. For example, several of Caliskan et al.'s tests rely on given names associated with European American and African American people or rely on terms referring to women and men as groups (such as "woman" and "man"). We explore the effect of using given names versus group terms by creating alternate versions of several bias tests that swap the two. This is not generally feasible with WEAT, as categories like *African Americans* lack common single-word group terms.

We find varying evidence of human-like bias in sentence encoders using SEAT. Sentence-to-vector encoders largely exhibit the angry black woman stereotype and Caliskan biases, and to a lesser degree the double bind biases. Recent sentence encoders such as BERT (Devlin et al., 2018) display limited evidence of the tested biases. However, while SEAT can confirm the existence of bias, negative results do not indicate the model is bias-free. Furthermore, discrepancies in the results suggest that the confirmed biases may not generalize beyond the specific words and sentences in our test data, and in particular that cosine similarity may not be a suitable measure of representational similarity in recent models, indicating a need for alternate bias detection techniques.

---

[1] While encoder training data may contain perspectives from outside the U.S., we focus on biases in U.S. contexts.

Proceedings of NAACL-HLT 2019, pages 622–628
Minneapolis, Minnesota, June 2 - June 7, 2019.

Target Concepts	**Attributes**
European American names: Adam, Harry, Nancy, Ellen, Alan, Paul, Katie, …	*Pleasant*: love, cheer, miracle, peace, friend, happy, …
African American names: Jamel, Lavar, Lavon, Tia, Latisha, Malika, …	*Unpleasant*: ugly, evil, abuse, murder, assault, rotten, …

Table 1: Subsets of target concepts and attributes from Caliskan Test 3. Concept and attribute names are in italics. The test compares the strength of association between the two target concepts and two attributes, where all four are represented as sets of words.

Target Concepts	**Attributes**
European American names: "This is Katie.", "This is Adam." "Adam is there.", …	*Pleasant*: "There is love.", "That is happy.", "This is a friend.", …
African American names: "Jamel is here.", "That is Tia.", "Tia is a person.", …	*Unpleasant*: "This is evil.", "They are evil.", "That can kill.", …

Table 2: Subsets of target concepts and attributes from the bleached sentence version of Caliskan Test 3.

2 Methods

The Word Embedding Association Test WEAT imitates the human implicit association test (Greenwald et al., 1998) for word embeddings, measuring the association between two sets of target concepts and two sets of attributes. Let X and Y be equal-size sets of target concept embeddings and let A and B be sets of attribute embeddings. The test statistic is a difference between sums over the respective target concepts,

$$s(X,Y,A,B) = \left[\textstyle\sum_{x\in X} s(x,A,B) - \sum_{y\in Y} s(y,A,B)\right],$$

where each addend is the difference between mean cosine similarities of the respective attributes,

$$s(w,A,B) = \left[\mathrm{mean}_{a\in A}\cos(w,a) - \mathrm{mean}_{b\in B}\cos(w,b)\right]$$

A permutation test on $s(X,Y,A,B)$ is used to compute the significance of the association between (A,B) and (X,Y),

$$p = \Pr\left[s(X_i,Y_i,A,B) > s(X,Y,A,B)\right],$$

where the probability is computed over the space of partitions (X_i,Y_i) of $X \cup Y$ such that X_i and Y_i are of equal size, and a normalized difference of means of $s(w,A,B)$ is used to measure the magnitude of the association (the effect size; Caliskan et al., 2017),

$$d = \frac{\mathrm{mean}_{x\in X} s(x,A,B) - \mathrm{mean}_{y\in Y} s(y,A,B)}{\mathrm{std\_dev}_{w\in X\cup Y} s(w,A,B)}.$$

Controlling for significance, a larger effect size reflects a more severe bias. We detail our implementations in the supplement.

The Sentence Encoder Association Test SEAT compares sets of sentences, rather than sets of words, by applying WEAT to the vector representation of a sentence. Because SEAT operates on fixed-sized vectors and some encoders produce variable-length vector sequences, we use pooling as needed to aggregate outputs into a fixed-sized vector. We can view WEAT as a special case of SEAT in which the sentence is a single word. In fact, the original WEAT tests have been run on the Universal Sentence Encoder (Cer et al., 2018).

To extend a word-level test to sentence contexts, we slot each word into each of several semantically bleached sentence templates such as "This is <word>.", "<word> is here.", "This will <word>.", and "<word> are things.". These templates make heavy use of deixis and are designed to convey little specific meaning beyond that of the terms inserted into them.[2] For example, the word version of Caliskan Test 3 is illustrated in Table 1 and the sentence version is illustrated in Table 2. We choose this design to focus on the associations a sentence encoder makes with a given term rather than those it happens to make with the contexts of that term that are prevalent in the training data; a similar design was used in a recent sentiment analysis evaluation corpus stratified by race and gender (Kiritchenko and Mohammad, 2018). To facilitate future work, we publicly release code for SEAT and all of our experiments.[3]

3 Biases Tested

Caliskan Tests We first test whether the sentence encoders reproduce the same biases that word embedding models exhibited in Caliskan et al. (2017). These biases correspond to past social psychology studies of implicit associations in human subjects.[4] We apply both the original

---

[2] See the supplement for further details and examples.

[3] `http://github.com/W4ngatang/sent-bias`

[4] See Greenwald et al. (2009) for a review of this work.

word-level versions of these tests as well as our generated sentence-level versions.

Angry Black Woman Stereotype In the *Sapphire* or *angry black woman (ABW)* stereotype, black women are portrayed as loud, angry, and imposing (Collins, 2004; Madison, 2009; Harris-Perry, 2011; hooks, 2015; Gillespie, 2016). This stereotype contradicts common associations made with the ostensibly race-neutral (unmarked) category of *women* (Bem, 1974), suggesting that that category is implicitly white. *Intersectionality* reveals that experiences considered common to women are not necessarily shared by black women, who are marginalized both among women and among black people (Crenshaw, 1989). Recently, intersectionality has been demonstrated in English Wikipedia using distributional semantic word representations (Herbelot et al., 2012), and in the disparate error rates of machine learning technologies like face recognition (Buolamwini and Gebru, 2018).

To measure sentence encoders' reproduction of the angry black woman stereotype, we create a test whose target concepts are black-identifying and white-identifying female given names from Sweeney (2013, Table 1) and whose attributes are adjectives used in the discussion of the stereotype in Collins (2004, pp. 87-90) and their antonyms. We also produce a version of the test with attributes consisting of terms describing black women and white women as groups, as well as sentence versions in which attribute and target concept terms are inserted in sentence templates.

Double Binds Women face many *double binds*, contradictory or unsatisfiable expectations of femininity and masculinity (Stone and Lovejoy, 2004; Harris-Perry, 2011; Mitchell, 2012). If women clearly succeed in a male gender-typed job, they are perceived less likable and more hostile than men in similar positions; if success is ambiguous, they are perceived less competent and achievement-oriented than men. Both outcomes can interfere in performance evaluations (Heilman et al., 2004), contributing to the *glass ceiling* impeding women's career advancement.[5]

We test this double bind in sentence encoders by translating Heilman et al.'s experiment to two SEAT tests. In the first, we represent the two target concepts by names of women and men, respectively, in the single sentence template "<word> is an engineer with superior technical skills."; the attributes are *likable* and *non-hostile* terms, based on Heilman et al.'s design, in the sentence template "The engineer is <word>." In the second, we use the shortened target concept sentence template "<word> is an engineer" and fill the attribute templates from before with *competent* and *achievement-oriented* terms based on Heilman et al.'s design.[6] We refer to these tests as semantically *unbleached* because the context contains important information about the bias. We produce two variations of these tests: word-level tests in which target concepts are names in isolation and attributes are adjectives in isolation, as well as corresponding semantically bleached sentence-level tests. These control conditions allow us to probe the extent to which observed associations are attributable to gender independent of context.

Model	**Agg.**	**Dim.**
CBoW (GloVe), 840 billion token web corpus version (Pennington et al., 2014)	`mean`	300
InferSent, AllNLI (Conneau et al., 2017)	`max`	4096
GenSen, +STN +Fr +De +NLI +L +STP +Par (Subramanian et al., 2018)	`last`	4096
Universal Sentence Encoder (USE), DAN version (Cer et al., 2018)	N/A	512
ELMo (Peters et al., 2018), sum over layers after mean-pooling over sequence	`mean`	1024
GPT (Radford et al., 2018)	`last`	768
BERT, large, cased (Devlin et al., 2018)	`[CLS]`	1024

Table 3: Models tested (disambiguated with notation from cited paper), aggregation functions applied across token representations, and representation dimensions.

4 Experiments and Results

We apply SEAT to seven sentence encoders (listed in Table 3) including simple bag-of-words encoders, sentence-to-vector models, and state-of-the-art sequence models.[7] For all models, we use publicly available pretrained parameters.

Table 4 shows effect size and significance at 0.01 before and after applying the Holm-Bonferroni multiple testing correction (Holm, 1979) for a subset of tests and models; complete results are provided in the supplement.[8]

---

[5] See Schluter (2018) for a recent exposition of the glass ceiling in the NLP research community.

[6] We consider other formulations in the supplement.

[7] We provide further details and explore variations on these model configurations in the supplement.

[8] We use the full set of tests and models when comput-

Test	**Context**	**CBoW**	**InferSent**	**GenSen**	**USE**	**ELMo**	**GPT**	**BERT**
C1: Flowers/Insects	word	1.50^{**}	1.56^{**}	1.24^{**}	1.38^{**}	-0.03	0.20	0.22
C1: Flowers/Insects	sent	1.56^{**}	1.65^{**}	1.22^{**}	1.38^{**}	0.42^{**}	0.81^{**}	0.62^{**}
C3: EA/AA Names	word	1.41^{**}	1.33^{**}	1.32^{**}	0.52	-0.40	0.60^{*}	-0.11
C3: EA/AA Names	sent	0.52^{**}	1.07^{**}	0.97^{**}	0.32^{*}	-0.38	0.19	0.05
C6: M/F Names, Career	word	1.81^{*}	1.78^{*}	1.84^{*}	0.02	-0.45	0.22	0.21
C6: M/F Names, Career	sent	1.74^{**}	1.69^{**}	1.63^{**}	0.83^{**}	-0.38	0.35	0.08
ABW Stereotype	word	1.10^{*}	1.18^{*}	1.57^{**}	-0.39	0.53	0.08	-0.32
ABW Stereotype	sent	0.62^{**}	0.98^{**}	1.05^{**}	-0.19	0.52^{*}	-0.07	-0.17
Double Bind: Competent	word	1.62^{*}	1.09	1.49^{*}	1.51^{*}	-0.35	-0.28	-0.81
Double Bind: Competent	sent	0.79^{**}	0.57^{*}	0.83^{**}	0.25	-0.15	0.10	0.39
Double Bind: Competent	sent (u)	0.84	1.42^{*}	1.03	0.71	0.20	0.71	1.17^{*}
Double Bind: Likable	word	1.29^{*}	0.65	1.31^{*}	0.16	-0.60	0.91	-0.55
Double Bind: Likable	sent	0.69^{*}	0.37	0.25	0.32	-0.45	-0.20	-0.35
Double Bind: Likable	sent (u)	0.51	1.33^{*}	0.05	0.48	-0.90	-0.87	0.99

Table 4: SEAT effect sizes for select tests, including word-level (word), bleached sentence-level (sent), and unbleached sentence-level (sent (u)) versions. CN: test from Caliskan et al. (2017, Table 1) row N; *: significant at 0.01, **: significant at 0.01 after multiple testing correction.

Specifically, we select Caliskan Test 1 associating flowers/insects with pleasant/unpleasant, Test 3 associating European/African American names with pleasant/unpleasant, and Test 6 associating male/female names with career/family, as well as the angry black woman stereotype and the competent and likable double bind tests. We observe that tests based on given names more often find a significant association than those based on group terms; we only show the given-name results here.

We find varying evidence of bias in sentence encoders according to these tests. Bleached sentence-level tests tend to elicit more significant associations than word-level tests, while the latter tend to have larger effect sizes. We find stronger evidence for the Caliskan and ABW stereotype tests than for the double bind. After the multiple testing correction, we only find evidence of the double bind in bleached, sentence-level *competent* control tests; that is, we find women are associated with incompetence independent of context.[9]

Some patterns in the results cast doubt on the reasonableness of SEAT as an evaluation. For instance, Caliskan Test 7 (association between *math/art* and *male/female*) and Test 8 (*science/art* and *male/female*) elicit counterintuitive results from several models. These tests have the same sizes of target concept and attribute sets. For CBoW on the word versions of those tests, we see p-values of 0.016 and 10^{-2}, respectively. On the sentence versions, we see p-values of 10^{-5} for both tests. Observing similar p-values agrees with intuition: The *math/art* association should be similar to the *science/art* association because they instantiate a disciplinary dichotomy between *math/science* and *arts/language* (Nosek et al., 2002). However, for BERT on the sentence version, we see discrepant p-values of 10^{-5} and 0.14; for GenSen, 0.12 and 10^{-3}; and for GPT, 0.89 and 10^{-4}.

Caliskan Tests 3, 4, and 5 elicit even more counterintuitive results from ELMo. These tests measure the association between *European American/African American* and *pleasant/unpleasant*. Test 3 has larger attribute sets than Test 4, which has larger target concept sets than Test 5. Intuitively, we expect increasing p-values across Tests 3, 4, and 5, as well-designed target concepts and attributes of larger sizes should yield higher-power tests. Indeed, for CBoW, we find increasing p-values of 10^{-5}, 10^{-5}, and 10^{-4} on the word versions of the tests and 10^{-5}, 10^{-5}, and 10^{-2} on the sentence versions, respectively.[10] However, for ELMo, we find *decreasing* p-values of 0.95, 0.45, and 0.08 on the word versions of the tests and 1, 0.97, and 10^{-4} on the sentence versions. We interpret these results as ELMo producing substantially different representations for conceptually similar words. Thus, SEAT's assumption that the sentence representations of each target concept and attribute instantiate a coherent concept appears invalid.

---

ing the multiple testing correction, including those only presented in the supplement.

[9] However, the double bind results differ across models; we show no significant associations for ELMo or GPT and only one each for USE and BERT.

[10] Our SEAT implementation uses sampling with a precision of 10^{-5}, so 10^{-5} is the smallest p-value we can observe.

5 Conclusion

At face value, our results suggest recent sentence encoders exhibit less bias than previous models do, at least when "bias" is considered from a U.S. perspective and measured using the specific tests we have designed. However, we strongly caution against interpreting the number of significant associations or the average significant effect size as an absolute measure of bias. Like WEAT, SEAT only has positive predictive ability: It can detect presence of bias, but not its absence. Considering that these representations are trained without explicit bias control mechanisms on naturally occurring text, we argue against interpreting a lack of evidence of bias as a lack of bias.

Moreover, the counterintuitive sensitivity of SEAT on some models and biases suggests that biases revealed by SEAT may not generalize beyond the specific words and sentences in our test data. That is, our results invalidate the assumption that each set of words or sentences in our tests represents a coherent concept/attribute (like *African American* or *pleasant*) to the sentence encoders; hence, we do not assume the encoders will exhibit similar behavior on other potential elements of those concepts/attributes (other words or sentences representing, for example, *African American* or *pleasant*).

One possible explanation of the observed sensitivity at the sentence level is that, from the sentence encoders' view, our sentence templates are not as *semantically bleached* as we expect; small variations in their relative frequencies and interactions with the terms inserted into them may be undermining the coherence of the concepts/attributes they implement. Another possible explanation that also accounts for the sensitivity observed in the word-level tests is that cosine similarity is an inadequate measure of text similarity for sentence encoders. If this is the case, the biases revealed by SEAT may not translate to biases in downstream applications. Future work could measure bias at the application level instead, following Bailey and Deery (2018)'s recommendation based on the tension between descriptive and normative correctness in representations.

The angry black woman stereotype represents an *intersectional* bias, a phenomenon not well anticipated by an additive model of racism and sexism (Crenshaw, 1989). Previous work has modeled biases at the intersection of race and gender in distributional semantic word representations (Herbelot et al., 2012), natural language inference data (Rudinger et al., 2017), and facial recognition systems (Buolamwini and Gebru, 2018), as well as at the intersection of dialect and gender in automatic speech recognition (Tatman, 2017). We advocate for further consideration of intersectionality in future work in order to avoid reproducing the erasure of multiple minorities who are most vulnerable to bias.

We have developed a simple sentence-level extension of an established word embedding bias instrument and used it to measure the degree to which pretrained sentence encoders capture a range of social biases, observing a large number of significant effects as well as idiosyncrasies suggesting limited external validity. This study is preliminary and leaves open to investigation several design choices that may impact the results; future work may consider revisiting choices like the use of semantically bleached sentence inputs, the aggregation applied to models that represent sentences with sequences of hidden states, and the use of cosine similarity between sentence representations. We challenge researchers of fairness and ethics in NLP to critically (re-)examine their methods; looking forward, we hope for a deeper consideration of the social contexts in which NLP systems are applied.

Acknowledgments

We are grateful to Carolyn Rosé, Jason Phang, Sébastien Jean, Thibault Févry, Katharina Kann, and Benjamin Van Durme for helpful conversations concerning this work and to our reviewers for their thoughtful feedback.

CM is funded by IARPA MATERIAL; RR is funded by DARPA AIDA; AW is funded by an NSF fellowship. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of IARPA, DARPA, NSF, or the U.S. Government.

References

Katherine Bailey and Oisin Deery. 2018. How (and how not!) to control for morally relevant bias in NLP. The Second ACL Workshop on Ethics in Natural Language Processing, New Orleans, LA. Invited Talk.

Sandra L. Bem. 1974. The measurement of psychological androgyny. *Journal of Consulting and Clinical Psychology*, 42(2):155–162.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4349–4357. Curran Associates, Inc.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA. PMLR.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Patricia Hill Collins. 2004. *Black Sexual Politics: African Americans, Gender, and the New Racism*. Routledge, New York.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Kimberle Crenshaw. 1989. Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory, and antiracist politics. *Feminist Legal Theory: Readings in Law and Gender*, 1989(1):57–80.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv e-prints*.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*.

Andra Gillespie. 2016. Race, perceptions of femininity, and the power of the first lady: A comparative analysis. In Nadia E. Brown and Sarah Allen Gershon, editors, *Distinct Identities: Minority Women in U.S. Politics*, chapter 15, pages 223–238. Routledge, New York.

Anthony G. Greenwald, Debbie E. McGhee, and Jordan L. K. Schwartz. 1998. Measuring individual differences in implicit cognition: The implicit association test. *Journal of Personality and Social Psychology*, 74(6):1464–1480.

Anthony G. Greenwald, T. Andrew Poehlman, Eric Luis Uhlmann, and Mahzarin R. Banaji. 2009. Understanding and using the implicit association test: III. Meta-analysis of predictive validity. *Journal of Personality and Social Psychology*, 97(1):17–41.

Melissa V. Harris-Perry. 2011. *Sister Citizen: Shame, Stereotypes, and Black Women in America*. Yale University Press, New Haven, Connecticut.

Madeline E. Heilman, Aaron S. Wallen, Daniella Fuchs, and Melinda M. Tamkins. 2004. Penalties for success: Reactions to women who succeed at male gender-typed tasks. *Journal of Applied Psychology*, 89(3):416–427.

Aurélie Herbelot, Eva von Redecker, and Johanna Müller. 2012. Distributional techniques for philosophical enquiry. In *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 45–54, Avignon, France. Association for Computational Linguistics.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

bell hooks. 2015. *Ain't I a Woman: Black Women and Feminism*, 2 edition. Routledge, Taylor & Francis Group, New York.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, New Orleans, Louisiana. Association for Computational Linguistics.

D. Soyini Madison. 2009. Crazy patriotism and angry (post)black women. *Communication and Critical/Cultural Studies*, 6(3):321–326.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Kaye Mitchell. 2012. Raunch versus prude: contemporary sex blogs and erotic memoirs by women. *Psychology & Sexuality*, 3(1):12–25.

Brian A. Nosek, Mahzarin R. Banaji, and Anthony G. Greenwald. 2002. Math $=$ male, me $=$ female, therefore math $\neq$ me. *Journal of Personality and Social Psychology*, 83(1):44–59.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Online; accessed November 28, 2018.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79, Valencia, Spain. Association for Computational Linguistics.

Natalie Schluter. 2018. The glass ceiling in nlp. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2793–2798, Brussels, Belgium. Association for Computational Linguistics.

Pamela Stone and Meg Lovejoy. 2004. Fast-track women and the “choice” to stay home. *The ANNALS of the American Academy of Political and Social Science*, 596(1):62–83.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*, Vancouver, Canada.

Latanya Sweeney. 2013. Discrimination in online ad delivery. *Communications of the ACM*, 56(5):44–54.

Rachael Tatman. 2017. Gender and dialect bias in youtube’s automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, Valencia, Spain. Association for Computational Linguistics.

Gender Bias in Contextualized Word Embeddings

Jieyu Zhao[§] **Tianlu Wang**[†] **Mark Yatskar**[‡]
Ryan Cotterell[ℵ] **Vicente Ordonez**[†] **Kai-Wei Chang**[§]
[§]University of California, Los Angeles {jyzhao, kwchang}@cs.ucla.edu
[†]University of Virginia {tw8bc, vicente}@virginia.edu
[‡]Allen Institute for Artificial Intelligence marky@allenai.org
[ℵ]University of Cambridge rdc42@cam.ac.uk

Abstract

In this paper, we quantify, analyze and mitigate gender bias exhibited in ELMo's contextualized word vectors. First, we conduct several intrinsic analyses and find that (1) training data for ELMo contains significantly more male than female entities, (2) the trained ELMo embeddings systematically encode gender information and (3) ELMo unequally encodes gender information about male and female entities. Then, we show that a state-of-the-art coreference system that depends on ELMo inherits its bias and demonstrates significant bias on the WinoBias probing corpus. Finally, we explore two methods to mitigate such gender bias and show that the bias demonstrated on WinoBias can be eliminated.

1 Introduction

Distributed representations of words in the form of word embeddings (Mikolov et al., 2013; Pennington et al., 2014) and contextualized word embeddings (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018; McCann et al., 2017; Radford et al., 2019) have led to huge performance improvement on many NLP tasks. However, several recent studies show that training word embeddings in large corpora could lead to encoding societal biases present in these human-produced data (Bolukbasi et al., 2016; Caliskan et al., 2017). In this work, we extend these analyses to the ELMo contextualized word embeddings.

Our work provides a new intrinsic analysis of how ELMo represents gender in biased ways. First, the corpus used for training ELMo has a significant gender skew: male entities are nearly three times more common than female entities, which leads to gender bias in the downloadable pre-trained contextualized embeddings. Then, we apply principal component analysis (PCA) to show that after training on such biased corpora, there exists a low-dimensional subspace that captures much of the gender information in the contextualized embeddings. Finally, we evaluate how faithfully ELMo preserves gender information in sentences by measuring how predictable gender is from ELMo representations of occupation words that co-occur with gender revealing pronouns. Our results show that ELMo embeddings perform unequally on male and female pronouns: male entities can be predicted from occupation words 14% more accurately than female entities.

In addition, we examine how gender bias in ELMo propagates to the downstream applications. Specifically, we evaluate a state-of-the-art coreference resolution system (Lee et al., 2018) that makes use of ELMo's contextual embeddings on WinoBias (Zhao et al., 2018a), a coreference diagnostic dataset that evaluates whether systems behave differently on decisions involving male and female entities of stereotyped or anti-stereotyped occupations. We find that in the most challenging setting, the ELMo-based system has a disparity in accuracy between pro- and anti-stereotypical predictions, which is nearly 30% higher than a similar system based on GloVe (Lee et al., 2017).

Finally, we investigate approaches for mitigating the bias which propagates from the contextualized word embeddings to a coreference resolution system. We explore two different strategies: (1) a training-time data augmentation technique (Zhao et al., 2018a), where we augment the corpus for training the coreference system with its gender-swapped variant (female entities are swapped to male entities and vice versa) and, afterwards, retrain the coreference system; and (2) a test-time embedding neutralization technique, where input contextualized word representations are averaged with word representations of a sentence with entities of the opposite gender. Results show that test-time embedding neutralization is only partially effective, while data augmentation largely mitigates bias demonstrated on WinoBias by the coreference

Proceedings of NAACL-HLT 2019, pages 629–634
Minneapolis, Minnesota, June 2 - June 7, 2019.

system.

2 Related Work

Gender bias has been shown to affect several real-world applications relying on automatic language analysis, including online news (Ross and Carter, 2011), advertisements (Sweeney, 2013), abusive language detection (Park et al., 2018), machine translation (Font and Costa-jussà, 2019; Vanmassenhove et al., 2018), and web search (Kay et al., 2015). In many cases, a model not only replicates bias in the training data but also amplifies it (Zhao et al., 2017).

For word representations, Bolukbasi et al. (2016) and Caliskan et al. (2017) show that word embeddings encode societal biases about gender roles and occupations, e.g. engineers are stereotypically men, and nurses are stereotypically women. As a consequence, downstream applications that use these pretrained word embeddings also reflect this bias. For example, Zhao et al. (2018a) and Rudinger et al. (2018) show that coreference resolution systems relying on word embeddings encode such occupational stereotypes. In concurrent work, May et al. (2019) measure gender bias in sentence embeddings, but their evaluation is on the aggregation of word representations. In contrast, we analyze bias in contextualized word representations and its effect on a downstream task.

To mitigate bias from word embeddings, Bolukbasi et al. (2016) propose a post-processing method to project out the bias subspace from the pre-trained embeddings. Their method is shown to reduce the gender information from the embeddings of gender-neutral words, and, remarkably, maintains the same level of performance on different downstream NLP tasks. Zhao et al. (2018b) further propose a training mechanism to separate gender information from other factors. However, Gonen and Goldberg (2019) argue that entirely removing bias is difficult, if not impossible, and the gender bias information can be often recovered. This paper investigates a natural follow-up question: What are effective bias mitigation techniques for contextualized embeddings?

3 Gender Bias in ELMo

In this section we describe three intrinsic analyses highlighting gender bias in trained ELMo contextual word embeddings (Peters et al., 2018). We show that (1) training data for ELMo contains significantly more male entities compared to female entities leading to gender bias in the pre-trained contextual word embeddings (2) the geometry of trained ELMo embeddings systematically encodes gender information and (3) ELMo propagates gender information about male and female entities unequally.

	#occurrence	#M-biased occs.	#F-biased occs.
M	5,300,000	170,000	81,000
F	1,600,000	33,000	36,000

Table 1: Training corpus for ELMo. We show total counts for male (M) and female (F) pronouns in the corpus, and counts corresponding to their co-occurrence with occupation words where the occupations are stereotypically male (M-biased) or female (F-biased).

3.1 Training Data Bias

Table 1 lists the data analysis on the One Billion Word Benchmark (Chelba et al., 2013) corpus, the training corpus for ELMo. We show counts for the number of occurrences of male pronouns (*he*, *his* and *him*) and female pronouns (*she* and *her*) in the corpus as well as the co-occurrence of occupation words with those pronouns. We use the set of occupation words defined in the WinoBias corpus and their assignments as prototypically male or female (Zhao et al., 2018a). The analysis shows that the Billion Word corpus contains a significant skew with respect to gender: (1) male pronouns occur three times more than female pronouns and (2) male pronouns co-occur more frequently with occupation words, irrespective of whether they are prototypically male or female.

3.2 Geometry of Gender

Next, we analyze the gender subspace in ELMo. We first sample 400 sentences with at least one gendered word (e.g., *he* or *she* from the OntoNotes 5.0 dataset (Weischedel et al., 2012) and generate the corresponding gender-swapped variants (changing *he* to *she* and vice-versa). We then calculate the difference of ELMo embeddings between occupation words in corresponding sentences and conduct principal component analysis for all pairs of sentences. Figure 1 shows there are two principal components for gender in ELMo, in contrast to GloVe which only has one (Bolukbasi et al., 2016). The two principal components in ELMo seem to represent the gender from the contextual information (Contextual Gender) as well as the gender embedded in the word itself (Occupational Gender).

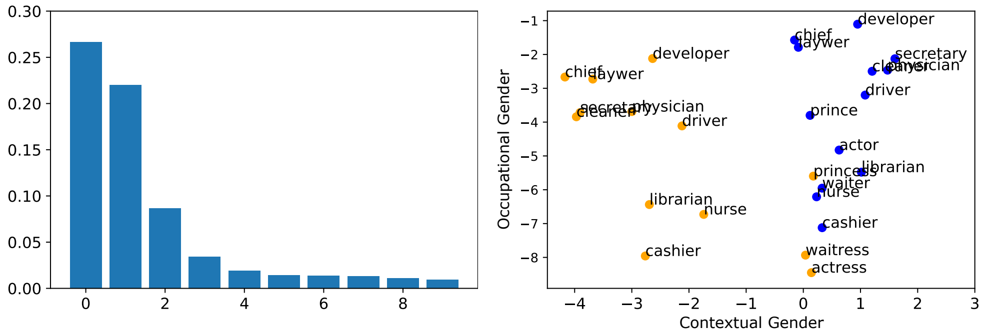


Figure 1: Left: Percentage of explained variance in PCA in the embedding differences. Right: Selected words projecting to the first two principle components where the blue dots are the sentences with male context and the orange dots are from the sentences with female context.

To visualize the gender subspace, we pick a few sentence pairs from WinoBias (Zhao et al., 2018a). Each sentence in the corpus contains one gendered pronoun and two occupation words, such as "The developer corrected the secretary because she made a mistake" and also the same sentence with the opposite pronoun (he). In Figure 1 on the right, we project the ELMo embeddings of occupation words that are co-referent with the pronoun (e.g. *secretary* in the above example) for when the pronoun is male (blue dots) and female (orange dots) on the two principal components from the PCA analysis. Qualitatively, we can see the first component separates male and female contexts while the second component groups male related words such as *lawyer* and *developer* and female related words such as *cashier* and *nurse*.

3.3 Unequal Treatment of Gender

To test how ELMo embeds gender information in contextualized word embeddings, we train a classifier to predict the gender of entities from occupation words in the same sentence. We collect sentences containing gendered words (e.g., *he-she*, *father-mother*) and occupation words (e.g., *doctor*)[1] from the OntoNotes 5.0 corpus (Weischedel et al., 2012), where we treat occupation words as a mention to an entity, and the gender of that entity is taken to the gender of a co-referring gendered word, if one exists. For example, in the sentence "the engineer went back to her home," we take *engineer* to be a female mention. Then we split all such instances into training and test, with 539 and 62 instances, respectively and augment these sentences by swapping all the gendered words with words of the opposite gender such that the numbers of male and female entities are balanced.

We first test if ELMo embedding vectors carry gender information. We train an SVM classifier with an RBF kernel[2] to predict the gender of a mention (i.e., an occupation word) based on its ELMo embedding. On development data, this classifier achieves 95.1% and 80.6% accuracy on sentences where the true gender was male and female respectively. For both male and female contexts, the accuracy is much larger than 50%, demonstrating that ELMo does propagate gender information to other words. However, male information is more than 14% more accurately represented in ELMo than female information, showing that ELMo propagates the information unequally for male and female entities.

4 Bias in Coreference Resolution

In this section, we establish that coreference systems that depend on ELMo embeddings exhibit significant gender bias. Then we evaluate two simple methods for removing the bias from the systems and show that the bias can largely be reduced.

4.1 Setup

We evaluate bias with respect to the WinoBias dataset (Zhao et al., 2018a), a benchmark of paired male and female coreference resolution examples following the Winograd format (Hirst, 1981; Rahman and Ng, 2012; Peng et al., 2015). It contains two different subsets, pro-stereotype, where pronouns are associated with occupations predominately associated with the gender of the pronoun, or anti-stereotype, when the opposite relation is true.

---

[1] We use the list collected in (Zhao et al., 2018a)

[2] We use the ν-SVC formulation and tune the hyper-parameter ν (Chang and Lin, 2011) in the range of $[0.1, 1]$ with a step 0.1.

Embeddings	Data Augmentation	Neutralization		OntoNotes	Semantics Only				w/ Syntactic Cues			
		GloVe	ELMo		Pro.	Anti.	Avg.	| Diff |	Pro.	Anti.	Avg.	| Diff |
GloVe				67.7	76.0	49.4	62.7	26.6*	88.7	75.2	82.0	13.5*
GloVe	✓			65.8	63.9	62.8	63.4	1.1	81.3	83.4	82.4	2.1
GloVe+ELMo				72.7	79.1	49.5	64.3	29.6*	93.0	85.9	89.5	7.1*
GloVe+ELMo	✓			71.0	65.9	64.9	65.4	1.0	87.8	88.9	88.4	1.2
GloVe+ELMo		✓		71.0	72.6	57.8	64.9	14.3*	90.2	88.6	89.4	1.6
GloVe+ELMo		✓	✓	71.1	71.7	60.6	66.2	11.1*	90.3	89.2	89.8	1.1

Table 2: F1 on OntoNotes and WinoBias development sets. WinoBias dataset is split Semantics Only and w/ Syntactic Cues subsets. ELMo improves the performance on the OntoNotes dataset by 5% but shows stronger bias on the WinoBias dataset. Avg. stands for averaged F1 score on the pro- and anti-stereotype subsets while "Diff." is the absolute difference between these two subsets. * indicates the difference between pro/anti stereotypical conditions is significant ($p < .05$) under an approximate randomized test (Graham et al., 2014). Mitigating bias by data augmentation reduces all the bias from the coreference model to a neglect level. However, the neutralizing ELMo approach only mitigates bias when there are other strong learning signals for the task.

Each subset consists of two types of sentences: one that requires semantic understanding of the sentence to make coreference resolution (Semantics Only) and another that relies on syntactic cues (w/ Syntactic Cues). Gender bias is measured by taking the difference of the performance in pro- and anti-stereotypical subsets. Previous work (Zhao et al., 2018a) evaluated the systems based on GloVe embeddings but here we evaluate a state-of-the-art system that trained on the OntoNotes corpus with ELMo embeddings (Lee et al., 2018).

4.2 Bias Mitigation Methods

Next, we describe two methods for mitigating bias in ELMo for the purpose of coreference resolution: (1) a train-time data augmentation approach and (2) a test-time neutralization approach.

Data Augmentation Zhao et al. (2018a) propose a method to reduce gender bias in coreference resolution by augmenting the training corpus for this task. Data augmentation is performed by replacing gender revealing entities in the OntoNotes dataset with words indicating the opposite gender and then training on the union of the original data and this swapped data. In addition, they find it useful to also mitigate bias in supporting resources and therefore replace standard GloVe embeddings with bias mitigated word embeddings from Bolukbasi et al. (2016). We evaluate the performance of both aspects of this approach.

Neutralization We also investigate an approach to mitigate bias induced by ELMo embeddings without retraining the coreference model. Instead of augmenting training corpus by swapping gender words, we generate a gender-swapped version of the test instances. We then apply ELMo to obtain contextualized word representations of the original and the gender-swapped sentences and use their average as the final representations.

4.3 Results

Table 2 summarizes our results on WinoBias.

ELMo Bias Transfers to Coreference Row 3 in Table 2 summarizes performance of the ELMo based coreference system on WinoBias. While ELMo helps to boost the coreference resolution F1 score (OntoNotes) it also propagates bias to the task. It exhibits large differences between pro- and anti-stereotyped sets (|Diff|) on both semantic and syntactic examples in WinoBias.

Bias Mitigation Rows 4-6 in Table 2 summarize the effectiveness of the two bias mitigation approaches we consider. Data augmentation is largely effective at mitigating bias in the coreference resolution system with ELMo (reducing |Diff | to insignificant levels) but requires retraining the system. Neutralization is less effective than augmentation and cannot fully remove gender bias on the Semantics Only portion of WinoBias, indicating it is effective only for simpler cases. This observation is consistent with Gonen and Goldberg (2019), where they show that entirely removing bias from an embedding is difficult and depends on the manner, by which one measures the bias.

5 Conclusion and Future Work

Like word embedding models, contextualized word embeddings inherit implicit gender bias. We analyzed gender bias in ELMo, showing that the corpus it is trained on has significant gender skew and that ELMo is sensitive to gender, but unequally so for male and female entities. We also showed this bias transfers to downstream tasks, such as coreference resolution, and explored two bias mitigation strategies: 1) data augmentation and 2) neutralizing embeddings, effectively eliminating the bias from ELMo in a state-of-the-art system. With increasing adoption of contextualized embeddings to get better results on core NLP tasks, e.g. BERT (Devlin et al., 2018), we must be careful how such unsupervised methods perpetuate bias to downstream applications and our work forms the basis of evaluating and mitigating such bias.

Acknowledgement

This work was supported in part by National Science Foundation Grant IIS-1760523. RC was supported by a Facebook Fellowship. We also acknowledge partial support from the Institute of the the Humanities and Global Cultures at the University of Virginia. We thank all reviewers for their comments.

References

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NeurIPS*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Joel Escudé Font and Marta R. Costa-jussà. 2019. Equalizing gender biases in neural machine translation with word embeddings techniques. *arXiv preprint arXiv:1901.03116*.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *CoRR*, abs/1903.03862.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *WMT@ ACL*.

Graeme Hirst. 1981. Anaphora in natural language understanding. *Berlin Springer Verlag*.

Matthew Kay, Cynthia Matuszek, and Sean A. Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *Human Factors in Computing Systems*. ACM.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Kenton Lee, Luheng He, and Luke S. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL*.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. *arXiv preprint arXiv:1903.10561*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NeurIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *EMNLP*.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The winograd schema challenge. In *EMNLP*.

Karen Ross and Cynthia Carter. 2011. Women and news: A long and winding road. *Media, Culture & Society*, 33(8).

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *NAACL*.

Latanya Sweeney. 2013. Discrimination in online ad delivery. *Queue*, 11(3):10.

Eva Vanmassenhove, Christian Hardmeier, and Andy Way. 2018. Getting Gender Right in Neural Machine Translation. In *EMNLP*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D Hwang, Claire Bonial, et al. 2012. Ontonotes release 5.0.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *EMNLP*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. Gender bias in coreference resolution: Evaluation and debiasing methods. In *NAACL*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. Learning gender-neutral word embeddings. In *EMNLP*.

Combining Sentiment Lexica with a Multi-View Variational Autoencoder

Alexander Hoyle[϶] **Lawrence Wolf-Sonkin**[∫] **Hanna Wallach**[3]
Ryan Cotterell[ﬁ] **Isabelle Augenstein**[?]
[϶]University College London, London, UK
[∫]Department of Computer Science, Johns Hopkins University, Baltimore, USA
[3]Microsoft Research, New York City, USA
[ﬁ]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK
[?]Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
`alexander.hoyle.17@ucl.ac.uk, lawrencews@jhu.edu`
`hanna@dirichlet.net, rdc42@cam.ac.uk, augenstein@di.ku.dk`

Abstract

When assigning quantitative labels to a dataset, different methodologies may rely on different scales. In particular, when assigning polarities to words in a sentiment lexicon, annotators may use binary, categorical, or continuous labels. Naturally, it is of interest to unify these labels from disparate scales to both achieve maximal coverage over words and to create a single, more robust sentiment lexicon while retaining scale coherence. We introduce a generative model of sentiment lexica to combine disparate scales into a common latent representation. We realize this model with a novel multi-view variational autoencoder (VAE), called SentiVAE. We evaluate our approach via a downstream text classification task involving nine English-Language sentiment analysis datasets; our representation outperforms six individual sentiment lexica, as well as a straightforward combination thereof.

1 Introduction

Sentiment lexica provide an easy way to automatically label texts with polarity values, and are also frequently transformed into features for supervised models, including neural networks (Palogiannidi et al., 2016; Ma et al., 2018). Indeed, given their utility, a veritable cottage industry has emerged focusing on the design of sentiment lexica. In practice, using any single lexicon, unless specifically and carefully designed for the particular domain of interest, has several downsides. For example, any lexicon will typically have low coverage compared to the language's entire vocabulary, and may have misspecified labels for the domain. In many cases, it may therefore be desirable to combine multiple sentiment lexica into a single representation. Indeed, some research on unifying such lexica has emerged (Emerson and Declerck, 2014; Altrabsheh et al., 2017), borrowing ideas from crowdsourcing (Raykar et al., 2010; Hovy et al., 2013). However, this is a non-trivial task, because lexica can use binary, categorical, or continuous scales to quantify polarity—in addition to different interpretations for each—and thus cannot easily be combined. In Fig. 1, we show an example of the same word labeled using different lexica to illustrate the nature of the challenge.

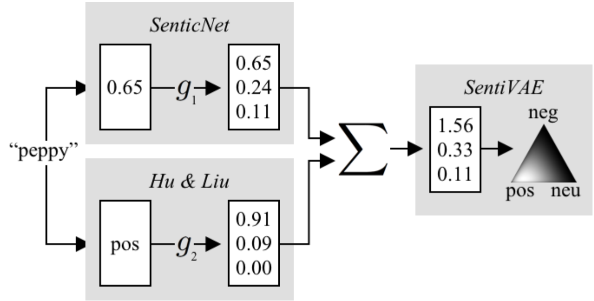


Figure 1: A depiction of the "encoder" portion of SentiVAE. The word *peppy* has polarity values of 0.65 and *pos* in the SenticNet and Hu-Liu lexica, respectively. These values are "encoded" into two three-dimensional vectors, which are then summed and added to $(1, 1, 1)$ (not shown) to form the parameters of a Dirichlet over the latent representation of the word's polarity value.

To combine sentiment lexica with disparate scales, we introduce SentiVAE, a novel multi-view variant of the variational autoencoder (VAE) (Kingma and Welling, 2014). SentiVAE, visualized as a graphical model in Fig. 2, differs from the original VAE in two ways: (i) it uses a Dirichlet latent variable (rather than a Gaussian) for each word in the combined vocabulary, and (ii) it has multiple emission distributions—one for each lexicon. Because the latent variables are shared across the lex-

Proceedings of NAACL-HLT 2019, pages 635–640
Minneapolis, Minnesota, June 2 - June 7, 2019.

Lexicon	Source	N	Dom
`SentiWordNet`	WordNet	14107	$[-1,1]^2$
`MPQA`	Newswire	4397	$\{0,1\}$
`SenticNet`	—	100000	$[-1,1]$
`Hu-Liu`	Product reviews	6790	$\{0,1\}$
`GI`	—	4206	$\{0,1\}$
`VADER`	Social media	7489	$\{0,\ldots,8\}^{10}$

Table 1: Descriptive statistics for the sentiment lexica. N: vocabulary size. Dom: Domain of polarity values.

ica, we are able to derive a common latent representation of the words' polarities. The resulting model is spiritually related to a multi-view learning approach (Sun, 2013), where each view corresponds to a different lexicon. Experimentally, we use SentiVAE to combine six commonly used English-language sentiment lexica with disparate scales.

We evaluate the resulting representation via a text classification task involving nine English-language sentiment analysis datasets. For each dataset, we transform each text into an average polarity value using either our representation, one of the six commonly used sentiment lexica, or a straightforward combination thereof. We then train a classifier to predict the overall sentiment of each text from its average polarity value. We find that our representation outperforms the individual lexica, as well as the straightforward combination for some datasets. Our representation is particularly efficacious for datasets from domains that are not well-supported by standard sentiment lexica.[1]

The existing research that is most closely related to our work is SentiMerge (Emerson and Declerck, 2014), a Bayesian approach for aligning sentiment lexica with different continuous scales. SentiMerge consists of two steps: (i) aligning the lexica via rescaling, and (ii) combining the rescaled lexica using a Gaussian distribution. The authors perform token-level evaluation using a single sentiment analysis dataset where each token is labeled with its contextually dependent sentiment. Because SentiMerge can only combine lexica with continuous scales, we do not include it in our evaluation.

2 Sentiment Lexica and Scales

We use the following commonly used English-language sentiment lexica: SentiWordNet (Baccianella et al., 2010), MPQA (Wilson et al., 2005), SenticNet 5 (Cambria et al., 2014), Hu-Liu (Hu and Liu, 2004), GI (Stone et al., 1962), and VADER (Hutto and Gilbert, 2014). Descriptive statistics for each lexicon are shown in Tab. 1. Each word in SentiWordNet is labeled with two real values, each in the interval $[0,1]$, corresponding to the strength of positive and negative sentiment (e.g., the label $(0\ 0)$ is neutral, while the label $(1\ 0)$ is maximally positive). Each word in VADER is labeled by ten different human evaluators, with each evaluator providing a polarity value on a nine-point scale (where the midpoint is neutral), yielding a 10-dimensional label. MPQA, Hu-Liu, and GI all use binary scales. Lastly, each word in SenticNet is labeled with a real value in the interval $[-1,1]$, where 0 is neutral.

3 SentiVAE

We first describe a figurative generative process for a single sentiment lexicon $d \in \mathcal{D}$, where $\mathcal{D}$ is a set of sentiment lexica. Imagine there is a true (latent) polarity value $\boldsymbol{z}^w$ associated with each word w in the lexicon's vocabulary. When the lexicon's creator labels that word according to their chosen scale (e.g., thumbs-up or thumbs-down, a real value in the interval $[0,1]$), they deterministically transform this true value to their chosen scale via a function $f(\cdot\,;\boldsymbol{\theta}_d)$.[2] Sometimes, noise is introduced during this labeling process, corrupting the label as it leaves the ethereal realm and producing the (observed) polarity label $\boldsymbol{x}_d^w$. They then add this potentially noisy label to the lexicon.

Given a lexicon of observed polarity labels, the latent polarity values can be inferred using a VAE. The original VAE posits a generative model of observed data $\mathcal{X}$ and latent variables $\mathcal{Z}$: $P(\mathcal{X},\mathcal{Z}) = P(\mathcal{X} \mid \mathcal{Z})\,P(\mathcal{Z})$. Inference of $\mathcal{Z}$ then proceeds by approximating the (intractable) posterior $P(\mathcal{Z} \mid \mathcal{X})$ with a Gaussian distribution, factorized over the individual latent variables. A parameterized encoder function compresses $\mathcal{X}$ into $\mathcal{Z}$, while a parameterized decoder function reconstructs $\mathcal{X}$ from $\mathcal{Z}$.

SentiVAE extends the original VAE model to combine multiple lexica with disparate scales, producing a common latent representation of the polarity value for each word in the combined vocabulary.

Generative process. Given a set of sentiment lexica $\mathcal{D}$ with a combined vocabulary $\mathcal{W}$, SentiVAE posits a common latent representation $\boldsymbol{z}^w$ of the polarity value for each word $w \in \mathcal{W}$, where $\boldsymbol{z}^w$ is a three-dimensional categorical distribution over

---

[1] Our representation and code are available at `https://github.com/ahoho/SentiVAE`.

[2] Parameterized by lexicon-specific weights $\boldsymbol{\theta}_d$.

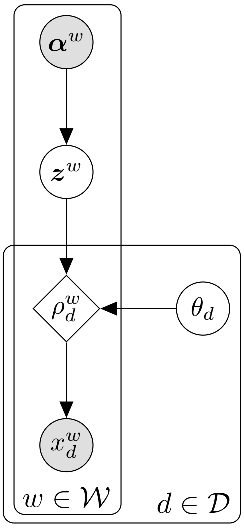


Figure 2: Generative model for SentiVAE.

the sentiments *positive*, *negative*, and *neutral*.

The generative process starts by drawing each latent polarity value $\boldsymbol{z}^w$ from a three-dimensional Dirichlet prior, parameterized by $\boldsymbol{\alpha}^w = (1, 1, 1)$:

$$\boldsymbol{z}^w \sim \text{Dir}(\boldsymbol{\alpha}^w). \tag{1}$$

If the word is uncontroversial,[3] we spur this prior somewhat using the number of lexica in which the word appears $c(w)$. Specifically, we add $c(w)$ to the parameter for the sentiment associated with that word in the lexica, e.g., $\boldsymbol{\alpha}^{\text{SUPERB}} = (1 + c(\text{SUPERB}), 1, 1)$. This has the effect of regularizing the inferred latent polarity value toward the desired distribution over sentiments.

Having generated $\boldsymbol{z}^w$, the process proceeds by "decoding" $\boldsymbol{z}^w$ into each lexicon's chosen scale. First, for each lexicon $d \in \mathcal{D}$, $\boldsymbol{z}^w$ is deterministically transformed via neural network $f(\,\cdot\,; \boldsymbol{\theta}_d)$ with a single 32-dimensional hidden layer, parameterized by lexicon-specific weights $\boldsymbol{\theta}_d$:

$$\boldsymbol{\rho}_d^w = f(\boldsymbol{z}^w; \boldsymbol{\theta}_d). \tag{2}$$

The transformed value $\boldsymbol{\rho}_d^w$ is then used to generate the (observed) polarity label $\boldsymbol{x}_d^w$ for that lexicon:

$$\boldsymbol{x}_d^w \sim P_d(\boldsymbol{x}_d^w \mid \boldsymbol{\rho}_d^w). \tag{3}$$

The dimensionality of $\boldsymbol{\rho}_d^w$ and the emission distribution P_d are lexicon-specific. For SentiWordNet, P_d is a two-dimensional Gaussian with mean $\boldsymbol{\rho}_d^w$ and a diagonal covariance matrix equal to $0.01\boldsymbol{I}$; for VADER, P_d consists of ten nine-dimensional categorical distributions, collectively parameterized by $\boldsymbol{\rho}_d^w$; for MPQA, Hu-Liu, and GI, P_d is a Bernoulli distribution, parameterized by $\boldsymbol{\rho}_d^w$; and for SenticNet, P_d is a univariate Gaussian with mean and variance each an element in a two-dimensional $\boldsymbol{\rho}_d^w$.

Dataset	Source	N	Classes
`IMDB`	Movies	25000	2
`Yelp`	Product reviews	100000	5 / 3
`SemEval`	Twitter	7668	3
`MultiDom`	Product reviews	6500	2
`ACL`	Scientific reviews	248	5 / 3
`ICLR`	Scientific reviews	2166	10 / 3

Table 2: Descriptive statistics for the training portions of the sentiment analysis datasets. N: number of texts.

Inference. Inference involves forming the posterior distribution over the latent polarity values $\mathcal{Z}$ given the observed polarity labels $\mathcal{X}$. Because computing the normalizing constant $P(\mathcal{X})$ is intractable, we instead approximate the posterior with a family of distributions $Q_{\boldsymbol{\lambda}}(\mathcal{Z})$, indexed by variational parameters $\boldsymbol{\lambda}$. Specifically, we use

$$Q_{\boldsymbol{\lambda}}(\mathcal{Z}) = \prod_{w \in \mathcal{W}} Q_{\boldsymbol{\beta}^w}(\boldsymbol{z}^w) = \prod_{w \in \mathcal{W}} \text{Dir}(\boldsymbol{\beta}^w). \tag{4}$$

To construct $\boldsymbol{\beta}^w$, we first define a neural network $g(\cdot; \boldsymbol{\phi}_d)$, with a single 32-dimensional hidden layer, which "encodes" $\boldsymbol{x}_d^w$ into a three-dimensional vector. The output of this neural network is then transformed via a softmax as follows:

$$\boldsymbol{\omega}_d^w = \text{softmax}\big(g(\boldsymbol{x}_d^w; \phi_d)\big) \tag{5}$$

$$\boldsymbol{\beta}^w = 1 + \sum_{d \in \mathcal{D}} \boldsymbol{\omega}_d^w. \tag{6}$$

The intuition behind $\boldsymbol{\beta}_w$ can be understood by appealing to the "pseudocount" interpretation of Dirichlet parameters. Each lexicon contributes exactly one pseudocount, divided among *positive*, *negative*, and *neutral*, to what would otherwise be a symmetric, uniform Dirichlet distribution. As a consequence of this construction, words that appear in more lexica will have more concentrated Dirichlets. Intuitively, this property is appealing.

We optimize the resulting ELBO objective (Blei et al., 2017) with respect to the variational parameters via stochastic variational inference (Hoffman

---

[3] We say that a word is uncontroversial if there is strong agreement across the sentiment lexica in which it appears. Even without this spurring, the inferred latent representation typically separates into the three sentiment classes, but performance on our text classification task is somewhat diminished.

	IMDB 2C	Yelp 5C	Yelp 3C	SemEval 3C	MultiDom 2C	ACL 5C	ACL 3C	ICLR 10C	ICLR 3C
SentiVAE $\mathrm{E}_Q[\boldsymbol{z}^w]$	72.7	49.8	57.5	46.0	70.8	66.7	73.3	92.6	**87.0**
SentiVAE $\boldsymbol{\beta}^w$	73.4	49.7	59.4	**52.2**	74.7	**73.3**	**80.0**	92.6	86.5
SentiWordNet	63.4	36.0	47.6	32.2	62.0	60.0	53.3	89.1	83.5
MPQA	65.4	44.0	53.0	29.9	67.4	60.0	53.3	89.1	83.5
SenticNet	60.5	38.4	43.4	37.2	62.3	60.0	53.3	89.1	83.9
Hu-Liu	67.2	46.6	56.4	31.5	69.4	60.0	53.3	89.1	83.5
GI	58.4	40.7	47.9	31.3	61.6	60.0	53.3	89.1	83.5
VADER	71.7	46.8	59.3	38.5	73.5	66.7	66.7	**94.3**	86.1
Combined	**75.6**	**51.0**	**64.1**	50.6	**75.4**	66.7	66.7	93.9	86.1

Table 3: Classification accuracies for our representation, six lexica, and a straightforward combination thereof.

et al., 2013) using Adam (Kingma and Ba, 2015) in the Pyro framework (Bingham et al., 2018). The standard reparameterization trick used in the original VAE does not apply to models with Dirichlet-distributed latent variables, so we use the generalized reparameterization trick of Ruiz et al. (2016).

4 Experiments and Results

To evaluate our approach, we first use SentiVAE to combine the six lexica described in §2. For each word w in the combined vocabulary, we obtain an estimate of $\boldsymbol{z}^w$ by taking the mean of $Q_{\boldsymbol{\beta}^w}(\boldsymbol{z}^w) = \mathrm{Dir}(\boldsymbol{\beta}^w)$—i.e., by normalizing $\boldsymbol{\beta}^w$. We compare this representation to using $\boldsymbol{\beta}^w$ directly, because $\boldsymbol{\beta}^w$ contains information about SentiVAE's certainty about the word's latent polarity value. We evaluate our common latent representation via a text classification task involving nine English-language sentiment analysis datasets: IMDB (Maas et al., 2011), Yelp (Zhang et al., 2015), SemEval 2017 Task 4 (SemEval, Rosenthal et al. (2017)), multi-domain sentiment analysis (MultiDom, Blitzer et al. (2007)), and PeerRead (Kang et al., 2018) with splits ACL 2017 and ICLR 2017 (Kang et al., 2018). Each dataset consists of multiple texts (e.g., tweets, articles), each labeled with an overall sentiment (e.g., *positive*). Descriptive statistics for each dataset are shown in Tab. 2. For the datasets with more than three sentiment labels, we consider two versions—the original and a version with only three (bucketed) sentiment labels.

For each dataset, we transform each text into an average polarity value using either our representation, one of the six lexica,[4] or a straightforward combination thereof, where the polarity value for each word in the (combined) vocabulary is a 16-dimensional vector that consists of a concatenation of polarity values. (Unlike SentiVAE, this concatenation does not yield a single sentiment lexicon that retains scale coherence, while achieving maximal coverage over words.) Specifically, we replace each token with its corresponding polarity value, and then average the these values (Go et al., 2009; Özdemir and Bergler, 2015; Kiritchenko et al., 2014). We then use the training portion of the dataset to learn a logistic regression classifier to predict the overall sentiment of each text from its average polarity value. Finally, we use the testing portion to compute the accuracy of the classifier.

Results. The results in Tab. 3 show that our representation using $\boldsymbol{\beta}^w$ outperforms the individual lexica for all but one dataset, and that our representation using the mean of $Q_{\boldsymbol{\beta}^w}(\boldsymbol{z}^w)$ outperforms them for six datasets. This is likely because SentiVAE has a richer representation of sentiment than any individual lexicon, and it has greater coverage over words (see Tab. 4). The results in Tab. 5 support the former reason: even when we limit the words in our representation to match those in an individual lexicon, our representation still outperforms the individual lexicon. Unsurprisingly, our representation especially outperforms lexica with unidimensional scales. We also find that our representation outperforms the straightforward combination for datasets from domains that are not well supported by the individual lexica (see Tabs. 1 and 2 for lexicon and dataset sources, respectively). By combining lexica from different domains, our representation captures a general notion of sentiment that is not tailored to any specific domain.

5 Conclusion

We introduced a generative model of sentiment lexica to combine disparate scales into a common

---

[4] We bucket the upper four and lower four points of VADER's nine-point scale, to yield a three-point scale. Without this bucketing, our representation outperforms VADER on four of the nine datasets. We do not bucket VADER when using it in SentiVAE or in the straightforward combination.

	IMDB	SemEval	Multi	ICLR
SentiVAE	70	64	81	71
SentiWordNet	15	14	24	16
MPQA	10	7	18	9
SenticNet	40	39	53	45
Hu-Liu	7	5	13	5
GI	8	7	15	6
VADER	7	6	13	5

Table 4: Coverage over words (percentage) by lexicon for the training portions of four of the nine datasets.

	IMDB 2C		**SemEval 3C**	
	SV	Lex	SV	Lex
SentiVAE	**74.7**	–	**72.4**	–
SentiWordNet	70.6	63.4	67.4	55.1
MPQA	73.5	66.6	62.6	51.8
SenticNet	74.4	60.9	72.1	59.5
Hu-Liu	73.6	68.4	59.1	51.1
GI	71.4	59.3	63.8	54.0
VADER	73.6	73.1	60.9	58.7

Table 5: Classification accuracies for a 10% validation portion of two of the datasets. The first row, labeled SentiVAE, contains the classification accuracy for our representation using β^w. Subsequent (lexicon-specific) rows compare our representation (SV), restricted to the vocabulary of that lexicon, to the lexicon itself (Lex).

latent representation, and realized this model with a novel multi-view variational autoencoder, called SentiVAE. We then used SentiVAE to combine six commonly used English-language sentiment lexica with binary, categorical, and continuous scales. Via a downstream text classification task involving nine English-language sentiment analysis datasets, we found that our representation outperforms the individual lexica, as well as a straightforward combination thereof. We also found that our representation is particularly efficacious for datasets from domains that are not well-supported by standard sentiment lexica. Finally, we note that our approach is more general than SentiMerge (Emerson and Declerck, 2014). While SentiMerge can only combine sentiment lexica with continuous scales, SentiVAE is designed to combine lexica with disparate scales.

Acknowledgements

We would like to thank to Adam Forbes for the design of Fig. 1. We further acknowledge the support of the NVIDIA Corporation with the donation of the Titan Xp GPU used to conduct this research.

References

Nabeela Altrabsheh, Mazen El-Masri, and Hanady Mansour. 2017. Combining Sentiment Lexicons of Arabic Terms. In *AMCIS*. Association for Information Systems.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. 10(2010):2200–2204.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. 2018. Pyro: Deep Universal Probabilistic Programming. *CoRR*, abs/1810.09538.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112:859–877.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. SenticNet 3: A Common and Commonsense Knowledge Base for Cognition-driven Sentiment Analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1515–1521. AAAI Press.

Guy Emerson and Thierry Declerck. 2014. SentiMerge: Combining Sentiment Lexicons in a Bayesian Framework. In *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, pages 30–38. Association for Computational Linguistics and Dublin City University.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. *Processing*, pages 1–6.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John William Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 168–177. ACM.

C. J. Hutto and Eric Gilbert. 2014. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard H. Hovy, and Roy Schwartz. 2018. A Dataset of Peer Reviews (PeerRead): Collection, Insights and NLP Applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1647–1661. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment Analysis of Short Informal Texts. *Journal of Machine Learning Research*, 50:723–762.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM. In *AAAI*, pages 5876–5883. AAAI Press.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Canberk Özdemir and Sabine Bergler. 2015. A Comparative Study of Different Sentiment Lexica for Sentiment Analysis of Tweets. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 488–496. INCOMA Ltd. Shoumen, BULGARIA.

Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Haris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 Task 4: Sentiment Analysis in Twitter Using Semantic-Affective Model Adaptation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 155–163. The Association for Computer Linguistics.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning From Crowds. *Journal of Machine Learning Research*, 11:1297–1322.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Francisco R. Ruiz, Michalis K. Titsias, and David M. Blei. 2016. The Generalized Reparameterization Gradient. In *Advances in Neural Information Processing Systems*, pages 460–468.

Philip J. Stone, Robert F. Bales, J. Zvi Namenwirth, and Daniel M. Ogilvie. 1962. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7(4):484–498.

Shiliang Sun. 2013. A Survey on Multi-view Learning. *Neural Computing and Applications*, 23(7-8):2031–2038.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Enhancing Opinion Role Labeling with Semantic-Aware Word Representations from Semantic Role Labeling

Meishan Zhang[1] and **Peili Liang**[2] and **Guohong Fu**[3]*
1. School of New Media and Communication, Tianjin University, China
2. School of Computer Science and Technology, Heilongjiang University, China
3. Institute of Artificial Intelligence, Soochow University, China
`mason.zms@gmail.com`,
`imlpliang@gmail.com`,
`ghfu@hotmail.com`

Abstract

Opinion role labeling (ORL) is an important task for fine-grained opinion mining, which identifies important opinion arguments such as holder and target for a given opinion trigger. The task is highly correlative with semantic role labeling (SRL), which identifies important semantic arguments such as agent and patient for a given predicate. As predicate agents and patients usually correspond to opinion holders and targets respectively, SRL could be valuable for ORL. In this work, we propose a simple and novel method to enhance ORL by utilizing SRL, presenting semantic-aware word representations which are learned from SRL. The representations are then fed into a baseline neural ORL model as basic inputs. We verify the proposed method on a benchmark MPQA corpus. Experimental results show that the proposed method is highly effective. In addition, we compare the method with two representative methods of SRL integration as well, finding that our method can outperform the two methods significantly, achieving 1.47% higher F-scores than the better one.

1 Introduction

Fine-grained opinion mining aims to detect structured user opinions in text, which has drawn much attention in the natural language processing (NLP) community (Kim and Hovy, 2006; Breck et al., 2007; Ruppenhofer et al., 2008; Wilson et al., 2009; Qiu et al., 2011; Irsoy and Cardie, 2013, 2014; Liu et al., 2015; Wiegand et al., 2016). A structured opinion includes the key arguments of one opinion, such as expressions, holders and targets (Breck et al., 2007; Yang and Cardie, 2012, 2013; Katiyar and Cardie, 2016). Here we focus on opinion role labeling (ORL) (Marasović and Frank, 2018), which identifies opinion holders and targets assuming that the opinion expressions are given. Figure 1 shows an example of the task.

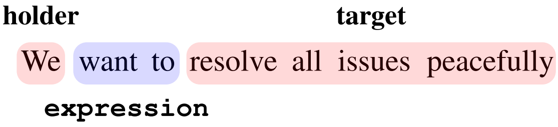


Figure 1: Examples of fine-grained opinion mining.

The focused task behaves very similar with semantic role labeling (SRL) which identifies the core semantic roles for given predicates. Earlier work attempts to exploit a well-trained SRL model to recognize possible semantic roles for a given opinion expression, and then map the semantic roles into opinion roles (Kim and Hovy, 2006; Ruppenhofer et al., 2008). The heuristic approach is unable to obtain high performance for ORL because there are large mismatches between SRL and ORL. For example, opinion expressions are different from verb/noun predicates in SRL, and meanwhile, opinion holders and targets may not always correspond to semantic agents (ARG0) and patients (ARG1), respectively.

We can exploit machine learning based method to solve the mismatching problem between ORL and SRL. With a small number of annotated ORL corpus, we can feed the SRL outputs as inputs to build a statistical model for ORL. By this way, the model can learn the consistencies and inconsistencies between SRL and ORL, arriving at a full exploration of SRL. The method is essentially a feature-based method, treating SRL outputs as a source of features for ORL. The main drawback of the method is that direct exploration of SRL outputs may lead to the error propagation problem. SRL errors can be further propagated into ORL outputs, resulting in degraded ORL performance.

In this work, we propose a simple and novel

---

*Corresponding author.

Proceedings of NAACL-HLT 2019, pages 641–646
Minneapolis, Minnesota, June 2 - June 7, 2019.

method by using implicit semantic-aware word representations from SRL to enhance ORL. The method is referred to as **SRL-SAWR** for brief. Thanks to the recent advances of encoder-decoder neural SRL models (Zhou and Xu, 2015; He et al., 2017), we can extract implicit vectorized features from the intermediate encoder module instead, avoiding the direct exploration of the final one-best SRL outputs. The vectorized features from the encoder part are implicit semantic-aware representations for input sentences. By taking the semantic-aware representations from SRL as ORL inputs, we are able to make use of SRL information and meanwhile alleviate the error propagation problem.

Here we exploit a neural conditional random field (CRF) model with deep bi-directional long short-term memory networks (Bi-LSTMs) as a baseline, most of which is borrowed from Katiyar and Cardie (2016) and Marasović and Frank (2018). Our preliminary experiments show that the model is able to achieve state-of-the-art performances for both ORL and SRL. Based on this model, we study the proposed implicit semantic-aware word representations for ORL. In addition, we compare this method with two other representative methods of SRL integration as well: one uses discrete SRL outputs as features directly for ORL and the other one exploits a multi-task-learning (MTL) framework to benefit ORL by SRL information.

Experiments are conducted on the MPQA 2.0 dataset, which is a standard benchmark for opinion mining. Results show that SRL is highly effective for ORL, which is consistent with previous findings (Kim and Hovy, 2006; Ruppenhofer et al., 2008; Marasović and Frank, 2018). Meanwhile, our implicit SRL-SAWR method can achieve the best ORL performance, 2.23% higher F-scores than the second best method. All the codes and datasets are released publicly available for research purpose under Apache Licence 2.0 at https://github.com/zhangmeishan/SRL4ORL.

2 Method

2.1 Baseline

ORL aims to identify important opinion arguments for a given opinion expression. The task can be modeled as a sequence labeling problem, similar to SRL (Zhou and Xu, 2015; He et al., 2017). We adopt the {BMESO} schema to convert opinion arguments into a sequence of boundary tags for each word, where B, M and E denote the beginning, middle and ending words of an argument, S denotes a single-word argument, and O denotes the other words. Formally, given a sentence $w_1 \cdots w_n$ and a span of opinion expression $w_s \cdots w_e (1 \leq s \leq e \leq n)$, we aim to assign each word in the sentence by a tag, outputting $t_1 \cdots t_n$.

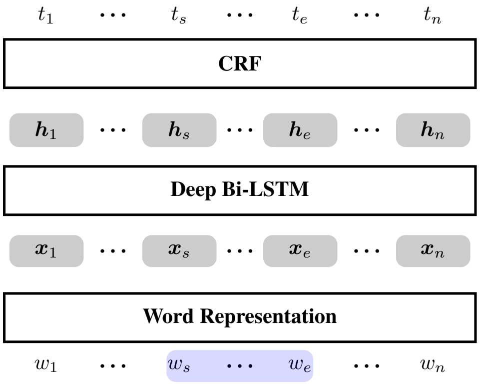


Figure 2: The overall architecture of the baseline.

Inspired by Katiyar and Cardie (2016) and Marasović and Frank (2018), we exploit a deep Bi-LSTM CRF model as the baseline. Figure 2 shows the overall architecture of the baseline model. This model can achieve state-of-the-art performances for both ORL and SRL, which facilitates our study. The key components of the baseline model include three parts: word representation, the deep Bi-LSTM encoder and the CRF decoder. The word representation takes sequential words and opinion expressions as input, mapping them into dense-valued feature vectors $\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$. Following we extract high-level neural features based on the vectors by deep Bi-LSTM, arriving at $\boldsymbol{h}_1 \cdots \boldsymbol{h}_n$. And finally a CRF decoder is applied to output the ORL results $t_1 \cdots t_n$.

2.2 SRL Integration

SRL aims to find the core semantic arguments for a given predicate, which is highly correlative with the ORL task. The semantic roles agent (ARG0) and patient (ARG1) are often corresponding to the opinion holder and target, respectively. Several works even directly transfer semantic roles into opinion roles for ORL (Kim and Hovy, 2006; Ruppenhofer et al., 2008), treating opinion expressions as the major predicates. These systems can achieve good performances, indicating that SRL information can be greatly useful for ORL.

Here we propose a novel method to encode

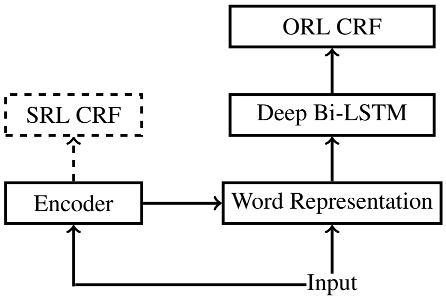


Figure 3: SRL integration methods for ORL.

the SRL information implicitly, enhancing ORL model with semantic-aware word representations from a neural SRL model (SRL-SAWR). Figure 3 shows the overall architectures of our SRL integration method. Instead of using the discrete outputs from the SRL model, the SRL-SAWR method exploits the intermediate encoder outputs as inputs for ORL, which can alleviate the problems in the above two methods. On the one hand, we do not rely on the discrete outputs of a well-trained SRL, reducing the error prorogation problem. And on the other hand, we handle ORL and SRL separately, avoiding the model structure dependencies between the two tasks.

We assume that the external SRL system is a neural-based encoder-decoder model. For fair comparisons with FS-MTL, here we use the same deep Bi-LSTM CRF model for SRL as well. Thus the encoder outputs are the hidden vectors from deep Bi-LSTMs. Assuming that the dumped hidden vector sequence from the SRL encoder is $\boldsymbol{h}_1^{\text{SRL}} \cdots \boldsymbol{h}_n^{\text{SRL}}$, we integrate it into the ORL model by the following equation:

$$\boldsymbol{x}_i^* = \boldsymbol{x}_i \oplus \boldsymbol{W}_{\text{SRL}} \boldsymbol{h}_i^{\text{SRL}}, \qquad (1)$$

where $\boldsymbol{W}_{\text{SRL}}$ is a projection matrix which is a model parameter, $\boldsymbol{x}_i$ is the baseline word representation of word w_i, and $\boldsymbol{x}_i^*$ is the new word representation, which will be further fed into the deep Bi-LSTM layer of the ORL model. Noticeably, the model parameters of the SRL encoder are also fine tuned according to the ORL objective, as the preliminary results indicate that fine-tuning can bring better performance.

3 Experiments

3.1 ORL Data

We exploit the MPQA version 2.0 corpus (Wiebe et al., 2005; Wilson, 2008) to evaluate our models,[1] which has been widely adopted as a benchmark dataset for opinion mining (Yang and Cardie, 2013; Katiyar and Cardie, 2016; Marasović and Frank, 2018). There are 482 documents in the dataset. Following these work, we set aside 132 documents as the development set, and the remaining 350 documents are used as the test set in our experiments. We conduct experiments using five-fold cross-validation (CV) on the test set at the document level. Following Marasović and Frank (2018), we focus on opinion holders and targets only. The gold standard opinion expressions, holders and targets correspond to the direct subjective annotations, agent annotations and target annotations, respectively.

3.2 Evaluation Metrics

We use recall (R), precision (P) and their F1-measure value to measure our proposed models. The average values of the five-fold CV results are reported in this work. We exploit exact matching as the major metric. Following Marasović and Frank (2018), two kinds of soft evaluation methods are also adopted for evaluation, namely binary and proportional overlapping, Binary overlap treats an entity as correct if it contains an overlapped region with the gold-standard entity, and the proportional overlap assigns a partial score proportional to the ratio of the overlapped region.

3.3 Setting

There are several hyper-parameters to define our neural network structures. We simply set their values according to previous work (He et al., 2017; Marasović and Frank, 2018), without much tuning work. Concretely, we set the dimension size of all embeddings to 100, the output hidden size of LSTMs to 200 and the layer number of Bi-LSTM to 3. For external word embeddings, we use the pretrained 100-dimensional glove embeddings (Pennington et al., 2014).

We exploit online training to learn model parameters, and train on the entire training instances for 40 epochs, choosing the best-epoch model according to the performance on the development corpus. We use Adam (Kingma and Ba, 2014) with a learning rate 10^{-3} to update model parameters, and use gradient clipping by a max norm 1.0 and l_2-regularization by a parameter 10^{-8}. We apply dropout with a ratio of 0.2 over word represen-

---

[1] Available at http://www.cs.pitt.edu/mpqa.

tations, output layers of Bi-LSTMs to avoid overfitting (Srivastava et al., 2014).

3.4 SRL

For SRL, we use the large-scale dataset of CoNLL-2012 shared task, which is extracted from OntoNotes v5.0 corpus. The description and separation of train, development and test data set can be found in Pradhan et al. (2013). The training corpus contains over 250K predicates, which is much larger than the number of opinion expressions in the ORL training corpus (averaged 3.6K).

We exploit the same neural network model as the ORL for SRL, in order to make fair comparisons between our proposed model with FS-MTL. According to the preliminary experiments, the SRL model can reach an F-measure of 81.8%, which is comparable to the reported result (81.7%) in He et al. (2017).

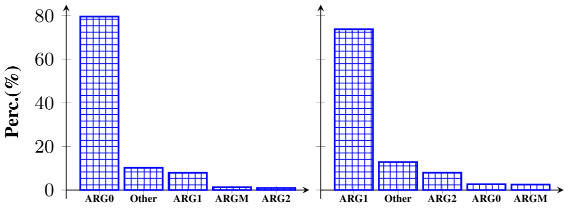
3.5 Results

Table 1 shows the final results on the test dataset. We report the overall as well as the fine-grained performance in term of opinion arguments (i.e., holder and target). Compared with the baseline system, our final SRL-SAWR model can bring significantly better results ($p < 10^{-5}$ under pairwise t-test). For fine-grained evaluations, the final model outperforms the baseline model consistently on opinion holders and targets. The tendencies are similar by exploiting the binary and proportional matching methods. The results show that SRL information is very helpful for ORL, which is consistent with previous studies (Kim and Hovy, 2006; Ruppenhofer et al., 2008; Marasović and Frank, 2018). The implicit SRL-SAWR method is highly effective to integrate SRL information into the ORL model.

Further, we compare the SRL-SAWR method with two other methods as well, namely SRL-TE and FS-MTL, respectively. The SRL-TE approach simply exploits the output SRL tags as inputs for ORL, embedding them as an additional source of word representations. The FS-MTL approach is exactly the proposed model by Marasović and Frank (2018). As shown in Table 1, all three methods can bring improved performance by integrating SRL, further demonstrating that SRL is indeed valuable for ORL. In addition, the SRL-SAWR method can achieve the best performance among the three methods, obtaining further significant improvements by at least $63.74 - 61.51 = 2.23$ points on overall F1-measure with exact matching ($p < 10^{-4}$). For fine-grained evaluations, the SRL-SAWR method can also give the best performance. The results demonstrate that SRL-SAWR is most effective to integrate the SRL information into a neural ORL model. The two methods, SRL-TE and FS-MTL, are comparable by evaluations based on the exact matching.

Model	Holder	Target	Overall
Exact F1			
Baseline	73.07	42.70	58.30
SRL-SAWR	**76.95**	**50.50**	**63.74**
SRL-TE	75.89	46.27	61.46
FS-MTL	75.58	46.40	61.51
Binary F1			
Baseline	81.57	68.34	75.15
SRL-SAWR	**84.91**	**73.29**	**79.10**
SRL-TE	83.47	68.79	76.33
FS-MTL	83.80	72.06	77.87
Proportional F1			
Baseline	79.35	61.22	70.55
SRL-SAWR	**82.82**	**67.31**	**75.08**
SRL-TE	81.56	64.74	72.40
FS-MTL	81.67	65.18	73.61

Table 1: Final results on the test dataset.



(a) Holder (b) Target

Figure 4: Percentages with respect to semantic roles.

3.6 Analysis

In this section, we conduct several experimental analysis on the test dataset to deeply understand the effectiveness of SRL information.

First, we examine the relationship between SRL and ORL. SRL identifies the semantic arguments for predicates, and ORL recognizes the opinion arguments for opinion expressions. Intuitively, in most cases, the opinion holders are corresponding to semantic agents/ARG0 of opinion triggers/expressions, and similarly, the opinion targets are usually corresponding to patients/ARG1. Figure 4 shows the percentages of opinion hold-

Model	Holder	Target	Overall
Baseline	**73.07**	**42.70**	**58.30**
SRL Mapping	68.56	25.33	46.29

Table 2: The performance of the SRL mapping method.

Model	Holder	Target	Overall
Consistent arguments			
Baseline	87.63	61.67	80.87
SRL-SAWR	**88.72**	**67.58**	**82.87**
SRL Mapping	82.57	40.36	63.77
SRL-TE	88.61	63.94	81.88
FS-MTL	88.16	66.80	82.28
Inconsistent arguments			
Baseline	42.90	36.28	38.37
SRL-SAWR	**49.09**	**44.24**	**44.65**
SRL Mapping	0.00	0.00	0.00
SRL-TE	42.30	39.23	40.14
FS-MTL	43.47	39.04	40.25

Table 3: Comparisons in terms of the consistent/inconsistent arguments between SRL and ORL.

ers/targets being corresponding to semantic roles, which are calculated according to the word-level mapping over the 1-best SRL outputs and the gold-standard ORL tags. We list only the five semantic roles with highest mapping percentages. As shown, the results are consistent with our intuition. Thus SRL and ORL are highly correlative. Considering the much larger scale of annotated SRL corpora, SRL can benefit ORL potentially.

According to the above findings, we design a simple system by mapping SRL outputs into ORL directly (Kim and Hovy, 2006; Ruppenhofer et al., 2008). We simply convert the semantic role ARG0 into holder, and ARG1 into target. Table 2 shows the performance. The results of the baseline system are shown for comparison. We can see that the simple mapping method is also one feasible alternative as a whole.

Further, we compare the SRL utilization capabilities of our proposed method and the other SRL-enhanced ORL systems, including the above SRL Mapping method. We categorize the opinion arguments by whether they can be directly mapped from the SRL outputs. The opinion arguments which can be directly mapped from SRL, referred to as consistent arguments, should be more easily identified by SRL enhanced models than the remaining inconsistent arguments. Table 3 shows the comparison results. We can see that all SRL-enhanced supervised models can achieve better performances for consistent arguments. For the inconsistent arguments, the tendency is similar, except the holder performance of SRL-TE. In addition, our method can gain much larger improvements, which indicates that our method can better handle the inconsistencies between SRL and ORL.

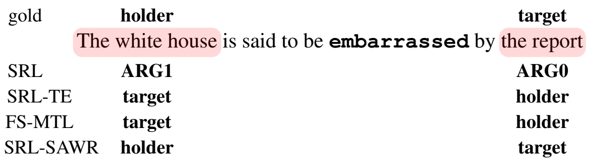


Figure 5: One example for case study.

Finally, we show one case study to illustrate the advantage of our SRL-SAWR method. Figure 5 shows one example. As shown, the SRL argument ARG0, which is more probably mapped onto holder, is annotated by target in the example. The SRL argument ARG1 is labeled as opinion holder, which is also one inconsistent case. Compared with SRL-TE and FS-MTL, our model can better handle these inconsistent cases. The observation further confirms our results in Table 3.

4 Conclusion

We proposed a simple and novel method (SRL-SAWR) to enhance ORL with SRL information by exploiting implicit semantic-aware word representations from SRL. The main idea is to export intermediate SRL encoder outputs as inputs to better word representations of an ORL model. This method does not impose any extra requirement for ORL, and meanwhile avoids the error proropagation problem from discrete SRL outputs. We conducted experiments to verify our method on a benchmark MPQA dataset. The results showed that our method can exploit SRL information effectively. We compared the proposed method with SRL-TE and FS-MTL, which are two representative approaches to enhance ORL by SRL. The results demonstrated our method can bring the best performance among the three approaches.

Acknowledgments

We thank all reviewers for their valuable comments. This work is supported by National Natural Science Foundation of China (NSFC) grants U1836222, 61602160 and 61672211.

References

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *IJCAI*, volume 7, pages 2683–2688.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of ACL)*, pages 473–483.

Ozan Irsoy and Claire Cardie. 2013. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *ACL*, pages 919–929.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, pages 1433–1443.

Ana Marasović and Anette Frank. 2018. Srl4orl: Improving opinion role labeling using multi-task learning with semantic role labeling. In *Proceedings of NAACL*, pages 583–594.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 EMNLP*, pages 1532–1543.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the CoNLL*, pages 143–152.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Josef Ruppenhofer, Swapna Somasundaran, and Janyce Wiebe. 2008. Finding the sources and targets of subjective expressions. In *LREC*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *LRE*, 39(2-3):165–210.

Michael Wiegand, Christine Bocionek, and Josef Ruppenhofer. 2016. Opinion holder and target extraction on opinion compounds – a linguistic approach. In *Proceedings of NAACL*, pages 800–810.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433.

Theresa Ann Wilson. 2008. *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *EMNLP-CoNLL*, pages 1335–1345. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*, pages 1127–1137.

Frowning Frodo, Wincing Leia, and a Seriously Great Friendship: Learning to Classify Emotional Relationships of Fictional Characters

Evgeny Kim and **Roman Klinger**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
`{evgeny.kim,roman.klinger}@ims.uni-stuttgart.de`

Abstract

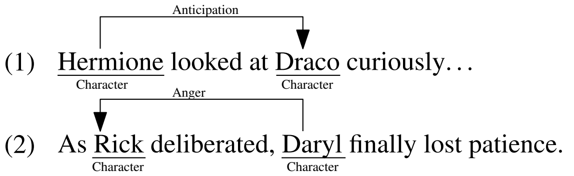
The development of a fictional plot is centered around characters who closely interact with each other forming dynamic social networks. In literature analysis, such networks have mostly been analyzed without particular relation types or focusing on roles which the characters take with respect to each other. We argue that an important aspect for the analysis of stories and their development is the emotion between characters. In this paper, we combine these aspects into a unified framework to classify emotional relationships of fictional characters. We formalize it as a new task and describe the annotation of a corpus, based on fan-fiction short stories. The extraction pipeline which we propose consists of character identification (which we treat as given by an oracle here) and the relation classification. For the latter, we provide results using several approaches previously proposed for relation identification with neural methods. The best result of 0.45 F_1 is achieved with a GRU with character position indicators on the task of predicting undirected emotion relations in the associated social network graph.

1 Introduction

Every fictional story is centered around characters in conflict (Ingermanson and Economy, 2009) which interact, grow closer or apart, as each of them has ambitions and concrete goals (Ackerman and Puglisi, 2012, p. 9). Previous work on computational literary studies includes two tasks, namely social network analysis and sentiment/emotion analysis, both contributing to a computational understanding of narrative structures. We argue that joining these two tasks leverages simplifications that each approach makes when considered independently. We are not aware of any such attempt and therefore propose the task of emotional character network extraction from fictional texts, in which, given a text, a network is to be generated, whose nodes correspond to characters and edges to emotions between characters. One of the characters is part of a trigger/cause for the emotion experienced by the other. Figure 1 depicts two examples for emotional character interactions at the text level. Such relation extraction is the basis for generating social networks of emotional interactions.

Dynamic social networks of characters are analyzed in previous work with different goals, *e.g.*, to test the differences in interactions between various adaptations of a book (Agarwal et al., 2013); to understand the correlation between dialogue and setting (Elson et al., 2010); to test whether social networks derived from Shakespeare's plays can be explained by a general sociological model (Nalisnick and Baird, 2013); in the task of narrative generation (Sack, 2013); to better understand the nature of character interactions (Piper et al., 2017). Further, previous work analyses personality traits of characters (mostly) independently of each other (Massey et al., 2015; Barth et al., 2018; Bamman et al., 2014).

Emotion analysis in literature has focused on the development of emotions over time, abstracting away who experiences an emotion (Reagan et al., 2016; Elsner, 2015; Kim et al., 2017; Piper and Jean So, 2015, *i.a.*). Fewer works have ad-



(1) Hermione looked at Draco curiously. . .

(2) As Rick deliberated, Daryl finally lost patience.

Figure 1: Examples for Emotional Character Interaction. (1) taken from Apryl_Zephyr (2016), (2) from EmmyR (2014). The arrow starts at the experiencer and points at the causing character.

Proceedings of NAACL-HLT 2019, pages 647–653
Minneapolis, Minnesota, June 2 - June 7, 2019.

dressed the annotation of emotion causes, *e.g.*, Neviarouskaya and Aono (2013), Ghazi et al. (2015), Saurí and Pustejovsky (2009), and Kim and Klinger (2018). To the best of our knowledge, there is no previous research that deals with emotional relationships of literary characters. The works that are conceptually the closest to our paper are Chaturvedi et al. (2017) and Massey et al. (2015), who use a more general set of relationship categories.

Most approaches to emotion classification from text build on the classes proposed by Plutchik (2001) and Ekman (1992). Here, we use a discrete emotion categorization scheme based on fundamental emotions as proposed by Plutchik. This model has previously been used in computational analysis of literature (Mohammad, 2012, *i.a.*). We refer the reader to social psychology literature for more details on the emotional relationship between people (Burkitt, 1997; Gaelick et al., 1985).

The main contributions of this paper are (1) to propose the new task of emotional relationship classification of fictional characters, (2) to provide a fan-fiction short story corpus annotated with characters and their emotional relationships, and (3) to provide results for relation extraction models for the task. We evaluate our models on the textual and the social network graph level and show that a neural model with positional indicators for character roles performs the best. An additional analysis shows that the task of character relationship detection leads to higher performance scores for polarity detection than for more fine-grained emotion classes. Differences between models are minimal when the task is cast as a polarity classification but are striking for emotion classification.

This work has potential to support a literary scholar in analyzing differences and commonalities across texts. As an example, one may consider Goethe's *The Sorrows of Young Werther* (Goethe, 1774), a book that gave rise to a plethora of imitations by other writers, who attempted to depict a similar love triangle between main characters found in the original book. The results of our study can potentially be used to compare the derivative works with the original (see also Barth et al., 2018).

2 Corpus

Data Collection and Annotation. Each emotion relation is characterized by a triple $(C_{\text{exp}}, e, C_{\text{cause}})$, in which the character C_{exp} feels the emotion e (mentioned in text explicitly or implicitly). The character C_{cause} is part of an event which triggers the emotion e. We consider the eight fundamental emotions defined by Plutchik (2001) (anger, fear, joy, anticipation, trust, surprise, disgust, sadness). Each character corresponds to a token sequence for the relation extraction task and to a normalized entity in the graph depiction.

Using WebAnno (Yimam et al., 2013), we annotate a sample of 19 complete English fan-fiction short stories, retrieved from the Archive of Our Own project[1] (due to availability, the legal possibility to process the texts and a modern language), and a single short story by Joyce (1914) (Counterparts) being an exception from this genre in our corpus. All fan-fiction stories were marked by the respective author as complete, are shorter than 1500 words, and depict at least four different characters. They are tagged with the keywords "emotion" and "relationships".

The annotators were instructed to mark every character mention with a canonical name and to decide if there is an emotional relationship between the character and another character. If so, they marked the corresponding emotion phrase with the emotion labels (as well as indicating if the emotion is amplified, downtoned or negated). Based on this phrase annotation, they marked two relations: from the emotion phrase to the experiencing character and from the emotion phrase to the causing character (if available, *i.e.*, C_{cause} can be empty). One character may be described as experiencing multiple emotions.

We generate a "consensus" annotation by keeping all emotion labels by all annotators. This is motivated by the finding by Schuff et al. (2017) that such high-recall aggregation is better modelled in an emotion prediction task. The data is available at http://www.ims.uni-stuttgart.de/data/relationalemotions.

Inter-Annotator Agreement We calculate the agreement along two dimensions, namely unlabelled vs. labeled and instance vs. graph-level. Table 1 reports the pairwise results for three annotators. In the *Inst. labelled* setting, we accept an instance being labeled as true positive if both annotators marked the same characters as experiencer and cause of an emotion and classified their in-

---

[1] https://archiveofourown.org

	a1–a2	a1–a3	a2–a3
Inst. labelled	24	19	24
Inst. unlab.	33	27	29
Graph labelled	66	69	66
Graph unlabelled	90	93	92

Table 1: F_1 scores in % for agreement between annotators on different levels. a1, a2, and a3 are different annotators.

teraction with the same emotion. In the *Inst. unlabelled* case, the emotion label is allowed to be different. On the graph level (*Graph labelled* and *Graph unlabelled*), the evaluation is performed on an aggregated graph of interacting characters, *i.e.*, a relation is accepted by one annotator if the other annotator marked the same interaction somewhere in the text. We use the F_1 score to be able to measure the agreement between two annotators on the span levels. For that, we treat the annotations from one annotator in the pair as correct and the annotations from the other as predicted.

As Table 1 shows, agreement on the textual level is the lowest with values between 19 and 33 % (depending on the annotator pair), which also motivated our aggregation strategy mentioned before. The values for graph-labelled agreement are more relevant for our use-case of network generation. The values are higher (66–93 %), showing that annotators agree when it comes to detecting relationships regardless of where exactly in the text they appear.

Statistics. Table 2 summarizes the aggregated results of the annotation. The column "All" lists the number of experiencer annotations (with an emotion), the column "Rel." refers to the counts of emotion annotations with both experiencer and cause.

Joy has the highest number of annotated instances and the highest number of relationship instances (413 and 308 respectively). In contrast, *sadness* has the lowest number of annotations with a total count of instances and relations being 97 and 64 respectively. Overall, we obtain 1335 annotated instances, which we use to build and test our models.

3 Methods

Figure 2 depicts the process flow for each of the models. We distinguish between *directed* and *undirected* relation prediction. In the directed scenario, we classify which character is the experiencer and which character is the cause, as well as what is the emotion between two characters. For the undirected scenario, we only classify the emotion relation between two characters. We do not tackle character name recognition here: our models build on top of gold character annotations.

Emotion	All	Rel.
anger	258	197
anticipation	307	239
disgust	163	122
fear	182	120
joy	413	308
sadness	97	64
surprise	143	129
trust	179	156
total	**1742**	**1335**

Table 2: Statistics of emotion and relation annotation. "All" indicates the total number of emotion annotations. "Rel." indicates total number of emotional relationships (including a causing character) instantiated with the given emotion.

Indicator	Implementation example
No-Ind.	Alice is angry with Bob
Role	`<e>`Alice`</e>`...`<c>`Bob`</c>`
MRole	`<e>`...`<c>`
Entity	`<et>`Alice`</et>`...`<et>`Bob`</et>`
MEntity	`<et>`...`</et>`

Table 3: Different indicators applied to the same instance. *No-Ind.* means no positional indicators are added. *M* in *MRole* and *MEntity* means that the name of the character is masked. Tag `<e>` indicates the experiencer. Tag `<c>` indicates the cause. Tag `<et>` indicates an entity.

The *baseline* model predicts the emotion for a character pair based on the NRC dictionary (Mohammad and Turney, 2013). It accepts the emotion associated with the words occurring in a window of n tokens around the two characters, with n being a parameter set based on results on a development set for each model (see supplementary material for more details).

Further we cast the relation detection as a machine learning-based classification task, in which each classification instance consists of two character mentions with up to n tokens context to the

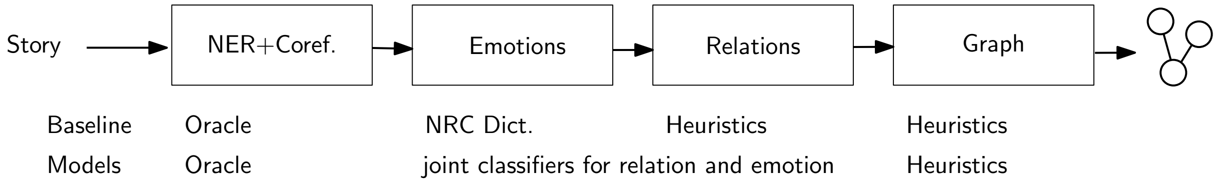


Figure 2: Models for the emotional relationship prediction. Oracle: a set of character pairs from the gold data.

left and to the right of the character mentions. We compare an extremely randomized tree classifier with bag-of-words features (Geurts et al., 2006) (*BOW-RF*) with a two-layer GRU neural network (Chung et al., 2014) with max and averaged pooling. In the latter, we use different variations of encoding the character positions with indicators (inspired by Zhou et al. (2016), who propose the use of positional indicators for relation detection). Our variations are exemplified in Table 3. Note that the case of predicting directed relations is simplified in the "Role" and "MRole" cases in contrast to "Entity" and "MEntity", as the model has access to gold information about the relation direction.

We obtain word vectors for the embedding layer from GloVe (pre-trained on Common Crawl, $d = 300$, Pennington et al., 2014) and initialize out-of-vocabulary terms with zeros (including the position indicators).

4 Experiments

Experimental Setting. In the classification experiments, we compare the performance of our models on different label sets. Namely, we compare the complete emotion set with 8 classes to a 5 class scenario where we join *anger* and *disgust*, *trust* and *joy*, as well as *anticipation* and *surprise* (based on preliminary experiments and inspection of confusion matrices). The 2-class scenario consists of positive (*anticipation, joy, trust, surprise*) and negative relations (*anger, fear, sadness, disgust*). For each set of classes, we consider a setting where directed relations are predicted with one where the direction is ignored. Therefore, in the *directed* prediction scenario, each emotion constitutes two classes to be predicted for both possible directions (therefore, 16, 10, and 4 labels exist).

The evaluation is performed with precision, recall and F_1 in a cross-story validation setting, in which each story is used as one separate test/validation source. For model selection and meta-parameter optimization, we use 50 % randomly sampled annotations from this respective test/validation instance as a validation set and the remainder as test data.

Further, we evaluate on three different levels of granularity: Given two character mentions, in the instance-level evaluation, we only accept the prediction to be correct if exactly the same mention has the according emotion annotation. We then aggregate the different true positive, false positive and false negative values across all stories before averaging to an aggregated score (similar to micro-averaging). On the story-level, we also accept a prediction to be a true positive the same way, but first calculate the result P/R/F_1 for the whole story before averaging (similar to macro-averaging). On the graph-level, we accept a prediction for a character pair to be correct without considering the exact position.

Results. Table 4 shows the results (precision and recall shown in supplementary material) on development data and independent test data for the best models. The GRU+MRole model achieves the highest performance with improvement over BOW-RF on the instance and story levels, and shows a clear improvement over the GRU+NoInd. model in the directed 8-class setting. GRU+Role achieves the highest performance on the graph level in the directed 8-class setting. In the undirected prediction setting, all models perform better in the 5-class experiment and 2-class experiment than in 8-class experiment. This is not always the case for the directed prediction, where some models perform better in 8-class experiment (GRU+NoInd., GRU+Entity, BOW-RF).

We observe that the difference in F_1 score between the baseline, bag-of-words model and our GRU models in a 2-class experiment is marginal. This may be an indicator that the binary representation harms the classification of emotional relations between characters, as they can be nuanced and do not always perfectly map to either positive and negative classes. On the other side, a more sophisticated classification approach is necessary to capture these nuanced differences.

		Undir.			Directed		
	Model	8c	5c	2c	8c	5c	2c
Dev Instance level	Baseline	24	30	56	–	–	–
	BOW-RF	18	31	56	20	19	35
	GRU+NoInd.	31	39	64	26	23	37
	GRU+Role	19	35	55	33	34	57
	GRU+MRole	30	44	67	38	44	65
	GRU+Entity	20	34	58	23	19	30
	GRU+MEntity	30	43	65	28	29	40
Dev Story level	Baseline	24	31	56	–	–	–
	BOW-RF	21	35	58	22	20	38
	GRU+NoInd.	33	41	66	25	23	38
	GRU+Role	19	34	55	33	35	56
	GRU+MRole	32	44	67	39	44	65
	GRU+Entity	21	31	57	22	18	30
	GRU+MEntity	33	46	65	28	30	39
Dev Graph-level	Baseline	31	46	65	–	–	–
	BOW-RF	27	36	71	34	34	54
	GRU+NoInd.	44	55	73	35	33	54
	GRU+Role	35	49	65	41	43	57
	GRU+MRole	45	58	73	40	48	65
	GRU+Entity	37	50	68	39	29	49
	GRU+MEntity	47	63	73	39	39	52
Test	GRU+MRole Inst.	30	44	64	38	43	65
	GRU+MRole Story	33	45	65	39	43	66
	GRU+MRole Graph	45	59	71	42	49	66

Table 4: Cross-validated results in % F_1 score, average of four runs. Inst. level: aggregated over all instances in the dataset. Story level: averaged performance on all stories. Graph-level: averaged performance on graph level on all stories. Test results are reported for the best indicator type. See Table 3 for the examples of the indicator implementation.

As expected, we observe a better performance on a graph level for all models, with the highest performance of 47 % F_1 (GRU+MEntity), 63 % F_1 (GRU+MEntity), and 73 % F_1 (GRU+MRole, GRU+MEntity, GRU+NoInd.) in undirected 8-, 5-, and 2-class experiments, respectively, on the development set. In the directed scenario, the highest performances are 41 % F_1 (GRU+Role), 48 % F_1 (GRU+MRole), and 65 % F_1 (GRU+MRole).

The results show that the sequential and embedding information captured by a GRU as well as additional positional information are all relevant for a substantial performance, at least on the fine-grained emotion prediction task.

5 Conclusion & Future Work

In this paper, we formulated the new task of emotional character network extraction from fictional texts. We argued that joining social network analysis of fiction with emotion analysis leverages simplifications that each approach makes when considered independently. We presented a publicly available corpus of fan-fiction short stories annotated with character relations and proposed several relation classification models. We showed that a recurrent neural architecture with positional indicators leads to the best results of relation classification. We also showed that differences between different machine learning models with binary mapping of emotion relation is almost leveled. This may suggest that emotion relation classification is best modeled in a multi-class setting, as emotional interactions of fictional characters are nuanced and do not simply map to either a positive or a negative class.

For future work we propose to develop a real-world application pipeline in which character pairs are not given by an oracle, but rather extracted from text automatically using named entity recognition. To better understand the relation between instance and graph levels, we propose to explore the best strategy for edge labeling either by a majority vote or accepting the edges with the highest confidence scores. Further, modeling the task in an end-to-end learning setting from text to directly predict the graph, in the spirit of multi-instance learning, is one of the next steps. To that end, we suggest obtaining more gold data with character relations and optimize the pipeline towards the best performance on additional data.

Acknowledgements

This research has been conducted within the CRETA project (http://www.creta.uni-stuttgart.de/) which is funded by the German Ministry for Education and Research (BMBF) and partially funded by the German Research Council (DFG), projects SEAT (Structured Multi-Domain Emotion Analysis from Text, KL 2869/1-1). We thank Laura-Ana-Maria Bostan and Heike Adel for fruitful discussions.

References

Angela Ackerman and Becca Puglisi. 2012. *The Emotion Thesaurus: A Writer's Guide to Character Expression*. JADD Publishing.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1202–1208. Asian Federation of Natural Language Processing.

Apryl_Zephyr. 2016. Friends. https://archiveofourown.org/works/8081986.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379. Association for Computational Linguistics.

Florian Barth, Evgeny Kim, Sandra Murr, and Roman Klinger. 2018. A reporting tool for relational visualization and analysis of character mentions in literature. In *Book of Abstracts – Digital Humanities im deutschsprachigen Raum*, pages 123–126, Cologne, Germany.

Ian Burkitt. 1997. Social relationships and emotions. *Sociology*, 31(1):37–55.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daume III. 2017. Unsupervised learning of evolving relationships between literary characters. In *AAAI Conference on Artificial Intelligence*, pages 3159–3165.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the Deep Learning and Representation Learning Workshop: NIPS 2014*.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Micha Elsner. 2015. Abstract representations of plot structure. *LiLT (Linguistic Issues in Language Technology)*, 12:1–29.

David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden. Association for Computational Linguistics.

EmmyR. 2014. PianoP. https://archiveofourown.org/works/2481311.

Lisa Gaelick, Galen V. Bodenhausen, and Robert S. Wyer. 1985. Emotional communication in close relationships. *Journal of Personality and Social Psychology*, 49(5):1246.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Computational Linguistics and Intelligent Text Processing*, pages 152–165, Cham. Springer International Publishing.

Johann Wolfgang von Goethe. 1774. Die Leiden des jungen Werthers.

Randy Ingermanson and Peter Economy. 2009. *Writing fiction for dummies*. John Wiley & Sons.

James Joyce. 1914. *Dubliners*. Grant Richards.

Evgeny Kim and Roman Klinger. 2018. Who feels what and why? annotation of a literature corpus with semantic roles of emotions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1345–1359, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Evgeny Kim, Sebastian Padó, and Roman Klinger. 2017. Investigating the relationship between literary genres and emotional plot development. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 17–26, Vancouver, Canada. Association for Computational Linguistics.

Philip Massey, Patrick Xia, David Bamman, and Noah A Smith. 2015. Annotating character relationships in literary texts. *arXiv preprint arXiv:1512.00728*.

Saif M Mohammad. 2012. From once upon a time to happily ever after: Tracking emotions in mail and books. *Decision Support Systems*, 53(4):730–741.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Eric T Nalisnick and Henry S Baird. 2013. Extracting sentiment networks from shakespeare's plays. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 758–762. IEEE.

Alena Neviarouskaya and Masaki Aono. 2013. Extracting causes of emotions from text. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 932–936, Nagoya, Japan. Asian Federation of Natural Language Processing.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Andrew Piper, Mark Algee-Hewitt, Koustuv Sinha, Derek Ruths, and Hardik Vala. 2017. Studying literary characters and character networks. In *Digital Humanities 2017: Conference Abstracts*, Montreal, Canada.

Andrew Piper and Richard Jean So. 2015. Quantifying the weepy bestseller. *New Republic*. https://newrepublic.com/article/126123/quantifying-weepy-bestseller.

Robert Plutchik. 2001. The nature of emotions. *American Scientist*, 89(4):344–350.

Andrew J Reagan, Lewis Mitchell, Dilan Kiley, Christopher M Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31.

Graham Alexander Sack. 2013. Character Networks for Narrative Generation: Structural Balance Theory and the Emergence of Proto-Narratives. In *2013 Workshop on Computational Models of Narrative*, volume 32 of *OpenAccess Series in Informatics (OASIcs)*, pages 183–197, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227.

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

Generalizing Unmasking for Short Texts

Janek Bevendorff * **Benno Stein** * **Matthias Hagen** † **Martin Potthast** ‡

*Bauhaus-Universität Weimar
†Martin-Luther-Universität Halle-Wittenberg
‡Leipzig University

`<first>.<last>@uni-{weimar, leipzig}.de`
`<first>.<last>@informatik.uni-halle.de`

Abstract

Authorship verification is the problem of inferring whether two texts were written by the same author. For this task, unmasking is one of the most robust approaches as of today with the major shortcoming of only being applicable to book-length texts. In this paper, we present a generalized unmasking approach which allows for authorship verification of texts as short as four printed pages with very high precision at an adjustable recall tradeoff. Our generalized approach therefore reduces the required material by orders of magnitude, making unmasking applicable to authorship cases of more practical proportions. The new approach is on par with other state-of-the-art techniques that are optimized for texts of this length: it achieves accuracies of 75–80 %, while also allowing for easy adjustment to forensic scenarios that require higher levels of confidence in the classification.

1 Introduction

With advances in computational stylometry, determining the original authorship of unknown literary publications can be accomplished with near certainty by state-of-the-art authorship verification. If the source material is abundant and sufficiently many known publications exist, linking them together is hardly a challenge. But the playing field changes entirely if only fragments are available for verification, either because few known works of an author exist or because the text to be verified is only a few pages long. With classification results significantly above chance, yet far below certainty, verification approaches struggle to produce reliable results in short-text scenarios and thus lack real-world practicality for material far below book length. Even the unmasking approach by Koppel and Schler (2004), which otherwise proved to be one of the most ingenious and robust verification approaches, fails in this setting and can only deliver below-average performance compared to more specialized verification systems, which still display high uncertainty themselves with an error of up to 25 % (Stamatatos et al., 2015). At PAN 2015, individual texts of the English-language dataset had an average size of about 1.5 kB (less than 400 words), so it does not come as a surprise that none of the participants employed unmasking.

To tackle the uncertainty problem of authorship verification on short texts, we propose a generalized unmasking approach which prioritizes precision so as to verify authorship with reliable results while rejecting cases of low certainty. We also present a new open-source general-purpose unmasking framework as a highly-customizable implementation of our approach.[1]

2 Related Work

Authorship analysis has been practiced since the late 19th century (Bourne, 1897). Although mostly the narrower task of authorship attribution has been considered, where texts are attributed to a set of given authors, recently, authorship verification has been proposed as a more fundamental task. For long texts, unmasking by Koppel and Schler (2004) has since been established as a gold standard. Sanderson and Guenter (2006) showed that its performance is far worse for short texts, though, whereas their own model produced acceptable results with a minimum of 5,000 words per training text. Newer research has emerged through a series of shared tasks at PAN (Juola and Stamatatos, 2013; Stamatatos et al., 2014, 2015), which focused on shorter texts and achieved higher scores than unmasking. In general, however, the verification problem could not be solved to a point where results are highly reliable. The winner of the shared task at PAN 2015 (Bagnall, 2015) achieved an accuracy of 76 %, thus delivering a false decision in one in four cases while being unclear about which 24 %

---

[1] Code and data: https://github.com/webis-de/NAACL-19.

Proceedings of NAACL-HLT 2019, pages 654–659
Minneapolis, Minnesota, June 2 - June 7, 2019.

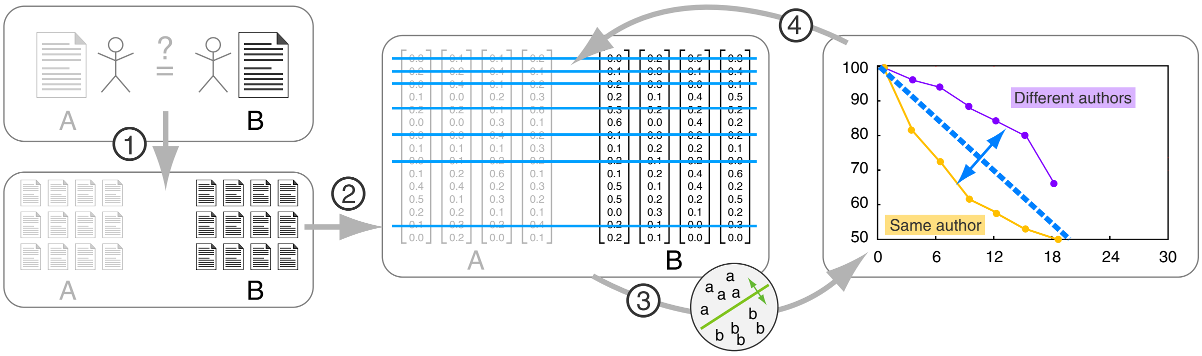


Figure 1: Schematic of the unmasking algorithm. Steps 1-4 are described in the text below. Dependent on whether the authors of texts A and B are the same or different, accuracy curves as exemplified can be expected.

are most likely to be incorrect. If applied in a real-world forensic scenario, such a verifier might give only hints as to whether two texts share authorship. Verification approaches based on text compression (Teahan and Harper, 2003; Khmelev and Teahan, 2003; Halvani et al., 2017) have been proposed, whose latest incarnations proved at least competitive to the approaches developed during the shared tasks; the state of the art in short-text authorship verification achieves around 75–80 % accuracy.

3 Unmasking for Short Texts

After reviewing the original unmasking algorithm, we introduce our generalization for short texts.

3.1 The Original Unmasking Algorithm

Unmasking as per Koppel and Schler (2004) is based on the idea that the style of texts from the same author differs only in a few superficial features. By iteratively removing these most discriminating style features, one can measure the "speed" at which cross-validation accuracy between sets of chunks of the two texts degrades. For texts written by the same author, the accuracy tends to decrease faster than otherwise. Combining the obtained accuracy values into curves for each pair, a meta classifier can be trained on the curves to determine the class of a pair (same / different author).

Koppel and Schler evaluated their approach on a corpus of 21 books (each at least 500 kB) by 10 different authors. The task was to verify for each book A whether it has been written by a given author, using all the latter's books B for an author profile, except book A, in case it was the same author. As described in their paper, the unmasking algorithm works as follows (see Figure 1):

1. From either text, create non-overlapping chunks of at least 500 words length without splitting paragraphs.
2. Use the 250 words with highest average frequency in A and B as features.
3. Obtain 10-fold cross-validation accuracy between A and B with a linear SVM kernel.
4. Eliminate the 3 highest positive and negative features for the model trained in each fold.
5. Go to Step 3 if there are features left.

The declining cross-validation accuracy values from curves on which a meta classifier is trained. Koppel and Schler used another SVM as the meta classifier, utilizing as features the curve points, the curves' point-wise first- and second-order derivatives, and the derivatives sorted by steepest point-wise drop. With this approach, they achieved a verification accuracy of over 95 %.

3.2 Generalization for Short Texts

While impressive as such, the performance of unmasking hinges on the availability of sufficiently many chunks per text, where each chunk has to be of at least the aforementioned 500 words length, or else the training data becomes too sparse and no descriptive curves can be generated. Short texts have the inherent problem that not many chunks can be extracted by cutting them into pieces.

To generate more training samples from short texts, one method would be to generate overlapping chunks, but this only ends in many almost identical chunks and provides only a marginal performance boost. Instead, we exploit the bag-of-words nature of the unmasking features and create the chunks by oversampling words in a bootstrap aggregating manner. We treat each text as a random pool of words from which we can draw without replacement to fill up a chunk. Once the pool is exhausted, we replenish it and draw again until we have generated a sufficient number of chunks. This is to guarantee that each word is drawn at least once.

Employing this bagging approach alone will not yield satisfying results, however. The curves will be quite random with high variance. To counteract this, we run unmasking on the generated chunks multiple times and average the curves to get smoother and more reproducible results. Our generalized unmasking algorithm works as follows:

1. From either text, create 30 chunks counting 700 words each by random chunk generation.
2. Use the 250 words with highest average frequency in A and B as features.
3. Obtain 10-fold cross-validation accuracy between A and B with a linear SVM kernel.
4. Eliminate the on average 5 most significant positive and negative features across folds (resulting in a total of 10 removals).
5. Go to Step 3 if there are still features left.

Another linear SVM classifier is trained on these training curves, their central-difference gradients (first- and second-order), as well as their gradients sorted by magnitude. This classifier is then used to classify curves generated in the same fashion from text pairs in the test set.

4 Evaluation

The data we use for our experiments is a collection of 180 text pairs (consisting of 90 *same-author* and 90 *different-authors* cases) for training and a similar set of 80 text pairs (consisting of 40 *same-author* and 40 *different-authors* cases) for testing. The texts were obtained from Project Gutenberg, comprise about 4,000 words each (23,000 characters), and were written by a total of 390 unique English-language authors. We took special care to select texts of similar genre and publication period to avoid accidental topic classification.

Since short-text authorship verification presents itself as such a difficult problem, it becomes all the more important to find a good measure for assessing the quality of a verifier. Due to the high uncertainty of many results, a standard two-class accuracy measure is not an optimal choice. Instead of trying to develop perfect verifiers, we can already build more useful tools today by optimizing for precision and sacrificing recall. Unfortunately, this approach—although more useful in general—does not perform well in a setting where accuracy is measured. In order to provide a more suitable evaluation quantity, PAN adopted the c@1 measure by Peñas and Rodrigo (2011):

$$\frac{1}{n} \cdot \left(n_{ac} + \frac{n_{ac}}{n} \cdot n_u \right) ,$$

where n denotes the number of problems, n_{ac} the number of correct answers and n_u the number of non-answers. The c@1 measure solves the uncertainty problem by rewarding non-answers in that it assigns them the same accuracy as the rest of the problems. All-correct answers still yield a score of 1, all-wrong or completely unanswered problem sets a score of 0. Hence, with this measure, a verifier is at liberty not to answer a given problem in case of doubt and still receive a reasonably good score, even if its overall sensitivity (or recall) is low. Most PAN participants did not exploit the c@1 measure to a larger extent, so c@1 scores of their submitted approaches are roughly the same as their accuracy (within margin of a few percent).

A problem with c@1 is that it is still designed for binary classification with equal weights for both classes. If we are primarily interested in whether two texts were written by the same author, but do not need a reliable decision in the other case, c@1 does not serve us well. For that reason, we propose as an alternative the $\mathrm{F}_{0.5}$ measure where we treat non-answers as false negatives:

$$\frac{(1 + 0.5^2) \cdot n_{tp}}{(1 + 0.5^2) \cdot n_{tp} + 0.5^2 \cdot (n_{fn} + n_u) + n_{fp}} ,$$

with n_{tp} denoting the number of true positives, n_{fn} the number of false negatives, and n_{fp} the number of false positives. As before, n_u is the number of unanswered problems. The parameter $\beta = 0.5$ of the F measure was chosen so as to weigh precision substantially higher than recall without diminishing its contribution entirely. Other values can be used depending on individual use cases. We call this specialized $\mathrm{F}_{0.5}$ measure $\mathrm{F}_{0.5\mathrm{u}}$.

4.1 Unmasking's Hyperparameters

The hyperparameters of unmasking have a direct and intuitive effect on its output. The most important hyperparameters are the number of chunks, the number of words per chunk, the size of the feature vectors, the number of feature removals per round, and the total number of averaged unmasking runs.

The degradation speed or curve slope is determined primarily by the number of chunks generated (more chunks result in generally shallower curves) and the amount of information lost in each round (shorter feature vectors or more removals cause the curves to plummet). Both parameters also control

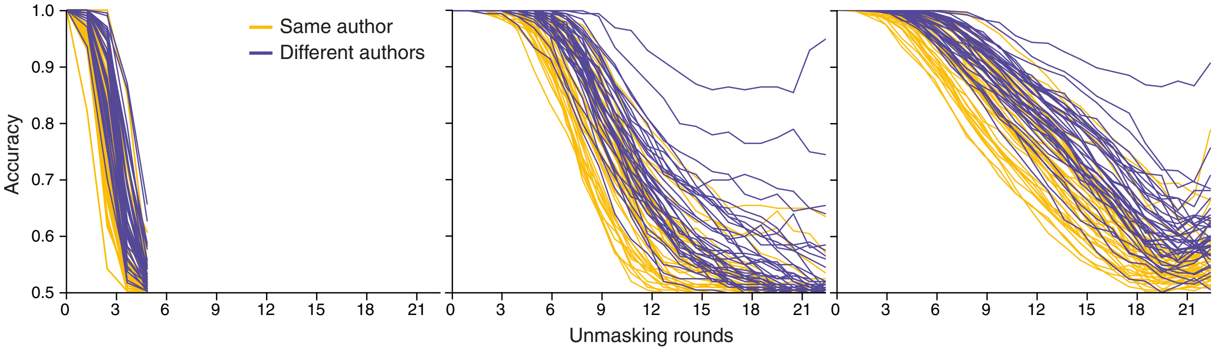


Figure 2: Unmasking curves on the test corpus with three different sets of hyperparameters. Each curve represents one text pair. *Left:* 100 features, 20 eliminations per round, 10 chunks of 1,000 words each. *Center:* 250 features, 10 eliminations, 10 chunks of 700 words each. *Right:* 250 features, 10 eliminations, 30 chunks of 700 words each. All curves are averages of 10 runs. Few features and chunks and many removals in a single round lead to curves dropping too quickly in straight lines. The center and right-hand configurations perform much better.

the granularity of the curves. Longer feature vectors and more chunks result in smoother curves, whereas very short vectors and very few chunks create larger straight-line segments. It is vital to find a good middle ground for both these parameters or else the curves will either altogether drop too quickly with all significant features eliminated in a single round, or runtime is wasted without benefit. Particularly removals and vector size need to be balanced so as to optimally capture enough of the highly significant features and not only the tail of the distribution. Furthermore, curves with insufficient granularity contain a lot less information and are often harder to distinguish, regardless of their slope. The number of words per chunk, on the other hand, appears to be mostly responsible for the slope of the first few initial rounds with larger chunks delaying the first major drop. Overall, this parameter's influence seems rather low, probably because we are only repeatedly sampling from an already small sample of the same distribution. Finally, the number of unmasking runs that are averaged smooths noisy curves and compensates for outliers. Here, more is generally better, but with diminishing returns beyond a certain point.

The hyperparameters given in Section 3 result from cross-validating several hundred configurations to find the most useful value ranges on our training data. Optimal choices can vary slightly between datasets, which leaves room for improvement. Figure 2 shows the differences between a select number of hyperparameter choices. We found that between 25 and 50 chunks, vector sizes of 250 to 400 features, and not fewer than 5, yet no more than 20 removals per round are necessary to achieve sufficient curve granularity and satisfying gradients. For chunk sizes, a large range between 300 and 1,000 words seems possible, though we found sizes of 500–700 words to work best without adding too much computational overhead. For stable classification, about 10 total runs need to be averaged, while 15–20 are still a sensible choice.

4.2 Verification Results

The overall performance of our approach is on par with other state-of-the-art verifiers, but we can improve our precision significantly by increasing the minimal distance a pair must have from the SVM hyperplane to be classified at all. We use this distance threshold as our confidence parameter c. Classification results at different values for c are shown in Table 1. At a threshold of 0.8, we can derive an answer for only 13.8 % of the cases, but are able to do so with a precision of 1.0, meaning all *same-author* classifications are correct. We can reduce the threshold down to a value of 0.4, at which we can classify about half the cases with a still all-correct answer set—but with lower confidence in the correctness. The actual numbers vary within a certain margin at low thresholds due to our random chunk generation. Thresholds below 0.4 will produce an increasing amount of false positives. The choice of c depends on the assurance level desired by the user. In medium- to high-assurance scenarios (where false positives are to be avoided, but not entirely critical), we recommend a threshold of 0.6 or higher. If false positives have to be entirely avoided, we recommend at least 0.7 or higher. On our dataset, results at $c \geq 0.8$ can be considered correct with near certainty. Cases which are not classifiable at the desired confidence threshold must be left undecided as "unknown authorship."

Hyperplane threshold c	**Classified cases [%]**	**Effectiveness**			
		Precision	Recall	$F_{0.5u}$	c@1
0.8	13.8	1.00	0.24	0.42	0.26
0.7	15.0	1.00	0.24	0.42	0.25
0.6	23.3	1.00	0.27	0.47	0.41
0.5	38.8	1.00	0.32	0.55	0.54
0.4	50.0	1.00	0.39	0.64	0.66
0.1	91.2	0.82	0.54	0.74	0.76
0.0	100.0	0.83	0.59	0.76	0.73

Table 1: Unmasking performance on our test data at various confidence thresholds. Recall was calculated after assigning all non-decisions the negative class. $F_{0.5u}$ and c@1 diverge significantly at high thresholds with an increasing class balance skew.

We compare the performance of our generalized unmasking approach to the two state-of-the-art short-text verification approaches by Bagnall (2015) and Halvani et al. (2017). Since no ready-to-use implementation of the latter approach is available, we reimplemented it using the PPMd compression algorithm and trained a random forest with 100 trees on the two suggested similarity measures Compression-based Cosine (CBC) and the Chen-Li metric (CLM). For a fairer comparison, we use a confidence threshold $c = 0.1$ for unmasking as this results in about the same amount of c@1 “optimization” as employed by Bagnall, although our $F_{0.5u}$ is higher at $c = 0$. The results of our performance comparison are shown in Table 2. The precision of our new generalized unmasking is the highest with slightly worse c@1 and $F_{0.5u}$ compared to Bagnall and CLM due to a lower recall. CBC appears to score the worst, although Halvani et al. found it to be best-performing. Overall, however, all approaches perform equally well and (treating non-answers as incorrect) no actual difference between their (binary) decision quality can be inferred from a McNemar test between unmasking and Bagnall ($\chi^2 = 0.35$, $p = 0.56$) or unmasking and Halvani et al. with CLM ($\chi^2 = 0.83$, $p = 0.36$).

In terms of runtime, compression performs best with under a minute on the whole corpus due to the optimized C implementation of the compressor, followed by our new generalized unmasking implementation with about 2–3 minutes. Bagnall runs the longest with 4:45 hours on all 80 cases in the corpus. It is worth noting that, unlike unmasking and compression models, Bagnall’s approach is not intrinsic and needs other texts from the input corpus to arrive at a decision for the single pair in question, which is a valid approach, but can effectively exploit potential biases in the corpus. We further noticed that compression appears to balance precision and recall more than the other approaches, whereas generalized unmasking heavily prioritizes precision, which is generally more preferable in a real-world scenario: here, false positives can have dire consequences, such as a wrong conviction. Another advantage of generalized unmasking is that it allows for easy optimization of all hyperparameters compared to the other approaches which are more of a blackbox. By building ensembles with other machine learning models, different features, chunk sizes, etc., we expect further improvements, but leave this for future work.

Approach	**Precision**	**Recall**	$\mathbf{F_{0.5u}}$	**c@1**
Generalized Unmasking	0.82	0.54	0.74	0.76
Bagnall	0.81	0.71	0.77	0.79
Halvani et al. (CLM)	0.78	0.78	0.78	0.78
Halvani et al. (CBC)	0.71	0.71	0.71	0.70

Table 2: Performance comparison with the state of the art in short-text authorship verification ($c = 0.1$ in generalized unmasking). Differences between the first three are non-significant.

4.3 Reproducibility

To enable our research on short-text unmasking, we developed an extensive general-purpose unmasking framework for running any kind of unmasking experiment with a plethora of different features, parameters, and aggregations for final or partial results from different runs. Each unmasking run comes with a detailed job configuration allowing for easy reproduction of previous experiments. We published the source code of our framework and all data used in our research under an open-source license alongside this paper.

5 Conclusion

We have successfully generalized and applied unmasking to short texts establishing another state-of-the-art verification approach for scenarios in which only little source material is available. This new unmasking approach prioritizes precision to deliver highly-reliable decisions even though short texts naturally show a high amount of uncertainty as a result of their low stylistic information content. We compared our approach to other state-of-the-art authorship verification systems specialized for short texts and can produce competitive results with low computational effort and high prospect of further optimization in the future.

References

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks—Notebook for PAN at CLEF 2015. In *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers*.

Edward Gaylord Bourne. 1897. The authorship of the federalist. *The American Historical Review*, 2(3):443–460.

Oren Halvani, Christian Winter, and Lukas Graner. 2017. Authorship verification based on compression-models. ArXiv 1706.00516.

Patrick Juola and Efstathios Stamatatos. 2013. Overview of the author identification task at PAN 2013. In *CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers*.

Dmitry V. Khmelev and William John Teahan. 2003. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of SIGIR 2003*, pages 104–110.

Moshe Koppel and Jonathan Schler. 2004. Authorship verification as a one-class classification problem. In *Proceedings of ICML 2004*, pages 1–7.

Anselmo Peñas and Álvaro Rodrigo. 2011. A simple measure to assess non-response. In *Proceedings of ACL 2011*, pages 1415–1424.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, Markov chains and author unmasking: An investigation. In *Proceedings of EMNLP 2006*, pages 482–491.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. 2015. Overview of the author identification task at PAN 2015. In *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers*.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Martin Potthast, Benno Stein, Patrick Juola, Miguel A. Sanchez-Perez, and Alberto Barrón-Cedeño. 2014. Overview of the author identification task at PAN 2014. In *CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers*.

William J. Teahan and David J. Harper. 2003. Using compression-based language models for text categorization. In *Language Modeling for Information Retrieval*, pages 141–165.

Adversarial Training for Satire Detection: Controlling for Confounding Variables

Robert McHardy[1], **Heike Adel**[1,2*] and **Roman Klinger**[1]
[1] Institut für Maschinelle Sprachverarbeitung, University of Stuttgart, Germany
[2] Bosch Center for Artificial Intelligence, Renningen, Germany
`{firstname.lastname}@ims.uni-stuttgart.de`

Abstract

The automatic detection of satire vs. regular news is relevant for downstream applications (for instance, knowledge base population) and to improve the understanding of linguistic characteristics of satire. Recent approaches build upon corpora which have been labeled automatically based on article sources. We hypothesize that this encourages the models to learn characteristics for different publication sources (*e.g.*, "The Onion" vs. "The Guardian") rather than characteristics of satire, leading to poor generalization performance to unseen publication sources. We therefore propose a novel model for satire detection with an adversarial component to control for the confounding variable of publication source. On a large novel data set collected from German news (which we make available to the research community), we observe comparable satire classification performance and, as desired, a considerable drop in publication classification performance with adversarial training. Our analysis shows that the adversarial component is crucial for the model to learn to pay attention to linguistic properties of satire.

1 Introduction

Satire is a form of art used to criticize in an entertaining manner (*cf.* Sulzer, 1771, p. 995ff.). It makes use of different stylistic devices, *e.g.*, humor, irony, sarcasm, exaggerations, parody or caricature (Knoche, 1982; Colletta, 2009). The occurrence of harsh, offensive or banal and funny words is typical (Golbert, 1962; Brummack, 1971).

Satirical news are written with the aim of mimicking regular news in diction. In contrast to misinformation and disinformation (Thorne and Vlachos, 2018), it does not have the intention of fooling the readers into actually believing something wrong in order to manipulate their opinion.

The task of satire detection is to automatically distinguish satirical news from regular news. This is relevant, for instance, for downstream applications, such that satirical articles can be ignored in knowledge base population. Solving this problem computationally is challenging. Even human readers are sometimes not able to precisely recognize satire (Allcott and Gentzkow, 2017). Thus, an automatic system for satire detection is both relevant for downstream applications and could help humans to better understand the characteristics of satire.

Previous work mostly builds on top of corpora of news articles which have been labeled automatically based on the publication source (*e.g.*, "The New York Times" articles would be labeled as regular while "The Onion" articles as satire[1]). We hypothesize that such distant labeling approach leads to the model mostly representing characteristics of the publishers instead of actual satire. This has two main issues: First, interpretation of the model to obtain a better understanding of concepts of satire would be misleading, and second, generalization of the model to unseen publication sources would be harmed. We propose a new model with adversarial training to control for the confounding variable of publication sources, *i.e.*, we debias the model.

Our experiments and analysis show that (1) the satire detection performance stays comparable when the adversarial component is included, and (2) that adversarial training is crucial for the model to pay attention to satire instead of publication characteristics. (3), we publish a large German data set for satire detection which is a) the first data set in German, b) the first data set including publication sources, enabling the experiments at hand, and c) the largest resource for satire detection so far.[2]

---

[*] Work was done at University of Stuttgart.

[1] https://www.theonion.com/, https://www.nytimes.com/

[2] Data/code: www.ims.uni-stuttgart.de/data/germansatire.

Proceedings of NAACL-HLT 2019, pages 660–665
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Previous Work

Previous work tackled the task of automatic English satire detection with handcrafted features, for instance, the validity of the context of entity mentions (Burfoot and Baldwin, 2009), or the coherence of a story (Goldwasser and Zhang, 2016). Rubin et al. (2016) use distributions of parts-of-speech, sentiment, and exaggerations. In contrast to these approaches, our model uses only word embeddings as input representations. Our work is therefore similar to Yang et al. (2017) and De Sarkar et al. (2018) who also use artificial neural networks to predict if a given text is satirical or regular news. They develop a hierarchical model of convolutional and recurrent layers with attention over paragraphs or sentences. We follow this line of work but our model is not hierarchical and introduces less parameters. We apply attention to words instead of sentences or paragraphs, accounting for the fact that satire might be expressed on a sub-sentence level.

Adversarial training is popular to improve the robustness of models. Originally introduced by Goodfellow et al. (2014) as generative adversarial networks with a generative and a discriminative component, Ganin et al. (2016) show that a related concept can also be used for domain adaptation: A domain-adversarial neural network consists of a classifier for the actual class labels and a domain discriminator. The two components share the same feature extractor and are trained in a minimax optimization algorithm with gradient reversal: The sign of the gradient of the domain discriminator is flipped when backpropagating to the feature extractor. Building upon the idea of eliminating domain-specific input representations, Wadsworth et al. (2018) debias input representations for recidivism prediction, or income prediction (Edwards and Storkey, 2016; Beutel et al., 2017; Madras et al., 2018; Zhang et al., 2018).

Debiasing mainly focuses on word embeddings, *e.g.*, to remove gender bias from embeddings (Bolukbasi et al., 2016). Despite previous positive results with adversarial training, a recent study by Elazar and Goldberg (2018) calls for being cautious and not blindly trusting adversarial training for debiasing. We therefore analyze whether it is possible at all to use adversarial training in another setting, namely to control for the confounding variable of publication sources in satire detection (see Section 3.1).

3 Methods for Satire Classification

3.1 Limitations of Previous Methods

The data set used by Yang et al. (2017) and De Sarkar et al. (2018) consists of text from 14 satirical and 6 regular news websites. Although the satire sources in train, validation, and test sets did not overlap, the sources of regular news were not split up according to the different data sets (Yang et al., 2017). We hypothesize that this enables the classifier to learn which articles belong to which publication of regular news and classify everything else as satire, given that one of the most frequent words is the name of the website itself (see Section 4.1). Unfortunately, we cannot analyze this potential limitation since their data set does not contain any information on the publication source[3]. Therefore, we create a new corpus in German (see Section 4.1) including this information and investigate our hypothesis on it.

3.2 Model

Motivated by our hypothesis in Section 3.1, we propose to consider two different classification problems (satire detection and publication identification) with a shared feature extractor. Figure 1 provides an overview of our model. We propose to train the publication identifier as an adversary.

3.2.1 Feature Extractor

Following De Sarkar et al. (2018), we only use word embeddings and no further handcrafted features to represent the input. We pretrain word embeddings of 300 dimensions on the whole corpus using word2vec (Mikolov et al., 2013). The feature generator f takes the embeddings of the words of each article as input for a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), followed by a self-attention layer as proposed by Lin et al. (2017). We refer to the union of all the parameters of the feature extractor as θ_f in the following.

3.2.2 Satire Detector

The gray part of Figure 1 shows the model part for our main task – satire detection. The satire detector feeds the representation from the feature extractor into a softmax layer and performs a binary classification task (satire: yes or no). Note that, in contrast to De Sarkar et al. (2018), we classify satire solely

---

[3] https://data.mendeley.com/datasets/hx3rzw5dwt/draft?a=377d5571-af17-4e61-bf77-1b77b88316de, v.1, 2017, accessed on 2018-11-23

on the document level, as this is sufficient to analyze the impact of the adversarial component and the influence of the publication source.

3.2.3 Publication Identifier

The second classification branch of our model aims at identifying the publication source of the input. Similar to the satire detector, the publication identifier consists of a single softmax layer which gets the extracted features as an input. It then performs a multi-class classification task since our dataset consists of 15 publication sources (see Table 1).

3.2.4 Adversarial Training

Let θ_f be the parameters of the feature extractors and θ_s and θ_p be the parameters of the satire detector and the publication identifier, respectively. The objective function for satire detection is

$$J_s = -\mathbb{E}_{(x,y_s)\sim p_{\text{data}}} \log P_{\theta_f \cup \theta_s}(y_s, x)\,, \quad (1)$$

while the objective for publication identification is

$$J_p = -\mathbb{E}_{(x,y_p)\sim p_{\text{data}}} \log P_{\theta_f \cup \theta_p}(y_p, x)\,. \quad (2)$$

Note that the parameters of the feature extractor θ_f are part of both model parts. Since our goal is to control for the confounding variable of publication sources, we train the publication identifier as an adversary: The parameters of the classification part θ_p are updated to optimize the publication identification while the parameters of the shared feature generator θ_f are updated to fool the publication identifier. This leads to the following update equations for the parameters

$$\theta_s := \theta_s - \eta \frac{\partial J_s}{\partial \theta_s} \quad (3)$$

$$\theta_p := \theta_p - \eta \frac{\partial J_p}{\partial \theta_p} \quad (4)$$

$$\theta_f := \theta_f - \eta \left(\frac{\partial J_s}{\partial \theta_f} - \lambda \frac{\partial J_p}{\partial \theta_f}\right) \quad (5)$$

with η being the learning rate and λ being a weight for the reversed gradient that is tuned on the development set. Figure 1 depicts the gradient flow.

4 Experiments

4.1 Experimental Setting

Dataset. We consider German regular news collected from 4 websites and German satirical news from 11 websites. Table 1 shows statistics and sources of the corpus, consisting of almost 330k articles. The corpus contains articles published between January 1st, 2000 and May 1st, 2018. Each publication has individual typical phrases and different most common words. Among the most common words is typically the name of each publication, *e.g.*, "Der Spiegel" has "SPIEGEL" as fifth and "Der Postillon" "Postillon" as third most common word. We did not delete those words to keep the dataset as realistic as possible. We randomly split the data set into training, development (dev) and test (80/10/10 %) with the same label distributions in all sets. Given the comparable large size of the corpus, we opt for using a well-defined test set for reproducability of our experiments in contrast to a crossvalidation setting.

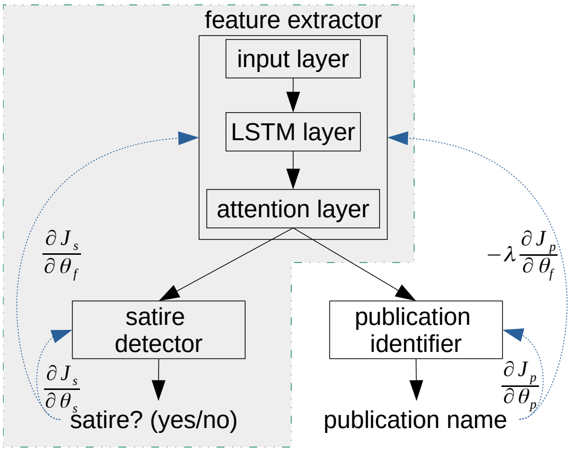


Figure 1: Architecture of the model. The gray area on the left shows the satire detector; the white area on the right is the adversary (publication identifier); the gradient flow with and without adversarial training is shown with blue arrows pointing upwards.

Research questions. We discuss two questions. RQ1: How does a decrease in publication classification performance through adversarial training affect the satire classification performance? RQ2: Is adversarial training effective for avoiding that the model pays most attention to the characteristics of publication source rather than actual satire?

Baseline. As a baseline model, we train the satire detector part (gray area in Figure 1) on the satire task. Then, we freeze the weights of the feature extractor and train the publication classifier on top of it. In addition, we use a majority baseline model which predicts the most common class.

Hyperparameters. We cut the input sentences to a maximum length of 500 words. This enables us to fully represent almost all satire articles and

	Publication	#Articles	Average Length		
			Article	Sent.	Title
Regular	Der Spiegel	31,180	556.74	19.04	7.47
	Der Standard	53,632	328.82	18.62	6.3
	Südd. Zeit.	177,605	635.77	17.58	7.74
	Die Zeit	57,802	1,116.53	17.0	5.2
Satire	Der Enthüller	324	404.3	13.87	9.67
	Eulenspiegel	192	1,072.17	17.45	4.38
	Nordd. Nach.	211	188.46	17.84	8.46
	Der Postillon	5,065	225.36	19.59	9.16
	Satirepatzer	193	262.99	12.26	7.53
	Die Tagespresse	1,271	301.28	16.39	10.83
	Titanic	149	292.88	16.04	7.79
	Welt (Satire)	1,249	291.45	21.76	9.02
	Der Zeitspiegel	171	315.76	18.69	9.71
	Eine Zeitung	416	265.16	16.04	13.35
	Zynismus24	402	181.59	17.67	11.96
	Regular	320,219	663.45	17.79	6.86
	Satire	9,643	269.28	18.73	9.52

Table 1: Corpus statistics (average length in words)

capture most of the content of the regular articles while keeping the training time low. As mentioned before, we represent the input words with 300 dimensional embeddings. The feature extractor consists of a biLSTM layer with 300 hidden units in each direction and a self-attention layer with an internal hidden representation of 600. For training, we use Adam (Kingma and Ba, 2014) with an initial learning rate of 0.0001 and a decay rate of 10^{-6}. We use mini-batch gradient descent training with a batch size of 32 and alternating batches of the two branches of our model. We avoid overfitting by early stopping based on the satire F1 score on the development set.

Evaluation. For evaluating satire detection, we use precision, recall and F1 score of the satire class. For publication identification, we calculate a weighted macro precision, recall and F1 score, *i.e.*, a weighted sum of class-specific scores with weights determined by the class distribution.

4.2 Selection of Hyperparameter λ

Table 2 (upper part) shows results for different values of λ, the hyperparameter of adversarial training, on dev. For $\lambda \in \{0.2, 0.3, 0.5\}$, the results are comparably, with $\lambda = 0.2$ performing best for satire detection. Setting $\lambda = 0.7$ leads to a performance drop for satire but also to $F_1 = 0$ for publication classification. Hence, we chose $\lambda = 0.2$ (the best performing model on satire classification) and $\lambda = 0.7$ (the worst performing model on publication identification) to investigate RQ1.

	Model	Satire			Publication		
		P	R	F1	P	R	F1
dev	majority class	0.0	0.0	0.0	29.5	54.3	38.3
	no adv	98.9	**52.6**	**68.7**	44.6	56.2	49.7
	adv, $\lambda = 0.2$	**99.3**	50.8	67.2	31.2	55.4	40.0
	adv, $\lambda = 0.3$	97.3	48.9	65.0	31.1	54.8	39.6
	adv, $\lambda = 0.5$	99.1	50.8	67.2	31.7	55.2	40.3
	adv, $\lambda = 0.7$	86.7	44.1	58.4	**26.9**	**0.0**	**0.0**
test	majority class	0.0	0.0	0.0	29.1	53.9	37.8
	no adv.	99.0	**50.1**	**66.5**	44.2	55.7	49.3
	adv, $\lambda = 0.2$	**99.4**	49.4	66.0	**30.8**	54.8	39.5
	adv, $\lambda = 0.7$	85.0	42.5	56.6	31.3	**0.0**	**0.0**

Table 2: Results on dev and independent test data.

5 Results (RQ1)

The bottom part of Table 2 shows the results on test data. The majority baseline fails since the corpus contains more regular than satirical news articles. In comparison to the baseline model without adversarial training (no adv), the model with $\lambda = 0.2$ achieves a comparable satire classification performance. As expected, the publication identification performance drops, especially the precision declines from 44.2 % to 30.8 %. Thus, a model which is punished for identifying publication sources can still learn to identify satire.

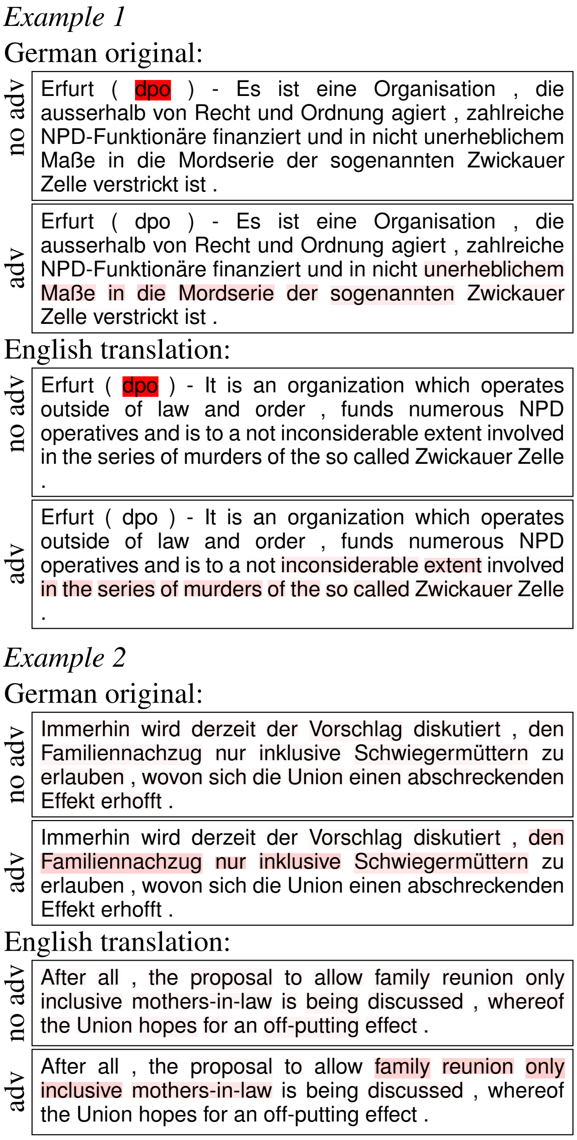
Similar to the results on dev, the recall of the model with $\lambda = 0.7$ drops to (nearly) 0 %. In this case, the satire classification performance also drops. This suggests that there are overlapping features (cues) for both satire and publication classification. This indicates that the two tasks cannot be entirely untangled.

6 Analysis (RQ2)

To address RQ2, we analyze the results and attention weights of the baseline model and our model with adversarial training.

6.1 Shift in Publication Identification

The baseline model (no adv) mostly predicts the correct publication for a given article (in 55.7 % of the cases). The model with $\lambda = 0.2$ mainly (in 98.2 % of the cases) predicts the most common publication in our corpus ("Süddeutsche Zeitung"). The model with $\lambda = 0.7$ shifts the majority of predictions (98.7 %) to a rare class (namely "Eine Zeitung"), leading to its bad performance.



Example 1

German original:

no adv: Erfurt (dpo) - Es ist eine Organisation , die ausserhalb von Recht und Ordnung agiert , zahlreiche NPD-Funktionäre finanziert und in nicht unerheblichem Maße in die Mordserie der sogenannten Zwickauer Zelle verstrickt ist .

adv: Erfurt (dpo) - Es ist eine Organisation , die ausserhalb von Recht und Ordnung agiert , zahlreiche NPD-Funktionäre finanziert und in nicht unerheblichem Maße in die Mordserie der sogenannten Zwickauer Zelle verstrickt ist .

English translation:

no adv: Erfurt (dpo) - It is an organization which operates outside of law and order , funds numerous NPD operatives and is to a not inconsiderable extent involved in the series of murders of the so called Zwickauer Zelle .

adv: Erfurt (dpo) - It is an organization which operates outside of law and order , funds numerous NPD operatives and is to a not inconsiderable extent involved in the series of murders of the so called Zwickauer Zelle .

Example 2

German original:

no adv: Immerhin wird derzeit der Vorschlag diskutiert , den Familiennachzug nur inklusive Schwiegermüttern zu erlauben , wovon sich die Union einen abschreckenden Effekt erhofft .

adv: Immerhin wird derzeit der Vorschlag diskutiert , den Familiennachzug nur inklusive Schwiegermüttern zu erlauben , wovon sich die Union einen abschreckenden Effekt erhofft .

English translation:

no adv: After all , the proposal to allow family reunion only inclusive mothers-in-law is being discussed , whereof the Union hopes for an off-putting effect .

adv: After all , the proposal to allow family reunion only inclusive mothers-in-law is being discussed , whereof the Union hopes for an off-putting effect .

Figure 2: Attention weight examples for satirical articles, with and without adversary.

6.2 Interpretation of Attention Weights

Figure 2 exemplifies the attention weights for a selection of satirical instances. In the first example the baseline model (no adv) focuses on a single word ("dpo" as a parody of the German newswire "dpa") which is unique to the publication the article was picked from ("Der Postillon"). In comparison the model using adversarial training ($\lambda = 0.2$) ignores this word completely and pays attention to "die Mordserie" ("series of murders") instead. In the second example, there are no words unique to a publication and the baseline spreads the attention evenly across all words. In contrast, the model with adversarial training is able to find cues for satire, being humor in this example ("family reunion [for refugees] is only allowed including mothers-in-law").

7 Conclusion and Future Work

We presented evidence that simple neural networks for satire detection learn to recognize characteristics of publication sources rather than satire and proposed a model that uses adversarial training to control for this effect. Our results show a considerable reduction of publication identification performance while the satire detection remains on comparable levels. The adversarial component enables the model to pay attention to linguistic characteristics of satire.

Future work could investigate the effect of other potential confounding variables in satire detection, such as the distribution of time and region of the articles. Further, we propose to perform more quantitative but also more qualitative analysis to better understand the behaviour of the two classifier configurations in comparison.

Acknowledgments

This work has been partially funded by the German Research Council (DFG), project KL 2869/1-1.

References

Hunt Allcott and Matthew Gentzkow. 2017. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*, 31(2):211–236.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H. Chi. 2017. Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. In *4th Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML) at 23rd SIGKDD conference on Knowledge Discovery and Data Mining (KDD 2017)*, Halifax, Canada.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Jürgen Brummack. 1971. Zu Begriff und Theorie der Satire. *Deutsche Vierteljahrsschrift für Literaturwissenschaft und Geistesgeschichte*, 45(1):275–377. In German.

Clint Burfoot and Timothy Baldwin. 2009. Automatic Satire Detection: Are You Having a Laugh? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 161–164. Association for Computational Linguistics.

Lisa Colletta. 2009. Political Satire and Postmodern Irony in the Age of Stephen Colbert and Jon Stewart. *The Journal of Popular Culture*, 42(5):856–874.

Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. 2018. Attending Sentences to detect Satirical Fake News. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3371–3380. Association for Computational Linguistics.

Harrison Edwards and Amos Storkey. 2016. Censoring Representations with an Adversary. In *International Conference on Learning Representations*.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 11–21, Brussels, Belgium. Association for Computational Linguistics.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(1):2030–2096.

Highet Golbert. 1962. *The Anatomy of Satire*. Princeton paperbacks. Princeton University Press.

Dan Goldwasser and Xiao Zhang. 2016. Understanding Satirical Articles Using Common-Sense. *Transactions of the Association for Computational Linguistics*, 4:537–549.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA. MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Ulrich Knoche. 1982. *Die römische Satire*. Orbis Biblicus Et Orientalis - Series Archaeologica. Vandenhoeck & Ruprecht. In German.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*.

David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. 2018. Learning adversarially fair and transferable representations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3384–3393, Stockholmsmässan, Stockholm Sweden. PMLR.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations*.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, San Diego, California. Association for Computational Linguistics.

Johann Georg Sulzer. 1771. *Allgemeine Theorie der Schönen Künste*, 1. edition. Weidmann; Reich, Leipzig. In German.

James Thorne and Andreas Vlachos. 2018. Automated Fact Checking: Task Formulations, Methods and Future Directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Christina Wadsworth, Francesca Vera, and Chris Piech. 2018. Achieving fairness through adversarial learning: an application to recidivism prediction. In *Proceedings of the Conference on Fairness, Accountability, and Transparency in Machine Learning (FATML)*, Stockholm, Sweden.

Fan Yang, Arjun Mukherjee, and Eduard Dragut. 2017. Satirical News Detection and Analysis using Attention Mechanism and Linguistic Features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1979–1989. Association for Computational Linguistics.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, pages 335–340, New York, NY, USA. ACM.

Keyphrase Generation: A Text Summarization Struggle

Erion Çano
Institute of Formal and Applied
Linguistics, Charles University,
Prague, Czech Republic
`cano@ufal.mff.cuni.cz`

Ondřej Bojar
Institute of Formal and Applied
Linguistics, Charles University,
Prague, Czech Republic
`bojar@ufal.mff.cuni.cz`

Abstract

Authors' keyphrases assigned to scientific articles are essential for recognizing content and topic aspects. Most of the proposed supervised and unsupervised methods for keyphrase generation are unable to produce terms that are valuable but do not appear in the text. In this paper, we explore the possibility of considering the keyphrase string as an abstractive summary of the title and the abstract. First, we collect, process and release a large dataset of scientific paper metadata that contains 2.2 million records. Then we experiment with popular text summarization neural architectures. Despite using advanced deep learning models, large quantities of data and many days of computation, our systematic evaluation on four test datasets reveals that the explored text summarization methods could not produce better keyphrases than the simpler unsupervised methods, or the existing supervised ones.

1 Introduction

A valuable concept for searching and categorizing scientific papers in digital libraries is the *keyphrase* (we use *keyphrase* and *keyword* interchangeably), a short set of one or few words that represent concepts. Scientific articles are commonly annotated with keyphrases based on taxonomies of concepts and the authors' judgment. Finding keyphrases that best describe the contents of a document is thus essential and rewarding.

Most of the proposed keyphrase extraction solutions tend to be unsupervised (Florescu and Caragea, 2017; Nguyen and Luong, 2010; Rose et al., 2010; Bougouin et al., 2013; Campos et al., 2018) and generate terms by selecting the most appropriate candidates, ranking the candidates based on several features and finally returning the top N. Another way is to utilize datasets of texts and keywords for training supervised models with linguistic or other features to predict if candidates are keywords or not (Witten et al., 1999; Turney, 2000; Medelyan, 2009; Hulth, 2003).

All above methods propose N keyphrases for each article which are joined together with "," (or other separator like ";") to form the *keyphrase string* of that article. They suffer from various problems or discrepancies. First, they are unable to find an optimal value for N and require it as a preset parameter. Furthermore, semantic and syntactic properties of article phrases are analyzed separately. The meaning of paragraphs, sections or entire document is thus missed. Lastly, only phrases that do appear in the article are returned. Meng et al. (2017) recently proposed a deep supervised keyphrase generation solution trained on a big dataset. It successfully solves the last two problems above, but not the first one.

Motivated by recent advances in neural machine translation and abstractive text summarization (Vaswani et al., 2017; Foster et al., 2018; Rush et al., 2015; See et al., 2017), in this paper, we explore the possibility of considering keyphrase generation as an abstractive text summarization task. Instead of generating keywords one by one and linking them to form the keyphrase string, we consider the later as an abstractive summary of the concatenated paper title and abstract. Different recently-proposed text summarization architectures are tried on four test datasets of article keyphrases (Tanti et al., 2017; Rush et al., 2015; See et al., 2017). We trained them with a newly created dataset of 2.2 million article titles, abstracts and keyphrase strings that we processed and released.[1]

The selected text summarization models are compared with popular unsupervised and supervised methods using ROUGE (Lin, 2004) and full-match F_1 metrics. The results show that though

---

[1] `http://hdl.handle.net/11234/1-2943`

Proceedings of NAACL-HLT 2019, pages 666–672
Minneapolis, Minnesota, June 2 - June 7, 2019.

trained with large data quantities for many days, the tried text summarization methods could not produce better keywords than the existing supervised or deep supervised predictive models. In our opinion, a possible explanation for this is the fact that the title and the abstract may not carry sufficient topical information about the article, even when joined together. In contrast, when assigning keywords annotations of their paper, authors are highly influenced by the topic aspects of it.

This paper carries several contributions, despite the fact that no progressive result scores were reached. It is the first work that considers keyphrase generation as an abstractive text summarization task. We produced a large dataset of article titles, abstracts, and keywords that can be used for keyword generation, text summarization or similar purposes. Finally, we evaluated the performance of different neural network architectures on summarization of article keyword strings, comparing them with popular unsupervised methods.

2 Scientific Paper Datasets

Because of the open source and open data initiatives, many public datasets from various domains can be found online (Çano and Morisio, 2015). Among the several collections of scientific articles, some of them have gained considerable popularity in research literature. In Meng et al. (2017), we found a recent and big collection of 20K paper abstracts and keyphrases. These metadata belong to articles of computer science from ACM Digital Library, ScienceDirect, and Web of Science. In Hulth (2003), we found a collection of 2000 (1500 for train/val and 500 for testing) abstracts in English, together with titles and authors' keywords. The corresponding articles were published from 1998 to 2002 and belong to the discipline of *Information Technology*. Furthermore, Krapivin et al. (2010) released a dataset of 2000 (1600 for train/val and 400 for testing) full articles published by ACM from 2003 to 2005 in Computer Science domain. More information about similar keyphrase data collections or other available resources can be found in Hasan and Ng (2014) and in online repositories.[2]

Regarding text summarization, some of the most popular datasets are: DUC-2004 [3] mainly used for testing, English Gigaword (Napoles et al., 2012), CNN/Daily Mail described in Section 4.3 of (Nallapati et al., 2016) and Newsroom, a heterogeneous bundle of news articles described in Grusky et al. (2018). These datasets are frequently used for the task of predicting titles from abstracts or short stories. However, no keyphrases are provided; they do not serve to our purpose. ArnetMiner is a recent attempt to crawl scientific paper data from academic networks (Tang et al., 2008). The system extracts profiles of researchers from digital resources and integrates their data in a common network. A spin-off is the Open Academic Graph (OAG) data collection (Sinha et al., 2015).

Attribute	**Train**	**Val**	**Test**	**Fullset**
Records	2M	100K	100K	2.2M
Keyphrases	12M	575K	870K	13.4M
Title tokens	24M	1.3M	1.6M	27M
Abstract tokens	441M	21M	37M	499M
Av. Keyphrase	6	5.8	8.7	6.1
Av. Title	12.1	12.8	15.9	12.3
Av. Abstract	220	211	372	227

Table 1: Statistics of OAGK dataset

To produce a usable collection for our purpose, we started from OAG. We extracted *title*, *abstract* and *keywords*. The list of keywords was transformed into a comma-separated string and a language identifier was used to remove records that were not in English. Abstracts and titles were lowercased, and Stanford CoreNLP tokenizer was used for tokenizing. Short records of fewer than 20 tokens in the abstract, 2 tokens in the title and 2 tokens in the keywords were removed. For the test portion, we selected documents of at least 27, 3 and 2 tokens in each field. Data preprocessing stopped here for the release version (no symbol filtering), given that many researchers want to filter text in their own way. This new dataset named OAGK can be used for both text summarization (predicting title from abstract) and keyphrase extraction (unsupervised, supervised or deep supervised) tasks. Some rounded measures about each set of released data are presented in Table 1.

3 Keyphrase Extraction Strategies

3.1 Unsupervised and Supervised Methods

TOPICRANK is an extractive method that creates topic clusters using the graph of terms and phrases (Bougouin et al., 2013). Obtained topics are then ranked according to their importance in the document. Finally, keyphrases are extracted by pick-

---

[2] `https://github.com/LIAAD/KeywordExtractor-Datasets`

[3] `https://duc.nist.gov/duc2004/`

ing one candidate from each of the most important topics. A more recent, unsupervised and feature-based method for keyphrase extraction is YAKE! (Campos et al., 2018). It heuristically combines features like *casing*, *word position* or *word frequency* to generate an aggregate score for each phrase and uses it to select the best candidates.

One of the first supervised methods is KEA described by Witten et al. (1999). It extracts those candidate phrases from the document that have good chances to be keywords. Several features like *TF-IDF* are computed for each candidate phrase during training. In the end, Naïve Bayes algorithm is used to decide if a candidate is a keyword or not (binary classification). An improvement and generalization of KEA is MAUI (Medelyan, 2009). Additional features are computed, and bagged decision trees are used instead of Naïve Bayes. The author reports significant performance improvements in precision, recall and F_1 scores.

The above keyphrase extraction methods and others like Florescu and Caragea (2017) or Nguyen and Luong (2010) reveal various problems. First, they are not able to find an optimal value for N (number of keywords to generate for an article) based on article contents and require it as a preset parameter. Second, the semantic and syntactic properties of article phrases (considered as candidate keywords) are analyzed separately. The meaning of longer text units like paragraphs or entire abstract/paper is missed. Third, only phrases that do appear in the paper are returned. In practice, authors do often assign words that are not part of their article.

Meng et al. (2017) overcome the second and third problem using an encoder-decoder model (COPYRNN) with a bidirectional Gated Recurrent Unit (GRU) and a forward GRU with attention. They train it on a datasets of hundred thousands of samples, consisting of abstract-keyword (one keyword only) pairs. The model is entirely data-driven and can produce terms that may not appear in the document. It still produces one keyword at a time, requiring N (first problem) as parameter to create the full keyphrase string.

3.2 Text Summarization Methods

To overcome the three problems mentioned in Section 3.1, we explore abstractive text summarization models proposed in the literature, trained with article abstracts and titles as sources and keyword strings as targets. They are expected to learn and paraphrase over entire source text and produce a summary in the form of a keyphrase string with no need for extra parameters. They should also introduce new words that do not appear in the abstract. Two simple encoder-decoder variants based on LSTMs are described in Figure 3 of Tanti et al. (2017). MERGE (Figure 3.a) encodes input and the current summary independently and merges them in a joint representation which is later decoded to predict the next summary token. INJECT model (Figure 3.b) on the other hand injects the source document context representation to the encoding part of the current summary before the decoding operation is performed.

ABS is presented in Figure 3.a of Rush et al. (2015). The encoder (Figure 3.b) takes in the input text and a learned soft alignment between the input and the summary, producing the context vector. This soft alignment is the attention mechanism (Bahdanau et al., 2014). To generate the summary words, Rush et al. apply a beam-search decoder with a window of K candidate words in each position of the summary.

Pointer-Generator network (POINTCOV) depicted in Figure 3 of See et al. (2017) is similar to ABS. It is composed of an attention-based encoder that produces the context vector. The decoder is extended with a pointer-generator model that computes a probability p_{gen} from the context vector, the decoder states, and the decoder output. That probability is used as a switch to decide if the next word is to be generated or copied from the input. This model is thus a compromise between abstractive and extractive (copying words from input) models. Another extension is the coverage mechanism for avoiding word repetitions in the summary, a common problem of encoder-decoder summarizers (Tu et al., 2016).

4 Results

We performed experiments with the unsupervised and supervised methods of Section 3 on the first three datasets of Section 2 and on OAGK. All supervised methods were trained with the 2M records of OAGK train part. An exception was MAUI which could be trained on 25K records at most (memory limitation). In addition to the processing steps of Section 2, we further replaced digit symbols with # and limited source and tar-

Method	Hulth (500)		Krapivin (400)		Meng (20K)		OAGK (100K)	
	F_1@5	**F_1@7**	**F_1@5**	**F_1@7**	**F_1@5**	**F_1@7**	**F_1@5**	**F_1@7**
YAKE!	19.35	21.47	17.98	17.4	17.11	15.19	15.24	14.57
TOPICRANK	16.5	20.44	6.93	6.92	11.93	11.72	11.9	12.08
MAUI	20.11	20.56	23.17	23.04	22.3	19.63	19.58	18.42
COPYRNN	**29.2**	**33.6**	**30.2**	**25.2**	**32.8**	**25.5**	**33.06**	**31.92**
MERGE	6.85	6.86	4.92	4.93	8.75	8.76	11.12	13.39
INJECT	6.09	6.08	4.1	4.11	8.09	8.09	9.61	11.22
ABS	14.75	14.82	10.24	10.29	12.17	12.09	14.54	14.57
POINTCOV	22.19	21.55	19.87	20.03	20.45	20.89	22.72	21.49

Table 2: Full-match scores of predicted keyphrases by various methods

Method	Hulth (500)		Krapivin (400)		Meng (20K)		OAGK (100K)	
	R_1F_1	R_LF_1	R_1F_1	R_LF_1	R_1F_1	R_LF_1	R_1F_1	R_LF_1
YAKE!	37.48	24.83	26.19	18.57	26.47	17.36	20.38	14.54
TOPICRANK	32.0	20.36	14.08	11.47	21.68	15.94	17.46	13.28
MAUI	36.88	27.16	28.29	23.74	34.33	28.12	32.16	25.09
COPYRNN	**44.58**	**35.24**	**39.73**	**30.29**	**42.93**	34.62	**43.54**	**36.09**
MERGE	15.19	9.45	9.66	7.14	16.53	12.31	17.3	14.43
INJECT	14.15	8.81	9.58	6.79	15.6	11.21	14.3	11.08
ABS	27.54	19.48	25.59	18.2	28.31	22.16	29.05	25.77
POINTCOV	37.16	33.69	35.81	29.52	38.47	**35.06**	38.66	34.04

Table 3: Rouge scores of predicted keyphrases by various methods

get text lengths to 270 and 21 tokens, respectively. Vocabulary size was also limited to the 90K most frequent words.

The few parameters of the unsupervised methods (length and windows of candidate keyphrases for YAKE!, ranking strategy for TOPICRANK) were tuned using the validation part of each dataset. For the evaluation, we used F_1 score of full matches between predicted and authors' keywords. Given that the average number of keywords in the data is about 6, we computed F_1 scores on top 5 and top 7 returned keywords (**F_1@5, F_1@7**).

Before each comparison, both sets of terms were stemmed with Porter Stemmer and duplicates were removed. In the case of summarization models, keyphrases were extracted from their comma-separated summaries. We also computed ROUGE-1 and ROUGE-L F_1 scores (R_1F_1, R_LF_1) that are suitable for evaluating short summaries (Lin, 2004). The keywords obtained from the unsupervised methods were linked together to form the keyphrase string (assumed summary). This was later compared with the original keyphrase string of the authors.

Full-match results on each dataset are reported in Table 2. From the unsupervised models, we see that YAKE! is consistently better than TOPICRANK. The next two supervised models perform even better, with COPYRNN being discretely superior than MAUI.

Results of the four summarization models seem disappointing. MERGE and INJECT are the worst on every dataset, with highest score 13.39 %. Various predictions of these models are empty or very short, and some others contain long word repetitions which are discarded during evaluation. As a result, there are usually fewer than five predicted keyphrases. This explains why **F_1@5** and **F_1@7** scores are very close to each other.

ABS works slightly better reaching scores from 10.24 to 14.75 %. POINTCOV is the best of the text summarizers producing keyphrase predictions that are usually clean and concise with few repetitions. This is probably the merit of the coverage mechanism. There is still a considerable gap between POINTCOV and COPYRNN. Rouge-1 and Rouge-L F_1 scores are reported in Table 3. COPYRNN is still the best but POINTCOV is close. ABS scores are also comparable to those of MAUI and YAKE!. TOPICRANK, MERGE and INJECT are again the worst.

Regarding the test datasets, the highest result scores are achieved on Hulth and the lowest on Krapivin. We checked some samples of the later and observed that each of them contains separation tags (e.g., –T, –A, –B, Figure etc.) for indicating different parts of text in the original paper. A more intelligent text cleaning step may be required on those data.

5 Discussion

The results show that the tried text summarization models perform poorly on full-match keyword predictions. Their higher ROUGE scores further indicate that the problem is not entirely in the summarization process. Observing a few samples, we found differences between the two evaluation strategies. For example, suppose we have the predicted keyword *"intelligent system"* compared against authors' keyword *"system design"*. Full-match evaluation adds nothing to $\mathbf{F}_1\mathbf{@5}$ and $\mathbf{F}_1\mathbf{@7}$ scores. However, in the case of ROUGE evaluation, the prediction is partially right and a certain value is added to $\boldsymbol{R_1F_1}$ score. In follow up works, one solution to this discrepancy could be to try partial-match comparison scores like overlap coefficients.

Another detail that has some negative effect in full-match scores is keyword separation. The predicted string:

"health care,,,,immune system; human -;
metabolism, immunity,,,,"

produces ["health care", "immune system", "human", "metabolism", "immunity"] as the list of keywords after removing the extra separators. Instead, we expected ["health care", "immune system", "human metabolism", "immunity"]. This again penalizes full-match scores but not $\boldsymbol{R_1F_1}$ score. A more intelligent keyword separation mechanism could thus help for higher full-match result scores.

A third reason could be the fact that we used the title and abstract of papers only. This is actually what most researchers do, as it is hard to find high quantities of article full texts for free. Article body is usually restricted. Abstractive summarization methods could still benefit from longer source texts. Using default hyperparameters for the models may have also influenced the results. Some parameter tuning could be beneficial, though.

The main reason could be even more fundamental. We trained abstractive summarization models on abstracts and titles with authors' keyphrases considered as golden ones. There might be two issues here. First, when setting their keywords, authors mostly consider the topical aspects of their work rather than paraphrasing over the contents. Abstracts and titles we used may not carry enough topical information about the article, even when joined together. Second, considering authors' keywords as golden ones may not be reasonable. One solution is to employ human experts and ask them to annotate each article based on what they read. This is however prohibitive when hundred thousands of samples are required. Extensive experiments on this issue may provide different facts and change the picture. For the moment, a safe way to go seems developing deep supervised generative models like the one of Meng et al. (2017) that predict one keyphrase at each step independently.

6 Conclusions

In this paper, we experimented with various unsupervised, supervised, deep supervised and abstractive text summarization models for predicting keyphrases of scientific articles. To the best of our knowledge, this is the first attempt that explores the possibility of conceiving article string of keywords as an abstractive summary of title and abstract. We collected and produced a large dataset of 2.2 million abstracts, titles and keyphrase strings from scientific papers available online. It can be used for future text summarization and keyphrase generation experiments. Systematic evaluation on four test datasets shows that the used summarization models could not produce better keywords than the supervised predictive models. Extensive experiments with more advanced summarizaiton methods and better parameter optimization may still reveal a different view of the situation.

Acknowledgments

This research work was supported by the project No. CZ.02.2.69/0.0/0.0/16_027/0008495 (International Mobility of Researchers at Charles University) of the Operational Programme Research, Development and Education, grant 19-26934X (NEUREM3) of the Czech Science Foundation and H2020-ICT-2018-2-825460 (ELITR) of the EU.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language*

Processing, pages 543–551. Asian Federation of Natural Language Processing.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *Advances in Information Retrieval*, pages 806–810. Springer International Publishing.

Erion Çano and Maurizio Morisio. 2015. Characterization of public datasets for recommender systems. In *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 249–257.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1115. Association for Computational Linguistics.

George Foster, Ashish Vaswani, Jakob Uszkoreit, Wolfgang Macherey, Lukasz Kaiser, Orhan Firat, Llion Jones, Noam Shazeer, Yonghui Wu, Ankur Bapna, Melvin Johnson, Mike Schuster, Zhifeng Chen, Macduff Hughes, Niki Parmar, and Mia Xu Chen. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL (1)*, pages 76–86. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autayeu, Maurizio Marchese, Enrico Blanzieri, and Nicola Segata. 2010. Keyphrases extraction from scientific documents. In *The Role of Digital Libraries in a Time of Global Change*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Olena Medelyan. 2009. *Human-competitive automatic topic indexing*. The University of Waikato, Phd Thesis.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 582–592. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thuy Dung Nguyen and Minh-Thang Luong. 2010. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 166–169, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083. Association for Computational Linguistics.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 243–246, New York, NY, USA. ACM.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA. ACM.

Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. 2017. What is the role of recurrent neural networks (rnns) in an image caption generator? *CoRR*, abs/1708.02043.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85. Association for Computational Linguistics.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, NY, USA. ACM.

SEQ^3: Differentiable Sequence-to-Sequence-to-Sequence Autoencoder for Unsupervised Abstractive Sentence Compression

Christos Baziotis[1,2], **Ion Androutsopoulos**[2], **Ioannis Konstas**[3], **Alexandros Potamianos**[1]
[1] School of ECE, National Technical University of Athens, Athens, Greece
[2] Department of Informatics, Athens University of Economics and Business, Athens, Greece
[3] Interaction Lab, School of Math. and Comp. Sciences, Heriot-Watt University, Edinburgh, UK
cbaziotis@mail.ntua.gr, ion@aueb.gr
i.konstas@hw.ac.uk, potam@central.ntua.gr

Abstract

Neural sequence-to-sequence models are currently the dominant approach in several natural language processing tasks, but require large parallel corpora. We present a sequence-to-sequence-to-sequence autoencoder (SEQ^3), consisting of two chained encoder-decoder pairs, with words used as a sequence of discrete latent variables. We apply the proposed model to unsupervised abstractive sentence compression, where the first and last sequences are the input and reconstructed sentences, respectively, while the middle sequence is the compressed sentence. Constraining the length of the latent word sequences forces the model to distill important information from the input. A pretrained language model, acting as a prior over the latent sequences, encourages the compressed sentences to be human-readable. Continuous relaxations enable us to sample from categorical distributions, allowing gradient-based optimization, unlike alternatives that rely on reinforcement learning. The proposed model does not require parallel text-summary pairs, achieving promising results in unsupervised sentence compression on benchmark datasets.

1 Introduction

Neural sequence-to-sequence models (SEQ2SEQ) perform impressively well in several natural language processing tasks, such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) or syntactic constituency parsing (Vinyals et al., 2015). However, they require massive parallel training datasets (Koehn and Knowles, 2017). Consequently there has been extensive work on utilizing non-parallel corpora to boost the performance of SEQ2SEQ models (Sennrich et al., 2016; Gülçehre et al., 2015), mostly in neural machine translation where models that require absolutely no parallel corpora have also been proposed (Artetxe et al., 2018; Lample et al., 2018b).

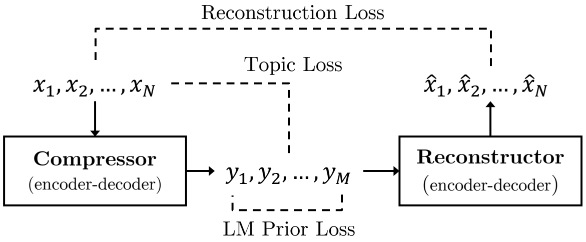


Figure 1: Overview of the proposed SEQ^3 autoencoder.

Unsupervised (or semi-supervised) SEQ2SEQ models have also been proposed for summarization tasks with no (or small) parallel text-summary sets, including unsupervised sentence compression. Current models, however, barely reach lead-N baselines (Fevry and Phang, 2018; Wang and Lee, 2018), and/or are non-differentiable (Wang and Lee, 2018; Miao and Blunsom, 2016), thus relying on reinforcement learning, which is unstable and inefficient. By contrast, we propose a sequence-to-sequence-to-sequence autoencoder, dubbed SEQ^3, that can be trained end-to-end via gradient-based optimization. SEQ^3 employs differentiable approximations for sampling from categorical distributions (Maddison et al., 2017; Jang et al., 2017), which have been shown to outperform reinforcement learning (Havrylov and Titov, 2017). Therefore it is a generic framework which can be easily extended to other tasks, e.g., machine translation and semantic parsing via task-specific losses. In this work, as a first step, we apply SEQ^3 to unsupervised abstractive sentence compression.

SEQ^3 (§2) comprises two attentional encoder-decoder (Bahdanau et al., 2015) pairs (Fig. 1): a compressor C and a reconstructor R. C (§2.1) receives an input text $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ of N words, and generates a summary $\mathbf{y} = \langle y_1, \ldots, y_M \rangle$ of M words ($M<N$), $\mathbf{y}$ being a latent variable. R and C communicate only through the discrete words of the summary $\mathbf{y}$ (§2.2). R (§2.3) produces a sequence $\hat{\mathbf{x}} = \langle \hat{x}_1, \ldots, \hat{x}_N \rangle$ of N words from $\mathbf{y}$, try-

Proceedings of NAACL-HLT 2019, pages 673–681
Minneapolis, Minnesota, June 2 - June 7, 2019.

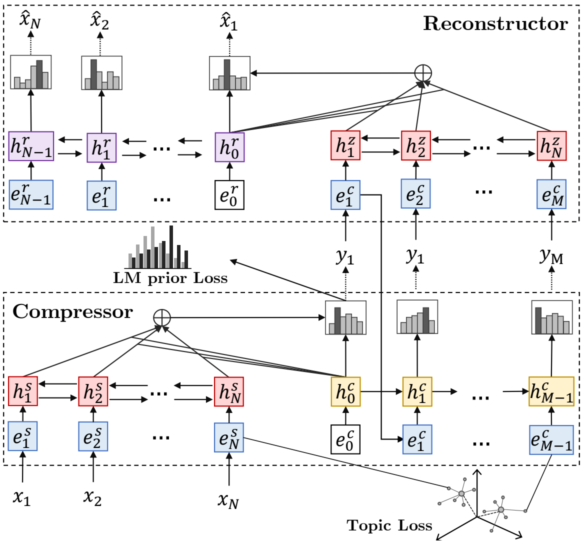


Figure 2: More detailed illustration of SEQ^3. The compressor (C) produces a summary from the input text, and the reconstructor (R) tries to reproduce the input from the summary. R and C comprise an attentional encoder-decoder each, and communicate only through the (discrete) words of the summary. The LM prior incentivizes C to produce human-readable summaries, while topic loss rewards summaries with similar topic-indicating words as the input text.

ing to minimize a reconstruction loss $L_R = (\mathbf{x}, \hat{\mathbf{x}})$ (§2.5). A pretrained language model acts as a prior on $\mathbf{y}$, introducing an additional loss $L_P(\mathbf{x}, \mathbf{y})$ that encourages SEQ^3 to produce human-readable summaries. A third loss $L_T(\mathbf{x}, \mathbf{y})$ rewards summaries $\mathbf{y}$ with similar topic-indicating words as $\mathbf{x}$. Experiments (§3) on the Gigaword sentence compression dataset (Rush et al., 2015) and the DUC-2003 and DUC-2004 shared tasks (Over et al., 2007) produce promising results.

Our contributions are: (1) a fully differentiable sequence-to-sequence-to-sequence (SEQ^3) autoencoder that can be trained without parallel data via gradient optimization; (2) an application of SEQ^3 to unsupervised abstractive sentence compression, with additional task-specific loss functions; (3) state of the art performance in unsupervised abstractive sentence compression. This work is a step towards exploring the potential of SEQ^3 in other tasks, such as machine translation.

2 Proposed Model

2.1 Compressor

The bottom left part of Fig. 2 illustrates the internals of the compressor C. An embedding layer projects the source sequence $\mathbf{x}$ to the word embeddings $\mathbf{e}^s = \langle e^s_1, \ldots, e^s_N \rangle$, which are then encoded by a bidirectional RNN, producing $\mathbf{h}^s = \langle h^s_1, \ldots, h^s_N \rangle$. Each h^s_t is the concatenation of the corresponding left-to-right and right-to-left states (outputs in LSTMs) of the bi-RNN.

$$h^s_t = [\overrightarrow{\text{RNN}}_s(e^s_t, \overrightarrow{h}^s_{t-1}); \overleftarrow{\text{RNN}}_s(e^s_t, \overleftarrow{h}^s_{t+1})]$$

To generate the summary $\mathbf{y}$, we employ the attentional RNN decoder of Luong et al. (2015), with their global attention and input feeding. Concretely, at each timestep ($t \in \{1, \ldots, M\}$) we compute a probability distribution a_i over all the states $h^s_1, \ldots, h^s_N$ of the source encoder conditioned on the current state h^c_t of the compressor's decoder to produce a context vector c_t.

$$a_i = \text{softmax}(h^{s\intercal}_i \, W_a \, h^c_t), \quad c_t = \sum_{i=1}^{N} a_i \, h^s_i$$

The matrix W_a is learned. We obtain a probability distribution for y_t over the vocabulary V by combining c_t and the current state h^c_t of the decoder.

$$o^c_t = \tanh(W_o \, [c_t; h^c_t] + b_o) \qquad (1)$$

$$u^c_t = W_v \, o^c_t + b_v \qquad (2)$$

$$p(y_t | y_{<t}, \mathbf{x}) = \text{softmax}(u^c_t) \qquad (3)$$

W_o, b_o, W_v, b_v are learned. c_t is also used when updating the state h^c_t of the decoder, along with the embedding e^c_t of y_t and a countdown argument $M - t$ (scaled by a learnable w_d) indicating the number of the remaining words of the summary (Fevry and Phang, 2018; Kikuchi et al., 2016).

$$h^c_{t+1} = \overrightarrow{\text{RNN}}_c(h^c_t, e^c_t, c_t, w_d \, (M - t)) \qquad (4)$$

For each input $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, we obtain a target length M for the summary $\mathbf{y} = \langle y_1, \ldots, y_M \rangle$ by sampling (and rounding) from a uniform distribution $U(\alpha N, \beta N)$; α, β are hyper-parameters ($\alpha < \beta < 1$); we set $M = 5$, if the sampled M is smaller. Sampling M, instead of using a static compression ratio, allows us to train a model capable of producing summaries with varying (e.g., user-specified) compression ratios. Controlling the output length in encoder-decoder architectures has been explored in machine translation (Kikuchi et al., 2016) and summarization (Fan et al., 2018).

2.2 Differentiable Word Sampling

To generate the summary, we need to sample its words y_t from the categorical distributions $p(y_t | y_{<t}, \mathbf{x})$, which is a non-differentiable process.

Soft-Argmax Instead of sampling y_t, a simple workaround during training is to pass as input to the next timestep of C's decoder and to the corresponding timestep of R's encoder a weighted sum of all the vocabulary's (V) word embeddings, using a peaked softmax function (Goyal et al., 2017):

$$\bar{e}_t^c = \sum_i^{|V|} e(w_i)\,\text{softmax}(u_t^c/\tau) \tag{5}$$

where u_t^c is the unnormalized score in Eq. 2 (i.e., the logit) of each word w_i and $\tau \in (0, \infty)$ is the temperature. As $\tau \to 0$ most of the probability mass in Eq. 5 goes to the most probable word, hence the operation approaches the $\arg\max$.

Gumbel-Softmax We still want to be able to perform sampling, though, as it has the benefit of adding stochasticity and facilitating exploration of the parameter space. Hence, we use the Gumbel-Softmax (GS) reparametrization trick (Maddison et al., 2017; Jang et al., 2017) as a low variance approximation of sampling from categorical distributions. Sampling a specific word y_t from the softmax (Eq. 3) is equivalent to adding (element-wise) to the logits an independent noise sample ξ from the Gumbel distribution[1] and taking the $\arg\max$:

$$y_t \sim \text{softmax}(u_t^c) \leftrightarrow y_t = \arg\max(u_t^c + \xi) \tag{6}$$

Therefore, using the GS trick, Eq. 5 becomes:

$$\tilde{e}_t^c = \sum_i^{|V|} e(w_i)\,\text{softmax}((u_t^c + \xi)/\tau) \tag{7}$$

Straight-Through Both relaxations lead to mixtures of embeddings, which do not correspond to actual words. Even though this enables the compressor to communicate with the reconstructor using continuous values, thus fully utilizing the available embedding space, ultimately our aim is to constrain them to communicate using *only* natural language. In addition, an unwanted discrepancy is created between training (continuous embeddings) and test time (discrete embeddings). We alleviate these problems with the Straight-Through estimator (ST) (Bengio et al., 2013). Specifically, in the forward pass of training we discretize $\tilde{e}_t^c$ by using the $\arg\max$ (Eq. 6), whereas in the backward pass we compute the gradients using the GS (Eq. 7). This is a biased estimator due to the mismatch between the forward and backward passes, but works well in practice. ST GS reportedly outperforms scheduled sampling (Goyal et al., 2017) and converges faster than reinforcement learning (Havrylov and Titov, 2017).

2.3 Reconstructor

The reconstructor (upper right of Fig. 2) works like the compressor, but its encoder operates on the embeddings $e_1^c, \ldots, e_M^c$ of the words $y_1, \ldots, y_M$ of the summary (exact embeddings of the sampled words y_t in the forward pass, approximate differentiable embeddings in the backward pass).

2.4 Decoder Initialization

We initialize the hidden state of each decoder using a transformation of the concatenation $[\overrightarrow{h_N^s}; \overleftarrow{h_1^s}]$ of the last hidden states (from the two directions) of its bidirectional encoder and a length vector, following Mallinson et al. (2018). The length vector for the decoder of the compressor C consists of the target summary length M, scaled by a learnable parameter w_v, and the compression ratio $\frac{M}{N}$.

$$h_0^c = \tanh(W_c\,[\overrightarrow{h_N^s}; \overleftarrow{h_1^s}; w_v M; \frac{M}{N}])$$

W_c is a trainable hidden layer. The decoder of the reconstructor R is initialized similarly.

2.5 Loss Functions

Reconstruction Loss $L_R(\mathbf{x}, \hat{\mathbf{x}})$ is the (negative) log-likelihood assigned by the (decoder of) R to the input (correctly reconstructed) words $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, where p_R is the distribution of R.

$$L_R(\mathbf{x}, \hat{\mathbf{x}}) = -\sum_{i=1}^{N} \log p_R(\hat{x}_i = x_i)$$

We do not expect $L_R(\mathbf{x}, \hat{\mathbf{x}})$ to decrease to zero, as there is information loss through the compression. However, we expect it to drive the compressor to produce such sentences that will increase the likelihood of the target words in the reconstruction.

LM Prior Loss To ensure that the summaries $\mathbf{y}$ are readable, we pretrain an RNN language model (see Appendix) on the *source* texts of the full training set. We compute the Kullback-Leibler divergence D_{KL} between the probability distributions of the (decoder of) the compressor ($p(y_t|y_{<t}, \mathbf{x})$, Eq. 3) and the language model ($p_{\text{LM}}(y_t|y_{<t}, \mathbf{x})$). Similar priors have been used in sentence compression (Miao and Blunsom, 2016) and agent communication (Havrylov and Titov, 2017).

---

[1] $\xi_i = -\log(-\log(x_i)),\ \ x_i \sim U(0, 1)$

We also use the following task-specific losses.

Topic Loss Words with high TF-IDF scores are indicative of the topic of a text (Ramos et al., 2003; Erkan and Radev, 2004). To encourage the compressor to preserve in the summary $\mathbf{y}$ the topic-indicating words of the input $\mathbf{x}$, we compute the TF-IDF-weighted average v^x of the word embeddings of $\mathbf{x}$ and the average v^y of the word embeddings of $\mathbf{y}$ and use their cosine distance as an additional loss $L_T = 1 - \cos(v^x, v^y)$.

$$v^x = \sum_{i=1}^{N} \frac{\text{IDF}(x_i)\, e_i^s}{\sum_{t=1}^{N} \text{IDF}(x_t)} \qquad v^y = \frac{1}{M} \sum_{i=1}^{M} e_i^c$$

(Using TF-IDF in v^y did not help.) All IDF scores are computed on the training set.

Length Penalty A fourth loss L_L (not shown in Fig. 1) helps the (decoder of the) compressor to predict the end-of-sequence (EOS) token at the target summary length M. L_L is the cross-entropy between the distributions $p(y_t|y_{<t}, \mathbf{x})$ (Eq. 3) of the compressor at $t = M + 1$ and onward, with the one-hot distribution of the EOS token.

2.6 Modeling Details

Parameter Sharing We tie the weights of layers encoding similar information, to reduce the number of trainable parameters. First, we use a shared embedding layer for the encoders and decoders, initialized with 100-dimensional GloVe embeddings (Pennington et al., 2014). Additionally, we tie the shared embedding layer with the output layers of both decoders (Press and Wolf, 2017; Inan et al., 2017). Finally, we tie the encoders of the compressor and reconstructor (see Appendix).

OOVs Out-of-vocabulary words are handled as in Fevry and Phang (2018) (see Appendix).

3 Experiments

Datasets We train SEQ^3 on the *Gigaword* sentence compression dataset (Rush et al., 2015).[2] It consists of pairs, each containing the first sentence of a news article ($\mathbf{x}$) and the article's headline ($\mathbf{y}$), a total of 3.8M/189k/1951 train/dev/test pairs. We also test (without retraining) SEQ^3 on DUC-2003 and DUC-2004 shared tasks (Over et al., 2007), containing 624/500 news articles each, paired with 4 reference summaries capped at 75 bytes.

Methods compared We evaluated SEQ^3 and an ablated version of SEQ^3. We only used the article sentences (sources) of the training pairs from Gigaword to train SEQ^3; our model is *never* exposed to target headlines (summaries) during training or evaluation, i.e., it is completely unsupervised. Our code is publicly available.[3]

We compare SEQ^3 to other unsupervised sentence compression models. We note that the extractive model of Miao and Blunsom (2016) relies on a pre-trained attention model using at least 500K parallel sentences, which is crucial to mitigate the inefficiency of sampling-based variational inference and REINFORCE. Therefore it is not comparable, as it is semi-supervised. The results of the extractive model of Fevry and Phang (2018) are also not comparable, as they were obtained on a different, not publicly available test set. We note, however, that they report that their system performs worse than the LEAD-8 baseline in ROUGE-2 and ROUGE-L on Gigaword. The only directly comparable unsupervised model is the abstractive 'Pretrained Generator' of Wang and Lee (2018). The version of 'Adversarial REINFORCE' that Wang and Lee (2018) consider unsupervised is actually weakly supervised, since its discriminator was exposed to the summaries of the same sources the rest of the model was trained on.

As baselines, we use LEAD-8 for Gigaword, which simply selects the first 8 words of the source, and PREFIX for DUC, which includes the first 75 bytes of the source article. We also compare to supervised abstractive sentence compression methods (Tables 1-3). Following previous work, we report the average F1 of ROUGE-1, ROUGE-2, ROUGE-L (Lin, 2004). We implemented SEQ^3 with LSTMs (see Appendix) and during inference we perform greedy-sampling.

Results Table 1 reports the Gigaword results. SEQ^3 outperforms the unsupervised Pretrained Generator across all metrics by a large margin. It also surpasses LEAD-8. If we remove the LM prior, performance drops, esp. in ROUGE-2 and ROUGE-L. This makes sense, since the pretrained LM rewards correct word order. We also tried removing the topic loss, but the model failed to converge and results were extremely poor (Table 1). Topic loss acts as a bootstrap mechanism, biasing the compressor to generate words that maintain the topic of the input text. This greatly reduces variance due to sampling in early stages of training, alleviating the need to pretrain individual

---

[2] `github.com/harvardnlp/sent-summary`

[3] `https://github.com/cbaziotis/seq3`

Type	Supervision	Methods	R-1	R-2	R-L
Supervised	3.8M	ABS (Rush et al., 2015)	29.55	11.32	26.42
		SEASS (Zhou et al., 2017)	36.15	17.54	33.63
		words-lvt5k-1sent (Nallapati et al., 2016)	**36.4**	**17.7**	**33.71**
Weakly supervised	(3.8M)	Adversarial REINFORCE (Wang and Lee, 2018)	28.11	9.97	25.41
Unsupervised	0	LEAD-8 (Baseline)	21.86	7.66	20.45
		Pretrained Generator (Wang and Lee, 2018)	21.26	5.60	18.89
		SEQ^3 (Full)	**25.39**	**8.21**	**22.68**
		SEQ^3 w/o LM prior loss	24.48	6.68	21.79
		SEQ^3 w/o TOPIC loss	3.89	0.1	3.75

Table 1: Average results on the (English) Gigaword dataset for abstractive sentence compression methods.

Model	R-1	R-2	R-L
ABS (Rush et al., 2015)	28.48	8.91	23.97
PREFIX	**21.3**	**6.38**	**18.82**
SEQ^3 (Full)	20.90	6.08	18.55

Table 2: Averaged results on the DUC-2003 dataset; the top part reports results of supervised systems.

Model	R-1	R-2	R-L
TOPIARY (Zajic et al., 2007)	25.12	6.46	20.12
Woodsend et al. (2010)	22	6	17
ABS (Rush et al., 2015)	28.18	8.49	23.81
PREFIX	20.91	5.52	18.20
SEQ^3 (Full)	**22.13**	**6.18**	**19.3**

Table 3: Averaged results on the DUC-2004 dataset; the top part reports results of supervised systems.

input: the american sailors who **thwarted** somali pirates flew home to the u.s. on wednesday but without their captain , who was still aboard a navy destroyer after being rescued from the **hijackers** .
gold: us sailors who thwarted pirate hijackers fly home
SEQ^3: the american sailors who **foiled** somali pirates flew home after crew **hijacked** .

input: the central election commission -lrb- cec -rrb- on monday **decided that taiwan will hold another election** of national assembly members on may # .
gold: national <unk> election scheduled for may
SEQ^3: the central election commission -lrb- cec UNK **announced elections** .

input: dave bassett resigned as manager of struggling english premier league side nottingham forest on saturday after they were knocked out of the f.a. cup in the third round , according to local reports on saturday .
gold: forest manager bassett quits .
SEQ^3: dave bassett resigned as manager of struggling english premier league side UNK forest on knocked round press

Figure 3: Good/bad example summaries on Gigaword.

components, unlike works that rely on reinforcement learning (Miao and Blunsom, 2016; Wang and Lee, 2018). Overall, both losses work in synergy, with the topic loss driving *what* and the LM prior loss driving *how* words should be included in the summary. SEQ^3 behaves similarly on DUC-2003 and DUC-2004 (Tables 2-3), although it was trained on Gigaword. In DUC-2003, however, it does not surpass the PREFIX baseline.

Finally, Fig. 3 illustrates three randomly sampled outputs of SEQ^3 on Gigaword. In the first one, SEQ^3 copies several words esp. from the beginning of the input (hence the high ROUGE-L) exhibiting extractive capabilities, though still being adequately abstractive (bold words denote paraphrases). In the second one, SEQ^3 showcases its true abstractive power by paraphrasing and compressing multi-word expressions to single content words more heavily, still without losing the overall meaning. In the last example, SEQ^3 progressively becomes ungrammatical though interestingly retaining some content words from the input.

4 Limitations and Future Work

The model tends to copy the first words of the input sentence in the compressed text (Fig. 3). We hypothesize that since the reconstructor is autoregressive, i.e., each word is conditioned on the previous one, errors occurring early in the generated sequence have cascading effects. This inevitably encourages the compressor to select the first words of the input. A possible workaround might be to modify SEQ^3 so that the first encoder-decoder pair would turn the inputs to longer sequences, and the second encoder-decoder would compress them trying to reconstruct the original inputs. In future work, we plan to explore the potential of SEQ^3 in other tasks, such as unsupervised machine translation (Lample et al., 2018a; Artetxe et al., 2018) and caption generation (Xu et al., 2015).

Acknowledgments

We would like to thank Ryan McDonald for helpful discussions and feedback. This work has been partially supported by computational time granted from the Greek Research & Technology Network (GR-NET) in the National HPC facility - ARIS. We thank NVIDIA for donating a TitanX GPU.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the Annual Meeting of International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, California.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the Conference of the NAACL:HLT*, pages 93–98, San Diego, California.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising autoencoders. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the Annual Meeting of the ACL*, pages 366–371, Vancouver, Canada.

Jiatao Gu, Daniel Jiwoong Im, and Victor O. K. Li. 2018. Neural machine translation with gumbel-greedy decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5125–5132.

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proceedings of the Advances in Neural Information Processing Systems*, pages 2149–2159.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumble-softmax. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, Banff, AB, Canada.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, Hervé Jégou, et al. 2018a. Word translation without parallel data. In *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2018. Sentence compression for arbitrary languages via multilingual pivoting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2453–2464, Brussels, Belgium.

Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing and Management*, 43(6):1506–1520.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the 1st International Conference on Machine Learning*, volume 242, pages 133–142.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the ACL*, pages 1073–1083, Vancouver, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the ACL*, pages 86–96, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2773–2781.

Yaushian Wang and Hung-yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Cambridge, MA.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, volume 37, pages 2048–2057, Lille, France. PMLR.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing Management Special Issue on Summarization*, 43(6):1549–1570.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the ACL*, pages 1095–1104, Vancouver, Canada.

A Appendix

A.1 Temperature for Gumbel-Softmax

Even though the value of the temperature τ does not affect the forward pass, it greatly affects the gradient computation and therefore the learning process. Jang et al. (2017) propose to anneal τ during training towards zero. Gulcehre et al. (2017) propose to learn τ as a function of the compressor's decoder state h_t^c, in order to reduce hyper-parameter tuning:

$$\tau(h_t^c) = \frac{1}{\log(1 + \exp(w_\tau^\intercal h_t^c)) + 1} \quad (8)$$

where w_τ is a trainable parameter and $\tau(h_t^c) \in (0, 1)$. Havrylov and Titov (2017) add τ_0 as a hyper-parameter which controls the upper bound of the temperature.

$$\tau(h_t^c) = \frac{1}{\log(1 + \exp(w_\tau^\intercal h_t^c)) + \tau_0} \quad (9)$$

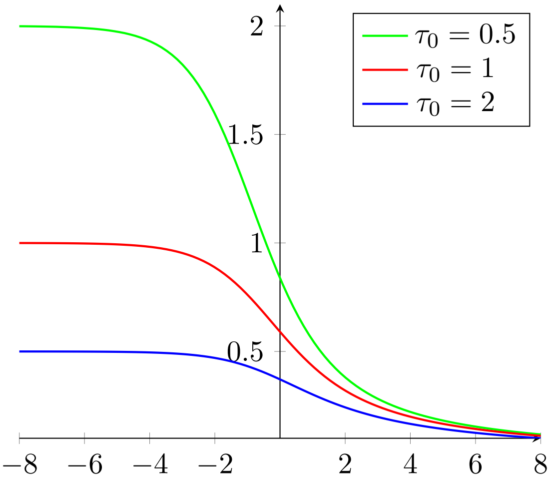


Figure 4: Plot of Eq. 9, with different values for the upper bound τ_0.

In our experiments, we had convergence problems with the learned temperature technique. We found that the compressor preferred values close to the upper bound, which led to unstable training, forcing us to set $\tau_0 > 1$ to stabilize the training process. Our findings align with the behavior reported by Gu et al. (2018). Consequently, we follow their choice and fix $\tau = 0.5$, which worked well in practice.

A.2 Out of Vocabulary (OOV) Words

The vocabulary of our experiments comprises the 15k most frequent words of Gigaword's training input texts (without looking at their summaries). To handle OOVs, we adopt the approach of Fevry and Phang (2018), which can be thought of as a simpler form of copying compared to pointer networks (See et al., 2017). We use a small set (10 in our experiments) of special OOV tokens $\text{OOV}_1, \text{OOV}_2, \ldots, \text{OOV}_{10}$, whose embeddings are updated during learning. Given an input text $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, we replace (before feeding $\mathbf{x}$ to SEQ^3) each unknown word x_i with the first unused (for the particular $\mathbf{x}$) OOV token, taking care to use the same OOV token for all the occurrences of the same unknown word in $\mathbf{x}$. For example, if 'John' and 'Rome' are not in the vocabulary, then "John arrived in Rome yesterday. While in Rome, John had fun." becomes "OOV_1 arrived in OOV_2 yesterday. While in OOV_2, OOV_1 had fun." If a new unknown word x_i is encountered in $\mathbf{x}$ and all the available OOV tokens have been used, x_i is replaced by 'UNK', whose embedding is also updated during learning. The OOV tokens (and 'UNK') are included in the vocabulary, and SEQ^3 learns to predict them as summary words, in effect copying the corresponding unknown words of $\mathbf{x}$. At test time, we replace the OOV tokens with the corresponding unknown words.

A.3 Reconstruction Word Drop

Our model is an instance of Variational Auto-Encoders (VAE) (Kingma and Welling, 2014). A common problem in VAEs is that the reconstructor tends to disregard the latent variable. We weaken the reconstructor R, in order to force it to fully utilize the latent sequence $\mathbf{y}$ to generate $\hat{\mathbf{x}}$. To this end, we employ word dropout as in Bowman et al. (2016) and randomly drop a percentage of the input words, thus forcing R to rely solely on $\mathbf{y}$ to make good reconstructions.

A.4 Implementation and Hyper-parameters

We implemented SEQ^3 in PyTorch (Paszke et al., 2017). All the RNNs are LSTMs (Hochreiter and Schmidhuber, 1997). We use a shared encoder for the compressor and the reconstructor, consisting of a two-layer bidirectional LSTM with size 300 per direction. We use separate decoders for the compressor and the reconstructor; each decoder is a two-layer unidirectional LSTM with size 300. The (shared) embedding layer of the compressor and the reconstructor is initialized with 100-dimensional GloVe embeddings (Pennington et al., 2014) and is tied with the output (projection) layers of the decoders and jointly finetuned during training. We apply layer normalization (Ba et al., 2016) to the context vectors (Eq. 1) of the compressor and the reconstructor. We apply word dropout (§A.3) to the reconstructor with $p = 0.5$.

During training, the summary length M is sampled from $U(0.4N, 0.6N)$; during testing, $M = 0.5N$. The four losses are summed, λs being scalar hyper-parameters.

$$L = \lambda_R L_R + \lambda_P L_P + \lambda_T L_T + \lambda_L L_L$$

We set $\lambda_R = \lambda_T = 1$, $\lambda_L = \lambda_P = 0.1$. We use the Adam (Kingma and Ba, 2015) optimizer, with batch size 128 and the default learning rate 0.001. The network is trained for 5 epochs.

LM Prior The pretrained language model is a two-layer LSTM of size 1024 per layer. It uses its own embedding layer of size 256, which is randomly initialized and updated when training the language model. We apply dropout with $p = 0.2$ to the embedding layer and dropout with $p = 0.5$ to the LSTM layers. We use Adam (Kingma and Ba, 2015) with batch size 128 and the network is trained for 30 epochs. The learning rate is set initially to 0.001 and is multiplied with $\gamma = 0.5$ every 10 epochs.

Evaluation Following Chopra et al. (2016), we filter out pairs with empty headlines from the test set. We employ the PYROUGE package with "*-m -n 2 -w 1.2*" to compute ROUGE scores. We use the provided tokenizations of the Gigaword and DUC-2003, DUC-2004 datasets. All hyper-parameters were tuned on the development set.

Crowdsourcing Lightweight Pyramids for Manual Summary Evaluation

Ori Shapira[1], **David Gabay**[1], **Yang Gao**[2], **Hadar Ronen**[1],
Ramakanth Pasunuru[3], **Mohit Bansal**[3], **Yael Amsterdamer**[1], **and Ido Dagan**[1]

[1]Bar-Ilan University [2]UKP Technische Universität Darmstadt [3]UNC Chapel Hill
`{obspp18, dawid.gabay}@gmail.com`
`gao@ukp.informatik.tu-darmstadt.de, hadarg@gmail.com`
`{ram, mbansal}@cs.unc.edu, {amstery, dagan}@cs.biu.ac.il`

Abstract

Conducting a manual evaluation is considered an essential part of summary evaluation methodology. Traditionally, the Pyramid protocol, which exhaustively compares system summaries to references, has been perceived as very reliable, providing objective scores. Yet, due to the high cost of the Pyramid method and the required expertise, researchers resorted to cheaper and less thorough manual evaluation methods, such as Responsiveness and pairwise comparison, attainable via crowdsourcing. We revisit the Pyramid approach, proposing a lightweight sampling-based version that is crowdsourcable. We analyze the performance of our method in comparison to original expert-based Pyramid evaluations, showing higher correlation relative to the common Responsiveness method. We release our crowdsourced Summary-Content-Units, along with all crowdsourcing scripts, for future evaluations.

1 Introduction

Evaluating *content* quality of summaries is an integral part of summarization research. Measuring the performance of a summarization system can be done through either automatic or manual evaluation. An automatic evaluation, in practice working at the lexical level, provides an inexpensive means of measuring the validity of a system, both for system comparisons and for quick development cycle testing. Due to the shallowness of the automatic approaches, their reliability is often perceived as insufficient (Owczarzak et al., 2012; Chaganty et al., 2018). This calls for the more expensive manual evaluation, which employs human-in-the-loop protocols for assessment.

The Pyramid method (Nenkova and Passonneau, 2004) is a prominent manual evaluation methodology that is considered highly reliable for comparing summarization systems. It relies on a small set of manually-crafted reference summaries, out of which all summary content units (SCUs) are manually extracted. System summaries are then manually checked for coverage of each individual SCU, from which an overall system score is derived. The Pyramid evaluation method's reliability comes at a cost. It requires laborious manual work performed by annotators who must browse through non-trivial guidelines (Passonneau, 2006). Due to these drawbacks, it was only used in a few DUC and TAC (NIST, 2014, 2018) benchmarks.

Instead, summarization work in recent years has mostly employed simpler manual evaluation approaches, such as Responsiveness and pairwise comparison, which do not rely on reference summaries and can be attained via crowdsourcing. Yet, these methods are quite subjective, since evaluators need to provide only a single global judgment for the quality of a summary (or a pair of summaries). Such judgments are far more subjective than the Pyramid score, which is derived from many, more objective, local decisions, each judging independently the presence of an individual SCU. Indeed, it was shown that the above subjective crowdsourcing-based evaluation methods are not reliable enough to produce consistent scores across experiments (Gillick and Liu, 2010).

We propose a simplified crowdsourcable and reproducible version of the Pyramid method, that suggests appealing advantages over prior crowdsourcable evaluation methods. Like the original Pyramid, our method leverages the strong signal of the reference summaries and similarly bases its score on less subjective SCU judgments. In contrast to the original Pyramid, we rely on statistical sampling rather than exhaustive SCU extraction and testing, lowering overall cost. Empirically, our method correlates with the original Pyra-

Proceedings of NAACL-HLT 2019, pages 682–687
Minneapolis, Minnesota, June 2 - June 7, 2019.

mid scores better than the common Responsiveness method, and shows better stability.

2 Background: Manual Summary Evaluation

The Pyramid method (Nenkova and Passonneau, 2004) consists of two manual phases. The first phase is *pyramid creation*, performed once when a dataset is constructed, per each input topic to be summarized (either a single document or a set of documents). In this phase, experts exhaustively extract all *SCU contributors* ("mentions"), each being a text span describing an individual fact. SCU contributors are extracted from *several* reference summaries of the source text. Coreferring SCU contributors across reference summaries are then merged into a single *SCU*, which is given a representative label. Each SCU is then assigned a weight, equal to the number of reference summaries in which it was found, indicating its salience.

The second phase is *system evaluation*, performed over the summaries produced by the evaluated system. Each Pyramid SCU for the source text is manually checked for its presence in the given system summary, whose Pyramid score is then computed as a normalized sum of the weights of the SCUs it contains. The overall system score is defined as the average Pyramid score over all its evaluated summaries. Although certain normalization variants attempt to weigh in SCU precision, the score is essentially an absolute "recall-style" interpretation reflecting the system's ability to cover the content units found in the reference summaries. Such a fairly robust score allows, in principle, system comparison across experiments (Nenkova and Passonneau, 2004).

We note that due to the Pyramid method's reliability, some research has been carried out on simulating the Pyramid method as a fully automatic one (Yang et al., 2016; Hirao et al., 2018). The hope of such a line of work is to find an automatic evaluation method that is more reliable than the commonly used ones, by taking the reference summary *semantic* content into account. Despite these efforts, automated Pyramid evaluations did not make their way yet to mainstream summary evaluation practices, where variants of the ROUGE metric (Lin, 2004) still prevail. In any case, as this paper focuses on manual evaluation, we compare our results to those of the manual Pyramid.

The *Responsiveness* method, introduced in DUC 2003 (NIST, 2003), does not require reference summaries. Instead, human evaluators typically read both the source text and the system summary. They then assign a single subjective score on a Likert scale for the summary quality, often with respect to a topic statement or guiding question. Finally, compared systems are ranked by the average score of their summaries. This method naturally developed into a crowdsourcing task, and is now used frequently in some variants (Grusky et al., 2018; Paulus et al., 2018).

Another common crowdsourcable evaluation method is pairwise comparison (Gao et al., 2018; Falke et al., 2017; Fan et al., 2018): an evaluator is asked to judge which of two competing summaries of the same text is superior, usually while observing the source text. This protocol allows comparing only two systems at a time, where the superior is determined by the total votes over all input texts. The obvious disadvantage of the approach is the difficulty of comparing many systems, in the absence of absolute scores. Also, this method may tend to suffer from transitivity inconsistencies when comparing multiple system pairs (Gillick and Liu, 2010).

The lightweight crowdsourcable Pyramid version we propose aims to preserve the interpretability and relative objectiveness of the Pyramid scores. This could provide absolute scores for comparing multiple systems, which the pairwise method does not, in a more reliable manner than Responsiveness evaluation.

3 Our Lightweight Pyramid Method

Our Lightweight Pyramid method mimics the two phases of the original Pyramid protocol in a crowdsourced setting, with some adjustments.

Pyramid creation. The input for this phase is several reference summaries of a topic. Each reference is presented to two crowd workers, asking to extract eight SCU-like statements, yielding 16 potential SCUs per reference summary. The instructions guide workers to copy-and-paste extractions from the text, possibly modifying them to stand-alone sentences, that should (a) be brief and focused on a single fact; (b) capture important information; (c) rely solely on the text rather than general knowledge of the worker. Further, the statements should appear in different places in the text.

The copy-and-paste approach allows us to easily detect and filter duplicate statements extracted from the *same* reference by both annotators, which we identify via bag-of-lemmas cosine similarity. Further, too long sentences are filtered. In our experiments (see Section 4), we were left with an average of about 13 SCUs per reference summary. Then, we take the union of SCUs from all reference summaries, which yielded in our experiments 51 SCUs on average per topic, coming from four reference summaries. These SCUs are used to create tasks for the system evaluation phase.

Recall that in the original Pyramid, SCUs are exhaustively collected; then, coreferring SCUs *between* reference summaries are merged and weighted by the number of reference summaries from which they originate. In contrast, our method enables using a *sample* of SCUs for evaluation, out of the SCUs collected in this phase (we have sampled, for uniformity, 32 SCUs per topic). Further, it avoids posing the task of merging coreferring SCUs *across* references, which is difficult and error-prone, particularly when expected from crowd workers. Instead, we rely on the higher likelihood of a repeated fact to be included in our sample, possibly more than once. This implicitly increases the expected impact of repeated facts on our evaluation.

System evaluation. In this phase, a crowd worker is presented with a system summary and a fixed-sized small set of SCUs (we used sets of 16 SCUs). The worker is asked whether each SCU can be inferred from the system summary text. The guidelines advise workers to refrain from using general knowledge and to ignore minor content differences between the SCU and the system summary. Each SCU should be assessed by a few crowd workers, to ensure the stability of the results (in our experiments, each SCU was assigned for evaluation to 5 workers).

Scoring. Following common practice in crowdsourcing, we use techniques of filtering out noisy workers who had high disagreement with others (pairwise worker agreement < 0.5). Then, using the remaining answers, we take the majority vote for each SCU to decide whether it appears in the system summary.[1] We resolve ties with a "not present" default, as the more likely answer. We then compute the system summary score as the percentage of SCUs it matched out of the set of judged SCUs. A system's final score is its average score over all topics.

4 Experiments

Experimental setup. We used the DUC 2005 and 2006 multi-document summarization datasets (NIST, 2014), which contain expert evaluations for both Pyramid and Responsiveness. Each of the two datasets includes 20 document clusters, each pertaining to a target topic, with four reference summaries and 25 (2005) or 22 (2006) system summaries per topic. All summaries are 250 words long. On average, 105 weighted SCUs were extracted, by experts, for each topic. In comparison, our setup gathers 32 sampled crowdsourced unweighted SCUs.

As suggested in Dang (2006) and Passonneau et al. (2006), the 2005 data tends to be easier to evaluate than the 2006 data, seemingly due to "less natural" document clusters with respect to practical summarization settings. Passonneau et al. (2006) show that the document sets in 2005 were overall more difficult for systems to summarize, as reflected by a lower average Pyramid score across all systems. The 2005 topics are more complex as they yield fewer general, context-independent SCUs. For example, as Dang (2006) indicates, there are more topics that had a relatively large number of specific named entities. Consequently, due to the topic hardness, Passonneau et al. (2006) indicate very few significant differences between overall system Pyramid scores, as evident by Tukey's HSD test. While 2006 systems can be divided into eight significantly different Pyramid score groups, in 2005 only two such groups emanate. Additionally, the guidelines and scoring method were slightly improved in 2006, relative to 2005. For these reasons, we focused on the 2006 dataset, fully annotating it, while utilizing half the topics, randomly chosen, from the 2005 data.

Using Amazon Mechanical Turk,[2] we qualified workers with over 5000 approved assignments and a 99% approval rate. We paid workers \$0.50 per reference summary annotation assignment (generating 8 SCUs), yielding a total Pyramid creation cost of \$48 (including fees) for the 2005 dataset (10 topics) and \$96 for 2006 (20 topics). Pyramid

---

[1] In our experiments, we have also examined the option of using the average answer, which was significantly worse.

[2] `https://www.mturk.com/`

	Pearson (ρ_p)		Spearman (ρ_s)	
	Ours	Expert Resp.	Ours	Expert Resp.
2005	0.81	0.81	**0.79**	0.77
2006	**0.74**	0.60	**0.69**	0.40

Table 1: Correlations to the original Pyramid scores, for our crowdsourced method and for *expert* Responsiveness method, for DUC '05 and '06.

creation cost per topic is thus \$4.8. For the system summary evaluation phase we split the 32 SCUs to two tasks of 16 SCUs each, in order to ensure that the crowdsourcing platform assigns each SCU to 5 distinct workers. We paid workers \$0.45, and evaluated all 25 (2005) and 22 (2006) systems. The total benchmark evaluation cost was \$1350 (including fees) for 2005 and \$2376 for 2006, equaling \$5.4 per system per topic, or \$108 per system evaluation over all 20 topics.

We release[3] our SCU dataset for DUC 2005 and DUC 2006 as a complementary resource, accompanied by the HTML pages for our tasks on Amazon Mechanical Turk and processing and evaluation scripts. In the SCU dataset, we mark the SCUs we used in our experiments, including their grouping as tasks in the system evaluation phase. These enable future crowdsourced Pyramid evaluations of new systems on these datasets, as well as developing new datasets with crowdsourced pyramids.

Correlations with original Pyramid. We first assess our evaluation methodology by computing the correlation of its system scores (and rankings) to those of the original Pyramid. These are compared with the analogous correlations for the expert Responsiveness scores, available in the datasets. As seen in Table 1, our method produces better correlations, and substantially so on the more characteristic 2006 dataset. Importantly, notice that Responsiveness scores here were obtained by *experts*, and therefore the gap for crowdsourced Responsiveness is expected to be greater, further indicating the advantage of our method as a crowdsourcable approach.

Stability. As an additional assessment, we test the robustness of our method, in terms of its reproducibility. To that end, we reran the system evaluation phase on eight randomly chosen systems of the 2006 data, which enabled us to compare our results with those obtained by Gillick and Liu (2010) for crowdsourced Responsiveness for a similar setting (8 random systems of the 2006 dataset). Notably, the lightweight Pyramid obtained an average 10% relative change in overall system scores, whereas crowdsourced Responsiveness exhibited lower stability with an average of 24% relative change.

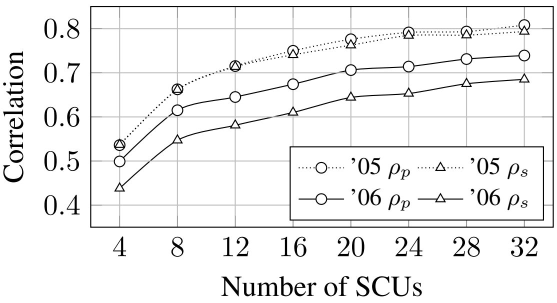


Figure 1: Average Pearson and Spearman correlations with Pyramid scores as a function of number of SCUs evaluated per topic, on the DUC '05 and '06 data.

Cost analysis. We analyze the impact of randomly reducing the various resources involved in our methodology, aiming to see whether overall cost might be reduced without harming correlation with the original Pyramid. The results below, reported as averages over 70 re-sampled iterations for each setting, suggest that such cost reductions would be harmful.

Number of workers. Reducing the number of workers per SCU judgment from five to three drops the correlations by about 8 points in 2006 and 6 points in 2005.

Number of SCUs. Figure 1 shows that correlation increases as a function of the number of judged SCUs per topic. The correlation improvement seems to stabilize around 32 SCUs.

Number of topics. Figure 2 presents the effect of the number of topics on which systems are evaluated, showing a steady correlation increase, which does not necessarily saturate at the number of 20 topics available in these datasets.

Qualitative analysis. To identify certain limitations of our methodology, we manually analyzed some "suspected" topics, for which either worker Krippendorff agreement or correlation with the original Pyramid was low. We noticed two interesting phenomena.

First, some topics seem inherently more difficult to evaluate, particularly for crowd workers. Such difficulty may be attributed to SCUs that are more difficult to assess or to less coher-

---

[3] `https://github.com/OriShapira/LitePyramids`

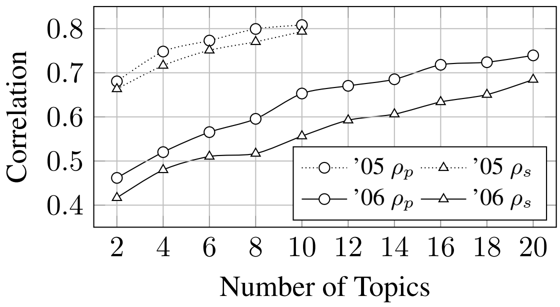


Figure 2: Average Pearson and Spearman correlations with Pyramid scores as a function of number of topics used for evaluation, on the DUC '05 and '06 data.

ent system summaries, due to the respective document set's complexity. Indeed, Passonneau et al. (2006) indicated that topic characteristics and annotator training experience effect evaluation quality. It seems worthwhile investigating, in future research, whether correlations improve by increasing further the overall number of topics, reducing the impact of the problematic ones.

Another possibility may be to filter out topics with low annotator agreement when computing systems' scores by the lightweight Pyramid method. We hypothesize that doing so might improve the reliability of this method, and hence increase its correlation with the original, expert-based, Pyramid method (when the latter is computed over all test topics). Indeed, in a preliminary test, we filtered out those 20% of the topics with lowest Krippendorff annotator agreement. This yielded a 6-point Spearman score increase (relative to the correlations reported in Table 1) when correlated with the original Pyramid ranking, as computed over the full set of topics. We note that while Figure 2 shows a slight decrease in average correlation when removing 4 random topics, removing specifically the 4 low-agreement topics seems to improves it notably. Further analysis might conclude that filtering problematic topics generically improves the reliability of the lightweight Pyramid method.

The second phenomenon observed among the difficult topics was that in some, the 32 sampled SCUs seem to miss important information, causing an unjustified degradation in system scores. In analogy to the variance in the number of SCUs in exhaustive Pyramids, it would be interesting to investigate methods for varying the sample size in our lightweight approach, based on some automatically detected parameters of topic complexity.

5 Conclusion and Future Work

To the best of our knowledge, our method is the first to mimic the reliable Pyramid method as an affordable crowdsourced procedure. Our experiments suggest that this lightweight Pyramid is more reliable than the common Responsiveness method. It also allows comparing multiple systems with absolute scores, which pairwise comparison does not.

Future work may improve correlation with the original Pyramid, or reduce annotation cost, by following our qualitative analysis and by reducing crowdsourcing noise (via qualification tests, enhanced guidelines, and post-processing result normalization (Hovy et al., 2013; Plank et al., 2014; Hosseini et al., 2012)). It would be appealing to investigate applying our methods to additional evaluation datasets, for which original Pyramid evaluations are not available for comparison. For example, addressing the CNN/DailyMail dataset (Nallapati et al., 2016) would involve testing single document summarization, utilizing a single reference summary per source text and addressing varying lengths of reference and system summaries.

The Pyramid method is mainly a measurement of recall, which thus also applies to our lightweight Pyramid; but other measurements for summary quality, such as precision, nonredundancy and grammaticality, may also be considered. In particular, it may be possible to extend our design of crowdsourcing tasks to supply indications for these complementary measurements as well.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments, as well as Ani Nenkova for her helpful remarks. This work was supported in part by the Bloomberg Data Science Research Grant Program; by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grants DA 1600/1-1 and GU 798/17-1); by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office; by the Israel Science Foundation (grants 1157/16 and 1951/17); by DARPA Young Faculty Award YFA17-D17AP00022; and by the ArguAna Project GU 798/20-1 (DFG).

References

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 643–653.

Hoa Trang Dang. 2006. Overview of duc 2006. In *Proceedings of the document understanding conference*.

Tobias Falke, Christian M. Meyer, and Iryna Gurevych. 2017. Concept-map-based multi-document summarization using concept coreference resolution and global importance optimization. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pages 801–811.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Association for Computational Linguistics.

Yang Gao, Christian M. Meyer, and Iryna Gurevych. 2018. April: Interactively learning to summarise by combining active preference learning and reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4120–4130. Association for Computational Linguistics.

Dan Gillick and Yang Liu. 2010. Non-expert evaluation of summarization systems is risky. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 148–151. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Tsutomu Hirao, Hidetaka Kamigaito, and Masaaki Nagata. 2018. Automatic pyramid evaluation exploiting edu-based extractive reference summaries. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4177–4186.

Mehdi Hosseini, Ingemar J Cox, Nataša Milić-Frayling, Gabriella Kazai, and Vishwa Vinay. 2012. On aggregating labels from multiple crowd workers to infer relevance of documents. In *European Conference on Information Retrieval*, pages 182–194. Springer.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004*.

NIST. 2003. Duc 2003: Documents, tasks, and measures. `https://duc.nist.gov/duc2003/tasks.html`.

NIST. 2014. Document understanding conferences. `https://duc.nist.gov/`.

NIST. 2018. Text analysis conference. `https://tac.nist.gov/`.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9. Association for Computational Linguistics.

Rebecca Passonneau. 2006. Pyramid annotation guide: Duc 2006. `http://www1.cs.columbia.edu/~becky/DUC2006/2006-pyramid-guidelines.html`.

Rebecca Passonneau, Kathleen McKeown, Sergey Sigelman, and Adam Goodkind. 2006. Applying the pyramid method in the 2006 document understanding conference.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *Sixth International Conference on Learning Representations*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (volume 2: Short Papers)*, volume 2, pages 507–511.

Qian Yang, Rebecca J. Passonneau, and Gerard de Melo. 2016. Peak: Pyramid evaluation via automated knowledge extraction. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*. AAAI Press.

Serial Recall Effects in Neural Language Modeling

Hassan Hajipoor[1], **Hadi Amiri**[2], **Maseud Rahgozar**[1], **Farhad Oroumchian**[3]
[1]School of Electrical and Computer Engineering, University of Tehran, Iran
[2]Department of Biomedical Informatics, Harvard University, Boston, USA
[3]University of Wollongong in Dubai, Knowledge Village, Dubai, UAE
`h.hajipoor@ut.ac.ir`, `hadi@hms.harvard.edu`,
`rahgozar@ut.ac.ir`,`farhadoroumchian@uowdubai.ac.ae`

Abstract

Serial recall experiments study the ability of humans to recall words in the order in which they occurred. The following serial recall effects are generally investigated in studies with humans: *word length* and *frequency*, *primacy* and *recency*, *semantic confusion*, *repetition*, and *transposition* effects. In this research, we investigate LSTM language models in the context of these serial recall effects. Our work provides a framework to better understand and analyze neural language models and opens a new window to develop accurate language models.

1 Introduction

The goal of language modeling is to estimate the probability of a sequence of words in natural language, typically allowing one to make probabilistic predictions of the next word given preceding ones (Bahl et al., 1983; Berger et al., 1996). For several years now, Long Short-Term Memory (LSTM) (Graves, 2013) language models have demonstrated state-of-the-art performance (Melis et al., 2018; Merity et al., 2018a; Sundermeyer et al., 2012). Recent studies have begun to shed light on the information encoded by LSTM networks. These models can effectively use distant history (about 200 tokens of context) and are sensitive to word order, replacement, or removal (Khandelwal et al., 2018), can learn function words much better than content words (Ford et al., 2018), can remember sentence lengths, word identity, and word order (Adi et al., 2017), can capture syntactic structures (Kuncoro et al., 2018) such as subject-verb agreement (Linzen et al., 2016). These characteristics are often attributed to LSTM's ability in overcoming the *curse of dimensionality*–by associating a distributed feature vector to each word (Hinton et al., 1986; Neubig, 2017)–and modeling long-range dependencies in faraway context (Khandelwal et al., 2018).

The goal of our research is to complement the prior work to provide a richer understanding about how LSTM language models use prior linguistic context. Inspired by investigations in cognitive psychology about serial recall in humans (Avons et al., 1994; Henson, 1998; Polišenská et al., 2015)–where participants are asked to recall a sequence of items in order in which they were presented, we investigate how word length or frequency (*word-frequency* effect), word position (*primacy*, *recency*, and *transposition* effects), word similarity (*semantic confusion* effect), and word repetition (*repetition* effect) influence learning in LSTM language models. Our investigation provides a framework to better understand and analyze language models at a considerably finer-grained level than previous studies, and opens a new window to develop more accurate language models.

We find that LSTM language models (a) can learn frequent/shorter words considerably better than infrequent/longer ones, (b) can learn recent words in sequences better than words in earlier positions,[1] (c) have a tendency to predict words that are semantically similar to target words - indicating that these networks have a tendency to group semantically similar words while suggesting one specific word as target based on prior context, (d) predict as output the words that are observed in prior context, i.e. repeat words from prior context, and (e) may transpose (switch adjacent) words in output depending on word syntactic function.

---

[1] Rats and humans recall the first and last items of sequences best and the middle ones worst (Bolhuis and Van Kampen, 1988; Ebbinghaus, 1913).

Proceedings of NAACL-HLT 2019, pages 688–694
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Serial Recall Effects

Language models estimate the probability of a sequence as $P(w_1^n) = \prod_{i=1}^n P(w_i|w_1^{i-1})$, where w_i is the ith word and w_i^j indicates the sub-sequence from w_i to w_j. These models minimize their prediction error against words in context, using e.g. the negative log likelihood loss function, during training: $\mathcal{L} = -\frac{1}{n}\Sigma_{i=1}^n \log P(w_i|w_1^{i-1})$. In this paper, we show the loss of a model against sequence s and word $w_i \in s$ by $\mathcal{L}(s)$ and $\mathcal{L}(s)[w_i]$ respectively.

We use the LSTM language model developed in (Merity et al., 2018b) for our experiments. Given a sequence of words w_1^{i-1} as context, this model predicts the next word in sequence. We refer to w_i and $\hat{w}_i$ as *target* and *predicted* words respectively; given the global vocabulary $\mathcal{V}$, $\hat{w}_i = \arg\max_{w_j \in \mathcal{V}} \Pr(w_j|w_1^{i-1})$. We study this LSTM in the context of serial recall effects.

2.1 Word-Frequency Effect

What is the effect of word frequency/length[2] on the performance of LSTM language models? For this effect, we report the average loss for each word frequency as follows:

$$\mathcal{L}_{WF}^k = 1/|\mathcal{S}_k| \sum_{s \in \mathcal{S}_k, w_i \in s} \mathcal{L}(s)[w_i], \qquad (1)$$

where $\mathcal{S}_k$ is the set of sequences that, at least, have one target word with term frequency of k, and $\mathcal{L}_{WF}^k$ is the overall loss for target words of frequency k which sheds light on the expected frequency of words for accurate language modeling.

2.2 Primacy and Recency Effect

What is the effect of word position on the performance of LSTM language models? To analyze this effect, we compute the average loss of network with respect to the position of target words as follows:

$$\mathcal{L}_{PR}^i = 1/Z \sum_s \mathcal{L}(s)[w_i], \qquad (2)$$

where w_i is the target word in sequence s, Z is the number of sequences as normalization factor, and $\mathcal{L}_{PR}^i$ is the average of loss at position i. This effect will shed light on network performance at specific positions in texts which can help rationalizing the need for new language modeling architectures, such as bidirectional LSTMs (Schuster and Paliwal, 1997), or the order by which input data should be processed (Sutskever et al., 2014).

2.3 Semantic Confusion Effect

Are predicted words semantically similar to the target ones in case of incorrect predictions? For this analysis, we report the average semantic similarity between target (w_i) and predicted ($\hat{w}_i$) words as follows:

$$\mathcal{SC} = 1/Z \sum_{s, w_i \in s, w_i \neq \hat{w}_i} \mathtt{sim}(w_i, \hat{w}_i), \qquad (3)$$

where the function $\mathtt{sim}(.,.)$ computes word similarity either through WordNet (Miller, 1998) or `cosine` similarity of the corresponding embeddings of its arguments. This effect will shed light on how effective LSTMs are in disentangling semantically-similar concepts. This gives us a powerful metric to compare networks semantically, especially, in case of equal loss/perplexity.

2.4 Repetition Effect

This effect refers to prediction of a word that already exists in context, i.e. in an earlier position in the sequence. Here, for each target w_n, we compute the probability that, instead of w_n, any of w_1^{n-1} words is predicted as output and report average across all samples:

$$\Pr(RE^i) = \qquad (4)$$
$$1/Z \sum_s \Pr(w_i \in w_1^{n-1}, w_i \neq w_n | w_1^{n-1}).$$

This effect will shed light on the extent to which network repeats/predicts observed words as possible responses. Given that words rarely repeat in sentences, the above metric can be used as a good regularizer for language modeling.

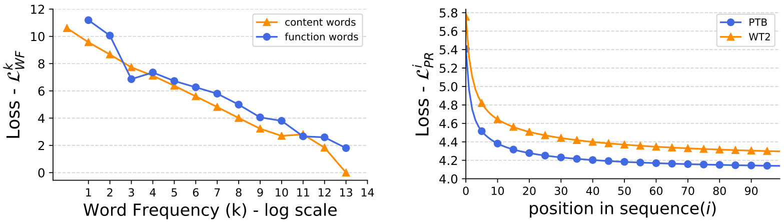
2.5 Transposition Effect

This effect refers to word prediction in transposed positions, i.e. the case where the word pair w_i^{i+1} in an original sequence is more likely to be predicted by the network as $w_{i+1}w_i$ in output. Here, for each pair w_i^{i+1} in target sequence, we count the number of times in which w_{i+1} is more probable to be predicted at position i (as compared to w_i) and w_i is more probable to be predicted at position $i+1$ (as compared to w_{i+1}). We report the average number of transposition occurrences for all samples at each word position. This effect will shed light on how network learn nearby grammatical orders such as conjunction and adjective order.

---

[2] Note that word length and frequency are directly correlated (Bell et al., 2009).

	PTB		WT2	
	#tokens	#words	#tokens	#words
NN	21k	3k	55k	8k
VB	10k	1.7k	25k	3k
RB	2.6k	286	5k	0.5k
JJ	4.9k	0.9k	12k	1.8k
Func	34.9k	0.2k	118k	0.9k



(a) Dataset statistics (b) Word frequency vs. loss (PTB) (c) Primacy and Recency effect

Figure 1: (a) Dataset statistics, (b): Word-frequency effect, network learns frequent words better than infrequent ones, (c): Primacy and Recency effect, loss decreases for target words that appear at the end of sequences.

3 Experiments

Datasets. We use two benchmark language modeling datasets: Penn Treebank (PTB) (Marcus et al., 1993; Mikolov et al., 2010) and WikiText-2 (WT2) (Merity et al., 2017). PTB and WT2 have vocabulary sizes of 10K and 33K respectively. We use the POS-tagged versions of these datasets provided by Khandelwal et al. (2018), and treat nouns (NN), verbs (VB), adjectives (JJ), and adverbs (RB) as content words, and others word classes as function words, see details in Figure 1a.

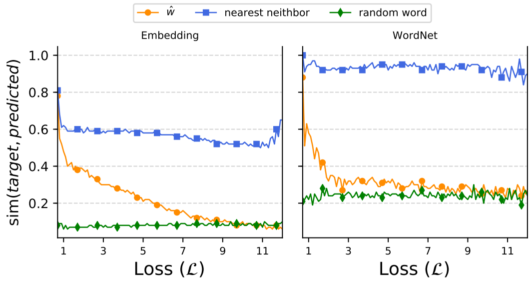
Settings. We set LSTM's parameters as suggested in (Merity et al., 2018b) for PTB and follow its suggested parameter tuning procedure for WT2. For both datasets, we set context size to $n = 100$ obtained from $\{5, 20, 50, 100, 200\}$ and validation data; note that the number of samples are equal for different sequence lengths.

3.1 Results

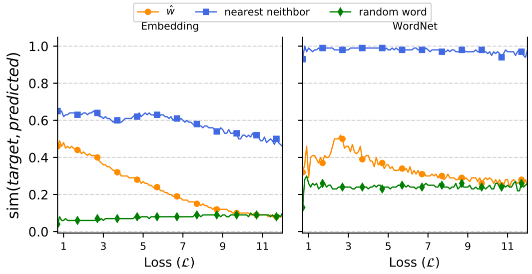
We report LSTM performance in terms of prediction loss on development sets for all experiments.

Word-Frequency Effect: *More frequent target words are predicted (learned) more accurately than less frequent ones.* Figure 1b shows strong inverse correlation between word frequency and LSTM prediction loss. This is expected as neural models learn better with more data. In addition, although the overall loss of function words is considerably lower than that of content words (because of their overall higher frequency), Figure 1b shows that, for the same word frequency, content words are learned better than function words.

Primacy and Recency Effects: *Target words that appear later in sequences are predicted considerably better than those at earlier positions.* Figure 1c shows that prediction loss considerably decreases for target words that appear toward the end of the sequences. The results are consistent across both datasets. This effect can explain why bidirectional LSTMs which read input from opposite directions usually work better in NLP applications such as machine translation (Firat et al., 2016; Domhan, 2018). A remaining question that is worth investigating is whether bidirectional LSTMs learn first and last few words of sequences better than those in the middle, and if yes, how can we make these models more robust against word position.



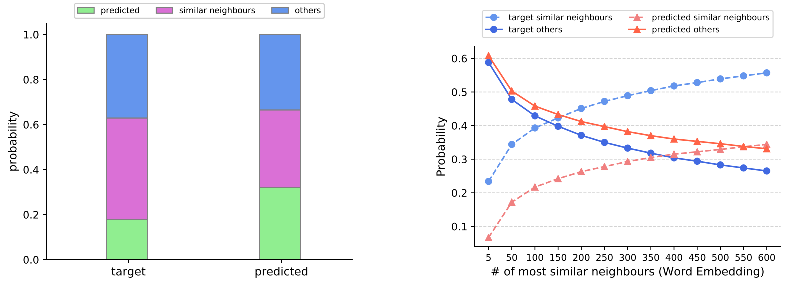
(a) Semantic confusion effect in PTB.



(b) Semantic confusion effect in WT2.

Figure 2: We report semantic similarity between predicted and target words, random and target words (lower bound), and nearest neighbor and target words (upper bound).

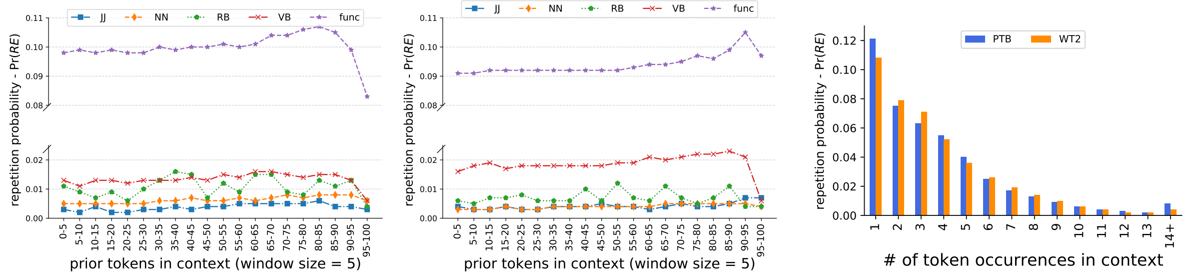
Semantic Confusion Effect. *There is significant tendency to predict words that resemble (are semantically-similar to) target words.* Figure 2 shows the average WordNet and Embedding similarity between target and predicted words in PTB and WT2 across loss values. The results indicate high similarity between predicted and target words



(a) Proportion of neighbors from prediction probability (1 − Loss)

(b) Finding appropriate neighbors size

Figure 3: Semantic confusion effect. (a) The chance of neighbors of target is equal to chance of other words. (b) When the neighbours size is set to 150, the probability of target neighbouring group became equal to the probability of target. Neighbours size of 600, leads to equal chance for predicted token.



(a) Repetition effect in PTB

(b) Repetition effect in WT2

(c) Repetition vs. occurrence in context

Figure 4: (a, b): Repetition effect across POS tag classes in PTB and WT2 and (c): Repetition probability decreases as a function of word occurrences in prior context.

for smaller loss values. However, confusion consistently increases as prediction loss increases. The upper bound similarity (obtained by treating the nearest neighbor of each target word as predicted word) indicates there exists better candidates which LSTM fails to predict. Our further analyses show that LSTM has a tendency to group semantically similar words and then suggest one of them. We consider most similar words as a group of neighbors and examine how network assigns probabilities to them as compared to others. Figure 3a shows that the chance of neighbors of target (with size 150) is equal to chance of other words (with size 9849). As Figure 3a shows, the neighbors of predicted words (with size 600) carry equal chance as compared to other words (with size 9399). To find these thresholds, we gradually increase the number of neighbors and track the trend of approaching the probability of neighboring group to target. As shown in Figure 3b, if the size of neighboring group is set to 150, these probabilities became equal. The similar way is repeated for finding the appropriate size for neighbors of predicted words.

Repetition Effect. *Repetition probability of function words is significantly higher than that of content words.* We report repetition effect, see Eq. (4), at POS tag level (where the predicted word should have the same POS tag as target word in prior context). As Figure 4a and 4b show, function words have higher repetition probability than content words. This is because function words are more frequent, and the average distance among them (i.e. number of intervening words) is considerably smaller than other POS tags (e.g. 2.1 words vs 28.9 and 12.4 words for `RB` and `JJ` respectively). We also find that repetition probability decreases as a function of word frequency in prior context, see Figure 4c. This is because words (especially `NN`s and `VB`s) are often self-contained and their occurrence in prior context helps LSTM to avoid "repeating" them. In addition, we find that function words repeat more frequently than other types and repetition among `NN`s and `VB`s is higher than other POS tag pairs; Table 1 shows the confusion matrix for repetition across POS tag classes Perhaps, this could explain the recent language modeling improvement obtained in (Ford et al., 2018) through developing separate yet sequentially connected network architectures with respect to POS tag class.

There are two factors determining the chance of

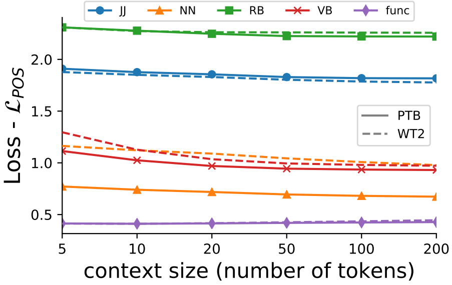


Figure 5: Effective context to learn syntax.

	NN	VB	JJ	RB	Func
NN	**0.07**	0.00	0.01	0.00	**0.24**
VB	0.01	**0.09**	0.00	0.01	**0.20**
JJ	0.03	0.01	0.01	0.00	**0.27**
RB	0.01	0.02	0.00	0.03	**0.31**
Func	0.01	0.01	0.00	0.00	**0.52**

Table 1: Confusion matrix for repetition effect. Rows and columns show POS tag classes of target and predicted words respectively.

repetition of a POS class: First, the average distance of consecutive tokens of that POS class; Table 2 reports the corresponding values from training set. Second, the accuracy of network in predicting POS classes which has been shown in Figure 5 and also reported in (Khandelwal et al., 2018). From these, the repetition probability of function words are expected to be higher than content words.

	NN	VB	JJ	RB	Func
PTB	3.6	7.1	12.4	28.9	2.1
WT2	3.6	8.5	15.8	38.5	1.9

Table 2: Average distance of tokens in POS classes.

Transposition Effect: *Transpositions occur more frequently at the beginning of sequences and rarely at the end.* Figure 6 shows average number of transpositions at each word position across datasets. This result is meaningful because miss-predictions occur more frequently at earlier positions (see results for primacy and recency effect). In addition, transpositions are rare at higher positions because more context information can help LSTM to make accurate (transposition-free) prediction. In addition, Table 3 shows the percentage of transpositions across POS tag classes on PTB. The result show that LSTM mainly transposes ‘`RB NN`’ word pairs with ‘`NN RB`.’ In future, we will conduct further analyses to understand the reason.

The findings presented in this paper provide insight into how LSTMs model context. This information can be useful for improving language models. For instance, the discovery that some word types are repeated in predictions more than others can help regularizing neural language models by making them adaptive to the different word types.

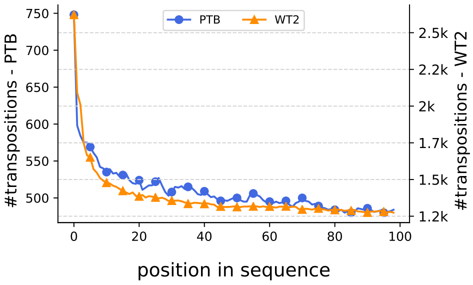


Figure 6: Transposition effect, transpositions occur more at the beginning of sequences.

	NN	VB	JJ	RB	Func
NN	0.025%	0.006%	0.032%	0.015%	0.001%
VB	0.011%	0.001%	0.004%	0.002%	0.001%
JJ	0.021%	**0.054%**	**0.045%**	**0.036%**	0.002%
RB	**0.073%**	0.017%	0.017%	0.023%	0.012%
Func	0.002%	0.002%	0.002%	0.001%	0.006%

Table 3: Confusion matrix for transpositions in PTB at POS tag level. LSTM transposes ‘`RB NN`’ and ‘`JJ VB|JJ|RB`,’ more than others pairs.

4 Conclusion

We investigate LSTM language models in the context of serial recall indicators. We find that frequent target words and target words that appear later in sequences are predicted more accurately, predictions often resemble (are semantically similar to) target words, function words of prior context are more likely to be predicted as target words, and word pair transpositions occur more frequently at the beginning of sequences.

Acknowledgments

We sincerely thank Nazanin Dehghani for her constructive collaboration during development of this paper and anonymous reviewers for their insightful comments and effective feedback.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *International Conference on Learning Representations (ICLR)*.

SE Avons, KL Wright, and Kristen Pammer. 1994. The word-length effect in probed and serial recall. *The*

Quarterly Journal of Experimental Psychology Section A, 47(1):207–231.

Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2):179–190.

Alan Bell, Jason M Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. 2009. Predictability effects on durations of content and function words in conversational english. *Journal of Memory and Language*, 60(1):92–111.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Johan J Bolhuis and Hendrik S Van Kampen. 1988. Serial position curves in spatial memory of rats: Primacy and recency effects. *The Quarterly Journal of Experimental Psychology*, 40(2):135–149.

Tobias Domhan. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1799–1808.

H Ebbinghaus. 1913. Memory: A contribution to experimental psychology (ha ruger & ce bussenius, trans.). new york, ny, us.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of NAACL-HLT*, pages 866–875.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. 2018. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2942–2946.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Richard NA Henson. 1998. Short-term memory for serial order: The start-end model. *Cognitive psychology*, 36(2):73–137.

Geoffrey E Hinton et al. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. *International Conference on Learning Representations*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018a. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018b. Regularizing and optimizing lstm language models. In *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Kamila Polišenská, Shula Chiat, and Penny Roy. 2015. Sentence repetition: what does the task measure? *International Journal of Language and Communication Disorders*, 50(1):106–118.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Martin Sundermeyer, Ralf Schlúter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Fast Concept Mention Grouping for Concept Map–based Multi-Document Summarization

Tobias Falke and **Iryna Gurevych**
Research Training Group AIPHES and UKP Lab
Department of Computer Science, Technische Universität Darmstadt
`https://www.aiphes.tu-darmstadt.de`

Abstract

Concept map–based multi-document summarization has recently been proposed as a variant of the traditional summarization task with graph-structured summaries. As shown by previous work, the grouping of coreferent concept mentions across documents is a crucial subtask of it. However, while the current state-of-the-art method suggested a new grouping method that was shown to improve the summary quality, its use of pairwise comparisons leads to polynomial runtime complexity that prohibits the application to large document collections. In this paper, we propose two alternative grouping techniques based on locality sensitive hashing, approximate nearest neighbor search and a fast clustering algorithm. They exhibit linear and log-linear runtime complexity, making them much more scalable. We report experimental results that confirm the improved runtime behavior while also showing that the quality of the summary concept maps remains comparable.[1]

1 Introduction

Concept maps are labeled graphs with nodes representing concepts and edges showing relationships between them (Novak and Gowin, 1984). Following earlier work on the automatic extraction of concept maps from text (Rajaraman and Tan, 2002; Valerio and Leake, 2006; Villalon, 2012; Zubrinic et al., 2015), concept maps have recently been promoted as an alternative representation for summaries (Falke and Gurevych, 2017; Handler and O'Connor, 2018). In the corresponding task, concept map–based multi-document summarization (CM-MDS), a set of documents has to be automatically summarized as a concept map that does not exceed a pre-defined size limit.

An important subtask of CM-MDS is *concept mention grouping*, in which all mentions that refer to a specific concept should be grouped together. Without grouping, duplicates can appear in a summary concept map that make the map harder to understand and that waste valuable space.

To approach the mention grouping subtask, Falke et al. (2017) proposed to make pairwise coreference classifications between mentions and to induce a partitioning from those predictions. Their experiments showed that this leads to better summary concept maps, establishing the current state-of-the-art for CM-MDS. However, the computational costs of the approach are high, as it exhibits a $\mathcal{O}(n^4)$ worst-case time complexity. When the number of documents that should be summarized is large, applying that technique can quickly become impractical. But exactly for those large document sets, a summary would be most helpful.

As the first contribution of this paper, we propose two faster grouping techniques. First, we apply *locality sensitive hashing (LSH)* (Charikar, 2002) to word embeddings in order to find similar mentions without making all pairwise comparisons. That directly leads to a simple $\mathcal{O}(n)$ grouping method. Second, we also propose a novel grouping technique that combines the hashing approach with a fast partitioning algorithm called *Chinese Whispers (CW)* (Biemann, 2006). It has $\mathcal{O}(n \log n)$ time complexity and the advantage of being more transparently controllable.

Since the reduced complexity of the two proposed techniques is gained through approximations, the resulting grouping could of course be of lower quality. As the second contribution of this paper, we therefore carry out end-to-end experiments in the context of CM-MDS to analyze this trade-off. We compare both techniques against the state-of-the-art approach in automatic and manual evaluations. For both, we observe orders of mag-

---

[1] Code used for experiments available at `https://github.com/UKPLab/naacl2019-cmaps-lshcw`

Proceedings of NAACL-HLT 2019, pages 695–700
Minneapolis, Minnesota, June 2 - June 7, 2019.

nitude faster runtimes with only small reductions in summary quality. In the future, the techniques could also be applied beyond CM-MDS to speed up other similarity-based partitioning problems in NLP and its applications.

2 Problem and Reference Approach

Given a set of concept mentions M identified in the input documents, the goal of concept mention grouping is to derive a partitioning C of M such that for every set of mentions in C, the set contains all mentions and only mentions of one unique concept. Let n denote the number of mentions $|M|$.

Previous work on concept map mining used stemming (Villalon, 2012), substring matches (Valerio and Leake, 2006) or WordNet (Aguiar et al., 2016) to detect coreferences between mentions. Falke et al. (2017) combined several of those features, including semantic similarities based on WordNet (Miller et al., 1990), latent semantic analysis (Deerwester et al., 1990) and word2vec embeddings (Mikolov et al., 2013), in a log-linear classifier to predict coreferences of mentions.

Since such pairwise predictions can be inconsistent, e.g. the model might classify (m_1, m_2) and (m_2, m_3) as coreferent, but not (m_1, m_3), Falke et al. (2017) further induce a transitive relation from the predictions to obtain a valid partitioning of M. They note that simply ignoring conflicting negative classifications by building the transitive closure over all positive ones typically yields undesired partitionings in which too many mentions are being lumped together. Following previous work on related NLP tasks (Barzilay and Lapata, 2006; Denis and Baldridge, 2007), they instead formulate an integer linear program (ILP) to find the transitive relation that maximally agrees with all pairwise predictions. However, as the resulting ILPs cannot be efficiently solved on the data they work with, they propose a local search algorithm that incrementally improves a greedy solution rather than finding the optimal partitioning,

This technique requires making classifications for all pairs of mentions in $\mathcal{O}(n^2)$ time and running the local search, which has a worst-case complexity of $\mathcal{O}(n^4)$. As we will show in Section 6, that can quickly become prohibitively expensive.

3 Locality Sensitive Hashing

The central idea of LSH is that specific families of hash functions can approximately preserve similarities. Charikar (2002) introduced such a family for cosine similarity between vectors.

3.1 Approximating Cosine Similarity

Let u, v be k-dimensional vectors. First, choose d unit random vectors $r_1, \ldots, r_d$ of k dimensions by sampling every dimension independently from a standard normal distribution. Then, for a vector u, compute a d-dimensional bit vector $h(u)$, the hash, with the i-th dimension defined as

$$h(u)_{[i]} = \begin{cases} 1 & : u \cdot r_i \geq 0 \\ 0 & : u \cdot r_i < 0, \end{cases} \tag{1}$$

where $u \cdot r_i$ is the dot product with the i-th random vector. The Hamming distance ham between two hashes $h(u)$ and $h(v)$, i.e. the number of differing bits, can then be used to approximate the cosine similarity of u and v (Charikar, 2002):

$$\frac{u \cdot v}{|u||v|} \approx \cos\left(\frac{\mathrm{ham}(h(u), h(v))}{d}\,\pi\right) \tag{2}$$

The longer the hashes are, i.e. the larger d is, the more accurate is the estimation of the similarity.

In the past, LSH has been successfully used to speed up a range of NLP tasks, including noun similarity list construction (Ravichandran et al., 2005), word sense induction (Mouton et al., 2009), gender classification (van Durme, 2012) and text classification (Bollegala et al., 2018).

3.2 Naive Partitioning

Given the mapping h from vectors to their bit hashes, we can partition a set of vectors by hash identity. Every unique hash becomes a group consisting of all vectors mapped to that hash. Since the hashes reflect similarity, the most similar vectors will be grouped together. The parameter d controls the degree of grouping: the smaller it is, the less unique hashes and thus fewer groups exist.

In order to apply this technique to concept mention grouping, every mention $m \in M$ has to be represented by a vector in a space where the cosine similarity is indicative of coreference. Since the classifier of Falke et al. (2017) already uses cosine similarity of word2vec embeddings as a feature, we also use those vectors for LSH.[2] Both the computation of the hashes and building groups can be done with a single pass over the mentions. Assuming d and k to be fixed, the overall time complexity of the grouping technique is thus $\mathcal{O}(n)$.

---

[2] Following their work, we represent a mention by the mean of the embedding vectors of the mention's tokens.

4 Fast Nearest Neighbor Partitioning

When grouping similar elements together, one typically wants to control the degree of grouping by defining a similarity threshold δ. For the naive LSH-based partitioning, we can only set d, which does not directly correspond to a similarity. Therefore, we propose a second, more transparent grouping technique with this property.

4.1 Approximate Nearest Neighbor Search

Given vectors and their LSH-based hashes, we can use *approximate nearest neighbor search (ANNS)* to find pairs with a cosine similarity of at least δ (Charikar, 2002; Ravichandran et al., 2005) without making all pairwise comparisons:

1. Sample q permutations of the bit hashes.
2. For each permutation, sort all mentions M according to their permuted hashes.
3. In each sorted list, estimate the cosine similarity of each $m \in M$ with the next b mentions based on the hashes. Keep pairs with a similarity of at least δ.

Since comparing neighbors in a sorted list of bit hashes will primarily find those that differ in the last positions, the random permutations are the key part of the algorithm that ensures similar hashes differing at varying positions are found. Rather than comparing each vector to all others in $\mathcal{O}(n^2)$, only qb comparisons are made for each. The dominant part becomes the sort, resulting in $\mathcal{O}(n \log n)$ time complexity as q and b are constants.

4.2 Chinese Whispers Partitioning

Using ANNS we can obtain an undirected graph of mentions connected with edges if their similarity is at least δ. However, as Falke et al. (2017) observed, simply taking the transitive closure over these pairs tends to yield too big groups that lump many mentions of different concepts together.

Rather than relying on the expensive $\mathcal{O}(n^4)$ local search of Falke et al. (2017) to address this problem, we here resort to the fast graph partitioning algorithm CW (Biemann, 2006). Given a graph $G = (V, E)$, it proceeds as follows:

1. Label nodes initially as $l(v_i) = i \;\; \forall \, v_i \in V$.
2. Iterate over V in randomized order. For each $v \in V$, set $l(v)$ to the label most frequent among the nodes reachable via a direct edge.
3. If at least one label changed, repeat step 2.

While it cannot be guaranteed in general, the algorithm typically converges to a stable labeling after a few iterations. Then, nodes having the same label form a group of the partitioning. In contrast to the local search, CW does not directly optimize the objective function proposed by Falke et al. (2017), however, we empirically found that it yields partitionings that score very well with regard to that objective. To guarantee termination, the number of iterations is bound by a parameter ϵ. Then, CW iterates at most ϵ times over n nodes and their at most $n - 1$ edges, resulting in $\mathcal{O}(n^2)$ complexity.

4.3 Combination

For concept mention grouping, we combine these techniques as follows: First, we represent each mention with a vector and compute its LSH-based hash. Second, we use ANNS to find pairs with a similarity of at least δ. Finally, we partition the resulting nearest neighbor graph with CW.

That grouping technique has four parameters δ, d, q and b. While δ determines the degree of grouping, d influences the quality of the similarity estimates and q and b define the size of the search space explored to find nearest neighbors. Note that the construction of the nearest neighbor graph guarantees that a node has at most qb edges, reducing the runtime of CW to $\mathcal{O}(n)$ in this setting. The runtime behavior of the combination is therefore dominated by ANNS and thus $\mathcal{O}(n \log n)$.

5 Experimental Setup

We evaluate the proposed concept mention grouping techniques for the task of CM-MDS.

Data and Metrics We use the benchmark corpus introduced by Falke and Gurevych (2017), the only existing dataset with manually created reference summary concept maps. It provides reference summaries for document sets of web pages on 30 different topics. As metrics, we compute the ROUGE and METEOR variants proposed with the dataset and also perform a human evaluation following the protocol of Falke et al. (2017).

Implementation As the *reference*, we use the state-of-the-art pipeline of Falke et al. (2017).[3] We test the naive LSH-based partitioning (*LSH-only*)

---

[3] `https://github.com/UKPLab/ijcnlp2017-cmaps`

	Average		Smallest		Largest	
Approach	Count	Runtime	Count	Runtime	Count	Runtime
Mentions	*5299*		*2475*		*13572*	
Reference	4029	3h 12m 32s	1847	24m 21s	10131	22h 48m 08s
LSH-only	3694	1s	1752	1s	7827	2s
LSH-CW	4085	23s	1875	11s	9861	58s

Table 1: Concept mention grouping runtimes on average and for the smallest and largest set. *Count* is the number of concepts after grouping the mentions given in the first row. Runtimes are measured on the same machine.

	METEOR			ROUGE-2		
Approach	Pr	Re	F1	Pr	Re	F1
Reference	15.1	17.3	16.1	9.4	11.9	10.4
lemma-only	13.9	15.4	*14.6*	8.2	8.6	*8.3*
w2v-only	14.1	16.2	*15.0*	8.3	9.9	*8.9*
LSH-only	14.9	16.9	15.8	9.1	11.2	9.9
LSH-CW	14.9	17.1	15.9	8.2	10.9	9.3

Table 2: Evaluation results for summary concept maps. Italics denote F1-scores that are significantly different from *Reference* (exact permutation test, $\alpha = 0.05$).

and the combined approach (*LSH-CW*) by substituting them into that pipeline. For a fair comparison, we use the same 300-dimensional word2vec embeddings (Mikolov et al., 2013) for LSH that have also been used in the log-linear model.

Tuning In the reference pipeline, the regularization constant of the scoring SVM was tuned with leave-one-out cross-validation on the training set. For LSH-only, we use the same procedure to tune d (together with regularization) and found $d = 17$ to be best (testing 10, 11, ..., 25). For LSH-CW, where four hyper-parameters have to be set, running cross-validation for the whole grid is too expensive. We instead evaluate a grid of 130 $d/q/b/\delta$-combinations by concept F1-score after grouping and tune the SVM with cross-validation only for the three best settings, leading to the parameters $d = 200, q = 20, b = 200, \delta = .89$.

6 Results

Runtime Table 1 shows the runtimes for grouping concept mentions.[4] It demonstrates two problems of the reference: First, even on the smallest document set (37 docs, 50k tokens), the grouping already takes hours. And second, on the biggest set (42 docs, 220k tokens), the runtime grows to almost a day, illustrating the analyzed time complexity. Applying the technique to more documents quickly becomes infeasible. Our newly proposed techniques, LSH-only and LSH-CW, are orders of magnitude faster in absolute terms and also show a more moderate runtime growth as expected given their preferable time complexity.

Comparison	Fo	Gr	Me	NR
Reference vs. LSH-only	47.3	47.3	46.7	42.7
Reference vs. LSH-CW	57.3	58.0	58.0	56.7
LSH-CW vs. LSH-only	52.7	50.7	50.0	47.3

Table 3: Human summary preferences, shown as the percentage of annotators preferring the first option.

Quality A crucial question is which price we have to pay for improving runtimes through approximations. Table 2 shows the automatic evaluation results for the created summaries. We included *lemma-only*, a baseline from previous work using lemmatization for grouping, and *w2v-only*, a variation of the reference grouping approach that uses embeddings as the only feature in the coreference classifier. The latter is important for comparison, as it uses the same information as the LSH-based techniques. While lemma-only and w2v-only perform significantly worse than the reference, the two LSH-based techniques come much closer to the more expensive reference.

Table 3 shows the results of our human evaluation. Following previous work, we collected pairwise preferences among the created summaries via Mechanical Turk (150 per pairing) for the dimensions focus (Fo), grammaticality (Gr), meaningfulness (Me) and non-redundancy (NR).[5] As shown, the preferences we collected are almost balanced and annotators repeatedly noted during the study that the summaries are very similar. None of the 12 preferences are significant at $\alpha =$

---

[4] Measured on an Intel Xeon ES-2620 2.1GHz processor.

[5] We payed $0.60 per comparison and anonymized worker IDs. The study was approved by the university's ethics committee and we obtained informed consent from participants.

0.05 (binomial test), showing that the alternative summary concept maps are practically indistinguishable. In contrast, Falke et al. (2017) observed preferences of up to 79% in their study.

Conclusion Based on the automatic and human evaluations, we conclude that both fast grouping techniques proposed in this paper do not substantially decrease the quality of the summaries. Since there is also no clear difference between LSH-only and LSH-CW, we recommend both techniques, which allows practitioners to choose between more transparency or even faster runtimes.

Future Work The comparison of w2v-only and the reference in Table 2 reveals that relying only on word2vec and dropping the other features of the log-linear model hurts performance, suggesting that also adding the remaining features to the LSH techniques could lead to further improvements. However, all other features of the reference model are pairwise features, which makes it difficult to incorporate them in the LSH-based techniques that only use mention features. As an alternative direction, one could instead rely on more powerful word embeddings. While we used word2vec to ensure comparability to previous work, using more recent embedding methods such as fastText (Bojanowski et al., 2017), InferSent (Conneau et al., 2017) or ELMO (Peters et al., 2018) seems to be worth exploring in the future.

7 Summary

In this paper, we proposed two fast concept mention grouping techniques for CM-MDS, the direct application of LSH and a novel combination of LSH and Chinese Whispers. Our analysis and experiments show that they are orders of magnitude faster than previous techniques with only small effects the quality of the resulting summary concept maps. Using these techniques, summary concept maps can now be created for much larger document sets than what was possible before.

Acknowledgements

We would like to thank Kevin Mayer for his support during preliminary experiments leading to this paper. This work has been supported by the German Research Foundation as part of the Research Training Group “Adaptive Preparation of Information from Heterogeneous Sources” (AIPHES) under grant No. GRK 1994/1.

References

Camila Z. Aguiar, Davidson Cury, and Amal Zouaq. 2016. Automatic Constrution of Concept Maps from Texts. In *Proceedings of the 7th International Conference on Concept Mapping*, pages 20–30, Tallinn, Estonia.

Regina Barzilay and Mirella Lapata. 2006. Aggregation via Set Partitioning for Natural Language Generation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366, New York, NY, USA.

Chris Biemann. 2006. Chinese Whispers - An Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: The First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York, NY, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danushka Bollegala, Vincent Atanasov, Takanori Maehara, and Ken-Ichi Kawarabayashi. 2018. ClassiNet - Predicting Missing Features for Short-Text Classification. *ACM Transactions on Knowledge Discovery from Data*, 12(5):1–29.

Moses S. Charikar. 2002. Similarity Estimation Techniques From Rounding Algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388, Montréal, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Pascal Denis and Jason Baldridge. 2007. Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 236–243, Rochester, NY, USA.

Tobias Falke and Iryna Gurevych. 2017. Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2951–2961, Copenhagen, Denmark.

Tobias Falke, Christian M. Meyer, and Iryna Gurevych. 2017. Concept-Map-Based Multi-Document Summarization using Concept Coreference Resolution and Global Importance Optimization. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pages 801–811, Taipei, Taiwan.

Abram Handler and Brendan O'Connor. 2018. Relational Summarization for Corpus Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1760–1769, New Orleans, LA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, NV, USA.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244.

Claire Mouton, Guillaume Pitel, Gaël de Chalendar, and Anne Vilnat. 2009. Unsupervised Word Sense Induction from Multiple Semantic Spaces with Locality Sensitive Hashing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 287–291, Borovets, Bulgaria.

Joseph D. Novak and D. Bob Gowin. 1984. *Learning How to Learn*. Cambridge University Press, Cambridge, MA, USA.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, LA, USA.

Kanagasabai Rajaraman and Ah-Hwee Tan. 2002. Knowledge Discovery from Texts: A Concept Frame Graph Approach. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 669–671, McLean, VA, USA.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Function for High Speed Noun Clustering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 622–629, Ann Arbor, MI, USA.

Alejandro Valerio and David B. Leake. 2006. Jump-Starting Concept Map Construction with Knowledge Extracted from Documents. In *Concept Maps: Theory, Methodology, Technology. Proceedings of the 2nd International Conference on Concept Mapping*, pages 296–303, San José, Costa Rica.

Benjamin van Durme. 2012. Streaming Analysis of Discourse Participants. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 48–58, Jeju Island, Korea.

Jorge J. Villalon. 2012. *Automated Generation of Concept Maps to Support Writing*. Ph.D. Thesis, University of Sydney.

Krunoslav Zubrinic, Ines Obradovic, and Tomo Sjekavica. 2015. Implementation of method for generating concept map from unstructured text in the Croatian language. In *23rd International Conference on Software, Telecommunications and Computer Networks*, pages 220–223, Split, Croatia.

Syntax-aware Neural Semantic Role Labeling with Supertags

Jungo Kasai♣* **Dan Friedman**♠* **Robert Frank**♢
Dragomir Radev♢ **Owen Rambow**♡
♣University of Washington ♠Google
♢Yale University ♡Elemental Cognition, LLP
jkasai@cs.washington.edu danfriedman@google.com
{robert.frank,dragomir.radev}@yale.edu
owenr@elementalcognition.com

Abstract

We introduce a new syntax-aware model for dependency-based semantic role labeling that outperforms syntax-agnostic models for English and Spanish. We use a BiLSTM to tag the text with supertags extracted from dependency parses, and we feed these supertags, along with words and parts of speech, into a deep highway BiLSTM for semantic role labeling. Our model combines the strengths of earlier models that performed SRL on the basis of a full dependency parse with more recent models that use no syntactic information at all. Our local and non-ensemble model achieves state-of-the-art performance on the CoNLL 09 English and Spanish datasets. SRL models benefit from syntactic information, and we show that supertagging is a simple, powerful, and robust way to incorporate syntax into a neural SRL system.

1 Introduction

Semantic role labeling (SRL) is the task of identifying the semantic relationships between each predicate in a sentence and its arguments (Gildea and Jurafsky, 2002). While early research assumed that SRL models required syntactic information to perform well (Punyakanok et al., 2008), recent work has demonstrated that neural networks can achieve competitive and even state-of-the-art performance without any syntactic information at all (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017). These systems have the benefits of being simpler to implement and performing more robustly on foreign languages and out-of-domain data, cases where syntactic parsing is more difficult (Marcheggiani et al., 2017).

In this paper, we show that using supertags is an effective middle ground between using full syntactic parses and using no syntactic information at all. A supertag is a linguistically rich description assigned to a lexical item. Supertags impose complex constraints on their local context, so supertagging can be thought of as "almost parsing" (Bangalore and Joshi, 1999). Supertagging has been shown to facilitate Tree-Adjoining Grammar (TAG) parsing (Bangalore et al., 2009; Friedman et al., 2017; Kasai et al., 2017, 2018) and Combinatory Categorial Grammar (CCG) parsing (Clark and Curran, 2007; Kummerfeld et al., 2010; Lewis et al., 2016; Xu, 2016).

We propose that supertags can serve as a rich source of syntactic information for downstream tasks without the need for full syntactic parsing. Following Ouchi et al. (2014), who used supertags to improve dependency parsing, we extract various forms of supertags from the dependency-annotated CoNNL 09 corpus. This contrasts with prior SRL work that uses TAG or CCG supertags (Chen and Rambow, 2003; Lewis et al., 2015). We train a bidirectional LSTM (BiLSTM) to predict supertags and feed the predicted supertag embedding, along with word and predicted part-of-speech embeddings, to another BiLSTM for semantic role labeling. Predicted supertags are represented by real-valued vectors, contrasting with approaches based on syntactic paths (Roth and Lapata, 2016; He et al., 2018) and syntactic edges (Marcheggiani and Titov, 2017; Strubell et al., 2018). This way of incorporating information alleviates the issue of error propagation from parsing.

Supertagging has many advantages as part of a natural language processing pipeline. First, as a straightforward sequence-labeling task, the supertagging architecture is much simpler than comparable systems for structured parsing. Second, it is simple to extract different forms of supertags from a dependency corpus to test different hypotheses about which kinds of syntactic information are most useful for downstream tasks. Our re-

---

[*] Work partially done at Yale University.

Proceedings of NAACL-HLT 2019, pages 701–709
Minneapolis, Minnesota, June 2 - June 7, 2019.

Token	Model 1	Model TAG
No	DEP/R	DEP/R
,	P/R	P/R
it	SBJ/R	-
was	ROOT+L_R	ROOT+SBJ/L_PRD/R
n't	ADV/L	ADV/L
black	NAME/R	NAME/R
Monday	PRD/L+L	-

Table 1: Supertags for the sentence "No, it wasn't black Monday."

Position	Feature	0	1	2	TAG
Obligatory Parent	Direction	•	•	•	
	Relation	•	•	•	
Optional Parent	Direction	•	•	•	•
	Relation	•	•	•	•
Obligatory Dep.	Direction		•	•	•
	Relation			•	•
Optional Dep.	Direction		•	•	

Table 2: Supertag models for SRL. Models 1 and 2 are from Ouchi et al. (2014) and Model 0 is from Nguyen and Nguyen (2016).

sults show that supertags, by encoding just enough information, can improve SRL performance even compared to systems that incorporate complete dependency parses.

2 Our Models

2.1 Supertag Design

We experiment with four supertag models, two from Ouchi et al. (2014), one from Nguyen and Nguyen (2016), and one of our own design inspired by Tree Adjoining Grammar supertags (Bangalore and Joshi, 1999). Each model encodes a different set of attributes about the syntactic relationship between a word, its parent, and its dependents. Table 2 summarizes what information is expressed in each supertag model.

Model 0. A Model 0 supertag for a word w encodes the dependency relation and the relative position (direction) between w and its head, i.e. left (L), right (R), or no direction (ROOT) (Nguyen and Nguyen, 2016).

Model 1. A Model 1 supertag for w adds to the "parent information" from Model 0 the information of whether w possesses dependents to its left (L) or right (R) (Ouchi et al., 2014).

Model 2. A Model 2 supertag for w extends Model 1 by encoding the dependency relation between w and its obligatory dependents.[1] When w lacks such obligatory children, we encode whether it possesses non-obligatory dependents to the left (L) or right (R) as in Model 1.

Model TAG. We propose Model TAG supertags that represent syntactic information analogously to TAG supertags (elementary trees) (Bangalore and Joshi, 1999). A Model TAG supertag encodes the dependency relation and the direction of the head of a word similarly to Model 0 if the dependency relation is non-obligatory (corresponding to *adjunction* nodes), and the information about obligatory dependents of verbs if any similarly to Model 2 (corresponding to *substitution* nodes).

2.2 Supertagger Model

Motivated by recent state-of-the-art supertaggers (TAG: Kasai et al. (2017, 2018); CCG: Lewis et al. (2016); Xu (2016)), we employ a bi-directional LSTM (BiLSTM) architecture for our supertagging. The input for each word is the conncatenation of a dense vector representation of the word, a vector embedding of a predicted PTB-style POS tag (only for English),[2] and a vector output by character-level Convolutional Neural Networks (CNNs) for morphological information.

For POS tagging before English supertagging, we use the same hyperparameters as in Ma and Hovy (2016). For supertagging, we follow the hyperparameters chosen in Kasai et al. (2018) regardless of the supertag model that is employed. We initialize the word embeddings by the pre-trained 100 dimensional GloVe (Pennington et al., 2014) and the 300 dimensional FastText (Bojanowski et al., 2017) vectors for English and Spanish respectively.

2.3 Semantic Role Labeling

Our SRL model is most similar to the syntax-agnostic SRL model proposed by Marcheggiani et al. (2017). Our model differs in two ways: 1) we add randomly initialized 50 dimensional supertag embeddings to the input layer (Fig. 1), and 2) we use a modified LSTM with highway layers and regularization (0.5 dropout) as in He et al. (2017).

We use the same hyperparameters as in Marcheggiani et al. (2017) with randomly initialized 50 dimensional embeddings for supertags.[3]

---

[1] Following Ouchi et al. (2014), we define obligatory dependents as those with relations 'SBJ,' 'OBJ,' 'PRD,' and 'VC.' For Spanish, we define obligatory syntactic arguments as 'dc,' 'suj,' 'cd,' and 'cpred.'

[2] For the English data, predicted PTB-style POS tags generally contribute to increases, approximately 0.2-0.4% in the dev set, whereas for Spanish adding predicted (coarse-grained) POS tags hurt the performance.

[3] We provide lists of hyperparameters in Appedix A.1.

	English				Spanish		
Supertag	# Stags	Dev	ID	OOD	# Stags	Dev	ID
Model 0	99	92.93	94.17	88.71	88	92.97	92.67
Model 1	298	91.07	92.50	86.51	220	90.63	90.37
Model 2	692	90.60	92.05	85.40	503	90.08	89.84
Model TAG	430	92.60	94.17	87.46	317	92.33	92.18

Table 3: Supertagging accuracies for English and Spanish. ID and OOD indicate the in-domain and out-of-domain evaluation data respectively. The # Stags columns show the number of supertags in the corresponding training set.

For pre-trained word embeddings, we use the same word embeddings as the ones in Marcheggiani et al. (2017) for English and the 300-dimensional FastText vectors (Bojanowski et al., 2017) for Spanish. We use the predicates predicted by the mate-tools (Björkelund et al., 2009) (English) and Zhao et al. (2009) (Spanish) system in our models, again following Marcheggiani et al. (2017) to facilitate comparison. Our code is available online for easy replication of our results.[4]

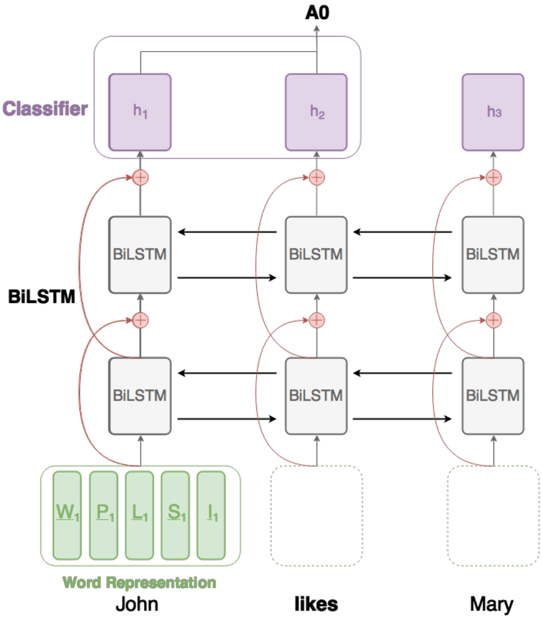


Figure 1: SRL architecture with a highway BiLSTM. W_1, P_1, L_1, S_1, I_1 indicate the word, POS, lemma, supertag, and predicate indicator embeddings for the first token, *John*. Here we only show two layers.

3 Results and Discussion

Table 3 provides our supertagging results for English and Spanish across the different types of supertag described above. Here we clearly see the general pattern that the more granular supertagging becomes, the less reliable it is, and finding the balance between granularity and predictability is critical. We present our SRL results in Tables 4-7 along with the results from a baseline BiLSTM model, which is our implementation of the syntax-agnostic model in Marcheggiani et al. (2017). We also present results for a BiLSTM model with dropout and highway connections but without supertags (BDH model), to distinguish the effects of supertags from the effects of better LSTM regularization. In every experiment we train the model five times, and present the mean score. Table 4 shows that Model 1 yields the best performance in the English dev set, and thus we only use Model 1 supertags for test evaluation. We primarily show results only with word type embeddings to conduct fair comparisons with prior work, but we also provide results with deep contextual word representations, ELMo (Peters et al., 2018), and compare our results with recent work that utilizes ELMo (He et al., 2018). [5]

English in-domain. Table 5 summarizes the results on the English in-domain test set. First, we were able to approximately replicate the results from Marcheggiani et al. (2017). Adding dropout and highway connections to our BiLSTM model improves performance by 0.5 points, to 88.1, and adding supertags improves results even further to 88.6. Our supertag model performs even better than the non-ensemble model in Marcheggiani and Titov (2017), in which the model is given the complete dependency parse of the sentence. This result suggests that supertags can be even more effective for SRL than a more complete representation of syntax. Furthermore, our supertag-based method with contextual representations achieves 90.2, a new state-of-the-art. Interestingly, the gain from supertagging decreases to 0.2 points (90.2 vs. 90.0) in the presence of contextual representations, suggesting that contextual representations encode some of the same syntactic information that supertags provide.

English out-of-domain. One of the advantages of using a syntax-agnostic SRL model is that such a model can perform relatively well on out-of-domain data, where the increased difficulty of syn-

---

[4] `https://github.com/jungokasai/stagging_srl`.

[5] We used the pretrained ELMo available at `https://tfhub.dev/google/elmo/2`.

Architecture	P	R	F_1
BiLSTM	87.27	85.16	86.20
BiLSTM + DOut	86.49	86.11	86.30
BiLSTM + DOut + HWay	86.97	86.43	86.70
BDH + Model 0	87.47	86.46	86.96
BDH + Model 1	87.69	86.72	**87.20**
BDH + Model 2	87.54	86.09	86.81
BDH + Model TAG	87.78	86.07	86.92

Table 4: Results on the CoNLL 2009 dev set for English. BDH stands for BiLSTM + Dropout + Highway.

tactic parsing can cause errors in a syntax-based system (Marcheggiani et al., 2017). Unfortunately we were not able to replicate the out-of-domain results of Marcheggiani et al. (2017): our implementation of the BiLSTM achieves a score of 76.4, compared to their reported score of 77.7. However, we note that incorporating supertags into our own model improves performance, with our best model achieving a score of 77.6. Our supertag-based model also substantially outperforms the full dependency-based models (Roth and Lapata, 2016; Marcheggiani and Titov, 2017). This suggests that syntax with a certain degree of granularity is useful even across domains. Our supertag-based method alleviates the issue of error propagation from syntactic parsing. Finally, our model with contextual representations yields 80.8, an improvement of 1.5 F1 points over the previous state-of-the-art (He et al., 2018), which also uses ELMo.

Spanish. Table 7 shows the results on the Spanish test data. Our BiLSTM implementation yields lower performance than Marcheggiani et al. (2017): our model achieves a score of 79.1, compared to their reported score of 80.3. However, our BDH model yields a score of 80.8, already achieving state-of-the-art performance. Adding supertags to BDH improves the score further to 81.0. This suggests that while the gains are relatively small, the supertag-based approach still helps Spanish SRL. Supertags slightly improve performance when contextual representations are used (83.0 vs. 82.9). See appendices for details.

Following the analysis in Roth and Lapata (2016), we show plots of the BiLSTM, BDH (BiLSTM + Dropout + Highway), and Model 1 role labeling performance for sentences with varying number of words (in-domain: Fig. 2; out-of-domain: Fig. 3). Note first that BDH outperforms the baseline BiLSTM model in a relatively uniform manner across varying sentence lengths. The benefits of Model 1 supertags, in contrast, come more from longer sentences, especially in the out-

Non-ensemble System	P	R	F_1
FitzGerald et al. (2015)	–	–	87.3
Roth and Lapata (2016)	90.0	85.5	87.7
Marcheggiani et al. (2017)	88.7	86.8	87.7
Marcheggiani and Titov (2017)	89.1	86.8	88.0
BiLSTM	88.5	86.7	87.6
BDH	88.3	87.8	88.1
BDH + Model 1	89.0	88.2	**88.6**
+ Contextual Representations			
He et al. (2018) (ELMo)	89.7	89.3	89.5
BDH + ELMo	90.3	89.7	90.0
BDH + Model 1 + ELMo	90.3	90.0	**90.2**
Ensemble System			
FitzGerald et al. (2015)	–	–	87.7
Roth and Lapata (2016)	90.3	85.7	87.9
Marcheggiani and Titov (2017)	90.5	87.7	89.1

Table 5: Results on the CoNLL 2009 in-domain test set for English. All standard deviations in $F_1 < 0.12$.

Non-ensemble System	P	R	F_1
FitzGerald et al. (2015)	–	–	75.2
Roth and Lapata (2016)	76.9	73.8	75.3
Marcheggiani et al. (2017)	79.4	76.2	**77.7**
Marcheggiani and Titov (2017)	78.5	75.9	77.2
BiLSTM	77.2	75.6	76.4
BDH	77.8	76.6	77.2
BDH + Model 1	78.0	77.2	77.6
+ Contextual Representations			
He et al. (2018) (ELMo)	81.9	76.9	79.3
BDH + ELMo	81.1	80.4	**80.8**
BDH + Model 1 + ELMo	81.0	80.5	**80.8**
Ensemble System			
FitzGerald et al. (2015)	–	–	75.5
Roth and Lapata (2016)	79.7	73.6	76.5
Marcheggiani and Titov (2017)	80.8	77.1	78.9

Table 6: Results on the CoNLL 2009 out-of-domain test set for English. The standard deviation in F_1 ranges between 0.2 and 0.35.

System	P	R	F_1
Zhao et al. (2009)	83.1	78.0	80.5
Roth and Lapata (2016)	83.2	77.4	80.2
Marcheggiani et al. (2017)	81.4	79.3	80.3
BiLSTM	79.8	78.4	79.1
BDH	82.0	79.7	80.8
BDH + Model 1	81.9	80.2	**81.0**
BDH + ELMo	83.1	82.8	82.9
BDH + Model 1 + ELMo	83.1	83.0	**83.0**

Table 7: Results on the CoNLL 2009 test set for Spanish. All standard deviations in $F_1 < 0.1$.

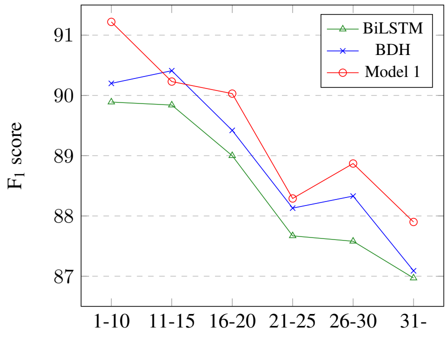


Figure 2: In-domain test results by sentence length.

Model	V / A0 P	V / A0 R	V / A0 F	V / A1 P	V / A1 R	V / A1 F	V / A2 P	V / A2 R	V / A2 F	V / AM P	V / AM R	V / AM F
Mate-tools	91.2	87.4	89.3	91.0	90.8	90.9	82.8	76.9	79.7	79.3	74.4	76.8
Path-LSTM	90.8	89.2	90.0	91.0	91.9	91.4	84.3	76.9	80.4	82.2	72.4	77.0
BiLSTM	91.1	89.7	90.4	92.1	90.9	91.5	84.0	75.0	79.2	77.7	76.9	77.3
BDH	90.9	90.8	90.9	91.5	92.4	92.0	80.3	76.1	78.1	79.6	79.1	79.3
Model 0	92.3	92.2	**92.3**	93.4	92.7	**93.0**	81.9	77.8	79.8	79.1	79.5	79.3
Model 1	92.5	91.6	92.0	93.0	92.8	92.9	80.9	80.3	**80.6**	80.1	78.6	**79.4**
Model 2	91.9	90.1	91.0	92.5	92.4	92.4	79.2	77.8	78.5	79.9	78.2	79.1
TAG	91.7	89.9	90.8	92.5	93.3	92.9	82.1	77.3	79.6	80.4	78.3	79.3

Model	N / A0 P	N / A0 R	N / A0 F	N / A1 P	N / A1 R	N / A1 F	N / A2 P	N / A2 R	N / A2 F	N / AM P	N / AM R	N / AM F
Mate-Tools	86.1	74.9	80.2	84.9	82.2	83.5	81.4	74.7	77.9	78.6	72.0	75.2
Path-LSTM	86.9	78.2	82.3	87.5	84.4	**85.9**	82.4	76.8	79.5	79.5	69.2	74.0
BiLSTM	85.1	79.5	82.2	85.8	83.4	84.6	81.0	76.4	78.7	72.8	71.4	72.1
BDH	83.9	80.1	82.0	84.8	86.1	85.5	80.6	77.0	78.8	71.3	77.9	74.5
Model 0	87.2	77.6	82.1	86.2	85.4	85.8	79.9	79.2	79.5	69.4	79.4	74.0
Model 1	84.1	80.7	82.4	85.2	86.0	85.6	79.6	79.6	**79.6**	75.2	76.3	**75.8**
Model 2	86.0	79.5	**82.6**	85.4	85.5	85.5	80.5	77.3	78.9	73.3	76.1	74.6
TAG	83.9	79.8	81.8	84.9	86.3	85.6	81.5	75.8	78.6	72.3	72.9	72.6

Table 8: English in-domain test results by predicate category and role label. The mate-tools (Björkelund et al., 2009) and Path-LSTM results are taken from Roth and Lapata (2016).

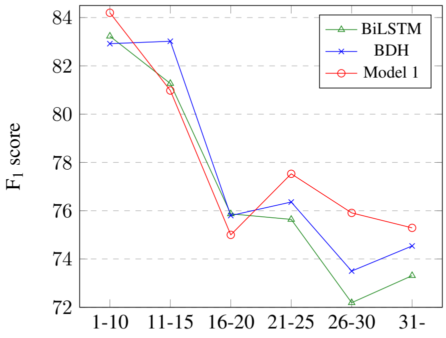


Figure 3: Out-of-domain results by sentence length.

of-domain test set. This implies that the supertag model is robust to the sentence length, probably because supertags encode relations between words that are linearly distant in the sentence, information that a simple BiLSTM is unlikely to recover.

Table 8 reports SRL results broken down by predicate category (V: Verb, Propbank; N: Noun, Nombank) and semantic role. We can observe that the various supertag models differ in their performance for different predicate-role pairs, suggesting that different kinds of linguistic information are relevant for identifying the different roles. Overall, Model 1 supertags achieve the most consistent improvements over BiLSTM and BiLSTM + Dropout + Highway (BDH) in V / A0, V / A1, V / A2, V / AM, N / A2, and N / AM. Moreover, Model 1 even improves on Path-LSTM (Roth and Lapata, 2016) by large margins in V / A0, V / A1, V / AM, and N / AM, even though the Path-LSTM model has the benefit of using the complete dependency path between each word and its head. This shows that supertags can be even more effective for SRL than more granular syntactic information–even quite simple supertags, like Model 0, which encode only the dependency arc between a word and its head.

4 Conclusion and Future Work

We presented state-of-the-art SRL systems on the CoNLL 2009 English and Spanish data that make crucial use of dependency-based supertags. We showed that supertagging serves as an effective middle ground between syntax-agnostic approaches and full parse-based approaches for dependency-based semantic role labeling. Supertags give useful syntactic information for SRL and allow us to build an SRL system that does not depend on a complex architecture. We have also seen that the choice of the linguistic content of a supertag makes a significant difference in its utility for SRL. In this work, all models are developed independently for English and Spanish. However, sharing some part of SRL models could improve performance (Mulcaire et al., 2018, 2019). In future work, we will explore crosslingual transfer for supertagging and semantic role labeling.

Acknowledgments

The authors thank Diego Marcheggiani for assistance in implementing SRL models and Diego Marcheggiani and the anonymous reviewers for their helpful feedback. This work was funded in part by the Funai Overseas Scholarship to JK.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A probabilistic dependency parser based on tree insertion grammars (application note). In *Proc. of NAACL-HLT*, pages 185–188, Boulder, Colorado. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proc. of CoNLL*, pages 43–48, Boulder, Colorado. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

John Chen and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proc. of EMNLP*, pages 41–48. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33:493–552.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Dan Friedman, Jungo Kasai, R. Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for TAG parsing. In *Proc. of TAG+*, pages 122–131, Umeå, Sweden. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. arXiv:1803.07640.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*, pages 2061–2071, Melbourne, Australia. Association for Computational Linguistics.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proc. of EMNLP*, pages 1713–1723, Copenhagen, Denmark. Association for Computational Linguistics.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. In *Proc. of NAACL-HLT*, pages 1181–1194, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A Method for Stochastic Optimization. In *Proc. of ICLR*, San Diego, California.

Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. 2010. Faster parsing by supertagger adaptation. In *Proc. of ACL*, pages 345–355, Uppsala, Sweden. Association for Computational Linguistics.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*, pages 1444–1454, Lisbon, Portugal. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proc. of NAACL-HLT*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proc. of CoNLL*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proc. of EMNLP*, pages 1507–1516, Copenhagen, Denmark. Association for Computational Linguistics.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proc. of NAACL-HLT*, page to appear.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proc. of ACL*, pages 667–672, Melbourne, Australia. Association for Computational Linguistics.

Kiet Van Nguyen and Ngan Luu-Thuy Nguyen. 2016. Vietnamese transition-based dependency parsing with supertag features. In *Proc. of KSE*, pages 175–180, Hanoi, Vietnam. Springer International Publishing.

Hiroki Ouchi, Kevin Duh, and Yuji Matsumoto. 2014. Improving dependency parsers with supertags. In *Proc. of EACL*, pages 154–158, Gothenburg, Sweden. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of ACL*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Wenduan Xu. 2016. LSTM shift-reduce CCG parsing. In *Proc. of EMNLP*, pages 1754–1764, Austin, Texas. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proc. of CoNLL*, pages 61–66, Boulder, Colorado. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of ACL-IJCNLP*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

A Appendices

A.1 Hyperparameters

All of our models are implemented in TensorFlow (Abadi et al., 2015).

Supertagging We follow the hyperparameters chosen in Kasai et al. (2018). Specifically, we list the hyperparameters in Table 9 for completeness and easy replication.

d_w (English word embeddings)	100
d_w (Spanish word embeddings)	300
d_{pos} (POS embeddings)	100
Char-CNN window size	3
Char-CNN # filters	30
Char-CNN character embedding size	30
d_h (LSTM hidden states)	512
k (BiLSTM depth)	4
LSTM dropout rate	0.5
Recurrent dropout rate	0.5
Batch Size	100
Adam (Kingma and Ba, 2015) lrate	0.01
Adam β_1	0.9
Adam β_2	0.999

Table 9: Supertagging Hyperparameters.

SRL We follow the hyperparameters of Marcheggiani et al. (2017) and add highway connections (He et al., 2017) and LSTM dropout. Concretely, we use the hyperparameters shown in Table 10.

Contextual Representations For English, we use the pretrained ELMo model available at `https://tfhub.dev/google/elmo/2`. For Spanish, we use a multilingual fork (Mulcaire

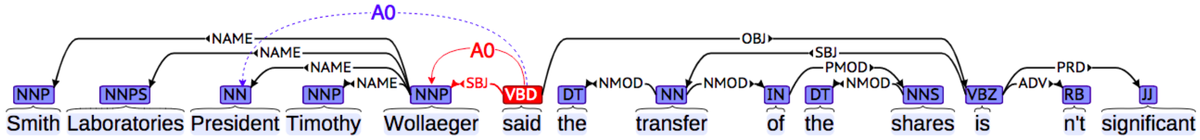


Figure 4: V / A0 case where BDH assigns A0 to *President* (blue arc) while Model 1 correctly assigns A0 to *Wollaeger* (red arc). The predicted Model 1 supertags for *President* and *Wollaeger* are NAME/R and SBJ/R+L respectively.

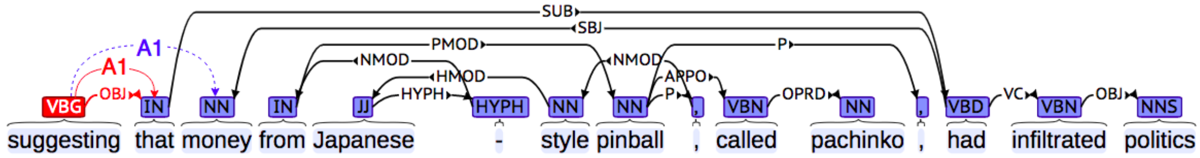


Figure 5: V / A1 where BDH assigns A1 to *money* (blue arc) while Model 1 correctly assigns A1 to *that* (red arc). The predicted Model 1 supertags for *that* and *money* are OBJ/L+R and SBJ/R+R respectively.

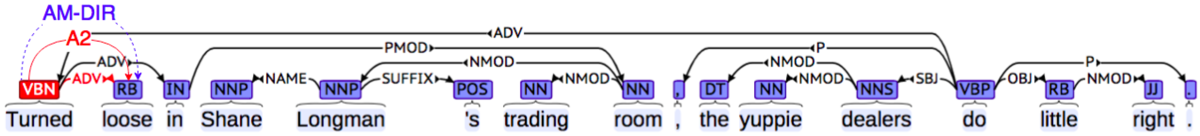


Figure 6: V / A2 case where BDH assigns AM-DIR to *loose* (blue arc) while Model 1 correctly assign A2 (red arc). The predicted supertag for *loose* is PRD/L (predicative complement). Notice that the “PRT” (particle) or “DIR” (adverbial of direction) feature is not predicted that could have misled the labeling. Interestingly, the gold parse and gold POS tag for *loose* treat it as an adverbial modifier to *turned*.

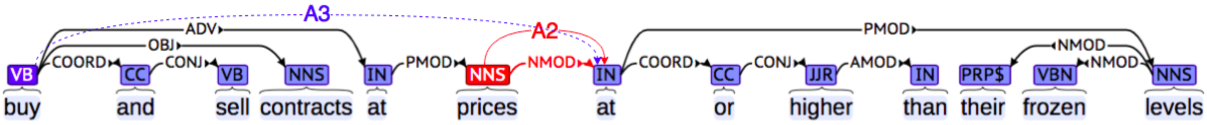


Figure 7: N / A2 case where BDH assigns A3 for the predicate *buy* to *at* (blue arc) while Model 1 correctly assigns A2 for the predicate *prices* (red arc). The predicted Model 1 supertag for *at* was NMOD/L+R, correctly resolving the PP attachment ambiguity.

d_w (English word embeddings)	100
d_w (Spanish word embeddings)	300
d_{pos} (POS embeddings)	16
d_l (lemma embeddings)	100
d_s (supertag embeddings)	50
d_h (LSTM hidden states)	512
d_r (role representation)	128
d'_l (output lemma representation)	128
k (BiLSTM depth)	4
α (word dropout)	.25
LSTM dropout rate	0.5
Batch Size	100
Adam lrate	0.01
Adam β_1	0.9
Adam β_2	0.999

Table 10: SRL Hyperparameters

Character CNNs	
Char embedding size	16
(# Window Size, # Filters)	(1, 32), (2, 32), (3, 68), (4, 128), (5, 256), 6, 512), (7, 1024)
Activation	Relu
Word-level LSTM	
LSTM size	2048
# LSTM layers	2
LSTM projection size	256
Use skip connections	Yes
Inter-layer dropout rate	0.1
Training	
Batch size	128
Unroll steps (Window Size)	20
# Negative samples	64
# Epochs	10
Adagrad (Duchi et al., 2011) lrate	0.2
Adagrad initial accumulator value	1.0

Table 11: Spanish Language Model Hyperparameters.

et al., 2019)[6] of the AllenNLP library (Gardner et al., 2018), and train a language model on the pre-segmented Spanish data provided by Ginter et al. (2017).[7] The original AllenNLP library uses a byte representation. On the other hand, the multilingual fork assigns a unique character id to each unicode character, thereby avoiding the need to recognize mutibyte representations. We follow the hyperparameters chosen in Mulcaire et al. (2019) (Table 11), and randomly sample 50 million tokens from the Spanish data for training.

A.2 Supplementary Analysis

We show examples from the dev set in Figures 4-7 where a model without supertags mislabels (dashed blue arcs) and Model 1 (red arcs) correctly labels. In all those cases, it is clear that the predicted supertags are playing a crucial role in guiding role labeling.

---

[6] https://github.com/pmulcaire/rosita/

[7] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989

Left-to-Right Dependency Parsing with Pointer Networks

Daniel Fernández-González
Universidade da Coruña
FASTPARSE Lab, LyS Group
Departamento de Computación
Elviña, 15071 A Coruña, Spain
`d.fgonzalez@udc.es`

Carlos Gómez-Rodríguez
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Group
Departamento de Computación
Elviña, 15071 A Coruña, Spain
`carlos.gomez@udc.es`

Abstract

We propose a novel transition-based algorithm that straightforwardly parses sentences from left to right by building n attachments, with n being the length of the input sentence. Similarly to the recent stack-pointer parser by Ma et al. (2018), we use the pointer network framework that, given a word, can directly point to a position from the sentence. However, our left-to-right approach is simpler than the original top-down stack-pointer parser (not requiring a stack) and reduces transition sequence length in half, from $2n-1$ actions to n. This results in a quadratic non-projective parser that runs twice as fast as the original while achieving the best accuracy to date on the English PTB dataset (96.04% UAS, 94.43% LAS) among fully-supervised single-model dependency parsers, and improves over the former top-down transition system in the majority of languages tested.

1 Introduction

Dependency parsing, the task of automatically obtaining the grammatical structure of a sentence expressed as a dependency tree, has been widely studied by natural language processing (NLP) researchers in the last decades. Most of the models providing competitive accuracies fall into two broad families of approaches: *graph-based* (McDonald et al., 2005a,b) and *transition-based* (Yamada and Matsumoto, 2003; Nivre, 2003) dependency parsers.

Given an input sentence, a graph-based parser scores trees by decomposing them into factors, and performs a search for the highest-scoring tree.

In the past two years, this kind of dependency parsers have been ahead in terms of accuracy thanks to the graph-based neural architecture developed by Dozat and Manning (2016), which not only achieved state-of-the-art accuracies on the Stanford Dependencies conversion of the English Penn Treebank (hereinafter, PTB-SD), but also obtained the best results in the majority of languages in the CoNLL 2017 Shared Task (Dozat et al., 2017). This tendency recently changed, since a transition-based parser developed by Ma et al. (2018) managed to outperform the best graph-based model in the majority of datasets tested.

Transition-based parsers incrementally build a dependency graph for an input sentence by applying a sequence of transitions. This results in more efficient parsers with linear time complexity for parsing projective sentences, or quadratic for handling non-projective structures, when implemented with greedy or beam search. However, their main weakness is the lack of access to global context information when transitions are greedily chosen. This favours error propagation, mainly affecting long dependencies that require a larger number of transitions to be built (McDonald and Nivre, 2011).

Many attempts have been made to alleviate the impact of error propagation in transition-based dependency parsing, but the latest and most successful approach was developed by Ma et al. (2018). In particular, they make use of *pointer networks* (Vinyals et al., 2015) to implement a new neural network architecture called *stack-pointer network*. The proposed framework provides a global view of the input sentence by capturing information from the whole sentence and all the arcs previously built, crucial for reducing the effect of error propagation; and, thanks to an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015), is able to return a position in that sentence that corresponds to a word related to the word currently on top of the stack. They take advantage of this and propose a novel transition system that follows a top-down depth-first strategy to perform the syntactic analysis. Concretely, it considers the word

Proceedings of NAACL-HLT 2019, pages 710–716
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

pointed by the neural network as the child of the word on top of the stack, and builds the corresponding dependency relation between them. This results in a transition-based algorithm that can process unrestricted non-projective sentences in $O(n^2)$ time complexity and requires $2n$-1 actions to successfully parse a sentence with n words.

We also take advantage of pointer network capabilities and use the neural network architecture introduced by Ma et al. (2018) to design a non-projective left-to-right transition-based algorithm, where the position value pointed by the network has the opposite meaning: it denotes the index that corresponds to the head node of the current focus word. This results in a straightforward transition system that can parse a sentence in just n actions, without the need of any additional data structure and by just attaching each word from the sentence to another word (including the root node). Apart from increasing the parsing speed twofold (while keeping the same quadratic time complexity), it achieves the best accuracy to date among fully-supervised single-model dependency parsers on the PTB-SD, and obtains competitive accuracies on twelve different languages in comparison to the original top-down version.

2 Preliminaries

Ma et al. (2018) propose a novel neural network architecture whose main backbone is a pointer network (Vinyals et al., 2015). This kind of neural networks are able to learn the conditional probability of a sequence of discrete numbers that correspond to positions in an input sequence (in this case, indexes of words in a sentence) and, by means of attention (Bahdanau et al., 2014; Luong et al., 2015), implement a pointer that selects a position from the input at decoding time.

Their approach initially reads the whole sentence, composed of the n words $w_1, \ldots, w_n$, and encodes each w_i one by one into an *encoder hidden state* e_i. As encoder, they employ a combination of CNNs and bi-directional LSTMs (Chiu and Nichols, 2016; Ma and Hovy, 2016). For each word, CNNs are used to obtain its character-level representation that is concatenated to the word and PoS embeddings to finally be fed into BiLSTMs that encode word context information.

As decoder they present a top-down transition system, where parsing configurations use the classic data structures (Nivre, 2008): a *buffer* (that contains unattached words) and a *stack* (that holds partially processed words).

The available parser actions are two transitions that we call Shift-Attach-p and Reduce. Given a configuration with word w_i on top of the stack, as the pointer network just returns a position p from a given sentence, they proceed as follows to determine which transition should be applied:

- If $p \neq i$, then the pointed word w_p is considered as a child of w_i; so the parser chooses a Shift-Attach-p transition to move w_p from the buffer to the stack and build an arc $w_i \rightarrow w_p$.
- On the other hand, if $p = i$, then w_i is considered to have found all its children, and a Reduce transition is applied to pop the stack.

The parsing process starts with a dummy root \$ on the stack and, by applying $2n$-1 transitions, a dependency tree is built for the input in a top-down depth-first fashion, where multiple children of a same word are forced during training to be created in an inside-out manner. More in detail, for each parsing configuration c_t, the decoder (implemented as a uni-directional LSTM) receives the encoder hidden state e_i of the word w_i on top of the stack to generate a *decoder hidden state* d_t. After that, d_t, together with the sequence s_i of encoder hidden states from words still in the buffer plus e_i, are used to compute the attention vector a^t as follows:

$$v_i^t = score(d_t, s_i) \tag{1}$$

$$a^t = softmax(v^t) \tag{2}$$

As attention scoring function ($score()$), they adopt the biaffine attention mechanism described in (Luong et al., 2015; Dozat and Manning, 2016). Finally, the attention vector a^t will be used to return the highest-scoring position p and choose the next transition. The parsing process ends when only the root remains on the stack.

As extra high-order features, Ma et al. (2018) add grandparent and sibling information, whose encoder hidden states are added to that of the word on top of the stack to generate the corresponding decoder hidden state d_t. They prove that these additions improve final accuracy, especially when children are attached in an inside-out fashion.

According to the authors, the original stack-pointer network is trained to maximize the likelihood of choosing the correct word for each possible top-down path from the root to a leaf. More in detail, a dependency tree can be represented as a sequence of top-down paths $p_1, \ldots, p_k$, where each path p_i corresponds to a sequence of words $\$, w_{i,1}, w_{i,2}, \ldots, w_{i,l_i}$ from the root to a leaf. Thus, the conditional probability $P_\theta(y|x)$ of the dependency tree y for an input sentence x can be factorized according to this top-down structure as:

$$\begin{aligned} P_\theta(y|x) &= \prod_{i=1}^{k} P_\theta(p_i | p_{<i}, x) \\ &= \prod_{i=1}^{k} \prod_{j=1}^{l_i} P_\theta(w_{i,j} | w_{i,<j}, p_{<i}, x) \end{aligned}$$

where θ represents model parameters, $p_{<i}$ stands for previous paths already explored, $w_{i,j}$ denotes the jth word in path p_i and $w_{i,<j}$ represents all the previous words on p_i.

For more thorough details of the stack-pointer network architecture and the top-down transition system, please read the original work by Ma et al. (2018).

3 Our approach

We take advantage of the neural network architecture designed by Ma et al. (2018) and introduce a simpler left-to-right transition system that requires neither a stack nor a buffer to process the input sentence and where, instead of selecting a child of the word on top of the stack, the network points to the parent of the current focus word.

In particular, in our proposed approach, the parsing configuration just corresponds to a *focus word pointer* i, that is used to point to the word currently being processed. The decoding process starts with i pointing at the first word of the sentence and, at each parsing configuration, only one action is available: the parameterized Attach-p transition, that links the focus word w_i to the head word w_p in position p of the sentence (producing the dependency arc $w_p \rightarrow w_i$) and moves i one position to the right. Note that, in our algorithm, p can equal 0, attaching, in that case, w_i to the dummy root node. The parsing process ends when the last word from the sentence is attached. This can be easily represented as a loop that traverses the input sentence from left to right, linking each word to another from the same sentence or to the dummy root. Therefore, we just need n steps to process the n words of a given sentence and build a dependency tree.

While our novel transition system intrinsically holds the single-head constraint (since, after attaching the word w_i, i points to the next word w_{i+1} in the sentence), it can produce an output with cycles.[1] Therefore, in order to build a well-formed dependency tree during decoding, attachments that generate cycles in the already-built dependency graph must be forbidden. Please note that the need of a cycle-checking extension does not increase the overall quadratic runtime complexity of the original implementation by Ma et al. (2018) since, as in other transition-based parsers such as (Covington, 2001; Gómez-Rodríguez and Nivre, 2010), cycles can be incrementally identified in amortized constant time by keeping track of connected components using path compression and union by rank. Therefore, the left-to-right algorithm requires n steps to produce a parse. In addition, at each step, the attention vector a_t needs to be computed and cycles must be checked, both in $O(n) + O(n) = O(n)$ runtime. This results in a $O(n^2)$ time complexity for decoding.[2]

On the other hand, while in the top-down decoding only available words in the buffer (plus the word on top of the stack) can be pointed to by the network and they are reduced as arcs are created (basically to keep the single-head constraint); our proposed approach is less rigid: all words from the sentence (including the root node and excluding w_i) can be pointed to, as long as they satisfy the acyclicity constraint. This is necessary because two different words might be attached to the same head node and the latter can be located in the sentence either before or after w_i. Therefore, the sequence s_i, required by the attention score function (Eq.(1)), is composed of the encoder hidden states of all words from the input, excluding e_i, and prepending a special vector representation denoting the root node.

We also add extra features to represent the current focus word. Instead of using grandparent and sibling information (more beneficial for a top-down approach), we just add the encoder hidden

---

[1] In practice, even with the cycle detection mechanism disabled, the presence of cycles in output parses is very uncommon (for instance, just in 1% of sentences in the PTB-SD dev set) since our system seems to adequately model well-formed tree structures.

[2] A practically faster version of the left-to-right parser might be implemented by just ignoring the presence of cycles during decoding, and destroying the cycles generated as a post-processing step that simply removes one of the arcs involved.

states of the previous and next words in the sentence to generate d_t, which seems to be more suitable for a left-to-right decoding.

In dependency parsing, a tree for an input sentence of length n can be represented as a set of n directed and binary links $l_1, \ldots, l_n$. Each link l_i is characterized by the word w_i in position i in the sentence and its head word w_h, resulting in a pair (w_i, w_h). Therefore, to train this novel variant, we factorize the conditional probability $P_\theta(y|x)$ to a set of head-dependent pairs as follows:

$$\begin{aligned} P_\theta(y|x) &= \prod_{i=1}^{n} P_\theta(l_i|l_{<i}, x) \\ &= \prod_{i=1}^{n} P_\theta(w_h|w_i, l_{<i}, x) \end{aligned}$$

Therefore, the left-to-right parser is trained by maximizing the likelihood of choosing the correct head word w_h for the word w_i in position i, given the previous predicted links $l_{<i}$.

Finally, following a widely-used approach (also implemented in (Ma et al., 2018)), dependency labels are predicted by a multiclass classifier, which is trained in parallel with the parser by optimizing the sum of their objectives.

4 Experiments

4.1 Data and Settings

We use the same implementation as Ma et al. (2018) and conduct experiments on the Stanford Dependencies (de Marneffe and Manning, 2008) conversion (using the Stanford parser v3.3.0)[3] of the English Penn Treebank (Marcus et al., 1993), with standard splits and predicted PoS tags. In addition, we compare our approach to the original top-down parser on the same twelve languages from the Universal Dependency Treebanks[4] (UD) that were used by Ma et al. (2018).[5]

Following standard practice, we just exclude punctuation for evaluating on PTB-SD and, for each experiment, we report the average Labelled and Unlabelled Attachment Scores (LAS and UAS) over 3 and 5 repetitions for UD and PTB-SD, respectively.

Parser	UAS	LAS
Chen and Manning (2014)	91.8	89.6
Dyer et al. (2015)	93.1	90.9
Weiss et al. (2015)	93.99	92.05
Ballesteros et al. (2016)	93.56	91.42
Kiperwasser and Goldberg (2016)	93.9	91.9
Alberti et al. (2015)	94.23	92.36
Qi and Manning (2017)	94.3	92.2
Fernández-G and Gómez-R (2018)	94.5	92.4
Andor et al. (2016)	94.61	92.79
Ma et al. (2018)*	95.87	94.19
This work*	**96.04**	**94.43**
Kiperwasser and Goldberg (2016)	93.1	91.0
Wang and Chang (2016)	94.08	91.82
Cheng et al. (2016)	94.10	91.49
Kuncoro et al. (2016)	94.26	92.06
Zhang et al. (2017)	94.30	91.95
Ma and Hovy (2017)	94.88	92.96
Dozat and Manning (2016)	95.74	94.08
Ma et al. (2018)*	95.84	94.21

Table 1: Accuracy comparison of state-of-the-art fully-supervised single-model dependency parsers on PT-SD. The first block contains transition-based algorithms and the second one, graph-based models. Systems marked with *, including the improved variant described in (Ma et al., 2018) of the graph-based parser by (Dozat and Manning, 2016), are implemented under the same framework as our approach and use the same training settings. Like (Ma et al., 2018), we report the average accuracy over 5 repetitions.

Finally, we use the same hyper-parameter values, pre-trained word embeddings and beam size (10 for PTB-SD and 5 for UD) as Ma et al. (2018).

4.2 Results

By outperforming the two current state-of-the-art graph-based (Dozat and Manning, 2016) and transition-based (Ma et al., 2018) models on the PTB-SD, our approach becomes the most accurate fully-supervised dependency parser developed so far, as shown in Table 1.[6]

In addition, in Table 2 we can see how, under the exactly same conditions, the left-to-right algorithm improves over the original top-down variant in nine out of twelve languages in terms of LAS, obtaining competitive results in the remaining three datasets.

Finally, in spite of requiring a cycle-checking procedure, our approach proves to be twice as fast as the top-down alternative in decoding time,

---

[3] `https://nlp.stanford.edu/software/lex-parser.shtml`

[4] `http://universaldependencies.org`

[5] Please note that, since they used a former version of UD datasets, we reran also the top-down algorithm on the latest treebank version (2.2) in order to perform a fair comparison.

[6] It is worth mentioning that all parsers reported in this section make use of pre-trained word embeddings previously learnt from corpora beyond the training dataset. However, it is common practice in the literature that systems that only use standard pre-trained word embeddings are classed as fully-supervised models, even though, strictly, they are not trained exclusively on the official training data.

	Top-down		Left-to-right	
	UAS	LAS	UAS	LAS
bu	**94.42±0.02**	**90.70±0.04**	94.28±0.06	90.66±0.11
ca	93.83±0.02	91.96±0.01	**94.07±0.06**	**92.26±0.05**
cs	93.97±0.02	91.23±0.03	**94.19±0.04**	**91.45±0.05**
de	**87.28±0.07**	**82.99±0.07**	87.06±0.05	82.63±0.01
en	90.86±0.15	88.92±0.19	**90.93±0.11**	**88.99±0.11**
es	93.09±0.05	91.11±0.03	**93.23±0.03**	**91.28±0.02**
fr	**90.97±0.09**	**88.22±0.12**	90.90±0.04	88.14±0.10
it	94.08±0.04	92.24±0.06	**94.28±0.06**	**92.48±0.02**
nl	**93.23±0.09**	90.67±0.07	93.13±0.07	**90.74±0.08**
no	95.02±0.05	93.75±0.05	**95.23±0.06**	**93.99±0.07**
ro	91.44±0.11	85.80±0.14	**91.58±0.08**	**86.00±0.07**
ru	94.43±0.01	93.08±0.03	**94.71±0.07**	**93.38±0.09**

Table 2: Parsing accuracy of the top-down and left-to-right pointer-network-based parsers on test datasets of twelve languages from UD. Best results for each language are shown in bold and, apart from the average UAS and LAS, we also report the corresponding standard deviation over 3 runs.

achieving, under the exact same conditions, a 23.08-sentences-per-second speed on the PTB-SD compared to 10.24 of the original system.[7]

5 Related work

There is previous work that proposes to implement dependency parsing by independently selecting the head of each word in a sentence, using neural networks. In particular, Zhang et al. (2017) make use of a BiLSTM-based neural architecture to compute the probability of attaching each word to one of the other input words, in a similar way as pointer networks do. During decoding, a post-processing step is needed to produce well-formed trees by means of a maximum spanning tree algorithm. Our approach does not need this post-processing, as cycles are forbidden during parsing instead, and achieves a higher accuracy thanks to the pointer network architecture and the use of information about previous dependencies.

Before Ma et al. (2018) presented their top-down parser, Chorowski et al. (2017) had already employed pointer networks (Vinyals et al., 2015) for dependency parsing. Concretely, they developed a pointer-network-based neural architecture with multitask learning able to perform pre-processing, tagging and dependency parsing exclusively by reading tokens from an input sentence, without needing POS tags or pre-trained word embeddings. Like our approach, they also use the capabilities provided by pointer networks to undertake the parsing task as a simple process of attaching each word as dependent of another. They also try to improve the network performance with POS tag prediction as auxiliary task and with different approaches to perform label prediction. They do not exclude cycles, neither by forbidding them at parsing time or by removing them by post-processing, as they report that their system produces parses with a negligible amount of cycles, even with greedy decoding (matching our observation for our own system, in our case with beam-search decoding). Finally, the system developed by Chorowski et al. (2017) is constrained to projective dependencies, while our approach can handle unrestricted non-projective structures.

6 Conclusion

We present a novel left-to-right dependency parser based on pointer networks. We follow the same neural network architecture as the stack-pointer-based approach developed by Ma et al. (2018), but just using a focus word index instead of a buffer and a stack. Apart from doubling their system's speed, our approach proves to be a competitive alternative on a variety of languages and achieves the best accuracy to date on the PTB-SD.

The good performance of our algorithm can be explained by the shortening of the transition sequence length. In fact, it has been proved by several studies (Fernández-González and Gómez-Rodríguez, 2012; Qi and Manning, 2017; Fernández-González and Gómez-Rodríguez, 2018) that by reducing the number of applied transitions, the impact of error propagation is alleviated, yielding more accurate parsers.

Our system's source code is freely available at `https://github.com/danifg/Left2Right-Pointer-Parser`.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from MINECO (FFI2014-51978-C2-2-R, TIN2017-85160-C2-1-R) and from Xunta de Galicia (ED431B 2017/01).

---

[7] Please note that the implementation by Ma et al. (2018), also used by our novel approach, was not optimized for speed and, therefore, the reported speeds are just intended for comparing algorithms implemented under the same framework, but not to be considered as the best speed that a pointer-network-based system can potentially achieve.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack LSTM parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2005–2010.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214. Association for Computational Linguistics.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jan Chorowski, Michal Zapotoczny, and Pawel Rychlikowski. 2017. Read, tag, and parse all at once, or fully-neural dependency parsing. *CoRR*, abs/1609.03441.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, New York, NY, USA. ACM.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 334–343.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2012. Improving transition-based dependency parsing with buffer transitions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 308–319. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Non-projective dependency parsing with non-local transitions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 693–700. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1492–1501.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69. Asian Federation of Natural Language Processing.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1403–1414.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Comput. Linguist.*, 37:197–230.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160. ACL/SIGPARSE.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 110–117.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 323–333.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 665–676.

Viable Dependency Parsing as Sequence Labeling

Michalina Strzyz **David Vilares** **Carlos Gómez-Rodríguez**
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Research Group, Departamento de Computación
Campus de Elviña, s/n, 15071 A Coruña, Spain
{michalina.strzyz,david.vilares,carlos.gomez}@udc.es

Abstract

We recast dependency parsing as a sequence labeling problem, exploring several encodings of dependency trees as labels. While dependency parsing by means of sequence labeling had been attempted in existing work, results suggested that the technique was impractical. We show instead that with a conventional BILSTM-based model it is possible to obtain fast and accurate parsers. These parsers are conceptually simple, not needing traditional parsing algorithms or auxiliary structures. However, experiments on the PTB and a sample of UD treebanks show that they provide a good speed-accuracy tradeoff, with results competitive with more complex approaches.

1 Introduction

The application of neural architectures to syntactic parsing, and especially the ability of long short-term memories (LSTMs) to obtain context-aware feature representations (Hochreiter and Schmidhuber, 1997), has made it possible to parse natural language with conceptually simpler models than before. For example, in dependency parsing, the rich feature models with dozens of features used in transition-based approaches (Zhang and Nivre, 2011) can be simplified when using feedforward neural networks (Chen and Manning, 2014), and even more with BiLSTM architectures (Kiperwasser and Goldberg, 2016), where in fact two positional features can suffice (Shi et al., 2017). Similarly, in graph-based approaches, Dozat and Manning (2017) have shown that an arc-factored model can achieve state-of-the-art accuracy, without the need for the higher-order features used in systems like (Koo and Collins, 2010).

In the same way, neural feature representations have made it possible to relax the need for structured representations. This is the case of sequence-to-sequence models that translate sentences into linearized trees, which were first applied to constituent (Vinyals et al., 2015) and later to dependency parsing (Wiseman and Rush, 2016; Zhang et al., 2017b; Li et al., 2018). Recently, Gómez-Rodríguez and Vilares (2018) have shown that sequence labeling models, where each word is associated with a label (thus simpler than sequence to sequence, where the mapping from input to output is not one to one) can learn constituent parsing.

Contribution We show that sequence labeling is useful for dependency parsing, in contrast to previous work (Spoustová and Spousta, 2010; Li et al., 2018). We explore four different encodings to represent dependency trees for a sentence of length n as a set of n labels associated with its words. We then use these representations to perform dependency parsing with an off-the-shelf sequence labeling model. The results show that we produce models with an excellent speed-accuracy tradeoff, without requiring any explicit parsing algorithm or auxiliary structure (e.g. stack or buffer). The source code is available at `https://github.com/mstrise/dep2label`

2 Parsing as sequence labeling

Sequence labeling is a structured prediction problem where a single output label is generated for every input token. This is the case of tasks such as PoS tagging, chunking or named-entity recognition, for which different approaches obtain accurate results (Brill, 1995; Ramshaw and Marcus, 1999; Reimers and Gurevych, 2017).

On the contrary, previous work on dependency parsing as sequence labeling is vague and reports results that are significantly lower than those provided by transition-, graph-based or sequence-to-sequence models (Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Zhang et al., 2017a). Spoustová and Spousta

Proceedings of NAACL-HLT 2019, pages 717–723
Minneapolis, Minnesota, June 2 - June 7, 2019.

(2010) encoded dependency trees using a relative PoS-based scheme to represent the head of a node, to then train an averaged perceptron. They did not provide comparable results, but claimed that the accuracy was between 5-10% below the state of the art in the pre-deep learning era. Recently, Li et al. (2018) used a relative positional encoding of head indexes with respect to the target token. This is used to train Bidirectional LSTM-CRF sequence-to-sequence models (Huang et al., 2015), that make use of sub-root decomposition. They compared their performance against an equivalent BiLSTM-CRF labeling model. The reported UAS for the sequence labeling model was 87.6% on the Penn Treebank, more than 8 points below the current best model (Ma et al., 2018), concluding that sequence-to-sequence models are required to obtain competitive results.

In this work, we show that these results can be clearly improved if simpler architectures are used.

3 Encoding of trees and labels

Given a sentence $w_1 \dots w_n$, we associate the words with nodes $\{0, 1, \dots, n\}$, where the extra node 0 is used as a dummy root for the sentence. A dependency parser will find a set of labeled relations encoded as edges of the form (h, d, l), where $h \in \{0, 1, \dots, n\}$ is the head, $d \in \{1, \dots, n\}$ the dependent, and l a dependency label. The resulting dependency graph must be acyclic and such that each node in $\{1, \dots, n\}$ has exactly one head, so it will be a directed tree rooted at node 0.

Thus, to encode a dependency tree, it suffices to encode the unique head position and dependency label associated with each word of $w_1 \dots w_n$. To do so, we will give each word w_i a discrete label of the form (x_i, l_i), where l_i is the dependency label and x_i encodes the position of the head in one of the following four ways (see also Figure 1):

1. Naive positional encoding: x_i directly stores the position of the head, i.e., a label (x_i, l_i) encodes an edge (x_i, i, l_i). This is the encoding used in the CoNLL file format.

2. Relative positional encoding: x_i stores the difference between the head index minus that of the dependent, i.e., (x_i, l_i) encodes an edge $(i + x_i, i, l_i)$. This was the encoding used for the sequence-to-sequence and sequence labeling models in (Li et al., 2018), as well as for the sequence-to-sequence model in (Kiperwasser and Ballesteros, 2018).

3. Relative PoS-based encoding: x_i is a tuple p_i, o_i. If $o_i > 0$, the head of w_i is the o_ith closest among the words to the right of w_i that have PoS tag p_i. If $o_i < 0$, the head of w_i is the $-o_i$th closest among the words to the left of w_i that have PoS tag p_i. For example, $(V, -2)$ means "the second verb to the left" of w_i. This scheme is closer to the notion of valency, and was used by Spoustová and Spousta (2010).

4. Bracketing-based encoding: based on (Yli-Jyrä, 2012; Yli-Jyrä and Gómez-Rodríguez, 2017). In each label (x_i, l_i), the component x_i is a string following the regular expression `(<)?((\)*|(/)*)(>)?` where the presence of character $<$ means that w_{i-1} has an incoming arc from the right, k copies of character $\backslash$ mean that w_i has k outgoing arcs towards the left, k copies of $/$ mean that w_{i-1} has k outgoing arcs towards the right, and the presence of $>$ means that w_i has an incoming arc from the left. Thus, each right dependency from a word i to j is encoded by a $(/, >)$ pair in the label components x_{i+1} and x_j, and each left dependency from j to i by a $(<, \backslash)$ pair in the label components x_{i+1} and x_j. Note that the intuition that explains why information related to a word is encoded in a neighboring node is that each x_i corresponds to a fencepost position (i.e., x_i represents the space between w_{i-1} and w_i), and the character pair associated to an arc is encoded in the most external fencepost positions covered by that arc. These pairs act as pairs of matching brackets, which can be decoded using a stack to reconstruct the dependencies.

The first three encodings can represent any dependency tree, as they encode any valid head position for each node, while the bracketing encoding only supports projective trees, as it assumes that brackets are properly nested. All the encodings are total and injective, but they are not surjective: head indexes can be out of range in the first three encodings, brackets can be unbalanced in encoding 4, and all the encodings can generate graphs with cycles. We will deal with ill-formed trees later.

4 Model

We use a standard encoder-decoder network, to show that dependency parsing as sequence label-

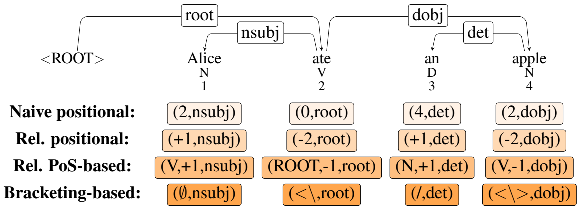


Figure 1: Types of encoding on an example tree.

ing works without the need of complex models.

Encoder We use bidirectional LSTMs (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Let $\text{LSTM}_\theta(\mathbf{x})$ be an abstraction of a long short-term memory network that processes the sequence of vectors $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_{|\mathbf{x}|}]$, then output for $\mathbf{x}_i$ is defined as $\mathbf{h}_i = \text{BiLSTM}_\theta(\mathbf{x}, i) = \text{LSTM}^l_\theta(\mathbf{x}_{[1:i]}) \circ \text{LSTM}^r_\theta(\mathbf{x}_{[|\mathbf{x}|:i]})$. We consider stacked BiLSTMs, where the output $\mathbf{h}^m_i$ of the mth BiLSTM layer is fed as input to the m+1th layer. Unless otherwise specified, the input token at a given time step is the concatenation of a word, PoS tag, and another word embedding learned through a character LSTM.

Decoder We use a feed-forward network, which is fed the output of the last BiLSTM. The output is computed as $P(y_i|\mathbf{h}_i) = \text{softmax}(W \cdot \mathbf{h}_i + \mathbf{b})$.

Well-formedness (i) Each token must be assigned a head (one must be the dummy root), and (ii) the graph must be acyclic. If no token is the real root (no head is the dummy root), we search for candidates by relying on the three most likely labels for each token.[1] If none is found, we assign it to the first token of the sentence. The single-head constraint is ensured by the nature of the encodings themselves, but some of the predicted head indexes might be out of bounds. If so, we attach those tokens to the real root. If a cycle exists, we do the same for the leftmost token in the cycle.

5 Experiments

We use the English Penn Treebank (PTB) (Marcus et al., 1993) and its splits for parsing. We transform it into Stanford Dependencies (De Marneffe et al., 2006) and obtain the predicted PoS tags using Stanford tagger (Toutanova et al., 2003). We also select a sample of UDv2.2 treebanks (Nivre et al., 2018): Ancient-Greek$_{\text{PROIEL}}$, Czech$_{\text{PDT}}$, Chinese$_{\text{GSD}}$, English$_{\text{EWT}}$, Finnish$_{\text{TDT}}$, Hebrew$_{\text{HTB}}$, Kazakh$_{\text{KTB}}$ and Tamil$_{\text{TTB}}$, as a representative sample, following (de Lhoneux et al., 2017). As evaluation metrics, we use Labeled (LAS) and Unlabeled Attachment Score (UAS). We measure speed in sentences/second, both on a single core of a CPU[2] and on a GPU[3].

Encoding	UAS	LAS
Li et al. (2018) (sequence labeling)	87.58	83.81
Li et al. (2018) (seq2seq)	89.16	84.99
Li et al. (2018) (seq2seq+beam+subroot)	93.84	**91.86**
Naive positional	45.41	42.65
Rel. positional	91.05	88.67
Rel. PoS-based	**93.99**	91.76
Bracketing-based	93.45	91.17

Table 1: Performance of our encodings on the PTB dev set with hyperparameters from Gómez-Rodríguez and Vilares (2018). We compare against previous sequence labeling and seq2seq models with more complex architectures, beam search and subroot decomposition.

Setup We use NCRFpp as our sequence labeling framework (Yang and Zhang, 2018). For PTB, we use the embeddings by Ling et al. (2015), for comparison to BIST parser (Kiperwasser and Goldberg, 2016), which uses a similar architecture, but also needs a parsing algorithm and auxiliary structures. For UD, we follow an end-to-end setup and run UDPipe[4] (Straka and Straková, 2017) for tokenization and tagging. We use the pretrained word embeddings by Ginter et al. (2017). Appendix A contains additional hyperparameters.

5.1 Encoding evaluation and model selection

We first examine the four encodings on the PTB dev set. Table 1 shows the results and also compares them against Li et al. (2018), who proposed seq2seq and sequence labeling models that use a relative positional encoding.

As the relative PoS-based encoding and bracketing-based encoding provide the best results, we will conduct the rest of our experiments with these two encodings. Furthermore, we perform a small hyperparameter search involving encoding, number of hidden layers, their dimension and presence of character embeddings, as these parameters influence speed and accuracy. From now on, we write $P^{\text{z}}_{\text{x,y}}$ for a PoS-based encoding model and $B^{\text{z}}_{\text{x,y}}$ for a bracketing-based encoding

---

[1] If single-rooted trees are a prerequisite, the most probable node will be selected among multiple root nodes.

[2] Intel Core i7-7700 CPU 4.2 GHz.

[3] GeForce GTX 1080.

[4] The pretrained models from the CoNLL18 Shared Task.

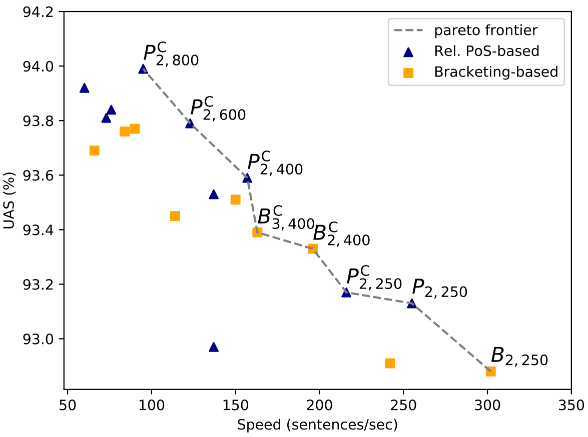


Figure 2: UAS/speed Pareto front on the PTB dev set.

model, where z indicates whether character representation was used in the model, x the number of BiLSTM layers, and y the word hidden vector dimension. We take as starting points (1) the hyperparameters used by the BIST parser (Kiperwasser and Goldberg, 2016), as it uses a BiLSTM architecture analogous to ours, with the difference that it employs a transition-based algorithm that uses a stack data structure instead of plain sequence labeling without explicit representation of structure, and (2) the best hyperparameters used by Gómez-Rodríguez and Vilares (2018) for constituent parsing as sequence labeling, as it is an analogous task for a different parsing formalism.

From there, we explore different combinations of parameters and evaluate 20 models on the PTB development set, with respect to accuracy (UAS) and speed (sentences/second on a single CPU core), obtaining the Pareto front in Figure 2. The two starting models based on previous literature ($P_{2,250}$ and $P^C_{2,800}$, respectively) happen to be in the Pareto front, confirming that they are reasonable hyperparameter choices also for this setting. In addition, we select two more models from the Pareto front (models $P^C_{2,400}$ and $B_{2,250}$) for our test set experiments on PTB, as they also provide a good balance between speed and accuracy.

Model	sent/s CPU	sent/s GPU	UAS	LAS
$P_{2,250}$	$267_{\pm1}$	$777_{\pm24}$	92.95	90.96
$P^C_{2,400}$	$165_{\pm1}$	$700_{\pm5}$	93.34	91.34
$P^C_{2,800}$	$101_{\pm2}$	$648_{\pm20}$	**93.67**	**91.72**
$B_{2,250}$	$310_{\pm30}$	$730_{\pm53}$	92.64	90.59
KG (transition-based)	$76_{\pm1}$		93.90	91.90
KG (graph-based)	$80_{\pm0}$		93.10	91.00
CM	$654^\diamond$		91.80	89.60
DM		$411^\diamond$	95.74	94.08
Ma et al. (2018)		$10_{\pm0}$	95.87	94.19

Table 2: Comparison of models on the PTB test set. KG refers to Kiperwasser and Goldberg (2016), CM to Chen and Manning (2014) and DM to Dozat and Manning (2017). ◇ indicates the speed is taken from their paper.

	UPoS-based			XPoS-based		
Treebank	UAS	LAS	# UPoS	UAS	LAS	# XPoS
Ancient Greek	76.58	71.70	14	**77.00**	**72.14**	23
Chinese	**61.01**	**57.28**	15	60.98	57.14	42
Czech	**89.82**	**87.63**	17	88.33	85.46	1417
English	**82.22**	**78.96**	17	82.05	78.70	50
Finnish	**80.31**	**76.39**	15	80.19	76.28	12
Hebrew[5]	67.23	62.86	17	67.23	62.86	17
Kazakh[6]	32.14	17.03	15	**32.93**	**17.07**	26
Tamil	**73.24**	**66.51**	13	59.70	52.57	210

Table 3: Performance of the $P^C_{2,800}$ model with UPoS- and XPoS-based encoding for each language on the dev set. # UPoS/XPoS represents the number of distinct UPoS/XPoS tags in the training set for each language.

5.2 Results and discussion

Table 2 compares the chosen models, on the PTB test set, against state-of-the-art models. Contrary to previous dependency-parsing-as-sequence-labeling attempts, we are competitive and provide a good speed-accuracy tradeoff. For instance, the $P^C_{2,800}$ model runs faster than the BIST parser (Kiperwasser and Goldberg, 2016) while being almost as accurate (-0.18 LAS). This comes in spite of its simplicity. While our BiLSTM architecture is similar to that of BIST, the sequence labeling approach does not need a stack, a specific transition system or a dynamic oracle. Using the BIST hyperparameters for our model ($P_{2,250}$) yields further increases in speed, at some cost to accuracy: 3.34x faster and -0.04 LAS score than the graph-based model, and 3.51x faster and

Treebank	PoS type	$\mathbf{P^C_{2,800}}$ **(sent/s) CPU**	**UAS**	**LAS**	**KG (transition-based) (sent/s) CPU**	**UAS**	**LAS**
Ancient Greek	XPOS	$123_{\pm1}$	75.31	70.87	$116_{\pm4}$	69.43	64.41
Chinese	UPOS	$105_{\pm0}$	63.20	59.12	$73_{\pm1}$	64.69	60.45
Czech	UPOS	$125_{\pm1}$	89.10	86.68	$94_{\pm3}$	89.25	86.11
English	UPOS	$139_{\pm1}$	81.48	78.64	$120_{\pm2}$	82.22	79.00
Finnish	UPOS	$168_{\pm0}$	80.12	76.22	$127_{\pm3}$	80.99	76.63
Hebrew	equal PoS	$120_{\pm0}$	63.04	58.66	$70_{\pm1}$	63.56	58.80
Kazakh	XPOS	$283_{\pm3}$	32.93	17.07	$178_{\pm5}$	23.09	12.73
Tamil[7]	UPOS	$150_{\pm2}$	71.59	64.00	$127_{\pm3}$	75.41	68.58

Table 4: Comparison on UD-CoNLL18 test sets.

---

[5] In Hebrew, UPoS and XPoS tags are the same.

[6] Kazakh is missing a development set. The scores are based on the test set.

[7] Tamil was run on gold segmented and tokenized inputs, as there is no pretrained UDpipe model. We did not use pretrained word embeddings either.

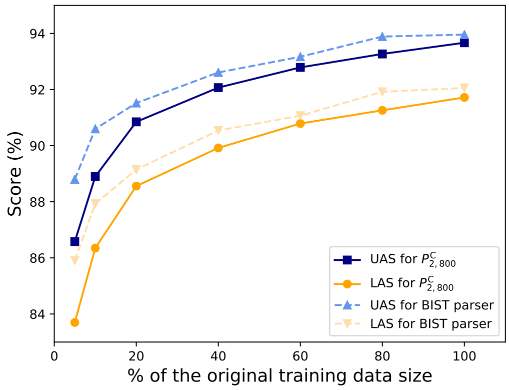


Figure 3: Impact of the PTB data size available for parsers during training on the results from the test set.

-0.94 LAS score than their transition-based one.

We now extend our experiments to the sample of UD-CoNLL18 treebanks. To this end, we focus on the $P^C_{2,800}$ model and since our PoS tag-based encoding can be influenced by the specific PoS tags used, we first conduct an experiment on the development sets to determine what tag set (UPoS, the universal PoS tag set, common to all languages, or XPoS, extended language-specific PoS tags) produces the best results for each dataset.

Table 3 shows how the number of unique UPoS and XPoS tags found in the training set differs in various languages. The results suggest that the performance of our system can be influenced by the size of the tag set. It appears that a very large tag set (for instance the XPoS tag set for Czech and Tamil) can hurt the performance of the model and significantly slow down the system, as it results into a large number of distinct labels for the sequence labeling model, increasing sparsity and making the classification harder. In case of Ancient Greek and Kazakh, the best performance is achieved with the XPoS-based encoding. In these corpora, the tag set is slightly bigger than the UPoS tag set. One can argue that the XPoS tags in this case were possibly more fine-grained and hence provided additional useful information to the system facilitating a correct label prediction, without being so large as to produce excessive sparsity.

Table 4 shows experiments on the UD test sets, with the chosen PoS tag set for each corpus. $P^C_{2,800}$ outperforms transition-based BIST in LAS in 3 out of 8 treebanks,[8] and is clearly faster in all analyzed languages. We believe that the variations between languages in terms of LAS difference with respect to BIST can be largely due to differences in the accuracy and granularity of predicted PoS tags, since our chosen encoding relies on them to encode arcs. The bracketing-based encoding, which does not use PoS tags, may be more robust to this. On the other hand, finding the optimal granularity of PoS tags for the PoS-based encoding can be an interesting avenue for future work.

In this work, we have also examined the impact of the training data size on the performance of our system compared to the performance of BIST parser. The results in Figure 3 suggest that our model requires more data during the training than BIST parser in order to achieve similar performance. The performance is slightly worse when little training data is available, but later on our model reduces the gap when increasing the training data size.

6 Conclusion

This paper has explored fast and accurate dependency parsing as sequence labeling. We tested four different encodings, training a standard BiLSTM-based architecture. In contrast to previous work, our results on the PTB and a subset of UD treebanks show that this paradigm can obtain competitive results, despite not using any parsing algorithm nor external structures to parse sentences.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). We gratefully acknowledge NVIDIA Corporation for the donation of a GTX Titan X GPU.

---

[8] For Ancient Greek, this may be related to the large amount of non-projectivity (BIST is a projective parser). For extra comparison, a non-projective variant of BIST (Smith et al., 2018) obtains 71.58 LAS with mono-treebank training, but from better segmentation and morphology than used here. UDpipe (Straka and Straková, 2017) obtains 67.57 LAS. Czech and Kazakh have a medium amount of non-projectivity.

References

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343. Association for Computational Linguistics.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden. Association for Computational Linguistics.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *TLT*, pages 99–110.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304. Association for Computational Linguistics.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414. Association for Computational Linguistics.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*, 19(2):313–330.

Joakim Nivre et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference* *on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark. Association for Computational Linguistics.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123. Association for Computational Linguistics.

Drahomíra Spoustová and Miroslav Spousta. 2010. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94(1):7–14.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Anssi Yli-Jyrä. 2012. *On Dependency Analysis via Contractions and Weighted FSTs*, pages 133–158. Springer Berlin Heidelberg, Berlin, Heidelberg.

Anssi Yli-Jyrä and Carlos Gómez-Rodríguez. 2017. Generic axiomatization of families of noncrossing graphs in dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1745–1755, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017a. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 665–676.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2017b. Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1682, Copenhagen, Denmark. Association for Computational Linguistics.

A Model parameters

During the training we use Stochastic Gradient Descent (SGD) optimizer with a batch size of 8, and the model is trained for up to 100 iterations. We keep the model that obtains the highest UAS on the development set. Additional hyperparameters are shown in Table 5.

Word embedding dimension	100
Char embedding dimension	30
PoS tag embedding dimension	25
Word hidden vector dimension	250,400,600, 800,1000,1200
Character hidden vector dimension	50
Initial learning rate	0.02
Time-based learning rate decay	0.05
Momentum	0.9
Dropout	0.5

Table 5: Common hyperparameters for the sequence labeling models.

Pooled Contextualized Embeddings for Named Entity Recognition

Alan Akbik
Zalando Research
Mühlenstraße 25
10243 Berlin

Tanja Bergmann
Zalando Research
Mühlenstraße 25
10243 Berlin

Roland Vollgraf
Zalando Research
Mühlenstraße 25
10243 Berlin

`{firstname.lastname}@zalando.de`

Abstract

Contextual string embeddings are a recent type of contextualized word embedding that were shown to yield state-of-the-art results when utilized in a range of sequence labeling tasks. They are based on character-level language models which treat text as distributions over characters and are capable of generating embeddings for any string of characters within any textual context. However, such purely character-based approaches struggle to produce meaningful embeddings if a rare string is used in a underspecified context. To address this drawback, we propose a method in which we dynamically aggregate contextualized embeddings of each unique string that we encounter. We then use a pooling operation to distill a global word representation from all contextualized instances. We evaluate these *pooled contextualized embeddings* on common named entity recognition (NER) tasks such as CoNLL-03 and WNUT and show that our approach significantly improves the state-of-the-art for NER. We make all code and pre-trained models available to the research community for use and reproduction.

1 Introduction

Word embeddings are a crucial component in many NLP approaches (Mikolov et al., 2013; Pennington et al., 2014) since they capture latent semantics of words and thus allow models to better train and generalize. Recent work has moved away from the original "one word, one embedding" paradigm to investigate *contextualized* embedding models (Peters et al., 2017, 2018; Akbik et al., 2018). Such approaches produce different embeddings for the same word depending on its context and are thus capable of capturing latent contextualized semantics of ambiguous words.

B-PER	E-PER		S-LOC		S-ORG
Fung	Permadi	(	Taiwan	) v	Indra

Figure 1: Example sentence that provides underspecified context. This leads to an underspecified contextual word embedding for the string "*Indra*" that ultimately causes a misclassification of "*Indra*" as an organization (ORG) instead of person (PER) in a downstream NER task.

Recently, Akbik et al. (2018) proposed a character-level contextualized embeddings approach they refer to as *contextual string embeddings*. They leverage pre-trained character-level language models from which they extract hidden states at the beginning and end character positions of each word to produce embeddings for any string of characters in a sentential context. They showed these embeddings to yield state-of-the-art results when utilized in sequence labeling tasks such as named entity recognition (NER) or part-of-speech (PoS) tagging.

Underspecified contexts. However, such contextualized character-level models suffer from an inherent weakness when encountering rare words in an underspecified context. Consider the example text segment shown in Figure 1: "*Fung Permadi (Taiwan) v Indra*", from the English CONLL-03 test data split (Tjong Kim Sang and De Meulder, 2003). If we consider the word "*Indra*" to be rare (meaning no prior occurrence in the corpus used to generate word embeddings), the underspecified context allows this word to be interpreted as either a person or an organization. This leads to an underspecified embedding that ultimately causes an incorrect classification of "*Indra*" as an organization in a downstream NER task.

Pooled Contextual Embeddings. In this paper, we present a simple but effective approach to address this issue. We intuit that entities are normally only used in underspecified contexts if they are expected to be known to the reader. That is, they are either more clearly introduced in an earlier sentence, or part of general in-domain knowl-

Proceedings of NAACL-HLT 2019, pages 724–728
Minneapolis, Minnesota, June 2 - June 7, 2019.

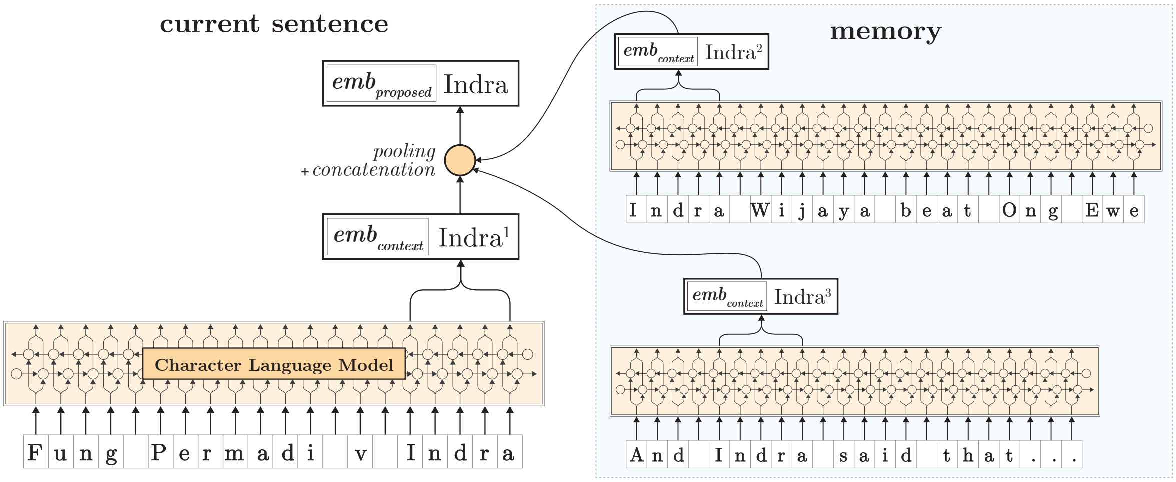


Figure 2: Example of how we generate our proposed embedding ($emb_{proposed}$) for the word "*Indra*" in the example text segment "*Fung Permadi v Indra*". We extract a contextual string embedding ($emb_{context}$) for this word and retrieve from the *memory* all embeddings that were produced for this string on previous sentences. We pool and concatenate all local contextualized embeddings to produce the final embedding.

edge a reader is expected to have. Indeed, the string "*Indra*" in the CoNLL-03 data also occurs in the earlier sentence "*Indra Wijaya (Indonesia) beat Ong Ewe Hock*".

Based on this, we propose an approach in which we dynamically aggregate contextualized embeddings of each unique string that we encounter as we process a dataset. We then use a pooling operation to distill a global word representation from all contextualized instances that we use in combination with the current contextualized representation as new word embedding. Our approach thus produces *evolving word representations* that change over time as more instances of the same word are observed in the data.

We evaluate our proposed embedding approach on the task of named entity recognition on the CoNLL-03 (English, German and Dutch) and WNUT datasets. In all cases, we find that our approach outperforms previous approaches and yields new state-of-the-art scores. We contribute our approach and all pre-trained models to the open source FLAIR[1] framework (Akbik et al., 2019), to ensure reproducibility of these results.

2 Method

Our proposed approach (see Figure 2) dynamically builds up a "memory" of contextualized embeddings and applies a pooling operation to distill a global contextualized embedding for each word. It requires an *embed*() function that produces a contextualized embedding for a given *word* in a *sentence* context (see Akbik et al. (2018)). It also requires a *memory* that records for each unique word all previous contextual embeddings, and a *pool*() operation to pool embedding vectors.

This is illustrated in Algorithm 1: to embed a word (in a sentential context), we first call the *embed*() function (line 2) and add the resulting embedding to the memory for this word (line 3). We then call the pooling operation over all contextualized embeddings for this word in the memory (line 4) to compute the pooled contextualized embedding. Finally, we concatenate the original contextual embedding together with the pooled representation, to ensure that both local and global interpretations are represented (line 5). This means that the resulting *pooled contextualized embedding* has twice the dimensionality of the original embedding.

Algorithm 1 Compute pooled embedding

```
Input: $sentence$, $memory$
1: for $word$ in $sentence$ do
2:     $emb_{context} \leftarrow$
           embed($word$) within $sentence$
3:     add $emb_{context}$ to $memory[word]$
4:     $emb_{pooled} \leftarrow$ pool($memory[word]$)
5:     $word.embedding \leftarrow$
           concat($emb_{pooled}$, $emb_{context}$)
6: end for
```

Pooling operations. We experiment with different pooling operations: *mean* pooling to average a word's contextualized embedding vectors, and *min* and *max* pooling to compute a vector of all

---

[1] https://github.com/zalandoresearch/flair

Approach	CoNLL-03 En	CoNLL-03 De	CoNLL-03 Nl	WNUT-17
Pooled Contextualized Embeddings$_{min}$	**93.18** ± 0.09	**88.27** ± 0.30	90.12 ± 0.14	49.07 ± 0.31
Pooled Contextualized Embeddings$_{max}$	93.13 ± 0.09	88.05 ± 0.25	90.26 ± 0.10	49.05 ± 0.26
Pooled Contextualized Embeddings$_{mean}$	93.10 ± 0.11	87.69 ± 0.27	**90.44** ± 0.20	**49.59** ± 0.41
Contextual String Emb. (Akbik et al., 2018)	92.86 ± 0.08	87.41 ± 0.13	90.16 ± 0.26	49.49 ± 0.75
best published				
BERT (Devlin et al., 2018)†	92.8			
CVT+Multitask (Clark et al., 2018)†	92.6			
ELMo (Peters et al., 2018)†	92.22			
Stacked Multitask (Aguilar et al., 2018)†				45.55
Character-LSTM (Lample et al., 2016)†	90.94	78.76	81.74	

Table 1: Comparative evaluation of proposed approach with different pooling operations (min, max, $mean$) against current state-of-the-art approaches on four named entity recognition tasks († indicates reported numbers). The numbers indicate that our approach outperforms all other approaches on the CoNLL datasets, and matches baseline results on WNUT.

element-wise minimum or maximum values.

Training downstream models. When training downstream task models (such as for NER), we typically make many passes over the training data. As Algorithm 2 shows, we reset the memory at the beginning of each pass over the training data (line 2), so that it is build up from scratch at each epoch.

Algorithm 2 Training

```
1: for $epoch$ in $epochs$ do
2:     $memory \leftarrow$ map of $word$ to list
3:     train and evaluate as usual
4: end for
```

This approach ensures that the downstream task model learns to leverage pooled embeddings that are built up (e.g. *evolve*) over time. It also ensures that pooled embeddings during training are only computed over training data. After training, (i.e. during NER prediction), we do not reset embeddings and instead allow our approach to keep expanding the memory and evolve the embeddings.

3 Experiments

We verify our proposed approach in four named entity recognition (NER) tasks: We use the English, German and Dutch evaluation setups of the CoNLL-03 shared task (Tjong Kim Sang and De Meulder, 2003) to evaluate our approach on classic newswire data, and the WNUT-17 task on emerging entity detection (Derczynski et al., 2017) to evaluate our approach in a noisy user-generated data setting with few repeated entity mentions.

3.1 Experimental Setup

We use the open source Flair framework in all our experiments. It implements the standard BiLSTM-CRF sequence labeling architecture (Huang et al., 2015) and includes pre-trained contextual string embeddings for many languages. To Flair, we add an implementation of our proposed *pooled contextualized embeddings*.

Hyperparameters. For our experiments, we follow the training and evaluation procedure outlined in Akbik et al. (2018) and follow most hyperparameter suggestions as given by the in-depth study presented in Reimers and Gurevych (2017). That is, we use an LSTM with 256 hidden states and one layer (Hochreiter and Schmidhuber, 1997), a locked dropout value of 0.5, a word dropout of 0.05, and train using SGD with an annealing rate of 0.5 and a patience of 3. We perform model selection over the learning rate $\in \{0.01, 0.05, 0.1\}$ and mini-batch size $\in \{8, 16, 32\}$, choosing the model with the best F-measure on the validation set. Following Peters et al. (2017), we then repeat the experiment 5 times with different random seeds, and train using both train and development set, reporting both average performance and standard deviation over these runs on the test set as final performance.

Standard word embeddings. The default setup of Akbik et al. (2018) recommends contextual string embeddings to be used in combination with standard word embeddings. We use GloVe embeddings (Pennington et al., 2014) for the English tasks and FastText embeddings (Bojanowski et al., 2017) for all newswire tasks.

Baselines. Our baseline are contextual string embeddings without pooling, i.e. the original setup proposed in Akbik et al. (2018)[2]. By comparing against this baseline, we isolate the impact of our proposed pooled contextualized embeddings.

---

[2] Our reproduced numbers are slightly lower than we reported in Akbik et al. (2018) where we used the official CoNLL-03 evaluation script over BILOES tagged entities. This introduced errors since this script was not designed for S-tagged entities.

Approach	CoNLL-03 En	CoNLL-03 De	CoNLL-03 Nl	WNUT-17
Pooled Contextualized Embeddings (*only*)	**92.42** ± 0.07	**86.21** ± 0.07	**88.25** ± 0.11	**44.29** ± 0.59
Contextual String Embeddings (*only*)	91.81 ± 0.12	85.25 ± 0.21	86.71 ± 0.12	43.43 ± 0.93

Table 2: Ablation experiment using contextual string embeddings without word embeddings. We find a more significant impact on evaluation numbers across all datasets, illustrating the need for capturing global next to contextualized semantics.

In addition, we list the best reported numbers for the four tasks. This includes the recent BERT approach using bidirectional transformers by Devlin et al. (2018), the semi-supervised multitask learning approach by Clark et al. (2018), the ELMo word-level language modeling approach by Peters et al. (2018), and the best published numbers for WNUT-17 (Aguilar et al., 2018) and German and Dutch CoNLL-03 (Lample et al., 2016).

3.2 Results

Our experimental results are summarized in Table 1 for each of the four tasks.

New state-of-the-art scores. We find that our approach outperforms all previously published results, raising the state-of-the-art for CoNLL-03 on English to 93.18 F1-score (↑0.32 pp vs. previous best), German to 88.27 (↑0.86 pp) and Dutch to 90.44 (↑0.28 pp). The consistent improvements against the contextual string embeddings baseline indicate that our approach is generally a viable option for embedding entities in sequence labeling.

Less pronounced impact on WNUT-17. However, we also find no significant improvements on the WNUT-17 task on emerging entities. Depending on the pooling operation, we find comparable results to the baseline. This result is expected since most entities appear only few times in this dataset, giving our approach little evidence to aggregate and pool. Nevertheless, since recent work has not yet experimented with contextual embeddings on WNUT, as side result we report a new state-of-the-art of 49.59 F1 vs. the previous best reported number of 45.55 (Aguilar et al., 2018).

Pooling operations. Comparing the pooling operations discussed in Section 2, we generally find similar results. As Table 1 shows, min pooling performs best for English and German CoNLL, while $mean$ pooling is best for Dutch and WNUT.

3.3 Ablation: Character Embeddings Only

To better isolate the impact of our proposed approach, we run experiments in which we do not use any classic word embeddings, but rather rely solely on contextual string embeddings. As Table 2 shows, we observe more pronounced improvements of pooling vis-a-vis the baseline approach in this setup. This indicates that pooled contextualized embeddings capture global semantics words similar in nature to classical word embeddings.

4 Discussion and Conclusion

We presented a simple but effective approach that addresses the problem of embedding rare strings in underspecified contexts. Our experimental evaluation shows that this approach improves the state-of-the-art across named entity recognition tasks, enabling us to report new state-of-the-art scores for CoNLL-03 NER and WNUT emerging entity detection. These results indicate that our embedding approach is well suited for NER.

Evolving embeddings. Our dynamic aggregation approach means that embeddings for the same words will change over time, even when used in exactly the same contexts. Assuming that entity names are more often used in well-specified contexts, their pooled embeddings will improve as more data is processed. The embedding model thus continues to "learn" from data even after the training of the downstream NER model is complete and it is used in prediction mode. We consider this idea of constantly evolving representations a very promising research direction.

Future work. Our pooling operation makes the conceptual simplification that all previous instances of a word are equally important. However, we may find more recent mentions of a word - such as words within the same document or news cycle - to be more important for creating embeddings than mentions that belong to other documents or news cycles. Future work will therefore examine methods to learn weighted poolings of previous mentions. We will also investigate applicability of our proposed embeddings to tasks beside NER.

Public release. We contribute our code to the FLAIR framework[3]. This allows full reproduction of all experiments presented in this paper, and al-

---

[3] The proposed embedding is added to FLAIR in release 0.4.1. as the `PooledFlairEmbeddings` class (see Akbik et al. (2019) for more details).

lows the research community to use our embeddings for training downstream task models.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732328 ("FashionBrain").

References

Gustavo Aguilar, Adrian Pastor Lopez Monroy, Fabio González, and Thamar Solorio. 2018. Modeling noisiness to recognize named entities using multitask neural networks on social media. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1401–1412.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: System Demonstrations*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. In *EMNLP*.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Better Modeling of Incomplete Annotations for Named Entity Recognition

Zhanming Jie[1], **Pengjun Xie**[2], **Wei Lu**[1], **Ruixue Ding**[2] and **Linlin Li**[2]
[1]StatNLP Research Group, Singapore University of Technology and Design
[2]DAMO Academy, Alibaba Group
`zhanming_jie@mymail.sutd.edu.sg, luwei@sutd.edu.sg`
`{chengchen.xpj, ada.drx, linyan.lll}@alibaba-inc.com`

Abstract

Supervised approaches to named entity recognition (NER) are largely developed based on the assumption that the training data is fully annotated with named entity information. However, in practice, annotated data can often be imperfect with one typical issue being the training data may contain incomplete annotations. We highlight several pitfalls associated with learning under such a setup in the context of NER and identify limitations associated with existing approaches, proposing a novel yet easy-to-implement approach for recognizing named entities with incomplete data annotations. We demonstrate the effectiveness of our approach through extensive experiments.[1]

1 Introduction

Named entity recognition (NER) (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) as one of the most fundamental tasks within natural language processing (NLP) has received significant attention. Most existing approaches to NER focused on a supervised setup, where fully annotated named entity information is assumed to be available during the training phase. However, in practice, obtaining high-quality annotations can be a very laborious and expensive process (Snow et al., 2008). One of the common issues with data annotations is there may be incomplete annotations.

Figure 1 shows an example sentence with two named entities "*John Lloyd Jones*" and "*BBC radio*" of type PER (person) and ORG (organization), respectively. Following the standard BIOES tagging scheme (Ramshaw and Marcus, 1999; Ratinov and Roth, 2009), the corresponding gold label sequence is shown below the sentence. When the data annotations are incomplete, certain labels may be missing from the label sequence. Properly defining the task is important, and we argue there are two possible potential pitfalls associated with modeling incomplete annotations, especially for the NER task.

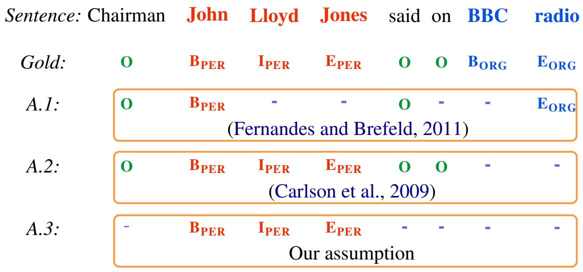


Figure 1: An example sentence with gold named entity annotations and different assumptions (i.e., *A.1* to *A.3*) on *available labels*. "-" represents a missing label.

Several previous approaches assume the incomplete annotations can be obtained by simply removing either word-level labels (Fernandes and Brefeld, 2011) or span-level labels (Carlson et al., 2009). As shown in Figure 1, under both assumptions (i.e., *A.1* and *A.2*), there will be words annotated with O labels. The former approach may even lead to sub-entity level annotations (e.g., "*radio*" is annotated as part of an entity). However, we argue such assumptions can be largely unrealistic. In practice, annotators are typically instructed to annotate named entities for complete word spans only (Settles et al., 2008; Surdeanu et al., 2010). Thus, sub-entity level annotations or O labels [2] should not be assumed to be avail-

---

[1] Our code and data are available at `http://statnlp.org/research/ie`.

[2] Why should the O labels be assumed unavailable? This is because the annotators typically do not actively specify the O labels when working on annotations. If the annotator chooses not to annotate a word, it could either mean it is not part of any entity, *or* the word is actually part of an entity but the annotator neglected it in the annotation process (therefore we have incomplete annotations). However, we note that assigning the O label to a word would precisely indicate it is strictly *not* part of any entity, which is not desirable.

Proceedings of NAACL-HLT 2019, pages 729–734
Minneapolis, Minnesota, June 2 - June 7, 2019.

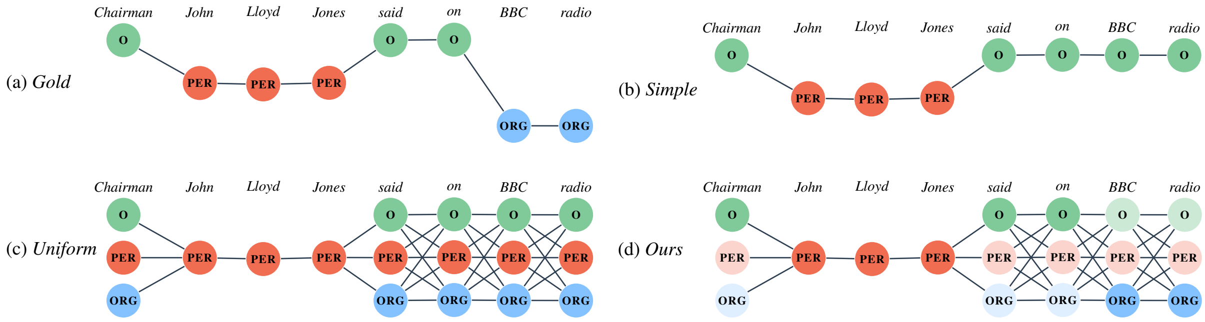

(a) *Gold*
(b) *Simple*
(c) *Uniform*
(d) *Ours*

Figure 3: Graphical illustrations on different assumptions on *unavailable labels*, where the entity "*John Lloyd Jones*" of type PER is labeled but "*BBC radio*" of type ORG is missing. Each path refers to one possible complete label sequence, and the density of the color indicates probability (we excluded B and E tags for brevity).

able (*A.3*). Therefore such approaches are making sub-optimal assumptions on the *available labels*.

When the proper assumptions on the available labels are made, one can typically model the missing labels as latent variables and train a latent-variable conditional random fields model (Quattoni et al., 2005). One such approach is presented in (Bellare and McCallum, 2007). Their work focused on the citation parsing[3] (i.e., sequence labeling) task which does not suffer from the above issue as no O label is involved. However, though the approach was shown effective in the citation parsing task, we found its effectiveness does not transfer to the NER task even in the absence of the above available labels issue. As we would highlight later, the reason is related to the undesirable assumptions on the *unavailable labels*.

In this work, we tackle the incomplete annotation problem when building an NER system, under a more realistic yet more challenging scenario. We present a novel, effective, yet easy-to-implement approach, and conduct extensive experiments on various datasets and show our approach significantly outperforms several previous approaches.

2 Related Work

Previous research efforts on partially annotated data are mostly based on the conditional random fields (CRF) (Lafferty et al., 2001), structured perceptron (Collins, 2002) and max-margin (Tsochantaridis et al., 2005) (e.g. structural support vector machine) models. Bellare and McCallum (2007) proposed a missing label linear-chain CRF[4] which is essentially a latent-variable CRF (Quattoni et al., 2005) on citation parsing (McCallum et al., 2000). This model had also been used in part-of-speeching tagging and segmentation task with incomplete annotations (Tsuboi et al., 2008; Liu et al., 2014; Yang and Vozila, 2014). Yang et al. (2018) showed the effectiveness of such a model on Chinese NER with incomplete annotations due to the fact that they required a certain number of fully annotated data to perform joint training. Greenberg et al. (2018) applied this model on a biomedical NER task and achieved promising performance with incomplete annotations. However, in their assumption for the incomplete annotations, the O labels are still considered, which we believe is not realistic. Carlson et al. (2009) modified the structured perceptron algorithm and defined features only on the tokens with annotated labels in partially labeled sequences. Fernandes and Brefeld (2011) and Lou et al. (2012) proposed to use a large-margin learning framework similar to structured support vector machines with latent variables (Yu and Joachims, 2009).

3 Approach

Given the input word sequence $\mathbf{x}$, the NER task is to predict a label sequence $\mathbf{y}$ that encodes the NER information (e.g., in a form following the BIOES tagging scheme). Given a training set that consists of completely labeled data $\mathcal{D}$, one can tackle this problem using a standard linear-chain conditional random field (CRF) (Lafferty et al., 2001) whose loss function is as follows:[5]

$$\mathcal{L}(\mathbf{w}) = -\sum_i \log p_{\mathbf{w}}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \qquad (1)$$

---

[3] The task is to tag the BibTex records with different labels (i.e., "title", "author", "affiliation" and so on).

[4] This model was also named as Partial CRF (Carlson et al., 2009) and EM Marginal CRF (Greenberg et al., 2018).

[5] In practice, we also have an L_2 regularization term, which we exclude from the formula for brevity.

where $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is the i-th instance from $\mathcal{D}$.

Now, assume we have an incomplete label sequence $\mathbf{y}_p^{(i)}$. From such a $\mathbf{y}_p^{(i)}$ we should be able to derive a set of all possible complete label sequences that are compatible with (i.e., contain) the incomplete label sequence, and let us call this set $\mathcal{C}(\mathbf{y}_p^{(i)})$. We can rewrite the above function as:

$$\mathcal{L}(\mathbf{w}) = -\sum_i \log \sum_{\mathbf{y} \in \mathcal{C}(\mathbf{y}_p^{(i)})} q_{\mathcal{D}}(\mathbf{y}|\mathbf{x}^{(i)}) p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}^{(i)})$$

We illustrate in Figure 3 several previous approaches as well as our approach. In this example, the entity *BBC radio* of type ORG is not annotated. Figure 3(a) shows a single path that corresponds to the gold label sequence. Figure 3(b) illustrates a naive approach, where we regard all the missing labels as O labels. This essentially assumes that the q distribution in the above equation puts all probability mass to this single label sequence, which is an incorrect assumption.

Now let us look at what assumptions on q have been made by the existing approach of Bellare and McCallum (2007). The model regards the missing labels as latent variables and learns a latent variable CRF using the following loss:

$$-\sum_i \log \sum_{\mathbf{y} \in \mathcal{C}(\mathbf{y}_p^{(i)})} p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}^{(i)}) \qquad (2)$$

The resulting model is called *missing label linear-chain CRF* (*M-CRF*) [6]. As we can see from the above function, this is essentially equivalent to say q is a uniform distribution that assigns equal probabilities to all possible complete label sequences in $\mathcal{C}(\mathbf{y}_p^{(i)})$.

We believe such an assumption on q that describes *unavailable labels* can be improved. As we can see from the above example in Figure 3(d), a more desirable assumption about q is to put more probability mass to a path that is close to the gold path. In practice, their approach worked for the task of citation parsing, where the q distribution may not deviate much from the uniform distribution (Figure 3(c)) in such a task. However, in the task of NER, we find such a simple treatment to the q distribution often leads to sub-optimal results (as we can see in the experiments later) as the q distribution is highly skewed due to the large amount of O labels. This observation motivates us to find a proper way to define q that can approximate the gold label distribution in this work.

Dataset	Training		Validation		Test		Entities	
	#entity	#sent	#entity	#sent	#entity	#sent	#	c (%)
CoNLL-2003	23,499	14,041	5,942	3,250	5,648	3,453	4	23.0
CoNLL-2002	18,796	8,322	4,338	1,914	3,559	1,516	4	12.4
Taobao	29,397	6,000	4,941	998	4,866	1,000	4	51.0
Youku	12,754	8,001	1,580	1,000	1,570	1,001	3	41.7

Table 1: Data statistics for the datasets.

3.1 Estimating q

Inspired by the *classifier stacking* technique used in Nivre and McDonald (2008), we empirically found that a reasonable q distribution can be acquired in a k-fold cross-validation fashion.

We first start with an initialization step where we assign specific labels to words without labels, forming complete label sequences (we will discuss our initialization strategy in experiments). Next, we perform k-fold cross-validation on the training set. Specifically, each time we train a model with (k-1) folds of the data and based on the learned model we define our q distribution.

We describe two different ways of defining the q distribution, namely the *hard* approach, and the *soft* approach. In the *hard* approach, the resulting q distribution is a collapsed distribution that assigns probability 1 to a single complete label sequence, whereas in the *soft* approach each possible label sequence will get a certain probability score.

In the *hard* approach, after training a model from (k-1) folds, we apply a constrained Viterbi procedure [7] to the sentences in the remaining fold. In the *soft* approach, we use a constrained version of the forward-backward procedure and calculate the marginal probabilities associated with each label at each unlabeled position. The score of each complete label sequence can then be calculated as a product of all such marginal probabilities. We note that in the above procedure the estimation to q depends on the initialization. Thus we iterate the above procedure, which allows us to converge to an improved q.

4 Experiments

We conduct experiments on two standard NER datasets – CoNLL-2003 English and CoNLL-2002 Spanish datasets that consist of news articles. We

---

[6] Similar assumptions have also been made by (Carlson et al., 2009; Fernandes and Brefeld, 2011), but they used structured perceptron (Collins, 2002) instead.

[7] The algorithm will ensure the resulting complete label sequence is compatible with the incomplete label sequence.

Approach	CoNLL-2003			CoNLL-2002			Taobao			Youku		
	P.	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*
Simple	93.6	68.6	79.2	86.8	57.0	68.8	83.1	46.7	59.8	91.1	49.1	63.8
LSTM-M-CRF	13.0	90.5	22.8	6.2	84.5	11.6	33.0	83.0	47.2	17.6	83.7	29.1
LSTM-Partial Perceptron	27.6	82.0	41.3	22.3	66.3	33.3	26.6	59.7	36.8	19.7	69.0	30.6
LSTM-Transductive Perceptron	11.7	90.5	20.7	6.1	84.3	11.4	32.9	82.2	47.0	16.0	81.9	26.7
Ours (*hard*)	88.1	89.9	89.0	80.8	82.1	81.5	69.3	77.4	73.1	77.2	79.8	78.5
Ours (*soft*)	89.0	90.1	**89.5**	81.3	82.7	**82.0**	69.7	78.1	**73.7**	78.1	79.6	**78.8**
Complete	91.0	90.8	90.9	85.7	85.8	85.8	82.3	82.6	82.4	83.0	81.7	82.4

Table 2: Performance comparison between different baseline models and our approaches on 4 datasets with $\rho = 0.5$ (for *Complete* model, $\rho = 1.0$).

notice that incomplete annotation issue is very common in the industry setup. Therefore we also consider two new datasets from industry – Taobao and Youku datasets[8] consisting of product and video titles in Chinese. We crawled and manually annotated such data with named entities[9]. Table 1 shows the statistics of the datasets. The last two columns show the number of entity types and the percentage of words (i.e., c in Table 1) that are parts of an NE. Based on our assumption on the *available labels* in Section 1, we randomly remove a certain number of entities as well as all O labels and use ρ to represent the ratio of annotated entities. For example, $\rho = 0.6$ means we keep 60% of all the entities and remove the annotations of 40% of the entities. Meanwhile, the O labels are considered unavailable.

We follow Lample et al. (2016) and apply the bidirectional long short-term memory (Hochreiter and Schmidhuber, 1997) (BiLSTM) networks as the neural architecture for all baselines and our approaches. Specifically, we implement the following baselines: a *Simple* model which is a linear-chain LSTM-CRF model and we treat all missing labels as O; the missing label CRF (Bellare and McCallum, 2007; Greenberg et al., 2018) (LSTM-M-CRF) model; the *partial perceptron* (Carlson et al., 2009) model, which is a structured perceptron (Collins, 2002) but only considers the scores on the words with available labels; the *transductive perceptron* (Fernandes and Brefeld, 2011) model where they introduce a Hamming loss function during the perceptron training process; lastly, we train an LSTM-CRF (Lample et al., 2016) with complete annotations as the upper bound (*Complete*). For English and Spanish, we use exactly the same embeddings used in Lample et al. (2016). We train our Chinese character embeddings on the Chinese Gigaword[10] corpus. The resulting implementation achieves 90.9 and 85.8 F-scores on CoNLL-2003 English and CoNLL-2002 Spanish datasets, respectively. These benchmark results are comparable with the results reported in the state-of-the-art NER systems (Lample et al., 2016; Ma and Hovy, 2016; Reimers and Gurevych, 2017).

For initialization in our approaches, we run the *Simple* model on each fold and use the results to initialize our q distribution[11]. Detailed descriptions on experiment settings (e.g., hidden dimension of LSTM and optimizer) and baseline systems are provided in supplementary material.

Main Results Table 2 presents the comparisons among all approaches on four datasets with $\rho = 0.5$ and $k = 2$. Our preliminary experiments show that a larger k value have a negligible effect on the results. A similar finding was also reported in Nivre and McDonald (2008). The *Simple* model has high precision and low recall as it treats unknown labels as O. Previous models for incomplete annotations achieve a much lower F-score compared to the *Simple* model and our approaches. Due to their uniform assumption on q over the missing labels, these models typically can recall more entities. The partial perceptron (Carlson et al., 2009) among these three models yields a relatively lower recall as features are not defined over the words with missing labels.

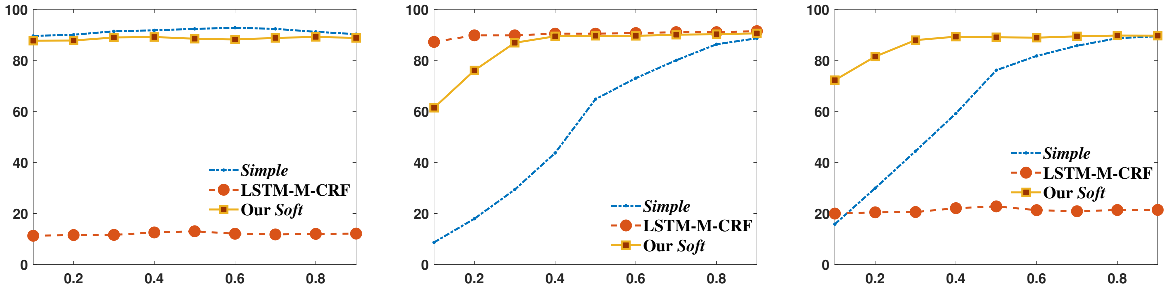
---

[8] http://www.taobao.com/ and http://www.youku.com/

[9] Details of all datasets can be found in the supplementary material.

[10] https://catalog.ldc.upenn.edu/LDC2003T09

[11] Similar to the EM procedure, a good initialization is crucial for our approach. We found using random initialization can lead to substantially worse results and a better initialization can be used to further improve the results.



(a) Precision

(b) Recall

(c) F-score

Figure 4: Precision, Recall and F-score with different ρ on CoNLL-2003 dataset.

The difference in F-score between these three models and the *Simple* model is more significant on the two CoNLL datasets than on Taobao and Youku. As shown in Table 1, the latter two datasets have more words labeled as parts of entities (i.e., a higher c). This means these industrial datasets have less O labels, making such baseline models suffer less from their assumptions on the *unavailable labels*. With a properly learned q distribution, our approaches improves the recall score over the *Simple* model while preserving a high precision. Our *soft* approach consistently achieves a better F-score compared with the *hard* approach on all datasets with $p < 0.001$. Compared to the *Complete* upper bound, our *soft* approach are still more than 3% lower in F-score on the CoNLL-2002, Taobao and Youku datasets. However, we can see that the *soft* approach achieves much higher performance compared to this variant on other datasets. We attribute this phenomenon to our approaches' ability in retrieving most of the entities in the training set. Empirically, we found our *soft* approach can recover 94% of the entities in the training set of the CoNLL-2003 dataset.

The overall results show the underlying scenario is challenging for commonly adopted models in handling incomplete annotations and our approaches can achieve better performance compared with them.

Effect of ρ We conduct experiments with different ρ from 0.1 to 0.9 for our *soft* approach against the *Simple* and LSTM-M-CRF models. Figure 4 shows how the precision, recall and F-score on CoNLL-2003 change as we increase ρ. The F-score of the *Simple* baseline increases progressively as ρ increases. LSTM-M-CRF always maintains a low F-score which is not sensitive to different ρ values because of their high recall and low precision values as we can see in Figure 4 (a, b). The improvement of our approach attributes to the increase of recall as the precision is constantly high and stable. We can see that our *soft* approach performs particularly well when ρ is larger than 0.3 which indicates a modest amount of missing labels in practice.

5 Conclusions and Future Work

In this work, we identified several limitations associated with previous assumptions when performing sequence labeling with incomplete annotations, and focused on the named entity recognition task. We presented a novel and easy-to-implement solution that works under a realistic and challenging assumption on the incomplete annotations. Through extensive experiments and analysis, we demonstrated the effectiveness of our approach.

Although we focused on the task of named entity recognition in this work, we believe the proposed approach may find applications in some other sequence labeling tasks or other more general structured prediction problems where the issue of incomplete annotations is involved. We leave them as future work.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments on this work. This work is done under a collaborative agreement between SUTD and Alibaba on an Alibaba Innovative Research (AIR) Program funded by Alibaba, where Alibaba provided data and helped with experiments. We appreciate Alibaba's generosity in the agreement that makes it possible for us to make all data and code in this research publicly available upon acceptance of this paper. This work is also partially supported by SUTD project PIE-SGP-AI-2018-01.

References

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Proceedings of International Workshop on Information Integration on the Web*.

Andrew Carlson, Scott Gaffney, and Flavian Vasile. 2009. Learning a named entity tagger from gazetteers with the partial perceptron. In *Proceedings of AAAI Spring Symposium: Learning by Reading and Learning to Read*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Eraldo R Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Proceedings of ECML-KDD*.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *Proceedings of EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of EMNLP*.

Xinghua Lou, Fred A Hamprecht, and IWR HCI. 2012. Structured learning from partial annotations. In *Proceedings of ICML*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL*.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2005. Conditional random fields for object recognition. In *Proceedings of NIPS*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Proceedings of Third Workshop on Very Large Corpora*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of EMNLP*.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of NIPS Workshop on Cost-Sensitive Learning*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.

Mihai Surdeanu, Ramesh Nallapati, and Christopher Manning. 2010. Legal claim identification: Information extraction with hierarchically labeled data. In *Proceedings of LREC*.

Erik F Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6:1453–1484.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of COLING*.

Fan Yang and Paul Vozila. 2014. Semi-supervised chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of EMNLP*.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *Proceedings of COLING*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of ICML*.

Event Detection without Triggers

Shulin Liu, Yang Li, Xinpeng Zhou, Tao Yang, Feng Zhang
Tencent AI Lab
{forestliu, youngyli, xinpengzhou}@tencent.com
{rigorosyang, jayzhang}@tencent.com

Abstract

The goal of event detection (ED) is to detect the occurrences of events and categorize them. Previous work solved this task by recognizing and classifying event triggers, which is defined as the word or phrase that *most clearly* expresses an event occurrence. As a consequence, existing approaches required both annotated triggers and event types in training data. However, triggers are nonessential to event detection, and it is time-consuming for annotators to pick out the "*most clearly*" word from a given sentence, especially from a long sentence. The expensive annotation of training corpus limits the application of existing approaches. To reduce manual effort, we explore detecting events without triggers. In this work, we propose a novel framework dubbed as **T**ype-aware **B**ias **N**eural **N**etwork with **A**ttention **M**echanisms (TBNNAM), which encodes the representation of a sentence based on target event types. Experimental results demonstrate the effectiveness. Remarkably, the proposed approach even achieves competitive performances compared with state-of-the-arts that used annotated triggers.

1 Introduction

This work tackles the task of event detection (ED), whose goal is to detect the occurrences of predefined events and categorize them. For example, consider the following sentence "*In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel.*", an ideal event detection system should recognize two events, *Death* and *Attack*(suppose that both *Death* and *Attack* are in the predefined event set) .

Previous work typically solved this task by recognizing and classifying event triggers. According to ACE (Automatic Context Extraction) event evaluation program, event trigger is defined as the word or phrase that *most clearly* expresses an event occurrence. Take the following sentence as an example:

> ***S*:** In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine Hotel.

"*died*" is the trigger word of *Death* event, and "*fired*" is the trigger word of *Attack* event. The majority of existing approaches modeled this task as word classification (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013; Nguyen and Grishman, 2015; Liu et al., 2016b,a; Chen et al., 2017), which predicted whether each word in a given sentence is an event trigger and what type of event it triggered. As a consequence, these approaches required both annotated triggers and event types for training.

However, event triggers are nonessential to this task. Remind that the goal of event detection is to recognize and categorize events, thus triggers could be viewed as intermediate results of this task. Furthermore, it is time-consuming for annotators to pick out the "*most clearly*" word from a given sentence, especially from a long sentence, which limits the application of existing ED approaches. To reduce manual effort, we explore detecting events without triggers. In this study, the only annotated information of each sentence is the types of events occurred in it. Consider the aforementioned example ***S*** again, its annotation is {*Death, Attack*}. On the contrast, previous work also required an annotated trigger for each event, which means the annotated information of ***S*** is {*Death*:died, *Attack*:fired} in previous work.

Without event triggers, it is intuitive to model this task via text classification. However, there are two challenges: (1) **Multi-label problem**: each sentence may contain arbitrary number of events, which means it could have zero or multiple target

Proceedings of NAACL-HLT 2019, pages 735–744
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

labels. In machine learning, this problem is called multi-label problem. (2) **Trigger absence problem**: previous work illustrated that trigger words play important roles in event detection(Chen et al., 2015; Liu et al., 2016a). It is challenging to model this information without annotated triggers.

To solve the first challenge, we transform multi-label classification to multiple binary classification problems. Specifically, a given sentence s attached each pre-defined event type t forms an instance, which is expected to be labeled with 0 or 1 according to whether s contains an event of type t. For example, suppose there totally are 3 predefined types of events(denoted by t_1, t_2 and t_3), and sentence s contains two events of type t_1 and t_3, then it could be transformed to the following three instances:

instance	label
$< s, t_1 >$	1
$< s, t_2 >$	0
$< s, t_3 >$	1

Table 1: Example of instances in binary classifications for sentence s, which contains events of type t_1 and t_3.

In this paradigm, sentences that convey multiple events will yield multiple positive pairs, thus the multi-label problem could be well solved.

Furthermore, each type of events are usually triggered by a set of specific words, which are called event trigger words. For example, *Death* events are usually triggered by “die”, “passed away”, ”gone”, etc. Therefore, event trigger words are important clues to this task. Since existing work explicitly exploited annotated trigger words in their approaches, they can directly model this observation. However, in our case, annotated triggers are unavailable. To model this information, we propose a simple but effective model, called **T**ype-aware **B**ias **N**eural **N**etwork with **A**ttention **M**echanisms (*TBNNAM*).

Figure 1 illustrates the framework of *TBNNAM*. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0 (the output should be 1 for the example given in Figure 1). Specifically, given a sentence, the proposed model first transforms the input tokens into embeddings, and applies an LSTM layer to calculate a context-dependent representation for each token. Then it computes an attention vector, α, based on the target event type, where the trigger word is expected to obtain higher score. Finally, the sentence representation s_{att} is calculated based on α. Here, s_{att} is expected to focus on local information (trigger word). To capture global information, the final output, o, is also connected to the last LSTM units, which encodes the global information of the input sentence. Furthermore, to reinforce the influence of positive samples, we devise a bias objective function in our model . We call our model “type-aware” because the representation of a sentence, s_{att}, is calculated based on the target event type.

We have conducted experimental comparisons on a widely used benchmark dataset ACE2005[1]. The results illustrate that our approach outperforms all the compared baselines, and even achieves competitive performances compared with exiting approaches that used annotated triggers. We publish our code for further study by the NLP community.[2]

In summary, the main contributions of this work are: (1) To the best of our knowledge, this is the first work that focuses on detecting events without triggers. Compared with existing approaches, the proposed method requires less manual annotations. (2) Without triggers, this task encounters two challenges: multi-label problem and trigger absence problem. We propose a simple but effective model, which even achieves competitive results compared with approaches that using annotated triggers. (3) Since this is the first work on detecting events without triggers, we implement a series of baseline models for this task, and systematically evaluate and analyze them.

2 Background

2.1 Task Definition

Event detection task requires that certain specified types of events, which are mentioned in the annotated data, to be detected. The most common used benchmark dataset in previous work is ACE 2005 corpus. This corpus includes 8 types of events, with 33 subtypes. Following previous work(Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al.,

---

[1] https://catalog.ldc.upenn.edu/LDC2006T06

[2] https://github.com/liushulinle/event_detection_without_triggers

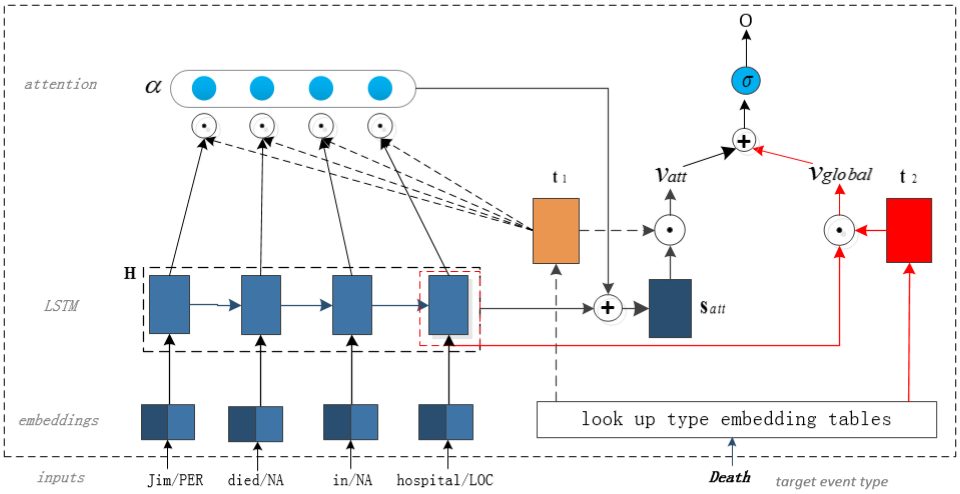


Figure 1: The framework of type-aware bias neural network with attention mechanisms. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. t_1 and t_2 are two different embedding vectors of the target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0 (the output should be 1 for the case in this figure). We call it "type-aware" because the representation of sentence, s_{att}, is calculated based on the target event type.

2013; Chen et al., 2015; Nguyen and Grishman, 2016), we treat them simply as 33 separate event types and ignore the hierarchical structure among them. Consider the following sentence "*In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel*", an ideal event detector should detect two events from this sentence: a *Die* event and an *Attack* event.

2.2 Related Work

Event detection is one of important topics in NLP. Many approaches have been proposed for this task. Nearly all the existing methods on ACE event task follow supervised paradigm. We further divide them into feature-based methods and representation-based methods.

In feature-based methods, a diverse set of strategies has been exploited to convert classification clues into feature vectors. Ahn (2006) uses the lexical features(e.g., full word), syntactic features (e.g., dependency features) and external-knowledge features(WordNet (Miller, 1998)) to extract the event. Inspired by the hypothesis of One Sense Per Discourse (Yarowsky, 1995), Ji and Grishman (2008) combined global evidence from related documents with local decisions for the event extraction. To capture more clues from the texts, Gupta and Ji (2009), Liao and Grishman (2010) and Hong et al. (2011) proposed the cross-event and cross-entity inference for the ACE event task. Li et al. (2013) proposed a joint model to capture the combinational features of triggers and arguments. Liu et al. (2016b) proposed a global inference approach to employ both latent local and global information for event detection.

In recent years, representation-based methods have dominated the research. In this paradigm, candidate event mentions are represented by embeddings, which typically are fed into neural networks. Chen et al. (2015) and Nguyen and Grishman (2015) are the first work in this paradigm. Their models are based on CNNs (Convolutional Neural Networks). To model the dependency of triggers and arguments, Nguyen and Grishman (2016) proposed a joint event extraction approach based RNNs(Recurrent Neural Networks). Liu et al. (2017) proposed to encode argument information in event detection via supervised attention mechanisms. Recently, Nguyen and Grishman (2018) and Sha et al. (2018) proposed to exploit syntactic information for event detection.

All the existing approaches required annotated triggers. The expensive annotation of training data limits the application of these approaches. To reduce manual effort, we perform this task without event triggers.

3 Methodology

To deal the multi-label problem, we model this task via multiple binary classifications. Given a sentence, it will be fed into a binary classifier with each candidate event type. We add the label *NA* to sentences that do not contain any events. To capture the hidden trigger information, we propose a simple but effective model, called **T**ype-aware **B**ias **N**eural **N**etwork with **A**ttention **M**echanisms (*TBNNAM*). Our model is "type-aware" because it calculates the representation of a sentence based on the target event type. Figure 1 illustrates the framework of *TBNNAM*. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0. Next, we describe the structure of this model in bottom-up order.

3.1 Input Tokens

Given a sentence, we use Stanford CoreNLP tools[3](Manning et al., 2014) to convert texts into tokens. The ACE 2005 corpus annotated not only events but also entities for each given sentence. Following previous work, we exploit the annotated entity tags in our model(Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2015, 2016; Liu et al., 2016b).

3.2 Word/Entity Embeddings

Word embeddings learned from a large amount of unlabeled data have been shown to be able to capture the meaningful semantic regularities of words(Bengio et al., 2003; Erhan et al., 2010). Much work(Socher et al., 2012; Zeng et al., 2014) has shown its power in many NLP tasks.

In this work, we use the Skip-gram model(Mikolov et al., 2013) to learn word embeddings on the NYT corpus[4]. Furthermore, we randomly initialized an embedding table for each entity tags. All the input word tokens and entity tags will be transformed into low-dimensional vectors by looking up these embedding tables. In this work, we denote the dimension of word embeddings by d_w, and that of entity embeddings by d_e.

3.3 Event Type Embeddings

As illustrated in Figure 1, an event type is transformed into two embedding vectors: $\mathbf{t}_1$ and $\mathbf{t}_2$. The first one (colored with brown) is designed to capture local information (hidden trigger word), and the latter one (colored with red) is designed to capture global information. Both of them are randomly initialized. The dimension of event type embeddings is denoted by d_{evt}.

3.4 LSTM Layer

As shown in Figure 1, the LSTM layer is run over the sequence of concatenation of word and entity embeddings. LSTM has three gates(input i, forget f and output o), and a cell memory vector c. The input gate can determine how incoming vectors $\mathrm{x(t)}$ alter the state of the memory cell. The output gate can allow the memory cell to have an effect on the outputs. Finally, the forget gate allows the cell to remember or forget its previous state.

3.5 Attention Layer

Each type of events are usually triggered by a set of specific words, which are called event trigger words. For example, *Death* events are usually triggered by "die", "passed away", "gone", etc. Therefore, event trigger words are important clues to this task. However, this information is hidden in our task, because annotated triggers are unavailable. To model the hidden triggers, we introduce attention mechanisms in our approach.

As illustrated in Figure 1, the attention vector α is calculated based on the target event type embedding $\mathbf{t}_1$ and the hidden states h yielded by LSTM. Specifically, the attention score for the k-th token in a given sentence is calculated by the following equation:

$$\alpha^k = \frac{\exp(\mathbf{h}_k \cdot \mathbf{t}_1^T)}{\sum_i \exp(\mathbf{h}_i \cdot \mathbf{t}_1^T)} \tag{1}$$

In this model, trigger words of the target event type are expected to obtain higher scores than other words. Finally, the representation of the sentence, $\mathbf{s}_{att}$, is computed by the following equation:

$$\mathbf{s}_{att} = \alpha^T \mathbf{H} \tag{2}$$

where $\alpha = [\alpha_1, ..., \alpha_n]$ is the attention vector, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]$ is a matrix, $\mathbf{h}_k$ is the LSTM's output for the k-th token, and $\mathbf{s}_{att}$ is the representation of the given sentence.

3.6 Output Layer

As illustrated in Figure 1, the final output o is connected to two components: v_{att} and v_{global}. On one hand, v_{att} is calculated by the dot product of $\mathbf{s}_{att}$ and $\mathbf{t}_1$, which is designed to capture local features (specifically, features about hidden trigger words). On the other hand, the last output of the LSTM layer, $\mathbf{h}_n$, encodes global information of the whole sentence, thus $v_{global} = \mathbf{h}_n \cdot \mathbf{t}_2^T$ is expected

---

[3] http://stanfordnlp.github.io/CoreNLP

[4] https://catalog.ldc.upenn.edu/LDC2008T19

to capture global features of a sentence. Finally, o is defined as the weighted sum of v_{att} and v_{global}:

$$o = \sigma(\lambda \cdot v_{att} + (1 - \lambda) \cdot v_{global}) \quad (3)$$

where σ is the Sigmoid function, $\lambda \in [0, 1]$ is a hyper-parameter for trade-off between v_{att} and v_{global}.

3.7 Bias Loss Function

We devise a bias loss function to reinforce the influence of positive samples because of the following reasons. 1) **positive samples are much less than negative samples.** In our approach, each training sample is a <sentence,event type> pair, whose label is 1 or 0 according to whether the given sentence conveys an event of type t. For example, we totally have 33 target event types, if a sentence only contains one event, then it will be transformed into 32 negative pairs and 1 positive pair. The majority of sentences convey at most two events, thus negative samples are much more than positive samples. 2) **positive samples are more informative than negatives.** A positive pair $< s, t >$ means that s conveys an event of type t, whereas negative pair means s does not convey any event of type t. Apparently, the former is more informative.

Given all of the (suppose T) training instances $(x^{(i)}, y^{(i)})$, the loss function is defined as follows:

$$J(\theta) = \frac{1}{T}\sum_{i=1}^{T}(o(x^i) - y^{(i)})^2(1 + y^{(i)} \cdot \beta) + \delta||\theta||^2 \quad (4)$$

where x is a pair consisted of a sentence and a target event type, $y \in \{0, 1\}$, θ is the parameter of our model and $\delta > 0$ is the weight of L2 normalization term. $(1 + y^{(i)} \cdot \beta)$ is the bias term. Specifically, the value of this term is 1 for negative samples ($y^{(i)}$ is 0) and 1 + β for positive samples ($y^{(i)}$ is 1), where $\beta \geq 0$.

3.8 Training

We train the model by using a simple optimization technique called stochastic gradient descent (SGD) over shuffled mini-batches with the Adadelta rule (Zeiler, 2012). Regularization is implemented by a dropout and L2 norm.

Given a instance x, the model assign it a label $\widetilde{y}$ according to the following equation:

$$\widetilde{y} = \begin{cases} 0 & o(x) < 0.5 \\ 1 & otherwise \end{cases} \quad (5)$$

where x is a pair $< s, t >$, $o(x)$ is the output of the model for x, and $\widetilde{y}$ is the final predicted result.

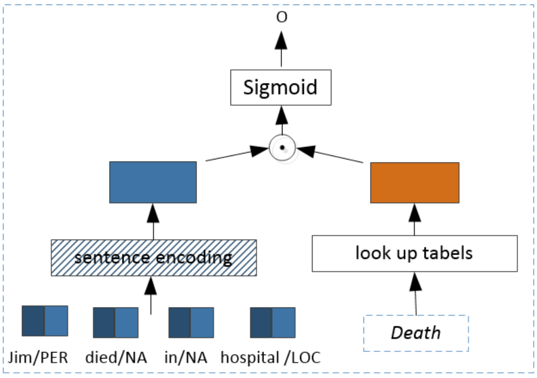


Figure 2: The framework of binary classification based approaches. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0.

4 Baseline Systems

Since this is the first work to perform event detection without triggers, we implement a series of baseline systems for comparisons, which could be divided into two categories: binary classification based methods and multi-class classification based methods.

4.1 Binary Classification

Similar with the proposed approach, baseline systems in this group solved this task via binary classification. Figure 2 illustrates the framework of these methods. These models take a sentence and a target event type as input. Then all the inputs are transformed into embeddings by looking up embedding tables. These models have the same loss function as the proposed approach (see Equation 4). The key component of these models is sentence encoder. According to the strategy of encoding sentence, we implement three models for comparison: *BC-CNN*, *BC-LSTM*$_{last}$, *BC-LSTM*$_{avg}$.

- *BC-CNN* employs a CNN model to encode sentence.
- *BC-LSTM*$_{last}$ employs LSTM model, and use the hidden state of the last token as the representation of a given sentence.
- *BC-LSTM*$_{avg}$ also employs LSTM model, but use the average of all hidden states as the representation of a given sentence.

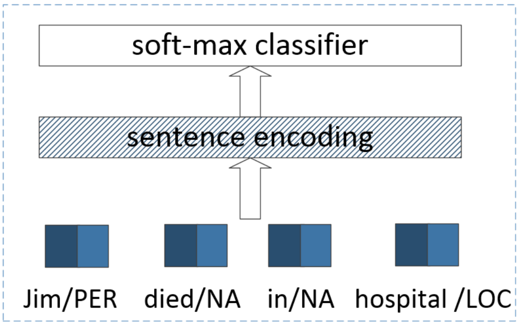


Figure 3: The framework of multi-class classification based approaches.

4.2 Multi-class Classification

All existing approaches model the task of event detection (with triggers) via multi-class classification[5]. Given a sentence, these methods predict whether each token is an event trigger and what type of event it triggered. We also implement several multi-class classification based systems for comparison. Since annotated triggers are unavailable in our task, the sentence is the input of our model. Figure 3 illustrates the framework of these models. Following existing work(Chen et al., 2015; Liu et al., 2017), we employ a negative log-likelihood loss function in the soft-max classifier: $J(\theta) = -\frac{1}{T}\sum_{i=1}^{T}\log(p(y^{(i)}|x^{(i)},\theta))$, where $(x^{(i)}, y^{(i)})$ is a training sample, $y^{(i)}$ is a label from the valid label set (all the predefined event types plus a *NA* for none event), T is the total number of training instances, θ is the parameters of the model. According to the strategy of encoding sentence, we implement three models: *MC-CNN*, $MC\text{-}LSTM_{last}$ and $MC\text{-}LSTM_{avg}$.

- *MC-CNN* employs a CNN model to encode sentence.
- $MC\text{-}LSTM_{last}$ employs LSTM model, and use the hidden state of the last token as the representation of a given sentence.
- $MC\text{-}LSTM_{avg}$ also employs LSTM model, but use the average of all hidden states as the representation of a given sentence.

5 Experimental Results

5.1 Experimental Setup

In this section, we introduce the dataset, evaluation metrics and the settings of hyper parameters.

5.1.1 Dataset

Our experiments are conducted on ACE 2005 dataset. Following the evaluation of previous work(Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2016; Liu et al., 2017), we randomly selected 30 articles from different genres as the development set, and subsequently conducted a blind test on a separate set of 40 ACE 2005 newswire documents. We used the remaining 529 articles as our training set.

This work focuses on detecting events without triggers. Therefore, we remove trigger annotations from the corpus. Specifically, we employ Stanford CoreNLP Toolkit to split each document into sentences, and assign each sentence with a set of labels according to the original annotations in ACE 2005 corpus. If a sentence does not contain any event, we assign it with a special label, *NA*. If a sentence contains multiple events of the same type (less than 3% in ACE corpus), we only keep one label for each type. Table 2 shows several samples of the our corpus.

sentence	labels
They got married in 1985.	{*Marry*}
They got married in 1985, and divorced 3 years latter.	{*Marry*, *Divorce*}
They are very happy every day.	{*NA*}

Table 2: Examples of instances in our corpus (without event trigger annotations).

5.1.2 Evaluation Metrics

Following previous work (Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2017), we use precision (P), recall (R) and F_1-measure (F_1) to evaluate the results.

Precision: the proportion of correctly predicted events in total predicted events.

Recall: the proportion of correctly predicted events in total gold events of the dataset.

F_1-measure: $\frac{2\times P\times R}{P+R}$

5.1.3 Hyper Parameters

Hyper parameters are tuned on the development dataset via grid search. In all experiments, we

---

[5] Multi-class classification means a classification task with more than two classes, but each sample belongs to only one class. "multi-class" is different from "multi-label".

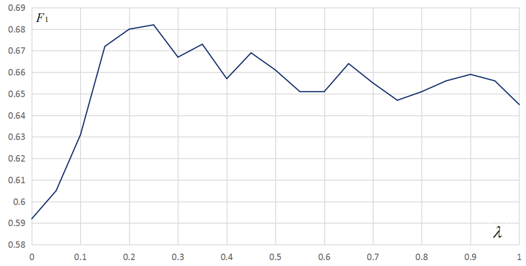


Figure 4: Experimental results on development dataset with different setting of λ.

methods	**P(%)**	**R(%)**	**F_1(%)**
MC-CNN	73.3	46.3	56.8
BC-CNN	**76.6**	**52.9**	**62.6**
MC-LSTM$_{last}$	57.9	42.3	48.9
BC-LSTM$_{last}$	**69.8**	**52.2**	**59.7**
MC-LSTM$_{avg}$	60.3	42.7	50.0
BC-LSTM$_{avg}$	**68.1**	**49.2**	**57.1**

Table 3: Experimental results on ACE 2005 corpus. Methods with name *MC-** are based on multi-class classification, and methods with name *BC-** are based on binary classification.

set the dimension of word embeddings as 200, the dimension of entity type embeddings as 50, batch size as 100, the hyper parameter for the L_2 norm as 10^{-5}, β in the bias term as 1.0. Furthermore, we also tune λ in Equation 3 on the development dataset. Figure 4 illustrates experimental results with different settings of λ, finally we set λ as 0.25. And in all the CNN-based baseline systems, the sizes of filter windows are set to 1, 2, 3 with 100 feature maps each.

5.2 Multi-class Classification vs. Binary Classification

Table 3 illustrates the experimental results, where methods with name *MC-** are based on multi-class classification, and methods with name *BC-** are based on binary classification. According to the strategy to encode a sentence, methods in Table 3 are grouped into three parts. From the table, we make the following observations:

- In each group, binary classification based approach significantly outperforms multi-class classification based approach. The reason is that *BC-** can solve the multi-label problem, but *MC-** can not. Moreover, *MC-** achieve much lower recall than *BC-**, because they predict at most one event for each sentence.

methods	**P(%)**	**R(%)**	**F_1(%)**
BC-CNN	76.6	52.9	62.6
BC-LSTM$_{last}$	69.8	52.2	59.7
BC-LSTM$_{avg}$	68.1	49.2	57.1
BC-LSTM$_{att}$	68.3	64.5	66.3
our TBNNAM	76.2	64.5	**69.9**
Nguyen's CNN†	71.8	66.4	69.0
Chen's DMCNN†	75.6	63.6	69.1
Liu's PSL†	75.3	64.4	69.4
DS-DMCNN‡†	75.7	66.0	70.5

Table 4: Experimental results on ACE 2005 corpus. Methods in the first group are baseline systems. Methods in the second group are the proposed approaches. Methods in the last group are state-of-the-art ED systems. † requiring annotated triggers, ‡ using external data

- Methods with CNN as sentence encoder achieve better performance than that with LSTM. The reason is that trigger words are important clues to event detection, and CNN is good at extracting such local features.

5.3 Overall Performances

In this section, we illustrates the results of the proposed approach (see Table 4). The results of baseline systems are listed in the first group. Methods in the second group are the proposed approaches. They have the same model structure as Figure 1. In *BC-LSTM*$_{att}$, λ (see Equation 3) is set as 1.0, which is designed to show the effects of the proposed attention strategy. In *TBNNAM*, λ is set as 0.25, which is designed to employ both local information (captured by the attention mechanism) and global information (captured by the last output of LSTM). Methods in the last group are state-of-the-art ED systems on ACE 2005 dataset. We give a brief introduction of them as follows:

1). *Nguyen's CNN*: the CNN model proposed by Nguyen and Grishman (2015)

2). *Chen's DMCNN*: the dynamic multi-pooling CNN model proposed by Chen et al. (2015)

3). *Liu's PSL*: the soft probabilistic soft logic model proposed by Liu et al. (2016b)

4). *DS-DMCNN*: the DMCNN model augmented with automatic labeled data, proposed by Chen et al. (2017)

From the table, we make the following observations:

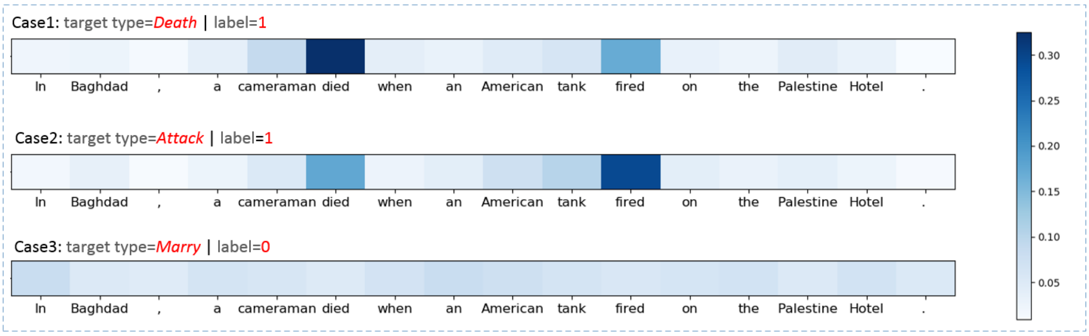


Figure 5: Visualization of attention weight vector α of sample instances learned by our model.

- *BC-LSTM*$_{att}$ outperforms all the baseline systems with remarkable gains, which demonstrates the effectiveness of the proposed attention mechanism.

- *TBNNAM* achieves better performance than *BC-LSTM*$_{att}$ (69.9% vs. 66.3%), which means that global information captured by the last state of LSTM is also important to this task. Such global information and local information captured by the attention mechanisms are complementary to each other.

- All state-of-the-art ED systems require annotated triggers. Without trigger annotations, our approach achieves competitive results, even outperforms some of them.

5.4 Analysis of Weight α

Figure 5 shows several examples of the attention vector α learned by our model. In the first case, “died” is the most significant keyword for the *Death* event, and our model succeeded to capture this feature by assigning it with a large attention score. Similarly, in the second case, “fired” is a key clue of *Attack* event, and our model also learned it and assigned it with a large attention score. Actually, “died” and “fired” are the trigger words of *Death* and *Attack* events, respectively. Therefore, we argue that, although annotated triggers are unavailable, our model still can exploit trigger information for this task. Moreover, our approach also could model the dependencies among different events, which has been demonstrated useful for this task(Liao and Grishman, 2010; Liu et al., 2016b). For example, *Attack* events often co-occur with *Death* events. In Case1 and Case2 (Figure 5), our approach models such information by paying attention on both words “died” and “fired”. Furthermore, the 3-rd case is a negative sample, thus there is no key clues. Our model assigned each token with nearly equivalent attention score.

methods	**P(%)**	**R(%)**	**F$_1$(%)**
BC-LSTM$_{att}\setminus$*Bias*	**74.5**	57.2	64.7
BC-LSTM$_{att}$	68.3	**64.5**	**66.3**
TBNNAM$\setminus$*Bias*	**76.6**	59.8	67.2
TBNNAM	76.2	**64.5**	**69.9**

Table 5: Results of systems without/with bias term in loss function, where *$\setminus$*Bias* do not use bias term.

5.5 Effects of Bias Term in Loss Function

In this section, we illustrates the effectiveness of the bias term in Equation 4. Table 5 shows experimental results. Methods named with “*$\setminus$*Bias*” do not use bias term. From the table, we observe that systems with bias term in loss function significantly outperform those without bias term, which demonstrates the correctness of our analysis in Section 3.7 that positive samples should be reinforced during training.

6 Conclusions

Existing event detection approaches required annotated triggers, which limits their applications because of the expensive annotations. To reduce manual effort, we investigate performing this task without event triggers. In this setting, event detection task encounters two challenges: multi-label problem and trigger absence problem. We propose a simple but effective model to solve them, which computes the representation of a sentence according to the target event type. Experimental results demonstrate its effectiveness. Remarkably, the proposed approach even achieves competitive performances compared with state-of-the-arts that used annotated triggers.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419, Vancouver, Canada.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 167–176. Association for Computational Linguistics.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 369–372. Association for Computational Linguistics.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2134–2143. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the thirtieth AAAI Conference on Artificail Intelligence*, pages 2993–2999. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Huu Thien Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 365–371. Association for Computational Linguistics.

Huu Thien Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing*, pages 1201–1211. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014*, pages 2335–2344.

Sub-Event Detection from Twitter Streams as a Sequence Labeling Problem

Giannis Bekoulis **Johannes Deleu** **Thomas Demeester** **Chris Develder**
Ghent University - imec, IDLab
Department of Information Technology
`firstname.lastname@ugent.be`

Abstract

This paper introduces improved methods for sub-event detection in social media streams, by applying neural sequence models not only on the level of individual posts, but also directly on the stream level. Current approaches to identify sub-events within a given event, such as a goal during a soccer match, essentially do not exploit the sequential nature of social media streams. We address this shortcoming by framing the sub-event detection problem in social media streams as a sequence labeling task and adopt a neural sequence architecture that explicitly accounts for the chronological order of posts. Specifically, we (i) establish a neural baseline that outperforms a graph-based state-of-the-art method for binary sub-event detection (2.7% micro-F_1 improvement), as well as (ii) demonstrate superiority of a recurrent neural network model on the posts sequence level for labeled sub-events (2.4% bin-level F_1 improvement over non-sequential models).

1 Introduction

Social media allow users to communicate via real-time postings and interactions, with Twitter as a notable example. Twitter user posts, i.e., tweets, are often related to events. These can be social events (concerts, research conferences, sports events, etc.), emergency situations (e.g., terrorist attacks) (Castillo, 2016), etc. For a single event, multiple tweets are posted, by people with various personalities and social behavior. Hence, even more so than (typically more neutral) traditional media, this implies many different perspectives, offering an interesting aggregated description.

Given this continuous and large stream of (likely duplicated) information in Twitter streams, and their noisy nature, it is challenging to keep track of the main parts of an event, such as a soccer match. Automating such extraction of different sub-events within an evolving event is known as sub-event detection (Nichols et al., 2012). For tracking each of the sub-events, the timing aspect is an important concept (i.e., consecutive tweets in time). Thus, a sequential model could successfully exploit chronological relations between the tweets in a Twitter stream as an informative feature for sub-event detection.

Several methods have been proposed for sub-event detection: clustering methods (Pohl et al., 2012), graph-based approaches (Meladianos et al., 2015), topic models (Xing et al., 2016) and neural network architectures (Wang and Zhang, 2017). None of these studies exploits the chronological relation between consecutive tweets. In contrast, our work does take into account that chronological order and we predict the presence and the type of a sub-event exploiting information from previous tweets. Specifically, we (i) propose a new neural baseline model that outperforms the state-of-the-art performance on the binary classification problem of detecting the presence/absence of sub-events in a sports stream, (ii) establish a new reasonable baseline for predicting also the sub-event *types*, (iii) explicitly take into account chronological information, i.e., the relation among consecutive tweets, by framing sub-event detection as a sequence labeling problem on top of our baseline model, and (iv) perform an experimental study, indicating the benefit of sequence labeling for sub-event detection in sports Twitter streams.

2 Related Work

Twitter streams have been extensively studied in various contexts, such as sentiment analysis (Kouloumpis et al., 2011), stock market prediction (Nguyen and Shirai, 2015) and traffic detection (D'Andrea et al., 2015). Specifically, for sub-event detection in Twitter, several approaches

Proceedings of NAACL-HLT 2019, pages 745–750
Minneapolis, Minnesota, June 2 - June 7, 2019.

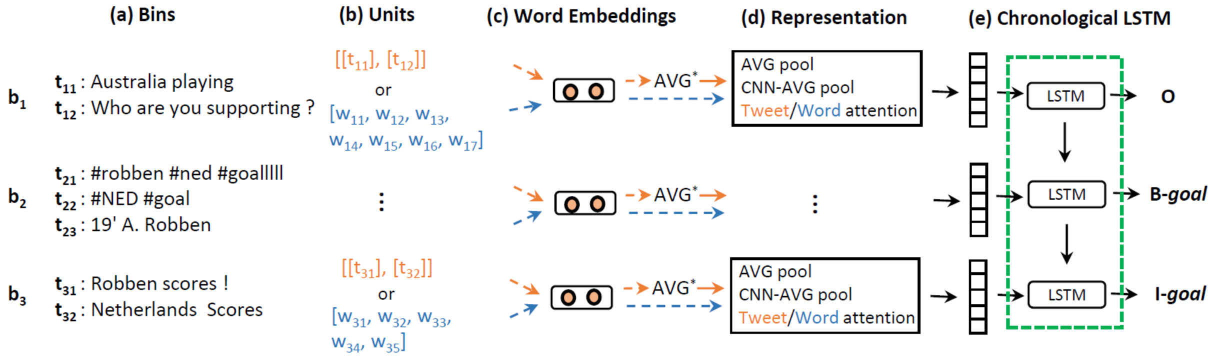


Figure 1: Our sub-event detection model comprises: (a) a bin layer, (b) a unit layer, (c) a word embeddings layer, (d) a representation layer and (e) a **chronological LSTM** layer to model the natural flow of the sub-events within the event. We represent each bin using either (i) a **tweet**- or (ii) a **word**-level representation. The AVG* represents an average pool operation, performed either directly on the embeddings or on the tweet's LSTM representation.

have been tried. *Unsupervised methods* such as clustering aim to group similar tweets to detect specific sub-events (Pohl et al., 2012; Abhik and Toshniwal, 2013) and use simple representations such as tf-idf weighting combined with a similarity measure. Other unsupervised algorithms use topic modeling approaches, based on assumptions about the tweets' generation process (Xing et al., 2016; Srijith et al., 2017). Several methods (Zhao et al., 2011; Zubiaga et al., 2012; Nichols et al., 2012) assume that a sub-event happens when there is a 'burst', i.e., a sudden increase in the rate of tweets on the considered event, with many people commenting on it. Recently, neural network methods have used more complicated representations (Wang and Zhang, 2017; Chen et al., 2018). Also *supervised methods* have been applied (Sakaki et al., 2010; Meladianos et al., 2018) for the sub-event detection task. These methods usually exploit graph-based structures or tf-idf weighting schemes. We believe to be the first to (i) exploit the chronological order of the Twitter stream and take into account its sequential nature, and (ii) frame the sub-event detection problem as a sequence labeling task.

3 Model

3.1 Task Definition

The goal is, given a main event (i.e., soccer match), to identify its core sub-events (e.g., goals, kick-off, yellow cards) from Twitter streams. Specifically, we consider a *supervised* setting, relying on annotated data (Meladianos et al., 2018).

3.2 Word- vs Tweet-Level Representations

Similar to previous works, we split a data stream into time periods (Meladianos et al., 2018): we form bins of tweets posted during consecutive time intervals. E.g., for a soccer game, one-minute intervals (bins) lead to more than 90 bins, depending on the content before and after the game, halftime, stoppage time, and possibly some pre-game and post-game buffer. Thus, for each bin, we predict either the presence/absence of a sub-event (Section 3.3) or the most probable sub-event type (Section 3.4), depending on the evaluation scenario.

We consider representing the content of each bin either on (i) a word-level or (ii) a tweet-level (see Fig. 1). Formally, we assume that we have a set of n bins $b_1, ..., b_n$, where each bin b_i consists of m_i tweets and k_i words (i.e., all words of tweets in bin b_i). Then, the *tweet-level representation* of bin b_i is symbolized as $t_{i1}, ..., t_{im_i}$, where t_{im_i} is the m_i^{th} tweet of bin b_i. In the *word-level representation*, we chronologically concatenate the words from the tweets in the bin: $w_{i1}, .., w_{ik_i}$, where w_{ik_i} is the k_i^{th} word of bin b_i.

3.3 Binary Classification Baseline

To compare with previous work (Meladianos et al., 2018), we establish a simple baseline for binary classification: presence/absence of a sub-event. For this case, we use as input the word-level representation of each bin. To do so, we use word embeddings (randomly initialized) with average (AVG) pooling (Iyyer et al., 2015) in combination with a multilayer perceptron (MLP) for binary classification, i.e., presence/absence of a sub-event. Note that we experimented with pre-trained embeddings as well as max-pooling, but those early experiments led to performance decrease compared to the presented baseline model. We found that training based on average bin representations works substantially better than with max-pooling, and we hypothesize that this is related to the noisy nature of the Twitter stream.

3.4 Sequence Labeling Approach

Building on the baseline above, we establish a new architecture that is able to capture the sub-event types as well as their duration. We phrase sub-event detection in Twitter streams as a sequence labeling problem. This means we assume that the label of a bin is not independent of neighboring bin labels, given the chronological order of bins of the Twitter stream, as opposed to independent prediction for each bin in the binary classification baseline above. For instance, when a *goal* is predicted as a label for bin b_i, then it is probable that the label of the next bin b_{i+1} will also be *goal*. Although a sub-event may occur instantly, an identified sub-event in a Twitter stream can span consecutive bins, i.e., minutes: users may continue tweeting on a particular sub-event for relatively long time intervals. For this reason, we apply the well-known BIO tagging scheme (Ramshaw and Marcus, 1995) for the sub-event detection problem. For example, the beginning of a *goal* sub-event is defined as B-*goal*, while I-*goal* (inside) is assigned to every consecutive bin within the same sub-event, and the O tag (outside) to every bin that is not part of any sub-event. To propagate chronological information among bins, we adopt an LSTM on the sequence of bins as illustrated in Fig. 1, layer (e). Note that this tagging approach assumes that sub-events do not overlap in time, i.e., only at most one is ongoing in the Twitter stream at any point in time.

4 Experimental Setup

We evaluated our system[1] on the dataset from Meladianos et al. (2018), with tweets on 20 soccer matches from the 2010 and 2014 FIFA World Cups, totalling over 2M pre-processed tweets filtered from 6.1M collected ones, comprising 185 events. The dataset includes a set of sub-events, such as *goal*, *kick-off*, *half-time*, etc. To compare our binary classification *baseline system* to previous methods (Table 1), we use the same train/test splits as Meladianos et al. (2018), where 3 matches are used for training and 17 matches as test set. In this setting, we predict only the presence/absence of a sub-event. Similar to previous work, we count a sub-event as correct if at least one of its comprising bins has been classified as a sub-event. For the experimental study of our proposed *sequence labeling approach* for sub-event detection, where sub-event types are predicted, we have randomly split the test set into test (10 matches) and development (7 matches) sets. We use the development set to optimize the F_1 score for tuning of the model parameters, i.e., the word/tweet embedding representation size, LSTM hidden state size, dropout probability. We adopt 2 evaluation strategies. The first one, referred to as *relaxed* evaluation, is commonly used in entity classification tasks (Adel and Schütze, 2017; Bekoulis et al., 2018a,c) and similar to the binary classification baseline system evaluation: score a multi-bin sub-event as correct if at least one of its comprising bin types (e.g., *goal*) is correct, assuming that the boundaries are given. The second evaluation strategy, *bin-level*, is stricter: we count each bin individually, and check whether its sub-event type has been predicted correctly, similar to the token-based evaluation followed in Bekoulis et al. (2018b).

5 Results

5.1 Baseline Results

Table 1 shows the experimental results of our baseline model. The Burst baseline system is based on the tweeting rate in a specific time window (i.e., bin) and if a threshold is exceed, the system identifies that a sub-event has occurred. We report evaluation scores as presented in Meladianos et al. (2018). The second approach is the graph-based method of Meladianos et al. (2018). We observe that our baseline system (Section 3.3) has a 1.2% improvement in terms of macro-F_1 and 2.7% improvement in terms of micro-F_1, compared to the graph-based model from Meladianos et al. (2018), mainly due to increased precision, and despite the recall loss.

5.2 Sequence Labeling Results

Table 2 illustrates the predictive performance of our proposed model (i.e., using the chronological

---

[1] `https://github.com/bekou/subevent_sequence_labeling`

LSTM) compared to models making independent predictions per bin. The upper part of Table 2 contains models without the chronological LSTM. Our experiments study both *word-level* and *tweet-level* bin representations (see Fig. 1), as reflected in the 'Word' vs. 'Tweet' prefix, respectively, in the Model column of Table 2.

The simplest *word-level* representation uses the tf-idf weighting scheme (as in Pohl et al. (2012)) followed by an MLP classifier. For the other word-level models, we exploit several architectures: AVG pooling (Iyyer et al., 2015), a CNN followed by AVG pooling (Kim, 2014) and hierarchical word-level attention (Yang et al., 2016).

For *tweet-level* representations, we adopt similar architectures, where the AVG, CNNs and attention are performed on sentence level rather than on the word-level representation of the bin. In this scenario, we have also exploited the usage of sequential LSTMs to represent the tweets. When comparing models with and without tweet-level LSTMs, we report the strategy that yields the best results, indicated by ✓ and ✗ in the tweet-level LSTM (TL) columns of Table 2. We do not present results for applying sequential LSTMs on the word-level bin representation, because of slow training on the long word sequences.

Benefit of Chronological LSTM: The bottom part of Table 2 presents the results of the same models followed by a chronological LSTM to capture the natural flow of the stream as illustrated in Fig. 1. We report results as described in Section 4, using the micro F_1 score with the two evaluation strategies (*bin-level* and *relaxed*). We observe that when using the chronological LSTM, the performance in terms of *bin-level* F_1 score is substantially improved for almost every model. Note that the best model using the chronological LSTM (Tweet-AVG) achieves 2.4% better F_1 than the best performing model without the use of chronological LSTM (Word-CNN-AVG). In most cases there is also a consistent improvement for both the precision and the recall metrics, which is thanks to the sequential nature of the upper level LSTM capturing the flow of the text.

Settings	Macro			Micro		
	P	R	F_1	P	R	F_1
Burst	78.00	54.00	64.00	72.00	54.00	62.00
Meladianos et al. (2018)	76.00	75.00	75.00	73.00	74.00	73.00
Our binary classif. baseline	89.70	69.99	**76.16**	83.65	69.05	**75.65**

Table 1: Comparing our neural network binary classification baseline model to state-of-the-art (P = precision, R = recall).

	Model	Bin-level				Relaxed			
		TL	P	R	F_1	TL	P	R	F_1
without chronol. LSTM	Word-tf-idf	-	49.40	52.06	50.69	-	56.10	56.10	56.10
	Word-AVG	-	51.40	45.96	48.53	-	56.10	56.10	56.10
	Word-CNN-AVG	-	56.93	56.01	56.47	-	75.60	75.60	75.60
	Word-attention	-	52.92	58.71	55.66	-	86.59	86.59	**86.59**
	Tweet-AVG	✓	49.04	45.96	47.45	✓	62.19	62.19	62.19
	Tweet-attention	✓	51.99	42.37	46.68	✗	80.48	80.48	80.48
	Tweet-CNN	✗	58.88	51.17	54.75	✗	70.73	70.73	70.73
with chronol. LSTM	Word-AVG	-	58.14	58.35	58.24	-	71.95	71.95	71.95
	Word-CNN-AVG	-	60.89	56.19	58.45	-	60.97	60.97	60.97
	Word-attention	-	52.99	42.90	47.42	-	60.97	60.97	60.97
	Tweet-AVG	✗	57.43	60.32	**58.84**	✗	64.63	64.63	64.63
	Tweet-attention	✓	48.26	52.24	50.17	✗	67.07	67.07	67.07
	Tweet-CNN	✗	65.33	49.73	56.47	✗	60.97	60.97	60.97

Table 2: Comparison of our baseline methods in terms of micro *bin-level* and *relaxed* F_1 score with and without chronological LSTM (see Fig. 1). The ✓and ✗ indicate whether the model uses a tweet-level LSTM (TL).

Limitations of *Relaxed* Evaluation: On the other hand, using the *relaxed* evaluation strategy, we observe that the best models are those without the chronological LSTM layer. Yet, we consider the *relaxed* evaluation strategy flawed for our scenario, despite the fact that it has been used for entity classification tasks (Bekoulis et al., 2018a; Adel and Schütze, 2017). Indeed, it is not able to properly capture sub-events which are characterized by duration: e.g., if a model assigns a different label to each of the bins that together constitute a single sub-event, then this sub-event counts as a true positive based on the *relaxed* evaluation strategy (similar to the evaluation proposed by Meladianos et al. (2018) and followed in Table 1). Thus, in this work, we propose to use the *bin-level* evaluation, since it is a more natural way to measure the duration of a sub-event in a supervised sequence labeling setting. Note that due to the noisy nature of Twitter streams, a tweet sequence spanning a particular sub-event is likely to contain also tweets that are not related to the given sub-event: a given bin inside the event may contain only a minority of tweets discussing the event. Therefore, we consider the standard sequence labeling evaluation (requiring to have types as well as boundaries correct) to be not applicable in sub-event detection.

Performance Comparison of the Top-3 Models: Figure 2 shows the performance of our three best performing models in terms of *bin-level* F_1 score on the validation set. The best performing model is the Tweet-AVG model since it attains its maximum performance even from the first training epochs. The Word-AVG model performs well

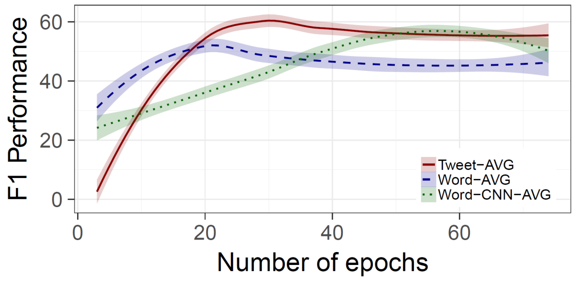


Figure 2: *Bin-level* F_1 performance of the three best performing models on the validation set with respect to the number of epochs. The smoothed lines (obtained by LOWESS smoothing) model the trends and the 95% confidence intervals.

from the first epochs, showing similar behavior to the Tweet-AVG model. This can be explained by the similar nature of the two models. The word-level CNN model attains maximum performance compared to the other two models in later epochs. Overall, we propose the use of the chronological LSTM with the Tweet-AVG model since this model does not rely on complex architectures and it gives consistent results.

6 Conclusion

In this work, we frame the problem of sub-event detection in Twitter streams as a sequence labeling task. Specifically, we (i) propose a binary classification baseline model that outperforms state-of-the-art approaches for sub-event detection (presence/absence), (ii) establish a strong baseline that additionally predicts sub-event *types*, and then (iii) extend this baseline model with the idea of exchanging chronological information between sequential posts, and (iv) prove it to be beneficial in almost all examined architectures.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback. Moreover, we would like to thank Christos Xypolopoulos and Giannis Nikolentzos for providing (i) the Twitter dataset (tweet ids) and (ii) instructions to reproduce the results of their graph-based approach.

References

Dhekar Abhik and Durga Toshniwal. 2013. Sub-event detection during natural hazards using features of social media data. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 783–788, New York, NY, USA. ACM.

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1723–1729. Association for Computational Linguistics.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018a. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2830–2836. Association for Computational Linguistics.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018b. An attentive neural architecture for joint segmentation and parsing and its application to real estate ads. *Expert Systems with Applications*, 102:100 – 112.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018c. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34 – 45.

Carlos Castillo. 2016. *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge University Press.

Guandan Chen, Nan Xu, and Weiji Mao. 2018. An encoder-memory-decoder framework for sub-event detection in social media. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1575–1578, New York, NY, USA. ACM.

Eleonora D'Andrea, Pietro Ducange, Beatrice Lazzerini, and Francesco Marcelloni. 2015. Real-time detection of traffic from Twitter stream analysis. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2269–2283.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the Fifth International AAAI conference on weblogs and social media*, pages 538–541.

Polykarpos Meladianos, Giannis Nikolentzos, Franois Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2015. Degeneracy-based real-time subevent detection in Twitter stream. In *Proceedings of the 9th International AAAI Conference on Web and Social Media*, pages 248–257. AAAI Press.

Polykarpos Meladianos, Christos Xypolopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. 2018. An optimization approach for sub-event detection and summarization in Twitter. In *Proceedings of the 40th European Conference in Information Retrieval*, pages 481–493. Springer International Publishing.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1354–1364. Association for Computational Linguistics.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using Twitter. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, pages 189–198, New York, NY, USA. ACM.

Daniela Pohl, Abdelhamid Bouchachia, and Hermann Hellwagner. 2012. Automatic sub-event detection in emergency management using social media. In *Proceedings of the 21st International Conference on World Wide Web*, pages 683–686, New York, NY, USA. ACM.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

P.K. Srijith, Mark Hepple, Kalina Bontcheva, and Daniel Preotiuc-Pietro. 2017. Sub-story detection in Twitter with hierarchical Dirichlet processes. *Information Processing & Management*, 53(4):989 – 1003.

Zhongqing Wang and Yue Zhang. 2017. A neural model for joint event detection and summarization. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4158–4164. AAAI Press.

Chen Xing, Yuan Wang, Jie Liu, Yalou Huang, and Wei-Ying Ma. 2016. Hashtag-based sub-event discovery using mutually generative LDA in Twitter. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2666–2672. AAAI Press.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Siqi Zhao, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2011. Human as real-time sensors of social and physical events: A case study of Twitter and sports games. *arXiv preprint arXiv:1106.4300*.

Arkaitz Zubiaga, Damiano Spina, Enrique Amigó, and Julio Gonzalo. 2012. Towards real-time summarization of scheduled events from Twitter streams. In *Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, pages 319–320, New York, NY, USA. ACM.

GraphIE: A Graph-Based Framework for Information Extraction

Yujie Qian[1], **Enrico Santus**[1], **Zhijing Jin**[2], **Jiang Guo**[1], **and Regina Barzilay**[1]
[1]Computer Science and Artificial Intelligence Laboratory, MIT
[2]Department of Computer Science, The University of Hong Kong
{yujieq, jiang_guo, regina}@csail.mit.edu, {esantus, zhijing}@mit.edu

Abstract

Most modern Information Extraction (IE) systems are implemented as sequential taggers and only model local dependencies. Non-local and non-sequential context is, however, a valuable source of information to improve predictions. In this paper, we introduce GraphIE, a framework that operates over a graph representing a broad set of dependencies between textual units (i.e. words or sentences). The algorithm propagates information between connected nodes through graph convolutions, generating a richer representation that can be exploited to improve word-level predictions. Evaluation on three different tasks — namely textual, social media and visual information extraction — shows that GraphIE consistently outperforms the state-of-the-art sequence tagging model by a significant margin.[1]

1 Introduction

Most modern Information Extraction (IE) systems are implemented as sequential taggers. While such models effectively capture relations in the local context, they have limited capability of exploiting non-local and non-sequential dependencies. In many applications, however, such dependencies can greatly reduce tagging ambiguity, thereby improving overall extraction performance. For instance, when extracting entities from a document, various types of non-local contextual information such as co-references and identical mentions may provide valuable cues. See for example Figure 1, in which the non-local relations are crucial to discriminate the entity type of the second mention of *Washington* (i.e. PERSON or LOCATION).

Most of the prior work looking at the non-local dependencies incorporates them by constraining the output space in a structured prediction framework (Finkel et al., 2005; Reichart and Barzilay, 2012; Hu et al., 2016). Such approaches, however, mostly overlook the richer set of structural relations in the input space. With reference to the example in Figure 1, the co-referent dependencies would not be readily exploited by simply constraining the output space, as they would not necessarily be labeled as entities (e.g. pronouns). In the attempt to capture non-local dependencies in the input space, alternative approaches define a graph that outlines the input structure and engineer features to describe it (Quirk and Poon, 2017). Designing effective features is however challenging, arbitrary and time consuming, especially when the underlying structure is complex. Moreover, these approaches have limited capacity of capturing node interactions informed by the graph structure.

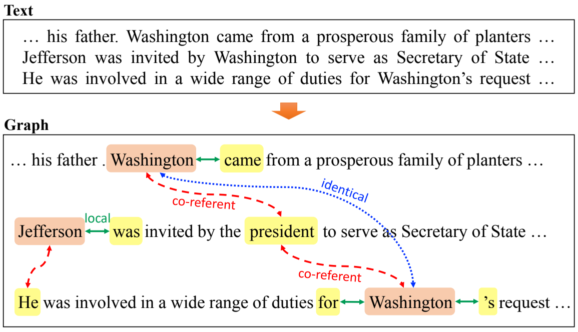


Figure 1: Example of the entity extraction task with an ambiguous entity mention (i.e. "...for Washington's request..."). Aside from the sentential forward and backward edges (green, solid) which aggregate local contextual information, non-local relations — such as the co-referent edges (red, dashed) and the identical-mention edges (blue, dotted) — provide additional valuable information to reduce tagging ambiguity (i.e. PERSON or LOCATION). Best viewed in color.

In this paper, we propose GraphIE, a framework

---

[1] Our code and data are available at https://github.com/thomas0809/GraphIE.

Proceedings of NAACL-HLT 2019, pages 751–761
Minneapolis, Minnesota, June 2 - June 7, 2019.

that improves predictions by automatically learning the interactions between local and non-local dependencies in the input space. Our approach integrates a graph module with the encoder-decoder architecture for sequence tagging. The algorithm operates over a graph, where nodes correspond to textual units (i.e. words or sentences) and edges describe their relations. At the core of our model, a recurrent neural network sequentially encodes local contextual representations and then the graph module iteratively propagates information between neighboring nodes using graph convolutions (Kipf and Welling, 2016). The learned representations are finally projected back to a recurrent decoder to support tagging at the word level.

We evaluate GraphIE on three IE tasks, namely textual, social media, and visual (Aumann et al., 2006) information extraction. For each task, we provide in input a simple task-specific graph, which defines the data structure without access to any major processing or external resources. Our model is expected to learn from the relevant dependencies to identify and extract the appropriate information. Experimental results on multiple benchmark datasets show that GraphIE consistently outperforms a strong and commonly adopted sequential model (SeqIE, i.e. a bi-directional long-short term memory (BiLSTM) followed by a conditional random fields (CRF) module). Specifically, in the textual IE task, we obtain an improvement of 0.5% over SeqIE on the CoNLL03 dataset, and an improvement of 1.4% on the chemical entity extraction (Krallinger et al., 2015). In the social media IE task, GraphIE improves over SeqIE by 3.7% in extracting the EDUCATION attribute from twitter users. In visual IE, finally, we outperform the baseline by 1.2%.

2 Related Work

The problem of incorporating non-local and non-sequential context to improve information extraction has been extensively studied in the literature. The majority of methods have focused on enforcing constraints in the output space during inference, through various mechanisms such as posterior regularization or generalized expectations (Finkel et al., 2005; Mann and McCallum, 2010; Reichart and Barzilay, 2012; Li et al., 2013; Hu et al., 2016).

Research capturing non-local dependencies in the input space have mostly relied on feature-based approaches. Roberts et al. (2008) and Swampillai and Stevenson (2011) have designed intra- and inter-sentential features based on discourse and syntactic dependencies (e.g., shortest paths) to improve relation extraction. Quirk and Poon (2017) used document graphs to flexibly represent multiple types of relations between words (e.g., syntactic, adjacency and discourse relations).

Graph-based representations can be also learned with neural networks. The most related work to ours is the graph convolutional network by Kipf and Welling (2016), which was developed to encode graph structures and perform node classification. In our framework, we adapt GCN as an intermediate module that learns non-local context, which — instead of being used directly for classification — is projected to the decoder to enrich local information and perform sequence tagging.

A handful of other information extraction approaches have used graph-based neural networks. Miwa and Bansal (2016) applied Tree LSTM (Tai et al., 2015) to jointly represent sequences and dependency trees for entity and relation extraction. On the same line of work, Peng et al. (2017) and Song et al. (2018) introduced Graph LSTM, which extended the traditional LSTM to graphs by enabling a varied number of incoming edges at each memory cell. Zhang et al. (2018) exploited graph convolutions to pool information over pruned dependency trees, outperforming existing sequence and dependency-based neural models in a relation extraction task. These studies differ from ours in several respects. First, they can only model word-level graphs, whereas our framework can learn non-local context either from word- or sentence-level graphs, using it to reduce ambiguity during tagging at the word level. Second, all these studies achieved improvements only when using dependency trees. We extend the graph-based approach to validate the benefits of using other types of relations in a broader range of tasks, such as co-reference in named entity recognition, *followed-by* link in social media, and layout structure in visual information extraction.

3 Problem Definition

We formalize information extraction as a sequence tagging problem. Rather than simply modeling inputs as sequences, we assume there exists a graph structure in the data that can be exploited to cap-

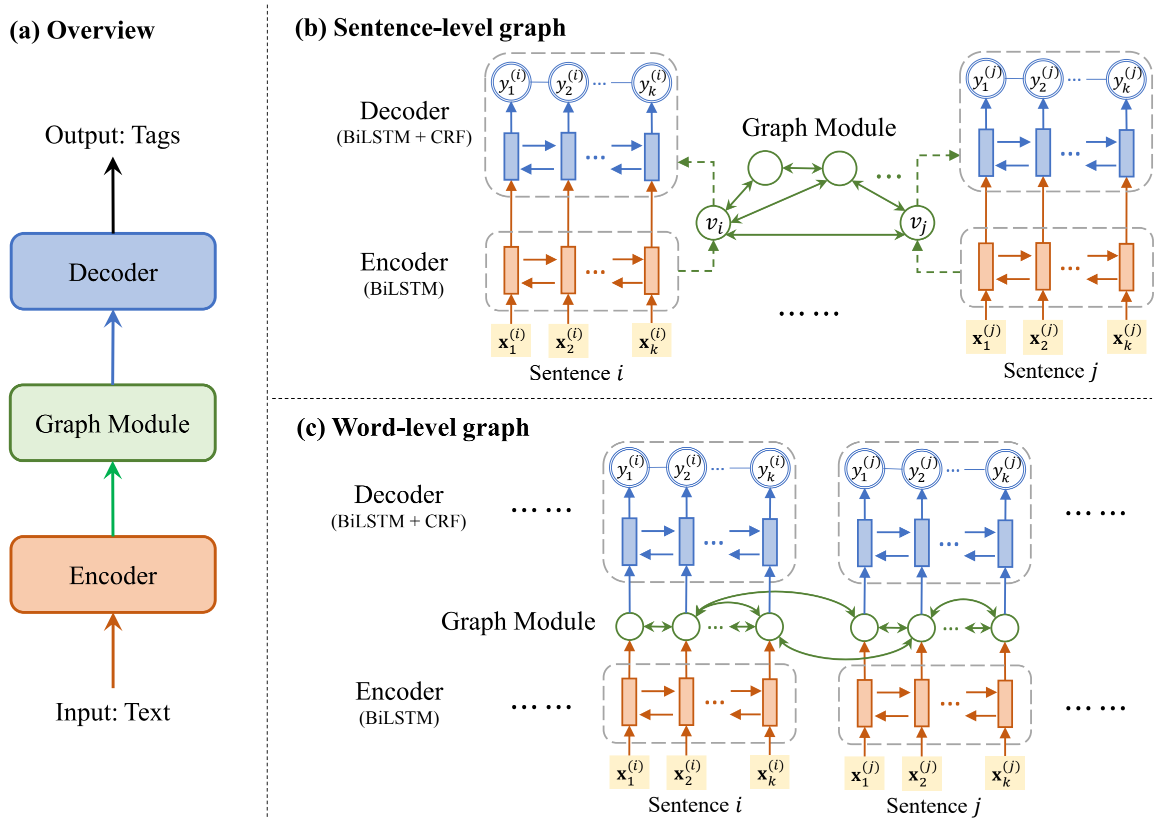


Figure 2: GraphIE framework: (a) an overview of the framework; (b) architecture for *sentence-level graph*, where each sentence is encoded to a node vector and fed into the graph module, and the output of the graph module is used as the initial state of the decoder; (c) architecture for *word-level graph*, where the hidden state for each word of the encoder is taken as the input node vector of the graph module, and then the output is fed into the decoder.

ture non-local and non-sequential dependencies between textual units, namely words or sentences.

We consider the input to be a set of sentences $S = \{s_1, \ldots, s_N\}$ and an auxiliary graph $G = (V, E)$, where $V = \{v_1, \ldots, v_M\}$ is the node set and $E \subset V \times V$ is the edge set. Each sentence is a sequence of words. We consider two different designs of the graph:

(1) *sentence-level graph*, where each node is a sentence (i.e. $M = N$), and the edges encode sentence dependencies;

(2) *word-level graph*, where each node is a word (i.e. M is the number of words in the input), and the edges connect pairs of words, such as co-referent tokens.

The edges $e_{i,j} = (v_i, v_j)$ in the graph can be either directed or undirected. Multiple edge types can also be defined to capture different structural factors underlying the task-specific input data.

We use the BIO (Begin, Inside, Outside) tagging scheme in this paper. For each sentence $s_i = (w_1^{(i)}, w_2^{(i)}, \ldots, w_k^{(i)})$,[2] we sequentially tag each word as $\mathbf{y}_i = (y_1^{(i)}, y_2^{(i)}, \ldots, y_k^{(i)})$.

4 Method

GraphIE jointly learns local and non-local dependencies by iteratively propagating information between node representations. Our model has three components:

- an *encoder*, which generates local context-aware hidden representations for the textual unit (i.e. word or sentence, depending on the task) with a recurrent neural network;
- a *graph module*, which captures the graph structure, learning non-local and non-sequential dependencies between textual units;
- a *decoder*, which exploits the contextual information generated by the graph module to perform labelling at the word level.

---

[2] While sentences may have different lengths, for notation simplicity we use a single variable k.

Figure 2 illustrates the overview of GraphIE and the model architectures for both sentence- and word-level graphs. In the following sections, we first introduce the case of the sentence-level graph, and then we explain how to adapt the model for the word-level graph.

4.1 Encoder

In GraphIE, we first use an encoder to generate text representations. Given a sentence $s_i = (w_1^{(i)}, w_2^{(i)}, \dots, w_k^{(i)})$ of length k, each word $w_t^{(i)}$ is represented by a vector $\mathbf{x}_t^{(i)}$, which is the concatenation of its word embedding and a feature vector learned with a character-level convolutional neural network (CharCNN; Kim et al. (2016)). We encode the sentence with a recurrent neural network (RNN), defining it as

$$\mathbf{h}_{1:k}^{(i)} = \mathtt{RNN}\left(\mathbf{x}_{1:k}^{(i)}\,; \mathbf{0}, \Theta_{\text{enc}}\right), \tag{1}$$

where $\mathbf{x}_{1:k}^{(i)}$ denotes the input sequence $[\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_k^{(i)}]$, $\mathbf{h}_{1:k}^{(i)}$ denotes the hidden states $[\mathbf{h}_1^{(i)}, \cdots, \mathbf{h}_k^{(i)}]$, $\mathbf{0}$ indicates the initial hidden state is zero, and Θ_{enc} represents the encoder parameters. We implement the RNN as a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), and encode each sentence independently.

We obtain the sentence representation for s_i by averaging the hidden states of its words, i.e. $\mathtt{Enc}(s_i) = \frac{1}{k}\left(\sum_{t=1}^{k} \mathbf{h}_t^{(i)}\right)$. The sentence representations are then fed into the graph module.

4.2 Graph Module

The graph module is designed to learn the non-local and non-sequential information from the graph. We adapt the graph convolutional network (GCN) to model the graph context for information extraction.

Given the sentence-level graph $G = (V, E)$, where each node v_i (i.e. sentence s_i) has the encoding $\mathtt{Enc}(s_i)$ capturing its local information, the graph module enriches such representation with neighbor information derived from the graph structure.

Our graph module is a GCN which takes as input the sentence representation, i.e. $\mathbf{g}_i^{(0)} = \mathtt{Enc}(s_i)$, and conducts graph convolution on every node, propagating information between its neighbors, and integrating such information into a new hidden representation. Specifically, each layer of GCN has two parts. The first gets the information of each node from the previous layer, i.e.

$$\boldsymbol{\alpha}_i^{(l)} = \mathbf{W}_v^{(l)}\, \mathbf{g}_i^{(l-1)}, \tag{2}$$

where $\mathbf{W}_v^{(l)}$ is the weight to be learned. The second aggregates information from the neighbors of each node, i.e. for node v_i, we have

$$\boldsymbol{\beta}_i^{(l)} = \frac{1}{d(v_i)} \cdot \mathbf{W}_e^{(l)} \left(\sum_{e_{i,j} \in E} \mathbf{g}_j^{(l-1)} \right), \tag{3}$$

where $d(v_i)$ is the degree of node v_i (i.e. the number of edges connected to v_i) and is used to normalize $\boldsymbol{\beta}_i^{(l)}$, ensuring that nodes with different degrees have representations of the same scale.[3] In the simplest case, where the edges in the graph are undirected and have the same type, we use the same weight $\mathbf{W}_e^{(l)}$ for all of them. In a more general case, where multiple edge types exist, we expect them to have different impacts on the aggregation. Thus, we model these edge types with different weights in Eq. 3, similar to the relational GCN proposed by Schlichtkrull et al. (2018). When edges are directed, i.e. edge $e_{i,j}$ is different from $e_{j,i}$, the propagation mechanism should mirror such difference. In this case, we consider directed edges as two types of edges (forward and backward), and use different weights for them.

Finally, $\boldsymbol{\alpha}_i^{(l)}$ and $\boldsymbol{\beta}_i^{(l)}$ are combined to obtain the representation at the l-th layer,

$$\mathbf{g}_i^{(l)} = \sigma\left(\boldsymbol{\alpha}_i^{(l)} + \boldsymbol{\beta}_i^{(l)} + b^{(l)}\right), \tag{4}$$

where $\sigma(\cdot)$ is the non-linear activation function, and $b^{(l)}$ is a bias parameter.

Because each layer only propagates information between directly connected nodes, we can stack multiple graph convolutional layers to get a larger receptive field, i.e. each node can be aware of more distant neighbors. After L layers, for each node v_i we obtain a contextual representation, $\mathtt{GCN}(s_i) = \mathbf{g}_i^{(L)}$, that captures both local and non-local information.

4.3 Decoder

To support tagging, the learned representation is propagated to the decoder.

---

[3] We choose this simple normalization strategy instead of the two-sided normalization in Kipf and Welling (2016), as it performs better in the experiments. The same strategy is also adopted by Zhang et al. (2018).

In our work, the decoder is instantiated as a BiLSTM+CRF tagger (Lample et al., 2016). The output representation of the graph module, $\mathtt{GCN}(s_i)$, is split into two vectors of the same length, which are used as the initial hidden states for the forward and backward LSTMs, respectively. In this way, the graph contextual information is propagated to each word through the LSTM. Specifically, we have

$$\mathbf{z}_{1:k}^{(i)} = \mathtt{RNN}\left(\mathbf{h}_{1:k}^{(i)} ; \mathtt{GCN}(s_i), \Theta_{\text{dec}}\right), \quad (5)$$

where $\mathbf{h}_{1:k}^{(i)}$ are the output hidden states of the encoder, $\mathtt{GCN}(s_i)$ represents the initial state, and Θ_{dec} is the decoder parameters. A simpler way to incorporate the graph representation into the decoder is concatenating with its input, but the empirical performance is worse than using as the initial state.

Finally, we use a CRF layer (Lafferty et al., 2001) on top of the BiLSTM to perform tagging,

$$\mathbf{y}_i^* = \underset{\mathbf{y}\in\mathbf{Y}_k}{\arg\max}\, p\left(\mathbf{y} \mid \mathbf{z}_{1:k}^{(i)} ; \Theta_{\text{crf}}\right), \quad (6)$$

where $\mathbf{Y}_k$ is the set of all possible tag sequences of length k, and Θ_{crf} represents the CRF parameters, i.e. transition scores of tags. CRF combines the local predictions of BiLSTM and the transition scores to model the joint probability of the tag sequence.[4]

4.4 Adaptation to Word-level Graphs

GraphIE can be easily adapted to model word-level graphs. In such case, the nodes represent words in the input, i.e. the number of nodes M equals the total number of words in the N sentences. At this point, each word's hidden state in the encoder can be used as the input node vector $\mathbf{g}_i^{(0)}$ of the graph module. GCN can then conduct graph convolution on the word-level graph and generate graph-contextualized representations for the words. Finally, the decoder directly operates on the GCN's outputs, i.e. we change the BiLSTM decoder to

$$\mathbf{z}_{1:k}^{(i)} = \mathtt{RNN}\left(\left[\mathtt{GCN}(w_1^{(i)}), \cdots, \mathtt{GCN}(w_k^{(i)})\right] ; \mathbf{0}, \Theta_{\text{dec}}\right),$$

where $\mathtt{GCN}(w_t^{(i)})$ is the GCN output for word $w_t^{(i)}$. In this case, the BiLSTM initial states are set to the default zero vectors. The CRF layer remains unchanged.

As it can be seen in Figure 2(c), the word-level graph module differs from the sentence-level one because it directly takes the word representations from the encoder and feeds its output to the decoder. In sentence-level graph, the GCN operates on sentence representations, which are then used as the initial states of the decoder BiLSTM.

5 Experimental Setup

We evaluate the model on three tasks, including two traditional IE tasks, namely textual information extraction and social media information extraction, and an under-explored task — *visual information extraction*. For each of these tasks, we created a simple task-specific graph topology, designed to easily capture the underlying structure of the input data without any major processing. Table 1 summarizes the three tasks.

5.1 Task 1: Textual Information Extraction

In this task, we focus on named entity recognition at discourse level (DiscNER). In contrast to traditional sentence-level NER (SentNER), where sentences are processed independently, in DiscNER, long-range dependencies and constraints across sentences have a crucial role in the tagging process. For instance, multiple mentions of the same entity are expected to be tagged consistently in the same discourse. Here we propose to use this (soft) constraint to improve entity extraction.

Dataset We conduct experiments on two NER datasets: the CoNLL-2003 dataset (CONLL03) (Tjong et al., 2003), and the CHEMDNER dataset for chemical entity extraction (Krallinger et al., 2015). We follow the standard split of each corpora. Statistics are shown in Table 2.

Graph Construction In this task, we use a word-level graph where nodes represent words. We create two types of edges for each document:

- *Local edges*: forward and backward edges are created between neighboring words in each sentence, allowing local contextual information to be utilized.
- *Non-local edges*: re-occurrences of the same token other than stop words are connected, so

---

[4] In GraphIE, the graph module models the input space structure, i.e. the dependencies between textual units (i.e. sentences or words), and the final CRF layer models the sequential connections of the output tags. Even though loops may exist in the input graph, CRF operates sequentially, thus the inference is tractable.

Evaluation Task	Graph Type	Node	Edge
Textual IE	word-level	word	1. non-local consistency (identical mentions) 2. local sentential forward and backward
Social Media IE	sentence-level	user's tweets	*followed-by*
Visual IE	sentence-level	text box	spatial layout (horizontal and vertical)

Table 1: Comparisons of graph structure in the three IE tasks used for evaluation.

DATASET		Train	Dev	Test
CONLL03	#doc	946	216	231
	#sent	14,987	3,466	3,684
CHEMDNER	#doc	3,500	3,500	3,000
	#sent	30,739	30,796	26,399

Table 2: Statistics of the CONLL03 and the CHEMDNER datasets (Task 1).

	EDUCATION	JOB
Users	7,208	1,772
Edges	11,167	3,498
Positive Tweets	49,793	3,694
Negative Tweets	393,683	172,349

Table 3: Statistics of the EDUCATION and JOB datasets (Task 2).

that information can be propagated through, encouraging global consistency of tagging.[5]

5.2 Task 2: Social Media Information Extraction

Social media information extraction refers to the task of extracting information from users' posts in online social networks (Benson et al., 2011; Li et al., 2014). In this paper, we aim at extracting *education* and *job* information from users' tweets. Given a set of tweets posted by a user, the goal is to extract mentions of the organizations to which they belong. The fact that the tweets are short, highly contextualized and show special linguistic features makes this task particularly challenging.

Dataset We construct two datasets, EDUCATION and JOB, from the Twitter corpus released by Li et al. (2014). The original corpus contains millions of tweets generated by $\approx$ 10 thousand users, where the *education* and *job* mentions are annotated using distant supervision (Mintz et al., 2009). We sample the tweets from each user, maintaining the ratio between positive and negative posts.[6] The obtained EDUCATION dataset consists of $443,476$ tweets generated by $7,208$ users, and the JOB dataset contains $176,043$ tweets generated by $1,772$ users. Dataset statistics are reported in Table 3.

The datasets are both split in 60% for training, 20% for development, and 20% for testing. We perform 5 different random splits and report the average results.

Graph Construction We construct the graph as *ego-networks* (Leskovec and Mcauley, 2012), i.e. when we extract information about one user, we consider the subgraph formed by the user and his/her direct neighbors. Each node corresponds to a Twitter user, who is represented by the set of posted tweets.[7] Edges are defined by the *followed-by* link, under the assumption that connected users are more likely to come from the same university or company. An example of the social media graph is reported in the appendices.

5.3 Task 3: Visual Information Extraction

Visual information extraction refers to the extraction of attribute values from documents formatted in various layouts. Examples include invoices and forms, whose format can be exploited to infer valuable information to support extraction.

Dataset The corpus consists of 25,200 Adverse Event Case Reports (AECR) recording drug-related side effects. Each case contains an average of 9 pages. Since these documents are produced by multiple organizations, they exhibit large variability in the layout and presentation styles (e.g.

---

[5] Note that other non-local relations such as co-references (cf. the example in Figure 1) may be used for further improvement. However, these relations require additional resources to obtain, and we leave them to future work.

[6] Positive and negative refer here to whether or not the *education* or *job* mention is present in the tweet.

[7] As each node is a set of tweets posted by the user, we encode every tweet with the encoder, and then average them to obtain the node representation. In the decoding phase, the graph module's output is fed to the decoder for each tweet.

text, table, etc.).[8] The collection is provided with a separate human-extracted ground truth database that is used as a source of distant supervision.

Our goal is to extract eight attributes related to the patient, the event, the drug and the reporter (cf. Table 6 for the full list). Attribute types include dates, words and phrases — which can be directly extracted from the document.

The dataset is split in 50% cases for training, 10% for development, and 40% for testing.

Graph Construction We first turn the PDFs to text using PDFMiner,[9] which provides words along with their positions in the page (i.e. bounding-box coordinates). Consecutive words are then geometrically joined into *text boxes*. Each text box is considered as a "sentence" in this task, and corresponds to a *node* in the graph.

Since the page layout is the major structural factor in these documents, we work on page-by-page basis, i.e. each page corresponds to a graph. The *edges* are defined to horizontally or vertically connect *nodes* (text boxes) that are close to each other (i.e. when the overlap of their bounding boxes, in either the vertical or horizontal direction, is over 50%). Four types of edge are considered: left-to-right, right-to-left, up-to-down, and down-to-up. When multiple nodes are aligned, only the closest ones are connected. An example of visual document graph is reported in the appendices.

5.4 Baseline and Our Method

We implement a two-layer BiLSTM with a conditional random fields (CRF) tagger as the sequential baseline (SeqIE). This architecture and its variants have been extensively studied and demonstrated to be successful in previous work on information extraction (Lample et al., 2016; Ma and Hovy, 2016). In the textual IE task (Task 1), our baseline is shown to obtain competitive results with the state-of-the-art method in the CONLL03 dataset. In the visual IE task (Task 3), in order to further increase the competitiveness of the baseline, we sequentially concatenate the horizontally aligned text boxes, therefore fully modeling the horizontal edges of the graph.

Our baseline shares the same encoder and decoder architecture with GraphIE, but without the graph module. Both architectures have similar computational cost. In Task 1, we apply GraphIE with word-level graph module (cf. Figure 2(c)), and in Task 2 and Task 3, we apply GraphIE with sentence-level graph module (cf. Figure 2(b)).

DATASET	Model	F1
CONLL03	Lample et al. (2016)	90.94
	Ma and Hovy (2016)	91.21
	Ye and Ling (2018)	91.38
	SeqIE	91.16
	GraphIE	**91.74***
CHEMDNER	Krallinger et al. (2015)	87.39
	SeqIE	88.28
	GraphIE	**89.71***

Table 4: NER accuracy on the CONLL03 and the CHEMDNER datasets (Task 1). Scores for our methods are the average of 5 runs. * indicates statistical significance of the improvement over SeqIE ($p < 0.01$).

5.5 Implementation Details

The models are trained with `Adam` (Kingma and Ba, 2014) to minimize the CRF objective. For regularization, we choose dropout with a ratio of 0.1 on both the input word representation and the hidden layer of the decoder. The learning rate is set to 0.001. We use the development set for early-stopping and the selection of the best performing hyperparameters. For CharCNN, we use 64-dimensional character embeddings and 64 filters of width 2 to 4 (Kim et al., 2016). The 100-dimensional pretrained GloVe word embeddings (Pennington et al., 2014) are used in Task 1 and 2, and 64-dimensional randomly initialized word embeddings are used in Task 3. We use a two-layer GCN in Task 1, and a one-layer GCN in Task 2 and Task 3. The encoder and decoder BiLSTMs have the same dimension as the graph convolution layer. In Task 3, we concatenate a positional encoding to each text box's representation by transforming its bounding box coordinates to a vector of length 32, and then applying a $\tanh$ activation.

6 Results

6.1 Task 1: Textual Information Extraction

Table 4 describes the NER accuracy on the CONLL03 (Tjong et al., 2003) and the CHEMDNER (Krallinger et al., 2015) datasets.

For CONLL03, we list the performance of existing approaches. Our baseline SeqIE obtains competitive scores compared to the best methods. The fact that GraphIE significantly outperforms

---

[8] This dataset cannot be shared for patient privacy and proprietary issues.

[9] `https://euske.github.io/pdfminer/`

DATASET	Dictionary			SeqIE			GraphIE		
	P	R	F1	P	R	F1	P	R	F1
EDUCATION	78.7	93.5	85.4	85.2	93.6	89.2	92.9	92.8	**92.9***
JOB	55.7	70.2	62.1	66.2	66.7	66.2	67.1	66.1	**66.5**

Table 5: Extraction accuracy on the EDUCATION and JOB datasets (Task 2). Dictionary is a naive method which creates a dictionary of entities from the training set and extracts their mentions during testing time. Scores are the average of 5 runs. * indicates the improvement over SeqIE is statistically significant (Welch's t-test, $p < 0.01$).

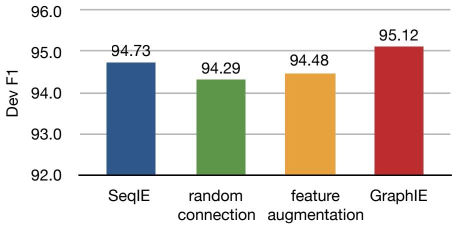


Figure 3: Analysis on the CoNLL03 dataset. We compare with two alternative designs: (1) *random connection*, where we replace the constructed graph by a random graph with the same number of edges; (2) *feature augmentation*, where we use the average embedding of each node and its neighbors as the input to the decoder, instead of the GCN which has additional parameters. We report F1 scores on the development set.

it, highlights once more the importance of modeling non-local and non-sequential dependencies and confirms that our approach is an appropriate method to achieve this goal.[10]

For CHEMDNER, we show the best performance reported in Krallinger et al. (2015), obtained with a feature-based method. Our baseline outperforms the feature-based method, and GraphIE further improves the performance by 1.4%.

Analysis To understand the advantage of GraphIE, we first investigate the importance of graph structure to the model. As shown in Figure 3, using random connections clearly hurts the performance, bringing down the F1 score of GraphIE from 95.12% to 94.29%. It indicates that the task-specific graph structures introduce beneficial inductive bias. Trivial feature augmentation also does not work well, confirming the necessity of learning the graph embedding with GCN.

We further conduct error analysis on the test set to validate our motivation that GraphIE resolves tagging ambiguity by encouraging consistency among identical entity mentions (cf. Figure 1). Here we examine the word-level tagging accuracy. We define the words that have more than one possible tags in the dataset as *ambiguous*. We find that among the 1.78% tagging errors of SeqIE, 1.16% are *ambiguous* and 0.62% are *unambiguous*. GraphIE reduces the error rate to 1.67%, with 1.06% to be *ambiguous* and 0.61% *unambiguous*. We can see that most of the error reduction indeed attributes to the *ambiguous* words.

6.2 Task 2: Social Media Information Extraction

Table 5 shows the results for the social media information extraction task. We first report a simple dictionary-based method as a baseline. Neural IE models achieve much better performance, showing that meaningful patterns are learned by the models rather than simply remembering the entities in the training set. The proposed GraphIE outperforms SeqIE in both the EDUCATION and JOB datasets, and the improvements are more significant for the EDUCATION dataset (3.7% versus 0.3%). The reason for such difference is the variance in the affinity scores (Mislove et al., 2010) between the two datasets. Li et al. (2014) underline that affinity value for EDUCATION is 74.3 while for JOB it is only 14.5, which means that in the datasets neighbors are 5 times more likely to have studied in the same university than worked in the same company. We can therefore expect that a model like GraphIE, which exploits neighbors' information, obtains larger advantages in a dataset characterized by higher affinity.

6.3 Task 3: Visual Information Extraction

Table 6 shows the results in the visual information extraction task. GraphIE outperforms the SeqIE baseline in most attributes, and achieves 1.2% improvement in the mirco average F1 score. It confirms that the benefits of using layout graph structure in visual information extraction.

The extraction performance varies across the attributes, ranging from 61.4% for *Drug Name* to

---

[10] We achieve the best reported performance among methods not using the recently introduced ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), which are pretrained on extra-large corpora and computationally demanding.

ATTRIBUTE	SeqIE			GraphIE		
	P	R	F1	P	R	F1
P. Initials	93.5	92.4	**92.9**	93.6	91.9	92.8
P. Age	94.0	91.6	92.8	94.8	91.1	**92.9**
P. Birthday	96.6	96.0	**96.3**	96.9	94.7	95.8
Drug Name	71.2	51.2	59.4	78.5	50.4	**61.4**
Event	62.6	65.2	63.9	64.1	68.7	**66.3**
R. First Name	78.3	95.7	86.1	79.5	95.9	**86.9**
R. Last Name	84.5	68.4	75.6	85.6	68.2	**75.9**
R. City	88.9	65.4	75.4	92.1	66.3	**77.1**
Avg. (macro)	83.7	78.2	80.3	85.7	78.4	**81.1**†
Avg. (micro)	78.5	73.8	76.1	80.3	74.6	**77.3**†

Table 6: Extraction accuracy on the AECR dataset (Task 3). Scores are the average of 5 runs. *P.* is the abbreviation for *Patient*, and *R.* for *Reporter*. † indicates statistical significance of the improvement over SeqIE ($p < 0.05$).

Model	Dev F1
GraphIE	77.8
– Edge types	77.0 (↓ 0.8)
– Horizontal edges	74.7 (↓ 3.1)
– Vertical edges	72.4 (↓ 5.4)
– CRF	72.1 (↓ 5.7)

Table 7: Ablation study (Task 3). Scores are micro average F1 on the development set. "–" means removing the element from GraphIE.

95.8% for *Patient Birthday* (similar variations are visible in the baseline). Similarly, the gap between GraphIE and SeqIE varies in relation to the attributes, ranging between -0.5% in *Patient Birthday* and 2.4% in *Event*.

In the ablation test described in Table 7, we can see the contribution of: using separate weights for different edge types ($+0.8\%$), horizontal edges ($+3.1\%$), vertical edges ($+5.4\%$), and CRF ($+5.7\%$).

Generalization We also assess GraphIE's capacity of dealing with unseen layouts through an extra analysis. From our dataset, we sample $2,000$ reports containing the three most frequent templates, and train the models on this subset. Then we test all models in two settings: 1) *seen templates*, consisting of $1,000$ additional reports in the same templates used for training; and 2) *unseen templates*, consisting of $1,000$ reports in two new template types.

The performance of GraphIE and SeqIE is reported in Figure 4. Both models achieve good results on *seen templates*, with GraphIE still scoring 2.8% higher than SeqIE. The gap becomes even larger when our model and the sequential one are tested on *unseen templates* (i.e. 20.3%), demonstrating that by explicitly modeling the richer structural relations, GraphIE achieves better generalizability.

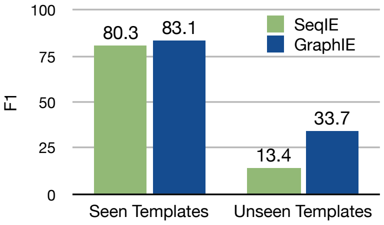


Figure 4: Micro average F1 scores tested on *seen* and *unseen* templates (Task 3).

7 Conclusions

We introduced GraphIE, an information extraction framework that learns local and non-local contextual representations from graph structures to improve predictions. The system operates over a task-specific graph topology describing the underlying structure of the input data. GraphIE jointly models the node (i.e. textual units, namely words or sentences) representations and their dependencies. Graph convolutions project information through neighboring nodes to finally support the decoder during tagging at the word level.

We evaluated our framework on three IE tasks, namely textual, social media and visual information extraction. Results show that it efficiently models non-local and non-sequential context, consistently enhancing accuracy and outperforming the competitive SeqIE baseline (i.e. BiLSTM+CRF).

Future work includes the exploration of automatically learning the underlying graphical structure of the input data.

Acknowledgments

We thank the MIT NLP group and the reviewers for their helpful comments. This work is supported by MIT-IBM Watson AI Lab. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Yonatan Aumann, Ronen Feldman, Yair Liberzon, Benjamin Rosenfeld, and Jonathan Schler. 2006. Visual information extraction. *Knowl. Inf. Syst.*, 10(1):1–15.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of ACL*, pages 389–398. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370. ACL.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of ACL*, pages 2410–2420.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*, pages 2741–2749. AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. Chemdner: The drugs and chemical names extraction challenge. *Journal of cheminformatics*, 7(1):S1.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, San Diego, California. ACL.

Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *NIPS*, pages 539–547.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of ACL*, volume 1, pages 165–174.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, volume 1, pages 73–82.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL*, pages 1064–1074, Berlin, Germany. ACL.

Gideon S Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11(Feb):955–984.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*, pages 1003–1011. ACL.

Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. 2010. You are who you know: inferring user profiles in online social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 251–260. ACM.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, volume 1, pages 2227–2237.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of ACL*, volume 1, pages 1171–1182.

Roi Reichart and Regina Barzilay. 2012. Multi event extraction guided by global constraints. In *Proceedings of NAACL-HLT*, pages 70–79. ACL.

Angus Roberts, Robert Gaizauskas, and Mark Hepple. 2008. Extracting clinical relationships from patient narratives. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 10–18. ACL.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state lstm. In *Proceedings of EMNLP*, pages 2226–2235.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting relations within and across sentences. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 25–32.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Kim Sang Tjong, F Erik, and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of NAACL-HLT*, pages 142–147. ACL.

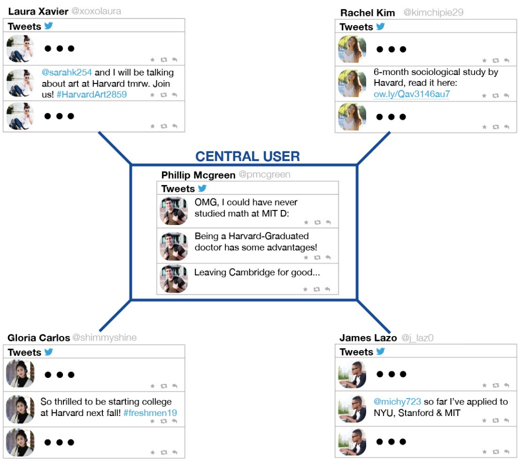
Zhixiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov crf for neural sequence labeling. In *Proceedings of ACL*, pages 235–240.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*.

A Appendices

We show some examples of the constructed graphs for different information extraction tasks.

A.1 Social Media Information Extraction



@sarahk254 and I will be talking about art at Harvard tmrw. Join us! #HarvardArt2859

6-month sociological study by Havard, read it here: ow.ly/Qav3146au7

OMG, I could have never studied math at MIT D:

Being a Harvard-Graduated doctor has some advantages!

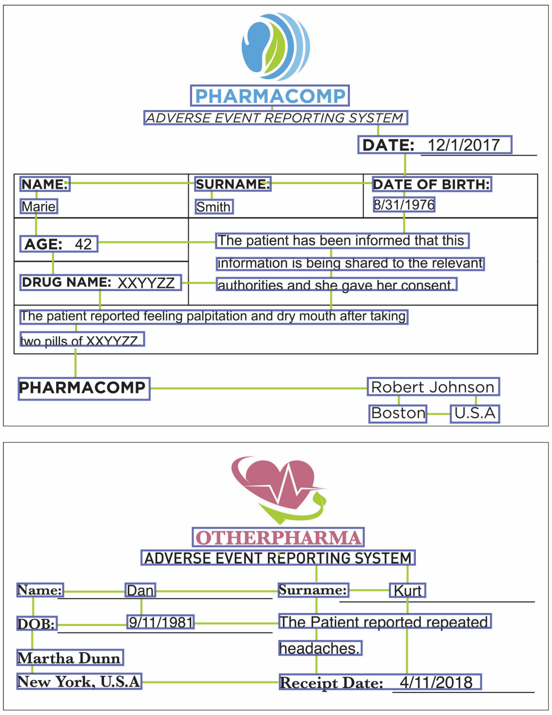
Leaving Cambridge for good...

So thrilled to be starting college at Harvard next fall! #freshmen19

@michy723 so far I've applied to NYU, Stanford & MIT

Figure 5: Mock-up example of Social Media Information Extraction (Task 2). Nodes are represented as users and edges are *follow-by* relations.

A.2 Visual Information Extraction



DATE: 12/1/2017

NAME: Marie

SURNAME: Smith

DATE OF BIRTH: 8/31/1976

AGE: 42

DRUG NAME: XXYYZZ

The patient has been informed that this information is being shared to the relevant authorities and she gave her consent.

The patient reported feeling palpitation and dry mouth after taking two pills of XXYYZZ

PHARMACOMP

Robert Johnson

Boston U.S.A

Name: Dan

Surname: Kurt

DOB: 9/11/1981

The Patient reported repeated headaches.

Martha Dunn

New York, U.S.A

Receipt Date: 4/11/2018

Figure 6: Mock-up example of Visual Information Extraction (Task 3). The two forms have different layouts. Graphical dependencies are shown as green lines connecting text in blue bounding-boxes.

OpenKI: Integrating Open Information Extraction and Knowledge Bases with Relation Inference

Dongxu Zhang[1]*, **Subhabrata Mukherjee**[2], **Colin Lockard**[2],
Xin Luna Dong[2], **Andrew McCallum**[1]
[1]University of Massachusetts Amherst
{dongxuzhang,mccallum}@cs.umass.com
[2]Amazon
{subhomj,clockard,lunadong}@amazon.com

Abstract

In this paper, we consider advancing web-scale knowledge extraction and alignment by integrating OpenIE extractions in the form of (subject, predicate, object) triples with Knowledge Bases (KB). Traditional techniques from universal schema and from schema mapping fall in two extremes: either they perform instance-level inference relying on embedding for (subject, object) pairs, thus cannot handle pairs absent in any existing triples; or they perform predicate-level mapping and completely ignore background evidence from individual entities, thus cannot achieve satisfying quality.

We propose *OpenKI* to handle sparsity of OpenIE extractions by performing instance-level inference: for each entity, we encode the rich information in its neighborhood in both KB and OpenIE extractions, and leverage this information in relation inference by exploring different methods of aggregation and attention. In order to handle unseen entities, our model is designed without creating entity-specific parameters. Extensive experiments show that this method not only significantly improves state-of-the-art for conventional OpenIE extractions like ReVerb, but also boosts the performance on OpenIE from semi-structured data, where new entity pairs are abundant and data are fairly sparse.

1 Introduction

Web-scale knowledge extraction and alignment has been a vision held by different communities for decades. The Natural Language Processing (NLP) community has been focusing on knowledge extraction from texts. They apply either closed information extraction according to an ontology (Mintz et al., 2009; Zhou et al., 2005), restricting to a subset of relations pre-defined in the ontology, or open information extraction (OpenIE) to extract free-text relations (Banko et al., 2007; Fader et al., 2011), leaving the relations unaligned and thus potentially duplicated. The Database (DB) community has been focusing on aligning relational data or WebTables (Cafarella et al., 2008) by schema mapping (Rahm and Bernstein, 2001), but the quality is far below adequate for assuring correct data integration.

We propose advancing progress in this direction by applying *knowledge integration* from OpenIE extractions. OpenIE extracts SPO (subject, predicate, object) triples, where each element is a text phrase, such as **E1:** *("Robin Hood", "Full Cast and Crew", "Leonardo Decaprio")* and **E2**: *("Ang Lee", "was named best director for", "Brokeback")*. OpenIE has been studied for text extraction extensively (Yates et al., 2007; Fader et al., 2011; Mausam et al., 2012), and also for semi-structured sources (Bronzi et al., 2013), thus serves an effective tool for web-scale knowledge extraction. The remaining problem is to align text-phrase predicates[1] from OpenIE to knowledge bases (KB). Knowledge integration answers the following question: given an OpenIE extraction (s, p, o), how can one populate an existing KB using relations in the pre-defined ontology?

The problem of knowledge integration is not completely new. The DB community has been solving the problem using schema mapping techniques, identifying mappings from a source schema (OpenIE extractions in our context) to a target schema (KB ontology in our context) (Rahm and Bernstein, 2001). Existing solutions consider *predicate-level* (*i.e.*, attribute) similarity on names, types, descriptions, instances, and so on, and generate mappings like "email" mapped to "email-

---

[*] This work was performed while at Amazon.

[1] We also need to align text-phrase entities, which falls in the area of entity linking (Dredze et al., 2010; Ji et al., 2014); it is out of scope of this paper and we refer readers to relevant references.

Proceedings of NAACL-HLT 2019, pages 762–772
Minneapolis, Minnesota, June 2 - June 7, 2019.

address"; "first name" and "last name" together mapped to "full name". However, for our example "Full Cast and Crew", which is a union of multiple KB relations such as "directed_by", "written_by", and "actor", it is very hard to determine a mapping at the predicate level.

On the other hand, the NLP community has proposed *Universal Schema* (Riedel et al., 2013) to apply *instance-level* inference from both OpenIE extractions and knowledge in existing knowledge bases: given a set of extractions regarding an entity pair (s, o) and also information of each entity, infer new relations for this pair. One drawback of this method is that it cannot handle unseen entities and entity pairs. Also, the technique tends to overfit when the data is sparse due to large number of parameters for entities and entity pairs. Unfortunately, in the majority of the real extractions we examined in our experiments, we can find only 1.4 textual triples on average between the subject and object. The latest proposal *Rowless Universal Schema* (Verga et al., 2017) removes the entity-specific parameters and makes the inference directly between predicates and relations, thereby allowing us to reason about unseen entity pairs. However, it completely ignores the entities themselves, so in a sense falls back to *predicate-level* decisions, especially when only one text predicate is observed.

In this paper we propose a solution that leverages information about the individual entities whenever possible, and falls back to predicate-level decisions only when both involved entities are new. Continuing with our example **E1** – if we know from existing knowledge that "Leonardo" is a famous actor and has rarely directed or written a movie, we can decide with a high confidence that this predicate maps to @film.actor in this triple, even if our knowledge graph knows nothing about the new movie "Robin Hood". In particular, we make three contributions in this paper.

1. We design an embedding for each entity by exploring rich signals from its neighboring relations and predicates in KB and OpenIE. This embedding provides a soft constraint on which relations the entities are likely to be involved in, while keeping our model free from creating new entity-specific parameters so allowing us to handle unseen entities during inference.
2. Inspired by predicate-level mapping from schema mapping and instance-level inference from universal schema, we design a joint model that leverages the neighborhood embedding of entities and relations with different methods of aggregation and attention.
3. Through extensive experiments on various OpenIE extractions and KB, we show that our method improves over state-of-the-arts by 33.5% on average across different datasets.

In the rest of the paper, we define the problem formally in Section 2, present our method in Section 3, describe experimental results in Section 4, and discuss related work in Section 5.

2 Problem Overview

Problem Statement. *Given (i) an existing knowledge base* KB *of triples* (s, p, o) *– where* $s, o \in E^{KB}$ *(set of KB entities) and* $p \in R^{KB}$ *(set of KB relations), and (ii) a set of instances* (s', p', o') *from OpenIE extraction (*s' *and* o' *may not belong to* E^{KB}*, and* p' *are text predicates)*[2]*: predict* $score(s', p, o')$ *– where* $p \in R^{KB}$.

For example, given **E1** and **E2** as OpenIE extractions and background knowledge bases (KB) like IMDB, we want to predict "@film.actor" relation given E1 and "@film.directed_by" relation given E2 as the target KB relations between the participating entities. Particularly, we want to perform this relation inference at *instance-level*, which can be different for different entities sharing the same predicate. Table 1 introduces important notations used in this paper.

s	Subject
o	Object
p	KB relation or text predicate
$v_s, v_p, v_o, v_{s,o}$	Embedding vectors of s, p, o and (s, o)
$S(s, p, o)$	Scoring function for (s, p, o) to be true
$Agg_{p \in R(s,o)}(v_p)$	Aggregation function over embeddings (v_p) of p shared by s and o.

Table 1: Notation table.

2.1 Existing Solution and Background

Universal Schema (F-Model) (Riedel et al., 2013) is modeled as a matrix factorization task where entity pairs, e.g., *(RobinHood, Leonardo Decaprio)* form the rows, and relations from OpenIE and KB form the columns (e.g., @film.actor, "Full Cast and Crew"). During training, we observe some positive entries in the matrix and the objective is to predict the missing cells at test time.

---

[2] In this paper, a 'relation' always refers to a KB relation, whereas a 'predicate' refers to an OpenIE textual relation.

Each (subject, predicate, object) triple is scored as:

$$S_F(s, p, o) = v_{s,o} \cdot v_p^T$$

where, $v_{s,o} \in \mathcal{R}^d$ is the embedding vector of the entity pair (subject, object), v_p is the embedding vector of a KB relation or OpenIE predicate, and the triple score is obtained by their dot product. The parameters v_p and $v_{s,o}$ are randomly initialized and learned via gradient descent.

One of the drawbacks of universal schema is the explicit modeling of entity pairs using free parameters $v_{s,o}$. Therefore, it cannot model unseen entities. This also makes the model overfit on our data as the number of OpenIE text predicates observed with each entity pair is rather small (1.4 on average in our datasets).

Universal Schema (E-Model) (Riedel et al., 2013) considers <u>e</u>ntity-level information, thus decomposing the scoring function from the F-model as follows:

$$\begin{aligned} S_E(s, p, o) &= S_{subj}(s, p) + S_{obj}(p, o) \\ &= v_s \cdot v_p^{subj^T} + v_o \cdot v_p^{obj^T} \end{aligned} \quad (1)$$

where each relation is represented by two vectors corresponding to its argument type for a subject or an object. The final score is an additive summation over the subject and object scores S_{subj} and S_{obj} that implicitly contain the argument type information of the predicate p. Thus, a joint F- and E-model of $S_{F+E} = S_F + S_E$ can perform relation inference at *instance-level* considering the entity information. Although the E-model captures rich information about entities, it still cannot deal with unseen entities due to the entity-specific free parameters v_s and v_o.

Rowless Universal Schema (Rowless) (Verga et al., 2017) handles new entities as follows. It considers all relations in KB and OpenIE that the subject s and object o co-participates in (denoted by $R(s, o)$), and represents the entity pair with an aggregation over embeddings of these relations.

$$\begin{aligned} v_{s,o}^{Rowless} &= Agg_{p' \in R(s,o)}(v_{p'}) \\ S_{Rowless}(s, p, o) &= v_{s,o}^{Rowless} \cdot v_p^T \end{aligned} \quad (2)$$

$Agg(.)$ is an aggregation function like average pooling, max pooling, hard attention (Rowless MaxR) or soft attention given query relations (Rowless Attention) (Verga et al., 2017). The Rowless model ignores the individual information of entities, and therefore falls back to making *predicate-level* decisions in a sense, especially when there are only a few OpenIE predicates for an entity pair.

3 Our Approach

We propose *OpenKI* for *instance-level* relation inference such that it (i) captures rich information about each entity from its neighborhood KB relations and text predicates to serve as background knowledge and generalizes to unseen entities by *not* learning any entity-specific parameters (only KB relations and OpenIE predicates are parameterized) (ii) considers both shared predicates and entity neighborhood information to encode entity pair information. Figure 1 shows the architecture of our model.

3.1 Entity Neighborhood Encoder (ENE)

The core of our model is the Entity Neighborhood Encoder. Recall that Rowless Universal Schema represents each entity pair with common relations shared by this pair. However, it misses critical information when entities do not only occur in the current entity pair, but also interact with other entities. This *entity neighborhood* can be regarded as a soft and fine-grained entity type information that could help infer relations when observed text predicates are ambiguous (polysemous), noisy (low quality of data source) or low-frequency (sparsity of language representation). [3]

Our aim is to incorporate this entity neighborhood information into our model for *instance-level* relation inference while keeping it free of entity-specific parameters. To do this, for each entity, we leverage all its *neighboring KB relations and OpenIE predicates* for relation inference. We aggregate their embeddings to obtain two scores for the subject and object separately in our ENE model. The subject score S_{subj}^{ENE} for an entity considers the aggregated embedding of its participating KB relations and OpenIE predicates where it serves as a subject (similar for the object score S_{obj}^{ENE}):

---

[3] Note that, the notion of *entity neighborhood* is different from the *Neighborhood* model in the Universal Schema work (Riedel et al., 2013). Our *entity neighborhood* captures information of each entity, whereas their Neighborhood model leverages prediction from similar predicates.

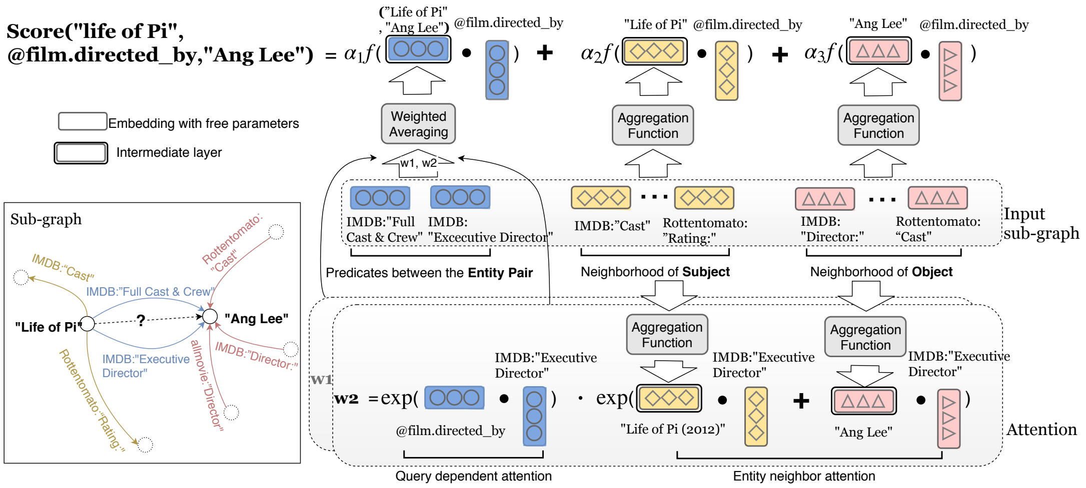


Figure 1: Architecture of the proposed method. In this example, the ENE model uses "Ang Lee's" neighboring predicates "IMDB:Director" and "allmovie:Director" for predicting the target KB relation "@film.directed_by". The attention mechanism assigns a larger weight over "IMDB:Executive Director" for generating entity pair embedding. Different colors of vectors represent different sets of parameters. The Entity Neighborhood Encoder (ENE) (square & triangle fillers) model contributes the following components to the final scoring function: (1) entity neighborhood scores S^{ENE}_{subj} and S^{ENE}_{obj} of the subject and object respectively; (2) neighborhood signal for the attention module to calculate the weight of each text predicate and the attention score $S^{Att}(s,p,o)$ (circle filler).

$$
\begin{aligned}
v^{agg}_{subj} &= Agg_{p' \in R(s,\cdot)}(v^{subj}_{p'}) \\
S^{ENE}_{subj}(s,p) &= v^{agg}_{subj} \cdot v^{subj\ T}_{p} \\
v^{agg}_{obj} &= Agg_{p' \in R(\cdot,o)}(v^{obj}_{p'}) \\
S^{ENE}_{obj}(p,o) &= v^{agg}_{obj} \cdot v^{obj\ T}_{p}
\end{aligned} \tag{3}
$$

$R(s,.)$ denotes all neighboring relations and predicates of the subject s (similar for the object). v^{subj}_p and v^{obj}_p are the only free parameters in ENE. These are randomly initialized and then learned via gradient descent. We choose average pooling as our aggregation function to capture the proportion of different relation and predicate types within the target entity's neighborhood.

3.2 Attention Mechanism

Given multiple predicates between a subject and an object, only some of them are important for predicting the target KB relation between them. For example, in Figure 1, the predicate "Executive Director" is more important than "Full Cast & Crew" to predict the KB relation "@film.directed_by" between "Life of Pi" and "Ang Lee".

We first present a query-based attention mechanism from earlier work, and then present our own solution with a neighborhood attention and combining both in a dual attention mechanism.

3.2.1 Query Attention

The first attention mechanism uses a query relation q (i.e., the target relation we may want to predict) to find out the importance (weight) $w_{p|q}$ of different predicates p with respect to q with v_p and v_q as the corresponding relation embeddings.

$$
w_{p|q} = \frac{\exp(v_q \cdot v_p^T)}{\sum_{p'} \exp(v_q \cdot v_{p'}^T)}
$$

Thus, given each query relation q, the model tries to find evidence from predicates that are most relevant to the query. Similar techniques have been used in (Verga et al., 2017). We can also use hard attention (referred as MaxR) instead of soft attention where the maximum weight is replaced with one and others with zero. One potential shortcoming of this attention mechanism is its sensitivity to noise, whereby it may magnify sparsely observed predicates between entities.

3.2.2 Neighborhood Attention

In this attention mechanism, we use the subject and object's neighborhood information as a filter to remove unrelated predicates. Intuitively, the entity representation generated by the ENE from its neighboring relations can be regarded as a soft and fine-grained entity type information.

Consider the embedding vectors v^{agg}_{subj} and v^{agg}_{obj} in Equation 3 that are aggregated from the entity's neighboring predicates and relations using an aggregation function. We compute the similarity $w_{p|Nb}$ between an entity's neighborhood information given by the above embeddings and a text predicate p to enforce a soft and fine-grained argument type constraint over the text predicate:

$$w_{p|Nb} = \frac{\exp(v_{subj}^{agg} \cdot v_p^{subj^T} + v_{obj}^{agg} \cdot v_p^{obj^T})}{\sum_{p'} \exp(v_{subj}^{agg} \cdot v_{p'}^{subj^T} + v_{obj}^{agg} \cdot v_{p'}^{obj^T})}$$

Finally, we combine both the query-dependent and neighborhood-based attention into a *Dual Attention* mechanism:

$$w_{p|q+Nb} = w_{p|q} \cdot w_{p|Nb}$$
$$w_p = \frac{w_{p|q+Nb}}{\sum_{p'} w_{p'|q+Nb}}$$

And the score function is given by:

$$\begin{aligned} S^{Att}(s,q,o) &= Agg_{p \in R(s,o)}(v_p) \cdot v_q^T \\ &= \Big(\sum_p w_p v_p\Big) \cdot v_q^T \qquad (4) \end{aligned}$$

3.3 Joint Model: OpenKI

All of the above models capture different types of features. Given a target triple (s,p,o), we combine scores from Eq. 3 and Eq. 4 in our final OpenKI model. It aggregates the neighborhood information of s and o and also uses an attention mechanism to focus on the important predicates between s and o. Refer to Figure 1 for an illustration. The final score of (s,p,o) is given by:

$$\begin{aligned} score(s,p,o) &= f_1(S^{Att}(s,p,o)) * ReLU(\alpha_1) \\ &+ f_2(S_{subj}^{ENE}(s,p)) * ReLU(\alpha_2) \\ &+ f_3(S_{obj}^{ENE}(p,o)) * ReLU(\alpha_3) \end{aligned}$$

where $f_i(X) = \sigma(a_i X + b_i)$ normalizes different scores to a comparable distribution. $ReLU(\alpha_i)$ enforces non-negative weights that allow scores to only contribute to the final model without canceling each other. a_i, b_i, α_i are free parameters that are learned during the back propagation gradient descent process.

3.4 Training Process

Our task is posed as a ranking problem. Given an entity pair, we want the observed KB relations between them to have higher scores than the unobserved ones. Thus, a pair-wise ranking based loss function is used to train our model:

$$\begin{aligned} L(s,p_{pos},p_{neg},o) = \max(0, \gamma - score(s,p_{pos},o) \\ + score(s,p_{neg},o)) \end{aligned}$$

where p_{pos} refers to a positive relation, p_{neg} refers to a uniformly sampled negative relation, and γ is the margin hyper-parameter. We optimize the loss function using Adam (Kingma and Ba, 2014). The training process uses early stop according to the validation set.

3.5 Explicit Argument Type Constraint

Subject and object argument types of relations help in filtering out a large number of candidate relations that do not meet the argument type, and therefore serve as useful constraints for relation inference. Similar to (Yu et al., 2017), we identify the subject and object argument type of each relation by calculating its probability of co-occurrence with subject / object entity types. During inference, we select candidate relations by performing a post-processing filtering step using the subject and object's type information when available.

4 Experiments

4.1 Data

We experiment with the following OpenIE datasets and Knowledge Bases.

(i) Ceres (Lockard et al., 2019) works on semi-structured web pages (e.g., IMDB) and exploits the DOM Tree and XPath (Olteanu et al., 2002) structure in the page to extract triples like *(Incredibles 2, Cast and Crew, Brad Bird)* and *(Incredibles 2, Writers, Brad Bird)*. We apply Ceres on the SWDE (Hao et al., 2011) movie corpus to generate triples. We align these triples to two different knowledge bases: (i) IMDB and (ii) subset of Freebase with relations under */film* domain. The average length of text predicates is 1.8 tokens for Ceres extractions.

(ii) ReVerb (Fader et al., 2011) works at sentence level and employs various syntactic constraints like part-of-speech-based regular expressions and lexical constraints to prune incoherent and uninformative extractions. We use 3 million ReVerb extractions from ClueWeb where the subject is already linked to Freebase (Lin et al., 2012) [4]. We align these extractions to (i) entire Freebase and (ii) subset of Freebase with relations under */film* domain. The average length of text predicates is 3.4 tokens for ReVerb extractions.

---

[4] Extractions are downloadable at `http://knowitall.cs.washington.edu/linked_extractions/`

	ReVerb + Freebase	ReVerb + Freebase(/film)	Ceres + Freebase(/film)	Ceres + IMDB
Training set				
# entity pairs for model training	40,878	1,102	23,389	64,539
# KB relation types	250	64	54	66
# OpenIE predicate types	124,836	35,366	124	178
Test set				
# test triples	4938	402	986	998
Avg./Med. # text edges per entity pair	1.74 / 1	1.49 / 1	1.35 / 1	1.23 / 1
Avg./Med. # edges for each subj	95.71 / 9	100.27 / 23	48.80 / 44	121.06 / 110
Avg./Med. # kb edges for each subj	61.41 / 4	8.30 / 6	7.00 / 7	30.77 / 29
Avg./Med. # edges for each obj	699.89 / 62	24.81 / 8	558.33 / 9	775.74 / 12
Avg./Med. # kb edges for each obj	325.70 / 23	10.11 / 6	340.96 / 3	606.31 / 6

Table 2: Data Statistics (Avg: Average, Med: Median).

Models	ReVerb + Freebase	ReVerb + Freebase(/film)	Ceres + Freebase (/flim)	Ceres + IMDB
$P(p\|\overline{p}')$ (similar to PMI (Angeli et al., 2015))	0.412	0.301	0.507	0.663
$P(p\|s, \overline{p}', o)$	0.474	0.317	0.627	0.770
E-model (Riedel et al., 2013)	0.215	0.156	0.431	0.506
ENE	0.479	0.359	0.646	0.808
Rowless with MaxR (Verga et al., 2017)	0.318	0.285	0.481	0.659
Rowless with Query Attn. (Verga et al., 2017)	0.326	0.278	0.512	0.695
OpenKI with MaxR	0.500	**0.378**	0.649	0.802
OpenKI with Query Att.	0.497	0.372	**0.663**	0.800
OpenKI with Neighbor Att.	0.495	0.372	0.650	0.813
OpenKI with Dual Att.	**0.505**	0.365	0.658	**0.814**

Table 3: Mean average precision (MAP) of different models over four data settings.

In order to show the generalizability of our approach to traditional (non OpenIE) corpora, we also perform experiments in the New York Times (NYT) and Freebase dataset (Riedel et al., 2010), which is a well known benchmark for distant supervision relation extraction. We consider the sentences (average length of 18.8 tokens) there to be a proxy for text predicates. These results are presented in Section 4.5.

Data preparation: We collect all entity mentions M from OpenIE text extractions, and all candidate entities E_{KB} from KB whose name exists in M. We retain the sub-graph G_{KB} of KB triples where the subject and object belongs to E_{KB}. Similar to (Riedel et al., 2013), we use string match to collect candidate entities for each entity mention. For each pair of entity mentions, we link them if two candidate entities in E_{KB} share a relation in KB. Otherwise, we link each mention to the most common candidate. For entity mentions that cannot be linked to KB, we consider them as new entities and link together mentions that share same text .

For validation and test, we randomly hold-out a part of the entity pairs from G_{KB} where text predicates are observed. Our training data consists of the rest of G_{KB} and all the OpenIE text extractions. In addition, we exclude direct KB triples from training where corresponding entity pairs appear in the test data (following the data setting of (Toutanova et al., 2015)). Table 2 shows the data statistics [5].

We adopt a similar training strategy as Universal Schema for the Ceres dataset – that not only learns direct mapping from text predicates to KB relations, but also clusters OpenIE predicates and KB relations by their co-occurrence. However, for the ReVerb data containing a large number of text predicates compared to Ceres, we only learn the direct mapping from text predicates to KB relations that empirically works well for this dataset.

4.2 Verifying Usefulness of Neighborhood Information: Bayesian Methods

To verify the usefulness of the entity's neighborhood information, we devise simple Bayesian methods as baselines. The simplest method counts

---

[5] Our datasets with train, test, validation split are downloadable at `https://github.com/zhangdongxu/relation-inference-naacl19` for benchmarking.

the co-occurrence of text predicates and KB relations (by applying Bayes rule) to find the conditional probability $P(p|\overline{p}')$ of a target KB relation p given a set of observed text predicates $\overline{p}'$. This performs relation inference at *predicate-level*.

Then, we can include the entity's relational neighbors in the Bayesian network by adding the neighboring predicates and relations of the subject (given by p_s^N) and object (given by p_o^N) to find $P(p|s, \overline{p}', o)$, which performs relation inference at the *instance-level*. The graph structures of these three Bayesian methods are shown in Figure 2. For detailed formula derivation, please refer to *Appendix* A.1.

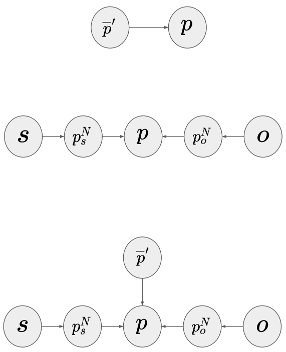


Figure 2: Structures of $P(p|\overline{p}')$, $P(p|s, o)$, $P(p|s, \overline{p}', o)$ are listed from top to bottom.

4.3 Baselines and Experimental Setup

Angeli et al. (2015) employ point-wise mutual information (PMI) between target relations and observed predicates to map OpenIE predicates to KB relations. This is similar to our Bayes conditional probability $P(p|\overline{p}')$. This baseline operates at *predicate-level*. To indicate the usefulness of entity neighborhood information, we also compare with $P(p|s, \overline{p}', o)$ as mentioned in Section 4.2. For the advanced embedding-based baselines, we compare with the E-model and the Rowless model (with MaxR and query attention) introduced in Section 2.1.

Hyper-parameters: In our experiments, we use 25 dimensional embedding vectors for the Rowless model, and 12 dimensional embedding vectors for the E- and ENE models. We use a batch-size of 128, and 16 negative samples for each positive sample in a batch. Due to memory constraints, we sample at most 8 predicates between entities and 16 neighbors for each entity during training. We use $\gamma = 1.0$ and set the learning rate to 5e-3 for ReVerb and 1e-3 for Ceres datasets.

Evaluation measures: Our task is a multi-label task, where each entity pair can share multiple KB relations. Therefore, we consider each KB relation as a query and compute the Mean Average Precision (MAP) – where entity pairs sharing the query relation should be ranked higher than those without the relation. In Section 4.4 we report MAP statistics for the 50 most common KB relations for ReVerb and Freebase dataset, and for the 10 most common relations in other domain specific datasets. The left out relations involve few triples to report any significant statistics. We also report the area under the precision recall curve (AUC-PR) for evaluation in Section 4.5.

4.4 Results

Table 3 shows that the overall results. OpenKI achieves significant performance improvement over all the baselines. Overall, we observe 33.5% MAP improvement on average across different datasets.

From the first two rows of Table 3, we observe the performance to improve as we incorporate neighborhood information into the Bayesian method. This depicts the strong influence of the entity's neighboring relations and predicates for relation inference.

The results show that our Entity Neighbor Encoder (ENE) outperforms the E-Model significantly. This is because the majority of the entity pairs in our test data have *at least one unseen entity* (refer to Table 4), which is very common in the OpenIE setting. The E-model cannot handle unseen entities because of its modeling of entity-specific parameters. This demonstrates the benefit of encoding entities with their neighborhood information (KB relations and text predicates) rather than learning entity-specific parameters. Besides, ENE outperforms the Rowless Universal Schema model, which does not consider any information surrounding the entities. This becomes a disadvantage in sparse data setting where only a few predicates are observed between an entity pair.

Finally, the results also show consistent improvement of OpenKI model over only-Rowless and only-ENE models. This indicates that the models are complementary to each other. We further observe significant improvements by applying different attention mechanisms over the OpenKI MaxR model – thus establishing the effectiveness of our attention mechanism.

Unseen entity: Table 4 shows the data statistics of unseen entity pairs in our test data. The most common scenario is that only one of the entity in a pair is observed during training, where our model benefits from the extra neighborhood information of the observed entity in contrast to the Rowless model.

Dataset	Both seen	One unseen	Both unseen
ReVerb + Freebase	864	3232	842
ReVerb + Freebase(/film)	27	147	228
Ceres + Freebase(/film)	383	561	42
Ceres + IMDB	462	533	3

Table 4: Statistics for unseen entities in test data. "Both seen" indicates both entities exist in training data; "One unseen" indicates only one of the entities in the pair exist in training data; "Both unseen" indicates both entities were unobserved during training.

Models	All data	At least one seen
Rowless Model	0.278	0.282
OpenKI with Dual Att.	0.365	0.419

Table 5: Mean average precision (MAP) of Rowless and OpenKI on ReVerb + Freebase (/film) dataset.

Table 5 shows the performance comparison on test data where at least one of the entity is known at test time. We choose ReVerb+Freebase(/film) for analysis because it contains the largest proportion of test triples where both entities are unknown during training. From the results, we observe that OpenKI outperforms the Rowless model by 48.6% when at least one of the entity in the triple is observed during training. Overall, we obtain 31.3% MAP improvement considering all of the test data. This validates the efficacy of encoding entity neighborhood information where at least one of the entities is known at test time. In the scenario where both entities are unknown at test time, the model falls back to the Rowless setting.

Models	MAP
Rowless Model	0.695
+Type Constraint	0.769 ↑ 10.6%
ENE Model	0.808
+Type Constraint	0.818 ↑ 1.2%
OpenKI with Dual Att.	0.814
+Type Constraint	**0.828** ↑ 1.7%

Table 6: MAP improvement with argument type constraints on Ceres + IMDB dataset.

Explicit Argument Type Constraint: As discussed in Section 3.5, incorporating explicit type constraints can improve the model performance. However, entity type information and argument type constraints are not always available especially for new entities. Table 6 shows the performance improvement of different models with entity type constraints. We observe the performance improvement of the ENE model to be much less than that of the Rowless model with explicit type constraint. This shows that the ENE model already captures soft entity type information while modeling the neighborhood information of an entity in contrast to the other methods that require explicit type constraint.

4.5 Results on NYT + Freebase Dataset

Prior works (Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016; Qin et al., 2018) on distantly supervised relation extraction performed evaluations on the New York Times (NYT) + Freebase benchmark data developed by Riedel et al. (2010)[6]. The dataset contains sentences whose entity mentions are annotated with Freebase entities as well as relations. The training data consists of sentences from articles in 2005-2006 whereas the test data consists of sentences from articles in 2007. There are 1950 relational facts in our test data[7]. In contrast to our prior experiments in the semi-structured setting with text predicates, in this experiment we consider the sentences to be a proxy for the text predicates.

Models	AUC-PR
PCNN + MaxR (Zeng et al., 2015)	0.325
PCNN + Att. (Lin et al., 2016)	0.341
ENE	0.421
OpenKI with Dual Att.	**0.461**

Table 7: Performances on NYT + Freebase data.

Table 7 compares the performance of our model with two state-of-the-art works (Zeng et al., 2015; Lin et al., 2016) on this dataset using AUC-PR as the evaluation metric.

Overall, OpenKI obtains 35% MAP improvement over the best performing PCNN baseline. In contrast to baseline models, our approach leverages the neighborhood information of each entity from the text predicates in the 2007 corpus and predicates / relations from the 2005-2006 corpus. This background knowledge contributes to the significant performance improvement.

---

[6] This data can be downloaded from `http://iesl.cs.umass.edu/riedel/ecml/`

[7] Facts of 'NA' (no relation) in the test data are not included in the evaluation process.

Note that, our model uses only the graph information from the entity neighborhood and does not use any text encoder such as Piecewise Convolutional Neural Nets (PCNN) (Zeng et al., 2015), where convolutional neural networks were applied with piecewise max pooling to encode textual sentences. This further demonstrates the importance of entity neighborhood information for relation inference. It is possible to further improve the performance of our model by incorporating text encoders as an additional signal. Some prior works (Verga et al., 2016; Toutanova et al., 2015) also leverage text encoders for relation inference.

5 Related Work

Relation Extraction: Mintz et al. (2009) utilize the entity pair overlap between knowledge bases and text corpus to generate signals for automatic supervision. To avoid false positives during training, many works follow the at-least-one assumption, where at least one of the text patterns between the entity pair indicate an aligned predicate in the KB (Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016). These works do not leverage graph information. In addition, Universal Schema (Riedel et al., 2013; Verga et al., 2017) tackled this task by low-rank matrix factorization. Toutanova et al. (2015) exploit graph information for knowledge base completion. However, their work cannot deal with unseen entities since entities' parameters are explicitly learned during training.

Schema Mapping: Traditional schema mapping methods (Rahm and Bernstein, 2001) involve three kinds of features, namely, language (name or description), type constraint, and instance level co-occurrence information. These methods usually involve hand-crafted features. In contrast, our model learns all the features automatically from OpenIE and KB with no feature engineering. This makes it easy to scale to different domains with little model tuning. Also, the entity types used in traditional schema mapping is always pre-defined and coarse grained, so cannot provide precise constraint of relations for each entity. Instead, our ENE model automatically learns soft and fine-grained constraints on which relations entities are likely to participate in. It is also compatible with pre-defined type systems.

Relation Grounding from OpenIE to KB: Instead of modeling existing schema, open information extraction (OpenIE) (Banko et al., 2007; Yates et al., 2007; Fader et al., 2011; Mausam et al., 2012) regards surface text mentions between entity pairs as separate relations, and do not require entity resolution or linking to KB. Since they do not model KB, it is difficult to infer KB relations only based on textual observations. Soderland et al. (2013) designed manual rules to map relational triples to slot types. Angeli et al. (2015) used PMI between OpenIE predicates and KB relations using distant-supervision from shared entity pairs for relation grounding. Yu et al. (2017) used word embedding to assign KB relation labels to OpenIE text predicates without entity alignment. These works do not exploit any graph information.

Entity Modeling for Relation Grounding: People leveraged several entity information to help relation extraction. Zhou et al. (2005) employed type information and observed 8% improvement of F-1 scores. Ji et al. (2017) encoded entity description to calculate attention weights among different text predicates within an entity pair. However, entity type and description information is not commonly available. Instead, the neighborhood information is easier to obtain and can also be regarded as entities' background knowledge. Universal Schema (Riedel et al., 2013) proposed an E-Model to capture entity type information. However, it can easily overfit in the OpenIE setting with large number of entities and a sparse knowledge graph.

6 Conclusion

In this work we jointly leverage relation mentions from OpenIE extractions and knowledge bases (KB) for relation inference and aligning OpenIE extractions to KB. Our model leverages the rich information (KB relations and OpenIE predicates) from the neighborhood of entities to improve the performance of relation inference. This also allows us to deal with new entities without using any entity-specific parameters. We further explore several attention mechanisms to better capture entity pair information. Our experiments over several datasets show 33.5% MAP improvement on average over state-of-the-art baselines.

Some future extensions include exploring more advanced graph embedding techniques without modeling entity-specific parameters and using text encoders as additional signals.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 344–354.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. 2013. Extraction and integration of partially overlapping web sources. *Proceedings of the VLDB Endowment*, 6(10):805–816.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1535–1545.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Heng Ji, Joel Nothman, Ben Hachey, et al. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*, pages 1333–1339.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas Lin, Oren Etzioni, et al. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. When open information extraction meets the semi-structured web. In *NAACL-HLT*. Association for Computational Linguistics.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 523–534.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Dan Olteanu, Holger Meuss, Tim Furche, and François Bry. 2002. Xpath: Looking forward. In *XML-Based Data Management and Multimedia Engineering - EDBT 2002 Workshops, EDBT 2002 Workshops XMLDM, MDDE, and YRWS, Prague, Czech Republic, March 24-28, 2002, Revised Papers*, pages 109–127.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. *arXiv preprint arXiv:1805.09927*.

Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Stephen Soderland, John Gilmer, Robert Bart, Oren Etzioni, and Daniel S Weld. 2013. Open information extraction to kbp relations in 3 hours. In *Proc. Text Analysis Conference (TAC2013)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California. Association for Computational Linguistics.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 613–622, Valencia, Spain. Association for Computational Linguistics.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael J. Cafarella, Oren Etzioni, and Stephen Soderland. 2007. Textrunner: Open information extraction on the web. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 25–26.

Dian Yu, Lifu Huang, and Heng Ji. 2017. Open relation extraction and grounding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 854–864.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

A Appendices

A.1 Derivation of Bayesian Inference Baselines

$$
\begin{aligned}
P(p|\overline{p}') &= \max_{p' \in R(s,o)} P(p|p') = \max_{p' \in R(s,o)} \frac{P(p,p')}{P(p')} \\
&= \max_{p' \in R(s,o)} \frac{\#(p,p')/\#entity\ pair}{\#p'/\#entity\ pair} \\
&\approx \max_{p' \in R(s,o)} \frac{\#(p,p') + \Delta}{\#(p') + |R^{text} \cup R^{KB}|\Delta}
\end{aligned}
$$

$R(s,o)$ is the set of observed predicates between subject s and object o. Δ is a smoothing factor which we choose 1e-6 in our implementation. Using conditional independence assumptions (refer to Figure 2):

$$
\begin{aligned}
P(p|s,o) &= \frac{P(p,s,o)}{P(s,o)} = \frac{P(p|s)P(s)P(p|o)P(o)}{P(s)P(o)} \\
&= \sum_{p_s^N \in R(s,\cdot)} P(p_s^N|s)P(p|p_s^N) \\
&\cdot \sum_{p_o^N \in R(\cdot,o)} P(p_o^N|o)P(p|p_o^N)
\end{aligned}
$$

$$
\begin{aligned}
P(p|s,\overline{p}',o) &= \frac{P(p,s,\overline{p}',o)}{P(s,\overline{p}',o)} \\
&= \frac{P(p|s)P(s)P(p|\overline{p}')P(\overline{p}')P(p|o)P(o)}{P(s)P(\overline{p}')P(o)} \\
&= P(p|\overline{p}') \sum_{p_s^N \in R(s,\cdot)} P(p_s^N|s)P(p|p_s^N) \\
&\cdot \sum_{p_o^N \in R(\cdot,o)} P(p_o^N|o)P(p|p_o^N)
\end{aligned}
$$

Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing

Wenhan Xiong[†], **Jiawei Wu**[†], **Deren Lei**[†], **Mo Yu**[*], **Shiyu Chang**[*], **Xiaoxiao Guo**[*], **William Yang Wang**[†]
[†] University of California, Santa Barbara
[*] IBM Research
{xwhan, william}@cs.ucsb.edu, yum@us.ibm.com, {shiyu.chang, xiaoxiao.guo}@ibm.com

Abstract

Existing entity typing systems usually exploit the type hierarchy provided by knowledge base (KB) schema to model label correlations and thus improve the overall performance. Such techniques, however, are not directly applicable to more open and practical scenarios where the type set is not restricted by KB schema and includes a vast number of free-form types. To model the underlying label correlations without access to manually annotated label structures, we introduce a novel label-relational inductive bias, represented by a graph propagation layer that effectively encodes both global label co-occurrence statistics and word-level similarities. On a large dataset with over 10,000 free-form types, the graph-enhanced model equipped with an attention-based matching module is able to achieve a much higher recall score while maintaining a high-level precision. Specifically, it achieves a 15.3% relative F1 improvement and also less inconsistency in the outputs. We further show that a simple modification of our proposed graph layer can also improve the performance on a conventional and widely-tested dataset that only includes KB-schema types.[1]

1 Introduction

Fine-grained entity typing is the task of identifying specific semantic types of entity mentions in given contexts. In contrast to general entity types (*e.g.*, organization, event), fine-grained types (*e.g.*, political party, natural disaster) are often more informative and can provide valuable prior knowledge for a wide range of NLP tasks, such as coreference resolution (Durrett and Klein, 2014), relation extraction (Yaghoobzadeh et al., 2016) and question answering (Lee et al., 2006; Yavuz et al., 2016).

Context	**Types**
Big Show then appeared at One Night Stand, attacking Tajiri, Super Crazy, and the Full Blooded Italians after their tag team match	person†, television_program*
	person, athlete, wrestler, entertainer
The womens pole vault at the 2010 IAAF World Indoor Championships was held at the ASPIRE Dome on 12 and 14 **March**.	month†, event*
	date, month

Table 1: Examples of inconsistent predictions produced by existing entity typing system that does not model label correlations. We use different subscript symbols to indicate contradictory type pairs and show the ground-truth types in *italics*.

In practical scenarios, a key challenge of entity typing is to correctly predict *multiple* ground-truth type labels from a large candidate set that covers a wide range of types in different granularities. In this sense, it is essential for models to effectively capture the inter-label correlations. For instance, if an entity is identified as a "criminal", then the entity must also be a "person", but it is less likely for this entity to be a "police officer" at the same time. When ignoring such correlations and considering each type separately, models are often inferior in performance and prone to inconsistent predictions. As shown in Table 1, an existing model that independently predicts different types fails to reject predictions that include apparent contradictions.

Existing entity typing research often address this aspect by explicitly utilizing a given type hierarchy to design hierarchy-aware loss functions (Ren et al., 2016b; Xu and Barbosa, 2018) or enhanced type label encodings (Shimaoka et al., 2017) that enable parameter sharing between related types. These methods rely on the assump-

---

[1] `https://github.com/xwhan/Extremely-Fine-Grained-Entity-Typing`

Proceedings of NAACL-HLT 2019, pages 773–784
Minneapolis, Minnesota, June 2 - June 7, 2019.

tion that the underlying type structures are predefined in entity typing datasets. For benchmarks annotated with the knowledge base (KB) guided distant supervision, this assumption is often valid since all types are from KB ontologies and naturally follow tree-like structures. However, since knowledge bases are inherently incomplete (Min et al., 2013), existing KBs only include a limited set of entity types. Thus, models trained on these datasets fail to generalize to lots of unseen types. In this work, we investigate entity typing in a more open scenario where the type set is not restricted by KB schema and includes over 10,000 free-form types (Choi et al., 2018). As most of the types do not follow any predefined structures, methods that explicitly incorporate type hierarchies cannot be straightforwardly applied here.

To effectively capture the underlying label correlations without access to known type structures, we propose a novel label-relational inductive bias, represented by a graph propagation layer that operates in the latent label space. Specifically, this layer learns to incorporate a label affinity matrix derived from global type co-occurrence statistics and word-level type similarities. It can be seamlessly coupled with existing models and jointly updated with other model parameters. Empirically, on the Ultra-Fine dataset (Choi et al., 2018), the graph layer alone can provide a significant 11.9% relative F1 improvement over previous models. Additionally, we show that the results can be further improved ($11.9\% \rightarrow 15.3\%$) with an attention-based mention-context matching module that better handles *pronouns* entity mentions. With a simple modification, we demonstrate that the proposed graph layer is also beneficial to the widely used OntoNotes dataset, despite the fact that samples in OntoNotes have lower label multiplicity (*i.e.*, average number of ground-truth types for each sample) and thus require less label-dependency modeling than the Ultra-Fine dataset.

To summarize, our major contribution includes:

- We impose an effective label-relational bias on entity typing models with an easy-to-implement graph propagation layer, which allows the model to implicitly capture type dependencies;
- We augment our graph-enhanced model with an attention-based matching module, which constructs stronger interactions between the mention and context representations;
- Empirically, our model is able to offer significant improvements over previous models on the Ultra-Fine dataset and also reduces the cases of inconsistent type predictions.

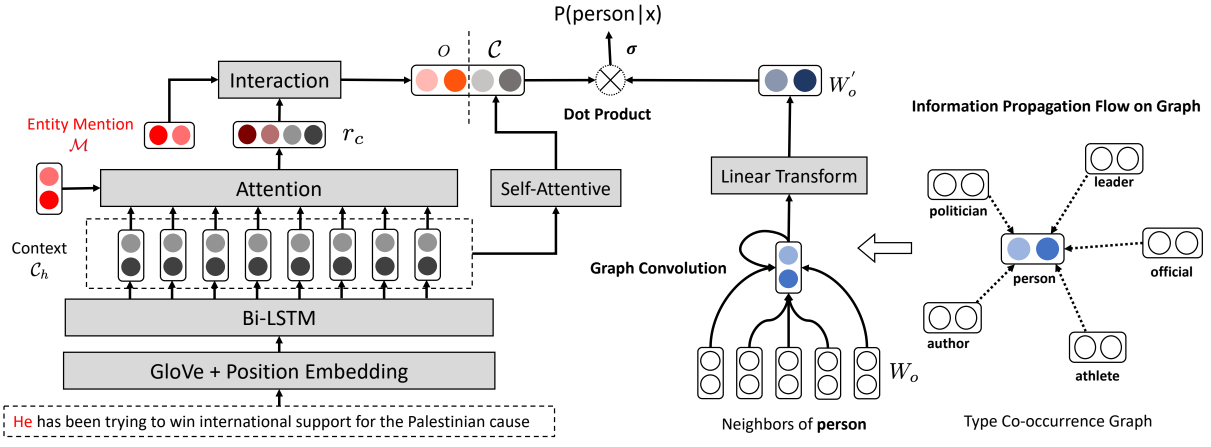
2 Related Work

Fine-Grained Entity Typing The task of fine-grained entity typing was first thoroughly investigated in (Ling and Weld, 2012), which utilized Freebase-guided distant supervision (DS) (Mintz et al., 2009) for entity typing and created one of the early large-scale datasets. Although DS provides an efficient way to annotate training data, later work (Gillick et al., 2014) pointed out that entity type labels induced by DS ignore entities' local context and may have limited usage in context-aware applications. Most of the following research has since focused on testing in context-dependent scenarios. While early methods (Gillick et al., 2014; Yogatama et al., 2015) on this task rely on well-designed loss functions and a suite of hand-craft features that represent both context and entities, Shimaoka et al. (2016) proposed the first attentive neural model which outperformed feature-based methods with a simple cross-entropy loss.

Modeling Entity Type Correlations To better capture the underlying label correlations, Shimaoka et al. (2017) employed a hierarchical label encoding method and AFET (Ren et al., 2016a) used the predefined label hierarchy to identify noisy annotations and proposed a partial-label loss to reduce such noise. A recent work (Xu and Barbosa, 2018) proposed hierarchical loss normalization which alleviated the noise of too specific types. Our work differs from these works in that we do not rely on known label structures and aim to learn the underlying correlations from data. Rabinovich and Klein (2017) recently proposed a structure-prediction approach which used type correlation features. The inference on their learned factor graph is approximated by a greedy decoding algorithm, which outperformed unstructured methods on their own dataset. Instead of using an explicit graphical model, we enforce a relational bias on model parameters, which does not introduce extra burden on label decoding.

3 Task Definition

Specifically, the task we consider takes a raw sentence C as well as an entity mention span M inside



a) Mention-Context Encoding

b) GCN over Type Vectors

Figure 1: Overview of the process to make predictions on the type **"person"**. **a)** Modules used to extract mention and context aware representations. **b)** An illustration of the graph layer operating over the type vector of **"person"**.

C as inputs, and aims to predict the correct type labels T_m of M from a candidate type set $\mathcal{T}$, which includes more than 10,000 free-form types. The entity span M here can be named entities, nominals and also pronouns. The ground-truth type set T_m here usually includes more than one types (approximately five types on average), making this task a multi-label classification problem.

4 Methodology

In this section, we first briefly introduce the neural architecture to encode raw text inputs. Then we describe the matching module we use to enhance the interaction between the mention span and the context sentence. Finally, we move to the label decoder, on which we impose the label-relational bias with a graph propagation layer that encodes type co-occurrence statistics and word-level similarities. Figure 1 provides a graphical overview of our model, with 1a) illustrating both the text encoders and the matching module, and 1b) showing an example of graph propagation.

4.1 Representation Model

Our base model to encode the context and the mention span follows existing neural approaches (Shimaoka et al., 2016; Xu and Barbosa, 2018; Choi et al., 2018). To encode the context, we first apply a standard Bi-LSTM, which takes GloVe (Pennington et al., 2014) embeddings and position embeddings (three vectors representing positions before, inside or after the mention span) as inputs and outputs the hidden states at each time step $t \in [1, l_c]$. With the derived hidden states $\mathcal{C}_h \in \mathbb{R}^{l_c \times h_c}$, we then apply a self-attentive encoder (McCann et al., 2017) on the top to get the final context representation $\mathcal{C}$. For the entity mention span, we concatenate the features derived by a character-level CNN and a similar self-attentive encoder. We denote the final mention representation as $\mathcal{M}$.[2]

4.2 Mention-Context Interaction

Since most previous datasets only consider *named entities*, a simple concatenation of the two features $[\mathcal{C}; \mathcal{M}]$ followed by a linear output layer (Shimaoka et al., 2016, 2017) usually works reasonably well when making predictions. This suggests that $\mathcal{M}$ itself provides important information for recognizing entity types. However, as in our target dataset, a large portion of entity mentions are actually *pronouns*, such as "he" or "it", this kind of mentions alone provide only limited clues about general entity types (*e.g.*, "he" is a "person") but little information about fine-grained types. In this case, directly appending representation of pronouns does not provide extra useful information for making fine-grained predictions. Thus, instead of using the concatenation operator, we propose to construct a stronger interaction between the mention and context with an attention-based matching module, which has shown its effectiveness in recent natural language inference models (Mou et al., 2016; Chen et al., 2017).

Formally consider the mention representation $\mathcal{M} \in \mathbb{R}^{h_m}$ and context's hidden feature $\mathcal{C}_h \in$

---

[2] Please refer to (Shimaoka et al., 2017) and (Choi et al., 2018) for more detailed descriptions.

$\mathbb{R}^{l_c \times h_c}$, where l_c indicates the number of tokens in the context sentence and h_m, h_c denote feature dimensions. We first project the mention feature $\mathcal{M}$ into the same dimension space as $\mathcal{C}_h$ with a linear layer ($W_1 \in \mathbb{R}^{h_m \times h_c}$) and a $tanh$ function[3]:

$$m_{proj} = tanh(W_1^T \mathcal{M}), \tag{1}$$

then we perform bilinear attention matching between m_{proj} and $\mathcal{C}_h$, resulting in an affinity matrix $\mathcal{A}$ with dimension $\mathcal{A} \in \mathbb{R}^{1 \times l_c}$:

$$\mathcal{A} = m_{proj} \times W_a \times \mathcal{C}_h, \tag{2}$$

where $W_a \in \mathbb{R}^{h_c \times h_c}$ is a learnable matrix. If we consider the mention feature as query and the context as memory, we can use the affinity matrix to retrieve the relevant parts in the context:

$$\bar{\mathcal{A}} = softmax(\mathcal{A}) \tag{3}$$

$$r_c = \bar{\mathcal{A}} \times \mathcal{C}_h. \tag{4}$$

With the projected mention representation m_{proj} and the retrieved context feature r_c, we define the following interaction operators:

$$r = \rho(W_r[r_c; m_{proj}; r_c - m_{proj}]) \tag{5}$$

$$g = \sigma(W_g[r_c; m_{proj}; r_c - m_{proj}]) \tag{6}$$

$$o = g * r + (1 - g) * m_{proj}, \tag{7}$$

where $\rho(\cdot)$ is a gaussian error linear unit (Hendrycks and Gimpel, 2016) and r is the fused context-mention feature; $\sigma(\cdot)$ indicates a *sigmoid* function and g is the resulting gating function, which controls how much information in mention span itself should be passed down. We expect the model to focus less on the mention representation when it is not informative. The concatenation $[r_c; m_{proj}; r_c - m_{proj}]$ here is supposed to capture different aspects of the interactions. To emphasize the context's impact, we finally concatenate the extracted context feature ($\mathcal{C}$) with the output (o) of the matching module ($f = [o; \mathcal{C}]$) for prediction.

4.3 Imposing Label-Relational Inductive Bias

For approaches that ignore the underlying label correlations, the type predictions are considered as N independent binary classification problems, with N being the number of types. If we denote the feature extracted by any arbitrary neural model as $f \in \mathbb{R}^{d_f}$, then the probability of being any given type is calculated by:

$$p = \sigma(W_o f), W_o \in \mathbb{R}^{N \times d_f}. \tag{8}$$

We can see that every row vector of W_o is responsible for predicting the probability of one particular type. We will refer the row vectors as type vectors for the rest of this paper. As these type vectors are independent, the label correlations are only implicitly captured by sharing the model parameters that are used to extract f. We argue that the paradigm of parameter sharing is not enough to impose strong label dependencies and the values of type vectors should be better constrained.

A straightforward way to impose the desired constraints is to add extra regularization terms on W_o. We first tested several auxiliary loss functions based on the heuristics from GloVe (Pennington et al., 2014), which operates on the type co-occurrence matrix. However, the auxiliary losses only offer trivial improvements in our experiments. Instead, we find that directly imposing a model-level inductive bias on the type vectors turns out to be a more principled solution. This is done by adding a graph propagation layer over randomly initialized W_o and generating the updated type vectors $W_o^{'}$, which is used for final prediction. Both W_o and the graph convolution layer are learned together with other model parameters. We view this layer as the key component of our model and use the rest of this section to describe how we create the label graph and compute the propagation over the graph edges.

Label Graph Construction In KB-supervised datasets, the entity types are usually arranged in tree-like structures. Without any prior about type structures, we consider a more general graph-like structure. While the nodes in the graph straightforwardly represent entity types, the meaning of the edges is relatively vague, and the connections are also unknown. In order to create meaningful edges using training data as the only resource, we utilize the type co-occurrence matrix: if two type t_1 and t_2 both appear to be the true types of a particular entity mention, we will add an edge between them. In other words, we are using the co-occurrence statistics to approximate the pair-wise dependencies and the co-occurrence matrix now serves as the adjacent matrix. Intuitively, if t_2 co-appears with t_1 more often than another type t_3, the probabilities of t_1 and t_2 should have stronger depen-

---

[3] $tanh$ here is used to make m_{proj} in the same scale as $\mathcal{C}_h$, which was the output of a $tanh$ function inside LSTM.

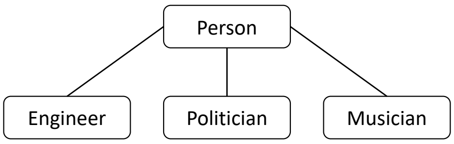


Figure 2: A snippet of the underlying type co-occurrence graph. Multiple edges between nodes are omitted here for clarity.

dencies and the corresponding type vectors should be more similar in the vector space. In this sense, we expect each type vector to effectively capture the local neighbor structure on the graph.

Correlation Encoding via Graph Convolution To encode the neighbor information into each node's representation, we follow the propagation rule defined in Graph Convolution Network (GCN) (Kipf and Welling, 2016). In particular, with the adjacent or co-occurrence matrix A, we define the following propagation rule on W_o:

$$W_o^{'} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}W_oT \tag{9}$$

$$\tilde{A} = A + I_N. \tag{10}$$

Here $T \in \mathbb{R}^{d_f \times d_f}$ is the transformation matrix and I_N is an identity matrix used to add self-connected edges. $\tilde{D}$ is a diagonal degree matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, which is used to normalize the feature vectors such that the number of neighbors does not affect the scale of transformed feature vectors. In our experiments, we find that an alternative propagation rule

$$W_o^{'} = \tilde{D}^{-1}\tilde{A}W_oT \tag{11}$$

works similarly well and is more efficient as it involves less matrix multiplications. If we look closely and take each node out, the propagation can be written as

$$W_o^{'}[i,:] = \frac{1}{\sum_j \tilde{A}_{ij}}(\sum_j \tilde{A}_{ij}W_o[j,:]T). \tag{12}$$

From this formula, we can see that the propagation is essentially gathering features from the first-order neighbors. In this way, the prediction on type t_i is dependent on its neighbor types.

Compared to original GCNs that often use multi-hop propagations (*i.e.*, multiple graph layers connected by nonlinear functions) to capture higher-order neighbor structures. We only apply one-hop propagation and argue that high-order label dependency is not necessarily beneficial in our scenario and might introduce false bias. A simple illustration is shown in Figure 2. We can see that propagating 2-hop information introduces undesired inductive bias, since types that are more than 1-hop away (*e.g.*, "Engineer" and "Politician") usually do not have any dependencies. In fact, some of the 2-hop type pairs can be contradictory types (*e.g.*, "police" and "prisoner"). This hypothesis is consistent with our experiment results: adding more than one graph layer leads to worse results. Additionally, we also omit GCN's nonlinear activation which introduces unnecessary constraints on the scale of $W_o^{'}$, with which we calculate the unscaled scores before calculating the probability via a sigmoid function.

4.4 Leveraging Label Word Embeddings

As the type labels are all written as text phrases, an interesting question is whether we can exploit the semantics provided by pre-trained word embeddings to improve entity typing. We explore this possibility by using the cosine similarity of word embeddings. We first calculate type embeddings by simply summing the embeddings of all tokens in the type name. Then we build a label affinity matrix A_{word} by calculating pair-wise cosine similarities. With the assumption that word-level similarity measures some degree of label dependency, we propose to integrate A_{word} into the graph convolution layer following

$$A_{word}^{'} = (A_{word} + 1)/2 \tag{13}$$

$$W_o^{'} = \tilde{D}^{-1}(\tilde{A} + \lambda A_{word}^{'})W_oT. \tag{14}$$

Here Equation 13 scales the similarity value into $(0, 1]$ to avoid negative edge weights, which might introduce numerical issues when calculating $\tilde{D}^{-1}$. λ is a trainable parameter used to weight the impact of word-level similarities. As will be shown in Section 5, this simple augmentation provides further improvement over our original model.

5 Experiments

5.1 Experiment Setup

Datasets Our experiments mainly focus on the Ultra-Fine entity typing dataset which has 10,331 labels and most of them are defined as free-form text phrases. The training set is annotated with heterogeneous supervisions based on KB, Wikipedia and head words in dependency trees,

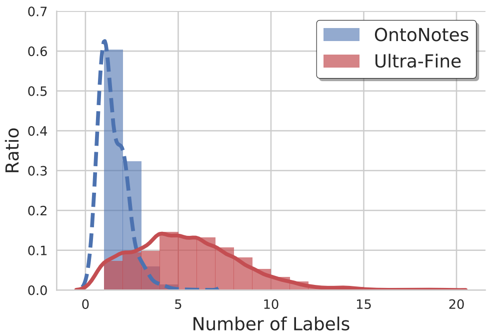


Figure 3: Label multiplicity distribution of the datasets.

resulting in about 25.2M[4] training samples. This dataset also includes around 6,000 crowdsourced samples. Each of these samples has five ground-truth labels on average. For a fair comparison, we use the original test split of the crowdsourced data for evaluation. To better understand the capability of our model, we also test our model on the commonly-used OntoNotes (Gillick et al., 2014) benchmark. It is worth noting that this dataset is much smaller and has lower label multiplicity than the Ultra-Fine dataset, *i.e.*, each sample only has around 1.5 labels on average. Figure 3 shows a comparison of these two datasets.

Baselines For the Ultra-Fine dataset, we compare our model with AttentiveNER (Shimaoka et al., 2016) and the multi-task model proposed with the Ultra-Fine dataset. Note that *other models that require pre-defined type hierarchy are not applicable to this dataset*. For experiments on OntoNotes, in addition to the two neural baselines for Ultra-Fine, we compare with several existing methods that explicitly utilize the pre-defined type structures in loss functions. Namely, these methods are AFET (Ren et al., 2016a), LNR (Ren et al., 2016b) and NFETC (Xu and Barbosa, 2018).

Evaluation Metrics On Ultra-Fine, we first evaluate the mean reciprocal rank (MRR), macro precision(P), recall (R) and F1 following existing research. As P, R and F1 all depend on a chosen threshold on probabilities, we also consider a more transparent comparison using precision-recall curves. On OntoNotes, we use the standard metrics used by baseline models: accuracy, macro, and micro F1 scores.

Implementation Details Most of the model hyperparameters, such as embedding dimensions, learning rate, batch size, dropout ratios on context and mention representations are consistent with existing models. Since the mention-context matching module brings more parameters, we apply a dropout layer over the extracted feature f to avoid overfitting. We list all the hyperparameters in the appendix. Models for OntoNotes are trained with standard binary cross-entropy (BCE) losses defined on all candidate labels. When training on Ultra-Fine, we adopt the multi-task loss proposed in Choi et al. (2018) which divides the cross-entropy loss into three separate losses over different type granularities. The multi-task objective avoids penalizing false negative types and can achieve higher recalls.

5.2 Evaluation on the Ultra-Fine Dataset

We report the results on Ultra-Fine in Table 2. It is worth mentioning that our model, denoted as LABELGCN, is trained using the unlicensed training set which is smaller than the one used by compared baselines. Even though our model significantly outperforms the baselines, for a fair comparison, we first test our model using the same decision threshold (0.5) used by previous models. In terms of F1, our best model (LABELGCN) outperforms existing methods by a large margin. Compared to Choi et al. (2018), our model improves on both precision and recall significantly. Compared to the AttentiveNER trained with standard BCE loss, our model achieves much higher recall but performs worse in precision. This is due to the fact that when trained with BCE loss, the model usually retrieves only one label per sample and these types are mostly general types[5] which are easier to predict. With higher recalls or more retrieved types, achieving high precision requires being accurate on fine-grained types, which are often harder to predict.

As the precision and recall scores both rely on the decision threshold, different models or different metrics can have different optimal thresholds. As shown by the "LABELGCN + thresh tuning" entry in Table 2, with threshold tuning, our model beats baselines in all metrics. We also see that recall is usually lagging behind precision on this dataset, indicating that F1 score is mainly affected

---

[4] Choi et al. (2018) use the licensed Gigaword to build part of the dataset, while in our experiments we only use the open-sourced training set which has approximately 6M training samples.

[5] According to the results of our own implementation of BCE-trained model which achieves similar performance as AttentiveNER.

	Dev				Test			
Model	**MRR**	**P**	**R**	**F1**	**MRR**	**P**	**R**	**F1**
AttentiveNER	0.221	53.7	15.0	23.5	0.223	54.2	15.2	23.7
Choi et al. (2018)	0.229	48.1	23.2	31.3	0.234	47.1	24.2	32.0
LabelGCN	**0.250**	50.5	**28.7**	**36.6**	**0.253**	50.3	**29.2**	**36.9**
- w/o word embedding	0.245	49.4	27.8	35.6	0.249	48.7	28.3	35.8
- w/o gcn propagation	0.231	47.8	25.7	33.5	0.239	45.4	25.8	32.9
- w/o mention-context interaction	0.249	53.2	25.0	34.0	0.253	54.3	25.8	35.0
LabelGCN + *threshold-tuning*	**0.250**	**55.6**	25.4	35.0	**0.253**	**54.8**	25.9	35.1

Table 2: Comparison with baseline models on the Ultra-Fine dataset. *Threshold-tuning* gives better performance on all metrics compared to both baselines.

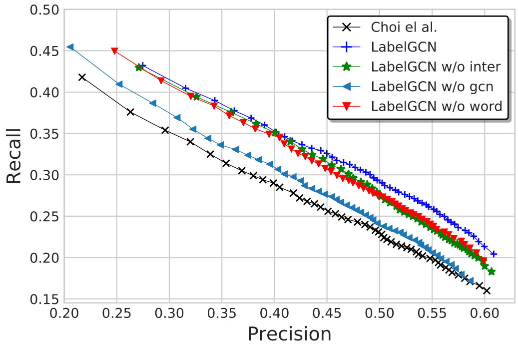


Figure 4: Precision-recall curves on Ultra-Fine. The trivial point derived by threshold 0 is omitted here.

Model	F1-pronouns	F1-else
Choi et al. (2018)	35.8	32.0
Choi et al. (2018) + inter	38.2 (↑ 2.4)	32.8
LabelGCN w/o inter	38.6	36.8
LabelGCN	39.3 (↑ 0.7)	36.5

Table 3: Decomposed validation performance on pronouns and the other entities. Each entry is obtained using the best threshold among the 50 equal-interval thresholds. The corresponding PR curves can be found in the appendix (Figure 5).

by the recall and tuning towards recall can usually lead to higher F1 scores. For more transparent comparisons, we show the precision-recall curves in Figure 4. These data points are based on the validation performance given by 50 equal-interval thresholds between 0 and 1. We can see there is a clear margin between our model and the multi-task baseline method (LabelGCN vs Choi et al.).

5.3 Ablation Studies

To quantify the effect of different model components, we report the performance of model variants in Table 2 and Figure 4. We can clearly see that the graph convolution layer is the most essential component. The information provided by word embedding is useful and can further improve both precision and recall. Although Table 2 seems to indicate the interaction module decreases the precision, we can see from Figure 4 that with a proper threshold, the enhanced interaction actually improves both precision and recall. In term of this, we recommend future research to use PR curves for more accurate model analysis.

5.4 Fine-Grained Performance for Pronouns

As discussed in Section 4.2, the mention representation of pronouns provide limited information about fine-grained types. We investigate the effect of the enhanced mention-context interaction by analyzing the decomposed performance on pronouns and other kinds of entities. From the results in Table 3, we can see that the enhanced interaction offers consistent improvements over pronouns entities and also maintains the performance on other kinds of entities.

5.5 Qualitative Analysis

To gain insights on the improvements provided by our model, we manually analyze 100 error cases[6] of the baseline model (Choi et al. (2018) with threshold 0.5) and see if our model can generate high-quality predictions. We first observe that many errors actually results from incomplete annotations. This suggests models' precision scores are often underestimated in this dataset. We discuss several typical error cases shown in Table 4 and list more samples in the appendix (Table 7).

A key observation is that while the baseline model tends to make inconsistent predictions (see examples 1, 2, 3), our model can avoid predicting such inconsistent type pairs. This indeed validates our model's ability to encode label correlations. We also notice that our model is more sensitive to gender information indicated by pronouns, while

---

[6] The baseline model achieves the lowest precision on these 100 samples.

1) Context	**Today**, Taiwan is manifesting the elegance of a democratic island, once again attracting global attention, as the people on this land create a new page in our history.
Groundtruth	*time, date, day, today, present*
Prediction	**Baseline**: {day†, person*, organization, religion} **Ours**: {day}
2) Context	**A gigantic robot** emerges, emitting a sound that paralyzes humans and disrupts all electrical systems in New York City.
Groundtruth	*object, device, machine, mechanism*
Prediction	**Baseline**: {object, person†, robot*} **Ours**: {object, robot}
3) Context	**He** also has been accused of genocide in Bosnia and other war crimes in Croatia, but the date to try those two indictments together has not been set.
Groundtruth	*person*
Prediction	**Baseline**:{person, god†, title, criminal*} **Ours**: {person, politician, criminal, male, prisoner}
4) Context	**Her** status was uncertain for Wimbledon, which begins June 23.
Groundtruth	*person, athlete, adult, player, professional, tennis player, contestant*
Prediction	**Baseline**: {person, female, woman, spouse} **Ours**: {person, artist, female, woman}
5) Context	For eight years **he** treated thousands of wounded soldiers of the armed forces led by the CPC.
Groundtruth	*person, doctor, caretaker, nurse*
Prediction	**Baseline**: {person, soldier, suspect, serviceman} **Ours**: {person, soldier, man}

Table 4: Qualitative analysis of validation samples. We use different colors and subscript symbols to mark inconsistencies. The bottom two rows show error cases for both models.

Model	Accuracy	Macro-F1	Micro-F1
AttentiveNER	51.7	71.0	64.9
AFET	55.1	71.1	64.7
LNR	57.2	71.5	66.1
NFETC	**60.2**	76.4	70.2
Choi et al. (2018)	59.5	76.8	71.8
LABELGCN	59.6	**77.8**	**72.2**

Table 5: Results on OntoNotes. Upper rows show the results of baselines that explicitly use the hierarchical type structures.

the baseline model sometimes holds the gender-indicating predictions and predict other types, our model predicts the gender-indicating types more often (examples 3, 4, 5). We conjecture that our model learns this easy way to maintain precision.

For cases that both models fail, some of them actually require background knowledge (example 4) to make accurate predictions. Another typical case is that both models predict some other entities in the context (example 5). We think this potentially results from the data bias introduced by the head-word supervision.

5.6 Evaluation on OntoNotes

To better understand the requirements for applying our model, we further evaluate on the OntoNotes dataset. Here we do not apply the proposed mention-context matching module as this dataset does not include any pronoun entities. To obtain more reliable co-occurrence statistics, we use the augmented training data released by Choi et al. (2018). However, since the training set is still much smaller than that of the Ultra-Fine dataset, the derived co-occurrence statistics are relatively noisy and might introduce undesired bias. We thus add an additional residual connection to our graph convolution layer, which allows the model to selectively use co-occurrence statistics. This indeed gives us improvements over previous state-of-the-arts, as shown in Table 5. However, compared to Ultra-Fine, the margin of the improvement is smaller. In view of the key differences of these two datasets, we highlight two key requirements for our proposed model to offer substantial improvements. First, there should be a large-scale training set so that the derived co-occurrence statistics can reasonably reflect the true label correlations. Second, the samples themselves should also have higher label multiplicity. In fact, most of the samples in OntoNotes only have 1 or 2 labels. This property actually alleviates the need for models to capture label dependencies.

6 Conclusion

In this paper, we present an effective method to impose label-relational inductive bias on fine-grained entity typing models. Specifically, we utilize a graph convolution layer to incorporate type co-occurrence statistics and word-level type similarities. This layer implicitly captures the label correlations in the latent vector space. Along with an attention-based mention-context matching module, we achieve significant improvements over previous methods on a large-scale dataset. As our method does not require external knowledge about the label structures, we believe our method is general enough and has the potential to be applied to other multi-label tasks with plain-text labels.

Acknowledgement

This research was supported in part by DARPA Grant D18AP00044 funded under the DARPA YFA program. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. *arXiv preprint arXiv:1807.04905*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. 2006. Fine-grained named entity recognition using conditional random fields for question answering. In *Asia Information Retrieval Symposium*, pages 581–587. Springer.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maxim Rabinovich and Dan Klein. 2017. Fine-grained entity typing with high-multiplicity assignments. *arXiv preprint arXiv:1704.07751*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1825–1834. ACM.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74. Association for Computational Linguistics.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1271–1280. Association for Computational Linguistics.

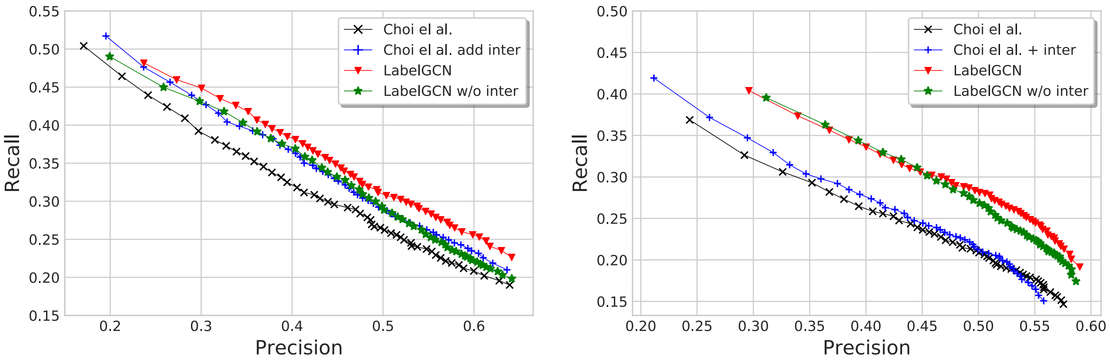
Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. *arXiv preprint arXiv:1803.03378*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2016. Noise mitigation for neural entity typing and relation extraction. *arXiv preprint arXiv:1612.07495*.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 149–159.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 291–296.

A Appendix



a) PR curves of pronoun entities

a) PR curves of non-pronoun entities

Figure 5: Precision-recall curves showing the decomposed results on pronoun and non-pronoun entity mentions. The enhanced mention-context interaction can consistently offer improvements for pronoun entity mentions while maintaining the performance for non-pronoun entity mentions.

learning rate	0.001
batch size	1000
position embedding size	50
dropout on context $\mathcal{C}$	0.2
dropout on mention $\mathcal{M}$	0.5
hidden dimension of LSTM	100
dropout on fused feature f (Ultra-Fine)	0.2
dropout on fused feature f (OntoNotes)	0.3

Table 6: Hyperparameters used in our experiments

Context	**They** have been asked to appear in court to face the charge on Feb. 3.
Groundtruth	*person, defendant, suspect, accused*
Prediction	**Baseline**:{person, engineer, officer, policeman†, prisoner*, married, serviceman} **Ours**:{person}
Context	"**It** is truly a war crime," she added.
Groundtruth	*event, crime, issue, offense, transgression, atrocity*
Prediction	**Baseline**: {internet†, event, art*, writing} **Ours**: {law}
Context	**She** added that Israeli military personnel had conducted a medical examination after the shooting in concert with Palestinian medics.
Groundtruth	*person*
Prediction	**Baseline**: {person†, art*, writing, convict, felon} **Ours**: {person, female, woman}
Context	The monument is located in Pioneer Park Cemetery in the Convention Center District of downtown Dallas, Texas, **USA**, next to the Dallas Convention Center and Pioneer Plaza.
Groundtruth	*location, place, country, area, nation, region*
Prediction	**Baseline**: {location†, person*, agency, artist, cemetery, country, language, title, republic} **Ours**:{nationality, location, place, country, area, license, nation}
Context	**The committee** undertook its work on Saturday 16/2/1426 A. H . The following is noteworthy :
Groundtruth	*group, organization, agency, company, institution, administration, body, management, party*
Prediction	**Baseline**: {committee†, person*, organization, government} **Ours**: {group, government, committee}
Context	**They** are accused of helping Libya develop a nuclear weapons programme and were alleged to have been in contact with Abdul Qadeer Khan , the disgraced father of Pakistan 's nuclear programme.
Groundtruth	*group, terrorist*
Prediction	**Baseline**: {military, person†, group, country*} **Ours**: {person, politician, prisoner, serviceman}
Context	**It** also marked the first major roundup of Islamist leaders by a government eager to demonstrate its commitment to the anti-terror fight waged by the United States.
Groundtruth	*event, consequence*
Prediction	**Baseline**:{internet†, event, art*, writing} **Ours**:{event}
Context	If you have ever watched a keynote speech by **Steve Jobs**, you know that he was the best of the best in launching a product.
Groundtruth	*person, adult, businessman, celebrity, professional*
Prediction	**Baseline**: {person, artist, athlete†, author, musician*} **Ours**:{person}
Context	**They** dined together, this time in Benedict's house, before the pope was driven back to his temporary residence in Regensburg 's St Wolfgang Seminary.
Groundtruth	*adult, man, supporter, serviceman*
Prediction	**Baseline**: {person, adult, female, woman} **Ours**: {person}
Context	Topic : **I** am grateful to the University of Science and Technology
Groundtruth	*person, individual, student*
Prediction	**Baseline**:{person, politician, employee, leader, minister, traveler, announcer, clergyman} **Ours**:{person, student}
Context	"**I** didn't think the speech was that long," Pataki said.
Groundtruth	*person, speaker*
Prediction	**Baseline**: {person, actor, politician, spokesperson, woman} **Ours**: {person, adult}
Context	"This is touching our troops," **she** said.
Groundtruth	*person, adult, female, reporter, woman*
Prediction	**Baseline**: {person, politician, official, spokesperson, communicator} **Ours**: {female, official, reporter, strategist, communicator, officeholder}

Table 7: More sample predictions. Our model is able to give more accurate type predictions and also reduce the inconsistency in the output type set.

Improving Event Coreference Resolution by Learning Argument Compatibility from Unlabeled Data

Yin Jou Huang[1], **Jing Lu**[2], **Sadao Kurohashi**[1], **Vincent Ng**[2]
[1]Graduate School of Informatics, Kyoto University
[2]Human Language Technology Research Institute, University of Texas at Dallas
`huang@nlp.ist.i.kyoto-u.ac.jp, kuro@i.kyoto-u.ac.jp`
`{ljwinnie,vince}@hlt.utdallas.edu`

Abstract

Argument compatibility is a linguistic condition that is frequently incorporated into modern event coreference resolution systems. If two event mentions have incompatible arguments in any of the argument roles, they cannot be coreferent. On the other hand, if these mentions have compatible arguments, then this may be used as information toward deciding their coreferent status. One of the key challenges in leveraging argument compatibility lies in the paucity of labeled data. In this work, we propose a transfer learning framework for event coreference resolution that utilizes a large amount of unlabeled data to learn the argument compatibility between two event mentions. In addition, we adopt an interactive inference network based model to better capture the (in)compatible relations between the context words of two event mentions. Our experiments on the KBP 2017 English dataset confirm the effectiveness of our model in learning argument compatibility, which in turn improves the performance of the overall event coreference model.

1 Introduction

Events are essential building blocks of all kinds of natural language text. An event can be described several times from different aspects in the same document, resulting in multiple surface forms of event mentions. The goal of event coreference resolution is to identify event mentions that correspond to the same real-world event. This task is critical for natural language processing applications that require deep text understanding, such as storyline extraction/generation, text summarization, question answering, and information extraction.

[1] KMT to **elect**$_{m1}$ new party chief

[2] The ruling Chinese Kuomingtang Party (KMT) in Taiwan is scheduled to **elect**$_{m2}$ a new chairperson in January 2012, it was decided at the top-level KMT **meeting**$_{m3}$ on Wednesday.

[3] KMT chairman Ma Ying-jeou stressed fair paly during the **election**$_{m4}$ at Wednesday's **meeting**$_{m5}$, saying guest-feting or gift-giving must be banned during the process.

[4] Ma was twice **elected**$_{m6}$ the party's chairman, first in 2005 and then in 2009.

ev_1: m_1, m_2, m_4
ev_2: m_3, m_5
ev_3: m_6

Figure 1: A document with three events described in six event mentions. Coreferent event mentions are highlighted with the same color.

Figure 1 shows a document consisting of three events described by six different event mentions. Among these event mentions, m_1, m_2 and m_4 are coreferent, since they all correspond to the event of the KMT party electing a new party chief. Similarly, m_3 and m_5 are also coreferent, while m_6 is not coreferent with any other event mentions.

An event mention consists of a trigger and zero or more arguments. The trigger of an event mention is the word/phrase that is considered the most representative of the event, such as the word *meeting* for m_3 or the word *elected* for m_6. Triggers of coreferent event mentions must be related, that is, they should describe the same type of events. For example, m_1 and m_3 cannot be coreferent, since their trigger words — *elect* and *meeting* — are not related.

Arguments are the participants of an event, each having its role. For example, *KMT* is the AGENT-argument and *new party chief* is the PATIENT-argument of m_1. Argument compatibility is an important linguistic condition for determining the coreferent status between two event mentions. Two arguments are incompatible if they do not correspond to the same real-world entity when they are expressed in the same level of specificity;

Proceedings of NAACL-HLT 2019, pages 785–795
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

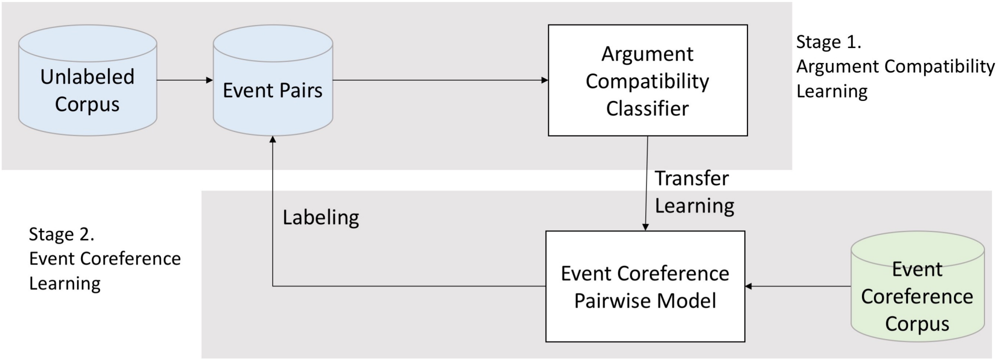


Figure 2: System overview.

otherwise, they are compatible. For example, a pair of TIME-arguments — *Wednesday* and *2005* — which are expressed in different level of specificity, are considered compatible. If two event mentions have incompatible arguments in some specific argument roles, they cannot be coreferent. For example, m_2 and m_6 are not coreferent since their TIME-arguments — *January 2012* and *2005* — and their PATIENT-arguments — *a new chairperson* and *Ma* — are incompatible. On the other hand, coreferent event mentions can only have compatible arguments. For example, m_3 and m_5 both have *Wednesday* as TIME-arguments. In this example, argument compatibility in the TIME argument role is a strong hint suggesting their coreference.

Despite its importance, incorporating argument compatibility into event coreference systems is challenging due to the lack of sufficient labeled data. Many existing works have relied on implementing argument extractors as upstream components and designing argument features that capture argument compatibility in event coreference resolvers. However, the error introduced in each of the steps propagates through these resolvers and hinders their performance considerably.

In light of the aforementioned challenge, we propose a framework for transferring argument (in)compatibility knowledge to the event coreference resolution system, specifically by adopting the interactive inference network (Gong et al., 2018) as our model structure. The idea is as follows. First, we train a network to determine whether the corresponding arguments of an event mention pair are compatible on automatically labeled training instances collected from a large unlabeled news corpus. Second, to transfer the knowledge of argument (in)compatibility to an event coreference resolver, we employ the network (pre)trained in the previous step as a starting point and train it to determine whether two event mentions are coreferent on manually labeled event coreference corpora. Third, we iteratively repeat the above two steps, where we use the learned coreference model to relabel the argument compatibility instances, retrain the network to determine argument compatibility, and use the resulting pretrained network to learn an event coreference resolver. In essence, we mutually bootstrap the argument (in)compatibility determination task and the event coreference resolution task.

Our contributions are three-fold. First, we utilize and leverage the argument (in)compatibility knowledge acquired from a large unlabeled corpus for event coreference resolution. Second, we employ the interactive inference network as our model structure to iteratively learn argument compatibility and event coreference resolution. Initially proposed for the task of natural language inference, the interactive inference network is suitable for capturing the semantic relations between word pairs. Experimental results on the KBP coreference dataset show that this network architecture is also suitable for capturing the argument compatibility between event mentions. Third, our model achieves state-of-the-art results on the KBP 2017 English dataset (Ellis et al., 2015, 2016; Getman et al., 2017), which confirms the effectiveness of our method.

2 Related Work

Ablation experiments conducted by Chen and Ng (2013) provide empirical support for the usefulness of event arguments for event coreference resolution. Hence, it should not be surprising that, with just a few exceptions (e.g., Sangeetha and

	event mention	DATE-compatibility with m_a
m_a	The result of the **election** last October surprised everyone.	-
m_1	He was **elected** as president in 2005.	no
m_2	The presidential **election** took place on October 20th.	yes
m_3	The opposition party won the **election**.	yes

Table 1: Examples of NER-based sample filtering. The phrases tagged as DATE are underlined, and the trigger words are boldfaced.

Arock (2012); Araki and Mitamura (2015); Lu and Ng (2017)), argument features have been extensively exploited in event coreference systems to capture the argument compatibility between two event mentions. Basic features such as the number of overlapping arguments and the number of unique arguments, and a binary feature encoding whether arguments are conflicting have been proposed (Chen et al., 2009; Chen and Ji, 2009; Chen and Ng, 2016). More sophisticated features based on different kinds of similarity measures have also been considered, such as the surface similarity based on Dice coefficient and the WuPalmer WordNet similarity between argument heads (McConky et al., 2012; Cybulska and Vossen, 2013; Araki et al., 2014; Liu et al., 2014; Krause et al., 2016). However, these features are computed using either the outputs of event argument extractors and entity coreference resolvers (Ahn, 2006; Chen and Ng, 2014, 2015; Lu and Ng, 2016) or semantic parsers (Bejan and Harabagiu, 2014; Yang et al., 2015; Peng et al., 2016) and therefore suffer from serious error propagation issues (see Lu and Ng (2018)). Several previous works proposed joint models to address this problem (Lee et al., 2012; Lu et al., 2016), while others utilized iterative methods to propagate argument information (Liu et al., 2014; Choubey and Huang, 2017) in order to alleviate this issue. However, all of these methods still rely on argument extractors to identify arguments and their roles.

3 Method

Our proposed transfer learning framework consists of two learning stages, the pretraining stage of an argument compatibility classifier and the fine-tuning stage of an event coreference resolver (Figure 2). We provide the details of both stages in sections 3.1 and 3.2, and describe the iterative strategy combining the two training stages in section 3.3. Details on the model structure are covered in section 3.4.

3.1 Argument Compatibility Learning

In the pretraining stage, we train the model as an argument compatibility classifier with event mentions extracted from a large unlabeled news corpus.

Task definition Given a pair of event mentions (m_a, m_b) with related triggers, predict whether their arguments are compatible or not.

Here, an event mention is represented by a trigger word and the context words within an n-word window around the trigger.

Related trigger extraction We analyze the event coreference resolution corpus and extract trigger pairs that are coreferent more than k times in the training data. We define these trigger pairs to be *related* triggers in our experiment. In this work, we set k to 10. Table 2 shows some examples of related triggers with high counts.

trigger pair	count
kill - death	86
shoot - shooting	35
retire - retire	34
demonstration - protest	30

Table 2: Examples of related triggers.

If the triggers of an event mention pair are related, their coreferent status cannot be determined by looking at the triggers alone, and this is the case in which argument compatibility affects the coreferent status most directly. Thus, we focus on the event mention pairs with related triggers in the pretraining stage of argument compatibility learning.

Compatible samples extraction From each document, we extract event mention pairs with related triggers and check whether the following conditions are satisfied:

1. DATE-compatibility (Table 1):
   First, we perform named entity recognition

(NER) on the context words. If both event mentions have phrases tagged as DATE in the context, these two phrases must contain at least one overlapping word. If there are multiple phrases tagged as DATE in the context, only the phrase closest to the trigger word is considered.

2. PERSON-compatibility: Similar to 1.
3. NUMBER-compatibility: Similar to 1.
4. LOCATION-compatibility: Similar to 1.
5. Apart from function words, the ratio of overlapping words in their contexts must be under 0.3 for both event mentions. We add this constraint in order to remove trivial samples of nearly identical sentences.

Conditions 1–4 are heuristic filtering rules based on NER tags, which aim to remove samples with apparent incompatibilities. Here, we consider four NER types — DATE, PERSON, NUMBER, and LOCATION — because these types of words are the most salient types of incompatibility that can be observed between event mentions. Condition 5 aims to remove event mention pairs that are "too similar". We add this condition because we do not want our model to base its decisions on the number of overlapping words between the event mentions.

We collect event mention pairs satisfying all the above conditions as our initial set of compatible samples.

Incompatible sample extraction From different documents in the corpus, we extract event mentions with related triggers and check whether the following conditions are satisfied:

1. The creation date of the two documents must be at least one month apart.
2. Apart from the trigger words and the function words, the context of the event mentions must contain at least one overlapping word.

In the unlabeled news corpus, articles describing similar news events are sometimes present. Thus, we use condition 1 to roughly assure that the event mention pairs extracted are not coreferent. Mention pairs extracted from the same document tend to contain overlapping content words, so to prevent our model to make decisions based on the existence of overlapping words, we add condition 2 as a constraint.

We collect event mention pairs satisfying all the above conditions as our initial set of incompatible samples.

Argument compatibility classifier With the initial set of compatible and incompatible samples acquired above, we train a binary classier to distinguish between samples of the two sets.

3.2 Event Coreference Learning

In the fine-tuning stage, we adapt the argument compatibility classifier on the labeled event coreference data to a mention-pair event coreference model.

3.2.1 Event Mention Detection

Before proceeding to the task of event coreference resolution, we have to identify the event mentions in the documents. We train a separate event mention detection model to identify event mentions along with their subtypes.

We model event mention detection as a multi-class classification problem. Given a candidate word along with its context, we predict the subtype of the event mention triggered by the word. If the given candidate word is not a trigger, we label it as NULL. We select the words that have appeared as a trigger at least once in the training data as candidate trigger words. We do not consider multi-word triggers in this work.

Given an input sentence, we first represent each of its comprising words by the concatenation of the word embedding and the character embedding of the word. These representation vectors are fed into a bidirectional LSTM (biLSTM) layer to obtain the hidden representation of each word.

For each candidate word in the sentence, its hidden representation is fed into the inference layer to predict the class label. Since the class distribution is highly unbalanced, with the NULL label significantly outnumbering all the other labels, we use a weighted softmax at the inference layer to obtain the probability of each class. In this work, we set the weight to 0.1 for the NULL class label and 1 for all the other class labels.

Intuitively, candidate triggers with the same surface form in the same document tend to have the same class label. However, it is difficult to model this consistency since our model operates at the sentence level. Thus, we account for this consistency across sentences by the following post-processing step: If a candidate word is assigned the NULL label but more than half of the candidates sharing the same surface form is detected as triggers of a specific subtype, then we change the label to this given subtype. Also, we disregard event mentions with types *contact*, *movement* and *transaction* in this post-processing step, since the subtypes under these three types do not have a good consistency across different sentences in the same document.

3.2.2 Mention-Pair Event Coreference Model

With the argument compatibility classifier trained in the previous stage, we use the labeled event coreference corpus to fine-tune the model into an event coreference resolver. We design the event coreference resolver to be a mention-pair model (Soon et al., 2001), which takes a pair of event mentions as the input and outputs the likelihood of them being coreferent.

With the pairwise event coreference predictions, we further conduct best-first clustering (Ng and Cardie, 2002) on the pairwise results to build the event coreference clusters of each document. Best-first clustering is an agglomerative clustering algorithm that links each event mention to the antecedent event mention with the highest coreference likelihood given the likelihood is above an empirically determined threshold.

3.3 Iterative Relabeling Strategy

Previously, we collected a set of compatible event mentions from the same document with simple heuristic filtering. Despite this filtering step, the initial compatible set is noisy. Here, we introduce an iterative relabeling strategy to improve the quality of the compatible set of event mentions.

First, we calculate the coreference likelihood of the event mentions in the initial compatible set. Mention pairs with a coreference likelihood above threshold θ_M are added to the new compatible set. On the other hand, mention pairs with a coreference likelihood below θ_m are added to the initial incompatible set to form the new incompatible set. With the new compatible and incompatible sets, we can start another iteration of transfer learning to train a coreference resolver with improved quality. In this work, we set θ_M to 0.8 and θ_m to 0.2.

3.4 Model Structure

We adopt an interactive inference network as the model structure of our proposed method (Figure 3). A qualitative analysis of an interactive inference network shows that it is good at capturing word overlaps, antonyms and paraphrases between sentence pairs (Gong et al., 2018). Thus, we believe this network is suitable for capturing the argument compatibility between two event mentions. The model consists of the following components:

Model inputs The input to the model is a pair of event mentions (m_a, m_b), with m_a being the antecedent mention of m_b:

$$\begin{aligned} m_a &= \{w_a^1, w_a^2, ..., w_a^N\} \\ m_b &= \{w_b^1, w_b^2, ..., w_b^N\} \end{aligned} \tag{1}$$

Each event mention is represented by a sequence of N tokens consisting of one trigger word and its context. Here, we take the context to be the words within an n-word window around the trigger. In this work, n is set to 10.

Embedding layer We represent each input token by the concatenation of the following components:

Word embedding The word representation of the given token. We use pretrained word vectors to initialize the word embedding layer.

Character embedding To identify (in)compatibilities regarding person, organization or location names, the handling of out-of-vocabulary (OOV) words is critical.

Adding character-level embeddings can alleviate the OOV problem (Yang et al., 2017). Thus, we apply a convolutional neural network over the comprising characters of each token to acquire the corresponding character embedding.

POS and NER one-hot vectors One-hot vectors of the part-of-speech (POS) tag and NER tag.

Exact match A binary feature indicating whether a given token appears in the context of both event mentions. This feature is proved useful for several NLP tasks operating on pairs of texts (Chen et al., 2017; Gong et al., 2018; Pan et al., 2018).

Trigger position We encode the position of the trigger word by adding a binary feature to indicate whether a given token is a trigger word.

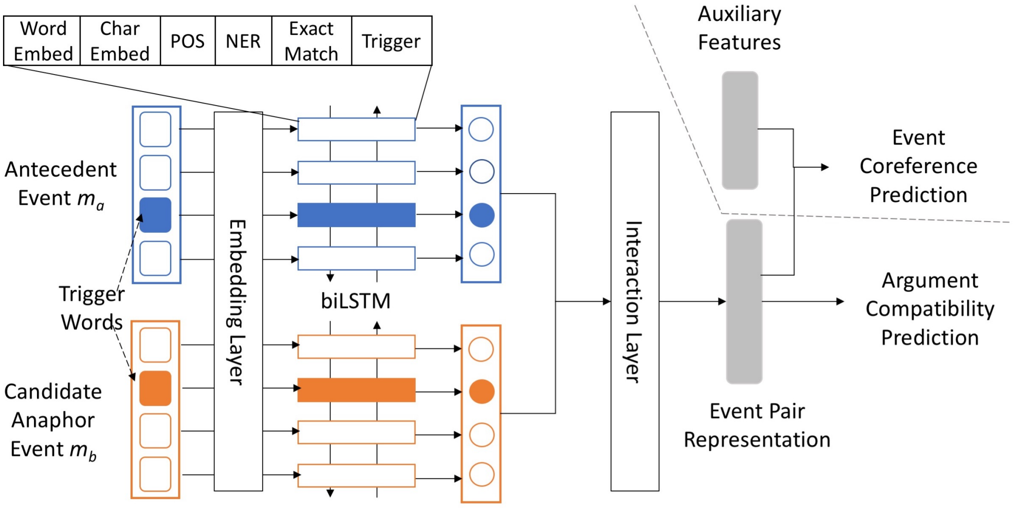


Figure 3: Model structure.

Encoding layer We pass the sequence of embedding vectors into a biLSTM layer (Hochreiter and Schmidhuber, 1997), resulting in a sequence of hidden vectors of size $|h|$:

$$\begin{aligned} h_a^i &= biLSTM(emb(w_a^i), h_a^{i-1}) \\ h_b^i &= biLSTM(emb(w_b^i), h_b^{i-1}) \end{aligned} \tag{2}$$

where $emb(w)$ is the embedding vector of token w.

Interaction layer The interaction layer captures the relations between two event mentions based on the hidden vectors h_a and h_b. The interaction tensor I, a 3-D tensor of shape (N, N, $|h|$), is calculated by taking the pairwise multiplication of the corresponding hidden vectors:

$$I_{ij} = h_a^i \circ h_b^j \tag{3}$$

Finally, we apply a multi-layer convolutional neural network to extract the event pair representation vector f_{ev}.

Inference layer In the pretraining stage, we feed f_{ev} to a fully-connected inference layer to make a binary prediction of argument compatibility.

As for the fine-tuning stage, we concatenate an auxiliary feature vector f_{aux} to f_{ev} before feeding it into the inference layer. f_{aux} consists of two features, a one-hot vector that encodes the sentence distance between the two event mentions and the difference of the word embedding vectors of the two triggers.

4 Evaluation

4.1 Experimental Setup

4.1.1 Corpora

We use English Gigaword (Parker et al., 2009) as the unlabeled corpus for argument compatibility learning. This corpus consists of the news articles from five news sources, each annotated with its creation date.

As for event coreference resolution, we use the English portion of the KBP 2015 and 2016 datasets (Ellis et al., 2015, 2016) for training, and the KBP 2017 dataset (Getman et al., 2017) for evaluation. The KBP datasets comprise news articles and discussion forum threads. The KBP 2015, 2016, and 2017 corpora contain 648, 169, and 167 documents, respectively. Each document is annotated with event mentions of 9 types and 18 subtypes, along with the coreference clusters of these event mentions.

4.1.2 Implementation Details

Preprocessing We use the Stanford CoreNLP toolkit (Manning et al., 2014) to perform preprocessing on the input data.

Network structure Each word embedding is initialized with the 300-dimensional pretrained GloVe embedding (Pennington et al., 2014). The character embedding layer is a combination of an 8-dimensional embedding layer and three 1D convolution layers with a kernel size of 5 with 100 filters. The size of the biLSTM layer is 200. The maximum length of a word is 16 characters; shorter words are padded with zero and longer

	MUC	B^3	$CEAF_e$	BLANC	AVG-F
biLSTM (standard)	29.49	43.15	39.91	24.15	34.18
biLSTM (transfer)	33.84	42.91	38.39	26.59	35.43
Interact (standard)	31.12	42.84	39.01	24.99	34.49
Interact (transfer)	34.28	42.93	39.95	32.12	36.24
Interact (transfer, 2^{nd} iter)	35.66	43.20	40.02	32.43	**36.75**
Interact (transfer, 3^{rd} iter)	36.05	43.07	39.69	28.06	36.72
Jiang et al. (2017)	30.63	43.84	39.86	26.97	35.33

Table 3: Event coreference resolution results of our proposed system, compared with the biLSTM baseline model and the current state-of-the-art system.

words are cropped. For the interaction layer, we use convolution layers with a kernel size of 3 in combination with max-pooling layers. The size of the inference layer is 128. Sigmoid activation is used for the inference layer, and all other layers use ReLU as the activation function.

Event mention detection model For word embeddings, we use the concatenation of a 300-dimensional pretrained GloVe embedding and the 50-dimensional embedding proposed by Turian et al. (2010). The character embedding layer is a combination of an 8-dimensional embedding layer and three 1D convolution layers with kernel sizes of 3, 4, 5 with 50 filters.

4.1.3 Evaluation Metrics

We follow the standard evaluation setup adopted in the official evaluation of the KBP event nugget detection and coreference task. This evaluation setup is based on four distinct scoring measures — MUC (Vilain et al., 1995) , B^3 (Bagga and Baldwin, 1998), $CEAF_e$ (Luo, 2005) and BLANC (Recasens and Hovy, 2011) — and the unweighted average of their F-scores (AVG-F). We use AVG-F as the main evaluation measure when comparing system performances.

4.2 Results

We present the experimental results on the KBP 2017 corpus in Table 3. In the following, we compare the performance of methods with different network architectures and experimental settings.

Comparison of network architectures We compare the results of the interactive inference network (Interact) with the biLSTM baseline model (biLSTM).

The biLSTM baseline model does not have the interaction layer. Instead, the last hidden vectors of the biLSTM layer are concatenated and fed into the inference layer directly.

When trained solely on the event coreference corpus (standard), the model with the interactive inference network performs slightly better than the biLSTM baseline model, as shown in rows 1 and 3. However, with an additional pretraining step of argument compatibility learning (transfer), the interact inference network outperforms the biLSTM baseline model by a considerable margin, as shown in rows 2 and 4. We conclude that the interactive inference network can better capture the complex interactions between two event mentions, accounting for the difference in performance.

Effect of transfer learning Regardless of the network structure, we observe a considerable improvement in performance by pretraining the model as an argument compatibility classifier. The biLSTM baseline model achieves an improvement of 1.25 points in AVG-F by doing transfer learning, as can be seen in rows 1 and 2. As for the interactive inference network, an improvement of 1.75 points in AVG-F is achieved, as can be seen in rows 3 and 4. These results provide suggestive evidence that our proposed transfer learning framework, which utilizes a large unlabeled corpus to perform argument compatibility learning, is effective.

Effect of iterative relabeling We achieve another boost in performance by using the trained event coreference resolver to relabel the training samples for argument compatibility learning. The best result is achieved after two iterations (row 5) with an improvement of 2.26 points in AVG-F compared to the standard interactive inference network (row 3). However, we are not able to obtain further gains with more iterations of relabeling (row 6). We speculate that the difference in event coreference model predictions across different iterations is not big enough to have a perceivable impact, but additional experiments are needed to determine the reason.

Type	Event Mention Pair	Gold	System
Explicit	m_1: ... the building where 13 people were **killed** will be razed, and a memorial ... m_2: In that case, George Hennard **killed** 23 people at a Luby 's restaurant, ...	non-coref	non-coref
Explicit	m_1: Ten relatives of the victims arrived at the airport Sunday before **traveling** to the city of Jiangshan. m_2: On Monday , the victims' relatives **went** to the Jiangshan Municipal Funeral Parlor.	non-coref	non-coref
Implicit	m_1: ... a young woman **protester** was brutally slapped while she was demonstrating ... m_2: ... explain why a women **protester** in her 60s was beaten up by policemen ...	non-coref	coref
Implicit	m_1: She died from a brain hemorrhage on July 10, 2003, ... m_2: ... has denied killing his second wife, whom he says **died** in a car accident.	non-coref	non-coref
Implicit	m_1: Nationwide demonstrations held in France to **protest** gay marriage. m_2: ... to **protest** against the country's plan to legalize same-sex marriage.	coref	coref
General	m_1: ... Connecticut elementary school **shooting** has reignited the debate over gun control. m_2: Gun supporters hold that people, not guns, are to blame for the **shootings**.	non-coref	coref
General	m_1: Industrial accidents have injured and **killed** Foxconn workers, and the company also experienced ... m_2: ... explosion in May 2011 at Foxconn 's Chengdu factory **killed** three workers ...	non-coref	non-coref

Table 4: Examples of event pairs with related triggers. Trigger words are boldfaced, and words with (in)compatibility information are colored in blue.

Comparison with the state of the art Comparing row 5 and 7, we can see that our method outperforms the previous state-of-the-art model (Jiang et al., 2017) by 1.42 points in AVG-F.

5 Discussion

In this section, we conduct a qualitative analysis of the outputs of our best-performing system (the Interact (transfer, 2^{nd} iter) system in Table 3) on the event coreference dataset and the unseen event mention pairs extracted from the unlabeled corpus.

5.1 Compatibility Classification

We focus on the samples with related triggers having either compatible or incompatible arguments (Table 4). These samples can be roughly classified into the following categories:

Explicit argument compatibility The existence of identical/distinct time phrases, numbers, location names or person names in the context is the most explicit form of (in)compatibility.

For these event pairs, the existence of identical/distinct phrases with the same NER type is a direct clue toward deciding their coreferent status. Making use of this nature, we perform filtering on the set of compatible samples acquired from the unlabeled corpus in order to remove samples with explicit incompatibility.

Our model can recognize this type of (in)compatibility with a relatively high accuracy. Both examples shown in Table 4 are predicted correctly.

Implicit argument compatibility Event pairs with implicit (in)compatible arguments require external knowledge to resolve.

We present three examples in Table 4. In the first example, the knowledge that a woman *in her 60s* is generally not referred to as being *young* is required to determine the incompatibility. Similarly, the knowledge that both *brain hemorrhage* and *car accident* are causes of people's death are required to classify the second example correctly.

While the performance on samples with implicit (in)compatibility is not as good as that on samples with explicit (in)compatibility, our system is able to capture implicit (in)compatibility to some extent. We believe that this type of (in)compatibility is difficult to be captured with the argument features that are designed based on the outputs of argument extractors and entity coreference resolvers, and that the ability to resolve implicit (in)compatibility contributes largely to our system's performance improvements.

General-specific incompatibilities Event mentions describing general events pose special challenges to the task of event coreference resolution.

In Table 4, we present two typical examples of this category. In the first example, the second event mention does not refer to any specific shooting event in the real world, in contrast to the first event mention, which describes a specific school shooting event. Similarly for the second example, where the first event mention depicts a general event and the second event mention depicts a specific one.

General event mentions typically have few or even no arguments and modifiers, making the identification of non-coreference relations very challenging. Since we cannot rely on argument compatibility, a deeper understanding of the semantics of the event mentions is needed. General

	Event Mention Pair	Type	System
I	m_1: What would have happened if Steve Jobs had never **left** Apple ...	-	-
	m_2^a: ...in the state that is today if John hadn't **left**.	Explicit	non-coref
	m_2^b: ...in the state that is today if she hadn't **left**.	Implicit	non-coref
	m_2^c: ...in the state that is today if he hadn't **left**.	Implicit	coref
II	m_1: Police arrest 6 men for **gangraping** housewife in northern India.	-	-
	m_2^a: Indian police have arrested six men for allegedly **gangraping** a 29-year-old housewife ...	Explicit	coref
	m_2^b: Indian police have arrested six men for allegedly **gangraping** a woman ...	Implicit	coref
	m_2^c: Indian police have arrested six men for allegedly **gangraping** a medical student ...	Implicit	non-coref
III	m_1: Nationwide demonstrations in France to **protest** gay marriage.	-	-
	m_2^a: ...took to the streets across the country to **protest** against the country's plan to legalize same-sex marriage.	Implicit	coref
	m_2^b: ...took to the streets across the country to **protest** against the contentious citizenship amendment bill.	Implicit	non-coref

Table 5: Case study on manually-generated event mention pairs. Trigger words are boldfaced, and the target arguments are colored in blue.

event mentions account for a considerable fraction of our system's error, since they are quite pervasive in both news articles and discussion forum threads.

5.2 Case Study

To better understand the behavior of our system, we perform a case study on manually-generated event pairs. Specifically, for a given pair of event mentions, we first alter only one of the arguments and keep the rest of the content fixed. We then observe the behavior of the system across different variations of the altered argument (Table 5).

Example I In this example, we pick the AGENT-argument as the target and alter the AGENT-argument of the second event mention. The event pair (m_1, m_2^a) is non-coreferent due to the explicit incompatibility between *Steve Jobs* and *John*, and the system's prediction is also non-coreferent. Further, we alter the target argument to the pronoun *she* (m_2^b), resulting in an implicit incompatibility in the AGENT argument since the *Steve Jobs* is generally not considered a feminine name. As expected, the system classifies the event pair (m_1, m_2^b) as non-coreferent. Finally, when we alter the target argument to *he* (m_2^c), the system correctly classifies the resulting pair as coreferent.

Example II In this example, we pick the PATIENT-argument as the target and alter the PATIENT-argument of the second event mention. The system classifies the event pair (m_1, m_2^a) as coreferent, which is reasonable considering the presence of the explicit compatible arguments *housewife* and *29-year-old housewife*. Further, when we alter the target argument to *woman* (m_2^b), the system output is still coreferent. This is consistent with our prediction: the event mentions are likely to be coreferent judging only from the context of the two event mentions. However, when we alter the target argument to *medical student* (m_2^c), the event pair would become non-coreferent due to the incompatibility between *medical student* and *housewife*. The system classifies the event pair correctly.

Example III In this example, we pick the REASON-argument as the target and alter the REASON-argument of the second event mention. The event pair (m_1, m_2^a) has a pair of implicit compatible arguments in the REASON-argument role and is likely to be coreferent. In contrast, altering the target argument to *contentious citizenship amendment bill* (m_2^b) would yield an pair of implicit incompatible arguments, and the resulting event pair would become non-coreferent. Our system classifies both event pairs correctly.

6 Conclusion

We proposed an iterative transfer learning framework for event coreference resolution. Our method exploited a large unlabeled corpus to learn a wide range of (in)compatibilities between arguments, which contributes to the improvement in performance on the event coreference resolution task. We achieved state-of-the-art results on the KBP 2017 English event coreference dataset, outperforming the previous state-of-the-art system. In addition, a qualitative analysis of the system output confirmed the ability of our system to capture (in)compatibilities between two event mentions.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037 and JST CREST Grant Number JPMJCR1301, Japan.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the COLING/ACL Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4553 – 4558.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2074–2080.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC Workshop on Linguistics Coreference*, pages 563–566.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Chen Chen and Vincent Ng. 2013. Chinese event coreference resolution: Understanding the state of the art. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 822–828.

Chen Chen and Vincent Ng. 2014. SinoCoreferencer: An end-to-end Chinese event coreference resolver. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4532–4538.

Chen Chen and Vincent Ng. 2015. Chinese event coreference resolution: An unsupervised probabilistic model rivaling supervised resolvers. In *Proceedings of Human Language Technologies: The 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1097–1107.

Chen Chen and Vincent Ng. 2016. Joint inference over a lightly supervised information extraction pipeline: Towards event coreference resolution for resource-scarce languages. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2913–2920.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the ACL-IJCNLP Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*, pages 54–57.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the RANLP Workshop on Events in Emerging Text Types*, 3, pages 17–22.

Prafulla Kumar Choubey and Ruihong Huang. 2017. Event coreference resolution by iteratively unfolding inter-dependencies among events. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2124–2133.

Agata Cybulska and Piek Vossen. 2013. Semantic relations between events and their time, locations and participants for event coreference resolution. In *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing*, pages 156–163.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2015 Workshop*.

Joe Ellis, Jeremy Getman, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2016. Overview of linguistic resources for the TAC KBP 2016 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2016 Workshop*.

Jeremy Getman, Joe Ellis, Zhiyi Song, Jennifer Tracey, and Stephanie Strassel. 2017. Overview of linguistic resources for the TAC KBP 2017 evaluations: Methodologies and results. In *Proceedings of the TAC KBP 2017 Workshop*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *Proceedings of the 6th International Conference on Learning Representations*.

Sepp Hochreiter and Jüngen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Shanshan Jiang, Yihan Li, Tianyi Qin, Qian Meng, and Bin Dong. 2017. SRCB entity discovery and linking (EDL) and event nugget systems for TAC 2017. In *Proceedings of the TAC KBP 2017 Workshop*.

Sebastian Krause, Feiyu Xu, Hans Uszkoreit, and Dirk Weissenborn. 2016. Event Linking with Sentential Features from Convolutional Neural Networks. In *Proceedings of the 20th Conference on Computational Natural Language Learning*, pages 239–249.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4539–4544.

Jing Lu and Vincent Ng. 2016. Event coreference resolution with multi-pass sieves. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 3996–4003.

Jing Lu and Vincent Ng. 2017. Joint learning for event coreference resolution. In *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 90–101.

Jing Lu and Vincent Ng. 2018. Event coreference resolution: A survey of two decades of research. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5479–5486.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3264–3275.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Katie McConky, Rakesh Nagi, Moises Sudit, and William Hughes. 2012. Improving event co-reference by context extraction and dynamic feature weighting. In *Proceedings of the 2012 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pages 38–43.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Boyuan Pan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He. 2018. Discourse marker augmented network with reinforcement learning for natural language inference. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 989–999.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English Gigaword fourth edition. In *Linguistic Data Consortium*.

Haoruo Peng, Yangqi Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 392–402.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand Index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Satyan Sangeetha and Michael Arock. 2012. Event coreference resolution using mincut based graph clustering. In *Proceedings of the 4th International Workshop on Computer Networks & Communications*, pages 253–260.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*, pages 45–52.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:517–528.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or Characters? Fine-grained gating for reading comprehension. In *Proceedings of the 5th International Conference on Learning Representations*.

Sentence Embedding Alignment for Lifelong Relation Extraction

Hong Wang[†], **Wenhan Xiong**[†], **Mo Yu**[‡*], **Xiaoxiao Guo**[‡*], **Shiyu Chang**[‡], **William Yang Wang**[†]
[†] University of California, Santa Barbara
[‡] IBM Research
{hongwang600, xwhan, william}@cs.ucsb.edu, yum@us.ibm.com, {xiaoxiao.guo, shiyu.chang}@ibm.com

Abstract

Conventional approaches to relation extraction usually require a fixed set of pre-defined relations. Such requirement is hard to meet in many real applications, especially when new data and relations are emerging incessantly and it is computationally expensive to store all data and re-train the whole model every time new data and relations come in. We formulate such a challenging problem as lifelong relation extraction and investigate memory-efficient incremental learning methods without catastrophically forgetting knowledge learned from previous tasks. We first investigate a modified version of the stochastic gradient methods with a replay memory, which surprisingly outperforms recent state-of-the-art lifelong learning methods. We further propose to improve this approach to alleviate the forgetting problem by anchoring the sentence embedding space. Specifically, we utilize an explicit alignment model to mitigate the sentence embedding distortion of the learned model when training on new data and new relations. Experiment results on multiple benchmarks show that our proposed method significantly outperforms the state-of-the-art lifelong learning approaches.

1 Introduction

The task of relation detection/extraction aims to recognize entity pairs' relationship from given contexts. As an essential component for structured information extraction, it has been widely used in downstream tasks such as automatic knowledge-based completion (Riedel et al., 2013) and question answering (Yih et al., 2015; Yu et al., 2017).

Existing relation detection methods always assume a closed set of relations and perform once-and-for-all training on a fixed dataset. While making the evaluation straightforward, this setting clearly limits the usage of these methods in realistic applications, where new relations keep emerging over time. To build an evolving system which automatically keeps up with the dynamic data, we consider a more practical lifelong learning setting (also called *continual learning*) (Ring, 1994; Thrun, 1998; Thrun and Pratt, 2012), where a learning agent learns from a sequence of tasks, where each of them includes a different set of relations. In such scenarios, it is often infeasible to combine the new data with all previous data and re-train the model using the combined dataset, especially when the training set for each task is huge.

To enable efficient learning in such scenarios, recent lifelong learning research (Kirkpatrick et al., 2016; Lopez-Paz and Ranzato, 2017) propose to learn the tasks incrementally, while at the same time preventing catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990; McClelland et al., 1995; French, 1999), i.e., the model abruptly forgets knowledge learned on previous tasks when learning on the new task. Current lifelong learning approaches address such challenge by either preserving the training loss on previously learned tasks (GEM) (Lopez-Paz and Ranzato, 2017), or selectively dimming the updates on important model parameters (EWC) (Kirkpatrick et al., 2016). These methods usually involve adding additional constraints on the model's parameters or the updates of parameters by utilizing stored samples. Despite the effectiveness of these methods on simple image classification tasks, there is little research validating the practical usage of these methods in realistic NLP tasks. In fact, when applying these methods to our relation extraction task, we observe that they underperform a simple baseline that updates the model parameters (i.e., learning by SGD) with a mix

---

[*] Co-mentoring

[-] Code and dataset can be found in this repository: `https://github.com/hongwang600/Lifelong_Relation_Detection`

Proceedings of NAACL-HLT 2019, pages 796–806
Minneapolis, Minnesota, June 2 - June 7, 2019.

of stored samples from previous tasks and new samples from the incoming task. We further test this simple baseline on commonly used continual learning benchmarks and get similar observations.

In this work, we thoroughly investigate two existing continual learning algorithms on the proposed lifelong relation extraction task. We observe that recent lifelong learning methods only operate on the models' parameter space or gradient space, and do not explicitly constraint the feature or embedding space of neural models. As we train the model on the new task, the embedding space might be distorted a lot, and become infeasible for previous tasks. We argue that the embedding space should not be distorted much in order to let the model work consistently on previous tasks. To achieve this, we propose an alignment model that explicitly anchors the sentence embeddings derived by the neural model. Specifically, the alignment model treats the saved data from previous tasks as anchor points and minimizes the distortion of the anchor points in the embedding space in the lifelong relation extraction. The aligned embedding space is then utilized for relation extraction. Experiment results show that our method outperforms the state-of-the-art significantly in accuracy while remaining efficient.

The main contributions of this work include:

- We introcduce the lifelong relation detection problem and construct lifelong relation detection benchmarks from two datasets with large relation vocabularies: SimpleQuestions (Bordes et al., 2015) and FewRel (Han et al., 2018).
- We propose a simple memory replay approach and find that current popular methods such as EWC and GEM underperform this method.
- We propose an alignment model which aims to alleviate the catastrophic forgetting problem by slowing down the fast changes in the embedding space for lifelong learning.

2 Problem Definition

Generic definition of lifelong learning problems In lifelong learning, there is a sequence of K tasks $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(K)}\}$. Each task $\mathcal{T}^{(k)}$ is a conventional supervised task, with its own label set $L^{(k)}$ and training/validation/testing data $(T^{(k)}_{\text{train}}, T^{(k)}_{\text{valid}}, T^{(k)}_{\text{test}})$, each of which is a set of labeled instances $\{(x^{(k)}, y^{(k)})\}$. Note that $x^{(k)}$ is the input data of the context and candidate relations, and $y^{(k)}$ is the ground-truth label. The goal of lifelong learning is to learn a classification model f. At each step k, f observes the task $\mathcal{T}^{(k)}$, and optimizes the loss function on its training data with a loss function $\ell(f(x), y)$. At the same time, we require the model f learned after step k could still perform well on the previous $k-1$ tasks. That is, we evaluate the model by using the average accuracy of k tasks at each step as $\frac{1}{k}\sum_{j=1}^{k} acc_{f,j}$.

To make f perform well on the previous tasks, during the lifelong learning process, we usually allow the learner to maintain and observe a memory $\mathcal{M}$ of samples from the previous tasks. Practically, with the growth of the number of tasks, it is difficult to store all the task data[1]. Therefore, in lifelong learning research, the learner is usually constrained on the memory size, denoted as a constant B. Thus at each step k, the learner is allowed to keep training samples from $\{\mathcal{T}^{(j)} | j = 1, \ldots, k-1\}$ with size less or equal to B.

Lifelong relation detection In this paper we introduce a new problem, *lifelong relation detection*. Relation detection is an important task that aims to detect whether a relation exists between a pair of entities in a paragraph. In many real-world scenarios, relation detection naturally forms a lifelong learning problem because new relation types emerge as new knowledge is constantly being discovered in various domains. For example, in the Wikidata (Vrandečić and Krötzsch, 2014) knowledge graph, the numbers of new items and properties are constantly increasing[2]. So we need to keep collecting data and updating the model over time in order to handle newly added relations.

The problem of lifelong relation detection has the same definition as above with only one difference: during prediction time, we hope to know whether an input paragraph contains any relation observed before. Therefore at time k, given an input x from task $j'<k$, instead of predicting an $y \in L^{(j')}$, we predict $y^{(k)} \in \bigcup_{j=1}^{k} L^{(j)}$. That says, the candidate label set is expanding as the learner observes more tasks, and the difficulty of each previous task is increasing over time as well.

---

[1] Even the data can be stored, it is unrealistic to make full usage of the stored data. For example, random sampling from all previous task data (e.g., for the methods in Section 4) will become statistically inefficient.

[2] https://www.wikidata.org/wiki/Wikidata:News

3 Evaluation Benchmarks for Lifelong Learning

3.1 Previous non-NLP Benchmarks

Lifelong MNIST MNIST is a dataset of handwriting ten digits (LeCun, 1998), where the input for each sample is an image, and the label is the digit the image represents. Two variants of the MNIST dataset were proposed for lifelong learning evaluation. One is MNIST *Permutations* (Kirkpatrick et al., 2016), where a task is created by rearranging pixels according to a fixed permutation. K different permutations are used to generate K tasks. Another variant is MNIST *Rotations* (Lopez-Paz and Ranzato, 2017), where each task is created by rotating digits by a fixed angle. K angles are chosen for creating K tasks. In our experiments, we follow (Lopez-Paz and Ranzato, 2017) to have $K = 20$ tasks for each benchmark.

Lifelong CIFAR CIFAR (Krizhevsky and Hinton, 2009) is a dataset used for object recognition, where the input is an image, and the label is the object the image contains. Lifelong CIFAR100 (Rebuffi et al., 2017a) is a variant of CIFAR-100 (CIFAR with 100 classes) by dividing 100 classes into K disjoint subsets. Each task contains samples from $\frac{100}{K}$ classes in one subset. Following (Lopez-Paz and Ranzato, 2017), we have $K = 20$ tasks, where each of them has 5 labels.

3.2 The Proposed Lifelong Relation Detection Benchmarks

Lifelong FewRel FewRel (Han et al., 2018) is a recently proposed dataset for few-shot relation detection. There are 80 relations in this dataset. We choose to create a lifelong benchmark based on FewRel because there are a sufficient number of relation labels. We extract the sentence-relation pairs from FewRel and build our lifelong FewRel benchmark as follows. Each sample contains a sentence with the ground-truth relation it refers, and a set of 10 randomly chosen false relations from all the whole relations set. The model is required to distinguish the right relation from the candidates. We apply K-Means over the averaged word embeddings of the relation names and divide 80 relations into 10 disjoint clusters. This results in 10 tasks in this benchmark, and each task contains relations from one cluster. Candidate relations will be masked if they do not appear in the history tasks.

Lifelong SimpleQuestions SimpleQuestions is a KB-QA dataset containing single-relation questions (Bordes et al., 2015). (Yu et al., 2017) created a relation detection dataset from SimpleQuestions that contains samples of question-relation pairs. For each sample, a candidate set of relations is also provided. Similar to lifelong FewRel, we divide relations into 20 disjoint clusters by using K-Means. This results in 20 tasks, and each task contains relations from one cluster.

4 Simple Episodic Memory Replay Algorithm for Lifelong Learning

Catastrophic forgetting is one of the biggest obstacles in lifelong learning. The problem is particularly severe in neural network models, because the learned knowledge of previous tasks is stored as network weights, while a slight change of weights when learning on the new task could have an unexpected effect on the behavior of the models on the previous tasks (French, 1999).

Currently, the memory-based lifelong learning approaches, which maintain a working memory of training examples from previous tasks, are proved to be one of the best solutions to the catastrophic forgetting problem. In this section, we first propose a memory-based lifelong learning approach, namely Episodic Memory Replay (EMR), which uses the working memory by sampling stored samples to replay in each iteration of the new task learning. Surprisingly, such a straightforward approach with a clear motivation was never used in previous research. We first compare EMR with the state-of-the-art memory-based algorithm Gradient Episodic Memory (GEM). We also show that the EMR outperforms GEM on many benchmarks, suggesting that it is likely to be among the top-performed lifelong learning algorithms, and it should never be ignored for comparison when developing new lifelong learning algorithms.

4.1 Episodic Memory Replay (EMR)

EMR is a modification over stochastic gradient descent algorithms. It replays randomly sampled data from memory while training on a new task, so the knowledge of previous tasks could be retained in the model. After training on each task k, EMR selects several training examples to store in the memory $\mathcal{M}$, denoted as $\mathcal{M} \bigcap T_{\text{train}}^{(k)}$.[3]

---

[3] (Rebuffi et al., 2017b) propose to dynamically change the size of memory set for each task during training. The

To handle the scalability, EMR stochastically replays the memory. Specifically, when training on task k with each mini-batch $D^{(k)}_{\text{train}} \subset T^{(k)}_{\text{train}}$, EMR samples from the memory $\mathcal{M}$ to form a second mini-batch $D^{(k)}_{\text{replay}} \subset \mathcal{M}$. Then two gradient steps are taken on the two mini-batches of $D^{(k)}_{\text{train}}$ and $D^{(k)}_{\text{replay}}$. Note that EMR could work with any stochastic gradient optimization algorithm, such as SGD, Adagrad, AdaDelta, and Adam, to optimize the model f with the mixed mini-batches.

We try two variations of $D^{(k)}_{\text{replay}}$ sampling: first, *task-level sampling*, which samples from one previous task j each time, i.e., $D^{(k)}_{\text{replay}} \subset \mathcal{M} \bigcap T^{(j)}_{\text{train}}$. Second, *sample-level sampling*, which samples all over the memory, i.e., $D^{(k)}_{\text{replay}} \subset \mathcal{M}$.

The two approaches differ in the task instance sampling probability. The task-level approach assumes a uniform distribution over tasks, while the sample-level approach has a marginal distribution on tasks that is proportional to the number of their training data in $\mathcal{M}$.[4] When tasks are balanced like MNIST and CIFAR, or when the stored data in the memory for different tasks are balanced, the two approaches become equivalent.

However, the sample-level strategy could sometimes make the code implementation more difficult: for some lifelong learning benchmarks such as MNIST Rotation, MNIST Permutation, and CIFAR-100 used in (Lopez-Paz and Ranzato, 2017), the tasks could differ from each other in the input or output distribution, leading to different computation graphs for different training examples. From our preliminary study, the task-level approach could always give results as good as those of the sample-level approach on our lifelong relation detection benchmarks (see Table 1) , so in our experiments in Section 6 we always use the task-level approach.

4.2 Comparing EMR with State-of-the-art Memory-based Lifelong Algorithm

In this part, we will first thoroughly introduce a state-of-the-art memory-based lifelong learning algorithm called Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017), and then compare EMR with it in both time complexity and experimental results on several benchmarks.

Gradient Episodic Memory (GEM) The key idea of GEM (Lopez-Paz and Ranzato, 2017) is to constrain the new task learning with previous task data stored in memory. Specifically, it constrains the gradients during training with the following operation. When training on task k, for each mini-batch $D^{(k)}_{\text{train}} \subset T^{(k)}_{\text{train}}$, it first computes the gradient $g^{(k)}_{\text{train}}$ on $D^{(k)}_{\text{train}}$, and the average gradients on the stored data of each previous task j, denoted as $g^{(j)}_{\text{task}}$. More concretely, we define

$$g^{(j)}_{\text{task}} = \frac{\sum_{i'} \nabla \ell(f(x^{(j)}_{i'}), y^{(j)}_{i'})}{|\mathcal{M} \bigcap T^{(j)}_{\text{train}}|},$$

where $j<k$, $\ell(\cdot)$ is the loss function, and $(x^{(j)}_{i'}, y^{(j)}_{i'}) \in \mathcal{M} \bigcap T^{(j)}_{\text{train}}$, i.e. $(x^{(j)}_{i'}, y^{(j)}_{i'})$ is a training instance in $\mathcal{T}^{(j)}$ that was stored in memory $\mathcal{M}$. Then the model f is updated along the gradient $\tilde{g}$ that solves the following problem:

$$\begin{aligned} &\min_{\tilde{g}} \quad ||\tilde{g} - g^{(k)}_{\text{train}}||^2 \\ &\text{s.t.} \quad \langle \tilde{g}, g^{(j)}_{\text{task}} \rangle \geq 0, \; j = 1, \ldots, k-1. \end{aligned}$$

$\tilde{g}$ is the closest gradient to the gradient on the current training mini-batch, $g^{(k)}_{\text{train}}$, without decreasing performance on previous tasks much since the angle between $\tilde{g}$ and $g^{(j)}_{\text{task}}$ is smaller than 90°.

Time Complexity One difference between EMR and GEM is that EMR deals with unconstrained optimization and does not require the gradient projection, i.e., solving $\tilde{g}$. But since the model f is deep networks, empirically the time complexity is mainly dominated by the computation of forward and backward passes. We analyze the time complexity as below:

In task k, suppose the mini-batch size is $|D|$ and the memory replay size is m, our EMR takes $|D| + m$ forward/backward passes in each training batch. Note that m is a fixed number and set to be equal to the number of instances stored for each previous task in our experiments. While for GEM, it needs to compute the gradient of all the data stored in the memory $\mathcal{M}$, thus $|D| + |\mathcal{M}|$ forward/backward passes are taken. Its complexity is largely dominated by the size $|\mathcal{M}|$ (upper bounded by the budget B). When the budget B is large, with the number of previous tasks increases, $\mathcal{M}$ grows linearly, and GEM will become infeasible.

---

followup work and this paper all use fixed sets, and we will investigate the usage of dynamic sets in future work.

[4] The two approaches hence favor different evaluation metrics – the former fits macro averaging better and the latter fits micro averaging better.

Task	EMR		GEM
	sample	task	
MNIST Rotation	–	0.828	**0.860**
MNIST Permutation	–	0.824	**0.826**
CIFAR-100	–	**0.675**	**0.675**
FewRel	0.606	**0.620**	0.598
SimpleQuestions	0.804	**0.808**	0.796

Table 1: The average accuracy across all the tasks at last time step for EMR and GEM on both non-NLP and our lifelong relation detection benchmarks. For the experiments on MNIST and CIFAR, we follow the setting in (Lopez-Paz and Ranzato, 2017) (see Appendix A.2 for details). For the experiments on FewRel and SimpleQuestions, we use the same setting in Section 6. We only implement task-level EMR for MNIST and CIFAR because of the relatively easy implementation.

Superior Empirical Results of EMR The EMR algorithm is much simpler compared to the GEM. However, one interesting finding of this paper is that the state-of-the-art GEM is unnecessarily more complex and more inefficient, because EMR, a simple stochastic gradient method with memory replay, outperforms it on several benchmarks.

The results are shown in Table 1. The numbers are the average accuracy, i.e. $\frac{1}{k}\sum_{j=1}^{k} acc_{f,j}$, at last time step. For both algorithms, the training data is randomly sampled to store in the memory, following (Lopez-Paz and Ranzato, 2017). On lifelong relation detection, the EMR outperforms GEM on both of our created benchmarks. To further show its generalizability, we apply the EMR to previous lifelong MNIST and CIFAR benchmarks and compare to the results in (Lopez-Paz and Ranzato, 2017) with all the hyperparameters set as the same. Still, EMR performs similarly to GEM except for the MNIST Rotation benchmark.[5]

From the above results, we learned the lesson that previous lifelong learning approaches actually fail to show improvement compared to doing memory replay in a stochastic manner. We hypothesise that GEM performs worse when there is positive transfer among tasks, making the gradient projection an inefficient way to use gradients computed from memory data. Therefore, in the next section, we start with the basic EMR and focus on more efficient usage of the historical data.

5 Embedding Aligned EMR (EA-EMR)

Based on our basic EMR, this section proposes our solution to lifelong relation detection. We improve the basic EMR with two motivations: (1) previous lifelong learning approaches work on the parameter space. However, the number of parameters in a deep network is usually huge. Also, deep networks are highly non-linear models, and the parameter dimensions have complex interactions, making the Euclidean space of parameters not a proper delegate of model behavior (French, 1999). That is, a slight change in parameter space could affect the model prediction unexpectedly. The above two reasons make it hard to maintain deep network behaviors on previous tasks with constraints or Fisher information. Therefore, we propose to alleviate catastrophic forgetting in the hidden space (i.e., the sentence embedding space). (2) for each task, we want to select the most informative samples to store in the memory, instead of random sampling like in (Lopez-Paz and Ranzato, 2017). Therefore the budget of memory can be better utilized.

5.1 Embedding Alignment for Lifelong Learning

This section introduces our approach which performs lifelong learning in the embedding space, i.e., the Embedding Aligned EMR (EA-EMR).

In EA-EMR, for each task k, besides storing the original training data $(x^{(k)}, y^{(k)})$ in the memory $\mathcal{M}$, we also store the embeddings of $x^{(k)}$. In the future after a new task is trained, the model parameters are changed thus the embeddings for the same $(x^{(k)}, y^{(k)})$ would be different. Intuitively, a lifelong learning algorithm should allow such parameter changes but ensure the changes do not distort the previous embedding spaces too much.

Our EA-EMR alleviates the distortion of embedding space with the following idea: if the embedding spaces at different steps are not distorted much, there should exist a simple enough transformation a (e.g., a linear transformation in our case) that could transform the newly learned embeddings to the original embedding space, without much performance degeneration on the stored instances. So we propose to add a transformation a on the top of the original embedding and learn the basic model f and the transformation a automatically. Specifically, at the k-th task, we start with the model $f^{(k-1)}$, and the transformation $a^{(k-1)}$,

---

[5] Even on MNIST Rotation, it has achieved a competitive result, since the conventional training on shuffled data from all the tasks in this benchmark gives ~ 0.83 according to (Lopez-Paz and Ranzato, 2017).

that trained on the previous $k-1$ tasks. We want to learn the basic model f and the transformation a such that the performance on the new task and stored instances are optimized without distorting the previous embedding spaces much.

$$\min_{f(\cdot),a(\cdot)} \sum_{(x,y)\in D^{(k)}_{\text{train}}} \ell(a(f(x)),y) + \sum_{(x,y)\in D^{(k)}_{\text{replay}}} \left(\ell(a(f(x)),y) + \|a(f(x)) - a^{(k-1)}(f^{(k-1)}(x))\|^2\right)$$

We propose to minimize the above objective through two steps. In the first step, we optimize the basic model f by:

$$\min_{f(\cdot)} \sum_{(x,y)\in D^{(k)}_{\text{train}} \cup D^{(k)}_{\text{replay}}} \ell\left(a^{(k-1)}(f(x)),y\right)$$

This step mainly focuses on learning the new task without performance drop on the stored samples.

In second step, we optimize a to keep the embedding space of the current task and restore the previous embedding space of all stored samples:

$$\min_{a(\cdot)} \sum_{(x,y)\in D^{(k)}_{\text{train}}} \|a(f(x)) - a^{(k-1)}(f(x))\|^2 + \sum_{(x,y)\in D^{(k)}_{\text{replay}}} \|a(f(x)) - a^{(k-1)}(f^{(k-1)}(x))\|^2$$

Embedding Alignment on Relation Detection Model We introduce how to add embedding alignment to relation detection models. The basic model we use is a ranking model that is similar to HR-BiLSTM (Yu et al., 2017). Two BiLSTMs (Hochreiter and Schmidhuber, 1997) are used to encode the sentence and relation respectively given their GloVe word embedding (Pennington et al., 2014). Cosine similarity between the sentence and relation embedding is computed as the score. Relation with maximum score is predicted by the model for the sentence. Ranking loss is used to train the model[6]. This base model is our model f, which is trained on a new task k at each step and results in an updated model $f^{(k)}$. Our proposed approach (Figure 1) inserts an alignment model a to explicitly align to embedding space for stored instances and maintain the embedding space of the current task. Note that the label y (the relation here) also has embedding, so it needs to pass through the alignment model a as well.

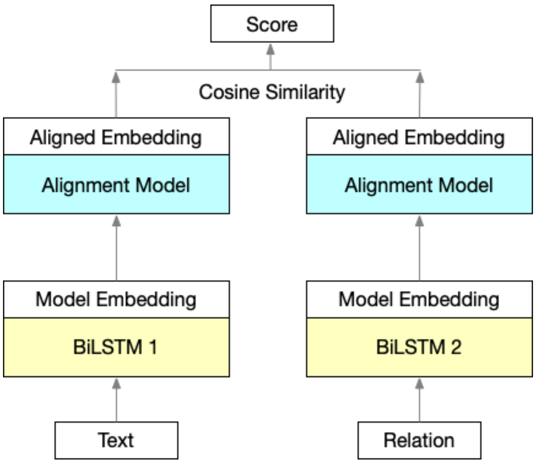


Figure 1: This figure shows how we add the alignment model (a linear model in our case) on the basic relation detection model, where two BiLSTMs are used to encode the text and relation, and cosine similarity between their embeddings are computed as the score.

5.2 Selective Storing Samples in Memory

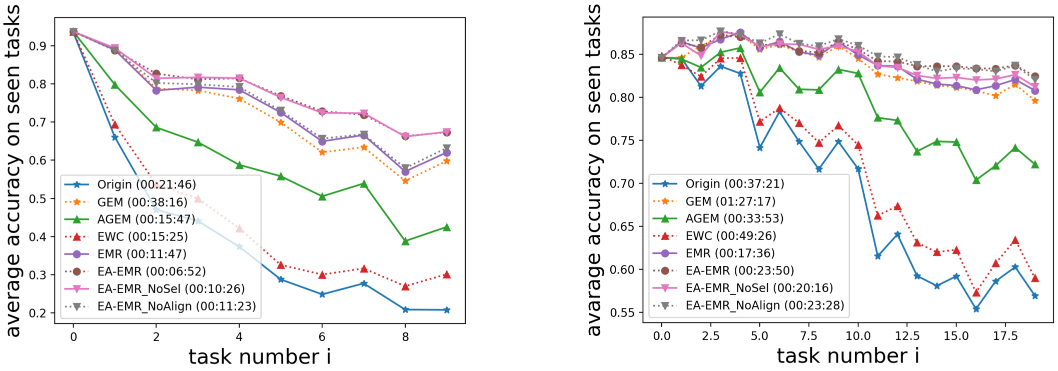
When the budget of memory is relatively smaller, how to select previous samples will greatly affect the performance. Ideally, in order to make the memory best represents a previous task, we hope to choose diverse samples that best approximate the distribution of task data. However, distribution approximation itself is a hard problem and will be inefficient due to its combinatorial optimization nature. Therefore, many recent works such as GEM ignore this step and randomly select samples from each task to store in the memory.

Rebuffi et al. (2017b) proposed to select exemplars that best approximate the mean of the distribution. This simplest distribution approximation does not give an improvement in our experiments because of the huge information loss. Therefore, we propose a better approach of sample selection by clustering over the embedding space from the model, and choose one representative from each cluster to store in the memory. More specifically, The embedding after alignment model is used to represent the input because the model makes prediction based on that. Then we apply K-Means (the number of clusters equals the budget given to the specific task) to cluster all the samples of the task. For each cluster, we select the sample closest to the centroid to store in the memory.

We leave more advanced approaches of representative sample selection and their empirical comparison to future work.

---

[6] Though the basic model is simple, it achieves reasonable results on the two datasets when training with all the data, i.e., 0.837 on FewRel and 0.927 on SimpleQuestions.



(a) FewRel (b) SimpleQuestions

Figure 2: This figure shows the average accuracy of all the observed tasks on the benchmarks of lifelong FewRel and lifelong SimpleQuestions during the lifelong learning process. The average performance of 5 runs is reported, and the **average running time** is shown in the brackets.

6 Experiments

6.1 Experimental Setting

We conduct experiments on our lifelong benchmarks: lifelong SimpleQuestions (Bordes et al., 2015) and lifelong FewRel (Han et al., 2018) to compare our proposed methods EA-EMR, EA-EMR without Selection (EA-EMR_NoSel), EA-EMR without Alignment (EA-EMR_noAlign), and EMR with the following baselines.

- **Origin**, which simply trains on new tasks based on the previous model.
- **EWC** (Kirkpatrick et al., 2016), which slows down updates on important parameters by adding L_2 regularization of parameter changes to the loss.
- **GEM** (Lopez-Paz and Ranzato, 2017), which projects the gradient to benefit all the tasks so far by keeping a constraint for each previous task.
- **AGEM** (Anonymous, 2019), which only uses one constraint that the projected gradient should decrease the average loss on previous tasks.

On both FewRel and SimpleQuestions, the epoch to train on each task is set to be 3. Learning rate for the basic model is set to be 0.001. The hidden size of LSTM is set to be 200. The batch size is set to be 50. For each sample in the memory, 10 candidate relations is randomly chosen from all observed relations to alleviate the problem that new relations are emerging incessantly.

Parameters for our model and baselines are set as follows. For EA-EMR and EA-EMR_NoSel, when training the alignment model, the learning rate is set to be 0.0001, and the training epoch is set to be 20 and 10 for FewRel and SimpleQuestions respectively. For AGEM, 100 samples are randomly chosen from all the previous tasks to form a constraint. For EWC, we set the balancing parameter $\alpha = 100$. For GEM and EMR related methods, memory size of each task is set to be 50.

Method	FewRel		SimpleQuestions	
	Whole	Avg	Whole	Avg
Origin	0.189	0.208	0.632	0.569
Baselines				
GEM	0.492	0.598	0.841	0.796
AGEM	0.361	0.425	0.776	0.722
EWC	0.271	0.302	0.672	0.590
Ours				
Full EA-EMR	**0.566**	0.673	**0.878**	**0.824**
w/o Selection	0.564	**0.674**	0.857	0.812
w/o Alignment	0.526	0.632	0.869	0.820
w/o Alignment but keep the architecture	0.545	0.655	0.871	0.813
EMR Only	0.510	0.620	0.852	0.808

Table 2: This table shows the accuracy on the whole testing data (”Whole” column), and average accuracy on all observed tasks (”Avg” column) after the last time step. The average performance of 5 runs are listed here and the best result on each dataset is marked in bold.

6.2 Lifelong Relation Detection Results

Evaluation Metrics We use two metrics to evaluate the performance of the model:

- Average performance on all seen tasks after time step k, which highlights the catastrophic problem:

$$\text{ACC}_{\text{avg}} = \frac{1}{k}\sum_{i=1}^{k} acc_{f,i}$$

- Accuracy on the whole testing data of all tasks:

$$\text{ACC}_{\text{whole}} = acc_{f,D_{\text{test}}}$$

Results on FewRel and SimpleQuestions We run each experiment 5 times independently by shuffling sequence of tasks, and the average performance is reported. The average accuracy over all observed tasks during the whole lifelong learning process is presented in Figure 2, and the accuracy on the whole testing data during the process is shown in Appendix A.1. We also list the result at last step in Table 2. From the results, we can see that EWC and GEM are better than the Origin baseline on both two datasets, which indicates that they are able to reduce the catastrophic forgetting problem. However, our EA-EMR perform significantly better than these previous state-of-the-arts. The proposed EMR method itself achieves better results than all baselines on both datasets. The ablation study shows that both the *selection* and the *alignment* modules help on both tasks.

The Effect of Embedding Alignment To investigate the effect of our embedding alignment approach, we conduct two ablation studies as below: First, we remove both the alignment loss in equation 5.1, as well as the alignment module a, which results in significant drop on most of the cases (the line "w/o Alignment" in Table 2). Second, to make sure that our good results do not come from introducing a deeper model with the module a, we propose to only remove the embedding alignment loss, but keep everything else unchanged. That means, we still keep the module a and the training steps, with the only change on replacing the loss in step 2 with the one in step 1 (the line "w/o Alignment but keep the architecture" in Table 2). We can see that this decreases the performance a lot. The above results indicate that by explicitly doing embedding alignment, the performance of the model can be improved by alleviating the distortion of previous embedding space.

Comparison of Different Sample Selection Strategies Here we compare different selection methods on lifelong FewRel and SimpleQuestions. EMR Only randomly choose samples. (Rebuffi et al., 2017b) propose to choose samples that can best approximate the mean of the distribution. We compare their sampling strategy (denoted as iCaRL) with our proposed method (K-Means) which encourages to choose diverse samples by choosing the central sample of the cluster in the embedding space. From the results in Table 3, we can see that our method outperforms iCaRL and the random baseline. While iCaRL is not significantly different from the random baseline.

Method	FewRel		SimpleQuestions	
	Whole	Avg	Whole	Avg
EMR Only	0.510	0.620	0.852	0.808
+ K-Means	**0.526**	**0.632**	**0.869**	**0.820**
+ iCaRL	0.501	0.615	0.854	0.806

Table 3: Comparison of different methods to select data for EMR. The accuracy on the whole testing data (”Whole” column), and average accuracy on all observed tasks (”Avg” column) is reported. We run each method 5 times, and give their average results.

7 Related Work

Lifelong Learning without Catastrophic Forgetting Recent lifelong learning research mainly focuses on overcoming the *catastrophic forgetting* phenomenon (French, 1999; McCloskey and Cohen, 1989; McClelland et al., 1995; Ratcliff, 1990), i.e., knowledge of previous tasks is abruptly forgotten when learning on a new task.

Existing research mainly follow two directions: the first one is *memory-based approach* (Lopez-Paz and Ranzato, 2017; Anonymous, 2019), which saves some previous samples and optimizes a new task with a forgetting cost defined on the saved samples. These methods have shown strength in alleviating catastrophic forgetting, but the computational cost grows rapidly with the number of previous tasks. The second direction is to *consolidate parameters that are important to previous tasks* (Kirkpatrick et al., 2016; Liu et al., 2018; Ritter et al., 2018; Zenke et al., 2017). For example, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2016) slows down learning on weights that are important to previous tasks. These methods usually do not need to save any previous data and only train on each task once. But their abilities to overcome catastrophic forgetting are limited.

Lifelong Learning with Dynamic Model Architecture There is another related direction on dynamically changing the model structure (i.e., adding new modules) in order to learn the new task without interfering learned knowledge for previous tasks, such as (Xiao et al., 2014; Rusu et al., 2016; Fernando et al., 2017). These approaches could successfully prevent forgetting. However, they do not suit many lifelong settings in NLP. First, it cannot benefit from the positive transfer between tasks. Second, the size of the model grows dramatically with the number of observed tasks, which makes it infeasible for real-world problems where there are a lot of tasks.

Remark It is worth noting that the term lifelong learning is also widely used in (Chen et al., 2015; Chen, 2015; Shu et al., 2016, 2017), which mainly focus on how to represent, reserve and extract knowledge of previous tasks. These works belong to a research direction different from lifelong learning without catastrophic forgetting.

8 Conclusion

In this paper, we introduce lifelong learning into relation detection, and find that two state-of-the-art lifelong learning algorithms, GEM and EWC, are outperformed by a simple memory replay method EMR on many benchmarks. Based on EMR, we further propose to use embedding alignment to alleviate the problem of embedding space distortion, which we think is one reason that causes catastrophic forgetting. Also, we propose to choose diverse samples to store in the memory by conducting K-Means in the model embedding space. Experiments verify that our proposed methods significantly outperform other baselines.

Acknowledgement

This research was supported in part by a UCSB Chancellor's Fellowship, an IBM Faculty Award, and a DARPA Grant D18AP00044 funded under the DARPA YFA program. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Anonymous. 2019. Efficient lifelong learning with a-gem. In *Submitted to International Conference on Learning Representations*. Under review.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Zhiyuan Chen. 2015. Lifelong machine learning for topic modeling and beyond. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 133–139. Association for Computational Linguistics.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong learning for sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 750–756. Association for Computational Linguistics.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4803–4809.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Yann LeCun. 1998. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Xialei Liu, Marc Masana, Luis Herranz, Joost van de Weijer, Antonio M. López, and Andrew D. Bagdanov. 2018. Rotate your networks: Better weight consolidation and less catastrophic forgetting. *CoRR*, abs/1802.02950.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6470–6479.

James L McClelland, Bruce L McNaughton, and Randall C O'reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419–457.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285–308.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017a. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5533–5542.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017b. icarl: Incremental classifier and representation learning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5533–5542. IEEE.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Mark Bishop Ring. 1994. *Continual learning in reinforcement environments*. Ph.D. thesis, University of Texas at Austin Austin, Texas 78712.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. *CoRR*, abs/1805.07810.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR*, abs/1606.04671.

Lei Shu, Bing Liu, Hu Xu, and Annice Kim. 2016. Lifelong-rl: Lifelong relaxation labeling for separating entities and aspects in opinion targets. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 225–235. Association for Computational Linguistics.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 148–154. Association for Computational Linguistics.

Sebastian Thrun. 1998. *Lifelong Learning Algorithms*, pages 181–209. Springer US, Boston, MA.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. 2014. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 177–186, New York, NY, USA. ACM.

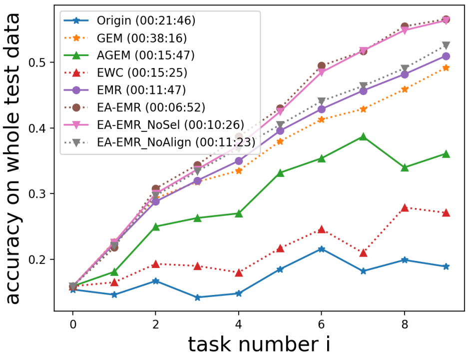
Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331. Association for Computational Linguistics.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 571–581.

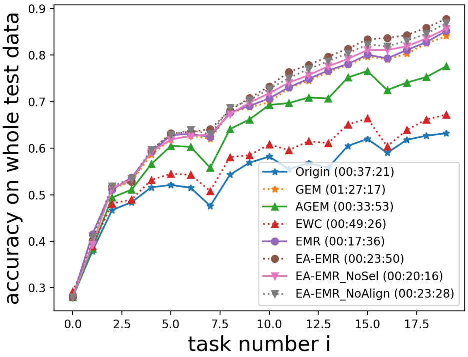
Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3987–3995.

A Appendix

A.1 Performance on the whole testing data over time



(a) FewRel



(b) SimpleQuestions

Figure 3: This figure shows the accuracy on the whole testing data on the benchmark of lifelong FewRel and lifelong SimpleQuestions during the lifelong learning process. The average performance of 5 runs is reported, and the **average running time** is shown in the brackets.

The performance on the whole testing data over time is shown in Figure 3.

A.2 Experiment setting for MNIST and CIFAR

Following the setting in (Lopez-Paz and Ranzato, 2017), the size of memory for each task is set to be 256. The learning rate is set to be 0.1. The epoch for training the model on each task is set to be 1. Plain SGD and minibatch of 10 samples are used. For the MNIST dataset, each task has 1000 samples of 10 classes. For the CIFAR dataset, each task has 2500 samples of 5 classes.

Description-Based Zero-shot Fine-Grained Entity Typing

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi and Prasad Tadepalli
School of Electrical Engineering and Computer Science, Oregon State University
{obeidatr, Xfern, shahbazh, tadepall}@eecs.oregonstate.edu

Abstract

Fine-grained Entity typing (FGET) is the task of assigning a fine-grained type from a hierarchy to entity mentions in the text. As the taxonomy of types evolves continuously, it is desirable for an entity typing system to be able to recognize novel types without additional training. This work proposes a *zero-shot* entity typing approach that utilizes the type description available from Wikipedia to build a distributed semantic representation of the types. During training, our system learns to align the entity mentions and their corresponding type representations on the known types. At test time, any new type can be incorporated into the system given its Wikipedia descriptions. We evaluate our approach on FIGER, a public benchmark entity tying dataset. Because the existing test set of FIGER covers only a small portion of the fine-grained types, we create a new test set by manually annotating a portion of the noisy training data. Our experiments demonstrate the effectiveness of the proposed method in recognizing novel types that are not present in the training data.

1 Introduction

Entity Typing assigns a semantic type (e.g., *person*, *location*, *organization*) to an entity mention in text based on the local context. It is useful for enhancing a variety of Natural Language Processing(NLP) tasks such as question answering (Han et al., 2017; Das et al., 2017), relation extraction (Liu et al., 2014; Yaghoobzadeh et al., 2016), and entity linking (Stern et al., 2012). Traditional Named Entity Typing systems consider a small set of coarse types (e.g., *person*, *location*, *organization*) (Tjong Kim Sang and De Meulder, 2003; Krishnan and Manning, 2006; Chieu and Ng, 2002). Recent studies address larger sets of fine-grained types organized in type hierarchies (e.g., *person/artist*, *person/author*) (Ling and Weld, 2012; Corro et al., 2015; Xu and Barbosa, 2018; Murty et al., 2018). Fine-Grained Entity Typing (FGET) is usually approached as a multi-label classification task where an entity mention can be assigned multiple types that usually constitute a path in the hierarchy (Ren et al., 2016).

In real-world scenarios, there is a need to deal with ever-growing type taxonomies. New types emerge, and existing types are refined into finer sub-categories. Traditional methods for entity typing assume that the training data contains all possible types, thus require new annotation effort for each new type that emerges. *Zero-shot learning (ZSL)*, a special kind of transfer learning, allows for new types to be incorporated at the prediction stage without the need for additional annotation and retraining. The main idea behind ZSL is to learn a shared semantic space for representing both the seen and unseen types, which allows the knowledge about how examples link to the seen types to be transferred to unseen types.

For fine-grained entity types, we observe that their associated Wikipedia pages often provide a rich description of the types. To capture this, we propose a *Description-based* Zero-shot Entity Typing (DZET) approach that utilizes the Wikipedia description of each type (e.g., see `https://en.wikipedia.org/wiki/Artist` for description of the type *person/artist*) to generate a representation of that type. We learn to project the entity-mention representations and the type representations into a shared semantic space, such that the mention is closer to the correct type(s) than the incorrect types. The mid-level type representation derived from the Wikipedia page along with the learned projection function allows the system to recognize new types requiring zero training examples.

We investigate different approaches for constructing the type representation based on

Proceedings of NAACL-HLT 2019, pages 807–814
Minneapolis, Minnesota, June 2 - June 7, 2019.

Wikipedia descriptions. Note that the descriptions can be quite long, often containing many different parts that are useful for recognizing different entity mentions. This motivates us to generate a bag of representations for each type and apply average pooling to aggregate the results.

We evaluate the performance of our methods on FIGER, a benchmark dataset for the FNET task, in which types are organized in 2-levels hierarchy. In this work, We focus on testing our method's capability in recognizing unseen fine-grained types (Level-2 types in this dataset). As the current test set of FIGER contains examples from only a few level-2 types, we created a new test data that covers most of the level-2 types by manually annotating a portion of the noisy training data. Below we summarize our main contributions.

- We proposed a description-based zero-shot fine-grained entity typing framework that uses Wikipedia descriptions to represent and detect novel types unseen in training.
- We created a new test set for fine-grained entity typing that provides much better coverage of the level-2 (fine-grained) types compared to the original FIGER test data.
- We provided experimental evidence of the effectiveness of our approach in comparison with established baselines.

2 Related Work

Existing work on FGET focuses on performing context-sensitive typing (Gillick et al., 2014; Corro et al., 2015), learning from noisy training data (Abhishek et al., 2017; Ren et al., 2016; Xu and Barbosa, 2018), and exploiting the type hierarchies to improve the learning and inference (Yogatama et al., 2015; Murty et al., 2018). More recent studies support even finer granularity (Choi et al., 2018; Murty et al., 2018). However, all the methods above have the limitation that they assume all types are present during training.

Zero-Shot Learning has been extensively studied in Computer Vision (CV) (Wang et al., 2019) for tasks such as image classification (Lampert et al., 2014; Zhang and Saligrama, 2015; Socher et al., 2013), object localization (Li et al., 2014, 2017) and image retrieval (Xu et al., 2017; Zhang et al., 2018). A common approach for zero-shot learning in CV is to represent each class (e.g., Zebra) by a set of semantic attributes such as its shape and color. The semantic attributes serve as the intermediate level that connects the visual features with the classes. The model is trained to learn an alignment between the semantic attributes and the visual features where a new class can be recognized using its semantic attributes without the need for any training examples. In contrast, this type of approach tends not to work well for NLP applications as the semantic concepts/classes in NLP are often more complex and cannot be easily described by a set of pre-defined attributes. This explains why the few studies of ZSL for NLP use very different methods to create the transferable intermediate representations.

Zero-Shot Learning has been studied for a number of NLP tasks including event extraction (Huang et al., 2018; Lee and Jha, 2018; Srivastava et al., 2018), relation extraction(Liu et al., 2014), Conversational Language Understanding (Lee and Jha, 2018). Specifically, Zero shot entity typing has also been explored, where most of the prior methods adopt the idea of learning a shared semantic space for representing the entities as well as the types, but differ in how they construct the type embeddings. In OTyper (Yuan and Downey, 2018), each type is represented by averaging the embedding of the words constitutes the type label. On the other hand, ProtoLE (Ma et al., 2016) represents each type by a prototype that consists of manually selected entity mentions, where the type embedding is obtained by averaging the prototype mentions' word embeddings.In contrast, our work differs from OTyper and ProtoLE by constructing the type representations based on the Wikipedia descriptions of the types, which not only carry more information about the type but also can be easily adapted to other tasks such as event typing and text classification.

3 Proposed Approaches

Following prior work on fine-grained entity typing, we formulate it as a multi-class multi-label classification problem. Given an entity mention m along with its left textual context c_l and right context c_r, We learn a classifier that predicts a binary label vector $y \in \{0,1\}^{|L|}$, where L denotes the set of all types, which forms a hierarchy Ψ. Here $y^{(t)} = 1$ if the mention m is of type t, and 0 otherwise. In the case of zero-shot FGET, new types can be introduced and added to L during testing.

3.1 The Typing Function

We will begin by introducing our typing function that is used to compute a score between a given mention and type pair, given their corresponding vector representations. We will discuss how to construct the representations in later sections.

Formally, the input to this typing function consists of the representation of the mention, denoted by $x \in \mathbf{R}^d$; and the representation of a candidate type t, denoted by $y_t \in \mathbf{R}^{\acute{d}}$. It computes a bi-linear score for the (x, y_t) pair as follows:

$$f(x, y_t, W) = x^T W y_t$$

where $W \in \mathbf{R}^{d \times \acute{d}}$ is a compatibility matrix. Following (Yogatama et al., 2015; Ma et al., 2016), we factorize W as a product of two low-rank matrices to reduce the number of parameters. That is $W = A^T B$, where $A \in \mathbf{R}^{h \times d}$ and $B \in \mathbf{R}^{h \times \acute{d}}$ (We use $h = 20$). The scoring function f can be rewritten as:

$$f(x, y_t, A, B) = \theta(x, A) \cdot \phi(y_t, B) = (Ax)^T B y_t$$

where $\theta(x, A) : x \to Ax$ and $\phi(y_t, B) : y_t \to B y_t$ serve as the projection functions that map x and y_t into a shared semantic space.

3.2 Entity Mention Representation

To obtain the representation for entity mentions, we adopt the same neural approach proposed by Shimaoka et al. (2017). Given an entity mention with its context, we compute a vector v_m to present the mention m itself, and another vector v_c to represent its left and right contexts c^l and c^r. v_m is computed by simply averaging the embedding of the individual words in m.

To compute the context embedding v_c, we first encode c^l and c^r using a bidirectional-LSTM. Let $c_1^l, ..., c_s^l$ and $c_1^r, ..., c_s^r$ be the word embedding of the left and the right context respectively, where s is the window size (we use $s = 10$), the output layer of the bi-LSTM is denoted as: $\overrightarrow{h_1^l}, \overleftarrow{h_1^l} ..., \overrightarrow{h_s^l}, \overleftarrow{h_s^l}$ and $\overrightarrow{h_1^r}, \overleftarrow{h_1^r} ..., \overrightarrow{h_s^r}, \overleftarrow{h_s^r}$. We then compute a scalar attention for each context word using a 2-level feedforward neural network:

$$e_i^j = tanh(W_e \begin{bmatrix} \overrightarrow{h_i^j} \\ \overleftarrow{h_i^j} \end{bmatrix}); \tilde{a}_i^j = exp(W_a e_i^j)$$

Where $W_e \in \mathbf{R}^{d_h \times 2 \times d_a}$, $W_a \in \mathbf{R}^{1 \times d_a}$, d_h is the dimension of LSTM, d_a is the attention dimension, $j \in \{l, r\}$. Next, we normalize a_i^js such that they sum up to 1. i.e., $a_i^j = \frac{\tilde{a}_i^j}{\sum_{i=1}^{s}(\tilde{a}_i^l + \tilde{a}_i^r)}$. Finally the context representation is computed as $v_c = \sum_{i=1}^{s}(a_i^l \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i^r} \\ \overleftarrow{h_i^r} \end{bmatrix})$. The final representation of the entity mention $x \in \mathbf{R}^d$ is a concatenation of v_m and v_c.

3.3 Type Representation

Let P_t be the Wikipedia page that is used to build a representation for type t. Some types do not have a Wikipedia page with a title the same as the type label. In such cases, we manually look for a Wikipedia page of a similar concept. For example, we represent the type *living-thing* by the Wikipedia page *organism*.

To get a type representation, We started by the simplest possible method which is averaging the embedding of words in the Wikipedia page (we call this **Avg encoder**). Since some words in the Wikipedia page carry more of the type semantic than the other words we also consider a (tf-idf)-weighted version of the Avg encoder.

Learning multiple representations. Wikipedia descriptions are often long and contain multiple parts, where different parts may capture different aspects of the type and relate to different mentions. Moreover, sequence models such as LSTM cannot be applied to such long sequences. This motivates us to consider the approach of constructing a bag of multiple representations for each type based on its Wikipedia description. To obtain a bag of representations for type t, we first use a fixed-length window to incrementally break P_t into multiple parts, one paragraph at a time. If a paragraph fits in the current Window, it is added. Otherwise, a new window is initiated. Each window of text r_{ti} is then used to generate one representation. To construct an embedding for r_{ti}, we adopt the same Bidirectional LSTM and attention mechanism that are used to embed the mention context.

To compute the score for type t given its multiple representations, we compute the score with each individual representation and average them to produce the final score. This is equivalent to applying average pooling to the multiple representations to obtain a single representation due to the bi-linear typing function.

3.4 Training and inference

Given the training data, we jointly train the representation and the scoring function by minimizing a ranking score. Let $\mathcal{Y}^{(i)}$ and $\overline{\mathcal{Y}}^{(i)}$ denote the set of correct and incorrect types assigned to the example $x^{(i)}$ respectively, we learn to score types in $\mathcal{Y}^{(i)}$ higher than types in $\overline{\mathcal{Y}}^{(i)}$ with a multi-label max-margin ranking objective as follows:

$$\sum_{y\in\mathcal{Y}}\sum_{\acute{y}\in\overline{\mathcal{Y}}} max(0, 1-f(x,y,A,B)+f(x,\acute{y},A,B))$$

At testing, both seen and unseen types are mapped to their learned representations, which are then scored for a given input. Given the scores, we conduct a top-down search following the type hierarchy Ψ. Starting from the root we recursively find the type with the highest score among the children. Since we focus on the fine-grained types, we stop the search when a leaf type is reached and predict that the mention is positive for all types along the path leading the to leaf type.

4 Experiments

Datasets. Our experiments use FIGER, a publicly available fine-grained entity typing benchmark dataset in which types are organized into a 2-level hierarchy. The training data consists of sentences sampled from Wikipedia articles and automatically annotated via distant supervision (Ling and Weld, 2012). The test data consisting of manually annotated sentences sampled from news reports.

Setting. To evaluate our capability to recognize fine-grained types in zero-shot setting, we assume all second-level types are unseen during training, i.e., we remove all level-2 types from the train and dev data but keep them in the test data. We observe that the FIGER test set covers only a small number of second-level types. This renders it insufficient for testing under the evaluation setting we adopt. Moreover, the training data is noisy since it is automatically annotated by distant supervision. As a result, we cannot just use part of it for testing.

	Original dataset			New dataset		
	train	dev	test	train	dev	test
# of mentions	2000k	10k	563	1999k	10k	917
# of types	111	111	47	46	46	66
# of level-2 types	65	65	26	0	0	40

Table 1: Statistics of FIGER dataset.

To overcome this limitation, We manually annotated a new test set from the noisy training data. We first divide the train set into clean and noisy as suggested in (Ren et al., 2016). Clean examples are those whose types fall on a single path (not necessarily ending with a leaf) in Ψ. For instance, the mention with labels *person*, *person/author*, and *person/doctor* is considered as noisy example because the labels form two paths. We then manually verify the correctness of up to 20 examples from the clean training data for every level-2 type. These examples are removed from training and added to the test set. We ignore the types with no clean examples. The statistics of the new and original datasets are reported in Table 1.

Baselines. We consider two baselines that employ the same neural architecture but use different type representations. The **Label embd** baseline use the average of the embedding of the words in the type label as the type representation. **ProtoLE** baseline uses the prototypes-based label embedding learned by Ma et al. (2016), where each type is represented by the set of the most representative entity mentions. The type embedding is the average of all mentions in the corresponding prototype.

Evaluation metrics. Following prior works in FGET, we report *Accuracy (Strict-$F1$)*, *loose Macro-averaged $F1$* ($F1_{ma}$) and *loose Micro-averaged $F1$* ($F1_{mi}$) (Ling and Weld, 2012). The training and hyperparameter tuning details are described in the Appendices.

Results and discussions. Table 2 presents the results on FIGER, evaluated on all types (Overall), the seen types (Level-1) and the unseen types (Level-2) respectively. From the results, we can see that our description based methods have a particularly strong advantage over baselines on level-2 types. This is consistent with our expectation because Wikipedia descriptions tend to be highly informative for fine-grained types, but less so for coarser types.

Among the average encoders, we found that weighting the word embedding by the word tf-idf produces better results than treating the words equivalently. As expected, using LSTM based multi-representation adds a noticeable benefit to our system as it produces the best performance among all tested methods, achieving the best performance for level-2 types and outperforming oth-

Approach	Overall			Level-1		Level-2	
	Acc	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mir}$	$F1_{ma}$	$F1_{mir}$
Label embd	0.2846	0.5510	0.5603	**0.8165**	**0.8163**	0.2854	0.2954
ProtoLE	0.2541	0.4982	0.5093	0.7424	0.7422	0.2541	0.2657
DZET + Avg encoder	0.3141	0.5522	0.5614	0.7903	0.7902	0.3141	0.3247
DZET + Weighted Avg encoder	0.3261	0.5500	0.5607	0.7740	0.7738	0.3261	0.3390
DZET + Multi-rep	**0.3806**	**0.5953**	**0.6045**	0.8100	0.8098	**0.3806**	**0.3926**

Table 2: Level-1 , Level-2 and overall performance of the models on FIGER dataset.

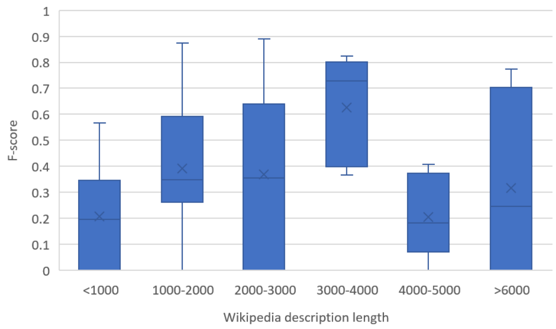


Figure 1: The relationship between the length of the Wikipedia description (word count) of level-2 types and the F-score obtained by *DZET+Multi-rep* method.

ers by a large margin while maintaining a highly competitive performance for level-1 types.

The effect of description quality. Figure 1 analyzes the relationship between the length of the Wikipedia description as one criterion of the description quality and the performance of *Multi-rep* method. In particular, we group the types based on the length of their Wikipedia descriptions and provide the five-number summary box plot of the F-scores for each group. It can be readily observed that the performance is low when the description of the type's Wikipedia page is too short ($<$ 1000 words) or too long ($>$ 4000 words). Short descriptions are less informative and carry less shared semantics with the type's mentions. On the other hand, overly long descriptions could also be confusing as it might share a significant number of common words with the descriptions of other types. A closer look into the results unveils some exceptions. For example, the F-score on the type '/education/educational-degree' is 0.7742 even it has a long description (6845 words). The description of this type is indeed very informative and includes a comprehensive list of the educational degrees awarded all around the world.

The length of the description is not the only factor that affects the performance of DZET methods. One factor is the performance on the Level-1 types. Since the inference is performed by following the type hierarchy, if an incorrect type is inferred at level-1, there is no hope to get the correct level-2 type. Another factor is the amount of overlapping between the descriptions of the related types. For instance, *Multi-rep* produces zero F-score on the types'/event/protest' and '/location/province' because they share a lot of common words with the types '/event/attack' and '/location/county' respectively, which negatively affects the ability of *Multi-rep* to distinguish between the related types. Both '/event/protest' and '/location/province' have a description length between 2000 and 3000 words.

To mitigate the effect of the contents overlapping between the highly related type, We plan to apply mention-sensitive attention mechanisms for future work to aggregate the scores in Multi-rep instead of max-pooling.

5 Conclusions

In this paper, we propose a novel zero-shot entity typing approach that uses Wikipedia descriptions to construct type embeddings. Our architecture relies on the type embeddings to make predictions for unseen types. Experimental results demonstrate the effectiveness of the proposed methods.

Acknowledgments

We thank Jordan University of Science and Technology for Ph.D. fellowship (to R. O.).

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 797–807.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *In Annual Meeting of the Association for Computational Linguistics*.

Luciano del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. Assoc. for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Sangdo Han, Soonchoul Kwon, Hwanjo Yu, and Gary Geunbae Lee. 2017. Answer ranking based on named entity types for question answering. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, page 71. ACM.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2018. Zero-shot transfer learning for event extractions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vijay Krishnan and Christopher D Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1121–1128. Association for Computational Linguistics.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Sungjin Lee and Rahul Jha. 2018. Zero-shot adaptive transfer for conversational language understanding. *arXiv preprint arXiv:1808.10059*.

Zhenyang Li, Efstratios Gavves, Thomas Mensink, and Cees GM Snoek. 2014. Attributes make sense on segmented objects. In *European Conference on Computer Vision*, pages 350–365. Springer.

Zhenyang Li, Ran Tao, Efstratios Gavves, Cees GM Snoek, and Arnold WM Smeulders. 2017. Tracking by natural language specification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7350–7358. IEEE.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2107–2116.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 97–109.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 306–316.

Rosa Stern, Benoît Sagot, and Frédéric Béchet. 2012. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. 2019. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):13.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 16–25.

Yahui Xu, Yang Yang, Fumin Shen, Xing Xu, Yuxuan Zhou, and Heng Tao Shen. 2017. Attribute hashing for zero-shot image retrieval. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 133–138. IEEE.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2016. Noise mitigation for neural entity typing and relation extraction. *arXiv preprint arXiv:1612.07495*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 291–296.

Zheng Yuan and Doug Downey. 2018. Otyper: A neural architecture for open named entity typing. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Haofeng Zhang, Yang Long, and Ling Shao. 2018. Zero-shot hashing with orthogonal projection for image retrieval. *Pattern Recognition Letters*.

Ziming Zhang and Venkatesh Saligrama. 2015. Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 4166–4174.

A Appendices

A.1 Training and Hyperparamters

For every model we trained, we tune all of the hyper-parameters using a dev set. We use the version of FIGER provided by (Shimaoka et al., 2017) which already withhold a portion of the train set as a dev set. For each experiment, we report that testing results of the model that has the best *accuracy* on the dev set. We adopt *glove* 300-dimensional word embedding (Pennington et al., 2014) throughout this work except for *prototype* baselines; we use word2vec (Mikolov et al., 2013) as it is used to compute the prototypes embedding in the original works (Ma et al., 2016). The hyper-parameters used in the feature representation component are the same as in (Shimaoka et al., 2017). we set both of the hidden-size of the LSTM was set and the hidden-layer size of the attention module to 100. We use Adam optimizer (Kingma and Ba, 2014) with the learning rate .001. The model is trained for five epochs. We use Window of size 200 to build a bag of representations for each type from its Wikipedia description.

Adversarial Decomposition of Text Representation

Alexey Romanov, Anna Rumshisky, Anna Rogers and David Donahue
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
`{aromanov,arum,arogers}@cs.uml.edu`
`david_donahue@student.uml.edu`

Abstract

In this paper, we present a method for adversarial decomposition of text representation. This method can be used to decompose a representation of an input sentence into several independent vectors, each of them responsible for a specific aspect of the input sentence. We evaluate the proposed method on two case studies: the conversion between different social registers and diachronic language change. We show that the proposed method is capable of fine-grained controlled change of these aspects of the input sentence. It is also learning a continuous (rather than categorical) representation of the style of the sentence, which is more linguistically realistic. The model uses adversarial-motivational training and includes a special motivational loss, which acts opposite to the discriminator and encourages a better decomposition. Furthermore, we evaluate the obtained meaning embeddings on a downstream task of paraphrase detection and show that they significantly outperform the embeddings of a regular autoencoder.

1 Introduction

Despite the recent successes in using neural models for representation learning for natural language text, learning a meaningful representation of input sentences remains an open research problem. A variety of approaches, from sequence-to-sequence models that followed the work of Sutskever et al. (2014) to the more recent proposals (Arora et al., 2017; Nangia et al., 2017; Conneau et al., 2017; Logeswaran and Lee, 2018; Subramanian et al., 2018; Cer et al., 2018) share one common drawback. Namely, all of them encode the input sentence into just *one* single vector of a fixed size. One way to bypass the limitations of a single vector representation is to use an attention mechanism (Bahdanau et al., 2014; Vaswani et al., 2017). We propose to approach this problem differently and design a method for adversarial decomposition of the learned input representation into multiple components. Our method encodes the input sentence into *several* vectors, where each vector is responsible for a specific aspect of the sentence.

In terms of learning different separable components of input representation, our work most closely relates to the style transfer work, which has been applied to a variety of different aspects of language, from diachronic language differences (Xu et al., 2012) to authors' personalities (Lipton et al., 2015) and even sentiment (Hu et al., 2017; Fu et al., 2018). The style transfer work effectively relies on the more classical distinction between **meaning** and **form** (de Saussure, 1959), which accounts for the fact that multiple surface realizations are possible for the same meaning. For simplicity, we will use this terminology throughout the rest of the paper.

Consider encoding an input sentence into a meaning vector and a form vector. This enables a controllable change of meaning or form by a simple change applied to these vectors. For example, we can encode two sentences written in two different styles, then swap the form vectors while leaving the meaning vectors intact. We can then generate new unique sentences with the original meaning, but written in a different style.

We propose a novel model for this type of decomposition based on adversarial-motivational training, GAN architecture (Goodfellow et al., 2014) and adversarial autoencoders (Makhzani et al., 2015). In addition to the adversarial loss, we use a special motivator (Albanie et al., 2017), which, in contrast to the discriminator, is used to provide a motivational loss to encourage better decomposition of the meaning and the form. All the code is available on GitHub [1].

---

[1] `https://github.com/text-machine-lab/adversarial_decomposition`

Proceedings of NAACL-HLT 2019, pages 815–825
Minneapolis, Minnesota, June 2 - June 7, 2019.

We evaluate the proposed methods for learning separate aspects of input representation in the following case studies:

1. Diachronic language change. Specifically, we consider the Early Modern English (e.g. *What would she have?*) and the contemporary English (*What does she want?*).
2. Social register (Halliday et al., 1968), i.e. subsets of language appropriate in a given context or characteristic of a certain group of speakers. Social registers include formal vs informal language, the language used in different genres (e.g., fiction vs. newspapers vs. academic texts), different dialects, and literary idiostyles. We experiment with the titles of scientific papers vs. newspaper articles.

2 Related work

As mentioned above, the most relevant previous work comes from research on style transfer[2]. It can be divided into two groups:

1. Approaches that aim to generate text in a given form. For example, the task may be to produce just any verse as long as it is in the "style" of the target poet.
2. Approaches that aim to induce a change in either the "form" or the "meaning" of an utterance. For example, "Good bye, Mr. Anderson." can be transformed to "Fare you well, good Master Anderson" (Xu et al., 2012)).

An example of the first group is the work of Potash et al. (2015), who trained several separate networks on verses by different hip-hop artists. An LSTM network successfully generated verses that were stylistically similar to the verses of the target artist (as measured by cosine distance on tf-idf vectors). More complicated approaches use language models that are conditioned in some way. For example, Lipton et al. (2015) produced product reviews with a target rating by passing the rating as an additional input at each timestep of an LSTM model. Tang et al. (2016) generated reviews not only with a given rating but also for a specific product. At each timestep a special context vector was provided as input, gated so as to enable the model to decide how much attention to pay to that vector and the current hidden state. Li et al. (2016) used "speaker" vectors as an additional input to a conversational model, improving consistency of dialog responses. Finally, Ficler and Goldberg (2017) performed an extensive evaluation of conditioned language models based on "content" (theme and sentiment) and "style" (professional, personal, length, descriptiveness). Importantly, they showed that it is possible to control both "content" and "style" simultaneously.

Work from the second group can further be divided into two clusters by the nature of the training data: parallel aligned corpora, or non-aligned datasets. The aligned corpora enable approaching the problem of form shift as a paraphrasing or machine translation problem. Xu et al. (2012) used statistical and dictionary-based systems on a dataset of original plays by Shakespeare and their contemporary translations. Carlson et al. (2017) trained an LSTM network on 33 versions of the Bible. Jhamtani et al. (2017) used a Pointer Network (Vinyals et al., 2015), an architecture that was successfully applied to a wide variety of tasks (Merity et al., 2016; Gulcehre et al., 2016; Potash et al., 2017), to enable direct copying of the input tokens to the output. All these works use BLEU (Papineni et al., 2002) as the main, or even the only evaluation measure. This is only possible in cases where a parallel corpus is available.

Recently, new approaches that do not require a parallel corpora were developed in both computer vision (CV) (Zhu et al., 2017) and NLP. Hu et al. (2017) succeeded in changing tense and sentiment of sentences with a two steps procedure based on a variational auto-encoder (VAE) (Kingma and Welling, 2013). After training a VAE, a discriminator and a generator are trained in an alternate manner, where the discriminator tries to correctly classify the target sentence attributes. A special loss component forces the hidden representation of the encoded sentence to not have any information about the target sentence attributes. Mueller et al. (2017) used a VAE to produce a hidden representation of a sentence, and then modify it to match the desired form. Unlike Hu et al. (2017), they do not separate the form and meaning embeddings. Shen et al. (2017) applied a GAN to align the hidden representation of sentences from two corpora and forced them not to have any information about the form an via adversarial loss. During the decoding, similarly to Lipton et al. (2015),

---

[2] The term "style" is not entirely appropriate here, but in NLP it is often used in work on any kind of form change while preserving meaning, from translation to changing sentiment polarity.

special “style” vectors are passed to the decoder at every timestep to produce a sentence with the desired properties. The model is trained using the Professor-Forcing algorithm (Lamb et al., 2016). Kim et al. (2017) worked directly on hidden space vectors that are constrained with the same adversarial loss instead of outputs of the generator, and use two different generators for different “styles”. Finally, Fu et al. (2018) generate sentences with the target properties using an adversarial loss, similarly to Shen et al. (2017) and Kim et al. (2017).

Comparison with previous work In contrast to the proposals of Xu et al. (2012), Carlson et al. (2017), Jhamtani et al. (2017), our solution does not require a parallel corpus. Unlike the model by Shen et al. (2017), our model works directly on representations of sentences in the hidden space.

Most importantly, in contrast to the proposals by Mueller et al. (2017), Hu et al. (2017), Kim et al. (2017), Fu et al. (2018), our model produces a representation for both meaning and form and does not treat the form as a categorical (in the vast majority of works, binary) variable[3].

Treating meaning and form not as binary/categorical, but continuous variables is more consistent with the reality of language use, since there are different degrees of overlap between the language used by different registers or in different diachronic slices. Indeed, language change is gradual, and the acceptability of expressions in a given register also forms a continuum, so one expects a substantial overlap between the grammar and vocabulary used, for example, on Twitter and by New York Times. To the best of our knowledge, this is the first model that considers linguistic form in the task of text generation as a continuous variable.

A significant consequence of learning a continuous representation for form is that it allows the model to work with a large, and potentially infinite, number of forms. Note that in this case the locations of areas of specific forms in the vector form space would reflect the similarity between these forms. For example, the proposed model could be directly applied to the authorship attribution problem: each author would have their own area in the form space, their proximity should mirror the similarity in writing style. Preliminary experiments on this are reported in subsection 6.4.

3 Formulation

Let us formulate the problem of decomposition of text representation on an example of controlled change of linguistic form and conversion of Shakespeare plays in the original Early Modern to contemporary English. Let $\boldsymbol{X}^a$ be a corpus of texts $\boldsymbol{x}_i^a \in \mathcal{X}^a$ in Early Modern English $\mathbf{f}^{\boldsymbol{a}} \in \mathcal{F}$, and $\boldsymbol{X}^b$ be a corpus of texts $\boldsymbol{x}_i^b \in \mathcal{X}^b$ in modern English $\mathbf{f}^{\boldsymbol{b}} \in \mathcal{F}$. We assume that the texts in both $\boldsymbol{X}^a$ and $\boldsymbol{X}^b$ have the same distribution of meaning $\mathbf{m} \in \mathcal{M}$. The form $\mathbf{f}$, however, is different and generated from a mixture of two distributions:

$$\mathbf{f}_i = \alpha_i^a p(\mathbf{f}^a) + \alpha_i^b p(\mathbf{f}^b)$$

where $\mathbf{f}^a$ and $\mathbf{f}^b$ are two different languages (Early Modern and contemporary English). Intuitively, we say that a sample $\boldsymbol{x}_i$ has the form $\mathbf{f}^{\boldsymbol{a}}$ if $\alpha_i^a > \alpha_i^b$, and it has the form $\mathbf{f}^{\boldsymbol{b}}$ if $\alpha_i^b > \alpha_i^a$.

The goal of dissociation meaning and form is to learn two encoders $E_{\mathbf{m}} : \mathcal{X} \rightarrow \mathcal{M}$ and $E_{\mathbf{f}} : \mathcal{X} \rightarrow \mathcal{F}$ for the meaning and form correspondingly, and the generator $G : \mathcal{M}, \mathcal{F} \rightarrow \mathcal{X}$ such that

$$\forall j \in \{a, b\}, \forall k \in \{a, b\} : G(E_{\mathbf{m}}(\boldsymbol{x}^k), E_{\mathbf{f}}(\boldsymbol{x}^j)) \rightarrow \mathcal{X}^j$$

The form of a generated sample depends exclusively on the provided $\mathbf{f}_j$ and can be in the same domain for two different $\mathbf{m}_u$ and $\mathbf{m}_v$ from two samples from different domains $\mathcal{X}^a$ and $\mathcal{X}^b$.

Note that, in contrast to the previous proposals, the form $\mathbf{f}$ is not a categorical variable but a continuous vector. This enables fine-grained controllable change of form: the original form $\mathbf{f}_i$ is changed to reflect the form of the specific target sentence $\mathbf{f}_j$ with its own unique α^a and α^b while preserving the original meaning $\mathbf{m}_i$.

An important caveat concerns the core assumption of the similar meaning distribution in the two corpora, which is also made in all other works reviewed in Section 2. It limits the possible use of this approach to cases where the distributions are in fact similar (i.e. comparable corpora are available; note that they do not have to be parallel). It does not apply to many cases that could be analyzed in terms of meaning and form. For example, books for children and scholarly papers are both registers, they have their own form (i.e. specific subsets of linguistic means and structure conventions) – but there is little overlap in the content.

---

[3] Although the form was represented as dense vectors in previous work, it is still just a binary feature, as they use a single pre-defined vector for each form, with all sentences of the same form assigned the same form vector.

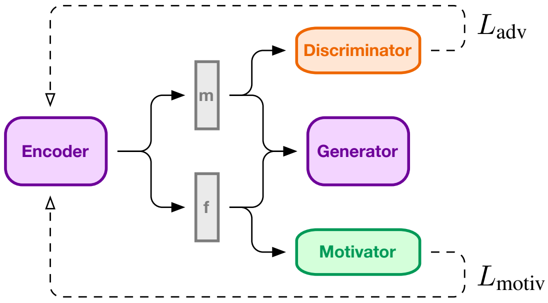


Figure 1: Overview of ADNet. *Encoder* encodes the inputs sentences into two latent vectors $\mathbf{m}$ and $\mathbf{f}$. The *Generator* takes them as the input and produces the output sentence. During the training, the *Discriminator* is used for an adversarial loss that forces $\mathbf{m}$ to not carry any information about the form, and the *Motivator* is used for a motivational loss that encourages $\mathbf{f}$ to carry the information about the form.

This would make it hard even for a professional writer to turn a research paper into a fairy tale.

4 Method description

Inspired by Makhzani et al. (2015), Kim et al. (2017), and Albanie et al. (2017), we propose ADNet, a new model for adversarial decomposition of text representation (Figure 1).

Our solution is based on a widely used sequence-to-sequence framework (Sutskever et al., 2014) and consists of four main parts. The encoder E encodes the input sequence $\boldsymbol{x}$ into two latent vectors $\mathbf{m}$ and $\mathbf{f}$ which capture the meaning and the form of the sentence correspondingly. The generator G then takes these two vectors as the input and produces a reconstruction of the original input sequence $\hat{\boldsymbol{x}}$.

The encoder and generator by themselves will likely not achieve the dissociation of the meaning and form. We encourage this behavior in a way similar to Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), which had an overwhelming success the past few years as a way to enforce a specific distribution and characteristics on the output of a model.

Inspired by the work of Albanie et al. (2017) and the principle of "carrot and stick" (Safire, 1995), in contrast to the majority of work that promotes purely adversarial approach (Goodfellow et al., 2014; Shen et al., 2017; Fu et al., 2018; Zhu et al., 2017), we propose two additional components, the discriminator D and the motivator M to force the model to learn the dissociation of the meaning and the form. Similarly to a regular GAN model, the adversarial discriminator D tries to classify the form $\mathbf{f}$ based on the latent meaning vector $\mathbf{m}$, and the encoder E is penalized to make this task as hard as possible.

Opposed to such vicious behaviour, the motivator M tries to classify the form based on the latent form vector $\mathbf{f}$, as it should be done, and encourages the encoder E to make this task as simple as possible. We could apply the adversarial approach here as well and force the distribution of the form vectors to fit a mixture of Gaussians (in this particular case, a mixture of two Guassians) with another discriminator, as it is done by Makhzani et al. (2015), but we opted for the "dualistic" path of two complimentary forces.

4.1 Encoder-Decoder

Both the encoder E and the generator G are neural networks. Gated Recurrent Unit (GRU) (Chung et al., 2014) is used for E to encode the input sentence $\boldsymbol{x}$ into a hidden vector

$$\boldsymbol{h} = \text{GRU}(\boldsymbol{x})$$

The vector $\boldsymbol{h}$ then passes through two different fully connected layers to produce the latent vectors of the form and the meaning of the input sentence:

$$\mathbf{m} = \tanh(\boldsymbol{W}_{\boldsymbol{m}}\boldsymbol{h} + \boldsymbol{b}_m)$$
$$\mathbf{f} = \tanh(\boldsymbol{W}_{\boldsymbol{f}}\boldsymbol{h} + \boldsymbol{b}_f)$$

We use $\boldsymbol{\theta}_E$ to denote the parameters of the encoder E: $\boldsymbol{W}_{\boldsymbol{m}}$, $\boldsymbol{b}_m$, $\boldsymbol{W}_{\boldsymbol{f}}$, $\boldsymbol{b}_f$, and the parameters of the GRU unit.

The generator G is also modelled with a GRU unit. The generator takes as input the meaning vector $\mathbf{m}$ and the form vector $\mathbf{f}$, concatenates them, and passes trough a fully-connected layer to obtain a hidden vector $\boldsymbol{z}$ that represents both meaning and form of the original input sentence:

$$\boldsymbol{z} = \tanh(\boldsymbol{W}_z[\mathbf{m}; \mathbf{f}] + \boldsymbol{b}_m)$$

After that, we use a GRU unit to generate the output sentence as a probability distribution over the vocabulary tokens:

$$p(\hat{\boldsymbol{x}}) = \prod_{t=1}^{T} p(\hat{\boldsymbol{x}}_t | \boldsymbol{z}, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{t-1})$$

We use $\boldsymbol{\theta}_G$ to denote the parameters of the generator G: $\boldsymbol{W}_z$, $\boldsymbol{b}_m$, and the parameters of the used GRU. The encoder and generator are trained using the standard reconstruction loss:

$$\mathcal{L}_{\text{rec}}(\boldsymbol{\theta}_E, \boldsymbol{\theta}_G) = \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{X}^a}[-\log p(\hat{\boldsymbol{x}}|\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{X}^b}[-\log p(\hat{\boldsymbol{x}}|\boldsymbol{x})]$$

4.2 Discriminator

The representation of the meaning $\mathbf{m}$ produced by the encoder E should not contain any information about the form $\mathbf{f}$. We achieve this by using an adversarial approach. First, we train a discriminator D, consisting of several fully connected layers with `ELU` activation function (Clevert et al., 2015) between them, to predict the form $\mathbf{f}$ of a sentence by its meaning vector:

$$\hat{\boldsymbol{f}}_D = D(\mathbf{m})$$

where $\hat{\boldsymbol{f}}$ is the score (logit) reflecting the probability of the sentence $\boldsymbol{x}$ to belong to one of the form domains.

Motivated by the Wasserstein GAN (Arjovsky et al., 2017), we use the following loss function instead of the standard cross-entropy:

$$\mathcal{L}_D(\boldsymbol{\theta}_D) = \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}^a}[D(E_{\mathbf{m}}(\boldsymbol{x}))] - \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}^b}[D(E_{\mathbf{m}}(\boldsymbol{x}))]$$

Thus, a successful discriminator will produce negative scores $\hat{\boldsymbol{f}}$ for sentences from $\boldsymbol{X}^a$ and positive scores for sentences from $\boldsymbol{X}^b$. This discriminator is then used in an adversarial manner to provide a learning signal for the encoder and force dissociation of the meaning and form by maximizing $\mathcal{L}_D$:

$$\mathcal{L}_{\text{adv}}(\boldsymbol{\theta}_E) = -\lambda_{\text{adv}}\mathcal{L}_D$$

where λ_{adv} is a hyperparameter reflecting the strength of the adversarial loss. Note that this loss applies to the parameters of the encoder.

4.3 Motivator

Our experiments showed that the discriminator D and the adversarial loss $\mathcal{L}_{\text{adv}}$ by themselves are sufficient to force the model to dissociate the form and the meaning. However, in order to achieve a better dissociation, we propose to use a motivator M (Albanie et al., 2017) and the corresponding motivational loss. Conceptually, this is the opposite of the adversarial loss, hence the name. As the discriminator D, the motivator M learns to classify the form $\mathbf{f}$ of the input sentence. However, its input is not the meaning vector but the form vector:

$$\hat{\boldsymbol{f}}_M = M(\mathbf{f})$$

The motivator has the same architecture as the discriminator, and the same loss function. While the adversarial loss forces the encoder E to produce a meaning vector $\mathbf{m}$ with no information about the form $\mathbf{f}$, the motivational loss encourages E to encode this information in the form vector by minimizing $\mathcal{L}_M$:

$$\mathcal{L}_{\text{motiv}}(\boldsymbol{\theta}_E) = \lambda_{\text{motiv}}\mathcal{L}_M$$

4.4 Training procedure

The overall training procedure follows the methods for training GANs (Goodfellow et al., 2014; Arjovsky et al., 2017) and consists of two stages: training the discriminator D and the motivator M, and training the encoder E and the generator G.

In contrast to Arjovsky et al. (2017), we do not train the D and M more than the E and the G. In our experiments we found that simple training in two stages is enough to achieve dissociation of the meaning and the form. Encoder and generator are trained with the following loss function that combines reconstruction loss with the losses from the discriminator and the motivator:

$$\mathcal{L}_{\text{total}}(\theta_E, \theta_G) = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{motiv}}$$

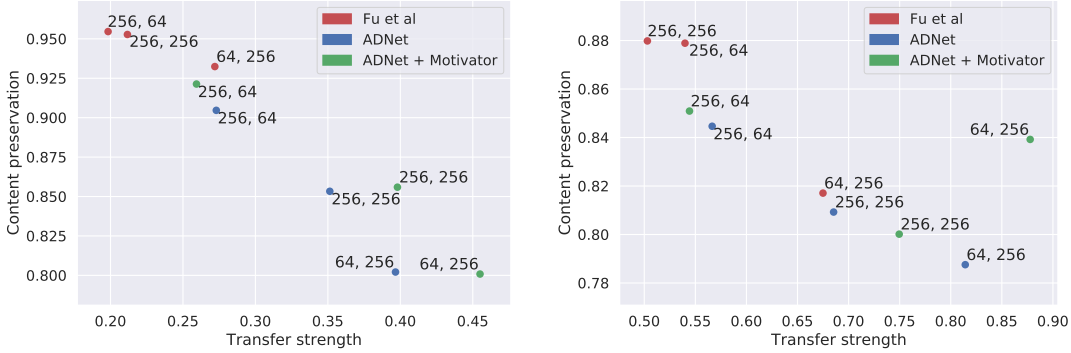
5 Experimental setup

5.1 Evaluation

Similarly to the evaluation of style transfer in CV (Isola et al., 2017), evaluation of this task is difficult. We follow the approach of Isola et al. (2017); Shen et al. (2017) and recently proposed by Fu et al. (2018) methods of evaluation of “transfer strength” and “content preservation”. The authors showed that the proposed automatic metrics correlate with human judgment to a large degree and can serve as a proxy. Below we give an overview of these metrics.

Transfer Strength. The goal of this metric is to capture whether the form has been changed successfully. To do that, a classifier C is trained on the two corpora, $\boldsymbol{X}^a$ and $\boldsymbol{X}^b$ to recognize the linguistic “form” typical of each of them. After that a sentence, for which the form/meaning has been changed, is passed to the classifier. The overall accuracy reflects the degree of success of changing the form/meaning. This approach is widely used in CV (Isola et al., 2017), and was applied in NLP as well (Shen et al., 2017).

In our experiments we used a GRU unit followed by four fully-connected layers with `ELU` activation functions between them as the classifier.



(a) Shakespeare dataset

(b) Headlines dataset

Figure 2: Transfer strength vs content preservation (see subsection 5.1) for different sizes of the meaning and form vectors. Each point is labeled with "⟨meaning vector size⟩, ⟨form vector size⟩".

Content preservation Note that the transfer strength by itself does not capture the overall quality of a changed sentence. A extremely overfitted model that produces the most characteristic sentence of one corpus all the time would have a high score according to this metric. Thus, we need to measure how much of the meaning was preserved while changing the form. To do that, Fu et al. (2018) proposed to use a cosine similarity based metric using pretrained word embeddings. First, a sentence embedding is computed by concatenation of max, mean, and average pooling over the timesteps:

$$\boldsymbol{v} = [\max(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T); \min(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T); \mathrm{mean}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T)]$$

Next, the cosine similarity score s_i between the embedding $\boldsymbol{v}_i^s$ of the original source sentence and the target sentence with the changed form $\boldsymbol{v}_i^t$ is computed, and the scores across the dataset are averaged to obtain the total score s.

5.1.1 Continuous form

The metrics described above treat the form as a categorical (in most cases, even binary) variable. This was not a problem in previous work since the change of form could be done by simply inverting the form vector. Since we treat the form as a continuous variable, we cannot just use the proposed metrics directly. To enable a fair comparison, we propose the following procedure.

For each sentence s_s^a in the test set from the corpus $\boldsymbol{X}^a$ we sample $k = 10$ random sentences from the corpus $\boldsymbol{X}^b$ of the opposite form. After that, we encode them into the meaning $\boldsymbol{m}_i$ and form $\boldsymbol{f}_i$ vectors, and average the form vectors to obtain a single form vector

$$\boldsymbol{f}_{\text{avg}} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{f}_i$$

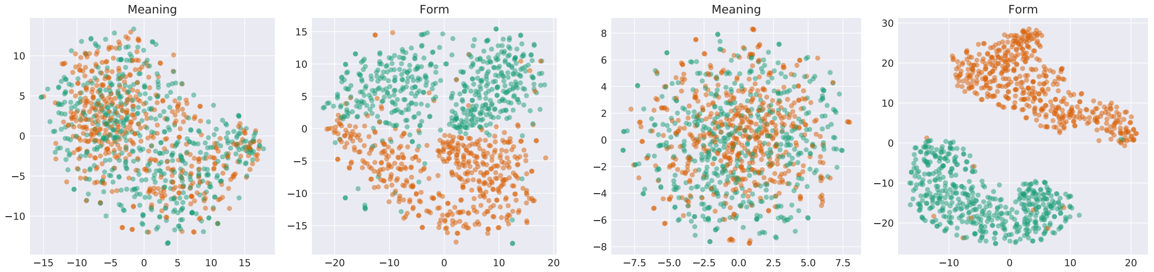
We then generate a new sentence with its original meaning vector $\boldsymbol{m}_s$ and the resulting form vector $\boldsymbol{f}_{\text{avg}}$, and use it for evaluation. This process enables a fair comparison with the previous approaches that treat form as a binary variable.

5.2 Datasets

We evaluated the proposed method on several datasets that reflect different changes of meaning and form.

Changing form: register. This experiment is conducted with a dataset of titles of scientific papers and news articles published by Fu et al. (2018). This dataset (referred to as "Headlines") contains titles of scientific articles crawled from online digital libraries, such as "ACM Digital Library" and "arXiv". The titles of the news articles are taken from the "News Aggregator Data Set" from UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017)

Changing form: language diachrony. Diachronic language change is explored with the dataset composed by Xu et al. (2012). It includes the texts of 17 plays by William Shakespeare in the original Early Modern English, and their translations into contemporary English. We randomly permuted all sentences from all plays and sampled the training, validation, and test sets. Note that this dataset is much smaller than the Headlines dataset.



(a) Without motivator

(b) With motivator

Figure 3: t-SNE visualization of the form and meaning embeddings of 1000 random sentences. Green point represent sentences form news headlines, and red points represent titles of scientific articles.

6 Results and discussion

The most recent and similar to our work is the model proposed by Fu et al. (2018), in particular the "style-embedding" model. We implemented this model to provide a baseline for comparison.

The classifier used in the transfer strength metric achieves high accuracy (0.832 and 0.99 for the Shakespeare and Headlines datasets correspondingly). These results concur with the results of Shen et al. (2017) and Fu et al. (2018), and show that the two corpora are significantly different.

Following Fu et al. (2018), we show the result of different configuration of the size of the form and meaning vectors on Figure 2. Namely, we report combinations of 64 and 256-dimensional vectors. Note that the sizes of the form vector are important. If the form vector is larger, the transfer strength is gre,ta erbut the content preservation is lessened. This is consistent with Fu et al. (2018), where they observed a similar behaviour.

It is clear that the proposed method achieves significantly better transfer strength than the previously proposed model. It also has a lower content preservation score, which means that it repeats fewer exact words from the source sentence. Note that a low transfer strength and very high (~0.9) content preservation score means that the model was not able to successfully learn to transfer the form and the target sentence is almost identical to the source sentence. The Shakespeare dataset is the hardest for the model in terms of transfer strength, probably because it is the smallest dataset, but the proposed method performs consistently well in transfer of both form and meaning and, in contrast to the baseline.

Storing meaning in the form vector Note that, theoretically, nothing is stopping the model from storing the meaning in the form vector, except from the size limitations, which would ensure that storing non-form-related information elsewhere would improve model performance. Figure 2 shows that as the meaning vectors get smaller, and the form vectors larger, the higher is transfer strength and the lower is content preservation. If the model would store meaning in the form vector, then the reduction in size of the meaning vector would not have negative impact on content preservation. This shows that the model tends to not store the meaning in the form vector.

Nevertheless, to force this behaviour we experimented with adding one more discriminator D_f. This discriminator works on the form vector $\mathbf{f}$ in the same manner as the discriminator D works on the meaning vector $\mathbf{m}$. Namely, during the training it tries to predict the meaning of a sentence from its form vector: $\boldsymbol{u} = D_f(\mathbf{f})$. Note that the vectors $\boldsymbol{u}$ and $\mathbf{m}$ are completely different. $\mathbf{m}$ is the meaning of a sentence for the purpose of the model, whereas $\boldsymbol{u}$ are pre-defined meaning of a sentence for training of the discriminator. In the simplest case, $\boldsymbol{u}$ can be a multi-hot representation of the input sentence, with the exception of pre-defined "style" words, which would always have 0 in the corresponding dimension, as it is done by John et al. (2018).

We, however, take a different approach. First, we find the "form" dimensions in the used word embeddings by taking the argmax of the difference between averaged word embeddings of the sentences from two forms (i.e. Early Modern English and contemporary English). Next, for a given sentence we discard the top-k tokens with the maximum and minimum values in those dimensions. Finally, we average word embeddings of the remaining tokens in the sentence to get the vector $\boldsymbol{u}$.

Aye, sir. (EME)	→	Yes, sir. (CE)
Fare thee well, my lord (EME)	→	Fare you well, my lord (CE)
This guy will tell us everything. (CE)	→	This man will tell us everything. (EME)
I've done no more to caesar than you will do to me. (CE)	→	I have done no more to caesar than, you shall do to me. (EME)

Table 1: Decoding of the source sentence from Early Modern English (EME) into contemporary English (CE), and vice versa.

A review: detection techniques for LTE system	⤨	Crisis management: media practices in telecommunication management
Situation management knowledge from social media		A review study against intelligence internet
Security flaw could not affect digital devices, experts say	⤨	Semantic approach approach: current multimedia networks as modeling processes
Semantic approach to event processing		Security flaw to verify leaks

Table 2: Flipping the meaning and the form embeddings of two sentence from different registers. Note the use of colon in the first example, and the use of the "to"-constructions in the second example, consistent with the form of the source sentences.

Such incorporation of the discriminator D_f helped to mitigate this issue.

Fluency of generated sentences Note that there is no guarantee that the generated sentences would be coherent after switching the form vector. In order to estimate how this switch affects the fluency of generated sentences, we trained a language model on the Shakespeare dataset and calculated the perplexity of the generated sentences using the original form vector and the average of form vectors of k random sentences from the opposite form (see subsubsection 5.1.1). While the perplexity of such sentences does go up, this change is not big (6.89 vs 9.74).

6.1 Impact of the motivational training

To investigate the impact of the motivator, we visualized form and meaning embeddings of 1000 random samples from the Headlines dataset using t-SNE algorithm (Van Der Maaten, 2014) with the Multicore-TSNE library (Ulyanov, 2016). The result is presented in Figure 3.

There are three important observations. First, there is no clear separation in the meaning embeddings, which means that any accurate form transfer is due to the form embeddings, and the dissociation of form and meaning was successful.

Second, even without the motivator the model is able to produce the form embeddings that are clustered into two groups. Recall from section 4 that without the motivational loss there are no forces that influence the form embeddings, but nevertheless the model learns to separate them.

However, the separation effect is much more pronounced in the presence of motivator. This explains why the motivator consistently improved transfer strength of ADNet, as shown in Figure 2.

6.2 Qualitative evaluation

Table 1 and Table 2 show several examples of successful form/meaning transfer achieved by ADNet. Table 1 presents the results of an experiment that to some extent replicates the approach taken by the authors who treat linguistic form as a binary variable (Shen et al., 2017; Fu et al., 2018). The sentences the original Shakespeare plays were averaged to get the "typical" Early Modern English form vector. This averaged vector was used to decode a sentence from the modern English translation back into the original. The same was done in the opposite direction.

Table 2 illustrates the possibilities of ADNet on fine-grained transfer applied to the change of register. We encoded two sentences in different registers from the Headlines dataset to produce form and meaning embeddings, and then decoded the first sentence with the meaning embedding of the second, and vice versa. Table 2 shows that the model correctly captures the meaning of sentences and decodes them using the form of the source sentences, preserving specific words and the structure of the source sentence. Note that in the first example, the model decided to put the colon after the "crisis management", as the source form sentence has this syntactic structure ("A review:"). This is not possible in the previously proposed models, as they treat form as just a binary variable.

6.3 Performance of meaning embeddings on downstream tasks

We conducted some experiments to test the assumption that the derived meaning embeddings should improve performance on downstream tasks

(a) Meaning embeddings

(b) Form embeddings

Figure 4: t-SNE visualization of the form and meaning embeddings. Each color corresponds to a different author.

BoW	Seq2Seq	InferSent	Fu et al. (2018)	ADNet
80.82	74.68	**83.17**	78.88	81.38

Table 3: F1 scores on the task of paraphrase detection using the SentEval toolkit (Conneau et al., 2017)

that require understanding of the meaning of the sentences regardless of their form. We evaluated embeddings produced by the ADNet, trained in the Headlines dataset, on the paraphrase detection task. We used the SentEval toolkit (Conneau et al., 2017) and the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). The F1 scores on this task for different models are presented in Table 3. Note that all models, except InferSent, are unsupervised. The InferSent model was trained on a big SNLI dataset, consisting of more than 500,000 manually annotated pairs. ADNet achieves the the highest score among the unsupervised systems and far outperforms the regular sequence-to-sequence autoencoder.

6.4 Multiple forms and stylistic similarities

In order to go beyond just two different forms, we experimented with training the model on a set of literature novels from six different authors from Project Gutenberg[4] written in two different time periods. A t-SNE visualization of the resulting meaning and form embeddings is presented in Figure 4. Note how form embeddings create a six-pointed star. After further examination, we observed that common phrases (for example, “Good morning” or “Hello!”) were embedded into the center of the star, whereas the most specific sentences from a given author were placed into the rays of the star. In particular, some sentences included character names, thus further research is required to mitigate this problem. Stamatatos (2017) provides a promising direction for solving this.

7 Conclusion

We presented ADNet, a new model that performs adversarial decomposition of text representation. In contrast to previous work, it does not require a parallel training corpus and works directly on hidden representations of sentences. Most importantly, it does not treat the form as a binary variable (as done in most previously proposed models), enabling a fine-grained change of the form of sentences or specific aspects of meaning. We evaluate ADNet on two tasks: the shift of language register and diachronic language change. Our solution achieves superior results, and t-SNE visualizations of the learned meaning and form embeddings illustrate that the proposed motivational loss leads to significantly better separation of the form embeddings.

References

Samuel Albanie, Sébastien Ehrhardt, and João F Henriques. 2017. Stopping gan violence: Generative unadversarial networks. *arXiv preprint arXiv:1703.02528*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representation*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Keith Carlson, Allen Riddell, and Daniel Rockmore. 2017. Zero-shot style transfer in text using recurrent neural networks. *arXiv preprint arXiv:1711.04731*.

---

[4] `http://www.gutenberg.org/`

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Dua Dheeru and Efi Karra Taniskidou. 2017. UCI machine learning repository.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

M. A. K. Halliday, A. McIntosh, and P. Stevens. 1968. *The Linguistic Sciences and Language Teaching*. Longmans, Green and Co., London.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *arXiv preprint arXiv:1707.01161*.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for text style transfer. *arXiv preprint arXiv:1808.04339*.

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative concatenative nets jointly learn to write and classify reviews. *arXiv preprint arXiv:1511.03683*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373.

William Safire. 1995. On Language – Gotcha! Gang Strikes Again.

Ferdinand de Saussure. 1959. *Course in General Linguistics*. New York : Philosophical Library.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Efstathios Stamatatos. 2017. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1138–1149.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.

Dmitry Ulyanov. 2016. Multicore-tsne. `https://github.com/DmitryUlyanov/Multicore-TSNE`.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. *Proceedings of COLING 2012*, pages 2899–2914.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2223–2232.

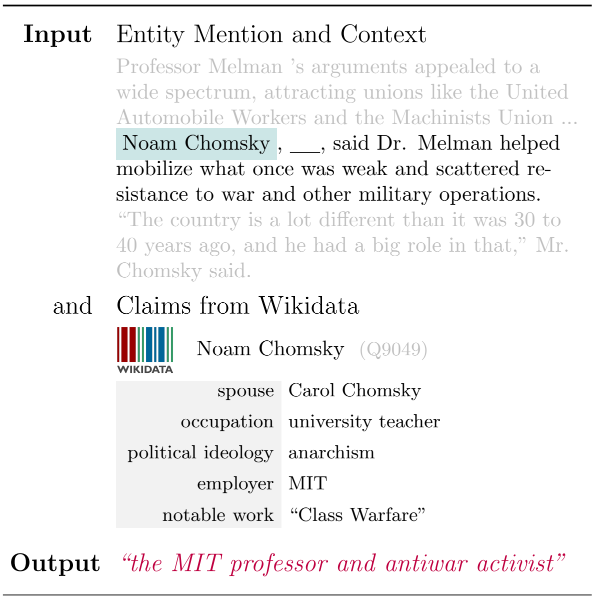
PoMo: Generating Entity-Specific Post-Modifiers in Context

Jun Seok Kang[1], **Robert L. Logan IV**[2], **Zewei Chu**[3], **Yang Chen**[3],
Dheeru Dua[2], **Kevin Gimpel**[4], **Sameer Singh**[2], **Niranjan Balasubramanian**[1]
[1]Stony Brook University, NY, USA
[2]University of California, Irvine, CA, USA
[3]University of Chicago, IL, USA
[4]Toyota Technological Institute at Chicago, IL, USA
`{junkang,niranjan}@cs.stonybrook.edu`
`{rlogan,ddua,sameer}@uci.edu`
`{zeweichu,yangc1}@uchicago.edu`
`{kgimpel}@ttic.edu`

Abstract

We introduce entity post-modifier generation as an instance of a collaborative writing task. Given a sentence about a target entity, the task is to automatically generate a post-modifier phrase that provides contextually relevant information about the entity. For example, for the sentence, "Barack Obama, ______, supported the #MeToo movement.", the phrase "a father of two girls" is a contextually relevant post-modifier. To this end, we build `PoMo`, a post-modifier dataset created automatically from news articles reflecting a journalistic need for incorporating entity information that is relevant to a particular news event. `PoMo` consists of more than 231K sentences with post-modifiers and associated facts extracted from Wikidata for around 57K unique entities. We use crowdsourcing to show that modeling contextual relevance is necessary for accurate post-modifier generation.

We adapt a number of existing generation approaches as baselines for this dataset. Our results show there is large room for improvement in terms of both identifying relevant facts to include (knowing which claims are relevant gives a $> 20\%$ improvement in BLEU score), and generating appropriate post-modifier text for the context (providing relevant claims is not sufficient for accurate generation). We conduct an error analysis that suggests promising directions for future research.



Input Entity Mention and Context

Professor Melman 's arguments appealed to a wide spectrum, attracting unions like the United Automobile Workers and the Machinists Union ... Noam Chomsky , ___, said Dr. Melman helped mobilize what once was weak and scattered resistance to war and other military operations. "The country is a lot different than it was 30 to 40 years ago, and he had a big role in that," Mr. Chomsky said.

and Claims from Wikidata

Noam Chomsky (Q9049)

spouse	Carol Chomsky
occupation	university teacher
political ideology	anarchism
employer	MIT
notable work	"Class Warfare"

Output *"the MIT professor and antiwar activist"*

Figure 1: Post-Modifier Generation Task

1 Introduction

The goal of machine-in-the-loop writing systems is to assist human writers by directly augmenting their text. Examples include systems that refine human text for grammar (Rao and Tetreault, 2018), collaborate on story plot generation systems (Clark et al., 2018; Yu and Riedl, 2012), or modify the content for style (Hu et al., 2017; Shen et al., 2017; Yang et al., 2018). In this paper, we introduce post-modifier generation as an instance of such an assistive writing task in the news domain. Journalists use post-modifiers to introduce background information about entities discussed in news articles. To write these post-modifiers journalists often need to look up relevant facts about entities. A post-modifier generation system can be seen as a collaborative assistant that automatically finds relevant facts and inserts a small text fragment that augments the text produced by the human writer.

Post-modifier generation is a *contextual* data-to-text generation problem, where the data is the set of known facts about the target entity, and the text to be generated is a post-modifier that is relevant to the rest of the information conveyed in the text. Figure 1 shows an example. Given a sentence about the anti-war resistance work of Noam Chomsky, the target entity, and a set of known facts about him, the task is to generate a post-modifier that introduces Chomsky as a professor and mentions

Proceedings of NAACL-HLT 2019, pages 826–838
Minneapolis, Minnesota, June 2 - June 7, 2019.

his background as an anti-war activist. An effective post-modifier generation system must: (i) select suitable facts about the entity given the text, and (ii) produce text that covers these facts in a way that fits in with the rest of the text.

We introduce `PoMo`, an automatically generated dataset for developing post-modifier generation systems.[1] `PoMo` is a collection of sentences that contain entity post-modifiers, along with a collection of facts about the entities obtained from Wikidata (Vrandečić and Krötzsch, 2014). We use a small number of dependency patterns to automatically identify and extract post-modifiers of entities in sentences. We then link the extracted entities with the entries in Wikidata. The resulting dataset has 231,057 instances covering 57,966 unique entities. Our analysis show that the post-modifiers often combine multiple facts and are specific to the sentential context.

We conduct two sets of experiments that highlight the challenges in post-modifier generation. **(i) Claim Selection:** Given an input sentence, the first step in generating a post-modifier is to figure out which facts to use. We formulate this as a distantly-supervised ranking problem, where we train neural models that learn to identify relevant claims for a given sentence. These claim ranking models perform well when predicting the relevance of coarse-grained facts (e.g. occupation), but fare poorly when predicting finer-grained facts (e.g. place of birth). **(ii) Generation:** We adapt recent sequence-to-sequence generation models for this task. Results show that generation remains a challenge. Even though our automatic claim ranking does not improve generation, further experiments with oracle selected claims demonstrate that when relevant claims are known, the models can generate post-modifiers which humans deem comparable in quality to ones written by professional journalists.

In summary, the main contributions of this work are: 1) a data-to-text problem that introduces new challenges, 2) an automated dataset creation pipeline and a large resulting dataset, 3) a crowdsourcing study that verifies the contextual relevance of post-modifiers, and 4) a characterization of the difficulty of the task via performance analysis of numerous baselines.

	CNN	DM	NYT	**Total**
Train	6,557	11,323	202,735	**220,615**
Valid	162	267	4,771	**5,200**
Test	181	288	4,773	**5,242**
Total	6,900	11,878	212,279	**231,057**

Table 1: Dataset distribution by sources.

2 PoMo: Task and Dataset

Post-modifier generation can be formulated as a data-to-text generation problem. The input is text mentioning a target entity and a set of known facts about the entity. The output is a phrase that: (i) fits as a post-modifier of the target entity mentioned in the input text, and (ii) conveys a subset of facts relevant to the context of the input text.

Figure 1 shows an example for the target entity `Noam Chomsky`. The input includes a sentence mentioning Chomsky's work on mobilizing anti-war groups along with its surrounding context, and a listing of all facts about Chomsky that are available in Wikidata. Given these inputs, the task is to output a post-modifier phrase that conveys facts about Chomsky that fit within the sentence. In this example the post-modifier conveys both general background information about Chomsky (his occupation), and specific information relevant to the context of the sentence (being an anti-war activist).

This task can be seen as an instance of collaborative writing, where the journalist writes text about specific news events involving entities, and the generation system assists the journalist by inserting new text that augments the story. Given a large collection of news articles, we can automatically create training data for such systems by removing the pieces of text that we want the assistant to generate. This requires reliable ways to identify text to remove and sources of information that can be used to generate the text. Here we describe a pipeline for generating such a dataset for our task.

2.1 Dataset

We construct the `PoMo` dataset using three different news corpora: NYTimes (Sandhaus, 2008), CNN and DailyMail (Hermann et al., 2015). We use Wikidata to collect facts about entities.[2]

---

[1] `https://stonybrooknlp.github.io/PoMo/`

[2] Wikidata dump from `https://www.wikidata.org/wiki/Wikidata:Database_download` (Dump date: 2018/06/25)

2.1.1 Post-Modifier and Entity Identification

We use Stanford CoreNLP (Manning et al., 2014) to parse each sentence in the news articles and to identify named entities. We extract post-modifiers by finding noun phrases that share an *appos* relation[3] with any recognized named entity in the sentence. In this work, we only consider post-modifiers for *people*. In the future, we plan to expand `PoMo` to include more post-modifiers for other targets, such as organizations. We extract only one such pair from a given sentence to reduce the possible noise in the extraction process.

In our running example from Figure 1, `Noam Chomsky` is recognized as a person entity. The word "professor" is an appositive dependency of the word "Chomsky" and therefore, we extract the NP "the Massachusetts Institute of Technology professor and antiwar activist" which includes the word "professor" as a post-modifier for the target entity `Noam Chomsky`.

2.1.2 Entity Claim Matching

Wikidata provides information about entities in the form of key-value pairs that are called *claims*. To collect the facts about a target entity, we need to link the target to a specific entity in Wikidata. We first search through Wikidata labels and aliases to find candidates with the same name as the target. We sort the candidates based on the number of claims that have a significant word overlap with the extracted post-modifier. We link the entity to the highest ranked candidate whose claims cover at least 30% of the non stop words in the post-modifier. If such a candidate is found we record the claims that overlap with the post-modifier. If no such candidate is found then we discard the entity.

We evaluate this simple heuristic by comparing the results to using an off-the-shelf entity linking system AIDA-light (Nguyen et al., 2014) and show the results in Table 2. We find that AIDA-light agrees with our entity linking in 91.2% of the cases. AIDA-light is able to link 94.3% of the entities we found from NYTimes, but for CNN and DailyMail, it links only 87.0% and 86.34% of the entities, respectively. This decrease is likely due to the fact that AIDA-light was last updated in 2014 while the CNN/DailyMail datasets contain articles collected until the end of April 2015. On the other hand, NYTimes articles range from 1987 to 2007. Our heuristic seems to be reasonably reliable as it does not depend on anything else but the data sources: news articles and Wikidata.

	AIDA Succ.	Agreement
Overall	93.66	91.22
Train	93.65	91.16
Valid	94.06	91.13
Test	93.80	93.65
CNN	87.03	90.34
DM	86.34	85.66
NYT	94.29	91.53

Table 2: Percent agreement with AIDA-light's named entity disambiguation results.

2.2 Analysis

Table 1 shows the distribution of the data sources over train, validation, and test sets. All splits maintain the relative distributions of the data sources to prevent stylistic mismatches from influencing generation. We also ensure that there is no entity overlap among the splits. Within the NYTimes data, we verify that the distribution over years between 1987 and 2007 is also similar over the sets.

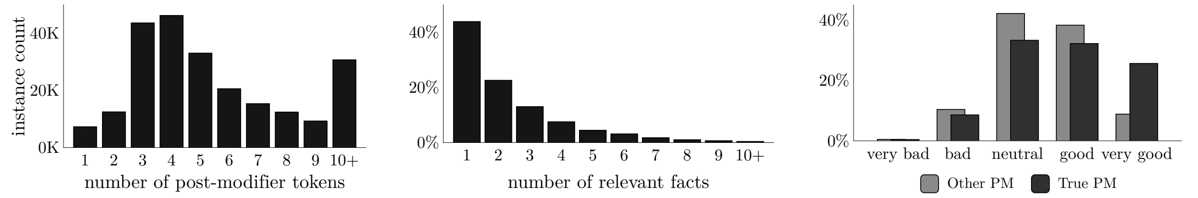
Distribution of Post-Modifiers and Entities Figure 2a shows the distribution of post-modifier lengths in terms of token counts. Most post-modifiers are three to eight words long, and about 17.3% are even longer. Figure 2b shows an estimate of the number of relevant facts covered by the post-modifiers; this estimate uses the number of claims that overlap with the post-modifier via heuristic matching. More than half of the post-modifiers convey two or more facts. About 11.4% convey five or more facts. These results suggest that generating post-modifiers requires composing together multiple relevant facts.

Table 3 lists the most frequent types of facts used in the post-modifiers in our dataset. Most relate to generic biographical information such as the entity's occupation, organizations they belong to, place of birth, etc. Here again we see a range of types of information being conveyed which is likely to present a challenge for generation systems.

The dataset also covers a wide variety of entity types. We cluster the target entities by their occupation listed in Wikidata. We also use WordNet (Miller, 1995) to traverse the hypernyms of the words to find frequent ones. Then, we manually select the top ten occupation types. Any entity that

---

[3] An *appos*itional modifier of an NP is another NP immediately to the right that defines or modifies the NP.



(a) Histogram of the token counts of the post-modifiers. Majority of the post-modifiers (171K instances, 74.14%) have 3 to 8 tokens. Average is 5.8 tokens.

(b) Number of relevant facts per instance in the dataset. More than a half of the post-modifiers are related to two or more facts.

(c) Histogram of the scores for post-modifiers, averaged over three annotations. The distribution of ratings for true and other post-modifiers.

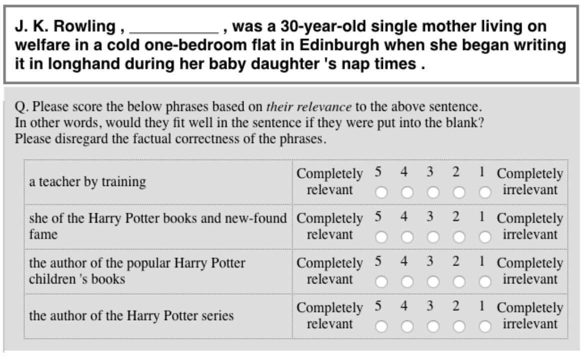
Figure 2: `PoMo` Post-Modifier Statistics

Fact Type	**Count**
position held	151,959
occupation	82,781
educated at	53,067
member of political party	42,416
member of sports team	41,602
employer	36,412
award received	31,618
position played on team / speciality	23,987
country of citizenship	17,444
nominated for	15,139
place of birth	9,185
participant of	8,520
member of	7,565
languages spoken, written or signed	4,827
place of death	4,071

Table 3: Top 15 frequent fact types based on heuristic fact coverage identification.

Occupation	**Count**	**Percentage**
athlete	13,560	23.39%
writer	9,177	15.83%
politician	8,518	14.69%
entertainer	6,488	11.19%
other	5,870	10.13%
scientist	4,487	7.74%
artist	4,175	7.20%
official	2,098	3.62%
lawyer	1,132	1.95%
educator	961	1.66%
capitalist	789	1.36%
scholar	711	1.23%

Table 4: Distribution of the inferred occupations of the target entities. Entities clustered by their occupation.



J. K. Rowling , __________ , was a 30-year-old single mother living on welfare in a cold one-bedroom flat in Edinburgh when she began writing it in longhand during her baby daughter 's nap times .

Q. Please score the below phrases based on *their relevance* to the above sentence.
In other words, would they fit well in the sentence if they were put into the blank?
Please disregard the factual correctness of the phrases.

a teacher by training	Completely relevant	5 ○	4 ○	3 ○	2 ○	1 ○	Completely irrelevant
she of the Harry Potter books and new-found fame	Completely relevant	5 ○	4 ○	3 ○	2 ○	1 ○	Completely irrelevant
the author of the popular Harry Potter children 's books	Completely relevant	5 ○	4 ○	3 ○	2 ○	1 ○	Completely irrelevant
the author of the Harry Potter series	Completely relevant	5 ○	4 ○	3 ○	2 ○	1 ○	Completely irrelevant

Figure 3: Screenshot of the crowdsourcing task. We asked crowd to rate the quality of post-modifiers.

does not belong to the top ten is assigned to a single *other* group. The resulting distribution is shown in Table 4.

Quality of Post-Modifiers We conduct a crowdsourcing study to understand how often the post-modifiers are specific to the particular context. For each (entity, context, post-modifier) triple in the validation set, we create multiple alternative post-modifiers by randomly choosing up to ten other post-modifiers that are found in some other sentences for the same entity. Crowd workers rate the quality of these post-modifiers. Figure 3 shows a screenshot of a task given to crowd workers. If the true post-modifier, the one that is actually used in the context, is rated the highest compared to the rest, then we assume the post-modifier is indeed specific to the context. On the other hand, if the crowd workers rate multiple other post-modifiers as good fits for the context, then the true post-modifier is not context specific. Figure 2c shows the distribution of ratings for true and other post-modifiers. The true post-modifiers tend to be rated *very good* or *good* more often than the other post-modifiers. This suggests that in many cases post-modifiers are specific to the context and cannot be simply replaced by other post-modifiers.

3 Relevant Claim Selection

One of the key challenges of generating post-modifiers is to identify the claims about an entity that are relevant to the given context. In this section, we explore methods for solving this task.

3.1 Methods

We consider three different models: a most-common claim baseline and two neural baselines.

Most-Common Claim This model employs a simple frequency heuristic: rank claims by the frequency of their types in the training post-modifiers (e.g. as in the order given in Table 3) and deem the top n claims in this ranking as relevant.

Neural Baselines We use two neural baselines with the following architecture. Word embeddings are used to represent words in the context (e.g. current and previous sentence) and claims. The sequences of embeddings are then fed through 2-layer LSTM's (Hochreiter and Schmidhuber, 1997) to obtain separate representations of the context and claims. These representations are subsequently concatenated together and fed through a fully-connected layer with sigmoid activation, producing a scalar value for each claim representing the probability that it is relevant. We use this model in two ways: as a classifier, and as a ranking model. When used as a classifier, any claim whose score exceeds a threshold τ is predicted to be relevant. When used as a ranking model, the top n highest-scoring claims are predicted to be relevant.

3.2 Experiments

We train our baselines on the `PoMo` dataset, using the claims detected during dataset collection as a (distant) source of supervision. Precision, recall, and F_1 score are used to evaluate model performance. Model hyperparameters are chosen using (coarse) grid search to maximize F_1 score on the validation set. The neural baselines use a vocabulary size of 50,000, 100-dimensional word embeddings, and 256 hidden units in the LSTM layers. Dropout (Srivastava et al., 2014) is applied between the LSTM layers with a 0.5 keep probability. The neural classifier uses threshold $\tau = 0.37$. We find the optimal value of n is 4 for the most-common claims model and 2 for the neural ranker.

Quantitative results are provided in Table 5. Both neural baselines perform considerably better than the most-common claims model. This indicates that the provided contexts and claim values contain useful information for claim selection that goes beyond the information captured by global statistics of the dataset alone. We additionally observe that the ranking-based approach outperforms the classification-based approach in terms of both precision and F_1 score, while having only slightly worse recall.

	Prec.	**Recall**	**F_1**
Most-Common Claim (n=4)	39.9	51.6	45.0
Neural Classifier (τ=0.37)	52.0	63.8	57.4
Neural Ranker (n=2)	66.5	62.7	64.5

Table 5: Baseline model performance on the claim selection task.

Fact Type	**F_1**
employer	76.95
position played on team / speciality	76.65
position held	63.10
occupation	50.02
member of political party	48.71
member of	45.60
member of sports team	38.53
award received	37.53
nominated for	30.87
educated at	29.56
participant of	29.04
country of citizenship	16.28
place of death	14.72
place of birth	6.80
languages spoken, written or signed	0.00

Table 6: F_1 score of neural ranker ($n = 2$) on top 15 fact types.

To better understand the cases where the neural models fail and succeed, we examine the distribution of F_1 scores over the top 15 fact types (see Table 6). Interestingly, when ranked by F_1 score we observe that fact types fall naturally into topically related groups:

1. position / occupation-related facts: *position played, position held, occupation*
2. membership-related facts: *member of political party, member of, member of sports team*
3. achievement-related facts: *award received, nominated for*
4. location-related facts: *country of citizenship, place of death, place of birth*

With the exception of *employer*, the overarching trend is that the model identifies the relevance of coarse-grained claims better than fine-grained claims (e.g occupations, political parties, and sports positions are much more likely to be shared between entities than birth and death places). This suggests that developing better methods for determining the relevance of fine-grained claims is a promising avenue for future research on this task.

4 Post-Modifier Generation

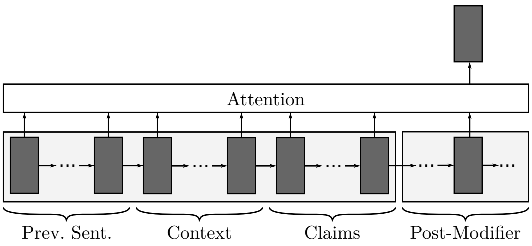
We move our focus to the main task of post-modifier generation.

4.1 Methods

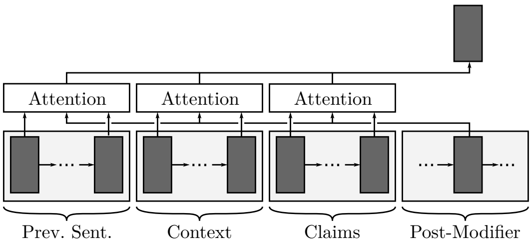
At its core, post-modifier generation involves producing a variable-length sequence output conditioned on two variable-length inputs: the words in the current and previous sentence (e.g. the context), and the collection of claims about the entity. Accordingly, the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014) is a natural fit for the task — we use it as the foundation for all of our baseline models. Since research has shown that attention (Bahdanau et al., 2015) and copy mechanisms (Gu et al., 2016) consistently improve seq2seq model performance, we use these in our baselines as well.

One choice that must be made when using this framework is how to combine the different inputs. The default approach we use is to concatenate the claim and context into a linear sequence of tokens during preprocessing (shown in Figure 4a). We also experiment with encoding the claims and each of the context sentences separately, then concatenating their vector representations before decoding. We refer to this as the *tri-encoder* approach (shown in Figure 4b).

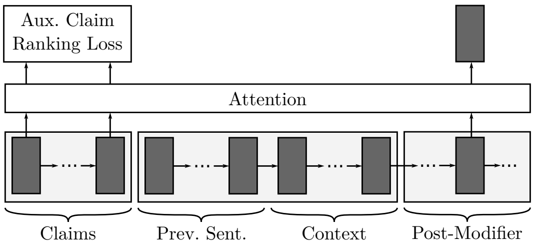
As discussed earlier, selecting relevant claims is crucial to generating good post-modifiers. One way to incorporate claim selection is to use our baseline models from Section 3 to cut out irrelevant claims from the input before feeding them to the encoder (e.g. performing hard claim selection). This pipelined approach is not differentiable, and can suffer from cascading errors. An alternative way is to use the model's attention mechanism as a form of soft claim selection that attends only to the relevant claims. The drawback of this approach is that it does not make use of the available claim annotations, which are an important source of supervision.



(a) Basic sequence-to-sequence model



(b) Tri-encoder model



(c) End-to-end claim selection model

Figure 4: `PoMo` Models for post-modifier generation. Grey boxes at the bottom represent individual encoder/decoder modules. (a) For baseline BiLSTM and transformer models all inputs are concatenated into one sequence. (b) The tri-encoder model has a separate encoder and attention for each type of input. The outputs of attention layers are concatenated together before generation. (c) The end-to-end claim selection model attends to only the claim embeddings and uses an auxiliary loss term to encourage high attention scores for relevant claims.

Building on these observations, we propose an *end-to-end claim selection* model which incorporates an additional term to the loss function that encourages the claim-level attention probabilities to be higher for the identified relevant claims as shown in Figure 4c. The process for computing this loss term works as follows. We begin by summing together attention scores for tokens within claims to obtain a claim-level score. These scores are then fed through a sigmoid activation function to obtain a soft claim selection probability. For each claim, we measure the binary cross entropy between the predicted selection probability and a binary variable indicating whether or not the claim was identified as relevant. The final loss term is the average of these binary cross entropies. Note that we do not use a copy mechanism in this model to avoid double-counting (since relevant claims were identified using word overlap).

4.2 Experiments

We experiment with two types of encoder/decoder modules: bidirectional LSTMs, and transformers (Vaswani et al., 2017). We use a vocabulary of size 50K, truncate the maximum input sequence length to 500, and use a batch size of 32 in all experiments. To help models distinguish between claims and context we demarcate claim fields with special `<claim>`, `<key>`, and `<value>` tokens. We train all the models for 150k steps, and evaluate on the validation dataset every 10k steps. Evaluation is performed using the BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) translation metrics, and Precision, Recall and F_1 score of the predicted bag-of-words (omitting stopwords). The model with the highest F_1 score on the validation set is used during test time.

For the bidirectional LSTM, we use 2 hidden layers with 512 hidden units, 500-dimensional word embeddings, and apply dropout between layers with a keep probability of 0.7. Models are trained using stochastic gradient descent with a learning rate of 1.0. For the transformer model, we use 4 attention heads, 4 layers of transformer blocks with 64 hidden units for the encoder and the decoder, a penultimate hidden layer with 256 units, and 64-dimensional word embeddings. Transformer models are trained using Adam (Kingma and Ba, 2015) with an initial learning rate of 2.0, and a label smoothing (Szegedy et al., 2016) factor of 0.1 when calculating loss.

We perform a variety of experiments, the results of which are displayed in Table 7. In this table, *Transformer* and *BiLSTM* refer to models trained using the default approach to combining context and claims, while *Tri-encoder* refers to a BiLSTM model trained using the approach described in 4.1 (we do not train a transformer version since its performance is lackluster). Here are detailed descriptions of the experiments performed in each section:

- **All Claims**: Results for vanilla seq2seq models.
- **Oracle**: Hard claim selection is performed using the oracle relevant claims.
- **Neural Ranker** ($n = 10$): Hard claim selection is performed using the top-10 claims returned by the neural ranker baseline.
- **End-to-End Claim Selection**: Results for the end-to-end claim selection model.

In order to understand the relative contribution of the different inputs, we also include results for the BiLSTM model trained using either only the claims, or only the context sentences. In Figure 5 and 6, we show the performances by post-modifier and sentence lengths to examine the impact of the such variables.

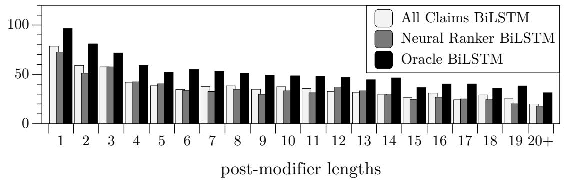
Discussion of Quantitative Results Our results contain a few key findings. The first is that knowing the relevant claims is critical to obtaining state-of-the-art performance; even knowing only oracle claims is sufficient to perform better than all of the other baselines, although there is a still a large improvement when context is additionally provided. However, model-based approaches for claim selection do not seem to help: hard claim selection using the neural ranker performs just as well as the vanilla models, and our proposed approach for end-to-end claim selection has a negative impact. This motivates the need for more effective methods of claim selection. The decreasing performances of the BiLSTM seq2seq models by the increasing target post-modifier and sentence lengths show the difficulty of generating long texts and handling long input data. Finally, we observe that the transformer-based seq2seq models are not particularly well-suited to this task. In all cases their performance is inferior to the BiLSTM-based approaches. Large-scale, pre-trained transformer-based language models, such as GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2018), might be an interesting addition to the baselines, by framing the task as filling in the blanks for post-modifiers. However, when restricted to approaches that only use our dataset for training, we expect those based on language models to struggle due to the separation of entities among train, validation, and test.

Qualitative Analysis A cursory examination of model predictions (see Table 8 for examples) provides insight into why post-modifier generation is a challenging task. One issue that consistently appears is temporal inconsistency between the target and generated post-modifiers. That is, the model may make an error since it is unaware of the time period that the article is written in (and also may not be aware of the periods of time for which a claim are true). For example, in the first instance in Table 8 the Oracle model predicts an almost correct post-modifier but misses the fact that Kenneth Clarke is a *former* Chancellor of the Exchequer. Another apparent issue is that models tend to generate shorter post-modifiers than humans. As is indicated in Figure 2a the post-modifiers in the dataset on average contain 5.8 tokens, whereas generated post-modifiers have only 3.8. Lastly, we

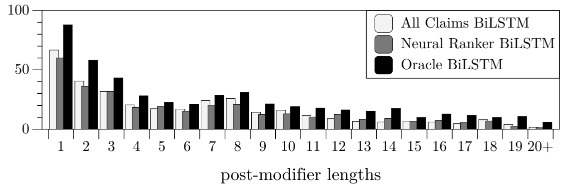
	Prec.	Rec.	F_1	BLEU	MET.
All Claims					
Transformer	41.9	22.2	29.0	7.0	12.1
Tri-Encoder	53.9	32.4	40.5	17.0	17.6
BiLSTM	51.1	34.7	41.4	19.4	18.8
Oracle					
Transformer	69.4	38.6	49.6	15.7	20.0
Tri-Encoder	68.8	47.3	56.1	24.0	24.5
BiLSTM	66.4	48.8	56.2	25.1	25.3
Neural Ranker ($n = 10$)					
Transformer	41.5	22.4	29.1	6.9	12.1
Tri-Encoder	53.5	34.1	41.6	17.6	18.3
BiLSTM	49.0	34.2	40.3	18.5	18.5
End-to-End Claim Selection					
BiLSTM	47.5	27.9	35.2	13.7	15.3
Context Only					
BiLSTM	13.3	8.5	10.3	3.4	6.2
Claims Only					
BiLSTM	47.3	28.5	35.6	13.5	15.0
Oracle Claims Only					
BiLSTM	63.8	44.7	52.5	21.3	22.7

Table 7: Post modifier generation model performances with seq2seq models. Precision, recall and F_1 scores are computed ignoring stopwords.

observe that our quantitative evaluation metrics can be too strict. Take for example the second instance in Table 8. Here the content of the target and generated post-modifiers is almost exactly the same, however our metrics would give very low scores due to low overlap.

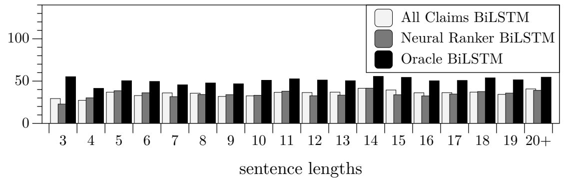


(a) F_1 scores

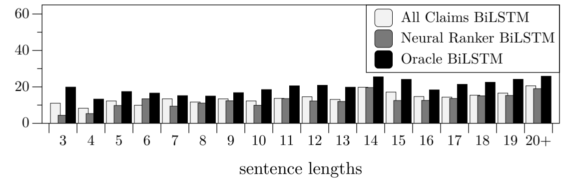


(b) BLEU scores

Figure 5: Performances by target post-modifier lengths of BiLSTM model. Post-modifiers with 20 or more tokens are put into one group, 20+.



(a) F_1 scores



(b) BLEU scores

Figure 6: Performances by input sentence lengths of BiLSTM model. Sentences with 20 or more tokens are put into one group, 20+.

Human Evaluation We additionally evaluate the generated post-modifiers by performing a human evaluation using Amazon Mechanical Turk. We randomly select 500 instances from test set and show crowdworkers the sentence context, along with the true post-modifier and a generated one. For each instance, workers are asked to select the better phrase, or indicate that the two phrases are of equal quality. For the Oracle BiLSTM model, the true post-modifiers are preferred 46% of the time, while generated post-modifiers are preferred 43.2% of the time. For the Neural Ranker ($n = 10$) BiLSTM model, true post-modifiers are favored much more (57.60%) than the generated ones (20%). Consistent with our quantitative results, we see that claim selection is a crucial factor in this task. We also observe a few trends in the results. People tend to prefer generated post-modifiers over the ones written by professional journalists when they are shorter and to use more general terms without elaborating too much about the entity. In contrast, longer and more detailed human written post-modifiers are preferred when they are especially relevant to the rest of the sentence.

5 Related Work

There is a large body of previous work on claim selection (Kukich, 1983; Duboue and McKeown, 2003; Reiter and Dale, 1997; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005) and language generation from structured data (Reiter et al., 2005; Goldberg et al., 1994). Initially, hand-crafted grammars were employed for language generation, which later evolved to statistical machine translation style models (Wong and Mooney, 2007) or PCFG based models (Belz, 2008). More recently, the focus has shifted to learning both fact selection and language generation jointly (Liang et al., 2009; Angeli et al., 2010; Kim and Mooney, 2010; Lu and Ng, 2011; Konstas and Lapata, 2013).

Input	Sky News reported Thursday night that Kenneth Clarke , _______, had not yet decided whether to support Mr. Howard 's candidacy , raising the possibility the party could face a divisive battle for leadership .
Claims	+ (position held: *Chancellor of the Exchequer*) (position held: *Secretary of State for the Home Department*)
Target	a former chancellor of the exchequer
All Claims	the Home Secretary
Oracle	the Chancellor of the Exchequer
Input	" A lot of people think it 's something we just started , but we actually opened the season with our first drive using it against Indianapolis , " said Howard Ballard , _______.
Claims	+ (member of sports team: *Buffalo Bills*) + (position played on team / speciality: *offensive tackle*) (mass: *325 pound*) (height: *78 inch*)
Target	Buffalo 's robust , 6-foot-6-inch , 325-pound right tackle
All Claims & Oracle	the Bills ' offensive tackle

Table 8: **Challenging `PoMo` instances**. Two examples along with outputs of the best All Claims and Oracle models are displayed. Claims deemed relevant during dataset curation are prefaced with a +. In the first example, knowing the relevant claims helps the Oracle model produce an output that closely matches the Target, however lack of temporal information causes the model to miss the word *former*. In the second example, the All Claims and Oracle models produce the same post-modifier. Although it is similar to the Target in meaning, it receives a low score using our evaluation metrics. Futhermore, our data curation method fails to identify relevant claims.

Modern approaches employ neural networks to solve this problem end-to-end. Mei et al. (2016) utilize an encoder-decoder framework to map weather conditions to a weather forecast. Ahn et al. (2016) and Yang et al. (2017) introduce a new class of language models which are capable of entity coreference and copying facts from an external knowledge base. Building upon these models, Wiseman et al. (2017) introduce an auxiliary reconstruction loss which use the hidden states of the decoder to recover the facts used to generate the text. Liu et al. (2018) introduce a hierarchical attention model for fact selection, with the higher level focusing on which records in the table to select and the lower level focusing on which cells in a particular row to pay attention to.

In order to train complex neural models, the quest for larger datasets has become paramount. Lebret et al. (2016) introduce the WikiBio dataset containing Wikipedia articles of famous people and the corresponding infobox tables. One drawback of this dataset is that it is easily solved using template-based models. To address this issue, Wiseman et al. (2017) introduce the ROTOWire dataset, which contains summaries of basketball games that are very long and syntactically diverse. A comprehensive list of datasets is provided in Appendix B.

6 Conclusions and Future Work

Inspired by recent work on collaborative writing and data-to-text generation, we introduce post-modifier generation, a task that bridges the gap between these two fields. The task is to generate a factual description of an entity which fits within the context of a human written sentence. In order to promote research on this task we present `PoMo`, a large dataset of automatically extracted post-modifiers from news articles, aligned to the Wikidata knowledge graph. We study the performance of numerous strong baseline models on this dataset, with a particular focus on the specific sub-task of claim selection. Our results demonstrate that when relevant claims are known, sequence-to-sequence models are capable of generating post-modifiers which humans deem comparable in quality to ones written by professional journalists. However, according to both quantitative metrics and human judgment, performance is much lower when models must determine for themselves which claims are relevant. These experiments suggest plausible pathways to achieving human-level performance on this task that are both challenging and interesting problems for future research.

Acknowledgments

We would like to thank the Toyota Technological Institute at Chicago for hosting the Workshop on Collaborative and Knowledge-Backed Language Generation which initiated the efforts for this project. The authors would also like to thank David Yarowsky, Jason Eisner, Kevin Duh, Kyle Gorman, and Philipp Koehn for feedback on early ideas for post-modifier generation.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, pages 329–340. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. *LDOW*, 1184.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Sudha Rao and Joel R. Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1282–1288. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 172–179.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, Copenhagen, Denmark. Association for Computational Linguistics.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7287–7298. Curran Associates, Inc.

Hong Yu and Mark O. Riedl. 2012. A sequential recommendation approach for interactive personalized story generation. In *AAMAS*.

A Additional Claim Selection Materials

Table 9 lists the evaluation results of most-common claim baseline for n, the number of claims to predict, from 1 to 5. We obtain highest F1 with $n = 4$.

n	Precision	Recall	F1
1	57.7	31.8	41.0
2	49.6	38.3	43.2
3	43.4	45.2	44.3
4	39.9	51.6	45.0
5	36.0	56.8	44.0

Table 9: Evaluation metrics for most-common claim baseline for different values of n.

Neural baseline shows improved performance compared to most-common claim baseline, showing its best performance when $n = 2$.

n	Precision	Recall	F1
1	75.2	42.4	54.3
2	66.5	62.7	64.5
3	56.0	69.6	62.1
4	48.7	76.2	59.4

Table 10: Evaluation metrics for neural baseline for different values of n.

B Existing Data-to-Text Datasets

Table 11 provides a comprehensive list of data-to-text datasets. `PoMo` presents a different set of challenges from these datasets. While the target text is shorter and less diverse, the task adds an additional challenge of figuring out which claims to use, a task which our evaluation shows is quite challenging.

Dataset	**Size**	**Domain of structured data to language**
WEATHER.GOV	29.5k	Weather conditions to forecast report
ALLRECIPES	31k	Table of ingredients to recipes
ROBOCUP	1.5k	Game statistics to summaries
ROTOWIRE	4.9k	Basketball statistics to game summaries
WIKIBIO	728k	Infobox to Wikipedia biography articles
SBNations	10.9K	Game statistic to fan written summaries
WikiFacts	40k	Freebase /film/actor facts to Wiki description of actor

Table 11: A comparative analysis of various datasets.

Improved Lexically Constrained Decoding for Translation and Monolingual Rewriting

J. Edward Hu **Huda Khayrallah** **Ryan Culkin** **Patrick Xia**
Tongfei Chen **Matt Post** **Benjamin Van Durme**
Johns Hopkins University
`{edward.hu,huda,rculkin,paxia,tongfei}@jhu.edu; {post,vandurme}@cs.jhu.edu`

Abstract

Lexically-constrained sequence decoding allows for explicit positive or negative phrase-based constraints to be placed on target output strings in generation tasks such as machine translation or monolingual text rewriting. We describe vectorized dynamic beam allocation, which extends work in lexically-constrained decoding to work with batching, leading to a five-fold improvement in throughput when working with positive constraints. Faster decoding enables faster exploration of constraint strategies: we illustrate this via data augmentation experiments with a monolingual rewriter applied to the tasks of natural language inference, question answering and machine translation, showing improvements in all three.

1 Introduction

For many natural language generation tasks, we often know word(s) that should (or should not) be in the output sentence. Examples include terminology databases in Machine Translation (MT) (Hokamp and Liu, 2017), names (and generic responses) in dialogue generation (Li et al., 2016; Gu et al., 2016), objects in image captioning (Anderson et al., 2017), and facts in abstractive summarization (See et al., 2017). One approach to enforce hard lexical constraints in the output is to modify the inference procedure to enforce their presence directly (Hokamp and Liu, 2017). These constraints could be either *positive* (a word must appear in the output) or *negative* (a word must be avoided). While negative constraints could be easily enforced by preventing hypotheses with prohibited tokens from entering the beam, placing positive constraints in natural and meaningful ways is less straightforward. We improve upon previous work by vectorizing the dynamic beam allocation (DBA) algorithm from Post and Vilar (2018) and by incorporating *multi-state* tries, which track a subset of nodes at each decoding timestep. These improvements lead to a five-fold speedup in decoding with positive constraints and in some cases better constraint placements (with respect to BLEU).

Post and Vilar (2018) motivated the utility of lexically-constrained decoding in MT for scenarios such as interactive translation and domain adaptation. Translation applications handling large amounts of data will clearly benefit from improvements in speed: the same is true for large-scale data augmentation via rewriting. In this case, a practitioner will ideally explore various task-specific rewriting strategies that may lead to improvements as observed during development, and then incorporate the best strategy into a test-final model. Recently, sentential paraphrasing gained the ability to enforce lexical constraints (Hu et al., 2019), but constrained decoding was still too inefficient to be practical (Hokamp and Liu, 2017) at a large scale. Even with the approach described by Post and Vilar, exploring the space of possible rewriting strategies on a task-specific basis may be overly time consuming: our performance improvements to their algorithm lowers the barrier of entry, where one may more practically experiment with various strategies during development. To illustrate our point, we build an improved monolingual sentential rewriter that can be conditioned on arbitrary positive and negative lexical constraints and use this to augment data for three external NLP tasks with different strategies: Natural Language Inference (NLI), Question Answering (QA) and MT.

Our main contributions are:

- A more efficient and robust approach to lexically-constrained decoding with vectorized DBA and trie representations;
- A trained and freely available lexically-

Proceedings of NAACL-HLT 2019, pages 839–850
Minneapolis, Minnesota, June 2 - June 7, 2019.

constrained monolingual rewriter[1] with improvements in both human-judged semantic similarity and fluency over the initial PARABANK rewriter (Hu et al., 2019);

- Monolingual rewriting constraint heuristics for automatic data augmentation leading to improvements on NLI / QA / MT.

2 Background

Constrained decoding Prior work explored methods to apply lexical constraints to a Neural Machine Translation (NMT) decoder (Hokamp and Liu, 2017; Anderson et al., 2017). However, most of these methods are slow and impractical as they change beam sizes at different time steps, which breaks the optimized computation graph. Post and Vilar (2018) proposed a means of dynamically allocating the slots in a fixed-size beam to ensure that even progress was made in meeting an arbitrary number of constraints provided with the input sentence. However, despite it being their motivation, their approach did not scale to batching, instead they sequentially processed constraints for sentences within the batch.

Paraphrases and Rewriting Many works sought to create paraphrases or paraphrastic expressions through existing corpora. For example, DIRT (Lin and Pantel, 2001) extracts paraphrastic expressions from paths in dependency trees. Weisman et al. (2012) explored learning inference relations between verbs in broader scopes (document or corpus level). PPDB (Ganitkevitch et al., 2013) constructs paraphrase pairs by linking words or phrases that share the same translation in another language. PARANMT (Wieting and Gimpel, 2018) and PARABANK (Hu et al., 2019) used back-translation to build a large paraphrase collection from bilingual corpora.

For arbitrary sentence rewriting, Napoles et al. (2016) used statistical machine translation in tandem with PPDB as a black box monolingual sentential rewriter. Mallinson et al. (2017) used a series of NMT model pairs to perform back-translations for monolingual paraphrasing. A similar approach was adopted by PARANMT to create a large paraphrase collection, which is used to train a monolingual sentence rewriter for canonicalization. PARABANK (Hu et al., 2019) extends PARANMT's approach and produced a NMT-based rewriter with the ability to apply lexical constraints to produce multiple paraphrases.

However, Hu et al. (2019) did not: evaluate the rewriter's performance on in-the-wild sentences; explore more sophisticated versions of the rewriter; nor demonstrate its utility on NLP tasks.

Data augmentation Data augmentation has been used to improve performance and robustness in deep neural models. In NMT, the most common approach is back-translation, where monolingual text in the target language is translated to create synthetic source sentences (Sennrich et al., 2016). Variants of back-translation target specific words with high prediction loss (Fadaee and Monz, 2018), employed sampling to increase diversity (Edunov et al., 2018), replace rare words (Fadaee et al., 2017), or replace at random (Wang et al., 2018).

Automatic data generation has also been successfully used for community question answering (Chen and Bunescu, 2017), semantic parsing (Jia and Liang, 2016), and task-oriented dialogue (Hou et al., 2018) by generating new data from the training dataset. In contrast, our model is trained on a much larger external corpus and is fixed, independent of the task. Kobayashi (2018) utilized a pretrained language model for automated data augmentation, though they only consider word-level rewrites and encourage label-preservation, while we paraphrase whole sentences with lexical constraints, independent of a gold label.

Most similar to our experiments, Iyyer et al. (2018) explored *syntactic* paraphrasing for augmentation in sentiment and NLI tasks, extending prior work on PARANMT.

3 Improved Constrained Decoding

Lexically-constrained decoding is a modification to beam search that yields decoder outputs honoring user-supplied constraints. These constraints can be provided in the form of: *positive constraints*, which specify that certain tokens or token sequences must be present in the output; or *negative constraints*, which specify token or token sequences that must *not* be generated. Take positive constraints for example, in translating the sentence *Das stimmt einfach nicht* to English, the user can specify the constraint "not the case" to (presumably) get the output *That's just not the case* instead of model-preferred output *That's just not*

---

[1] `http://nlp.jhu.edu/parabank`

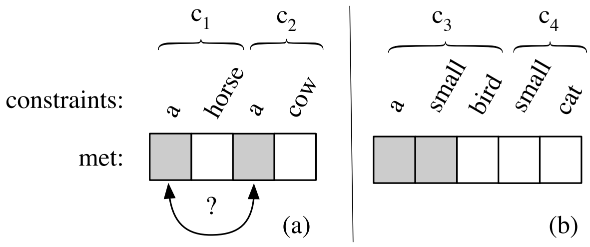


Figure 1: Two corner cases that arise when using an array implementation to track constraints. The gray boxes denote constraints that have been generated by the currently-tracked hypothesis. In (a), the decoder has to guess which constraint it is generating since they share a prefix. In (b), the decoder may start tracking constraint c_3 only to generate c_4 without realizing it.

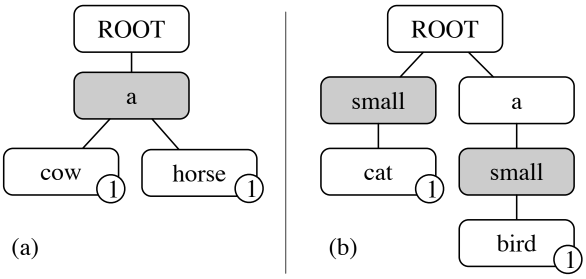


Figure 2: The trie representation solves both problems depicted in Figure 1. The constraint tracker no longer has to predict the constraint to track (a), and it can track multiple constraints by using multiple states (b). Counters are added to end nodes to denote how many times each constraint must be generated.

true. While there is no guarantee that the decoder will use the constraints in a sensible way, constraints are often well-placed empirically.

The implementation of positively constrained decoding comprises two key pieces: tracking which of the supplied constraints each hypothesis has already generated, and ensuring progress through the constraints by dynamically allocating the beam to hypotheses that have generated different numbers of them. We describe an improvement to each of these over Post and Vilar (2018), which includes: (1) a vectorized approach to beam allocation that works with batch decoding; and (2) the use of tries for recording constraint state, and thereby offsetting certain corner cases.

These contributions allow the decoder to find much better placement of constraints (as evidenced by an almost 2 point BLEU score increase) and to increase throughput for batch decoding.

Here, we assume the reader is familiar with beam decoding for NMT, the details of which are provided by Post and Vilar (2018).

3.1 Tracking constraints with tries

The implementation by Post and Vilar used a flat one-dimension array listing the word indexes of all positive constraints (duplicates allowed). A parallel array was used to mark which words in this list were non-final words in a sequence, so that progress could be tracked through sequences of tokens. Progress through the constraints was tracked by maintaining, for each slot in the beam, a third array, which marked which of the constraints had already been generated by that hypothesis.

However, this leads to corner cases when two constraints c_1 and c_2 share a subsequence. The first case occurs when two constraints have an identical prefix. Consider the constraints in Fig. 1(a) when translating the French sentence *une vache et un cheval*. The array-based implementation has to choose which constraint to generate when it has only generated the first word of the English translation, *a cow and a horse*. Suppose it chooses constraint c_1, *a horse*. If the subsequent step generates *cow* instead, the constraint tracking for the phrase *a horse* will be marked as incomplete and reset, and the decoder will not realize that it has satisfied a different constraint.

A second corner case arises when a constraint c_4 is a non-prefix substring of a constraint c_3. In this situation, the decoder may begin generating the longer constraint, only to generate the shorter one, without realizing it. For example, consider a target sentence that should be *a small cat saw a small bird*, with constraints *a small bird* and *small cat* (Figure 1b). When generating the first word, *a*, the decoder begins tracking c_3. It continues by adding to this hypothesis the second word, *small*. However, suppose it then extends this hypothesis with *cat*. It will abort tracking of c_3, and not realize that it completed c_4.

A more natural representation that addresses these corner cases is to organize constraints that haven't yet been generated into a trie. Nodes in the trie that represent the ends of constraints are augmented with a counter that indicates how many times that constraint must be generated.[2] Each time a constraint is completed, the number is decremented, and nodes of the trie can be trimmed when they lead only to paths ending in zero counts.

---

[2] Because one constraint can be a subsequence of another, some interior nodes will also have these counts.

sentno	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
unmet	0	0	1	1	1	1	2	2	2	3	4	4	5	5	5
step	**0**	**1**	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**0**	**0**	**1**	**0**	**1**	**2**
score	1.23	1.07	1.13	1.01	0.89	0.87	1.08	0.92	0.77	0.83	0.69	0.54	0.71	0.46	0.41
hypo #	4	2	1	0	6	3	2	0	1	5	1	0	1	0	0
vocab ID	4241	923	135	712	2122	14	805	4209	212	324	712	441	5243	853	5071

Figure 3: The hypothesis matrix during vectorized dynamic beam allocation (VDBA). Each column denotes a candidate hypothesis, each of which contains sufficient information to identify it. Hypotheses are sorted by sentence number, number of unmet constraints, and sequence scores. The "unmet" row indicates how many constraints each hypothesis has not yet generated. After sorting by sentence ID and "unmet", we assign monotonically increasing numbers across columns that share (sentno, unmet) values (the "step" row). We can now sort by the "step" row to ensure that the beam for a sentence contains hypotheses having met each number of constraints. We populate the next beam by taking the first k hypotheses (in gray) after sorting the matrix by **step**. Here, $k = 7$.

Finally, in order to address the second corner case, we track multiple states in each constraint trie.

In summary, we represent all the constraints as a compact trie. Each hypothesis in the decoder beam has its version of the trie. The set of active states in each hypothesis' trie tracks all suffixes of the target words that match against the constraint trie. When a constraint is generated, its counter is decremented and zero paths are pruned.

3.2 Negative constraints

Negative constraints are used to denote words and phrases that the decoder must not generate. Blocking single-word negative constraints can be done by setting their costs to infinity before doing top-k selection, at each time step. These negative constraints are also represented in a trie, although it is slightly different, because it does not have a counter and never needs to be pruned. Instead, it records at each node the list of word IDs that end phrases for which the current word is the penultimate. We similarly track all suffixes of the current hypothesis' target word string that match the negative constraint trie. At each time-step, we block the generation of active phrases by setting to infinity all word IDs marked in the current node (if any). This includes the root node, which handles single-word constraints. Each state is then extended by following outgoing arcs, if present, or else resetting them to the root state.

3.3 Vectorized Dynamic Beam Allocation

Post and Vilar (2018) describe an algorithm that divides the beam among hypotheses that have generated different numbers of positive constraints. For a beam of size k and with C positive constraint tokens, the algorithm produces a set of candidate extensions of the k hypotheses from the beam. They assemble these extensions from three sources: (a) the top-k best-scoring tokens across all hypotheses in the beam (without respect to constraints); (b) the set of tokens that advance the constraint trie for each hypothesis; and (c) the best-scoring extension of each hypothesis.[3] After constructing this candidate list, they whittle it down to a list of size k and use it to construct the beam at the next time step. This way, the algorithm ensures that portions of the beam are devoted to candidates having met different number of constraints, and thereby that progress is made towards meeting all the constraints as decoding proceeds.

However, their implementation used a procedural approach which is incompatible with batching; that is, constraints for input segments within a batch are processed sequentially, so increasing the batch size does not produce any speed gains. We replace the procedural approach with a vectorized one, which uses GPU operations to quickly assemble the list of candidates and allocate them to the beam such that we do benefit from batching.

A sketch of our algorithm follows. We assemble candidates from the same three sources described above. Sets (a) and (c) already use fast GPU operations. These operations can be done efficiently even batch-wise. Set (b) is less amenable to vectorization, but can be assembled by querying each hypothesis for its unmet constraints. We now use a sorting-based algorithm to parallelize the divvying

---

[3] The difference between (a) and (c) is that the items constituting (a) typically come from different extensions of the top hypotheses, whereas (c) ensures that one extension of each hypothesis is in the candidates list.

Constraints	Speed (sent/sec)				BLEU	
	batch size 1		batch size 20			
	Baseline	+MST,VDBA	Baseline	+MST,VDBA	Baseline	+MST,VDBA
none	4.78	4.84	**17.92**	17.51	40.9	40.9
phr4	0.75	0.89	0.93	**4.71**	47.8	**49.2**
rand3	0.78	0.80	0.87	**4.59**	43.0	**43.2**

Table 1: Comparison between the baseline implementation (SOCKEYE 1.18.57, commit 59180f3) and our approach with multi-state tries (MST) and vectorized Dynamic Beam Allocation (VDBA) in placing different constraints randomly extracted from the reference. All runs use a beam size of 10. The pruning threshold was set to 0 for no constraints ("none") and 30 for "phr4" and "rand3". Decoding speed is measured on an NVIDIA GTX 1080 Ti in sentences per second (the higher the better). Output quality is measured using SacreBLEU (the higher the better).

up of the beam to hypotheses having met different numbers of constraints.

We do this by assembling a matrix with all the candidates for all sentences in the batch (Fig. 3). This matrix contains a column for each candidate, including the sentence number, the number of unmet constraints for that hypothesis, its sequence score, the hypothesis it extends, and the next vocabulary ID. With this matrix, we can quickly select the k hypothesis extensions for the next timestep using a multi-key sort. The first key is the sentence number. Next, it is the number of unmet constraints in each hypothesis. We then make use of a "step" row, which assigns increasing indices *within* each group of hypotheses with the same number of unmet constraints. Sorting on this row as the third key establishes a round-robin assignment of the k-sized beam to items having met different numbers of constraints. In the end, we select the top k items (in the example, $k = 7$ and the selected columns are in gray).

3.4 Evaluation

We use SOCKEYE (Hieber et al., 2017)[4] for our evaluations. We trained a 6-layer German–English Transformer using the default settings on the WMT'18 training data and the `newstest2018` test set for evaluation (Bojar et al., 2018). Following Post and Vilar (2018), we compare decoding results in an unconstrained setting and with two sets of positive constraints: "rand3", which selects 3 random words from the reference, and "phr4", which selects a single 4-word phrase. We report decoding speed (in sentences per second) and BLEU score (Papineni et al., 2002), as measured by SacreBLEU (Post, 2018). The results are shown in Table 1.

Our approach is faster than existing approaches when decoding with positive constraints and produces the same or higher BLEU scores, which we take as a sign of more fluent and natural hypotheses under constraints. Without batching, there is no speedup, but at a batch size of 20, we see roughly a 5× speedup.

4 Improved Monolingual Rewriter

Inspired by the approach described in PARABANK (Hu et al., 2019), we trained a more powerful English monolingual rewriter by using a multi-head self-attention NMT model, Transformer (Vaswani et al., 2017). We used a 6-layer encoder and decoder with a model size of 512 and 8 attention heads. The encoder and decoder embeddings share the same weight. Unlike PARABANK, which trained a rewriter on a subset of 50M paraphrase pairs out of its collection, we trained on *all* of the paraphrastic pairs in PARABANK originated from CzEng[5] that: (1) have a regression score over 0.50; (2) only consist of ASCII characters after punctuation normalization; and (3) have a reference/paraphrase token Jaccard index between 0.25 and 0.65.

We retain 141,381,887 paraphrastic pairs, out of over 220 million, as training data after applying these filters. To ensure output quality, we only use back-translated paraphrases as source.

PARABANK is a real-cased resource. We mark all words that have first-character capitalization and convert them to lowercase. The marking is

---

[4] `https://github.com/awslabs/sockeye/`

[5] PARABANK generated paraphrases from two large bilingual corpora, CzEng (Bojar et al., 2016a) and GigaFrEn (Callison-Burch et al., 2009). We picked paraphrases from only CzEng, the larger one of the two.

used as a source factor (Sennrich and Haddow, 2016) to the encoder. This helps us to decrease the vocabulary size of the the training data.

We learn a shared byte-pair encoding (BPE) over the entire training data with 30,000 BPE operations (Sennrich et al., 2016), keeping all vocabulary items with a frequency over 50 in the post-BPE data. We follow Sennrich and Haddow (2016) and use "BIOE" tagging to annotate BPE segmentation and broadcast the casing factor accordingly. The encoder uses both source factors.

The model is trained on 2 NVIDIA GTX 1080Ti's until convergence (5 days).

Rewriter Evaluation We randomly sampled 100 instances from both MNLI matched and mismatched development set. Each instance consists of 4 sentences: premise, entailed, contradicting, and neutral. We use the following 3 different rewriters to rewrite all 800 sentences: (1) an LSTM-based rewriter trained on PARABANK alpha; following (Hu et al., 2019); (2) a Transformer-based rewriter trained on PARABANK alpha; and (3) a Transformer-based rewriter trained on full PARABANK with the filters and improvements described here.

Inspired by the interface of EASL (Sakaguchi and Van Durme, 2018), we ask crowd-workers to give each paraphrase a score between 0 and 100 depending its semantically similarity to the original, reference sentence. Independently, we provide options for flagging ungrammatical or nonsensical sentences. Paraphrases are judged by up to 3 different workers, with 11 workers participating. We randomly include an attention check consisting of reference sentence itself.[6]

The result is shown in Table 2. Switching the rewriter architecture from LSTM to Transformer improves the human-judged semantic similarity by 5.1% and fluency by 6.5%. The improvements described here leads to a gain of 9.6% in semantic similarity and 10.2% in fluency overall.

This improved Transformer-based rewriter is subsequently used for data augmentation.

	Similarity	STD	Fluency
LSTM alpha	74.5	25.0	80.7%
Transf. alpha	78.3	22.9	87.2%
Transf. Full	**81.7**	**20.9**	**90.9%**

Table 2: Comparison between three monolingual rewriting systems. Systems will "alpha" are trained on PARABANK alpha, which the other one is trained on the full data. Similarity is the mean human-judged semantic similarity score; the higher the better. STD described the standard deviation of similarity. Fluency is the percentage of paraphrases judged to be both grammatical and meaningful.

5 Paraphrastic Data Augmentation

We demonstrate the utility of our improved lexically-constrained decoding via data augmentation with some simple rewriting heuristics and two augmentation strategies. First, the model could be trained on the augmented (training) data. Orthogonally, predictions can be made on the all of the augmented (evaluation) data, which can then be *aggregated*. We show experimental results on natural language inference (NLI, Section 5.1), question answering (QA, Section 5.2), and NMT (Section 5.3) tasks.

These results are merely indicative of the potential in data augmentation via constrained paraphrasing, and are by no means a thorough investigation of strategies that yield the best improvements. Such an investigation, however, could be enabled by our algorithmic improvements and practitioners' domain expertise.

5.1 Natural Language Inference

Natural language inference is the task of determining entailment. Two sentences, a *premise* p and a *hypothesis* h, are labelled with ENTAILMENT, CONTRADICTION, or NEUTRAL depending on whether p logically entails, contradicts, or does not interact with h. MultiNLI (MNLI) (Williams et al., 2018) is a large, multi-genre dataset for natural language inference. The dataset is also divided into *matched* and *mismatched* portions based on whether the source of the evaluation data matches the source of the training set. Recent models rely on contextual sentence encoders pretrained on vast amounts of English monolingual text (Peters et al., 2018; Devlin et al., 2018).

We train and evaluate a model on MNLI, and find that data augmentation leads to improvements exceeding and complementary to those by ELMo, possibly due to improved lexical diversity during training and at inference.

---

[6] Only workers who pass the test at least 90% of the time and contribute at least 9 judgments are included in the result.

Model We use the model described in Bowman et al. (2019)[7] with the default parameters. They train a sentence representation model (possibly on top of ELMo) on the MNLI training set and subsequently train a clean task-specific model for each task (for this model, MNLI again). The task-specific MNLI model roughly follows BiDAF (Seo et al., 2016), followed by an MLP.

We also train a model without the ELMo contextual layers to compare contextual sentence representations against data augmentation. Since there is minor variance between different random seeds,[8] we train each model twice and evaluate the best-performing model on the development set.

Paraphrase Generation We generate paraphrases for our data augmentation experiments by negatively constraining on the most content-bearing token of each input sequence, as determined by inverse document frequency (IDF). For a given input sequence s we calculate the IDF of each token $t_i \in \mathrm{s}$ as $\log \frac{|D|}{|d \in D : t_i \in d|}$ where D is the set of all English sentences in the train set. This relatively simple lexical constraint tends to force the decoder to rewrite the input sequence using different (but semantically related) words while maintaining fluency. In practice, we observed an average unigram precision of 67.6%; i.e., 32.4% of tokens in paraphrases were not contained in their corresponding inputs.

Additional results using a positively constrained rewriter in Appendix A.

Data Setup We first rewrite all premises P to P' and all hypotheses H to H', then for each $p_i \in P, h_i \in H$, we include all four examples, (p_i, h_i), (p_i, h'_i), (p'_i, h_i), and (p'_i, h'_i) into the training set, always preserving the original corresponding gold label. We include two copies of the original dataset in training to increase its weight. The original MNLI dataset contains 393K training pairs, and 20K in each dev and test, while the augmented dataset consists of 1.96M training pairs, and 79K in dev and test.

At test time, we rewrite the test sentence pairs. A trained model can also make predictions on each of three rewritten sentence pairs. Together with the original prediction, these four can then be aggregated by assigning weights to each prediction source. In our experiments, we perform this weighted aggregation (+Agg) for each model, tuning on the development set (Appendix B).

	Dev.	Test. (m/mm)
Baseline	74.8	74.7 (74.8/74.6)
+Agg.	75.0	74.9 (74.9/74.8)
+Train	75.4	75.2 (75.0/75.3)
+Train+Agg.	75.6	75.4 (75.1/75.7)
+ELMo	75.8	75.0 (75.1/75.0)
+Agg.	75.9	75.2 (75.3/75.1)
+Train	76.4	75.6 (75.6/75.6)
+Train+Agg.	**76.7**	**75.8 (75.9/75.7)**

Table 3: F1 scores on MNLI. +Train denotes training on augmented data; +Agg. denotes using a weighted aggregation. Scores on the development set are a weighted average between the matched (m) and mismatched (mm) portions of the dataset, while the test set scores are additionally broken down into each category.

Experimental Results We find in Table 3 that data augmentation helps during training and inference. Not only are the total gains from augmentation comparable to those from ELMo, they are apparent even in the presence of the contextual sentence encoder. This suggests that the gains from data augmentation through rewrites complement recent gains from contextual sentence encoders.

The fairest external comparison is with Bowman et al. (2019), as our model is identical. Their best models achieve 76.2 F1 on development and 75.4 F1 on test. On the development set, they see a gain of 0.6 points by using multi-task training and external datasets. On that set, we report a total gain of up to 0.9 points purely through data augmentation. With respect to absolute test set scores, our best model outperforms theirs by 0.4, showing that rewriter-based data augmentation can be a powerful method for NLP tasks.

Analysis NLI systems have been shown to be brittle when the input is perturbed (Alzantot et al., 2018). Even when the premise or hypothesis is changed in a way that preserves the entailment semantics, the NLI system may make an incorrect prediction where it was previously correct. We present evidence showing that data augmentation for NLI reduces the brittleness of our model.

To demonstrate the brittleness of the baseline models, we analyze how predictions change. The model trained on the *original un-augmented*

---

[7] `https://github.com/jsalt18-sentence-repl/jiant`

[8] Bowman et al. (2019) found the variance to be 0.2

	P', H	P, H'	P', H'	+Agg.
No change	88.51	84.23	81.95	96.20
$- \rightarrow +$	4.23	5.33	6.08	1.75
$+ \rightarrow -$	5.84	8.44	9.67	1.49
$-_1 \rightarrow -_2$	1.42	2.00	2.30	0.57
No change	88.20	83.26	80.68	96.00
$- \rightarrow +$	4.03	5.45	6.11	1.80
$+ \rightarrow -$	6.42	9.22	10.78	1.72
$-_1 \rightarrow -_2$	1.35	2.08	2.42	0.47

Table 4: Percentage of changed predictions on the MNLI development set using the baseline model without (top) and with (bottom) ELMo. $- \rightarrow +$ (correct after rewrite), $+ \rightarrow -$ (originally correct), and $-_1 \rightarrow -_2$ (different incorrect) are changes after rewriting. +Agg. denotes the predictions after weighted aggregation.

P: Visit at sundown or out of season to get the full flavor of the setting
H: The setting is better to visit at sundown or during low season
H': It is better to visit at sunset or during low season

Gold: Entailed
P, H: Predict Entailed
P, H': Predict Neutral

P: I had rejected it as absurd , nevertheless it persisted
H: It persisted even after I rejected it as an absurdity
H': It went on even after I turned it down as an absurdity

Gold: Entailed
P, H: Predict Entailed
P, H': Predict Contradiction

Figure 4: Cases where the baseline system changes its prediction on rewritten examples.

dataset is evaluated on the original development set and each of the rewritten development sets, and we investigate the differences. Table 4 shows how often original predictions are different from the corresponding predictions on the rewritten development sets; predictions can be (1) unchanged, (2) newly correct, (3) newly incorrect, or (4) changed but still incorrect, while Figure 4 shows how even relatively modest, semantically valid paraphrases can cause the NLI model change incorrectly.

Given a perfect rewriter that always generates semantically equivalent paraphrases and a perfect NLI model robust to perturbations, we would expect no change in predictions between the original development set and the rewritten ones. However, this is not what we observe; Table 4 shows that rewriting leads to a greater percentage of newly incorrect predictions than newly correct predictions.

We believe that the higher percentage of newly incorrect predictions on the rewritten development sets demonstrates the brittleness of the NLI system rather than semantic dissimilarity that may be introduced by the rewriter. We note that the aggregated predictions shows the opposite pattern: we see a higher percentage of newly *correct* predictions than incorrect ones. If the paraphrases were largely semantically dissimilar we would not expect any gain by combining predictions.

Given both the numerical boost seen by aggregation and the above examples, we hypothesize that the rewriter does not frequently change entailment semantics. Because the semantics remain similar, and because the paraphrases were generated with constraints designed to introduce lexical diversity, we believe that the label-preserving data augmentation improves the NLI model by making it more tolerant of minor lexical differences, better able to generalize, and less inclined to memorize.

5.2 Question Answering

We apply our paraphrastic rewriter to the task of question answer sentence selection to see if augmenting with paraphrases leads to improvements. The task is defined as follows: Given a question q and a set of candidate sentences $\{c_i\}$, select the candidates which answer q.

Model We adapt a popular neural architecture for NLI, InferSent (Conneau et al., 2017), to our QA sentence selection task. In InferSent, the questions and answers (originally the premises and hypotheses) are embedded using an uncontextualized word embedding (e.g. GloVe), which we also experiment with ELMo (Peters et al., 2018) to incorporate recent advancements in large-scale contextualized pre-training. Bidirectional LSTMs (Graves and Schmidhuber, 2005) are run atop of these contextualized embeddings and a max-pooling layer is used to generate a feature vector for both the question and the answer. Following various matching methods (Mou et al., 2016) and a multi-layer feed-forward neural network, the model produces a final score.

We train the system following the method proposed by Rao et al. (2016), utilizing a ranking loss (Weston and Watkins, 1999) that contrasts positive answers against negative ones.

Paraphrase Generation We augment each answer candidate sentence with exactly 1 paraphrase in the dataset using the following heuristics: (1) named entities shared between a specific answer and its corresponding question are retained as positive constraints; (2) correct answer spans are retained as positive constraints; (3) words with the top-k IDFs (inverse document frequencies; hence "important" words) that are not positive constraints are selected as negative constraints to promote the lexical diversity of the paraphrases.[9]

Data Setup We augment the raw TREC-QA dataset (Wang et al., 2007) under the following orthogonal strategies: (1) augmenting the training set with the paraphrases generated via the approach described above; (2) augmenting the answer candidates at evaluation time, and choosing the max score among the paraphrases as the score (aggregation by *voting*).

Experimental Results We evaluate our models using average precision (MAP) and mean reciprocal rank (MRR). Model selection is done with early stopping to choose the epoch with the maximum MAP score. Note that the "Baseline (+ELMo)" settings below falls back to the standard QA selection task, and our score under ELMo is comparable to earlier state-of-the-art results, e.g. by Rao et al. (2016).

	MAP	MRR
Baseline	71.42	75.16
+Voting	72.94	77.17
+Train	71.57	74.63
+Train +Voting	73.96	80.77
+ELMo	77.49	81.86
+Voting	**80.61**	**85.65**
+Train	75.58	80.30
+Train +Voting	77.86	84.34

Table 5: Experimental results on QA selection.

It is shown that augmenting at evaluation time (aggregation by voting) result in stable improvement (around +2~3% MAP and +2~6% MRR for both scenarios that either augments the training data or not)—this shows that increasing the paraphrastic diversity of the answer candidates could potentially make the system more robust. However augmenting the training set does not yield such improvements—we speculate that this may introduce some noise to the training data.

5.3 Machine Translation

We apply our paraphrastic rewriter to the WMT 2016 Turkish-English translation task (Bojar et al., 2016b). We see no improvement in English to Turkish translation, but see a 1.1 BLEU improvement when training an initial NMT system on half paraphrased and half original data, and continued training on the original data. Full details of the experiments are in Appendix C. This was the highest concentration of standard data we experimented with, and future work will explore additional ways of data augmentation using paraphrases.

6 Conclusion

Lexically-constrained sequence decoding provides control over whether certain tokens or token sequences appear in the output. Motivated by applications such as large-scale MT, we improved the speed for constrained decoding significantly by proposing a vectorized dynamic beam allocation algorithm. We also added multi-state trie representations for robustness to corner cases.

Also reliant on the efficiency of constrained decoding is data augmentation via rewriting, where one might need to explore a variety of strategies with task-specific constraints on development data. We trained an improved monolingual sentential rewriter and used it to rewrite data for NLP tasks. We experimented with augmenting training data, aggregating predictions on rewritten test data, and both. Using a few simple constraint heuristics, we showed improvements additive to ELMo in NLI and QA, and in MT. The rewriter, along with the augmented data files, can be found at `http://nlp.jhu.edu/parabank`. We hope this will enable future exploration of augmentation strategies for a variety of NLP tasks.

Acknowledgments

Thanks to Michael Denkowski, who first suggested using a trie to represent constraints in a group discussion. This research was supported in part by DARPA AIDA and DARPA LORELEI.

---

[9] Stopwords and tokens with non-letter characters (e.g. *with*, *42*, *n't*) are excluded. $k \in \{2, 3, 4\}$ is a hyperparameter we tune – we found out that generally $k = 2$ works the best.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of EMNLP 2017*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016a. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Computer Science, pages 231–238, Cham / Heidelberg / New York / Dordrecht / London. Masaryk University, Springer International Publishing.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016b. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Samuel R. Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, and Berlin Chen. 2019. Looking for ELMo's friends: Sentence-level pretraining beyond language modeling.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Charles Chen and Razvan Bunescu. 2017. An exploration of data augmentation and rnn architectures for question ranking in community question answering. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 442–447, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of EMNLP 2018*, pages 489–500. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Marzieh Fadaee and Christof Monz. 2018. Back-translation sampling by targeting difficult words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 436–446. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings NAACL-HLT 2013*, pages 758–764.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of ACL*, pages 1535–1546, Vancouver, Canada.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. PARABANK: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. *AAAI*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA. ACM.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain. Association for Computational Linguistics.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proc. ACL*, page 130.

Courtney Napoles, Chris Callison-Burch, and Matt Post. 2016. Sentential paraphrasing as black-box machine translation. In *Proceedings of the NAACL 2016*, pages 62–66, San Diego, California. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of NAACL 2018*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Keisuke Sakaguchi and Benjamin Van Durme. 2018. Efficient online scalar annotation with bounded support. In *Proceedings of ACL*, pages 208–218, Melbourne, Australia. ACL.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861. Association for Computational Linguistics.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 194–204, Jeju Island, Korea. Association for Computational Linguistics.

Jason Weston and Chris Watkins. 1999. Support vector machines for multi-class pattern recognition. In *ESANN 1999, 7th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 21-23, 1999, Proceedings*, pages 219–224.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of ACL*, pages 451–462. ACL.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Courteously Yours: Inducing courteous behavior in Customer Care responses using Reinforced Pointer Generator Network

Hitesh Golchha*, Mauajama Firdaus*, Asif Ekbal, Pushpak Bhattacharyya

Department of Computer Science and Engineering
Indian Institute of Technology Patna
Patna, India
(hitesh,mauajama.pcs16,asif,pb)@iitp.ac.in

Abstract

In this paper, we propose an effective deep learning framework for inducing courteous behavior in customer care responses. The interaction between a customer and the customer care representative contributes substantially to the overall customer experience. Thus, it is imperative for customer care agents and chatbots engaging with humans to be personal, cordial and emphatic to ensure customer satisfaction and retention. Our system aims at automatically transforming neutral customer care responses into courteous replies. Along with stylistic transfer (of courtesy), our system ensures that responses are coherent with the conversation history, and generates courteous expressions consistent with the emotional state of the customer. Our technique is based on a reinforced pointer-generator model for the sequence to sequence task. The model is also conditioned on a hierarchically encoded and emotionally aware conversational context. We use real interactions on Twitter between customer care professionals and aggrieved customers to create a large conversational dataset having both forms of agent responses: generic and courteous. We perform quantitative and qualitative analyses on established and task-specific metrics, both automatic and human evaluation based. Our evaluation shows that the proposed models can generate emotionally-appropriate courteous expressions while preserving the content. Experimental results also prove that our proposed approach performs better than the baseline models.

1 Introduction

With the advancement of artificial intelligence (AI) and natural language processing (NLP), automatic systems have made immense impact on human lives by assisting them in their everyday works. Human-computer interaction is pervasive in many applications such as chatbots, personal assistants and many more. Natural language generation (NLG) component of such systems is an important aspect of every human computer interaction. Thus research in recent years have been on modulating biases, styles and control in text generation to enhance these interactions.

Customer care is an essential tool used by companies to provide guidance, assistance and in building stable customer relations. The ease of access, ease in following-up and immediacy of social media has made it a strong platform for companies and applications to interact with their customers. In this platform, we see the usage of courteous and emphatic language, which is the center of our current study. For the growth of any company or application it is necessary for the customer care agents to be cordial and amicable to the customer. Thus along with handling queries, it is important for agents to provide customer satisfaction by greeting, empathizing, appreciating feedback, apologizing at the right time, and thus build a strong relation with the customer. In Table 1, we showcase different situations in which an agent can behave courteously, thereby providing a good customer experience.

In this work, we focus on proposing an effective deep learning framework to enhance the existing NLG systems by converting their replies to courteous ones, by staying conversationally grounded, and emotionally aware of the user. For any Natural Language Generation (NLG) module (generic or task oriented), courteous response can play an important role in keeping the user engaged with the system. Also, it will make the system more human-like while generating responses. Inducing courteous behavior in responses can be fused with any existing NLG system to give them humanly essence and simultaneously make users

---

[*] First two authors are jointly the first authors

Proceedings of NAACL-HLT 2019, pages 851–860
Minneapolis, Minnesota, June 2 - June 7, 2019.

Generic	Courteous	Behaviour
How can we help?	*Help has arrived! We are sorry to see that you are having trouble, how can we help?*	Apology
Can you send us a screenshot of what you're seeing?	*Hey Craig, help's here! Can you send us a screenshot of what you're seeing?*	Greet
Let's discuss it in DM.	*We want to help. Let's discuss it in DM.*	Assurance
What is happening with your internet?	*Oh no that's not good. I can help! What is happening with your internet?*	Empathy
Enjoy your show while flying!	*Thanks for your kind words and enjoy your show while flying!*	Appreciation

Table 1: Examples of Courteous Responses

more comfortable in using these systems leading to an increase in user association with the brand or product. This would eventually lead to customer satisfaction with an increase in customer retention. Moreover, such language conditioning shall ensure that responses are more human-like. Thus, the major motivation behind this task is to create systems that are able to converse with humans efficiently and generate replies in accordance with the emotions of the customer. Courteousness is a virtue of humans and to be able to make a machine behave courteously is a challenging task.

Unlike a generic NLG system that focuses in generating responses, our system adds courteous nature and emotional sense to the replies, thereby, making the responses interesting and engaging to the users. Such systems have high applications in many areas/companies that employ chatbots to deal with the customers. We thus propose a novel research direction of inducing courteous behavior in the natural language responses for the customer care domain whilst being contextually consistent. The key contributions of our work are summarized as follows:

(i) Creation of a high quality and a large conversational dataset, Courteously Yours Customer Care Dataset (CYCCD) prepared from the actual conversations on Twitter. We provide both forms of agent responses: generic and courteous.

(ii) Proposal of a strong benchmark model based on a context and emotionally aware reinforced pointer-generator approach which demonstrates very strong performance (both on quantitative and qualitative analyses) on established and task-specific metrics, both automatic and human evaluation based.

The rest of the paper is structured as follows: In section 2, we discuss the related works. In Section 3 we explain the proposed methodology followed by the dataset description in section 4. Experimental details, evaluation metrics and results are presented in section 5 and 6 respectively. In section 7, we present the concluding remarks followed by future directions.

2 Related Work

Natural language generation (NLG) module has been gaining importance in wide applications such as dialogue systems (Vinyals and Le, 2015; Shen et al., 2018; Wu et al., 2018; Serban et al., 2017a; Raghu et al., 2018; Zhang et al., 2018; Li et al., 2016), question answering systems (Reddy et al., 2017; Duan et al., 2017), and many other natural language interfaces. To help the users achieve their desired goals, response generation provides the medium through which a conversational agent is able to communicate with its user. In (Serban et al., 2017b), the authors have proposed an hierarchical encoder-decoder model for capturing the dependencies in the utterances of a dialogue. Conditional auto-encoders have been employed in (Zhao et al., 2017), that generates diverse replies by capturing discourse-level information in the encoder. Our work differentiates from these previous works in dialogue generation in a way that we embellish the appropriate response content with courteous phrases and sentences, according to the conversation. Hence, our system is an accompaniment to any standalone natural language generation system to enhance its acceptability, usefulness and user-friendliness.

Emotion classification and analysis (Herzig et al., 2016) in customer support dialogue is important for better understanding of the customer and to provide better customer support. Lately, a number of works have been done on controlled text generation (Hu et al., 2017; Li et al., 2017; Subramanian et al., 2017; Fedus et al., 2018; Peng et al., 2018) in order to generate responses with desired attributes. Emotion aware text generation (Zhou and Wang, 2018; Zhou et al., 2018; Huang et al., 2018) have gained popularity as it generates responses depending on a specific emotion. Previous works in conditioned text generation have worked on inducing specific biases and behaviors (Herzig et al., 2017) while generation (like emotion, style, and personality trait). Our work is different in the sense that it can encompass different emotional states (like joy, excitement, sadness, disappointment) and traits (like friendliness, apologetic, thankfulness, empathy), as is the demand of the situation.

Style transfer has been an emerging field in natural language processing (NLP). A couple of works have been done in changing the style of an input text and designing the output text according to some particular styles. In (Rao and Tetreault, 2018), a dataset has been introduced for formality style transfer. Unsupervised text style transfer has encouraged in transforming a given text without parallel data (Shen et al., 2017; Carlson et al., 2017; Fu et al., 2018; Li et al., 2018; Niu and Bansal, 2018). Overall our system is novel as it is motivated by the need for inducing specific behavior and style in an existing NLG systems (neural, or template-based) as a means of post editing, by simultaneously being emotionally and contextually consistent. We have successfully demonstrated this behavior through empirical analysis for a specific application of customer care.

3 Methodology

Given the Conversation History (previous few exchanges in the dialog), and the Generic Response, the task is to generate the Courteous Response. The architectural diagram of our proposed model is in Figure 1.

3.1 Conversational History Representation

The conversation history C is a sequence of utterances $(u_1, u_2, \ldots, u_D)$ and each utterance u_d is a sequence of words $w_1, w_2, \ldots, w_N$ which are represented by their embeddings. For encoding the emotional states associated with these utterances, we use the output distribution from DeepMoji (Felbo et al., 2017) which is pre-trained on the emoji prediction task.

Let the utterance u_d be a sequence of sentences $s_1, s_2, \ldots, s_N$, where the n^{th} sentence has an emotional embedding $e_{n,d}$. Then the emotional representation of the utterance is:

$$e_d[i] = \max_n e_{n,d}[i] \tag{1}$$

The first bi-directional layer over any utterance u_d yields the hidden states $h^1_{1d}, h^1_{2d}, \ldots, h^1_{Nd}$, where N is the word length of the utterance. The final representation of any utterance r_d is given by the concatenation of the emotional representation as well as the last hidden state of the Bi-directional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) encoder.

$$r_d = [e_d \cdot h^1_{Nd}] \tag{2}$$

The second hierarchical layer Bi-LSTM encodes the utterance representations $r_1, r_2, \ldots, r_D$ as hidden states $h^2_1, h^2_2, \ldots, h^2_D$. The last hidden state h^2_D is the representative of the conversational history, and is renamed as the conversational context vector c.

3.2 Encoder states

Another single layer unidirectional LSTM network encodes the generic response word embedding sequence to obtain the encoder hidden states h_i.

3.3 Decoder states and Attention calculation

At the decoder time step t, the decoder LSTM state s_t is used to calculate the attention distribution over the encoder states α^t:

$$e^t_i = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \tag{3}$$
$$\alpha^t = \text{softmax}(e^t) \tag{4}$$

where v, W_h, W_s and b_{attn} are trainable parameters.

This attention distribution helps to identify the relevant encoder states necessary for the current decoding step. The representation of the encoder for this time step is an attention weighted sum of its states, called the context vector:

$$h^*_t = \sum\nolimits_i \alpha^t_i h_i \tag{5}$$

The LSTM state s_t is updated using s_{t-1}, the previous time step's context vector h^*_{t-1}, word embedding of the previously generated word $w_{emb}(y_{t-1})$, and the conversation context vector c.

$$s_t = LSTM(s_{t-1}, W_p[w_{emb}(y_{t-1}), h^*_{t-1}, c] + \tilde{b}) \tag{6}$$

where, W_p and $\tilde{b}$ are the trainable parameters.

3.4 Output distribution calculation

To aid the copying of words from the generic response while generating the courteous response, we use the mechanism similar to (See et al., 2017). For the pointer generator network, the model computes two distributions, one over the

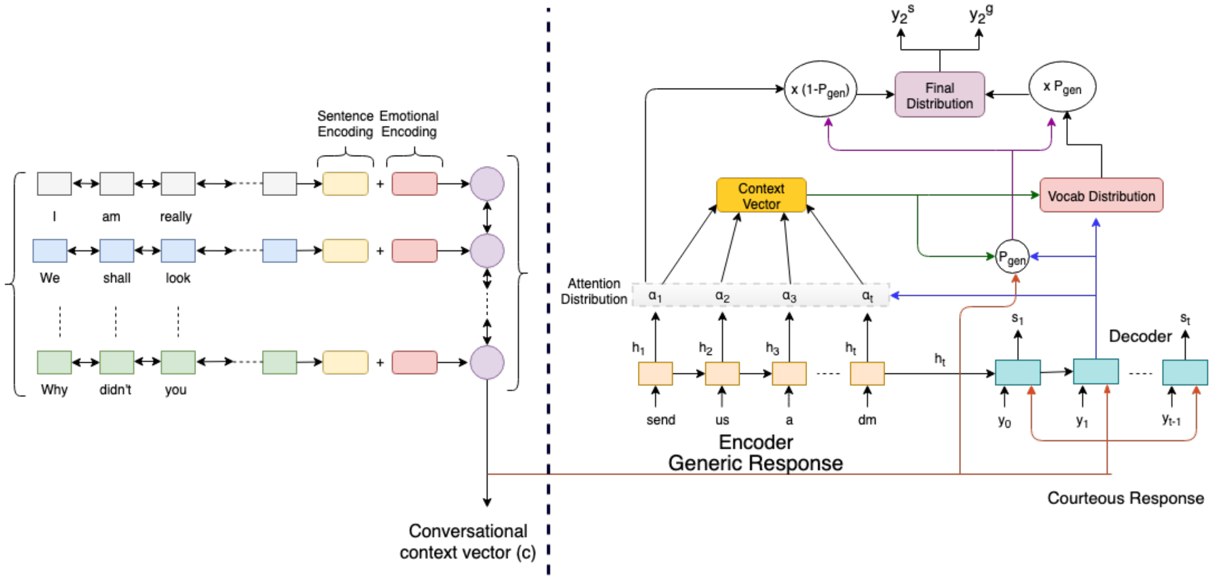


Figure 1: Architectural Diagram of the Proposed Model. Inputs to the model: Conversation History (left), Generic Response (centre) Output: Courteous Response (right). The Conversation History is encoded by hierarchical BiLSTM to a Conversational Context vector c. The encoder encodes the Generic Response into hidden states h_i. Response tokens are decoded one at a time. Attention α_i, and vocabulary distributions (p_{vocab}) are computed, and combined using p_{gen} to produce output distribution. Sampling it yields y_i^s and taking its argmax yields y_i^g.

encoder words (α^t) and one over the vocabulary (p_{vocab}).

$$p_{vocab} = \text{softmax}(W'(W[s_t, h_t^*] + b) + b') \quad (7)$$

where W, W', b and b' are the trainable parameters.

The trade-off between the two distributions is computed dynamically in the form of the *generation probability* $p_{gen} \in [0, 1]$ from the context vector h_t^*, the decoder state s_t, the decoder input x_t, and conversational context vector c:

$$p_{gen} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + w_c^T c + b_{\text{gen}}) \quad (8)$$

where vectors w_{h^*}, w_s, w_x, w_c and scalar b_{gen} are trainable parameters and σ is the sigmoid function.

The final distribution over the union of the vocabulary words and the words of the generic response is calculated by:

$$P(w) = p_{gen} p_{vocab}(w) + (1 - p_{gen}) \sum\nolimits_{i:w_i=w} \alpha_i^t \quad (9)$$

3.5 Model training

We use the joint reinforcement learning (RL) and machine learning (ML) training as used in (Paulus et al., 2017). If $\tilde{y} = \{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{n'}\}$ is the gold output tokens for given generic response tokens x_1 and conversation history x_2, the maximum-likelihood objective using teacher forcing is given by:

$$L_{ML} = -\sum_{t=1}^{n'} \log p(\tilde{y}_t | \tilde{y}_1, \ldots, \tilde{y}_{t-1}, x_1, x_2) \quad (10)$$

Along with training with the maximum likelihood error, we also use reinforcement learning to learn from maximizing discrete metrics that are task specific (which we design as the rewards). We use the self-critical policy gradient algorithm proposed in (Rennie et al., 2017).

Here the REINFORCE (Williams, 1992) algorithm is baselined with the reward obtained by the inference time algorithm (which performs greedy decoding), without the need for training a critic network for estimating value functions. During training, two output sequences are produced: y^s, obtained by sampling $p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x)$ probability distribution, and y^g, the baseline output, obtained by greedily maximizing the output probability distribution at each time step.

$$L_{RL} = (r(y^g) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x_1, x_2) \quad (11)$$

Our reward function $r(y)$, used for evaluating y against the gold standard output is

$$r(y, \tilde{y}) = \lambda_1 \cdot m1(y, \tilde{y}) + \lambda_2 \cdot m2(y, \tilde{y}) \quad (12)$$

It is the weighted mean of the two terms:

(i) BLEU metric $m1$: Ensures the content matching between the reference and the decoded outputs.

(ii) Emotional accuracy $m2$: Measured by the cosine similarity of the emoji distributions of the gold and generated responses (using pretrained DeepMoji). It ensures that the emotional states of the generated courteous behavior is consistent with the gold.

We first pre-train using the maximum likelihood (ML) objective (Eq. 10) and then using a mixed objective function with a reduced learning rate:

$$L_{mixed} = \eta L_{RL} + (1 - \eta) L_{ML}, \quad (13)$$

3.6 Baselines

We develop the following models:

1. **Model-1**: This is a Seq2Seq model with attention (Luong et al., 2015) and decoder conditioned on the conversational context vector c (without concatenating emotional embedding i.e. instead of Eq. 2, $r_d = h^1_{Nd}$)
2. **Model-2**: This model is developed using Model-1 along with the copying mechanism of Pointer Generator Network.
3. **Model-3**: This model is developed using Model-2 along with emotional embeddings in the conversational context vector as in E.g., 2.

	Train	***Valid***	***Test***
# Conversation	140203	20032	40065
# Utterances	179034	25642	51238

Table 2: Dataset Statistics

4 Dataset

In this section we describe the details of the dataset that we create for our experiments.

4.1 Dataset source and description

We use the data of the interactions between customers and professional customer care agents of companies on their Twitter handles. We source the requisite Twitter data from the dataset made available on Kaggle by Thought vector [1]. Tweets have labels of company names, anonymized user ids, time stamps, and response tweet ids - essential for reconstructing conversations, and nuanced analyses. We filter out conversations having multiple responses to a single tweet, and those starting by a tweet by a company. This was done to ensure correct conversation flow and to acquire suggestion / complaint based exchanges, respectively.

4.2 Process for data creation

As there exists no dataset with generic and courteous versions of utterances we create our own dataset. We prepare responses of generic styles by filtering out courteous sentences, phrases and expressions from the actual responses. We presume actual responses as the courteous form of response.

An example conversation:

Customer utterance (conversation history): *y'all just came to my house like last week and I'm having problems with my internet again smh*

Tweet by the Customer Care professional: *Oh no that's not good. I can help! What is happening with your internet?*

We use this conversation to prepare the courteous and the generic response

1. **Courteous response**: *Oh no that's not good. I can help! What is happening with your internet?*
2. **Generic response**: *What is happening with your internet?*

As we want to filter out courteous phrases / sentences from a given customer care tweet, we segment the tweet into sentences. Purely courteous (and non-informative) sentences must be removed, purely informative sentences must be retained, and informative sentences with courteous expressions must be transformed (to remove only the courteous part from the sentence). We define these three forms of sentences as:

(i) *Courteous sentences*: Sentences which do not contain any information/ suggestions, and are purely non-informative. These may include personalized greetings and expression of appreciation, apology, empathy, assurance, or enthusiasm. Example: *Sorry to hear about the trouble!*

(ii) *Informative sentences without courteous expressions*: These sentences contain the actual content of the tweet and are generally assertions, instructions, imperatives or suggestions. Example: *Simply visit url_name to see availability in that area!*

---

[1] https://www.kaggle.com/thoughtvector/customer-support-on-twitter

(iii) *Hybrid-Informative sentences with courteous expressions*: These are the sentences of the second type also containing some expressions of the first type. Example: ***We appreciate the feedback,*** *we'll pass this along to the appropriate team.*

4.3 Scaling up for large data creation

We annotate sentences in isolation by grouping similar sentences together to speed up annotations and then reconstruct the generic sentences by post-processing rules. We follow the following procedure to prepare the dataset for each company separately:

1. *Sentence segmentation*: We first extract the tweets from customer care agents. Each tweet is segmented into sentences to eventually identify three forms of the sentences.

2. *Clustering*: As expressions and sentences used by professionals of a company often follow similar patterns. Grouping similar sentences together before annotation would therefore significantly make the process faster. The vector-semantic representations of sentences are obtained using the sentence encoder(Conneau et al., 2017) trained on the SNLI corpus(Bowman et al., 2015). We use the K-Means clustering(Aggarwal and Zhai, 2012)(k = 300) to cluster these sentences.

3. *Annotations*: Three annotators proficient in the English language were assigned to annotate the sentences into the three categories: *purely courteous, purely informative, hybrid*. For sentences having both informative and courteous clauses/expressions (hybrid), they were asked to manually prepare the generic sentence by removing the courteous part. Also they were asked to identify non English conversations (and filter them). We observe the multi-rater Kappa agreement ratio of approximately 80%, which may be considered as reliable.

4. *Preparing generic responses*: Now let us assume we have a courteous response S with n sentences $s_1, s_2, \ldots, s_n$. We obtain the generic response by removing the courteous sentences, retaining the informative sentences, and replacing the hybrid sentences with the prepared generic equivalents.

We divide the conversation into train, validation and test sets as given in Table 2. Each training example is of the form: conversational history (last three utterances), generic response and courteous response.

5 Experiments

Implementation Details: We use a vocabulary of size 30k for the task (as the range of courteous expressions is limited, and informative contents can be copied even if they are out-of-vocabulary-OOV). We use 256 dimensional hidden states and 128 dimensional word embeddings (not pre-trained). We use AdaGrad as the optimizer with gradient clipping (magnitude 2). We train with batches of size 16, and use the same size for beam search decoding. We monitor smoothened running loss on the validation set for early stopping and finding the best models for decoding. We use η = 0.99 (similar to (Paulus et al., 2017)) for the joint loss. For the reward function the values of λ_1 and λ_2 are 0.75 and 0.25, respectively.

Automatic Evaluation: For automatic evaluation, in addition to the standard metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and perplexity, we also use two task-specific metrics:

1. Content preservation (CP): We want to measure how much of the informative content from the original generic response(X) is reflected in the generated courteous response(Y). We use a measure similar to ROUGE-L recall.

$$CP = LCS(X, Y)/len(X) \tag{14}$$

where LCS is the longest common subsequence.

2. Emotional accuracy (EA): To measure the consonance between the generated courteous expressions (source of emotion) and the gold, we find the cosine similarity between the MojiTalk emoji distributions of the two responses (X_e and Y_e).

$$EA = X_e \cdot Y_e/(|X_e||Y_e|) \tag{15}$$

Human Evaluation: In order to understand the quality of the responses we adopt human evaluation to compare the performance of different models. We randomly sample 500 responses from the test set for human evaluation. Given a generic response along with conversation history, three human annotators with post-graduate exposure were assigned to evaluate the courteous responses generated by the different models for the three metrics:

1. Fluency (F): The courteous response is grammatically correct and is free of any errors.

2. Content Adequacy (CA): The generated response contains the information present in the generic form of the response and there is no loss of information while adding the courteous part to the responses.

3. Courtesy Appropriateness (CoA): The courtesy part added to the generic responses is in accordance to the conversation history.

The scoring scheme for fluency and content adequacy is 0: incorrect or incomplete, 1: moderately correct, 2: correct, whereas for courtesy appropriateness the scoring scheme is -1: inappropriate, 0: non-courteous, 1: appropriate, respectively. We computed the Fleiss' kappa (Fleiss, 1971) for the above metrics to measure inter-rater consistency. The kappa score for fluency is 0.75 and courtesy appropriateness is 0.77 indicating "substantial agreement" and the score is 0.67 for content adequacy denoting "considerable agreement".

6 Results and Analysis

Automatic evaluation results: Results of the different models are presented in Table 3. The proposed model performs significantly better than the other baselines for all the evaluation metrics and the improvement in each model is statistically significant compared to the other models.[2]. The attention based sequence to sequence model (Model 1) is a decent baseline with good scores (56.80 BLEU). The Pointer generator model (Model 2) is aided by the copying mechanism. Thus, it is better modeled to include portions of the content from the generic response into the courteous response. This is corroborated by the increased score in CP (+9.33%). Its emotional accuracy is slightly reduced from Model 1 (-0.45%), probably because of eagerness to copy rather than generate. The advantage of the emotional embedding in Model 3 over Model 2 is reflected with the increased scores(+3.77%), because of its ability to better understand the emotional states and generate more appropriate courteous responses. The perplexity values are slightly reduced in Model 3 and Model 4, apparently because of the emotion embedding confusing the actual content from the conversation history. The final model performs decently better than other models. The reinforcement learning objective helps it to improve upon the desired metrics rather than just learn to be accurate at the token level.

Model		BLEU	ROUGE			PPL	CP	EA
			1	2	L			
1	***Seq2Seq***	56.80	63.8	59.06	64.52	58.21	68.34	82.43
2	***Seq2Seq + P***	66.11	69.92	64.85	66.40	**42.91**	**77.67**	81.98
3	***Seq2Seq + P + EE***	68.16	72.18	67.92	71.17	43.52	76.05	85.75
4	***Proposed Model***	**69.22**	**73.56**	**69.92**	**72.37**	43.77	77.56	**86.87**

Table 3: Results of various Models; P: Pointer Generator Model; EE: Emotional embedding

Model	F			CA			CoA		
	0	1	2	0	1	2	-1	0	1
Model 1	15.70	42.50	41.80	16.21	41.69	42.10	23.71	51.08	25.21
Model 2	14.23	42.77	43.00	15.62	39.65	44.73	22.05	39.43	38.52
Model 3	11.15	44.10	44.75	13.66	41.12	45.22	15.23	41.22	43.55
Our Model	10.05	44.90	44.60	13.85	38.48	47.67	14.11	41.11	44.78

Table 4: Human evaluation results for Fluency, Content Adequacy and Courtesy Appropriateness (All values are in percentages.)

Human evaluation results: In Table 4, we present the results of human evaluation. In case of fluency, our proposed model and the third model show similar performance, whereas Models 1 and 2 are relatively less fluent. Model 2 shows great improvement with respect to Model 1 as it is able to copy the content from the input. Also, for content adequacy our proposed model has been able to generate 38.48% moderate replies that have adequate amount of information in it while it generates around 47.67% correct responses that contain all the information present in the input. For courtesy appropriateness, Model 1 and Model 2 show lower performance while our proposed model has been able to capture the courteous behavior. As score 1 is given to the responses that are courteous as well as the nature of courteousness is in accordance to the conversation, it can be seen that our model achieves 44.78% performance level which is higher than the other models. From this evaluation, we can infer that the responses generated by our model are not only adequate in terms of information preservation, but also able to induce the courteous behavior by making the responses interesting and informative.

Error Analysis: We further analyse the outputs generated from our proposed model to perform a detailed qualitative analysis of the responses. In Table 5, we present few examples of the responses generated by the different models given the generic input. Some common forms of mistakes include:

1. Unknown Tokens: As Model 1 does not have the copying mechanism, the number of unknown

---

[2] we perform statistical significance tests (Welch, 1947) and it is conducted at 5% (0.05) significance level

Generic Input	Model 1	Model 2	Model 3	Our Model
dm us more info and well take a look into it for you	we'll look into it	im sorry to hear this please dm us more info and we'll take a look into it for you	were here to help please dm us more info and well take a look into it for you	were here to help please dm us more info and well take a look into it for you at the earliest
adjust the brightness via your display settings on your device	whos the brightness via your display settings on your device	were here to help adjust the brightness via your display settings on your device	we have several ways to change the display brightness on your device and were happy to help	thanks for reaching out we have several ways to change the display brightness on your device and were happy to help
we'll follow up with the store	we'd like to help well follow up	were here to help well follow up with the store	sorry to hear that well follow up with the store	thats disappointing to hear, we'll follow up with the store
can you confirm which platform you are using for video access ? what is the error ?	what is the error ?	I am sorry for the frustration ! can you confirm which platform you are using for video access ? what is the error ?	I am sorry to hear this can you confirm which platform you are using for video access? what is the error?	I am sorry for any frustration, can you please confirm which platform you are using for video access? Please tell us what is the error.
fill this form <url>	please fill this form <url>	were here to help fill this form <url>and I'll contact you at the earliest a	apologies for the hassle, please fill this form <url>and we'll contact you thank you for reaching out to us we will follow up with the store	i am sorry for the hassle, please fill this form <url> and ill contact you at the earliest

Table 5: Examples of Courteous Responses Generated by the Different Models

tokens is predicted the most in this. Also often the model predicts 'end of sequence' token just after the 'out of vocabulary' token, thus leaving sequences incomplete.

2. Wrong copying: Sometimes pointer network makes mistakes while copying (being influenced by language model): Gold: .. *which store in gillingham did you visit ?*; Predicted: .. *which store in belgium did you visit ?*

3. Mistakes in emotion identification: These mistakes are more prominent in Models 1 and 2 (they don't have emotional embeddings), where the generated courteous phrases denote mistakes in identifying the emotional state of the customer. For example, Gold: *you're very welcome, hope the kids have an amazing halloween !*; Predicted: *we apologize for the inconvenience. hope the kids have an amazing halloween !*

4. Extra information: Models 1, 2, 3 sometimes generate extra informative sentences than in the generic response: Gold: *please send us a dm*; Predicted: *please send us a dm please let us know if you did not receive it*

5. Contextually wrong courteous phrases: These mistakes are common across models while generating courteous phrases with content in them: Gold: *we want to help, reply by dm and ..*; Predicted: *im sorry you havent received it. please reply by dm and ..*

6. Difference in phrases: Generated responses differ from reference responses in their use of (equivalent) courteous phrases, and are hence wrongly penalized by some metrics.

7 Conclusion and Future Work

In this paper, we propose a new research problem of accentuating customer care responses with courteous behavior. Incorporation of courteousness is important for attaining user satisfaction and to improve the performance of the application leading to user retention. We successfully prepare a large benchmark corpus, created from the actual showcasing of courteous behavior by human professionals on Twitter. Our developed models appropriately model the dialogue history and are informed of the past emotional states through emotional embeddings. We have used both automatic and human based metrics for evaluating the performance of our model. In qualitative and quantitative analyses of the generated responses, we observe contextually correct courteous behavior and content preservation, along with minor inaccuracies as discussed in the error analysis section. Overall the performance of our model shows the variations in responses with the other models keeping the information and courtesy nature of the generated responses intact.

In future, along with the opportunity of extending the architectural designs and training methodologies to enhance the performance of our systems, we look forward to designing a specific component to enhance the natural language generation component of an end to end chatbot, by including appropriate mechanisms to interact with all its components (memory, database, and the dialog manager). Moreover, studies will be conducted on courtesy transfer for the other domains, and also transfer learning from one domain to the another (like customer care to hospitality).

Acknowledgement

Asif Ekbal acknowledges the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Charu C Aggarwal and ChengXiang Zhai. 2012. *Mining text data*. Springer Science & Business Media.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Keith Carlson, Allen Riddell, and Daniel Rockmore. 2017. Zero-shot style transfer in text using recurrent neural networks. *arXiv preprint arXiv:1711.04731*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Jonathan Herzig, Guy Feigenblat, Michal Shmueli-Scheuer, David Konopnicki, Anat Rafaeli, Daniel Altman, and David Spivak. 2016. Classifying emotions in customer support dialogues in social media. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 64–73.

Jonathan Herzig, Michal Shmueli-Scheuer, Tommy Sandbank, and David Konopnicki. 2017. Neural response generation for customer service based on personality traits. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 252–256.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

Chenyang Huang, Osmar Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 49–54.

Jiwei Li, Will Monroe, Alan Ritter, Daniel Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Daniel Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *TACL*, 6:373–389.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Yehong Peng, Yizhen Fang, Zhiwen Xie, and Guangyou Zhou. 2018. Topic-enhanced emotional conversation generation with attention mechanism. *Knowledge-Based Systems*.

Dinesh Raghu, Nikhil Gupta, et al. 2018. Hierarchical pointer memory network for task oriented dialogue. *arXiv preprint arXiv:1805.01216*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 129–140.

Sathish Reddy, Dinesh Raghu, Mitesh M Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 376–385.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. In *AAAI*.

Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. 2017. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 241–251.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Bernard L Welch. 1947. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Xianchao Wu, Ander Martinez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2049–2059.

Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2018. Reinforcing coherence for sequence to sequence model in dialogue generation. In *IJCAI*, pages 4567–4573.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*.

Xianda Zhou and William Yang Wang. 2018. Mojitalk: Generating emotional responses at scale. In *ACL*.

How to Avoid Sentences *Spelling* Boring? Towards a Neural Approach to Unsupervised Metaphor Generation

Zhiwei Yu and **Xiaojun Wan**
Center for Data Science, Peking University
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
`{yuzw,wanxiaojun}@pku.edu.cn`

Abstract

Metaphor generation attempts to replicate human creativity with language, which is an attractive but challengeable text generation task. Previous efforts mainly focus on template-based or rule-based methods and result in a lack of linguistic subtlety. In order to create novel metaphors, we propose a neural approach to metaphor generation and explore the shared inferential structure of a metaphorical usage and a literal usage of a verb. Our approach does not require any manually annotated metaphors for training. We extract the metaphorically used verbs with their metaphorical senses in an unsupervised way and train a neural language model from wiki corpus. Then we generate metaphors conveying the assigned metaphorical senses with an improved decoding algorithm. Automatic metrics and human evaluations demonstrate that our approach can generate metaphors with good readability and creativity.

1 Introduction

Metaphor is a kind of language filled with vitality and elasticity. It employs words in a way that deviates from their normal meaning to represent another concept (Li and Sporleder, 2010). Using metaphor allows us to express not just information but also real feelings and complex attitudes. There is a clear need in computational metaphor generation whose insightful results can be used in many applications from entertainment to education (Veale, 2016). Besides, a unified metaphor annotation procedure and creation of a large publicly available metaphor corpus are in great demands. Such resources make it possible to interpretate the identified metaphorical expressions and enhance the performance on other Natural Language Processing (NLP) applications (Shutova, 2010).

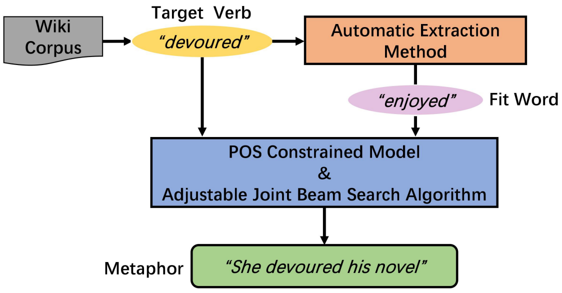


Figure 1: The Metaphor Generation Process

Although metaphor has a long history of academic studies in both philosophy and linguistics (Genesereth, 1980; G. and M., 1985), it still remains a tough problem for the NLP community. As the metaphor is hardly to be well-defined and modeled, little work focuses on the metaphor generation. Most of the previous efforts rely on hand-coded knowledge (Martin, 1990; Feldman and Narayanan, 2004; Agerri et al., 2007) , which heavily constrains the diversity of generated metaphors.

The end-to-end approach presented to sequence learning has been proved effective on the generation tasks like machine translation (Sutskever et al., 2014), abstractive summarization (Tan et al., 2017), product review generation (Zang and Wan, 2017) and multi-label classification (Yang et al., 2018). The approach is able to train a language model which can generate fluent and creative sentences with sufficient corpus. Unfortunately, in spite of the industrious exploration of the metaphor corpus, the annotated data available now is still far from training a good language model. To the best of our knowledge, there has been no work combining metaphor generation with the end-to-end approach.

In this paper, we propose a neural approach for metaphor generation trained with Wiki corpus rather than the limited annotated metaphor corpus, which assures the quality of the language model. The approach is shown in Figure 1. Relevant sta-

Proceedings of NAACL-HLT 2019, pages 861–871
Minneapolis, Minnesota, June 2 - June 7, 2019.

tistics demonstrate that the most frequent type of metaphor is expressed by a verb (Martin, 2006; Steen, 2010). In this paper, we focus on generating verb-oriented metaphors. We use an unsupervised method to extract the metaphorically used verbs in Wiki corpus. A metaphorical pair consists of a target verb (e.g. "devoured") and a fit word (e.g. "enjoyed"). And it is used to model the metaphorical usage of the target verb. According to Narayanan (1997), a metaphorical usage and a literal usage share inferential structure. We follow this intuition to find an intersection between the metaphorical usage and the literal usage of a word. For example, in "she devoured (enjoyed) his novels", the literal sense of "enjoyed" represents the contextual sense of "devoured" in such contexts. But the similarity between "enjoyed" and "devoured" is low, which means the target verb "devoured" is merely used in such sense and can be seen as a metaphorically used verb.

For metaphor generation, we first propose a POS constrained language model to generate a sentence containing a given verb and use an elaborately designed algorithm to consider its fit word simultaneously while decoding. For a profound exploration, we introduce automatic metrics as well as manual ways to evaluate the generation results. Experimental results demonstrate that our approach is capable of generating fluent and seemly metaphors. All the generated metaphors are novel and do not exist in the corpus.

To summarise, the contributions of our work are as follows[1]:

- As far as we know, our work is the first endeavor to adopt the end-to-end framework on metaphor generation. Besides, the proposed method does not require any manually labeled metaphor corpus.

- We automatically extract the verbs and their fit words in the corpus in an unsupervised way and use them (e.g. "devoured", "enjoyed") to model the metaphorical senses of the verbs for the generation process. We further explore the semantic shift of a verb by changing the adjustable factors.

- Both automatic metrics and human evaluation results demonstrate the efficacy of our model. Our model outperforms the baseline models on 3 aspects significantly[2].

2 Related Work

Metaphor is highly frequent in language and its computational processing is indispensable for real-world NLP applications addressing semantic tasks. Automatic processing of metaphor can be clearly divided into two subtasks: metaphor identification and metaphor interpretation (Shutova, 2010), little research has been devoted to the metaphor generation. In this subsection, we briefly review some prior work on metaphor generation.

Jones (1992) aims to generate a specific class of metaphors: Transparently-Motivated Metaphor, which is based on universal groundings that are often linked to bodily experience. Abe et al. (2006), Terai and Nakagawa (2010) generate metaphors in the form of "A (target) like B (vehicle)". They firstly compute the probabilistic relationship between concepts and words with a statistical analysis of language data and then select candidates to fill in the template. From a given mapping between the concepts of two domains, Hervás et al. (2007) present an approach to the application of metaphors for referring to concepts in an automatically generated text. Mason (2004) obtains domain-specific selectional preferences of verbs from large corpora and maps their common nominal arguments in other domains. The corresponding metaphorical mappings are achieved by such systematic variations and can generate simple conceptual metaphors in the form of: "A is (a) B". Ovchinnikova et al. (2014) also rely on characteristic predicate but use general propositions instead of the verb and adjective phrases. As some metaphors' target domain is lexically divorced from the source, Lederer (2016) identifies constellations of source-domain triggers in limited source domains. To make the metaphor generators more comprehensible and forceful, Veale (2016) presents a knowledge-base to generate XYZ metaphors such as "Bruce Wayne is the Donald Trump of Gotham City".

Previous methods make groundbreaking explorations on metaphor generation. However, these approaches mainly focus on modeling the phrase-level metaphor expressions and the generation process depends on the templates, which causes the lack of linguistic subtlety to some extent and

---

[1] https://github.com/ArleneYuZhiwei/Metaphor-Generation

[2] Based on two-tailed paired t-test with $p<0.05$.

they are not able to build a large publicly available metaphor corpus for further study. Our approach focuses on generating sentence-level metaphor in a template-free way.

3 Our Approach

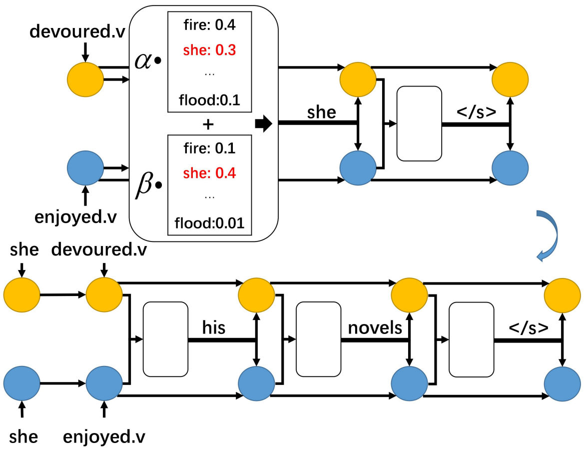


Figure 2: Adjustable Joint Model. Top: A metaphorical pair (e.g. "*devoured*" and "*enjoyed*") are given to the backward model, to generate the backward sequence. Bottom: Input the reversed backward sequence to the forward model, to generate the forward sequence. The inputs and outputs of the forward model are concatenated to form a metaphor.

Metaphors are ubiquitous in the normal corpus but lack of annotation. First, we extract the metaphorical pair which consists of a target word (e.g. "devoured") and its fit word (e.g. "enjoyed") automatically. Then we adopt an end-to-end neural framework to train a POS constrained language model which can generate a sentence containing the assigned verb. For metaphor inference, we apply an adjustable joint beam search algorithm to the decoding phase. In this way, the target verb is metaphorically used in the generated sentence. The proposed model is named Adjustable Joint Model and is shown in Figure 2.

3.1 Automatic Extraction of Metaphor Pairs

Our automatic extraction method is based on the hypotheses as follows:

H1. A metaphorical word is employed in the sentence to represent another concept and deviates from its normal meaning (Wilks, 1978; Li and Sporleder, 2010; Mao et al., 2018).

H2. The metaphorical senses of words occur with relatively lower frequency in the corpus than their literal senses do (Cameron, 2003; Martin, 2006; Mao et al., 2018).

Algorithm 1 Automatic Extraction

```
Require: $S$: the corpus, a set of sentences containing the target words
Require: $T$: the set of target words in $S$
Require: $M$: a trained CBOW model. $v_a^i$ denotes the input vector of word $a$. $v_a^o$ denotes the output vector of word $a$. $v_{context}^i$ denotes the average input vectors of the words in the $context$.
Require: $\varepsilon$: the threshold that determines the metaphoricity.
$Inflect(C)$ gets the inflections of each word in the set $C$. $Sim(x, y)$ calculates the cosine similarity between two vectors $x, y$.
$P = \emptyset$
For any sentence $s \in S$ and the contained target word $t \in T$
$context$ = the set consists of the words in $s$ excluding $t$
$syn$ = the set consists of the synonyms of $t$
$hyp$ = the set consists of the hypernyms of $t$
$candidates = Inflect(syn \cup hyp \cup t)$
$w = \underset{k}{argmax} Sim(v_k^o, v_{context}^i), k \in candidates$
$sim = Sim(v_w^i, v_t^i)$
if $sim < \varepsilon$ then:
    $P \leftarrow P \cup (t, w)$
return $P$
```

As metaphors begin their lives with marked rhetorical effects, whose comprehension requires a special imaginative leap (Nunberg, 1987), it is intuitive to assume that a metaphorical word can be distinguished from the literal one in the corpus with the violation of semantic constraints within a context. It has been proved that the dissimilarity between neural embeddings of the two words in a phrase is indicative of identifying the metaphoricity of the phrase (Shutova et al., 2016; Rei et al., 2017). Thus we find the semantic violation in the corpus based on word embeddings. The word embeddings are obtained by using Continuous Bag-of-Words Model(CBOW) (Mikolov et al., 2013).

Inspired by Mao et al. (2018), we use a fit word to model the contextual sense of the target word. To find a fit word for the target word t, we construct a candidate word set $candidates$ which consists of the target word as well as its synonyms and direct hypernyms extracted from WordNet(Miller, 1998). The target word may have several senses. Each of these senses has a set of corresponding synonyms and hypernyms. We extract the synonyms and hypernyms of all the verb senses. We augment the set with the inflections of the items in it. The word in the $candidates$ which has the highest similarity with the given context represents the contextual sense of the target verb in the sentence.

For example, "*i am afraid this spells trouble*" is a sentence in the corpus and the target word t is "*spells*". The word (e.g. "*means*") in $candidates$ has the highest similarity with the given context (e.g. "*i am afraid this [ ] trouble.*"), and is the fit word of t.

We then compute the similarity between the target verb (e.g. "*spells*") and its fit word (e.g. "*means*") . If the similarity is less than or equal to a threshold $\varepsilon = 0.6$[3], we extract the metaphorical word together with the fit word to form a metaphorical pair. The extraction process is described in Algorithm 1.

Follow the previous work (Nalisnick et al., 2016; Mao et al., 2018), we use OUT-IN vectors to measure the similarity between a fit word and its given context, use IN-IN vectors to measure the similarity between two words in a metaphorical pair[4] .

3.2 POS Constrained Language Model

Our goal is to generate a sentence containing a metaphorically used verb. However, the vanilla end-to-end model cannot guarantee the target word appearing in the generated sequence all the time, letting alone the appearance of a word with a specific part-of-speech. To solve this problem, we present a neural language model which can ensure an assigned verb to appear in the generated sentence. Our design is inspired by the asynchronous forward/backward generation model proposed by (Mou et al., 2016).

The POS constrained language model is trained end-to-end. Given a target verb as input, the model first generates the backward sequence starting from the target word w_t at position t of the sentence and ending up with "</s>" at position 0 of the sentence. n is the position of the last word in the sentence. $p(w_t^1)$ denotes the probability of the backward sequence. Then we reverse the output of the backward sequence as the input to the forward model. And it generates the rest part of the sentence accordingly. $p(w_t^n)$ denotes the probability of the forward sequence. The generated sentence is a connection of the input and output of the forward model whose probability can be decomposed as:

$$p(\overleftarrow{w_t^1}||\overrightarrow{w_t^n})=p(w_t)\prod_{i=1}^{t-1}p^{(bw)}(\overleftarrow{w_{t-i}}|\cdot)\prod_{i=1}^{n-t}p^{(fw)}(\overrightarrow{w_{t+i}}|\cdot), \tag{1}$$

where $p(\cdot||\cdot)$ means the probability of a particular backward/forward sequence. $p^{(bw)}(\overleftarrow{w_t}|\cdot)$ or $p^{(fw)}(\overrightarrow{w_t}|\cdot)$ indicates the probability of w_t given previous sequence $\cdot$ in the backward or forward model respectively. So far, the model is able to generate a sentence containing an assigned word. But the target word is not used as a verb in the generated sequence all the time.

We regard a word with various parts-of-speech as a polyseme. If the word is used with a specific part-of-speech which we are concerned about, we label it with the specific part-of-speech tag, otherwise, the word remains unchanged. In our case, we use a POS tagger (Bird et al., 2009) to label all the words in the metaphorical pairs we extracted into two categories: verb and the other. For instance, "*spells*" is labeled as "*spells.v*" in the corpus if it is used as a verb otherwise "*spells*". We train the POS constrained model with the labeled corpus, and we generate sentences like "*she spells.v her husband at the wheel*" rather than "*he whispered spells as he moved his hands.*" when given "*spells.v*" as input.

3.3 Adjustable Joint Beam Search

In the end-to-end model, the goal of the decoder is to find a sequence $\hat{y}$ which maximizes the conditional probability given by a specific model θ and an input sequence x:

$$\hat{y} = argmax_{y \in \mathcal{Y}} p_\theta(y|x), \tag{2}$$

where $\mathcal{Y}$ is a set of all the possible sequences in the output space. It is impracticable to explore the whole space. Models decompose this problem into a sequence of time steps. The original beam search algorithm produces a probability distribution at each time step over the vocabulary V. And the log function is applied on the probability distribution to get the score distribution. Instead of simply choosing the token with the highest score, the beam search algorithm keeps top k candidates from a large matrix of dimensions $k \times |V|$ which expands the search space, where k denotes

---

[3] Keep in line with Mao et al. (2018)

[4] IN vectors are input vectors of a trained CBOW model. OUT vectors are output vectors of a trained CBOW model.

Algorithm 2 Adjustable Joint Beam Search

```
Require: $k$ : beam size
Require: $\alpha$, $\beta$: adjustment factors
Require: $L$ : maximum sequence length
Require: $w_1, w_2$ : input words
$Decoder-Init(w_i, n)$ copies the initial state of the decoder $n$ times when inputting $w_i$ to the decoder.
$Decoder-Step(beam)$ calculates the score distributions on the beam.
$Top-K(scores, b)$ selects $b$ candidates with highest scores in the score distribution and returns corresponding beam ids ($beam.id$) and word ids ($beam.indices$).
$t = 0; l = k; n = k/2$
$beam1 \leftarrow Decoder-Init(w_1, n)$
$beam2 \leftarrow Decoder-Init(w_2, n)$
while $l > 0$ and $t < L$ do
    $scores1 = Decoder-Step(beam1)$
    $scores2 = Decoder-Step(beam2)$
    $scores = \alpha \cdot scores1 + \beta \cdot scores2$
    $beam1 = Top-K(scores, n)$
    $beam2.ids \leftarrow beam1.ids + n$
    $beam2.indices \leftarrow beam1.indices$
    $t = t + 1$
    $n_e$ ←number of ("</s>") selected in $beam1$ and $beam2$.
    $l = l - n_e$
return $beam1, beam2$
```

the beam size. When "</s>" is selected among the highest scoring candidates the beam is reduced by one. When the beam is zero, the beam search algorithm stops (Lowerre, 1976; Post and Vilar, 2018).

As the metaphorical usage of the verb is represented by the metaphorical pair. We need to generate a sentence for the target verb where the contextual sense of the target word equals to the literal sense of its fit word, which means both target word and its fit word should be suitable in the generated sentence. Given two different verbs in a metaphorical pair as inputs (e.g. "*devoured*.v", "*enjoyed*.v"), we hope to generate a same context for them. However, the original beam search algorithm can hardly choose the same candidates at each time step for them. Yu et al. (2018) propose the joint beam search algorithm for pun generation. The algorithm selects candidates for the two inputs according to the joint score distribution on all beam while decoding. Nevertheless, the semantic distance between the metaphorical sense and the literal sense of the target verb is not the same with the distance between the metaphorical sense and contextual sense of the target word. In addition, the frequencies of the words in each metaphorical pair differ, and there is no reason to believe that two words in a metaphorical pair have the same influence on generating metaphors. Therefore, we use the adjustment factors α, β to modify the weights of the score distributions for two inputs. The proposed adjustable joint beam search algorithm is described in Algorithm 2.

Different from the original beam search algorithm, to start the adjustable joint beam search algorithm, the initial states of the decoder are copied $k/2$ times for each input. At each time step, candidates are chosen from a weighed summation of the score distributions from two sets of the beam. The beam for the next time step is filled by taking the states corresponding to the selected beam ids and word ids. Half of the chosen words are duplicate but come from distinct beam, which means the outputs are one-to-one correspondent for the two distinct inputs. Although the corresponding two sequences select the same word at each time step, they have distinct hidden states, and thus their score distributions are different. The decoder finally outputs k sentences in parallel and the corresponding two sentences are the same except for the input words. We find an intersection between a metaphorical usage and a literal usage of a verb by this means. To avoid that the generated sentence is semantically inclined to the word in a metaphorical pair whose frequency is relatively higher and take the second hypothesis (**H2**) into consideration, we calculate the adjustment factors as follows:

$$\alpha = \sigma\left(1 - \frac{wf(word1)}{wf(word1) + wf(word2)}\right), \quad (3)$$

$$\beta = \sigma\left(1 - \frac{wf(word2)}{wf(word1) + wf(word2)}\right), \quad (4)$$

where $wf(a)$ denotes the frequency of the word a in the corpus. As metaphorical senses of words occur with relatively lower frequency in the corpus, we adjust the weights negatively correlated to the word frequency. And we use σ, the sigmoid function which is differentiable and widely used in the neural network models, to smooth the adjustment factors.

4 Experiments

4.1 Data Preprocessing

Adequate data is a prerequisite condition for training a good language model. It is unrealistic to achieve the goal based on the limited annotated metaphor corpus. In this paper, we use the English Wikipedia corpus to train the POS constrained language model. The corpus is split into sentences whose maximum length is 50 words. We lowercase and tokenize the sentences. All the numeric tokens are replaced with "#". We automatically extract 2812 metaphorical pairs from the corpus, and label words in them into two forms: "*word.v*" and "*word*". We use 461,685 sentences as the training set. We keep 120,000 most frequently occurring words and replace other words with the "<*unk*>" token. We call this processed corpus a **normal corpus**. For comparison, we extract the metaphors in the normal corpus using an unsupervised metaphor identification approach (Mao et al., 2018) and build a **metaphor corpus** with 310,908 sentences. We keep the same vocabulary size. To explore the influence of a fine-grained sense in the metaphor generation task, we use a Word Sense Disambiguation (WSD) tool[5] to label the verbs in the normal corpus. As the polysemes are regarded as different words which obviously increases the vocabulary size, we keep 165,000 most frequently occurring words. And this corpus is named **sense corpus**.

4.2 Compared Models

Since there has been no neural model applied on metaphor generation and the previous template-based models can not generate such verb-oriented metaphors, we implement six neural models for comparison and explore the intrinsic characteristics of metaphor generation. Models are trained on the **normal corpus**, unless otherwise specified.

Normal Model: This model is a basic end-to-end framework whose inputs are verbs and outputs are sentences. We use the teacher forcing algorithm (Williams and Zipser, 1989) while training. The input of the encoder is a verb, and the reference for the decoder is a sentence containing the verb.

POS constrained Model: This model combines the basic end-to-end framework with the POS constrained language model. The model can generate a sentence containing a given verb.

Fit Word Model: The model is trained in the same way as the POS constrained model. The input is a fit word, and we directly replace the fit word in the generated sentence with the corresponding target word.

Metaphor Based Model: The model is the same as the POS constrained model but trained on the **metaphor corpus**.

Uncommon Sense Model: Based on the second hypothesis(**H2**), metaphorical sense appears less common in the corpus than literal sense. It is intuitive to associate the metaphorical sense of the target word with a low-frequency sense. We keep the senses of a target word which appear in the corpus more than 9 times in a sense list. Then we choose the sense with the lowest frequency in the sense list as the metaphorical sense of the target verb. This model generates sentences similarly to the POS constrained model, except that it is trained on the **sense corpus.**

Adjustable Joint Model: The training process is exactly the same as the POS constrained model's, but we use the adjustable joint beam search algorithm while decoding.

4.3 Automatic Evaluation

We extract the metaphorical pairs automatically in the normal corpus and conduct experiments. The automatic evaluation results are shown in Table 1.

Model	l.ave	w.num	ppl.
Normal	10.34	16029	97.84
POS constrained	10.51	16361	133.24
Fit Word	10.70	16666	154.49
Metaphor Based	9.37	14526	**61.88**
Uncommon Sense	7.45	11569	96.52
Adjustable Joint	**11.35**	**16887**	97.03

Table 1: Automatic evaluation results for generated metaphors based on the automatically extracted verbs.

Each target verb may form distinct metaphorical pairs with different fit words. Both the normal model and the POS constrained model generate the same sentences for the same target verbs while the fit word model generates the same sentences for the same fit words. Taking the metaphorical pairs into consideration, adjustable joint model can generate various sentences when the target words are the same but used in diffe-

---

[5] https://github.com/alvations/pywsd

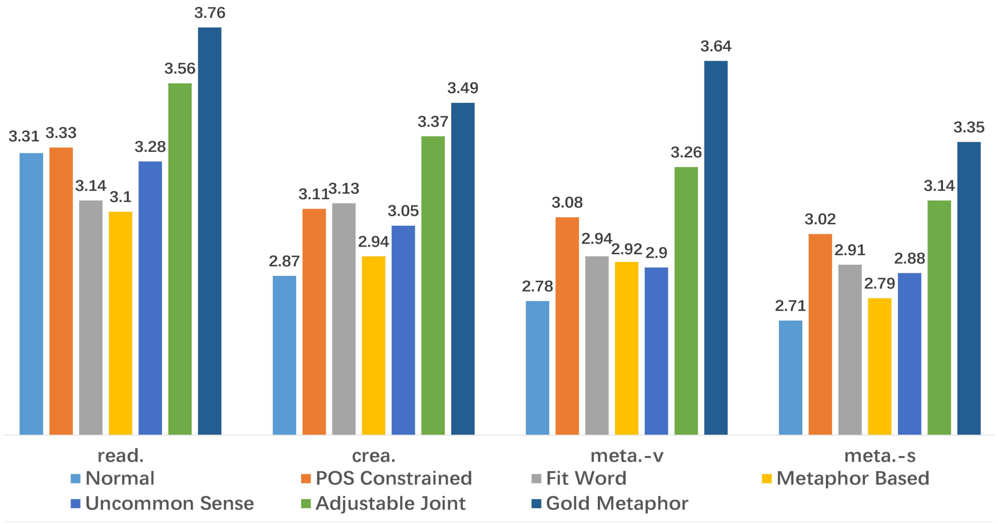


Figure 3: Results of human evaluation.

rent senses. For a fair quantitative analysis, each model generates 1555 sentences for distinct target verbs. **Length-average (l.ave)** is the average length of the generated sentences of each model. **Word.number(w.num)** is the total number of the distinct words in the generated sentences, and the adjustable joint model decodes words with the highest diversity compared to the other models. We use the language modeling toolkit SRILM[6] to evaluate the **perplexity scores (ppl.)**. Theoretically, the normal model should generate more fluent sentences without considering the constraint that a given word must appear in the outputs, however the ppl. is calculated as Eq 5 shows:

$$ppl = 10^{\wedge}\left(\frac{-logP(T)}{s.num + w.num - OOVs}\right), \qquad (5)$$

where $P(T)$ denotes the probabilities of all the sentences, $s.num$ denotes the total number of the sentences. $OOVs$ denotes the number of out-of-vocabulary words. The outputs of the metaphor based models tend to be shorter which results in a higher $P(T)$ and thus a low ppl. Although the adjustable joint model generates sentences with lexical constraints, it chooses the candidates with the highest joint score at each time step and uses more distinct words, and thus it obtains a high $P(T)$ as well as a large $w.num$. It achieves a relatively low ppl. What's more, all the sentences generated by the adjustable joint model are novel and do not exist in the corpus according to the duplicate checking . The fit word model decodes according to the fit words and then the fit words in the sentences are replaced by their target words directly, which results in a higher ppl. Since the amount of the training data corresponding to the inputs of the uncommon sense model is the least, the generated sentences are not so fluent.

4.4 Human Evaluation

For a thorough comparison, we select 80 gold metaphors with high confidence (>0.6) from the data set proposed by (Mohammad et al., 2016). Each verb in the data set was annotated for metaphoricity by 10 annotators and we use the verbs in the selected metaphors as the target words. As metaphor is such a creative and delicate language that automatic evaluation is not adequate. We ask native English speakers on Amazon Mechanical Turk to evaluate all the sentences generated by the neural models and corresponding gold metaphors in four aspects. Each sentence is scored from 1 to 5 by 5 judges. **Readability(read.)** indicates whether the sentence is fluent and consistent with the rules of grammar. **Creativity(crea.)** indicates whether the sentence is distinct and creative. **Metaphorical or Literal Usage of the Verb(meta.-v)** indicates whether the target word is used literally or metaphorically. 1 denotes that the usage of the verb is definitely literal and 5 denotes the verb is obviously metaphorically used. We display typical properties of metaphorical and literal senses as follows: literal usages tend to be more basic, straightforward meaning, more physical and closely tied to our senses; Metaphorical usages tend to be more abstract, more vague and often surprising, someti-

---

[6] http://www.speech.sri.com/projects/srilm/

Model	Generated Sentences
Target Verb: *absorbed*	Fit Word: *learn*
Normal	*it is absorbed by the united states .*
POS constrained	*it absorbed the united states in the early century .*
Fit Word	*they absorbed that they are able to find themselves .*
Metaphor Based	*they absorb water from the water to be used as a result of the disease .*
Uncommon Sense	*he absorbed more than a few hundred feet .*
Adjustable Joint	***he absorbed his studies at the university of birmingham .***
Gold Metaphor	*he absorbed the knowledge or beliefs of his tribe .*
Target Verb: *pour*	Fit Word: *crowd*
Normal	*while drinking , he is able to kill him .*
POS constrained	*being poured , the band was released by the band .*
Fit Word	*they poured for the first time , and the team was the first to win the championship .*
Metaphor Based	*it was poured in # and # .*
Uncommon Sense	*she poured the police and her husband .*
Adjustable Joint	***millions of trees poured out of the area .***
Gold Metaphor	*People were pouring out of the theater.*

Table 2: Examples of generated sentences.

mes bring in imagery from a different domain. The aspects above are rated according to comparison. **Metaphoricity of the sentence(meta.-s)** indicates whether the sentence is a metaphor. The score ratings are defined as: 1. not a metaphor at all; 2. pathetic metaphor; 3. not-so-bad metaphor; 4. interesting metaphor; 5. gold metaphor.

Results are shown in Figure 3. The metaphor based model generates shorter sentences which results in a low **ppl.**, but it replicates some words at times, which makes it difficult for human to interpret. The fit word model replaces the fit words with the target words directly and may break the grammatical collocations. Although the adjustable joint model ensures the intersection sense of two words in the metaphorical pair to appear in one sentence, the generated utterance is still readable. As for the **crea.**, without any lexical constraint, the normal model always generates the sentences which are similar to the relatively high-frequency sentences in the corpus and results in the lack of novelty. In contrast, the target words are used in its less used literal senses in the uncommon sense model and the results seem to be kind of creative. The adjustable joint model performs closely to the gold metaphor on the **crea.**, which proves its ability in modeling the creative usage of the verbs. **meta.-v** directly reveals the capability of modeling the metaphorical senses of the target verb. The normal model is uncompetitive as it can not even ensure the appearance of the target verb. The difference between the POS constrained model and the uncommon sense model inspires us that although the metaphorical senses of the words occur with relatively lower frequency in the corpus than their literal senses, the metaphorical senses cannot be easily defined as the most uncommon senses of the target verbs, and thus the exploration on modeling the metaphorical senses is essential. As for the **meta.-s**, it reveals the comprehensiveness of the sentence-level metaphor generation and reminds us of modeling the metaphors in a more thoughtful perspective. The adjustable joint model outperforms the other models on the 3 aspects (**crea.**, **meta.-v**, **meta.-s**) significantly with paired t-test with $p < 0.05$.

4.5 Case Study

To illustrate concretely, we show some examples generated by different models in Table 2. Both the normal model and the POS constrained model generate sentences without considering the senses of the verbs and tend to generate monotonous sentences. And the fit word model generates sentences which are suitable for the fit word but inappropriate for the target word. For example, "*they crowed for the first time*" is fluent while "*they poured for the first time*" is strange. Other models decoding with a sense constraint generate sentences conveying the assigned senses to different degree. However, the uncommon sense model is fine-grained and the training corpus is annotated with WSD tools which results in not only the lack of corpora for some senses but also a sense-label error, as there are no WSD tools that could tag the senses of the verbs with a high precision (> 0.6) (Luo et al., 2018b,a). To solve these problems, we use the extracted metaphorical pairs to depict a metaphorical sense and the generated metaphors are readable.

We also explore the shared inferential structure of a metaphorical usage and a literal usage of a verb by changing the adjustable factors. Table 2 demonstrates the semantic shift of the verb. The POS constrained model can be seen as a special adjustable joint model whose adjustable factors are $\alpha = 1, \beta = 0$ and it generates sentences only considering the target words. In this way the target words are literally used. For contrast, the fit word model with a special setting of the adjustable factors $\alpha = 0, \beta = 1$ generates sentences completely dependent on the fit words, which may result in sentences that are not semantically appropriate. When α, β are caculated by Eq 3 and Eq 4, the generated sentences covey the metaphorical senses of verbs.

5 Conclusion

We make an exploration on verb-oriented metaphor generation and propose a neural approach on automatic metaphor generation. The approach identifies metaphorically used verbs in the normal corpus and extracts the metaphorical pairs in the sentences. We propose a POS constrained model to ensure the appearance of the given verbs and decode with the adjustable joint beam search algorithm, which takes the metaphorical senses of the given verbs into consideration. We generate metaphors which are not only fluent and readable but also creative. However, we can only generate metaphors based on the verbs whose metaphorical senses can be found in the corpus. For future work, we will explore techniques to generate metaphors without extracting its fit words in the corpus and improve the quality of generated metaphors.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate Hongyu Zang for the detailed suggestions and thoughts. We thank Fuli Luo, Pengcheng Yang, Rui Mao and the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Keiga Abe, Kayo Sakamoto, and Masanori Nakagawa. 2006. A computational model of the metaphor generation process. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 28.

Rodrigo Agerri, John Barnden, Mark Lee, and Alan Wallington. 2007. Metaphor, inference and domain independent mappings. In *Proceedings of RANLP*, pages 17–23. Citeseer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. Ö'Reilly Media, Inc.".

Lynne Cameron. 2003. *Metaphor in educational discourse*. A&C Black.

Jerome Feldman and Srinivas Narayanan. 2004. Embodied meaning in a neural theory of language. *Brain and language*, 89(2):385–392.

Lakoff G. and Johnson M. 1985. Metaphors we live by. *Artif. Intell.*, 27(3):357–361.

Michael R. Genesereth. 1980. Metaphors and models. In *Proceedings of the 1st Annual National Conference on Artificial Intelligence, Stanford University, CA, USA, August 18-21, 1980.*, pages 208–211.

Raquel Hervás, Rui P Costa, Hugo Costa, Pablo Gervás, and Francisco C Pereira. 2007. Enrichment of automatically generated texts using metaphor. In *Mexican International Conference on Artificial Intelligence*, pages 944–954. Springer.

Mark Alan Jones. 1992. Generating a specific class of metaphors. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 321–323. Association for Computational Linguistics.

Jenny Lederer. 2016. Finding metaphorical triggers through source (not target) domain lexicalization patterns. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 1–9.

Linlin Li and Caroline Sporleder. 2010. Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 297–300.

Bruce T Lowerre. 1976. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1402–1411.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating glosses into neural word sense disambiguation. *arXiv preprint arXiv:1805.08028*.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and wordnet based metaphor identification and interpretation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

James H Martin. 1990. *A computational model of metaphor interpretation*. Academic Press Professional, Inc.

James H Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension. *TRENDS IN LINGUISTICS STUDIES AND MONOGRAPHS*, 171:214.

Zachary J. Mason. 2004. Cormet: A computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. *arXiv preprint arXiv:1607.00970*.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84. International World Wide Web Conferences Steering Committee.

Srini Narayanan. 1997. Knowledge-based action representations for metaphor and aspect (karma). *Computer Science Division, University of California at Berkeley dissertation*.

Geoffrey Nunberg. 1987. Poetic and prosaic metaphors. *Theoretical Issues in Natural Language Processing 3*.

Ekaterina Ovchinnikova, Vladimir Zaytsev, Suzanne Wertheim, and Ross Israel. 2014. Generating conceptual metaphors from proposition stores. *arXiv preprint arXiv:1409.7619*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Marek Rei, Luana Bulat, Douwe Kiela, and Ekaterina Shutova. 2017. Grasping the finer point: A supervised similarity network for metaphor detection. *arXiv preprint arXiv:1709.00575*.

Ekaterina Shutova. 2010. Models of metaphor in NLP. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 688–697.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 160–170.

Gerard Steen. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1171–1181.

Asuka Terai and Masanori Nakagawa. 2010. A computational system of metaphor generation with evaluation mechanism. In *International Conference on Artificial Neural Networks*, pages 142–147. Springer.

Tony Veale. 2016. Round up the usual suspects: Knowledge-based metaphor generation. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 34–41.

Yorick Wilks. 1978. Making preferences more active. *Artif. Intell.*, 11(3):197–223.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3915–3926.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1650–1660.

Hongyu Zang and Xiaojun Wan. 2017. Towards automatic generation of product reviews from aspect-sentiment scores. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 168–177.

Incorporating Context and External Knowledge for Pronoun Coreference Resolution

Hongming Zhang♣[*], **Yan Song**♠, **and Yangqiu Song**♣
♣Department of CSE, The Hong Kong University of Science and Technology
♠Tencent AI Lab
`hzhangal@cse.ust.hk, clksong@gmail.com, yqsong@cse.ust.hk`

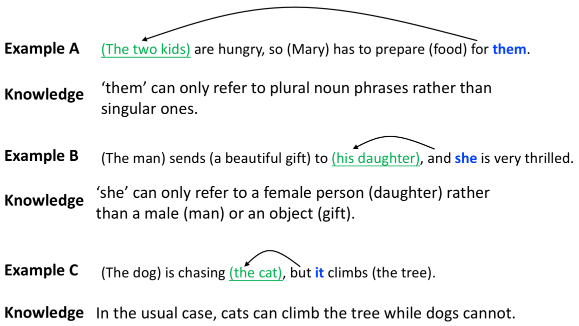
Abstract

Linking pronominal expressions to the correct references requires, in many cases, better analysis of the contextual information and external knowledge. In this paper, we propose a two-layer model for pronoun coreference resolution that leverages both context and external knowledge, where a knowledge attention mechanism is designed to ensure the model leveraging the appropriate source of external knowledge based on different context. Experimental results demonstrate the validity and effectiveness of our model, where it outperforms state-of-the-art models by a large margin.

1 Introduction

The question of how human beings resolve pronouns has long been of interest to both linguistics and natural language processing (NLP) communities, for the reason that pronoun itself has weak semantic meaning (Ehrlich, 1981) and brings challenges in natural language understanding. To explore solutions for that question, pronoun coreference resolution (Hobbs, 1978) was proposed. As an important yet vital sub-task of the general coreference resolution task, pronoun coreference resolution is to find the correct reference for a given pronominal anaphor in the context and has been shown to be crucial for a series of downstream tasks (Mitkov, 2014), including machine translation (Mitkov et al., 1995), summarization (Steinberger et al., 2007), information extraction (Edens et al., 2003), and dialog systems (Strube and Müller, 2003).

Conventionally, people design rules (Hobbs, 1978; Nasukawa, 1994; Mitkov, 1998) or use features (Ng, 2005; Charniak and Elsner, 2009; Li et al., 2011) to resolve the pronoun coreferences. These methods heavily rely on the coverage and quality of the manually defined rules and features. Until recently, end-to-end solution (Lee et al., 2017) was proposed towards solving the general coreference problem, where deep learning models were used to better capture contextual information. However, training such models on annotated corpora can be biased and normally does not consider external knowledge.



Example A (The two kids) are hungry, so (Mary) has to prepare (food) for **them**.

Knowledge 'them' can only refer to plural noun phrases rather than singular ones.

Example B (The man) sends (a beautiful gift) to (his daughter), and **she** is very thrilled.

Knowledge 'she' can only refer to a female person (daughter) rather than a male (man) or an object (gift).

Example C (The dog) is chasing (the cat), but **it** climbs (the tree).

Knowledge In the usual case, cats can climb the tree while dogs cannot.

Figure 1: Pronoun coreference examples, where each example requires different knowledge for its resolution. Blue bold font refers to the target pronoun, where the correct noun reference and other candidates are marked by green underline and brackets, respectively.

Despite the great efforts made in this area in the past few decades (Hobbs, 1978; Mitkov, 1998; Ng, 2005; Rahman and Ng, 2009), pronoun coreference resolution remains challenging. The reason behind is that the correct resolution of pronouns can be influenced by many factors (Ehrlich, 1981); many resolution decisions require reasoning upon different contextual and external knowledge (Rahman and Ng, 2011), which is also proved in other NLP tasks (Song et al., 2017, 2018; Zhang et al., 2018). Figure 1 demonstrates such requirement with three examples, where Example A depends on the plurality knowledge that 'them' refers to plural noun phrases; Example B illustrates the gender requirement of pronouns where 'she' can

---

[*] This work was done during the internship of the first author in Tencent AI Lab.

Proceedings of NAACL-HLT 2019, pages 872–881
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

only refer to a female person (girl); Example C requires a more general type of knowledge[1] that 'cats can climb trees but a dog normally does not'. All of these knowledge are difficult to be learned from training data. Considering the importance of both contextual information and external human knowledge, how to jointly leverage them becomes an important question for pronoun coreference resolution.

In this paper, we propose a two-layer model to address the question while solving two challenges of incorporating external knowledge into deep models for pronoun coreference resolution, where the challenges include: first, different cases have their knowledge preference, i.e., some knowledge is exclusively important for certain cases, which requires the model to be flexible in selecting appropriate knowledge per case; second, the availability of knowledge resources is limited and such resources normally contain noise, which requires the model to be robust in learning from them.

Consequently, in our model, the first layer predicts the relations between candidate noun phrases and the target pronoun based on the contextual information learned by neural networks. The second layer compares the candidates pair-wisely, in which we propose a knowledge attention module to focus on appropriate knowledge based on the given context. Moreover, a softmax pruning is placed in between the two layers to select high confident candidates. The architecture ensures the model being able to leverage both context and external knowledge. Especially, compared with conventional approaches that simply treat external knowledge as rules or features, our model is not only more flexible and effective but also interpretable as it reflects which knowledge source has the higher weight in order to make the decision. Experiments are conducted on a widely used evaluation dataset, where the results prove that the proposed model outperforms all baseline models by a great margin.[2]

Above all, to summarize, this paper makes the following contributions:

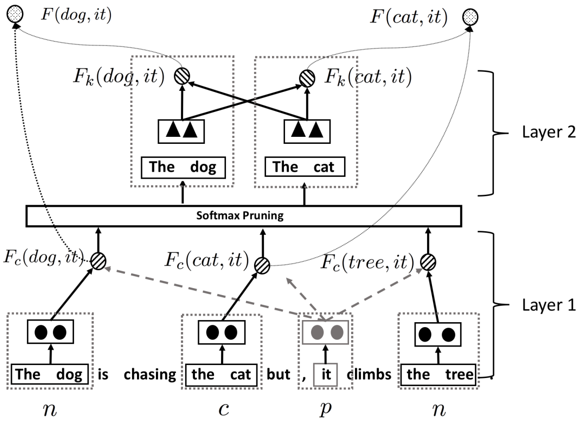


Figure 2: The architecture of the two-layer model for pronoun coreference resolution. The first layer encodes the contextual information for computing F_c. The second layer leverages external knowledge to score F_k. A pruning layer is applied in between the two layers to control computational complexity. The dashed boxes in the first and second layer refer to span representation and knowledge scoring, respectively.

1. We propose a two-layer neural model to combine contextual information and external knowledge for the pronoun coreference resolution task.
2. We propose a knowledge attention mechanism that allows the model to select salient knowledge for different context, which predicts more precisely and can be interpretable through the learned attention scores.
3. With our proposed model, the performance of pronoun coreference resolution is boosted by a great margin over the state-of-the-art models.

2 The Task

Following the conventional setting (Hobbs, 1978), the task of pronoun coreference resolution is defined as: for a pronoun p and a candidate noun phrase set $\mathcal{N}$, the goal is to identify the correct non-pronominal references set[3] $\mathcal{C}$. the objective is to maximize the following objective function:

$$\mathcal{J} = \frac{\sum_{c\in\mathcal{C}} e^{F(c,p)}}{\sum_{n\in\mathcal{N}} e^{F(n,p)}}, \tag{1}$$

where c is the correct reference and n the candidate noun phrase. $F(\cdot)$ refers to the overall coreference scoring function for each n regarding p. Following (Mitkov, 1998), all non-pronominal noun phrases in the recent three sentences of the pronoun p are selected to form N.

Particularly in our setting, we want to leverage both the local contextual information and external

---

[1] This is normally as selectional preference (SP) (Hobbs, 1978), which is defined as given a predicate (verb), a human has the preference for its argument (subject in this example).

[2] All code and data are available at:`https://github.com/HKUST-KnowComp/Pronoun-Coref`.

[3] It is possible that a pronoun has multiple references.

knowledge in this task, thus for each n and p, $F(.)$ is decomposed into two components:

$$F(n,p) = F_c(n,p) + F_k(n,p), \quad (2)$$

where $F_c(n,p)$ is the scoring function that predicts the relation between n and p based on the contextual information; $F_k(n,p)$ is the scoring function that predicts the relation between n and p based on the external knowledge. There could be multiple ways to compute F_c and F_k, where a solution proposed in this paper is described as follows.

3 The Model

The architecture of our model is shown in Figure 2, where we use two layers to incorporate contextual information and external knowledge. Specifically, the first layer takes the representations of different n and the p as input and predict the relationship between each pair of n and p, so as to compute F_c. The second layer leverages the external knowledge to compute F_k, which consists of pair-wise knowledge score f_k among all candidate n. To enhance the efficiency of the model, a softmax pruning module is applied to select high confident candidates into the second layer. The details of the aforementioned components are described in the following subsections.

3.1 Encoding Contextual Information

Before F_c is computed, the contextual information is encoded through a span[4] representation (SR) module in the first layer of the model. Following Lee et al. (2017), we adopt the standard bidirectional LSTM (biLSTM) (Hochreiter and Schmidhuber, 1997) and the attention mechanism (Bahdanau et al., 2015) to generate the span representation, as shown in Figure 3. Given that the initial word representations in a span n_i are $\mathbf{x}_1, ..., \mathbf{x}_T$,

we denote their representations $\mathbf{x}_1^*, ..., \mathbf{x}_T^*$ after encoded by the biLSTM. Then we obtain the inner-span attention by

$$a_t = \frac{e^{\alpha_t}}{\sum_{k=1}^{T} e^{\alpha_k}}, \quad (3)$$

where α_t is computed via a standard feed-forward neural network[5] $\alpha_t = NN_\alpha(\mathbf{x}_t^*)$. Thus, we have the weighted embedding of each span $\hat{x}_i$ through

$$\hat{\mathbf{x}}_i = \sum_{k=1}^{T} a_k \cdot \mathbf{x}_k. \quad (4)$$

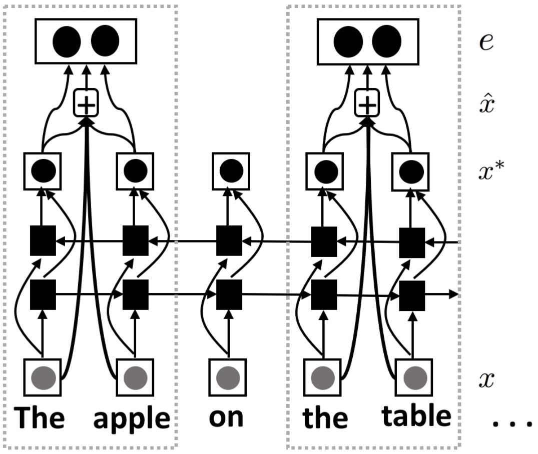


Figure 3: The structure of span representation. Bidirectional LSTM and inner-span attention mechanism are employed to capture the contextual information.

Afterwards, we concatenate the starting ($\mathbf{x}_{start}^*$) and ending ($\mathbf{x}_{end}^*$) embedding of each span, as well as its weighted embedding ($\hat{\mathbf{x}}_i$) and the length feature ($\phi(i)$) to form its final representation e:

$$\mathbf{e}_i = [\mathbf{x}_{start}^*, \mathbf{x}_{end}^*, \hat{\mathbf{x}}_i, \phi(i)]. \quad (5)$$

Once the span representation of $n \in \mathcal{N}$ and p are obtained, we compute F_c for each n with a standard feed-forward neural network:

$$F_c(n,p) = NN_c([\mathbf{e}_n, \mathbf{e}_p, \mathbf{e}_n \odot \mathbf{e}_p]), \quad (6)$$

where $\odot$ is the element-wise multiplication.

3.2 Processing External Knowledge

In the second layer of our model, external knowledge is leveraged to evaluate all candidate n so as to give them reasonable F_k scores. In doing so, each candidate is represented as a group of features from different knowledge sources, e.g., 'the cat' can be represented as a singular noun, unknown gender creature, and a regular subject of the predicate verb 'climb'. For each candidate, we conduct a series of pair-wise comparisons between it and all other ones to result in its F_k score. An attention mechanism is proposed to perform the comparison and selectively use the knowledge features. Consider there exists noise in external knowledge, especially when it is automatically

---

[4] Both noun phrases and the pronoun are treated as spans.

[5] We use NN to present feed-forward neural networks throughout this paper.

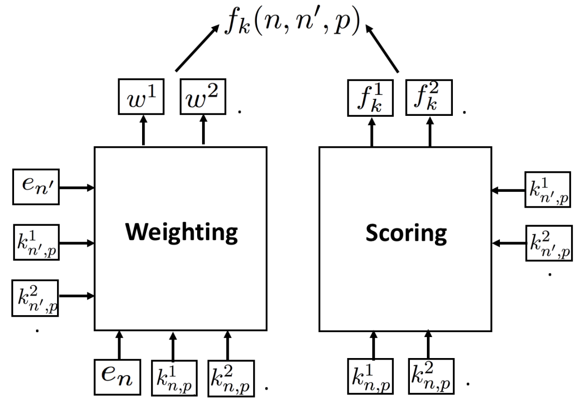


Figure 4: The structure of the knowledge attention module. For each feature k_i from knowledge source i, the the weighting component predict its weight w^i and the scoring component computes its knowledge score f_k^i. Then a weighted sum is obtained for f_k.

generated, such attention mechanism ensures that, for each candidate, reliable and useful knowledge is utilized rather than ineffective ones. The details of the knowledge attention module and the overall scoring are described as follows.

Knowledge Attention Figure 4 demonstrates the structure of the knowledge attention module, where there are two components: (1) weighting: assigning weights to different knowledge features regarding their importance in the comparison; (2) scoring: valuing a candidate against another one based on their features from different knowledge sources. Assuming that there are m knowledge sources input to our model, each candidate can be represented by m different features, which are encoded as embeddings. Therefore, two candidates n and n' regarding p have their knowledge feature embeddings $\mathbf{k}^1_{n,p}, \mathbf{k}^2_{n,p}, ..., \mathbf{k}^m_{n,p}$ and $\mathbf{k}^1_{n',p}, \mathbf{k}^2_{n',p}, ..., \mathbf{k}^m_{n',p}$, respectively. The weighting component receives all features $\mathbf{k}$ for n and n', and the span representations $\mathbf{e}_n$ and $\mathbf{e}_{n'}$ as input, where $\mathbf{e}_n$ and $\mathbf{e}_{n'}$ help selecting appropriate knowledge based on the context. As a result, for a candidate pair (n, n') and a knowledge source i, its knowledge attention score is computed via

$$\beta_i(n,n',p) = NN_{ka}([\mathbf{o}^i_{n,p}, \mathbf{o}^i_{n',p}, \mathbf{o}^i_{n,p} \odot \mathbf{o}^i_{n',p}]), \tag{7}$$

where $\mathbf{o}^i_{n,p} = [\mathbf{e}_n, \mathbf{k}^i_{n,p}]$ and $\mathbf{o}^i_{n',p} = [\mathbf{e}_{n'}, \mathbf{k}^i_{n',p}]$ are the concatenation of span representation and external knowledge embedding for candidate n and n' respectively. The weight for features from different knowledge sources is thus computed via

$$w_i = \frac{e^{\beta_i}}{\sum_{j=1}^{m} e^{\beta_j}}. \tag{8}$$

Similar to the weighting component, for each feature i, we compute its score $f_k^i(n,n',p)$ for n against n' in the scoring component through

$$f_k^i(n,n',p) = NN_{ks}([\mathbf{k}^i_{n,p}, \mathbf{k}^i_{n',p}, \mathbf{k}^i_{n,p} \odot \mathbf{k}^i_{n',p}]). \tag{9}$$

where it is worth noting that we exclude $\mathbf{e}$ in this component for the reason that, in practice, the dimension of $\mathbf{e}$ is normally much higher than $\mathbf{k}$. As a result, it could dominate the computation if $\mathbf{e}$ and $\mathbf{k}$ is concatenated.[6]

Once the weights and scores are obtained, we have a weighted knowledge score for n against n':

$$f_k(n,n',p) = \sum_{i=1}^{m} w_i \cdot f_k^i(n,n',p). \tag{10}$$

Overall Knowledge Score After all pairs of n and n' are processed by the attention module, the overall knowledge score for n is computed through the averaged $f_k(n,n',p)$ over all n':

$$F_k(n,p) = \frac{\sum_{n' \in \mathcal{N}_o} f_k(n,n',p)}{|\mathcal{N}_o|}, \tag{11}$$

where $\mathcal{N}_o = \mathcal{N} - n$ for each n.

3.3 Softmax Pruning

Normally, there could be many noun phrases that serve as the candidates for the target pronoun. One potential obstacle in the pair-wise comparison of candidate noun phrases in our model is the squared complexity $O(|\mathcal{N}|^2)$ with respect to the size of $\mathcal{N}$. To filter out low confident candidates so as to make the model more efficient, we use a softmax-pruning module between the two layers in our model to select candidates for the next step. The module takes F_c as input for each n, uses a softmax computation:

$$\hat{F}_c(n,p) = \frac{e^{F_c(n,p)}}{\sum_{n_i \in \mathcal{N}} e^{F_c(n_i,p)}}. \tag{12}$$

where candidates with higher $\hat{F}_c$ are kept, based on a threshold t predefined as the pruning standard. Therefore, if candidates have similar F_c

---

[6] We do not have this concern for the weighting component because the softmax (c.f. Eq. 8) actually amplifies the difference of β even if they are not much differentiated.

type	train	dev	test	all
Third Personal	21,828	2,518	3,530	27,876
Possessive	7,749	1,007	1,037	9,793
All	29,577	3,525	4,567	37,669

Table 1: Statistics of the evaluation dataset. Number of selected pronouns are reported.

scores, the module allow more of them to proceed to the second layer. Compared with other conventional pruning methods (Lee et al., 2017, 2018) that generally keep a fixed number of candidates, our pruning strategy is more efficient and flexible.

4 Experiment Settings

4.1 Data

The CoNLL-2012 shared task (Pradhan et al., 2012) corpus is used as the evaluation dataset, which is selected from the Ontonotes 5.0[7]. Following conventional approaches (Ng, 2005; Li et al., 2011), for each pronoun in the document, we consider candidate n from the previous two sentences and the current sentence. For pronouns, we consider two types of them following Ng (2005), i.e., third personal pronoun (*she*, *her*, *he*, *him*, *them*, *they*, *it*) and possessive pronoun (*his*, *hers*, *its*, *their*, *theirs*). Table 1 reports the number of the two type pronouns and the overall statistics for the experimental dataset. According to our selection range of candidate n, on average, each pronoun has 4.6 candidates and 1.3 correct references.

4.2 Knowledge Types

In this study, we use two types of knowledge in our experiments. The first type is linguistic features, i.e., plurality and animacy & gender. We employ the Stanford parser[8], which generates plurality, animacy, and gender markups for all the noun phrases, to annotate our data. Specifically, the plurality feature denotes each n and p to be singular or plural. For each candidate n, if its plurality status is the same as the target pronoun, we label it 1, otherwise 0. The animacy & gender (AG) feature denotes whether a n or p is a living object, and being male, female, or neutral if it is alive. For each candidate n, if its AG feature matches the target pronoun's, we label it 1, otherwise 0.

The second type is the selectional preference (SP) knowledge. For this knowledge, we create a knowledge base by counting how many times a predicate-argument tuple appears in a corpus and use the resulted number to represent the preference strength. Specifically, we use the English Wikipedia[9] as the base corpus for such counting. Then we parse the entire corpus through the Stanford parser and record all dependency edges in the format of *(predicate, argument, relation, number)*, where predicate is the governor and argument the dependent in the original parsed dependency edge[10]. Later for sentences in the training and test data, we firstly parse each sentence and find out the dependency edge linking p and its corresponding predicate. Then for each candidate[11] n in a sentence, we check the previously created SP knowledge base and find out how many times it appears as the argument of different predicates with the same dependency relation (i.e., *nsubj* and *dobj*). The resulted frequency is grouped into the following buckets [1, 2, 3, 4, 5-7, 8-15, 16-31, 32-63, 64+] and we use the bucket id as the final SP knowledge. Thus in the previous example:

The dog is chasing *the cat* but **it** climbs *the tree*.

Its parsing result indicates that '**it**' is the subject of the verb 'climb'. Then for '*the dog*', '*the cat*', and '*the tree*', we check their associations with 'climb' in the knowledge base and group them in the buckets to form the SP knowledge features.

4.3 Baselines

Several baselines are compared in this work. The first two are conventional unsupervised ones:

- **Recent Candidate**, which simply selects the most recent noun phrase that appears in front of the target pronoun.
- **Deterministic** model (Raghunathan et al., 2010), which proposes one multi-pass seive model with human designed rules for the coreference resolution task.

Besides the unsupervised models, we also compare with three representative supervised ones:

- **Statistical** model, proposed by Clark and Manning (2015), uses human-designed entity-level

---

[7] https://catalog.ldc.upenn.edu/LDC2013T19

[8] https://stanfordnlp.github.io/CoreNLP/

[9] https://dumps.wikimedia.org/enwiki/

[10] In Stanford parser results, when a verb is a linking verb (e.g., am, is), an 'nsubj' edge is created between its predicative and subject. Thus for this case the predicative is treated as the predicate for the subject (argument) in our study.

[11] If a noun phrase contains multiple words, we use the parsed result to locate its keyword and use it to represent the entire noun phrase.

Model	Third Personal Pronoun			Possessive Pronoun			All		
	P	R	F1	P	R	F1	P	R	F1
Recent Candidate	50.7	40.0	44.7	64.1	45.5	53.2	54.4	41.6	47.2
Deterministic (Raghunathan et al., 2010)	68.7	59.4	63.7	51.8	64.8	57.6	62.3	61.0	61.7
Statistical (Clark and Manning, 2015)	69.1	62.6	65.7	58.0	65.3	61.5	65.3	63.4	64.3
Deep-RL (Clark and Manning, 2016)	72.1	68.5	70.3	62.9	74.5	68.2	68.9	70.3	69.6
End2end (Lee et al., 2018)	75.1	83.7	79.2	73.9	82.1	77.8	74.8	83.2	78.8
Feature Concatenation	73.5	88.3	80.2	72.5	87.3	79.2	73.2	87.9	79.9
The Complete Model	75.4	87.9	**81.2**	74.9	87.2	**80.6**	75.2	87.7	**81.0**

Table 2: Pronoun coreference resolution performance of different models on the evaluation dataset. Precision (P), recall (R), and F1 score are reported, with the best one in each F1 column marked bold.

features between clusters and mentions for coreference resolution.

- **Deep-RL** model, proposed by Clark and Manning (2016), a reinforcement learning method to directly optimize the coreference matrix instead of the traditional loss function.
- **End2end** is the current state-of-the-art coreference model (Lee et al., 2018), which performs in an end-to-end manner and leverages both the contextual information and a pre-trained language model (Peters et al., 2018).

Note that the Deterministic, Statistical, and Deep-RL models are included in the Stanford CoreNLP toolkit[12], and experiments are conducted with their provided code. For End2end, we use their released code[13] and replace its mention detection component with gold mentions for the fair comparison.

To clearly show the effectiveness of the proposed model, we also present a variation of our model as an extra baseline to illustrate the effect of different knowledge incorporation manner:

- **Feature Concatenation**, a simplified version of the complete model that removes the second knowledge processing layer, but directly treats all external knowledge embeddings as features and concatenates them to span representations.

4.4 Implementation

Following previous work (Lee et al., 2018), we use the concatenation of the 300d GloVe embeddings (Pennington et al., 2014) and the ELMo (Peters et al., 2018) embeddings as the initial word representations. Out-of-vocabulary words are initialized with zero vectors. Hyper-parameters are set as follows. The hidden state of the LSTM module is set to 200, and all the feed-forward networks in our model have two 150-dimension hidden layers. The default pruning threshold t for softmax pruning is set to 10^{-7}. All linguistic features (plurality and AG) and external knowledge (SP) are encoded as 20-dimension embeddings.

For model training, we use cross-entropy as the loss function and Adam (Kingma and Ba, 2015) as the optimizer. All the aforementioned hyper-parameters are initialized randomly, and we apply dropout rate 0.2 to all hidden layers in the model. Our model treats a candidate as the correct reference if its predicted overall score $F(n, p)$ is larger than 0. The model training is performed with up to 100 epochs, and the best one is selected based on its performance on the development set.

5 Experimental Results

Table 2 compares the performance of our model with all baselines. Overall, our model performs the best with respect to all evaluation metrics. Several findings are also observed from the results. First, manually defined knowledge and features are not enough to cover rich contextual information. Deep learning models (e.g., End2end and our proposed models), which leverage text representations for context, outperform other approaches by a great margin, especially on the recall. Second, external knowledge is highly helpful in this task, which is supported by that our model outperforms the End2end model significantly.

Moreover, the comparison between the two variants of our models is also interesting, where the final two-layer model outperforms the Feature Concatenation model. It proves that simply treating external knowledge as the feature, even though they are from the same sources, is not as effective as learning them in a joint framework. The reason

---

[12] https://stanfordnlp.github.io/CoreNLP/coref.html

[13] https://github.com/kentonl/e2e-coref

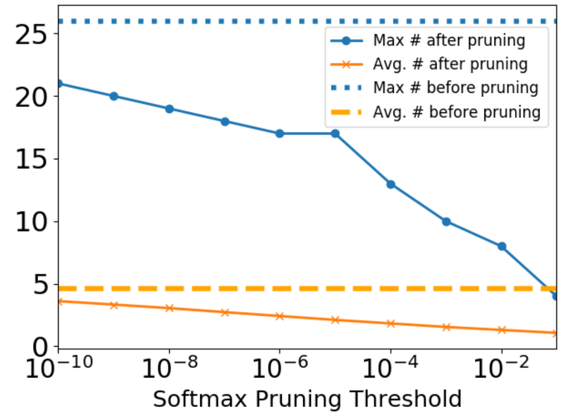


Figure 5: Effect of different thresholds on candidate numbers. Max and Average number of candidates after pruning are represented with solid lines in blue and orange, respectively. Two dashed lines indicate the max and the average number of candidates before pruning.

	F1	ΔF1
The Complete Model	81.0	-
–Plurality knowledge	80.7	-0.3
–AG knowledge	80.5	-0.5
–SP knowledge	80.4	-0.6
–Knowledge Attention	80.1	-0.9

Table 3: Performance of our model with removing different knowledge sources and knowledge attention.

behind this result is mainly from the noise in the knowledge source, e.g., parsing error, incorrectly identified relations, etc. For example, the plurality of 17% noun phrases are wrongly labeled in the test data. As a comparison, our knowledge attention might contribute to alleviate such noise when incorporating all knowledge sources.

Effect of Different Knowledge To illustrate the importance of different knowledge sources and the knowledge attention mechanism, we ablate various components of our model and report the corresponding F1 scores on the test data. The results are shown in Table 3, which clearly show the necessity of the knowledge. Interestingly, AG contributes the most among all knowledge types, which indicates that potentially more cases in the evaluation dataset demand on the AG knowledge than others. More importantly, the results also prove the effectiveness of the knowledge attention module, which contributes to the performance gap between our model and the Feature Concatenation one.

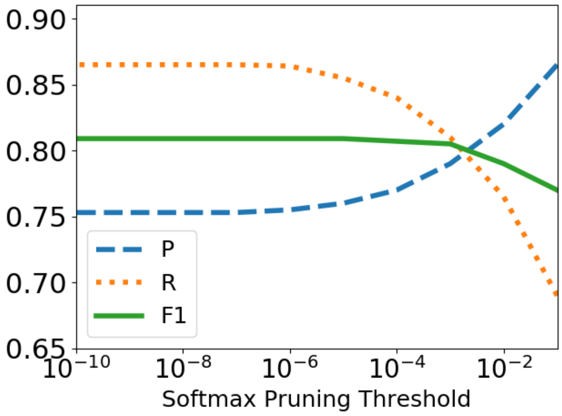


Figure 6: Effect of different pruning thresholds on model performance. With the threshold increasing, the precision increases while the recall and F1 drop.

Effect of Different Pruning Thresholds We try different thresholds t for the softmax pruning in selecting reliable candidates. The effects of different thresholds on reducing candidates and overall performance are shown in Figure 5 and 6 respectively. Along with the increase of t, both the max and the average number of pruned candidates drop quickly, so that the space complexity of the model can be reduced accordingly. Particularly, there are as much as 80% candidates can be filtered out when $t = 10^{-1}$. Meanwhile, when referring to Figure 6, it is observed that the model performs stable with the decreasing of candidate numbers. Not surprisingly, the precision rises when reducing candidate numbers, yet the recall drops dramatically, eventually results in the drop of F1. With the above observations, the reason we set $t = 10^{-7}$ as the default threshold is straightforward: on this value, one-third candidates are pruned with almost no influence on the model performance in terms of precision, recall, and the F1 score.

6 Case Study

To further demonstrate the effectiveness of incorporating knowledge into pronoun coreference resolution, two examples are provided for detailed analysis. The prediction results of the End2end model and our complete model are shown in Table 4. There are different challenges in both examples. In Example A, 'Jesus', 'man', and 'my son' are all similar (male) noun phrases matching the target pronoun '*He*'. The End2end model predicts all of them to be correct references because their context provides limited help in dis-

	Example A	Example B
Sentences	... (A large group of people) met (Jesus). (A man in the group) shouted to him: "(Teacher), please come and look at (my son). **He** is the only child I have" ...	... (My neighbor) told me that there was (an accident), and everyone else was intact, except (his father), who was in (hospital) for fractures. I comforted him first and asked (my friend) to rush me to (the hospital). (My neighbor) showed me the police report at (the hospital), which indicated **it** was all my neighbor's fault. ...
Pronoun	**He**	**it**
Candidate NPs	A large group of people, Jesus, A man in the group, Teacher, my son.	My friend, an accident, his father, hospital, my friend, the hospital, My neighbor, the hospital.
End2end	Jesus, A man in the group, my son	None
Our Model	my son	an accident

Table 4: The comparison of End2end and our model on two examples drawn from the test data. Pronouns are marked as blue bold font. Correct references are indicated in green underline font and other candidates are indicated with brackets. 'None' refers to that none of the candidates is predicated as the correct reference.

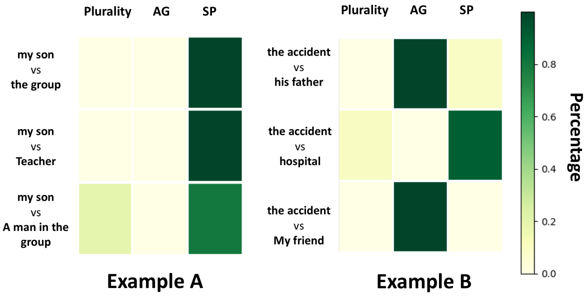


Figure 7: Heatmaps of knowledge attention for two examples, where in each example the knowledge attention weights of the correct references against other candidates are illustrated. Darker color refers to higher weight on the corresponding knowledge type.

tinguishing them. In Example B, the distance between 'an accident' and the pronoun 'it' is too far. As a result, the 'None' result from the End2end model indicates that the contextual information is not enough to make the decision. As a comparison, in our model, integrating external knowledge can help to solve such challenges, e.g., for Example A, SP knowledge helps when Plurality and AG cannot distinguish all candidates.

To clearly illustrate how our model leverages the external knowledge, we visualize the knowledge attention of the correct reference against other candidates[14] via heatmaps in Figure 7. Two interesting observations are drawn from the visualization. First, given two candidates, if they are significantly different in one feature, our model tends to pay more attention to that feature. Take AG as an example, in Example A, the AG features of all candidates consistently match the pronoun 'he' (all male/neutral). Thus the comparison between 'my son' and all candidates pay no attention to the AG feature. While in Example B, the target pronoun 'it' cannot describe human, thus 'father' and 'friend' are 0 on the AG feature while 'hospital' and 'accident' are 1. As a result, the attention module emphasizes AG more than other knowledge types. Second, The importance of SP is clearly shown in these examples. In example A, Plurality and AG features cannot help, the attention module weights higher on SP because 'son' appears 100 times as the argument of the parsed predicate 'child' in the SP knowledge base, while other candidates appear much less at that position. In example B, as mentioned above, once AG helps filtering 'hospital' and 'accident', SP plays an important role in distinguishing them because 'accident' appears 26 times in the SP knowledge base as the argument of the 'fault' from the results of the parser, while 'hospital' never appears at that position.

7 Related Work

Coreference resolution is a core task for natural language understanding, where it detects mention span and identifies coreference relations among them. As demonstrated in (Lee et al., 2017), mention detection and coreference prediction are the two major focuses of the task. Different from the general coreference task, pronoun coreference resolution has its unique challenge since the semantics of pronouns are often not as clear as normal noun phrases, in general, how to leverage the context and external knowledge to resolve the coreference for pronouns becomes its focus (Hobbs,

---

[14] Only candidates entered the second layer are considered.

1978; Rahman and Ng, 2011; Emami et al., 2018).

In previous work, external knowledge including manually defined rules (Hobbs, 1978; Ng, 2005), such as number/gender requirement of different pronouns, and world knowledge (Rahman and Ng, 2011), such as selectional preference (Wilks, 1975; Zhang and Song, 2018), have been proved to be helpful for pronoun coreference resolution. Recently, with the development of deep learning, Lee et al. (2017) proposed an end-to-end model that learns contextual information with an LSTM module and proved that such knowledge is helpful for coreference resolution when the context is properly encoded. The aforementioned two types of knowledge have their own advantages: the contextual information covers diverse text expressions that are difficult to be predefined while the external knowledge is usually more precisely constructed and able to provide extra information beyond the training data. Different from previous work, we explore the possibility of joining the two types of knowledge for pronoun coreference resolution rather than use only one of them. To the best of our knowledge, this is the first attempt that uses deep learning model to incorporate contextual information and external knowledge for pronoun coreference resolution.

8 Conclusion

In this paper, we proposed a two-layer model for pronoun coreference resolution, where the first layer encodes contextual information and the second layer leverages external knowledge. Particularly, a knowledge attention mechanism is proposed to selectively leverage features from different knowledge sources. As an enhancement to existing methods, the proposed model combines the advantage of conventional feature-based models and deep learning models, so that context and external knowledge can be synchronously and effectively used for this task. Experimental results and case studies demonstrate the superiority of the proposed model to state-of-the-art baselines. Since the proposed model adopted an extensible structure, one possible future work is to explore the best way to enhance it with more complicated knowledge resources such as knowledge graphs.

Acknowledgements

This paper was partially supported by the Early Career Scheme (ECS, No.26206717) from Research Grants Council in Hong Kong. In addition, Hongming Zhang has been supported by the Hong Kong Ph.D. Fellowship and the Tencent Rhino-Bird Elite Training Program. We also thank the anonymous reviewers for their valuable comments and suggestions that help improving the quality of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Eugene Charniak and Micha Elsner. 2009. Em works for pronoun anaphora resolution. In *Proceedings of EACL 2009*, pages 148–156.

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of ACL-IJCNLP 2015*, volume 1, pages 1405–1415.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of EMNLP 2016*, pages 2256–2262.

Richard J Edens, Helen L Gaylard, Gareth JF Jones, and Adenike M Lam-Adesina. 2003. An investigation of broad coverage automatic pronoun resolution for information retrieval. In *Proceedings of SIGIR 2003*, pages 381–382. ACM.

Kate Ehrlich. 1981. Search and inference strategies in pronoun resolution: An experimental study. In *Proceedings of ACL 1981*, pages 89–93.

Ali Emami, Paul Trichelair, Adam Trischler, Kaheer Suleman, Hannes Schulz, and Jackie Chi Kit Cheung. 2018. The hard-core coreference corpus: Removing gender and number cues for difficult pronominal anaphora resolution. *arXiv preprint arXiv:1811.01747*.

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP 2017*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL-HLT 2018*, pages 687–692.

Dingcheng Li, Tim Miller, and William Schuler. 2011. A pronoun anaphora resolution system based on factorial hidden markov models. In *Proceedings of ACL 2011*, pages 1169–1178.

Ruslan Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proceedings of ACL 1998*, pages 869–875.

Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.

Ruslan Mitkov et al. 1995. Anaphora resolution in machine translation. In *Proceedings of the Sixth International conference on Theoretical and Methodological issues in Machine Translation*. Citeseer.

Testuya Nasukawa. 1994. Robust method of pronoun resolution using full-text information. In *Proceedings of CCL 1994*, pages 1157–1163.

Vincent Ng. 2005. Supervised ranking for pronoun resolution: Some recent improvements. In *Proceedings of EMNLP 2005*, volume 20, page 1081.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP 2012*, pages 1–40.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of EMNLP 2010*, pages 492–501.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of EMNLP 2009*, pages 968–977.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of ACL 2011*, pages 814–824.

Yan Song, Chia-Jung Lee, and Fei Xia. 2017. Learning word representations with regularization from prior knowledge. In *Proceedings of CoNLL 2017*, pages 143–152.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of NAACL-HLT 2018*, pages 175–180.

Josef Steinberger, Massimo Poesio, Mijail A Kabadjov, and Karel Jevzek. 2007. Two uses of anaphora resolution in summarization. *Information Processing & Management*, 43(6):1663–1680.

Michael Strube and Christoph Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of ACL 2003*, pages 168–175.

Yorick Wilks. 1975. A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74.

Hongming Zhang and Yangqiu Song. 2018. A distributed solution for winograd schema challenge. In *Proceedings of ICMLC 2018*, pages 322–326.

Yingyi Zhang, Jing Li, Yan Song, and Chengzhi Zhang. 2018. Encoding conversation context for neural keyphrase extraction from microblog posts. In *Proceedings of NAACL-HLT 2018*, pages 1676–1686.

Unsupervised Deep Structured Semantic Models for Commonsense Reasoning

Shuohang Wang[1]*, **Sheng Zhang**[2], **Yelong Shen**[4], **Xiaodong Liu**[3],
Jingjing Liu[3], **Jianfeng Gao**[3], **Jing Jiang**[1]
[1]Singapore Management University,[2]Johns Hopkins University, [3]Microsoft, [4]Tencent AI Lab
`{shwang.2014,jingjiang}@smu.edu.sg, zsheng2@jhu.edu`
`{xiaodl,jingjl,jfgao}@microsoft.com, yelongshen@tencent.com`

Abstract

Commonsense reasoning is fundamental to natural language understanding. While traditional methods rely heavily on human-crafted features and knowledge bases, we explore learning commonsense knowledge from a large amount of raw text via unsupervised learning. We propose two neural network models based on the Deep Structured Semantic Models (DSSM) framework to tackle two classic commonsense reasoning tasks, Winograd Schema challenges (WSC) and Pronoun Disambiguation (PDP). Evaluation shows that the proposed models effectively capture contextual information in the sentence and coreference information between pronouns and nouns, and achieve significant improvement over previous state-of-the-art approaches.

1 Introduction

Commonsense reasoning is concerned with simulating the human ability to make presumptions about the type and essence of ordinary situations they encounter every day (Davis and Marcus, 2015). It is one of the key challenges in natural language understanding, and has drawn increasing attention in recent years (Levesque et al., 2011; Roemmele et al., 2011; Zhang et al., 2017; Rashkin et al., 2018a,b; Zellers et al., 2018; Trinh and Le, 2018). However, due to the lack of labeled training data or comprehensive hand-crafted knowledge bases, commonsense reasoning tasks such as Winograd Schema Challenge (Levesque et al., 2011) are still far from being solved.

In this work, we propose two effective unsupervised models for commonsense reasoning, and evaluate them on two classic commonsense reasoning tasks: Winograd Schema Challenge (WSC) and Pronoun Disambiguation Problems (PDP). Compared to other commonsense reasoning tasks, WSC and PDP better approximate real human reasoning, and can be more easily solved by native English-speaking adults (Levesque et al., 2011). In addition, they are also technically challenging. For example, the best reported result on WSC is only 20 percentage points better than random guess in accuracy (Radford et al., 2019).

1. *The city councilmen refused the demonstrators a permit because **they** feared violence.*

   Who feared violence?

   A. **The city councilmen** B. The demonstrators

2. *The city councilmen refused the demonstrators a permit because **they** advocated violence.*

   Who advocated violence?

   A. The city councilmen B. **The demonstrators**

Table 1: Examples from Winograd Schema Challenge (WSC). The task is to identify the reference of the pronoun in bold.

Table 1 shows two examples from WSC. In order to resolve the co-reference in these two examples, one cannot predict what "***they***" refers to unless she is equipped with the commonsense knowledge that "*demonstrators usually cause violence and city councilmen usually fear violence*".

As no labeled training data is available for these tasks, previous approaches are based on either hand-crafted knowledge bases or large-scale language models. For example, Liu et al. (2017) used existing knowledge bases such as ConceptNet (Liu and Singh, 2004) and WordNet (Miller, 1995) for external supervision to train word embeddings and solve the WSC challenge. Recently, Trinh and Le (2018) first used raw text from books/news to train a neural Language Model (LM), and then em-

---

[*] Work done when the author was at Microsoft

Proceedings of NAACL-HLT 2019, pages 882–891
Minneapolis, Minnesota, June 2 - June 7, 2019.

ployed the trained model to compare the probabilities of the sequences, where the pronouns are replaced by each of the candidate references, and to pick the candidate that leads to the highest probability as the answer.

Because none of the existing hand-crafted knowledge bases is comprehensive enough to cover all the world knowledge[1], we focus on building unsupervised models that can learn commonsense knowledge directly from unlimited raw text. Different from the neural language models, our models are optimized for co-reference resolution and achieve much better results on both the PDP and WSC tasks.

In this work we formulate the two commonsense reasoning tasks in WSC and PDP as a pairwise ranking problem. As the first example in Table 1, we want to develop a pair-wise scoring model $\text{Score}_\theta(x_i, y)$ that scores the correct antecedent-pronoun pair ("*councilmen*", "*they*") higher than the incorrect one ("*demonstrators*", "*they*"). These scores depend to a large degree upon the contextual information of the pronoun (e.g., they) and the candidate antecedent (e.g., councilmen). In other words, it requires to capture the semantic meaning of the pronoun and the candidate antecedent based on the sentences where they occur, respectively.

To tackle this issue, we propose two models based on the framework of Deep Structured Similarity Model (DSSM) (Huang et al., 2013), as shown in Figure 1(a). Formally, let S^x be the sentence containing the candidate antecedent x_i and S^y the sentence containing the pronoun y which we're interested in. DSSM measures the semantic similarity of a pair of inputs (x_i, y) by 1) mapping x_i and y, together with their context information, into two vectors in a semantic space using deep neural networks f_1 and f_2, parameterized by θ; and 2) computing cosine similarity[2] between them. In our case, we need to learn a task-specific semantic space where the distance between two vectors measures how likely they co-refer. Commonsense knowledge such as "demonstrators usually cause violence" can be implicitly captured in the semantic space through DSSM, which is trained on a large amount of raw text.

DSSM requires labeled pairs for training.Since there is no labeled data for our tasks, we propose two unsupervised DSSMs, or UDSSMs. As shown in Figure 1(b) and 1(c), $(\mathbf{S}^x, \mathbf{S}^y)$ are encoded into contextual representations by deep neural networks f_1 and f_2; then we compute pair-wise their co-reference scores.

In what follows, we will describe two assumptions we propose to harvest training data from raw text. **Assumption I: A pronoun refers to one of its preceding nouns in the same sentence.** The sentences generated by this assumption will be used for training UDSSM-I. Some examples will be shown in the "data generation" section. **Assumption II: In a sentence, pronouns of the same gender and plurality are more likely to refer to the same antecedent than other pronouns.** Similarly, the sentences following the assumption will be used for training UDSSM-II.

Note that the two models, UDSSM-I and UDSSM-II are trained on different types of pairwise training data, thus the model structures are different, as illustrated in Figure 1(b) and 1(c), respectively. Experiments demonstrated that our methods outperform stat-of-the-art performance on the tasks of WSC and PDP.

2 Related Work

As a key component of natural language understanding, commonsense reasoning has been included in an increasing number of tasks for evaluation: COPA (Roemmele et al., 2011) assesses commonsense causal reasoning by selecting an alternative, which has a more plausible causal relation with the given premise. Story Cloze Test (ROCStories, Mostafazadeh et al. 2016) evaluates story understanding, story generation, and script learning by choosing the most sensible ending to a short story. JOCI (Zhang et al., 2017) generalizes the natural language inference (NLI) framework (Cooper et al., 1996; Dagan et al., 2006; Bowman et al., 2015; Williams et al., 2018) and evaluates commonsense inference by predicting the ordinal likelihood of a hypothesis given a context. Event2Mind (Rashkin et al., 2018b) models stereotypical intents and reactions of people, described in short free-form text. SWAG (Zellers et al., 2018) frames commonsense inference as multiple-choice questions for follow-up events given some context. ReCoRD (Zhang et al., 2018)

---

[1] We don't believe it is possible to construct such a knowledge base given that the world is changing constantly.

[2] DSSMs can be applied to a wide range of tasks depending on the definition of (x, y). For example, (x, y) is a query-document pair for Web search ranking, a document pair in recommendation, a question-answer pair in QA, and so on. See Chapter 2 of (Gao et al., 2018) for a survey.

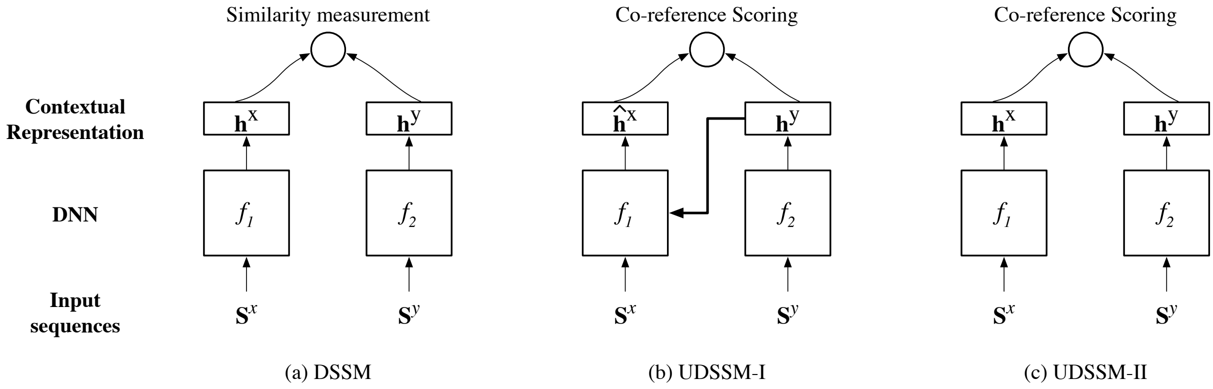


Figure 1: An overview of (a) the general framework of Deep Structured Semantic Model (DSSM) and our two unsupervised models based on DSSM: (b) UDSSM-I and (c) UDSSM-II. Compared with DSSM, both UDSSM-I and UDSSM-II compute co-reference scores instead of similarity.

evaluates a machine's ability of commonsense reasoning in reading comprehension.

Among all these commonsense reasoning tasks, the Winograd Schema Challenge (WSC) and Pronoun Disambiguation Problems (PDP) (Levesque et al., 2011) are known as the most challenging tasks for commonsense reasoning. Although both tasks are based on pronoun disambiguation, a subtask of coreference resolution (Soon et al., 2001; Ng and Cardie, 2002; Peng et al., 2016), PDP and WSC differ from normal pronoun disambiguation due to their unique properties, which are based on commonsense, selecting the most likely antecedent from both candidates in the directly preceding context.

Previous efforts on solving the Winograd Schema Challenge and Pronoun Disambiguation Problems mostly rely on human-labeled data, sophisticated rules, hand-crafted features, or external knowledge bases (Peng et al., 2015; Bailey et al., 2015; Schüller, 2014). Rahman and Ng (2012) hired workers to annotate supervised training data and designed 70K hand-crafted features. Sharma et al. (2015); Schüller (2014); Bailey et al. (2015); Liu et al. (2017) utilized expensive knowledge bases in their reasoning processes. Recently, Trinh and Le 2018 applied neural language models trained with a massive amount of unlabeled data to the Winograd Schema Challenge and improved the performance by a large margin. In contrast, our unsupervised method based on DSSM significantly outperforms the previous state-of-the-art method, with the advantage of capturing more contextual information in the data.

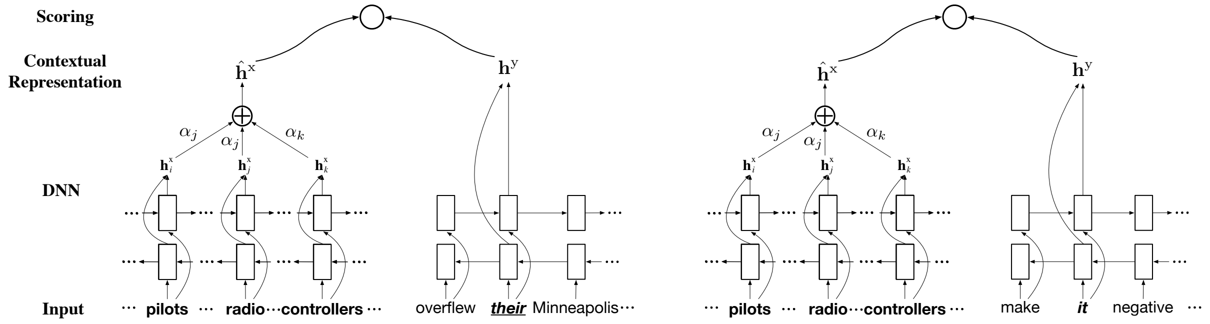
3 Approach

As shown in Figure 1, we propose two unsupervised deep structured semantic models (**UDSSM-I** and **UDSSM-II**), which consist of two components: DNN encoding and co-reference scoring. For the model UDSSM-I, the co-referred word pairs are automatically learned through an attention mechanism, where the attention weights are the co-reference scores for word pairs. For the second model UDSSM-II, we will directly optimize the co-reference score during training. After all, we will get the co-reference scoring function, $\text{Score}_\theta(x_i, y)$, to compare the candidate answers in the tasks of PDP/WSC. Next, we will show the details of our models trained in an unsupervised way.

In the following sections, we will use uppercase symbols in bold, e.g., $\mathbf{S}^\text{x}$, to represent matrices. Lowercase symbols in bold, e.g., $\mathbf{h}^\text{x}$, represent vectors. A regular uppercase symbol, e.g., S^x, represents a lexical sequence. A regular lowercase symbol, e.g., x_i or y, represents a token.

3.1 UDSSM-I Model

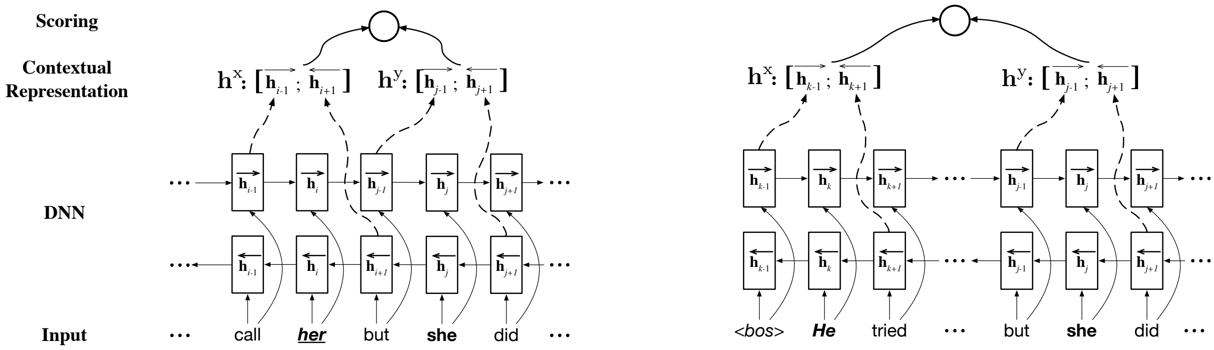
This model is developed based on Assumption I. Its architecture is shown in Figure 2. The sentences generated based on this assumption contain a pronoun y and a set of its preceding nouns $\{x_i, x_j ...\}$, which includes the referred word by pronoun. For example, the sentence in Figure 2. As there is no clear label for the co-referred word pairs under this assumption, our model will rank the set of nouns $\{x_i, x_j ...\}$ which contains the noun that the pronoun y refers to higher than the set which does not. And the co-reference score between words will not be optimized directly during



(a) Positive example

(b) Negative example

Figure 2: The procedure of using UDSSM-I to compute the co-reference scores of a positive example and a negative example respectively. The positive example is generated from the sentence '*Two **Northwest Airlines pilots** failed to make **radio contact** with **ground controllers** for more than an hour and overflew **their** Minneapolis destination by 150 miles before discovering the mistake and turning around.*". The negative one replaces the second sequence with one sequence from different sentence.



(a) Positive example

(b) Negative example

Figure 3: The procedure of using UDSSM-II to compute the co-reference scores of a positive example and a negative example respectively. Both examples are generated from the sentence "*He tried twice to call her but she did not answer the phone*".

training, but is learned indirectly through the attention mechanism. We will describe in turn how the training data is generated from raw text, the model architecture, and the co-reference scoring function for the final prediction on the tasks of PDP/WSC.

3.1.1 Data Generation

The main challenge of PDP/WSC tasks is that it has no labeled training data. Here we introduce a simple method to collect unsupervised training data by leveraging some linguistic patterns. Following Assumption 1, the first hypothesis we make is that "the pronoun refers to one of the preceding nouns", which is a common phenomenon in well-written stories or news. In this way, we generate (S^{x}, S^{y}) pairs from raw text as follows:

- Parse the sentences in the raw text to obtain entity names, nouns and pronouns.
- Pick sentences that contain at least one pronoun and multiple nouns preceding it.
- Split each sentence into two sub-sentences to form a positive pair (S^{x}, S^{y}), where S^{x} is the first sub-sentence with identified nouns and entity names while S^{y} is the second sub-sentence with a pronoun.
- One or more negative pairs are generated from (S^{x}, S^{y}) by replacing S^{y} with $S^{\mathrm{y}_{neg}}$ randomly sampled from other positive pairs.

We split the sentence with pronouns and nouns into two sub-sequences separated by the previous word of the pronoun. Therefore, the example sentence in the Figure 2 can be split into two sub-sentences as shown below:

- S^{x}: " **Two Northwest Airlines pilots** failed to make **radio contact** with **ground controllers** for more than an hour and"

- S^{y}: "overflew **their** Minneapolis destination by 150 miles before discovering the mistake and turning around".

As the sentences are collected from raw text, the co-reference words are not given. Our proposed UDSSM-I model will learn the co-reference scoring function through attention mechanism based on the generated sequence pairs. Next, we will introduce the details of this model.

3.1.2 Model Architecture

This method takes the pair of sequences, (S^{x}, S^{y}), as inputs, and computes similarity between the sequences collected from the same sentence. As we hypothesize that one of the nouns in the first sequence and the pronoun in the second are co-referred, we only use the contextual representations of nouns and pronoun to represent the sequences. To obtain the contextual representation, we first use a bi-directional LSTM to process these sequences [3]:

$$\mathbf{H}^{\mathrm{x}} = \text{Bi-LSTM}(\mathbf{S}^{\mathrm{x}}), \mathbf{H}^{\mathrm{y}} = \text{Bi-LSTM}(\mathbf{S}^{\mathrm{y}}), \tag{1}$$

where $\mathbf{S}^{\mathrm{x}} \in \mathbb{R}^{d \times X}$, $\mathbf{S}^{\mathrm{y}} \in \mathbb{R}^{d \times Y}$ are the word embeddings of the two sequences. d is the dimension of the word embeddings. X, Y are the lengths of the two sequences. $\mathbf{H}^{\mathrm{x}} \in \mathbb{R}^{l \times X}$ and $\mathbf{H}^{\mathrm{y}} \in \mathbb{R}^{l \times Y}$ are the hidden states of bi-directional LSTM. Our model is task-specifically constructed, so we directly use the hidden state of the first pronoun in the second sequence as its representation:

$$f_2(S^y) = \mathbf{h}^{\mathrm{y}} = \mathbf{h}^{\mathrm{y}}_2, \tag{2}$$

where $\mathbf{h}^{\mathrm{y}}_2 \in \mathbb{R}^{\mathrm{l}}$ is the second[4] vector from $\mathbf{H}^{\mathrm{y}}$ and it represents the contextual information of the pronoun. Next, we will get the representation of the first sequence. As there are multiple nouns in the first sequence and the pronoun usually refers to only one of them, we use the weighted sum of all the LSTM hidden states of the nouns to represent the sequence, $\hat{\mathbf{h}}^{\mathrm{x}} \in \mathbf{R}^{l}$, as follows:

$$\begin{aligned}
\mathbf{H}^{\mathrm{n}} &= [\mathbf{h}^{\mathrm{x}}_i; \mathbf{h}^{\mathrm{x}}_j; ...] \\
\alpha &= \text{SoftMax}\left((\mathbf{W}^{\mathrm{g}}\mathbf{H}^{\mathrm{n}} + \mathbf{b}^{\mathrm{g}} \otimes \mathbf{e}_N)^{\mathrm{T}}\mathbf{h}^{\mathrm{y}}\right), \\
f_1(S^x) &= \hat{\mathbf{h}}^{\mathrm{x}} = \mathbf{H}^{\mathrm{n}}\alpha,
\end{aligned} \tag{3}$$

where $i, j...$ are the positions of the nouns in the sequence S^x and $[;]$ is the concatenation of two vectors. $\mathbf{H}^{\mathrm{n}} \in \mathbb{R}^{l \times N}$ are all the hidden states of the nouns[5] in $\mathbf{H}^{\mathrm{x}}$ in the sequence. N is the number of nouns in the sequence. $\alpha \in \mathbb{R}^{N}$ is the weights assigned for the different nouns and $\hat{\mathbf{h}}^{\mathrm{x}} \in \mathbb{R}^{l}$ is the weighted sum of all the hidden states of the nouns. $\mathbf{W}^{\mathrm{g}} \in \mathbb{R}^{l \times l}$ and $\mathbf{b}^{\mathrm{g}} \in \mathbb{R}^{l}$ are the parameters to learn; $\mathbf{e}_N \in \mathbb{R}^{\mathrm{N}}$ is a vector of all 1s and it is used to repeat the bias vector N times into the matrix. Then we will maximize the similarity of the contextual representations of ($\hat{\mathbf{h}}^{\mathrm{x}}, \mathbf{h}^{\mathrm{y}}$). Meanwhile, we also need some negative samples $\mathbf{h}^{\mathrm{y}_{neg}}_k$ for $\hat{\mathbf{h}}^{\mathrm{x}}$. Then our loss function for this method is constructed:

$$L = -\log\left(\frac{\exp\left(\hat{\mathbf{h}}^{\mathrm{x}}\mathbf{h}^{\mathrm{y}}\right)}{\exp\left(\hat{\mathbf{h}}^{\mathrm{x}}\mathbf{h}^{\mathrm{y}}\right) + \sum_k^K \exp\left(\hat{\mathbf{h}}^{\mathrm{x}}\mathbf{h}^{\mathrm{y}_{neg}}_k\right)}\right), \tag{4}$$

where $\mathbf{h}^{\mathrm{y}_{neg}}_k \in \mathbb{R}^{l}$ is the randomly sampled hidden state of pronoun from the sequences not in the same sentence with S^y.

3.1.3 Co-reference Scoring Function

Overall, the model tries to make the co-reference states similar to each other. The co-reference scoring function is defined:

$$\text{Score}_\theta(x_i, y) = g(\mathbf{h}^{\mathrm{x}}_i, \mathbf{h}^{\mathrm{y}}) = (\mathbf{W}^{\mathrm{g}}\mathbf{h}^{\mathrm{x}}_i + \mathbf{b}^{\mathrm{g}})^{\mathrm{T}}\mathbf{h}^{\mathrm{y}}, \tag{5}$$

where the candidate located at the i-th position is represented by its LSTM hidden state $\mathbf{h}^{\mathrm{x}}_i$ and the pronoun in the snippet is represented by $\mathbf{h}^{\mathrm{y}}$. And the output value of this function for each candidate will be used for the final prediction. Next, we will introduce the other unsupervised method.

3.2 UDSSM-II Model

This model is developed based on Assumption II. Its architecture is shown in Figure 3. As the model is similar to the previous one, we will introduce the details in a similar way.

3.2.1 Data Generation

The second assumption is that "the pronoun pairs in a single sentence are co-reference words if they are of the same gender and plurality; otherwise they are not." Based on this assumption, we can directly construct the co-reference training pairs as follows:

---

[3] We use two different LSTMs to process the sequences S^{x} and S^{Y} here. This is to make the negative sampling in Eqn. (4) more efficient, so that we can directly use the other representations in the same batch as negative ones.

[4] We assign the word just before the pronoun to the second sequence, so the pronoun always appears in the second position of the sequence.

[5] We use the toolkit of spaCy in Python for POS and NER, and we will remove the sequences that contain less than 2 nouns.

- Parse the raw sentences to identify pronouns.
- Pick sentences that contain at least two pronouns.
- The sub-sequence pair with pronouns of the same gender and plurality is labeled as a positive pair; otherwise it is labeled as negative.
- Replace the corresponding pronoun pairs with a special token "**@Ponoun**".

Take the following sentence as an example: "**He** tried twice to call **her** but **she** did not answer the phone." There are three pronouns detected in the sentence, and we assume that the words **her** and **she** are co-reference words, while pairs (**she**, **He**) and (**her**, **He**) are not. Thus we can obtain three training examples from the given sentence. However, in the PDP and WSC tasks, models are asked to compute the co-reference scores between pronoun and candidate nouns, instead of two pronouns. Therefore, we replace the first pronoun in the sentence with a place holder; i.e., a negative training pair is generated by splitting the raw sentence into the following two sub-sequences:

- S^{x}: " **@Ponoun** tried twice to call her"
- S^{y}: "but **she** did not answer the phone."
- label: Negative

and the positive training pair can be generated by the same way:

- S^{x}: " He tried twice to call **@Ponoun**"
- S^{y}: "but **she** did not answer the phone."
- label: Positive

Thus, we could directly train the encoder and co-reference scoring components through the generated training pairs.

3.2.2 Model Architecture

The previous method, UDSSM-I, follows the task setting of PDP/WSC, and builds the model based on the similarity of the representations between nouns and the pronoun. As there is no signal indicating the exact alignment between co-reference words, the model tries to learn it based on the co-occurrence information from large scale unlabelled corpus. For the method of UDSSM-II, each representation pair $(\mathbf{h}^{\text{x}}, \mathbf{h}^{\text{y}})$ has a clear signal, r, indicating whether they are co-referred or not. For simplicity, we do not have to split the sentence into two parts. We first use LSTM to process the sentence as follows:

$$\overrightarrow{\mathbf{H}} = \overrightarrow{\text{LSTM}}([\mathbf{S}^{\text{x}}; \mathbf{S}^{\text{y}}]), \overleftarrow{\mathbf{H}} = \overleftarrow{\text{LSTM}}([\mathbf{S}^{\text{x}}; \mathbf{S}^{\text{y}}]), \quad (6)$$

where we can concatenate the word embeddings, $[\mathbf{S}^{\text{x}}; \mathbf{S}^{\text{y}}]$, of two sequences collected under Assumption II. $\overrightarrow{\text{LSTM}}$ and $\overleftarrow{\text{LSTM}}$ are built in different directions, and $\overrightarrow{\mathbf{H}}$, $\overleftarrow{\mathbf{H}}$ are the hidden states of the corresponding LSTM. Suppose that the pronoun pair in the sentence are located at the i-th and j-th positions as shown in the bottom part of Figure 3(a). We use the hidden states around the pronouns as their contextual representations as follows:

$$f_1(S^x) = \mathbf{h}^{\text{x}} = \begin{bmatrix} \overrightarrow{\mathbf{h}_{i-1}} \\ \overleftarrow{\mathbf{h}_{i+1}} \end{bmatrix}, f_2(S^y) = \mathbf{h}^{\text{y}} = \begin{bmatrix} \overrightarrow{\mathbf{h}_{j-1}} \\ \overleftarrow{\mathbf{h}_{j+1}} \end{bmatrix}, \quad (7)$$

where $\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$ is the concatenation of all the vectors inside it. Then we further concatenate these representation pair:

$$\mathbf{h}^c = \begin{bmatrix} \mathbf{h}^{\text{x}} \\ \mathbf{h}^{\text{y}} \end{bmatrix}, \quad (8)$$

where $\mathbf{h}^c \in \mathbb{R}^{4l}$, and it will be the input of loss function with cross entropy as follows:

$$\begin{aligned} L &= -r \log\left(\frac{\exp(\mathbf{w}^p \mathbf{h}^c)}{\exp(\mathbf{w}^p \mathbf{h}^c) + \exp(\mathbf{w}^n \mathbf{h}^c)}\right) \\ &- (1-r) \log\left(\frac{\exp(\mathbf{w}^n \mathbf{h}^c)}{\exp(\mathbf{w}^p \mathbf{h}^c) + \exp(\mathbf{w}^n \mathbf{h}^c)}\right), \end{aligned}$$

where $r \in \{0, 1\}$ indicates whether the pronouns at the m-th and n-th positions should be considered co-reference or not. $\mathbf{w}^p \in \mathbb{R}^{4l}$ and $\mathbf{w}^n \in \mathbb{R}^{4l}$ are the parameters to learn.

3.2.3 Co-reference Scoring Function

Similar to the Eqn.(5), for each candidate, we use co-reference scoring function $\text{Score}_\theta(x_i, y)$ for the answer selection:

$$\text{Score}_\theta(x_i, y) = g(\mathbf{h}_i^{\text{x}}, \mathbf{h}^{\text{y}}) = \mathbf{w}^p \begin{bmatrix} \overrightarrow{\mathbf{h}_{i-1}} \\ \overleftarrow{\mathbf{h}_{i+1}} \\ \overrightarrow{\mathbf{h}_{j-1}} \\ \overleftarrow{\mathbf{h}_{j+1}} \end{bmatrix}, \quad (9)$$

where i is the position of the candidate in the sentence and j is the position of the pronoun.

	PDP	WSC
Co-reference Resolution Tool	41.7%	50.5
Patric Dhondt (WS Challenge 2016)	45.0%	-
Nicos Issak (WS Challenge 2016)	48.3%	-
Quan Liu (WS Challenge 2016 - winner)	58.3%	-
Unsupervised Semantic Similarity Method (USSM)	48.3%	-
Neural Knowledge Activated Method (NKAM)	51.7%	-
USSM + Cause-Effect Knowledge Base	55.0%	52.0%
USSM + Cause-Effect + WordNet + ConceptNet Knowledge Bases	56.7%	52.8%
USSM + NKAM	53.3%	
USSM + NKAM + 3 Knowledge Bases	66.7%	52.8%
ELMo	56.7%	51.5%
Google Language Model (Trinh and Le, 2018)	60.0%	56.4%
UDSSM-I	75.0%	54.5%
UDSSM-II	**75.0%**	**59.2%**
Google Language Model (ensemble)	70.0%	61.5%
UDSSM-I (ensemble)	76.7%	57.1%
UDSSM-II (ensemble)	**78.3%**	**62.4%**

Table 2: The experiment results on PDP and WSC datasets. We compare our models to Goolge LM trained on the same corpus [6].

4 Experiments

In this section, we will introduce the datasets to train and evaluate our models for commonsense reasoning, the hyper-parameters of our model, and the analysis of our results.

4.1 Datasets

Training Corpus We make use of the raw text from Gutenberg [7], a corpus offerring over 57,000 free eBooks, and 1 Billion Word [8], a corpus of news, to train our model. We first ignore the sentences that contain less than 10 tokens or longer than 50 tokens. Then, for the model UDSSM-I, we collect all the sentences with the pronoun before which there're at least two nouns. For UDSSM-II, we collect all the sentences with at least 2 pronouns. In total, we collect around 4 million training pairs from each corpus for our proposed method respectively, and we split 5% as validation set.

Evaluation Dataset We evaluate our model on the commonsense reasoning datasets, Pronoun Disambiguation Problems (PDP) [9] and Winograd Schema challenges (WSC) [10], which include 60 and 285 questions respectively. Both of the tasks are constructed for testing commonsense reasoning and all the questions from these challenges are obvious for human beings to solve with commonsense knowledge, but hard for machines to solve with statistical techniques.

4.2 Experimental Setting

We use the same setting for both our models. The hidden state dimension of a single-directional LSTM is set to be 300. We use 300 dimensional GloVe embeddings [11] for initialization. We use Adamax to optimise the model, set learning rate to be 0.002, dropout rate on all layers are tuned from [0, 0.1, 0.2] and the batch size from [30, 50, 100, 200]. For the model UDSSM-I, in one batch, we treat all sequence pairs not from the same sentence as negative cases. And it takes around 30 hours on a single K40 GPU to train our models, which are much faster than training a large LM (Jozefowicz et al., 2016) taking weeks on multiple GPUs.

---

[6] The best models reported in the works of Radford et al. (2019) and Trinh and Le (2018) are trained on a much larger corpus from Common Crawl.

[7] `http://www.gutenberg.org`

[8] `https://github.com/ciprian-chelba/1-billion-word-language-modeling-benchmark`

[9] `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/PDPChallenge2016.xml`

[10] `https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.xml`

[11] `https://github.com/stanfordnlp/GloVe`

4.3 Experimental Results

The experiment results are shown in Table 2. Most of the performance in the top of the Table 2 are the models trained with external knowledge bases, such as Cause-Effect (Liu et al., 2016), WordNet (Miller, 1995), ConceptNet (Liu and Singh, 2004) knowledge bases. Unsupervised Semantic Similarity Method (USSM) (Liu et al., 2017) is based on the skip-gram model (Mikolov et al., 2013) to train word embeddings and the embeddings of all the words connected by knowledge bases are optimized to be closer. Neural Knowledge Activated Method (NKAM) (Liu et al., 2017) trained a binary classification model based on whether the word pairs appear in the knowledge base. One limitation of these methods is that they rely heavily on the external knowledge bases. Another limitation is that they just linearly aggregate the embeddings of the words in the context, and that's hard to integrate the word order information. Instead, our model with LSTM can better represent the contextual information. Besides, our model don't need any external knowledge bases, and achieve a significant improvement on both of the datasets.

We further compare our models with the unsupervised baselines, ELMo (Peters et al., 2018) which selects the candidate based on the cosine similarity of the hidden states of noun and pronoun. Another unsupervised baseline, Google Language Model for commonsense reasoning (Trinh and Le, 2018), which compares the perplexities of the new sentences by replacing the pronoun with candidates. To make a fair comparison to Trinh and Le (2018)'s work, we also train our single model on the corpus of Gutenberg only. We can see that both of our methods get significant improvement on the PDP dataset, and our UDSSM-II can achieve much better performance on the WSC dataset. We also report our ensemble model (nine models with different hyperparameters) trained with both corpus of Gutenberg and 1 Billion Word, and it also achieve better performance than Google Language Model trained with the same corpus.

Finally, we also compare to the pre-trained Coreference Resolution Tool (Clark and Manning, 2016a,b)[12], and we can see that it doesn't adapt to our commonsense reasoning tasks and can't tell the difference between each pair of sentences from WSC. In this way, our model can get much better performance.

4.4 Analysis

WSC 1:	**Paul** tried to call George on the phone, but **he** wasn't successful.
Ours 1:	**He** tried to call 911 using her cell phone but that **he** could n't get the phone to work.
WSC 2:	Paul tried to call **George** on the phone, but **he** was n't available .
Ours 2:	He tried twice to call **her** but **she** did not answer the phone .

Table 3: Comparison of the data from WSC and our training data. Our sentences are retrieved from the UDSSM-II training dataset based on the BM25 value for analysis. The pseudo labels in our training data can help identify the co-references in WSC.

In this subsection, we will conduct further analysis on the reason that our models work, the benefit of our models comparing to a baseline, and the limitation of our proposed models.

We have a further analysis on the pair-wise sentences, which we collected for training, to check how our model can work. We find that some reasoning problems can somehow be converted to the paraphrase problem. For example, in Table 3, we make use of Lucene Index[13] with BM25 to retrieve the similar sentences to the WSC sentences from our training dataset, and make a comparison. We can see that these pairs are somehow paraphrased each other respectively. For the first pair, the contextual representations of "Paul" and "he" in WSC could be similar to the contextual representations of "he" in our training sentence. As these representations are used to compute the co-reference score, the final scores would be similar. The pseudo label "positive" for our first sentence will make the positive probability of the golden co-references "Paul" and "he" in WSC higher. And for the second pair in Table 3, the pseudo label of positive in our second sentence will make the positive probability of the golden co-references "George" and "he" in WSC 2 higher. In this way, these kinds of co-reference patterns from training data can be directly mapped to solve the Winograd Schema Challenges.

---

[12] `https://github.com/huggingface/neuralcoref`

[13] `http://lucene.apache.org/pylucene/`

Here's another example from PDP demonstrating the benefit of our method: "Always before, Larry had helped Dad with his work. But *he* could not help him now, for Dad said that ". Trinh and Le (2018) failed on this one, probably because language models are not good at solving long distance dependence, and tends to predict that "he " refers to "his" in the near context rather the correct answer "Larry". And our model can give the correct prediction.

We further analysis the predictions of our model. We find that some specific commonsense knowledge are still hard to learn, such as the following pairs:

- The trophy doesn't fit into the brown **suitcase** because **it** is too small.
- The **trophy** doesn't fit into the brown suitcase because **it** is too large.

To solve this problem, the model should learn the knowledge to compare the size of the objects. However, all of our models trained with different hyper-parameters select the same candidate as the co-referred word for "it" in both sentences. To solve the problem, broader data need to collect for learning more commonsense knowledge.

5 Conclusion

In conclusion, to overcome the lack of human labeled data, we proposed two unsupervised deep structured semantic models (UDSSM) for commonsense reasoning. We evaluated our models on the commonsense reasoning tasks of Pronoun Disambiguation Problems (PDP) and Winograd Schema Challenge (Levesque et al., 2011), where the questions are quite easy for human to answer, but quite challenging for the machine. Without using any hand-craft knowledge base, our model achieved stat-of-the-art performance on the two tasks.

In the future work, we will use Transformer, which is proved to be more powerful than LSTM, as the encoder of our unsupervised deep structured semantic models, and we will collect a larger corpus from Common Crawl to train our model.

References

Daniel Bailey, Amelia Harrison, Yuliya Lierler, Vladimir Lifschitz, and Julian Michael. 2015. The winograd schema challenge and reasoning about correlation. In *Knowledge Representation; Coreference Resolution; Reasoning*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kevin Clark and Christopher D Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. *Proceedings of the Association for Computational Linguistics*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. *arXiv preprint arXiv:1809.08267*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *AAAI spring symposium: Logical formalizations of commonsense reasoning*.

Hugo Liu and Push Singh. 2004. Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Quan Liu, Hui Jiang, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2017. Combing context and commonsense knowledge through neural networks for solving winograd schema problems. In *AAAI Spring Symposium Series*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the conference on empirical methods in natural language processing*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The winograd schema challenge. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018a. Modeling naive psychology of characters in simple commonsense stories. *arXiv preprint arXiv:1805.06533*.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018b. Event2mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the Association for Computational Linguistics*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Peter Schüller. 2014. Tackling winograd schemas by formalizing relevance theory in knowledge graphs. In *Knowledge Representation and Reasoning Conference*.

Arpit Sharma, Nguyen H. Vo, Somak Aditya, and Chitta Baral. 2015. Towards addressing the winograd schema challenge: Building and using a semantic parser and a knowledge hunting module. In *Proceedings of the International Conference on Artificial Intelligence*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies)*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. *arXiv preprint arXiv:1810.12885*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association for Computational Linguistics*.

Recovering Dropped Pronouns in Chinese Conversations via Modeling Their Referents

Jingxuan Yang, Jianzhuo Tong, Si Li [*]**, Sheng Gao, Jun Guo, Nianwen Xue** [†]
Beijing University of Posts and Telecommunications
[†]Brandeis University
{yjx, tongjianzhuo, lisi, gaosheng, guojun}@bupt.edu.cn
xuen@brandeis.edu

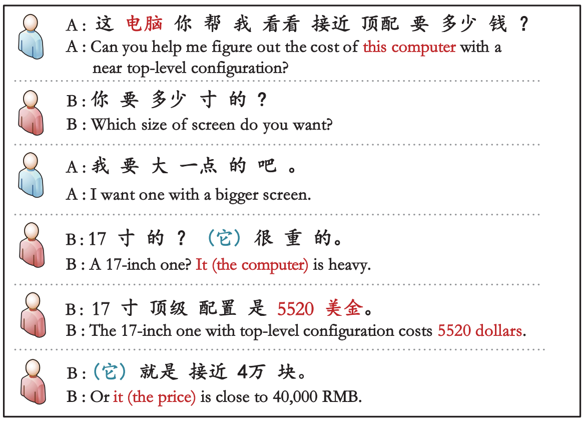
Abstract

Pronouns are often dropped in Chinese sentences, and this happens more frequently in conversational genres as their referents can be easily understood from context. Recovering dropped pronouns is essential to applications such as Information Extraction where the referents of these dropped pronouns need to be resolved, or Machine Translation when Chinese is the source language. In this work, we present a novel end-to-end neural network model to recover dropped pronouns in conversational data. Our model is based on a structured attention mechanism that models the referents of dropped pronouns utilizing both sentence-level and word-level information. Results on three different conversational genres show that our approach achieves a significant improvement over the current state of the art.

1 Introduction

Chinese is a pro-drop language, meaning that it is not always necessary to have an overt pronoun when the referent is clear from the context. This is in contrast with a non-pro-drop language like English, where an overt pronoun is always needed. For example, Kim (2000) shows that an overt subject is used only in 64% of the cases in Chinese while that percentage is over 96% in English.

Even in pro-drop languages like Chinese, pronouns are omitted to different degrees in different genres. Pronouns are dropped more often in informal conversational genres than in formal genres like newswire. Recovering these dropped pronouns (DPs) is important to applications such as Machine Translation where dropped pronouns need to be made explicit when Chinese is translated into a target language (Wang et al., 2016a,b, 2018) or Information Extraction where relations might involve entities referred to by these DPs.



A : 这 电脑 你 帮 我 看看 接近 顶配 要 多少 钱 ?
A : Can you help me figure out the cost of this computer with a near top-level configuration?

B : 你 要 多少 寸 的 ?
B : Which size of screen do you want?

A : 我 要 大 一点 的 吧 。
A : I want one with a bigger screen.

B : 17 寸 的 ? （它） 很 重 的。
B : A 17-inch one? It (the computer) is heavy.

B : 17 寸 顶级 配置 是 5520 美金。
B : The 17-inch one with top-level configuration costs 5520 dollars.

B : （它） 就是 接近 4万 块。
B : Or it (the price) is close to 40,000 RMB.

Figure 1: An example conversation between two people. The first dropped pronoun refers to "this computer" in the first sentence and is translated into "it" in English. It needs to be recovered from the wider context.

More concretely, recovering DPs involves i) first locating the position in a sentence where a pronoun is dropped, and ii) determining the type of the pronoun that is dropped. This is illustrated in Figure 1, where the recovered dropped pronouns are in parenthesis. Both instances of dropped pronouns can be replaced with the overt pronoun 它 ("it"), but they refer to different things in the conversation. Dropped pronoun recovery is different from zero pronoun (ZP) resolution (Chen and Ng, 2015; Yin et al., 2017, 2018) where the focus is on resolving an anaphoric pronoun to its antecedent, assuming the position of the zero pronoun is already determined. Here we do not attempt to resolve the dropped pronoun to its referent. Dropped pronoun recovery can thus be viewed as the first step of zero pronoun resolution, but this task is also important in its own right. In applications like machine translation, dropped pronouns only need to be identified and translated correctly but they do not need to be resolved.

Traditionally dropped pronoun recovery has

---

[*] Corresponding author

Proceedings of NAACL-HLT 2019, pages 892–901
Minneapolis, Minnesota, June 2 - June 7, 2019.

been formulated as a sequence labeling problem where each word in the sentence receives a tag that indicates whether the word has a dropped pronoun before it and which pronoun is dropped. Yang et al. (2015) leverages lexical, contextual and syntactic features to detect DPs before each word from a predefined list of pronouns. Giannella et al. (2017) utilizes a linear-chain conditional random field (CRF) classifier to simultaneously predict the position as well as the person number of a dropped pronoun based on lexical and syntactic information. These featured-based methods require labor-intensive feature engineering.

In contrast, Zhang et al. (2016) uses a multi-layer perceptron based neural network model to recover dropped pronouns, which eliminates the need for feature engineering. The input to their model is a context embedding constructed by concatenating the embeddings of word tokens in a fixed-length window. However, the referent of a dropped pronoun that provides crucial information to determine the identity of a pronoun is typically found outside the local context of the dropped pronoun, and thus cannot be effectively modeled with a window-based multi-layer perceptron model. For example, in Figure 1 the referent of the first dropped pronoun, the third person singular pronoun 它 (“it”) is 电脑 (“computer”), which appears in the very first utterance, several utterances before the pro-drop sentence. Therefore, long-distance contextual information needs to be captured in order to determine the type of the dropped pronoun.

In this paper, we describe a novel Neural Dropped Pronoun Recovery framework, named NDPR that can model the referent information in a much larger context. The model makes use of contextual information at two levels of granularity: the sentence level and the word level. An illustration of the NDPR framework is given in Figure 2. The referent modeling process is implemented with a structured attention mechanism. For each word token in the pro-drop sentence, the model attends to the utterance in which the referent of the dropped pronoun is mentioned by sentence attention and then zeros in to the referent by word attention. The resulting referent representation is combined with the representation of the dropped pronoun to make the final prediction on the location and type of the dropped pronoun. We demonstrate the effectiveness of our proposed model on three Chinese conversational datasets and results show that our approach outperforms the current state of the art by a fairly large margin.

We also perform ablation studies to explore the contribution of the different components of our model. We show that word-level attention is more effective in recovering *concrete* pronouns because it serves as a matching mechanism that matches the representation of the dropped pronoun against that of the DP referent. In contrast, sentence-level attention can be considered as an auxiliary tool for word-level attention to improve model accuracy by filtering out irrelevant referents. All code is available at `https://github.com/ningningyang/NDPR`.

Our contributions are summarized as follows:

- We propose a novel attention-based neural network to recover dropped pronouns in Chinese conversational data by modeling their referent.
- We evaluate our system on three conversational genres and show that our model consistently outperforms the state of the art by a large margin on all three datasets.
- We also present experimental results that demonstrate the effectiveness of various components in our framework and analyze some mistakes made by our model on each dataset.

2 Related Work

2.1 Dropped pronoun recovery.

Dropped pronoun detection originates from Empty Category (EC) detection and resolution, a task aimed to recover certain dropped elements in syntactic treebanks (Chung and Gildea, 2010; Cai et al., 2011; Xue and Yang, 2013). Dropped pronoun recovery was first proposed as an independent task in (Yang et al., 2015), which leveraged a set of specially designed features to recover DPs in Chinese text messages. Giannella et al. (2017) employed a linear-chain CRF classifier to jointly determine the position and person number of dropped pronouns in Chinese SMS messages using hand-crafted features. These traditional feature-based methods require heavy feature engineering. Zhang et al. (2016) for the first time utilized a multi-layer perceptron model to recover dropped pronouns. Each word is expressed as a concatenation of embeddings of word tokens in a fixed-length window. This method can not resolve the cases when

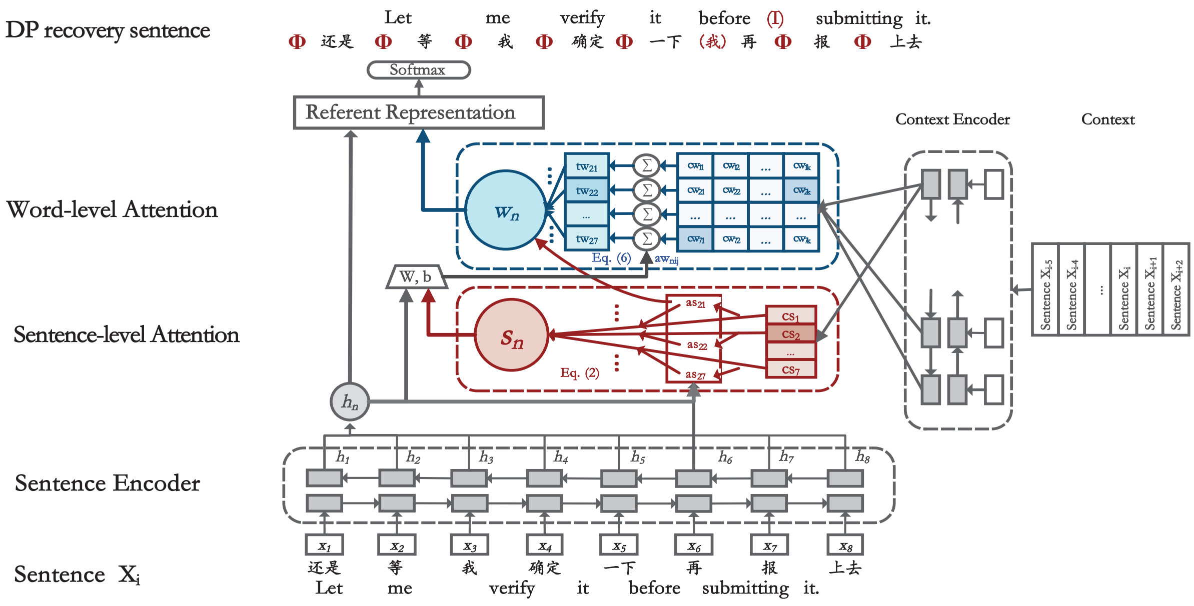


Figure 2: Neural Dropped Pronoun Recovery Framework.

referents are outside the local context, which is a common occurrence in conversational data. There has also been some work that attempts to recover dropped pronoun in pro-drop languages to help Machine Translation (Wang et al., 2016a,b, 2018). There are some inherent differences between their work and ours. In their work, they attempt to create training data by projecting pronouns from another language to Chinese, and the specific location of a dropped pronoun may not matter as much as long as it is translated correctly when Chinese is the source language. In contrast, we focus on recovering DPs from conversational context, and determining the correct position of a dropped pronoun is critical part of the task.

2.2 Zero pronoun resolution.

A line of research that is closely related to our task is zero pronoun (ZP) resolution, which aims to resolve Chinese pronouns to their antecedents. Converse and Palmer (2006) presented a rule-based method to resolve ZP by utilizing the Hobbs algorithm (Hobbs, 1978). Learning-based anaphoric resolution approaches have also been extensively explored. Zhao and Ng (2007) and Kong and Zhou (2010) proposed systems that perform ZP resolution by integrating syntactic features and position information in a system based on decision trees and context-sensitive convolution tree kernels. With the powerful learning capacity of neural networks, recent work focuses on learning representations to resolve zero pronouns or common noun phrases (Ng, 2007; Yin et al., 2017, 2018). The DP recovery task explored in our work focuses on determining the position and type of dropped pronoun without attempting to resolve it to an antecedent.

2.3 Attention and memory network.

The attention mechanism is first proposed for neural machine translation (Bahdanau et al., 2014) and has been attempted in a variety of natural language processing tasks (Bahdanau et al., 2014; Hermann et al., 2015). Kim (2017) extended the basic attention mechanism to incorporate structural biases by combining graphical models with neural networks. Yang et al. (2016) proposed a word and sentence-level hierarchical attention network for document classification. Xing et al. (2018) also presented a hierarchical recurrent attention network (HRAN) to model the hierarchy of conversation context in order to generate multi-turn responses in chatbots. In contrast with these bottom-up mechanisms, our model adopts a top-down structured attention mechanism to construct the representation for a DP. Our model attempts to identify utterances which contain the referent of a DP, and then focus in on the representation of the referent itself. Our model draws inspiration from the memory network and its variants (Weston et al., 2015; Sukhbaatar et al., 2015; Henaff et al., 2017; Miller et al., 2016), where an external memory component is used to store and update knowledge.

3 Approach

Following (Yang et al., 2015), we formulate DP recovery as a sequential tagging problem. Given an input sentence $X = (x_1, x_2, ..., x_s)$ and its contextual utterances $\boldsymbol{C} = (X_1, ..., X_m)$, our task is to model $P(Y|X, \boldsymbol{C})$ and predict a set of labels $Y = (y_1, y_2, ..., y_s)$. Each element of Y indicates whether a pronoun is dropped and which type it should be before each word in the sentence X. The NDPR model consists of three components: (1) Input Layer; (2) Referent Modeling Layer; (3) Output Layer. We describe them in detail below.

3.1 Input Layer

This layer converts the raw inputs X and $\boldsymbol{C}$ into distributed representations.

3.1.1 Sentence X

During the encoding process, the input sentence of length s, is transformed into a sequence of d-dimensional word embeddings at first, and then fed into two different kinds of bidirectional recurrent neural networks (RNN) (Elman, 1991):

- **BiGRU** (Bahdanau et al., 2014): Each word $x_n, n \in \{1, ..., s\}$ of the sentence is represented as a concatenation of forward and backward hidden states as: $\boldsymbol{h_n} = [\overleftarrow{\boldsymbol{h}}_{\boldsymbol{n}}, \overrightarrow{\boldsymbol{h}}_{\boldsymbol{n}}]$. We aim to express DP by the hidden state of the first word after DP.
- **PC-BiGRU** (Yin et al., 2017): Pronoun-centered BiGRU (PC-BiGRU) also contains two independent GRU networks. For each word $x_n, n \in \{1, ..., s\}$, the forward GRU_f encodes the preceding context of DP from left to right as $\overrightarrow{\boldsymbol{h}}_{\boldsymbol{n-1}}$, and the backward GRU_b models the succeeding context as $\overleftarrow{\boldsymbol{h}}_{\boldsymbol{n}}$ in the reverse direction. Final hidden state of DP is also concatenated as $\boldsymbol{h_n} = [\overleftarrow{\boldsymbol{h}}_{\boldsymbol{n}}, \overrightarrow{\boldsymbol{h}}_{\boldsymbol{n-1}}]$. The intuition of this structure is to express DP by the last word before DP and the first word after DP.

3.1.2 Context C

The context provides the necessary background information to recover DPs. In our work, we utilize five utterances preceding the one the DP is in and two utterances following current input sentence X as context **C**. The size of **C** is determined empirically since our statistics show that 97% of the dropped pronouns can be inferred based on information in the 7 sentences surrounding the sentence in which the dropped pronoun occurs. The context **C** passes through the same encoder as X, yielding two kinds of memories: (1) sentence-level memory: concatenated final states of the forward and backward GRU for each contextual utterance $i = \{1, ..., m\}$ as $\boldsymbol{cs_i} = [\overleftarrow{\boldsymbol{cs}}_{\boldsymbol{i}}, \overrightarrow{\boldsymbol{cs}}_{\boldsymbol{i}}]$, holding sentence level background knowledge. (2) word-level memory: a set of concatenated hidden states at each time step $j = \{1, ..., k\}$ as $\boldsymbol{cw_{i,j}} = [\overleftarrow{\boldsymbol{cw}}_{\boldsymbol{i,j}}, \overrightarrow{\boldsymbol{cw}}_{\boldsymbol{i,j}}]$, expressing the contextual information of words in memory.

3.2 Referent Modeling layer

The referent modeling layer models the interaction between the input sentence X and its context $\boldsymbol{C}$ and outputs representation of DP referents. The relevant information for predicting the dropped pronoun is retrieved by a two-step attention-based inference mechanism in which the intermediate results of sentence-level attention is utilized to support subsequent word-level inference.

3.2.1 Identify utterances mentioning DP

This operation aims to figure out which utterances mention the intended referent of DP. We compute the relevance between the hidden states $\boldsymbol{h_n}$ representing the DP and its sentence level contextual representation $\boldsymbol{cs_i}$, resulting in $rs_{n,i}$. After passing it through softmax, we obtain a sentence-level attention distribution $as_{n,i}$ as:

$$rs_{n,i} = \boldsymbol{h_n^T cs_i} \tag{1}$$

$$as_{n,i} = \frac{e^{rs_{n,i}}}{\sum_{i=1}^{m} e^{rs_{n,i}}} \tag{2}$$

We then conclude sentence-level background information $\boldsymbol{s_n}$ of DP as a weighted sum of contextual utterance representations:

$$\boldsymbol{s_n} = \sum_{i=1}^{m} as_{n,i} \cdot \boldsymbol{cs_i} \tag{3}$$

3.2.2 Update DP representation

This operation updates representation of the DP by combining the original DP information $\boldsymbol{h_n}$ with sentence-level background knowledge $\boldsymbol{s_n}$ through a linear transformation as:

$$\boldsymbol{hs_n} = \boldsymbol{W^{2d \times 4d}}[\boldsymbol{h_n}, \boldsymbol{s_n}] + \boldsymbol{b^{2d}}, \tag{4}$$

The updated DP state $\boldsymbol{hs_n}$ will be used in subsequent word level inference.

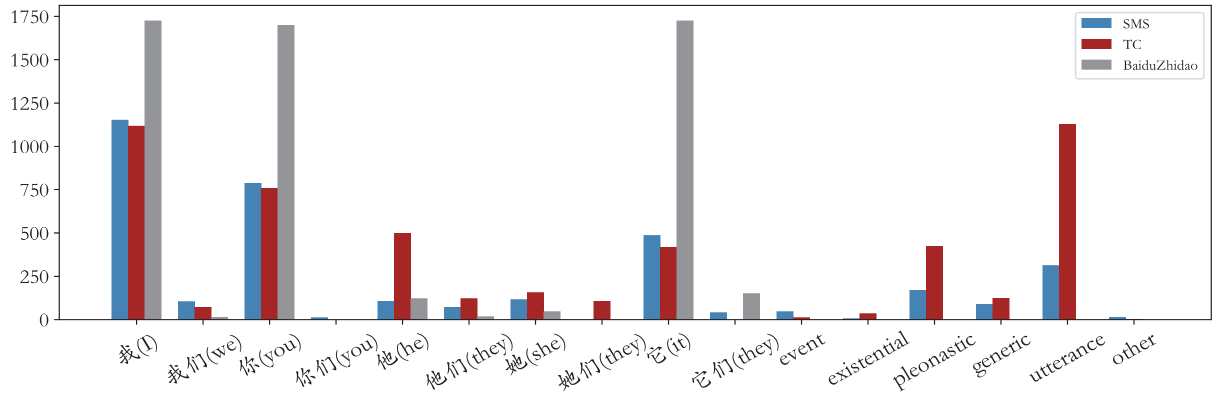


Figure 3: Counts of each type of dropped pronouns in different conversational data sets. Pronouns have a more balanced distribution in Chinese SMS and telephone conversation(TC). For BaiduZhidao, only concrete pronouns are annotated and the distribution of these pronouns is extremely uneven.

3.2.3 Identify DP referents

This operation aims to capture the word sequence that represents the intended referent of the DP. We cast the updated DP state $\boldsymbol{hs_n}$ onto the space of contextual words $\boldsymbol{cw_{i,j}}$ by measuring the similarity $rw_{n,i,j}$ between them. The casting operation also serves as a regularization measure to restrict the inference space. The similarities are fed into the softmax function to give a word-level attention distribution $aw_{n,i,j}$ for each contextual word as:

$$rw_{n,i,j} = \boldsymbol{W^{1\times 2d}}(\boldsymbol{hs_n} \odot \boldsymbol{cw_{i,j}}) + b^1 \qquad (5)$$

$$aw_{n,i,j} = \frac{e^{rw_{n,i,j}}}{\sum_{j=1}^{k} e^{rw_{n,i,j}}}, \qquad (6)$$

The attention distribution $aw_{n,i,j}$ also provides an interpretation for DP recovery results based on the words it attends to.

Finally, we derive the word-level representation $\boldsymbol{w_n}$ from word states $\boldsymbol{cw_{i,j}}$. Word-level attention $aw_{n,i,j}$ is used to compute a weighted sum of contextual word states $\boldsymbol{cw_{i,j}}$ of each utterance, yielding the referent representation $\boldsymbol{tw_{n,i}}$. Then, sentence-level attention as_i further filters out irrelevant referents, yielding the final referent representation $\boldsymbol{w_n}$ as:

$$\boldsymbol{tw_{n,i}} = \sum_{j=1}^{k} aw_{n,i,j} \cdot \boldsymbol{cw_{i,j}} \qquad (7)$$

$$\boldsymbol{w_n} = \sum_{i=1}^{m} as_{n,i} \cdot \boldsymbol{tw_{n,i}} \qquad (8)$$

3.3 Output Layer and Training Objective

The output layer predicts the recovery result based on DP state $\boldsymbol{h_n}$ and referent representation $\boldsymbol{w_n}$. We feed the concatenation of these two parts into a 2-layer fully connected softmax classifier to give a categorical probability distribution over 17 candidate pronouns as:

$$\boldsymbol{\alpha_n} = tanh(\boldsymbol{W_1} \cdot [\boldsymbol{h_n}; \boldsymbol{w_n}] + \boldsymbol{b}) \qquad (9)$$

$$P(y_n|x_n,\boldsymbol{C}) = softmax(\boldsymbol{W_2} \cdot \boldsymbol{\alpha_n} + \boldsymbol{b}) \qquad (10)$$

We train our model by minimizing cross-entropy between the predicted label distributions and the annotated labels for all sentences. The training objective is defined as:

$$loss = -\sum_{l\in N}\sum_{n=1}^{s}\delta(y_n|x_n,\boldsymbol{C})\log(P(y_n|x_n,\boldsymbol{C})), \qquad (11)$$

where N represents all training instances, s represents the number of words in each sentence; $\delta(y_n|x_n,c)$ represents the annotated label of x_n.

4 Datasets and Experimental Setup

4.1 Datasets

We perform experiments on three Chinese conversational datasets.

- Chinese text message (SMS) data, which consists of 684 SMS/Chat files. We use the same dataset split as (Yang et al., 2015), which reserves 16.7% of the training set as a held-out development set to tune the hyper-parameters and evaluate the model on a separate test set.

- OntoNotes Release 5.0, which is used in the CoNLL 2012 Shared Task. We use a portion of OntoNotes 5.0 that consists of transcripts of Chinese telephone conversation (TC) speech. The 9,507-sentence subset of OntoNotes only has coreference annotations for anaphoric zero pronouns. In order to formulate the task as an end-to-end sequence labeling task, we annotate all the dropped pronouns following annotation guidelines described in (Yang et al., 2015).
- BaiduZhidao, which is a question answering dialogue dataset used in (Zhang et al., 2016). It contains 11,160 sentences that are annotated with 10 types of concrete dropped pronouns.

Figure 3 shows the statistics of each type of pronouns in the three data sets. According to (Yang et al., 2015), 5 out of 15 types of dropped pronouns in the SMS data set are *abstract pronouns* that do not correspond to any actual pronoun in Chinese. These can refer to an event (Event), the previous utterance (Previous Utterance), or an generic or unspecific entity (Generic). The other two abstract pronouns are used to indicate the subject of an existential construction (Existential) or a pleonastic subject (Pleonastic). The other ten types of Chinese pronouns are classified as concrete pronouns. The same 15 types of pronouns are used to annotate the TC data. We can see that the BaiDuZhidao data set only has concrete pronouns.

4.2 Evaluation metrics

We use the same evaluation metrics as (Yang et al., 2015): precision (P), recall (R) and F-score (F).

4.3 Training details

Our model is implemented with Tensorflow. The vocabulary is generated from the training set, which contains 17,199 word types. Out-of-vocabulary (OOV) words are represented as UNK. The BiGRU encoder uses a hidden layer of 150 units. To train the model, we use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0003. We train the model for 8 epochs and select the model with the highest F-score on the development set for testing. Dropout rate is set at 0.2 on the fully connected layers. We use uniform initialization for the weight matrices and zero initialization for biases.

4.4 Baseline Methods and Model Variations

We compare our proposed NDPR framework with three baselines and implement two parallel experiments:

- **MEPR**: This model is provided by Yang (2015), which uses a maximum entropy (ME) classifier with hand-crafted features to recover dropped pronouns.
- **NRM**: This model is proposed by Zhang (2016), which uses two independent multi-layer perceptrons to locate the position of dropped pronouns and determine the type of dropped pronouns.
- **BiGRU**: This model encodes each word in the target sentence with a bidirectional GRU. The output representation of the BiGRU encoder is used to predict the type of the dropped pronoun. This method can be seen as a degenerate variant of our NDPR model without the attention mechanism.
- **NDPR**: This model uses the BiGRU encoder with both sentence-level and word-level attention. The word embeddings are initialized with pre-trained 300-D Zhihu QA vectors (Li et al., 2018) and fine-tuned when training the DP recovery model.
- **NDPR-rand**: Same as NDPR except that the word embeddings are randomly initialized.
- **NDPR-PC-BiGRU**: Same as NDPR but the encoder of utterance X is replaced with PC-BiGRU.

We also perform two ablation experiments to explore the effectiveness of sentence-level attention and word-level attention:

- **NDPR-S**: Same as NDPR but the referent modeling layer only utilizes sentence-level attention. The output layer makes prediction with only DP state $\boldsymbol{h_n}$ and sentence level information $\boldsymbol{s_n}$.
- **NDPR-W**: Same as NDPR but the referent modeling layer only utilizes word-level attention. The output layer makes prediction with only DP state $\boldsymbol{h_n}$ and word-level information $\boldsymbol{w_n}$.

5 Results and Discussion

5.1 Main Results

Table 1 shows experimental results of three baseline systems and variants of our proposed NDPR

Model	Chinese SMS			TC of OntoNotes			BaiduZhidao		
	P(%)	R(%)	F	P(%)	R(%)	F	P(%)	R(%)	F
MEPR (Yang et al., 2015)	37.27	45.57	38.76	-	-	-	-	-	-
NRM* (Zhang et al., 2016)	37.11	44.07	39.03	23.12	26.09	22.80	26.87	49.44	34.54
BiGRU	40.18	45.32	42.67	25.64	36.82	30.93	29.35	42.38	35.83
NDPR-rand	46.47	43.23	43.58	28.98	41.50	33.38	35.44	43.82	37.79
NDPR-PC-BiGRU	46.34	46.21	46.27	36.69	40.12	38.33	38.42	48.01	41.68
NDPR-W	46.78	**46.61**	45.76	38.67	41.56	39.64	38.60	**50.12**	**43.36**
NDPR-S	46.99	46.32	44.89	37.40	40.32	38.81	39.32	46.40	41.53
NDPR	**49.39**	44.89	**46.39**	**39.63**	**43.09**	**39.77**	**41.04**	46.55	42.94

Table 1: Results in terms of precision, recall and F-score on 16 types of pronouns produced by the baseline systems and variants of our proposed NDPR model. For NRM* (Zhang et al., 2016), we implement the proposed model as described in the paper.

Tag	NDPR-S	NDPR-W	NDPR
他们 (masculine they)	17.05	23.28	**24.44**
她 (she)	32.35	33.72	**35.14**
previous utterance	84.90	86.08	**87.55**
他 (he)	29.05	31.20	**34.92**
它 (it)	25.00	26.67	**26.95**
她们 (feminine they)	0	0	**40.00**
我 (I)	50.66	50.90	**52.98**
我们 (we)	31.49	33.57	**34.81**
你 (singular you)	42.88	44.15	**44.31**
pleonastic	25.89	22.29	**28.46**
generic	11.61	11.08	**16.83**
event	6.15	0	**16.27**
existential	34.17	30.84	**38.71**
你们 (plural you)	0	0	**5.41**
它们 (inanimate they)	16.00	**19.15**	13.89

Table 2: F-scores of our proposed model NDPR and its two variants (NDPR-S, NDPR-W) for concrete and abstract pronouns on the Chinese SMS test set.

model on Chinese SMS, TC section of OntoNotes, and BaiduZhidao. We can see that our proposed model and its variants outperform the baseline methods on all these datasets by different margins. Our best model, NDPR, outperforms MEPR by 7.63% in terms of F-score on the Chinese SMS dataset, and outperforms NRM by 16.97% and 8.40% on the OntoNotes and BaiduZhidao datasets respectively. Compared with the degenerate variant model BiGRU, our NDPR model also performs better on all three datasets, which demonstrate the effectiveness of referent modeling mechanism composed of sentence-level and word-level attention.

The experimental results also show that all components of our model have made a positive contribution, which is evidenced by the fact that the full model NDPR outperforms the other variants by a small margin. The results also show that pretrained word embeddings have had a positive impact on the model, as shown by the higher accuracy of the NDPR model over the NDPR-rand model, where the word embeddings are randomly initialized.

PC-BiGRU encoder seems to perform worse than vanilla BiGRU encoder, as shown by the slightly lower F-score of the NDPR-PC-BiGRU model. But it alleviates what we call the local pronoun repetition problem, the situation where a DP is recovered redundantly in a sentence before a verb and its adverbial modifier. We attribute this to that PC-BiGRU uses the context state h_{n-1} of the last word before DP instead of h_n, which makes the model pay more attention to the preceding modifier.

For the BaiduZhidao dataset, the NDPR-W model actually performs better than NDPR model, as indicated by the higher F-score for NDPR-W in the last column of Table 1. We attribute this to the fact that there are only concrete pronouns in this dataset. The combination of “我 (I)”, “你 (singular you)” and “它 (it)” accounts for 94.47% of the overall dropped pronoun population, for which the referent can be easily determined by word-level attention. Moreover, the fewer conversation turns in this data set mean there are few irrelevant referents that need to be filtered out by sentence-level attention.

5.2 Ablation Study

In this section, we dive a bit deeper and look at the impact of the attention mechanism on concrete and

Pro-drop sentence: A: (她)(she) 是 用的 锤头 还是 扳手?
A: What did (she) use, a hammer or wrench?

A: 你 终于 出现 了 。
You finally showed up.
B: 你 别 乱说 。
Don't talk nonsense.
A: 啊 哈哈 。
Haha.
B: 最近 我 的 电脑 它 让 我 女儿(daughter) 搞坏 了
My daughter broke my computer recently.
A: 哈哈 她(she) 还 有 这 功能 她(she) 不 简单
Haha. She can even do that. She is not to be under-estimated.
B: 嗯 , 破坏力 很 强 。
Yes, very destructive.
B: 她(she) 用 了 一 根 手指 。
She just used a finger.

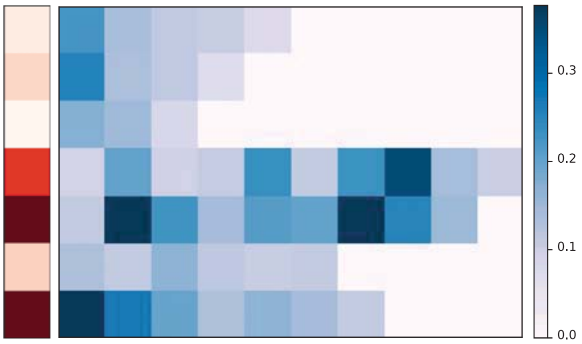


Figure 4: Visualization of attention in the NDPR model. The red pattern on the left shows the distribution of sentence-level attention and the blue pattern on the right shows the distribution of word-level attention. Darker color indicates higher attention weight.

Chinese SMS

Context : A: 我还以为你大过年要搬家呢！
A: I thought you would make a move during the Spring Festival.
A: 我希望你能在报社附近找个房子。
A: I hope you can rent a house near the newspaper office.
B: 嗯，我能接上最好了。
B: Well, it is best if the new house is ready.
Gold : B: (event) 不行就只能搬回去了。
B: I have to move back if the new house is not accessible.
NDPR: B: (我) 不行就只能搬回去了。
B: I have to move back if I can not.

Context : A: 我可以从上海直飞到纽约吗？
A: Can I take a direct flight from Shanghai to New York?
Gold : B: (generic) 直飞都没有
B: There is no direct flight.
NDPR : B: (我) 直飞都没有。
B: I can not flight directly.

TC of OntoNotes

Gold : (你) 有空，(你) 有空 (你) 可能还会到北京去，是吧？
You may go to Beijing again when you are free, you are free, right?
NDPR : (你) 有空，(我) 有空 (我) 可能还会到北京去，是吧？
I may go to Beijing again when you are free, I am free, right?

BaiduZhidao

Context : A: 医院开的那些退烧药有用吗？
A: Are those antipyretics prescribed by the hospital useful?
Gold : B: (它们) 都有什么成分？
B: What are the ingredients of them?
NDPR : B: (它) 都有什么成分？
B: What are the ingredients of it?

Figure 5: Example errors made by NDPR on different conversational datasets.

abstract pronouns, respectively. Table 2 shows the F-scores of three variants of our proposed model for each type of pronouns on the entire Chinese SMS test set. The best results among these three variants are in boldface, and the better results between NDPR-S and NDPR-W are underlined.

We can see that for all types of concrete pronouns, the NDPR-W model outperforms the NDPR-S model by a significant margin. In general, the referent of concrete dropped pronoun is usually realized in the form of a phrase that consists of one or more word, and they can be accurately captured by word-level attention. The NDPR model that incorporates both word-level and sentence-level attention further improves upon NDPR-W, with the lone exception of 它们 (“they”). We believe the reason is that the sentence-level encoder cannot adequately represent the interaction between the multiple referents for plural pronouns like 它们 and only add noise to the representation.

However, this observation does not hold for abstract dropped pronouns. The NDPR-W performs comparably or slightly worse on four out of the five types of abstract pronouns. This is consistent with the fact that the referent of abstract pronouns like “Event” is often an entire sentence. The full NDPR model still outperforms both NDPR-W and NDPR-S for all abstract pronouns.

5.3 Attention Visualization

Intuitively, sentence-level attention models the interaction between the dropped pronoun and each context utterance. The utterance containing the referent of the dropped pronoun should receive more attention. Similarly, word-level attention models the interaction between each word in the context and the dropped pronoun. Thus words that describe the referent of the pronoun should receive more attention if the model works as intended. Figure 4 shows a instance of attention distribution produced by our model.

In this example, the model correctly gives higher attention weights to three utterances that contain words that indicate the referent of the pronoun. At word level, the model also gives higher attention weights to the specific words that indicate the referent of the dropped pronoun. For example, words such as 她 (“she”) and 女儿 (“daughter”) have received higher weights, as indicated by the darker color. This suggests that in this case the model attends to the right utterances and words and works as we have expected.

5.4 Error Analysis

Figure 5 shows some typical mistakes made by our model on each genre. For the Chinese SMS dataset, the distribution of different types of pronouns is relatively balanced (See Table 3), and our model does a better job on concrete pronouns but stumbles on abstract pronouns like "event" and "generic" as shown in Table 2 as it is harder to attend to the correct parts of the context in these cases.

The TC data of OntoNotes is a transcription of telephone conversations where there are often repetitions, and our model struggles when people repeat what they said. If the same pronoun is mentioned repeatedly, our model can not capture this interaction, since each pronoun is recover independently. This suggests that one future improvement might involve using a sequence-based decoder.

In the BaiduZhidao dataset, only concrete pronouns are annotated as shown in Figure 3. Pronouns like 它 ("it") and 它们 ("they") account for a large proportion of all dropped pronouns. For these two categories, the performance of our proposed method is hit-and-miss, which can be attributed to the absence of pronoun resolution common sense.

6 Conclusions

We have proposed an end-to-end neural network architecture that attempts to model the interaction between the dropped pronoun and its referent in order to recover dropped pronouns in Chinese conversational data. Our model is based on sentence-level and word-level attention and results show that our model consistently outperforms baseline methods when evaluated on three separate datasets. We further investigate the effectiveness of different components of our model by performing ablation experiments and demonstrate the interpretability of our model using attention visualization.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61702047, No.61300080) and Beijing Natural Science Foundation (No.4174098).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Y Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science.*, 1409.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. pages 212–216. Proceedings of the Meeting of the Association for Computational Linguistics.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.*, volume 2, pages 320–326.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. pages 636–645.

Susan P Converse and Martha Stone Palmer. 2006. *Pronominal anaphora resolution in Chinese*. Citeseer.

Jeffrey L. Elman. 1991. *Distributed Representations, Simple Recurrent Networks, And Grammatical Structure*. Springer US.

Chris Giannella, Ransom K Winder, and Stacy Petersen. 2017. Dropped personal pronoun recovery in chinese sms. *Natural Language Engineering*, 23(06):905–927.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann Lecun. 2017. Tracking the world state with recurrent entity networks. *International Conference on Learning Representations.*

Karl Moritz Hermann, Tomas Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems.*, pages 1693–1701.

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *International Conference on Learning Representations*.

Young Joo Kim. 2000. Subject/object drop in the acquisition of korean: A cross-linguistic comparison. *Journal of East Asian Linguistics.*, 9(4):325–351.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations.*

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.*, pages 882–891. Association for Computational Linguistics.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. *Proceedings of the Meeting of the Association for Computational Linguistics.*

Alexander H Miller, Adam Fisch, Jesse Dodge, Amirhossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *Conference on Empirical Methods in Natural Language Processing.*, pages 1400–1409.

Vincent Ng. 2007. Semantic class induction and coreference resolution. In *Proceedings of the Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, pages 536–543.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems.*, pages 2440–2448.

Longyue Wang, Zhaopeng Tu, Shuming Shi, Tong Zhang, Yvette Graham, and Qun Liu. 2018. Translating pro-drop languages with reconstruction models. *arXiv preprint arXiv:1801.03257.*

Longyue Wang, Zhaopeng Tu, Xiaojun Zhang, Hang Li, Andy Way, and Qun Liu. 2016a. A novel approach to dropped pronoun translation. *arXiv preprint arXiv:1604.06285.*

Longyue Wang, Xiaojun Zhang, Zhaopeng Tu, Hang Li, and Qun Liu. 2016b. Dropped pronoun generation for dialogue machine translation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6110–6114. IEEE.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *International Conference on Learning Representations (ICLR).*

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. *National Conference on Artificial Intelligence.*

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. pages 1051–1060.

Yaqin Yang, Yalin Liu, and Nianwen Xue. 2015. Recovering dropped pronouns from chinese text messages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.*, volume 2, pages 309–313.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*, pages 1480–1489.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. Chinese zero pronoun resolution with deep memory network. In *Conference on Empirical Methods in Natural Language Processing.*, pages 1309–1318.

Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu, and William Yang Wang. 2018. Deep reinforcement learning for chinese zero pronoun resolution. *meeting of the association for computational linguistics*, pages 569–578.

Weinan Zhang, Ting Liu, Qingyu Yin, and Yu Zhang. 2016. Neural recovery machine for chinese dropped pronoun. *arXiv: Computation and Language.*

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.*

The Problem with Probabilistic DAG Automata for Semantic Graphs

Ieva Vasiljeva* and **Sorcha Gilroy*** and **Adam Lopez**
Institute for Language, Cognition, and Computation
School of Informatics
University of Edinburgh
{vasiljeva.ieva, gilroysorcha}@gmail.com, alopez@inf.ed.ac.uk

Abstract

Semantic representations in the form of directed acyclic graphs (DAGs) have been introduced in recent years, and to model them, we need probabilistic models of DAGs. One model that has attracted some attention is the DAG automaton, but it has not been studied as a probabilistic model. We show that some DAG automata cannot be made into useful probabilistic models by the nearly universal strategy of assigning weights to transitions. The problem affects single-rooted, multi-rooted, and unbounded-degree variants of DAG automata, and appears to be pervasive. It does not affect planar variants, but these are problematic for other reasons.

1 Introduction

Abstract Meaning Representation (AMR; Banarescu et al. 2013) has prompted a flurry of interest in probabilistic models for semantic parsing. AMR annotations are directed acyclic graphs (DAGs), but most probabilistic models view them as strings (e.g. van Noord and Bos, 2017) or trees (e.g. Flanigan et al., 2016), removing their ability to represent coreference—one of the very aspects of meaning that motivates AMR. Could we we instead use probabilistic models of DAGs?

To answer this question, we must define probability distributions over sets of DAGs. For inspiration, consider probability distributions over sets of strings or trees, which can be defined by weighted finite automata (e.g. Mohri et al., 2008; May et al., 2010): a finite automaton generates a set of strings or trees—called a language—and if we assume that probabilities factor over its transitions, then any finite automaton can be weighted to define a probability distribution over this language. This assumption underlies powerful dynamic programming algorithms like the Viterbi, forward-backward, and inside-outside algorithms.

What is the equivalent of weighted finite automata for DAGs? There are several candidates (Chiang et al., 2013; Björklund et al., 2016; Gilroy et al., 2017), but one appealing contender is the *DAG automaton* (Quernheim and Knight, 2012) which generalises finite tree automata to DAGs explicitly for modeling semantic graphs. These DAG automata generalise an older formalism called *planar DAG automata* (Kamimura and Slutzki, 1981) by adding weights and removing the planarity constraint, and have attracted further study (Blum and Drewes, 2016; Drewes, 2017), in particular by Chiang et al. (2018), who generalised classic dynamic programming algorithms to DAG automata. But while Quernheim and Knight (2012) clearly intend for their weights to define probabilities, they stop short of claiming that they do, instead ending their paper with an open problem: "*Investigate a reasonable probabilistic model.*"

We investigate probabilistic DAG automata and prove a surprising result: **For some DAG automata, it is impossible to assign weights that define non-trivial probability distributions**. We exhibit a very simple DAG automaton that generates an infinite language of graphs, and for which the only valid probability distribution that can be defined by weighting transitions is one in which the support is a single DAG, with all other graphs receiving a probability of zero.

Our proof relies on the fact that a non-planar DAG automaton generates DAGs so prolifically that their number grows factorially in their size, rather than exponentially as in other automata. It holds for DAG automata that allow multiple roots or nodes of unbounded degree. But it breaks down when applied to the planar DAGs of Kamimura and Slutzki (1981), which are nevertheless too restrictive to model semantic graphs. Our result does

---

[*] Equal contribution. Work while Ieva Vasiljeva was at the University of Edinburgh

Proceedings of NAACL-HLT 2019, pages 902–911
Minneapolis, Minnesota, June 2 - June 7, 2019.

not mean that it is impossible to define a probability distribution for the language that a DAG automaton generates. But it does mean that this distribution does not factor over the automaton's transitions, so crucial dynamic programming algorithms do not generalise to DAG automata that are expressive enough to model semantic graphs.

2 DAGs, DAG Automata, and Probability

We are interested in AMR graphs like the one below for "Rahul bakes his cake" (Figure 1, left), which represents entities and events as nodes, and relationships between them as edges. Both nodes and edges have labels, representing the type of an entity, event, or relationship. But the graphs we model will only have labels on nodes. These node-labeled graphs can simulate edge labels using a node with one incoming and one outgoing edge, as in the graph on the right of Figure 1.

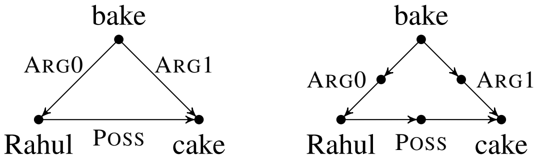


Figure 1: A graph with both node and edge labels (left) and an equivalent graph with only node labels (right).

Definition 1. *A node-labeled directed* ***graph*** *over a label set* Σ *is a tuple* $G = (V, E, lab, src, tar)$ *where* V *is a finite set of nodes,* E *is a finite set of edges,* $lab\colon V \to \Sigma$ *is a function assigning labels to nodes,* $src\colon E \to V$ *is a function assigning a source node to every edge, and* $tar\colon E \to V$ *is a function assigning a target node to every edge.*

Sometimes we will discuss the set of edges coming into or going out of a node, so we define functions IN: $V \to E^*$ and OUT: $V \to E^*$.

$$\textsc{In}(v) = \{e \mid \text{tar}(e) = v\}$$
$$\textsc{Out}(v) = \{e \mid \text{src}(e) = v\}$$

A node with no incoming edges is called a **root**, and a node with no outgoing edges is called a **leaf**. The **degree** of a node is the number of edges connected to it, so the degree of v is $|\textsc{In}(v) \cup \textsc{Out}(v)|$.

A **path** in a directed graph from node v to node v' is a sequence of edges $(e_1, \ldots, e_n)$ where $\text{src}(e_1) = v$, $\text{tar}(e_n) = v'$ and $\text{src}(e_{i+1}) = \text{tar}(e_i)$ for all i from 1 to $n-1$. A **cycle** in a directed graph is any path in which the first and last nodes are the same (i.e., $v = v'$). A directed graph without any cycles is a **directed acyclic graph (DAG)**.

A DAG is **connected** if every pair of its nodes is connected by a sequence of edges, not necessarily directed. Because DAGs do not contain cycles, they must always have at least one root and one leaf, but they can have multiple roots and multiple leaves. However, our results apply in different ways to single-rooted and multi-rooted DAG languages, so, given a label set Σ, we distinguish between the set of all connected DAGs with a single root, $\mathcal{G}^1_\Sigma$; and those with one or more roots, $\mathcal{G}^*_\Sigma$.

2.1 DAG Automata

Finite automata generate strings by transitioning from state to state. Top-down tree automata generalise string finite automata by transitioning from a state to an ordered sequence of states, generating trees top-down from root to leaves; while bottom-up tree automata transition from an ordered sequence of states to a single state, generating trees bottom-up from leaves to root. The planar DAG automata of Kamimura and Slutzki (1981) generalise tree automata, transitioning from one ordered sequence of states to another ordered sequence of states (Section 4). Finally, the DAG automata of Quernheim and Knight (2012) transition from *multisets* of states to multisets of states, rather than from sequences to sequences, and this allows them to generate non-planar DAGs. We summarise the differences in Table 1 below.

Automaton	Transitions	Example
string	one-to-one	$p \to p'$
top-down tree	one-to-many	$p \to (p', q')$
bottom-up tree	many-to-one	$(p', q') \to p$
planar DAG	many-to-many	$(p, q) \to (p', q')$
non-planar DAG	many-to-many	$\{p, q\} \to \{p', q'\}$

Table 1: The forms of transitions in different automata.

For the remainder of this section and the next, we will focus only on non-planar DAG automata, and when we refer to DAG automata, we mean this type. To formally define them, we need a notation for multisets—sets that can contain repeated elements. A **multiset** is a pair (S, m) where S is a finite set and $m : S \to \mathbb{N}$ is a count function—that is, $m(x)$ counts the number of times x appears in the multiset. The set of all finite multisets over S is $M(S)$. When we write multisets, we will often simply enumerate their elements. For example, $\{p, q, q\}$ is the multiset containing one p and two

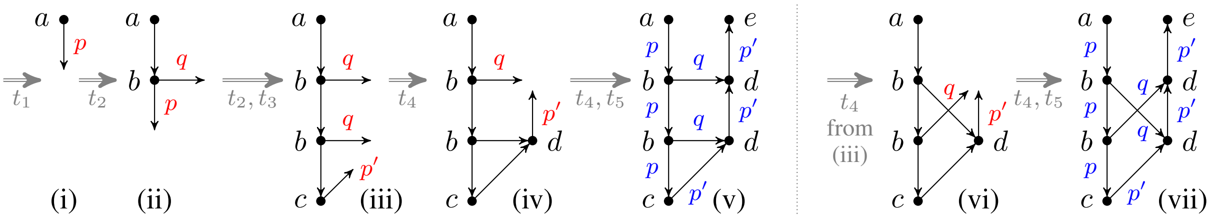


Figure 2: Two derivations using the automaton of Example 1. Parts (i), (ii), and (iii) are common to both derivations. Parts (iv) and (v) represent one completion, while (vi) and (vii) represent an alternative completion. Grey double edges denote derivation steps, labeled with the corresponding transition(s); red edge labels on partial graphs (i–iv) and (vi) denote frontier states; blue edge labels on complete graphs (v) and (vii) denote an accepting run.

q's, and since multisets are unordered, it can also be written $\{q, p, q\}$ or $\{q, q, p\}$. We write $\emptyset$ for a multiset containing no elements.

Definition 2. *A **DAG automaton** is a triple* $A = (Q, \Sigma, T)$ *where* Q *is a finite set of states;* Σ *is a finite set of node labels; and* T *is a finite set of transitions of the form* $\alpha \xrightarrow{\sigma} \beta$ *where* $\sigma \in \Sigma$ *is a node label,* $\alpha \in M(Q)$ *is the left-hand side, and* $\beta \in M(Q)$ *is the right-hand side.*

Example 1. Let $A = (Q, \Sigma, T)$ be a DAG automaton where $Q = \{p, p', q\}$, $\Sigma = \{a, b, c, d, e\}$ and the transitions in T are as follows:

$$\emptyset \xrightarrow{a} \{p\} \quad (t_1)$$
$$\{p\} \xrightarrow{b} \{p, q\} \quad (t_2)$$
$$\{p\} \xrightarrow{c} \{p'\} \quad (t_3)$$
$$\{p', q\} \xrightarrow{d} \{p'\} \quad (t_4)$$
$$\{p'\} \xrightarrow{e} \emptyset \quad (t_5)$$

2.1.1 Generating Single-rooted DAGs

A DAG automaton generates a graph from root to leaves. To illustrate this, we'll focus on the case where a DAG is allowed to have only a single root, and return to the multi-rooted case in Section 3.1. To generate the root, the DAG automaton can choose any transition with $\emptyset$ on its left-hand side—these transitions behave like transitions from the start state in a finite automaton on strings, and always generate roots. In our example, the only available transition is t_1, which generates a node labeled a with a dangling outgoing edge in state p, as in Figure 2(i). The set of all such dangling edges is the **frontier** of a partially-generated DAG.

While there are edges on the frontier, the DAG automation must choose and apply a transition whose left-hand side matches some subset of them. In our example, the automaton can choose either t_2 or t_3, each matching the available p edge. The edges associated with the matched states are attached to a new node with new outgoing frontier edges specified by the transition, and the matched states are removed from the frontier. If our automaton chooses t_2, it arrives at the configuration in Figure 2(ii), with a new node labeled b, new edges on the frontier labeled p and q, and the incoming p state forgotten. Once again, it must choose between t_2 and t_3—it cannot use the q state because that state can only be used by t_4, which also requires a p' on the frontier. So each time it applies t_2, the choice between t_2 and t_3 repeats.

If the automaton applies t_2 again and then t_3, as it has done in Figure 2(iii), it will face a new set of choices, between t_4 and t_5. But notice that choosing t_5 will leave the q states stranded, leaving a partially derived DAG. We consider a run of the automaton successful only when the frontier is empty, so this choice leads to a dead end.

If the automaton chooses t_4, it has an additional choice: it can combine p' with *either* of the available q states. If it combines with the lowermost q, it arrives at the graph in Figure 2(iv), and it can then apply t_4 to consume the remaining q, followed by t_5, which has $\emptyset$ on its right-hand side. Transitions to $\emptyset$ behave like transitions to a final state in a finite automaton, and generate leaf nodes, so we arrive at the complete graph in Figure 2(v). If the p' state in Figure 2(iii) had instead combined with the upper q, a different DAG would result, as shown in Figure 2(vi-vii).

The DAGs in Figure 2(v) and Figure 2(vii) are planar, which means they can be drawn without crossing edges.[1] But this DAG automaton can also produce non-planar DAGs like the one in Figure 3. To see that it is non-planar, we first *contract* each dotted edge by removing it and fusing its endpoints into a single node. This gives us the *minor*

---

[1] While the graph in Figure 2(vii) is drawn with crossing $b-d$ edges, one of these edges can be redrawn so that they do not cross.

subgraph $K_{3,3}$, and any graph with a $K_{3,3}$ minor is non-planar (Wagner, 1937).

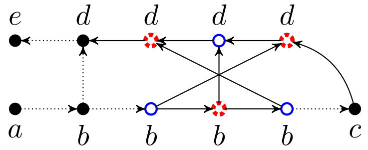


Figure 3: A non-planar graph that can be generated by the automaton of Example 1. When the dotted edges are contracted, we obtain $K_{3,3}$, the complete (undirected) bipartite graph over two sets of three nodes. One set is denoted by hollow blue nodes (○), the other by dotted red nodes (⁘).

2.1.2 Recognising DAGs and DAG Languages

We define the language generated by a DAG automaton in terms of recognition, which asks if an input DAG could have been generated by an input automaton. We recognise a DAG by finding a run of the automaton that could have generated it. To guess a run on a DAG, we guess a state for each of its edges, and then ask whether those states simulate a valid sequence of transitions.

A **run** of a DAG automaton $A = (Q, \Sigma, T)$ on a DAG $G = (V, E, \text{lab}, \text{src}, \text{tar})$ is a mapping $\rho : E \rightarrow Q$ from edges of G to automaton states Q. We extend ρ to multisets by saying $\rho(\{e_1, \ldots, e_n\}) = \{\rho(e_1), \ldots, \rho(e_n)\}$, and we call a run **accepting** if for all $v \in V$ there is a corresponding transition $\rho(\textsc{In}(v)) \xrightarrow{\text{lab}(v)} \rho(\textsc{Out}(v))$ in T. DAG G is **recognised** by automaton A if there is an accepting run of A on G.

Example 2. The DAGs in Figure 2(v) and 2(vii) are recognised by the automaton in Example 1. The only accepting run for each DAG is denoted by the blue edge labels.

The **single-rooted language** $L_s(A)$ of a DAG automaton A is $\{G \in \mathcal{G}^1_\Sigma \mid A \text{ recognizes } G\}$.

2.2 Probability and Weighted DAG Automata

Definition 3. *Given a language L of DAGs, a **probability distribution** over L is any function $p : L \rightarrow \mathbb{R}$ meeting two requirements:*

(R1) *Every DAG must have a probability between 0 and 1, inclusive. Formally, we require that for all $G \in L$, $p(G) \in [0, 1]$.*

(R2) *The probabilities of all DAGs must sum to one. Formally, we require $\sum_{G \in L} p(G) = 1$.*

R1 and R2 suffice to define a probability distribution, but in practice we need something stronger than R1: all DAGs must receive a *non-zero* weight, since in practical applications, objects with probability zero are effectively not in the language.

Definition 4. *A probability distribution p has **full support** of L if and only if it meets condition R1'.*

(R1') *Every DAG must have a probability greater than 0 and less than or equal to 1. Formally, we require that for all $G \in L$, $p(G) \in (0, 1]$.*

While there are many ways to define a function that meets requirements R1' and R2, probability distributions in natural language processing are widely defined in terms of weighted automata or grammars, so we adapt a common definition of weighted grammars (Booth and Thompson, 1973) to DAG automata.

Definition 5. *A **weighted DAG automaton** is a pair (A, w) where $A = (Q, \Sigma, T)$ is a DAG automaton and $w : T \rightarrow \mathbb{R}$ is a function that assigns real-valued weights to the transitions of A.*

Since weights are functions of transitions, we will write them on transitions following the node label and a slash (/). For example, if $p \xrightarrow{a} q$ is a transition and 2 is its weight, we write $p \xrightarrow{a/2} q$.

Example 3. Let (A, w) be a weighted DAG automaton with $A = (Q, \Sigma, T)$, where $Q = \{p, q\}$, $\Sigma = \{a, b, c\}$, and the weighted transitions of T are as follows:

$$\emptyset \xrightarrow{a/0.5} \{p, q\} \quad (t'_1)$$

$$\{p\} \xrightarrow{b/0.5} \{p\} \quad (t'_2)$$

$$\{p, q\} \xrightarrow{c/1} \emptyset \quad (t'_3)$$

We use the weights on transitions to weight runs.

Definition 6. *Given a weighted DAG automaton (A, w) and a DAG $G = (V, E, lab, src, tar)$ with an accepting run ρ, we extend w to compute the **weight of the run** $w(\rho)$ by multiplying the weights of all of its transitions:*

$$w(\rho) = \prod_{v \in V} w(\rho(\textsc{In}(v)) \xrightarrow{lab(v)} \rho(\textsc{Out}(v)))$$

Example 4. The DAG automaton of Example 3 generates the DAG in Figure 4, shown with its only accepting run in blue and the weighted transitions that generated it in grey. The weight of the accepting run is $0.5 \times 0.5 \times 0.5 \times 1 = 0.125$.

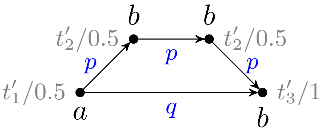


Figure 4: A DAG generated by the automaton in Example 3. Blue edge labels denote an accepting run; grey node labels denote weighted transitions used in the run.

Let $R_A(G)$ be the set of all accepting runs of a DAG G using the automaton A. We extend w to calculate the weight of a DAG G as the sum of the weights of all the runs that produce it:

$$w(G) = \sum_{\rho \in R_A(G)} w(\rho).$$

While all weighted DAG automata assign real values to DAGs, not all weighted DAG automata define probability distributions. To do so, they must also satisfy requirements R1 and R2.

Definition 7. *A weighted automaton (A, w) over language $L(A)$ is **probabilistic** if and only if function $w : L(A) \to \mathbb{R}$ is a probability distribution.*

Example 5. Consider the weighted automaton in Example 3. Every DAG generated by this automaton must use t'_1 and t'_3 exactly once, and can use t'_2 any number of times. If we let G_n be the DAG that uses t'_2 exactly n times, then the language L defined by this automaton is $\bigcup_{n \in \mathbb{N}} G_n$. Since $w(G_n) = w(t'_1)w(t'_2)^n w(t'_3)$ and $w(t'_1)$, $w(t'_2)$ and $w(t'_3)$ are positive, w satisfies R1 and:

$$\sum_{G \in L} w(G) = \sum_{n=0}^{\infty} w(G_n) = \sum_{n=0}^{\infty} w(t'_1)w(t'_2)^n w(t'_3)$$
$$= \sum_{n=0}^{\infty} 0.5^{n+1} = 1$$

Thus w also satisfies R2 and the weighted automaton in Example 3 is probabilistic.

Definition 8. *A probabilistic automaton (A, w) over language $L(A)$ is **probabilistic with full support** if and only if w has full support of $L(A)$.*

For every finite automaton over strings or trees, there is a weighting of its transitions that makes it probabilistic (Booth and Thompson, 1973), and it is easy to show that it can be made probabilistic with full support. For example, string finite automata have full support if for every state the sum of weights on its outgoing transitions is 1 and each weight is greater than 0.[2] But as we will show, this is not always possible for DAG automata.

3 Non-probabilistic DAG Automata

We will exhibit a DAG automaton that generates factorially many DAGs for a given number of nodes, and we will show that for any nontrivial assignment of weights, this factorial growth rate causes the weight of all DAGs to sum to infinity.

Theorem 1. *Let $\mathcal{A}$ be the automaton defined in Example 1. There is no w that makes $(\mathcal{A}, w)$ probabilistic with full support over $L_s(\mathcal{A})$.*

Proof. In any run of the automaton, transition t_1 is applied exactly once to generate the single root, placing a p on the frontier. This gives a choice between t_2 and t_3. If the automaton chooses t_2, it keeps one p on the frontier and adds a q, and must then repeat the same choice. Suppose it chooses t_2 exactly n times in succession, and then chooses t_3. Then the frontier will contain n edges in state q and one in state p'. The only way to consume all of the frontier states is to apply transition t_4 exactly n times, consuming a q at each step, and then apply t_5 to consume p' and complete the derivation. Hence in any accepting run, t_1, t_3 and t_5 are each applied once, and t_2 and t_4 are each applied n times, for some $n \geq 0$. Since transitions map uniquely to node labels, it follows that every DAG in $L_s(\mathcal{A})$ will have exactly one node each labeled a, c, and e; and n nodes each labeled b and d.

When the automaton applies t_4 for the first time, it has n choices of q states to consume, each distinguished by its unique path from the root. The second application of t_4 has $n-1$ choices of q, and the ith application of t_4 has $n-(i-1)$ choices. Therefore, there are $n!$ different ways to consume the q states, each producing a unique DAG.

Let $f(n)$ be the weight of a run where t_2 has been applied n times, and to simplify our notation, let $B = w(t_1)w(t_3)w(t_5)$, and $C = w(t_2)w(t_4)$. Let $c(n)$ be the number of unique runs where t_2 has been applied n times. By the above:

$$f(n) = w(t_1)w(t_2)^n w(t_3)w(t_4)^n w(t_5) = BC^n$$
$$c(n) = n!$$

Now we claim that any DAG in $L_s(\mathcal{A})$ has exactly one accepting run, because the mapping of

---

[2] Assuming no epsilon transitions, in our notation for DAG automata restricted to strings this would include transitions to $\emptyset$, which correspond to states with a final probability of 1 (Mohri et al., 2008).

node labels to transitions also uniquely determines the state of each edge in an accepting run. For example, a b node must result from a t_2 transition and a d node from a t_4 transition, and since the output states of t_2 and input states of t_4 share only a q, any edge from a b node to a d node must be labeled q in any accepting run. Now let $G \in L_s(\mathcal{A})$ be a DAG with n nodes labeled b. Since G has only one accepting run, we have:

$$w(G) = f(n)$$

Let L_n be the set of all DAGs in $L_s(\mathcal{A})$ with n nodes labeled b. Then $L_s(\mathcal{A}) = \bigcup_{n=0}^{\infty} L_n$ and:

$$\begin{aligned}\sum_{G \in L_s(\mathcal{A})} w(G) = \sum_{n=0}^{\infty} \sum_{G \in L_n} w(G) &= \sum_{n=0}^{\infty} c(n) f(n) \\ &= \sum_{n=0}^{\infty} (n!)(BC^n)\end{aligned}$$

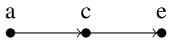
Hence for $(\mathcal{A}, w)$ to be probabilistic with full support, R1' and R2 require us to choose B and C so that, respectively, $BC^n \in (0, 1]$ for all n and $\sum_{n=0}^{\infty} n!BC^n = 1$. Note that this does not constrain the component weights of B or C to be in $(0, 1]$—they can be any real numbers. But since R1' requires BC^n to be positive for all n, both B and C must also be positive. If either were 0, then BC^n would be 0 for $n > 0$; if either were negative, then BC^n would be negative for some or all values of n.

Now we show that any choice of positive C causes $\sum_{G \in L_s(\mathcal{A})} w(G)$ to diverge. Given an infinite series of the form $\sum_{n=0}^{\infty} a_n$, the **ratio test** (D'Alembert, 1768) considers the ratio between adjacent terms in the limit, $\lim_{n \to \infty} \frac{|a_{n+1}|}{|a_n|}$. If this ratio is greater than 1, the series diverges; if less than 1 the series converges; if exactly 1 the test is inconclusive. In our case:

$$\lim_{n \to \infty} \frac{|(n+1)!BC^{n+1}|}{|n!BC^n|} = \lim_{n \to \infty} (n+1)|C| = \infty.$$

Hence $\sum_{G \in L_s(\mathcal{A})}$ diverges for any choice of C, equivalently for any choice of weights. So there is no w for which $(\mathcal{A}, w)$ is probabilistic with full support over $L_s(\mathcal{A})$. $\square$

Note that any automaton recognising $L_s(\mathcal{A})$ must accept factorially many DAGs in the number of nodes. Our proof implies that there is *no* probabilistic DAG automaton for language $L_s(\mathcal{A})$, since no matter how we design its transitions—each of which must be isomorphic to one in $\mathcal{A}$ apart from the identities of the states—the factorial will eventually overwhelm the constant factor corresponding to C in our proof, no matter how small it is.

Theorem 1 does not rule out *all* probabilistic variants of $\mathcal{A}$. It requires R1'—if we only require the weaker R1, then a solution of B=1 and C=0 makes the automaton probabilistic. But this trivial distribution is not very useful: it assigns all of its mass to the singleton language {  }.

Theorem 1 also does not mean that it is impossible to define a probability distribution over $L_s(\mathcal{A})$ with full support. If, for every DAG G with n nodes labeled b, we let $p(G) = \frac{1}{2^{n+1} n!}$, then:

$$\sum_{G \in L_s(\mathcal{A})} w(G) = \sum_{n=0}^{\infty} \frac{1}{2^{n+1} n!} n! = \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} = 1$$

But this distribution does not factor over transitions, so it cannot be used with the dynamic programming algorithms of Chiang et al. (2018).

A natural way to define distributions using a DAG automaton is to define two conditional probabilities: one over the choice of nodes to rewrite, given a frontier; and one over the choice of transition, given the chosen nodes. The latter factors over transitions, but the former does not, so it also cannot use the algorithms of Chiang et al. (2018).[3]

Theorem 1 only applies to single-rooted, non-planar DAG automata of bounded degree. Next we ask whether it extends to other DAG automata, including those that recognise multi-rooted DAGs, DAGs of unbounded degree, and planar DAGs.

3.1 Multi-rooted DAGs

What happens when we consider DAG languages that allow multiple roots? In one reasonable interpretation of AMRbank, over three quarters of the DAGs have multiple roots (Kuhlmann and Oepen, 2016), so we want a model that permits this.[4]

Section 2.1.1 explained how a DAG automaton can be constrained to generate single-rooted languages, by restricting start transitions (i.e. those

---

[3] In this model, the subproblems of a natural dynamic program depend on the set of possible frontiers, rather than subsets of nodes as in the algorithms of Chiang et al. (2018). We do not know whether this could be made efficient.

[4] AMR annotations are single-rooted, but they achieve this by duplicating edges: every edge type, like ARG0, has an inverse type, like ARG0-OF. The number cited here assumes edges of the second type are converted to the first type by reversing their direction.

with $\emptyset$ on the left-hand side) to a single use at the start of a derivation. To generate DAGs with multiple roots, we simply allow start transitions to be applied at any time. We still require the resulting DAGs to be connected. For an automaton A, we define its **multi-rooted language** $L_m(A)$ as $\{G \in \mathcal{G}^*_\Sigma | A \text{ recognises } G\}$.

Although one automaton can define both single- and multi-rooted DAG languages, these languages are incomparable. Drewes (2017) uses a construction very similar to the one in Theorem 1 to show that single-rooted languages have very expressive path languages, which he argues are too expressive for modeling semantics.[5] Since the constructions are so similar, it natural to wonder if the problem that single-rooted automata have with probabilities is related to their problem with expressivity, and whether it likewise disappears when we allow multiple roots. We now show that multi-rooted languages have the same problem with probability, because any multi-rooted language contains the single-rooted language as a sublanguage.

Corollary 1. *Let $\mathcal{A}$ be the automaton defined in Example 1. There is no w that makes $(\mathcal{A}, w)$ probabilistic with full support over $L_m(\mathcal{A})$.*

Proof. By their definitions, $L_s(\mathcal{A}) \subset L_m(\mathcal{A})$, so:

$$\sum_{G \in L_m(\mathcal{A})} w(G) = \sum_{G \in L_s(\mathcal{A})} w(G) + \sum_{G \in L_m(\mathcal{A}) \setminus L_s(\mathcal{A})} w(G)$$

The first term is ∞ by Theorem 1 and the second is positive by R1', so the sum diverges. Hence there is no w for which $(\mathcal{A}, w)$ is probabilistic with full support over $L_m(\mathcal{A})$. $\square$

3.2 DAGs of Unbounded Degree

The maximum degree of any node in any DAG recognised by a DAG automaton is bounded by the maximum number of states in any transition, because any transition $\alpha \xrightarrow{\sigma} \beta$ generates a node with $|\alpha|$ incoming edges and $|\beta|$ outgoing edges. So, the families of DAG languages we have considered all have bounded degree.

DAG languages with unbounded degree could be useful to model phenomena like coreference in meaning representations, and they have been studied by Quernheim and Knight (2012) and Chiang et al. (2018). These families generalise and strictly contain the family of bounded-degree DAG languages, so they too, include DAG automata that cannot be made probabilistic.

3.3 Implications for semantic DAGs

We introduced DAG automata as a tool for modeling the meaning of natural language, but the DAG automaton in Theorem 1 is very artificial, so it's natural to ask whether it has any real relevance to natural language. We will argue informally that this example illustrates a pervasive problem with DAG automata—specifically, we conjecture that the factorial growth we observe in Theorem 1 arises under very mild conditions that arise naturally in models of AMR.

Consider object control in a sentence like "I help Ruby help you" and its AMR in Figure 5.

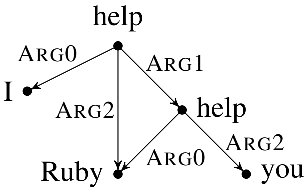


Figure 5: The AMR for "I help Ruby help you".

We can extend the control structure unboundedly with additional helpers, as in "I help Briony help Kim-Joy help Ruby help you", and this leads to unboundedly long repetitive graphs like the one in Figure 6. These graphs can be cut to separate the sequence of "help" predicates from their arguments, as illustrated by the dashed blue line.

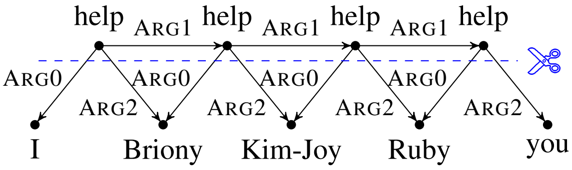


Figure 6: The AMR for "I help Briony help Kim-Joy help Ruby help you" shown with a cut.

Let a **cut** be a set of edges such that removing them splits the graph into two connected subgraphs: one containing the root, and the other containing all the leaves. Any cut in a complete graph

---

[5] The **path language** of a DAG is the set of strings that label a path from a root to a leaf, and the path language of a DAG language is the set of all such strings over all DAGs. For example, the path language of the DAG in Figure 2(v) is $\{abde, abbdde, abbcdde\}$. Berglund et al. (2017) show that path languages of multi-rooted DAG automata are regular, while those of single-rooted DAG automata characterised by a partially blind multi-counter automaton.

could have been the frontier of a partially-derived graph. What if the number of edges in a cut—or **cut-width**—can be unbounded, as in the language of AMR graphs that model object control?

Since a DAG automaton can have only a finite number of states, there is some state that can occur unboundedly many times in a graph cut. All edges in a cut with this state can be rewired by permuting their target nodes, and the resulting graph will still be recognised by the automaton, since the rewiring would not change the multiset of states into or out of any node. If each possible rewiring results in a unique graph then the number of recognised graphs will be factorial in the number of source nodes for these edges, and the argument of Theorem 1 can be generalised to show that no weighting of any DAG automaton over the graph language makes it probabilistic with full support. For example, in the graph above, all possible rewirings of the ARG2 edges result in a unique graph.[6] Although edge labels are not states, their translation into node labels implies that they can only be associated to a finite number of transitions, hence to a finite number of states in any corresponding DAG automaton. A full investigation of conditions under which Theorem 1 generalises is beyond the scope of this paper.

Conjecture 1. *Under mild conditions, if language* $L(A)$ *of a DAG automaton* A *has unbounded cut-width, there is no* w *that makes* (A, w) *probabilistic with full support.*

4 Planar DAG Automata

The fundamental problem with trying to assign probabilities to non-planar DAG automata is the factorial growth in the number of DAGs with respect to the number of nodes. Does this problem occur in planar DAG automata?

Planar DAG automata are similar to the DAG automata of Section 2 but with an important difference: they transition between *ordered* sequences of states rather than unordered multisets of states. We write these sequences in parentheses, and their order matters: (p, q) differs from (q, p). We write ϵ for the empty sequence. When a planar DAG automaton generates DAGs, it keeps a strict order over the set of frontier states at all times. A transition whose left-hand side is (p, q) can only be applied to adjacent states p and q in the frontier, with p preceding q. The matched states are replaced in the frontier by the sequence of states in the transition's right-hand side, maintaining order.

Example 6. Consider a planar DAG automaton with the following transitions:

$$\begin{array}{rcl} \epsilon \xrightarrow{a} (p) & & (t''_1) \\ (p) \xrightarrow{b} (p, q) & & (t''_2) \\ (p) \xrightarrow{c} (p') & & (t''_3) \end{array} \qquad \begin{array}{rcl} (p', q) \xrightarrow{d} (p') & & (t''_4) \\ (p') \xrightarrow{e} \epsilon & & (t''_5) \end{array}$$

In the non-planar case, n applications of t_2 can generate $n!$ unique DAGs, but n applications of the corresponding transition t''_2 in this automaton can only generate one DAG. To see this, consider the partially derived DAG on the left of Figure 7, with its frontier drawn in order from left to right. The p' state can only combine with the q state immediately to its right, and since dead-ends are not allowed, the only possible choice is to apply t''_4 twice followed by t''_5, so the DAG on the right is the only possible completion of the derivation.

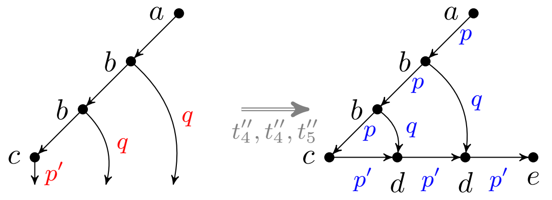


Figure 7: A partial derivation using the planar DAG automaton of Example 6 (left; red edge labels denote frontier states) and its only possible completion (right; blue edge labels denote an accepting run).

This automaton is probabilistic when $w(t''_1) = w(t''_2) = 1/2$, $w(t''_3) = w(t''_4) = w(t''_5) = 1$, and indeed the argument in Theorem 1 does not apply to planar automata since the number of applicable transitions is linear in the size of the frontier. But planar DAG automata have other problems that make them unsuitable for modeling AMR.

The first problem is that there are natural language constructions that naturally produce non-planar DAGs in AMR. For example, consider the sentence "Four contestants mixed, baked and ate cake." Its AMR, shown in Figure 8, is not planar because it has a $K_{3,3}$ minor, and it is easy to see from this example that any coordination of three predicates sharing two arguments produces this structure. In the first release of AMR, 117 out of 12844 DAGs are non-planar.

The second problem is that planar DAG au-

---

[6] This is also a problem linguistically, since many of the rewired graphs no longer model object control.

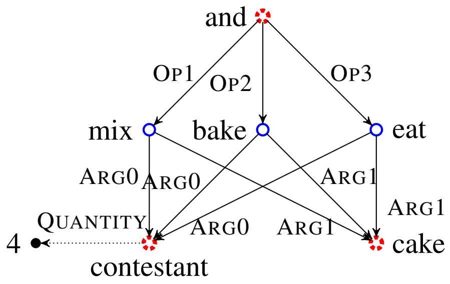


Figure 8: An AMR for the sentence "Four contestants mixed, baked and ate a cake". As in Figure 3, contracting the dotted edge yields a $K_{3,3}$ minor, with one set denoted by hollow blue nodes (○), the other by dotted red nodes (◌).

tomata model Type-0 string derivations by design (Kamimura and Slutzki, 1981). This seems more expressive than needed to model natural language and means that many important decision problems are undecidable—for example, emptiness, which is decidable in polynomial time for non-planar DAG automata (Chiang et al., 2018).

5 Conclusions

Table 2 summarises the properties of several different variants of DAG automata. It has been argued that all of these properties are desirable for probabilistic models of meaning representations (Drewes, 2017). Since none of the variants supports all properties, this suggests that no variant of the DAG automaton is a good candidate for modeling meaning representations. We believe other formalisms may be more suitable, including several subfamilies of hyperedge replacement grammars (Drewes et al., 1997) that have recently been proposed (Björklund et al., 2016; Matheja et al., 2015; Gilroy et al., 2017).

	non-planar				planar
bounded degree	yes		no		yes
roots	1	1+	1	1+	1
probabilistic	no	no	no	no	?
decidable	yes	yes	yes	yes	no
regular paths	no	yes	no	yes	no

Table 2: DAG automata variants and their properties.

Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh. We thank Esma Balkir, Nikolay Bogoychev, Shay Cohen, Marco Damonte, Federico Fancellu, Joana Ribeiro, Nathan Schneider, Miloš Stanojević, Ida Szubert, Clara Vania, and the anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Martin Berglund, Henrik Björklund, and Frank Drewes. 2017. Single-rooted dags in regular dag languages: Parikh image and path languages. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 94–101, Umeå, Sweden.

Henrik Björklund, Frank Drewes, and Petter Ericson. 2016. Between a rock and a hard place - uniform parsing for hyperedge replacement DAG grammars. In *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, pages 521–532.

Johannes Blum and Frank Drewes. 2016. Properties of regular DAG languages. In *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, pages 427–438.

T.L. Booth and R.A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 924–932, Sofia, Bulgaria.

David Chiang, Frank Drewes, Daniel Gildea, Adam Lopez, and Giorgio Satta. 2018. Weighted DAG automata for semantic graphs. *Computational linguistics*, 44(1).

Jean D'Alembert. 1768. *Opuscules*, volume V.

Frank Drewes. 2017. Dag automata for meaning representation. In *Proceedings of the 15th Meeting on the Mathematics of Language*, pages 88–99, London, UK.

Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. 1997. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proc. of NAACL-HLT*, pages 731–739.

Sorcha Gilroy, Adam Lopez, Sebastian Maneth, and Pijus Simonaitis. 2017. (Re)introducing regular graph languages. In *Proceedings of the 15th Meeting on the Mathematics of Language (MoL 15)*, pages 100–113.

Tsutomu Kamimura and Giora Slutzki. 1981. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control*, 49(1):10–51.

Marco Kuhlmann and Stephan Oepen. 2016. Squibs: Towards a catalogue of linguistic graph banks. *Computational Linguistics*, 42(4):819–827.

Christoph Matheja, Christina Jansen, and Thomas Noll. 2015. *Tree-Like Grammars and Separation Logic*, pages 90–108. Springer International Publishing, Cham.

Jonathan May, Kevin Knight, and Heiko Vogler. 2010. Efficient inference through cascades of weighted tree transducers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1058–1066.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In Larry Rabiner and Fred Juang, editors, *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*, pages 69–88. Springer.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Daniel Quernheim and Kevin Knight. 2012. Towards probabilistic acceptors and transducers for feature structures. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-6 '12, pages 76–85.

Klaus Wagner. 1937. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590.

A Systematic Study of Leveraging Subword Information for Learning Word Representations

Yi Zhu, Ivan Vulić, and **Anna Korhonen**
Language Technology Lab
University of Cambridge
`{yz568, iv250, alk23}@cam.ac.uk`

Abstract

The use of subword-level information (e.g., characters, character n-grams, morphemes) has become ubiquitous in modern word representation learning. Its importance is attested especially for morphologically rich languages which generate a large number of rare words. Despite a steadily increasing interest in such *subword-informed word representations*, their systematic comparative analysis across typologically diverse languages and different tasks is still missing. In this work, we deliver such a study focusing on the variation of two crucial components required for subword-level integration into word representation models: 1) *segmentation of words* into subword units, and 2) *subword composition functions* to obtain final word representations. We propose a general framework for learning subword-informed word representations that allows for easy experimentation with different segmentation and composition components, also including more advanced techniques based on position embeddings and self-attention. Using the unified framework, we run experiments over a large number of subword-informed word representation configurations (60 in total) on 3 tasks (general and rare word similarity, dependency parsing, fine-grained entity typing) for 5 languages representing 3 language types. Our main results clearly indicate that there is no "one-size-fits-all" configuration, as performance is both language- and task-dependent. We also show that configurations based on unsupervised segmentation (e.g., BPE, Morfessor) are sometimes comparable to or even outperform the ones based on supervised word segmentation.

1 Introduction

Word representations are central to a wide variety of NLP tasks (Collobert et al., 2011; Chen and Manning, 2014; Jia and Liang, 2016; Ammar et al., 2016; Goldberg, 2017; Peters et al., 2018; Kudo, 2018, *inter alia*). Standard word representation models are based on the distributional hypothesis (Harris, 1954) and induce representations from large unlabeled corpora using word co-occurrence statistics (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014). However, as pointed out by recent work (Bojanowski et al., 2017; Vania and Lopez, 2017; Pinter et al., 2017; Chaudhary et al., 2018; Zhao et al., 2018), mapping a finite set of word types into corresponding word representations limits the capacity of these models to learn beyond distributional information, which leads to several fundamental limitations.

The standard approaches ignore the internal structure of words, that is, the syntactic or semantic composition from subwords or morphemes to words, and are incapable of parameter sharing at the level of subword units. Assigning only a single vector to each word causes the data sparsity problem, especially in resource-poor settings where huge amounts of training data cannot be guaranteed. The issue is also prominent for morphologically rich languages (e.g., Finnish) with productive morphological systems that generate a large number of infrequent/rare words (Gerz et al., 2018). Although potentially useful information on word relationships is hidden in their internal *subword-level structure*,[1] subword-agnostic word representation models do not take these structure features into account and are effectively unable to represent rare words accurately, or unseen words at all.

Therefore, there has been a surge of interest in *subword-informed* word representation architectures aiming to address these gaps. A large number of architectures has been proposed in related research, and they can be clustered over the two main axes (Lazaridou et al., 2013; Luong et al., 2013;

---

[1] For example, nouns in Finnish have 15 cases and 3 plural forms; Spanish verbs may contain over 40 inflected forms, sharing the lemma and taking up standard suffixes.

Proceedings of NAACL-HLT 2019, pages 912–932
Minneapolis, Minnesota, June 2 - June 7, 2019.

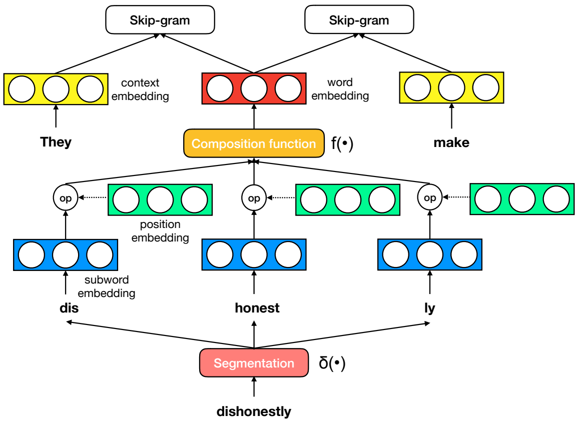


Figure 1: Illustration of the general framework for learning subword-informed word representations, with the focus on two crucial components: 1) segmentation of words and 2) subword embedding composition. By varying the two components, and optionally including or excluding position embeddings from the computations, we obtain a wide spectrum of different subword-informed configurations used in the study (see §2). Our word-level representation model in this work is skip-gram (on the top layer of the figure), but it can be replaced by any other distributional word-level model.

Qiu et al., 2014; Cotterell and Schütze, 2015; Wieting et al., 2016; Avraham and Goldberg, 2017; Vania and Lopez, 2017; Pinter et al., 2017; Cotterell and Schütze, 2018). First, the models differ in the chosen method for *segmenting words into subwords*. The methods range from fully supervised approaches (Cotterell and Schütze, 2015) to e.g. unsupervised approaches based on BPE (Heinzerling and Strube, 2018). Second, another crucial aspect is the *subword composition function* used to obtain word embeddings from the embeddings of each word's constituent subword units. Despite a steadily increasing interest in such subword-informed word representations, their systematic comparative analysis across the two main axes, as well as across typologically diverse languages and different tasks is still missing.[2]

In this work, we conduct a systematic study of a variety of subword-informed word representation architectures that all can be described by a general framework illustrated by Figure 1. The framework enables straightforward experimentation with prominent word segmentation methods (e.g., BPE, Morfessor, supervised segmentation systems) as well as subword composition functions (e.g., addition, self-attention), resulting in a large number of different *subword-informed configurations*.[3]

Our study aims at providing answers to the following crucial questions: **Q1)** How generalizable are subword-informed models across typologically diverse languages and across different downstream tasks? Do different languages and tasks require different configurations to reach peak performances or is there a single best-performing configuration? **Q2)** How important is it to choose an appropriate segmentation and composition method? How effective are more generally applicable unsupervised segmentation methods? Is it always better to resort to a supervised method, if available? **Q3)** Is there a difference in performance with and without the full word representation? Can more advanced techniques based on position embeddings and self-attention yield better task performance?

We evaluate subword-informed word representation configurations originating from the general framework in three different tasks using standard benchmarks and evaluation protocols: 1) general and rare word similarity and relatedness, 2) dependency parsing, and 3) fine-grained entity typing for 5 languages representing 3 language families (fusional, introflexive, agglutinative). We show that different tasks and languages indeed require diverse subword-informed configurations to reach peak performance: this calls for a more careful language- and task-dependent tuning of configuration components. We also show that more sophisticated configurations are particularly useful for representing rare words, and that unsupervised segmentation methods can be competitive to supervised segmentation in tasks such as parsing or fine-grained entity typing. We hope that this paper will provide useful points of comparison and comprehensive guidance for developing next-generation subword-informed word representation models for typologically diverse languages.

2 Methodology

The general framework for learning subword-informed word representations, illustrated by Figure 1, is introduced in §2.1. We then describe its main components: segmentation of words into subword units (§2.2), subword and position embeddings (§2.3), and subword embedding composition

---

[2] A preliminary study of Vania and Lopez (2017) limits its focus on the use of subwords in the language modeling task.

[3] Following a similar work on subword-agnostic word embedding learning (Levy et al., 2015), our system design choices resulting in different configurations can be seen as a set of hyper-parameters that also have to be carefully tuned for each language and each application task.

functions (§2.4), along with all the configurations for these components used in our evaluation.

2.1 General Framework

Formally, given a word w, its word embedding $\mathbf{w}$ can be computed by the composition of its subword embeddings as follows:

$$\mathbf{w} = f_{\mathbf{\Theta}}(\delta(w), \mathbf{W}_s, \mathbf{W}_p), \quad (1)$$

where $\delta(w)$ is a deterministic function that segments w into an ordered sequence of its constituent subword units $S_w = (s_{w_i})_1^n$, with $s_{w_i} \in \mathcal{S}$ being a subword type from the subword vocabulary $\mathcal{S}$ of size $|\mathcal{S}|$. Optionally, some segmentation methods can also generate a sequence of the corresponding morphotactic tags $T_w = (t_{w_i})_1^n$. Alone or together with T_w, S_w is embedded into a sequence of subword representations $\mathbf{S}_w = (\mathbf{s}_{w_i})_1^n$ from the subword embedding matrix $\mathbf{W}_s \in \mathbb{R}^{|\mathcal{S}|\times d}$, where d is the dimensionality of subword embeddings. Another optional step is to obtain a sequence of position embeddings $\mathbf{P}_w = (\mathbf{p}_{w_i})_1^n$: they are taken from the position embedding matrix $\mathbf{W}_p \in \mathbb{R}^{p\times d}$, where p is the maximum number of the unique positions. $\mathbf{P}_w$ can interact with $\mathbf{S}_w$ to compute the final representations for subwords $\mathbf{R}_w = (\mathbf{r}_{w_i})_1^n$ (Vaswani et al., 2017). $f_{\mathbf{\Theta}}$ is a composition function taking $\mathbf{R}_w$ as input and outputting a single vector $\mathbf{w}$ as the word embedding of w.

For the distributional "word-level" training, similar to prior work (Bojanowski et al., 2017), we adopt the standard skip-gram with negative sampling (SGNS) (Mikolov et al., 2013) with bag-of-words contexts. However, we note that other distributional models can also be used under the same framework. Again, following Bojanowski et al. (2017), we calculate the word embedding $\mathbf{w}_t \in \mathbb{R}^d$ for each target word w_t using the formulation from Eq. (1), and parametrize context words with another word embedding matrix $\mathbf{W}_c \in \mathbb{R}^{|\mathcal{V}|\times d}$, where $|\mathcal{V}|$ is the size of word vocabulary $\mathcal{V}$.

2.2 Segmentation of Words into Subwords

We consider three well-known segmentation methods for the function δ, briefly outlined here.

Supervised Morphological Segmentation We use CHIPMUNK (Cotterell et al., 2015) as a representative supervised segmentation system, proven to provide a good trade-off between accuracy and speed.[4] It is based on semi-Markov conditional random fields (Sarawagi and Cohen, 2005). For each word, apart from generating S_w, it also outputs the corresponding morphotactic tags T_w.[5] In §2.3 we discuss how to incorporate information from T_w into subword representations.

δ	δ(*dishonestly*)
CHIPMUNK	(*dis*, *honest*, *ly*) (*prefix*, *root*, *suffix*)
Morfessor	(*dishonest*, *ly*)
BPE	(*dish*, *on*, *est*, *ly*)

Table 1: Segmentations of the word *dishonestly*.

Morfessor Morfessor (Smit et al., 2014) denotes a family of generative probabilistic models for unsupervised morphological segmentation used, among other applications, to learn morphologically-aware word embeddings (Luong et al., 2013).

BPE Byte Pair Encoding (BPE; Gage (1994)) is a simple data compression algorithm. It has become a de facto standard for providing subword information in neural machine translation (Sennrich et al., 2016). The input word is initially split into a sequence of characters, with each unique character denoted as a byte. BPE then iteratively replaces the most common pair of consecutive bytes with a new byte that does not occur within the data, and the number of iterations can be set in advance to control the granularity of the byte combinations.

An example output for all three methods is shown in Table 1. Note that a standard practice in subword-informed models is to also insert the entire word token into S_w (Bojanowski et al., 2017).[6] This is, however, again an optional step and we evaluate configurations with and without the inclusion of the word token in S_w.

2.3 Subword and Position Embeddings

The next step is to encode S_w (or the tuple (S_w, T_w) for CHIPMUNK) to construct a sequence of subword representations $\mathbf{S}_w$. Each row of the subword embedding matrix $\mathbf{W}_s$ is simply defined as the embedding of a unique subword. For CHIPMUNK, we define each row in $\mathbf{W}_s$ as the concatenation of the subword s and its predicted tag t. We also test

---

[4] `http://cistern.cis.lmu.de/chipmunk`

[5] In our experiments, we use only basic information on affixes such as prefixes and suffixes, and leave the integration of fine-grained information such as inflectional and derivational affixes as future work.

[6] We only do the insertion if $|S_w| > 1$. For CHIPMUNK, a generic tag `word` is added to the sequence T_w.

CHIPMUNK configurations without the use of T_w to analyze its contribution.[7]

After generating $\mathbf{S}_w$, an optional step is to have a *learnable* position embedding sequence $\mathbf{P}_w$ further operate on $\mathbf{S}_w$ to encode the order information. Similar to $\mathbf{W}_s$, the definition of the position embedding matrix $\mathbf{W}_p$ also varies: for Morfessor and BPE, we use the absolute positions of subwords in the sequence S_w, whereas for CHIPMUNK morphotactic tags are encoded directly as positions.

Finally, following prior work (Gehring et al., 2017; Mikolov et al., 2018), we use addition and element-wise multiplication between each subword vector $\mathbf{s}$ from $\mathbf{S}_w$ and the corresponding position vector $\mathbf{p}$ from $\mathbf{P}_w$ to compute each entry $\mathbf{r}$ for the final sequence of subword vectors $\mathbf{R}_w$:

$$\mathbf{r} = \mathbf{s} + \mathbf{p} \quad \text{or} \quad \mathbf{r} = \mathbf{s} \odot \mathbf{p}. \tag{2}$$

2.4 Composition Functions

A composition function f_Θ is then applied to the sequence of subword embeddings $\mathbf{R}_w$ to compute the final word embedding $\mathbf{w}$. We investigate three composition functions: 1) addition, 2) single-head and 3) multi-head self-attention (Vaswani et al., 2017; Lin et al., 2017).[8] Addition is used in the original `fastText` model of Bojanowski et al. (2017), and remains a strong baseline for many tasks. However, addition treats each subword with the same importance, ignoring semantic composition and interactions among the word's constituent subwords. Therefore, we propose to use a self-attention mechanism, that is, a learnable weighted addition as the composition function on subword sequences. To the best of our knowledge, we are the first to apply a self-attention mechanism to the problem of subword composition.

Composition Based on Self-Attention Our self-attention mechanism is inspired by Lin et al. (2017). It is essentially a multilayer feed-forward neural network without bias term, which generates a weight matrix for the variable length input $\mathbf{R}_w$:

$$\mathbf{H}_w = tanh(\mathbf{W}_{h_1}\mathbf{R}_w^T) \tag{3}$$

$$\mathbf{A}_w = softmax(\mathbf{W}_{h_2}\mathbf{H}_w) \tag{4}$$

Component	Option	Label
Segmentation	CHIPMUNK	*sms*
	Morfessor	*morf*
	BPE	*bpe*
Morphotactic tag	concatenated with subword (only *sms*)	*st*
Word token	exclusion	*w-*
	inclusion	*w+*
Position embedding	exclusion	*p-*
	additive	*pp*
	multiplicative	*mp*
Composition function	addition	*add*
	single head attention	*att*
	multi-head attention	*mtx*

Table 2: Different components used to construct subword-informed configurations, and their labels.

Each row of $\mathbf{A}_w$ is a weight vector for rows of $\mathbf{R}_w$, which models different aspects of semantic compositions and interactions. For the single-head self-attention, $\mathbf{A}_w$ degenerates to a row vector as the final attention vector $\mathbf{a}_w$. For the multi-head self-attention, we average the rows of $\mathbf{A}_w$ to generate $\mathbf{a}_w$.[9] Finally, $\mathbf{w}$ is computed as the weighted addition of subword embeddings from $\mathbf{R}_w$: $\mathbf{w} = \sum_{w_1}^{w_n} a_{w_i}\mathbf{r}_{w_i}$.

3 Experimental Setup

We train different subword-informed model configurations on 5 languages representing 3 morphological language types: English (EN), German (DE), Finnish (FI), Turkish (TR) and Hebrew (HE), see Table 3. We then evaluate the resulting subword-informed word embeddings in three distinct tasks: 1) general and rare word similarity and relatedness, 2) syntactic parsing, and 3) fine-grained entity typing. The three tasks have been selected in particular as they require different degrees of syntactic and semantic information to be stored in the input word embeddings, ranging from a purely semantic task (word similarity) over a hybrid syntactic-semantic task of entity typing to syntactic parsing.

Subword-Informed Configurations We train a large number of subword-informed configurations by varying the segmentation method δ (§2.2), subword embeddings $\mathbf{W}_s$, the inclusion of position embeddings $\mathbf{W}_p$ and the operations on $\mathbf{W}_s$ (§2.3), and the composition functions f_Θ (§2.4). The configurations are based on the following variations of

---

[7] The extra information on tags should lead to a more expressive model resolving subword ambiguities. For instance, the subword *post* in *postwar* and noun *post* are intrinsically different: the former is the prefix and the later is the root.

[8] Using more complex compositions based on CNNs and RNNs is also possible, but we have not observed improvement in our evaluation tasks with such compositions, which is also in line with findings from recent work (Li et al., 2018).

[9] We have also experimented with adding an extra transformation layer over attention matrix to generate the attention vector, but without any performance gains.

constituent components: (1) For the segmentation δ, we test a supervised morphological system CHIPMUNK (*sms*), Morfessor (*morf*) and BPE (*bpe*). A word token can be optionally inserted into the subword sequence S_w for all three segmentation methods (*ww*) or left out (*w-*); (2) We can only embed the subword s for *morf* and *bpe*, while with *sms* we can optionally embed the concatenation of the subword and its morphotactic tag $s:t$ (*st*);[10] (3) We test subword embedding learning without position embeddings (*p-*), or we integrate them using addition (*pp*) or element-wise multiplication (*mp*); (4) For the composition function function f_Θ, we experiment with addition (*add*), single head self-attention (*att*), and multi-head self-attention (*mtx*). Table 2 provides an overview of all components used to construct a variety of subword-informed configurations used in our evaluation.

The variations of components from Table 2 yield 24 different configurations in total for *sms*, and 18 for *morf* and *bpe*. We use pretrained CHIPMUNK models for all test languages except for Hebrew, as Hebrew lacks gold segmentation data. Following Vania and Lopez (2017), we use the default parameters for Morfessor, and 10k merge operations for BPE across languages. We use available BPE models pre-trained on Wikipedia by Heinzerling and Strube (2018).[11]

Two well-known word representation models, which can also be described by the general framework from Figure 1, are used as insightful baselines: the subword-agnostic SGNS model (Mikolov et al., 2013) and `fastText` (FT)[12] (Bojanowski et al., 2017). FT computes the target word embedding using addition as the composition function, while the segmentation is straightforward: the model simply generates all character n-grams of length 3 to 6 and adds them to S_w along with the full word.

Training Setup Our training data for all languages is Wikipedia. We lowercase all text and replace all digits with a generic tag #. The statistics of the training corpora are provided in Table 3.

Typology	Language	#tokens	#types
Fusional	English (EN)	600M	900K
	German (DE)	200M	940K
Agglutinative	Finnish (FI)	66M	600K
	Turkish (TR)	52M	300K
Introflexive	Hebrew (HE)	90M	410K

Table 3: Statistics of our Wikipedia training corpora. For faster training, we use one third of the entire Wikipedia corpus for EN and DE.

All subword-informed variants are trained on the same data and share the same parameters for the SGNS model.[13] Further, we use ADAGRAD (Duchi et al., 2011) with a linearly decaying learning rate, and do a grid search of learning rate and batch size for each δ on the German[14] WordSim-353 data set (WS; Leviant and Reichart (2015)). The hyperparameters are then fixed for all other languages and evaluation runs. Finally, we set the learning rate to 0.05 for *sms* and *bpe*, and 0.075 for *morf*, and the batch size to 1024 for all the settings.

3.1 Evaluation Tasks

Word Similarity and Relatedness These standard intrinsic evaluation tasks test the semantics of word representations (Pennington et al., 2014; Bojanowski et al., 2017). The evaluations are performed using the Spearman's rank correlation score between the average of human judgement similarity scores for word pairs and the cosine similarity between two word embeddings constituting each word pair. We use Multilingual SimLex-999 (SIMLEX; Hill et al. (2015); Leviant and Reichart (2015); Mrkšić et al. (2017)) for English, German and Hebrew, each containing 999 word pairs annotated for true semantic similarity. We further evaluate embeddings on FinnSim-300 (FS300) produced by Venekoski and Vankka (2017) for Finnish and AnlamVer (AN; Ercan and Yıldız (2018)) for Turkish. We also run experiments on the WordSim-353 test set (WS; Finkelstein et al. (2002)), and its portions oriented towards true similarity (WS-SIM) and broader relatedness (WS-REL) portion for English and German.

Finally, to analyze the importance of subword information for learning embeddings of rare words, we evaluate on the recently released CARD-660 dataset (CARD; Pilehvar et al. (2018)) for English, annotated for true semantic similarity.

---

[10] Once *st* is applied, we do not use position embeddings anymore, because the morphotactic tags are already encoded in subword embeddings, i.e., *st* and *pp* are mutually exclusive.

[11] `https://github.com/bheinzerling/bpemb`

[12] `https://github.com/facebookresearch/fastText`

[13] We rely on the standard choices: 300-dimensional subword and word embeddings, 5 training epochs, the context window size is 5, 5 negative samples, the subsampling rate of 10^{-5}, and the minimum word frequency is 5.

[14] German has moderate morphological complexity among the five languages, so we think the hyperparameters tuned on it could be applicable to other languages.

		Best	2nd Best	Worst	`sgns`	`ft`
EN	WS	**.656** (*sms.w-.st.att*)	.655 (*sms.w-.pp.att*)	.440 (*bpe.w-.mp.mtx*)	.634	.643
	WS-SIM	**.708** (*sms.ww.st.mtx*)	.707 (*sms.w-.st.att*)	.475 (*bpe.w-.mp.att*)	.702	.706
	WS-REL	**.625** (*sms.w-.p-.add*)	.620 (*sms.w-.st.att*)	.438 (*bpe.w-.mp.mtx*)	.579	.586
	SIMLEX	.283 (*sms.ww.p-.add*)	.282 (*morf.w-.p-.add*)	.182 (*bpe.w-.mp.add*)	.300	**.307**
DE	WS	**.633** (*sms.ww.pp.add*)	**.633** (*sms.ww.p-.add*)	.328 (*bpe.w-.mp.mtx*)	.596	.624
	WS-SIM	.673 (*sms.ww.pp.add*)	.668 (*sms.ww.p-.add*)	.363 (*bpe.w-.mp.add*)	.669	**.677**
	WS-REL	**.616** (*sms.ww.p-.add*)	.610 (*sms.ww.pp.add*)	.332 (*bpe.w-.mp.mtx*)	.530	.590
	SIMLEX	**.401** (*sms.ww.p-.add*)	.398 (*sms.ww.pp.add*)	.189 (*bpe.w-.mp.mtx*)	.359	.393
FI	FS300	.259 (*sms.w-.p-.add*)	.258 (*sms.w-.pp.add*)	.123 (*morf.ww.mp.mtx*)	.211	**.279**
TR	AN-SIM	**.355** (*bpe.ww.mp.add*)	.325 (*bpe.ww.mp.att*)	.112 (*bpe.ww.pp.mtx*)	.232	.271
	AN-REL	.459 (*bpe.ww.mp.add*)	.444 (*sms.w-.pp.add*)	.273 (*morf.ww.mp.att*)	.183	**.520**
HE	SIMLEX	.338 (*bpe.ww.pp.add*)	.338 (*bpe.ww.pp.mtx*)	.128 (*bpe.w-.mp.mtx*)	.379	**.388**
EN	CARD	**.370** (*sms.ww.pp.add*)	.328 (*sms.w-.pp.mtx*)	.000 (*bpe.w-.pp.add*)	.009	.249

Table 4: Results on word similarity and relatedness across languages. The highest score for each row is in bold, and we choose randomly in case of a tie. All scores are obtained after computing the embeddings of OOV words.

		Dev set		Test set	
		UAS	LAS	UAS	LAS
EN	*sms.w-.mp.mtx*	**92.3**	90.3	92.0	90.1
	bpe.ww.p-.mtx	**92.3**	**90.4**	**92.1**	90.0
	`sgns`	**92.3**	**90.4**	91.9	89.8
	`ft`	**92.3**	90.3	**92.1**	**90.2**
DE	*bpe.ww.pp.add*	91.2	87.7	**89.6**	**84.7**
	bpe.ww.mp.mtx	91.2	87.7	89.4	**84.7**
	`sgns`	91.4	**87.9**	89.3	84.4
	`ft`	**91.6**	**87.9**	89.1	84.4
FI	*bpe.ww.mp.add*	**89.9**	**86.9**	**90.7**	**87.4**
	sms.w-.pp.add	89.3	86.1	90.5	87.1
	`sgns`	**88.9**	85.6	89.5	86.2
	`ft`	89.7	**86.9**	90.4	87.1
TR	*sms.ww.mp.add*	71.1	**63.5**	72.8	64.7
	sms.w-.mp.att	70.5	62.5	72.7	64.5
	`sgns`	70.5	62.5	72.2	63.5
	`ft`	**71.2**	63.3	**73.1**	**65.1**
HE	*morf.w-.pp.mtx*	92.3	89.5	91.3	88.5
	morf.ww.p-.add	92.3	89.5	91.2	88.5
	`sgns`	92.4	**89.8**	**91.5**	**88.7**
	`ft`	**92.6**	89.7	91.2	88.3

Table 5: Results on the dependency parsing task. The two best configurations are selected according to LAS.

Dependency Parsing Next, we use the syntactic dependency parsing task to analyze the importance of subword information for syntactically-driven downstream applications. For all test languages, we rely on the standard Universal Dependencies treebanks (UD v2.2; Nivre et al. (2016)). We use subword-informed word embeddings from different configurations to initialize the deep biaffine parser of Dozat and Manning (2017) which has shown competitive performance in shared tasks (Dozat et al., 2017) and among other parsing models (Ma and Hovy, 2017; Shi et al., 2017; Ma et al., 2018).[15] We use default settings for the biaffine parser for all experimental runs

Fine-Grained Entity Typing The task is to map entities, which could comprise more than one entity token, to predefined entity types (Yaghoobzadeh and Schütze, 2015). It is a suitable semi-semantic task to test our subword models, as the subwords of entities usually carry some semantic information from which the entity types can be inferred. For example, *Lincolnshire* will belong to `/location/county` as *-shire* is a suffix that strongly indicates a location. We rely on an entity typing dataset of Heinzerling and Strube (2018) built for over 250 languages by obtaining entity mentions from Wikidata (Vrandečić and Krötzsch, 2014) and their associated FIGER-based entity types (Ling and Weld, 2012): there only exists a one-to-one mapping between the entity and one of the 112 FIGER types.

We randomly sample the data to obtain a train/dev/test split with the size of 60k/20k/20k for all languages. For evaluation we extend the RNN-based model of Heinzerling and Strube (2018), where they stacked all the subwords of entity tokens into a flattened sequence: we use the hierarchical embedding composition instead. For each entity token, we first compute its word embeddings with our subword configurations,[16] then feed the word embeddings of entity tokens to a bidirectional LSTM with 2 hidden layers of size 512, followed by a projection layer which predicts the entity type.

---

[15] `https://github.com/tdozat/Parser-v2`

[16] Although it is true that case information can be very important to the task, we conform to Heinzerling and Strube (2018) lowercasing all letters.

4 Results and Analysis

To get a better grasp of the overall performance without overloading the tables, we focus on reporting two best configurations and the worst configuration for each task and language from the total of 60 configurations, except for Hebrew with 36 configurations, where there is no gold segmentation data for training *sms* model. We also analyze the effects of different configurations on different tasks based on language typology. The entire analysis revolves around the key questions Q1-Q3 posed in the introduction. The reader is encouraged to refer to the supplementary material for the complete results.

Tables 4, 5, and 6 summarize the main results on word similarity and relatedness, dependency parsing and entity typing, respectively. In addition, the comparisons of different configurations across tasks and language types are shown in Figure 2 (as well as Figure 3 to 7 in the supplementary material). There, we center the comparison around two crucial components: segmentation and composition. The value in each pixel block is the percentage rank of the row configuration minus that of column configuration. We compute such percentage ranks by performing three levels of averaging over: 1) all related datasets for the same task; 2) all sub-configurations that entail the configuration in question; 3) all languages from the same language types.

Q1. Tasks and Languages Regarding the absolute performance of our subword-informed configurations, we notice that they outperform SGNS and FT in 3/5 languages on average, and for 8/13 datasets on word similarity and relatedness. The gains are more prominent over the subword-agnostic SGNS model and for morphologically richer languages such as Finnish and Turkish. The results on the two other tasks are also very competitive, with strong performance reported especially on the entity typing task. This clearly indicates the importance of integrating subword-level information into word vectors. A finer-grained comparative analysis shows that best-performing configurations vary greatly across different languages.

The comparative analysis across tasks also suggests that there is no single configuration that outperforms the others in all three tasks, although certain patterns in the results emerge. For instance, the supervised segmentation (*sms*) is very useful for word similarity and relatedness (seen in Figure 4). This result is quite intuitive: *sms* is trained according to the readily available gold standard morphological segmentations. However, *sms* is less useful for entity typing, where almost all best-performing configurations are based on *morf* (see also Figure 4). This result is also interpretable: *morf* is a conservative segmenter that captures longer subwords, and is not distracted by short and nonsensical subwords (like *bpe*) that have no contribution to the prediction.[17] The worst configurations on word similarity are based on *bpe*: its aggressive segmentation often results in non-interpretable or nonsensical subwords which are unable to recover useful semantic information. Due to the same reason, the results in all three tasks indicate that the best configurations with *bpe* are always coupled with *ww*, and the worst are obtained with *w-* (i.e., without the inclusion of the full word).

The results on parsing reveal similar performances for a spectrum of heterogeneous configurations. In other words, while the chosen configuration is still important, its impact on performance of state-of-the-art dependency parsers (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) is decreased, as such parsers are heavily parametrized multi-component methods (e.g., besides word embeddings they rely on biLSTMs, intra-sentence attention, character representations).[18] Therefore, a larger space of subword-informed configurations for word representations leads to optimal or near-optimal results. However, *sms* seems to yield highest scores on average in agglutinative languages (see also Figure 2).

Figure 2 clearly demonstrates that, apart from entity typing heatmaps (the third row) which show very similar trends over different language types, the patterns for different tasks and language types tend to vary in general. Similarly, other figures in the supplementary material also show diverging trends across languages representing different language types.

---

[17] For example, *sms* and *bpe* both split "Valberg" (`/location/city`) and "Robert Valberg" (`/person/actor`) with a suffix "berg". Since "berg" could represent both a place or a person, it is not useful alone as a suffix to predict the correct entity type, whereas *morf* does not split the word and makes the prediction solely based on surrounding entity tokens.

[18] We also experimented with removing token features such as POS tags and character embeddings in some settings, but we observed similar trends in the final results.

	Best	2nd Best	Worst	`sgns`	`ft`
EN	**55.70** (*bpe.ww.mp.add*)	55.68 (*morf.w-.pp.att*)	51.15 (*bpe.w-.p-.att*)	51.00	55.15
DE	54.06 (*morf.w-.pp.add*)	54.01 (*morf.w-.pp.att*)	50.21 (*bpe.w-.p-.att*)	50.14	**54.55**
FI	**57.41** (*morf.w-.pp.add*)	57.38 (*morf.w-.pp.mtx*)	52.18 (*bpe.w-.p-.att*)	49.87	57.18
TR	**56.31** (*morf.w-.pp.add*)	56.14 (*morf.w-.pp.mtx*)	46.97 (*bpe.w-.pp.mtx*)	54.35	54.62
HE	**60.34** (*morf.w-.pp.att*)	60.09 (*morf.ww.pp.add*)	51.31 (*bpe.w-.p-.att*)	54.55	59.09

Table 6: Test accuracy, the evaluation metric used by Heinzerling and Strube (2018), on the fine-grained entity typing task. The results are averaged over 5 runs with random seeds.

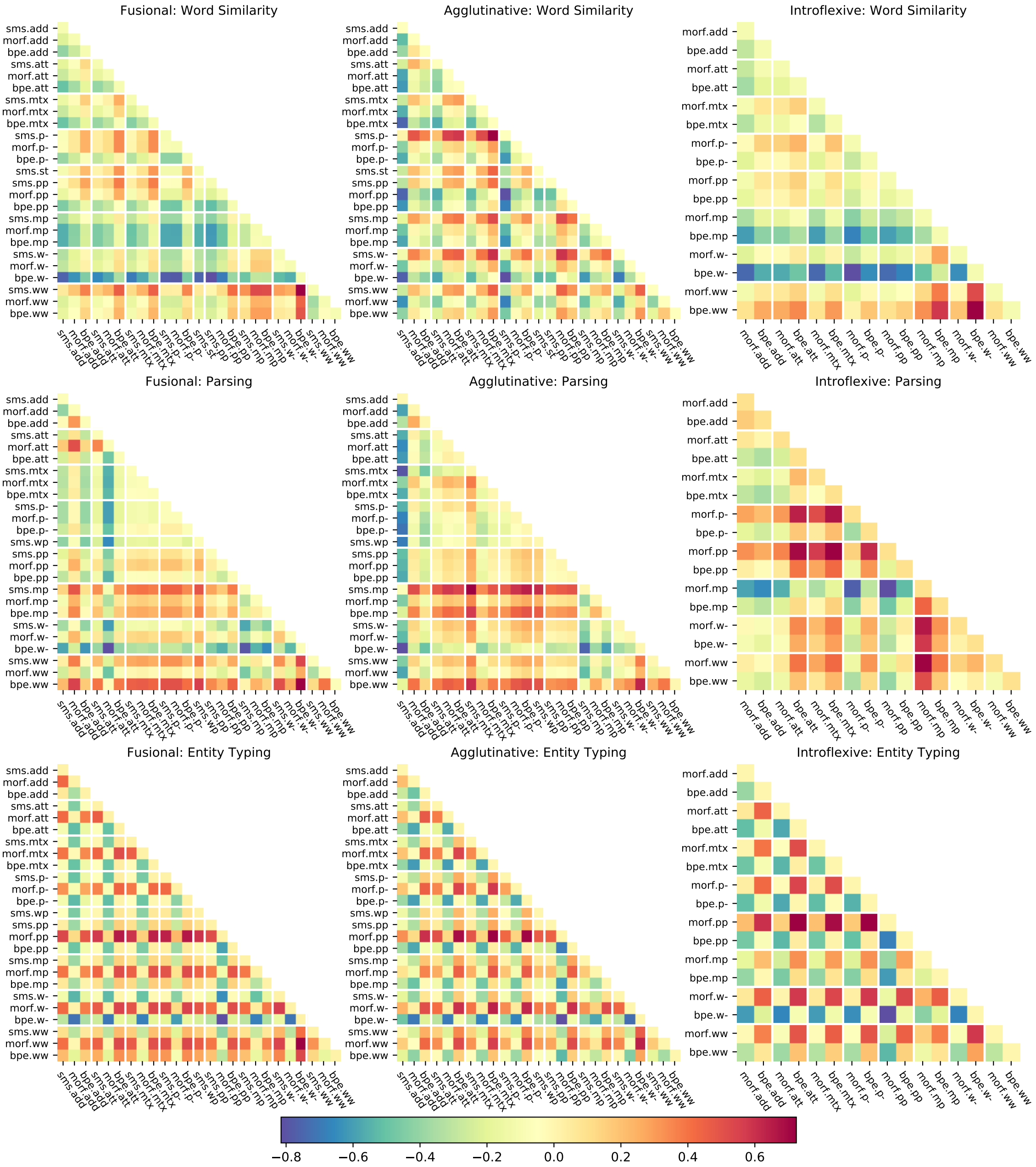


Figure 2: Comparisons of different configurations across tasks grouped by language types. The value in each pixel block is the percentage rank of the row configuration minus that of column configuration. For example, in word similarity for fusional language, the rank for the row *sms.ww* is 0.787 and 0.071 for column *bpe.w-*, and the block value is 0.716. The higher the value, the better the performance of the row compared to the column configuration.

Q2 and Q3. Configurations As mentioned, different tasks reach high scores with different segmentation and composition components. The crucial components for word similarity are *sms* and *ww*, and *sms* is generally better than *morf* and *bpe* in fusional and agglutinative languages (see Figure 4 and 7). The presence of *ww* is desired in this task as also found by Bojanowski et al. (2017): it enhances the information provided by the segmentation. As discussed before, the best configurations with *bpe* are always coupled with *ww*, and the worst with *w-*. *ww* is less important for the more conservative *morf* where the information stored in *ww* can be fully recovered from the generated subwords. Interestingly, *pp* and *mp* do not have positive effects on this task for fusional and introflexive languages, but they seem to resonate well with agglutinative languages, and they are useful for the two other tasks (seen in Figure 6). In general, position embeddings have shown potential benefits in all tasks, where they selectively emphasize or filter subwords according to their positions. *pp* is extremely useful in entity typing for all languages, because it indicates the root position.

Concerning composition functions, *add* still remains an extremely robust choice across tasks and languages. Surprisingly, the more sophisticated self-attention composition prevails only on a handful of datasets: compare the results with *add* vs. *att* and *mtx*. In fact, the worst configurations mostly use *att* and *mtx* (see also Figure 5). In sum, our results suggest that, when unsure, *add* is by far the most robust choice for the subword composition function. Further, morphotactic tags encoded in subword embeddings (*st*) seem to be only effective combined with self-attention in word similarity and relatedness. These findings call for further investigation in future work, along with the inclusion of finer-grained morphotactic tags into the proposed modeling framework.

Further Discussion A recurring theme of this study is that subword-informed configurations are largely task- and language-dependent. We can extract multiple examples from the reported results affirming this conjecture. For instance, in fusional and agglutinative languages *mp* is critical to the model on dependency parsing, while for Hebrew, an introflexive language, *mp* is among the most detrimental components on the same task. Further, for Turkish word similarity *bpe.ww* outperforms *sms.ww*: due to affix concatenation in Turkish, *sms* produces many affixes with only syntactic functions that bring noise to the task. Interestingly, SGNS performs well in Hebrew on parsing and word similarity: it shows that it is still difficult for linear segmentation methods to capture non-concatenative morphology.

Finally, fine-tuning subword-informed representations seems especially beneficial for rare word semantics: our best configuration outperforms FT by 0.111 on CARD, and even surpasses all the state-of-the-art models on the rare word similarity task, as reported by Pilehvar et al. (2018). We hope that our findings on the CARD dataset will motivate further work on building more accurate representations for rare and unseen words (Bhatia et al., 2016; Herbelot and Baroni, 2017; Schick and Schütze, 2018) by learning more effective and more informed components of subword-informed configurations.

5 Conclusion

We have presented a general framework for learning subword-informed word representations which has been used to perform a systematic analysis of 60 different subword-aware configurations for 5 typologically diverse languages across 3 diverse tasks. The large space of presented results has allowed us to analyze the main properties of subword-informed representation learning: we have demonstrated that different components of the framework such as segmentation and composition methods, or the use of position embeddings, have to be carefully tuned to yield improved performance across different tasks and languages. We hope that this study will guide the development of new subword-informed word representation architectures. Code is available at: `https://github.com/cambridgeltl/sw_study`.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (no 648909). We thank the reviewers for their insightful comments, and Roi Reichart for many fruitful discussions.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the ACL*, 4:431–444.

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. In *Proceedings of EACL*, pages 422–426.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of EMNLP*, pages 490–500.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of EMNLP*, pages 3285–3295.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Ryan Cotterell, Thomas Müller, Alexander M. Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-Markov models. In *Proceedings of CoNLL*, pages 164–174.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of NAACL*, pages 1287–1292.

Ryan Cotterell and Hinrich Schütze. 2018. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the ACL*, 6:33–48.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gökhan Ercan and Olcay Taner Yıldız. 2018. Anlamver: Semantic model evaluation dataset for turkish - word similarity and relatedness. In *Proceedings of COLING*, pages 3819–3836.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of ICML*, pages 1243–1252.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of EMNLP*, pages 316–327.

Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of LREC*.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of EMNLP*, pages 304–309.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*, pages 12–22.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transaction of the ACL*, 4:313–327.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of ACL*, pages 66–75.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of ACL*, pages 1517–1526.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the NIPS*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 38–48.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR*.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of AAAI*, pages 94–100.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113.

Xuezhe Ma and Eduard H. Hovy. 2017. Neural probabilistic model for non-projective MST parsing. In *Proceedings of IJCNLP*, pages 59–69.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of ACL*, pages 1403–1414.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of LREC*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gasic, Anna Korhonen, and Steve J. Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the ACL*, 5:309–324.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proccedings of NAACL*, pages 2227–2237.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: A reliable evaluation framework for rare word representation models. In *Proceedings of EMNLP*, pages 1391–1401.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of EMNLP*, pages 102–112.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING*, pages 141–150.

Sunita Sarawagi and William W Cohen. 2005. Semi-Markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Timo Schick and Hinrich Schütze. 2018. Learning semantic representations for novel words: Leveraging both form and context. *CoRR*, abs/1811.03866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 86–96.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of EMNLP*, pages 12–23.

Peter Smit, Sami Virpioja, Stig-Arne Grnroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of EACL*, pages 21–24.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of ACL*, pages 2016–2027.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Viljami Venekoski and Jouko Vankka. 2017. Finnish resources for evaluating language model semantics. In *Proceedings of NoDaLiDa*, pages 231–236.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of EMNLP*, pages 1504–1515.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of EMNLP*, pages 715–725.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of EMNLP*, pages 601–606.

A Supplemental Material

In this supplementary material we present the full results of different configurations of our subword-informed representations and baseline models in three tasks across five languages. For details and notations of different model configurations, please refer to the original paper.

Since `fastText` (FT) package cannot generate each subword embedding for a given word, we also implemented our own version (`our_ft`) within the same subword-informed training framework. We trained `our_ft` on all the five languages with the same training corpus, and kept the same hyperparameters as FT package except that we used 1024 as batch size for faster training. We show that our implementation yields comparable results with the original FT in word similarity and relatedness and parsing across languages. For entity typing, we experimented with the extended hierarchical composition architecture described in the paper on `our_ft`, i.e., using subword embeddings to compute word representations and updating subword embeddings of `our_ft` during training, and only use word embeddings as input for SGNS and FT and directly update them. Additionally, we also experimented with some of our model configurations with character n-grams as our segmentation methods, but did not observe performance gains. Due to its large parameter space (usually 2-4 times larger than SGNS) and the computational limits, we leave it as our future work.

Table 7, 8 and 9 show the results of word similarity and relatedness based on different segmentation methods, and Table 10, 11 and 12 for the results of dependency parsing. Table 13 shows the accuracy on development and test set for entity typing across all five languages. Figure 3 shows the configuration comparisons in languages with available datasets for word relatedness. Figure 4 to 7 show the results on all downstream tasks across languages when each test data point is categorized according to some specific component. Specifically, Figure 4 shows the results when only different segmentation methods are considered, and Figure 5, 6 and 7 show the results on varying composition functions, position embedding types and whether including the word token embedding, respectively.

	SIMLEX	WS	WS-REL	WS-SIM	FS300	AN-REL	AN-SIM	CARD
sms.w-.p-.add	0.267(EN)	0.650	0.625	0.688	0.259(FI)	0.277(TR)	0.432(TR)	0.206(EN)
	0.350(DE)	0.517	0.490	0.586				
sms.w-.st.add	0.267	0.654	0.619	0.704	0.228	0.276	0.434	0.208
	0.350	0.509	0.475	0.577				
sms.w-.pp.add	0.270	0.645	0.615	0.693	0.258	0.261	0.444	0.294
	0.332	0.494	0.467	0.563				
sms.w-.mp.add	0.207	0.635	0.598	0.683	0.247	0.269	0.424	0.213
	0.295	0.437	0.401	0.507				
sms.ww.p-.add	0.283	0.640	0.589	0.696	0.226	0.273	0.435	0.267
	0.401	0.633	0.616	0.668				
sms.ww.st.add	0.277	0.637	0.576	0.697	0.204	0.239	0.404	0.269
	0.385	0.622	0.601	0.658				
sms.ww.pp.add	0.282	0.632	0.572	0.702	0.240	0.240	0.423	0.370
	0.398	0.633	0.610	0.673				
sms.ww.mp.add	0.240	0.633	0.579	0.696	0.211	0.279	0.443	0.268
	0.353	0.599	0.557	0.639				
sms.w-.p-.att	0.250	0.642	0.613	0.684	0.228	0.289	0.416	0.244
	0.339	0.518	0.478	0.598				
sms.w-.st.att	0.255	0.656	0.620	0.697	0.227	0.263	0.358	0.236
	0.327	0.512	0.468	0.583				
sms.w-.pp.att	0.252	0.655	0.619	0.707	0.227	0.212	0.334	0.323
	0.324	0.500	0.456	0.570				
sms.w-.mp.att	0.206	0.629	0.593	0.674	0.205	0.263	0.405	0.219
	0.290	0.452	0.425	0.517				
sms.ww.p-.att	0.277	0.642	0.591	0.698	0.192	0.311	0.440	0.251
	0.360	0.605	0.571	0.640				
sms.ww.st.att	0.261	0.632	0.570	0.693	0.212	0.243	0.366	0.260
	0.342	0.588	0.549	0.625				
sms.ww.pp.att	0.271	0.630	0.584	0.695	0.228	0.204	0.367	0.310
	0.357	0.608	0.567	0.660				
sms.ww.mp.att	0.220	0.616	0.547	0.682	0.191	0.241	0.376	0.222
	0.340	0.569	0.526	0.613				
sms.w-.p-.mtx	0.245	0.639	0.602	0.694	0.248	0.260	0.404	0.220
	0.336	0.505	0.457	0.585				
sms.w-.st.mtx	0.252	0.648	0.617	0.702	0.234	0.231	0.381	0.224
	0.330	0.506	0.456	0.578				
sms.w-.pp.mtx	0.260	0.638	0.606	0.694	0.244	0.184	0.335	0.328
	0.331	0.488	0.429	0.564				
sms.w-.mp.mtx	0.211	0.620	0.599	0.671	0.228	0.265	0.417	0.226
	0.282	0.447	0.417	0.506				
sms.ww.p-.mtx	0.279	0.631	0.575	0.696	0.206	0.280	0.426	0.251
	0.357	0.615	0.578	0.657				
sms.ww.st.mtx	0.277	0.642	0.581	0.708	0.192	0.231	0.394	0.261
	0.355	0.603	0.561	0.645				
sms.ww.pp.mtx	0.273	0.640	0.588	0.705	0.226	0.210	0.368	0.300
	0.357	0.611	0.575	0.662				
sms.ww.mp.mtx	0.253	0.622	0.565	0.684	0.183	0.275	0.414	0.260
	0.339	0.598	0.578	0.632				
`sgns`	0.300	0.634	0.579	0.702	0.211	0.232	0.183	0.009
	0.359	0.596	0.530	0.669				
`ft`	0.307	0.643	0.586	0.706	0.279	0.271	0.520	0.249
	0.393	0.624	0.590	0.677				
`our_ft`	0.265	0.629	0.574	0.702	0.241	0.295	0.523	0.281
	0.380	0.610	0.596	0.649				

Table 7: Results on word similarity and relatedness across languages for CHIPMUNK (*sms*). All scores are obtained after computing the embeddings of OOV words.

	SIMLEX	WS	WS-REL	WS-SIM	FS300	AN-REL	AN-SIM	CARD
morf.w-.p-.add	0.282(EN) 0.368(DE) 0.322(HE)	0.629 0.577	0.574 0.557	0.691 0.616	0.166(FI)	0.273(TR)	0.359(TR)	0.134(EN)
morf.w-.pp.add	0.279 0.349 0.317	0.617 0.549	0.562 0.530	0.680 0.594	0.189	0.232	0.375	0.172
morf.w-.mp.add	0.245 0.303 0.324	0.597 0.484	0.529 0.439	0.664 0.550	0.185	0.302	0.351	0.127
morf.ww.p-.add	0.273 0.376 0.326	0.632 0.583	0.573 0.569	0.697 0.614	0.159	0.264	0.339	0.198
morf.ww.pp.add	0.277 0.361 0.324	0.616 0.573	0.557 0.549	0.684 0.609	0.182	0.232	0.375	0.262
morf.ww.mp.add	0.244 0.331 0.304	0.589 0.515	0.520 0.484	0.666 0.574	0.149	0.267	0.300	0.099
morf.w-.p-.att	0.281 0.354 0.323	0.623 0.560	0.565 0.536	0.682 0.603	0.159	0.295	0.362	0.132
morf.w-.pp.att	0.276 0.352 0.308	0.619 0.556	0.555 0.537	0.682 0.597	0.170	0.235	0.358	0.138
morf.w-.mp.att	0.246 0.310 0.291	0.586 0.495	0.519 0.453	0.661 0.548	0.158	0.274	0.304	0.107
morf.ww.p-.att	0.271 0.372 0.321	0.622 0.574	0.568 0.545	0.686 0.621	0.167	0.292	0.346	0.218
morf.ww.pp.att	0.274 0.364 0.319	0.613 0.565	0.549 0.537	0.680 0.601	0.167	0.145	0.312	0.206
morf.ww.mp.att	0.234 0.327 0.307	0.586 0.499	0.514 0.469	0.666 0.562	0.147	0.270	0.273	0.109
morf.w-.p-.mtx	0.280 0.348 0.320	0.632 0.564	0.573 0.544	0.693 0.602	0.133	0.300	0.357	0.132
morf.w-.pp.mtx	0.275 0.355 0.325	0.621 0.549	0.562 0.530	0.693 0.582	0.186	0.231	0.369	0.135
morf.w-.mp.mtx	0.254 0.295 0.301	0.602 0.488	0.536 0.427	0.671 0.542	0.205	0.235	0.295	0.130
morf.ww.p-.mtx	0.275 0.371 0.325	0.624 0.577	0.558 0.545	0.688 0.611	0.177	0.292	0.356	0.222
morf.ww.pp.mtx	0.272 0.359 0.326	0.617 0.592	0.558 0.575	0.684 0.613	0.160	0.151	0.321	0.080
morf.ww.mp.mtx	0.238 0.320 0.318	0.591 0.500	0.522 0.465	0.674 0.559	0.123	0.279	0.326	0.080
`sgns`	0.300 0.359 0.379	0.634 0.596	0.579 0.530	0.702 0.669	0.211	0.232	0.183	0.009
`ft`	0.307 0.393 0.388	0.643 0.624	0.586 0.590	0.706 0.677	0.279	0.271	0.520	0.249
`our_ft`	0.265 0.380 0.354	0.629 0.610	0.574 0.596	0.702 0.649	0.241	0.295	0.523	0.281

Table 8: Results on word similarity and relatedness across languages for Morfessor (*morf*). All scores are obtained after computing the embeddings of OOV words.

	SIMLEX	WS	WS-REL	WS-SIM	FS300	AN-REL	AN-SIM	CARD
bpe.w-.p-.add	0.209(EN) 0.229(DE) 0.156(HE)	0.488 0.416	0.499 0.442	0.508 0.474	0.177(FI)	0.228(TR)	0.390(TR)	0.011(EN)
bpe.w-.pp.add	0.209 0.225 0.162	0.474 0.404	0.484 0.417	0.490 0.467	0.170	0.229	0.407	0.000
bpe.w-.mp.add	0.182 0.193 0.132	0.460 0.331	0.478 0.351	0.484 0.363	0.157	0.283	0.406	0.021
bpe.ww.p-.add	0.274 0.378 0.331	0.642 0.597	0.577 0.582	0.706 0.631	0.203	0.293	0.429	0.262
bpe.ww.pp.add	0.265 0.365 0.338	0.631 0.615	0.571 0.595	0.689 0.637	0.245	0.267	0.429	0.273
bpe.ww.mp.add	0.236 0.348 0.321	0.587 0.566	0.519 0.538	0.658 0.608	0.147	0.355	0.459	0.240
bpe.w-.p-.att	0.216 0.233 0.155	0.506 0.386	0.504 0.409	0.529 0.422	0.200	0.176	0.288	0.010
bpe.w-.pp.att	0.210 0.226 0.152	0.503 0.383	0.507 0.396	0.528 0.428	0.190	0.183	0.331	0.027
bpe.w-.mp.att	0.197 0.209 0.151	0.456 0.398	0.462 0.422	0.475 0.467	0.173	0.200	0.308	0.035
bpe.ww.p-.att	0.265 0.350 0.334	0.621 0.555	0.566 0.525	0.687 0.591	0.197	0.246	0.358	0.257
bpe.ww.pp.att	0.270 0.333 0.334	0.618 0.567	0.570 0.536	0.678 0.604	0.187	0.197	0.314	0.228
bpe.ww.mp.att	0.255 0.338 0.307	0.613 0.545	0.564 0.526	0.681 0.584	0.147	0.325	0.419	0.226
bpe.w-.p-.mtx	0.202 0.224 0.169	0.486 0.394	0.485 0.414	0.512 0.433	0.184	0.174	0.293	0.015
bpe.w-.pp.mtx	0.198 0.223 0.167	0.504 0.399	0.500 0.396	0.547 0.462	0.179	0.195	0.355	0.045
bpe.w-.mp.mtx	0.185 0.189 0.128	0.440 0.328	0.438 0.332	0.476 0.372	0.144	0.176	0.282	0.022
bpe.ww.p-.mtx	0.267 0.354 0.337	0.624 0.573	0.565 0.543	0.690 0.604	0.164	0.238	0.334	0.272
bpe.ww.pp.mtx	0.263 0.337 0.338	0.621 0.553	0.569 0.500	0.677 0.608	0.193	0.112	0.290	0.210
bpe.ww.mp.mtx	0.260 0.336 0.298	0.620 0.546	0.564 0.514	0.681 0.596	0.198	0.257	0.369	0.247
`sgns`	0.300 0.359 0.379	0.634 0.596	0.579 0.530	0.702 0.669	0.211	0.232	0.183	0.009
`ft`	0.307 0.393 0.388	0.643 0.624	0.586 0.590	0.706 0.677	0.279	0.271	0.520	0.249
`our_ft`	0.265 0.380 0.354	0.629 0.610	0.574 0.596	0.702 0.649	0.241	0.295	0.523	0.281

Table 9: Results on word similarity and relatedness across languages for BPE (*bpe*). All scores are obtained after computing the embeddings of OOV words.

	DEV(EN)	TEST	DEV(DE)	TEST	DEV(FI)	TEST	DEV(TR)	TEST
sms.w-.p-.add	92.2(UAS)	91.5	91.3	89.0	88.8	89.6	70.0	71.9
	90.3(LAS)	89.6	87.7	84.2	85.5	86.1	62.1	63.7
sms.w-.st.add	92.4	91.7	91.1	88.7	89.1	89.7	71.0	72.0
	90.5	89.7	87.6	83.8	86.0	86.6	62.9	63.8
sms.w-.pp.add	92.2	91.9	91.4	88.7	89.3	90.5	70.9	72.7
	90.2	89.9	87.8	84.0	86.1	87.1	62.8	64.4
sms.w-.mp.add	92.2	91.9	91.1	89.1	89.0	89.9	70.5	72.6
	90.3	90.0	87.6	84.3	85.8	86.6	62.6	64.5
sms.ww.p-.add	92.2	92.1	91.6	89.1	89.4	90.1	71.0	71.8
	90.2	90.0	88.1	84.3	86.4	86.9	62.8	63.6
sms.ww.st.add	92.4	91.9	91.3	89.2	89.4	89.9	71.0	72.0
	90.5	89.8	87.9	84.5	86.2	86.5	63.0	64.0
sms.ww.pp.add	92.5	91.9	91.3	88.9	89.5	90.3	70.7	72.5
	90.5	90.0	87.8	84.2	86.5	87.0	63.0	64.4
sms.ww.mp.add	92.2	91.7	91.4	89.2	89.4	90.3	71.1	72.8
	90.3	89.6	87.8	84.4	86.3	87.1	63.5	64.7
sms.w-.p-.att	92.1	91.7	91.3	88.9	89.0	89.6	71.3	72.4
	90.2	89.6	87.7	83.9	85.9	86.4	63.1	63.8
sms.w-.st.att	92.2	91.6	91.4	89.0	88.8	89.4	70.0	71.9
	90.3	89.6	87.7	84.2	85.6	86.0	62.2	63.4
sms.w-.pp.att	91.4	91.2	91.1	89.0	89.2	89.7	70.7	71.4
	89.4	89.0	87.6	84.3	85.7	86.1	62.6	63.0
sms.w-.mp.att	92.1	91.8	91.3	89.0	89.0	90.0	70.5	72.7
	90.2	89.8	87.7	84.1	85.9	86.6	62.5	64.5
sms.ww.p-.att	92.1	91.5	91.2	89.3	89.2	89.8	71.2	72.3
	90.2	89.6	87.7	84.5	85.9	86.4	63.2	64.0
sms.ww.st.att	92.3	91.9	91.3	89.2	88.7	89.7	70.4	72.2
	90.4	89.9	87.8	84.3	85.5	86.2	62.3	63.8
sms.ww.pp.att	92.2	91.8	91.5	89.0	88.5	89.3	71.0	72.2
	90.3	89.8	87.9	84.4	84.9	85.8	62.7	63.7
sms.ww.mp.att	92.3	91.9	91.5	89.1	88.6	89.5	70.7	72.5
	90.5	90.0	88.0	84.3	85.4	86.0	62.9	64.3
sms.w-.p-.mtx	92.0	91.9	91.5	88.6	89.0	89.7	70.5	71.7
	90.1	89.8	87.9	83.7	85.9	86.2	62.5	63.5
sms.w-.st.mtx	92.1	91.8	91.1	88.5	89.0	89.5	69.8	71.8
	90.2	89.7	87.8	83.5	85.6	85.9	61.8	63.3
sms.w-.pp.mtx	92.3	91.7	91.2	89.1	88.9	89.6	70.7	72.0
	90.4	89.7	87.6	84.2	85.9	86.3	62.6	63.1
sms.w-.mp.mtx	92.3	92.0	91.4	89.1	89.4	90.2	70.7	72.4
	90.3	90.1	87.8	84.3	86.0	86.9	62.5	64.2
sms.ww.p-.mtx	92.2	91.6	91.4	89.1	89.2	89.8	70.5	71.9
	90.4	89.6	87.9	84.3	86.0	86.4	62.5	63.6
sms.ww.st.mtx	92.2	91.6	91.4	88.7	89.1	89.8	71.0	71.8
	90.4	89.7	87.8	83.9	85.8	86.5	62.7	63.3
sms.ww.pp.mtx	92.3	92.0	91.2	89.0	88.7	89.4	70.4	71.8
	90.4	89.9	87.7	84.2	85.2	85.8	62.7	63.6
sms.ww.mp.mtx	92.3	92.0	91.4	88.9	88.8	89.6	71.0	71.5
	90.4	89.9	87.8	84.0	85.2	85.9	62.7	63.3
`sgns`	92.3	91.9	91.4	89.3	88.9	89.5	70.5	72.2
	90.4	89.8	87.9	84.4	85.6	86.2	62.5	63.5
`ft`	92.3	92.1	91.6	89.1	89.7	90.4	71.2	73.1
	90.3	90.2	87.9	84.4	86.9	87.1	63.3	65.1
`our_ft`	92.7	92.2	91.6	89.6	90.5	90.9	71.5	73.0
	90.7	90.2	87.9	84.9	87.6	87.8	63.5	64.9

Table 10: Results on dependency parsing across languages for CHIPMUNK (*sms*).

	DEV(EN)	TEST	DEV(DE)	TEST	DEV(FI)	TEST	DEV(TR)	TEST	DEV(HE)	TEST
morf.w-.p-.add	92.3	91.7	91.1	89.0	89.0	90.0	70.7	71.8	92.5	91.0
	90.4	89.8	87.7	84.2	85.8	86.6	62.4	63.3	89.6	88.3
morf.w-.pp.add	92.5	91.8	91.4	89.0	89.2	89.8	70.3	71.3	92.3	91.2
	90.6	89.8	87.6	84.1	86.0	86.4	62.2	63.1	89.6	88.4
morf.w-.mp.add	92.2	91.9	91.4	89.0	89.2	89.8	71.2	71.7	92.1	90.8
	90.2	90.0	87.8	84.1	85.9	86.4	63.2	63.6	89.2	88.0
morf.ww.p-.add	92.2	91.8	91.4	89.0	89.4	89.6	70.8	72.2	92.3	91.2
	90.3	89.7	87.8	84.1	86.2	86.2	62.9	64.2	89.5	88.5
morf.ww.pp.add	92.3	91.7	91.3	88.9	88.8	89.9	70.6	71.9	92.2	91.0
	90.4	89.8	87.6	84.1	85.7	86.5	62.5	63.4	89.6	88.1
morf.ww.mp.add	92.3	91.7	91.2	89.0	89.3	89.8	70.5	72.4	92.1	90.9
	90.4	89.6	87.6	84.2	86.0	86.4	62.8	64.2	89.2	88.0
morf.w-.p-.att	92.2	91.8	91.2	89.1	89.1	90.2	70.3	72.1	92.3	91.1
	90.3	89.7	87.6	83.9	85.9	86.6	61.9	63.6	89.6	88.3
morf.w-.pp.att	92.3	92.0	91.5	89.3	89.1	89.7	71.0	72.2	92.6	91.2
	90.3	89.9	87.9	84.6	85.9	86.4	63.1	63.6	89.7	88.4
morf.w-.mp.att	92.0	91.8	91.4	89.1	89.1	89.9	70.2	71.5	92.1	90.3
	90.0	89.8	87.7	84.3	85.7	86.4	62.1	62.8	89.2	87.4
morf.ww.p-.att	92.2	91.9	91.4	89.1	89.4	89.8	70.0	71.2	92.2	91.1
	90.2	89.8	88.0	84.2	86.0	86.5	62.2	62.8	89.3	88.2
morf.ww.pp.att	92.3	91.9	91.6	89.3	89.0	89.6	70.9	72.0	92.2	91.1
	90.3	89.9	88.0	84.5	85.8	86.3	62.7	63.4	89.6	88.3
morf.ww.mp.att	92.2	91.9	91.3	89.0	88.9	90.0	70.9	71.8	92.3	91.0
	90.4	90.0	87.9	84.1	85.8	86.7	62.8	63.4	89.4	88.0
morf.w-.p-.mtx	92.1	91.9	91.4	88.9	89.3	89.8	70.6	71.9	92.2	90.6
	90.1	89.8	87.8	84.1	86.1	86.4	62.8	63.6	89.4	87.9
morf.w-.pp.mtx	92.4	91.8	91.3	88.8	89.3	90.0	71.0	72.2	92.3	91.3
	90.4	89.9	87.7	84.0	86.1	86.7	63.1	63.7	89.5	88.5
morf.w-.mp.mtx	92.2	92.1	91.4	88.8	89.2	90.3	70.2	72.2	92.2	90.5
	90.3	90.0	87.9	83.9	85.9	86.8	62.2	63.7	89.3	87.7
morf.ww.p-.mtx	92.2	91.8	91.3	88.9	89.3	89.6	70.1	71.4	92.5	91.4
	90.3	89.8	87.9	84.0	85.9	86.1	62.1	63.3	89.6	88.4
morf.ww.pp.mtx	92.2	91.7	91.4	89.0	89.2	89.9	71.3	72.1	92.6	90.8
	90.3	89.6	87.9	84.1	86.0	86.7	62.9	63.8	89.7	88.0
morf.ww.mp.mtx	92.3	91.8	91.2	89.1	89.5	89.8	71.1	72.2	92.1	90.8
	90.4	89.9	87.7	84.2	86.3	86.4	62.5	63.8	89.3	88.0
`sgns`	92.3	91.9	91.4	89.3	88.9	89.5	70.5	72.2	92.4	91.5
	90.4	89.8	87.9	84.4	85.6	86.2	62.5	63.5	89.8	88.7
`ft`	92.3	92.1	91.6	89.1	89.7	90.4	71.2	73.1	92.6	91.2
	90.3	90.2	87.9	84.4	86.9	87.1	63.3	65.1	89.7	88.3
`our_ft`	92.7	92.2	91.6	89.6	90.5	90.9	71.5	73.0	92.52	91.6
	90.7	90.2	87.9	84.9	87.6	87.8	63.5	64.9	89.72	88.8

Table 11: Results on dependency parsing across languages for Morfessor (*morf*).

	DEV(EN)	TEST	DEV(DE)	TEST	DEV(FI)	TEST	DEV(TR)	TEST	DEV(HE)	TEST
bpe.w-.p-.add	92.3	91.5	91.1	89.1	89.3	89.9	70.2	72.1	91.9	91.0
	90.3	89.6	87.6	84.2	85.9	86.4	62.6	63.9	89.2	88.2
bpe.w-.pp.add	92.2	91.6	91.5	89.1	89.0	90.0	71.2	71.6	92.0	90.9
	90.2	89.5	87.9	84.3	85.8	86.7	63.2	63.2	89.1	88.1
bpe.w-.mp.add	91.9	91.6	91.5	89.2	89.3	89.8	70.7	71.6	92.4	91.2
	90.0	89.7	87.9	84.3	86.1	86.3	62.7	63.4	89.7	88.2
bpe.ww.p-.add	92.1	91.8	91.3	89.2	89.5	89.8	70.2	71.4	92.5	91.1
	90.2	89.9	87.7	84.4	86.2	86.5	62.1	63.2	89.5	88.2
bpe.ww.pp.add	92.3	91.6	91.2	89.6	89.2	90.2	70.8	72.3	92.3	91.2
	90.4	89.6	87.7	84.7	86.2	87.0	62.9	64.2	89.4	88.4
bpe.ww.mp.add	92.3	91.9	91.6	89.2	89.9	90.7	71.2	72.3	92.2	91.0
	90.3	89.9	88.1	84.5	86.9	87.4	63.1	64.0	89.4	88.2
bpe.w-.p-.att	92.3	91.3	91.4	89.1	88.8	89.4	70.4	71.9	92.0	90.9
	90.3	89.4	87.9	84.2	85.3	85.7	62.3	63.0	89.1	87.9
bpe.w-.pp.att	92.1	91.6	91.4	88.6	88.7	89.5	70.9	71.5	91.9	91.0
	90.2	89.7	87.6	83.7	85.4	86.0	62.5	63.3	89.0	88.3
bpe.w-.mp.att	92.1	91.5	91.5	89.3	89.1	89.9	70.5	72.4	92.2	90.8
	90.2	89.6	87.9	84.4	85.8	86.3	62.5	63.9	89.5	88.0
bpe.ww.p-.att	92.3	92.0	91.4	88.6	89.1	90.1	70.8	71.3	92.0	90.8
	90.4	89.9	87.9	83.8	85.8	86.8	62.4	63.1	89.3	88.0
bpe.ww.pp.att	92.3	91.8	91.4	89.2	89.2	90.0	71.0	72.1	92.1	91.1
	90.5	89.8	87.7	84.6	85.9	86.4	63.2	64.0	89.5	88.2
bpe.ww.mp.att	92.5	92.0	91.5	89.3	89.5	90.3	70.6	71.7	91.9	90.9
	90.6	89.9	88.1	83.9	86.4	86.9	62.6	63.4	89.1	88.0
bpe.w-.p-.mtx	91.9	91.7	91.1	88.9	88.6	89.5	69.9	71.5	91.5	91.2
	90.1	89.7	87.5	83.9	85.2	85.9	61.9	63.1	88.7	88.3
bpe.w-.pp.mtx	92.1	91.7	91.1	88.8	89.0	89.5	70.7	71.8	92.0	91.0
	90.2	89.7	87.5	84.0	85.6	86.0	62.8	63.3	89.2	88.2
bpe.w-.mp.mtx	92.1	91.6	91.1	89.1	89.0	89.9	70.3	72.0	92.2	90.6
	90.2	89.5	87.6	84.1	85.7	86.3	62.6	63.6	89.3	87.7
bpe.ww.p-.mtx	92.3	92.1	91.6	89.0	89.3	90.0	71.7	72.3	91.9	90.5
	90.4	90.0	88.0	84.2	86.0	86.7	63.4	63.7	88.9	87.7
bpe.ww.pp.mtx	92.4	91.9	91.3	89.1	89.3	89.7	69.8	72.2	92.0	90.7
	90.4	89.8	87.7	84.2	85.9	86.2	62.0	63.8	89.2	87.9
bpe.ww.mp.mtx	92.3	91.7	91.2	89.4	89.8	90.2	71.2	72.3	92.2	91.0
	90.4	89.6	87.7	84.7	86.7	86.8	63.1	64.2	89.4	88.1
`sgns`	92.3	91.9	91.4	89.3	88.9	89.5	70.5	72.2	92.4	91.5
	90.4	89.8	87.9	84.4	85.6	86.2	62.5	63.5	89.8	88.7
`ft`	92.3	92.1	91.6	89.1	89.7	90.4	71.2	73.1	92.6	91.2
	90.3	90.2	87.9	84.4	86.9	87.1	63.3	65.1	89.7	88.3
`our_ft`	92.7	92.2	91.6	89.6	90.5	90.9	71.5	73.0	92.52	91.6
	90.7	90.2	87.9	84.9	87.6	87.8	63.5	64.9	89.72	88.8

Table 12: Results on dependency parsing across languages for BPE (*bpe*).

	DEV(EN)	TEST	DEV(DE)	TEST	DEV(FI)	TEST	DEV(TR)	TEST	DEV(HE)	TEST
sms.w-.p-.add	52.88	52.57	51.09	50.95	55.15	55.08	53.87	53.52		
sms.w-.st.add	52.68	52.7	51.2	51.23	55.38	55.13	54.03	53.68		
sms.w-.pp.add	53.19	52.94	50.48	50.83	55.44	55.30	54.05	53.88		
sms.w-.mp.add	52.87	52.76	51.1	51.11	55.18	55.00	54.24	53.96		
sms.ww.p-.add	53.56	53.69	51.73	51.77	55.92	55.84	55.01	54.76		
sms.ww.st.add	54.21	54.26	52.53	52.63	56.21	55.95	55.96	55.43		
sms.ww.pp.add	54.51	54.66	52.98	53.04	56.06	56.00	55.92	55.75		
sms.ww.mp.add	54.27	54.43	52.77	52.88	56.04	55.71	56.38	55.99		
sms.w-.p-.att	52.81	52.7	50.7	50.8	54.85	54.67	53.62	53.45		
sms.w-.st.att	52.57	52.64	51.64	51.65	54.87	54.69	53.27	53.14		
sms.w-.pp.att	52.94	52.66	52.1	52.5	54.91	54.83	53.39	53.21		
sms.w-.mp.att	52.99	52.91	51.7	51.9	55.02	55.03	54.06	53.72		
sms.ww.p-.att	53.68	53.67	51.48	51.69	55.59	55.73	55.17	54.92		
sms.ww.st.att	53.55	53.66	51.64	51.65	55.27	55.35	53.84	53.26		
sms.ww.pp.att	53.14	53.08	52.1	52.5	54.66	54.52	54.95	54.77		
sms.ww.mp.att	53.82	53.79	51.67	51.89	54.02	54.04	54.1	53.76		
sms.w-.p-.mtx	52.7	52.67	50.29	50.52	54.61	54.51	53.44	53.24		
sms.w-.st.mtx	52.75	52.61	50.42	50.85	54.15	54.09	53.44	52.84		
sms.w-.pp.mtx	52.89	52.87	51.07	51.38	54.58	54.70	53.42	53.16		
sms.w-.mp.mtx	53.03	52.89	50.52	50.95	55.19	54.90	54.23	53.68		
sms.ww.p-.mtx	53.75	53.81	51.47	51.48	55.34	55.37	54.51	54.36		
sms.ww.st.mtx	53.75	53.64	51.24	51.51	55.4	55.36	54.97	54.72		
sms.ww.pp.mtx	53.80	53.96	51.92	52.1	54.57	54.54	54.94	54.57		
sms.ww.mp.mtx	54.23	54.2	52	52.2	55.02	54.84	55.8	55.45		
morf.w-.p-.add	54.86	55.35	53.41	53.28	57.01	56.94	55.92	55.8	60.37	59.67
morf.w-.pp.add	55.55	55.57	54.02	54.06	57.69	57.41	56.79	56.31	60.61	59.99
morf.w-.mp.add	55.01	55.18	53.5	53.35	57.09	56.82	55.26	54.7	60.1	59.24
morf.ww.p-.add	54.94	55.09	53.27	53.02	56.61	56.55	55.5	54.9	60.23	59.65
morf.ww.pp.add	55.20	55.60	53.83	53.82	57.32	57.2	56.41	55.92	60.57	60.09
morf.ww.mp.add	54.63	54.67	53.46	53.4	56.81	56.73	54.42	53.88	59.67	58.87
morf.w-.p-.att	54.84	55.02	52.98	52.91	56.93	56.83	55.86	55.52	60.3	59.78
morf.w-.pp.att	55.28	55.68	53.92	54.01	57.36	57.24	56.62	56.13	60.83	60.34
morf.w-.mp.att	55.4	55.3	53.72	53.7	56.85	56.93	55.24	54.6	59.89	59.38
morf.ww.p-.att	54.94	54.87	53.07	53.12	56.7	56.36	55.5	55.07	60.31	59.73
morf.ww.pp.att	55.26	55.35	53.83	53.92	57.39	57.15	56.39	55.99	60.5	59.94
morf.ww.mp.att	55.16	55.27	53.22	53.26	56.84	56.67	54.89	54.32	59.59	59.04
morf.w-.p-.mtx	54.97	55.06	52.39	52.42	56.76	56.88	55.93	55.39	60.26	59.66
morf.w-.pp.mtx	55.18	55.4	53.95	53.92	57.49	57.38	56.62	56.14	60.65	60.02
morf.w-.mp.mtx	55.18	55.25	52.92	53.01	56.97	56.69	55.02	54.48	60.07	59.36
morf.ww.p-.mtx	54.9	55.07	53.06	53.07	56.61	56.63	55.6	54.87	60.3	59.65
morf.ww.pp.mtx	55.25	55.45	53.64	53.55	57.31	57.17	56.28	55.86	60.52	59.98
morf.ww.mp.mtx	55.1	55.16	53.43	53.4	56.81	56.69	54.57	53.89	59.71	58.95
bpe.w-.p-.add	50.74	51.3	50.43	50.42	52.74	52.53	47.85	47.6	51.67	51.44
bpe.w-.pp.add	50.92	51.59	50.77	50.68	52.93	52.74	48.32	47.71	51.65	51.33
bpe.w-.mp.add	51.35	51.93	51	50.81	53.37	53.1	47.71	47.3	52.07	51.78
bpe.ww.p-.add	54.6	55.01	53.45	53.39	56.38	55.96	53.35	53.05	59.53	59.03
bpe.ww.pp.add	54.86	55.03	53.47	53.53	56.65	56.27	53.03	52.71	59.95	59.26
bpe.ww.mp.add	55.28	55.7	53.90	53.61	57.14	56.82	53.9	53.52	60.1	59.24
bpe.w-.p-.att	50.65	51.15	50.35	50.21	52.71	52.18	47.52	47.05	51.45	51.31
bpe.w-.pp.att	50.85	51.57	50.66	50.45	52.94	52.47	47.94	47.58	51.71	51.35
bpe.w-.mp.att	51.28	51.94	50.92	50.89	53.13	52.93	47.51	47.21	52.05	51.69
bpe.ww.p-.att	53.19	53.49	51.99	52.09	54.96	54.71	52.3	52.37	57.6	57.17
bpe.ww.pp.att	53.18	53.56	52.13	51.92	55.13	54.91	53.41	53.29	58.67	57.96
bpe.ww.mp.att	54.23	54.71	52.88	52.78	55.85	55.64	53.02	53.15	58.87	58.21
bpe.w-.p-.mtx	50.81	51.48	50.42	50.33	52.76	52.36	47.58	47.14	51.67	51.34
bpe.w-.pp.mtx	51.05	51.79	50.53	50.4	52.89	52.29	47.36	46.97	51.79	51.46
bpe.w-.mp.mtx	51.45	51.88	51.15	50.93	53.27	52.93	47.7	47.34	52.18	51.87
bpe.ww.p-.mtx	53.19	53.49	52.35	52.07	55.15	54.77	52.87	52.6	58.43	57.84
bpe.ww.pp.mtx	53.17	53.57	52.35	52.18	55.08	54.58	52.44	52.02	57.92	57.43
bpe.ww.mp.mtx	54.64	55.05	53.11	52.86	55.73	55.49	53.34	53.11	59.16	58.37
`sgns`	50.56	51	50.55	50.14	49.88	49.87	55.09	54.35	54.93	54.55
`ft`	54.97	55.15	54.46	54.55	57.18	57.18	55.46	54.62	59.67	59.09
`our_ft`	54.59	55.12	54.26	54.4	57.67	57.57	56.19	55.73	60.05	59.65

Table 13: Accuracy on fine-grained entity typing across languages.

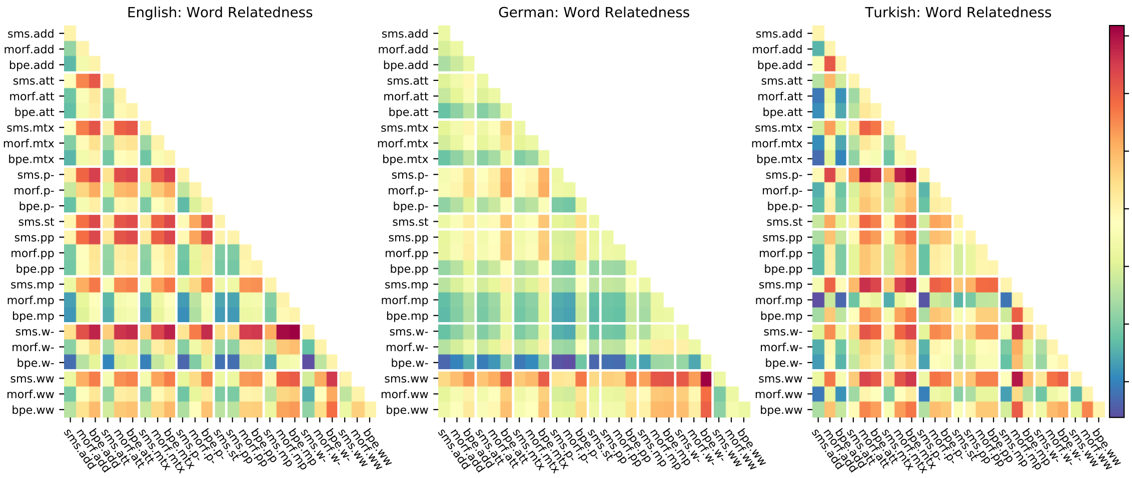


Figure 3: Comparisons of word relatedness tasks for English, German and Turkish.

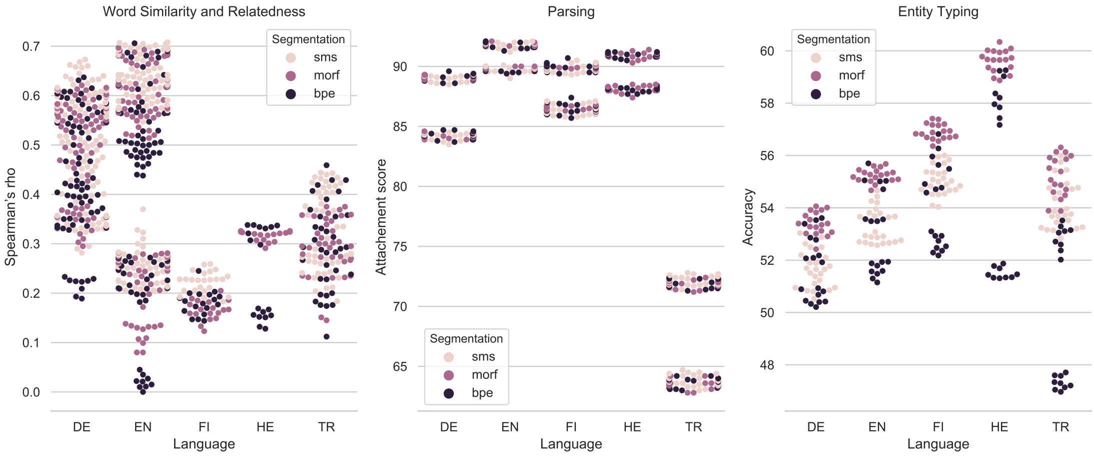


Figure 4: Results with different segmentation methods.

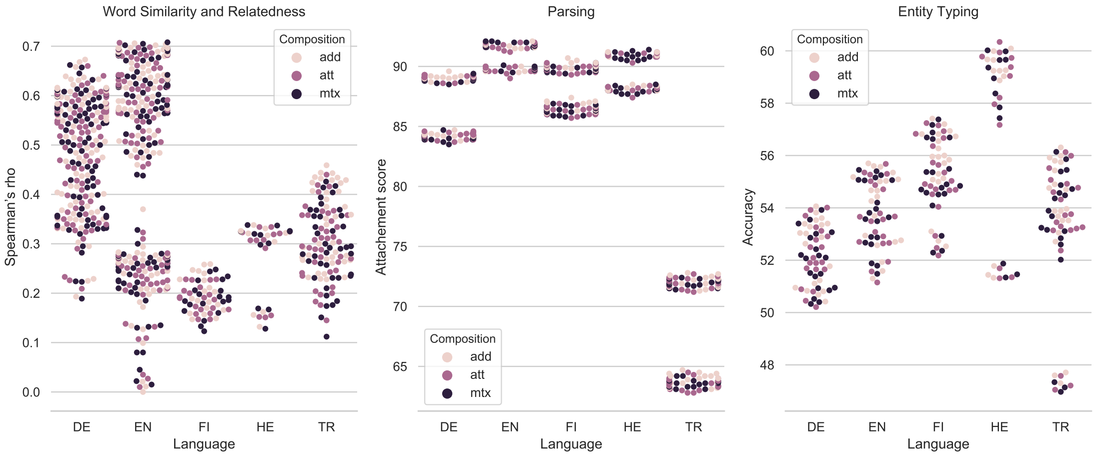


Figure 5: Results with different composition functions.

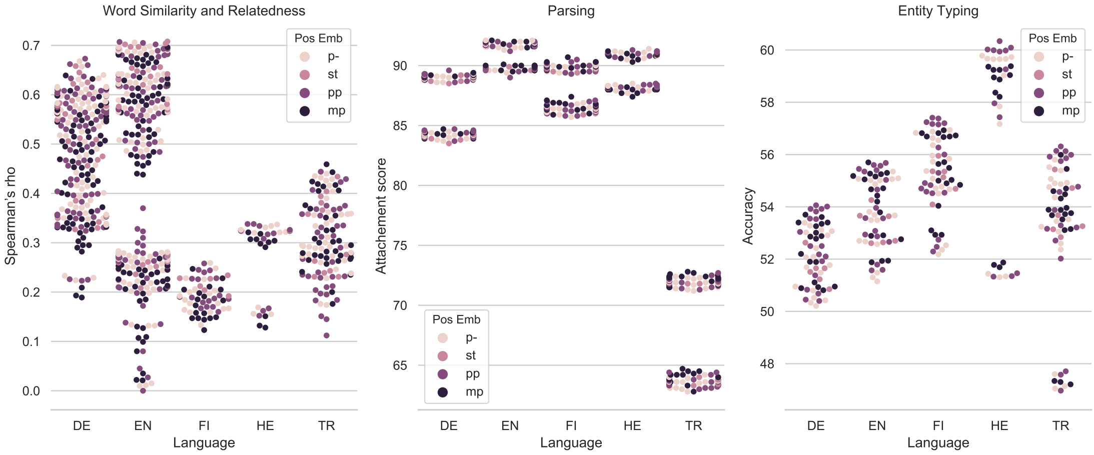


Figure 6: Results with different position embeddings.

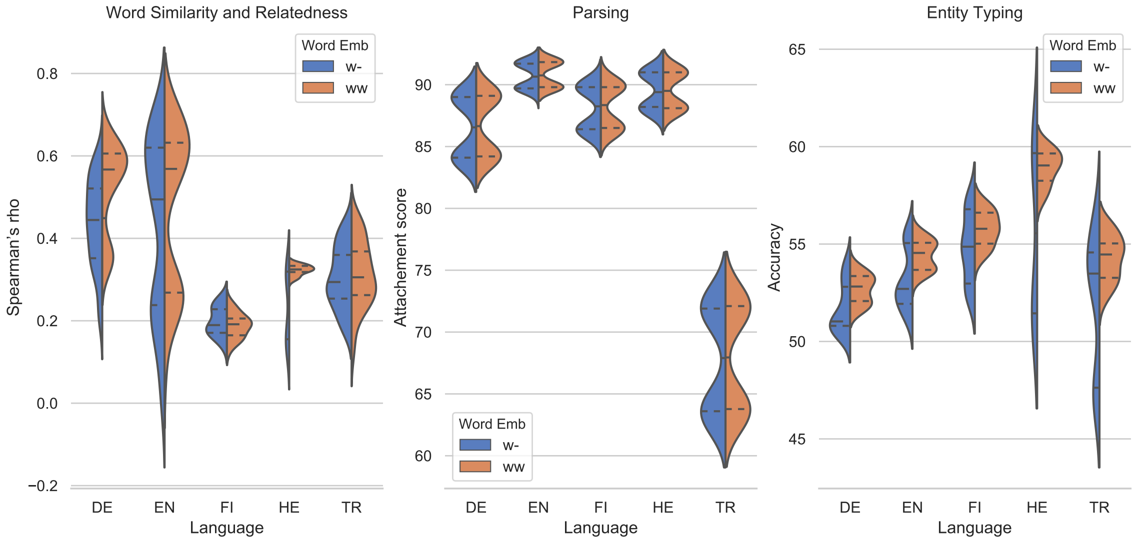


Figure 7: Results w/o word token embedding.

Better Word Embeddings by Disentangling Contextual n-Gram Information

Prakhar Gupta*
EPFL
Switzerland
prakhar.gupta@epfl.ch

Matteo Pagliardini*
Iprova SA, Switzerland
mpagliardini@iprova.com

Martin Jaggi
EPFL
Switzerland
martin.jaggi@epfl.ch

Abstract

Pre-trained word vectors are ubiquitous in Natural Language Processing applications. In this paper, we show how training word embeddings jointly with bigram and even trigram embeddings, results in improved unigram embeddings. We claim that training word embeddings along with higher n-gram embeddings helps in the removal of the contextual information from the unigrams, resulting in better stand-alone word embeddings. We empirically show the validity of our hypothesis by outperforming other competing word representation models by a significant margin on a wide variety of tasks. We make our models publicly available.

1 Introduction

Distributed word representations are essential building blocks of modern NLP systems. Used as features in downstream applications, they often enhance generalization of models trained on a limited amount of data. They do so by capturing relevant distributional information about words from large volumes of unlabeled text.

Efficient methods to learn word vectors have been introduced in the past, most of them based on the distributional hypothesis of Harris (1954); Firth (1957): *"a word is characterized by the company it keeps"*. While a standard approach relies on global corpus statistics (Pennington et al., 2014) formulated as a matrix factorization using mean square reconstruction loss, other widely used methods are the bilinear word2vec architectures introduced by Mikolov et al. (2013a): While skip-gram aims to predict nearby words from a given word, CBOW predicts a target word from its set of context words.

Recently, significant improvements in the quality of the word embeddings were obtained by augmenting word-context pairs with sub-word information in the form of character n-grams (Bojanowski et al., 2017), especially for morphologically rich languages. Nevertheless, to the best of our knowledge, no method has been introduced leveraging collocations of words with higher order word n-grams such as bigrams or trigrams as well as character n-grams together.

In this paper, we show how using higher order word n-grams along with unigrams during training can significantly improve the quality of obtained word embeddings. The addition furthermore helps to disentangle contextual information present in the training data from the unigrams and results in overall better distributed word representations.

To validate our claim, we train two modifications of CBOW augmented with word-n-gram information during training. One is a recent sentence embedding method, Sent2Vec (Pagliardini et al., 2018), which we repurpose to obtain word vectors. The second method we propose is a modification of CBOW enriched with character n-gram information (Bojanowski et al., 2017) that we again augment with word n-gram information. In both cases, we compare the resulting vectors with the most widely used word embedding methods on word similarity and analogy tasks and show significant quality improvements. The code used to train the models presented in this paper as well as the models themselves are made available to the public[1].

2 Model Description

Before introducing our model, we recapitulate fundamental existing word embeddings methods.

CBOW and skip-gram models. Continuous bag-of-words (CBOW) and skip-gram models are

---

* indicates equal contribution

[1] publicly available on http://github.com/epfml/sent2vec

Proceedings of NAACL-HLT 2019, pages 933–939
Minneapolis, Minnesota, June 2 - June 7, 2019.

standard log-bilinear models for obtaining word embeddings based on word-context pair information (Mikolov et al., 2013a). Context here refers to a symmetric window centered on the target word w_t, containing the surrounding tokens at a distance less than some window size ws: $C_t = \{w_k \,|\, k \in [t - \mathsf{ws}, t + \mathsf{ws}]\}$. The CBOW model tries to predict the target word given its context, maximizing the likelihood $\prod_{t=1}^{T} p(w_t|C_t)$, whereas skip-gram learns by predicting the context for a given target word maximizing $\prod_{t=1}^{T} p(C_t|w_t)$. To model those probabilities, a softmax activation is used on top of the inner product between a target vector $\mathbf{u}_{w_t}$ and its context vector $\frac{1}{|C_t|}\sum_{w \in C_t} \mathbf{v}_w$.

To overcome the computational bottleneck of the softmax for large vocabulary, negative sampling or noise contrastive estimation are well-established (Mikolov et al., 2013b), with the idea of employing simpler pairwise binary classifier loss functions to differentiate between the valid context C_t and fake contexts N_{C_t} sampled at random. While generating target-context pairs, both CBOW and skip-gram also use input word subsampling, discarding higher-frequency words with higher probability during training, in order to prevent the model from overfitting the most frequent tokens. Standard CBOW also uses a dynamic context window size: for each subsampled target word w, the size of its associated context window is sampled uniformly between 1 and ws (Mikolov et al., 2013b).

Adding character n-grams. Bojanowski et al. (2017) have augmented CBOW and skip-gram by adding character n-grams to the context representations. Word vectors are expressed as the sum of its unigram and average of its character n-gram embeddings W_w:

$$\mathbf{v} := \mathbf{v}_w + \frac{1}{|W_w|} \sum_{c \in W_w} \mathbf{v}_c$$

Character n-grams are hashed to an index in the embedding matrix . The training remains the same as for CBOW and skip-gram. This approach greatly improves the performances of CBOW and skip-gram on morpho-syntactic tasks. For the rest of the paper, we will refer to the CBOW and skip-gram methods enriched with subword-information as *CBOW-char* and *skip-gram-char* respectively.

GloVe. Instead of training online on local window contexts, GloVe vectors (Pennington et al., 2014) are trained using global co-occurrence statistics by factorizing the word-context co-occurrence matrix.

Ngram2vec. In order to leverage the performance of word vectors, training of word vectors using the skip-gram objective function with negative sampling is augmented with n-gram co-occurrence information (Zhao et al., 2017).

2.1 Improving unigram embeddings by adding higher order word-n-grams to contexts

CBOW-char with word n-grams. We propose to augment CBOW-char to additionally use word n-gram context vectors (in addition to char n-grams and the context word itself). More precisely, during training, the context vector for a given word w_t is given by the average of all word-n-grams N_t, all char-n-grams, and all unigrams in the span of the current context window C_t:

$$\mathbf{v} := \frac{\sum_{w \in C_t} \mathbf{v}_w + \sum_{n \in N_t} \mathbf{v}_n + \sum_{w \in C_t} \sum_{c \in W_w} \mathbf{v}_c}{|C_t| + |N_t| + \sum_{w \in C_t} |W_w|} \tag{1}$$

For a given sentence, we apply input subsampling and a sliding context window as for standard CBOW. In addition, we keep the mapping from the subsampled sentence to the original sentence for the purpose of extracting word n-grams from the original sequence of words, within the span of the context window. Word n-grams are added to the context using the hashing trick in the same way char-n-grams are handled. We use two different hashing index ranges to ensure there is no collision between char n-gram and word n-gram representations.

Sent2Vec for word embeddings. Initially implemented for sentence embeddings, Sent2Vec (Pagliardini et al., 2018) can be seen as a derivative of word2vec's CBOW. The key differences between CBOW and Sent2Vec are the removal of the input subsampling, considering the entire sentence as context, as well as the addition of word-n-grams.

Here, word and n-grams embeddings from an entire sentence are averaged to form the corresponding sentence (context) embedding.

For both proposed CBOW-char and Sent2Vec models, we employ dropout on word n-grams during training. For both models, word embeddings are obtained by simply discarding the higher order n-gram embeddings after training.

Model	WS 353	WS 353 Relatedness	WS 353 Similarity
CBOW-char	.709 ± .006	.626 ± .009	.783 ± .004
CBOW-char + bi.	.719 ± .007	.652 ± .010	.778 ± .007
CBOW-char + bi. + tri.	.727 ± .008	.664 ± .008	.783 ± .004
Sent2Vec uni.	.705 ± .004	.593 ± .005	.793 ± .006
Sent2Vec uni. + bi.	.755 ± .005	.683 ± .008	.817 ± .007
Sent2Vec uni. + bi. + tri.	**.780 ± .003**	**.721 ± .006**	**.828 ± .003**

Model	SimLex-999	MEN	Rare Words	Mechanical Turk
CBOW-char	.424 ± .004	.769 ± .002	.497 ± .002	.675 ± .007
CBOW-char + bi.	.436 ± .004	.786 ± .002	.506 ± .001	.671 ± .007
CBOW-char + bi. + tri.	.441 ± .003	.788 ± .002	**.509 ± .003**	**.678 ± .010**
Sent2Vec uni.	.450 ± .003	.765 ± .002	.444 ± .001	.625 ± .005
Sent2Vec uni. + bi.	.440 ± .002	.791 ± .002	.430 ± .002	.661 ± .005
Sent2Vec uni. + bi. + tri.	**.464 ± .003**	**.798 ± .001**	.432 ± .003	.658 ± .006

Model	Google (Syntactic Analogies)	Google (Semantic Analogies)	MSR
CBOW-char	.920 ± .001	.799 ± .004	.842 ± .002
CBOW-char + bi.	.928 ± .003	.798 ± .006	.856 ± .004
CBOW-char + bi. + tri.	**.929 ± .001**	.794 ± .005	**.857 ± .002**
Sent2Vec uni.	.826 ± .003	.847 ± .003	.734 ± .003
Sent2Vec uni. + bi.	.843 ± .004	.844 ± .002	.754 ± .004
Sent2Vec uni. + bi. + tri.	.837 ± .003	**.853 ± .003**	.745 ± .001

Table 1: Impact of using word n-grams: Models are compared using Spearman correlation measures for word similarity tasks and accuracy for word analogy tasks. Top performances on each dataset are shown in bold. An underline shows the best model(s) restricted to each architecture type. The abbreviations uni., bi., and tri. stand for unigrams, bigrams, and trigrams respectively.

3 Experimental Setup

3.1 Training

We train all competing models on a wikipedia dump of 69 million sentences containing 1.7 billion words, following (Pagliardini et al., 2018).

Sentences are tokenized using the Stanford NLP library (Manning et al., 2014). All algorithms are implemented using a modified version of the fasttext (Bojanowski et al., 2017; Joulin et al., 2017) and sent2vec (Pagliardini et al., 2018) libraries respectively. Detailed training hyperparameters for all models included in the comparison are provided in Table 3 in the supplementary material. During training, we save models checkpoints at 20 equidistant intervals and found out that the best performance for CBOW models occurs around $60-80\%$ of the total training. As a result, we also indicate the checkpoint at which we stop training the CBOW models. We use 300-dimension vectors for all our word embedding models. For the Ngram2vec model, learning source and target embeddings for all the n-grams upto bigrams was the best performing model and is included in the comparison.

For each method, we extensively tuned hyperparameters starting from the recommended values. For each model, we select the parameters which give the best averaged results on our word-similarity and analogy tasks. After selecting the best hyperparameters, we train 5 models for each method, using a different random seed. The reported results are given as mean and standard deviation for those five models.

3.2 Evaluation

In order to evaluate our model, we use six datasets covering pair-wise word-similarity tasks and two datasets covering word-analogy tasks.

Word-similarity tasks. Word-similarity tasks consist of word pairs along with their human annotated similarity scores. To evaluate the performance of our models on pair-wise word-similarity tasks, we use *WordSim353* (353 word-pairs) (Finkelstein et al., 2002) divided into two datasets, *WordSim Similarity* (203 word-pairs) and

Model	WS 353	WS 353 Relatedness	WS 353 Similarity
CBOW-char + bi. + tri.	.727 ± .008	.664 ± .008	.783 ± .004
Sent2Vec uni. + bi. + tri.	**.780 ± .003**	**.721 ± .006**	**.828 ± .003**
CBOW-char	.709 ± .006	.626 ± .009	.783 ± .004
CBOW	.722 ± .008	.634 ± .008	.796 ± .005
Skip-gram-char	.724 ± .007	.655 ± .008	.789 ± .004
Skip-gram	.736 ± .004	.672 ± .007	.796 ± .005
GloVe	.559 ± .002	.484 ± .005	.665 ± .008
Ngram2vec bi. - bi.	.745 ± .003	.687 ± .003	.797 ± .004

Model	SimLex-999	MEN	Rare Words	Mechanical Turk
CBOW-char + bi. + tri.	.441 ± .003	.788 ± .002	**.509 ± .003**	.678 ± .010
Sent2Vec uni. + bi. + tri.	**.464 ± .003**	**.798 ± .001**	.432 ± .003	.658 ± .006
CBOW-char	.424 ± .004	.769 ± .002	.497 ± .002	.675 ± .007
CBOW	.432 ± .004	.757 ± .002	.454 ± .002	.674 ± .006
Skip-gram-char	.395 ± .003	.762 ± .001	.487 ± .002	**.684 ± .003**
Skip-gram	.405 ± .001	.770 ± .001	.468 ± .002	**.684 ± .005**
GloVe	.375 ± .002	.690 ± .001	.327 ± .002	.622 ± .004
Ngram2vec bi. - bi.	.424 ± .000	.756 ± .001	.462 ± .002	.681 ± .004

Model	Google (Syntactic Analogies)	Google (Semantic Analogies)	MSR
CBOW-char + bi. + tri.	**.929 ± .001**	.794 ± .005	**.857 ± .002**
Sent2Vec uni. + bi. + tri.	.837 ± .003	**.853 ± .003**	.745 ± .001
CBOW-char	.920 ± .001	.799 ± .004	.842 ± .002
CBOW	.816 ± .002	.805 ± .005	.713 ± .004
Skip-gram-char	.860 ± .001	.828 ± .005	.796 ± .003
Skip-gram	.829 ± .002	.837 ± .002	.753 ± .005
GloVe	.767 ± .002	.697 ± .007	.678 ± .003
Ngram2vec bi. - bi.	.834 ± .001	.812 ± .003	.761 ± .001

Table 2: Improvement over existing methods: Models are compared using Spearman correlation measures for word similarity tasks and accuracy for word analogy tasks. Top performance(s) on each dataset is(are) shown in bold. The abbreviations uni., bi., and tri. stand for unigrams, bigrams, and trigrams respectively.

WordSim Relatedness (252 word-pairs) (Agirre et al., 2009); *MEN* (3000 word-pairs) (Bruni et al., 2012); *Mechanical Turk* dataset (Radinsky et al., 2011) (287 word-pairs); *Rare words dataset* (2034 word-pairs) (Luong et al., 2013); and *SimLex-999* (999 word-pairs) (Hill et al., 2015) dataset.

To calculate the similarity between two words, we use the cosine similarity between their word representations. The similarity scores then, are compared to the human ratings using Spearman's ρ (Spearman, 1904) correlation scores.

Word-analogy tasks. Word analogy tasks pose analogy relations of the form "x is to y as $x^\star$ is to $y^\star$", where y is hidden and must be guessed from the dataset vocabulary.

We use the *MSR* (Mikolov et al., 2013c) and the *Google* (Mikolov et al., 2013a) analogy datasets. The *MSR* dataset contains 8000 syntactic analogy quadruplets while the *Google* set has 8869 semantic and 10675 syntactic relations.

To calculate the missing word in the relation, we use the 3CosMul method (Levy and Goldberg, 2014):

$$y^\star := \arg \max_{z \in \mathcal{V} \setminus \{x, y, x^\star\}} \frac{cos(\mathbf{v}_z, \mathbf{v}_y) \cdot cos(\mathbf{v}_z, \mathbf{v}_{x^\star})}{cos(\mathbf{v}_z, \mathbf{v}_x) + \varepsilon} \tag{2}$$

where $\varepsilon = 0.0001$ is used to prevent division by 0 and $\mathcal{V}$ is the dataset vocabulary.

We remove all the out of vocabulary words and are left with 6946 syntactic relations for the *MSR* dataset and 1959 word-pairs for the *Rare Words* dataset. All other datasets do not have any out of vocabulary words.

4 Results

Impact of word n-grams. In Table 1, we evaluate the impact of adding contextual word n-grams to two CBOW variations: CBOW-char and Sent2Vec. By adding n-gram information, we consistently observe a boost in the Spearman correlation on the word similarity tasks. On the few datasets where we do not observe an improvement, the results for word-n-gram augmented methods are within standard deviation reach. The *Rare Words* dataset for Sent2Vec is the only exception, despite getting some improvement for CBOW-char based methods. This observation can be attributed to the fact that character ngrams are shared between unigrams, enhancing generalization to infrequent words. Without char n-grams, the model might underfit those rare words, even more so with word n-grams.

We also see that the boost obtained by adding n-grams on word-similarity tasks is much larger for Sent2Vec models as compared to the CBOW-char ones possibly due to the fact that during training, Sent2Vec models use a much larger context and hence can use much more n-gram information for obtaining a better context representation.

For analogy tasks, we see an improvement in the augmented CBOW-char methods for morpho-syntactic analogy datasets with little or no gain for semantic analogy datasets. Yet, for Sent2Vec models, the gain is the other way around. This observation indicates the strong role played by character n-grams in boosting the performance on the syntactic tasks as well as restricting the word n-grams from improving the performance on semantic analogies.

Comparison with competing methods. In Table 2, we compare word n-gram augmented methods with the most prominent word embedding models. We obtain state-of-the-art results for standalone unigram embeddings on most of the datasets confirming our hypothesis. The *Mechanical Turk* dataset is the only exception.

We notice that Sent2Vec trigrams model dominates the word-similarity tasks as well as the semantic analogy tasks. However, character n-grams are quite helpful when it comes to syntactic analogy tasks underlining the importance of subword information. We also note that the Ngram2vec model outperforms our augmented CBOW-char model in some of the tasks but is always inferior to Sent2Vec in those cases.

5 Conclusion and Future Work

We empirically show how augmenting the context representations using higher-order word n-grams improves the quality of word representations. The empirical success also calls for a new theoretical model on the composite effect of training higher order n-grams simultaneously with unigrams. Also, the success of Sent2Vec on word-level tasks, a method originally geared towards obtaining general purposed sentence embeddings, hints towards the additional benefits of using compositional methods for obtaining sentence/phrase representations.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Z Harris. 1954. Distributional structure.(j. katz, ed.) word journal of the international linguistic association, 10 (23), 146-162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS - Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 244–253.

A Training parameters for selected models

Training parameters for all models except GloVe and Ngram2vec are provided in Table 3. For the GloVe model , the minimum word count is set to 10; the window size is set to 10; we use 10 epochs for training; X_{max}, the weighting parameter for the word-context pairs is set to 100; all other parameters are set to default. For Ngram2vec, the minimum word count is set to 10; the window size is set to 5; both source and target vectors are trained for unigrams and bigrams; overlap between the target word and source n-grams is allowed. All other features are set to default. To train the Ngram2vec models, we use the library provided by (Zhao et al., 2017)[2].

---

[2] `https://github.com/zhezhaoa/ngram2vec`

Model	Sent2Vec uni.	Sent2Vec uni.+bi.	Sent2Vec uni.+bi+tri.	CBOW (char.)	CBOW (char.)+bi.	CBOW (char.)+bi.+tri.	CBOW	Skip-gram (char.)	Skip-gram
Embedding Dimensions	300	300	300	300	300	300	300	300	300
Max Vocab. Size	750k	750k	750k	750k	750k	750k	750k	750k	750k
Minimum Word Count	5	5	5	5	5	5	5	5	5
Initial Learning Rate	0.2	0.2	0.2	0.05	0.05	0.05	0.05	0.05	0.05
Epochs	9	9	9	9	9	9	5	15	15
Subsampling hyper-param.	1×10^{-5}	5×10^{-5}	5×10^{-6}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Word-Ngrams Bucket Size	-	2M	4M	-	2M	4M	-	-	-
Char-Ngrams Bucket Size	-	-	-	2M	2M	2M	-	2M	-
Word-Ngrams Dropped per context	-	4	4	-	2	2	-	-	-
Window Size	-	-	-	10	10	10	10	5	5
Number of negatives sampled	10	10	10	5	5	5	5	5	5
Max Char-Ngram Size	-	-	-	6	6	6	-	6	-
Min Char-Ngram Size	-	-	-	3	3	3	-	3	-
Percentage at which training is halted (For CBOW models only)	-	-	-	75%	80%	80%	60%	-	-

Table 3: Training parameters for all non-GloVe models

Integration of Knowledge Graph Embedding into Topic Modeling with Hierarchical Dirichlet Process

Dingcheng Li, Siamak Zamani Dadaneh, Jingyuan Zhang, Ping Li
Cognitive Computing Lab (CCL)
Baidu Research USA
10900 NE Eighth St, Bellevue, WA 98004, USA
1195 Bordeaux Dr, Sunnyvale, CA 94089, USA
`lidingcheng@baidu.com,` `zamanys4@gmail.com`
`zhangjingyuan03@baidu.com,` `liping11@baidu.com`

Abstract

Leveraging domain knowledge is an effective strategy for enhancing the quality of inferred low-dimensional representations of documents by topic models. In this paper, we develop *topic modeling with knowledge graph embedding* (TMKGE), a Bayesian nonparametric model to employ knowledge graph (KG) embedding in the context of topic modeling, for extracting more coherent topics. Specifically, we build a hierarchical Dirichlet process (HDP) based model to flexibly borrow information from KG to improve the interpretability of topics. An efficient online variational inference method based on a stick-breaking construction of HDP is developed for TMKGE, making TMKGE suitable for large document corpora and KGs. Experiments on three public datasets illustrate the superior performance of TMKGE in terms of topic coherence and document classification accuracy, compared to state-of-the-art topic modeling methods.

1 Introduction

Topic models, such as Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 2017) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003), play significant roles in helping machines interpret text documents. Topic models consider documents as a bag of words. Given the word information, topic models formulate documents as mixtures of latent topics, where these topics are generated via the multinomial distributions over words. Bayesian methods are utilized to extract topical structures from the document-word frequency representations of the text corpus. Without supervision, however, it is found that the topics generated from these models are often not interpretable (Chang et al., 2009; Mimno et al., 2011). In recent studies, incorporating knowledge of different forms as a supervision has become a powerful strategy for discovering meaningful topics (Andrzejewski et al., 2009).

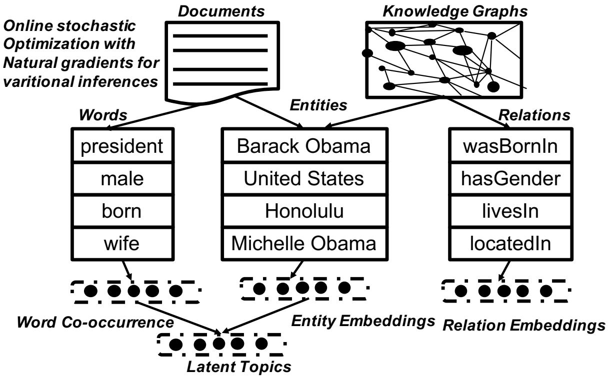


Figure 1: An overview of the proposed TMKGE framework. Entities are shared by both documents and knowledge graphs. Entity embeddings generated by *TransE* a knowledge graph embedding package are passed into TMKGE to generate hidden topics.

Most conventional approaches take prior domain knowledge into account to improve the topic coherence (Andrzejewski et al., 2009; Andrzejewski and Zhu, 2009; Hu et al., 2014; Jagarlamudi et al., 2012; Doshi-Velez et al., 2015). One commonly used domain knowledge is based on word correlations (Andrzejewski et al., 2009; Chen et al., 2013; Chen and Liu, 2014). For example, *must-links* and *cannot-links* among words are generated by domain experts to help topic modeling (Andrzejewski et al., 2009). Another useful form of knowledge for topic discoveries is based on word semantics (Andrzejewski and Zhu, 2009; Chemudugunta et al., 2008; Hu et al., 2014; Jagarlamudi et al., 2012; Doshi-Velez et al., 2015). In particular, word embedding (Pennington et al., 2014; Goldberg and Levy, 2014), in which bag of words are transformed into vector representations so that contexts are embedded into those word vectors, are used as semantic regularities to enhance topic models (Nguyen et al., 2015; Li et al., 2016; Das et al., 2015; Batmanghelich et al., 2016).

Proceedings of NAACL-HLT 2019, pages 940–950
Minneapolis, Minnesota, June 2 - June 7, 2019.

Knowledge graph (KG) embedding (Bordes et al., 2013) learns a low-dimensional continuous vector space for entities and relations to preserve the inherent structure of KGs. Yao et al. (2017) proposes KGE-LDA to incorporate embeddings of KGs into topic models to extract better topic representations for documents and shows promising performance. However, KGE-LDA forces words and entities to have identical latent representations, which is a rather restrictive assumption that prevents the topic model from recovering correct underlying latent structures of the data, especially in scenarios where only partial KGs are available.

This paper develops *topic modeling with knowledge graph embedding* (TMKGE), a hierarchical Dirichlet process (HDP) based model to extract more coherent topics by taking advantage of the KG structure. Unlike KGE-LDA, the proposed TMKGE allows for more flexible sharing of information between words and entities, by using a multinomial distribution to model the words and a multivariate Gaussian mixture to model the entities. With this approach, we introduce two proportional vectors, one for words and one for entities. In contrast, KGE-LDA only uses one, shared by both words and entities. Similar to HDP, TMKGE includes a collection of Dirichlet processes (DPs) at both corpus and document levels. The atoms of corpus-level DP form the base measure for document levels DPs of words and entities. Therefore, the atoms of corpus-level DP can represent word topics, entity mixture components, or both of them. Figure 1 provides an overview of TMKGE, where two sources of inputs, bag of words and KG embedding, extracted from corpus and KGs respectively, are passed into TMKGE.

As a nonparametric model, TMKGE does not assume a fix number of topics or entity mixture components as constraints. Instead, it learns the number of topics and entity mixture components automatically from the data. Furthermore, an efficient online variational inference algorithm is developed, based on Sethuraman's stick-breaking construction of HDP (Sethuraman, 1994). We in fact construct stick-breaking inference in a minibatch fashion (Wang et al., 2011; Bleier, 2013), to derive a more efficient and scalable coordinate-accent variational inference for TMKGE.

Summary of contributions: TMKGE is a Bayesian nonparametric model to extract more coherent topics by taking advantage of knowledge graph structures. We introduce two proportional vectors for more flexible sharing of information between words and entities. We derive an efficient and scalable parameter estimation algorithm via online variational inference. Finally, we empirically demonstrate the effectiveness of TMKGE in topic discovering and document classification.

2 Background and Related Work

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a popular probabilistic model that learns latent topics from documents and words, by using Dirichlet priors to regularize the topic distributions. The generated topics from LDA models, however, are often not interpretable (Chang et al., 2009; Mimno et al., 2011), in part because LDA models are unsupervised without using prior knowledge or external resources.

In recent years, prior knowledge are leveraged to guide the process of topic modeling (Andrzejewski and Zhu, 2009; Hu et al., 2014; Jagarlamudi et al., 2012; Doshi-Velez et al., 2015). For example, the deep forest LDA (DF-LDA) model (Andrzejewski et al., 2009) is proposed to incorporate *must-links* and *cannot-links* among words into topic modeling. One weakness of the DF-LDA model is that the link information is domain-dependent. Later, general knowledge based LDA is introduced to leverage must-links from multiple domains (Chen et al., 2013). More recently, MetaLDA (Zhao et al., 2017) proposes to improve topic modeling by incorporating diverse meta information as priors for both document hyperparameter α and word hyperparameter β.

Besides the word correlations, word semantics are also utilized as one type of useful knowledge for topic modeling (Chemudugunta et al., 2008; Hu et al., 2014; Jagarlamudi et al., 2012). Word embeddings, as a low-dimensional continuous vectors of words (Mikolov et al., 2013; Bengio et al., 2003; Pennington et al., 2014) are regarded to be an efficient representations of word semantics. Latent Feature Topic Modeling (LFTM) is proposed to use pre-trained word embeddings in topic modeling (Nguyen et al., 2015). It incorporates the embedding of a word and its topics into the traditional multinomial distribution over words as the probability function of topic modeling. TopicVec extends LFTM by combining a word and its local contextual words together into the conventional multinomial distribution over words. It also learns embedding representations for topics (Li et al., 2016). Gaussian-LDA goes further to improve topic modeling (Das et al., 2015) by taking into considerations the continuous nature of word embeddings. Shi et al. (2017) constructs a more unified framework, STE (skip-gram topic embedding) to address the problem of polysemy. Li et al. (2019) proposes a unified framework TMSA (Topic Modeling and Sparse Autoencoder) to improve topic discovery and word embedding simultaneously via a mutual learning mechanism.

Hu et al. (2016) proposes topic-based embeddings for learning from large knowledge graphs (KGE). KGE learns low-dimensional continuous vector space for both entities and relations to preserve the inherent structure of knowledge graphs. A Bayesian method is introduced by considering the embeddings of entities and relations as topics. Later, Yao et al. (2017) proposes knowledge graph embedding LDA (KGE-LDA) to encode entity embeddings learned from knowledge graphs into LDA and show that knowledge graph embeddings boost topic discoveries. Inspired by this work, we explore to utilize entity embeddings to encode prior knowledge for topic modeling.

3 Method

This section presents the TMKGE model and an efficient online variational inference for learning its parameters. We first provide a review of hierarchical Dirichlet process (HDP) (Teh et al., 2005).

3.1 Preliminaries of HDP

Dirichlet process (DP) (MacEachern and Müller, 1998) $G \sim \text{DP}(\gamma_0, G_0)$, with a base measure G_0 and a concentration parameter $\gamma_0 > 0$, is the distribution of a random probability measure G over a measurable space $(\Omega, \mathcal{B})$, such that for any measurable disjoint partition $(A_1, ..., A_Q)$ of Ω,

$$(G(A_1), ..., G(A_Q)) \sim \text{Dir}(\gamma_0 G_0(A_1), ..., \gamma_0 G_0(A_Q))$$

where “Dir” denotes a Dirichlet distribution.

Hierarchical Dirichlet process (HDP) (Teh et al., 2005), introduced for dealing with multiple (D) groups of data, is a distribution over a set of random probability measures over $(\Omega, \mathcal{B})$: one probability measure $G_d \sim \text{DP}(\alpha_0, G_0)$ for each group $d \in \{1, 2, ..., D\}$, and a global probability measure $G_0 \sim \text{DP}(\gamma_0, H)$ with a base measure H.

Stick-breaking construction Teh et al. (2005) shows that the draws from G_0 and G_d can be expressed as weighted sums of point masses:

$$G_0 = \sum_{k=0}^{\infty} \beta_k \delta_{\phi_k}, \qquad G_d = \sum_{k=0}^{\infty} \pi_{dk} \delta_{\phi_k}.$$

A more convenient stick-breaking construction, especially for deriving closed-form variational inference (Wang et al., 2011), is Sethuraman (1994)’s construction, which proceeds as follows. First, the global-level DP draw is represented as

$$\beta'_k \sim \text{Beta}(1, \gamma_0), \qquad \beta_k = \beta'_k \prod_{\ell=1}^{k-1} (1 - \beta'_\ell),$$

Note that the distribution for $\boldsymbol{\beta} = \{\beta_k\}_{k=1}^{\infty}$ is also commonly written as $\boldsymbol{\beta} \sim \text{GEM}(\gamma_0)$ (Pitman, 2002). Subsequently, the group-level draws are constructed as

$$\begin{aligned} \psi_{dt} \sim & G_0, \qquad \pi'_{dt} = \text{Beta}(1, \alpha_0), \\ \pi_{dt} = & \pi'_{dt} \prod_{\ell=1}^{t-1} (1 - \pi'_{d\ell}), \qquad G_d = \sum_{t=1}^{\infty} \pi_{dt} \delta_{\psi_{dt}}. \end{aligned} \tag{1}$$

Alternatively, the group-level atoms $\{\psi_{dt}\}_{t=1}^{\infty}$ can be represented as $\psi_{dt} = \phi_{c_{dt}}$, where the auxiliary indicator variables c_{dt} are independently drawn from a multinomial $\text{Mult}(\boldsymbol{\beta})$.

Teh et al. (2008) also proposes a collapsed inference method as an alternative of stick-breaking inference. However, following Fox et al. (2011), we stick to the uncollapsed HDP model considering our truncated Dirichlet process has more computational efficiency and is simple to implement.

3.2 The TMKGE Model

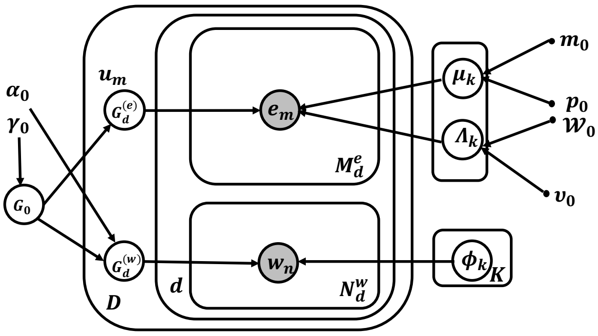


Figure 2: Graphical representation of the TMKGE framework. There are two components, the lower of which is the one for words and the upper of which is the one for entities. Both components share the Dirichlet process as priors. Since entities are represented with knowledge graph embeddings, therefore, each entity is generated with Gaussian priors while the one for words is still generated with Dirichlet priors.

Figure 2 is the graphical representation of TMKGE. Let D denote the number of documents in the corpus, where each document $d \in \{1, 2, ..., D\}$ contains $N_d^{(w)}$ words and $N_d^{(e)}$ entities. Throughout this work, superscripts (w) and (e) indicate word and entity related parameters, respectively. In each document d, the n-th word is represented by w_{dn}, where each word belongs to a vocabulary of size V, i.e., $w_{dn} \in \{1, 2, ..., V\}$. Furthermore, the P-dimensional embedding of the m-th entity is $\boldsymbol{e}_{dm}$, where the total number of unique entities in the corpus is E. We assume that entity embeddings are obtained from the "complete" knowledge graph, and hence they contain information independent of the corpus. In this paper, we use TransE (Bordes et al., 2013), a simple and effective tool for knowledge encoding, to calculate the embeddings of entities extracted from the documents. We should mention that we remove the normalization step of TransE and thus the output vectors ($\boldsymbol{e}_{dm}$) do not have unit ℓ_2 norm.

TMKGE builds upon HDP for joint modeling of word topics and entity mixtures. At the corpus level, word topics and entity mixtures correspond to atoms of a Dirichlet process $G_0 \sim \text{DP}(\gamma_0, H)$. At the document level, word topics and entity mixture components are atoms of independent DPs, with shared base measure G_0. Mathematically, for document d, we have

$$G_d^{(w)} \sim \text{DP}(\alpha_0, G_0), \qquad G_d^{(e)} \sim \text{DP}(\alpha_0, G_0),$$

where $G_d^{(w)}$ and $G_d^{(e)}$ are word and entity related DPs. Sethuraman's construction in (1) yields

$$G_d^{(w)} = \sum_{t=1}^{\infty} \pi_{dt}^{(w)} \delta_{\psi_{dt}^{(w)}}, \qquad G_d^{(e)} = \sum_{t=1}^{\infty} \pi_{dt}^{(e)} \delta_{\psi_{dt}^{(e)}}. \qquad (2)$$

These DPs are then used to assign words and entities to topics and mixture components, respectively. In document d, let $z_{dn}^{(w)}$ denote the topic assigned to the n-th word, and $z_{dm}^{(e)}$ denote the mixture component assigned to the m-th entity. Using the mixing proportions of DPs in (2), we have

$$p(z_{dn}^{(w)} = t) = \pi_{dt}^{(w)}, \qquad p(z_{dn}^{(e)} = t) = \pi_{dt}^{(e)}.$$

For simplicity, we use index t to denote both word and entity related atoms, although they can correspond to different atoms of the global DPs.

The mixing proportions of corpus-level DP are used to map the document atoms to the shared global atoms. More precisely, we introduce the word and entity atoms mapping auxiliary variables $\boldsymbol{c}_d^{(w)} = \{c_{dt}^{(w)}\}_{t=1}^{\infty}$ and $\boldsymbol{c}_d^{(e)} = \{c_{dt}^{(e)}\}_{t=1}^{\infty}$. The mapping probabilities then can be expressed as

$$p(c_{dt}^{(w)} = k) = \beta_k, \qquad p(c_{dt}^{(e)} = k) = \beta_k.$$

TMKGE allows flexible sharing of information between knowledge graphs and documents. This is an important advantage, as in practice only partial relational information are available, and thus strictly forcing the topics and entity mixtures to share components may lead to reducing the power of model to correctly recover the latent structure of the data. Furthermore, the nonparametric nature of the model enables the automatic discovery of number of atoms for both words and entities, at document and corpus levels.

Each atom of corpus DP (G_0) corresponds to a set of parameters for both words and entities. Atom k contains topic-word Dirichlet distribution $\boldsymbol{\phi}_k = (\phi_{k1}, ..., \phi_{kV})^T$, and entity Gaussian mixture parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\}$. Given $\boldsymbol{\phi}_k$ and topic assignment variables, the generative process for n-th word of document d is

$$\begin{aligned} z_{dn}^{(w)} &\sim \text{Mult}(\boldsymbol{\pi}_d^{(w)}), \\ (w_{dn} | z_{dn}^{(w)} = t, c_{dt}^{(w)} = k, \boldsymbol{\phi}_k) &\sim \text{Mult}(\boldsymbol{\phi}_k). \end{aligned}$$

In a similar fashion, the generative process of m-th entity of document d is

$$\begin{aligned} z_{dm}^{(e)} &\sim \text{Mult}(\boldsymbol{\pi}_d^{(e)}), \\ (\boldsymbol{e}_{dm} | z_{dm}^{(e)} = t, c_{dt}^{(e)} = k, \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) &\sim \text{N}(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}), \end{aligned}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ are the mean vector and precision matrix of multivariate Gaussian distribution.

Furthermore, we impose conjugate priors on both word and entity components parameters as:

$$\begin{aligned} \boldsymbol{\phi}_k &\sim \text{Dir}(\eta, ..., \eta), \qquad \boldsymbol{\mu}_k \sim \text{N}\big(\boldsymbol{m}_0, (\rho_0 \boldsymbol{\Lambda}_k)^{-1}\big), \\ \boldsymbol{\Lambda}_k &\sim \text{Wishart}(\nu_0, \boldsymbol{W}_0). \end{aligned}$$

3.3 Online Variational Inference

In this section, inspired by (Wang et al., 2011), we propose an online variational inference algorithm for efficient learning of TMKGE model parameters. We use a fully factorized variational distribution based on stick-breaking construction, and perform online mean-field variational inference.

In addition to topic parameters $\boldsymbol{\phi}_k$ and entity mixture parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k\}$, other parameters of interest are corpus-level stick proportions $\boldsymbol{\beta}' = \{\beta'_k\}_{k=1}^{\infty}$, document-level stick proportions for words $\boldsymbol{\pi}_d'^{(w)} = \{\pi_{dt}'^{(w)}\}_{t=1}^{\infty}$ and entities $\boldsymbol{\pi}_d'^{(e)} = \{\pi_{dt}'^{(e)}\}_{t=1}^{\infty}$, topic assignments for words

$\boldsymbol{z}_d^{(w)} = \{z_{dn}^{(w)}\}_{n=1}^{N_d^{(w)}}$, mixture assignments for entities $\boldsymbol{z}_d^{(e)} = \{z_{dm}^{(e)}\}_{m=1}^{N_d^{(e)}}$, and mapping variables $\boldsymbol{c}_d^{(w)}$ and $\boldsymbol{c}_d^{(e)}$. Denote $\Theta^{(w)}$ and $\Theta^{(e)}$ respectively the word and entity related parameters. Then the variational distribution factorizes as

$$q(\boldsymbol{\beta}', \Theta^{(w)}, \Theta^{(e)}) = q(\boldsymbol{\beta}')q(\Theta^{(w)})q(\Theta^{(e)}).$$

For corpus-level stick proportions, we assume a Beta distribution:

$$q(\boldsymbol{\beta}') = \prod_{k=1}^{K-1} \text{Beta}(\beta'_k | u_k, v_k),$$

where the number of global atoms is truncated at K, thereby $q(\beta'_K = 1) = 1$. For the word related parameters $\Theta^{(w)}$, we have

$$\begin{aligned} q(\Theta^{(w)}) &= q(\boldsymbol{c}^{(w)})q(\boldsymbol{z}^{(w)})q(\boldsymbol{\pi}'^{(w)})q(\boldsymbol{\phi}), \\ q(\boldsymbol{c}^{(w)}) &= \prod_{d=1}^{D} \prod_{t=1}^{T-1} \text{Mult}(\boldsymbol{\varphi}_{dt}^{(w)}), \\ q(\boldsymbol{z}^{(w)}) &= \prod_{d=1}^{D} \prod_{n=1}^{N_d^{(w)}} \text{Mult}(\varsigma_{dt}^{(w)}), \\ q(\boldsymbol{\pi}'^{(w)}) &= \prod_{d=1}^{D} \prod_{t=1}^{T-1} \text{Beta}(\pi'_{dt} | a_{dt}^{(w)}, b_{dt}^{(w)}), \\ q(\boldsymbol{\phi}) &= \prod_{k=1}^{K} \text{Dir}(\boldsymbol{\lambda}_k). \end{aligned}$$

The variational distributions for entity related parameters have a similar form to the above distributions, except the Gaussian mixture parameters, which are expressed as follows:

$$q(\boldsymbol{\mu}_k) = \text{N}\big(\boldsymbol{m}_k, (\rho_k \boldsymbol{\Lambda}_k)^{-1}\big), \quad q(\boldsymbol{\Lambda}_k) = \text{Wishart}(\nu_k, \boldsymbol{W}_k).$$

In standard variational inference theory, the evidence lower bound (ELBO), which is the lower bound to the marginal log likelihood of the observed data, is maximized to find the best variational approximation to the true intractable posterior. Given the modeling framework of TMKGE, the ELBO can be written as

$$\begin{aligned} \mathcal{L}(q) = \sum_d \Bigg\{ &\mathbb{E}\Big[\log\Big(p(\boldsymbol{w}_d | \boldsymbol{c}_d^{(w)}, \boldsymbol{z}_d^{(w)}, \boldsymbol{\phi}) p(\boldsymbol{c}_d^{(w)} | \boldsymbol{\beta}') \\ &\times p(\boldsymbol{z}_d^{(w)} | \boldsymbol{\pi}'^{(w)}_d) p(\boldsymbol{e}_d | \boldsymbol{c}_d^{(e)}, \boldsymbol{z}_d^{(e)}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \\ &\times p(\boldsymbol{c}_d^{(e)} | \boldsymbol{\beta}') p(\boldsymbol{z}_d^{(e)} | \boldsymbol{\pi}'^{(e)}_d) p(\boldsymbol{\pi}'^{(w)}_d | \alpha_0) p(\boldsymbol{\pi}'^{(e)}_d | \alpha_0)\Big)\Big] \\ &+ H\big(q(\boldsymbol{c}_d^{(w)})\big) + H\big(q(\boldsymbol{z}_d^{(w)})\big) + H\big(q(\boldsymbol{\pi}'^{(w)}_d)\big) \\ &+ H\big(q(\boldsymbol{c}_d^{(e)})\big) + H\big(q(\boldsymbol{z}_d^{(e)})\big) + H\big(q(\boldsymbol{\pi}'^{(e)}_d)\big) \Bigg\} \\ &+ \mathbb{E}\Big[\log\Big(p(\boldsymbol{\beta}')p(\boldsymbol{\phi})p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\Big)\Big] + H\big(q(\boldsymbol{\beta}')\big) \\ &+ H\big(q(\boldsymbol{\phi})\big) + H\big(q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\big), \end{aligned}$$

where $H(\cdot)$ is the entropy term for variational distribution. By taking derivatives of this lower bound with respect to each variational parameter, we derive the coordinate ascent update steps.

We develop an online variational inference for TMKGE, to process large datasets (Wang et al., 2011; Hoffman et al., 2010). Given the existing corpus-level parameters, first a document d is sampled and then its optimal document-level variational parameters are computed. For word related variational parameters, these updates include

$$\begin{aligned} a_{dt}^{(w)} &= 1 + \sum_n \varsigma_{dnt}^{(w)}, \\ b_{dt}^{(w)} &= \alpha_0 + \sum_n \sum_{s=t+1}^{T} \varsigma_{dns}^{(w)}, \\ \varphi_{dtk}^{(w)} &\propto \exp\Big(\sum_n \varsigma_{dns}^{(w)} \mathbb{E}_q\big[\log p(w_{dn}|\boldsymbol{\phi}_k)\big] \mathbb{E}_q\big[\log \beta_k\big]\Big), \\ \varsigma_{dnt}^{(w)} &\propto \exp\Big(\sum_k \varphi_{dtk}^{(w)} \mathbb{E}_q\big[\log p(w_{dn}|\boldsymbol{\phi}_k)\big] \mathbb{E}_q\big[\log \pi_{dt}^{(w)}\big]\Big) \end{aligned} \tag{3}$$

where expectations are with respect to variational distributions and have closed forms. For entity related variational parameters, similar updates can be derived, with the term $\mathbb{E}_q\big[\log p(\boldsymbol{e}_{dm}|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)\big]$ replacing $\mathbb{E}_q\big[\log p(w_{dn}|\boldsymbol{\phi}_k)\big]$. Following Wang et al. (2011), for the corpus-level variational parameters, we use the following gradients:

$$\begin{aligned} \partial\lambda_{kv} &= -\lambda_{kv} + \eta + D\sum_t \varphi_{dtk}^{(w)} \big(\sum_n \varsigma_{dnt}^{(w)} I[w_{dn} = v]\big), \\ \partial\boldsymbol{m}_k &= -\boldsymbol{m}_k + \frac{D\sum_{m,t} \varphi_{dtk}^{(e)} \varsigma_{dmt}^{(e)} \boldsymbol{e}_{dm} + \rho_0 \boldsymbol{m}_0}{Dr_k + \rho_0}, \\ \partial\rho_k &= -\rho_k + \rho_0 + Dr_k, \\ \partial\nu_k &= -\nu_k + \nu_0 + Dr_k, \\ \partial\boldsymbol{W}_k &= -\boldsymbol{W}_k + \Big(\boldsymbol{W}_0^{-1} + D\sum_{m,t} \varphi_{dtk}^{(e)} \varsigma_{dmt}^{(e)} \boldsymbol{e}_{dm} \boldsymbol{e}_{dm}^T\Big)^{-1}, \\ \partial u_k &= -u_k + 1 + D\sum_t \big(\varphi_{dtk}^{(w)} + \varphi_{dtk}^{(e)}\big), \\ \partial v_k &= -v_k + \gamma_0 + D\sum_t \sum_{\ell=k+1}^{K} \big(\varphi_{dt\ell}^{(w)} + \varphi_{dt\ell}^{(e)}\big), \end{aligned} \tag{4}$$

where r_k is defined as $\sum_{m,t} \varphi_{dtk}^{(e)} \varsigma_{dmt}^{(e)}$. The corpus-level parameters are then updated using these gradients (among them, the first, the fifth and the sixth are natural gradients while the other four are approximations from the posterior of Gaussian Wishart scale matrix W. It appears difficult to obtain natural gradients for those four.) and a learning rate parameter ϵ_t. For instance, for topic-words distribution parameters we have

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \epsilon_{t_0} \partial\boldsymbol{\lambda}. \tag{5}$$

			number of top words and PMI scores					
model	parameters	data source	5	10	15	20	25	30
TMKGE	K=300, T=20	20 Newsgroups	**20.8**	91.1	210.0	380.0	**602.0**	**876.0**
HDP	K=300, T=20		20.0	**91.6**	**212.6**	**384.1**	598.4	868.7
LDA	K=100		13.5	64.6	163.4	285.0	455.2	671.1
KGE-LDA	K=30		18.9	69.8	187.5	320.6	482.7	616.5
TMKGE	K=300, T=20	NIPS	16.6	**97.1**	160.3	**299.6**	**474.5**	**685.5**
HDP	K=300, T=20		**16.7**	66.8	157.2	280.2	444.0	643.1
LDA	K=100		13.9	67.6	**161.9**	297.0	471.2	681.1
KGE-LDA	K=30		14.3	97.2	163.4	285.3	453.3	645.4
TMKGE	K=300, T=20	Ohsumed	**21.6**	**123.3**	**237.3**	**407.7**	**624.2**	**895.5**
HDP	K=300, T=20		15.6	70.7	168.2	338.9	582.9	864.9
LDA	K=100		11.9	65.6	131.9	257.0	481.2	691.1
KGE-LDA	K=30		15.6	116.5	185.4	354.2	585.4	795.6

Table 1: Topic Coherence of all models on three datasets with different number of top words. A higher PMI score implies a more coherent topic. Improvements of TMKGE over other methods are significant.

The rest of corpus-level variational parameters in (4) can be similarly updated. To ensure that the parameters converge to a stationary point, the learning rate satisfies (Hoffman et al., 2010; Sato, 2001) $\sum_{t_0=1}^{\infty} \epsilon_{t_0} = \infty$ and $\sum_{t_0=1}^{\infty} \epsilon_{t_0}^2 < \infty$.

Following Wang et al. (2011), we use $\epsilon_{t_0} = (\tau_0 + t_0)^{-\kappa}$, where $\kappa \in (0.5, 1]$ and $\tau_0 > 0$. To improve the stability of online variational inference, we use a mini-batch of documents to compute the natural gradients. That is, the contribution of the single document d in (4) is replaced by sum of contributions of documents in the mini-batch $\mathcal{S}$, and the factor D is replaced by $D/|\mathcal{S}|$. The overall scheme of online variational inference for TMKGE is shown in Algorithm 1.

Algorithm 1 Online variational inference for the proposed TMKGE framework.

```
Initialize corpus-level variational parameters.
while Stopping criterion is not met do
    Sample a random document $d$ from the corpus.
    Update $\boldsymbol{a}_d^{(w)}$, $\boldsymbol{b}_d^{(w)}$, $\boldsymbol{\varphi}_d^{(w)}$ and $\boldsymbol{\varsigma}_d^{(w)}$ using (3).
    Update $\boldsymbol{a}_d^{(e)}$, $\boldsymbol{b}_d^{(e)}$, $\boldsymbol{\varphi}_d^{(e)}$ and $\boldsymbol{\varsigma}_d^{(e)}$ similar to (3).
    Compute the natural gradients using (4).
    Set $\epsilon_{t_0} = (\tau_0 + t_0)^{-\kappa}$ and $t_0 \leftarrow t_0 + 1$.
    Update all corpus-level parameters as (5).
end
```

4 Experiments

We evaluate TMKGE on two experimental tasks and compare its performance to those of LDA, HDP and KGE-LDA. For LDA and HDP, we use the online variational inference implementations. More precisely, we will evaluate our framework by the test whether it finds coherent and meaningful topics and the test whether it can achieve good performance in document classification.

We run our experiments on three popular datasets; 20 Newsgroups, NIPS and the Ohsumed corpus. The 20 Newsgroups dataset contains 18,846 documents evenly categorized into 20 different categories.

The NIPS dataset contains 1,740 papers from the NIPS conference. The Ohsumed corpus is from the MEDLINE database. We consider the 13,929 unique Cardiovascular diseases abstracts in the first 20,000 abstracts of the year 1996. Each document in the set has one or more associated categories from the 23 disease categories. The documents belonging to multiple categories are eliminated so that 7,400 documents belonging to only one category remain. The datasets are tokenized with Stanford CoreNLP (Manning et al., 2014). After standard pre-processing (such as removing stop words), there are 20,881 distinct words in the 20 Newsgroups dataset, 14,482 distinct words in the NIPS dataset and 8,446 distinct words in the Ohsumed dataset.

4.1 External knowledge source

The knowledge graph we employ is WordNet (Miller, 1995). WordNet is a large lexical knowledge graph. Entities in WordNet are synonyms which express distinct concepts. Relations in WordNet mainly involve conceptual-semantic and lexical relations. We use a subset of WordNet (WN18) introduced in Bordes et al. (2011) and employed in Yao et al. (2017) as well. WN18 contains 151,442 triplets with 40,943 entities and 18 relations. We link tokenized words to entities in WN18 via NLTK (Bird and Loper, 2004).

20 Newsgroups			NIPS			Ohsumed		
lord God elohim jesus subject israel armenian christ john group	tcp/ip drive system computer information space windows data message software	kuwait iraq kuwaiti sabah abdulla gulf amir ahmed sheikh saudi	distribution gaussian posterior covariance ensemble matrix KL divergence approximate algorithm	network learning model neural data figure information units problem recognition	tube regression svs support fraction erros vapnik algorithm smola vector	vietnam veterans mthfr income white proportion drinking era lifetime interview	hemagglutinin anti-tumor mthfr tumor pbl antibody meh ab test verapamil radioactivity	shbg patients globulin testicular hormone levels group sex binding treatment
101.2	98.5	119.3	152.6	91.1	106.3	105.4	135.4	152.2
20 Newsgroups			NIPS			Ohsumed		
internet mail email list message address fax network send e-mail	drive windows dos card disk mac scsi memory system apple	car cars engine oil miles dealer speed buy ford drive	distribution bayesian gaussian prior posterior probability variables markov distribution approximation	control trajectory robot controller arm model forward motor trajectories inverse	kernel support xi vector margin examples set kernels svm machines	gene dna protein region genetic analysis mutation sequence molecular mrna	cancer tumor survival tumors carcinoma breast stage malignant chemotherapy primary	treatment therapy dose drug effects placebo trial oral mg effective
89.2	84.4	63.6	154.8	86.2	88.3	149.9	107.7	106.5

Table 2: Example topics learned from three datasets by TMKGE with $K = 300$ and $T = 20$, and KGE-LDA with $K = 30$. The last row for each model is the topic coherence computed using the 4,776,093 Wikipedia documents as reference. Some medical short words: pbl = Peripheral blood leucocyte, meh = Mean erythrocyte hemoglobin.

4.2 Model parameters

In the experiments, for each method, we report the results based on the hyperparameter settings that obtain the best performances. For TMKGE and HDP, we report the results for $K = 300$, $T = 20$ and $K = 100$, $T = 10$ cases. For LDA and KGE-LDA, respectively, we have $K = 100$ and $K = 30$. Throughout this work we fix the dimension of entity embedding as $P = 5$. For online variational inference, we run the algorithms for 1000 iterations, with mini-batch size of 100.

4.3 Topic Coherence

We assess the performance of the proposed TMKGE model based on topic coherence. Topic coherence has been shown to be more consistent with human judgment than other typical topic model metrics such as perplexity (Chang et al., 2009; Newman et al., 2010). We perform both quantitative and qualitative analysis of the topics discovered by TMKGE, and compare its performance to those of LDA, HDP and KGE-LDA.

4.3.1 Quantitative Analysis

We evaluate the coherence of discovered topics by the point-wise mutual information (PMI) Topic Coherence metric. The PMI Topic Coherence is implemented following Newman et al. (2010):

$$PMI(k) = \sum_{j=2}^{N} \sum_{i=1}^{j-1} \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

where k refers to a topic, N refers to the number of top words of k, $p(w_i)$ is the probability that w_i appears in a document, $p(w_i, w_j)$ is the probability that w_i and w_j co-occur in the same document. A higher PMI score implies a more coherent topic. Following KGE-LDA, 4,776,093 Wikipedia articles are employed for obtaining topic coherence scores. Different from Yao et al. (2017), which used a fixed value of N (the number of top words, e.g. $N = 5$ or $N = 10$), we vary N in a range from 5 to 30. (Lau and Baldwin, 2016) suggests that calculating topic coherence over several different cardinalities and averaging results in a substantially more stable evaluation.

Table 1 shows the average topic coherence for different methods and datasets. We can observe that for the three datasets, TMKGE achieves highest topic coherence in almost all top word sizes. In the few cases which TMKGE does not rank highest, there only exist subtle differences with the top performing result. This shows that knowledge graph embedding improves the coherence of discovered topics. Further, for the top 10 words, the topic coherence of all three datasets are higher than those obtained by KGE-LDA. This shows that topic modeling based on HDP for both entity embedding and words enjoys incomparable advantages over LDA-based modeling.

4.3.2 Qualitative Analysis

Table 2 shows example topics with their PMI scores learned from the three corpora by KGE-LDA and our TMKGE model. For comparison, we report similar topics to those listed in the KGE-LDA paper. It can be seen that TMKGE finds quite closely related words in a topic. For example, for the second column of 20 Newsgroups, topic words from both TMKGE and KGE-LDA are related to computers. However, it can be noted that words from TMKGE focus more on the core words of computer science. In contrast, words from the same topic in KGE-LDA seems to be closer to the brand, such as windows, mac or apple. In addition, topics found from TMKGE are more diverse than those found in KGE-LDA. For 20 Newsgroups, the three topics we list here refer to theology, computer science and middle east respectively while the three topics from KGE-LDA refer to internet, computer and car respectively. Both TMKGE and KGE-LDA discover probability-related and machine learning topics with different top words from NIPS dataset. Roughly speaking, KGE-LDA discovers gene-related, cancer-related and treatment-related topics from Ohsumed corpus. TMKGE discovers more diverse and more specific topics. For example, one topic TMKGE discovers is about Vietnamese veterans, cancer-related and sexual-disease topics. From the perspective of topic coherence, we can also see that TMKGE obtains higher PMI score in most of those topics. The whole trend is consistent with the average PMI score reported in the last section. Overall, TMKGE performs better than other topic models, including LDA, HDP and KGE-LDA in terms of average PMI and also in qualitative case studies.

4.4 Document Classification

We evaluate our proposed method through document classification, we follow the approach in (Li and McCallum, 2006) for document classification.

We have conducted a five-way classification on the `comp` subject of 20 Newsgroups dataset and on the top five most frequent labels of Ohsumed dataset (no labels for nips dataset), where each class of documents is divided into 75% training and 25% testing. For each class, the LDA, HDP and TMKGE models are trained on the training documents, and then the predictive likelihood for the test documents is calculated using the E-step in the variational inference procedure of LDA. A document is classified correctly if its corresponding model produces the highest likelihood.

class	LDA	HDP	KGE-LDA	TMKGE
20 Newsgroup				
pc	68.6	78.9	67.2	**78.9**
os	71.7	80.7	70.7	**82.3**
mac	82.0	**87.1**	68.1	86.5
windows.x	84.0	83.5	64.4	**84.9**
graphics	81.2	81.9	65.4	**83.0**
Ohsumed				
C04	50.6	73.0	59.1	**73.8**
C10	46.2	63.0	54.4	**64.9**
C14	51.5	44.6	33.2	**52.3**
C21	86.5	89.5	83.7	**89.7**
C23	68.2	81.9	75.3	**86.1**

Table 3: Document classification accuracy a five-way classification on the comp subject of 20 Newsgroups dataset and on the top five most frequent labels of ohsumed dataset (no labels for NIPS dataset).

Table 3 presents the average classification accuracy for TMKGE, HDP and LDA over five repeated simulations. The table includes the classification accuracy for KGE-LDA, where the learned topic proportions are used as features for SVM classifier. For the majority of document classes, TMKGE has the best classification accuracy, except for the class `mac`. As shown, the SVM classifier based on KGE-LDA has significantly worst performance. For more complete comparisons, we run experiments on all subjects of 20 Newsgroups and also report experimental results published in Shi et al. (2017) in Table 4. TMKGE achieves the best performance on all models.

Model	Acc (%)	Model	Acc (%)
BOW	79.7	STE-Diff	82.9
Skip-Gram	75.4	LDA	77.5
TWE	81.5	TMSA	83.5
PV	75.4	HDP	82.4
GPU-DMM	48.0	KGE-LDA	70.5
STE-Same	80.4	TMKGE	**88.8**

Table 4: Document classification: all subjects of 20 Newsgroups dataset for more complete comparisons. Clearly shown is the best performances of TMKGE

A few points can be observed from the superior performance of TMKGE. Firstly, it looks the addition of unnormalized knowledge graph embedding into TMKGE as a proportional vector to the word vector boosts the performance. Secondly, the selection of HDP over LDA plays an essential role. This can be indicated from the poor performance of KGE-LDA (which is even worse than BOW). More impressively, TMKGE achieves even much better performances than STE-Diff, TWE and TMSA, all of which involve the integration of word embedding and topic modeling. Impressively, TMKGE shows its supremacy over the state of the art model, TMSA with high margins. This shows that the knowledge graph structure included into the entity embedding conveys more information than pure word embedding. Meanwhile, this also shows that the two proportional vectors generated with online HDP enables the flexible sharing of information between words and entities. Accordingly, more coherent topics are extracted and the classification result are boosted as well.

5 Conclusion

This paper presents TMKGE, a Bayesian nonparametric model based on hierarchical Dirichlet process for incorporation of entity embeddings from external knowledge graphs into topic modeling. The proposed method allows for flexible sharing of information between documents and knowledge graph. Specifically, TMKGE avoids forcing the words and entities to identical latent factors, thus making it a suitable framework for scenarios where only partial relational information are available. Furthermore, as a Bayesian nonparameteric model, TMKGE learns the number of word topics and entity mixture components automatically from the data. We have derived an efficient and scalable online variational inference for TMKGE.

Comprehensive experiments on three different datasets suggest that TMKGE significantly outperforms SOA methods in terms of both topic coherence and document classification accuracy.

References

David Andrzejewski and Xiaojin Zhu. 2009. Latent dirichlet allocation with topic-in-set knowledge. In *SemiSupLearn '09 Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 43–48, Boulder, Colorado.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th annual international conference on machine learning (ICML)*, pages 25–32, Montreal, Canada.

Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. 2016. Nonparametric spherical topic modeling with word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Arnim Bleier. 2013. Practical collapsed stochastic variational inference for the HDP. *CoRR*, abs/1312.0412.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795, Lake Tahoe, NV.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, San Francisco, CA.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 288–296, Vancouver, Canada.

Chaitanya Chemudugunta, America Holloway, Padhraic Smyth, and Mark Steyvers. 2008. Modeling documents by combining semantic concepts with unsupervised statistical learning. In *Proceedings of the 7th International Semantic Web Conference (ISWC)*, pages 229–244, Karlsruhe, Germany.

Zhiyuan Chen and Bing Liu. 2014. Mining topics in documents: standing on the shoulders of big data. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1116–1125, New York, NY.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Discovering coherent topics using general knowledge. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM)*, pages 209–218, San Francisco, CA.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL)*, pages 795–804, Beijing, China.

Finale Doshi-Velez, Byron C Wallace, and Ryan Adams. 2015. Graph-sparse LDA: A topic model with structured sparsity. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2575–2581, Austin, TX.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. 2011. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, pages 1020–1056.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 856–864, Vancouver, Canada.

Thomas Hofmann. 2017. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*, pages 211–218. ACM.

Changwei Hu, Piyush Rai, and Lawrence Carin. 2016. Topic-based embeddings for learning from large knowledge graphs. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1133–1141, Cadiz, Spain.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine learning*, 95(3):423–469.

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. 2012. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 204–213, Avignon, France.

Jey Han Lau and Timothy Baldwin. 2016. The sensitivity of topic coherence evaluation to topic cardinality. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 483–487, San Diego, CA.

Dingcheng Li, Jingyuan Zhang, and Ping Li. 2019. Tmsa: A mutual learning model for topic discovery and wordembedding. In *Proceedings of the SIAM conference on Data Mining (SDM)*, Calgary, Canada.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative topic embedding: a continuous representation of documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 666–675, Berlin, Germany.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 577–584, Pittsburgh, PA.

Steven N MacEachern and Peter Müller. 1998. Estimating mixture of dirichlet process models. *Journal of Computational and Graphical Statistics*, 7(2):223–238.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics (ACL): system demonstrations*, pages 55–60, Baltimore, MD.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe, NV.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 262–272, Edinburgh, UK.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 100–108, Los Angeles, LA.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Jim Pitman. 2002. Poisson–dirichlet and gem invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability and Computing*, 11(5):501–514.

Masa-Aki Sato. 2001. Online model selection based on the variational bayes. *Neural computation*, 13(7):1649–1681.

Jayaram Sethuraman. 1994. A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650.

Bei Shi, Wai Lam, Shoaib Jameel, Steven Schockaert, and Kwun Ping Lai. 2017. Jointly learning word embeddings and latent topics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 375–384, Shinjuku, Tokyo.

Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems (NIPS)*, pages 1385–1392, Vancouver, Canada.

Yee W Teh, Kenichi Kurihara, and Max Welling. 2008. Collapsed variational inference for HDP. In *Advances in neural information processing systems (NIPS)*, pages 1481–1488, Vancouver, Canada.

Chong Wang, John Paisley, and David Blei. 2011. Online variational inference for the hierarchical dirichlet process. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 752–760, Fort Lauderdale, FL.

Liang Yao, Yin Zhang, Baogang Wei, Zhe Jin, Rui Zhang, Yangyang Zhang, and Qinfei Chen. 2017. Incorporating knowledge graph embeddings into topic modeling. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 3119–3126, San Francisco, CA.

He Zhao, Lan Du, Wray L. Buntine, and Gang Liu. 2017. Metalda: A topic model that efficiently incorporates meta information. In *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*, pages 635–644, New Orleans, LA.

Correlation Coefficients and Semantic Textual Similarity

Vitalii Zhelezniak, Aleksandar Savkov, April Shen & Nils Y. Hammerla
Babylon Health
`{firstname.lastname}@babylonhealth.com`

Abstract

A large body of research into semantic textual similarity has focused on constructing state-of-the-art embeddings using sophisticated modelling, careful choice of learning signals and many clever tricks. By contrast, little attention has been devoted to similarity measures between these embeddings, with cosine similarity being used unquestionably in the majority of cases. In this work, we illustrate that for all common word vectors, cosine similarity is essentially equivalent to the Pearson correlation coefficient, which provides some justification for its use. We thoroughly characterise cases where Pearson correlation (and thus cosine similarity) is unfit as similarity measure. Importantly, we show that Pearson correlation is appropriate for some word vectors but not others. When it is not appropriate, we illustrate how common non-parametric rank correlation coefficients can be used instead to significantly improve performance. We support our analysis with a series of evaluations on word-level and sentence-level semantic textual similarity benchmarks. On the latter, we show that even the simplest averaged word vectors compared by rank correlation easily rival the strongest deep representations compared by cosine similarity.

1 Introduction

Textual embeddings are immensely popular because they help us reason about the abstract and fuzzy notion of semantic similarity in purely geometric terms. Distributed representations of words in particular (Bengio et al., 2003; Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2017; Joulin et al., 2017) have had a massive impact on machine learning (ML), natural language processing (NLP), and information retrieval (IR).

Recently, much effort has also been directed towards learning representations for larger pieces of text, with methods ranging from clever compositions of word embeddings (Mitchell and Lapata, 2008; De Boom et al., 2016; Arora et al., 2017; Wieting et al., 2016; Wieting and Gimpel, 2018; Zhelezniak et al., 2019) to sophisticated neural architectures (Le and Mikolov, 2014; Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017; Gan et al., 2017; Tang et al., 2017; Zhelezniak et al., 2018; Subramanian et al., 2018; Pagliardini et al., 2018; Cer et al., 2018).

Comparatively, there is little research into similarity measures for textual embeddings. Despite some investigations into alternatives (Camacho-Collados et al., 2015; De Boom et al., 2015; Santus et al., 2018; Zhelezniak et al., 2019), cosine similarity has persistently remained the default and unquestioned choice across the field. This is partly because cosine similarity is very convenient and easy to understand. Sometimes, however, we have to resist what is convenient and instead use what is appropriate. The core idea behind our work is to treat each word or sentence embedding as a sample of (e.g. 300) observations from some scalar random variable. Hence, no matter how mysterious word vectors appear to be, just like any samples, they become subject to the full power of traditional statistical analysis. We first show that in practice, the widely used cosine similarity is nothing but the Pearson correlation coefficient computed from the paired sample. However, Pearson's r is extremely sensitive to even slight departures from normality, where a single outlier can conceal the underlying association. For example, we find that Pearson's r (and thus cosine similarity) is acceptable for word2vec and fastText but not for GloVe embeddings. Perhaps surprisingly, when we average word vectors to represent sentences, cosine similarity remains acceptable for word2vec, but not for fastText any longer. We show that this seemingly counterintuitive behaviour can be predicted

Proceedings of NAACL-HLT 2019, pages 951–962
Minneapolis, Minnesota, June 2 - June 7, 2019.

by elementary univariate statistics, something that is already well known to researchers and practitioners alike. Furthermore, when there are clear indications against cosine similarity, we propose to repurpose rank-based correlation coefficients, such as Spearman's ρ and Kendall's τ, as *similarity measures* between textual embeddings. We support this proposition by a series of experiments on word- and sentence-level semantic textual similarity (STS) tasks. Our results confirm that rank-based correlation coefficients are much more effective when the majority of vectors break the assumptions of normality. Moreover, we show how even the simplest sentence embeddings (such as averaged word vectors) compared by rank correlation easily rival recent deep representations compared by cosine similarity.

2 Related Work

At the heart of our work is a simple statistical analysis of pre-trained word embeddings and exploration of various correlation coefficients as proxies for semantic textual similarity. Hence, any research that combines word embeddings with tools from probability and statistics is relevant. Of course, word embeddings themselves are typically obtained as the learned parameters of statistical machine learning models. These models can be trained on large corpora of text to predict a word from its context or vice versa (Mikolov et al., 2013a). Alternatively, there are also supervised approaches (Wieting et al., 2015, 2016; Wieting and Gimpel, 2017, 2018).

A different line of research tries to move away from learning word embeddings as point estimates and instead model words as parametric densities (Vilnis and McCallum, 2014; Barkan, 2017; Athiwaratkun and Wilson, 2017). These approaches are quite appealing because they incorporate semantic uncertainty directly into the representations. Of course, such representations need to be learned explicitly. In some cases one could estimate the densities even for off-the-shelf embeddings, but this still requires access to the training data and the usefulness of such post-factum densities is limited (Vilnis and McCallum, 2014). In other words, these approaches are not very helpful to practitioners who are accustomed to using high-quality pre-trained word embeddings directly.

Arguably, statistical analysis of pre-trained word embeddings is not as principled as applying a probabilistic treatment end-to-end. Any such analysis, however, is very valuable as it provides insights and justifications for methods that are already in widespread use. For example, removing the common mean vector and a few top principal components makes embeddings even stronger and is now a common practice (Mu and Viswanath, 2018; Arora et al., 2016, 2017; Ethayarajh, 2018). These works view word embeddings as observations from some D-dimensional distribution; such treatment is naturally suitable for studying the overall geometry of the embedding space. We, on the other hand, are interested in studying the similarities between individual word vectors and require a completely different perspective. To this end, we see each word embedding itself as a sample of D observations from a scalar random variable. It is precisely this shift in perspective that allows us to reason about semantic similarity in terms of correlations between random variables and make the connection to the widely used cosine similarity.

Finally, we propose using rank-based correlation coefficients when cosine similarity is not appropriate. Recently, Santus et al. (2018) introduced a rank-based similarity measure for word embeddings, called APSynP, and demonstrated its efficacy on outlier detection tasks. However, the results on the word-level similarity benchmarks were mixed, which, interestingly enough, could have been predicted in advance by our analysis.

3 Correlation Coefficients and Semantic Similarity

Suppose we have a vocabulary of N words $\mathcal{V} = \{w_1, w_2, \ldots, w_N\}$ and the word embeddings matrix $\mathbf{W} \in \mathbb{R}^{N \times D}$, where each row $\mathbf{w}^{(i)}$ for $i = 1, \ldots, N$ is a D-dimensional word vector. Popular pre-trained embeddings in practice typically have dimension $D = 300$, while the vocabulary size N can range from thousands to millions of words.

We now consider the following: what kinds of statistical analyses can we apply to $\mathbf{W}$ in order to model semantic similarity between words? One option is to view all word embeddings $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots \mathbf{w}^{(N)}$ as a sample of N observations from some D-variate distribution $P(E_1, \ldots E_D)$. For example, we can fit a Gaussian and study how all 300 dimensions correlate with each other. Perhaps we can fit a mixture model and see how the embeddings cluster. We

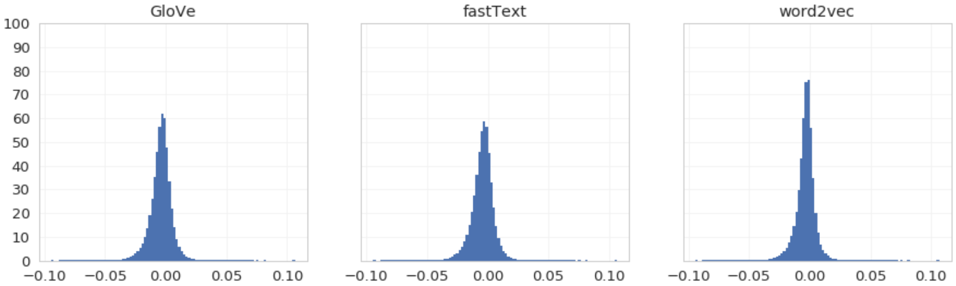


Figure 1: Normalised histograms of the mean distribution for three commonly used word embedding models: GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), and word2vec (Mikolov et al., 2013b,c).

could also normalise them and study their distribution on the unit sphere. It is clear by now that $P(E_1, \ldots, E_D)$ is suitable for describing the overall geometry of the embedding space but is not very useful for our goals.

If we are to reason about similarities between individual word vectors, we should instead be looking at the transpose of $\mathbf{W}$. Putting it differently, we see $\mathbf{W}^T$ as a sample of D observations from an N-variate distribution $P(W_1, W_2, \ldots, W_N)$, where W_i is a scalar random variable corresponding to the word w_i. This distribution is exactly what we need because the associations between W_i captured by P will become a proxy for semantic similarity. Often we are only interested in pairwise similarities between two given words w_i and w_j; thus the main object of our study is the bivariate marginal $P(W_i, W_j)$. To lighten up the notation slightly, we denote the two words as w_x and w_y, and the corresponding random variables as X and Y. We also refer to $P(X, Y)$ as the joint and $P(X)$, $P(Y)$ as the marginals. In practice, of course, the actual $P(X, Y)$ is unknown but we can make inferences about it based on our sample $(\mathbf{x}, \mathbf{y}) = \{(x_1, y_1), (x_2, y_2), \ldots (x_D, y_D)\}$.

First, we might want to study the degree of linear association between X and Y, so we compute the sample Pearson correlation coefficient

$$\hat{r} = \frac{\sum_{i=1}^{D}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{D}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{D}(y_i - \bar{y})^2}}, \quad (1)$$

where $\bar{x}$ and $\bar{y}$ are the sample means

$$\bar{x} = \sum_{i=1}^{D} x_i, \qquad \bar{y} = \sum_{i=1}^{D} y_i. \quad (2)$$

Let's view $\mathbf{x}$ and $\mathbf{y}$ as word embeddings momentarily and compute cosine similarity between them

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{D} x_i y_i}{\sqrt{\sum_{i=1}^{D} x_i^2}\sqrt{\sum_{i=1}^{D} y_i^2}}. \quad (3)$$

We see now that Equation (1) and Equation (3) look very similar; when the sample means $\bar{x}$, $\bar{y}$ are zero, cosine similarity and Pearson's $\hat{r}$ are equal. The real question here is whether or not they coincide in practice. Putting it differently, if we take any single word vector $\mathbf{w}$ and compute the mean (across the D dimensions), is this mean close to zero? It turns out that it is, and we can show this by plotting the distribution of the means across the whole vocabulary for various popular word embeddings (see Figure 1). We find that the means are indeed highly concentrated around zero; quantitatively, only 0.03% of them are above 0.05 in magnitude. It follows that in practice when we compute cosine similarity between word vectors, we are actually computing Pearson correlation between them.

However, is this always the right thing to do? When the joint $P(X, Y)$ is bivariate normal, Pearson correlation indeed provides a complete summary of association between X and Y, simply because the covariance is given by $\text{cov}(X, Y) = r_{XY}\sigma_X\sigma_Y$. However, Pearson correlation is extremely sensitive to even the slightest departures from normality – a single outlier can easily conceal the underlying association (Pernet et al., 2013). When the normality of $P(X, Y)$ is in doubt, it is preferable to use robust correlation coefficients such as Spearman's $\hat{\rho}$ or Kendall's $\hat{\tau}$.

Spearman's $\hat{\rho}$ is just a Pearson's $\hat{r}$ between ranked variables

$$\hat{\rho} = \frac{\sum_{i=1}^{D}(r[x_i] - \overline{r[x]})(r[y_i] - \overline{r[y]})}{\sqrt{\sum_{i=1}^{D}(r[x_i] - \overline{r[x]})^2}\sqrt{\sum_{i=1}^{D}(r[y_i] - \overline{r[y]})^2}}, \quad (4)$$

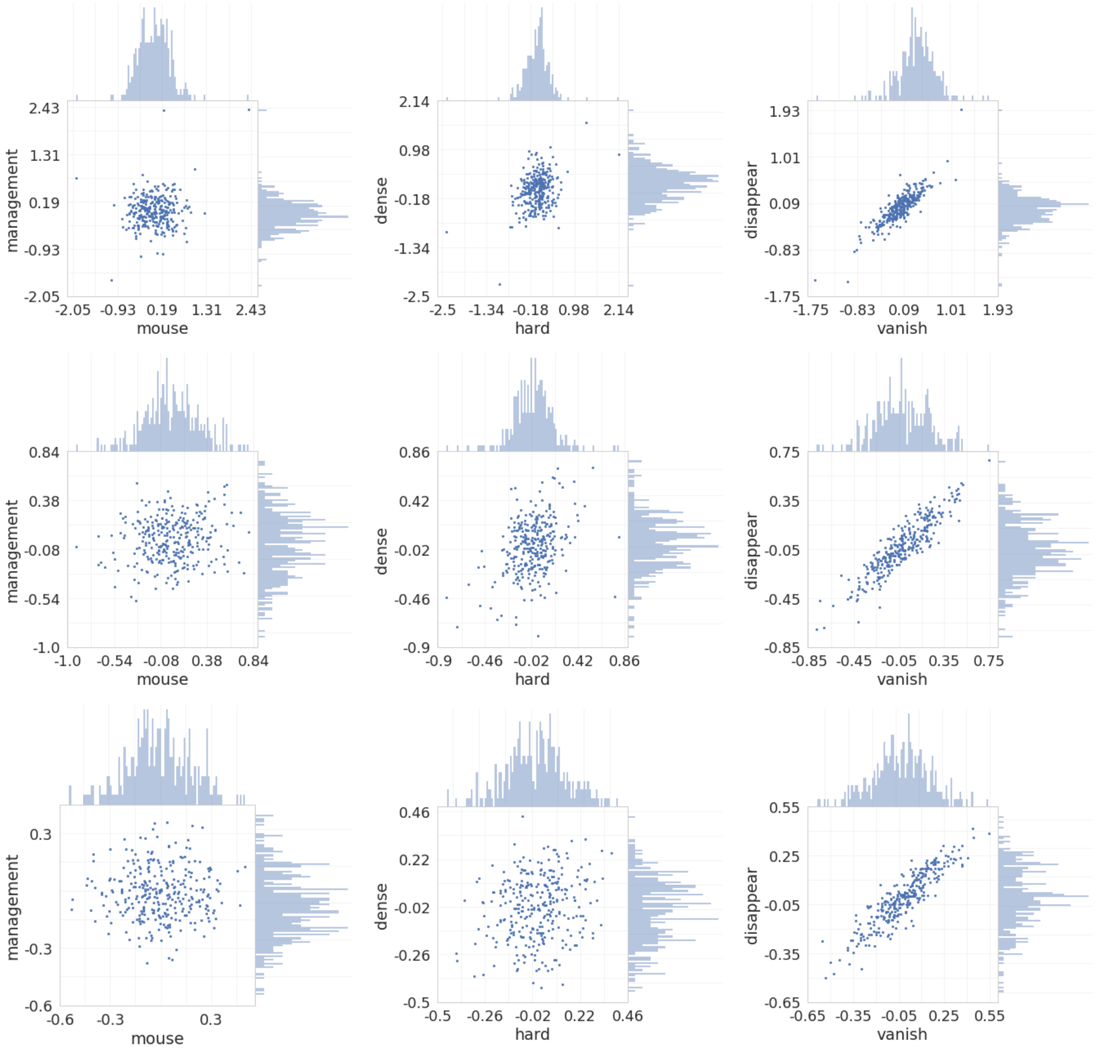


Figure 2: Scatter plots of paired word vectors, along with histograms (100 bins) of individual word vectors. Rows from top to bottom correspond to one of three common models: GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), and word2vec (Mikolov et al., 2013b,c). Columns from left to right correspond to increasing degrees of semantic similarity between the words, and accordingly increasingly pronounced linear correlation between the word vectors. Both the scatter plots and the histograms exhibit the presence of heavy outliers for GloVe vectors, which damage the efficacy of Pearson correlation in reliably capturing statistical associations. The outliers are relatively less pronounced for fastText vectors and much less pronounced for word2vec vectors.

where $r[x_i]$ denotes the integer rank of x_i in a vector $\mathbf{x}$ (similarly $r[y_i]$), while $\overline{r[x]}$ and $\overline{r[y]}$ denote the means of the ranks. Kendall's $\hat{\tau}$ is given by

$$\hat{\tau} = \frac{2}{D(D-1)} \sum_{i<j} \operatorname{sgn}(x_i - x_j)\operatorname{sgn}(y_i - y_j) \quad (5)$$

and can be interpreted as a normalised difference between the number of concordant pairs and the number of discordant pairs. These rank correlation coefficients are more robust to outliers than Pearson's $\hat{r}$ because they limit the effect of outliers to their ranks: no matter how far the outlier is, its rank cannot exceed D or fall below 1 in our case. There are also straightforward extensions to account for the ties in the ranks.

The main point here is the following. It is tempting to chose cosine similarity as the default and apply it everywhere regardless of the embedding type. Sometimes, however, we should resist using what is convenient and instead use what is appropriate. For example, if the samples corresponding to the marginals $P(X)$ and $P(Y)$ already look non-normal, then we conclude the joint $P(X, Y)$ cannot be a bivariate normal and the appropriateness of cosine similarity should be seriously questioned. In some of these cases, using a rank-based coefficient as a similarity measure be-

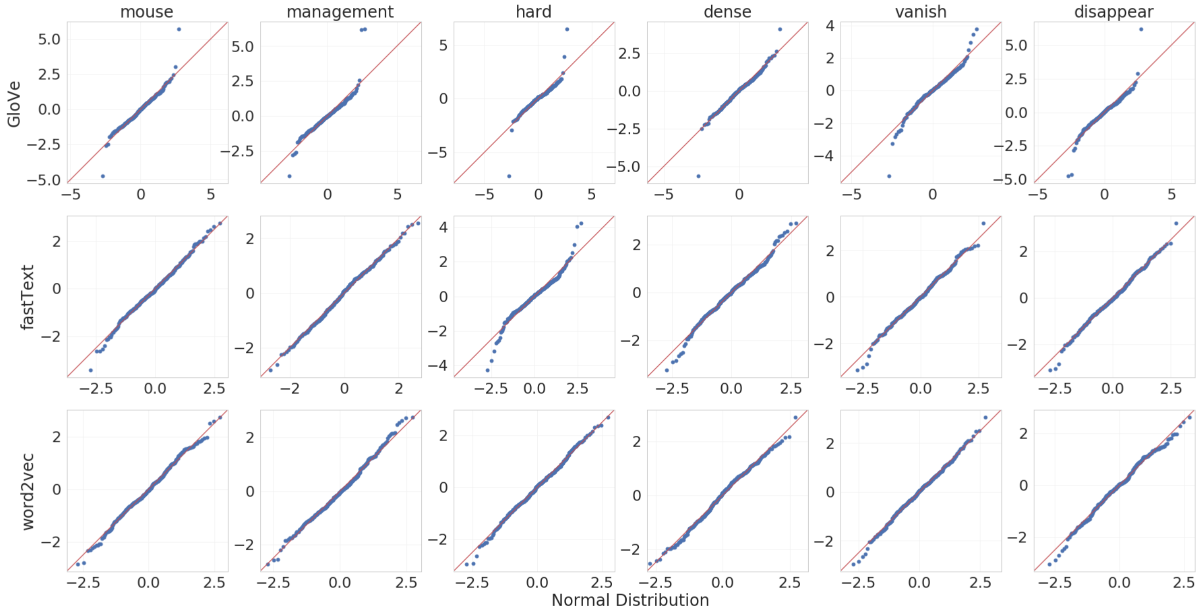


Figure 3: Q-Q plots comparing the theoretical quantiles of a standard normal distribution (horizontal axis) against the sample quantiles of standardised (Mean 0, SD 1) word vectors from three commonly used models: GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), and word2vec (Mikolov et al., 2013b,c). Perfect fit to the 45-degree reference line would indicate perfect normality. Note the pronounced discrepancy between the normal distribution and GloVe vectors due to the presence of heavy outliers. The discrepancy is relatively less pronounced for fastText vectors and much less pronounced for word2vec vectors. Figure 2 provides an alternative visualisation of the same phenomena.

tween word embeddings would be a much better alternative. It will capture the association better, which could in turn lead to large improvements in performance on the downstream tasks. In general, of course, even normal marginals do not imply a normal joint and care should be exercised either way; however we found the normality of marginals to be a good indication for cosine similarity within the scope of the present work. In the next section we illustrate how the ideas discussed here can be applied in practice.

4 Statistical Analysis of Word Embeddings: A Practical Example

No matter how mysterious word vectors appear to be, just like any samples, they are subject to the full power of traditional statistical analysis. As a concrete example, let's say we decided to use GloVe vectors (Pennington et al., 2014). We treat each vector $\mathbf{w}_i$ as if it was a sample of 300 observations from some scalar random variable W_i. We take a few hundred of these vectors, run a normality test such as Shapiro-Wilk (Shapiro and Wilk, 1965) and find that the majority of them look non-normal ($p < 0.05$). As there is a considerable evidence against normality, we flag these vectors as 'suspicious' and look at them closer. We pick a few vectors and examine their histograms and Q-Q plots, seen in Figure 2 and Figure 3 respectively; the latter in particular is a statistical tool used to compare empirical and theoretical data distributions, and is explained further in the caption of Figure 3. In both cases we observe that while the bulk of the distribution looks bell-shaped, we always get a couple of very prominent outliers.

Next, we can also visualise our word vectors in a way more directly relevant to the task at hand. We take some pairs of words that are very similar (e.g. 'vanish' and 'disappear'), moderately similar ('hard' and 'dense'), and completely dissimilar ('mouse' and 'management') and make the scatter plots for the corresponding pairs of word vectors. These are also presented in Figure 2. We see that for similar pairs the relationship is almost linear; it becomes less linear as the similarity decreases, until we see a spherical blob (no relationship) for the most dissimilar pair. However, we again face the presence of bivariate outliers that are too far away from the main bulk of points.

Given this evidence, which course of action shall we take? Based on the presence of heavy outliers, we reject the normality of GloVe vectors and rule out the use of Pearson's r and cosine similarity. Instead we can use rank correlation coefficients, such as Spearman's ρ or Kendall's τ, as they offer more robustness to outliers. Note that in this particular case, it may also be acceptable to winsorize (clip) the vectors and only then proceed with the standard Pearson's r. We evaluate the proposed solution on word-level similarity tasks and observe good improvement in performance over cosine similarity, as seen in Table 1.

Of course this exploration is in no way specific to GloVe vectors. Note that from Figure 2 and Figure 3, we also see that word2vec vectors in particular tend to be much more normally distributed, meaning that we don't find strong evidence against using Pearson correlation; this is again backed up by Table 1.

This example helps illustrate that proper statistical analysis applied to existing textual embeddings is extremely powerful and comparatively less time-consuming than inventing new approaches. Of course, this analysis can be made as fine-grained as desired. Quite coarsely, we could have rejected the use of cosine similarity right after the Shapiro-Wilk test; on the other hand, we could have used even more different tests and visualisations. The decision here rests with the practitioner and depends on the task and the domain.

5 Experiments

To empirically validate the utility of the statistical framework presented in Section 3, we run a set of evaluations on word- and sentence-level STS tasks. In all experiments we rely on the following publicly available word embeddings: GloVe (Pennington et al., 2014) trained on Common Crawl (840B tokens), fastText (Bojanowski et al., 2017) trained on Common Crawl (600B tokens), and word2vec (Mikolov et al., 2013b,c) trained on Google News. All the source code for our experiments is available on GitHub[1]; in the case of the sentence-level tasks we rely also on the SentEval toolkit (Conneau and Kiela, 2018).

	task	**N**	**V**	**COS**	**PRS**	**SPR**	**KEN**
GloVe	YP-130	.01	=	57.1	57.0	60.2	59.9
	MTURK-287	.13	=	69.3	69.3	70.8	70.9
	SIMLEX-999	.04	R	40.8	40.9	46.0	46.0
	MC-30	.10	=	78.6	79.2	77.0	77.4
	SIMVERB-3500	.04	R	28.3	28.3	34.3	34.3
	RG-65	.14	=	76.2	75.9	71.0	71.1
	WS-353-SIM	.06	=	80.3	80.2	80.1	80.1
	VERB-143	.00	=	34.1	33.9	37.8	37.4
	RW-STANFORD	.16	R	46.2	46.2	52.8	52.9
fastText	YP-130	.73	=	62.5	62.6	65.3	65.0
	MTURK-287	.88	=	72.6	72.7	73.4	73.3
	SIMLEX-999	.76	=	50.3	50.2	50.4	50.2
	MC-30	.90	=	85.2	85.2	84.6	84.5
	SIMVERB-3500	.68	=	42.6	42.6	42.6	42.5
	RG-65	.90	N	85.9	85.8	83.9	84.1
	WS-353-SIM	.84	N	84.0	83.8	82.4	82.2
	VERB-143	.21	=	44.7	44.9	43.8	44.3
	RW-STANFORD	.80	=	59.5	59.4	59.0	58.9
word2vec	YP-130	.95	=	55.9	56.1	55.0	54.7
	MTURK-287	.94	=	68.4	68.3	67.1	67.2
	SIMLEX-999	.94	=	44.2	44.2	43.9	44.0
	MC-30	.92	=	78.8	77.9	76.9	76.9
	SIMVERB-3500	.96	=	36.4	36.4	36.0	36.0
	RG-65	.94	=	75.0	74.3	73.9	74.2
	WS-353-SIM	.92	N	77.2	76.9	75.8	75.8
	VERB-143	.98	=	49.7	50.1	48.9	49.0
	RW-STANFORD	.95	N	53.4	53.5	52.5	52.5

Table 1: Spearman's ρ on word similarity tasks for combinations of word vectors and the following similarity metrics: cosine similarity (COS), Pearson's r (PRS), Spearman's ρ (SPR), and Kendall τ (KEN). **N** indicates the proportion of sentence vectors in a task for which the null hypothesis of normality in a Shapiro-Wilk test was *not* rejected at $\alpha = 0.05$. The **V** column indicates the type of the best performing method: a rank-based correlation coefficient (R), a non-rank-based correlation or measure (N), or a tie (=). The winners in **V** were determined by comparing the top rank-based method for that vector/task combination with the top non-rank-based method. Winners were assigned only when the difference was statistically significant as determined by 95% BCa confidence intervals.

First we consider a group of word-level similarity datasets that are commonly used as benchmarks in previous research: *WS-353-SIM* (Finkelstein et al., 2001), *YP-130* (Yang and Powers, 2005), *SIMLEX-999* (Hill et al., 2015), *SimVerb-3500* (Gerz et al., 2016), *RW-STANFORD* (Luong et al., 2013), *Verb-143* (Baker et al., 2014), *MTurk-287* (Radinsky et al., 2011), *MC-30* (Miller and Charles, 1991). These datasets contain pairs of words and a human-annotated similarity score for each pair. The success metric for the experiments is the Spearman correlation between the human-

---

[1] `https://github.com/Babylonpartners/corrsim`

annotated similarity scores and the scores generated by the algorithm. To avoid any confusion whatsoever, note that here Spearman correlation serves as an evaluation criterion; this is completely unrelated to using Spearman correlation as a similarity measure between word embeddings as proposed in Section 3. Bias-corrected and accelerated bootstrap (Efron, 1987) 95% confidence intervals were used to determine statistical significance. We report the results for different combinations of word vectors and similarity measures in Table 1. The main takeaways from these experiments are the following:

- There is no significant difference between the results obtained with cosine similarity and Pearson correlation. This is because empirically, the means across dimensions of these word vectors are approximately zero, in which case cosine similarity and Pearson correlation are approximately the same.
- Rank correlation coefficients tend to perform on par or better than cosine and Pearson on tasks and word vectors where there is a high proportion of non-normally distributed word vectors (over 90%). This makes sense because it is precisely in the non-normal cases where Pearson correlation fails.
- When word vectors seem mostly normal, our analysis does not tell us definitively whether cosine similarity or rank correlation should perform better, and indeed we see that cosine and Pearson perform on par or better than Spearman and Kendall.

In the second set of experiments, we use the datasets from the sentence-level Semantic Textual Similarity shared task series 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017). The success metric for these experiments is the Pearson correlation between the human-annotated sentence similarity scores and the scores generated by the algorithm. Again, this use of Pearson correlation as an evaluation criterion is completely unrelated to its use as a similarity measure between sentence embeddings. Note that the dataset for the STS13 SMT subtask is no longer publicly available, so the mean Pearson correlations reported in our experiments involving this task have been re-calculated accordingly.

For these experiments we use averaged word vectors as a sentence representation for various types of word vector, with similarity computed by the different correlation coefficients as well as cosine similarity and APSynP (Santus et al., 2018). We report these results in Table 2, and the full significance analysis for each subtask in Table 4. We also compare the top performing combination of averaged word vectors and correlation coefficient against several popular approaches from the literature that use cosine similarity: BoW with ELMo embeddings (Peters et al., 2018), Skip-Thought (Kiros et al., 2015), InferSent (Conneau et al., 2017), Universal Sentence Encoder with DAN and Transformer (Cer et al., 2018), and STN multitask embeddings (Subramanian et al., 2018). These results are presented in Table 3. Our observations for the sentence-level experiments are as follows:

	task	**N**	**COS**	**PRS**	**SPR**	**KEN**	**APS**
GloVe	STS12	.01	52.1	52.0	53.4	52.6	**53.8**
	STS13	.00	49.6	49.6	56.2	**56.7**	55.9
	STS14	.00	54.6	54.5	**63.2**	63.0	63.0
	STS15	.00	56.1	56.0	64.5	**65.3**	64.2
	STS16	.00	51.4	51.4	62.1	**63.7**	60.8
fastText	STS12	.01	58.3	58.3	**60.2**	59.0	58.4
	STS13	.01	57.9	58.0	65.1	**65.3**	61.8
	STS14	.00	64.9	65.0	**70.1**	69.6	68.5
	STS15	.00	67.6	67.6	74.4	**74.6**	72.7
	STS16	.00	64.3	64.3	73.0	**73.5**	70.7
word2vec	STS12	.95	51.6	51.6	51.7	**53.1**	45.3
	STS13	.94	58.2	**58.3**	57.9	58.2	57.2
	STS14	.96	**65.6**	**65.6**	65.5	**65.6**	64.1
	STS15	.96	67.5	67.5	67.3	**68.3**	66.5
	STS16	.96	64.7	64.7	64.6	**65.6**	63.9

Table 2: Mean Pearson correlation on STS tasks for methods using combinations of word vectors and similarity metrics. All methods use averaged word vectors to represent sentences. The similarity measures are: cosine similarity (COS), Pearson's r (PRS), Spearman's ρ (SPR), Kendall τ (KEN) and APSynP (APS). **N** indicates the proportion of sentence vectors in a task for which the null hypothesis of normality in a Shapiro-Wilk test was *not* rejected at $\alpha = 0.05$

- The conclusions from the word-level tasks continue to hold and are even more pronounced: in particular, cosine and Pearson are essentially equivalent, and the increase in performance of rank-based correlation coefficients over cosine similarity on non-normal sentence vectors is quite dramatic.

Approach	STS 12	13	14	15	16
ELMo (BoW)	55	53	63	68	60
Skip-Thought	41	29	40	46	52
InferSent	**61**	56	68	71	71
USE (DAN)	59	59	68	72	70
USE (Transformer)	**61**	64	**71**	74	**74**
STN (multitask)	60.6	54.7†	65.8	74.2	66.4
fastText - COS	58.3	57.9	64.9	67.6	64.3
fastText - SPR	60.2	65.1	70.1	74.4	73.0
fastText - KEN	59.0	**65.3**	69.6	**74.6**	73.5

Table 3: Mean Pearson correlation on STS tasks for a variety of methods in the literature compared to averaged fastText vectors with different similarity metrics: cosine similarity (COS), Spearman's ρ (SPR), and Kendall τ (KEN). Values in bold indicate best results per task. Previous results are taken from Perone et al. (2018) (only two significant figures provided) and Subramanian et al. (2018). † indicates the only STS13 result (to our knowledge) that includes the SMT subtask.

- Averaged word vectors compared with rank correlation easily rival modern deep representations compared with cosine similarity.

Finally, the fraction of non-normal word vectors used in sentence-level tasks is consistent with the results reported for the word-level tasks in Table 1. However, we observe the following curious phenomenon for fastText. While there is no evidence against normality for the majority of fastText vectors, perhaps surprisingly, when we average them to represent sentences, such sentence embeddings are almost entirely non-normal (Table 2). Empirically we observe that many high-frequency words or stopwords have prominently non-normal fastText vectors. Although stopwords constitute only a small fraction of the entire vocabulary, they are very likely to occur in any given sentence, thus rendering most sentence embeddings non-normal as well. While it's tempting to invoke the Central Limit Theorem (at least for longer sentences), under our formalism, averaging word vectors corresponds to averaging scalar random variables used to represent words, which are neither independent nor identically distributed. In other words, there are no easy guarantees of normality for such sentence vectors.

6 Discussion

In this work, we investigate statistical correlation coefficients as measures for semantic textual similarity and make the following contributions:

- We show that in practice, for commonly used word vectors, cosine similarity is equivalent to the Pearson correlation coefficient, motivating an alternative statistical view of word vectors as opposed to the geometric view, which is more prevalent in the literature.
- We illustrate via a concrete example the power and benefits of using elementary statistics to analyse word vectors.
- We characterise when Pearson correlation is applied inappropriately and show that these conditions hold for some word vectors but not others, providing a basis for deciding whether or not cosine similarity is a reasonable choice for measuring semantic similarity.
- We demonstrate that when Pearson correlation is not appropriate, non-parametric rank correlation coefficients, which are known to be more robust to various departures from normality, can be used as similarity measures to significantly improve performance on word- and sentence-level STS tasks.
- Finally, we show in particular that sentence representations consisting of averaged word vectors, when compared by rank correlation, can easily rival much more complicated representations compared by cosine similarity.

We hope that these contributions will inspire others to carefully investigate and understand alternative measures of similarity. This is particularly important in the realm of sentence representations, where there are many more complex ways of constructing sentence representations from word embeddings besides the simple averaging procedure tested here. It is worth exploring whether a more subtle application of rank correlation could help push these more complex sentence representations to even better performance on STS tasks.

A final and fascinating direction of future work is to explain the non-normality of certain types of word vectors (and in particular the presence of outliers) by analysing their training procedures. Preliminary investigations suggest that unsupervised

		GloVe			**fastText**			**word2vec**		
		SPR	**COS**	**Δ95% BCa CI**	**SPR**	**COS**	**Δ95% BCa CI**	**SPR**	**COS**	**Δ95% BCa CI**
STS12	MSRpar	35.90	**42.55**	[-10.74, -2.52]	**39.66**	**40.39**	[-3.22, 1.80]	38.79	**39.72**	[-1.77, -0.16]
	MSRvid	**68.80**	66.21	[1.31, 4.09]	**81.02**	73.77	[6.16, 8.53]	**77.88**	**78.11**	[-0.52, 0.06]
	SMTeuroparl	**48.73**	**48.36**	[-5.26, 6.48]	50.29	**53.03**	[-5.41, -0.17]	**16.96**	16.06	[0.21, 1.34]
	surprise.OnWN	**66.66**	57.03	[6.89, 12.76]	**73.15**	68.92	[2.19, 6.56]	**70.75**	**71.06**	[-0.73, 0.09]
	surprise.SMTnews	**47.12**	**46.27**	[-4.27, 5.50]	**56.67**	**55.20**	[-2.50, 5.50]	**53.93**	**52.91**	[-0.13, 2.09]
STS13	FNWN	**43.21**	**38.21**	[-0.54, 10.24]	**49.40**	39.83	[2.74, 16.46]	**40.73**	**41.22**	[-2.07, 1.07]
	headlines	**67.59**	63.39	[2.58, 5.89]	**71.53**	**70.83**	[-0.17, 1.58]	**65.48**	**65.22**	[-0.12, 0.66]
	OnWN	**57.66**	47.20	[8.10, 13.02]	**74.33**	63.03	[9.27, 13.50]	67.49	**68.29**	[-1.29, -0.33]
STS14	deft-forum	**39.03**	30.02	[5.24, 13.52]	**46.20**	40.19	[2.88, 10.00]	**42.95**	**42.66**	[-0.43, 1.03]
	deft-news	**68.99**	**64.95**	[-0.39, 8.72]	**73.08**	**71.15**	[-0.36, 4.39]	**67.33**	**67.28**	[-0.70, 0.91]
	headlines	**61.87**	58.67	[1.15, 5.48]	**66.33**	**66.03**	[-0.68, 1.28]	**62.09**	**61.88**	[-0.22, 0.66]
	images	**70.36**	62.38	[6.30, 10.00]	**80.51**	71.45	[7.44, 10.96]	76.98	**77.46**	[-0.89, -0.09]
	OnWN	**67.45**	57.71	[7.89, 11.97]	**79.37**	70.47	[7.42, 10.50]	74.69	**75.12**	[-0.81, -0.08]
	tweet-news	**71.23**	53.87	[13.98, 21.67]	**74.89**	70.18	[2.60, 7.21]	68.78	**69.26**	[-0.92, -0.01]
STS15	answers-forums	**50.25**	36.66	[10.18, 17.55]	**68.28**	56.91	[7.99, 15.23]	**53.74**	**53.95**	[-1.28, 0.86]
	answers-students	**69.99**	63.62	[4.25, 9.59]	**73.95**	71.81	[0.69, 3.56]	**72.45**	**72.78**	[-0.70, 0.04]
	belief	**58.77**	44.78	[10.11, 19.05]	**73.71**	60.62	[9.64, 19.50]	**61.73**	**61.89**	[-0.84, 0.46]
	headlines	**69.61**	66.21	[1.65, 5.29]	**72.93**	**72.53**	[-0.40, 1.20]	**68.58**	**68.72**	[-0.48, 0.23]
	images	**73.85**	69.09	[3.45, 6.29]	**83.18**	76.12	[5.76, 8.58]	**80.04**	**80.22**	[-0.55, 0.18]
STS16	answer-answer	**43.99**	40.12	[0.90, 7.36]	**54.51**	45.13	[5.14, 15.93]	**43.41**	**43.14**	[-1.03, 1.43]
	headlines	**67.05**	61.38	[2.43, 9.44]	**71.00**	**70.37**	[-0.93, 2.13]	**66.55**	**66.64**	[-0.66, 0.51]
	plagiarism	**72.25**	54.61	[12.69, 23.74]	**84.45**	74.49	[6.38, 14.81]	75.21	**76.46**	[-2.31, -0.37]
	postediting	**69.03**	53.88	[12.01, 19.06]	**82.73**	68.76	[7.55, 22.96]	**73.87**	**73.35**	[-0.08, 1.21]
	question-question	**58.32**	47.21	[7.02, 18.18]	**72.29**	62.62	[6.35, 13.64]	**63.94**	**63.74**	[-1.03, 1.38]

Table 4: Pearson correlations between human sentence similarity score and a generated score. Generated scores were produced via measuring Spearman correlation (SPR), as explained in Section 3, and cosine similarity (COS) between averaged word vectors. Values in bold represent the best result for a subtask given a set of word vectors, based on a 95% BCa confidence interval (Efron, 1987) on the differences between the two correlations. In cases of no significant difference, both values are in bold.

objectives based on the distributional hypothesis are probably not to blame, as word vectors trained without relying on the distributional hypothesis, such as those of Wieting et al. (2015), still exhibit non-normality to some degree. The actual causes remain to be determined. We believe that understanding the reasons for these empirically-observed characteristics of textual embeddings would be a significant step forwards in our overall understanding of these crucial building blocks for data-driven natural language processing.

Acknowledgements

We would like to thank Dan Busbridge for his useful comments and suggestions.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation*

(SemEval 2014), pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. *International Conference on Learning Representations*.

Ben Athiwaratkun and Andrew Wilson. 2017. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1645–1656. Association for Computational Linguistics.

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289. Association for Computational Linguistics.

Oren Barkan. 2017. Bayesian neural word embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3135–3143. AAAI Press.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Nasari: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

C. De Boom, S. Van Canneyt, S. Bohez, T. Demeester, and B. Dhoedt. 2015. Learning semantic similarity for very short texts. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1229–1234.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn. Lett.*, 80(C):150–156.

Bradley Efron. 1987. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline.

In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 91–100. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414, New York, NY, USA. ACM.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2400. Association for Computational Linguistics.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, pages 1188–1196.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. Association for Computational Linguistics.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Cyril R. Pernet, Rand Wilcox, and Guillaume A. Rousselet. 2013. Robust correlation analyses: False positive and power validation using a new open source matlab toolbox. *Frontiers in Psychology*, 3.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Enrico Santus, Hongmin Wang, Emmanuele Chersoni, and Yue Zhang. 2018. A rank-based similarity metric for word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 552–557. Association for Computational Linguistics.

S. S. Shapiro and M. B. Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations*.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2017. Exploring asymmetric encoder-decoder structure for context-based sentence representation learning. *CoRR*, abs/1710.10380.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR*, abs/1412.6623.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *International Conference on Learning Representations*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Dongqiang Yang and David M. W. Powers. 2005. Measuring semantic similarity in the taxonomy of wordnet. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38*, ACSC '05, pages 315–322, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Vitalii Zhelezniak, Dan Busbridge, April Shen, Samuel L. Smith, and Nils Y. Hammerla. 2018. Decoding Decoders: Finding Optimal Representation Spaces for Unsupervised Similarity Tasks. *CoRR*, abs/1805.03435.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019. Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In *International Conference on Learning Representations*.

Generating Token-Level Explanations for Natural Language Inference

James Thorne
University of Cambridge
jt719@cam.ac.uk

Andreas Vlachos
University of Cambridge
av308@cam.ac.uk

Christos Christodoulopoulos
Amazon
chrchrs@amazon.co.uk

Arpit Mittal
Amazon
mitarpit@amazon.co.uk

Abstract

The task of Natural Language Inference (NLI) is widely modeled as supervised sentence pair classification. While there has been a lot of work recently on generating explanations of the predictions of classifiers on a single piece of text, there have been no attempts to generate explanations of classifiers operating on pairs of sentences. In this paper, we show that it is possible to generate token-level explanations for NLI without the need for training data explicitly annotated for this purpose. We use a simple LSTM architecture and evaluate both LIME and Anchor explanations for this task. We compare these to a Multiple Instance Learning (MIL) method that uses thresholded attention make token-level predictions. The approach we present in this paper is a novel extension of zero-shot single-sentence tagging to sentence pairs for NLI. We conduct our experiments on the well-studied SNLI dataset that was recently augmented with manually annotation of the tokens that explain the entailment relation. We find that our white-box MIL-based method, while orders of magnitude faster, does not reach the same accuracy as the black-box methods.

1 Introduction

Large-scale datasets for Natural Language Inference (NLI) (Bowman et al., 2015; Williams et al., 2018) have enabled the development of many deep-learning models (Rocktäschel et al., 2016; Peters et al., 2018; Radford et al., 2018). The task is modeled as 3-way classification of the entailment relation between a pair of sentences. Model performance is assessed through accuracy on a held-out test set. While state-of-the-art models achieve high accuracy, their complexity makes it difficult to interpret their behavior.

Explaining the predictions made by classifiers has been of increasing concern (Doshi-Velez and Kim, 2017). It has been studied in natural language processing through both black-box analysis, and through modifications to the models under investigation; we refer to the latter approaches as *white-box*. Common black-box techniques generate explanations of predictions through training meta-models by perturbing input tokens (Ribeiro et al., 2016; Nguyen, 2018; Ribeiro et al., 2018) or through interpretation of model sensitivity to input tokens (Li et al., 2016; Feng et al., 2018). White-box methods induce new features (Aubakirova and Bansal, 2016), augment models to generate explanations accompanying their predictions (Lei et al., 2016; Camburu et al., 2018), or expose model internals such as magnitude of hidden states (Linzen et al., 2016), gradients (as a proxy for model sensitivity to input tokens (Li et al., 2016)) or attention (Bahdanau et al., 2014; Xu et al., 2015).

Premise:	Children smiling and waving at a camera
Hypothesis:	The kids are frowning
Label:	Contradiction

Figure 1: Example of token-level highlights from the e-SNLI dataset (Camburu et al., 2018). Annotators were provided a premise and hypothesis and asked to highlight words considered essential to explain the label.

Model explanations typically comprise a list of features (such as tokens) that contributed to the prediction and can serve two distinct purposes: acting either as a diagnostic during model development or to allow for a rationale to be generated for a system user. While methods for explaining predictions may output what was salient to the model, there is no guarantee these will correspond to the features that users deem important.

In this paper we introduce a white-box method that thresholds the attention matrix of a neural entailment model to induce token-level explanations.

Proceedings of NAACL-HLT 2019, pages 963–969
Minneapolis, Minnesota, June 2 - June 7, 2019.

To encourage the model's prediction of salient tokens to correspond better to the tokens users would find important, our approach uses Multiple Instance Learning (MIL) (Maron and Lozano-Pérez, 1998) to regularize the attention distributions.

We compare this against two black-box methods: LIME (Ribeiro et al., 2016) and Anchor Explanations (Ribeiro et al., 2018); both white- and black-box methods are applied to a simple neural architecture relying on independent sentence encoding with cross-sentence attention, and thus could also be applied to more complex architectures of the same family. Finally, we also compare against a fully supervised baseline trained to jointly predict entailment relation and token-level explanations. Our experiments are conducted on e-SNLI (Camburu et al., 2018), a recently introduced extension to SNLI (Bowman et al., 2015), containing human-selected highlights of which words are required to explain the entailment relation between two sentences (see Fig. 1).

Our experimental results indicate that regularizing the model's attention distributions encourages the explanations generated to be better aligned with human judgments (even without our model having explicit access to the labels which tokens annotators found important). Compared to the baseline thresholded attention mechanism, our method provides an absolute increase in token-level precision and recall by 6.68% and 28.05% respectively for the hypothesis sentence for e-SNLI explanations.

We also found that attention based explanations are not as reliable as black-box model explanation techniques, as indicated by higher F_1 scores for both LIME and Anchor Explanations. This is consistent with findings of contemporaneous work by Jain and Wallace (2019). However, we do show that, if generating explanations from a model is a requirement, incorporating an explicit objective in training can be beneficial. This can be particularly useful in practicw due to the computational cost of black-box model explanations, which in empirical evaluation we found to be orders of magnitude slower (0.01 seconds vs 64 seconds per instance).

2 NLI Model

The model we use for both white- and black-box experiments is based on an architecture widely adopted for sentence-pair classification (Lan and Xu, 2018). It comprises the following:

Word Embeddings We use pretrained GloVe embeddings (Pennington et al., 2014) that were fixed during training.

Sentence Encoding Both the premise and hypothesis are independently encoded with the same LSTM (Hochreiter and Schmidhuber, 1997), yielding $\mathbf{h}^p$ and $\mathbf{h}^h$ respectively.

Attention A matrix of soft alignments between tokens in the premise sentence and the hypothesis sentence is computed using attention (Bahdanau et al., 2014) over the encodings. Like Parikh et al. (2016), we project the encoded sentence representations using a feed-forward network, f_{attend}, ($u_i = f_{attend}(h_i^p)$, $v_j = f_{attend}(h_j^h)$) before computing the inner product: $\tilde{A}_{ij} = u_i^T v_j$. Given a premise of length m, the attention distribution for the hypothesis sentence is $\mathbf{a}^h = \text{normalize}(\tilde{A}_{m,*})$ where linear normalization is applied ($\text{normalize}(w) = \frac{w}{\|w\|_1}$). Likewise for the corresponding hypothesis of length n, the premise attention distribution is $\mathbf{a}^p = \text{normalize}(\tilde{A}_{*,n})$.

Output Classifier We predict the class label through a feed-forward neural network, f_{cls}, where both attended encodings of the premise and hypothesis final hidden states are concatenated as input: $f_{cls}([a_m^p h_m^p; a_n^h h_n^h])$. The logits are normalized using the softmax function, yielding a distribution over class labels $\hat{y}$.

Training The model is trained in a supervised environment using cross-entropy loss between the predicted class labels for an instance $\hat{y}$ and the labeled value in the dataset, formally defined in Section 3.

3 Generating Token-Level Explanations

Let $\mathbf{x}^p = (x_1^p, \ldots, x_m^p)$ and $\mathbf{x}^h = (x_1^h, \ldots, x_n^h)$ be sequences of tokens of length m and n respectively for the input premise and hypothesis sentences. Let y represent an entailment relation between $\mathbf{x}^p$ and $\mathbf{x}^h$ where $y \in \{\text{entails}, \text{contradicts}, \text{neutral}\}$. Labeled training data is provided of the form $\{(\mathbf{x}_k^p, \mathbf{x}_k^h, y_k)\}_{k=1}^K$. For each instance, the model must generate an explanation $\mathbf{e}$ defined as a subset of zero or more tokens from both the premise and hypothesis sentences: $\mathbf{e}^p \in \mathcal{P}(\mathbf{x}^p)$, $\mathbf{e}^h \in \mathcal{P}(\mathbf{x}^h)$.

We generate token-level explanations by thresholding token attention weights. Concretely, we select all tokens, x, with a weight greater than a threshold. While similar to Rei and Søgaard (2018), we incorporate a re-scaling using the tanh function: $\mathbf{e}^p = \{x_i^p | \tilde{a}_i^p \in \tilde{A}_{*,n} \wedge \tanh(\tilde{a}_i^p) \geq \tau\}$ and likewise for the hypothesis.

3.1 Multiple Instance Learning

Thresholding the attention distributions from our model will give an indication of which tokens the model is weighting strongly for the entailment task. However, as mentioned in the introduction, there is no guarantee that this method of explaining model behavior will correspond with tokens that humans judge as a reasonable explanation of entailment. To better align the attention-based explanations with the human judgments, we model the generation of explanations as Multiple Instance Learning (MIL) (Maron and Lozano-Pérez, 1998). In training the model sees labeled "bags" (sentences) of unlabeled features (tokens) and learns to predict labels both for the bags and the features. In MIL, this is often achieved by introducing regularizers when training. To encourage our NLI model to predict using sparser attention distributions (which we expect to correspond more closely with human token-level explanations), we introduce the following regularizers into the loss function:

R_1: This entropy-based term allows us to penalize a model that uniformly distributes probability mass between tokens.

$$\begin{aligned} R_1 &= \sum_{k=1}^{K} \left(\mathbb{H}(\mathbf{a}_k^p) + \mathbb{H}(\mathbf{a}_k^h)\right) \\ &= -\sum_{k=1}^{K} \left(\sum_{i=1}^{m} a_{k,i}^p \log a_{k,i}^p + \sum_{j=1}^{n} a_{k,j}^h \log a_{k,j}^h\right) \end{aligned} \tag{1}$$

R_2: This term, adapted from a loss function for zero-shot tagging on single sentences (Rei and Søgaard, 2018), penalizes the model for breaking the assumption that at least one token must be selected from both premise and hypothesis sentences to form an explanation. The only exception is that, following the e-SNLI dataset annotation by Camburu et al. (2018), if the neutral entailment is predicted, no tokens are selected from the premise.

$$\begin{aligned} R_2 = \sum_{k=1}^{K} \Big(&(\max_i a_{k,i}^p - \mathbb{I}[k_c \neq \text{neutral}])^2 \\ &+ (\max_j a_{k,j}^h - 1)^2 \Big) \end{aligned} \tag{2}$$

R_3: This term, also adapted from Rei and Søgaard (2018), encodes the assumption that not all tokens must be selected in the explanation. This is achieved by penalizing the smallest non-zero attention weight, which has the effect of encouraging at least one weight to be close to zero.

$$R_3 = \sum_{k=1}^{K} \left((\min_i a_{k,i}^p)^2 + (\min_j a_{k,j}^h)^2\right) \tag{3}$$

The loss function used for training of our proposed model incorporating the regularizers which are controlled with hyperparameters is:

$$L = \sum_{k=1}^{K} \sum_{c \in C} y_{k,c} \log \hat{y}_{k,c} + \alpha R_1 + \beta R_2 + \gamma R_3 \tag{4}$$

4 Alternative Models

4.1 Black-box explanations of NLP models

We use two established black-box model explanation techniques for generating token-level explanations: LIME (Ribeiro et al., 2016) and Anchors (Ribeiro et al., 2018). Both techniques probe a classifier by making perturbations to a single input and modeling which of these perturbations influence the classification. To adapt these for use in NLI, we make a simple modification that runs the analysis twice: once for the premise sentence and once for the hypothesis sentence on the NLI model described in Section 2.

LIME Generates local explanations for a classifier through the introduction of a simple meta-model that is trained to replicate a local decision boundary of an instance under test. The training data is generated through observing the impact on classification when removing tokens from the input string.

Anchor Explanations Considers the distribution of perturbed instances in the neighborhood of the instance under test through word substitution to identify a rule (a set of tokens in our case) for which the classification remains unchanged.

4.2 Supervised Model

For a supervised model we build upon the model discussed in Section 2, adding components to support LSTM-CRF-based tagging (Lample et al., 2016). We use the following architecture:

Model	Runtime (s) per instance	Token Explanation (%)					
		Premise			Hypothesis		
		P	R	F1	P	R	F1
Fully Supervised LSTM-CRF	0.02	86.91	40.98	55.70	81.16	54.79	65.41
Thresholded Attention (Linear)	**0.01**	19.96	19.67	19.56	46.70	34.92	39.89
+ MIL Regularizers (R1)	-	16.59	15.67	16.12	50.02	42.44	46.01
+ MIL Regularizers (R2 + R3)	-	18.19	20.18	19.13	51.29	50.73	51.00
+ MIL Regularizers (R1 + R2 + R3)	-	19.23	26.21	22.18	53.38	62.97	57.78
LIME	64	**60.56**	**48.28**	**53.72**	**57.04**	**66.92**	**61.58**
Anchors	10	42.06	20.04	27.14	53.12	63.94	58.03

Table 1: Token-level scores for human-selected explanations of NLI using the e-SNLI dataset. The select-all baseline precision for the premise is 18.5% and 35.2% for the hypothesis.

Context Encoding We use the same pretrained GloVe embeddings (Pennington et al., 2014) that were fixed during training. The premise and hypothesis sentence were independently encoded with the same LSTM (Hochreiter and Schmidhuber, 1997) yielding $\mathbf{h}^p$ and $\mathbf{h}^h$ respectively and attended to as per the description in Section 2.

Outputs The model is jointly trained with two output objectives: a labeling objective and a tagging objective. During training, the losses for both tasks are equally weighted. The first output objective is the three-way SNLI classification over the pair of sentences. This is the same component as the model presented in Section 2.

The second objective is a binary tagging objective over the highlighted token-level explanations. We use a jointly-trained LSTM-CRF decoder architecture (Lample et al., 2016) which operates a CRF over encoded representations for each token. In our model, we independently decode the premise and hypothesis sentences. The inputs to our CRF are the attended premise and hypothesis: $\mathbf{a}^p \odot \mathbf{h}^p$ and $\mathbf{a}^h \odot \mathbf{h}^h$ respectively (where $\odot$ is the point-wise multiplication between the attention vector and the encoded tokens).

5 Experiments

We evaluate the generated explanations through evaluation of token-level F_1 scores comparing them against tokens selected by humans to explain the entailment relation using the e-SNLI dataset (Camburu et al., 2018). The development split of the e-SNLI dataset is used for hyperparameter selection and we report results on the test split. Where multiple annotations are available for a sentence pair, the union of the annotations is taken. We also report average runtime per sentence in seconds measured using 1 thread on an AWS c4.xlarge instance.

Implementation Details The model is implemented in AllenNLP (Gardner et al., 2018) and we optimized our model with Adagrad (Duchi et al., 2011), selecting the models which attained high hypothesis F_1 without greatly affecting the accuracy of entailment task (approx 81% for the thresholded attention model). The cell state and hidden dimension was 200 for the LSTM sentence encoder. The projection for attention, f_{attend}, was a single layer 200 dimension feed forward network with ReLU activation. The final feed forward classifier, f_{cls}, dimension was $(200, 200, 3)$ and ReLU activation over the first 2 layers. For the comparison against black-box explanation mechanisms, we use the code made public by the authors of the respective works setting any hyperparameters to the default values or those suggested in the papers.

Results Our experimental results (Table 1) indicate that the LIME black-box explanation technique over the model described in Section 2 provides token-level explanations that are more similar to human judgments than thresholding the attention distributions. We show that the addition of MIL regularizers for generating explanations using thresholded attention improved precision and recall hypothesis explanations. However, similar improvements were not realized for the premise sentence. While the black-box methods generated better explanations than thresholded attention, they were 3 orders of magnitude slower.

Only LIME was able to generate good token-level explanations for the premise. This is in contrast to the attention-based explanations of the premise (in the model that LIME was run on) which could not generate satisfactory explanations (see row 2 of Table 1). This supports findings in recent works (Jain and Wallace, 2019) that indicate that attention does not always correspond to other measures of feature importance. We also found that the black-box model explanation methods behave differently given the same model under test: the premise explanation generated by the Anchors method was more in line with what the model attended to, reflected by the lower recall.

The fully supervised model had high precision yet (relatively) low recall. We observed it has a bias towards predicting common words that often appear in highlights (e.g. 'man', 'woman', 'dog', 'people') for both premise and hypothesis sentences rather than highlighting keywords that would form an instance-specific explanation. This behaviour is also more pronounced in the premise sentence highlights rather than the hypothesis. We reason that this due to how the SNLI dataset was constructed: a premise sentence was used to generate 3 hypothesis sentences (entailed, contradicted and neutral). This is corroborated by a survey of 250 instances from the SNLI dataset, where we found that all or part of the subject noun phrase remained unchanged between the premise and hypothesis sentences 60% of the time. While the supervised model correctly captured commonly occurring text patterns, as demonstrated by the high F_1 scores, this behaviour alone was not sufficient to identify tokens that correlated with the entailment label. We found that most of the commonly predicted tokens by our supervised model did not appear in lists of features highly correlated with the entailment label (Poliak et al., 2018; Gururangan et al., 2018).

6 Conclusions

In this paper we explored how to generate token-level explanations from NLI models. We compared the LIME and Anchors black-box methods against a novel, white-box Multiple Instance Learning (MIL) method and a fully supervised baseline. The explanations generated by LIME were more similar to the human judgments of the tokens that justify an entailment relation than the attention thresholding approach. This corroborates contemporaneous work (Jain and Wallace, 2019) indicating a lack of correspondence between attention and other measures of feature importance.

The MIL method we introduced steered the attention distributions over tokens in our model to correspond closer to the human judgments allowing better explanations to be generated. Even though, when considering the token-level F_1 score, the attention-based explanations were not as good as the black-box techniques we evaluated, they were orders of magnitude faster.

The attention thresholding model we tested did not generate satisfactory explanations had low F_1 for the premise sentences. A possible explanation for the poor performance is what is found by Rei and Søgaard (2018) who show that MIL regularizers performed better when there is a higher degree of association between the sentence-level label and the token-level labels. Our model used *independent* encodings of the premise and hypothesis but in NLI there is a strong dependence between the two sentences; thus the entailment prediction should be explained through pairwise token comparisons (e.g. synonyms, upward entailment, etc.). In future work we plan to address this by adding explicit cross-sentence semantic knowledge (Joshi et al., 2018).

Acknowledgements

This work was conducted while James Thorne was an intern at Amazon. Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139).

References

Malika Aubakirova and Mohit Bansal. 2016. Interpreting Neural Networks to Improve Politeness Comprehension. pages 2035–2041.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference.

Oana-maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-SNLI : Natural Language Inference with Natural Language Explanations. In *32nd Conference on Neural Information Processing Systems (NIPS)*, pages 1–13.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Shi Feng, Eric Wallace, Alvin Grissom, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of Neural Models Make Interpretations Difficult.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation Artifacts in Natural Language Inference Data.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2018. pair2vec: Compositional word-pair embeddings for cross-sentence inference. *arXiv preprint arXiv:1810.08854*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. *arXiv preprint arXiv:1612.08220*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Oded Maron and Tomás Lozano-Pérez. 1998. A framework for multiple-instance learning. In *Advances in neural information processing systems*, pages 570–576.

Dong Nguyen. 2018. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. *Naacl*, pages 1069–1078.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, New Orleans, Louisiana.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. (1):180–191.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. *arXiv*, pages 1–12.

Marek Rei and Anders Søgaard. 2018. Zero-shot Sequence Labeling: Transferring Knowledge from Sentences to Tokens. pages 293–302.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. 39(2011):117831.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *Proc. of 32nd Conference on Artificial Intelligence (AAAI)*, pages 1527–1535.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. *Iclr*, (2015):1–9.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org.

Strong Baselines for Complex Word Identification across Multiple Languages

Pierre Finnimore[1], **Elisabeth Fritzsch**[1], **Daniel King**[1], **Alison Sneyd**[1],
Aneeq Ur Rehman[1], **Fernando Alva-Manchego**[1] and **Andreas Vlachos**[2]
[1]Department of Computer Science, University of Sheffield
[2]Department of Computer Science and Technology, University of Cambridge
`{pmfinnimore,fritzsch.elisabeth,danielking1903}@gmail.com,`
`a.sneyd@shef.ac.uk,a.neeq8394@gmail.com,f.alva@shef.ac.uk,`
`andreas.vlachos@cst.cam.ac.uk`

Abstract

Complex Word Identification (CWI) is the task of identifying which words or phrases in a sentence are difficult to understand by a target audience. The latest CWI Shared Task released data for two settings: monolingual (i.e. train and test in the same language) and cross-lingual (i.e. test in a language not seen during training). The best monolingual models relied on language-dependent features, which do not generalise in the cross-lingual setting, while the best cross-lingual model used neural networks with multi-task learning. In this paper, we present monolingual and cross-lingual CWI models that perform as well as (or better than) most models submitted to the latest CWI Shared Task. We show that carefully selected features and simple learning models can achieve state-of-the-art performance, and result in strong baselines for future development in this area. Finally, we discuss how inconsistencies in the annotation of the data can explain some of the results obtained.

1 Introduction

Complex Word Identification (CWI) consists of deciding which words (or phrases) in a text could be difficult to understand by a specific type of reader. In this work, we follow the CWI Shared Tasks (Paetzold and Specia, 2016; Yimam et al., 2018) and assume that a target word or multi-word expression (MWE[1]) in a sentence is given, and our goal is to determine if it is complex or not (an example is shown in Table 1). Under this setting, CWI is normally treated using supervised learning and feature engineering to build monolingual models (Paetzold and Specia, 2016; Yimam et al., 2018). Unfortunately, this approach is infeasible for languages with scarce resources of annotated data. In this paper, we are interested in both monolingual and cross-lingual CWI; in the latter, we build models to make predictions for languages not seen during training.

Sentence	Target word/MWE	Complex?
Both China and the Philippines flexed their muscles on Wednesday.	flexed	Yes
	flexed their muscles	Yes
	muscles	No

Table 1: An annotated sentence in the English dataset of the Second CWI Shared Task.

While monolingual CWI has been studied extensively (see a survey in Paetzold and Specia (2017)), the cross-lingual setup of the task was introduced only recently by Yimam et al. (2017b), who collected human annotations from native and non-native speakers of Spanish and German, and integrated them with similar data previously produced for three English domains (Yimam et al., 2017a): News, WikiNews and Wikipedia.

For the Second CWI Shared Task (Yimam et al., 2018), participants built monolingual models using the datasets previously described, and also tested their cross-lingual capabilities on newly collected French data. In the monolingual track, the best systems for English (Gooding and Kochmar, 2018) differed significantly in terms of feature set size and the model's complexity, to the best systems for German and Spanish (Kajiwara and Komachi, 2018). The latter used Random Forests with eight features, whilst the former used AdaBoost with 5000 estimators or ensemble voting combining AdaBoost and Random Forest classifiers, with about 20 features.

In the cross-lingual track, only two teams achieved better scores than the baseline: Kajiwara and Komachi (2018) who used length and frequency based features with Random Forests, and

---

[1] We consider n-grams with $n \geq 2$ as MWEs, while Yimam et al. (2018) used $n \geq 3$.

Proceedings of NAACL-HLT 2019, pages 970–977
Minneapolis, Minnesota, June 2 - June 7, 2019.

Bingel and Bjerva (2018) who implemented an ensemble of Random Forests and feed-forward neural networks in a multi-task learning architecture.

Our approach to CWI differs from previous work in that we begin by building competitive monolingual models, but using the same set of features and learning algorithm across languages. This reduces the possibility of getting high scores due to modelling annotation artifacts present in the dataset of one language. Our monolingual models achieve better scores for Spanish and German than the best systems in the Second CWI Shared Task. After that, we focus on language-independent features, and keep those that achieve good performance in cross-lingual experiments across all possible combinations of languages. This results in a small set of five language-independent features, which achieve a score as high as the top models in the French test set. Finally, we analyse the annotation of the datasets and find some inconsistencies that could explain some of our results.

Code for all our models can be found at: `https://github.com/sheffieldnlp/cwi`

2 Problem Formulation

We tackle the binary classification task in the Second CWI Shared Task (Yimam et al., 2018), in which a model decides if a target word/MWE in a sentence is complex or not. Following common practice, we extract features from the target word/MWE and its context, and then use a supervised learning algorithm to train a classifier. For training and testing our models, we use the annotated datasets provided for the Second CWI Shared Task (see Table 2 for some statistics).

Dataset	Train	Dev	Test
English (EN) - News	14,002	1,764	2,095
English (EN) - WikiNews	7,746	870	1,287
English (EN) - Wikipedia	5,551	694	870
Spanish (ES)	13,750	1,622	2,232
German (DE)	6,151	795	959
French (FR)	N/A	N/A	2,251

Table 2: Number of annotated samples in each dataset for each language.

3 Monolingual Models

3.1 Features Description

Our feature set consists of 25 features that can be extracted for all languages considered (English, German, Spanish and French). They can be divided into three broad categories: features based on the target word/MWE, sub-word level features, and sentence-level features to capture information from the target's context. As we intended that our features be applicable across languages, we drew on features found to be useful in previous work on CWI (Yimam et al., 2017b, 2018). We made use of the python libraries spaCy[2] (Honnibal and Montani, 2017) and NLTK[3] (Loper and Bird, 2002). Details on the resources used for extracting each feature can be found in Appendix A.

At the **target word/MWE level**, we experimented with features such as Named Entity (NE) type, part-of-speech, hypernym counts, number of tokens in the target, language-normalised number of characters in each word, and simple unigram probabilities. These features are linguistically motivated. The perceived complexity of a MWE may be higher than that of a single word, as each component word can be complex, or simple component words can be synthesised into a complex whole. Similarly, infrequent words are less familiar, so we would expect low-probability target words to be found more complex. Along these lines, proper nouns could be more complex, as there is a vast number of NEs, and the chance that a person has encountered any one of them is low. We would expect this trend to reverse for the NE type of organisations, in combination with the Enlgish-News dataset, as organisations mentioned in news articles are frequently global, and so the chance that a person has encountered a proper noun *that is an organisation* is often higher than for proper nouns in general. In total, 14 features were used at the target word/MWE level.

Our **sub-word level** features include prefixes, suffixes, the number of syllables, and the number of complex punctuation marks (i.e. punctuation within the target word/MWE, such as hyphens, that could denote added complexity). We would expect certain affixes to be useful features, as language users use sub-word particles like these to identify unknown words: by breaking up a word like "granted" into "grant-" and "-ed", readers can fall back on their knowledge of these component pieces to clarify the whole. A total of 9 sub-word features were used in the monolingual models.

Finally, **sentence level** features with linguistic

---

[2] `https://spacy.io/`

[3] `https://www.nltk.org/`

motivations were also considered. Long sentences could be harder to understand, which makes it more difficult to figure out the meaning of unknown words contained within them. Also, long sentences are more likely to include more unknown words or ambiguous references. Therefore, we considered sentence length (i.e., number of tokens in the sentence) as a feature. In addition, we extracted N-grams (unigrams, bigrams and trigrams) from the whole sentence, since certain sentence constructions can help a reader understand the target word/MWE. For example, "A of the B" suggests a relation between A and B. We used 2 sentence-level features in total.

3.2 Experiments and Results

Following Yimam et al. (2018), we used Macro-F1 score to evaluate performance and for comparison with previous work on the datasets. We used Logistic Regression for all our experiments, as it allowed for easy exploration of feature combinations, and in initial experiments we found that it performed better than Random Forests. We evaluated both using the full feature set described before, as well as a two-feature baseline using the number of tokens of the target and its language-normalised number of characters. Results of our monolingual experiments are shown in Table 3.

Dataset	Dev		Test		
	BL	MA	BL	MA	SotA
EN - News	83.6	85.5	69.7	86.0	**87.4**
EN - WikiNews	80.4	82.8	65.8	81.6	**84.0**
EN - Wikipedia	74.2	76.6	70.1	76.1	**81.2**
ES	78.0	77.1	69.6	**77.6**	77.0
DE	79.5	74.6	72.4	74.8	**75.5**
Mean	79.1	79.3	69.5	79.2	N/A

Table 3: Macro-F1 for the baseline (BL), our monolingual approach (MA), and the state of the art (SotA) on the Dev and Test splits of each dataset.

In the test set, our baseline results (BL in Table 3) are strong, especially in German. Our full 25-features model improves on the baseline in all cases, with the biggest increase of over 16 percentage points seen for the EN-News dataset. Our system beats the best performing system from the Shared Task in Spanish (77.0) and German (74.5), both obtained by Kajiwara and Komachi (2018). However, the state of the art for German remains the Shared Task baseline (75.5) (Yimam et al., 2018). The best results for all three English datasets were obtained by Gooding and Kochmar (2018); ours is within two percentage points of their News dataset score. Furthermore, the mean score for our system (79.2) is close to the mean of the best performing models (81.0), which are different systems, while using simpler features and learning algorithm. The best-performing model in English, for example, used Adaboost with 5000 estimators (Gooding and Kochmar, 2018).

4 Cross-lingual Models

4.1 Features Description

Linguistically, the cross-lingual approach can be motivated by the relation between certain languages (such as French and Spanish both being Romance languages). In addition, there may be features identifying complex words that are shared even across language families.

To be able to test a model on a language that was unseen during training, the features the model works with must be cross-lingual (or language-independent) themselves. For example, the words themselves are unlikely to transfer across languages (apart from those that happen to be spelled identically), but the popularity of the words would transfer. This rules out some of the features we used for the monolingual approach (see Sec. 3.1), as they were language-dependent. One such feature is N-grams for the target word/MWE, which depend on the language, and so will only occur with extreme sparsity outside of their source language. For example, if applying a system trained on English to unseen French, the English phrases "à la mode" or "film noir" might reoccur in the French, since they originate from that language, but these are rare exceptions. What is more, a French loan-phrase may have different complexity characteristics to the same N-grams occurring in their native language. Therefore, we did not use these features in the cross-lingual system.

4.2 Experiments and Results

To find out which features were best suited for the cross-lingual approach, we performed an iterative ablation analysis (see Appendix B for details). Using this process, we arrived at our final cross-lingual feature set: number of syllables in the target, number of tokens in the target, number of complex punctuation marks (such as hyphens), sentence length, and unigram probabilities.

Furthermore, we analyse the effect of different language combinations on the performance of the cross-lingual model in order to investigate how the relationship between the languages trained and tested on would influence model performance. Recall that we only have training data for English, Spanish and German, but not French. We train models using all possible combinations (each language independently, each pairing, and all three) and evaluate on each of the four languages that have test data (i.e. the former three and French), excluding training combinations that include the test language. Results are shown in Table 4.

EN	ES	DE	Eval	Source	Test	Dev
	✓	✓	EN	WikiNews	61.8	63.7
		✓	EN	WikiNews	62.3	63.6
	✓		EN	WikiNews	61.6	63.8
	✓	✓	EN	Wikipedia	62.8	64.4
		✓	EN	Wikipedia	62.6	64.4
	✓		EN	Wikipedia	63.1	65.2
	✓	✓	EN	News	67.1	65.6
		✓	EN	News	67.0	65.6
	✓		EN	News	**67.2**	**65.9**
✓		✓	ES	N/A	70.8	71.3
		✓	ES	N/A	**72.6**	**74.1**
✓			ES	N/A	69.1	70.0
✓	✓		DE	N/A	**73.4**	**78.3**
	✓		DE	N/A	72.6	77.4
✓			DE	N/A	73.0	76.0
✓	✓	✓	FR	N/A	73.1	N/A
	✓	✓	FR	N/A	75.7	N/A
✓	✓		FR	N/A	73.4	N/A
✓		✓	FR	N/A	70.5	N/A
		✓	FR	N/A	**75.8**	N/A
	✓		FR	N/A	73.4	N/A
✓			FR	N/A	69.2	N/A

Table 4: Comparison of Test and Dev results for all permutations of training languages.

When testing on French, we achieved the highest performance by training on German only (75.8), followed closely by training on a combination of German and Spanish (75.7) and only Spanish (75.5). The worst performance was achieved by training only on English (69.2), and the performance also noticeably decreased for all training combinations that included English.

When testing on German, language choice had a weaker effect. The highest score came from combining English and Spanish (73.4), but using only one of those languages gave comparable results (72.6 for Spanish, 73.0 for English).

For Spanish, the best results were achieved when training only on German (72.6). Adding English to the training languages decreased the performance (70.8), which was even lower when training only on English (69.1).

	Spanish	German	French
Monolingual SotA	**77.0**	**75.5**	N/A
Cross-lingual SotA	N/A	N/A	**76.0**
Our cross-lingual	72.6	73.4	75.8

Table 5: Comparison between the monolingual and cross-lingual state of the art (SotA), and our cross-lingual system.

It is noteworthy that adding English to the training languages noticeably decreases performance for both Spanish and French, but not for German. One possible reason for Spanish and French not benefiting from English when German does is that both English and German are Germanic languages, whereas Spanish and French are Romance languages. Another possible explanation for the decrease of performance caused by training with English is that there are inconsistencies in the way MWEs in the datasets were labelled across languages, which we explore in Sec. 5.

We finally compare our cross-lingual models against the state of the art: the best monolingual system for Spanish and German, and the best cross-lingual system for French, where no monolingual systems exist. As Table 5 shows, our cross-lingual models come close to the best monolingual models for Spanish and especially for German. This is remarkable given how simple our model and features are, and that the approaches we compare against train complex models for each language. Furthermore, this points towards the possibility of extending CWI to more languages which lack training data.

Finally, Table 6 compares the coefficients for models trained on Romance and Germanic languages. Notably, use of complex punctuation (such as the hyphenation in "laser-activated" or "drug-related") and the number of tokens are inversely correlated w.r.t. the word or MWE being complex. More words in the target was correlated with complexity for English and German, and inversely correlated for Spanish.

5 Dataset Analysis

While examining our models' incorrect predictions, we observed inconsistencies in labelling in the datasets between target MWEs and their sub-words/sub-expressions (SWs).

Feature	Train	Coefficient
number of complex punctuation marks	EN	-0.693
	DE	-0.559
	ES	1.111
number of tokens	EN	-2.200
	DE	-0.534
	ES	1.420

Table 6: Coefficients for cross-lingual models trained on Germanic and Romance languages.

The First CWI Shared Task (Paetzold and Specia, 2016) used the annotations of a group (i.e. ten annotators on the training data) to predict the annotation of an individual (i.e. one annotator on the test data). The resulting inconsistencies in labelling may have contributed to the low F-scores of systems in the task (Zampieri et al., 2017). Although the Second CWI Shared Task improved on the first by having multiple annotators for all splits of the data, it contains some labelling inconsistencies arising from annotators now being able to label phrases, and words within them, separately.

More concretely, we found that across all datasets, 72% of MWEs contain at least one SW with the opposite label (see Table 7). While this makes sense in some cases, every SW in 25% of MWE instances has the opposite label. For example, “numerous falsifications and ballot stuffing” is not annotated as complex, despite its SWs “numerous”, “numerous falsifications”, “falsifications”, “ballot”, “ballot stuffing” and “stuffing” all being complex. Conversely, “crise des marchés du crédit” is complex, despite “crise”, “marchés” and “crédit” being labelled non-complex. It is difficult to see how classifiers that extract features for MWEs from their individual SWs could predict the labels of both correctly.

Furthermore, every target MWE in the Spanish, German and French datasets is labelled complex. This may bias a classifier trained on the Spanish or German datasets towards learning MWEs and long individual words (if length is a feature) are complex. In particular, this observation may help explain why adding English as a training language decreased the performance of our cross-lingual system when testing on French and Spanish (where all MWEs are complex). An analysis in Bingel and Bjerva (2018) further found that their cross-lingual French model was effective at predicting long complex words/MWEs but had difficulty predicting long non-complex words.

	C	NC	$\geq$ 1 Irreg.	All Irreg.
English	3,750	982	3,315	950
Spanish	2,309	0	1,747	760
German	502	0	374	178
French	242	0	192	82
Total	6,803	982	5,628	1,970

Table 7: MWE annotation analysis: numbers of MWEs labelled complex (C) and non-complex (NC), numbers with at least one SW ($\geq$ 1 Irreg) and all SWs (All Irreg.) having the opposite label.

It is also worth noting that considering a word or MWE as complex is subjective and may differ from person to person, even within the same target audience. Bingel et al. (2018) investigated predicting complex words based on the gaze patterns of children with reading difficulties. They found a high degree of specificity in misreadings between children, that is, which words they found complex when reading aloud. This variety of complexity judgements even within one target group points towards the high degree of subjectivity in the task, which may also partly explain the inconsistencies in the dataset.

6 Conclusion and Future Work

The monolingual and cross-lingual models presented achieve comparable results against more complex, language-specific state-of-the-art models, and thus can serve as strong baselines for future research in CWI. In addition, our analysis of the dataset could help in the design of better guidelines when crowdsourcing annotations for the task. Dataset creators may wish to only allow single words to be chosen as complex to avoid labelling inconsistencies. In case MWEs are being permitted, we suggest instructing annotators to chose the smallest part of a phrase they find complex (French annotators for the Second CWI Shared Task sometimes grouped individual complex words into a complex MWE (Yimam et al., 2018)).

Acknowledgements

This work was initiated in a class project for the NLP module at the University of Sheffield. The authors would like to acknowledge the contributions of Thomas Dakin, Sanjana Khot and Harry Wells who contributed their project code to this work. Andreas Vlachos is supported by the EPSRC grant eNeMILP (EP/R021643/1).

References

Joachim Bingel, Maria Barrett, and Sigrid Klerke. 2018. Predicting misreadings from gaze in children with reading difficulties. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 24–34, New Orleans, Louisiana. Association for Computational Linguistics.

Joachim Bingel and Johannes Bjerva. 2018. Cross-lingual complex word identification with multitask learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 166–174. Association for Computational Linguistics.

Sian Gooding and Ekaterina Kochmar. 2018. Camb at cwi shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Tomoyuki Kajiwara and Mamoru Komachi. 2018. Complex word identification based on frequency in a learner corpus. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 195–199. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569. Association for Computational Linguistics.

Gustavo H. Paetzold and Lucia Specia. 2017. A survey on lexical simplification. *Journal of Artificial Intelligence Research*, 60:549–593.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78. Association for Computational Linguistics.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017a. Cwig3g2 - complex word identification task across three text genres and two user groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407. Asian Federation of Natural Language Processing.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017b. Multilingual and cross-lingual complex word identification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 813–822. INCOMA Ltd.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63. Asian Federation of Natural Language Processing.

A Detailed Feature Set

Level	Name	Description	Resource
Target word/MWE	NER_tag_counts	Counts of each Named Entity tag in target	spaCy
	pos_tag_counts	Counts of each part-of-speech tag in target	spaCy
	hypernym_count	Number of hypernyms	WordNet (NLTK)
	len_tokens	Absolute length in tokens	N/A
	len_tokens_norm	Normalised length in tokens	N/A
	len_chars_norm	Normalised length in characters	N/A
	unigram_prob	Log of the product of unigram probabilities	EN: Brown Corpus (NLTK) ES: CESS-ESP (NLTK) DE: TIGER Corpus[4] FR: Europal[5]
	bag_of_shapes	Bag of morphological shapes	spaCy
	rare_word_count	Count of rare words in target	EN: subset of Google's Trillion Word Corpus[6] DE: list of the most common 3,000 words[7] ES: word frequency list by M. Buchmeier[8]
	rare_trigram_count	Count of rare trigrams in target	Same as rare_word_count
	is_stop	Frequency of stopwords in target	NLTK, Ranks NL[9]
	is_nounphrase	If target is a noun phrase	spaCy
	avg_chars_per_word	Avg. word length (in characters) of the target	N/A
	iob_tags	Count of BIO tags in target	spaCy
Sub-word	lemma_feats	Bag of lemmas for target sentence	spaCy
	len_sylls	Length of target in syllables	Pyphen[10]
	num_complex_punct	Count of complex punctuation in target	N/A
	char_n_gram_feats	Character N-Grams, incl. prefixes and suffixes	N/A
	char_tri_sum	Sum of character trigrams' corpus frequencies	EN: Brown Corpus (NLTK) ES: CESS-ESP (NLTK) DE: TIGER Corpus
	char_tri_avg	Average of character trigrams' corpus frequencies	same as char_tri_sum
	consonant_freq	Count of consonants in target	N/A
	gr_or_lat	If target has Greek or Latin affixes	List of Greek and Latin roots in English[11]
	is_capitalised	If target's first letter is uppercased	N/A
Sentence	sent_length	Number of tokens in the sentence	N/A
	sent_n_gram_feats	Unigrams, bigrams and trigrams in the sentence	N/A

Table 8: Monolingual and Cross-lingual Feature Set Summary

---

[4] https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html

[5] http://www.statmt.org/europarl/

[6] https://github.com/first20hours/google-10000-english

[7] http://germanvocab.com/

[8] https://en.wiktionary.org/wiki/User:Matthias_Buchmeier/Spanish_frequency_list-1-5000

[9] https://www.ranks.nl/stopwords

[10] https://pyphen.org/

[11] https://www.oakton.edu/user/3/gherrera/Greek%20and%20Latin%20Roots%20in%20English/greek_and_latin_roots.pdf

B Cross-lingual Features Ablation

Iteration	Current features	Features increasing performance	Features decreasing performance
1	len_tokens	num_complex_punct len_sylls sent_length	is_nounphrase len_tokens_norm consonant_freq is_capitalised bag_of_shapes pos_tag_count
2	len_tokens len_sylls num_complex_punct sent_length	unigram_prob	gr_or_lat
3	len_tokens len_sylls num_complex_punct sent_length unigram_prob		char_ngram_feats iob_tags lemma_feats NER_tag_counts

Table 9: Ablation analysis for the cross-lingual features

Adaptive Convolution for Multi-Relational Learning

Xiaotian Jiang[1], **Quan Wang**[2], **Bin Wang**[3]
[1]Cloud and Smart Industries Group, Tencent
[2]Baidu Inc.
[3]Xiaomi AI Lab
`johnxtjiang@tencent.com`, `quanwang1012@gmail.com`,
`wangbin11@xiaomi.com`

Abstract

We consider the problem of learning distributed representations for entities and relations of multi-relational data so as to predict missing links therein. Convolutional neural networks have recently shown their superiority for this problem, bringing increased model expressiveness while remaining parameter efficient. Despite the success, previous convolution designs fail to model full interactions between input entities and relations, which potentially limits the performance of link prediction. In this work we introduce ConvR, an adaptive convolutional network designed to maximize entity-relation interactions in a convolutional fashion. ConvR adaptively constructs convolution filters from relation representations, and applies these filters across entity representations to generate convolutional features. As such, ConvR enables rich interactions between entity and relation representations at diverse regions, and all the convolutional features generated will be able to capture such interactions. We evaluate ConvR on multiple benchmark datasets. Experimental results show that: (1) ConvR performs substantially better than competitive baselines in almost all the metrics and on all the datasets; (2) Compared with state-of-the-art convolutional models, ConvR is not only more effective but also more efficient. It offers a 7% increase in MRR and a 6% increase in Hits@10, while saving 12% in parameter storage.

1 Introduction

Multi-relational data refers to directed graphs whose nodes correspond to entities and edges different types of relations between entities. An edge of the form (*subject*, *relation*, *object*) indicates that there exists a specific relation between the subject and object entities. Learning with multi-relational data plays a pivotal role in many application domains, ranging from social networks or recommender systems to large-scale knowledge bases (KBs) (Bordes et al., 2013; Jenatton et al., 2012). This work focuses on modeling multi-relational data from KBs, with the aim of predicting missing facts on KBs, a challenging task known as link prediction in statistical relational learning (SRL) (Getoor and Taskar, 2007).

Various SRL techniques (Nickel et al., 2016a) have been proposed for this task, among which vector space embedding models (Wang et al., 2017) are gaining increasing attention due to their superior performance and potential scalability. The key idea there is to learn and operate on latent features (embeddings) of entities and relations, so as to uncover non-trivial connectivity patterns in multi-relational data. Previous works of this kind tend to adopt shallow, simple models to extract latent features, e.g., the translation based models (Bordes et al., 2013; Wang et al., 2014) or the bilinear models and their variants (Jenatton et al., 2012; Yang et al., 2015; Trouillon et al., 2016). Using these simple models allows one to easily handle large-scale KBs, but usually at the cost of learning less expressive features. In fact, such simple models typically generate a single feature with each entry of the embeddings. The only way to increase the number of features (and thus their expressiveness) is to increase the embedding size (Dettmers et al., 2018). This potentially limits the performance of link prediction with a given number of parameters. To increase model expressiveness, there emerge some deeper, more complicated designs, in particular those on the basis of neural network architectures (Socher et al., 2013; Bordes et al., 2014; Dong et al., 2014; Schlichtkrull et al., 2017a). Such approaches, however, often have more parameters and are prone to overfit, at least on the (relatively small) benchmark datasets used by the scientific community (Nickel et al., 2016a).

Recently, Dettmers et al. (2018) devised ConvE,

Proceedings of NAACL-HLT 2019, pages 978–987
Minneapolis, Minnesota, June 2 - June 7, 2019.

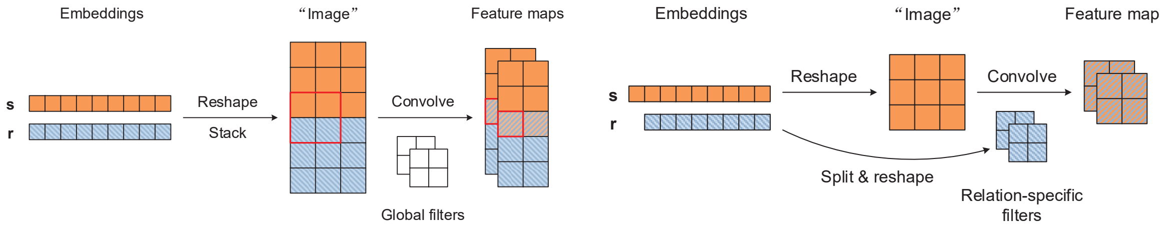
a multi-layer convolutional network which enables expressive feature learning while remaining highly parameter efficient. Given a subject-relation-object triple (s, r, o), ConvE first reshapes the vector representation of the subject s and that of the relation r into 2D matrices, and then concatenates the two matrices and feeds them into a 2D convolutional layer to extract higher-level, non-linear features, as illustrated in Figure 1(a). The resultant convolutional features are finally projected and matched with the vector representation of the object o via an inner product. Note that by sliding across the embeddings using small-sized filters, the convolution operator can easily generate much more features without increasing the embedding size. As such, ConvE offers increased expressiveness and achieves competitive performance in link prediction.

Nevertheless, despite its success, ConvE is still insufficient to fully capture the interactions between input entities and relations, which has long been recognized as crucial for modeling multi-relational data (Nickel et al., 2011; García-Durán et al., 2014; Trouillon et al., 2016). In ConvE, the (reshaped) representations of input entities and relations are simply stacked together and fed into a convolutional layer. Although 2D convolution is better than 1D convolution in modeling entity-relation interactions, typical 2D convolution with global filters on such a stacked matrix, however, can only model interactions around the concatenation line (Dettmers et al., 2018). Consider the example in Figure 1(a), where two matrices of size 3×3 are formed after reshaping, stacked, and fed as input to a convolutional layer. Convolving across the input with a global filter of size 2×2 will then be able to model interactions only in the regions where the two matrices adjoin (e.g., the region outlined in red). That means, only a small proportion of the output convolutional features (20% in this example, striped with orange and blue) will effectively capture entity-relation interactions, and the vast majority others will be entity- or relation-independent. This poses potential negative impacts on the link prediction task.

This paper, aiming at maximizing the interactions between input entities and relations, introduces ConvR, an adaptive convolutional network specifically designed for multi-relational data. As illustrated in Figure 1(b), the key idea of ConvR is to facilitate convolution across entity representations with its filters adaptively constructed from relation representations. Such adaptive convolution will model the interactions between the two types of input not only more naturally but also more effectively. Specifically, given a triple, the vector representation of the subject is reshaped and fed as input to a convolutional layer, while that of the relation is split and reshaped into a set of filters. ConvR then convolves across the input with these filters, enabling each filter (a part of the relation representation) to interact with diverse regions of the input (the entity representation). Through this adaptive convolution process, all the features generated will be able to capture entity-relation interactions (striped with orange and blue in Figure 1(b)). These convolutional features are finally projected and matched with the representation of the object.

Besides being more effective, adaptive convolution enables potentially more efficient modeling (in terms of the number of parameters). Compared with ConvE (Figure 1(a)), ConvR (Figure 1(b)) needs no global filters and generates smaller feature maps, making the follow-up projection layer roughly half as large as that of ConvE. The idea of adaptive convolution, in fact, is rather generic for the multi-relational scenario. By splitting and reshaping relation vectors, ConvR can be easily generalized to other paradigms such as 1D or 3D convolution, not restricted to the 2D setting. To facilitate a direct and fair comparison to ConvE where only 2D convolution is considered and tested, this paper takes the 2D setting as an example, and shows the superiority of ConvR over ConvE in this setting. We will investigate higher dimensional convolution in our future work.

Our contributions are as follows. (1) We propose a novel adaptive convolution model for learning with multi-relational data. Our approach, ConvR, takes full advantage of entity-relation interactions in a convolutional fashion, while still remaining highly parameter efficient. (2) We evaluate ConvR in the link prediction task on KBs and achieve very promising results on multiple benchmark datasets, including not only the popular WN18 and FB15K (Bordes et al., 2013), but also the more difficult WN18RR (Dettmers et al., 2018) and FB15K-237 (Toutanova and Chen, 2015). (3) We systematically compare the efficiency and effectiveness of ConvR and ConvE on FB15K-237, showing that ConvR can perform substantially better with a good variety of parameter settings. In particular, it offers a 7%



(a) ConvE: Convolution with global filters. (b) ConvR: Convolution with relation-specfic filters.

Figure 1: Reshaping and convolution in ConvE and ConvR. Entity-related neurons are marked in orange, relation-related ones striped with blue backslash, and those capturing entity-relation interactions striped with slash. White blocks stand for global filters applied to all input entities and relations.

increase in MRR and a 6% increase in Hits@10, with the total parameter number only 88% as large as that of ConvE.

2 Background

We consider multi-relational data represented as a graph, which can also be formalized as a set of subject-relation-object triples $\mathcal{G} = \{(s, r, o)\} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Here, $\mathcal{E}$ is the set of entities, and $\mathcal{R}$ the set of relations. Each triple (s, r, o) is composed of a subject entity $s \in \mathcal{E}$, a relation $r \in \mathcal{R}$, and an object entity $o \in \mathcal{E}$, indicating that there exists a relation of type r between the two entities s and o. Such triples are also called facts in knowledge bases (KBs).

We follow (Dettmers et al., 2018) and formalize link prediction on multi-relational data as a point-wise learning to rank problem, where the objective is to learn a scoring function $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$. For any input triple (s, r, o), the higher the score $\psi(s, r, o)$, the more likely the triple is true. Various statistical relational learning (SRL) techniques have been proposed for this task. See (Nickel et al., 2016a) for a thorough review of such techniques, with their application on large-scale KBs.

This paper focuses on vector space embedding models, a branch of SRL with superior performance and potential scalability. Given an input triple (s, r, o), a model of this kind first maps the entities s, o and relation r to their distributed representations (i.e., embeddings), usually vectors $\mathbf{s}, \mathbf{r}, \mathbf{o} \in \mathbb{R}^d$ for efficient learning. A score is then defined for the triple by operating on these distributed representations, i.e., $\psi(s, r, o) = \phi(\mathbf{s}, \mathbf{r}, \mathbf{o})$. A great many approaches of this kind have been devised in the last few years, where a key difference is the designing of the scoring function $\phi(\mathbf{s}, \mathbf{r}, \mathbf{o})$. See (Wang et al., 2017) for a recent survey.

3 Adaptive Convolution on Multi-relational Data

This section presents ConvR, an adaptive convolutional network specifically designed for learning with multi-relational data. The key idea of ConvR is to facilitate convolution across entity representations with its filters adaptively constructed from relation representations, so as to maximize the interactions between the two types of input. Figure 1(b) provides a simple illustration of this idea. In the rest of this section, we detail the ConvR model, discuss parameter learning of it, and show its advantages over ConvE, a convolutional network achieving promising results in multi-relational link prediction (Dettmers et al., 2018). To facilitate a direct and fair comparison to ConvE, we focus on the 2D setting. But the idea can be easily generalized to other convolution paradigms.

The ConvR model Given a triple (s, r, o), ConvR maps the two entities s, o to vectors $\mathbf{s}, \mathbf{o} \in \mathbb{R}^{d_e}$, and the relation r to vector $\mathbf{r} \in \mathbb{R}^{d_r}$, where d_e and d_r are the embedding size of entities and relations, respectively. Then, the subject vector $\mathbf{s}$ is reshaped into a 2D matrix $\mathbf{S} \in \mathbb{R}^{d_e^h \times d_e^w}$ (where $d_e = d_e^h d_e^w$) and fed as input to a convolutional layer. As shown in (Dettmers et al., 2018), using 2D rather than 1D convolution would be able to extract more feature interactions and increase model expressiveness. The relation vector $\mathbf{r}$ is further split into blocks $\mathbf{r}^{(1)}, \cdots, \mathbf{r}^{(c)}$ with equal size, where each $\mathbf{r}^{(\ell)} \in \mathbb{R}^{d_r/c}$ is reshaped into a 2D convolution filter $\mathbf{R}^{(\ell)} \in \mathbb{R}^{h \times w}$. Here, c is the number of filters, h and w the height and width of each filter, and $d_r = chw$. Figure 1(b) gives a simple example of this reshaping process, where a subject vector of length 9 is reshaped into a 3×3 matrix, and a relation vector of length 8 is split and reshaped into

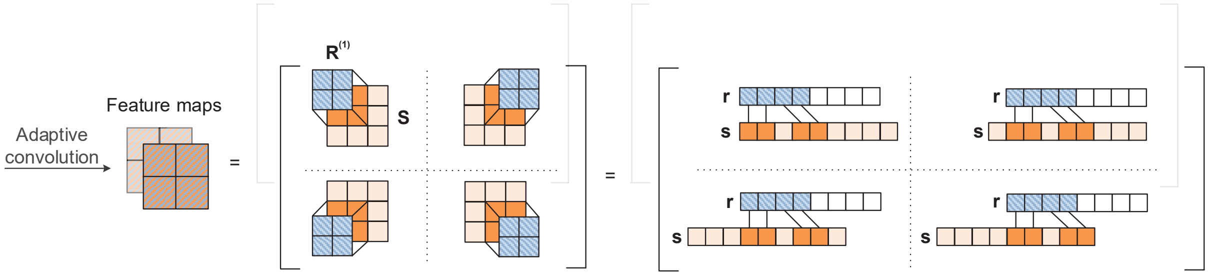


Figure 2: A simple illustration of adaptive convolution, where a 2×2 filter $\mathbf{R}^{(1)}$ (constructed from the first half of the relation vector $\mathbf{r}$) is convolved across a 3×3 input $\mathbf{S}$ (reshaped from the subject vector $\mathbf{s}$), generating a feature map of size 2×2. Each entry of the feature map could be calculated with certain dimensions of $\mathbf{r}$ and $\mathbf{s}$, marked with blue backslash and orange respectively.

two 2×2 filters.[1]

After reshaping, ConvR convolves across the input $\mathbf{S}$ using these adaptively constructed, relation-specific filters. For each filter $\mathbf{R}^{(\ell)}$, a convolutional feature map $\mathbf{C}^{(\ell)} \in \mathbb{R}^{(d_e^h - h + 1)\times(d_e^w - w + 1)}$ will be generated, with the mn-th entry calculated as:

$$c^{(\ell)}_{m,n} = f\left(\sum\nolimits_{i,j} s_{m+i-1,n+j-1} \times r^{(\ell)}_{i,j}\right), \qquad (1)$$

where $f(\cdot)$ is a non-linear function, e.g., ReLU (Krizhevsky et al., 2012). Figure 2 visualizes how such a feature map could be generated by convolving across the input with a relation-specific filter (the first equality sign "="), and how each entry of the feature map could be calculated with the original entity and relation vectors (the second equality sign "="). We can see that the adaptive convolution paradigm is quite effective in modeling entity-relation interactions. It enables rich interactions between input entity and relation representations at diverse regions, and all the convolutional features generated will be able to capture such interactions.

Finally, to compute the triple score $\psi(s, r, o)$, we flatten the convolutional feature maps $\mathbf{C}^{(1)}, \cdots, \mathbf{C}^{(c)}$ and stack them into a single vector $\mathbf{c}$. This vector is then projected into $\mathbb{R}^{d_e}$ by a fully-connected layer, and matched with the object vector $\mathbf{o}$ with an inner product, i.e.,

$$\psi(s, r, o) = f(\mathbf{W}\mathbf{c} + \mathbf{b})^{\top}\mathbf{o}, \qquad (2)$$

where $\mathbf{W} \in \mathbb{R}^{d_e \times c(d_e^h - h + 1)(d_e^w - w + 1)}$ and $\mathbf{b} \in \mathbb{R}^{d_e}$ are parameters of the fully-connected layer, and $f(\cdot)$ is again a non-linear function.

Parameter learning For learning the model parameters, we follow (Dettmers et al., 2018) and use 1-to-many scoring to speed-up training and evaluation. Unlike traditional 1-to-1 scoring which takes a triple (s, r, o) as input and directly scores it, 1-to-many scoring takes (s, r) as input and scores it against all candidate objects $o \in \mathcal{E}$ simultaneously, generating a score vector $\mathbf{p}^{s,r} \in \mathbb{R}^{|\mathcal{E}|}$. Each dimension of this score vector corresponds to an entity $o \in \mathcal{E}$, calculated as $p_o^{s,r} = \sigma(\psi(s, r, o))$, where $\psi(s, r, o)$ is the triple score defined in Eq. (2) and $\sigma(x) = \frac{1}{1+e^{-x}}$ the sigmoid function. For each input (s, r), we minimize the following cross-entropy loss:

$$\begin{aligned}\mathcal{L}(s,r) = -\frac{1}{|\mathcal{E}|}\sum_{o \in \mathcal{E}} y_o^{s,r}\log(p_o^{s,r}) + \\ (1 - y_o^{s,r})\log(1 - p_o^{s,r}),\end{aligned} \qquad (3)$$

where $y_o^{s,r}$ is a binary label. We have $y_o^{s,r} = 1$ if (s, r, o) is a valid triple and $y_o^{s,r} = 0$ otherwise. During optimization, we use dropout (Srivastava et al., 2014) to prevent overfitting. Specifically, we use dropout on the reshaped subject representations, the convolutional feature maps, and the projected vectors after the fully-connected layer. We also use batch normalization (Ioffe and Szegedy, 2015) on these representations to stabilize and speed up convergence. We use Adam (Kingma and Ba, 2014) optimizer and label smoothing (Szegedy et al., 2016) as suggested by ConvE.

Advantages over ConvE The most prominent advantage of ConvR over ConvE is its high ability to model entity-relation interactions in a convolutional fashion, which is crucial for learning with multi-relational data. ConvE, which convolves across stacked entity-relation representations with global filters, can only model interactions between

---

[1] During reshaping we consider the most natural ordering of the embedding entries. That means, a length-x vector is reshaped into a $y \times z$ matrix ($x = yz$) such that the first row of the matrix comes from the first z entries of the vector, the second row from the second z entries, and the y-th row from the last z entries.

the two types of input around the concatenation line, and only a small proportion of the convolutional features would be able to capture such interactions (see Figure 1(a)). ConvR, by contrast, enables entity-relation interactions at diverse regions, and all the convolutional features are able to capture such interactions (see Figure 1(b) and Figure 2).

Besides being more effective, ConvR might potentially be more efficient (in terms of the number of parameters). ConvE has a space complexity of $\mathcal{O}(d|\mathcal{E}| + d|\mathcal{R}| + chw + cd(2d^h - h + 1)(d^w - w + 1))$, where $d|\mathcal{E}|$ is to store entity vectors, $d|\mathcal{R}|$ relation vectors, chw the c global filters with size $h \times w$, $cd(2d^h - h + 1)(d^w - w + 1)$ the projection matrix in the fully-connected layer, and $d = d^h d^w$. As entity and relation representations need to be stacked in ConvE, they are usually of the same size, say d. In ConvR, convolution filters are adaptively constructed from relation vectors, so there is no need for global filters. Also, the input of the convolutional layer will be half-sized, generating smaller feature maps, and hence requires a smaller fully-connected layer. The space complexity of ConvR would be $\mathcal{O}(d_e|\mathcal{E}| + d_r|\mathcal{R}| + cd_e(d_e^h - h + 1)(d_e^w - w + 1))$. Although it could be possible to use different configuration of those common arguments in the two methods (e.g., different number of filters or entity vectors with different size), which may result in different memory cost, we empirically show that ConvR can perform substantially better than ConvE with a good variety of configurations, even those with fewer parameters (see the section “Parameter efficiency of ConvR” for details).

4 Experiments

In this section, we evaluate ConvR against competitive baselines in the link prediction task on multiple benchmark KBs. We also investigate parameter efficiency of ConvR against ConvE to further show its superiority.

4.1 Experimental Setup

Datasets We use four datasets for our experiments. The first two are the popular WN18 and FB15k, both released by (Bordes et al., 2013).[2] WN18 is a subset of WordNet for lexical relationships between words, and FB15k a subgraph of Freebase for generic facts. In most cases WN18 and FB15k encode a relation and its inverse relation at the same time. That means, once a fact is observed, there are usually two distinct triples created for it, e.g., $(s, \text{hyponym}, o)$ and $(o, \text{hypernym}, s)$, or $(s, \text{director-of}, o)$ and $(o, \text{directed-by}, s)$. As pointed out by (Toutanova and Chen, 2015) and (Dettmers et al., 2018), encoding inverse relations might suffer from test leakage, i.e., for each test triple (s, r, o), it is likely to find its inverse (o, r^{-1}, s) in the training set. To avoid this test leakage issue, we further use WN18RR (Dettmers et al., 2018),[3] a subset of WN18 with inverse relations removed, and FB15k-237 (Toutanova and Chen, 2015),[4] a filtered version of FB15k with both inverse and duplicate relations removed. Table 1 summarizes the statistics of the four datasets, where the training sets are used for parameter learning, the validation sets for hyperparameter tuning, and the test sets for evaluation.

Dataset	# Rel	# Ent	# Train	# Valid	# Test
FB15k	1,345	14,951	483,142	50,000	59,071
WN18	18	40,943	141,442	5,000	5,000
FB15k-237	237	14,541	272,115	17,535	20,466
WN18RR	11	40,943	86,835	3,034	3,134

Table 1: Statistics of the four datasets. Columns stand for the number of relations, number of entities, and number of triples in training/validation/test sets.

Evaluation protocol We adopt the ranking process proposed in (Bordes et al., 2013) for evaluation. For each triple (s, r, o) in the test set, we replace the subject s with every entity $e \in \mathcal{E}$, and calculate a score for the corrupted triple (e, r, o). Then we sort these scores in descending order to get the rank of the correct subject s. Since corrupted triples may also be valid, we remove those that already exist in either the training, validation, or test set during ranking, i.e., the *filtered* setting as called by (Bordes et al., 2013). This whole procedure is repeated while replacing the object o. We aggregate over all test triples, and report the mean reciprocal rank (MRR) and the proportion of correct entities ranked in the top n (Hits@n), with $n = 1, 3, 10$.

Implementation details We implement ConvR in PyTorch. In our experiments, we fix mini-batch

---

[2] `https://everest.hds.utc.fr/doku.php?id=en:smemlj12`

[3] `https://github.com/TimDettmers/ConvE/blob/master/WN18RR.tar.gz`

[4] `https://www.microsoft.com/en-us/download/details.aspx?id=52312`

Dataset	d_e	c	$h \times w$	ρ_1	ρ_2	ρ_3
FB15k	200	100	3×3	0.1	0.4	0.2
WN18	200	100	3×3	0.4	0.3	0.3
FB15k-237	100	100	5×5	0.3	0.2	0.3
WN18RR	200	200	3×3	0.2	0.2	0.5

Table 2: Optimal configurations of ConvR on the four datasets. Columns are the entity embedding size, number and size of filters, and dropout ratios.

size to 128, initial learning rate to 0.001, and label smoothing coefficient to 0.1. Other hyperparameters are selected with grid search on the validation set. Specifically, we tune the entity embedding size $d_e \in \{100, 200\}$, filter number $c \in \{50, 100, 150, 200\}$, and filter size $h \times w \in \{3 \times 3, 4 \times 4, 5 \times 5\}$. All dropout ratios, i.e., ρ_1 on reshaped subject representations, ρ_2 on convolutional feature maps, and ρ_3 on projected vectors after the fully-connected layer, are tuned in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. On each dataset, we choose the optimal configuration with the highest MRR on the validation set within 1000 epochs, and report its performance on the test set. Table 2 lists the optimal configurations of ConvR on the four datasets.

Baseline methods We compare ConvR against a variety of competitive baselines, which can be roughly categorized into two groups:

- Methods that use (relatively) simple operations in vector space to model multi-relational data, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015) and its reimplementation (Kadlec et al., 2017), HolE (Nickel et al., 2016b), ComplEx (Trouillon et al., 2016), ANALOGY (Liu et al., 2017), TorusE (Ebisu and Ichise, 2017), Gaifman (Niepert, 2016), KBGAN (Cai and Wang, 2017), KBLRN (Garcia-Duran and Niepert, 2018), and Node+LinkFeat (Toutanova and Chen, 2015).

- Methods that further introduce multi-layer structures and non-linearity, in particular those based on neural networks, including R-GCN (Schlichtkrull et al., 2017a), Neural LP (Yang et al., 2017), ConvE (Dettmers et al., 2018), and ConvKB (Nguyen et al., 2018).

4.2 Link Prediction Results

Table 3 reports the results on WN18 and FB15k, and Table 4 the results on WN18RR and FB15k-237. On all the four datasets, the results for the baselines are taken directly from previous literature to avoid re-implementation bias. Since not all baselines have their results reported on all the four datasets, we cannot make the two sets of baselines compared in Table 3 and Table 4 exactly the same. From the results, we can see that: (1) On WN18 and FB15k, ConvR performs better than or at least as well as the baselines in almost all the metrics. (2) Compared to ConvE, it offers a 5% increase in MRR, a 7% increase in Hits@1, and a 2% increase in Hits@10 on FB15k. (3) On the more difficult WN18RR and FB15k-237, ConvR consistently outperforms most of the baselines, except for MRR score of ConvKB on FB15k-237. However, on WN18RR ConvR outperforms ConvKB on all known metrices, especially MRR. This discrepancy may be attributed to ConvKB's initialization with TransE on FB15k-237. (4) Compared to ConvE, it offers a 3% increase in MRR, a 14% increase in Hits@1, a 12% increase in Hits@10 on WN18RR, and an 11% increase in MRR, a 9% increase in Hits@1, an 8% increase in Hits@10 on FB15k-237.

4.3 Parameter Efficiency of ConvR

We further investigate parameter efficiency of ConvR against ConvE on FB15k-237. Specifically, we tune the number of filters $c \in \{20, 40, 60, 80, 100\}$ and the filter size $h \times w \in \{2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5\}$, fix the other hyperparameters to their optimal configurations (see Table 2 for details), and show how the performance of ConvR (on the test set) will change as the number of parameters varies. For comparison, we directly show the performance and parameter efficiency of the optimal ConvE model, as reported in (Dettmers et al., 2018). The results are given in Table 5.[5]

From the results, we can see that: (1) The parameter number of ConvR steadily grows as the filter number c and filter size $h \times w$ increase, but the performance does not change much. That means, ConvR might achieve relatively good (though not best) performance with a potentially small number

---

[5] Note that some results reported here are even better than those reported in Table 4. This is because in Table 4 we determine optimal configurations according to MRR on the validation set, which may not necessarily lead to best performance on the test set.

	WN18				FB15k			
			Hits				Hits	
	MRR	@1	@3	@10	MRR	@1	@3	@10
TransE (Bordes et al., 2013) †	0.454	0.089	0.823	0.934	0.380	0.231	0.472	0.641
DistMult (Yang et al., 2015) †	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824
DistMult(Kadlec et al., 2017)	0.797	–	–	0.946	0.798	–	–	**0.893**
HolE (Nickel et al., 2016b)	0.938	0.930	0.945	0.949	0.524	0.402	0.613	0.739
ComplEx (Trouillon et al., 2016)	0.941	0.936	0.945	0.947	0.692	0.599	0.759	0.840
ANALOGY (Liu et al., 2017)	0.942	0.939	0.944	0.947	0.725	0.646	0.785	0.854
TorusE (Ebisu and Ichise, 2017)	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832
Gaifman (Niepert, 2016)	–	0.761	–	0.939	–	0.692	–	0.842
KBLRN (Garcia-Duran and Niepert, 2018)	–	–	–	–	0.794	**0.748**	–	0.875
Node+LinkFeat (Toutanova and Chen, 2015)	0.940	–	–	0.943	**0.822**	–	–	0.870
R-GCN (Schlichtkrull et al., 2017b)	0.814	0.686	0.928	0.955	0.651	0.541	0.736	0.825
Neural LP (Yang et al., 2017)	0.94	–	–	0.945	0.76	–	–	0.837
ConvE (Dettmers et al., 2018)	0.942	0.935	0.947	0.955	0.745	0.670	0.801	0.873
ConvR (this work)	**0.951**	**0.947**	**0.955**	**0.958**	0.782	0.720	**0.826**	0.887

Table 3: Link prediction results on the test sets of WN18 and FB15k. Results marked by † are taken from (Trouillon et al., 2016). Other results are taken from the original papers. Missing scores not reported are denoted by "–". Best scores highlighted in bold.

	WN18RR				FB15k-237			
			Hits				Hits	
	MRR	@1	@3	@10	MRR	@1	@3	@10
DistMult (Yang et al., 2015) ‡	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
ComplEx (Trouillon et al., 2016) ‡	0.44	0.41	0.46	0.51	0.247	0.158	0.275	0.428
KBGAN (Cai and Wang, 2017)	0.214	–	–	0.472	0.278	–	–	0.458
KBLRN (Garcia-Duran and Niepert, 2018)	–	–	–	–	0.309	0.219	–	0.493
Node+LinkFeat (Toutanova and Chen, 2015)	–	–	–	–	0.226	–	–	0.347
R-GCN (Schlichtkrull et al., 2017b)	–	–	–	–	0.248	0.153	0.258	0.417
Neural LP (Yang et al., 2017)	–	–	–	–	0.24	–	–	0.362
ConvE (Dettmers et al., 2018)	0.46	0.39	0.43	0.48	0.316	0.239	0.350	0.491
ConvKB (Nguyen et al., 2018)	0.248	–	–	0.525	**0.396**	–	–	0.517
ConvR (this work)	**0.475**	**0.443**	**0.489**	**0.537**	0.350	**0.261**	**0.385**	**0.528**

Table 4: Link prediction results on the test sets of WN18RR and FB15k-237. Results marked by ‡ are taken from (Dettmers et al., 2018). Other results are taken from the original papers. KBGAN refers to the "TransD + DistMult" setting which shows best performance. ConvKB is initialized with TransE on FB15k-237, and randomly on WN18RR. Missing scores not reported are denoted by "–". Best scores highlighted in bold.

of parameters. (2) ConvR consistently and substantially outperforms the best performing ConvE with all the configurations listed in Table 5. (3) In particular, even the most efficient configuration (i.e., $c = 20$ and $h \times w = 2 \times 2$) offers a 7% increase in MRR and a 6% increase in Hits@10, with its parameter number only 88% as large as that of ConvE.

5 Related Work

Link prediction is a crucial task for knowledge bases (KBs). A good variety of statistical relational learning techniques have been proposed for this task (Nickel et al., 2016a), among which vector space embedding models are most particular due to their superior performance and potential scalability. Early works of this kind tend to employ simple vector space operations for link prediction. For example, TransE (Bordes et al., 2013) takes relations as translations between subject and object entities. DistMult (Yang et al., 2015) uses multilinear dot product to characterize three-way interactions among subjects, relations, and objects. ComplEx (Trouillon et al., 2016) further generalizes DistMult to complex vector space. Using simple models allows one to easily handle large-scale KBs, but usually at the cost of less model expressiveness (Dettmers et al., 2018). HolE (Nickel et al., 2016b) tries to increase model expressiveness while keeping simplicity. It uses cross-correlation, i.e., the

	$h \times w = 2\times 2$	$h \times w = 3\times 3$	$h \times w = 4\times 4$	$h \times w = 5\times 5$
ConvE	–	1.89M | 0.32 | 0.49	–	–
ConvR, $c = 20$	1.67M | 0.342 | 0.520	1.68M | 0.342 | 0.522	1.72M | 0.342 | 0.522	1.78M | 0.342 | 0.522
ConvR, $c = 40$	1.87M | 0.345 | 0.526	1.90M | 0.344 | 0.524	1.97M | 0.348 | 0.529	2.09M | 0.347 | 0.526
ConvR, $c = 60$	2.07M | 0.347 | 0.525	2.11M | 0.348 | 0.530	2.22M | 0.350 | 0.529	2.40M | 0.350 | 0.527
ConvR, $c = 80$	2.27M | 0.347 | 0.528	2.32M | 0.348 | 0.532	2.47M | 0.350 | 0.532	2.71M | 0.348 | 0.528
ConvR, $c = 100$	2.47M | 0.348 | 0.531	2.54M | 0.348 | 0.527	2.72M | 0.349 | 0.527	3.02M | 0.350 | 0.528

Table 5: Parameter efficiency on FB15k-237. Each cell reports number of parameters, MRR, and Hits@10 in turn. Results of ConvE are taken from (Dettmers et al., 2018).

inverse of circular convolution, to match subject and object entities, which has some similarity to our work. But HolE is not a typical neural network architecture. It does not learn multiple layers of non-linear features, and hence is less expressive than our approach.

A more direct way of increasing model expressiveness is to employ deeper, more complicated neural network architectures, e.g., multi-layer perceptron (Dong et al., 2014), semantic matching energy networks (Bordes et al., 2014), and neural tensor networks (Socher et al., 2013). This kind of approaches, however, often have more parameters and are prone to overfit (Nickel et al., 2016a). (Dettmers et al., 2018) recently devised ConvE, a multi-layer convolutional network which offers increased model expressiveness while remaining highly parameter efficient. After that, (Nguyen et al., 2018) propose ConvKB that explores the global relationships among same dimensional entries of the entity and relation embeddings. However, neither of them models the interactions between various positions of entities and relations. R-GCN (Schlichtkrull et al., 2017a) is another convolutional network designed for KBs, generalized from GCN (Kipf and Welling, 2016) for uni-relational data. But the convolution of R-GCN is conducted in a message passing manner, quite different from our work.

Convolutional neural networks have been successfully applied to a wide variety of domains, ranging from speech or visual recognition (Abdel-Hamid et al., 2014; Krizhevsky et al., 2012) to natural language processing (Collobert et al., 2011). Similar ideas of using adaptive or dynamic convolutional filters have been studied before (Lee et al., 2010; Jia et al., 2016; Kang et al., 2017). But most of such works focus on image or video processing. This work focuses on multi-relational data and devises an adaptive convolution paradigm particularly suitable for this scenario.

6 Conclusion

In this paper, we propose ConvR, an adaptive convolutional network specially designed for learning with multi-relational data. In contrast to previous work which convolves across stacked representations with global filters, ConvR adaptively constructs convolution filters from relation representations, and applies these filters across entity representations to generate convolutional features. This adaptive convolution paradigm enables rich interactions between entity and relation representations at diverse regions, and all convolutional features generated in this way will be able to capture such interactions. Experimental results on multiple benchmark knowledge bases show that ConvR achieves significant and consistent improvements against a variety of baselines. In particular, it is not only more effective but also more efficient than state-of-the-art convolutional models, offering a 7% increase in MRR and a 6% increase in Hits@10, while saving 12% in parameter storage.

As future work, we plan to devise convolutional paradigms that can maximize interactions not only between subject entities and relations, but also between object entities and relations. In ConvR, we use 1-to-many scoring to speed up training and evaluation. As a side effect, object representations can only interact with a hidden vector (output of the fully-connected layer) via an inner product, which potentially limits the performance of ConvR. It is worth investigating modeling these interactions while keeping the merit of fast training and evaluation.

Acknowledgments

We would like to thank all the anonymous reviewers for their insightful and valuable suggestions, which help to improve the quality of this paper. This work is supported by the National Natural Science Foundation of China (grant No. 61876223).

References

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Liwei Cai and William Yang Wang. 2017. KBGAN: adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Takuma Ebisu and Ryutaro Ichise. 2017. TorusE: Knowledge graph embedding on a lie group. *arXiv preprint arXiv:1711.05435*.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2014. Effective blending of two and three-way interactions for modeling multi-relational data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 434–449.

Alberto Garcia-Duran and Mathias Niepert. 2018. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. MIT press.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R. Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. 2016. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74.

Di Kang, Debarun Dhar, and Antoni Chan. 2017. Incorporating side information by adaptive convolution. In *Advances in Neural Information Processing Systems*, pages 3870–3880.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Kyong Joon Lee, Dongjin Kwon, Il Dong Yun, and Sang Uk Lee. 2010. Optical flow estimation with adaptive convolution kernel prior on discrete framework. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2504–2511.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2168–2178.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 327–333.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Mathias Niepert. 2016. Discriminative gaifman models. In *Advances in Neural Information Processing Systems*, pages 3405–3413.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017a. Modeling relational data with graph convolutional networks. *arXiv:1703.06103*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017b. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2316–2325.

Graph Pattern Entity Ranking Model for Knowledge Graph Completion

Takuma Ebisu[1,2] and **Ryutaro Ichise**[2,1,3]
[1]SOKENDAI, Chiyoda-ku, Tokyo, Japan
[2]National Institute of Informatics, Chiyoda-ku, Tokyo, Japan
[3]National Institute of Advamced Industrial Science and Technology, Koto-ku, Tokyo, Japan
{takuma,ichise}@nii.ac.jp

Abstract

Knowledge graphs have evolved rapidly in recent years and their usefulness has been demonstrated in many artificial intelligence tasks. However, knowledge graphs often have lots of missing facts. To solve this problem, many knowledge graph embedding models have been developed to populate knowledge graphs and these have shown outstanding performance. However, knowledge graph embedding models are so-called black boxes, and the user does not know how the information in a knowledge graph is processed and the models can be difficult to interpret. In this paper, we utilize graph patterns in a knowledge graph to overcome such problems. Our proposed model, the *graph pattern entity ranking model* (GRank), constructs an entity ranking system for each graph pattern and evaluates them using a ranking measure. By doing so, we can find graph patterns which are useful for predicting facts. Then, we perform link prediction tasks on standard datasets to evaluate our GRank method. We show that our approach outperforms other state-of-the-art approaches such as ComplEx and TorusE for standard metrics such as HITS@n and MRR. Moreover, our model is easily interpretable because the output facts are described by graph patterns.

1 Introduction

Knowledge graphs can be used to describe real-world relations as facts in a form that a computer can easily process and has been used for many artificial intelligence tasks (Hakimov et al., 2012; Daiber et al., 2013; Bordes et al., 2014). In a knowledge graph, a fact is represented by a labeled and directed edge, called a triple (h, r, t), where h and t are entity nodes and r is a relation label of an edge from h to t. Knowledge graphs such as YAGO (Suchanek et al., 2007), DBpedia (Auer et al., 2007), and Freebase (Bollacker et al., 2008) have developed rapidly in recent years and are used for many artificial intelligence tasks such as question answering, content tagging, fact-checking, and knowledge inference. Although some knowledge graphs already contain millions of entities and billions of facts, they might still be incomplete and some facts may be missing. Hence, we need to develop a system that can predict missing facts to complete knowledge graphs automatically.

Many kinds of models for *link prediction* have been developed to estimate unknown facts. Knowledge graph embedding models, which are the most widely used approach in this field, map entities and relations in a knowledge graph onto a vector space and obtain the latent underlying features. However, these models are generally difficult to interpret, as we do not know how information is processed in the models and the predicted facts are output without explanation.

In this paper, we construct statistical models based on *graph pattern matching*. These models are not only easy to interpret compared to knowledge graph embedding models but also outperform state-of-the-art models for link prediction.

Our main contributions in this paper are as follows:

- Defining *graph pattern association rules* (GPARs) for a knowledge graph.
- Introducing *a graph pattern probability model* (GPro) and discussing its flaws.
- Proposing a novel model, *the graph pattern entity ranking model* (GRank), which uses graph patterns to rank entities.
- Proposing distributed rankings to address the problem arising from having the same score for multiple entities.

Proceedings of NAACL-HLT 2019, pages 988–997
Minneapolis, Minnesota, June 2 - June 7, 2019.

- Evaluating the proposed models through link prediction tasks for standard datasets: It is shown that our model outperforms most state-of-the-art knowledge graph embedding models for the HITS@n and MRR metrics.

The remainder of this paper is organized as follows. In Section 2, we discuss related work on link prediction. In Section 3, we define the terms and notation used in this paper. In Section 4, we define standard confidences for GPARs and discuss their problems. In Section 5, we propose the GRank model to deal with these problems. In Section 6, we present an experimental study in which we compare our models with baseline results for benchmark datasets. In Section 7, we conclude this paper.

2 Related Work

We categorize related work for link prediction into two groups: work on knowledge graph embedding models (which are latent feature models) and work on observed feature models.

2.1 Knowledge Graph Embedding Models

Recently, knowledge graph embedding models have yielded great results in link prediction tasks. Knowledge graph embeddings models embed entities and relations on a continuous space and can be roughly classified into three types: translation-based models, bilinear models, and neural network-based models.

The first translation-based model was the TransE (Bordes et al., 2013) model, which gained attention because of its effectiveness and simplicity. TransE employs the principle $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$, where $\boldsymbol{h}, \boldsymbol{r}$ and $\boldsymbol{t}$ are the embeddings of h, r and t, respectively. While this principle efficiently captures first-order rules, the TransE approach still has some problems. The conflict between principle and regularization is one of these problems and the TorusE (Ebisu and Ichise, 2018) model was recently proposed to solve this problem by embedding entities and relations on a torus manifold.

RESCAL (Nickel et al., 2011) was the first bilinear model, where each relation is represented by a square matrix and the score of the triple (h, r, t) is calculated by a bilinear map which corresponds to the matrix of the relation r and whose arguments are $\boldsymbol{h}$ and $\boldsymbol{t}$. Hence, RESCAL represents the most general form of a bilinear model. Extensions of RESCAL have been proposed by restricting bilinear functions, for example, DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) restrict the matrices representing the relations to diagonal matrices.

Neural network-based models have layers and an activation function like a neural network. The Neural Tensor Network (NTN) (Socher et al., 2013) has a standard linear neural network structure and a bilinear tensor structure, and can be considered as a generalization of RESCAL, where the weight of the network is trained for each relation. Graph Convolutional Networks (GCNs) (Duvenaud et al., 2015; Defferrard et al., 2016; Kipf and Welling, 2017) exploit the convolution operator to capture local information for a graph, however these models are for undirected graphs. Relational GCNs (Schlichtkrull et al., 2017) and ConvE (Dettmers et al., 2018) are generalizations of GCNs for knowledge graphs.

Knowledge graph embedding is the standard approach for link prediction. However, it suffers from low interpretability, resulting in triples which are predicted without any clear reason.

2.2 Observed Feature Models

The main advantage of observed feature models over knowledge graph embedding models is their interpretability. Additionally, Toutanova et al. (2015) proposed a relatively simple logistic regression model, the Node+LinkFeat model, which utilizes only one-hop information in a knowledge graph and demonstrated that it performs far better for link prediction on standard datasets than most existing knowledge graph embedding models. However, it has also been shown that the Node+LinkFeat model cannot deal with a low-redundancy dataset because the model uses information which is too local. On the other hand, it has shown that a logistic regression model, the PRA model (Lao and Cohen, 2010; Lao et al., 2011), which utilizes multi-hop information do not have sufficient accuracy (Liu et al., 2016). This suggests logistic regression does not have enough power to deal with deep information. These studies have motivated research toward developing a more efficient model utilizing deeper information.

We begin by discussing GPARs, which were proposed recently by Fan et al. (2015), and have shown their usefulness for social network graphs

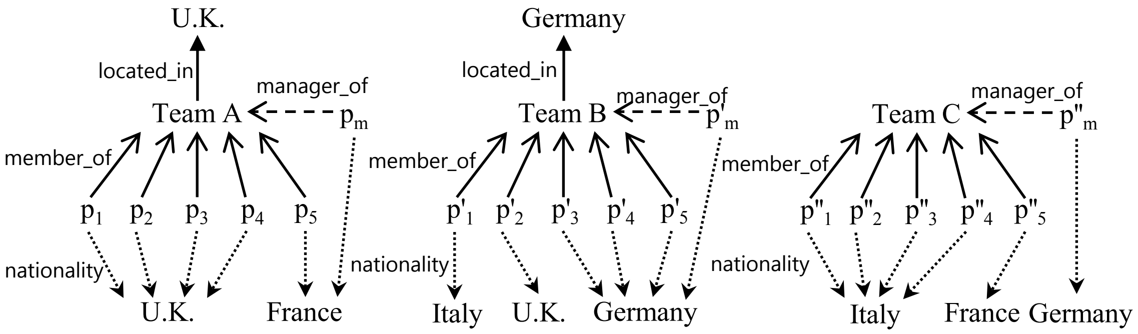


Figure 1: Graph G_{ex} of sports teams.

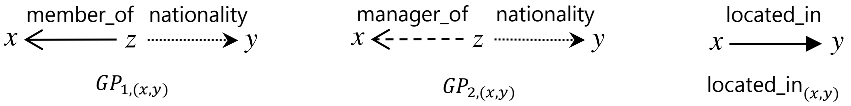


Figure 2: Graph patterns on G_{ex}.

because graph patterns can capture deeper information lying in a knowledge graph and a GPAR explicitly describe the process of prediction. However, the definition of GPARs by Fan et al. cannot be applied to a knowledge graph because Fan et al. assumes a different structure for a social network graph than a knowledge graph. In the following section, we define GPARs for a knowledge graph.

3 Preliminaries

In this section, we introduce the definitions and notation required to discuss GPAR-based models.

3.1 Graph Pattern Association Rules on Knowledge Graphs

We modify GPARs for application to a knowledge graph following the definitions of Fan et al. (2015).

Knowledge Graph: A *graph* is defined as $G = \{(h, r, t)\} \subset E \times R \times E$, where E denotes a set of entities and R denotes a set of relations. An element (h, r, t) of G is called a triple and represents the directed relation r between h and t.

An example graph G_{ex} is shown in Figure 1, where p_i represents a person, Teams "A", "B", and "C" represent sports teams, and countries are entities in E_{ex} with labeled arrows between two entities representing directed relations in R_{ex}.

Graph Pattern: A *graph pattern* on G is a graph $GP_{(x,y)} = \{(z_i, r, z_j)\} \subset V_{GP} \times R \times V_{GP}$, where V_{GP} denotes a set of variables, x and y are two designated variables, and R is the set of relations of G. We suppose V_GP has no redundancy, in other words, $\forall z \in V_{GP}, \exists(z_i, r, z_j) \in GP_{(x,y)}, z = z_i \lor z = z_j$.

Some examples of graph patterns on G_{ex} are shown in Figure 2, where $GP_{1,(x,y)} = \{(z, \text{member\_of}, x), (z, \text{nationality}, y)\}$, $GP_{2,(x,y)} = \{(z, \text{manager\_of}, x), (z, \text{nationality}, y)\}$, and $\text{located\_in}_{(x,y)} = \{(x, \text{located\_in}, y)\}$. Our focus in this paper is on finding useful graph patterns for link prediction.

Graph Pattern Matching Function: A *matching function* of $GP_{(x,y)}$ on $(h, t) \in E \times E$ is an injective function $m : V_{GP} \to E$ that satisfies the following conditions: $m(x) = h$, $m(y) = t$, and for all $(z_i, r, z_j) \in GP_{(x,y)}$, $(m(z_i), r, m(z_j)) \in G$. $M(GP_{(x,y)}, (h, t))$ denotes the set of all matching functions of $GP_{(x,y)}$ on (h, t). We say $GP_{(x,y)}$ matches (h, t) if there is at least one matching function of $GP_{(x,y)}$ on (h, t) (i.e. $M(GP_{(x,y)}, (h, t)) \neq \emptyset$).

For example, $m : V_{GP_{1,(x,y)}} \to E_{ex}$ $(m(x) = \text{Team A}, m(z) = \text{p}_1, m(y) = \text{U.K.})$ is a matching function of $GP_{1,(x,y)}$ on $(\text{Team A}, \text{U.K.})$.

GPAR: A *graph pattern association rule* (GPAR) AR is defined as $GP_{(x,y)} \Rightarrow r_{(x,y)}$, where $GP_{(x,y)}$ and $r_{(x,y)}$ are graph patterns and $r_{(x,y)} = \{(x, r, y)\}$.

For example, a GPAR $AR_1 = P_{1,(x,y)} \Rightarrow \text{located\_in}_{(x,y)}$ would indicate that if there is a matching function of $GP_{1,(x,y)}$ on (h,t), then it is likely that there is also a matching function of $\mathsf{located\_in}_{(x,y)}$ on (h,t), i.e. $(h, \mathsf{located\_in}, t)$ is a fact.

3.2 Reconstruction of Knowledge Graph to Queries

Our task is the *link prediction* of a knowledge graph, i.e. to predict the missing entity of a *query*, which is formally defined as follows:

Query: A *query* is a triple which is missing an entity: $(h, r, ?)$ or $(?, r, t)$.

We divide a knowledge graph G into queries and answers to use as training data for our model. Let $Q_{r,\text{head}}$ ($Q_{r,\text{tail}}$) denote the set of training queries missing a head (tail) entity for a relation r obtained from G; then $Q_{r,\text{head}}$ ($Q_{r,\text{tail}}$) is defined as follows:

$$\begin{aligned} Q_{r,\text{head}} &= \{(?, r, t) \mid (h, r, t) \in G\}, \\ Q_{r,\text{tail}} &= \{(h, r, ?) \mid (h, r, t) \in G\} \end{aligned}$$

In this case, the answers of training queries are defined as follows:

$$\begin{aligned} a_{(?,r,t)} &= \{h \mid (h, r, t) \in G\}, \\ a_{(h,r,?)} &= \{t \mid (h, r, t) \in G\} \end{aligned}$$

A knowledge graph usually contains only positive triples. Hence, we adopt the *partial completeness assumption* (PCA) (Galárraga et al., 2013, 2015) to generate negative answers.

Partial Completeness Assumption: if (h, r, t) is in G, then

$$\forall t' \in E, ((h, r, t') \notin G \Rightarrow (h, r, t') \text{ is negative}) \quad (1)$$

$$\forall h' \in E, ((h', r, t) \notin G \Rightarrow (h', r, t) \text{ is negative}) \quad (2)$$

The standard PCA definition consists only of Equation (1), but we add Equation (2) because we also need to allow negative answers for $Q_{r,\text{head}}$. Under PCA, negative answers for each question are defined as follows:

$$n_{(?,r,t)} = E \setminus a_{(?,r,t)}, \; n_{(h,r,?)} = E \setminus a_{(h,r,?)}$$

4 Standard Confidence and Problems

4.1 AMIE with GPARs

An association rule is essentially a binary classifier, i.e. the antecedent of an association rule matches or does not match, and an association rule is thus evaluated. Following this idea, we suggest the most straightforward way to define the *confidence*, which indicates the reliability of an association rule, is the conditional probability, which is the probability of the consequent given the antecedent for a GPAR. The conditional probability $Pr_{tail}(r_{(x,y)} \mid GP_{(x,y)})$ of a GPAR $GP_{(x,y)} \Rightarrow r_{(x,y)}$ to predict a tail is defined as follows:

$$conf_{tail}(GP_{(x,y)} \Rightarrow r_{(x,y)}) = Pr_{tail}(r_{(x,y)} \mid GP_{(x,y)}) = \frac{\sum_{(h,r,?) \in Q_{r,\text{tail}}} |\{t \in a_{(h,r,?)} \mid M(GP_{(x,y)}, (h,t)) \neq \emptyset\}|}{\sum_{(h,r,?) \in Q_{r,\text{tail}}} |\{t' \in E \mid M(GP_{(x,y)}, (h,t')) \neq \emptyset\}|}$$

For each query, the candidate entities found by the graph pattern are counted for the denominator while only correct entities are counted for the numerator. This confidence is used to evaluate GPARs only to answer queries with a missing tail because $Q_{r_{tail}}$ and its answers are used to define it.

Interestingly, GPARs with this confidence are equivalent to AMIE (Galárraga et al., 2013, 2015), which was proposed to find horn clauses for a knowledge graph, although AMIE was proposed before the appearance of GPARs. However, AMIE originally has only one confidence value for a GPAR because AMIE is not designed for link prediction. Hence, we introduce the following alternative definition for the confidence value to answer a query missing a head entity.

4.2 Standard Link Prediction Model and Problems

We define another confidence to deal with a query with a missing head entity as follows:

$$conf_{head}(GP_{(x,y)} \Rightarrow r_{(x,y)}) = Pr_{head}(r_{(x,y)} \mid GP_{(x,y)}) = \frac{\sum_{(?,r,t) \in Q_{r,\text{head}}} |\{h \in a_{(?,r,t)} \mid M(GP_{(x,y)}, (h,t)) \neq \emptyset\}|}{\sum_{(?,r,t) \in Q_{r,\text{head}}} |\{h' \in E \mid M(GP_{(x,y)}, (h',t)) \neq \emptyset\}|}$$

Additionally, we restrict matching functions to injective functions as defined in Section 3.1, which is different from AMIE, because the restriction avoids redundant matching functions which map multiple variables to the same entity and gives a good bias for real-world knowledge. For example, an GPAR $GP_{3,(x,y)} \Rightarrow \mathsf{sibling\_of}_{(x,y)}$, where $GP_{3,(x,y)} = \{(z, \mathsf{parent\_of}, x), (y, \mathsf{child\_of}, z)\}$, is helpful to predict siblings. However, let p represent a person, $GP_{3,(x,y)}$ matches (p,p) although p is not a sibling of p. The above restriction omits such concerns. For another example, a GPAR $\{(z_1, \mathsf{manager\_of}, x), (z_1, \mathsf{manager\_of}, z_2), (z_2, \mathsf{located\_in}, y)\} \Rightarrow \mathsf{located\_in}_{(x,y)}$ on the graph

G_{ex} in Figure 1 should not be considered helpful because $m(x) = m(z_2)$ holds for a matching function m of the antecedent pattern and as a result, the GPAR is almost tautological. We consider two confidence values for GPARs, $conf_{tail}$ and $conf_{head}$, referred to as the *graph pattern probability model* (GPro).

However, GPro cannot deal with queries where counting the number of matching functions is crucial. An example where the number of matching functions is important is shown in Figure 1. In G_{ex}, the country that Team C is located in is missing. One might guess that Team C is located in Italy because most of the Team C players have Italian nationality and the nationality of a player often matches the country that the team is located in. However, GPro underestimates the GPAR $AR_{1,(x,y)} = GP_{1,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$, which is equivalent to one's guessing process: $conf_{tail}(AR_{1,(x,y)}) = 2/5$, while $conf_{tail}(AR_{2,(x,y)}) = 1/2$, where $AR_{2,(x,y)} = GP_{2,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$. Hence, GPro judges $AR_{2,(x,y)}$ is more useful than $AR_{1,(x,y)}$, and as a result, GPro predicts Team C is located in Germany rather than Italy.

This problem is caused by considering a GPAR as a binary classifier, i.e. the matching number is not taken into account. For example, if we apply $AR_{1,(x,y)} = P_{1,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$ to a query (Team A, $\mathsf{located\_in}$, ?) in the traditional way (as a binary classifier), the output will contain two entities with equal weighting, the U.K. and France, because $P_{1,(x,y)}$ matches (Team A, U.K.) and (Team A, France). Then, one of the output entities is correct and the other is incorrect. This is the reason why $AR_{1,(x,y)}$ is underestimated.

To deal with this problem, in this paper, we consider a GPAR as an entity ranking system by counting the number of matching functions of the antecedent graph pattern rather than considering as a binary classifier.

5 GPAR as Entity Ranking System and Evaluation Metrics

As well as considering a GPAR as a binary classifier, we consider it as an entity ranking system. Entities are ranked according to a score, based on their number of matching functions.

Moreover, we introduce the *distributed rankings* for entities, which are proposed to deal with situations where multiple entities have the same score. Then, we define the evaluation metrics for the distributed rankings to evaluate GPARs for link prediction.

These approaches overcome the problems shown in Section 4.2.

5.1 GPAR as Entity Ranking System

We consider a GPAR as a ranking system in this section to rank queries for which counting the number of matching functions of the antecedent is helpful, as shown in Section 4.2.

First, we define a scoring function whose arguments are a graph pattern $GP_{(x,y)}$ and a pair of entities (h, t). The scoring function returns the number of matching functions of a pattern on a pair, which is formally defined as follows:

$$score(GP_{(x,y)}, (h, t)) = |M(GP_{(x,y)}, (h, t))|$$

Given a pattern $GP_{(x,y)}$ and a query $(h, r, ?)$, we can obtain the $score(GP_{(x,y)}, (h, t'))$ for each candidate tail entity t'. Then we obtain the rankings of the tail entities in descending order of the scores. The head entity rankings for a query $(?, r, t)$ are also obtained in this way. This ranking method gives us new perspective when we apply GPARs to answer a query. For example, if we apply $AR_{1,(x,y)} = P_{1,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$ to a query (Team A, $\mathsf{located\_in}$, ?) the U.K. will be ranked first and France second. In this situation, we can say that $AR_{1,(x,y)}$ works because the correct entity ranks higher than the wrong entity. We can basically evaluate a GPAR as an entity ranking system by evaluating output rankings by an evaluation metric for an ranking system such as the *mean average precision*. However, often multiple entities have the same score and traditional metrics cannot deal with the situation. To deal with this problem, we propose a new concept, called *distributed rankings*, and the corresponding metrics in the following sections.

5.2 Distributed Rankings

We propose distributed rankings where each entity can distribute over multiple ranks and each rank can have multiple entities, to deal with situations where multiple entities have the same score.

Traditional rankings of entities are represented by a matrix $Rank = (rank_{i,j}) \in \{0, 1\}^{n \times n}$, where n is the number of entities, and for each column and row there is one 1 element. In this matrix, columns represent entities and rows represent ranks. For example, $rank_{i,j} = 1$ means that the entity j has rank i. On the other hand, distributed rankings of entities are represented by a matrix $dRank = (drank_{i,j}) \in [0,1]^{n\times n}$, where the summation of a column or a row is equal to 1. Different from traditional rankings, the value of each element is continuous and multiple elements can be greater than 0 in a column or a row. For example, $rank_{i,j} = 0.5$ means that half of the entity j has rank i. Note that a traditional ranking matrix is a distributed ranking matrix.

Given a pattern $GP_{(x,y)}$ and a query $(h,r,?)$, We obtain distributed rankings of entities, $dRANK(GP_{(x,y)},(h,r,?))$, according to their scores as follows. Let a be the number of entities whose scores are greater than the entity represented by j and let b be the number of entities whose scores are the same as the entity represented by j. Then, $drank_{i,j}$, an element of $dRANK(GP_{(x,y)},(h,r,?))$, is determined to be $1/b$ for $a+1 \leqq i \leqq a+b$ and 0 otherwise. Distributed rankings of head entities for a query $(?,r,t)$ are obtained in the same way, and we refer to them as $dRANK(GP_{(x,y)},(?,r,t))$. Unlike traditional rankings, distributed rankings are uniquely determined from the scores of entities.

Traditional rankings can be evaluated by metrics such as the average precision or the cumulative gain. However, distributed rankings cannot be evaluated by these metrics. Hence, we require a different evaluation metric for distributed rankings.

5.3 Evaluation of GPARs as Entity Ranking System

We use a GPAR to obtain distributed entity rankings as shown in Section 5.1. In this section, we define a metric to evaluate distributed rankings of entities by generalizing the *average precision* to evaluate a GPAR.

For a pattern $GP_{(x,y)}$ and a training query $(h,r,?)$, the *distributed precision at k*, $dPre_k$, of $dRANK(GP_{(x,y)},(h,r,?))$ is defined as follows:

$$dPre_k(GP_{(x,y)},(h,r,?)) = \frac{\sum_{i=1}^{k}\sum_{t_j\in a_{(h,r,?)}} drank_{i,j}}{k}$$

where t_j is an entity represented by j and $drank_{i,j}$ is an element of $dRANK(GP_{(x,y)},(h,r,?))$. The elements related with correct entities ranked higher or equal to k are summed up as the traditional precision at k.

Dataset	# Entities	# Relations	# Training	# Validation	# Test
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466

Table 1: Statistics of benchmark datasets. The numbers of entities, relations, training triples, validation triples, and test triples are shown.

Then, the *distributed average precision*, *dAP*, is defined for a pattern $GP_{(x,y)}$ and a training query $(h,r,?)$ as follows:

$$dAP(GP_{(x,y)},(h,r,?)) = \frac{\sum_{t_j\in a_{(h,r,?)}}\sum_{k=1}^{n} dPre_k(GP_{(x,y)},(h,r,?))) \times drank_{k,j}}{|a_{(h,r,?)}|}$$

where t_j is an entity represented by j, $drank_{i,j}$ is an element of $dRANK(GP_{(x,y)},(h,r,?))$, and n is the number of entities. The numerator of the average precision for traditional rankings is the summation of the precision at k for relevant entities. However, a relevant entity represented by j is distributed over multiple ranks in $dRANK$ so that the precision at k multiplied by $drank_{k,j}$ is summed over k where a relevant entity j is distributed. $dAP(GP_{(x,y)},(?,r,t))$ for a training query with a missing head can be defined in the same way. The *distributed mean average precision* for a GPAR $GP_{(x,y)} \Rightarrow r_{(x,y)}$ is defined as follows:

$$\begin{aligned} &dMAP_{head}(GP_{(x,y)} \Rightarrow r_{(x,y)}) \\ &\quad = \sum_{(?,r,t)\in Q_{r,\mathrm{head}}} \frac{dAP(GP_{(x,y)},(?,r,t))}{|Q_{r,\mathrm{head}}|} \\ &dMAP_{tail}(GP_{(x,y)} \Rightarrow r_{(x,y)}) \\ &\quad = \sum_{(h,r,?)\in Q_{r,\mathrm{tail}}} \frac{dAP(GP_{(x,y)},(h,r,?))}{|Q_{r,\mathrm{tail}}|} \end{aligned}$$

We also define dMAP with for the "*filtered*" (Bordes et al., 2013) rankings which are obtained from original rankings by eliminating entities whose corresponding triples (except the target triple) were included in the training dataset. "Filtered" dMAP (fdMAP) is the mean of the dAP of "filtered" rankings for each answer of queries.

We refer to GPARs considered as entity ranking systems with these dMAPs or fdMAPs as the *graph pattern entity ranking model* (GRank).

By using a graph pattern to rank entities, GRank is able to properly estimate GPARs where the number of matches is important as

	WN18				FB15k				WN18RR				FB15k-237			
	MRR	HITS@			MRR	HITS@			MRR	HITS@			MRR	HITS@		
Model		1	3	10		1	3	10		1	3	10		1	3	10
TransE	0.397	0.040	0.745	0.923	0.414	0.247	0.534	0.688	0.182	0.027	0.295	0.444	0.257	0.174	0.284	0.420
TorusE	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832	–	–	–	–	–	–	–	–
RESCAL	0.890	0.842	0.904	0.928	0.354	0.235	0.409	0.587	–	–	–	–	–	–	–	–
DistMult	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
ComplEx	0.941	0.936	0.945	0.947	0.692	0.599	0.759	0.840	0.44	0.41	0.46	0.51	0.240	0.152	0.263	0.419
R-GCN	0.814	0.686	0.928	0.955	0.651	0.541	0.736	0.825	–	–	–	–	0.248	0.153	0.258	0.417
ConvE	0.942	0.935	0.947	0.955	0.745	0.670	0.801	0.873	0.46	0.39	0.43	0.48	0.316	**0.239**	0.350	**0.491**
PRA	0.458	0.422	–	0.481	0.336	0.303	–	0.392	–	–	–	–	–	–	–	–
Node+LinkFeat	0.940	–	–	0.943	0.822	–	–	0.870	–	–	–	–	0.272	–	–	0.414
GPro	**0.950**	**0.946**	**0.954**	**0.959**	0.793	0.759	0.810	0.858	0.467	0.430	**0.485**	**0.543**	0.229	0.163	0.250	0.360
GRank (dMAP)	**0.950**	**0.946**	0.953	0.957	0.841	0.814	0.855	0.890	0.466	0.434	0.480	0.530	0.312	0.233	0.340	0.473
GRank(fdMAP)	**0.950**	**0.946**	**0.954**	0.958	**0.842**	**0.816**	**0.856**	**0.891**	**0.470**	**0.437**	0.482	0.539	**0.322**	**0.239**	**0.352**	0.489

Table 2: Mean Reciprocal Rank (MRR) and HITS@n scores obtained for the link prediction tasks on the WN18, FB15k, WN18RR, and FB15k-237 datasets. The highest result for each column is shown in bold. The results of TransE and TorusE were reported by Ebisu and Ichise (2018), the results of RESCAL were reported by Nickel et al. (2016), the results of DistMult and ComplEx were reported by Trouillon et al. (2016), the results of R-GCN and ConvE were reported by Dettmers et al. (2018), the results of PRA were reported by Liu et al. (2016), and the results of Node+LinkFeat were reported by Toutanova and Chen (2015).

shown in Section 4.2, unlike GPro. For example, $dMAP_{tail}(AR_{1,(x,y)}) = 1$, is the maximum value, while $dMAP_{tail}(AR_{2,(x,y)}) = 1/2$ in Figure 1. Hence, GRank can answer the query (Team C, located_in, ?) by applying $AR_{1,(x,y)}$.

6 Experiments

Our proposed models, GPro (Section 4.2) and GRank (Section 5), are evaluated through link prediction tasks and compared with other state-of-the-art link prediction models.

6.1 Datasets

Experiments were conducted on four benchmark datasets: WN18, FB15k (Bordes et al., 2013), WN18RR (Dettmers et al., 2018), and FB15k-237 (Toutanova and Chen, 2015) (details of these datasets are provided in Table 1). These datasets have been widely used in previous studies for evaluating model performance in link prediction tasks.

WN18 and FB15k were extracted from the real knowledge graphs WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), respectively. WordNet is a well-known human-curated lexical database, and hence, WN18 is an easy benchmark of link prediction because it is well constructed and there are few missing or wrong facts. Therefore, link prediction models should perform well on WN18. Freebase is a huge knowledge graph of general facts and there are many missing facts. It is known that WN18 and FB15k have redundancy in the form of reverse relations. For this reason, when WN18RR and FB15k-237 are extracted from WN18 and FB15k, the inverse relations of other relations are removed.

6.2 Evaluation Protocol

We conducted the link prediction task following the same approach reported in (Bordes et al., 2013) to evaluate our models qualitatively and quantitatively. For each test triple (h_t, r_t, t_t) in a dataset, two queries, $(h_t, r_t, ?)$ and $(?, r_t, t_t)$, were constructed in the same way as in Section 3.2. Then, we obtained the rankings of entities for each query from each model as outlined in the following paragraphs. The rankings were ”filtered” by eliminating entities whose corresponding triples (except the target test triple) were included in the training, validation, or test dataset. The obtained rankings were scored by their mean reciprocal rank (MRR) and HITS@n, where MRR is the mean of the inverse of the ranks of corresponding entities and HITS@n is the proportion of test queries whose corresponding entities are ranked in the top n of the obtained rankings.

Next, we describe how to obtain rankings from models. We restricted antecedent graph patterns of GPARs to *connected* and *closed* (Galárraga et al., 2013, 2015) patterns whose size $|GP_{(x,y)}|$ was less than or equal to L to restrict the search space. A connected and closed patterns is a pattern connecting x and y without branches, as shown in Figure 2. L was chosen for each model among $\{1, 2, 3\}$ by MRR from the validation triples of each dataset. It took about four days to evaluate all candidate GPARs for GRank with dMAPs in FB15k using an *Intel Xeon Gold 6154* (3.00 GHz, 18 cores).

We now explain how we obtained the rankings for queries with missing heads. For each relation r, we chose the top 1,000 GPARs in descending order of the standard confidence, the dMAP, or the fdMAP to predict the heads. Let $GP_{i,(x,y)} \Rightarrow r_{(x,y)}$ be the obtained GPAR, where i denotes the rank. We defined the ordering for two entities for query $(?, r_t, t_t)$ as follows: for entities e_1 and e_2, we define $e_1 > e_2$ if there exists i' for which $score(GP_{i,(x,y)}, (e_1, t_t)) = score(GP_{i,(x,y)}, (e_2, t_t))$ for $i > i'$ and $score(GP_{i',(x,y)}, (e_1, t_t)) > score(GP_{i',(x,y)}, (e_2, t_t))$. We obtained the entity rankings with this ordering for each query. Rankings for queries with missing tails were obtained in the same way.

6.3 Results

The results of the link prediction tasks for our proposed models, GPro, GRank with dMAP, and GRank with fdMAP, are shown in Tables 2, where the results reported in previous studies are included for comparison.

In Table 2, the first seven models are knowledge graph embedding models and the following two models are observed feature models. Table 2 shows the effectiveness of the Node+LinkFeat model (Toutanova and Chen, 2015), although this model is very simple (high MRRs imply that the model also has high HITS@1s or HITS@3s). The Node+LinkFeat model performed well on WN18 and FB15k because these datasets often contain the reverse relations of other relations. In other words, it shows that knowledge graph embedding models failed to capture this redundancy. On the other hand, our proposed models, GPro and GRank, generally yield better results than the knowledge graph embedding models and results which are better than or comparable to Node+LinkFeat, which means that our models can also handle such redundancy. In particular, GRank with dMAP and fdMAP yielded the best results on FB15k. This indicates that taking the multiplicity of matchings and deeper information into account is important for knowledge graphs such as FreeBase that contain miscellaneous relations and are not well curated like WordNet. As a result, GRank performed well.

Table 2 also shows GPro and GRank yield better results for the WN18RR dataset than the other models. For FB15k-237, the performance of Node+LinkFeat is comparable with most of the other more sophisticated knowledge graph models and GPro does not yield good results because FB15k-237 has less redundancy. GRank also performs better than the most other models for the FB15k-237 dataset for the same reason as the FB15k dataset. However, our models do not utilize the information related to the co-occurrence of entities and relations in triples (node features (Toutanova and Chen, 2015)), while ConvE, Node+LinkFeat, and other models do. We also limited the size and the shapes of graph patterns because of the calculation time; we will address these and improve our models further in our future work.

Quality of Obtained Paths The examples of antecedent patterns ranked high by GRank with $dMAP_{tail}$ for FB15k are shown in Figure 3. The patterns shown for predicting the sibling relation are all correct as the antecedents of GPARs; however, the MAP of $GP'_{2,(x,y)}$ and $GP'_{3,(x,y)}$ are low. The reason for this is that $GP'_{2,(x,y)}$ works when an individual has more than two siblings. The MAP of $GP'_{3,(x,y)}$ is low because individual's parents are often missing in FB15k. However, they are still ranked higher than other patterns.

The produces_film relation is the inverse relation of the exective_produced_by relation in FB15k. Such patterns are very helpful when performing link prediction tasks, and GRank is able to find them. However, the MAP is not as high because of missing facts. GRank is able to use majority rules such as $GP'_{5,(x,y)} \Rightarrow$ film_produced_by(x, y) instead in such cases. This rule can be interpreted as stating that a particular film was likely to have been produced by a person who produced many films in the same production company.

Output triples of GRank (and GPAR-based models) are described by antecedent patterns unlike knowledge graph embedding models as shown here.

7 Conclusions

In this paper, we first defined GPARs for a knowledge graph and the standard confidence measures of GPARs for link prediction. Then, we pointed out the problems with the standard confidence measures and we introduced a new perspective using GPARs to rank entities to overcome these problems. We also proposed distributed rank-

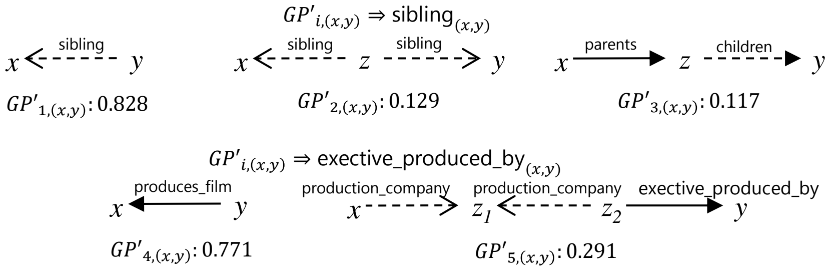


Figure 3: Examples of antecedent patterns for $dMAP_{tail}$ which were given high ranks by GRank for the FB15k dataset.

ings for situations where multiple entities have the same scores and defined metrics for them. This idea led us to propose the GRank model. GRank is easy to interpret because outputs are described by GPARs, unlike knowledge graph embedding models, and so efficient that it outperformed the state-of-the-art knowledge graph embedding models in link prediction tasks.

In future work, we will extend GRank to use more complex patterns. We considered only antecedent graph patterns whose sizes were less than or equal to 3.If we allow antecedent graph patterns to have larger sizes, then we may find more useful GPARs. We also restricted graph patterns to contain only variables and not constants. Hence, we did not use all of the available information contained in the knowledge graph. We believe that using such complex graph patterns will improve GRank further.

Acknowledgements

This work was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

We would like to thank Patrik Schneider for helpful writing advice.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 722–735.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232.

Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. 2015. Association rules with graph patterns. *VLDB J.*, 8(12):1502–1513.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference*, pages 413–422.

Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. 2012. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, SWIM '12, pages 1–7.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the Fifth International Conference on Learning Representations*.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random walk inference and learning in A large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

Qiao Liu, Liuyi Jiang, Minghao Han, Yao Liu, and Zhiguang Qin. 2016. Hierarchical random walk inference in knowledge graphs. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–454.

George A. Miller. 1995. Wordnet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the Third International Conference on Learning Representations*.

Adversarial Training for Weakly Supervised Event Detection

Xiaozhi Wang[1*], **Xu Han**[1*], **Zhiyuan Liu**[1†], **Maosong Sun**[1], **Peng Li**[2]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
[2]Pattern Recognition Center, WeChat, Tencent Inc, China
`{xz-wang16, hanxu17}@mails.tsinghua.edu.cn`
`{liuzy, sms}@tsinghua.edu.cn`
`patrickpli@tencent.com`

Abstract

Modern weakly supervised methods for event detection (ED) avoid time-consuming human annotation and achieve promising results by learning from auto-labeled data. However, these methods typically rely on sophisticated pre-defined rules as well as existing instances in knowledge bases for automatic annotation and thus suffer from low coverage, topic bias, and data noise. To address these issues, we build a large event-related candidate set with good coverage and then apply an adversarial training mechanism to iteratively identify those informative instances from the candidate set and filter out those noisy ones. The experiments on two real-world datasets show that our candidate selection and adversarial training can cooperate together to obtain more diverse and accurate training data for ED, and significantly outperform the state-of-the-art methods in various weakly supervised scenarios. The datasets and source code can be obtained from `https://github.com/thunlp/Adv-ED`.

1 Introduction

Event detection (ED) aims at detecting event triggers, which are often words or phrases evoking events in instances, and then identifying their specific event types. For example, we can extract the trigger "`married`" of the event "`Marry`" from the text "Mark Twain and Olivia Langdon `married` in 1870". Detecting and identifying events is an important subtask of event extraction and also beneficial for various downstream NLP applications, such as question answering (Yang et al., 2003), information retrieval (Basile et al., 2014), and reading comprehension (Cheng and Erk, 2018). Hence, many efforts have been devoted to detecting event triggers and types.

Most prior methods for ED are based on feature engineering, such as the token-level features (Ahn, 2006; Ji and Grishman, 2008) and the structured features (Li et al., 2013; Araki and Mitamura, 2015). As the rapid development of neural networks, various neural models have been proposed to directly embed textual semantic information into a low-dimensional space and then detect event triggers based on those feature vectors (Chen et al., 2015; Nguyen and Grishman, 2015; Nguyen et al., 2016; Ghaeini et al., 2016; Feng et al., 2016). These methods follow a supervised learning approach to train models on human-annotated data, and their requirement of human-annotated data is a bottleneck in practice. Considering weak supervision is widely adopted to take full advantages of large-scale raw data, especially some specific work for information extraction (Mintz et al., 2009; Riedel et al., 2010; Zeng et al., 2015; Cao et al., 2018), weak supervision has been explored to automatically label training data for ED (Chen et al., 2017; Zeng et al., 2018; Yang et al., 2018; Araki and Mitamura, 2018). Compared with those supervised ED methods, the weakly supervised methods can be generalized to real-world ED applications efficiently without intensive labor.

Although promising results have been achieved by these weakly supervised methods, there are still some severe problems for these weakly supervised ED models: (1) Weakly supervised methods naturally suffer from the inevitable noise in data. (2) Current weakly supervised ED models adopt sophisticated pre-defined rules and incomplete knowledge bases to automatically obtain data, which results in the auto-labeled data with low coverage and topic bias.

In order to construct a large-scale dataset with better coverage and reduce topic bias, we avoid adopting sophisticated pre-defined rules and heavy toolkits for semantic component analysis. Instead,

---

[*] indicates equal contribution

[†] Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

Proceedings of NAACL-HLT 2019, pages 998–1008
Minneapolis, Minnesota, June 2 - June 7, 2019.

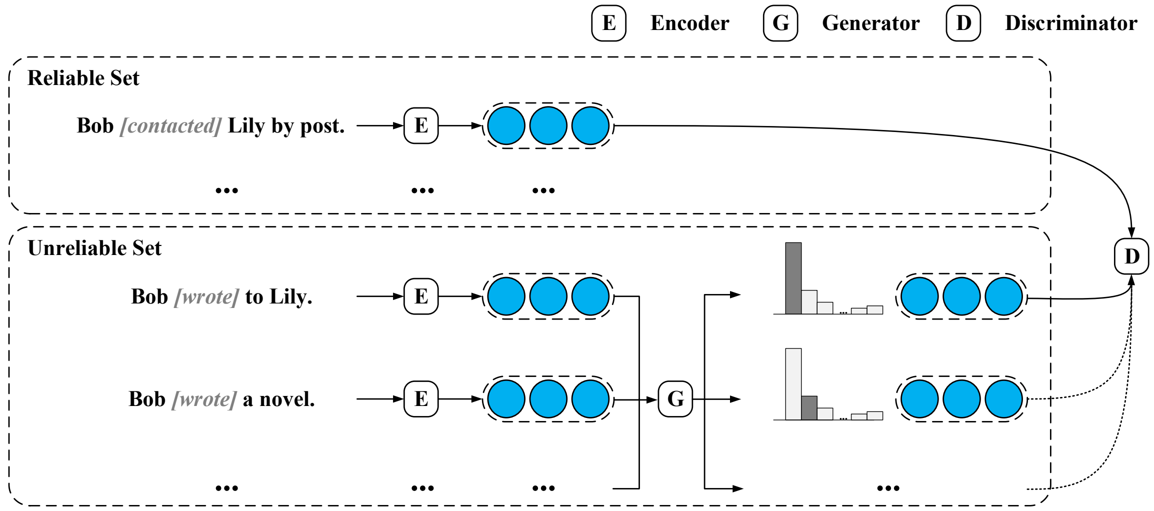


Figure 1: The overall architecture of adversarial training method for ED. The event type is `Contact`.

we propose a simple trigger-based latent instance discovery strategy, by applying an assumption that if a given word [1] serves as the trigger in a known event instance, all instances mentioning this word may also express an event. As compared with the sophisticated rules, this strategy is less restrictive in the correlation among words, triggers and event types. Hence, our strategy can obtain a candidate set covering more topics and instances without any manual design.

We further propose an adversarial training mechanism like Goodfellow et al. (2014); Radford et al. (2016), which can not only distill those informative instances from the candidate set but also improve the performance of ED model on a noisy scenario such as distant supervision. As shown in Figure 1, we split the dataset into a reliable set and an unreliable set respectively, and design a discriminator and a generator. The discriminator is applied to judge whether a given instance is informative and annotated correctly, and the generator is used to select the most confusing instances from raw data to fool the discriminator. The discriminator is trained with the reliable data as positive instances and the data selected by the generator as negative ones. Meanwhile, the generator is trained to select data to fool the discriminator. During the training process, the generator can provide large amounts of latent noisy data to enhance the discriminator, and the discriminator can influence the generator to select those more informative data. Since noisy data makes no effect on optimizing both the generator and the discriminator, when the generator and the discriminator reach a balance, the discriminator can boost resistance to noise and better categorize events, and the generator can effectively select informative instances to the discriminator.

We conduct experiments on both semi-supervised and distantly supervised scenarios. The experimental results demonstrate that our trigger-based latent instance discovery strategy and adversarial training method can cooperate to obtain more diverse and accurate training data as well as reduce the side effect of the noise problem, and thus significantly outperform the state-of-the-art ED models.

2 Related Work

ED has attracted wide attention recently. Traditional feature-based methods (Ahn, 2006; Ji and Grishman, 2008; Gupta and Ji, 2009; Riedel et al., 2010; Hong et al., 2011; McClosky et al., 2011; Huang and Riloff, 2012a,b; Araki and Mitamura, 2015; Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2016b) rely on manually designed features to detect the event triggers and event types. With the development of neural networks, various neural methods have also been proposed (Chen et al., 2015; Nguyen and Grishman, 2015; Nguyen et al., 2016; Duan et al., 2017; Nguyen et al., 2016; Ghaeini et al., 2016; Lin et al., 2018).

Furthermore, some efforts have been made to improve the performance of ED systems with external knowledge (Liu et al., 2016a, 2017), contextual information (Liu et al., 2018b), document-level information (Duan et al., 2017; Zhao et al.,

---

[1] We treat phrases as words in this paper.

2018) and multimodal integration (Zhang et al., 2017). Some advanced architectures have also been applied, such as attention mechanism (Liu et al., 2017, 2018a), graph convolutional networks (Nguyen and Grishman, 2018) and generative adversarial networks (Hong et al., 2018).

All the supervised methods above rely on human-annotated data, and the data is often restricted to a small scale due to the expensive human annotation. Hence, unsupervised methods (Huang et al., 2016; Yuan et al., 2018) and various weakly supervised methods on ED are proposed. Muis et al. (2018) adopt distant supervision to create training instances for low-resource language. Araki and Mitamura (2018) adopt WordNet (Miller et al., 1990) and rule-based methods to generate open-domain data without event-type labels. Chen et al. (2017) and Zeng et al. (2018) use distant supervision to generate large-scale data from existing structured event knowledge in knowledge bases. Liao and Grishman (2010a), Huang and Riloff (2012a) and Ferguson et al. (2018) conduct semi-supervised ED with bootstrapping. Nevertheless, due to the low coverage of existing knowledge bases as well as lack of advanced denoising mechanism, those weakly supervised methods still suffer from the problem of low coverage and noisy data.

Inspired by Szegedy et al. (2013) and Goodfellow et al. (2014), adversarial training has been explored for several NLP applications recently to resist noise, such as text classification (Miyato et al., 2016) and text generation (Xie et al., 2017; Chen et al., 2018). Adversarial training has also been adopted for information extraction (Wu et al., 2017; Hong et al., 2018; Qin et al., 2018; Wang et al., 2018; Han et al., 2018). These adversarial information extraction methods either generate adversarial instances by adding simple noise perturbation to embeddings (Wu et al., 2017; Hong et al., 2018), or mainly adopt models to denoise data and neglect to discover more training instances from raw data (Qin et al., 2018; Han et al., 2018). Compared with these methods, our adversarial method samples adversarial examples from the real-world data rather than generating pseudo noisy perturbations. Furthermore, our method not only denoises auto-labeled data but also labels unlabeled instances to extend datasets for higher coverage. Hence, our method can effectively alleviate low coverage, topic bias, and noise problem in ED.

3 Methodology

In this section, we introduce the overall framework of our proposed models for weakly supervised ED.

3.1 Framework

As shown in Figure 1, the overall framework consists of three modules, including instance encoder, adversarial training strategy, and their adaption for various weakly supervised ED scenarios.

The instance encoder is applied to encode the instances into its corresponding embeddings to provide semantic features for the other modules of our models. Given an instance $x = \{w_1, \ldots, t, \ldots, w_n\}$ consisting of n words and its candidate trigger t, we adopt several effective neural models to represent the semantic features of the instance x with the embedding $\mathbf{x}$. Details of the instance encoder are shown in Section 3.2.

After representing instances into their embeddings by the instance encoder, an adversarial training strategy is applied, which aims at highlighting those informative instances and filtering out those noisy instances from a large-scale unreliable dataset $\mathcal{U}$ under the guidance of another reliable dataset $\mathcal{R}$. The adversarial training strategy is the core module of our framework, and we will introduce its details in Section 3.3.

Each instance x in $\mathcal{U}$ and $\mathcal{R}$ will be labeled with a trigger word t and an event type $e \in \mathcal{E}$. If an instance does not have a trigger and cannot express any definite events, it will be labeled with a special event `NA`, which indicates that the event of this instance is not available. Before applying adversarial training, $\mathcal{U}$ and $\mathcal{R}$ are automatically labeled. The details of splitting and automatically labeling $\mathcal{U}$ and $\mathcal{R}$ for various weakly supervised ED scenarios, as well as utilizing adversarial training strategy to extend datasets, will be introduced in Section 3.4.

3.2 Instance Encoder

In this paper, we select CNN (Chen et al., 2015) and BERT (Devlin et al., 2018) as representative encoders to encode the given instances.

CNN After representing all words in the instance x into their input embeddings, including both word embeddings and position embeddings which encode the relative position to candidate triggers, CNN slides a convolution kernel over the input embeddings to get hidden embeddings as follows,

$$\{\mathbf{h}_1, \dots, \mathbf{h}_n\} = \text{CNN}\big(w_1, \dots, t, \dots, w_n\big). \quad (1)$$

BERT Similar to CNN, after summing word piece (Wu et al., 2016), segment and position embeddings of all words in the instance x as input embeddings, BERT adopt a multi-layer bidirectional transformer encoder (Vaswani et al., 2017) to get hidden embeddings as follows,

$$\{\mathbf{h}_1, \dots, \mathbf{h}_n\} = \text{BERT}\big(w_1, \dots, t, \dots, w_n\big). \quad (2)$$

Because the candidate trigger t splits the instance x into two parts, we follow Chen et al. (2015) to adopt a dynamic multi-pooling operation over the hidden embeddings to achieve the instance embedding $\mathbf{x}$,

$$\begin{aligned} [\overleftarrow{\mathbf{x}}]_j &= \max\{[\mathbf{h}_1]_j, \dots, [\mathbf{h}_i]_j\}, \\ [\overrightarrow{\mathbf{x}}]_j &= \max\{[\mathbf{h}_{i+1}]_j, \dots, [\mathbf{h}_n]_j\}, \\ \mathbf{x} &= [\overleftarrow{\mathbf{x}}; \overrightarrow{\mathbf{x}}], \end{aligned} \quad (3)$$

where $[\cdot]_j$ is the j-th value of a vector and i is the position of the trigger t. As CNN and BERT adopt a dynamic multi-pooling operation, we name them "**DMCNN**" and "**DMBERT**" in this paper.

3.3 Adversarial Training

As shown in Figure 1, the overall framework of our adversarial strategy consists of a discriminator and a generator. The discriminator is adopted to detect event triggers and identify event types for each instance in datasets. When given a noisy instance, the discriminator is also expected to resist noise and explicitly point out that there are no triggers and events. The generator is used to select instances from the unreliable dataset $\mathcal{U}$ to confuse the discriminator as much as possible.

Each instance $x \in \mathcal{R}$ is assumed to explicitly express its labeled trigger t and event type e. In contrast, each instance $x \in \mathcal{U}$ is assumed to be untrustworthy during the adversarial training, i.e., there is a certain probability that it is labeled incorrectly. Hence, we design the discriminator to judge whether a given instance can expose its labeled event type, which aims at maximizing the conditional probability $P(e|x,t), x \in \mathcal{R}$ and $1 - P(e|x,t), x \in \mathcal{U}$. The generator is trained to select the most confusing instances from $\mathcal{U}$ to fool the discriminator, i.e., selecting the instances by $P(e|x,t), x \in \mathcal{U}$. The training process is an adversarial min-max game as follows,

$$\begin{aligned} \phi_D &= \max\big(E_{x\sim P_{\mathcal{R}}}\big[\log\big(P(e|x,t)\big)\big] \\ &\quad + E_{x\sim P_{\mathcal{U}}}\big[\log\big(1 - P(e|x,t)\big)\big]\big), \\ \phi_G &= \max E_{x\sim P_{\mathcal{U}}}\big[\log\big(P(e|x,t)\big)\big], \end{aligned} \quad (4)$$

where $P_{\mathcal{R}}$ is the reliable data distribution, and the generator samples adversarial examples from the unreliable data according to the probability distribution $P_{\mathcal{U}}$. Although ϕ_D and ϕ_G are conflicting, noisy data in $\mathcal{U}$ has the side effect for both ϕ_D and ϕ_G. Hence, when the generator and the discriminator reaching a balance after sufficient training, the generator tends to select those informative instances with a higher probability compared with those noisy ones, and the discriminator boosts resistance to noise and can better categorize events.

Discriminator

Given an instance x and its labeled trigger t and event type e, the discriminator is responsible for judging whether the given instance exposes its labeled trigger and event type. After representing the instance x with its embedding $\mathbf{x}$, we implement the discriminator as follows,

$$\begin{aligned} D(e|x,t) &= \mathbf{e} \cdot \mathbf{x}, \\ P(e|x,t) &= \frac{\exp\big(D(e|x,t)\big)}{\sum_{\hat{e}\in\mathcal{E}} \exp\big(D(\hat{e}|x,t)\big)}, \end{aligned} \quad (5)$$

where $\mathbf{e}$ is the embedding of the event type $e \in \mathcal{E}$.

An optimized discriminator will assign high scores to those instances in $\mathcal{R}$, and meanwhile distrust those instances and their labels in $\mathcal{U}$. Hence, in practice, we formalize the loss function to optimize the discriminator as follows,

$$\begin{aligned} \mathcal{L}_D = &-\sum_{x\in\mathcal{R}} \frac{1}{|\mathcal{R}|} \log\big(P(e|x,t)\big) \\ &-\sum_{x\in\mathcal{U}} P_{\mathcal{U}}(x) \log\big(1 - P(e|x,t)\big). \end{aligned} \quad (6)$$

When optimizing the discriminator, we regard the component of the encoder and $D(e|x,t)$ as parameters for updating. This loss function $\mathcal{L}_D$ is corresponding to ϕ_D in Eq. (4).

Generator

The generator aims at selecting the most confusing instances from $\mathcal{U}$ to cheat the discriminator. We design the generator to optimize the probability distribution $P_{\mathcal{U}}$ to select instances. The generator computes confusing scores for all instances in

$\mathcal{U}$ to evaluate their perplexity and further computes the confusing probability $P_{\mathcal{U}}$ as follows,

$$\begin{aligned} f(x) &= \mathbf{W} \cdot \mathbf{x} + \mathbf{b}, \\ P_{\mathcal{U}}(x) &= \frac{\exp\big(f(x)\big)}{\sum_{\hat{x} \in \mathcal{U}} \exp\big(f(\hat{x})\big)}. \end{aligned} \tag{7}$$

where $\mathbf{x}$ is the embedding of the instance x computed by the encoder. $\mathbf{W}$ and $\mathbf{b}$ are parameters for a separating hyperplane.

We regard that the higher scores computed by the discriminator the instances have, the more confusing the instances are, because they are more likely to fool the discriminator to make a wrong decision. We expect that an optimized generator pays more attention to those most confusing instances. Hence, given an instance $x \in \mathcal{U}$ and its unreliable-labeled trigger t and event type e, we formalize the loss function to optimize the generator as follows,

$$\mathcal{L}_G = -\sum_{x \in \mathcal{U}} P_{\mathcal{U}}(x) \log\big(P(e|x,t)\big), \tag{8}$$

where $P(e|x,t)$ is computed by the discriminator. When optimizing the generator, we regard the component to compute $P_{\mathcal{U}}(x)$ as parameters for updating. This loss function $\mathcal{L}_G$ is corresponding to ϕ_G in Eq. (4).

There may be some instances in $\mathcal{U}$ labeled `NA` and these instances are always wrongly predicted into some other events. Thus we specifically use the average scores over all feasible events to replace their $P(e|x,t)$ in Eq. (8) as follows,

$$P(\text{NA}|x,t) = \frac{1}{|\mathcal{E}| - 1} \sum_{e \in \mathcal{E}, e \neq \text{NA}} P(e|x,t), \tag{9}$$

where $\mathcal{E}$ indicates the set of event types.

Training and Implementation Details

Because there may be large amounts of instances in $\mathcal{R}$ and $\mathcal{U}$, directly computing $\mathcal{L}_D$ and $\mathcal{L}_G$ is time-consuming, and frequently traversing the whole dataset of $\mathcal{R}$ and $\mathcal{U}$ also accordingly becomes difficult. For improving training efficiency, we sample subsets of $\mathcal{R}$ and $\mathcal{U}$ to approximate the essential probability distribution, and formalize a new loss function for optimization,

$$\begin{aligned} \tilde{\mathcal{L}}_D &= -\sum_{x \in \tilde{\mathcal{R}}} \frac{1}{|\tilde{\mathcal{R}}|} \log\big(P(e|x,t)\big) \\ &\quad - \sum_{x \in \tilde{\mathcal{U}}} P_{\tilde{\mathcal{U}}}(x) \log\big(1 - P(e|x,t)\big), \\ \tilde{\mathcal{L}}_G &= -\sum_{x \in \tilde{\mathcal{U}}} P_{\tilde{\mathcal{U}}}(x) \log\big(P(e|x,t)\big), \end{aligned} \tag{10}$$

where $\tilde{\mathcal{R}}$ and $\tilde{\mathcal{U}}$ are the subsets sampled from $\mathcal{R}$ and $\mathcal{U}$, and $P_{\tilde{\mathcal{U}}}$ is the approximation to Eq. (7),

$$P_{\tilde{\mathcal{U}}}(x) = \frac{\exp\big(f(x)^{\alpha}\big)}{\sum_{\hat{x} \in \tilde{\mathcal{U}}} \exp\big(f(\hat{x})^{\alpha}\big)}. \tag{11}$$

α is a hyperparameter that controls the sharpness of the probability distribution to avoid the weights concentrating on some specific instances. Finally, the overall optimization function is,

$$\mathcal{L} = \tilde{\mathcal{L}}_D + \lambda \tilde{\mathcal{L}}_G, \tag{12}$$

where λ is a harmonic factor. In practice, $\tilde{\mathcal{L}}_D$ and $\tilde{\mathcal{L}}_G$ in adversarial training are optimized alternately, and λ is also integrated into the learning rate of $\tilde{\mathcal{L}}_G$ to avoid adjusting λ additionally.

3.4 Adaption for Weakly Supervised Scenarios

In this section, we introduce the adaption of adversarial training strategy for various weakly supervised ED scenarios (semi-supervised scenarios and distantly supervised scenarios), as well as the method to automatically label and split the reliable set and unreliable set used for adversarial training.

Trigger-based Latent Instance Discovery

To utilize unlabeled data, we propose a simple trigger-based latent instance discovery strategy, which can automatically label trigger words and event types for raw data. The trigger-based strategy is based on a heuristic assumption that if a given word serves as the trigger in a known instance, all other instances mentioning this word in raw data are latent instances and may also express an event. For example, the word "`married`" serves as the trigger in the instance "Mark Twain and Olivia Langdon `married` in 1870" to expose the event "`Marry`", and then all instances in unlabeled data containing the word "`married`" will be picked up and added into a latent instance candidate set. As compared with the sophisticated rules used in existing weakly supervised ED models, our trigger-based latent instance discovery is simple, without the need of considering the correlation among words, triggers, and event types. Because our strategy is less restrictive, it is effective and efficient to obtain a large-scale candidate set without any special manual design. Meanwhile, the candidate set can cover much more instances and topics than the existing strategies.

Semi-supervised Scenarios

When adapting our adversarial training strategy for semi-supervised scenarios, we first use the small-scale labeled data to pretrain the encoder and discriminator to let them gain the ability to detect event triggers and identify event types to a certain extent. Then, we construct a large-scale latent candidate set based on our instance discovery strategy with the trigger words in the labeled data as heuristic seeds. We use the pretrained encoder and discriminator to automatically label triggers and event types for all instances in the candidate set to build noisy large-scale data. With the small-scale labeled data as the reliable set $\mathcal{R}$ and the large-scale auto-labeled data as the unreliable set $\mathcal{U}$, we can optimize the encoder, discriminator, and generator together to carry out adversarial training. During the adversarial training, when the discriminator and generator reach a balance after certain training epochs, all instances from the unreliable set $\mathcal{U}$ recommended by the generator and regarded as being labeled correctly by the discriminator will be adjusted from $\mathcal{U}$ to $\mathcal{R}$. Conducting adversarial training iteratively can identify informative instances and filter out noisy instances in $\mathcal{U}$, and accomplish utilizing large-scale unlabeled data to enrich small-scale labeled data.

Distantly Supervised Scenarios

The adaption for distantly supervised scenarios is similar to the adaption for semi-supervised scenarios. We first use the whole auto-labeled data to pretrain the encoder and discriminator. Then, the encoder and discriminator are used to compute confident scores for all instances in the auto-labeled set. By setting a particular threshold, we can split the whole auto-labeled set into two parts. The instances with scores higher than the threshold will be added into the reliable set $\mathcal{R}$, and the other instances with lower scores will be added into the unreliable set $\mathcal{U}$. After the whole auto-labeled set being split into $\mathcal{R}$ and $\mathcal{U}$, we can conduct adversarial training to reduce the side effect of those noise in $\mathcal{U}$ and enhance the discriminator for better identifying events. Intuitively, the reliable set $\mathcal{R}$ isolated from the auto-labeled set can be used as seeds to utilize more raw data in a similar way applied in semi-supervised scenarios.

4 Experiments

We evaluate our models on both semi-supervised and distantly supervised scenarios. Before introducing the detailed experimental settings and results, we list the hyperparameters first.

4.1 Hyperparameter Settings

For DMCNN, following the settings of previous work, we use the pre-trained word embeddings learned by Skip-Gram (Mikolov et al., 2013) as the initial word embeddings. We implement DMCNN by ourselves and follow the same hyperparameters used in Chen et al. (2015) for fair comparisons. For DMBERT, we follow the same hyperparameters used for $\text{BERT}_{\text{BASE}}$ in Devlin et al. (2018) and apply the pre-trained model[2] to initialize the parameters. We list the essential hyperparameters of the discriminator and the generator for adversarial training in Table 1.

Dropout Probability p	5×10^{-1}
Learning Rate α_{gc} for the generators (DMCNN)	5×10^{-3}
Learning Rate α_{dc} for the discriminators (DMCNN)	2×10^{-2}
Learning Rate α_{gb} for the generators (DMBERT)	2×10^{-5}
Learning Rate α_{db} for the discriminators (DMBERT)	1×10^{-4}

Table 1: Hyperparameter settings.

4.2 Distantly Supervised Scenarios

Dataset and Evaluation

For distantly supervised scenarios, we utilize the distantly supervised dataset developed by Chen et al. (2017) with FreeBase (Bollacker et al., 2008). The dataset contains $142,611$ labeled instances and 21 event types. Following previous work (Mintz et al., 2009; Chen et al., 2017), we evaluate our adversarial training mechanism by held-out evaluation. We report the precision-recall curves of recall under 0.7 since we mainly focus on the performance of those top-ranked results. To give a complete view of the overall performance, we also report the area under the curve (AUC).

---

[2] `https://github.com/google-research/bert`

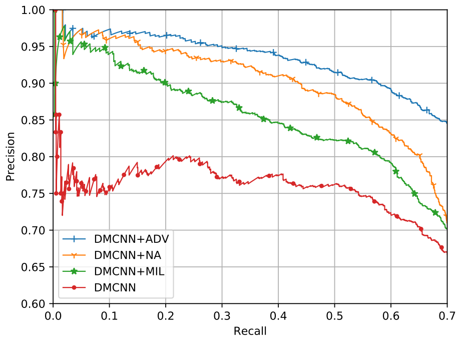


Figure 2: The aggregated precision-recall curves of DMCNN models.

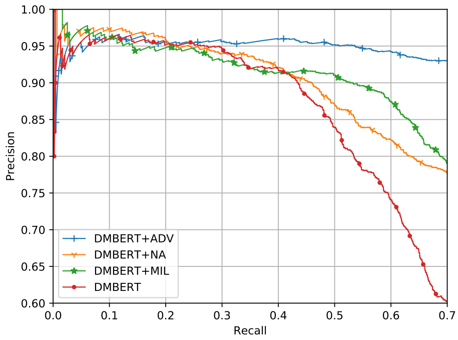


Figure 3: The aggregated precision-recall curves of DMBERT models.

To evaluate the effectiveness of our proposed models, we compare our adversarial training models (**DMCNN+ADV**, **DMBERT+ADV**) with various neural baselines, including: (1) **DMCNN** and **DMBERT** proposed in Chen et al. (2015) and this paper respectively, which are the basic models without any adaption to the noisy distant supervision. (2) **+MIL** models, which improve the basic models with multi-instance learning proposed in Chen et al. (2017) to alleviate the noise problem. (3) **+NA** models, which simply treat the instances in the unreliable set as negative instances with the label `NA`. This method could be regarded as a simplified version of our adversarial training to conduct ablation study. In this experiment, we separate the reliable and unreliable set by the confidence of the basic models following Section 3.4.

Overall Evaluation Results

The precision-recall curves of DMCNN models and DMBERT models are shown in Figure 2 and Figure 3, and the results of AUC are shown in Table 2. From the results, we can observe that: (1) BERT-based models significantly outperform the CNN-based models, which is due to the ability to capture contextual information as well as large-scale pre-training of BERT. And benefiting from the effective pre-trained parameters, the BERT-based models all have high precision when the recall is under 0.3. (2) The **+NA** models achieve similar performance with **+MIL** and even outperform them in low-recall range, but **+NA** models have the worst macro AUC. It indicates that the separation of reliable and unreliable set is effective but also have severe side effects, and our adversarial training method works well to overcome the side effect. (3) Our adversarial training method significantly outperforms all the baselines in every metric. This demonstrates the strong ability of our method to alleviate the noise problem on distantly supervised scenarios.

Method	AUC Micro	AUC Macro
DMCNN	67.6	38.7
DMCNN+MIL	75.7	43.3
DMCNN+NA	70.6	25.8
DMCNN+ADV	**85.5**	**50.7**
DMBERT	70.6	42.2
DMBERT+MIL	79.4	47.3
DMBERT+NA	74.0	38.6
DMBERT+ADV	**91.5**	**67.6**

Table 2: The AUC results (%) of various models.

4.3 Semi-supervised Scenarios

Dataset and Evaluation

For semi-supervised scenarios, we conduct experiments on a widely-used benchmark dataset ACE-2005 (Walker et al., 2006) containing 599 documents annotated with 8 types and 33 subtypes of events. Following the previous work (Liao and Grishman, 2010b; Li et al., 2013; Chen et al., 2015), we use the same test set containing 40 newswire documents, development set with 30 randomly selected documents and training set with the remaining 529 documents.

As described in Section 3.4, using existing triggers in ACE-2005 training set as heuristic seeds and our trigger-based latent instance discovery strategy, we construct a large-scale candidate set from the New York Times corpus (Sandhaus, 2008) and use our adversarial training strategy to filter out the noisy instances to build a new ACE-style dataset. We extend the ACE-2005 training set with the new dataset, and then test the models trained on the extended training set on the original test set. Our models trained on the original training set are named **DMCNN** and **DMBERT**, and our bootstrapped models trained on the extended dataset are named **DMCNN+Boot** and **DMBERT+Boot**.

We compare our bootstrapped models with various state-of-the-art methods on the ACE-2005 dataset, including: (1) The feature-based models. We select **Li's joint** (Li et al., 2013) as the representative, which achieves the best performance among feature-based models. (2) The vanilla neural network models, including the **DMCNN** (Chen et al., 2015) and **JRNN** (Nguyen et al., 2016). (3) The neural network models with external information, including **ANN-FN** (Liu et al., 2016a) leveraging the information of FrameNet (Baker et al., 1998), **DLRNN** (Duan et al., 2017) using document-level information, **GMLATT** (Liu et al., 2018a) utilizing multilingual attentions, and the bootstrapped model **DMCNN+Chen's DS** (Chen et al., 2017) trained with additional data distantly supervised by FreeBase (Bollacker et al., 2008). (4) The neural network models with advanced architecture, including: **Bi-LSTM+GAN** (Hong et al., 2018) utilizing GAN to conduct self-regulation, **GCN-ED** (Nguyen and Grishman, 2018) utilizing graph convolutional network to model dependency trees.

Overall Evaluation Results

The results are shown in Table 3. From the results, we have the following observations: (1) As compared with the basic **DMCNN** and **DMBERT**, the bootstrapped models achieve significant improvement (+1.7% and +0.5%). Furthermore, our **DMCNN+Boot** model achieves similar performance with the **ANN-FN** and **DLRNN** which design complex architectures to utilize the additional information. These results indicate that our methods can construct high-quality dataset without sophisticated rules and large-scale knowledge bases, and can effectively collect diverse instances which will benefit training models. (2) **DMBERT** and **DMBERT+Boot** achieve the best performance among all the models. This is benefiting from the effective architecture and the large-scale pre-training information of BERT, as well as the dynamic multi-pooling mechanism for ED. Our methods augment the training data to further enhance BERT, which achieve better performance and demonstrate the effectiveness of our models.

Method	Trigger Identification +Classification		
	P	R	F1
Li's Joint (Li et al., 2013)	73.7	62.3	67.5
DMCNN (Chen et al., 2015)	75.6	63.6	69.1
JRNN (Nguyen et al., 2016)	66.0	**73.0**	69.3
ANN-FN (Liu et al., 2016a)	77.6	65.2	70.7
DLRNN (Duan et al., 2017)	77.2	64.9	70.5
GMLATT (Liu et al., 2018a)	78.9	66.9	72.4
DMCNN+Chen's DS (Chen et al., 2017)	75.7	66.0	70.5
Bi-LSTM+GAN (Hong et al., 2018)	71.3	74.7	73.0
GCN-ED (Nguyen and Grishman, 2018)	**77.9**	68.8	73.1
DMBERT	77.6	71.8	74.6
DMCNN+Boot	77.7	65.1	70.8
DMBERT+Boot	**77.9**	72.5	**75.1**

Table 3: The overall performance (%) of different models on ACE-2005.

Method	Average Precision	Fleiss's Kappa
Chen et al. (2017)	88.9	–
Zeng et al. (2018)	91.0	–
Our First Iteration	91.7	61.3
Our Second Iteration	87.5	52.0

Table 4: The human evaluation results (%) of auto-labeled data in different iterations.

Manual Evaluation

To perform a fine-grained evaluation for the quality of the dataset constructed with our trigger-based instance discovery strategy and adversarial training strategy, we manually evaluate the precision of the constructed dataset. To be specific, we randomly select 150 instances from the newly constructed dataset and recruit four well-trained annotators to annotate the instances independently. We ask the annotators to label an instance as correct if and only if the trigger and event-type are both correct. We use the Fleiss' kappa (Fleiss, 1971) to measure the annotation consistency among these annotators. The results of the data distilled in different iterations during adversarial training are shown in Table 4. From the results, we can observe that the precision of the dataset constructed with our models is comparable to existing distant supervision methods (Chen et al., 2017; Zeng et al., 2018) using sophisticated human-designed rules and knowledge bases, and even outperforms them in the first iteration. It indicates that our models can distill informative instances with high precision.

Case Study

To further show the effectiveness of our models to improve the coverage of the dataset, we give an example in Table 5. The instance in the "In ACE-2005" row is a typical instance of the `Sue`

	Event-Type: `Justice` Subtype: `Sue`
In ACE-2005	Dell **sued** for ”bait and switch” and false promises.
Discovered	1. The lawyers for the four former state officials who have been **sued** told the jurors . . . 2. But **litigation** held up the project until . . . .

Table 5: The examples with highlighting triggers.

events, and the two instances in the “Discovered” row are sampled from the dataset constructed with our methods. In the “Discovered” row, the first instance is with an existing trigger in ACE-2005 but different in syntax, and the second instance is with a newly discovered trigger which is not contained in ACE-2005. In our extended dataset, there are 1.2% of the triggers are newly discovered. This demonstrates that our methods can not only find new instances from the unlabeled data which is similar to those instances in the labeled data, but also discover new triggers and extend the coverage of datasets substantially.

5 Conclusion and Future Work

In this paper, we take advantages of adversarial training and propose an effective method for weakly supervised ED. To be specific, our method is able to denoise and enhance distantly supervised ED models, as well as automatically construct more diverse and accurate training data for semi-supervised ED models. The experiments on two real-world datasets show that our method achieves the state-of-the-art results on the settings of both distant supervision and semi-supervision. In the future, we plan to explore the following directions: (1) We will extend our method to further extract event arguments and perform event extraction. (2) We will develop a large-scale and clean dataset for ED based on our method, which will benefit further research in this field.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2018YFB1004503), the National Natural Science Foundation of China (NSFC No. 61572273) and China Association for Science and Technology (2016QNRC001). Wang is also supported by the Tsinghua University Initiative Scientific Research Program.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of ACL Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of EMNLP*, pages 2074–2080.

Jun Araki and Teruko Mitamura. 2018. Open-domain event detection using distant supervision. In *Proceedings of COLING*, pages 878–891.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING*, pages 86–90.

P Basile, A Caputo, G Semeraro, and L Siciliani. 2014. Extending an information retrieval system through time event extraction. In *Proceedings of DART*, pages 36–47.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen, and Tiansi Dong. 2018. Joint representation learning of cross-lingual words and entities via attentive distant supervision. In *Proceedings of EMNLP*, pages 227–237.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of ACL*, pages 2587–2597.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of ACL*, pages 409–419.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 167–176.

Pengxiang Cheng and Katrin Erk. 2018. Implicit argument prediction with event knowledge. In *Proceedings of ACL*, pages 831–840.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of IJCNLP*, pages 352–361.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of ACL*, pages 66–71.

James Ferguson, Colin Lockard, Daniel Weld, and Hannaneh Hajishirzi. 2018. Semi-supervised event extraction with paraphrase clusters. In *Proceedings of NAACL*, pages 359–364.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of ACL*, pages 369–373.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*, pages 2574–2582.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of ACL-IJCNLP*, pages 369–372.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distant supervision for relation extraction via instance-level adversarial training. *arXiv preprint arXiv:1805.10959*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL-HLT*, pages 1127–1136.

Yu Hong, Wenxuan Zhou, Guodong Zhou, Qiaoming Zhu, et al. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of ACL*, pages 515–526.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of ACL*, pages 258–268.

Ruihong Huang and Ellen Riloff. 2012a. Bootstrapped training of event extraction classifiers. In *Proceedings of EACL*, pages 286–295.

Ruihong Huang and Ellen Riloff. 2012b. Modeling textual cohesion for event extraction. In *Proceedings of AAAI*, pages 1664–1670.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL*, pages 254–262.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010a. Filtered ranking for bootstrapping in event extraction. In *Proceedings of COLING*, pages 680–688.

Shasha Liao and Ralph Grishman. 2010b. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for Chinese event detection. In *Proceedings of ACL*, pages 1565–1574.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *Proceedings of AAAI*, pages 4865–4872.

Shaobo Liu, Rui Cheng, Xiaoming Yu, and Xueqi Cheng. 2018b. Exploiting contextual information via dynamic memory network for event detection. In *Proceedings of EMNLP*, pages 1030–1035.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging Framenet to improve automatic event detection. In *Proceedings of ACL*, volume 1, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of ACL*, pages 1789–1798.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of AAAI*, pages 2993–2999.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL*, pages 1626–1635.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. *Proceedings of ICLR*.

Aldrian Obaja Muis, Naoki Otani, Nidhi Vyas, Ruochen Xu, Yiming Yang, Teruko Mitamura, and Eduard Hovy. 2018. Low-resource cross-lingual event type detection via distant supervision with minimal effort. In *Proceedings of COLING*, pages 70–82.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of AAAI*, pages 5900–5907.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL-IJCNLP*, pages 365–371.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. DSGAN: Generative adversarial training for distant supervision relation extraction. In *Proceedings of ACL*, pages 496–505.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of ICLR*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, pages 148–163.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Adversarial multi-lingual neural relation extraction. In *Proceedings of COLING*, pages 1156–1166.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*, pages 1778–1783.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *Proceedings of ICLR*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of NAACL*, pages 289–299.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. DCFEE: a document-level Chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL*, pages 50–55.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of SIGIR*, pages 33–40.

Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, and Jiawei Han. 2018. Open-schema event profiling for massive news corpora. In *Proceedings of CIKM*, pages 587–596.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Ying Zeng, Yansong Feng, Rong Ma, and Zheng Wang. 2018. Scale up event extraction learning via automatic training data generation. In *Proceedings of AAAI*, pages 878–891.

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017. Improving event extraction via multimodal integration. In *Proceedings of MM*, pages 270–278.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of ACL*, pages 414–419.

A Submodular Feature-Aware Framework for Label Subset Selection in Extreme Classification Problems

Elham J. Barezi[1,2,3], **Ian D. Wood** [1], **Pascale Fung** [1,2,4], **Hamid R. Rabiee** [3]
[1]Center for Artificial Intelligence Research (CAiRE)
[2]Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[3]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[4]EMOS Technologies Inc. HK Limited
`ejs@cse.ust.hk, pascale@ust.hk, rabiee@sharif.edu`

Abstract

Extreme classification is a classification task on an extremely large number of labels (tags). User generated labels for any type of online data can be sparing per individual user but intractably large among all users. It would be useful to automatically select a smaller, standard set of labels to represent the whole label set. We can then solve efficiently the problem of multi-label learning with an intractably large number of interdependent labels, such as automatic tagging of Wikipedia pages. We propose a submodular maximization framework with linear cost to find informative labels which are most relevant to other labels yet least redundant with each other. A simple prediction model can then be trained on this label subset. Our framework includes both label-label and label-feature dependencies, which aims to find the labels with the most representation and prediction ability. In addition, to avoid information loss, we extract and predict outlier labels with weak dependency on other labels. We apply our model to four standard natural language data sets including Bibsonomy entries with users assigned tags, web pages with user assigned tags, legal texts with EUROVOC descriptors(A topic hierarchy with almost 4000 categories regarding different aspects of European law) and Wikipedia pages with tags from social bookmarking as well as news videos for automated label detection from a lexicon of semantic concepts. Experimental results show that our proposed approach improves label prediction quality, in terms of precision and nDCG, by 3% to 5% in three of the 5 tasks and is competitive in the others, even with a simple linear prediction model. An ablation study shows how different data sets benefit from different aspects of our model, with all aspects contributing substantially to at least one data set.

1 Introduction

Multi-label learning has recently attracted attention in the research community due to an increase in applications such as semantic labeling of images and videos, bio-informatics, genetic functions, and music categorization. In addition, multi-label learning can address machine learning problems in web data mining, including recommender systems, multimedia sharing websites, and ranking (Zhang and Zhang, 2010).

An important application of extreme multi-label learning is automatic tagging and social tagging of large information collections such as Wikipedia or the Web. A user can add their own keywords to a text, as if they were the keywords they would use to look for the article in a search engine. Since tags use an open vocabulary, the number of tags is increasing continually in order to adjust to the needs of new information. Moreover, different users can assign different tags to the same resource, resulting in a great diversity of tags for that resource.

The biggest challenge of extreme multi-label learning is the dimension of the output space. As the number of output labels increases, the number of output states increases exponentially. In order to overcome this exponential growth, it is necessary to use label dependencies to simplify the problem (Zhang and Zhang, 2010; Tsoumakas et al., 2010).

We propose a submodular maximization approach with a linear cost to find an informative set of labels. In contrast to the other similar approaches (Balasubramanian and Lebanon, 2012; Bi and Kwok, 2013) which consider only label-label dependencies, we also consider label-feature dependencies and outlier labels that are highly independent of other labels. Solving the problem using the selected (smaller number of) labels leads to minimizing both representation and training error.

Proceedings of NAACL-HLT 2019, pages 1009–1018
Minneapolis, Minnesota, June 2 - June 7, 2019.

Representation ability is equivalent to the power of the selected subset to reconstruct the remaining labels, and prediction ability is equivalent to training accuracy leading to less error propagation from predicted label subset to the remaining labels during reconstruction.

Submodular maximization approaches have proved very effective in many applications, such as finding the most influential nodes in social networks to maximize the spread of information (for applications such as advertising and marketing (Kempe et al., 2003; Ohsaka et al., 2014)) and video and image collection summarization (Gygli et al., 2015; Tschiatschek et al., 2014). There are many effective algorithms such as (Mirzasoleiman et al., 2015) to make submodular optimization approaches much faster or do them in a distributed way (Mirrokni and Zadimoghaddam, 2015) to perform faster parallel processing for very large scale datasets.

2 Related Work

Many of the early proposed multi-label learning approaches struggle with large-scale applications, as they learn each label separately or investigate the label dependencies in a way that leads to a costly and complicated model (Tsoumakas et al., 2010).

The other research trends is to transform the label space to a smaller space and map back the predicted results in the compressed space to the original space. Hsu et al. (2009) presented the first approach targeting label space compression based on compressed sensing, which assumes sparsity of the label space. An expensive optimisation problem has to be solved in the prediction step. Tai and Lin (2012); Chen and Lin (2012); Yu et al. (2014), and (Lin et al., 2014) used orthogonal projections and low-rank assumptions to extract a label matrix decomposition and find a low-dimensional embedding space. In (Bhatia et al., 2015b), the authors perform local embedding of the label vectors. To achieve stronger locality, they cluster the data into smaller regions, which is unstable and costly for high-dimensional spaces and one needs an ensemble of the learners to overcome this instability and achieve a good prediction accuracy.

Although the previously proposed approaches make the embedding space smaller and more tractable, they may lead to loss of information as a result of transforming the label space to lower-dimensional spaces. Many of these approaches rely on low-rank assumptions which transform the sparse label space to a new dense embedding space resulting in even lower accuracy, with a higher prediction cost in the new complicated space (Bhatia et al., 2015a).

Balasubramanian and Lebanon (2012) and Bi and Kwok (2013) proposed to select a subset of the labels, and solve the problem in the original label space, based on structure sparsity optimization and SVD decomposition, correspondingly. However, these methods are not tractable for large scale data and not compatible for the real application data. In addition, they have ignored the training error in the label selection step which can lead to selection of the labels that are hard to predict resulting in training error propagation through the next steps.

Another recent thread of research includes the methods that partition the data into smaller groups: In the framework proposed by Barezi et al. (2017), the label space is divided into smaller independent groups, while Agrawal et al. (2013); Prabhu and Varma (2014); Prabhu et al. (2018) propose tree-based methods which partition the data into tree-structured hierarchical groups. These partitioning-based approaches avoid information loss from dimension reduction. However, finding a partitioning tree is a very complicated and time-consuming problem and these approaches require solving a complicated optimization problem to perform partitioning at each node, which is expensive and needs many training samples. In addition, the tree-based approaches suffer from error propagation through the hierarchy and need many training samples to avoid under-fitting in the lower levels of the partitioning tree (Liu et al., 2005).

Instead of making the structural assumption on the relation between the labels, Yen et al. (2016) assume the label space is highly sparse and has a strong correlation with the feature space. They ignore the label space correlation information. Yen et al. (2017) proposed the parallel version of (Yen et al., 2016).

3 Methodology

In this paper, we propose a landmark selection framework for selecting the most informative labels and to solve the multi-label learning problem with these labels. As an example, consider predicting the commercial impact of a new event on some global organizations (equivalent to the labels in our problem) given a history of the impact of previous events (equivalent to the features and training data in our problem). Instead of predicting the impact on each organization individually, we predict only the impact on a small number of organizations which are both easier to predict and analyze according to available data as well as being more indicative of the economy and the other organizations. Being indicative means that if we know the impact of the new event on these organizations, it can help us to predict the reaction of the other organizations. More formally, we optimize the above set function $f(S)$ in Equation 1.

The proposed method includes both label-label and label-feature dependencies in order to minimize both representation and training error. Previous similar methods ignore label-feature dependencies in the subset selection step, allowing the training error for the selected subset of the labels to be propagated to the reconstructed labels and affecting the final predictions. In addition, to avoid information loss, we also extract and predict outlier labels with weak dependency on other labels and treat them separately.

Our construction results in a monotone submodular function of label sets allowing us to use a maximization framework that benefits from a good theoretical bound by a fast greedy approach with linear cost (Nemhauser et al., 1978). We use a method based on Alternating Direction Method of Multipliers (ADMM) (Boyd, 2011) optimization to learn a linear mapping back to original label space. Therefore, during training, we can select and learn the most informative label subset using a submodular maximization framework of linear cost. During the prediction time, we can use the selected subset to represent the remaining labels using a linear equation with a linear cost in number of the labels.

3.1 Overview of the Submodular Maximization Theorem

Submodular functions have a natural diminishing property which makes them suitable for many applications. A submodular function is a set function with the property that as the size of the selected subset increases, the incremental value of the function by adding a new element to the selected subset does not increase.

The formal definition of a submodular function is as follows:

Definition 1. *For a set function* $f(S) : 2^V \rightarrow R$ *defined for a finite ground set* $V = 1, 2, ...n$*, the marginal gain of adding each new member can be computed as* $\Delta_f(e|S) = f(e \cup S) - f(S)$*. The function* $f(.)$ *is submodular, if for each* $A \subseteq B \subseteq V$*,* $e \in V \backslash A \cap V \backslash B$*, then* $\Delta_f(e|A) \geq \Delta_f(e|B)$*. Equivalently, the function* $f(S) : 2^V \rightarrow R$ *is submodular if for any two sets* A*,* $B \in V$*,* $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$*.*

Monotony of sunmodular functions is a useful property which means that the value of the function would not decrease by adding each new member to the input set, and can be defined as following.

Definition 2. *A submodular function* $f(.)$ *is monotone (non-decreasing) if for every* $T \subseteq S$*, we have that* $f(T) \leq f(S)$*.*

A simple example of a submodular function is the setup cost in a factory. Suppose that a factory is capable of making any one of a large finite set V of products. In order to produce product $e \in V$, it is necessary to set up the machines needed to manufacture e, and this costs money. The setup cost is non-linear, and it depends on which other products you choose to produce. For example, if you are already producing iPhones, then the setup cost for also producing iPads is small, but if you are not producing iPhones, the setup cost for producing iPads is large.

We can find a good approximation of the optimum answer for a monotone submodular maximization problems by using the greedy approach and considering the selected subset size constraint. More formally:

Theorem 3. *(Nemhauser et al., 1978) For a non-negative, monotone submodular function* f*, let* S *be a set of size* k *obtained by the greedy strategy similar to Algorithm 1. Then,* $f(S) \geq (1 - 1/e)f(S^*)$*, where* S^* *is the optimum solution, and* e *is Euler's constant approximately equal to* 2.71828

3.2 Submodularity for Label Subset Selection

We propose two submodular functions, aiming to select the most informative subset of the labels. The first function is a penalized version of the graph cut function. It scores label sets with correlation to the other labels and penalizes their similarity to the previously selected labels (f_{pen} in

$$\begin{aligned} f(S) &= (How\ members\ of\ set\ S\ are\ individually\ predictable) + \\ &\quad (How\ members\ of\ set\ S\ can\ represent\ the\ members\ not\ included\ in\ S) \\ &= (Prediction\ ability) + (Representation\ ability) \\ &= (Label - Feature\ dependency) + (Label - Label\ dependency) \end{aligned} \tag{1}$$

Algorithm 1 $argmax_S\ f(S)\ s.t.\ \|S\| = k$.

```
Input: $V = 1, 2, ...n$
Initialization: $S = \emptyset$.
Repeat:
    1: $a^\star = argmax_{a \in V \setminus S} f(S \cup \{a\}) - f(S)$
    2: $S = S \cup \{a^\star\}$
Until $|S| = k$.
Output: $S$.
```

Equation 3). The graph is constructed using the labels as nodes and label correlations as weights for the graph edges. The second function scores the predictability of labels with respect to problem input features (f_{score} in Equation 5). Our final function for identifying the optimal subset of labels is a weighted sum of these (Equation 6).

We consider the label correlations as graph weights w. The graph cut function $f_{cut}(.)$ aims to find a subset of the graph nodes (labels) with the highest weights (strongest dependencies) to the remaining nodes (labels). This captures strong correlation of a label set to the other labels and thus its ability to reconstruct the other labels. The penalised version $f_{pen}(.)$ adds one more term to increase the diversity of the selected labels and avoid choosing similar labels.

$$f_{cut}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j} \tag{2}$$

$$f_{pen}(S) = f_{cut}(S) - \lambda \sum_{\substack{i,j \in S \\ i \neq j}} w_{i,j}\ ,\ \lambda \geq 0 \tag{3}$$

Theorem 4. *$f_{cut}(S)$ is a submodular function and it is monotone for non-large values of $|S|$ (Nemhauser et al., 1978).*

Theorem 5. *$f_{pen}(S)$ is a submodular function and it is monotone for non-large values of λ (Lin et al., 2009).*

The proofs for Theorem 4 and 5 is provided in supplementary Section.

It is important also to consider predictability, which is the training error for the selected subset of the labels, in order to avoid the prediction error of labels with high training error being propagated to the whole label space.

As an estimate of predictability we use either a G^2 or χ^2 independence test for the discrete data, and Fishers Z or t test for the continuous data in order to reject or accept the null hypothesis of independence (Tsamardinos and Borboudakis, 2010). Since, this measures include an implicit normalization, the frequency of the classes in training data does not affect the sampling step.

A higher dependency score for each label and the input feature space means a stronger correlation of the label with the feature space and higher predictability. Given label predictability scores f_{ij} for label i and input feature j and D input features, we calculate dependency scores f_i of the i-th label and the input features:

$$f_i = \sum_{j=1}^{D} f_{ij} \tag{4}$$

Note that $f_i \geq 0$. We then define the following set function, which also is monotone and submodular (Theorem 6):

$$f_{score}(S) = \sum_{i \in S} f_i \tag{5}$$

Theorem 6. *$f_{score}(S)$ is a submodular monotone function.*

Proof. For w_i = the sum over the dependency scores of the i-th label and the feature space, $f(S) = \sum_{i \in S} w_i$ is a linear function with $w_i \geq 0$. Any linear function of the form $f(S) = \sum_{i \in S} w_i$ is a submodular function. If $S \subset R$, $\Delta_f(k|S) - \Delta_f(k|R) = 0 \Rightarrow \Delta_f(k|S) \geq \Delta_f(k|R)$.
Additionally, if $\forall i\, w_i \geq 0$, then f is monotone, because $f(S \cup k) - f(S) = w_k,\ w_k \geq 0$.
$max_{|S|=k} f(S) = max \sum_{i \in S} w_i$. Therefore $f(S)$ is a monotone submodular function. □

Since, any sum of submodular functions with positive coefficients is a submodular function, we can combine $f_{pen}(.)$, and $f_{score}(.)$ by positive weights, which results in a new submodular function that includes both representation ability and prediction ability of the selected labels. We choose a model parameter $\gamma > 0$ giving us our final submodular function:

$$f(S) = f_{pen}(S) + \gamma . f_{score}(S) \tag{6}$$

3.3 Landmark Information Propagation

The main step of our proposed framework is to propagate the predicted value for the selected label subset to the full set of labels in order to recover the original space. Therefore, we aim to find a linear relation including the dependency of the selected labels and all the other labels. In the prediction step, this linear function obtains the full label set by combining the subset (Y_s) and outlier predictions (E) predicted by the regression functions discussed in next section 3.4. Given 1-hot representations Y_s over the reduced set of labels and Y the full set of labels, we seek matrices Z and E that recover the original labels:

$$Y = Y_s Z + E \tag{7}$$

To find optimal Z and E, we solve the optimization problem Equation 8, where Y and Y_s are matrices populated with our training data. Note that $\|E\|_{2,1}$ is the L_1 norm of the L_2 norms of the columns of E.

$$\begin{aligned} \underset{Z,E}{\text{argmin}} \quad & (\|Z\|_1 + \alpha \|E\|_{2,1}) \\ \text{s.t.} \quad & Y = Y_s Z + E \end{aligned} \tag{8}$$

The sparse matrix Z is a $k \times L$ matrix which includes a few representative labels (due to the sparsity constraint $\|Z\|_1$) for each label ($Y = Y_k Z$). The Z matrix includes the dependency information and performs propagation of the predicted label subset to the full label set, while nonzero columns of matrix E show the outlier and tail labels set O, which cannot be computed perfectly through their relation to the other labels. The index set of the nonzero columns of matrix E indicates the outlier labels. α is a model parameter.

The alternating direction method of multipliers (ADMM) method (Boyd, 2011; Nesterov, 2004; Beck and Teboulle, 2009) provides an efficient algorithm for solving this problem, achieving a convergence rate of $O(1/T^2)$ (where T is the number of iterations). ADMM solves the problem with more than one unknown variable, (Z and E in our case), by alternating between optimizing each variable using augmented Lagrangian. Please see the supplementary materials for more detail on the ADMM method and how it is applied in this case.

3.4 Prediction and Mapping Back to the Original Label Space

We now train a linear classifier to predict labels in the reduced label set $S \cup O$ and map back to the full label set. Given features of the training data X, corresponding labels from selected and outlier labels Y_S and Y_O, we learn linear regression parameters w_s, b_s for the selected labels and w_e, b_e for the outlier labels:

$$\begin{aligned} & \underset{w_s,b_s}{\text{argmin}} \quad \|Y_s - (X * w_s + b_s)\| + \tfrac{\lambda_1}{2}\|w_s\|^2 \\ & \underset{w_e,b_e}{\text{argmin}} \quad \|Y_o - (X * w_e + b_e)\| + \tfrac{\lambda_2}{2}\|w_e\|^2 \end{aligned} \tag{9}$$

Since all these training tasks are independent of each other, this step is highly parallelizable. The final values for the labels are computed by propagating the selected label subset through the linear relation 7:

$$\begin{aligned} \hat{Y}_s &= X * w_s + b_s \\ \hat{E} &= X * w_e + b_e \end{aligned} \tag{10}$$

$$\hat{Y} = \hat{Y}_s Z + \hat{E} \tag{11}$$

An overview of steps for training and prediction are shown in Algorithms 2 and 3.

Algorithm 2 Training Algorithm.

```
Input: Training Data $X$ and $Y$.
Find the best label subset by submodular optimization over function 6;
Find the linear propagation equation through ADMM optimization over problem 8.
Find the linear regression models over small subset of labels and outliers by Equation 9
Output: Label subset, outliers, propagation and regression models.
```

4 Experiments

4.1 Datasets

We used six different datasets in the experiments. The "Bibtex" dataset is a text dataset extracted from the BibSonomy website (Katakis et al., 2008)

```
Algorithm 3 Prediction Algorithm.
Input: prediction samples $X$.
  1: Predict candidate label subset and outlier labels using regression model 10.
  2: Use 11 to produce full set of labels from candidate subset and outlier labels.
Output: Full label set for input $X$.
```

contains metadata for the bibtex items like the title of the paper, the authors, etc and extracts the features according to the term frequency. The "Mediamill" dataset is extracted from the Mediamill contest datasets, which include low-level multimedia features (visual and textual features) extracted from 85 hours of international news videos from the TRECVID 2005/2006 benchmark datasets (Snoek et al., 2006) labeled using a lexicon of 101 semantic concepts, like commercials, nature, and baseball.

The "Eurlex" dataset includes 19,348 legal documents from European nations, containing several different types of documents, including treaties, legislation, case-law and legislative proposals, classified according to the EUROVOC descriptor using 3993 different classes, and 5000 features extracted using common TF-IDF term weighting (Mencia and Fürnkranz, 2008). The "Delicious" dataset is a text dataset extracted from the `del.icio.us` social bookmarking site on the 1st of April 2007 and contains textual data of web pages along with their user defined tags (Tsoumakas et al., 2008). The content of web pages was represented using the Boolean bag-of-words model. "Wiki10-31K" is a collection of social tags for given Wikipedia pages with TF-IDF features (Zubiaga, 2012). The statistics of these datasets are provided in Table 1.

4.2 Experimental Setup

For the small datasets, "Bibtex", "Mediamill", "Delicious", and "Eurlex", the reported results are the average of 10 different experiments for random partitions of each dataset. For the larger dataset, "Wiki10-31K", we did one experiment with the training and testing partition reported in Table 1.

For all experiments we chose a label subset size of 100, except for Mediamill where we chose 30 since 100 would represent all labels. Model tuning is done in two phases: first we tune α for group sparsity (Equation 8), and γ for weighting of the submodular functions (Equation 6), then we tune for λ_1 and λ_2, the regression parameters for mapping back to the original label set (Equation 9) with α and γ fixed. All parameters were chosen by measuring the precision of 10-fold cross validation and using a grid search over the values $\{0, 10^{-3,\ldots,+3}\}$ for each dataset.

The proposed method was compared with several state-of-the-art methods with diverse approaches. LEML (Yu et al., 2014), CPLST (Chen and Lin, 2012), CS (Hsu et al., 2009) and SLEEC (Bhatia et al., 2015b) which are embedding based approaches with a low-rank or sparse assumption in the label space. ML-CSSP (Bi and Kwok, 2013) which solves the problem in the original label space which ignores the training error in the subset selection step. FastXML (Prabhu and Varma, 2014), and PD-sparse (Yen et al., 2016) which do not use an embedding transformation and aim to solve the problem without using compression or sampling. We have used the reported results, if available, and otherwise tuned the parameters for the baseline algorithms by means of 10-fold cross validation.

5 Results and Discussion

Table 2 shows the average and standard deviation of Precision@k for the four small-scale datasets, "Bibtex", "Mediamill", "Delicious", and "Eurlex", and the large-scale dataset "Wiki10-31k". For "Wiki10-31k", results are reported only for those baselines that were tractable. The results for nDCG@k are included in supplementary Material, Table 5. Since the SLEEC and FastXML methods are ensemble-based, using multiple non-linear models, it is not fair to compare them with the single model methods such as our own. These methods partition the sample space into smaller tractable clusters and obtain separate classifiers for each partition. We compare our method with these in Table 3.

The proposed approach in most cases has significantly better results than other methods on both measures. The embedding based approaches suffer from accumulation of the embedding and training error (Balasubramanian and Lebanon, 2012), however in the proposed approach, we have removed the embedding step and considered the training error minimization at the label subset selection step. On the other hand, the non-embedding approaches such as PD-sparse (Yen et al., 2016) ignore the label space inter-

Dataset	**Domain**	**Number of Features**	**Number of Labels**	**Training Points**	**Testing Points**
Bibtex	Text	1836	159	4880	2515
Delicious	Text(Web)	500	983	12920	3185
Mediamill	Video	120	101	30993	12914
Eurlex	Text	5000	3993	17413	1935
Wiki10-31K	Text	101938	30938	14146	6616

Table 1: Dataset statistics

	Proposed	PD-sparse	LEML	CPLST	CS	ML-CSSP
			Bibtex			
P@1	**64.56**±0.79	61.29±0.65	62.54±0.52	62.38±0.63	58.87±0.61	44.98±1.15
P@3	**39.51**±0.34	35.82±0.46	38.41±0.42	37.84±0.48	33.53±0.49	30.43±0.59
P@5	**28.80**±0.26	25.74±0.30	28.21±0.24	27.62±0.27	23.72±0.29	23.53±0.37
			Delicious			
p@1	*65.13*±0.39	51.82±1.40	**65.67**±0.73	*65.31*±0.88	61.36±0.38	63.04±1.28
P@3	59.07±0.41	44.18±1.04	**60.55**±0.48	59.95±0.43	56.46±0.33	56.26±1.13
P@5	54.52±0.34	38.95±0.94	**56.08**±0.43	55.31±0.50	52.07±0.30	50.16±0.83
			Mediamill			
P@1	**84.25**±0.27	81.86±4.08	*84.01*±0.31	83.35±0.33	*83.82*±5.92	78.95±0.23
P@3	*67.29*±0.24	62.52±2.31	*67.20*±0.23	66.18±0.22	**67.32**±4.42	60.93±0.24
P@5	**52.90**±0.15	45.11±1.14	52.80±0.18	51.46±0.20	52.80±2.61	44.27±0.20
			Eurlex			
P@1	**81.04**±0.81	76.43±1.04	63.40±1.58	72.28±0.99	58.52±1.06	62.09±2.12
P@3	**67.91**±0.97	60.37±0.74	50.35±1.44	58.16±1.11	45.51±0.71	48.39±1.31
P@5	**56.81**±0.97	49.72±0.74	41.28±1.07	47.73±0.97	32.47±0.58	40.11±1.10
			Wiki10-31k			
p@1	**86.05**	82.14	73.47	-	-	-
P@3	**76.85**	69.68	62.43	-	-	-
P@5	**67.77**	58.76	54.35	-	-	-

Table 2: Non-ensemble models with k=100 or 30 (Mediamill). Best in **bold** and not significantly different to best at p=0.05 in *italics*.

	Proposed	SLEEC	FastXML
		Bibtex	
P@1	*64.56*±0.79	**65.08**±0.65	63.42±0.67
P@3	*39.51*±0.34	**39.64**±0.39	39.23±0.57
P@5	*28.80*±0.26	**28.87**±0.32	28.86±0.38
		Delicious	
P@1	65.13±0.39	67.59±0.53	**69.61**±0.58
P@3	59.07±0.41	61.38±0.59	**64.12**±0.75
P@5	54.52±0.34	56.56±0.54	**59.27**±0.65
		Mediamill	
P@1	84.25±0.27	**87.82**±0.33	84.22±0.27
P@3	67.29±0.24	**73.45**±0.30	67.33±0.20
P@5	52.90±0.15	**59.17**±0.34	53.04±0.18
		Eurlex	
P@1	**81.04**±0.81	79.26±0.86	71.36±1.63
P@3	**67.91**±0.97	64.30±0.88	59.90±1.58
P@5	**56.81**±0.97	52.33±0.80	50.39±1.40
		Wiki10-31k	
p@1	**86.05**	85.88	83.03
P@3	**76.85**	72.98	67.47
P@5	**67.77**	62.70	57.76

Table 3: Ensemble-based nonlinear models. Best in **bold** and not significantly different to best in *italics*.

dependency information which can be useful to improve the prediction accuracy for the labels which are not easy to predict only from input features.

ML-CSSP (Bi and Kwok, 2013) and the work of Balasubramanian and Lebanon (2012) attempt, like us, to find the most informative labels in order to perform label subset selection. However, our approach improves on their results, supporting the idea that considering only the label space information (ignoring label-feature dependency information) in the label selection step can lead to label sets that are not easy to predict whose training error will be propagated through to final model predictions.

The SLEEC and FastXML methods are ensemble-based methods using multiple nonlinear models and can be expected to outperform single model methods such as ours. SLEEC aims to partition the sample space into smaller tractable clusters to obtain a nonlinear embedding and trained model for each partition. FastXML finds a partitioning tree by using nonlinear binary classifiers to partition the samples at each node, which is a very complicated and unstable problem for high-dimensional spaces. Therefore, for both SLEEC and FastXML methods, they need an ensemble of the learners in order to overcome this instability and achieve a good prediction accuracy. Table 3 shows that SLEEC performs best on the Mediamill and FastXML performs best on the Delicious dataset. This shows that finding a representative subset using a linear method is not a consistent assumption for these datasets than the low-rank and tree-based assumptions. However, for Bibtex dataset, our proposed method is competitive with the best results, and for Eurlex and Wiki10-31k, our method is substantially better than both SLEEC and FastXML, a notable achievement for a single model approach.

5.1 Ablation study

The ablation study results in Table 4 shows how different data sets benefit from different parts of our proposed framework, with all parts contributing substantially to at least one data set. We have reported the results by considering only label-label dependency information (f_{pen}), label-feature dependency information (f_{score}) and combining all 3 parts (f_{pen}, f_{score} and outlier information). The results support the assertions that considering only the label space information (ignoring label-feature dependency information) in the label selection step causes prediction error of labels with high training error to be propagated to the whole label space and that it is important to also select outlier labels that are hard to predict from other selected labels.

	f_{pen}	f_{score}	$f_{pen} + \alpha f_{score}$	**+Outliers**
Bibtex				
P@1	60.98	**63.27**	63.29	**64.55**
P@3	34.86	**37.10**	37.55	**39.51**
P@5	25.94	**26.73**	27.05	**28.78**
Mediamill				
P@1	81.12	81.83	**84.25**	84.25
P@3	64.15	65.92	**67.79**	67.99
P@5	51.26	51.66	**52.70**	52.90
Delicious				
P@1	62.71	62.71	**64.33**	65.14
P@3	56.95	56.95	**58.30**	59.10
P@5	52.63	52.63	**53.58**	**54.55**
Eurlex				
P@1	**56.60**	3.84	56.60	**81.04**
P@3	**37.88**	3.11	37.88	**67.91**
P@5	**29.71**	3.01	29.71	**56.81**
Wiki10-31k				
P@1	**81.86**	54.34	81.86	**86.05**
P@3	**68.51**	40.41	68.51	**76.85**
P@5	**56.77**	33.00	56.77	**67.77**

Table 4: Ablation Study. Bold indicates a difference of $\geq 0.8\%$

We also investigated the effect of changing the subset size S on the final prediction quality (we have ignored the outlier effect in these experiments). Figure 1 shows an initial marked increase in performance with subset size, however the results gets more stable when the subset size gets larger. This observation, which is consistent with the submodular property, provides a clue that using a more complicated training model, like a nonlinear model, for a smaller selected set of labels may lead to higher performance than increasing the subset size while using a linear model.

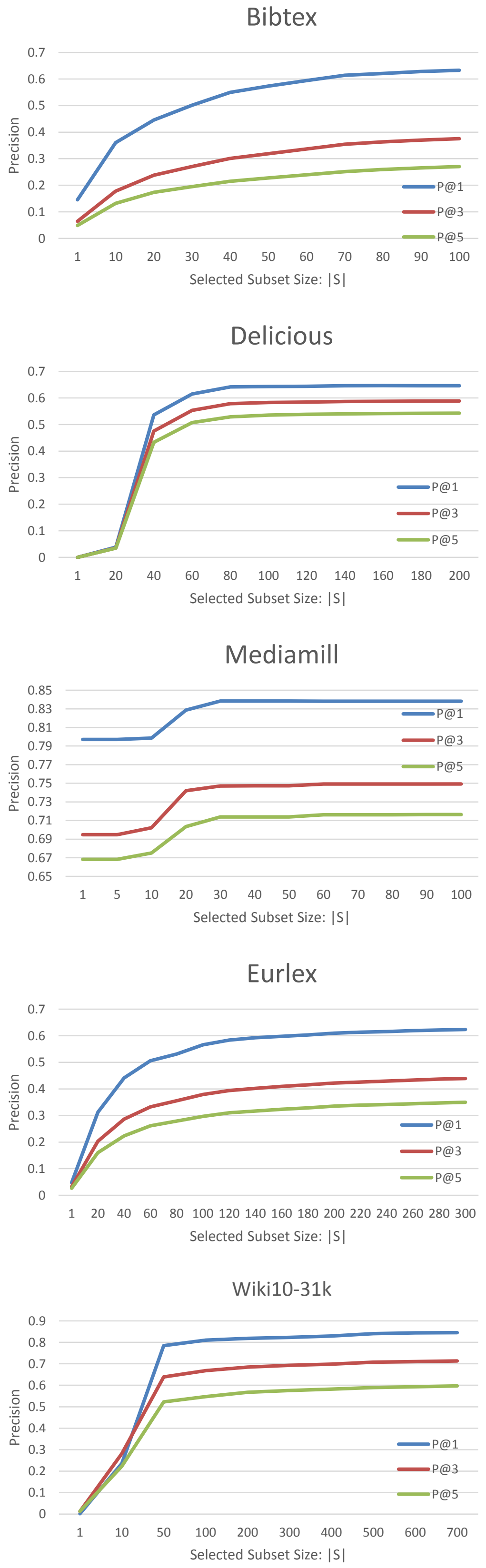


Figure 1: Precision score changes by subset size without outliers.

6 Conclusion and Future Work

We propose a novel approach for extreme multi-label classification that simplifies the problem by selecting an informative and easily modelled subset of labels and subsequently mapping back to the full set of labels. While the method is very well applicable to text datasets, it is applicable as a general ML method for different domains. Our novel label selection mechanism follows three principles: A new submodular maximisation framework that combines label-label dependencies and label training error together with a mechanism to identify outlier labels that are hard to reconstruct. Modelling only the most informative labels helps to avoid transforming the label space to a new embedding space leading to accumulation of training and embedding errors. We use a greedy approach for our monotone submodular framework with linear cost and good theoretical convergence. Extensive experiments using a linear prediction model on selected labels conducted on five standard real-world datasets demonstrate that our method achieves better performance than single model approaches, and better or comparable performance to ensemble based methods. In future, we can improve our model by using nonlinear training model instead of a simple linear regression model for the selected subset of the labels. Moreover, ablation study results suggest that a nonlinear propagation model to reconstruct the full label set may be of benefit.

Acknowledgments

This work was partially funded by grants #16214415 and #16248016 of the Hong Kong Research Grants Council, ITS/319/16FP of Innovation Technology Commission, and RDC 1718050-0 of EMOS.AI.

References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of WWW-13*, pages 13–24, Seoul, Korea.

Krishnakumar Balasubramanian and Guy Lebanon. 2012. The landmark selection method for multiple output prediction. In *Proceedings of ICML-12*, pages 983–990, Edinburgh, Scotland.

Elham J Barezi, James T Kwok, and Hamid R Rabiee. 2017. Multi-label learning in the independent label sub-spaces. *Pattern Recognition Letters*, 97:8–12.

Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on Imaging Sciences*, 2(1):183–202.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Prateek Jain, and Manik Varma. 2015a. Locally nonlinear embeddings for extreme multi-label learning. In *NIPS*, Montreal, Canada.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015b. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738.

Wei Bi and James Kwok. 2013. Efficient multi-label classification with many labels. In *Proceedings of ICML-13*, pages 405–413, Atlanta.

Stephen Boyd. 2011. Alternating direction method of multipliers. In *Talk at NIPS workshop on optimization and machine learning*.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Yao-Nan Chen and Hsuan-Tien Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1529–1537, Harrahs and Harveys.

Michael Gygli, Helmut Grabner, and Luc Van Gool. 2015. Video summarization by learning submodular mixtures of objectives. In *Proceedings CVPR 2015*, pages 3090–3098.

Bingsheng He and Xiaoming Yuan. 2012. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709.

Daniel Hsu, Sham Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *NIPS*, volume 22, pages 772–780, Vancouver CANADA.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, page 75.

David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM.

Hui Lin, Jeff Bilmes, and Shasha Xie. 2009. Graph-based submodular selection for extractive summarization. In *IEEE Workshop on ASRU*, pages 381–386.

Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. 2014. Multi-label classification via feature-aware implicit label space encoding. In *Proceedings of ICML-14*, pages 325–333, Beijing, China.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *Acm Sigkdd Explorations Newsletter*, 7(1):36–43.

Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Machine Learning and Knowledge Discovery in Databases*, pages 50–65. Springer.

Vahab Mirrokni and Morteza Zadimoghaddam. 2015. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162. ACM.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. 2015. Lazier than lazy greedy. In *AAAI*, pages 1812–1818.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294.

Y Nesterov. 2004. Introductory lectures on convex programming: a basic course, volume i.

Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *AAAI*, pages 138–144.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 993–1002. International World Wide Web Conferences Steering Committee.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 263–272, New York City. ACM.

Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *14th Annual ACM International Conference on Multimedia*, pages 421–430. ACM.

Farbound Tai and Hsuan-Tien Lin. 2012. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542.

Ioannis Tsamardinos and Giorgos Borboudakis. 2010. Permutation testing improves bayesian network learning. *Machine Learning and Knowledge Discovery in Databases*, pages 322–337.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. 2014. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421.

G Tsoumakas, I Katakis, and I Vlahavas. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD08)*, pages 30–44.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer.

Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM.

Ian EH Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of ICML-16*.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S Dhillon. 2014. Large-scale multi-label learning with missing labels. In *Proceedings of ICML-14*, volume 32, Beijing, China.

Min-Ling Zhang and Kun Zhang. 2010. Multi-label learning by exploiting label dependency. In *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 999–1008. ACM.

Arkaitz Zubiaga. 2012. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*.

Relation Extraction with Temporal Reasoning Based on Memory Augmented Distant Supervision

Jianhao Yan[1], Lin He[1], Ruqin Huang[2], Jian Li[3]*, and Ying Liu[1]
[1]Institute for Network Sciences and Cyberspace, Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University
[3]Institute for Interdisciplinary Information Sciences, Tsinghua University
`{yjh16, he-l14, hrq16}@mails.tsinghua.edu.cn`
`lijian83@tsinghua.edu.cn, liuying@cernet.edu.cn`

Abstract

Distant supervision (DS) is an important paradigm for automatically extracting relations. It utilizes existing knowledge base to collect examples for the relation we intend to extract, and then uses these examples to automatically generate the training data. However, the examples collected can be very noisy, and pose significant challenge for obtaining high quality labels. Previous work has made remarkable progress in predicting the relation from distant supervision, but typically ignores the temporal relations among those supervising instances. This paper formulates the problem of relation extraction with temporal reasoning and proposes a solution to predict whether two given entities participate in a relation at a given time spot. For this purpose, we construct a dataset called WIKI-TIME[1] which additionally includes the valid period of a certain relation of two entities in the knowledge base. We propose a novel neural model to incorporate both the temporal information encoding and sequential reasoning. The experimental results show that, compared with the best of existing models, our model achieves better performance in both WIKI-TIME dataset and the well-studied NYT-10 dataset.

1 Introduction

As an important technique to automatically complete the knowledge base and reduce labeling efforts, distant supervision (DS) for relation extraction has drawn much attention. In DS, we align the entity pair $(head, tail)$ from a triple $\langle head, rel, tail\rangle$ extracted from a huge knowledge base (e.g., Freebase, Wikidata) with sentences from free texts (e.g., Wikipedia, New York Times) to obtain the training examples, and the label of such an example is the corresponding relation *rel*. Therefore, DS can automatically create a set of training data for each entity pair.

However, the noisy training data problem (Riedel et al., 2010) significantly affects the performance of DS. Therefore, most of the recent approaches (Riedel et al., 2010; Hoffmann et al., 2011; Zeng et al., 2015; Lin et al., 2016) follow a common assumption called the at-least-once assumption, which treats all aligned sentences of each entity pair as one training sample. We refer to a sentence as an instance and all sentences aligned to one entity pair as a mention set in the following, respectively.

The models in previous work (Zeng et al., 2015; Lin et al., 2016; Luo et al., 2017) generally include two parts, **encoding** and **fusion**. The former encodes each instance into a low-dimensional representation. The latter combines representation of each instance. Then, their combination is used to predict the relation.

Although the approaches mentioned above seem promising, they have the following limitations:

1. They all use a separate but identical encoding module among instances and introduce no difference temporally.
2. They only adopt single step of fusion and introduce no sentence-level reasoning.

We remark that the aforementioned approaches may be enough for the standard NYT-10 dataset (Riedel et al., 2010), because the dataset only extracts instances from New York Times corpus from the year 2005 to 2007 and consists of few mention sets with long time span. However, as one can easily imagine, ignoring temporal information may cause inaccurate predictions, especially when a mention set has a long time span and some

---

*Jian Li is the corresponding author.

[1] https://github.com/ElliottYan/DS_Temporal

Proceedings of NAACL-HLT 2019, pages 1019–1030
Minneapolis, Minnesota, June 2 - June 7, 2019.

instances express different relations. For example, suppose we want to predict the relation between Angelina Jolie and Brad Pitt (using Wikidata). The knowledge base contains a factual relation of *spouse* between them with the valid period from August 2014 to September 2016. However, the extracted mention set contains instances about their marriage in 2014, as well as their divorce in 2016. Because existing models do not encode temporal information, the relation they extract is likely to be the one with highest confidence. In this example, their models may predict the relation of marriage since the instances may suggest a higher confidence for the relation of marriage. But the correct prediction should be divorce. As shown in the above example, we can see it is necessary to include temporal information in DS.

On the other hand, in fusion module, most existing work focused on denoising using methods such as attention or reinforcement learning. We want to argue that a sentence-level reasoning can also be useful since there are instances which are not direct positive examples for the given relation, but can provide supporting evidence. We call them remote instances. Consider the Jolie-Pitt example again. Suppose we are to predict their relation after their divorce. The instances about their marriage also indirectly help to infer their divorce since marriage is the premise of divorce. Hence, we need an algorithm that can incorporate temporal information and perform reasoning over remote instances.

In this paper, we address both limitations and extend the task to predict the relation of a particular entity pair at any specific time spot. The problem can be formulated as a sequence labeling problem (See § 2). We propose a novel relation extraction architecture that can address both aforementioned limitations. Our model follows the popular encoding-fusion architecture, but makes two crucial modifications. Firstly, we introduce temporal encoding to model the temporal information among the instances in the encoding. Secondly, we use the Memory Network (Sukhbaatar et al., 2015; Miller et al., 2016) to iteratively reason over temporally augmented encodings in the fusion part.

Moreover, we evaluate our model on the widely studied NYT-10 dataset (Riedel et al., 2010) and a new WIKI-TIME dataset. The construction of WIKI-TIME is similar to that of the NYT-10 dataset except for two important differences. One is that we only consider triples $\langle head, rel, tail\rangle$ with the valid period $(T1, T2)$. For example, the triple $\langle Jolie, married, Pitt\rangle$ has a valid period of $(2014.08, 2016.09)$. The other is that we extract contextual temporal information for each aligned instance. We use Wikidata (Vrandečić and Krötzsch, 2014) as knowledge base and Wikipedia as free corpus. Both automatic and manual evaluation are applied in the experiments. The experimental results show that, compared with existing models, our model can achieve comparable/better performance in both WIKI-TIME and standard NYT-10 datasets.

Our main contributions can be summarized as follows:

- We introduce a new task aiming to solve the problem of relation extraction with temporal information.
- We propose a novel relation extraction architecture, which encodes both the temporal and semantic information and includes remote instances for temporal reasoning.
- We construct a new WIKI-TIME dataset by aligning Wikidata to Wikipedia, which is specially designed for the task of relation extraction with temporal information.
- The experiment results show that, compared with the best of existing models, our model achieves comparable/better performance both in WIKI-TIME dataset and standart NYT-10 dataset.

2 Formulation

2.1 Traditional Distant Supervision (DS)

The traditional distant supervision (DS) task can be defined as:

Given two entities $\langle head, tail\rangle$ and their corresponding mention set $S = \{s_1, s_2, \cdots, s_T\}$, where s_i denotes the ith instance, the task aims to predict the probability for specific relation r of $\langle head, tail\rangle$:

$$P(r|S = \{s_1, s_2, \cdots, s_T\}). \tag{1}$$

The task can be seen as a multi-label multi-instance classification problem.

2.2 Distant Supervision with Temporal Reasoning

In distant supervision with temporal reasoning, our goal is to predict the relation between two entities at any specific time spot. Because modeling over any specific time spot is non-trivial, we relax the goal to predict the relation between two given entities at any mentioned time spots. Note that we can infer the relations at other time spots using prediction at mentioned ones. Formally, the relation r_t at $t \in (t_1, t_2]$ can be infered by r_{t_1}. Therefore, we can model the problem as a sequence labeling problem with noisy inputs.

Given two entities, we collect the chronologically sorted list of its mention instances and the time spot associated with each instance . We denote the list by $S = \{(s_1, t_1), \cdots, (s_T, t_T)\}$, where (s_i, t_i) is the ith instance and the associated time spot. Our goal is to predict the probability of relation r at time spot t_i:

$$P(r_{t_i}|S = \{(s_1, t_1), \cdots, (s_T, t_T)\}, t_i). \quad (2)$$

3 Methodology

3.1 Overview of TempMEM

Note that RNN-like models are not suitable for this sequence labeling problem, because the input sequence contains noisy sentences and lacks direct dependency between time steps. We propose a neural model called TempMEM which models the sequence labeling problem by creating query sequence based on each mentioned time spot.

TempMEM also follows the encoding-fusion framework (Zeng et al., 2015; Lin et al., 2016; Luo et al., 2017). However, we make two crucial modifications to the original framework. First, for the encoding part, we use time-aware encoding modules for instances instead of identical ones. Second, we use the memory network to iteratively reason over instances, which makes use of remote instances.

In the following sections, we introduce how to encode the temporal and semantic information and include remote instances for temporal reasoning.

3.2 Encoding

3.2.1 Sentence Encoding

For sentence encoding, here we apply the Convolutional Neural Network (CNN) and the Piecewise Convolutional Neural Network (PCNN) (Zeng et al., 2015). Note that, since TempMEM has no preferance over specific sentence encoding, other encoding modules like word memory (Feng et al., 2017) or self-attention can also be used here.

The inputs of convolution layers are word embeddings concatenated with position features. For a detailed description of the inputs, we refer the readers to (Zeng et al., 2015).

First, the convolution layer extracts local features with sliding window $\hat{w}$ over the input representation. Formally, the convolution operates on the concatenation of the input representations $\boldsymbol{X}_{k:k+\hat{w}}$ of instance j with the shared parameters $\boldsymbol{W}_c \in \mathbb{R}^{D*\hat{w}}$ and $b_c \in \mathbb{R}^1$:

$$o_{c,k} = \boldsymbol{W}_c \cdot \boldsymbol{X}_{k:k+\hat{w}} + b_c, \quad (3)$$

where $o_{c,k}$ is the kth output of channel c.

Then, we use the piece-wise max-pooling layer. It divides the outputs of filters into three parts $\{\boldsymbol{o}_{c,0:h}, \boldsymbol{o}_{c,h:t}, \boldsymbol{o}_{c,t:N}\}$ and performs max-pooling over each part:

$$\boldsymbol{o}_c = [\max_{0 \le k < h}(o_{c,k}), \max_{h \le k < t}(o_{c,k}), \max_{t \le k < N}(o_{c,k})], \quad (4)$$

where h and t denote the indices of the head and tail entities, respectively. The concatenation of the output of all channels c is considered as the convolutional representation of instance j:

$$\boldsymbol{O}_j = [\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_C], \quad (5)$$

where C denotes the number of filters.

3.2.2 Temporal Encoding

In order to introduce temporal priorities among instances, it is necessary to inject temporal information into the encoding part. We want the temporal encoding to have the following characteristics:

- The temporal encodings should comply with the chronological order of instances.
- The difference between two time spots determines the similarity between two temporal encodings.

Since directly encoding the time spot value leads to huge difference among mention sets of the dataset, we propose an approximate approach with PE encoding (Vaswani et al., 2017) based on the rank (i.e. position of an instance in a mention set with chronological order):

$$\boldsymbol{PE(j)} = \begin{cases} sin(j/10000^{d/d_m}) & if\ d\%2 = 0 \\ cos(j/10000^{(d-1)/d_m}) & if\ d\%2 = 1 \end{cases}, \quad (6)$$

where j is the rank of instance s, d is the dimension, and d_m is the dimension of temporal encoding. Obviously, the PE encoding complies with the chronological order and the similarity between

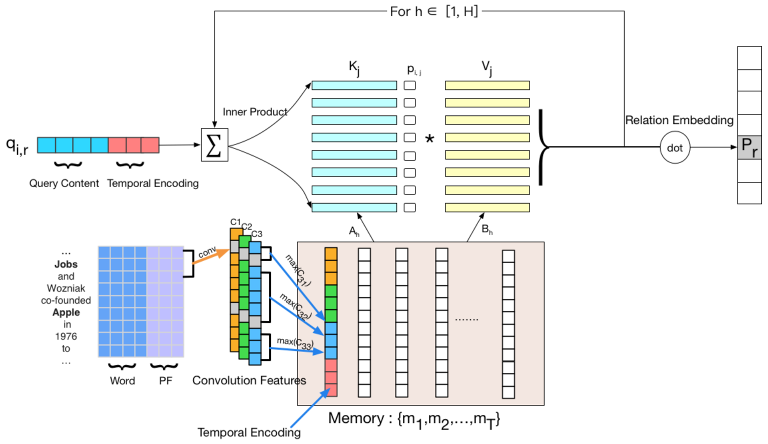


Figure 1: Overall TempMEM architecture

two PE encodings (by dot product) is determined by their rank difference.

Then, we concatenate the corresponding temporal encoding with the convolutional features of instance j to form the final representation of each instance with a learnable scale factor λ:

$$\boldsymbol{m}_j = [\boldsymbol{O}_j; \lambda \cdot \boldsymbol{PE(j)}]. \tag{7}$$

3.3 Fusion

In the fusion part, we use the Memory Network to perform temporal reasoning among different instances. Each encoded instance is considered to be a memory slot. Then, we construct a time specific query and iteratively compute the weighted attention over all instances. We detail the process in the following sections.

3.3.1 Query Construction

We construct each query with the guidance of the following intuition.

- Relation extraction within instances is equal to the query "what is the *relation* between *head* and *tail* at time spot t_i?".

So, we construct our queries based on four key variables, $(relation, head, tail, t_i)$.

Specifically, we combine the embeddings (pretrained by TransE (Bordes et al., 2013)) of *head* and *tail* and project the combination through an affine matrix $\boldsymbol{\Phi}_q \in \mathbb{R}^{D_e * D_r}$, where D_e and D_r denote the dimension of relation and entity embedding, respectively. After the projection, we add the randomly initialized relation embedding. The formal definition of a query is given below:

$$\boldsymbol{q}_r = \boldsymbol{R}_r + (\boldsymbol{E}_{head} + \boldsymbol{E}_{tail}) * \boldsymbol{\Phi}_q, \tag{8}$$

where $\boldsymbol{R}_r \in \mathbb{R}^{D_r}$ is the embedding of specific relation r and $\boldsymbol{E}_* \in \mathbb{R}^{D_e}$ is the entity embedding. Finally, we also concatenate the query with the same temporal encoding defined in § 3.2.2 to obtain the ith query:

$$\boldsymbol{q}_{r,i} = [\boldsymbol{q}_r; \lambda \cdot \boldsymbol{PE(i)}]. \tag{9}$$

3.3.2 Iterative Reasoning

In this part, we introduce how to use the queries to perform temporal reasoning. Two operations are involved, memory addressing and reading.

One of our key motivations is to consider the remote instances. So, instead of using single step attention computation as in previous work (Lin et al., 2016; Luo et al., 2017; Ji et al., 2017), we perform an overall H steps of memory addressing and reading to obtain the final prediction. Within each step (also called hop), we update the query value by adding the output of the previous step, which provides a gradual shift in attention. Next, we introduce the whole process in detail.

Memory Addressing In addressing, we compute the similarity between the query vector $\boldsymbol{q}_{i,r}$ and each candidate memory slot key $\boldsymbol{K}_j$. Note that the encoding output $\boldsymbol{m}_j$ is not in the same continuous space as the query vector. So, we adopt linear projections to both memory keys:

$$\boldsymbol{K}_j = \boldsymbol{A}_h^T \cdot \boldsymbol{m}_j, \tag{10}$$

where $\boldsymbol{A}_h \in \mathbb{R}^{D_m * D_r}$. Then, we compute the similarity score and importance probability using the bilinear form,

$$s_{i,j} = \boldsymbol{q}_{i,r}^T \cdot \boldsymbol{W}_a \cdot \boldsymbol{K}_j, \tag{11}$$

$$p_{i,j} = \frac{exp(s_{i,j})}{\sum_{\hat{j}=1}^{M} exp(s_{i,\hat{j}})}, \tag{12}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{D_r * D_m}$ is the model parameter to be learned and i, j are the indices of queries and memory slots.

As for the addressing step, it worths noting that the query and memory slots are both concatenated with temporal encodings. If we define the embedding layer $\boldsymbol{A}$ as the identity matrix, each similarity score of a query-memory pair can be divided into two parts,

$$s_{i,j} = \boldsymbol{q}_i^T \cdot \boldsymbol{o}_j + \lambda^2 \cdot \boldsymbol{PE(i)}^T \cdot \boldsymbol{PE(j)}. \quad (13)$$

Each query can automatically attend to instances with either close encoding representations or close temporal encodings. This tradeoff also accords with our intuition, since the confidence of a relational factual statement decreases when the time span increases.

Memory reading The value of each memory slot, which is also projected by an affine matrix $\boldsymbol{B} \in \mathbb{R}^{D_m * D_r}$, is read by computing the weighted sum over all memory slots with the importance probability derived in the addressing step:

$$\hat{\boldsymbol{q}}_i = \sum_j p_{i,j} \boldsymbol{V}_j, \quad (14)$$

where $\boldsymbol{V}_j = \boldsymbol{B}_h^T \cdot \boldsymbol{m}_j$.

Iterative computation Here, we combine the above two operations as a single step for reasoning. We use $h \in [1, H]$ to denote a particular step, where H is the total step number. To achieve a step-by-step reasoning, we update the next step query q^{h+1} with the summation of the current step output $\hat{q}^h$ and the current query q^h:

$$\boldsymbol{q}^{h+1} = \boldsymbol{q}^h + \hat{\boldsymbol{q}}^h. \quad (15)$$

During training, we add dropout with probability p_b at the final query step. By combining the previous hop query and the output in this way, TempMEM can gain information from the last read output and shift addressing attention to remote instances.

3.3.3 Output

In the output module, we define the conditional probability $P(r|S, \theta)$ through a softmax layer as follows:

$$P(r|S, \theta) = \frac{exp(\boldsymbol{R}_r^T \cdot \boldsymbol{q}_r^H)}{\sum_{\hat{r}=1}^{N_r} exp(\boldsymbol{R}_{\hat{r}}^T \cdot \boldsymbol{q}_r^H)}, \quad (16)$$

where N_r is the total amount of pre-defined relations. Also, since we construct and predict relations using the same relation embedding in the query construction part, we remove the original query from the last hop query q^H to keep the prediction unbiased.

3.4 Optimization

Here we introduce the learning and optimization details of TempMEM. We use query-level CrossEntropy loss as our objective function:

$$J(\theta) = \sum_{s=1}^{N_s} \sum_{i=1}^{T} y_t \cdot \log p(\hat{y}_t | S_s, \theta, t_i), \quad (17)$$

where N_s is the number of sets and T is the length of query sequence.

We use stochastic gradient descent (SGD) to minimize our objective function. For the exploration of optimization, we add small white noise to the gradients (Neelakantan et al., 2015). We also anneal the learning rate l by $0 < \rho < 1$ (i.e., $l \leftarrow \rho \cdot l$) for every τ epochs.

4 Experiment

4.1 Dataset

We evaluate our model on two datasets, the widely used NYT-10 dataset which is developed by (Riedel et al., 2010) and the WIKI-TIME dataset we created.

4.1.1 NYT-10

This dataset is generated by aligning Freebase entities to New York Times corpus (NYT) of years from 2005 to 2007. There are 53 pre-defined relations including a particular relation NA which indicates no relation between *head* and *tail*. The training data contains 522,611 sentences, 281,270 entity pairs, and 18,252 of them are relational facts. The testing data contains 172,448 sentences, 96,678 entity pairs, and 1,950 of them are relational facts.

4.1.2 WIKI-TIME

Similar to NYT-10, the WIKI-TIME dataset is also generated by aligning knowledge base entities to free corpus, except that we choose Wikidata and Wikipedia instead of Freebase and NYT news.

The motivation of creating WIKI-TIME is to generate a time aligned dataset that can support temporal reasoning. Hence, we filter knowledge base entities that participate in relations with informative temporal features, such as start time, end time. Besides, we tag the aligned sentences with their time expressions in contexts. Then, we align the contextual time expressions with the valid period of each relation to achieve labeling. For example, sentence like:

"On September 19, 2016, Jolie filed for divorce from Pitt, citing irreconcilable differences."

is labeled with no relation (NA).

The dataset contains 57 relations. The training set contains 97,616 sentences and 20,085 entity pairs. The test set contains 39,990 sentences and 8,641 entity pairs. [2]

4.2 Experiment Details

Hyper-Parameter Settings For WIKI-TIME experiments, we construct query over each appeared time spot in the mention set. On the other hand, for NYT-10 experiments, we adopt a single query without temporal encoding to compare results with other baseline methods since the dataset only contains one label for each mention set.

Among all experiments, we use 230 convolution kernels with windows size 3. The dropout probability p_d is set to 0.5. We try various max hops values H (from 1 to 5) to test how reasoning works in our model. We train the models with 20 epochs and 50 epochs for NYT-10 dataset and WIKI-TIME dataset and report the best performance.

As for optimization step, we adopt SGD with gradient plus Gaussian noise with standard deviation of 0.01, which helps to better generalize. Also, we apply gradient decay of rate $(\rho = 0.5)$ over every $\tau = 10$ epochs. The learning rates for NYT-10 and WIKI-TIME experiments are set to 0.001 and 0.01, respectively.

4.2.1 Input Vectors

With regard to inputs, we use 50-d Glove (Pennington et al., 2014) word embeddings pretrianed on Wikipedia and Gigaword and 5-d postion embedding. The temporal encodings are either initialized with random 50-d vectors, which are learned during training, or set directly with PE. For entity embeddings, we use the TransE (Bordes et al., 2013) entity vectors pretrained on Wikidata released by the OpenKE platform.

4.2.2 Evaluation Metric

The performance of comparative experiment is reported by precision-recall (PR) curve. Specifically, we sort the prediction scores of the model in descending order (without NA relation) and compute the precision with threshold for each recall value. Also, we report the $P@N$ values which indicate the precision over N predictions with the highest confidence scores.

Symbol	Remarks
CNN_ONE	(Zeng et al., 2015).
CNN_ATT	(Lin et al., 2016).
CNN_AVE	(Lin et al., 2016).
TempMEM	Our model without temporal encoding.
TempMEM+R	Our model with random intialized temporal encoding.
TempMEM+P	Our model with PE encoding.

Table 1: Notations.

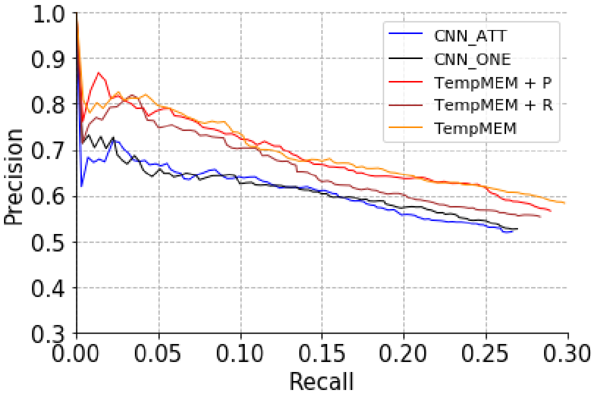


Figure 2: Precison-Recall curve of bag-level experiment on WIKI-TIME. Best viewed in color.

4.3 Performance of iterative reasoning

To test the effect of iterative reasoning over instances, we implement the neural models proposed in previous work (Zeng et al., 2015; Lin et al., 2016), from the source code released by authors. Since the previous models perform prediction in bag-level, the label is given by the latest relation appeared in KB. As for our models, we fix the number of hops $H = 2$ and set the encoding to CNN. [3] The notations of the experiments are shown in Table 1.

As shown in Figure 2 and Table 2, we have the following observations: (1) All TempMEM models achieve better performance compared with the previous neural models (CNN_ONE, CNN_ATT).

---

[2] The details of construction of WIKI-TIME can be found in *Appendix A*.

[3] The PCNN encoding is not used in WIKI-TIME dataset. The detailed explanation is given in *Appendix A*.

Method	P@N_100	P@N_200	P@N_300
CNN_ATT	67.33	67.66	66.45
CNN_ONE	70.3	68.66	65.78
TempMEM	81.18	**82.09**	**78.41**
TempMEM+R	79.21	78.61	75.42
TempMEM+P	**81.19**	79.1	77.41

Table 2: Comparison with previous models. P@N_100/200/300 refers to the precision for the highest 100, 200 and 300 predictions in WIKI-TIME.

Method	Bag-level F1	Query-level F1
CNN_ATT	39.66	-
CNN_ONE	40.15	-
TempMEM	47.88	54.75
TempMEM+R	46.76	47.83
TempMEM+P	**54.86**	**60.01**

Table 3: Manual evaluation of Bag-level and Query-level F1 scores in WIKI-TIME.

Recall that the hop number is set to 2. This can be seen as an ablation experiment. The results suggest that the remote instances can generally help relation extraction task. (2) TempMEM + P clearly outperforms TempMEM + R, which proves that the properly chosen temporal encodings help the performance.

Note that, in the columns "P@N_200" and "P@N_300" of Table 2, we find that the pure TempMEM outperforms TempMEM + P and TempMEM + R. Based on the results in Table 3, their drop of performance comes from the noisy labeling problem of distant supervision.

Also, TempMEM can catch relation changes through the timeline of two entities. We refer the readers to the case study in *Appendix B*.

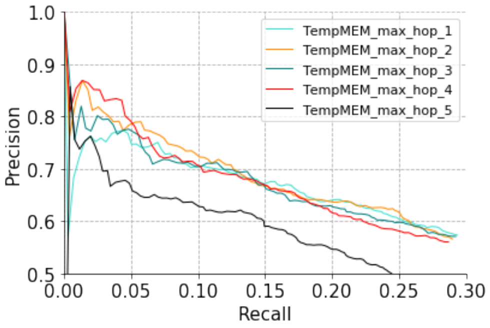


Figure 3: Number-of-hops experiment on WIKI-TIME. Best viewed in color.

4.4 Manual Evaluation on WIKI-TIME

Since the WIKI-TIME is distantly collected, we want to obtain a more precise view of how the models perform. So, we apply the manual evaluation to verify our experimental results. We randomly pick 200 mention sets in the test set of WIKI-TIME and ask two annotators to label the relation for each instance. The annotation rule is to label the instance with the relation that can be inferred from the instance itself or previous instances. As shown in Table 3, the manual evaluated F1 scores are basically consistent with the PR curves in Figure 2, which indirectly proves the WIKI-TIME's quality. Also, we find that the TempMEM + P achieves the best performance and shows obvious advantages in both query-level and bag-level F1 scores over the naive TempMEM (i.e., with no temporal encodings). This proves the effectiveness of our temporal encodings.

4.5 Effect of the Number of Hops

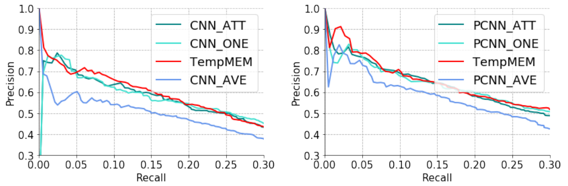
In this section, we discuss the effect of different number of hops in TempMEM. We change the hop value from 1 to 5 and evaluate the precision and recall of our models in query-level. The hyperparameters are fixed. The temporal encoding is set to PE and each model is trained for 30 epochs.

The results of the hop number experiment are depicted in Figure 3. From the results, we can observe that models show better performance with hop number 2 and 4. Most of the improvement of the model with hop number 4 resides in the recall range [0, 0.05], but the performance remains in the similar trend with other models in the recall range [0.05, 0.2]. In addition, we notice that the performance of the models fluctuates with the increase in the number of hops and the model with even hop number generally perform better than its predecessor, e.g. models with hop number 4 and 5. We believe that the reason might lie in the distribution of the hop distance between origin instance and useful remote instance.

4.6 Performance on NYT-10 Dataset

In this section, we report our results on the well-studied NYT-10 dataset. By evaluating our model in the NYT-10 dataset, our objective is to prove the power of reasoning among remote instances.

Note that, in the NYT-10 dataset, there is no temporal information for each instance, so we only use one query for each mention set and there's no



(a) Results with CNN encoding. (b) Results with PCNN encoding.

Figure 4: Precison-Recall curve on **NYT-10**. Best viewed in color.

temporal encoding for each instance. Also, we do not use the entity embedding for the NYT-10 experiments.

The results are shown in Figure 4. For both CNN and PCNN models, We can see that our models exceed the performance of all other models (CNN_ATT, CNN_ONE, CNN_AVE, PCNN_ATT, PCNN_ONE, PCNN_AVE) in the range of low recall values. In the high recall range, our models also have results about the same as the best model among others. This suggests that even without the temporal encoding, reasoning over remote instances is indeed useful in relation extraction task.

5 Related Work

5.1 Distant Supervision

Distant supervision for relation extraction is an important, automatic method of completing knowledge base.

(Riedel et al., 2010) made the at-least-once assumption that led the distant supervision for relation extraction to multi-instance learning. (Hoffmann et al., 2011) and (Surdeanu et al., 2012) tried to model the task with a multi-instance, multi-label setting using the classical graph model.

Recently, some work focused on applying deep neural network to the DS task. (Zeng et al., 2014) was the first trial to apply deep learning in relation extraction by solving a classification problem with fully supervised approach. (Zeng et al., 2015) moved a step further and introduced the multi-instance learning paradigm by using only the most important instance to predict relation. (Lin et al., 2016; Liu et al., 2017; Ji et al., 2017) improved the previous work by adding attention mechanism to instances and automatically reducing the weights of noisy instances. There are other approaches that tried to reduce the impact of noise in DS by using active learning (Sterckx et al., 2014) and reinforcement learning (Feng et al., 2018).

However, previous work focused on denoising but ignored the exploration of the remote instances and introduced no temporal information to support relation extraction. In this paper, we introduce temporal information into DS and combine it with the memory network to perform reasoning over instances.

(Feng et al., 2017) also used the memory network in the context of distant supervision. Their work performed word-level and relation-level reasoning to model the importance of words and dependency between relations. Their motivation was to gain better sentence encoding and relation modeling, while in our model, we apply the sentence-level memory network to understand the inference process among instances.

5.2 Temporal Relation Extraction

Also, this work is related to temporal relation extraction. (Dligach et al., 2017) was the first approach to use neural models for temporal relation extraction. (Tourille et al., 2017) used a Bi-LSTM to identify narrative containers between events and time expressions. (Cheng and Miyao, 2017) introduced dependency paths and used a "common-root" to solve the cross-sentence dependency. (Meng and Rumshisky, 2018) leveraged the Neural Turing Machine to enhance context-awareness of the temporal relation extraction model.

Previous work in temporal relation extraction was dedicated to event timelining and focused on dealing with relations between event and time expression. In constrast, our model aims to solve general entity to entity relation extraction by instance-level temporal reasoning based on a coarse-grained timelining.

5.3 Temporal Slot Filling

Another related research aspect is the temporal slot filling (TSF) task introduced in knowledge base population (Surdeanu, 2013; Ji et al., 2014). Distant supervision approaches (Garrido et al., 2013; Cucerzan and Sil, 2013; Sil and Cucerzan, 2014) (from Freebase and Wikipedia infoboxes) are widely applied to address the lack of supervising data. (Reinanda and De Rijke, 2014) performed the prior-sampling on distant supervision data to correct the mismatch of distributions.

The TSF task is similar to the task defined in this paper in the sense that TSF also asks the model to identify the start and end date of one knowledge triple $\langle head, rel, tail\rangle$. The difference is that the TSF task's objective is to predict validity period given the *head*, *rel* and *tail*, while in our setting, we predict *rel* between two entities in different periods.

6 Conclusion

In this paper, we formulate the task of distant supervision with temporal relation reasoning by modeling it as a sequence labeling problem. Following the DS paradigm, we created a new dataset called WIKI-TIME which is designed for the temporal relation extraction task. In addition, we propose an encoding-fusion model, TempMEM, which combines both encoding and reasoning temporally. At each computation step, our model can automatically attend information with either close representation or close temporal encoding. In experiments, we compare our model with the existing methods in both the well-known NYT-10 dataset and our WIKI-TIME dataset. Both automatic and manual evaluation are applied in the experiments. The experimental results show that our model not only realizes better performance in relation extraction by introducing instance-level reasoning but also improves the reasoning by bringing the temporal information in.

In the future, we plan to further explore the effect of different encoding modules like Bi-LSTM or self-attention and try to model temporal information with more sophisticated choices.

Acknowledgments

The research is supported in part by the National Basic Research Program of China Grant 2015CB358700, the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016, 61761146003, and a grant from Microsoft Research Asia.
Also, we want to thank the very kind comments from anonymous reviewers, which help improve this paper.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional lstm over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 1–6.

Silviu Cucerzan and Avirup Sil. 2013. The msr systems for entity linking and temporal slot filling at tac 2013. In *TAC*.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 746–751.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 19–25.

Guillermo Garrido, Anselmo Penas, and Bernardo Cabaleiro. 2013. Uned slot filling and temporal slot filling systems at tac kbp 2013: System description. In *TAC*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Heng Ji, Taylor Cassidy, Qi Li, and Suzanne Tamang. 2014. Tackling representation, annotation and classification challenges for temporal knowledge base population. *Knowledge and Information Systems*, 41(3):611–646.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: enhance distantly supervised relation extraction with dynamic transition matrix. *arXiv preprint arXiv:1705.03995*.

Yuanliang Meng and Anna Rumshisky. 2018. Context-aware neural model for temporal information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ridho Reinanda and Maarten De Rijke. 2014. Prior-informed distant supervision for temporal evidence classification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 996–1006.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Avirup Sil and Silviu-Petru Cucerzan. 2014. Towards temporal scoping of relational facts based on wikipedia data. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 109–118.

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2014. Using active learning and semantic clustering for noise reduction in distant supervision. In *4e Workshop on Automated Base Construction at NIPS2014 (AKBC-2014)*, pages 1–6.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A bi-lstm approach for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 224–230.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

A Appendix A : WIKI-TIME Construction

Here we illustrate the construction process of WIKI-TIME dataset in detail. The construction of WIKI-TIME follows common distant supervision framework. In distant supervision, we align the knowledge base to free corpus. The chosen knowledge base and free corpus are Wikidata and Wikipedia, respectively.

As depicted in Figure 5, the construction of WIKI-TIME consists of the following procedures.

1. We extract the relations with the valid period (i.e., $\langle t_s, t_e \rangle$) that appeared in WikiData.
2. Based on the extracted relation set, we further extract entity pairs that participate in any relation in the set. Here, each extracted triple should have a valid period (i.e., $\langle E_1, rel, E_2 \rangle : \langle t_s, t_e \rangle$).
3. We tag each sentence in Wikipedia corpus using either time expression appeared in

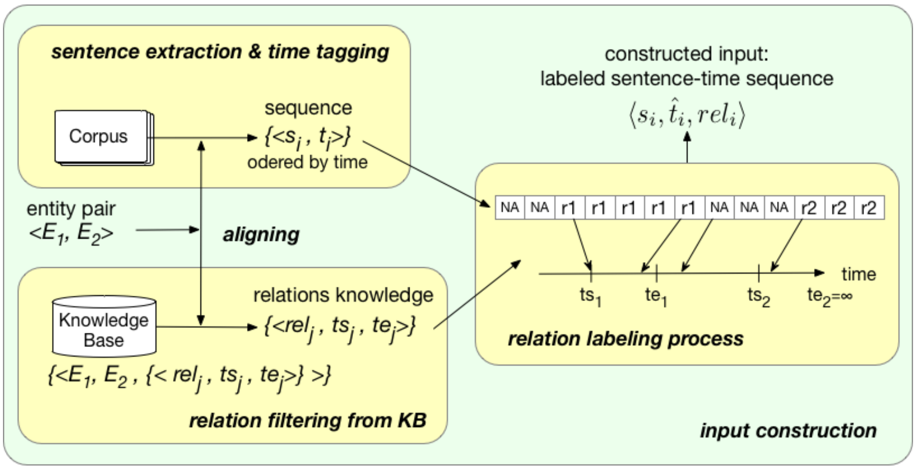


Figure 5: Preprocess of inputs

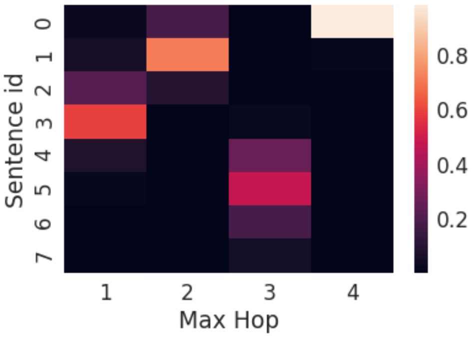
the sentence or last appeared time expression. We implement our own time expression recognition on the basis of NLTK-contrib's timex tool [4] with some modification according to our needs. Each tagged sentence is denoted by tuple $\langle s_i, \hat{t}_i \rangle$.

4. We align the extracted entity pairs to tagged corpus to retrieve our final mention set. Note that we additionally take sentences mentioning entity E_2 in the wiki page of entity E_1 as training sentences for (E_1, E_2) to obtain a larger dataset. In this case, the position of E_1 may not exist in our dataset. Because PCNN splits sentences by positions of entities, it is unclear how to apply it directly to our WIKI-TIME dataset.
5. We order the extracted sentences $\{\langle s_1, \hat{t}_1 \rangle, \cdots \langle s_m, \hat{t}_m \rangle\}$ into a timeline and align them with the corresponding knowledge triple for sentence-level relation $\langle s_i, \hat{t}_i, rel_i \rangle$.

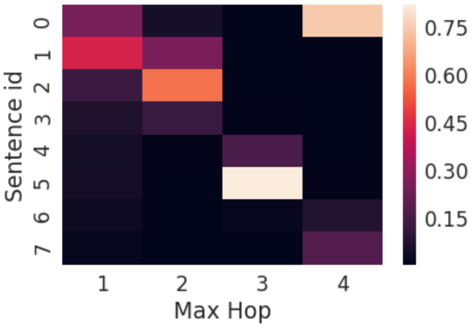
The final dataset contains 57 relations. The training set contains 97,616 sentences and 20,085 entity pairs. The test set contains 39,929 sentences and 8,641 entity pairs. The train/test split is done on entity pair level, so there is no overlap between train and test set.

B Appendix B : Case Study

Table 4 shows an example of our tagged data. For each sentence, we show the corresponding time slot and its relation. Also, the mentions of the entity are highlighted with the bold font and each sentence is indexed by *id* for clearness.



(a) q_3 over relation *Spouse*



(b) q_0 over relation *Spouse*

Figure 6: Attention weights for queries with $H = 4$

For each query, we show its corresponding heat map of attention values over different hops. The case we choose is $\langle Stelios \quad Kazantzidis, spouse, Marinella \rangle$. In Figure 6 (a), the attention of q_3 focuses on sentences with *id* 2 and 3 which support the relation *spouse* a lot. Then in the next hop, its attention shifts to sentences with *id* 0 and 1, which are the "remote" instances. Then in the third hop, its attention shifts again toward another part

---

[4] `https://github.com/nltk/nltk_contrib/blob/master/nltk_contrib/timex.py`

id	Relation	Time Spot	Sentence
0	NA	1957-01-01	Her early career was marked by her collaboration with singer **Stelios Kazantzidis**.
1	NA	1960-01-01	... instances of **Marinella** in films of Greek cinema, from the 1960 by 1966 with **Stelios Kazantzidis** ...
2	Spouse	1964-05-07	**Marinella** married **Stelios Kazantzidis** on 7 May 1964 ...
3	Spouse	1964-05-07	**Stelios Kazantzidis** married **Marinella** on 7 May 1964 ...
4	NA	1966-09-01	In September 1966 he divorced **Marinella**...
5	NA	1968-01-01	Following **Marinella**'s departure Litsa Diamandi ...
6	NA	1968-01-01	**Marinella** sang on some songs ...
7	NA	1968-01-01	**Marinella** had an ”answer back” to that latter song ...

Table 4: Aligned sentences of ⟨ Stelios Kazantzidis , Marinella ⟩

of the sentence set (*id* 4 to 6), which contains sentences related to divorce, departure and so on. These findings prove the claim that our model can achieve more accurate relation extraction by exploitation of remote instances.

Also, we demonstrate that the model has different attentions for queries at different time spots. In Figure 6 (b), we still observe the case of ⟨ *Stelios Kazantzidis, spouse, Marinella* ⟩. But, the attention of query q_0 focuses on a different path compared with query q_3. It first focuses on the remote instances with *id* 0 and 1. Then the model shifts to the supporting sentences. After that, it highly focuses on sentence 5, which supports two entities' departure. All these findings show that the temporal encodings significantly affect the result of attention.

The differences between the attentions of quries q_0 and q_3 show that the temporal encodings significantly affect the result of attention.

Integrating Semantic Knowledge to Tackle Zero-shot Text Classification

Jingqing Zhang *
Data Science Institute
Imperial College London
London, UK

Piyawat Lertvittayakumjorn *
Department of Computing
Imperial College London
London, UK

Yike Guo
Data Science Institute
Imperial College London
London, UK

`{jingqing.zhang15,pl1515,y.guo}@imperial.ac.uk`

Abstract

Insufficient or even unavailable training data of emerging classes is a big challenge of many classification tasks, including text classification. Recognising text documents of classes that have never been seen in the learning stage, so-called *zero-shot text classification*, is therefore difficult and only limited previous works tackled this problem. In this paper, we propose a two-phase framework together with data augmentation and feature augmentation to solve this problem. Four kinds of semantic knowledge (word embeddings, class descriptions, class hierarchy, and a general knowledge graph) are incorporated into the proposed framework to deal with instances of unseen classes effectively. Experimental results show that each and the combination of the two phases achieve the best overall accuracy compared with baselines and recent approaches in classifying real-world texts under the zero-shot scenario.

1 Introduction

As one of the most fundamental problems in machine learning, automatic classification has been widely studied in several domains. However, many approaches, proven to be effective in traditional classification tasks, cannot catch up with a dynamic and open environment where new classes can emerge after the learning stage (Romera-Paredes and Torr, 2015). For example, the number of topics on social media is growing rapidly, and the classification models are required to recognise the text of the new topics using only general information (e.g., descriptions of the topics) since labelled training instances are unfeasible to obtain for each new topic (Lee et al., 2011). This scenario holds in many real-world domains such as object recognition and medical diagnosis (Xian et al., 2017; World Health Organization, 1996).

Zero-shot learning (ZSL) for text classification aims to classify documents of classes which are absent from the learning stage. Although it is challenging for a machine to achieve, humans are able to learn new concepts by transferring knowledge from known to unknown domains based on high-level descriptions and semantic representations (Thrun and Pratt, 1998). Therefore, without labelled data of unseen classes, a zero-shot learning framework is expected to exploit supportive semantic knowledge (e.g., class descriptions, relations among classes, and external domain knowledge) to generally infer the features of unseen classes using patterns learned from seen classes.

So far, three main types of semantic knowledge have been employed in general zero-shot scenarios (Fu et al., 2018). The most widely used one is semantic attributes of classes such as visual concepts (e.g., colours, shapes) and semantic properties (e.g., behaviours, functions) (Lampert et al., 2009; Zhao et al., 2018). The second type is concept ontology, including class hierarchy and knowledge graphs, which represents relationships among classes and features (Wang et al., 2018; Fergus et al., 2010). The third type is semantic word embeddings which capture implicit relationships between words thanks to a large training text corpus (Socher et al., 2013; Norouzi et al., 2013). Nonetheless, concerning ZSL in text classification particularly, there are few studies exploiting one of these knowledge types and none has considered the combinations of them (Pushp and Srivastava, 2017; Dauphin et al., 2013). Moreover, some previous works used different datasets to train and test, but there is similarity between classes in the training and testing set. For example, in (Dauphin et al., 2013), the class "imdb.com" in the training set naturally corresponds to the class "Movies" in

---

* Piyawat Lertvittayakumjorn and Jingqing Zhang contributed equally to this project.

Proceedings of NAACL-HLT 2019, pages 1031–1040
Minneapolis, Minnesota, June 2 - June 7, 2019.

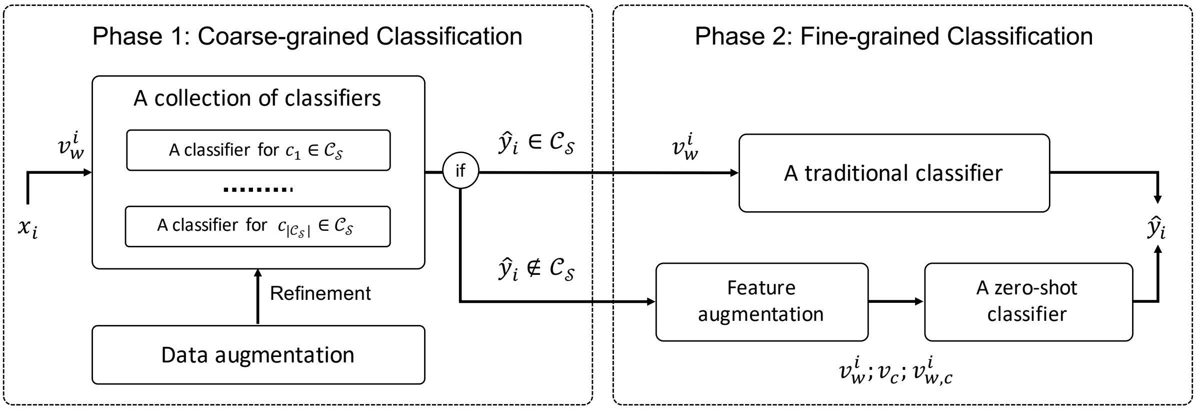


Figure 1: The overview of the proposed framework with two phases. The coarse-grained phase judges if an input document x_i comes from seen or unseen classes. The fine-grained phase finally decides the class $\hat{y}_i$. All notations are defined in section 2.1-2.2.

the testing set. Hence, these methods are not working under a strict zero-shot scenario.

To tackle the zero-shot text classification problem, this paper proposes a novel two-phase framework together with data augmentation and feature augmentation (Figure 1). In addition, four kinds of semantic knowledge including word embeddings, class descriptions, class hierarchy, and a general knowledge graph (ConceptNet) are exploited in the framework to effectively learn the unseen classes. Both of the two phases are based on convolutional neural networks (Kim, 2014). The first phase called **coarse-grained classification** judges if a document is from seen or unseen classes. Then, the second phase, named **fine-grained classification**, finally decides its class. Note that all the classifiers in this framework are trained using labelled data of seen classes (and augmented text data) only. None of the steps learns from the labelled data of unseen classes.

The contributions of our work can be summarised as follows.

- We propose a novel deep learning based two-phase framework, including coarse-grained and fine-grained classification, to tackle the zero-shot text classification problem. Unlike some previous works, our framework does not require semantic correspondence between classes in a training stage and classes in an inference stage. In other words, the seen and unseen classes can be clearly different.
- We propose a novel data augmentation technique called topic translation to strengthen the capability of our framework to detect documents from unseen classes effectively.
- We propose a method to perform feature augmentation by using integrated semantic knowledge to transfer the knowledge learned from seen to unseen classes in the zero-shot scenario.

In the remainder of this paper, we firstly explain our proposed zero-shot text classification framework in section 2. Experiments and results, which demonstrate the performance of our framework, are presented in section 3. Related works are discussed in section 4. Finally, section 5 concludes our work and mentions possible future work.

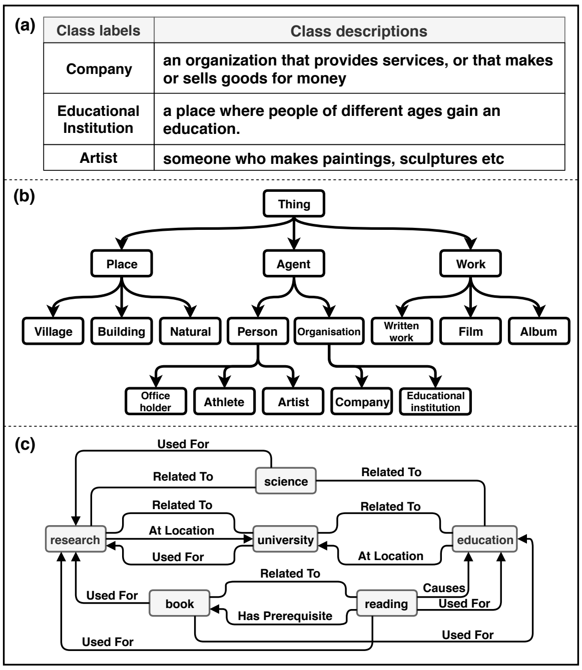
2 Methodology

2.1 Problem Formulation

Let $\mathcal{C}_S$ and $\mathcal{C}_U$ be disjoint sets of seen and unseen classes of the classification respectively. In the learning stage, a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is given where x_i is the i-th document containing a sequence of words $[w_1^i, w_2^i, \ldots, w_t^i]$ and $y_i \in \mathcal{C}_S$ is the class of x_i. In the inference stage, the goal is to predict the class of each document, $\hat{y}_i$, in a testing set which has the same data format as the training set except that y_i comes from $\mathcal{C}_S \cup \mathcal{C}_U$. Note that *(i)* every class comes with a class label and a class description (Figure 2a); *(ii)* a class hierarchy showing superclass-subclass relationships is also provided (Figure 2b); *(iii)* the documents from unseen classes cannot be observed to train the framework.

2.2 Overview and Notations

As discussed in the Introduction, our proposed classification framework consists of two phases



(a)

Class labels	Class descriptions
Company	an organization that provides services, or that makes or sells goods for money
Educational Institution	a place where people of different ages gain an education.
Artist	someone who makes paintings, sculptures etc

Figure 2: Illustrations of semantic knowledge integrated into our framework: (a) class labels and class descriptions (b) class hierarchy and (c) a subgraph of the general knowledge graph (ConceptNet).

(Figure 1). The first phase, coarse-grained classification, predicts whether an input document comes from seen or unseen classes. We also apply a data augmentation technique in this phase to help the classifiers be aware of the existence of unseen classes without accessing their real data. Then the second phase, fine-grained classification, finally specifies the class of the input document. It uses either a traditional classifier or a zero-shot classifier depending on the coarse-grained prediction given by Phase 1. Also, feature augmentation based on semantic knowledge is used to provide additional information which relates the document and the unseen classes to generalise the zero-shot reasoning.

We use the following notations in Figure 1 and throughout this paper.

- The list of embeddings of each word in the document x_i is denoted by $v_w^i = [v_{w_1}^i, v_{w_2}^i, \ldots, v_{w_t}^i]$.
- The embedding of each class label c is denoted by v_c, $\forall c \in \mathcal{C_S} \cup \mathcal{C_U}$. It is assumed that each class has a one-word class label. If the class label has more than one word, a similar one-word class label is provided to find v_c.
- As augmented features, the relationship vector $v_{w_j,c}^i$ shows the degree of relatedness between the word w_j and the class c according to semantic knowledge. Hence, the list of relationship vectors between each word in x_i and each class $c \in \mathcal{C_S} \cup \mathcal{C_U}$ is denoted by $v_{w,c}^i = [v_{w_1,c}^i, v_{w_2,c}^i, \ldots, v_{w_t,c}^i]$. We will explain the construction method in section 2.4.1.

2.3 Phase 1: Coarse-grained Classification

Given a document x_i, Phase 1 performs a binary classification to decide whether $\hat{y}_i \in \mathcal{C_S}$ or $\hat{y}_i \notin \mathcal{C_S}$. In this phase, each seen class $c_s \in \mathcal{C_S}$ has its own CNN classifier (with a subsequent dense layer and a sigmoid output) to predict the confidence that x_i comes from the class c_s, i.e., $p(\hat{y}_i = c_s | x_i)$. The classifier uses v_w^i as an input and it is trained using a binary cross entropy loss with all documents of its class in the training set as positive examples and the rest as negative examples.

For a test document x_i, this phase computes $p(\hat{y}_i = c_s | x_i)$ for every seen class c_s in $\mathcal{C_S}$. If there exists a class c_s such that $p(\hat{y}_i = c_s | x_i) > \tau_s$, it predicts $\hat{y}_i \in \mathcal{C_S}$; otherwise, $\hat{y}_i \notin \mathcal{C_S}$. τ_s is a classification threshold for the class c_s, calculated based on the threshold adaptation method from (Shu et al., 2017).

2.3.1 Data Augmentation

During the learning stage, the classifiers in Phase 1 use negative examples solely from seen classes, so they may not be able to differentiate the positive class from unseen classes. Hence, when the names of unseen classes are known in the inference stage, we try to introduce them to the classifiers in Phase 1 via augmented data so they can learn to reject the instances likely from unseen classes. We do data augmentation by translating a document from its original seen class to a new unseen class using analogy. We call this process *topic translation*.

In the word level, we translate a word w in a document of class c to a corresponding word w' in the context of a target class c' by solving an analogy question "c:w :: c':?". For example, solving the analogy "company:firm :: village:?" via word embeddings (Mikolov et al., 2013), we know that the word "firm" in a document of class "company" can be translated into the word "hamlet" in the context of class "village". Our framework adopts the 3CosMul method by Levy and Goldberg (2014) to solve the analogy question and find candidates of w':

$$w' = \underset{x \in V}{\operatorname{argmax}} \frac{\cos(x, c') \cos(x, w)}{\cos(x, c) + \epsilon}$$

where V is a vocabulary set and $\cos(a, b)$ is a cosine similarity score between the vectors of word a and word b. Also, ϵ is a small number (i.e., 0.001) added to prevent division by zero.

In the document level, we follow Algorithm 1 to translate a document of class c into the topic of another class c'. To explain, we translate all nouns, verbs, adjectives, and adverbs in the given document to the target class, word-by-word, using the word-level analogy. The word to replace must have the same part of speech as the original word and all the replacements in one document are 1-to-1 relations, enforced by replace_dict in Algorithm 1. With this idea, we can create augmented documents for the unseen classes by topic-translation from the documents of seen classes in the training dataset. After that, we can use the augmented documents as additional negative examples for all the CNNs in Phase 1 to make them aware of the tone of unseen classes.

Algorithm 1: Document-level topic translation

```
Input : a document $x_i = [w_1^i, w_2^i, \ldots, w_t^i]$,
        an original class label $c$, a target class label $c'$
Output: a translated document $x'_i$
replace_dict = dict(); $x'_i$ = [];
foreach $w \in x_i$ do
    if is_valid_pos($w$) then
        if $w \notin$ replace_dict then
            cands = solve_analogy($w$, $c$, $c'$,
              top_k=20);
            for $j$ = 0 to len(cands)-1 do
                if cands[$j$] $\notin$ replace_dict.values() $\wedge$
                  pos_of($w$) $\in$ pos_list(cands[$j$])
                  then
                    replace_dict[$w$] = cands[$j$];
                    break;
            if $j$ == len(cands) then
                $x'_i$.append($w$);
                continue;
        $x'_i$.append(replace_dict[$w$]);
    else
        $x'_i$.append($w$);
return $x'_i$
```

2.4 Phase 2: Fine-grained Classification

Phase 2 decides the most appropriate class $\hat{y}_i$ for x_i using two CNN classifiers: a traditional classifier and a zero-shot classifier as shown in Figure 1. If $\hat{y}_i \in \mathcal{C_S}$ predicted by Phase 1, the traditional classifier will finally select a class $c_s \in \mathcal{C_S}$ as $\hat{y}_i$. Otherwise, if $\hat{y}_i \notin \mathcal{C_S}$, the zero-shot classifier will be used to select a class $c_u \in \mathcal{C_U}$ as $\hat{y}_i$.

The traditional classifier and the zero-shot classifier have an identical CNN-based structure followed by two dense layers but their inputs and outputs are different. The traditional classifier is a multi-class classifier ($|\mathcal{C_S}|$ classes) with a softmax output, so it requires only the word embeddings v_w^i as an input. This classifier is trained using a cross entropy loss with a training dataset whose examples are from seen classes only.

In contrast, the zero-shot classifier is a binary classifier with a sigmoid output. Specifically, it takes a text document x_i and a class c as inputs and predicts the confidence $p(\hat{y}_i = c | x_i)$. However, in practice, we utilise v_w^i to represent x_i, v_c to represent the class c, and also augmented features $v_{w,c}^i$ to provide more information on how intimate the connections between words and the class c are. Altogether, for each word w_j, the classifier receives the concatenation of three vectors (i.e., $[v_{w_j}^i; v_c; v_{w_j,c}^i]$) as an input. This classifier is trained using a binary cross entropy loss with a training data from seen classes only, but we expect this classifier to work well on unseen classes thanks to the distinctive patterns of $v_{w,c}^i$ in positive examples of every class. This is how we transfer knowledge from seen to unseen classes in ZSL.

2.4.1 Feature Augmentation

The relationship vector $v_{w_j,c}$ contains augmented features we input to the zero-shot classifier. $v_{w_j,c}$ shows how the word w_j and the class c are related considering the relations in a general knowledge graph. In this work, we use ConceptNet providing general knowledge of natural language words and phrases (Speer and Havasi, 2013). A subgraph of ConceptNet is shown in Figure 2c as an illustration. Nodes in ConceptNet are words or phrases, while edges connecting two nodes show how they are related either syntactically or semantically.

We firstly represent a class c as three sets of nodes in ConceptNet by processing the class hierarchy, class label, and class description of c. (1) ***the_class_nodes*** is a set of nodes of the class label c and any tokens inside c if c has more than one word. (2) ***superclass_nodes*** is a set of nodes of all the superclasses of c according to the class hierarchy. (3) ***description_nodes*** is a set of nodes of all nouns in the description of the class c. For example, if c is the class "Educational Institution", according to Figure 2a-2b, the three sets of Con-

ceptNet nodes for this class are:
(1) educational_institution, educational, institution
(2) organization, agent
(3) place, people, ages, education.

To construct $v_{w_j,c}$, we consider whether the word w_j is connected to the members of the three sets above within K hops by particular types of relations or not[1]. For each of the three sets, we construct a vector with $3K+1$ dimensions.

- $v[0]=1$ if w_j is a node in that set; otherwise, $v[0]=0$.
- for $k=0,\dots,K-1$:
  - $v[3k+1]=1$ if there is a node in the set whose shortest path to w_j is $k+1$. Otherwise, $v[3k+1]=0$.
  - $v[3k+2]$ is the number of nodes in the set whose shortest path to w_j is $k+1$.
  - $v[3k+3]$ is $v[3k+2]$ divided by the total number of nodes in the set.

Thus, the vector associated to each set shows how w_j is semantically close to that set. Finally, we concatenate the constructed vectors from the three sets to become $v_{w_j,c}$ with $3\times(3K+1)$ dimensions.

3 Experiments

3.1 Datasets

We used two textual datasets for the experiments. The vocabulary size of each dataset was limited by 20,000 most frequent words and all numbers were excluded. (1) **DBpedia** ontology dataset (Zhang et al., 2015) includes 14 non-overlapping classes and textual data collected from Wikipedia. Each class has 40,000 training and 5,000 testing samples. (2) The **20newsgroups** dataset [2] has 20 topics each of which has approximately 1,000 documents. 70% of the documents of each class were randomly selected for training, and the remaining 30% were used as a testing set.

3.2 Implementation Details [3]

In our experiments, two different rates of unseen classes, 50% and 25%, were chosen and the corresponding sizes of $\mathcal{C}_\mathcal{S}$ and $\mathcal{C}_\mathcal{U}$ are shown in Table 1. For each dataset and each unseen rate, the random selection of ($\mathcal{C}_\mathcal{S}$, $\mathcal{C}_\mathcal{U}$) were repeated ten times and these ten groups were used by all the experiments with this setting for a fair comparison. All documents from $\mathcal{C}_\mathcal{U}$ were removed from the training set accordingly. Finally, the results from all the ten groups were averaged.

In Phase 1, the structure of each classifier was identical. The CNN layer had three filter sizes [3, 4, 5] with 400 filters for each filter size and the subsequent dense layer had 300 units. For data augmentation, we used gensim with an implementation of 3CosMul (Řehůřek and Sojka, 2010) to solve the word-level analogy (line 5 in Algorithm 1). Also, the numbers of augmented text documents per unseen class for every setting (if used) are indicated in Table 1. These numbers were set empirically considering the number of available training documents to be translated.

In Phase 2, the traditional classifier and the zero-shot classifier had the same structure, in which the CNN layer had three filter sizes [2, 4, 8] with 600 filters for each filter size and the two intermediate dense layers had 400 and 100 units respectively. For feature augmentation, the maximum path length K in ConceptNet was set to 3 to create the relationship vectors[4]. The DBpedia ontology[5] was used to construct a class hierarchy of the DBpedia dataset. The class hierarchy of the 20newsgroups dataset was constructed based on the namespaces initially provided by the dataset. Meanwhile, the classes descriptions of both datasets were picked from Macmillan Dictionary[6] as appropriate.

For both phases, we used 200-dim GloVe vectors[7] for word embeddings v_w and v_c (Pennington et al., 2014). All the deep neural networks were implemented with TensorLayer (Dong et al., 2017a) and TensorFlow (Abadi et al., 2016).

Dataset	Unseen rate	$\mid\mathcal{C}_\mathcal{S}\mid$	$\mid\mathcal{C}_\mathcal{U}\mid$	#Augmented docs per c_u
DBpedia (14 classes)	25%	11	3	12,000
	50%	7	7	8,000
20news (20 classes)	25%	15	5	4,000
	50%	10	10	3,000

Table 1: The rates of unseen classes and the numbers of augmented documents (per unseen class) in the experiments

---

[1] In this paper, we only consider the most common types of positive relations which are *RelatedTo*, *IsA*, *PartOf*, and *AtLocation*. They cover ~60% of all edges in ConceptNet.

[2] http://qwone.com/~jason/20Newsgroups/

[3] Code: https://github.com/JingqingZ/KG4ZeroShotText.

[4] Based on our observation, most of the related words stay within 3 hops from the class nodes in ConceptNet.

[5] http://mappings.dbpedia.org/server/ontology/classes/

[6] https://www.macmillandictionary.com/

[7] glove6B.zip in https://nlp.stanford.edu/projects/glove/

3.3 Baselines and Evaluation Metrics

We compared each phase and the overall framework with the following approaches and settings.

Phase 1: Proposed by (Shu et al., 2017), **DOC** is a state-of-the-art open-world text classification approach which classifies a new sample into a seen class or “reject” if the sample does not belong to any seen classes. The DOC uses a single CNN and a 1-vs-rest sigmoid output layer with threshold adjustment. Unlike DOC, the classifiers in the proposed Phase 1 work individually. However, for a fair comparison, we used DOC only as a binary classifier in this phase ($\hat{y}_i \in \mathcal{C}_S$ or $\hat{y}_i \notin \mathcal{C}_S$).

Phase 2: To see how well the augmented feature $v_{w,c}$ work in ZSL, we ran the zero-shot classifier with **different combinations of inputs**. Particularly, five combinations of v_w, v_c, and $v_{w,c}$ were tested with documents from unseen classes only (traditional ZSL).

The whole framework: (1) **Count-based model** selected the class whose label appears most frequently in the document as $\hat{y}_i$. (2) **Label similarity** (Sappadla et al., 2016) is an unsupervised approach which calculates the cosine similarity between the sum of word embeddings of each class label and the sum of word embeddings of every n-gram ($n = 1, 2, 3$) in the document. We adopted this approach to do single-label classification by predicting the class that got the highest similarity score among all classes. (3) **RNN AutoEncoder** was built based on a Seq2Seq model with LSTM (512 hidden units), and it was trained to encode documents and class labels onto the same latent space. The cosine similarity was applied to select a class label closest to the document on the latent space. (4) **RNN+FC** refers to the architecture 2 proposed in (Pushp and Srivastava, 2017). It used an RNN layer with LSTM (512 hidden units) followed by two dense layers with 400 and 100 units respectively. (5) **CNN+FC** replaced the RNN in the previous model with a CNN, which has the identical structure as the zero-shot classifier in Phase 2. Both RNN+FC and CNN+FC predicted the confidence $p(\hat{y}_i = c|x_i)$ given v_w and v_c. The class with the highest confidence was selected as $\hat{y}_i$.

For Phase 1, we used the accuracy for binary classification ($y, \hat{y}_i \in \mathcal{C}_S$ or $y, \hat{y}_i \notin \mathcal{C}_S$) as an evaluation metric. In contrast, for Phase 2 and the whole framework, we used the multi-class classification accuracy ($\hat{y}_i = y_i$) as a metric.

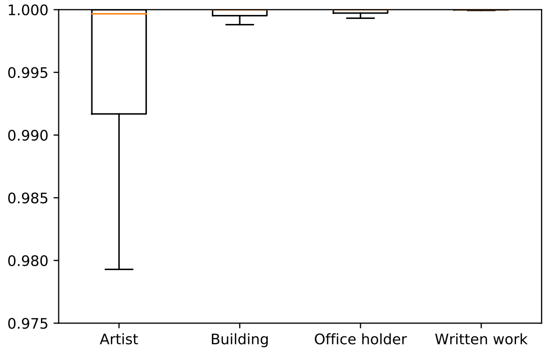


Figure 3: The distributions of confidence scores of positive examples from four seen classes of DBpedia in Phase 1.

3.4 Results and Discussion

The evaluation of Phase 1 (coarse-grained classification) checks if each x_i was correctly delivered to the right classifier in Phase 2. Table 3 shows the performance of Phase 1 with and without augmented data compared with DOC. Considering test documents from seen classes only, our framework outperformed DOC on both datasets. In addition, the augmented data improved the accuracy of detecting documents from unseen classes clearly and led to higher overall accuracy in every setting. Despite no real labelled data from unseen classes, the augmented data generated by topic translation helped Phase 1 better detect documents from unseen classes. Table 4 shows some examples of augmented data from the DBpedia dataset. Even if they are not completely understandable, they contain the tone of the target classes.

Although Phase 1 provided confidence scores for all seen classes, we could not use them to predict $\hat{y}_i$ directly since the distribution of scores of positive examples from different CNNs are different. Figure 3 shows that the distribution of confidence scores of the class “Artist” had a noticeably larger variance and was clearly different from the class “Building”. Hence, even if $p(\hat{y}_i =$ “Building”$|x_i) > p(\hat{y}_i =$ “Artist”$|x_i)$, we cannot conclude that x_i is more likely to come from the class “Building”. This is why **a traditional classifier in Phase 2 is necessary**.

Regarding Phase 2, fine-grained classification is in charge of predicting $\hat{y}_i$ and it employs two classifiers which were tested separately. Assuming Phase 1 is perfect, the classifiers in Phase 2 should be able to find the right class. The purpose of Table 5 is to show that **the traditional CNN classifier in Phase 2** was highly accurate.

Dataset	Unseen rate	y_i	Count-based	Label Similarity (Sappadla et al., 2016)	RNN Autoencoder	RNN + FC (Pushp and Srivastava, 2017)	CNN + FC	Ours
DBpedia	25%	seen	0.322	0.377	0.250	0.895	**0.985**	0.975
		unseen	0.372	**0.426**	0.267	0.046	0.204	0.402
		overall	0.334	0.386	0.254	0.713	0.818	**0.852**
	50%	seen	0.358	0.401	0.202	0.960	**0.991**	0.982
		unseen	0.304	**0.369**	0.259	0.044	0.069	0.197
		overall	0.333	0.386	0.230	0.502	0.530	**0.590**
20news	25%	seen	0.205	0.279	0.263	0.614	**0.792**	0.745
		unseen	0.201	**0.287**	0.149	0.065	0.134	0.280
		overall	0.204	0.280	0.236	0.482	**0.633**	**0.633**
	50%	seen	0.219	0.293	0.275	0.709	0.684	**0.767**
		unseen	0.196	**0.266**	0.126	0.052	0.126	0.168
		overall	0.207	0.280	0.200	0.381	0.405	**0.469**

Table 2: The accuracy of the whole framework compared with the baselines.

Dataset Unseen rate	y_i	DOC	Ours w/o aug.	Ours w/ aug.
DBpedia 25%	seen	0.980	**0.982**	**0.982**
	unseen	0.471	0.388	**0.536**
	overall	0.871	0.855	**0.886**
DBpedia 50%	seen	0.983	0.986	**0.987**
	unseen	0.384	0.345	**0.512**
	overall	0.684	0.666	**0.749**
20news 25%	seen	0.800	**0.838**	0.831
	unseen	0.573	0.431	**0.577**
	overall	0.745	0.754	**0.770**
20news 50%	seen	0.824	**0.856**	0.843
	unseen	0.562	0.419	**0.603**
	overall	0.694	0.639	**0.724**

Table 3: The accuracy of Phase 1 with and without augmented data compared with DOC .

Animal (Original)	Mitra perdulca is a species of sea snail a marine gastropod mollusk in the family Mitridae the miters or miter snails.
Animal → Plant	Arecaceae perdulca is a flowering of port aster a naval mollusk gastropod in the fabaceae Clusiaceae the tiliaceae or rockery amaryllis.
Animal → Athlete	Mira perdulca is a swimmer of sailing sprinter an Olympian limpets gastropod in the basketball Middy the miters or miter skater.

Table 4: Examples of augmented data translated from a document of the original class "Animal" into two target classes "Plant" and "Athlete".

Dataset	DBpedia		20news	
Input \ Unseen rate	50%	25%	50%	25%
v_w	0.993	0.992	0.878	0.861

Table 5: The accuracy of the traditional classifier in Phase 2 given documents from seen classes only.

Dataset	DBpedia		20news	
Inputs \ Unseen rate	50%	25%	50%	25%
Random guess	0.143	0.333	0.100	0.200
$v_{w,c}$	0.154	0.443	0.104	0.210
$[v_c; v_{w,c}]$	0.163	0.400	0.099	0.215
$[v_w; v_{w,c}]$	0.266	0.460	0.122	0.307
$[v_w; v_c]$	0.381	0.711	0.274	0.431
$[v_w; v_c; v_{w,c}]$	**0.418**	**0.754**	**0.302**	**0.500**

Table 6: The accuracy of the zero-shot classifier in Phase 2 given documents from unseen classes only.

Besides, given test documents from unseen classes only, the performance of **the zero-shot classifier in Phase 2** is shown in Table 6. Based on the construction method, $v_{w,c}$ quantified the relatedness between words and the class but, unlike v_w and v_c, it did not include detailed semantic meaning. Thus, the classifier using $v_{w,c}$ only could not find out the correct unseen class and neither $[v_w; v_{w,c}]$ and $[v_c; v_{w,c}]$ could do. On the other hand, the combination of $[v_w; v_c]$, which included semantic embeddings of both words and the class label, increased the accuracy of predicting unseen classes clearly. However, the zero-shot classifier fed by the combination of all three types of inputs $[v_w; v_c; v_{w,c}]$ achieved the highest accuracy in all settings. It asserts that the integration of semantic knowledge we proposed is an effective means for knowledge transfer from seen to unseen classes in the zero-shot scenario.

Last but most importantly, we compared **the whole framework** with four baselines as shown in Table 2. First, the count-based model is a rule-based model so it failed to predict documents from seen classes accurately and resulted in unpleasant overall results. This was similar to the label similarity approach even though it had higher degree of flexibility. Next, the RNN Autoencoder was trained without any supervision since $\hat{y}_i$ was predicted based on the cosine similarity. We believe the implicit semantic relatedness between classes caused the failure of the RNN Autoencoder. Besides, the CNN+FC and RNN+FC had same inputs and outputs and it was clear that CNN+FC performed better than RNN+FC in the experiment. However, neither CNN+FC nor RNN+FC was able to transfer the knowledge learned from seen to unseen classes. Finally, our two-phase framework has competitive prediction accuracy on unseen classes while maintaining the accuracy on seen classes. This made it achieve the highest overall accuracy on both datasets and both unseen rates. In conclusion, by using integrated semantic knowledge, the proposed two-phase framework with data and feature augmentation is a promising step to tackle this challenging zero-shot problem.

Furthermore, another benefit of the framework is high flexibility. As the modules in Figure 1 has less coupling to one another, it is flexible to improve or customise each of them. For example, we can deploy an advanced language understanding model, e.g., BERT (Devlin et al., 2018), as a traditional classifier. Moreover, we may replace ConceptNet with a domain-specific knowledge graph to deal with medical texts.

4 Related Work

4.1 Zero-shot Text Classification

There are a few more related works to discuss besides recent approaches we compared with in the experiments (explained in section 3.3). Dauphin et al. (2013) predicted semantic utterance of texts by mapping class labels and text samples into the same semantic space and classifying each sample to the closest class label. Nam et al. (2016) learned the embeddings of classes, documents, and words jointly in the learning stage. Hence, it can perform well in domain-specific classification, but this is possible only with a large amount of training data. Overall, most of the previous works exploited semantic relationships between classes and documents via embeddings. In contrast, our proposed framework leverages not only the word embeddings but also other semantic knowledge. While word embeddings are used to solve analogy for data augmentation in Phase 1, the other semantic knowledge sources (in Figure 2) are integrated into relationship vectors and used as augmented features in Phase 2. Furthermore, our framework does not require any semantic correspondences between seen and unseen classes.

4.2 Data Augmentation in NLP

In the face of insufficient data, data augmentation has been widely used to improve generalisation of deep neural networks especially in computer vision (Krizhevsky et al., 2012) and multimodality (Dong et al., 2017b), but it is still not a common practice in natural language processing. Recent works have explored data augmentation in NLP tasks such as machine translation and text classification (Saito et al., 2017; Fadaee et al., 2017; Kobayashi, 2018), and the algorithms were designed to preserve semantic meaning of an original document by using synonyms (Zhang and LeCun, 2015) or adding noises (Xie et al., 2017), for example. In contrast, our proposed data augmentation technique translates a document from one meaning (its original class) to another meaning (an unseen class) by analogy in order to substitute unavailable labelled data of the unseen class.

4.3 Feature Augmentation in NLP

Apart from improving classification accuracy, feature augmentation is also used in domain adaptation to transfer knowledge between a source and a target domain (Pan et al., 2010b; Fang and Chiang, 2018; Chen et al., 2018). An early research paper applying feature augmentation in NLP is Daume III (2007) which targeted domain adaptation on sequence labelling tasks. After that, feature augmentation was used in several NLP tasks such as cross-domain sentiment classification (Pan et al., 2010a), multi-domain machine translation (Clark et al., 2012), semantic argument classification (Batubara et al., 2018), etc. Our work is different from previous works not only that we applied this technique to zero-shot text classification but also that we integrated many types of semantic knowledge to create the augmented features.

5 Conclusion and Future Work

To tackle zero-shot text classification, we proposed a novel CNN-based two-phase framework together with data augmentation and feature augmentation. The experiments show that data augmentation by topic translation improved the accuracy in detecting instances from unseen classes, while feature augmentation enabled knowledge transfer from seen to unseen classes for zero-shot learning. Thanks to the framework and the integrated semantic knowledge, our work achieved the highest overall accuracy compared with all the baselines and recent approaches in all settings. In the future, we plan to extend our framework to do multi-label classification with a larger amount of data, and also study how semantic units defined by linguists can be used in the zero-shot scenario.

Acknowledgments

We would like to thank Douglas McIlwraith, Nontawat Charoenphakdee, and three anonymous reviewers for helpful suggestions. Jingqing and Piyawat would also like to thank the support from the LexisNexis® Risk Solutions HPCC Systems® academic program and Anandamahidol Foundation, respectively.

References

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Dina Khaira Batubara, Moch Arif Bijaksana, and Adiwijaya. 2018. On feature augmentation for semantic argument classification of the quran english translation using support vector machine. *Journal of Physics: Conference Series*, 971(1):012043.

Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. 2018. Semantic feature augmentation in few-shot learning. *CoRR*, abs/1804.05298.

Jonathan H Clark, Alon Lavie, and Chris Dyer. 2012. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics.

Yann N Dauphin, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2013. Zero-shot learning for semantic utterance classification. *arXiv preprint arXiv:1401.0509*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. 2017a. Tensorlayer: a versatile library for efficient deep learning development. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1201–1204. ACM.

Hao Dong, Jingqing Zhang, Douglas McIlwraith, and Yike Guo. 2017b. I2t2i: Learning text to image synthesis with textual data augmentation. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 2015–2019. IEEE.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. *arXiv preprint arXiv:1705.00440*.

Wen-Chieh Fang and Yi-Ting Chiang. 2018. A discriminative feature mapping approach to heterogeneous domain adaptation. *Pattern Recognition Letters*, 106:13–19.

Rob Fergus, Hector Bernal, Yair Weiss, and Antonio Torralba. 2010. Semantic label sharing for learning with many categories. In *Computer Vision – ECCV 2010*, pages 762–775, Berlin, Heidelberg. Springer Berlin Heidelberg.

Y. Fu, T. Xiang, Y. Jiang, X. Xue, L. Sigal, and S. Gong. 2018. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 35(1):112–125.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE.

Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. 2011. Twitter trending topic classification. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 251–258. IEEE.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010a. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Sinno Jialin Pan, Qiang Yang, et al. 2010b. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. 2017. Train once, test anywhere: Zero-shot learning for text classification. *arXiv preprint arXiv:1712.05972*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161.

Itsumi Saito, Jun Suzuki, Kyosuke Nishida, Kugatsu Sadamitsu, Satoshi Kobashikawa, Ryo Masumura, Yuji Matsumoto, and Junji Tomita. 2017. Improving neural text normalization with data augmentation at character-and morphological levels. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 257–262.

Prateek Veeranna Sappadla, Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *Proc. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Lei Shu, Hu Xu, and Bing Liu. 2017. Doc: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916. Association for Computational Linguistics.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 935–943, USA. Curran Associates Inc.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The Peoples Web Meets NLP*, pages 161–176. Springer.

Sebastian Thrun and Lorien Pratt. 1998. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer.

Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6857–6866.

World Health Organization. 1996. Infectious diseases kill over 17 million people a year. *Malaria Weekly*, 3:11–16.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *arXiv preprint arXiv:1707.00600*.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Bo Zhao, Yanwei Fu, Rui Liang, Jiahong Wu, Yonggang Wang, and Yizhou Wang. 2018. A large-scale attribute dataset for zero-shot learning. *arXiv preprint arXiv:1804.04314*.

Word-Node2Vec: Improving Word Embedding with Document-Level Non-Local Word Co-occurrences

Procheta Sen[1]
procheta.sen2@mail.dcu.ie

Debasis Ganguly[2]
debasga1@ie.ibm.com

Gareth J.F. Jones[1]
Gareth.Jones@dcu.ie

[1] ADAPT Centre, School of Computing, Dublin City University, Dublin, Ireland
[2] IBM Research, Dublin, Ireland

Abstract

Standard word embedding algorithms, such as `word2vec` and `Glove`, make a restrictive assumption that words are likely to be semantically related only if they co-occur locally within a window of fixed size. However, this restrictive assumption may not capture the semantic association between words that co-occur frequently but non-locally within documents. To alleviate this restriction, in this paper, we propose a graph-based word embedding method, named 'word-node2vec'. By relaxing the strong constraint of locality, our method is able to capture both local and non-local co-occurrences. Word-node2vec constructs a weighted graph, where each node represents a word and the weight of an edge between two nodes represents a combination of both local (e.g. word2vec) and document-level co-occurrences. Our experiments show that word-node2vec outperforms word2vec and glove on a range of different tasks, such as word-pair similarity prediction, word analogy and concept categorization.

1 Introduction

Word embedding, the process of obtaining vector representations of words, is a first step towards addressing language semantics, in which discrete entities, such as words, are embedded as vectors over a continuous space of reals. This not only facilitates to obtain semantic similarities between words to improve tasks such as semantic search (Ganguly et al., 2015; Roy et al., 2016), but is also useful in a number of down-stream NLP tasks including concept categorization (Jastrzebski et al., 2017), information retrieval (Guo et al., 2016), sentence similarity prediction (Mueller and Thyagarajan, 2016), sentiment analysis (Faruqui et al., 2015) and POS tagging (Tsvetkov et al., 2016) etc.

Word embedding approaches such as `word2vec` (Mikolov et al., 2013a) and `Glove` (Pennington et al., 2014) rely on a large corpus to learn the association between words. The architecture of existing word embedding approaches mimics the process of human cognition of word association by learning the representation of each word with an objective of maximizing the likelihood of predicting the words around its *local* context (defined by a fixed length word window). A limitation of existing word embedding approaches, such as word2vec and glove, is that they use a strong constraint that words are likely to be semantically related to each other only if one occurs within a local context of the another, where the local context is given by a word window of specified length.

On the other hand, non-local or document-level co-occurrences between words have been widely used to estimate semantic similarities between words. More specifically, the latent semantic analysis (LSA) method proposed by Deerwester et al. (1990) uses a spectral analysis (method of principal component analysis) of the term-document matrix of a collection to obtain the most informative *concepts* (word classes), and then expresses each document as a linear combination of these principal components. Blei et al. (2003) estimate a generative model from a given collection by assuming that documents are mixtures of a preset number of topics, where each topic represents a word distribution over the vocabulary. This is largely similar to decomposing a term-document matrix as a product of matrices with non-negative components, a process commonly known as non-negative matrix factorization (NMF) (Gaussier and Goutte, 2005). The underlying common idea among all these approaches is to make use of the frequent document-level word co-occurrences to identify likely semantic association between words.

Despite the presence of a vast volume of lit-

Proceedings of NAACL-HLT 2019, pages 1041–1051
Minneapolis, Minnesota, June 2 - June 7, 2019.

erature on document-level (non-local) word co-occurrences, word embedding approaches do not utilize this information to derive the word representations. In this paper, we propose to augment the document-level non-local word co-occurrence information with the local co-occurrence information that methods such as word2vec and glove use. More specifically, we propose a graph-based word embedding method, named `word-node2vec`, that by relaxing the strong constraint of locality, is able to capture both the local and non-local co-occurrences. To represent the local dependencies, each node, representative of a word (hence the name 'word-node'), is initialized with a vector representation obtained with a standard method, e.g. word2vec. We then define the weight of the edge between a pair of word-nodes to reflect their likelihood of non-local co-occurrence, computed with the help of the global term-document matrix for the whole collection.

The rest of the paper is organized as follows. In Section 2, we survey existing literature on word embedding. In Section 3, we revisit the skip-gram approach and propose a graph-based view of the skip-gram objective as a pre-cursor to developing our model. In Section 4, we extend the skip-gram graph model with non-local document-level co-occurrence information. Section 5 describes our experimental setup. Section 6 reports the results of our new embedding approach against a number of baselines. Finally, Section 7 concludes the paper with directions for future work.

2 Related Work

The word2vec (Mikolov et al., 2013a) embedding model shifts a window of a predefined size (a parameter) across the text of a collection of documents in order to train a linear classifier for each word to predict itself given its context (continuous bag-of-words), or its context given the word (skip-gram). The parameter vector transforming a word to its context (or vice-versa) gives its embedded representation. In addition to making use of the words in the context as positive samples, word2vec also relies on the use of words randomly sampled from the collection (outside the current context) as negative examples. Levy and Goldberg (2014) showed that the negative sampling based skip-gram (SGNS) objective function of `word2vec` is mathematically equivalent to factorizing a positive point-wise mutual information gain (PPMI) matrix shifted by $log(k)$, where k is the number of negative samples.

The key idea behind the glove algorithm proposed in (Pennington et al., 2014) is to make use of the ratio of the co-occurrence probabilities between word pairs to better distinguish semantically related words from non-related ones. The study ultimately shows that factorizing the log of the co-occurrence matrix leads to effective embedded representation of words. The co-occurrences in both word2vec and glove are essentially local in nature. In contrast, our proposed algorithm leverages both local and non-local co-occurrences.

More recently, Peters et al. (2018) proposed `ELMO`, a deep contextualized word representation with layers of stacked bi-directional LSTMs to model both a) complex characteristics of word use (e.g., syntax and semantics), and b) their diversity across various linguistic contexts. A limitation of ELMO is that a word representation may effectively be learned mainly in the presence of an associated context, as a result of which the method is likely to find applications mostly in downstream tasks, e.g. question answering and sentiment analysis. However, in contrast, our proposed method can learn the representation of a word in isolation, which means that, similar to word2vec and Glove, word vectors obtained using our method can be applied directly to (and is also likely to work well for) word similarity and word analogy tasks. We included ELMO as of our baseline approaches in our experiments.

Grover and Leskovec (2016) proposed a skip-gram based objective function to embed each node of a graph. Analogous to skip-gram based word embedding, each node vector is given as input to a linear classifier to predict the context vector around a node. The context vector around a node, in this case, consists of a sequence of nodes visited by a random walk starting from that node. In our method, we use a similar graph-based construction to train vector representations of a node (each node a word). However, we use a stratified sampling approach within a maximum distance (hop-count) of 2, instead of allowing the random walk to proceed along in a combined depth-first and breadth-first manner, as in (Grover and Leskovec, 2016). Through our experiments, we find that larger hop-counts (i.e. longer transitive dependencies) introduce noise in the document-level word co-occurrence estimation process.

3 Generalized Word Embedding

In this section, we propose a general word embedding framework based on the skip-gram objective function of word2vec. Our proposed method relies on a general construction of the *context* around a word. We modify the skip-gram objective function of word2vec to take into account this general context of words. Before describing our proposed approach, we revisit the objective function of negative sampling based skip-gram word2vec (SGNS).

Skip-gram. In word2vec, the context of a word comprises words occurring within a window of a fixed size (say k) pivoted at a particular instance of w in the collection. More formally, let $\Lambda(w)$ denote the set of indexes where the word w occurs in a collection $C = \{t_1, \ldots, t_T\}$, T denoting the total number of tokens in the collection C, i.e.

$$\Lambda(w) = \{i : t_i = w\}. \tag{1}$$

We then construct the context $c(w)$ of a word as

$$c(w) = \cup_{i \in \Lambda(w)} \cup_{\substack{j=-k \\ j \neq 0}}^{k} t_{i+j} \tag{2}$$

Let Ω denote the set of all observed word-context pairs $(w, c(w))$, i.e.

$$\Omega^{+} = \cup_{w \in V} \{w, c(w)\}, \tag{3}$$

where V denotes the vocabulary set, and Ω^{-} denote the set of negative samples of word-context pairs, i.e.

$$\Omega^{-} = \cup_{w \in V} \{w, \cup\{v : v \sim (V - c(w))\}\}, \tag{4}$$

where words v's in the negative context set are randomly sampled from the complement set $c(w)$.

Let y be an indicator random variable denoting semantic relatedness of a word with its context. For a word w and its context $c(w)$ (as defined in Equation 2, the SGNS algorithm seeks to maximize the objective function

$$\begin{aligned} J(\theta) = & \sum_{w,c(w) \in \Omega^{+}} p(y = 1 | \mathbf{w}, \mathbf{c_w}) + \\ & \sum_{w,c(w) \in \Omega^{-}} p(y = 0 | \mathbf{w}, \mathbf{c_w})), \end{aligned} \tag{5}$$

where $p(.)$ is the log-likelihood function, and $\theta \in \mathbb{R}^{d \times |V|}$ represents the trainable matrix of parameters, each d dimensional column vector of the matrix θ denoting the vector representation of word w, i.e. $\mathbf{w} = \theta_w$. Note that the vector for a set of context words $c(w)$ is obtained by some aggregation function (sum or average) over the constituent words, i.e.

$$\mathbf{c}(\mathbf{w}) = \sum_{u \in c(w)} \mathbf{u}. \tag{6}$$

In order to optimize $J(\theta)$, the word2vec approach shifts a window of size k pivoted around a word $w = t_i$ (token positioned at offset i in the corpus), and applies stochastic gradient descent (SGD) to update the parameters for the corresponding word w and its context vector $c(w)$.

A Graph Formulation of SGNS. We now propose a general framework that allows contexts to be defined in a more general way. The solution relies on defining a graph $G = (\mathcal{V}, \mathcal{E})$, where each node corresponds to a word from the vocabulary of the given collection, i.e.

$$\mathcal{V} = \{x_w : w \in V\}. \tag{7}$$

In general, an edge $(x_u, x_v) \in \mathcal{E}$ represents a *relation* between two words u and v of weight $w(x_u, x_v) \in \mathbb{R}$. For example, in order to define the context of SGNS (Equation 2), the edge set is defined as

$$\mathcal{E} = \{(x_w, x_u) : u \in \cup_{i \in \Lambda(w)} \cup_{\substack{j=-k \\ j \neq 0}}^{k} t_{i+j}\}. \tag{8}$$

Learning the vector representations for each node of the graph G leads to learning the vector representation for each word, because there is a one-one mapping between the set of nodes $\mathcal{V}$ and the set of words V (henceforth we refer to a node of this general class of graphs, defined as per Equation 7, as a *word-node*). The objective of the embedding is to learn vector representations of nodes such that two nodes are close in the embedded space if, as per the edge relations of the graph, these nodes are within a κ-adjacency neighborhood of each other. The κ-adjacency neighborhood of a graph is the set

$$N_\kappa(x_w) = \{x_u \in \mathcal{V} : h(x_w, x_u) \leq \kappa\}, \tag{9}$$

where $h(u, v)$ denotes the hop-count or adjacency number between nodes u and v. In the general formulation, the set of $N_\kappa(x_w)$, constituting the set of nodes reachable from paths of length at most k starting at x_w, act as positive examples to learn the embedding of node x_w. This is because these positive examples seek to make the vector representation of x_w similar to the vector representations of nodes in $N_\kappa(x_w)$. More formally,

$$\begin{aligned}\Omega^+ &= \cup_{x_w \in \mathcal{V}}\{x_w, N_\kappa(x_w)\},\\ \Omega^- &= \cup_{x_w \in \mathcal{V}}\{x_w, \cup\{x_u : u \sim \mathcal{V} - N_\kappa(x_w)\}\}.\end{aligned} \tag{10}$$

Instead of iterating over the words in a corpus, the SGNS equivalent is then achieved by iterating over the set of nodes and maximizing the same objective function of Equation 5 using the definitions of the positive and negative example sets from Equation 10. Note that to achieve the SGNS objective the value of κ is set to 1 in the definition of Ω^+ in Equation 10, i.e. the set of context for a word-node comprises one-hop neighbours as defined by the edge relations of Equation 8.

4 Extending the Graph Model for Non-Local Co-occurrences

The graph based approach of Section 3 allows alternative ways to define the context and learn the objective function to obtain word-node representations. In this section, we describe how to augment the non-local document-level co-occurrence information in the graph-based framework.

Co-occurrence Weights. The first step to include non-local co-occurrences is to modify the edge relations of SGNS (Equation 8) to accommodate weighted document-level co-occurrences. Instead of considering the collection $C = \{t_1, \ldots, t_T\}$ as a stream of words, we consider C as a set of M documents $\{D_i\}_{i=1}^M$.

First, we make provision to include weighted edges of the form $(x_w, x_u, \omega(x_w, x_u))$ in the edge construction process of Equation 8. The weight $\omega(x_w, x_u)$ between word-nodes x_w and x_u is intended to represent a measure of association between these words.

Next, we describe how to compute the non-local co-occurrence weight between a pair of words. First, we compute the co-occurrence probability of two words w and u as

$$P(w,u) = \frac{\sum_{i=1}^M \mathbb{I}(w,u,D_i)}{\sum_{i=1}^M \mathbb{I}(w,D_i)\sum_{i=1}^M \mathbb{I}(u,D_i)}, \tag{11}$$

where the numerator denotes the total number of times that the words w and u co-occur in the collection of all documents, and the denominator denotes the number of times each occur independently. In our approach, we use a generalized form of Equation 11, where analogous to the Jelinek-Mercer smoothing method (Ponte and Croft, 1998), we take into account the informativeness of the co-occurrences by linearly combining the frequencies with the global statistics of inverse collection frequency. More specifically,

$$P_\alpha(w,u) = \alpha P(w,u) + \frac{(1-\alpha)T^2}{|\Lambda(w)||\Lambda(u)|}, \tag{12}$$

where $P(w,u)$ represents the maximum likelihood estimate computed by Equation 11 and the denominator denotes the product of the collection frequencies of the terms (as per the notation of Equation 1). It can be seen that Equation 12 allows relative weighting of the term frequency and the informativeness components.

Combination with Local Co-occurrences. The next step in our word-node2vec method is to augment the non-local co-occurrence information computed as per Equation 12 with the local co-occurrence of SGNS as defined in Equation 8. For this, analogous to (Pennington et al., 2014), we compute the probability of co-occurrence between a word pair restricted within a window of size k over the whole collection. More formally,

$$P_k(w,u) = \frac{1}{|\Lambda(w)|}\sum_{i\in\Lambda(w)} \mathbb{I}(t_{i+j} = u)_{j=-k}^{k} \tag{13}$$

Next, we assign weight to an edge by combining the local and non-local co-occurrence probabilities estimated from Equations 13 and 12 respectively. Formally speaking,

$$\omega(x_w, x_u) = P_\alpha(w,u)P_k(w,u). \tag{14}$$

Context with Weighted Edges. Constructing the context of a node x_w (Section 3), requires a modification aimed to take into account the edge weights while selecting the neighboring nodes of x_w. Instead of defining the context as the entire set of κ-neighborhood $N_\kappa(x_w)$ of a node x_w, we define a κ-neighbourhood of length (hop-count), l, which is a subset of l samples drawn from the overall neighbourhood.

The likelihood of sampling a node x_u from the neighbourhood set is proportional to the weight of the edge (x_w, x_u), i.e., $\omega(x_w, x_u)$. This way of defining the context allows the algorithm to make use of the edge weights (local and non-local co-occurrences) in learning the node representations, i.e. assigning more importance to associations with higher weights in seeking to embed the current word-node close to them.

Our idea, in general, is to use stratified sampling, where each stratum corresponds to a neighbourhood of particular length. The priors assigned to the strata in increasing sequence of adjacency length form a decreasing sequence, which means that the most emphasis is put on direct co-occurrence evidence (i.e. the 1-adjacent neighborhood), than to the 2-adjacent nodes and so on.

Stratified sampling requires the strata to be mutually disjoint of each other. This means that the κ-neighbourhood of Equation 9 needs to be redefined to ensure that any node belongs to exactly one of the partitions (defined by its hop-count). To state this formally, we define the set of nodes of (*not up to*) hop-count j as

$$H_j(x_w) = \cup\{x_u : h(x_w, x_u) = j\} \qquad (15)$$

The κ-neighbourhood is then defined as

$$N_\kappa(x_w) = \cup_{j=1}^{\kappa}(H_j(x_w) - \cup_{j'=1}^{j-1} H_{j'}(x_w)). \qquad (16)$$

A subset of size l, comprised of stratified samples from $N_\kappa(x_w)$, is then sampled with decreasing priors $\beta_1, \ldots, \beta_\kappa$, i.e., $\beta_j < \beta_{j-1} \forall j = 2, \ldots, \kappa$ and $\sum_{j=1}^{\kappa} \beta_j = 1$.

Putting things together, the probability of sampling a node from the set $N_\kappa(x_w)$ defined as per Equation 16 is then given by

$$P(x_u|N_\kappa(x_w)) = \beta_j P(x_u|H_j(x_w)) = \beta_j \frac{\omega(x_w, x_u)}{\omega(x_w, .)}, \qquad (17)$$

where $\omega(x_w, x_u)$ are edge weights computed with Equation 14 and $\omega(x_w, .)$ denotes the sum of edges emanating from node x_w.

As a point of note, for our experiments, we obtained optimal results by using $\kappa = 2$. Consequently, to simplify the description of our experiments, we name the parameter β_1 as β (the parameter β_2 is then identical to $1 - \beta$). We would also mention at this point that our proposed way of constructing the context by sampling neighboring nodes is different from the one proposed in (Grover and Leskovec, 2016), which uses a combination of breadth-first (BFS) and depth-first (DFS) traversals, with parameters p and q respectively.

#Documents	4,641,754
#Avg. Doc Length (#words)	43.23
#Vocabulary size	461,572
#Tokens	202,575,916

Table 1: Dataset characteristics of DBPedia-2014.

Our experiments reveal that our sampling strategy outperforms that of Grover and Leskovec (2016) (treated as a baseline).

5 Experimental Setup

In this section, we describe our experimental setup to evaluate our new word embedding method.

5.1 Dataset

A word embedding algorithm requires a collection to learn word representations. To compare the various word embedding approaches (i.e. our method and the baselines), we use the DBPedia (2014) corpus, which is a collection of abstracts of Wikipedia pages crawled in 2014[1]. Dataset characteristics are outlined in Table 1. As part of preprocessing, we removed words with collection frequency less than 10 and also removed stopwords[2].

5.2 Baselines and Implementation

The objective of our experiments is two-fold. First, to show that a combination of local and global approaches is likely to yield effective embedded representations of word vectors, and second that our proposed graph-based formalism is likely to work better than a trivial black-box way of combining the two sources of information.

Local Co-occurrence approaches. As approaches that use local co-occurrence information, we use three state-of-the-art embedding approaches namely skip-gram word2vec with negative sampling (SGNS) (Mikolov et al., 2013a), Glove (Pennington et al., 2014) and Fasttext (Joulin et al., 2016). All these methods rely only on co-occurrences (at the level of words for the first two and at the level of character n-grams for the last one) within a word or character n-gram window of specified length k (acting as a parameter). Fasttext learns the vector representation of each word by aggregating (vector sum) the vector representations of its constituent n-grams.

---

[1] `http://downloads.dbpedia.org/2014/en/long_abstracts_en.ttl.bz2`

[2] `http://www.lextek.com/manuals/onix/stopwords2.html`

Additionally, we also employ a more recent approach, namely ELMO (Peters et al., 2018), which relies on a pre-trained model (comprised of stacked bidirectional LSTMs) to infer vectors for a given context (typically a sequence of words). For our experiments,

Document-level Co-occurrence approaches. Although not an embedding approach, the LDA topic modeling algorithm outputs two matrices, namely $\theta \in \mathbb{R}^{M \times d}$ and $\phi \in \mathbb{R}^{d \times V}$, representing the document-topic and topic-words distribution respectively (Blei et al., 2003). LDA uses document-level word co-occurrences to estimate both these matrices. In principle, one can then use the ϕ matrix as a substitute for the word embedding parameter matrix of SGNS (see Equation 5). This gives d dimensional vectors for each word purely with a global co-occurrence based approach.

Although it is possible to choose other non-local co-occurrence approaches as baselines, e.g. PLSA (Hofmann, 1999) or LSA, (Deerwester et al., 1990), it was shown in (Blei et al., 2003) that LDA outperforms each of these. Consequently, we use the stronger baseline of LDA in our experiments.

Combination of Local and Non-local Co-occurrences. To empirically demonstrate the effectiveness of our proposed graph-based word-node embedding, we employ an additional baseline that is a linear combination of the word vectors obtained individually with the local and non-local approaches. More formally, the vector of each word w is given as

$$\mathbf{w} = \lambda \mathbf{w}_{\text{Local}} + (1 - \lambda) \mathbf{w}_{\text{LDA}}, \qquad (18)$$

where $\mathbf{w}_{\text{Local}}$ is the vector representation of word w obtained by a local co-occurrence baseline, i.e. SGNS and Glove, whereas $\mathbf{w}_{\text{LDA}}$ represents the vector for the word w obtained with LDA.

Additionally, we employ the node2vec approach as a baseline. In particular, we use node2vec to learn the word-node representations of the graph constructed as per Section 4. The purpose of this baseline is to show that our way of defining the contexts around word-nodes is more suitable for our task of word embedding than a general-purpose graph node embedding approach.

5.3 Evaluation Tasks and Datasets

To compare the relative performance of word-node2vec with the baselines, we use a number of datasets, each corresponding to one of the following three evaluation tasks.

Dataset	Composition	Example
MSR	Syntactic	good:better rough:X
Google	Syntactic and Semantic	Athens:Greece Berlin:X
SemEval	Syntactic and Semantic	dog:bone bird:X

Table 2: Word analogy datasets overview.

Word Similarity. A standard way to measure the effectiveness of embedded words is to measure how well the similarity between a pair of words correlates with human judgments. Two such standard datasets that we use for our experiments are the WSIM-353 (Finkelstein et al., 2014) and the MEN (Bruni et al., 2014) datasets. Both comprise a list of word pairs, with an associated human judged similarity value. This similarity value is expected to be high for semantically similar words, such as 'morning' and 'sunrise' (human assigned score of 49 out of 50), and low for semantically unrelated words, such as 'angel' and 'gasoline' (score of 1 out of 50), both examples being taken from the MEN dataset.

Word Analogy. The word analogy task consists of templates of the form "A:B as C:X", where A, B, and C are given words, whereas X is unknown. Using a vector representation of words this analogy task is solved by retrieving the vector most similar to that of $\mathbf{B} + \mathbf{C} - \mathbf{A}$. A word embedding is considered effective if it finds a greater number of correct answers (resulting in higher accuracy).

We employed three different analogy datasets, namely, the Google Analogy (Mikolov et al., 2013a), the MSR Analogy (Mikolov et al., 2013b) and the SemEval-2012 task 2 (Jurgens et al., 2012) datasets. The MSR dataset contains syntactic questions only involving morphological variations. The Google dataset on the other hand contains both syntactic and semantic questions.

Given an analogy 'A:B as C:D', the Semeval-2012 task requires prediction of the degree to which the semantic relations between A and B are similar to those between C and D. In our experiments, we treat the given entity D as unknown and seek to predict D, similar to the MSR and Google analogy datasets. Table 2 provides an overview of examples from these datasets.

Concept Categorization Task. The concept categorization task requires classifying nouns into a concept type derived from an ontology. For this task, we employ the AP (Almuhareb and Poesio, 2005), BLESS (Baroni and Lenci, 2011) and ESSL_{2b} (Marco Baroni and Lenci, 2008) datasets. The AP dataset contains 402 nouns from 21 WordNet classes, e.g., nouns such as 'ceremony', 'feast', and 'graduation' belong to the class 'Social Occasion'. The BLESS dataset, designed for the evaluation of distributional semantic models, contains 200 distinct English concrete nouns as target concepts. These nouns are categorized into 17 broad classes.

Evaluation Metrics and Pipeline. The word similarity prediction effectiveness is measured with the help of Spearman's rank correlation coefficient ρ. This measures the rank correlation (higher is better) between the list of word pairs sorted in decreasing order of inter-similarity values as predicted by a word embedding algorithm and the reference list of human judged word pairs. For the analogy and the concept categorization tasks, we report the accuracy in predicting the reference word and that of the class, respectively.

Parameters and Settings. In our experiments, for all the methods, except ELMO, we set the number of dimensions to 200. To find optimal settings for each method (except ELMO), we use the MEN dataset as a development set for tuning the parameters of each method. Each method with the optimal parameter settings is then applied for the rest of the datasets and tasks.

Since we used a pre-trained model for ELMO, the number of dimensions corresponds to the size of the output layer of the network, the value of which in the default configuration of the Python implementation[3] is 1024.

The parameters of SGNS are window size (k) and the number of negative samples (NS). For the baseline approach SGNS, we varied k from 5 to 40 in steps of 5 and found that the best results are obtained when $k = 10$ and $NS = 5$. Similarly, for Glove we chose the optimal settings by varying k within the same range of $[5, 40]$ and found that the optimal ρ for the MEN dataset is obtained for $k = 20$. We obtain the LDA results by setting the number of topics to 200 (so as to match with the dimensionality). As LDA hyper-parameters, we use settings as prescribed in (Griffiths and Steyvers, 2004), i.e., $\beta = 0.1$ and $\alpha = 0.25$ $(50/(\#topics = 200))$.

Method	Spearman's ρ	
	MEN	WSIM
SGNS ($k = 10, NS = 5$)	0.7432	0.6977
Glove ($k = 20$)	0.7066	0.6706
FastText	0.7307	0.6518
ELMO	0.4225	0.4631
LDA	0.4933	0.4074
SGNS-LDA ($\lambda = 0.9$)	0.7367	0.6548
Node2vec ($p = 0.5, q = 0.5, l = 40$)	0.7440	0.6988
Word-node2vec ($\alpha = 0.5, \beta = 0.7, l = 20$)	**0.7491**	**0.7032**

Table 3: Word similarity prediction results.

Since we found that SGNS performed significantly better than Glove, we use SGNS vectors for the linear combination method (Equation 1), which we call `SGNS-LDA` from hereon. The parameter λ was varied within a range of $[0.1, 0.9]$ in steps of 0.1 ($\lambda = 0$ and $\lambda = 1$ degenerate to that of LDA and SGNS respectively). We found that the best results are obtained for $\lambda = 0.9$.

For node2vec baseline approach of word-node embedding, we varied the parameters p and q (BFS and DFS parameters) within a range of $[0.1, 5]$ and found that the best results on the MEN dataset are given for $p = 1$ and $q = 1$ (Grover and Leskovec, 2016). Another parameter in node2vec is the random walk length, l, for which the optimal value was found to be 80.

For word-node2vec, in addition to window size (k) and number of negative samples (NS), three more parameters are: i) α, i.e., the importance of the presence of a term relative to its informativeness (Equation 12, ii) β, the prior assigned to sampling from the 1-adjacent neighborhood, and iii) the size of the context sampled from the neighborhood, l (this is analogous to the random walk length parameter of node2vec). Instead of separately optimizing the parameters common to SGNS, we directly use the optimal values of $k = 10$ and $NS = 5$ for word-node2vec. The optimal results of the additional parameters, tuned on the MEN dataset, are shown in Table 3.

6 Results

Word Similarity Prediction. Table 3 shows the results obtained by the competing methods on the word similarity prediction task. It can be seen that Glove turns out to be relatively ineffective in modeling the semantic representations of words

---

[3] `https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md`

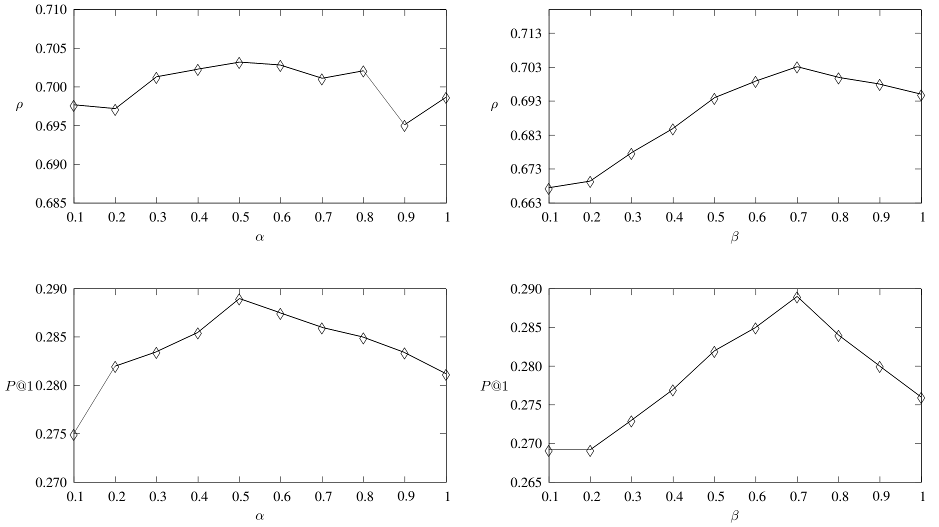


Figure 1: Parameter sensitivity of word-node2vec on word prediction (left column) and word analogy (right column) tasks using WSIM (top row) and MSR (bottom row) datasets.

Method	Accuracy (P@1)		
	Google	MSR	SemEval'12
SGNS	0.5615	0.2777	0.1460
Glove	0.4841	0.2485	0.1419
FastText	0.4930	0.2607	**0.1592**
ELMO	**0.5986**	0.2789	0.1439
LDA	0.0578	0.0158	0.0596
SGNS-LDA	0.5491	0.2776	0.1413
Node2vec	0.5588	0.2785	0.1427
Word-node2vec	0.5627	**0.2890**	0.1464

Table 4: Word analogy results.

Method	Accuracy		
	AP	BLESS	ESSLI$_{2b}$
SGNS	0.6194	0.7500	0.7500
Glove	0.6343	0.7200	0.7250
FastText	0.6119	**0.7950**	0.7250
ELMO	0.6368	0.7350	0.7500
LDA	0.3383	0.3900	0.6500
SGNS-LDA	0.5796	0.7850	**0.7750**
Node2vec	0.6355	0.7500	0.7350
Word-node2vec	**0.6393**	**0.7950**	**0.7750**

Table 5: Concept categorization results.

as compared to human judgments. SGNS performs significantly better and the settings trained on MEN dataset generalize well on the WSIM-353 dataset as well. LDA performs rather poorly indicating that only global co-occurrences can lead to noisy representations of words. FastText performs worse as compared to SGNS. It is worth mentioning that the performance of ELMO is disappointing on this task of semantic similarity prediction, because of the most likely reason that it better learns vector representations of word in the presence of a context.

A linear combination of SGNS and LDA (Equation 1 with $\lambda = 0.9$) does not perform better than SGNS, which means that a simple way of combining the embedded representations obtained individually with local and non-local approaches does not work well.

The node2vec approach of embedding nodes of the word-nodes graph constructed as per the description of Section 4 relies on a random walk based construction of the context of a word node. This random walk based context construction is only able to improve the SGNS results slightly, indicating that random walks can introduce noise in the contexts of word-nodes.

The word-node based graph construction (incorporating local and non-local co-occurrences in a principled way) works particularly well in conjunction with the stratified sampling based approach of selecting context words from the κ-neighborhood. The optimal value of $\alpha = 0.5$ suggests that document-level co-occurrences should be computed by assigning equal importance to term presence and informativeness. A value of $\beta = 0.7$ confirms the hypothesis that more emphasis should be put on direct co-occurrences.

Word Analogy and Concept Categorization. Similar trends are observed in the word analogy and concept categorization tasks in Tables 4 and 5 respectively. Relatively higher improve-

Rank	SGNS		word-node2vec	
1	albums	0.929	albums	0.926
2	selftitled	0.885	selftitled	0.883
3	rerecorded	0.868	rerecorded	0.863
4	promotional	0.815	**released**	0.852
5	reissue	0.790	**song**	0.810

Table 6: Nearest neighbors of the word ‘album’ obtained by SGNS and word-node2vec.

ments with word-node2vec are noted for the MSR analogy task (comprised of syntactic categories). Among the baseline approaches, both node2vec and SGNS-LDA work well on the concept categorization task. However, the performance improvements are inconsistent across datasets, e.g. SGNS-LDA performs well on ESSLI$_{2b}$ and poorly on AP. Our proposed method configured on the MEN dataset works consistently well across all datasets, which indicates that word-node2vec can generalize well for different tasks.

As a side observation, we note that ELMO performs well for the analogy and concept categorization tasks (yielding the best results in particular on the Google analogy dataset). Although the results are not directly comparable because of differences in the dimensionality of the vectors and also in the collection of documents used in the pretrained ELMO vectors (Billion word benchmark as against DBPedia in our case), it could possibly be reasoned that the additional contextual information of the ELMO vectors turns out to be useful for in the analogy task.

Embedding Examples. Table 6 shows an example of the change in the neighbourhood of a sample word in the embedded space obtained by SGNS and word-node2vec. It can be seen from the table that word-node2vec is able to push relevant words, such as ‘released’ and ‘song’ within the top 5-NN of the word ‘album’. Although the words ‘promotional’ and ‘reissue’ are related to ‘album’, the semantic association of ‘released’ and ‘song’ with ‘album’ is apparently higher. We found that the word ‘song’ occurs in the local context of the word ‘album’ only $133,494$ number of times out of a total number of $177,487$ instances of the word ‘album’. This means that a significant percentage of times (almost 25%), ‘song’ co-occurs with ‘album’ at a document-level. Our embedding algorithm is able to leverage this information by making the vector for ‘song’ closer to ‘album’.

Sensitivity Analysis. Tables 3-5 show word-node2vec results with optimal parameter settings. We now investigate the effect of varying these parameters on each individual evaluation task. We observe that both term presence and term informativeness are important to model document-level co-occurrences as seen from the fact that the ρ and accuracy values decrease as α gets close to 0 or 1 (the 1st and 3rd plots from the left of Figure 1). Similarly, it can be seen that the results tend to improve with higher values of β, which confirms that direct associations between words in the word-node graph are more important than transitive ones (2nd plot from the left and the rightmost plot of Figure 1). However, second-order transitive associations are still important because the results tend to decrease for β close to 1.

7 Conclusions and Future work

We proposed a word embedding approach that leverages document-level non-local co-occurrences, in addition to the window-based local co-occurrences. We proposed a graph-based framework, in which words are represented as nodes and the edges between a pair of words reflect the degree of association between them. This association is a function of both the local and the document-level co-occurrences, which enables our approach to achieve ‘the best of both worlds’ in word embedding. Experiments show that our proposed method outperforms local approaches, namely word2vec, Glove and FastText, on a number of different tasks. Our approach also outperforms a naive black-box combination of embeddings obtained separately by local and document-level approaches. This proves the importance of addressing both these sources of information jointly in an embedding objective.

In future, we would like to explore ways of applying a similar graph based formalism for learning vectors for documents.

Acknowledgements

This work was supported by Science Foundation Ireland as part of the ADAPT Centre (Grant No. 13/RC/2106) (`www.adaptcentre.ie`). This work started as an internship during the first author’s visit to IBM Research Lab, Ireland.

References

Abdulrahman Almuhareb and Massimo Poesio. 2005. Concept learning and categorization from the web. In *Proc. of COGSCI*, pages 103–108.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proc. of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49:1–47.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. of NAACL HLT*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2014. Placing search in context: The concept revisited. In *Proc. of WWW 2014*, pages 406–414.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J. F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proc. of SIGIR'15*, pages 795–798.

Eric Gaussier and Cyril Goutte. 2005. Relation between plsa and nmf and implications. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 601–602.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proc. of the 22Nd ACM SIGKDD 2016*, pages 855–864.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proc. of CIKM '16*, pages 55–64.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 289–296. Morgan Kaufmann Publishers Inc.

Stanislaw Jastrzebski, Damian Lesniak, and Wojciech Marian Czarnecki. 2017. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *CoRR*, abs/1702.02170.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

David A. Jurgens, Peter D. Turney, Saif M. Mohammad, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proc. of the Main Conference and the Shared Task, and Volume 2: Proc. of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 356–364.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

Stefan Evert. Marco Baroni and Alessandro Lenci. 2008. Esslli workshop on distributional lexical semantics. *ESSLLI Workshop on Distributional Lexical Semantics*, 101:1–70.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS 2013*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc. of NAACL 2013*, pages 746–751.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proc. of AAAI'16*, pages 2786–2792.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP 2014*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL 2018*.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281. ACM.

Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J. F. Jones. 2016. Word vector compositionality based relevance feedback using kernel density estimation. In *Proc. of CIKM'16*, pages 1281–1290. ACM.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *Proc. of ACL'16*, pages 130–139.

Cross-Topic Distributional Semantic Representations Via Unsupervised Mappings

Eleftheria Briakou[1,2]*, **Nikos Athanasiou**[2], **Alexandros Potamianos**[2,3]

[1]University of Maryland, College Park, MD
[2]School of ECE, National Technical University of Athens, Athens, Greece
[3]Signal Analysis and Interpretation Laboratory (SAIL), USC, Los Angeles, USA

ebriakou@cs.umd.edu, athn.nik@gmail.com, potam@central.ntua.gr

Abstract

In traditional Distributional Semantic Models (DSMs) the multiple senses of a polysemous word are conflated into a single vector space representation. In this work, we propose a DSM that learns multiple distributional representations of a word based on different topics. First, a separate DSM is trained for each topic and then each of the topic-based DSMs is aligned to a common vector space. Our unsupervised mapping approach is motivated by the hypothesis that words preserving their relative distances in different topic semantic sub-spaces constitute robust *semantic anchors* that define the mappings between them. Aligned cross-topic representations achieve state-of-the-art results for the task of contextual word similarity. Furthermore, evaluation on NLP downstream tasks shows that multiple topic-based embeddings outperform single-prototype models.

1 Introduction

Word-level representation learning algorithms adopt the *distributional hypothesis* (Harris, 1954), presuming a correlation between the distributional and the semantic relationships of words. Typically, these models encode the contextual information of words into dense feature vectors—often referred to as *embeddings*—of a k-dimensional space, thus creating a Vector Space Model (VSM) of lexical semantics. Such embeddings have been successfully applied to various natural language processing applications, including information retrieval (Manning et al., 2008), sentiment analysis (Tai et al., 2015), and machine translation (Amiri et al., 2016; Sharaf et al., 2017).

Despite their popularity, traditional DSMs rely solely on models where each word is uniquely represented by one point in the vector space. From a linguistic perspective, these models cannot capture the distinct meanings of polysemous words (e.g., *bank* or *cancer*), resulting in conflated word representations of diverse contextual semantics.

To alleviate this problem, DSMs with multiple representations per word have been proposed in the literature, based on clustering local contexts of individual words (Reisinger and Mooney, 2010; Tian et al., 2014; Neelakantan et al., 2014). An alternative way to train multiple representation DSMs is to utilize semantic lexical resources (Rothe and Schütze, 2015; Pilehvar and Collier, 2016). Christopoulou et al. (2018), based on the assumption that typically words appear with a specific sense in each topic, proposed a topic-based semantic mixture model that exploits a combination of similarities estimated on topic-based DSMs for the computation of semantic similarity between words. Their model performs well for a variety of semantic similarity tasks; however, it lacks a unified representation of multiple senses in a common semantic space. The problem of defining transformations between embeddings—trained independently under different corpora—has been previously examined in various works, such as machine translation (Mikolov et al., 2013b; Xing et al., 2015; Artetxe et al., 2016), induction of historical embeddings (Hamilton et al., 2016) and lexical resources enrichment (Prokhorov et al., 2017).

Following this line of research, we induce the creation of multiple cross-topic word embeddings by projecting the semantic representations of topic-based DSMs to a unified semantic space. We investigate different ways to perform the mappings from the topic sub-spaces to the unified semantic space, and propose a completely unsupervised approach to extract *semantic anchors* that define those mappings. Furthermore, we claim that polysemous words change their meaning in different topic domains; this is reflected in rela-

---

[*] The research was performed when the author was an undergraduate researcher at School of ECE, NTUA in Athens, Greece.

Proceedings of NAACL-HLT 2019, pages 1052–1061
Minneapolis, Minnesota, June 2 - June 7, 2019.

tive shifts of their distributional representations in different topic-based DSMs. On the other hand, semantic anchors should have consistent semantic relationships regardless of the domain they reside in. Hence, their distributions of similarity values should also be similar across different domains. Finally, we apply a smoothing technique to each word's set of topic embeddings, resulting in representations with fine-grained semantics.

To our knowledge, this is the first time that mappings between semantic spaces are applied to the problem of learning multiple embeddings for polysemous words. Our multi-topic word representations are evaluated on the contextual semantic similarity task and yield state-of-the-art performance compared to other unsupervised multi-prototype word embedding approaches. We further perform experiments on two NLP downstream tasks: text classification and paraphrase identification and demonstrate that our learned word representations consistently provide higher performance than single-prototype word embedding models. The code of the present work is publicly available[1].

2 Related Work

Methods that assign multiple distributed representations per word can be grouped into two broad categories.[2] Unsupervised methods induce multiple word representations without leveraging semantic lexical resources. Reisinger and Mooney (2010) were the first to create a multi-prototype DSM with a fixed number of vectors assigned to each word. In their model, the centroids of context-dependent clusters were used to create a set of "sense-specific" vectors for each target word. Based on similar clustering approaches, follow-up works introduced neural network architectures that incorporated both local and global context in a joint training objective (Huang et al., 2012), as well as methods that jointly performed word sense clustering and embedding learning as in Neelakantan et al. (2014); Li and Jurafsky (2015). A probabilistic framework was introduced by Tian et al. (2014), where the Skip-Gram model of Word2Vec was modified to learn multiple embedding vectors. Furthermore, latent topics were integrated into the Skip-Gram model, resulting in topical word embeddings which modeled the semantics of a word under different contexts (Liu et al., 2015b,a; Nguyen et al., 2017). Another topic-related embedding creation approach was proposed in Christopoulou et al. (2018) where a mixture of topic-based semantic models was extracted by topical adaptation of in-domain corpora. Other approaches used autoencoders (Amiri et al., 2016), convolutional neural networks designed to produce context representations that reflected the order of words in a context (Zheng et al., 2017) and reinforcement learning (Lee and Chen, 2017; Guo et al., 2018).

Supervised approaches, based on prior knowledge acquired by sense inventories (e.g., WordNet) along with word sense disambiguation algorithms, were also introduced for sense-specific representations extraction (Chen et al., 2014; Iacobacci et al., 2015). In other works, pre-trained word embeddings have been extended to embeddings of lexemes and synsets (Rothe and Schütze, 2015) or were de-conflated into their constituent sense representations (Pilehvar and Collier, 2016) by exploiting semantic lexical resources.

3 Unified Multi-Topic DSM (UTDSM)

Our system follows a four-step approach:

1. **Global Distributional Semantic Model.** Given a large collection of text data we train a DSM that encodes the contextual semantics of each word into a single representation, also referred to as Global-DSM.

2. **Topic-based Distributional Semantic Models.** Next, a topic model is trained using the same corpus. The topic model splits the corpus into K (possibly overlapping) sub-corpora. A DSM is then trained from each sub-corpus resulting in K topic-based DSMs (TDSMs). The topical adaptation of the semantic space takes into account the contextual variations a word exhibits under different thematic domains and therefore leads to the creation of "topic-specific" vectors (topic embeddings).

3. **Mappings of topic embeddings.** Next, we map the vector space of each topic-based DSM to the shared space of the Global-DSM, using a list of anchor words selected through

---

[1] `https://github.com/Elbria/utdsm_naacl2018`

[2] We limit our discussion to related works that use monolingual DSMs and corpora.

an unsupervised self-learning scheme. In the unified semantic space each word is represented by a set of topic embeddings that were previously isolated in distinct vector spaces, thus creating a Unified multi-Topic DSM (UTDSM).

4. **Smoothing of topic embeddings.** As an optional step, we employ a smoothing approach in order to cluster a word's topic embeddings into N Gaussian distributions via a Gaussian Mixture Model (GMM). This step lessens the noise introduced to our system through the semantic mappings and sparse training data.

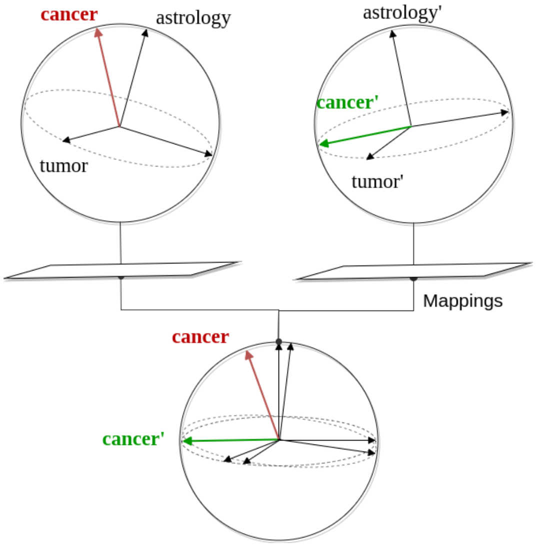


Figure 1: Simplified depiction summarizing the intuition behind the alignment process of topic embeddings. In the unified vector space, the polysemous word *cancer* is represented by two topic vectors that capture different semantic properties of the word under a zodiacal and a medical topic. Words *astrology* and *tumor* are examples of *semantic anchors* that define the mappings.

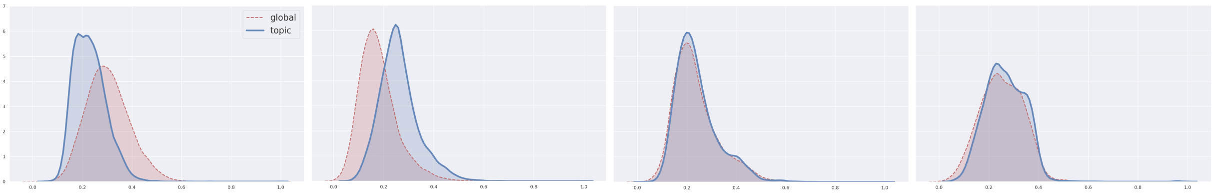
3.1 Topic-Based Distributional Semantic Models

The first step towards the thematic adaptation of the semantic space is the induction of in-domain corpora, using the Latent Dirichlet Algorithm (LDA) (Blei et al., 2003). LDA is a generative probabilistic model of a corpus. Its core idea is that documents are represented as random mixtures over topics; where each topic is defined as a probability distribution over a collection of words. Given as input a corpus of documents, LDA trains a topic model and creates a distribution of words for each topic in the corpus. Using the trained LDA model, we infer a topic distribution for each sentence in the corpus. Afterward, following a soft clustering scheme each sentence is included in a topic-specific corpus when the posterior probability for the corresponding topic exceeds a predefined threshold. The resulting topic sub-corpora are then used to train topic-based DSMs. Any of the DSM training algorithms proposed in the literature can be used for this purpose; in this paper, we opt for the Word2Vec model (Mikolov et al., 2013a).

3.2 Mappings Of Topic Embeddings

The intrinsic non-determinism of the Word2Vec algorithm leads to the creation of continuous vector spaces that are not naturally aligned to a unified semantic reference space, precluding the comparison between words of different thematic domains. To circumvent this limitation, we need to map the word representations of TDSMs to a shared vector space. In particular, we hypothesize that TDSMs capture meaningful variations in usage of polysemous words, while the relative semantic distance between monosemous words is preserved. This hypothesis motivated us to think of monosemous words as *anchors* between semantic spaces, as illustrated in Figure 1. One way to retrieve the list of anchors is to extract monosemous words from lexical resources such as WordNet (Prokhorov et al., 2017). However, this method is restricted to languages where such lexical resources exist and depends on the lexical coverage and quality of such resources.

To overcome the above limitations, we propose a fully unsupervised method for semantic anchor induction. Although the embeddings of the topic and global semantic vector spaces are not aligned, their corresponding similarity matrices (once normalized) are. Based on this observation, we compute the similarity between a given word and every other word in the vocabulary (similarity distribution) for the different topic and global spaces. Then, we assume that good semantic anchors should have similar similarity distributions across the topic-specific and the global space, as illustrated in Figure 2. Artetxe et al. (2018) was based on a similar observation to align vector semantic spaces in bilingual machine translation context.



view crater professor october

Figure 2: Similarity distributions of four different words (corresponding to the smoothed density estimates of the similarity matrices) in topic domain space as defined in Equation 1 and global space s_g^i. Selected anchors ("professor" and "october") have more similar distributions in the global and topic spaces, when compared to unselected ones ("view" and "crater"). We observe that the selected anchors are less ambiguous, while the not selected ones are expected to have diverse contextual semantics.

Let V be the intersection of the Global-DSM and the K TDSMs vocabularies and d the embedding dimension. We then define $X_k \in \mathbb{R}^{|V| \times d}$ as the embedding matrix of the k-th TDSM, and $Y \in \mathbb{R}^{|V| \times d}$ as the embedding matrix of the global DSM, where the i-th row of each matrix corresponds to the unit normalized representation of a word in V. Then, we define $S_k = X_k X_k^T$, $S_g = YY^T \in \mathbb{R}^{|V| \times |V|}$ to be the similarity distribution matrices for the k-th TDSM and the global-DSM, respectively. Then our objective is to extract a list of semantic anchors A that minimizes the Euclidean distance between the two different similarity distributions. Specifically, for every word i we calculate the average semantic distribution across all topics:

$$< s_k^i >_k = \frac{1}{K} \sum_{k=1}^{K} s_k^i \tag{1}$$

$$\| < s_k^i >_k - s_g^i \|_2 , \quad \forall\, i = 1, \ldots, |V| \tag{2}$$

where s_g^i, s_k^i is the i-th row of the S_g and S_k similarity matrix, respectively, representing the similarity distribution between word i and every other word in the vocabulary V. We then choose $|A|$ anchors as the words with the smallest values according to criterion 2. Furthermore, we assume that there exists an orthogonal transformation matrix between the topic embeddings of the extracted semantic anchors of each TDSM (source space) and the corresponding representations of the global-DSM (target space). The orthogonality constraint on the transformation matrix is widely adopted by the literature for various semantic space alignment tasks (Xing et al., 2015; Artetxe et al., 2016; Hamilton et al.). Assume $\alpha_k^j \in \mathbb{R}^d$ is the vector representation of the j-th *anchor* word in the source space and $\alpha_g^j \in \mathbb{R}^d$ is its corresponding vector representation in the target space. The transformation matrix $M_k \in \mathbb{R}^{d \times d}$ that projects the first space to the latter is learned via solving the following constraint optimization problem:[3]

$$\min_{M_k} \sum_{j=1}^{|A|} \| M_k \alpha_k^j - \alpha_g^j \|_2^2, \quad \text{s.t. } M_k M_k^T = \mathbb{I} \tag{3}$$

The induction of multiple topic embeddings in the unified vector space is achieved via applying Equation 3 to each TDSM. Specifically, given a word and its k-th topic distributed representation $x_k \in \mathbb{R}^d$, we compute its projected representation $x'_k \in \mathbb{R}^d$ as follows:

$$x'_k = M_k x_k \tag{4}$$

3.3 Smoothing Of Topic Embeddings

Starting from the set of aligned topic embeddings $\{x'_k\}_{k=1}^K$ for each word, we learn a Gaussian Mixture Model with N components, where closely positioned topic embeddings are assigned to the same component. This step operates as an implicit way of segmenting the space of topic embeddings for each word in order to capture more useful hyper-topics—union of topics—which better represent their different meanings. We suggest that each Gaussian distribution forms a semantically coherent unit that corresponds to closely related semantics of the target word. Subsequently, the mean vector of each Gaussian distribution is used as a representative vector of each component, leading to a new set of *smoothed* topic embeddings $\{x_n^*\}_{n=1}^N$ for each word, where $x_n^* \in \mathbb{R}^d$.

---

[3] This problem is known as the orthogonal Procrustes problem and it has a closed form solution as proposed in (Schönemann, 1966).

4 Experimental Setup

4.1 DSM Settings

As our initial corpus we used the English Wikipedia, containing 8.5 million articles (Turney, 2012). During the training of the topic model, we used the articles found in the Wikipedia corpus and employed the Gensim implementation of LDA (Rubenstein and Goodenough, 1965) setting the number of topics K to 50. Using a threshold of 0.1, we followed a soft-clustering approach, to bootstrap the creation of topic sub-corpora, using our trained topic model. Finally, we used Gensim's implementation of Word2Vec and Continuous Bag-of-Words method to train both the global-DSM and the TDSMs. The context window parameter of Word2Vec is set to 5, while the dimensionality d of all the constructed DSMs is equal to 300 or 500.[4]

4.2 Semantic Anchors

The number of *semantic anchors* that determine the mappings between our source and target spaces is set to $|A| = 5\,000$ [5] according to our unsupervised approach (criterion 2). Those are selected from the common set of words that are represented in all semantic spaces with $|V| \sim 12\,000$.

As a second experiment, we randomly sample $|A|$ words from the vocabulary of each TDSM to define its transformation matrix. We repeat this experiment 10 times, every time sampling a different list from the corresponding vocabulary and report average performance results.

4.3 Gaussian Mixture Model

To apply the smoothing technique on the set of a word's topic embeddings we use the Scikit-learn implementation of Gaussian Mixture Model clustering algorithm (Pedregosa et al., 2011). We initialize the mean vector of each component using k-means algorithm and the parameters of the model are estimated using Expectation-Maximization (EM) algorithm.

4.4 Contextual Semantic Similarity

To estimate the semantic similarity between a pair of words provided in sentential context, we use the standard evaluation Stanford Contextual Word Similarity (SCWS) (Huang et al., 2012) dataset which consists of 2 003 word-pairs with assigned semantic similarity scores computed as the average estimations of several human annotators. Following the evaluation guidelines proposed in literature, we employ the $\mathrm{AvgSimC}$ and $\mathrm{MaxSimC}$ contextual metrics, firstly discussed in Reisinger and Mooney (2010). In particular, given the word-pair (w, w'), and their provided contexts (c, c') we define:

$$\mathrm{AvgSimC}(w, w') = \frac{1}{K^2}\sum_{j=1}^{K}\sum_{k=1}^{K}\mathrm{p}(j|w,c)\mathrm{p}(k|w',c')\mathrm{d}(x'_j(w), x'_k(w')), \tag{5}$$

$$\mathrm{MaxSimC}(w, w') = \mathrm{d}(\hat{x}'(w), \hat{x}'(w')), \tag{6}$$

Following the notation used in 3.2, K is the number of topics returned by the trained LDA model, x'_j is the word embedding trained on the sub-corpus corresponding to the j-th topic after being projected to the unified vector space, $\mathrm{p}(j|w,c)$ denotes the posterior probability of topic j returned by LDA given as input the context c of word w, d denotes the cosine similarity between the two input representations and finally $\hat{x}'(w) = u_{\mathrm{argmax}_{1\leq j\leq K}\,\mathrm{p}(j|w,c)}(w)$ is the vector representation of word w that corresponds to the topic with the maximum posterior for c. Intuitively, a higher score in $\mathrm{MaxSimC}$ indicates the existence of more robust multi-topic word representations. On the other hand, $\mathrm{AvgSimC}$ provides a topic-based smoothed result across different embeddings.

4.5 Downstream NLP Tasks

Besides the standard evaluation benchmark of contextual word similarity, we also investigate the effectiveness of our mapped cross-topic embeddings on document and sentence level downstream NLP tasks: text classification and paraphrase identification. We report weighted-averaging precision, recall, F1-measure and accuracy performance metrics.

Text classification. We used the 20NewsGroup[6] dataset, which consists of about 20 000 documents. Our goal is to classify each document into one of the 20 different newsgroups based on its content.

---

[4] Any parameter not mentioned is set to default values of the corresponding implementations (e.g., Word2Vec, Gensim LDA).

[5] We have experimented with different values of anchors from $\{1\,000, 2\,000, 3\,000, 4\,000, 5\,000\}$ and report results for the best setup.

[6] http://qwone.com/ jason/20Newsgroups/

Paraphrase Identification. For this task we aimed at identifying whether two given sentences can be considered paraphrases or not, using the Microsoft Paraphrase dataset (Dolan et al., 2004).

Document and Sentence level representations. Given a document or a sentence D, where w_d corresponds to the d-th word in D, we extract its feature representation using three different ways:

$$\mathrm{AvgC_D} = \frac{1}{|D|}\sum_{d=1}^{|D|}\sum_{k=1}^{K} \mathrm{p}(k|D) x'_k(w_d), \tag{7}$$

$$\mathrm{Avg_D} = \frac{1}{|D|}\sum_{d=1}^{|D|}\sum_{k=1}^{K} \frac{1}{K} x'_k(w_d), \tag{8}$$

$$\mathrm{MaxC_D} = \frac{1}{|D|}\sum_{w=1}^{|D|} x'_m(w_d) \tag{9}$$

$$\text{s.t.} \quad m = \operatorname*{argmax}_{k=1,..,K}\{\mathrm{p}(k|D)\},$$

where $\mathrm{p}(k|D)$ denotes the posterior probability of topic k returned by LDA given as input the sentence/document D and $x'_k(w_d)$ is the mapped representation of word w_d for topic k. For the case of paraphrase identification, we extract a single feature vector for each sentence-pair via concatenating the features of the individual sentences.

After feature extraction, we train a linear Support Vector Classifier (SVM) (Pedregosa et al., 2011) using the proposed train/test sets for both tasks. We report the best results for each experimental configuration after tuning the SVM's penalty parameter of the error term using 500-dimensional word embeddings.

5 Results

In Table 1 we compare our model (UTDSM) with our baseline (Global-DSM) and other state-of-the-art multi-prototype approaches for the contextual semantic similarity task. It is clear that all different setups of UTDSM perform better than the baseline for both contextual semantic similarity metrics. Using a single Gaussian distribution (UTDSM + GMM (1)) at the smoothing step of our method produces similar results to the baseline model. This is anticipated as both methods provide a centroid representation of a word's diverse semantics. In terms of $\mathrm{MaxSimC}$ the model consistently yields higher performance when the list of semantic anchors is induced via our unsupervised method instead of using randomly selected anchor words (UTDSM Random). We also observe that random anchoring performs slightly worse than UTDSM with respect to $\mathrm{AvgSimC}$. This result validates our hypothesis that the representations of words, which share consistent similarity distributions across different topic domains, constitute informative *semantic anchors* that determine the mappings between semantic vector spaces.

Method	AvgSimC	MaxSimC
Liu et al. (2015a)	67.3	68.1
Liu et al. (2015b)	69.5	67.9
Amiri et al. (2016)	70.9	-
Lee and Chen (2017)	68.7	67.9
Guo et al. (2018)	69.3	68.2
300-dimensions		
Global-DSM	67.1	67.1
UTDSM Random	69.1 ± 0.1	66.4 ± 0.2
UTDSM	**69.6**	67.1
UTDSM + GMM (1)	67.4	67.4
UTDSM + GMM (2)	68.4	**68.3**
UTDSM + GMM (3)	68.9	**68.3**
UTDSM + GMM (8)	69.1	68.0
UTDSM + GMM (10)	69.0	67.8
500-dimensions		
Global-DSM	67.6	67.6
UTDSM Random	69.4 ± 0.1	66.5 ± 0.3
UTDSM	**70.2**	68.0
UTDSM + GMM (1)	67.6	67.6
UTDSM + GMM (2)	68.8	**68.6**
UTDSM + GMM (3)	69.0	68.5
UTDSM + GMM (8)	69.5	68.5
UTDSM + GMM (10)	69.2	68.0

Table 1: Performance comparison between different state-of-the-art approaches on SCWS, in terms of Spearman's correlation. UTDSM refers to the projected cross-topic representation, UTDSM Random refers to the case when random words served as anchors and GMM (c) corresponds to GMM smoothing with c components.

Furthermore, we observe that GMM smoothing has a different effect on the $\mathrm{MaxSimC}$ and $\mathrm{AvgSimC}$ metrics. Specifically, for $\mathrm{AvgSimC}$ we consistently report lower results when GMM smoothing is applied for different number of components. We attribute this behavior to a possible loss of model capacity—decrease in the number of topic embeddings—that is capable of capturing additional topic information. At the same time, our smoothing technique highly improves the performance of $\mathrm{MaxSimC}$ for all possible configurations. Given that this metric is more sensitive to noisy word representations, this result indicates that our technique lessens the noise introduced to our system and captures finer-grained topic senses of words.

Overall, the performance of our model is highly competitive to the state-of-the-art models in terms of $\mathrm{AvgSimC}$, for 500-dimensional topic embeddings. We also achieve state-of-the-art performance for the $\mathrm{MaxSimC}$ metric, using smoothed topic embeddings of 300 or 500 dimensions with 2 or 3 Gaussian components.

Method	Precision	Recall	F1-score	Accuracy
LDA	39.7	41.8	38.8	41.8
Global-DSM	62.9	63.3	62.9	63.3
$\mathrm{MaxC_D}$	61.9	63.0	62.0	63.0
$\mathrm{Avg_D}$	63.5	64.6	63.3	64.3
$\mathrm{AvgC_D}$	**64.6**	**65.5**	**64.5**	**65.5**

Table 2: Evaluation results of multi-class text classification.

Evaluation results on text classification are presented in Table 2. We observe that our model performs better than the baseline across all metrics for both averaging approaches ($\mathrm{AvgC_D}$, $\mathrm{Avg_D}$), while the usage of dominant topics appears to have lower performance ($\mathrm{MaxC_D}$). Specifically, we get an improvement of $2 - 2.5\%$ on topic-based average and $0.5 - 1\%$ on simple average combination compared to using Global-DSM.

Method	Precision	Recall	F1-score	Accuracy
Global-DSM	68.6	69.2	62.0	69.2
$\mathrm{MaxC_D}$	**69.0**	69.3	62.1	69.3
$\mathrm{Avg_D}$	67.7	**69.4**	**64.0**	**69.4**
$\mathrm{AvgC_D}$	68.8	69.4	62.6	69.4

Table 3: Evaluation results on paraphrase detection task.

Results for the paraphrase identification task are presented in Table 3. $\mathrm{Avg_D}$ yields the best results especially in F1 metric showing that cross-topic representations are semantically richer than single embeddings baseline (Global-DSM). Although we apply the topic distributions $\mathrm{p}(k|D)$ extracted from LDA (document-level model) to a sentence-level task, improvements over the baseline are also shown in the $\mathrm{AvgC_D}$ and $\mathrm{MaxC_D}$ cases.

Overall, the proposed UTDSM model outperforms the baseline Global-DSM model on both semantic similarity and downstream tasks.[7]

6 Cross-Domain Semantic Analysis

Finally, we carry out a cross-domain semantic analysis to detect the variations of a word's meaning in different topic domains. To that end, we use a list of known polysemous words and measure the semantic similarity between different topic representations of the same ambiguous word. The ultimate goal of this analysis is to validate that our model captures known thematic variations in semantics of polysemous words.

Table 4 includes examples of our analysis. The most probable words of the topics (second column) give an intuitive sense of their major contexts, while their nearest neighbors (third column) infer the sense of the target word in the corresponding topic domain. For example, the word *drug* is mostly related to "medication" in a broad medical domain; it experiences though a slight shift from this meaning when it resides in a topic about "illegal substances". Furthermore, the highly polysemous word *act* shifts from meaning "statute" to meaning "performance" under the corresponding law and art topics. Similar semantic variations are observed for words *python*, *rock* and *nursery*.

Moreover, in Figure 3 we visualize the topic embeddings of seven words before and after projecting the topic-based DSMs to the unified space, using principal component analysis. We additionally depict the Gaussian distribution learned from the topic representations of each word reflecting the uncertainty of their meanings. The center of each distribution is specified by the mean vector and contour surface by the covariance matrix. On the left, we depict the position of words prior to applying the unsupervised mapping approach where the topic sub-spaces are unaligned. In the unaligned space, words demonstrate similar area coverage regardless of their polysemy. After the mappings, we see on the right that the area under a word's distribution is indicative of its degree of polysemy. Specifically, we observe that the variance of the learned representations becomes larger for the cases of polysemous words such as

---

[7] Similar results were obtained for each metric using smoothed word embeddings. Also, there are no standard evaluation approaches for comparison of previous works on downstream tasks.

[8] Note that a topic domain is described as a distribution over words in our model.

Word	Topic Words	Nearest Neighbors	Similarity
drug	health, medical, cancer, treatment, disease	insulin, therapy, heparin, chemotherapy, vaccines	0.61
	drug, health, marijuana, alcohol, effects	meth, cocaine, methamphetamine, mdma, heroin	
act	law, court, legal, tax, state	bylaw, legislature, complying, entities, entitlement	0.39
	music, guitar, piano, dance, theatre	touring, pantomime, weekend, shakespeare, musical	
python	garden, plant, fish, bird, animal	macaw, crocodile, hamster, albino, rattlesnake	0.27
	software, forum, download, windows, web	algorithm, parser, notepad, gui, tutorial	
rock	mountain, river, park, road, trail	geology, slab, limestone, waterfalls, canyon	0.43
	music, guitar, piano, dance, theatre	touring, acoustic, americana, songwriter, combo	
nursery	garden, plant, tree, flower, gardening	camellias, succulents, greenhouse, ornamental, grower	0.46
	university, school, college, education, program	prep, montessori, grammar, preschool, infant	

Table 4: Examples of polysemous words and the change of meaning between different topic domains. First column lists the example target words. Second column includes the most probable words of the topic domains[8] these words are assigned to. Each row corresponds to a different topic domain. Third column shows the nearest monosemous neighbors of the target word in the corresponding topic domain. The last column corresponds to the cosine similarity between the two topic representations of the target word.

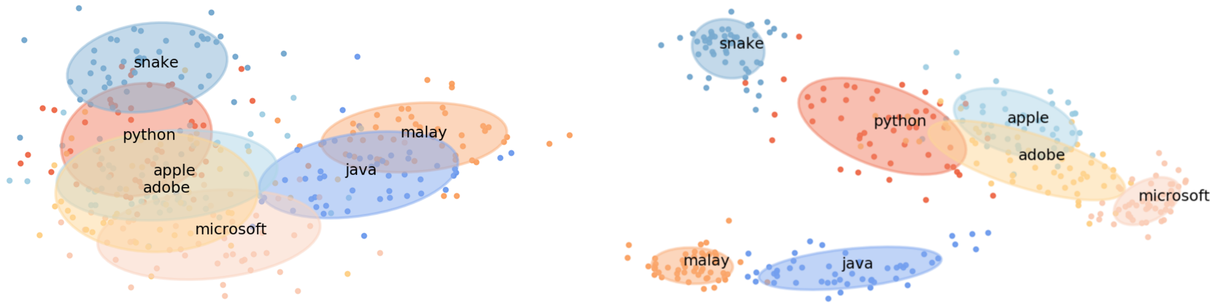


Figure 3: A 2-dimensional projection of the latent semantic space encoded in our unified vector space model, depicting the topic word representations of 7 words before (left) and after (right) mapping the TDSMs to the global semantic space.

"python", "java", "adobe" in order to assign some probability to their diverse meanings. Monosemous words such as "snake", "microsoft" and "malay" have smaller variances. Furthermore, we observe that the semantic relationships between words are much better captured by their corresponding positions in the aligned space.

7 Conclusion

We present an unsupervised approach of mapping multiple topic-based DSMs to a unified vector space in order to capture different contextual semantics of words. We assume that words having consistent similarity distributions regardless of the domain they exist in could be considered informative semantic anchors that determine the mappings between semantic spaces. The projected word embeddings yield state-of-the-art results on contextual similarity compared to previously proposed unsupervised approaches for multiple word embeddings creation, while they also outperform single vector representations in downstream NLP tasks. In addition, we provide insightful visualizations and examples that demonstrate the capability of our model to capture variations in topic semantics of words.

As future work, one can hypothesize that the area a word covers in the mapped space reveals its semantic range. In this direction, a refinement of the semantic anchor selection approach could be explored in an iterative way assuming that the variance of a word's Gaussian distribution denotes its degree of polysemy (Vilnis and McCallum, 2015). Moreover, we would like to explore a more sophisticated smoothing technique where the number of Gaussian components is adapted for each word. Given that Gaussian mixture embeddings could capture the uncertainty of a word's representation in the semantic space one could also investigate different metrics for measuring the semantic relationship between word pairs that go beyond their point-wise comparison. Finally, it may be helpful to investigate non-linear mappings between semantic spaces using deep neural network architectures.

Acknowledgments

Thi work has been partially funded by the Baby-Robot project, supported by the EU Horizon 2020 Program undergrant #687831.

References

Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1882–1892.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proc. Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035.

Fenia Christopoulou, Eleftheria Briakou, Elias Iosif, and Alexandros Potamianos. 2018. Mixture of topic-based distributional semantic and affective models. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 203–210.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*.

Fenfei Guo, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Inducing and embedding senses with scaled gumbel softmax. *arXiv preprint arXiv:1804.08077*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1489–1501.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics*.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2–3):146–162.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 873–882.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 95–105.

Guang-He Lee and Yun-Nung Chen. 2017. Muse: Modularizing unsupervised sense embeddings. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 327–337.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1732.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015a. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI*, pages 1284–1290.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015b. Topical word embeddings. In *Proc. AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. volume abs/1301.3781.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069.

Dai Quoc Nguyen, Dat Quoc Nguyen, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2017. A mixture model for learning multi-sense word embeddings.

In *Proc. 6th Joint Conference on Lexical and Computational Semantics*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1680–1690.

Victor Prokhorov, Mohammad Taher Pilehvar, Dimitri Kartsaklis, and Nigel Collier. 2017. Learning rare word representations using semantic bridging.

Joseph Reisinger and Raymond Mooney. 2010. Mixture Model with Sharing for Lexical Semantics. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1182.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1793–1803.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, pages 627–633.

Peter H. Schönemann. 1966. *A generalized solution of the orthogonal procrustes problem.*

Amr Sharaf, Shi Feng, Khanh Nguyen, Kiante Brantley, and Hal Daumé III. 2017. The umd neural machine translation systems at wmt17 bandit learning task. In *Proceedings of the Second Conference on Machine Translation*, pages 667–673.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proc. 53rd Annual Meeting of the Association for Computational Linguistics*, 1:1556–1566.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proc. International Conference on Computational Linguistics (COLING)*, pages 151–160.

Peter D Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *International Conference on Learning Representations (ICLR)*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Xiaoqing Zheng, Jiangtao Feng, Yi Chen, Haoyuan Peng, and Wenqing Zhang. 2017. Learning context-specific word/character embeddings. In *Proc. AAAI Conference on Artificial Intelligence*, pages 3393–3399.

What just happened? Evaluating retrofitted distributional word vectors

Dmetri Hayes
International Computer Science Institute
University of California, Berkeley
Berkeley, CA, 94720, USA
`dmetri@berkeley.edu`

Abstract

Recent work has attempted to enhance vector space representations using information from structured semantic resources. This process, dubbed retrofitting Faruqui et al. (2015), has yielded improvements in word similarity performance. Research has largely focused on the retrofitting algorithm, or on the kind of structured semantic resources used, but little research has explored why some resources perform better than others. We conducted a fine-grained analysis of the original retrofitting process, and found that the utility of different lexical resources for retrofitting depends on two factors: the coverage of the resource and the evaluation metric. Our assessment suggests that the common practice of using correlation measures to evaluate increases in performance against full word similarity benchmarks 1) obscures the benefits offered by smaller resources, and 2) overlooks incremental gains in word similarity performance. We propose root-mean-square error (RMSE) as an alternative evaluation metric, and demonstrate that correlation measures and RMSE sometimes yield opposite conclusions concerning the efficacy of retrofitting. This point is illustrated by word vectors retrofitted with novel treatments of the FrameNet data (Fillmore and Baker, 2010).

1 Introduction

One of the most challenging tasks in the field of Natural Language Processing (NLP) is accurately encoding meaning into a computational system. Currently, the predominant approach is to represent the meanings of linguistic units, such as words or phrases, as vectors in a high-dimensional space. Vector embeddings are trained over large text corpora using machine-learning techniques, and have proven useful for a wide range of applications, such as named entity recognition (Turian et al., 2010), semantic role labeling (Collobert et al., 2011), sentiment analysis (Socher et al., 2013), and machine translation (Zou et al., 2013).

Word vectors are typically trained solely on the distributional information from text corpora. Recent work has attempted to improve word vectors by infusing them with information from semantic resources in a post-processing step. This technique, referred to as *retrofitting*, was introduced by Faruqui et al. (2015). They adjusted pre-trained embeddings based on lexical relations in WordNet (Miller, 1995), FrameNet (Fillmore and Baker, 2010), and the Paraphrase Database (Ganitkevitch et al., 2013). In some cases, this method yielded gains in word similarity performance.

Retrofitting has been extended in a variety of ways. Briefly, these include 1) adding word-to-word relations to encompass more than just similarity relations, such as by directly introducing antonymy relations (Mrkšić et al., 2016), or by explicitly modeling the pairwise relations between items (Lengerich et al., 2017); 2) increasing the size of the output vocabulary (Speer et al., 2017), or extending the process to affect the word vectors of words outside of the semantic resource (Glavaš and Vulić, 2018); and 3) constructing sense-specific word vectors using a word sense ontology (Jauhar et al., 2015), or word sense information learned from parallel text corpora (Ettinger et al., 2016).

However, while Faruqui et al. (2015) has certainly spawned a productive line of research into improving pre-trained word vectors, the original study contained a puzzling finding: retrofitting with certain semantic resources actually appeared to harm the quality of the word embeddings. This seems counter-intuitive. In principle, if semantic resources contain information that is not already captured by the word vectors, then retrofitting should always improve them.

Proceedings of NAACL-HLT 2019, pages 1062–1072
Minneapolis, Minnesota, June 2 - June 7, 2019.

In order to understand why some semantic resources appear better suited for retrofitting word vectors, we conducted a fine-grained analysis of Faruqui et al.'s original technique. Given their popularity, we focused on word similarity evaluations. We observe that the perceived usefulness of a semantic resource depends on its coverage of the words in the evaluation benchmark. Furthermore, we report that the choice of evaluation metric can lead to different conclusions. We note that some gains in performance are not captured by correlation measures, and propose that root-mean-square error (RMSE) is more appropriate for measuring changes in word similarity performance.

2 Methods

2.1 Retrofitting

The original retrofitting algorithm from Faruqui et al. (2015) is described below. The process essentially moves the word vectors of related words closer together. A semantic resource can be regarded as a graph which covers a vocabulary $V = \{w_1, ..., w_n\}$ and denotes relations between them as edges $(w_i, w_j) \in E$. Given a set of pre-trained distributional vectors $\hat{\vec{W}} = \{\hat{\vec{w_1}}, ..., \hat{\vec{w_d}}\}$ and a semantic resource with edges E, the goal is to learn a new set of vectors $\vec{W} = \{\vec{w_1}, ..., \vec{w_d}\}$. Here $\vec{w_i}$ is the word vector corresponding to vocabulary item w_i. The objective function to be minimized is the following:

$$\sum_{w_i}^{V} \left(\alpha_i ||\vec{w_i} - \hat{\vec{w_i}}||^2 + \sum_{(w_i,w_j)}^{E} \beta_{ij} ||\vec{w_i} - \vec{w_j}||^2 \right) \quad (1)$$

The first term of the inner sum ensures that the vectors do not stray too far away from their original representations (controlled by α), while the second term compels the vectors to move closer to their neighbors in the semantic resource (controlled by β). In Faruqui et al.'s experiments, all $\alpha_i = 1$, and all $\beta_{ij} = \frac{1}{degree(w_i)}$, where $degree(w_i)$ refers to the number of neighbors w_i had in the resource. This is equivalent to specifying that half of the new retrofitted vector will come from the distributional data while the other half will be an average of its neighbors' word vectors. They allowed the process to run for 10 iterations. We retained these settings in our experiments.

Resource	Terms	Groupings
`WordNet+`	147,306	117,659
`PPDB`	84,467	102,899
`FrameNet`	8,483	1,074
`FrameNet-Anno`	37,855	7,146

Table 1: Number of word forms and word groupings per semantic resource.

2.2 Semantic resources

We employed three semantic resources in our analyses. Table 1 shows the number of terms and groupings in each resource after removing terms containing numbers or punctuation.[1]

WordNet. WordNet (Miller, 1995) is a large lexical database of English words. The resource is composed of synsets, groupings of synonyms. Synsets are linked together through a small number of semantic relations. We follow Faruqui et al. (2015) and link each word form to its synonyms, hypernyms, and hyponyms (`WN+`). For instance, the word *dog* is linked to *canine* (synonym), *corgi* (hyponym) and *domestic_animal* (hypernym). In order to faithfully replicate Faruqui et al., we collapsed part of speech and sense distinctions, meaning that a word form was linked to all of its related words through all of its synsets. For instance, *dog*'s neighbors include *corgi* through the noun *dog* (e.g. "Sam pet the dog.") and *track* through the verb *to dog* (e.g. "The task dogged me.") Although the word vectors and evaluations used in this study are insensitive to part of speech and sense distinctions, the number and order of groupings affects the retrofitting procedure. In particular, as noted by Speer and Chin (2016), the results depend on the order in which the groupings are iterated over. Though we attempted to group words by their synsets, this appeared to lead to poorer performance and we do not report those results here.

PPDB. The paraphrase database (Ganitkevitch et al., 2013) contains millions of English paraphrases automatically extracted from bilingual parallel corpora. The core idea is that if a non-English phrase translates to two distinct English strings, then these may be considered paraphrases of each other. For instance, since German *festgenommen* translates to both "thrown into jail"

---

[1] The number of groupings for PPDB is approximate, taken as the number of unique sets of words in Faruqui et al.'s pre-processed lexicon file.

and "imprisoned", the latter two are listed as paraphrases. Faruqui et al. (2015) used the XL lexical pack from PPDB 1.0. Since this version is no longer publically available, we used their preprocessed file (`PPDB`).

FrameNet. FrameNet (Fillmore and Baker, 2010) is a highly-interconnected lexical database of English containing sense-annotated sentences. The basic units of FrameNet are semantic frames, which specify the conceptual structure necessary to understand sets of lexical units (LUs). For instance, the frame Attack contains LUs such as *attack.v*, *attack.n* and *offensive.a*, which can be understood in light of the frame elements (FEs) *Assailant* and *Victim*. We performed two experiments with the FrameNet data. In the first, we grouped words together if they shared a frame (`FN`). Note that this differs from the treatment of WordNet because the frame groupings retain part of speech and sense distinctions. Although this method follows Faruqui et al. (2015), we located a bug in their code which led to a loss of about 1/3 of the data: the original code did correctly handle polysemy, which is widespread in FrameNet.

For our second experiment, we grouped words together based on the FEs that they filled (`FN-ANNO`). All of the FrameNet FEs were used in this task (i.e. both core and non-core FEs). Since FEs are defined with respect to their frames, each semantic role is frame-specific. The rationale is that words which can occupy the same semantic role should be more similar. We created groupings from the last nouns which appeared in the FE fillers in the annotation data. To illustrate, since the annotation data linked to the FE *Assailant* of the Attack frame included the nouns *enemy*, *troop*, *terrorist* and *forces*, their corresponding word vectors were moved closer together. Note that all of our retrofitting analyses ignored the frequency of a word's neighbor: even if *enemy* filled the FE *Assailant* 100 times, its effect on its neighbors would be identical to if it had only filled the FE once.

We recognize that the last noun heuristic is simplistic. However, we estimate that around 73% of the syntactic heads of FE fillers are nouns. Of these, 68% contain only one noun, and 18% contain only two nouns. Taken together, this implies that a more sophisticated approach is unlikely to alter the results. In addition to the last noun heuristic, we considered grouping the first nouns in the FE fillers, all of the nouns in the FE fillers, and the nouns from FE fillers which contained only one noun. All of these experiments yielded similar results, so we only report the last noun condition here. Nouns were identified using the default NLTK (Bird and Loper, 2004) English part-of-speech tagger.

2.3 Word vectors

Our analyses included two popular pre-trained word vector embeddings.

SG. `word2vec` (Mikolov et al., 2013) is widely-used to learn vector representations from distributional information. In the continuous skip-gram architecture (SG), the target word is fed into a log-linear classifier to predict surrounding words within a given context window. The available vectors were trained on about 100 billion words from the Google News dataset.

GloVe. Global Vectors for Word Representation (Pennington et al., 2014) is a global log-bilinear regression model which captures both global and local word co-occurrence statistics. We use the 300 dimension vectors trained on 6 billion words from Wikipedia and the English Gigaword corpus.

2.4 Word similarity

Word similarity judgments are the most widely-used method of intrinsic evaluation. We chose four commonly used word similarity datasets comprised of nouns, verbs and adjectives.

MEN3K (Bruni et al., 2012) contains 3,000 pairs of words from a set of labels for an image database. Interestingly, although Bruni et al. claim that their dataset "contains 3,000 pairs of randomly selected words that occur [as labels]", it only contains 751 unique words.[2] Therefore, as an additional evaluation of high-frequency words, we included MTURK-771 (Halawi et al., 2012), a crowd-sourced dataset of 771 word-pairs consisting of 1,113 unique words which we will refer to as **MT771**. The Stanford Rare Words (**RW**) dataset (Luong et al., 2013) is comprised of 2,034 word-pairs formed from 2,951 unique words. **SL999** (Hill et al., 2015) explicitly quantifies semantic similarity between pairs of words. The dataset contains 999 word pairs from 1,028 unique words. The word pairs in SL999 were chosen to cover the full range of concreteness within each part of speech category. We included RW

---

[2] By our calculations, the expected number of unique words obtained from 3,000 random pairs drawn from 20,515 labels (the number in their image database) is around 5,200.

and SL999 to examine whether the results of our analyses would differ for benchmarks containing common vs. rare words and for those capturing association and relatedness vs. similarity only.

3 Evaluation procedure

The standard approach to evaluate the performance of word vectors on word similarity judgments is to compute the cosine similarity values between each pair of words in the dataset and then calculate the correlation between these values and the similarity scores collected from human raters. A similar technique is used to assess the utility of different semantic resources in retrofitting word vectors: increases in correlation are taken to be indicative that information from the resource has been successfully injected into the word vectors. For both types of evaluations, Spearman correlation has become the preferred correlation measure.

However, there are several reasons that this method may be misleading. The first concerns the issue of the relative coverage of each resource. Simply put, not every resource contains all of the words in the evaluation dataset. If a resource lacks the words for a particular similarity judgment, then the predicted score will be the same for both the baseline and retrofitted vectors. This may have important consequences on the evaluation metric: the fixed scores can throw off the global ranking of the predicted scores, which is measured by the Spearman correlation.

For every word pair in a word similarity dataset, a resource can contain 1) both words, 2) one of the words, or 3) neither of the words. If the goal of the evaluation is to determine whether the knowledge of particular semantic resources can be added to word vectors, then it seems reasonable to only evaluate the resource on the word pairs it covers. In this case, the resource will either group the two words together or place them in separate groups, which can be interpreted as explicitly indicating whether the two words are semantically related or not. Conversely, it is obvious that retrofitting will not improve the vectors for the word pairs for which neither word is in the semantic resource.

The situation where only one word is present is more complicated. For example, imagine that a resource contained the word *view* but not the word *skyline*. Following retrofitting the vector for *view* will move while the vector for *skyline* will stay the same. The relationship between *view* and *skyline* will either become more accurate or less accurate, but this change does not directly stem from the semantic resource. If the goal of the retrofitting evaluation is to assess the usefulness of particular semantic resources, then including these kinds of word pairs is misleading, since the observed changes are incidental and do not reflect the semantic groupings in the resource.

In our analyses, "all pairs" shows the performance of the word vectors using all of the word similarity judgments, and "pairs in resource" shows their performance using only the subset comprised of judgments for which both words were contained in the semantic resource.

Our more radical proposal is to consider an entirely different evaluation metric altogether. Measures of correlation indicate how well word vectors are able to predict the similarity judgments. Spearman correlation specifically measures how well word vectors are able to predict the correct rankings of similarity judgments. For example, according to the MEN3K dataset, *brick* and *construction* should be ranked as less similar than *town* and *village*. Another conceivable way to test the word vectors ability to capture word similarity knowledge would be to directly compare the word vectors' predicted score with the human score. According to MEN3K, the average rated similarity for *town* and *village* was 43 out of 50. Taken literally, after normalizing the original scores the cosine similarity should be exactly 0.86. We operationalized this by evaluating word vectors using root-mean-square error (RMSE). This approach seems particularly appealing for measuring the effects of retrofitting because each similarity judgment contributes independently to the RMSE score.

One may wonder whether Pearson correlation, which measures linear association, might serve as a better comparison to RMSE. To address this concern, we employed the harmonic mean of the Pearson and Spearman correlations as our correlation measure. This blends the linear measure (Pearson) with the standardly-employed measure (Spearman). However, we note that the resulting baseline and retrofitted scores were very similar across correlation measures, and so our conclusions regarding the choice of evaluation metric were unaffected by this decision.

In the analysis that follows, we considered the effect of resource coverage and evaluation metric

	NB	GloVe	SG
MT771	**0.80 / 0.35**	0.65 / 0.36	0.66 / 0.39
MEN3K	**0.85** / 0.28	0.75 / 0.27	0.78 / **0.27**
RW	**0.54 / 0.37**	0.35 / 0.55	0.45 / 0.45
SL999	**0.66 / 0.21**	0.38 / 0.26	0.45 / 0.25

Table 2: Baseline word vector similarity performance. Scores are listed in the form "Correlation/RMSE". Bold face indicates the best-performing set of vectors for each similarity dataset for their correlation score and for their RMSE score. Recall that lower RMSE is better.

on the results of retrofitting. There were four conditions: 1) Correlation, all word pairs in the benchmark, 2) Correlation, only those pairs in which both words were in resource, 3) RMSE, all word pairs in the benchmark, and 4) RMSE, only those pairs in which both words were in resource. If one of the words in a word pair was missing from the word vectors, then it was assigned a predicted cosine similarity of zero. (This only occurred with the RW dataset, and was limited to the all pairs conditions.)

4 Results

Table 2 shows the baseline word similarity performance according to the harmonic mean of the correlation measures and RMSE. As a reference, we include the NumberBatch (NB) vectors, which recently demonstrated state-of-the-art word similarity performance (Speer et al., 2017). Correlation and RMSE give similar baseline results among the vector sets and their ability to predict the four similarity benchmarks: NB performs the best. The exception is that SG scores a slightly better RMSE score on the MEN3K dataset.

4.1 All word pairs

Figure 1 shows the measured improvements in correlation due to retrofitting. This mirrors Faruqui et al. (2015)'s original finding that the PPDB offers the most improvements, and that grouping words by FrameNet frames (`FN`) usually leads to worse performance. Note that this finding is observed after correcting for the issue from Faruqui et al. which omitted data from FrameNet. This plot also suggests that using FrameNet frame elements (`FN-ANNO`) to group words is very detrimental to word vectors.

As shown in Figure 2, simply switching the evaluation metric to RMSE paints a much different picture. (Since RMSE measures error rather than improvement, the y-axis has been inverted so that improvement is still in the upward direction.) The most obvious difference is that according to RMSE all of the semantic resources appear to help. Compared to Figure 1, there is a noticeable boost in performance for `WN+`, especially when evaluated against RW. Remarkably, `FN-ANNO` almost completely flips polarity. The result is especially dramatic against the evaluation sets containing common words (i.e. MT771 and MEN3K): `FN-ANNO` goes from being the worst-performing resource to one of the best-performing resources.

4.2 Word pairs in resource

Figure 3 shows the measured improvements in correlation when considering only the word pairs in which both words were present in resources. The ranked order of the semantic resources is virtually the same. Note, however, that the measured performance of the relatively low-coverage resource, FrameNet (`FN`), has jumped considerably: in the RW with GloVe condition, it overtakes PPDB as the resource providing the best improvement.

Figure 4 measures the change of RMSE for the word pairs covered by the resources. FrameNet (`FN`) appears to yield a substantial gain in performance for the subset of the similarity judgments that it covers, and again emerges as the highest-performance resource when evaluated against RW. A direct comparison of the "all pairs" to "pairs in resource" figures shows that the scores of the other resources change very little. The difference is attenuated because these resources are much larger and therefore cover most of the words in the similarity datasets.

We interpret the jumps in performance from the "all pairs" to "pairs in resource" condition as evidence that evaluating a resource on word pairs containing a mixture of words within and outside of its vocabulary may obscure its benefits. Of course, low coverage is problematic if the goal is to improve word vectors on a large number of word judgments. The "pairs in resource" assessment is particularly antithetical to the spirit of RW, which is often employed to assess word vector coverage, and we admit that `FN` only contains 6.3% of the RW word pairs. However, we

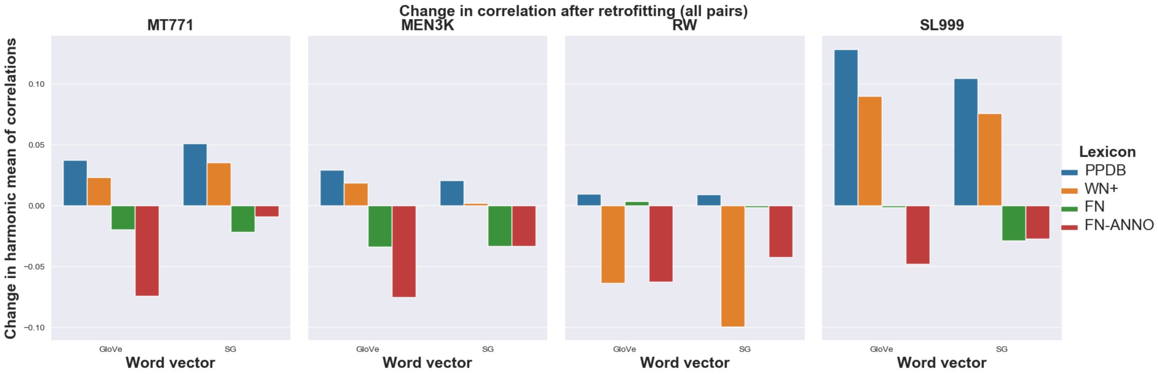


Figure 1: Change in correlation after retrofitting, considering all word pairs

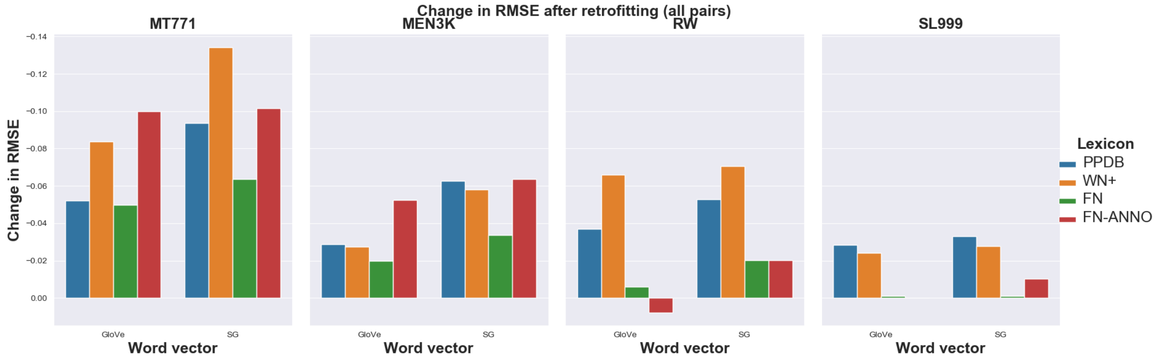


Figure 2: Change in Root-mean-square error after retrofitting, considering all word pairs

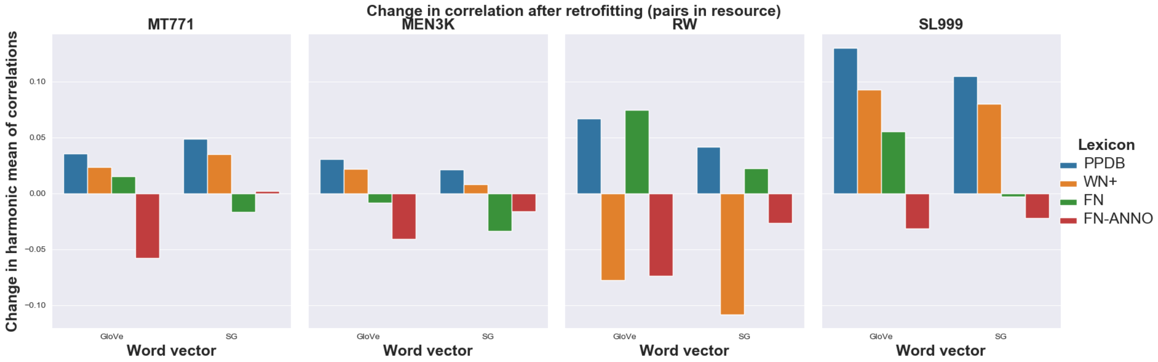


Figure 3: Change in correlation after retrofitting, considering only the word pairs in each resource

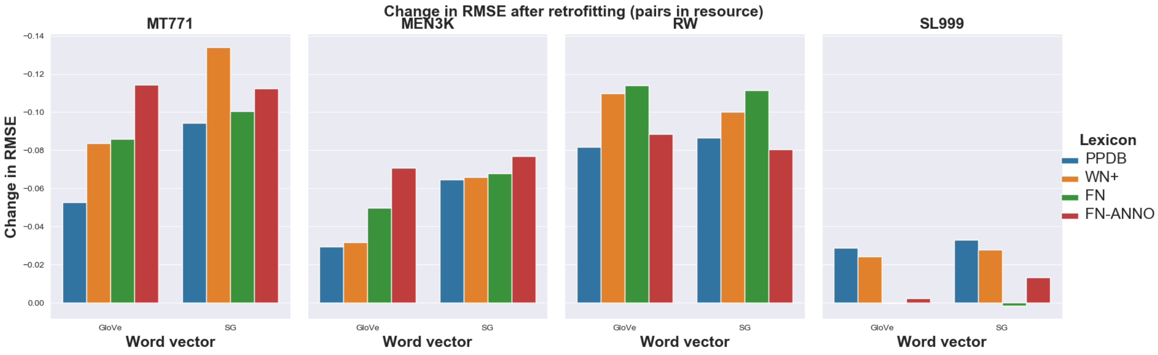


Figure 4: Change in Root-mean-square error after retrofitting, considering only the word pairs in each resource

would argue that there is an important difference between concluding that a semantic resource does not yield gains in retrofitting vs. concluding that the resource improves the quality of the vectors it covers.

4.3 SimLex

We note that our four conditions yield similar conclusions according to the SL999 evaluation set. `PPDB` and `WN+` consistently offer strong improvements, in contrast to `FN` and `FN-ANNO`. This is not surprising, and follows from the design principles underlying each resource: while PPDB and WordNet specifically group synonyms, FrameNet groups words which evoke the same semantic frame. In particular, some frames intentionally contain antonyms. As discussed above, the FrameNet groupings still appear useful in improving against MT771, MEN3K and RW, which have been argued to conflate association and similarity (Hill et al., 2015).

4.4 Further analysis

Our most striking finding is that correlation measures and RMSE occasionally yield opposite conclusions regarding the utility of semantic resources. How can the retrofitted data simultaneously show a drop in correlation and a gain in RMSE? To examine this further, we plotted the effects of retrofitting GloVe with `FN-ANNO` against the MT771 benchmark (Figure 5). Vector cosine similarity (x-axis) is plotted against the human similarity judgments (y-axis). The left and right panels compare the vector performance before and after retrofitting. Each point represents a single word pair in the MT771 dataset. The dashed line corresponds to a model which perfectly predicts the gold standard. Points are color-coded with respect to this line: green points mark word pairs whose computed cosine similarity moved closer to the human judgments, while red points indicate word pairs who moved in the opposite direction. A small number of blue points indicate predictions which were unaffected by retrofitting because the word pairs were not present in the resource.

The color-coding in Figure 5 helps illustrate how both Spearman correlation (a measure of goodness) and RMSE (a measure of error) decrease. Most of the points are green, which means that from the perspective of individual word pairs, the predictions from the retrofitted vectors are more in line with the gold standard. This is directly reflected in RMSE. However, while most of the mass moves closer to the dashed line, retrofitting increases the scatter of the points, resulting in a worse association between the vector cosine similarity human similarity judgments.

Three points are labeled in Figure 5 to show the effect of retrofitting on individual word similarity predictions. The diamond marks the word pair *find & occurrence*, which yields the most improvement according to MT771, with its absolute residual (i.e. distance from the human judgment) dropping 0.25. In comparison, the worst-performing word pair is *occasion & second*, marked with an X, whose residual increases by 0.14. This point is part of a noticeable band of red points located near the dotted line. Interestingly, for these points the predicted scores for the baseline word vectors were nearly correct, and retrofitting pushed them to overpredict similarity. The square marks *film & movie*, whose residual drops an almost imperceptible 0.003.

The reason that retrofitting may lead to a worse correlation but a better RMSE score stems from how these measures are computed from the data. Each word pair contributes independently to the RMSE score. Whether a word pair improves or decreases in performance, it is simply tallied onto the running RMSE score. In this case, it is irrelevant whether retrofitting leads to a large increase in scatter. In contrast, correlation measures are anchored to the sample means of the two variables. After retrofitting, there may be an increase in the scatter in the predicted cosine similarity values. Since on average the word pairs will be further away from the sample mean, there will be a drop in correlation. Put another way, a word pair's contribution to the correlation score depends on the positions of the all of the other word pairs.

The particularly large drop in correlation for `FN-ANNO` likely stems from the unusual heterogeneity of its groupings. For example, the word *film* occurs in the annotation data of 108 distinct FEs in FrameNet, and is grouped with dozens of varied words, such as *book*, *movie*, but also *DNA* and *meeting*. Each of the 108 retrofitting adjustments introduces some scatter. In contrast, the neighbors in other resources can be straightforwardly interpreted as related words, and each word will appear in a small number of groupings.

We note that while it may be instructive to track the performance of individual word pairs, it is dif-

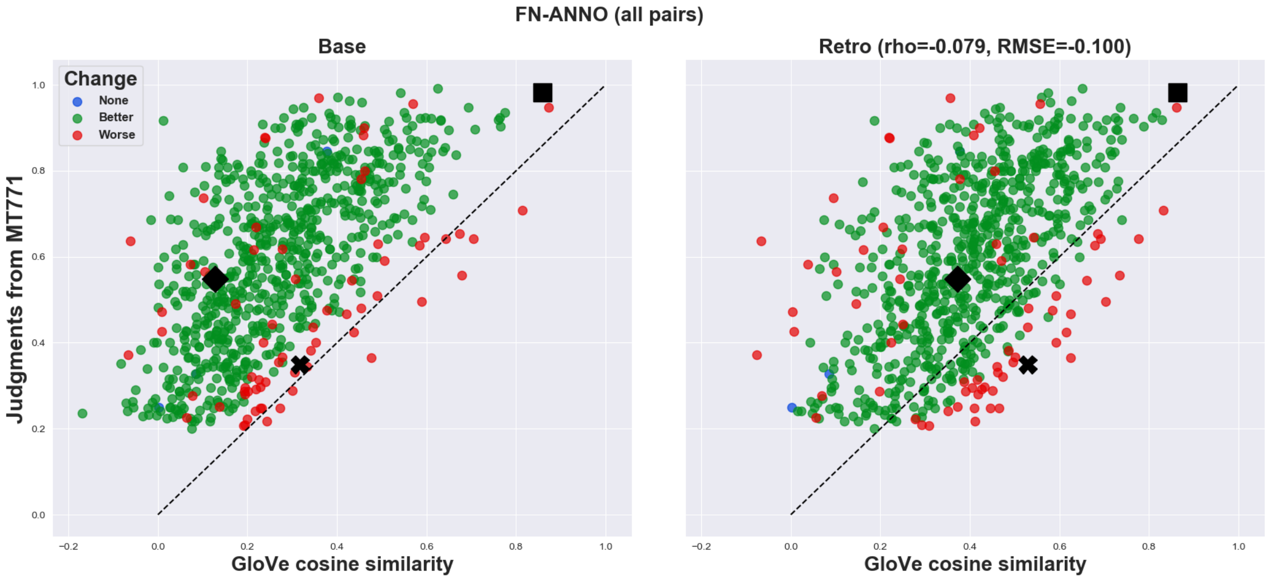


Figure 5: Effects of retrofitting GloVe by grouping nouns filling the same frame element in the FrameNet annotation data, considering all word pairs. Vector-computed similarity is plotted against the MT771 gold standard judgments using the original word vectors (left panel) and the retrofitted vectors (right panel). The dashed line illustrates a model which exactly predicts the human judgments. Predicted scores which moved closer to that line are colored green, while points which moved away from the line are colored red. Blue points represent word pairs which were not present in the resource, and so were unaffected by retrofitting. The changes in Spearman correlation and RMSE are shown above the right panel. The symbols are discussed in the text.

ficult to pinpoint the exact source of the change. For instance, in Figure 5 the words corresponding to the square (little change) and the X (worse change) are paired together, while the word pair linked to the diamond (best change) are not.

5 Related work

Faruqui et al. (2015) attributed FrameNet's comparatively poor performance to the fact that it groups words according to abstract concepts, noting that *push* and *grow* are in the same frame. Such an argument might explain why FrameNet does not yield gains in performance against SL999, which was designed to capture true similarity judgments. However, we have shown that conclusions on the other similarity benchmarks rest on the evaluation metric and on the types of word pairs considered. In the RMSE and "pairs in resource" condition, grouping words by FrameNet frames appears at least as useful as PPDB and WordNet. Alternatively, FrameNet can be interpreted as a useful resource for retrofitting the vectors of the words it contains as lexical units.

Our novel treatment of FrameNet groups nouns using its collection of sense-annotated sentences. Although all of the frame elements in these sentences were annotated by hand, the words filling the FEs are not, adding a component of randomness. Especially with more semantically general frames, frame elements can be realized by a large number of words. This contrasts with FrameNet frames, in which the placement of word senses are painstakingly deliberated, and a particular sense can only be put into one frame.

PropBank (Bonial et al., 2014) is a large semantically-annotated corpus. The semantic roles ("rolesets") in PropBank are defined with respect to individual verb and noun word senses. The types of words that fill these roles are presumably less varied than those that fill the semantically broader FrameNet frame elements. Additionally, PropBank is considerably larger than FrameNet. Consequently, we might predict that retrofitting word vectors to PropBank would yield stronger gains in word similarity judgment than to the FrameNet annotation data. We leave this task for future research.

Grouping nouns using the FrameNet annotation data led to large drops in correlation against word similarity benchmarks. However, these same data yielded large gains in RMSE performance. It might be inferred that semantic resources which have a similar stochastic component may result lower correlation. The PPDB is automatically generated, introducing a similar element of randomness, but this is curtailed by its conservative criteria: paraphrases must be attested as translation equivalents.

BabelNet (Navigli and Ponzetto, 2012) and ConceptNet (Speer et al., 2017) are knowledge resources derived from a number of collaboratively-constructed sources, such as Wikipedia and Wiktionary. Though their collaborative nature likely makes them less accurate than hand-curated resources such as WordNet, they have potential in improving the quality of word vectors (e.g. Speer and Chin, 2016). As we observed with `FN-ANNO`, RMSE may be a more informative measure of comparison than correlation in future retrofitting experiments involving heterogeneous resources.

More generally, there does not seem to be a strong theoretical reason to prefer correlation-based measures over residual-based ones. Although the current practice is to report the Spearman's rank correlation coefficient between the vector cosine similarities and human word similarity judgments, for over a decade the standard was to report Pearson product-moment correlation coefficient. When Resnik (1995) pioneered the technique of comparing computed measures of similarity with human similarity ratings, he used (Pearson) correlation as "one reasonable way to judge [computational measures of semantic similarity]".

The switch to Spearman correlation appears to have occurred in Gabrilovich and Markovitch (2007), who employed it without comment. Agirre et al. (2009) did provide a justification, saying, "In our belief Pearson is less informative, as the Pearson correlation suffers much when the scores of two systems are not linearly correlated, something which happens often due to the different nature of the techniques applied." Unfortunately, Agirre et al. (2009) mischaracterized the popularity of Spearman correlation by claiming that all researchers have used Spearman in evaluating WordSim-353 dataset (Finkelstein et al., 2002). This likely stems from a misinterpretation of Gabrilovich and Markovitch's Table 4, which compares their methodology with earlier studies using Spearman correlation. The latter authors apparently recomputed word relatedness with the associated algorithms, as the cited studies report Pearson correlation values.

Willmott (1981; 1982) specifically argues that Pearson correlation should not be used to evaluate model performance, and that RMSE is superior at comparing observed and simulated data.[3] However, as far as we know, no previous work has seriously considered evaluating the performance of computed word similarity scores using RMSE. Reliance on Spearman correlation may lead to incorrect conclusions regarding the quality of word vectors.

6 Conclusion

Retrofitting distributional word vectors using relational information in semantic resources can yield improvements in word similarity performance. Our fine-grained analysis of the original retrofitting process shows that 1) the evaluation metric matters: root-mean-square error (RMSE) is more sensitive to gains in performance than correlation measures; and 2) coverage matters: improvements offered by resources are highly dependent on their coverage of the evaluation benchmark. Future attempts to improve word vectors can only succeed if gains in word vector performance are inspected carefully.

Acknowledgments

This research was supported in part by the Defense Threat Reduction Agency (DTRA). Disclaimer: The project or effort depicted was or is sponsored by the Department of the Defense, Defense Threat Reduction Agency. The content of the information does not necessarily reflect the position or the policy of the federal government, and no official endorsement should be inferred.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL*

---

[3] Willmott and Matsuura (2005) list several advantages of a related metric, mean absolute error (MAE), over RMSE. In our experiments, there were no qualitative differences between MAE and RMSE.

2004 on Interactive poster and demonstration sessions, page 31. Association for Computational Linguistics.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In *LREC*, pages 3013–3019.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1378–1383.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, pages 1606–1615.

Charles J Fillmore and Collin Baker. 2010. A frames approach to semantic analysis. In *The Oxford handbook of linguistic analysis*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, volume 7, pages 1606–1611.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 34–45.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 683–693.

Benjamin J Lengerich, Andrew L Maas, and Christopher Potts. 2017. Retrofitting distributional embeddings to knowledge graphs with functional relations. *arXiv preprint arXiv:1708.00112*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 1, pages 448–453.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Robert Speer and Joshua Chin. 2016. An ensemble method to produce high-quality word embeddings. *arXiv preprint arXiv:1604.01692*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451. Association for the Advancement of Artificial Intelligence.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Cort J Willmott. 1981. On the validation of models. *Physical geography*, 2(2):184–194.

Cort J Willmott. 1982. Some comments on the evaluation of model performance. *Bulletin of the American Meteorological Society*, 63(11):1309–1313.

Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Linguistic Knowledge and Transferability of Contextual Representations

Nelson F. Liu♠♡* **Matt Gardner**♣ **Yonatan Belinkov**◇
Matthew E. Peters♣ **Noah A. Smith**♠♣
♠Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
♡Department of Linguistics, University of Washington, Seattle, WA, USA
♣Allen Institute for Artificial Intelligence, Seattle, WA, USA
◇Harvard John A. Paulson School of Engineering and Applied Sciences and
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
`{nfliu,nasmith}@cs.washington.edu`
`{mattg,matthewp}@allenai.org`, `belinkov@seas.harvard.edu`

Abstract

Contextual word representations derived from large-scale neural language models are successful across a diverse set of NLP tasks, suggesting that they encode useful and transferable features of language. To shed light on the linguistic knowledge they capture, we study the representations produced by several recent pretrained contextualizers (variants of ELMo, the OpenAI transformer language model, and BERT) with a suite of sixteen diverse probing tasks. We find that linear models trained on top of frozen contextual representations are competitive with state-of-the-art task-specific models in many cases, but fail on tasks requiring fine-grained linguistic knowledge (e.g., conjunct identification). To investigate the transferability of contextual word representations, we quantify differences in the transferability of individual layers within contextualizers, especially between recurrent neural networks (RNNs) and transformers. For instance, higher layers of RNNs are more task-specific, while transformer layers do not exhibit the same monotonic trend. In addition, to better understand what makes contextual word representations transferable, we compare language model pretraining with eleven supervised pretraining tasks. For any given task, pretraining on a closely related task yields better performance than language model pretraining (which is better on average) when the pretraining dataset is fixed. However, language model pretraining on *more data* gives the best results.

1 Introduction

Pretrained word representations (Mikolov et al., 2013; Pennington et al., 2014) are a key component of state-of-the-art neural NLP models. Traditionally, these word vectors are static—a single vector is assigned to each word. Recent work has explored *contextual* word representations (henceforth: CWRs), which assign each word a vector that is a function of the entire input sequence; this enables them to model the use of words in context. CWRs are typically the outputs of a neural network (which we call a *contextualizer*) trained on tasks with large datasets, such as machine translation (McCann et al., 2017) and language modeling (Peters et al., 2018a). CWRs are extraordinarily effective—using them in place of traditional static word vectors within the latest models leads to large gains across a variety of NLP tasks.

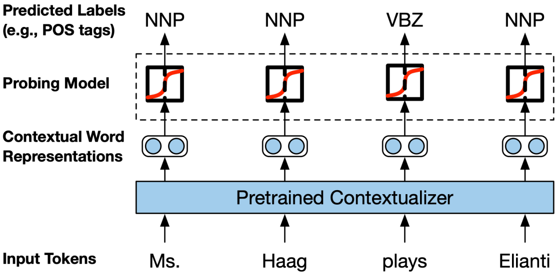


Figure 1: An illustration of the probing model setup used to study the linguistic knowledge within contextual word representations.

The broad success of CWRs indicates that they encode useful, transferable features of language. However, their *linguistic knowledge* and *transferability* are not yet well understood.

Recent work has explored the linguistic knowledge captured by language models and neural machine translation systems, but these studies often focus on a single phenomenon, e.g., knowledge of hierarchical syntax (Blevins et al., 2018) or morphology (Belinkov et al., 2017a). We extend prior work by studying CWRs with a diverse set of sixteen probing tasks designed to assess a wide array of phenomena, such as coreference, knowledge of semantic relations, and entity information, among

---

[*] Work done while at the Allen Institute for Artificial Intelligence.

Proceedings of NAACL-HLT 2019, pages 1073–1094
Minneapolis, Minnesota, June 2 - June 7, 2019.

others. The result is a broader view of the linguistic knowledge encoded within CWRs.

With respect to transferability, pretraining contextualizers on the language modeling task has had the most empirical success, but we can also consider pretraining contextualizers with other supervised objectives and probing their linguistic knowledge. We examine how the pretraining task affects the linguistic knowledge learned, considering twelve pretraining tasks and assessing transferability to nine target tasks.

Better understanding the linguistic knowledge and transferability of CWRs is necessary for their principled enhancement through new encoder architectures and pretraining tasks that build upon their strengths or alleviate their weaknesses (Linzen, 2018). This paper asks and answers:

1. What features of language do these vectors capture, and what do they miss? (§4)
2. How and why does transferability vary across representation layers in contextualizers? (§5)
3. How does the choice of pretraining task affect the vectors' learned linguistic knowledge and transferability? (§6)

We use probing models[1] (Shi et al., 2016b; Adi et al., 2017; Hupkes et al., 2018; Belinkov and Glass, 2019) to analyze the linguistic information within CWRs. Concretely, we generate features for words from pretrained contextualizers and train a model to make predictions from those features alone (Figure 1). If a simple model can be trained to predict linguistic information about a word (e.g., its part-of-speech tag) or a pair of words (e.g., their semantic relation) from the CWR(s) alone, we can reasonably conclude that the CWR(s) encode this information.

Our analysis reveals interesting insights such as:

1. Linear models trained on top of frozen CWRs are competitive with state-of-the-art task-specific models in many cases, but fail on tasks requiring fine-grained linguistic knowledge. In these cases, we show that *task-trained contextual features* greatly help with encoding the requisite knowledge.
2. The first layer output of long short-term memory (LSTM) recurrent neural networks is consistently the most transferable, whereas it is the middle layers for transformers.
3. Higher layers in LSTMs are more task-specific (and thus less general), while the transformer layers do not exhibit this same monotonic increase in task-specificity.
4. Language model pretraining yields representations that are more transferable *in general* than eleven other candidate pretraining tasks, though pretraining on related tasks yields the strongest results for individual end tasks.

2 Probing Tasks

We construct a suite of sixteen diverse English probing tasks and use it to better understand the linguistic knowledge contained within CWRs. In contrast to previous studies that analyze the properties and task performance of sentence embeddings (Adi et al., 2017; Conneau et al., 2018, *inter alia*), we specifically focus on understanding the CWRs of individual or pairs of words. We release this analysis toolkit to support future work in probing the contents of representations.[2] See Appendix A for details about task setup.

2.1 Token Labeling

The majority of past work in probing the internal representations of neural models has examined various token labeling tasks, where a decision is made independently for each token (Belinkov et al., 2017a,b; Blevins et al., 2018, *inter alia*). We synthesize these disparate studies and build upon them by proposing additional probing tasks.

The **part-of-speech tagging (POS)** task assesses whether CWRs capture basic syntax. We experiment with two standard datasets: the Penn Treebank (PTB; Marcus et al., 1993) and the Universal Dependencies English Web Treebank (UD-EWT; Silveira et al., 2014).

The **CCG supertagging (CCG)** task assesses the vectors' fine-grained information about the syntactic roles of words in context. It is considered "almost parsing" (Bangalore and Joshi, 1999), since a sequence of supertags maps a sentence to a small set of possible parses. We use CCGbank (Hockenmaier and Steedman, 2007), a conversion of the PTB into CCG derivations.

The **syntactic constituency ancestor tagging** tasks are designed to probe the vectors' knowledge of hierarchical syntax. For a given word, the probing model is trained to predict the constituent la-

---

[1] Sometimes called auxiliary or diagnostic classifiers.

[2] `http://nelsonliu.me/papers/contextual-repr-analysis`

(1) I was booked **for**/Duration 2 nights **at**/Locus this hotel **in**/Time Oct 2007 .

(2) I went **to**/Goal ohm **after**/Explanation⤳Time reading some **of**/Quantity⤳Whole the reviews .

(3) It was very upsetting to see this kind **of**/Species behavior especially **in_front_of**/Locus **my**/SocialRel⤳Gestalt four year_old .

Figure 2: Annotated sentences from the STREUSLE 4.0 corpus, used in the preposition supersense disambiguation task. Prepositions are marked by boldface, immediately followed by their labeled function. If applicable, ⤳ precedes the preposition's labeled role. Figure reproduced from Schneider et al. (2018).

bel of its parent **(Parent)**, grandparent **(GParent)**, or great-grandparent **(GGParent)** in the phrase-structure tree (from the PTB).

In the **semantic tagging** task (**ST**), tokens are assigned labels that reflect their semantic role in context. These semantic tags assess lexical semantics, and they abstract over redundant POS distinctions and disambiguate useful cases within POS tags. We use the dataset of Bjerva et al. (2016); the tagset has since been developed as part of the Parallel Meaning Bank (Abzianidze et al., 2017).

Preposition supersense disambiguation is the task of classifying a preposition's lexical semantic contribution (the function; **PS-fxn**) and the semantic role or relation it mediates (the role; **PS-role**). This task is a specialized kind of word sense disambiguation, and examines one facet of lexical semantic knowledge. In contrast to the tagging tasks above, the model is trained and evaluated on single-token prepositions (rather than making a decision for every token in a sequence). We use the STREUSLE 4.0 corpus (Schneider et al., 2018); example sentences appear in Figure 2.

The **event factuality (EF)** task involves labeling phrases with the factuality of the events they describe (Saurí and Pustejovsky, 2009, 2012; de Marneffe et al., 2012). For instance, in the following example reproduced from Rudinger et al. (2018), "(1a) conveys that the leaving didn't happen, while the superficially similar (1b) does not".

(1) a. Jo didn't remember to **leave**.
b. Jo didn't remember **leaving**.

We use the Universal Decompositional Semantics It Happened v2 dataset (Rudinger et al., 2018), and the model is trained to predict a (non)factuality value in the range $[-3, 3]$. Unlike the tagging tasks above, this task is treated as a regression problem, where a prediction is made only for tokens corresponding to events (rather than every token in a sequence). Performance is measured using Pearson correlation (r); we report ($r \times 100$) so metrics for all tasks fall between 0 and 100.

2.2 Segmentation

Several of our probing tasks involve segmentation using BIO or IO tags. Here the model is trained to predict labels from *only* a single word's CWR.

Syntactic chunking (Chunk) tests whether CWRs contain notions of spans and boundaries; the task is to segment text into shallow constituent chunks. We use the CoNLL 2000 shared task dataset (Tjong Kim Sang and Buchholz, 2000).

Named entity recognition (NER) examines whether CWRs encode information about entity types. We use the CoNLL 2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003).

Grammatical error detection (GED) is the task of identifying tokens which need to be edited in order to produce a grammatically correct sentence. Given that CWRs are extracted from models trained on large amounts of grammatical text, this task assesses whether embeddings encode features that indicate anomalies in their input (in this case, ungrammaticality). We use the First Certificate in English (Yannakoudakis et al., 2011) dataset, converted into sequence-labeling format by Rei and Yannakoudakis (2016).

The **conjunct identification (Conj)** task challenges the model to identify the tokens that comprise the conjuncts in a coordination construction. Doing so requires highly specific syntactic knowledge. The data comes from the coordination-annotated PTB of Ficler and Goldberg (2016).

2.3 Pairwise Relations

We also design probing tasks that examine whether relationships *between* words are encoded in CWRs. In these tasks, given a word pair w_1, w_2, we input $[w_1, w_2, w_1 \odot w_2]$ into the probing model; it is trained to predict information about the relation between the tokens (Belinkov, 2018).

We distinguish between **arc prediction** and **arc classification** tasks. Arc prediction is a binary classification task, where the model is trained to identify *whether* a relation exists between two tokens. Arc classification is a multiclass classification task, where the model is provided with two tokens that are linked via some relationship and trained to identify *how* they are related.

For example, in the **syntactic dependency arc prediction** task, the model is given the representations of two tokens (w_a, w_b) and trained to predict whether the sentence's syntactic dependency parse contains a dependency arc with w_a as the head and w_b as the modifier. The **syntactic dependency arc classification** task presents the model with the representations of two tokens (w_{head}, w_{mod}), where w_{mod} is the modifier of w_{head}, and the model is trained to predict the type of syntactic relation that link them (the label on that dependency arc). We use the PTB (converted to UD) and the UD-EWT.

Similarly, **semantic dependency arc prediction** trains the model to predict whether two tokens are connected by a semantic dependency arc, while the **semantic dependency arc classification** task trains models to classify the semantic relations between tokens. We use the dataset from the SemEval 2015 shared task (Oepen et al., 2015) with the DELPH-IN MRS-Derived Semantic Dependencies (DM) target representation.

The syntactic and semantic dependency arc prediction and classification tasks are closely related to state-of-the-art models for semantic and syntactic dependency parsing, which score pairs of CWRs to make head attachment and arc labeling decisions (Dozat and Manning, 2016, 2018).

To generate negative examples for the dependency arc prediction tasks, we take each positive example (w_{head}, w_{mod}) and generate a new negative example (w_{rand}, w_{mod}). w_{rand} is a random token in the sentence that is not the head of w_{mod}. Thus, the datasets used in these tasks are balanced.

We also consider a **coreference arc prediction** task, where the model is trained to predict whether two entities corefer from their CWRs. We use the dataset from the CoNLL 2012 shared task (Pradhan et al., 2012). To generate negative examples, we follow a similar procedure as the dependency arc prediction tasks: given a positive example (w_a, w_b), where w_b occurs after w_a and the two tokens share a coreference cluster, we create a negative example (w_{random_entity}, w_b), where w_{random_entity} is a token that occurs before w_b and belongs to a different coreference cluster.

3 Models

Probing Model We use a linear model as our probing model; limiting its capacity enables us to focus on what information can be *easily* extracted from CWRs. See Appendix B for probing model training hyperparameters and other details.

Contextualizers We study six publicly-available models for contextualized word representation in English.

ELMo (Peters et al., 2018a) concatenates the output of two contextualizers independently trained on the bidirectional language modeling (biLM) task. **ELMo (original)** uses a 2-layer LSTM for contextualization. We also study two variations from Peters et al. (2018b): **ELMo (4-layer)** uses a 4-layer LSTM, and **ELMo (transformer)** uses a 6-layer transformer (Vaswani et al., 2017). Each of these models is trained on 800M tokens of sentence-shuffled newswire text (the 1 Billion Word Benchmark; Chelba et al., 2014).

The **OpenAI transformer** (Radford et al., 2018) is a left-to-right 12-layer transformer language model trained on 800M tokens of contiguous text from over 7,000 unique unpublished books (BookCorpus; Zhu et al., 2015).

BERT (Devlin et al., 2018) uses a bidirectional transformer jointly trained on a masked language modeling task and a next sentence prediction task. The model is trained on BookCorpus and the English Wikipedia, a total of approximately 3300M tokens. We study **BERT (base, cased)**, which uses a 12-layer transformer, and **BERT (large, cased)**, which uses a 24-layer transformer.

4 Pretrained Contextualizer Comparison

To better understand the linguistic knowledge captured by pretrained contextualizers, we analyze each of their layers with our set of probing tasks. These contextualizers differ in many respects, and it is outside the scope of this work to control for all differences between them. We focus on probing the models that are available to us, leaving a more systematic comparison of training regimes and model architectures to future work.

4.1 Experimental Setup

Our probing models are trained on the representations produced by the individual layers of each contextualizer. We also compare to a linear probing model trained on noncontextual vectors (300-dimensional GloVe trained on the cased Common Crawl; Pennington et al., 2014) to assess the gains from contextualization.

Pretrained Representation			POS						Supersense ID		
	Avg.	CCG	PTB	EWT	Chunk	NER	ST	GED	PS-Role	PS-Fxn	EF
ELMo (original) best layer	81.58	93.31	97.26	95.61	90.04	82.85	93.82	29.37	75.44	84.87	73.20
ELMo (4-layer) best layer	81.58	93.81	**97.31**	95.60	89.78	82.06	**94.18**	29.24	74.78	85.96	73.03
ELMo (transformer) best layer	80.97	92.68	97.09	95.13	93.06	81.21	93.78	30.80	72.81	82.24	70.88
OpenAI transformer best layer	75.01	82.69	93.82	91.28	86.06	58.14	87.81	33.10	66.23	76.97	74.03
BERT (base, cased) best layer	84.09	93.67	96.95	95.21	92.64	82.71	93.72	43.30	**79.61**	87.94	75.11
BERT (large, cased) best layer	**85.07**	**94.28**	96.73	**95.80**	**93.64**	**84.44**	93.83	**46.46**	79.17	**90.13**	**76.25**
GloVe (840B.300d)	59.94	71.58	90.49	83.93	62.28	53.22	80.92	14.94	40.79	51.54	49.70
Previous state of the art (without pretraining)	83.44	94.7	97.96	95.82	95.77	91.38	95.15	39.83	66.89	78.29	77.10

Table 1: Performance of the best layerwise linear probing model for each contextualizer compared against a GloVe-based linear probing baseline and the previous state of the art. The best contextualizer for each task is bolded. Results for all layers on all tasks, and papers describing the prior state of the art, are given in Appendix D.

4.2 Results and Discussion

Table 1 compares each contextualizer's best-performing probing model with the GloVe baseline and the previous state of the art for the task (excluding methods that use pretrained CWRs).[3,4]

With just a linear model, we can readily extract much of the information needed for high performance on various NLP tasks. In all cases, CWRs perform significantly better than the noncontextual baseline. Indeed, we often see probing models rivaling or exceeding the performance of (often carefully tuned and task-specific) state-of-the-art models. In particular, the linear probing model surpasses the published state of the art for grammatical error detection and preposition supersense identification (both role and function).

Comparing the ELMo-based contextualizers, we see that ELMo (4-layer) and ELMo (original) are essentially even, though both recurrent models outperform ELMo (transformer). We also see that the OpenAI transformer significantly underperforms the ELMo models and BERT. Given that it is also the only model trained in a unidirectional (left-to-right) fashion, this reaffirms that bidirectionality is a crucial component for the highest-quality contextualizers (Devlin et al., 2018). In addition, the OpenAI transformer is the only model trained on lowercased text, which hinders its performance on tasks like NER. BERT significantly improves over the ELMo and OpenAI models.

Our probing task results indicate that current methods for CWR do not capture much transferable information about entities and coreference phenomena in their input (e.g., the NER results in Table 1 and the coreference arc prediction results in Appendix D). To alleviate this weakness, future work could augment pretrained contextualizers with explicit entity representations (Ji et al., 2017; Yang et al., 2017; Bosselut et al., 2017).

Probing Failures While probing models are at or near state-of-the-art performance across a number of tasks, they also do not perform as well on several others, including NER, grammatical error detection, and conjunct identification. This may occur because (1) the CWR simply does not encode the pertinent information or any predictive correlates, or (2) the probing model does not have the capacity necessary to extract the information or predictive correlates from the vector. In the former case, learning *task-specific contextual features* might be necessary for encoding the requisite task-specific information into the CWRs. Learning task-specific contextual features with a contextual probing model also helps with (2), but we would expect the results to be comparable to increasing the probing model's capacity.

To better understand the failures of our probing model, we experiment with (1) a contextual probing model that uses a task-trained LSTM (unidirectional, 200 hidden units) before the linear output layer (thus adding task-specific contextualization) or (2) replacing the linear probing model with a multilayer perceptron (MLP; adding more parameters to the probing model: a single 1024d hidden layer activated by ReLU). These alternate probing models have nearly the same number of parameters (LSTM + linear has slightly fewer).

We also compare to a full-featured model to

---

[3] See Appendix C for references to the previous state of the art (without pretraining).

[4] For brevity, in this section we omit probing tasks that cannot be compared to prior work. See Appendix D for pretrained contextualizer performance for all layers and all tasks.

Probing Model	NER	GED	Conj	GGParent
Linear	82.85	29.37	38.72	67.50
MLP (1024d)	87.19	47.45	55.09	78.80
LSTM (200d) + Linear	**88.08**	**48.90**	**78.21**	**84.96**
BiLSTM (512d) + MLP (1024d)	90.05	48.34	87.07	90.38

Table 2: Comparison of different probing models trained on ELMo (original); best-performing probing model is bolded. Results for each probing model are from the highest-performing contextualizer layer. Enabling probing models to learn task-specific contextual features (with LSTMs) yields outsized benefits in tasks requiring highly specific information.

estimate an upper bound on performance for our probing setup. In this model, the CWRs are inputs to a 2-layer BiLSTM with 512 hidden units, and the output is fed into a MLP with a single 1024-dimensional hidden layer activated by a ReLU to predict a label. A similar model, augmented with a conditional random field (CRF; Lafferty et al., 2001), achieved state-of-the-art results on the CoNLL 2003 NER dataset (Peters et al., 2018a). We remove the CRF, since other probing models have no global context.

For this experiment, we focus on the ELMo (original) pretrained contextualizer. Table 2 presents the performance of the best layer within each alternative probing model on the two tasks with the largest gap between the linear probing model and state-of-the-art methods: NER and grammatical error detection. We also include great-grandparent prediction and conjunct identification, two tasks that require highly specific syntactic knowledge. In all cases, we see that adding more parameters (either by replacing the linear model with a MLP, or using a contextual probing model) leads to significant gains over the linear probing model. On NER and grammatical error detection, we observe very similar performance between the MLP and LSTM + Linear models—this indicates that the probing model simply needed more capacity to extract the necessary information from the CWRs. On conjunct identification and great-grandparent prediction, two tasks that probe syntactic knowledge unlikely to be encoded in CWRs, adding parameters as a task-trained component of our probing model leads to large gains over simply adding parameters to the probing model. This indicates that the pretrained contextualizers do not capture the information necessary for the task, since such information is learnable by a task-specific contextualizer.

This analysis also reveals insights about contextualizer fine-tuning, which seeks to specialize the CWRs for an end task (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). Our results confirm that task-trained contextualization is important when the end task requires specific information that may not be captured by the pretraining task (§4). However, such end-task–specific contextualization can come from either fine-tuning CWRs or using fixed output features as inputs to a task-trained contextualizer; Peters et al. (2019) begins to explore when each approach should be applied.

5 Analyzing Layerwise Transferability

We quantify the transferability of CWRs by how well they can do on the range of probing tasks—representations that are more transferable will perform better than alternatives across tasks. When analyzing the representations produced by each layer of pretrained contextualizers, we observe marked patterns in layerwise transferability (Figure 3). The first layer of contextualization in recurrent models (original and 4-layer ELMo) is consistently the most transferable, even outperforming a scalar mix of layers on most tasks (see Appendix D for scalar mix results). Schuster et al. (2019) see the same trend in English dependency parsing. By contrast, transformer-based contextualizers have no single most-transferable layer; the best performing layer for each task varies, and is usually near the middle. Accordingly, a scalar mix of transformer layers outperforms the best individual layer on most tasks (see Appendix D).

Pretraining encourages the model to encode pretraining-task–specific information; they learn transferable features incidentally. We hypothesize that this is an inherent trade-off—since these models used fixed-sized vector representations, task-specificity comes at the cost of generality and transferability. To investigate the task-specificity of the representations generated by each contextualizer layer, we assess how informative each layer of representation is for the pretraining task, essentially treating it as a probe.

5.1 Experimental Setup

We focus on the ELMo-based models, since the authors have released code for training their con-

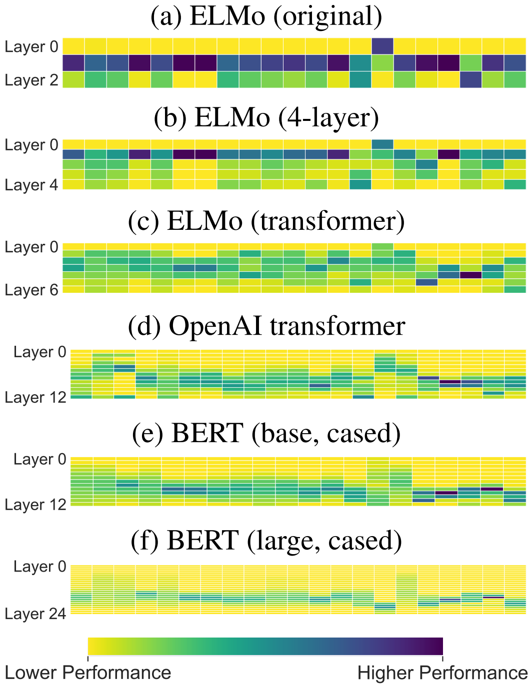


Figure 3: A visualization of layerwise patterns in task performance. Each column represents a probing task, and each row represents a contextualizer layer.

textualizers. Furthermore, the ELMo-based models facilitate a controlled comparison—they only differ in the contextualizer architecture used.

We evaluate how well CWR features perform the pretraining task—bidirectional language modeling. Specifically, we take the pretrained representations for each layer and relearn the language model softmax classifiers used to predict the next and previous token. The ELMo models are trained on the Billion Word Benchmark, so we retrain the softmax classifier on similar data to mitigate any possible effects from domain shift. We split the held-out portion of the Billion Word Benchmark into train (80%, 6.2M tokens) and evaluation (20%, 1.6M tokens) sets and use this data to retrain and evaluate the softmax classifiers. We expect that biLM perplexity will be lower when training the softmax classifiers on representations from layers that capture more information about the pretraining task.

5.2 Results and Discussion

Figure 4 presents the performance of softmax classifiers trained to perform the bidirectional language modeling task, given just the CWRs as input. We notice that higher layers in recurrent models consistently achieve lower perplexities. Interestingly, we see that layers 1 and 2 in the 4-layer ELMo model have very similar performance—this warrants further exploration. On the other hand, the layers of the ELMo (transformer) model do not exhibit such a monotonic increase. While the topmost layer is best (which we expected, since this is the vector originally fed into a softmax classifier during pretraining), the middle layers show varying performance. Across all models, the representations that are better-suited for language modeling are also those that exhibit worse probing task performance (Figure 3), indicating that contextualizer layers trade off between encoding general and task-specific features.

These results also reveal a difference in the layerwise behavior of LSTMs and transformers; moving up the LSTM layers yields more task-specific representations, but the same does not hold for transformers. Better understanding the differences between transformers and LSTMs is an active area of research (Chen et al., 2018; Tang et al., 2018), and we leave further exploration of these observations to future work.

These observations motivate the gradual unfreezing method of Howard and Ruder (2018), where the model layers are progressively unfrozen (starting from the final layer) during the fine-tuning process. Given our observation that higher-level LSTM layers are less general (and more pretraining task-specific), they likely have to be fine-tuned a bit more in order to make them appropriately task specific. Meanwhile, the base layer of the LSTM already learns highly transferable features, and may not benefit from fine-tuning.

6 Transferring Between Tasks

Successful pretrained contextualizers have used self-supervised tasks such as bidirectional language modeling (Peters et al., 2018a) and next sentence prediction (Devlin et al., 2018), which enable the use of large, unannotated text corpora. However, contextualizers can also be pretrained on explicitly supervised objectives, as done in pretrained *sentence* embedding methods (Conneau et al., 2017). To better understand how the choice of pretraining task affects the linguistic knowledge within and transferability of CWRs, we compare pretraining on a range of different explicitly-supervised tasks with bidirectional language model pretraining.

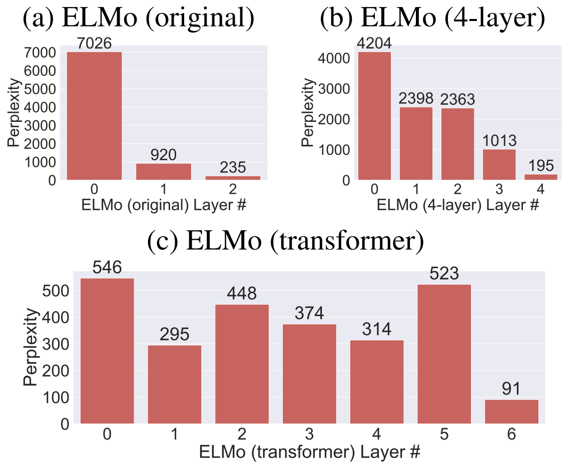


Figure 4: Bidirectional language modeling as a probe: average of forward and backward perplexity (lower is better) of each ELMo contextualizer layer. We see a monotonic decrease in BiLM perplexity when trained on the outputs of higher LSTM layers, but transformer layers do not exhibit the same pattern.

Pretraining Task	Layer Average Target Task Performance			
	0	1	2	Mix
CCG	56.70	64.45	63.71	66.06
Chunk	54.27	62.69	63.25	63.96
POS	56.21	63.86	64.15	65.13
Parent	54.57	62.46	61.67	64.31
GParent	55.50	62.94	62.91	64.96
GGParent	54.83	61.10	59.84	63.81
Syn. Arc Prediction	53.63	59.94	58.62	62.43
Syn. Arc Classification	56.15	64.41	63.60	66.07
Sem. Arc Prediction	53.19	54.69	53.04	59.84
Sem. Arc Classification	56.28	62.41	61.47	64.67
Conj	50.24	49.93	48.42	56.92
BiLM	66.53	65.91	65.82	66.49
GloVe (840B.300d)	60.55			
Untrained ELMo (original)	52.14	39.26	39.39	54.42
ELMo (original) (BiLM on 1B Benchmark)	64.40	79.05	77.72	78.90

Table 3: Performance (averaged across target tasks) of contextualizers pretrained on a variety of tasks.

6.1 Experimental Setup

To ensure a controlled comparison of different pretraining tasks, we fix the contextualizer's architecture and pretraining dataset. All of our contextualizers use the ELMo (original) architecture, and the training data from each of the pretraining tasks is taken from the PTB. Each of the (identical) models thus see the same tokens, but the supervision signal differs.[5] We compare to (1) a noncontextual baseline (GloVe) to assess the effect of contextualization, (2) a randomly-initialized, untrained ELMo (original) baseline to measure the effect of pretraining, and (3) the ELMo (original) model pretrained on the Billion Word Benchmark to examine the effect of training the bidirectional language model on more data.

6.2 Results and Discussion

Table 3 presents the average target task performance of each layer in contextualizers pretrained on twelve different tasks (biLM and the eleven tasks from §2 with PTB annotations). Bidirectional language modeling pretraining is the most effective on average. However, the settings that achieve the highest performance for individual target tasks often involve transferring between related tasks (not shown in Table 3; see Appendix E). For example, when probing CWRs on the syntactic dependency arc classification (EWT) task, we see the largest gains from pretraining on the task itself, but with a different dataset (PTB). However, pretraining on syntactic dependency arc prediction (PTB), CCG supertagging, chunking, the ancestor prediction tasks, and semantic dependency arc classification all give better performance than bidirectional language model pretraining.

Although related task transfer is beneficial, we naturally see stronger results from training on more data (the ELMo original BiLM trained on the Billion Word Benchmark). This indicates that the transferability of pretrained CWRs relies on pretraining on large corpora, emphasizing the utility and importance of self-supervised pretraining.

Furthermore, layer 0 of the BiLM is the highest-performing single layer among PTB-pretrained contextualizers. This observation suggests that lexical information is the source of the language model's initial generalizability, since layer 0 is the output of a character-level convolutional neural network with no token-level contextual information.

7 Related Work

Methodologically, our work is most similar to Shi et al. (2016b), Adi et al. (2017), and Hupkes et al. (2018), who use the internal representations of neural models to predict properties of interest. Conneau et al. (2018) construct probing tasks to study the linguistic properties of sentence embed-

---

[5] We omit the OpenAI transformer and BERT from this comparison, since code for pretraining these contextualizers is not publicly available.

ding methods. We focus on contextual word representations, which have achieved state-of-the-art results on a variety of tasks, and examine a broader range of linguistic knowledge.

In contemporaneous work, Tenney et al. (2019) evaluate CoVe (McCann et al., 2017), ELMo (Peters et al., 2018a), the OpenAI Transformer (Radford et al., 2018), and BERT (Devlin et al., 2018) on a variety of sub-sentence linguistic analysis tasks. Their results also suggest that the aforementioned pretrained models for contextualized word representation encode stronger notions of syntax than higher-level semantics. They also find that using a scalar mix of output layers is particularly effective in deep transformer-based models, aligned with our own probing results and our observation that transformers tend to encode transferable features in their intermediate layers. Furthermore, they find that ELMo's performance cannot be explained by a model with access to only local context, indicating that ELMo encodes linguistic features from distant tokens.

Several other papers have examined how architecture design and choice of pretraining task affect the quality of learned CWRs. Peters et al. (2018b) study how the choice of neural architecture influences the end-task performance and qualitative properties of CWRs derived from bidirectional language models (ELMo). Bowman et al. (2018) compare a variety of pretraining tasks and explore the the impact of multitask learning.

Prior work has employed a variety of other methods to study the learned representations in neural models, such as directly examining the activations of individual neurons (Karpathy et al., 2015; Li et al., 2015; Shi et al., 2016a, *inter alia*), ablating components of the model and dataset (Kuncoro et al., 2017; Gaddy et al., 2018; Khandelwal et al., 2018), or interpreting attention mechanisms (Bahdanau et al., 2015); see Belinkov and Glass (2019) for a recent survey. One particularly relevant line of work involves the construction of synthetic tasks that a model can only solve if it captures a particular phenomenon (Linzen et al., 2016; Jumelet and Hupkes, 2018; Wilcox et al., 2018; Futrell and Levy, 2019, *inter alia*). Zhang and Bowman (2018) compare the syntactic knowledge of language models and neural machine translation systems. We widen the range of pretraining tasks and target probing model tasks to gain a more complete picture. We also focus on a stronger contextualizer architecture, ELMo (original), that has produced state-of-the-art results.

Several studies have sought to intrinsically evaluate noncontextual word representations with word similarity tasks, such as analogies (Mikolov et al., 2013). These methods differ from our approach in that they require no extra parameters and directly assess the vectors, while our probing models must be trained. In this regard, our method is similar to QVEC (Tsvetkov et al., 2015).

8 Conclusion

We study the linguistic knowledge and transferability of contextualized word representations with a suite of sixteen diverse probing tasks. The features generated by pretrained contextualizers are sufficient for high performance on a broad set of tasks. For tasks that require specific information not captured by the contextual word representation, we show that learning task-specific contextual features helps to encode the requisite knowledge. In addition, our analysis of patterns in the transferability of contextualizer layers shows that the lowest layer of LSTMs encodes the most transferable features, while transformers' middle layers are most transferable. We find that higher layers in LSTMs are more task-specific (and thus less general), while transformer layers do not exhibit this same monotonic increase in task-specificity. Prior work has suggested that higher-level contextualizer layers may be expressly encoding higher-level semantic information. Instead, it seems likely that certain high-level semantic phenomena are incidentally useful for the contextualizer's pretraining task, leading to their presence in higher layers. Lastly, we find that bidirectional language model pretraining yields representations that are more transferable *in general* than eleven other candidate pretraining tasks.

Acknowledgments

We thank Johannes Bjerva for sharing the semantic tagging dataset used in Bjerva et al. (2016). We also thank the members of the Noah's ARK group at the University of Washington, the researchers at the Allen Institute for Artificial Intelligence, and the anonymous reviewers for their valuable feedback. NL is supported by a Washington Research Foundation Fellowship and a Barry M. Goldwater Scholarship. YB is supported by the Harvard Mind, Brain, and Behavior Initiative.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proc. of EACL*.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proc. of ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Yonatan Belinkov. 2018. *On Internal Language Representations in Deep Learning: An Analysis of Machine Translation and Speech Recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. 2017a. What do neural machine translation models learn about morphology? In *Proc. of ACL*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proc. of IJCNLP*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proc. of COLING*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proc. of ACL*.

Bernd Bohnet, Ryan T. McDonald, Gonalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proc. of ACL*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks. In *Proc. of ICLR*.

Samuel R. Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, and Berlin Chen. 2018. Looking for ELMo's friends: Sentence-level pretraining beyond language modeling. ArXiv:1812.10860.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Proc. of INTERSPEECH*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proc. of ACL*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proc. of EMNLP*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proc. of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proc. of ACL*.

Jessica Ficler and Yoav Goldberg. 2016. Coordination annotation extension in the Penn Treebank. In *Proc. of ACL*.

Richard Futrell and Roger P. Levy. 2019. Do RNNs learn human-like abstract word order preferences? In *Proc. of SCiL*.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *Proc. of NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proc. of NLP-OSS*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proc. of EMNLP*.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of ACL*.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. In *Proc. of IJCAI*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proc. of EMNLP*.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? On the ability of LSTMs to understand negative polarity items. In *Proc. of BlackboxNLP*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *Proc. of ICLR (Workshop)*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proc. of ACL*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proc. of NAACL*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. In *Proc. of NAACL*.

Tal Linzen. 2018. What can linguistics and deep learning contribute to each other? *Language*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38:301–333.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. of NeurIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NeurIPS*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval 2015*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? Adapting pretrained representations to diverse tasks. ArXiv:1903.05987.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proc. of NAACL*.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proc. of EMNLP*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proc. of CoNLL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Marek Rei and Anders Sogaard. 2019. Jointly learning to label sentences and tokens. In *Proc. of AAAI*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proc. of ACL*.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proc. of NAACL*.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43:227–268.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? Assessing the factuality degree of events in text. *Computational Linguistics*, 38:261–299.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proc. of ACL*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proc. of NAACL*.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016a. Why neural translations are the right length. In *Proc. of EMNLP*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016b. Does string-based neural MT learn source syntax? In *Proc. of EMNLP*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proc. of LREC*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proc. of EMNLP*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *Proc. of ICLR*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of LLL and CoNLL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler-gap dependencies? In *Proc. of BlackboxNLP*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proc. of EMNLP*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proc. of ACL*.

Michihiro Yasunaga, Jungo Kasai, and Dragomir R. Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proc. of NAACL*.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. In *Proc. of BlackboxNLP*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proc. of ICCV*.

Appendices

A Probing Task Setup Details

Syntactic Constituency Ancestor Tagging We remove the top-level ROOT node in each sentence. For words that do not have a parent, grandparent, or great-grandparent, we set the label to ”None”. The example is then treated as any other, and the probing model is required to predict this ”None” label during training and evaluation.

Preposition Supersense Disambiguation Since we focus on the linguistic knowledge within individual or pairs of CWRs, we train and evaluate our probing models on only single-word adpositions.

Conjunct Identification Our probing models are only trained and evaluated on sentences with a coordination construction in them.

B Probing Model Training Details

Our probing models are trained with Adam (Kingma and Ba, 2015), using a learning rate of 0.001. We train for 50 epochs, using early stopping with a patience of 3. Our models are implemented in the AllenNLP framework (Gardner et al., 2018).

For contextualizers that use subword representations (e.g., the OpenAI transformer and BERT), we aggregate subword representations into token representations by taking a token's representation to be the representation of its final subword.

C References to State-of-the-Art Task-Specific Models (Without Pretraining)

Task	Previous state of the art (without pretraining)
CCG	94.7 (Lewis et al., 2016)
POS (PTB)	97.96 (Bohnet et al., 2018)
POS (EWT)	95.82 (Yasunaga et al., 2018)
Chunk	95.77 (Hashimoto et al., 2017)
NER	91.38 (Hashimoto et al., 2017)
ST	95.15 (Bjerva et al., 2016)
GED	39.83 (Rei and Sogaard, 2019)
PS-Role	66.89 (Schneider et al., 2018)
PS-Fxn	78.29 (Schneider et al., 2018)
EG	77.10 (Rudinger et al., 2018)

Table 4: Performance of prior state of the art models (without pretraining) for each task.

Note that the performance reported in this paper for the preposition supersense identification models of Schneider et al. (2018) differs from their published result. Their published result is the accuracy on all adpositions; since we only train and evaluate our model on single-word adpositions, the number we report in this paper is the performance of the Schneider et al. (2018) model on only single-word adpositions.

D Performance of Pretrained Contextualizers on All Tasks

D.1 Token Labeling (ELMo and OpenAI Transformer)

Pretrained Representation		POS						Supersense ID		
	CCG	PTB	EWT	Parent	GParent	GGParent	ST	PS-Role	PS-Fxn	EF
ELMo Original, Layer 0	73.43	93.31	89.71	85.23	54.58	41.57	83.99	41.45	52.41	52.49
ELMo Original, Layer 1	93.31	97.26	95.61	95.56	81.61	67.50	93.82	74.12	84.87	73.20
ELMo Original, Layer 2	91.23	96.45	94.52	94.35	76.22	62.32	92.41	75.44	83.11	72.11
ELMo Original, Scalar Mix	92.96	97.19	95.09	95.56	81.56	67.42	93.86	74.56	84.65	72.96
ELMo (4-layer), Layer 0	73.41	93.42	89.30	85.45	55.40	42.22	83.95	40.13	55.26	53.58
ELMo (4-layer), Layer 1	93.81	97.31	95.60	95.70	81.57	67.66	94.18	74.78	85.96	73.03
ELMo (4-layer), Layer 2	92.47	97.09	95.08	95.01	77.08	63.04	93.43	74.12	85.53	70.97
ELMo (4-layer), Layer 3	91.56	96.82	94.56	94.65	75.58	61.04	92.82	74.12	83.55	70.66
ELMo (4-layer), Layer 4	90.67	96.44	93.99	94.24	75.70	61.45	91.90	73.46	83.77	72.59
ELMo (4-layer), Scalar Mix	93.23	97.34	95.14	95.55	81.36	67.47	94.05	76.10	84.65	72.70
ELMo (transformer), Layer 0	73.06	93.27	89.42	85.59	55.03	41.38	83.81	41.45	54.39	53.13
ELMo (transformer), Layer 1	91.66	97.09	94.78	94.43	77.28	62.69	93.78	65.13	80.04	67.19
ELMo (transformer), Layer 2	92.68	96.93	95.13	95.15	81.37	67.39	93.71	69.74	80.26	70.88
ELMo (transformer), Layer 3	92.82	96.97	94.74	95.28	82.16	68.06	93.45	70.61	82.24	70.24
ELMo (transformer), Layer 4	91.86	96.71	94.41	94.97	81.48	67.33	92.82	72.81	82.02	69.97
ELMo (transformer), Layer 5	91.06	96.24	93.85	94.30	79.65	64.92	91.92	69.52	79.82	70.21
ELMo (transformer), Layer 6	90.19	96.33	93.62	93.98	77.40	63.49	91.78	65.57	80.48	70.82
ELMo (transformer), Scalar Mix	93.66	97.35	94.59	95.16	83.38	69.29	94.26	72.59	82.46	71.94
OpenAI transformer, Layer 0	71.58	89.54	87.44	84.50	56.24	46.31	81.18	37.72	48.90	55.03
OpenAI transformer, Layer 1	78.08	93.32	89.93	88.75	63.59	53.28	85.73	43.64	61.40	63.13
OpenAI transformer, Layer 2	78.19	92.71	85.27	88.22	65.85	56.34	85.54	52.41	66.45	65.69
OpenAI transformer, Layer 3	79.53	93.43	89.67	88.73	67.34	58.10	86.17	53.51	70.18	68.39
OpenAI transformer, Layer 4	80.95	93.82	91.28	90.07	69.34	60.74	87.34	58.55	71.27	69.82
OpenAI transformer, Layer 5	82.03	93.82	91.11	90.51	71.41	62.69	87.81	60.75	73.46	70.92
OpenAI transformer, Layer 6	82.38	93.45	88.09	90.32	72.10	63.68	87.46	64.04	74.12	72.08
OpenAI transformer, Layer 7	82.61	93.25	86.50	90.71	72.60	63.69	86.49	65.13	76.32	73.87
OpenAI transformer, Layer 8	81.43	92.10	86.66	91.00	72.66	64.01	86.65	66.23	76.97	73.86
OpenAI transformer, Layer 9	81.73	91.99	86.60	90.84	72.34	63.72	86.19	66.01	76.54	74.03
OpenAI transformer, Layer 10	81.73	92.05	86.37	90.74	71.41	62.45	86.22	63.38	75.88	73.30
OpenAI transformer, Layer 11	81.97	91.64	86.62	90.43	70.48	60.84	85.91	63.16	76.97	71.99
OpenAI transformer, Layer 12	82.69	92.18	90.87	90.89	69.14	58.74	87.43	63.60	75.66	71.34
OpenAI transformer, Scalar Mix	83.94	94.63	92.60	92.08	73.11	64.64	88.73	64.69	79.17	74.25
GloVe (840B.300d)	71.58	90.49	83.93	81.77	54.01	41.21	80.92	40.79	51.54	49.70
Previous state of the art	94.7	97.96	96.73	-	-	-	95.15	66.89	78.29	77.10

Table 5: Token labeling task performance of a linear probing model trained on top of the ELMo and OpenAI contextualizers, compared against a GloVe-based probing baseline and the previous state of the art.

D.2 Token Labeling (BERT)

Pretrained Representation		POS						Supersense ID		
	CCG	PTB	EWT	Parent	GParent	GGParent	ST	PS-Role	PS-Fxn	EF
BERT (base, cased), Layer 0	71.45	89.99	86.77	84.41	55.92	46.07	82.25	42.11	54.82	52.70
BERT (base, cased), Layer 1	81.67	93.80	90.58	89.47	62.92	50.93	88.89	50.88	67.76	59.83
BERT (base, cased), Layer 2	88.43	95.76	93.72	92.98	71.73	57.84	92.23	63.60	75.00	64.91
BERT (base, cased), Layer 3	89.77	96.08	94.30	93.92	73.24	58.57	92.85	64.69	78.95	65.58
BERT (base, cased), Layer 4	91.41	96.57	94.58	94.67	76.09	61.17	93.38	66.23	79.17	67.55
BERT (base, cased), Layer 5	92.22	96.68	94.93	95.10	77.79	63.56	93.47	68.20	82.89	69.08
BERT (base, cased), Layer 6	93.14	96.95	95.15	95.46	79.75	65.36	93.72	76.10	84.65	71.26
BERT (base, cased), Layer 7	93.51	96.92	95.12	95.70	80.38	65.96	93.62	77.85	86.40	71.54
BERT (base, cased), Layer 8	93.67	96.80	95.21	95.60	81.04	66.66	93.37	79.61	87.94	73.49
BERT (base, cased), Layer 9	93.51	96.68	94.94	95.64	80.70	66.53	93.18	79.39	86.84	75.11
BERT (base, cased), Layer 10	93.25	96.54	94.51	95.26	79.60	65.49	92.90	79.17	86.18	74.70
BERT (base, cased), Layer 11	92.75	96.40	94.31	95.00	78.50	64.34	92.64	77.41	85.53	75.11
BERT (base, cased), Layer 12	92.21	96.09	93.86	94.55	76.95	62.87	92.34	78.07	84.65	73.77
BERT (base, cased), Scalar Mix	93.78	97.02	95.63	95.83	81.67	67.48	93.85	78.51	85.96	74.88
BERT (large, cased), Layer 0	71.06	89.84	86.81	84.28	55.84	46.17	82.31	38.38	54.61	52.81
BERT (large, cased), Layer 1	79.49	92.58	89.45	88.50	60.96	49.88	87.16	53.51	65.13	59.49
BERT (large, cased), Layer 2	83.30	94.03	91.70	90.48	64.91	51.94	89.47	58.55	71.93	62.49
BERT (large, cased), Layer 3	83.32	94.09	91.92	90.76	64.99	52.26	89.67	58.33	72.81	62.52
BERT (large, cased), Layer 4	88.51	95.61	93.36	93.26	70.99	56.22	92.58	65.35	78.29	65.06
BERT (large, cased), Layer 5	89.69	95.95	94.15	93.94	72.62	57.58	93.05	62.06	76.97	65.79
BERT (large, cased), Layer 6	90.91	96.14	94.35	94.47	75.59	60.80	93.35	62.72	78.51	67.00
BERT (large, cased), Layer 7	91.72	96.30	94.64	94.55	76.35	60.98	93.55	67.98	81.36	66.42
BERT (large, cased), Layer 8	91.56	96.36	94.80	94.61	76.40	61.93	93.50	66.89	80.26	68.56
BERT (large, cased), Layer 9	91.76	96.31	94.86	94.70	75.95	61.60	93.44	66.89	82.02	69.12
BERT (large, cased), Layer 10	91.71	96.27	94.89	94.88	75.84	61.44	93.42	68.64	79.39	69.37
BERT (large, cased), Layer 11	92.01	96.26	94.96	95.10	77.01	62.79	93.39	70.83	81.80	71.12
BERT (large, cased), Layer 12	92.82	96.48	95.27	95.31	78.66	64.51	93.61	74.34	84.21	72.44
BERT (large, cased), Layer 13	93.48	96.73	95.56	95.72	80.51	65.85	93.83	76.54	85.75	72.91
BERT (large, cased), Layer 14	93.85	96.73	95.54	95.98	81.89	67.02	93.81	78.95	87.94	72.72
BERT (large, cased), Layer 15	94.21	96.72	95.80	96.10	82.46	67.53	93.76	79.17	89.25	72.79
BERT (large, cased), Layer 16	94.28	96.67	95.62	96.05	82.78	67.90	93.61	78.73	90.13	74.27
BERT (large, cased), Layer 17	94.13	96.53	95.55	95.92	82.56	67.74	93.45	79.17	87.06	75.52
BERT (large, cased), Layer 18	93.76	96.38	95.45	95.57	81.47	67.11	93.21	79.17	87.06	75.95
BERT (large, cased), Layer 19	93.36	96.25	95.30	95.38	80.47	66.08	93.01	76.10	85.96	76.25
BERT (large, cased), Layer 20	93.06	96.10	94.96	95.20	79.32	64.86	92.78	78.29	87.72	75.92
BERT (large, cased), Layer 21	91.83	95.38	94.05	94.16	76.84	62.43	91.65	74.12	82.89	75.16
BERT (large, cased), Layer 22	89.66	93.88	92.30	92.62	74.73	60.76	89.42	73.90	82.02	74.28
BERT (large, cased), Layer 23	88.70	93.02	91.90	92.36	73.33	59.27	88.92	69.08	80.70	73.54
BERT (large, cased), Layer 24	87.65	92.60	90.84	91.81	71.98	57.95	88.26	69.74	78.73	72.65
BERT (large, cased), Scalar Mix	94.48	97.17	96.05	96.27	83.51	68.90	93.96	78.95	87.06	76.13

Table 6: Token labeling task performance of a linear probing model trained on top of the BERT contextualizers.

D.3 Segmentation (ELMo and OpenAI Transformer)

Pretrained Representation	Chunk	NER	GED	Conj
ELMo Original, Layer 0	70.68	64.39	18.49	15.59
ELMo Original, Layer 1	90.04	82.85	29.37	38.72
ELMo Original, Layer 2	86.47	82.80	26.08	29.08
ELMo Original, Scalar Mix	89.29	82.90	27.54	39.57
ELMo (4-layer), Layer 0	70.57	63.96	8.46	15.15
ELMo (4-layer), Layer 1	89.78	81.04	28.07	36.37
ELMo (4-layer), Layer 2	87.18	80.19	29.24	31.44
ELMo (4-layer), Layer 3	86.20	81.56	28.51	28.57
ELMo (4-layer), Layer 4	85.07	82.06	23.85	26.31
ELMo (4-layer), Scalar Mix	86.67	82.37	30.46	28.42
ELMo (transformer), Layer 0	71.01	64.23	13.25	15.69
ELMo (transformer), Layer 1	91.75	78.51	25.29	26.56
ELMo (transformer), Layer 2	92.18	80.92	28.63	34.99
ELMo (transformer), Layer 3	92.14	80.80	29.16	38.23
ELMo (transformer), Layer 4	91.32	80.47	29.71	38.52
ELMo (transformer), Layer 5	89.18	81.21	30.80	35.49
ELMo (transformer), Layer 6	87.96	79.77	27.20	29.17
ELMo (transformer), Scalar Mix	92.08	81.68	26.56	38.45
OpenAI transformer, Layer 0	66.59	46.29	14.78	16.84
OpenAI transformer, Layer 1	77.87	48.88	19.72	17.59
OpenAI transformer, Layer 2	79.67	52.13	21.59	20.72
OpenAI transformer, Layer 3	80.78	52.40	22.58	22.36
OpenAI transformer, Layer 4	82.95	54.62	25.61	23.04
OpenAI transformer, Layer 5	84.67	56.25	29.69	25.53
OpenAI transformer, Layer 6	85.46	56.46	30.69	27.25
OpenAI transformer, Layer 7	86.06	57.73	33.10	30.68
OpenAI transformer, Layer 8	85.75	56.50	32.17	33.06
OpenAI transformer, Layer 9	85.40	57.31	31.90	32.65
OpenAI transformer, Layer 10	84.52	57.32	32.08	30.27
OpenAI transformer, Layer 11	83.00	56.94	30.22	26.60
OpenAI transformer, Layer 12	82.44	58.14	30.81	25.19
OpenAI transformer, Scalar Mix	87.44	59.39	34.54	31.65
GloVe (840B.300d)	62.28	53.22	14.94	10.53
Previous state of the art	95.77	91.38	34.76	-

Table 7: Segmentation task performance of a linear probing model trained on top of the ELMo and OpenAI contextualizers, compared against a GloVe-based probing baseline and the previous state of the art.

D.4 Segmentation (BERT)

Pretrained Representation	Chunk	NER	GED	Conj
BERT (base, cased), Layer 0	69.86	53.50	12.63	16.24
BERT (base, cased), Layer 1	75.56	66.94	16.85	21.83
BERT (base, cased), Layer 2	86.64	71.08	22.66	22.87
BERT (base, cased), Layer 3	87.70	73.83	25.80	25.50
BERT (base, cased), Layer 4	90.64	77.28	31.35	29.39
BERT (base, cased), Layer 5	91.21	78.81	32.34	30.58
BERT (base, cased), Layer 6	92.29	80.81	37.85	35.26
BERT (base, cased), Layer 7	92.64	81.50	40.14	35.86
BERT (base, cased), Layer 8	92.11	82.45	42.08	42.26
BERT (base, cased), Layer 9	91.95	82.71	43.20	43.93
BERT (base, cased), Layer 10	91.30	82.66	42.46	43.38
BERT (base, cased), Layer 11	90.71	82.42	43.30	41.35
BERT (base, cased), Layer 12	89.38	80.64	39.87	39.34
BERT (base, cased), Scalar Mix	92.96	82.43	43.22	43.15
BERT (large, cased), Layer 0	70.42	53.95	13.44	16.65
BERT (large, cased), Layer 1	73.98	65.92	16.20	19.58
BERT (large, cased), Layer 2	79.82	67.96	17.26	20.01
BERT (large, cased), Layer 3	79.50	68.82	17.42	21.83
BERT (large, cased), Layer 4	87.49	71.13	24.06	23.21
BERT (large, cased), Layer 5	89.81	72.06	30.27	24.13
BERT (large, cased), Layer 6	89.92	74.30	31.44	26.75
BERT (large, cased), Layer 7	90.39	75.93	33.27	27.74
BERT (large, cased), Layer 8	90.28	76.99	33.34	29.94
BERT (large, cased), Layer 9	90.09	78.87	33.16	30.07
BERT (large, cased), Layer 10	89.92	80.08	33.31	30.17
BERT (large, cased), Layer 11	90.20	81.23	34.49	31.78
BERT (large, cased), Layer 12	91.22	83.00	37.27	34.10
BERT (large, cased), Layer 13	93.04	83.66	40.10	35.04
BERT (large, cased), Layer 14	93.64	84.11	43.11	39.67
BERT (large, cased), Layer 15	93.18	84.21	44.92	43.12
BERT (large, cased), Layer 16	93.14	84.34	45.37	46.54
BERT (large, cased), Layer 17	92.80	84.44	45.60	47.76
BERT (large, cased), Layer 18	91.72	84.03	45.82	47.34
BERT (large, cased), Layer 19	91.48	84.29	46.46	46.00
BERT (large, cased), Layer 20	90.78	84.25	46.07	44.81
BERT (large, cased), Layer 21	87.97	82.36	44.53	41.91
BERT (large, cased), Layer 22	85.19	77.58	43.03	37.49
BERT (large, cased), Layer 23	84.23	77.02	42.00	35.21
BERT (large, cased), Layer 24	83.30	74.83	41.29	34.38
BERT (large, cased), Scalar Mix	93.59	84.98	47.32	45.94

Table 8: Segmentation task performance of a linear probing model trained on top of the BERT contextualizers.

D.5 Pairwise Relations (ELMo and OpenAI Transformer)

Pretrained Representation	Syntactic Dep. Arc Prediction		Syntactic Dep. Arc Classification		Semantic Dep. Arc Prediction	Semantic Dep. Arc Classification	Coreference Arc Prediction
	PTB	EWT	PTB	EWT			
ELMo (original), Layer 0	78.27	77.73	82.05	78.52	70.65	77.48	72.89
ELMo (original), Layer 1	89.04	86.46	96.13	93.01	87.71	93.31	71.33
ELMo (original), Layer 2	88.33	85.34	94.72	91.32	86.44	90.22	68.46
ELMo (original), Scalar Mix	89.30	86.56	95.81	91.69	87.79	93.13	73.24
ELMo (4-layer), Layer 0	78.09	77.57	82.13	77.99	69.96	77.22	73.57
ELMo (4-layer), Layer 1	88.79	86.31	96.20	93.20	87.15	93.27	72.93
ELMo (4-layer), Layer 2	87.33	84.75	95.38	91.87	85.29	90.57	71.78
ELMo (4-layer), Layer 3	86.74	84.17	95.06	91.55	84.44	90.04	70.11
ELMo (4-layer), Layer 4	87.61	85.09	94.14	90.68	85.81	89.45	68.36
ELMo (4-layer), Scalar Mix	88.98	85.94	95.82	91.77	87.39	93.25	73.88
ELMo (transformer), Layer 0	78.10	78.04	81.09	77.67	70.11	77.11	72.50
ELMo (transformer), Layer 1	88.24	85.48	93.62	89.18	85.16	90.66	72.47
ELMo (transformer), Layer 2	88.87	84.72	94.14	89.40	85.97	91.29	73.03
ELMo (transformer), Layer 3	89.01	84.62	94.07	89.17	86.83	90.35	72.62
ELMo (transformer), Layer 4	88.55	85.62	94.14	89.00	86.00	89.04	71.80
ELMo (transformer), Layer 5	88.09	83.23	92.70	88.84	85.79	89.66	71.62
ELMo (transformer), Layer 6	87.22	83.28	92.55	87.13	84.71	87.21	66.35
ELMo (transformer), Scalar Mix	90.74	86.39	96.40	91.06	89.18	94.35	75.52
OpenAI transformer, Layer 0	80.80	79.10	83.35	80.32	76.39	80.50	72.58
OpenAI transformer, Layer 1	81.91	79.99	88.22	84.51	77.70	83.88	75.23
OpenAI transformer, Layer 2	82.56	80.22	89.34	85.99	78.47	85.85	75.77
OpenAI transformer, Layer 3	82.87	81.21	90.89	87.67	78.91	87.76	75.81
OpenAI transformer, Layer 4	83.69	82.07	92.21	89.24	80.51	89.59	75.99
OpenAI transformer, Layer 5	84.53	82.77	93.12	90.34	81.95	90.25	76.05
OpenAI transformer, Layer 6	85.47	83.89	93.71	90.63	83.88	90.99	74.43
OpenAI transformer, Layer 7	86.32	84.15	93.95	90.82	85.15	91.18	74.05
OpenAI transformer, Layer 8	86.84	84.06	94.16	91.02	85.23	90.86	74.20
OpenAI transformer, Layer 9	87.00	84.47	93.95	90.77	85.95	90.85	74.57
OpenAI transformer, Layer 10	86.76	84.28	93.40	90.26	85.17	89.94	73.86
OpenAI transformer, Layer 11	85.84	83.42	92.82	89.07	83.39	88.46	72.03
OpenAI transformer, Layer 12	85.06	83.02	92.37	89.08	81.88	87.47	70.44
OpenAI transformer, Scalar Mix	87.18	85.30	94.51	91.55	86.13	91.55	76.47
GloVe (840B.300d)	74.14	73.94	77.54	72.74	68.94	71.84	72.96

Table 9: Pairwise relation task performance of a linear probing model trained on top of the ELMo and OpenAI contextualizers, compared against a GloVe-based probing baseline.

D.6 Pairwise Relations (BERT)

Pretrained Representation	Syntactic Dep. Arc Prediction		Syntactic Dep. Arc Classification		Semantic Dep. Arc Prediction	Semantic Dep. Arc Classification	Coreference Arc Prediction
	PTB	EWT	PTB	EWT			
BERT (base, cased), Layer 0	83.00	80.36	83.47	79.15	80.26	80.35	74.93
BERT (base, cased), Layer 1	83.66	81.69	86.92	82.62	80.81	82.69	75.35
BERT (base, cased), Layer 2	84.00	82.66	91.90	88.51	79.34	87.45	75.19
BERT (base, cased), Layer 3	84.12	82.86	92.80	89.49	79.05	88.41	75.83
BERT (base, cased), Layer 4	85.50	84.07	93.91	91.02	81.37	90.20	76.14
BERT (base, cased), Layer 5	86.67	84.69	94.87	92.01	83.41	91.34	76.35
BERT (base, cased), Layer 6	87.98	85.91	95.57	93.01	85.73	92.47	75.95
BERT (base, cased), Layer 7	88.24	86.30	95.65	93.31	85.96	92.75	75.37
BERT (base, cased), Layer 8	88.64	86.49	95.90	93.39	86.59	93.18	76.39
BERT (base, cased), Layer 9	88.76	86.17	95.84	93.32	86.74	92.68	76.62
BERT (base, cased), Layer 10	88.16	85.86	95.42	92.82	86.29	91.79	76.84
BERT (base, cased), Layer 11	87.74	85.40	95.09	92.37	85.83	91.07	76.88
BERT (base, cased), Layer 12	85.93	83.99	94.79	91.70	82.71	90.10	76.78
BERT (base, cased), Scalar Mix	89.06	86.58	95.91	93.10	87.10	93.38	77.88
BERT (large, cased), Layer 0	82.22	79.92	83.57	79.32	79.04	81.25	73.75
BERT (large, cased), Layer 1	81.65	80.04	85.23	80.95	77.97	81.36	73.99
BERT (large, cased), Layer 2	81.84	80.09	87.39	83.80	77.17	82.44	73.89
BERT (large, cased), Layer 3	81.66	80.35	87.36	83.74	76.92	82.91	73.62
BERT (large, cased), Layer 4	83.56	82.17	91.44	88.45	78.43	87.32	72.99
BERT (large, cased), Layer 5	84.24	82.94	92.33	89.62	79.28	88.85	73.34
BERT (large, cased), Layer 6	85.05	83.50	93.75	91.02	80.18	90.14	74.02
BERT (large, cased), Layer 7	85.43	84.03	94.06	91.65	80.64	90.69	74.55
BERT (large, cased), Layer 8	85.41	83.92	94.18	91.66	80.64	90.82	75.92
BERT (large, cased), Layer 9	85.35	83.76	94.11	91.10	80.64	90.62	76.00
BERT (large, cased), Layer 10	85.51	83.92	94.09	91.17	81.51	90.43	76.19
BERT (large, cased), Layer 11	85.91	83.88	94.48	91.73	82.05	91.13	75.86
BERT (large, cased), Layer 12	86.80	85.13	95.03	92.37	83.99	92.08	75.13
BERT (large, cased), Layer 13	87.64	86.00	95.54	93.02	84.91	92.74	74.63
BERT (large, cased), Layer 14	88.62	86.50	95.94	93.62	85.91	93.51	75.16
BERT (large, cased), Layer 15	88.87	86.95	96.02	93.66	86.49	93.86	75.58
BERT (large, cased), Layer 16	89.36	87.25	96.18	93.86	87.79	93.83	75.15
BERT (large, cased), Layer 17	89.62	87.47	96.01	93.88	88.14	93.41	75.93
BERT (large, cased), Layer 18	89.41	87.00	95.82	93.47	87.77	93.00	77.85
BERT (large, cased), Layer 19	88.78	86.60	95.59	92.98	87.16	92.27	80.47
BERT (large, cased), Layer 20	88.24	85.87	95.12	92.47	86.45	91.33	80.94
BERT (large, cased), Layer 21	86.48	84.21	94.21	91.12	83.94	89.42	81.14
BERT (large, cased), Layer 22	85.42	83.24	92.94	90.02	82.01	88.17	80.36
BERT (large, cased), Layer 23	84.69	82.81	92.28	89.47	81.07	87.32	79.64
BERT (large, cased), Layer 24	83.24	81.48	91.07	87.88	78.24	85.98	79.35
BERT (large, cased), Scalar Mix	90.09	87.51	96.15	93.61	88.49	94.25	81.16

Table 10: Pairwise relation task performance of a linear probing model trained on top of the BERT contextualizers.

E Full Results for Transferring Between Pretraining Tasks

E.1 Token Labeling

Pretrained Representation			Supersense ID		
	POS (EWT)	ST	PS-Role	PS-Fxn	EF
Untrained ELMo (original), Layer 0	77.05	76.09	36.99	48.17	43.08
Untrained ELMo (original), Layer 1	56.03	68.63	16.01	24.71	45.57
Untrained ELMo (original), Layer 2	55.89	68.51	16.01	25.44	46.06
Untrained ELMo (original), Scalar Mix	78.58	82.45	38.23	48.90	47.37
CCG, Layer 0	84.33	79.53	38.38	53.29	47.71
CCG, Layer 1	88.02	87.97	46.27	58.48	57.96
CCG, Layer 2	87.81	87.38	43.79	58.55	57.98
CCG, Scalar Mix	90.44	91.21	50.07	65.57	60.24
Chunk, Layer 0	82.51	78.45	37.06	49.12	38.93
Chunk, Layer 1	87.33	87.42	44.81	59.36	55.66
Chunk, Layer 2	86.61	87.04	39.91	58.11	56.95
Chunk, Scalar Mix	88.62	89.77	44.23	60.01	56.24
PTB (POS), Layer 0	84.58	79.95	37.43	49.49	46.19
PTB (POS), Layer 1	90.53	90.10	42.47	59.80	61.28
PTB (POS), Layer 2	90.45	89.83	44.37	58.92	62.14
PTB (POS), Scalar Mix	90.75	91.13	45.39	60.67	62.77
Parent, Layer 0	81.84	78.47	36.33	49.71	38.35
Parent, Layer 1	87.21	87.36	45.98	58.85	54.45
Parent, Layer 2	86.57	86.18	42.69	58.48	54.58
Parent, Scalar Mix	89.10	90.01	44.88	61.92	55.64
GParent, Layer 0	81.85	78.77	37.06	51.75	40.46
GParent, Layer 1	86.05	86.78	46.86	60.82	55.58
GParent, Layer 2	85.64	86.17	45.25	62.13	55.65
GParent, Scalar Mix	88.08	89.48	48.03	63.38	55.96
GGParent, Layer 0	81.44	77.88	38.74	49.12	42.17
GGParent, Layer 1	83.51	85.23	44.08	57.68	55.77
GGParent, Layer 2	83.17	84.10	39.40	56.29	55.82
GGParent, Scalar Mix	86.18	88.84	44.52	61.62	55.50
Syn. Arc Prediction (PTB), Layer 0	79.97	77.34	36.26	47.15	38.81
Syn. Arc Prediction (PTB), Layer 1	80.67	82.60	40.06	54.61	47.86
Syn. Arc Prediction (PTB), Layer 2	78.83	80.91	34.65	52.12	45.64
Syn. Arc Prediction (PTB), Scalar Mix	85.76	88.13	40.79	54.17	50.91
Syn. Arc Classification (PTB), Layer 0	83.61	79.61	37.21	51.97	42.07
Syn. Arc Classification (PTB), Layer 1	89.28	88.70	47.22	61.11	55.55
Syn. Arc Classification (PTB), Layer 2	88.77	88.12	44.66	58.92	56.16
Syn. Arc Classification (PTB), Scalar Mix	90.18	90.99	48.17	62.21	56.90
Sem. Arc Prediction, Layer 0	78.64	76.95	34.43	49.78	39.64
Sem. Arc Prediction, Layer 1	74.66	74.83	33.92	47.88	36.46
Sem. Arc Prediction, Layer 2	74.06	73.42	30.85	45.39	35.63
Sem. Arc Prediction, Scalar Mix	83.77	85.06	38.45	57.16	48.27
Sem. Arc Classification, Layer 0	83.17	79.17	38.60	51.54	44.79
Sem. Arc Classification, Layer 1	86.45	87.04	44.81	58.19	55.18
Sem. Arc Classification, Layer 2	85.42	85.87	41.45	58.55	52.87
Sem. Arc Classification, Scalar Mix	88.44	90.00	45.03	61.33	56.07
Conj, Layer 0	72.21	73.87	37.43	47.95	36.33
Conj, Layer 1	64.95	68.96	27.70	41.89	42.10
Conj, Layer 2	64.03	67.17	27.56	37.21	40.59
Conj, Scalar Mix	76.96	80.22	36.33	50.66	42.79
BiLM, Layer 0	87.54	90.22	50.88	67.32	59.65
BiLM, Layer 1	86.55	87.19	50.22	67.11	59.32
BiLM, Layer 2	86.49	89.67	49.34	66.01	59.45
BiLM, Scalar Mix	86.76	90.11	50.44	67.32	67.32
ELMo (original), Layer 0	89.71	83.99	41.45	52.41	52.49
ELMo (original), Layer 1	95.61	93.82	74.12	84.87	73.20
ELMo (original), Layer 2	94.52	92.41	75.44	83.11	72.11
ELMo (original), Scalar Mix	95.09	93.86	74.56	84.65	84.65
GloVe (840B.300d)	83.93	80.92	40.79	51.54	49.70

Table 11: Target token labeling task performance of contextualizers pretrained on a variety of different tasks. The probing model used is linear, and the contextualizer architecture is ELMo (original).

E.2 Segmentation

Pretrained Representation	NER	GED
Untrained ELMo (original), Layer 0	24.71	0.00
Untrained ELMo (original), Layer 1	0.00	0.00
Untrained ELMo (original), Layer 2	0.00	0.00
Untrained ELMo (original), Scalar Mix	34.28	1.81
CCG, Layer 0	32.30	8.89
CCG, Layer 1	44.01	22.68
CCG, Layer 2	42.45	25.15
CCG, Scalar Mix	49.07	4.52
Chunk, Layer 0	23.47	5.80
Chunk, Layer 1	45.44	5.46
Chunk, Layer 2	43.59	24.11
Chunk, Scalar Mix	46.83	4.30
PTB (POS), Layer 0	32.64	7.87
PTB (POS), Layer 1	52.03	5.80
PTB (POS), Layer 2	52.04	9.76
PTB (POS), Scalar Mix	53.51	3.19
Parent, Layer 0	25.11	6.66
Parent, Layer 1	42.76	6.22
Parent, Layer 2	42.49	8.33
Parent, Scalar Mix	47.06	3.01
GParent, Layer 0	30.39	4.58
GParent, Layer 1	47.67	6.20
GParent, Layer 2	47.87	10.34
GParent, Scalar Mix	50.06	1.71
GGParent, Layer 0	28.57	2.25
GGParent, Layer 1	46.21	4.32
GGParent, Layer 2	45.34	3.74
GGParent, Scalar Mix	48.19	1.54
Syn. Arc Prediction (PTB), Layer 0	26.77	1.82
Syn. Arc Prediction (PTB), Layer 1	43.93	5.94
Syn. Arc Prediction (PTB), Layer 2	41.83	14.50
Syn. Arc Prediction (PTB), Scalar Mix	46.58	1.47
Syn. Arc Classification (PTB), Layer 0	33.10	3.51
Syn. Arc Classification (PTB), Layer 1	50.76	3.92
Syn. Arc Classification (PTB), Layer 2	49.64	5.77
Syn. Arc Classification (PTB), Scalar Mix	53.00	1.27
Sem. Arc Prediction, Layer 0	24.47	1.05
Sem. Arc Prediction, Layer 1	34.47	10.78
Sem. Arc Prediction, Layer 2	31.30	10.77
Sem. Arc Prediction, Scalar Mix	36.97	0.32
Sem. Arc Classification, Layer 0	34.00	5.08
Sem. Arc Classification, Layer 1	48.07	5.39
Sem. Arc Classification, Layer 2	46.67	6.24
Sem. Arc Classification, Scalar Mix	50.80	1.75
Conj, Layer 0	17.15	3.99
Conj, Layer 1	37.61	0.87
Conj, Layer 2	34.78	2.38
Conj, Scalar Mix	40.97	0.33
BiLM, Layer 0	56.05	3.99
BiLM, Layer 1	57.19	1.22
BiLM, Layer 2	57.05	1.03
BiLM, Scalar Mix	58.50	1.29
ELMo (original), Layer 0	64.39	18.49
ELMo (original), Layer 1	82.85	29.37
ELMo (original), Layer 2	82.80	26.08
ELMo (original), Scalar Mix	82.90	27.54
GloVe (840B.300d)	53.22	14.94

Table 12: Target segmentation task performance of contextualizers pretrained on a variety of different tasks. The probing model used is linear, and the contextualizer architecture is ELMo (original).

E.3 Pairwise Prediction

Pretrained Representation	Syn. Arc Prediction (EWT)	Syn. Arc Classification (EWT)	Coreference Arc Prediction
Untrained ELMo (original), Layer 0	73.75	66.27	66.25
Untrained ELMo (original), Layer 1	68.40	56.73	62.82
Untrained ELMo (original), Layer 2	68.86	56.62	63.15
Untrained ELMo (original), Scalar Mix	72.24	70.62	69.72
CCG, Layer 0	75.92	69.84	67.84
CCG, Layer 1	84.93	85.59	62.10
CCG, Layer 2	84.45	84.59	59.19
CCG, Scalar Mix	85.44	88.11	70.14
Chunk, Layer 0	76.67	69.72	65.60
Chunk, Layer 1	85.18	86.50	62.74
Chunk, Layer 2	84.80	84.84	60.23
Chunk, Scalar Mix	85.42	87.57	68.92
PTB (POS), Layer 0	76.07	70.32	67.50
PTB (POS), Layer 1	83.97	86.64	63.43
PTB (POS), Layer 2	83.88	86.44	61.61
PTB (POS), Scalar Mix	84.17	87.72	69.61
Parent, Layer 0	76.20	68.99	67.80
Parent, Layer 1	84.93	86.15	62.69
Parent, Layer 2	85.57	85.61	59.10
Parent, Scalar Mix	86.01	87.49	69.34
GParent, Layer 0	76.59	69.51	68.99
GParent, Layer 1	85.96	85.33	60.84
GParent, Layer 2	85.69	84.38	58.76
GParent, Scalar Mix	86.17	87.49	70.24
GGParent, Layer 0	76.28	69.91	69.24
GGParent, Layer 1	85.74	83.45	59.73
GGParent, Layer 2	85.49	82.12	58.89
GGParent, Scalar Mix	86.27	86.57	70.58
Syn. Arc Prediction (PTB), Layer 0	77.04	68.01	68.28
Syn. Arc Prediction (PTB), Layer 1	90.39	81.00	60.29
Syn. Arc Prediction (PTB), Layer 2	90.82	76.50	57.46
Syn. Arc Prediction (PTB), Scalar Mix	91.66	84.18	69.15
Syn. Arc Classification (PTB), Layer 0	76.14	71.80	68.40
Syn. Arc Classification (PTB), Layer 1	86.55	90.04	62.10
Syn. Arc Classification (PTB), Layer 2	87.46	89.35	59.74
Syn. Arc Classification (PTB), Scalar Mix	87.78	90.98	70.00
Sem. Arc Prediction, Layer 0	76.25	67.73	69.44
Sem. Arc Prediction, Layer 1	84.91	73.11	57.62
Sem. Arc Prediction, Layer 2	85.86	69.75	55.91
Sem. Arc Prediction, Scalar Mix	86.37	80.74	69.72
Sem. Arc Classification, Layer 0	75.85	70.12	68.96
Sem. Arc Classification, Layer 1	85.30	86.21	60.25
Sem. Arc Classification, Layer 2	86.10	84.50	58.39
Sem. Arc Classification, Scalar Mix	86.53	87.75	70.36
Conj, Layer 0	72.62	58.40	68.50
Conj, Layer 1	80.84	68.12	58.46
Conj, Layer 2	80.46	64.30	57.89
Conj, Scalar Mix	80.96	73.89	71.96
BiLM, Layer 0	84.27	86.74	71.75
BiLM, Layer 1	86.36	86.86	70.47
BiLM, Layer 2	86.44	86.19	70.14
BiLM, Scalar Mix	86.42	85.93	71.62
ELMo (original), Layer 0	77.73	78.52	72.89
ELMo (original), Layer 1	86.46	93.01	71.33
ELMo (original), Layer 2	85.34	91.32	68.46
ELMo (original), Scalar Mix	86.56	91.69	73.24
GloVe (840B.300d)	73.94	72.74	72.96

Table 13: Target pairwise prediction task performance of contextualizers pretrained on a variety of different tasks. The probing model used is linear, and the contextualizer architecture is ELMo (original).

Mutual Information Maximization for Simple and Accurate Part-Of-Speech Induction

Karl Stratos
Toyota Technological Institute at Chicago
`stratos@ttic.edu`

Abstract

We address part-of-speech (POS) induction by maximizing the mutual information between the induced label and its context. We focus on two training objectives that are amenable to stochastic gradient descent (SGD): a novel generalization of the classical Brown clustering objective and a recently proposed variational lower bound. While both objectives are subject to noise in gradient updates, we show through analysis and experiments that the variational lower bound is robust whereas the generalized Brown objective is vulnerable. We obtain strong performance on a multitude of datasets and languages with a simple architecture that encodes morphology and context.

1 Introduction

We consider information theoretic objectives for POS induction, an important unsupervised learning problem in computational linguistics (Christodoulopoulos et al., 2010). The idea is to make the induced label syntactically informative by maximizing its mutual information with respect to local context. Mutual information has long been a workhorse in the development of NLP techniques, for instance the classical Brown clustering algorithm (Brown et al., 1992). But its role in today's deep learning paradigm is less clear and a subject of active investigation (Belghazi et al., 2018; Oord et al., 2018).

We focus on fully differentiable objectives that can be plugged into an automatic differentiation system and efficiently optimized by SGD. Specifically, we investigate two training objectives. The first is a novel generalization of the Brown clustering objective obtained by relaxing the hard clustering constraint. The second is a recently proposed variational lower bound on mutual information (McAllester, 2018).

A main challenge in optimizing these objectives is the difficulty of stochastic optimization. Each objective involves entropy estimation which is a nonlinear function of all data and does not decompose over individual instances. This makes the gradients estimated on minibatches inconsistent with the true gradient estimated from the entire dataset. To our surprise, in practice we are able to optimize the variational objective effectively but not the generalized Brown objective. We analyze the estimated gradients and show that the inconsistency error is only logarithmic in the former but linear in the latter.

We validate our approach on POS induction by attaining strong performance on a multitude of datasets and languages. Our simple architecture that encodes morphology and context reaches up to 80.1 many-to-one accuracy on the 45-tag Penn WSJ dataset and achieves 4.7% absolute improvement to the previous best result on the universal treebank. Unlike previous works, our model does not rely on computationally expensive structured inference or hand-crafted features.

2 Background

2.1 Information Theory

Mutual information. Mutual information between two random variables measures the amount of information gained about one variable by observing the other. Unlike the Pearson correlation coefficient which only captures the degree of linear relationship, mutual information captures any nonlinear statistical dependencies (Kinney and Atwal, 2014).

Formally, the mutual information between discrete random variables X, Y with a joint distribution p is the KL divergence between the joint distribution $p(x, y)$ and the product distribution

Proceedings of NAACL-HLT 2019, pages 1095–1104
Minneapolis, Minnesota, June 2 - June 7, 2019.

$p(x)p(y)$ over X, Y:

$$I(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$
$$= \mathop{\mathbf{E}}_{(x,y)\sim p} \left[\log \frac{p(x,y)}{p(x)p(y)} \right] \quad (1)$$

It is thus nonnegative and zero iff X and Y are independent. We assume that the marginals $p(x)$ and $p(y)$ are nonzero.

It is insightful to write mutual information in terms of entropy. The entropy of X is

$$H(X) = -\sum_x p(x) \log p(x) = \mathop{\mathbf{E}}_{x\sim p} \left[\log \frac{1}{p(x)} \right]$$

corresponding to the number of bits for encoding the behavior of X under p.[1] The entropy of X given the information that Y equals y is

$$H(X|Y=y) = -\sum_x p(x|y) \log p(x|y)$$

Taking expectation over Y yields the conditional entropy of X given Y:

$$H(X|Y) = \sum_y p(y) \left(-\sum_x p(x|y) \log p(x|y) \right)$$
$$= \mathop{\mathbf{E}}_{(x,y)\sim p} \left[\log \frac{1}{p(x|y)} \right] \quad (2)$$

By manipulating the terms in mutual information, we can write

$$I(X,Y) = \mathop{\mathbf{E}}_{x\sim p} \left[\log \frac{1}{p(x)} \right] - \mathop{\mathbf{E}}_{(x,y)\sim p} \left[\log \frac{1}{p(x|y)} \right]$$
$$= H(X) - H(X|Y)$$

which expresses the amount of information on X gained by observing Y. Switching X and Y shows that $I(X,Y) = H(Y) - H(Y|X)$.

Cross entropy. If p and q are full-support distributions over the same discrete set, the cross entropy between p and q is the asymmetric quantity

$$H(p,q) = -\sum_x p(x) \log q(x) = \mathop{\mathbf{E}}_{x\sim p} \left[\log \frac{1}{q(x)} \right]$$

corresponding to the number of bits for encoding the behavior of X under p by using q. It is revealing to write entropy in terms of KL divergence. Multiplying the term inside the log by $p(x)/p(x)$ we derive

$$H(p,q) = \mathop{\mathbf{E}}_{x\sim p} \left[\log \frac{1}{p(x)} \right] + \mathop{\mathbf{E}}_{x\sim p} \left[\log \frac{p(x)}{q(x)} \right]$$
$$= H(p) + D_{\mathrm{KL}}(p||q)$$

Thus $H(p,q) \geq H(p)$ with equality iff $p = q$.

2.2 Brown Clustering

Our primary inspiration comes from Brown clustering (Brown et al., 1992), a celebrated word clustering technique that had been greatly influential in unsupervised and semi-supervised NLP long before continuous representations based on neural networks were popularized. It finds a clustering $C : V \to [m]$ of the vocabulary V into m classes by optimizing the mutual information between the clusters of a random bigram (X, Y). Given a corpus of N words $(x_1 \ldots x_N)$, it assumes a uniform distribution over consecutive word pairs (x_{i-1}, x_i) and optimizes the following empirical objective

$$\max_{C:V\to[m]} \sum_{c,c'\in[m]} \frac{\#(c,c')}{N} \log \left(\frac{\#(c,c')N}{\#(c)\#(c')} \right) \quad (3)$$

where $\#(c, c')$ denotes the number of occurrences of the cluster pair (c, c') under C. While this optimization is intractable, Brown et al. (1992) derive an effective heuristic that 1. initializes m most frequent words as singleton clusters and 2. repeatedly merges a pair of clusters that yields the smallest decrease in mutual information. The resulting clusters have been useful in many applications (Koo et al., 2008; Owoputi et al., 2013) and has remained a strong baseline for POS induction decades later (Christodoulopoulos et al., 2010). But the approach is tied to highly nontrivial combinatorial optimization tailored for the specific problem and difficult to scale/generalize.

3 Objectives

In the remainder of the paper, we assume discrete random variables $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ with a joint distribution D that represent naturally co-occurring observations. In POS induction experiments, we will set D to be a context-word distribution where Y is a random word and X is the surrounding context of Y (thus $\mathcal{Y}$ is the vocabulary and $\mathcal{X}$ is the space of all possible contexts). Let m be the number of labels to induce. We introduce a pair of trainable classifiers that define conditional label distributions $p(z|x)$ and $q(z|y)$ for all $x \in \mathcal{X}$,

---

[1] The paper will always assume log base 2 to accommodate the bit interpretation.

$y \in \mathcal{Y}$, and $z \in [m]$. For instance, $p(\cdot|y)$ can be the output of a softmax layer on some transformation of y.

Our goal is to learn these classifiers without observing the latent variable z by optimizing an appropriate objective. For training data, we assume N iid samples $(x_1, y_1) \dots (x_N, y_N) \sim D$.

3.1 Generalized Brown Objective

Our first attempt is to maximize the mutual information between the predictions of p and q. Intuitively, this encourages p and q to agree on some annotation scheme (up to a permutation of labels), modeling the dynamics of inter-annotator agreement (Artstein, 2017). It can be seen as a differentiable generalization of the Brown clustering objective. To this end, define

$$\begin{aligned} p(z) &= \mathop{\mathbf{E}}_{x \sim D}\left[p(z|x)\right] & \forall z \in [m] \\ q(z) &= \mathop{\mathbf{E}}_{y \sim D}\left[q(z|y)\right] & \forall z \in [m] \end{aligned}$$

The mutual information between the predictions of p and q on a single sample (x, y) is then

$$J^{\text{mi}}_{x,y} = \sum_{z,z'} p(z|x) q(z'|y) \log \frac{p(z|x) q(z'|y)}{p(z) q(z')}$$

and the objective (to maximize) is

$$J^{\text{mi}} = \mathop{\mathbf{E}}_{(x,y) \sim D}\left[J^{\text{mi}}_{x,y}\right]$$

Note that this becomes exactly the original Brown objective (3) if (X, Y) is defined as a random bigram and p and q are tied and constrained to be a hard clustering.

Empirical objective. In practice, we work with the value of empirical mutual information $\widehat{J}^{\text{mi}}$ estimated from the training data:

$$\begin{aligned} \hat{p}(z) &= \frac{1}{N} \sum_{i=1}^{N} p(z|x_i) & \forall z \in [m] \\ \hat{q}(z) &= \frac{1}{N} \sum_{i=1}^{N} q(z|y_i) & \forall z \in [m] \\ \widehat{J}^{\text{mi}}_i &= \sum_{z,z'} p(z|x_i) q(z'|y_i) \log \frac{p(z|x_i) q(z'|y_i)}{\hat{p}(z) \hat{q}(z')} \\ \widehat{J}^{\text{mi}} &= \frac{1}{N} \sum_{i=1}^{N} \widehat{J}^{\text{mi}}_i & (4) \end{aligned}$$

Our task is to maximize (4) by taking gradient steps at random minibatches. Note, however, that the objective cannot be written as a sum of local objectives because we take log of the estimates $\hat{q}(z)$ and $\hat{p}(z)$ computed from all samples. This makes the stochastic gradient estimator biased (i.e., it does not match the gradient of (4) in expectation) and compromises the correctness of SGD. This bias is investigated more closely in Section 4.

3.2 Variational Lower Bound

The second training objective we consider can be derived in a rather heuristic but helpful manner as follows. Since X, Y are always drawn together, if $q(z|y)$ is the target label distribution for the pair (x, y), then we can train $p(z|x)$ by minimizing the cross entropy between q and p over samples

$$H(q, p) = \mathop{\mathbf{E}}_{(x,y) \sim D}\left[-\sum_{z} q(z|y) \log p(z|x)\right]$$

which is minimized to zero at $p = q$. However, q is also untrained and needs to be trained along with p. Thus this loss alone admits trivial solutions such as setting $p(1|x) = p(1|y) = 1$ for all (x, y). This undesirable behavior can be prevented by simultaneously maximizing the entropy of q. Let Z denote a random label from q with distribution $q(z) = \mathop{\mathbf{E}}_{y \sim D}\left[q(z|y)\right]$ (thus Z is a function of q). The entropy of Z is

$$H(Z) = -\sum_{z} q(z) \log q(z)$$

Putting together, the objective (to maximize) is

$$J^{\text{var}} = H(Z) - H(q, p)$$

Variational interpretation. The reason this objective is named a variational lower bound is due to McAllester (2018) who shows the following. Consider the mutual information between Z and the raw signal X:

$$I(X, Z) = H(Z) - H(Z|X) \qquad (5)$$

Because Z is drawn from q conditioning on Y, which is co-distributed with X, we have a Markov chain $X \to Y \xrightarrow{q} Z$. Thus maximizing $I(X, Z)$ over the choice of q is a reasonable objective that enforces "predictive coding": the label predicted by q from Y must be as informative of X as possible. It can be seen as a special case of the objective underlying the information bottleneck method (Tishby et al., 2000).

So what is the problem with optimizing (5) directly? The problem is that the conditional entropy under the model

$$H(Z|X) = \mathop{\mathbf{E}}_{\substack{(x,y)\sim D \\ z\sim q(\cdot|y)}} \left[\log \frac{1}{\pi(z|x)} \right] \tag{6}$$

involves the posterior probability of z given x

$$\pi\left(z|x\right) = \frac{\sum_y D(x,y)q(z|y)}{\sum_{y,z} D(x,y)q(z|y)}$$

This *conditional* marginalization is generally intractable and cannot be approximated by sampling since the chance of seeing a particular x is small. However, we can introduce a variational distribution $p(z|x)$ to model $\pi\left(z|x\right)$. Plugging this into (6) we observe that

$$\mathop{\mathbf{E}}_{\substack{(x,y)\sim D \\ z\sim q(\cdot|y)}} \left[\log \frac{1}{p(z|x)} \right] = H(q,p)$$

Moreover,

$$\begin{aligned} &\mathop{\mathbf{E}}_{\substack{(x,y)\sim D \\ z\sim q(\cdot|y)}} \left[\log \frac{1}{p(z|x)} \right] \\ &\quad = \mathop{\mathbf{E}}_{\substack{(x,y)\sim D \\ z\sim q(\cdot|y)}} \left[\log \frac{\pi\left(z|x\right)}{\pi\left(z|x\right)p(z|x)} \right] \\ &\quad = H(Z|X) + D_{\mathrm{KL}}(\pi||p) \end{aligned}$$

Thus $H(q,p)$ is an upper bound on $H(Z|X)$ for any p with equality iff p matches the true posterior distribution π. This in turn means that $J^{\mathrm{var}} = H(Z) - H(q,p)$ is a lower bound on $I(X,Z)$, hence the name.

Empirical objective. As with the generalized Brown objective, the variational lower bound can be estimated from the training data as

$$\begin{aligned} \widehat{H}(Z) &= -\sum_z \hat{q}(z)\log\hat{q}(z) \\ \widehat{H}(q,p) &= \frac{1}{N}\sum_{i=1}^{N}\left(-\sum_z q(z|y_i)\log p(z|x_i)\right) \\ \widehat{J}^{\mathrm{var}} &= \widehat{H}(Z) - \widehat{H}(q,p) \end{aligned} \tag{7}$$

where $\hat{q}(z)$ is defined as in Section 3.1. Our task is again to maximize this empirical objective (7) by taking gradient steps at random minibatches. Once again, it cannot be written as a sum of local objectives because the entropy term involves log of $\hat{q}(z)$ computed from all samples. Thus it is not clear if stochastic optimization will be effective.

4 Analysis

The discussion of the two objectives in the previous section is incomplete because the stochastic gradient estimator is biased under both objectives. In this section, we formalize this issue and analyze the bias.

4.1 Setting

Let $B_1 \ldots B_K$ be a partition of the N training examples $(x_1, y_1) \ldots (x_N, y_N)$ into K (iid) minibatches. For simplicity, assume $|B_k| = M$ for all k and $N = MK$. We will adopt the same notation in Section 3 for $\hat{p}(z)$ and $\hat{q}(z)$ estimated from all N samples. Define analogous estimates based on the k-th minibatch by

$$\begin{aligned} \hat{p}_k(z) &= \frac{1}{M}\sum_{x\in B_k} p(z|x) \qquad \forall z \in [m] \\ \hat{q}_k(z) &= \frac{1}{M}\sum_{y\in B_k} q(z|y) \qquad \forall z \in [m] \end{aligned}$$

If l_N denotes an objective function computed from all N samples in the training data and l_k denotes the same objective computed from B_k, a condition on the correctness of SGD is that the gradient of l_k (with respect to model parameters) is consistent with the gradient of l_N on average:

$$\nabla l_N = \frac{1}{K}\sum_{k=1}^{K}\nabla l_k + \epsilon \tag{8}$$

where ϵ denotes the bias of the stochastic gradient estimator. In particular, any loss of the form $l_N = (1/K)\sum_k l_k$ that decomposes over independent minibatches (e.g., the cross-entropy loss for supervised classification) satisfies (8) with $\epsilon = 0$. The bias is nonzero for the unsupervised objectives considered in this work due to the issues discussed in Section 3.

4.2 Result

The following theorem precisely quantifies the bias for the empirical losses associated with the variational bound and the generalized Brown objectives. We only show the result with the gradient with respect to q, but the result with the gradient with respect to p is analogous and omitted for brevity.

Theorem 4.1. *Assume the setting in Section 4.1 and the gradient is taken with respect to the pa-*

rameters of q*. For* $l_N = -\widehat{J}^{\text{var}}$ *defined in* (7)*,*

$$\epsilon = \frac{1}{K}\sum_{k=1}^{K}\sum_{z}\log\frac{\hat{q}(z)}{\hat{q}_k(z)}\nabla\hat{q}_k(z)$$

On the other hand, for $l_N = -\widehat{J}^{\text{mi}}$ *defined in* (4)*,*

$$\epsilon = \frac{1}{N}\sum_{k=1}^{K}\sum_{z,z'}\Bigg(\epsilon_k(z,z')\nabla\hat{q}_k(z') + \log\frac{\hat{p}(z)\hat{q}(z')}{\hat{p}_k(z)\hat{q}_k(z')}\sum_{(x,y)\in B_k}p(z|x)\nabla q(z'|y)\Bigg)$$

where

$$\epsilon_k(z,z') = \frac{1}{K}\sum_{i=1}^{N}\frac{p(z|x_i)q(z'|y_i)}{\hat{q}(z')} - \sum_{(x,y)\in B_k}\frac{p(z|x)q(z'|y)}{\hat{q}_k(z')}$$

A proof can be found in the appendix. We see that both biases go to zero as $\hat{p}_k$ and $\hat{q}_k$ approach $\hat{p}$ and $\hat{q}$. However, the bias is logarithmic in the ratio $\hat{q}(z)/\hat{q}_k(z)$ for the variational lower bound but roughly linear in the difference between $\frac{1}{\hat{q}(z')}$ and $\frac{1}{\hat{q}_k(z')}$ for the generalized Brown objective. In this sense, the variational lower bound is exponentially more robust to noise in minibatch estimates than the generalized Brown objective. This is confirmed in experiments: we are able to optimize $\widehat{J}^{\text{var}}$ with minibatches as small as 80 examples but unable to optimize $\widehat{J}^{\text{mi}}$ unless minibatches are prohibitively large.

5 Experiments

We demonstrate the effectiveness of our training objectives on the task of POS induction. The goal of this task is to induce the correct POS tag for a given word in context (Merialdo, 1994). As typical in unsupervised tasks, evaluating the quality of induced labels is challenging; see Christodoulopoulos et al. (2010) for an in-depth discussion. To avoid complications, we follow a standard practice (Berg-Kirkpatrick et al., 2010; Ammar et al., 2014; Lin et al., 2015; Stratos et al., 2016) and adopt the following setting for all compared methods.

- We use many-to-one accuracy as a primary evaluation metric. That is, we map each induced label to the most frequently coinciding ground-truth POS tag in the annotated data and report the resulting accuracy. We also use the V-measure (Rosenberg and Hirschberg, 2007) when comparing with CRF autoencoders to be consistent with reported results (Ammar et al., 2014; Lin et al., 2015).
- We use the number of ground-truth POS tags as the value of m (i.e., number of labels to induce). This is a data-dependent quantity, for instance 45 in the Penn WSJ and 12 in the universal treebank. Fixing the number of tags this way obviates many evaluation issues.
- Model-specific hyperparameters are tuned on the English Penn WSJ dataset. This configuration is then fixed and used for all other datasets: 10 languages in the universal treebank[2] and 7 languages from CoNLL-X and CoNLL 2007.

5.1 Setting

We set D to be a uniform distribution over context-word pairs in the training corpus. Given N samples $(x_1, y_1) \ldots (x_N, y_N) \sim D$, we optimize the variational objective (7) or the generalized Brown objective (4) by taking gradient steps at random minibatches. This gives us conditional label distributions $p(z|x)$ and $q(z|y)$ for all contexts x, words y, and labels z. At test time, we use

$$z^* = \arg\max_z q(z|y)$$

as the induced label of word y. We experimented with different inference methods such as taking $\arg\max_z p(z|x)q(z|y)$ but did not find it helpful.

5.2 Definition of (X, Y)

Let V denote the vocabulary. We assume an integer $H \geq 1$ that specifies the width of local context. Given random word $y \in V$, we set $x \in V^{2H}$ to be an ordered list of H left and H right words of y. For example, with $H = 2$, a typical context-target pair $(x, y) \sim D$ may look like

$$x = (\text{“had”}, \text{“these”}, \text{“in”}, \text{“my”})$$
$$y = \text{“keys”}$$

We find this simple fixed-window definition of observed variables to be the best inductive bias for POS induction. The correct label can be inferred from either x or y in many cases: in the above example, we can infer that the correct POS tag is plural noun by looking at the target or the context.

---

[2] https://github.com/ryanmcd/uni-dep-tb

5.3 Architecture

We use the following simple architecture to parameterize the label distribution $p(\cdot|x)$ conditioned on context $x \in V^H$ and the label distribution $q(\cdot|y)$ conditioned on word $y \in V$.

Context architecture. The parameters of p are word embeddings $e_w \in \mathbb{R}^d$ for all $w \in V$ and matrices $W_j \in \mathbb{R}^{m\times d}$ for all $j = 1 \ldots 2H$. Given $2H$ ordered contextual words $x = (w_j)_{j=1}^{2H}$, we define

$$p\left(\cdot|x\right) = \mathrm{softmax}\left(\sum_{j=1}^{2H} W_j e_{w_j}\right)$$

Word architecture. The parameters of q are the same word embeddings $e_w \in \mathbb{R}^d$ shared with p, character embeddings $e_c \in \mathbb{R}^{d/2}$ for all distinct characters c, two single-layer LSTMs with input/output dimension $d/2$, and matrices $W_c, W_w \in \mathbb{R}^{m\times d}$. Given the word y with character sequence $c_1 \ldots c_T$, we define

$$\begin{aligned}(f_1 \ldots f_T) &= \mathrm{LSTM}_f(e_{c_1} \ldots e_{c_T})\\ (b_1 \ldots b_T) &= \mathrm{LSTM}_b(e_{c_T} \ldots e_{c_1})\\ q(\cdot|y) &= \mathrm{softmax}\left(W_c \begin{bmatrix} f_T \\ b_T \end{bmatrix} + W_w e_y\right)\end{aligned}$$

The overall architecture is illustrated in Figure 1. Our hyperparameters are the embedding dimension $d = 200$, the context width $H = 2$, the learning rate of the Adam optimizer $r = 0.001$, and the minibatch size $B = 80$.[3] Their values are tuned on the 45-tag Penn WSJ dataset to maximize accuracy.

5.4 Baselines

We focus on comparing with the following models which are some of the strongest baselines in the literature we are aware of. Berg-Kirkpatrick et al. (2010) extend a standard hidden Markov Model (HMM) to incorporate linguistic features. Stratos et al. (2016) develop a factorization-based algorithm for learning a constrained HMM. Ammar et al. (2014) propose a CRF autoencoder that reconstructs words from a structured label sequence. Lin et al. (2015) extend Ammar et al. (2014) by switching a categorical reconstruction distribution with a Gaussian distribution. In addition to these baselines, we also report results with Brown clustering (Brown et al., 1992), the Baum-Welch algorithm (Baum and Petrie, 1966), and k-means clustering of 300-dimensional GloVe vectors (Pennington et al., 2014).

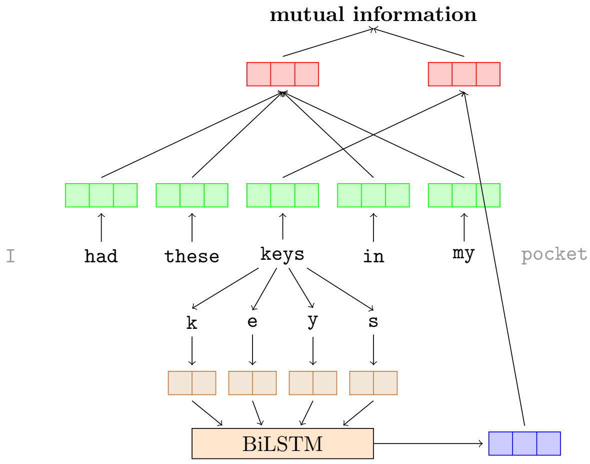


Figure 1: Architecture illustrated on the example text "had these keys in my" with target Y = "keys".

5.5 Results

The 45-tag Penn WSJ dataset. The 45-tag Penn WSJ dataset is a corpus of around one million words each tagged with one of $m = 45$ tags. It is used to optimize hyperparameter values for all compared methods. Table 1 shows the average accuracy over 10 random restarts with the best hyperparameter configurations; standard deviation is given in parentheses (except for deterministic methods Stratos et al. (2016) and Brown clustering).

Our model trained with the variational objective (7) outperforms all baselines.[4] We also observe that our model trained with the generalized Brown objective (4) does not work. We have found that unless the minibatch size is as large as 10,000 the gradient steps do not effectively increase the true data-wide mutual information (4). This supports our bias analysis in Section 4. While it may be possible to develop techniques to resolve the difficulty, for instance keeping a moving average of estimates to stabilize estimation, we leave this as future work and focus on the variational objective in the remainder of the paper.

Table 2 shows ablation experiments on our best

---

[3] An implementation is available at: `https://github.com/karlstratos/mmi-tagger`.

[4] We remark that Tran et al. (2016) report a single number 79.1 with a neuralized HMM. We also note that the concurrent work by He et al. (2018) obtains 80.8 by using word embeddings carefully pretrained on one billion words.

Method	Accuracy
Variational $\widehat{J}^{\text{var}}$ (7)	**78.1** ($\pm$0.8)
Generalized Brown $\widehat{J}^{\text{mi}}$ (4)	48.8 ($\pm$0.9)
Berg-Kirkpatrick et al. (2010)	74.9 ($\pm$1.5)
Stratos et al. (2016)	67.7
Brown et al. (1992)	65.6
Baum-Welch	62.6 ($\pm$1.1)
k-MEANS	32.6 ($\pm$0.7)

Table 1: Many-to-one accuracy on the 45-tag Penn WSJ with the best hyperparameter configurations. The average accuracy over 10 random restarts is reported and the standard deviation is given in parentheses (except for deterministic methods).

Configuration	Accuracy
Best	80.1
$H = 3$	75.9
$H = 1$	75.9
Sentence-level batching	72.4
GloVe initialization	67.6
No character encoding	65.6

Table 2: Ablation of the best model on Penn WSJ.

model (accuracy 80.1) to better understand the sources of its strong performance. Context size $H = 2$ is a sizable improvement over $H = 3$ or $H = 1$. Random sampling is significantly more effective than sentence-level batching (i.e., each minibatch is the set of context-word pairs within a single sentence as done in McAllester (2018)). GloVe initialization of word embeddings e_w is harmful. As expected for POS tagging, morphological modeling with LSTMs gives the largest improvement.

While it may be surprising that GloVe initialization is harmful, it is well known that pretrained word embeddings do not necessarily capture syntactic relationships (as evident in the poor performance of k-means clustering). Consider the top ten nearest neighbors of the word "made" under GloVe embeddings (840B.300d, within PTB vocab) shown in Table 3. The neighbors are clearly not in the same syntactic category. The embeddings can be made more syntactic by controlling the context window. But we found it much more effective (and simpler) to start from randomly initialized embeddings and let the objective induce appropriate representations.

Cosine Similarity	Nearest Neighbor
0.7426	making
0.7113	make
0.6851	that
0.6613	they
0.6584	been
0.6574	would
0.6533	brought
0.6521	had
0.6514	came
0.6494	but
0.6486	even

Table 3: Nearest neighbors of "made" under GloVe embeddings (840B.300d, within PTB vocab).

The 12-tag universal treebank. The universal treebank v2.0 is a corpus in ten languages tagged with $m = 12$ universal POS tags (McDonald et al., 2013). We use this corpus to be compatible with existing results. Table 4 shows results on the dataset, using the same setting in the experiments on the Penn WSJ dataset. Our model significantly outperforms the previous state-of-the-art, achieving an absolute gain of 4.7 over Stratos et al. (2016) in average accuracy.

Comparison with CRF autoencoders. Table 5 shows a direct comparison with CRF autoencoders (Ammar et al., 2014; Lin et al., 2015) in many-to-one accuracy and the V-measure. We compare against their reported numbers by running our model once on the same datasets using the same setting in the experiments on the Penn WSJ dataset. The data consists of the training portion of CoNLL-X and CoNLL 2007 labeled with 12 universal tags. Our model is competitive with all baselines.

6 Related Work

Information theory, in particular mutual information, has played a prominent role in NLP (Church and Hanks, 1990; Brown et al., 1992). It has intimate connections to the representation learning capabilities of neural networks (Tishby and Zaslavsky, 2015) and underlies many celebrated modern approaches to unsupervised learning such as generative adversarial networks (GANs) (Goodfellow et al., 2014).

There is a recent burst of effort in learning continuous representations by optimizing various lower bounds on mutual information (Belghazi et al., 2018; Oord et al., 2018; Hjelm et al., 2018). These representations are typically eval-

Method	de	en	es	fr	id	it	ja	ko	pt-br	sv	Mean
Variational $\widehat{J}^{\text{var}}$ (7)	(±1.5) **75.4**	(±1.7) **73.1**	(±1.0) 73.1	(±2.9) 70.4	(±1.5) **73.6**	(±3.3) **67.4**	(±0.4) 77.9	(±1.2) **65.6**	(±2.3) **70.7**	(±1.5) **67.1**	**71.4**
Stratos et al.	63.4	71.4	**74.3**	**71.9**	67.3	60.2	69.4	61.8	65.8	61.0	66.7
Berg-Kirkpatrick et al.	(±1.8) 67.5	(±3.5) 62.4	(±3.1) 67.1	(±4.5) 62.1	(±3.9) 61.3	(±2.9) 52.9	(±2.9) **78.2**	(±3.6) 60.5	(±2.2) 63.2	(±2.5) 56.7	63.2
Brown et al.	60.0	62.9	67.4	66.4	59.3	66.1	60.3	47.5	67.4	61.9	61.9
Baum-Welch	(±4.8) 45.5	(±3.4) 59.8	(±2.2) 60.6	(±3.6) 60.1	(±3.1) 49.6	(±2.6) 51.5	(±2.1) 59.5	(±0.6) 51.7	(±3.7) 59.5	(±3.0) 42.4	54.0

Table 4: Many-to-one accuracy on the 12-tag universal treebank dataset. We use the same setting in Table 1. All models use a fixed hyperparameter configuration optimized on the 45-tag Penn WSJ.

Metric	Method	Arabic	Basque	Danish	Greek	Hungarian	Italian	Turkish	Mean
M2O	Variational $\widehat{J}^{\text{var}}$ (7)	**74.3**	**70.4**	**71.7**	**66.1**	**61.2**	**67.4**	**64.2**	**67.9**
	Ammar et al.	69.1	68.1	60.9	63.5	57.1	60.4	60.4	62.8
	Berg-Kirkpatrick et al.	66.8	66.2	60.0	60.2	56.8	64.1	62.0	62.3
	Baum-Welch	49.7	44.9	42.4	39.2	45.2	39.3	52.7	44.7
VM	Variational $\widehat{J}^{\text{var}}$ (7)	**56.9**	43.6	**56.0**	**56.3**	47.9	**53.3**	38.5	**50.4**
	Lin et al.	50.5	**51.7**	51.3	50.0	**55.9**	46.3	**43.1**	49.8
	Ammar et al.	49.1	41.1	46.1	49.1	41.1	43.1	35.0	43.5
	Berg-Kirkpatrick et al.	33.8	33.4	41.1	40.9	39.0	46.6	31.6	38.8
	Baum-Welch	15.3	8.2	11.1	9.6	10.1	9.9	11.6	10.8

Table 5: Comparison with the reported results with CRF autoencoders in many-to-one accuracy (M2O) and the V-measure (VM).

uated on extrinsic tasks as features. In contrast, we learn discrete representations by optimizing a novel generalization of the Brown clustering objective (Brown et al., 1992) and a variational lower bound on mutual information proposed by McAllester (2018). We focus on intrinsic evaluation of these representations on POS induction. Extrinsic evaluation of these representations in downstream tasks is an important future direction.

The issue of biased stochastic gradient estimators is a common challenge in unsupervised learning (e.g., see Wang et al., 2015). This arises mainly because the objective involves a nonlinear transformation of all samples in a training dataset, for instance the whitening constraints in deep canonical correlation analysis (CCA) (Andrew et al., 2013). In this work, the problem arises because of entropy. This issue is not considered in the original work of McAllester (2018) and the error analysis we present in Section 4 is novel. Our finding is that the feasibility of stochastic optimization greatly depends on the size of the bias in gradient estimates, as we are able to effectively optimize the variational objective while not the generalized Brown objective.

Our POS induction system has some practical advantages over previous approaches. Many rely on computationally expensive structured inference or pre-optimized features (or both). For instance, Tran et al. (2016) need to calculate forward/backward messages and is limited to truncated sequences by memory constraints. Berg-Kirkpatrick et al. (2010) rely on extensively hand-engineered linguistic features. Ammar et al. (2014), Lin et al. (2015), and He et al. (2018) rely on carefully pretrained lexical representations like Brown clusters and word embeddings. In contrast, the model presented in this work requires no expensive structured computation or feature engineering and uses word/character embeddings trained from scratch. It is easy to implement using a standard neural network library and outperforms these previous works in many cases.

Acknowledgments

The author thanks David McAllester for many insightful discussions, and Sam Wiseman for helpful comments. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems*, pages 3311–3319.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis.

In *Proceedings of the 30th International Conference on Machine Learning*, pages 1247–1255.

Ron Artstein. 2017. Inter-annotator agreement. In *Handbook of linguistic annotation*, pages 297–313. Springer.

Leonard E. Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

Justin B Kinney and Gurinder S Atwal. 2014. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, page 201309933.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1311–1316, Denver, Colorado. Association for Computational Linguistics.

David McAllester. 2018. Information theoretic co-training. *arXiv preprint arXiv:1802.07572*.

Ryan T. McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B. Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria B. Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *ACL*, pages 92–97.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing*, volume 12.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*, 4:245–257.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71. Association for Computational Linguistics.

Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. 2015. Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 688–695. IEEE.

A Proof of Theorem 4.1

We first analyze the variational loss

$$\widehat{H}(q,p) - \widehat{H}(Z)$$

Note that the cross entropy term decomposes over samples and causes no bias. Thus we focus on the negative entropy term

$$-\widehat{H}(Z) = \sum_z \hat{q}(z) \log \hat{q}(z)$$

whose gradient with respect to q is

$$\begin{aligned} &\sum_z \left(1 + \log \hat{q}(z)\right) \nabla \hat{q}(z) \\ &= \frac{1}{K} \sum_{k=1}^{K} \sum_z \left(1 + \log \hat{q}(z)\right) \nabla \hat{q}_k(z) \qquad (9) \end{aligned}$$

where we expand $\nabla \hat{q}(z)$ by the identity

$$\nabla \hat{q}(z) = \frac{1}{N} \sum_{i=1}^{N} \nabla q(z|y_i) = \frac{1}{K} \sum_{k=1}^{K} \nabla \hat{q}_k(z) \tag{10}$$

In contrast, the gradient of the negative entropy term averaged over minibatches is

$$\frac{1}{K} \sum_{k=1}^{K} \sum_z \left(1 + \log \hat{q}_k(z)\right) \nabla \hat{q}_k(z) \tag{11}$$

Hence the difference between (9) and (11) is

$$\frac{1}{K} \sum_{k=1}^{K} \sum_z \log \frac{\hat{q}(z)}{\hat{q}_k(z)} \nabla \hat{q}_k(z)$$

This shows the first result. Now we analyze the generalized Brown loss

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{z,z'} p(z|x_i) q(z'|y_i) \log \frac{\hat{p}(z)\hat{q}(z')}{p(z|x_i) q(z'|y_i)}$$

When we expand the log fraction, we see that the denominator decomposes over samples and causes no bias. Thus we focus on the numerator term

$$\frac{1}{N} \sum_{z,z'} \log \left(\hat{p}(z)\hat{q}(z')\right) \sum_{i=1}^{N} p(z|x_i) q(z'|y_i)$$

By the product rule, its gradient with respect to q is a sum of two terms. The first term is (using (10) again)

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \left(\frac{1}{K} \sum_{i=1}^{N} \frac{p(z|x_i) q(z'|y_i)}{\hat{q}(z')} \right) \nabla \hat{q}_k(z') \tag{12}$$

The second term is (as a sum over batches)

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \log \left(\hat{p}(z)\hat{q}(z')\right) \sum_{(x,y)\in B_k} p(z|x) \nabla q(z'|y) \tag{13}$$

In contrast, the numerator term estimated as an average over minibatches is

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \log \left(\hat{p}_k(z)\hat{q}_k(z')\right) \sum_{(x,y)\in B_k} p(z|x) q(z'|y)$$

and the two terms of its gradient with respect to q (corresponding to (12) and (13)) are

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \left(\sum_{(x,y)\in B_k} \frac{p(z|x) q(z'|y)}{\hat{q}_k(z')} \right) \nabla \hat{q}_k(z') \tag{14}$$

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \log \left(\hat{p}_k(z)\hat{q}_k(z')\right) \sum_{(x,y)\in B_k} p(z|x_i) \nabla q(z'|y) \tag{15}$$

Thus the difference between (12) and (14) is

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \epsilon_k(z,z') \nabla \hat{q}_k(z')$$

where

$$\begin{aligned} \epsilon_k(z,z') = &\frac{1}{K} \sum_{i=1}^{N} \frac{p(z|x_i) q(z'|y_i)}{\hat{q}(z')} \\ &- \sum_{(x,y)\in B_k} \frac{p(z|x) q(z'|y)}{\hat{q}_k(z')} \end{aligned}$$

The difference between (13) and (15) is

$$\frac{1}{N} \sum_{k=1}^{K} \sum_{z,z'} \log \frac{\hat{p}(z)\hat{q}(z')}{\hat{p}_k(z)\hat{q}_k(z')} \sum_{(x,y)\in B_k} p(z|x) \nabla q(z'|y)$$

Adding these differences gives the second result.

Unsupervised Recurrent Neural Network Grammars

Yoon Kim[†] **Alexander M. Rush**[†] **Lei Yu**[◇]
Adhiguna Kuncoro[‡,◇] **Chris Dyer**[◇] **Gábor Melis**[◇]

[†]Harvard University [‡]University of Oxford [◇]DeepMind
{yoonkim,srush}@seas.harvard.edu
{leiyu,akuncoro,cdyer,melisgl}@google.com

Abstract

Recurrent neural network grammars (RNNG) are generative models of language which jointly model syntax and surface structure by incrementally generating a syntax tree and sentence in a top-down, left-to-right order. Supervised RNNGs achieve strong language modeling and parsing performance, but require an annotated corpus of parse trees. In this work, we experiment with unsupervised learning of RNNGs. Since directly marginalizing over the space of latent trees is intractable, we instead apply amortized variational inference. To maximize the evidence lower bound, we develop an inference network parameterized as a neural CRF constituency parser. On language modeling, unsupervised RNNGs perform as well their supervised counterparts on benchmarks in English and Chinese. On constituency grammar induction, they are competitive with recent neural language models that induce tree structures from words through attention mechanisms.

1 Introduction

Recurrent neural network grammars (RNNGs) (Dyer et al., 2016) model sentences by first generating a nested, hierarchical syntactic structure which is used to construct a context representation to be conditioned upon for upcoming words. Supervised RNNGs have been shown to outperform standard sequential language models, achieve excellent results on parsing (Dyer et al., 2016; Kuncoro et al., 2017), better encode syntactic properties of language (Kuncoro et al., 2018), and correlate with electrophysiological responses in the human brain (Hale et al., 2018). However, these all require annotated syntactic trees for training. In this work, we explore unsupervised learning of recurrent neural network grammars for language modeling and grammar induction.

The standard setup for unsupervised structure learning is to define a generative model $p_\theta(\mathbf{x}, \mathbf{z})$ over observed data $\mathbf{x}$ (e.g. sentence) and unobserved structure $\mathbf{z}$ (e.g. parse tree, part-of-speech sequence), and maximize the log marginal likelihood $\log p_\theta(\mathbf{x}) = \log \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z})$. Successful approaches to unsupervised parsing have made strong conditional independence assumptions (e.g. context-freeness) and employed auxiliary objectives (Klein and Manning, 2002) or priors (Johnson et al., 2007). These strategies imbue the learning process with inductive biases that guide the model to discover meaningful structures while allowing tractable algorithms for marginalization; however, they come at the expense of language modeling performance, particularly compared to sequential neural models that make no independence assumptions.

Like RNN language models, RNNGs make no independence assumptions. Instead they encode structural bias through operations that compose linguistic constituents. The lack of independence assumptions contributes to the strong language modeling performance of RNNGs, but make unsupervised learning challenging. First, marginalization is intractable. Second, the biases imposed by the RNNG are relatively weak compared to those imposed by models like PCFGs. There is little pressure for non-trivial tree structure to emerge during unsupervised RNNG (URNNG) learning.

In this work, we explore a technique for handling intractable marginalization while also injecting inductive bias. Specifically we employ amortized variational inference (Kingma and Welling, 2014; Rezende et al., 2014; Mnih and Gregor, 2014) with a *structured* inference network. Variational inference lets us tractably optimize a lower bound on the log marginal likelihood, while employing a structured inference network encourages non-trivial structure. In particular, a con-

---

Work done while the first author was an intern at DeepMind. Code available at https://github.com/harvardnlp/urnng

Proceedings of NAACL-HLT 2019, pages 1105–1117
Minneapolis, Minnesota, June 2 - June 7, 2019.

ditional random field (CRF) constituency parser (Finkel et al., 2008; Durrett and Klein, 2015), which makes significant independence assumptions, acts as a guide on the generative model to learn meaningful trees through regularizing the posterior (Ganchev et al., 2010).

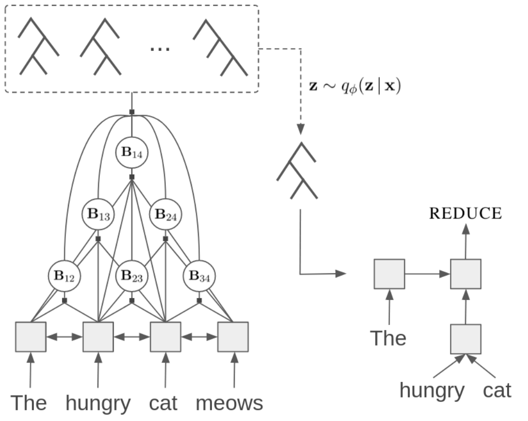
We experiment with URNNGs on English and Chinese and observe that they perform well as language models compared to their supervised counterparts and standard neural LMs. In terms of grammar induction, they are competitive with recently-proposed neural architectures that discover tree-like structures through gated attention (Shen et al., 2018). Our results, along with other recent work on joint language modeling/structure learning with deep networks (Shen et al., 2018, 2019; Wiseman et al., 2018; Kawakami et al., 2018), suggest that it is possible learn generative models of language that model the underlying data well (i.e. assign high likelihood to held-out data) and at the same time induce meaningful linguistic structure.

2 Unsupervised Recurrent Neural Network Grammars

We use $\mathbf{x} = [x_1, \dots, x_T]$ to denote a sentence of length T, and $\mathbf{z} \in \mathcal{Z}_T$ to denote an unlabeled binary parse tree over a sequence of length T, represented as a a binary vector of length $2T - 1$. Here 0 and 1 correspond to SHIFT and REDUCE actions, explained below.[1] Figure 1 presents an overview of our approach.

2.1 Generative Model

An RNNG defines a joint probability distribution $p_\theta(\mathbf{x}, \mathbf{z})$ over sentences $\mathbf{x}$ and parse trees $\mathbf{z}$. We consider a simplified version of the original RNNG (Dyer et al., 2016) by ignoring constituent labels and only considering binary trees. The RNNG utilizes an RNN to parameterize a stack data structure (Dyer et al., 2015) of partially-completed constituents to incrementally build the parse tree while generating terminals. Using the current stack representation, the model samples an action (SHIFT or REDUCE): SHIFT generates a terminal symbol, i.e. word, and shifts it onto the stack,[2] REDUCE pops the last two elements off the stack, composes them, and shifts the composed representation onto the stack.



Inference Network $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ Generative Model $p_\theta(\mathbf{x}, \mathbf{z})$

Figure 1: Overview of our approach. The inference network $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ (left) is a CRF parser which produces a distribution over binary trees (shown in dotted box). $\mathbf{B}_{ij}$ are random variables for existence of a constituent spanning i-th and j-th words, whose potentials are the output from a bidirectional LSTM (the global factor ensures that the distribution is only over valid binary trees). The generative model $p_\theta(\mathbf{x}, \mathbf{z})$ (right) is an RNNG which consists of a stack LSTM (from which actions/words are predicted) and a tree LSTM (to obtain constituent representations upon REDUCE). Training involves sampling a binary tree from $q_\phi(\mathbf{z} \,|\, \mathbf{x})$, converting it to a sequence of shift/reduce actions, and optimizing the log joint likelihood $\log p_\theta(\mathbf{x}, \mathbf{z})$.

Formally, let $S = [(\mathbf{0}, \mathbf{0})]$ be the initial stack. Each item of the stack will be a pair, where the first element is the hidden state of the stack LSTM, and the second element is an input vector, described below. We use $\text{top}(S)$ to refer to the top pair in the stack. The push and pop operations are defined imperatively in the usual way. At each time step, the next action z_t (SHIFT or REDUCE) is sampled from a Bernoulli distribution parameterized in terms of the current stack representation. Letting $(\mathbf{h}_{\text{prev}}, \mathbf{g}_{\text{prev}}) = \text{top}(S)$, we have

$$z_t \sim \text{Bernoulli}(p_t), \qquad p_t = \sigma(\mathbf{w}^\top \mathbf{h}_{\text{prev}} + b).$$

Subsequent generation depend on z_t:

- If $z_t = 0$ (SHIFT), the model first generates a terminal symbol via sampling from a categorical distribution whose parameters come from an affine transformation and a softmax,

  $$x \sim \text{softmax}(\mathbf{W}\mathbf{h}_{\text{prev}} + \mathbf{b}).$$

  Then the generated terminal is shifted onto the stack using a stack LSTM,

  $$\mathbf{h}_{\text{next}} = \text{LSTM}(\mathbf{e}_x, \mathbf{h}_{\text{prev}}),$$
  $$\text{push}(S, (\mathbf{h}_{\text{next}}, \mathbf{e}_x)),$$

  where $\mathbf{e}_x$ is the word embedding for x.

---

[1] The cardinality of $\mathcal{Z}_T \subset \{0, 1\}^{2T-1}$ is given by the $(T-1)$-th Catalan number, $|\mathcal{Z}_T| = \frac{(2T-2)!}{T!(T-1)!}$.

[2] A better name for SHIFT would be GENERATE (as in Dyer et al. (2016)), but we use SHIFT to emphasize similarity with the shift-reduce parsing.

- If $z_t = 1$ (REDUCE), we pop the last two elements off the stack,
$$(\mathbf{h}_r, \mathbf{g}_r) = \text{pop}(S), \quad (\mathbf{h}_l, \mathbf{g}_l) = \text{pop}(S),$$
and obtain a new representation that combines the left/right constituent representations using a tree LSTM (Tai et al., 2015; Zhu et al., 2015),
$$\mathbf{g}_{\text{new}} = \text{TreeLSTM}(\mathbf{g}_l, \mathbf{g}_r).$$
Note that we use $\mathbf{g}_l$ and $\mathbf{g}_r$ to obtain the new representation instead of $\mathbf{h}_l$ and $\mathbf{h}_r$.[3] We then update the stack using $\mathbf{g}_{\text{new}}$,
$$\begin{aligned}&(\mathbf{h}_{\text{prev}}, \mathbf{g}_{\text{prev}}) = \text{top}(S),\\ &\mathbf{h}_{\text{new}} = \text{LSTM}(\mathbf{g}_{\text{new}}, \mathbf{h}_{\text{prev}}),\\ &\text{push}(S, (\mathbf{h}_{\text{new}}, \mathbf{g}_{\text{new}})).\end{aligned}$$

The generation process continues until an end-of-sentence symbol is generated. The parameters θ of the generative model are $\mathbf{w}, b, \mathbf{W}, \mathbf{b}$, and the parameters of the stack/tree LSTMs. For a sentence $\mathbf{x} = [x_1, \ldots, x_T]$ of length T, the binary parse tree is given by the binary vector $\mathbf{z} = [z_1, \ldots, z_{2T-1}]$.[4] The joint log likelihood decomposes as a sum of terminal/action log likelihoods,

$$\log p_\theta(\mathbf{x}, \mathbf{z}) = \underbrace{\sum_{t=1}^{T} \log p_\theta(x_t \,|\, \mathbf{x}_{<t}, \mathbf{z}_{<n(t)})}_{\log p_\theta(\mathbf{x} \,|\, \mathbf{z})} + \underbrace{\sum_{j=1}^{2T-1} \log p_\theta(z_j \,|\, \mathbf{x}_{<m(j)}, \mathbf{z}_{<j})}_{\log p_\theta(\mathbf{z} \,|\, \mathbf{x}_{<\mathbf{z}})}, \quad (1)$$

where $\mathbf{z}_{<n(t)}$ refers to all actions before generating the t-th word, and similarly $\mathbf{x}_{<m(j)}$ refers to all words generated before taking the j-th action. For brevity, from here on we will use $\log p_\theta(\mathbf{x} \,|\, \mathbf{z})$ to refer to the first term (terminal log likelihood) and $\log p_\theta(\mathbf{z} \,|\, \mathbf{x}_{<\mathbf{z}})$ to refer to the second term (action log likelihood) in the above decomposition.[5]

In the supervised case where ground-truth $\mathbf{z}$ is available, we can straightforwardly perform gradient-based optimization to maximize the joint log likelihood $\log p_\theta(\mathbf{x}, \mathbf{z})$. In the unsupervised case, the standard approach is to maximize the log marginal likelihood,

$$\log p_\theta(\mathbf{x}) = \log \sum_{\mathbf{z}' \in \mathcal{Z}_T} p_\theta(\mathbf{x}, \mathbf{z}').$$

However this summation is intractable because z_t fully depends on all previous actions $[z_1, \ldots, z_{t-1}]$. Even if this summation were tractable, it is not clear that meaningful latent structures would emerge given the lack of explicit independence assumptions in the RNNG (e.g. it is clearly not context-free). We handle these issues with amortized variational inference.

2.2 Amortized Variational Inference

Amortized variational inference (Kingma and Welling, 2014) defines a trainable inference network ϕ that parameterizes $q_\phi(\mathbf{z} \,|\, \mathbf{x})$, a variational posterior distribution, in this case over parse trees $\mathbf{z}$ given the sentence $\mathbf{x}$. This distribution is used to form an evidence lower bound (ELBO) on the log marginal likelihood,

$$\text{ELBO}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \,|\, \mathbf{x})}\right].$$

We maximize the ELBO with respect to both model parameters θ and inference network parameters ϕ. The ELBO is still intractable to calculate exactly, but this formulation will allow us to obtain unbiased gradient estimators based on Monte Carlo sampling.

Observe that rearranging the ELBO gives the following optimization problem,

$$\max_{\theta, \phi} \; \log p_\theta(\mathbf{x}) - \text{KL}[q_\phi(\mathbf{z} \,|\, \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \,|\, \mathbf{x})].$$

Thus, ϕ is trained to match the variational posterior $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ to the true posterior $p_\theta(\mathbf{z} \,|\, \mathbf{x})$, but θ is *also* trained to match the true posterior to the variational posterior. Indeed, there is some evidence to suggest that generative models trained with amortized variational inference (i.e. variational autoencoders) learn posterior distributions that are close to the variational family (Cremer et al., 2018).

We can use this to our advantage with an inference network that injects inductive bias. We propose to do this by using a context-free model for the inference network, in particular, a neural CRF parser (Durrett and Klein, 2015). This choice

---

[3] The update equations for the tree LSTM (and the stack LSTM) also involve *cell* states in addition to the hidden states. To reduce notational clutter we do not explicitly show the cell states and instead subsume them into $\mathbf{g}$. If one (or both) of the inputs to the tree LSTM is a word embedding, the associated cell state is taken to be zero. See Tai et al. (2015) for the exact parameterization.

[4] As it stands, the support of $\mathbf{z}$ is $\{0, 1\}^{2T-1}$, all binary vectors of length $2T - 1$. To restrict our distribution to $\mathcal{Z}_T$ (binary vectors which describe valid trees), we constrain z_t to be valid at each time step, which amounts to deterministically choosing $z_t = 0$ (SHIFT) if there are fewer than two elements (not counting the initial zero tuple) on the stack.

[5] The action log likelihood is the sum of log *conditional* priors, which is obviously different from the unconditional log prior $\log p_\theta(\mathbf{z}) = \log \sum_{\mathbf{x}} p_\theta(\mathbf{x}, \mathbf{z})$.

can seen as a form of posterior regularization that limits posterior flexibility of the overly powerful RNNG generative model.[6,7]

The parameterization of span scores is similar to recent works (Wang and Chang, 2016; Stern et al., 2017; Kitaev and Klein, 2018): we add position embeddings to word embeddings and run a bidirectional LSTM over the input representations to obtain the forward $[\overrightarrow{\mathbf{h}}_1, \dots, \overrightarrow{\mathbf{h}}_T]$ and backward $[\overleftarrow{\mathbf{h}}_1, \dots, \overleftarrow{\mathbf{h}}_T]$ hidden states. The score $s_{ij} \in \mathbb{R}$ for a constituent spanning x_i to x_j is given by,

$$s_{ij} = \text{MLP}([\overrightarrow{\mathbf{h}}_{j+1} - \overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_{i-1} - \overleftarrow{\mathbf{h}}_j]).$$

Letting $\mathbf{B}$ be the binary matrix representation of a tree ($\mathbf{B}_{ij} = 1$ means there is a constituent spanning x_i and x_j), the CRF parser defines a distribution over binary trees via the Gibbs distribution,

$$q_\phi(\mathbf{B} \,|\, \mathbf{x}) = \frac{1}{Z_T(\mathbf{x})} \exp\Big(\sum_{i \le j} \mathbf{B}_{ij} s_{ij}\Big),$$

where $Z_T(\mathbf{x})$ is the partition function,

$$Z_T(\mathbf{x}) = \sum_{\mathbf{B}' \in \mathcal{B}_T} \exp\Big(\sum_{i \le j} \mathbf{B}'_{ij} s_{ij}\Big),$$

and ϕ denotes the parameters of the inference network (i.e. the bidirectional LSTM and the MLP). Calculating $Z_T(\mathbf{x})$ requires a summation over an exponentially-sized set $\mathcal{B}_T \subset \{0, 1\}^{T \times T}$, the set of all binary trees over a length T sequence. However we can perform the summation in $O(T^3)$ using the inside algorithm (Baker, 1979), shown in Algorithm 1. This computation is itself differentiable and amenable to gradient-based optimization. Finally, letting $f : \mathcal{B}_T \to \mathcal{Z}_T$ be the bijection between the binary tree matrix representation and a sequence of SHIFT/REDUCE actions, the inference network defines a distribution over $\mathcal{Z}_T$ via $q_\phi(\mathbf{z} \,|\, \mathbf{x}) \triangleq q_\phi(f^{-1}(\mathbf{z}) \,|\, \mathbf{x})$.

Algorithm 1 Inside algorithm for calculating $Z_T(\mathbf{x})$

```
procedure INSIDE($s$)                              ▷ scores $s_{ij}$ for $i \le j$
    for $i := 1$ to $T$ do                          ▷ length-1 spans
        $\beta[i, i] = s_{ii}$
    for $\ell := 1$ to $T - 1$ do                   ▷ span length
        for $i := 1$ to $T - \ell$ do               ▷ span start
            $j = i + \ell$                          ▷ span end
            $\beta[i, j] = \sum_{k=i}^{j-1} s_{ij} \cdot \beta[i, k] \cdot \beta[k + 1, j]$
    return $\beta[1, T]$              ▷ return partition function $Z_T(\mathbf{x})$
```

2.3 Optimization

For optimization, we use the following variant of the ELBO,

$$\mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z})] + \mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})],$$

where $\mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})] = \mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}[-\log q_\phi(\mathbf{z} \,|\, \mathbf{x})]$ is the entropy of the variational posterior. A Monte Carlo estimate for the gradient with respect to θ is

$$\nabla_\theta \, \text{ELBO}(\theta, \phi; \mathbf{x}) \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}^{(k)}),$$

with samples $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(K)}$ from $q_\phi(\mathbf{z} \,|\, \mathbf{x})$. Sampling uses the intermediate values calculated during the inside algorithm to sample split points recursively (Goodman, 1998; Finkel et al., 2006), as shown in Algorithm 2. The gradient with respect to ϕ involves two parts. The entropy term $\mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$ can be calculated exactly in $O(T^3)$, again using the intermediate values from the inside algorithm (see Algorithm 3).[8] Since each step of this dynamic program is differentiable, we can obtain the gradient $\nabla_\phi \mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$ using automatic differentation.[9] An estimator for the gradient with respect to $\mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z})]$ is obtained via the score function gradient estimator (Glynn, 1987; Williams, 1992),

$$\begin{aligned}
&\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z})] \\
&\quad = \mathbb{E}_{q_\phi(\mathbf{z} \,|\, \mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z} \,|\, \mathbf{x})] \\
&\quad \approx \frac{1}{K} \sum_{k=1}^{K} \log p_\theta(\mathbf{x}, \mathbf{z}^{(k)}) \nabla_\phi \log q_\phi(\mathbf{z}^{(k)} \,|\, \mathbf{x}).
\end{aligned}$$

---

[6] While it has a similar goal, this formulation differs the from posterior regularization as formulated by Ganchev et al. (2010), which constrains the distributional family via linear constraints on posterior expectations. In our case, the conditional independence assumptions in the CRF lead to a *curved* exponential family where the vector of natural parameters has fewer dimensions than the vector of sufficient statistics of the full exponential family. This curved exponential family is a subset of the marginal polytope of the full exponential family, but it is an intersection of both linear and nonlinear manifolds, and therefore cannot be characterized through linear constraints over posterior expectations.

[7] In preliminary experiments, we also attempted to learn latent trees with a transition-based parser (which does not make explicit independence assumptions) that looks at the entire sentence. However we found that under this setup, the inference network degenerated into a local minimum whereby it always generated left-branching trees despite various optimization strategies. Williams et al. (2018) observe a similar phenomenon in the context of learning latent trees for classification tasks. However Li et al. (2019) find that it is possible use a transition-based parser as the inference network for dependency grammar induction, if the inference network is constrained via posterior regularization (Ganchev et al., 2010) based on universal syntactic rules (Naseem et al., 2010).

[8] We adapt the algorithm for calculating tree entropy in PCFGs from Hwa (2000) to the CRF case.

[9] $\nabla_\phi \mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$ can also be computed using the inside-outside algorithm and a second-order expectation semiring (Li and Eisner, 2009), which has the same asymptotic runtime complexity but generally better constants.

Algorithm 2 Top-down sampling a tree from $q_\phi(\mathbf{z} \,|\, \mathbf{x})$

```
procedure SAMPLE($\beta$)                    ▷ $\beta$ from running INSIDE(s)
  $\mathbf{B} = \mathbf{0}$                  ▷ binary matrix representation of tree
  $Q = [(1, T)]$                             ▷ queue of constituents
  while $Q$ is not empty do
    $(i, j) = \text{pop}(Q)$
    $\tau = \sum_{k=i}^{j-1} \beta[i, k] \cdot \beta[k+1, j]$
    for $k := i$ to $j - 1$ do               ▷ get distribution over splits
      $w_k = (\beta[i, k] \cdot \beta[k+1, j])/\tau$
    $k \sim \text{Cat}([w_i, \dots, w_{j-1}])$   ▷ sample a split point
    $\mathbf{B}_{i,k} = 1,\ \mathbf{B}_{k+1,j} = 1$   ▷ update B
    if $k > i$ then                          ▷ if left child has width > 1
      push($Q$, $(i, k)$)                    ▷ add to queue
    if $k + 1 < j$ then                      ▷ if right child has width > 1
      push($Q$, $(k+1, j)$)                  ▷ add to queue
  $\mathbf{z} = f(\mathbf{B})$               ▷ $f : \mathcal{B}_T \to \mathcal{Z}_T$ maps matrix representation of tree to sequence of actions.
  return z
```

The above estimator is unbiased but typically suffers from high variance. To reduce variance, we use a control variate derived from an average of the other samples' joint likelihoods (Mnih and Rezende, 2016), yielding the following estimator,

$$\frac{1}{K}\sum_{k=1}^{K}(\log p_\theta(\mathbf{x}, \mathbf{z}^{(k)}) - r^{(k)})\nabla_\phi \log q_\phi(\mathbf{z}^{(k)} \,|\, \mathbf{x}),$$

where $r^{(k)} = \frac{1}{K-1}\sum_{j \neq k} \log p_\theta(\mathbf{x}, \mathbf{z}^{(j)})$. This control variate worked better than alternatives such as estimates of baselines from an auxiliary network (Mnih and Gregor, 2014; Deng et al., 2018) or a language model (Yin et al., 2018).

3 Experimental Setup

3.1 Data

For English we use the Penn Treebank (Marcus et al., 1993, PTB) with splits and preprocessing from Dyer et al. (2016) which retains punctuation and replaces singleton words with Berkeley parser's mapping rules, resulting in a vocabulary of 23,815 word types.[10] Notably this is much larger than the standard PTB LM setup from Mikolov et al. (2010) which uses 10K types.[11] Also different from the LM setup, we model each sentence separately instead of carrying information across sentence boundaries, as the RNNG is a generative model of sentences. Hence our perplexity numbers are not comparable to the PTB LM results (Melis et al., 2018; Merity et al., 2018; Yang et al., 2018).

Since the PTB is rather small, and since the URNNG does not require annotation, we also test our approach on a subset of the one billion word corpus (Chelba et al., 2013). We randomly sample 1M sentences for training and 2K sentences for validation/test, and limit the vocabulary to 30K word types. While still a subset of the full corpus (which has 30M sentences), this dataset is two orders of magnitude larger than PTB. Experiments on Chinese utilize version 5.1 of the Chinese Penn Treebank (CTB) (Xue et al., 2005), with the same splits as in Chen and Manning (2014). Singleton words are replaced with a single ⟨UNK⟩ token, resulting in a vocabulary of 17,489 word types.

Algorithm 3 Calculating the tree entropy $\mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$

```
procedure ENTROPY($\beta$)                   ▷ $\beta$ from running INSIDE(s)
  for $i := 1$ to $T$ do                     ▷ initialize entropy table
    $H[i, i] = 0$
  for $l := 1$ to $T - 1$ do                 ▷ span length
    for $i := 1$ to $T - l$ do               ▷ span start
      $j = i + l$                            ▷ span end
      $\tau = \sum_{u=i}^{j-1} \beta[i, u] \cdot \beta[u+1, j]$
      for $u := i$ to $j - 1$ do
        $w_u = (\beta[i, u] \cdot \beta[u+1, j])/\tau$
      $H[i, j] = \sum_{u=i}^{j-1}(H[i, u] + H[u+1, j]$
                    $- \log w_u) \cdot w_u$
  return $H[1, T]$                           ▷ return tree entropy $\mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$
```

3.2 Training and Hyperparameters

The stack LSTM has two layers with input/hidden size equal to 650 and dropout of 0.5. The tree LSTM also has 650 units. The inference network uses a one-layer bidirectional LSTM with 256 hidden units, and the MLP (to produce span scores s_{ij} for $i \leq j$) has a single hidden layer with a ReLU nonlinearity followed by layer normalization (Ba et al., 2016) and dropout of 0.5. We share word embeddings between the generative model and the inference network, and also tie weights between the input/output word embeddings (Press and Wolf, 2016).

Optimization of the model itself required standard techniques for avoiding posterior collapse in VAEs.[12] We warm-up the ELBO objective by linearly annealing (per batch) the weight on the conditional prior $\log p_\theta(\mathbf{z} \,|\, \mathbf{x}_{<\mathbf{z}})$ and the entropy $\mathbb{H}[q_\phi(\mathbf{z} \,|\, \mathbf{x})]$ from 0 to 1 over the first two epochs (see equation (1) for definition of $\log p_\theta(\mathbf{z} \,|\, \mathbf{x}_{<\mathbf{z}})$). This is analogous to KL-annealing in VAEs with continuous latent variables (Bowman et al., 2016; Sønderby et al., 2016). We train for 18 epochs (enough for convergence for all models) with a batch size of 16 and $K = 8$ samples for the Monte Carlo gradient estimators. The generative model is optimized with SGD with learning rate equal to 1,

---

[10] https://github.com/clab/rnng

[11] Both versions of the PTB data can be obtained from http://demo.clab.cs.cmu.edu/cdyer/ptb-lm.tar.gz.

[12] Posterior collapse in our context means that $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ always produced trivial (always left or right branching) trees.

except for the affine layer that produces a distribution over the actions, which has learning rate 0.1. Gradients of the generative model are clipped at 5. The inference network is optimized with Adam (Kingma and Ba, 2015) with learning rate 0.0001, $\beta_1 = 0.9, \beta_2 = 0.999$, and gradient clipping at 1. As Adam converges significantly faster than SGD (even with a much lower learning rate), we stop training the inference network after the first two epochs. Initial model parameters are sampled from $\mathcal{U}[-0.1, 0.1]$. The learning rate starts decaying by a factor of 2 each epoch after the first epoch at which validation performance does not improve, but this learning rate decay is not triggered for the first eight epochs to ensure adequate training. We use the same hyperparameters/training setup for both PTB and CTB. For experiments on (the subset of) the one billion word corpus, we use a smaller dropout rate of 0.1. The baseline RNNLM also uses the smaller dropout rate.

All models are trained with an end-of-sentence token, but for perplexity calculation these tokens are not counted to be comparable to prior work (Dyer et al., 2016; Kuncoro et al., 2017; Buys and Blunsom, 2018). To be more precise, the inference network does not make use of the end-of-sentence token to produce parse trees, but the generative model is trained to generate the end-of-sentence token after the final REDUCE operation.

3.3 Baselines

We compare the unsupervised RNNG (URNNG) against several baselines: (1) RNNLM, a standard RNN language model whose size is the same as URNNG's stack LSTM; (2) Parsing Reading Predict Network (PRPN) (Shen et al., 2018), a neural language model that uses gated attention layers to embed soft tree-like structures into a neural network (and among the current state-of-the-art in grammar induction from words on the full corpus); (3) RNNG with trivial trees (left branching, right branching, random); (4) supervised RNNG trained on unlabeled, binarized gold trees.[13] Note that the supervised RNNG also trains a discriminative parser $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ (alongside the generative model $p_\theta(\mathbf{x}, \mathbf{z})$) in order to sample parse forests for perplexity evaluation (i.e. importance sampling). This discriminative parser has the same architecture as URNNG's inference network. For all models, we perform early stopping based on validation perplexity.

Model	PTB PPL	PTB F_1	CTB PPL	CTB F_1
RNNLM	93.2	–	201.3	–
PRPN (default)	126.2	32.9	290.9	32.9
PRPN (tuned)	96.7	41.2	216.0	36.1
Left Branching Trees	100.9	10.3	223.6	12.4
Right Branching Trees	93.3	34.8	203.5	20.6
Random Trees	113.2	17.0	209.1	17.4
URNNG	90.6	40.7	195.7	29.1
RNNG	88.7	68.1	193.1	52.3
RNNG → URNNG	85.9	67.7	181.1	51.9
Oracle Binary Trees	–	82.5	–	88.6

Table 1: Language modeling perplexity (PPL) and grammar induction F_1 scores on English (PTB) and Chinese (CTB) for the different models. Note that our PTB setup from Dyer et al. (2016) differs considerably from the usual language modeling setup (Mikolov et al., 2010) since we model each sentence independently and use a much larger vocabulary (see §3.1).

4 Results and Discussion

4.1 Language Modeling

Table 1 shows perplexity for the different models on PTB/CTB. As a language model URNNG outperforms an RNNLM and is competitive with the supervised RNNG.[14] The left branching baseline performs poorly, implying that the strong performance of URNNG/RNNG is not simply due to the additional depth afforded by the tree LSTM composition function (a left branching tree, which always performs REDUCE when possible, is the "deepest" model). The right branching baseline is essentially equivalent to an RNNLM and hence performs similarly. We found PRPN with default hyperparameters (which obtains a perplexity of 62.0 in the PTB setup from Mikolov et al. (2010)) to not perform well, but tuning hyperparameters improves performance.[15] The supervised RNNG performs well as a language model, despite being trained on the joint (rather than marginal) likelihood objective.[16] This indicates that explicit

---

[13] We use right branching binarization—Matsuzaki et al. (2005) find that differences between various binarization schemes have marginal impact. Our supervised RNNG therefore differs the original RNNG, which trains on non-binarized trees and does not ignore constituent labels.

[14] For RNNG and URNNG we estimate the log marginal likelihood (and hence, perplexity) with $K = 1000$ importance-weighted samples, $\log p_\theta(\mathbf{x}) \approx \log\left(\frac{1}{K}\sum_{k=1}^{K} \frac{\log p(\mathbf{x},\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)} \,|\, \mathbf{x})}\right)$. During evaluation only, we also flatten $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ by dividing span scores s_{ij} by a temperature term 2.0 before feeding it to the CRF.

[15] Using the code from https://github.com/yikangshen/PRPN, we tuned model size, initialization, dropout, learning rate, and use of batch normalization.

[16] RNNG is trained to maximize $\log p_\theta(\mathbf{x}, \mathbf{z})$ while URNNG is trained to maximize (a lower bound on) the language modeling objective $\log p_\theta(\mathbf{x})$.

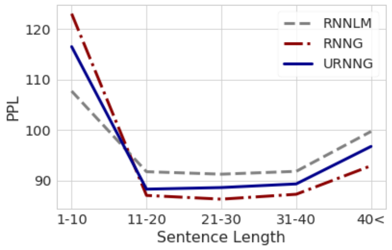


Figure 2: Perplexity of the different models grouped by sentence length on PTB.

modeling of syntax helps generalization even with richly-parameterized neural models. Encouraged by these observations, we also experiment with a hybrid approach where we train a supervised RNNG first and continue fine-tuning the model (including the inference network) on the URNNG objective (RNNG → URNNG in Table 1).[17] This approach results in nontrivial perplexity improvements, and suggests that it is potentially possible to improve language models with supervision on parsed data. In Figure 2 we show perplexity by sentence length. We find that a standard language model (RNNLM) is better at modeling short sentences, but underperforms models that explicitly take into account structure (RNNG/URNNG) when the sentence length is greater than 10. Table 2 (top) compares our results against prior work on this version of the PTB, and Table 2 (bottom) shows the results on a 1M sentence subset of the one billion word corpus, which is two orders of magnitude larger than PTB. On this larger dataset URNNG still improves upon the RNNLM. We also trained an RNNG (and RNNG → URNNG) on this dataset by parsing the training set with the self-attentive parser from Kitaev and Klein (2018).[18] These models improve upon the RNNLM but not the URNNG, potentially highlighting the limitations of using predicted trees for supervising RNNGs.

PTB	PPL
KN 5-gram (Dyer et al., 2016)	169.3
RNNLM (Dyer et al., 2016)	113.4
Original RNNG (Dyer et al., 2016)	102.4
Stack-only RNNG (Kuncoro et al., 2017)	101.2
Gated-Attention RNNG (Kuncoro et al., 2017)	100.9
Generative Dep. Parser (Buys and Blunsom, 2015)	138.6
RNNLM (Buys and Blunsom, 2018)	100.7
Sup. Syntactic NLM (Buys and Blunsom, 2018)	107.6
Unsup. Syntactic NLM (Buys and Blunsom, 2018)	125.2
PRPN† (Shen et al., 2018)	96.7
This work:	
RNNLM	93.2
URNNG	90.6
RNNG	88.7
RNNG → URNNG	85.9
1M Sentences	**PPL**
PRPN† (Shen et al., 2018)	77.7
RNNLM	77.4
URNNG	71.8
RNNG‡	72.9
RNNG‡ → URNNG	72.0

Table 2: (Top) Comparison of this work as a language model against prior works on sentence-level PTB with preprocessing from Dyer et al. (2016). Note that previous versions of RNNG differ from ours in terms of parameterization and model size. (Bottom) Results on a subset (1M sentences) of the one billion word corpus. PRPN† is the model from Shen et al. (2018), whose hyperparameters were tuned by us. RNNG‡ is trained on predicted parse trees from Kitaev and Klein (2018).

4.2 Grammar Induction

Table 1 also shows the F_1 scores for grammar induction. Note that we induce latent trees directly from words on the full dataset.[19] For RNNG/URNNG we obtain the highest scoring tree from $q_\phi(\mathbf{z} \,|\, \mathbf{x})$ through the Viterbi inside (i.e. CKY) algorithm. We calculate unlabeled F_1 using `evalb`, which ignores punctuation and discards trivial spans (width-one and sentence spans).[20] Since we compare F_1 against the original, non-binarized trees (per convention), F_1 scores of models using oracle binarized trees constitute the upper bounds.

We confirm the replication study of Htut et al. (2018) and find that PRPN is a strong model for grammar induction. URNNG performs on par with PRPN on English but PRPN does better on Chinese; both outperform right branching baselines. Table 3 further analyzes the learned trees and shows the F_1 score of URNNG trees against

---

[17] We fine-tune for 10 epochs and use a smaller learning rate of 0.1 for the generative model.

[18] To parse the training set we use the `benepar_en2` model from https://github.com/nikitakit/self-attentive-parser, which obtains an F_1 score of 95.17 on the PTB test set.

[19] Past work on grammar induction usually train/evaluate on short sentences and also assume access to gold POS tags (Klein and Manning, 2002; Smith and Eisner, 2004; Bod, 2006). However more recent works do train directly words (Jin et al., 2018; Shen et al., 2018; Drozdov et al., 2019).

[20] Available at https://nlp.cs.nyu.edu/evalb/. We evaluate with `COLLINS.prm` parameter file and `LABELED` option equal to 0. We observe that the setup for grammar induction varies widely across different papers: lexicalized vs. unlexicalized; use of punctuation vs. not; separation of train/test sets; counting sentence-level spans for evaluation vs. ignoring them; use of additional data; length cutoff for training/evaluation; corpus-level F_1 vs. sentence-level F_1; and, more. In our survey of twenty or so papers, almost no two papers were identical in their setup. Such variation makes it difficult to meaningfully compare models across papers. Hence, we report grammar induction results mainly for the models and baselines considered in the present work.

Tree	PTB	CTB
Gold	40.7	29.1
Left	9.2	8.4
Right	68.3	51.2
Self	92.3	87.3
RNNG	55.4	47.1
PRPN	41.0	47.2

Label	URNNG	PRPN
SBAR	74.8%	28.9%
NP	39.5%	63.9%
VP	76.6%	27.3%
PP	55.8%	55.1%
ADJP	33.9%	42.5%
ADVP	50.4%	45.1%

Table 3: (Left) F_1 scores of URNNG against other trees. "Self" refers to another URNNG trained with a different random seed. (Right) Recall of constituents by label for URNNG and PRPN. Recall for a particular label is the fraction of ground truth constituents of that label that were identified by the model.

	F_1	+PP
PRPN-UP‡	39.8	45.4
PRPN-LM‡	42.8	42.4
ON-LSTM‡ (Shen et al., 2019)	49.4	–
DIORA‡ (Drozdov et al., 2019)	49.6	56.2
PRPN (tuned)	49.0	49.9
URNNG	52.4	52.4

Table 4: PTB F_1 scores using the same evaluation setup as Drozdov et al. (2019), which evaluates against binarized trees, counts punctuation and trivial spans, and uses sentence-level F_1. +PP indicates a post-processing heuristic which directly attaches trailing punctuation to the root. This does not change URNNG results since it learns to do so anyway. Results with ‡ are copied from Table 1 of Drozdov et al. (2019).

other trees (left), and the recall of URNNG/PRPN trees against ground truth constituents (right). We find that trees induced by URNNG and PRPN are quite different; URNNG is more sensitive to SBAR and VP, while PRPN is better at identifying NP. While left as future work, this naturally suggests a hybrid approach wherein the intersection of constituents from URNNG and PRPN is used to create a corpus of partially annotated trees, which can be used to guide another model, e.g. via posterior regularization (Ganchev et al., 2010) or semi-supervision (Hwa, 1999). Finally, Table 4 compares our results using the same evaluation setup as in Drozdov et al. (2019), which differs considerably from our setup.

4.3 Distributional Metrics

Table 5 shows some standard metrics related to the learned generative model/inference network. The "reconstruction" perplexity based on $\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$ is much lower than regular perplexity, and further, the Kullback-Leibler divergence between the conditional prior and the variational posterior, given by

$$\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x}_{<\mathbf{z}})}\right],$$

	PTB		CTB	
	RNNG	URNNG	RNNG	URNNG
PPL	88.7	90.6	193.1	195.7
Recon. PPL	74.6	73.4	183.4	151.9
KL	7.10	6.13	11.11	8.91
Prior Entropy	7.65	9.61	9.48	15.13
Post. Entropy	1.56	2.28	6.23	5.75
Unif. Entropy	26.07	26.07	30.17	30.17

Table 5: Metrics related to the generative model/inference network for RNNG/URNNG. For the supervised RNNG we take the "inference network" to be the discriminative parser trained alongside the generative model (see §3.3). Recon. PPL is the reconstruction perplexity based on $\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$, and KL is the Kullback-Leibler divergence. Prior entropy is the entropy of the conditional prior $p_\theta(\mathbf{z} \mid \mathbf{x}_{<\mathbf{z}})$, and uniform entropy is the entropy of the uniform distribution over binary trees.

is highly nonzero. (See equation (1) for definitions of $\log p_\theta(\mathbf{x} \mid \mathbf{z})$ and $\log p_\theta(\mathbf{z} \mid \mathbf{x}_{<\mathbf{z}})$). This indicates that the latent space is being used in a meaningful way and that there is no posterior collapse (Bowman et al., 2016). As expected, the entropy of the variational posterior is much lower than the entropy of the conditional prior, but there is still some uncertainty in the posterior.

4.4 Syntactic Evaluation

We perform a syntactic evaluation of the different models based on the setup from Marvin and Linzen (2018): the model is given two minimally different sentences, one grammatical and one ungrammatical, and must identify the grammatical sentence by assigning it higher probability.[21] Table 6 shows the accuracy results. Overall the supervised RNNG significantly outperforms the other models, indicating opportunities for further work in unsupervised modeling. While the URNNG does slightly outperform an RNNLM, the distribution of errors made from both models are similar, and thus it is not clear whether the outperformance is simply due to better perplexity or learning different structural biases.

4.5 Limitations

There are several limitations to our approach. For one, the URNNG takes considerably more time/memory to train than a standard language model due to the $O(T^3)$ dynamic program in the inference network, multiple samples to obtain low-variance gradient estimators, and dynamic computation graphs that make efficient batching

---

[21] We modify the publicly available dataset from https://github.com/BeckyMarvin/LM_syneval to only keep sentence pairs that did not have any unknown words with respect to our vocabulary, resulting in 80K sentence pairs for evaluation.

	RNNLM	PRPN	RNNG	URNNG
PPL	93.2	96.7	88.7	90.6
Overall	62.5%	61.9%	69.3%	64.6%
Subj.	63.5%	63.7%	89.4%	67.2%
Obj. Rel.	62.6%	61.0%	67.6%	65.7%
Refl.	60.7%	68.8%	57.3%	60.5%
NPI	58.7%	39.5%	46.8%	55.0%

Table 6: Syntactic evaluation based on the setup from Marvin and Linzen (2018). Subj. is subject-verb agreement in sentential complement, across prepositional phrase/subjective relative clause, and VP coordination; Obj. Rel. refers to subject-verb agreement in/across an objective relative clause; Refl. refers to reflexive pronoun agreement with antecedent; NPI is negative polarity items.

nontrivial. The model is sensitive to hyperparameters and required various optimization strategies (e.g. separate optimizers for the inference network and the generative model) to avoid posterior collapse. Finally, the URNNG also seemed to rely heavily on punctuation to identify constituents and we were unable to improve upon a right-branching baseline when training the URNNG on a version of PTB where punctuation is removed.[22]

5 Related Work

There has been much work on incorporating tree structures into deep models for syntax-aware language modeling, both for unconditional (Emami and Jelinek, 2005; Buys and Blunsom, 2015; Dyer et al., 2016) and conditional (Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017; Aharoni and Goldberg, 2017; Eriguchi et al., 2017; Wang et al., 2018; Gū et al., 2018) cases. These approaches generally rely on annotated parse trees during training and maximizes the joint likelihood of sentence-tree pairs. Prior work on combining language modeling and unsupervised tree learning typically embed soft, tree-like structures as hidden layers of a deep network (Cho et al., 2014; Chung et al., 2017; Shen et al., 2018, 2019). In contrast, Buys and Blunsom (2018) make Markov assumptions and perform exact marginalization over latent dependency trees. Our work is also related to the recent line of work on learning latent trees as part of a deep model through supervision on other tasks, typically via differentiable structured hidden layers (Kim et al., 2017; Bradbury and Socher, 2017; Liu and Lapata, 2018; Tran and Bisk, 2018; Peng et al., 2018; Niculae et al., 2018; Liu et al., 2018), policy gradient-based approaches (Yogatama et al., 2017; Williams et al., 2018; Havrylov et al., 2019), or differentiable relaxations (Choi et al., 2018; Maillard and Clark, 2018).

The variational approximation uses amortized inference (Kingma and Welling, 2014; Mnih and Gregor, 2014; Rezende et al., 2014), in which an inference network is used to obtain the variational posterior for each observed $\mathbf{x}$. Since our inference network is structured (i.e., a CRF), it is also related to CRF autoencoders (Ammar et al., 2014) and structured VAEs (Johnson et al., 2016; Krishnan et al., 2017), which have been used previously for unsupervised (Cai et al., 2017; Drozdov et al., 2019; Li et al., 2019) and semi-supervised (Yin et al., 2018; Corro and Titov, 2019) parsing.

6 Conclusion

It is an open question as to whether explicit modeling of syntax significantly helps neural models. Strubell et al. (2018) find that supervising intermediate attention layers with syntactic heads improves semantic role labeling, while Shi et al. (2018) observe that for text classification, syntactic trees only have marginal impact. Our work suggests that at least for language modeling, incorporating syntax either via explicit supervision or as latent variables does provide useful inductive biases and improves performance.

Finally, in modeling child language acquisition, the complex interaction of the parser and the grammatical knowledge being acquired is the object of much investigation (Trueswell and Gleitman, 2007); our work shows that apparently grammatical constraints can emerge from the interaction of a constrained parser and a more general grammar learner, which is an intriguing but underexplored hypothesis for explaining human linguistic biases.

Acknowledgments

We thank the members of the DeepMind language team for helpful feedback. YK is supported by a Google Fellowship. AR is supported by NSF Career 1845664.

---

[22] Many prior works that induce trees directly from words often employ additional heuristics based on punctuation (Seginer, 2007; Ponvert et al., 2011; Spitkovsky et al., 2013; Parikh et al., 2014), as punctuation (e.g. comma) is usually a reliable signal for start/end of constituent spans. The URNNG still has to *learn* to rely on punctuation, similar to recent works such as depth-bounded PCFGs (Jin et al., 2018) and DIORA (Drozdov et al., 2019). In contrast, PRPN (Shen et al., 2018) and Ordered Neurons (Shen et al., 2019) induce trees by directly training on corpus without punctuation. We also reiterate that punctuation is used during training but ignored during evaluation (except in Table 4).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards String-to-Tree Neural Machine Translation. In *Proceedings of ACL*.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured Decoding with Doubly-Recurrent Neural Networks. In *Proceedings of ICLR*.

Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional Random Field Autoencoders for Unsupervised Structured Prediction. In *Proceedings of NIPS*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. In *Proceedings of NIPS*.

James K. Baker. 1979. Trainable Grammars for Speech Recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*.

Rens Bod. 2006. An All-Subtrees Approach to Unsupervised Parsing. In *Proceedings of ACL*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyal, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proceedings of CoNLL*.

James Bradbury and Richard Socher. 2017. Towards Neural Machine Translation with Latent Tree Attention. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*.

Jan Buys and Phil Blunsom. 2015. Generative Incremental Dependency Parsing with Neural Networks. In *Proceedings of ACL*.

Jan Buys and Phil Blunsom. 2018. Neural Syntactic Generative Models with Exact Marginalization. In *Proceedings of NAACL*.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. CRF Autoencoder for Unsupervised Dependency Parsing. In *Proceedings of EMNLP*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *arXiv:1312.3005*.

Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to Compose Task-Specific Tree Structures. In *Proceedings of AAAI*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. In *Proceedings of ICLR*.

Caio Corro and Ivan Titov. 2019. Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder. In *Proceedings of ICLR*.

Chris Cremer, Xuechen Li, and David Duvenaud. 2018. Inference Suboptimality in Variational Autoencoders. In *Proceedings of ICML*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. 2018. Latent Alignment and Variational Attention. In *Proceedings of NIPS*.

Andrew Drozdov, Patrick Verga, Mohit Yadev, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Auto-Encoders. In *Proceedings of NAACL*.

Greg Durrett and Dan Klein. 2015. Neural CRF Parsing. In *Proceedings of ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent Neural Network Grammars. In *Proceedings of NAACL*.

Ahmad Emami and Frederick Jelinek. 2005. A Neural Syntactic Language Model. *Machine Learning*, 60:195–227.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of ACL*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, Feature-based, Conditional Random Field Parsing. In *Proceedings of ACL*.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines. In *Proceedings of EMNLP*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*, 11:2001–2049.

Peter Glynn. 1987. Likelihood Ratio Gradient Estimation: An Overview. In *Proceedings of Winter Simulation Conference*.

Joshua Goodman. 1998. Parsing Inside-Out. *PhD thesis, Harvard University*.

Jetic Gū, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing. In *Proceedings of EMNLP*.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R. Brennan. 2018. Finding Syntax in Human Encephalography with Beam Search. In *Proceedings of ACL*.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative Learning of Disjoint Syntax and Semantics. In *Proceedings of NAACL*.

Phu Mon Htut, Kyunghyun Cho, and Samuel R. Bowman. 2018. Grammar Induction with Neural Language Models: An Unusual Replication. In *Proceedings of EMNLP*.

Rebecca Hwa. 1999. Supervised Grammar Induction Using Training Data with Limited Constituent Information. In *Proceedings of ACL*.

Rebecca Hwa. 2000. Sample Selection for Statistical Grammar Induction. In *Proceedings of EMNLP*.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised Grammar Induction with Depth-bounded PCFG. In *Proceedings of TACL*.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian Inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL*.

Matthew Johnson, David K. Duvenaud, Alex Wiltschko, Ryan P. Adams, and Sandeep R. Datta. 2016. Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference. In *Proceedings of NIPS*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2018. Unsupervised Word Discovery with Segmental Neural Language Models. *arXiv:1811.09353*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured Attention Networks. In *Proceedings of ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of ACL*.

Dan Klein and Christopher Manning. 2002. A Generative Constituent-Context Model for Improved Grammar Induction. In *Proceedings of ACL*.

Rahul G. Krishnan, Uri Shalit, and David Sontag. 2017. Structured Inference Networks for Nonlinear State Space Models. In *Proceedings of AAAI*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In *Proceedings of EACL*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better. In *Proceedings of ACL*.

Bowen Li, Jianpeng Cheng, Yang Liu, and Frank Keller. 2019. Dependency Grammar Induction with a Neural Variational Transition-based Parser. In *Proceedings of AAAI*.

Zhifei Li and Jason Eisner. 2009. First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of EMNLP*.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured Alignment Networks for Matching Sentences. In *Proceedings of EMNLP*.

Yang Liu and Mirella Lapata. 2018. Learning Structured Text Representations. In *Proceedings of TACL*.

Jean Maillard and Stephen Clark. 2018. Latent Tree Learning with Differentiable Parsers: Shift-Reduce Parsing and Chart Parsing. In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted Syntactic Evaluation of Language Models. In *Proceedings of EMNLP*.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of ACL*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. In *Proceedings of ICLR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *Proceedings of ICLR*.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proceedings of INTERSPEECH*.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of ICML*.

Andriy Mnih and Danilo J. Rezende. 2016. Variational Inference for Monte Carlo Objectives. In *Proceedings of ICML*.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of EMNLP*.

Vlad Niculae, André F. T. Martins, and Claire Cardie. 2018. Towards Dynamic Computation Graphs via Sparse Latent Structure. In *Proceedings of EMNLP*.

Ankur P. Parikh, Shay B. Cohen, and Eric P. Xing. 2014. Spectral Unsupervised Parsing with Additive Tree Metrics. In *Proceedings of ACL*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through Structured Argmax using a SPIGOT. In *Proceedings of ACL*.

Elis Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simpled Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Methods. In *Proceedings of ACL*.

Ofir Press and Lior Wolf. 2016. Using the Output Embedding to Improve Language Models. In *Proceedings of EACL*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of ACL*.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of ICML*.

Yoav Seginer. 2007. Fast Unsupervised Incremental Parsing. In *Proceedings of ACL*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural Language Modeling by Jointly Learning Syntax and Lexicon. In *Proceedings of ICLR*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks. In *Proceedings of ICLR*.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On Tree-based Neural Sentence Modeling. In *Proceedings of EMNLP*.

Noah A. Smith and Jason Eisner. 2004. Annealing Techniques for Unsupervised Statistical Language Learning. In *Proceedings of ACL*.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder Variational Autoencoders. In *Proceedings of NIPS*.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction. In *Proceedings of EMNLP*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of ACL*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of ACL*.

Ke Tran and Yonatan Bisk. 2018. Inducing Grammars with and for Neural Machine Translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*.

John C. Trueswell and Lila R. Gleitman. 2007. Learning to Parse and its Implications for Language Acquisition. *The Oxford Handbook of Psycholinguistics*.

Wenhui Wang and Baobao Chang. 2016. Graph-based Dependency Parsing with Bidirectional LSTM. In *Proceedings of ACL*.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A Tree-based Decoder for Neural Machine Translation. In *Proceedings of EMNLP*.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do Latent Tree Learning Models Identify Meaningful Structure in Sentences? In *Proceedings of TACL*.

Ronald J. Williams. 1992. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2018. Learning Neural Templates for Text Generation. In *Proceedings of EMNLP*.

Naiwen Xue, Fei Xia, Fu dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11:207–238.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *Proceedings of ICLR*.

Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of ACL*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing. In *Proceedings of ACL*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Sentences with Reinforcement Learning. In *Proceedings of ICLR*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long Short-Term Memory Over Tree Structures. In *Proceedings of ICML*.

Cooperative Learning of Disjoint Syntax and Semantics

Serhii Havrylov [*]
ILCC, University of Edinburgh / Edinburgh, UK
s.havrylov@ed.ac.uk

Germán Kruszewski & Armand Joulin
Facebook AI Research
{germank,ajoulin}@fb.com

Abstract

There has been considerable attention devoted to models that learn to jointly infer an expression's syntactic structure and its semantics. Yet, Nangia and Bowman (2018) has recently shown that the current best systems fail to learn the correct parsing strategy on mathematical expressions generated from a simple context-free grammar. In this work, we present a recursive model inspired by Choi et al. (2018) that reaches near perfect accuracy on this task. Our model is composed of two separated modules for syntax and semantics. They are cooperatively trained with standard continuous and discrete optimisation schemes. Our model does not require any linguistic structure for supervision, and its recursive nature allows for out-of-domain generalisation. Additionally, our approach performs competitively on several natural language tasks, such as Natural Language Inference and Sentiment Analysis.

1 Introduction

Standard linguistic theories propose that natural language is structured as nested constituents organised in the form of a tree (Partee et al., 1990). However, most popular models, such as the Long Sort-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997), process text without imposing a grammatical structure. To bridge this gap between theory and practice models that process linguistic expressions in a tree-structured manner have been considered in recent work (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015; Bowman et al., 2016). These tree-based models explicitly require access to the syntactic structure for the text, which is not entirely satisfactory.

Indeed, parse tree level supervision requires a significant amount of annotations from expert linguists. These trees have been annotated with different goals in mind than the tasks we are using them for. Such discrepancy may result in a deterioration of the performance of models relying on them. Recently, several attempts were made to learn these models without explicit supervision for the parser (Yogatama et al., 2016; Maillard et al., 2017; Choi et al., 2018). However, Williams et al. (2018a) has recently shown that the structures learned by these models cannot be ascribed to discovering meaningful syntactic structure. These models even fail to learn the simple context-free grammar of nested mathematical operations (Nangia and Bowman, 2018).

In this work, we present an extension of Choi et al. (2018), that successfully learns these simple grammars while preserving competitive performance on several standard linguistic tasks. Contrary to previous work, our model makes a clear distinction between the parser and the compositional function. These two modules are trained with different algorithms, cooperating to build a semantic representation that optimises the objective function. The parser's goal is to generate a tree structure for the sentence. The compositional function follows this structure to produce the sentence representation. Our model contains a continuous component, the compositional function, and a discrete one, the parser. The whole system is trained end-to-end with a mix of reinforcement learning and gradient descent. Drozdov and Bowman (2017) has noticed the difficulty of mixing these two optimisation schemes without one dominating the other. This typically leads to the "coadaptation problem" where the parser simply follows the compositional function and fails to produce meaningful syntactic structures. In this work, we show that this pitfall can be avoided by synchronising the learning paces of the two optimisation schemes. This is achieved by com-

---

[*] Work done while the author was an intern at Facebook AI Research.

Proceedings of NAACL-HLT 2019, pages 1118–1128
Minneapolis, Minnesota, June 2 - June 7, 2019.

bining several recent advances in reinforcement learning. First, we use input-dependent control variates to reduce the variance of our gradient estimates (Ross, 1997). Then, we apply multiple gradient steps to the parser's policy while controlling for its learning pace using the Proximal Policy Optimization (PPO) of Schulman et al. (2017). The code for our model is publicly available[1].

2 Preliminaries

In this section, we present existing works on Recursive Neural Networks and their training in the absence of supervision on the syntactic structures.

2.1 Recursive Neural Networks

A Recursive Neural Network (RvNN) has its architecture defined by a directed acyclic graph (DAG) given alongside with an input sequence (Goller and Kuchler, 1996). RvNNs are commonly used in NLP to generate sentence representation that leverages available syntactic information, such as a constituency or a dependency parse trees (Socher et al., 2011).

Given an input sequence and its associated DAG, a RvNN processes the sequence by applying a transformation to the representations of the tokens lying on the lowest levels of the DAG. This transformation, or compositional function, merges these representations into representations for the nodes on the next level of the DAG. This process is repeated recursively along the graph structure until the top-level nodes are reached. In this work, we assume that the compositional function is the same for every node in the graph.

Tree-LSTM. We focus on a specific type of RvNNs, the tree-based long short-term memory network (Tree-LSTM) of Tai et al. (2015) and Zhu et al. (2015). Its compositional function generalizes the LSTM cell of Hochreiter and Schmidhuber (1997) to tree-structured topologies, i.e.,

$$\begin{bmatrix}\mathbf{z}\\\mathbf{i}\\\mathbf{f}_l\\\mathbf{f}_r\\\mathbf{o}\end{bmatrix} = \begin{bmatrix}\mathtt{tanh}\\\sigma\\\sigma\\\sigma\\\sigma\end{bmatrix}\left(\mathbf{R}\begin{bmatrix}\mathbf{h}_l\\\mathbf{h}_r\end{bmatrix}+\mathbf{b}\right),$$

$$\mathbf{c}_p = \mathbf{z}\odot\mathbf{i}+\mathbf{c}_l\odot\mathbf{f}_l+\mathbf{c}_r\odot\mathbf{f}_r,$$

$$\mathbf{h}_p = \mathtt{tanh}(\mathbf{c}_p)\odot\mathbf{o},$$

where σ and `tanh` are the sigmoid and hyperbolic tangent functions. Tree-LSTM cell is differentiable with respect to its recursion matrix $\mathbf{R}$, bias $\mathbf{b}$ and its input. The gradients of a Tree-LSTM can thus be computed with backpropagation through structure (BPTS) (Goller and Kuchler, 1996).

2.2 Learning with RvNNs

A tree-based RvNN is a function f_θ parameterized by a d dimensional vector θ that predicts an output y given an input x and a tree t. Given a dataset $\mathcal{D}$ of N triplets (x, t, y), the parameters of the RvNN are learned with the following minimisation problem:

$$\min_{\theta\in\mathbb{R}^d}\frac{1}{N}\sum_{(x,t,y)\in\mathcal{D}}\ell(f_\theta(x,t),y), \qquad (1)$$

where ℓ is a logistic regression function. These models need an externally provided parsing tree for each input sentence during both training and evaluation. Alternatives, such as the shift-reduce-based SPINN model of Bowman et al. (2016), learn an internal parser from the given trees. While these solutions do not need external trees during evaluation, they still require tree level annotations for training. More recent work has focused on learning a latent parser with no direct supervision.

2.3 Latent tree models

Latent tree models aim at jointly learning the compositional function f_θ and a parser without supervision on the syntactic structures (Yogatama et al., 2016; Maillard et al., 2017; Choi et al., 2018). The latent parser is defined as a parametric probability distribution over trees conditioned on the input sequence. The parameters of this tree distribution $p_\phi(.|x)$ are represented by a vector ϕ. Given a dataset $\mathcal{D}$ of pairs of input sequences x and outputs y, the parameters θ and ϕ are jointly learned by minimising the following objective function:

$$\min_{\theta,\phi}\mathcal{L}(\theta,\phi) = \frac{1}{N}\sum_{(x,y)}\ell(\mathbb{E}_\phi[f_\theta(x,t)],y), \qquad (2)$$

where $\mathbb{E}_\phi$ is the expectation with respect to the $p_\phi(.|x)$ distribution. Directly minimising this objective function is often difficult due to expensive marginalisation of the unobserved trees. Hence, when ℓ is a convex function (e.g. cross entropy of an exponential family) usually an upper bound of Eq. (2) can be derived by applying

---

[1] `https://github.com/facebookresearch/latent-treelstm`

Jensen's inequality:

$$\hat{\mathcal{L}}(\theta, \phi) = \frac{1}{N} \sum_{(x,y)} \mathbb{E}_\phi[\ell(f_\theta(x, t), y)]. \quad (3)$$

Learning a distribution over a set of discrete items involves a discrete optimisation scheme. For example, the RL-SPINN model of Yogatama et al. (2016) uses a mix of gradient descent for θ and REINFORCE for ϕ (Williams et al., 2018a). Drozdov and Bowman (2017) has recently observed that this optimisation strategy tends to produce poor parsers, e.g., parsers that only generate left-branching trees. The effect, called the coadaptation issue, is caused by both bias in the parsing strategy and a difference in convergence paces of continuous and discrete optimisers. Typically, the parameters θ are learned more rapidly than ϕ. This limits the exploration of the search space to parsing strategies similar to those found at the beginning of the training.

2.3.1 Gumbel Tree-LSTM

In their Gumbel Tree-LSTM model, Choi et al. (2018) propose an alternative parsing strategy to avoid the coadaptation issue. Their parser incrementally merges a pair of consecutive constituents until a single one remains. This strategy reduces the bias towards certain tree configurations observed with RL-SPINN.

Each word i of the input sequence is represented by an embedding vector. A leaf transformation maps this vector to pair of vectors $\mathbf{r}_i^0{=}(\mathbf{h}_i^0, \mathbf{c}_i^0)$. We considered three types of leaf transformations: affine transformation, LSTM and bidirectional LSTM. The resulting representations form the initial states of the Tree-LSTM. In the absence of supervision, the tree is built in a bottom-up fashion by recursively merging consecutive constituents $(i, i+1)$ based on merge-candidate scores. On each level k of the bottom-up derivation, the merge-candidate score of the pair $(i, i{+}1)$ is computed as follow:

$$s_k(i) = \langle \mathbf{q}, \texttt{Tree-LSTM}(\mathbf{r}_i^k, \mathbf{r}_{i+1}^k) \rangle,$$

where $\mathbf{q}$ is a trainable query vector and $\mathbf{r}_i^k$ is the constituent representation at position i after k mergings. We merge a pair (i^*, i^*+1) sampled from the Categorical distribution built on the merge-candidate scores. The representations of the constituents are then updated as follow:

$$\mathbf{r}_i^{k+1} = \begin{cases} \mathbf{r}_i^k, & i < i^*, \\ \texttt{Tree-LSTM}(\mathbf{r}_i^k, \mathbf{r}_{i+1}^k) & i = i^*, \\ \mathbf{r}_{i+1}^k & i > i^*. \end{cases}$$

This procedure is repeated until one constituent remains. Its hidden state is the input sentence representation. This procedure is non-differentiable. Choi et al. (2018) use an approximation based on the Gumbel-Softmax distribution (Maddison et al., 2016; Jang et al., 2016) and the reparametrization trick (Kingma and Welling, 2013).

This relaxation makes the problem differentiable at the cost of a bias in the gradient estimates (Jang et al., 2016). This difference between the real objective function and their approximation could explain why their method cannot recover simple context-free grammars (Nangia and Bowman, 2018). We investigate this question by proposing an alternative optimisation scheme that directly aims for the correct objective function.

3 Our model

We consider the problem defined in Eq. (3) to jointly learn a composition function and an internal parser. Our model is composed of the parser of Choi et al. (2018) and the Tree-LSTM for the composition function. As suggested in past work (Mnih et al., 2016; Schulman et al., 2017), we added an entropy $\mathcal{H}$ over the tree distribution to the objective function:

$$\min_{\theta,\,\phi} \hat{\mathcal{L}}(\theta, \phi) - \lambda \sum_x \mathcal{H}(t \mid x), \quad (4)$$

where $\lambda > 0$. This regulariser improves exploration by preventing early convergence to a suboptimal deterministic parsing strategy. The new objective function is differentiable with respect to θ, but not ϕ, the parameters of the parser. Learning θ follows the same procedure with BPTS as if the tree would be externally given.

In the rest of this section, we discuss the optimization of the parser and a cooperative training strategy to reduce the coadaptation issue.

3.1 Unbiased gradient estimation

We cast the training of the parser as a reinforcement learning problem. The parser is an agent whose reward function is the negative of the loss function defined in Eq. (3). Its action space is the space of binary trees. The agent's policy is a probability distribution over binary trees that decomposes as a sequence of K merging actions:

$$p_\phi(t|x) = \prod_{k=0}^{K} \pi_\phi(a_k^i|\mathbf{r}^k), \tag{5}$$

where $\mathbf{r}^k = (\mathbf{r}_0^k, \dots, \mathbf{r}_{K-k}^k)$. The loss function is optimised with respect to ϕ with REINFORCE (Williams, 1992). REINFORCE requires a considerable number of random samples to obtain a gradient estimate with a reasonable level of variance. This number is positively correlated with the size of the search space, which is exponentially large in the case of binary trees. We consider several extensions of REINFORCE to circumvent this problem.

Variance reduction. An alternative solution to increasing the number of samples is the control variates method (Ross, 1997). It takes advantage of random variables with known expected values and positive correlation with the quantity whose expectation is tried to be estimated. Given an input-output pair (x, y) and tree t sampled from $p_\phi(t|x)$, let's define the random variable G as:

$$G(t) = \ell(f_\theta(x,t), y)\frac{\partial \log p_\phi(t|x)}{\partial \phi}. \tag{6}$$

According to REINFORCE, calculating the gradient with respect to ϕ for the pair (x, y) is then equivalent to determining the unknown mean of the random variable $G(t)$[2]. Let's assume there is a control variate, i.e., a random variable $b(t)$ that positively correlates with G and has known expected value with respect to $p_\phi(.|x)$. Given N samples of the $G(t)$ and the control variate $b(t)$, the new gradient estimator is:

$$G_{\text{CV}} = \mathbb{E}_{p_\phi(t|x)}[b(t)] + \frac{1}{N}\left[\sum_{i=1}^{N}(G(t_i) - b(t_i))\right].$$

A popular control variate, or baseline, used in REINFORCE is the moving average of recent rewards multiplied by the score function (Ross, 1997):

$$b(t) = c\nabla_\phi \log p_\phi(t|x).$$

It has a zero mean under the $p_\phi(.|x)$ distribution and it positively correlates with $G(t)$.

Surrogate loss. REINFORCE often is implemented via a surrogate loss defined as follow:

$$\hat{\mathbb{E}}_t\left[r_\phi(t)\ell(f_\theta(x,t), y)\right], \tag{7}$$

where $\hat{\mathbb{E}}_t$ is the empirical average over a finite batch of samples and $r_\phi(t) = \frac{p_\phi(t|x)}{p_{\phi_{\text{old}}}(t|x)}$ is the probability ratio with ϕ_{old} standing for the parameters before the update.

Input-dependent baseline. The moving average baseline cannot detect changes in rewards caused by structural differences in the inputs. In our case, a long arithmetic expression is much harder to parse than a short one, systematically leading to their lower rewards. This structural differences in the rewards aggravate the credit assignment problem by encouraging REINFORCE to discard actions sampled for longer sequences even though there might be some subsequences of actions that produce correct parsing subtrees.

A solution is to make the baseline input-dependent. In particular, we use the self-critical training (SCT) baseline of Rennie et al. (2017), defined as:

$$b(t, x) = c_{\theta,\phi}(x)\nabla_\phi \log p_\phi(t \mid x),$$

where $c_{\theta,\phi}$ is the reward obtained with the policy used at test time, i.e., $\hat{t} = \arg\max p_\phi(t|x)$. This control variate has a zero mean under the $p_\phi(t|x)$ distribution and correlates positively with the gradients. Computing the $\arg\max$ of a policy among all possible binary trees has exponential complexity. We replace it with a simpler greedy decoding, i.e, a tree t is selected by following a sequence of greedy actions $\hat{a}_k$:

$$\hat{a}_k = \arg\max \pi_\phi(a_k \mid \hat{\mathbf{r}}^k).$$

This approximation is very efficient and computing the baseline requires only one additional forward pass.

Gradient normalization. We empirically observe significant fluctuations in the gradient norms. This creates instability that can not be reduced by additive terms, such as the input-dependent baselines. A solution is to divide the gradients by a coarse approximation of their norm, e.g., a running estimate of the reward standard deviation (Mnih and Gregor, 2014). This trick ensures that the rewards remain approximately in the unit ball, making the learning process less sensitive to steep changes in the loss.

---

[2] Note that while we are computing the gradients using ℓ, we could also directly optimise the parser with respect to downstream accuracy.

3.2 Synchronizing syntax and semantics learning with PPO

The gradients of the loss function from the Eq. (4) are calculated using two different schemes, BPST for the composition function parameters θ and REINFORCE for the parser parameters ϕ. Then, both are updated with SGD. The estimate of the gradient with respect to ϕ has higher variance compared to the estimate with respect to θ. Hence, using the same learning rate schedule does not necessarily correspond to the same real pace of learning. It is ϕ parameters that are harder to optimise, so to improve training stability and convergence it is reasonable to aim for such updates that does not change the policy too much or too little. A simple yet effective solution is the Proximal Policy Optimization (PPO) of Schulman et al. (2017). It considers the next surrogate loss:

$$\hat{\mathbb{E}}_t \left[\max \left\{ r_\phi(t) \ell \left(f_\theta(x,t), y \right), r^c_\phi(t) \ell \left(f_\theta(x,t), y \right) \right\} \right],$$

Where $r^c_\phi(t) = \texttt{clip}\left(r_\phi(t), 1-\epsilon, 1+\epsilon\right)$ and ϵ is a real number in $(0; 0.5]$. The first argument of the $\max$ is the surrogate loss for REINFORCE. The clipped ratio in the second argument disincentivises the optimiser from performing updates resulting in large tree probability changes. With this, the policy parameters can be optimised with repeated K steps of SGD to ensure a similar "pace" of learning between the parser and the compositional function.

4 Related work

Besides the works mentioned in Sec. 2 and Sec. 3, there is a vast literature on learning latent parsers. Early connectionist work in inferring context-free grammars proposed stack-augmented models and relied on explicit supervision on the strings that belonged to the target language and those that did not (Giles et al., 1989; Sun, 1990; Das et al., 1992; Mozer and Das, 1992). More recently, new stack-augmented models were shown to learn latent grammars from positive evidence alone (Joulin and Mikolov, 2015). In parallel to these, other statistical approaches were proposed to automatically induce grammars from unparsed text (Sampson, 1986; Magerman and Marcus, 1990; Carroll and Charniak, 1992; Brill, 1993; Klein and Manning, 2002). Our work departs from these approaches in that we aim at learning a latent grammar in the context of performing some given task.

Socher et al. (2011) uses a surrogate auto-encoder objective to search for a constituency structure, merging nodes greedily based on the reconstruction loss. Maillard et al. (2017) defines a relaxation of a CYK-like chart parser that is trained for a particular task. A similar idea is introduced in Le and Zuidema (2015) where an automatic parser prunes the chart to reduce the overall complexity of the algorithm. Another strategy, similar in nature, has been recently proposed by Corro and Titov (2018), where Gumbel noise is used with differentiable dynamic programming to generate dependency trees. In contrast, Yogatama et al. (2016) learns a Shift-Reduce parser using reinforcement learning. Maillard and Clark (2018) further proposes a beam search strategy to overcome learning trivial trees. On a different vein, Vlad Niculae (2018) proposes a quadratic penalty term over the posterior distribution of non-projective dependency trees to enforce sparsity of the relaxation. Finally, there is a large body of work in Reinforcement Learning that aims at discovering how to combine elementary modules to solve complex tasks (Singh, 1992; Chang et al., 2018; Sahni et al., 2017). Due to the limited space, we will not discuss them in further details.

5 Experiments

We conducted experiments on three different tasks: evaluating mathematical expressions on the ListOps dataset (Nangia and Bowman, 2018), sentiment analysis on the SST dataset (Socher et al., 2013) and natural language inference task on the SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018b) datasets.

Technical details. For ListOps, we follow the experimental protocol of Nangia and Bowman (2018), i.e., a 128 dimensional model and a ten-way softmax classifier. However, we replace their multi-layer perceptron (MLP) by a linear classifier. The validation set is composed of 1k examples randomly selected from the training set. For SST and NLI, we follow the setup of Choi et al. (2018): we initialise the word vectors with GloVe300D (Pennington et al., 2014) and train an MLP classifier on the sentence representations. The hyperparameters are selected on the validation set using 5 random seeds for each configuration. Our hyperparameters are the learning rate, weight decay, the regularisation parameter λ, the leaf transformations, variance reduction hyperpa-

	No baseline		Moving average		Self critical	
	No PPO	PPO	No PPO	PPO	No PPO	PPO
min	61.7	61.4	61.7	59.4	63.7	**98.2**
max	70.1	76.6	74.3	96.0	64.1	**99.6**
mean ± std	66.2 ±3.2	66.5 ±5.9	65.5 ± 4.7	67.5 ±14.3	64.0 ±0.1	**99.2 ±0.5**

Table 1: Accuracy on ListOps test set for our model with three different baselines, with and without PPO. We use $K = 15$ for PPO.

Model	Accuracy
LSTM*	71.5±1.5
RL-SPINN*	60.7±2.6
Gumbel Tree-LSTM*	57.6±2.9
Ours	**99.2±0.5**

Table 2: Accuracy on the ListOps dataset. All models have 128 dimensions. Results for models with * are taken from Nangia and Bowman (2018).

rameters and the number of updates K in PPO. We use an adadelta optimizer (Zeiler, 2012).

5.1 ListOps

The ListOps dataset probes the syntax learning ability of latent tree models (Nangia and Bowman, 2018). It is designed to have a single correct parsing strategy that a model must learn in order to succeed. It is composed of prefix arithmetic expressions and the goal is to predict the numerical output associated with the evaluation of the expression. The sequences are made of integers in $[0, 9]$ and 4 operations: `MIN`, `MAX`, `MED` and `SUM_MOD`. The output is an integer in the range $[0, 9]$. For example, the expression `[MIN 2 [MAX 0 1] [MIN 6 3 ] 5 ]` is mapped to the output `1`. The ListOps task is thus a sequence classification problem with 10 classes. There are 90k training examples and 10k test examples. It is worth mentioning that the underlying semantic of operations and symbols is not provided. In other words, a model has to infer from examples that `[MIN 0 1] = 0`.

As shown in Table 2, the current leading latent tree models are unable to learn the correct parsing strategy on ListOps (Nangia and Bowman, 2018). They even achieve performance worse than purely sequential recurrent networks. On the other hand, our model achieves near perfect accuracy on this task, suggesting that our model is able to discover the correct parsing strategy. Our model differs in several ways from the Gumbel Tree-LSTM of Choi et al. (2018) that could explain this gap in performance. In the rest of this section, we perform an ablation study on our model to understand the importance of each of these differences.

Impact of the baseline and PPO. We report the impact of our design choices on the performance in Table 1. Our model without baseline nor PPO is vanilla REINFORCE. The baselines only improve performance when PPO is used. Furthermore, these ablated models without PPO perform on-par with the RL-SPINN model (see Table 2). This confirms our expectations for models that fail to synchronise syntax and semantics learning.

Interestingly, using PPO has a positive impact on both baselines, but accuracy remains low with the moving average baseline. The reduction of variance induced by the SCT baseline leads to a near-perfect recovery of the good parsing strategy in all five experiments. This shows the importance of this baseline for the stability of our approach.

Sensitivity to hyperparameters. Our model is relatively robust to hyperparameters changes when we use the SCT baseline and PPO. For example, changing the leaf transformation or dimensionality of the model has a minor impact on performance. However, we have observed that the choice of the optimiser has a significant impact. For example, the average performance drops to 73.0% if we replace Adadelta by Adam (Kingma and Ba, 2014). Yet, the maximum value out of 5 runs remains relatively high, 99.0%.

Untied parameters. As opposed to previous work, the parameters of the parser and the composition function are not tied in our model. Without this separation between syntax and semantics, it would be impossible to update one module with-

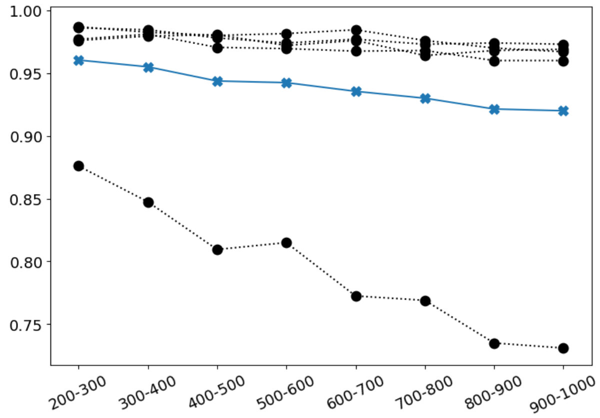


Figure 1: Blue crosses depict an average accuracy of five models on the test examples that have lengths within certain range. Black circles illustrate individual models.

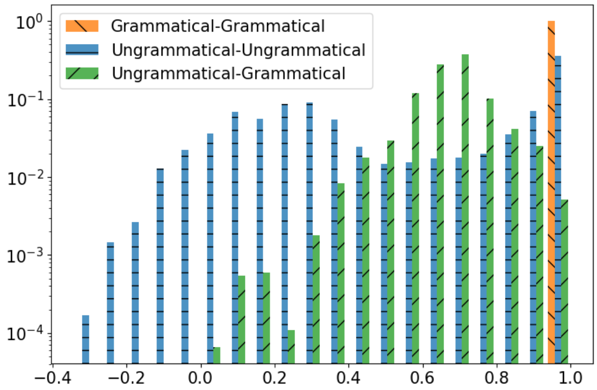


Figure 2: The distributions of cosine similarity for elements from the different sets of mathematical expressions. A logarithmic scale is used for y-axis.

out changing the other. The gradient direction is then dominated by the low variance signal from the semantic component, making it hard to learn the parser. We confirmed experimentally that our model with tied parameters fails to find the correct parser and its accuracy drops to 64.7%.

Extrapolation and Grammaticality. Recursive models have the potential to generalise to any sequence length. Our model was trained with sequences of length up to 130 tokens. We test the ability of the model to generalise to longer sequences by generating additional expressions of lengths 200 to 1000. As shown in Fig.1, our model has a little loss in accuracy as the length increases to ten times the maximum length seen during training.

On the other hand, we notice that final representations produced by the parser are very similar to each other. Indeed, the cosine similarity between these vectors for the test set has a mean value of 0.998 with a standard deviation of 0.002. There are two possible explanations for this observation: either our model assigns similar representations to valid expressions, or it produces a trivial uninformative representation regardless of the expression. To verify which explanation is correct, we generate ungrammatical expressions by removing either one operation token or one closing bracket symbol for each sequence in the test set. As shown in Figure 2, in contrast to grammatical expressions, ungrammatical ones tend to be very different from each other: "Happy families are all alike; every unhappy family is unhappy in its own way." The only exception, marked by a mode near 1, come from ungrammatical expressions that represent incomplete expressions because of missing a closing bracket at the end. This kind of sequences were seen by the parser during training and they indeed have to be represented by the same vector. These observations show that our model does not produce a trivial representation, but identifies the rules and constraints of the grammar. Moreover, vectors for grammatical sequences are so different from vectors for ungrammatical ones that you can tell them apart with 99.99% accuracy by simply measuring their cosine similarity to a randomly chosen grammatical vector from the training set. Interestingly, we have not observed a similar signal from the vectors generated by the composition function. Even learning a naive classifier between grammatical and ungrammatical expressions on top of these representations achieves an accuracy of only 75%. This suggests that most of the syntactic information is captured by the parser, not the composition function.

5.2 Natural Language Inference

We next evaluate our model on natural language inference using the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) and MultiNLI (Williams et al., 2018b) datasets. Natural language inference consists in predicting the relationship between two sentences which can be either entailment, contradiction, or neutral. The task can be formulated as a three-way classification problem. The results are shown in Tables 3 and 4. When training the model on MultiNLI dataset we augment the training data with the SNLI data and use *matched* versions of the de-

Model	Dim.	Acc.
Yogatama et al. (2016)	100	80.5
Maillard et al. (2017)	100	81.6
Choi et al. (2018)	100	82.6
Ours	100	**84.3±0.3**
Bowman et al. (2016)	300	83.2
Munkhdalai and Yu (2017)	300	84.6
Choi et al. (2018)	300	**85.6**
Choi et al. (2018)†	300	83.7
Choi et al. (2018)*	300	84.9± 0.1
Ours	300	85.1±0.2
Chen et al. (2017)	600	85.5
Choi et al. (2018)	600	**86.0**
Ours	600	84.6±0.2

Table 3: Results on SNLI. *: publicly available code and hyperparameter optimization was used to obtain results. †: results are taken from Williams et al. (2018a)

Model	Dim.	Acc.
LSTM†	300	69.1
SPINN†	300	67.5
RL-SPINN†	300	67.4
Gumbel Tree-LSTM†	300	69.5
Ours	300	**70.7±0.3**

Table 4: Results on MultiNLI. †: results are taken from Williams et al. (2018a).

	SST-2	SST-5
Sequential sentence representation		
Radford et al. (2017)	**91.8**	52.9
McCann et al. (2017)	90.3	53.7
Peters et al. (2018)	-	**54.7**
RvNN based models with external tree		
Socher et al. (2013)	85.4	45.7
Tai et al. (2015)	88.0	51.0
Munkhdalai and Yu (2017)	89.3	53.1
Looks et al. (2017)	89.4	52.3
RvNN based models with latent tree		
Yogatama et al. (2016)	86.5	-
Choi et al. (2018)	90.7	53.7
Choi et al. (2018)*	90.3±0.5	51.6±0.8
Ours	90.2±0.2	51.5±0.4

Table 5: Accuracy results of models on the SST. All the numbers are from Choi et al. (2018) but * where we used their publicly available code and performed hyperparameter optimization.

velopment and test sets. Surprisingly, two out of four models for MultiNLI task collapsed to left-branching parsing strategies. This collapse can be explained by the absence of the entropy regularisation and the small number of PPO updates $K = 1$, which were determined to be optimal via hyperparameter optimisation. As with ListOps, using an Adadelta optimizer significantly improves the training of the model.

5.3 Sentiment Analysis

We evaluate our model on a sentiment classification task using the Stanford Sentiment Treebank (SST) of Socher et al. (2013). All sentences in SST are represented as binary parse trees, and each subtree of a parse tree is annotated with the corresponding sentiment score. There are two versions of the dataset, with either binary labels, “negative” or “positive”, (SST-2) or five labels, representing fine-grained sentiments (SST-5). As shown in Table 5, our results are in line with previous work, confirming the benefits of using latent syntactic parse trees instead of the predefined syntax.

We noticed that all models trained on NLI or sentiment analysis tasks have parsing policies with relatively high entropy. This indicates that the algorithm does not prefer any specific grammar. Indeed, generated trees are very similar to balanced ones. This result is in line with Shi et al. (2018) where they observe that binary balanced tree encoder gets the best results on most classification tasks.

We also compare with state-of-the-art sequence-based models. For the most part, these models are pre-trained on larger datasets and fine-tuned on these tasks. Nonetheless, they outperform recursive models by a significant margin. Performance on these datasets is more impacted by pre-training than by learning the syntax. It would be interesting to see if a similar pre-training would also improve the performance of recursive models with latent tree learning.

6 Conclusion

In this paper, we have introduced a novel model for learning latent tree parsers. Our approach relies on a separation between syntax and semantics. This allows dedicated optimisation schemes for each module. In particular, we found that it is important to have an unbiased estimator of the parser gradients and to allow multiple gradient steps with PPO. When tested on a CFG, our learned parser generalises to sequences of any length and distinguishes grammatical from ungrammatical expressions by forming meaningful representations for well-formed expressions. For natural language tasks, instead, the model prefers to fall back to trivial strategies, in line with what was previously observed by Shi et al. (2018). Additionally, our approach performs competitively on several real natural language tasks. In the future, we would like to explore further relaxation-based techniques for learning the parser, such as REBAR (Tucker et al., 2017) or ReLAX (Grathwohl et al., 2017). Finally, we plan to look into applying recursive approaches to language modelling as a pre-training step and measure if it has the same impact on downstream tasks as sequential models.

Acknowledgments

We would like to thank Alexander Koller, Ivan Titov, Wilker Aziz and anonymous reviewers for their helpful suggestions and comments.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings EMNLP 2015*, pages 632–642.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the ACL 2016, Volume 1: Long Papers*.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of ACL 93*, pages 259–265.

Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

Michael B. Chang, Abhishek Gupta, Sergey Levine, and Thomas L. Griffiths. 2018. Automatically composing representation transformations as a means for generalization. *CoRR*, abs/1807.04640.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of ACL 2017, Volume 1: Long Papers*, pages 1657–1668.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of AAAI 2018*.

Caio Corro and Ivan Titov. 2018. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *CoRR*, abs/1807.09875.

Sreerupa Das, C Lee Giles, and Guo-Zheng Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of CogSci 1992*, page 14.

Andrew Drozdov and Samuel Bowman. 2017. The coadaptation problem when learning how and what to compose. *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

C. Lee Giles, Guo-Zheng Sun, Hsing-Hen Chen, Yee-Chun Lee, and Dong Chen. 1989. Higher order recurrent networks and grammatical inference. In *Proceedings of NIPS 1989*, pages 380–387.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*, 1:347–352.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David K. Duvenaud. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *CoRR*, abs/1711.00123.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of NIPS 2015*, pages 190–198.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL 2002*, pages 128–135.

Phong Le and Willem H. Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of EMNLP 2015*, pages 1155–1164.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. *arXiv preprint arXiv:1702.02181*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712.

David M Magerman and Mitchell P Marcus. 1990. Parsing a natural language using mutual information statistics. In *AAAI*, volume 90, pages 984–989.

Jean Maillard and Stephen Clark. 2018. Latent tree learning with differentiable parsers: Shift-reduce parsing and chart parsing. *CoRR*, abs/1806.00840.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *CoRR*, abs/1705.09189.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS 2017*, pages 6294–6305.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of ICML 2016*, pages 1928–1937.

Michael Mozer and Sreerupa Das. 1992. A connectionist symbol manipulator that discovers the structure of context-free languages. In *Proceedings of NIPS 1992*, pages 863–870.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *Proceedings of EACL 2017*, volume 1, page 11. NIH Public Access.

Nikita Nangia and Samuel R. Bowman. 2018. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of NAACL-HLT 2018, Student Research Workshop*, pages 92–99.

Barbara BH Partee, Alice G ter Meulen, and Robert Wall. 1990. *Mathematical methods in linguistics*, volume 30. Springer Science & Business Media.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of CVPR 2017*, pages 1179–1195.

Sheldon M. Ross. 1997. *Simulation (2. ed.)*. Statistical modeling and decision science. Academic Press.

Himanshu Sahni, Saurabh Kumar, Farhan Tejani, and Charles L. Isbell. 2017. Learning to compose skills. *CoRR*, abs/1711.11289.

Geoffrey Sampson. 1986. A stochastic approach to parsing. In *Proceedings Coling 1986 Volume 1*, volume 1.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. *CoRR*, abs/1808.09644.

Satinder P. Singh. 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP 2011*, pages 151–161.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*, pages 1631–1642.

G Sun. 1990. Connectionist pushdownautomata that learn context-free gram-mars. In *Proc. IJCNN'90*, volume 1, pages 577–580.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL 2015, Volume 1: Long Papers*, pages 1556–1566.

George Tucker, Andriy Mnih, Chris J. Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. REBAR: low-variance, unbiased gradient estimates for discrete latent variable models. In *Proceedings of NIPS 2017*, pages 2624–2633.

Claire Cardie Vlad Niculae, André F. T. Martins. 2018. Towards dynamic computation graphs via sparse latent structure. *CoRR*, abs/1809.00653.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *CoRR*, abs/1611.09100.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of ICML 2015*, pages 1604–1612.

Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Autoencoders

Andrew Drozdov*, **Pat Verga***, **Mohit Yadav***,
Mohit Iyyer, **and Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
`{adrozdov, pat, ymohit, miyyer, mccallum}@cs.umass.edu`

Abstract

We introduce deep inside-outside recursive autoencoders (DIORA), a fully-unsupervised method for discovering syntax that simultaneously learns representations for constituents within the induced tree. Our approach predicts each word in an input sentence conditioned on the rest of the sentence and uses inside-outside dynamic programming to consider all possible binary trees over the sentence. At test time the CKY algorithm extracts the highest scoring parse. DIORA achieves a new state-of-the-art F1 in unsupervised binary constituency parsing (unlabeled) in two benchmark datasets, WSJ and MultiNLI.

1 Introduction

Syntactic parse trees are useful for downstream tasks such as relation extraction (Gamallo et al., 2012), semantic role labeling (Sutton and McCallum, 2005; He et al., 2018), machine translation (Aharoni and Goldberg, 2017; Eriguchi et al., 2017; Zaremoodi and Haffari, 2018), and text classification (Li and Roth, 2006; Tai et al., 2015). Traditionally, supervised parsers trained on datasets such as the Penn Treebank (Marcus et al., 1993) are used to obtain syntactic trees. However, the treebanks used to train these supervised parsers are typically small and restricted to the newswire domain. Unfortunately, models trained on newswire treebanks tend to perform considerably worse when applied to new types of data, and creating new domain specific treebanks with syntactic annotations is expensive and time-consuming.

Motivated by the desire to address the limitations of supervised parsing and by the success of large-scale unsupervised modeling such as ELMo and BERT (Peters et al., 2018a; Devlin et al., 2019), we propose a new deep learning method of unsupervised parser training that can extract both shallow parses (i.e., noun phrases or entities) and full syntactic trees from any domain or language automatically *without requiring any labeled training data*. In addition to producing parses, our model simultaneously builds representations for internal constituents that reflect syntactic and semantic regularities which can be leveraged by downstream tasks.

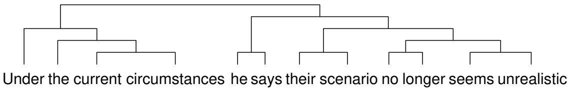


Figure 1: An unlabeled binary constituency parse from DIORA matching the ground truth.

Our model builds on existing work developing latent tree chart parsers (Socher et al., 2011b; Le and Zuidema, 2015; Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018). These methods produce representations for all internal nodes in the tree (cells in the chart), each generated as a soft weighting over all possible sub-trees (§2). Unfortunately, they still require sentence-level annotations during training, as they are all trained to optimize a downstream task, typically natural language inference.

To address these limitations, we present deep inside-outside recursive autoencoders (DIORA) which enable unsupervised discovery and representation of constituents without requiring any supervised training data. DIORA incorporates the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) into a latent tree chart parser. The bottom-up inside step calculates a representation for all possible constituents within a binary tree over the input sentence. This step is equivalent to the forward-pass of previous latent tree chart parsers (Maillard et al., 2017). These inside representations only encode the current subtree, ignor-

---

[*] Equal contribution, randomly ordered.

Proceedings of NAACL-HLT 2019, pages 1129–1141
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

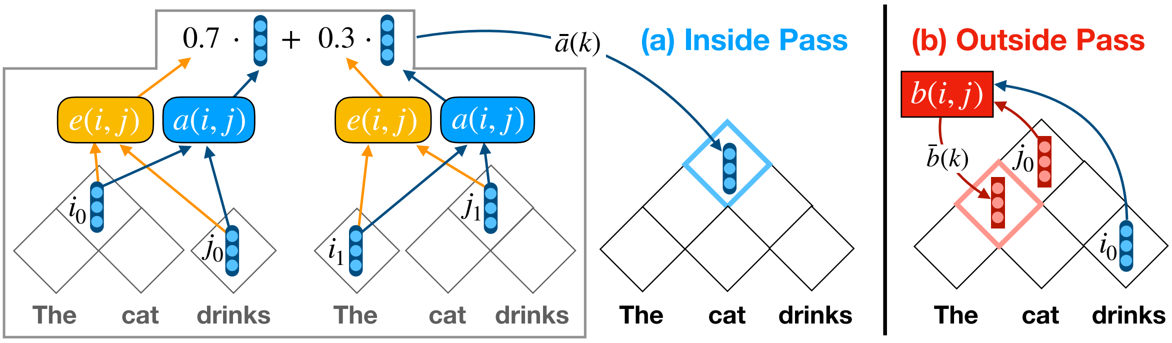


Figure 2: The illustrated inside and outside pass of DIORA operating over an input of length three, 'the cat drinks'. a) The inside pass: The inside vector $\bar{a}(k)$ for the phrase 'the cat drinks' is a weighted average of the compositions for the two possible segmentations - ((the cat), drinks) and (the, (cat drinks)). The scalar weights come from a learned compatibility function. b) The outside pass: The outside vector $\bar{b}(k)$ for the phrase 'the cat' is a function of the outside vector of its parent 'the cat drinks' and the inside vector of its sibling 'drinks'.

ing all outside context. Thus, we perform an additional top-down outside calculation for each node in the tree, providing external context into the subtree representations in each chart cell. The model is then trained with the objective that the outside representations of the leaf cells should reconstruct the corresponding leaf input word, analogous to masked language model (Devlin et al., 2019) pretraining, except by using dynamic programming we predict every word from a completely unmasked context. The single most likely tree can be recovered using the CKY algorithm and compatibility scores between constituents. Previous work either predict trees that are not well aligned with known treebanks (Yogatama et al., 2017; Choi et al., 2018), or has no mechanism for explicitly modeling phrases, requiring a complex procedure to extract syntactic structures (Shen et al., 2018).

To probe different properties of our model, we run experiments on unsupervised parsing, segment recall, and phrase representations. DIORA achieves multiple new state-of-the-art results for unsupervised constituency parsing (absolute improvements of **13.7%**, **11.5%**, and **7.8%** on WSJ, WSJ-40, and MultiNLI), has a greater recall on more constituent types than a strong baseline, and produces meaningful phrase representations.

2 DIORA: Deep Inside-Outside Recursive Autoencoders

Our goal is to design a model and unsupervised training procedure that learns structure from raw text. The design of DIORA is based on our hypothesis is that the most effective compression of a sentence will be derived from following the true syntactic structure of the underlying input. Our approach builds on previous latent tree chart parsers which are augmented with the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) and trained to reproduce each input word from its outside context. Based on our hypothesis, loosely inspired by the linguistic "substitution principle" (Frege, 1960), the model will best reconstruct the input by discovering and exploiting syntactic regularities of the text.

The inside pass of our method recursively compresses the input sequence, at each step inputting the vector representations of the two children into a composition function (§2.1.1) that outputs an inside vector representation of the parent. This process continues up to the root of the tree, eventually yielding a single vector representing the entire sentence (Figure 2a). This is loosely analogous to the compression step of an autoencoder and equivalent to existing latent tree chart parsers forward pass (Maillard et al., 2017). Following this, we initiate the outside pass of our algorithm with a generic (root) representation that is learned as a separate parameter. As the outside step of the inside-outside algorithm (Figure 2b), we unfold until finally producing representations of the leaf nodes. These leaves are then optimized to reconstruct the input sentence as done in an autoencoder-based deep neural network.

2.1 Filling the Chart with Inside-Outside

Each inside representation is the root of a particularly sub-tree, and that representation is generated by considering only the descendant constituents within that sub-tree, ignoring any outside context. After the inside representations are calculated, we perform a top-down outside pass to compute outside representations. The outside representations are encoded by looking at only the context of a given sub-tree. Once the chart is filled, each constituent k (cell in the chart) is associated with an inside vector $\bar{a}(k)$, an outside vector $\bar{b}(k)$, inside compatibility score $\bar{e}(k)$ and outside compatibility score $\bar{f}(k)$.

The input to our model is a sentence $\boldsymbol{x}$ made up of T tokens, $x_0, x_1, ..., x_{T-1}$. Each token x_i has a corresponding pre-trained embedded vector v_i.

2.1.1 Inside Pass

For each pair of neighboring constituents i and j [1], we compute a *compatibility* score and a *composition* vector. The score and vector that represent a particular span k are computed using a soft weighting over all possible pairs of constituents, that together fully cover the span (we refer to this set of constituent pairs as $\{k\}$).

Vectors for spans of length 1 are initialized as a non-linear transformation [2] of the embedded input v_i, and the scores associated with these spans are set to 0:

$$\begin{bmatrix} x \\ o \\ u \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} (U_\psi v_k + b)$$
$$\bar{a}(k) = o + \tanh(x \odot u)$$
$$\bar{e}(k) = 0$$

Higher levels of the chart are computed as a weighted summation of constituent pairs:

$$\bar{a}(k) = \sum_{i,j \in \{k\}} e(i,j)\, a(i,j)$$
$$\bar{e}(k) = \sum_{i,j \in \{k\}} e(i,j)\, \hat{e}(i,j)$$

The compatibility function $\hat{e}$ is meant to produce a score for how likely a pair of neighboring cells are to be merged. We implement this as a bilinear function of the vectors from neighboring spans, using a learned parameter matrix S. We additionally add the individual scores from each two merging cells. Intuitively, these individual scores correspond to how likely each of the cells would exist in the final binary tree independently. The formula for the compatibility function (and its normalized form e) is defined as follows:

$$e(i,j) = \frac{\exp(\hat{e}(i,j))}{\sum_{\hat{i},\hat{j} \in \{k\}} \exp(\hat{e}(\hat{i},\hat{j}))}$$
$$\hat{e}(i,j) = \phi(\bar{a}(i), \bar{a}(j); S_\alpha) + \bar{e}(i) + \bar{e}(j)$$

Where the bilinear projection ϕ is defined as:

$$\phi(u, v; W) = u^\top W v$$

For the composition function a we used either a TreeLSTM (Tai et al., 2015) or a 2-layer MLP (see Appendix A.2 for more precise definitions on both methods). In order for the remainder of equations to remain agnostic to the choice of composition function, we refer to the function as Compose, which produces a hidden state vector h and, in the case of TreeLSTM, a cell state vector c, resulting in:

$$a(i,j) = \text{Compose}_\alpha(\bar{a}(i), \bar{a}(j))$$

2.1.2 Outside Pass

The outside computation is similar to the inside pass (depicted in Figure 2b).

The root node of the outside chart is learned as a bias. Descendant cells are predicted using a disambiguation over the possible outside contexts. Each component of the context consists of a sibling cell from the inside chart and a parent cell from the outside chart.

The function f is analogous to the function e. It is normalized over constituent pairs i, j for the span k, and is used to disambiguate among the many outside contexts. The function b generates a phrase representation for the missing sibling cell. Equations for the outside computation follow:

$$\bar{b}(k) = \sum_{i,j \in \{k\}} f(i,j)\, b(i,j)$$
$$\bar{f}(k) = \sum_{i,j \in \{k\}} f(i,j)\, \hat{f}(i,j)$$
$$b(i,j) = \text{Compose}_\beta(\bar{a}(i), \bar{b}(j))$$
$$\hat{f}(i,j) = \phi(\bar{a}(i), \bar{b}(j); S_\beta) + \bar{e}(i) + \bar{f}(j)$$

In the majority of our experiments, the Compose used in b shares parameters with a used in the inside pass, as do the compatibility functions $\hat{e}$ and $\hat{f}$ (see §3.4 for results on the effects of parameter sharing).

---

[1] The symbols i, j, and k are identifiers of spans from the input $\boldsymbol{x}$. The symbol i^* identifies a token from the set of negative examples $\{\boldsymbol{x}^*\}$.

[2] This function shares its bias term b with Compose_α, although U_ψ is not tied to any other weights.

2.2 Training Objective

To train our model we use an autoencoder-like language modeling objective. In a standard autoencoder, the entire input $\boldsymbol{x}$ is compressed into a single lower dimensional representation. This representation, $\boldsymbol{z}$, is then decompressed and trained to reconstruct $\boldsymbol{x}$. In our model, we never condition the reconstruction of $\boldsymbol{x}$ on a single $\boldsymbol{z}$ because the root's outside representation is initialized with a bias rather than the root's own inside vector. Instead, we reconstruct $\boldsymbol{x}$ conditioned on the many sub-tree roots, each of which is only a compression of a *subset* of the input.

To approximate this reconstruction we use a max-margin loss considering a set $\{\boldsymbol{x}^*\}$ of N negative examples that are sampled according to their frequency from the vocabulary (further details in Appendix A.1). The terminal outside vector $\bar{b}(i)$ is trained to predict its original input v_i.

The per-instance loss function is described in Equation 1:

$$L_{\boldsymbol{x}} = \sum_{i=0}^{T-1} \sum_{i^*=0}^{N-1} \max(0, 1 - \bar{b}(i) \cdot \bar{a}(i) + \bar{b}(i) \cdot \bar{a}(i^*)) \qquad (1)$$

The max-margin loss does not provide a gradient if the predicted vector is closer to its ground truth than the negative example by a margin greater than 1. For that reason, we also experimented with an objective based on cross-entropy, described in Equation 2:

$$Z^* = \sum_{i^*=0}^{N-1} \exp(\bar{b}(i) \cdot \bar{a}(i^*))$$

$$L_{\boldsymbol{x}} = -\sum_{i=0}^{T-1} \log \frac{\exp(\bar{b}(i) \cdot \bar{a}(i))}{\exp(\bar{b}(i) \cdot \bar{a}(i)) + Z^*} \qquad (2)$$

2.3 DIORA CKY Parsing

To obtain a parse with DIORA, we populate an inside and outside chart using the input sentence. We can extract the maximum scoring parse based on our single grammar rule using the CKY procedure (Kasami, 1966; Younger, 1967). The steps for this procedure are described in Algorithm 1 and its runtime complexity in Appendix A.4.

3 Experiments

To evaluate the effectiveness of DIORA, we run experiments on unsupervised parsing, unsupervised segment recall, and phrase similarity. The model has been implemented in PyTorch (Team, 2018) and the code is published online.[3] For training details, see Appendix A.1.

Algorithm 1 Parsing with DIORA

```
procedure CKY(chart)
    Initialize terminal values.
    for each $k \in$ chart | SIZE$(k) = 1$ do
        $x_k \leftarrow 0$
    Calculate a maximum score for each span,
    and record a backpointer.
    for each $k \in$ chart do
        $x_k \leftarrow \max_{i,j \in \{k\}} [x_i + x_j + e(i,j)]$
        $\pi_k^i, \pi_k^j \leftarrow \arg\max_{i,j \in \{k\}} [x_i + x_j + e(i,j)]$
    Backtrack to get the maximal tree.
    procedure BACKTRACK($k$)
        if SIZE$(k) = 1$ then
            return k
        $i \leftarrow$ BACKTRACK$(\pi_k^i)$
        $j \leftarrow$ BACKTRACK$(\pi_k^j)$
        return (i, j)
    return BACKTRACK($k \leftarrow$ root)
```

3.1 Unsupervised Parsing

We first evaluate how well our model predicts a full unlabeled constituency parse. We look at two data sets used in prior work (Htut et al., 2018), The Wall Street Journal (WSJ) section of Penn Treebank (Marcus et al., 1993), and the automatic parses from MultiNLI (Williams et al., 2018b). WSJ has gold human-annotated parses and MultiNLI contains automatic parses derived from a supervised parser (Manning et al., 2014).

In addition to PRPN (Shen et al., 2018),[4] we compare our model to deterministically constructed left branching, right branching, balanced, and random trees. We also compare to ON-LSTM (Shen et al., 2019), an extension of the PRPN model, RL-SPINN (Yogatama et al., 2017), an unsupervised shift-reduce parser, and ST-Gumbel (Choi et al., 2018), an unsupervised chart parser. The latter two of these models are trained to predict the downstream task of natural language inference (NLI).

---

[3] `https://github.com/iesl/diora`

[4] We consider the PRPN models using LM stopping criteria, which outperformed UP.

3.1.1 Binarized WSJ and MultiNLI results

For the full WSJ test set and MultiNLI datasets we follow the experimental setup of previous work (Williams et al., 2018a). We binarize target trees using Stanford CoreNLP (Manning et al., 2014) and do not remove punctuation (experiments in §3.1.2 do remove punctuation).

Latent tree models have been shown to perform particularly poorly on attachments at the beginning and end of the sequence (Williams et al., 2018a). To address this, we incorporate a post-processing heuristic (denoted as +PP in result tables)[5]. This heuristic simply attaches trailing punctuation to the root of the tree, regardless of its predicted attachment.

In Table 1, we see that DIORA^{+PP} achieves the highest average and maximum F1 from five random restarts. This model achieves a mean F1 7 points higher than ON-LSTM and an increase of over 6.5 max F1 points. We also see that DIORA exhibits much less variance between random seeds than ON-LSTM. Additionally, we find that PRPN-UP and DIORA benefit much more from the +PP heuristic than PRPN-LM. This is consistent with qualitative analysis showing that DIORA and PRPN-UP incorrectly attach trailing punctuation much more often than PRPN-LM.

On the MultiNLI dataset, PRPN-LM is the top performing model without using the +PP heuristic while DIORA matches PRPN-UP (Table 2. Using the heuristic, DIORA greatly surpasses both variants of PRPN. However, it is worth noting that this is not a gold standard evaluation and instead evaluates a model's ability to replicate the output of a trained parser (Manning et al., 2014). A second caveat is that SNLI (Bowman et al., 2015) and MultiNLI contain several non-newswire domains. Syntactic parsers often suffer significant performance drops when predicting outside of the newswire domain that the models were trained on.

Model	**F1**$_\mu$	**F1**$_{max}$	δ
LB	13.1	13.1	12.4
RB	16.5	16.5	12.4
Random	21.4	21.4	5.3
Balanced	21.3	21.3	4.6
RL-SPINN†	13.2	13.2	-
ST-Gumbel - GRU†	22.8 ±1.6	25.0	-
PRPN-UP	38.3 ±0.5	39.8	5.9
PRPN-LM	35.0 ±5.4	42.8	6.2
ON-LSTM	47.7 ±1.5	49.4	5.6
DIORA	48.9 ±0.5	49.6	8.0
PRPN-UP^{+PP}	-	45.2	6.7
PRPN-LM^{+PP}	-	42.4	6.3
DIORA^{+PP}	**55.7** ±0.4	**56.2**	8.5

Table 1: Full WSJ (test set) unsupervised unlabeled binary constituency parsing including punctuation. † indicates trained to optimize NLI task. Mean and max are calculated over five random restarts. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). The depth (δ) is the average tree height. +PP refers to post-processing heuristic that attaches trailing punctuation to the root of the tree. The top F1 value in each column is bolded.

Model	**F1**$_{median}$	**F1**$_{max}$	δ
Random	27.0	27.0	4.4
Balanced	21.3	21.3	3.9
PRPN-UP	48.6	-	4.9
PRPN-LM	50.4	-	5.1
DIORA	51.2	53.3	6.4
PRPN-UP^{+PP}	-	54.8	5.2
PRPN-LM^{+PP}	-	50.4	5.1
DIORA^{+PP}	59.0	**59.1**	6.7

Table 2: NLI unsupervised unlabeled binary constituency parsing comparing to CoreNLP predicted parses. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). F1 median and max are calculated over five random seeds and the top F1 value in each column is bolded. Note that we use median rather than mean in order to compare with previous work.

3.1.2 WSJ-10 and WSJ-40 results

We also compare our models to two subsets of the WSJ dataset that were used in previous unsupervised parsing evaluations. WSJ-10 and WSJ-40 contain sentences up to length 10 and 40 respectively after punctuation removal. We do not binarize either of these two splits in order to compare to previous work (see Appendix A.3 for more details on WSJ split differences). Not binarizing the target trees sets an upper-bound on the performance of our models, denoted as UB in Table 3.

We compare against previous notable models for this task: CCM (Klein and Manning, 2002) uses the EM algorithm to learn probable nested

---

[5] We did not have access to predictions or an implementation of the concurrent ON-LSTM model and therefore could not apply the +PP heuristic.

bracketings over a sentence using gold or induced part-of-speech tags, and PRLG (Ponvert et al., 2011) performs constituent parsing through consecutive rounds of sentence chunking.

In Table 3, we see that DIORA outperforms the previous state of the art for WSJ-40, PRLG, in max F1. The WSJ-10 split has been difficult for latent tree parsers such as DIORA, PRPN, and ON-LSTM, none of which (including our model) are able to improve upon previous non-neural methods. However, when we compare trends between WSJ-10 and WSJ-40, we see that DIORA does a better job at extending to longer sequences.

3.2 Unsupervised Phrase Segmentation

In many scenarios, one is only concerned with extracting particular constituent phrases rather than a full parse. Common use cases would be identifying entities, noun phrases, or verb phrases for downstream analysis. To get an idea of how well our model can perform on phrase segmentation, we consider the maximum recall of spans in our predicted parse tree. We leave methods for cutting the tree to future work and instead consider the maximum recall of our model which serves as an upper bound on its performance. Recall here is the percentage of labeled constituents that appear in our predicted tree relative to the total number of constituents in the gold tree. These scores are separated by type and presented in Table 4.

In Table 4 we see the breakdown of constituent recall across the 10 most common types. DIORA achieves the highest recall across the most types and is the only model to perform effectively on verb-phrases. Interestingly, DIORA performs worse than PRPN-LM at prepositional phrases.

3.3 Phrase Similarity

One of the goals of DIORA is to learn meaningful representations for spans of text. Most language modeling methods focus only on explicitly modeling token representations and rely on ad-hoc post-processing to generate representations for longer spans, typically relying on simple arithmetic functions of the individual tokens.

To evaluate our model's learned phrase representations, we look at the similarity between spans of the same type within labeled phrase datasets. We look at two datasets. CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000) is a shallow parsing dataset containing spans of noun phrases, verb phrases, etc. CoNLL 2012 (Pradhan et al., 2012) is a named entity dataset containing 19 different entity types.

	WSJ-10		**WSJ-40**	
Model	**F1**$_{\mu}$	**F1**$_{max}$	**F1**$_{\mu}$	**F1**$_{max}$
UB	87.8	87.8	85.7	85.7
LB	28.7	28.7	12.0	12.0
RB	61.7	61.7	40.7	40.7
CCM†	-	63.2	-	-
CCM$_{gold}$†	-	71.9	-	33.7
PRLG †	-	**72.1**	-	54.6
PRPN$_{NLI}$	66.3 ±0.8	68.5	-	-
PRPN‡	70.5 ±0.4	71.3	-	52.4
ON-LSTM‡	65.1 ±1.7	66.8	-	-
DIORA	67.7 ±0.7	68.5	60.6 ±0.2	**60.9**

Table 3: WSJ-10 and WSJ-40 unsupervised non-binary unlabeled constituency parsing with punctuation removed. † indicates that the model predicts a full, non-binary parse with additional resources. ‡ indicates model was trained on WSJ data and PRPN$_{NLI}$ was trained on MultiNLI data. CCM uses predicted POS tags while CCM$_{gold}$ uses gold POS tags. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). LB and RB are the left and right-branching baselines. UB is the upper bound attainable by a model that produces binary trees.

For each of the labeled spans with length greater than one, we first generate its phrase representation. We then calculate its cosine similarity to all other labeled spans. We then calculate if the label for that query span matches the labels for each of the K most similar other spans in the dataset. In Table 5 we report precision@K for both datasets and various values of K.

The first baseline we compare against produces phrase representations from averaging context-insensitive (CI) ELMo vectors of individual tokens with the span. The second uses sentence-insensitive (SI) ELMo vectors, running the full ELMo over only the relevant tokens and ignoring the rest of the sentence. We also look at ELMo's output when given the entire sentence. When analyzing our baselines that run the full ELMo, we follow the procedure described in (Peters et al., 2018b) and represent phrases as a function of its first and last hidden state. We extract these states from the final ELMo layer (3rd BiLSTM) as these consistently gave the best performance among other options. For DIORA, we use the concatenation of the inside and outside representations ($[\bar{a}; \bar{b}]$).

Label	Count	DIORA	P-UP	P-LM
NP	297,872	**0.767**	0.687	0.598
VP	168,605	**0.628**	0.393	0.316
PP	116,338	0.595	0.497	**0.602**
S	87,714	**0.798**	0.639	0.657
SBAR	24,743	**0.613**	0.403	0.554
ADJP	12,263	**0.604**	0.342	0.360
QP	11,441	**0.801**	0.336	0.545
ADVP	5,817	**0.693**	0.392	0.500
PRN	2,971	**0.546**	0.127	0.144
SINV	2,563	0.926	0.904	**0.932**

Table 4: Segment recall from WSJ separated by phrase type. The 10 most frequent phrase types are shown above, and the highest value in each row is bolded. P-UP=PRNP-UP, P-LM=PRPN-LM

For CoNLL 2000, we find that our model outperforms all baselines for all values of K. This demonstrates DIORA's ability to capture and represent syntactic information within phrases. For CoNLL 2012, we find that DIORA outperforms both ELMo$_{CI}$ and ELMo$_{SI}$ while ELMo performs best overall. ELMo$_{CI}$ is surprisingly effective on this dataset even though it performed more poorly on CoNLL 2000. These results indicate that DIORA is capturing syntax quite well, but still has room to improve on more fine-grained semantic representations.

3.4 Impact of Modeling Choices

To test the impact of our modeling choices, we compared the performance of two different losses and four different composition functions on the full WSJ validation set. The losses were covered in Equations 1 (Margin) and 2 (Softmax). The two primary methods of composition we considered were TreeLSTM (Tai et al., 2015) and MLP (a 2-hidden layer neural network). In addition, we experimented with a simple *kernel* of the MLP input $[x; y; x \odot y; x - y]$ and with a setting where both the inside and outside parameters are *shared*.

The results are shown in Table 6. We see that MLP composition consistently performs better than with TreeLSTM, that MLP benefits from the Softmax loss, and that the best performance comes from sharing parameters. All other experimental results use this highly performant setting unless otherwise specified.

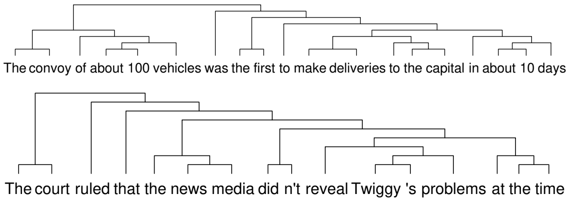


Figure 3: DIORA can match the ground truth exactly.

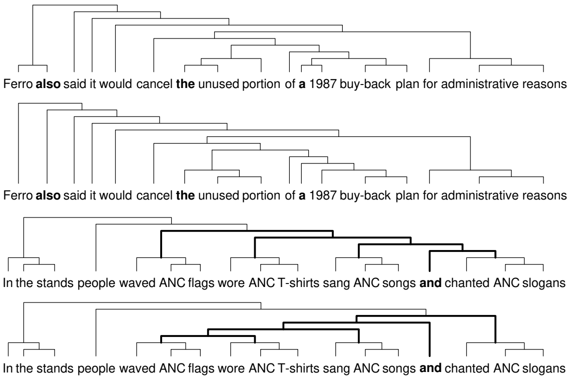


Figure 4: At times, DIORA exhibits contrary behavior to the ground truth inevitably leading to some error. DIORA's output is shown above the ground truth.[6]

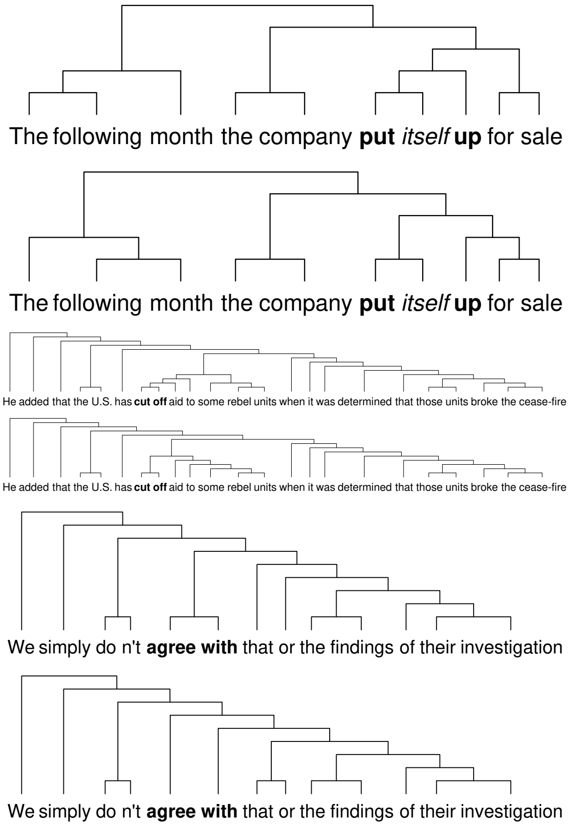


Figure 5: DIORA often groups verbs and particles (top), sometimes exactly as the ground truth (middle). Occasionally, errors are particle-like (bottom). DIORA's output is shown above the ground truth.[6]

---

[6] Ground truth parses are binarized unless otherwise specified. All examples of DIORA parses are already binary. Some punctuation has been removed for easier readability.

		CoNLL 2000			**CoNLL 2012**		
Model	Dim	P@1	P@10	P@100	P@1	P@10	P@100
Random	800	0.684	0.683	0.680	0.137	0.133	0.135
ELMo$_{CI}$	1024	0.962	0.955	0.957	0.708	0.643	0.544
ELMo$_{SI}$	4096	0.970	0.964	0.955	0.660	0.624	0.533
ELMo	4096	0.987	0.983	0.974	**0.896**	**0.847**	**0.716**
DIORA$_{In/Out}$	800	**0.990**	**0.985**	**0.979**	0.860	0.796	0.646

Table 5: P@1, P@10, and P@100 for labeled chunks from CoNLL-2000 and CoNLL 2012 datasets. For all metrics, higher is better. The top value in each column is bolded. Diora uses the concatenation of the inside and outside vector at each cell which performed better than either in isolation.

3.5 Qualitative Results

Looking at our model's output, we see that some trees are an exact replication of the binarized ground truth (Fig. 3), or very close (Fig. 4). For future work we intend to explore common patterns in DIORA's learned structure, although some patterns are already recognizable, such as the affinity to group particles and verbs (Fig. 5).

4 Related Work

Latent Tree Learning A brief survey of neural latent tree learning models was covered in (Williams et al., 2018a). The first positive result for neural latent tree parsing was shown in (Htut et al., 2018), which used a language modeling objective. The model in (Liu et al., 2018) uses an inside chart and an outside procedure to calculate marginal probabilities in order to align spans between sentences in entailment.

		F1$_\mu$	
Composition	**Loss**	∅	+PP
TreeLSTM	Margin	49.9	53.1
TreeLSTM	Softmax	52.0	52.9
MLP	Margin	49.7	54.4
MLP	Softmax	52.6	55.5
MLP$_{\text{Kernel}}$	Softmax	51.8	54.8
MLP$_{\text{Shared}}$	Softmax	50.8	56.7

Table 6: F1 for different model variants on the binary WSJ validation set with included punctuation. The binary trees are as-is (∅) or modified according to the post-processing heuristic (+PP). The mean F1 is shown across three random seeds.

Neural Inside-Outside Parsers The Inside-Outside Recursive Neural Network (IORNN) (Le and Zuidema, 2014) is closest to ours. It is a graph-based dependency parser that uses beam search and can reliably find accurate parses when retaining a k-best list. In contrast, our model produces the most likely parse given the learned compatibility of the constituents. The Neural CRF Parser (Durrett and Klein, 2015), similar to DIORA, performs exact inference on the structure of a sentence, although requires a set of grammar rules and labeled parse trees during training. DIORA, like Liu et al. (2018), has a single grammar rule that applies to any pair of constituents and does not use structural supervision.

Learning from Raw Text Unsupervised learning of syntactic structure has been an active research area (Brill et al., 1990), including for unsupervised segmentation (Ando and Lee, 2000; Goldwater et al., 2009; Ponvert et al., 2011) and unsupervised dependency parsing (Spitkovsky et al., 2013). Some models exploit the availability of parallel corpora in multiple languages (Das and Petrov, 2011; Cohen et al., 2011). Others have shown that dependency parsing can be used for unsupervised constituency parsing (Spitkovsky et al., 2013; Klein and Manning, 2004), or that it's effective to prune a random subset of possible trees (Bod, 2006). These approaches aren't necessarily orthogonal to DIORA. For instance, our model may benefit when combined with an unsupervised dependency parser.

5 Conclusion

In this work we presented DIORA, an unsupervised method for inducing syntactic trees and representations of constituent spans. We showed inside-outside representations constructed with a latent tree chart parser and trained with an autoencoder language modeling objective learns syntactic structure of language effectively. In experiments on unsupervised parsing, chunking, and phrase representations we show our model is comparable to or outperforms previous methods, achieving the state-of-the-art performance on unsupervised unlabeled constituency parsing for the full WSJ (with punctuation), WSJ-40, and NLI datasets. We also show our model obtains higher segment recall than a comparable model and outperforms strong baselines on phrase representations on a chunking dataset.

While the current model seems to focus primarily on syntax, future work can improve the model's ability to capture fine-grained semantics. Potential avenues include training larger models over much larger corpora, extra unsupervised or weakly-supervised phrase classification objectives, and other modeling enhancements. We are also eager to apply DIORA to other domains and languages which do not have rich linguistically annotated training sets.

Acknowledgements

We are grateful to Carolyn Anderson, Adina Williams, Phu Mon Htut, and our colleagues at UMass for help and advice, and to the UMass NLP reading group and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1514053 and CNS-0958392. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Association for Computational Linguistics (ACL)*.

Rie Kubota Ando and Lillian Lee. 2000. Mostly-unsupervised statistical segmentation of japanese: Applications to kanji. In *North American Association for Computational Linguistics (NAACL)*.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Association for Computational Linguistics (ACL)*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. 1990. Deducing linguistic structure from the statistics of large corpora. In *Information Technology, 1990.'Next Decade in Information Technology', Proceedings of the 5th Jerusalem Conference on (Cat. No. 90TH0326-9)*, pages 380–389. IEEE.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Association for Computational Linguistics (ACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *North American Association for Computational Linguistics (NAACL)*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Association for Computational Linguistics (ACL)*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Association for Computational Linguistics (ACL)*.

Friedrich Ludwig Gottlob Frege. 1960. On sense and reference. In *Zeitschrift für Philosophie und philosophische Kritik 100 (1892) 25-50; translated in Translations from the Philosophical Writings of Gottlob Frege (ed. by P. Geach and M. Black)*. Oxford.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Joint workshop on unsupervised and semi-supervised learning in NLP*. Association for Computational Linguistics (ACL).

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112:21–54.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Association for Computational Linguistics (ACL)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel R Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Association for Computational Linguistics (ACL)*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Association for Computational Linguistics (ACL)*.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Phong Le and Willem Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1155–1164.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured alignment networks for matching sentences. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *arXiv preprint arXiv:1705.09189*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics (ACL): System Demonstrations*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Association for Computational Linguistics (ACL)*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations (ICLR)*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Conference on Computational Natural Language Learning (CoNLL)*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Association for Computational Linguistics (ACL)*.

Pytorch Core Team. 2018. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. `http://pytorch.org/`. Accessed: 2018-09-26.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association of Computational Linguistics (TACL)*, 6:253–267.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Association for Computational Linguistics (NAACL)*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *International Conference on Computational Linguistics*.

A Appendices

A.1 Training Details

Training Data. Sentences of length ≤ 20 from the SNLI and MultiNLI training sets.
Optimization. We train our model using stochastic gradient descent with the Adam optimization algorithm (Kingma and Ba, 2014). Cells were normalized to have magnitude of 1, following Socher et al. (2011a). For instance, $\bar{a}(k) := \bar{a}(k)/\left\|\bar{a}(k)\right\|^2$. Gradients are clipped to a maximum L2-norm of 5.
Hyperparameters. Chosen using grid search over cell-dimension $\{400\text{D}, 800\text{D}\}$ and learning rate $\{2, 4, 8, 10, 20\} \cdot 10^{-4}$.
Early Stopping. Using unlabeled parsing F1 against the binarized WSJ validation set.
Vocabulary. The model is trained in an open-vocabulary setting using pre-trained context-insensitive character embeddings. The embedder is taken from ELMo (Peters et al., 2018a).
Batching. Batches were constructed such that they contained sentences of uniform length. Using batch size 128 for 400D and 64 for 800D.
Sampling. N negatives are sampled for each batch. All experiments use $N = 100$.
Training Steps. 1M parameter updates, taking 3 days using 4x Nvidia 1080ti.

A.2 Composition and Input Transform

TreeLSTM. The $\mathrm{TreeLSTM}$ (Tai et al., 2015) function produces a hidden state vector h and cell state vector c given two input vectors h_i and h_j.

$$\begin{bmatrix} x \\ f_i \\ f_j \\ o \\ u \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(U \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b + \begin{bmatrix} 0 \\ \omega \\ \omega \\ 0 \\ 0 \end{bmatrix} \right)$$

$$\begin{aligned} c &= c_i \odot f_i + c_j \odot f_j + x \odot u \\ h &= o + \tanh(c) \end{aligned}$$

The constant ω is set to 1 for the inside, 0 for the outside. U and b are learned.

MLP. MLP (Multi-Layer Perceptron) is a deep non-linear composition with the following form:

$$h = W_1 \left(W_0 \left\langle h_i, h_j \right\rangle + b \right) + b_1$$

The operator $\langle h_i, h_j \rangle$ is a concatenation $[h_i; h_j]$. For the MLP^{Kernel} $\langle h_i, h_j \rangle$ is more involved to support further interaction between the two input vectors $[h_i; h_j; h_i \odot h_j; h_i - h_j]$. The variables W_0, W_1, b, b_1 are learned and c is unused.

A.3 Reproducing Parsing Results

In Table 7, we've organized a reference for creating various splits of the WSJ for the purpose of evaluating unsupervised parsing. Some splits use only the test set (section 23), others use all of the training, validation, and test data. Optionally, punctuation is stripped and sentences greater than a specified length are ignored. Predictions can be compared to the full parse trees in the annotated data, or to a binarized version. The PARSEVAL specification calculated bracketing F1 considering all spans, although some previous work diverts from PARSEVAL and ignores spans that are trivially correct (ones over the entire sentence).

	WSJ	WSJ-10	WSJ-40
Split	Test	All	Test
w/ Punctuation	Yes	No	No
Max Length	∞	10	40
Binarized	Yes	No	No
Trivial Spans	Yes	No	No

Table 7: Settings for unlabeled binary bracketing evaluation for different splits of the WSJ corpus.

A.4 Runtime Complexity

The runtime complexities for DIORA's methods are shown in Table 8. The parallel column represents the complexity when the values for all constituent pairs are computed simultaneously, assuming that these computations are independent and do not depend on values that have yet to be computed. Linear complexity is theoretically feasible depending on batch size, input length, and number of computational cores. In practice, one might experience super-linear performance.

Although both the inside pass and outside pass have an upper bound of n^3 operations, the outside pass will have more operations than the inside pass for sentences of length > 1.

As a point of reference, our implementation computes the loss over the entire WSJ corpus in 5 minutes 30 seconds at a rate of 3,500 words per second using a single GPU.

Method	Serial	Parallel
Inside Pass	$O(n^3)$	$O(n)$
Outside Pass	$O(n^3)$	$O(n)$
Training Objective	$O(n \cdot N)$	$O(n)$
CKY	$O(n^3)$	$O(n)$

Table 8: Runtime complexity for methods associated with DIORA in terms of sentence length n and number of negative examples per token N. Each column represents the complexity when the values for each constituent are computed serially or in parallel.

A.5 Parse Trees

Examples of parse trees derived from the compatibility scores are shown in Figures 6 and 7. Some punctuation has been removed for readability.

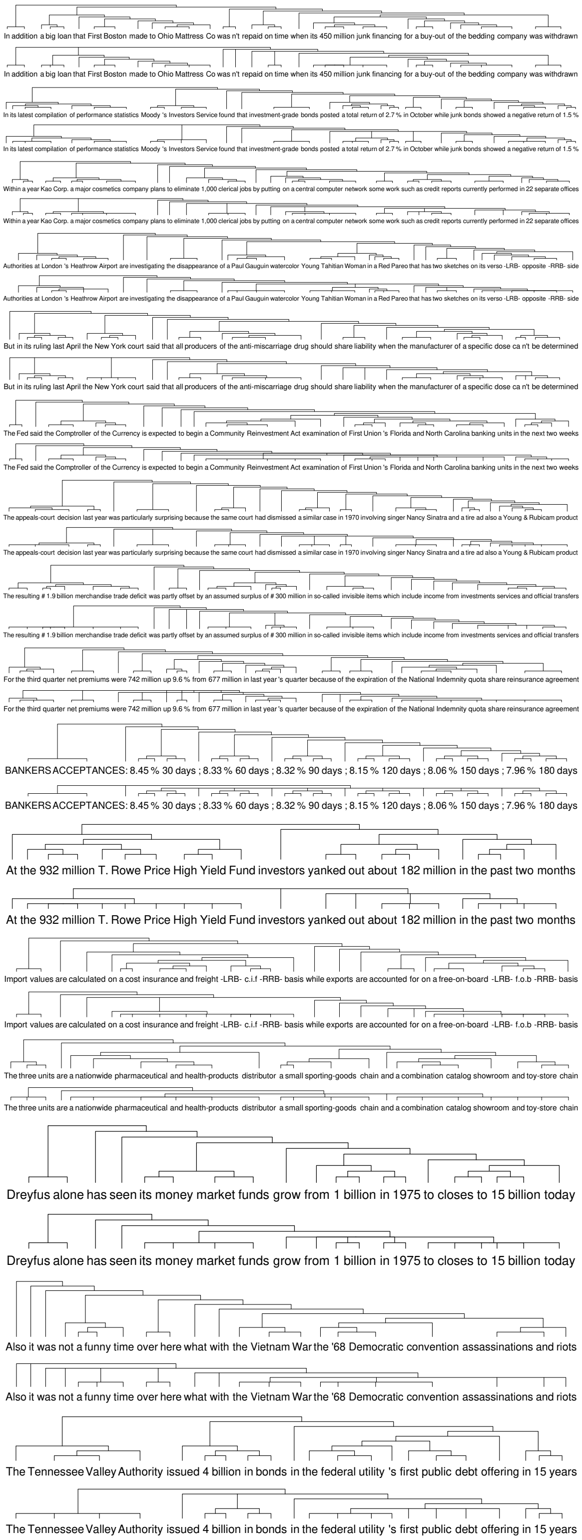


Figure 6: Examples where DIORA achieves 100% recall compared with the raw (n-ary) ground truth, but less than 100% accuracy on the binarized ground truth. DIORA's output is shown above the ground truth.

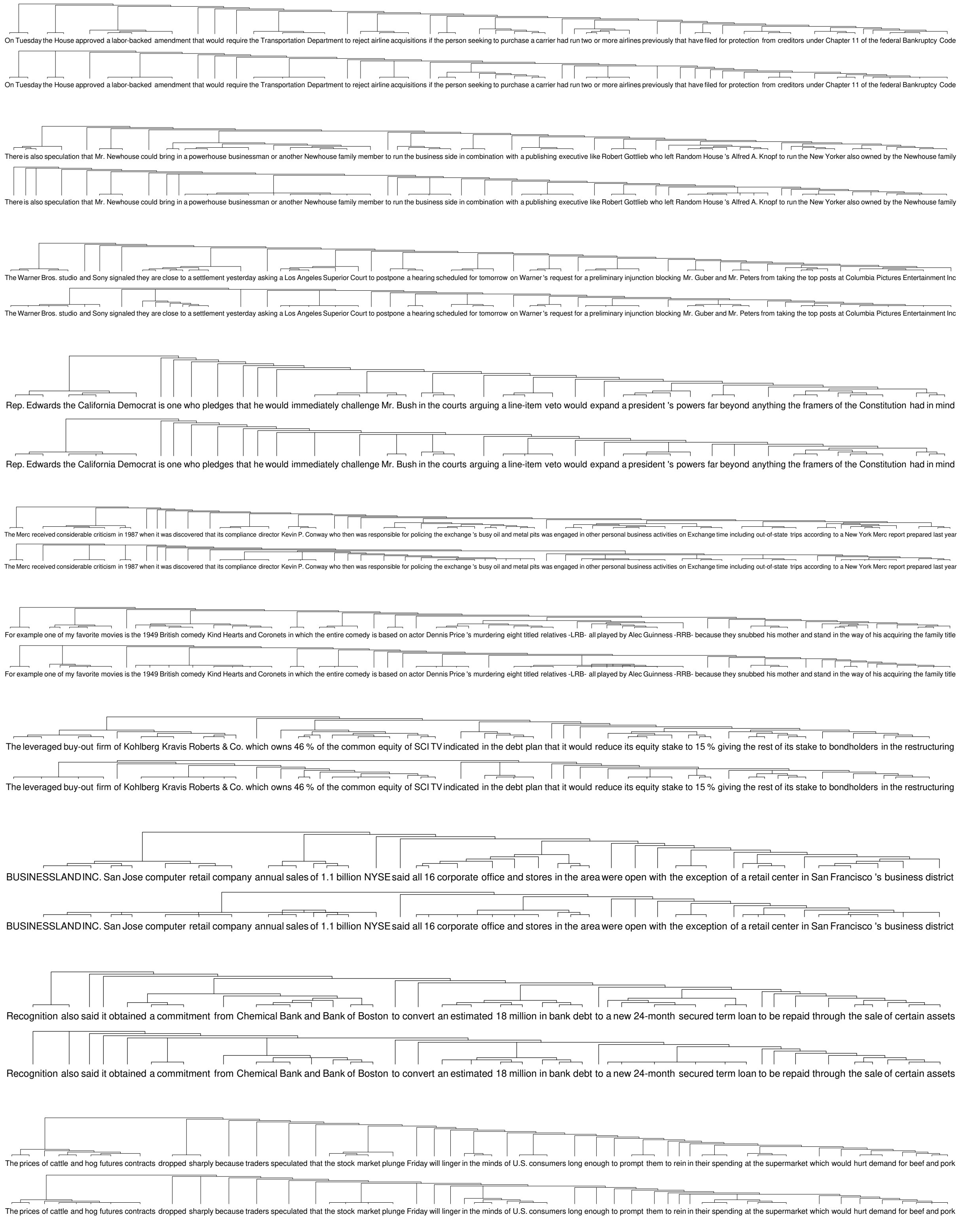


Figure 7: DIORA can perform close to the ground truth even on long sentences. In this figure, n-ary trees are shown for the ground truth. DIORA's output is shown above the ground truth.

Knowledge-Augmented Language Model and Its Application to Unsupervised Named-Entity Recognition

Angli Liu
Facebook AI
anglil@cs.washington.edu

Jingfei Du
Facebook AI
jingfeidu@fb.com

Veselin Stoyanov
Facebook AI
ves@fb.com

Abstract

Traditional language models are unable to efficiently model entity names observed in text. All but the most popular named entities appear infrequently in text providing insufficient context. Recent efforts have recognized that context can be generalized between entity names that share the same type (e.g., *person* or *location*) and have equipped language models with access to an external knowledge base (KB). Our Knowledge-Augmented Language Model (KALM) continues this line of work by augmenting a traditional model with a KB. Unlike previous methods, however, we train with an end-to-end predictive objective optimizing the perplexity of text. We do not require any additional information such as named entity tags. In addition to improving language modeling performance, KALM learns to recognize named entities in an entirely unsupervised way by using entity type information latent in the model. On a Named Entity Recognition (NER) task, KALM achieves performance comparable with state-of-the-art supervised models. Our work demonstrates that named entities (and possibly other types of world knowledge) can be modeled successfully using predictive learning and training on large corpora of text without any additional information.

1 Introduction

Language modeling is a form of unsupervised learning that allows language properties to be learned from large amounts of unlabeled text. As components, language models are useful for many Natural Language Processing (NLP) tasks such as generation (Parvez et al., 2018) and machine translation (Bahdanau et al., 2014). Additionally, the form of predictive learning that language modeling uses is useful to acquire text representations that can be used successfully to improve a number of downstream NLP tasks (Peters et al., 2018; Devlin et al., 2018). In fact, models pre-trained with a predictive objective have provided a new state-of-the-art by a large margin.

Current language models are unable to encode and decode factual knowledge such as the information about entities and their relations. Names of entities are an open class. While classes of named entities (e.g., *person* or *location*) occur frequently, each individual name (e.g, *Atherton* or *Zhouzhuang*) may be observed infrequently even in a very large corpus of text. As a result, language models learn to represent accurately only the most popular named entities. In the presence of external knowledge about named entities, language models should be able to learn to generalize across entity classes. For example, knowing that *Alice* is a name used to refer to a person should give ample information about the context in which the word may occur (e.g., *Bob visited Alice*).

In this work, we propose Knowledge Augmented Language Model (KALM), a language model with access to information available in a KB. Unlike previous work, we make no assumptions about the availability of additional components (such as Named Entity Taggers) or annotations. Instead, we enhance a traditional LM with a gating mechanism that controls whether a particular word is modeled as a general word or as a reference to an entity. We train the model end-to-end with only the traditional predictive language modeling perplexity objective. As a result, our system can model named entities in text more accurately as demonstrated by reduced perplexities compared to traditional LM baselines. In addition, KALM learns to recognize named entities completely unsupervised by interpreting the predictions of the gating mechanism at test time. In fact, KALM learns an unsupervised named entity tagger that rivals in accuracy supervised counterparts.

Proceedings of NAACL-HLT 2019, pages 1142–1150
Minneapolis, Minnesota, June 2 - June 7, 2019.

KALM works by providing a language model with the option to generate words from a set of entities from a database. An individual word can either come from a general word dictionary as in traditional language model or be generated as a name of an entity from a database. Entities in the database are partitioned by type. The decision of whether the word is a general term or a named entity from a given type is controlled by a gating mechanism conditioned on the context observed so far. Thus, KALM learns to predict whether the context observed is indicative of a named entity of a given type and what tokens are likely to be entities of a given type.

The gating mechanism at the core of KALM is similar to attention in Neural Machine Translation (Bahdanau et al., 2014). As in translation, the gating mechanism allows the LM to represent additional latent information that is useful for the end task of modeling language. The gating mechanism (in our case entity type prediction) is latent and learned in an end-to-end manner to maximize the probability of observed text. Experiments with named entity recognition show that the latent mechanism learns the information that we expect while LM experiments show that it is beneficial for the overall language modeling task.

This paper makes the following contributions:

- Our model, KALM, achieves a new state-of-the art for Language Modeling on several benchmarks as measured by perplexity.
- We learn a named entity recognizer without any explicit supervision by using only plain text. Our unsupervised named entity recognizer achieves a performance on par with the state-of-the supervised methods.
- We demonstrate that predictive learning combined with a gating mechanism can be utilized efficiently for generative training of deep learning systems beyond representation pre-training.

2 Related Work

Our work draws inspiration from Ahn et al. (2016), who propose to predict whether the word to generate has an underlying fact or not. Their model can generate knowledge-related words by copying from the description of the predicted fact. While theoretically interesting, their model functions only in a very constrained setting as it requires extra information: a shortlist of candidate entities that are mentioned in the text.

Several efforts successfully extend LMs with entities from a knowledge base and their types, but require that entity models are trained separately from supervised entity labels. Parvez et al. (2018) and Xin et al. (2018) explicitly model the type of the next word in addition to the word itself. In particular, Parvez et al. (2018) use two LSTM-based language models, an entity type model and an entity composite (entity type) model. Xin et al. (2018) use a similarly purposed entity typing module and a LM-enhancement module. Instead of entity type generation, Gu et al. (2018) propose to explicitly decompose word generation into sememe (a semantic language unit of meaning) generation and sense generation, but requires sememe labels.Yang et al. (2016) propose a pointer-network LM that can point to a 1-D or 2-D database record during inference. At each time step, the model decides whether to point to the database or the general vocabulary.

Unsupervised predictive learning has been proven effective in improving text understanding. ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) used different unsupervised objectives to pre-train text models which have advanced the state-of-the-art for many NLP tasks. Similar to these approaches KALM is trained end-to-end using a predictive objective on large corpus of text.

Most unsupervised NER models are rule-based (Collins and Singer, 1999; Etzioni et al., 2005; Nadeau et al., 2006) and require feature engineering or parallel corpora (Munro and Manning, 2012). Yang and Mitchell (2017) incorporate a KB to the CRF-biLSTM model (Lample et al., 2016) by embedding triples from a KB obtained using *TransE* (Bordes et al., 2013). Peters et al. (2017) add pre-trained language model embeddings as knowledge to the input of a CRF-biLSTM model, while still requiring labels in training.

To the best of our knowledge, KALM is the first unsupervised neural NER approach. As we discuss in Section 5.4, KALM achieves results comparable to supervised CRF-biLSTM models.

3 Knowledge-Augmented Language Model

KALM extends a traditional, RNN-based neural LM. As in traditional LM, KALM predicts probabilities of words from a vocabulary V_g, but it can also generate words that are names of entities of a specific type. Each entity type has a separate vocabulary $\{V_1, ..., V_K\}$ collected from a KB. KALM learns to predict from context whether to expect an entity from a given type and generalizes over entity types.

3.1 RNN language model

At its core, a language model predicts a distribution for a word y_{t+1} given previously observed words $c_t := [y_1, ..., y_{t-1}, y_t]$. Models are trained by maximizing the likelihood of the observed next word. In an LSTM LM, the probability of a word, $P(y_{t+1}|c_t)$, is modeled from the hidden state of an LSTM (Hochreiter and Schmidhuber, 1997):

$$P(y_{t+1} = i|c_t) = \frac{\exp(\boldsymbol{W}^p_{i,:} \cdot \boldsymbol{h}_t)}{\sum_{w=1}^{|V_g|} \exp(\boldsymbol{W}^p_{w,:} \cdot \boldsymbol{h}_t)} \tag{1}$$

$$\boldsymbol{h}_t, \boldsymbol{\gamma}_t = lstm(\boldsymbol{h}_{t-1}, \boldsymbol{\gamma}_{t-1}, \boldsymbol{y}_t) \tag{2}$$

where $lstm$ refers to the LSTM step function and h_i, $\boldsymbol{\gamma}_i$ and y_i are the hidden, memory and input vectors, respectively. $\boldsymbol{W}^p$ is a projection layer that converts LSTM hidden states into logits that have the size of the vocabulary $|V_g|$.

3.2 Knowledge-Augmented Language Model

KALM builds upon the LSTM LM by adding type-specific entity vocabularies $V_1, V_2, ..., V_K$ in addition to the general vocabulary V_g. Type vocabularies are extracted from the entities of specific type in a KB. For a given word, KALM computes a probability that the word represents an entity of that type by using a type-specific projection matrix $\{\boldsymbol{W}^{p,j}|j = 0, ..., K\}$. The model also computes the probability that the next word represents different entity types given the context observed so far. The overall probability of a word is given by the weighted sum of the type probabilities and the probability of the word under the give type.

More precisely, let τ_i be a latent variable denoting the type of word i. We decompose the probability in Equation 1 using the type τ_{t+1}:

$$\begin{aligned} P(y_{t+1}|c_t) &= \sum_{j=0}^{K} P(y_{t+1}, \tau_{t+1} = j|c_t) \\ &= \sum_{j=0}^{K} P(y_{t+1}|\tau_{t+1} = j, c_t) \\ &\quad \cdot P(\tau_{t+1} = j|c_t) \end{aligned} \tag{3}$$

Where $P(y_{t+1}|\tau_{t+1}, c_t)$ is a distribution of entity words of type τ_{t+1}. As in a general LM, it is computed by projecting the hidden state of the LSTM and normalizing through softmax (eq. 4). The type-specific projection matrix $\boldsymbol{W}^{p,j}$ is learned during training.

We maintain a type embedding matrix $\boldsymbol{W}^e$ and use it in a similar manner to compute the probability that the next word has a given type $P(\tau_{t+1}|c_t)$ (eq. 5). The only difference is that we use an extra projection matrix, $\boldsymbol{W}^h$ to project $\boldsymbol{h}_t$ into lower dimensions. Figure 1a illustrates visually the architecture of KALM.

$$P(y_{t+1} = i|\tau_{t+1} = j, c_t) = \frac{\exp(\boldsymbol{W}^{p,j}_{i,:} \cdot \boldsymbol{h}_t)}{\sum_{w=1}^{|V_j|} \exp(\boldsymbol{W}^{p,j}_{w,:} \cdot \boldsymbol{h}_t)} \tag{4}$$

$$P(\tau_{t+1} = j|c_t) = \frac{\exp(\boldsymbol{W}^e_{j,:} \cdot (\boldsymbol{W}^h \cdot \boldsymbol{h}_t))}{\sum_{k=0}^{K} \exp(\boldsymbol{W}^e_{k,:} \cdot (\boldsymbol{W}^h \cdot \boldsymbol{h}_t))} \tag{5}$$

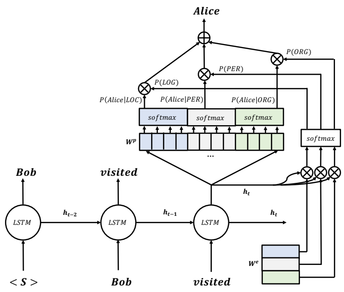
3.3 Type representation as input

In the base KALM model the input for word y_t consists of its embedding vector $\boldsymbol{y}_t$. We enhance the base model by adding as inputs the embedding of the type of the previous word. As type information is latent, we represent it as the weighted sum of the type embeddings weighted by the predicted probabilities:

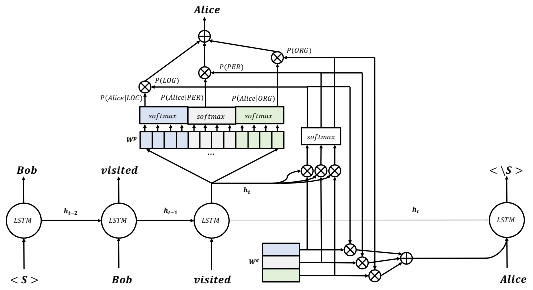
$$\boldsymbol{\nu}_{t+1} = \sum_{j=0}^{K} P(\tau_{t+1} = j|c_t) \cdot \boldsymbol{W}^e_{j,:} \tag{6}$$

$$\tilde{\boldsymbol{y}}_{t+1} = [\boldsymbol{y}_{t+1}; \boldsymbol{\nu}_{t+1}] \tag{7}$$

$P(\tau_{t+1} = j|c_t)$ is computed using Equation 5 and e_j is the type embedding vector.



(a) Basic model architecture of KALM.



(b) Adding type representation as input to KALM.

Figure 1: KALM's architectures

Adding type information as input serves two purposes: in the forward direction, it allows KALM to model context more precisely based on predicted entity types. During back propagation, it allows us to learn latent types more accurately based on subsequent context. The model enhanced with type input is illustrated in Figure 1b.

4 Unsupervised NER

The type distribution that KALM learns is latent, but we can output it at test time and use it to predict whether a given word refers to an entity or a general word. We compute $P(\tau_{t+1}|c_t)$ using eq. 5 and use the most likely entity type as the named entity tag for the corresponding word y_{t+1}.

This straightforward approach, however, predicts the type based solely on the left context of the tag being predicted. In the following two subsections, we discuss extensions to KALM that allow it to utilize the right context and the word being predicted itself.

4.1 Bidirectional LM

While we cannot use a bidirectional LSTM for generation, we can use one for NER, since the entire sentence is known.

For each word, KALM generates the hidden vectors $\boldsymbol{h}_{l,t}$ and $\boldsymbol{h}_{r,t}$ representing context coming from left and right directions, as shown in Equations 8 and 9.

$$\boldsymbol{h}_{l,t}, \boldsymbol{\gamma}_{l,t} = lstm_l(\boldsymbol{h}_{l,t-1}, \boldsymbol{\gamma}_{l,t-1}, [\boldsymbol{y}_{l,t}; \boldsymbol{\nu}_{l,t}]) \quad (8)$$

$$\boldsymbol{h}_{r,t}, \boldsymbol{\gamma}_{r,t} = lstm_r(\boldsymbol{h}_{r,t+1}, \boldsymbol{\gamma}_{r,t+1}, [\boldsymbol{y}_{r,t}; \boldsymbol{\nu}_{r,t}]) \quad (9)$$

We concatenate the hidden vectors from the two directions to form an overall context vector $\boldsymbol{h}_t$, and generate the final type distribution using Equation 5.

Training the bidirectional model requires that we initialize the hidden and cell states from both ends of a sentence. Suppose the length of a sentence is n. The the cross entropy loss is computed for only the $n-2$ symbols in the middle. Similarly, we compute only the types of the $n-2$ symbols in the middle during inference.

4.2 Current word information

Even bidirectional context is insufficient to predict the word type by itself. Consider the following example: *Our computer models indicate Edouard is going to ___*[1] The missing word can be either a location (e.g., *London*), or a general word (e.g., *quit*). In an NER task we observe the underlined words: *Our computer models indicate Edouard is going to London.* In order to learn predictively, we cannot base the type prediction on the current token. Instead, we can use a prior type information $P(\tau_t|y_t)$, pre-computed from entity popularity information available in many KBs. We incorporate the prior information $P(\tau_t|y_t)$ in two different ways described in the two following subsections.

4.2.1 Decoding with type prior

We incorporate the type prior $P(\tau_t|y_t)$ directly by combining it linearly with the predicted type:

$$\begin{aligned} P(\tau_t|c_l, c_r, y_t) =& \frac{P(y_t|\tau_t, c_l, c_r)}{2P(y_t|c_l, c_r)} \cdot P(\tau_t|c_l, c_r) \\ &+ \frac{P(c_l, c_r|\tau_t, y_t)}{2P(c_l, c_r|y_t)} \cdot P(\tau_t|y_t) \\ =& \alpha \cdot P(\tau_t|c_l, c_r) \\ &+ \beta \cdot P(\tau_t|y_t) \end{aligned} \quad (10)$$

The coefficients α and β are free parameters and are tuned on a small amount of withheld data.

---

[1] All the examples are selected from the CoNLL 2003 training set.

4.2.2 Training with type priors

An alternative for incorporating the pre-computed $P(\tau_t|y_t)$ is to use it during training to regularize the type distribution. We use the following optimization criterion to compute the loss for each word:

$$\begin{aligned} L = & H(P(y_i|c_l, c_r), P(\hat{y}_i|c_l, c_r)) \\ & + \lambda \cdot ||KL(P(\tau_i|c_l, c_r), P(\tau_i|y_i))||^2 \end{aligned} \quad (11)$$

where $\hat{y}_i$ is the actual word, $H(.)$ is the cross entropy function, and $KL(.)$ measures the KL divergence between two distributions. A hyperparameter λ (tuned on validation data) controls the relative contribution of the two loss terms. The new loss forces the learned type distribution, $P(\tau_i|c_l, c_r)$, to be close to the expected distribution $P(\tau_i|y_i)$ given the information in the database. This loss is specifically tailored to help with unsupervised NER.

5 Experiments

We evaluate KALM on two tasks: language modeling and NER. We use two datasets: *Recipe* used only for LM evaluation and *CoNLL 2003* used for both the LM and NER evaluations.

5.1 Data

Recipe The recipe dataset[2] is composed of 95,786 recipes, We follow the same preprocessing steps as in Parvez et al. (2018) and divide the crawled dataset into training, validation and testing. A typical sentence after preprocessing looks like the following: "in a large mixing bowl combine the butter sugar and the egg yolks". The entities in the recipe KB are recipe ingredients. The 8 supported entity types are *dairy*, *drinks*, *fruits*, *grains*, *proteins*, *seasonings*, *sides*, and *vegetables*. In the sample sentence above, the entity names are *butter*, *sugar*, *egg* and *yolks*, typed as *dairy*, *seasonings*, *proteins* and *proteins*, respectively.

CoNLL 2003 Introduced in Tjong Kim Sang and De Meulder (2003), the CoNLL 2003 dataset is composed of news articles. It contains text and named entity labels in English, Spanish, German and Dutch. We experiment only with the English version. We follow the CoNLL labels and separate the KB into four entity types: *LOC* (location), *MISC* (miscellaneous), *ORG* (organization), and *PER* (person).

Statistics about the recipe and the CoNLL 2003 dataset are presented in Table 1.

	train	valid	test
#sent	61302	15326	19158
#tok	7223474	1814810	2267797

	train	valid	test
#sent	14986	3465	3683
#tok	204566	51577	46665

Table 1: Statistics of recipe and CoNLL 2003 datasets

The information about the entities in each of the KBs is shown in Table 2. The recipe KB is provided along with the recipe dataset[3] as a conglomeration of typed ingredients. The KB used by CoNLL 2003 is extracted from WikiText-2. We filtered the entities which are not belonging to the 4 types of CoNLL 2003 task.

type	dairy	drinks	fruits	grains
#entities	80	84	110	158

type	proteins	seasonings	sides	vegetables
#entities	316	180	140	156

type	LOC	MISC	ORG	PER
#entity words	1503	1211	3005	5404

Table 2: Statistics of recipe and CoNLL 2003 KBs

5.2 Implementation details

We implement KALM by extending the AWD-LSTM[4] language model in the following ways:

Vocabulary We use the entity words in Table 2 to form $V_1, ..., V_K$ We extract 51,677 general words in the recipe dataset, and 17,907 general words in CoNLL 2003 to form V_0. Identical words that fall under different entity types, such as *Washington* in *George Washington* and *Washington D.C.*, share the same input embeddings.

Model The model has an embedding layer of 400 dimensions, LSTM cell and hidden states of 1,150 dimensions, and 3 stacked LSTM layers. We scale the final LSTM's hidden and cell states to 400 dimensions, and share weights between the projection layer $\boldsymbol{W}^p$ and the word embedding layer. Each entity type in the knowledge base is represented by a trainable 100-dimensional embedding vector. When concatenating the weighted

---

[2] Crawled from `http://www.ffts.com/recipes.htm`

[3] The KB can be found in `https://github.com/uclanlp/NamedEntityLanguageModel`

[4] `https://github.com/salesforce/awd-lstm-lm`

average of the type embeddings to the input, we expand the input dimension of the first LSTM layer to 500. All trainable parameters are initialized uniformly randomly between -0.1 and 0.1, except for the bias terms in the decoder linear layer, which are initialized to 0.

For regularization, we adopt the techniques in AWD-LSTM, and use an LSTM weight dropout rate of 0, an LSTM first-layers locked dropout rate of 0.3, an LSTM last-layer locked dropout rate of 0.4, an embedding Bernoulli dropout rate of 0.1, and an embedding locked dropout rate of 0.65. Also, we impose L2 penalty on LSTM pre-dropout weights with coefficient 1, and L2 penalty on LSTM dropout weights with coefficient 2, both added to the cross entropy loss.

Optimization We use the same loss penalty, dropout schemes, and averaged SGD (ASGD) as in Merity et al. (2017). The initial ASGD learning rate is 10, weight decay rate is 1.2×10^{-6}, non-monotone trigger for ASGD is set to 5, and gradient clipping happens at 0.25. The models are trained until the validation set performance starts to decrease.

5.3 Language modeling

First, we test how good KALM is as a language model compared to two baselines.

5.3.1 Baselines

- **AWD-LSTM** (Merity et al., 2017) is the state-of-the-art word-level language model as measured on WikiText-2 and Penn Treebank. It uses ASGD optimization, a new dropout scheme and novel penalty terms in the loss function to improve over vanilla LSTM LMs.

- **Named-entity LM** (NE-LM) (Parvez et al., 2018) consists of a *type model* that outputs $P(\tau_{i+1}|\tau_i, \tau_{i-1}, ...)$ and an *entity composite model* that outputs $P(y_{i+1}|\{y_i, \tau_i\}, \{y_{i-1}, \tau_{i-1}\}, ...)$. The *type model* is trained on corpora with entity type labels, whereas the *entity composite model* has an input for words and another input for the corresponding types, and so needs to be trained on both the labeled corpus and the unlabeled version of the same corpus. At inference time, a joint inference heuristic aggregates type model and entity composite model predictions into a word prediction. Since both models require type labels as input, each generation step of NE-LM requires not only the previously generated words $[y_i, y_{i-1}, ...]$, but also the type labels for these words $[\tau_i, \tau_{i-1}, ...]$.

5.3.2 Results

For language modeling we report word prediction perplexity on the recipe dataset and CoNLL 2003. Perplexity is defined as the following.

$$PP = e^{-\sum\limits_{t=1}^{N} \frac{1}{N} \log P(y_t)} = \sqrt[N]{\prod_{t=1}^{N} \frac{1}{P(y_t)}} \quad (12)$$

We use publicly available implementations to produce the two baseline results. We also compare the language models in the bidirectional setting, which the reference implementations do not support. In that setting, we transform both models in NE-LM to be bidirectional.

Discussion Table 3 shows that KALM outperforms the two baselines in both unidirectional and bidirectional settings on both datasets. The improvement relative to NE-LM is larger in the unidirectional setting compared to the bidirectional setting. We conjecture that this is because in that setting NE-LM trains a bidirectional NER in a supervised way. The improvement relative to NE-LM is larger on CoNLL 2003 than on the recipe dataset. We believe that the inference heuristic used by NE-LM is tuned specifically to recipes and is less suitable to the CoNLL setting.

We also find that training KALM on more unlabeled data further reduces the perplexity (see Table 4), and study how the quality of the KB affects the perplexity. We discuss both these results in Section 5.4.

5.4 NER

In this section, we evaluate KALM in NER against two supervised baselines.

5.4.1 Baselines

We train two supervised models for NER on the CoNLL 2003 dataset: a **biLSTM** and a **CRF-biLSTM**. We replicate the hyperparameters used by Lample et al. (2016), who demonstrate the state-of-the-art performance on this dataset. We use a word-level model, and 100 dimensional pretrained GloVe embeddings (Pennington et al., 2014) for initialization. We train for 50 epochs, at which point the models converge.

	model	unidirectional		bidirectional	
		validation	test	validation	test
Recipe	AWD-LSTM	3.14	2.99	1.98	2
	NE-LM	2.96	2.24	1.85	1.73
	KALM	**2.75**	**2.20**	**1.85**	**1.71**
CoNLL 2003	AWD-LSTM	5.48	5.94	4.85	5.3
	NE-LM	5.67	5.77	4.68	4.94
	KALM	**5.36**	**5.43**	**4.64**	**4.74**

Table 3: Language modeling results on the recipe and CoNLL 2003 datasets

5.4.2 Results

We evaluate the unsupervised KALM model under the following configurations:

- Basic: bidirectional model with aggregated type embeddings fed to the input at the next time step;
- With type priors: using $P(\tau_t|y_t)$ in the two ways described in Section 4.2;
- Extra data: Since KALM is unsupervised, we can train it on extra data. We use the WikiText-2 corpus in addition to the original CoNLL training data.

WikiText-2 is a standard language modeling dataset released with Merity et al. (2016). It contains Wikipedia articles from a wide range of topics. In contrast, the CoNLL 2003 corpus consists of news articles. Table 4 show statistics about the raw / characer level WikiText-2 and the CoNLL 2003 corpora. Despite the domain mismatch between the WikiText and CoNLL corpora, the WikiText coverage of the entity words that exist in the CoNLL dataset is high. Specifically, most of the person, location and organization entity words that appear in CoNLL either have a Wikipedia section, or are mentioned in a Wiki article. Therefore, we expect that the addition of WikiText can guide the unsupervised NER model to learn better entity type regularities. Indeed, the result presented in the rightmost column of Table 4 shows that when adding WikiText-2 to CoNLL 2003, the perplexity for the KALM model for the news text of CoNLL 2003 is decreased: from 4.69 down to **2.29**.

We show NER results in Table 5. The table lists the F1 score for each entity types, as well as the overall F1 score.

Discussion Even the basic KALM model learns context well – it achieves an overall F1 score of 0.72 for NER. This illustrates that KALM has learned to model entity classes entirely from surrounding context. Adding prior information as to whether a word represents different entity types helps to bring the F1 score to 0.76.

The strength of an unsupervised model is that it can be trained on large corpora. Adding the Wikitext-2 corpus improves the NER score of KALM to 0.84.

To give a sense of how the unsupervised models compare with the supervised model with respect to training data size, we trained biLSTM and CRF-biLSTM on a randomly sampled subset of the training data of successively decreasing sizes. The resulting F1 scores are shown in Figure 2.

Our best model scores 0.86, same as a CRF-biLSTM trained on around 40% of the training data. It is less than 0.03 behind the best supervised CRF-biLSTM. The best KALM model almost always scores higher than biLSTM without the CRF loss.

Lastly, we perform an ablation experiment to gauge how sensitive KALM is to the quality of the knowledge base. Previous studies (Liu et al., 2016; Zhang et al., 2012) have shown that the amount of knowledge retrieved from KBs can impact the performance of NLP models such as relation extraction systems substantially. In this experiment, we deliberately corrupt the entity vocabularies $V_0, ..., V_{K-1}$ by moving a certain percentage of randomly selected entity words from V_i to the general vocabulary V_g. Figure 3 shows language modeling perplexities on the validation set, and NER F1 scores on the test set as a function of the corruption percentage. The language modeling performance stops reacting to KB corruption beyond a certain extent, whereas the NER performance keeps dropping as the number of entities removed from $V_1, V_2, ...$ increases. This result shows the importance of the quality of the KB

entity ratio	unique entity ratio	size ratio	LM perplexity
92.80%	82.56%	2.62	2.29 : 4.69

Table 4: Characterization of WikiText-2 relative to CoNLL 2003 training set. *Entities extracted from WikiText cover* 92.80% *of the entities in CoNLL 2003 overall, and cover* 82.56% *of the unique entities. WikiText's size is* 2.62 *times as large. And adding WikiText to CoNLL training reduces the perplexity from* 4.69 *to* 2.29.

		LOC	MISC	ORG	PER	overall
unsupervised	basic^0	0.75	0.67	0.64	0.83	0.72
	+$P(\tau\|y)$ in dec^1	0.81	0.67	0.65	0.88	0.76
	+wiki-concat^2	0.83	0.83	0.76	0.95	0.84
	+wiki-concat+$P(\tau\|y)$ in dec^3	0.84	**0.84**	0.77	**0.96**	0.86
supervised	biLSTM	0.88	0.71	0.81	0.95	0.86
	CRF-biLSTM	**0.90**	0.76	**0.84**	0.95	**0.89**

Table 5: Results of KALM (above the double line in the table) and supervised NER models (under the double line). Superscript annotations: 0*: The basic bidirectional KALM with type embedding features as described in Section 4.1.* 1*: Adding* $P(\tau|y)$ *in decoding, as described in Section 4.2.1, where* α *and* β *are tuned to be* 0.4 *and* 0.6. 2*: The basic model trained on CoNLL 2003 concatenated with WikiText-2.* 3*: Adding* $(\tau|y)$ *in decoding, with the model trained on CoNLL 2003 concatenated with WikiText-2.*

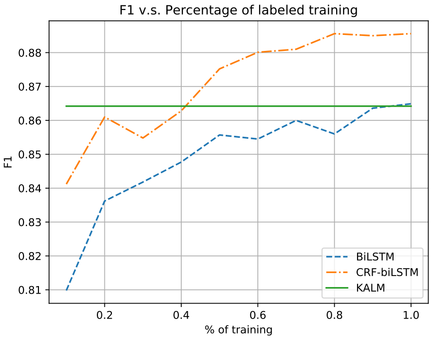


Figure 2: Unsupervised v.s. supervised NER trained on different portions of training data

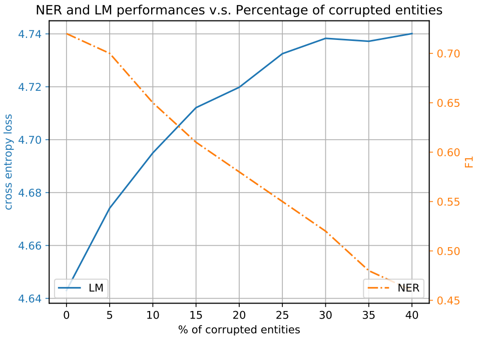


Figure 3: NER and LM performances on different portions of mislabeled entities

6 Conclusion

We propose Knowledge Augmented Language Model (KALM), which extends a traditional RNN LM with information from a Knowledge Base. We show that real-world knowledge can be used successfully for natural language understanding by using a probabilistic extension. The latent type information is trained end-to-end using a predictive objective without any supervision. We show that the latent type information that the model learns can be used for a high-accuracy NER system. We believe that this modeling paradigm opens the door for end-to-end deep learning systems that can be enhanced with latent modeling capabilities and trained in a predictive manner end-to-end. In ways this is similar to the attention mechanism in machine translation where an alignment mechanism is added and trained latently against the overall translation perplexity objective. As with our NER tags, machine translation alignments are empirically observed to be of high quality.

In future work, we look to model other types of world knowledge beyond named entities using predictive learning and training on large corpora of text without additional information, and to make KALM more robust against corrupted entities.

to KALM.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. 2018. Language modeling with sparse product of sememe experts. *arXiv preprint arXiv:1810.12387*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H Lin, Xiao Ling, and Daniel S Weld. 2016. Effective crowd annotation for relation extraction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 897–906.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Robert Munro and Christopher D Manning. 2012. Accurate unsupervised joint named-entity extraction from unaligned parallel text. In *Proceedings of the 4th Named Entity Workshop*, pages 21–29. Association for Computational Linguistics.

David Nadeau, Peter D Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 266–277. Springer.

Md Rizwan Parvez, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2018. Building language models for text with named entities. *arXiv preprint arXiv:1805.04836*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Put it back: Entity typing with language model enhancement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 993–998.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1436–1446.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *arXiv preprint arXiv:1611.01628*.

Ce Zhang, Feng Niu, Christopher Ré, and Jude Shavlik. 2012. Big data versus the crowd: Looking for relationships in all the right places. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 825–834. Association for Computational Linguistics.

Syntax-Enhanced Neural Machine Translation with Syntax-Aware Word Representations

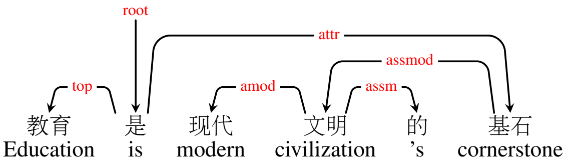
Meishan Zhang[1] and **Zhenghua Li**[2] and **Guohong Fu**[3]* and **Min Zhang**[2]
1. School of New Media and Communication, Tianjin University, China
2. School of Computer Science and Technology, Soochow University, China
3. Institute of Artificial Intelligence, Soochow University, China
`mason.zms@gmail.com,`
`{zhli13, minzhang}@suda.edu.cn,`
`ghfu@hotmail.com`

Abstract

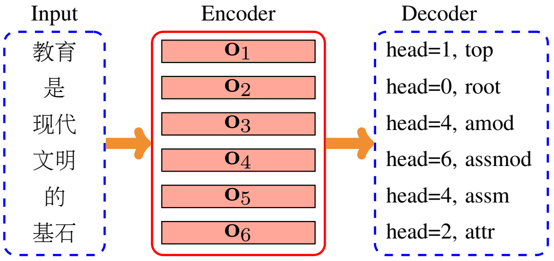
Syntax has been demonstrated highly effective in neural machine translation (NMT). Previous NMT models integrate syntax by representing 1-best tree outputs from a well-trained parsing system, e.g., the representative Tree-RNN and Tree-Linearization methods, which may suffer from error propagation. In this work, we propose a novel method to integrate source-side syntax implicitly for NMT. The basic idea is to use the intermediate hidden representations of a well-trained end-to-end dependency parser, which are referred to as syntax-aware word representations (SAWRs). Then, we simply concatenate such SAWRs with ordinary word embeddings to enhance basic NMT models. The method can be straightforwardly integrated into the widely-used sequence-to-sequence (Seq2Seq) NMT models. We start with a representative RNN-based Seq2Seq baseline system, and test the effectiveness of our proposed method on two benchmark datasets of the Chinese-English and English-Vietnamese translation tasks, respectively. Experimental results show that the proposed approach is able to bring significant BLEU score improvements on the two datasets compared with the baseline, 1.74 points for Chinese-English translation and 0.80 point for English-Vietnamese translation, respectively. In addition, the approach also outperforms the explicit Tree-RNN and Tree-Linearization methods.

1 Introduction

In the past few years, neural machine translation (NMT) has drawn increasing interests due to its simplicity and promising performance (Bahdanau et al., 2014; Jean et al., 2015; Luong and Manning, 2015; Luong et al., 2015; Shen et al., 2016; Vaswani et al., 2017). The widely used sequence-to-sequence (Seq2Seq) framework combined with attention mechanism achieves significant improvement over the traditional statistical machine translation (SMT) models on a variety of language pairs, such as Chinese-English (Shi et al., 2016; Mi et al., 2016; Vaswani et al., 2017; Cheng et al., 2018). Under an encoder-decoder architecture, the Seq2Seq framework first encodes the source sentence into a sequence of hidden vectors, and then incrementally predicts the target sentence (Cho et al., 2014a).



• An example of input dependency tree.



• SAWRs, where the encoder outputs are used as inputs for NMT similar to source-side word embeddings.

Figure 1: An example to illustrate our method of encoding source dependency syntax, where the English translation is “Education is the cornerstone of modern civilization” for the source Chinese input.

Recently, inspired by the success of syntax-based SMT (Williams et al., 2016), researchers propose a range of interesting approaches for exploiting syntax information in NMT models, as syntactic trees could offer long-distance relations in sentences (Shi et al., 2016; Wu et al., 2017b; Li

---

[*] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 1151–1161
Minneapolis, Minnesota, June 2 - June 7, 2019.

et al., 2017; Bastings et al., 2017; Hashimoto and Tsuruoka, 2017).

As a straightforward method, tree-structured recurrent neural network (**Tree-RNN**) can elegantly model the source-side syntax and globally encode the whole trees. Eriguchi et al. (2016), Chen et al. (2017a) and Yang et al. (2017) show that Tree-RNN can effectively integrate syntax-oriented trees into Seq2Seq NMT models.

Regardless of the effectiveness of Tree-RNN, we find that it suffers from a severe low-efficiency problem because of the heterogeneity of different syntax trees, which leads to increasing difficulties for batch computation compared with sequential inputs. Even with deliberate batching method of Neubig et al. (2017), our preliminary experiments show that Tree-RNN with gated recurrent unit (GRU) can lead to nearly four times slower performance when it is integrated into a classical Seq2Seq system.

To solve the problem, **Tree-Linearization** is a good alternative for syntax encoding. The main idea is to linearize syntax trees into sequential symbols, and then exploit the resulting sequences as inputs for NMT. Li et al. (2017) propose a depth-first method to traverse a constituent tree, converting it into a sequence of symbols mixed with sentential words and syntax labels. Similarly, Wu et al. (2017b) combine several strategies of tree traversing for dependency syntax integration.

In this work, we present an implicit syntax encoding method for NMT, enhancing NMT models by syntax-aware word representations (**SAWRs**). Figure 1 illustrates the basic idea, where trees are modeled indirectly by sequential vectors extracted from an encoder-decoder dependency parser. On the one hand, the method avoids the structural heterogeneity and thus can be integrated efficiently, and on the other hand, it does not require discrete 1-best tree outputs, alleviating the error propagation problem induced from syntax parsers. Concretely, the vector outputs are extracted from the encoding outputs of the encoder-decoder dependency parser. As shown in Figure 1, the encoding outputs, denoted as $\boldsymbol{o} = \boldsymbol{o}_1 \cdots \boldsymbol{o}_6$, are then integrated into Seq2Seq NMT models by directly concatenated with the source input word embeddings after a linear projection.

We start with a Seq2Seq baseline with attention mechanism (Bahdanau et al., 2014) for study, following previous studies of the same research line, and then integrate source dependency syntax by SAWRs. We conduct experiments on Chinese-English and English-Vietnamese translation tasks, respectively. The results show that our method is very effective in source syntax integration. With source dependency syntax, the performances of Chinese-English and English-Vietnamese translation can be significantly boosted by 1.74 BLEU points and 0.80 BLEU points, respectively. We also compare the method with the representative Tree-RNN and Tree-Linearization approaches of syntax integration, finding that our method is able to achieve larger improvements than the two approaches for both tasks. All the codes are released publicly available at https://github.com/zhangmeishan/SYN4NMT under Apache License 2.0.

2 Baseline

We take the simple yet effective Seq2Seq model with attention mechanism proposed by Luong et al. (2015) as our baseline. Under the standard encoder-decoder architecture, an encoder first maps the source-language input sentence into a sequence of hidden vectors, and a decoder then incrementally predicts the target output sentence. In particular, we should notice that several recent models (Vaswani et al., 2017; Zheng et al., 2017; Cheng et al., 2018) which have been shown to be more powerful can also serve as our baseline, since these models focus on very different aspects of NMT, which could be potentially complementary with our focus of syntax integration. We will demonstrate it by experimental analysis as well.

2.1 Encoder

In the encoder part, a single-layer bi-directional recurrent neural network (Bi-RNN) is employed to encode the sentence in order to capture features from the current word and the unbounded left and right contextual words. Given a source-language input sentence $\boldsymbol{x} = x_1 \cdots x_n$ and its embedding sequence $\boldsymbol{e}^{x_1} \cdots \boldsymbol{e}^{x_n}$, the Bi-RNN produces an encoding sequence of dense vectors $\boldsymbol{h} = \boldsymbol{h}_1 \cdots \boldsymbol{h}_i \cdots \boldsymbol{h}_n$:

$$\begin{aligned} \boldsymbol{h}_i &= \overrightarrow{\boldsymbol{h}}_i \oplus \overleftarrow{\boldsymbol{h}}_i, \\ \overrightarrow{\boldsymbol{h}}_i &= \mathtt{rnn}^{\mathrm{L}}(\boldsymbol{e}^{x_i}, \overrightarrow{\boldsymbol{h}}_{i-1}) \\ \overleftarrow{\boldsymbol{h}}_i &= \mathtt{rnn}^{\mathrm{R}}(\boldsymbol{e}^{x_i}, \overleftarrow{\boldsymbol{h}}_{i+1}) \end{aligned} \tag{1}$$

where $\mathtt{rnn}^{\mathrm{L}/R}$ can be either GRU (Cho et al., 2014b) or LSTM. We use GRU all through this paper for efficiency following Chen et al. (2017a).

2.2 Decoder

The decoder part incrementally predicts the target word sequence $\boldsymbol{y} = y_1 \cdots y_m$, whose translation probability is defined as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{m} p(y_j|y_1 \cdots y_{j-1}, \boldsymbol{h}). \tag{2}$$

The training objective is to maximize the probability of the reference translation. During evaluation, we aim to search for a target sentence with the highest probability for a given source sentence.

The probability of the j-th target word is computed by a two-layer feed-forward neural network:

$$p(y_j|y_1 \cdots y_{j-1}, \boldsymbol{h}) = g(\boldsymbol{s}_{j-1}, \boldsymbol{c}_j), \tag{3}$$

where $\boldsymbol{s}_{j-1} = \mathtt{rnn}^{\text{tgt}}(\boldsymbol{e}^{y_{j-1}} \oplus \boldsymbol{c}_{j-1}, \boldsymbol{s}_{j-2})$ is the output of a left-to-right RNN over the predicted words, and the $\boldsymbol{c}_j/\boldsymbol{c}_{j-1}$ is the weighted sum over the encoding sequence $\boldsymbol{h}$ of the source sentence via the attention mechanism, which is computed as follows:

$$\begin{aligned} \boldsymbol{c}_j &= \sum_{k=1}^{n} \alpha_{j,k} \boldsymbol{h}_k \\ \alpha_{j,k} &= \frac{\exp(\beta_{j,k})}{\sum_{l=1}^{n} \exp(\beta_{j,l})} \\ \beta_{j,l} &= \boldsymbol{s}_{j-1}^{\mathrm{T}} \boldsymbol{W}^a \boldsymbol{h}_l \end{aligned} \tag{4}$$

where $\boldsymbol{W}^a$ is the model parameter in attention.

3 Our Method

Syntax information has been demonstrated to be valuable for NMT. Previously, there were two representative approaches to encode syntax into an NMT model. The first approach directly represents an input syntax tree by **Tree-RNN**, and then uses the Tree-RNN outputs as additional encoder inputs for NMT. The second approach models source syntax trees indirectly by first converting a hierarchical tree into a sequence of symbols, and then use the symbols as inputs for NMT. The second method is referred to as **Tree-Linearization** here.

Tree-RNN is able to represent the syntax structures fully and comprehensively. However, because of the heterogeneity of different syntax trees, this approach suffers serious inefficiency problem as the increased difficulty of batch computation for GPU neural computation. The second approach exploits an alternative sequence to substitute the original trees, which solves the inefficiency problem. But it may bring loss of syntax information because the hierarchical tree structure is no longer maintained in the new representation, which could be potentially useful for NMT.

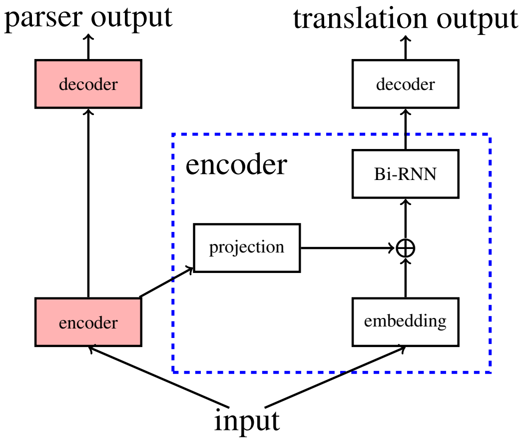


Figure 2: The framework of the SAWR approach, where the left part shows the encoder-decoder of a supervised dependency parsing model and the right part shows the NMT encoder-decoder.

Both the two syntax integration approaches are based on discrete 1-best outputs of a supervised dependency parser, which may suffer from the error propagation problem. Incorrect syntax trees as inputs for NMT may produce erroneous outputs, leading to inappropriate translation results. In order to alleviate the problem, we present a novel method not using the discrete parsing outputs.

We focus on supervised dependency parsing models which can be formalized as an encoder-decoder architecture, and exploit the encoder outputs as the inputs for our Seq2Seq NMT model. The encoder outputs are sequences of dense vectors aligning with the source sentential words, as shown in Figure 1, and thus they could be easily combined with the encoder part of our NMT model. We refer to this method as **SAWR** for brief. Our approach takes the implicit hidden outputs from a supervised parser as inputs for NMT, which greatly reduces the direct influence brought from discrete 1-best parser outputs.

Figure 2 shows the framework of SAWR. Concretely, we first project the encoder outputs of a dependency parsing model into a sequence of vectors by a feed-forward linear layer, as shown by the projection module in Figure 2:

$$\boldsymbol{s}_i = \boldsymbol{W}\boldsymbol{o}_i + \boldsymbol{b} \quad (5)$$

where $\boldsymbol{o}{=}\boldsymbol{o}_1 \cdots \boldsymbol{o}_n$ is the encoder output of a parsing model, $\boldsymbol{W}$ and $\boldsymbol{b}$ are model parameters. Then we concatenate the resulting vectors with the source embeddings as inputs for the baseline Bi-RNN Encoder. Thus the encoder process can be formalized as follows:

$$\boldsymbol{h} = \text{Bi-RNN}\big(\boldsymbol{e}^{x_1} \oplus \boldsymbol{s}_1, \cdots, \boldsymbol{e}^{x_n} \oplus \boldsymbol{s}_n\big). \quad (6)$$

Noticeably, the SAWR method can be regarded as an adaption of joint learning as well. We can train both dependency parsing and machine translation model parameters concurrently. In this work, we focus on the machine translation task and do not involve the training objective of dependency parsing. However, we can still finetune model parameters of the encoder part of dependency parsing by back-propagating the training losses of NMT into this part as well.

Actually, SAWRs are also similar to the ELMO embeddings (Peters et al., 2018). ELMO learns context word representations by using language model as objective, while SAWRs learn syntax-aware word representations by using dependency parsing as objective. On the other hand, compared with the Tree-RNN and Tree-Linearization methods which encode syntax trees by neural networks directly, SAWRs are less sensitive to the output syntax trees. Thus the SAWR method can alleviate the error propagation problem.

4 Experiments

4.1 Settings

Data. We conduct experiments on the Chinese-English and English-Vietnamese translation tasks, respectively. For Chinese-English, we use the parallel training data from the publicly available LDC corpora,[1] with 28.3M Chinese words and 34.5M English words, respectively, consisting of 1.25M sentence pairs, and test model performances on the NIST datasets, using NIST MT02 as the development data, and MT03-06 as test datasets. For English-Vietnamese, we use the standard IWSLT 2015 dataset,[2] which consists of about 133K sentence pairs, and evaluate our models by exploiting the TED tst2012 and tst2013 as the development and test datasets, respectively.

For the source side sentences, we construct vocabularies of the most frequent 50K words, while for the target side sentences, we apply byte-pair encodings (BPE) (Sennrich et al., 2016) with 32K merges to obtain subword units, and construct the target vocabularies by the most frequent 32K subwords. During training, we use only the sentence pairs whose source and target lengths both are no longer than 50 and 150 for Chinese-English and English-Vietnamese translations, respectively.

Evaluation. We use the case insensitive 4-gram BLEU score as the main evaluation metrics (Papineni et al., 2002), and adopt the script `multi-bleu.perl` in the Mose toolkit.[3] Significance tests are conducted based on the best-BLEU results for each approach by using bootstrap resampling (Koehn, 2004).

Alternatively, in order to compare the effectiveness of our model with other syntax integration methods, we implement a **Tree-RNN** approach and a **Tree-Linearization** approach, respectively:

- Tree-RNN: We build a one-layer bi-directional Tree-RNN with GRU over input word embeddings, producing syntax-enhanced word representations, which are then fed into the encoder of NMT as basic inputs. The method is similar to the model proposed by Chen et al. (2017a).
- Tree-Linearization: We first convert dependency trees into constituent trees (Sun and Wan, 2013), and then feed it into the NMT model proposed by Li et al. (2017).

Hyperparameters. We set the dimension sizes of all hidden neural layers to 1024, except the input layers for RNNs (i.e. input word embeddings and the projection layer of SAWR), which are set to 512. We initialize all model parameters by random uniform distribution between $[-0.1, 0.1]$. We apply dropout on the output layer of word translation with a ratio of 0.5.

We adopt the Adam algorithm (Kingma and Ba, 2014) for parameter optimization, with the initial learning rate of 5×10^{-4}, the gradient clipping threshold of 5, and the mini-batch size of 80. During translation, we employ beam search for decoding with the beam size of 5.

---

[1] LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[2] https://nlp.stanford.edu/projects/nmt/

[3] http://www.statmt.org/moses

System	MT03	MT04	MT05	MT06	Average/Δ
Baseline	36.44	39.35	36.26	36.32	37.09
SAWR	**38.42**	**40.60**	**38.27**	**38.04**	**38.83/+1.74**
Tree-RNN	38.12	40.35	37.86	37.32	38.41/+1.32
Tree-Linearization	37.95	40.24	37.64	37.44	38.32/+1.23
Previous Work					
Chen et al. (2017a)	35.64	36.63	34.35	30.57	34.30/**+2.59**
Li et al. (2017)	34.9	38.6	**35.5**	**35.6**	**36.15**/+1.45
Chen et al. (2017b)	**35.91**	**38.73**	34.18	33.76	35.65/+1.52

Table 1: Final results of Chinese-English translation. All syntax-integrated approaches are significantly better than the baseline system ($p < 0.05$).

Source-Side Parsing. We employ the state-of-the-art BiAffine dependency parser recently proposed by Dozat and Manning (2016) to obtain the source-side dependency syntax information. The BiAffine parser can also be understood as an encoder-decoder model, where the encoder part is a three-layer bi-directional LSTM over the input words, and the decoder uses BiAffine operations to score all candidate dependency arcs and finds the highest-scoring trees via dynamic programming.

For Chinese-English translation, we train the dependency parser on Chinese Treebank 7.0 with Stanford dependencies,[4] using 50K random sentences as the training data and the remaining as the test data. The parser achieves 81.02% parsing accuracy (labeled attached score, LAS) on the test dataset. For English-Vietnamese translation, we train the dependency parser on English WSJ corpus, following the same data split as Dozat and Manning (2016), and obtaining a LAS of 93.84% on the test dataset.[5]

4.2 Speed Comparison

All our experiments are run on a single GPU NVIDIA TITAN Xp. We report the averaged one-epoch training time on the Chinese-English translation dataset (consuming all 125M sentence pairs) as follows:

Baseline	105 min
SAWR	142 min
Tree-RNN	498 min
Tree-Linearization	137 min

The SAWR system spends averaged 142 minutes,[6] 37 minutes slower than the baseline model. The Tree-Linearization spends averaged 137 minutes per epoch, which is the fastest syntax integration method. Our SAWR approach spends 5 more minutes than Tree-Linearization, appropriate 3.5% of the total spend time per epoch, which could be negligible. The Tree-RNN model spends 498 minutes per epoch, nearly four times slower than the baseline model.[7] According to the results, we can conclude that the Tree-RNN model is highly inefficient for encoding dependency syntax, whereas the SAWR and Tree-Linearization are almost as efficient as the baseline Seq2Seq system.

4.3 Main Results

4.3.1 Chinese-English Translation

Table 1 shows the main results of all approaches on Chinese-English datasets. Considering the effect of random initialization, we train three individual models for each approach, and use the averaged BLEU scores for fair comparisons.

According to the results, we can see that all syntax-integrated approaches can bring significant improvements over the baseline system, which denotes that syntax is highly effective for Chinese-English machine translation. In addition, the proposed SAWR approach obtains the largest BLEU improvements, averaged $\Delta = 1.74$ BLEU points better than the baseline system. The Tree-RNN and Tree-Linearization approaches bring improve-

---

[4] https://nlp.stanford.edu/software/stanford-dependencies.shtml

[5] For simplicity, we use only words as inputs for both Chinese and English dependency parsing, avoiding the influences brought by other inputs, such as automatic POS tags.

[6] We exclude the time consumed by the encoder part of the dependency parsing model for fair comparisons, as other methods require to perform parsing in an offline way.

[7] The Tree-RNN model is implemented with deliberate batching motivated by Neubig et al. (2017), without which the model is intolerably slow, reaching about 1,900 minutes per epoch.

System	tst 2013 / Δ
Baseline	28.29
SAWR	**29.09/+0.80**
Tree-RNN	28.51/+0.22
Tree-Linearization	28.93/+0.64

Table 2: Final Results on the IWSLT 2015 English-Vietnamese translation task. Only SAWR is significantly better than the baseline system ($p < 0.05$).

ments of averaged $\Delta = 1.32$ and $\Delta = 1.23$ BLEU points, respectively. The results show that our implicit syntax-aware encoding method is better than Tree-RNN and Tree-Linearization.

We compare our NMT models with other state-of-the-art methods as well. The results are just for reference since experimental details could be very different. In particular, we list the relative improvements over the corresponding baseline models by integrating syntax structures, which are calculated according to their papers. All these studies exploit lower baselines compared with our models. The Tree-RNN and Tree-Linearization are essentially similar to Chen et al. (2017a) and Li et al. (2017), respectively. As shown, our approaches can still obtain large improvements based on a stronger baseline.

4.3.2 English-Vietnamese Translation

Table 2 shows the final results on the IWSLT 2015 English-Vietnamese translation task. The overall tendency is similar to that of Chinese-English translation. The syntax information can boost the translation performances by using any of the three approaches. The SAWR approach gives the best translation performance, significantly outperform the baseline system by $\Delta = 0.80$ BLEU points. While although the other two approaches bring better performances, the improvements are not significant. The results demonstrate the advantage of the proposed implicit SAWR approach. By not using the 1-best parser outputs, our approach can reduce the error propagation problem, thus bring larger improvements with syntax.

In particular, we find that the increases of BLEU scores are smaller than that of Chinese-English translation by integrating syntactic features. The averaged BLEU increases are 0.55 for English-Vietnamese and 1.43 for Chinese-English. The possible reason may be due to that the source English sentences are more grammatically rigorous than Chinese sentences. For example, the English functional words such as “of” and “‘s” which indicate the possessive relationship, should be always kept in sentences by standard, while their Chinese correspondence “的” may be omitted in sentences.

Parser	MT03	MT04	MT05	MT06	Average
no Tune	**38.42**	**40.60**	**38.27**	**38.04**	**38.83**
Tune	37.33	39.45	36.93	37.03	37.69

Table 3: The influence of fine-tuning parser parameters in the SAWR system.

4.4 Analysis

In this section, we conduct analysis on Chinese-English translation from different aspects to better understand the SAWR approach of integrating source-side dependency syntax for NMT.

4.4.1 Fine-Tuning Syntax-Oriented Inputs

The SAWR approach directly uses the encoder outputs of a dependency parser as extra inputs for NMT. In the above experiments, we keep the parser model parameters fixed, letting them uninfluenced from NMT optimization. Actually, this part can be further fine tuned along with the NMT learning, by treating them as one kind model parameters. Thus there arises a question that whether fine-tuning the parser model parameters can bring better performance.

As an interesting attempt, we can simultaneously fine tune the parameters of both the parser and the Seq2Seq NMT model during training. Figure 3 shows the results. We can see that fine-tuning decreases the average BLEU score by $38.83 - 37.69 = 1.14$ significantly. This may be because that fine-tuning disorders the representation ability of the parser and makes its function more overlapping with other network components. This further demonstrates that pretrained syntax-aware word representations are helpful for NMT.

4.4.2 Alignment Study

Alignment quality is an important metric to illustrate and evaluate machine translation outputs. Here we study how syntax features influence the alignment results for NMT. We approximate the alignment scores by the attention probabilities as shown in Equation 4.[8] For better understanding

---

[8] We aim to offer an intuitive interpretation by a carefully-selected example. In fact, the alignment computation method here may be problematic (Koehn and Knowles, 2017).

System	MT03	MT04	MT05	MT06	Average/Δ
Baseline×3	40.90	43.25	40.64	40.16	41.24
SAWR×3	41.94	44.59	41.91	41.97	42.60/+1.36
Tree-RNN×3	42.03	44.15	41.50	41.41	42.27/+1.03
Tree-Linearization×3	41.74	44.23	41.32	41.44	42.18/+0.94
Hybrid	**42.72**	**45.14**	**42.38**	**42.15**	**43.10/+1.86**

Table 4: Ensemble performances, where the Hybrid model denotes SAWR + Tree-RNN + Tree-Linearization.

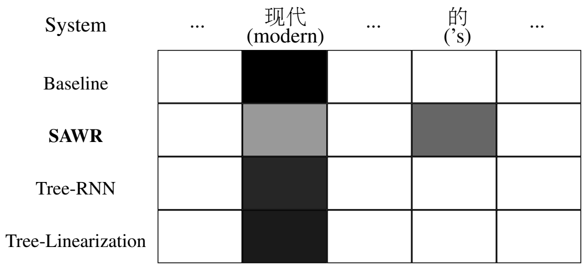


Figure 3: Alignments for the baseline and syntax-integrated systems, where the same example in Figure 1 is analyzed and the target English word is "of".

the effectiveness of syntax, we choose the target-side English word "of" for comparison, which is a grammatical functional word.

Figure 3 shows the alignment probability distributions returned by different approaches. Intuitively, this word should be aligned with the Chinese word "的(de)". But according to the results, we can see that only the SAWR model distributes a high attention score to it, which is consistent with our intuition. The other three models are all aligned to the source word "现代 (modern)" with high confidence over 85%. The possible reason for "of" being aligned to "现代 (modern)" could be due to that "of modern" is a high-frequency collocation in the training corpora.

4.4.3 Ensemble Study

Here we perform model ensembles to examine the divergences of the three syntax-integration approaches (Zhou et al., 2017b; Denkowski and Neubig, 2017). Intuitively, the hetero-approach ensemble which combines three NMT models of different methods should obtain better performances than homo-approach ensembles which combine three NMT models of the same method, since NMT models of different syntax-integrations approaches have larger divergences.

Table 4 shows the results. First, we can see that ensemble is one effective technique to improve the translation performances. More importantly, the results show that the heterogeneous ensemble achieves averaged BLEU improvements by $43.10 - 41.24 = 1.86$ points, better than the gains achieved by all three homo-approach ensembles, denoting that the three approaches could be mutually complementary in representing dependency syntax, and the resulting models of the three approaches are highly diverse.

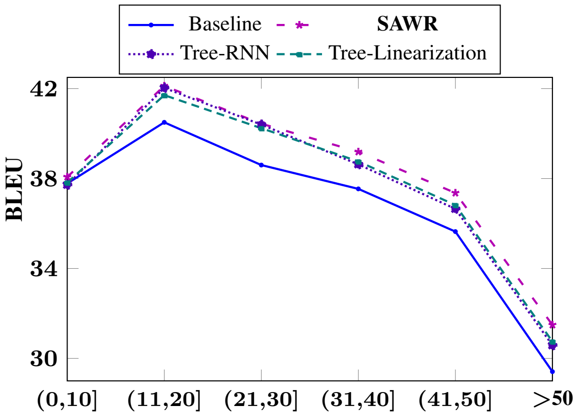


Figure 4: The effect of source input length.

4.4.4 Analysis by Source Sentence Length

Intuitively, by introducing the source syntax into the NMT model, relations between long-distance words are explicitly modeled by dependency trees, thus we can expect that models enhanced by source syntax are able to bring better translations for longer sentences. Figure 4 shows the performances of the baseline and all syntax-enriched models in terms of source sentence lengths, where we bin all the MT03-MT06 sentences by their lengths into six intervals. The results show that the BLEU scores are improved significantly when source sentential lengths are over 10, which confirms our intuition.

4.4.5 Effect of Parsing Performance

Finally, we examine how the performance of the dependency parser influences the final translation quality. While the full dependency parser is

System	MT03	MT04	MT05	MT06	Average/Δ
Transformer	40.45	42.76	40.09	39.67	40.74
SAWR	**41.63**	**43.60**	**41.68**	**40.21**	**41.78/+1.04**
Tree-RNN	41.24	43.38	41.04	40.02	41.42/+0.68
Tree-Linearization	41.12	43.02	41.04	39.86	41.26/+0.52

Table 5: Final results based on the transformer. Only the SAWR results are significantly better ($p < 0.05$).

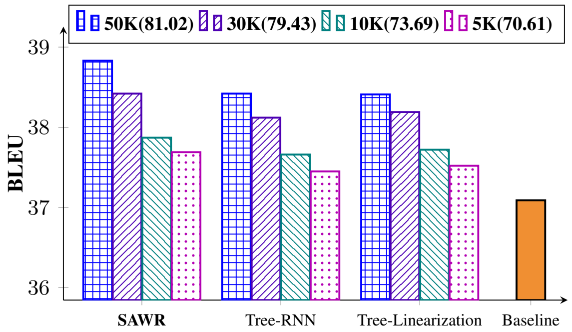


Figure 5: The effect of dependency parsing performances on our proposed approaches.

trained on 50K sentences, we retrain three weaker dependency parsers on 30K, 10K and 5K sentences, respectively. Figure 5 shows the NMT BLEU scores and the parsing accuracies. It is clear that the parsing accuracy directly influences the translation quality, indicating the effectiveness and importance of exploiting syntactic information.

4.4.6 Transformer as Baseline

Here we conduct experiments based on the transformer NMT model (Vaswani et al., 2017), which is a stronger baseline, to further verify the effectiveness of our proposed method. This also demonstrates that the proposed SAWR method does not limit to a certain NMT baseline. Concretely, we extend the bottom word representations by incorporating syntactic encodings $s{=}s_1 \cdots s_n$ (shown in Equation 5) into them, and then feed them into the transformer encoder by a linear projection layer to align with the input dimension. We implement Tree-RNN and Tree-Linearization for Transformer in a similar way, only adapting the source input word representing. We adopt a widely-used setting with 8 heads, 6 layers and the hidden dimension size of 512.

Table 5 shows the results. As shown, the transformer results are indeed much better than RNN-based baseline. The BLEU scores show an average increase of $40.74 - 37.09 = 3.65$. In addition, we can see that syntax information can still give positive influences based on the transformer. The SAWR approach can also outperform the baseline system significantly. Particularly, we find that our SAWR approach is much more effective than the Tree-RNN and Tree-Linearization approaches. The results further demonstrate the effectiveness of SAWRs in syntax integration for NMT.

5 Related Work

By explicitly expressing the structural connections between words and phrases, syntax trees been demonstrated helpful in SMT (Liu et al., 2006; Cowan et al., 2006; Marton and Resnik, 2008; Xie et al., 2011; Li et al., 2013; Williams et al., 2016). Although the representative Seq2Seq NMT models are able to capture latent long-distance relations by using neural network structures such GRU and LSTM (Sutskever et al., 2014; Wu et al., 2016), recent studies show that explicitly integrating syntax trees into NMT models can bring further gains (Sennrich and Haddow, 2016; Shi et al., 2016; Zhou et al., 2017a; Wu et al., 2017a; Aharoni and Goldberg, 2017). Under the NMT setting, the exploration of syntax trees could be more flexible, because of the strong capabilities of neural network in representing arbitrary structures.

Recursive neural networks based on LSTM or GRU have been one natural method to model syntax trees (Zhu et al., 2015; Tai et al., 2015; Li et al., 2015; Zhang et al., 2016; Teng and Zhang, 2016; Miwa and Bansal, 2016; Kokkinos and Potamianos, 2017), which are capable of representing the entire trees globally. Eriguchi et al. (2016) present the first work to apply a bottom-up Tree-LSTM for NMT. The major drawback is that its bottom-up composing strategy is insufficient for bottom nodes. Thus bi-directional extensions have been suggested (Chen et al., 2017a; Yang et al., 2017). Since Tree-RNN suffers serious inefficiency problem, Li et al. (2017) suggest a Tree-Linearization alternative, which converts constituent trees into a sequence of symbols mixed with words and syntactic tags. The method is as effective as Tree-RNN approaches yet more effective. Noticeably, all these studies focus on constituent trees.

There have been several studies for NMT using dependency syntax. Hashimoto and Tsuruoka (2017) propose to combine the head information with sequential words together as source encoder inputs, where their input trees are latent dependency graphs. Recently, there are several studies by using convolutional neural structures to represent source dependency trees, where tree nodes are modeled individually (Chen et al., 2017b; Bastings et al., 2017). Wu et al. (2017b) build a syntax enhanced encoder by multiple Bi-RNNs over several different word sequences based on different traversing orders over dependency trees, i.e., the original sequential order and several tree-based orders. All these methods require certain extra efforts to encode the source dependency syntax over a baseline Seq2Seq NMT.

6 Conclusion

We proposed a novel syntax integration method, SAWR, to incorporate source dependency-based syntax for NMT. It encodes dependency syntax implicitly, not requiring discrete syntax trees as inputs. Experiments showed that the method can bring significantly better performances for both Chinese-English and English-Vietnamese translation tasks. In addition, we compared the method with two approaches based on Tree-RNN and Tree-Linearization, which has been previously exploited for syntax integration, finding that our method is more effective and meanwhile very efficient. We conducted several experimental analyses to study our proposed methods deeper.

Acknowledgments

We thank all anonymous reviewers for their valuable comments. We thank Huadong Chen, Haoran Wei and Zaixiang Zheng for their help in implementing baseline neural machine translation models. This work is supported by National Natural Science Foundation of China (NSFC) grants 61525205, U1836222, and 61672211.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th ACL*, pages 132–140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP*, pages 1957–1967.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of ACL*, pages 1936–1945.

Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita, and Tiejun Zhao. 2017b. Neural machine translation with source dependency representation. In *Proceedings of EMNLP*, pages 2846–2852.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *ACL*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Brooke Cowan, Ivona Kučerová, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of EMNLP*, pages 232–241.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *The First Workshop on Neural Machine Translation (NMT)*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL*, pages 823–833.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of EMNLP*, pages 125–135.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of SMT*, pages 134–140.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proceedings of EACL*, pages 586–591.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the EMNLP*, pages 2304–2314.

Junhui Li, Philip Resnik, and Hal Daumé III. 2013. Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *Proceedings of NAACL*, pages 540–549.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL*, pages 688–697.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL*, pages 609–616.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of IWSLT 2015*, pages 76–79.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 EMNLP*, pages 1412–1421.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL*, pages 1003–1011.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of EMNLP*, pages 955–960.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, pages 1105–1116.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017. On-the-fly operation batching in dynamic computation graphs. In *Conference on Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, pages 2227–2237.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th ACL*, pages 1715–1725, Berlin, Germany.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*, pages 1683–1692.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of EMNLP*, pages 1526–1534.

Weiwei Sun and Xiaojun Wan. 2013. Data-driven, pcfg-based and pseudo-pcfg-based models for chinese dependency parsing. *TACL (TACL)*, 1(1):301–314.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Philip Williams, Rico Sennrich, Matt Post, and Philipp Koehn. 2016. Syntax-based statistical machine translation. *Synthesis Lectures on Human Language Technologies*, 9(4):1–208.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017a. Sequence-to-dependency neural machine translation. In *Proceedings of ACL*, pages 698–707.

Shuangzhi Wu, Ming Zhou, and Dongdong Zhang. 2017b. Improved neural machine translation with source syntax. In *Proceedings of IJCAI 2017*, pages 4179–4185.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of EMNLP*, pages 216–226.

Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao, and Jingbo Zhu. 2017. Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of the 2017 Conference on EMNLP*, pages 1432–1441.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of NAACL*, pages 310–320.

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2017. Modeling past and future for neural machine translation. *arXiv preprint arXiv:1711.09502*.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017a. Chunk-based bi-scale decoder for neural machine translation. In *Proceedings of the 55th ACL*, pages 580–586.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017b. Neural system combination for machine translation. In *Proceedings of ACL*, pages 378–384.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612.

Competence-based Curriculum Learning for Neural Machine Translation

Emmanouil Antonios Platanios[†], **Otilia Stretcu**[†], **Graham Neubig**[‡], **Barnabas Poczos**[†], **Tom M. Mitchell**[†]
[†]Machine Learning Department, [‡]Language Technologies Institute
Carnegie Mellon University
`{e.a.platanios,ostretcu,gneubig,bpoczos,tom.mitchell}@cs.cmu.edu`

Abstract

Current state-of-the-art NMT systems use large neural networks that are not only slow to train, but also often require many heuristics and optimization tricks, such as specialized learning rate schedules and large batch sizes. This is undesirable as it requires extensive hyperparameter tuning. In this paper, we propose a *curriculum learning framework for NMT* that reduces training time, reduces the need for specialized heuristics or large batch sizes, and results in overall better performance. Our framework consists of a principled way of deciding which training samples are shown to the model at different times during training, based on the estimated *difficulty* of a sample and the current *competence* of the model. Filtering training samples in this manner prevents the model from getting stuck in bad local optima, making it converge faster and reach a better solution than the common approach of uniformly sampling training examples. Furthermore, the proposed method can be easily applied to existing NMT models by simply modifying their input data pipelines. We show that our framework can help improve the training time and the performance of both recurrent neural network models and Transformers, achieving up to a 70% decrease in training time, while at the same time obtaining accuracy improvements of up to 2.2 BLEU.

1 Introduction

Neural Machine Translation (NMT; Kalchbrenner and Blunsom (2013); Bahdanau et al. (2015)) now represents the state-of-the-art adapted in most machine translation systems (Wu et al., 2016; Crego et al., 2016; Bojar et al., 2017a), largely due to its ability to benefit from end-to-end training on massive amounts of data. In particular, recently-introduced self-attentional Transformer architectures (Vaswani et al., 2017) are rapidly becoming the de-facto standard in NMT, having demonstrated both superior performance and training speed compared to previous architectures using recurrent neural networks (RNNs; (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014)). However, large scale NMT systems are often hard to train, requiring complicated heuristics which can be both time-consuming and expensive to tune. This is especially true for Transformers which, when carefully tuned, have been shown to consistently outperform RNNs (Popel and Bojar, 2018), but on the other hand, also rely on a number of heuristics such as specialized learning rates and large-batch training.

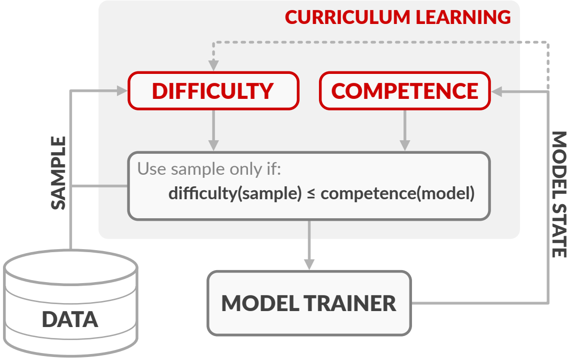


Figure 1: Overview of the proposed curriculum learning framework. During training, *difficulty* of each training sample is estimated and a decision whether to use it is made based on the current *competence* of the model.

In this paper, we attempt to tackle this problem by proposing a *curriculum learning framework* for training NMT systems that reduces training time, reduces the need for specialized heuristics or large batch sizes, and results in overall better performance. It allows us to train both RNNs and, perhaps more importantly, Transformers, with relative ease. Our proposed method is based on the idea of teaching algorithms in a similar manner as humans, from easy concepts to more difficult ones. This idea can be traced back to the work

Proceedings of NAACL-HLT 2019, pages 1162–1172
Minneapolis, Minnesota, June 2 - June 7, 2019.

of Elman (1993) and Krueger and Dayan (2009). The main motivation is that training algorithms can perform better if training data is presented in a specific order, starting from *easy* examples and moving on to more *difficult* ones, as the learner becomes more *competent*. In the case of machine learning, it can also be thought of as a means to avoid getting stuck in bad local optima early on in training. An overview of the proposed framework is shown in Figure 1.

Notably, we are not the first to examine curriculum learning for NMT, although other related works have met with mixed success. Kocmi and Bojar (2017) explore impact of several curriculum heuristics on training a translation system for a single epoch, presenting the training examples in an easy-to-hard order based on sentence length and vocabulary frequency. However, their strategy introduces all training samples during the first epoch, and how this affects learning in following epochs is not clear, with official evaluation results (Bojar et al., 2017b) indicating that final performance may indeed be hurt with this strategy. Contemporaneously to our work, Zhang et al. (2018) further propose to split the training samples into a predefined number of bins (5, in their case), based on various difficulty metrics. A manually designed curriculum schedule then specifies the bins from which the model samples training examples. Experiments demonstrate that benefits of curriculum learning are highly sensitive to several hyperparameters (e.g., learning rate, number of iterations spent in each phase, etc.), and largely provide benefits in convergence speed as opposed to final model accuracy.

In contrast to these previous approaches, we define a continuous curriculum learning method (instead of a discretized regime) with only one tunable hyperparameter (the duration of curriculum learning). Furthermore, as opposed to previous work which only focuses on RNNs, we also experiment with Transformers, which are notoriously hard to train (Popel and Bojar, 2018). Finally, unlike any of the work described above, we show that our curriculum approach helps not only in terms of convergence speed, but also in terms of the learned model performance. In summary, our method has the following desirable features:

1. **Abstract:** It is a novel, generic, and extensible formulation of curriculum learning. A number of previous heuristic-based approaches, such as that of Kocmi and Bojar (2017), can be formulated as special cases of our framework.
2. **Simple:** It can be applied to existing NMT systems with only a small modification to their training data pipelines.
3. **Automatic:** It does not require any tuning other than picking the value of a single parameter, which is the length of the curriculum (i.e., for how many steps to use curriculum learning, before easing into normal training).
4. **Efficient:** It reduces training time by up to 70%, whereas contemporaneous work of Zhang et al. (2018) reports reductions of up to 46%.
5. **Improved Performance:** It improves the performance of the learned models by up to 2.2 BLEU points, where the best setting reported by Zhang et al. (2018) achieves gains of up 1.55 BLEU after careful tuning.

In the next section, we introduce our proposed curriculum learning framework.

2 Proposed Method

We propose *competence-based curriculum learning*, a training framework based on the idea that training algorithms can perform better if training data is presented in a way that picks examples appropriate for the model's current *competence*. More specifically, we define the following two concepts that are central to our framework:

Difficulty: A value that represents the difficulty of a training sample and that may depend on the current state of the learner. For example, sentence length is an intuitive difficulty metric for natural language processing tasks. The only constraint is that difficulty scores are comparable across different training samples (i.e., the training samples can be ranked according to their difficulty).

Competence: A value between 0 and 1 that represents the progress of a learner during its training. It is defined as a function of the learner's state. More specifically, we define the competence, $c(t)$ at time t (measured in terms of training steps), of a learner as the proportion of training data it is allowed to use at that time. The training examples are ranked according to their difficulty and the learner is only allowed to use the top $c(t)$ portion of them at time t.

Using these two concepts, we propose Algorithm 1 (a high-level overview is shown in Figure 1, an example visualization of the first two steps is shown

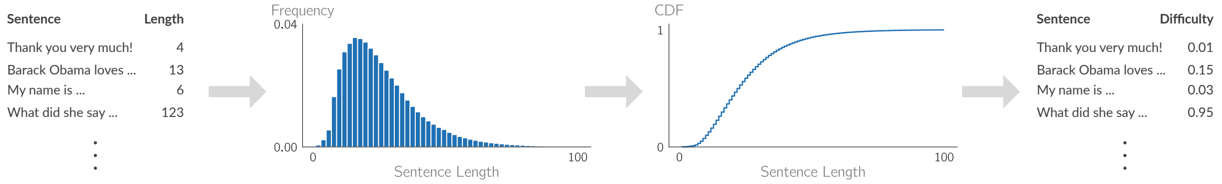


Figure 2: Example visualization of the preprocessing sequence used in the proposed algorithm. The histogram shown is that of sentence lengths from the WMT-16 `En→De` dataset used in our experiments. Here sentence lengths represent an example difficulty scoring function, d. "CDF" stands for the empirical "cumulative density function" obtained from the histogram on the left plot.

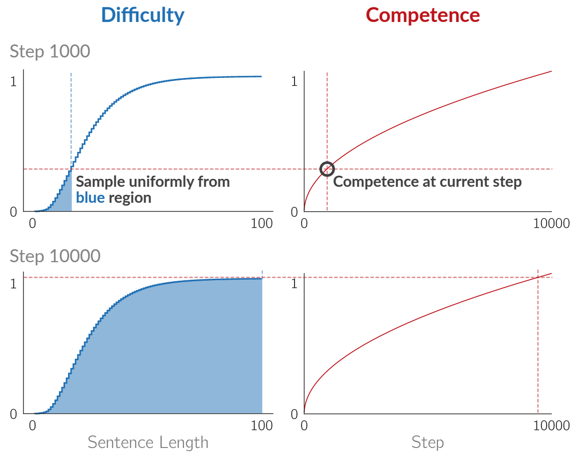


Figure 3: Example illustration of the training data "filtering" performed by our curriculum learning algorithm. At each training step: (i) the current competence of the model is computed, and (ii) a batch of training examples is sampled uniformly from all training examples whose difficulty is lower than that competence. In this example, we are using the sentence length difficulty heuristic shown in Equation 1, along with the square root competence model shown in Equation 7.

in Figure 2, and an example of the interaction between difficulty and competence is shown in Figure 3).

Note that, at each training step, we are not changing the relative probability of each training sample under the input data distribution, but we are rather constraining the domain of that distribution, based on the current competence of the learner. Eventually, once the competence becomes 1, the training process becomes equivalent to that without using a curriculum, with the main difference that the learner should now be more capable to learn from the more difficult examples. Given the dependence of this algorithm on the specific choices of the difficulty scoring function, d, and the competence function, c, we now describe our instantiations for training NMT models.

Algorithm 1: Competence-based curriculum learning algorithm.

```
Input: Dataset, $\mathcal{D} = \{s_i\}_{i=1}^{M}$, consisting of $M$
       samples, model trainer, $\mathcal{T}$, that takes as input
       batches of training data to use at each step,
       difficulty scoring function, $d$, and competence
       function, $c$.
Compute the difficulty, $d(s_i)$, for each $s_i \in \mathcal{D}$.
Compute the cumulative density function (CDF) of
  the difficulty scores. This results in one difficulty
  CDF score per sample, $\bar{d}(s_i) \in [0, 1]$. Illustrated in
  Figure 2.
for training step $t = 1, \ldots$ do
    Compute the model competence, $c(t)$.
    Sample a data batch, $B_t$, uniformly from all
      $s_i \in \mathcal{D}$, such that $\bar{d}(s_i) \leq c(t)$. Illustrated in
      Figure 3.
    Invoke the trainer, $\mathcal{T}$, using $B_t$ as input.
Output: Trained model.
```

2.1 Difficulty Metrics

There are many possible ways of defining the difficulty of translating a sentence. We consider two heuristics inspired by what we, as humans, may consider difficult when translating, and by factors which can negatively impact the optimization algorithms used when training NMT models. In the rest of this section we denote our training corpus as a collection of M sentences, $\{s_i\}_{i=1}^{M}$, where each sentence is a sequence of words. $s_i = \{w_0^i, \ldots, w_{N_i}^i\}$.

Sentence Length: We argue that it is harder to translate longer sentences, as longer sentences require being able to translate their component parts, which often consist of short sentences. Furthermore, longer sentences are intuitively harder to translate due to the propagation of errors made early on when generating the target language sentence. Therefore, a simple way to define the difficulty of a sentence $s_i = \{w_0^i, \ldots, w_{N_i}^i\}$ is as follows:

$$d_{\text{length}}(s_i) \triangleq N_i. \tag{1}$$

Note that we can compute this difficulty metric on either the source language sentence or the target language sentence. We only consider the source sentence in this paper [1].

Word Rarity: Another aspect of language that can affect the difficulty of translation is the frequency with which words appear. For example, humans may find rare words hard to translate because we rarely ever see them and it may be hard to recall their meaning. The same can be true for NMT models where: (i) the statistical strength of the training examples containing rare words is low and thus the model needs to keep revisiting such words in order to learn robust representations for them, and (ii) the gradients of the rare word embeddings tend to have high variance; they are overestimates of the true gradients in the few occasions where they are non-zero, and underestimates otherwise. This suggests that using word frequencies may be a helpful difficulty heuristic. Given a corpus of sentences, $\{s_i\}_{i=1}^M$, we define relative word frequencies as:

$$\hat{p}(w_j) \triangleq \frac{1}{N_{\text{total}}} \sum_{i=1}^{M} \sum_{k=1}^{N_i} \mathbb{1}_{w_k^i = w_j}, \qquad (2)$$

where $j = 1, \ldots, \#\{\text{unique words in corpus}\}$ and $\mathbb{1}_{\text{condition}}$ is the indicator function which is equal to 1 if its condition is satisfied and 0 otherwise. Next we need to decide how to aggregate the relative word frequencies of all words in a sentence to obtain a single difficulty score for that sentence. Previous research has proposed various pooling operations, such as minimum, maximum, and average (Zhang et al., 2018), but they show that they do not work well in practice. We propose a different approach. Ultimately, what might be most important is the overall likelihood of a sentence as that contains information about both word frequency and, implicitly, sentence length. An approximation to this likelihood is the product of the unigram probabilities, which is related to previous work in the area of active learning (Settles and Craven, 2008). This product can be thought of as an approximate language model (assuming words are sampled independently) and also implicitly incorporates information about the sentence length that was proposed earlier (longer sentence scores are products over more terms in $[0, 1]$ and are thus likely to be smaller). We thus propose the following difficulty heuristic:

$$d_{\text{rarity}}(s_i) \triangleq -\sum_{k=1}^{N_i} \log \hat{p}(w_k^i), \qquad (3)$$

where we use logarithms of word probabilities to prevent numerical errors. Note that negation is used because we define less likely (i.e., more rare) sentences as more difficult.

These are just two examples of difficulty metrics, and it is easy to conceive of other metrics such as the occurrence of homographs (Liu et al., 2018) or context-sensitive words (Bawden et al., 2018), the examination of which we leave for future work.

2.2 Competence Functions

For this paper, we propose two simple functional forms for $c(t)$ and justify them with some intuition. More sophisticated strategies that depend on the loss function, the loss gradient, or on the learner's performance on held-out data, are possible, but we do not consider them in this paper.

Linear: This is a simple way to define $c(t)$. Given an initial value $c_0 \triangleq c(0) \geq 0$ and a slope parameter r, we define:

$$c(t) \triangleq \min\left(1, tr + c_0\right). \qquad (4)$$

In this case, new training examples are constantly being introduced during the training process, with a constant rate r (as a proportion of the total number of available training examples). Note that we can also define $r = (1 - c_0)/T$, where T denotes the time after which the learner is fully competent, which results in:

$$c_{\text{linear}}(t) \triangleq \min\left(1, t\frac{1 - c_0}{T} + c_0\right). \qquad (5)$$

Root: In the case of the linear form, the same number of new and more difficult, examples are added to the training set, at all times t. However, as the training data grows in size, it gets less likely that any single data example will be sampled in a training batch. Thus, given that the newly added examples are less likely to be sampled, we propose to reduce the number of new training examples per unit time as training progresses to give the learner

---

[1] NMT models typically first pick up information about producing sentences of correct length. It can be argued that presenting only short sentences first may lead to learning a strong bias for the sentence lengths. In our experiments, we did not observe this to be an issue as the models kept improving and predicting sentences of correct length, throughout training.

sufficient time to assimilate their information content. More specifically, we define the rate in which new examples are added as inversely proportional to the current training data size:

$$\frac{dc(t)}{dt} = \frac{P}{c(t)}, \tag{6}$$

for some constant $P \geq 0$. Solving this simple differential equation, we obtain:

$$\int c(t)dc(t) = \int Pdt \Rightarrow c(t) = \sqrt{2Pt + D},$$

for some constants P and D. Then, we consider the following constraint: $c_0 \triangleq c(0) = \sqrt{D} \Rightarrow D = c_0^2$. Finally, we also have that $c(T) = 1 \Rightarrow P = (1 - c_0^2)/2T$, where T denotes the time after which the learner is fully competent. This, along with the constraint that $c(t) \in [0, 1]$ for all $t \geq 0$, results in the following definition:

$$c_{\text{sqrt}}(t) \triangleq \min\left(1, \sqrt{t\frac{1 - c_0^2}{T} + c_0^2}\right). \tag{7}$$

In our experiments, we refer to this specific formulation as the "square root" competence model. If we want to make the curve *sharper*, meaning that even more time is spent per sample added later on in training, then we can consider the following more general form, for $p \geq 1$:

$$c_{\text{root-}p}(t) \triangleq \min\left(1, \sqrt[p]{t\frac{1 - c_0^p}{T} + c_0^p}\right). \tag{8}$$

We observed that best performance is obtained when $p = 2$ and then, as we increase p, performance converges to that obtained when training without a curriculum. Plots of the competence functions we presented are shown in Figure 4.

2.3 Scalability

Our method can be easily used in large-scale NMT systems. This is because it mainly consists of a preprocessing step of the training data that computes the difficulty scores. The implementation we are releasing with this paper computes these scores in an efficient manner by building a graph describing their dependencies, as well as whether they are sentence-level scores (e.g., sentence length), or corpus-level (e.g., CDF), and using that graph to optimize their execution. Using only 8GB of memory, we can process up to 20k sentences per second when computing sentence rarity scores, and up to 150k sentences per second when computing sentence length scores.

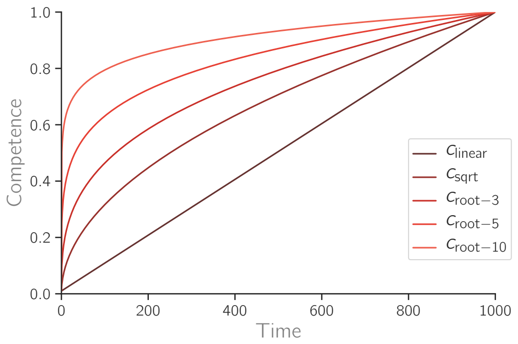


Figure 4: Plots of various competence functions with $c_0 = 0.01$ (initial competence value) and $T = 1,000$ (total duration of the curriculum learning phase).

3 Experiments

For our experiments, we use three of the most commonly used datasets in NMT, that range from a small benchmark dataset to a large-scale dataset with millions of sentences. Statistics about the datasets are shown in Table 1. We perform experiments using both RNNs and Transformers. For the RNN experiments we use a bidirectional LSTM for the encoder, and an LSTM with the attention model of Bahdanau et al. (2015) for the decoder. The number of layers of the encoder and the decoder are equal. We use a 2-layer encoder and a 2-layer decoder for all experiments on IWSLT datasets, and a 4-layer encoder and a 4-layer decoder for all experiments on the WMT dataset, due to the dataset's significantly larger size. For the Transformer experiments we use the BASE model proposed by Vaswani et al. (2017). It consists of a 6-layer encoder and decoder, using 8 attention heads, and 2,048 units for the feed-forward layers. The multi-head attention keys and values depth is set to the word embedding size. The word embedding size is 512 for all experiments. Furthermore, for the Transformer experiments on the two smaller datasets we do not use any learning rate schedule, and for the experiments on the largest dataset we use the default Transformer schedule. A detailed discussion on learning rate schedules for Transformers is provided near the end of this section. All of our experiments were conducted on a machine with a single Nvidia V100 GPU, and 24 GBs of system memory.

During training, we use a label smoothing factor of 0.1 (Wu et al., 2016) and the AMSGrad optimizer (Reddi et al., 2018) with its default parameters in TensorFlow, and a batch size of 5,120 tokens

Dataset	# Train	# Dev	# Test
IWSLT-15 `En→Vi`	133k	768	1268
IWSLT-16 `Fr→En`	224k	1080	1133
WMT-16 `En→De`	4.5m	3003	2999

Table 1: Number of parallel sentences in each dataset. “k” stands for “thousand” and “m” stands for “million”.

(due to GPU memory constraints). During inference, we employ beam search with a beam size of 10 and the length normalization scheme of Wu et al. (2016).[2]

Curriculum Hyperparameters. We set the initial competence c_0 to 0.01, in all experiments. This means that all models start training using the 1% easiest training examples. The curriculum length T is effectively the only hyperparameter that we need to set for our curriculum methods. In each experiment, we set T in the following manner: we train the baseline model without using any curriculum and we compute the number of training steps it takes to reach approximately 90% of its final BLEU score. We then set T to this value. This results in T being set to 5,000 for the RNN experiments on the IWSLT datasets, and 20,000 for the corresponding Transformer experiments. For WMT, we set T to 20,000 and 50,000 for RNNs and Transformers, respectively. Furthermore, we use the following notation and abbreviations when presenting our results:

- Plain: Trained without using any curriculum.
- SL: Curriculum with sentence length difficulty.
- SR: Curriculum with sentence rarity difficulty.
- Linear: Curriculum with the linear competence shown in Equation 5.
- Sqrt: Curriculum with the square root competence shown in Equation 7.

Data Preprocessing. Our experiments are performed using the machine translation library released by Platanios et al. (2018). We use the same data preprocessing approach the authors used in their experiments. While training, we consider sentences up to length 200. Similar to them, for the IWSLT-15 experiments we use a per-language vocabulary which contains the 20,000 most frequently occurring words, while ignoring words that appear less than 5 times in the whole corpus. For the IWSLT-16 and WMT-16 experiments we use a byte-pair encoding (BPE) vocabulary (Sennrich et al., 2016) trained using 32,000 merge operations, similar to the original Transformer paper by Vaswani et al. (2017).

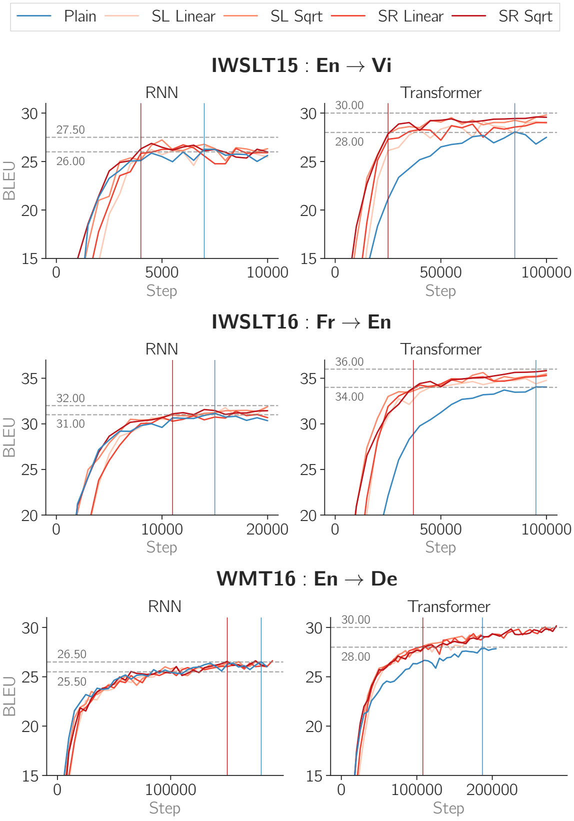


Figure 5: Plots illustrating the performance of various models on the test set, as training progresses. **Blue** lines represent the baseline methods when no curriculum is used, and **red** lines represent the same models when different versions of our curriculum learning framework are used to train them. The vertical lines represent the step in which the models attain the BLEU score that the baseline models attain at convergence.

Results. We present a summary of our results in Table 2 and we also show complete learning curves for all methods in Figure 5. The evaluation metrics we use are the test set BLEU score and the time it takes for the models using curriculum learning to obtain the BLEU score that the baseline models attain at convergence. We observe that Transformers consistently benefit from our curriculum learning approach, achieving gains of up to 2 BLEU, and reductions in training time of up to 70%. RNNs also benefit, but to a lesser extent. This is consistent with our motivation for this paper, which

---

[2] We emphasize that we did not run experiments with other architectures or configurations, and thus our baseline architectures were not chosen because they were favorable to our method, but rather because they were frequently mentioned in existing literature.

		RNN					TRANSFORMER					
		Plain	SL Curriculum		SR Curriculum		Plain	Plain*	SL Curriculum		SR Curriculum	
			c_{linear}	c_{sqrt}	c_{linear}	c_{sqrt}			c_{linear}	c_{sqrt}	c_{linear}	c_{sqrt}
BLEU	En→Vi	26.27	26.57	**27.23**	26.72	26.87	28.06	29.77	29.14	29.57	29.03	**29.81**
	Fr→En	31.15	31.88	**31.92**	31.39	31.57	34.05	34.88	34.98	35.47	35.30	**35.83**
	En→De	26.53	26.55	26.54	**26.62**	**26.62**	–	27.95	28.71	29.28	29.93	**30.16**
Time	En→Vi	1.00	0.64	0.61	0.71	**0.57**	1.00	1.00	0.44	0.33	0.35	**0.31**
	Fr→En	1.00	1.00	0.93	1.10	**0.73**	1.00	1.00	0.49	0.44	0.42	**0.39**
	En→De	1.00	0.86	0.89	1.00	**0.83**	–	1.00	0.58	**0.55**	**0.55**	**0.55**

Table 2: Summary of experimental results. For each method and dataset, we present the test set BLEU score of the best model based on validation set performance. We also show the relative time required to obtain the BLEU score of the best performing baseline model. For example, if an RNN gets to 26.27 BLEU in $10,000$ steps and the SL curriculum gets to the same BLEU in $3,000$ steps, then the plain model gets a score of 1.0 and the SL curriculum receives a score of $3,000/10,000 = 0.3$. "Plain" stands for the model trained without a curriculum and, for Transformers, "Plain*" stands for the model trained using the learning rate schedule shown in Equation 9.

stems from the observation that training RNNs is easier and more robust than training Transformers. Furthermore, the square root competence model consistently outperforms the linear model, which fits well with our intuition and motivation for introducing it. Regarding the difficulty heuristics, sentence length and sentence rarity both result in similar performance.

We also observe that, for the two small datasets, RNNs converge faster than Transformers in terms of both the number of training iterations and the overall training time. This is contrary to other results in the machine translation community (e.g., Vaswani et al., 2017), but could be explained by the fact that we are not using any learning rate schedule for training Transformers. However, they never manage to outperform Transformers in terms of test BLEU score of the final model. Furthermore, to the best of our knowledge, for IWSLT-15 we achieve state-of-the-art performance. The highest previously reported result was 29.03 BLEU (Platanios et al., 2018), in a multi-lingual setting. Using our curriculum learning approach we are able to achieve a BLEU score of 29.81 for this dataset.

Overall, we have shown that *our curriculum learning approach consistently outperforms models trained without any curriculum*, in both limited data settings and large-scale settings.

Learning Rate Schedule. In all of our IWSLT experiments so far, we use the default AMSGrad learning rate of 0.001 and intentionally avoid using any learning rate schedules. However, Transformers are not generally trained without a learning rate schedule, due to their instability. Such schedules typically use a warm-up phase, which means that the learning rate starts at a very low value and keeps increasing until the end of the warm-up period, after which a decay rate is typically used. In order to show that our curriculum learning approach can act as a principled alternative to such highly tuned learning rate schedules, we now present the results we obtain when training our Transformers using the following learning rate schedule:

$$\mathrm{lr}(t) \triangleq d_{\text{embedding}}^{-0.5} \min\left(t^{-0.5}, t \cdot T_{\text{warmup}}^{-1.5}\right), \qquad (9)$$

where t is the current training step, $d_{\text{embedding}}$ is the word embeddings size, and T_{warmup} is the number of warmup steps and is set to 10,000 in these experiments. This schedule was proposed in the original Transformer paper (Vaswani et al., 2017), and was tuned for the WMT dataset.

The results obtained when using this learning rate schedule are also shown in table 2, under the name "Plain*". *In both cases, our curriculum learning approach obtains a better model in about 70% less training time*. This is very important, especially when applying Transformers in new datasets, because such learning rate heuristics often require careful tuning. This tuning can be both very expensive and time consuming, often resulting in very complex mathematical expressions, with no clear motivation or intuitive explanation (Chen et al., 2018). Our curriculum learning approach achieves better results, in significantly less time, while only requiring one parameter (the length of the curriculum).

Note that even without using any learning rate schedule, our curriculum methods were able to achieve performance comparable to the "Plain*" in about twice as many training steps. "Plain" was not able to achieve a BLEU score above 2.00 even after fives times as many training steps, at which point we stopped these experiments.

Implementation and Reproducibility. We are releasing an implementation of our proposed method and experiments built on top of the machine translation library released by Platanios et al. (2018), using TensorFlow Scala (Platanios, 2018), and is available at `https://github.com/eaplatanios/symphony-mt`. Furthermore, all experiments can be run on a machine with a single Nvidia V100 GPU, and 24 GBs of system memory. Our most expensive experiments — the ones using Transformers on the WMT-16 dataset — take about 2 days to complete, which would cost about $125 on a cloud computing service such as Google Cloud or Amazon Web Services, thus making our results reproducible, even by independent researchers.

4 Related work

The idea of teaching algorithms in a similar manner as humans, from easy concepts to more difficult ones, has existed for a long time (Elman, 1993; Krueger and Dayan, 2009). Machine learning models are typically trained using stochastic gradient descent methods, by uniformly sampling mini-batches from the pool of training examples, and using them to compute updates for the model parameters. Deep neural networks, such as RNNs and Transformers, have highly non-convex loss functions. This makes them prone to getting stuck in saddle points or bad local minima during training, often resulting in long training times and bad generalization performance. Bengio et al. (2009) propose a curriculum learning approach that aims to address these issues by changing the mini-batch sampling strategy. They propose starting with a distribution that puts more weight on easy samples, and gradually increase the probability of more difficult samples as training progresses, eventually converging to a uniform distribution. They demonstrate empirically that such curriculum approaches indeed help decrease training times and sometimes even improve generalization.

Perhaps the earliest attempt to apply curriculum learning in MT was made by Zou et al. (2013). The authors employed a curriculum learning method to learn Chinese-English bilingual word embeddings, which were subsequently used in the context of phrase-based machine translation. They split the word vocabulary in 5 separate groups based on word frequency, and learned separate word embeddings for each of these groups in parallel. Then, they merged the 5 different learned embeddings and continued training using the full vocabulary. While this approach makes use of some of the ideas behind curriculum learning, it does not directly follow the original definition introduced by Bengio et al. (2009). Moreover, their model required 19 days to train. There have also been a couple of attempts to apply curriculum learning in NMT that were discussed in section 1.

There also exists some relevant work in areas other than curriculum learning. Zhang et al. (2016) propose training neural networks for NMT by focusing on hard examples, rather than easy ones. They report improvements in BLEU score, while only using the hardest 80% training examples in their corpus. This approach is more similar to boosting by Schapire (1999), rather than curriculum learning, and it does not help speed up the training process; it rather focuses on improving the performance of the trained model. The fact that hard examples are used instead of easy ones is interesting because it is somewhat contradictory to that of curriculum learning. Also, in contrast to curriculum learning, no ordering of the training examples is considered.

Perhaps another related area is that of active learning, where the goal is to develop methods that request for specific training examples. Haffari et al. (2009), Bloodgood and Callison-Burch (2010), and Ambati (2012) all propose methods to solicit training examples for MT systems, based on the occurrence frequency of n-grams in the training corpus. The main idea is that if an n-gram is very rare in the training corpus, then it is difficult to learn to translate sentences in which it appears. This is related to our sentence rarity difficulty metric and points out an interesting connection between curriculum learning and active learning.

Regarding training Transformer networks, Shazeer and Stern (2018) perform a thorough experimental evaluation of Transformers, when using different optimization configurations. They show that a significantly higher level of performance can be reached by not using momentum during optimization, as long as a carefully chosen learning rate schedule is used. Such learning rate schedules are often hard to tune because of the multiple seemingly arbitrary terms they often contain. Furthermore, Popel and Bojar (2018) show that, when using Transformers, increasing the batch size results in a better model at convergence. We believe this is indicative of very noisy gradients when starting to train Transformers and that higher batch sizes help increase the signal-to-noise ratio. We show that our proposed curriculum learning method offers a more principled and robust way to tackle this problem. Using our approach, we are able to train Transformers to state-of-the-art performance, using small batch sizes and without the need for peculiar learning rate schedules, which are typically necessary.

5 Conclusion and Future Work

We have presented a novel *competence-based* curriculum learning approach for training neural machine translation models. Our resulting framework is able to boost performance of existing NMT systems, while at the same time significantly reducing their training time. It differs from previous approaches in that it does not depend on multiple hyperparameters that can be hard to tune, and it does not depend on a manually designed discretized training regime. We define the notions of *competence*, for a learner, and *difficulty*, for the training examples, and propose a way to filter training data based on these two quantities. Perhaps most interestingly, we show that our method makes training Transformers faster and more reliable, but has a much smaller effect in training RNNs.

In the future, we are mainly interested in: (i) exploring more difficulty heuristics, such as measures of alignment between the source and target sentences (Kocmi and Bojar, 2017), sentence length discrepancies, or even using a pre-trained language model to score sentences, which would act as a more robust replacement of our sentence rarity heuristic, and (ii) exploring more sophisticated competence metrics that may depend on the loss function, the loss gradient, or on the learner's performance on held-out data. Furthermore, it would be interesting to explore applications of curriculum learning to multilingual machine translation (e.g., it may be easier to start with high-resource languages and move to low-resource ones later on). We would also like to explore the usefulness of our framework in more general machine learning tasks, outside of NMT.

Acknowledgments

We would like to thank Maruan Al-Shedivat and Dan Schwartz for the useful feedback they provided in early versions of this paper. This research was supported in part by AFOSR under grant FA95501710218.

References

Vamshi Ambati. 2012. *Active Learning and Crowdsourcing for Machine Translation in Low Resource Scenarios*. Ph.D. thesis, Pittsburgh, PA, USA. AAI3528171.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Michael Bloodgood and Chris Callison-Burch. 2010. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *ACL*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017a. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Ondřej Bojar, Jindřich Helcl, Tom Kocmi, Jindřich Libovický, and Tomáš Musil. 2017b. Results of the WMT17 Neural MT Training Task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 525–533. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 76–86. Association for Computational Linguistics.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. SYSTRAN's Pure Neural Machine Translation Systems. *CoRR*, abs/1610.05540.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 415–423, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum Learning and Minibatch Bucketing in Neural Machine Translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 379–386.

Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

Frederick Liu, Han Lu, and Graham Neubig. 2018. Handling homographs in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1336–1345, New Orleans, Louisiana. Association for Computational Linguistics.

Emmanouil A. Platanios. 2018. TensorFlow Scala. `https://github.com/eaplatanios/tensorflow_scala`.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual Parameter Generation for Universal Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium.

Martin Popel and Ondřej Bojar. 2018. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

Robert E. Schapire. 1999. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, Hawaii. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604, Stockholmsmässan, Stockholm Sweden. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Dakun Zhang, Jungi Kim, Josep Crego, and Jean Senellart. 2016. Boosting neural machine translation. *arXiv preprint arXiv:1612.06138*.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. 2018. An Empirical Exploration of Curriculum Learning for Neural Machine Translation. *CoRR*, abs/1811.00739.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398. Association for Computational Linguistics.

Extract and Edit: An Alternative to Back-Translation for Unsupervised Neural Machine Translation

Jiawei Wu and **Xin Wang** and **William Yang Wang**
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`{jiawei_wu,xwang,william}@cs.ucsb.edu`

Abstract

The overreliance on large parallel corpora significantly limits the applicability of machine translation systems to the majority of language pairs. Back-translation has been dominantly used in previous approaches for unsupervised neural machine translation, where pseudo sentence pairs are generated to train the models with a reconstruction loss. However, the pseudo sentences are usually of low quality as translation errors accumulate during training. To avoid this fundamental issue, we propose an alternative but more effective approach, **extract-edit**, to extract and then edit real sentences from the target monolingual corpora. Furthermore, we introduce a comparative translation loss to evaluate the translated target sentences and thus train the unsupervised translation systems. Experiments show that the proposed approach consistently outperforms the previous state-of-the-art unsupervised machine translation systems across two benchmarks (English-French and English-German) and two low-resource language pairs (English-Romanian and English-Russian) by more than 2 (up to 3.63) BLEU points.

1 Introduction

Promising results have been achieved in Neural Machine Translation (NMT) by representation learning (Cho et al., 2014b; Sutskever et al., 2014). But recent studies (Koehn and Knowles, 2017; Isabelle et al., 2017; Sennrich, 2017) highlight the overreliance of current NMT systems on large parallel corpora. In real-world cases, the majority of language pairs have very little parallel data, so the models need to leverage monolingual data to address this challenge (Gulcehre et al., 2015; Zhang and Zong, 2016; He et al., 2016; Yang et al., 2018).

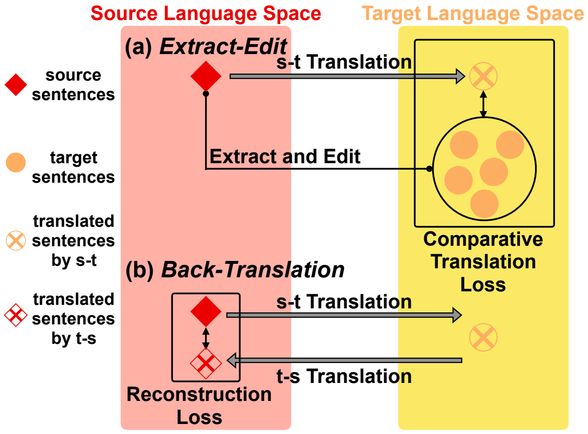


Figure 1: The comparison between two approaches of unsupervised NMT, extract-edit and back-translation. When training the source-to-target (s-t) translation model, instead of using the t-s back-translated sentences to train the model, we directly set the extracted-edited sentences as pivotal points to guide the training.

While many studies have explored how to use the monolingual data to improve translation performance with limited supervision, latest approaches (Lample et al., 2018a; Artetxe et al., 2018; Lample et al., 2018b) focus on the fully unsupervised scenario. Back-translation has been dominantly used in these approaches, where pseudo sentence pairs are generated to train the translation systems with a reconstruction loss. However, it is inefficient because the generated pseudo sentence pairs are usually of low quality. During the dual learning of back-translation, the errors could easily accumulate and thus the learned target language distribution would gradually deviate from the real target distribution. This critical drawback hinders the further development of the unsupervised NMT systems.

An alternative solution is to extract real parallel sentences from comparable monolingual corpora, and then use them to train the NMT systems. Recently, neural-based methods (Chu et al., 2016; Grover and Mitra, 2017; Grégoire and Langlais, 2018) aim to select potential parallel sentences

Proceedings of NAACL-HLT 2019, pages 1173–1183
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

from monolingual corpora in the same domain. However, these neural models need to be trained on a large parallel dataset first, which is not applicable to language pairs with limited supervision.

In this paper, we propose a radically different approach for unsupervised NMT—**extract-edit**, a powerful alternative to back-translation (see Figure 1). Specifically, to train the source-to-target translation model, we first extract potential parallel sentence candidates in the target language space given a source language sentence. Since it cannot be guaranteed that there always exist potential parallel sentence pairs in monolingual corpora, we further propose a simple but effective editing mechanism to revise the extracted sentences, making them aligned with the source language sentence. Then a comparative translation loss is introduced to evaluate the translated sentence based on the extracted-and-edited ones and train the translation model. Compared to back-translation, extract-edit avoids the distribution deviation issue by extracting and editing real sentences from the target language space. Those extracted-and-edited sentences serve as pivotal points in the target language space to guide the unsupervised learning. Thus, the learned target language distribution could be closer to the real one. The extract-edit model and the translation model, the two major parts of our method, can be jointly trained in a fully unsupervised way.

Empirical results on popular benchmarks show that exact-edit consistently outperforms the state-of-the-art unsupervised NMT system (Lample et al., 2018b) with back-translation across four different languages pairs. In summary, our main contributions are three-fold[1]:

- We propose a more effective alternative paradigm to back-translation, extract-edit, to train the unsupervised NMT systems with potentially real sentence pairs;
- We introduce a comparative translation loss for unsupervised learning, which optimizes the translated sentence by maximizing its relative similarity with the source sentence among the extracted-and-edited pairs;
- Our method advances the previous state-of-the-art NMT systems across four different language pairs under monolingual corpora only scenario.

2 Background

Without parallel sentence pairs as constraints on mapping language spaces, training NMT systems is an ill-posed problem because there are many potential mapping solutions. Nevertheless, some promising methods have been proposed in this field (Lample et al., 2018a; Artetxe et al., 2018; Lample et al., 2018b). The main technical protocol of these approaches can be summarized as three steps: *Initialization*, *Language Modeling*, and *Back-Translation*. In this section, we mainly introduce the three steps and the crucial settings that we have followed in our work.

In the remainder of the paper, we denote the space of source and target languages by $\mathcal{S}$ and $\mathcal{T}$, respectively. enc and dec refer to the encoder and decoder models in the sequence-to-sequence systems. $V_{s\to t}$ stands for the composition of enc in the source language and dec in the target language, which can be viewed as the source-to-target translation system.

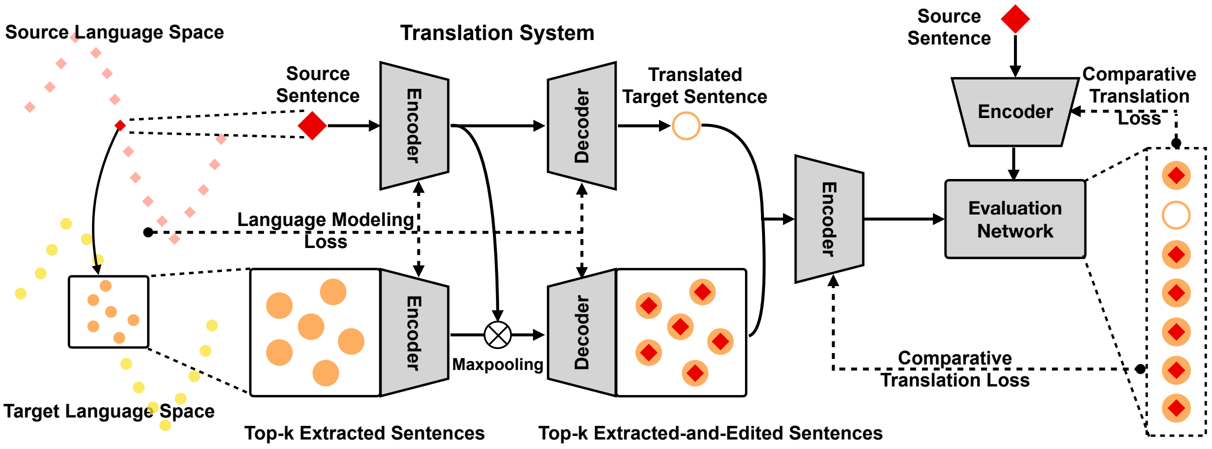
Initialization Given the ill-posed nature of the unsupervised NMT task, a suitable initialization method can help model the natural priors over the mapping of two language spaces we expect to reach. There are mainly two initialization methods: (1) bilingual dictionary inference (Conneau et al., 2018; Artetxe et al., 2018; Lample et al., 2018a) and (2) byte-pair encoding (BPE) (Sennrich et al., 2016b; Lample et al., 2018b). As shown in Lample et al. (2018b), the inferred bilingual dictionary can provide a rough word-by-word alignment of semantics, and the BPE can reduce the vocabulary size and eliminate the presence of unknown words in the output results.

In our *extract-edit* approach, to extract potential parallel sentence pairs, we need to compare the semantic similarity of sentences between two languages first. A proper initialization can also help align the semantic spaces and extract potential parallel pairs within them. Thus, following the previous methods, we use the inferred bilingual dictionary as described in Conneau et al. (2018) for unrelated language pairs and the shared BPE in Lample et al. (2018b) as initialization for related ones.

Language Modeling After a proper initialization, given large amounts of monolingual data, we

---

[1] The source code can be found in this repository: `https://github.com/jiaweiw/Extract-Edit-Unsupervised-NMT`



(a) Extract via Nearest Neighbor Search **(b) Edit via Information Integration** **(c) Comparative Translation**

Figure 2: The overview of our unsupervised NMT model based on the *extract-edit* approach. Given a source sentence, (a) the top-k potential parallel sentences of the target language are extracted via nearest neighbor search. (b) The extracted sentences are further edited with the source sentence. (c) The evaluation network evaluates the translated sentence and the extracted-and-edited sentences based on their similarities with the source sentence. Note that (1) all the encoders share the same parameters (same for decoders); (2) the decoding processes are non-differentiable, so the language modeling loss and the comparative translation loss are used to train the learning modules before and after the decoding processes, respectively.

can train language models on both source and target languages. These models express a data-driven prior about the composition of sentences in each language. In NMT, language modeling is accomplished via denosing autoencoding, by minimizing:

$$\begin{aligned}\mathcal{L}_{lm}(\theta_{enc},\theta_{dec}) =& \mathbb{E}_{x\sim\mathcal{S}}[-\log V_{s\to s}(x|C(x))]+\\ & \mathbb{E}_{y\sim\mathcal{T}}[-\log V_{t\to t}(y|C(y))]\end{aligned} \tag{1}$$

where C is a noise model with some words dropped and swapped, θ_{enc} and θ_{dec} are the learnable parameters of enc and dec. $V_{s\to s}$ and $V_{t\to t}$ are the encoder-decoder language models on the source and target sides, respectively.

In our *extract-edit* approach, we follow similar settings and adopt the noise model proposed by Lampe et al. (2018a). Note that the parameters of all enc are shared (same for dec) in our framework to ensure a strong alignment and mapping between two languages. This sharing operation is essential for both the translation model and the extract-edit model. Thus, we use enc to represent encoders in source language modeling $enc_{\mathcal{S}}$ and in target language modeling $enc_{\mathcal{T}}$ (same for dec).

Back-Translation Back-translation (Sennrich et al., 2016b) has been dominantly used in prior work to train the unsupervised NMT system. It couples the source-to-target translation model with a backward target-to-source model and trains the whole system with a reconstruction loss. This can be viewed as converting the unsupervised problem into a supervised scenario by generating pseudo language pairs (He et al., 2016).

Despite the popularity of back-translation in the previous methods (Lample et al., 2018a; Artetxe et al., 2018; Lample et al., 2018b), we argue that it suffers from the low-quality pseudo language pairs. Thus, in this work, we propose a new paradigm, extract-edit, to address this issue by extracting and editing potential real parallel sentences. Below we describe our approach in details.

3 Extract-Edit

The overview of our *extract-edit* approach is shown in Figure 2. We first extract and edit real sentences from the target language space according to their similarities with the source sentence. These extracted-and-edited sentences serve as pivotal points in the target language space, which locate a probable region where the real target sentence could be. Then we introduce a comparative translation loss to evaluate the translated sentence and train the system. Basically, the comparative translation loss encourages the translated sentence to approximate the real sentence by maximizing its relative similarity with the source sentence compared to the extracted-and-edited sentences. As a result, we manage to minimize the deviation of the learned target language distribution and the mapping noises between two language spaces.

3.1 Extract

Most existing methods in comparable corpora mining introduce two encoders to represent sentences of two languages separately, and then use another network to measure the similarity (Chu et al., 2016; Grover and Mitra, 2017; Grégoire and Langlais, 2018). However, owing to the shared encoders and decoders in language modeling, the semantic spaces of two languages are already strongly connected in our scenario.

Therefore, to avoid extra computation resources, we directly use the enc in language modeling to obtain sentence embeddings for two languages. As shown in Figure 2 (a), for a given source sentence s, we use the nearest neighbor search based on L_2 distance to find top-k real sentences from the target language space (k is a hyper-parameter decided empirically). The sentence embeddings used for searching are computed based on the shared encoder enc. The reason to choose top-k sentences rather than top-1 is to keep a high recall rate and obtain more related samples from the target language space. Finally, given the source sentence s, we denote M as a set of the k potential parallel target sentences:

$$M = \{t| \min_{1,\cdots,k}(||e_s - e_t||), t \in \mathcal{T}\}, \quad (2)$$

where e_s and e_t are sentence embeddings encoded by the shared encoder enc.

3.2 Edit

Even though the extracted sentences could serve as pivotal points to guide NMT, there is no guarantee that there always exists a parallel sentence in the target corpus. Thus, in order to make it closer to the real paired sentence in the target language space, we propose an editing mechanism to revise the extracted target sentence $t \in M$ based on the semantics of the source sentence s. As described in Figure 2 (b), we employ a maxpooling layer to reserve the more significant features between the source sentence embedding e_s and the extracted sentence embedding e_t ($t \in M$), and then decode it into a new sentence t':

$$M' = \{t'|t' = dec(\text{maxpooling}(e_s, e_t)), t \in M\}, \quad (3)$$

where M' is the set of the extracted-and-edited sentences. Based on the semantic information of the source sentence s, we can further improve the extracted results with this editing mechanism. Unlike other studies using the editing to generate more structural sentences (Guu et al., 2018; Hashimoto et al., 2018), here the revised sentences are designed to serve as better pivotal points in the target language space to guide the translation procedure. This can also be viewed as adding constraints when aligning the two language spaces.

3.3 Evaluate

Given a source sentence s, we can translate it as t^* using the source-to-target translation model $P_{s\to t}$. Meanwhile, a set M' of k sentences can also be generated by the extract-edit approach described above. Although the M' may contain potential parallel sentences t' for s, we cannot directly use (s, t') as ground-truth sentence pairs to train the translation model $V_{s\to t}$ because the NMT system is sensitive to noises (Cho et al., 2014a; Cheng et al., 2018). The rough operation like this will result in sub-optimal translation performance.

Therefore, in order to assess the quality of the translated sentence t^* and train the translation model $V_{s\to t}$, we introduce an evaluation network R for evaluating the relative similarities between the source and target sentences among all sentence pairs. The evaluation network R is a multilayer perceptron; it takes the target sentence embedding e_t and source sentence embedding e_s as inputs, and converts them into the joint embedding space as r_t and r_s. So the similarity

$$\alpha(t|s) = cosine(r_t, r_s) = \frac{r_t \cdot r_s}{||r_t||||r_s||}. \quad (4)$$

Then, a softmax-like formulation is used to compute the ranking score for the translated sentence t^* given the extracted-and-edited sentence set M':

$$P(t^*|s, M') = \frac{exp(\lambda\alpha(t^*|s))}{\sum_{t' \in M' \cup \{t^*\}} exp(\lambda\alpha(t'|s))}, \quad (5)$$

where the hyper-parameter λ is similar to the inverse temperature of the softmax function. Lower λ encourages the model to treat all extracted-edited sentences equally, while higher λ highlights the importance of sentences with higher-score.

3.4 Learning

Comparative Translation As introduced above, the ranking score calculates the relative similarity between the $< s, t^* >$ pair and all the extracted-and-edited pairs $< s, t' >$. Assuming we have a good evaluation network R with θ_R denoting its parameters, we further introduce the comparative translation loss $\mathcal{L}_{com}$ for unsupervised machine translation:

$$\mathcal{L}_{com}(\theta_{enc}|\theta_R) = -\mathbb{E}(\log P(t^* = V_{s\to t}(s)|s, M')), \tag{6}$$

where θ_{enc} is the parameters of the shared encoder enc. Basically, the translation model is trying to minimize the relative distance of the translated sentence t^* to the source sentence s compared to the top-k extracted-and-edited sentences in the target language space. Intuitively, we view the top-k extracted-and-edited sentences as the anchor points to locate a probable region in the target language space, and iteratively improve the source-to-target mapping via the comparative learning scheme.

Combined with the language modeling constraints as described in Equation 1, the final loss function for training the the translation model $V_{s\to t}$ is defined as:

$$\begin{aligned}\mathcal{L}_{s\to t}(\theta_{enc}, \theta_{dec}|\theta_R) =& \omega_{lm}\mathcal{L}_{lm}(\theta_{enc}, \theta_{dec}) + \\ & \omega_{com}\mathcal{L}_{com}(\theta_{enc}|\theta_R),\end{aligned} \tag{7}$$

where ω_{lm} and ω_{com} are hyper-parameters weighing the importance of the language modeling and the comparative learning.

Adversarial Objective Meanwhile, we need to learn a good evaluation network R to transform sentence embedding of the shared encoder into the comparable space. The evaluation network R is also shared by two languages to ensure a strong connection between two language spaces. Inspired by adversarial learning (Goodfellow et al., 2014), we can view our translation system as a "generator" that learns to generate a good translation with a higher similarity score than the extracted-and-edited sentences, and the evaluation network R as a "discriminator" that learns to rank the extracted-and-edited sentences (real sentences in the target language space) higher than the translated sentences. Thus, we have the following objective function for the evaluation network R:

$$\mathcal{L}_R(\theta_R) = -\mathbb{E}_{t' \in M'}(\log P(t^{'}|s, M')). \tag{8}$$

Algorithm 1: The algorithm of our unsupervised NMT system with *extract-edit* approach.

```
Given two monolingual corpora, source $\mathcal{S}$ and target $\mathcal{T}$;
Initialization as in Section 2;
Language Modeling as in Section 2 to obtain the
    initialized translation model $V^{(0)}_{s\to t} = enc^{(0)} \circ dec^{(0)}$;
for $n \leftarrow 1$ to $N$ do
    Given a source sentence $s$;
    Extract the top-$k$ target sentences as the set $M$;
    Edit the sentences in $M$ to obtain the set $M'$;
    Update the evaluation network
      $R : \theta_R \leftarrow \text{argmin}\mathcal{L}_R$;
    Update the shared encoder and decoder $R$:
      $\theta_{enc}, \theta_{dec} \leftarrow \text{argmin}\mathcal{L}_{s\to t}$;
    Update the translation model:
      $V^{(n+1)}_{s\to t} = enc^{(n)} \circ dec^{(n)}$;
return $V^{(N+1)}_{s\to t} = enc^{(N)} \circ dec^{(N)}$.
```

Based on Equation 7 and 8, the final adversarial objective is defined as

$$\begin{aligned}&\min_{\theta_{enc},\theta_{dec}} \max_{\theta_R} \mathcal{L}(\theta_{enc}, \theta_{dec}, \theta_R) \\ &= -\mathcal{L}_R(\theta_R) + \mathcal{L}_{s\to t}(\theta_{enc}, \theta_{dec}|\theta_R),\end{aligned} \tag{9}$$

where the translation model $V_{s\to t}$ and the evaluation network R play the two-player mini-max game. We evenly alternately update between the encoder-decoder translation model and the evaluation network. The detailed training procedure is described in Algorithm 1.

3.5 Model Selection

In the fully unsupervised setting, we do not have access to parallel sentence pairs. Thus, we need to find a criterion correlated with the translation quality to select hyper-parameters. For a neural translation model $V_{s\to t}$, we propose the following criterion $D_{s\to t}$ to tune the hyper-parameters:

$$D_{s\to t} = \mathbb{E}_{s\in\mathcal{S}}[\mathbb{E}(\log P(t^*|s, M'))], \tag{10}$$

where $t^* = V_{s\to t}(s)$. Basically, we choose the hyper-parameters with the maximum expectation of the ranking scores of all translated sentences.

4 Experiments

4.1 Datasets

We consider four language pairs: English-French (*en-fr*), English-German (*en-de*), English-Russian (*en-ru*) and English-Romanian (*en-ro*) for evaluation. For a fair comparison, we use the same corpora as in Lample et al. (2018b) for these languages for fair comparison. For English, French,

Model	***en→fr***	***fr→en***	***en→de***	***de→en***
		LSTM Cell		
Lample et al. (2018b)	24.28	23.74	14.71	19.60
Ours (Top-1 Extract)	24.43 (+0.15)	23.90 (+0.16)	14.54 (−0.17)	19.49 (−0.11)
Ours (Top-1 Extract + Edit)	24.54 (+0.26)	24.08 (+0.34)	14.63 (−0.08)	19.57 (−0.03)
Ours (Top-10 Extract)	26.12 (+1.84)	25.83 (+2.09)	17.01 (+2.30)	21.40 (+1.80)
Ours (Top-10 Extract + Edit)	**26.97** (+2.69)	**26.66** (+2.92)	**17.48** (+2.77)	**21.93** (+2.33)
		Transformer Cell		
Lample et al. (2018b)	25.14	24.18	17.16	21.00
Ours (Top-1 Extract)	25.30 (+0.16)	24.23 (+0.05)	17.12 (−0.04)	21.06 (+0.06)
Ours (Top-1 Extract + Edit)	25.44 (+0.30)	24.36 (+0.18)	17.14 (−0.02)	21.10 (+0.10)
Ours (Top-10 Extract)	26.91 (+1.77)	25.64 (+1.46)	19.11 (+1.95)	22.84 (+1.84)
Ours (Top-10 Extract + Edit)	**27.56** (+2.42)	**26.90** (+2.72)	**19.55** (+2.39)	**23.29** (+2.29)

Model	***en→ro***	***ro→en***	***en→ru***	***ru→en***
		LSTM Cell		
Lample et al. (2018b)	19.65	18.52	6.24	7.83
Ours (Top-1 Extract)	19.73 (+0.08)	18.56 (+0.04)	6.32 (+0.08)	7.99 (+0.16)
Ours (Top-1 Extract + Edit)	19.81 (+0.16)	18.69 (+0.17)	6.44 (+0.20)	8.12 (+0.29)
Ours (Top-10 Extract)	21.57 (+1.92)	20.32 (+1.80)	8.87 (+2.63)	9.76 (+1.93)
Ours (Top-10 Extract + Edit)	**22.08** (+2.43)	**20.83** (+2.31)	**9.35** (+3.11)	**10.21** (+2.38)
		Transformer Cell		
Lample et al. (2018b)	21.18	19.44	7.98	9.09
Ours (Top-1 Extract)	21.15 (−0.03)	19.52 (+0.08)	8.03 (+0.05)	9.20 (+0.11)
Ours (Top-1 Extract + Edit)	21.23 (+0.05)	19.59 (+0.15)	8.16 (+0.18)	9.28 (+0.19)
Ours (Top-10 Extract)	23.04 (+1.86)	21.43 (+1.99)	10.24 (+2.26)	12.29 (+3.20)
Ours (Top-10 Extract + Edit)	**23.31** (+2.13)	**21.60** (+2.16)	**11.07** (+3.09)	**12.72** (+3.63)

Table 1: The experimental results on all four language pairs and directions. The results are evaluated with BLEU metric on *newstest* 2014 for *en↔fr*and *newstest* 2016 for *en↔de*, *en↔ro* and *en↔ru*. The (+) and (−) stand for performance gains and loss separately compared with baseline models with the same NMT cells.

German and Russian, all the available sentences are used from the WMT monolingual News Crawl datasets from years 2007 through 2017. As for Romanian, we combine the News Crawl dataset and WMT'16 monolingual dataset. The translation results are evaluated on newstest 2014 for *en-fr*, and newstest 2016 for *en-de*, *en-ro* and *en-ru*.

4.2 Implementation Details

We follow previous methods (Koehn et al., 2007; Lample et al., 2018b) to initialize our models.

Initialization We use Moses scripts (Koehn et al., 2007) for tokenization. While the system requires cross-lingual BPE embeddings to initialize the shared lookup table for related languages, we set the number of BPE codes as $60,000$. Following the previous preprocessing protocol (Lample et al., 2018b), the embeddings are then generated using fastText (Bojanowski et al., 2017) with an embedding dimension of 512, a context window of size 5 and 10 negative samples.

Model Structure In this work, the NMT models can be built upon long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017) cells. For LSTM cells, both the encoder and decoder have 3 layers. As for Transformer, we use 4 layers both in the encoder and the decoder. As for both LSTM and Transformer, all encoder parameters are shared across two languages. Similarly, we share all decoder parameters across two languages. Both two model structure are optimized using Adam (Kingma and Ba, 2014) with a batch size of 32. The rate for LSTM cell is 0.0003 while Transformer's is set as 0.0001. The weights in Equation 7 are $\omega_{lm} = \omega_{ext} = 1$. The λ for calculating ranking scores is 0.5. As for the evaluation network R, we use a multilayer perceptron with two hidden layers of size 512. For efficient nearest neighbor search in the extracting step, we use the open-source Faiss library (Johnson et al., 2017)[2]. We calculate the similarity of sentences in

---

[2] `https://github.com/facebookresearch/faiss`

each episode instead of each batch for computational efficiency. At decoding time, sentences are generated using greedy decoding.

4.3 Results and Analysis

In this study, we aim to validate the effectiveness of extract-edit versus back-translation for unsupervised neural machine translation (NMT), so we set the unsupervised NMT method in Lample et al. (2018b) as the baseline because it currently achieves the state-of-the-art performance on all language pairs.[3] The overall translation results across four language pairs are shown in Table 1. In most of the cases, our proposed extract-edit approach can outperform the baseline models trained with back-translation. Our full models (LSTM/Transformer + Top-10 Extract + Edit) achieve more than 2 BLEU points improvement consistently across all the language pairs. Especially, on the *ru* → *en* translation with the Transformer cell, our full model surpasses the baseline score by 3.63 BLEU points. These results validate the effectiveness of our approach and indicate that the proposed extract-edit learning framework can learn a better mapping and alignment between language spaces than back-translation.

However, if extracting only top-1 target sentence in our approach, the performances are not always improved (e.g., *en* → *de*, *de* → *en*, and *en* → *ro*). Besides, *Top-10 Extract + Edit* models consistently outperforms *Top-1 Extract + Edit*. This is because more extracted-and-edited sentences lead to a higher recall, so more useful information will be used to guarantee the translation quality. The comparative translation loss can avoid the model suffering from the noise while taking advantage of more information. In other words, it is more likely to project the source sentence into the probable region in the target language space with more sentences serving as the anchor points, and the comparative learning scheme iteratively approximates towards more accurate target points. This highlights the importance of the extraction number k, which we further discuss next.

4.4 Ablation Study

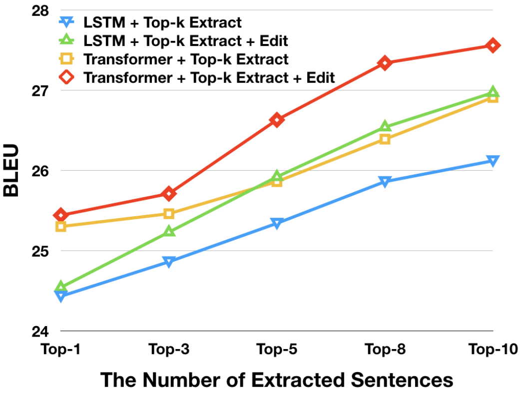


Figure 3: The effect of the number k of the extracted sentences in our approach on *en*→*fr* translation.

The Effect of Extraction Number k As shown in Table 1, the number k of the extracted-and-edited sentences plays a vital role in our approach. Thus for a more intuitive overview of its impact, we further train and evaluate multiple models with $k = 1, 3, 5, 8, 10$ on *en* → *fr* translation task. The detailed results are shown in Figure 3. This study shows that the translation performance of our approach is indeed improving as k increases. Because as we analyze above, larger k ensure a higher recall and thus more critical semantic information can be utilized to assist the translation. Besides, the diversity of extracted-and-edited sentences can potentially provide a more accurate localization of the probable region where the target sentence should be. Although we can infer that the models would perform even better with $k > 10$ from Figure 3, more computational resources will be required for that and we already observe a decelerated growth of BLEU scores from $k = 8$ to $k = 10$. Therefore, in this paper, we set $k = 10$ for the full models.

The Quality of Extraction Model In this section, we quantitatively evaluate the unsupervised extraction part of our model and compare it with the state-of-the-art supervised extraction model. Following Grégoire and Langlais (2018), we train a fully supervised parallel pair extraction model, where two Bi-LSTMs are implemented to encode sentences of two languages, and a feed-forward network is followed to culminate in a sigmoid output layer. The model is trained with around 500,000 English-French parallel sentence pairs sampled from Europarl corpus (Koehn, 2005). As for our unsupervised extraction model, we directly use the jointly trained extraction part in our frame-

---

[3] Note that for a fair comparison, we are not including the results of the unsupervised phrase-based statistical machine translation system (PBSMT). But theoretically, our extract-edit learning framework can be generalized to other types of machine translation systems such as PBSMT.

Noise	Model	Hits@1	Hits@3	Hits@5	Hits@8	Hits@10	Hits@15	Hits@20
0%	Supervised (Upperbound)	67.3	80.7	89.9	94.5	97.1	98.7	99.3
	Unsupervised (Ours)	52.2	54.6	68.8	80.2	89.1	91.8	93.3
50%	Supervised (Upperbound)	64.8	78.0	86.8	91.3	95.6	97.4	99.0
	Unsupervised (Ours)	46.9	49.7	62.1	73.4	83.2	87.6	89.2
90%	Supervised (Upperbound)	63.7	76.4	84.2	89.1	93.8	96.5	98.1
	Unsupervised (Ours)	41.5	46.8	58.0	69.3	77.2	83.9	87.8

Table 2: The experimental results of parallel sentence mining on the *newstest* 2012 *en* → *fr* translation dataset with different levels of added sentence noises. Metric: The percentage of Hits@k.

Cell	Learning	BLEU
LSTM	MLE Loss	12.40
	Comparative Loss	24.54
Transformer	MLE Loss	14.15
	Comparative Loss	25.44

Table 3: The performance of the unsupervised NMT systems with different learning objectives on *en* → *fr newstest* 2014.

work to extract the potential parallel sentences based on the scores computed by Equation 4. For evaluation, we sample $1,000$ parallel sentences from the *newstest* 2014 corpus and create three test sets with a noise ratio 0%, 50%, and 90% to simulate noisy real-world data. We report Hits@k results, which shows the percentage of the golden parallel sentences appear within the top-k place.

The detailed results are shown in Table 2. Although our extraction model structure is different from the supervised extraction model, it can still give us a good insight into the upperbound and gap of performance. We can observe a noticeable gap between unsupervised and supervised methods, but the gap is narrowing as the rank increases. Meanwhile, in our unsupervised method, the performance grows quickly when $k \leq 10$. From Table 2 we also notice that $k = 10$ is a sweet point, where the accuracy is high and the computational cost is relatively acceptable.

The Effect of Comparative Translation Finally, we aim to roughly evaluate the effect of the proposed comparable translation loss in our model. Thus, we compare our model with a two-staged NMT system, where we extract and edit the parallel pairs and retrain the NMT system with the standard maximum likelihood estimation (MLE) loss in a supervised way (by taking the extracted-and-edited sentences as the ground-truth targets). We compare the performance on the *en* → *fr newstest* 2014 dataset, and the results are shown in Table 3. We can observe that with the MLE loss, the translation performance will drop nearly 50%. The results indirectly reflect that the NMT systems are sensitive to noises in the training datasets. Meanwhile, it demonstrates by treating extracted-edit sentences as pivotal points instead of ground truth, our proposed comparative translation loss can avoid the NMT model suffers from the noise.

4.5 Discussion

Although our extract-edit approach can achieve better performance than the back-translation mechanism, it is still worth mentioning that our approach has more strict constraints on the domains of the source and target corpus. The extract-edit approach will work well when there is information overlap in the two language spaces. When there is little overlap in terms of domains, it will be much harder to find a good cluster of initial candidates, which may also complicate the editing process. As for the back-translation mechanism, it requires less overlap in terms of the language spaces because the language priors can be learnt in any domains. However, the corpus with matching domains can be easily obtained nowadays (e.g., Wikipedia and the news articles), which makes our extract-edit approach still widely applicable.

5 Related Work

Unsupervised NMT The current NMT systems (Sutskever et al., 2014; Cho et al., 2014a; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) are known to easily overfit and result in an inferior performance when the training data is limited (Koehn and Knowles, 2017; Isabelle et al., 2017; Sennrich, 2017). Many research efforts have been spent on how to utilize the monolingual data to improve the NMT system when only limited supervision is available (Gulcehre et al., 2015; Sennrich et al., 2016a; He et al., 2016; Zhang and Zong, 2016; Yang et al., 2018). Recently, Lample et al. (2018a); Artetxe et al. (2018); Lample et al. (2018b) make encouraging progress on unsupervised NMT structure mainly based on initialization, denoising language modeling, and back-translation. However, all these unsupervised models are based on the back-translation learning framework to generate pseudo language pairs for training. Our work leverages the information from real target language sentences.

Comparable Corpora Mining Comparable corpora mining aims at extracting parallel sentences from comparable monolingual corpora such as news stories written on the same topic in different languages. Most of the previous methods align the documents based on metadata and then extract parallel sentences using human-defined features (Munteanu and Marcu, 2002, 2006; Hewavitharana and Vogel, 2011). Recent neural-based methods (Chu et al., 2016; Grover and Mitra, 2017; Grégoire and Langlais, 2018) learn to identify parallel sentences in the semantic spaces. However, these methods require large amounts of parallel sentence pairs to train the systems first and then test the performance on raw comparable corpora, which does not apply to languages with limited resources. Instead, we explore the corpora mining in an unsupervised fashion and propose a joint training framework with machine translation.

Retrieval-Augmented Text Generation Our work is also related to the recent work on applying retrieval mechanisms to augment text generation, such as image captioning (Kuznetsova et al., 2013; Mason and Charniak, 2014), dialogue generation (Song et al., 2016; Yan et al., 2016; Wu et al., 2018) and style transfer (Lin et al., 2017; Li et al., 2018). Some editing-based models (Guu et al., 2018; Hashimoto et al., 2018) are proposed to further enhance the retrieved text. Recent work in machine translation (Gu et al., 2018) augments an NMT model with sentence pairs retrieved by an off-the-shelf search engine. However, these methods are two-staged with supervised retrieval first. In our work, the extracted-edited sentences are not directly used as the ground truth to train the translation model. Instead, we view these sentences as pivotal points in the target language space and further we propose a comparative translation loss to train the system in a fully unsupervised way.

6 Conclusion

In this paper, we propose an *extract-edit* approach, an effective alternative to the widely-used back-translation in unsupervised NMT. Instead of generating pseudo language pairs to train the systems with the reconstruction loss, we design a comparative translation loss that leverages real sentences in the target language space. Empirically, our method advances the previous state-of-the-art NMT systems across four language pairs using the monolingual corpora only. Theoretically, we believe the extract-edit learning framework can be generalized to other types of unsupervised machine translation systems and even some other unsupervised learning tasks.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful comments. The work was supported by the Facebook Low Resource Neural Machine Translation Research Award. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the 4th International Conference for Learning Representations (ICLR)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics (TACL)*, 5(1):135–146.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2016. Parallel sentence extraction from comparable corpora with neural network features. In *Proceedings of 9th Language Resources and Evaluation Conference (LREC)*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1243–1252.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of 28th Conference on neural information processing systems (NeurIPS)*, pages 2672–2680.

Francis Grégoire and Philippe Langlais. 2018. Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Jeenu Grover and Pabitra Mitra. 2017. Bilingual word embeddings with bucketed cnn for parallel sentence extraction. In *Proceedings of ACL 2017, Student Research Workshop*, pages 11–16.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided nonparametric neural machine translation. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI)*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics (TACL)*, 6:437–450.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Proceedings of 32th Conference on Neural Information Processing Systems (NeurIPS)*, pages 10073–10083.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 820–828.

Sanjika Hewavitharana and Stephan Vogel. 2011. Extracting parallel phrases from comparable data. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 61–68.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2486–2496.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Polina Kuznetsova, Vicente Ordonez, Alexander Berg, Tamara Berg, and Yejin Choi. 2013. Generalizing image captions for image-text parallel corpus. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 790–796.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5039–5049.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1865–1874.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Proceedings of the 31th Conference on Neural Information Processing Systems (NeurIP)*, pages 3155–3165.

Rebecca Mason and Eugene Charniak. 2014. Domain-specific image captioning. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL)*, pages 11–20.

Dragos Stefan Munteanu and Daniel Marcu. 2002. Processing comparable corpora with bilingual suffix trees. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dragos Stefan Munteanu and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 81–88.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 376–382.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems (NeurIPS)*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. *arXiv preprint arXiv:1806.07042*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR)*, pages 55–64.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 46–55.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545.

Consistency by Agreement in Zero-shot Neural Machine Translation

Maruan Al-Shedivat[*]
Carnegie Mellon University
Pittsburgh, PA 15213
`alshedivat@cs.cmu.edu`

Ankur P. Parikh
Google AI Language
New York, NY 10011
`aparikh@google.com`

Abstract

Generalization and reliability of multilingual translation often highly depend on the amount of available parallel data for each language pair of interest. In this paper, we focus on zero-shot generalization—a challenging setup that tests models on translation directions they have not been optimized for at training time. To solve the problem, we (i) reformulate multilingual translation as probabilistic inference, (ii) define the notion of zero-shot consistency and show why standard training often results in models unsuitable for zero-shot tasks, and (iii) introduce a consistent agreement-based training method that encourages the model to produce equivalent translations of parallel sentences in auxiliary languages. We test our multilingual NMT models on multiple public zero-shot translation benchmarks (IWSLT17, UN corpus, Europarl) and show that agreement-based learning often results in 2-3 BLEU zero-shot improvement over strong baselines without any loss in performance on supervised translation directions.

1 Introduction

Machine translation (MT) has made remarkable advances with the advent of deep learning approaches (Bojar et al., 2016; Wu et al., 2016; Crego et al., 2016; Junczys-Dowmunt et al., 2016). The progress was largely driven by the encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014) and typically supplemented with an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015b).

Compared to the traditional phrase-based systems (Koehn, 2009), neural machine translation (NMT) requires large amounts of data in order to reach high performance (Koehn and Knowles, 2017). Using NMT in a multilingual setting exacerbates the problem by the fact that given k languages translating between all pairs would require $O(k^2)$ parallel training corpora (and $O(k^2)$ models).

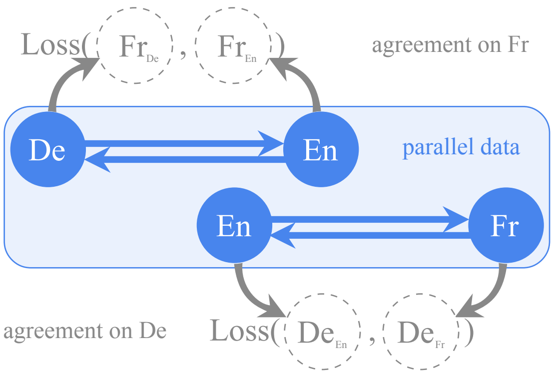


Figure 1: Agreement-based training of a multilingual NMT. At training time, given English-French (`En` ↔ `Fr`) and English-German (`En` ↔ `De`) parallel sentences, the model not only is trained to translate between the pair but also to agree on translations into a third language.

In an effort to address the problem, different multilingual NMT approaches have been proposed recently. Luong et al. (2015a); Firat et al. (2016a) proposed to use $O(k)$ encoders/decoders that are then intermixed to translate between language pairs. Johnson et al. (2016) proposed to use a single model and prepend special symbols to the source text to indicate the target language, which has later been extended to other text preprocessing approaches (Ha et al., 2017) as well as language-conditional parameter generation for encoders and decoders of a single model (Platanios et al., 2018).

Johnson et al. (2016) also show that a single multilingual system could potentially enable *zero-shot* translation, *i.e.*, it can translate between language pairs not seen in training. For example, given 3 languages—German (`De`), English (`En`), and French (`Fr`)—and training parallel data only for (`De`, `En`) and (`En`, `Fr`), at test time, the system could additionally translate between (`De`, `Fr`).

Zero-shot translation is an important problem. Solving the problem could significantly improve data efficiency—a single multilingual model would

---

[*] Work done at Google.

Proceedings of NAACL-HLT 2019, pages 1184–1197
Minneapolis, Minnesota, June 2 - June 7, 2019.

be able to generalize and translate between any of the $O(k^2)$ language pairs after being trained only on $O(k)$ parallel corpora. However, performance on zero-shot tasks is often unstable and significantly lags behind the supervised directions. Moreover, attempts to improve zero-shot performance by fine-tuning (Firat et al., 2016b; Sestorain et al., 2018) may negatively impact other directions.

In this work, we take a different approach and aim to improve the training procedure of Johnson et al. (2016). First, we analyze multilingual translation problem from a probabilistic perspective and define the notion of *zero-shot consistency* that gives insights as to why the vanilla training method may not yield models with good zero-shot performance. Next, we propose a novel training objective and a modified learning algorithm that achieves consistency via agreement-based learning (Liang et al., 2006, 2008) and improves zero-shot translation. Our training procedure encourages the model to produce equivalent translations of parallel training sentences into an auxiliary language (Figure 1) and is provably zero-shot consistent. In addition, we make a simple change to the neural decoder to make the agreement losses fully differentiable.

We conduct experiments on IWSLT17 (Mauro et al., 2017), UN corpus (Ziemski et al., 2016), and Europarl (Koehn, 2017), carefully removing complete pivots from the training corpora. Agreement-based learning results in up to +3 BLEU zero-shot improvement over the baseline, compares favorably (up to +2.4 BLEU) to other approaches in the literature (Cheng et al., 2017; Sestorain et al., 2018), is competitive with pivoting, and does not lose in performance on supervised directions.

2 Related work

A simple (and yet effective) baseline for zero-shot translation is pivoting that chain-translates, first to a pivot language, then to a target (Cohn and Lapata, 2007; Wu and Wang, 2007; Utiyama and Isahara, 2007). Despite being a pipeline, pivoting gets better as the supervised models improve, which makes it a strong baseline in the zero-shot setting. Cheng et al. (2017) proposed a joint pivoting learning strategy that leads to further improvements.

Lu et al. (2018) and Arivazhagan et al. (2018) proposed different techniques to obtain "neural interlingual" representations that are passed to the decoder. Sestorain et al. (2018) proposed another fine-tuning technique that uses dual learning (He et al., 2016), where a language model is used to provide a signal for fine-tuning zero-shot directions.

Another family of approaches is based on distillation (Hinton et al., 2014; Kim and Rush, 2016). Along these lines, Firat et al. (2016b) proposed to fine tune a multilingual model to a specified zero-shot-direction with pseudo-parallel data and Chen et al. (2017) proposed a teacher-student framework. While this can yield solid performance improvements, it also adds multi-staging overhead and often does not preserve performance of a single model on the supervised directions. We note that our approach (and agreement-based learning in general) is somewhat similar to distillation at training time, which has been explored for large-scale single-task prediction problems (Anil et al., 2018).

A setting harder than zero-shot is that of fully unsupervised translation (Ravi and Knight, 2011; Artetxe et al., 2017; Lample et al., 2017, 2018) in which no parallel data is available for training. The ideas proposed in these works (*e.g.*, bilingual dictionaries (Conneau et al., 2017), backtranslation (Sennrich et al., 2015a) and language models (He et al., 2016)) are complementary to our approach, which encourages agreement among different translation directions in the zero-shot multilingual setting.

3 Background

We start by establishing more formal notation and briefly reviewing some background on encoder-decoder multilingual machine translation from a probabilistic perspective.

3.1 Notation

Languages. We assume that we are given a collection of k languages, $L_1, \dots, L_k$, that share a common vocabulary, V. A language, L_i, is defined by the marginal probability $\mathbb{P}(\mathbf{x}_i)$ it assigns to sentences (*i.e.*, sequences of tokens from the vocabulary), denoted $\mathbf{x}_i := (x^1, \dots, x^l)$, where l is the length of the sequence. All languages together define a joint probability distribution, $\mathbb{P}(\mathbf{x}_1, \dots, \mathbf{x}_k)$, over k-tuples of *equivalent sentences*.

Corpora. While each sentence may have an equivalent representation in all languages, we assume that we have access to only partial sets of equivalent sentences, which form *corpora*. In this work, we consider *bilingual* corpora, denoted $\mathcal{C}_{ij}$, that contain pairs of sentences sampled from $\mathbb{P}(\mathbf{x}_i, \mathbf{x}_j)$ and *monolingual* corpora, denoted $\mathcal{C}_i$, that contain sentences sampled from $\mathbb{P}(\mathbf{x}_i)$.

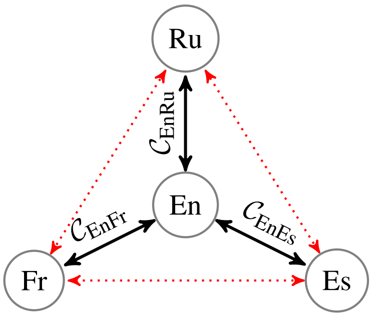


Figure 2: Translation graph: Languages (nodes), parallel corpora (solid edges), and zero-shot directions (dotted edges).

Translation. Finally, we define a *translation task* from language L_i to L_j as learning to model the conditional distribution $\mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i\right)$. The set of k languages along with translation tasks can be represented as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a set of k nodes, $\mathcal{V}$, that represent languages and edges, $\mathcal{E}$, that indicate translation directions. We further distinguish between two disjoint subsets of edges: (i) supervised edges, $\mathcal{E}_s$, for which we have parallel data, and (ii) zero-shot edges, $\mathcal{E}_0$, that correspond to zero-shot translation tasks. Figure 2 presents an example translation graph with supervised edges (`En` $\leftrightarrow$ `Es`, `En` $\leftrightarrow$ `Fr`, `En` $\leftrightarrow$ `Ru`) and zero-shot edges (`Es` $\leftrightarrow$ `Fr`, `Es` $\leftrightarrow$ `Ru`, `Fr` $\leftrightarrow$ `Ru`). We will use this graph as our running example.

3.2 Encoder-decoder framework

First, consider a purely bilingual setting, where we learn to translate from a source language, L_s, to a target language, L_t. We can train a translation model by optimizing the conditional log-likelihood of the bilingual data under the model:

$$\hat{\theta} := \arg\max_\theta \sum\nolimits_{\mathcal{C}_{st}} \log \mathbb{P}_\theta\left(\mathbf{x}_t \mid \mathbf{x}_s\right) \qquad (1)$$

where $\hat{\theta}$ are the estimated parameters of the model.

The encoder-decoder framework introduces a latent sequence, $\mathbf{u}$, and represents the model as:

$$\mathbb{P}_\theta\left(\mathbf{x}_t \mid \mathbf{x}_s\right) = \mathbb{P}_\theta^{\text{dec}}\left(\mathbf{x}_t \mid \mathbf{u} = f_\theta^{\text{enc}}(\mathbf{x}_s)\right) \qquad (2)$$

where $f_\theta^{\text{enc}}(\mathbf{x}_s)$ is the encoder that maps a source sequence to a sequence of latent representations, $\mathbf{u}$, and the decoder defines $\mathbb{P}_\theta^{\text{dec}}\left(\mathbf{x}_t \mid \mathbf{u}\right)$.[1] Note that $\mathbf{u}$ is usually deterministic with respect to $\mathbf{x}_s$ and accurate representation of the conditional distribution highly depends on the decoder. In neural machine translation, the exact forms of encoder and decoder are specified using RNNs (Sutskever et al., 2014), CNNs (Gehring et al., 2016), and attention (Bahdanau et al., 2014; Vaswani et al., 2017) as building blocks. The decoding distribution, $\mathbb{P}_\theta^{\text{dec}}\left(\mathbf{x}_t \mid \mathbf{u}\right)$, is typically modeled autoregressively.

3.3 Multilingual neural machine translation

In the multilingual setting, we would like to learn to translate in *all directions* having access to only few parallel bilingual corpora. In other words, we would like to learn a collection of models, $\{\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)\}_{i,j\in\mathcal{E}}$. We can assume that models are independent and choose to learn them by maximizing the following objective:

$$\mathcal{L}^{\text{ind}}(\theta) = \sum_{i,j\in\mathcal{E}_s} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}_{ij}} \log \mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right) \qquad (3)$$

In the statistics literature, this estimation approach is called *maximum composite likelihood* (Besag, 1975; Lindsay, 1988) as it composes the objective out of (sometimes weighted) terms that represent conditional sub-likelihoods (in our example, $\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)$). Composite likelihoods are easy to construct and tractable to optimize as they do not require representing the full likelihood, which would involve integrating out variables unobserved in the data (see Appendix A.1).

Johnson et al. (2016) proposed to train a multilingual NMT systems by optimizing a composite likelihood objective (3) while representing all conditional distributions, $\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)$, with a *shared* encoder and decoder and using language tags, l_t, to distinguish between translation directions:

$$\mathbb{P}\left(\mathbf{x}_t \mid \mathbf{x}_s\right) = \mathbb{P}_\theta^{\text{dec}}\left(\mathbf{x}_t \mid \mathbf{u}_{st} = f_\theta^{\text{enc}}(\mathbf{x}_s, l_t)\right) \qquad (4)$$

This approach has numerous advantages including: (a) simplicity of training and the architecture (by slightly changing the training data, we convert a bilingual NMT into a multilingual one), (b) sharing parameters of the model between different translation tasks that may lead to better and more robust representations. Johnson et al. (2016) also show that resulting models seem to exhibit some degree of zero-shot generalization enabled by parameter sharing. However, since we lack data for zero-shot directions, composite likelihood (3) misses the terms that correspond to the zero-shot models, and hence has no statistical guarantees for performance on zero-shot tasks.[2]

---

[1] Slightly abusing the notation, we use θ to denote all parameters of the model: embeddings, encoder, and decoder.

[2] In fact, since the objective (3) assumes that the models are independent, plausible zero-shot performance would be more indicative of the limited capacity of the model or artifacts in the data (*e.g.*, presence of multi-parallel sentences) rather than zero-shot generalization.

4 Zero-shot generalization & consistency

Multilingual MT systems can be evaluated in terms of *zero-shot performance*, or quality of translation along the directions they have not been optimized for (*e.g.*, due to lack of data). We formally define zero-shot generalization via consistency.

Definition 1 (Expected Zero-shot Consistency) *Let $\mathcal{E}_s$ and $\mathcal{E}_0$ be supervised and zero-shot tasks, respectively. Let $\ell(\cdot)$ be a non-negative loss function and $\mathcal{M}$ be a model with maximum expected supervised loss bounded by some $\varepsilon > 0$:*

$$\max_{(i,j)\in\mathcal{E}_s} \mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j}\left[\ell(\mathcal{M})\right] < \varepsilon$$

We call $\mathcal{M}$ zero-shot consistent with respect to $\ell(\cdot)$ if for some $\kappa(\varepsilon) > 0$

$$\max_{(i,j)\in\mathcal{E}_0} \mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j}\left[\ell(\mathcal{M})\right] < \kappa(\varepsilon),$$

where $\kappa(\varepsilon) \to 0$ as $\varepsilon \to 0$.

In other words, we say that a machine translation system is zero-shot consistent if low error on supervised tasks implies a low error on zero-shot tasks in expectation (*i.e.*, the system generalizes). We also note that our notion of consistency somewhat resembles error bounds in the domain adaptation literature (Ben-David et al., 2010).

In practice, it is attractive to have MT systems that are guaranteed to exhibit zero-shot generalization since the access to parallel data is always limited and training is computationally expensive. While the training method of Johnson et al. (2016) does not have guarantees, we show that our proposed approach is provably zero-shot consistent.

5 Approach

We propose a new training objective for multilingual NMT architectures with shared encoders and decoders that avoids the limitations of pure composite likelihoods. Our method is based on the idea of agreement-based learning initially proposed for learning consistent alignments in phrase-based statistical machine translation (SMT) systems (Liang et al., 2006, 2008). In terms of the final objective function, the method ends up being reminiscent of distillation (Kim and Rush, 2016), but suitable for joint multilingual training.

5.1 Agreement-based likelihood

To introduce agreement-based objective, we use the graph from Figure 2 that defines translation tasks between 4 languages ($\texttt{En}$, $\texttt{Es}$, $\texttt{Fr}$, $\texttt{Ru}$). In particular, consider the composite likelihood objective (3) for a pair of $\texttt{En} - \texttt{Fr}$ sentences, $(\mathbf{x}_{\texttt{En}}, \mathbf{x}_{\texttt{Fr}})$:

$$\begin{aligned}
&\mathcal{L}^{\text{ind}}_{\texttt{EnFr}}(\theta) \qquad (5)\\
&\quad = \log\left[\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}} \mid \mathbf{x}_{\texttt{En}}\right)\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{En}} \mid \mathbf{x}_{\texttt{Fr}}\right)\right]\\
&\quad = \log\Bigg[\sum_{\mathbf{z}'_{\texttt{Es}},\mathbf{z}'_{\texttt{Ru}}} \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}}, \mathbf{z}'_{\texttt{Es}}, \mathbf{z}'_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{En}}\right) \times\\
&\qquad\qquad \sum_{\mathbf{z}''_{\texttt{Es}},\mathbf{z}''_{\texttt{Ru}}} \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{En}}, \mathbf{z}''_{\texttt{Es}}, \mathbf{z}''_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{Fr}}\right)\Bigg]
\end{aligned}$$

where we introduced latent translations into Spanish ($\texttt{Es}$) and Russian ($\texttt{Ru}$) and marginalized them out (by virtually summing over all sequences in the corresponding languages). Again, note that this objective assumes independence of $\texttt{En} \to \texttt{Fr}$ and $\texttt{Fr} \to \texttt{En}$ models.

Following Liang et al. (2008), we propose to tie together the single prime and the double prime latent variables, $\mathbf{z}_{\texttt{Es}}$ and $\mathbf{z}_{\texttt{Ru}}$, to encourage agreement between $\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{En}}, \mathbf{z}_{\texttt{Es}}, \mathbf{z}_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{Fr}}\right)$ and $\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}}, \mathbf{z}_{\texttt{Es}}, \mathbf{z}_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{En}}\right)$ on the latent translations. We interchange the sum and the product operations inside the $\log$ in (5), denote $\mathbf{z} := (\mathbf{z}_{\texttt{Es}}, \mathbf{z}_{\texttt{Ru}})$ to simplify notation, and arrive at the following new objective function:

$$\begin{aligned}
&\mathcal{L}^{\text{agree}}_{\texttt{EnFr}}(\theta) := \qquad (6)\\
&\quad \log \sum_{\mathbf{z}} \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}}, \mathbf{z} \mid \mathbf{x}_{\texttt{En}}\right)\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{En}}, \mathbf{z} \mid \mathbf{x}_{\texttt{Fr}}\right)
\end{aligned}$$

Next, we factorize each term as:

$$\mathbb{P}\left(\mathbf{x}, \mathbf{z} \mid \mathbf{y}\right) = \mathbb{P}\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}\right)\mathbb{P}_\theta\left(\mathbf{z} \mid \mathbf{y}\right)$$

Assuming $\mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}} \mid \mathbf{z}, \mathbf{x}_{\texttt{En}}\right) \approx \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}} \mid \mathbf{x}_{\texttt{En}}\right)$,[3] the objective (6) decomposes into two terms:

$$\begin{aligned}
&\mathcal{L}^{\text{agree}}_{\texttt{EnFr}}(\theta) \qquad (7)\\
&\quad \approx \underbrace{\log \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{Fr}} \mid \mathbf{x}_{\texttt{En}}\right) + \log \mathbb{P}_\theta\left(\mathbf{x}_{\texttt{En}} \mid \mathbf{x}_{\texttt{Fr}}\right)}_{\text{composite likelihood terms}} +\\
&\qquad \underbrace{\log \sum_{\mathbf{z}} \mathbb{P}_\theta\left(\mathbf{z} \mid \mathbf{x}_{\texttt{En}}\right)\mathbb{P}_\theta\left(\mathbf{z} \mid \mathbf{x}_{\texttt{Fr}}\right)}_{\text{agreement term}}
\end{aligned}$$

---

[3] This means that it is sufficient to condition on a sentence in one of the languages to determine probability of a translation in any other language.

We call the expression given in (7) *agreement-based likelihood*. Intuitively, this objective is the likelihood of observing parallel sentences $(\mathbf{x}_{\texttt{En}}, \mathbf{x}_{\texttt{Fr}})$ *and* having sub-models $\mathbb{P}_\theta(\mathbf{z} \mid \mathbf{x}_{\texttt{En}})$ and $\mathbb{P}_\theta(\mathbf{z} \mid \mathbf{x}_{\texttt{Fr}})$ agree on all translations into `Es` and `Ru` at the same time.

Lower bound. Summation in the agreement term over $\mathbf{z}$ (*i.e.*, over possible translations into `Es` and `Ru` in our case) is intractable. Switching back from $\mathbf{z}$ to $(\mathbf{z}_{\texttt{Es}}, \mathbf{z}_{\texttt{Ru}})$ notation and using Jensen's inequality, we lower bound it with cross-entropy:[4]

$$\begin{aligned} \log \sum_{\mathbf{z}} \mathbb{P}_\theta(\mathbf{z} \mid \mathbf{x}_{\texttt{En}}) \, \mathbb{P}_\theta(\mathbf{z} \mid \mathbf{x}_{\texttt{Fr}}) & \\ \geq \mathbb{E}_{\mathbf{z}_{\texttt{Es}} \mid \mathbf{x}_{\texttt{En}}} \left[\log \mathbb{P}_\theta(\mathbf{z}_{\texttt{Es}} \mid \mathbf{x}_{\texttt{Fr}}) \right] + & \\ \mathbb{E}_{\mathbf{z}_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{En}}} \left[\log \mathbb{P}_\theta(\mathbf{z}_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{Fr}}) \right] & \end{aligned} \tag{8}$$

We can estimate the expectations in the lower bound on the agreement terms by sampling $\mathbf{z}_{\texttt{Es}} \sim \mathbb{P}_\theta(\mathbf{z}_{\texttt{Es}} \mid \mathbf{x}_{\texttt{En}})$ and $\mathbf{z}_{\texttt{Ru}} \sim \mathbb{P}_\theta(\mathbf{z}_{\texttt{Ru}} \mid \mathbf{x}_{\texttt{En}})$. In practice, instead of sampling we use greedy, continuous decoding (with a fixed maximum sequence length) that also makes $\mathbf{z}_{\texttt{Es}}$ and $\mathbf{z}_{\texttt{Ru}}$ differentiable with respect to parameters of the model.

5.2 Consistency by agreement

We argue that models produced by maximizing agreement-based likelihood (7) are zero-shot consistent. Informally, consider again our running example from Figure 2. Given a pair of parallel sentences in $(\texttt{En}, \texttt{Fr})$, agreement loss encourages translations from `En` to $\{\texttt{Es}, \texttt{Ru}\}$ and translations from `Fr` to $\{\texttt{Es}, \texttt{Ru}\}$ to coincide. Note that $\texttt{En} \to \{\texttt{Es}, \texttt{Fr}, \texttt{Ru}\}$ are supervised directions. Therefore, agreement ensures that translations along the zero-shot edges in the graph match supervised translations. Formally, we state it as:

Theorem 2 (Agreement Zero-shot Consistency) *Let L_1, L_2, and L_3 be a collection of languages with $L_1 \leftrightarrow L_2$ and $L_2 \leftrightarrow L_3$ be supervised while $L_1 \leftrightarrow L_3$ be a zero-shot direction. Let $\mathbb{P}_\theta(\mathbf{x}_j \mid \mathbf{x}_i)$ be sub-models represented by a multilingual MT system. If the expected agreement-based loss, $\mathbb{E}_{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3}[\mathcal{L}_{12}^{\text{agree}}(\theta) + \mathcal{L}_{23}^{\text{agree}}(\theta)]$, is bounded by some $\varepsilon > 0$, then, under some mild technical assumptions on the true distribution of the equivalent translations, the zero-shot cross-entropy loss is bounded as follows:*

$$\mathbb{E}_{\mathbf{x}_1, \mathbf{x}_3} \left[-\log \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_1) \right] \leq \kappa(\varepsilon)$$

where $\kappa(\varepsilon) \to 0$ as $\varepsilon \to 0$.

Algorithm 1 Agreement-based M-NMT training

```
input Architecture (GNMT), agreement coefficient ($\gamma$)
Initialize: $\theta \leftarrow \theta_0$
while not (converged or step limit reached) do
    Get a mini-batch of parallel src-tgt pairs, $(\mathbf{X}_s, \mathbf{X}_t)$
    Supervised loss: $\mathcal{L}^{\text{sup}}(\theta) \leftarrow \log \mathbb{P}_\theta(\mathbf{X}_t \mid \mathbf{X}_s)$
    Auxiliary languages: $L_a \sim \text{Unif}(\{1, \ldots, k\})$
    Auxiliary translations:
        $\mathbf{Z}_{a \leftarrow s} \leftarrow \text{Decode}(\mathbf{Z}_a \mid f_\theta^{\text{enc}}(\mathbf{X}_s, L_a))$
        $\mathbf{Z}_{a \leftarrow t} \leftarrow \text{Decode}(\mathbf{Z}_a \mid f_\theta^{\text{enc}}(\mathbf{X}_t, L_a))$
    Agreement log-probabilities:
        $\ell_{a \leftarrow s}^t \leftarrow \log \mathbb{P}_\theta(\mathbf{Z}_{a \leftarrow s} \mid \mathbf{X}_t)$
        $\ell_{a \leftarrow t}^s \leftarrow \log \mathbb{P}_\theta(\mathbf{Z}_{a \leftarrow t} \mid \mathbf{X}_s)$
    Apply stop-gradients to supervised $\ell_{a \leftarrow s}^t$ and $\ell_{a \leftarrow t}^s$
    Total loss: $\mathcal{L}^{\text{total}}(\theta) \leftarrow \mathcal{L}^{\text{sup}}(\theta) + \gamma(\ell_{a \leftarrow s}^t + \ell_{a \leftarrow t}^s)$
    Update: $\theta \leftarrow \text{optimizer\_update}(\mathcal{L}^{\text{total}}, \theta)$
end while
output $\theta$
```

For discussion of the assumptions and details on the proof of the bound, see Appendix A.2. Note that Theorem 2 is straightforward to extend from triplets of languages to arbitrary connected graphs, as given in the following corollary.

Corollary 3 *Agreement-based learning yields zero shot consistent MT models (with respect to the cross entropy loss) for arbitrary translation graphs as long as supervised directions span the graph.*

Alternative ways to ensure consistency. Note that there are other ways to ensure zero-shot consistency, *e.g.*, by fine-tuning or post-processing a trained multilingual model. For instance, pivoting through an intermediate language is also zero-shot consistent, but the proof requires stronger assumptions about the quality of the supervised source-pivot model.[5] Similarly, using model distillation (Kim and Rush, 2016; Chen et al., 2017) would be also provably consistent under the same assumptions as given in Theorem 2, but for a single, pre-selected zero-shot direction. Note that our proposed agreement-based learning framework is provably consistent for *all* zero-shot directions and does not require any post-processing. For discussion of the alternative approaches and consistency proof for pivoting, see Appendix A.3.

---

[4] Note that expectations in (8) are conditional on $\mathbf{x}_{\texttt{En}}$. Symmetrically, we can have a lower bound with expectations conditional on $\mathbf{x}_{\texttt{Fr}}$. In practice, we symmetrize the objective.

[5] Intuitively, we have to assume that source-pivot model does not assign high probabilities to unlikely translations as the pivot-target model may react to those unpredictably.

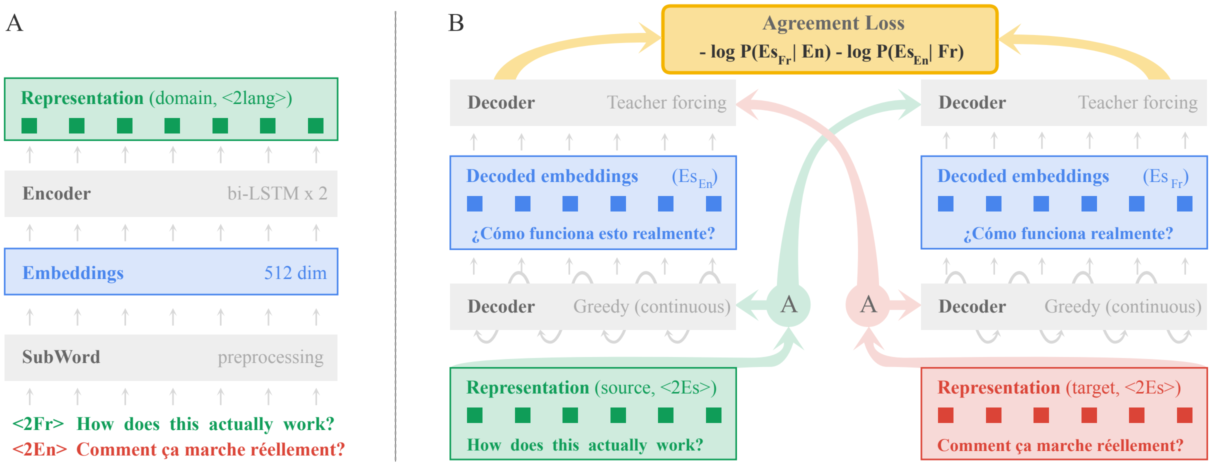


Figure 3: **A.** Computation graph for the encoder. The representations depend on the input sequence and the target language tag. **B.** Computation graph for the agreement loss. First, encode source and target sequences with the auxiliary language tags. Next, decode $\mathbf{z}_{\texttt{Es}}$ from both $\mathbf{x}_{\texttt{En}}$ and $\mathbf{x}_{\texttt{Fr}}$ using continuous greedy decoder. Finally, evaluate log probabilities, $\log \mathbb{P}_\theta\left(\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}}) \mid \mathbf{x}_{\texttt{Fr}}\right)$ and $\log \mathbb{P}_\theta\left(\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}}) \mid \mathbf{x}_{\texttt{En}}\right)$, and compute a sample estimate of the agreement loss.

5.3 Agreement-based learning algorithm

Having derived a new objective function (7), we can now learn consistent multilingual NMT models using stochastic gradient method with a couple of extra tricks (Algorithm 1). The computation graph for the agreement loss is given in Figure 3.

Subsampling auxiliary languages. Computing agreement over *all* languages for each pair of sentences at training time would be quite computationally expensive (to agree on k translations, we would need to encode-decode the source and target sequences k times each). However, since the agreement lower bound is a sum over expectations (8), we can approximate it by subsampling: at each training step (and for each sample in the mini-batch), we pick an auxiliary language uniformly at random and compute stochastic approximation of the agreement lower bound (8) for that language only. This stochastic approximation is simple, unbiased, and reduces per step computational overhead for the agreement term from $O(k)$ to $O(1)$.[6]

Overview of the agreement loss computation. Given a pair of parallel sentences, $\mathbf{x}_{\texttt{En}}$ and $\mathbf{x}_{\texttt{Fr}}$, and an auxiliary language, say $\texttt{Es}$, an estimate of the lower bound on the agreement term (8) is computed as follows. First, we concatenate $\texttt{Es}$ language tags to both $\mathbf{x}_{\texttt{En}}$ and $\mathbf{x}_{\texttt{Fr}}$ and encode the sequences so that both can be translated into $\texttt{Es}$ (the encoding process is depicted in Figure 3A). Next, we decode each of the encoded sentences and obtain auxiliary translations, $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}})$ and $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}})$, depicted as blue blocks in Figure 3B. Note that we now can treat pairs $(\mathbf{x}_{\texttt{Fr}}, \mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}}))$ and $(\mathbf{x}_{\texttt{En}}, \mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}}))$ as new parallel data for $\texttt{En} \to \texttt{Es}$ and $\texttt{Fr} \to \texttt{Es}$.

Finally, using these pairs, we can compute two log-probability terms (Figure 3B):

$$\begin{aligned} &\log \mathbb{P}_\theta\left(\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}}) \mid \mathbf{x}_{\texttt{En}}\right) \\ &\log \mathbb{P}_\theta\left(\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}}) \mid \mathbf{x}_{\texttt{Fr}}\right) \end{aligned} \tag{9}$$

using encoding-decoding with teacher forcing (same way as typically done for the supervised directions). Crucially, note that $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}})$ corresponds to a supervised direction, $\texttt{En} \to \texttt{Es}$, while $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}})$ corresponds to zero-shot, $\texttt{Fr} \to \texttt{Es}$. We want each of the components to (i) improve the zero-shot direction while (ii) minimally affecting the supervised direction. To achieve (i), we use continuous decoding, and for (ii) we use stop-gradient-based protection of the supervised directions. Both techniques are described below.

Greedy continuous decoding. In order to make $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}})$ and $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}})$ differentiable with respect to θ (hence, *continuous* decoding), at each decoding step t, we treat the output of the RNN, $\mathbf{h}^t$, as the key and use dot-product attention over the embedding vocabulary, $\mathbf{V}$, to construct $\mathbf{z}_{\texttt{Es}}^t$:

$$\mathbf{z}_{\texttt{Es}}^t := \operatorname{softmax}\left\{(\mathbf{h}^t)^\top \mathbf{V}\right\} \mathbf{V} \tag{10}$$

In other words, auxiliary translations, $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{En}})$ and $\mathbf{z}_{\texttt{Es}}(\mathbf{x}_{\texttt{Fr}})$, are fixed length sequences of differentiable embeddings computed in a greedy fashion.

---

[6] In practice, note that there is still a constant factor overhead due to extra encoding-decoding steps to/from auxiliary languages, which is about ×4 when training on a single GPU. Parallelizing the model across multiple GPUs would easily compensate this overhead.

Protecting supervised directions. Algorithm 1 scales agreement losses by a small coefficient γ. We found experimentally that training could be sensitive to this hyperparameter since the agreement loss also affects the supervised sub-models. For example, agreement of `En` $\to$ `Es` (supervised) and `Fr` $\to$ `Es` (zero-shot) may push the former towards a worse translation, especially at the beginning of training. To stabilize training, we apply the `stop_gradient` operator to the log probabilities and samples produced by the supervised sub-models before computing the agreement terms (9), to zero-out the corresponding gradient updates.

6 Experiments

We evaluate agreement-based training against baselines from the literature on three public datasets that have multi-parallel *evaluation data* that allows assessing zero-shot performance. We report results in terms of the BLEU score (Papineni et al., 2002) that was computed using `mteval-v13a.perl`.

6.1 Datasets

UN corpus. Following the setup introduced in Sestorain et al. (2018), we use two datasets, *UNcorpus-1* and *UNcorpus-2*, derived from the United Nations Parallel Corpus (Ziemski et al., 2016). *UNcorpus-1* consists of data in 3 languages, `En`, `Es`, `Fr`, where *UNcorpus-2* has `Ru` as the 4th language. For training, we use parallel corpora between `En` and the rest of the languages, each about 1M sentences, sub-sampled from the official training data in a way that ensures no multi-parallel training data. The *dev* and *test* sets contain 4,000 sentences and are all multi-parallel.

Europarl v7[7]**.** We consider the following languages: `De`, `En`, `Es`, `Fr`. For training, we use parallel data between `En` and the rest of the languages (about 1M sentences per corpus), preprocessed to avoid multi-parallel sentences, as was also done by Cheng et al. (2017) and Chen et al. (2017) and described below. The *dev* and *test* sets contain 2,000 multi-parallel sentences.

IWSLT17[8]**.** We use data from the official multilingual task: 5 languages (`De`, `En`, `It`, `Nl`, `Ro`), 20 translation tasks of which 4 zero-shot (`De` $\leftrightarrow$ `Nl` and `It` $\leftrightarrow$ `Ro`) and the rest 16 supervised. Note that this dataset has a significant overlap between parallel corpora in the supervised directions (up to 100K sentence pairs per direction). This implicitly makes the dataset multi-parallel and defeats the purpose of zero-shot evaluation (Dabre et al., 2017). To avoid spurious effects, we also derived **IWSLT17*** dataset from the original one by restricting supervised data to only `En` $\leftrightarrow \{$`De`, `Nl`, `It`, `Ro`$\}$ and removing overlapping pivoting sentences. We report results on both the official and preprocessed datasets.

Preprocessing. To properly evaluate systems in terms of zero-shot generalization, we preprocess Europarl and IWSLT* to avoid multi-lingual parallel sentences of the form *source-pivot-target*, where *source-target* is a zero-shot direction. To do so, we follow Cheng et al. (2017); Chen et al. (2017) and randomly split the overlapping pivot sentences of the original *source-pivot* and *pivot-target* corpora into two parts and merge them separately with the non-overlapping parts for each pair. Along with each parallel training sentence, we save information about source and target tags, after which all the data is combined and shuffled. Finally, we use a shared multilingual subword vocabulary (Sennrich et al., 2015b) on the training data (with 32K merge ops), separately for each dataset. Data statistics are provided in Appendix A.5.

6.2 Training and evaluation

Additional details on the hyperparameters can be found in Appendix A.4.

Models. We use a smaller version of the GNMT architecture (Wu et al., 2016) in all our experiments: 512-dimensional embeddings (separate for source and target sides), 2 bidirectional LSTM layers of 512 units each for encoding, and GNMT-style, 4-layer, 512-unit LSMT decoder with residual connections from the 2nd layer onward.

Training. We trained the above model using the *standard method* of Johnson et al. (2016) and using our proposed *agreement-based* training (Algorithm 1). In both cases, the model was optimized using Adafactor (Shazeer and Stern, 2018) on a machine with 4 P100 GPUs for up to 500K steps, with early stopping on the dev set.

Evaluation. We focus our evaluation mainly on zero-shot performance of the following methods:
(a) `Basic`, which stands for directly evaluating a multilingual GNMT model after standard training (Johnson et al., 2016).

---

[7] `http://www.statmt.org/europarl/`

[8] `https://sites.google.com/site/iwsltevaluation2017/TED-tasks`

	Sestorain et al. (2018)†			Our baselines		
	PBSMT	NMT-0	Dual-0	Basic	Pivot	Agree
En → Es	61.26	51.93	—	56.58	56.58	56.36
En → Fr	50.09	40.56	—	44.27	44.27	44.80
Es → En	59.89	51.58	—	55.70	55.70	55.24
Fr → En	52.22	43.33	—	46.46	46.46	46.17
Supervised (avg.)	55.87	46.85	—	50.75	50.75	50.64
Es → Fr	52.44	20.29	36.68	34.75	**38.10**	37.54
Fr → Es	49.79	19.01	39.19	37.67	**40.84**	40.02
Zero-shot (avg.)	51.11	19.69	37.93	36.21	**39.47**	38.78

†Source: https://openreview.net/forum?id=ByecAoAqK7.

Table 1: Results on UNCorpus-1.

	Sestorain et al. (2018)			Our baselines		
	PBSMT	NMT-0	Dual-0	Basic	Pivot	Agree
En → Es	61.26	47.51	44.30	55.15	55.15	54.30
En → Fr	50.09	36.70	34.34	43.42	43.42	42.57
En → Ru	43.25	30.45	29.47	36.26	36.26	35.89
Es → En	59.89	48.56	45.55	54.35	54.35	54.33
Fr → En	52.22	40.75	37.75	45.55	45.55	45.87
Ru → En	52.59	39.35	37.96	45.52	45.52	44.67
Supervised (avg.)	53.22	40.55	36.74	46.71	46.71	46.27
Es → Fr	52.44	25.85	34.51	34.73	35.93	**36.02**
Fr → Es	49.79	22.68	37.71	38.20	39.51	**39.94**
Es → Ru	39.69	9.36	24.55	26.29	27.15	**28.08**
Ru → Es	49.61	26.26	33.23	33.43	**37.17**	35.01
Fr → Ru	36.48	9.35	22.76	23.88	24.99	**25.13**
Ru → Fr	43.37	22.43	26.49	28.52	**30.06**	29.53
Zero-shot (avg.)	45.23	26.26	29.88	30.84	**32.47**	32.29

Table 2: Results on UNCorpus-2.

	Previous work		Our baselines		
	Soft‡	Distill†	Basic	Pivot	Agree
En → Es	—	—	34.69	34.69	33.80
En → De	—	—	23.06	23.06	22.44
En → Fr	31.40	—	33.87	33.87	32.55
Es → En	31.96	—	34.77	34.77	34.53
De → En	26.55	—	29.06	29.06	29.07
Fr → En	—	—	33.67	33.67	33.30
Supervised (avg.)	—	—	31.52	31.52	30.95
Es → De	—	—	18.23	20.14	**20.70**
De → Es	—	—	20.28	**26.50**	22.45
Es → Fr	30.57	**33.86**	27.99	32.56	30.94
Fr → Es	—	—	27.12	**32.96**	29.91
De → Fr	23.79	**27.03**	21.36	25.67	24.45
Fr → De	—	—	18.57	**19.86**	19.15
Zero-shot (avg.)	—	—	22.25	26.28	24.60

†Soft pivoting (Cheng et al., 2017). ‡Distillation (Chen et al., 2017).

Table 3: Zero-shot results on Europarl. Note that *Soft* and *Distill* are not multilingual systems.

	Previous work		Our baselines		
	SOTA†	CPG‡	Basic	Pivot	Agree
Supervised (avg.)	24.10	19.75	24.63	24.63	23.97
Zero-shot (avg.)	20.55	11.69	19.86	19.26	**20.58**

†Table 2 from Dabre et al. (2017). ‡Table 2 from Platanios et al. (2018).

Table 4: Results on the official IWSLT17 multilingual task.

	Basic	Pivot	Agree
Supervised (avg.)	28.72	28.72	**29.17**
Zero-shot (avg.)	12.61	**17.68**	15.23

Table 5: Results on our proposed IWSLT17*.

(b) `Pivot`, which performs pivoting-based inference using a multilingual GNMT model (after standard training); often regarded as gold-standard.
(c) `Agree`, which applies a multilingual GNMT model trained with agreement losses directly to zero-shot directions.

To ensure a fair comparison in terms of model capacity, all the techniques above use the same multilingual GNMT architecture described in the previous section. All other results provided in the tables are as reported in the literature.

Implementation. All our methods were implemented using TensorFlow (Abadi et al., 2016) on top of tensor2tensor library (Vaswani et al., 2018). Our code will be made publicly available.[9]

6.3 Results on UN Corpus and Europarl

UN Corpus. Tables 1 and 2 show results on the UNCorpus datasets. Our approach consistently outperforms `Basic` and `Dual-0`, despite the latter being trained with additional monolingual data (Sestorain et al., 2018). We see that models trained with agreement perform comparably to `Pivot`, outperforming it in some cases, *e.g.*, when the target is Russian, perhaps because it is quite different linguistically from the English pivot.

Furthermore, unlike `Dual-0`, `Agree` maintains high performance in the supervised directions (within 1 BLEU point compared to `Basic`), indicating that our agreement-based approach is effective as a part of a single multilingual system.

Europarl. Table 3 shows the results on the Europarl corpus. On this dataset, our approach consistently outperforms `Basic` by 2-3 BLEU points but lags a bit behind `Pivot` on average (except on `Es → De` where it is better). Cheng et al. (2017)[10] and Chen et al. (2017) have reported zero-resource results on a subset of these directions and our approach outperforms the former but not the latter on these pairs. Note that both Cheng et al. (2017) and Chen et al. (2017) train separate models for each language pair and the approach of Chen et al. (2017) would require training $O(k^2)$ models to encompass all the pairs. In contrast, we use a single multilingual architecture which has more limited model capacity (although in theory, our approach is also compatible with using separate models for each direction).

---

[9] www.cs.cmu.edu/~mshediva/code/

[10] We only show their best zero-resource result in the table since some of their methods require direct parallel data.

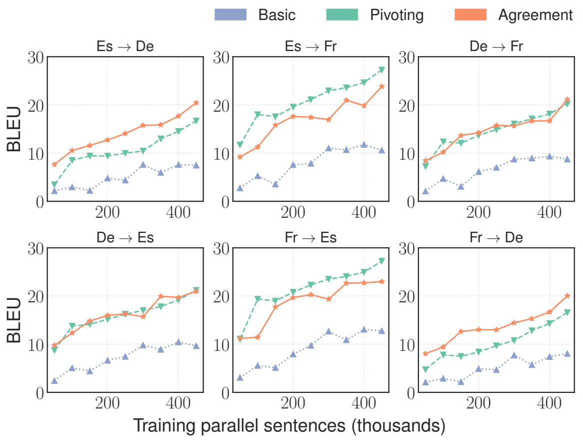


Figure 4: BLEU on the dev set for `Agree` and the baselines trained on smaller subsets of the Europarl corpus.

6.4 Analysis of IWSLT17 zero-shot tasks

Table 4 presents results on the original IWSLT17 task. We note that because of the large amount of data overlap and presence of many supervised translation pairs (16) the vanilla training method (Johnson et al., 2016) achieves very high zero shot performance, even outperforming `Pivot`. While our approach gives small gains over these baselines, we believe the dataset's pecularities make it not reliable for evaluating zero-shot generalization.

On the other hand, on our proposed preprocessed IWSLT17* that eliminates the overlap and reduces the number of supervised directions (8), there is a considerable gap between the supervised and zero-shot performance of `Basic`. `Agree` performs better than `Basic` and is slightly worse than `Pivot`.

6.5 Small data regime

To better understand the dynamics of different methods in the small data regime, we also trained all our methods on subsets of the Europarl for 200K steps and evaluated on the dev set. The training set size varied from 50 to 450K parallel sentences. From Figure 4, `Basic` tends to perform extremely poorly while `Agree` is the most robust (also in terms of variance across zero-shot directions). We see that `Agree` generally upper-bounds `Pivot`, except for the (`Es`, `Fr`) pair, perhaps due to fewer cascading errors along these directions.

7 Conclusion

In this work, we studied zero-shot generalization in the context of multilingual neural machine translation. First, we introduced the concept of zero-shot consistency that implies generalization. Next, we proposed a provably consistent agreement-based learning approach for zero-shot translation. Empirical results on three datasets showed that agreement-based learning results in up to +3 BLEU zero-shot improvement over the Johnson et al. (2016) baseline, compares favorably to other approaches in the literature (Cheng et al., 2017; Sestorain et al., 2018), is competitive with pivoting, and does not lose in performance on supervised directions.

We believe that the theory and methodology behind agreement-based learning could be useful beyond translation, especially in multi-modal settings. For instance, it could be applied to tasks such as cross-lingual natural language inference (Conneau et al., 2018), style-transfer (Shen et al., 2017; Fu et al., 2017; Prabhumoye et al., 2018), or multilingual image or video captioning. Another interesting future direction would be to explore different hand-engineered or learned data representations, which one could use to encourage models to agree on during training (*e.g.*, make translation models agree on latent semantic parses, summaries, or potentially other data representations available at training time).

Acknowledgments

We thank Ian Tenney and Anthony Platanios for many insightful discussions, Emily Pitler for the helpful comments on the early draft of the paper, and anonymous reviewers for careful reading and useful feedback.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey. 2018. The missing ingredient in zero-shot neural machine translation. *OpenReview.net*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Julian Besag. 1975. Statistical analysis of non-lattice data. *The statistician*, pages 179–195.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *ACL 2016 FIRST CONFERENCE ON MACHINE TRANSLATION (WMT16)*, pages 131–198. The Association for Computational Linguistics.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. *arXiv preprint arXiv:1705.00753*.

Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. 2017. Joint training for pivot-based neural machine translation. In *Proceedings of IJCAI*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 728–735.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*.

Raj Dabre, Fabien Cromieres, and Sadao Kurohashi. 2017. Kyoto university mt system description for iwslt 2017. *Proc. of IWSLT, Tokyo, Japan*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multilingual neural machine translation. *arXiv preprint arXiv:1606.04164*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2017. Effective strategies in zero-shot neural machine translation. *arXiv preprint arXiv:1711.07893*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Dark knowledge. *Presented as the keynote in BayLearn*, 2.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Philip Koehn. 2017. Europarl: A parallel corpus for statistical machine translation.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Percy S Liang, Dan Klein, and Michael I Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems*, pages 913–920.

Bruce G Lindsay. 1988. Composite likelihood methods. *Contemporary mathematics*, 80(1):221–239.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. *arXiv preprint arXiv:1804.08198*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Cettolo Mauro, Federico Marcello, Bentivogli Luisa, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017. Overview of the iwslt 2017 evaluation campaign. In *International Workshop on Spoken Language Translation*, pages 2–14.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. *arXiv preprint arXiv:1808.08493*.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Lierni Sestorain, Massimiliano Ciaramita, Christian Buck, and Thomas Hofmann. 2018. Zero-shot dual machine translation. *arXiv preprint arXiv:1805.10338*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Michal Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In *LREC*.

A Appendices

A.1 Complete likelihood

Given a set of conditional models, $\{\mathbb{P}_\theta(\mathbf{x}_j \mid \mathbf{x}_i)\}$, we can write out the full likelihood over equivalent translations, $(\mathbf{x}_1, \dots, \mathbf{x}_k)$, as follows:

$$\mathbb{P}_\theta(\mathbf{x}_1, \dots, \mathbf{x}_k) := \frac{1}{Z} \prod_{i,j \in \mathcal{E}} \mathbb{P}_\theta(\mathbf{x}_j \mid \mathbf{x}_i) \tag{11}$$

where $Z := \sum_{\mathbf{x}_1,\dots,\mathbf{x}_k} \prod_{i,j\in\mathcal{E}} \mathbb{P}_\theta(\mathbf{x}_j \mid \mathbf{x}_i)$ is the normalizing constant and $\mathcal{E}$ denotes *all* edges in the graph (Figure 5). Given only bilingual parallel corpora, $\mathcal{C}_{ij}$ for $i, j \in \mathcal{E}_s$, we can observe only certain pairs of variables. Therefore, the log-likelihood of the data can be written as:

$$\mathcal{L}(\theta) := \sum_{i,j\in\mathcal{E}_s} \sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{C}_{ij}} \log \sum_{\mathbf{z}} \mathbb{P}_\theta(\mathbf{x}_1, \dots, \mathbf{x}_k) \tag{12}$$

Here, the outer sum iterates over available corpora. The middle sum iterates over parallel sentences in a corpus. The most inner sum marginalizes out unobservable sequences, denoted $\mathbf{z} := \{\mathbf{x}_l\}_{l\neq i,j}$, which are sentences equivalent under this model to $\mathbf{x}_i$ and $\mathbf{x}_j$ in languages other than L_i and L_j. Note that due to the inner-most summation, computing the log-likelihood is intractable.

We claim the following.

Claim 4 *Maximizing the full log-likelihood yields zero-shot consistent models (Definition 1).*

Proof. To better understand why this is the case, let us consider example in Figure 5 and compute the log-likelihood of $(\mathbf{x}_1, \mathbf{x}_2)$:

$$\begin{aligned}
&\log \mathbb{P}_\theta(\mathbf{x}_1, \mathbf{x}_2) \\
&\quad = \log \sum\nolimits_{\mathbf{x}_3,\mathbf{x}_4} \mathbb{P}_\theta(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) \\
&\quad \propto \log \mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_2) + \log \mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_1) + \\
&\qquad \log \sum\nolimits_{\mathbf{x}_3,\mathbf{x}_4} \mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_3)\, \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_1) \times \\
&\qquad\qquad \mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_3)\, \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_2) \times \\
&\qquad\qquad \mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_4)\, \mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_1) \times \\
&\qquad\qquad \mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_4)\, \mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_2) \times \\
&\qquad\qquad \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_4)\, \mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_3)
\end{aligned}$$

Note that the terms that encourage agreement on the translation into L_3 are colored in green (similarly, terms that encourage agreement on the translation into L_4 are colored in blue). Since all other terms are probabilities and bounded by 1, we have:

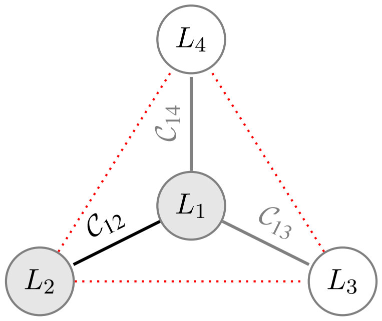


Figure 5: Probabilistic graphical model for a multilingual system with four languages (L_1, L_2, L_3, L_4). Variables can only be observed only in pairs (shaded in the graph).

$$\begin{aligned}
&\log \mathbb{P}_\theta(\mathbf{x}_1, \mathbf{x}_2) + \log Z \\
&\quad \leq \log \mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_2) + \log \mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_1) + \\
&\qquad \log \sum\nolimits_{\mathbf{x}_3,\mathbf{x}_4} \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_1) \mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_2) \times \\
&\qquad\qquad \mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_1) \mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_2) \\
&\quad \equiv \mathcal{L}^{\text{agree}}(\theta)
\end{aligned}$$

In other words, the full log likelihood lower-bounds the agreement objective (up to a constant $\log Z$). Since optimizing for agreement leads to consistency (Theorem 2), and maximizing the full likelihood would necessarily improve the agreement, the claim follows. ■

Remark 5 *Note that the other terms in the full likelihood also have a non-trivial purpose: (a) the terms* $\mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_3)$, $\mathbb{P}_\theta(\mathbf{x}_1 \mid \mathbf{x}_4)$, $\mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_3)$, $\mathbb{P}_\theta(\mathbf{x}_2 \mid \mathbf{x}_4)$*, encourage the model to correctly reconstruct* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *when back-translating from unobserved languages,* L_3 *and* L_4*, and (b) terms* $\mathbb{P}_\theta(\mathbf{x}_3 \mid \mathbf{x}_4)$, $\mathbb{P}_\theta(\mathbf{x}_4 \mid \mathbf{x}_3)$ *enforce consistency between the latent representations. In other words, full likelihood accounts for a combination of agreement, back-translation, and latent consistency.*

A.2 Proof of agreement consistency

The statement of Theorem 2 mentions an assumption on the true distribution of the equivalent translations. The assumption is as follows.

Assumption 6 *Let* $\mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_j, \mathbf{x}_k)$ *be the ground truth conditional distribution that specifies the probability of* $\mathbf{x}_i$ *to be a translation of* $\mathbf{x}_j$ *and* $\mathbf{x}_k$ *into language* L_i*, given that* $(\mathbf{x}_j, \mathbf{x}_k)$ *are correct translations of each other in languages* L_j *and* L_k*, respectively. We assume:*

$$0 \leq \delta \leq \mathbb{E}_{\mathbf{x}_k \mid \mathbf{x}_i, \mathbf{x}_j} \left[\mathbb{P}(\mathbf{x}_i \mid \mathbf{x}_j, \mathbf{x}_k)\right] \leq \xi \leq 1$$

This assumption means that, even though there might be multiple equivalent translations, there must be not too many of them (implied by the δ lower bound) and none of them must be much more preferable than the rest (implied by the ξ upper bound). Given this assumption, we can prove the following simple lemma.

Lemma 7 *Let $L_i \to L_j$ be one of the supervised directions, $\mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j}\left[-\log \mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)\right] \leq \varepsilon$. Then the following holds:*

$$\mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[\frac{\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)}{\mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k\right)}\right] \geq \log\frac{1}{\xi} - \varepsilon\delta$$

Proof. First, using Jensen's inequality, we have:

$$\log \mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[\frac{\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)}{\mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k\right)}\right] \geq$$
$$\mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[\log \mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right) - \log \mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k\right)\right]$$

The bound on the supervised direction implies that

$$\mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[-\log \mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)\right] \geq -\varepsilon\delta$$

To bound the second term, we use Assumption 6:

$$\mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[-\log \mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k\right)\right] \geq \log\frac{1}{\xi}$$

Putting these together yields the bound. ■

Now, using Lemma 7, we can prove Theorem 2.

Proof. By assumption, the agreement-based loss is bounded by ε. Therefore, expected cross-entropy on all supervised terms, $L_1 \leftrightarrow L_2$, is bounded by ε. Moreover, the agreement term (which is part of the objective) is also bounded:

$$-\mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j}\left[\sum_{\mathbf{x}_k} \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right) \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)\right] \leq \varepsilon$$

Expanding this expectation, we have:

$$\sum_{\mathbf{x}_i,\mathbf{x}_j} \mathbb{P}\left(\mathbf{x}_i,\mathbf{x}_j\right) \sum_{\mathbf{x}_k} \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right) \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)]$$
$$= \sum_{\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k} \mathbb{P}\left(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k\right) \times$$
$$\frac{\mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right)}{\mathbb{P}\left(\mathbf{x}_k \mid \mathbf{x}_i,\mathbf{x}_j\right)} \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)$$
$$= \sum_{\mathbf{x}_i,\mathbf{x}_k} \mathbb{E}_{\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k}\left[\frac{\mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right)}{\mathbb{P}\left(\mathbf{x}_k \mid \mathbf{x}_i,\mathbf{x}_j\right)}\right] \times$$
$$\mathbb{P}\left(\mathbf{x}_i,\mathbf{x}_k\right) \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)$$

Combining that with Lemma 7, we have:

$$\mathbb{E}_{\mathbf{x}_i,\mathbf{x}_k}\left[-\log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)\right] \leq \frac{\varepsilon}{\log\frac{1}{\xi} - \delta\varepsilon} \equiv \kappa(\varepsilon)$$

Since by Assumption 6, δ and ξ are some constants, $\kappa(\varepsilon) \to 0$ as $\varepsilon \to 0$. ■

A.3 Consistency of distillation and pivoting

As we mentioned in the main text of the paper, distillation (Chen et al., 2017) and pivoting yield zero-shot consistent models. Let us understand why this is the case.

In our notation, given $L_1 \to L_2$ and $L_2 \to L_3$ as supervised directions, distillation optimizes a KL-divergence between $\mathbb{P}_\theta\left(\mathbf{x}_3 \mid \mathbf{x}_2\right)$ and $\mathbb{P}_\theta\left(\mathbf{x}_3 \mid \mathbf{x}_1\right)$, where the latter is a zero-shot model and the former is supervised. Noting that KL-divergence lower-bounds cross-entropy, it is a loser bound on the agreeement loss. Hence, by ensuring that KL is low, we also ensure that the models agree, which implies consistency (a more formal proof would exactly follow the same steps as the proof of Theorem 2).

To prove consistency of pivoting, we need an additional assumption on the quality of the source-pivot model.

Assumption 8 *Let $\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)$ be the source-pivot model. We assume the following bound holds for each pair of equivalent translations, $\left(\mathbf{x}_j,\mathbf{x}_k\right)$:*

$$\mathbb{E}_{\mathbf{x}_i \mid \mathbf{x}_j,\mathbf{x}_k}\left[\frac{\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right)}{\mathbb{P}\left(\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k\right)}\right] \leq C$$

where $C > 0$ is some constant.

Theorem 9 (Pivoting consistency) *Given the conditions of Theorem 2 and Assumption 8, pivoting is zero-shot consistent.*

Proof. We can bound the expected error on pivoting as follows (using Jensen's inequality and the conditions from our assumptions):

$$\mathbb{E}_{\mathbf{x}_i,\mathbf{x}_k}\left[-\log \sum_{\mathbf{x}_j} \mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right) \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right)\right]$$
$$\leq \mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k}\left[-\mathbb{P}_\theta\left(\mathbf{x}_j \mid \mathbf{x}_i\right) \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right)\right]$$
$$\leq \sum_{\mathbf{x}_i,\mathbf{x}_k} \mathbb{E}_{\mathbf{x}_j \mid \mathbf{x}_i,\mathbf{x}_k}\left[\frac{\mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_j\right)}{\mathbb{P}\left(\mathbf{x}_k \mid \mathbf{x}_i,\mathbf{x}_j\right)}\right] \times$$
$$\mathbb{P}\left(\mathbf{x}_i,\mathbf{x}_k\right) \log \mathbb{P}_\theta\left(\mathbf{x}_k \mid \mathbf{x}_i\right)$$
$$\leq C\varepsilon$$

■

A.4 Details on the models and training

Architecture. All our NMT models used the GNMT (Wu et al., 2016) architecture with Luong attention (Luong et al., 2015b), 2 bidirectional encoder, and 4-layer decoder with residual connections. All hidden layers (including embeddings) had 512 units. Additionally, we used separate embeddings on the encoder and decoder sides as well as tied weights of the softmax that produced logits with the decoder-side (*i.e.*, target) embeddings. Standard dropout of 0.2 was used on all hidden layers. Most of the other hyperparameters we set to default in the T2T (Vaswani et al., 2018) library for the text2text type of problems.

Training and hyperparameters. We scaled agreement terms in the loss by $\gamma = 0.01$. The training was done using Adafactor (Shazeer and Stern, 2018) optimizer with 10,000 burn-in steps at 0.01 learning rate and further standard square root decay (with the default settings for the decay from the T2T library). Additionally, implemented agreement loss as a subgraph as a loss was not computed if γ was set to 0. This allowed us to start training multilingual NMT models in the burn-in mode using the composite likelihood objective and then switch on agreement starting some point during optimization (typically, after the first 100K iterations; we also experimented with 0, 50K, 200K, but did not notice any difference in terms of final performance). Since the agreement subgraph was not computed during the initial training phase, it tended to accelerate training of agreement models.

A.5 Details on the datasets

Statistics of the IWSLT17 and IWSLT17* datasets are summarized in Table 6. UNCorpus and and Europarl datasets were exactly as described by Sestorain et al. (2018) and Chen et al. (2017); Cheng et al. (2017), respectively.

Corpus	**Directions**	**Train**	**Dev** (dev2010)	**Test** (tst2010)
IWSLT17	`De` → `En`	206k	888	1568
	`De` → `It`	205k	923	1567
	`De` → `Nl`	0	1001	1567
	`De` → `Ro`	201k	912	1677
	`En` → `De`	206k	888	1568
	`En` → `It`	231K	929	1566
	`En` → `Nl`	237k	1003	1777
	`En` → `Ro`	220k	914	1678
	`It` → `De`	205k	923	1567
	`It` → `En`	231k	929	1566
	`It` → `Nl`	205k	1001	1669
	`It` → `Ro`	0	914	1643
	`Nl` → `De`	0	1001	1779
	`Nl` → `En`	237k	1003	1777
	`Nl` → `It`	233k	1001	1669
	`Nl` → `Ro`	206k	913	1680
	`Ro` → `De`	201k	912	1677
	`Ro` → `En`	220k	914	1678
	`Ro` → `It`	0	914	1643
	`Ro` → `Nl`	206k	913	1680
IWSLT17*	`De` → `En`	124k	888	1568
	`De` → `It`	0	923	1567
	`De` → `Nl`	0	1001	1567
	`De` → `Ro`	0	912	1677
	`En` → `De`	124k	888	1568
	`En` → `It`	139k	929	1566
	`En` → `Nl`	155k	1003	1777
	`En` → `Ro`	128k	914	1678
	`It` → `De`	0	923	1567
	`It` → `En`	139k	929	1566
	`It` → `Nl`	0	1001	1669
	`It` → `Ro`	0	914	1643
	`Nl` → `De`	0	1001	1779
	`Nl` → `En`	155k	1003	1777
	`Nl` → `It`	0	1001	1669
	`Nl` → `Ro`	0	913	1680
	`Ro` → `De`	0	912	1677
	`Ro` → `En`	128k	914	1678
	`Ro` → `It`	0	914	1643
	`Ro` → `Nl`	0	913	1680

Table 6: Data statistics for IWSLT17 and IWSLT17*. Note that training data in IWSLT17* was restricted to only `En` ↔ {`De`, `It`, `Nl`, `Ro`} directions and cleaned from complete pivots through `En`, which also reduced the number of parallel sentences in each supervised direction.

Modeling Recurrence for Transformer

Jie Hao
Florida State University
haoj8711@gmail.com

Xing Wang
Tencent AI Lab
brightxwang@tencent.com

Baosong Yang
University of Macau
nlp2ct.baosong@gmail.com

Longyue Wang
Tencent AI Lab
vinnylywang@tencent.com

Jinfeng Zhang
Florida State University
jinfeng@stat.fsu.edu

Zhaopeng Tu*
Tencent AI Lab
zptu@tencent.com

Abstract

Recently, the Transformer model (Vaswani et al., 2017) that is based solely on attention mechanisms, has advanced the state-of-the-art on various machine translation tasks. However, recent studies reveal that the lack of recurrence hinders its further improvement of translation capacity (Chen et al., 2018; Dehghani et al., 2019). In response to this problem, we propose to directly model recurrence for Transformer with an additional recurrence encoder. In addition to the standard recurrent neural network, we introduce a novel *attentive recurrent network* to leverage the strengths of both attention and recurrent networks. Experimental results on the widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks demonstrate the effectiveness of the proposed approach. Our studies also reveal that the proposed model benefits from a *short-cut* that bridges the source and target sequences with a single recurrent layer, which outperforms its deep counterpart.

1 Introduction

Recently, Transformer (Vaswani et al., 2017) – a new network architecture based solely on attention mechanisms, has advanced the state-of-the-art on various translation tasks across language pairs. Compared with the conventional recurrent neural network (RNN) (Schuster and Paliwal, 1997) based model that leverages recurrence as the basic building module (Sutskever et al., 2014; Bahdanau et al., 2015; Chen et al., 2018), Transformer replaces RNN with self-attention network (SAN) to model the dependencies among input elements. One appealing strength of SAN is that it breaks down the sequential assumption to obtain the ability of highly parallel computation: input elements interact with each other simultaneously without regard to their distance.

However, prior studies empirically show that the lack of recurrence modeling hinders Transformer from further improvement of translation quality (Dehghani et al., 2019). Modeling recurrence is crucial for capturing several essential properties of input sequence, such as structural representations (Tran et al., 2016) and positional encoding (Shaw et al., 2018), which are exactly the weaknesses of SAN (Tran et al., 2018). Recently, Chen et al. (2018) show that the representations learned by SAN-based and RNN-based encoders are complementary to each other, and merging them can improve translation performance for RNN-based NMT models.

Starting from these findings, we propose to directly model recurrence for Transformer with an additional recurrence encoder. The recurrence encoder recurrently reads word embeddings of input sequence and outputs a sequence of hidden states, which serves as an additional information source to the Transformer decoder. In addition to the standard RNN, we propose to implement recurrence modeling with a novel *attentive recurrent network* (ARN), which combines advantages of both SAN and RNN. Instead of recurring over the individual symbols of sequences like RNN, the ARN recurrently revises its representations over a set of feature vectors, which are extracted by an attention model from the input sequence. Accordingly, ARN combines the strong global modeling capacity of SAN with the recurrent bias of RNN.

We evaluate the proposed approach on widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks. Experimental results show that the additional recurrence encoder, implemented with either RNN or ARN,

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Jie Hao and Baosong Yang were interning at Tencent AI Lab.

Proceedings of NAACL-HLT 2019, pages 1198–1207
Minneapolis, Minnesota, June 2 - June 7, 2019.

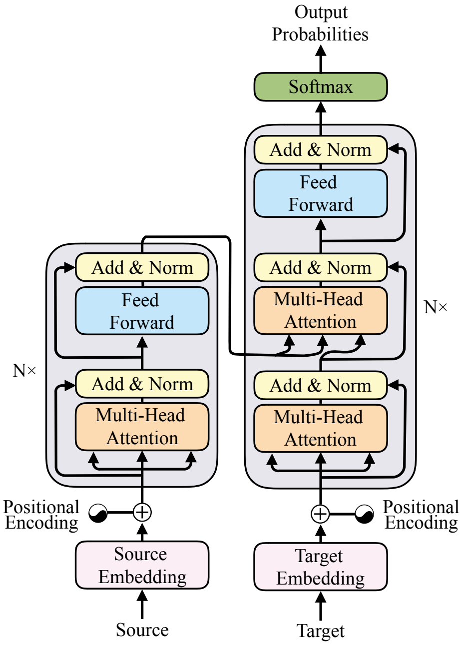


Figure 1: The architecture of Transformer.

consistently improves translation performance, demonstrating the necessity of modeling recurrence for Transformer. Specifically, the ARN implementation outperforms its RNN counterpart, which confirms the strength of ARN.

Further analyses reveal that our approach benefits from a *short-cut* that bridges the source and target sequences with shorter path. Among all the model variants, the implementation with shortest path performs best, in which the recurrence encoder is single layer and its output is only fed to the top decoder layer. It consistently outperforms its multiple deep counterparts, such as multiple-layer recurrence encoder and feeding the output of recurrence encoder to all the decoder layers. In addition, our approach indeed generates more informative encoder representations, especially representative on syntactic structure features, through conducting linguistic analyses on probing tasks (Conneau et al., 2018).

2 Background

Figure 1 shows the model architecture of Transformer. The encoder is composed of a stack of N identical layers, each of which has two sub-layers. The first sub-layer is a self-attention network, and the second one is a position-wise fully connected feed-forward network. A residual connection (He et al., 2016) is employed around each of two sub-layers, followed by layer normalization (Ba et al., 2016). Formally, the output of the first sub-layer $\mathbf{C}_e^n$ and the second sub-layer $\mathbf{H}_e^n$ are sequentially calculated as:

$$\mathbf{C}_e^n = \textsc{Ln}\big(\textsc{Self-Att}(\mathbf{H}_e^{n-1}) + \mathbf{H}_e^{n-1}\big), \quad (1)$$

$$\mathbf{H}_e^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{C}_e^n) + \mathbf{C}_e^n\big), \quad (2)$$

where $\textsc{Self-Att}(\cdot)$, $\textsc{Ln}(\cdot)$, and $\textsc{Ffn}(\cdot)$ are respectively self-attention mechanism, layer normalization, and feed-forward network with ReLU activation in between.

In transformer, $\textsc{Self-Att}(\cdot)$ computes attention over the input $\mathbf{H}_e^{n-1}$ as follows:

$$\textsc{Self-Att}(\mathbf{H}_e^{n-1}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \quad (3)$$

where $\{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}$ are query, key and value vectors that are transformed from the input representations $\mathbf{H}_e^{n-1}$. $\sqrt{d_k}$ is the scaling factor where the d_k is the dimension size of the query and key vectors.

The decoder is also composed of a stack of N identical layers. In addition to two sub-layers in each decoder layer, the decoder inserts a third sub-layer $\mathbf{D}_d^n$ to perform attention over the output of the encoder $\mathbf{H}_e^N$:

$$\mathbf{C}_d^n = \textsc{Ln}\big(\textsc{Self-Att}(\mathbf{H}_d^{n-1}) + \mathbf{H}_d^{n-1}\big), \quad (4)$$

$$\mathbf{D}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_e^N) + \mathbf{C}_d^n\big), \quad (5)$$

$$\mathbf{H}_d^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{D}_d^n) + \mathbf{D}_d^n\big), \quad (6)$$

where $\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_e^N)$ denotes attending the top encoder layer $\mathbf{H}_e^N$ with $\mathbf{C}_d^n$ as query. The top layer of the decoder $\mathbf{H}_d^N$ is used to generate the final output sequence.

3 Approach

In this section, we first describe the architecture of the introduced recurrence encoder and elaborate two types of neural network that are used as recurrence encoder in this work. Then we introduce the integration of recurrence encoder into the Transformer. Specifically, two strategies are presented to fuse the representations produced by the recurrence encoder and the conventional encoder. Finally we present the *short-cut* connection between the recurrence encoder and the decoder that we found very effective to use the learned representation to improve the translation performance under the proposed architecture.

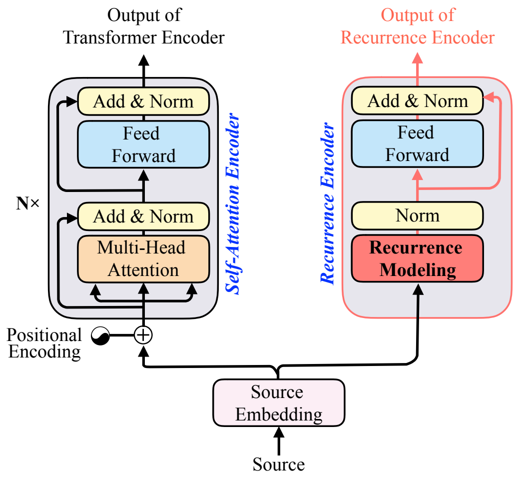


Figure 2: The architecture of Transformer augmented with an additional *recurrence encoder*, the output of which is directly fed to the top decoder layer.

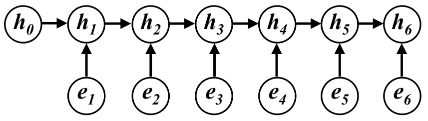
3.1 Recurrence Modeling

Figure 2 shows the architecture of the introduced *recurrence encoder* which reads word embeddings of source words and outputs a sequence of hidden states that embeds recurrent information. Similar to the Transformer encoder, it has a stack of N identical layers, each of which has two sub-layers. The first one is a recurrence modeling network and the second is a fully connected feed-forward network:

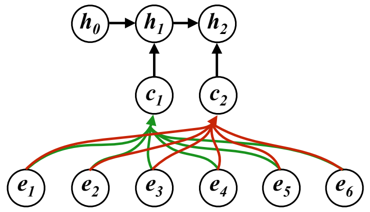
$$
\begin{aligned}
\mathbf{C}_r^n &= \textsc{Ln}(\textsc{Rec}(\mathbf{H}_r^{n-1}) + \mathbf{H}_r^{n-1}), &(7)\\
\mathbf{H}_r^n &= \textsc{Ln}(\textsc{Ffn}(\mathbf{C}_r^n) + \mathbf{C}_r^n), &(8)
\end{aligned}
$$

where $\textsc{Rec}(\cdot)$ is the function of recurrence modeling. Note that at the bottom layer of the recurrence encoder (N=1), we do not employ a residual connection on the recurrence sub-layer (i.e. Equation 7), which releases the constraint that $\mathbf{C}_r^1$ should share the same length with input embeddings sequence $\mathbf{E}_{in}$[1]. This offers a more flexible choice of the recurrence functions.

There are many possible ways to implement the general idea of recurrence modeling $\textsc{Rec}(\cdot)$. The aim of this paper is not to explore this whole space but simply to show that some fairly straightforward implementations work well. In this work, we investigate two representative implementations, namely RNN and its variation *attentive current network* that combines advantages of both RNN and attention models, as shown in Figure 3.



(a) Recurrent Neural Network



(b) Attentive Recurrent Network

Figure 3: Two implementations of recurrence modeling: (a) standard RNN, and (b) the proposed ARN.

Recurrent Neural Network (RNN) An intuitive choice of recurrence modeling is RNN, which is a standard network to model sequence orders. In this work, we use a bidirectional RNN (BiRNN), which is widely applied in RNN-based NMT models (Bahdanau et al., 2015; Chen et al., 2018). Each hidden state in the output representations $\mathbf{H}_{\text{RNN}}^n = \{\mathbf{h}_1^n, \ldots, \mathbf{h}_J^n\}$ is calculated as

$$
\begin{aligned}
\mathbf{h}_j^n &= [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j], &(9)\\
\overrightarrow{\mathbf{h}}_j &= \overrightarrow{f}(\overrightarrow{\mathbf{h}}_{j-1}, \mathbf{h}_j^{n-1}), &(10)\\
\overleftarrow{\mathbf{h}}_j &= \overleftarrow{f}(\overleftarrow{\mathbf{h}}_{j+1}, \mathbf{h}_j^{n-1}), &(11)
\end{aligned}
$$

where $\overrightarrow{f}(\cdot)$ and $\overleftarrow{f}(\cdot)$ are the activation functions of forward and backward RNN respectively, which can be implemented as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014). $\mathbf{h}_0^n$ is the initial state of RNN, which is the mean of $\mathbf{H}_{\text{RNN}}^{n-1}$. $\mathbf{H}_{\text{RNN}}^0$ represents the word embeddings of the input sequence.

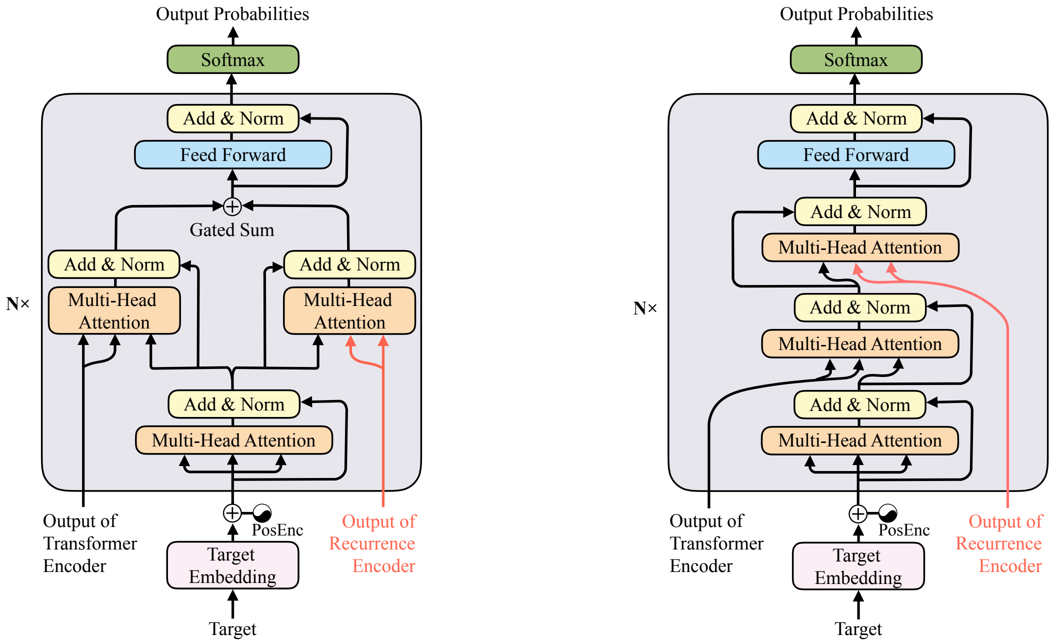
Attentive Recurrent Network (ARN) We can also extend RNN by recurring over a set of feature vectors extracted with an attention model, which allows the model to learn a compact, abstractive feature vectors over the input sequence. Specifically, the ARN performs T recurrent steps on the attentive output of the input representation $\mathbf{H}_r^{n-1}$:

$$
\begin{aligned}
\mathbf{h}_t^n &= f(\mathbf{h}_{t-1}^n, \mathbf{c}_t^n), &(12)\\
\mathbf{c}_t^n &= \textsc{Att}(\mathbf{h}_{t-1}^n, \mathbf{H}_r^{n-1}). &(13)
\end{aligned}
$$

The output representations $\mathbf{H}_{\text{ARN}}^n = \{\mathbf{h}_1^n, \ldots, \mathbf{h}_T^n\}$ are fed to the subsequent modules. Analogous to Equations 9-11, ARN can be extended to the bidirectional variant, i.e. BiARN,

---

[1] The input of the lowest layer in the recurrence encoder is the word embeddings of input sequence $\mathbf{E}_{in}$.



(a) Gated Sum (b) Stack

Figure 4: Different strategies to integrate the output of the additional *recurrence encoder* into the decoder.

except that the input is the attentive context vector $\mathbf{c}_t^n$ rather than the individual representation vector of the input sequence.

Note that, the number of recurrence step T is allowed to be unequal to the length of input sequence J. In contrast to RNN which recurs over the individual symbols of the input sequences, ARN recurrently revises its representations of all symbols in the sequence with an attention model.

3.2 Integrating into Transformer

Since the output of recurrence encoder unnecessarily shares the same length with that of Transformer encoder (e.g. when ARN is used as recurrence function), combination strategy on the encoder side, such as concatenating the outputs of both encoders (Chen et al., 2018), is not an universal solution in this scenario. Accordingly, we feed the information of the additional recurrence encoder into the decoder of Transformer. Specifically, we serve an additional attention layer $\mathbf{R}_d^n$ as the fourth sub-layer in each decoder block to perform attention over the output of the recurrence encoder $\mathbf{H}_r^N$. As shown in Figure 4, we present two strategies to integrate $\mathbf{R}_d^n$, namely *gated sum* and *stack*, which differ at how $\mathbf{R}_d^n$ interacts with the output of attention over the Transformer encoder, i.e., $\mathbf{D}_d^n$ in Equation 5.

Gated Sum The first strategy combines the outputs of the two attention sub-layers in a gating fusion (Figure 4(a)), in which the outputs of both encoders are attended simultaneously:

$$\begin{aligned}
\mathbf{R}_d^n &= \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_r^N) + \mathbf{C}_d^n\big), && (14)\\
\widehat{\mathbf{D}}_d^n &= \lambda_n \mathbf{D}_d^n + (1-\lambda_n)\mathbf{R}_d^n, && (15)\\
\mathbf{H}_d^n &= \textsc{Ln}\big(\textsc{Ffn}(\widehat{\mathbf{D}}_d^n) + \widehat{\mathbf{D}}_d^n\big), && (16)
\end{aligned}$$

where λ_n is an interpolation weight calculated by a logistic sigmoid function:

$$\lambda_n = sigmoid(\mathbf{D}_d^n, \mathbf{R}_d^n) \quad (17)$$

As seen, the output of self-attention layer $\mathbf{C}_d^n$ serves as a query to attend the outputs of both encoders (Equations 5 and 14), and the outputs of both attention models $\{\mathbf{D}_d^n, \mathbf{R}_d^n\}$ are combined via a gated sum (Equation 15), which is subsequently fed to the feed-forward layer (Equation 16).

Stack We can also arrange the sub-layers in a *stack* (Figure 4(b)), in which the outputs of both encoders are attended sequentially:

$$\begin{aligned}
\mathbf{R}_d^n &= \textsc{Ln}\big(\textsc{Att}(\mathbf{D}_d^n, \mathbf{H}_r^N) + \mathbf{D}_d^n\big), && (18)\\
\mathbf{H}_d^n &= \textsc{Ln}\big(\textsc{Ffn}(\mathbf{R}_d^n) + \mathbf{R}_d^n\big), && (19)
\end{aligned}$$

The decoder first attends the output of Transformer encoder, and the attention output $\mathbf{D}_d^n$ serves as the query to attend the output of recurrence encoder (Equation 18).

3.3 Short-Cut Effect

The introduced recurrence encoder provides an additional computation path ranging from the input sequence to the output sequence. Chung et al. (2017) and Shen et al. (2019) have shown that a shortcut for gradient back-propagation benefits language modeling. Inspired from them, we use a shorter path to transform the learned recurrence. We call this the "*short-cut effect*".

Among all the model variants, we implement shortest path as: the recurrence encoder is single layer and its output is only fed to the top decoder layer while the first $N-1$ decoder layers perform the same as the standard Transformer (e.g. Equations 4-6). Accordingly, the computation path is $\mathbf{E}_{in} \rightarrow \mathbf{H}_r \rightarrow \mathbf{R}_d^N \rightarrow \mathbf{H}_d^N$, then the decoder uses $\mathbf{H}_d^N$ to make a target word prediction. It is much simpler than that of the conventional Transformer, which transfers information learned from input sequences across multiple stacking encoder and decoder layers. We expect it outperforms its multiple deep counterparts, such as multiple-layer recurrence encoder and feeding the output of recurrence encoder to all the decoder layers.

4 Related Work

Improving Transformer Encoder From the perspective of representation learning, there has been an increasing amount of work on improving the representation power of SAN encoder. Bawden et al. (2018) and Voita et al. (2018) exploit external context for SAN encoder, while Yang et al. (2019) leverage the intermediate representations to contextualize the transformations in SAN. A number of recent efforts have explored ways to improve multi-head SAN by encouraging individual attention heads to extract distinct information (Strubell et al., 2018; Li et al., 2018). Concerning multi-layer SAN encoder, Dou et al. (2018, 2019) and Wang et al. (2018) propose to aggregate the multi-layer representations, and Dehghani et al. (2019) recurrently refine these representations. Our approach is complementary to theirs, since they focus on improving the representation power of SAN encoder, while we aim to complement SAN encoder with an additional recurrence encoder.

Along the direction of modeling recurrence for SAN, Vaswani et al. (2017) and Shaw et al. (2018) inject absolute position encoding and relative positional encoding to consider the position information respectively. Shen et al. (2018) introduce a directional self-attention network (DiSAN), which allows each token to attend to previous (or following) tokens only. Both studies verify the necessity of modeling recurrence for SAN. We re-implemented these approaches on top of Transformer, and experimental results show that our approach outperforms them by explicitly augmenting Transformer with an additional recurrence encoder. It should be emphasized that our approach is complementary to theirs, and combining them together is expected to further improve performance, which we leave for future work.

Closely related to our work, Chen et al. (2018) propose to combine SAN encoder with an additional RNN encoder. The main differences between our work and theirs are: 1) we enhance the state-of-the-art Transformer with recurrence information, while Chen et al. (2018) augment RNN-based models with SAN encoder. To this end, we propose a novel attentive recurrent network to implement the additional recurrence encoder in Transformer. We re-implemented the approach proposed by Chen et al. (2018) on top of Transformer. Experimental results indicate the superiority of our approach, which confirms our claim. In addition, we elaborately design the integration strategy to effectively feed the recurrence information to the decoder, and empirically show that the proposed model benefits from the *short-cut* effect.

Comparison to Reviewer Network Attentive recurrent network are inspired by the reviewer network, which is proposed by Yang et al. (2016) for the image caption generation task. There are two key differences which reflect how we have generalized from the original model. First, we perform attention steps over the source embeddings instead of the encoder representations. The main reason is that the Transformer encoder is implemented as multiple layers, and higher layers generally encode global information, as indicated by Peters et al. (2018). Second, we feed the feature vectors together with the original encoder representations to the decoder. In image caption generation, the source side (i.e. image) contains much more information than the target side (i.e. caption) (Tu et al., 2017). Therefore, they aim at learning a compact and abstractive representation from the source information, which serves as the only input to the decoder. In this work, we focus on leveraging the attention model to better learn the recurrence, which we expect to complement the Transformer model. In our preliminary experiments, attending over the encoder representations does not improve performance, while feeding the feature vectors only to the decoder seriously harms performance.

5 Experiment

5.1 Setup

We conducted experiments on the widely-used WMT14 English-to-German (4.6M sentence pairs, En⇒De) and WMT17 Chinese-to-English (20.6M sentence pairs, Zh⇒En) translation tasks. All the data had been tokenized and segmented into subword symbols using byte-pair encoding (Sennrich et al., 2016) with 32K merge operations[2]. We used case-sensitive NIST BLEU score (Papineni et al., 2002) as the evaluation metric, and *bootstrap resampling* (Koehn et al., 2003) for statistical significance test.

We implemented the proposed approaches on top of the Transformer model (Vaswani et al., 2017). Both in our model and related model of Subsection 5.3, the RNN is implemented with GRU (Cho et al., 2014) for fair comparison. We followed the configurations in Vaswani et al. (2017), and reproduced their reported results on the En⇒De task.

We initialized parameters of the proposed models by the pre-trained baseline model. We have tested both *Base* and *Big* models, which differ at hidden size (512 vs. 1024), filter size (2048 vs. 4096), and number of attention heads (8 vs. 16). In consideration of computation cost, we studied model variations with *Base* model on En⇒De task, and evaluated overall performances with both *Base* and *Big* models on both En⇒De and Zh⇒En translation tasks.

5.2 Impact of Components

In this subsection, we conducted ablation studies to evaluate the different implementations of the proposed model, e.g., recurrence encoder and integration strategy, under the proposed architecture.

Effect of Recurrence Modeling We first investigated the effect of recurrence encoder implementations, as listed in Table 1. We observed that introducing an additional recurrence encoder improves translation performance in all cases. Among all model variations, BiARN outperforms its BiRNN counterpart.

Model	Rec. Encoder	Speed	BLEU
Base	n/a	1.28	27.31
Ours	6-Layer BiRNN	1.10	27.54
	6-Layer BiARN	1.09	27.72
	3-Layer BiARN	1.15	28.10
	1-Layer BiARN	1.24	**28.21**

Table 1: Evaluation of recurrence encoder implementations. The output of recurrence encoder is fed to the top decoder layer in a stack fusion. "Speed" denotes the training speed (steps/second).

Model	Integration	to Dec.	BLEU
Base	n/a	n/a	27.31
Ours	Gated Sum	Top	28.12
	Gated Sum	All	28.02
	Stack	Top	**28.21**
	Stack	All	27.93

Table 2: Evaluation of decoder integration strategies.

Concerning BiARN models, reducing the layers consistently improves performance. Specifically, the 1-Layer BiARN achieves the best performances in both translation quality and training speed. This confirms the claim that the proposed approach benefits from a *short-cut* on gradient back-propagation. Accordingly, we adopted 1-Layer BiARN as the default setting in the following experiments.

Effect of Integration Strategies We then tested the effect of different integration strategies, as showed in Table 2. We have two observations. First, feeding only to the top decoder layer consistently outperforms feeding to all decoder layers with different integration strategies. This empirically reconfirms the short-cut effect. Second, the stack strategy marginally outperforms its gated sum counterpart. Therefore, in the following experiments, we adopted the "Stack + Top" model in Table 2 as defaulting setting.

5.3 Results

Performances across Languages Finally, we evaluated the proposed approach on the widely used WMT17 Zh⇒En and WMT14 En⇒De data, as listed in Table 3.

To make the evaluation convincing, we reviewed the prior reported systems, and built strong

---

[2] https://github.com/rsennrich/subword-nmt

System	Architecture	Zh⇒En		En⇒De	
		# Para.	BLEU	# Para.	BLEU
Existing NMT systems					
(Vaswani et al., 2017)	TRANSFORMER-BASE	n/a	n/a	65M	27.3
	TRANSFORMER-BIG	n/a	n/a	213M	28.4
(Hassan et al., 2018)	TRANSFORMER-BIG	n/a	24.2	n/a	n/a
(Chen et al., 2018)	RNMT + SAN Encoder	n/a	n/a	n/a	28.84
Our NMT systems					
this work	TRANSFORMER-BASE	107.9M	24.13	88.0M	27.31
	+ 1-Layer BIARN	+9.4M	$24.70^{\Uparrow}$	+9.4M	$28.21^{\Uparrow}$
	TRANSFORMER-BIG	303.9M	24.56	264.1M	28.58
	+ 1-Layer BIARN	+69.4M	$25.10^{\Uparrow}$	+69.4M	$28.98^{\uparrow}$

Table 3: Comparing with the existing NMT systems on WMT17 Zh⇒En and WMT14 En⇒De test sets. "↑ / ⇑": significant over the conventional self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling.

Model	BLEU
TRANSFORMER-BASE	27.31
+ RELPOS	27.64
+ DISAN	27.58
+ RNN Encoder	27.47
+ BIARN Encoder (OURS)	**28.21**

Table 4: Comparison with re-implemented related work: "RELPOS": relative position encoding (Shaw et al., 2018), "DISAN": directional SAN (Shen et al., 2018), "RNN Encoder": combining SAN and RNN encoders with multi-column strategy (Chen et al., 2018).

baselines which outperform the reported results on the same data. As seen in Table 3, modeling recurrence consistently improves translation performance across model variations (BASE and BIG models) and language pairs (Zh⇒En and En⇒De), demonstrating the effectiveness and universality of our approach.

Comparison with Previous Work In order to directly compare our approach with the previous work on modeling recurrence, we re-implemented their approaches on top of the TRANSFORMER-BASE in WMT14 En⇒De translation task. For relative position encoding, we used unique edge representations per layer and head with clipping distance $k = 16$. For the DiSAN strategy, we applied a mask to the TRANSFORMER encoder, which constrains the SAN to focus on forward or backward elements. For the multi-column encoder, we re-implemented the additional encoder with six RNN layers.

Table 4 lists the results. As seen, all the recurrence enhanced approaches achieve improvements over the baseline model TRANSFORMER-BASE, which demonstrates the necessity of modeling recurrence for TRANSFORMER. Among these approaches, our approach (*i.e.,* 1-Layer BIARN Encoder) achieves the best performance.

5.4 Analysis

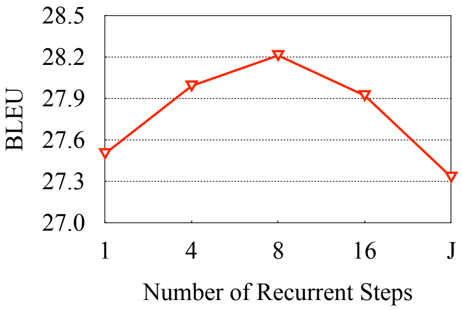


Figure 5: Effect of recurrent steps. The recurrence encoder is implemented as a single-layer BIARN. J denotes the length of the input sequence.

Effect of Recurrent Steps To verify the recurrence effect on the proposed model, we conducted experiments with different recurrent steps on single-layer BIARN model. As shown in Figure 5, the BLEU score typically goes up with the increase of the recurrent steps, while the trend does not hold when $T > 8$. This finding is consistent with Yang et al. (2016), which indicates that conducting too many recurrent steps fails to generate a compact representation. This is exactly one of the ARN's strengths.

Model	Surface		Syntactic			Semantic				
	SeLen	WC	TrDep	ToCo	BShif	Tense	SubN	ObjN	SoMo	CoIn
BASE	**92.20**	63.00	44.74	79.02	71.24	**89.24**	84.69	84.53	52.13	62.47
6-Layer BIRNN	89.90	**77.46**	44.47	79.55	71.53	89.17	**85.99**	84.96	51.75	61.92
6-Layer BIARN	89.78	72.02	44.45	79.21	71.31	88.38	85.64	**85.00**	53.27	62.38
3-Layer BIARN	89.80	72.61	44.28	79.43	71.84	88.93	85.79	84.99	53.30	62.42
1-Layer BIARN	90.91	73.68	**45.15**	**79.62**	**72.21**	89.00	85.54	84.54	**53.44**	**62.71**

Table 5: Classification accuracies on 10 probing tasks of evaluating linguistics embedded in the encoder outputs.

Linguistic Analyses In this section, we conducted 10 probing tasks[3] to study what linguistic properties are captured by the encoders (Conneau et al., 2018). A probing task is a classification problem that focuses on simple linguistic properties of sentences. 'SeLen' predicts the length of sentences in terms of number of words. 'WC' tests whether it is possible to recover information about the original words given its sentence embedding. 'TrDep' checks whether an encoder infers the hierarchical structure of sentences. In 'ToCo' task, sentences should be classified in terms of the sequence of top constituents immediately below the sentence node. 'BShif' tests whether two consecutive tokens within the sentence have been inverted. 'Tense' asks for the tense of the main-clause verb. 'SubN' focuses on the number of the main clause's subject. 'ObjN' tests for the number of the direct object of the main clause. In 'SoMo', some sentences are modified by replacing a random noun or verb with another one and the classifier should tell whether a sentence has been modified. 'CoIn' contains sentences made of two coordinate clauses. Half of sentences are inverted the order of the clauses and the task is to tell whether a sentence is intact or modified.

We used the pre-trained encoders of model variations in Table 1 to generate the sentence representations of input, which are used to carry out probing tasks. For the TRANSFORMER-BASE model, the mean of the encoder top layer representations is used as the sentence representation. For the proposed models, which have two encoders, two sentence representations are generated from the same way in base model. To make full use of the learned representations, we combined these two sentence representations via a gate as the final sentence representation to conduct the experiments.

Table 5 lists the results. Clearly, the proposed models significantly improve the classification accuracies, although there is still considerable difference among different variants. More specifically,

- Concerning surface properties, among the ARN variants, multi-layer ARN inversely decreases the accuracies, while 1-layer ARN consistently improves the accuracies. Considering the related results presented in Table 1 (Row 3-5), we believe that ARN benefits from the shallow structure.
- ARN tends to capture deeper linguistic properties, both syntactic and semantic. Especially, among these probing tasks, 'TrDep' and 'Toco' tasks are related to syntactic structure modeling. As expected, TRANSFORMER augmented with an additional encoders outperforms the baseline model, which demonstrates that the proposed models successfully model the syntactic structure.

6 Conclusion

In this work, we propose to directly model recurrence for Transformer with an additional recurrence encoder. We implement the recurrence encoder with a novel attentive recurrent network as well as RNN. The recurrence encoder is used to generate recurrence representations for the input sequence. To effectively feed the recurrence representations to the decoder to guide the output sequence generation, we study two strategies to integrate the recurrence encoder into the Transformer. To evaluate the effectiveness of the proposed model, we conduct experiments on large-scale WMT14 EN⇒DE and WMT17 ZH⇒EN datasets. Experimental results on two language pairs show that the proposed model achieves significant improvements over the baseline TRANSFORMER. Linguistic analyses on probing tasks

---

[3] https://github.com/facebookresearch/SentEval/tree/master/data/probing

further show that our model indeed generates more informative representations, especially representative on syntactic structure features.

Future work includes validating the proposed model in other tasks, such as reading comprehension, language inference, and sentence classification. Another promising direction is to directly augment Transformer encoder on recurrence modeling without the additional encoder.

Acknowledgments

J.Z. was supported by the National Institute of General Medical Sciences of the National Institute of Health under award number R01GM126558. We thank the anonymous reviewers for their insightful comments.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating Discourse Phenomena in Neural Machine Translation. In *NAACL*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

JunYoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. In *ICLR*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *ICLR*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation. In *AAAI*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *ACL*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: directional self-attention network for RNN/CNN-free language understanding. In *AAAI*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *EMNLP*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *TACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *ACL*.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018. Multi-layer representation fusion for neural machine translation. In *COLING*.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019. Context-aware self-attention networks. In *AAAI*.

Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W Cohen. 2016. Review networks for caption generation. In *NIPS*.

Rethinking Action Spaces for Reinforcement Learning in End-to-end Dialog Agents with Latent Variable Models

Tiancheng Zhao[1], **Kaige Xie**[2] **and Maxine Eskenazi**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Shanghai Jiao Tong University
`{tianchez, max+}@cs.cmu.edu, lightyear0117@sjtu.edu.cn`

Abstract

Defining action spaces for conversational agents and optimizing their decision-making process with reinforcement learning is an enduring challenge. Common practice has been to use handcrafted dialog acts, or the output vocabulary, e.g. in neural encoder decoders, as the action spaces. Both have their own limitations. This paper proposes a novel latent action framework that treats the action spaces of an end-to-end dialog agent as latent variables and develops unsupervised methods in order to induce its own action space from the data. Comprehensive experiments are conducted examining both continuous and discrete action types and two different optimization methods based on stochastic variational inference. Results show that the proposed latent actions achieve superior empirical performance improvement over previous word-level policy gradient methods on both DealOrNoDeal and MultiWoz dialogs. Our detailed analysis also provides insights about various latent variable approaches for policy learning and can serve as a foundation for developing better latent actions in future research. [1]

1 Introduction

Optimizing dialog strategies in multi-turn dialog models is the cornerstone of building dialog systems that more efficiently solve real-world challenges, e.g. providing information (Young, 2006), winning negotiations (Lewis et al., 2017), improving engagement (Li et al., 2016) etc. A classic solution employs reinforcement learning (RL) to learn a dialog policy that models the optimal action distribution conditioned on the dialog state (Williams and Young, 2007). However, since there are infinite human language possibilities, an enduring challenge has been to define what the action space is. For traditional modular systems, the action space is defined by hand-crafted semantic representations such as dialog acts and slot-values (Raux et al., 2005; Chen et al., 2013) and the goal is to obtain a dialog policy that chooses the best hand-crafted action at each dialog turn. But it is limited because it can only handle simple domains whose entire action space can be captured by hand-crafted representations (Walker, 2000; Su et al., 2017). This cripples a system's ability to handle conversations in complex domains.

Conversely, end-to-end (E2E) dialog systems have removed this limit by directly learning a response generation model conditioned on the dialog context using neural networks (Vinyals and Le, 2015; Sordoni et al., 2015). To apply RL to E2E systems, the action space is typically defined as the entire vocabulary; every response output word is considered to be an action selection step (Li et al., 2016), which we denote as the word-level RL. Word-level RL, however, has been shown to have several major limitations in learning dialog strategies. The foremost one is that direct application of word-level RL leads to degenerate behavior: the response decoder deviates from human language and generates utterances that are incomprehensible (Lewis et al., 2017; Das et al., 2017; Kottur et al., 2017). A second issue is that since a multi-turn dialog can easily span hundreds of words, word-level RL suffers from credit assignment over a long horizon, leading to slow and suboptimal convergence (Kaelbling et al., 1996; He et al., 2018).

This paper proposes Latent Action Reinforcement Learning (LaRL), a novel framework that overcomes the limitations of word-level RL for E2E dialog models, marrying the benefits of a traditional modular approach in an unsupervised manner. The key idea is to develop E2E models that can invent their own discourse-level ac-

---

[1] Data and code are available at `https://github.com/snakeztc/NeuralDialog-LaRL`

Proceedings of NAACL-HLT 2019, pages 1208–1218
Minneapolis, Minnesota, June 2 - June 7, 2019.

tions. These actions must be expressive enough to capture response semantics in complex domains (i.e. have the capacity to represent a large number of actions), thus decoupling the discourse-level decision-making process from natural language generation. Then any RL technique can be applied to this induced action space in the place of word-level output. We propose a flexible latent variable dialog framework and investigate several approaches to inducing latent action space from natural conversational data. We further propose (1) a novel training objective that outperforms the typical evidence lower bound used in dialog generation and (2) an attention mechanism for integrating discrete latent variables in the decoder to better model long responses.

We test this on two datasets, DealOrNoDeal (Lewis et al., 2017) and MultiWoz (Budzianowski et al., 2018), to answer two key questions: (1) what are the advantages of LaRL over Word-level RL and (2) what effective methods can induce this latent action space. Results show that LaRL is significantly more effective than word-level RL for learning dialog policies and it does not lead to incomprehensible language generation. Our models achieve 18.2% absolute improvement over the previous state-of-the-art on MultiWoz and discover novel and diverse negotiation strategies on DealOrNoDeal. Besides strong empirical improvement, our model analysis reveals novel insights, e.g. it is crucial to reduce the exposure bias in the latent action space and discrete latent actions are more suitable than continuous ones to serve as action spaces for RL dialog agents.

2 Related Work

Prior RL research in modular dialog management has focused on policy optimization over hand-crafted action spaces in task-oriented domains (Walker, 2000; Young et al., 2007). A dialog manager is formulated as a Partially Observable Markov Decision Process (POMDP) (Young et al., 2013), where the dialog state is estimated via dialog state tracking models from the raw dialog context (Lee, 2013; Henderson et al., 2014; Ren et al., 2018). RL techniques are then used to find the optimal dialog policy (Gasic and Young, 2014; Su et al., 2017; Williams et al., 2017). Recent deep-learning modular dialog models have also explored joint optimization over dialog policy and state tracking to achieve stronger performance (Wen et al., 2016; Zhao and Eskenazi, 2016; Liu and Lane, 2017).

A related line of work is reinforcement learning for E2E dialog systems. Due to the flexibility of encoder-decoder dialog models, prior work has applied reinforcement learning to more complex domains and achieved higher dialog-level rewards, such as open-domain chatting (Li et al., 2016; Serban et al., 2017a), negotiation (Lewis et al., 2017), visual dialogs (Das et al., 2017), grounded dialog (Mordatch and Abbeel, 2017) etc. As discussed in Section 1, these methods consider the output vocabulary at every decoding step to be the action space; they suffer from limitations such as deviation from natural language and sub-optimal convergence.

Finally, research in latent variable dialog models is closely related to our work, which strives to learn meaningful latent variables for E2E dialog systems. Prior work has shown that learning with latent variables leads to benefits like diverse response decoding (Serban et al., 2017b; Zhao et al., 2017; Cao and Clark, 2017), interpretable decision-making (Wen et al., 2017; Zhao et al., 2018) and zero-shot domain transfer (Zhao and Eskenazi, 2018). Also, driven by similar motivations of this work, prior studies have explored to utilize a coarse discrete node, either handcrafted or learned, to decouple the word generation process from dialog policy in E2E systems for better dialog policy (He et al., 2018; Yarats and Lewis, 2017). Our work differs from prior work for two reasons: (1) latent action in previous work is only auxiliary, small-scale and mostly learned in a supervised or semi-supervised setting. This paper focuses on unsupervised learning of latent variables and learns variables that are expressive enough to capture the entire action space by itself. (2) to our best knowledge, our work is the first comprehensive study of the use of latent variables for RL policy optimization in dialog systems.

3 Baseline Approach

E2E response generation can be treated as a conditional language generation task, which uses neural encoder-decoders (Cho et al., 2014) to model the conditional distribution $p(\mathbf{x}|\mathbf{c})$ where $\mathbf{c}$ is the observed dialog context and $\mathbf{x}$ is the system's response to the context. The format of the dialog context is domain dependent. It can vary from tex-

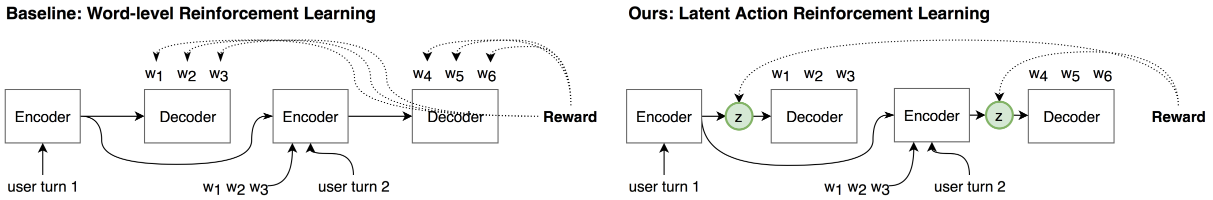


Figure 1: High-level comparison between word-level and latent-action reinforcement learning in a sample multi-turn dialog. The decoder network generates the response given the latent code $\mathbf{z}$. Dashed line denotes places where policy gradients from task rewards are applied to the model.

tual raw dialog history (Vinyals and Le, 2015) to visual and textual context (Das et al., 2017). Training with RL usually has 2 steps: supervised pretraining and policy gradient reinforcement learning (Williams and Zweig, 2016; Dhingra et al., 2017; Li et al., 2016). Specifically, the supervised learning step maximizes the log likelihood on the training dialogs, where θ is the model parameter:

$$\mathcal{L}_{SL}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{c}}[\log p_\theta(\mathbf{x}|\mathbf{c})] \tag{1}$$

Then the following RL step uses policy gradients, e.g. the REINFORCE algorithm (Williams, 1992) to update the model parameters with respect to task-dependent goals. We assume that we have an environment that the dialog agent can interact with and that there is a turn-level reward r_t at every turn t of the dialog. We can then write the expected discounted return under a dialog model θ as $J(\theta) = \mathbb{E}[\sum_0^T \gamma^t r_t]$, where $\gamma \in [0, 1]$ is the discounting factor and T is the length of the dialog. Often a baseline function b is used to reduce the variance of the policy gradient (Greensmith et al., 2004), leading to $R_t = \sum_{k=0}^{T-t} \gamma^k (r_{t+k} - b)$.

Word-level Reinforcement Learning: as shown in Figure 1, the baseline approach treats every output word as an action step and its policy gradient is:

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta[\sum_{t=0}^{T}\sum_{j=0}^{U_t} R_{tj} \nabla_\theta \log p_\theta(w_{tj}|w_{<tj}, \mathbf{c}_t)] \tag{2}$$

where U_t is the number of tokens in the response at turn t and j is the word index in the response. It is evident that Eq 2 has a very large action space, i.e. $|V|$ and a long learning horizon, i.e. TU. Prior work has found that the direct application of Eq 2 leads to divergence of the decoder. The common solution is to alternate with supervised learning with Eq 2 at a certain ratio (Lewis et al., 2017). We denote this ratio as RL:SL=A:B, which means for every A policy gradient updates, we run B supervised learning updates. We use RL:SL=off for the case where only policy gradients are used and no supervised learning is involved.

4 Latent Action Reinforcement Learning

We now describe the proposed LaRL framework. As shown in Figure 1, a latent variable $\mathbf{z}$ is introduced in the response generation process. The conditional distribution is factorized into $p(\mathbf{x}|\mathbf{c}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|\mathbf{c})$ and the generative story is: (1) given a dialog context $\mathbf{c}$ we first sample a latent action $\mathbf{z}$ from $p_{\theta_e}(\mathbf{z}|\mathbf{c})$ and (2) generate the response by sampling $\mathbf{x}$ based on $\mathbf{z}$ via $p_{\theta_d}(\mathbf{x}|\mathbf{z})$, where p_{θ_e} is the dialog encoder network and p_{θ_d} is the response decoder network. Given the above setup, LaRL treats the latent variable $\mathbf{z}$ as its action space instead of outputting words in response $\mathbf{x}$. We can now apply REINFORCE in the latent action space:

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta[\sum_{t=0}^{T} R_t \log p_\theta(\mathbf{z}|\mathbf{c}_t)] \tag{3}$$

Compared to Eq 2, LaRL differs by:

- Shortens the horizon from TU to T.
- Latent action space is designed to be low-dimensional, much smaller than V.
- The policy gradient only updates the encoder θ_e and the decoder θ_d stays intact.

These properties reduce the difficulties for dialog policy optimization and decouple high-level decision-making from natural language generation. The p_{θ_e} are responsible for choosing the best latent action given a context $\mathbf{c}$ while p_{θ_d} is only responsible for transforming $\mathbf{z}$ into the surface-form words. Our formulation also provides a flexible framework for experimenting with various types of model learning methods. In this paper, we focus on two key aspects: the type of latent variable

$\mathbf{z}$ and optimization methods for learning $\mathbf{z}$ in the supervised pre-training step.

4.1 Types of Latent Actions

Two types of latent variables have been used in previous research: continuous isotropic Gaussian distribution (Serban et al., 2017b) and multivariate categorical distribution (Zhao et al., 2018). These two types are both compatible with our LaRL framework and can be defined as follows:

Gaussian Latent Actions follow M dimensional multivariate Gaussian distribution with a diagonal covariance matrix, i.e. $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$. Let the encoder p_{θ_e} consist of two parts: a context encoder $\mathcal{F}$, a neural network that encodes the dialog context $\mathbf{c}$ into a vector representation $\mathbf{h}$, and a feed forward network π that projects $\mathbf{h}$ into $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The process is defined as follows:

$$\mathbf{h} = \mathcal{F}(\mathbf{c}) \tag{4}$$

$$\begin{bmatrix}\boldsymbol{\mu} \\ \log(\boldsymbol{\sigma}^2)\end{bmatrix} = \pi(\mathbf{h}) \tag{5}$$

$$p(\mathbf{x}|\mathbf{z}) = p_{\theta_d}(\mathbf{z}) \quad \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \tag{6}$$

where the sampled $\mathbf{z}$ is used as the initial state of the decoder for response generation. Also we use $p_\theta(\mathbf{z}|\mathbf{c}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$ to compute the policy gradient update in Eq 3.

Categorical Latent Actions are M independent K-way categorical random variables. Each $\mathbf{z}_m$ has its own token embeddings to map latent symbols into vector space $\mathbf{E}_m \in \mathbb{R}^{K \times D}$ where $m \in [1, M]$ and D is the embedding size. Thus M latent actions can represent exponentially, K^M, unique combinations, making it expressive enough to model dialog acts in complex domains. Similar to Gaussian Latent Actions, we have

$$\mathbf{h} = \mathcal{F}(\mathbf{c}) \tag{7}$$

$$p(Z_m|\mathbf{c}) = \text{softmax}(\pi_m(\mathbf{h})) \tag{8}$$

$$p(\mathbf{x}|\mathbf{z}) = p_{\theta_d}(\mathbf{E}_{1:M}(\mathbf{z}_{1:M})) \quad \mathbf{z}_m \sim p(Z_m|\mathbf{c}) \tag{9}$$

For the computing policy gradient in Eq 3, we have $p_\theta(\mathbf{z}|\mathbf{c}) = \prod_{m=1}^{M} p(Z_m = \mathbf{z}_m|\mathbf{c})$

Unlike Gaussian latent actions, a matrix $\mathbb{R}^{M \times D}$ comes after the embedding layers $\mathbf{E}_{1:M}(\mathbf{z}_{1:M})$, whereas the decoder's initial state is a vector of size $\mathbb{R}^D$. Previous work integrated this matrix with the decoder by summing over the latent embeddings, i.e. $\mathbf{x} = p_{\theta_d}(\sum_1^M \mathbf{E}_m(\mathbf{z}_m))$, denoted as **Summation Fusion** for later discussion (Zhao et al., 2018). A limitation of this method is that it could lose fine-grained order information in each latent dimension and have issues with long responses that involve multiple dialog acts. Therefore, we propose a novel method, **Attention Fusion**, to combine categorical latent actions with the decoder. We apply the attention mechanism (Luong et al., 2015) over latent actions as the following. Let i be the step index during decoding. Then we have:

$$\alpha_{mi} = \text{softmax}(\mathbf{h}_i^T \mathbf{W}_a \mathbf{E}_m(\mathbf{z}_m)) \tag{10}$$

$$\mathbf{c}_i = \sum_{m=1}^{M} \alpha_{mi} \mathbf{E}_m(\mathbf{z}_m) \tag{11}$$

$$\widetilde{\mathbf{h}}_i = \tanh(\mathbf{W}_s \begin{bmatrix}\mathbf{h}_i \\ \mathbf{c}_i\end{bmatrix}) \tag{12}$$

$$p(w_i|\mathbf{h}_i, \mathbf{c}_i) = \text{softmax}(W_o \widetilde{\mathbf{h}}_i) \tag{13}$$

The decoder's next state is updated by $\mathbf{h}_{i+1} = \text{RNN}(\mathbf{h}_i, w_{i+1}), \widetilde{\mathbf{h}}_i)$ and $\mathbf{h}_0$ is computed via summation-fusion. Thus attention fusion lets the decoder focus on different latent dimensions at each generation step.

4.2 Optimization Approaches

Full ELBO: Now given a training dataset $\{\mathbf{x}, \mathbf{c}\}$, our base optimization method is via stochastic variational inference by maximizing the evidence lowerbound (ELBO), a lowerbound on the data log likelihood:

$$\mathcal{L}_{full}(\theta) = p_{q(\mathbf{z}|\mathbf{x},\mathbf{c})}(\mathbf{x}|\mathbf{z}) - \text{D}_{\text{KL}}[q(\mathbf{z}|\mathbf{x}, \mathbf{c})\|p(\mathbf{z}|\mathbf{c})] \tag{14}$$

where $q_\gamma(\mathbf{z}|\mathbf{x}, \mathbf{c})$ is a neural network that is trained to approximate the posterior distribution $q(\mathbf{z}|\mathbf{x}, \mathbf{c})$ and $p(\mathbf{z}|\mathbf{c})$ and $p(\mathbf{x}|\mathbf{z})$ are achieved by $\mathcal{F}$, π and p_{θ_d}. For Gaussian latent actions, we use the reparametrization trick (Kingma and Welling, 2013) to backpropagate through Gaussian latent actions and the Gumbel-Softmax (Jang et al., 2016) to backpropagate through categorical latent actions.

Lite ELBO: a major limitation is that Full ELBO can suffer from exposure bias at latent space, i.e. the decoder only sees $\mathbf{z}$ sampled from $q(\mathbf{z}|\mathbf{x}, \mathbf{c})$ and never experiences $\mathbf{z}$ sampled from $p_\theta(\mathbf{z}|\mathbf{c})$, which is always used at testing time. Therefore, in this paper, we propose a simplified ELBO for encoder-decoder models with stochastic latent variables:

$$\mathcal{L}_{lite}(\theta) = p_{p(\mathbf{z}|\mathbf{c})}(\mathbf{x}|\mathbf{z}) - \beta \text{D}_{\text{KL}}[p(\mathbf{z}|\mathbf{c}))\|p(\mathbf{z})] \tag{15}$$

Essentially this simplified objective sets the posterior network the same as our encoder, i.e. $q_\gamma(\mathbf{z}|\mathbf{x},\mathbf{c}) = p_{\theta_e}(\mathbf{z}|\mathbf{c})$, which makes the KL term in Eq 14 zero and removes the issue of exposure bias. But this leaves the latent spaces unregularized and our experiments show that if we only maximize $p_{p(\mathbf{z}|\mathbf{c})}(\mathbf{x}|\mathbf{z})$ there is overfitting. For this, we add the additional regularization term $\beta \mathrm{D}_{\mathrm{KL}}[p(\mathbf{z}|\mathbf{c}))\|p(\mathbf{z})]$ that encourages the posterior be similar to certain prior distributions and β is a hyper-parameter between 0 and 1. We set the $p(\mathbf{z})$ for categorical latent actions to be uniform, i.e. $p(\mathbf{z}) = 1/K$, and set the prior for Gaussian latent actions to be $\mathcal{N}(\mathbf{0},\mathbf{I})$, which we will show that are effective.

5 Experiment Settings

5.1 DealOrNoDeal Corpus and RL Setup

DealOrNoDeal is a negotiation dataset that contains 5805 dialogs based on 2236 unique scenarios (Lewis et al., 2017). We hold out 252 scenarios for testing environment and randomly sample 400 scenarios from the training set for validation. The results are evaluated from 4 perspectives: Perplexity (PPL), Reward, Agree and Diversity. PPL helps us to identify which model produces the most human-like responses, while Reward and Agree evaluate the model's negotiation strength. Diversity indicates whether the model discovers a novel discourse-level strategy or just repeats dull responses to compromise with the opponent. We closely follow the original paper and use the same reward function and baseline calculation. At last, to have a fair comparison, all the compared models shared the identical judge model and user simulator, which are a standard hierarchical encoder-decoder model trained with Maximum Likelihood Estimation (MLE).

5.2 Multi-Woz Corpus and Novel RL Setup

Multi-Woz is a slot-filling dataset that contains 10438 dialogs on 6 different domains. 8438 dialogs are for training and 1000 each are for validation and testing. Since no prior user simulator exists for this dataset, for a fair comparison with the previous state-of-the-art we focus on the Dialog-Context-to-Text Generation task proposed in (Budzianowski et al., 2018). This task assumes that the model has access to the ground-truth dialog belief state and is asked to generate the next response at every system turn in a dialog. The results are evaluated from 3 perspectives: BLEU, Inform Rate and Success Rate. The BLEU score checks the response-level lexical similarity, while Inform and Success Rate measure whether the model gives recommendations and provides all the requested information at dialog-level. Current state-of-the-art results struggle in this task and MLE models only achieve 60% success (Budzianowski et al., 2018). To transform this task into an RL task, we propose a novel extension to the original task as follows:

1. For each RL episode, randomly sample a dialog from the training set
2. Run the model on every system turn, and do not alter the original dialog context at every turn given the generated responses.
3. Compute Success Rate based on the generated responses in this dialog.
4. Compute policy gradient using Eq 3 and update the parameters.

This setup creates a variant RL problem that is similar to the Contextual Bandits (Langford and Zhang, 2008), where the goal is to adjust its parameters to generate responses that yield better Success Rate. Our results show that this problem is challenging and that word-level RL falls short.

5.3 Language Constrained Reward (LCR) curve for Evaluation

It is challenging to quantify the performance of RL-based neural generation systems because it is possible for a model to achieve high task reward and yet not generate human language (Das et al., 2017). Therefore, we propose a novel measure, the Language Constrained Reward (LCR) curve as an additional robust measure. The basic idea is to use an ROC-style curve to visualize the tradeoff between achieving higher reward and being faithful to human language. Specifically, at each checkpoint i over the course of RL training, we record two measures: (1) the PPL of a given model on the test data $p_i = \mathrm{PPL}(\theta_i)$ and (2) this model's average cumulative task reward in the test environment R_i^t. After RL training is complete, we create a 2D plot where the x-axis is the maximum PPL allowed, and the y-axis is the best achievable reward within the PPL budget in the testing environments:

$$y = \max_i R_i^t \quad \text{subject to} \quad p_i < x \tag{16}$$

As a result, a perfect model should lie in the upper left corner whereas a model that sacrifices language quality for higher reward will lie in the lower right corner. Our results will show that the LCR curve is an informative and robust measure for model comparison.

6 Results: Latent Actions or Words?

We have created 6 different variations of latent action dialog models under our LaRL framework. To demonstrate the advantages of LaRL, during the RL training step, we set RL:SL=off for all latent action models, while the baseline word-level RL models are free to tune RL:SL for best performance. For latent variable models, their perplexity is estimated via Monte Carlo $p(\mathbf{x}|\mathbf{c}) \approx \mathbb{E}_{p(\mathbf{z}|\mathbf{c})}[p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|\mathbf{c})]$. For the sake of clarity, this section only compares the best performing latent action models to the best performing word-level models and focuses on the differences between them. A detailed comparison of the 6 latent space configurations is addressed in Section 7.

Model	Var Type	Loss	Integration
Gauss	Gaussian	$\mathcal{L}_{full}$	/
Cat	Categorical	$\mathcal{L}_{full}$	sum
AttnCat	Categorical	$\mathcal{L}_{full}$	attn
LiteGauss	Gaussian	$\mathcal{L}_{lite}$	/
LiteCat	Categorical	$\mathcal{L}_{lite}$	sum
LiteAttnCat	Categorical	$\mathcal{L}_{lite}$	attn

Table 1: All proposed variations of LaRL models.

6.1 DealOrNoDeal

The baseline system is a hierarchical recurrent encoder-decoder (HRED) model (Serban et al., 2016) that is tuned to reproduce results from (Lewis et al., 2017). Word-level RL is then used to fine-tune the pre-trained model with RL:SL=4:1. On the other hand, the best performing latent action model is LiteCat. Best models are chosen based on performance on the validation environment.

The results are summarized in Table 2 and Figure 2 shows the LCR curves for the baseline with the two best models plus LiteAttnCat and baseline without RL:SL. From Table 2, it appears that the word-level RL baseline performs better than LiteCat in terms of rewards. However, Figure 2 shows that the two LaRL models achieve strong task rewards with a much smaller performance drop in language quality (PPL), whereas the word-level model can only increase its task rewards by deviating significantly from natural language.

	PPL	Reward	Agree%	Diversity
Baseline	5.23	3.75	59	109
LiteCat	5.35	2.65	41	58
Baseline +RL	8.23	**7.61**	86	5
LiteCat +RL	**6.14**	7.27	**87**	**202**

Table 2: Results on DealOrNoDeal. Diversity is measured by the number of unique responses the model used in all scenarios from the test data.

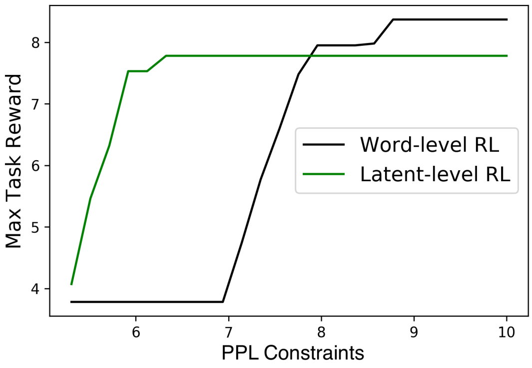


Figure 2: LCR curves on DealOrNoDeal dataset.

Closer analysis shows the word-level baseline severely overfits to the user simulator. The caveat is that the word-level models have in fact discovered a loophole in the simulator by insisting on 'hat' and 'ball' several times and the user model eventually yields to agree to the deal. This is reflected in the diversity measure, which is the number of unique responses that a model uses in all 200 testing scenarios. As shown in Figure 3, after RL training, the diversity of the baseline model drops to only 5. It is surprising that the agent can achieve high reward with a well-trained HRED user simulator using only 5 unique utterances. On the contrary, LiteCat increases its response diversity after RL training from 58 to 202, suggesting that LiteCat discovers novel discourse-level strategies in order to win the negotiation instead of exploiting local loopholes in the same user simulator. Our qualitative analysis confirms this when we observe that our LiteCat model is able to use multiple strategies in negotiation, e.g. elicit preference question, request different offers, insist on key objects etc. See supplementary material for example conversations.

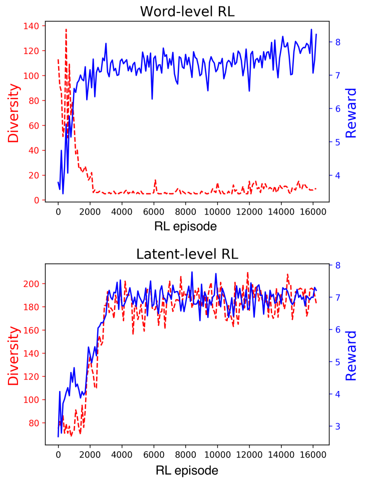


Figure 3: Response diversity and task reward learning curve over the course of RL training for both word RL:SL=4:1 (left) and LiteCat (right).

6.2 MultiWoz

For MultiWoz, we reproduce results from (Budzianowski et al., 2018) as the baseline. After RL training, the best LaRL model is LiteAttnCat and the best word-level model is word RL:SL=off. Table 3 shows that LiteAttnCat is on par with the baseline in the supervised learning step, showing that multivariate categorical latent variables alone are powerful enough to match with continuous hidden representations for modeling dialog actions. For performance after RL training, LiteAttnCat achieves near-human performance in terms of success rate and inform rate, obtaining 18.24% absolute improvement over the MLE-based state-of-the-art (Budzianowski et al., 2018). More importantly, perplexity only slightly increases from 4.05 to 5.22. On the other hand, the word-level RL's success rate also improves to 79%, but the generated responses completely deviate from natural language, increasing perplexity from 3.98 to 17.11 and dropping BLEU from 18.9 to 1.4.

	PPL	BLEU	Inform	Success
Human	/	/	90%	82.3%
Baseline	3.98	18.9	71.33%	60.96%
LiteAttnCat	4.05	19.1	67.98%	57.36%
Baseline +RL	17.11	1.4	80.5%	79.07%
LiteAttnCat +RL	**5.22**	**12.8**	**82.78%**	**79.2%**

Table 3: Main results on MultiWoz test set. RL models are chosen based on performance on the validation set.

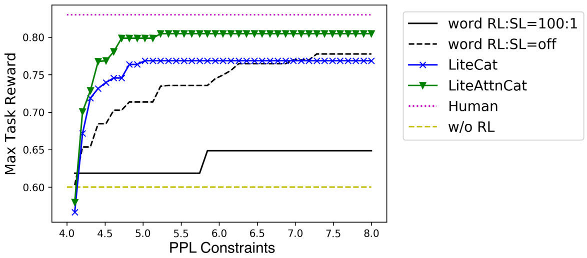


Figure 4: LCR curves on the MultiWoz dataset.

Figure 4 shows the LCR curves for MultiWoz, with a trend similar to the previous section: the word-level models can only achieve task reward improvement by sacrificing their response decoder PPL. Figure 4 also shows the LCR curve for the baseline trained with RL:SL=100:1, hoping that supervised learning can force the model to conform to natural language. While PPL and BLEU are indeed improved, it also limits final reward performance. The latent-level models, on the contrary, do not suffer from this tradeoff. We also observe that LiteAttnCat consistently outperforms LiteCat on MultiWoz, confirming the effectiveness of Attention Fusion for handling long dialog responses with multiple entities and dialog acts. Lastly, Table 4 qualitatively exhibits the generation differences between the two approaches. The RL:SL=off model learns to continuously output entities to fool the evaluation script for high success rate, whereas LiteCatAttn learns to give more information while maintaining the language quality.

Context	**Sys** I have [value_count] trains matching your request . Is there a specific day and time you would like to travel? **Usr** I would like to leave on [value_day] and arrive by [value_time].
Model	**Generated Response**
word RL:SL=off	[train_id] is leaving [value_place] on [value_day] on [value_day] on [train_id] [train_id] [value_count] [train_id] leaving ...
word RL:SL=100	[train_id] leaves at [value_time] . would you like me to book you a ticket ?
LiteAttnCat	[train_id] leaves [value_place] at [value_time] and arrives in [value_place] at [value_time]. Would you like me to book that for you ?

Table 4: Example responses from baselines and LiteCatAttn on MultiWoz.

7 Model Analysis

We compare the 6 variants of latent action models on DealOrNoDeal and MultiWoz. Table 5 shows performance of the models that are pre-trained only with supervised learning. Figure 5 shows LCR curves for the 3 models pre-trained with $\mathcal{L}_{lite}$ and fine-tuned with policy gradient reinforcement learning. The following are the main findings based on these results.

Deal	PPL	Reward	Agree%	Diversity
Baseline	3.23	3.75	59	109
Gauss	110K	2.71	43	176
LiteGauss	5.35	4.48	65	91
Cat	80.41	3.9	62	115
AttnCat	118.3	3.23	51	145
LiteCat	5.35	2.67	41	58
LiteAttnCat	5.25	3.69	52	75
MultiWoz	PPL	BLEU	Inform%	Succ%
Baseline	3.98	18.9	71.33	60.96
Gauss	712.3	7.54	60.5	23.0
LiteGauss	4.06	19.3	56.46	48.06
Cat	7.07	13.7	54.15	42.04
AttnCat	12.01	12.6	63.9	45.8
LiteCat	4.10	19.1	61.56	49.15
LiteAttnCat	4.05	19.1	67.97	57.36

Table 5: Comparison of 6 model variants with only supervised learning training.

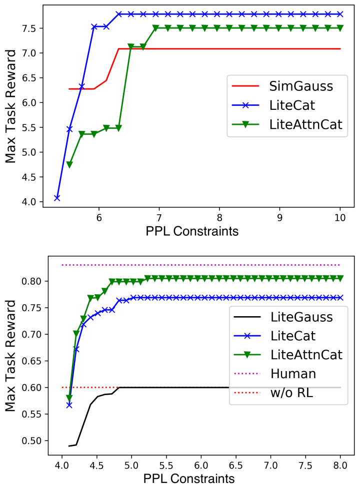


Figure 5: LCR curves on DealOrNoDeal and MultiWoz. Models with $\mathcal{L}_{full}$ are not included because their PPLs are too poor to compare to the Lite models.

$\mathcal{L}_{lite}$ outperforms $\mathcal{L}_{full}$ as a pre-train objective. Table 5 shows that models with $\mathcal{L}_{full}$ fall behind their Lite counterparts on PPL and BLEU. We attribute this to the exposure bias in the latent space, i.e. the decoder is not trained to consider the discrepancy between the posterior network and actual dialog policy network. Meanwhile, the full models tend to enjoy higher diversity at pre-training, which agrees with the diversity-promoting effect observed in prior research (Zhao et al., 2017). However, our previous discussion on Figure 3 shows that Lite models are able to increase their response diversity in order to win more in negotiation through RL training. This is fundamentally different from diversity in pre-training, since diversity in LaRL is optimized to improve task reward, rather than to better model the original data distribution. Table 6 shows the importance of latent space regularization. When β is 0, both LiteCat and LiteGauss reach suboptimal policies with final reward that are much smaller than the regularized versions ($\beta = 0.01$). The reason behind this is that the unregularized pre-trained policy has very low entropy, which prohibits sufficient exploration in the RL stage.

β	0.0	0.01	β	0.0	0.01
LiteCat	4.23	7.27	LiteGauss	4.83	6.67

Table 6: Best rewards in test environments on DealOrNoDeal with various β.

Categorical latent actions outperform Gaussian latent actions. Models with discrete actions consistently outperform models with Gaussian ones. This is surprising since continuously distributed representations are a key reason for the success of deep learning in natural language processing. Our finding suggests that (1) multivariate categorical distributions are powerful enough to model complex natural dialog responses semantics, and can achieve on par results with Gaussian or non-stochastic continuous representations. (2) categorical variables are a better choice to serve as action spaces for reinforcement learning. Figure 5 shows that Lite(Attn)Cat easily achieves strong rewards while LiteGauss struggles to improve its reward. Also, applying REINFORCE on Gaussian latent actions is unstable and often leads to model divergence. We suspect the reason for this is the unbounded nature of continuous latent space: RL exploration in the continuous space may lead to areas in the manifold that are not covered in supervised training, which causes undefined decoder behavior given $\mathbf{z}$ in these unknown areas.

8 Conclusion and Future Work

In conclusion, this paper proposes a latent variable action space for RL in E2E dialog agents. We present a general framework with a regularized ELBO objective and attention fusion for discrete variables. The methods are assessed on two dialog tasks and analyzed using the proposed LCR curve. Results show our models achieve superior performance and create a new state-of-the-art success rate on MultiWoz. Extensive analyses enable us to gain insight on how to properly train latent variables that can serve as the action spaces for dialog agents. This work is situated in the approach concerning practical latent variables in dialog agents, being able to create action abstraction in an unsupervised manner. We believe that our findings are a basic first step in this promising research direction.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 182–187.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 120–125. IEEE.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2970–2979. IEEE.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495.

Milica Gasic and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge naturallyin multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967.

John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *Proc. Interspeech 2017*, pages 2506–2510.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*. Citeseer.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2786.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017a. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Pei-Hao Su, Paweł Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 147–157.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, pages 387–416.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. In *International Conference on Machine Learning*, pages 3732–3741.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Denis Yarats and Mike Lewis. 2017. Hierarchical text generation and planning for strategic dialogue. *arXiv preprint arXiv:1712.05846*.

Stephanie Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The hidden information state approach to dialog management. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–149. IEEE.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Steve J Young. 2006. Using pomdps for dialog management. In *SLT*, pages 8–13.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 1.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

A Supplemental Material

A.1 Training Details

Supervised Pre-train	
Word Embedding	256
Utterance Encoder	Attn GRU (128)
Context (De)Encoder	GRU (256)
Optimizer	Adam (lr=1e-3)
Dropout	0.5
β	0.01
Categorical $\mathbf{z}$	M=10, K=20
Gaussian $\mathbf{z}$	M=200
Reinforce	
Optimizer	SGD (lr=0.2 grad_clip=0.1)
γ	0.95

Table 7: Training details for DealOrNoDeal experiments. Attn GRU refers to (Yang et al., 2016)

Supervised Pre-train	
Word Embedding	256
Encoder	Attn GRU (300)
Decoder	LSTM (150)
Optimizer	Adam (1e-3)
Dropout	0.5
β	0.01
Categorical $\mathbf{z}$	M=10, K=20
Gaussian $\mathbf{z}$	M=200
Reinforce	
Optimizer	SGD(lr=0.01 grad_clip=0.5)
γ	0.99

Table 8: Training details for MultiWoz experiments

A.2 DealOrNoDeal Example Conversations

The followings are examples dialogs generated from word-level models (Table 9) and latent-level models (Table 10).

Scenario 1
1 book, 1 hat and 3 balls available.
Value Function
Agent: book=1, hat=6, ball=1.
User: book=1, hat=6, ball=1.
Dialog
Agent: I'd like the hat and 2 balls.
User: Deal.
Agent: SELECTION
Outcome: Agreement
Agent: 1 hat and 2 balls; User: 1 book and 1 ball
Agent reward = 8; User reward = 2
Scenario 2
4 books, 1 hat and 1 ball available.
Value Function
Agent: book=0, hat=7, ball=3;
User: book=1, hat=6, ball=0.
Dialog
User: I can offer you 1 hat and 2 books.
Agent: I need the hat and ball.
User: That won't work for me. I can offer you one book and the hat.
Agent: I need the hat and ball.
User: I have to have the hat.
Agent: I need the hat and ball.
User: Okay.
Agent: SELECTION
Outcome: Agreement
Agent: 1 hat and 1 ball; User: 4 books
Agent reward = 10; User reward = 4

Table 9: Example dialogs between baseline with the user model. Agent is trained with word-level policy gradient and the user is a supervised pre-trained model.

Scenario 1
2 books, 2 hats and 1 ball available.
Value Function
Agent: book=3, hat=0, ball=4;
User: book=2, hat=3, ball=0.
Dialog
Agent: Can i have the books and one ball?
User: I would like the books and the hats then.
Agent: I would like the ball and books.
User: Deal.
Agent: SELECTION
Outcome: Agreement
Agent: 2 books and 1 ball
User: 2 hats
Agent reward = 10, User reward = 6
Scenario 2
2 books, 1 hat and 3 ball available.
Value Function
Agent: book=0, hat=4, ball=2;
User: book=3, hat=1, ball=1.
Dialog
Agent: I would like the hat and balls.
User: I cannot accept that. I need the hat and books.
Agent: Can I have the hat and balls, you get the rest?
User: Okay, deal.
Agent: SELECTION.
Outcome: Agreement
Agent: 1 hat and 3 ball
User: 2 books
Agent reward = 10, Simulator reward = 6

Table 10: Example dialogs between LiteCat and the user model. Agent is trained with latent-level policy gradient and the user is a supervised pre-trained model.

Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory

Deng Cai[1]*, **Yan Wang**[2]†, **Wei Bi**[2],
Zhaopeng Tu[2], **Xiaojiang Liu**[2], **Wai Lam**[1] and **Shuming Shi**[2]
[1]The Chinese University of Hong Kong, [2]Tencent AI Lab
thisisjcykcd@gmail.com, wlam@se.cuhk.edu.hk
{brandenwang,victoriabi,zptu,kieranliu,shumingshi}@tencent.com

Abstract

Traditional generative dialogue models generate responses solely from input queries. Such information is insufficient for generating a specific response since a certain query could be answered in multiple ways. Recently, researchers have attempted to fill the information gap by exploiting information retrieval techniques. For a given query, similar dialogues are retrieved from the entire training data and considered as an additional knowledge source. While the use of retrieval may harvest extensive information, the generative models could be overwhelmed, leading to unsatisfactory performance. In this paper, we propose a new framework which exploits retrieval results via a skeleton-to-response paradigm. At first, a skeleton is extracted from the retrieved dialogues. Then, both the generated skeleton and the original query are used for response generation via a novel response generator. Experimental results show that our approach significantly improves the informativeness of the generated responses.

1 Introduction

This paper focuses on tackling the challenges to develop a chit-chat style dialogue system (also known as chatbot). Chit-chat style dialogue system aims at giving meaningful and coherent responses given a dialogue query in the open domain. Most modern chit-chat systems can be categorized into two categories, namely, information retrieval-based (IR) models and generative models.

The IR-based models (Ji et al., 2014; Hu et al., 2014) directly copy an existing response from a training corpus when receiving a response request. Since the training corpus is usually collected from real-world conversations and possibly post-edited by a human, the retrieved responses are informative and grammatical. However, the performance of such systems drops when a given dialogue history is substantially different from those in the training corpus.

The generative models (Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016a), on the other hand, generate a new utterance from scratch. While those generative models have better generalization capacity in rare dialogue contexts, the generated responses tend to be universal and non-informative (e.g., “I don’t know”, “I think so” etc.) (Li et al., 2016a). It is partly due to the diversity of possible responses to a single query (i.e., the *one-to-many* problem). The dialogue query alone cannot decide a meaningful and specific response. Thus a well-trained model tends to generate the most frequent (safe but boring) responses instead.

To summarize, IR-based models may give informative but inappropriate responses while generative models often do the opposite. It is desirable to combine both merits. Song et al. (2016) used an extra encoder for the retrieved response. The resulted dense representation, together with the original query, is used to feed the decoder in a standard SEQ2SEQ model (Bahdanau et al., 2014). Weston et al. (2018) used a single encoder that takes the concatenation of the original query and the retrieved as input. Wu et al. (2019) noted that the retrieved information should be used in awareness of the context difference, and further proposed to construct an edit vector by explicitly encoding the lexical differences between the input query and the retrieved query.

However, in our preliminary experiments, we found that the IR-guided models are inclined to degenerate into a copy mechanism, in which the generative models simply repeat the retrieved response without necessary modifications. Sharp performance drop is caused when the retrieved re-

---

[*] Work done while DC was interning at Tencent AI Lab.
[†] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 1219–1228
Minneapolis, Minnesota, June 2 - June 7, 2019.

sponse is irrelevant to the input query. A possible reason is that both useful and useless information is mixed in the dense vector space, which is uninterpretable and uncontrollable.

To address the above issue, we propose a new framework, skeleton-to-response, for response generation. Our motivations are two folds: (1) The guidance from IR results should only specify a response aspect or pattern, but leave the query-specific details to be elaborated by the generative model itself; (2) The retrieval results typically contain excessive information, such as inappropriate words or entities. It is necessary to filter out irrelevant words and derive a useful skeleton before use.

Our approach consists of two components: a skeleton generator and a response generator. The skeleton generator extracts a response skeleton by detecting and removing unwanted words in a retrieved response. The response generator is responsible for adding query-specific details to the generated skeleton for query-to-response generation. A dialogue example illustrating our idea is shown in Fig. 1. Due to the discrete choice of skeleton words, the gradient in the training process is no longer differentiable from the response to the skeleton generator. Two techniques are proposed to solve this issue. The first technique is to employ the policy gradient method for rewarding the output of the skeleton generator based on the feedback from a pre-trained critic. An alternative technique is to solve both the skeleton generation and the response generation in a multi-task learning fashion.

Our contributions are summarized as below: (1) We develop a novel framework to inject the power of IR results into generative response models by introducing the idea of skeleton generation; (2) Our approach generates response skeletons by detecting and removing unnecessary words, which facilitates the generation of specific responses while not spoiling the generalization ability of the underlying generative models; (3) Experimental results show that our approach significantly outperforms other compared methods, resulting in more informative and specific responses.

2 Models

In this work, we propose to construct a response skeleton based on the results of IR systems for guiding the response generation. The skeleton-to-response paradigm helps reduce the search space of possible responses and provides useful elements missing in the given query.

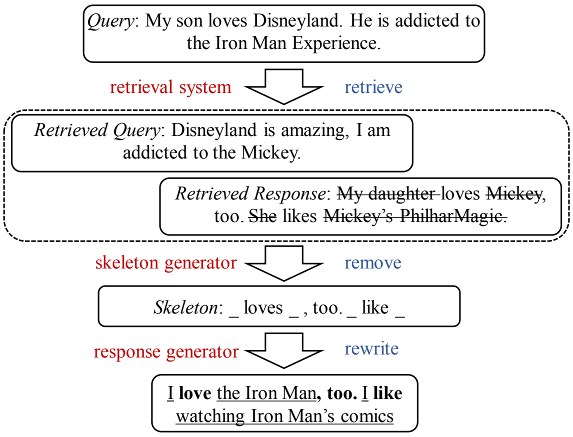


Figure 1: Our idea of leveraging the retrieved query-response pair. It first constructs a response skeleton by removing some words in the retrieved response, then a response is generated via rewriting based on the skeleton.

Our model consists of two components, namely, the skeleton generator and the response generator. These components are parameterized by the above two probabilistic models, denoted by θ_{ske} and θ_{res} respectively. Fig. 2 depicts the overall architecture of our proposed framework.

2.1 Skeleton Generator

The skeleton generator transforms a retrieved response into a skeleton by explicitly removing inappropriate or useless information regarding the input query q. We consider this procedure as a series of word-level masking actions. Following Wu et al. (2019), we first construct an edit vector by comparing the difference between the original query q and the retrieved query q'. In (Wu et al., 2019) the edit vector is used to guide the response generation directly. In our model, the edit vector is used to estimate the probability of being reserved or being masked for every word in a sentence. We define two word sets, namely *insertion words* I and *deletion words* D. The insertion words include words that are in the original query q, but not in the retrieved query q', while the deletion words do the opposite.

The two bags of words highlight the changes in the dialogue context, corresponding to the changes in the response. The edit vector z is thus defined as the concatenation of the representations of the two bags of words. We use the weighted sum of the

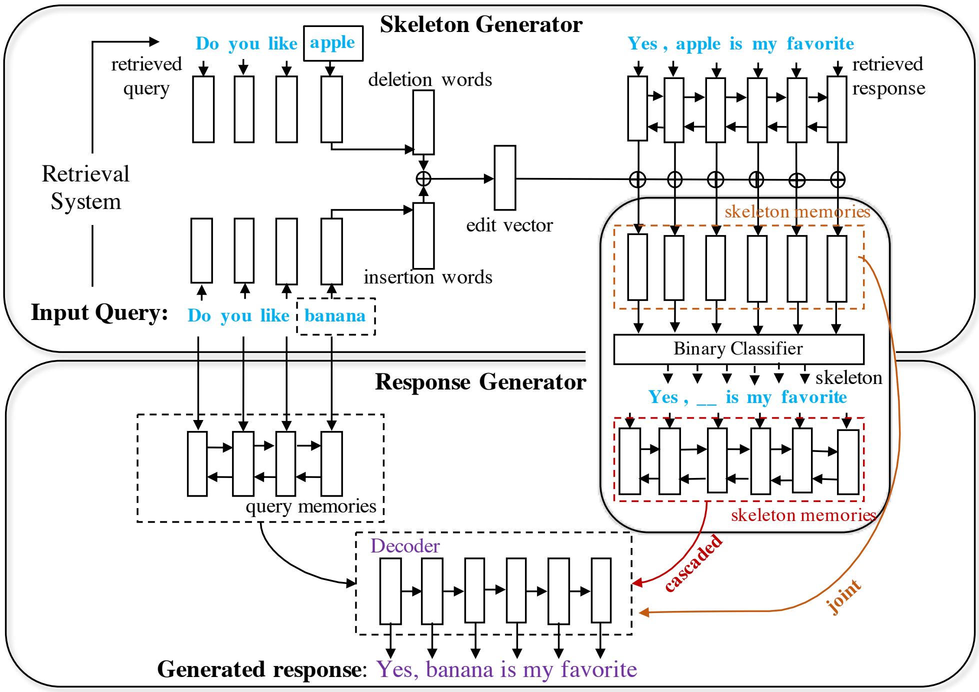


Figure 2: The architecture of our framework. Given a query "Do you like banana", a similar historical query "Do you like apple" is retrieved along with its response, i.e., "Yes, apple is my favorite". Upper: The skeleton generator removes inappropriate words and extracts a response skeleton. Lower: The response generator generates a response based on both the skeleton and the query.

word embeddings to get the dense representations of I and D. The edit vector is computed as:

$$z = \sum_{w_1 \in I} \alpha_{w_1} \Phi(w_1) \oplus \sum_{w_2 \in D} \beta_{w_2} \Phi(w_2), \quad (1)$$

where $\oplus$ is the concatenation operation. Φ maps a word to its corresponding embedding vector, α_{w_1} and β_{w_2} are the weights of an insertion word w_1 and a deletion word w_2 respectively. The weights of different words are derived by an attention mechanism (Luong et al., 2015). Formally, the retrieved response $r' = (r'_1, r'_2 \ldots, r'_{|r'|})$ is processed by a bidirectional GRU network (biGRU). We denote the states of the biGRU (i.e. concatenation of forward and backward GRU states) as $(h_1, h_2, \ldots, h_{|r'|})$. The weight α_{w_1} is calculated by:

$$\alpha_{w_1} = \frac{\exp(s_{w_1})}{\sum_{w \in I} \exp(s_w)},$$

$$s_{w_1} = v_I^\top \tanh(W_I[\Phi(w_1) \oplus h_{|r'|}]), \quad (2)$$

where v_I and W_I are learnable parameters. The weight β_{w_2} is obtained in a similar way with another set of parameters v_D and W_D.

After acquiring the edit vector, we transform the prototype response r' to a skeleton t by the following equations:

$$t = (\phi(r'_1, h_1, z), \phi(r'_2, h_2, z), \cdots, \phi(r'_{|r'|}, h_{|r'|}, z)),$$

$$\phi(r'_i, h_i, z) = \begin{cases} < \text{blank} > & \text{if } \hat{m}_i = 0, \\ r'_i & \text{else} \end{cases}, \quad (3)$$

where $\hat{m}_i$ is the indicator and equals 0 if r'_i is replaced with a placeholder "<blank>" and 1 otherwise. The probability of $\hat{m}_i = 1$ is computed by

$$P(\hat{m}_i = 1) = \text{sigmoid}(W_m[h_i \oplus z] + b_m). \quad (4)$$

2.2 Response Generator

The response generator can be implemented using most existing IR-augmented models (Song et al., 2016; Weston et al., 2018; Pandey et al., 2018), just by replacing the retrieved response input with the corresponding skeleton. We discuss our choices below.

Encoders Two separate bidirectional LSTM (biLSTM) networks are used to obtain the distributed representations of the query memories and the skeleton memories, respectively. For biLSTM, the concatenation of the forward and the backward hidden states at each token position is considered a memory slot, producing two memory pools: $\mathcal{M}_q = \{h_1, h_2, \dots, h_{|q|}\}$ for the input query, and $\mathcal{M}_t = \{h'_1, h'_2, \dots, h'_{|t|}\}$ for the skeleton.[1]

Decoder During the generation process, our decoder reads information from both the query and the skeleton using attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). To query the memory pools, the decoder uses the hidden state s_t of itself as the searching key. The matching score function is implemented by bilinear functions:

$$\alpha(h_k, s_t) = {h_k}^T W_q s_t; \beta(h'_k, s_t) = {h'_k}^T W_t s_t, \tag{5}$$

where W_q and W_t are trainable parameters. A query context vector c_t is then computed as a weighted sum of all memory slots in $\mathcal{M}_q$, where the weight for a memory slot h_k is $\exp(\alpha(h_k, s_t))/(\sum_{i=1}^{|q|} \exp(\alpha(h_i, s_t)))$. A skeleton context vector c'_t is computed in a similar spirit by using $\beta(h'_k, s_t)$'s.

The probability of generating the next word r_t is then jointly determined by the decoder's state s_t, the query context c_t and the skeleton context c'_t. We first fuse the information of s_t and c_t by a linear transformation. For c'_t, a gating mechanism is additionally introduced to control the information flow from skeleton memories. Formally, the probability of the next token r_t is estimated by y_t followed by a softmax function over the vocabulary:

$$y_t = (W_c[s_t \oplus c_t]) \cdot g_t + c'_t \cdot (1 - g_t), \tag{6}$$

where $g_t = f_g(s_t, c_t, c'_t)$ is implemented by a single layer neural network with sigmoid output layer.

3 Learning

Given that our skeleton generator performs non-differentiable hard masking, the overall model cannot be trained end-to-end using the standard maximum likelihood estimate (MLE). A possible solution that circumvents this problem is to treat the skeleton generation and the response generation as two parallel tasks and solve them jointly in a multi-task learning fashion. An alternative is to bridge the skeleton generator and the final response output using reinforcement learning (RL) methods, which can exclusively inform the skeleton generator with the ultimate goal. The latter option is referred as *cascaded integration* while the former is called *joint integration*.

Recall that we have formulated the skeleton generation as a series of binary classifications. Nevertheless, most of the dialogue datasets are end-to-end query-response pairs without explicit skeletons. Hence, we propose to construct *proxy skeletons* to facilitate the training.

Definition 1 Proxy Skeleton: *Given a training quadruplet (q, q', r, r') and a stop word list S, the proxy skeleton for r is generated by replacing some tokens in r' with a placeholder "<blank>". A token r'_i is kept if and only if it meets the following conditions*

1. $r'_i \notin S$

2. r'_i is a part of the longest common subsequence (LCS) (Wagner and Fischer, 1974) of r and r'.

The proxy skeletons are used in different manners according to the integration method, which we will introduce below.

3.1 Joint Integration

To avoid breaking the differentiable computation, we connect the skeleton generator and the response generator via a shared network architecture rather than by passing the discrete skeletons. Concretely, the last hidden states in our skeleton generator (i.e, the hidden states that are utilized to make the masking decisions) are used as the skeleton memories in response generation. The training objective is the sum of the proxy skeleton labels likelihood $L(\theta_{ske})$ and the response likelihood $L(\theta_{res})$:

$$L(\theta_{res} \cup \theta_{ske}) = L(\theta_{res}) + \eta L(\theta_{ske}), \tag{7}$$

where η is a harmonic weight, and it is set as 1.0 in our experiments.

3.2 Cascaded Integration

Policy gradient methods (Williams, 1992) can be applied to optimize the full model while keeping it running as cascaded process. We regard the skeleton generator as the first RL agent, and the response generator as the second one. The final output generated by the pipeline process and the

---

[1] Note the skeleton memory pool $\mathcal{M}_t$ could contain multiple response skeletons, further discussed in the experiment section.

intermediate skeleton are denoted by $\hat{r}$ and $\hat{t}$ respectively. Given the original query q and the generated response $\hat{r}$, a reward $R(q, \hat{r})$ for generating $\hat{r}$ is calculated. All network parameters are then optimized to maximize the expected reward by the policy gradient.

The reward function R should convey both the naturalness of the generated response and its relevance to the given query q. A pre-trained critic is utilized to make the judgment. Inspired by comparative adversarial learning in (Li et al., 2018), we design the critic as a classifier that receives four inputs every time: the query q, a human-written response r, a machine-generated response $\hat{r}$ and a random response $\overline{r}$ (yet written by human). The critic is trained to pick the human-written response r among others correctly. Formally, the following objective is maximized:

$$\log D(r|q, \hat{r}, \overline{r}, r) = \log \frac{\exp({h_r}^{\mathrm{T}} M_D h_q)}{\sum\limits_{x \in \{\hat{r}, \overline{r}, r\}} \exp({h_x}^{\mathrm{T}} M_D h_q)}, \tag{8}$$

where h_x is a vector representation of x, produced by a bidirectional LSTM (the last hidden state), and M_D is a trainable matrix.[2]

4 Related Work

Multi-source Dialogue Generation Chit-chat style dialogue system dates back to ELIZA (Weizenbaum, 1966). Early work uses hand-crafted rules, while modern systems usually use data-driven approaches, e.g., information retrieval techniques. Recently, end-to-end neural approaches (Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2016a; Sordoni et al., 2015) have attracted increasing interest. For those generative models, a notorious problem is the “safe response” problem: the generated responses are dull and generic, which may attribute to the lack of sufficient input information. The query alone cannot specify an informative response. To mitigate the issue, many research efforts have been paid to introducing other information source, such as unsupervised latent variable (Serban et al., 2017; Zhao et al., 2018; Cao and Clark, 2017; Shen et al., 2017), discourse-level variations (Zhao et al., 2017), topic information (Xing et al., 2017), speaker personality (Li et al., 2016b) and knowledge base (Ghazvininejad et al., 2018; Zhou et al., 2018). Our work follows the similar motivation and uses the output of IR systems as the additional knowledge source.

Combination of IR and Generative models To combine IR and generative models, early work (Qiu et al., 2017) tried to re-rank the output from both models. However, the performance of such models is limited by the capacity of individual methods. Most related to our work, Song et al. (2016); Weston et al. (2018) and Wu et al. (2019) encoded the retrieved result into distributed representation and used it as the additional conditionals along with the standard query representation. While the former two only used the target side of the retrieved pairs, the latter took advantages of both sides. In a closed domain conversation setting, Pandey et al. (2018) further proposed to weight different training instances by context similarity. Our model differs from them in that we take an extra intermediate step for skeleton generation to filter the retrieval information before use, which shows the effectiveness in avoiding the erroneous copy in our experiments.

Multi-step Language Generation Our work is also inspired by the recent success of decomposing an end-to-end language generation task into several sequential sub-tasks. For document summarization, Chen and Bansal (2018) first select salient sentences and then rewrite them in parallel. For sentiment-to-sentiment translation, Xu et al. (2018) first use a neutralization module to remove emotional words and then add sentiment to the neutralized content. Not only does their decomposition improve the overall performance, but also makes the whole generation process more interpretable. Our skeleton-to-response framework also sheds some light on the use of retrieval memories.

5 Experiments

5.1 Data

We use the preprocessed data in (Wu et al., 2019) as our test bed. The total dataset consists of about 20 million single-turn query-response pairs collected from Douban Group[3]. Since similar contexts may correspond to totally different responses, the training quadruples (q, r, q', r') for

---

[2] Note the classifier could be fine-tuned with the training of our generators, which falls into the adversarial learning setting (Goodfellow et al., 2014).

[3] https://www.douban.com/group

IR-augmented models are constructed based on response similarity. All response are indexed by Lucene.[4] For each (q, r) pair, top 30 similar responses with their corresponding contexts are retrieved $\{(q'_i, r'_i)\}_{i=1}^{30}$. However, only those satisfying $0.3 \leq Jaccard(r, r'_i) \leq 0.7$ are leveraged for training, where $Jaccard$ measures the Jaccard distance. The reason for the data filter is that nearly identical responses drive the model to do simple copy while distantly different responses make the model ignore the retrieval input. About 42 million quadruples are obtained afterward.

For computational efficiency, we randomly sample 5 million quadruples as training data for all experiments. The test set consists of 1,000 randomly selected queries that are not in our training data.[5] For a fair comparison, when training a generative model without the help of IR, the quadruples are split into pairs.

5.2 Model Details

We implement the skeleton generator based on a bidirectional recurrent neural network with 500 LSTM units. We concatenate the hidden states from both directions. The word embedding size is set to 300. For the response generator, the encoder for queries, the encoder for skeletons and the decoder are three two-layer recurrent neural networks with 500 LSTM units, where both encoders are bidirectional. We use dropout (Srivastava et al., 2014) to alleviate overfitting. The dropout rate is set to 0.3 across different layers. The same architecture for the encoders and the decoder is shared across the following baseline models, if applicable.

5.3 Compared Methods

- **Seq2Seq** the standard attention-based RNN encoder-decoder model (Bahdanau et al., 2014).
- **MMI** SEQ2SEQ with Maximum Mutual Information (MMI) objective in decoding (Li et al., 2016a). In practice, an inverse (response-to-query) SEQ2SEQ model is used to rerank the N-best hypothesizes from the standard SEQ2SEQ model (N equals 100 in our experiments).

model	human score	dist-1	dist-2
IR	2.093	**0.238**	**0.723**
IR+rerank	2.520	0.208	0.586
Seq2Seq	2.433	0.156	0.336
MMI	2.554	0.170	0.464
EditVec	2.588†	0.154	0.394
SKP	2.581	0.152	0.406
JNT	2.612†	0.147	0.377
CAS	**2.747**	0.156	0.411

Table 1: Response performance of different models. Sign tests on human score show that the CAS is significantly better than all other methods with p-value $<$ 0.05, and the p-value $<$ 0.01 except for those marked by $\dagger$.

model	P	R	F_1	Acc.
JNT	0.32	0.61	0.42	0.60
CAS	0.50	0.86	0.63	0.76

Table 2: Performance of skeleton generator.

- **EditVec** the model proposed by Wu et al. (2019), where the edit vector z is used directly at each decoding step by concatenating it to the word embeddings.
- **IR** the Lucene system is also used a benchmark.[6]
- **IR+rerank** rerank the results of **IR** by **MMI**.

Besides, We use **JNT** to denote our model with joint integration, and **CAS** for our model with cascaded integration. To validate the usefulness of the proposed skeletons. We design a response generator that takes an intact retrieval response as its skeleton input (i.e., to completely skip the skeleton generation step), denoted by **SKP**.[7]

5.4 Evaluation Metrics

Our method is designed to improve the informativeness of the generative model and alleviate the inappropriateness problem of the retrieval model. To measure the performance effectively, we use

---

[4] https://lucene.apache.org/core/

[5] Note the retrieval results for test data are based on query similarity, and no data filter is adopted.

[6] Note IR selects response candidates from the entire data collection, not restricted to the filtered one.

[7] There are some other IR-augmented models using standard SEQ2SEQ models as SKP. Weston et al. (2018) used a rule to select either the generated response or the retrieved response as output, while we would like to focus on improving the quality of generated responses. Pandey et al. (2018) concentrated on closed domain conversations, their hierarchical encoder is not suitable for our open domain setting. We thus omit the empirical comparison with them.

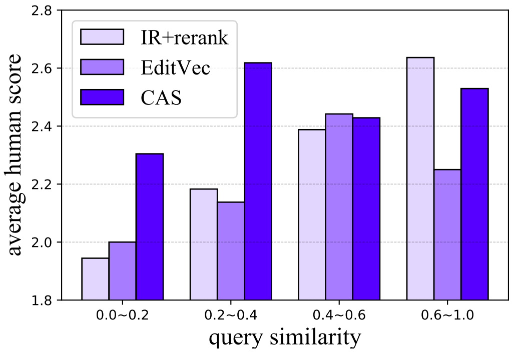


Figure 3: Response quality v.s. query similarity.[8]

human evaluation along with two automatic evaluation metrics.

- **Human evaluation** We asked three experienced annotators to score the group of responses (the best output of each model) for 300 test queries. The responses are rated on a five-point scale. A response should be scored 1 if it can hardly be considered a valid response, 3 if it is a valid but not informative response, 5 if it is an informative response, which can deepen the discussion of the current topic or lead to a new topic. 2 and 4 are for decision dilemmas.

- **dist-1 & dist-2** It is defined as the number of unique uni-grams (**dist-1**) or bi-grams (**dist-2**) dividing by the total number of tokens, measuring the diversity of the generated responses (Li et al., 2016a). Note the two metrics do not necessarily reflect the response quality as the target queries are not taken into consideration.

5.5 Response Generation Results

The results are depicted in Table 1. Overall, both of our models surpass all other methods, and our cascaded model (CAS) gives the best performance according to human evaluation. The contrast with the SKP model illustrates that the use of skeletons brings a significant performance gain.

According to the dist-1&2 metrics, the generative models achieve significantly better diversity by the use of retrieval results. The retrieval method yields the highest diversity, which is consistent with our intuition that the retrieval responses typically contain a large amount of information though they are not necessarily appropriate. The model of MMI also gives strong diversity, yet we find that it tends to merely repeat the words in queries. By removing the words in queries, the dist-2 of MMI and CAS become 0.710 and 0.751 respectively. It indicates our models are better at generating new words.

To further reveal the source of performance gain, we study the relation between response quality and query similarity (measured by the Jaccard similarity between the input query and the retrieved query). Our best model (CAS) is compared with the strong IR system (IR-rerank) and the previous state-of-the-art (EditVec) in Fig. 3. The CAS model significantly boosts the performance when query similarity is relatively low, which indicates that introducing skeletons can alleviate erroneous copy and keep a strong generalization ability of the underlying generative model.

5.6 More Analysis of Our Framework

Here, we present further discussions and empirical analysis of our framework.

Generated Skeletons Although generating skeletons is not our primary goal, it is interesting to assess the skeleton generation. The word-level precision (P), recall (R), F_1 score (F_1) and accuracy (Acc.) of the well-trained skeleton generators are reported in Table 2, taking the proxy skeletons as golden references.

Table 3 shows some skeleton-to-response examples of the CAS model and a case study among different models. In the leftmost example in Table 3, the MMI and the EditVec simply repeat the query while the retrieved response is weakly related to the query. Our CAS model extracts a useful word 'boy' from the retrieved response and generates a more interesting response. In the middle example, the MMI response makes less sense, and some private information is included in the retrieved response. Our CAS model removes the privacy without the loss of informativeness, while the outputs by other models are less informative. The rightmost case shows that our response generator is able to recover the possible mistakes made by the skeleton generator.

Retrieved Response v.s. Generated Response To measure the extent that the generative models are copying the retrieval, we compute the edit distances between generated responses and retrieved

---

[8] We merge the ranges $[0.6, 0.8]$ and $[0.8, 1.0]$ due to the sparsity of highly similar pairs.

Query	善良的妹子会幸福一生的 May you kind girls have a happy life	好的到时候有时间骑车出来玩嘛， 我觉得在大理最适合骑车玩了 OK, Let’s ride when free. It is Dali.	把你联系方式给我 Tell me your contact
Retrieved query	谢谢善良的妹子 Thanks to the kind girl	改天一起出来骑车吧,我们都喜欢玩单车 Let’s ride when free, as all of us like riding	把你姐姐联系方式给我 Tell me your sister’s contact
Retrieved response	我是汉子哈哈 I am a boy, LOL	ok 加微信abcdefg ok add my wechat abcdefg	我姐已婚 My sister is married.
Skeleton	_汉子_ _boy_	ok加微信_ OK, add my wechat _	_姐_已婚 _ sister _ married.
Generated response (CAS)	哈哈哈，我是善良的**汉子** LOL, I am a kind **boy**	你**加**我**微信**吧，我也在大理 **Add** my **wechat**, I am in Dali, too	我**已婚**哈哈 I am **married**, LOL
IR+rerank	谢谢嘛 Thanks	ok 加微信: abcdefg OK, add my Wechat: abcdefg	我姐已婚 My sister is married.
MMI	我不是善良的妹纸 I am not a kind girl	我也想骑在大理 I want to ride Dali, too	你怎么知道我联系方式 How did you know my contact
EditVec	谢谢，善良的妹子也会幸福的 Thanks, a kind girl will be happy too	有时间可以去玩 Let’s go when free	哈哈哈哈好啊 LOL, Ok!
JNT	我不是善良的妹子 I am not a kind girl	我也在大理 I am in Dali, too	我不是联系的 I am not the contact person

Table 3: Upper: Skeleton-to-response examples of the CAS model. Lower: Responses from different models are for comparison.

responses. As shown in Fig. 4, in the comparison between the SKP and other models, the use of skeletons makes the generated response deviate more from its prototype response. Ideally, when the retrieved context is very similar to the input query, the changes between the generated response and the prototype response should be minor. Conversely, the changes should be drastic. Fig. 4 also shows that our models can learn this intuition.

Single v.s. Multiple Retrieval Pair(s) For a given query q, the retrieval pair set R_q could contain multiple query-response pairs. We investigate two ways of using it under the CAS setting.

- **Single** For each query-response pair $(q'_i, r'_i) \in R_q$, a response $\hat{r}_i$ is generated solely based on q, and (q'_i, r'_i). The resulted responses are re-ranked by generation probability.

- **Multiple** The whole retrieval set R_q is used in a single run. Multiple skeletons are generated and concatenated in the response generation stage.

The results are shown in Table 4. We attribute the failure of **Multiple** to the huge variety of the retrieved responses. The response generator receives many heterogeneous skeletons, yet it has no idea which to use. It remains an open question on how to effectively use multiple retrieval pairs for generating one single response, and we leave it for future work.

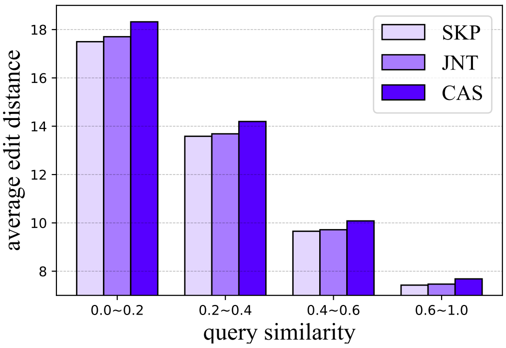


Figure 4: Changes between retrieved and generated responses v.s. query similarity.

setting	human score	dist-1	dist-2
Single	2.747	0.156	0.411
Multiple	1.976	0.178	0.414

Table 4: Comparison of the usages of the retrieval set.

6 Conclusion

In this paper, we proposed a new methodology to enhance generative models with information retrieval technologies for dialogue response generation. Given a dialogue context, our methods generate a skeleton based on historical responses that respond to a similar context. The skeleton serves as an additional knowledge source that helps specify the response direction and complement the response content. Experiments on real world data validated the effectiveness of our method for more informative and appropriate responses.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. The work described in this paper is partially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *EACL*, pages 182–187.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *AAAI*, pages 5110–5117.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Dianqi Li, Xiaodong He, Qiuyuan Huang, Ming-Ting Sun, and Lei Zhang. 2018. Generating diverse and accurate visual captions by comparative adversarial learning. *arXiv preprint arXiv:1804.00861*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*, pages 994–1003.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL*, pages 1329–1338.

Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *ACL*, pages 498–503.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *ACL*, pages 504–509.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, pages 196–205.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML (Deep Learning Workshop)*.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, pages 3351–3357.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *ACL*, page 675–686.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *ACL*, pages 1098–1107.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*, pages 654–664.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Jointly Optimizing Diversity and Relevance in Neural Response Generation

Xiang Gao **Sungjin Lee** **Yizhe Zhang**
Chris Brockett **Michel Galley** **Jianfeng Gao** **Bill Dolan**
Microsoft Research, Redmond, WA, USA
`{xiag,sule,yizzhang,chrisbkt,mgalley,jfgao,billdol}@microsoft.com`

Abstract

Although recent neural conversation models have shown great potential, they often generate bland and generic responses. While various approaches have been explored to diversify the output of the conversation model, the improvement often comes at the cost of decreased relevance (Zhang et al., 2018). In this paper, we propose a SpaceFusion model to jointly optimize diversity and relevance that essentially fuses the latent space of a sequence-to-sequence model and that of an autoencoder model by leveraging novel regularization terms. As a result, our approach induces a latent space in which the distance and direction from the predicted response vector roughly match the relevance and diversity, respectively. This property also lends itself well to an intuitive visualization of the latent space. Both automatic and human evaluation results demonstrate that the proposed approach brings significant improvement compared to strong baselines in both diversity and relevance. [1]

1 Introduction

The field of neural response generation is advancing rapidly both in terms of research and commercial applications (Gao et al., 2019; Zhou et al., 2018; Yoshino et al., 2019; Zhang et al., 2019). Nevertheless, vanilla sequence-to-sequence (S2S) models often generate bland and generic responses (Li et al., 2016a). Li et al. (2016a) encourage diversity by re-ranking the beam search results according to their mutual information with the conversation context. However, as beam search itself often produces lists of nearly identical sequences, this method can require a large beam width (e.g. 200). As a result, re-ranking can be extremely time-consuming, raising difficulties for real-time applications. This highlights the need to improve the diversity of candidates before re-ranking, and the need to optimize for diversity during training rather than just at the decoding stage.

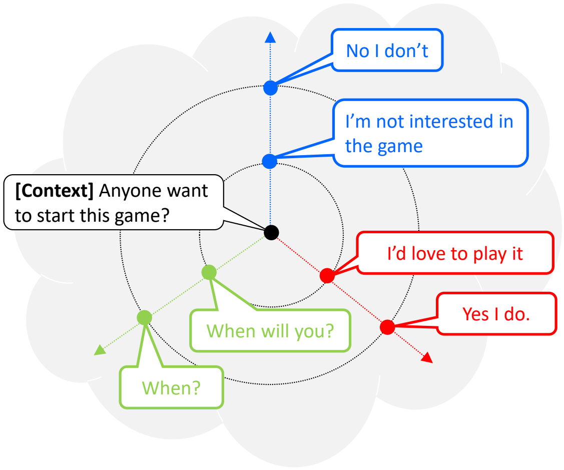


Figure 1: Illustration of one **context** and its multiple **responses** in the latent space induced by our model. Distance and direction from the predicted response vector given the context roughly match the relevance and diversity, respectively. Based on the example in Table 2.[2]

While various approaches have been explored to diversify the output of conversation models, the improvement often comes at the cost of decreased response relevance along other dimensions. For instance, Zhao et al. (2017) present an approach to enhancing diversity by mapping diverse responses to a probability distribution using a conditional variational autoencoder (CVAE). Despite the improved response diversity, this approach reduces response relevance as measured against the baseline. One possible reason for this diversity-relevance trade-off is that such probabilistic approaches are not explicitly encouraged to induce a disentangled representation in latent space for

---

[1] An implementation of our model is available at `https://github.com/golsun/SpaceFusion`

[2] For simplicity, we omitted the response at the center: "I would love to play this game". See Table 2 for more details.

Proceedings of NAACL-HLT 2019, pages 1229–1238
Minneapolis, Minnesota, June 2 - June 7, 2019.

controlling diversity and relevance independently. Consider a Gaussian distribution, which is widely used for CVAE. A Gaussian distribution naturally brings frequent responses near its mean, and the resulting responses are often generic and boring. To generate diverse and interesting responses, one needs to sample a little distance from the mean. But doing so naturally leads to infrequent and thus even irrelevant responses.

In this paper, we propose a novel geometrical approach that explicitly encourages a structured latent space in which the distance and direction from a predicted response vector roughly match the relevance and diversity, respectively, as illustrated in Figure 1. To induce such a latent space, we leverage two different models: 1) a S2S model, producing the predicted response vector (the black dot at the center in Figure 1), and 2) an autoencoder (AE) model, yielding the vectors for potential responses (the colored dots). In order to make the S2S and AE share the same latent space (the cloud), we use the same decoder for both and train them jointly end-to-end with novel regularization terms. As this fuses the two latent spaces, we refer to our model as SPACEFUSION.

Regularization is necessary because only sharing the decoder, as in (Luan et al., 2017), does not necessarily align the latent spaces obtained by S2S and AE respectively or impose a disentangled structure onto the space. We introduce two regularization terms to tackle this issue. 1) interpolation term: we encourage a smooth semantic transition along the path between the predicted response vector and each target response vector (arrowed lines in Figure 1). This term effectively prevents semantically different responses from aligning in the same direction, essentially scattering them over different directions. 2) fusion term: we want the vectors from the two models to be distributed in a homogeneous manner, rather than forming two separate clusters (Figure 5) that can potentially make sampling non-trivial. With the resulting latent space, we can control relevance and diversity by respectively adjusting distance and direction from a predicted response vector, without sacrificing each other greatly.

Our approach also lends itself well to the intuitive visualization of latent space. Since our model allows us to geometrically find not only the predicted response vector but also the target response vector as in Figure 5, we can visually interpret the structure of latent space and identify major issues thereof. We devote Section 5.1 to show comprehensive examples for visualization-based analysis.

Automatic and human evaluations demonstrate that the proposed approach improves both the diversity and relevance of the responses, compared to strong baselines on two datasets with one-to-many context-response mapping.

2 Related Work

Grounded conversation models utilize extra context inputs besides conversation history, such as persona (Li et al., 2016b), textual knowledge (Ghazvininejad et al., 2017; Galley et al., 2019), dialog act (Zhao et al., 2017) and emotion (Huber et al., 2018). Our approach does not depend on such extra input and thus is complementary to this line of studies.

Variational autoencoder (VAE) models explicitly model the uncertainty of responses in latent space. Bowman et al. (2016) used VAE with Long-Short Term Memory (LSTM) cells to generate sentences. The basic idea of VAE is to encode the input x into a probability distribution (e.g. Gaussian) z instead of a point encoding. However, it suffers from the *vanishing latent variable problem* (Bowman et al., 2016; Zhao et al., 2017) when applied to text generation tasks. Bowman et al. (2016); Fu et al. (2019) proposed to tackle this problem with word dropping and specific KL annealing methods. Zhao et al. (2017) proposed to add a *bag-of-word* loss, complementary to KL annealing. Applying this to a CVAE conversation model, they showed that even greedy decoding can generate diverse responses. However, as VAE/CVAE conversation models can be limited to a simple latent representations such as standard Gaussian distribution, Gu et al. (2018) proposed to enrich the latent space by leveraging a Gaussian mixture prior. Our work takes a geometrical approach that is fundamentally different from probabilistic approaches to tackle the limitations of parameteric distributions in representation and difficulties in training.

Decoding and ranking encourage diversity during the decoding stage. As "vanilla" beam search often produces lists of nearly identical sequences, Vijayakumar et al. (2016) propose to include a dissimilarity term in the objective of beam search decoding. Li et al. (2016a) re-ranked the results obtained by beam search based on mutual information with the context using a separately trained response-to-context S2S model.

Multi-task learning is another line of studies related to the present work (see Section 3.2). Sennrich et al. (2016) use multi-task learning to improve neural machine translation by utilizing monolingual data, which usually far exceeds the amount of parallel data. A similar idea is applied by Luan et al. (2017) to conversational modeling, involving two tasks: 1) a S2S model that learns a context-to-response mapping using conversation data, and 2) an AE model that utilizes speaker-specific non-conversational data. The decoders of S2S and AE were shared, and the two tasks were trained alternately.

3 The SPACEFUSION Model

3.1 Problem statement

Let $\mathcal{D} = [(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)]$ denote a conversational dataset, where x_i and y_i are a context and its response, respectively. x_i consists of one or more utterances. Our aim is to train a model on $\mathcal{D}$ to generate relevant and diverse responses given a context.

3.2 Fusing latent spaces

We design our model to induce a latent space where different responses for a given context are in different directions around the predicted response vector, as illustrated in Figure 1. Then we can obtain diverse responses by varying the direction and keep their relevance by sampling near the predicted response vector.

To fulfill this goal, we first produce the predicted response representation z_{S2S} and target response representations z_{AE} using an S2S model and an AE model, respectively, as illustrated in Figure 2. Both encoders are implemented using stacked Gated Recurrent Unit (GRU) (Cho et al., 2014) cells followed by a noise layer that adds multivariate Gaussian noise $\epsilon \sim N(0, \sigma^2\mathbf{I})$. We then explicitly encourage smooth semantic transition along the path from z_{S2S} to z_{AE} by imposing any interpolation between them to generate the same response via the following loss term:

$$\mathcal{L}_{\text{interp}} = -\frac{1}{|y|} \log p(y|z_{\text{interp}}) \tag{1}$$

where $z_{\text{interp}} = u z_{\text{S2S}} + (1-u) z_{\text{AE}}$ and $u \sim U(0,1)$ is a uniformly distributed random variable. $|y|$ is the number of words in y. Note that it is this regularization term that effectively prevents significantly different responses from aligning in the same direction, essentially scattering them over different directions. In order for this interpolation loss to work, we share the same decoder for both AE and S2S models as in (Luan et al., 2017). The decoder consists of stacked GRU cells followed by a softmax layer. It is worth mentioning that z_{interp} is not just randomly drawn from a single line but from a richer probabilistic region as both z_{interp} and z_{S2S} are stochastic due to the random component ϵ.

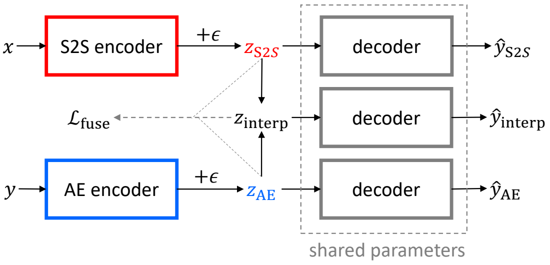


Figure 2: SPACEFUSION model architecture.

Now, we want vectors from both the AE and S2S models to be distributed in a homogeneous manner scattered over the entire space while keeping the distance between z_{S2S} and z_{AE} as small as possible for any (context-response) pair in the training data. This objective is represented in the following regularization term:

$$\begin{aligned}\mathcal{L}_{\text{fuse}} = &\sum_{i \in \text{batch}} \frac{d(z_{\text{S2S}}(x_i), z_{\text{AE}}(y_i))}{n} \\ &- \sum_{i,j \in \text{batch}, i \neq j} \frac{d(z_{\text{S2S}}(x_i), z_{\text{S2S}}(x_j))}{n^2 - n} \\ &- \sum_{i,j \in \text{batch}, i \neq j} \frac{d(z_{\text{AE}}(y_i), z_{\text{AE}}(y_j))}{n^2 - n}\end{aligned} \tag{2}$$

where n is the batch size and $d(a, b)$ is the root mean square of the difference between a and b. For each batch, we basically disperse vectors obtained by the same model and pull the predicted response vectors to the corresponding target response vectors. In practice, we found that the performance is better if the Euclidean distance is clipped to a prescribed maximum value.[3]

Finally, with weight parameters α and β, the

---

[3] This value is set as 0.3 for the present experiments

loss function is defined as:

$$\begin{aligned}\mathcal{L} = &- \frac{1}{|y|}\log p(y|z_{\text{S2S}}) \\ &- \frac{1}{|y|}\log p(y|z_{\text{AE}}) \\ &+ \alpha\mathcal{L}_{\text{interp}} + \beta\mathcal{L}_{\text{fuse}}\end{aligned} \tag{3}$$

As $\mathcal{L}_{\text{interp}}$ and $\mathcal{L}_{\text{fuse}}$ encourage the path between z_{S2S} and z_{AE} to be smooth and short while scattering vectors over the entire space, they effectively fuse the z_{S2S} latent space and the z_{AE} latent space. Accordingly we refer this approach as SPACEFUSION with path regularization.

3.3 Training

In contrast to previous multi-task conversation model (Luan et al., 2017), where S2S and AE are trained alternately, our approach trains S2S and AE at the same time by minimizing the loss function of Equation 3.

3.4 Inference

Like Zhao et al. (2017); Bowman et al. (2016), for a given context, we sample different latent vectors to obtain multiple hypotheses. This is done by adding a random vector r that is uniformly sampled from a hypersphere of radius $|r|$ to the prediction $z_{\text{S2S}}(x)$.

$$z(x, r) = z_{\text{S2S}}(x) + r \tag{4}$$

where $|r|$ is tuned on the validation set to optimize the trade-off between relevance and diversity. $z(x, r)$ is then fed to the decoder as the initial state of GRU cells. We then generate responses using greedy decoding.[4]

4 Experiment Setup

4.1 Datasets

We used the following datasets. Some of their key features are presented in Table 1.

Switchboard: We use the version offered by Zhao et al. (2017), which is an extension of the original version by Godfrey and Holliman (1997). Zhao et al. (2017) collected multiple references for the test set using information retrieval (IR) techniques followed by human filtering, and randomly split the data into 2316/60/62 conversations for train/validate/test, respectively. Each conversation has multiple turns and thus multiple (x, y) pairs, as listed in Table 1. As our approach does not utilize extra information except conversation history, we removed the meta data (e.g. gender, age, prompt) from this dataset.

	Switchboard	Reddit
train (x, y) samples	0.2M	7.3M
test (x, y) samples	5418	5000
ref. source	IR+filtering	natural
ref. availability	test only	train/vali/test
ref. per context	7.7	24.1

Table 1: Key features of the datasets.

Reddit: As the Switchboard dataset is relatively small and multiple references are synthetically constructed, we have developed another multi-reference dataset by extracting posts and comments on Reddit.com during 2011 collected by a third party.[5] As each Reddit post and comment may have multiple comments, it is a natural source of multi-reference responses. We further filtered the data based on the number of replies to obtain the final conversation dataset in which each context has at least 10 different responses, and on average the number of responses is 24.1 for a given context. The size is significantly larger than Switchboard, as listed in Table 1. The conversations are randomly shuffled before being split into train/valid/test subsets.

4.2 Model setup

Both encoders and the shared decoder consist of two GRU cells, each with 128 hidden units. The variance of the noise layer in each decoder is $\sigma^2 = 0.1^2$. The word embedding dimension is 128. The weight parameters (see Equation 3) are set as $\alpha = 1$ and $\beta = 30$. For both datasets, the inference radius $|r|$ (see Equation 4) is set to 1.5 which optimizes F1 score on the validation set. All models are trained using the Adam method (Kingma and Ba, 2014) with a learning rate of 0.001 on both datasets until convergence (around 4 epochs for Reddit and 10 epochs for Switchboard).

4.3 Automatic evaluation

For a given context x, we have N_r reference responses and generate the same number of hypothe-

---

[4] Although we use greedy decoding in this work, other decoding techniques, such as beam search, can be applied.

[5] `http://files.pushshift.io/reddit/comments/`

ses.[6] We define the following metrics based on 4-gram BLEU (Papineni et al., 2002), as suggested by Zhao et al. (2017).

$$\text{Precision} = \frac{1}{N_r}\sum_{i=1}^{N_r}\max_{j\in[1,N_r]}\text{BLEU}(r_j, h_i)$$

$$\text{Recall} = \frac{1}{N_r}\sum_{j=1}^{N_r}\max_{i\in[1,N_r]}\text{BLEU}(r_j, h_i)$$

$$\text{F1} = 2\,\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$$

We use Precision as an approximate surrogate metric for relevance and Recall for diversity. It should be noted that recall is not equivalent to other diversity metrics, e.g., distinct (Li et al., 2016a) and entropy (Zhang et al., 2018), which only depend on hypotheses. One potential issue of these metrics is that even randomly generated responses may yield a high diversity score. F1 is the harmonic average of these two and is used to measure the overall response quality.

4.4 Human evaluation

We conduct a human evaluation using crowdworkers. For each hypothesis, given its context, we ask three annotators to individually measure the quality, on a scale of 1 to 5, in terms of two aspects: relevance and interest. Interestingness is treated as an estimation of the diversity, as these two are often correlated. The hypotheses from all systems are shuffled before being provided to annotators. System names are invisible to the annotators.

4.5 Baselines

We compare the proposed model with the following baseline models:

S2S+Sampling: We consider a vanilla version of S2S model. The dimensions are similar to our model: both encoder and decoder consist of two stacked GRU cells with 128 hidden units, and the word embedding size is 128. As in the baseline in Zhao et al. (2017), we applied softmax sampling at inference time to generate multiple hypotheses.

CVAE+BOW: For the CVAE conversation model, we use the original implementation and hyperparameters of Zhao et al. (2017) with the bag-of-words (BOW) loss. The number of trainable model parameters is 15.4M, which is much larger than our model (3.2M).

MTask: Since our approach utilizes a multi-task learning scheme, we also compare it against a vanilla multi-task learning model, MTask, similar to (Luan et al., 2017), to illustrate the effect of space fusion. The model architecture and hyperparameters are identical to the proposed model except that the loss function is $\mathcal{L} = -\log p(y|z_{\text{S2S}}) - \log p(y|z_{\text{AE}})$.

5 Results and Analysis

5.1 In-depth analysis of latent space

In this section, we undertake an in-depth analysis to verify whether the latent space induced by our method manifests desirable properties, namely: 1) disentangled space structure between relevance and diversity, 2) homogeneous space distribution in which semantics changes smoothly without holes. We first provide a qualitative investigation based on real examples. Then, we present a set of corpus-level quantitative analyses focused on geometric properties.

5.1.1 Qualitative examples

In Table 2, we investigate three different directions from the context "`Anyone want to start this game?`" , which is a real example taken from Reddit. The three different directions correspond to clearly different semantics: "`No I don't`", "`when?`" and "`Yes I do.`" If we generate a response with the vector predicted by the S2S model ($u = 0$), our model outputs "`I would love to play this game`" which is highly relevant to the context. Now as we move along each direction, we can see our model gradually transforms the response toward the corresponding responses of each direction. For instance, towards "`No I don't`", our model gradually transforms the response to "`I am not interested in the game`" ($u = 0.18$) and then "`I am not interested.`" ($u = 0.21$). In contrary, towards "`Yes I do`", the response transforms to "`I would love to play it.`" ($u = 0.15$). Besides the positive or negative directions, the same transition applies to other directions such as "`When?`". This example clearly shows that there is a rough correspondence

---

[6] We set the number of hypotheses equal to the number of references to encourage precision and recall have comparable impact on F1

<table>
<tr><th colspan="6">context x: Anyone want to start this game?
response at $u = 0$: I would love to play this game.</th></tr>
<tr><th>u</th><th>towards “No I don’t.”</th><th>u</th><th>towards “when?”</th><th>u</th><th>towards “Yes I do.”</th></tr>
<tr><td>0.18</td><td>I am not interested in the game.</td><td>0.15</td><td>I’d be interested in the game</td><td>0.15</td><td>I’d love to play it.</td></tr>
<tr><td>0.21</td><td>I am not interested.</td><td>0.31</td><td>When is it?</td><td>0.27</td><td>Yes I do.</td></tr>
<tr><td>0.30</td><td>No I don’t.</td><td>0.40</td><td>When will you?</td><td></td><td></td></tr>
<tr><td></td><td></td><td>1.00</td><td>When?</td><td></td><td></td></tr>
</table>

Table 2: Semantic interpolation along different directions y. Results decoded from z_{interp} See Fig. 1 for a visualization.

.

<table>
<tr><th colspan="4">context x: Anyone want to start this game?
towards one possible target y: Yes I do.</th></tr>
<tr><th>u</th><th>with regularization</th><th>u</th><th>without regularization</th></tr>
<tr><td>0.00</td><td>I would love to play this game.</td><td>0.00</td><td>I would have to play with the game.</td></tr>
<tr><td>0.15</td><td>I would love to play it.</td><td>0.29</td><td>Dude, I know, but, or etc.</td></tr>
<tr><td>0.30</td><td>Yes I do</td><td>0.61</td><td>Op I was after though today</td></tr>
<tr><td></td><td></td><td>0.85</td><td>I’m single :(though</td></tr>
<tr><td></td><td></td><td>0.90</td><td>Yes I do.</td></tr>
</table>

Table 3: Semantic interpolation with and without regularization. Results decoded from z_{interp} .

between geometric properties and semantic properties in the latent space induced by our method as shown in Figure 1– the relevance of the response decreases as we move away from the predicted response vector and different directions are associated with semantically different responses.

5.1.2 Direction vs. diversity

In order to quantitatively verify the correspondence between *direction* and *diversity*, we visualize the distribution of cosine similarities among multiple references for each context for a set of 1000 random samples drawn from the test dataset. Specifically, for a context x_k and its associated reference responses $[y_{k,0}, y_{k,1}, \cdots]$, we compute the cosine similarity between $z_{\text{AE}}(y_{k,i}) - z_{\text{S2S}}(x_k)$ and $z_{\text{AE}}(y_{k,j}) - z_{\text{S2S}}(x_k)$. In Figure 3, we compare the distribution of our model with that of MTask, which does not employ our regularization terms. While our method yields a bell shaped curve with average cosine similarity being close to zero (0.38), the distribution of MTask is extremely skewed with average cosine similarity being close to 1 (0.95). This indicates that the directions of the reference responses are more evenly distributed in our latent space whereas everything is packed in a narrow band in the MTask’s space. This essentially makes the inference process simple and robust in that one can choose arbitrary directions to generate diverse responses.

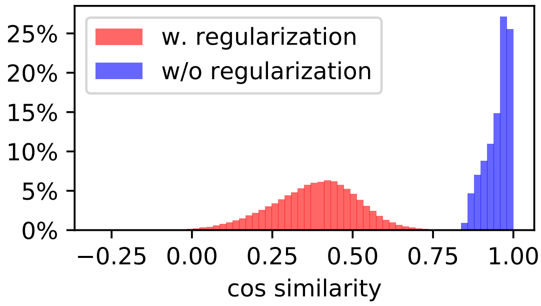


Figure 3: Distribution of the directions from a given context to its multiple responses, measured by the cosine similarity between $z_{\text{AE}}(y_{k,i}) - z_{\text{S2S}}(x_k)$ and $z_{\text{AE}}(y_{k,j}) - z_{\text{S2S}}(x_k)$. Histogram calculated based on 1000 x_k from Reddit test data and visualized with bin width of 0.02.

5.1.3 Distance vs. relevance

In order to quantitatively verify the correspondence between *distance* and *relevance*, we visualize the perplexity of reference responses along the path from the associated z_{S2S} ($u = 0$) to the z_{AE} ($u = 1$) corresponding to the predicted response. In Figure 4, we compare our model with MTask, which as already noted, does not employ our regularization terms. While our model shows a gradual increase in perplexity, there is a huge bump for MTask’s line. This clearly indicates that there is a rough correspondence between distance and relevance in our latent space whereas even a slight change can lead to an irrelevant response in the MTask’s space.

We further illustrate the smooth change in relevance according to distance for a specific example in Table 3. Given the context “`Anyone want to start this game?`”, our model

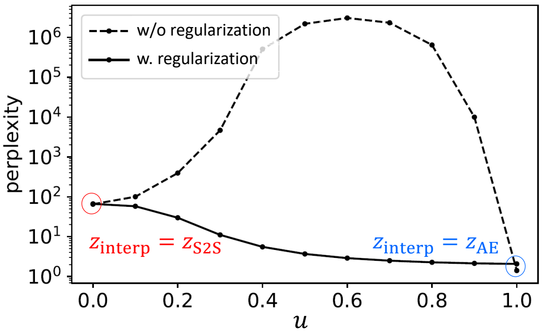


Figure 4: Perplexity of z_{interp} on the Reddit test dataset as a function of u for simple multi-task model (without regularization, dashed line) and SPACEFUSION (with regularization, solid line).

is able to transition from the predicted response “`I would love to play this game`” to a one of reference responses “`Yes I do`”. The relevance smoothly descreases, generating intermediate responses such as “`I would love to play it.`” In contrary, the MTask model tends to produce irrelevant or ungrammatical responses as it moves away from the predicted response.

5.1.4 Homogeneity and Convexity

Other desirable properties, with which we want to equip our latent space are *homogeneity* and *convexity*. If the space is not homogeneous, we have to sample differently depending on the regional traits. If the space is not convex, we have to worry about running into the holes that are not properly associated with valid semantic meanings. In order to verify homogeneity and convexity, we visualize our latent space in a 2D space produced by the multidimensional scaling (MDS) algorithm (Borg and Groenen, 2003), which approximately preserves pairwise distance. For comparison, we also provide a visualization for MTask. As shown in Figure 5, our latent space offers great homogeneity and convexity regardless of which model is used to produce a dot (i.e. z_{S2S} or z_{AE}). In contrary, MTask’s latent space forms two separate clusters for z_{S2S} and z_{AE} with a large gap in-between where no training samples were mapped to.

5.2 Automatic evaluation

We let each system generate 100 hypotheses $\{h_j\}$ for each context x_i in the test dataset. Assuming x_i has $N_{r,i}$ references, we pick the top $N_{r,i}$ distinct hypotheses ranked by $\log p(h_j|x_i) + \lambda|h_j|$. Similar to (Li et al., 2016a; Wu et al., 2016), we takes $|h_j|$ into consideration, as BLEU is sensitive to length.

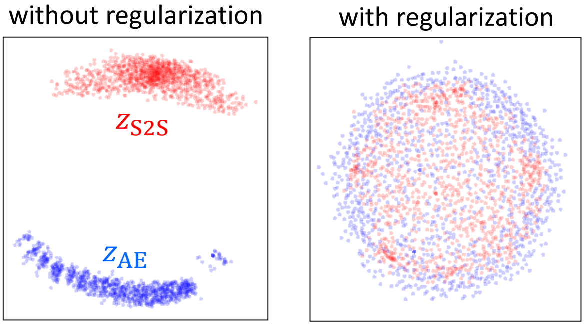


Figure 5: MDS visualization of the two latent spaces: z_{s2s} (red dots) and z_{AE} (blue dots) of 1000 randomly picked (x, y) pairs from the Reddit test dataset. Left: multi-task model (without regularization); right: SPACEFUSION (with regularization).

For fair comparison, λ is tuned such that the average hypothesis length becomes roughly the same for all systems and approaches the average length of the references.[7]

The automatic evaluation results are reported in Table 4. On both datasets, the proposed system consistently outperforms the baselines by a large margin in Precision, Recall, and F1.

Examples of system outputs and human references can be found in Table 5 and Table 6 for Reddit and Switchboard, respectively. As shown in the examples, CVAE+BOW and other baseline models may generate diverse but not-so-relevant responses.

dataset	model	Precision	Recall	F1
Switchboard	SPACEFUSION	**1.22**	**0.66**	**0.86**
	CVAE+BOW	0.76	0.57	0.65
	MTask	0.75	0.43	0.54
	S2S+Sampling	0.57	0.48	0.52
Reddit	SPACEFUSION	**0.40**	**0.26**	**0.31**
	CVAE+BOW	0.16	0.18	0.17
	MTask	0.31	0.18	0.23
	S2S+Sampling	0.10	0.11	0.11

Table 4: Performance of each model on automatic measures. The highest score in each row is in **bold** for each dataset. Note that our BLEU scores are normalized to $[0, 100]$.

5.3 Human evaluation

We randomly sampled 500 contexts from the Reddit test dataset and picked the top 1 hypothesis generated for each context ranked by $\log p(h_j|x_i) + \lambda|h_j|$. As in the automatic evaluation, we tuned λ such that all systems have roughly

---

[7] Approximately 10 words/tokens for Switchboard and 12 for Reddit

context	Everything about this movie is awesome!
SpaceFusion	○ I love this movie. ○ It's so awesome!!! I have no idea how to watch this movie. I can't wait for the trailer. ○ I don't think i'm a fan of the movie. ○ I would love to see this. ○ I want to watch this movie.
CVAE +BOW	○ Smartphones of the best games!. ○ I'm in the same boat! I feel the same way about this ○ I don't know why but can't tell if the New York will be the only one. ○ Caves would never say yes, but I'd love to know. ○ I know where this movie is.
MTask	○ This is so funny. ○ I think I saw this. ○ I don't know how many people do you. ○ I would. ○ I remember watching was was disappointed.
S2S+ sampling	○ Whoa man, this is amazing!!! ○ Man, I love Tom Cruise and I love the show. ○ Is this a bill on the right? ○ More like samurai jack. ○ I weep for the hivemind.
human	○ I love this movie. ○ Watched this yesterday! so amazing. ○ My favorite Muppet movie by far. ○ Seriously. Good times. Great oldies. ○ Now you have got the song into my head.

Table 5: Example system outputs and human reference on Reddit dataset

the same average hypothesis length. We also randomly select one reference for each context and compare them with the systems (labeled ”human” in Table 7)

As illustrated in Table 7, the proposed model outperforms all systems except human, consistent with our automatic evaluation results.

	relevance	interest	average
SpaceFusion	**2.72**	**2.53**	**2.63**
CVAE+BOW	2.51	2.37	2.44
Multi-Task	2.34	2.14	2.24
S2S+Sampling	2.58	2.43	2.50
human	3.59	3.41	3.50

Table 7: Performance of each model on human evaluation. The highest score, except human, in each row is in **bold**.

6 Conclusion

We propose a SpaceFusion model to jointly optimize diversity and relevance that leverages novel regularization terms to essentially fuse the latent space of a S2S model with that of an autoencoder model. This fused latent space exhibits desirable properties such as smooth semantic interpolation between two points. The distance and direction from the predicted response vector roughly match relevance and diversity, respectively. These properties also enable intuitive visualization of the latent space. Both automatic and human evaluation results demonstrate that the proposed approach brings significant improvement compared to strong baselines in terms of both diversity and relevance. In future work, we will provide theoretical justification of the effectiveness of the proposed regularization terms. We expect that this technique will find application as an efficient ”mixing board” for conversation that draws on multiple sources of information.

context	**A**: Are they doing a lot of recycling out in ***? **B**: Well at my workplace they are. We have places for aluminum cans and we have everybody been issued a separate trash can for recyclable paper. Let's see them get that word straight. **A**: Uh-huh **B**: As far as the community goes, it's pretty much voluntary. Nothing's been done. ***. We have just private garbage services.
SpaceFusion	○ They're not recycling as a matter of fact. ○ And it's a waste of time ○ You know the paper is a big recycling program. ○ But they just put it in the trash ○ Or if you have a recycling bag or anything like that.
CVAE +BOW	○ Um - hum. ○ Oh that would be nice. ○ Is that right? ○ It's a it's a luxury ○ Well that's interesting.
MTask	○ Um - hum. ○ That's right. ○ I don't know how much money you're going to do it. ○ That's right. ○ Yeah
S2S+ sampling	○ That's great. ○ What do you feel about this uh - huh. ○ It's just kind of a glass of town and it's even irritating to me. I don't know ○ Somebody and it's so much better. ○ It now and it's hard to go.
human	○ Yeah it is to some degree inconvenient i'd have to say that

Table 6: Example system outputs and human reference on Switchboard dataset. Controversial or offensive words are replaced by ***.

References

Ingwer Borg and P Groenen. 2003. Modern multidimensional scaling: theory and applications. *Journal of Educational Measurement*, 40(3):277–280.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*.

Michel Galley, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan. 2019. Grounded response generation task at dstc7.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932*.

John J Godfrey and Edward Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*, 926:927.

Xiaodong Gu, Kyunghyun Cho, Jungwoo Ha, and Sunghun Kim. 2018. DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. *arXiv preprint arXiv:1805.12352*.

Bernd Huber, Daniel McDuff, Chris Brockett, Michel Galley, and Bill Dolan. 2018. Emotional dialogue generation using image-grounded language models. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 277. ACM.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Yi Luan, Chris Brockett, Bill Dolan, Jianfeng Gao, and Michel Galley. 2017. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 605–614.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Koichiro Yoshino, Chiori Hori, Julien Perez, Luis Fernando D'Haro, Lazaros Polymenakos, Chulaka Gunasekara, Walter S Lasecki, Jonathan K Kummerfeld, Michel Galley, Chris Brockett, et al. 2019. Dialog system technology challenge 7. *arXiv preprint arXiv:1901.03461*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Advances in Neural Information Processing Systems*, pages 1813–1823.

Yizhe Zhang, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2018. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*.

Disentangling Language and Knowledge in Task-Oriented Dialogs

Dinesh Raghu*[1,2], Nikhil Gupta[1] and Mausam[1]

[1] IIT Delhi, New Delhi, India
[2] IBM Research, New Delhi, India
diraghu1@in.ibm.com, nikhilgupta1997@gmail.com, mausam@cse.iitd.ac.in

Abstract

The Knowledge Base (KB) used for real-world applications, such as booking a movie or restaurant reservation, keeps changing over time. End-to-end neural networks trained for these task-oriented dialogs are expected to be immune to any changes in the KB. However, existing approaches breakdown when asked to handle such changes. We propose an encoder-decoder architecture (BoSsNet) with a novel Bag-of-Sequences (BoSs) memory, which facilitates the disentangled learning of the response's language model and its knowledge incorporation. Consequently, the KB can be modified with new knowledge without a drop in interpretability. We find that BoSsNet outperforms state-of-the-art models, with considerable improvements (>10%) on bAbI OOV test sets and other human-human datasets. We also systematically modify existing datasets to measure disentanglement and show BoSsNet to be robust to KB modifications.

1 Introduction

Task-oriented dialog agents converse with a user with the goal of accomplishing a specific task and often interact with a knowledge-base (KB). For example, a restaurant reservation agent (Henderson et al., 2014) will be grounded to a KB that contains the names of restaurants, and their details.

In real-world applications, the KB information could change over time. For example, (1) a KB associated with a movie ticket booking system gets updated every week based on new film releases, and (2) a restaurant reservation agent, trained with the knowledge of eateries in one city, may be deployed in other cities with an entirely different range of establishments. In such situations, the system should have the ability to conform to new-found knowledge unseen during its training. Ideally, the training algorithm must learn to disentangle the language model from the knowledge interface model. This separation will enable the system to generalize to KB modifications, without a loss in performance.

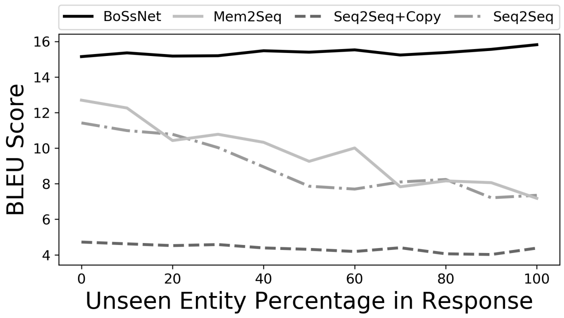


Figure 1: Performance of various task-oriented dialog systems on the CamRest dataset as the percentage of unseen information in the KB changes.

Moreover, for achieving good progress towards the user's task, the agent must also retain the ability to draw inferences based on past utterances and the KB. Notably, we find that existing approaches either achieve this disentanglement or effective progress towards the task, but not both.

For instance, Mem2Seq (Madotto et al., 2018) exhibits satisfactory performance when tested on the training KB. It represents the dialog history and the KB knowledge as a *bag of words* in a flat memory arrangement. This enables Mem2Seq to revisit each word several times, as needed, obtaining good performance. But at the same time, flat memory prevents it from capturing any surrounding context – this deteriorates its performance rapidly when the amount of new unseen information in the KB increases, as shown in Figure 1. On the other hand, the performance of copy augmented sequence-to-sequence network (Seq2Seq+Copy) (Eric and Manning, 2017), is robust to changes in the KB, but fails to achieve acceptable task-oriented performance. It captures context by representing the entire dialog history as one continuous *sequence*. However, it can be difficult for a sequence encoder to reason

---

[*] D. Raghu is an employee at IBM Research. This work was carried out as part of PhD research at IIT Delhi.

Proceedings of NAACL-HLT 2019, pages 1239–1255
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

over long dialogs found in real-world datasets and its ability to learn the task gets hampered.

We propose BoSsNet, a novel network that effectively disentangles the language and knowledge models, and also achieves state-of-the-art performance on three existing datasets.

To achieve this, BoSsNet makes two design choices. First, it encodes the conversational input as a *bag of sequences* (BoSs) memory, in which the input representation is built at two levels of abstraction. The *higher level* flat memory encodes the KB tuples and utterances to facilitate effective inferencing over them. The *lower level* encoding of each individual utterance and tuple is constructed via a sequence encoder (Bi-GRU). This enables the model to maintain the sequential context surrounding each token, aiding in better interpretation of unseen tokens at test time. Second, we augment the standard cross-entropy loss used in dialog systems with an additional loss term to encourage the model to only copy KB tokens in a response, instead of generating them via the language model. This combination of sequence encoding and additional loss (along with dropout) helps in effective disentangling between language and knowledge.

We perform evaluations over three datasets – bAbI (Bordes and Weston, 2017), CamRest (Wen et al., 2016), and Stanford Multi-Domain Dataset (Eric et al., 2017). Of these, the last two are real-world datasets. We find that BoSsNet is competitive or significantly better on standard metrics in all datasets as compared to state-of-the-art baselines. We also introduce a *knowledge adaptability* (KA) evaluation, in which we systematically increase the percentage of previously unseen entities in the KB. We find that BoSsNet is highly robust across all percentage levels. Finally, we also report a human-based evaluation and find that BoSsNet responses are frequently rated higher than other baselines.

Overall, our contributions are:

1. We propose BoSsNet, a novel architecture to disentangle the language model from knowledge incorporation in task-oriented dialogs.

2. We introduce a *knowledge adaptability* evaluation to measure the ability of dialog systems to scale performance to unseen KB entities.

3. Our experiments show that BoSsNet is competitive or significantly better, measured via standard metrics, than the existing baselines on three datasets.

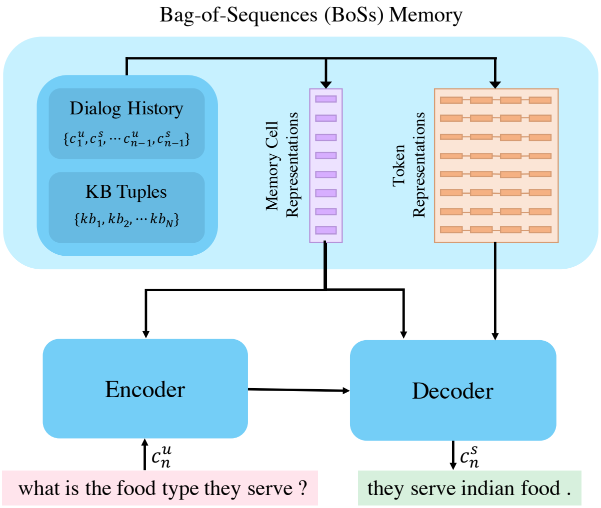


Figure 2: The dialog history and KB tuples stored in the memory have memory cell representations and token representations. The encoder understands the last user utterance using only the memory cell representations. The decoder generates the next response using both representations.

We release our code and *knowledge adaptability* (KA) test sets for further use by the research community.[1]

2 The BoSsNet Architecture

The proposed Bag-of-Sequences Memory Network has an encoder-decoder architecture that takes as input (1) dialog history, which includes a sequence of previous user utterances $\{c_1^u, \ldots, c_n^u\}$ and system responses $\{c_1^s, \ldots, c_{n-1}^s\}$, and (2) KB tuples $\{kb_1, \ldots, kb_N\}$. The network then generates the next system response $c_n^s = \langle y_1 y_2 \ldots y_T \rangle$ word-by-word. The simplified architecture of BoSsNet is shown in Figure 2.

In this section, we first describe the BoSs memory which contains the dialog history and KB tuples, followed by how the memory is consumed by the encoder and the decoder. We finally define the loss function, which, along with dropout, enables disentangled learning of language and knowledge.

2.1 Bag-of-Sequences Memory

The memory M contains the dialog history $\{c_1^u, c_1^s, \ldots, c_{n-1}^u, c_{n-1}^s\}$ and the KB tuples $\{kb_1, \ldots, kb_N\}$. Each utterance in the dialog history and each KB tuple is placed in a memory cell. As utterances and tuples are inherently a sequence, we represent each memory cell m_i as

---

[1] https://github.com/dair-iitd/BossNet

an ordered sequence of tokens $\langle w_i^1 w_i^2 \dots w_i^{|m_i|} \rangle$. For an utterance, the word tokens are followed by a temporal indicator and a speaker indicator {\$u, \$s}. For example, {`good, morning, #1, $s`} indicates this was the first utterance by the system. For a KB tuple, the tokens are sequenced as {*subject, predicate, object*} followed by temporal indicator and a kb indicator (`$db`).

Token representation is generated using a bidirectional GRU. Let the outputs of the forward and backward GRUs for the token w_i^j be denoted as $\overrightarrow{h_i^j}$ and $\overleftarrow{h_i^j}$ respectively. Then the token representation $\phi(w_i^j)$ is given by Eq. 1. Memory cell representation $\psi(m_i)$ is computed by concatenating the forward GRU output of its last token and the backward GRU output of its first token as in Eq. 2.

$$\phi(w_i^j) = [\overrightarrow{h_i^j}; \overleftarrow{h_i^j}] \tag{1}$$

$$\psi(m_i) = [\overrightarrow{h_i^{|m_i|}}; \overleftarrow{h_i^1}] \tag{2}$$

2.2 The BoSsNet Encoder

The encoder used in BoSsNet is similar to the multi-hop attention encoder with layer-wise weights proposed by Sukhbaatar et al. (2015). The encoder in Sukhbaatar et al. (2015) uses two different embedding matrices, whereas we use just one to reduce the number of parameters. The encoder considers the last user utterance as the query $q = \psi(c_n^u)$ and computes the reduced representation q_r using the memory M as follows:

$$\begin{aligned} p_i &= \text{softmax}(q^T \psi(m_i)) && (3)\\ o &= W_r \sum\nolimits_i p_i \psi(m_i) && (4)\\ q_r &= o + W_o q && (5) \end{aligned}$$

where $W_r, W_o \in \mathbb{R}^{d \times d}$ are learnable parameters. The hop step can be re-iterated, by assigning the output of the previous hop as the new input query, i.e., setting $q = q_r$. The output of the encoder after K hops, q_r^k, is assigned as the initial state of the BoSsNet decoder.

2.3 The BoSsNet Decoder

BoSsNet models a copy-augmented sequence decoder, which generates the response one word at a time. At any decode time step t, the decoder can either *generate* a word from the decode vocabulary or *copy* a word from the memory. Consequently, the decoder computes: (1) generate distribution $P_g(y_t)$ over the decode vocabulary, and (2) copy distribution $P_c(y_t)$ over words in the memory.

The generate distribution is computed using a standard sequence decoder (Sutskever et al., 2014) by attending (Luong et al., 2015) over the memory cell representations ψ. The copy distribution is generated by using a *two-level attention*. Given the decoder state s_t, it first computes attention α_t over the memory cells. Then it computes attention over the tokens in each memory cell m_i. Finally it multiplies both these attentions to compute $P_c(y_t)$ as follows:

$$\alpha_i^t = \text{softmax}(s_t \psi(m_i)) \tag{6}$$

$$e_{ij}^t = s_t \phi(w_i^j) \tag{7}$$

$$\beta_{ij}^t = \alpha_i^t * \frac{\exp(e_{ij}^t)}{\sum_k \exp(e_{ik}^t)} \tag{8}$$

$$P_c(y_t = w) = \sum_{ij: w_i^j = w} \beta_{ij}^t \tag{9}$$

The copy and generate distributions are combined using a soft gate $g_s^t \in [0, 1]$ as in See et al. (2017). g_s^t is a function of the decoder state at time t and the word decoded in the previous time step.

2.4 Loss

The decoder is trained using cross-entropy loss. The loss per response is defined as:

$$\mathcal{L}_{ce} = -\sum_{t=1}^{T} \log\Big(g_s^t P_g(y_t) + (1 - g_s^t) P_c(y_t)\Big) \tag{10}$$

where T is the number of words in the sequence to be generated and y_t is the word to be generated at time step t. The decision to generate or copy is learnt implicitly by the network. However, to attain perfect disentanglement, the KB words should be copied, while the language should be generated. In other words, any word in the response that is present in the BoSs KB memory should have a low g_s. To obtain this behavior, we define a disentangle label D_l for each word in the response. This label is set to 1 if the word is present in the BoSs KB memory and 0 otherwise. We define a disentangle loss as follows:

$$\mathcal{L}_d = -\sum_{t=1}^{T} g_s^t \log D_l^t + (1 - g_s^t) \log(1 - D_l^t) \tag{11}$$

We randomly drop some words with disentangle label set to 1. This *Disentangle Label Dropout (DLD)* works in tandem with the disentangle loss and BoSs memory – it encourages the model to copy KB words whenever possible, based on their surrounding words. The overall loss is given as:

$$\mathcal{L} = \mathcal{L}_{ce} + \gamma \mathcal{L}_d \tag{12}$$

The relative weight of $\mathcal{L}_d$ in the overall loss is controlled using a hyper-parameter (γ). The dropout rate is also a hyper-parameter.

3 Experimental Setup

We perform experiments on three task-oriented dialog datasets: bAbI Dialog (Bordes and Weston, 2017), CamRest (Wen et al., 2016), and Stanford Multi-Domain Dataset (Eric et al., 2017).

bAbI Dialog consists of synthetically generated dialogs with the goal of restaurant reservation. The dataset consists of five different tasks, all grounded to a KB. This KB is split into two mutually exclusive halves. One half is used to generate the train, validation, and test sets, while the other half is used to create a second test set called the OOV test set.

CamRest is a human-human dialog dataset, collected using the Wiz-of-Oz framework, also aimed at restaurant reservation. It is typically used to evaluate traditional slot filling systems. In order to make it suitable for end-to-end learning, we stripped the handcrafted state representations and annotations in each dialog, and divided the 676 available dialogs into train, validation, and test sets (406, 135, and 135 dialogs, respectively).

Stanford Multi-Domain Dataset (SMD) is another human-human dialog dataset collected using the Wiz-of-Oz framework. Each conversation is between a driver and an in-car assistant. The other datasets consist of dialogs from just one domain (restaurant reservation), whereas SMD consists of dialogs from multiple domains (calendar scheduling, weather information retrieval, and navigation).

3.1 Knowledge Adaptability (KA) Test Sets

Each bAbI dialog task has an additional OOV test set, which helps to evaluate a model's robustness to change in information in the KB. A model that perfectly disentangles language and knowledge should have no drop in accuracy on the OOV test set when compared to the non-OOV test set. To measure the degree of disentanglement in a model, we generated 10 additional test sets for each real-world corpus by varying the percentage (in multiples of 10) of unseen entities in the KB. We systematically picked random KB entities and replaced all their occurrences in the dialog with new entity names. We will refer to these generated dialogs as the *Knowledge Adaptability* (KA) test sets.

3.2 Baselines

We compare BoSsNet against several existing end-to-end task-oriented dialog systems. These include retrieval models, such as the query reduction network (QRN) (Seo et al., 2017), memory network (MN) (Bordes and Weston, 2017), and gated memory network (GMN) (Liu and Perez, 2017). We also compare against generative models such as a sequence-to-sequence model (Seq2Seq), a copy augmented Seq2Seq (Seq2Seq+Copy) (Gulcehre et al., 2016), and Mem2Seq (Madotto et al., 2018).[2] For fairness across models, we do not compare against key-value retrieval networks (Eric et al., 2017) as they simplify the dataset by canonicalizing all KB words in dialogs.

We noticed that the reported results in the Mem2Seq paper are not directly comparable, as they pre-processed[3] training data in SMD and bAbI datasets. For fair comparisons, we re-run Mem2Seq on the original training datasets. For completeness we mention their reported results (with pre-processing) as Mem2Seq*.

3.3 Evaluation Metrics

We evaluate BoSsNet and other models based on their ability to generate valid responses. The per-response accuracy (Bordes and Weston, 2017) is the percentage of generated responses that exactly match their respective gold response. The per-dialog accuracy is the percentage of dialogs with all correctly generated responses. These accuracy metrics are a good measure for evaluating datasets with boilerplate responses such as bAbI.

To quantify performance on other datasets, we use BLEU (Papineni et al., 2002) and Entity F1 (Eric and Manning, 2017) scores. BLEU measures the overlap of n-grams between the generated response and its gold response and has become a popular measure to compare task-oriented dialog systems. Entity F1 is computed by micro-F1 over KB entities in the entire set of gold responses.

---

[2] We thank the authors for releasing a working code at `https://github.com/HLTCHKUST/Mem2Seq`

[3] Mem2Seq used the following pre-processing on the data: 1) The subject (restaurant name) and object (rating) positions of the rating KB tuples in bAbI dialogs are flipped 2) An extra fact was added to the navigation tasks in SMD which included all the properties (distance, address, etc.) combined together as the subject and *poi* as the object. See Appendix.

3.4 Human Evaluation

We use two human evaluation experiments to compare (1) the *usefulness* of a generated response with respect to solving the given task, and (2) the *grammatical correctness* and *fluency* of the responses on a 0–3 scale. We obtain human annotations by creating Human Intelligence Tasks (HITs) on Amazon Mechanical Turk (AMT). For each test condition (percentage of unseen entities), we sampled 50 dialogs from Camrest and SMD each, and two AMT workers labeled each system response for both experiments, resulting in 200 labels per condition per dataset per system. We evaluate four systems in this study, leading to a total of 1600 labels per condition. The detailed setup is given in the Appendix.

3.5 Training

We train BoSsNet using an Adam optimizer (Kingma and Ba, 2014) and apply gradient clipping with a clip-value of 40. We identify hyper-parameters based on the evaluation of the held-out validation sets. We sample word embedding, hidden layer, and cell sizes from $\{64, 128, 256\}$ and learning rates from $\{10^{-3}, 5\times10^{-4}, 10^{-4}\}$. The hyper-parameter γ in the loss function is chosen between [0-1.5]. The Disentangle Label Dropout rate is sampled from $\{0.1, 0.2\}$. The number of hops for multi-hop attention in the encoder is sampled from $\{1, 3, 6\}$. The best hyper-parameter setting for each dataset is reported in the Appendix.

4 Experimental Results

Our experiments evaluate three research questions.

1. *Performance Study*: How well is BoSsNet able to perform the tasks of our three datasets as compared to the baseline models?
2. *Disentanglement Study*: How robust are the models in generalizing on the KA test sets?
3. *Ablation Study*: What is the performance gain from each novel feature in BoSsNet?

4.1 Performance Study

Table 1 reports the per-response and per-dialog (in parentheses) accuracies on the bAbI dialog tasks. The multi-hop retrieval-based models such as QRN, MN and GMN perform well on the non-OOV test sets for tasks 1, 2, and 5, but fail to exhibit similar performance on the corresponding OOV test sets. This result is expected as these models are trained to retrieve from a pre-defined set of responses. Their poor non-OOV performance on tasks 3 and 4 is attributed to an error in the bAbI dataset construction, due to which, the non-OOV and OOV test conditions are the same for these tasks (see Appendix).

A simple generative model (Seq2Seq) achieves accuracies comparable to the multi-hop retrieval models. Enabling it with the ability to copy from the context (Seq2Seq+Copy) shows a considerable increase in performance, especially on the OOV test sets (and non-OOV tests for tasks 3 and 4).

The strong performance of simple sequence encoders when compared with multi-hop encoders (in retrieval models) raises a question about the value of multi-hop inference. Mem2Seq answers this question, by obtaining improvements in several tasks, specifically on their OOV test sets. This clearly shows that multi-hop inference and the copy mechanism are essentials for task-oriented dialogs.

Despite gains from the Mem2Seq model, the performance difference between the non-OOV and OOV test sets remains large. BoSsNet succeeds to bridge this gap with its ability to better interpret unseen words, using their surrounding context. It obtains significant improvements on average of about 34% per-dialog accuracy and 10% per-response accuracy for the bAbI OOV test sets.

In Table 2, we report results on the real-world datasets. BoSsNet greatly outperforms other models in both Entity F1 metric and BLEU scores on CamRest. On SMD, BoSsNet achieves the best only in Entity F1. On further analysis of the generated responses we observe that BoSsNet responses often convey the necessary entity information from the KB. However, they consist of meaningful phrases with little lexical overlap with the gold response, reducing the BLEU scores. We investigate this further in our human evaluation.

Human Evaluation: We summarize the human evaluation results for real-world datasets in Table 3. BoSsNet shows the best performance on Camrest, and is judged useful 77 times out of 100. Also, it has the highest average grammatical correctness score of 2.28 (very close to Seq2Seq and Mem2Seq). BoSsNet performs on par with Mem2Seq and Seq2Seq in its ability to relay appropriate information to solve SMD dialog tasks, and has a slightly higher grammaticality score.

4.2 Disentanglement Study

We use our generated knowledge adaptability (KA) test sets to measure the robustness of BoSsNet

	Retrieval Models			**Generative Models**				
Task	**QRN**	**MN**	**GMN**	**Mem2Seq***	**Seq2Seq**	**Seq2Seq+Copy**	**Mem2Seq**	BoSsNet
T1	99.9 (-)	99.6 (99.6)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T2	99.5 (-)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T3	74.8 (-)	74.9 (2.0)	74.9 (0)	94.7 (62.1)	74.8 (0)	85.1 (19.0)	74.9 (0)	**95.2 (63.8)**
T4	57.2 (-)	59.5 (3.0)	57.2 (0)	100 (100)	57.2 (0)	100 (100)	100 (100)	100 (100)
T5	**99.6 (-)**	96.1 (49.4)	96.3 (52.5)	97.9 (69.6)	97.2 (64.4)	96 (49.1)	97.7(66.3)	97.3 (65.6)
T1-OOV	83.1 (-)	72.3 (0)	82.4 (0)	94.0 (62.2)	81.7 (0)	92.5 (54.7)	94.0 (62.2)	**100 (100)**
T2-OOV	78.9 (-)	78.9 (0)	78.9 (0)	86.5 (12.4)	78.9 (0)	83.2 (0)	86.5 (12.4)	**100 (100)**
T3-OOV	75.2 (-)	74.4 (0)	75.3 (0)	90.3 (38.7)	75.3 (0)	82.9 (0)	75.2 (0)	**95.7 (66.6)**
T4-OOV	56.9 (-)	57.6 (0)	57.0 (0)	100 (100)	57.0 (0)	100 (100)	100 (100)	100 (100)
T5-OOV	67.8 (-)	65.5 (0)	66.7 (0)	84.5 (2.3)	67.4 (0)	73.6 (0)	75.6 (0)	**91.7 (18.5)**

Table 1: Per-response and per-dialog accuracies (in brackets) on bAbI dialog tasks of BoSsNet and baselines
.

	CamRest		**SMD**	
	BLEU	**Ent. F1**	**BLEU**	**Ent. F1**
Mem2Seq*	12.7	39	12.6	33.4
Seq2Seq	11.4	40.6	8.7	34.9
Seq2Seq+Copy	4.7	32.2	3.23	16.9
Mem2Seq	12.7	39	**10.3**	31.8
BoSsNet	**15.2**	**43.1**	8.3	**35.9**

Table 2: Performance of BoSsNet and baselines on the CamRest and SMD datasets

	CamRest		**SMD**	
	Info	**Grammar**	**Info**	**Grammar**
Seq2Seq	46	2.24	35	2.38
Seq2Seq+Copy	27	1.1	21	1.04
Mem2Seq	51	2.2	**38**	2.0
BoSsNet	**77**	**2.28**	36	**2.5**

Table 3: AMT Evaluations on CamRest and SMD

KB (restaurant—address)	
r_bangkok_overpriced_thai_8—r_bangkok_overpriced_thai_8_addr *r_bangkok_overpriced_thai_7—r_bangkok_overpriced_thai_7_addr* *r_bangkok_overpriced_thai_4—r_bangkok_overpriced_thai_4_addr* *r_bangkok_overpriced_thai_2—r_bangkok_overpriced_thai_2_addr*	
usr-1	may i have a table in an *overpriced* price range for *nine* people with *thai* food in *bangkok* ?
sys-1	what do you think of : *r_bangkok_overpriced_thai_8* ?
usr-2	can you provide the address ?
Gold	here it is *r_bangkok_overpriced_thai_8_addr*
Seq2Seq+Copy	here it is *r_bangkok_overpriced_thai_4_addr*
Seq2Seq	here it is *r_london_moderate_spanish_6_addr*
Mem2Seq	here it is *r_bangkok_overpriced_thai_4_addr*
BoSsNet	here it is *r_bangkok_overpriced_thai_4_addr*

Table 4: Example from bAbI Task 5 KA test set with 100% OOV entities. Identifying the address of an unseen restaurant is challenging for all models.

and the other baselines to changes in the KB. We perform this experiment on 4 different tasks, namely bAbI tasks 1 and 5, CamRest, and SMD.

Figures 3 and 4 show the per-response accuracies of the two bAbI dialog tasks plotted against the percentage of unseen entities in KA sets. From Figure 3 we observe that BoSsNet remains immune to any variablity in the KB content, whereas the performance of Mem2Seq and Seq2Seq models drops drastically due to their inability to capture semantic representations of the injected KB entities. We see a similar trend in Figure 4, but here all the models show a drop in performance, with BoSsNet appearing the most steady. We explain this trend using the example dialog in Table 4. In the current dialog context, the system is required to provide the address of the selected restaurant, but since more than one restaurant in the KB is unseen, it becomes ambiguous for the network to identify the correct restaurant and infer its address. In the end, the system is forced to pick a random address – the probability of which being correct decreases as more restaurants become unseen.

The performance on the CamRest KA test sets is illustrated in Figures 1 and 5. BoSsNet has the best performance with even a slight increase in both BLEU and Entity F1 metrics as more OOV content is injected in the dialog, probably because it is clear that it needs to copy when processing unseen entities. Seq2Seq+Copy is unable to perform well in CamRest as the length of the input (dialog history + KB tuples) is long and the size of the training set is also small. We believe that Seq2Seq+Copy works best in an environment with an abundance of short dialog training data (e.g., bAbI task 1 in Figure 3).

SMD consists of dialogs with a large KB and a highly varying response pattern. This makes it very difficult to learn the language model – reflected in the low BLEU scores for all the systems. BoSsNet still provides the best F1 entity score due to

	CamRest		SMD	
	Info	**Grammar**	**Info**	**Grammar**
Seq2Seq	26	2.28	22	**2.44**
Seq2Seq+Copy	22	1.22	16	1.04
Mem2Seq	35	2.06	26	1.9
BoSsNet	**80**	**2.44**	**51**	2.28

Table 5: AMT Evaluations on CamRest and SMD (50% unseen) KA datasets

its ability to inference efficiently on the large KB (Figure 6). Mem2Seq shows the best BLEU score performance on the original test set, but its performance drop of 42.5%, from 10.3 at 0% unseen to 5.93 at 100% unseen, is a lot heavier than that of BoSsNet which only drops 7.6% – 8.27 at 0% unseen to 7.64 at 100% unseen.

Human Evaluation: We summarize the human evaluation results for real-world datasets on the 50% unseen KA test set in Table 5. BoSsNet again outperforms the baselines and is labeled *successful* twice more often than the next best model on both Camrest and SMD. Seq2Seq appears to produce better sentence structures on the SMD dataset, primarily because it does not attempt to learn inference on the KB, allowing it to solely focus on learning the language model better.

4.3 Ablation Study

We assess the value of each model element, by removing it from BoSsNet. Table 6 reports the per-response accuracy scores for various configurations of BoSsNet on bAbI dialog tasks. It also reports the BLEU and entity F1 metric of various configurations on CamRest.

Without BoSs Memory: This configuration uses the Bag-of-Bags (BoB) Memory rather than BoSs memory. The BoB memory is a simplified representation, similar to the one in the original Memory Networks. Here the token representation is the vector embedding of the token with no influence from the surrounding words and the memory cell representation is the sum of all its token embeddings. As a result, each word w representation is influenced equally by all words in a memory cell, irrespective of its distance from w. This makes capturing context in the immediate neighbourhood harder. Inability to capture the correct context prevents the configuartion from generalizing to OOV test sets.

Without Disentangled Loss: Disentangled Loss ($\mathcal{L}_d$) plays an important role in enforcing that KB words be copied and other language be generated. By removing this loss component, it achieves better BLEU score in CamRest, but with a drop in Entity F1. Without the disentangled loss, the model sometimes learns to generate KB words. This severely affects OOV performance. As described earlier, an error in bAbI dataset construction tasks 3 and 4 effectively injects the validation set with a lot of OOVs. This anomaly in conjunction with the dropout (DLD), helps the configuration in achieving an acceptable performance for those tasks.

Without Disentangled Label Dropout: BoSsNet learns to generate language and copy KB words. Without DLD, the model learns to memorize words to be copied rather than learning the context under which a word should be copied. Hence, the performance on OOV test sets is much inferior compared to the non-OOV setting.

Overall, we notice that combining all three model elements is necessary in obtaining the best performance across all tasks.

4.4 Qualitative Evaluation

We qualitatively compare the performance of BoSsNet with other baselines using examples.

Table 7, demonstrates the ability of BoSsNet to copy entities (restaurant name and address) in its response. The other baselines either generate unwanted or irrelevant entities in their response, or fail to copy altogether. BoSsNet also best captures the language model effectively with a slight paraphrasing of the gold response.

Table 8 contains only unseen entities. This example highlights the shortcomings of the Seq2Seq model as it ends up predicting a restaurant encountered during training. Mem2Seq copies a restaurant name without learning to sort the restaurants based on rating. BoSsNet, with its efficient memory addressing, is seen to be able to solve both issues.

5 Related Work

Compared to the traditional slot-filling based dialog (Williams and Young, 2007; Wen et al., 2017; Williams et al., 2017), end-to-end training methods (e.g., (Bordes and Weston, 2017), this work) do not require handcrafted state representations and their corresponding annotations in each dialog. Thus, they can easily be adapted to a new domain. We discuss end-to-end approaches along two verticals: 1) decoder: whether the response is retrieved or generated and 2) encoder: how the dialog history and KB tuples are encoded.

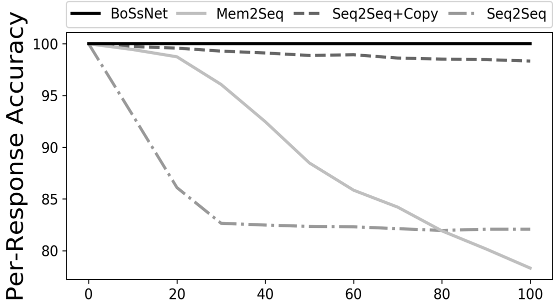


Figure 3: bAbI Task 1: Per-response accuracy comparison on KA sets

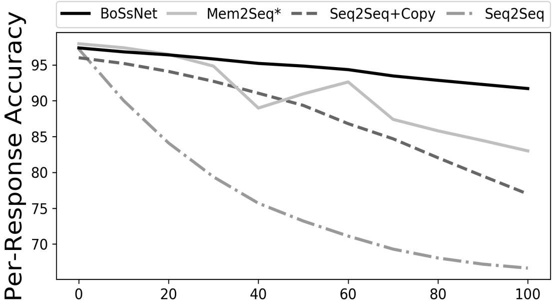


Figure 4: bAbI Task 5: Per-response accuracy comparison on KA sets

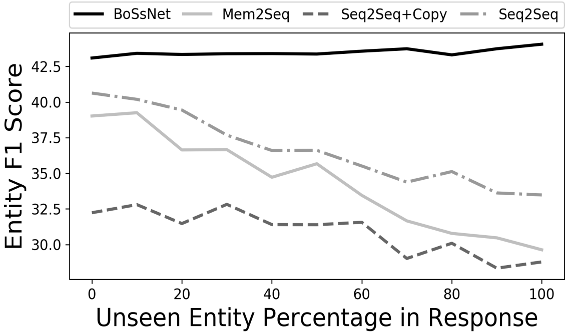


Figure 5: CamRest: Entity F1 comparison on KA sets

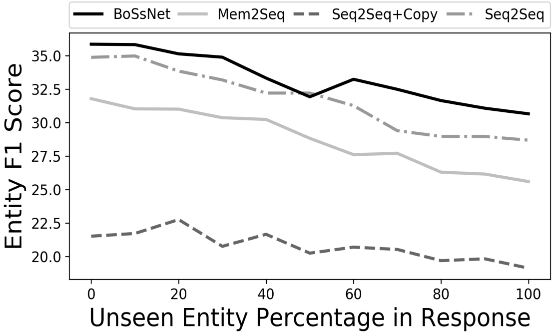


Figure 6: SMD: Entity F1 comparison on KA sets

	bAbI Dialog Tasks					**bAbI Dialog Tasks (OOV)**					**CamRest**	
	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	BLEU	Ent. F1
BoSsNet w/o BoSs Memory	100	100	74.9	57.2	95.6	93.5	78.9	74.9	57	81.4	10.13	29
BoSsNet w/o $\mathcal{L}_d$	100	100	91.7	100	94.3	83.2	78.9	92.7	100	66.7	15.5	40.1
BoSsNet w/o DLD	100	100	93.4	100	95.3	79.2	84.6	90.7	100	78.1	12.4	40.45
BoSsNet	100	100	95.2	100	97.3	100	100	95.7	100	91.7	15.2	43.1

Table 6: Ablation study: impact of each model element on BoSsNet

Most of the existing end-to-end approaches *retrieve* a response from a pre-defined set (Bordes and Weston, 2017; Liu and Perez, 2017; Seo et al., 2017). These methods are generally successful when they have to provide boilerplate responses – they cannot construct responses by using words in KB not seen during training. Alternatively, generative approaches are used where the response is *generated* one word at a time (Eric and Manning, 2017; Madotto et al., 2018). These approaches mitigate the unseen entity problem by incorporating the ability to copy words from the input (Vinyals et al., 2015; Gu et al., 2016). The copy mechanism has also found success in summarization (Nallapati et al., 2016; See et al., 2017) and machine translation (Gulcehre et al., 2016). BoSsNet is also a copy incorporated generative approach.

For encoding, some approaches represent the dialog history as a sequence (Eric and Manning, 2017; Gulcehre et al., 2016). Unfortunately, using a single long sequence for encoding also enforces an order over the set of KB tuples making it harder to perform inferencing over them. Other approaches represent the dialog context as a bag. Original Memory Networks (Bordes and Weston, 2017) and its extensions encode each memory element (utterance) as an average of all constituent words – this cannot point to individual words, and hence cannot be used with a copy mechanism. Mem2Seq encodes each word individually in a flat memory. Unfortunately, this loses the contextual information around a word, which is needed to decipher an unseen word. In contrast, BoSsNet uses a bag of sequences encoding, where KB tuples are a set for easier inference, and also each utterance is a sequence for effectively learning when to copy.

KB (restaurant—cuisine—address—phone)	
pizza_hut_fen_ditton—italian—cambridge_retail_park_newmarket_road_fen_ditton—01223_323737	
usr-1	may i have information for an *italian* restaurant in the *east* part of town ?
sys-1	yes sure . there is only on *italian* restaurant called *pizza_hut_fen_ditton* in the *east* . what else do you want to know ?
usr-2	what is their address please ?
Gold	the address for *pizza_hut_fen_ditton* is *cambridge_retail_park_newmarket_road_fen_ditton* . can i help you with anything else ?
Seq2Seq+Copy	their address is at
Seq2Seq	*curry_prince* is at *451_newmarket_road_fen_ditton*
Mem2Seq	*pizza_hut_fen_ditton* is located at *7_barnwell_road_fen_ditton* is *7_barnwell_road_fen_ditton* and the phone number is *01223_244955*
BOSSNET	*pizza_hut_fen_ditton* is located at *cambridge_retail_park_newmarket_road_fen_ditton* is there anything else i can help you with

Table 7: An example of responses generated by BOSSNET and baselines on the CamRest test set. Thia example has no unseen entities.

KB (restaurant—cuisine—price_range—location—rating)	
resto_tokyo_affordable_vietnamese_8stars—vietnamese—affordable—tokyo—8 *resto_tokyo_affordable_vietnamese_7stars—vietnamese—affordable—tokyo—7* *resto_tokyo_affordable_vietnamese_6stars—vietnamese—affordable—tokyo—6* *resto_tokyo_affordable_vietnamese_5stars—vietnamese—affordable—tokyo—5*	
usr-1	can you book a table in *tokyo* with *vietnamese* food in a *affordable* price range ?
sys-1	what do you think of this option : *resto_tokyo_affordable_vietnamese_8stars* ?
usr-2	no this does not work for me .
sys-2	what do you think of this option : *resto_tokyo_affordable_vietnamese_7stars* ?
usr-3	do you have something else ?
Gold	what do you think of this option : *resto_tokyo_affordable_vietnamese_6stars*
Seq2Seq+Copy	what do you think of this option : what ?
Seq2Seq	what do you think of this option : *resto_london_moderate_british_2stars* ?
Mem2Seq	what do you think of this option : *resto_tokyo_affordable_vietnamese_5stars* ?
BOSSNET	what do you think of this option : *resto_tokyo_affordable_vietnamese_6stars* ?

Table 8: An example of responses generated by BOSSNET and baselines on bAbI dialog Task-5. This example is from the KA test set with 100% unseen entities.

6 Conclusions

We propose BOSSNET for training task-oriented dialog systems in an end-to-end fashion. BOSSNET combines a novel bag of sequences memory for storing a dialog history and KB tuples, with a copy-augmented generative decoder to construct dialog responses. It augments standard cross entropy loss of a sequence decoder with an additional term to encourage the model to copy KB words. BOSS memory and new loss term, in conjunction with a disentangle label dropout, enables the decoder to disentangle its language and knowledge models.

BOSSNET achieves the state of the art results on bAbI dialog dataset, outperforming existing models by 10 points or more in its OOV conditions. In the knowledge adaptability test, we find that BOSSNET is highly robust to increasing the percentage of unseen entities at test time, suggesting a good language-knowledge disentanglement. Human evaluations show that BOSSNET responses are highly informative and slightly more grammatical compared to baselines. We will release our code and all curated datasets for further research.

Acknowledgments

We thank Danish Contractor, Gaurav Pandey and Sachindra Joshi for their comments on an earlier version of this work. This work is supported by IBM AI Horizons Network grant, an IBM SUR award, grants by Google, Bloomberg and 1MG, and a Visvesvaraya faculty award by Govt. of India. We thank Microsoft Azure sponsorships, and the IIT Delhi HPC facility for computational resources.

References

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Dialog System Technology Challenges, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with re- current neural networks. In *In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1–10.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

A. Madotto, CS. Wu, and P. Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *International Conference on Learning Representations*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference*, volume 1, pages 438–449.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*, pages 2153–2162, Austin, Texas. ACL.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. volume 21, pages 393–422. Elsevier.

A Two-Level attention on BoSs Memory

To visualize the benefit of two-level attention used on BOSS memory by the decoder, we compare attention weights for two models: our proposed *two-level attention* and a variant with just *one-level attention* (over all the words in the memory). In the example of a sample dialog from bAbI Task 3, shown in Figure 7, the decoder is aimed at predicting the second best restaurant *3 stars*, given that the restaurant with rating *8 stars* has already been suggested and rejected. We show attention only on the KB entries for brevity.

The models share some similarities in their distribution of attention. First, the attention weights are localized over the restaurant names, indicating the preference of the system to point to a specific restaurant. This is supported by the g_s values, 3.14 x 10^{-5} and 1.15 x 10^{-4} for two-level attention and one-level attention respectively, i.e., both models prefer to copy rather than generate. Moreover, entries with the same restaurant name have similar attention weights, reflecting the robustness of the distribution.

We also observe that two-level attention is able to perform the difficult task of *sorting* the restaurant entries based on decreasing order of rating (number of stars). It gives more weight to entries with a high rating (*3 stars* $>$ *2 stars* $>$ *1 star*) and suppresses the weights of any previously suggested restaurant.

The attention over memory cells provides BOSSNET with the ability to infer over multiple sets of tuples. The ability to sort the restaurants and reject a previously seen restaurant can be observed by the attention heat map of Memory cells. Attention over tokens on the other hand can push the attention weights towards either the subject or object in the KB tuple, based on the query's request. Thus using both in conjunction helps BOSSNET perform significantly better than the baselines and illustrates the importance of the BOSS memory in comparison to a flat memory layout.

B Reproducibility

We list out the complete set of hyperparameters used to train BOSSNET for the various datasets in Table 9. Our code will be made publicably accessible for future research purposes. Our trained models and evaluation scripts will also be provided. We will also make our end-to-end reconstruced Camrest dataset along with our whole batch of knowledge adaptability test sets available.

C Example Predictions of BOSSNET and Baselines

Examples from SMD is shown in Table 12 respectively. Examples from KA test set with percentage of unseen entites set to 50 from CamRest and SMD are shown in Table 11 and Table 13 respectively. Examples from KA test set with percentage of unseen entites set to 100 from bAbI dialog Task 1 is shown in Table 10.

D Dataset Preprocessing and Faults

D.1 Mem2Seq Preprocessing

Mem2Seq paper used the following pre-processing on the data:

1. The subject (restaurant name) and object (rating) positions of the rating KB tuples in bAbI dialogs are flipped, while the order remains the same for other tuples remains the same. This pre-processing is illustrated in Figure 8

2. an extra fact was added to the navigation tasks in In-Car Assistant with all the properties (such as distance, address) combined together as the subject and *poi* as the object. This pre-processing is illustrated in Figure 9

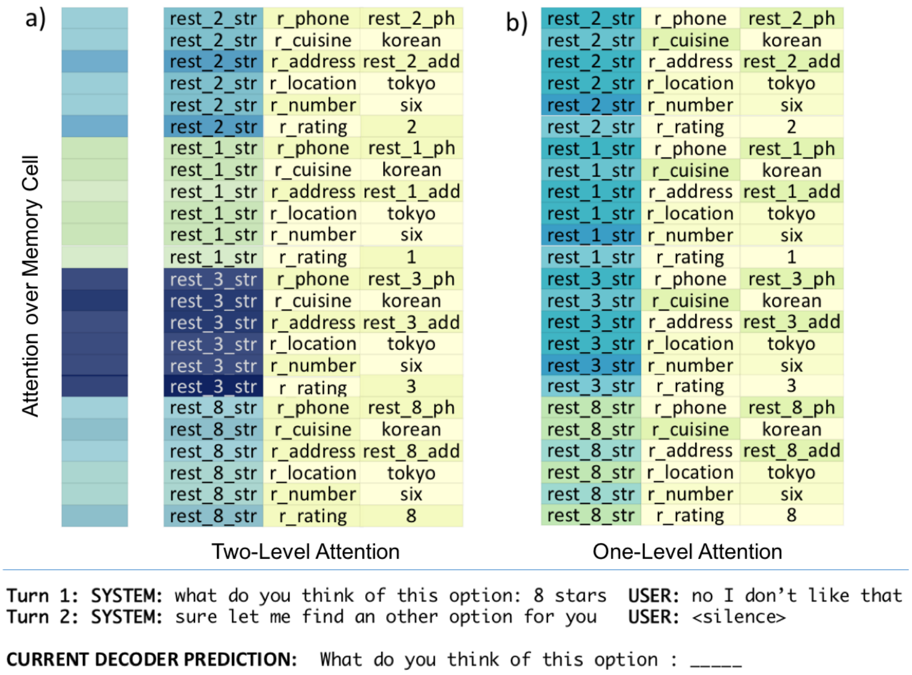
The pre-processing has major impact on the performance of Mem2Seq, as it can only copy objects of a KB tuple, while the subject and relation can never be copied.

D.2 bAbI Dataset Faults

The KB entities present in validation and non-OOV test sets for task 3 and 4 do not overlap with those in the train set. This effectively means that non-OOV and OOV test conditions are the same for tasks 3 and 4. This explains the low performance of baseline models on task 3 and 4 non-OOV test sets.

E AMT Setup

Response Relevance Test We show a sample of an Human Intelligence Task (HIT) on Amazon Mechanical Turk in Figure 10a. We randomize the responses generated by the three baseline models and BOSSNET on the same dialog and ask the user to tick all those response options that seem to capture the relevant information of the given sample response. A total of 200 such annotations were collected for Camrest and SMD each.



rest_2_str	r_phone	rest_2_ph
rest_2_str	r_cuisine	korean
rest_2_str	r_address	rest_2_add
rest_2_str	r_location	tokyo
rest_2_str	r_number	six
rest_2_str	r_rating	2
rest_1_str	r_phone	rest_1_ph
rest_1_str	r_cuisine	korean
rest_1_str	r_address	rest_1_add
rest_1_str	r_location	tokyo
rest_1_str	r_number	six
rest_1_str	r_rating	1
rest_3_str	r_phone	rest_3_ph
rest_3_str	r_cuisine	korean
rest_3_str	r_address	rest_3_add
rest_3_str	r_location	tokyo
rest_3_str	r_number	six
rest_3_str	r_rating	3
rest_8_str	r_phone	rest_8_ph
rest_8_str	r_cuisine	korean
rest_8_str	r_address	rest_8_add
rest_8_str	r_location	tokyo
rest_8_str	r_number	six
rest_8_str	r_rating	8

Two-Level Attention

rest_2_str	r_phone	rest_2_ph
rest_2_str	r_cuisine	korean
rest_2_str	r_address	rest_2_add
rest_2_str	r_location	tokyo
rest_2_str	r_number	six
rest_2_str	r_rating	2
rest_1_str	r_phone	rest_1_ph
rest_1_str	r_cuisine	korean
rest_1_str	r_address	rest_1_add
rest_1_str	r_location	tokyo
rest_1_str	r_number	six
rest_1_str	r_rating	1
rest_3_str	r_phone	rest_3_ph
rest_3_str	r_cuisine	korean
rest_3_str	r_address	rest_3_add
rest_3_str	r_location	tokyo
rest_3_str	r_number	six
rest_3_str	r_rating	3
rest_8_str	r_phone	rest_8_ph
rest_8_str	r_cuisine	korean
rest_8_str	r_address	rest_8_add
rest_8_str	r_location	tokyo
rest_8_str	r_number	six
rest_8_str	r_rating	8

One-Level Attention

Turn 1: SYSTEM: what do you think of this option: 8 stars **USER:** no I don't like that
Turn 2: SYSTEM: sure let me find an other option for you **USER:** <silence>

CURRENT DECODER PREDICTION: What do you think of this option : _____

Figure 7: Visualization of attention weights on selected portions of memory in (a) BOSSNET with two-level attention vs (b) BOSSNET with one-level attention

Task	Learning Rate	Hops	Embedding Size	Disentangle Loss Weight	DLD
T1	0.001	1	128	1.0	0.2
T2	0.001	1	128	1.0	0.2
T3	0.0005	3	128	1.5	0.2
T4	0.001	1	128	1.0	0.2
T5	0.0005	3	256	1.0	0.2
CamRest	0.0005	6	256	1.0	0.2
SMD	0.0005	3	256	1.0	0.1

Table 9: The hyperparameters used to train BOSSNET on the different datasets
.

Response Grammar Test We show a sample of an Human Intelligence Task (HIT) on Amazon Mechanical Turk in Figure 10b. We randomize the responses generated by the three baseline models and BOSSNET on the same dialog and ask the user to rate each response based on the grammatical correctness and natural flow of the sentence. The rating ranges from 0-3 where 0 being the worst and 3 being the best. Note) the sentences were not asked to be rated with respect to each other, but instead as individual occurrences. A total of 200 such annotations were collected for Camrest and SMD each.

F Multi-Hop vs 1-Hop Encoders

Table 14 shows the performance of bAbI tasks and CamRest on two BOSSNET encoder settings. Multi-hops in encoder helps in bAbI task 3 and 5, as they require inferencing over the KB tuples (sorting restaurants by rating) to recommend a restaurant. We also see substantial improvements on CamRest in both BLEU and entity F1 metric.

usr-1 **sys-1** **usr-2** **sys-2** **usr-3**	good morning hello what can i help you with today can you book a table with *korean* cuisine for *five* in *beijing* i ' m on it . which price range are looking for ? i am looking for a *overpriced* restaurant
Gold	api_call *korean beijing five overpriced*
Seq2Seq+Copy	api_call *korean beijing five overpriced*
Seq2Seq	api_call *french london four moderate*
Mem2Seq	api_call *overpriced korean korean overpriced*
BOSSNET	api_call *korean beijing five overpriced*

Table 10: Example from bAbI dialog Task 1 with 100% OOV.

<table>
<tr><th colspan="2">KB (restaurant—cuisine—address—phone—postcode)
prezzo—ethiopian—22_atlantis_road—01799_521260—cb30ad</th></tr>
<tr><td>**usr-1**
sys-1
usr-2
sys-2
usr-3</td><td>i want a *moderately_expensive ethiopian* restaurant in the *south* side of town
i ' m sorry . i can not find a *moderately_expensive ethiopian* restaurant in the *south* side of town
i would like to find a *moderately* priced restaurant in the *west* part of town
prezzo is a moderately priced *ethiopian* restaurant in the *west* part of town . there is also *indian* and *british* food available in that area and price range . would you like more info ?
yes can i get thier address and postcode ?</td></tr>
<tr><td>**Gold**</td><td>of course , *prezzo* ' s address is *22_atlantis_road* and postcode is *cb30ad*</td></tr>
<tr><td>**Seq2Seq+Copy**</td><td>postcode a is is *01223350688* UNK there get i for for today</td></tr>
<tr><td>**Seq2Seq**</td><td>the address is *83_regent_street* and their postcode is *cb58aq*</td></tr>
<tr><td>**Mem2Seq**</td><td>the phone address is *cb21uj*</td></tr>
<tr><td>BOSSNET</td><td>the address is *22_atlantis_road at 22_atlantis_road* the postcode is *cb30ad* is there anything else i can help you with</td></tr>
</table>

Table 11: Example from Camrest with 50% OOV. The OOV entities present in the dialog are {*ethiopian*, *22_atlantis_road*}

<table>
<tr><th colspan="2">KB (location—distance—traffic info—address)
teavana—4_miles—no_traffic—145_amherst_st
cafe_venetia—5_miles—moderate_traffic—269_alger_dr
town_and_country—3_miles—moderate_traffic—383_university_ave</th></tr>
<tr><td>**usr-1**
sys-1
usr-2</td><td>where can i get tea ?
there is 2 tea shops near you *teavana* is *4_miles* away and *cafe_venetia* is *5_miles* away which one do you want ?
which one is the quickest is there any traffic ?</td></tr>
<tr><td>**Gold**</td><td>*teavana* is the fastest journey because there is *no_traffic* on the way there . *cafe_venetia* has *moderate_traffic* should we go to *teavana*</td></tr>
<tr><td>**Seq2Seq+Copy**</td><td>there is</td></tr>
<tr><td>**Seq2Seq**</td><td>there is *moderate_traffic* on the way to *palo_alto_medical_foundation*</td></tr>
<tr><td>**Mem2Seq**</td><td>there is *moderate_traffic* on our way</td></tr>
<tr><td>BOSSNET</td><td>*teavana* there is a *moderate_traffic* nearby but i might find another route to t*own_and_country* *town_and_country* address is *383_university_ave* i sent it on your screen</td></tr>
</table>

Table 12: Example from SMD

<table>
<tr><td colspan="2">KB (location—day—weather)
<i>mountain_view—sunday—foggy</i>
<i>mountain_view—monday—overcast</i>
<i>pittsburgh—sunday—overcast</i>
<i>pittsburgh—monday—warm</i>
<i>downtown_chicago—sunday—foggy</i>
<i>downtown_chicago—monday—warm</i></td></tr>
<tr><td colspan="2"><i>today—monday</i></td></tr>
<tr><td>usr-1
sys-1
usr-2</td><td>is the weather <i>cloudy</i> right now ?
what city do you want the weather for ?
<i>pittsburgh</i></td></tr>
<tr><td>Gold</td><td>there are no clouds in <i>pittsburgh</i> right now</td></tr>
<tr><td>Seq2Seq+Copy</td><td>which is no</td></tr>
<tr><td>Seq2Seq</td><td>it is currently raining in <i>san_francisco</i> right now</td></tr>
<tr><td>Mem2Seq</td><td>it will not be cloudy in in</td></tr>
<tr><td>BoSsNet</td><td>it will be <i>cloudy</i> on <i>sunday</i> in <i>pittsburgh</i></td></tr>
</table>

Table 13: Example from SMD with 50% OOV. The OOV entity present in the dialog is {*pittsburgh*}

	bAbI Dialog Tasks					**bAbI Dialog Tasks (OOV)**					**CamRest**	
	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	BLEU	Ent. F1
BoSsNet with 1-Hop Encoder	100	100	92.3	100	90.5	100	100	91.4	100	89	10.5	36.9
BoSsNet with Multi-Hop Encoder	100	100	95.2	100	97.3	100	100	95.7	100	91.7	15.2	43.1

Table 14: Ablation study: impact of hops in BoSsNet encoder

(a) Original bAbI Data

Subject	Predicate	Object
resto_rome_cheap_indian_6stars	R_phone	resto_rome_cheap_indian_6stars_phone
resto_rome_cheap_indian_6stars	R_cuisine	indian
resto_rome_cheap_indian_6stars	R_address	resto_rome_cheap_indian_6stars_address
resto_rome_cheap_indian_6stars	R_location	rome
resto_rome_cheap_indian_6stars	R_number	eight
resto_rome_cheap_indian_6stars	R_price	cheap
resto_rome_cheap_indian_6stars	R_rating	6
resto_rome_cheap_indian_7stars	R_phone	resto_rome_cheap_indian_7stars_phone
resto_rome_cheap_indian_7stars	R_cuisine	indian
resto_rome_cheap_indian_7stars	R_address	resto_rome_cheap_indian_7stars_address
resto_rome_cheap_indian_7stars	R_location	rome
resto_rome_cheap_indian_7stars	R_number	eight
resto_rome_cheap_indian_7stars	R_price	cheap
resto_rome_cheap_indian_7stars	R_rating	7

Subject	Predicate	Object
resto_rome_cheap_indian_6stars	R_phone	resto_rome_cheap_indian_6stars_phone
resto_rome_cheap_indian_6stars	R_cuisine	indian
resto_rome_cheap_indian_6stars	R_address	resto_rome_cheap_indian_6stars_address
resto_rome_cheap_indian_6stars	R_location	rome
resto_rome_cheap_indian_6stars	R_number	eight
resto_rome_cheap_indian_6stars	R_price	cheap
6	R_rating	resto_rome_cheap_indian_6stars
resto_rome_cheap_indian_7stars	R_phone	resto_rome_cheap_indian_7stars_phone
resto_rome_cheap_indian_7stars	R_cuisine	indian
resto_rome_cheap_indian_7stars	R_address	resto_rome_cheap_indian_7stars_address
resto_rome_cheap_indian_7stars	R_location	rome
resto_rome_cheap_indian_7stars	R_number	eight
resto_rome_cheap_indian_7stars	R_price	cheap
7	R_rating	resto_rome_cheap_indian_7stars

(a) Pre-Processed bAbI Data

Figure 8: Pre-processing of bAbI dialog data used in Mem2Seq paper

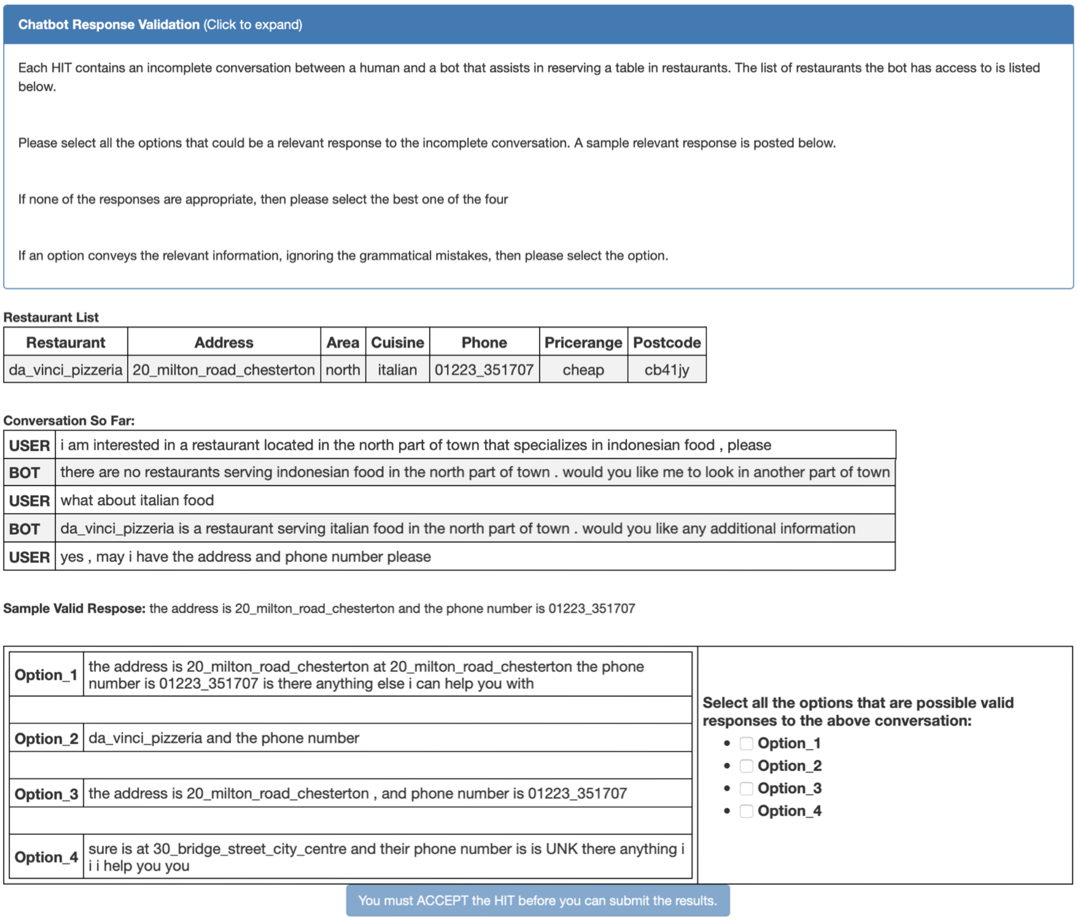
(a) Original SMD Navigate Data

Subject	Predicate	Object
the_westin	distance	2_miles
the_westin	traffic_info	moderate_traffic
the_westin	poi_type	rest_stop
the_westin	address	329_el_camino_real
toms_house	distance	1_miles
toms_house	traffic_info	heavy_traffic
toms_house	poi_type	friends_house
toms_house	address	580_van_ness_ave

Subject	Predicate	Object
2_miles moderate_traffic rest_stop	poi	the_westin
the_westin	distance	2_miles
the_westin	traffic_info	moderate_traffic
the_westin	poi_type	rest_stop
the_westin	address	329_el_camino_real
1_miles heavy_traffic friends_house	poi	toms_house
toms_house	distance	1_miles
toms_house	traffic_info	heavy_traffic
toms_house	poi_type	friends_house
toms_house	address	580_van_ness_ave

(a) Pre-Processed SMD Navigate Data

Figure 9: Pre-processing of SMD Navigate data used in Mem2Seq paper



Chatbot Response Validation (Click to expand)

Each HIT contains an incomplete conversation between a human and a bot that assists in reserving a table in restaurants. The list of restaurants the bot has access to is listed below.

Please select all the options that could be a relevant response to the incomplete conversation. A sample relevant response is posted below.

If none of the responses are appropriate, then please select the best one of the four

If an option conveys the relevant information, ignoring the grammatical mistakes, then please select the option.

Restaurant List

Restaurant	Address	Area	Cuisine	Phone	Pricerange	Postcode
da_vinci_pizzeria	20_milton_road_chesterton	north	italian	01223_351707	cheap	cb41jy

Conversation So Far:

USER	i am interested in a restaurant located in the north part of town that specializes in indonesian food , please
BOT	there are no restaurants serving indonesian food in the north part of town . would you like me to look in another part of town
USER	what about italian food
BOT	da_vinci_pizzeria is a restaurant serving italian food in the north part of town . would you like any additional information
USER	yes , may i have the address and phone number please

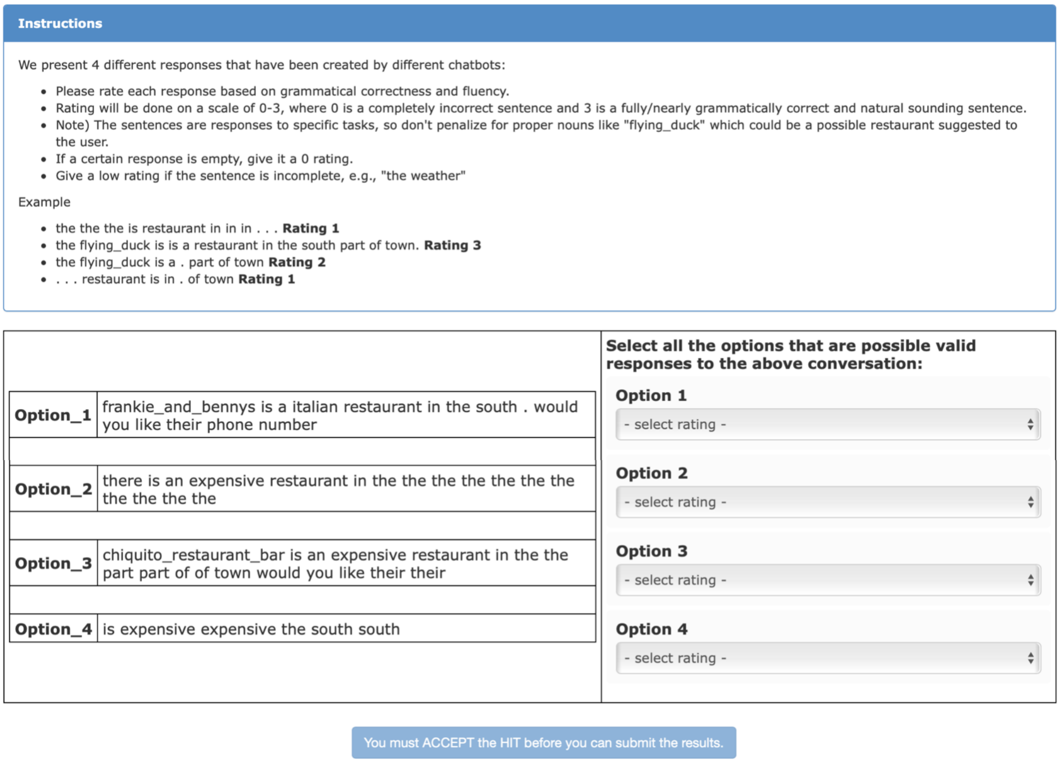
Sample Valid Respose: the address is 20_milton_road_chesterton and the phone number is 01223_351707

Option_1	the address is 20_milton_road_chesterton at 20_milton_road_chesterton the phone number is 01223_351707 is there anything else i can help you with
Option_2	da_vinci_pizzeria and the phone number
Option_3	the address is 20_milton_road_chesterton , and phone number is 01223_351707
Option_4	sure is at 30_bridge_street_city_centre and their phone number is is UNK there anything i i i help you you

Select all the options that are possible valid responses to the above conversation:

- ☐ **Option_1**
- ☐ **Option_2**
- ☐ **Option_3**
- ☐ **Option_4**

(a)



Instructions

We present 4 different responses that have been created by different chatbots:

- Please rate each response based on grammatical correctness and fluency.
- Rating will be done on a scale of 0-3, where 0 is a completely incorrect sentence and 3 is a fully/nearly grammatically correct and natural sounding sentence.
- Note) The sentences are responses to specific tasks, so don't penalize for proper nouns like "flying_duck" which could be a possible restaurant suggested to the user.
- If a certain response is empty, give it a 0 rating.
- Give a low rating if the sentence is incomplete, e.g., "the weather"

Example

- the the the is restaurant in in in . . . **Rating 1**
- the flying_duck is is a restaurant in the south part of town. **Rating 3**
- the flying_duck is a . part of town **Rating 2**
- . . . restaurant is in . of town **Rating 1**

Option_1	frankie_and_bennys is a italian restaurant in the south . would you like their phone number
Option_2	there is an expensive restaurant in the the the the the the the the the the
Option_3	chiquito_restaurant_bar is an expensive restaurant in the the part part of of town would you like their their
Option_4	is expensive expensive the south south

Select all the options that are possible valid responses to the above conversation:

Option 1

- select rating -

Option 2

- select rating -

Option 3

- select rating -

Option 4

- select rating -

(b)

Figure 10: A sample HIT on Amazon Mechanical Turk to (a) validate useful responses based on the given dialog context, and (b) validate grammatical correctness of different responses on a scale of 0-3

Tensorized Self-Attention: Efficiently Modeling Pairwise and Global Dependencies Together

Tao Shen[1], **Tianyi Zhou**[2], **Guodong Long**[1], **Jing Jiang**[1], **Chengqi Zhang**[1]
[1] Centre for Artificial Intelligence, School of Software, University of Technology Sydney
[2] Paul G. Allen School of Computer Science & Engineering, University of Washington
`tao.shen@student.uts.edu.au`, `tianyizh@uw.edu`,
{`guodong.long`, `jing.jiang`, `chengqi.zhang`}`@uts.edu.au`

Abstract

Neural networks equipped with self-attention have parallelizable computation, light-weight structure, and the ability to capture both long-range and local dependencies. Further, their expressive power and performance can be boosted by using a vector to measure pairwise dependency, but this requires to expand the alignment matrix to a tensor, which results in memory and computation bottlenecks. In this paper, we propose a novel attention mechanism called "Multi-mask Tensorized Self-Attention" (MTSA), which is as fast and as memory-efficient as a CNN, but significantly outperforms previous CNN-/RNN-/attention-based models. MTSA 1) captures both pairwise (token2token) and global (source2token) dependencies by a novel compatibility function composed of dot-product and additive attentions, 2) uses a tensor to represent the feature-wise alignment scores for better expressive power but only requires parallelizable matrix multiplications, and 3) combines multi-head with multi-dimensional attentions, and applies a distinct positional mask to each head (subspace), so the memory and computation can be distributed to multiple heads, each with sequential information encoded independently. The experiments show that a CNN/RNN-free model based on MTSA achieves state-of-the-art or competitive performance on nine NLP benchmarks with compelling memory- and time-efficiency.

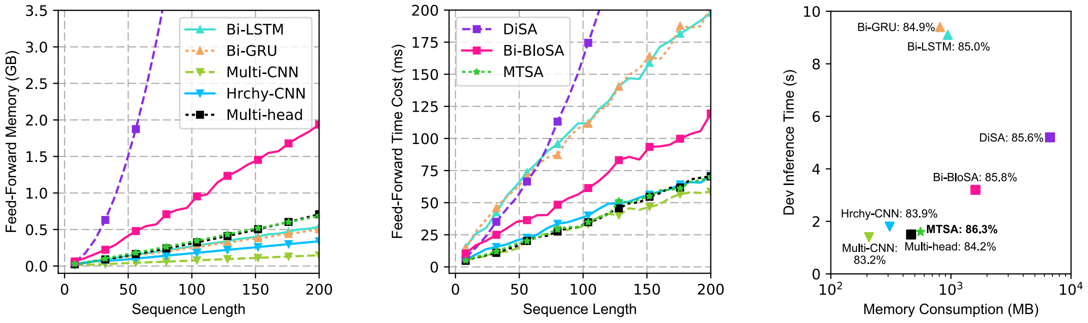
1 Introduction

Recurrent neural network (RNN) and convolutional neural network (CNN) have been broadly used as context fusion modules for natural language processing (NLP) tasks. Recently, RNN/CNN in conjunction with an attention mechanism has been proven to be effective for contextual feature modeling in a wide range of NLP tasks, including sentiment classification (Li et al., 2018), machine translation (Bahdanau et al., 2015), reading comprehension (Seo et al., 2017; Yu et al., 2018), etc. More recently, self-attention mechanisms have been developed for context fusion and syntactic dependency modeling with the advantage of fewer parameters, more parallelizable computation, and better empirical performance (Hu et al., 2017; Vaswani et al., 2017; Shen et al., 2018a). In addition, neural networks based solely on self-attention mechanisms have achieved state-of-the-art quality on many NLP tasks, e.g., machine translation (Vaswani et al., 2017), sentence embedding (Shen et al., 2018a) and semantic role labeling (Tan et al., 2017).

Self-attention mechanisms can be categorized into two classes according to the type of dependency each aims to model. The first category is *token2token* self-attention (Hu et al., 2017; Vaswani et al., 2017; Shen et al., 2018a) that captures syntactic dependency between every two tokens in a sequence. An efficient dot-product compatibility function is usually deployed to measure this pairwise dependency (Vaswani et al., 2017). In contrast, additive compatibility function captures the dependency by multi-layer perceptron (MLP), and can usually achieve better performance (Britz et al., 2017). Its expressive power can be further improved if expanded to multiple dimensions (Shen et al., 2018a). This multi-dim self-attention empirically surpasses dot-product one, but suffers from expensive computation and memory, which grow linearly with the number of features and quadratically with the sequence length. Hence, it is not scalable to long sequences in practice.

The second category is *source2token* self-attention (Liu et al., 2016; Lin et al., 2017; Shen et al., 2018a) aiming to capture global dependency, i.e., the importance of each token to the entire sequence for a specific task. Its time and space complexities grow linearly, rather than quadratically,

Proceedings of NAACL-HLT 2019, pages 1256–1266
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) (b) (c)

Figure 1: (a) Memory consumption and (b) time cost *vs.* sequence length on synthetic data; (c) memory load (x-axis), inference time on dev set (y-axis) and test accuracy on the SNLI dataset.

with the sequence length. Hence, it is empirically efficient in terms of memory and computation even if expanded to multiple dimensions, i.e., using a vector of feature-wise scores instead of a scalar for the global dependency. But, it is hard to reach state-of-the-art performance on NLP tasks due to the lack of pairwise and local dependencies.

In this paper, we propose a novel attention mechanism called multi-mask tensorized self-attention (MTSA), for context fusion. In MTSA, 1) the pairwise dependency is captured by an efficient dot-product based token2token self-attention, while the global dependency is modeled by a feature-wise multi-dim source2token self-attention, so they can work jointly to encode rich contextual features; 2) self-attention alignment scores are tensorized for more expressive power in that each pair of tokens has one score for each feature, but no tensor computation is required other than simple and efficient matrix multiplications when implemented; 3) the tensors above are computed in multiple subspaces (i.e., in a multi-head fashion) rather than in the original input space, so the required memory and computation can be distributed to multiple subspaces; and 4) a distinct positional mask is applied to each head in order to encode rich structural information such as the sequential order and relative position of tokens.

In the experiments, we build CNN/RNN-free neural networks based on MTSA for sentence embedding and sequence tagging tasks, including natural language inference, semantic role labeling, sentiment analysis, question-type classification, machine translation, etc. The results demonstrate that MTSA achieves state-of-the-art or competitive performance on nine benchmark datasets. To summarize the comparison of MTSA with recently popular models, we show the memory consumption and time cost *vs.* sequence length respectively in Figure 1(a) and 1(b) on synthetic data (batch size of 64 and feature channels of 300). On the SNLI (Bowman et al., 2015), a public dataset for language inference, as shown in Figure 1(c), MTSA achieves the best result but is as fast and as memory-efficient as the CNNs (all baselines and the benchmark are detailed in Section 4).

Notations: 1) lowercase denotes a vector; 2) bold lowercase denotes a sequence of vectors (stored as a matrix); and 3) uppercase denotes a matrix or tensor.

2 Background

2.1 Attention Mechanism

Given an input sequence of token embeddings or memory slots $\boldsymbol{x} = [x_1, \dots, x_n] \in \mathbb{R}^{d_e \times n}$, and a vector representation of a query $q \in \mathbb{R}^{d_q}$, attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) computes an alignment score between each token x_i and q by a compatibility function $f(x_i, q)$, which aims to measure the dependency/relevance between x_i and q, or the attention of q to x_i, w.r.t. a given task. The scores are transformed into probabilities through a $\mathrm{softmax}$ function. These probabilities are then used as weights to sum all the tokens and generate a contextual embedding for q, i.e.,

$$p(z|\boldsymbol{x}, q) = \mathrm{softmax}(a), \quad a = [f(x_i, q)]_{i=1}^{n},$$
$$s = \sum_{i=1}^{n} p(z = i|\boldsymbol{x}, q) \cdot x_i = \mathbb{E}_{i \sim p(z|\boldsymbol{x}, q)}[x_i], \quad (1)$$

where $a \in \mathbb{R}^n$ denotes the vector of n alignment scores, $p(z|\boldsymbol{x}, q)$ is the categorical distribution for attention probabilities, which is derived from applying softmax function to a. And, $s \in \mathbb{R}^{d_e}$ is the output vector for the query q.

There are two major types of compatibility functions, leading to the two most frequently used attention mechanisms. The first one is dot-product or multiplicative compatibility function (Eq.(2)), which composes **dot-product attention** mechanism (Luong et al., 2015) using cosine similarity to model the dependencies. The other one is additive or multi-layer perceptron (MLP) compatibility function (Eq.(3)) that results in **additive attention** mechanism (Bahdanau et al., 2015) using MLP to model the dependencies.

$$f(x_i, q) = \langle W^{(d1)} x_i, W^{(d2)} q \rangle, \qquad (2)$$

$$f(x_i, q) = w^T \sigma_a(W^{(a)}[x_i; q] + b^{(a)}) + b, \qquad (3)$$

where $W^{(d1)} \in \mathbb{R}^{d_i \times d_e}$, $W^{(d2)} \in \mathbb{R}^{d_i \times d_q}$, $W^{(a)} \in \mathbb{R}^{d_a \times (d_e + d_q)}$, $w \in \mathbb{R}^{d_a}$ are learnable parameters, $\langle \cdot, \cdot \rangle$ denotes inner-product. Empirically, networks with additive attention usually outperform those with dot-product attention, but require more computation time and memory (Britz et al., 2017).

Multi-dim attention mechanism (Shen et al., 2018a) expands the alignment score in previous attention mechanisms to a vector for feature-wise scores, each computed on a feature dimension. It has greater capacity to model complex dependencies, and can handle context variation and polysemy problems harassing many NLP tasks. **In particular, it replaces vector $w^T \in \mathbb{R}^{1 \times d_a}$ in additive compatibility function (Eq.(3)) with a matrix $W \in \mathbb{R}^{d_e \times d_a}$, and thus produces d_e scores to describe the attention of q to x_i.**

2.2 Self-Attention Mechanism

Self-attention mechanism is a special case of attention mechanisms, where the query q stems from the input sequence itself. Self-attention mechanisms can be classified into token2token or source2token self-attention mechanism according to the type of dependency each aims to model.

A) Token2token self-attention mechanism (Vaswani et al., 2017; Shen et al., 2018a) aims at producing a context-aware representation for each token in light of its syntactic dependencies on other tokens from the same sequence. Two examples of token2token self-attention are 1) scaled dot-product self-attention which composes the multi-head self-attention (Vaswani et al., 2017), and 2) masked self-attention used in directional self-attention (Shen et al., 2018a).

A.1) Scaled dot-product attention mechanism (Vaswani et al., 2017) in general form has three arguments: query tokens $\boldsymbol{q} \in \mathbb{R}^{d_i \times m}$, key tokens $\boldsymbol{k} \in \mathbb{R}^{d_i \times n}$ and value tokens $\boldsymbol{v} \in \mathbb{R}^{d_h \times n}$ associated with the key tokens. It uses a scaled dot-product function to model the relationship between each query and key, and finally outputs a sequence $\boldsymbol{s} = [s_1, \dots, s_m] \in \mathbb{R}^{d_h \times m}$ such that

$$\boldsymbol{s} = \mathrm{sdpAttn}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) \triangleq \boldsymbol{v}\, \mathrm{softmax}(\frac{\boldsymbol{q}^T \boldsymbol{k}}{\sqrt{d_q}})^T \qquad (4)$$

A special case of this mechanism is that the three input arguments are derived from the same source, i.e., ***q*/*k*/*v*** $= f^{q/k/v}(\boldsymbol{x})$, which can be referred to as a token2token self-attention, namely scaled dot-product self-attention. As for **multi-head attention** mechanism, the input is projected into multiple subspaces, then parameter-untied scaled dot-product attention is applied to the embeddings in each subspace. The results for multiple subspaces are concatenated to form the final output $\boldsymbol{s}$, i.e.,

$$\boldsymbol{s} = W^{(o)}[H_1; \dots; H_h], \qquad (5)$$
$$\text{where } H_c = \mathrm{sdpAttn}(W_c^q \boldsymbol{q}, W_c^k \boldsymbol{k}, W_c^v \boldsymbol{v}).$$

A.2) Masked self-attention mechanism (Shen et al., 2018a) uses multi-dim compatibility function to model the dependency between every two tokens in a sequence, and uses positional mask to encode sequential information. It overcomes inherent problem appearing in self-attention compared to RNNs on the lack of sequential information. Its compatibility function is defined as

$$f(x_i, x_j) = c \cdot \tanh\{(W^{(m)}[x_i; x_j] + b^{(m)})/c\} + M_{i,j} \qquad (6)$$

where c is a constant scalar, $W^{(m)} \in \mathbb{R}^{d_e \times 2d_e}$ is learnable weight matrix, and M is a positional mask with each entry $M_{i,j} \in \{-\infty, 0\}$. When $M_{i,j} = -\infty$, applying softmax function to the alignment scores results in a zero attention probability, which cuts off the attention of x_j to x_i. Hence, masked self-attention with an asymmetric mask, where $M_{ij} \neq M_{ji}$, can encode sequential or other structural information (Shen et al., 2018a; Im and Cho, 2017). To this end, two positional masks have been proposed to encode the forward and backward order information respectively, i.e.,

$$M_{i,j}^{fw} = \begin{cases} 0, & i < j \\ -\infty, & \text{otherwise} \end{cases} \quad M_{i,j}^{bw} = \begin{cases} 0, & i > j \\ -\infty, & \text{otherwise} \end{cases}$$

Furthermore, **directional self-attention** (DiSA) (Shen et al., 2018a) concatenates the features produced by masked self-attention mechanisms with the forward and backward positional masks (i.e., M^{fw}, M^{bw}), leading to context-ware representations with bi-directional information encoded.

B) Source2token self-attention mechanism (Liu et al., 2016; Lin et al., 2017; Shen et al., 2018a) is designed for sentence embedding or sequence compression, which is based on the importance of each token x_i to the entire source sequence $\boldsymbol{x}$ for a specific task. Specifically, it removes the query q from the compatibility function $f(x_i, q)$ when computing the alignment score. For example, the compatibility function of additive source2token self-attention mechanism is to simply remove q from Eq.(3).

3 Proposed Models

In this section, we firstly elaborate on tensorized self-attention (TSA) in Section 3.1, which captures both pairwise and global dependencies by combining the two types of self-attention mechanisms introduced in Section 2.2. Then, we extend TSA to multi-mask tensorized self-attention (MTSA) in Section 3.2 by applying different positional masks to TSA in multiple subspaces (multi-head fashion). Lastly, in Section 3.3, we present an efficient computation scheme for MTSA without any high-rank tensor computation involved even if tensorized alignment scores are used.

3.1 Tensorized Self-Attention (TSA)

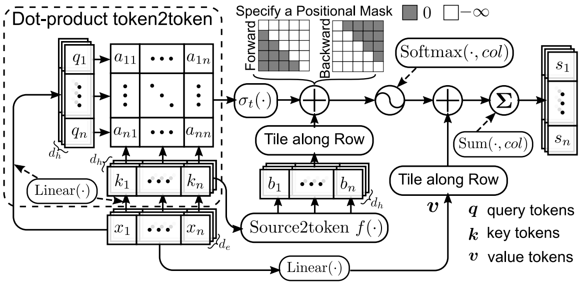


Figure 2: Tensorized self-attention (TSA) Mechanism.

Tensorized self-attention (TSA), whose structure is illustrated in Figure 2, is a neural mechanism that can be trained to model both pairwise and global dependencies, while any previous self-attention mechanism only focuses on one type of dependencies. TSA models both types by combining the aforementioned token2token and source2token self-attention mechanisms. This generates an $n \times n \times d_h$ tensor containing the alignment scores between every two tokens on each feature dimension. These scores are then normalized and transformed into probability weights, which are used to sum all dependent tokens and then generate the contextual embedding for each input token. We will demonstrate later in Section 3.3 that only matrix rather than tensor operation is required when executing the procedures above.

To facilitate the elaboration of proposed models and keep the consistent notation with prior attention mechanisms, TSA first projects the input embeddings $\boldsymbol{x}$ into three spaces to represent the query, key and value tokens, respectively.

$$\boldsymbol{q} = W^{(t1)}\boldsymbol{x}, \ \boldsymbol{k} = W^{(t2)}\boldsymbol{x}, \text{ and } \boldsymbol{v} = W^{(t3)}\boldsymbol{x}, \quad (7)$$

where $W^{(t1)}, W^{(t2)} \in \mathbb{R}^{d_i \times d_e}$ and $W^{(t3)} \in \mathbb{R}^{d_h \times d_e}$ are learnable weights for projections.

TSA then integrates two kinds of compatibility functions from two self-attention mechanisms respectively. Firstly, the scaled dot-product self-attention is used to capture dependency between every two tokens. Dot-product operations are fast, and sufficient to model the **pairwise dependency** in most tasks. Its compatibility function is

$$f^t(k_i, q_j) = \langle k_i, \ q_j \rangle / \sqrt{d_i}, \ \ \forall i, j \in [n], \quad (8)$$

where $\langle \cdot, \cdot \rangle$ is inner-product operation. Then, a multi-dim source2token self-attention mechanism is used to estimate the contribution of each token to the given task on each feature dimension. It aims at capturing the importance of each token to the entire input sequence w.r.t. the task, i.e., the **global dependency**. The multi-dim extension only linearly increases the memory and computation of source2token self-attention by a multiplicative factor d_h, but is essentially helpful to improve expressive capability in line with prior works (Shen et al., 2018a). Its compatibility function is

$$f^s(k_i) = W^{(s2)}\sigma_m(W^{(s1)}k_i + b^{(s1)}) + b^{(s2)}, \quad (9)$$

where $\forall i \in [n]$, $W^{(s1)} \in \mathbb{R}^{d_a \times d_i}, W^{(s2)} \in \mathbb{R}^{d_h \times d_a}$ are the learnable weights, and $\sigma_m(\cdot)$ is an activation function. The compatibility function used in TSA broadcasts the scalar alignment score $f^t(k_i, q_j) \in \mathbb{R}$ computed by the token2token self-attention to all d_h feature dimensions, and then adds them to the feature-wise score vector

$f^s(k_i) \in \mathbb{R}^{d_h}$ computed by the source2token self-attention. In addition, the positional masks from masked self-attention (in Section 2.2) are also integrated to encode sequential and structural information. These yield following compatibility function of TSA.

$$\left[f^{tsa}(k_i, q_j)\right]_l = \sigma_t\left(f^t(k_i, q_j)\right) + \sigma_s([f^s(k_i)]_l) + M_{i,j}, \tag{10}$$

where $\forall i, j \in [n]$, $\forall l \in [d_h]$. $\sigma_t(\cdot)$ and $\sigma_t(\cdot)$ are two scale functions. They control the way to combine two kinds of scores and their weights.

For each query token q_j, a $\mathrm{softmax}$ function is applied to the alignment scores $[f^{tsa}(k_i, q_j)]_{i=1}^n$ on each feature dimension, resulting in a categorical distribution over all value tokens $[v_i]_{i=1}^n$ based on corresponding key tokens $[k_i]_{i=1}^n$. The probability of token q_j attending to v_i on the l^{th} feature dimension (i.e., $z_l = i$) is

$$p(z_l = i|\boldsymbol{k}, q_j) \triangleq [p_i^j]_l \triangleq \frac{e^{\left[f^{tsa}(k_i, q_j)\right]_l}}{\sum_{g=1}^n e^{\left[f^{tsa}(k_g, q_j)\right]_l}}, \tag{11}$$

where, $\forall i, j \in [n]$, $\forall l \in [d_h]$. TSA outputs a contextual embedding for each input token on every feature dimension as the weighted sum of all the value token embeddings on that dimension, where the weights are provided by the probabilities in Eq.(11). It is the expectation of sampling a value token embeddings on each feature dimension according to the feature-wise probability distribution, i.e.,

$$\boldsymbol{s} \triangleq [s_j]_{j=1}^n, \quad \text{where} \tag{12}$$

$$s_j \triangleq \left[\mathbb{E}_{i \sim p(z_l|\boldsymbol{k}, q_j)}([v_i]_l)\right]_{l=1}^{d_h} = \sum\nolimits_{i=1}^n p_i^j \cdot v_i$$

3.2 Multi-Mask Tensorized Self-Attention (MTSA) Mechanism

Rather than computing attention in the original input space, multi-head attention (Vaswani et al., 2017) projects the input sequence to multiple subspaces, applies attention to the projected embedding in each subspace, and concatenates their outputs at last. The computations associated with multiple heads can be completed in parallel. By using adequate heads, each with a low-dimensional subspace (i.e., the representation dimension for each head is updated by $d_h \leftarrow d_h/h$ where h is the number of head), it reduces parameters and memory/computation cost and increases diversity of the attention. In addition, to encode different kinds of sequential or structural information, multiple different positional masks (e.g., forward, backward and multi-length window) can be further applied to the multiple heads.

The memory-/time-efficiency and expressive power of TSA can be improved by using the combination of the multi-head and multi-mask techniques introduced above. By writing TSA mechanism as a function $\mathrm{TSA}(\boldsymbol{x}, M)$ with input sequence $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$ and a positional mask $M \in \mathbb{R}^{n \times n}$, and the output given by Eq.(12), multi-mask tensorized self-attention (MTSA) produces

$$\boldsymbol{s} = W^{(o)}[H_1; \ldots; H_h], \quad \text{where } H_c = \mathrm{TSA}^c(\boldsymbol{x}, M^c), \tag{13}$$

where $W^{(o)} \in \mathbb{R}^{h \cdot d_h \times h \cdot d_h}$, h is the number of heads, TSA^c denotes the c^{th} parameter-independent TSA block that produces a d_h-dim representation in the c^{th} subspace, M^c represents the positional mask applied to attention in the c^{th} subspace, $[\cdot; \ldots; \cdot]$ denotes a vertical concatenation operation, and $\boldsymbol{s} \in \mathbb{R}^{h \cdot d_h \times n}$ is the output of MTSA. In our experiments, we apply forward mask to half of the heads and apply backward mask to the other half to encode bi-directional order information of the input sequence.

3.3 Computation-Optimized MTSA

Algorithm 1 Multi-Mask Tensorized Self-Attention

```
Input: input sequence $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$, head number $h$, subspace dimension $d_h$, positional masks $\{M^c\}_{c=1}^h$, and weights/biases: $\{W_c^{(t1)}, W_c^{(t2)} \in \mathbb{R}^{d_i \times d_e}, W_c^{(t3)} \in \mathbb{R}^{d_h \times d_e}, W_c^{(s1)} \in \mathbb{R}^{d_a \times d_i}, W_c^{(s2)} \in \mathbb{R}^{d_h \times d_a}$, and $b_c^{(s1)}, b_c^{(s2)}\}_{c=1}^h$, and $W^{(o)}$
Output: contextual embeddings $\boldsymbol{s} = [s_1, \ldots, s_n] \in \mathbb{R}^{h \cdot d_h \times n}$
for all $c = 1, \ldots, h$ do    ▷ Computing $h$-head in parallel
    $\boldsymbol{q^c}, \boldsymbol{k^c}, \boldsymbol{v^c} \leftarrow W_c^{(t1)}\boldsymbol{x}, W_c^{(t2)}\boldsymbol{x}, W_c^{(t3)}\boldsymbol{x}$
    $R_c \leftarrow \frac{(\boldsymbol{k^c})^T \boldsymbol{q^c}}{\sqrt{d_h}}$    ▷ $n \times n$ token2token attention scores
    $S_c \leftarrow W_c^{(s2)}\sigma_m(W_c^{(s1)}\boldsymbol{k^c} + b_c^{(s1)}) + b_c^{(s2)}$    ▷ $d_h \times n$ scores of source2token attention
    $E_c^R \leftarrow \exp(\sigma_t(R_c)) \cdot \exp(M^c)$    ▷ Applying mask $M^c$ to token2token weights
    $E_c^S \leftarrow \exp(\sigma_s(S_c))$; $E_c^X \leftarrow \boldsymbol{v^c} \cdot E_c^S$    ▷ Applying source2token weights $E_c^S$ to $\boldsymbol{v^c}$
    $H_c \leftarrow E_c^X E_c^R / E_c^S E_c^R$    ▷ Applying masked token2token weights $E_c^R$ and normalizing
end for
Return $\boldsymbol{s} \leftarrow W^{(o)}[H_1; \ldots; H_h]$    ▷ Vertical concatenation of the outputs from all $h$ heads
```

As shown in Eq.(10) and Eq.(11), TSA or each head of MTSA needs to compute the attention scores and probabilities as $n \times n \times d_h$ tensors. In accordance with multi-dim self-attention (Shen et al., 2018a), this makes TSA more expressively powerful and improves the final performance for sequence modeling, but terribly leads to memory explosion and computational bottleneck on long sequences with large n and d_h. Fortunately, in MTSA, it is possible to significantly reduce the demand on computations to matrix-only operations by exploring the computational structure.

A memory-optimized and highly-parallelizable computation scheme for MTSA is given in Algorithm 1. For each head, the score matrices of token2token and source2token are computed in steps 3 and 4 respectively. Then, we combine token2token scores with the positional mask to form a new mask in step 5, and compute the $d_h \times n$ output embedding with the weighs from the multi-dim source2token self-attention in step 6. Finally, in step 7, we apply the new mask from step 5 to the weighted embedding from step 6 and complete the normalization. This procedure generates the exactly same output as Eq.(13)but no any tensor operation is incurred.

4 Experiments

We compare MTSA with commonly-used context fusion baselines on several NLP tasks[1]. When addressing a sentence embedding problem, a multi-dim source2token self-attention is applied on the top of context fusion module to produce the sequence embedding. Codes are implemented in Python with Tensorflow and executed on a single NVIDIA GTX 1080Ti graphics card. In addition, data for both time cost and memory consumption are collected under Tensorflow-1.7 with CUDA9 and cuDNN7.

The context fusion baselines include **1) Bi-LSTM** (Graves et al., 2013): 600D bi-directional LSTM consisting of 300D forward plus 300D backward LSTMs, **2) Bi-GRU** (Chung et al., 2014): 600D bi-directional GRU, **3) Multi-CNN** (Kim, 2014): three CNNs with 200D kernels to model 3/4/5-grams respectively, **4) Hrchy-CNN** (Gehring et al., 2017): 3-layer 300D stacked CNN with kernel size 5, gated linear units (Dauphin et al., 2016) and residual connections (He et al., 2016), **5) Multi-head** (Vaswani et al., 2017): 600D multi-head self-attention with 8 heads (75-dim subspace per head) and positional embedding used by Vaswani et al. (2017), **6) DiSA** (Shen et al., 2018a): 600D directional self-attention mechanism consisting of 300D forward and 300D backward masked self-attentions, and **7) Bi-BloSA** (Shen et al., 2018c): 600D bi-directional block self-attention with intra-/inter-block self-attention, aiming to reduce the time and space complexities of multi-dim self-attention by using hierarchical structure.

4.1 Natural Language Inference

Natural language inference (NLI) aims at speculating on the relationship between a premise and a corresponding hypothesis, where the relationship could be *entailment*, *neutral* or *contradiction*. In experiments, we first compare MTSA with other baselines on the Stanford Natural Language Inference (Bowman et al., 2015) (SNLI) dataset.

Following the method of applying *sentence-encoding* model to NLI given by Bowman et al. (2016), two parameter-tied sentence-encoding models are used to generate embeddings for premise and hypothesis, resulting in s^p and s^h respectively. The concatenation of s^p, s^h, $s^p - s^h$ and $s^p \odot s^h$ representing the relationship is passed into a 3-way neural classifier for final prediction.

The experimental results of the models from the official leaderboard, baselines, and MTSA are shown in Table 1. MTSA achieves state-of-the-art performance with less time and memory cost. Compared to the methods from the leaderboard, MTSA outperforms RNN-based encoders (e.g., Residual stacked enc.), RNN+attention encoders (e.g., Deep Gated Attn.) and even parsing trees based encoders (e.g., Gumbel TreeLSTM enc.) by a large margin. Compared to the two competitive self-attention networks with complicated and expensive training computations, MTSA trained in end-to-end manner achieves the same state-of-the-art performance by using much fewer parameters and less computational time.

Compared to baselines, MTSA is $4 \sim 5\times$ faster than RNN-based models and outperforms CNN-based models given a similar number of parameters and computation time. Moreover, compared to the dot-product self-attention (Multi-head), MTSA costs similar time and memory but performs more expressively powerful self-attention, and thus achieves better performance. Furthermore, compared to the multi-dim self-

---

[1] Codes for Experiments are released at `https://github.com/taoshen58/mtsa`.

Model	$\lvert\theta\rvert$	Time/Epoch	Inf. Time	Memory	Train Acc.	Test Acc.
300D SPINN-PI encoders (Bowman et al., 2016)	3.7m				89.2	83.2
600D Bi-LSTM encoders (Liu et al., 2016)	2.0m				86.4	83.3
600D Bi-LSTM enc.+intra-attn (Liu et al., 2016)	2.8m				84.5	84.2
600D Deep Gated Attn. (Chen et al., 2017)	11.6m				90.5	85.5
600D Gumbel TreeLSTM enc. (Choi et al., 2018)	10.0m				93.1	86.0
600D Residual stacked enc. (Nie and Bansal, 2017)	29.0m				91.0	86.0
300D Reinforced SAN (Shen et al., 2018b)	3.1m	404s			92.6	86.3
Distance-based SAN (Im and Cho, 2017)	4.7m	416s			89.6	86.3
Bi-LSTM (Graves et al., 2013)	2.9m	854s	9.1s	942MB	90.4	85.0
Bi-GRU (Chung et al., 2014)	2.5m	850s	9.4s	810MB	91.9	84.9
Multi-CNN (Kim, 2014)	1.4m	137s	1.4s	208MB	89.3	83.2
Hrchy-CNN (Gehring et al., 2017)	3.4m	195s	1.8s	309MB	91.3	83.9
Multi-head (Vaswani et al., 2017)	2.0m	179s	1.5s	466MB	89.6	84.2
DiSA (Shen et al., 2018a)	2.3m	390s	5.2s	6682MB	91.1	85.6
Bi-BloSA (Shen et al., 2018c)	4.1m	303s	3.2s	1600MB	91.6	85.8
MTSA	2.9m	180s	1.6s	558MB	91.8	**86.3**

Table 1: Experimental results for different methods with comparative parameter number on SNLI. $\lvert\theta\rvert$: the number of parameters (excluding word embedding part); Time/Epoch: averaged training time per epoch with batch size 128; Inf. Time: averaged dev inference time with batch size 128; Memory: memory load on synthetic data of sequence length 64 and batch size 64 with back-propagation considered; Train Acc. and Test Acc.: the accuracies on training/test sets. All state-of-the-art methods in leaderboard are listed in Table 1&2 up to Sep. 2018.

Model	SNLI		MultiNLI	
	Dev	Test	Match	Mismatch
BiLSTM w/ Shortcut[a]	–	86.0	74.6	73.6
BiLSTM w/ Gen-Pooling[b]	–	86.6	73.8	74.0
HBMP[c]	–	86.6	73.7	73.0
Transfer + Multi-Head	86.9	86.6	76.3	75.7
Transfer + MTSA	**87.2**	**86.9**	**76.7**	**76.4**

Table 2: Experimental results on *sentence-encoding* based SNLI and MultiNLI benchmark tasks. "*Transfer*" denotes pretrained language model on large corpus for transfer learning, which detailed by Radford et al. (2018). References: [a](Nie and Bansal, 2017), [b](Chen et al., 2018), [c](Talman et al., 2018).

Model	$\lvert\theta\rvert$	Inf. Time	Test Acc.
MTSA	2.9m	1.6	86.3
MTSA w/o fw&bw masks	2.9m	1.6	85.3 (-1.0)
MTSA w/o token2token	2.5m	1.5	85.8 (-0.5)
MTSA w/o source2token	2.5m	1.4	84.9 (-1.4)
MTSA w/o proposed modules	1.8m	1.1	84.3 (-2.0)

Table 3: An ablation study of MTSA on SNLI.

attention (DiSA and Bi-BloSA), MTSA uses much less memory and time but even produces much better prediction quality.

In addition, to further improve the state-of-the-art performance, in contrast to training from scratch, a language model built on the Transformer (Vaswani et al., 2017) unsupervisedly pretrained on large English corpus (detailed by Radford et al. (2018)) is transfered for the baseline and proposed models for *sentence-encoding* based NLI tasks. As shown in Table 2, MTSA integrated with pretrained language model can achieve new state-of-the-art accuracy on both SNLI and Multi-Genre Natural Language Inference (MultiNLI) (Williams et al., 2017)[2] among all *sentence-encoding* models.

An ablation study of MTSA is shown in Table 3 to verify the capability of its each part in context fusion. The results show that token2token (modeling pairwise dependency), source2token (modeling global dependency), and positional masks (encoding sequential information) all contribute important information to sequence modeling, and the contributions are complementary.

4.2 Semantic Role Labeling

To verify the capability of MTSA in generating context-aware representation of each token, we compare it with baselines on semantic role labeling (SRL) task, which aims to tag each token from an input sequence with a label for its semantic role. Particularly, given a sentence, the goal of SRL is to identify the arguments of each target verb into semantic roles, which can benefit many downstream NLP tasks. SRL has two steps:

---

[2] All test results are Evaluated on Kaggle official websites: https://www.kaggle.com/c/multinli-matched-open-evaluation and https://www.kaggle.com/c/multinli-mismatched-open-evaluation

Models	Training	Development				WSJ Test				Brown Test			
	Time	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.
Täckström et al. (2015)		81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	**74.3**	68.6	71.3	39.8
Zhou and Xu (2015)		79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-
He et al. (2017)		81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.8	71.4	72.1	44.8
He et al. (2018)		-	-	-	-	-	-	83.9	-	-	-	73.7	-
Strubell et al. (2018)		-	-	-	-	**84.7**	84.2	84.5	-	73.9	72.4	73.1	-
Bi-LSTM (Graves et al., 2013)	72h	81.8	83.4	82.6	63.3	83.0	84.0	83.5	64.6	72.3	72.8	72.5	46.8
Multi-CNN (Kim, 2014)	19h	75.2	79.6	77.3	53.6	77.3	80.9	79.0	55.5	68.3	70.3	69.3	41.9
Multi-head* (Tan et al., 2017)	20h	82.6	83.6	83.1	65.2	84.5	85.2	**84.8**	66.4	73.5	**74.6**	74.1	48.4
MTSA	20h	**82.8**	**84.4**	**83.6**	**65.4**	84.2	**85.3**	**84.8**	**67.0**	**74.3**	**74.6**	**74.5**	**49.1**

Table 4: Experimental Results of SRL for single models on CoNLL-05 with gold predicates. *Multi-head baseline is equivalent to the model in Tan et al. (2017). For fair comparisons, first, we use the hyper-parameters provided by Tan et al. (2017) instead of tuning them; second, all listed models are independent of external linguistics information, e.g., PoS, dependency parsing.

Model	CR	MPQA	SUBJ	TREC	SST-5
cBoW[a]	79.9	86.4	91.3	87.3	/
Skip-thought[b]	81.3	87.5	93.6	92.2	/
DCNN[c]	/	/	/	93.0	48.5
SRU[d]	84.8(1.3)	89.7(1.1)	93.4(0.8)	93.9(0.6)	/
CNNs[d]	82.2(.2)	88.8(1.2)	92.9(0.7)	93.2(0.5)	/
Bi-LSTM	84.6(1.6)	90.2(0.9)	**94.7(0.7)**	94.4(0.3)	49.9(0.8)
Multi-head	82.6(1.9)	89.8(1.2)	94.0(0.8)	93.4(0.4)	48.2(0.6)
DiSA	84.8(2.0)	90.1(0.4)	94.2(0.6)	94.2(0.1)	51.0(0.7)
Bi-BloSA	84.8(0.9)	90.4(0.8)	94.5(0.5)	94.8(0.2)	50.6(0.5)
MTSA	**84.9(2.4)**	**90.5(0.6)**	94.5(0.6)	**95.3(0.3)**	**51.3(0.7)**

Table 5: Experimental results on five sentence classification benchmarks. References: [a](Mikolov et al., 2013), [b](Kiros et al., 2015), [c](Kalchbrenner et al., 2014), [d](Lei and Zhang, 2017).

1) assigning either a semantic argument or non-argument to a given predicate and 2) labeling a specific semantic role for the identified argument.

We follow the experimental setup in Tan et al. (2017), where the SRL task is treated as a BIO tagging problem. Tan et al. (2017) designed a deep attentive neural net by stacking multi-head self-attention, named as deepatt, to perform context fusion, whose output is then passed to a neural classifier to make the final decision. The results achieved by previous methods, baselines, and MTSA are shown in Table 4, which demonstrates that MTSA achieves new state-of-the-art performance on the CoNLL-05 dataset by costing similar training time as CNN and multi-head self-attention baselines.

4.3 Sentence Classifications

The goal of sentence classification is to predict the correct label for a sentence in various scenarios. We evaluate the models on five sentence classification benchmarks for different NLP tasks, which include **1) CR** (Hu and Liu, 2004): customer reviews of various products to predict whether the review is positive or negative, **2) MPQA** (Wiebe et al., 2005): an opinion polarity detection subtask of the MPQA dataset, **3) SUBJ** (Pang and Lee, 2004): subjectivity dataset where a label indicates whether a sentence is subjective or objective, **4) TREC** (Li and Roth, 2002): question-type classification dataset which classifies the question sentences into six classes, **5) SST-5** (Socher et al., 2013): the Stanford Sentiment Treebank dataset with five sentiment labels. The reported accuracies for CR, MPQA, and SUBJ are the mean of 10-fold cross validation. The accuracies for TREC are the mean of five runs on the dev set, and the accuracies for SST-5 are the mean of five runs on the test set. All standard deviations are shown in parentheses.

The prediction accuracies achieved on these five benchmarks are shown in Table 5. MTSA achieves the best prediction accuracy on CR, MPQA, TREC and SST-5 benchmarks with better time efficiency and a lower memory load.

4.4 Machine Translation

We also evaluate proposed model on WMT 2014 English-German translation task for exhaustive comparisons with multi-head attention. We replace multi-head self-attention modules in the encoder of official Transformer implementation with MTSA module and do not tune the hyperparameters. Although our computation resources is limited, we use two training setups and also introduce t-test to ensure that MTSA consistently outperforms multi-head self-attention in Transformer.

Model	Multi-head (Transformer)	MTSA
Param#	61.38M	61.58M
Setup1	23.64	24.09
	p-value: 0.001 (6 runs)	
Setup2	26.98	27.21
	p-value: 0.080 (3 runs)	

Table 6: Results for the Transformer with either multi-head self-attention or proposed MTSA. The reported BLEU values for Setup 1 and 2 are the mean of 5 and 3 runs respectively.

For **Setup1**, we use default hyperparameter set of *transformer_base_single_gpu* provided by official implementation with 1 × P100 , batch size of 2048 and training step of 250K, and report BLEU value for the last checkpoint. For **Setup2**, we use the hyperparameter set of *transformer_base* with the modification of 1) using 4× instead of 8 × P100, 2) increasing batch size from 4096 to 6144 per GPU, and 3) using training step of 133K.

As shown in Table 6, with small p-value for both training setup 1 and 2, the encoder with MTSA significantly outperforms that with multi-head self-attention, which demonstrates that multi-dim based MTSA modeling both pairwise and global dependencies is more expressive than dot-product based multi-head self-attention. Although the results do not improve state-of-the-art BLEU value of machine translation task, the purpose of this experiment to verify the effectiveness of MTSA in contrast to dot-product based multi-head self-attention is accomplished.

5 Conclusion

In conclusion, MTSA is highly parallelizable with more expressive power since it efficiently captures the pairwise dependency at token level, but delicately models the global dependency at feature level, and distributes computations to multiple heads, each equipped with a distinct positional mask. These lead to a sweet spot of the trade-off between performance and efficiency, and make MTSA as memory-efficient as CNN and scalable to long sequences but outperform previous (and even multi-dim) self-attention mechanisms in terms of prediction quality. The experiments conducted on nine NLP tasks verify that the MTSA can reach state-of-the-art performance with appealing efficiency.

Acknowledgments

This research was funded by the Australian Government through the Australian Research Council (ARC) under grants 1) LP160100630 partnership with Australia Government Department of Health and 2) LP150100671 partnership with Australia Research Alliance for Children and Youth (ARACY) and Global Business College Australia (GBCA). We also acknowledge the support of NVIDIA Corporation and MakeMagic Australia with the donation of GPUs.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *ACL*.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. *arXiv preprint arXiv:1806.09828*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *RepEval@ EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what?s next. In *ACL*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *ACL*.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Research*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018b. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018c. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Aarne Talman, Anssi Yli-Jyra, and Jorg Tiedemann. 2018. Natural language inference with hierarchical bilstm max pooling architecture. *arXiv preprint arXiv:1808.08762*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586*.

Ashish Vaswani, Shazeer, Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL-IJCNLP*.

WiC: the Word-in-Context Dataset
for Evaluating Context-Sensitive Meaning Representations

Mohammad Taher Pilehvar[1,2] and **Jose Camacho-Collados**[3]
[1]DTAL, University of Cambridge, UK
[2]Tehran Institute for Advanced Studies (TeIAS), Tehran, Iran
[3]School of Computer Science and Informatics, Cardiff University, UK
mp792@cam.ac.uk, camachocolladosj@cardiff.ac.uk

Abstract

By design, word embeddings are unable to model the dynamic nature of words' semantics, i.e., the property of words to correspond to potentially different meanings. To address this limitation, dozens of specialized meaning representation techniques such as sense or contextualized embeddings have been proposed. However, despite the popularity of research on this topic, very few evaluation benchmarks exist that specifically focus on the dynamic semantics of words. In this paper we show that existing models have surpassed the performance ceiling of the standard evaluation dataset for the purpose, i.e., Stanford Contextual Word Similarity, and highlight its shortcomings. To address the lack of a suitable benchmark, we put forward a large-scale Word in Context dataset, called WiC, based on annotations curated by experts, for generic evaluation of context-sensitive representations. WiC is released in https://pilehvar.github.io/wic/.

1 Introduction

One of the main limitations of mainstream word embeddings lies in their static nature, i.e., a word is associated with the same embedding, independently from the context in which it appears. Therefore, these embeddings are unable to reflect the dynamic nature of ambiguous words[1], in that they can correspond to different (potentially unrelated) meanings depending on their usage in context (Camacho-Collados and Pilehvar, 2018). To get around this limitation dozens of proposals have been put forward, mainly in two categories: multi-prototype embeddings (Reisinger and Mooney, 2010; Neelakantan et al., 2014; Pelevina et al., 2016), which usually leverage context clustering in order to learn distinct representations for individual meanings of words, and contextualized word embeddings (Melamud et al., 2016; Peters et al., 2018), which instead compute a single dynamic embedding for a given word which can adapt itself to arbitrary contexts for the word.

Despite the popularity of research on these specialised embeddings, very few benchmarks exist for their evaluation. Most works in this domain either perform evaluations on word similarity datasets (in which words are presented in isolation; hence, they are not suitable for verifying the dynamic nature of word semantics) or carry out impact analysis in downstream NLP applications (usually, by taking word embeddings as baseline). Despite providing a suitable means of verifying the effectiveness of the embeddings, the downstream evaluation cannot replace generic evaluations as it is difficult to isolate the impact of embeddings from many other factors involved, including the algorithmic configuration and parameter setting of the system. To our knowledge, the Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012) is the only existing benchmark that specifically focuses on the dynamic nature of word semantics.[2] In Section 4 we will explain the limitations of this dataset for the evaluation of recent work in the literature.

In this paper we propose WiC, a novel dataset that provides a high-quality benchmark for the evaluation of context-sensitive word embeddings. WiC provides multiple interesting characteristics: (1) it is suitable for evaluating a wide range of techniques, including contextualized word and sense representation and word sense disambiguation; (2) it is framed as a binary classification dataset, in which, unlike SCWS, identical words are paired with each other (in different con-

---

[1] Ambiguous words are important as they constitute the most frequent words in a natural language (Zipf, 1949).

[2] With a similar goal in mind but focused on hypernymy, Vyas and Carpuat (2017) developed a benchmark to assess the capability of automatic systems to detect hypernymy relations in context.

Proceedings of NAACL-HLT 2019, pages 1267–1273
Minneapolis, Minnesota, June 2 - June 7, 2019.

F	There's a lot of trash on the *bed* of the river — I keep a glass of water next to my *bed* when I sleep
F	*Justify* the margins — The end *justifies* the means
T	*Air* pollution — Open a window and let in some *air*
T	The expanded *window* will give us time to catch the thieves — You have a two-hour *window* of clear weather to finish working on the lawn

Table 1: Sample positive (T) and negative (F) pairs from the WiC dataset (target word in *italics*).

texts); hence, a context-insensitive word embedding model would perform similarly to a random baseline; and (3) it is constructed using high quality annotations curated by experts.

2 WiC: the Word-in-Context dataset

We frame the task as binary classification. Each *instance* in WiC has a target word w, either a verb or a noun, for which two contexts, c_1 and c_2, are provided. Each of these contexts triggers a specific meaning of w. The task is to identify if the occurrences of w in c_1 and c_2 correspond to the same meaning or not. Table 1 lists some examples from the dataset. In what follows in this section, we describe the construction procedure of the dataset.

2.1 Construction

Contextual sentences in WiC were extracted from example usages provided for words in three lexical resources: (1) WordNet (Fellbaum, 1998), the standard English lexicographic resource; (2) VerbNet (Kipper-Schuler, 2005), the largest domain-independent verb-based resource; and (3) Wiktionary[3], a large collaborative-constructed online dictionary. We used WordNet as our core resource, exploiting BabelNet's mappings (Navigli and Ponzetto, 2012) as a bridge between Wiktionary and VerbNet to WordNet. Lexicographer examples constitute a reliable base for the construction of the dataset, as they are curated in a way to be clearly distinguishable across different senses of a word.

2.1.1 Compilation

As explained above, the dataset is composed of instances, each of which contain a target word and two examples containing the target word. An instance can be either positive or negative, depending on whether the corresponding c_1 and c_2 are listed for the same sense of w in the target resource. In order to compile the dataset, we first obtained all the possible positive and negative instances from all resources, with the only condition of the surface word form occurring in both c_1 and c_2.[4] The total number of initial examples extracted from all resources at this stage were 23,949, 10,564 and 636 for WordNet, Wiktionary and VerbNet, respectively. We first compiled the test and development sets with two constraints: (1) not having more than three instances for the same target word, and (2) not having repeated contextual sentences across instances. These constraints were enforced to have a diverse and balanced set which covers as many unique words as possible. With all these constraints in mind, we set apart 1,600 and 800 instances for the test and development sets, respectively. We ensured that all the splits were balanced for their positive and negative examples. The remaining instances whose examples did not overlap with test and development formed our initial training dataset.

Semi-automatic check. Even though very few in number, all resources (even exprt-based ones) contain errors such as incorrect part-of-speech tags or ill-formed examples. Moreover, the extraction of examples and the mappings across resources were not always accurate. In order to have as few resource-specific and mapping errors as possible, all training, development and test sets were semi-automatically post-processed, either with small fixes whenever possible or by removing problematic instances otherwise.

2.1.2 Pruning

WordNet is known to be a fine-grained resource (Navigli, 2006). Often, different senses of the same word are hardly distinguishable from one another even for humans. For example, more than 40 senses are listed for the verb *run*, with many of them corresponding to similar concepts, e.g., "move fast", "travel rapidly", and "run with the ball". In order to avoid this high-granularity, we performed an automatic pruning of the resource, removing instances with subtle sense distinctions. Sense clustering is not a very well-defined problem (McCarthy et al., 2016) and there are different strategies to perform this sense distinction (Snow et al., 2007; Pilehvar et al., 2013; Mancini et al., 2017). We adopted a simple strategy and

---

[3] https://www.wiktionary.org/

[4] Given that WordNet provides examples for synsets (rather than word senses), a target word (sense) might not occur in all the examples of its corresponding synset.

removed all pairs whose senses were first degree connections in the WordNet semantic graph, including sister senses, and those which belonged to the same supersense, i.e. sense clusters from the Wordnet lexicographer files[5]. There are a total of 44 supersenses in WordNet, comprising semantic categories such as *shape*, *substance* or *event*. This coarsening of the WordNet sense inventory has been shown particularly useful in downstream applications (Rüd et al., 2011; Severyn et al., 2013; Flekova and Gurevych, 2016; Pilehvar et al., 2017). In the next section we show that the pruning resulted in a significant boost in the clarity of the dataset.

2.2 Quality check

To verify the quality and the difficulty of the dataset and to estimate the human-level performance upperbound, we randomly sampled four sets of 100 instances from the test set, with an overlap of 50 instances between two of the annotators. Each set was assigned to an annotator who was asked to label each instance based on whether they thought the two occurrences of the word referred to the same meaning or not.[6] The annotators were not provided with knowledge from any external lexical resource (such as WordNet). Specifically, the number of senses and the sense distinctions of the word (in the target sense inventory) were unknown to the annotators.

We found the average human accuracy on the dataset to be 80.0% (individual scores of 79%, 79%, 80% and 82%). We take this as an estimation of the human-level performance upperbound of the dataset. For the overlapping section, we computed the agreement between the two annotators to be 80%. Note that the annotators were not provided with sense distinctions to resemble the more difficult scenario for unsupervised models (which do not benefit from sense-based knowledge resources). Having access to sense definitions/distinctions would have substantially raised the performance bar.

Impact of pruning. To check the effectiveness of our pruning strategy, we also sampled a set of 100 instances from the batch of instances that were pruned from the dataset. Similarly, the annotators were asked to independently label instances in the set. We computed the average accuracy on this set to be 57% (56% and 58%), which is substantially lower than that for the final pruned set (i.e. 80%). This indicates the success of our pruning strategy in improving the semantic clarity of the dataset.

Split	Instances	Nouns	Verbs	Unique words
Training	5,428	49%	51%	1,256
Dev	638	62%	38%	599
Test	1,400	59%	41%	1,184

Table 2: Statistics of different splits of WiC.

2.3 Statistics

Table 2 shows the statistics of the different splits of WiC. The test set contains a large number of unique target words (1,256), reflecting the variety of the dataset. The large training split of 5,428 instances makes the dataset suitable for various supervised algorithms, including deep learning models. Only 36% of the target words in the test split overlap with those in the training, with no overlap of contextual sentences across the splits. This makes WiC extremely challenging for systems that heavily rely on pattern matching.

3 Experiments

We experimented with recent multi-prototype and contextualized word embedding techniques. Evaluation of other embedding models as well as word sense disambiguation systems is left for future work.

Contextualized word embeddings. One of the pioneering contextualized word embedding models is **Context2Vec** (Melamud et al., 2016), which computes the embedding for a word in context using a multi-layer perceptron which is built on top of a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) language model. We used the 600-d UkWac pre-trained models[7]. **ELMo** (Peters et al., 2018) is a character-based model which learns dynamic word embeddings that can change depending on the context. ELMo embeddings are essentially the internal states of a deep LSTM-based language model, pre-trained on a large text corpus. We used the 1024-d pre-trained models[8] for two configurations: ELMo$_1$, the first LSTM hidden state, and ELMo$_3$, the weighted sum of the

---

[5] wordnet.princeton.edu/documentation/lexnames5wn

[6] Annotators were not lexicographers. To make the task more understandable, they were asked if in their opinion the two words would belong to the same dictionary entry or not.

[7] https://github.com/orenmel/context2vec

[8] https://www.tensorflow.org/hub/modules/google/elmo/1

3 layers of LSTM. A more recent contextualized model is **BERT** (Devlin et al., 2019). The technique is built upon earlier contextual representations, including ELMo, but differs in the fact that, unlike those models which are mainly unidirectional, BERT is bidirectional, i.e., it considers contexts on both sides of the target word during representation. We experimented with two pre-trained BERT models: *base* (768 dimensions, 12 layer, 110M parameters) and *large* (1024 dimensions, 24 layer, 340M parameters).[9] Around 22% of the pairs in the test set had at least one of their target words not covered by these models. For such out-of-vocabulary cases, we used BERT's default tokenizer to split the unknown word to subwords and computed its embedding as the centroid of the corresponding subwords' embeddings.

Multi-prototype embeddings. We experiment with three recent techniques that release 300-d pre-trained multi-prototype embeddings[10]. **JBT**[11] (Pelevina et al., 2016) induces different senses by clustering graphs constructed using word embeddings and computes embedding for each cluster (sense). **DeConf**[12] (Pilehvar and Collier, 2016) exploits the knowledge encoded in WordNet. For each sense, it extracts from the resource the set of semantically related words, called sense biasing words, which are in turn used to compute the sense embedding. **SW2V**[13] (Mancini et al., 2017) is an extension of Word2Vec (Mikolov et al., 2013a) for jointly learning word and sense embeddings, producing a shared vector space of words and senses as a result. For these three methods we follow the disambiguation strategy suggested by Pelevina et al. (2016): for each example we retrieve the closest sense embedding to the context vector, which is computed by averaging its contained words' embeddings.

Sentence-level baselines. We also report results for two baseline models which view the task as context (sentence) similarity. The **BoW** system views the sentence as a bag of words and computes a simple embedding as average of its words. The system makes use of Word2vec (Mikolov et al., 2013b) 300-d embeddings pre-trained on the Google News corpus. **Sentence LSTM** is another baseline, which differently from the other models, does not obtain explicit encoded representations of the target word or sentence. The system has two LSTM layers with 50 units, one for each context side, which concatenates the outputs and passes that to a feedforward layer with 64 neurons, followed by a dropout layer at rate 0.5, and a final one-neuron output layer of sigmoid activation.

	MLP	**Threshold**
Contextualized word-based models		
Context2vec	57.9 ± 0.9	59.7
ElMo$_1$	56.4 ± 0.6	57.1
ElMo$_3$	57.2 ± 0.8	56.3
BERT$_{large}$	57.4 ± 1.0	**63.8**
BERT$_{base}$	**60.2** ± 0.4	63.6
Multi-prototype models		
DeConf*	52.4 ± 0.8	62.1
SW2V*	54.1 ± 0.5	59.1
JBT	54.1 ± 0.6	54.5
Sentence-level baselines		
BoW	54.2 ± 1.3	61.0
Sentence LSTM	53.1 ± 0.9	

Table 3: Accuracy % performance of different models on the WiC dataset. The estimated (human-level) performance is 80.0 (cf. Section 2.2) and a random baseline would perform at 50.0. Systems marked with * make use of external lexical resources.

We used two simple binary classifiers in our experiments on top of all comparison systems (except for the LSTM baseline). **MLP**: a simple dense network with 100 hidden neurons (ReLU activation), and one output neuron (sigmoid activation), tuned on the development set (batch size: 32; optimizer: Adam; loss: binary crossentropy). Given the stochasticity of the network optimizer, we report average results for five runs (± standard deviation). **Threshold**: a simple threshold-based classifier based on the cosine distance of the two input vectors, tuned with step size 0.02 on the development set.

3.1 Results

Table 3 shows the results on WiC. In general, the dataset proves to be very difficult for all the techniques, with the best model, i.e., BERT$_{large}$, providing around 14% absolute improvement over a random baseline. Among the two classifiers, the simple threshold-based strategy, which computes

---

[9] https://github.com/google-research/bert/blob/master/

[10] *Multi-prototype* embeddings are also referred to as *sense* embeddings in the literature.

[11] https://github.com/uhh-lt/sensegram

[12] https://pilehvar.github.io/deconf/

[13] http://lcl.uniroma1.it/sw2v

the cosine distance between the two encodings, proves to be more efficient than the MLP network which might not be suitable for this setting with small amount of training data. The 16.2% absolute accuracy difference between human-level upperbound and state-of-the-art performance suggests, however, a challenging dataset and encourages future research in context-sensitive word embeddings to leverage WiC in their evaluations.

Among the contextualized word-based models, after BERT, Context2vec provides more competitive results on the dataset. However, surprisingly, neither ELMo nor Context2vec are able to improve over the simple sentence BoW baseline (which also outperforms the sentence LSTM baseline) using the threshold strategy. This raises a question about the ability of these models in capturing fine-grained semantics of words in various contexts. Finally, as far as multi-prototype techniques are concerned, DeConf is the best performer. We note that DeConf indirectly benefits from sense-level information from WordNet encoded in its embeddings. The same applies to SW2V, which leverages knowledge from a significantly larger lexical resource, i.e., BabelNet (Navigli and Ponzetto, 2012).

4 Related work

The Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012) comprises 2003 word pairs and is analogous to standard word similarity datasets, such as RG-65 (Rubenstein and Goodenough, 1965) and SimLex (Hill et al., 2015), in which the task is to automatically estimate the semantic similarity of word pairs. Ideally, the estimated similarity scores should have high correlation with those given by human annotators. However, there is a fundamental difference between SCWS and other word similarity datasets: each word in SCWS is associated with a context which triggers a specific meaning of the word. The unique property of the dataset makes it a suitable benchmark for multi-prototype and contextualized word embeddings. However, in the following, we highlight some of the limitations of the dataset which hinder its suitability for evaluating existing techniques.

Inter-rater agreement (IRA) is widely accepted as a metric to assess the annotation quality of a dataset. The metric reflects the homogeneity of ratings which is expected to be high for a well-defined task and a qualified set of annotators. For each word pair in SCWS ten scores were obtained through crowdsourcing. We computed the pairwise IRA to be 0.35 (in terms of Spearman ρ correlation) which is a very low figure. The mean IRA (between each annotator and the average of others), which can be taken as a human-level performance upperbound, is 0.52. Moreover, most of the instances in SCWS have context pairs with different target words.[14] This makes it possible to test context-independent models, which only considers word pairs in isolation, on the dataset. Importantly, such a context-independent model can easily surpass the human-level performance upperbound. For instance, we computed the performance of the Google News Word2vec pre-trained word embeddings (Mikolov et al., 2013b) on the dataset to be 0.65 (ρ), which is significantly higher than the optimistic IRA for the dataset. In fact, Dubossarsky et al. (2018) showed how the reported high performance of multi-prototype techniques in this dataset was not due to an accurate sense representation, but rather to a subsampling effect, which had not been controlled for in similarity datasets. In contrast, a context-insensitive word embedding model would perform no better than a random baseline on our dataset.

5 Conclusions

In this paper we have presented a benchmark for evaluating context-sensitive word representations. The proposed dataset, WiC, is based on lexicographic examples, which constitute a reliable basis to validate different models in their ability to perceive and discern different meanings of words. We tested some of the recent state-of-the-art contextualized and multi-prototype embedding models on our dataset. The considerable gap between the performance of these models and the human-level upperbound suggests ample room for future work on modeling the semantics of words in context.

Acknowledgments

We would like to thank Luis Espinosa-Anke and Carla Pérez-Almendros for their help with the manual evaluation and Kiamehr Rezaee for running the BERT experiments.

---

[14] Only 8% (12% if ignoring PoS) of SCWS pairs are identical but their assigned scores (by average 6.8) are substantially higher than the dataset average of 3.6 on a [0,10] scale.

References

Jose Camacho-Collados and Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, Minneapolis, United States.

Haim Dubossarsky, Eitan Grossman, and Daphna Weinshall. 2018. Coming to your senses: on controls and evaluation sets in polysemy research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1732–1740, Brussels, Belgium.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *Proceedings of ACL*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

Karin Kipper-Schuler. 2005. VerbNet: A broad-coverage, comprehensive verb lexicon. *Ph.D. thesis*.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of CoNLL*, pages 100–111, Vancouver, Canada.

Diana McCarthy, Marianna Apidianaki, and Katrin Erk. 2016. Word sense clustering and clusterability. *Computational Linguistics*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Roberto Navigli. 2006. Meaningful clustering of senses helps boost Word Sense Disambiguation performance. In *Proceedings of COLING-ACL*, pages 105–112, Sydney, Australia.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*, pages 1059–1069, Doha, Qatar.

Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, New Orleans, LA, USA.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proceedings of ACL*, Vancouver, Canada.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of EMNLP*, pages 1680–1690, Austin, TX.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: a Unified Approach for Measuring Semantic Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1341–1351, Sofia, Bulgaria.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of ACL*, pages 109–117.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of ACL-HLT*, pages 965–975, Portland, Oregon, USA.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *Proceedings of ACL (2)*, pages 714–718, Sofia, Bulgaria.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to merge word senses. In *Proceedings of EMNLP*, pages 1005–1014, Prague, Czech Republic.

Yogarshi Vyas and Marine Carpuat. 2017. Detecting asymmetric semantic relations in context: A case-study on hypernymy detection. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 33–43. Association for Computational Linguistics.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA.

Does My Rebuttal Matter? Insights from a Major NLP Conference

Yang Gao*, **Steffen Eger***,
Ilia Kuznetsov, **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab
Department of Computer Science
Technische Universität Darmstadt
`www.ukp.tu-darmstadt.de`

Yusuke Miyao
Department of Computer Science
Graduate School of Information
Science and Technology
University of Tokyo
`yusuke@is.s.u-tokyo.ac.jp`

Abstract

Peer review is a core element of the scientific process, particularly in conference-centered fields such as ML and NLP. However, only few studies have evaluated its properties empirically. Aiming to fill this gap, we present a corpus that contains over 4k reviews and 1.2k author responses from ACL-2018. We quantitatively and qualitatively assess the corpus. This includes a pilot study on paper weaknesses given by reviewers and on quality of author responses. We then focus on the role of the rebuttal phase, and propose a novel task to predict after-rebuttal (i.e., final) scores from initial reviews and author responses. Although author responses do have a marginal (and statistically significant) influence on the final scores, especially for borderline papers, our results suggest that a reviewer's final score is largely determined by her initial score and the distance to the other reviewers' initial scores. In this context, we discuss the *conformity bias* inherent to peer reviewing, a bias that has largely been overlooked in previous research. We hope our analyses will help better assess the usefulness of the rebuttal phase in NLP conferences.

1 Introduction

Peer review is a widely adopted quality control mechanism in which the value of scientific work is assessed by several reviewers with a similar level of competence. Although peer review has been at the core of the scientific process for at least 200 years (Birukou et al., 2011), it is also a subject of debate: for instance, it has been found that peer reviewing can hardly recognize prospectively well-cited papers or major flaws (Ragone et al., 2013). Further, Langford and Guzdial (2015) observed substantial disagreement between two sets of reviews on the same set of submissions for the prestigious Conference on Neural Information Processing Systems (NeurIPS) 2014.

The *rebuttal* phase plays an important role in peer reviewing especially in top-tier conferences in Natural Language Processing (NLP). It allows authors to provide *responses* to address the criticisms and questions raised in the reviews and to defend their work. Although there is evidence that reviewers do update their evaluations after the rebuttal phase[1], it remains unclear what causes them to do so, and especially, whether they react to the author responses per se, or rather adjust to the opinions of their co-reviewers ("*peer pressure*").

In order to obtain further insights into the reviewing process, especially regarding the role of the rebuttal phase in peer reviewing, in this work we present and analyze a review corpus of *the 56th Annual Meeting of the Association for Computational Linguistics (ACL-2018)*. Every reviewer/author was asked whether she consented to freely using her review/author-response for research purposes and publishing the data under an appropriate open-source license within at earliest 2 years from the acceptance deadline (see supplementary material for the original consent agreement). 85% reviewers and 31% authors have consented to sharing their data. The corpus comprises over 4k reviews (including review texts and scores) and 1.2k author responses. Uniquely, the corpus includes both before- and after-rebuttal reviews for both accepted and rejected papers, making it a highly valuable resource for the community to study the role of the rebuttal phase. The corpus as well as our source code and annotations are publicly avail-

---

* Equal contribution.

[1] For example, see discussions at `https://naacl2018.wordpress.com/2018/02/04/analysis-of-long-paper-reviews/` and `https://acl2017.wordpress.com/2017/03/27/author-response-does-it-help/`.

Proceedings of NAACL-HLT 2019, pages 1274–1290
Minneapolis, Minnesota, June 2 - June 7, 2019.

able at `https://github.com/UKPLab/naacl2019-does-my-rebuttal-matter`.

Our contributions are threefold. First, in §3, we assess the corpus both *quantitatively* (e.g., correlating Overall Score with aspect scores such as Originality and Readability) and *qualitatively* (e.g., identifying key terms that differentiate "good" from "bad" author responses, annotating paper weaknesses given by reviewers, and rating the quality of individual author responses). Second, in §4, we develop a model to predict whether a reviewer will increase/decrease/keep her initial scores after the rebuttal. We do so in order to analyze and disentangle the sources of review updates during the rebuttal stage. We find that factoring in the author responses only marginally (but statistically significantly) improves the classification performance, and the score update decision is largely determined by the scores of peer reviewers. Third, in §5, we discuss multiple types of biases in the score update process, some of which potentially undermine the 'crowd-wisdom' of peer reviewing.

2 Related Work

Several sources provide review and author response data. Since 2013, the NeurIPS main conference publishes the reviews of accepted papers and their author responses. However, these reviews only include the review texts for after-rebuttal reviews. Also, reviews of rejected papers and author responses are not published. Some Machine Learning and NLP conferences, for instance ICLR (International Conference on Learning Representations) and ESWC (Extended Semantic Web Conference), adopt the *open review* model, which allows anyone to access the reviews and author responses. However, most major NLP conferences have not yet adopted the open-review model, and the reviews and author responses in open- and non-open-review venues are likely to be different because people behave differently when their actions are observable (Andreoni and Petrie, 2004).

Kang et al. (2018) provide a corpus of computer science papers from ACL, NeurIPS, CoNLL (The SIGNLL Conference on Computational Natural Language Learning) and ICLR, together with the accept/reject decisions and reviews for a subset of the papers. They suggest several tasks with respective baselines, such as predicting review aspect scores from paper- and review-based features. However, their corpus contains neither before-rebuttal reviews nor author responses, and the size of their review set from NLP conferences (only 275 reviews from ACL-2017 and 39 reviews from CoNLL-2016) is much smaller than ours.

Hua et al. (2019) compile a corpus consisting of 14.2k reviews from major NLP and machine learning conferences. In addition, they annotate 10k argumentative propositions in 400 reviews, and train state-of-the-art proposition segmentation and classification models on the data. But similar to Kang et al. (2018), their corpus does not include before-rebuttal reviews or author responses.

Several publications specifically address the peer reviewing process. Falkenberg and Soranno (2018) investigate what makes a paper review helpful to a journal editor within a specific scientific field. Birukou et al. (2011) and Kovanis et al. (2017) discuss the shortcomings of the review process in general, such as its inability to detect major flaws in papers (Godlee et al., 1998) and its ineffectiveness in selecting papers that will have high citation counts in the future (Ragone et al., 2013). They discuss alternatives to the standard review process such as crowd-based reviewing and review-sharing, i.e., resubmitting a rejected work to another venue along with its past reviews. Ragone et al. (2013) analyze peer reviews across nine anonymized computer science conferences and, among others, identify reviewer biases of multiple types (affiliation, gender, geographical, as well as rating bias: consistently giving higher or lower scores than other reviewers) and propose means for debiasing reviews. However, none of these works quantitatively measures the influence of the rebuttal phase on the final review scores, nor do they provide any corpora facilitating such studies.

Our work is also related to *meta science*, which studies the scientific process in general, i.e., how scientific information is created, verified and distributed (cf. Fortunato et al. (2018)). In this context, our work can be seen as a study on how scientific information is verified.

3 Review Corpus

ACL-2018 adopts a reviewing workflow similar to that of other major NLP conferences: after paper assignment, typically three reviewers evaluate a paper independently. After the rebuttal, reviewers can access the author responses and other peer reviews, and discuss their viewpoints. Reviews

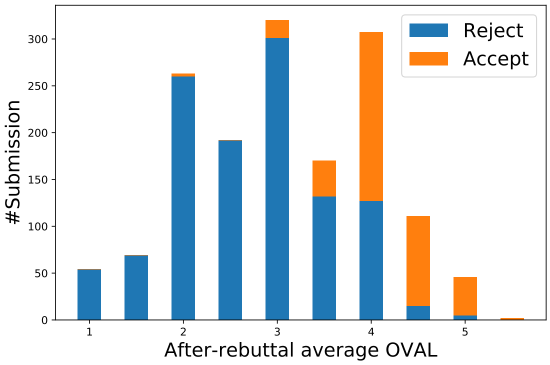
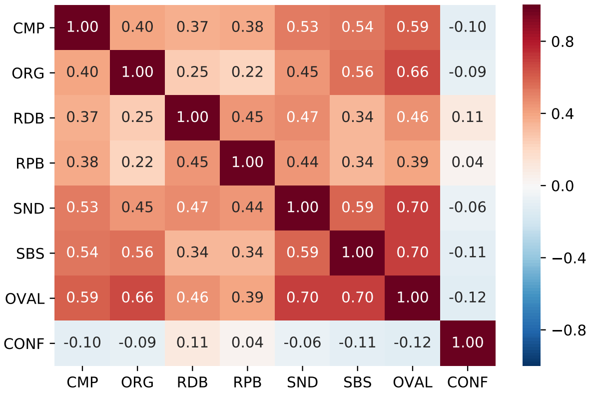


Figure 1: Distribution of accept/reject decisions.

Category	Size
Before-rebuttal reviews	3875 (1213 reviewers, 1538 submissions)
After-rebuttal reviews	4054 (1275 reviewers, 1542 submissions)
Author responses	1227 (499 submissions)

Table 1: Statistics of the ACL-2018 corpus. Some reviewers submitted their reviews after the rebuttal started, hence the size of the after-rebuttal reviews is larger than that of the before-rebuttal reviews.



	CMP	ORG	RDB	RPB	SND	SBS	OVAL	CONF
CMP	1.00	0.40	0.37	0.38	0.53	0.54	0.59	-0.10
ORG	0.40	1.00	0.25	0.22	0.45	0.56	0.66	-0.09
RDB	0.37	0.25	1.00	0.45	0.47	0.34	0.46	0.11
RPB	0.38	0.22	0.45	1.00	0.44	0.34	0.39	0.04
SND	0.53	0.45	0.47	0.44	1.00	0.59	0.70	-0.06
SBS	0.54	0.56	0.34	0.34	0.59	1.00	0.70	-0.11
OVAL	0.59	0.66	0.46	0.39	0.70	0.70	1.00	-0.12
CONF	-0.10	-0.09	0.11	0.04	-0.06	-0.11	-0.12	1.00

Figure 2: Score correlation matrix.

include both *scores* (Overall Score `OVAL`, Reviewer Confidence `CONF`, Soundness `SND`, Substance `SBS`, Originality `ORG`, Meaningful Comparison `CMP` and Readability `RDB`) and free-text comments. `OVAL` are integers in $[1, 6]$, while all other scores are integers in $[1, 5]$.

We first provide an overview of our corpus in §3.1, and then present analyses for the reviews and author responses in §3.2 and §3.3, respectively.

3.1 Overview of the Corpus

The corpus has three parts: the before-rebuttal reviews (including review texts and scores), the after-rebuttal reviews, and the author responses. The corpus does not contain the submissions, nor the information of the reviewers, e.g., their gender, country, affiliation or seniority level; nevertheless, we perform some analyses on the submissions and the reviewers' information and present the statistics in the supplementary material.

Basic statistics of our corpus are summarized in Table 1. 1542 submissions (1016 long, 526 short) have at least one review opted in. 1538 submissions have at least one before- and one after-rebuttal review opted in. Among the 1542 submissions, 380 submissions (24.6%) were accepted: 255 long, 125 short, and the remaining 1162 were rejected: 761 long, 401 short. The distribution of their accept/reject decisions is illustrated in Fig. 1.

3.2 Reviews

Score Correlation. In line with Kang et al. (2018), we first assess the impact of individual aspect scores on the overall score by measuring their Pearson correlation, illustrated in Fig. 2. We find that `OVAL` is most strongly correlated with `SND` and `SBS`, followed by `ORG` and `CMP`. `CONF` shows weak positive correlation to `RDB`: the less readable a paper is, the less confident the reviewers will be. Note that our correlation results are different from those reported by Kang et al. (2018), who report that the `OVAL` has low Pearson correlation with `SND` (0.01) and `ORG` (0.08). While the differences might be caused by a variation in aspect definitions, we believe that our estimate is more reliable as the dataset analyzed in Kang et al. (2018) is substantially smaller than ours.

Review Texts. ACL-2018 adopts the novel *argument-based* review template, which asks reviewers to provide *positive* and *negative* arguments for and against the submission, respectively. In addition, reviewers can also list their questions to the authors in the *questions* section of the review template. Most reviewers made good use of the argument-based template: among the 4054 after-rebuttal reviews, 3258 (80.4%) provide positive arguments, 3344 (82.5%) provide negative arguments, and 1627 (40.1%) provide questions. The number and length of arguments/questions are summarized in Table 2.

Score Changes. Table 3 shows how many reviews increase (INC), decrease (DEC) or keep (KEEP) their overall scores after rebuttal. For

Component	Number	Length (token)
Pos. Arg.	1.92±1.31	22±17
Neg. Arg.	2.38±1.56	56±53
Questions	0.87±1.36	35±31

Table 2: Numbers and lengths of different components in each review (mean±standard deviation).

Type	Num.	#Paper	Acpt.%	$\Delta_{\texttt{OVAL}}$
INC	245	227	49.8	2.65 → 3.76
DEC	248	221	7.2	4.17 → 3.04
KEEP	3377	1119	22.8	3.13 → 3.13
Total	3870	1538	24.7	3.17 → 3.17

Table 3: Statistics of different types of reviews.

the 227 papers that receive at least one INC review (first row in Table 3), their acceptance rate is 49.8%, much higher than those 221 papers with at least one DEC (7.2%) and those 1119 papers with no score update (22.8%). Hence, the score update has a large impact on the final accept/reject decision. Note that 29 papers receive both INC and DEC reviews, of which five were accepted finally.

Fig. 3 summarizes the `OVAL` updates. Most reviewers stick to their initial scores after rebuttal. For those who update, the score change usually amounts to just one point in absolute value. However, most updates happen in the borderline area (overall score 3-4) where the score update might influence the overall acceptance decision. We find that the changes in aspect scores occur much less often than the changes in overall scores: only 5% of the reviews have any of the aspect scores updated after rebuttal, and only 1% of the reviews change the confidence value. In these rare cases, aspect score changes are consistent with their `OVAL` changes, e.g., if the `OVAL` increases, no aspect score decreases.

Submission Time. Fig. 4 illustrates the distribution of the first submission time of reviews. 51.6% reviews were submitted within the last 3 days before the deadline. We also find that the mean submission time of the INC reviews is around 20 hours earlier than that of the DEC reviews, and the difference is statistically significant (p-value 0.009, double-tailed t-test). Moreover, we find that submission time is weakly positively correlated with initial score, which means that reviewers who submit early have slightly lower scores on average, which may explain their tendency to increase their scores later on, given our results in §4.

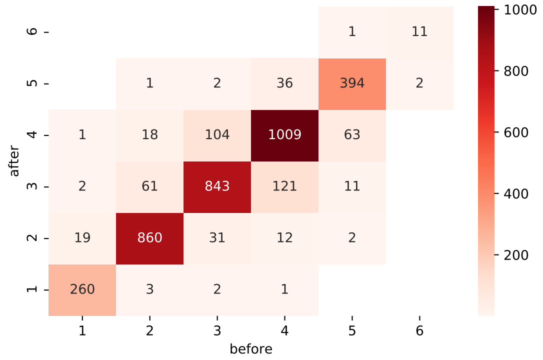


Figure 3: Before vs after rebuttal `OVAL`.

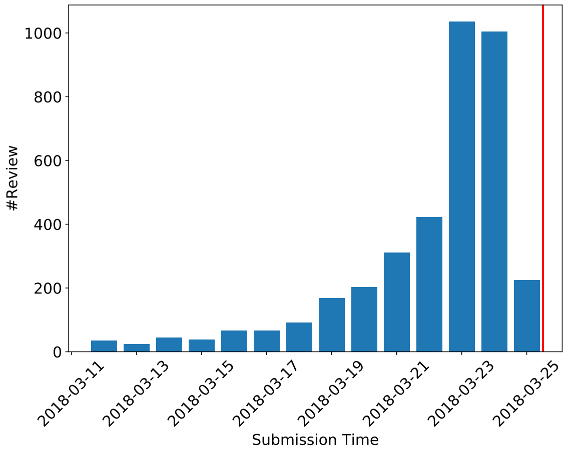


Figure 4: Distribution of review submission time. The review submission deadline (26th March 2018) is marked as the red vertical line towards the right end.

A reason why early submitters have lower scores may be that it takes less time to reject a paper, as the majority of papers is rejected anyway.

Criticism in Reviews. To study the most common paper weaknesses identified in reviews, we manually assess about 300 weakness statements from the reviews. Table 4 summarizes the main results, excluding concerns about *Technical weaknesses*. In our sample, most weaknesses refer to *Evaluation & Analysis*, i.e., criticize the lack of: error analysis, ablation tests, significance tests, human evaluations (opposed to indirect measures such as BLEU) and strong baselines as well as insufficient comparisons (either external or internal). Other frequent targets of criticism are *Writ-*

Eval	Writing	Nov	Data	Motivation
28%	18%	8%	8%	5%

Table 4: Frequent weakness types identified in reviews.

Type	Num.	Length (token)
iResp	100	373±191
dResp	80	260 ±140
kResp	1047	297±182
Total	1227	300±181

Table 5: Statistics of author responses (mean±standard deviation for Length).

ing quality, as well as *Data*: e.g., too few datasets being used (only English data or only synthetic data), missing agreement scores for newly labeled datasets, and resources not being publicly available. Reviewers also criticize the lack of *Novelty* and proper *Motivation* of approaches.

3.3 Author Responses

We align author responses with their corresponding reviews (if opted in), and term the author responses corresponding to INC, DEC and KEEP reviews as iResp, dResp and kResp, respectively. Table 5 presents an overview on these groups.

To qualitatively compare iResps and dResps, we extract and rank n-grams in both iResps and dResps according to the *log-likelihood ratio* (LLR) statistic (Dunning, 1993), treating iResps and dResps as two corpora[2]. The results are reported in Table 6. We find both iResps and dResps express gratitude and promise revisions in the final versions, but iResps address review questions and criticisms by referring back to certain lines and tables in the original paper while dResps fail to do so. We revisit these differences in §4.2.

iResps	dResps
the final version	thanks for your
in line DIGIT	DIGIT reply to
in table DIGIT	to question DIGIT
in the final	will add more
for example the	DIGIT we will
in order to	thank the reviewer
final version EOS	argument DIGIT reply
due to space	due to the
for your comments	paper is accepted
camera ready version	the revised version
DIGIT and DIGIT	we agree that

Table 6: Top trigrams based on LLR ranking. All digits were replaced by DIGIT. EOS: end of sentence.

To gain further insights, we analyze the *quality* of the author responses to the 300 weakness statements from Table 4. We advertised no formal definition of quality, and assessed a subjective, perceived quality score in a range from 1 (low) to 10 (high). We find that the *weak* author responses (scores 1-3) are substantially shorter than the *strong* ones (scores 8-10): the average token number in weak and strong responses are 53 and 90, respectively. Responses evaluated as weak are less specific and make vague promises (*"Thanks for the suggestion, we will try this in the camera-ready"*), off-topic (addressing different points than those raised by the reviewer), or apologetic (*"the deadline was very close", "our native language is not English"*). Interestingly, with some exceptions (*"We take your review as an example of bad writing"*), the weak responses are usually polite and admit the weaknesses suggested by the reviewers, but they tend not to detail how they would address the weaknesses. Strong responses, in contrast, are specific (referring to specific line numbers in the submission, as well as providing numerical values), detailed, longer, and often do not agree with the criticism, but explain why the reviewer's requirement is hard to meet or beyond the scope of the work.

4 After-Rebuttal Score Prediction

To measure the influence of different factors on the score update decisions, we propose and study the *after-rebuttal score prediction* task. Because most score updates after rebuttal do not exceed 1 point (see Fig. 3), we formulate this problem as a classification task. Specifically, given a before-rebuttal review, its corresponding author response and other peer reviews, we try to predict whether the reviewer will increase (INC), decrease (DEC) or keep (KEEP) her overall score after the rebuttal. We avoid predicting the final accept/reject decisions because they are not only based on the final scores (see Fig. 1, where a few low-score papers are accepted while some high-score papers are rejected), but also based on additional factors such as the balance of areas and diversity of papers, which are difficult to measure. The score updating of reviews, in contrast, only depends on the peer reviews and the authors responses.

We choose a classic feature-rich classification model for this task, for two reasons: a) model capacity is lower compared to, e.g., a deep neural network, which is beneficial in our small data scenario, and b) the results are easier to interpret.

---

[2] We only include n-grams that appear in at least 7 different author responses.

4.1 Features

Score features (`Score`). We use all peer review scores for a given submission to build an array of score-based features. These include review i's before-rebuttal `OVAL` (`self_score`), statistics of the other peer reviews' `OVAL` (denoted by `oth_X`, where `X` can be `max`/`min`/`mean`/`median`/`std`), statistics of all peer reviews' `OVAL` (`all_X`), and elementary arithmetic operations on the above features (e.g., `oth_mean-self` denotes the mean `OVAL` of the peer reviews minus review i's before-rebuttal `OVAL`). `CONF` are considered in a similar manner. We do not consider aspect scores such as `ORG` because they yielded no improvements in our preliminary experiments. The full list of features can be found in the supplementary material. We also include features based on the author response texts, as detailed below.

Length of response (`log_leng`). We have found that high-quality author responses are usually longer than the low-quality ones (see §3.3). We use the logarithm of the number of tokens in author responses as a feature.

Review-Response Similarity (`sim`). Lack of similarity between a review and its response may indicate that the response is "off-topic". To measure similarity, we have trained 300-dimensional skip-gram word embeddings on 5611 papers extracted from the cs.CL (computational and language) and cs.LG (learning) categories of ArXiv which were published between January 1, 2015 and December 31, 2017. We represent reviews and responses by averaging the embeddings of their words, and measure semantic similarity by cosine similarity.[3] We find it important to use word embeddings trained on *CL/LG* domain data: for example, nearest neighbors of "neural" in a model trained on Wikipedia are "axonal", "salience", while on Arxiv its nearest neighbors are "feedforward" and "deep". We find that iResps are more similar to their reviews than dResps and kResps: the average cosine similarity between the reviews and iResps, dResps and kResps are .38, .30 and .29, respectively.

Specificity (`spec`). In our human annotation experiments, unspecific responses were typically judged as weak because they did not address specific questions or weaknesses given by reviews. To measure the specificity of author responses, we use a feature-rich sentence-level specificity model by Li and Nenkova (2015) trained on multiple news corpora. The produced scores are in the $[0, 1]$ range, with higher values meaning higher specificity. iResps are slightly more specific than the other responses: the mean specificity scores for iResps, dResps and kResps are .29, .24 and .28, respectively. For each author response, we compute the `spec` scores for all their sentences and use statistics (`max`/`min`/`mean`/`median`/`std`) of the `spec` scores as features. The same strategy is used to build the politeness and convincingness features introduced below.

Politeness (`plt`). We employ the sentence-level politeness framework suggested by Danescu-Niculescu-Mizil et al. (2013) to quantify the politeness of the author responses. We have trained a simple bag-of-words based multi-layer perceptron (MLP) model using their Wikipedia and StackExchange data and applied it to the author responses, generating a politeness score in $[-1, 1]$ for each sentence in author responses, where higher scores mean higher politeness. While the mean politeness scores in iResps, dResps and kResps have no marked differences (all around 0.19), the score for the most polite sentence in iResps (.91) is higher than that of dResps (.68) and kResps (.90).

Convincingness (`cvc`). To approximate rebuttal convincingness we use the sentence-level convincingness model developed by Simpson and Gurevych (2018), trained on ∼1.2k argument pairs from web debate forums. We normalize all convincingness scores to $[0, 1]$, where larger scores mean higher convincingness. Mean convincingness scores for iResps, dResps and kResps are .60, .49 and .58, respectively.

Score validation. Since the `spec`, `plt` and `cvc` models are not trained on review-rebuttal data, we need to perform human evaluations to validate the produced scores. We rank the sentences in author responses in terms of their `spec`, `plt` and `cvc` scores and analyze the top and bottom 10 sentences in each ranking (see the supplementary material). We find that the scores successfully distinguish the most and least specific/polite/convincing sentences. To further validate the scores, for each type of score, we have randomly sampled 15 pairs of sentences from author responses and presented the pairs to 3 ex-

---

[3] We also used ROUGE (Lin, 2004) to measure the similarity but find the ROUGE scores to be highly correlated with the cosine similarities (Pearson correlation > 0.9), so we include only the cosine similarities in our models.

	`spec`	`plt`	`cvc`
Inter-User	.87	.87	.64
User-Score	.93	.87	.67

Table 7: Percentage of agreement for `spec`, `plt` and `cvc` scores. "User-Score" means the agreement between the aggregated (by majority voting) users' preferences and score-induced preferences.

perienced annotators, asking them to indicate the more specific/polite/convincing sentence in each pair. The agreement is presented in Table 7. The agreement between the users' aggregated preferences and score-induced preferences is quite high for all three types, confirming the validity of the scores. Note that the agreement for `cvc` is lower than the other two; the reason might be that it is difficult even for humans to judge convincingness of arguments, particularly when evaluated on the sentence level without surrounding context nor the corresponding review. The distribution of the `spec`, `plt` and `cvc` scores for iResps, dResps and kResps is in the supplementary material.

4.2 Results and Analyses

We perform experiments on a subset of the corpus which only includes the submissions that have author responses and three or more reviews opted in. We term this subset of the corpus *Submissions with Complete Reviews (Full)*. Training models on submissions with fewer reviews would bias certain features (e.g. `all_mean`) and thus bias the trained models. Also, we separate out the submissions from the Full set whose before-rebuttal average `OVAL` are between 3 and 4.5 (note that `OVAL` are in $[1, 6]$), so as to train and test a model specifically on borderline submissions for which score changes may be decisive for an accept or reject decision. We term this subset *Borderline Submissions (BRD)*. Full includes 791 submissions (80 INC, 60 DEC, 652 KEEP) and BRD includes 590 (69 INC, 48 DEC and 474 KEEP). All results and weights presented in this section are averaged over 5000 repeats of 10-fold cross validation; data entries are randomly shuffled for each repeat.

Feature Selection. We filter out features whose information gain is ranked in the bottom 50% of all features on the training set. For highly correlated features in the upper 50% (i.e. Pearson correlation ≥ 0.5), we filter out all but the one with the highest information gain. Remaining features are used to train a multinomial logistic regression model (i.e., MLP with no hidden layer and softmax activation function in the output layer). To balance the number of instances for the three classes, on the training set, in each fold of cross-validation we randomly down-sample cases with class KEEP to ensure that the number of KEEP is the same as the sum of INC and DEC. We also tried random forest, decision tree, support vector machines and Gaussian processes as classifiers, but their performances were similar or worse than that of logistic regression.

Feature Set	BRD	Full
`spec`	.324	.309
`plt`	.306	.310
`cvc`	.303	.304
`log_leng`	.340	.341
`sim`	.323	.302
`Score`	**.495**	**.526**
All but `Score`	.343	.336
All	**.522**	**.540**
Majority Baseline	.297	.301
Random Baseline	.258	.251

Table 8: Macro F-1 scores.

Results. Classification results are presented in Table 8. In addition, we compare to two baselines: the *majority baseline* always picks the majority decision (in our case, KEEP); the *random baseline* selects an action at random. Full results, including precision, recall and F1-scores for each label, can be found in the supplementary material.

We find that score-based features are most effective among all features. However, text-based features are also useful, supported by the observations that: **(i)** models using only text features all significantly (p-value < 0.01, double-tailed t-test) outperform the majority and random baseline; and **(ii)** using all features gives the best performance, significantly (p-value < 0.01) better than using any feature set alone.

Among the non-`Score` features, `log_leng` performs best. But we find it has high correlation with multiple `Score` features, and hence when all features are used, it is filtered out. The features `spec` and `sim` perform much better in BRD than in Full, which suggests that, for borderline papers, more weight is placed on whether the response explicitly addresses the points raised in reviews (similarity) and the specificity of the response.

Analysis. To interpret our results, we study the weights of the features in our logistic regression model shown in Tables 9 and 10. We observe the following trends:

- **"Peer pressure" is the most important factor of score change**: in both Full and BRD, features reflecting the gap between own and others' review scores (`oth_mean-self` and `self-oth_min`) have by far the largest weights compared to other feature groups. For example, in Full, the `Score` features have (absolute) weights of 0.4 or higher for the class INC, while all other features are substantially below 0.2. The weights make intuitive sense: e.g., when the mean of the other reviewers' scores is above a reviewer's initial score, she has a strong tendency to increase her own score and not to decrease her own score. Similarly, when a review contains a very convincing sentence, this substantially decreases the probability of a score decrease.

- **To improve the score for a borderline paper, a more convincing, specific and explicit response may be helpful**: in Full, no weight of a text-based feature is above 0.2 for INC; however, in BRD, the weights for `cvc_min`, `spec_median` and `sim` are all above 0.2. This asymmetry of the text-based features across Full and BRD also suggests that reviewers do appear to pay more attention to the author responses in situations where they may matter (e.g., make the difference between accept or reject decisions).

- **An impolite author response may harm the final score**: in both Full and BRD, the weight of `plt_max` is negative for DEC. In addition, in Full a more polite response helps increase the final score (positive weight for INC, close to 0 weight for KEEP). In BRD, in contrast, a more polite response may not increase the score but only keep it unchanged (positive weight for KEEP, close to 0 weight for INC). If we take BRD papers as those for which the author responses really matter, this means that politeness has an asymmetrical effect: it may push a paper below the acceptance threshold, but not above it. Indeed, `plt_max` is the second best text-feature for predicting decrease for BRD papers.

Feature	INC	DEC	KEEP
`oth_mean-self`	1.044	-1.265	.221
`self-oth_min`	-.378	.188	.190
`cvc_max`	.078	-.271	.193
`spec_median`	.159	-.224	-.065
`plt_max`	.170	-.174	.004
`sim`	.019	.099	-.119
`spec_max`	.022	.029	-.051

Table 9: Feature weights in multinomial logistic regression trained on Full.

Feature	INC	DEC	KEEP
`oth_mean-self`	.855	-1.026	.171
`self-oth_min`	-.372	.191	.181
`cvc_min`	.224	-.258	-.034
`spec_median`	.293	-.122	-.171
`sim`	.214	-.161	-.053
`cvc_max`	.117	-.085	-.033
`plt_max`	.016	-.192	.176

Table 10: Feature weights in multinomial logistic regression trained on BRD.

5 Discussion

The opinion update process we have described in §4.2 is closely related to the work on *opinion dynamics* (DeGroot, 1974; Acemoglu and Ozdaglar, 2011), which studies how human subjects change their opinions as a reaction to those of peers.

The "peer pressure" effect (opinions being updated to mean opinions) is widely observed in opinion formation of human subjects in controlled experiments. Lorenz et al. (2011) find that in simple estimation tasks ("What's the population density of Switzerland?"), human subjects tend to lean towards a consensus once they are exposed to the opinions of others. Similarly, Moussaid et al. (2013) find two dominant effects for simple factual questions: human subjects tend towards the mean opinion and towards the opinions of highly confident individuals. Our experiments also show that the mean opinion plays a very prominent role in peer reviews, but they show no evidence supporting the confidence effect: features based on the confidence scores do not play a significant role in deciding the final scores (see §4.2). We believe this is due to two main differences between peer reviewing and the controlled experiments in the above works: (i) there does not exist a ground-truth score for a submission, while such true answers about factual questions do exist in the controlled experiments; and (ii) participants of the controlled experiments lose money if they give incorrect answers, but a reviewer loses nothing when she does not adjust to a (self-assessed) expert.

Three types of *biases* have been studied in explanatory models of opinion dynamics in recent years. The first is opposition between members of different groups (e.g., due to *group-identity*) leading to distancing from certain subjects' opinions (Altafini, 2013; Eger, 2016). The second is *homophily*: individuals ignore opinions too different from their own (Deffuant et al., 2000; Hegselmann and Krause, 2002). The third is *conformity* (Buechel et al., 2015), i.e., the desire to conform to a group norm/opinion. Conformity bias can be strong and persist even in the presence of overwhelming evidence that a group opinion is wrong (Asch, 1951). Our observation that reviewers tend to converge to the mean of all reviews (§4.2) suggests that conformity bias also plays a prominent role in peer reviewing. We found no evidence (on an aggregate level) for the other two biases.

To summarize, conformity bias is the main bias we identified in the peer reviewing process. However, conformity bias has a negative effect on crowd-wisdom in estimation tasks (Lorenz et al., 2011), which strengthens confidence of human subjects in the correctness of their converged answer, while the actual correctness of their consensus is often even worse than the mean of multiple independent answers. A simple method to reduce conformity bias is to blind reviewers from each other, only allowing reviewers to update their reviews based on the author responses; the area chair (who can see all reviews for a paper) is then responsible for considering all (possibly conflicting) reviews and making the accept/reject recommendation. We believe that peer reviewing is to a large degree an opinion dynamics process, a neglected insight hitherto, and that lessons from this field should therefore be beneficial for peer reviewing for NLP conferences and beyond.

Finally, concerning the helpfulness of individual review based feature groups, we believe it reflects a weakness of the current rebuttal stage that politeness does matter, because this is *merely* a social aspect unrelated to the quality of the assessed papers. However, we also showed that filling up author responses with "thank you"s is unlikely to increase a reviewer's score for a borderline paper—so at least, authors do not seem to be able to sneak their papers in via social effects.

6 Conclusion

We presented a review corpus consisting of over 4k reviews and 1.2k author responses from ACL-2018. To the best of our knowledge, it is the first corpus that includes both before- and after-rebuttal reviews for both accepted and rejected papers in a major NLP conference. We qualitatively and quantitatively analyzed the corpus, including a manual classification of paper weaknesses outlined by reviewers and a quality rating study of the corresponding author responses.

In addition, we proposed a classification model to predict whether a reviewer will increase/decrease/keep her overall score after rebuttal. By analyzing the feature weights in our model, we quantitatively measured the importance of different decision variables for score updates. We found that the gap between a reviewer's initial score and her peers' scores is the main explanatory variable. Rebuttal-related factors like convincingness, specificity and politeness of responses are considerably less important but still have a statistically significant effect, especially for borderline papers.[4] Our findings shed light on the predominant role of the *conformity bias* in peer reviewing (see §5), and we discuss alternative peer review models addressing this bias. We hope our analyses will help the community better understand the strengths and weaknesses of the current peer review workflow, spurring further discussions.

Finally, provided that the rebuttal phase remains a key feature in many peer reviewed conferences, we think that our novel after-rebuttal score change prediction task can be practically beneficial for authors to restructure their author responses and thereby make them more effective.

Acknowledgements

The authors thank the anonymous reviewers and Dan Jurafsky for constructive comments and helpful remarks. This work has been supported by the ArguAna Project GU 798/20-1 (DFG), the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1 and grant GU 798/17-1), and the DFG-funded research training group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES, GRK 1994/1). Ilia Kuznetsov has been supported by the FAZIT Foundation.

---

[4] We believe that they might become more important when further argumentation/text-based features are integrated.

References

Daron Acemoglu and Asuman Ozdaglar. 2011. Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1(1):3–49.

Claudio Altafini. 2013. Consensus problems on networks with antagonistic interactions. *IEEE Trans. Automat. Contr.*, 58(4):935–946.

James Andreoni and Ragan Petrie. 2004. Public goods experiments without confidentiality: a glimpse into fund-raising. *Journal of public Economics*, 88(7-8):1605–1623.

Solomon E. Asch. 1951. Effects of group pressure upon the modification and distortion of judgments. *Groups, Leadership, and Men*, pages 177–190.

Aliaksandr Birukou, Joseph Wakeling, Claudio Bartolini, Fabio Casati, Maurizio Marchese, Katsiaryna Mirylenka, Nardine Osman, Azzurra Ragone, Carles Sierra, and Aalam Wassef. 2011. Alternatives to peer review: Novel approaches for research evaluation. *Frontiers in Computational Neuroscience*, 5:56.

Berno Buechel, Tim Hellmann, and Stefan Klößner. 2015. Opinion dynamics and wisdom under conformity. *Journal of Economic Dynamics and Control*, 52(C):240–257.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 250–259.

Guillaume Deffuant, David Neau, Frederic Amblard, and Gérard Weisbuch. 2000. Mixing beliefs among interacting agents. *Advances in Complex Systems (ACS)*, 03(01n04):87–98.

Morris H DeGroot. 1974. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74.

Steffen Eger. 2016. Opinion dynamics and wisdom under out-group discrimination. *Mathematical Social Sciences*, 80(C):97–107.

Laura J. Falkenberg and Patricia A. Soranno. 2018. Reviewing reviews: An evaluation of peer reviews of journal article submissions. *Limnology and Oceanography Bulletin*, 27(1):1–5.

Santo Fortunato, Carl T. Bergstrom, Katy Börner, James A. Evans, Dirk Helbing, Staša Milojević, Alexander M. Petersen, Filippo Radicchi, Roberta Sinatra, Brian Uzzi, Alessandro Vespignani, Ludo Waltman, Dashun Wang, and Albert-László Barabási. 2018. Science of science. *Science*, 359(6379).

Fiona Godlee, Catharina R. Gale, and Christopher N. Martyn. 1998. Effect on the quality of peer review of blinding reviewers and asking them to sign their reports: A randomized controlled trial. *JAMA*, 280(3):237–240.

Rainer Hegselmann and Ulrich Krause. 2002. Opinion dynamics and bounded confidence: Models, analysis and simulation. *Journal of Artificial Societies and Social Simulation*, 5:1–24.

Xinyu Hua, Mitko Nikolov, Nikhil Badugu, and Lu Wang. 2019. Argument mining for understanding peer reviews. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, USA, June 2 - 7, 2019, Volume 2 (Short Papers)*.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard H. Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (PeerRead): Collection, insights and NLP applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1647–1661.

Michail Kovanis, Ludovic Trinquart, Philippe Ravaud, and Raphaël Porcher. 2017. Evaluating alternative systems of peer review: A large-scale agent-based modelling approach to scientific publication. *Scientometrics*, 113(1):651–671.

John Langford and Mark Guzdial. 2015. The arbitrariness of reviews, and advice for school administrators. *Commun. ACM*, 58(4):12–13.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2281–2287.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Jan Lorenz, Heiko Rauhut, Frank Schweitzer, and Dirk Helbing. 2011. How social influence can undermine the wisdom of crowd effect. *Proceedings of the National Academy of Sciences*, 108(22):9020–9025.

Mehdi Moussaid, Juliane E. Kaemmer, Pantelis P. Analytis, and Hansjoerg Neth. 2013. Social Influence and the Collective Dynamics of Opinion Formation. *PLoS ONE*, 8(11):e78433+.

Azzurra Ragone, Katsiaryna Mirylenka, Fabio Casati, and Maurizio Marchese. 2013. On peer review in computer science: Analysis of its effectiveness and suggestions for improvement. *Scientometrics*, 97(2):317–356.

Edwin Simpson and Iryna Gurevych. 2018. Finding convincing arguments using scalable bayesian preference learning. *Transactions of the Association for Computational Linguistics*, 6:357–371.

Appendicies

A Consent Message

Before a reviewer or an author enters her reviews or author responses, the following message appears to ask for her consent for data sharing:

> ATTENTION: this time, we plan to do some analytics on anonymized reviews and rebuttal statements, upon the agreement of the reviewers and authors, with the purpose of improving the quality of reviews. The data will be compiled into a unique corpus, which we potentially envisage as a great resource for NLP, e.g. for sentiment analysis and argumentation mining, and made available to the community properly anonymized at earliest in 2 years. We hope to provide data on ”how to review” to younger researchers, and improve transparency of the reviewing process in ACL in general.
>
> By default, you agree that your anonymised rebuttal statement can be freely used for research purposes and published under an appropriate open-source license within at earliest 2 years from the acceptance deadline.
>
> Place an ’x’ mark in the NO box if you would like to opt out of the data collection.
>
> `[x]`: YES
> `[ ]`: NO

Analyses on Submissions

We rank n-grams in both accepted and rejected papers according to the *log-likelihood ratio* (LLR) statistic, taking both accepted and rejected papers as one big corpus, respectively. The goal is to find n-grams that occur unusually frequently in one of the two groups, relative to the respective other.

Table 11 shows a few hand-selected n-grams with highest LLR for *accepted papers*; high-LLR n-grams for rejected papers are not presented due to licensing. To filter out noise, we only include n-grams that occur in at least 7 different papers. We can observe some interesting patterns: accepted papers appear to cite recent work, which reflects potential novelty and appropriate comparison to state of the art; tend to use more mathematics (of a particular kind); have an appendix; do significance testing; release code upon publication; and have multiple figures including subfigures.

Hot n-grams	Possible Interpretation
(2017)	Cite recent work
($z\|x$)	Math
artetxe et al	Authors working on a hot topic
dozat and manning	Authors of an influential method
in the supplementary	Paper has appendix
contextualized word representations	Trendy method
upon publication .	Code/data will be released
statistical significance of	Mathematically rigorous
figure 3 (	Multiple figures with subfigures

Table 11: Selected 3-grams that distinguish accepted from rejected papers based on the LLR statistics.

B Statistics on Reviewer Information

In this section, we present some statistics of all 1440 reviewers of ACL-18.

Country. The reviewers work in 53 different countries. The top 10 countries where the reviewers work are presented in Fig. 5. The distribution of the reviewer working places is heavily long-tailed: the United States alone contributes 36.9% of all reviewers, followed by China (8.7%), the United Kingdom (7.8%) and Germany (7.6%). Seven countries have more than 50 reviewers, and 19 countries have more than 10 reviewers.

Affiliation. The reviewers are from around 700 organisations. But as reviewers use different names to refer to the same organisation (e.g., both MIT and Massachusetts Institute of Technology are used), the real number of organisations can be much lower. The top 10 organisations and their reviewers numbers are presented in Fig. 6. Nine organisations contribute more than 20 reviewers, and 19 organisations contribute more than 10 reviewers.

Seniority. Most reviewers (69.9%) do not report their seniority levels. Among those that

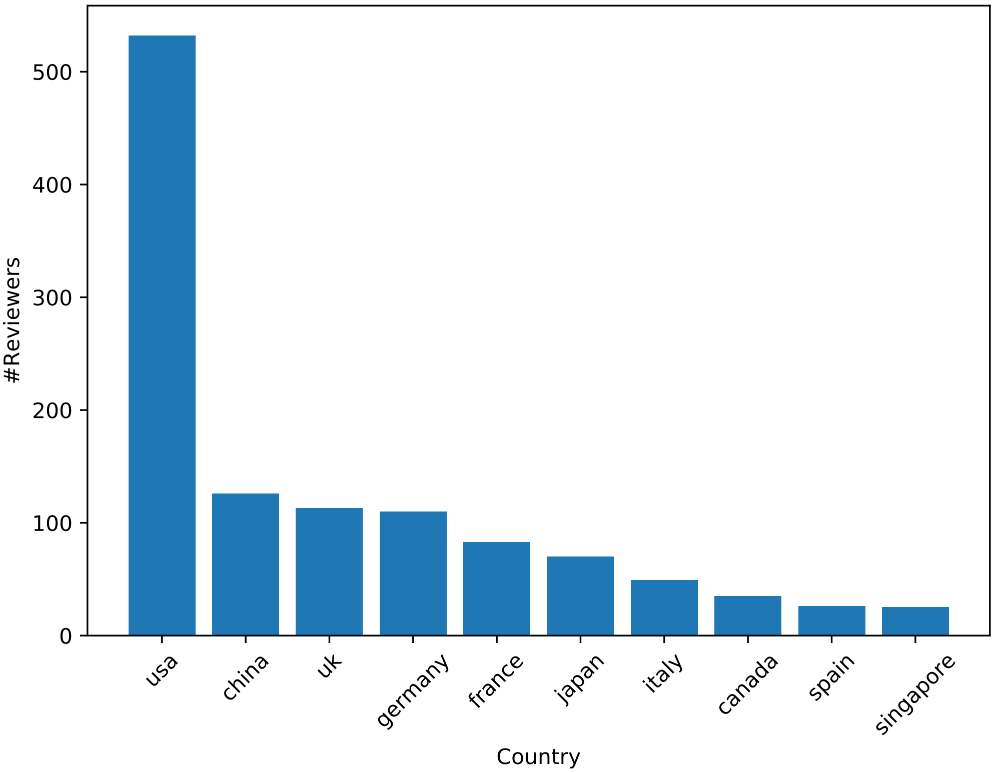


Figure 5: Distribution of countries where reviewers work.

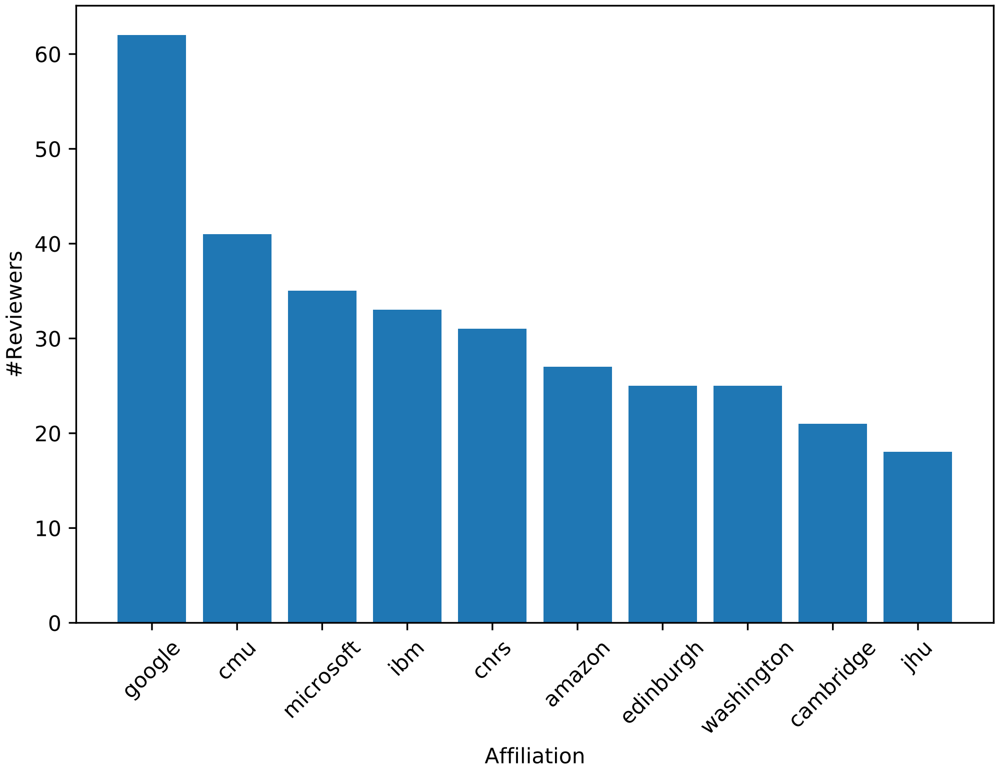


Figure 6: Distribution of organisations where reviewers work.

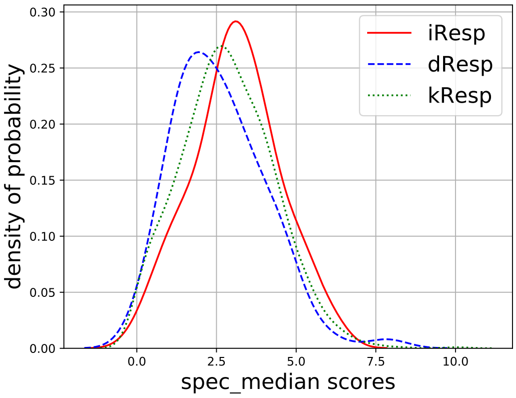


Figure 7: Smoothed distribution of specificity scores.

have reported their seniority, 50.2% are Professors, 27.6% are PhD students, and 22.2% are Post-Doc/Assistant-Professor.

Gender. We estimate the gender of the reviewers from their first names, using the tool available at `https://github.com/kensk8er/chicksexer`. 73.4% reviewers are estimated to be male and the rest 26.6% are estimated to be female.

C Full Results

The precision, recall and F1-scores for each label in both Full and BRD are presented in Table 12 and 13, respectively.

D Features

The full list of our hand-crafted features is presented in Table 14.

E Specificity Scores

We tokenize author responses with `nltk`, remove sentences with fewer than 10 tokens and rank the remaining sentences by their specificity scores. All scores are normalized to $[0, 10]$, with higher scores meaning higher specificity. The distribution of the specificity scores for author responses leading to increased, decreased and unchanged scores is illustrated in Fig. 7.

Top 10 sentences are presented below[5], and they all receive a specificity score 10.

- *We have already checked it. We can change the sentence in the last paragraph of Section ### to ''Since the proposed method only substituted ### based on ###, then the naturalness of ### using the proposed method is better than ###. This method was used because we have to maintain the context; The result can be more than 100% because we assume that the ### of original was 100% while based on human judgement, there are possibility that the ### of resulting sentences using the proposed method is better than the original one.*
- *###| ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# |*
- *###| ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# |*
- *There are two reasons why we mention that: (i) many papers exist, however, many previous papers made the same (or similar) conclusions, so some are picked up as representatives and (ii) because ### is a high-level conference, it's thought that there was no need to explain too much, and also because there are limited pages, space was wanted to be left to explain the analysis as detailed as possible and put focus on the analysis.*
- *Other external knowledge sources apart from ### do not add much: In principle, all resources we used originate in ###, the difference is the degree of knowledge we use. The novelty in this work does not lie in the use of ### as a knowledge resource but more generally in the principled ### of the classes.*
- *We will include this discussion in the paper. Other ### models (e.g., ###; ###) can in theory predict ###, however, they are not directly applicable to ### since they cannot handle ### representations, i.e., variables can refer to a ### representation (e.g., variable ### refers to an entire proposition and variable ### refers to a segment of meaning).*
- *As noted in our response to reviewer 3 - our results on the ### dataset of ### are on par with the ### model stated in the ### paper provided by reviewer 3 (which is a SOTA non-neural ### model) - although we used a very basic set of features and apply very limited task-specific tuning to our models.*

---

[5] The examples are anonymized by replacing citations, venues, method names, exact scores, etc. with placeholders; we also include cases where our system has erroneously rated non-text data (i.e. tables).

Feature Set	INC-p	INC-r	INC-f1	DEC-p	DEC-r	DEC-f1	KEEP-p	KEEP-r	KEEP-f1
`spec`	.110	.023	.035	0	0	0	.820	.976	.892
`plt`	.043	.063	.047	.029	.029	.029	.824	.912	.864
`cvc`	.020	.014	.107	0	0	0	.824	.977	.893
`log_leng`	.187	.167	.154	0	0	0	.827	.930	.874
`sim`	.013	.011	.013	0	0	0	.810	.990	.897
`Score`	**.331**	.485	**.386**	**.380**	.527	.409	.878	.790	.829
All but `Score`	.142	.202	.162	.025	.033	.029	.820	.820	.818
All	.299	**.555**	.374	.364	**.569**	**.438**	**.889**	.757	.817
Majority Baseline	0	0	0	0	0	0	.823	**1**	**.903**
Random Baseline	.100	.332	.154	.076	.334	.123	.825	.332	.474

Table 12: Macro F-1 scores on Full. All results are averaged over 5000 repeats of 10-fold cross validation.

Feature Set	INC-p	INC-r	INC-f1	DEC-p	DEC-r	DEC-f1	KEEP-p	KEEP-r	KEEP-f1
`spec`	.119	.101	.102	0	0	0	.804	.956	.872
`plt`	.100	.012	.022	.020	.014	.017	.804	.982	.883
`cvc`	.033	.020	.025	0	0	0	.803	.988	.885
`log_leng`	.180	.229	.184	0	0	0	.811	.879	.840
`sim`	.096	.133	.110	0	0	0	.805	.927	.861
`Score`	**.313**	.556	**.394**	.377	.356	.302	.851	.743	.792
All but `Score`	.205	.331	.231	.050	.011	.018	.801	.768	.780
All	.295	**.570**	.376	**.387**	**.548**	**.418**	**.875**	.710	.782
Majority Baseline	0	0	0	0	0	0	.802	**1**	**.890**
Random Baseline	.117	.333	.173	.082	.335	.131	.802	.333	.470

Table 13: Macro F-1 scores on BRD. All results are averaged over 5000 repeats of 10-fold cross validation.

Feature set	Features
`Score`	`self_before`, `self_conf`, `oth_max`, `oth_min`, `oth_mean`, `oth_median`, `oth_std`, `oth_conf_max`, `oth_conf_min`, `oth_conf_mean`, `oth_conf_median`, `oth_conf_std`, `oth_mean-self`, `oth_median-self`, `oth_max-self`, `self-oth_min`, `oth_conf_std`, `all_max`, `all_min`, `all_mean`, `all_median`, `all_std`, `self_before**2`, `all_mean-self`, `all_max-self`, `all_median-self`, `self-all_min`
`spec`	`spec_max`, `spec_min`, `spec_mean`, `spec_median`, `spec_std`
`cvc`	`cvc_max`, `cvc_min`, `cvc_mean`, `cvc_median`, `cvc_std`
`plt`	`plt_max`, `plt_min`, `plt_mean`, `plt_median`, `plt_std`
`log_leng`	Logarithm of the token number of the author response
`sim`	Cosine similarity of the embeddings of a review and its corresponding author response

Table 14: The full list of hand-crafted features.

- *Although the models used are general to all seq2seq generation problems, the heuristics we used to select ### are specific to generating the ### (take for example, the heuristic based on ### - it was motivated by the fact that ### have a higher readability, hence the network has to focus towards better readable information in the ### in order to generate ###).*

- *Because the size of the training data for ### task is very small, ### instances for ### task and ### instances for ### task, whereas the number of the parameters of the whole network is very big, we pre-training the ### network based on ###, released for ### task, and pre-training the ### network based on the training data for ### task.*

- *Related workshop and share tasks, including ### (collocated with ###), ### (collocated with ###), ### (collocated with ###), and ### (collocated with ###), show a great potential on applying NLP technologies to the ### domain.*

Bottom 10 sentences are presented below. Their specificity scores are all smaller than 0.001.

- *It would be a little difficult to build this connection.*

- *It is not accurate and we will use 'obvious' instead.*

- *We are not quite sure which part is not identical.*

- *I will check that again and will write it as you said*

- *Therefore we can see that they have no relation with each other.*

- *We will try to do this in our future work.*

- *But we do not see this as a weakness of our approach.*

- *That is why we do not do that in the first submission.*

- *So this is really true for all the "models".*

- *Thank you very much for the reviews and for the very useful*

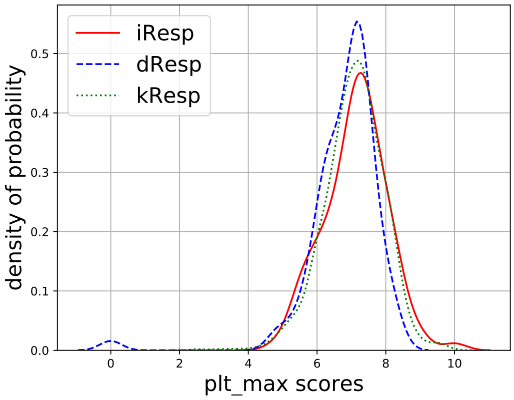


Figure 8: Smoothed distribution of politeness scores.

F Politeness Scores

We use the politeness scorer to rate the same set of sentences as in the specificity evaluation. We normalize all politeness scores to [0,10], with higher values meaning higher politeness. The distribution of the politeness scores is illustrated in Fig. 8. **Top 10 sentences** and their politeness scores are presented below.

- *(9.6) We thank this reviewer for his helpful comments that help improving the paper.*

- *(9.5) Thanks for the suggestion, we found that in many cases the two sentences that are separated by ### also have similar patterns to ###, and the size of the dataset would be too small to train a representative ### model if we only picked out the separate sentences examples.*

- *(9.5) Thank you for the helpful suggestion of including more qualitative results to more thoroughly understand the proposed approach.*

- *(9.4) We again thank the reviewer for the detailed and carefully constructed review and assure that the main concerns raised by the reviewer are fixable and we will fix them in the final version of the paper.*

- *(9.4) Meanwhile, thanks for your suggestion for more in-depth discussion on ###*

- *(9.3) We apologize for this error, and will correct this in the final version of the paper upon acceptance.*

- *(9.3) An interesting alternative approach would be the one proposed by the reviewer, but we chose this model because we wanted to encourage the model to aggregate information from a variety of positions, and in our experience ### has trouble learning to ### in this way because by design ### focuses on one position only.*
- *###| ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# |*
- *###| ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# | ##.# |*
- *(9.2) Depends on the task and the characteristic of two datasets, each proposed method shows its effectiveness, e.g., the ### using the ### between two entities is appropriate for the ### task since ### is systematically organized.*

Bottom 10 sentences and their politeness scores are presented below.

- *(1.7) By comparing ### with ###-, we know whether employing a ### helps; By comparing ### with ###, we know whether employ a ### helps; By comparing ### with ###, we know whether the ### helps.*
- *(2.2) ### = ### ∗ ###, where ### is a matrix of n samples with ### features followed by ### features, hence the size of ### is ###.*
- *(2.3) In other words, our coverage is ### times larger than theirs, so our proposed system can deal much better with the noise when learning ###.*
- *(2.4) And another difference lies in the ### layer, which contains ###, so when we process ### in ### independently which encourages our model to learn diverse features.*
- *(2.4) We will implement their method on our corpora and make some comparison with our method in the next version of our manuscript.*
- *(2.4) We are not giving up ### nor are we claiming that ### is more powerful.*
- *(2.5) If our paper is accepted we will make sure additional relevant technical details are added.*

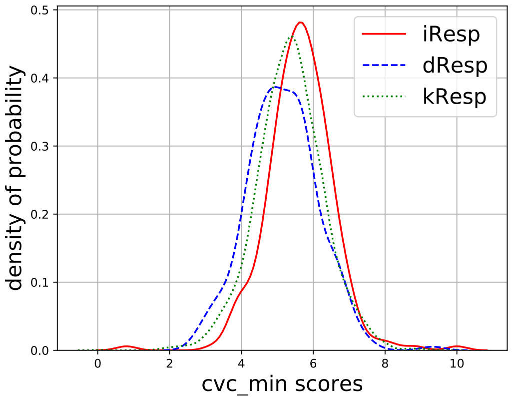


Figure 9: Smoothed distribution of the convincingess scores.

- *(2.6) In response to your general remark: we can see how our discussion and conclusions would lead a reader to conclude that; rather, this paper is an exploration in an area that is, as you say, worth exploring.*
- *(2.7) Our main contribution is introduction of ### without requiring neither supervision nor feature engineering.*
- *(2.7) The most salient problem encountered in our system is that a user might change ###, also brought up by R3 (Please refer to our response to weakness4 of R3).*

G Convincingness Scores

We use the convincingness scorer to rate the same set of sentences as in the previous two studies. The convincingness scores are normalized to $[0, 10]$, with higher values meaning higher convincingness. The distribution of the convincingness scores is illustrated in Fig. 9. **Top 10 sentences** in terms of convincingness are presented below. All top 10 sentences' convincingness scores are above 9.8.

- *In the revision, we perform the evaluation of the model with ### and ###, respectively.*
- *Deepening the ### system would inevitably increase model parameters, and slow the training and decoding, which is not what we expect.*
- *A technical document is defined as the document that assumes sufficient background*

knowledge and familiarity with the key technical or central/important terms in the document.

- *As reported in our paper, the success rate of our optimization algorithm is ### while, on average, only ###% of words are altered.*
- *The focus of this work is not a comparison of ### methods with ### methods, but how to mitigate the lack of labeled data problem in learning of a ### model.*
- *Our model works well on datasets that are deemed small for deep architectures to work and belong to special domains for which ### is not possible.*
- *We conduct t-test and get the p value as ###, which shows good agreement.*
- *Particularly, we will strive to improve the presentation quality and to make the draft more readable and better organized for more potential readers.*
- *Furthermore, ### can help ### to alleviate the performance degradation by ###.*
- *The ### experiments in Section ### show that our ### framework can achieve higher accuracy than the methods that rely on the same set of resources, while the state-of-the-art ### methods also require some other resources.*

Bottom 10 sentences in terms of convincingness scores are presented below[6]. Their convincingness scores are all below 0.01.

- *"Weakness 3:""why ... report on ... the '###' if you then dismiss it"""*
- *It is **not** used in the **testing** (###).*
- *Annotator 1: "Are you a citizen?" No =¿ Answer: No*
- *"Rev: ""It seems that ...""""*
- *"Weakness 3:""how did you learn the embeddings? ... ### model? How"*
- *Please refer to the reply regarding Weakness argument 1 in Review 1.*
- *"Are you over 21?" Yes =¿ Answer: Yes*
- *Please see our reply to Review 1's weakness argument 3.*
- [*Please see our response to R2's argument 3*]
- *We are sorry we didn't explain the notation.*

---

[6] Note the large number of references to other responses and to the original reviews

Casting Light on Invisible Cities:
Computationally Engaging with Literary Criticism

Shufan Wang
University of Massachusetts, Amherst
shufanwang@cs.umass.edu

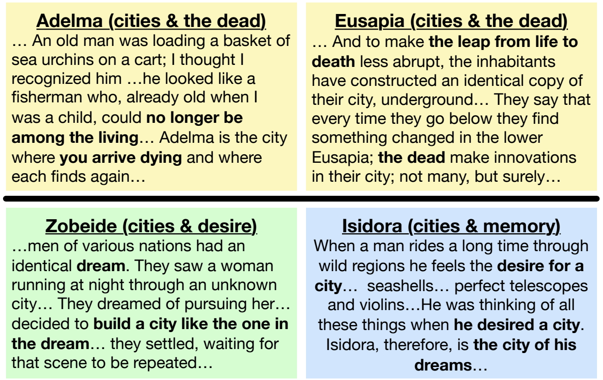
Mohit Iyyer
University of Massachusetts, Amherst
miyyer@cs.umass.edu

Abstract

Literary critics often attempt to uncover meaning in a single work of literature through careful reading and analysis. Applying natural language processing methods to aid in such literary analyses remains a challenge in digital humanities. While most previous work focuses on "distant reading" by algorithmically discovering high-level patterns from large collections of literary works, here we sharpen the focus of our methods to a single literary theory about Italo Calvino's postmodern novel *Invisible Cities*, which consists of 55 short descriptions of imaginary cities. Calvino has provided a classification of these cities into eleven thematic groups, but literary scholars disagree as to how trustworthy his categorization is. Due to the unique structure of this novel, we can computationally weigh in on this debate: we leverage pretrained contextualized representations to embed each city's description and use unsupervised methods to cluster these embeddings. Additionally, we compare results of our computational approach to similarity judgments generated by human readers. Our work is a first step towards incorporating natural language processing into literary criticism.

1 Introduction

Literary critics form interpretations of meaning in works of literature. Building computational models that can help form and test these interpretations is a fundamental goal of digital humanities research (Benzon and Hays, 1976). Within natural language processing, most previous work that engages with literature relies on "distant reading" (Jockers, 2013), which involves discovering high-level patterns from large collections of stories (Bamman et al., 2014; Chaturvedi et al., 2018). We depart from this trend by showing that computational techniques can also engage with literary criticism at a closer distance: concretely, we use recent advances in text representation learning to test a single literary theory about the novel *Invisible Cities* by Italo Calvino.



Adelma (cities & the dead)
… An old man was loading a basket of sea urchins on a cart; I thought I recognized him …he looked like a fisherman who, already old when I was a child, could **no longer be among the living**… Adelma is the city where **you arrive dying** and where each finds again…

Eusapia (cities & the dead)
… And to make **the leap from life to death** less abrupt, the inhabitants have constructed an identical copy of their city, underground… They say that every time they go below they find something changed in the lower Eusapia; **the dead** make innovations in their city; not many, but surely…

Zobeide (cities & desire)
…men of various nations had an identical **dream**. They saw a woman running at night through an unknown city… They dreamed of pursuing her… decided to **build a city like the one in the dream**… they settled, waiting for that scene to be repeated…

Isidora (cities & memory)
When a man rides a long time through wild regions he feels the **desire for a city**… seashells… perfect telescopes and violins…He was thinking of all these things when **he desired a city**. Isidora, therefore, is **the city of his dreams**…

Figure 1: Calvino labels the thematically-similar cities in the top row as *cities & the dead*. However, although the bottom two cities share a theme of desire, he assigns them to different groups.

Framed as a dialogue between the traveler Marco Polo and the emperor Kublai Khan, *Invisible Cities* consists of 55 prose poems, each of which describes an imaginary city. Calvino categorizes these cities into eleven thematic groups that deal with human emotions (e.g., desires, memories), general objects (eyes, sky, signs), and unusual properties (continuous, hidden, thin). Many critics argue that Calvino's labels are not meaningful, while others believe that there is a distinct thematic separation between the groups, including the author himself (Calvino, 2004). The unique structure of this novel — each city's description is short and self-contained (Figure 1) — allows us to computationally examine this debate.

As the book is too small to train any models, we leverage recent advances in large-scale language model-based representations (Peters et al., 2018a; Devlin et al., 2018) to compute a representation of each city. We feed these representa-

Proceedings of NAACL-HLT 2019, pages 1291–1297
Minneapolis, Minnesota, June 2 - June 7, 2019.

tions into a clustering algorithm that produces exactly eleven clusters of five cities each and evaluate them against both Calvino's original labels and crowdsourced human judgments. While the overall correlation with Calvino's labels is low, both computers and humans can reliably identify some thematic groups associated with concrete objects.

While prior work has computationally analyzed a single book (Eve, 2019), our work goes beyond simple word frequency or n-gram counts by leveraging the power of pretrained language models to engage with literary criticism. Admittedly, our approach and evaluations are specific to *Invisible Cities*, but we believe that similar analyses of more conventionally-structured novels could become possible as text representation methods improve. We also highlight two challenges of applying computational methods to literary criticisms: (1) text representation methods are imperfect, especially when given writing as complex as Calvino's; and (2) evaluation is difficult because there is no consensus among literary critics on a single "correct" interpretation.

2 Literary analyses of *Invisible Cities*

Before describing our method and results, we first review critical opinions on both sides of whether Calvino's thematic groups meaningfully characterize his city descriptions.

The groups are meaningful: Some scholars believe that the thematic grouping imposed by Calvino reflects properties of the cities he describes; Vrbani (2012), for example, argues that Calvino's structure are "ontologically grounded in different ways". Buitendijk (2018) further provides examples of cities with the same label that are clearly thematically similar, pointing at the "cities of desire" as "informed by 20th century theories of desires associated with Sigmund Freud". Calvino (2004) himself claims that he creates most categorizations of cities with clear labels in mind, especially the cities of memory and desire, which he deemed as "fundamental cornerstones" of the novel. However, many critics argue that authorial intent is irrelevant when analyzing literature (Wimsatt and Beardsley, 1946; Barthes, 1994).

The groups are arbitrary: On the other hand, a large body of criticism focuses on the apparent mismatch between a city's assigned thematic group and the content of its descriptions. Bloom (2002) claims that the "cities are totally interchangeable"; Springer (1985) agrees, stating that "even the categories themselves seem both chosen and assigned arbitrarily". Teichert (1985) contends that "the catalogue is superimposed on, but does not cover, the elusive, fluid mass of an unwritten world".

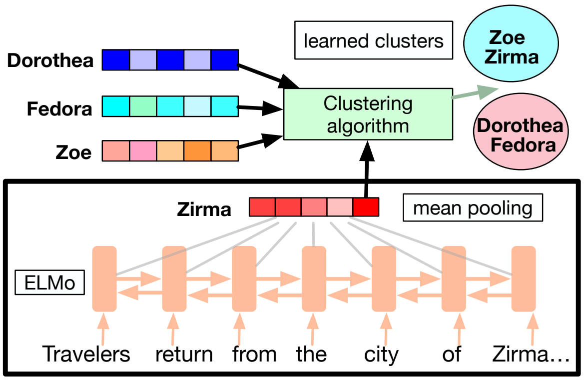


Figure 2: We first embed each city by averaging token representations derived from a pretrained model such as ELMo. Then, we feed the city embeddings to a clustering algorithm and analyze the learned clusters.

While out of scope for our computational analysis, many possible theories exist regarding *why* the groupings appear largely incoherent. For instance, Boeck (2004) posits that the structural incoherence exists because all of the cities actually describe different facets of Marco Polo's hometown of Venice. Breiner (1988) argues instead that Calvino's labels "may refer only to a projection of the Khan's occupational thirst for order, unrelated to the structure of the text", while Knowles (2015) hypothesizes that the mismatch is one of many obstacles that readers need to "untangle" to understand the central substance of the novel.

3 A Computational Analysis

We focus on measuring to what extent computers can recover Calvino's thematic groupings when given just raw text of the city descriptions. At a high level, our approach (Figure 2) involves (1) computing a vector representation for every city and (2) performing unsupervised clustering of these representations. The rest of this section describes both of these steps in more detail.

3.1 Embedding city descriptions

While each of the city descriptions is relatively short, Calvino's writing is filled with rare words, complex syntactic structures, and figurative language.[1] Capturing the essential components of each city in a single vector is thus not as simple as it is with more standard forms of text. Nevertheless, we hope that representations from language models trained over billions of words of text can extract some meaningful semantics from these descriptions. We experiment with three different pretrained representations: ELMo (Peters et al., 2018a), BERT (Devlin et al., 2018), and GloVe (Pennington et al., 2014). To produce a single city embedding, we compute the TF-IDF weighted element-wise mean of the token-level representations.[2] For all pretrained methods, we additionally reduce the dimensionality of the city embeddings to 40 using PCA for increased compatibility with our clustering algorithm.

3.2 Clustering city representations

Given 55 city representations, how do we group them into eleven clusters of five cities each? Initially, we experimented with a graph-based community detection algorithm that maximizes cluster modularity (Newman, 2006), but we found no simple way to constrain this method to produce a specific number of equally-sized clusters. The brute force approach of enumerating all possible cluster assignments is intractable given the large search space ($\frac{55!}{(5!)^{11}}$ possible assignments). We devise a simple clustering algorithm to approximate this process. First, we initialize with random cluster assignments and define "cluster strength" to be the relative difference between "intra-group" Euclidean distance and "inter-group" Euclidean distance.[3] Then, we iteratively propose random exchanges of memberships, only accepting these proposals when the cluster strength increases, until convergence. To evaluate the quality of the computationally-derived clusters against those of Calvino, we measure *cluster purity* (Manning et al., 2008):[4] given a set of predicted clusters M and ground-truth clusters D that both partition a set of N data points,

$$\text{purity} = \frac{1}{N}\sum_{m\in M} max_{d\in D}|m \cap d|.$$

Method	Purity	Accuracy
Random	0.32	33.3
GloVe	0.35	35.9
BERT	0.40	39.3
ELMo	0.42	44.6
Human	-	48.8

Table 1: Results from cluster purity and accuracy on the "odd-one-out" task suggests that Calvino's thematic groups are not completely arbitrary.

4 Evaluating clustering assignments

While the results from the above section allow us to compare our three computational methods against each other, we additionally collect human judgments to further ground our results. In this section, we first describe our human experiment before quantitatively analyzing our results.

Human clustering: We conduct a crowdsourced experiment to measure how well humans can disambiguate thematically different cities. Filling in the entire 55×55 adjacency matrix with human similarity judgments is expensive and time-consuming. Thus, we instead design a proxy "odd-one-out" task for collecting human judgments: given three city descriptions, two of which come from the same ground-truth thematic group and the other from a different group, workers are asked to identify the *intruder* city. We use the Figure Eight crowdsourcing platform[5] to collect three annotations each for 100 different city triples. Our interface initially displays only the first and last sentences of each city's description; workers can optionally click to reveal the full description. As workers are likely unfamiliar with *Invisible Cities* and its different thematic groups, this crowdsourced task provides a fair comparison to our computational approaches.

4.1 Quantitative comparison

We compare clusters computed on different representations using community purity; additionally, we compare these computational methods to humans by their accuracy on the odd-one-out task.

---

[1] The book contains a vocabulary of 5,372 word types, and the average length of a city description is 380 tokens.

[2] Using other composition functions such as the span representation of Peters et al. (2018b) had little impact on the learned clusters.

[3] The choice of distance metric (e.g., cosine, word mover) did not meaningfully impact our results.

[4] Purity ranges between 0 and 1, and a larger purity indicates a higher degree of agreement.

[5] Workers were restricted to English-speaking countries and paid $0.30 per judgment.

Purity of learned clusters: City representations computed using language model-based representation (ELMo and BERT) achieve significantly higher purity than a clustering induced from random representations, indicating that there is at least some meaningful coherence to Calvino's thematic groups (first row of Table 1). ELMo representations yield the highest purity among the three methods, which is surprising as BERT is a bigger model trained on data from books (among other domains). Both ELMo and BERT outperform GloVe, which intuitively makes sense because the latter do not model the order or structure of the words in each description.

Comparison to humans: While the purity of our methods is higher than that of a random clustering, it is still far below 1. To provide additional context to these results, we now switch to our "odd-one-out" task and compare directly to human performance. For each triplet of cities, we identify the intruder as the city with the maximum Euclidean distance from the other two. Interestingly, crowd workers achieve only slightly higher accuracy than ELMo city representations; their inter-annotator agreement is also low,[6] which indicates that close reading to analyze literary coherence between multiple texts is a difficult task, even for human annotators. Overall, results from both computational and human approaches suggests that the author-assigned labels are not entirely arbitrary, as we can reliably recover some of the thematic groups.

5 Examining the learned clusters

Our quantitative results suggest that while vector-based city representations capture some thematic similarities, there is much room for improvement. In this section, we first investigate whether the learned clusters provide evidence for any arguments put forth by literary critics on the novel. Then, we explore possible reasons that the learned clusters deviate from Calvino's.

Do learned clusters support existing analyses? The argument that *cities of desire* constitute a particularly coherent thematic group (Buitendijk, 2018) is partially supported by our clustering results. Three of the five *cities of desire* are grouped into the same cluster using BERT (two for ELMo), which makes it one of the most "internally coherent" groups. Similarly, some literary critics along with Calvino himself (Calvino, 2004) describe the *thin cities* as a fairly arbitrary group, which is supported by our results: when using BERT, no two *thin cities* are grouped into the same cluster. However, Calvino also suggests that the *cities of memory* group is a "fundamental substance" of the book and therefore should be highly coherent. Our computational methods cannot pick up this theme, instead scattering all *cities of memory* into different clusters.

Why do computers disagree with Calvino? In cases where the learned clusters deviate from the opinions of Calvino or literary critics, identifying the cause of the discrepancy is difficult: our computational methods are flawed, but there is also no one "correct" literary interpretation. Here we qualitatively analyze some of the learned clusters in an attempt to understand why the algorithm arrived at a particular assignment. First, we examine two cities from different thematic groups, *Beersheba* from "cities and the sky" and *Valdrada* from "cities and eyes", that belong to the same learned cluster (and are each other's nearest neighbors). The first two paragraphs of *Beersheba* describe a noble city "suspended in the heavens" with an identical but immoral "fecal" city underground, while the remaining paragraphs focus on the heavenly city. The description of *Valdrada*, which is built on a lake, shares this theme of twin cities: arriving travelers see "two cities: one erect above the lake, and the other reflected, upside down". While Calvino likely classified *Beersheba* based on its location in the sky, the two cities share undeniable thematic similarities. Rerunning the clustering algorithm after removing the first two paragraphs of *Beersheba* results in each city being assigned to a different cluster, which supports our hypothesis.

Another interesting case is the previously-mentioned "thin cities", supposedly bound together by airy and ambiguous themes (Knowles, 2015), which Calvino (2004) states were written after all of the other cities and are more incoherent than the other groups. While BERT does not group any thin cities together, ELMo categorizes *Isaura* and *Armilla* into the same learned cluster. The two cities appear largely dissimilar: *Isaura* is a city with a thousand wells dug by its inhabitants, while *Armilla* is an "unfinished" city without walls, ceilings, or floors. However, both cities' descriptions

---

[6] Fleiss $\kappa = 0.14$, indicating slight agreement, and two or more workers agreed on the intruder only 64% of the time.

mention supernatural beings living underground. In *Isaura*, some people believe "gods live in the depths" and "in the black lake that feeds the underground streams", while the last paragraph of *Armilla*'s description conjectures that it is "in the possession of nymphs and naiads" who "travel along underground veins". Removing these descriptions on underground gods and nymphs and rerunning our clustering algorithm yields a new assignment in which each of these cities belongs to different clusters.

When do humans and computers agree? Our computational approach yields generally comparable accuracies and more consistent results than human annotators in the "odd-one-out" task. On cities with concrete themes such as sky and trading, our approach with BERT and ELMo obtains accuracy of 0.44 and 0.45 respectively, (0.47 and 0.48 for humans). ELMo also performs on par with humans in some case: for example, humans achieve an accuracy of 42% on "cities and eyes", compared to ELMo's 43%. On groups where the theme word frequently occurs in the passage, such as "eyes", our approach even slightly outperforms the human readers. However, human readers are better at recognizing abstract intangible topics, such as memory.

6 Related work

Most previous work within the NLP community applies distant reading (Jockers, 2013) to large collections of books, focusing on modeling different aspects of narratives such as plots and event sequences (Chambers and Jurafsky, 2009; McIntyre and Lapata, 2010; Goyal et al., 2010; Eisenberg and Finlayson, 2017), characters (Bamman et al., 2014; Iyyer et al., 2016; Chaturvedi et al., 2016, 2017), and narrative similarity (Chaturvedi et al., 2018). In the same vein, researchers in computational literary analysis have combined statistical techniques and linguistics theories to perform quantitative analysis on large narrative texts (Michel et al., 2011; Franzosi, 2010; Underwood, 2016; Jockers and Kirilloff, 2016; Long and So, 2016), but these attempts largely rely on techniques such as word counting, topic modeling, and naive Bayes classifiers and are therefore not able to capture the meaning of sentences or paragraphs (Da, 2019). While these works discover general patterns from multiple literary works, we are the first to use cutting-edge NLP techniques to engage with specific literary criticism about a single narrative.

There has been other computational work that focuses on just a single book or a small number of books, much of it focused on network analysis: Agarwal et al. (2013) extract character social networks from *Alice in Wonderland*, while Elson et al. (2010) recover social networks from 19th century British novels. Wallace (2012) disentangles multiple narrative threads within the novel *Infinite Jest*, while Eve (2019) provides several automated statistical methods for close reading and test them on the award-winning novel *Cloud Atlas* (2004). Compared to this work, we push further on modeling the content of the narrative by leveraging pretrained language models.

7 Conclusion

Our work takes a first step towards computationally engaging with literary criticism on a single book using state-of-the-art text representation methods. While we demonstrate that NLP techniques can be used to support literary analyses and obtain new insights, they also have clear limitations (e.g., in understanding abstract themes). As text representation methods become more powerful, we hope that (1) computational tools will become useful for analyzing novels with more conventional structures, and (2) literary criticism will be used as a testbed for evaluating representations.

Acknowledgement

We thank the anonymous reviewers for their insightful comments. Additionally, we thank Nader Akoury, Garrett Bernstein, Chenghao Lv, Ari Kobren, Kalpesh Krishna, Saumya Lal, Tu Vu, Zhichao Yang, Mengxue Zhang and the UMass NLP group for suggestions that improved the paper's clarity, coverage of related work, and analysis experiments.

References

A. Agarwal, A. Kotalwar, and O. Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *In Proceedings of the Sixth International Joint Conference on Natural Language Processing,*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *ACL*.

Roland Barthes. 1994. 11 the death of the author. *Media Texts, Authors and Readers: A Reader*, page 166.

William Benzon and David G. Hays. 1976. Computational linguistics and the humanist. *Computers and the Humanities*, 10(5):265–274.

Harold Bloom. 2002. *Bloom's Major Short Story Writers Italo Calvino*. Chelsea House Publishers.

Filip De Boeck. 2004. *Kinshasa: Tales of the Invisible City*. Leuven University Press.

Laurence Breiner. 1988. Italo calvino: The place of the emperor in "invisible cities". *Modern Fiction Studies*, 34(4):559–573.

Tomas Buitendijk. 2018. Port cities and desire in the work of italo calvino. *Port Towns and Urban Cultures*, (Article).

Italo Calvino. 2004. On "invisible cities. *Columbia: A Journal of Literature and Art*, (40):177–182.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *In Proceedings of the Joint Conference of ACL and AFNLP*.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daume III. 2017. Unsupervised learning of evolving relationships between literary characters. In *n Proceedings of the Thirty First AAAI Conference on Artificial Intelligence*.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daume III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence,*.

Snigdha Chaturvedi, Shashank Srivastava, and Dan Roth. 2018. Where have i heard this story before?: Identifying narrative similarity in movie remakes. In *NAACL-HLT*.

Nan Z. Da. 2019. The computational case against computational literary studies. *Critical Inquiry 45*, 23567.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

J.D. Eisenberg and M. A. Finlayson. 2017. A simpler and more generalizable story detector using verb and character features. In *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

D. Elson, N. Dames, and K. McKeown. 2010. Extracting social networks from literary fiction. In *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Martin Paul Eve. 2019. *Close Reading with Computers: Textual Scholarship, Computational Formalism, and David Mitchell's Cloud Atlas*. Library of Congress Cataloging-in-Publication Data. 9781503609365.

Robert Franzosi. 2010. *Quantitative Narrative Analysis*. Library of Congress Cataloging-in-Publication Data. SAGE Publication.

Amit Goyal, Ellen Riloff, and Hal Daume III. 2010. Automatically producing plot unit representations for narrative text. In *In Proceedings of Empirical Methods in Natural Language Processing*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jor dan L. Boyd-Graber, and Hal Daume III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *NAACL-HLT*.

Matthew Jockers and Gabi Kirilloff. 2016. Understanding gender and character agency in the nineteenth-century novel,. *Journal of Cultural Analytics*, culturalanalytics .org/2016/12/understanding-gender-and-character-agency-in-the-19th-century-novel/.

Matthew L. Jockers. 2013. *Macroanalysis: Digital Methods and Literary History*. Topics in the Digital Humanities. University of Illinois Press.

Dominick Knowles. 2015. A redemption of meaning in three novels by italo calvino. *Digital Commons at Ursinus College*, English Honor Papers(2).

Hoyt Long and Richard Jean So. 2016. Literary pattern recognition: Modernism be- tween close reading and machine learning. *Critical Inquiry 42*, 23567.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *ACL*.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Peter Norvig Dan Clanc and, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science (New York, N.Y.)*, 331(6014), 176182. doi:10.1126/science.1199644.

Mark Newman. 2006. Modularity and community structure in networks. *PNAS*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *North American Association for Computational Linguistics*.

Matthew E. Peters, Mark Neumann, Luke S. Zettlemoyer, and Wen tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *In Proceedings of Empirical Methods in Natural Language Processing*.

Carolyn Springer. 1985. Textual geography: The role of the reader in "invisible cities". *Modern Languages*, 15(4):289–299.

Evelyne Teichert. 1985. Words about nothing: Writing the ineffable in calvino and ma yuan. *Comparative Literature PhD Thesis, University of British Columbia*.

Ted Underwood. 2016. The life cycle of genres. *Journal of Cultural Analytics*, culturalanalytics.org/2016/05/the-life-cycles-of-genres/.

Mario Vrbani. 2012. A dream of the perfect map calvinos invisible cities. *The Zone and Zones - Radical Spatiality in our Times*, (2).

Byron C Wallace. 2012. Multiple narrative disentanglement: Unraveling infinite jest. In *North American Association for Computational Linguistics*.

William K Wimsatt and Monroe C Beardsley. 1946. The intentional fallacy. *The Sewanee Review*, 54(3).

PAWS: Paraphrase Adversaries from Word Scrambling

Yuan Zhang **Jason Baldridge** **Luheng He**
Google AI Language
{zhangyua,jridge,luheng}@google.com

Abstract

Existing paraphrase identification datasets lack sentence pairs that have high lexical overlap without being paraphrases. Models trained on such data fail to distinguish pairs like *flights from New York to Florida* and *flights from Florida to New York*. This paper introduces PAWS (**P**araphrase **A**dversaries from **W**ord **S**crambling), a new dataset with 108,463 well-formed paraphrase and non-paraphrase pairs with high lexical overlap. Challenging pairs are generated by controlled word swapping and back translation, followed by fluency and paraphrase judgments by human raters. State-of-the-art models trained on existing datasets have dismal performance on PAWS (<40% accuracy); however, including PAWS training data for these models improves their accuracy to 85% while maintaining performance on existing tasks. In contrast, models that do not capture non-local contextual information fail even with PAWS training examples. As such, PAWS provides an effective instrument for driving further progress on models that better exploit structure, context, and pairwise comparisons.

1 Introduction

Word order and syntactic structure have a large impact on sentence meaning. Even small perturbation in word order can completely change interpretation. Consider the following related sentences.

(1) Flights from New York to Florida.
(2) Flights to Florida from NYC.
(3) Flights from Florida to New York.

All three have high bag-of-words (BOW) overlap. However, (2) is a paraphrase of (1), while (3) has a very different meaning from (1).

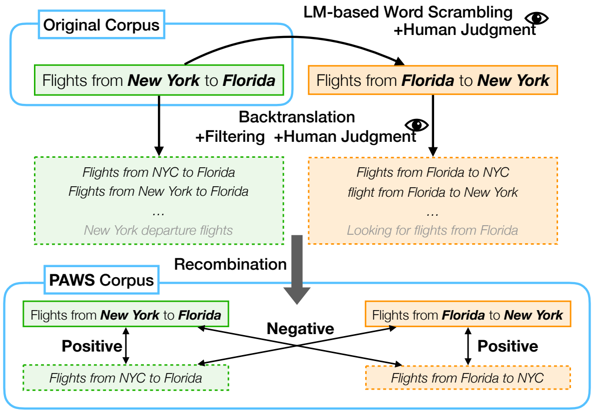


Figure 1: PAWS corpus creation workflow.

Existing datasets lack non-paraphrase pairs like (1) and (3). The Quora Question Pairs (QQP) corpus contains 400k real world pairs, but its negative examples are drawn primarily from related questions. Few have high word overlap, and of the ∼1,000 pairs with the same BOW, only 20% are not paraphrases. This provides insufficient representative examples to evaluate models' performance on this problem, and there are too few examples for models to learn the importance of word order. Table 1 shows that models trained on QQP are inclined to mark any sentence pairs with high word overlap as paraphrases despite clear clashes in meaning. Models trained or evaluated with only this data may not perform well on real world tasks where such sensitivity is important.

To address this, we introduce a workflow (outlined in Figure 1) for generating pairs of sentences that have high word overlap, but which are balanced with respect to whether they are paraphrases or not. Using this process, we create PAWS (**P**araphrase **A**dversaries from **W**ord **S**crambling), a dataset constructed from sentences in Quora and

Proceedings of NAACL-HLT 2019, pages 1298–1308
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

	Sentence 1	Sentence 2	Gold	BOW	BERT	BERT+ PAWS
(1)	Can a bad person become good?	Can a good person become bad?	**N**	Y	Y	**N**
(2)	Which is the cheapest flight from anywhere in South America to Europe?	Which is the cheapest flight from anywhere in Europe to South America?	**N**	Y	**N**	**N**
(3)	"Taunton Castle" was on August 1 in Rio de Janeiro and on October 31 in Penang.	"Taunton Castle" was at Penang on 1 August and Rio de Janeiro on 31 October.	**N**	Y	Y	**N**
(4)	Although interchangeable, the body pieces on the 2 cars are not similar.	Although similar, the body parts are not interchangeable on the 2 cars.	**N**	Y	Y	**N**
(5)	Katz was born in Sweden in 1947 and moved to New York City at the age of 1.	Katz was born in 1947 in Sweden and moved to New York at the age of one.	**Y**	**Y**	**Y**	**Y**
(6)	It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.	That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.	**N**	Y	Y	**N**

Table 1: Paraphrase/Non-paraphrase (Y/N) pairs with high bag-of-words (BOW) overlap. (1)-(5) are drawn from PAWS, and (6) is from Mitchell and Lapata (2008). Both a simple BOW and the state-of-the-art BERT (Devlin et al., 2018) models, if trained/fine-tuned on the Quora Question Pairs (QQP) corpus, classify all of them (with one exception) as paraphrases (Y). A BERT model fine-tuned on both QQP and PAWS examples (BERT+PAWS), however, is able to get them correct.

Wikipedia. Examples are generated from controlled language models and back translation, and given five human ratings each in both phases. A final rule recombines annotated examples and balances the labels. Our final PAWS dataset will be released publicly with 108,463 pairs at `https://g.co/dataset/paws`.

We show that existing state-of-the-art models fail miserably on PAWS when trained on existing resources, but some perform well when given PAWS training examples. BERT (Devlin et al., 2018) fine-tuned on QQP achieves over 90% accuracy on QQP, but only 33% accuracy on PAWS data in the same domain. However, the accuracy on PAWS boosts to 85% by including 12k PAWS training pairs (without reducing QQP performance). Table 1 also shows that the new model is able to correctly classify challenging pairs. Annotation scale is also important: our learning curves show strong models like BERT improve with tens of thousands of training examples.

Our experimental results also demonstrate that PAWS effectively measures sensitivity of models to word order and structure. Unlike BERT, a simple BOW model fails to learn from PAWS training examples, demonstrating its weakness at capturing non-local contextual information. Our experiments show that the gains from PAWS examples correlate with the complexity of models.

2 Related Work

Existing data creation techniques have focused on collecting paraphrases, e.g. from co-captions for images (Lin et al., 2014), tweets with shared URLs (Lan et al., 2017), subtitles (Creutz, 2018), and back translation (Iyyer et al., 2018). Unlike all previous work, we emphasize the collection of challenging negative examples.

Our work closely relates to the idea of crafting adversarial examples to break NLP systems. Existing approaches mostly focused on adding label-preserving perturbations to inputs, but with the effect of distracting systems from correct answers. Example perturbation rules include adding noise to inputs (Jia and Liang, 2017; Chen et al., 2018), word replacements (Alzantot et al., 2018; Ribeiro et al., 2018), and syntactic transformation (Iyyer et al., 2018). A notable exception is Glockner et al. (2018): they generated both entailment and contradiction examples by replacing words with their synonyms or antonyms. Our work presents two main departures. We propose a novel method that generates challenging examples with balanced class labels and more word reordering variations than previous work. In addition, we release to public a large set of 108k example pairs with high-quality human labels. We believe the new dataset will benefit future research on both adversarial example generation and improvement of model robustness.

In our work, we demonstrate the importance of capturing non-local contextual information in the problem of paraphrase identification. This relates to prior work on probing sentence representations for their linguistic properties, such as how much syntactic information is encoded in representations (Conneau et al., 2018; Tenney et al.,

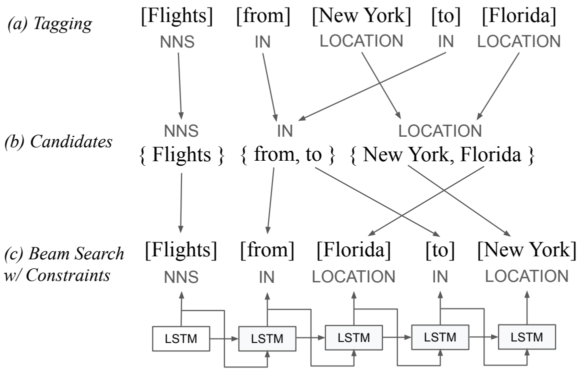


Figure 2: Illustration of the generation method in three steps. (a) Tag words and phrases with part-of-speech (POS) and named entities. (b) Build candidate sets by grouping words and phrases with the same tag. (c) Under the constraints of tag sequence template and candidate sets, find sentences with high language model scores using beam search.

2019; Ettinger et al., 2018). There also exists prior work that directly uses structural information in modeling (Filice et al., 2015; Liu et al., 2018). All these prior approaches were evaluated on existing datasets. In contrast, we perform studies on PAWS, a new dataset that emphasizes the importance of capturing structural information in representation learning. While developing new models is beyond the scope of this paper, this new dataset can facilitate research in this direction.

3 PAWS Example Generation

We define a **PAWS pair** to be a pair of sentences with high bag-of-words (BOW) overlap but different word order. In the Quora Question Pairs corpus, 80% of such pairs are paraphrases. Here, we describe a method to automatically generate non-trivial and well-formed PAWS pairs from real-world text in any domain (this section), and then have them annotated by human raters (Section 4).

Our automatic generation method is based on two ideas. The first swaps words to generate a sentence pair with the same BOW, controlled by a language model. The second uses back translation to generate paraphrases with high BOW overlap but different word order. These two strategies generate high-quality, diverse PAWS pairs, balanced evenly between paraphrases and non-paraphrases.

3.1 Word Swapping

Our first phase generates well-formed sentences by swapping words in real world text. Most text generation models rely on large amount of training data (Iyyer et al., 2018; Guu et al., 2018; Gupta et al., 2018; Li et al., 2018), which is unfortunately not available in our case. We thus propose a novel generation method based on language modeling and constrained beam search. The goal is to find a sentence that achieves high language model score as well as satisfying all constraints. High scores indicate that generated sentences are natural and well-formed, and constraints ensure generated pairs have the same BOW.

Figure 2 illustrates the generation procedure. First, given an input sentence, a CRF-based part-of-speech tagger tags each word. We further detect person names, locations, and organizations using a named entity recognizer, and replace POS with entity tags if probability scores are above 95%.[1] The sequence of tags of words and phrases form a template for the input.

Our beam search method then fills in each slot of the template from left to right, scoring each state by a language model trained on one billion words (Chelba et al., 2014). The candidate words and phrases for each slot are drawn from the input based on its tag. In Figure 2, for example, the second slot must be filled with a LOCATION from two candidate *New York* and *Florida*. Candidates are drawn without replacement so the generated sentence and the input have exactly the same bag-of-words. Note that this template-based constraint is more restrictive than the BOW requirement, but we choose it because it significantly reduces the search space. With this constraint, the method achieves high generation quality without a large beam. In practice, beam size is set to 100, which produces near-optimal results in most cases.

Let s' be the best sentence in the beam other than the input sentence s, and $LM(\cdot)$ be their log-likelihood by the language model. We take (s, s') as a good word-swapping pair if $LM(s') \geq LM(s) - t$.[2] We manually pick the threshold t=3.0 for a good balance between generation quality and coverage. Examples (1) and (2) in Table 2 are representative examples from this generation method.

---

[1] We pick this threshold to achieve about 95% precision.

[2] In a preliminary stage, we noticed that many pairs were simply a permutation of a list, like "A and B" changed to "B and A". For the diversity of the dataset, 99% of these are pruned via hand-crafted, heuristic rules.

	Sentence 1	Sentence 2	Generation Type
(1)	Can a **bad** person become **good**?	Can a **good** person become **bad**?	Adjective swap
(2)	**Jerry** looks over **Tom**'s shoulder and gets punched.	**Tom** looks over **Jerry**'s shoulder and gets punched.	Named entity swap
(3)	The team also toured in Australia **in 1953**.	**In 1953,** the team also toured in Australia.	Temporal phrase swap
(4)	Erikson **formed** the rock band Spooner with two fellow musicians.	Erikson **founded** the rock band Spooner with two fellow musicians.	Word replacement

Table 2: Examples of typical types of generation. (1) and (2) are from the word swapping method, while (3) and (4) are from the back translation method. Boldface indicates changes in each example.

3.2 Back Translation

Because word order impacts meaning, especially in English, the swapping method tends to produce non-paraphrases. Our preliminary results showed that the distribution of paraphrase to non-paraphrases from this method is highly imbalanced (about 1:4 ratio). However, we seek to create a balanced dataset, so we use an additional strategy based on back translation—which has the opposite label distribution and also produces greater diversity of paraphrases while still maintaining a high BOW overlap.

The back translation method takes a sentence pair and label (s_1, s_2, l) as input. For each sentence, the top-k translations are obtained from an English-German neural machine translation model (NMT); then each of these is translated back to English using another German-English NMT model, providing a resulting top-k results. We chose German as the pivot language because it produced more word reordering variations than other languages and the translation quality was good. Both models have the same architecture (Wu et al., 2016) and are trained on WMT14. This results in k^2 back translations before deduplication. We chose k=5. To obtain more pairs with the PAWS property, we further filter back translations by their BOW similarities to the input and their word-order inversion rates, as described below.

We define BOW similarity as the cosine similarity α between the word count vectors of a sentence pair. Pairs generated from the swapping strategy have score $\alpha = 1.0$, but here we relax the threshold to 0.9 because it brings more data diversity and higher coverage, while still generating paraphrases of the input with high quality.

To define the word-order inversion rate, we first compute word alignments between a sentence pair in a heuristic way by assuming they are one-to-one mapping and are always monotonic. For example, if the first sentence has three instances of *dog* and the second has two, we align the first two instances of *dog* in the same order and skip the third one. The inversion rate is then computed as the ratio of cross alignments. Figure 3 is an example pair with six alignments. There are 15 alignment pairs in total and 9 of them are crossed, e.g. alignments of *on* and *married*. The inversion rate of this example is therefore $9/15 = 0.6$. We sample back translation results such that at least half of the pairs have inversion rate over 0.02; this way, the final selected pairs cover interesting transformations of both word-order changes and word replacement. Examples (3) and (4) in Table 2 are representative examples from back translation.

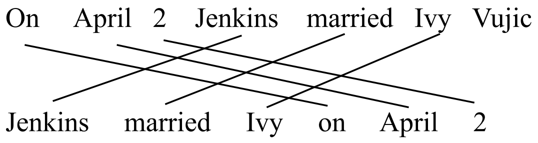


Figure 3: An example of how to compute inversion rate.

Label Balancing Figure 1 illustrates the process of constructing the final label-balanced set based on human annotations. The set first includes all pairs from back translation, which are mostly paraphrases. For each labeled pair (s_1, s_2) from swapping and a labeled pair (s_1, s'_1) from back translation, the set further includes the pair (s_2, s'_1) based on the rules: (1) (s_2, s'_1) is paraphrase if both (s_1, s_2) and (s_1, s'_1) are paraphrases; (2) (s_2, s'_1) is non-paraphrase if exactly one of (s_1, s_2) and (s_1, s'_1) is non-paraphrase; (3) otherwise (s_2, s'_1) is not included because its label is unknown. We also consider pairs (s'_2, s_1) and (s'_2, s'_1) in the similar way if s'_2 is a back translation of s_2 with human labels.

4 PAWS Dataset

Using the example generation strategies described in Section 3 combined with human paraphrase an-

	Quora	Wikipedia
# Raw pairs	16,280	50,000
Sentence correction		
# Accepted pairs	10,699	39,903
# Fixed pairs	3,626	7,387
# Rejected pairs	1,955	2,710
Paraphrase identification		
Total # pairs	14,325	47,290
paraphrase	4,693	5,725
non-paraphrase	9,632	41,565
Human agreement	92.0%	94.7%
After post-filtering		
Total # pairs	12,665	43,647
Human agreement	95.8%	97.5%

Table 3: Detailed counts for examples created via the swapping strategy, followed by human filtering and paraphrase judgments.

notations, we create a large new dataset, **PAWS** that contains both paraphrase and non-paraphrase pairs that have both high bag-of-words overlap and word reordering. Source sentences are drawn from both the Quora Question Pairs (QQP) corpus (Iyer et al., 2017) and Wikipedia.[3] From these, we produce two datasets, PAWS_{QQP} and $\text{PAWS}_{\text{Wiki}}$.

We start by producing swapped examples from both QQP and Wikipedia. Both sources contain naturally occurring sentences covering many topics. On both corpora only about 3% of candidates are selected for further processing—the rest are filtered because there is no valid generation candidate that satisfies all swapping constraints or because the language model score of the best candidate is below the threshold. The remaining pairs (16,280 for QQP and 50k for Wikipedia) are passed to human review.

Sentence correction The examples generated using both of our strategies are generally of high quality, but they still need to be checked with respect to grammar and coherence. Annotators evaluate each generated sentence without seeing its source sentence. The sentence is accepted as is, fixed, or rejected. Table 3 shows the number of pairs of each action on each domain. Most of fixes are minor grammar corrections like *a apple→an apple*. Accepted and fixed sentences are then passed to the next stage for paraphrase annotation.

Total # back translation pairs	26,897
paraphrase	25,521
non-paraphrase	1,376
Human agreement	94.8%

Table 4: Paraphrase judgments on example pairs generated by back translation on Wikipedia sentences.

Overall 88% of generated examples passed the human correction phase on both domains.

Paraphrase identification Sentence pairs are presented to five annotators, each of which gives a binary judgment as to whether they are paraphrases or not. We choose binary judgments to make our dataset have the same label schema as the QQP corpus. Table 3 shows aggregated annotation statistics on both domains, including the number of paraphrase (positive) and non-paraphrase (negative) pairs and human agreement, which is the percentage ratio of agreement between each individual label and the majority vote of five labels on each example pair. Overall, human agreement is high on both Quora (92.0%) and Wikipedia (94.7%) and each label only takes about 24 seconds. As such, answers are usually straightforward to human raters.

To ensure the data is comprised of clearly paraphrase or non-paraphrase pairs, only examples with four or five raters agreeing are kept.[4] An example of low agreement is *Why is the 20th-century music so different from the 21st music?* v.s. *Why is the 21st century music so different from the 20th century music?*, where three out of five raters gave negative labels on this pair. The bottom block of Table 3 shows the final number of pairs after this filtering, and human agreement further goes up to over 95%. Finally, source and generated sentences are randomly flipped to mask their provenance.

The swapping strategy generally produces non-paraphrase examples—67% for QQP and 88% for Wikipedia. Because (a) the label imbalance is less pronounced for QQP and (b) NMT models perform poorly on Quora questions due to domain mismatch, we only apply the back translation strategy to Wikipedia pairs. Doing so creates 26,897 candidate example pairs after filtering. As before, each pair is rated by five annotators on the paraphrase identification task.[5] Table 4 shows that

---

[3] `https://dumps.wikimedia.org`

[4] We exclude low agreement pairs from our experiments, but we include them in our data release for further study.

[5] Sentence correction was not necessary for these because

	Train	Dev	Test	Yes%
PAWS$_{\text{QQP}}$	11,988	677	–	31.3%
PAWS$_{\text{Wiki}}$	49,401	8,000	8,000	44.2%
PAWS$_{\text{Wiki-Swap}}$	30,397	–	–	9.6%

Table 5: Counts of experimental split for each PAWS dataset. The final column gives the proportion of paraphrase (positive) pairs. There are 108,463 PAWS pairs in total.

most of the examples (94.9%) are paraphrases (as expected), with high human agreement (94.8%). Finally, we expand the pairs using the the rules described in Section 3.2.

Table 5 provides counts for each split in the final PAWS datasets. The training portion of PAWS$_{\text{QQP}}$ is a subset of the QQP training set; however, PAWS$_{\text{QQP}}$'s development set is a subset of both QQP's development and test sets because there are only 677 pairs. PAWS$_{\text{Wiki}}$ randomly draws 8,000 pairs for each of its development and test sets and takes the rest as its training set, with no overlap of source sentences across sets. Finally, any trivial pairs with identical sentences from development and test sets are removed.[6] The final PAWS$_{\text{QQP}}$ has a total of 12,665 pairs (443k tokens), where 31.3% of them have positive labels (paraphrases). PAWS$_{\text{Wiki}}$ has a total of 65,401 pairs (2.8m tokens), where 44.2% of them are paraphrases.

Note that we have human annotations on 43k pairs generated by the word swapping method on Wikipedia, but 30k of them have no back translation counterparts and therefore they are not included in our final PAWS$_{\text{Wiki}}$ dataset. Nevertheless, they are high-quality pairs with manual labels, so we include them as an auxiliary training set (PAWS$_{\text{Wiki-Swap}}$ in Table 5), and empirically show its impact in Section 6.

Unlabeled PAWS$_{\text{Wiki}}$ In addition to the fully labeled PAWS$_{\text{Wiki}}$ dataset, we also construct an unlabeled PAWS$_{\text{Wiki}}$ set at large scale. The idea is to simply treat all pairs from word swapping as non-paraphrases and all pairs from back translation as paraphrase, and construct the dataset in the same way as labeled PAWS$_{\text{Wiki}}$. The result is a total of 656k pairs with silver labels. We show empirically the impact of using this silver set in pre-training in Section 6.

	BOW	BiLSTM & ESIM	DecAtt	DIIN & BERT
Non-local context	×	✓	×	✓
Word interaction	×	×	✓	✓

Table 6: Complexity of each evaluated model.

5 Evaluated Models

PAWS is designed to probe models' ability to go beyond recognizing overall sentence similarity or relatedness. As noted in the introduction, models—even the best avaliable—trained on existing resources tend to classify any example with high BOW overlap as a paraphrase. Can any of these models learn finer structural sensitivity when provided with PAWS examples as part of their training?

We consider six different models that cover a wide range of complexity and expressiveness: two baseline encoders and four recent advanced models that achieved state-of-the-art or strong performance on paraphrase identification. Table 6 summarizes the models with respect to whether they represent non-local contexts or support cross-sentential word interaction.

The baseline models use cosine similarity with simple sentence encoders: a bag-of-words (**BOW**) encoder based on token unigram and bigram encodings and a bi-directional LSTM (**BiLSTM**) that produces a contextualized sentence encoding. A cosine value above .5 is taken as a paraphrase.

ESIM. The Enhanced Sequential Inference Model (Chen et al., 2017) achieved competitive performance on eight sentence pair modeling tasks (Lan and Xu, 2018). It encodes each sentence using a BiLSTM, concatenates the encodings for each sentence in the pair, and passes them through a multi-layer perceptron (MLP) for classification. The additional layers allow ESIM to capture more complex sentence interaction than cosine similarity in the baseline models.

DecAtt. The Decomposable Attention Model (Parikh et al., 2016) is one of the earliest models to introduce attention for paraphrase identification. It computes word pair interaction between two sentences and aggregates aligned vectors for final classification. This model achieved state-of-the-art results without explicitly modeling word

---

NMT generates fluent output.

[6] Such trivial examples exist because annotators sometimes fix a swapped sentence back to its source. We keep such examples in the training set (about 8% of the corpus) because otherwise a trained model would actually predict low similarity scores to identical pairs.

order. In our experiments, we show the limitations of this modeling choice on PAWS pairs.

DIIN. The Densely Interactive Inference Network (Gong et al., 2018) adopts DenseNet (Huang et al., 2017), a 2-dimensional convolution architecture, to extract high-order word-by-word interaction between n-gram pairs. This model achieved state-of-the-art performance without relying on pre-trained deep contextualized representations like ELMo (Peters et al., 2018). It outperformed ESIM and DecAtt models by a large margin on both paraphrase identification and natural language inference tasks.

BERT. The Bidirectional Encoder Representations from Transformers (Devlin et al., 2018) recently obtained new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE benchmark to 80.4% (7.6% absolute improvement). BERT involves pre-training a Transformer encoder (Vaswani et al., 2017) on a large corpus with over three billion words. This large network is then fine-tuned with just one additional output layer.

6 Experiments

We seek to understand how well models trained on standard datasets perform on PAWS pairs and to see which models are most able to learn from PAWS pairs. A strong model should improve significantly on PAWS when trained on PAWS pairs without diminishing performance on existing datasets like QQP. Overall, both DIIN and BERT prove remarkably able to adapt to PAWS pairs and perform well on both PAWS$_{\text{QQP}}$ and PAWS$_{\text{Wiki}}$ while the other models prove far less capable.

6.1 Experimental Setup

We use two metrics: classification accuracy and area-under-curve (AUC) scores of precision-recall curves. For all classification models, 0.5 is the threshold used to compute accuracy. We report results on testing sets for QQP and PAWS$_{\text{Wiki}}$, and on the development set for PAWS$_{\text{QQP}}$ (which has no test set).

For BERT, we use the implementation provided by the authors[7] and apply their default fine-tuning configuration. We use the provided BERT$_{\text{BASE}}$ pre-trained model instead of BERT$_{\text{LARGE}}$ due to GPU memory limitations. For all other models, we use our own (re-)implementations that matched reported performance on QQP. We use 300 dimensional GloVe embeddings (Pennington et al., 2014) to represent words and fix them during training.

6.2 Results

Main Results on PAWS$_{\text{QQP}}$ Table 7 summarizes results on the Quora domain. We first train models on the Quora Question Pairs (QQP) training set, and column "QQP→QQP" shows that all models achieve over 83% accuracy on QQP. However, when evaluating on PAWS$_{\text{QQP}}$, all models, including BERT, obtain abysmal accuracy under 40% (column "QQP→PAWS$_{\text{QQP}}$").

We hypothesize the performance on PAWS$_{\text{QQP}}$ relies on two factors: the number of representative training examples, and the capability of models to represent complex interactions between words in each sentence and across the sentences in the pair. To verify that, we further train models on a combination of QQP and PAWS$_{\text{QQP}}$ training sets and the last two columns of Table 7 show the results on PAWS$_{\text{QQP}}$. As expected, all models benefit from new training examples, but to different extents. Gains are much larger on state-of-the-art models like BERT, while the BOW model learns almost nothing from new examples. As a consequence, performance changes are more drastic on PAWS$_{\text{QQP}}$ than on QQP. For example, the absolute difference between BiLSTM and BERT is 4.2% on QQP, but it goes up to 27% on PAWS$_{\text{QQP}}$, which is a 60% relative reduction in error.

It is also noteworthy that adding PAWS$_{\text{QQP}}$ training examples has no negative impact to QQP performance at all. For example, a BERT model fine-tuned on QQP+PAWS$_{\text{QQP}}$ achieves the same 90.5% classification accuracy as training on QQP alone. We therefore obtain a single model that performs well on both datasets.

Main Results on PAWS$_{\text{Wiki}}$ In our second experiment we train and evaluate models on our PAWS$_{\text{Wiki}}$ dataset. Table 8 presents the results. DIIN and BERT outperform others by a substantial margin (>17% accuracy gains). This observation gives more evidence that PAWS data effectively measures models' sensitivity to word order and syntactic structure.

One interesting observation is that DecAtt performs as poorly as BOW on this dataset. This is likely due to the fact that DecAtt and BOW both consider only local context information. We there-

---

[7] `https://github.com/google-research/bert`

MODELS	QQP→QQP (Acc)	QQP→QQP (AUC)	QQP→PAWS$_{QQP}$ (Acc)	QQP→PAWS$_{QQP}$ (AUC)	QQP+PAWS$_{QQP}$→PAWS$_{QQP}$ (Acc)	QQP+PAWS$_{QQP}$→PAWS$_{QQP}$ (AUC)
BOW	83.2	89.5	29.0	27.1	30.0 (+1.0)	27.3 (+0.2)
BiLSTM	86.3	91.6	34.8	**37.9**	57.6 (+22.9)	52.3 (+14.5)
ESIM (Chen et al., 2017)	85.3	92.8	**38.9**	26.9	66.5 (+27.7)	48.1 (+17.2)
DecAtt (Parikh et al., 2016)	87.8	93.9	33.3	26.3	67.4 (+34.1)	51.1 (+24.9)
DIIN (Gong et al., 2018)	89.2	95.2	32.8	32.4	83.8 (+51.1)	77.8 (+45.5)
BERT (Devlin et al., 2018)	**90.5**	**96.3**	33.5	35.1	**85.0** (+51.5)	**83.1** (+48.0)

Table 7: **Acc**uracy (%) of classification and **AUC** scores (%) of precision-recall curves on Quora Question Pairs (**QQP**) testing set and our PAWS$_{QQP}$ development set. QQP→PAWS$_{QQP}$ indicates that models are trained on QQP and evaluated on PAWS$_{QQP}$. Other columns are defined in a similar way. QQP+PAWS$_{QQP}$ is a simple concatenation of the two training sets. Boldface numbers indicate the best accuracy for each testing scenario. Numbers in parentheses indicate absolute gains from adding PAWS$_{QQP}$ training data.

MODELS	Supervised (Acc)	Supervised (AUC)	Pretrain+Fine-tune (Acc)	Pretrain+Fine-tune (AUC)
BOW	55.8	41.1	55.6	44.9
BiLSTM	71.1	75.6	80.8	87.6
ESIM	67.2	69.6	81.9	85.8
DecAtt	57.1	52.6	55.8	45.4
+BiLSTM	68.6	70.6	88.8	92.3
DIIN	88.6	91.1	91.8	**94.4**
BERT	**90.4**	**93.7**	**91.9**	94.3

Table 8: Accuracy (%) and AUC scores (%) of different models on PAWS$_{Wiki}$ testing set. **Supervised** models are trained on human-labeled data only, while **Pretrain+Fine-tune** models are first trained on noisy unlabeled PAWS$_{Wiki}$ data and then fine-tuned on human-labeled data.

fore tested an enhancement of DecAtt by replacing its word representations with encodings from a BiLSTM encoder to capture non-local context information. The enhanced model significantly outperforms the base, yielding an 11.5% (57.1% vs. 68.6%) absolute gain on accuracy.

We further evaluate the impact of using silver PAWS$_{Wiki}$ data in pre-training, as discussed in Section 4. The last two columns of Table 8 show the results. Comparing to supervised performance, pre-training with silver data gives consistent improvements across all models except BOW and vanilla DecAtt. Perhaps surprisingly, adding silver data gives more than 10% absolute improvements on AUC scores for BiLSTM and ESIM, much higher than the gains on DIIN and BERT.

Size of Training Set To analyze how many PAWS examples are sufficient for training, we train multiple models on QQP plus different number of PAWS$_{QQP}$ examples. Figure 4 plots AUC score curves of DIIN and BERT as a function of the number of PAWS$_{QQP}$ training examples. $x = 0$ corresponds to models trained on QQP only, and the rightmost points correspond to models trained on QQP and full PAWS$_{QQP}$. Both models improve from 30% to 74% AUC scores with 6,000 PAWS$_{QQP}$ examples. Furthermore, neither curve reaches convergence, so they would likely still benefit from more PAWS training examples.

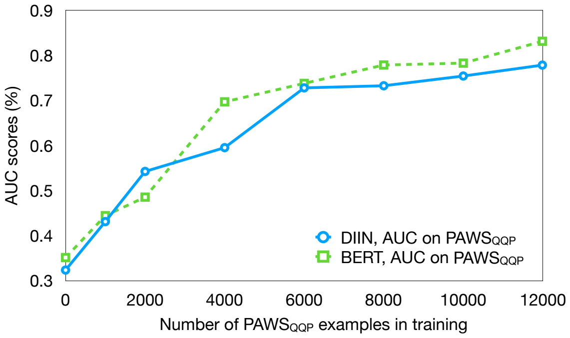


Figure 4: AUC scores (y-axis) as a function of the number of PAWS$_{QQP}$ examples in the training set (x-axis).

Cross-domain Results The PAWS datasets cover two domains: Quora and Wikipedia. Here we demonstrate that a model trained on one domain also generalizes to another domain, although not as well as training on in-domain data. Table 9 shows that a DIIN model trained on Quora (QQP+PAWS$_{QQP}$) achieves 70.5% AUC on the Wikipedia domain. This is lower than training on in-domain data (92.9%), but higher than the model trained without any PAWS data (46.0%). We also observe similar patterns when

TRAINING DATA	QQP (Test)	PAWS_{QQP} (Dev)	$\text{PAWS}_{\text{Wiki}}$ (Test)
QQP (Train)	95.2	32.4	46.0
$\text{QQP+PAWS}_{\text{QQP}}$	**95.3**	77.8	70.5
$\text{QQP+PAWS}_{\text{Wiki}}$	**95.3**	58.5	92.9
$\text{+PAWS}_{\text{Wiki-Swap}}$	**95.3**	70.6	93.5
$\text{QQP+PAWS}_{\text{QQP+Wiki}}$	95.1	87.0	93.4
$\text{+PAWS}_{\text{Wiki-Swap}}$	**95.3**	**89.9**	**93.8**

Table 9: AUC scores (%) when training DIIN models on different sets of training data. Boldface numbers indicate the best accuracy for each testing set.

training on Wikipedia ($\text{QQP+PAWS}_{\text{Wiki}}$) and testing on PAWS_{QQP}. Interestingly, using out-of-domain data also boosts in-domain performance. As Table 9 shows, training on both domains ($\text{QQP+PAWS}_{\text{QQP+Wiki}}$) leads to 9.2% absolute AUC gains on PAWS_{QQP} over the model trained only on $\text{QQP+PAWS}_{\text{QQP}}$.

The auxiliary training set on Wikipedia ($\text{PAWS}_{\text{Wiki-Swap}}$) helps further. As Table 9 shows, adding this auxiliary training set is particularly helpful to the performance on PAWS_{QQP}, yielding a 12.1% (70.6% vs 58.5%) gain on AUC when training on $\text{QQP+PAWS}_{\text{Wiki}}$. On $\text{PAWS}_{\text{Wiki}}$, this addition lifts the (no pre-training) DIIN model AUC from 91.1% (Table 8) to 93.8% (Table 9).

BERT vs DIIN Both models achieve top scores on PAWS, but interestingly, the two models disagree on many pairs and are not correlated in their errors. For example, of 687 of BERT's mistakes on the $\text{PAWS}_{\text{Wiki}}$ test set, DIIN got 280 (41%) correct. As such, performance might improve with combinations of these two existing models.

It is also worth noting that the DIIN model used in our experiments has only 590k model parameters, whereas BERT has over 100m. Furthermore, the computational cost of BERT is notably higher than DIIN. Given this, and the fact that DIIN is competitive with BERT (especially when pre-trained on noisy pairs, see Table 8), DIIN is likely the better choice in computationally constrained scenarios—especially those with strict latency requirements.

7 Conclusion

Datasets are insufficient for differentiating models if they lack examples that exhibit the necessary diagnostic phenomena. This has led, for example, to new datasets for noun-verb ambiguity (Elkahky et al., 2018) and gender bias in coreference (Webster et al., 2018; Rudinger et al., 2018; Jieyu Zhao, 2018). Our new PAWS datasets join these efforts and provide a new resource for training and evaluating paraphrase identifiers. We show that including PAWS training data for state-of-the-art models dramatically improves their performance on challenging examples and makes them more robust to real world examples. We also demonstrate that PAWS effectively measures sensitivity of models on word order and syntactic structure.

Acknowledgement

We would like to thank our anonymous reviewers and the Google AI Language team, especially Emily Pitler, for the insightful comments that contributed to this paper. Many thanks also to the Data Compute team, especially Ashwin Kakarla and Henry Jicha, for their help with the annotations

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, pages 2635–2639. ISCA.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2587–2597.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of* *the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Mathias Creutz. 2018. Open subtitles paraphrase corpus for six languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ali Elkahky, Kellie Webster, Daniel Andor, and Emily Pitler. 2018. A challenge set and methods for noun-verb ambiguity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2562–2572.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655. Association for Computational Linguistics.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *International Conference on Learning Representations.*

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI*, pages 5149–5156. AAAI Press.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL*, 6:437–450.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL-HLT*, pages 1875–1885. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Mark Yatskar Vicente Ordonez Kai-Wei Chang Jieyu Zhao, Tianlu Wang. 2018. Gender bias in coreference resolution:evaluation and debiasing methods. In *NAACL (short).*

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1224–1234.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *COLING*, pages 3890–3902. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *EMNLP*, pages 3865–3878. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft COCO: Common Objects in Context. In *CoRR*.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured alignment networks for matching sentences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1554–1564.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 236–244.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguous pronouns. In *Transactions of the ACL*, page to appear.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Cross-Corpora Evaluation and Analysis of Grammatical Error Correction Models — Is Single-Corpus Evaluation Enough?

Masato Mita[1,2]**, Tomoya Mizumoto**[1]**, Masahiro Kaneko**[3,1]**, Ryo Nagata**[4,1]**, Kentaro Inui**[2,1]
[1]RIKEN AIP, [2]Tohoku University, [3]Tokyo Metropolitan University, [4]Konan University
{masato.mita, tomoya.mizumoto}@riken.jp,
kaneko-masahiro@ed.tmu.ac.jp, nagata-naacl@ml.hyogo-u.ac.jp.,
inui@ecei.tohoku.ac.jp

Abstract

This study explores the necessity of performing cross-corpora evaluation for grammatical error correction (GEC) models. GEC models have been previously evaluated based on a single commonly applied corpus: the CoNLL-2014 benchmark. However, the evaluation remains incomplete because the task difficulty varies depending on the test corpus and conditions such as the proficiency levels of the writers and essay topics. To overcome this limitation, we evaluate the performance of several GEC models, including NMT-based (LSTM, CNN, and transformer) and an SMT-based model, against various learner corpora (CoNLL-2013, CoNLL-2014, FCE, JFLEG, ICNALE, and KJ). Evaluation results reveal that the models' rankings considerably vary depending on the corpus, indicating that single-corpus evaluation is insufficient for GEC models.

1 Introduction

Grammatical error correction (GEC) is the task of correcting various grammatical errors in a given text, which is typically written by non-native speakers. Previous studies focused on typical errors such as those in the use of articles (Han et al., 2006), prepositions (Felice and Pulman, 2008), and noun numbers (Nagata et al., 2006). Machine translation approaches are being presently applied for GEC (Junczys-Dowmunt et al., 2018; Chollampatt and Ng, 2018; Ge et al., 2018; Junczys-Dowmunt and Grundkiewicz, 2016). In these approaches, GEC is treated as a translation problem from the erroneous text to the correct text (Mizumoto et al., 2012; Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014).

However, the evaluation of GEC performance is unfortunately not complete because researchers tend to evaluate their models on a single corpus. The CoNLL-2014 shared task dataset (Ng et al., 2014) has been recently used for such evaluation.

Single-corpus evaluation may be insufficient in cases wherein a GEC model generally aims to robustly correct grammatical errors in *any* written text partly because the task difficulty varies depending on proficiency levels and essay topics. Although a model outperforms a baseline in one corpus, the model in another corpus may perform better, leading to different conclusions from what we know. This study explores the necessity of performing cross-corpora evaluation for GEC models. The performance of four recent models, namely three neural machine translation (NMT)-based models (LSTM, CNN, and transformer) and a statistical machine translation (SMT)-based model is evaluated against six learner corpora (CoNLL-2014, CoNLL-2013 (Ng et al., 2013), FCE (Yannakoudakis et al., 2011), JFLEG (Napoles et al., 2017), KJ (Nagata et al., 2011), and ICNLAE (Ishikawa, 2013)). Evaluation results show that the models' rankings considerably vary depending on the corpus. Empirical results reveal that models must be evaluated using multiple corpora from different perspectives.

The contributions of this study are as follows:

- We first explore the necessity of performing cross-corpora evaluation for GEC models.
- We empirically show that the single-corpus evaluation may be unreliable.
- Our source code is published for cross-corpora evaluation so that researchers in the community can adequately and easily evaluate their models based on multiple corpora. [1]

2 Related Work

We are motivated by the issue of robustness in the parsing community. This field pre-

---

[1] https://github.com/tomo-wb/GEC_CCE

Proceedings of NAACL-HLT 2019, pages 1309–1314
Minneapolis, Minnesota, June 2 - June 7, 2019.

viously focused on improving parsing accuracy on Penn Treebank (Marcus et al., 1993). However, robustness was largely improved by evaluation using multiple corpora including Ontonotes (Hovy et al., 2006) and Google Web Treebank (Petrov and McDonald, 2012). A situation similar to this might also occur in GEC. In other words, evaluation in GEC has relied heavily on the CoNLL-2014 benchmark, which implies that the field is overdeveloping on this dataset.

Other corpora are used for evaluation, such as KJ (Mizumoto et al., 2012) and JFLEG (Sakaguchi et al., 2017; Junczys-Dowmunt et al., 2018; Chollampatt and Ng, 2018; Ge et al., 2018; Xie et al., 2018). However, these corpora still depend on one or at most two corpora.

3 Experimental Setup

3.1 Corpora for Evaluation

Cross-corpora evaluation is discussed herein using six corpora, namely CoNLL-2014, CoNLL-2013, FCE, JFLEG, KJ, and ICNALE. The following conditions were considered when selecting corpora:

- The corpus must be used at least once in the GEC community.
- Based on the hypothesis that writers' proficiency affects the error distribution of any given text, we add a corpus with relatively low proficiency (KJ) compared to CoNLL-2014.

We explicitly describe each learner corpus as follows:

CoNLL-2014 (Ng et al., 2014), the official dataset of CoNLL-2014 shared task, is a collection of essays written by students at the National University of Singapore and is commonly used as test data for the CoNLL-2014 benchmark. This dataset contains only two essay topics.

CoNLL-2013 (Ng et al., 2013), the official dataset of CoNLL-2013 shared tasks, is commonly used as the development data for the CoNLL-2014 benchmark and contains only two essay topics.

Cambridge ESOL First Certificate in English (FCE) (Yannakoudakis et al., 2011) is a dataset containing 1,244 examination scripts of the Cambridge FCE examination. Topics and first languages (L1s) in the dataset are diversified because it contains essays for 10 topics written by non-native speakers from various countries.

JHU FLuency-Extended GUG Corpus (JFLEG) (Napoles et al., 2017) contains approximately 1,500 sentences from an English proficiency test. It contains sentences written by learners of the English language having various L1s and proficiency levels.

Konan-JIEM Learner Corpus (KJ) (Nagata et al., 2011) contains 233 essays written on 10 topics by students of a Japanese college, which are manually error-tagged and shallow-parsed.

International Corpus Network of Asian Learners of English, Written Essays (ICNALE) (Ishikawa, 2013) contains essays written by college and graduate students from ten Asian countries/regions (China, Hong Kong, Indonesia, Japan, Korea, Pakistan, the Philippines, Singapore, Taiwan, and Thailand). The original ICNALE is not error annotated. Therefore, we sampled a total number of 1,736 sentences, which are manually annotated with grammatical errors based on KJ 's annotation scheme.

Table 1 summarizes the properties of these corpora. Let N and M denote the total number of source words and sentences in a corpus, respectively. Word error rate (WER) is defined as follows:

$$\mathrm{WER} = \frac{\sum_{m=1}^{M} d(X^m, Y^m)}{\sum_{m=1}^{M} N^m}$$

where X^m denotes each source sentence, Y^m denotes each corrected sentence, and $d(X^m, Y^m)$ denotes the edit distance between X^m and Y^m using dynamic programming.

The following conclusions are derived: (1) CoNLL-2014 has narrow coverage of topics, proficiency and L1s compared with other corporas such as JFLEG and FCE. (2) Several learner corpora are available for the evaluation of GEC models. These corpora can help investigate the performance of GEC models under different conditions.

3.2 Models

The following factors are considered while selecting our model.

- The models must be recent and commonly used.

Corpus	# sent.	# refs.	WER	# topics	Multiple L1	Multiple proficiency	Public available
CoNLL-2014	1,312	2	12.35	2	No	No	Yes
CoNLL-2013	1,381	1	14.85	2	No	No	Yes
FCE	32,199	1	12.00	10	Yes	Yes	Yes
JFLEG	747	4	20.86	Many	Yes	Yes	Yes
KJ	3,081	1	13.53	10	No	No	Yes
ICNALE	1,736	1	7.64	2	Yes	Yes	No

Table 1: Properties of evaluation corpora. Yes/No indicates whether the corpus exhibits each property in terms of multiple L1, multiple proficiency and public available.

- Each model must be implemented to have a competitive performance on CoNLL-2014.

We employed the following models based on the aforementioned factors:

LSTM: We use a bi-directional LSTM in the encoder and an LSTM with an attention mechanism in the decoder. Both the encoder and the decoder comprise two layers. The LSTM hidden state and word embedding sizes are set to be 500.

CNN: We follow the previous study (Chollampatt and Ng, 2018), namely a fully convolutional encoder–decoder architecture with seven convolutional layers. The hyperparameters used in a previous study are used herein (Chollampatt and Ng, 2018).

Transformer: Transformer is the self-attention-based model proposed by Vaswani et al. (2017). Six layers are used for both the encoder and decoder along with eight attention heads. The word embedding size is set to 1024 dimensions, and the size of position-wise feed-forward networks is set to 4096 dimensions at each inner layer.

SMT: We essentially follow the idea used in a previous study (Junczys-Dowmunt and Grundkiewicz, 2016), with some key differences. Specifically, we only use English Wikipedia for language model training and only the NUS Corpus of Learner English (NUCLE) and the Lang-8 Learner Corpora (Lang-8) for translation model training to make the experimental settings equal in all models.

3.3 Experimental Settings

We use two public datasets, namely Lang-8 (Mizumoto et al., 2011) and NUCLE (Dahlmeier et al., 2013), for training. Our pre-processing and experimental setup is similar to that reported previously (Chollampatt and Ng, 2018). In particular, a subset of NUCLE (5.4K) is utilized as the development data for selecting the model; the remaining subset (1.3M) is utilized as the training data. All the models are trained, tuned, and tested in the same way. The models are tested on each test data shown in Table 1. As an evaluation metric, we use $F_{0.5}$ score computed by applying the MaxMatch scorer (Dahlmeier and Ng, 2012) and GLEU (Napoles et al., 2015). We determine the average $F_{0.5}$ and average GLEU scores of the four models, which are trained with different random initializations, following a previously reported approach (Chollampatt and Ng, 2018).

4 Cross-Corpora Evaluation

Figure 1 shows the performance of each model sorted from best to worst based on their $F_{0.5}$ score, revealing that the performance substantially varies depending on the corpus. For example, the performance of the transformer ranges from the score of $F_{0.5}$, which is as low as 36.20 on CoNLL-2013, to as high as 60.06 on JFLEG. Notably, their rankings also considerably vary. Transformer performs best on CoNLL-2014. However, it exhibits third-best performance among FCE, KJ, and ICNALE; LSTM outperforms the other models by a large margin of up to 5.3 $F_{0.5}$ points. Some examples of the model outputs are presented in Table 2 and Table 3. Some situations are successfully corrected using transformer (Table 2), whereas it failed to perform in other situations (Table 3). The reason for difference in the model rankings cannot be generally stated because it is influenced by various factors such as the learner's proficiency, essay topic, and L1. The experimental results show, however, that discussions based on the performance on CoNLL-2014 may only hold under certain conditions.

Figure 2 shows the performance measured in GLEU having a similar trend. However, their

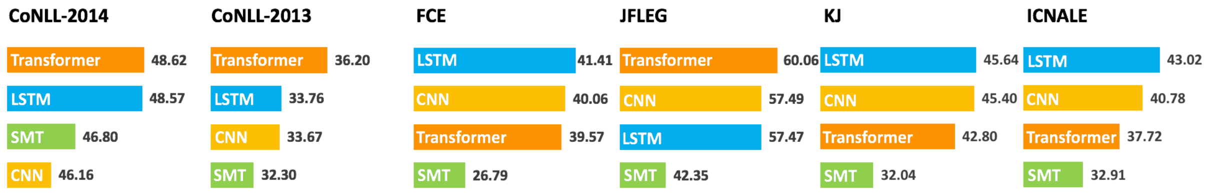


Figure 1: Average $F_{0.5}$ of the four models (trained with different random initializations), ranked best to worst.

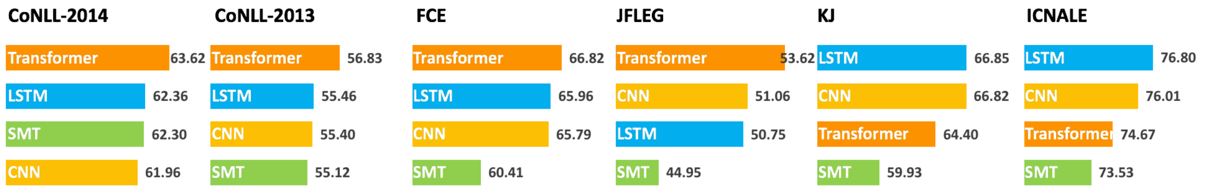


Figure 2: Average GLEU of the four models (trained with different random initializations), ranked best to worst.

rankings on FCE show different trends in Figure 1 and Figure 2. This is partly because $F_{0.5}$ and GLEU evaluate different perspectives of the models. Furthermore, evaluation data and metric must be appropriately set depending on the factors that need to be evaluated in the model.

5 Discussion

5.1 Is Diverse Single-Corpus Evaluation Sufficient?

Experimental results indicate that the benchmark single-corpus evaluation is not robust; however, more diverse corpora remain undetermined. Both JFLEG and FCE can be diverse corpora because they contain examination scripts written by language learners from all over the world. JFLEG is particularly designed to contain more diverse corpus for developing and evaluating GEC models (Napoles et al., 2017). If a diverse single-corpus evaluation suffices, the rankings of the models will remain the same. However, experimental results have shown that the model rankings on both JFLEG and FCE are different (Figure 1). Thus, single-corpus evaluation is deemed weak regardless of its diversity.

5.2 Advantage of Cross-Corpora Evaluation

This study discusses the importance of evaluating GEC models from various perspectives using multiple corpora. Multi-perspective evaluation does not necessarily mean using multiple corpora. Many aspects in a corpus can be used for analysis, such as the proficiency of the writers, essay topics, and the writer 's native language. As a case study, we evaluate and analyze the models regarding the essay WER. Table 4 shows the performance (in precision, recall, and $F_{0.5}$) of all the models when WER is the lowest (7.64 % for ICNALE) and the highest (20.86 % for JFLEG). Transformer and LSTM outperform all the other models in the highest and the lowest error-rated corpora, respectively. Experimental results show that LSTM and transformer may be more precision-oriented and recall-oriented, respectively. Further, precision-oriented models have an advantage over recall-oriented models when a given text contains several errors, and vice versa. This knowledge enables choosing a model based on the task that has to be completed.

6 Conclusion

This study explored the necessity of performing cross-corpora evaluation for GEC models, for which the performance of several GEC models was investigated against various learner corpora. Empirical evaluation results revealed that the model performance and rankings considerably vary depending on the corpus, suggesting that a single-corpus evaluation can be unreliable. Therefore, cross-corpora evaluation should be applied to GEC models. We also published our source code for the cross-corpora evaluation framework so that researchers in the community can adequately and easily evaluate their models based on multiple corpora. Our future study will further examine the robustness of several existing evaluation metrics and explore new metrics appropriate for cross-corpora and/or cross-domain evaluation.

	Sentence
Source	Hence , some *seen* it as being considerate in keeping the genetic risk of getting the disease *in* confidential .
Reference	Hence , some **see** it as being considerate in keeping the genetic risk of getting the disease **[DEL]** confidential .
1. Transformer	Hence , some **see** it as being considerate in keeping the genetic risk of getting the disease **[DEL]** confidential .
2. LSTM	Hence , some seen it as being considerate in keeping the genetic risk of getting the disease in **confidentiality** .
3. CNN	Hence , some seen it as being considerate in keeping the genetic risk of getting the disease in **confidentiality** .

Table 2: Examples of model outputs on CoNLL-2014.

	Sentence
Source	*In* that day , the time I left school was about eleven p.m .
Reference	**On** that day , the time I left school was about eleven p.m .
1. LSTM	**On** that day , the time I left school was about eleven p.m .
2. CNN	**On** that day , the time I left school was about eleven p.m .
3. Transformer	**That** day , the time I left school was about eleven p.m .

Table 3: Examples of model outputs on KJ.

WER (%)	Low (7.64)			High (20.86)		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Transformer	37.69	**37.67**	37.72	67.27	**42.05**	**60.06**
LSTM	**48.68**	29.37	**43.02**	**72.97**	31.09	57.47
CNN	44.35	30.87	40.78	70.85	32.77	57.49
SMT	40.73	18.60	32.91	67.95	16.89	42.35

Table 4: Performance in precision, recall, and $F_{0.5}$ of all models on the corpora when the WER is lowest and highest.

Acknowledgments

We are grateful to the members of the Tohoku University Natural Language Processing Laboratory as well as the anonymous reviewers for their insightful comments and suggestions.

References

Shamil Chollampatt and Hwee Tou Ng. 2018. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction. In *Proceedings of AAAI*, pages 5755–5762.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better Evaluation for Grammatical Error Correction. In *Proceedings of NAACL*, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of BEA*, pages 22–31.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of CoNLL 2014 Shared Task*, pages 15–24.

Rachele De Felice and Stephen G. Pulman. 2008. A Classifier-Based Approach to Preposition and Determiner Error Correction in L2 English. In *Proceedings of COLING*, pages 169–176.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study. *arXiv*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting Errors in English Article Usage by Non-Native Speakers. *Natural Language Engineering*, 12(2):115–129.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of NAACL*, pages 57–60.

Shin'ichro Ishikawa. 2013. The ICNALE and Sophisticated Contrastive Interlanguage Analysis of Asian learners of English. *Learner Corpus Studies in Asia and the World*, 1:91–118.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by

Data-Intensive and Feature-Rich Statistical Machine Translation. In *Proceedings of CoNLL 2014 Shared Task*, pages 25–33.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. In *Proceedings of EMNLP*, pages 1546–1556.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In *Proceedings of NAACL*, pages 595–606.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The Effect of Learner Corpus Size in Grammatical Error Correction of ESL Writings. In *Proceedings of COLING*, pages 863–872.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of IJCNLP*, pages 147–155.

Ryo Nagata, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu. 2006. A Feedback-Augmented Method for Detecting Errors in the Writing of Learners of English. In *Proceedings of COLING-ACL*, pages 241–248.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed corpus. In *Proceedings of ACL*, pages 1210–1219.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of ACL*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of EACL*, pages 229–234.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2014 Shared Task*, pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2013 Shared Task*, pages 1–12.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 Shared Task on Parsing the Web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language*.

Keisuke Sakaguchi, Matt Post, and Van Benjamin Durme. 2017. Grammatical Error Correction with Neural Reinforcement Learning. In *Proceedings of IJCNLP*, pages 366–372.

Ashish Vaswani, Noam Shazeer, Parmar Niki, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*, pages 5998–6008.

Ziang Xie, Guillaume Genthial, Andrew Y. Ng, and Dan Jurafsky. 2018. Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of NAACL*, pages 619–628.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of ACL*, pages 180–189.

Star-Transformer

Qipeng Guo[†]
qpguo16@fudan.edu.cn

Xipeng Qiu[†] [*]
xpqiu@fudan.edu.cn

Pengfei Liu[†]
pfliu14@fudan.edu.cn

Yunfan Shao[†]
yfshao15@fudan.edu.cn

Xiangyang Xue[†]
xyxue@fudan.edu.cn

Zheng Zhang[‡]
zz@nyu.edu

[†]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[†]School of Computer Science, Fudan University
[‡]New York University

Abstract

Although *Transformer* has achieved great successes on many NLP tasks, its heavy structure with fully-connected attention connections leads to dependencies on large training data. In this paper, we present *Star-Transformer*, a lightweight alternative by careful sparsification. To reduce model complexity, we replace the fully-connected structure with a star-shaped topology, in which every two non-adjacent nodes are connected through a shared relay node. Thus, complexity is reduced from quadratic to linear, while preserving the capacity to capture both local composition and long-range dependency. The experiments on four tasks (22 datasets) show that Star-Transformer achieved significant improvements against the standard Transformer for the modestly sized datasets.

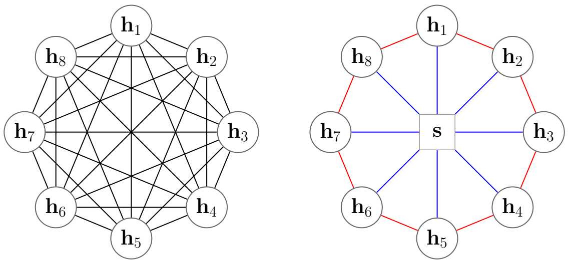


Figure 1: Left: Connections of one layer in Transformer, circle nodes indicate the hidden states of input tokens. Right: Connections of one layer in Star-Transformer, the square node is the virtual relay node. Red edges and blue edges are ring and radical connections, respectively.

1 Introduction

Recently, the fully-connected attention-based models, like Transformer (Vaswani et al., 2017), become popular in natural language processing (NLP) applications, notably machine translation (Vaswani et al., 2017) and language modeling (Radford et al., 2018). Some recent work also suggest that Transformer can be an alternative to recurrent neural networks (RNNs) and convolutional neural networks (CNNs) in many NLP tasks, such as GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-XL (Dai et al., 2019) and Universal Transformer (Dehghani et al., 2018).

More specifically, there are two limitations of the Transformer. First, the computation and memory overhead of the Transformer are quadratic to the sequence length. This is especially problematic with long sentences. Transformer-XL (Dai et al., 2019) provides a solution which achieves the acceleration and performance improvement, but it is specifically designed for the language modeling task. Second, studies indicate that Transformer would fail on many tasks if the training data is limited, unless it is pre-trained on a large corpus. (Radford et al., 2018; Devlin et al., 2018).

A key observation is that Transformer does not exploit prior knowledge well. For example, the local compositionality is already a robust inductive bias for modeling the text sequence. However, the Transformer learns this bias from scratch, along with non-local compositionality, thereby increasing the learning cost. The key insight is then whether leveraging strong prior knowledge can help to "lighten up" the architecture.

To address the above limitation, we proposed a new lightweight model named "Star-Transformer". The core idea is to sparsify the

---

[*] Corresponding Author.

[‡] work done at NYU Shanghai, now with AWS Shanghai AI Lab.

Proceedings of NAACL-HLT 2019, pages 1315–1325
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

architecture by moving the fully-connected topology into a star-shaped structure. Fig-1 gives an overview. Star-Transformer has two kinds of connections. The radical connections preserve the non-local communication and remove the redundancy in fully-connected network. The ring connections embody the local-compositionality prior, which has the same role as in CNNs/RNNs. The direct outcome of our design is the improvement of both efficiency and learning cost: the computation cost is reduced from quadratic to linear as a function of input sequence length. An inherent advantage is that the ring connections can effectively reduce the burden of the unbias learning of local and non-local compositionality and improve the generalization ability of the model. What remains to be tested is whether one shared relay node is capable of capturing the long-range dependencies.

We evaluate the Star-Transformer on three NLP tasks including Text Classification, Natural Language Inference, and Sequence Labelling. Experimental results show that Star-Transformer outperforms the standard Transformer consistently and has less computation complexity. An additional analysis on a simulation task indicates that Star-Transformer preserve the ability to handle with long-range dependencies which is a crucial feature of the standard Transformer.

In this paper, we claim three contributions as the following and our code is available on Github [1]:

- Compared to the standard Transformer, Star-Transformer has a lightweight structure but with an approximate ability to model the long-range dependencies. It reduces the number of connections from n^2 to $2n$, where n is the sequence length.

- The Star-Transformer divides the labor of semantic compositions between the radical and the ring connections. The radical connections focus on the non-local compositions and the ring connections focus on the local composition. Therefore, Star-Transformer works for modestly sized datasets and does not rely on heavy pre-training.

- We design a simulation task "Masked Summation" to probe the ability dealing with long-range dependencies. In this task, we verify that both Transformer and Star-Transformer are good at handling long-range dependencies compared to the LSTM and BiLSTM.

2 Related Work

Recently, neural networks have proved very successful in learning text representation and have achieved state-of-the-art results in many different tasks.

Modelling Local Compositionality A popular approach is to represent each word as a low-dimensional vector and then learn the local semantic composition functions over the given sentence structures. For example, Kim (2014); Kalchbrenner et al. (2014) used CNNs to capture the semantic representation of sentences, whereas Cho et al. (2014) used RNNs.

These methods are biased for learning local compositional functions and are hard to capture the long-term dependencies in a text sequence. In order to augment the ability to model the non-local compositionality, a class of improved methods utilizes various self-attention mechanisms to aggregate the weighted information of each word, which can be used to get sentence-level representations for classification tasks (Yang et al., 2016; Lin et al., 2017; Shen et al., 2018a). Another class of improved methods augments neural networks with a re-reading ability or global state while processing each word (Cheng et al., 2016; Zhang et al., 2018).

Modelling Non-Local Compositionality There are two kinds of methods to model the non-local semantic compositions in a text sequence directly.

One class of models incorporate syntactic tree into the network structure for learning sentence representations (Tai et al., 2015; Zhu et al., 2015).

Another type of models learns the dependencies between words based entirely on self-attention without any recurrent or convolutional layers, such as Transformer (Vaswani et al., 2017), which has achieved state-of-the-art results on a machine translation task. The success of Transformer has raised a large body of follow-up work. Therefore, some Transformer variations are also proposed, such as GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-XL (Dai et al., 2019) , Universal Transformer (Dehghani et al., 2018) and CN^3 (Liu et al., 2018a).

---

[1] `https://github.com/dmlc/dgl` and `https://github.com/fastnlp/fastNLP`

However, those Transformer-based methods usually require a large training corpus. When applying them on modestly sized datasets, we need the help of semi-supervised learning and unsupervised pretraining techniques (Radford et al., 2018).

Graph Neural Networks Star-Transformer is also inspired by the recent graph networks (Gilmer et al., 2017; Kipf and Welling, 2016; Battaglia et al., 2018; Liu et al., 2018b), in which the information fusion progresses via message-passing across the whole graph.

The graph structure of the Star-Transformer is star-shaped by introducing a virtual relay node. The radical and ring connections give a better balance between the local and non-local compositionality. Compared to the previous augmented models (Yang et al., 2016; Lin et al., 2017; Shen et al., 2018a; Cheng et al., 2016; Zhang et al., 2018), the implementation of Star-Transform is purely based on the attention mechanism similar to the standard Transformer, which is simpler and well suited for parallel computation.

Due to its better parallel capacity and lower complexity, the Star-Transformer is faster than RNNs or Transformer, especially on modeling long sequences.

3 Model

3.1 Architecture

The Star-Transformer consists of one relay node and n satellite nodes. The state of i-th satellite node represents the features of the i-th token in a text sequence. The relay node acts as a virtual hub to gather and scatter information from and to all the satellite nodes.

Star-Transformer has a star-shaped structure, with two kinds of connections in the: the radical connections and the ring connections.

Radical Connections For a network of n satellite nodes, there are n radical connections. Each connection links a satellite node to the shared relay node. With the radical connections, every two non-adjacent satellite nodes are two-hop neighbors and can receive non-local information with a two-step update.

Ring Connections Since text input is a sequence, we bake such prior as an inductive bias. Therefore, we connect the adjacent satellite nodes to capture the relationship of local compositions. The first and last nodes are also connected. Thus, all these local connections constitute a ring-shaped structure. Note that the ring connections allow each satellite node to gather information from its neighbors and plays the same role to CNNs or bidirectional RNNs.

With the radical and ring connections, Star-Transformer can capture both the non-local and local compositions simultaneously. Different from the standard Transformer, we make a division of labor, where the radical connections capture non-local compositions, whereas the ring connections attend to local compositions.

3.2 Implementation

The implementation of the Star-Transformer is very similar to the standard Transformer, in which the information exchange is based on the attention mechanism (Vaswani et al., 2017).

Multi-head Attention Just as in the standard Transformer, we use the scaled dot-product attention (Vaswani et al., 2017). Given a sequence of vectors $\mathbf{H} \in \mathbb{R}^{n \times d}$, we can use a query vector $\mathbf{q} \in \mathbb{R}^{1 \times d}$ to soft select the relevant information with attention.

$$\mathrm{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \quad (1)$$

where $\mathbf{K} = \mathbf{H}\mathbf{W}^K, \mathbf{V} = \mathbf{H}\mathbf{W}^V$, and $\mathbf{W}^K, \mathbf{W}^V$ are learnable parameters.

To gather more useful information from $\mathbf{H}$, similar to multi-channels in CNNs, we can use multi-head attention with k heads.

$$\mathrm{MultiAtt}(\mathbf{q}, \mathbf{H}) = (\mathbf{a}_1 \oplus \cdots \oplus \mathbf{a}_k)\mathbf{W}^O, \quad (2)$$

$$\mathbf{a}_i = \mathrm{Att}(\mathbf{q}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V), i \in [1, k] \quad (3)$$

where $\oplus$ denotes the concatenation operation, and $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}^O$ are learnable parameters.

Update Let $\mathbf{s}^t \in \mathbb{R}^{1 \times d}$ and $\mathbf{H}^t \in \mathbb{R}^{n \times d}$ denote the states for the relay node and all the n satellite nodes at step t. When using the Star-Transformer to encode a text sequence of length n, we start from its embedding $\mathbf{E} = [\mathbf{e}_1; \cdots; \mathbf{e}_n]$, where $\mathbf{e}_i \in \mathbb{R}^{1 \times d}$ is the embedding of the i-th token.

We initialize the state with $\mathbf{H}^0 = \mathbf{E}$ and $\mathbf{s}^0 = average(\mathbf{E})$.

The update of the Star-Transformer at step t can be divided into two alternative phases: (1) the update of the satellite nodes and (2) the update of the relay node.

At the first phase, the state of each satellite node $\mathbf{h}_i$ are updated from its adjacent nodes, including the neighbor nodes $\mathbf{h}_{i-1}, \mathbf{h}_{i+1}$ in the sequence, the relay node $\mathbf{s}^t$, its previous state, and its corresponding token embedding.

$$\mathbf{C}_i^t = [\mathbf{h}_{i-1}^{t-1}; \mathbf{h}_i^{t-1}; \mathbf{h}_{i+1}^{t-1}; \mathbf{e}^i; \mathbf{s}^{t-1}], \quad (4)$$

$$\mathbf{h}_i^t = \mathrm{MultiAtt}(\mathbf{h}_i^{t-1}, \mathbf{C}_i^t), \quad (5)$$

where $\mathbf{C}_i^t$ denotes the context information for the i-th satellite node. Thus, the update of each satellite node is similar to the recurrent network, except that the update fashion is based on attention mechanism. After the information exchange, a layer normalization operation (Ba et al., 2016) is used.

$$\mathbf{h}_i^t = \mathrm{LayerNorm}(\mathrm{ReLU}(\mathbf{h}_i^t)), i \in [1, n]. \quad (6)$$

At the second phase, the relay node $\mathbf{s}^t$ summarizes the information of all the satellite nodes and its previous state.

$$\mathbf{s}^t = \mathrm{MultiAtt}(\mathbf{s}^{t-1}, [\mathbf{s}^{t-1}; \mathbf{H}^t]), \quad (7)$$

$$\mathbf{s}^t = \mathrm{LayerNorm}(\mathrm{ReLU}(\mathbf{s}^t)). \quad (8)$$

By alternatively updating update the satellite and relay nodes, the Star-Transformer finally captures all the local and non-local compositions for an input text sequence.

Position Embeddings To incorporate the sequence information, we also add the learnable position embeddings, which are added with the token embeddings at the first layer.

The overall update algorithm of the Star-Transformer is shown in the Alg-1.

3.3 Output

After T rounds of update, the final states of $\mathbf{H}^T$ and $\mathbf{s}^T$ can be used for various tasks such as sequence labeling and classification. For different tasks, we feed them to different task-specific modules. For classification, we generate the fix-length sentence-level vector representation by applying a max-pooling across the final layer and mixing it with $\mathbf{s}^T$, this vector is fed into a Multiple Layer Perceptron (MLP) classifier. For the sequence labeling task, the $\mathbf{H}^T$ provides features corresponding to all the input tokens.

Algorithm 1 The Update of Star-Transformer

Input: Number of layers T, embedding of input tokens $e_1, \cdots, e_n$

```
// Initialization
$\mathbf{h}_1^0, \cdots, \mathbf{h}_n^0 \leftarrow \mathbf{e}_1, \cdots, \mathbf{e}_n$
$\mathbf{s}^0 \leftarrow average(\mathbf{e}_1, \cdots, \mathbf{e}_n)$
for $t$ from $1$ to $T$ do
    // update the satellite nodes
    for $i$ from $1$ to $n$ do
        $\mathbf{C}_i^t = [\mathbf{h}_{i-1}^{t-1}; \mathbf{h}_i^{t-1}; \mathbf{h}_{i+1}^{t-1}; \mathbf{e}^i; \mathbf{s}^{t-1}]$
        $\mathbf{h}_i^t = \mathrm{MultiAtt}(\mathbf{h}_i^{t-1}, \mathbf{C}_i^t)$
        $\mathbf{h}_i^t = \mathrm{LayerNorm}(\mathrm{ReLU}(\mathbf{h}_i^t))$
    // update the relay node
    $\mathbf{s}^t = \mathrm{MultiAtt}(\mathbf{s}^{t-1}, [\mathbf{s}^{t-1}; \mathbf{H}^t])$
    $\mathbf{s}^t = \mathrm{LayerNorm}(\mathrm{ReLU}(\mathbf{s}^t))$
```

4 Comparison to the standard Transformer

Since our goal is making the Transformer lightweight and easy to train with modestly sized dataset, we have removed many connections compared with the standard Transformer (see Fig-1). If the sequence length is n and the dimension of hidden states is d, the computation complexity of one layer in the standard Transformer is $O(n^2 d)$. The Star-Transformer has two phases, the update of ring connections costs $O(5nd)$ (the constant 5 comes from the size of context information $\mathbf{C}$), and the update of radical connections costs $O(nd)$, so the total cost of one layer in the Star-Transformer is $O(6nd)$.

In theory, Star-Transformer can cover all the possible relationships in the standard Transformer. For example, any relationship $\mathbf{h}_i \rightarrow \mathbf{h}_j$ in the standard Transformer can be simulated by $\mathbf{h}_i \rightarrow \mathbf{s} \rightarrow \mathbf{h}_j$. The experiment on the simulation task in Sec-5.1 provides some evidence to show the virtual node s could handle long-range dependencies. Following this aspect, we can give a rough analysis of the path length of dependencies in these models. As discussed in the Transformer paper (Vaswani et al., 2017), the maximum dependency path length of RNN and Transformer are $O(n)$, $O(1)$, respectively. Star-Transformer can pass the message from one node to another node via the relay node so that the maximum dependency path length is also $O(1)$, with a constant two comparing to Transformer.

Compare with the standard Transformer, all positions are processed in parallel, pair-wise connec-

Dataset		Train	Dev.	Test	$\lvert V \rvert$	H DIM	#head	head DIM
Masked Summation		10k	10k	10k	-	100	10	10
SST (Socher et al., 2013)		8k	1k	2k	20k	300	6	50
MTL-16 † (Liu et al., 2017)	Apparel Baby Books Camera DVD Electronics Health IMDB Kitchen Magazines MR Music Software Sports Toys Video	1400	200	400	8k∼28k	300	6	50
SNLI (Bowman et al., 2015)		550k	10k	10k	34k	600	6	100
PTB POS (Marcus et al., 1993)		38k	5k	5k	44k	300	6	50
CoNLL03 (Sang and Meulder, 2003)		15k	3k	3k	25k	300	6	50
OntoNotes NER (Pradhan et al., 2012)		94k	14k	8k	63k	300	6	50

Table 1: An overall of datasets and its hyper-parameters, "H DIM, #head, head DIM" indicates the dimension of hidden states, the number of heads in the Multi-head attention, the dimension of each head, respectively. MTL-16† consists of 16 datasets, each of them has 1400/200/400 samples in train/dev/test.

tions are replaced with a "gather and dispatch" mechanism. As a result, we accelerate the Transformer 10 times on the simulation task and 4.5 times on real tasks. The model also preserves the ability to handle long input sequences. Besides the acceleration, the Star-Transformer achieves significant improvement on some modestly sized datasets.

5 Experiments

We evaluate Star-Transformer on one simulation task to probe its behavior when challenged with long-range dependency problem, and three real tasks (Text Classification, Natural Language Inference, and Sequence Labelling). All experiments are ran on a NVIDIA Titan X card. Datasets used in this paper are listed in the Tab-1. We use the Adam (Kingma and Ba, 2014) as our optimizer. On the real task, we set the embedding size to 300 and initialized with GloVe (Pennington et al., 2014). And the symbol "Ours + Char" means an additional character-level pre-trained embedding JMT (Hashimoto et al., 2017) is used. Therefore, the total size of embedding should be 400 which as a result of the concatenation of GloVe and JMT. We also fix the embedding layer of the Star-Transformer in all experiments.

Since semi- or unsupervised model is also a feasible solution to improve the model in a parallel direction, such as the ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), we exclude these models in the comparison and focus on the relevant architectures.

5.1 Masked Summation

In this section, we introduce a simulation task on the synthetic data to probe the efficiency and non-local/long-range dependencies of LSTM, Transformer, and the Star-Transformer. As mentioned in (Vaswani et al., 2017), the maximum path length of long-range dependencies of LSTM and Transformer are $O(n)$ and $O(1)$, where n is the sequence length. The maximum dependency path length of Star-Transformer is $O(1)$ with a constant two via the relay node. To validate the ability to deal with long-range dependencies, we design a simulation task named "Masked Summation". The input of this task is a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, it has n columns and each column has d elements. The first dimension indicates the mask value $\mathbf{X}_{i0} \in \{0, 1\}$, 0 means the column is ignored in summation. The rest $d-1$ elements are real numbers in drawn uniformly from the range $[0, 1)$. The target is a $d-1$ dimensional vector which equals the summation of all the columns with the mask value 1. There is an implicit variable k to control the number of 1 in the input. Note that a simple baseline is always guessing the value $k/2$.

The evaluation metric is the Mean Square Error (MSE), and the generated dataset has (10k/10k/10k) samples in (train/dev/test) sets. The Fig-2 show a case of the masked summation task.

The mask summation task asks the model to recognize the mask value and gather columns in different positions. When the sequence length n is significantly higher than the number of the columns k, the model will face the long-range dependencies problem. The Fig-3a shows the per-

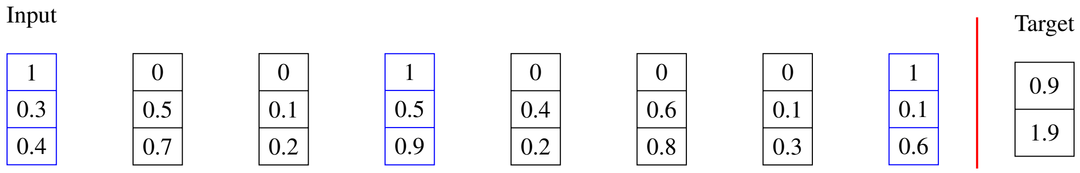
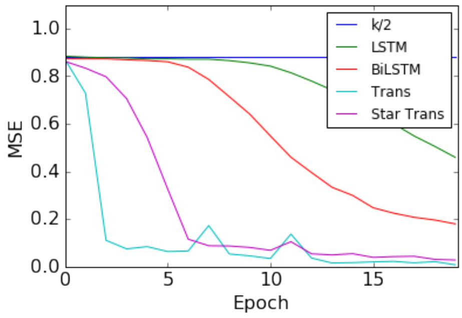
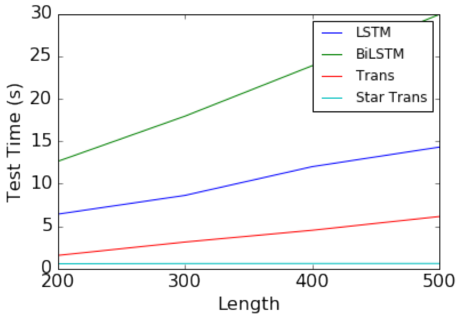


Figure 2: An example of the masked summation task, the input is a sequence of n vectors, each vector has d dimension, and there are total k vectors which have the mask value equals 1. The target is the summation of such masked vectors. In this figure, $n = 8, k = 3, d = 3$.



(a) MSE loss on the masked summation when $n = 200, k = 10, d = 10$. The $k/2$ line means the MSE loss when the model always guess the exception value $k/2$.



(b) Test Time, $k = 10, d = 10$

formance curves of models on various lengths. Although the task is easy, the performance of LSTM and BiLSTM dropped quickly when the sequence length increased. However, both Transformer and Star-Transformer performed consistently on various lengths. The result indicates the Star-Transformer preserves the ability to deal with the non-local/long-range dependencies.

Besides the performance comparison, we also study the speed with this simulation task since we could ignore the affection of padding, masking, and data processing. We also report the inference time in the Fig-3b, which shows that Transformer is faster than LSTM and BiLSTM a lot, and Star-Transformer is faster than Transformer, especially on the long sequence.

Model	Acc
BiLSTM (Li et al., 2015)	49.8
Tree-LSTM (Tai et al., 2015)	51.0
CNN-Tensor (Lei et al., 2015)	51.2
Emb + self-att (Shen et al., 2018a)	48.9
BiLSTM + self-att (Yoon et al., 2018)	50.4
CNN + self-att (Yoon et al., 2018)	50.6
Dynamic self-att (Yoon et al., 2018)	50.6
DiSAN (Shen et al., 2018a)	51.7
Transformer	50.4
Ours	**52.9**

Table 2: Test Accuracy on SST dataset.

5.2 Text Classification

Text classification is a basic NLP task, and we select two datasets to observe the performance of our model in different conditions, Stanford Sentiment Treebank(SST) dataset (Socher et al., 2013) and MTL-16 (Liu et al., 2017) consists of 16 small datasets on various domains. We truncate the sequence which its length higher than 256 to ensure the standard Transformer can run on a single GPU card.

For classification tasks, we use the state of the relay node $\mathbf{s}^T$ plus the feature of max pooling on satellite nodes $\max(\mathbf{H}^T)$ as the final representation and feed it into the softmax classifier. The description of hyper-parameters is listed in Tab-1 and Appendix.

Results on SST and MTL-16 datasets are listed in Tab-2,3, respectively. On the SST, the Star-Transformer achieves 2.5 points improvement against the standard Transformer and beat the most models.

Also, on the MTL-16, the Star-Transformer outperform the standard Transformer in all 16 datasets, the improvement of the average accuracy is 4.2. The Star-Transformer also gets better results compared with existing works. As we mentioned in the introduction, the standard Transformer requires large training set to reveal its power. Our experiments show the Star-

Dataset	Acc (%)				Test Time (ms)			Len.
	Star-Transformer	Transformer	BiLSTM	SLSTM	Star-Transformer	Transformer	BiLSTM	
Apparel	88.25	82.25	86.05	85.75	11	34	114	65
Baby	87.75	84.50	84.51	86.25	11	50	141	109
Books	87.00	81.50	82.12	83.44	11	55	135	131
Camera	92.25	86.00	87.05	90.02	11	52	125	116
DVD	86.00	77.75	83.71	85.52	11	58	143	139
Electronics	82.75	81.50	82.51	83.25	10	49	136	105
Health	87.50	83.50	85.52	86.50	10	45	131	85
IMDB	88.00	82.50	86.02	87.15	12	77	174	201
Kitchen	85.50	83.00	82.22	84.54	11	47	130	92
Magazines	92.75	89.50	92.52	93.75	11	51	136	115
MR	79.00	77.25	75.73	76.20	12	14	26	22
Music	83.50	79.00	78.74	82.04	11	53	137	118
Software	90.50	85.25	86.73	87.75	11	53	140	117
Sports	86.25	84.75	84.04	85.75	10	49	134	103
Toys	88.00	82.00	85.72	85.25	11	46	137	96
Video	86.75	84.25	84.73	86.75	11	56	146	131
Average	**86.98**	82.78	84.01	85.38	**10.94**	49.31	130.3	109.1

Table 3: Test Accuracy over MTL-16 datasets. “Test Time” means millisecond per batch on the test set (batch_size is 128). “Len.” means the average sequence length on the test set.

Transformer could work well on the small dataset which only has 1400 training samples. Results of the time-consuming show the Star-Transformer could be 4.5 times fast than the standard Transformer on average.

5.3 Natural Language Inference

Natural Language Inference (NLI) asks the model to identify the semantic relationship between a premise sentence and a corresponding hypothesis sentence. In this paper, we use the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) for evaluation. Since we want to study how the model encodes the sentence as a vector representation, we set Star-Transformer as a sentence vector-based model and compared it with sentence vector-based models.

In this experiment, we follow the previous work (Bowman et al., 2016) to use concat($\mathbf{r}_1, \mathbf{r}_2, \|\mathbf{r}_1 - \mathbf{r}_2\|, \mathbf{r}_1 - \mathbf{r}_2$) as the classification feature. The $\mathbf{r}_1, \mathbf{r}_2$ are representations of premise and hypothesis sentence, it is calculated by $\mathbf{s}^T + \max(\mathbf{H}^T)$ which is same with the classification task. See Apendix for the detail of hyper-parameters.

As shown in Tab-4, the Star-Transformer outperforms most typical baselines (DiSAN, SPINN) and achieves comparable results compared with the state-of-the-art model. Notably, our model beats standard Transformer by a large margin, which is easy to overfit although we have made a careful hyper-parameters' searching for Transformer.

Model	Acc
BiLSTM (Liu et al., 2016)	83.3
BiLSTM + self-att (Liu et al., 2016)	84.2
300D SPINN-PI (Bowman et al., 2016)	83.2
Tree-based CNN (Mou et al., 2016)	82.1
4096D BiLSTM-max (Conneau et al., 2017)	84.5
300D DiSAN (Shen et al., 2018a)	85.6
Residual encoders (Nie and Bansal, 2017)	86.0
Gumbel TreeLSTM (Choi et al., 2018)	86.0
Reinforced self-att (Shen et al., 2018b)	86.3
2400D Multiple DSA (Yoon et al., 2018)	87.4
Transformer	82.2
Star-Transformer	**86.0**

Table 4: Test Accuracy on SNLI dataset.

The SNLI dataset is not a small dataset in NLP area, so improving the generalization ability of the Transformer is a significant topic.

The best result in Tab-4 (Yoon et al., 2018) using a large network and fine-tuned hyper-parameters, they get the best result on SNLI but an undistinguished result on SST, see Tab-2.

5.4 Sequence Labelling

To verify the ability of our model in sequence labeling, we choose two classic sequence labeling tasks: Part-of-Speech (POS) tagging and Named Entity Recognition (NER) task.

Three datasets are used as our benchmark: one POS tagging dataset from Penn Treebank (PTB) (Marcus et al., 1993), and two NER datasets from CoNLL2003 (Sang and Meulder, 2003), CoNLL2012 (Pradhan et al., 2012). We use the fi-

Model	Adv Tech		POS	NER	
			PTB	CoNLL2003	CoNLL2012
	char	CRF	Acc	F1	F1
(Ling et al., 2015)	✓	✓	97.78	-	-
(Collobert et al., 2011)	✓	✓	97.29	89.59	-
(Huang et al., 2015)	✓	✓	97.55	90.10	-
(Chiu and Nichols, 2016a)	✓	✓	-	90.69	86.35
(Ma and Hovy, 2016)	✓	✓	97.55	91.06	-
(Nguyen et al., 2016)	✓	✓	-	91.2	-
(Chiu and Nichols, 2016b)	✓	✓	-	91.62	86.28
(Zhang et al., 2018)	✓	✓	97.55	91.57	-
(Akhundov et al., 2018)	✓	✓	97.43	91.11	87.84
Transformer			96.31	86.48	83.57
Transformer + Char	✓		97.04	88.26	85.14
Star-Transformer			97.14	90.93	86.30
Star-Transformer + Char	✓		97.64	91.89	87.64
Star-Transformer + Char + CRF	✓	✓	**97.68**	**91.98**	**87.88**

Table 5: Results on sequence labeling tasks. We list the "Advanced Techniques" except widely-used pre-trained embeddings (GloVe, Word2Vec, JMT) in columns. The "Char" indicates character-level features, it also includes the Capitalization Features, Suffix Feature, Lexicon Features, etc. The "CRF" means an additional Conditional Random Field (CRF) layer.

Model	SNLI Acc	CoNLL03 Acc	MS MSE
Star-Transformer	86.0	90.93	0.0284
variant (a) -radical	84.0	89.35	0.1536
variant (b) -ring	77.6	79.36	0.0359

Table 6: Test Accuracy on SNLI dataset, CoNLL2003 NER task and the Masked Summation $n = 200, k = 10, d = 10$.

nal state of satellite nodes $\mathbf{H}^T$ to classify the label in each position. Since we believe that the complex neural network could be an alternative of the CRF, we also report the result without CRF layer.

As shown in Tab-5, Star-Transformer achieves the state-of-the-art performance on sequence labeling tasks. The "Star-Transformer + Char" has already beat most of the competitors. Star-Transformer could achieve such results without CRF, suggesting that the model has enough capability to capture the partial ability of the CRF. The Star-Transformer also outperforms the standard Transformer on sequence labeling tasks with a significant gap.

5.5 Ablation Study

In this section, we perform an ablation study to test the effectiveness of the radical and ring connections.

We test two variants of our models, the first variants (a) remove the radical connections and only keep the ring connections. Without the radical connections, the maximum path length of this variant becomes $O(n)$. The second variant (b) removes the ring connections and remains the radical connections. Results in Tab-6 give some insights, the variant (a) loses the ability to handle long-range dependencies, so it performs worse on both the simulation and real tasks. However, the performance drops on SNLI and CoNLL03 is moderate since the remained ring connections still capture the local features. The variant (b) still works on the simulation task since the maximum path length stays unchanged. Without the ring connections, it loses its performance heavily on real tasks. Therefore, both the radical and ring connections are necessary to our model.

6 Conclusion and Future Works

In this paper, we present Star-Transformer which reduce the computation complexity of the standard Transformer by carefully sparsifying the topology. We compare the standard Transformer with other models on one toy dataset and 21 real datasets and find Star-Transformer outperforms the standard Transformer and achieves comparable results with state-of-the-art models.

This work verifies the ability of Star-Transformer by excluding the factor of unsupervised pre-training. In the future work, we will investigate the ability of Star-Transformer by unsupervised pre-training on the large corpus. Moreover, we also want to introduce more NLP prior knowledge into the model.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), National Key Research and Development Program of China (No. 2017YFB1002104), and National Natural Science Foundation of China (No. 61672162 and 61751201).

References

Adnan Akhundov, Dietrich Trautmann, and Georg Groh. 2018. Sequence labeling: A practical approach. *CoRR*, abs/1808.03926.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642. The Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *ACL (1)*. The Association for Computer Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on EMNLP*, pages 551–561.

Jason Chiu and Eric Nichols. 2016a. Sequential labeling with bidirectional lstm-cnns. In *Proc. International Conf. of Japanese Association for NLP*, pages 937–940.

Jason P. C. Chiu and Eric Nichols. 2016b. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*, pages 5094–5101. AAAI Press.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. *CoRR*, abs/1807.03819.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*, pages 1923–1933. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*, pages 1565–1575. The Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In *EMNLP*, pages 2304–2314. The Association for Computational Linguistics.

Zhouhan Lin, Mo Feng, Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, pages 1520–1530. The Association for Computational Linguistics.

Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2018a. Contextualized non-local neural networks for sequence learning. *arXiv preprint arXiv:1811.08600*.

Pengfei Liu, Jie Fu, Yue Dong, Xipeng Qiu, and Jackie Chi Kit Cheung. 2018b. Multi-task learning over graph structures. *arXiv preprint arXiv:1811.10211*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL (1)*, pages 1–10. Association for Computational Linguistics.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL (1)*. The Association for Computer Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *ACL (2)*. The Association for Computer Linguistics.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: joint named entity recognition and disambiguation with rich linguistic features. *TACL*, 4:215–229.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *RepEval@EMNLP*, pages 41–45. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*, pages 1–40. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147. ACL.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455. AAAI Press.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018b. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *IJCAI*, pages 4345–4352. ijcai.org.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL (1)*, pages 1556–1566. The Association for Computer Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of NAACL*, pages 1480–1489.

Deunsol Yoon, Dongbok Lee, and SangKeun Lee. 2018. Dynamic self-attention : Computing attention over words dynamically for sentence embedding. *CoRR*, abs/1808.07383.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *ACL (1)*, pages 317–327. Association for Computational Linguistics.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612.

Adaptation of Hierarchical Structured Models for Speech Act Recognition in Asynchronous Conversation

Tasnim Mohiuddin[1*] and **Thanh-Tung Nguyen**[1⋆] and **Shafiq Joty**[1,2⋆]
[⋆]Nanyang Technological University, Singapore
[†]Salesforce Research Asia, Singapore
{mohi0004@e.,NG0155NG@e.,srjoty@}ntu.edu.sg

Abstract

We address the problem of speech act recognition (SAR) in asynchronous conversations (forums, emails). Unlike synchronous conversations (e.g., meetings, phone), asynchronous domains lack large labeled datasets to train an effective SAR model. In this paper, we propose methods to effectively leverage abundant unlabeled conversational data and the available labeled data from synchronous domains. We carry out our research in three main steps. First, we introduce a neural architecture based on hierarchical LSTMs and conditional random fields (CRF) for SAR, and show that our method outperforms existing methods when trained on in-domain data only. Second, we improve our initial SAR models by semi-supervised learning in the form of pretrained word embeddings learned from a large unlabeled conversational corpus. Finally, we employ adversarial training to improve the results further by leveraging the labeled data from synchronous domains and by explicitly modeling the distributional shift in two domains.

1 Introduction

With the ever-increasing popularity of Internet and mobile technologies, communication media like emails and forums have become an integral part of people's daily life where they discuss events, issues and experiences. Participants interact with each other *asynchronously* in these media by writing at different times, generating a type of conversational discourse that is different from synchronous conversations such as meeting and phone conversations (Louis and Cohen, 2015). In the course of the interactions, the participants perform certain communicative acts like asking questions, requesting information, or suggesting something, which are known as **speech acts** (Austin, 1962). For example, consider the forum conversation in Figure 1. The participant who posted the initial comment **C$_1$**, describes his situation and asks a couple of *questions*. Other participants *respond* to the initial post with more information and provide *suggestions*. In this process, the participants get into a conversation by taking turns, each of which consists of one or more speech acts.

C$_1$: hoping to do the XinJiang Tibet Highway. [Statement]
Has anyone done it? [Question]
Am hoping to hire a 4-wheel drive. [Statement]
I know the roads are bad and I would need an experienced driver and guide - any recommendations? [Question]

C$_2$: I never done this routine,however i been to Xinjiang twice,in my opinion the local people not friendly, not safe to do this. [Response]
I still have relative stay in Xinjiang, however don't know what they can offer for help... [Response]

C$_3$: I'm not sure if travelling overland from Xinjiang to Tibet is officially legal yet. [Response]
You might want to post your question on the North-East Asia branch of Lonely Planet's ThornTree forum for more (useful) answers. [Suggestion]

C$_4$: a frend and i are trying this route as well, we will likely be in urumuqi and northern part of xinjiang from 8th apr to end apr; looking at doing the xin jiang tibet highway from end apr. (truncated) [Statement]
contact me at [email] if you want to hook up for possible transport sharing [Suggestion]
cheers. [Polite]

Figure 1: Example of speech acts in a forum thread.

Speech act recognition (SAR) is an important step towards deep conversational analysis, and can benefit many downstream applications. Availability of large labeled datasets such as the Switchboard-DAMSL (SWBD) (Jurafsky et al., 1997) and the Meeting Recorder Dialog Act (MRDA) (Dhillon et al., 2004) corpora has fostered research in data-driven SAR methods in synchronous domains. However, such large corpora

---

[*] All authors contibuted equally.

Proceedings of NAACL-HLT 2019, pages 1326–1336
Minneapolis, Minnesota, June 2 - June 7, 2019.

are not available in the asynchronous domains, and many of the existing (small-sized) ones use task-specific tagsets as opposed to a standard one. The unavailability of large annotated datasets with standard tagsets is one of the main reasons for SAR not getting much attention in asynchronous domains, and it is often quite expensive to annotate such datasets for each domain of interest.

SAR methods proposed before the neural 'tsunami', *e.g.,* (Qadir and Riloff, 2011; Jeong et al., 2009; Tavafi et al., 2013), used mostly bag-of-ngram representation (*e.g.,* unigram, bigram) of a sentence, and most of these methods disregard conversational dependencies (discourse structure) between sentences. Recently, Joty and Hoque (2016) proposed a neural-CRF framework for SAR in forum conversations. In their approach, a bi-LSTM (trained on the SAR task) first encodes the sentences separately into task-specific embeddings, which are then used in a separate CRF model to capture the conversational dependencies between sentences. They also use labeled data from the MRDA meeting corpus, without which their LSTMs perform worse than simple feed-forward networks. Although their method attempts to model sentence structure (using LSTM) and conversational dependencies (using CRF), the approach has several limitations.

First, the LSTM-CRF framework was disjoint, and thus cannot be trained end-to-end. Second, when using the MRDA meeting data, their method simply concatenates it with the target domain data assuming they have the same distribution. However, asynchronous domains (forum, email) differ from synchronous (MRDA) in their underlying conversational structure (Louis and Cohen, 2015), in style (spoken vs. written), and in vocabulary usage (meetings on some focused agenda vs. conversations on any topic of interests in a public forum). Therefore, we hypothesize that to make the best use of labeled data from synchronous domains, one needs to model the shift in domains.

In this work, we advance the state-of-the-art of SAR in asynchronous conversations in three main steps. First, we introduce an end-to-end neural architecture based on a hierarchical LSTM encoder with a Softmax or CRF output layer. Second, we improve our initial SAR model by semi-supervised learning in the form of word embeddings learned from a large unlabeled conversational corpus. Most importantly, we adapt our hierarchical LSTM encoder using domain adversarial training (Ganin et al., 2016) to leverage the labeled data from synchronous domains by explicitly modeling the shift in the two domains.

We evaluate our models on three different asynchronous datasets containing forum and email conversations, and on the MRDA meeting corpus. Our main findings are: (*i*) the hierarchical LSTMs outperform existing methods when trained on in-domain data for both synchronous and asynchronous domains, setting a new state-of-the-art; (*ii*) conversational word embeddings yield significant improvements over off-the-shelf ones; and (*iii*) domain adversarial training improves the results by inducing domain-invariant features. The source code, the conversational word embeddings, and the datasets are available at `https://ntunlpsg.github.io/demo/project/speech-act/`.

2 Related Work

Previous studies on **SAR in asynchronous conversation** have used supervised, semi-supervised and unsupervised methods. Cohen et al. (2004) classify emails into acts like 'deliver' and 'meeting'. Their approach however does not take email context into account. Carvalho and Cohen (2005) use an iterative algorithm containing two different classifiers: the content classifier that only looks at the content of the message, and the context classifier that takes into account both the content and contextual speech acts in the email thread structure. Other supervised approaches use classifiers and sequence taggers with hand-crafted features (Qadir and Riloff, 2011; Tavafi et al., 2013).

Jeong et al. (2009) use semi-supervised boosting to induce informative patterns from labeled spoken domains (MRDA, SWBD). Given a sentence represented as a set of trees (dependency, POS tags, n-grams), the boosting algorithm iteratively learns the sub-tree features. This approach does not consider the dependencies between the act types, something we successfully exploit in our work. Also, we leverage labeled data from synchronous conversations while adapting our model to account for the domain shift. Joty and Hoque (2016) use a bi-LSTM to encode a sentence, then use a separate CRF to model conversational dependencies. To learn an effective bi-LSTM model, they use the MRDA meeting data; however, without modeling the domain differences.

The unsupervised methods use variations of Hidden Markov Models (HMM) including HMM-Topic (Ritter et al., 2010), HMM-Mix (Joty et al., 2011), and Mixed Membership (Paul, 2012).

Several neural methods have been proposed in recent years for **SAR in synchronous conversations**. Kalchbrenner and Blunsom (2013) use a simple recurrent neural network (RNN) to model sequential dependencies between act types in phone conversations. They use a convolutional network to compose sentence representations from word vectors. Lee and Dernoncourt (2016) use a similar model, but also experiment with RNNs to compose sentence representations. Khanpour et al. (2016) use a stacked LSTM to compose word vectors into a sentence vector. Kumar et al. (2018) also use a hierarchical LSTM-CRF. However, none of these methods were applied to asynchronous conversations, where not much labeled data is available. Also to the best of our knowledge, no prior work attempted to do domain adaptation from the synchronous conversation, which is our main contribution in this paper.

3 The Base Model

We use a bidirectional long short-term memory or bi-LSTM (Hochreiter and Schmidhuber, 1997) to encode each sentence into a vector representation. Given an input sentence $\mathbf{x}_i = (w_1, \cdots, w_m)$ of length m, we first map each word w_t to its corresponding vector representation $\mathbf{v}_t$ by looking up the word embedding matrix. The LSTM recurrent layer then computes a compositional representation $\mathbf{z}_t$ at every time step t by performing nonlinear transformations of the current input $\mathbf{v}_t$ and the output of the previous time step $\mathbf{z}_{t-1}$. The output of the last time step $\mathbf{z}_m$ is considered as the representation of the sentence. A bi-LSTM composes a sentence in two directions: left-to-right and right-to-left, yielding a representation $\mathbf{h}_i = [\overrightarrow{\mathbf{z}_m}; \overleftarrow{\mathbf{z}_m}]$, where ';' denotes concatenation. Similar to (Joty and Hoque, 2016), we could use $\mathbf{h}_i$ to classify sentence $\mathbf{x}_i$ into one of the speech act types using a Softmax output layer. However, in that case, we would disregard the discourse-level dependencies between sentences in a conversation. To take conversational dependencies into account, we explore two methods as we describe below.

3.1 Hierarchical LSTM

We consider a conversation as a sequence of utterances (sentences). Given an input sequence of n sentences $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$, the sentence-level bi-LSTM generates a sequence of n vectors $\mathbf{H} = (\mathbf{h}_1, \cdots, \mathbf{h}_n)$. To consider interdependencies between sentences, we place another bi-LSTM layer on top of $\mathbf{H}$ to connect the sentence vectors sequentially in both directions, and encode each sentence within its left and right contexts. As shown in Figure 2, the upper bi-LSTM combines the current input $\mathbf{h}_i$ with its previous hidden state $\overrightarrow{\mathbf{u}}_{i-1}$ (resp., $\overleftarrow{\mathbf{u}}_{i+1}$) to generate a representation for the current sentence $\overrightarrow{\mathbf{u}}_i$ (resp., $\overleftarrow{\mathbf{u}}_i$). The hierarchically encoded sentence vectors $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_n)$ (where $\mathbf{u}_i = [\overrightarrow{\mathbf{u}_i}; \overleftarrow{\mathbf{u}_i}]$) are fed into a Softmax classifier for speech act classification.

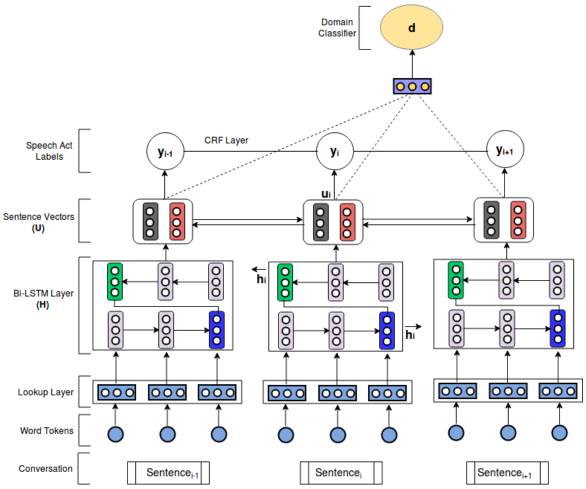


Figure 2: Hierarchical bi-LSTM-CRF model with domain adversarial training for speech act recognition.

$$p(y_i = k|\mathbf{X}, W, \theta) = \frac{\exp\ (\mathbf{w}_k^T \mathbf{u}_i)}{\sum_{k=1}^{K} \exp\ (\mathbf{w}_k^T \mathbf{u}_i)}$$

where W are the classifier weights, and θ are the parameters of the hierarchical LSTM encoder. We train the model by minimizing the cross entropy:

$$\mathcal{L}_c(W, \theta) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k} \log p(y_i = k|\mathbf{X}, W, \theta)$$

with $y_{i,k}$ being the one-hot encoding of the label.

3.2 Hierarchical LSTM with CRF

The hierarchical LSTM (H-LSTM) captures contextual information by propagating information through hidden layers, and has been shown to be effective in similar tasks such as context encoding in dialog systems (Serban et al., 2016). Despite this, its modeling strength is limited compared to structured models that use global inference to model consistency in the output, especially when there are strong dependencies between output labels (Collobert et al., 2011). Therefore, instead of classifying sentences independently with a Softmax layer, our second method is to model them jointly with a CRF layer (Lafferty et al., 2001). For an input-output sequence pair $(\mathbf{X}, \mathbf{y})$, we define the joint probability distribution:

$$p(\mathbf{y}|\mathbf{X}) = \frac{1}{Z(\mathbf{U}, A, V, \theta)} \prod_{i=1}^{n} \underbrace{\psi_n(y_i|\mathbf{u}_i, V)}_{\text{node factor}} \prod_{i=0}^{n} \underbrace{\psi_e(y_{i,i+1}|A)}_{\text{edge factor}}$$

where $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_n)$ is the hierarchically encoded sentence vectors as before, and $\psi_n(y_i = k|\mathbf{u}_i, V) = \exp(V_k^T \mathbf{u}_i)$ is the node-level score with V being the weight matrix, ψ_e is the transition matrix parameterized by A, and $Z(.)$ is the global normalization constant that ensures a valid probability distribution. The cross entropy loss for the $(\mathbf{X}, \mathbf{y})$ sequence pair can be written as:

$$\mathcal{L}_c(V, A, \theta) = -\sum_{i=1}^{n} \log \psi_n(y_i|\mathbf{u}_i, V) - \sum_{i=0}^{n} \log A_{i,i+1} + \log Z$$

We use Viterbi decoding to infer the most probable tag sequence for an input sequence of sentences, $\mathbf{y}* = \text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{X}, V, A, \theta)$. We will demonstrate later in our experiments that a CRF layer helps the H-LSTM to adapt quickly (*i.e.,* with less labeled data) to a target domain by exploiting the tag dependencies in the source domain.

4 Adaptation Methods

The hierarchical models have many parameters. Given enough training data, they should be able to encode a sentence, capturing its syntactic and semantic properties, and discourse-level dependencies. However, when it comes to SAR in asynchronous domains, not many large annotated corpora are available. Because of the large number of parameters, the models usually overfit when trained on small datasets of asynchronous conversations (shown in Sec. 6). We propose two solutions to address this problem. Our first (simple but effective) solution is to leverage large unlabeled conversational corpus to learn better task-agnostic word embeddings, and use it to initialize our models for better generalization. In the interests of coherence, we present this method in Section 5.

Our second solution is to leverage data from synchronous domains for which large annotated corpus is available (*e.g.,* MRDA corpus). However, as we will see, simple concatenation of the datasets is not quite effective in our case, because the conversations in synchronous and asynchronous domains differ in their conversational structures, modality (spoken vs. written), and vocabulary usage. To get the best out of the available synchronous domain data, we need to adapt our models by explicitly modeling the domain shift. More precisely, our goal is to adapt the hierarchical encoder so that it learns to encode sentence representations $\mathbf{U}$ (*i.e.,* features used for classification) that is not only discriminative for the act classification, but also invariant across the domains. We propose to use the **domain adversarial training** proposed by Ganin et al. (2016).

Let $\mathcal{D}_S = \{\mathbf{X}_p, \mathbf{y}_p\}_{p=1}^{P}$ denote the set of P labeled training conversations in the **source** domain (MRDA). We consider two adaptation scenarios.

***(i)* Unsupervised adaptation:** In this scenario, we have only *unlabeled* examples in the **target** domain (*e.g.,* forum). Let $\mathcal{D}_T^u = \{\mathbf{X}_p\}_{p=P+1}^{Q}$ be the set of $(Q - P - 1)$ unlabeled training instances in the target domain with Q being the total number of training instances in the two domains.

***(ii)* Semi-supervised/supervised adaptation:** In addition to the unlabeled instances $\mathcal{D}_T^u$, here we have access to some *labeled* training instances in the target domain, $\mathcal{D}_T^l = \{\mathbf{X}_p, \mathbf{y}_p\}_{p=Q+1}^{R}$, with R being the total number of training examples in the two domains. Depending on the amount of labeled data in the target domain, this setting is referred to as semi-supervised or supervised adaptation.

4.1 Unsupervised Adaptation

The dashed lines in Figure 2 show the extension of our base model for adaptation. The input conversation $\mathbf{X}$ is sampled either from a synchronous domain (*e.g.,* meeting) or from an asynchronous domain (*e.g.,* forum). Our goal is to adapt the H-LSTM encoder (parameterized by θ) to generate $\mathbf{U}$ such that it is not only informative for the SAR task but also invariant across domains. Upon achieving this, we can use the adapted encoder to encode a target sentence, and use the source classifier (Softmax or CRF) to classify the sentences.

We achieve this by adding a domain discriminator (dashed lines in Figure 2), another neural network that takes $\mathbf{U}$ as input, and tries to discriminate the domains of the input conversation $\mathbf{X}$ (*e.g.,* meeting vs. forum). The output of the discriminator is defined by a sigmoid function:

$$\hat{d}_\omega = p(d = 1|\mathbf{u}_i, \omega, \theta) = \text{sigm}(\mathbf{w}_d^T \mathbf{h}_d) \quad (1)$$

where $d \in \{0, 1\}$ denotes the domain (1 for meeting, 0 for forum), $\mathbf{w}_d$ are the final layer weights of the discriminator, and $\mathbf{h}_d = g(U_d \mathbf{u}_i)$ defines the hidden layer of the discriminator with U_d being the layer weights, and $g(.)$ being the activations. We use cross entropy as the discrimination loss:

$$\mathcal{L}_d(\omega, \theta) = -d \log \hat{d}_\omega - (1 - d) \log \left(1 - \hat{d}_\omega\right) \quad (2)$$

The composite network has three players: the hierarchical **LSTM encoder**, the **classifier** (Softmax or CRF), and the domain **discriminator**. During training, the encoder and the classifier play a co-operative game, while the encoder and the discriminator play an adversarial game. The training objective $\mathcal{L}(W, \theta, \omega)$ of the composite model is:

$$\underbrace{\sum_{p=1}^{P} \mathcal{L}_c^p(W, \theta)}_{\text{act classif (src)}} - \lambda \Big[\underbrace{\sum_{p=1}^{P} \mathcal{L}_d^p(\omega, \theta)}_{\text{domain disc (src)}} + \underbrace{\sum_{p=P+1}^{Q} \mathcal{L}_d^p(\omega, \theta)}_{\text{domain disc (tar)}} \Big] \quad (3)$$

where θ are the parameters of the encoder, W are the classifier weights, and $\omega = \{U_d, \mathbf{w}_d\}$ are the parameters of the discriminator.[1] The hyper-parameter λ controls the relative strength of the act classifier and the discriminator. We learn θ that optimizes the following **min-max** criterion:

$$\theta^* = \underset{W,\theta}{\text{argmin}} \max_{U_d, \mathbf{w}_d} \mathcal{L}(W, \theta, \omega) \quad (4)$$

Note that the updates of the shared encoder for the two networks (classifier and discriminator) work *adversarially* with respect to each other. Algorithm 1 provides pseudocode of our training method. The main challenge in adversarial training is to balance the networks (Arjovsky et al., 2017). In our experiments, we found the discriminator to be weaker initially. To balance the two components, we would need the error signals from the discriminator to be fairly weak initially, with full power unleashed only as the classification errors start to dominate. We follow the weighting schedule proposed in (Ganin et al., 2016, p. 21), which initializes λ to 0, and then changes it gradually to 1 as training progresses.

Algorithm 1: Adversarial training with SGD.

```
Input  : Data $\mathcal{D}_S$, $\mathcal{D}_T^u$, and batch size $b$
Output: Adapted model parameters $\theta$, W
1. Initialize model parameters;
2. repeat
    (a) Randomly sample $\frac{b}{2}$ labeled examples from $\mathcal{D}_S$
    (b) Randomly sample $\frac{b}{2}$ unlabeled examples from
        $\mathcal{D}_T^u$
    (c) Compute $\mathcal{L}_c(W, \theta)$ and $\mathcal{L}_d(\omega, \theta)$
    (d) Set $\lambda = \frac{2}{1+\exp(-10*p)} - 1$; $p$ is the training
        progress linearly changing form 0 to 1.
   // Classifier & Encoder
    (e) Take a gradient step for $\frac{2}{b} \nabla_{W,\theta} \mathcal{L}_c(W, \theta)$
   // Discriminator
    (f) Take a gradient step for $\frac{2\lambda}{b} \nabla_{U_d, \mathbf{w}_d} \mathcal{L}_d(\omega, \theta)$
   // Gradient reversal
    (g) Take a gradient step for $-\frac{2\lambda}{b} \nabla_\theta \mathcal{L}_d(\omega, \theta)$
until convergence;
```

4.2 Semi-supervised/supervised Adaptation

It is straight-forward to extend our adaptation method to a semi-supervised/supervised setting. Similar to the instances in the source domain, the labeled instances in the target domain $\mathcal{D}_T^l$ are used for act classification and domain discrimination. The total training loss in this case is

$$\mathcal{L}(W, \theta, \omega) = \underbrace{\sum_{p=1}^{P} \mathcal{L}_c^p(W, \theta)}_{\text{act classif. (source)}} + \underbrace{\sum_{p=Q+1}^{R} \mathcal{L}_c^p(W, \theta)}_{\textit{act classif. (target)}} -$$
$$\lambda \Big[\underbrace{\sum_{p=1}^{P} \mathcal{L}_d^p(\omega, \theta)}_{\text{dom classif. (source)}} + \underbrace{\sum_{n=P+1}^{R} \mathcal{L}_d^p(\omega, \theta)}_{\text{dom classif. (target)}} \Big]$$

where the second term is the classification loss on the target dataset $\mathcal{D}_T^l$, and the last term is the discrimination loss on both labeled and unlabeled data in the target domain.

5 Corpora

We now describe the datasets and the act tagset that we use, and the conversational word embeddings that we learn from a large unlabeled corpus.

5.1 Labeled Datasets

As mentioned, asynchronous domains lack large corpora that are annotated with a standard speech act tagset. Jeong et al. (2009) annotated sentences in TripAdvisor **(TA) forum** threads with the standard 12 act types defined in MRDA. They also remapped the **BC3 email** corpus (Ulrich et al., 2008) according to these tags. Subsequent studies (Tavafi et al., 2013; Oya and Carenini, 2014; Joty

---

[1] For simplicity, we describe adaptation of the encoder with Softmax output, but this generalizes naturally to CRF.

	Asynchronous			Synchronous
	TA	BC3	QC3	MRDA
Total # of conversations	200	39	47	73
Avg. # of comments/conv	4.02	6.54	13.32	N.A
Avg. # of sentences/conv	18.56	34.15	33.28	955.10
Avg. # of words/sen	14.90	12.61	19.78	10.11

Table 1: Basic statistics about our corpora.

		Asynchronous			**Synchronous**
Tag	Description	TA	BC3	QC3	MRDA
SU	Suggestion	7.71	5.48	17.38	5.97
R	Response	2.4	3.75	5.24	15.63
Q	Questions	14.71	8.41	12.59	8.62
P	Polite	9.57	8.63	6.13	3.77
ST	Statement	65.62	73.72	58.66	66.00

Table 2: Distribution of speech acts in our corpora.

		Train	**Dev.**	**Test**
Source	MRDA	59 (50865)	6 (8366)	8 (10492)
Target	TA	90 (1667)	20 (427)	90 (1617)
	QC3	20 (675)	7 (230)	20 (660)
	BC3	16 (557)	7 (233)	16 (542)
Total		126 (2899)	34 (890)	126 (2819)

Table 3: Train, dev. and test sets for the datasets. Numbers in parentheses indicate the number of sentences.

	# of Threads	# of Tokens	# of Words
W3C	23,940	21,465,830	546,921
TripAdvisor	25,000	2,037,239	127,333
Qatar Living	219,690	103,255,922	1,157,757
MRDA	-	675,110	18,514
SWBD	-	1,131,516	57,075

Table 4: Datasets and their statistics used for training the conversational word embeddings.

and Hoque, 2016) used these datasets but grouped the 12 acts into 5 coarser classes. Joty and Hoque (2016) also created a new dataset of QatarLiving[2] **forum** threads called **QC3**.[3] We use these three asynchronous datasets in our experiments. For our experiments on synchronous domains, we use the **MRDA meeting** corpus that was also used in related studies (Jeong et al., 2009; Joty and Hoque, 2016). Tables 1 and 2 show some basic statistics of the datasets and the tag distributions. Note that the tagset used by us and other related studies in asynchronous (written) conversation is different from the one used in synchronous spoken conversations (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Kumar et al., 2018). The later tagset contains acts like *backchannel*, *filter* and *disruption* that are more specific to speech.

The train-dev-test splits of the asynchronous datasets are done uniformly at random at the **conversation level**. Since the asynchronous datasets are quite small in size, to have a reliable test set, we create the train:test splits with an equal number of conversations (Table 3). Joty and Hoque (2016) also created conversation level datasets to train and test their CRF models. Their test sets however contain only 20% of the conversations, providing only 5 conversations for QC3 and BC3, and 20 for TA. Our experiments on these small test sets showed unstable results for all the models. Therefore, we use a larger test set (50%), and we report more general results on the whole corpus based on **2-fold** cross-validation, where the second fold was created by interchanging the train and test splits in Table 3. The same development set was used to tune the hyperparameters of the models for experiments on each fold. For experiments on MRDA, we use the same train:test:dev split as in (Jeong et al., 2009; Joty and Hoque, 2016).

5.2 Conversational Word Embeddings

One simple and effective approach to semi-supervised learning is to use word embeddings pretrained from a large unlabeled corpus. In our work, we use generic *off-the-shelf* pretrained embeddings to boost the performance of our models. In addition, we have also trained word embeddings from a large conversational corpus to get more relevant **conversational** word embeddings.

We use Glove (Pennington et al., 2014) to train our word embeddings from a corpus that contains 24K email threads from W3C (w3c.org), 25K threads from TripAdvisor, 220K threads from QatarLiving, and all conversations from SWBD and MRDA (a total of 120M tokens). Table 4 shows some statistics of the datasets used for training the conversational word embeddings. We also trained skip-gram word2vec (Mikolov et al., 2013), but its performance was worse than Glove.

6 Experiments

We followed similar preprocessing steps as Joty and Hoque (2016); specifically: normalize all characters to lower case, spell out digits and URLs, and tokenize the texts using TweetNLP (Gimpel et al., 2011). For performance compar-

---

[2] http://www.qatarliving.com/

[3] https://ntunlpsg.github.io/project/speech-act/

	QC3	**TA**	**BC3**	**MRDA**
SVM$_{\text{c-gl}}$	$16.96_{\pm 0.00}$	$20.17_{\pm 0.00}$	$17.20_{\pm 0.00}$	$31.47_{\pm 0.00}$
FFN$_{\text{c-gl}}$	$48.29_{\pm 0.25}$	$61.36_{\pm 0.21}$	$39.58_{\pm 0.26}$	$71.12_{\pm 0.13}$
FFN$_{\text{skip-th}}$	$50.80_{\pm 1.21}$	$61.44_{\pm 0.92}$	$47.67_{\pm 0.74}$	$71.73_{\pm 0.48}$
B-LSTM$_{\text{rand}}$	$50.25_{\pm 0.57}$	$62.11_{\pm 0.64}$	$45.08_{\pm 1.03}$	$70.72_{\pm 0.02}$
B-LSTM$_{\text{gl}}$	$53.21_{\pm 0.77}$	$63.23_{\pm 0.80}$	$49.04_{\pm 0.90}$	$72.23_{\pm 0.18}$
B-GRU$_{\text{c-gl}}$	$60.50_{\pm 0.36}$	$67.23_{\pm 0.76}$	$55.45_{\pm 1.05}$	$72.04_{\pm 0.35}$
B-LSTM$_{\text{c-gl}}$	$\mathbf{61.01_{\pm 0.60}}$	$67.23_{\pm 0.70}$	$55.32_{\pm 0.68}$	$72.42_{\pm 0.14}$
S-LSTM$_{\text{c-gl}}$	$56.70_{\pm 0.58}$	$62.28_{\pm 1.23}$	$52.31_{\pm 0.86}$	$71.32_{\pm 0.28}$
H-LSTM$_{\text{c-gl}}$	$60.76_{\pm 0.99}$	$\mathbf{68.38_{\pm 0.65}}$	$\mathbf{57.17_{\pm 0.87}}$	$\mathbf{72.91_{\pm 0.14}}$
H-LSTM-CRF$_{\text{c-gl}}$	$59.83_{\pm 1.27}$	$68.10_{\pm 0.68}$	$56.37_{\pm 0.61}$	$72.77_{\pm 0.17}$

Table 5: Macro-F_1 scores for in-domain training.

ison, we use **accuracy** and **macro-F$_1$**. Like other related studies, we consider macro-F$_1$ as the main metric (more appropriate when class distributions are imbalanced), and select our model based on the best F_1 on the development set. Due to space limitations, we report only macro-F$_1$ here. Please refer to the Appendix for the accuracy numbers.

6.1 Experiments on In-domain Training

We first evaluate our base models on in-domain datasets by comparing with state-of-the-art models. In the next subsection, we evaluate our adaptation method in the three adaptation scenarios.

Settings. To validate the efficacy of our model, we compare it with two baselines: a Support Vector Machine (**SVM**) and a feed-forward network (**FFN**). In one setting, we use the concatenated word vectors as the input sentence representation, while in another, we use the pretrained skip-thought vectors (Kiros et al., 2015). We also compare our models with the bi-LSTM (**B-LSTM**) model of Joty and Hoque (2016) and the stacked LSTM (**S-LSTM**) of Khanpour et al. (2016).

We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001, and use dropout to avoid over-fitting. We use the Xavier initializer (Glorot and Bengio, 2010) to initialize the weights, and uniform $\mathcal{U}(-0.05, 0.05)$ to initialize the word vectors randomly. For pretrained word embeddings, we experiment with off-the-shelf embeddings that come with Glove as well as with our conversational word embeddings. For both random and pretrained initialization, we **fine-tune** our word embeddings on the SAR task.

We construct sequences from the **chronological order** of the sentences in a conversation. Since MRDA conversations are much longer compared to those in asynchronous domains (955 vs. 18-34 sentences in Table 1), we split the MRDA conversations into smaller parts containing a maximum of 100 sentences.[4] The number of epochs and batch size were fixed to 30 and 5 (conversations), respectively. We ran each experiment 5 times, each time with a different random seed, and report the average of the (2-fold×5=10) runs along with the standard deviation. Recently, Crane (2018) show that the main source of variability in results for neural models come from the random seed, and the author has recommended to report the distribution of results from a range of seeds.

Results. We present the results in Table 5. From the **first block** of results, we notice that both SVM and FFN baselines perform poorly compared to other models that tune the word embeddings and learn the sentence representation on the SAR task.

The **second block** contains five LSTM variants: (*i*) **B-LSTM$_{\text{rand}}$**, referring to bi-LSTM with *random* initialization; (*ii*) **B-LSTM$_{\text{gl}}$**, referring to bi-LSTM initialized with *off-the-shelf* Glove embeddings; (*iii*) **B-GRU$_{\text{c-gl}}$**, referring to bidirectional Gated Recurrent Unit (Cho et al., 2014) initialized with our *conversational* Glove; (*iv*) **B-LSTM$_{\text{c-gl}}$**, referring to bi-LSTM initialized with *conversational* Glove, and (*v*) **S-LSTM$_{\text{c-gl}}$**, referring to a 2-layer stacked LSTM with *conversational* Glove.[5] From the results, we can make the following conclusions. First, B-LSTM$_{\text{rand}}$ overfits extremely on

---

[4] In a different setting, we created sequences by connecting each non-initial comment with the initial comment generating many 2-comment sequences. This is considering the fact that in many QA forums, users mostly answer to the questions asked in the initial post. In our experiments on in-domain training, we found this competitive with our 'one long-chain' structure. However, the adaptation in this setting was much worse because of the mismatch in discourse structures of synchronous and asynchronous conversations.

[5] Increasing the number of layers in **S-LSTM$_{\text{c-gl}}$** did not give any gain (see Table 2 in the Appendix).

the asynchronous datasets, giving the worst results among the LSTMs. Second, pretrained vectors help to achieve better results, however, compared to the off-the-shelf vectors, our conversational word vectors yield much higher F_1, especially, in the asynchronous datasets that are smaller in size (5 - 11% absolute gains). This demonstrates that pretrained word embeddings provide an effective method to perform semi-supervised learning, when they are learned from relevant datasets.

The **last block** shows the results of our models. It is evident that both H-LSTM and H-LSTM-CRF outperform other models in all the datasets except QC3 where the difference is very small. They also give the best F_1 reported so far on MRDA, outperforming the B-LSTM models of Joty and Hoque (2016) and S-LSTM model of Khanpour et al. (2016). When we compare the two models, we notice that H-LSTM outperforms H-LSTM-CRF in all the datasets. A reason for this could be that the contextual dependency is already captured by the upper LSTM layer and the data may be too small for the CRF to offer anything more.

6.2 Experiments on Domain Adaptation

Settings. We compare our adversarial adaptation method with three baseline methods: Transfer, Merge and Fine-tune. **Transfer** models are trained on the source (MRDA) and tested on the target (QC3, TA, BC3). Our **adversarial unsupervised adaptation** method is comparable to the transfer method as they use labeled data only from the source domain. In **Merge**, models are trained on the concatenated training set of source and target datasets. **Fine-tune** is a widely used adaptation method for neural models (Chu and Wang, 2018). In this method, we first train a model on the source domain until convergence, then we fine-tune it on the target by training it further on the target dataset. Both merge and fine-tune are comparable to our **semi-supervised/supervised adaptation** as these methods use labeled data from the target domain. For semi-supervised experiments, we take smaller subsets (*e.g.,* 25%, 50%, and 75% of the labeled data) from the target domain.

We also compare our method with **Neural SCL** (Ziser and Reichart, 2017), which is another domain adaption method in the neural framework. We used the implementation made available by the authors.[6] For training our adaptation models, we use SGD (Algorithm 1 in the Appendix) with a momentum term of 0.9 and a dynamic learning rate as suggested by Ganin et al. (2016).

Method	**Model**	**QC3**	**TA**	**BC3**
Transfer	SVM	$17.78_{\pm 0.00}$	$20.44_{\pm 0.00}$	$17.85_{\pm 0.00}$
	FFN	$46.91_{\pm 0.00}$	$56.30_{\pm 0.00}$	$46.74_{\pm 0.00}$
	S-LSTM	$49.89_{\pm 1.29}$	$62.52_{\pm 1.49}$	$36.36_{\pm 1.28}$
	B-LSTM	$50.50_{\pm 0.91}$	$\mathbf{65.47}_{\pm \mathbf{0.62}}$	$35.92_{\pm 0.62}$
	H-LSTM	$50.22_{\pm 0.64}$	$64.43_{\pm 0.52}$	$35.11_{\pm 1.64}$
	H-LSTM-CRF	$\mathbf{50.83}_{\pm \mathbf{0.70}}$	$63.80_{\pm 0.81}$	$34.45_{\pm 1.42}$
Unsup. adapt	Neural SCL	$37.73_{\pm 0.92}$	$53.98_{\pm 0.33}$	$\mathbf{46.90}_{\pm \mathbf{0.89}}$
	Adv-S-LSTM	$43.36_{\pm 1.44}$	$48.51_{\pm 0.51}$	$42.05_{\pm 0.21}$
	Adv-B-LSTM	$47.39_{\pm 0.74}$	$58.49_{\pm 1.29}$	$32.86_{\pm 1.35}$
	Adv-H-LSTM	$46.53_{\pm 1.48}$	$52.90_{\pm 1.20}$	$31.36_{\pm 1.91}$
	Adv-H-LSTM-CRF	$47.06_{\pm 1.24}$	$61.58_{\pm 0.78}$	$29.54_{\pm 1.06}$
Merge (50%)	S-LSTM	$55.39_{\pm 0.29}$	$67.79_{\pm 0.15}$	$50.82_{\pm 0.98}$
	B-LSTM	$55.08_{\pm 0.67}$	$68.99_{\pm 0.35}$	$51.05_{\pm 0.60}$
	H-LSTM	$51.74_{\pm 0.44}$	$69.09_{\pm 0.64}$	$47.82_{\pm 1.47}$
	H-LSTM-CRF	$50.92_{\pm 0.48}$	$68.66_{\pm 0.12}$	$48.58_{\pm 0.59}$
Fine-tune (50%)	S-LSTM	$53.94_{\pm 1.04}$	$66.07_{\pm 0.67}$	$51.73_{\pm 1.53}$
	B-LSTM	$54.81_{\pm 0.48}$	$68.43_{\pm 0.51}$	$52.26_{\pm 0.82}$
	H-LSTM	$54.34_{\pm 0.71}$	$69.16_{\pm 0.66}$	$50.81_{\pm 0.97}$
	H-LSTM-CRF	$54.97_{\pm 0.87}$	$69.91_{\pm 0.53}$	$51.42_{\pm 1.24}$
Semisup. adapt (50%)	Neural SCL	$41.46_{\pm 0.75}$	$58.85_{\pm 0.27}$	$48.32_{\pm 0.19}$
	Adv-S-LSTM	$60.20_{\pm 0.32}$	$68.71_{\pm 0.75}$	$57.97_{\pm 0.39}$
	Adv-B-LSTM	$58.57_{\pm 0.80}$	$66.51_{\pm 0.81}$	$54.39_{\pm 1.28}$
	Adv-H-LSTM	$60.19_{\pm 1.10}$	$69.43_{\pm 0.64}$	$58.39_{\pm 1.12}$
	Adv-H-LSTM-CRF	$\mathbf{61.81}_{\pm \mathbf{0.63}}$	$\mathbf{70.34}_{\pm \mathbf{0.62}}$	$\mathbf{59.43}_{\pm \mathbf{1.41}}$
Merge (100%)	S-LSTM	$59.18_{\pm 1.40}$	$66.93_{\pm 1.70}$	$54.87_{\pm 1.47}$
	B-LSTM	$58.33_{\pm 0.84}$	$70.12_{\pm 0.39}$	$55.89_{\pm 0.89}$
	H-LSTM	$59.85_{\pm 0.57}$	$70.40_{\pm 0.41}$	$57.19_{\pm 0.87}$
	H-LSTM-CRF	$59.53_{\pm 0.66}$	$69.88_{\pm 0.68}$	$56.04_{\pm 1.15}$
Fine-tune (100%)	S-LSTM	$56.67_{\pm 0.85}$	$67.41_{\pm 0.34}$	$56.40_{\pm 0.44}$
	B-LSTM	$59.74_{\pm 0.53}$	$69.87_{\pm 0.82}$	$57.09_{\pm 1.14}$
	H-LSTM	$60.12_{\pm 0.44}$	$70.96_{\pm 0.61}$	$58.09_{\pm 1.03}$
	H-LSTM-CRF	$59.95_{\pm 0.59}$	$70.44_{\pm 0.76}$	$57.17_{\pm 0.97}$
Sup. adapt	Neural SCL	$43.35_{\pm 0.30}$	$60.40_{\pm 0.23}$	$48.88_{\pm 0.70}$
	Adv-S-LSTM	$61.15_{\pm 0.48}$	$70.33_{\pm 0.66}$	$59.19_{\pm 0.67}$
	Adv-B-LSTM	$60.60_{\pm 0.68}$	$69.30_{\pm 0.50}$	$60.12_{\pm 1.26}$
	Adv-H-LSTM	$\mathbf{63.10}_{\pm \mathbf{0.83}}$	$72.82_{\pm 0.59}$	$\mathbf{60.38}_{\pm \mathbf{1.07}}$
	Adv-H-LSTM-CRF	$62.24_{\pm 0.74}$	$\mathbf{73.04}_{\pm \mathbf{0.38}}$	$59.84_{\pm 0.75}$

Table 6: Domain adaptation results on our datasets. All models use conversational word embeddings. Results are averaged over (2 folds×5) 10 runs.

Results. The adaptation results are shown in Table 6. We observe that without any labeled data from the target (**Unsup. adap**), our adversarial adapted models (Adv-H-LSTM, Adv-H-LSTM-CRF) perform worse than the transfer baseline in all three datasets. In this case, since the out-of-domain labeled dataset (MRDA) is much larger, it overwhelms the model inducing features that are not relevant for the task in the target domain. However, when we provide the models with some labeled in-domain examples in the **semi-supervised (50%)** setting, we observe about 11% absolute gains in QC3 and BC3 over the corresponding Merge baselines, and 7 - 8% gains over the corresponding Fine-tune baselines. As we add more target labels (**100%**), performance of our adapted models (**Sup. adap**) improve further, yielding sizable improvements ($\sim$ 3% absolute) over the corresponding baselines in all datasets.

---

[6] https://github.com/yftah89/structural-correspondence-learning-SCL

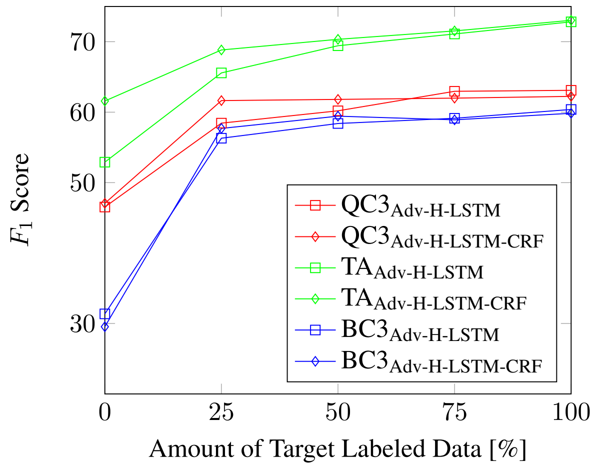


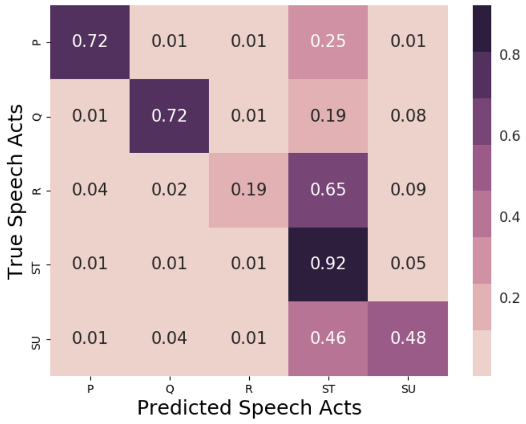
Figure 3: F_1 with varying amount of target labels.

Also notice that our adversarial adaptation outperforms Merge and Fine-tune methods for all models over all datasets, showing its effectiveness.

Figure 3 presents the F_1 scores of our adapted models with varying amount of labeled data in the target domain. We notice that the largest improvements for all three datasets come from the first 25% of the target labels. The gains from the second quartile are also relatively higher than the last two quartiles for TA and BC3. Another interesting observation is that H-LSTM-CRF performs better in unsupervised and semi-supervised settings (*i.e.,* with less target labels). In other words, H-LSTM-CRF adapts better than H-LSTM with small target datasets by exploiting the tag dependencies in the source. As we include more labeled data from the target, H-LSTM catches up with H-LSTM-CRF. Surprisingly, Neural SCL performs the worst. We suspect this is due to the mismatches between pivot features of the source and target domains.

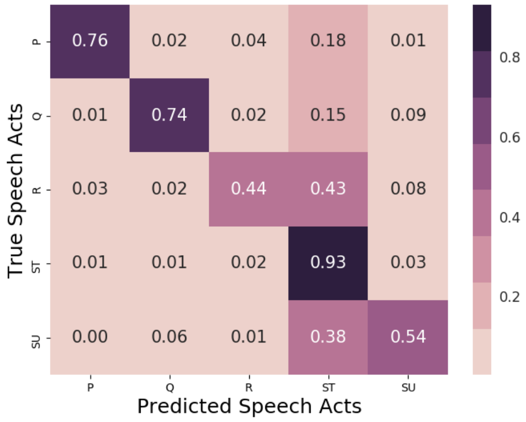
If we compare our **adaptation results** with the **in-domain results** in Table 5, we notice that using the same amount of labeled data in the target, our supervised adaptation gives 3-4% gains across the datasets. Our semi-supervised adaptation using half of the target labels (50%) also outperforms the in-domain models that use all the target labels.

To further analyze the cases where our adapted models make a difference, Figure 4 shows the confusion matrices for the adapted H-LSTM and the non-adapted H-LSTM on the concatenated testsets of QC3, TA, and BC3. In general, our classifiers get confused between *Response* and *Statement*, and between *Suggestion* and *Statement* the most. We noticed similar phenomena in the human annotations, where annotators had difficulties with these three acts. It is however noticeable that the adapted H-LSTM is less affected by class imbalance, and it can detect the *Suggestion* and *Polite* acts more correctly than the non-adapted one.



	P	Q	R	ST	SU
P	0.72	0.01	0.01	0.25	0.01
Q	0.01	0.72	0.01	0.19	0.08
R	0.04	0.02	0.19	0.65	0.09
ST	0.01	0.01	0.01	0.92	0.05
SU	0.01	0.04	0.01	0.46	0.48

(a) Non-adapted H-LSTM



	P	Q	R	ST	SU
P	0.76	0.02	0.04	0.18	0.01
Q	0.01	0.74	0.02	0.15	0.09
R	0.03	0.02	0.44	0.43	0.08
ST	0.01	0.01	0.02	0.93	0.03
SU	0.00	0.06	0.01	0.38	0.54

(b) Adapted H-LSTM

Figure 4: Confusion matrices on the combined test sets.

7 Conclusion

We proposed an adaptation framework for speech act recognition in asynchronous conversation. Our base model is a hierarchical LSTM encoder with a Softmax or CRF output layer, which achieves state-of-the-art results for in-domain training. Crucial to its performance is the conversational word embeddings. We adapted our base model with adversarial training to effectively leverage out-of-domain meeting data, and to improve the results further. A comparison with existing methods and baselines in different training scenarios demonstrates the effectiveness of our approach.

Acknowledgments

Shafiq Joty would like to thank the funding support from MOE Tier-1 (Grant M4011897.020).

References

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR*, abs/1701.07875.

John Langshaw Austin. 1962. How to do things with words. *Harvard University Press*.

Vitor R. Carvalho and William W. Cohen. 2005. On the collective classification of email "speech acts". In *SIGIR '05*, pages 345–352, New York, NY, USA.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP'14*, Doha, Qatar.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319. Association for Computational Linguistics.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *EMNLP'04*, pages 309–316.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. *TACL'18*, 6:241–252.

Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. 2004. Meeting Recorder Project: Dialog Act Labeling Guide. Technical report, ICSI Tech. Report.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *HLT'11: short papers-Volume 2*, pages 42–47.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *JMLR*, Sardinia, Italy.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *EMNLP'09*, pages 1250–1259, Singapore.

Shafiq Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised Modeling of Dialog Acts in Asynchronous Conversations. In *IJCAI'11*, pages 1–130, Barcelona.

Shafiq Joty and Enamul Hoque. 2016. Speech act modeling of written asynchronous conversations with task-specific embeddings and conditional structured models. In *ACL '16*, pages 1746–1756, Berlin, Germany.

Dan Jurafsky, Liz Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual. Technical report, UC Boulder & SRI International.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *CVSC'13*, pages 119–126.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *COLING'16*, pages 2012–2021.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3294–3302, Cambridge, MA, USA.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, Sachindra Joshi, and Arun Kumar. 2018. Dialogue act sequence labeling using hierarchical encoder with CRF. In *AAAI'18*.

John Lafferty, Andrew K. McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *HLT'16*, pages 515–520.

Annie P Louis and Shay B Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. In *ACL '15*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *NAACL-HLT '13*, pages 746–751, Atlanta, GA, USA.

Tatsuro Oya and Giuseppe Carenini. 2014. Extractive summarization and dialogue act modeling on email threads: An integrated probabilistic approach. In *SIGDIAL'14*, page 133–140, Philadelphia, PA, U.S.A.

Michael J. Paul. 2012. Mixed Membership Markov Models for Unsupervised Conversation Modeling. In *EMNLP-CoNLL '12*, pages 94–104, Stroudsburg, PA, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*, pages 1532–1543, Doha, Qatar.

Ashequl Qadir and Ellen Riloff. 2011. Classifying sentences as speech acts in message board posts. In *EMNLP'11*, pages 748–758, Edinburgh, Scotland, UK.

A. Ritter, C. Cherry, and B. Dolan. 2010. Unsupervised modeling of twitter conversations. In *HLT: NAACL'10*, LA, California.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI'16*.

Maryam Tavafi, Yashar Mehdad, Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2013. Dialogue act recognition in synchronous and asynchronous conversations. In *SIGDIAL'13*, Metz, France.

Jan Ulrich, Gabriel Murray, and Giuseppe Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *AAAI'08*, Chicago, USA.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *CoNLL 2017*, pages 400–410.

From Legal to Technical Concept: Towards an Automated Classification of German Political Twitter Postings as Criminal Offenses

Frederike Zufall[1†], **Tobias Horsmann**[2†], **Torsten Zesch**[2]
[1] Waseda Institute for Advanced Study, Waseda University, Tokyo, Japan
[2] Language Technology Lab, University of Duisburg-Essen, Germany
f.zufall@kurenai.waseda.jp
{tobias.horsmann|torsten.zesch}@uni-due.de

Abstract

Advances in the automated detection of offensive Internet postings make this mechanism very attractive to social media companies, who are increasingly under pressure to monitor and action activity on their sites. However, these advances also have important implications as a threat to the fundamental right of free expression. In this article, we analyze which Twitter posts could actually be deemed offenses under German criminal law. German law follows the deductive method of the Roman law tradition based on abstract rules as opposed to the inductive reasoning in Anglo-American common law systems. This allows us to show how legal conclusions can be reached and implemented without relying on existing court decisions. We present a data annotation schema, consisting of a series of binary decisions, for determining whether a specific post would constitute a criminal offense. This schema serves as a step towards an inexpensive creation of a sufficient amount of data for automated classification. We find that the majority of posts deemed morally offensive actually do not constitute a criminal offense and still contribute to public discourse. Furthermore, laymen can provide sufficiently reliable data to an expert reference but are, for instance, more lenient in the interpretation of what constitutes a disparaging statement.

1 Introduction

The Internet is frequently used for discussing a variety of topics and an important medium for the exchange of opinions, considered crucial for healthy democratic societies. However, the rough tone in the Internet frequently leads to defamatory or abusive comments in these discussions. The EU has tried to tackle the problem by defining the term 'illegal hate speech'.[1] Additionally, in 2017, the European Commission published a communication entitled 'Tackling Illegal Content Online' aiming for enhanced responsibility of online platforms.[2] Independently from these recent developments on the EU level, Germany adopted the 'Network Enforcement Act'[3] in 2017. The Act provides for a regulatory framework for 'illegal content'[4] on social network platforms like Twitter or Facebook. It imposes the obligation on these providers to delete illegal content upon notification within seven days; in case of evidently illegal content within 24 hours.[5] From a practical point of view, given the number of statements on social media along with their possible notification, feasibility and accuracy of the required legal assessment becomes an important issue. Natural Language Processing might thus provide the necessary means to assist the legal assessment.

In this work, we investigate at which point morally offensive statements in social media constitute defamatory offenses under the German Criminal Code (StGB)[6], thus representing 'illegal content' according to the Network Enforcement Act and thereby triggering a deletion obligation for platform providers.[7] We analyze the legal decision-making process to determine defam-

---

[†] Equal contribution.

[1] Framework Decision 2008/913/JHA of 28 November 2008 on combating certain forms and expressions of racism and xenophobia by means of criminal law and national laws transposing it.

[2] COM(2017) 555 final.

[3] Netzwerkdurchsetzungsgesetz v. 1.9.2017 (BGBl. I S. 3352).

[4] See § 1(3) 'rechtswidrige Inhalte'.

[5] See § 3(2)(2),(3) of the Act. It is however doubtful whether these strict procedural requirements violate EU law, namely Art. 3, Art. 14 e-Commerce Directive (2000/31/EC) i.e. require acting 'expeditiously' after obtaining knowledge.

[6] Strafgesetzbuch v. 13.11.1998 (BGBl. I S. 3322).

[7] It is not guaranteed that a judge would necessarily arrive at the same conclusion, but a lawyer's expertise serves as a strong indicator for potentially punishable conduct.

Proceedings of NAACL-HLT 2019, pages 1337–1347
Minneapolis, Minnesota, June 2 - June 7, 2019.

atory offenses (§ 185 to § 187 StGB), which also clarifies the tension between the right to honor and the freedom of expression. Due to its additional complexity, we leave out incitement to hatred against a national, racial, religious or ethnic group or segments of the population (§ 130 StGB) as an offense against public peace in this paper. Furthermore, we investigate automated detection of postings protected by the freedom of expression in order to assist social media moderators. We focus in particular on the process of inexpensive and scalable data annotation, as access to legal expertise is a major bottleneck for providing a sufficient amount of data for classifier training.

2 Related Work

An automated detection of Internet discourse in which individuals or groups are verbally attacked has been intensively investigated under a variety of names, for instance: abusive language (Waseem et al., 2017), ad hominem arguments (Habernal et al., 2018), aggression (Kumar et al., 2018), cyberbullying (Xu et al., 2012; Macbeth et al., 2013), hate speech (Warner and Hirschberg, 2012; Ross et al., 2016; Del Vigna et al., 2017), offensive language usage (Razavi et al., 2010), profanity (Schmidt and Wiegand, 2017), threats (Oostdijk and van Halteren, 2013) or socially unacceptable discourse (Fišer et al., 2017).

The majority of the work focuses on the English language with few exceptions for instance for German (Ross et al., 2016), Dutch (Oostdijk and van Halteren, 2013), Italian (Del Vigna et al., 2017) or Slovene (Fišer et al., 2017). The dataset annotated in Fišer et al. (2017) is the only one that includes a coarse-grained binary annotation category indicating if an utterance violates Slovene law. To the best of our knowledge, automatic determination as to whether the (textual) content of a posting constitutes a criminal offense has never been previously attempted. Previous work focused on detecting postings with socially unacceptable content but without considering actual legal implications for freedom of expression.

Approaches that bring together Natural Language Processing with the legal perspective are in contrast significantly fewer, especially considering the fact that the legal evaluation depends on the applicable legal regime. Previous work focused on predicting the outcome of court trials, which all have in common that they derive their data from a rather large set of court-provided information. Bruninghaus and Ashley (2003) works on a combination of U.S. case law and normative rules: they experiment with clustering and regression models for predicting the outcome of U.S. cases. Katz et al. (2017) predicts U.S. supreme court rulings by using a random forest classifier; Kastellec (2010) investigates mappings from case facts to court decisions as outcomes. Waltl et al. (2017) predicts the outcome of decisions in German tax law. Aletras et al. (2016) predicts decisions of the European Court of Human Rights. Deriving data from court decisions might be an approach that is practical if relevant case law exists for the respective legal problem, which particularly makes sense from the perspective of the Anglo-American common law system.[8]

3 Operationalising Legal Assessment

Unlike under Anglo-American common law, for legal systems based on Roman law (‘civil law’ systems), the dogmatic perception of the respective legal disposition lies at the heart of legal decision-making. Our approach thus differs from the above-cited works by placing the focus on the abstract concept of an existing legal norm. The advantage of our approach is therefore that we pursue a solution to address legal problems by creating new data out of abstract legal rules, independently of whether they have been decided by a court. We rely solely on the Internet posting for this consideration, which is the same information available to moderators of social media platforms. To build the bridge from legal thinking to a technical implementation, we start by analyzing the legal requirements for social media content. We find that the decision-making process to determine criminal offenses can be formulated as a sequence of binary decisions when applying the legal dependencies between German criminal law and the fundamental rights of the individual as shown in Figure 1. The derived schema of binary decisions is shown in Figure 2, which we will use in the following section. We now turn to a discussion and analysis of the legal decision process to clarify how we derived this sequence of binary decisions.

---

[8] ‘Common law’ refers to the Anglo-American legal system that derives the law from judicial decisions, in contrast to the ‘civil law’ system of continental Europe that focuses on the abstraction of legal concepts in codified statutory law. See: B.A. Garner (2001) A Dictionary of Modern Legal Usage (2nd, revised ed.) New York: OUP.

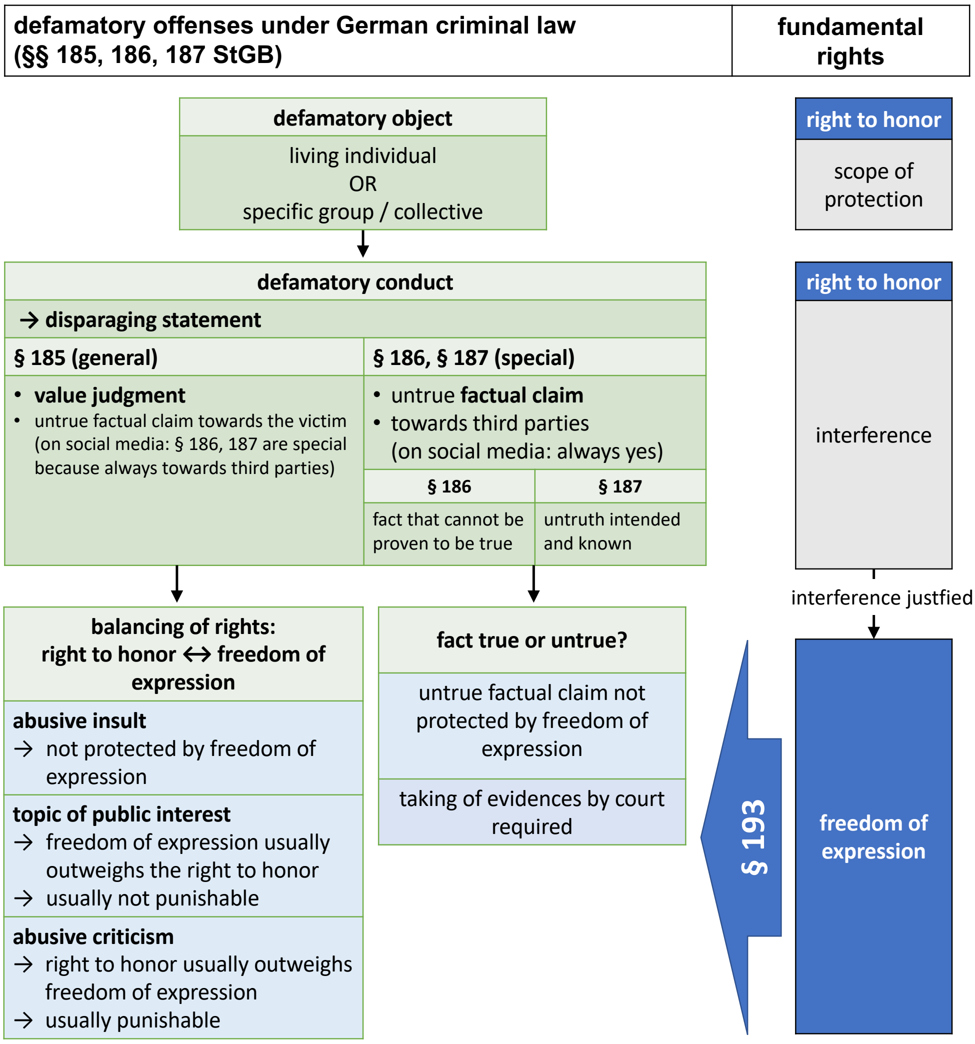


Figure 1: Conditions for defamatory offenses and fundamental rights' background under German law

Scope So what constitutes 'illegal content' that the Network Enforcement Act is targeting? The legal definition of the term 'illegal content'[9] is referring to offenses stipulated in the German Criminal Code. These references include, inter alia, defamatory offenses in § 185 to § 187 StGB[10] that cover the criminal punishment of insulting or defamatory statements. Accordingly, if a statement posted on social media fulfills the required elements of these offenses, the provider has the above-described obligation based on the Network Enforcement Act to delete said statement upon notification.[11] For this paper, we exclude § 130 StGB[12], that covers incitement to hatred against a national, racial, religious group or a group defined by their ethnic origins, due to an additional complexity of the assessment.

3.1 The Relevance of Fundamental Rights

To understand their elements in detail, it is crucial to refer to the more general legal concept behind these criminal offenses: as illustrated in Figure 1 the intention behind § 185 to § 187 StGB is leading to the protection of the victim's personality right, namely their right to honor under the German Constitution.[13] It is this right that is potentially at stake when social media users are disseminating statements with third parties as potential victims.

---

[9] See § 1(3) 'rechtswidrige Inhalte'.

[10] § 185: 'insult'(*Beleidigung*), § 186: 'defamation' *Üble Nachrede*) and § 187: 'intentional defamation' (*Verleumdung*). The reference to these defamatory offenses has however been criticized in literature, see: Erbs/Kohlhaas, Strafrechtliche Nebengesetze, 220. EL Juli 2018, § 1 NetzDG, Rn. 16-18.

[11] See § 3(2)(2),(3) of the Act.

[12] § 130, 'incitement to hatred' (*Volksverhetzung*).

[13] Derived from Art. 2(1) and Art. 1(1) of the German Constitution (Grundgesetz); BVerfGE 35, 202; E 54, 148, 155.

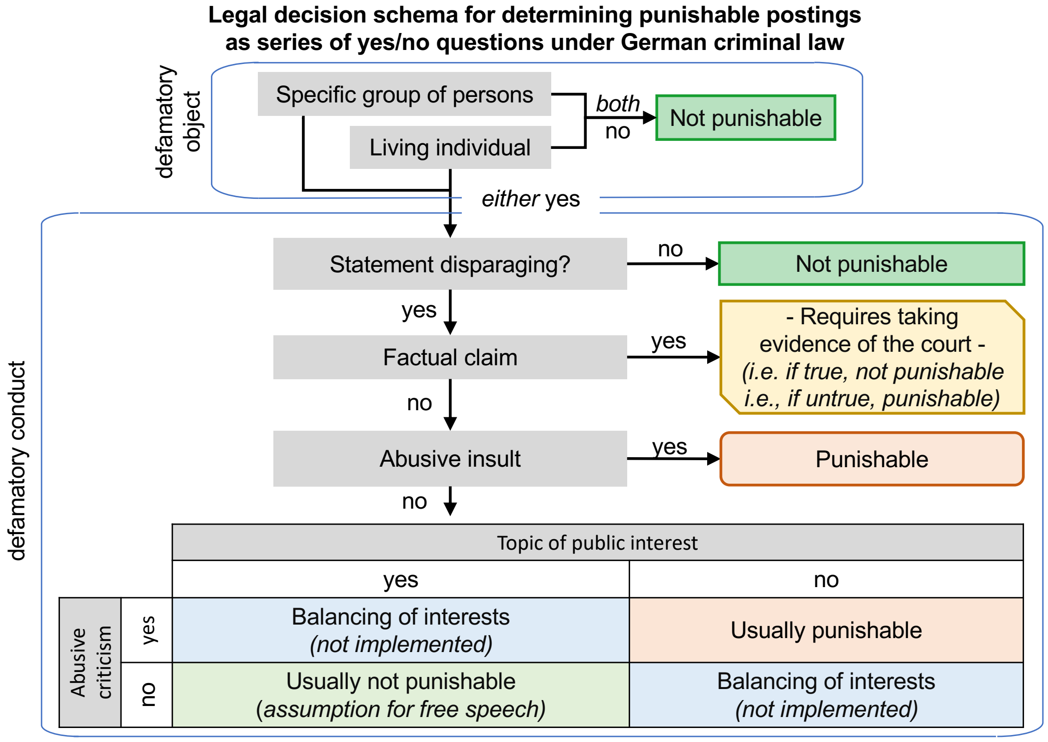


Figure 2: Series of binary decisions for determining criminal offenses under German criminal law

3.2 Defamatory Object

Consequently, the scope of protection of § 185 to § 187 StGB follows the respective interpretation of the right to honor. Thus, all three offenses share the approach to the possible victim as a holder of the right to honor: a **living individual** that might be addressed by a name, personal pronoun or user-mention as shown in Example 1.

(a) Are *you* kidding?
(b) *John* is this true?
(c) *@user* I don't believe you.

Example 1: Addressing living individuals

A **group of persons** can be considered as a potential victim if that group is distinguishable from the general public such that every member of that group could feel their honor is infringed as shown in Example 2.[14]

(a) *My school's language teachers* are all idiots.
(b) *The female students of this year's grad class* are all dump.

Example 2: Addressing distinguishable group of persons (highlighted in italic)

Consequently, only certain groups do qualify as potential defamatory object. Example 3 illustrates groups that would be too broad to be distinguishable from the general public.[15]

(a) All international conflicts are caused by *men*.
(b) *Refugees* out!!

Example 3: Counterexamples for addressing too unspecific or large groups

Collective entities such as governments or press companies with a recognized social role and who act with a collective, single will are included in the right to honor as shown in Example 4.[16]

(a) *The federal government* is lying.
(b) I don't like *the Christian Democratic Party*
(c) *The New York Times* got it all wrong.

Example 4: Addressing collective entities (highlighted in italic)

We translate these conditions of § 185 to § 187 StGB into an either/or-question, respectively whether *either* a living individual *or* a specific group is an object of the respective statement.

---

[14] BGHSt 19, 235, 238.

[15] These groups may, however, still qualify as a potential victim of 'incitement to hatred' (§ 130).

[16] See § 194 StGB; BGHSt 6, 186, 191, 192.

3.3 Defamatory Conduct

Disparaging Statement The next step in the legal assessment is then the existence of insulting or defamatory conduct with respect to the above-mentioned object, in the form of an expressed disparaging statement. This requirement is again shared by § 185 to § 187 StGB. It is already fulfilled by expressing contempt or disrespect through the allegation of shortcomings that could reduce the victim's social standing as shown in Example 5.[17]

(a) John is *an idiot.*
(b) The government is *lying*.
(c) Minister M *slept during the discussion.*

Example 5: Disparaging statements

From the perspective of the underlying fundamental rights, it is this disparaging statement which constitutes the interference with the potential victim's right to honor. The existence of a *disparaging statement* is implemented by a *yes/no-question.*[18]

Value Judgment or Factual Claim? As simplified in Figure 1, the legal assessment then varies between § 185 StGB as general disposition and § 186 and § 187 StGB with special rules and an increased penalty range.

For the different scope of these dispositions, the difference between the legal terms *'value judgment'* and *'factual claim'* (i.e. the assertion of facts, may they be true or untrue) is crucial. Value judgments constitute expressions of personal opinions as shown in Example 6:[19]

(a) *Merkels decisions are bullshit.*
(b) *@user I don't like you.*

Example 6: Value judgments

A factual claim can be clearly classified as true or untrue and is accordingly capable of proof as shown in Example 7.[20]

(a) *I saw Mr. A buying drugs yesterday evening.*
(b) *Minister M slept during the discussion.*

Example 7: Factual claim

§ 185 StGB, stipulating the insult ('Beleidigung'), comprises value judgments and untrue factual claims, irrespective of their dissemination towards third parties. § 186 and § 187 StGB on the other side provide for special rules for the assertion or dissemination of untrue facts, i.e. towards third parties. As the publication of statements on social media constitutes an 'assertion' or 'dissemination', untrue facts - for our study - are only treated by § 186, § 187 StGB. This reduces the scope of § 185 StGB to value judgments only.

From the perspective of the right to honor, only untrue factual claims may constitute a violation, while the assertion of true facts is always covered by the freedom of expression.[21] The distinction has consequences on the procedural level: because only the assertion of untrue facts violates the right to honor, during criminal proceedings, the court has to assess the truth by taking evidence. A technical implementation of this assessment would therefore require access to unlimited knowledge that goes beyond the textual information on which we work. Accordingly, we stop our examination in case of a factual claim.[22]

3.4 Value Judgments: Balancing of Rights

As the distinction between value judgment and factual claim is an alternative decision,[23] we continue our implementation for value judgments. In criminal proceedings, the court would have to consider at this point once more fundamental rights: value judgments - being not classifiable as untrue or true - generally fall under the scope of the freedom of expression of the potential offender.[24]

In the German Criminal Code, this is reflected by § 193 StGB: even if a statement falls under the scope of said criminal offenses, it might still be

---

[17] BGHSt 36, 145, 148.

[18] Regarding the mere *expression* of a statement, we assume a *yes-answer* for statements published on social media that are subject to this study, and therefore do not implement this condition.

[19] BVerfGE 61, 1; for Twitter postings: OLG Karlsruhe, 24.10.2018, 6 U 65/18.

[20] RG 41, 193; 55, 131; BVerfGE 94, 8.

[21] BVerfGE 99, 185, 197; E 97, 381, 403.

[22] Consequently, we do not implement subsequent conditions of § 186, § 187 StGB, as shown in Figure 1, respectively whether facts cannot be proven to be true (§ 186 StGB) or whether the untruth was intended and known (§ 187 StGB).

[23] Ambiguous statements that are based on facts, but are overall characterized by a valuation of these facts, fall under the category of 'value judgments'.

[24] Art. 5(1)1 of the German Constitution (Grundgesetz). According to Art. 5(2) the freedom of expression then again is limited by the right to honor.

justified based on § 193 StGB as exercise of legitimate interests. The most prominent example of one of these conflicting interests is the offender's freedom of expression. On the constitutional level, then, the decision of whether a social media posting constitutes a punishable criminal offense and leads to the platform provider's deletion obligation can thus ultimately be perceived as a balancing between freedom of expression and the right to honor.

Consequently, the court would have to balance these concurrent rights depending on the case at hand. But how could that balancing, usually comprising an evaluation of various factors, be carried over to a technical implementation? Over the years, German case law from the Federal Constitutional Court has developed guidelines for this balancing to be considered by the judge, which take the step of implying the typical outcome of the balancing. We implement these guidelines in three yes/no-questions:[25]

Abusive Insult Statements that constitute breaking a taboo by themselves and intend only the defamation of the victim without any substantiated contribution are classified as *'abusive insult' (Formalbeleidigung)*. According to settled case law, these statements are already excluded from the scope of freedom of expression.[26] Consequently, a justification based on § 193 StGB is, in this regard, denied and the elements of § 185 StGB are fulfilled along with a violation of the right to honor. Given these severe consequences for free speech, the German Constitutional Court has so far only once approved a statement as constituting an 'abusive insult' as shown in Example 8:[27]

A disabled person is called *"cripple"*

Example 8: Abusive insult

Topic of Public Interest For statements that contain a contribution to the public discourse with respect to a particular relevant topic of public interest, the settled case law of the German Federal Constitutional Court mandates a presumption in favor of free speech.[28]

Merkel prostitutes herself for the German car industry costing tax payers

Example 9: Topic of public interest

Example 9 comments on the right to stay of refugees, by this participating to the public debate in Germany about refugees from Syria. Accordingly, such statements usually outweigh the right to honor. They thus usually can be made, justified as having a legitimate interest based on § 193 StGB, therefore *usually not punishable*.

Abusive Criticism Finally, as *'abusive criticism' (Schmähkritik)* settled case law has defined statements that go beyond plausible criticism by primarily intending to abusively offend the victim, hereby neglecting a substantiated contribution.[29] In Example 10, the statement:

Minister M, that asshole, is lying to all of us!! noone has money to pay for this...

Example 10: No abusive criticism

despite the word 'asshole', still contributes to the public discourse, which is why its primary purpose is not (only) to abusively offend. Abusive criticism thus usually leads to favoring the right to honor over freedom of expression. Without justification pursuant to § 193 StGB, such statements are therefore *usually punishable*.

4 Proof of Concept

In this section, we now use the schema in Figure 2 to annotate data and learn more about the reliability of an automated classification.

4.1 Dataset

In order to legally assess social media postings, we first need to annotate a corpus as a starting point for an analysis. Randomly sampling postings from the Internet is a possible strategy to collect data for an annotation, but we would not have any certainty that enough offending postings occur. Therefore, we decide to use an existing corpus that has already been annotated for moral offensiveness. We use the corpus provided by the GermEval shared

---

[25] As illustrated in Figure 2, the judge would perform the balancing freely based on all circumstances (which we do not implement) if there is no *'abusive insult'* and if *'topic of public interest'* and *'abusive criticism'* are both yes or both no.

[26] Maunz/Dürig, Grundgesetz-Kommentar, 84. EL August 2018, Art. 5 Abs. 1, Rn. 62.

[27] BVerfGE 86, 1, 45 (*"Krüppel"*).

[28] BVerfGE 7, 198, 212.

[29] BVerfGE 61, 1, 12; E 82, 272 (284); for Twitter postings: LG Berlin, 13.10.2012, 33 O 434/11.

Decision	Agreement Acc	 Cohen's κ
Living individual	.985	.961
Specific group	.940	.809
Disparaging	.925	.867
Factual claim	.925	.821
Abusive insult	.925	.820
Of public interest	.940	.855
Abusive criticism	.866	.678
Joint-decision	.821	.720

Table 1: Agreement between two legal experts on two-hundred randomly selected postings

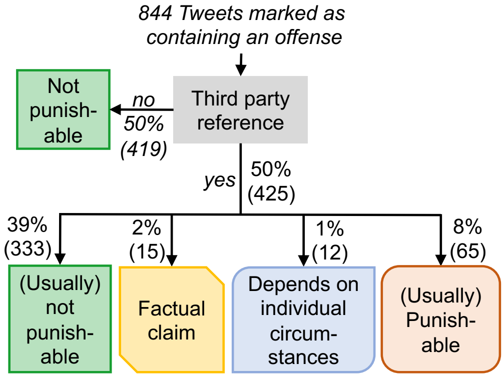


Figure 3: Legal categorization of annotated Tweets that were marked as containing an offense

task for *detecting offensive language usage* (Ruppenhofer et al., 2018). This dataset contains a mixture of German Twitter postings with a focus on German politics that are marked if the tweet is considered morally offensive from the subjective perception of the annotator. We work with a subset of 1,100 postings from this corpus, two-thirds of the postings (844) are marked as morally offensive. This enables us to investigate which statements commonly found in political debates are protected by the freedom of expression and which are not.

Annotation The reference annotation of these postings is provided by a fully-qualified lawyer of German law applying the schema in Figure 2. We additionally received 200 postings from a second fully-qualified lawyer in order to compute an agreement score between the two legal experts, which is shown in Table 1. We report accuracy and Cohen's κ (Cohen, 1960) for each decision and show the agreement for a *joint-decision* where we treat all decisions for a posting as a single decision. The legal experts disagree slightly on the assumption of abusive criticism. This is not surprising as the evaluation of courts might differ in different instances, especially regarding the balancing of interests in the case at hand.

Analysis Figure 3 shows the annotation results of the postings marked as *morally offensive*. We find that about half of the postings have to be categorized early on as *not punishable* for not containing a defamatory object, i.e. no living individual addressed or the addressed group is too unspecific. The remaining half is still to a large extent *usually not punishable*, mostly because the posts still contribute to a topic of public interest, despite of being disparaging. A small number of cases are either factual claims that would require taking evidence by the court or value judgments that do not concern topics of public interest. Thus, despite containing statements that may be deemed morally offensive, the vast majority of statements are legally acceptable, i.e. protected by the freedom of expression. The punishable cases often contain insulting buzzwords such as *slut, fat-ass* or *scumbag* when directed at a private individual, not at a person of public interest. Furthermore, punishable statements addressing a specific group use more frequently offending comparisons or descriptions but no typical single or two-word insults. However, it is important to recall that the dataset has a focus on political debates. Accordingly, most statements tackle a topic of public interest, and are thus considered *usually not punishable* granting a high degree of protection under the freedom of expression.

This analysis also shows that shared tasks such as *OffensEval* (Zampieri et al., 2019) tackle essentially only one step in the legal assessment, namely whether a statement is *disparaging*. Thus, they fall short of valuing the freedom of expression, which is in particular a problem for public discourse such as political debates, where opinions are often accompanied by 'bad' language.

4.2 Automated Detection

For an automated detection, it would seem straight forward to distinguish between *punishable* and *not punishable* postings. This approach requires an extremely large amount of data for each of the two classes, which we do not have. The data distribution is skewed with the *punishable* class being extremely small, which makes this direct ap-

	Decision	Acc	Class	F1
Defam. Object	Living individual	.794	Yes	.685
			No	.847
	Specific group	.835	Yes	.431
			No	.903
	either of the above	.744	Yes	.775
			No	.702
Defamatory Conduct	Disparaging	.727	Yes	.656
			No	.774
	Factual claim	.977	Yes	.000
			No	.989
	Abusive insult	.984	Yes	.000
			No	.992
	Is of public interest	.715	Yes	.514
			No	.798
	Abusive criticism	.952	Yes	.036
			No	.975

Table 2: Averaged 10-fold CV results for each decision on 1,100 Tweets, using an LSTM

proach infeasible. Instead, we investigate how well each of the binary decisions shown in Figure 2 can be learned independently, which has a less skewed distribution. We use a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) for classification.[30] We use the 300-dimensional German pre-trained word embeddings provided by Grave et al. (2018), which are trained on the German common crawl.

Table 2 shows averaged 10-fold CV results for each decision point. We observe that the accuracy is close to the underlying distribution of the two classes. The classification of the *defamatory object* has a mediocre performance. In particular, an insufficient coverage of group names and names of individuals in the dataset seem to be the main cause as the *no* classes usually perform considerably better than the *yes* classes. The classification of the decisions under *defamatory conduct* follows a similar trend. The few positive instances: *factual claim*, *abusive insult* and *abusive criticism* prevent a reliable distinction of these cases.

The next step would be to investigate how well the classification works in sequence, i.e. continuing the classification with the positively categorized instances of the previous step. However, the independent classification shows already that the amount of data is insufficient. Therefore, we turn next to the more pressing question of how to generate more data in a scalable way, especially without relying on expensive legal experts as annotators.

5 Data Annotation by Laymen

A scalable annotation of more data requires that laymen can be instructed in a way that enables them to solve the task at hand. Laymen are readily available, for instance via crowdsourcing but also as student assistants who can be more cheaply employed than legal experts for annotating data.

Setup We compare the annotation performance of both random crowd workers and student assistants. The crowd workers and the student assistants were required to speak German. We have no information on the educational background of the crowd workers, but we ensured that the student assistants were not students of law-related subjects. We prepared a simplified manual[31] based on Figure 2, which is supplemented with text examples for each decision to guide the layman through the annotation of each decision. We use the crowdsourcing platform figure-eight.com to let crowd workers and student assistants re-annotate the 1,100 postings for which we have a reference annotation by a legal expert. Each posting is annotated by three annotators.

The annotation results are shown in Table 3. It is to be expected that some annotators will perform better than others, but distinguishing the ‘good’ from the ‘bad’ is an additional challenge, which we will not deal with here. Instead, we aggregate the annotations of all participants in a voting-like fashion, taking in each case the majority vote for each decision.[32] This provides us with an approximation of the average layman performance on this task, which is the key information that we are interested in.

Analysis The results show that student assistants solve this task considerably better than crowd workers. In particular, the recognition of references to *specific group* poses the biggest challenges for crowd workers, which also explains why this group performs much more poorly than the student assistants. As shown in Figure 2, the evaluation for a post ends if neither a living individual nor specific group is addressed. If either of the first two decisions is incorrect, an annotator automatically makes up to five additional follow-up errors. The student assistants applied the manual considerably more consistently than the crowd

---

[30] We train 30 epochs, with 0.2 dropout and initialize using Glorot, and ReLU as activation function.

[31] `github.com/Horsmann/NAACL-2019-legal`

[32] We restrict the comparisons to postings for which we have three votes of the respective sub-group.

Decision	All users Acc	All users κ	Crowd-workers Acc	Crowd-workers κ	Student assistants Acc	Student assistants κ
Living individual	.822	.628	.800	.555	.826	.655
Specific group	.745	.357	.600	.192	.817	.502
Disparaging	.649	.401	.475	.158	.765	.574
Factual claim	.654	.381	.492	.131	.774	.577
Abusive insult	.590	.285	.400	.005	.669	.436
Is of public interest	.672	.263	.592	-.043	.691	.388
Abusive criticism	.589	.161	.575	-.049	.530	.224
Joint-decision	.357	.201	.175	.050	.383	.250

Table 3: Agreement between the reference annotation by a legal expert and the aggregated laymen annotations of: *all users* (on 1,000 posts), only *crowd-workers* (on 402 posts) and only *student assistants* (on 390 posts). Results for *crowd-workers* and *student-assistants* are limited to postings where all three votes per posting were provided by users from the respective group.

workers, leading to fewer follow-up errors. Determining the referenced individual is also frequently challenging when several Twitter users are referenced by an at-mention, which introduces an uncertainty that the statement might refer to one of the linked users. We also find that the laymen tend to apply a more lenient interpretation of what is *disparaging* and consider many statements as non-disparaging, i.e. already an allegation of shortcomings[33], which could reduce the victim's social standing is disparaging in the legal sense.

The annotation results of the student assistants are encouraging for obtaining sufficient training data for a larger study on automated classification, i.e. a correct automated classification of the first two decisions would already be able to exclude many cases that do not have to be deleted based on the Network Enforcement Act.

6 Conclusion

We investigated which offenses found in German political Tweets constitute defamatory offenses under German criminal law, that social media operators are obliged to delete under the Network Enforcement Act. Following the dogmatic approach of civil law systems, we started with an analysis of the legal framework for defamatory offenses in the German Criminal Code along with its foundations in the balancing between the potential offender's freedom of expression and the potential victim's right to honor. We derived from this consideration a schema suited for data annotation consisting of a sequence of binary decisions to determine if a statement constituted a defamatory offense, which we used for annotating data. We find that the majority of the morally offensive postings in our dataset still contribute to the public discourse and are, hence, protected by the freedom of expression. We also investigated if laymen can be instructed to use this annotation schema to facilitate an inexpensive annotation of more data for classifier training. We find that laymen suited to the task can be found, but in particular the legal notions of a specific *group of persons* and the scope of what is considered *disparaging* are challenging for them.

In future work, we will investigate the usefulness of layman-annotated data for an automated classification. Furthermore, we will expand our work by investigating additionally the criminal offense of *incitement to hatred* (§ 130 StGB) and its implication on the freedom of expression.

Acknowledgements

We would like to thank Tatiana Günster for taking the time to provide us with a second legal opinion. Furthermore, we would like to thank Emilie Mathieu for helpful corrections and Michael Wojatzki for the helpful discussions about the user-study design.

---

[33] e.g. *I am not sure whether John knows what he's doing.*

References

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. 2016. Predicting judicial decisions of the European Court of Human Rights: a Natural Language Processing perspective. *PeerJ Computer Science*.

Stefanie Bruninghaus and Kevin D. Ashley. 2003. Predicting Outcomes of Case Based Legal Arguments. In *Proceedings of the International Conference on Artificial Intelligence and Law*, pages 233–242, New York, NY, USA. ACM.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate Me, Hate Me Not: Hate Speech Detection on Facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2017. Legal framework, dataset and annotation schema for socially unacceptable online discourse practices in slovene. In *Proceedings of the First Workshop on Abusive Language Online*, pages 46–51. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before Name-Calling: Dynamics and Triggers of Ad Hominem Fallacies in Web Argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 386–396. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, pages 1735–1780.

Jonathan Kastellec. 2010. The Statistical Analysis of Judicial Decisions and Legal Rules with Classification Trees. *Journal of Empirical Legal Studies*, 7(2):202–230.

Daniel Martin Katz, Michael J. Bommarito, II, and Josh Blackman. 2017. A general approach for predicting the behavior of the Supreme Court of the United States. *PLOS ONE*, 12(4):1–18.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11. Association for Computational Linguistics.

Jamie Macbeth, Hanna Adeyema, Henry Lieberman, and Christopher Fry. 2013. Script-based story matching for cyberbullying prevention. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 901–906.

Nelleke Oostdijk and Hans van Halteren. 2013. N-Gram-Based Recognition of Threatening Tweets. In *Computational Linguistics and Intelligent Text Processing*, pages 183–196, Berlin, Heidelberg. Springer Berlin Heidelberg.

Amir H. Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence*, pages 16–27, Berlin, Heidelberg. Springer-Verlag.

Björn Ross, Michael Rist, Guillermo Carbonell, Ben Cabrera, Nils Kurowsky, and Michael Wojatzki. 2016. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. In *Proceedings of NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, pages 6–9.

Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand. 2018. GermEval 2018: Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval 2018: Shared Task on the Identification of Offensive Language*, Vienna, Austria.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10. Association for Computational Linguistics.

Bernhard Waltl, Georg Bonczek, Elena Scepankova, Jörg Landthaler, and Florian Matthes. 2017. Predicting the Outcome of Appeal Decisions in Germany's Tax Law. In *Electronic Participation*, pages 89–99, Cham. Springer International Publishing.

William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from Bullying Traces in Social Media. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 656–666, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The International Workshop on Semantic Evaluation (SemEval)*.

Joint Multi-Label Attention Networks for Social Text Annotation

Hang Dong[1,2], **Wei Wang**[2], **Kaizhu Huang**[3], and **Frans Coenen**[1]

[1]Dept. of Computer Science, University of Liverpool
[2]Dept. of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University
[3]Dept. of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University
`{HangDong,Coenen}@liverpool.ac.uk`
`{Wei.Wang03,Kaizhu.Huang}@xjtlu.edu.cn`

Abstract

We propose a novel attention network for document annotation with user-generated tags. The network is designed according to the human reading and annotation behaviour. Usually, users try to digest the title and obtain a rough idea about the topic first, and then read the content of the document. Present research shows that the title metadata could largely affect the social annotation. To better utilise this information, we design a framework that separates the title from the content of a document and apply a title-guided attention mechanism over each sentence in the content. We also propose two semantic-based loss regularisers that enforce the output of the network to conform to label semantics, i.e. similarity and subsumption. We analyse each part of the proposed system with two real-world open datasets on publication and question annotation. The integrated approach, Joint Multi-label Attention Network (JMAN), significantly outperformed the Bidirectional Gated Recurrent Unit (Bi-GRU) by around 13%-26% and the Hierarchical Attention Network (HAN) by around 4%-12% on both datasets, with around 10%-30% reduction of training time.

1 Introduction

Social annotation, or tagging, is a popular functionality allowing users to assign "keywords" to online resources for better semantic search and recommendation (Vander Wal, 2007; Singer et al., 2014; Gedikli and Jannach, 2014). Common socially annotated textual resources include questions, papers, (micro-)blogs, product reviews, etc. In practice, however, only a limited number of resources is annotated with tags. Annotating a large number of documents requires much cognitive effort and can be time-consuming. This has driven research on document annotation based on existing tag sets (Belém et al., 2017; Nie et al., 2014).

Recent studies formalise the automated social annotation task as a multi-label classification problem (Gibaja and Ventura, 2015) and apply deep learning approaches (Li et al., 2016; Huang et al., 2016; Hassan et al., 2018). A strong baseline is the use of Bi-directional RNN (Schuster and Paliwal, 1997) with GRU (Cho et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997). Another more recent improvement is achieved through Hierarchical Attention Network (HAN) (Yang et al., 2016) which discriminates important words and sentences from others, as adapted in (Hassan et al., 2018) for annotation. These models, however, suffer from two issues: (i) simply scanning over the words and sentences, the models do not fully mimic the way users read and annotate documents, and (ii) semantic relations, similarity and subsumption, among the labels are not considered.

Our model focuses on simulating users' reading and annotation behaviour with attention mechanisms. The title of a document is highly abstract while informative about the topics and has a direct impact on users' annotation choice (Lipczak and Milios, 2010), showing high descriptive capacity and effectiveness for annotation (Figueiredo et al., 2013); the content provides complementary information for annotation. Usually, users firstly read the title, and based on their understanding of the title, proceed to the content of the document. To simulate this behaviour, we propose an attention network with separated inputs (title and content) and parallelled attention layers at both the word-level and the sentence-level. One major distinction to previous approaches is to represent the content with a title-guided attention mechanism; this enables the network to discriminate among sentences based on its understanding of the title.

In addition, in the social context, users tend to annotate documents collectively with tags of

Proceedings of NAACL-HLT 2019, pages 1348–1354
Minneapolis, Minnesota, June 2 - June 7, 2019.

various semantic forms and granularities (Peters, 2009; Heymann and Garcia-Molina, 2006). One challenging issue is how to exploit the relations among labels (user-generated tags) (Zhang and Zhou, 2014; Gibaja and Ventura, 2015) to improve the learning performance. Among neural network based methods, a recent attempt is to initialise weights for dedicated neurons in the last layer to memorise the label relations (Kurata et al., 2016; Baker and Korhonen, 2017), however, the limitation is the large number of neurons to be assigned, making it inefficient (or inapplicable) for systems with large number of labels. To incorporate the label semantics inferred from the data or from external knowledge bases into the network, we design two loss regularisers, for similarity and subsumption relations, respectively. The regularisers enforce the output layer of the network to satisfy the semantic constraints of the labels.

2 Proposed Method

We propose a parallelled two-layered attention network that simulates users' reading and annotation behaviour for document annotation. The proposed Joint Multi-label Attention Network (JMAN) approach is depicted in Figure 1. The model inputs the title and content separately into two Bidirectional-RNNs with word-level attention and sentence-level attention mechanisms to capture the important words and sentences. Each target is a multi-hot (as opposed to an one-hot) representation of the labels in the label set $y_d \in \{0,1\}^{|T|}$, where T is a list all labels, "1" indicates that a label appears in the label set of the document d, "0" otherwise. In Figure 1, attention mechanisms are indicated with dotted edges. One key distinction from the HAN model (Yang et al., 2016) is the title-guided sentence-level attention that models the reading order for annotation (the dotted edges linking c_t and c_{ta}). The output layer $s_d = \sigma(W_c c_d + b_c)$, activated with the sigmoid function σ, is further constraint by two loss regularisers, emphasising two types of label relations, similarity and subsumption, respectively.

For the RNN encoder, we apply the Gated Recurrent Unit (GRU) which can capture long term dependencies and is usually more time-efficient than LSTM (Hochreiter and Schmidhuber, 1997) in training. The Bidirectional-GRU (Bi-GRU) encoder (Cho et al., 2014) concatenates the hidden states generated from two GRUs, one reading the words (or sentences) forward and the other reading them backwards. This helps form a more complete understanding of the current word (or sentence).

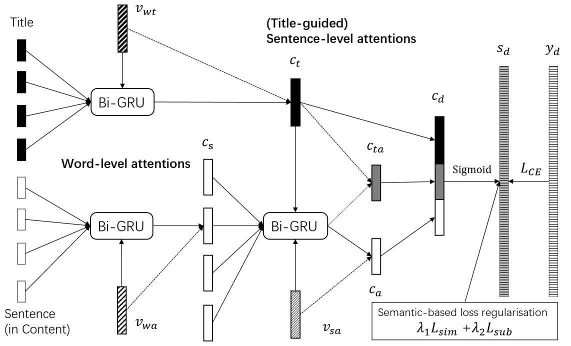


Figure 1: The Proposed Joint Multi-label Attention Network (JMAN) for Social Text Annotation

2.1 Hierarchical Attention

Hierarchical Attention captures the structure of a document by a word-level attention on each word's hidden state to create a sentence representation, then a sentence-level attention to form a content representation (Yang et al., 2016). The attention coefficients are computed based on the dot product between a non-linearly transformed weight vector of the hidden state and an "informative" vector, which encodes "what is the most informative word (or sentence)" in the sequence. This "informative" vector is commonly treated as a sequence of weights (Yang et al., 2016; Kumar et al., 2018; Hassan et al., 2018), trained along with other weights in the network. We applied parallelled word-level attention on the title and each sentence in the content. The attention coefficient and the final representation of a sequence is calculated as (taking words in title as an example):

$$c_t = \sum_i \alpha_i h_i = \sum_i \frac{\exp(v_{wt} \bullet v_i)}{\sum_j \exp(v_{wt} \bullet v_j)} h_i \tag{1}$$

where $v_i = \tanh(W_t h_i + b_t)$ is the output of a fully-connected layer of the hidden state h_i for each word in the title, v_{wt} is the "informative" vector for titles, and c_t is the resulting title representation. We can compute each sentence representation c_s and the content representation c_a in a similar manner (see Figure 1).

2.2 Title-guided Sentence-level Attention

The attention mechanisms above do not capture the interaction between the title and content of the document. Title represents highly abstract while important information about the topics of a document. Selection of the important sentences in the content should conform to the document's general topic, e.g. title. We can thus model the title-guided sentence-level attention as:

$$c_{ta} = \sum_{r} \alpha_r h_r = \sum_{r} \frac{\exp(c_t \bullet v_r)}{\sum_k \exp(c_t \bullet v_k)} h_r \tag{2}$$

where $v_r = \tanh(W_s h_r + b_s)$ is a fully connected layer with the hidden state of the rth sentence h_r as input and c_t is the title representation obtained from Equation 1.

Guiding sentence reading through title representation facilitates content understanding, but may lead to an overemphasis on the title in the annotation. In fact, the content itself, carrying more terms, conveys detailed information not covered by the title and may help suggest further tags for annotation (Figueiredo et al., 2013). We thus concatenate the title guided content representation c_{ta} and the content representation c_a from the original sentence-label attention, to form a more comprehensive representation of the content. The final content representation is then concatenated with the title representation $c_d = [c_t, c_{ta}, c_a]$. In the experiment, we will show the effectiveness of this design against several variations of the model.

2.3 Semantic-based Loss Regularisers

Users tend to annotate documents collectively with semantically related tags. Two major semantic relations in user-generated tags are similarity and subsumption (Stock, 2010; Peters, 2009). To deal with this label correlation issue, we propose two loss regularisers jointly learned with the binary cross entropy loss function. The intuition is that the output values of the neural network s_d, having the dimensions as the label space $|T|$, should satisfy semantic relations among labels. Such relations can be inferred from the label sets or observed in external knowledge bases. The whole joint loss is defined as $L = L_{CE} + \lambda_1 L_{sim} + \lambda_2 L_{sub}$. L_{CE} is the binary cross entropy loss adopted for multi-label text classification (Nam et al., 2014). L_{sim} and L_{sub} are defined as:

$$\begin{aligned} L_{sim} &= \frac{1}{2} \sum_{d} \sum_{(j,k)|T_j,T_k \in y_d} Sim_{jk} |s_{dj} - s_{dk}|^2 \\ L_{sub} &= \frac{1}{2} \sum_{d} \sum_{(j,k)|T_j,T_k \in y_d} Sub_{jk} R(s_{dj})(1 - R(s_{dk})) \end{aligned} \tag{3}$$

where y_d is the label set (annotated tags) of the document d. T is a list of all labels, where j and k are the indices of the list T, corresponding to the indices of nodes s_{dj} and s_{dk} in the output layer s_d. $R()$ is the rounding function for binary prediction, $R(s_{dj}) = 0$ if $S_{dj} < 0.5$, otherwise $R(s_{dj}) = 1$.

The similarity matrix $Sim \in (0,1)^{|T|*|T|}$ indicates pairwise similarity between labels, the larger the value of Sim_{jk}, the more similar the labels T_j and T_k are. Each element Sub_{jk} in the subsumption matrix $Sub \in \{0,1\}^{|T|*|T|}$ indicates whether the label T_j is a child label of T_k. Both the Sim and Sub matrix can be inferred from the training data or from external knowledge bases before training. In implementation, Sim (if thresholded) and Sub can be treated as sparse matrix to reduce computational complexity. We also used an adapted version of the loss regularisers in mini-batch training (the same set of label pairs that co-occurred within all documents in the same batch) to further to reduce computational complexity.

The rationale is that the less the difference of the two outputs of the similar labels is, the lower the L_{sim}. On the contrary, for output values not reflecting the label similarity, i.e. large $|s_{dj} - s_{dk}|^2$ when Sim_{jk} is close to 1, the error will be penalised with higher L_{sim}.

Given a document and a subsumption pair of labels, if the child label is used for annotation, its parent label has a relatively higher chance being used as well. In L_{sub}, if a subsumption relation $< T_j \rightarrow T_k >$ presents in the label set y_d, the case that the parent label T_k is predicted as false, i.e. $R(s_{dk}) = 0$, when its child label T_j is predicted as true, i.e. $R(s_{dj}) = 1$, will be penalised. Such a case will result in a positive penalty, while the penalty will be 0 in all other cases.

Thus, L_{sim} constrains similar labels to have similar outputs, while L_{sub} reinforces each co-occurring subsumption pair to satisfy the dependency of the parent label on the child label.

3 Experiments

3.1 Datasets

We evaluate our proposed approach for automated social annotation on two representative open datasets in social tagging, Bibsonomy[1] (academic publication annotation) and Zhihu[2] (general domain social question annotation). For Bibsonomy, we used the cleaned dataset from (Dong et al.,

---

[1] `https://www.kde.cs.uni-kassel.de/bibsonomy/dumps`

[2] `https://biendata.com/competition/zhihu/`

Bibsonomy	Precision	Recall	F_1 Score	Time/Fold
Bi-GRU	.522±.020*	.217±.016*	.306±.019*	1480±92s
HAN	.572±.008*	.246±.012*	.344±.013*	1164±52s
JMAN-s-tg	.591±.010	.269±.006*	.370±.007*	1075±87s
JMAN-s-att	.586±.009	.269±.005*	.369±.006*	968±81s
JMAN-s	.586±.004	.282±.005	.380±.005	**894±55s**
JMAN	**.592±.009**	**.284±.006**	**.384±.007**	1044±73s

* Paired t-tests at 95 percent significance level against the JMAN model.

Table 1: Comparison Results on the Bibsonomy dataset

Zhihu	Precision	Recall	F_1 Score	Time/Fold
Bi-GRU	.238±.011*	.154±.009*	.187±.010*	1455±69s
HAN	.257±.012	.167±.010*	.203±.011*	1387±78s
JMAN-s-tg	.257±.005	.175±.003*	.208±.006**	1220±81s
JMAN-s-att	.254±.007**	.174±.005*	.207±.005*	1275±99s
JMAN-s	.257±.008	.177±.005	.210±.007	1147±44s
JMAN	**.260±.006**	**.179±.003**	**.212±.004**	**1135±52s**

* Paired t-tests at 95 percent significance level against the JMAN model.
** Paired t-tests at 90 percent significance level against the JMAN model.

Table 2: Comparison Results on the Zhihu dataset

2017) and further selected the tags related to Computer Sciences according to the ACM Computing Classification System[3] and selected the document that have both title and abstract (content); for Zhihu, we randomly sampled around 100,000 questions from the original data dump.

The cleaned Bibsonomy dataset has 12,101 documents, 17,619 vocabularies and 5,196 labels; the average number of labels per document is 11.59. The sample Zhihu dataset has 108,168 documents (questions), 62,519 vocabularies and 1,999 labels; the average number of labels per document is 2.45.

3.2 Implementation Details

To calculate Sim, we used cosine similarity, normalised to between 0 and 1, of self-trained skip-gram embedding (Mikolov et al., 2013) on all label sets in each dataset. To obtain Sub, about subsumption relations, for Bibsonomy, we resorted to an external knowledge source Microsoft Concept Graph[4] for label mapping and semantic grounding; for Zhihu, we used the provided crowd-sourced label subsumption relations. We tuned the λ_1 and λ_2 in L based on 10-fold cross-validation[5].

We implemented the proposed Joint Multi-label Attention Network (JMAN) model in Figure 1 on Tensorflow (Abadi et al., 2016) along with the baselines[6] based on brightmart's implementation[7] of TextRNN and HAN under the MIT license. Two strong baselines were chosen **Bi-GRU** (Schuster and Paliwal, 1997; Cho et al., 2014) and **HAN** (Yang et al., 2016; Hassan et al., 2018). Several variations of **JMAN** were also considered: (i) **JMAN-s**, the proposed model without semantic-based loss regularisers; (ii) **JMAN-s-tg**, the proposed model without semantic-based regularisers and title guided sentence-level attention, $c_d = [c_t, c_a]$; (iii) **JMAN-s-att**, the proposed model without semantic-based regularisers and the original sentence-level attention, $c_d = [c_t, c_{ta}]$.

We optimised the joint loss L using the Adam optimiser (Kingma and Ba, 2014) and set the number of hidden units as 100, learning rate as 0.01 and dropout rate as 0.5 (Srivastava et al., 2014) for all models. The batch sizes for Bibsonomy and Zhihu were set as 128 and 1,024, respectively. The sequence lengths of the title (also the length of each sentence) and the content were padded to 30 and 300 for Bibsonomy and 25 and 100 for Zhihu. Non-static input embedding for the title and the sentences were initialised as 100-dimension self-trained skip-gram embedding (Mikolov et al.,

---

[3] https://www.acm.org/publications/class-2012

[4] https://concept.research.microsoft.com/Home

[5] λ_1, λ_2 were tuned to 1e-4, 1e-1 for Bibsonomy and 1e-3, 1e-1 for Zhihu, respectively.

[6] Our code and datasets are available at https://github.com/acadTags/Automated-Social-Annotation.

[7] https://github.com/brightmart/text_classification

2013). We decayed the learning rate by half when the loss on validation set increased and set an early stopping point when learning rate is below 2e-5. All experiments were run on a GPU server, NVIDIA GeForce GTX 1080 Ti.

3.3 Results

We report the mean and the standard deviation of the testing results on models trained with 10-fold cross-validation. The cleaned user-generated tags, i.e. labels, for each dataset were taken as the ground truth and the widely used example-based metrics, Precision, Recall and F_1 score (Godbole and Sarawagi, 2004; Tsoumakas et al., 2010; Zhang and Zhou, 2014), were adopted. The average training time per fold was also recorded.

The results with respect to the two datasets are presented in the Table 1 and 2 respectively. Our proposed JMAN model significantly outperforms Bi-GRU and HAN. In terms of F_1, with the Bibsonomy dataset, the proposed JMAN model provides a 7.8% absolute increase (by 25.5%) over Bi-GRU and 4.0% (by 11.6%) over HAN; on the Zhihu dataset, our model is 2.5% absolutely (by 13.4%) better than Bi-GRU and 0.9% (by 4.4%) than HAN. This is mostly attributed to the boost of recall through modeling the title metadata and the title-guided attention mechanism. The JMAN model also converges ("understands") much faster than HAN with around 10.3% (for Bibsonomy) and 18.2% (for Zhihu) less training time per fold and converges even faster than Bi-GRU (by 29.5% and 22.0% for the Bibsonomy and Zhihu dataset in terms of training time, respectively). Recall and F_1 score drop significantly, with training time increased, when the title-guided or the original sentence-level attention is removed. Adding semantic-based loss regularisers further boosts the precision, recall and F_1 of the model.

We also noticed that, compared to the results on the Bibsonomy dataset, the improvement on the Zhihu dataset with the proposed model is less significant. This may be related to the characterstics of the dataset: Zhihu has shorter texts (padded to 1/3 of the Bibsonomy dataset), more vocabularies (over 3 folds), less number of labels (about 40%) and less average number of labels per document (about 1/5) than the Bibsonomy dataset. This would warrant further study on the datasets and on validating the model with datasets from other social media platforms.

4 Conclusion

We proposed a parallelled two-layer attention network for text annotation based on user-generated tags. It models the behaviour how human users read and understand document with the title-guided attention mechanism and leverages label semantics through two loss regularisers to constrain the network outputs. Experimental results show the effectiveness of this method with superior performance and training speed. This system can be applied to various types of social media platforms to support document organisation.

Future studies will explore the possibility of applying the title-guided attention mechanism to other large datasets on major social media platforms. It is also interesting to see whether the semantic-based loss regularisers can be adapted to improve the performance of the recent pre-trained transferable deep learning models, such as the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018).

Acknowledgments

We thank all the anonymous reviewers for their constructive feedback. The implementation is based on brightmart's TextRNN and Hierarchical Attention Network under the MIT license[8]. This research is funded by the Research Development Fund at Xi'an Jiaotong-Liverpool University, contract number RDF-14-01-10 and partially supported by the following: The National Natural Science Foundation of China under no. 61876155; The Natural Science Foundation of the Jiangsu Higher Education Institutions of China under no. 17KJD520010; Suzhou Science and Technology Program under no. SYG201712, SZS201613; Natural Science Foundation of Jiangsu Province BK20181189; Key Program Special Fund in XJTLU under no. KSF-A-01, KSF-P-02, and KSF-E-26.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016.

---

[8] https://github.com/brightmart/text_classification

Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA. USENIX Association.

Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. *BioNLP 2017*, pages 307–315.

Fabiano M Belém, Jussara M Almeida, and Marcos A Gonçalves. 2017. A survey on tag recommendation methods. *Journal of the Association for Information Science and Technology*, 68(4):830–844.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hang Dong, Wei Wang, and Coenen Frans. 2017. Deriving dynamic knowledge from academic social tagging data: a novel research direction. In *iConference 2017 Proceedings*, pages 661–666. iSchools.

Flavio Figueiredo, Henrique Pinto, Fabiano Belm, Jussara Almeida, Marcos Gonalves, David Fernandes, and Edleno Moura. 2013. Assessing the quality of textual features in social media. *Information Processing & Management*, 49(1):222 – 247.

Fatih Gedikli and Dietmar Jannach. 2014. Recommender systems, semantic-based. In *Encyclopedia of Social Network Analysis and Mining*, pages 1501–1510, New York, NY. Springer New York.

Eva Gibaja and Sebastián Ventura. 2015. A tutorial on multilabel learning. *ACM Computing Survey*, 47(3):52:1–52:38.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, pages 22–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hebatallah A. Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli. 2018. Semantic-based tag recommendation in scientific bookmarking systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 465–469, New York, NY, USA. ACM.

Paul Heymann and Hector Garcia-Molina. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 943–952.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Abhishek Kumar, Daisuke Kawahara, and Sadao Kurohashi. 2018. Knowledge-enriched two-layered attention network for sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 253–258.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label cooccurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526.

Yang Li, Ting Liu, Jing Jiang, and Liang Zhang. 2016. Hashtag recommendation with topical attention-based lstm. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3019–3029.

Marek Lipczak and Evangelos Milios. 2010. The impact of resource title on tags in collaborative tagging systems. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 179–188, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification — revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452, Berlin, Heidelberg. Springer Berlin Heidelberg.

Liqiang Nie, Yi-Liang Zhao, Xiangyu Wang, Jialie Shen, and Tat-Seng Chua. 2014. Learning to recommend descriptive tags for questions in social forums. *ACM Transactions on Information Systems (TOIS)*, 32(1):5:1–5:23.

Isabella Peters. 2009. Knowledge representation in Web 2.0: Folksonomies. In *Folksonomies. Indexing and Retrieval in Web 2.0*, Knowledge and Information, pages 153–282. De Gruyter.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Philipp Singer, Thomas Niebler, Andreas Hotho, and Markus Strohmaier. 2014. Folksonomies. In *Encyclopedia of Social Network Analysis and Mining*, pages 542–547, New York, NY. Springer New York.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wolfgang G. Stock. 2010. Concepts and semantic relations in information science. *Journal of the American Society for Information Science and Technology*, 61(10):1951–1969.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowl. Discovery Handbook*, pages 667–685, Boston, MA. Springer US.

Thomas Vander Wal. 2007. Folksonomy. `http://vanderwal.net/folksonomy.html`. [Online; accessed 7-March-2019].

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Multi-Channel Convolutional Neural Network for Twitter Emotion and Sentiment Recognition

Jumayel Islam[1], **Robert E. Mercer**[1], **Lu Xiao**[2]
[1]Department of Computer Science, The University of Western Ontario
[2]School of Information Studies, Syracuse University
`jislam3@uwo.ca, mercer@csd.uwo.ca, lxiao04@syr.edu`

Abstract

The advent of micro-blogging sites has paved the way for researchers to collect and analyze huge volumes of data in recent years. Twitter, being one of the leading social networking sites worldwide, provides a great opportunity to its users for expressing their states of mind via short messages which are called tweets. The urgency of identifying emotions and sentiments conveyed through tweets has led to several research works. It provides a great way to understand human psychology and impose a challenge to researchers to analyze their content easily. In this paper, we propose a novel use of a multi-channel convolutional neural architecture which can effectively use different emotion and sentiment indicators such as hashtags, emoticons and emojis that are present in the tweets and improve the performance of emotion and sentiment identification. We also investigate the incorporation of different lexical features in the neural network model and its effect on the emotion and sentiment identification task. We analyze our model on some standard datasets and compare its effectiveness with existing techniques.

1 Introduction

Social networking sites (e.g., Twitter) have become immensely popular in the last decade. User generated content (e.g., blog posts, statuses, tweets etc.) in social media provides a wide range of opinionated, emotional and sentimental content which gives researchers a massive data source to explore. For example, Twitter, being one of the leading social networking giants, provides an online environment that allows people of various backgrounds and locations to share their opinions and views on different matters. As of July 2018, over 500 million tweets are sent per day having over 300 million monthly active users.[1]

There is often a misconception about sentiments and emotions as these subjectivity terms have been used interchangeably (Munezero et al., 2014). Munezero et al. (2014) differentiate these two terms along with other subjectivity terms and provide the computational linguistics community with clear concepts for effective analysis of text. While sentiment classification tasks deal with the polarity of a given text (whether a piece of text expresses positive, negative or neutral sentiment) and the intensity of the sentiment expressed, emotion mining tasks naturally deal with human emotions which in some end purposes are more desirable (Ren and Quan, 2012; Desmet and Hoste, 2013; Mohammad et al., 2015). Detecting emotion and sentiment from noisy twitter data is really challenging due to its nature. Tweets tend to be short in length and have a diverse vocabulary making them harder to analyze due to the limited contextual information they contain. In this study, we are interested in tackling these two tasks with a novel use of a single neural network architecture.

A number of emotion theories are available which suggest different sets of basic emotions. Interestingly, *joy, sadness, anger, fear* and *surprise* are common to all. To the best of our knowledge, the model suggested by Ekman (1999) is the most broadly used emotion model. In this study, we use Ekman's basic emotions together with other sets of emotions (Plutchik, 1984; Shaver et al., 1987).

In early textual emotion mining and sentiment analysis research, the usefulness of using external lexicons along with predefined rules has been demonstrated (Aman and Szpakowicz, 2008; Neviarouskaya et al., 2007; Bandhakavi et al., 2017; Thelwall et al., 2010; Gilbert, 2014). Aman and Szpakowicz (2008) used *Roget's Thesaurus* along with *WordNet-Affect* for fine-grained emotion prediction from blog data. Bandhakavi et al. (2017) propose a unigram mixture model (UMM)

---

[1] http://www.internetlivestats.com/twitter-statistics/

Proceedings of NAACL-HLT 2019, pages 1355–1365
Minneapolis, Minnesota, June 2 - June 7, 2019.

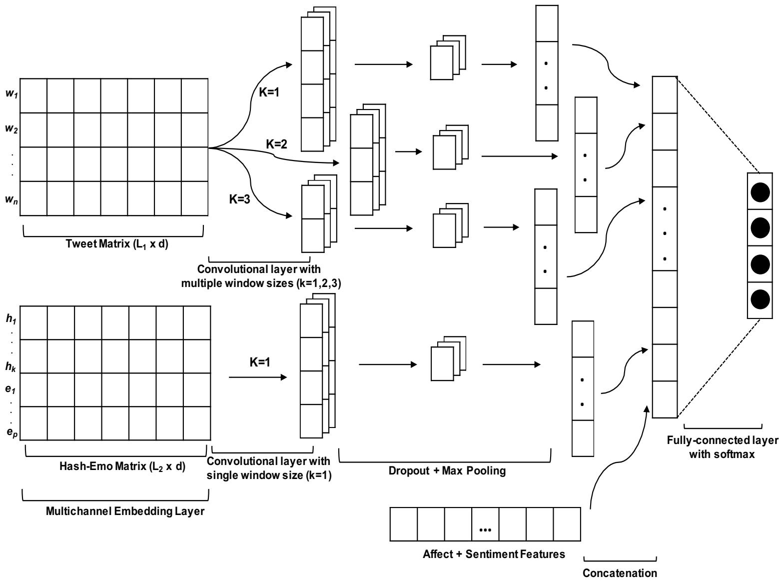


Figure 1: Overview of the MC-CNN model

to create a domain-specific lexicon which performs better in extracting features than Point-wise Mutual Information and supervised Latent Dirichlet Allocation methods. Neviarouskaya et al. (2007) propose a rule-based system which can handle informal texts in particular. They built a database of abbreviations, emoticons, affect words, etc., in which each entry is labeled with an emotion and its intensity. Thelwall et al. (2010) propose an algorithm, *SentiStrength*, which utilizes a dictionary of sentiment words associated with strength measures to deal with short informal texts from social media. Gilbert (2014) propose *VADER*, a rule-based model for sentiment analysis. They built a lexicon which is specially attuned to microblog-like contexts and their model outperforms individual human raters. More recently, deep learning models have proven to be very successful when applied on various text-related tasks (Kim, 2014; Kalchbrenner et al., 2014; dos Santos and Gatti, 2014; Tai et al., 2015; Wang et al., 2016; Felbo et al., 2017; Abdul-Mageed and Ungar, 2017). Kim (2014) showed the effectiveness of a simple CNN model that leverages pre-trained word vectors for a sentence classification task. Kalchbrenner et al. (2014) propose a dynamic CNN model using a dynamic k-max pooling mechanism which is able to generate a feature graph which captures a variety of word relations. They showed the efficacy of their model by achieving high performances on binary and multi-class sentiment classification tasks without any feature engineering. dos Santos and Gatti (2014) propose a deep CNN model that uses both character and word-level information allowing them to achieve state-of-the-art performance on both binary and fine-grained multi-class sentiment classification for one of the twitter datasets. Tai et al. (2015) propose a Tree-LSTM model which captures syntactic properties in text. Their model performs particularly well on sentiment classification. Wang et al. (2016) propose a regional CNN-LSTM model for dimensional sentiment analysis. Their proposed model computes *valence-arousal* ratings from texts and outperforms several regression-based methods. Felbo et al. (2017) propose a bi-directional LSTM model with attention showing that their model learns better representations when distant supervision is expanded to a set of noisy labels. Abdul-Mageed and Ungar (2017) also used distant supervision to build a large twitter dataset and proposed a Gated Recurrent Neural Network model for fine-grained emotion detection.

The recent success of neural based models motivated us to take a different look at the sentiment and emotion prediction from the noisy twitter data task. Compared with sequential models, CNN models train relatively faster and seem to work very well on noisy data such as tweets which are grammatically error-prone. We decided to work with CNN models after our initial experiments suggested that they perform comparatively better than a simple BiLSTM model on twitter dataset. We address the following questions in this paper:

- Can CNN models be used in a way that can improve the performance of detecting emotion and sentiment from noisy Twitter data?
- How important are hashtag words, emoticons and emojis as predictors of emotion and sentiment in micro-blogging sites? How can we encode them in a multi-channel convolutional neural network?
- How can we add external features to a CNN model effectively?

The remainder of the paper is organized as follows: We describe our model architecture in detail in Section 2. In Section 3, we describe the datasets and lexicons used in our experiments. As well, we describe the experimental setup required for working with Twitter data. In Section 4, we discuss the results from our experiments. In Section 5, we discuss our findings with particular attention paid to answering the above questions. Finally, in Section 6, we give a summary of our work followed by our remarks on future studies.

2 Multi-channel CNN Model

We represent the architecture of our model in Fig. 1. The model consists of an embedding layer with two channels, a convolution layer with different kernel sizes and multiple filters, a dropout layer for regularization, a max pooling layer, multiple hidden layers and a softmax layer. We now describe each of these layers in detail.

2.1 Embedding Layer

In this layer, two embedding matrices, the Tweet Matrix and the Hash-Emo Matrix, are passed through two different channels of our convolutional neural network. The first matrix represents a particular tweet. Each tweet t_i consists of a sequence of tokens $w_1, w_2, \ldots, w_{n_i}$. L_1 is the maximum tweet length. The height of the Tweet Matrix is L_1. Short tweets are padded using zero padding.

In the Tweet Matrix, every word is represented as a d-dimensional word vector. Since tweets are usually noisy, short in length, and have different kinds of features other than text, it's useful to have a word embedding specially trained on a large amount of Tweet data (Tang et al., 2014). Previous research (Collobert et al., 2011; Socher et al., 2011) has shown the usefulness of using pre-trained word vectors to improve the performance of various models. As a result, in our experiments, we have used the publicly available pre-trained `GloVe` word vectors for Twitter by (Pennington et al., 2014a). The word vectors are trained on 27B word tokens in an unsupervised manner.

In this layer, we also pass another matrix called the Hash-Emo Matrix through a different channel in our network. This matrix is composed of three different sets of features: hashtags, emoticons and emojis. These are considered as distinguishable traits to showcase one's mood (Zhao et al., 2012). People like to use hashtags to express their emotional state through various micro-blogging sites (e.g., Twitter) (Qadir and Riloff, 2014). Also graphical emoticons or emojis can convey strong emotion or sentiment. So for each tweet t_i, we extract hashtags $h_1, h_2, \ldots, h_{k_i}$ and emoticons/emojis $e_1, e_2, \ldots, e_{p_i}$. We concatenate the hashtags and emoticon/emoji vectors for each tweet t_i to get the Hash-Emo Matrix. We introduce a hyper-parameter L_2 as a threshold on the height of the Hash-Emo Matrix. Tweets with the number of hash-emo features less than L_2 are padded with zero while tweets with more hash-emo features than L_2 are truncated. We use word vectors from `GloVe` with dimension d for hashtags words. In the case that no word vector is found for a particular word we randomly initialize it. We also do random initialization of word vectors for emoticons. For emojis, we first map it to something descriptive (to be discussed in more detail in Section 3.2) and then generate random word vectors. These word vectors are tuned during the training phase.

2.2 Convolutional Layer

In this layer, we apply m filters of varying window sizes over the Tweet Matrix from the embedding layer. Here, window size (k) refers to the number of adjacent word vectors in the Tweet Matrix that are filtered together (when $k > 1$). Sliding our filter down we repeat this for the rest of the word vectors. Let $w_i \in \mathbb{R}^d$ be the d-dimensional word vector corresponding to the i-th word in a tweet. Also let $w_{i:i+j}$ denote the concatenation of word vectors $w_i, w_{i+1}, \ldots, w_{i+j}$ and $F \in \mathbb{R}^{k \times d}$ denote the filter matrix. Thus a feature f_i is generated by:

$$f_i = F \otimes w_{i:i+k-1} + b \tag{1}$$

where b is a bias term and $\otimes$ represents the convolution action (a sum over element-wise multiplications). At this stage, we apply a nonlinear activation function such as *ReLU* (Nair and Hinton, 2010) before passing it through the dropout layer. We use multiple filters with the same window size in order to learn complementary features from the same window. Different window sizes (k) allow us to extract active local k-gram features.

For the Hash-Emo Matrix, we apply m filters to each vector to generate local unigram features in different scales before passing it to the next layer.

2.3 Pooling Layer

In this layer, employing a max-over pooling operation (Collobert et al., 2011) on the output from the previous layer for each channel extracts the most salient features. In this way, for each filter, we get the maximum value. So we get features equal to the number of filters in this stage. We chose max pooling instead of other pooling schemes because Zhang and Wallace (2017) showed that max pooling consistently performs better than other pooling strategies for various sentence classification tasks.

2.4 Hidden Layers

We concatenate all the feature vectors from the previous layer. In addition, we concatenate additional sentiment and affect feature vectors (which are described in detail in Section 3.2) as well which forms a large feature vector. This is then passed through a number of hidden layers. A non-linear activation function (i.e., ReLU (Nair and Hinton, 2010)) is applied in each layer before the vector is finally passed through the output layer. We tried a different activation function (tanh) as well, but ReLU worked the best for us.

2.5 Output Layer

This is a fully connected layer which maps the inputs to a number of outputs corresponding to the number of classes we have. For multi-class classification task, we use softmax as the activation function and categorical cross-entropy as the loss function. The output of the softmax function is equivalent to a categorical probability distribution which generally indicates the probability that any of the classes are true. For binary classification task, we use sigmoid as the activation function and binary cross-entropy as our loss function.

3 Experiments

In this section, we describe in detail the datasets and experimental procedures used in our study.

Emotion	**Dataset**			
	BTD	TEC	CBET	SE
joy	409,983	8,240	10,691	3,011
sadness	351,963	3,830	8,623	2,905
anger	311,851	1,555	9,023	3,091
love	175,077	–	9,398	–
thankfulness	80,291	–	8,544	–
fear	76,580	2,816	9,021	3,627
surprise	14,141	3,849	8,552	–
guilt	–	–	8,540	–
disgust	–	761	8,545	–
Total	1419,886	21,051	80,937	12,634

(a)

Dataset	**#Positive**	**#Negative**	**#Neutral**
STS-Gold	632	1,402	–
STS-Test	182	177	139
SS-Twitter	1,252	1,037	1,953

(b)

Table 1: (a) Basic statistics of the emotion datasets used in our experiments. (b) Basic statistics of sentiment labeled datasets used in our experiments.

3.1 Datasets

We used a number of emotion and sentiment datasets for our experiments. A description of each dataset is given below:

BTD. Big Twitter Data is an emotion-labeled Twitter dataset provided by Wang et al. (2012). The dataset had been automatically annotated based on the seven emotion category seed words (Shaver et al., 1987) being a hashtag and the quality of the data was verified by two annotators as described in (Wang et al., 2012). We were only able to retrieve a portion of the original dataset as many tweets were either removed or not available at the time we fetched the data using the Twitter API. We applied the heuristics from (Wang et al., 2012) to remove any hashtags from the tweets which belong to the list of emotion seed words.

TEC. Twitter Emotion Corpus has been published by Mohammad (2012) for research purposes. About 21,000 tweets were collected based on hashtags corresponding to Ekman's (1999) six basic emotions. The dataset has been used in related works (Shahraki and Zaiane, 2017; Balahur, 2013; Mohammad and Kiritchenko, 2015).

CBET. The Cleaned Balanced Emotional Tweet dataset is provided by Shahraki and Zaiane (2017). To the best of our knowledge, this is one of the largest publically available balanced datasets for twitter emotion detection research. The dataset contains 80,937 tweets with nine emotion categories including Ekman's six basic emotions.

SE. The SemEval-2018 Task 1 - Affect dataset was provided by Mohammad et al. (2018). The SemEval task was to estimate the intensity of a given tweet and its corresponding emotion. However, in this study, we utilize the labeled dataset only to classify the tweets into four emotion categories and use the training, development and test sets provided in this dataset in our experiments.

STS-Gold. This dataset was constructed by Saif et al. (2013) for Twitter sentiment analysis. The dataset contains a total of 2,034 tweets labeled (positive/negative) by three annotators. This dataset has been extensively used in several works for model evaluation (Saif et al., 2014b; Krouska et al., 2017; Saif et al., 2014a).

STS. The Stanford Twitter Sentiment dataset was introduced by Go et al. (2009). It consists of a training set and a test set. The training set contains around 1.6 million tweets, whereas the test set contains 498 tweets. The training set was built automatically based on several emoticons as potential identifiers of sentiment. However, the test set was manually annotated and heavily used for model evaluation in related research. We perform one experiment with all three labels (positive/negative/neutral) to compare the performance of different variants of our model and another one with two labels (positive/negative) to make comparison with related works (Jianqiang et al., 2018; dos Santos and Gatti, 2014; Go et al., 2009).

SS-Twitter. The Sentiment Strength Twitter dataset was constructed by Thelwall et al. (2012) to evaluate SentiStrength. The tweets were manually labeled by multiple persons. Each tweet is assigned a number between 1 and 5 for both positive and negative sentiments, 1 represents weak sentiment strength and 5 represents strong sentiment strength. We followed the heuristics used by Saif et al. (2013) to obtain a single sentiment label for each tweet, giving us a total of $4,242$ positive, negative and neutral tweets. The transformed dataset has been used in other literature (Go et al., 2009; Zhang et al., 2018).

We provide basic statistics of the datasets used in our experiments in Table 1.

3.2 Experimental Setup

Data Cleaning. Twitter data is unstructured and highly informal (Yoon et al., 2013) and thus it requires a great deal of effort to make it suitable for any model. NLTK (Bird and Loper, 2004) provides a regular-expression based tokenizer for Twitter, TweetTokenizer, which preserves user mentions, hashtags, urls, emoticons and emojis in particular. It also reduces the length of repeated characters to three (i.e. "Haaaaaapy" will become "Haaapy")". In our experiments, we utilized the TweetTokenizer to tokenize tweets.

To accommodate the pretrained word vectors from (Pennington et al., 2014b), we pre-processed each tweet in a number of ways. We lowercased all the letters in the tweet. User mentions have been replaced with <user> token (i.e. @username1 will become <user>). In addition, we also removed urls from the tweets as urls do not provide any emotional value. We also normalized certain negative words (e.g., "won't" will become "will not"). Using slang words is a very common practice in social media. We compiled a list of the most common slang words from various online resources[2] and replaced all of the occurrences with their full form (e.g., "nvm" will become "never mind"). Our list of slang words doesn't contain any word which has multiple meanings. Usage of certain punctuation is often crucial in social media posts as it helps the user to emphasize certain things. We found that two punctuation symbols (! and ?) are common among social media users to express certain emotional states. We kept these symbols in our text and normalized the repetitions (e.g., "!!!" will become "! <repeat>").

The use of emojis and emoticons has increased significantly with the advent of various social media sites. Emoticons (e.g., :-D) are essentially a combination of punctuation marks, letters and numbers used to create pictorial icons which generally display an emotion or sentiment. On the other hand, emojis are pictographs of faces, objects and symbols. The primary purpose of using emojis and emoticons is to convey certain emotions and sentiments (Dresner and Herring, 2010). One advantage of using the TweetTokenizer is that it gives us emoticons and emojis as tokens. Though we use the emoticons as is in our experiment, we utilize a python library called "emoji" to get descriptive details about the pictorial image. For example, "☺" represents *"smiling_face"*.

In our experiments, we removed stop-words and replaced numbers occurring in the tweets with the token <number>. We also stripped off "#" symbols from all the hashtags within the tweets (e.g.,

---

[2] Example: https://slangit.com/terms/social_media

Emotion	**Dataset**											
	BTD			TEC			CBET			SemEval		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
joy	68.4	77.4	72.6	67.4	77.1	71.8	58.1	56.1	57.1	78.5	70.1	74.1
sadness	72.7	74.5	73.6	48.8	53.7	50.9	38.0	43.3	40.5	62.6	41.0	49.6
anger	74.7	79.1	76.8	34.5	23.8	27.7	49.3	52.1	50.7	59.7	63.6	61.6
love	57.0	46.4	51.1	–	–	–	65.4	53.3	58.7	–	–	–
thankfulness	63.2	55.3	59.0	–	–	–	66.1	68.0	67.0	–	–	–
fear	57.6	38.3	46.0	61.5	57.2	58.6	70.3	69.6	70.0	51.6	71.9	60.1
surprise	88.1	16.1	27.1	55.9	50.2	52.5	51.0	55.3	53.0	–	–	–
guilt	–	–	–	–	–	–	53.8	49.6	51.6	–	–	–
disgust	–	–	–	67.4	77.1	71.8	59.3	61.0	60.2	–	–	–
Avg.	**68.9**	**55.3**	**58.0**	**55.9**	**56.5**	**55.6**	**56.8**	**56.5**	**56.5**	**63.1**	**61.7**	**61.3**

Table 2: Results (in %) of our model (MC-CNN) for four emotion-labeled datasets.

"#depressed" will become "depressed") and used the stripped version of hashtags on the second channel of our model. We only kept tokens with more than one character.

Input Features. Along with word embeddings, we used additional affect and sentiment features in our network. In our experiments, we used a feature vector V_f where each value in the vector corresponds to a particular lexical feature ranging between $[0, 1]$. We utilized a number of publicly available lexicons which are described briefly below to construct the vector.

Warriner et al. (2013) provides a lexicon consisting of 14,000 English lemmas with valence, arousal and dominance scores. Three components of emotion are scored for each word between 1 and 9 in this lexicon. We calculate the average score for each component across all tokens in a tweet and normalize them in the range [0, 1]. Gilbert (2014) provides a list of lexical features along with their associated sentiment intensity measures. We use this lexicon to calculate the average of positive, negative, and neutral scores over all the tokens in a tweet. In addition, we used the *NRC Emotion Lexicon* provided by Mohammad and Turney (2013) which consists of a list of unigrams and their association with one of the emotion categories (anger, anticipation, disgust, fear, joy, sadness, surprise, trust). We use the percentage of tokens belonging to each emotion category as features. We also used the *NRC Affect Intensity Lexicon* provided by Mohammad and Bravo-Marquez (2017) and *NRC Hashtag Emotion Lexicon* provided by Mohammad and Kiritchenko (2015) which contain real-valued fine-grained word-emotion association scores for words and hashtag words.

We combined two lexicons *MPQA* and *BingLiu* provided by Wilson et al. (2005) and Hu and Liu (2004), respectively, and used them to calculate the percentage of positive and negative tokens belonging to each tweet. We also used *AFINN* (Nielsen, 2011) which contains a list of English words rated for valence with an integer between −5 (negative) to +5 (positive). We first normalized the scores in the range [0,1] and then calculated the average of this score over all the tokens in a tweet. Lastly, we detect the presence of consecutive exclamation (!) and question marks (?) in a tweet and use them as boolean features.

Network Parameters and Training. Zhang and Wallace (2017) performed a sensitivity analysis on various parameters of a one-layer CNN model and showed how tuning the parameters can affect the performance of a model. Inspired by the work done by (Zhang and Wallace, 2017), we also searched for the optimal parameter configurations in our network. Table 3 shows different hyper-parameter configurations that we tried and the final configuration that was used in our model. The final configuration was based on both performance and training time. The embedding dimension has been set to 100 for both channels of our network as it worked best for us among other dimensions. We also experimented with a different number of fil-

Hyper-parameter	**Ranges**	**Selected**
Embedding dimension	50/100/200	100
Number of filters	64/128/256	128
Kernel sizes	1/2/3/4/5	1/2/3
Batch size	16/30/50	16
Epochs	10/20	10
Dropout rate	0.1/0.2/0.5	0.5
Learning rate	0.015/0.001/0.01	0.001

Table 3: Ranges of different hyper-parameters searched during tuning and the final configurations selected for our experiments

Datasets	Methods	Positive			Negative			Average			Accuracy
		P	R	F1	P	R	F1	P	R	F1	
STS-Gold	A	70.5	74.1	72.2	88.0	86.0	87.0	79.3	88.0	79.6	82.3
	B	–	–	–	–	–	–	79.5	77.9	78.6	82.1
	C	–	–	–	–	–	–	–	–	77.5	80.3
	D	75.4	74.9	75.1	90.2	90.3	90.2	82.8	82.6	82.7	86.0
	Ours	87.9	82.0	84.5	92.1	94.6	93.3	90.0	88.3	88.9	**90.7**
STS-Test	D	88.0	89.5	88.7	87.2	85.4	86.3	87.6	87.4	87.5	87.6
	E	–	–	–	–	–	–	–	–	–	86.4
	F	–	–	–	–	–	–	–	–	–	83.0
	Ours	90.2	91.2	90.5	91.3	89.3	89.9	90.8	90.3	90.2	**90.3**
SS-Twitter	G	–	–	–	–	–	–	67.8	52.7	59.3	61.9
	F	–	–	76.6	–	–	69.2	–	–	72.9	73.4
	Ours	81.3	84.7	82.0	72.5	72.7	72.2	76.9	78.7	77.1	**79.3**

Table 4: Results (in %) of our model (MC-CNN) from 10-fold cross-validation compared against other methods for sentiment labeled datasets (2-class). Bold text indicates the best performance in a column. **A:** Thelwall-Lexicon (Updated + Expanded) (Saif et al., 2014b). **B:** SentiStrength (Krouska et al., 2017). **C:** SentiCircle with Pivot (Saif et al., 2014a). **D:** Deep Convolutional Neural Network (Jianqiang et al., 2018). **E:** Character to Sentence Convolutional Neural Network (CharSCNN) (dos Santos and Gatti, 2014). **F:** Maximum Entropy (Saif et al., 2013). **G:** Quantum Language Model + Quantum Relative Entropy (Zhang et al., 2018).

ters and varying kernel sizes. The combination of kernel sizes, $(k = 1, 2, 3)$ in the first channel and $k = 1$ in the second channel worked the best for us. We also experimented with various batch sizes and the performance of the model remained reasonably constant, though the training time varied significantly. In our network, we used three hidden layers. In addition, we used the Adam optimizer (Kingma and Ba, 2014) and the back-propagation (Rumelhart et al., 1986) algorithm for training our model. Keras 2.2.0 was used for implementing the model.

Regularization. In order to reduce overfitting, it is a common practice to employ regularization strategies in CNNs. In our experiments, we used dropout regularization (Srivastava et al., 2014) for both of the channels after the convolutional layer. We experimented with three different dropout rates as seen in Table 3 and also with no dropout at all. The model works better when we apply dropouts after the convolutional layer.

4 Results

In this section, we describe the results obtained in our experiments. We use precision, recall, F1-score and accuracy as our evaluation metrics.

In recent emotion category recognition studies on Twitter data, people tend to construct their own dataset by collecting tweets from Twitter for their experiments. Hence, it is hard to find a large enough benchmark dataset to compare the performance with other people's work. In this study, we experimented with four emotion labeled datasets which have been made publicly available by their authors. Table 2 shows the results for each emotion category for all of the datasets. For the BTD dataset, we trained our model with $1,136,305$ tweets, while we used $140,979$ and $142,602$ tweets as development and test data respectively. We used the same training, development and test sets as (Wang et al., 2012) except that our retrieved dataset contains fewer samples. We achieved relatively high F1-scores of $\mathbf{72.6\%}, \mathbf{73.6\%}$ and $\mathbf{76.8\%}$ for *joy, sadness* and *anger*, respectively, whereas for *surprise* we get a low F1-score of $\mathbf{27.1\%}$. This is probably due to the imbalanced nature of the dataset as can be seen in Table 1. The number of samples for *joy, sadness* and *anger* is much higher than for *surprise*. Our model achieves an accuracy of $\mathbf{69.2\%}$, whereas Wang et al. (2012) reported an accuracy of $\mathbf{65.6\%}$ when trained on a much larger dataset. We can not make direct comparison with (Wang et al., 2012) since we were not able to retrieve the full test set due to the unavailability of some tweets at the time of fetching data from Twitter. For the TEC dataset, we evaluated our model with 10-fold cross validation. Mohammad (2012) reported an F1-score of $\mathbf{49.9\%}$ with SVM, whereas our model achieves an F1-score of $\mathbf{55.6\%}$. For the CBET dataset, we used 80% of the data as the training set and the remaining 20% as the test set. We get an average F1-score of $\mathbf{56.5\%}$. We also used 10-fold cross-validation for the SemEval dataset and achieved an F1-score of $\mathbf{61.3\%}$. Table 4 shows the performance of our model with 10-fold cross-

Datasets	Methods	Positive			Negative			Neutral			Accuracy
		P	R	F1	P	R	F1	P	R	F1	
STS-Test	CNN	73.6	84.1	76.8	73.5	74.2	72.8	74.2	64.3	65.8	75.1
	MC-CNN†	63.1	83.4	70.3	76.5	70.4	72.8	71.6	53.9	60.8	70.6
	MC-CNN†‡	**80.3**	**83.8**	**81.4**	**87.5**	**81.2**	**83.5**	**79.2**	**77.9**	**77.7**	**81.5**
SS-Twitter	CNN	48.9	51.7	49.3	43.9	53.7	47.4	67.8	**66.8**	64.3	59.1
	MC-CNN†	61.6	62.0	61.4	55.3	65.5	59.7	71.6	62.7	66.4	63.2
	MC-CNN†‡	**65.1**	**65.4**	**64.0**	**56.2**	**65.7**	**60.0**	**72.2**	62.7	**66.7**	**64.6**

Table 5: Results (in %) of three variants of our model from 10-fold cross-validation for sentiment labeled datasets (3-class). Bold text indicates the best performance in a column.† represents the inclusion of Hash-Emo embedding into the network. ‡ represents the inclusion of external features into the network.

validation on different sentiment datasets with two classes (positive and negative). For the STS-Gold dataset, our model achieves an accuracy of $\mathbf{90.7\%}$ whereas the previous best accuracy ($\mathbf{86.0\%}$) was reported by Jianqiang et al. (2018) with a deep CNN model. Our model achieves the best accuracy ($\mathbf{90.3\%}$) for the STS-Test dataset as well, while the previous best ($\mathbf{87.6\%}$) was reported in (Jianqiang et al., 2018). dos Santos and Gatti (2014) also experimented with the same dataset with their Character to Sentence CNN model, reporting an accuracy of $\mathbf{86.4\%}$. Lastly, for the SS-Twitter dataset, our model achieves an accuracy of $\mathbf{79.3\%}$ whereas Zhang et al. (2018) and Saif et al. (2013) reported an accuracy of $\mathbf{61.9\%}$ and $\mathbf{73.4\%}$, respectively.

Tables 5 and 6 show the performance of three variants of our model on the sentiment labeled datasets and the emotion labeled datasets respectively. The first variant is a basic CNN model without hash-emo embedding or any additional features. The second variant includes the hash-emo embedding, while the last variant combines additional lexical features as well. It can be observed that when we introduce the second channel with hash-emo embedding, we get a significant increase in accuracy for most of the datasets. We can see in Table 5 that, for STS-Test and SS-Twitter datasets, we get better F1-scores for all three sentiment labels when we include the hash-emo embedding along with external lexical features. In Table 6, we can see that, inclusion of hash-emo embedding in the network gives us $\mathbf{2.4}, \mathbf{3.3}, \mathbf{2.3}$ and $\mathbf{3.5}$ percentage points increase in accuracy and the inclusion of additional features as well gives us $\mathbf{3.1}, \mathbf{4.6}, \mathbf{2.6}$ and $\mathbf{5.7}$ percentage points increase in accuracy for BTD, TEC, CBET and SE datasets, respectively, over the base models.

Models	Dataset			
	BTD	*TEC*	*CBET*	*SE*
CNN	66.1	54.3	53.8	56.3
MC-CNN†	68.5	57.6	56.1	59.8
MC-CNN†‡	69.2	58.9	56.4	62.0

Table 6: Comparison of results (accuracy in %) of three variants of our model. † represents the inclusion of Hash-Emo embedding into the network. ‡ represents the inclusion of external features into the network.

5 Discussion

In this study, we have showed the effectiveness of encoding hashtags, emoticons and emojis through a separate channel in a CNN network for emotion and sentiment detection tasks. Our MC-CNN model with hash-emo embedding performs well when compared to the basic CNN model. To the best of our knowledge, our model achieves the best accuracies on the three sentiment datasets, and has significant improvement in performance on the four emotion labeled datasets over the basic CNN model. The results show the importance of hashtags, emoticons and emojis in social media as predictors of emotion and sentiment. The model performs even better when additional lexical features are introduced into the network.

6 Conclusions and Future Work

In this paper, we propose a novel use of a multi-channel convolutional neural architecture which effectively encodes different types of emotion indicators which are found in social media posts. Results suggest that encoding the emotion indicators through a separate channel provides significant improvement over the traditional CNN based models. We also demonstrate a simple approach to incorporate different lexical features in the network giving us comparatively better results when used along with our MC-CNN model. Our model performs particularly well on two important tasks in social media: emotion detection and sentiment analysis. This model can be extended to perform other tasks as well. In future, we would like to explore character embedding as this can give us crucial linguistic features from noisy twitter data.

Acknowledgements

We are thankful for the support by The Social Sciences and Humanities Research Council of Canada (SSHRC) through our Partnership Development Grant titled “Living Archives of Rwandan Exiles and Survivors in Canada”.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 718–728.

Saima Aman and Stan Szpakowicz. 2008. Using roget’s thesaurus for fine-grained emotion recognition. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Alexandra Balahur. 2013. Sentiment analysis in social media texts. In *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 120–128.

Anil Bandhakavi, Nirmalie Wiratunga, Deepak Padmanabhan, and Stewart Massie. 2017. Lexicon based feature extraction for emotion text classification. *Pattern recognition letters*, 93:133–142.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Bart Desmet and VéRonique Hoste. 2013. Emotion detection in suicide notes. *Expert Systems with Applications*, 40(16):6351–6358.

Eli Dresner and Susan C Herring. 2010. Functions of the nonverbal in cmc: Emoticons and illocutionary force. *Communication theory*, 20(3):249–268.

Paul Ekman. 1999. Basic emotions. *Handbook of cognition and emotion*, pages 45–60.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625. Association for Computational Linguistics.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Zhao Jianqiang, Gui Xiaolin, and Zhang Xuejun. 2018. Deep convolution neural networks for twitter sentiment analysis. *IEEE Access*, 6:23253–23260.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Akrivi Krouska, Christos Troussas, and Maria Virvou. 2017. Comparative evaluation of algorithms for sentiment analysis over social networking services. *J Univers Comput Sci*, 23(8):755–768.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 65–77. Association for Computational Linguistics.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17.

Saif M Mohammad. 2012. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Saif M Mohammad, Xiaodan Zhu, Svetlana Kiritchenko, and Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management*, 51(4):480–499.

Myriam D Munezero, Calkin Suero Montero, Erkki Sutinen, and John Pajunen. 2014. Are they different? affect, feeling, emotion, sentiment, and opinion detection in text. *IEEE transactions on affective computing*, 5(2):101–111.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. 2007. Textual affect sensing for sociable and expressive online communication. In *International Conference on Affective Computing and Intelligent Interaction*, pages 218–229. Springer.

Finn Årup Nielsen. 2011. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages, Heraklion, Crete, Greece, May 30, 2011*, pages 93–98.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014a. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014b. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robert Plutchik. 1984. Emotions: A general psychoevolutionary theory. *Approaches to emotion*, 1984:197–219.

Ashequl Qadir and Ellen Riloff. 2014. Learning emotion indicators from tweets: Hashtags, hashtag patterns, and phrases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1203–1209. Association for Computational Linguistics.

Fuji Ren and Changqin Quan. 2012. Linguistic-based emotion analysis and recognition for measuring consumer satisfaction: an application of affective computing. *Information Technology and Management*, 13(4):321–332.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2013. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. *first ESSEM workshop*.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2014a. Senticircles for contextual and conceptual semantic sentiment analysis of twitter. In *European Semantic Web Conference*, pages 83–98. Springer.

Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. 2014b. Adapting sentiment lexicons using contextual semantics for sentiment analysis of twitter. In *European Semantic Web Conference*, pages 54–63. Springer.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Ameneh Gholipour Shahraki and Osmar R Zaiane. 2017. Lexical and learning-based emotion mining from text. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*.

Phillip Shaver, Judith Schwartz, Donald Kirson, and Cary O'connor. 1987. Emotion knowledge: Further exploration of a prototype approach. *Journal of personality and social psychology*, 52(6):1061.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 1555–1565. Association for Computational Linguistics.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 225–230.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter "big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 587–592. IEEE.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Sunmoo Yoon, Noémie Elhadad, and Suzanne Bakken. 2013. A practical approach for content mining of tweets. *American journal of preventive medicine*, 45(1):122–129.

Yazhou Zhang, Dawei Song, Xiang Li, and Peng Zhang. 2018. Unsupervised sentiment analysis of twitter posts using density matrix representation. In *European Conference on Information Retrieval*, pages 316–329. Springer.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263. Asian Federation of Natural Language Processing.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1528–1531. ACM.

Detecting Cybersecurity Events from Noisy Short Text

Semih Yagcioglu, Mehmet Saygin Seyfioglu[*], Begum Citamak, Batuhan Bardak
Seren Guldamlasioglu, Azmi Yuksel, Emin Islam Tatli
STM A.Ş., Ankara, Turkey
{syagcioglu, msaygin.seyfioglu, begum.citamak, batuhan.bardak,
sguldamlasioglu, azyuksel, emin.tatli}@stm.com.tr

Abstract

It is very critical to analyze messages shared over social networks for cyber threat intelligence and cyber-crime prevention. In this study, we propose a method that leverages both domain-specific word embeddings and task-specific features to detect cyber security events from tweets. Our model employs a convolutional neural network (CNN) and a long short-term memory (LSTM) recurrent neural network which takes word level meta-embeddings as inputs and incorporates contextual embeddings to classify noisy short text. We collected a new dataset of cyber security related tweets from Twitter and manually annotated a subset of 2K of them. We experimented with this dataset and concluded that the proposed model outperforms both traditional and neural baselines. The results suggest that our method works well for detecting cyber security events from noisy short text.

1 Introduction

Twitter has become a medium where people can share and receive timely messages on about anything. People share facts, opinions, broadcast news and communicate with each other through these messages. Due to the low barrier to tweeting, and growth in mobile device usage, tweets might provide valuable information as people often share instantaneous updates such as the breaking news before even being broadcasted in the newswire *c.f.* Petrović et al. (2010). People also share cyber security events in their tweets such as zero day exploits, ransomwares, data leaks, security breaches, vulnerabilities *etc*. Automatically detecting such events might have various practical applications such as taking the necessary precautions promptly as well as creating self-awareness as illustrated in Fig. 1. Recently, working with the cyber security related text has garnered a lot of interest in both computer security and natural language processing (NLP) communities (*c.f.* Joshi et al. (2013); Ritter et al. (2015); Roy et al. (2017)). Nevertheless, detecting cyber security events from tweets pose a great challenge, as tweets are noisy and often lack sufficient context to discriminate cyber security events due to length limits. Recently, deep learning methods have shown to be outperforming traditional approaches in several NLP tasks (Chen and Manning, 2014; Bahdanau et al., 2014; Kim, 2014; Hermann et al., 2015). Inspired by this progress, our goal is to detect cyber security events in tweets by learning domain-specific word embeddings and task-specific features using neural architectures. The key contribution of this work is two folds. First, we propose an end-to-end learning system to effectively detect cyber security events from tweets. Second, we propose a noisy short text dataset with annotated cyber security events for unsupervised and supervised learning tasks. To our best knowledge, this will be the first study that incorporates domain-specific meta-embeddings and contextual embeddings for detecting cyber security events.

> Dear @AppleSupport, we noticed a *HUGE* security issue at MacOS High Sierra. Anyone can login as "root" with empty password after clicking on login button several times. Are you aware of it @Apple?

Figure 1: A cyber security event. A recently discovered security issue has been reported on Twitter which caught public attention. A security fix has been published right afterward.

2 Method

In the subsequent sections, we address the challenges to solve our task. The proposed system overview is illustrated in Fig. 2.

---

[*] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 1366–1372
Minneapolis, Minnesota, June 2 - June 7, 2019.

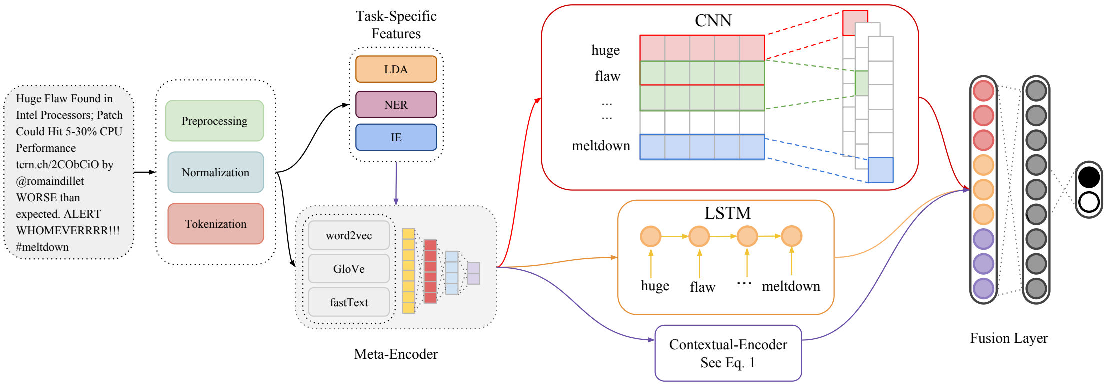


Figure 2: System Overview. A tweet is first pre-processed, then task-specific features and word level meta-embeddings are extracted to represent tokens. Finally, a Bi-LSTM, CNN, and Contextual Encoder are fused to classify the encoded tweet.

2.1 Meta-Embeddings

Word embedding methods might capture different semantic and syntactic features about the same word. To exploit this variety without losing the semantics, we learn meta-embeddings for words.

Word Embeddings. Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and fastText (Joulin et al., 2016; Bojanowski et al., 2016) are trained for learning domain specific word embeddings on the unlabeled tweet corpus.

Meta-Encoder. Inspired by Yin and Schütze (2015) we learn meta-embeddings for words with the aforementioned word embeddings. We use a Convolutional Autoencoder (Masci et al., 2011) for encoding $3xD$ size embeddings to a $1xD$ dimensional latent variable and to reconstruct the original embeddings from this latent variable. Both encoder and decoder are comprised of 2 convolutional layers where 32 neurons are used on each. The encoder part is shown in Fig. 3. We argue that this network learns a much simpler mapping while capturing the semantic and syntactic relations from each of these embeddings, thus leading to a richer word-level representation. Another advantage of learning meta-embeddings for words is that the proposed architecture alleviates the Out-of-Vocabulary (OOV) embeddings problem, as we still get embeddings from the fastText channel, in contrast to GloVe and word2vec, where no embeddings are available for OOV words.

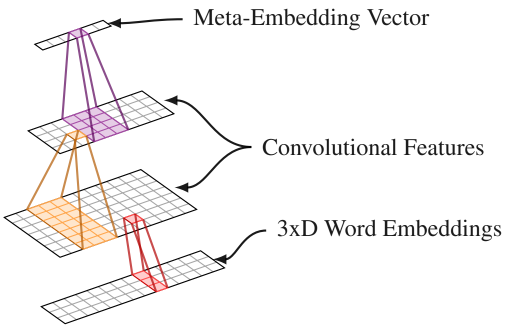


Figure 3: Convolutional encoder as a feature extractor. The decoder is symmetric to the encoder, and in inference time we drop the decoder and use only the encoder network.

2.2 Contextual Embeddings

To capture the contextual information, we learn task-specific features from tweets.

LDA. Latent Dirichlet Allocation (LDA) is a generative probabilistic model to discover topics from a collection of documents (Blei et al., 2003). LDA works in an unsupervised manner and learns a finite set of categories from a collection, thus represents documents as mixtures of topics. We train an LDA model to summarize each tweet by using the topic with the maximum likelihood *e.g.* with the topic “vulnerability” for the tweet in Fig 1.

NER. Named Entity Recognition (NER) tags the specified named entities from raw text into predefined categories. Named entities could be more general categories such as people, organizations, or specific entities can be learned by creating a dataset containing specific entity tags. We employ an automatically annotated dataset that contains entities from cyber security domain (Bridges et al., 2013) to train our Conditional Random Field model using handcrafted features, *i.e.*, uni-gram, bi-gram, and gazetteers. The dataset comprises of 850K tokens that contain named entities such as ‘Relevant Term’, ‘Operating System’,‘Hardware’, ‘Software’, ‘Vendor’, in the standard IOB-tagging format. Our NER model tags “password” as ‘Relevant Term’ and “Apple” as ‘Vendor’ for the tweet in Fig 1.

IE. Uncovering entities and the relations between those entities is an important task for detecting cyber security events. In order to address this we use Information Extraction (IE), in particular OpenIE annotator(Angeli et al., 2015) from the Stanford CoreNLP (Manning et al., 2014). Subsequently, we extract relations between noun phrases with the following dependency triplet $\langle arg_1, rel, arg_2 \rangle$, where arg_1, arg_2 denote the arguments and rel represents an implicit semantic relation between those arguments. Hence, the following triplet is extracted from the tweet in Fig. 1, $\langle we, noticed, huge\ security\ issue \rangle$.

Contextual-Encoder. We use the outputs of LDA, NER and IE algorithms to obtain a combined vector representation using meta-embeddings described in Sec. 2.1. Thus, contextual embeddings are calculated as follows[1].

$$\gamma(\tau) = \frac{f(\varphi(\tau)) + \sum_{i=1}^{N} f(\varsigma(\tau)^i) + \sum_{j=1}^{M} f(\delta(\tau)^j)}{N + M + 1} \quad (1)$$

where γ function extracts contextual embeddings and τ denotes a tweet, f, φ, ς and δ represent meta-embedding, LDA, NER, and IE functions, respectively. Lastly, N and M denote the output tokens.

2.3 Event Detection

Inspired by the visual question answering task (Antol et al., 2015), where different modalities are combined by CNNs and RNNs, we adopt a similar network architecture for our task. Prior to training, and inference we preprocess, normalize and tokenize each tweet as described in Sec. 3.

CNN. We employ a CNN model similar to that of (Kim, 2014) where we feed the network with static meta-embeddings. Our network is comprised of one convolutional layer with varying filter sizes, that is $2, 3, 5$. All tweets are zero padded to the maximum tweet length. We use $ReLU$ as activation and global max pooling at the end of CNN.

RNN. We use a bi-directional LSTM (Hochreiter and Schmidhuber, 1997) and read the input in both directions and concatenate forward and backward hidden states to encode the input as a sequence. Our LSTM model is comprised of a single layer and employs 100 neurons.

3 Experiments

Data Collection. We collected $2.5M$ tweets using the Twitter's streaming API over a period from `2015-01-01` to `2017-12-31` using an initial set of keywords, henceforth referred as *seed keywords* to retrieve cyber security related tweets. In particular, we use the main group names of cyber security taxonomy described in Le Sceller et al. (2017) as seed keywords e.g. 'denial of service', 'botnet', 'malware', 'vulnerability', 'phishing', 'data breach' to retrieve relevant tweets. Using seed keywords is a practical way to filter out noise considering sparsity of cyber security related tweets in the whole tweet stream. After the initial retrieval, we use `langid.py` (Lui and Baldwin, 2012) to filter out non-English tweets.

Data Preprocessing. We substitute user handles with $\$mention\$$, and hyperlinks with $\$url\$$. We remove *emoticons* and reserved keyword RT which denotes retweets. We substitute hashtags by removing the prefix # character. We limit characters that repeat more than two times, remove capitalization and tokenize tweets using the Twitter tokenizer in `nltk` library. We normalize non-standard forms, *i.e.* writing *cu tmrrw* instead of *see you tomorrow*. Although there are several reasons for that, the most prominent one is that people tend to mimic prosodic effects in speech (Eisenstein, 2013). To overcome this, we use lexical normalization, where we substitute OOV tokens with in-Vocabulary (IV) standard forms, *i.e.* a standard form available in a dictionary. In particular we use UniMelb (Han et al., 2012), UTDallas (Liu et al., 2011) datasets. Lastly, we remove identical tweets and check the validity by removing tweets with less than 3 non-special tokens.

Data Annotation. We instructed cyber security domain experts for manual labelling of the dataset. Annotators are asked to provide a binary label for whether there is a cyber security event in the given tweet or not. Annotators are told to skip tweets if they are unsure about their decisions. Finally, we validated annotations by only accepting annotations if at least 3 among 4 annotators agreed on. Therefore, we presume the quality of attained ground truth labels is dependable. Overall, $2K$ tweets are annotated.

Dataset Statistics. After preprocessing, our initial $2.5M$ tweet dataset is reduced to $1.7M$ tweets where $2K$ of them are labeled[2]. The labeled dataset is somewhat balanced as there are 843 event-related tweets and 1157 non-event tweets. The training and testing sets have 1600 and 400 samples, respectively.

---

[1] We used zero vectors for the non-existent relations.

[2] Available at `http://stmai.github.io/cydec`

Training. We used Keras with Tensorflow backend in our neural models. For fastText and word2vec embeddings we used Gensim, and for GloVe we used `glove-python` library. For training the word embeddings, we use the entire tweet text corpus and obtain 100 dimensional word embeddings. We set word2vec and fastText model's alpha parameter to 0.025 and window size to 5. For GloVe embedding model, we set the learning rate to 0.01, alpha to 0.75 and maximum count parameter to 100. For embedding models, we determined the minimum count parameter to 5, culminating in the elimination of infrequent words. Consequently, we have 3, 100-dimensional word embedding tensor in which first, second and third channels consist of word2vec, fastText and GloVe embeddings respectively. We then, encode these $3x100$ dimensional embeddings into $1x128$ dimensional representations by using our Meta-Encoder. We train our two channel architecture that combines both LSTM and CNN with 2 inputs: meta-embeddings and contextual embeddings. We use meta-embeddings for feature learning via LSTM and CNN, and their feature maps are concatenated with contextual embeddings in the Fusion Layer. In the end, fully connected layers and a softmax classifier are added, and the whole network is trained to minimize binary cross entropy loss with a learning rate of 0.01 by using the Adam optimizer (Kingma and Ba, 2014).[3]

Baselines. To compare with our results, we implemented the following baselines: `SVM with BoW`: We trained an SVM classifier using Bag-of-words (BoW) which provides a simplified representation of textual data by calculating the occurrence of words in a document. `SVM with meta-embeddings`: We trained an SVM classifier with the aforementioned meta-embeddings. `CNN-Static`: We used Kim (2014)'s approach using word2vec embeddings.

Results. Table 1 summarizes the overall performance of each method. To compare the models, we used four different metrics: accuracy, recall, precision and F1-score. Each reported result is the mean of a 5-fold cross validation experiment. It is clear that our method outperforms various simple and neural baselines. Also, in Table 2, we provide results of our proposed model along with the ground-truth annotations. We also provide results with the different combinations of contextual features, i.e., LDA, NER, IE[4].

Models	Accuracy	Precision	Recall	F1
SVM+BoW	0.75	0.71	0.70	0.70
SVM+Meta-Emcoder	0.71	0.64	0.61	0.63
CNN-static (Yoon Kim, 2014)	0.76	0.72	0.69	0.70
Human	0.65	0.70	0.87	0.59
CNN+Meta-Encoder	0.78	0.78	0.63	0.70
LSTM+Meta-Encoder	0.78	0.74	0.70	0.72
Ours (see Fig. 2)	**0.82**	**0.79**	**0.72**	**0.76**

Table 1: Results

Human Study. 8 different subjects are thoroughly instructed about what is considered as a cyber security event and individually asked to label 50 randomly selected tweets from the test set. The results are provided in Table 3.

Error Analysis. In order to understand how our system performs, we randomly select a set of erroneously classified instances from the test dataset.

Type I Errors. Our model identifies this tweet as an event "uk warned following breach in air pollution regulation \$url\$" whereas it is clearly about the a breach of a regulation. We hypothesize that this is due to the lack of sufficient training data. Following tweet is also identified as an event "wannacry ransomware ransomwareattack ransomwarewannacry malware \$url\$". We suspect that the weights of multiple relevant terms deceive the model.

Type II Errors. Our model fails to identify the following positive sample as an event. For "playstation network was the target of miraibotnet ddos attack guiding tech rss news feed search" our model fails to recognize the 'miraibotnet' from the tweet. We suspect this is due to the lack of hashtag decomposition; otherwise, the model could recognize 'mirai' and 'botnet' as separate words.

Discussions. Cyber security related tweets are complicated and analysing them requires in-depth domain knowledge. Although human subjects are properly instructed, the results of the human study indicate that our task is challenging and humans can hardly discriminate cyber security events amongst cyber security related tweets. To further investigate this, we plan to increase the number of human subjects. One limitation of this study is that we do not consider hyperlinks and user handles which might provide additional information. One particular problem we have not addressed in this work is hashtag decomposition. Error analysis indicates that our model might get confused by challenging examples due to ambiguities and lack of context.

---

[3] See supplementary for hyperparameter choices.

[4] See supplementary for feature combination details.

Tweet	Our Model	GT
that thing where you run corporation phishing test and user does'nt click it but clicks the next message which is real phishing email sigh	0	0
march 03 the fbi investigating alleged data breach at the center for election systems at kennesaw state university	1	1
cia malware codenames are freaking amazing	1	0
proprietary software on malware vista10 is more malicious	0	1
in huge breach of trust deidentified medical history data from millions of australians can be matched to individuals url2	0	1
hackers take aim at your $mention$ account with this new phishing attack cdwsocia	0	1
wannacry ransomware ransomwareattack ransomwarewannacry malware url	1	0

Table 2: Some Example Results

Subjects	Accuracy	Precision	Recall	F1	Cohen's κ
#1	0.62	0.54	1	0.7	0.43
#2	0.54	0.5	0.95	0.65	0.33
#3	0.66	0.58	0.91	0.71	0.42
#4	0.66	0.57	1	0.73	0.46
#5	0.8	0.8	0.73	0.77	0.28
#6	0.66	0.57	0.95	0.72	0.41
#7	0.7	0.63	0.82	0.71	0.31
#8	0.6	0.56	0.60	0.58	0.28
Average	0.65	0.70	0.87	0.59	0.36

Table 3: Human Study Results

4 Related Work

Event detection on Twitter is studied extensively in the literature (Petrović et al., 2010; Sakaki et al., 2010; Weng and Lee, 2011; Ritter et al., 2012; Yuan et al., 2013; Atefeh and Khreich, 2015). Banko et al. (2007) proposed a method to extract relational tuples from web corpus without requiring hand labeled data. Ritter et al. (2012) proposed a method for categorizing events in Twitter. Luo et al. (2015) suggested an approach to infer binary relations produced by open IE systems. Recently, Ritter et al. (2015) introduced the first study to extract event mentions from a raw Twitter stream for event categories DDoS attacks, data breaches, and account hijacking. Chang et al. (2016) proposed an LSTM based approach which learns tweet level features automatically to extract events from tweet mentions. Lately, Le Sceller et al. (2017) proposed a model to detect cyber security events in Twitter which uses a taxonomy and a set of seed keywords to retrieve relevant tweets. Tonon et al. (2017) proposed a method to detect events from Twitter by using semantic analysis. Roy et al. (2017) proposed a method to learn domain-specific word embeddings for sparse cyber security text. Prior art in this direction (Ritter et al., 2015; Chang et al., 2016) focuses on extracting events and in particular predicting the events' posterior given the presence of particular words. Le Sceller et al. (2017); Tonon et al. (2017) focus on detecting cyber security events from Twitter. Our work distinguishes from prior studies as we formulate cyber security event detection problem as a classification task and learn meta-embeddings from domain-specific word embeddings while incorporating task-specific features and employing neural architectures.

5 Conclusion

We introduced a novel neural model that utilizes meta-embeddings learned from domain-specific word embeddings and task-specific features to capture contextual information. We present a unique dataset of cyber security related noisy short text collected from Twitter. The experimental results indicate that the proposed model outperforms the traditional and neural baselines. Possible future research direction might be detecting cyber security related events in different languages.

Acknowledgments

We would like to thank Merve Nur Yılmaz and Benan Bardak for their invaluable help with the annotation process on this project. This research is fully supported by STM A.Ş. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the sponsor.

References

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the ACL 2015*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE ICCV*, pages 2425–2433.

Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. Version 7.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. arXiv:1607.04606. Version 2.

Robert A Bridges, Corinne L Jones, Michael D Iannacone, Kelly M Testa, and John R Goodall. 2013. Automatic labeling for entity extraction in cyber security. arXiv:1308.4941. Version 3.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. In *HLT-NAACL*, pages 400–410.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on EMNLP*, pages 740–750.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *HLT-NAACL*, pages 359–369.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 joint conference on EMNLP and CoNLL*, pages 421–432. ACL.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in NIPS*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arnav Joshi, Ravendar Lal, Tim Finin, and Anupam Joshi. 2013. Extracting cybersecurity related linked data from text. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 252–259. IEEE.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. arXiv:1607.01759. Version 3.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv:1408.5882. Version 2.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Version 9.

Quentin Le Sceller, ElMouatez Billah Karbab, Mourad Debbabi, and Farkhund Iqbal. 2017. Sonar: Automatic detection of cyber security events over the twitter stream. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 23. ACM.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies: short papers-Volume 2*, pages 71–76. ACL.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. ACL.

Kangqi Luo, Xusheng Luo, and Kenny Qili Zhu. 2015. Inferring binary relation schemas for open information extraction. In *EMNLP*, pages 555–560.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781. Version 3.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *EMNLP*.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the NAACL*, pages 181–189. ACL.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on KDD*, pages 1104–1112. ACM.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th International Conference on World Wide Web*, pages 896–905. International World Wide Web Conferences Steering Committee.

Arpita Roy, Youngja Park, and SHimei Pan. 2017. Learning domain-specific word embeddings from sparse cybersecurity texts. arXiv:1709.07470. Version 1.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Alberto Tonon, Philippe Cudré-Mauroux, Albert Blarer, Vincent Lenders, and Boris Motik. 2017. Armatweet: Detecting events by semantic tweet analysis. In *European Semantic Web Conference*, pages 138–153. Springer.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. *ICWSM*, 11:401–408.

Wenpeng Yin and Hinrich Schütze. 2015. Learning meta-embeddings by using ensembles of embedding sets. arXiv:1508.04257. Version 2.

Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Who, where, when and what: discover spatio-temporal topics for twitter users. In *Proceedings of the 19th ACM SIGKDD international conference on KDD*, pages 605–613. ACM.

White-to-Black: Efficient Distillation of Black-Box Adversarial Attacks

Yotam Gil[1]* and **Yoav Chai**[2]* and **Or Gorodissky**[1]* and **Jonathan Berant**[2,3]
[1]School of Electrical Engineering, Tel-Aviv University
[2]School of Computer Science, Tel-Aviv University
[3]Allen Institute for Artificial Intelligence
`{yotamgil@mail, yoavchai1@mail, orarieg@mail, joberant@cs}.tau.ac.il`

Abstract

Adversarial examples are important for understanding the behavior of neural models, and can improve their robustness through adversarial training. Recent work in natural language processing generated adversarial examples by assuming white-box access to the attacked model, and optimizing the input directly against it (Ebrahimi et al., 2018). In this work, we show that the knowledge implicit in the optimization procedure can be distilled into another more efficient neural network. We train a model to emulate the behavior of a white-box attack and show that it generalizes well across examples. Moreover, it reduces adversarial example generation time by 19x-39x. We also show that our approach transfers to a black-box setting, by attacking The Google Perspective API and exposing its vulnerability. Our attack flips the API-predicted label in 42% of the generated examples, while humans maintain high-accuracy in predicting the gold label.

1 Introduction

Adversarial examples (Goodfellow et al., 2014) have gained tremendous attention recently, as they elucidate model limitations, and expose vulnerabilities in deployed systems. Work in natural language processing (NLP) either (a) used simple heuristics for generating adversarial examples (Jia and Liang, 2017; Belinkov and Bisk, 2017; Iyyer et al., 2018), or (b) assumed white-box access, where the attacker has access to gradients of the model with respect to the input (Feng et al., 2018; Ebrahimi et al., 2018). In this approach, adversarial examples are constructed through an optimization process that uses gradient descent to search for input examples that maximally change the predictions of a model. However, developing attacks with only black-box access to a model (no access to gradients) is still under-explored in NLP.

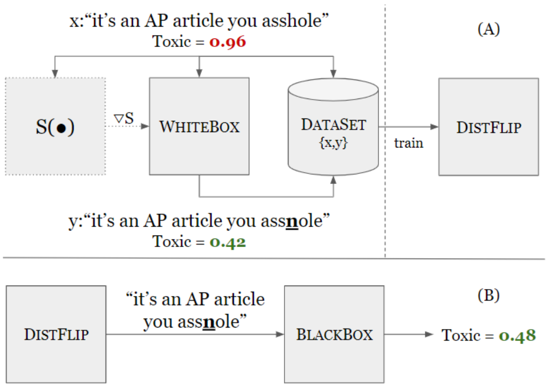


Figure 1: (A) Using a white-box attack we generate adversarial examples for a source toxicity model $S(\cdot)$. We train our black-box attacker, DISTFLIP, to emulate the white-box attack. (B) We use DISTFLIP to attack a black-box model.

Inspired by work in computer vision (Papernot et al., 2016; Liu et al., 2017), we show in this work that a neural network can learn to emulate the optimization process of a white-box attack and generalize well to new examples. Figure 1 gives a high-level overview of our approach. We assume a text classification model and a white-box attack that flips characters in the input to modify the model prediction (Ebrahimi et al., 2018). We generate output adversarial examples using the white-box attack and train a neural network from these input-output examples to imitate the white-box attack. This results in a much more efficient attack whose run-time is independent of the optimization process. Moreover, assuming adversarial examples transfer between different models, our distilled model can now be used to generate adversarial examples for black-box attacks directly.

We use our approach to attack a toxicity classifier, aimed at detecting toxic language on social media (Hosseini et al., 2017). We show that our model achieves a speed-up of 19x-39x in generating adversarial examples while maintaining similar attack quality, compared to an optimization-based

---

[*] Equal contribution

Proceedings of NAACL-HLT 2019, pages 1373–1379
Minneapolis, Minnesota, June 2 - June 7, 2019.

method. We then use our model for a black-box attack against Google Perspective API for detecting toxic sentences, and find that 42% of our generated sentences are misclassified by the API, while humans agree that the sentences are toxic.

Our code can be downloaded from `http://github.com/orgoro/white-2-black`.

2 Background

Adversarial examples have been extensively used recently in NLP for probing and understanding neural models (Jia and Liang, 2017; Weber et al., 2018). Methods for generating such examples include adding random or heuristically constructed noise (Belinkov and Bisk, 2017; Rodriguez and Rojas-Galeano, 2018; Gao et al., 2018), meaning-preserving modifications that change the surface form (Iyyer et al., 2018; Ribeiro et al., 2018), and human-in-the-loop generation (Wallace et al., 2018). A weakness of such models is that they do not directly try to modify the prediction of a model, which can reduce efficacy (Kurakin et al., 2016). In the white-box setting, Feng et al. (2018) have changed the meaning of an input without changing model output using access to gradients, and Ebrahimi et al. (2018) proposed HOTFLIP, the aforementioned white-box attack that we emulate for flipping input characters.

In computer vision, Papernot et al. (2016) and Liu et al. (2017) have shown that adversarial examples generated by a white-box model can be helpful for a black-box attack. Generating adversarial text examples is more challenging than adversarial images, because images are points in a continuous space, and thus it is easier to apply norm restrictions. Text examples have a discrete structure and thus such approaches have not been investigated for adversarial text generation yet.

3 HOTFLIP

HOTFLIP (Ebrahimi et al., 2018) is a white-box method for generating adversarial examples for a character-level neural model. It uses the gradient with respect to a 1-hot input representation to estimate the character flip that incurs the highest cost. We briefly describe HOTFLIP, which we use to generate training examples for our distilled model.

Let $\mathrm{x} = ((x_1^1, \ldots, x_1^n), \ldots, (x_m^1, \ldots, x_m^n))$ be a sentence represented as a sequence of m characters, encoded as 1-hot vectors over a vocabulary of size n. Define $L(\mathrm{x}, y)$ to be the loss of a trained model for the input x with respect to a label y.

HOTFLIP requires one function evaluation (forward pass) and one gradient computation (backward pass) to compute a first-order estimate of the best possible character flip in x. Flipping the i^{th} character from a to b can be represented by this vector: $\overrightarrow{v_{i_b}} = (\ldots, (0, \ldots, -1, \ldots, 1, \ldots, 0)_i, \ldots)$, where -1 and 1 are in the positions for the a^{th} and b^{th} characters respectively. A first-order estimate of the change in loss can be obtained by computing $\nabla_x L(\mathrm{x}, y)$ with back-propagation, and taking a directional derivative along $\overrightarrow{v_{i_b}}$:

$$\nabla_{\overrightarrow{v_{i_b}}} L(\mathrm{x}, y) = \nabla_x L(\mathrm{x}, y) \cdot \overrightarrow{v_{i_b}}.$$

We can now choose the character-flip a to b that maximizes this estimate using the gradients with respect to the input x:

$$\arg\max_{i,b} [\nabla L(\mathrm{x}, y) \cdot \overrightarrow{v_{i_b}}] = \arg\max_{i,b} [\frac{\partial L}{\partial x_i^b} - \frac{\partial L}{\partial x_i^a}].$$

To perform a sequence of flips, any search procedure can be applied. HOTFLIP uses beam search of r steps, keeping at each step the top-K flip sequences that increased $L(\mathrm{x}, y)$ the most. This require $O(K \cdot r)$ forward and backward passes. Character insertions and deletions are modeled as multiple flips, but for simplicity, we only consider *character flips* in our work.

The main drawbacks of HOTFLIP are that it does not gain any knowledge from optimizing over multiple examples, and that its efficiency is strongly tied to the search procedure used ($O(K \cdot r)$ forward and backward passes per example for beam-search). Next, we present our model that overcomes these limitations.

4 Distilling a Black-box Attack

We are interested in whether (a) the knowledge in the optimization process of HOTFLIP can be distilled into a neural model, and (b) whether this model can generalize to a black-box attack. Therefore, the outline of our approach is as follows:

1. Train a *source text classification model* on data from a similar distribution to the data used to train the *target black-box model*.
2. Generate adversarial examples by performing white-box optimization (with HOTFLIP) on the source model.
3. Train an efficient *attacker* to generate adversarial examples, and perform a black-box attack against the target model.

We assume a training set $\mathcal{D} = \{(\mathrm{x_i}, y_i)\}_{i=1}^{N}$ used to train a character-based source model $S(\cdot)$ that takes a character sequence x as input, and returns a distribution over the output space $\mathcal{Y}$ (details on the source model are in Section 5). We now elaborate on the processes of data generation and training of the attacker.

Data generation We take examples (x, y) from $\mathcal{D}$ and run HOTFLIP with search until we obtain an adversarial example $\bar{\mathrm{x}}$ such that the probability of the gold label is low, that is, $[S(\bar{\mathrm{x}})]_y < \tau$, where $[S(\bar{\mathrm{x}})]_y$ is the probability given by $S(\bar{\mathrm{x}})$ to $y \in \mathcal{Y}$ and τ is a threshold (we use $\tau = 0.15$).

Let $s = ((\mathrm{x}^{(0)} = \mathrm{x}), \mathrm{x}^{(1)}, \dots, (\mathrm{x}^{(l)} = \bar{\mathrm{x}}))$ be the sequence of sentences generating $\bar{\mathrm{x}}$, where every consecutive pair $(\mathrm{x}^{(\mathrm{i})}, \mathrm{x}^{(\mathrm{i}+1)})$ differs by a single character: the character in position $j^{(i)}$ was flipped to the character $c^{(i)}$. Our attacker is trained from examples $(\mathrm{x}^{(\mathrm{i})}, (j^{(i)}, c^{(i)}))$ generated from every pair of consecutive sentences in s. For example, if the sentence is the one-word sentence "Asshole", and after flipping one character it becomes "Assnole", the example would be ("Asshole", (4, 'n')).

Model training Our attacker takes a character sequence x as input and outputs a pair (j^*, c^*), where $j^* \in [1, \dots, m]$ is the position of the character to be flipped, and c^* is the target character.

Figure 2 describes the architecture of our model.

Our model embeds each character using a pre-trained 300-dimensional character embedding[1], and then passes the character sequence through a 1-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with 512-dimensional hidden states. The BiLSTM h_j hidden state in every position are passed through two feed-forward networks, one for replacement prediction (which character to flip) and one for target prediction (what target character to choose). The network that perfoms replacement prediction has 2 hidden layers of dimensions 100 and 50 with ReLU activations, and a single logit l_j as output per position. The output distribution over the sentence positions is given by a softmax over all character positions. At inference time j^* is computed with an argmax instead of a softmax.

The network that produces target prediction has two-hidden layers of dimensions 100 and 100 with ReLU activations and outputs a vector of logits $v_j \in \mathbb{R}^{96}$ per position with a softmax layer, which provides a distribution over the character vocabulary. The target character c^* is computed at inference time with an argmax over the target position $\mathrm{x}_{\mathrm{j}^*}$.

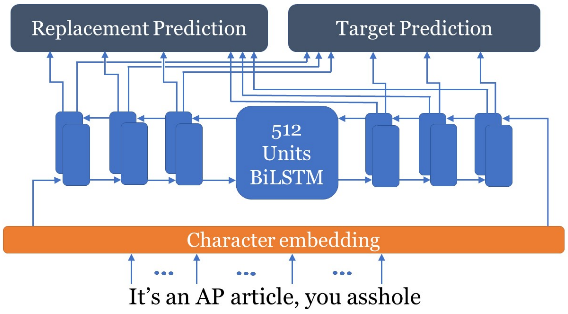


Figure 2: The architecture of our attacker network.

Our loss function is simply the sum of two cross-entropy terms: one for the gold position, and one for the gold character in the gold position.

Running our model is more efficient that HOTFLIP and run-time is independent of the optimization procedure. A forward pass in our model is equivalent to $2 \cdot K$ steps in HOTFLIP with beam-search. We show this leads to large practical speed-ups in Section 5.

5 Experiments

We now empirically investigate whether our method can be used to attack classifiers for detecting "toxic" language on social media. Recently a challenge by Alphabet aimed to improve labeling of toxic comments that are rude and disrespectful. Alphabet released a dataset[2] of 160K comments from Wikipedia discussions, and classified each comment to six labels. We focus on the *Toxic* label only, which represents 9.5% of the dataset.

We used the dataset to train the source model $S(\cdot)$, which runs a 2-layer bidirectional GRU (Cho et al., 2014) over the input x, and then uses an attention (Bahdanau et al., 2015) pooling layer to obtain a fixed dimensional vector for x. This vector is passed through a feed-forward layer to compute the probability that x is toxic. The accuracy of the source model is 96.5% AUC – comparable to the top submissions in the challenge.

We used the 13,815 toxic-labeled sentences from the training set to generate adversarial examples for training the attacker as described above. Dataset

---

[1] `https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt`

[2] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data`

generation depends on the search procedure, and we experiment with 3 setups: (a) HOTFLIP-5: beam-search with K=5, (b) HOTFLIP-10: beam-search with K=10, and (C) HOTFLIP+: a more expensive search procedure that calls $S(\cdot)$ more frequently. We describe the details of this procedure in Appendix A. Because our attacker is independent of the search procedure, inference time is not affected by the search procedure at data generation time. This results in three distilled models: DISTFLIP-5, DISTFLIP-10, and DISTFLIP+.[3]

Attacking the source model We compare the performance of DISTFLIP variants against HOTFLIP variant, including HOTFLIP-1 (K=1). We also compare to a RANDOM baseline, which chooses a position and target character randomly, and to an ATTENTION baseline, which uses the attention layer of $S(\cdot)$ to choose the character position with maximum attention to flip, and selects a target character randomly.

Table 1 summarizes our results. We report the average number of flips required to change the prediction of toxic sentences in the source model, the slow-down per single character-flip, and the slow-down per attack, which is computed by multiplying slow-down per flip by the ratio of the number of flips required per attack. Because roughly 15% of the examples in the dataset mostly contain repeated profanities that require many flips, we also report the average number of flips for the other 85% of the examples.

We observe that HOTFLIP+ requires the fewest flips to change model prediction, but attacks are very slow. The number of flips per attack for DISTFLIP+ is comparable to HOTFLIP-5 and HOTFLIP-10, but it achieves a speed-up of 19x-39x. DISTFLIP-5 and DISTFLIP-10 require a few more flips compared to DISTFLIP+. Overall attack quality is high, with less than two flips necessary for 85% of the examples for DISTFLIP+.

Figure 3 provides a more fine-grained view of the results by showing the proportion of sentences classified as toxic as a function of the number of flips. Overall, the picture is similar to Table 1, with DISTFLIP+ being comparable to HOTFLIP-10.

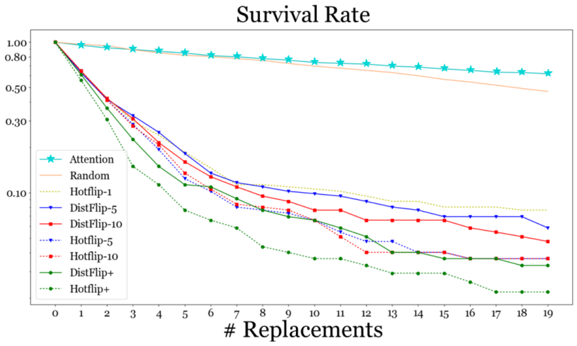


Figure 3: Proportion of sentences classified as toxic as a function of the number of flips for all models on the test set.

Attacking The Google Perspective API The Google perspective API[4] returns the probability that a sentence is toxic, where probability > 0.7 is classified as *toxic*, < 0.3 is *non-toxic*, and otherwise *uncertain*. The model itself is not publicly available. We randomly selected 136 toxic examples from the validation set and attacked them with DISTFLIP+ until the source model probability was < 0.5.

We measured the toxicity probability before and after our attack and found that the average toxicity probability decreased from 0.9 to 0.67, with an average of 5.0 flips per sentence. The label is flipped from *toxic* to *uncertain* or *non-toxic* in 42% of these examples.

Human validation To validate that toxic sentences remain toxic after our attack, we showed 5 independent annotators a total of 150 sentences from three classes: toxic sentences, non-toxic sentences, and attacked sentences (attacked by DISTFLIP-5). The same annotator was never shown a toxic sentence and its attacked counterpart. We asked annotators whether sentences are toxic, and measured average annotated toxicity.

We found that 89.8% of the toxic sentences were annotated as toxic, compared to 87.6% in the attacked toxic sentences. This shows that humans clearly view the sentences as toxic, even after our attack. Table 2 shows examples for sentences attacked by DISTFLIP-10 and the change in toxicity score according to The Google Perspective API.

6 Conclusion

In this work we present a general approach for distilling the knowledge in any white-box attack into a trained model. We show that this results in substantial speed-up (19x-39x) while maintaining comparable quality. Moreover, we show that our

---

[3] DISTFLIP-5 is trained on 64K generated examples, DISTFLIP-10 on 62K examples, and DISTFLIP+ on 46K.

[4] `https://www.perspectiveapi.com`

	DistFlip+	DistFlip-10	DistFlip-5	HotFlip+	HotFlip-10	HotFlip-5	HotFlip-1	Rand.	Att.
#flips	5.05	7.5	8.6	3.2	4.5	4.6	13.8	56.0	70.6
#flips for 85%	1.88	2.27	2.30	1.66	2.16	2.18	2.24	24.57	36.09
Flip slow-down	1x	1x	1x	168.8x	43.3x	21.3x	6.1x	-	-
Attack slow-down	1x	1.48x	1.7x	108x	**38.6x**	**19.4x**	16.7x	-	-

Table 1: Average number of flips to change the prediction of toxic sentences, average number of flips to change the prediction for 85% of the examples that do not contain repeated profanities, slow-down per flip compared to DistFlip+, and slow-down per sentence attack compared to DistFlip+. One character-flip using DistFlip takes 12ms on an Nvidia GTX1080Ti.

Original	**Adversarial**	**Google score**
decline=I don't fucking think so	decline=I don't **v**ucking think so	0.79 → 0.07
its an AP article you asshole, how is it not neutral???	its an AP article you ass**n**ole, how is it not neutral???	0.96 → 0.48
I think the 1 million sales is total bullshit though.	I think the 1 million sales is total bullsh**k**t though.	0.79 → 0.07
do not know juden! silence is golde juden, now shut up.	do not know juden! silence is golden juden, now **x**h**j**t up.	0.69 → 0.37
This article sucks . . woo woo wooooooo	This article **x**u**x**ks . . woo woo wooooooo	0.93 → 0.22
Also bring back the brendle article you piece of shit.	Also bring back the brendle article you p**k**ece of **x**h**k**t.	0.98 → 0.32
to be driven away and die	to be driven away and d**k**e	0.82 → 0.32

Table 2: Examples of sentences attacked by DistFlip-10 and The Google Perspective toxicity score before and after the attack.

attack transfers to a black-box setting: we expose the vulnerability of The Google Perspective API, and are able to change the prediction for 42% of input toxic examples, while humans easily detect that examples are still toxic.

Acknowledgements

We thank Raja Giryes for supporting this research, and the anonymous reviewers for their constructive feedback. This research was partially supported by The Israel Science Foundation grant 942/16, the Blavatnik Computer Science Research Fund, and The Yandex Initiative for Machine Learning.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 31–36.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *2018 IEEE Security and Privacy Workshops (SPW)*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *The Bright and Dark Sides of Computer Vision: Challenges and Opportunities for Privacy and Security workshop@CVPR*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *NAACL*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *EMNLP*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *ICLR workshop*.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. *ICLR*.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 856–865.

Nestor Rodriguez and Sergio Rojas-Galeano. 2018. Shielding google's language toxicity model against adversarial attacks.

Eric Wallace, Pedro Rodriguez, Shi Feng, and Jordan Boyd-Graber. 2018. Trick me if you can: Adversarial writing of trivia challenge questions. *ACL Student Research Workshop*.

Noah Weber, Leena Shekhar, and Niranjan Balasubramanian. 2018. The fine line between linguistic generalization and failure in seq2seq-attention models. *arXiv preprint arXiv:1805.01445*.

A Appendix: HotFlip+

Algorithm 1 describes the search procedure of HOTFLIP+. The main motivation behind this search procedure is that HOTFLIP uses gradients to estimate the character-flip that maximally changes the predictions of a model, but this estimate is not guaranteed to be correct. In HOTFLIP+ we try to overcome this limitation by performing pruning with gradients, and then actually evaluating a larger number of possible flips by running the source model on many possible flips. This makes the search procedure slower, because we have to run the source model (forward pass) much more frequently. For this algorithm we use beam size 3.

In Algorithm 1, the *toxicity score* of a sentence is the result of running it through the source model, and the *beam score* is the first-order estimate described in Section 3.

Algorithm 1 HotFlip+

```
procedure HOTFLIP+(sentence)
    beam ← Initialize beam with the original sentence and its toxicity score.
    while True do
        $bf, tox$ ← Pop from the beam the flipped sentence with lowest toxicity, and its toxicity.
        if $tox < 0.5$ then
            $break$
        create a new beam.
        for every beam entry in the current beam do
            compute all possible flipped sentences and their beam score (as in HOTFLIP).
            for $flip\_sent, flip\_score$ in flipped sentences do
                $min\_score$ ← minimal beam score on the beam.
                if $flip\_score > min\_score$ then                ▷ Prune using beam score
                    $tox$ ← Compute the toxicity of $flip\_sent$ with a forward pass.
                    $max\_tox\_in\_beam$ ← Pop from the new beam the maximal toxicity score.
                    if $tox < max\_tox\_in\_beam$ then
                        push $flip\_sent, flip\_score, tox$ to the new beam.
    return $bf$
```

Analyzing the Perceived Severity of Cybersecurity Threats Reported on Social Media

Shi Zong[1], Alan Ritter[1], Graham Mueller[2] and Evan Wright[3]
[1]The Ohio State University, OH, USA
[2]Leidos Inc., VA, USA
[3]FireEye LLC, CA, USA
`{zong.56, ritter.1492}@osu.edu`
`muellerwg@leidos.com`
`evan.wright@fireeye.com`

Abstract

Breaking cybersecurity events are shared across a range of websites, including security blogs (FireEye, Kaspersky, etc.), in addition to social media platforms such as Facebook and Twitter. In this paper, we investigate methods to analyze the severity of cybersecurity threats based on the language that is used to describe them online. A corpus of 6,000 tweets describing software vulnerabilities is annotated with authors' opinions toward their severity. We show that our corpus supports the development of automatic classifiers with high precision for this task. Furthermore, we demonstrate the value of analyzing users' opinions about the severity of threats reported online as an early indicator of important software vulnerabilities. We present a simple, yet effective method for linking software vulnerabilities reported in tweets to Common Vulnerabilities and Exposures (CVEs) in the National Vulnerability Database (NVD). Using our predicted severity scores, we show that it is possible to achieve a Precision@50 of 0.86 when forecasting high severity vulnerabilities, significantly outperforming a baseline that is based on tweet volume. Finally we show how reports of severe vulnerabilities online are predictive of real-world exploits.[1]

1 Introduction

Software vulnerabilities are flaws in computer systems that leave users open to attack; vulnerabilities are generally unknown at the time a piece of software is first published, but are gradually identified over time. As new vulnerabilities are discovered and verified they are assigned CVE numbers (unique identifiers), and entered into the National Vulnerability Database (NVD).[2] To help prioritize response efforts, vulnerabilities in the NVD are assigned severity scores using the Common Vulnerability and Scoring System (CVSS). As the rate of discovered vulnerabilities has increased in recent years,[3] the need for efficient identification and prioritization has become more crucial. However, it is well known that a large time delay exists between the time a vulnerability is first publicly disclosed to when it is published in the NVD; a recent study found that the median delay between the time a vulnerability is first reported online and the time it is published in the NVD is seven days; also, 75% of threats are first disclosed online giving attackers time to exploit the vulnerability.[4]

[redacted]

Just tested the Dirty COW vulnerability. Nasty. Especially when thinking about all the devices that will never be patched, like in the IoT.

3:26 PM - 28 Oct 2016

Figure 1: Example tweet discussing the dirty copy-on-write (COW) security vulnerability in the Linux kernel.

In this paper we present the first study of whether natural language processing techniques can be used to analyze users' opinions about the severity of software vulnerabilities reported online. We present a corpus of 6,000 tweets annotated with opinions toward threat severity, and empirically demonstrate that this dataset supports automatic classification. Furthermore, we propose a simple, yet effective method for linking software vulnerabilities reported on Twitter to entries in the NVD, using CVEs found in linked URLs. We then use our threat severity analyzer to conduct a large-scale study to validate the accuracy of users' opinions online against experts' severity

---

[1] Our code and data are available at https://github.com/viczong/cybersecurity_threat_severity_analysis.

[2] https://nvd.nist.gov/

[3] https://www.cvedetails.com/browse-by-date.php

[4] https://www.recordedfuture.com/vulnerability-disclosure-delay/

Proceedings of NAACL-HLT 2019, pages 1380–1390
Minneapolis, Minnesota, June 2 - June 7, 2019.

ratings (CVSS scores) found in the NVD. Finally, we show that our approach can provide an early indication of vulnerabilities that result in real exploits in the wild as measured by the existence of Symantec virus signatures associated with CVEs; we also show how our approach can be used to retrospectively identify Twitter accounts that provide reliable warnings about severe vulnerabilities.

Recently there has been increasing interest in developing NLP tools to identify cybersecurity events reported online, including denial of service attacks, data breaches and more (Ritter et al., 2015; Chang et al., 2016; Chambers et al., 2018). Our proposed approach in this paper builds on this line of work by evaluating users *opinions* toward the severity of cybersecurity threats.

Prior work has also explored forecasting software vulnerabilities that will be exploited in the wild (Sabottke et al., 2015). Features included structured data sources (e.g., NVD), in addition to the volume of tweets mentioning a list of 31 keywords. Rather than relying on a fixed set of keywords, we analyze message content to determine whether the author believes a vulnerability is severe. As discussed by Sabottke et al. (2015), methods that rely on tracking keywords and message volume are vulnerable to adversarial attacks from Twitter bots or sockpuppet accounts (Solorio et al., 2013). In contrast, our method is somewhat less prone to such attacks; by extracting users' opinions expressed in individual tweets, we can track the provenance of information associated with our forecasts for display to an analyst, who can then determine whether or not they trust the source of information.

2 Analyzing Users' Opinions Toward the Severity of Cybersecurity Threats

Given a tweet t and named entity e, our goal is to predict whether or not there is a serious cybersecurity threat towards the entity based on context. For example, given the context in Figure 2, we aim at predicting the severity level towards *adobe flash player*. We define an author's perceived severity toward a threat using three criteria: (1) does the author believe that their followers should be worried about the threat? (2) is the vulnerability easily exploitable? and (3) could the threat affect a large number of users? If one or more of these criteria are met, then we consider the threat to be severe.

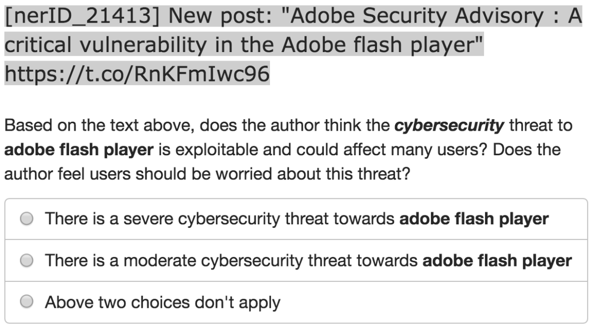
2.1 Data Collection

To collect tweets describing cybersecurity events for annotation, we tracked the keywords "ddos" and "vulnerability" from Dec 2017 to July 2018 using the Twitter API. We then used the Twitter tagging tool described by Ritter et. al. (2011) to extract named entities,[5] retaining tweets that contain at least one named entity. To cover as many linguistic variations as possible, we used Jaccard similarity with a threshold of 0.7 to identify and remove duplicated tweets with same date.[6]

2.2 Mechanical Turk Annotation

We paid crowd workers on Amazon Mechanical Turk to annotate our dataset. The annotation was performed in two phases; during the first phase, we asked workers to determine whether or not the tweet describes a cybersecurity threat toward a target entity, in the second phase the task is to determine whether the author of the tweet believes the threat is severe; only tweets that were judged to express a threat were annotated in the second phase. Each HIT contained 10 tweets to be annotated; workers were paid $0.20 per HIT. In pilot studies we tried combining these two annotations into a single task, but found low inter-rater agreement, especially for the threat severity judgments, motivating the need for separation of the annotation procedure into two tasks.

Figure 2 shows a portion of the annotation interface presented to workers during the second phase of annotation. Details of each phase are described below, and summarized in Table 1.



[nerID_21413] New post: "Adobe Security Advisory : A critical vulnerability in the Adobe flash player" https://t.co/RnKFmIwc96

Based on the text above, does the author think the ***cybersecurity*** threat to **adobe flash player** is exploitable and could affect many users? Does the author feel users should be worried about this threat?

○ There is a severe cybersecurity threat towards **adobe flash player**

○ There is a moderate cybersecurity threat towards **adobe flash player**

○ Above two choices don't apply

Figure 2: A portion of the annotation interface shown to MTurk workers during the threat severity annotation.

Threat existence annotation: Not all tweets in our dataset describe cybersecurity threats, for example many tweets discuss different senses of the

---

[5] https://github.com/aritter/twitter_nlp

[6] We sampled a dataset of 6,000 tweets to annotate.

Anno. Tweets Total	1st Annotation (5 workers per tweet)			2nd Annotation (10 workers per tweet)		
	Label	# Tweets	%	Label	# Tweets	%
6,000	With threat	2,543 (1,966 for 2nd anno.)	42.4	Severe threat	506	25.7
				Moderate threat	1,460	74.3
	Without threat	3,457	57.6	/		

Table 1: Number of annotated tweets with break-down percentages to each category. In 1st annotation, a tweet contains a threat if more than 3 workers vote for it. In 2nd annotation, a threat is severe if more than 6 workers agree on it. Number of workers cut-offs are determined by comparing to our golden annotations in pilot studies.

word "vulnerability" (e.g., "It's OK to show vulnerability"). During the first phase of our annotation process, workers judged whether or not there appears to be a cybersecurity threat towards the target entity based on the content of the corresponding tweet. We provide workers with 3 options: the tweet indicates (a) a cybersecurity threat towards given entity, (b) a threat, but not towards the target entity, or (c) no cybersecurity threat. Each tweet is annotated by 5 workers.

Threat severity annotation: In the second phase, we collect all tweets judged to contain threats by more than 3 workers in the first phase and annotated them for severity. 1,966 tweets were selected out of 6,000.[7] For each tweet we provided workers with 3 options: the tweet contains (a) a severe, (b) a moderate or (c) no threat toward the target entity. During our pilot study, we found this to be a more challenging annotation task, therefore we increased the number of annotators per tweet to 10 workers, which we found to improve agreement with our expert judgments.

Inter-annotator agreement: During both phases, we monitored the quality of workers' annotations using their agreement with each other. We calculated the annotation agreement of each worker against the majority vote of other workers. We manually removed data from workers who have an agreement less than 0.5, filling in missing annotations with new workers. We also manually removed data from workers who answered either uniformly or randomly for all HITs.

Agreement with expert judgments: To validate the quality of our annotated corpus we compared the workers' aggregated annotations against our own expert annotations. We independently annotated 150 randomly sampled tweets, 61 tweets of which are marked as containing severe or moderate threats. For threat existence annotation, we observe a 0.66 value of Cohen's κ (Artstein and Poesio, 2008) between the expert judgements and majority vote of 5 crowd workers. Although our threat severity annotation task may require some cybersecurity knowledge for accurate judgment, we still achieve 0.52 Cohen κ agreement by comparing majority vote from 10 workers with expert annotations.

2.3 Analyzing Perceived Threat Severity

Using the annotated corpus described in Section 2.2, we now develop classifiers that detect threats reported online and analyze users' opinions toward their severity. Specifically, given a named entity and tweet, $\langle e, t\rangle$, our goal is to estimate the probability the tweet describes a cybersecurity threat towards the entity, $p_{\text{threat}}(y|\langle e, t\rangle)$ and also the probability that the threat is severe, $p_{\text{severe}}(y|\langle e, t\rangle)$. In this section, we describe the details of these classifiers and evaluate their performance.

We experimented with two baselines to detect reports of cyberthreats and analyze opinions about their severity: logistic regression using bag-of-ngram features, and 1D convolutional neural networks. In the sections below we describe the input representations and details of these two models.

Logistic regression: We use logistic regression as our first baseline model for both classifiers. Input representations are bag-of-ngram features extracted from the entire tweet content. Example features are presented in Table 4. We use context windows of size 2, 3 and 4 to extract features. We map extracted n-grams that occur only once to a $\langle\texttt{UNK}\rangle$ token. In all our experiments, we replace named entities with a special token $\langle\texttt{TARGET}\rangle$; this helps prevent our models from biasing towards specific entities that appear in our training corpus. All digits are replaced with 0.

---

[7] We further deduplicate pairs of tweets where the longest common subsequence covers the majority of the text contents. During deduplication all hashtags and URLs were removed and digits were replaced with 0.

Convolutional neural networks: We also experimented with 1D convolutional neural networks (Collobert et al., 2011; Kim, 2014). Given a tweet, the model first applies convolutional operations on input sequences with various filters of different sizes. The intermediate representations for each filter are aggregated using max-pooling over time, followed by a fully connected layer. We choose convolution kernel sizes to be 3, 4 and 5-grams with 100 filters for each. We minimize cross-entropy loss using Adam (Kingma and Ba, 2015); the learning rate is set to 0.001 with a batch size of 1 and 5 epochs.

Word embeddings: We train our own cybersecurity domain word embeddings based on GloVe (Pennington et al., 2014), as 39.7% of our tokens are treated as OOV words in GloVe pre-trained Twitter embeddings. We used a corpus of 609,470 cybersecurity-related tweets (described in Section 2.1) as our training corpus. The dimension of word embeddings is 50. Table 2 shows nearest neighbors for some sampled cybersecurity terms based on the learned embeddings.

During network training, we initialize word embedding layer with our own embeddings. We initialize tokens not in our trained embeddings by randomized vectors with uniform distribution from -0.01 to 0.01. We fine-tune the word embedding layer during training.

Token	Nearest Neighbors
#ddos	attacks, ddos, datacenter-insider, attack, #cyberattack
#hackers	hackers, sec_cyber, #blackberryz00, #malware, #hacking
threats	defenses, cyberrisk, #cybersecurity, threat, #iot-based
vulnerability	risk, ..., #vulnerability, strength, critical

Table 2: Nearest neighbors to some cybersecurity related tokens in our trained word embeddings. Embeddings are trained by using GloVe. Similar tokens are sorted by cosine similarity scores.

2.3.1 Experimental Setup

For threat existence classification, we randomly split our dataset of 6,000 tweets into a training set of 4,000 tweets, a development set of 1,000 tweets, and test set of 1,000 tweets. For the threat severity classifier, we only used data from 2nd phase of annotation. This dataset consists of 1,966 tweets that were judged by the mechanical turk workers to describe a cybersecurity threat towards the target entity. We randomly split this dataset into a training set of 1,200 tweets, a development set of 300 tweets, and a test set of 466 tweets. We collapsed the three annotated labels into two categories based on whether or not the author expresses an opinion that the threat towards the target entity is severe.

2.3.2 Results

Threat existence classifier: The logistic regression baseline has good performance at identifying threats, which we found to be a relatively easy task; area under the precision-recall curve (AUC) on the development and test set presented in Table 5. This enables accurate detection of trending threats online by tracking cybersecurity keywords using the Twitter streaming API, following an approach that is similar to prior work on entity-based Twitter event detection (Ritter et al., 2012; Zhou et al., 2014; Wei et al., 2015). Table 3 presents an example of threats detected using this procedure on Nov. 22, 2018.[8]

Threat severity classifier: Figure 3 shows precision recall curves for the threat severity classifiers. Logistic regression with bag-of-ngram features provides a strong baseline for this task. Table 4 presents examples of high-weight features from the logistic regression model. These features often intuitively indicate severe threats, e.g. “critical vulnerability”, “a massive”, “million”, etc. Without much hyperparameter tuning on the development set, the convolutional neural network consistently achieves higher precision at the same level of recall as compared to logistic regression. We summarize the performance of our threat existence and severity classifiers in Table 5.

3 Forecasting Severe Cybersecurity Threats

In Section 2 we presented methods that can accurately detect threats reported online and analyze users’ opinions about their severity. We now explore the effectiveness of this model for forecasting. Specifically, we aim to answer the following questions: **(1)** To what extent do users’ opinions about threat severity expressed online align with expert judgments? **(2)** Can these opinions provide an early indicator to help prioritize threats based

---

[8] A live demo is available at: http://kb1.cse.ohio-state.edu:8123/events/threat

Named Entity	Example Tweet	Existence	Severity
apple	RT AsturSec: A kernel vulnerability in Apple devices gives access to remote code execution - Packt Hub #infosec #CyberSecurity https://t....	0.96	0.59
google	RT binitamshah: Unfixed spoofing vulnerability in Google Inbox mobile apps https://t.co/TWx7jSi1gc	0.78	0.17
adobe	RT Anomali: Adobe released patches for three "important-ranked" severity vulnerabilities, including one vulnerability in Adobe Acrobat and...	0.76	0.32
flash	Vulnerability in Flash player allowing code execution. Patch before Black Friday: https://t.co/4idb570d1E #CyberSecurity #vulnerability	0.71	0.43
mac	adobe's flash player for windows, mac and linux has a critical vulnerability that should be patched as a top priori... https://t.co/LLlPATy9vR	0.69	0.88

Table 3: Top five threats extracted with highest confidence on Nov. 22, 2018. For each entity we aggregate tweets, and average threat existence scores. The tweet with the maximum threat severity score is shown in each instance.

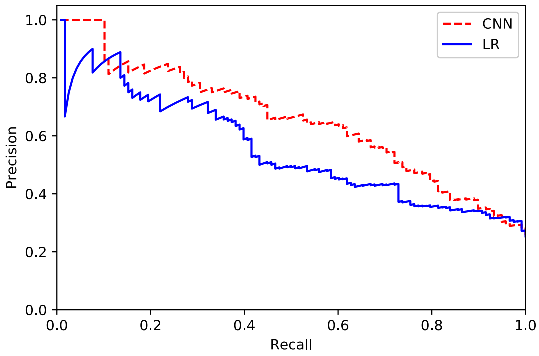


Figure 3: Precision/Recall curves showing performances of convolutional model (CNN) and logistic regression model (LR) for threat severity classification task in test set.

Features	Weight	Features	Weight
ddos attack	1.40	⟨TARGET⟩ ,	0.89
hackers to	1.11	take over	0.87
a massive	1.07	00 countries	0.85
critical vulnerability	1.03	attackers to	0.84
0 billion	0.96	discovered in	0.82
lets attackers	0.95	000 million	0.82
⟨TARGET⟩ users	0.91	: #ddos	0.81
a critical	0.91	abuse and	0.81
of a	0.89	, ddos	0.81
many ⟨TARGET⟩	0.89	a severe	0.79

Table 4: High-weight n-gram features from logistic regression model for threat severity classification task.

Task	Model	Dev AUC	Test AUC
Existence	LR	0.88	0.85
Severity	LR	0.62	0.54
	CNN	0.70	0.65

Table 5: Performance of our threat existence and severity classifiers. We show area under the precision-recall curve (AUC) for both development and test sets.

on their severity?

A large corpus of users' opinions: We follow the same procedure described in Section 2.1 to prepare another dataset for a large-scale evaluation. For this purpose, we collected data from Jan 2016 to Nov 2017; this ensures no tweets overlap with those that were annotated in Section 2.2. We collect all English tweets that explicitly contain the keyword "vulnerability" within this time period, which results in a total number of 976,180 tweets. 377,468 tweets remain after removing tweets without named entities.

National Vulnerability Database (NVD): NVD is the U.S. government database of software vulnerabilities. Started in 2000, NVD covers over 100,000 vulnerabilities, assigning a unique CVE number for each threat. These CVE numbers serve as common identifiers. NVD uses the Common Vulnerability Scoring System (CVSS) to measure the severity of threats. CVSS currently has two versions: CVSS v2.0 and CVSS v3.0 standards. CVSS v3.0 is the latest version released in July 2015. We summarize the two standards in Table 6.[9]

Severity	Base Score	Severity	Base Score
		None	0.0
Low	0.0-3.9	Low	0.1-3.9
Medium	4.0-6.9	Medium	4.0-6.9
High	7.0-10.0	High	7.0-8.9
		Critical	9.0-10.0

Table 6: Qualitative severity rankings of vulnerabilities in NVD. (Left) CVSS v2.0 standards and (Right) CVSS v3.0 standards.

Matching tweets with NVD records: Evaluating our forecasts of high severity vulnerabilities

---

[9] https://nvd.nist.gov/vuln-metrics/cvss

relies on accurately matching tweets describing vulnerabilities to their associated NVD records. To achieve this we present a simple, yet effective method that makes use of content in linked webpages. We find that 82.4% of tweets contain external urls in our dataset.

Our approach to link tweets to CVEs is to search for CVE numbers either in url addresses or in corresponding web pages linked in tweets reporting vulnerabilities.[10] We ignore web pages that contain more than one unique CVE to avoid potential ambiguities. Using this approach, within our dataset, 79,383 tweets were linked to 10,565 unique CVEs. In order to stimulate a forecasting scenario, we only consider CVEs where more than two associated tweets were posted at least 5 days ahead of official NVD publication date. In our dataset, 13,942 tweets are finally selected for forecast evaluation, covering 1,409 unique CVE numbers. To evaluate the accuracy of this linking procedure, we randomly sampled 100 matched pairs and manually checked them. We find the precision of our matching procedure to be very high: only 2 mismatches out of 100 are found.

3.1 Forecasting Models

Now that we have a linking between tweets and CVE numbers, our goal is to produce a sorted list of CVEs with those that are indicated to be severe threats the top. We consider two ranking procedures, detailed below; the first is based on users' opinions toward the severity of a threat, and the second is a baseline that simply uses the volume of tweets describing a specific vulnerability to measure its severity. To simplify the exposition below, we denote each CVE number as CVE_i, and the collection of tweets linked to this CVE number as $T_{\text{CVE}_i} = \{k | \text{tweet } t_k \text{ is mapped to } \text{CVE}_i\}$.

Our model: Our severe threat classifier assigns a severity score $p_{\text{severity}}(y|\langle e, t\rangle)$ for each tuple of name entity e and corresponding tweet t. For a specific CVE, we define our severity forecast score to be the maximum severity scores among all tuples from matched tweets $\langle \cdot, t_k\rangle$ (a single tweet may contain more than one name entity):

$$(\text{CVE}_i)_{\text{forecast score}} = \max_{k \in T_{\text{CVE}_i}} p_{\text{severity}}(y|\langle \cdot, t_k\rangle).$$

Tweet volume baseline: Intuitively, the number of tweets and retweets can indicate people's concern about a specific event. Specifically, the severity for threat CVE_i according to the volume model is defined by the cardinality of T_{CVE_i}:

$$(\text{CVE}_i)_{\text{volume score}} = |T_{\text{CVE}_i}|.$$

3.2 Forecasting CVSS Ratings

In our first set of experiments, we compare our forecasted threat severity scores against CVSS ratings from the NVD. We define a threat as being severe if its CVSS score is ≥ 7.0. This cut-off corresponds to qualitative severity ratings provided by CVSS (marked as HIGH or CRITICAL in Table 6).[11] We use the newest v3.0 scoring system, which was developed to improve v2.0.[12] Large software vendors have announced of the adaptation of the CVSS v3.0 standards, including Cisco, Oracle, SUSE Linux, and RedHat.

We evaluate our models' performance at identifying severe threats five days ahead of the NVD publication date, within their top k predictions. Table 7 shows our results. We observe that tweet volume performs better than a random baseline; having a large number of tweets beforehand is a good indicator for high severity, however our approach which analyzes the content of messages discussing software vulnerabilities achieves significantly better performance; 86% of its top 50 forecasts were indeed rated as HIGH or CRITICAL severity in the NVD.

	P@10	P@50	P@100	AUC
Random	59.0	61.2	58.8	0.595
Volume model	70.0	68.0	70.0	0.583
Our model	100.0	86.0	78.0	0.658

Table 7: Model performance of identifying severe threats (CVSS scores $\geq$ 7.0) with Precision@k and area under the precision-recall curve (AUC) metrics. For majority random baseline, we average over 10 trails.

---

[10] Readers may be wondering why a CVE number has been generated before it is officially published in the database. This is due to the mechanism of assigning CVEs. Some identified companies have the right to assign CVEs or have already reversed some CVEs. When a threat appears, a CVE number is assigned immediately before any further evaluation. NVD only officially publishes a threat after all evaluations are completed. Therefore, there is a time delay between CVE entry established date and the official publication date.

[11] The Forum of Incident Response and Security Teams (FIRST) also provides an example guideline that recommends patching all vulnerabilities with CVSS scores $\geq$ 7.0. See https://www.first.org/cvss/cvss-based-patch-policy.pdf.

[12] https://www.first.org/cvss/user-guide

	CVE Num / Name Entity	CVE Description / Matched Tweets	CVSS Scores / Our Severity	Publish Date (# Days Ahead)
(a)	CVE-2016-0728	The join_session_keyring function in security/keys/process_keys.c in the Linux kernel before 4.4.1 mishandles object references in a certain error case, which allows local users to gain privileges or cause a denial of service (integer overflow and use-after-free) via crafted keyctl commands.	7.2 HIGH (v2.0) 7.8 HIGH (v3.0)	2016-02-08
	Android	Vulnerability in the Linux kernel could allow attackers to gain access to millions of Android devices! http://thenextweb.com/insider/2016/01/20/newly-discovered-security-flaw-could-let-hackers-control-66-of-all-android-devices/ ...	0.98	2016-01-20 (+19)
	Android	A Serious Vulnerability in the Linux Kernel Hits Millions of PCs, Servers and Android Devices http://ift.tt/1OvB4JA	0.89	2016-01-20 (+19)
	Android	Millions of PCs and Android devices are at risk from a recently discovered critical zero-day vulnerability. http://goo.gl/r95ZYZ #infosec	0.89	2016-01-20 (+19)
(b)	CVE-2017-6753	A vulnerability in Cisco WebEx browser extensions for Google Chrome and Mozilla Firefox could allow an unauthenticated, remote attacker to execute arbitrary code with the privileges of the affected browser on an affected system.	9.3 HIGH (v2.0) 8.8 HIGH (v3.0)	2017-07-25
	Cisco WebEx Extensions	The Hacker News : Critical RCE Vulnerability Found in Cisco WebEx Extensions, Again - Patch Now! http://ow.ly/gR3l30dJXlj #CDTTweets	0.98	2017-07-19 (+6)
	Cisco Systems	A critical vulnerability has been discovered in the Cisco Systems' WebEx browser extension for #Chrome and #Firefox: http://s.cgvpn.net/Zu	0.94	2017-07-18 (+7)
	Cisco WebEx Extensions	"Critical RCE Vulnerability Found in Cisco WebEx Extensions, Again - Patch Now!" via The Hacker News #security http://ift.tt/2va8Wrx	0.93	2017-07-17 (+8)
(c)	CVE-2016-5195	Race condition in mm/gup.c in the Linux kernel 2.x through 4.x before 4.8.3 allows local users to gain privileges by leveraging incorrect handling of a copy-on-write (COW) feature to write to a read-only memory mapping, as exploited in the wild in October 2016, aka "Dirty COW."	7.2 HIGH (v2.0) 7.8 HIGH (v3.0)	2016-11-10
	Linux	Serious Dirty COW bug leaves millions of Linux users vulnerable to attack: A vulnerability discovered in the ... http://tinyurl.com/zjdp268	0.97	2016-10-22 (+19)
	Linux OS	A critical vulnerability has been discovered in all versions of the Linux OS and is being exploited in the wild http://ift.tt/2es31Xc	0.95	2016-10-25 (+16)
	Linux COW	Serious vulnerability found in the Linux COW, may have persisted for a decade. http://www.bbc.co.uk/news/technology-37728010?ocid=socialflow_twitter ... http://arstechnica.com/security/2016/10/most-serious-linux-privilege-escalation-bug-ever-is-under-active-exploit/ ...	0.82	2016-10-21 (+20)
(d)	CVE-2016-7855	Use-after-free vulnerability in Adobe Flash Player before 23.0.0.205 on Windows and OS X and before 11.2.202.643 on Linux allows remote attackers to execute arbitrary code via unspecified vectors, as exploited in the wild in October 2016.	10.0 HIGH (v2.0) 9.8 CRITICAL (v3.0)	2016-11-01
	Flash	ICYMI Critical vulnerability found in Flash, being actively exploited. Patch Flash NOW https://www.grahamcluley.com/patch-flash/	0.97	2016-10-27 (+5)
	Adobe	Adobe has released a Flash Player update to patch a critical vulnerability that malicious actors have been ex... http://bit.ly/2eaTxhO	0.95	2016-10-26 (+6)
	Adobe Flash Player	A critical vulnerability for Adobe Flash Player that allows an attacker to take control of the affected system. https://helpx.adobe.com/security/products/flash-player/apsb16-36.html ...	0.80	2016-10-27 (+5)

Table 8: Top 4 threats identified by our forecast model. Severity scores are generated by using threat severity classifier in Section 2.3.

Table 8 presents top 4 forecast results from our model. We observe that our model can predict accurate severity level even 19 days ahead of the official published date in NVD (Table 8(a), (c)).

3.3 Predicting Real-World Exploits

In addition to comparing our forecasted severity scores against CVSS, as described above, we also explored several alternatives suggested by the security community to evaluate our methods: (1) Symantec's anti-virus (AV) signatures[13] and intrusion-protection (IPS) signatures,[14] in addition to (2) Exploit Database (EDB).[15]

Sabottke et al. (2015) suggested Symantec's AV and IPS signatures are the best available indicator for real exploitable threats in the wild. We follow their method of explicitly querying for CVE numbers from the descriptions of signatures to generate exploited threats ground truth. Exploit Database (EDB) is an archive of public exploits and software vulnerabilities. We query EDB for all threats that have been linked into NVD.[16] In total we gathered 134 CVEs verified by Symantec and EDB to be real exploits within the 1,409 CVEs used in our forecasting evaluation.

We evaluate the number of exploited threats identified within our top ranked CVEs. Table 9 presents our results. We observe that 7 of top 10 threats from our model were exploited in the wild. We also observe that for the actual CVSS v3.0 scores, only 1 out of the top 10 vulnerabilities was

---

[13] https://www.symantec.com/security-center/a-z

[14] https://www.symantec.com/security_response/attacksignatures/

[15] https://www.exploit-db.com/

[16] http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html

exploited.

	Top 10		Top 50		Top 100	
	P	R	P	R	P	R
True CVSS	10.0	0.7	16.0	6.0	16.0	11.9
Volume model	60.0	4.5	22.0	8.2	19.0	14.2
Our model	70.0	5.2	28.0	10.4	21.0	15.7

Table 9: Model performance against real-world exploited threats identified by Symantec and Exploit-DB. "True CVSS" refers to ranking CVEs based on actual CVSS scores in NVD. This model is only for reference and can not be used in real practice, as we do not know true CVSS scores when forecasting.

3.4 Identifying Accounts that Post Reliable Warnings

Finally we perform an analysis of the reliability of individual Twitter accounts. We evaluate all accounts with more than 5 tweets exceeding 0.5 confidence score from our severity classifier. Table 10 presents our results. Accounts in our data whose warnings were found to have highest precision when compared against CVSS include "@securityaffairs" and "@EduardKovacs", which are known to post security related information, and both have more than 10k followers.

Account Name	# Corr / # Fcst	Acc. (%)
jburnsconsult	15 / 15	100
securityaffairs	10 / 10	100
EduardKovacs	6 / 6	100
cripperz	5 / 5	100
cipherstorm	4 / 5	80

Table 10: List of users with top accuracies on forecasting severe cybersecurity threats.

4 Related Work

There is a long history of prior work on analyzing users' opinions online (Wiebe et al., 2004), a large body of prior work has focused on sentiment analysis (Pang et al., 2002; Rosenthal et al., 2015), e.g., determining whether a message is positive or negative. In this paper we developed annotated corpora and classifiers to analyze users' opinions toward the severity of cybersecurity threats reported online, as far as we are aware this is the first work to explore this direction.

Forecasting real-world exploits is a topic of interest in the security community. For example, Bozorgi et al. (2010) train SVM classifiers to rank the exploitability of threats. Several studies have also predicted CVSS scores from various sources including text descriptions in NVD (Han et al., 2017; Bullough et al., 2017).

Prior work has also explored a variety of forecasting methods that incorporate textual evidence (Smith, 2010), including the use of Twitter message content to forecast influenza rates (Paul et al., 2014), predicting the propagation of social media posts based on their content (Tan et al., 2014) and forecasting election outcomes (O'Connor et al., 2010; Swamy et al., 2017).

5 Conclusion

In this paper, we presented the first study of the connections between the severity of cybersecurity threats and language that is used to describe them online. We annotate a corpus of 6,000 tweets describing software vulnerabilities with authors' opinions toward their severity, and demonstrated that our corpus supports the development of automatic classifiers with high precision for this task. Furthermore, we demonstrate the value of analyzing users' opinions about the severity of threats reported online as an early indicator of important software vulnerabilities. We presented a simple, yet effective method for linking software vulnerabilities reported in tweets to Common Vulnerabilities and Exposures (CVEs) in the National Vulnerability Database (NVD). Using our predicted severity scores, we show that it is possible to achieve a Precision@50 of 0.86 when forecasting high severity vulnerabilities, significantly outperforming a baseline that is based on tweet volume. Finally we showed how reports of severe vulnerabilities online are predictive of real-world exploits.

Acknowledgments

We thank our anonymous reviewers for their valuable feedback. We also thank Tudor Dumitraş for helpful discussion on identifying real exploited threats. Funding was provided by the the Office of the Director of National Intelligence (ODNI) and Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C0114, in addition to the Defense Advanced Research Projects Agency (DARPA) via the U.S. Army Research Office (ARO) and under Contract Number W911NF-17-C-0095, in addition to an Amazon Research Award and an NVIDIA GPU grant. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation here on.

References

Luca Allodi and Fabio Massacci. 2012. A preliminary analysis of vulnerability scores for attacks in wild: The ekits and sym datasets. In *Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, BADGERS '12, pages 17–24, New York, NY, USA. ACM.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2010. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 105–114, New York, NY, USA. ACM.

Benjamin L. Bullough, Anna K. Yanchenko, Christopher L. Smith, and Joseph R. Zipkin. 2017. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, IWSPA '17, pages 45–53, New York, NY, USA. ACM.

Nathanael Chambers, Ben Fry, and James McMasters. 2018. Detecting denial-of-service attacks from social media text: Applying nlp to computer security. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1626–1635.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 400–410.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Zhuobing Han, Xiaohong Li, Zhenchang Xing, Hongtao Liu, and Zhiyong Feng. 2017. Learning to predict severity of software vulnerability using only vulnerability description. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 125–136.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics.

Michael J Paul, Mark Dredze, and David Broniatowski. 2014. Twitter improves influenza forecasting. *PLOS Currents Outbreaks*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th International Conference on World Wide Web*, pages 896–905. International World Wide Web Conferences Steering Committee.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*.

Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 1041–1056, Washington, D.C. USENIX Association.

Noah A. Smith. 2010. *Text-driven forecasting*.

Thamar Solorio, Ragib Hasan, and Mainul Mizan. 2013. A case study of sockpuppet detection in wikipedia. In *Proceedings of the Workshop on Language Analysis in Social Media*.

Sandesh Swamy, Alan Ritter, and Marie-Catherine de Marneffe. 2017. " i have a feeling trump will win..................": Forecasting winners and losers from user predictions on twitter. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. In *ACL*.

Wei Wei, Kenneth Joseph, Wei Lo, and Kathleen M Carley. 2015. A bayesian graphical model to discover latent events from twitter. In *ICWSM*.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Deyu Zhou, Liangyu Chen, and Yulan He. 2014. A simple bayesian modelling approach to event extraction from twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

A Linking Algorithm

We describe our approach to match tweets with NVD records in full detail in Algorithm 1.

B Limitations of CVSS and Real-World Exploits Ground Truth

In Section 3.2 - Section 3.3, we compare our forecast results with (1) CVSS ratings, and (2) real exploited threats identified by Symantec signatures and Exploit Database. Each of these sources of ground truth have limitations, which we discuss below.

Algorithm 1 Linking tweets to NVD records.

```
// Linking
for every tweet $t$ do
    if CVE number in tweet context or in url links then
        match CVEs to this tweet
    else
        query webpage contents to search for CVEs

// Check linking results
Keep tweets that matched to only one unique CVE to avoid ambiguities

// Apply time constraints
Select out tweets that are posted at least 5 days ahead of official NVD publication date (at most 365 days[17])
```

CVSS ratings are widely used as standard indicators for risk measurement in practice. However, one problem of CVSS ratings is that high severity threats do not necessarily lead to real-world exploits. Allodi and Massacci (2012) show that only a small portion (around 2%) of reported vulnerabilities were found to be exploited in the wild. Furthermore, more than half of the threats in NVD are marked as HIGH or CRITICAL, causing a large burden on vendors to fix.[18] We also notice these CVSS scores are closely tied with specific categories of threats. For example, 85.6% of buffer errors are marked as HIGH or CRITICAL, while 72.5% of information leaks were marked as MEDIUM or LOW. All these issues post challenges on how to prioritize real exploitable threats, with the goal of reducing false positives and false negatives simultaneously. Our work provides one such additional source of information for helping to prioritize threats.

The ground truth we use for real exploited threats is still an incomplete list. For example, Linux kernel vulnerabilities are less likely to appear in Symantec signatures, as Symantec does not have a security product for Linux. Identifying real exploited threats is a difficult task; to the best of our knowledge, there does not exist an easy-to-access list covering all exploited threats currently.

C Additional Analysis of Results

In this section, we present further analyses of people's online behaviors when discussing cybersecurity threats on social media.

We find that the real severity of threats is pre-

---

[17] If a tweet or its associate urls explicitly contains a CVE number, then we ignore this maximum time range constraint.

[18] https://www.riskbasedsecurity.com/2017/05/cvssv3-when-every-vulnerability-appears-to-be-high-priority/

	CVE Num	Name Entity	Tweet	Our Score	Real Severity
(a)	CVE-2017-4984	EMC VNX1VNX2 OE	threatmeter: Vuln: EMC VNX1/VNX2 OE for File CVE-2017-4984 Remote Code Execution Vulnerability http://ift.tt/2rWXQXa	0.01	10.0 HIGH (v2.0) 9.8 CRITICAL (v3.0)
(b)	CVE-2016-1730	iPhone	A newly discovered vulnerability may expose iPhone users to attack when using a Wi-Fi hotspot - via @InfosecurityMag http://owl.li/Xw3VO	0.76	5.8 MEDIUM (v2.0) 5.4 MEDIUM (v3.0)
		iPhone	Apple iOS Flaw Enables Attacks via Hotspot: The vulnerability opens up iPhone users to a raft of problems, inc... http://bit.ly/1JqGtD9	0.45	

Table 11: Some examples of forecast errors made by our model. (a) False negative examples: there is no clear language clue for demonstrating the severity of threats, experts are needed for threats of this kind. (b) False positive examples: there exist some signals captured by our model for being severe threats, but actual severity might be overestimated.

Adj.	Ratio	Adj.	Ratio	Adj.	Ratio
serious	2.01	aware	1.61	fast	1.39
pivotal	1.95	most	1.61	original	1.39
sure	1.95	vivid	1.61	able	1.39
free	1.95	accessible	1.39	blind	1.39
active	1.79	popular	1.39	arbitrary	1.35
intelligent	1.79	deep	1.39	high	1.30
static	1.79	black	1.39	incomplete	1.25
critical	1.67	top	1.39	malicious	1.20
severe	1.61	dangerous	1.39	wily	1.10
great	1.61	wild	1.39	evil	1.10

Table 12: Top ranked log-odds ratio of subjective adjectives describing severe threats (CVSS scores ≥ 7.0) versus non-severe threats (CVSS scores < 7.0). Subjective adjectives are identified by using Subjectivity Lexicon (SUB) (Wilson et al., 2005).

dictable based on users' opinions online. We observe several repeated patterns in how people describe severe threats. We summarize some of these patterns below:

- describing severity levels (see Appendix C.1), such as "critical", "serious", "highly";
- describing the number of users or devices affected, such as "millions of ⟨TARGET⟩ devices", "huge number of";
- potential consequences, such as "allows hackers to", "could allow for remote code execution", "malware";
- alerts or warnings, such as "please be aware", "warning";
- suggesting immediate actions, such as "patch now".

C.1 Usage of Subjective Adjectives

We notice people rely on adjectives for describing the level of severity for threats, rather than numerical scores. These subjective adjectives form our initial impressions on these threats.

We examine subjective adjectives people use for measuring threats. We run POS tagging to extract all tokens marked as `JJ`, `JJP`, and `JJS`. We then rank subjective adjectives in Subjectivity Lexicon (SUB) (Wilson et al., 2005) by log-odds ratio of their occurrences in NVD descriptions for HIGH or CRITICAL threats versus MEDIUM or LOW threats. Table 12 presents top ranked subjective adjectives. We observe variants people are using for severe threats, e.g. "serious", "severe", "malicious", etc.

C.2 Temporal Analysis

We collect all CVEs having matched tweets posted at least 1 day ahead of the official NVD publication date, resulting in a set of 3,678 CVEs. Within our dataset, 84.7% of CVEs are reported within 60 days after the first disclosure on social media. We observe a median of 5 days delay in our dataset, whereas some of threats have significant longer delays. For example, CVE-2016-2123[19] (Overflow Remote Code Execution Vulnerability) first appears at Twitter on Dec. 19, 2016[20], but is published in NVD on Nov. 1, 2018. It again shows the difficulty of threat evaluation and management.

C.3 Error Analysis

We evaluate two types of errors with respect to forecasting high severity vulnerabilities: false positive and false negative examples. We observe that some severe threats are difficult to predict based on contents in general, such as Table 11(a). There is no clear clue for estimating the severity level merely on tweet contents.

We present another incorrect example extracted by our forecast system in Table 11(b). We notice tokens like "expose users to attack", "opens up to a raft of problems", etc. This threat does seem to be exploitable and harmful to a lot of users. However, experts mark it as of medium severity. It might be the case that the actual severity level of some threats are overestimated by some accounts.

---

[19] https://nvd.nist.gov/vuln/detail/CVE-2016-2123

[20] https://twitter.com/ryf_feed/status/810981102768758784

Fake News Detection using Deep Markov Random Fields

Duc Minh Nguyen[⋆†], **Tien Huu Do**[⋆†], **Robert Calderbank**[‡], **Nikos Deligiannis**[⋆†],
[⋆]Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
[†]imec, Kapeldreef 75, B-3001 Leuven, Belgium
[‡]Duke University, Durham, North Carolina
[⋆†]`{mdnguyen, thdo, ndeligia}@etrovub.be`
[‡]`robert.calderbank@duke.edu`

Abstract

Deep-learning-based models have been successfully applied to the problem of detecting fake news on social media. While the correlations among news articles have been shown to be effective cues for online news analysis, existing deep-learning-based methods often ignore this information and only consider each news article individually. To overcome this limitation, we develop a graph-theoretic method that inherits the power of deep learning while at the same time utilizing the correlations among the articles. We formulate fake news detection as an inference problem in a Markov random field (MRF) which can be solved by the iterative mean-field algorithm. We then unfold the mean-field algorithm into hidden layers that are composed of common neural network operations. By integrating these hidden layers on top of a deep network, which produces the MRF potentials, we obtain our deep MRF model for fake news detection. Experimental results on well-known datasets show that the proposed model improves upon various state-of-the-art models.

1 Introduction

The term “fake news” refers to news articles that is intentionally and verifiably false (Shu et al., 2017). The problem of fake news has existed since the appearance of the printing press, but only gained a lot of momentum and visibility during the age of social media. This is due to the large audience, easy access and fast dissemination mechanism of social media, where more and more users are consuming news in a daily basis (Shu et al., 2017). Traditional methods for verifying the veracity of news that rely on human experts, despite being reliable, do not scale well to the massive volume of news nowadays. This renders the automatic detection of fake news on social media an important problem, drawing a lot of attention from both the academic and industrial communities.

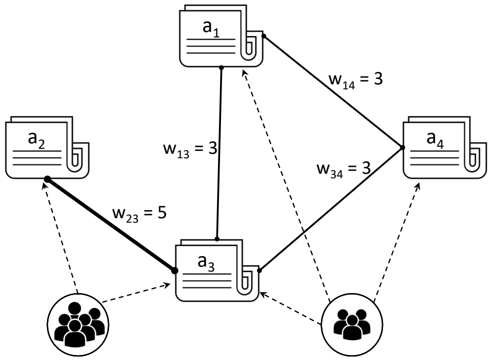


Figure 1: Modeling the relationship among news articles (or events): the dash lines represent engagements of users to articles, the solid lines represent the relationships between articles with weights determined by the number of common engaged users.

The recent literature has witnessed the success of deep learning models in detecting fake news on social media (Ma et al., 2016; Yu et al., 2017; Ruchansky et al., 2017; Rashkin et al., 2017; Ma et al., 2018a; Kochkina et al., 2018). By leveraging the capability of deep networks in learning high-level representations, these models have achieved state-of-the-art performance on various benchmark datasets. Nevertheless, one limitation of existing deep-learning-based methods is that they often ignore the correlations among news articles, which have been proved to be effective for analysing online events (Freire et al., 2016; Fairbanks et al., 2018).

To overcome this limitation, we aim at a model that leverages the capability of deep neural networks while effectively incorporating the correlations among articles when determining their credibility. To this end, we first model the relationship between two articles by the number of the com-

Proceedings of NAACL-HLT 2019, pages 1391–1400
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

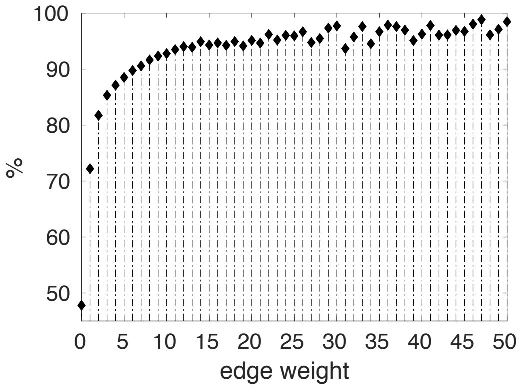


Figure 2: Percentage of edges having certain weights, which connects two articles with the same labels on the news graph constructed from the Weibo dataset (Ma et al., 2016).

mon users that engage to them, e.g., by means of tweeting, re-tweeting, commenting. An example is illustrated in Fig. 1: the articles a_2 and a_3 have a strong relationship as they are engaged to by 5 common users, whereas, there is no relationship between the articles a_1 and a_2 because there is no user engage to both of them. With this modeling, we can construct a *news graph*, where each node corresponds to an article and an edge encodes the relationship between two articles.

Our underlying assumption is that if there exists a strong relationship between two articles, they are likely to share the same labels. To verify this assumption, we calculate the percentages of the edges that connect two articles with the same labels, among those whose weights are equal to certain values, and plot the results for the news graph constructed from the Weibo dataset (Ma et al., 2016) in Fig. 2. It is clear from Fig. 2 that the higher the edge weight, the more likely it is that the corresponding articles share the same labels (*fake* or *true*). Similar patterns are observed on other datasets such as the Twitter (Ma et al., 2016) and the PHEME (Zubiaga et al., 2017) datasets. Evidently, these results support our assumption.

In order to incorporate the correlations among news articles, we formulate fake news detection as an inference problem in a Markov random field (MRF). Our motivation behind this formulation is to leverage the capability of MRF in capturing dependencies among random variables. We solve the resulting inference problem using the mean-field algorithm (Koller and Friedman, 2009). We then propose a method to unfold this algorithm into hidden layers that can be integrated on top of a deep network that computes the potentials of the MRF. By doing this, we obtain our deep MRF model for detecting fake news, referred to as the DMFN model. To the best of our knowledge, this is the first integration of deep networks and MRF for detecting fake news. Our main contributions are as follows:

- We formulate fake news detection as an inference problem in an MRF model. This allows us to incorporate the correlations among news articles when deciding their credibility.
- We propose a method to unfold the mean-field algorithm into specially-designed neural network layers, and build a deep MRF model for detecting fake news.
- We carry out comprehensive experiments on widely-used datasets collected from popular social networks. Experimental results show the effectiveness of the proposed model compared to various state-of-the-art models.

The remainder of the paper is organized as follows: we review the related work in Section 2, and describe our formulation of fake news detection as an inference problem MRF in Section 3. In Section 4, we describe our model in detail. We present our experimental studies in Section 5 and finally draw the conclusions in Section 6.

2 Related Work

Early work in fake news detection focused on finding a good set of features that are useful for separating fake news from genuine news. Linguistic patterns, such as special characters, specific keywords and expression types, have been explored to spot fake news (Castillo et al., 2011; Liu et al., 2015; Zhao et al., 2015). Different feature types have also been considered, such as the characteristics of users involved in spreading the news, e.g. the number of followers, the users' ages and genders (Castillo et al., 2011; Yang et al., 2012), and the news' propagation patterns (Castillo et al., 2011; Kwon et al., 2013). Instead of relying on a single feature type, existing works normally made use of multiple types of feature at the same time.

Recent years have witnessed the use of deep learning for fake news detection. The idea is to leverage deep neural networks to overcome the limitation of the *shallow* hand-craft features employed in the earlier methods (Ma et al., 2016).

Many works proposed to represent news articles as multivariate time series using the timestamp information and to formulate fake news detection as a sequence classification problem (Ma et al., 2016; Yu et al., 2017; Ma et al., 2018a; Liu and Brook Wu, 2018; Kochkina et al., 2018). As it is common in the literature to utilize multiple types of features in detecting fake news, deep networks with multiple branches to incorporate various feature types were also proposed (Ruchansky et al., 2017; Volkova et al., 2017; Yang et al., 2018). In general, deep learning based methods yield higher accuracy compared to shallow-feature-based approaches, leading to state-of-the-art performance.

The existing deep-learning models, however, often ignore the correlations among news articles which haven been shown to be effective cues for analysing online news and events (Freire et al., 2016; Fairbanks et al., 2018). Freire et al. proposed a method to detect breaking news on Wikipedia by exploring the graph of related events (Freire et al., 2016). This graph is built by connecting any pair of pages on Wikipedia which were edited by the same users in a small time window. In (Fairbanks et al., 2018), a graph of news was constructed by connecting the corresponding web pages using the links between them in form of `html` tags. The credibility of the news was assessed by applying the loopy belief propagation algorithm to perform semi-supervised learning on this graph. In their experiments, the authors showed that the correlations among the news, encoded in the constructed graph, were more effective than the textual content of the news for predicting their credibility.

In this work, we propose a deep MRF model for fake news detection. Apart from leveraging the power of deep networks, our model incorporates the correlations among news articles when determining their credibility. In this regard, our method shares a similar motivation with the graph-based methods for social event analysis as mentioned above. Nevertheless, while these graph-based methods do not utilize any information of the news articles beyond the labels, our model is more generic and allows incorporating an arbitrary number of features.

3 Fake News Detection on MRF

We employ a Markov random field (MRF) model (Freeman et al., 2000) to incorporate the correlations among the events due to its capability in capturing dependencies among random variables. To this end, we first construct an event graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ the set of vertices and $\mathcal{E}$ the set of edges as described in Section 1. We then define an MRF model over $\mathcal{G}$. In this model a node k is associated with a random variable X_k, which represents the label of the k-th event. The random variables $X_k, \forall k \in \{1, \dots, n\}$ have domain $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2 \dots \mathcal{L}_s\}$, which represent the s possible labels. For notation brevity, we refer to the nodes in $\mathcal{V}$ by their indices, namely, $k \in \{1, \dots, n\}$. We are interested in inferring the distribution $P(X)$ of the MRF, from which the labels for the events can be obtained. In the MRF, the probability $P(X = \mathrm{x})$, with x a set of values of the random variables, is given by (Koller and Friedman, 2009)

$$P(X = \mathrm{x}) = \frac{1}{Z} \exp(-E(\mathrm{x})), \tag{1}$$

with Z the partition function ensuring a valid distribution and $E(\mathrm{x})$ the energy of the MRF, which has the following form

$$E(\mathrm{x}) = \sum_{k \in \mathcal{V}} \Phi\left(\mathrm{x}_k^u\right) + \lambda \sum_{k,l \in N} \Psi\left(\mathrm{x}_k^u, \mathrm{x}_l^v\right). \tag{2}$$

In (2), N is the set of nodes that are connected in the MRF . The *unary potential*, $\Phi\left(\mathrm{x}_k^u\right)$, measures the cost of assigning the label $\mathcal{L}_u$ to the node k, while the *pairwise potential*, $\Psi(\mathrm{x}_k^u, \mathrm{x}_l^v)$, measures the cost of assigning the nodes k and l respectively the labels $\mathcal{L}_u$ and $\mathcal{L}_v$. As such, the pairwise potentials capture the dependencies among the nodes of the MRF. λ is a hyperparameter.

As exactly computing $P(X)$ is intractable, we employ the mean-field algorithm (Koller and Friedman, 2009) to approximate $P(X)$ by a fully-factorized proposal distribution $Q(X) = \prod_{\forall k \in \mathcal{V}} Q_k(X_k)$. $Q_k(X_k = \mathcal{L}_u)$ is the probability that the node k have the label $\mathcal{L}_u$ according to the distribution Q_k. Denote $Q_k(X_k = \mathcal{L}_u)$ as q_k^u, the mean-field algorithm iteratively calculates $q_k^u, \forall k \in \{1, \dots, n\}$, $u \in \{1, \dots, s\}$ according to (Koller and Friedman, 2009)

$$\begin{aligned} q_k^u = \frac{1}{Z_k} \exp \Bigg\{ - \Bigg(& \Phi\left(\mathrm{x}_k^u\right) \\ & + \lambda \sum_{l \in N_k} \sum_{v \in \mathcal{L}} q_l^v \Psi\left(\mathrm{x}_k^u, \mathrm{x}_l^v\right) \Bigg) \Bigg\}, \end{aligned} \tag{3}$$

with N_k the set of nodes connected to k and $Z_k = \sum_{u \in \mathcal{L}} q_k^u$ the normalization factor ensuring $q_k^u, \forall u \in \{1, \dots, s\}$, add up to 1. The generic mean-field update in (3) requires the unary potentials and pairwise potentials. In Section 4, we show how we realize these potentials.

4 Deep MRF Model for Fake News Detection

In this section, we present our realizations of the unary and pairwise potentials, with which we obtain the final mean-field update equation. After that, we present a method to unfold the mean-field update into specially-designed neural network layers, and describe our deep MRF model for fake news detection.

4.1 Computing the Unary and Pairwise Potentials

We compute the unary potential $\Phi(\mathrm{x}_k^u)$ of the MRF as the negative log likelihood:

$$\Phi(\mathrm{x}_k^u) = -\ln p(X_k = \mathcal{L}_u), \tag{4}$$

with $p(X_k = \mathcal{L}_u)$ the likelihood that X_k has label $\mathcal{L}_u$. As such, $\Phi(\mathrm{x}_k^u)$ will be high if this node is not likely to have the label $\mathcal{L}_u$, and vice versa. The likelihood $p(X_k = \mathcal{L}_u)$ is computed as $p(X_k = \mathcal{L}_u) = \mathcal{F}_\theta(\mathcal{O}_k)$, where $\mathcal{F}_\theta$ is a non-linear function represented by a deep neural network parameterized by θ and O_k is the observations, i.e, features associated to the k-th event. The design of this network is described later in Section 4.

The pairwise potential is computed by

$$\Psi(\mathrm{x}_k^u, \mathrm{x}_l^v) = a(k,l) \times \mu(u,v), \tag{5}$$

where $a(k,l)$ is the weight of the edge between the nodes k and l, namely, $a(k,l)$ represents how strong the relationship between the nodes k and l is; and $\mu(u,v)$ is the *label compatibility*, which is calculated using the Pott model (Boykov et al., 1998):

$$\mu(u,v) = \begin{cases} 1, & \text{if } u \neq v, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Substituting the pairwise potential in (5) into (3), we obtain the final mean-field update equation:

$$q_k^u = \frac{1}{Z_k} \exp \Bigg\{ -\Phi\left(x_k^u\right) \\ - \lambda \sum_{l \in N_k} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu\left(u,v\right) \Bigg\}. \tag{7}$$

We refer to the term $\sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v)$ in (7) as the *compatibility transform* step, and to the term $\sum_{l \in N_k} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu\left(u,v\right)$ summing up information from the nodes connected to the node k as the *message passing* step.

4.2 Mean-Field Updates as Neural Network Layers

We now describe a method to implement the update equation in (7) using operations that are common in the deep learning literature. Abusing the notation, we denote also by $Q \in \mathbb{R}^{n \times s}$ a matrix, with an entry $Q_{k,u}$ corresponding to the value q_k^u, i.e., the probability that the node k has the label $\mathcal{L}_u$. Denote by $A \in \mathbb{R}^{n \times n}$ the adjacency matrix, containing all the edge weights in $\mathcal{G}$. We set $A_{k,k} = 0, \forall k \in \{1, \dots, n\}$. We further denote by $M \in \mathbb{R}^{s \times s}$ a square matrix whose the entry $M_{u,v}$ corresponds the label compatibility $\mu(u,v)$ calculated using (6), and by $\Phi \in \mathbb{R}^{n \times s}$ the matrix containing the unary potentials $\Phi(\mathrm{x}_k^u)$, for all $k \in \{1, \dots, n\}$ and $u \in \{1, \dots, s\}$. Φ is calculated by taking the logarithm of Q element-wise.

4.2.1 The Compatibility Transform Step

The compatibility transform in (7) can be performed via a 1-D convolutional layer applied on the matrix Q. This convolutional layer has s filters of kernel size $1 \times s$. The weights of the u-th filter are set equal to the values along the u-th row of M. We do not employ any padding and set the stride to 1. When applying this operation, the u-th filter slides vertically across Q, and calculates the inner product between its weights and the rows of Q. The output is denoted as $\mathcal{Q}' \in \mathbb{R}^{n \times s}$, with an entry $Q'_{l,u}$ given by

$$\begin{aligned} Q'_{l,u} &= \sum_{v \in \mathcal{L}} Q_{l,v} \times M_{u,v} \\ &= \sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v), \end{aligned} \tag{8}$$

for all $l \in \{1, \dots, n\}$, $u \in \{1, \dots, s\}$, which is equal to the result of the compatibility transform step in (7).

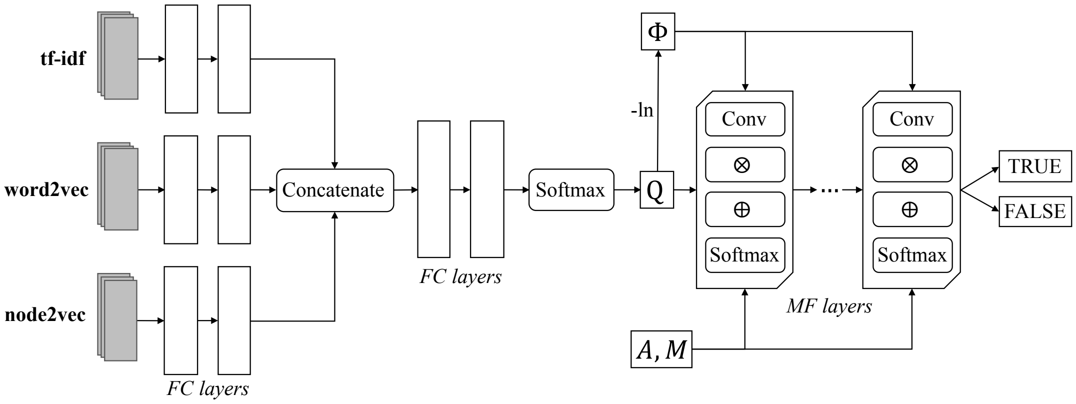


Figure 3: The architecture of the DMFN model: the first block consists of three feature branches, each with configurable numbers of fully-connected (FC) layers to process a type of feature, a concatenation layer and a number of FC layers with softmax activation function at the end to produce labels' probability matrix Q; Each *mean-field* (MF) layer consists of four main operations, namely convolution (conv), matrix multiplication ($\otimes$), point-wise addition ($\oplus$) and the softmax function. T MF layers are stacked one after another to implement the mean-field algorithm with T iterations. All MF layers share the same set of parameters, namely, the matrix of the unary potentials Φ, which is computed from Q, the adjacency matrix A encoding the relationships among the input events, and the compatibility matrix M.

4.2.2 The Message Passing Step

Given the adjacency matrix $A \in \mathbb{R}^{n \times n}$ and the output $Q^{'} \in \mathbb{R}^{n \times s}$ from the compatibility transform step, the message passing step can be performed simply by multiplying A with $Q^{'}$. This multiplication results in $Q^{''} \in \mathbb{R}^{n \times s}$, in which:

$$\begin{aligned} Q^{''}_{k,u} &= \sum_{l=1}^{n} A_{k,l} . Q^{'}_{l,u} \\ &= \sum_{l=1}^{n} a(k,l) \sum_{v \in \mathcal{L}} q_l^v \times \mu(u,v). \end{aligned} \tag{9}$$

As $A_{k,l} = 0$ if $l \notin N_k$, the operation in (9) is equivalent to the message passing step in (7).

4.2.3 The Mean-field Layer

To finish the update in (7), one needs to element-wise multiply $Q^{''}$ with λ, add Φ and negate the result. The exponential and normalization can be performed jointly using the *softmax* function, resulting in the new matrix $Q \in \mathbb{R}^{n \times s}$ whose entries correspond to the values of q_k^u after one mean-field iteration, for all $k \in \{1, \ldots, n\}$, $u \in \{1, \ldots, s\}$. We can consider these operations together with those implementing the compatibility transform and the message passing steps, as the operations of specially-designed a neural network layer, which we call the mean-field (MF) layer. As all the component operations are differentiable, the operation implemented by the MF layer is also differentiable. Each MF layer, hence, implements one iteration of the mean-field algorithm. Clearly, by stacking T MF layers, we can implement the mean-field algorithm with T iterations.

4.3 The DMFN Model

4.3.1 Model Architecture

The architecture of the DMFN model is illustrated in Figure 3, with two blocks: the first block corresponds to the deep network $\mathcal{F}_\theta$ that produces the unary potentials, and the second block is composed of T MF layers, which implements the T-iteration mean-field algorithm. The deep network $\mathcal{F}_\theta$ takes as inputs several feature types extracted to represent the given set of events. Each feature type is processed by a feature branch with a number of fully-connected (FC) layers. The outputs of the last layers in all feature branches are concatenated to produce high-level representations for all the events. These representations are then fed to another set of FC layers and a softmax function to produce the label probabilities. These label probabilities form the matrix Q, as described in Section 4.2. All the FC layers in the model are followed by a batch normalization layer (Ioffe and Szegedy, 2015), a ReLU activation function (Glorot et al., 2011), and the dropout regularization (Srivastava et al., 2014).

In the second block, all the MF layers share the same parameters, namely, the matrix Φ, the adjacency matrix A that encodes the relationships among the given events, and the label compatibility matrix M. The first MF layer takes as input the matrix Q, whereas, the later MF layers operate on the outputs from their previous MF layers. The output after the last MF layer is the final label probabilities predicted for the given events. It should be noted that the parameters of the MF layers are pre-computed and shared. As such, adding MF layers after the layers in the first block does not increase the the risk of overfitting of the model.

4.3.2 Feature Extraction

We extract multiple types of features to capture the characteristics of the events, namely, the textual contents and social engagements. For the textual features, we first group all the tweets related to an event into a document. We pre-process the documents by removing stop words, converting the words into lower case and tokenizing them. From the pre-processed documents, we extract the *term frequency - inverse document frequency* (tf-idf) (Wu et al., 2008), and *word2vec* (Mikolov et al., 2013) feature vectors. We use the word2vec model pre-trained on the Google News dataset to map each word in a document to a $300-$dimensional embedding vector, and then obtain the embedding for the document by taking the average of the word embeddings.

We use a graph embedding technique to capture the social engagements associated to the events. Concretely, we first build a graph of users from the given dataset. Two users are connected by an edge if they engage to at least one event in common, with the edge's weight determined by the total number of common engaged events. We then employ the *node2vec* algorithm (Grover and Leskovec, 2016) to learn an embedding for each user. The social engagement feature of an event is then calculated by taking the average of the embeddings of all users engaged to it. It is worth noting that the graph of users used in this step is different from the graph of events, constructed as described in Section 1.

4.3.3 Training and Testing the DMFN Model

We employ the weighted cross entropy loss function to train the DMFN model. When calculating the loss, the weight given to the training samples of a particular label is inversely proportional to the number of samples in the current batch which have this label. This technique is highly beneficial when dealing with imbalanced dataset, e.g., the PHEME dataset (Zubiaga et al., 2016). The model's parameters are learned by using the SGD algorithm with the Adam parameter update (Kingma and Ba, 2015).

At the testing stage, we select for the testing event k the label $\mathcal{L}_{\hat{u}} \in \mathcal{L}$ with $\hat{u}$ determined by $\hat{u} = \arg\max_{u \in \{1,\dots,s\}} q_k^u$. As we want to utilize the correlations among events in the prediction, we provide an adjacency matrix encoding the relationships among a set of events and run a forward pass with all their features as input. Without the adjacency matrix or when setting the adjacency matrix to the identity matrix, the model predicts the labels for each events without considering their correlations.

5 Experiments

5.1 Experimental Settings

We employ three well-known benchmark datasets, namely the Twitter, Weibo (Ma et al., 2016) and PHEME datasets (Zubiaga et al., 2017) for our experiments. The Twitter dataset consists of 992 events, involving 233.7 thousand users and 592.4 thousand tweets. The Weibo dataset is larger, with $4,664$ events, 2.8 million users and 3.8 million posts. Events in these datasets are labeled as either *fake* or *true*, and the two labels are relatively balanced. The PHEME dataset consists of $5,802$ comment threads collected from Twitter, with approximately 103 thousand tweets in total. This dataset is imbalanced, with $1,972$ threads labeled as *rumour* and $3,830$ threads labeled as *non-rumour*.

5.2 Hyperparameter Selection

For the DMFN model, we employ one hidden layer in each feature branch, and one hidden layer after the concatenation layer, all with 100 hidden units. We train the model for maximum 100 epochs with learning rate 0.001 and stop training early if the validation loss does not improve over the average of those of the previous 25 epochs. To control overfitting, we employ dropout with high dropping rate of 0.9.

We determine the values of λ which balance the weight between the unary and pairwise potentials in the MRF, and of the number of MF layers, T by cross validation on a separate split on the Weibo dataset. First, we fix T to 30, with which the mean-field algorithm is highly likely to converge (Krähenbühl and Koltun, 2011), and experi-

λ	0.005	0.050	0.100	0.500
Weibo	0.958	**0.960**	0.959	0.949

Table 1: Accuracy of the DMFN model when varying λ on the Weibo datasets (Ma et al., 2016).

T	1	5	10	15
Weibo	0.949	0.960	0.960	0.960

Table 2: Accuracy of the DMFN model when varying T on the Weibo datasets (Ma et al., 2016).

ment with different values of λ. The result of this is presented in Table 1.

Fixing λ to 0.05, we experiment with different number of MF layers by varying T. The results are summarized in Table 2. As can be seen from the table, employing multiple MF layers improves the results over just one MF layer. Even though we still observe improvements in the performance with more than 5 MF layers, the difference is small. As a result, we select $T = 5$ as it produces the best trade-off between accuracy and computational complexity.

5.3 Experimental Results

We compare the results of different models in two experimental settings, namely *late detection* and *early detection*. The former setting allows the models to use all the available users posts in the entire time span of the given datasets, whereas in the latter setting, the models are only allowed to use posts that have appeared within a specific deadline (in hours) since the appearances of the events.

5.3.1 Late Detection Setting

We compare the results of the proposed models in the late detection setting with those of reference models, including the decision tree classifier (DTC) (Castillo et al., 2011), the SVM classifier (SVM-RBF) (Yang et al., 2012), the random forest classifier (RFC) (Kwon et al., 2013), the SVM classifier with timeseries features (SVM-TS) (Ma et al., 2015), the 2-layer GRU model (GRU-2) (Ma et al., 2016) and the convolutional neural network (CAMI) (Yu et al., 2017), the Tree-structured Recursive Neural Networks (TD-RvNN) (Ma et al., 2018b), and the CRF and Naive Bayes classifiers in (Zubiaga et al., 2017). The performance of the models is assessed using four metrics, namely, accuracy, average precision, average recall and macro F1 score. Similar to (Yu et al., 2017), on the Twitter and Weibo datasets, we randomly reserve 10% of the samples for parameter tuning and perform four-fold cross validation on the remaining. On the PHEME dataset, we follow the leave-one-event-out approach as in (Zubiaga et al., 2017). We report the average results for the models.

The results for different models on the Twitter and the Weibo datasets (Ma et al., 2016) are shown in Table 3. On these two datasets, we do not include the results of the TD-RvNN model (Ma et al., 2018b) as this model requires a tree-like connections among the tweets to represent an event. As can be seen from the table, the DMFN model consistently outperforms the reference models on both datasets. The results of all the models are better on the Weibo dataset than on the Twitter dataset. This is possibly because number of posts available on the Weibo dataset is much larger than that available on the Twitter dataset.

The results on the PHEME dataset is illustrated in Table. 4. As this dataset is imbalanced, the prediction accuracy is not a good metric for comparison. Similar to (Zubiaga et al., 2017), we focus on the other metrics, namely, precision, recall and macro F1 score in the comparison. The CRF model with content features (Zubiaga et al., 2017) yields the highest precision, whereas the Naive Bayes classifiers yield the highest recalls.. While the reference models are either biased toward high precision or high recall scores, the DMFN model produces more balanced precision and recall, and the best Macro F1 score. The DF-RvNN model (Ma et al., 2018a) also achieves balanced precision and recall, nevertheless its performance is lower than that of the DMFN model in all metrics (with p-values of 0.004, 0.04, 0.03 respectively for the precision, recall and macro f1 scores in a pairwise t-test).

On the PHEME dataset, the average number of tweets per event is 17.8, which is much smaller than those on the Twitter and Weibo datasets (805 and 815 posts per event respectively). The lower number of tweets, and the class imbalance render this dataset highly challenging. As such, the performance of all the considered models on this dataset is lower than that on the Twitter and Weibo datasets. Overall on all the considered datasets, we observe consistent performance of the DMFN model: The variance of the Macro F1 score over 10 repetitions are relatively low, which are equal to

Model	Twitter				Weibo			
	Acc.	Prec.	Rec.	Macro F1	Acc.	Prec.	Rec.	Macro F1
SVM-RBF	0.715	0.720	0.710	0.709	0.818	0.819	0.818	0.818
DTC	0.718	0.718	0.718	0.718	0.831	0.831	0.831	0.831
RFC	0.728	0.728	0.728	0.728	0.849	0.866	0.849	0.847
SVM-TS	0.745	0.741	0.741	0.740	0.857	0.859	0.858	0.859
GRU-2	0.757	0.760	0.757	0.771	0.910	0.914	0.910	0.910
CAMI	0.777	0.782	0.777	0.776	0.933	0.933	0.933	0.933
DMFN	**0.800**	**0.803**	**0.803**	**0.799**	**0.962**	**0.963**	**0.962**	**0.970**

Table 3: Results for different models on the Twitter and Weibo datasets (Ma et al., 2016).

Model	Prec.	Rec.	Macro F1
Naive Bayes Content	0.309	**0.723**	0.433
CRF Content	**0.683**	0.545	0.606
Naive Bayes Content + Social	0.310	**0.723**	0.434
CRF Content + Social	0.667	0.556	0.607
TD-RvNN	0.616	0.612	0.609
DMFN	0.667	0.670	**0.657**

Table 4: Results for different models on the PHEME datasets (Zubiaga et al., 2017).

Model	Twitter		Weibo		PHEME	
	Acc.	Macro F1	Acc.	Macro F1	Acc.	Macro F1
DMFN-base	0.763	0.762	0.944	0.944	0.682	0.607
DMFN-separate	0.789	0.788	0.958	0.959	0.690	0.641
DMFN	**0.800**	**0.799**	**0.962**	**0.970**	**0.703**	**0.657**

Table 5: Results of different variants of the DMFN model on the three datasets.

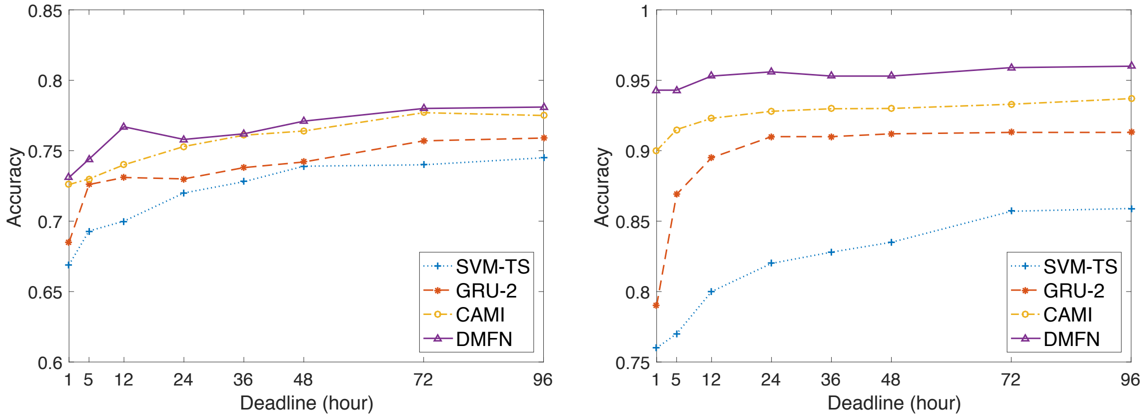
1e−4, 2e−4 and 6e−5 respectively on the Twitter, Weibo and PHEME datasets.

5.3.2 Early Detection Setting

We perform the early detection experiments on the Twitter and Weibo datasets with different deadlines, in $\{1, 5, 12, 24, 36, 48, 72, 96\}$ (hours). Fig. 4a and Fig. 4b illustrate the results on the Twitter and the Weibo datasets, respectively. As can be seen, on both datasets, the DMFN model performs the best among the selected models, followed by the CAMI model. On the Weibo dataset, the average number of posts per event is approximately 168, 353, 497 within the 1-hour, 5-hour and 12-hour deadlines respectively.These figures suggest that Weibo users are responsive and quickly react to a newly broadcasted event. The large number of posts per event, even at 1-hour deadline, gives enough information for the DMFN, as well as the CAMI models to produce good results, even within short deadlines.

5.3.3 The Effects of Jointly Training the Deep Network with Mean-field Inference

An advantage of the proposed model is that it allows training the deep network that produces the unary potentials with feedback from the mean-field inference. To verify this argument, we compare the performance when using different variants of the proposed model: (*i*) training and testing without the MF inference, (*ii*) training without the MF inference and testing with the MF inference, and (*iii*) training and testing with the MF inference (the full DMFN model). We denote the three variants as *DMFN-base*, *DMFN-separate* and *DMFN*. As can be seen from the table, applying the MF inference improves the results of the base model even if it has been trained without the MF inference. This proves the benefits of enforcing the dependencies between the events in the MRF when making predictions. Nevertheless, training and testing with MF inference consistently yields the best results among the three variants. This proves



(a) Twitter dataset (b) Weibo dataset

Figure 4: Accuracy of different models when considering only posts within specific deadlines on the Twitter and Weibo datasets.

the benefits of unfolding the MF inference and incorporate it on top of the base network.

6 Conclusion

We formulated the fake news detection on social media as an inference problem in an MRF model that can be solved using the mean-field algorithm. By translating each update step in this algorithm into common operations in the deep learning literature, we can unfold it into hidden layers that can be integrated on top of another deep neural network. This results in our deep MRF model (DMFN) for detecting fake news. As such, the DMFN carries the advantages of both deep neural networks in learning high-level representations, and of MRF in incorporating correlations among the news articles. Experiments on well-known benchmark datasets show that the proposed model consistently improves over the state of the art in fake news detection in both the late and early detection settings.

Acknowledgments

The authors acknowledge the financial support from the Vrije Universiteit Brussel (PhD bursary - Duc Minh Nguyen), the Fonds Wetenschappelijk Onderzoek Vlaanderen (FWO) and the Francqui Foundation (2016-2017 International Francqui Chair - Robert Calderbank).

References

Yuri Boykov, Olga Veksler, and Ramin Zabih. 1998. Markov random fields with efficient approximations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–655.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *ACM International Conference on World Wide Web (WWW)*, pages 675–684.

James Fairbanks, Natalie Fitch, Nathan Knauf, and Erica Briscoe. 2018. Credibility assessment in the news : Do we need to read ? In *Misinformation and Misbehavior Mining on the Web Workshop (MIS2)*, pages 1–8.

William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. 2000. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47.

Ana Freire, Matteo Manca, Diego Saez-Trumper, David Laniado, Ilaria Bordino, Francesco Gullo, and Andreas Kaltenbrunner. 2016. Graph-based breaking news detection on wikipedia. In *AAAI Conference on Web and Social Media (ICWSM) Workshops*, pages 1–2.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *International Conference on Computational Linguistics (COLING)*.

Daphne Koller and Nir Friedman. 2009. In *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Philipp Krähenbühl and Vladlen Koltun. 2011. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, pages 109–117.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *IEEE International Conference on Data Mining (ICDM)*, pages 1103–1108.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1867–1870.

Yang Liu and Yi-Fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI Conference on Artificial Intelligence*, pages 1–8.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Joint Conference on Artificial Intelligence (IJCAI)*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1751–1754.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018a. Detect rumor and stance jointly by neural multi-task learning. In *The Web Conference (WWW)*, pages 585–593.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018b. Rumor detection on twitter with tree-structured recursive neural networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1980–1989.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2931–2937.

Natali Ruchansky, Sungyong Seo, and Yan Lin. 2017. CSI : A hybrid deep model for fake news detection. In *ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 797–806.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explorations Newsletter*, 19(1):22–36.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15:1929–1958.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 647–653.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting TF-IDF term weights as making relevance decisions. *ACM Transactions on Information Systems*, 26(3):13:1–13:37.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *ACM SIGKDD Workshop on Mining Data Semantics*, pages 13:1–13:7.

Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Xiaoming Zhang, Zhoujun Li, and Philip S. Yu. 2018. TI-CNN : Convolutional neural networks for fake news detection. *ArXiv e-prints 1806.00749*.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2017. A convolutional approach for misinformation identification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3901–3907.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *ACM International Conference on World Wide Web (WWW)*, pages 1395–1405.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *Social Informatics*, pages 109–123.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE*, 11(3):1–29.

Issue Framing in Online Discussion Fora

Mareike Hartmann[1] **Tallulah Jansen**[2] **Isabelle Augenstein**[1] **Anders Søgaard**[1]

[1]Dep. of Computer Science, University of Copenhagen, Denmark
{hartmann,augenstein,soegaard}@di.ku.dk
[2]Inst. of Cognitive Science, Osnabrück University, Germany
taljansen@uni-osnabrueck.de

Abstract

In online discussion fora, speakers often make arguments for or against something, say birth control, by highlighting certain aspects of the topic. In social science, this is referred to as *issue framing*. In this paper, we introduce a new issue frame annotated corpus of online discussions. We explore to what extent models trained to detect issue frames in newswire and social media can be transferred to the domain of discussion fora, using a combination of multi-task and adversarial training, assuming only unlabeled training data in the target domain.

1 Introduction

The *framing* of an issue refers to a choice of perspective, often motivated by an attempt to influence its perception and interpretation (Entman, 1993; Chong and Druckman, 2007). The way issues are framed can change the evolution of policy as well as public opinion (Dardis et al., 2008; Iyengar, 1991). As an illustration, contrast the statement *Illegal workers depress wages* with *This country is abusing and terrorizing undocumented immigrant workers*. The first statement puts focus on the economic consequences of immigration, whereas the second one evokes a morality frame by pointing out the inhumane conditions under which immigrants may have to work. Being exposed to primarily one of those perspectives might affect the publics attitude towards immigration.

Computational methods for frame classification have previously been studied in news articles (Card et al., 2015) and social media posts (Johnson et al., 2017). In this work, we introduce a new benchmark dataset, based on a subset of the 15 generic frames in the *Policy Frames Codebook* by Boydstun et al. (2014). We focus on frame classification in *online discussion fora*, which have become crucial platforms for public dialogue on social and political issues. Table 1 shows example annotations, compared to previous annotations for news articles and social media. Dialogue data is substantially different from news articles and social media, and we therefore explore ways to transfer information from these domains, using multi-task and adversarial learning, providing non-trivial baselines for future work in this area.

Platform: Online discussions
Economic Frame, Topic: Same sex marriage `But as we have seen, supporting same-sex marriage saves money.`
Legality Frame, Topic: Same sex marriage `So you admit that it is a right and it is being denied?`
Platform: News articles
Economic Frame, Topic: Immigration `Study Finds That Immigrants Are Central to Long Island Economy`
Legality Frame, Topic: Same sex marriage `Last week, the Iowa Supreme Court granted same-sex couples the right to marry.`
Platform: Twitter
Legality Frame, Topic: Same sex marriage `Congress must fight to ensure LGBT people have the full protection of the law everywhere in America. #EqualityAct`

Table 1: Example instances from the datasets described in §2 and 3.

Contributions We present a new issue-frame annotated dataset that is used to evaluate issue frame classification in online discussion fora. Issue frame classification was previously limited to news and social media. As manual annotation is expensive, we explore ways to overcome the lack of labeled training data in the target domain with

Proceedings of NAACL-HLT 2019, pages 1401–1407
Minneapolis, Minnesota, June 2 - June 7, 2019.

Frames	`1`	`13`	`5`	`6`	`7`
# instances	78	96	234	166	186

Table 2: Class distribution in the online discussion test set. The frame labels correspond to the classes *Economic* (`1`), *Political* (`13`), *Legality, Jurisprudence and Constitutionality* (`5`), *Policy prescription and evaluation* (`6`) and *Crime and Punishment* (`7`).

multi-task and adversarial learning, leading to improved results in the target domain.[1]

Related Work Previous work on automatic frame classification focused on news articles and social media. Card et al. (2016) predict frames in news articles at the document level, using clusters of latent dimensions and word-based features in a logistic regression model. Ji and Smith (2017) improve on previous work integrating discourse structure into a recursive neural network. Naderi and Hirst (2017) use the same resource, but make predictions at the sentence level, using topic models and recurrent neural networks. Johnson et al. (2017) predict frames in social media data at the micro-post level, using probabilistic soft logic based on lists of keywords, as well as temporal similarity and network structure. All the work mentioned above uses the generic frames of Boydstun et al. (2014)'s Policy Frames Codebook. Baumer et al. (2015) predict words perceived as frame-evoking in political news articles with hand-crafted features. Field et al. (2018) analyse how Russian news articles frame the U.S. using a keyword-based cross-lingual projection setup. Tsur et al. (2015) use topic models to analyze issue ownership and framing in public statements released by the US congress. Besides work on frame classification, there has recently been a lot of work on aspects closely related to framing, such as subjectivity detection (Lin et al., 2011), detection of biased language (Recasens et al., 2013) and stance detection (Mohammad et al., 2016; Augenstein et al., 2016; Ferreira and Vlachos, 2016).

2 Online Discussion Annotations

We create a new resource of issue-frame annotated online fora discussions, by annotating a subset of the Argument Extraction Corpus (Swanson et al., 2015) with a subset of the frames in the Policy Frames Codebook. The Argument Extraction Corpus is a collection of argumentative dialogues across topics and platforms.[2] The corpus contains posts on the following topics: *gay marriage*, *gun control*, *death penalty* and *evolution*. A subset of the corpus was annotated with argument quality scores by Swanson et al. (2015), which we exploit in our multi-task setup (see §3).

We collect new issue frame annotations for each argument in the argument-quality annotated data.[3] We refer to this new issue-frame annotated corpus as *online discussion corpus* henceforth. Each argument can have one or multiple frames. Following Naderi and Hirst (2017), we focus on the five most frequent issue frames: *Economic*, *constitutionality and jurisprudence*, *policy prescription and evaluation*, *law and order/crime and justice*, and *political*. See Table 1 for examples and Table 2 for the class distribution in the resulting online discussions test set. Phrases which do not match the five categories are labeled as *Other*, but we do not consider this class in our experiments. The annotations were done by a single annotator. A second annotator labeled a subset of 200 instances that we use to compute agreement as macro-averaged F-score, assuming one of the annotations as gold standard. Results are 0.73 and 0.7, respectively. The averaged Cohen's Kappa is 0.71.

3 Additional Data

The dataset described in the previous section serves as evaluation set for the online discussions domain. As we do not have labeled training data for this domain, we exploit additional corpora and additional annotations, which are described in the next subsection. Statistics of the filtered datasets as well as preprocessing details are given in Appendix A.

Media Frames Corpus The Media Frames Corpus (Card et al., 2015) contains US newspaper articles on three topics: *Immigration*, *smoking* and *same-sex marriage*. The articles are annotated with the 15 framing dimensions defined in the Policy Frames Codebook.[4] The annotations are on

---

[1] Code and annotations are available at `https://github.com/coastalcph/issue_framing`.

[2] The corpus is a combination of dialogues from `http://www.createdebate.com/`, and Walker et al. (2012)'s Internet Argument Corpus, which contains dialogues from `4forums.com`.

[3] Topic cluster *Evolution* was dropped, because it contained too few examples matching our frame categories.

[4] We discard all instances that do not correspond to the frame categories in the online discussions data.

Model	Task	Domain	Labelset	# classes	# sequences
Baseline	Main task	News articles	Frames	5	10,480
	Target task	Online disc. (test)	Frames	5	692
Multitask	+Aux task	Tweets	Frames	5	1,636
	+Aux task	Online disc.	Argument quality	2	3,785
Adversarial	+Adv task	Online disc. + News articles	Domain	2	4,731 + 10,480
		Online disc. (dev)	Frames	5	176

Table 3: Overview over the data and labelsets for the different tasks. The baseline model trains on the main task and predicts the target task. The multi-task model uses one or both auxiliary tasks in addition to the main task. The adversarial model uses the adversarial task in addition to the main task. All models use the online disc. dev set for model selection.

span-level and can cross sentence boundaries. We convert span annotations to sentence-level annotations as follows: if a span annotated with label l lies within sentence boundaries and covers at least 50% of the tokens in the sentence, we label the sentence with l. We only keep sentence annotations if they are indicated by at least two annotators.

Congressional Tweets Dataset The congressional tweets dataset (Johnson et al., 2017) contains tweets authored by 40 members of the US Congress, annotated with the frames of the Policy Frames Codebook. The tweets are related to one or two of the following six issues: *abortion*, *the Affordable Care Act*, *gun rights vs. gun control*, *immigration*, *terrorism*, and *the LGBTQ community*, where each tweet is annotated with one or multiple frames.

Argument Quality Annotations The corpus of online discussions contains additional annotations that we exploit in the multi-task setup. Swanson et al. (2015) sampled a subset of 5,374 sentences, using various filtering methods to increase likelihood of high quality argument occurrence, and collected annotations for argument quality via crowdsourcing. Annotators were asked to rate argument quality using a continuous slider [0-1]. Seven annotations per sentence were collected. We convert these annotations into binary labels (1 if $\geq$ 0.5, 0, otherwise) and generate an approximately balanced dataset for a binary classification task that is then used as an auxiliary task in the multi-task setup. Balancing is motivated by the observation that balanced datasets tend to be better auxiliary tasks (Bingel and Søgaard, 2017).

4 Models

The task we are faced with is (multi-label) sequence classification for online discussions. However, we have no labeled training data (and only a small labeled validation set) for the target task in the target domain. Hence, we train our model on a dataset which is labeled with the target labels, but from a different domain. The largest such dataset is the news articles corpus, which we consequently use as main task. Our baseline model is a two-layer LSTM (Hochreiter and Schmidhuber) trained on only the news articles data. We then apply two strategies to facilitate the transfer of information from source to target domain, multi-task learning and adversarial learning. We briefly describe both setups in the following. An overview over tasks and data used in the different models is shown in Table 3.

Multi-Task Learning To exploit synergies between additional datasets/annotations, we explore a simple multi-task learning with hard parameter sharing strategy, pioneered by Caruana (1993), introduced in the context of NLP by Collobert et al. (2011), and to RNNs by Søgaard and Goldberg (2016), which has been shown to be useful for a variety of NLP tasks, e.g. sequence labelling (Rei, 2017; Ruder et al., 2019; Augenstein and Søgaard, 2017), pairwise sequence classification (Augenstein et al., 2018) or machine translation (Dong et al., 2015). Here, parameters are shared between hidden layers. Intuitively, it works by training several networks in parallel, tying a subset of the hidden parameters so that updates in one network affect the parameters of the others. By sharing parameters, the networks regularize each other, and the network for one task can benefit from repre-

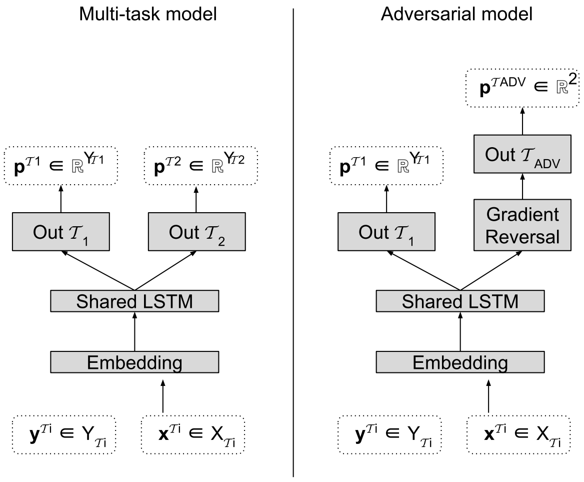


Figure 1: Overview over the multi-task model (left) and the adversarial model (right). The baseline LSTM model corresponds to the same architecture with only one task.

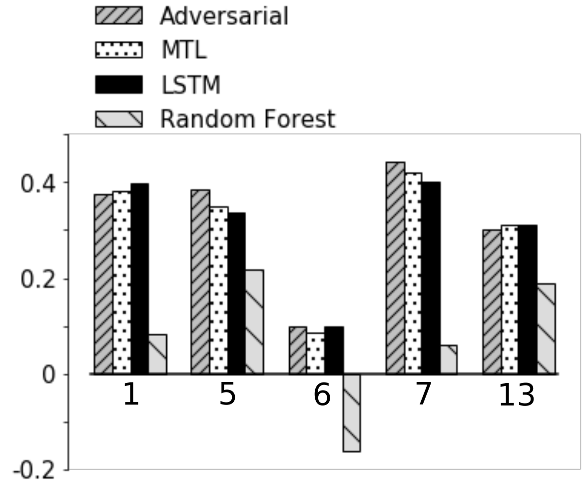


Figure 2: Improvement in F-score over the random baseline by class. The absolute F-scores for the best performing system for classes 1, 5, 6, 7, and 13, are 0.529, 0.625, 0.298, 0.655, and 0.499, respectively.

sentations induced for the others.

Our multi-task architecture is shown in Figure 1. We have N different datasets $\mathcal{T}_1, \cdots, \mathcal{T}_N$. Each dataset $\mathcal{T}_i$ consists of tuples of sequences $x^{\mathcal{T}_i} \in X_{\mathcal{T}_i}$ and labels $y^{\mathcal{T}_i} \in Y_{\mathcal{T}_i}$. A model for task $\mathcal{T}_i$ consists of an input layer, an LSTM layer (that is shared with all other tasks) and a feed forward layer with a softmax activation as output layer. The input layer embeds a sequence $x^{\mathcal{T}_i}$ using pretrained word embeddings. The LSTM layer recurrently processes the embedded sequence and outputs the final hidden state h. The output layer outputs a vector of probabilities $p^{\mathcal{T}_i} \in R^{Y_{\mathcal{T}_i}}$, based on which the loss $\mathcal{L}_i$ is computed as the categorical cross-entropy between prediction $p^{\mathcal{T}_i}$ and true label $y^{\mathcal{T}_i}$. In each iteration, we sample a data batch for one of the tasks and update the model parameters using stochastic gradient descent. If we sample a batch from the main task or an auxiliary task is decided by a weighted coin flip.

Adversarial Learning Ganin and Lempitsky (2015) proposed adversarial learning for domain adaptation that can exploit unlabeled data from the target domain. The idea is to learn a classifier that is as good as possible at assigning the target labels (learned on the source domain), but as poor as possible in discriminating between instances of the source domain and the target domain. With this strategy, the classifier learns representations that contain information about the target class but abstract away from domain-specific features. During training, the model alternates between 1) predicting the target labels and 2) predicting a binary label discriminating between source and target instances. In this second step, the gradient that is backpropagated is flipped by a Gradient-Reversal layer.[5] Consequently, the model parameters are updated such that the classifier becomes worse at solving the task. The architecture is shown in the right part of Figure 1. In our implementation, the model samples batches from the adversarial task or the main task based on a weighted coinflip.

5 Experiments

We compare the multi-task learning and the adversarial setup with two baseline models: (a) a Random Forest classifier using tf-idf weighted bag-of-words-representations, and (b) the LSTM baseline model. For the multi-task model, we use both the Twitter dataset and the argument quality dataset as auxiliary tasks. For all models, we report results on the test set using the optimal hyper-parameters that we found averaged over 3 runs on the validation set. For the neural models, we use 100-dimensional GloVe embeddings (Pennington et al., 2014), pre-trained on Wikipedia and Gigaword.[6] Details about hyper-parameter tuning and optimal settings can be found in Appendix B.

Results The results in Table 5 show that both the multi-task and the adversarial model improve over

---

[5] In the forward pass, this layer multiplies its input with the identity matrix.

[6] `https://nlp.stanford.edu/projects/glove/`

Nr.	Gold	Adv	MTL	LSTM	Sentence
(1)	5	5	5	7	But, star gazer, we had guns then when the Constitution was written and enshrined in the BOR and now incorporated into th 14th Civil Rights Amendment.
(2)	6	6	5	1	Gun control is about preventing such security risks.
(3)	7	7	5	1	First, you warn me of the dangers of using violent means to stop a crime .
(4)	5	6	6	6	So I don't see restrictions on handguns in D.C. as being a clear violation of the Second Amendment.

Table 4: Examples for model predictions on the online discussion dev set. The first column shows the gold label and the following columns the prediction made by the adversarial model (Adv), the Multi-Task model (MTL) and the LSTM baseline (LSTM).

Model	P_{ma}	R_{ma}	F_{ma}	F_{mi}
Random Baseline	0.196	0.198	0.189	0.196
Random Forest Baseline	0.496	0.335	0.267	0.279
LSTM Baseline	0.512	0.510	0.503	0.521
Multi-Task	0.526	0.525	0.505	0.534
Adversarial	**0.533**	**0.534**	**0.515**	**0.548**

Table 5: Macro- (ma) and micro-averaged (mi) scores for the online discussion test data averaged over 3 runs. The multi-task model uses the Twitter and argument quality datasets as auxiliary tasks. The micro-average F of a baseline that predicts the majority class is 0.307.

the baselines. The multi-task model achieves minor improvements over the LSTM baseline, with a bigger improvement in the micro-averaged score, indicating bigger improvements with frequent labels. The adversarial model performs best, with an error reduction in micro-averaged F over the LSTM baseline of 5.6%.

Figure 2 shows the system performances for each class. Each bar indicates the difference between the F-score of the respective system and the random baseline. The adversarial model achieves the biggest improvements over the baseline for classes 5 and 7, which are the two most frequent classes in the test set (cf. Table 6). For classes 1 and 13, the adversarial model is outperformed by the LSTM. Furthermore, we see that the hardest frame to predict is the *Policy prescription and evaluation frame* (6), where the models achieve the lowest improvement over the baseline and the lowest absolute F-score. This might be because utterances with this frame tend to address specific policies that vary according to topic and domain of the data, and are thus hard to generalize from source to target domain.

Analysis Table 4 contains examples of model predictions on the dialogue dev set. In Example (1), the adversarial and the multi-task model correctly predict a *Constitutionality* frame, while the LSTM model incorrectly predicts a *Crime and punishment* frame. In Examples (2) and (3), only the adversarial model predicts the correct frames. In both cases, the LSTM model incorrectly predicts an *Economic* frame, possibly because it is misled by picking up on a different sense of the terms *means* and *risks*. In Example (4), all models make an incorrect prediction. We speculate this might be because the models pick up on the phrase *restrictions on handguns* and interpret it as referring to a policy, whereas to correctly label the sentence they would have to pick up on the *violation of the Second Amendment*, indicating a *Constitutionality* frame.

6 Conclusion

This work introduced a new benchmark of political discussions from online fora, annotated with issue frames following the Policy Frames Cookbook. Online fora are influential platforms that can have impact on public opinion, but the language used in such fora is very different from newswire and other social media. We showed, however, how multi-task and adversarial learning can facilitate transfer learning from such domains, leveraging previously annotated resources to improve predictions on informal, multi-party discussions. Our best model obtained a micro-averaged F1-score of 0.548 on our new benchmark.

Acknowledgements

We acknowledge the resources provided by CSC in Helsinki through NeIC-NLPL (www.nlpl.eu), and the support of the Carlsberg Foundation and the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of EMNLP*.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-Task Learning of Pairwise Sequence Classification Tasks over Disparate Label Spaces. In *NAACL-HLT*, pages 1896–1906. Association for Computational Linguistics.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-Task Learning of Keyphrase Boundary Classification. In *Proceedings of ACL*.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and Comparing Computational Approaches for Identifying the Language of Framing in Political News. In *Proceedings of HLT-NAACL*, pages 1472–1482. The Association for Computational Linguistics.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of EACL*.

Amber E. Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2014. Tracking the Development of Media Frames within and across Policy Issues. In *Proceedings of APSA*.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The Media Frames Corpus: Annotations of Frames Across Issues. In *Proceedings of ACL*, pages 438–444.

Dallas Card, Justin Gross, Amber Boydstun, and Noah A Smith. 2016. Analyzing Framing through the Casts of Characters in the News. In *Proceedings of EMNLP*, pages 1410–1420.

Richard Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of ICML*, pages 41–48. Morgan Kaufmann.

Dennis Chong and James Druckman. 2007. Framing Theory. *Annual Review of Political Science*, 10.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR*, 999888:2493–2537.

Frank E. Dardis, Frank R. Baumgartner, Amber E. Boydstun, Suzanna de Boef, and Fuyuan Shen. 2008. Media Framing of Capital Punishment and Its Impact on Individuals' Cognitive Responses. *Mass Communication and Society*, 11(2):115–140.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task Learning for Multiple Language Translation. In *Proceedings of ACL*.

Robert M. Entman. 1993. Framing: Toward Clarification of a Fractured Paradigm. *Journal of Communication*, 43(4):51–58.

William Ferreira and Andreas Vlachos. 2016. Emergent: A Novel Data-Set for Stance Classification. In *Proceedings of NAACL HLT*.

Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky, and Yulia Tsvetkov. 2018. Framing and Agenda-setting in Russian News: a Computational Analysis of Intricate Political Strategies. In *Proceedings of EMNLP*, pages 3570–3580. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France. PMLR.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural Computation*.

Shanto Iyengar. 1991. *Is Anyone Responsible? How Television Frames Political Issues*. University of Chicago Press.

Yangfeng Ji and Noah Smith. 2017. Neural Discourse Structure for Text Categorization. In *Proceedings of ACL*.

Kristen Johnson, Di Jin, and Dan Goldwasser. 2017. Leveraging Behavioral and Social Information for Weakly Supervised Collective Classification of Political Discourse on Twitter. In *Proceedings of ACL*.

Chenghua Lin, Yulan He, and Richard Everson. 2011. Sentence Subjectivity Detection with Weakly-Supervised Learning. In *Proceedings of IJCNLP*, pages 1153–1161, Chiang Mai, Thailand.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of SemEval*.

Nona Naderi and Graeme Hirst. 2017. Classifying Frames at the Sentence Level in News Articles. In *Proceedings of RANLP*, pages 536–542.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Daniel Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of ACL*.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of ACL*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Multi-Task Architecture Learning. In *AAAI*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*.

Reid Swanson, Brian Ecker, and Marilyn A. Walker. 2015. Argument Mining: Extracting Arguments from Online Dialogue. In *SIGDIAL Conference*.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A Frame of Mind: Using Statistical Models for Detection of Framing and Agenda Setting Campaigns. In *Proceedings of ACL-IJCNLP*, pages 1629–1638. Association for Computational Linguistics.

Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, pages 812–817. European Language Resources Association (ELRA).

Appendix A Data Preprocessing

For the Twitter and news articles datasets, we remove all instances that do not correspond to the five target frames. Table 6 shows the class distributions in the filtered datasets. We tokenize all sequences using spaCy [7], which we also use for sentence splitting in the news articles dataset. For the Twitter dataset, we follow Johnson et al. (2017) in removing URLs and @-mentions.

Appendix B Hyperparameters in Experiments

The hyperparameters for all neural models were tuned on the online disc. dev set. We report test results for the optimal settings found by averaging over 3 training runs, which we determine by the best macro-averaged F-score and smallest variance between the runs. We set the DyNet weight decay parameter to 1e-7 for all neural models, batch size is 128, and the word embeddings are not updated during training.

For the multi-task and adversarial model, we do a grid-search over the weight of the coin flip used to decide on sampling from main/aux or main/adversarial task in the range of [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]. The optimal weight for sampling the main task is 0.5 for the multi-task model and 0.3 for the adversarial task.

All models are trained using early stopping (after at least 80 epochs of training) with a patience of 5 epochs. The number of iterations (updates) per epoch is a hyperparameter, that we set by default as twice the number of data batches for the main task. For a fair coin flip, the models hence see as much data for the main task as for the auxiliary/adversarial task per epoch.

Dataset	# instances	# instances per class					# multi
		`1`	`13`	`5`	`6`	`7`	
NEWSPAPER (TRAIN)	10,480	1088	1959	2023	924	890	45
TWITTER (TRAIN)	1,636	73	300	137	27	174	554
ONLINE DISC. (TEST)	692	78	96	234	166	186	67
			`0`	`1`			
ARGUMENT QUALITY	3,785		1,350	2,435			0
ONLINE DISC. UNLABELED	4731						

Table 6: Dataset statistics and class distributions. The frame labels correspond to the classes *Economic* (`1`), *Political* (`13`), *Legality, Jurisprudence and Constitutionality* (`5`), *Policy prescription and evaluation* (`6`) and *Crime and Punishment* (`7`). *# multi* refers to the number of multi-label instances. For Argument quality, label 1 indicates a score greater or equal 0.5.

---

[7] `https://spacy.io/`

Vector of Locally Aggregated Embeddings for Text Representation

Hadi Amiri[a] **and Mitra Mohtarami**[b]
[a]Harvard, Department of Biomedical Informatics, Boston, MA, USA
[b]MIT, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
`hadi@hms.harvard.edu`, `mitra@csail.mit.edu`

Abstract

We present **V**ector of **L**ocally **A**ggregated **E**mbeddings (`VLAE`) for effective and, ultimately, lossless representation of textual content. Our model encodes each input text by effectively identifying and integrating the representations of its semantically-relevant parts. The proposed model generates high quality representation of textual content and improves the classification performance of current state-of-the-art deep averaging networks across several text classification tasks.

1 Introduction

Representation learning algorithms can reveal intrinsic low-dimensional structure in data (Rumelhart et al., 1986; Bengio et al., 2013; LeCun et al., 2015). In particular, deep averaging networks (DANs) are effective for text classification (Shen et al., 2018; Arora et al., 2017; Wieting et al., 2016; Iyyer et al., 2015). They achieve their improvement through use of word embeddings, weighted averaging, and deepening networks. The above works show that DANs can outperform RNNs and CNNs in text classification while taking only a fraction of their training time.

In this work, with a special focus on DANs, we study the effect of information loss associated with average word embeddings and develop algorithms that are robust against information loss for text representation. We show that divergence of word embeddings from their average can be considered as a good proxy to quantify information loss; in particular, longer documents suffer from significant information loss when represented by average word embeddings. These results inspire our work to develop a novel representation learning approach based on Vector of Locally Aggregated Descriptors (VLAD) (Jégou et al., 2010; Arandjelovic and Zisserman, 2013)–an effective approach to integrate image descriptors for large scale image datasets. Our model identifies semantically-relevant parts of documents and locally integrates their representations through clustering and autoencoding. In contrast to averaging, our model prevents larger semantically-relevant parts of inputs to dominate final representations. It improves DANs by 5.30 macro-F1 points in classifying longer texts and show comparable performance to them on shorter text.

2 Preliminary Analysis

How can information loss be quantified when word embeddings are averaged? How important it is to address information loss when representing textual content? Are representation learning algorithms robust against information loss? We conduct experiments to answer these questions with respect to deep averaging network (DANs). Our study can inspire works in more complex averaging approaches such as those reported in (Torabi Asr et al., 2018; Kiela et al., 2015) as well as recent works on unsupervised semantic similarity (Pagliardini et al., 2018). We use the DAN developed in (Joulin et al., 2017) and several datasets containing short and long documents to answer these questions.

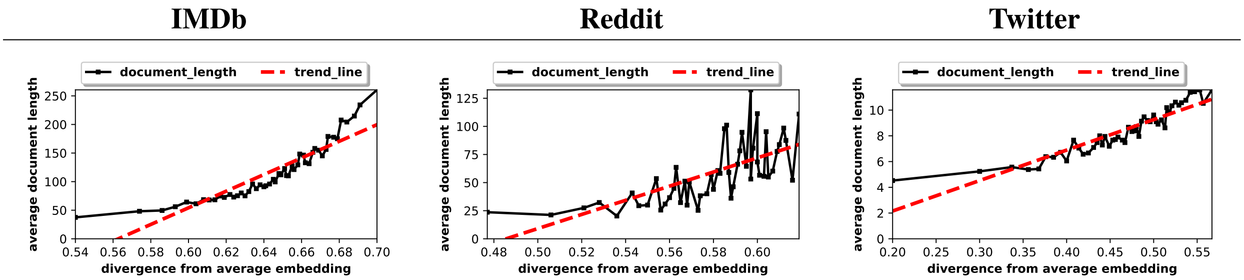
2.1 Quantifying Information Loss

Let's assume a d-dimensional word embedding space. We quantify the amount of information loss in the average word embedding vector of a given document $\mathbf{S} \in \mathbb{R}^{n \times d}$ by computing the average divergence (or distance) between its word embeddings, $\mathbf{w}_i \in \mathbb{R}^d \; \forall i \in \{1 \dots n\}$, and their average vector, $\bar{\mathbf{s}} = 1/n \sum_i \mathbf{w}_i$, $\bar{\mathbf{s}} \in \mathbb{R}^d$, as follows:

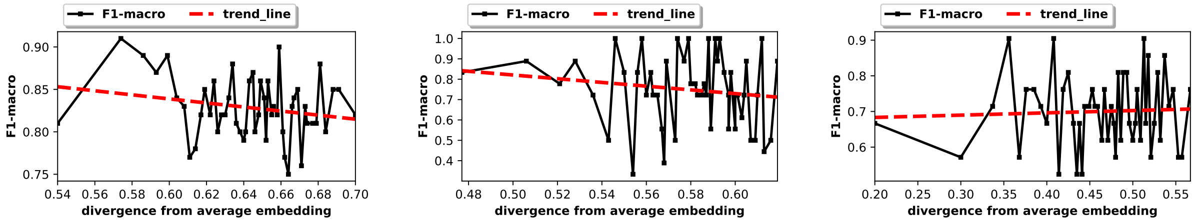
$$divergence = \frac{1}{n} \sum_i (1 - cosine(\bar{\mathbf{s}}, \mathbf{w}_i)). \quad (1)$$

Figures 1(a)–1(c) show strong positive correlation between divergence and document length

Proceedings of NAACL-HLT 2019, pages 1408–1414
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) divergence vs. length (b) divergence vs. length (c) divergence vs. length



(d) divergence vs. F1 (e) divergence vs. F1 (f) divergence vs. F1

Figure 1: Quantification of information loss associated with average word embeddings. Divergence indicates the average distance between individual word embeddings (of size $d = 300$) and their average embedding, see Equation (1). (a-c): show strong positive correlation between divergence and average document length (number of words in a document) across datasets. (d-f): show macro F1 performance of the deep averaging network developed in (Joulin et al., 2017) across datasets: performance considerably drops for higher values of information loss/divergence, e.g. divergence values above 0.55. Note that, we sort and bin instances based on their divergence values and report average length (a-c) and macro-F1 (d-f) for each bin.

across long and short text datasets. Given these results and if we assume longer documents should suffer from greater information loss if represented by average word embeddings, divergence from mean can be a good proxy to quantify information loss associated with average embeddings.

2.2 Effect of Information Loss

As Figures 1(d) and 1(e) show, DAN's macro-F1 classification performance considerably decreases as divergence (or text length) increases for IMDb and Reddit datasets; note that we sort and bin instances based on their divergence values and report average macro-F1 for each bin. In particular, as the trend lines in Figures 1(d) and 1(e) show, the average macro-F1 performance drops from 0.86 and 0.82 on shorter IMDb and Reddit posts to 0.82 and 0.71 on their longer posts respectively. In addition, the result on Twitter dataset, Figure 1(f), shows that DANs are robust against small information loss, i.e. small divergence values below 0.55 do not inversely affect macro-F1 performance. This result is also observed on the other two datasets, see macro-F1 performance for small divergence values (≤ 0.55) in Figures 1(d) and 1(e).

The above experiments show that (a): significant information loss can occur when word embeddings are averaged, in particular, when representing longer documents, and (b) such information loss can inversely affect the performance of downstream classifiers like DANs on longer texts. In this paper, we develop an effective representation learning model to tackle this problem.

3 Method

We propose to utilize semantically-relevant parts of inputs to tackle information loss associated with average word embeddings. Assuming that semantically-relevant words are closer to each other in semantic space (constructed over a global vocabulary), we expect divergence between words in semantically-relevant parts of inputs (i.e. information loss associated to their average word embedding) to be very small. Thus, as Figure 2 illustrates, we propose to cluster the semantic space to first identify semantically-relevant parts of inputs over a global vocabulary; we then effectively integrate these parts to represent documents.

Let's assume a global vocabulary $\mathcal{V}$ in which words are represented in a d-dimensional space, $\mathbf{w} \in \mathbb{R}^d$. As Figure 2 shows, we first cluster this semantic space into k clusters through the following objective function over $\mathcal{V}$:

$$min \sum_{\mathbf{w} \in \mathcal{V}} ||f(\mathcal{C}, \mathbf{w}) - \mathbf{w}||^2, \tag{2}$$

where $\mathcal{C}$ is the set of k cluster centers, $|\mathcal{C}| = k$, and $f(\mathcal{C}, \mathbf{w})$ returns the nearest cluster center $\mathbf{c} \in \mathcal{C}$ to the embedding vector $\mathbf{w}$ based on `cosine` similarity among embeddings or Euclidean distance in case of K-Means.[1]

Given a document $\mathbf{S} \in \mathbb{R}^{n \times d}$ with an arbitrary number of $n \geq 1$ words, and the above k cluster centers, we compute the representation of the document in each cluster $\mathbf{c}_i$, $i = 1 \ldots k$ as follows:

$$\mathbf{a}_i = \frac{1}{z_i} \sum_{j: f(\mathcal{C}, \mathbf{w}_j) = \mathbf{c}_i} \mathbf{w}_j \tag{3}$$

$$z_i = |j : f(\mathcal{C}, \mathbf{w}_j) = \mathbf{c}_i|,$$

where $\mathbf{a}_i \in \mathbb{R}^d$ indicates the representation of the document at cluster $\mathbf{c}_i$ and is obtained by taking the average embedding of words of the document that have been assigned to cluster $\mathbf{c}_i$ according to Equation (2), and z_i is the number of such words in cluster $\mathbf{c}_i$. To this end, each document can be represented by $\mathbf{A} \in \mathbb{R}^{d \times k}$ which is obtained by concatenating its cluster-level representations. Note that we didn't observe any performance difference between the above averaging process versus computing *residuals* (differences between word embeddings and corresponding cluster centroids) which is commonly used to represent cluster-level image descriptors (Jégou et al., 2010; Arandjelovic and Zisserman, 2013) in image processing.

Since $\mathbf{A}$s are of fixed length, they can be readily used as features in traditional classification and clustering algorithms. However, they can cause efficiency issues because of their large size ($d \times k$); note that the typical value for embedding dimension d is 300 (Pennington et al., 2014; Mikolov et al., 2013). To tackle this issue, we further integrate cluster-level representations, at the cost of some further information loss, to create representations of lower dimension for inputs.

In particular, given all input documents with k cluster-level representations $\mathbf{A} \in \mathbb{R}^{d \times k}$ for each document, we develop an autoencoder with one hidden layer that integrates these cluster-level representations to create a final representation for each document, vector $\mathbf{a} \in \mathbb{R}^{d \times m}$ where m is the dimensionality reduction parameter and $m \times d$ is length of the representation (final layer of the encoder) and is smaller than $d \times k$ for $m < k$. Training a single-layer autoencoder corresponds to optimizing the learning parameters to minimize the overall loss between inputs and their reconstructions. For real-valued $\mathbf{A}$, squared loss is often used (Vincent et al., 2010), i.e. $l(\mathbf{A}) = ||\mathbf{A} - \hat{\mathbf{A}}||^2$ where $\hat{\mathbf{A}} \in \mathbb{R}^{d \times k}$ is reconstruction of $\mathbf{A}$ and generated by the decoder from $\mathbf{a}$. Our intuition is that if $\mathbf{a}$ leads to a good reconstruction of $\mathbf{A}$, it has retained all information available in the input.

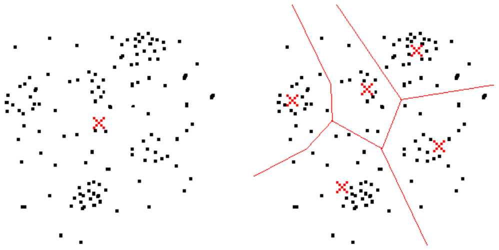


Figure 2: Illustration of VLAEs: squares show word embeddings of a hypothetical document and crosses show their corresponding average. (a): shows average word embedding in center - the case of high information loss or divergence from mean, (b): shows average word embeddings in corresponding word clusters; these embeddings are used to produce VLAEs.

We refer to $\mathbf{a} \in \mathbb{R}^{d \times m}$ as the Vector of Locally Aggregated Embeddings (VLAE). We expect this final representation to be robust against information loss due to its cluster-level local aggregation which prevents larger portions of semantically-similar words to dominate the representation.

4 Experiments

Data: We investigate VLAEs in three binary classification tasks: sentiment classification on IMDb (Maas et al., 2011), disease-text classification on Reddit, where the task is to classify reddit posts as relevant or irrelevant to specific diseases, and churn prediction on Twitter (Amiri and Daumé III, 2015), where the task is to classify/predict if given tweets indicate user intention about leaving brands, e.g. the tweet "my days with BRAND are numbered" is a *churny* tweet. See details in Table 1. For pre-processing, we change all texts to lowercase, and remove stop words, user names, and URLs from texts.

---

[1] This problem can be solved through gradient descent seeded with an initial set of k examples drawn uniformly at random from $\mathcal{V}$ (Bottou and Bengio, 1995; Sculley, 2010).

	Train	Val	Test	Unlabeled
IMDb	40K	5K	5K	50K
Reddit	2K	1K	1K	100K
Twitter	3K	1K	1K	100K

Table 1: Statistics of dataset used in experiments.

Settings: We use validation data for hyperparameter tuning and model selection. We use 300-dimensional word embeddings ($d = 300$) provided by Google (Mikolov et al., 2013), and for greater number of ds, we train `word2vec` on unlabeled data, see Table 1. In addition, we set the dimensionality reduction parameter m from $\{1 \dots 4\}$ using validation data. The best value of m is the same across tasks/datasets, $m = 2$. Furthermore, we determine the number of clusters k for VLAEs by choosing the optimal k from $\{2^i, i = \{1 \dots 7\}\}$ using validation data of each dataset. We learn optimal k with respect to task, but not embedding space, due to significant density of the semantic space of word embeddings, see Note on Clustering Word Embeddings.

Baselines: We consider two versions of DANs as baselines: `Avg_small` and `Avg_large` which represent documents by average word embedding of size $d = 300$ and $d = m \times 300$ respectively. Note that, for fair comparison, `Avg_large` has the exact same size as our model (`VLAE`); however, depending on m, their network size is 1.3-1.6 times greater than that of `Avg_small` due to difference in input dimensions. We use 3 hidden layers of size 300 for above networks. Also, to directly evaluate the effect of averaging, we do not adjust initial word embeddings during training.

Experimental Results: Table 2 shows the performance of different models across datasets. The results show that `VLAE` significantly outperforms `Avg_small` and `Avg_large` by 2.6 and 7.2 points in Macro-F1 on IMDb. The corresponding values on Reddit dataset are 6.7 and 3.4 points respectively. We believe these improvements are due to more effective and lossless representation of inputs. We note that `Avg_large` performs worse than `Avg_small` on IMDb. This could be attributed to the size of training data which may not be enough to train `Avg_large`, or to lower quality of input representations in `Avg_large` compared to `Avg_small` in case of IMDb. Note that although `VLAE` has the same number of parameters as `Avg_large`, it uses autoencoding to effectively filter redundant information. Verifying these hypotheses will be the subject of future work. In addition, `VLAE` show lower performance than `Avg_large` on Twitter dataset, F1 of 72.62 versus 73.08. We attribute this result to the shorter length of tweets for which, as we experimentally showed before, averaging does not cause major divergence in representations. On average, `VLAE` improves `Avg_small` and `Avg_large` by 4.7 and 5.3 F1 points on IMDb and Reddit (longer texts) respectively. It also shows comparable performance to best performing model on Twitter (shorter texts).

	Avg_small	Avg_large	VLAE
IMDb	83.11	78.52	**85.72***
Reddit	59.42	62.72	**66.10***
Twitter	61.42	**73.08***	72.62
AVG	67.98	71.44	**74.81**

Table 2: Macro-F1 performance of different models across datasets. Asterisk mark (*) indicates significant difference between top two systems. `VLAE` outperforms other models on longer texts, and show comparable performance to `Avg_large` on shorter text.

We also compare models in terms of the quality of their representations. For this comparison, we ignore input preparation time and assume a model that generates better representations should converge faster than other models; note that the overall turnaround time of `VLAE` is greater than that of `Avg_small` or `Avg_large` because of its input preparation time which we ignore for the purpose of this experiment. The result show that `VLAE` leads to 7.5, 1.3, and 1.3 times faster convergence than `Avg_small` and 14.9, 2.6, and 1.8 times faster convergence than `Avg_large` on IMDb, Reddit, and Twitter datasets respectively. Considering the size of these networks, these results indicate that representations obtained from VLAE are much better than those of its counterparts.

Note on Clustering Word Embeddings: In experiments, we observe clusters obtained from word embeddings are often very dense. This is a challenge for our model because with small number of clusters (ks) potentially dissimilar words can appear in the same cluster, while with large ks semantically-similar words may appear in different clusters. Neither of these are desired.

To illustrate the above challenge, we report Silhouette Coefficient (SC) (Rousseeuw, 1987) of k-means with different number of clusters obtained from words embeddings across datasets. SC indicates how well cluster boundaries are detected

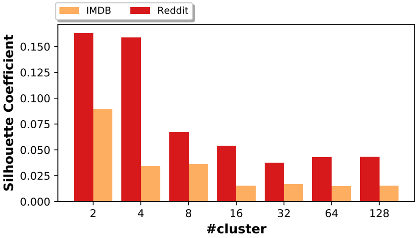


Figure 3: Mean Silhouette Coefficient computed for different number of clusters; a higher Silhouette Coefficient score indicates better defined clusters.

by a clustering model. It is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. Specifically, the mean distance between each embedding and all other embeddings in the same cluster (mc), and the mean distance between the embedding and all other embeddings in the next nearest cluster (the nearest cluster that the embedding is not part of) (mn) are used to measure SC for the embedding:

$$\frac{mn - mc}{\max(mn, mc)}.$$

The best and worst SC scores are 1 and -1 which indicate ideal and worst clustering respectively. Also, values near 0 indicate overlapping clusters.

Figure 3 shows the mean SCs computed over all word embeddings for IMDb and Reddit datasets.[2] The results show that (a): the best number of clusters is $k = 2$ on both datasets, and (b): Silhouette Coefficient scores generally home in on values close to zero as the number of clusters increases. These results show significant density of embeddings in semantic space. Therefore, we optimize the number of clusters for creating `VLAEs` by resorting to validation data and measuring task-specific performance. From these results, we conclude that a hierarchical clustering approach that recursively combines pairs of semantically-similar clusters could help better defining these clusters and perhaps improve the performance of our model.

5 Related Work

Deep averaging networks (DANs) (Joulin et al., 2017; Iyyer et al., 2015; Arora et al., 2017; Shen et al., 2018) were developed based on the successes of vector operations in embedding space. In contrast to their simplicity, DANs showed high performance in text classification tasks.

Arora et al. (2017) showed that sentences can be effectively represented by the weighted average of their word embeddings modified by PCA/SVD. In addition, the DANs developed in (Iyyer et al., 2015), (Joulin et al., 2017), and (Shen et al., 2018) were feed-forward networks that used average word embeddings to represent inputs; they were effective for several NLP tasks such as document categorization, text pair similarity, and short sentence classification. Furthermore, feed-forward architectures like DANs have been used for language modeling (Bengio et al., 2003) and greedy transition-based dependency parsing (Chen and Manning, 2014) with fast turnaround time.

In addition, previous research investigated a variety of vector operations that could replace the averaging operation used in the DANs. Many of these operations have been studied in (Mitchell and Lapata, 2008) for modeling the compositionality of short phrases, or showing the utility of simple vector computations(Banea et al., 2014). The operations in (Mitchell and Lapata, 2008) were also extended to use syntactic relation between words and grammar (Erk and Padó, 2008; Collobert and Weston, 2008). Also, clustering semantic space was studied in (Mekala et al., 2017) to learn context information for words and for tasks like topic coherence and information retrieval.

In this work, we built on previous work on DANs and investigated and tackled information loss associated with average word embeddings.

6 Conclusion and Future Work

We investigate information loss associated with average word embeddings. We show that averaging lead to significant information loss and propose to tackle the issue by identify semantically-similar parts of documents through clustering of semantic space at word-level and integrating cluster-level representations through autoencoding. A promising future direction is to use hierarchical clustering to create better cluster-level representations.

Acknowledgments

We thank anonymous reviewers for their insightful comments and constructive feedback.

---

[2] The corresponding values for Twitter data are [0.52, 0.48, 0.42, 0.31, 0.21, 0.13, 0.09] respectively. We have not reported these values in Figure 3 to better illustrate the scores for other datasets.

References

Hadi Amiri and Hal Daumé III. 2015. Target-dependent churn classification in microblogs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2361–2367. AAAI Press.

Relja Arandjelovic and Andrew Zisserman. 2013. All about vlad. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1578–1585.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations (ICLR)*.

Carmen Banea, Di Chen, Rada Mihalcea, Claire Cardie, and Janyce Wiebe. 2014. Simcompass: Using deep learning word embeddings to assess cross-level similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 560–565.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8).

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Leon Bottou and Yoshua Bengio. 1995. Convergence properties of the k-means algorithms. In *Advances in neural information processing systems*, pages 585–592.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP*.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating local descriptors into a compact image representation. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311. IEEE Computer Society.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape, and Harish Karnick. 2017. Scdv : Sparse composite document vectors using soft clustering over distributional representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 659–669. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *Proceedings of ACL-08: HLT*, pages 236–244.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–540.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

DE Rumelhart, G Eoo Hinton, and RJ Williams. 1986. Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition*, 1.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Fatemeh Torabi Asr, Robert Zinkov, and Michael Jones. 2018. Querying word embeddings for similarity and relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 675–684.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515. Association for Computational Linguistics.

Predicting the Type and Target of Offensive Posts in Social Media

Marcos Zampieri[1], Shervin Malmasi[2], Preslav Nakov[3], Sara Rosenthal[4]
Noura Farra[5], Ritesh Kumar[6]
[1]University of Wolverhampton, UK, [2]Harvard Medical School, USA
[3]Qatar Computing Research Institute, HBKU, Qatar, [4]IBM Research, USA
[5]Columbia University, USA, [6]Bhim Rao Ambedkar University, India
`m.zampieri@wlv.ac.uk`

Abstract

As offensive content has become pervasive in social media, there has been much research in identifying potentially offensive messages. However, previous work on this topic did not consider the problem as a whole, but rather focused on detecting very specific types of offensive content, e.g., hate speech, cyberbulling, or cyber-aggression. In contrast, here we target several different kinds of offensive content. In particular, we model the task hierarchically, identifying the type and the target of offensive messages in social media. For this purpose, we complied the Offensive Language Identification Dataset (OLID), a new dataset with tweets annotated for offensive content using a fine-grained three-layer annotation scheme, which we make publicly available. We discuss the main similarities and differences between OLID and pre-existing datasets for hate speech identification, aggression detection, and similar tasks. We further experiment with and we compare the performance of different machine learning models on OLID.

1 Introduction

Offensive content has become pervasive in social media and thus a serious concern for government organizations, online communities, and social media platforms. One of the most common strategies to tackle the problem is to train systems capable of recognizing offensive content, which can then be deleted or set aside for human moderation. In the last few years, there have been several studies on the application of computational methods to deal with this problem. Prior work has studied offensive language in Twitter (Xu et al., 2012; Burnap and Williams, 2015; Davidson et al., 2017; Wiegand et al., 2018), Wikipedia comments,[1] and Facebook posts (Kumar et al., 2018).

Previous studies have looked into different aspects of offensive language such as the use of abusive language (Nobata et al., 2016; Mubarak et al., 2017), (cyber-)aggression (Kumar et al., 2018), (cyber-)bullying (Xu et al., 2012; Dadvar et al., 2013), toxic comments[1], hate speech (Kwok and Wang, 2013; Djuric et al., 2015; Burnap and Williams, 2015; Davidson et al., 2017; Malmasi and Zampieri, 2017, 2018), and offensive language (Wiegand et al., 2018).

Recently, Waseem et al. (2017) analyzed the similarities between different approaches proposed in previous work and argued that there was a need for a typology that differentiates between whether the (abusive) language is directed towards a specific individual or entity, or towards a generalized group, and whether the abusive content is explicit or implicit. Wiegand et al. (2018) further applied this idea to German tweets. They experimented with a task on detecting offensive vs. non-offensive tweets, and also with a second task on further sub-classifying the offensive tweets as profanity, insult, or abuse. However, to the best of our knowledge, no prior work has explored the target of the offensive language, which might be important in many scenarios, e.g., when studying hate speech with respect to a specific target. Below, we aim at bridging this gap.

More generally, in this paper, we expand on the above ideas by proposing a novel three-level hierarchical annotation schema that encompasses the following three general categories:

A: Offensive Language Detection

B: Categorization of Offensive Language

C: Offensive Language Target Identification

---

[1] `http://bit.ly/2FhLMVz`

Proceedings of NAACL-HLT 2019, pages 1415–1420
Minneapolis, Minnesota, June 2 - June 7, 2019.

Tweet	A	B	C
@USER He is so generous with his offers.	NOT	—	—
IM FREEEEE!!!! WORST EXPERIENCE OF MY FUCKING LIFE	OFF	UNT	—
@USER Fuk this fat cock sucker	OFF	TIN	IND
@USER Figures! What is wrong with these idiots? Thank God for @USER	OFF	TIN	GRP

Table 1: Four tweets from the dataset, with their labels for each level of the annotation schema.

We further use the above schema to annotate a large dataset of English tweets, which we make publicly available online.[2]

The key contributions of this paper can be summarized as follows:

- We propose a new three-level hierarchical annotation schema for abusive language detection and characterization.
- We apply the schema to create *Offensive Language Identification Dataset (OLID)*, a new large-scale dataset of English tweets with high-quality annotation of the target and type of offenses.
- We perform experiments on OLID using different machine learning models for each level of the annotation, thus setting important baselines to compare to in future work.

While each of these sub-tasks tackles a particular type of abuse or offense, they share similar properties and the hierarchical annotation model proposed in this paper aims to capture this. Considering that, for example, an insult targeted at an individual is commonly known as cyberbulling and that insults targeted at a group are known as hate speech, we believe that OLID's use of a hierarchical annotation schema makes it a useful resource for various offensive language identification and characterization tasks.

2 Hierarchically Modelling Offensive Content

In the OLID dataset, we use a hierarchical annotation schema split into three levels to distinguish between whether the language is offensive or not (A), its type (B), and its target (C). Each level is described in more detail in the following subsections and examples are shown in Table 1.

2.1 Level A: Offensive language Detection

Level A discriminates between the following types of tweets:

- **Not Offensive (NOT):** Posts that do not contain offense or profanity;
- **Offensive (OFF):** Posts containing any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct. This includes insults, threats, and posts containing profane language or swear words.

2.2 Level B: Categorization of Offensive Language

Level B categorizes the type of offense:

- **Targeted Insult (TIN):** Posts containing insult/threat to an individual, a group, or others;
- **Untargeted (UNT):** Posts containing non-targeted profanity and swearing. Posts with general profanity are not targeted, but they contain non-acceptable language.

2.3 Level C: Offensive Language Target Identification

Level C categorizes the targets of insults/threats:

- **Individual (IND):** Posts targeting an individual. This can be a famous person, a named individual or an unnamed participant in the conversation. Insults and threats targeted at individuals are often defined as cyberbulling.
- **Group (GRP):** Posts targeting a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or other common characteristic. Many of the insults and threats targeted at a group correspond to what is commonly understood as hate speech.
- **Other (OTH)** The target of these offensive posts does not belong to any of the previous two categories (e.g., an organization, a situation, an event, or an issue).

---

[2] The data can be downloaded from the following address: `http://scholar.harvard.edu/malmasi/olid`

Keyword	**Offensive %**
medical marijuana	0.0
they are	5.9
to:NewYorker	8.3
you are	21.0
she is	26.6
to:BreitBartNews	31.6
he is	32.4
gun control	34.7
-filter:safe	58.9
conservatives	23.2
antifa	26.7
MAGA	27.7
liberals	38.0

Table 2: The keywords from the full dataset (except for the first three rows) and the percentage of offensive tweets for each keyword.

3 Data Collection

We retrieved the examples in OLID from Twitter using its API and searching for keywords and constructions that are often included in offensive messages, such as 'she is' or 'to:BreitBartNews'[3]. The full list of keywords we used is shown in Table 2.

We first carried out a round of trial annotation of 300 instances with six experts using nine keywords. The goal of the trial annotation was (*i*) to evaluate the proposed tagset, (*ii*) to evaluate the data retrieval method, and (*iii*) to create a gold standard with instances that could be used as test questions to ensure the quality of the annotators for the rest of the data, which was carried out using crowdsourcing. The keywords used in the trial annotation are shown in the first nine rows of Table 2. We included left (*@NewYorker*) and far-right (*@BreitBartNews*) news accounts because there tends to be political offense in the comments for such accounts. The keyword that resulted in the highest concentration of offensive content was the Twitter 'safe' filter, corresponding to tweets that were flagged as unsafe by Twitter (the '-' symbiol indicates 'not safe').

Since the vast majority of content on Twitter is not offensive, we tried different strategies to keep the distribution of offensive tweets at around 30% of the dataset. We excluded some keywords that were not high in offensive content during the trial annotation such as 'they are' and 'to:NewYorker'. Although 'he is' was poor in offensive content in the trial dataset (15%), we kept it as a keyword in order to avoid gender bias, and we found that in the full dataset it was more offensive (32.4%). The trial keywords that we ultimately decided to exclude due to low percentage of offensive tweets are shown in the top portion of Table 2.

We computed Fleiss' $kappa$ on the trial dataset for the five annotators on 21 of the tweets. The value was .83 for Layer A (OFF vs. NOT) indicating high agreement. As to normalization and anonymization, we did not store any user metadata or Twitter IDs, and we substituted the URLs and the Twitter mentions by placeholders.

During the full annotation task, we decided to search for more political keywords as they tend to be richer in offensive content. Thus, we sampled our full dataset, so that 50% of the tweets come from political keywords, and the other 50% come from non-political keywords. Within these two groups, tweets were evenly sampled for the keywords. In addition to 'gun control', and 'to:BreitbartNews' used during the trial annotation, four new political keywords were used to collect tweets for the full dataset: 'MAGA', 'antifa', 'conservatives', and 'liberals'. The breakdown of keywords and their offensive content in the full dataset is shown in the bottom of Table 2.

We follow prior work in related areas (Burnap and Williams, 2015; Davidson et al., 2017) and we annotate our data using crowdsourcing. We used Figure Eight[4] and we ensured data quality by (*i*) only hiring annotators who were experienced in the platform, and (*ii*) using test questions to discard annotations by individuals who did not reach a certain threshold. Each instance in the dataset was annotated by multiple annotators and inter-annotator agreement was calculated at the end.

We first acquired two annotations for each instance. In the case of disagreement, we requested a third annotation, and we then took a majority vote. The annotators were asked to label each tweet at all three levels of the annotation scheme, and we considered there to be agreement only when the annotators agreed on the labels for all levels. Approximately 60% of the time, the two annotators agreed, and thus no additional annotation was needed. A third annotation was requested for the rest of the tweets; there was no instance when more than three annotations were needed.

---

[3] *to* is a special Twitter API word indicating that the tweet was directed at a specific account (e.g., BreitBartNews).

[4] `http://www.figure-eight.com`

A	B	C	Training	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Table 3: Distribution of label combinations in OLID.

The breakdown of the data into training and testing for the labels from each level is shown in Table 3. It is worth noting that one of the key challenges we observed when collecting for OLID was producing a dataset containing a sufficient number of instances for each class. This is particularly evident in the sizes for Subtasks B and C. Other studies also had this issue when collecting similar datasets. For example, in (Davidson et al., 2017), only 5% of the tweets were considered hate speech by the majority of the annotators, and in (Burnap and Williams, 2015) only 11.6% of the examples were labeled as hate speech.

4 Experiments and Evaluation

We experiment with various models:

SVM Our simplest machine learning model is a linear SVM trained on word unigrams. SVMs have achieved state-of-the-art results for many text classification tasks (Zampieri et al., 2018).

BiLSTM We also experiment with a bidirectional Long Short-Term-Memory (BiLSTM) model, which we adapted from a pre-existing model for sentiment analysis (Rasooli et al., 2018). The model consists of (*i*) an input embedding layer, (*ii*) a bidirectional LSTM layer, and (*iii*) an average pooling layer of input features. The concatenation of the LSTM layer and the average pooling layer is further passed through a dense layer, whose output is ultimately passed through a *softmax* to produce the final prediction. We set two input channels for the input embedding layers: pre-trained FastText embeddings (Bojanowski et al., 2017), as well as updatable embeddings learned by the model during training.

CNN Finally, we experiment with a Convolutional Neural Network (CNN) model based on the architecture of (Kim, 2014), and using the same multi-channel inputs as the above BiLSTM.

Our models are trained on the training dataset, and evaluated by predicting the labels for the held-out test set. As the label distribution is highly imbalanced (see Table 3), we evaluate and we compare the performance of the different models using macro-averaged F1-score. We further report per-class Precision (P), Recall (R), and F1-score (F1), and weighted average. Finally, we compare the performance of the models against simple majority and minority class baselines.

4.1 Offensive Language Detection

The performance on discriminating between offensive (OFF) and non-offensive (NOT) posts is reported in Table 4. We can see that all models perform significantly better than chance, with the neural models performing substantially better than the SVM. The CNN outperforms the RNN model, achieving a macro-F1 score of 0.80.

4.2 Categorization of Offensive Language

In this set of experiments, the models were trained to discriminate between targeted insults and threats (TIN) and untargeted (UNT) offenses, which generally refer to profanity. The results are shown in Table 5. We can see that the CNN performs better than the BiLSTM, with a macro-F1 score of 0.69. Note that all models perform better at identifying TIN compared to UNT.

4.3 Offensive Language Target Identification

The results for the offensive target identification experiment are shown in Table 6. Here the models were trained to distinguish between three targets: a group (GRP), an individual (IND), or others (OTH). We can see that all three models achieved similar results, far surpassing the random baselines, with a slight performance edge for the neural models.

The performance of all models for the OTH class is 0, which can be explained by two factors. First, unlike the two other classes, OTH is a heterogeneous collection of targets. It includes offensive tweets targeted at organizations, situations, events, etc., thus making it more challenging for models to learn discriminative properties for this class. Second, there are fewer training instances for this class compared to the other two: there are only 395 instances for OTH vs. 1,075 for GRP and 2,407 for IND.

	NOT			OFF			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
SVM	0.80	0.92	0.86	0.66	0.43	0.52	0.76	0.78	0.76	0.69
BiLSTM	0.83	0.95	0.89	0.81	0.48	0.60	0.82	0.82	0.81	0.75
CNN	0.87	0.93	0.90	0.78	0.63	0.70	0.82	0.82	0.81	**0.80**
All NOT	-	0.00	0.00	0.72	1.00	0.84	0.52	0.72	0.	0.42
All OFF	0.28	1.00	0.44	-	0.00	0.00	0.08	0.28	0.12	0.22

Table 4: Results for offensive language detection (Level A). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (NOT, OFF), and weighted averages. Macro-F1 is also listed (best in bold).

	TIN			UNT			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
SVM	0.91	0.99	0.95	0.67	0.22	0.33	0.88	0.90	0.88	0.64
BiLSTM	0.95	0.83	0.88	0.32	0.63	0.42	0.88	0.81	0.83	0.66
CNN	0.94	0.90	0.92	0.32	0.63	0.42	0.88	0.86	0.87	**0.69**
All TIN	0.89	1.00	0.94	-	0.00	0.00	0.79	0.89	0.83	0.47
All UNT	-	0.00	0.00	0.11	1.00	0.20	0.01	0.11	0.02	0.10

Table 5: Results for offensive language categorization (level B). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (TIN, UNT), and weighted averages. Macro-F1 is also listed (best in bold).

	GRP			IND			OTH			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
SVM	0.66	0.50	0.57	0.61	0.92	0.73	0.33	0.03	0.05	0.58	0.62	0.56	0.45
BiLSTM	0.62	0.69	0.65	0.68	0.86	0.76	0.00	0.00	0.00	0.55	0.66	0.60	**0.47**
CNN	0.75	0.60	0.67	0.63	0.94	0.75	0.00	0.00	0.00	0.57	0.66	0.60	**0.47**
All GRP	0.37	1.00	0.54	-	0.00	0.00	-	0.00	0.00	0.13	0.37	0.20	0.18
All IND	-	0.00	0.00	0.47	1.00	0.64	-	0.00	0.00	0.22	0.47	0.30	0.21
All OTH	-	0.00	0.00	-	0.00	0.00	0.16	1.00	0.28	0.03	0.16	0.05	0.09

Table 6: Results for offense target identification (level C). We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (GRP, IND, OTH), and weighted averages. Macro-F1 is also listed (best in bold).

5 Conclusion and Future Work

We presented OLID, a new dataset with annotation of type and target of offensive language. It is the official dataset of the shared task *SemEval 2019 Task 6: Identifying and Categorizing Offensive Language in Social Media* (OffensEval) (Zampieri et al., 2019).[5] In OffensEval, each annotation level in OLID is an independent sub-task. To the best of our knowledge, this is the first dataset to contain annotation of type and target of offenses in social media, and it opens interesting research directions.

We further presented baseline experiments using SVMs and neural networks, which have shown that this is a challenging, yet doable task.

In future work, we would like to make a cross-corpus comparison of OLID vs. datasets annotated for similar tasks such as aggression identification (Kumar et al., 2018) and hate speech detection (Davidson et al., 2017). We further plan to create similar datasets for other languages, following OLID's hierarchical annotation scheme.

Acknowledgments

We would like to thank the anonymous NAACL reviewers for their valuable suggestions and Nikola Ljubešić for the feedback provided.

This research presented was partially supported by an ERAS fellowship awarded to Marcos Zampieri by the University of Wolverhampton.

---

[5] `http://competitions.codalab.org/competitions/20011`

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving Cyberbullying Detection with User Context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate Speech Detection with Comment Embeddings. In *Proceedings of WWW*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of EMNLP*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of TRAC*.

Irene Kwok and Yuzhou Wang. 2013. Locate the Hate: Detecting Tweets Against Blacks. In *Proceedings of AAAI*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of RANLP*.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1 – 16.

Hamdy Mubarak, Darwish Kareem, and Magdy Walid. 2017. Abusive Language Detection on Arabic Social Media. In *Proceedings of ALW*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive Language Detection in Online User Content. In *Proceedings of WWW*.

Mohammad Sadegh Rasooli, Noura Farra, Axinia Radeva, Tao Yu, and Kathleen McKeown. 2018. Cross-lingual Sentiment Transfer with Limited Resources. *Machine Translation*, 32(1-2):143–165.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. *Proceedings of ALW*.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of GermEval*.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from Bullying Traces in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, et al. 2018. Language Identification and Morphosyntactic Tagging: The Second VarDial Evaluation Campaign. In *Proceedings of VarDial*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of SemEval*.

Biomedical Event Extraction Based on Knowledge-driven Tree-LSTM

Diya Li[†]**, Lifu Huang**[†]**, Heng Ji**[†]**, Jiawei Han**[*]
[†]Computer Science Department, Rensselaer Polytechnic Institute
`{lid18,huangl7,jih}@rpi.edu`
[*]Computer Science Department, University of Illinois at Urbana Champaign
`hanj@illinois.edu`

Abstract

Event extraction for the biomedical domain is more challenging than that in the general news domain since it requires broader acquisition of domain-specific knowledge and deeper understanding of complex contexts. To better encode contextual information and external background knowledge, we propose a novel knowledge base (KB)-driven tree-structured long short-term memory networks (Tree-LSTM) framework, incorporating two new types of features: (1) dependency structures to capture wide contexts; (2) entity properties (types and category descriptions) from external ontologies via entity linking. We evaluate our approach on the BioNLP shared task with Genia dataset and achieve a new state-of-the-art result. In addition, both quantitative and qualitative studies demonstrate the advancement of the Tree-LSTM and the external knowledge representation for biomedical event extraction.

1 Introduction

Biomedical information extraction is widely used to assist the biology community on knowledge acquisition and ontology construction. Biomedical events generally refer to a change of status, particularly on proteins or genes. The goal of event extraction is to identify triggers and their arguments from biomedical text, and then assign an event type to each trigger and a role to each argument. For example, in the sentence shown in Figure 1, it includes a **gene expression** and a **positive regulation** event mention, both triggered by the word *transduced*. *Tax* is the **Theme** argument of the **gene expression** event. An event could also serve as an argument of another event, leading to a nested structure. For instance, the **gene expression** event triggered by *transduced* is also a **Theme** argument of the **positive regulation** event as shown in Figure 1.

Earlier studies on biomedical event extraction rely on kernel classification methods like the support vector machines (SVMs) (Björne and Salakoski, 2011; Venugopal et al., 2014) using hand-crafted features, which require high engineering effort and domain-specific knowledge. Recent distributional representation based approaches (Rao et al., 2017; Björne and Salakoski, 2018) explore deep neural networks which only require distributed semantic features. However, different from event extraction in the general news domain, biomedical event extraction requires broad acquisition of domain-specific knowledge and deep understanding of complex contexts. For example, in Genia event extracton of BioNLP shared task 2011 (Kim et al., 2011), about 80% of entity mentions are abbreviations of genes, proteins and diseases while more than 36% of event triggers and arguments are separated with more than 10 words.

In order to efficiently capture indicative information from broad contexts, we first adopt tree structure based long short-term memory (Tree-LSTM) networks. Compared to the linear chain structured LSTM, the Tree-LSTM takes tree-structured network topology into consideration. As shown in the top frame of Figure 1, Tree-LSTM takes the dependency tree structure of each sentence as input and gradually incorporates the information from the whole subtree into each node. Dependency tree structure can connect semantically related concepts, and thus shorten the distance between a trigger and its arguments significantly. For instance, in the following sentence "*... , which* **binds** *to the enhancer A located in the promoter of the mouse MHC class I gene* **H-2Kb**, ...", when determining the trigger type of *binds*, we need to carefully select its contextual words, such as *H-2Kb*, which indicates the object of *binds*. However, *binds* and *H-2Kb* are sepa-

Proceedings of NAACL-HLT 2019, pages 1421–1430
Minneapolis, Minnesota, June 2 - June 7, 2019.

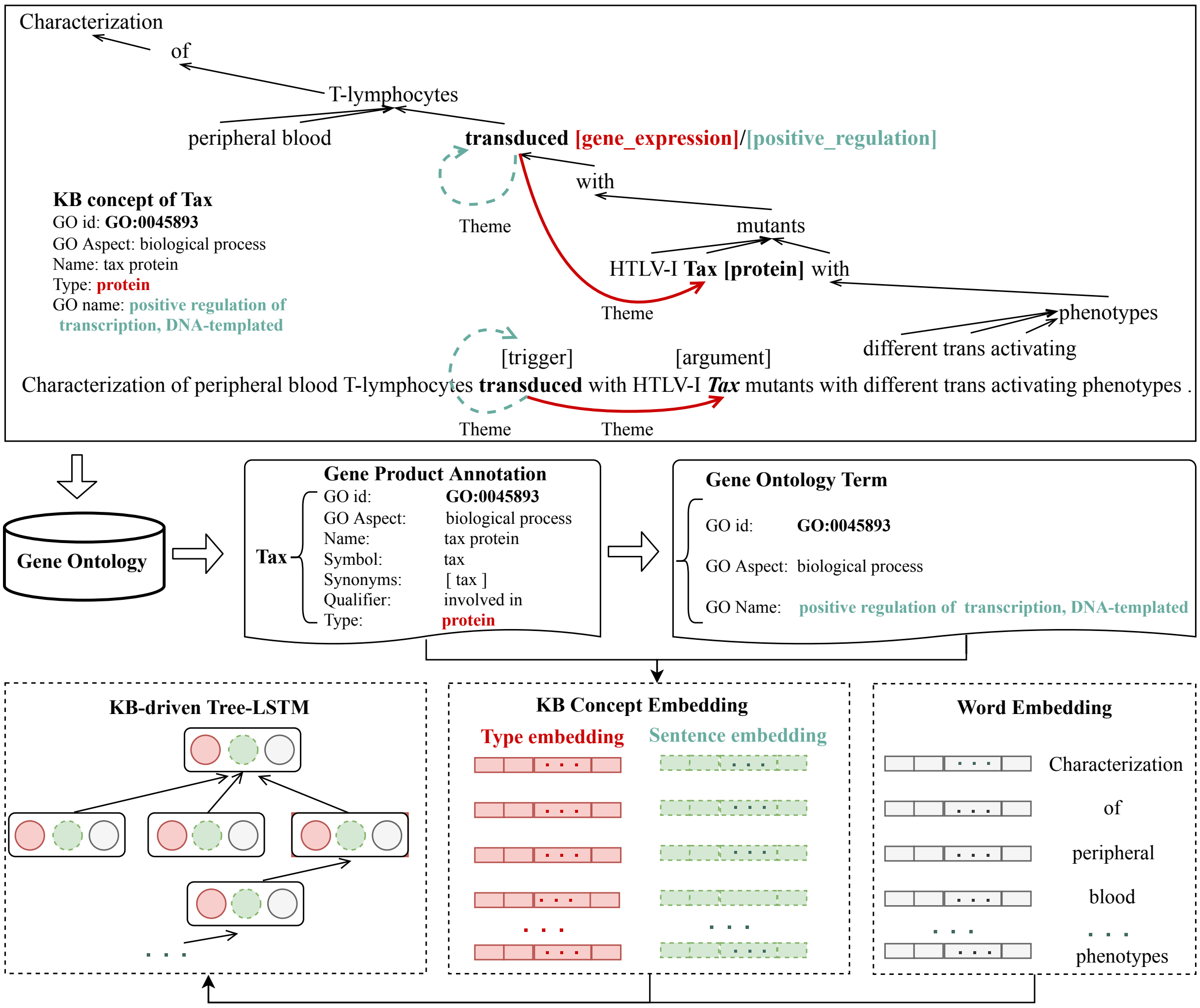


Figure 1: The framework of the KB-driven Tree-LSTM model. The upper frame shows the dependency tree structure and event annotations of a sentence; the middle frame demonstrates the knowledge base information obtained from the Gene Ontology for *Tax*; the bottom frame describes the KB-driven Tree-LSTM which takes the KB concept embedding and word embedding as input.

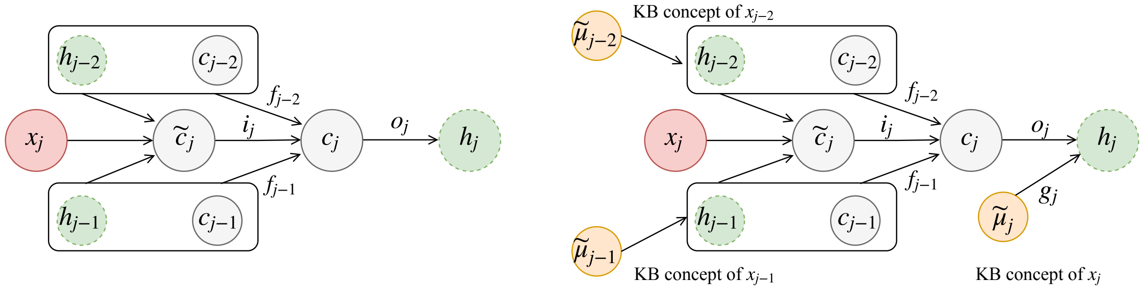
rated with 16 words which is difficult for a chain-structured LSTM to capture their long distance dependency, while within dependency tree structure, their distance is significantly shortened to 7.

Moreover, to better capture domain-specific knowledge, we further propose to leverage the external knowledge bases (KBs) to acquire properties of all the biomedical entities. The KB properties are extremely beneficial for our model to learn patterns more explicitly. Take the entity *Tax* in Figure 1 as an example, it's a protein often involved in the biological process of *positive regulation of transcription* referred to Gene Ontology (Ashburner et al., 2000). This function description provides crucial clues to determine the type of *transduced* as **positive regulation**. Therefore, to capture such knowledge from external KBs, for each entity, we first learn a KB concept embedding from its properties, and then automatically incorporate the KB representation into its Tree-LSTM hidden state with a gate function.

Our contributions are twofold: First, to the best of our knowledge, it's the first time to adopt Tree-LSTM for biomedical event extraction to effectively capture the wide contexts. Second, we further incorporate external knowledge from domain-specific KBs into the Tree-LSTM, which yields state-of-the-art performance on Genia event extraction shared task.

2 KB-driven Tree-LSTM for Event Extraction

In this section, we present our KB-driven Tree-LSTM approach for biomedical event extraction.



A. a Tree-LSTM unit **B. a KB-driven Tree-LSTM unit**

Figure 2: (A): a Tree-LSTM unit. (B): a KB-driven Tree-LSTM unit. The yellow circles with $\widetilde{\mu}$ notations denote external KB concept embeddings.

We first introduce the Tree-LSTM framework, and then describe the construction of KB concept embedding for each entity. Finally we incorporate the KB concept embedding into a Tree-LSTM and apply it for event trigger and argument extraction.

2.1 Tree-LSTM

The Tree-LSTM (Tai et al., 2015) is a variation of LSTM (Hochreiter and Schmidhuber, 1997) to a tree-structured network topology. It shows improvement in representing sentence semantic meaning compared to sequential LSTM such as Bidirectional LSTM (BiLSTM) (Graves et al., 2013). The main difference between sequential LSTM and Tree-LSTM is, at each time step, the former calculates its hidden state from the input at the current time step and the hidden state from previous step, while Tree-LSTM computes its hidden state from the input token and the hidden states of all its children nodes from the tree structure. A Tree-LSTM reduces to sequential LSTM when each node in the tree only has one child. Figure 2 (A) shows a Tree-LSTM unit. In order to obtain the hidden state h_j of an input token x_j, the unit calculates all of its children hidden states (h_{j-1}, h_{j-2}) through depth-first traversal.

2.2 Constructing KB Concept Embedding

For the biomedical event extraction, we mainly explore the Gene Ontology as our external KB since it provides detailed descriptions for each gene and gene product attributes across all species. It consists of two types of information: (1) the gene ontology (GO) defines all the gene functions, relations between these gene functions, and aspects used to describe the gene functions, including molecular function, cellular component and biological process. (2) the gene product annotations (GO Anno) provide all entity related attributes, such as the full entity name, entity type, as well as the gene functions it is related to. For example, in Figure 1, given the entity *tax*, from the gene product annotations, we can get its full entity name as *tax protein* which is a type of *proteins* and it's related to a function about *biological process*. From the gene ontology, we can further determine the specific function that *tax* is related to *positive regulation of transcription* in terms of *biological process* aspect.

In order to leverage the external KB information, we first apply QuickGO API (Binns et al., 2009) to link each entity mention to the Gene Ontology and retrieve all the KB annotations. For each entity, we carefully select two types of properties which are beneficial for event extraction task: the entity type (e.g., *protein* for *tax*) and the gene ontology function it is related to (e.g., *positive regulation of transcription* for *tax*). The entity type can facilitate the explicit pattern learning for argument role labeling, for example, the gene expression event pattern (Theme: Protein, Trigger: transduced) is more popular than (Theme: Tax, Trigger: transduced) in Figure 1. The gene ontology function can provide implicit clues to determine the trigger type as aforementioned in Section 1.

As shown in Figure 1, we assign a word embedding which pretrained on PubMed and PMC texts (Moen and Ananiadou, 2013) to represent each entity type. For each gene ontology function which is usually a long phrase, we use a state-of-the-art sentence embedding approach (Conneau et al., 2017) to automatically learn a vector representation. We then concatenate these two types of KB property representations as the final KB concept embedding.

2.3 Event Trigger Extraction

After obtaining the KB concept embeddings, we further incorporate them into the Tree-LSTM to leverage the domain-specific knowledge.

Given a sentence, for example the sentence shown in Figure 3, we first perform the dependency parsing with the Stanford dependency parser (Chen and Manning) and obtain a dependency tree structure. For each node j in the tree structure, $C(j)$ is the set of children nodes of node j and μ_k is the KB concept embedding of node k. We set μ_k to 0 if node k is not a biomedical entity. $\widetilde{\mu}_j$ denotes the sum of the KB concept embeddings of j's children nodes and $\widetilde{h}_j$ is the sum of the hidden states of j's children nodes:

$$\widetilde{h}_j = \sum_{k \in C(j)} h_k$$

$$\widetilde{\mu}_j = \sum_{k \in C(j)} \mu_k$$

where h_k is the hidden state of node k.

Then we incorporate the KB concept embeddings into the input, forget, and output gates of the Tree-LSTM:

$$\begin{aligned} i_j &= \sigma(W_i[x_j, \widetilde{h}_j, \widetilde{\mu}_j] + b_i) \\ f_{jk} &= \sigma(W_f[x_j, h_k, \widetilde{\mu}_k] + b_f) \\ o_j &= \sigma(W_o[x_j, \widetilde{h}_j, \widetilde{\mu}_j] + b_o) \end{aligned}$$

where i_j and o_j are the input gate and the output gate for node j respectively. f_{jk} is the forget gate for node j in terms of its child node k. W_i, W_f, and W_o are learnable parameters, b_i, b_f and b_o are bias terms. Thus, for each node j, the input gate gathers all KB information from its children nodes, and the output gate balances the meaningful information from its local contexts and the KB concept embeddings of its children nodes.

Besides adding the KB concept embeddings into the three gates to select useful KB formation implicitly, similar to Ma et al. (2018), we also introduce a knowledge specific output gate g_j to explicitly incorporate knowledge information into each node's hidden state. While different from Ma et al. (2018) which only considers the knowledge concept embedding of each node itself, we use the sum of the KB concept embeddings of the whole subtree instead:

$$g_j = \sigma(W_g[x_j, \widetilde{h}_j, \widetilde{\mu}_j] + b_g)$$

where W_g is a weight matrix to be learned, b_g is the bias term.

As demonstrated in Figure 2 (B), we eventually combine the implicit way of incorporating KB information into the input, output and forget gates and an explicit way of directly incorporating the KB information into a node's hidden state:

$$\begin{aligned} \widetilde{c}_j &= \tanh(W_c[x_j, \widetilde{h}_j] + b_c) \\ c_j &= \sum_{k \in C(j)} f_{jk} \odot c_k + i_j \odot \widetilde{c}_j \\ h_j &= o_j \odot \tanh(c_j) + g_j \odot \tanh(W_\mu \widetilde{\mu}_j) \end{aligned}$$

where c_j is the memory cell, W_c and W_μ are weight matrices to be learned.

After getting the hidden state h_j of each node j, we use a softmax classifier to predict a label for each node, and optimize the parameters by minimizing a negative log-likelihood loss.

2.4 Event Argument Role Labeling

After detecting all candidate triggers, we further extract arguments for each trigger. The Genia event extraction shared task provides the annotations of all entity mentions. Thus, for each trigger, we use all the entity mentions that occur in the same sentence as its candidate arguments, and then assign an argument role or *None*. Different from trigger extraction, we use the shortest dependency path (SDP) within the dependency tree structure instead of the surface contexts to better capture the dependency between the trigger and each argument.

Taking the sentence in Figure 3 as an example, given a trigger *transcription* and a candidate argument *OBF-1*, we first perform dependency parsing and extract the shortest dependency path between *transcription* and *OBF-1* with the Dijkstra's algorithm (Johnson, 1973) and obtain the shortest dependency path *transcription* → *of* → *genes* → *OBF-1*. We use the same KB-driven Tree-LSTM architecture as introduced in Section 2.3 to encode each node into a new hidden state representation. We use the hidden state of the root node h_0 as the overall vector representation of the whole dependency path. Finally, we feed the concatenation of h_0 with the hidden state of the trigger and argument as input to another softmax to predict the argument role. We also optimize the model by minimizing a negative log-likelihood loss.

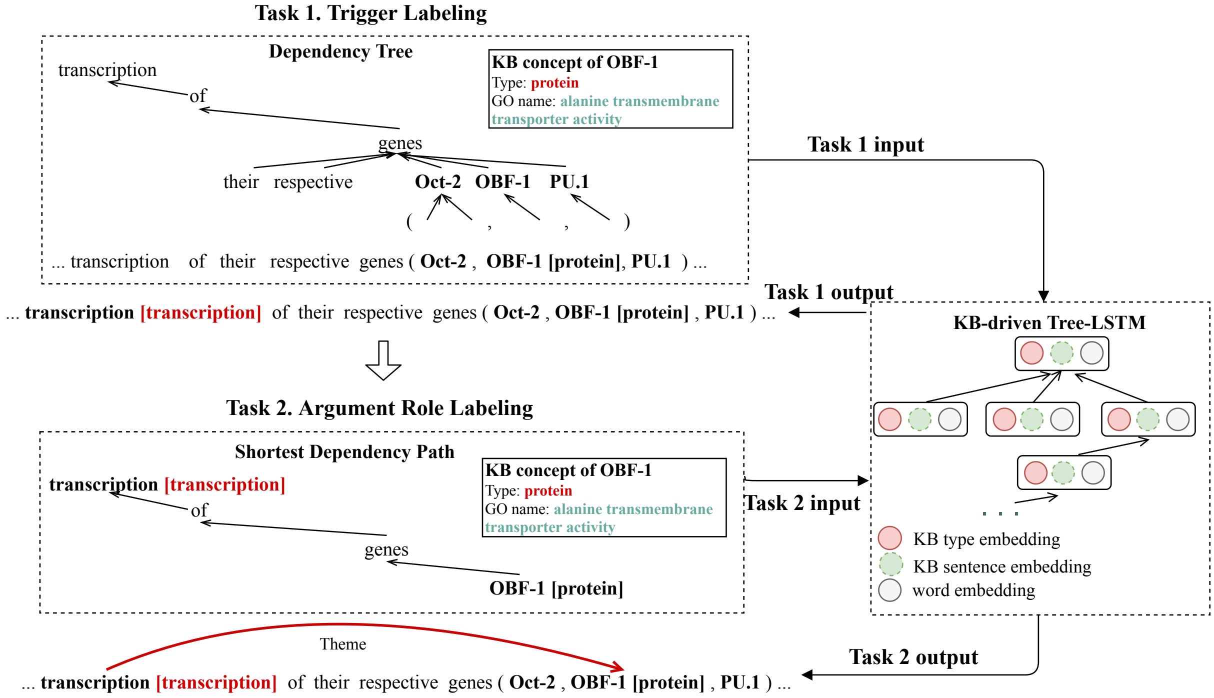


Figure 3: Examples of trigger labeling and argument role labeling via a KB-driven Tree-LSTM.

Event Type	Core Arguments
Gene expression	Theme(P)
Transcription	Theme(P)
Protein catabolism	Theme(P)
Phosphorylation	Theme(P)
Localization	Theme(P)
Binding	Theme(P)+
Regulation	Theme(P/E), Cause(P/E)
Positive regulation	Theme(P/E), Cause(P/E)
Negative regulation	Theme(P/E), Cause(P/E)
Genia corpus 2011 statistics	
events	14496
sentences	11581
nested events	37.2%
intersentence events	6.0%
abbrev. of entities	15912

Table 1: Predefined event types with accepted argument roles in Genia event extraction task, and data statistics of Genia event extraction 2011 dataset. P: protein; E: event.

3 Experiment

3.1 Task Description

The Genia Event Extraction task is the main task in the BioNLP Shared Task series (Kim et al., 2009, 2011; Nédellec et al., 2013). The Genia task defines 9 fine-grained event types as shown in Table 1. Note that a **Binding** event may take more than one protein as its **Theme** arguments. A **Regulation** event may take one protein or event as its **Theme** argument and also optionally take one protein or event as its **Cause** argument. A **Regulation** event taking an event as its argument will lead to a nested structure. 37.2% nested events are observed in Genia 2011 corpus (Björne and Salakoski, 2011). There are 6.0% inter-sentence events while our model only focuses on sentence-level event extraction.

3.2 Experimental Setup

We apply our KB-driven Tree-LSTM model on Genia 2011 data set. The entities in Genia data set are manually annotated and given as part of the input.

We evaluate our results on the test set using the official online tool provided by the Genia task organizers.[1] Following previous studies (Björne and Salakoski, 2011; Venugopal et al., 2014; Rao et al., 2017; Björne and Salakoski, 2018), we report scores obtained by the approximate span (allowing trigger spans to differ from gold spans by single words). As we only focus on matching core arguments, we use recursive matching criterion for evaluation which not requires matching of additional arguments for events referred from other events (Kim et al., 2011).

We use the word embedding pretrained on PubMed and PMC texts (Moen and Ananiadou,

---

[1] http://bionlp-st.dbcls.jp/GE/2011/eval-test/

2013) for word and type embeddings. The hyper-parameters are tuned on the development set and listed in Table 2. Word representations are updated during training with an initial learning rate of 0.1.

Parameter	Value
Word embedding size	200
Type embedding size	200
Sentence embedding size	4096
Tree-LSTM hidden size	100
Batch size	25
Epoch size	30
Dropout rate	0.5
Learning rate	0.05
Initial embedding learning rate	0.1
Optimizer	AdaGrad

Table 2: Hyper-parameters.

3.3 Results and Error Analysis

Table 3 shows the final event extraction results of applying our KB-driven Tree-LSTM model on Genia 2011 dataset with the comparison of only using Tree-LSTM and a standard BiLSTM model. Tree-LSTM outperforms the BiLSTM baseline which indicates the power of Tree-LSTM in dealing with long-distance dependency structure in biomedical literature. By incorporating external KB information, our approach achieves about 2.12% F-score gain comparing to Tree-LSTM, which demonstrates the effectiveness of the KB properties for biomedical event extraction. We will show detailed analysis in Section 3.4.

Table 4 presents the previous event extraction results from the BioNLP shared task using the same corpus. Our approach outperforms all previous methods. Among them, the systems TEES (Björne and Salakoski, 2011), EventMine-CR (Miwa et al., 2012) and Stacked Generalization (Majumder et al., 2016) are based on SVMs with well designed features. FAUST (Riedel and McCallum, 2011) and BioMLN (Venugopal et al., 2014) use jointed inference models. Björne and Salakoski (2018) adopts a convolutional neural networks (CNNs) with abundant features derived from TEES system. In our work, instead of using high-dimensional features with manual effort as in these previous models, our approach only requires pretrained distributed word representations as input features.

We notice that our approach achieves high scores on **Simple** event types but get relatively low scores on **Binding** event and **Regulation** event types. We analyze the results and find that **Binding** event extraction is more challenging since it usually has multiple arguments. For example, Figure 4 shows two sentences which are chosen from the output of the development data set. There are two **Binding** event mentions in the first sentence: E1 (Trigger: interacting, Type: Binding, Theme: RUNX1, Theme2: p3000) and E2 (Trigger: binding, Type: Binding, Theme: CREB). Our model mistakenly extracts *CREB* as a *Theme* of E1 since *CREB* is highly related to protein p300 in the dependency tree structure.

Regulation events are considered as the most challenging event type because they usually have an optional **Cause** argument and are involved in nested structures, which are not handled well by most of current event extraction approaches. In addition, intuitively, most trigger words are verbs or nouns. We rank all the trigger words in the train-

System	Event Type	Rec	Prec	F1
KB-driven Tree-LSTM	Gene expression	74.35	87.24	80.28
	Transcription	69.54	82.31	75.39
	Protein catabolism	46.67	87.50	60.87
	Phosphorylation	81.62	87.28	84.36
	Localization	59.69	80.28	68.47
	Simple total	72.62	85.95	78.73
	Binding	37.68	53.16	44.10
	Regulation	36.62	53.61	43.52
	Positive regulation	41.37	57.90	48.26
	Negative regulation	46.06	52.39	49.02
	Regulation total	41.73	55.73	47.72
	Event total	**52.14**	**67.01**	**58.65**
Tree-LSTM	**Simple total**	71.22	83.41	76.83
	Binding	34.83	48.72	40.62
	Regulation total	39.78	53.54	45.64
	Event total	**50.28**	**64.56**	**56.53**
BiLSTM	**Simple total**	68.09	78.75	73.03
	Binding	38.49	43.05	40.65
	Regulation total	37.64	53.81	44.30
	Event total	**48.44**	**62.18**	**54.46**

Table 3: Precision (Prec), recall (Rec) and F-score (F1) results achieved by the KB-driven Tree-LSTM model on the test set of BioNLP Genia 2011, evaluated on approximate span and recursive criteria.

System	Rec	Prec	F1
TEES(Björne and Salakoski, 2011)	49.56	57.65	53.30
FAUST(Riedel and McCallum, 2011)	49.41	64.75	56.04
EventMine-CR(Miwa et al., 2012)	53.35	63.48	57.98
BioMLN(Venugopal et al., 2014)	53.42	63.61	**58.07**
Stacked Generalization(Majumder et al., 2016)	48.96	66.46	56.38
CNN(Björne and Salakoski, 2018)	49.94	69.45	**58.07**

Table 4: State-of-the-art system results evaluated on BioNLP Genia 2011 test dataset with approximate span and recursive criteria.

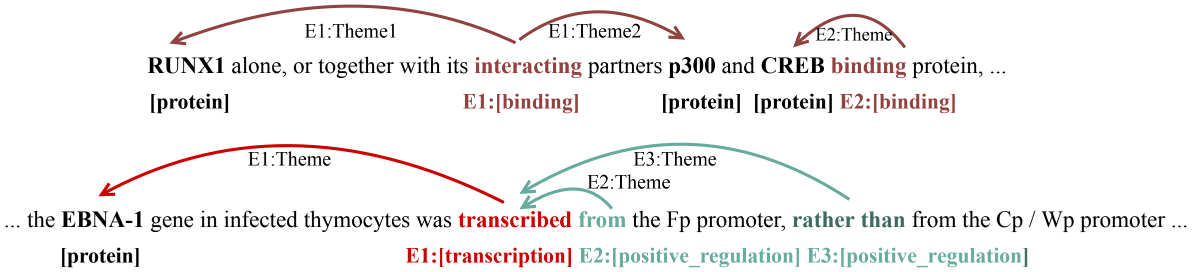
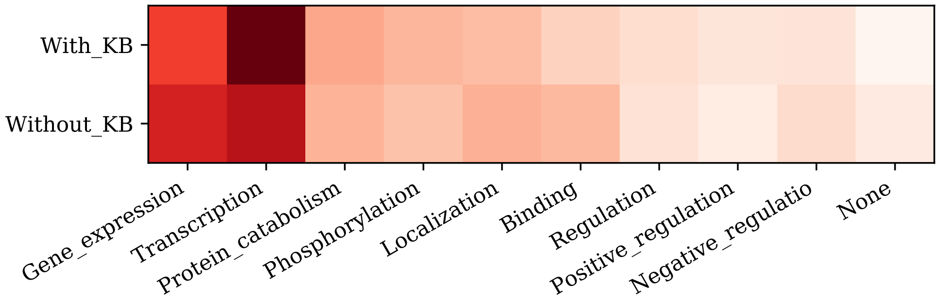


Figure 4: Case study on binding event and regulation event types.



... **transcription** **[transcription]** of their respective genes (**Oct-2 [protein]** , **OBF-1 [protein]**, **PU.1[protein]**) ...

Figure 5: Visualization of the effect of KB concept embeddings on trigger labeling for the word *transcription*.

ing data set according to their frequency, and find that most of spurious errors for **Regulation** event trigger extraction occur when the trigger words are prepositions or conjunctions. For instance, in Figure 4, the second sentence contains two positive **Regulation** events triggered by a preposition *from* and a conjunction *rather than*. Such function words are rarely annotated as triggers and our KB-aware Tree-LSTM cannot well collect meaningful contexts from their subtrees.

3.4 Effect of KB concepts

As shown in Table 3, we achieve about 3.5% and 2.1% F1 score gain on **Binding** and **Regulation** event types by leveraging external KB information into the Tree-LSTM. In order to show the effect of KB concept embeddings, we visualize the probabilities of word *transcription* to be predicted for each event type. As Figure 5 shows, by adding KB concept embeddings, the function description *positive regulation of transcription, DNA-templated* provided by the biomedical entity *OBF-1* significantly enhances the probability of *transcription* being predicted to a *Transcription* event type.

Similarly, Figure 6 visualizes the probabilities of the E1 event mention (Trigger: transduced, Type: gene_expression, Theme: Tax) to be predicted as an argument of E2 event mention (Trigger: transduced, Type: positive regulation, Theme: E1). We can see that, without using KB information, the Tree-LSTM mistakenly predict the argument role of E1 as **None**. In contrast, by incorporating KB concept embeddings, especially

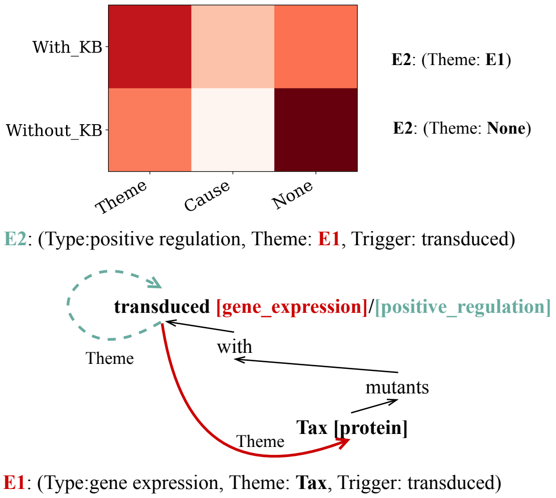


Figure 6: Visualization of the effect of KB concept embedding on argument role labeling for a Positive Regulation event triggered by *transduced* and a Gene Expression event E1 (Theme: Tax, Trigger: transduced).

the information from the function description *positive regulation of transcription, DNA-templated* for *Tax*, our approach successfully promotes the probability of E1 being predicted as the *Theme* of E2.

4 Related Work

As a crucial task in information extraction, event extraction has gained a lot of interest. In general news domain, previous work on event extraction can be divided into two main categories. The first is feature-based methods which mainly focus on feature design, leveraging local features (Grishman et al., 2005; Ahn, 2006) and global features (Ji and Grishman, 2008; Liao and Grishman, 2011; Huang and Riloff, 2012) to improve the performance. Some studies proposed joint models to overcome the error propagation problem (Poon and Vanderwende, 2010; Riedel et al., 2009; Li et al., 2013; Venugopal et al., 2014; Li et al., 2014). The second category includes distributional representation based methods which have been applied into event extraction extensively. Most of these approaches are based on the standard Convolutional Neural Networks (CNNs) (Chen et al., 2015; Nguyen and Grishman, 2015, 2016), Recurrent Neural Networks (RNNs) (Nguyen et al., 2016), generative adversarial networks (Hong et al., 2018), zero-shot learning (Huang et al., 2017) and advanced attention mechanisms (Liu et al., 2018b; Chen et al., 2018).

Our work is also related to the studies which leverage the external knowledge base for information extraction. Liu et al. (2017) takes advantage of external resources, such as FrameNet, to label events while Chen et al. (2017) adopts distance supervision to augment the training data. Liu et al. (2018a) develops an attention-based model for event extraction. What's more, shortest dependency path is broadly explored for information extraction, especially for relation classification (Xu et al., 2015; Miwa and Bansal, 2016) and shows promising benefits.

Biomedical event extraction task part of the BioNLP Shared Task series (Kim et al., 2009, 2011; Nédellec et al., 2013). Previous studies mainly explore local and global features with SVM model (Miwa et al., 2010, 2012; Björne and Salakoski, 2013; Majumder et al., 2016). Riedel and McCallum (2011) develop a joint model with dual decomposition. Cohen et al. (2009), Kilicoglu and Bergler (2011) and Bui et al. (2013) develop rule-based methods and achieve high precision. Venugopal et al. (2014) leverage Markov logic networks for joint inference. Rao et al. (2017) uses the Abstract Meaning Representations (AMR) to extract events based on the assumption that an event structure can be derived from an AMR subgraph. Recently, some representation-based models (Jagannatha and Yu, 2016; Rao et al., 2017; Björne and Salakoski, 2018) have been proposed while most of them adopt the widely used CNNs and RNNs with features derived from the biomedical text. Lim et al. (2018) implements a binary Tree-LSTM architecture for biomedical relation extraction. Compared with these methods, our approach only requires pre-trained distributed word representations as input features and incorporates meaningful KB information into a Tree-LSTM.

5 Conclusions and Future Work

In this paper, we show the effectiveness of using a KB-driven tree-structured LSTM for event extraction in biomedical domain. The Tree-LSTM can efficiently capture semantically related concepts for each node within the tree structure. By leveraging the external KB concept properties including the entity type and the function description, our approach is able to perform deep understanding of domain-specific expressions and connections. Without using manually designed high-dimensional features, our approach significantly outperforms all previous methods. In the future, we plan to explore a broader range of properties from KB to facilitate biomedical information extraction tasks.

Acknowledgments

This work was supported by the U.S. NSF No. 1741634, Air Force No. FA8650-17-C-7715 and ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. 2000. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25.

David Binns, Emily Dimmer, Rachael Huntley, Daniel Barrell, Claire O'donovan, and Rolf Apweiler. 2009. Quickgo: a web-based tool for gene ontology searching. *Bioinformatics*, 25(22):3045–3046.

Jari Björne and Tapio Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*.

Jari Björne and Tapio Salakoski. 2013. Tees 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*.

Jari Björne and Tapio Salakoski. 2018. Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*.

Quoc-Chinh Bui, David Campos, Erik van Mulligen, and Jan Kors. 2013. A fast rule-based approach for biomedical event extraction. In *proceedings of the BioNLP shared task 2013 workshop*.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proc. EMNLP2014*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proc. ACL2017*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. ACL-IJCNLP2015*.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multilevel attention mechanisms. In *Proc. EMNLP2018*.

K Bretonnel Cohen, Karin Verspoor, Helen L Johnson, Chris Roeder, Philip V Ogren, William A Baumgartner Jr, Elizabeth White, Hannah Tipney, and Lawrence Hunter. 2009. High-precision biological event extraction with a concept recognizer. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proc. EMNLP2017*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. *ACE*, 5.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, pages 1735–1780.

Yu Hong, Wenxuan Zhou, jingli zhang jingli, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proc. ACL2018*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2017. Zero-shot transfer learning for event extraction. *arXiv preprint arXiv:1707.01066*.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proc. EACL2012*.

Abhyuday N Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *Proc. NAACL-HLT2016*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*.

Donald B Johnson. 1973. A note on dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20(3):385–388.

Halil Kilicoglu and Sabine Bergler. 2011. Adapting a general semantic interpretation approach to biological event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP Shared Task 2009 Workshop*.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011. Overview of genia event task in bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL2013*.

Qi Li, Heng Ji, Hong Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *Proc. EMNLP2014*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *Proc. RANLP2011*.

Sangrak Lim, Kyubum Lee, and Jaewoo Kang. 2018. Drug drug interaction extraction from the literature using a recursive neural network. *PloS one*, 13(1):e0190926.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. *Statistics*, 1000:1250.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proc. ACL2017*.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proc. EMNLP2018*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Proc. AAAI2018*.

Amit Majumder, Asif Ekbal, and Sudip Kumar Naskar. 2016. Biomolecular event extraction using a stacked generalization based classifier. In *Proc. ICON2016*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(01):131–146.

Makoto Miwa, Paul Thompson, and Sophia Ananiadou. 2012. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, 28(13):1759–1765.

SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proc. LBM2013*.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proc. NAACL-HLT2016*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proc. ACL-IJCNLP2015*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proc. EMNLP2016*.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proc. NAACL-HLT2010*.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using abstract meaning representation. In *Proceedings of the BioNLP 2017 workshop*.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proc. EMNLP2014*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. EMNLP2015*.

Detecting Cognitive Impairments by Agreeing on Interpretations of Linguistic Features

Zining Zhu[1,2], **Jekaterina Novikova**[1], **Frank Rudzicz**[3,5,2,4,1]
[1] Winterlight Labs [2] University of Toronto
[3] Li Ka Shing Knowledge Institute, St Michael's Hospital, Toronto ON, Canada
[4] Vector Institute for Artificial Intelligence [5] Surgical Safety Technologies
`{zining, jekaterina}@winterlightlabs.com, frank@cs.toronto.edu`

Abstract

Linguistic features have shown promising applications for detecting various cognitive impairments. To improve detection accuracies, increasing the amount of data or the number of linguistic features have been two applicable approaches. However, acquiring additional clinical data can be expensive, and hand-crafting features is burdensome. In this paper, we take a third approach, proposing Consensus Networks (CNs), a framework to classify after reaching agreements between modalities. We divide linguistic features into non-overlapping subsets according to their modalities, and let neural networks learn low-dimensional representations that agree with each other. These representations are passed into a classifier network. All neural networks are optimized iteratively.

In this paper, we also present two methods that improve the performance of CNs. We then present ablation studies to illustrate the effectiveness of modality division. To understand further what happens in CNs, we visualize the representations during training. Overall, using all of the 413 linguistic features, our models significantly outperform traditional classifiers, which are used by the state-of-the-art papers.

1 Introduction

Alzheimer's disease (AD) and its usual precursor, mild cognitive impairment (MCI), are prevalent neurodegerative conditions that inhibit cognitive abilities. Cognitive impairments are traditionally diagnosed only with standard clinical tests like MoCA (Nasreddine et al., 2005) and the Rey-Auditory Verbal learning Test (Rey, 1941), but hiring clinicians to administer these tests and analyze their results is costly. Fortunately, many cognitive impairments can be observable in daily life, because they impact one's language abilities. For example, cognitively impaired people tend to use more pronouns instead of nouns, and pause more often between sentences in narrative speech (Roark et al., 2011).

This insight makes automatic detection possible. Machine learning classifiers can detect cognitive impairments given descriptive linguistic features. In recent work, linguistic features including pronoun-noun-ratios, pauses, and so on, are used to train classifiers to detect cognitive diseases in various tasks. For example, Fraser et al. (2015) achieved up to 82% accuracy on DementiaBank[1], the largest publicly available dataset on detecting cognitive impairments from speech, and Weissenbacher et al. (2016) achieved up to 86% accuracy on a corpus of 500 subjects. Yancheva et al. (2015) estimated Mini-Mental State Estimation scores (MMSEs), describing the cognitive status and characterizing the extent of cognitive impairment.

To improve the accuracy of automated assessment using engineered linguistic features, there are usually two approaches: incorporating more clinical data or calculating more features. Taking the first approach, Noorian et al. (2017) incorporated normative data from Talk2Me[2] and the Wisconsin Longitudinal Study (Herd et al., 2014) in addition to DementiaBank, which increased AD:control accuracy up to 93%, and moderateAD:mildAD:control three-way classification accuracy to 70%. Taking the second approach, Yancheva and Rudzicz (2016) used 12 features derived from vector space models and reached a .80 F-score on DementiaBank. Santos et al. (2017) calculated features depicting characteristics of co-occurrence graphs of narrative transcripts (e.g., the degree of each vertex in the graph). Their classifiers reached 65% accuracy on DementiaBank

---

[1] `https://talkbank.org/DementiaBank`

[2] `https://www.cs.toronto.edu/talk2me/`

Proceedings of NAACL-HLT 2019, pages 1431–1441
Minneapolis, Minnesota, June 2 - June 7, 2019.

(MCI versus a subset of Control).

There are limitations in either of the two approaches. On one hand, acquiring additional clinical data can be expensive (Berndt and Cockburn, 2013). Moreover, the additional data should be similar enough to existing training data to be helpful. On the other hand, crafting new features requires creativity and collaboration with subject matter experts, and the implementation can be time consuming. Neither of these approaches is satisfactory.

These limitations motivate us to take a third, novel approach. Instead of using new data or computing new features, we use the existing linguistic features.

If the speaker is cognitively impaired, and their language ability is affected, features from each of the acoustic, syntactic, and semantic modalities should reflect such change (Szatloczki et al., 2015; Moro et al., 2015; Fraser et al., 2015). We therefore need to distill the common information revealed by features from multiple, mainly distinct, modalities.

To utilize information common across different modalities, Becker and Hinton (1992) and de Sa (1994) let classifiers look at each modality and supervise each other. These examples illustrated the effectiveness of multi-view learning in utilizing common information among different observations, but their algorithms fail to train useful classifiers for cognitive impairments in our datasets. Without explicit supervision, self-supervised models almost always converge to a state producing the same predictions for all people, giving trivial classifiers.

Instead of aligning the predictions from modalities, we let the representations of the modalities agree. Generative adversarial networks (GANs) provide an approach. In GANs, a "discriminator" network is trained to tell whether a vector is drawn from the real world or produced synthetically by a "generator" neural network, while the generator is trained to synthesize images as close to real data as possible. We borrow this setting, and encourage the neural networks interpreting different modalities to produce representations of modalities as similar to each other as possible. This leads to our classifier framework, consensus networks (CNs).

Consensus networks constitute a framework using adversarial training to utilize common information among modalities for classification. In this framework, several neural networks ("ePhysicians") are juxtaposed, each learning the representation of a partition of linguistic features for each transcript. Being trained towards producing agreed representations, we show they are increasingly able to capture common information contained across disparate subsets of linguistic features.

We empirically add two extensions to CN that improve the classification accuracies, called the "noise modality" and "cooperative optimization" respectively, as explained below. To illustrate the effectiveness of the consensus mechanisms, we present two ablation studies. First, we compare neural networks built with consensus (CN) and those without (MLP). On partial or complete modalities, CN outperforms MLP significantly. Second, we compare CNs built with linguistic features divided into random subsets. Division according to their natural modalities train better consensus networks. We also visualize the representations during training procedure, and show that when the representations agree, their distributions appear symmetric.

Overall, taking all 413 linguistic features, our models significantly outperform traditional classifiers (e.g., support vector machines, quadratic discriminant analysis, random forest, Gaussian process), which are used by the state-of-the-art.

2 Related Works

Generative Adversarial Networks The idea of aligning representations by making them indistinguishable is inspired by GAN (Goodfellow et al., 2014), where a generator produces fake images (or other data) that are as similar to real data as possible. However, our model does not have a generator component as GANs do. Instead, we only compress features into representations while trying to align them.

Multi-view Learning Learning from multiple modalities is also referred to as multi-view learning. Becker and Hinton (1992) set up multiple neural networks to look at separate parts of random-dot stereograms of curved surfaces, and urge their prediction to equal each other. The trained neural networks were able to discover depth without prior knowledge about the third dimension. de Sa (1994) divided linguistic features into two modalities, and passed them to two separate neural networks. The two neural networks supervised each other (i.e., output labels that are used to train the other) during alternative optimization steps to reach a consensus. Their self-supervised system reached 79±2% accuracy using the Peterson-Barney vowel recognition dataset (Peterson and Barney, 1952). Benediktsson et al. (1997) computed multiple views from the same feature sets and classified by taking their majority votes. Pou-Prom and Rudzicz (2018) used canonical correlation analysis (CCA) to classify using multiple aspects. Contrary to that work, our consensus networks take in distinct subsets of features as modalities. Co-training (Blum and Mitchell, 1998) and tri-training (Zhou and Li, 2005) use distinct subsets of features, but they use them to train distinct classifiers, and let the results directly supervise each other. Their approach 'bootstrapped' classifications based on a few labeled data, but our method explicitly uses a modality discriminator that enforces alignments between modalities.

Domain Adaptation In domain adaptation and multi-task learning, there have been many attempts to learn indistinguishable embeddings between domains. For example, Ganin et al. (2016) and Joty et al. (2017) applied a gradient reversal layer to let encoders minimize the domain classification loss. Baktashmotlagh et al. (2013) minimized the maximum-mean discrepancy (MMD) loss in a reproductive kernel Hilbert space (RKHS) of the latent representations. Motiian et al. (2017) used semantic similarity loss between latent representations of different class data to encourage alignments between domains. Liu et al. (2017) and Chen and Cardie (2018) used shared and private networks to learn information contained either commonly in domains or domain-specific. Our work is unique. First, there is only one domain in our problem setting. Second, we use iterative optimization to encourage discrepancies between domains. Third, we have two empirical improvements (noise modality and cooperative optimization) that make our Consensus Networks outperform traditional classifiers.

3 Methods

3.1 Dataset

We use DementiaBank, the largest publicly available dataset for detecting cognitive impairments. It includes verbal descriptions (and associated transcripts) of the Cookie Theft picture description task from the Boston Diagnostic Aphasia Examination (Becker et al., 1994). The version we have access to contains 240 speech samples labeled Control (from 98 people), 234 with AD (from 148 people), and 43 with MCI (from 19 people)[3]. All participants were older than 44 years.

3.2 Linguistic Features

The dataset contains narrative speech descriptions and their transcriptions. We preprocess them by extracting 413 linguistic features for each speech sample. These linguistic features are proposed by and identified as the most indicative of cognitive impairments by various previous works, including Roark et al. (2007), Chae and Nenkova (2009), Roark et al. (2011), Fraser et al. (2015), and Hernández-Domínguez et al. (2018). After calculating these features, we use KNN imputation to replace the undefined values (resulting from divide-by-zero, for example), and then normalize the features by their z-scores. The following are brief descriptions of these features, grouped by their natural categories. More detailed descriptions are included in the Appendix.

There are 185 acoustic features (e.g., average pause time), 117 syntactic features (e.g., Yngve statistics (Yngve, 1960) of the parse tree, computed by the LexParser in CoreNLP (Manning et al., 2014)), and 31 semantic features (e.g., cosine similarity between pairs of utterances) Moreover, we use 80 part-of-speech features that relate to both syntax and semantics but are here primarily associated with the latter.

Modality Division After representing each sample with a 413-dimensional vector $\mathbf{x}$ consisting of all available linguistic features, we divide the vector into M partitions ('modalities') of approximately equal sizes $[\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_M}]$, according to the groups mentioned above. Unless mentioned otherwise (e.g., in the ablation study shuffling

---

[3] The version of DementiaBank dataset we acquired contains a slightly different number of samples from what some previous works used. In Control:AD, Fraser et al. (2015) used 233 Control and 240 AD samples; Yancheva and Rudzicz (2016) had 241 Control and 255 AD samples; Hernández-Domínguez et al. (2018) had 242 Control and 257 AD samples (with 10% control samples excluded from the evaluation). In Control:MCI, Santos et al. (2017) used all 43 transcriptions from MCI and 43 sampled from the Control group. With no clear descriptions of the sampling procedures, the constituents of the Control group might differ from our sample. In this paper, we run our models on the same tasks (i.e., Control:AD) and compare to the results of models used in the literature.

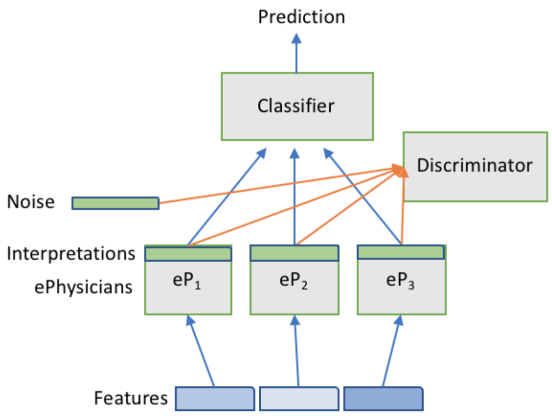


Figure 1: Overview of model structure when features (blue rectangles) are divided into three modalities (non-overlapping subsets). Each subset of features are passed into an "ePhysician" neural network whose outputs (green rectangles) are the representations. They are passed (one by one) into a "Discriminator" neural network and (after combined) into a "Classifier" network, respectively.

modalities), this is our default choice for assigning modalities.

3.3 Model

Figure 1 is an example of our model structure (with $M = 3$ modalities), and this section elaborates the inference procedure, the training algorithm, and our two improvements.

Inference With the extracted linguistic features divided into subsets by their modalities, each speech sample is described by $M = 3$ non-overlapping feature vectors $\mathbf{x} = [\mathbf{x_1}, ..., \mathbf{x_m}]$. These feature vectors are then passed into corresponding ePhysician networks, each outputting a vector $\mathbf{i_m}$, which is a distilled representation of the subject from a modal-specific perspective (e.g., the semantic). We also refer to it as the *interpretation vector* and use them interchangeably. Formally, the m^{th} ePhysician can be written as a function, $f_m(.)$ generating the representation:

$$\mathbf{i_m} = f_m(\mathbf{x_m})$$

To challenge the similarity of representations from different modalities, we let a discriminator neural network $f_D(.)$ take in each of the M representations and predict the likelihood of the originating modality m.

$$P(m = k \,|\, \mathbf{i}) = \frac{e^{f_D(\mathbf{i})_k}}{\sum_k e^{f_D(\mathbf{i})_k}}$$

where $k = 1, ..., M$.

To attempt a diagnosis, a classifier network $f_C(.)$ takes in the combination of M representations of each speech sample, and outputs a prediction probability for detection result y:

$$P(y = l \,|\, \mathbf{x}) = \frac{e^{f_C(\mathbf{i_{1..M}})_l}}{\sum_l e^{f_C(\mathbf{i_{1..M}})_l}}$$

where $l \in \{0, 1\}$ for two-class classification (i.e., 0 for healthy and 1 for dementia). The predicted class corresponds to those with the highest probability:

$$\hat{y} = \arg\max_l P(y = l|\mathbf{x})$$

Optimization The training procedure optimizes the adversarial objective, and the conventional classifier objective:

- The adversarial objective sets up the ePhysicians and the Discriminator to work in an adversarial manners. The ePhysicians try to produce indistinguishable representations, while the discriminator tries to tell them apart.

$$\min_D \max_P \mathcal{L_D} \text{ where}$$
$$\mathcal{L_D} = \mathbb{E}_\mathbf{x} \mathbb{E}_{m=1..M} \{-\text{log}P(\hat{m} = m|\mathbf{i_m})\} \tag{1}$$

- Make the classifier network as accurate as possible. This is done by minimizing the cross entropy classification loss:

$$\min_C \mathcal{L_C} \text{ where}$$
$$\mathcal{L_C} = \mathbb{E}_\mathbf{x} \{-\text{log}P(\hat{y} = y|\mathbf{i_{1..M}})\} \tag{2}$$

Overall, $\min_C \mathcal{L_C}$ and $\min_D \max_P \mathcal{L_D}$ set up a complex optimization problem. We use iterative optimization steps, similar to Goodfellow et al. (2014).

There are two tricks that we found to improve the performance of the models. Namely, the *noise modality* and the *cooperative optimization*. We explain them below.

Noise Modality For each participant session, we add a "noise modality representation" $\mathbf{i_0}$ drawn from a Gaussian distribution with the mean and variance identical to those of other representation vectors.

$$\mathbf{i_0} \sim \mathcal{N}(\mu_{\mathbf{i_{1..M}}}, \sigma^2_{\mathbf{i_{1..M}}})$$

This additional representation vector is passed into the discriminator, but not passed into the classifier. The first optimization goal (1) is therefore:

$$\min_D \max_P \mathcal{L}_{\mathcal{D}} \text{ where}$$
$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{\mathbf{x}} \mathbb{E}_{m=0..M} \left\{ -\log P(\hat{m} = m | \mathbf{i_m}) \right\} \tag{3}$$

To some extent, the noise representation vector works like a regularization mechanism to refrain the discriminator from making decisions based on superficial statistics. We show in 4.1 that this addition empirically improves classifier performance.

Cooperative Optimization When optimizing the classifier, we find that allowing gradients to propagate back to the ePhysicians improves the model's overall performance. During optimization, the ePhysicians need to *cooperate* with the classifier (while adversarial to the discriminator). The second optimization goal (2) is therefore:

$$\min_{C,P} \mathcal{L}_{\mathcal{C}} \text{ where}$$
$$\mathcal{L}_{\mathcal{C}} = \mathbb{E}_{\mathbf{x}} \left\{ -\log P(\hat{y} = y | \mathbf{i_{1..M}}) \right\} \tag{4}$$

Implementation As a note of implementation, all ePhysicians, the classifier, and the discriminator networks are fully connected networks with Leaky ReLU activations (Nair and Hinton, 2010) and batch normalization (Ioffe and Szegedy, 2015). The hidden layer sizes are all 10 for all ePhysician networks, and there are no hidden layers for the discriminator or classifier networks. Although modalities might contain slightly different numbers of input dimensions, we do not scale the ePhysician sizes. This choice comes from the intuition that the ePhysicians should extract into the representation as similar information as possible. We use three Adam optimizers (Kingma and Ba, 2014), each corresponding to the minimization of ePhysician, Discriminator, and the Classifier, and optimize iteratively for no more than 100 steps. The optimization is stopped prior to step 100 if the classification loss $\mathcal{L}_{\mathcal{C}}$ converges (i.e., does not differ from the previous iteration by more than 1×10^{-4}) on training set. The train / validation / test set are divided randomly in 60/20/20 proportions.

Algorithm 1 The overall algorithm

```
Initialize the networks
for step := 1 to N do        ▷ N is a hyper-param
    for minibatch $\mathbf{x}$ in training data $\mathcal{X}$ do
        for modality m := 1 to M do
            $\mathbf{i_m} = I_m(\mathbf{x_m})$
        Sample the noise modality $\mathbf{i_0}$
        Calculate $\mathcal{L}_{\mathcal{D}}$ with $\mathbf{i_{0..M}}$
        Concatenate $\mathbf{i_{1..M}}$ and calculate $\mathcal{L}_{\mathcal{C}}$
        $\min_{C,P} \mathcal{L}_{\mathcal{C}}$        ▷ Cooperative optimization
        $\max_{P} \mathcal{L}_{\mathcal{D}}$
        for k:=1 to K do        ▷ K is a hyper-param
            $\min_{D} \mathcal{L}_{\mathcal{D}}$
```

4 Experiments

We first show the effectiveness of our two improvements to the model. Next, we do two ablation studies on the arrangements of modalities. Then, we evaluate our model against several traditional supervised learning classifiers used by state-of-the-art works. To understand the model further, we also visualize the principal components of the representation vectors throughout several runs.

4.1 Noise Modality Improves Performance

We compare a CN model with a noise modality to one without (with other hyper parameters including hidden dimensions and learning rates identical).

Table 1 shows that in the AD:MCI classification task, the model with an additional noise modality is better than the one without ($p = 0.04$ on 2-tailed t-test with 18 DoF). Here is a possible explanation. Without adding a noise modality, the discriminators may simply look at the superficial statistics, like the mean and variances of the representations. This strategy tends to neglect the detailed aspects encoded in the representation vectors. Adding in the noise modality penalizes this strategy and trains better discriminators by forcing them to *study the details*.

In the following experiments, all models contain the additional noise modality.

4.2 Effectiveness of Cooperative Optimization

The second improvement, cooperative optimization, also significantly improves model perfor-

Model	Micro F1	Macro F1
Noise	**.7995 ± .0450**	**.7998 ± .0449**
No noise	.7572 ± .0461	.7577 ± .0456

Table 1: Comparison of models with and without representations in noise modality. The models containing a Gaussian noise modality outperform those without.

mance. We compare Consensus Network classifiers trained with cooperative optimization (i.e., $\min_{C,P_{1..M}} \mathcal{L}_\mathcal{C}$) to models with the same hyper-parameters but trained non-cooperatively (i.e., $\min_C \mathcal{L}_\mathcal{C}$). As shown in Table 2, the cooperative variant produces higher-score classifiers than the non-cooperative one ($p < 0.001$ on a 2-tailed t-test with 18 DoF). With the cooperative optimization setting, the ePhysicians are encouraged towards producing representations both indistinguishable (by the discriminator) and beneficial (for the classifier). Although the representations might agree less with each other, they could contain more *complementary* information, leading to better overall classifier performances.

In other experiments, all of our models use cooperative optimization.

Optimization	Micro F1	Macro F1
Non-coop	.6696 ± .0511	.6743 ± .0493
Cooperative	**.7995 ± .0450**	**.7998 ± .0449**

Table 2: Comparison of models optimized in cooperative and non-cooperative manner.

4.3 Effectiveness of Agreement Among Modalities

In this and the next experiment, we illustrate the effectiveness of our models on different configurations of modalities in an ablation study. We show that our models work well because of the effectiveness of the "consensus between modalities" scheme.

In this experiment, we compare our Consensus Network models (i.e., with agreements) with fully-connected neural network classifiers (i.e., without agreements) taking the same partial input features. The networks are all simple multiple layer perceptrons containing the same total number of neurons as the 'classifier pipeline' of our models (i.e., ePhysicians plus the classifier)[4] with batch normalization between hidden layers. A few observations could be made from Table 3:

1. Some features from particular modalities are more expressive than others. For example, acoustic features could be used for building better classifiers than those in the semantic ($p = .005$ for 2-tailed t-test with 18 DoF) or syntactic modality ($p < .001$ for 2-tailed t-test with 18 DoF). More specifically, the syntactic features themselves do not tell much. We think this is because the syntactic features are largely based on the contents of the speech, and remain similar across speakers. For example, almost none of the speakers asked questions, giving zero values in "occurrences" of corresponding syntactic patterns.

2. Our model is able to utilize multiple modalities better than MLP. For MLP classifiers, combining features from different modalities does not always give better models. The syntactic modality features confuse MLP and "drag down" the accuracy. However, our models built with the consensus framework are able to utilize the less informative features from additional modalities. In all scenarios using two modalities, our models achieve accuracies higher than neural networks trained on *any* of the two individual modalities.

3. Given the same combinations of features, letting neural networks produce representation in agreement *does* improve the accuracy in all four scenarios[5].

4.4 Dividing Features by Natural Modalities is Preferred

This is the second ablation study towards modality arrangement. We show that dividing features into subsets according to their natural modalities (i.e., the categories in which they are defined) is better than doing so randomly.

In this experiment, we train CNs on features grouped by either their natural modalities, or randomly divided. For natural groupings, we try to

---

[4] For example, for models taking in two modalities, if our model contain ePhysicians with one layer of 20 hidden neurons, the interpretation vector dimension 10, and classifier 5 neurons, then the benchmarking neural network contains three hidden layers with [20×2, 10×2, 5] neurons.

[5] $p = 3 \times 10^{-12}$ on syntactic+semantic features, $p = 0.044$ on acoustic + semantic, $p = 0.005$ on acoustic + syntactic, and $p = 0.046$ on all modalities. All 18 DoF one-tailed t-tests.

Models (Modality)	Accuracy
MLP (Acoustic)	.7519 ± .0245
MLP (Syntactic)	.5222 ± .0180
MLP (Semantic)	.6987 ± .0278
MLP (Syntactic + Semantic)	.5819 ± .0216
CN (Syntactic + Semantic)	.7257 ± .0344
MLP (Acoustic + Semantic)	.7002 ± .1128
CN (Acoustic + Semantic)	.7542 ± .0433
MLP (Acoustic + Syntactic)	.6776 ± .0952
CN (Acoustic + Syntactic)	.7574 ± .0361
MLP (All 3 modalities)	.7528 ± .0520
CN (All 3 modalities)	**.7995** ± **.0450**

Table 3: Performance comparison between Consensus Networks and fully-connected neural network classifiers having certain modality information.

let each group contain comparable number of features, resulting in the following settings:

- Two groups, natural: (a) acoustic + semantic, 216 features; (b) syntactic + PoS, 197 features.
- Three groups, natural: (a) acoustic, 185 features; (b) semantic and PoS, 111 features; and (c) syntactic, 117 features. This is the default configuration used in other experiments in this paper.

For random grouping, we divide the features into almost equal-sized 2/3/4 groups randomly. As shown in Table 4. The two natural modality division methods produce higher accuracies than those produced by any of the random modality division methods.

Modality division method	Accuracy
Three groups, random	.7408 ± .0340
Two groups, random	.7623 ± .0164
Four groups, random	.7666 ± .0141
Two groups, natural	.7769 ± .0449
Three groups, natural	**.7995** ± **.0450**

Table 4: Performance comparison between different modality division methods, sorted by accuracy.

4.5 Visualizing the Representations

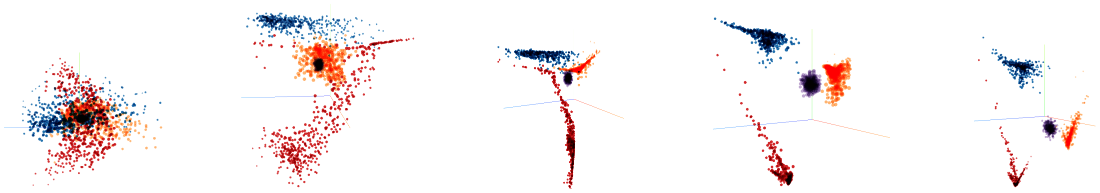
To further understand what happens inside consensus network models during training, we visualize the representation vectors with PCA. Figure 2 consists of the visualizations drawn from an arbitrary trial in training the model. Each representation vector is shown on the figure as a data point, with its color representing its originating modality (including the noise modality).

Several common themes could be observed:

1. *The clusters are symmetric*. Initially the configurations of representations largely depend on the initializations of the network parameters. Gradually the representations of the same modality tend to form clusters. Optimizing the ePhysicians towards both targets make they compress modalities into representations which are symmetrical in an *aggregate* manner.
2. *The agreements are simple*. The variances explained by the first a few principal components usually increase as the optimizations proceed. When distilling information relevant to detection, the agreement tend to become simple.
3. *The agreements are imperfect*. As shown in Figure 2, the modal representations do not overlap. Also, the discriminator loss is low (usually at 10^{-3} when training is done). This confirms that these representations are still easily distinguishable. This may because the modalities inherently have some *complementary* information, leading to the ePhysicians projecting the modalities differently.
4. *The representations are complex*. Their shapes do not resemble the noise representations (Gaussian) lying at the center of the three petals. This shows that the representations are not simply Gaussian.
5. *The accuracy increases*. The accuracy in validation set generally increases as the training proceeds. Note that the distributions of representation vectors are increasingly similar in shape but remains distinct in spatial allocations. This confirms our conjecture that the information about cognitive impairment resides in complicated details instead of superficial statistics, which neural networks could represent.

4.6 Evaluation Against Benchmarks

With the previous sets of experiments, we have a best working architecture. We now evaluate it against traditional classifiers, which are used by



(a) Step 5
$\mathcal{L}_{\mathcal{D}} = 1.34$
Val accr .74%
Variance 71.2%

(b) Step 10
$\mathcal{L}_{\mathcal{D}} = 1.32$
Val accr .76%
Variance 72.9%

(c) Step 20
$\mathcal{L}_{\mathcal{D}} = 1.17$
Val accr .77%
Variance 77.1%

(d) Step 30
$\mathcal{L}_{\mathcal{D}} = 0.89$
Val accr .79%
Variance 77.0%

(e) Step 40
$\mathcal{L}_{\mathcal{D}} = 0.64$
Val accr .79%
Variance 80.8%

Figure 2: Initially, the representations from the three modalities are mixed. As the training go on, the three modalities gradually form three symmetric "petals", while the (Gaussian) noise modality stays in the center. These petals do not overlap, as they contain complementary information when combined and passed into the classifier. Instead, their distributions become symmetric. (Best viewed in colours.)

the state-of-the-art papers (Hernández-Domínguez et al., 2018; Santos et al., 2017; Fraser et al., 2015) on our 413 features. Note that the results could be different from what they reported, because the feature sets are different.

We test several traditional supervised learning benchmark algorithms here: support vector machine (SVM), quadratic discriminant analysis (QDA), random forest (RF), and Gaussian process (GP). For completeness, multiple layer perceptrons (MLPs) containing all features as inputs are also mentioned in Table 5. On the binary classification task (healthy control vs. dementia), our model does better than them all.

Classifier	Micro F1	Macro F1
SVM	.4810 ± .0383	.6488 ± .0329
QDA	.5243 ± .0886	.5147 ± .0904
RF	.6184 ± .0400	.6202 ± .0422
GP	.6775 ± .0892	.6873 ± .0819
MLP	.7528 ± .0520	.7561 ± .0444
CN	**.7995 ± .0450**	**.7998 ± .0449**

Table 5: Comparison with different traditional classifiers in AD:Control classification task. In particular, our model has higher accuracy than the best traditional classifier, MLP.

5 Conclusion and Future Works

We introduce the consensus network framework, in which neural networks are encouraged to compress various modalities into indistinguishable representations ('interpretation vectors'). We show that consensus networks, with the noise modality and cooperative optimization, improve upon traditional neural network baselines given the same features. Specifically, with all 413 linguistic features, our models outperform fully-connected neural networks and other traditional classifiers used by state-of-the-art papers.

In the future, the "agreement among modalities" concept may be applied to design objective functions for training classifiers in various tasks, and from other data sets (for example, education and occupation "modalities" for the bank marketing prediction task). Furthermore, the mechanisms that represent linguistic features into symmetric spaces should be analyzed within the context of explainable AI.

References

Mahsa Baktashmotlagh, Mehrtash T. Harandi, Brian C. Lovell, and Mathieu Salzmann. 2013. Unsupervised Domain Adaptation by Domain Invariant Projection. In *International Conference on Computer Vision (ICCV)*, pages 769–776. IEEE.

James T Becker, François Boiler, Oscar L Lopez, Judith Saxton, and Karen L McGonigle. 1994. The natural history of Alzheimer's disease: description of study cohort and accuracy of diagnosis. *Archives of Neurology*, 51(6):585–594.

Suzanna Becker and Geoffrey E Hinton. 1992. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161.

Jon Atli Benediktsson, Johannes R Sveinsson, Okan K Ersoy, and Philip H Swain. 1997. Parallel consensual neural networks. *IEEE Transactions on Neural Networks*, 8(1):54–64.

Ernst R Berndt and Iain M Cockburn. 2013. Price indexes for clinical trial research: a feasibility study. Technical report, National Bureau of Economic Research.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Pro-*

ceedings of the eleventh annual conference on Computational learning theory, pages 92–100. ACM.

Jieun Chae and Ani Nenkova. 2009. Predicting the fluency of text with shallow structural features: case studies of machine translation and human-written text. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 139–147. Association for Computational Linguistics.

Xilun Chen and Claire Cardie. 2018. Multinomial Adversarial Networks for Multi-Domain Text Classification. In *Proc. of NAACL*.

Michael A Covington and Joe D McFall. 2010. Cutting the Gordian knot: The moving-average type–token ratio (MATTR). In *Journal of quantitative linguistics*, volume 17, pages 94–100. Taylor & Francis.

Kathleen C Fraser, Jed A Metlzer, and Frank Rudzicz. 2015. Linguistic Features Identify Alzheimer's Disease in Narrative Speech. *Journal of Alzheimer's Disease 49(2016)407-422*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, Urun Dogan, Marius Kloft, Francesco Orabona, and Tatiana Tommasi. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of Advances in neural information processing systems*, pages 2672–2680.

Curry I Guinn and Anthony Habash. 2012. Language Analysis of Speakers with Dementia of the Alzheimer's Type. In *AAAI Fall Symposium: Artificial Intelligence for Gerontechnology*, pages 8–13. Menlo Park, CA.

Pamela Herd, Deborah Carr, and Carol Roan. 2014. Wisconsin longitudinal study (WLS). In *International journal of epidemiology*, volume 43, pages 34–41. Oxford University Press.

Laura Hernández-Domínguez, Sylvie Ratté, Gerardo Sierra-Martínez, and Andrés Roche-Bergua. 2018. Computer-based evaluation of Alzheimer's disease and mild cognitive impairment patients during a picture description task. *Alzheimer's and Dementia: Diagnosis, Assessment and Disease Monitoring*, 10(3):260–268.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language Learning with Adversarial Neural Networks: Application to Community Question Answering. In *CoNLL-2017*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Bernd Kortmann and Benedikt Szmrecsanyi. 2004. Global synopsis: morphological and syntactic variation in English. *A handbook of varieties of English*, 2:1142–1202.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1–10.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Andrea Moro, Valentina Bambini, Marta Bosia, Simona Anselmetti, Roberta Riccaboni, Stefano Cappa, Enrico Smeraldi, and Roberto Cavallaro. 2015. Detecting syntactic and semantic anomalies in schizophrenia. *Neuropsychologia*, 79.

Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. 2017. Unified Deep Supervised Domain Adaptation and Generalization. In *Internatilnal Conference on Computer Vision (ICCV)*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Ziad S Nasreddine, Natalie A Phillips, Valérie Bédirian, Simon Charbonneau, Victor Whitehead, Isabelle Collin, Jeffrey L Cummings, and Howard Chertkow. 2005. The Montreal Cognitive Assessment, MoCA: a brief screening tool for mild cognitive impairment. *Journal of the American Geriatrics Society*, 53(4):695–699.

Zeinab Noorian, Chloé Pou-Prom, and Frank Rudzicz. 2017. On the importance of normative data in speech-based assessment. In *Proceedings of Machine Learning for Health Care Workshop (NIPS MLHC)*.

Gordon E Peterson and Harold L Barney. 1952. Control methods used in a study of the vowels. In *The Journal of the acoustical society of America*, volume 24, pages 175–184. ASA.

Chloé Pou-Prom and Frank Rudzicz. 2018. Learning multiview embeddings for assessing dementia. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A Rey. 1941. L'examen psychologique dans les cas d'encéphalopathie traumatique. (Les problems.). [The psychological examination in cases of traumatic encepholopathy. Problems.]. *Archives de Psychologie*, 28:215–285.

Brian Roark, Margaret Mitchell, and Kristy Hollingshead. 2007. Syntactic complexity measures for detecting mild cognitive impairment. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 1–8. Association for Computational Linguistics.

Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey Kaye. 2011. Spoken Language Derived Measures for Detecting Mild Cognitive Impairment. In *IEEE transactions on audio, speech, and language processing*. U.S. National Library of Medicine.

Virginia R de Sa. 1994. Learning classification with unlabeled data. In *Proceedings of Advances in neural information processing systems*, pages 112–119.

Leandro Santos, Edilson Anselmo Corrêa Júnior, Osvaldo Oliveira Jr, Diego Amancio, Letícia Mansur, and Sandra Aluísio. 2017. Enriching Complex Networks with Word Embeddings for Detecting Mild Cognitive Impairment from Speech Transcripts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1284–1296, Vancouver, Canada. Association for Computational Linguistics.

Greta Szatloczki, Ildiko Hoffmann, Veronika Vincze, Janos Kalman, and Magdolna Pakaski. 2015. Speaking in Alzheimer's Disease, is That an Early Sign? Importance of Changes in Language Abilities in Alzheimer's Disease. *Frontiers in Aging Neuroscience*, 7:195.

Vanessa Taler, Ekaterini Klepousniotou, and Natalie A Phillips. 2009. Comprehension of lexical ambiguity in healthy aging, mild cognitive impairment, and mild Alzheimer's disease. In *Neuropsychologia*, volume 47, pages 1332–1343. Elsevier.

Davy Weissenbacher, Travis A Johnson, Laura Wojtulewicz, Amylou Dueck, Dona Locke, Richard Caselli, and Graciela Gonzalez. 2016. Automatic prediction of linguistic decline in writings of subjects with degenerative dementia. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1198–1207.

Maria Yancheva, Kathleen Fraser, and Frank Rudzicz. 2015. Using linguistic features longitudinally to predict clinical scores for Alzheimer's disease and related dementias. In *Proceedings of the 6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT 2015)*, pages 134–139, Dresden, Germany. Association for Computational Linguistics.

Maria Yancheva and Frank Rudzicz. 2016. Vector-space topic models for detecting Alzheimer's disease. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2337–2346.

Victor H Yngve. 1960. A model and an hypothesis for language structure. In *Proceedings of the American philosophical society*, volume 104, pages 444–466. JSTOR.

Shunan Zhao, Frank Rudzicz, Leonardo G. Carvalho, Cesar Marquez-Chin, and Steven Livingstone. 2014. Automatic detection of expressed emotion in Parkinson's Disease. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (May):4813–4817.

Luke Zhou, Kathleen C. Fraser, and Frank Rudzicz. 2016. Speech recognition in Alzheimers disease and in its assessment. In *Interspeech 2016*, pages 1948–1952.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

A Appendices

Linguistic Features

Acoustic

- The fluency of speech. We quantify it with phonation rate, duration of pauses, and number of filled pauses (e.g., *"um"*) of various lengths.
- Following the convention of speech processing literatures (Zhou et al., 2016; Yancheva et al., 2015; Zhao et al., 2014), we compute Mel-scaled cepstral coefficients (MFCCs) containing the amount of energy in 12 different frequency intervals for each time frame of 40 milliseconds, as well as their first- and second-order derivatives. We calculate the mean, variance, kurtosis, and skewness of the MFCCs and include them as features.

Semantic and Lexical

- Lexical norms, including age-of-acquisition, familiarity, imageability, and frequency (Taler et al., 2009). These are averaged over the entire transcript and specific PoS categories, respectively.
- Lexical richness, including moving-average type-token ratio over different window sizes

(Covington and McFall, 2010), Brunet's index, and Honoré's statistics (Guinn and Habash, 2012).

- Cosine similarity statistics (minimum, maximum, average, etc.) between pairs of utterances (represented as sparse vectors based on lemmatized words)
- Average word length, counts of total words, not-in-dictionary words, and fillers. The dictionary we use contains around 98,000 entries, including common words, plural forms of countable nouns, possessive forms of subjective nouns, different tenses of verbs, etc.

Syntactic

- Composition of languages. We describe it by several features, including the average proportion of context-free grammar (CFG) phrase types[6], the rates of these phrase types[7], and the average phrase type length[8] (Chae and Nenkova, 2009)
- The syntactic complexity of languages. We characterize it by the average heights of the context-free grammar (CFG) parse trees, across all utterances in each transcript. Each tree comes from an utterance parsed by a context free grammar parser (`LexParser` implemented in Stanford CoreNLP (Manning et al., 2014)). In addition, we compute the Yngve scores statistics of CFG parse trees (Yngve, 1960; Roark et al., 2007), where Yngve score is the degree of left-branching of each node in a parsed tree.
- Syntactic components. We describe them by including the number of occurrences of a set of 104 context-free production rules (e.g.,`S->VP`) in the CFG parse trees.

Part-of-speech

- The number of occurrences of part-of-speech (PoS) tags from Penn-treebank[9].
- The ratio of occurrences of several PoS tags, including noun-pronoun ratio.
- Number of occurrences of words in each of the five categories: subordinate (e.g: "because", "since", etc.), demonstratives (e.g: "this", "that"), function (e.g: words with PoS tag CC, DT, and IN), light verbs (e.g: "be", "have"), and inflected verbs (words with PoS tag VBD, VBG, VBN, and VBZ), borrowing the categorization method in Kortmann and Szmrecsanyi (2004)

---

[6] number of words in these types of phrases, divided by the total number of words in the transcript

[7] number of occurrences in a transcript, divided by the total number of words in the transcript

[8] number of words belonging to this phrase type in a transcript, divided by the occurrences of this phrase type in a transcript

[9] Using `https://spacy.io`

Relation Extraction using Explicit Context Conditioning

Gaurav Singh[*]
University College London
g.singh@cs.ucl.ac.uk

Parminder Bhatia
Amazon
parmib@amazon.com

Abstract

Relation Extraction (RE) aims to label relations between groups of marked entities in raw text. Most current RE models learn context-aware representations of the target entities that are then used to establish relation between them. This works well for intra-sentence RE and we call them first-order relations. However, this methodology can sometimes fail to capture complex and long dependencies. To address this, we hypothesize that at times two target entities can be explicitly connected via a context token. We refer to such indirect relations as second-order relations and describe an efficient implementation for computing them. These second-order relation scores are then combined with first-order relation scores. Our empirical results show that the proposed method leads to state-of-the-art performance over two biomedical datasets.

1 Introduction

There are wide applications for Information Extraction in general (Jin et al., 2018) and Relation Extraction (RE) in particular, one reason why relation extraction continues to be an active area of research (Bach and Badaskar, 2007; Kambhatla, 2004; Kumar, 2017). Traditionally, a standard RE model would start with entity recognition and then pass the extracted entities as inputs to a separate relation extraction model, which meant that the errors in entity recognition were propagated to RE. This problem was addressed by end-to-end models (Miwa and Bansal, 2016; Zheng et al., 2017; Adel and Schütze, 2017; Bhatia et al., 2018) that jointly learn both NER and RE.

Generally, these models consist of an encoder followed by a relationship classification (RC) unit (Verga et al., 2018; Christopoulou et al., 2018; Su et al., 2018). The encoder provides context-aware vector representations for both target entities, which are then merged or concatenated before being passed to the relation classification unit, where a two layered neural network or multi-layered perceptron classifies the pair into different relation types.

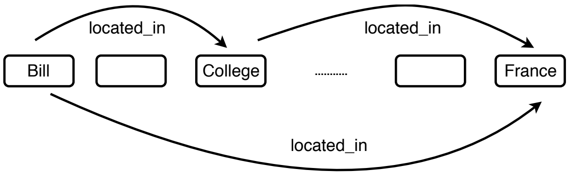


Figure 1: Pictorial representation of a second-order relation between two entities (*Bill* & *France*) connected by a context token (*College*).

Such RE models rely on the encoder to learn 'perfect' context-aware entity representations that can capture complex dependencies in the text. This works well for intra-sentence relation extraction i.e. the task of extracting relation from entities contained in a sentence (Christopoulou et al., 2018; Su et al., 2018). As these entities are closer together, the encoder can more easily establish connection based on the language used in the sentence. Additionally, these intra-sentence RE models can use linguistic/syntactical features for an improved performance e.g. shortest dependency path.

Unfortunately, success in intra-sentence RE has not been replicated for cross-sentence RE. As an example, a recent RE method called BRAN (Verga et al., 2018) proposed to use encoder of Transformer (Vaswani et al., 2017) for obtaining token representations and then used these representations for RE. However, our analysis revealed that it wrongly marks many cross-sentence relations as negative, especially when the two target entities were connected by a string of logic spanning over

---

[*] G. Singh was an intern at Amazon at the time of work

Proceedings of NAACL-HLT 2019, pages 1442–1447
Minneapolis, Minnesota, June 2 - June 7, 2019.

multiple sentences. This showed that reliance on the encoder alone to learn these complex dependencies does not work well.

In this work we address this issue of over-reliance on the encoder. We propose a model based on the hypothesis that two target entities, whether intra-sentence or cross-sentence, could also be explicitly connected via a third context token (Figure 1). More specifically, we find a token in the text that is most related to both target entities, and compute the score for relation between the two target entities as the summation of their relation scores with this token. We refer to these relations as second-order relations. At the end, we combine these second-order scores with first-order scores derived from a traditional RE model, and achieve state-of-the-art performance over two biomedical datasets. To summarize the contribution of this work:

1. We propose using second-order relation scores for improved relation extraction.

2. We describe an efficient algorithm to obtain second-order relation scores.

2 Background

In this section we describe the encoder and relation classification unit of a SOTA RE model called BRAN (Verga et al., 2018). This model computes relation scores between two entities directly from their representations, therefore we refer to these as first-order relation scores.

2.1 Transformer Encoder

BRAN uses a variant of *Transformer* (Vaswani et al., 2017) encoder to generate token representations.

The encoder contains repeating blocks and each such block consists of two sublayers: multi-head self-attention layer followed by position-wise convolutional feedforward layer. There are residual connections and layer normalization (Ba et al., 2016) after each sublayer. The only difference from a standard transformer-encoder is the presence of a convolution layer of kernel width 5 between two consecutive convolution layers of kernels width 1 in the feedforward sublayer. It takes as input word embeddings that are added with positional embeddings (Gehring et al., 2017).

2.2 First-Order Relations

The relation classification unit takes as input token representations from the described encoder. These are then passed through two MLPs to generate head/tail representation e_i^{head}/e_i^{tail} for each token corresponding to whether it serves the first (head) or second (tail) position in the relation.

$$e_i^{head} = W_{head_2}(ReLU(W_{head_1} b_i)) \quad (1)$$

$$e_i^{tail} = W_{tail_2}(ReLU(W_{tail_1} b_i)) \quad (2)$$

where b_i is the representation of the i_{th} token generated by the encoder.

These are then combined with a bi-affine transformation operator to compute a $N \times R \times N$ tensor A of pairwise affinity scores for every token pair and all relation types, scoring all triplets of the form $(head, relation, tail)$:

$$A_{irj} = (e_i^{head} L) e_j^{tail}, \quad (3)$$

where L is a learned tensor of dimension $d \times R \times d$ to map pairs of tokens to scores over each of the R relation types and d is the dimension of head/tail representations. Going forward we will drop the subscript r for clarity.

The contributions from different mention pairs are then aggregated to give us **first-order** relation scores. This aggregation is done using $LogSumExp$, which is a smooth approximation of max that prevents sparse gradients:

$$\text{scores}^{(1)}(p^{head}, p^{tail}) = \log \sum_{\substack{i \in P^{head} \\ j \in P^{tail}}} \exp(A_{ij}), \quad (4)$$

where $P^{head}(P^{tail})$ contains mention indices for head (tail) entity.

3 Proposed Second-Order Relations

In this section we describe in detail our proposed method to obtain second-order relation scores.

We use the encoder described in Sec 2.1 for getting token representations. These token representations are then passed through two MLPs (as in previous section), which generate head/tail representations for each token corresponding to whether it serves the first or the second position in the relation. We used a separate set of these head/tail MLPs for second-order scores than the ones used for getting first-order scores. This was

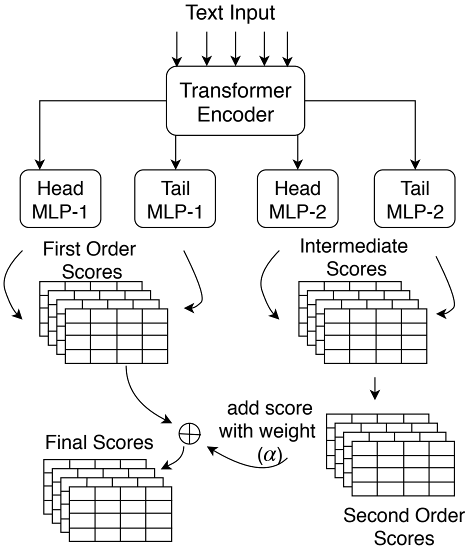


Figure 2: Schematic of the model architecture.

motivated by the need for representations focused on establishing relations with context tokens, as opposed to first-order relations (described in previous section) that attempt to directly connect two target entities.

The head and tail representations are then combined with a $d \times R \times d$ bilinear transformation tensor M to get a $N \times R \times N$ tensor B of intermediate pairwise scores.

$$B_{ij} = (e_i^{head} M) e_j^{tail} \tag{5}$$

After that we arbitrarily define the scores between tokens i and j when *conditioned* on a context token k as the sum of the scores of relations (i, k) and (k, j).

$$C(i, j|k) = B_{ik} + B_{kj} \tag{6}$$

These context-conditioned scores are computed for every triplet of the form (i, j, k).

Second-order relation scores are then derived by aggregating over all context tokens and mention pairs using $LogSumExp$.

$$\text{scores}^{(2)}(p^{head}, p^{tail}) = \log \sum_{\substack{k \\ i \in P^{head} \\ j \in P^{tail}}} \exp(C(i, j|k)) \tag{7}$$

Here $LogSumExp$ ensures that one specific mention pair connected via one specific context token is responsible for the relation. This is equivalent to max-pooling over all context tokens that could potentially connect the two target entities, which reduces over-fitting by removing contributions from noisy associations of the target entities with random tokens e.g. stopwords.

It is important to mention that a naive implementation of this would require $O(N^3)$ space to store context-conditioned scores between all pairs of token i.e. $C(i, j|k)$. To address this, we describe an efficient method in Section 3.1 that avoids explicitly storing these.

At the end, the final score for relation between two entities is given as a weighted sum of first (eq. 4) and second (eq. 7) order scores.

$$\begin{aligned} \text{scores}(p^{head}, p^{tail}) = {} & \text{scores}^{(1)}(p^{head}, p^{tail}) \\ & + \alpha * \text{scores}^{(2)}(p^{head}, p^{tail}) \end{aligned} \tag{8}$$

where α is a hyper-parameter denoting the weight of second-order relation scores.

Entity Recognition. We do entity recognition alongside relation extraction, as the transfer of knowledge between the two tasks has been shown to improve relation extraction performance (Verga et al., 2018; Miwa and Bansal, 2016). For this we feed encoder output b_i to a linear classifier W_{er} that predicts scores for each entity type.

$$d_i = W_{er}(b_i) \tag{9}$$

3.1 Efficient Implementation

The problem lies in storing score for every intermediate relation of the form $C(i, j|k)$, as that would require space of the order $O(N^3)$. Here we describe a space-time efficient method to compute final second-order relation scores.

The intermediate scores (eq. 5) are a tensor of dimension $b \times N \times R \times N$ comprising of pairwise scores for b batches. We create two tensors out of these intermediate scores, namely T_1 and T_2. T_1 computes the exponential of indices $(\{b, i \in P^{head}, j \in \mathcal{C}, R\})$ corresponding to pairwise scores between head entity and all the context tokens ($\mathcal{C}$ i.e., all the tokens except the two target entities), and sets other indices to 0. Similarly, T_2 computes exponential of indices $(\{b, i \in P^{tail}, j \in \mathcal{C}, R\})$ corresponding to pairwise scores between tail entity and context tokens, setting all other indices to 0. To get the context conditioned scores one needs to compute the batch product of

Data	Model	Pr	Re	F1
DCN	BRAN	0.614	0.850	0.712
	+ SOR	**0.643**	**0.879**	**0.734**
i2b2	HDLA	0.378	**0.422**	0.388
	BRAN	0.396	0.403	0.395
	+ SOR	**0.424**	0.419	**0.407**
CDR	BRAN	0.552	0.701	0.618
	+ SOR	0.552	0.701	0.618

Table 1: The performance of proposed model using second-order relations. BRAN is the model used in (Verga et al., 2018) and +SOR is our proposed model with second-order relations. Results for HDLA are quoted from Chikka and Karlapalem (2018). Results on CDR are identical for both BRAN and our proposed model as α was set to 0 after tuning over the dev set at which point our model is the same as BRAN. All the metrics are macro in nature.

R two dimensional slices of size $N \times N$ from T_1 and T_2 along the dimension of context, but this would be sequential in R. Instead we can permute T_1 and T_2 to $b \times R \times N \times N$ followed by reshaping to $bR \times N \times N$ and perform a batch matrix multiplication along the context dimension to get $bR \times N \times N$. Afterwards, we can sum along the last two dimensions to get a tensor of size bR. Finally, we can take the log succeeded by reshaping to $b \times R$ to obtain second-order scores.

4 Experimentation

4.1 Datasets

We have used three datasets in this work, i2b2 2010 challenge (Uzuner et al., 2011) dataset, a de-identified clinical notes dataset and a chemical-disease relations dataset known as BioCreative V (CDR) (Li et al., 2016; Wei et al., 2016).

First is a publicly available subset of the dataset used for the i2b2 2010 challenge. It consists of documents describing relations between different diseases and treatments. Out of the 426 documents available publicly, 10% are used each for both dev and test and the rest for training. There are 3244/409 relations in train/test set and 6 predefined relations types including one negative relation e.g. TrCP (**Tr**eatment Causes **Pr**oblem), TrIP (Tr Improves Pr), TrWP (Tr Worsens Pr). We have used the exact same dataset as Chikka *et al.* (Chikka and Karlapalem, 2018).

Second is a dataset of 4200 de-identified clinical notes (DCN), with vocabulary size of 50K. It contains approximately 170K relations in the train set and 50K each in dev/test set. There are 7 predefined relation types including one negative relation type. These are mostly between medication name and other entities e.g. "*paracetamol* ***every*** *day*","*aspirin* ***with_dosage*** *100mg*". The frequency of different relations in this dataset is fairly balanced.

Third is a widely used and publicly available dataset called CDR (Li et al., 2016; Wei et al., 2016). It was derived from Comparative Toxicogenomics Database (CTD) and contains documents describing the effect of chemicals (drugs) on diseases. There are only two relation types between any two target entities i.e. positive/negative and these relations are annotated at the document level. It consists of 1500 documents that are divided equally between train/dev/test sets. There are 1038/1012/1066 positive and 4280/4136/4270 negative relations in train/dev/test sets respectively. We performed the same preprocessing as done in BRAN (Verga et al., 2018).

4.2 Experimental Settings

We jointly solve for NER and RE tasks using cross-entropy loss. During training we alternate between mini-batches derived from each task. We fix the learn rate to 0.0005 and clip gradient for both tasks at 5.0. For training, we used adams optimizer with $\beta = (\beta_1, \beta_2) = (0.1, 0.9)$. We tune over the weight of second-order relations denoted by α to get $\alpha = 0.2$ for DCN/i2b2 and $\alpha = 0.0$ for CDR dataset.

Our final network had two encoder layers, with 8 attention heads in each multi-head attention sublayer and 256 filters for convolution layers in position-wise feedforward sublayer. We used dropout with probability 0.3 after: embedding layer, head/tail MLPs, output of each encoder sublayer. We also used a word dropout with probability 0.15 before the embedding layer.

4.3 Results

To show the benefits of using second-order relations we compared our model's performance to BRAN. The two models are different in the weighted addition of second-order relation scores. We tune over this weight parameter on the dev set and observed an improvement in MacroF1 score from 0.712 to 0.734 over DCN data and from 0.395 to 0.407 over i2b2 data. For further comparison a recently published model called HDLA (Chikka and Karlapalem, 2018) reported a macro-

F1 score of 0.388 on the same i2b2 dataset. It should be mentioned that HDLA used syntactic parsers for feature extraction but we do not use any such external tools.

In the case of CDR dataset we obtained $\alpha = 0$ after tuning, which means that the proposed model converged to BRAN and the results were identical for the two models. These results are summarized in Table 1.

4.4 Ablation Study

We experimented with different ablations of BRAN and noticed an improvement in results for DCN dataset upon removing multi-head self-attention layer. Also, our qualitative analysis showed that relations between distant entities were often wrongly marked negative. We attribute these errors to the token representations generated by the encoder. To this effect, our experiments showed that incorporating relative position (Shaw et al., 2018) information in the encoder to improve token representations does not lead to superior RE. Separately, we observed that the proposed method improved results when using a standard CNN encoder as well.

5 Conclusions and Future Work

We proposed a method that uses second-order relation scores to capture long dependencies for improved RE. These relations are derived by explicitly connecting two target entities via a context token. These second-order relations (SORs) are then combined with traditional relation extraction models, leading to state-of-the-art performance over two biomedical datasets. We also describe an efficient implementation for obtaining these SORs.

Despite restricting ourselves to SORs, it should be noted that the proposed method can be generalized to third and fourth order relations. We conjecture that these may serve well for cross-sentence relation extraction in long pieces of texts. Also, we only considered one relation type between each entity and bridge token but it is possible, and very likely that two different relation types may lead to a third relation type. We will explore both these aspects in future work.

Acknowledgements

We would like to thank Busra Celikkaya and Mohammed Khalilia of Amazon, Zahra Sabetsarvestani and Sebastian Riedel of University College London and the anonymous reviewers of NAACL for their valuable feedback on the paper.

References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Nguyen Bach and Sameer Badaskar. 2007. A survey on relation extraction. *Language Technologies Institute, Carnegie Mellon University*.

Parminder Bhatia, Busra Celikkaya, and Mohammed Khalilia. 2018. End-to-end joint entity extraction and negation detection for clinical text. *arXiv preprint arXiv:1812.05270*.

Veera Raghavendra Chikka and Kamalakar Karlapalem. 2018. A hybrid deep learning approach for medical relation extraction. *arXiv preprint arXiv:1806.11189*.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 81–88.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Mengqi Jin, Mohammad Taha Bahadori, Aaron Colak, Parminder Bhatia, Busra Celikkaya, Ram Bhakta, Selvan Senthivel, Mohammed Khalilia, Daniel Navarro, Borui Zhang, et al. 2018. Improving hospital mortality prediction with medical named entities and multimodal learning. *arXiv preprint arXiv:1811.12276*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Shantanu Kumar. 2017. A survey of deep learning methods for relation extraction. *arXiv preprint arXiv:1705.03645*.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gur, Huan Sun, and Xifeng Yan. 2018. Global relation embedding for relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2016. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database*, 2016.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL 2017*.

Conversation Model Fine-Tuning for Classifying Client Utterances in Counseling Dialogues

Sungjoon Park [1], **Donghyun Kim** [2], **Alice Oh** [1]
[1] School of Computing, KAIST, Republic of Korea
[2] Trost, Humart Company, Inc., Republic of Korea
sungjoon.park@kaist.ac.kr, kdhyeon821@gmail.com
alice.oh@kaist.edu

Abstract

The recent surge of text-based online counseling applications enables us to collect and analyze interactions between counselors and clients. A dataset of those interactions can be used to learn to automatically classify the client utterances into categories that help counselors in diagnosing client status and predicting counseling outcome. With proper anonymization, we collect counselor-client dialogues, define meaningful categories of client utterances with professional counselors, and develop a novel neural network model for classifying the client utterances. The central idea of our model, ConvMFiT, is a pre-trained conversation model which consists of a general language model built from an out-of-domain corpus and two role-specific language models built from unlabeled in-domain dialogues. The classification result shows that ConvMFiT outperforms state-of-the-art comparison models. Further, the attention weights in the learned model confirm that the model finds expected linguistic patterns for each category.

1 Introduction

Some mental disorders are known to be treated effectively through psychotherapy. However, people in need of psychotherapy may find it challenging to visit traditional counseling services because of time, money, emotional barriers, and social stigma (Bearse et al., 2013). Recently, technology-mediated psychotherapy services emerged to alleviate these barriers. Mobile-based psychotherapy programs (Mantani et al., 2017), fully automated chatbots (Ly et al., 2017; Fitzpatrick et al., 2017), and intervention through smart devices (Torrado et al., 2017) are examples. Among them, text-based online counseling services with professional counselors are becoming popular because clients can receive these services without traveling to an office and with reduced financial burden compared to traditional face-to-face counseling sessions (Hull, 2015).

In text-based counseling, the communication environment changes from face-to-face counseling sessions. The counselor cannot read nonverbal cues from their clients, and the client uses text messages rather than spoken utterances to deliver their thoughts and feelings, resulting in changes of dynamics in the counseling relationship. Previous studies explored computational approaches to analyzing the dynamic patterns of relationship between the counselor and the client by focusing on the language of counselors (Imel et al., 2015; Althoff et al., 2016), clustering topics of client issues (Dinakar et al., 2014), and looking at therapy outcomes (Howes et al., 2014; Hull, 2015).

Unlike previous studies, we take a computational approach to analyze client responses from the *counselor's perspective*. Client responses in counseling are crucial factors for judging the counseling outcome and for understanding the status of the client. So we build a novel categorization scheme of client utterances, and we base our categorization scheme on the cognitive behavioral theory (CBT), a widely used theory in psychotherapy. Also, in developing the categories, we consider whether they are adequate for the unique text-only communication environment, and appropriate for the annotation of the dialogues as training data. Then using the corpus of text-based counseling sessions annotated according to the categorization scheme, we build a novel conversation model to classify the client utterances.

This paper presents the following contributions:

- First, we build a novel categorization method as a labeling scheme for client utterances in text-based counseling dialogues.

Proceedings of NAACL-HLT 2019, pages 1448–1459
Minneapolis, Minnesota, June 2 - June 7, 2019.

- Second, we propose a new model, Conversation Model Fine-Tuning (ConvMFiT) to classify the utterances. We explicitly integrate pre-trained language-specific word embeddings, language models and a conversation model to take advantage of the pre-trained knowledge in our model.
- Third, we empirically evaluate our model in comparison with other models including a state-of-the-art neural network text classification model. Also, we show typical phrases of counselors and clients for each category by investigating the attention layers.

2 Categorization of Client Utterances

Client responses provide essential clues to understanding the client's internal status which can vary throughout counseling sessions. For example, client's responses describing their problems prevail at the early stage of counseling (Hill, 1978; E. Hill et al., 1983), but as counseling progresses, problem descriptions decrease while insights and discussions of plans continue to increase (Seeman, 1949). Client responses can also help in predicting counseling outcomes. For example, a higher proportion of insights and plans in client utterances indicates a high positive effect of counseling (Hill, 1978; E. Hill et al., 1983).

Categorization Objective. Our final aim is to build a machine learning model to classify the client utterances. Thus, the categorization of the utterances should satisfy the following criteria:

- *Suitable for the text-only environment*: Categories should be detected only using the text response of clients.
- *Available as a labeling scheme*: The number of categories should be small enough for manual annotation by counseling experts.
- *Meaningful to counselors*: Categories should be meaningful for outcome prediction or counseling progress tracking.

Previous studies in psychology proposed nine and fourteen categories for client and counselor verbal responses, respectively, by analyzing transcriptions from traditional face-to-face counseling sessions (Hill, 1978; Hill et al., 1981). But these categories were developed for face-to-face spoken interactions, and we found that for online text-only counseling dialogues, these categories are not directly applicable. Using text without non-verbal cues, a client's responses are inherently different from the transcriptions of verbally spoken responses which include categories such as 'silence' (no response for more than 5 seconds) and 'non-verbal referent' (physically pointing at a person). Another relevant study, derived from text-based counseling sessions with suicidal adolescents proposes 19 categories which we judged to be too many to be practical for manual annotation (Kim, 2010).

The last criterion of "meaningful to counselors" is perhaps the most important. To meet that criterion, we base the categorization process on the *cognitive behavioral theory (CBT)* which is the underlying theory behind psychotherapy counseling. The details of using CBT for the categorization process is explained next.

Categorization Process and Results. In developing the categories, we follow the Consensual Qualitative Research method (Hill et al., 1997). Two professional counselors with clinical experience participated in this qualitative research method to define the categorization.

To begin, we randomly sample ten client cases considering demographic information including age, gender, education, job, and previous counseling experiences. We then start the categorization process with the fundamental components of the CBT which are *events*, *thoughts*, *emotions*, and *behavior* (Hill et al., 1981). The professional counselors annotate every client utterance to with those initial component categories with tags that add detail. For example, if an utterance is annotated as 'emotion', we add 'positive/negative' or concrete label such as 'hope'. If these tags are categorized to be a new category, we add that category to the list until it is saturated. When the number of categories becomes more than 40, we define higher level categories that cover the existing categories.

In the second stage, annotators discuss and merge these categories into five high-level categories. Category 1 is *informative responses to counselors*, and category 2 is *providing factual information and experiences*. Categories 3 and 4 are related to the client factors, *expressing appealing problems* and *psychological changes*. The last category is about the *logistics of the counseling sessions* including scheduling the next session. The categories in detail are as follows:

Characteristic	Informative		Client Factors		Process
Category Name	**Factual Information (Fact.)**	**Anecdotal Experience (Anec.)**	**Appealing Problem (Prob.)**	**Psychological Change (Chan.)**	**Counseling Process (Proc.)**
Explanation	Brief mention of categorical information	Clients experience contributing to the appealing problem	Clients factors related to the appealing problem	Statement at the resolution stage of the appealing problem	Statement of counseling structure and relationship
Examples	• Objective Fact • Living conditions • Demographic information • Limited conditions	• Experience with others • Comments from others • Trauma • Interpersonal situations	• Negative Emotion • Cognitive distortion • Interpersonal problems • Family problems	• Positive Prediction • Expectation, Determination • Coping behaviors • Self-awareness	• A message to counselor • Gratitude, Greetings • Time appointment • Questions about the consultation

Table 1: Final Categorization of client utterances. Five categories are discovered, two for informative giving information to a counselor (Factual information, Anecdotal Experience), two for client factors (appealing problems, psychological change), and the last one for counseling process.

- *Factual information (Fact.)* Informative responses to counselor's utterances, including age, gender, occupation, education, family, previous counseling experience, etc.

- *Anecdotal Experience (Anec.)* Responses describing past incidents and current situations related to the formation of appealing problems. Responses include traumatic experiences, interactions with other people, comments from other people, and other anecdotal experiences.

- *Appealing Problems (Prob.)* Utterances addressing the main appealing problem which is yet to be resolved, including client's internal factors or their behaviors related to the problems. Specifically, the utterances include cognition, emotion, physiological reaction, and diagnostic features of the problem and desire to be changed.

- *Psychological Change (Chan.)* Utterances describing insights, cognition of small and big changes in internal factors or behaviors. That is, an utterance at the point where the appealing problem is being resolved.

- *Counseling Processes (Proc.)* Utterances that include the objective of counseling, requests to the counselor, plans about the counseling sessions, and counseling relationship. This category also covers greetings and making an appointment for the next session.

We summarize the category explanations and examples in Table 1.

3 Dataset

In this section, we explain how counseling dialogues differ from general dialogues, describe the dialogues we collected and annotated, and explain how we preprocessed the data.

3.1 Characteristics of Counseling Dialogues

The counseling dialogues consist of multiple turns taken by a counselor and a client, and each turn can contain multiple utterances. Here we describe two unique characteristics of text-based online counseling conversations compared to general non-goal oriented conversations.

Distinctive roles of speakers. Counseling conversations are goal-oriented with the aim to produce positive counseling outcomes, and the two speakers have distinctive roles. The client gives objective information about themselves and subjective experiences and feelings to the counselor to appeal the problems they are suffering from. Then the counselor establishes a therapeutic relationship with the client and elicits various strategies to induce psychological changes in the client. These distinct roles of the conversational participants distinguish counseling conversations from

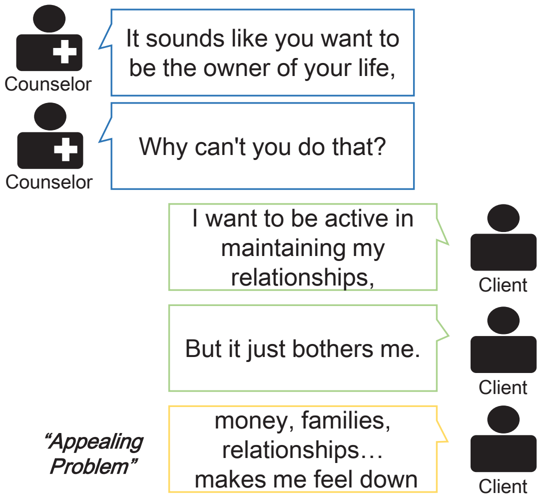


Figure 1: Translated example conversation between a counselor and a client. Each turn can have multiple utterances, and we annotate the client turn at the *utterance*-level. The first two boxes indicate 1) *Counselor's utterance* and the next two are 2) *Client's context utterance*, and the last box is 3) *Client's target utterance*. The annotation for the last client utterance is "appealing problem".

non-goal oriented open-domain response generation modeling.

Multiple utterances in a turn. We define an *utterance* as a single text bubble. Counselors and clients can generate multiple utterances in a turn. Especially in a client turn, various information not to be missed by a counselor may occur across multiple utterances. Thus, we treat every utterance separately, as shown in Fig. 1

3.2 Collected Dataset

Total dialogues. We collect counseling dialogues of clients with their corresponding professional counselors from Korean text-based online counseling platform Trost. [1] Overall, we use 1,448 counseling dialogues which are anonymized before researchers obtain access to the data, by removing personally identifiable information in the dialogues. All meta-data of the dialogues are not provided to the researchers, and named entities such as client's and counselor's names are replaced with random numeric identifiers. The research process including data anonymization and pre-processing is validated by the KAIST Institutional Review Board (IRB).

Labeled Dialogues. We randomly choose and label 100 dialogues with the discovered five categories. Note that we only label client utterances. Based on these categories, five professional counselors annotated their own client's every utterance in the conversations, as shown in Fig. 1. Note that each utterance can have multiple labels if it includes information across multiple categories.

Table. 2 shows the descriptive statistics of our labeled dataset. The first two rows present the average lengths of each utterance of counselors and clients in terms of words and characters, showing there is a small difference between the counselor utterance and the client utterance. On the other hand, the average number of utterances in a single counseling session differs; on average, clients write more utterances than counselors.

	Counselor		Client	
	Mean	Std.	Mean	Std.
# of words	6.26	6.25	5.91	14.62
# of chars	25.71	23.44	24.31	61.10
# of utters	163.2	236.97	238.72	578.49

Table 2: Descriptive statistics of labeled counseling dialogues.

3.3 Preprocessing

We intentionally leave in punctuations and emojis since they can help to infer the categories of the client utterances, treating them as separate tokens. Then we construct triples from the labeled dialogues consisting of 1) counselor's utterances (blue in Fig. 1), 2) client's context utterances (green), and 3) client's target utterances to be categorized. (yellow)

We split the dataset into train, validation, and test sets. Table. 3 shows the number of triples in each set, showing factual information (Fact.) and psychological change (Chan.) categories appear less frequently than the others.

4 Model

We introduce ConvMFiT (Conversation Model Fine-Tuning), fine-tuning pre-trained seq2seq based conversation model to classify the client's utterances.

Background. Our corpus of 100 labeled conversations is not large enough to fully capture the linguistic patterns of the categories without external knowledge. The small size of the dataset is difficult to overcome because the labeling by professional counselors is costly. We found a poten-

---

[1] https://www.trost.co.kr/

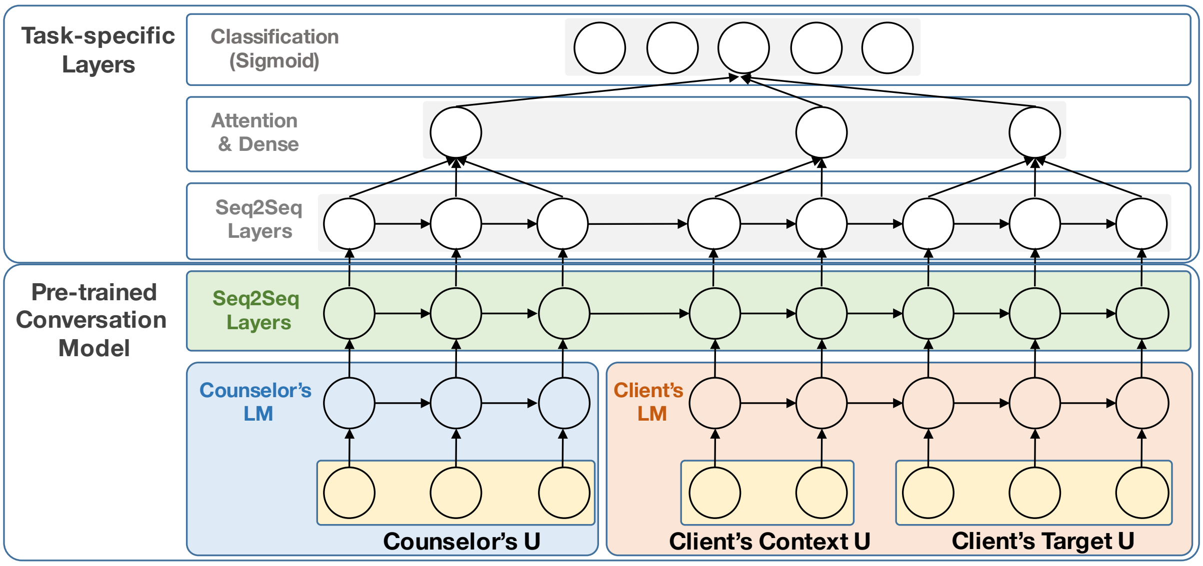


Figure 2: ConvMFiT (Conversation Model Fine-Tuning) model architecture. We use a pre-trained conversation model using seq2seq models. The model is based on pre-trained counselor's and client's language models. (lower colored part) Then, we stack additional task-specific seq2seq layers and classification layers over the conversation model to learn specific features. Between the two layers, attention layer is added to enhance its interpretability.

	Train	Valid	Test	# of labels
Fact.	817	140	172	1129
Anec.	5317	1180	1175	7726
Prob.	4728	1004	981	6713
Chan.	887	211	199	1297
Pros.	4570	964	961	6459
# of triples	14679	3166	3165	21100

Table 3: The number of triples (partner's utterance, client's context utterance, client's target utterance) and corresponding labels for each set. Factual information (Fact.) and psychological change (Chan.) categories have less number of instances compared to the others.

tially effective solution for a small labeled dataset by using pre-trained language models for various NLP tasks (Ramachandran et al., 2017; Howard and Ruder, 2018). Therefore, we focus on transferring knowledge from unlabeled in-domain dialogues as well as a general out-of-domain corpus.

Overview. We illustrate the overall model architecture. We first stack pre-trained seq2seq layers which represent a conversation model (see Fig. 2, lower colored part). Then we stack additional seq2seq layers and classification layers over it to capture task-specific features, and an attention layer is added between the two to enhance the model's interpretability (see Fig. 2, upper white part).

This approach leverages the advantages of using a language model based conversation model for transfer learning. The model can be regarded as an extended version of ULMFiT (Howard and Ruder, 2018) with modifications to fit our task. In ConVMFiT, the model accepts a pre-trained seq2seq conversation model that requires two pre-trained language models, like an encoder and a decoder, learning the dependencies between them on the seq2seq layers. This approach is shown to be effective for machine translation, applying a source language model for the encoder and target language model for the decoder (Ramachandran et al., 2017).

4.1 Model Components

Word Vectors. Counseling dialogues consist of natural Korean text which is morphologically rich, so we train word vectors specifically developed for the Korean language (Park et al., 2018a). We train the vectors over a Korean corpus including out-of-domain general documents, 1) Korean Wikipedia, 2) online news articles, and 3) Sejong Corpus, as well as in-domain unlabeled counseling dialogues. The corpus contains 0.13 billion tokens. To validate the trained vector quality, we check performance on word similarity task (WS353) for Korean (Park et al., 2018a). The Spearmans correlation is 0.682, which is comparable to the state-of-the-art performance. These vectors are used as inputs (see Fig. 2, yellow).

Pre-trained Language Models. We assume that counselors and clients have different language models because they have distinctive roles in the dialogue. Therefore, we train a counselor language model and a client language model separately. We collect counselor utterances in the total dialogue dataset, except dialogues in the test set, to train a counselor language model, and all others are used for training a client language model. Then, We fine-tune the two trained LMs with utterances in the labeled dialogues.

For each model, we train word-level language models by using multilayer LSTMs. We apply various regularization techniques, weight tying (Inan et al., 2017), embedding dropout and variational dropout (Gal and Ghahramani, 2016), which are used to regularize LSTM language models (Inan et al., 2017; Ramachandran et al., 2017; Merity et al., 2018).

We use a 3-layer LSTM model having 300 hidden units for every layer. We set embedding dropout and output dropout for each layer to 0.2, 0.1, respectively. The pre-trained language models generate inputs to seq2seq layer of a conversation model (Fig. 2, blue and red).

Pre-trained Conversation Model. Next, we train a seq2seq conversation model (Vinyals and Le, 2015). We use the pre-trained counselor language model as an encoder, and the client language model as a decoder. The dependency of the decoder on the encoder is trained by seq2seq layers, stacked over the pre-trained models. (Fig. 2, green)

We stack 2-layer LSTMs over the pre-trained counselor and client language models, respectively. The final states of the LSTMs on the counselor language model is used as an initial state of the LSTMs on the client language model. We set the hidden unit size of 300 for every LSTMs and set output dropout to 0.05. The outputs of pre-trained conversation models is used for inputs to seq2seq layers of the task-specific layers.

During training the conversation model, we regularize the parameters of the model by adding cross entropy losses of the pre-trained counselor and client language models to seq2seq cross entropy loss of the conversation model. Three losses are weighted equally. This prevents catastrophic forgetting of the pre-trained language models and is important to achieve high performance (Ramachandran et al., 2017). Also, there is room for improvement of the architecture of a conversation model, which integrates pre-trained language models, to capture dialogue patterns better and thus leads to higher classification performance. We will explore the architecture in future work.

Task-specific layers. By leveraging pre-trained language model based conversation model, we finally add layers for classification. In order to capture task-specific features, we first stack seq2seq layers over the conversation model. Then we add attention mechanism for document classification (Yang et al., 2016). Lastly, we use a sigmoid function as an output layer to predict whether the information is included in utterances because multiple categories can appear in a single utterance. (Fig. 2, gray)

We use a 2-layer LSTM model for the seq2seq layers. We set 300 hidden units for every LSTMs and set output dropout to 0.05. The size of attention layer is set to 500.

4.2 Model Training

Thus the model is trained by three steps: 1) training word vectors and two language models, 2) training seq2seq conversation model with pre-trained LMs, and 3) fine-tuning task-specific classification layers, after removing softmax of the conversation model. In the last step, we compute binary logistic loss between predicted probability for each category and label as a loss function. We use Adam with default parameters for the optimizer, in order to train language models, conversation model, and fine-tuning the classifier. Also, gradual unfreezing is applied while training the model, starting updating parameters from the task-specific layers and unfreezing the next lower frozen layer for each epoch. We unfreeze layers every other epoch until all layers are tuned, and we stop training when the validation loss is minimized. All hyperparameters are tuned over the development set.

5 Experiments

5.1 Comparison Models

We compare our model with baseline models in Table 4. Models 1-5 are classifiers which only use the target client utterance to classify it, and models 6-8 are conversation model-based classifiers considering counselor's utterances and client's context utterances. All models use the same pre-trained word vectors for a fair comparison.

(1) Random Forest, (2) SVM with RBF kernel. We represent an utterance by an average of all word vectors in it, then feed it to the classifier input.

(3) CNN for text classification (Kim, 2014). In the convolution layer, we set the filter size to 1-10, and 30 filters each, then we apply max-over-time pooling. Then, a dense layer and sigmoid activation is applied over the layer.

(4) RNN Bidirectional LSTM is used where the final states for each direction are concatenated to represent the utterances. Then, a dense layer and a sigmoid activation are stacked.

(5) ULMFiT. A pre-trained client language model is used as a universal language model. Details are the same as described in Section 5.3. In addition, 2-LSTM layers and dense layer with sigmoid activations are stacked over the LM for classification. Gradual unfreezing is applied during training (Howard and Ruder, 2018).

(6) Seq2Seq. Like encoder-decoder based conversation model (Vinyals and Le, 2015), three LSTMs are assigned for each Counselor's utterances, client's context/target utterances. The initial states of the client language models are set to the final states of preceding utterances. Then dense & sigmoid layers for classification are stacked over the final state of the client's target utterances.

(7) HRED. Hierarchical encoder-decoder model (HRED) is used as a conversation model (Serban et al., 2016). For the encoder RNN, counselor's utterances and client's context utterances are given as inputs, and their information is stored in context RNN, which delivers it to the decoder accepting client's target utterances. Like (6) Seq2Seq, dense & sigmoid layers for classification are stacked over the final state of the client's target utterances.

5.2 Ablation Study

We conduct an ablation study to investigate the effect of the pre-trained models in Table 5.

(1-4) Adding pre-trained models. Model 1-4 have the same architecture as ConvMFiT, which is Model (8) in Table. 4. Model (1) in Table. 5 initializes every parameter randomly. Model (2) starts training only with pre-trained word vectors. Model (3) leverages counselor and client language models as well, and Model (4) shows the performance of ConvMFiT. As Model (3) and (4) use more than two pre-trained components, gradual unfreezing in applied, unfreezing shallower layers first during training.

(4-1) Task-specific Seq2Seq Layers. Model (4-1) removes task-specific Seq2Seq layers in the model, which leaves only attention and dense layer to capture task-specific features. It may result in an insufficient model capacity to capture relevant features for the task.

(4-2) Effect of Gradual Unfreezing. Model (4-2) is trained without gradual unfreezing, allowing the parameters of every layer in the model change by the gradients from the first epoch.

6 Results

6.1 Classification Performance

We show the performance of our model and comparison models in Table. 4. (1) Random Forests and (2) Support Vector Machines underperform to classify utterances correctly which belong to rarely occurred classes. Compared to (1) and (2), (3) CNN and (4) RNN show better performance since they look at the sequence of words (.416, .431, respectively). RNN shows slightly better performance than CNN. (6) ULMFiT outperforms the others by using a pre-trained client language models (.455).

The client target utterances have their context, and they also depend on the counselor's preceding utterances, so using the preceding counselor utterance as well as the client context utterances helps to improve the classification performance. When integrating the information using simple (6) seq2seq model, it shows better performance (.530) than (5) ULMFiT. (7) HRED adds a higher-level RNN to seq2seq models, but we find there is little performance gain (.001) which makes the model overfit easily.

(8) ConvMFiT employs pre-trained conversation model based on pre-trained LMs and so outperforms all other baseline models (.642). This is because ConvMFiT integrates conversational contexts and the counselor language model, which helps to capture the patterns of client's language better. Also, the improvement is higher for the class (Fact.) and (Chan.) which have small numbers of examples since the ConvMFiT could leverage pre-trained knowledge to classify them.

6.2 Effect of Pre-trained Components

With an ablation study applying pre-trained components step by step to our model, we show the

No.	Dep.	Model	F1 (Fact.)	F1 (Anec.)	F1 (Prob.)	F1 (Chan.)	F1 (Proc.)	Macro Prec	Macro Rec	Macro F1
1	X	RF	.000	.564	.420	.000	.723	.476	.269	.341
2	X	SVM(rbf)	.012	.683	.457	.000	.766	.602	.385	.384
3	X	CNN	.211	.528	.506	.128	.706	.450	.397	.416
4	X	RNN	.193	.574	.570	.046	.770	.607	.375	.431
5	X	ULMFiT	.205	.641	.591	.057	.784	.613	.413	.455
6	O	Seq2Seq	.263	.662	.678	.226	.823	.695	.472	.530
7	O	HRED	.261	.706	.675	.193	.820	.680	.475	.531
8	O	ConvMFiT	**.441**	**.761**	**.726**	**.447**	**.835**	**.716**	**.602**	**.642**

Table 4: Classification results. Among the models 1-6 which only use client's utterances to predict categories, (6) ULMFiT show better performance to the others. Models 7-9 are classifiers based on the conversation model, and ConvMFiT outperforms the others (.642).

No.	Emb.	LM	Conv. seq2seq	Grad. Unf.	Task. seq2seq	F1 (Fact.)	F1 (Anec.)	F1 (Prob.)	F1 (Chan.)	F1 (Proc.)	Macro Prec	Macro Rec	Macro F1
1	X	X	X	-	O	.032	.706	.602	.222	.711	.418	.575	.455
2	O	X	X	-	O	.043	.758	.661	.258	.748	.459	.662	.494
3	O	O	X	O	O	.425	.782	.727	.365	.831	.587	.719	.626
4	O	O	O	O	O	**.441**	**.761**	**.726**	**.447**	**.835**	**.602**	**.716**	**.642**
4-1	O	O	O	O	X	.307	.738	.666	.304	.802	.513	.687	.563
4-2	O	O	O	X	O	.417	.768	.721	.399	.824	.591	.695	.626
4	O	O	O	O	O	**.441**	**.761**	**.726**	**.447**	**.835**	**.602**	**.716**	**.642**

Table 5: Ablation study results. All models use the same architecture of ConvMFiT. Adding pre-trained word vectors (2), counselor and client language models (3), seq2seq layers of conversation model (4) results in a performance improvement. Also, adding task-specific seq2seq layers to ensure the model's sufficient capacity for capturing relevant features (4-1), and applying gradual unfreezing lead to performance improvement (4-2).

sources of performance improvement in Table 5. Model (1) uses the same architecture but no pre-trained models are applied, showing poor performance due to overfitting (.455). The f1 score is lower than that of the simple seq2seq model. Model (2) only uses pre-trained word vectors, and it helps to increase the performance slightly (.494). Also, Model (3) adds two pre-trained LMs with gradual unfreezing, resulting in a substantial performance increase (.626.), so we find pre-trained LMs are essential to the improvement. Lastly, Model (4) adds the dependency between two language models to fully leverage the pre-trained conversation model. This also helps classify utterances better (.642).

In addition, we find only adding attention and dense layers are not sufficient to learn task-specific features, so providing the model more capacity helps improving the performance. Without task-specific seq2seq layers, model (4-1) shows decreased f1 score (.563). Meanwhile, model (4-2) shows careful training scheme affects to the performance as well.

6.3 Qualitative Results

To investigate how linguistic patterns of counselors and clients differ in the various categories, we report qualitative results based on the activation values of the attention layer. To protect the anonymity of the clients, we explore key phrases from utterances rather than publish parts of the conversation in any form.

To this end, we compute the relative importance of n-grams. For any n of n-gram in an utterance of length N where $n < N$, the relative importance r is computed as follows:

$$\left(\prod_{i=1}^{n} a_i - (1/N)^n \right) / (1/N)^n \qquad (1)$$

where a_i is corresponding attention value of a word, $\prod_{i=1}^{n} a_i$ is the product of the values for every words in the n-gram, normalized by the expected attention weights $(1/N)^n$. The normalization term considers the length of utterance since a word in short utterances tends to have high attention value because of $\sum_{i=1}^{N} a_i = 1$.

We name this measure 'relative importance' r meaning that the degree of the n-gram is how much it is attended to compared to the expectation. Based on this, we select examples from 100 top-ranked key phrases for each category and the results are presented in Appendix (Table. 6). All of the presented key phrases are translated to English from Korean.

Factual Information. Clients provide demographic information and previous experience of visits to counselors or psychiatrists. In some case, clients talk more about the motivation of counseling. Since this information is explored in an early stage of counseling sessions, we also find counselor greetings with client's names.

Anecdotal Experience. Clients describe their experiences by using past tense verbs. Usually, utterances include phrases such as 'I thought that', 'I was totally wrong'. Counselors show simple responses 'well..', and reflections.

Appealing Problem. Like anecdotal experience, clients describe their problems, but with using the present tense of verbs. They are appealing their thinking and emotions. Counselors also show simple responses or reflections. Since some clients immediately start pouring out their problems right after counseling sessions starts, so greetings from counselor appear in key phrases.

Psychological Change. Clients obviously report their change of feelings, emotions or thoughts. It includes looking back on the past and then determining to change in the future. Counselors give supportive responses and empathetic understanding.

Counseling Process. Clients and counselors exchange greetings with each other. Also, they discuss making an appointment for the next session. In some cases, counselors respond to client's questions about the logistics of the sessions.

7 Related Work

Researchers have explored psycho-linguistic patterns of people with mental health problems (Gkotsis et al., 2016), depression (Resnik et al., 2015), Asperger's and autism (Ji et al., 2014) and Alzheimer's disease (Orimaye et al., 2014). In addition, these linguistic patterns can be quantified, for example, overall mental health (Loveys et al., 2017; Coppersmith et al., 2014), and schizophrenia (Mitchell et al., 2015).

To aid people with those mental issues, large portion of studies are dedicated to detecting those issues from natural language. Depression (Morales et al., 2017; Jamil et al., 2017; Fraser et al., 2016), anxiety (Shen and Rudzicz, 2017), distress (Desmet et al., 2016), and self-harm risk (Yates et al., 2017) can be effectively detected from narratives or social media postings.

8 Discussion and Conclusion

In this paper, we developed five categories of client utterances and built a labeled corpus of counseling dialogue. Then we developed the ConvMFiT for classifying the client utterances into the five categories, leveraging a pre-trained conversation model. Our model outperformed comparison models, and this is because of transferring knowledge from the pre-trained models. We also explored and showed typical linguistic patterns of counselors and clients for each category.

Our ConvMFiT model will be useful in other classification tasks based on dialogues. ConvMFiT is a seq2seq model for counselor-client conversation, however, another approach would be to model with existing non-goal oriented conversation models incorporating Variational Autoencoder (VAE) (Serban et al., 2017; Park et al., 2018b; Du et al., 2018). We plan to attempt these models in future work.

We expect to apply our trained model to various text-based psychotherapy applications, such as extracting and summarizing counseling dialogues or using the information to build a model addressing the privacy issue of training data. We hope our categorization scheme and our ConvMFiT model become a stepping stone for future computational psychotherapy research.

Ethical Approval

The study reported in this paper was approved by the KAIST Institutional Review Board (#IRB-17-95).

Acknowledgments

This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. *Transactions of the ACL*, 4:463–476.

Jennifer L Bearse, Mark R McMinn, Winston Seegobin, and Kurt Free. 2013. Barriers to psychologists seeking mental health care. *Professional Psychology: Research and Practice*, 44(3):150.

Glen Coppersmith, Mark Dredze, and Craig Harman. 2014. Quantifying mental health signals in twitter. In *Proceedings of the Workshop on CLPsych*. ACL.

Bart Desmet, Gilles Jacobs, and Véronique Hoste. 2016. Mental distress detection and triage in forum posts: The lt3 clpsych 2016 shared task system. In *Proceedings of the Third Workshop on CLPsych*. ACL.

Karthik Dinakar, Allison JB Chaney, Henry Lieberman, and David M Blei. 2014. Real-time topic models for crisis counseling. In *Twentieth ACM Conference on Knowledge Discovery and Data Mining, Data Science for the Social Good Workshop*.

Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. 2018. Variational autoregressive decoder for neural response generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3154–3163.

Clara E. Hill, Jean A. Carter, and Mary K. O'Farrell. 1983. Reply to howard and lambert: Case study methodology. 30:26–30.

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR mental health*, 4(2).

Kathleen C. Fraser, Frank Rudzicz, and Graeme Hirst. 2016. Detecting late-life depression in alzheimer's disease through analysis of speech and language. In *Proceedings of the Third Workshop on CLPsych*. ACL.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in NIPS 29*.

George Gkotsis, Anika Oellrich, Tim Hubbard, Richard Dobson, Maria Liakata, Sumithra Velupillai, and Rina Dutta. 2016. The language of mental health problems in social media. In *Proceedings of the Third Workshop on CLPsych*. ACL.

CE Hill, C Greenwald, KG Reed, D Charles, MK O'Farrell, and JA Carter. 1981. Manual for counselor and client verbal response category systems. *Columbus, OH: Marathon*.

Clara E Hill. 1978. Development of a counselor verbal response category. *Journal of Counseling Psychology*, 25(5):461.

Clara E Hill, Barbara J Thompson, and Elizabeth Nutt Williams. 1997. A guide to conducting consensual qualitative research. *The counseling psychologist*, 25(4):517–572.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Christine Howes, Matthew Purver, and Rose McCabe. 2014. Linguistic indicators of severity and progress in online text-based therapy for depression. In *Proceedings of the Workshop on CLPsych*. ACL.

Thomas D. Hull. 2015. A preliminary study of talkspaces text-based psychotherapy.

Zac E Imel, Mark Steyvers, and David C Atkins. 2015. Computational psychotherapy research: Scaling up the evaluation of patient–provider interactions. *Psychotherapy*, 52(1):19.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*.

Zunaira Jamil, Diana Inkpen, Prasadith Buddhitha, and Kenton White. 2017. Monitoring tweets for depression to detect at-risk users. In *Proceedings of the Fourth Workshop on CLPsych*, pages 32–40. ACL.

Yangfeng Ji, Hwajung Hong, Rosa Arriaga, Agata Rozga, Gregory Abowd, and Jacob Eisenstein. 2014. Mining themes and interests in the asperger's and autism community. In *Proceedings of the Workshop on CLPsych*. ACL.

Seung-Hee Jee Hea-Young Oh Kyung-Min Kim. 2010. The response analysis of single session chat counseling - the cases of adolescents with the suicidal crisis -. *The Korea Journal of Youth Counseling*, 18(2):167–186.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*.

Kate Loveys, Patrick Crutchley, Emily Wyatt, and Glen Coppersmith. 2017. Small but mighty: Affective micropatterns for quantifying mental health from social media language. In *Proceedings of the Fourth Workshop on CLPsych*. ACL.

Kien Hoa Ly, Ann-Marie Ly, and Gerhard Andersson. 2017. A fully automated conversational agent for promoting mental well-being: A pilot rct using mixed methods. *Internet Interventions*, 10:39–46.

Akio Mantani, Tadashi Kato, Toshi A Furukawa, Masaru Horikoshi, Hissei Imai, Takahiro Hiroe, Bun Chino, Tadashi Funayama, Naohiro Yonemoto, Qi Zhou, et al. 2017. Smartphone cognitive behavioral therapy as an adjunct to pharmacotherapy for refractory depression: randomized controlled trial. *Journal of medical Internet research*, 19(11).

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *ICLR*.

Margaret Mitchell, Kristy Hollingshead, and Glen Coppersmith. 2015. Quantifying the language of schizophrenia in social media. In *Proceedings of the 2nd Workshop on CLPsych*. ACL.

Michelle Morales, Stefan Scherer, and Rivka Levitan. 2017. A cross-modal review of indicators for depression detection systems. In *Proceedings of the Fourth Workshop on CLPsych*. ACL.

Sylvester Olubolu Orimaye, Jojo Sze-Meng Wong, and Karen Jennifer Golden. 2014. Learning predictive linguistic features for alzheimers disease and related dementias using verbal utterances. In *Proceedings of the Workshop on CLPsych*. ACL.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018a. Subword-level word vector representations for korean. In *Proceedings of the 56th Annual Meeting of the ACL*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018b. A hierarchical latent structure for variational conversation modeling.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on EMNLP*.

Philip Resnik, William Armstrong, Leonardo Claudino, Thang Nguyen, Viet-An Nguyen, and Jordan Boyd-Graber. 2015. Beyond lda: Exploring supervised topic modeling for depression-related language in twitter. In *Proceedings of the 2nd Workshop on CLPsych*. ACL.

Julius Seeman. 1949. A study of the process of nondirective therapy. *Journal of Consulting Psychology*, 13(3):157.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Judy Hanwen Shen and Frank Rudzicz. 2017. Detecting anxiety through reddit. In *Proceedings of the Fourth Workshop on CLPsych*, pages 58–65. ACL.

Juan C Torrado, Javier Gomez, and Germán Montoro. 2017. Emotional self-regulation of individuals with autism spectrum disorders: smartwatches for monitoring and interaction. *Sensors*, 17(6):1359.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the NAACL*.

Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums. In *Proceedings of the 2017 Conference on EMNLP*. ACL.

A Appendices

Factual Information	
Counselor	nice to meet you of course first I see. now . . [clients name]s well . .
Client	never visited neuropsychiatry female , worker never received psychological counseling motivation of counseling age # , female

Anecdotal Experience	
Counselor	might be left behind be in a peer relationships I see. actually well . . I understand what you mean
Client	too hard didnt get along seem to be I thought that I was totally wrong

Appealing Problem	
Counselor	right . all but now the relationship is nice to meet you you told me well more comfortable?
Client	sick and sad have many thoughts keep thinking I think I did it no future and frustrating

Psychological Change	
Counselor	right . and if you do that someday suddenly I did I hope so question arises
Client	did not think so but I was so shocked it could have been longer but looking back I think I will

Counseling Process	
Counselor	hard to express but counseling proceeds Hello ? [clients name] , lets talk at # yes the counseling time is
Client	how are you ? what do you think ? thank you for listening hello counselor time is not enough

Table 6: Representative phrases in utterances of both conversational parties, extracted based on the relative importance of the phrase in terms of classification by using attention weights.

Using Similarity Measures to Select Pretraining Data for NER

Xiang Dai[1,2] **Sarvnaz Karimi**[1] **Ben Hachey**[2,3] **Cecile Paris**[1]
[1]CSIRO Data61, Sydney, Australia
[2]University of Sydney, Sydney, Australia
[3]Digital Health CRC, Sydney, Australia
{dai.dai,sarvnaz.karimi,cecile.paris}@csiro.au
ben.hachey@gmail.com

Abstract

Word vectors and Language Models (LMs) pretrained on a large amount of unlabelled data can dramatically improve various Natural Language Processing (NLP) tasks. However, the measure and impact of similarity between pretraining data and target task data are left to intuition. We propose three cost-effective measures to quantify different aspects of similarity between source pretraining and target task data. We demonstrate that these measures are good predictors of the usefulness of pretrained models for Named Entity Recognition (NER) over 30 data pairs. Results also suggest that pretrained LMs are more effective and more predictable than pretrained word vectors, but pretrained word vectors are better when pretraining data is dissimilar.

1 Introduction

Modern neural architectures for NLP are highly effective when provided a large amount of labelled training data (Zhang et al., 2015; Conneau et al., 2017; Bowman et al., 2015). However, a large labelled data set is not always readily accessible due to the high cost of expertise needed for labelling or even due to legal barriers. Researchers working on such tasks usually spend a considerable amount of effort and resources on collecting useful external data *sources* and investigating how to transfer knowledge to their *target* tasks (Qi et al., 2009; Kim et al., 2017). Recent transfer learning techniques make the most of limited labelled data by incorporating word vectors or LMs pretrained on a large amount of unlabelled data. This produces dramatic improvements over a range of NLP tasks where appropriate unlabelled data is available (Peters et al., 2017, 2018; Akbik et al., 2018; Devlin et al., 2019).

However, there is still a lack of systematic study on how to select appropriate data to pretrain word vectors or LMs. We observe a range of heuristic strategies in the literature: (1) collecting a large amount of generic data, e.g., web crawl (Pennington et al., 2014; Mikolov et al., 2018); (2) selecting data from a similar *field* (the subject matter of the content being discussed), e.g., biology (Chiu et al., 2016; Karimi et al., 2017); and, (3) selecting data from a similar *tenor* (the participants in the discourse, their relationships to each other, and their purposes), e.g., Twitter, or online forums (Li et al., 2017; Chronopoulou et al., 2019). In all these settings, the decision is based on heuristics and varies according to the individual's experience. We also conducted a pilot study that suggests that the practitioner's intuition is to prioritise field over tenor (see Section 3).

Our overarching goal is to develop a cost-effective approach that, given a NER data set, nominates the most suitable source data to pretrain word vectors or LMs from several options. Our approach builds on the hypothesis that the more similar the source data is to the target data, the better the pretrained models are, all other aspects (such as source data size) being equal. We propose using target vocabulary covered rate and language model perplexity to select pretraining data. We also introduce a new measure based on the change from word vectors pretrained on source data to word vectors initialized from source data and then trained on target data. Experiments leverage 30 data pairs from five source and six target NER data sets, each selected to provide a range of fields (i.e., biology, computer science, medications, local business) and tenors (i.e., encyclopedia articles, journal articles, experimental protocols, online reviews).

Our contributions can be summarized as below:

- We propose methods to quantitatively measure different aspects of similarity between

Proceedings of NAACL-HLT 2019, pages 1460–1470
Minneapolis, Minnesota, June 2 - June 7, 2019.

source and target data sets and find that these measures are predictive of the impact of pretraining data on final accuracy. To the best of our knowledge, this is the first systematic study to investigate LMs pretrained on various data sources.[1]

- We find that it is important to consider tenor as well as field when selecting pretraining data, contrary to human intuitions.
- We show that models pretrained on a modest amount of similar data outperform pretrained models that take weeks to train over very large generic data.

2 Related Work

Text Similarity Word similarity following the hypothesis that similar words tend to occur in similar contexts (Harris, 1954) is well studied and forms the foundation of neural word embedding architectures. Hill et al. (2015) and Budanitsky and Hirst (2006) evaluate functional similarity (as in *school* versus *college*) and associative similarity (as in *school* versus *teacher*) captured by semantic models, respectively. Pavlick et al. (2015) study sentence-level similarity, using entailment relation, vector embedding and stylistic variation measures. Kusner et al. (2015) propose Word Mover's Distance to measure the similarity between documents and evaluate on document classification tasks. We extend the study of similarity to corpus-level, and focus on its implication on unsupervised pretraining.

Pretrained Word Vectors The effectiveness of pretrained word vectors mainly depends on three factors: source data, training algorithm, and its hyper-parameters. Turian et al. (2010) and Levy et al. (2015) systematically compare count-based distributional models and distributed neural embedding models. They find that both models can improve the performance of downstream tasks. Chiu et al. (2016) identify the most influential hyper-parameters of neural embedding methods. They also investigate the impact of the source data size and find that larger pretraining data do not necessarily produce better word vectors for biomedical NER. Our work regarding pretrained word vectors is conducted using skip-gram model with default hyper-parameter setting (Mikolov et al., 2013), and our focus is on the impact of similarity between source data and target task data on the effectiveness of pretrained word vectors for NER tasks. Our observations are a useful supplement to the literature as a practitioners' guide.

Pretrained Language Models Dai and Le (2015) investigate different methods to transfer knowledge to supervised recurrent neural networks. They establish that a pretrained recurrent LM can improve the generalization ability of the supervised models. They use unlabelled data from Amazon reviews to pretrain the LM and find that it can improve classification accuracy on the Rotten Tomatoes data set. Joshi et al. (2018) empirically showed that, for their vaccination behaviour detection task on twitter data, LMs pretrained on a small amount of movie reviews outperform the ones pretrained on large size of Wikipedia data. Peters et al. (2017) successfully inject the information captured by a bidirectional LM into a sequence tagger, and extend this approach to other NLP tasks (Peters et al., 2018). Our work is based on (Peters et al., 2018) and investigates the impact of pretraining data on the effectiveness of pretrained LMs for downstream NER tasks.

Transfer Learning While our study falls into the paradigm of semi-supervised learning, we distinguish ourselves from other studies in transfer learning. One sub-area of transfer learning is domain adaptation, which aims to learn transferable representation from a source domain and apply it to a target domain (Blitzer et al., 2006; Yang and Eisenstein, 2015). The question in domain adaptation is usually framed as 'Given a source and a target, how to transfer?'. In contrast, the question we address is 'Given a specific target, which source to choose from?'. The other sub-area of transfer learning is transferring from multiple sources (Yin and Schütze, 2015; Li et al., 2018). Our work focuses, instead, on the selection of a single external data source. Our work is inspired by the methodology proposed by Johnson et al. (2018) where they predict a system's accuracy using larger training data from its performance on much smaller pilot data. However, we aim to predict the usefulness of pretrained models for target tasks from the similarity between the source pretraining data and the target task data.

Named Entity Recognition Our work builds on the literature on deep neural networks applied to sequence tagging tasks. Architectures based on

---

[1] Our pretrained word vectors and LMs are publicly available: https://bit.ly/2O0mOOG.

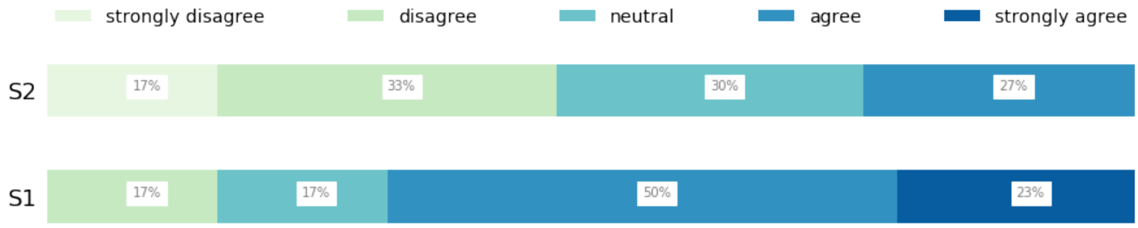


Figure 1: Likert scale ratings from NLP and ML practitioners ($N = 30$) for the statement 'Unsupervised pretraining on S would be useful for supervised named entity recognition learning on T.' Target data T is described as 'Online forum posts about medications,' source data S1 as 'Research papers about biology and health,' and source data S2 as 'Online reviews about restaurants, hotels, barbers, mechanics, etc.'

different combinations of convolutional and recurrent neural networks have achieved state-of-the-art results on many NER tasks. A detailed review and comparison of these methods can be found in (Yang et al., 2018). Our experiments on the usefulness of pretrained word vectors and pretrained LMs for NER tasks are based on one variant proposed by Lample et al. (2016).

3 What Human Intuition Indicates

Results of a survey capturing intuition regarding selection of pretraining data across 30 NLP or machine learning practitioners is shown in Figure 1. Participants were provided short descriptions of the target data set T, and two possible source data sets S1 and S2 as

- T: Online forum posts about medications;
- S1: Research papers about biology and health;
- S2: Online reviews about restaurants, hotels, barbers, mechanics, etc.

We constructed each of these descriptions as 't about f' where t is intended to describe the tenor and f the field. Each participant rated both sources on a five-point Likert, indicating agreement with the statement "Unsupervised pretraining on S would be useful for supervised named entity recognition learning on T". 73% of participants agreed or strongly agreed that S1 would be useful, while only 27% agreed that S2 would be useful. A Wilcoxon signed-rank test indicates that scores are significantly higher for S1 than for S2 ($Z = 43.0, p < 0.001$).

Although small in scale, these results show that intuition varies across practitioners, motivating our work on identifying quantitative measures that are predictive of performance. These results also suggest that practitioners favour field over tenor when selecting pretraining data, which would be detrimental to accuracy of the target NER tasks in later experiments (Section 7.2).

4 Similarity Measures

To measure the similarity between source and target data, we start from identifying linguistic concepts behind these human intuitions. Then, we propose several measures to quantify these attributes which lead to the perception that two data sets are similar.

Researchers who select pretraining data from a similar field believe that, if the source data has a similar field to the target data, they tend to share similar vocabulary. Conversely, vocabularies are different from each other if source and target are from different fields. Imagine data sets about medications and restaurants. Those who select pretraining data from a similar tenor believe that tenor may impact the writing style of text. Imagine the participants in online reviews and scientific papers, their relationships to each other, their purposes and how these affect text style, including punctuation, lexical normalization, politeness, emotiveness and so on (Lee, 2001; Solano-Flores, 2006; Pavlick and Tetreault, 2016).

Below, we detail different measures based on these intuitions to quantify different aspects of similarity between two data sets.

4.1 Target Vocabulary Covered

The first measure is simply the percentage of the target vocabulary that is also present in the source data. An extremely dissimilar example is that of different languages. They have a totally different vocabulary and are considered dissimilar, even if they are written in a similar style and talking about the same subject [2]. We propose *Target Vocabulary Covered (TVC)* as a measure of field, calculated as

$$TVC(D_S, D_T) = \frac{|V_{D_S} \cap V_{D_T}|}{|V_{D_T}|},$$

where V_{D_S} and V_{D_T} are sets of unique tokens in source and target data sets respectively. We also investigate a variant where only content words (nouns, verbs, adjectives) are used to calculate V_{D_S} and V_{D_T}. We denote this variant as *VCcR*.

4.2 Language Model Perplexity

A language model can assign a probability to any sequence of words $< w_1, \cdots, w_N >$ using chain rule of probability:

$$p(w_1, w_2, \cdots, w_N) = \prod_{i=1}^{N} p(w_i | w_1^{w-1}),$$

where N is the length of the sequence and w_1^{i-1} are all words before word w_i. In practice, this equation can be simplified by n-gram models based on Markov Assumption:

$$p(w_1, w_2, \cdots, w_N) = \prod_{i=1}^{N} p(w_i | w_{i-n+1}^{i-1}),$$

where w_{i-n+1}^{i-1} represents only n preceding words of w_i. To make the model generalize better, smoothing techniques can be used to assign non-zero probabilities to unseen events. In this study, we use Kneser-Ney smoothed 5-gram models (Heafield, 2011). To measure the similarity between two data sets using language modeling, we first train the language model on the source data, then evaluate it on the target data using perplexity to represent the degree of similarity. The intuition is that, if the model finds a sentence very unlikely (dissimilar from the data where this language model is trained on), it will assign a low probability and therefore high perplexity. The summed up *perplexity (PPL)* is then:

$$PPL(D_S, D_T) = \sum_{i=1}^{m} P(D_T^i)^{-\frac{1}{N_i}},$$

where m is the number of sentences in the target data set, and $P(D_T^i)$ is the probability assigned by the language model trained on the source data to the i-th sentence from the target data set, whose sentence length is N_i.

PPL is token-based, similar to TVC, but also captures surface structure. We therefore propose PPL as a proxy to measure tenor as well as field.

4.3 Word Vector Variance

Pretrained word vectors capture semantic and syntactic regularities of words (Artetxe et al., 2018). The variance of a word vector that is first trained on the source data and then on the target data can reflect the difference of linguistic regularities between the two data sets.

Intuitively, if the context words around a given word are very different in the source and target data, then the word vector of this word learned from the source will be updated more than those words whose context words are similar between source and target. Therefore, we use *Word Vector Variance (WVV)* as another combined measure of tenor and field.

To calculate word vector variance, we first train word vectors on the source data set using skip-gram model (Mikolov et al., 2013). The trained word vectors are denoted as $WS \in \mathbb{R}^{|V_S| \times d}$, where $|V_S|$ is the vocabulary size of the source data set and d is the vector dimension. Then, we use WS as initial weights of a new skip-gram model, and train this new model on the target data. We denote the final word vectors as WT. The WVV can be calculated as:

$$WVV(D_S, D_T) = \frac{1}{|V_S|} \frac{1}{d} \sum_{i}^{|V_S|} \sum_{j}^{d} (WS_i^j - WT_i^j)^2.$$

The smaller the word vector variance, the more similar context surrounds the same words from the two data sets, and therefore the more similar the two data sets are.

5 Data Sets

Source data sets We use five data sets as source data, covering a range of fields (i.e., clinical, biomedical, local business and Wiki with diverse fields) and tenors (i.e., popular reporting, notes, scholarly publications, online reviews and encyclopedia). To isolate the impact of source size, we sample all source data to approximately 100 million tokens. We also analyze the impact of source data size separately in Section 7.3. The specifications of these source data sets are given in Table 1.

---

[2] Our focus is on transferring through pretrained models using one single source and we do not consider multilingual similarity.

Data set	Description
1BWB	The original one billion word language model benchmark data (Chelba et al., 2013), produced from News Crawl data. It has been randomly shuffled and we use the last 13 out of 100 files.
MIMIC	A clinical database comprising over 58,000 hospital admissions for intensive care unit (ICU) patients (Johnson et al., 2016). We use the first 50,000 notes associated with hospital stays.
PubMed	Around 30 million citations for biomedical literature covering the fields of biomedical and health. We use articles published after October 2017 and utilize their titles and abstracts only.
Wiki	WikiText-103, released by Merity et al. (2016) and consisting of around 28K *Good* or *Featured* articles from Wikipedia. These articles are reviewed by human editors, and they are selected based on the writing quality. We refer to this data set as Wiki.
Yelp	An online forum where customers can write reviews about local businesses. We use data released in round 12 of the Yelp Data set Challenge and select the first 2 out of 6 million reviews.

Table 1: List of the source data sets.

Target data sets Six NER data sets are used as target data: CADEC (Karimi et al., 2015), CoNLL2003 (Sang and Meulder, 2003), CRAFT (Bada et al., 2012), JNLPBA (Collier and Kim, 2004), ScienceIE (Augenstein et al., 2017) and WetLab (Kulkarni et al., 2018). Details of these target data are listed in Table 2. We choose these data sets based on two considerations:

1. NER is a popular structured NLP task. Using NER, we want to observe how the similarity between source and target data may affect the effectiveness of different pretrained word vectors and LMs on downstream tasks.

2. NER is highly sensitive to word representations, because the model needs to make token level decisions. That is, each token needs to be assigned a proper label. Past studies have shown that removing pretrained word vectors from a tagging system results in a large drop in performance (Huang et al., 2015; Lample et al., 2016).

6 Experimental Setup

To investigate the impact of source data on pretrained word vectors and LMs, we pretrain word vectors and LMs on different sources separately, then observe how the effectiveness of these pretrained models varies in different NER data sets.

We use the BiLSTM-CRF model, a state-of-the-art model for sequence tagging tasks, as a supervised model for the target NER task. We follow the architecture proposed in (Lample et al., 2016), except that we use two BiLSTM-layers and employ a CNN network to learn character-level representations (Ma and Hovy, 2016). Micro average F_1 score is used to evaluate the performance of the tagger (Sang and Meulder, 2003).

Word vectors are pretrained using word2vec with its default hyper-parameter setting (Mikolov et al., 2013). In different experiments, we only replace the word embedding weights initialized by word vectors pretrained on different source data, then make these weights trained jointly with other model parameters. The baseline is denoted as *None* in Table 3, where word embedding weights are randomly initialized.

LMs are pretrained using the architecture proposed by Jozefowicz et al. (2016) with hyper-parameters in (Peters et al., 2018). The supervised model used for NER is the same BiLSTM-CRF model mentioned above, and we follow the approach proposed by Peters et al. (2018) to incorporate the pretrained LMs. Note that these pretrained LMs are character-based. Therefore, words in the target data set are first converted into a sequence of characters, and then fed into the LMs. The contextualized representation of each word is generated using the outputs of all layers of the pretrained LMs, then injected to the input of the second BiLSTM layer of the supervised model.

7 Experimental Results

Using proposed similarity measures, we first quantify the similarity between all source-target pairs (Section 7.1), then investigate how these measures can be used to predict the usefulness of pretraining data (Section 7.2). Finally, we take the source data size into consideration, and observe its impact on the effectiveness of pretrained model on both similar and dissimilar source-target settings (Section 7.3).

Data set	Size	Entity Types	Description
CADEC	120,341	Adverse Drug Event, Disease, Drug, Finding, Symptom	Posts taken from AskaPatient, which is a forum where consumers can discuss their experiences with medications.
CoNLL2003	301,418	Person, Organization, Location, Miscellany	Newswire from the Reuters RCV1 corpus.
CRAFT	561,015	Cell, Chemical entity, Biological taxa, Protein, Biomacromolecular sequence, Entrez gene, Biological process and molecular function, Cellular component	Full-length, open-access journal articles about biology.
JNLPBA	593,590	Protein, DNA, RNA, Cell line and Cell type	Abstract of journal articles about biology.
ScienceIE	99,555	Process (including methods, equipment), Task and Material (including corpora, physical materials)	Journal articles about Computer Science, Material Sciences and Physics.
Wetlab	220,618	Action, 9 object-based (Amount, Concentration, Device, Location, Method, Reagent, Speed, Temperature, Time) entity types, 5 measure-based (Numerical, Generic-Measure, Size, pH, Measure-Type) and 3 other (Mention, Modifier, Seal) types	Protocols written by researchers about conducting biology and chemistry experiments.

Table 2: List of the target NER data sets and their specifications. Size is shown in number of tokens.

7.1 Similarity Between Source and Target Data Sets

Different aspects of similarity measured between five source and six target data sets are shown in the left side of Table 3. The language model trained on PubMed achieves lower perplexity when evaluated on CRAFT, JNLPBA and ScienceIE compared to other sources. On one hand, it is expected that PubMed is similar to CRAFT and JNLPBA, since they are all sampled journal articles about biology and health, thus being similar in terms of both field and tenor. On the other hand, although ScienceIE does not have the same field as PubMed (computer science, material and physics versus biology and health), they are similar because they share a similar tenor (scholarly publications).

The measures calculated on CADEC also show that tenor is reflected more than field by PPL and WVV. Source data set Yelp is more similar to CADEC than PubMed and MIMIC from both PPL and WVV perspectives. CADEC is a data set focusing on recognizing drugs, diseases and adverse drug events. The field of CADEC is therefore more similar to PubMed which includes journal articles in health discipline and MIMIC which contains clinical notes. However, CADEC is written by patients, and can be considered as 'drug reviews'. The tenor is therefore closer to the one in Yelp, where customers use informal language to describe their experiences.

All sources are measured against WetLab with relatively high PPL and WVV values. This reflects the fact that the tenor of WetLab (experimental protocols) is different from the tenor of all sources, although WetLab has a similar field (biology) with PubMed which is therefore more similar than other sources. For CoNLL2003, 1BWB which is News Crawl data is the most similar source, while PubMed is the most dissimilar source from PPL perspective, and MIMIC is the most dissimilar one using WVV measure.

Although WVV does not distinguish between different sources as PPL does, it still reflects the same trend as PPL regarding which source is the most similar to a given target data set.

Can these different similarity measures reach a consensus? Similarity results in Table 3 indicate that using different measures can lead to almost the same answer regarding which source is the most similar one to a given target. To further investigate the level of agreement between different similarity measures, we employ inter-method agreement that we ask a fine-grained question on the results in Table 3: given a target and two sources, do similarity measures make the same conclusion as to which source is more similar? Using the five source and six target data sets, we generate a total of 60 binary comparisons. For example, given WetLab, is 1BWB a more similar source than Wiki? PPL shows that 1BWB is more simi-

		Similarity				**NER F_1 Score**			
						Pretrained word vectors		**Pretrained LMs**	
Target	**Source**	**PPL**	**WVV**	**TVC (%)**	**TVcC (%)**	**F_1 score**	**Δ**	**F_1 score**	**Δ**
CADEC	None	–	–	–	–	66.14 (± 0.53)	–	66.14 (± 0.53)	–
	1BWB	307.4	1.137	**81.73**	**82.94**	69.44 (± 0.52)	3.30	70.08 (± 0.43)	3.94
	MIMIC	1007.0	1.134	78.19	81.69	69.65 (± 0.43)	3.51	70.11 (± 0.48)	3.97
	PubMed	927.4	1.195	78.81	79.79	69.84 (± 0.55)	3.70	70.15 (± 0.50)	4.01
	Wiki	519.8	1.196	79.74	76.71	69.62 (± 0.15)	3.48	69.32 (± 0.65)	3.18
	Yelp	**291.1**	**1.104**	80.76	82.28	**70.27 (± 0.34)**	**4.13**	**70.46 (± 0.52)**	**4.32**
CoNLL2003	None	–	–	–	–	82.08 (± 0.38)	–	82.08 (± 0.38)	–
	1BWB	**480.6**	**1.020**	**75.64**	**87.35**	**86.36 (± 0.29)**	**4.28**	**89.78 (± 0.12)**	**7.70**
	MIMIC	2945.0	1.542	34.47	39.55	84.94 (± 0.35)	2.86	83.68 (± 0.30)	1.60
	PubMed	3143.1	1.356	53.29	68.41	85.56 (± 0.46)	3.48	84.15 (± 0.22)	2.07
	Wiki	650.4	1.159	66.21	80.87	86.32 (± 0.28)	4.24	89.11 (± 0.23)	7.03
	Yelp	2025.5	1.399	53.92	68.95	85.58 (± 0.26)	3.50	85.19 (± 0.38)	3.11
CRAFT	None	–	–	–	–	69.17 (± 0.64)	–	69.17 (± 0.64)	–
	1BWB	1328.1	2.073	59.07	62.98	73.97 (± 0.06)	4.80	71.23 (± 0.81)	2.06
	MIMIC	2427.5	2.390	48.73	50.03	73.01 (± 0.22)	3.84	71.90 (± 0.26)	2.73
	PubMed	**360.3**	**1.838**	**76.29**	**80.69**	**75.45 (± 0.28)**	**6.28**	**75.45 (± 0.09)**	**6.28**
	Wiki	974.7	2.075	63.66	63.12	74.07 (± 0.40)	4.90	69.75 (± 0.09)	0.58
	Yelp	2085.7	2.187	48.01	50.85	72.48 (± 0.13)	3.31	72.75 (± 0.26)	3.58
JNLPBA	None	–	–	–	–	70.45 (± 0.21)	–	70.45 (± 0.21)	–
	1BWB	1190.8	2.000	39.90	53.54	72.39 (± 0.23)	1.94	72.54 (± 0.34)	2.09
	MIMIC	2533.4	2.172	36.95	50.04	**73.24 (± 0.29)**	**2.79**	71.76 (± 0.13)	1.31
	PubMed	**205.9**	**1.597**	**58.87**	**80.17**	72.77 (± 0.65)	2.32	**74.29 (± 0.40)**	**3.84**
	Wiki	717.9	2.036	42.34	53.05	72.77 (± 0.27)	2.32	72.42 (± 0.23)	1.97
	Yelp	2134.4	2.155	30.78	41.41	72.53 (± 0.18)	2.08	72.51 (± 0.21)	2.06
ScienceIE	None	–	–	–	–	26.85 (± 0.17)	–	26.85 (± 0.17)	–
	1BWB	884.6	1.197	71.50	76.78	34.40 (± 0.50)	7.55	38.10 (± 0.31)	11.25
	MIMIC	2706.7	1.461	54.29	59.34	31.23 (± 0.15)	4.38	35.27 (± 0.43)	8.42
	PubMed	**345.6**	**1.037**	**83.25**	**87.01**	**37.91 (± 0.12)**	**11.06**	**42.07 (± 0.03)**	**15.22**
	Wiki	684.2	1.127	76.99	78.01	36.15 (± 0.11)	9.30	40.39 (± 0.05)	13.54
	Yelp	1562.2	1.347	62.32	66.42	33.92 (± 0.14)	7.07	36.05 (± 0.02)	9.20
WetLab	None	–	–	–	–	76.91 (± 0.10)	–	76.91 (± 0.10)	–
	1BWB	1526.0	2.167	59.67	61.47	78.66 (± 0.35)	1.75	78.94 (± 0.05)	2.26
	MIMIC	3046.1	2.393	53.83	55.31	78.68 (± 0.14)	1.13	78.65 (± 0.13)	1.74
	PubMed	**1104.7**	**2.078**	**71.39**	**74.46**	**78.93 (± 0.28)**	**2.02**	**79.62 (± 0.07)**	**2.71**
	Wiki	1617.8	2.158	61.02	60.31	78.45 (± 0.20)	1.54	79.05 (± 0.21)	2.14
	Yelp	1784.5	2.240	54.16	54.96	78.48 (± 0.15)	1.57	79.04 (± 0.19)	2.13

Table 3: Similarity between source and target data sets (left), and the effectiveness of word vectors and LMs pretrained using different sources for NER (right). Lower PPL or WVV values indicate higher similarity between source and target, while higher TVC and TVcC values indicate higher similarity. *None* rows refer to the models that word embedding weights are randomly initialized with no pretrained LMs. Δ shows absolute improvement. We repeat every NER experiment 5 times, and report mean and standard deviation of test F_1 scores.

lar, while WVV gives an opposite answer. Fleiss's kappa (Fleiss, 1971) (a variant of Cohen's kappa for more than two raters) is a robust metric used to measures inter-rater agreement, since it takes random chance into consideration. We use it to measure the inter-method agreement between the 60 binary comparisons inferred using PPL, WVV and TVC. Our results achieve a Fleiss's kappa of 0.733, which shows a high agreement between conclusions inferred using different measures.

Overall, we find that these similarity measures can reach high level of consensus. To simplify our following discussion, from here on *similar* means low PPL (because of its clear distinction between different sources), unless otherwise stated.

7.2 Impact of Pretraining Data

After we quantify the similarity between source and target data sets, the next question is how these similarity measures can be used to predict the effectiveness of pretrained models for NER tasks.

Results in Table 3 show that, although all pretrained word vectors and LMs can improve the performance of the target model, the improvement varies in different target data sets. In other words, no single source is suitable for all target NER data

	Word vectors	**LMs**
TVC	0.454	0.666
TVcC	0.469	0.739
PPL	-0.398	-0.618
WVV	-0.406	-0.747

Table 4: Correlation coefficients between similarity measures and the effectiveness of pretrained models. The coefficients vary between -1 (negative correlation) and 1 (positive correlation). Zero means no correlation.

sets. Word vectors and LMs pretrained on a source similar to the target outperform the ones pretrained on other sources (except pretrained word vectors for JNLPBA data set).

We also observe that pretrained LMs provide more benefits than pretrained word vectors if source data is similar to the target (see 1BWB-CoNLL2003 and PubMed-JNLPBA data pairs). However, if the source is dissimilar to the target, pretrained word vectors outperform pretrained LMs (see these pairs: MIMIC-CoNLL2003, PubMed-CoNLL2003, MIMIC-CRAFT).

Predictiveness of similarity measures To analyze how proposed similarity measures correlate to the effectiveness of pretrained word vectors and LMs for the target NER tasks, we employ the Pearson correlation analysis to find out the relationships between improvement due to pretrained models and TVC, TVcC, PPL and WVV. The results in Table 4 show that our proposed similarity measures are predictive of the effectiveness of the pretraining data. In terms of pretrained word vectors, VCcR is the most informative factor in predicting the effectiveness of pretrained word vectors given a target data set. It implies that finding a source data set which has large vocabulary intersection with the target data set is a promising first step to generate effective pretrained word vectors. The results regarding the LM performance show that it has a stronger correlation with similarity measures than the one of word vectors, thus more predictable using our proposed measures.

Comparison to publicly available pretrained models Recent literature shows substantial improvements are sometimes possible when pretraining on very large generic corpora. Given that pretrained models are freely available, is it even necessary to pretrain on similar data as proposed above? We compare to publicly available (1) word vectors trained on 6 billion tokens of encyclopaedia articles and news stories about various fields [3] and (2) LMs trained on 5.5 billion tokens of encyclopaedia articles and news stories about various fields [4]. We use the same experimental setup described in Section 6, that pretrained word vectors are used to initialize the weights of word embedding layer, whereas outputs of pretrained LMs are used as input features of the supervised model. We find that word vectors and LMs pretrained on small similar sources can achieve competitive or even better performance than the ones pretrained on larger sources (Table 5). On JNLPBA, ScienceIE and Wetlab, LMs pretrained on the small similar source perform better, while word vectors pretrained on the small similar source perform better on CRAFT, JNLPBA, and ScienceIE.

	Word vectors		**LMs**	
	GloVe	Ours	ELMo	Ours
CADEC	**70.30**	70.27	**71.91**	70.46
CoNLL2003	**90.25**	86.36	**91.34**	89.78
CRAFT	74.22	**75.45**	**75.77**	75.45
JNLPBA	73.19	**73.24**	73.65	**74.29**
ScienceIE	37.10	**37.91**	41.15	**42.07**
WetLab	**79.15**	78.93	79.57	**79.62**

Table 5: Comparison between our best performance pretrained models and the publicly available ones, which are pretrained on much larger corpora.

	ScienceIE		**WetLab**	
	Def	**Opt**	**Def**	**Opt**
1BWB	34.40	34.57	78.66	79.12
MIMIC	31.23	34.14	78.68	78.65
PubMed	**37.91**	**38.86**	**78.93**	**79.28**
Wiki	36.15	35.63	78.45	78.99
Yelp	33.92	34.25	78.48	78.78

Table 6: Impact of hyper-parameter setting on the effectiveness of pretrained word vectors. 'Opt' is hyper-parameter setting proposed in (Chiu et al., 2016), whereas 'Def' is the default setting in word2vec.

These results indicate that a small similar source reduces the computational cost without sacrificing the performance. This is especially important in practice, because collecting data and pretraining models are expensive. For example, a LM pretrained on 1 billion tokens takes three weeks to train on 32 GPUs (Jozefowicz et al., 2016).

Comparison to other hyper-parameter settings Chiu et al. (2016) propose a hyper-parameter combination of skip-gram model that is empirically

---

[3] https://nlp.stanford.edu/projects/glove/

[4] https://allennlp.org/elmo

identified on NER tasks. They find that a narrow context window size can boost the performance since it can capture better word function rather than domain similarity. We use their proposed hyper-parameter setting to train word vectors on different source data, and evaluate these pretrained word vectors on the ScienceIE and WetLab data sets. The reason for hand-picking these two is that benefits of pretrained word vectors on these two sets vary with a large margin. Our results suggest that this hyper-parameter setting can overall (except Wiki-ScienceIE and MIMIC-WetLab pairs) produce better performance compare to the default setting (Table 6). Most importantly we observe that our observation that similar sources generate better pretrained models can still holds with these hyper-parameters: PubMed, which is the most similar source to both target data sets, still outperforms other sources.

7.3 Controlling for source data size

To further investigate how source data size affects pretrained word vectors and LMs for NER tasks, we sample six PubMed subsets of different size. For target data sets, we use CoNLL2003, to which PubMed is the most dissimilar source, and JNLPBA, to which PubMed is the most similar source. We observe that 500 MB of pretraining data appears to be sufficient to calculate similarity, and capping factors out the impact of size (Figure 2). As discussed, VCcR is the most influential factor affecting the usefulness of pretrained word vectors for NER task. Increasing source data size may provide a larger vocabulary intersection with the target data set, but the resulting absolute F_1 score increase is less than 0.5, after the source data has been large enough. We also observe that if source and target data are dissimilar (PubMed-CoNLL2003 pair), pretrained word vectors is a better option than pretrained LMs, no matter how large source data is. However, pretrained LMs outperform pretrained word vectors, if source is similar to target (PubMed-JNLPBA pair).

We leave exploration of the combined effect of size and similarity to future work, but believe size should be considered separately, noting that results here suggest that similarity is more important.

8 Conclusions

We studied whether there are cost-effective methods to identify data sets to pretrain word vectors and LMs that are building blocks of NER models. We proposed using three measures, Target Vocabulary Covered, Language Model Perplexity, and Word Vector Variance, to measure different aspects of similarity between source and target data. We investigated how these measures correlate with the effectiveness of pretrained word vectors and LMs for NER tasks. We found that the effectiveness of pretrained word vectors strongly depends on whether the source data have a high vocabulary intersection with target data, while pretrained LMs can gain more benefits from a similar source. While different NLP tasks may rely on different aspects of language, our study is a step towards systematically guiding researchers on their choice of data for pretraining. As a future study, we will explore how these similarity measures predict performance of pretrained models in other NLP tasks.

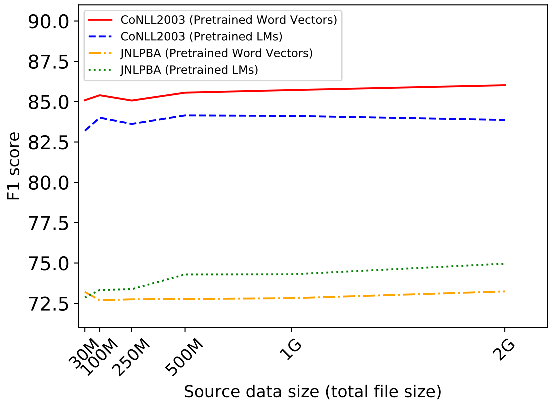


Figure 2: Impact of source data size on the effectiveness of pretrained models for NER.

Acknowledgments

We would like to thank Massimo Piccardi and Mark Dras for their constructive feedback. The authors also thank the members of CSIRO Data61's Language and Social Computing (LASC) team for helpful discussions, as well as anonymous reviewers for their insightful comments.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*, pages 1638–1649, Santa Fe, New Mexico.

Mikel Artetxe, Gorka Labaka, Inigo Lopez-Gazpio, and Eneko Agirre. 2018. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *CoNLL*, pages 282–291, Brussels, Belgium.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *SemEval*, pages 546–555, Vancouver, Canada.

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, and Lawrence E Hunter. 2012. Concept annotation in the CRAFT corpus. *BMC bioinformatics*, 13:161.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128, Sydney, Australia.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, Lisbon, Portugal.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1).

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR abs/1312.3005*.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *BioNLP*, pages 166–174, Berlin, Germany.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. In *NAACL*, Minneapolis, Minnesota.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *BioNLP*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *EACL*, pages 1107–1116, Valencia, Spain.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, Minneapolis, Minnesota.

Joseph Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychol. Bull.*, 76(5):378.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *WMT*, pages 187–197, Edinburgh, Scotland.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Comput. Linguist.*, 41(4):665–695.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR abs/1508.01991*.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Sci. Data*, 3:160035.

Mark Johnson, Peter Anderson, Mark Dras, and Mark Steedman. 2018. Predicting accuracy on large datasets from smaller pilot data. In *ACL*, pages 450–455, Melbourne, Australia.

Aditya Joshi, Xiang Dai, Sarvnaz Karimi, Ross Sparks, Cecile Paris, and C Raina MacIntyre. 2018. Shot or not: Comparison of NLP approaches for vaccination behaviour detection. In *EMNLP-SMM4H*, pages 43–47.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR abs/1602.02410*.

Sarvnaz Karimi, Xiang Dai, Hamedh Hassanzadeh, and Anthony Nguyen. 2017. Automatic diagnosis coding of radiology reports: A comparison of deep learning and conventional classification methods. In *BioNLP*, pages 328–332, Vancouver, Canada,.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. CADEC: A corpus of adverse drug event annotations. *J Biomed Inform*, 55:73–81.

Youngjun Kim, Ellen Riloff, and Stéphane M Meystre. 2017. Exploiting unlabeled texts with clustering-based instance selection for medical relation classification. In *AMIA*, pages 1060–1069.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An annotated corpus for machine reading of instructions in wet lab protocols. In *NAACL*, pages 97–106, New Orleans, Louisiana.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *ICML*, pages 957–966, Lille, France.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270, San Diego, California.

David YW Lee. 2001. Genres, registers, text types, domains and styles: Clarifying the concepts and nevigating a path through the BNC jungle. *LLT*, 5(3):37–72.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Quanzhi Li, Sameena Shah, Xiaomo Liu, and Armineh Nourbakhsh. 2017. Data sets: Word embeddings learned from tweets and general data. In *ICWSM*, pages 428–436, Montreal, Canada.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. What's in a domain? learning domain-robust text representations using adversarial training. In *NAACL*, pages 474–479, New Orleans, Louisiana.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, pages 1064–1074, Berlin, Germany.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR abs/1609.07843*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR abs/1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL and IJCNLP*, pages 425–430, Beijing, China.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *TACL*, 4:61–74.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*, pages 1756–1765, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237, New Orleans, Louisiana.

Yanjun Qi, Ronan Collobert, Pavel Kuksa, Koray Kavukcuoglu, and Jason Weston. 2009. Combining labeled and unlabeled data with word-class distribution learning. In *CIKM*, pages 1737–1740, Hong Kong, China.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Guillermo Solano-Flores. 2006. Language, dialect, and register: Sociolinguistics and the estimation of measurement error in the testing of english language learners. *Teach. Coll. Rec.*, 108(11):2354.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394, Uppsala, Sweden.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *COLING*, pages 3879–3889, Santa Fe, New Mexico.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *NAACL*, pages 672–682, Denver, Colorado.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *CoNLL*, pages 204–214, Beijing, China.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Predicting Annotation Difficulty to Improve Task Routing and Model Performance for Biomedical Information Extraction

Yinfei Yang
Google AI
yinfeiy@google.com

Oshin Agarwal
University of Pennsylvania
oagarwal@seas.upenn.edu

Chris Tar
Google AI
ctar@google.com

Byron C. Wallace
Northeastern University
b.wallace@northeastern.edu

Ani Nenkova
University of Pennsylvania
nenkova@seas.upenn.edu

Abstract

Modern NLP systems require high-quality annotated data. In specialized domains, expert annotations may be prohibitively expensive. An alternative is to rely on crowdsourcing to reduce costs at the risk of introducing noise. In this paper we demonstrate that directly modeling instance difficulty can be used to improve model performance, and to route instances to appropriate annotators. Our difficulty prediction model combines two learned representations: a 'universal' encoder trained on out-of-domain data, and a task-specific encoder. Experiments on a complex biomedical information extraction task using expert and lay annotators show that: (i) simply excluding from the training data instances predicted to be difficult yields a small boost in performance; (ii) using difficulty scores to weight instances during training provides further, consistent gains; (iii) assigning instances predicted to be difficult to domain experts is an effective strategy for task routing. Our experiments confirm the expectation that for specialized tasks expert annotations are higher quality than crowd labels, and hence preferable to obtain if practical. Moreover, augmenting small amounts of expert data with a larger set of lay annotations leads to further improvements in model performance.

1 Introduction

Assembling training corpora of annotated natural language examples in specialized domains such as biomedicine poses considerable challenges. Experts with the requisite domain knowledge to perform high-quality annotation tend to be expensive, while lay annotators may not have the necessary knowledge to provide high-quality annotations. A practical approach for collecting a sufficiently large corpus would be to use crowdsourcing platforms like Amazon Mechanical Turk (MTurk). However, crowd workers in general are likely to provide noisy annotations (Abad and Moschitti, 2016; Plank et al., 2014; Alonso et al., 2015), an issue exacerbated by the technical nature of specialized content. Some of this noise may reflect worker quality and can be modeled (Abad and Moschitti, 2016; Plank et al., 2014; Cohn and Specia, 2013; Nguyen et al., 2017), but for some instances lay people may simply lack the domain knowledge to provide useful annotation.

In this paper we report experiments on the EBM-NLP corpus comprising crowdsourced annotations of medical literature (Nye et al., 2018). We operationalize the concept of annotation difficulty and show how it can be exploited during training to improve information extraction models. We then obtain expert annotations for the abstracts predicted to be most difficult, as well as for a similar number of randomly selected abstracts. The annotation of highly specialized data and the use of lay and expert annotators allow us to examine the following key questions related to lay and expert annotations in specialized domains:

Can we predict item difficulty? We define a training instance as difficult if a lay annotator or an automated model disagree on its labeling. We show that difficulty can be predicted, and that it is distinct from inter-annotator agreement. Further, such predictions can be used during training to improve information extraction models.

Are there systematic differences between expert and lay annotations? We observe decidedly lower agreement between lay workers as compared to domain experts. Lay annotations have high precision but low recall with respect to expert annota-

Proceedings of NAACL-HLT 2019, pages 1471–1480
Minneapolis, Minnesota, June 2 - June 7, 2019.

tions in the new data that we collected. More generally, we expect lay annotations to be lower quality, which may translate to lower precision, recall, or both, compared to expert annotations.

Can one rely solely on lay annotations? Reasonable models can be trained using lay annotations alone, but similar performance can be achieved using markedly less expert data. This suggests that the optimal ratio of expert to crowd annotations for specialized tasks will depend on the cost and availability of domain experts. Expert annotations are preferable whenever its collection is practical. But in real-world settings, a combination of expert and lay annotations is better than using lay data alone.

Does it matter what data is annotated by experts? We demonstrate that a system trained on combined data achieves better predictive performance when experts annotate difficult examples rather than instances selected at i.i.d. random.

Our contributions in this work are summarized as follows. We define a task difficulty prediction task and show how this is related to, but distinct from, inter-worker agreement. We introduce a new model for difficulty prediction combining learned representations induced via a pre-trained 'universal' sentence encoder (Cer et al., 2018), and a sentence encoder learned from scratch for this task. We show that predicting annotation difficulty can be used to improve the task routing and model performance for a biomedical information extraction task. Our results open up a new direction for ensuring corpus quality. We believe that item difficulty prediction will likely be useful in other, non-specialized tasks as well, and that the most effective data collection in specialized domains requires research addressing the fundamental questions we examine here.

2 Related Work

Crowdsourcing annotation is now a well-studied problem (Snow et al., 2008; Abad and Moschitti, 2016; Plank et al., 2014; Alonso et al., 2015). Due to the noise inherent in such annotations, there have also been considerable efforts to develop aggregation models that minimize noise (Abad and Moschitti, 2016; Plank et al., 2014; Cohn and Specia, 2013; Nguyen et al., 2017).

There are also several surveys of crowdsourcing in biomedicine specifically (Good and Su, 2013; Khare et al., 2015; Lee et al., 2017). Some work in this space has contrasted model performance achieved using expert vs. crowd annotated training data (Zhai et al., 2013; Cocos et al., 2017; Dumitrache et al., 2018). Dumitrache *et al.* (2018) concluded that performance is similar under these supervision types, finding no clear advantage from using expert annotators. This differs from our findings, perhaps owing to differences in design. The experts we used already hold advanced medical degrees, for instance, while those in prior work were medical students. Furthermore, the task considered here would appear to be of greater difficulty: even a system trained on $\sim$5k instances performs reasonably, but far from perfect. By contrast, in some of the prior work where experts and crowd annotations were deemed equivalent, a classifier trained on 300 examples can achieve very high accuracy (Cocos et al., 2017).

More relevant to this paper, prior work has investigated methods for 'task routing' in active learning scenarios in which supervision is provided by heterogeneous labelers with varying levels of expertise (Yan et al., 2011; Donmez and Carbonell, 2008; Nguyen et al., 2015; Wallace et al., 2011; Yan et al., 2011). The related question of whether effort is better spent collecting additional annotations for already labeled (but potentially noisily so) examples or novel instances has also been addressed (Sheng et al., 2008). What distinguishes the work here is our focus on providing an operational definition of instance *difficulty*, showing that this can be predicted, and then using this to inform task routing.

3 Application Domain

Our specific application concerns annotating abstracts of articles that describe the conduct and results of randomized controlled trials (RCTs). Experimentation in this domain has become easy with the recent release of the EBM-NLP (Nye et al., 2018) corpus, which includes a reasonably large training dataset annotated via crowdsourcing, and a modest test set labeled by individuals with advanced medical training. More specifically, the training set comprises 4,741 medical article abstracts with crowdsourced annotations indicating snippets (sequences) that describe the Participants (P), Interventions (I), and Outcome (O) elements of the respective RCT, and the test set is composed of 191 abstracts with P, I, O sequence annotations from three medical experts.

Difficult Sentences

[Population]
1. Primary RP were screened and assigned to either the nifedipine SR group (Group N) or the Ginkgo biloba extract group (Group G) in the ratio of 2:1 .
2. A positive correlation was found for all methods in the controls (r=0.83-0.94) and RA patients (r=0.51-0.69).

[Interventions/Comparators]
1. They were all enrolled in mainstream compulsory education.
2. RA patients reported that they were less sedentary and engaged in more higher intensity PA than what was objectively assessed.

[Outcomes]
1. To develop a cycle - based risk prediction model for neutropenic complications (NC) during chemotherapy with doxorubicin (DOX) or a pegy lated liposomal formulation (PLD) for patients with metastatic breast cancer (MBC).
2. The purpose of this study was to evaluate the long-term safety and efficacy of risperidone.

Easy Sentences

[Population]
1. A prospective study in 80 patients was carried out.
2. We studied 200 women aged 35 years and older who had a family history of breast cancer in a first-degree relative.

[Interventions/Comparators]
1. Hormonal contraceptives are used widely but their effects on HIV-1 risk are unclear.
2. In the second group, custom - fit MRI - based pin guides were used.

[Outcomes]
1. Extrapyramidal AEs were reported in 6(8%) patients.
2. The overall progression - free survival rates were similar between the two arms (P=.095).

Table 1: Example sentences are difficult or easy to annotate for crowd workers. The underlined text are reference annotations from domain experts.

Table 1 shows an example of difficult and easy examples according to our definition of difficulty. The underlined text demarcates the (consensus) reference label provided by domain experts. In the difficult examples, crowd workers marked text distinct from these reference annotations; whereas in the easy cases they reproduced them with reasonable fidelity. The difficult sentences usually exhibit complicated structure and feature jargon.

An abstract may contain some ‘easy’ and some ‘difficult’ sentences. We thus perform our analysis at the sentence level. We split abstracts into sentences using spaCy.[1] We excluded sentences that comprise fewer than two tokens, as these are likely an artifact of errors in sentence splitting. In total, this resulted in 57,505 and 2,428 sentences in the train and test set abstracts, respectively.

4 Quantifying Task Difficulty

The test set includes annotations from both crowd workers and domain experts. We treat the latter as ground truth and then define the difficulty of sentences in terms of the observed agreement between expert and lay annotators. Formally, for annotation task t and instance i:

$$\text{Difficulty}_{ti} = \frac{\sum_{j=1}^{n} f(\text{label}_{ij}, y_i)}{n} \quad (1)$$

where f is a scoring function that measures the quality of the label from worker j for sentence i, as compared to a ground truth annotation, y_i. The difficulty score of sentence i is taken as an average over the scores for all n layworkers. We use Spearmans’ correlation coefficient as a scoring function. Specifically, for each sentence we create two vectors comprising counts of how many times each token was annotated by crowd and expert workers, respectively, and calculate the correlation between these. Sentences with no labels are treated as maximally easy; those with *only* either crowd worker or expert label(s) are assumed maximally difficult.

The training set contains only crowdsourced annotations. To label the training data, we use a 10-fold validation like setting. We iteratively retrain the LSTM-CRF-Pattern sequence tagger of Patel *et al.* (2018) on 9 folds of the training data and use that trained model to predict labels for the 10th. In this way we obtain predictions on the full training set. We then use predicted spans as proxy ‘ground truth’ annotations to calculate the diffi-

---

[1] `https://spacy.io/`

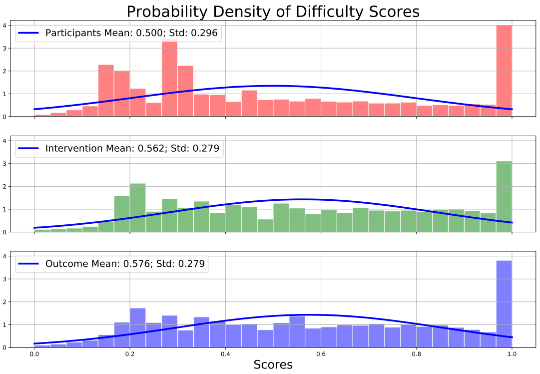


Figure 1: Distributions of difficulty scores over all sentences that contain any span annotations

culty score of sentences as described above; we normalize these to the $[0, 1]$ interval. We validate this approximation by comparing the proxy scores against reference scores over the test set, the Pearson's correlation coefficients are 0.57 for Population, 0.71 for Intervention and 0.68 for Outcome.

There exist many sentences that contain neither manual nor predicted annotations. We treat these as maximally easy sentences (with difficulty scores of 0). Such sentences comprise 51%, 42% and 36% for Population, Interventions and Outcomes data respectively, indicating that it is easier to identify sentences that have no Population spans, but harder to identify sentences that have no Interventions or Outcomes spans. This is intuitive as descriptions of the latter two tend to be more technical and dense with medical jargon.

We show the distribution of the automatically labeled scores for sentences that do contain spans in Figure 1. The mean of the Population (P) sentence scores is significantly lower than that for other types of sentences (I and O), again indicating that they are easier on average to annotate. This aligns with a previous finding that annotating Interventions and Outcomes is more difficult than annotating Participants (Nye et al., 2018).

Many sentences contain spans tagged by the LSTM-CRF-Pattern model, but missed by all crowd workers, resulting in a maximally difficult score (1). Inspection of such sentences revealed that some are truly difficult examples, but others are tagging model errors. In either case, such sentences have confused workers and/or the model, and so we retain them all as 'difficult' sentences.

Content describing the P, I and O, respectively, is quite different. As such, one sentence usually contains (at most) only one of these three content types. We thus treat difficulty prediction for the respective label types as separate tasks.

Workers	P	I	O
crowd workers	0.52	0.43	0.41
domain experts	0.74	0.68	0.57

Table 2: Average inter-worker agreement.

5 Difficulty is not Worker Agreement

Our definition of difficulty is derived from agreement between expert and crowd annotations for the test data, and agreement between a predictive model and crowd annotations in the training data. It is reasonable to ask if these measures are related to inter-annotator agreement, a metric often used in language technology research to identify ambiguous or difficult items. Here we explicitly verify that our definition of difficulty only weakly correlates with inter-annotator agreement.

We calculate inter-worker agreement between crowd and expert annotators using Spearman's correlation coefficient. As shown in Table 2, average agreement between domain experts are considerably higher than agreements between crowd workers for all three label types. This is a clear indication that the crowd annotations are noisier.

Furthermore, we compare the correlation between inter-annotator agreement and difficulty scores in the training data. Given that the majority of sentences do not contain a PICO span, we only include in these calculations those that contain a reference label. Pearson's r are 0.34, 0.30 and 0.31 for P, I and O, respectively, confirming that inter-worker agreement and our proposed difficulty score are quite distinct.

6 Predicting Annotation Difficulty

We treat difficulty prediction as a regression problem, and propose and evaluate neural model variants for the task. We first train RNN (Chung et al., 2014) and CNN (Kim, 2014) models.

We also use the *universal sentence encoder* (USE) (Cer et al., 2018) to induce sentence representations, and train a model using these as features. Following (Cer et al., 2018), we then experiment with an ensemble model that combines the 'universal' and task-specific representations to predict annotation difficulty. We expect these universal embeddings to capture general, high-level semantics, and the task specific representations to capture more granular information. Figure 2 de-

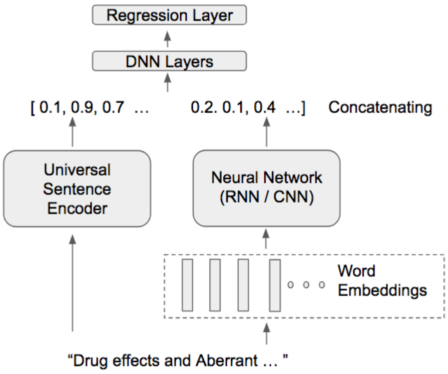


Figure 2: Model architecture.

picts the model architecture. Sentences are fed into both the universal sentence encoder and, separately, a task specific neural encoder, yielding two representations. We concatenate these and pass the combined vector to the regression layer.

6.1 Experimental Setup and Results

We trained models for each label type separately. Word embeddings were initialized to 300d GloVe vectors (Pennington et al., 2014) trained on common crawl data;[2] these are fine-tuned during training. We used the Adam optimizer (Kingma and Ba, 2014) with learning rate and decay set to 0.001 and 0.99, respectively. We used batch sizes of 16.

We used the large version of the universal sentence encoder[3] with a transformer (Vaswani et al., 2017). We did not update the pretrained sentence encoder parameters during training. All hyperparamaters for all models (including hidden layers, hidden sizes, and dropout) were tuned using Vizier (Golovin et al., 2017) via 10-fold cross validation on the training set maximizing for F1.[4]

As a baseline, we also trained a linear Support-Vector Regression (Fan et al., 2008) model on n-gram features (n ranges from 1 to 3).[5]

Table 3 reports Pearson correlation coefficients between the predictions with each of the neural models and the ground truth difficulty scores. Rows 1-4 correspond to individual models, and row 5 reports the ensemble performance. Columns correspond to label type. Results from all models outperform the baseline SVR model: Pearson's correlation coefficients range from 0.550 to 0.622. The regression correlations are the lowest.

	P	I	O
NGRAM+SVR	0.455	0.311	0.541
RNN	0.521	0.555	0.601
CNN	0.470	0.522	0.550
USE	0.492	0.518	0.580
USE+RNN	0.550	0.604	0.622

Table 3: Pearson correlation coefficients of sentence difficulty predictions.

The RNN model realizes the strongest performance among the stand-alone (non-ensemble) models, outperforming variants that exploit CNN and USE representations. Combining the RNN and USE further improves results. We hypothesize that this is due to complementary sentence information encoded in universal representations.

For all models, correlations for Intervention and Outcomes are higher than for Population, which is expected given the difficulty distributions in Figure 1. In these, the sentences are more uniformly distributed, with a fair number of difficult and easier sentences. By contrast, in Population there are a greater number of easy sentences and considerably fewer difficult sentences, which makes the difficulty ranking task particularly challenging.

7 Better IE with Difficulty Prediction

We next present experiments in which we attempt to use the predicted difficulty during training to improve models for information extraction of descriptions of Population, Interventions and Outcomes from medical article abstracts. We investigate two uses: (1) simply removing the most difficult sentences from the training set, and, (2) reweighting the most difficult sentences.

We again use LSTM-CRF-Pattern as the base model and experimenting on the EBM-NLP corpus (Nye et al., 2018). This is trained on either (1) the training set with difficult sentences removed, or (2) the full training set but with instances reweighted in proportion to their predicted difficulty score. Following (Nye et al., 2018), we use the Adam optimizer with learning rate of 0.001, decay 0.9, batch size 20 and dropout 0.5. We use pretrained 200d GloVe vectors (Pennington et al., 2014)[6] to initialize word embeddings, and use

---

[2] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

[3] `https://tfhub.dev/google/universal-sentence-encoder-large/3`

[4] This performs random search over the number of hidden layers (1-5), hidden sizes (128-1024), and dropout (0.1- 0.5).

[5] We perform gridsearch over the hyperparameter C.

[6] `http://nlp.stanford.edu/data/glove.6B.zip`

Model	Precision			Recall			F1		
	P	I	O	P	I	O	P	I	O
Base model	*81.54*	81.99	78.01	64.22	54.19	54.84	71.85	65.25	64.40
Re-weight by agreement	78.63	*82.24*	*82.19*	66.47	54.45	55.22	72.04	65.53	66.06
Re-weight by difficulty	79.57	74.69	73.92	*70.31*	*63.71*	*64.96*	**74.65**	**68.76**	**69.15**

Table 4: Medical IE performance by re-weighting sentences according to predicted agreement or difficulty scores.

100d hidden char representations. Each word is thus represented with 300 dimensions in total. The hidden size is 100 for the LSTM in the character representation component, and 200 for the LSTM in the information extraction component. We train for 15 epochs, saving parameters that achieve the best F1 score on a nested development set.

7.1 Removing Difficult Examples

We first evaluate changes in performance induced by training the sequence labeling model using less data by removing difficult sentences prior to training. The hypothesis here is that these difficult instances are likely to introduce more noise than signal. We used a cross-fold approach to predict sentence difficulties, training on 9/10ths of the data and scoring the remaining 1/10th at a time. We then sorted sentences by predicted difficulty scores, and experimented with removing increasing numbers of these (in order of difficulty) prior to training the LSTM-CRF-Pattern model.

Figure 3 shows the results achieved by the LSTM-CRF-Pattern model after discarding increasing amounts of the training data: the x and y axes correspond to the the percentage of data removed and F1 scores, respectively. We contrast removing sentences predicted to be difficult with removing them *(a)* randomly (i.i.d.), and, *(b)* in inverse order of predicted inter-annotator agreement. The agreement prediction model is trained exactly the same like difficult prediction model, with simply changing the difficult score to annotation agreement. F1 scores actually improve (marginally) when we remove the most difficult sentences, up until we drop 4% of the data for Population and Interventions, and 6% for Outcomes. Removing training points at i.i.d. random degrades performance, as expected. Removing sentences in order of disagreement seems to have similar effect as removing them by difficulty score when removing small amount of the data, but the F1 scores drop much faster when removing more data. These findings indicate that sentences predicted to be difficult are indeed noisy, to the extent that they do not seem to provide the model useful signal.

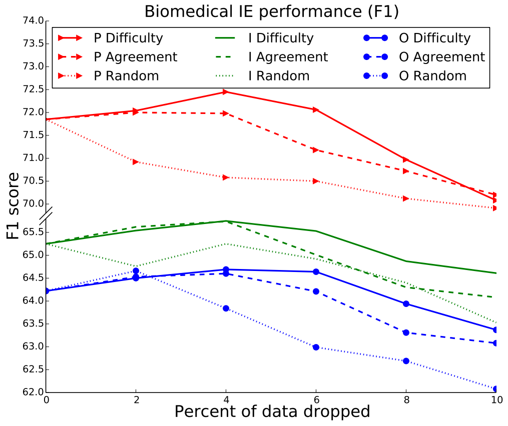


Figure 3: F1 scores achieved when removing increasingly large fractions of the training data.

7.2 Re-weighting by Difficulty

We showed above that removing a small number of the most difficult sentences does not harm, and in fact modestly improves, medical IE model performance. However, using the available data we are unable to test if this will be useful in practice, as we would need additional data to determine how many difficult sentences should be dropped.

We instead explore an alternative, practical means of exploiting difficulty predictions: we re-weight sentences during training inversely to their *predicted* difficulty. Formally, we weight sentence i with difficulty scores above τ according to: $1-a\cdot(d_i-\tau)/(1-\tau)$, where d_i is the difficulty score for sentence i, and a is a parameter codifying the minimum weight value. We set τ to 0.8 so as to only re-weight sentences with difficulty in the top 20th percentile, and we set a to 0.5. The re-weighting is equivalent to down-sampling the difficult sentences. LSTM-CRF-Pattern is our base model.

Table 4 reports the precision, recall and F1 achieved both with and without sentence re-weighting. Re-weighting improves all metrics modestly but consistently. All F1 differences are statistically significant under a sign test ($p < 0.01$). The model with best precision is different for Patient, Intervention and Outcome labels. However re-weighting by difficulty does consistently yield the best recall for all three extraction types, with the most notable improvement for I and O, where recall improved by 10 percentage points. This performance increase translated to improvements in F1 across all types, as compared to the base model and to re-weighting by agreement.

8 Involving Expert Annotators

The preceding experiments demonstrate that re-weighting difficult sentences annotated by the crowd generally improves the extraction models. Presumably the performance is influenced by the annotation quality.

We now examine the possibility that the higher quality and more consistent annotations of domain experts on the difficult instances will benefit the extraction model. This simulates an annotation strategy in which we route difficult instances to domain experts and easier ones to crowd annotators. We also contrast the value of difficult data to that of an i.i.d. random sample of the same size, both annotated by experts.

8.1 Expert annotations of Random and Difficult Instances

We re-annotate by experts a subset of most difficult instances and the same number of random instances. As collecting annotations from experts is slow and expensive, we only re-annotate the difficult instances for the interventions extraction task. We re-annotate the abstracts which cover the sentences with predicted difficulty scores in the top 5 percentile. We rank the abstracts from the training set by the count of difficult sentences, and re-annotate the abstracts that contain the most difficult sentences. Constrained by time and budget, we select only 2000 abstracts for re-annotation; 1000 of these are top-ranked, and 1000 are randomly sampled. This re-annotation cost $3,000. We have released the new annotation data at: `https://github.com/bepnye/EBM-NLP`.

Following (Nye et al., 2018), we recruited five medical experts via Up-work[7] with advanced medical training and strong technical reading/writing skills. The expert annotator were asked to read the entire abstract and highlight, using the BRAT toolkit (Stenetorp et al., 2012), all spans describing medical Interventions. Each abstract is only annotated by one expert. We examined 30 re-annotated abstracts to ensure the annotation quality before hiring the annotator.

Table 5 presents the results of LSTM-CRF-Pattern model trained on the reannotated difficult subset and the random subset. The first two rows show the results for models trained with expert annotations. The model trained on random data has a slightly better F1 than that trained on the same amount of difficult data. The model trained on random data has higher precision but lower recall.

Rows 3 and 4 list the results for models trained on the same data but with crowd annotation. Models trained with expert-annotated data are clearly superior to those trained with crowd labels with respect to F1, indicating that the experts produced higher quality annotations. For crowdsourced annotations, training the model with data sampled at i.i.d. random achieves 2% higher F1 than when difficult instances are used. When expert annotations are used, this difference is less than 1%. This trend in performance may be explained by differences in annotation quality: the randomly sampled set was more consistently annotated by both experts and crowd because the difficult set is harder. However, in both cases expert annotations are better, with a bigger difference between the expert and crowd models on the difficult set.

The last row is the model trained on all 5k abstracts with crowd annotations. Its F1 score is lower than either expert model trained on only 20% of data, suggesting that expert annotations should be collected whenever possible. Again the crowd model on complete data has higher precision than expert models but its recall is much lower.

8.2 Routing To Experts or Crowd

So far a system was trained on one type of data, either labeled by crowd or experts. We now examine the performance of a system trained on data that was routed to either experts or crowd annotators depending on their predicted difficult. Given the results presented so far mixing annotators may be beneficial given their respective trade-offs of precision and recall. We use the annotations from experts for an abstract if it exists otherwise use crowd

---

[7] https://www.upwork.com

	Precision	**Recall**	**F1**
Difficult-Expert	68.46	65.06	66.72
Random-Expert	70.84	63.46	67.04
Difficult-Crowd	83.68	44.63	58.45
Random-Crowd	78.55	49.23	60.52
Base (All-Crowd)	81.99	54.19	65.25

Table 5: Interventions IE model performance trained crowd or expert. The first four models are trained with a subset of 1k abstracts and the base model is trained with all 5k abstracts.

	Precision	**Recall**	**F1**
(D)ifficult-Expert	68.46	65.06	66.72
(R)andom-Expert	70.84	63.46	67.04
D+R	68.57	67.54	69.62
D+Other	83.55	60.73	70.33
R+Other	85.10	57.08	68.34
D+R+Other	82.19	61.88	70.60

Table 6: Interventions IE model performance trained by mixing annotations from experts and crowd workers. [D]: Difficult-Expert; [R]: Random-Expert; [Other]: the rest of the abstracts with crowd annotation only.

annotations. The results are presented in Table 6.

Rows 1 and 2 repeat the performance of the models trained on difficult subset and random subset with expert annotations only respectively. The third row is the model trained by combining difficult and random subsets with expert annotations. There are around 250 abstracts in the overlap of these two sets, so there are total 1.75k abstracts used for training the D+R model. Rows 4 to 6 are the models trained on all 5k abstracts with mixed annotations, where *Other* means the rest of the abstracts with crowd annotation only.

The results show adding more training data with crowd annotation still improves at least 1 point F1 score in all three extraction tasks. The improvement when the difficult subset with expert annotations is mixed with the remaining crowd annotation is 3.5 F1 score, much larger than when a random set of expert annotations are added. The model trained with re-annotating the difficult subset (D+Other) also outperforms the model with re-annotating the random subset (R+Other) by 2 points in F1. The model trained with re-annotating both of difficult and random subsets (D+R+Other), however, achieves only marginally higher F1 than the model trained with the re-annotated difficult subset (D+Other).

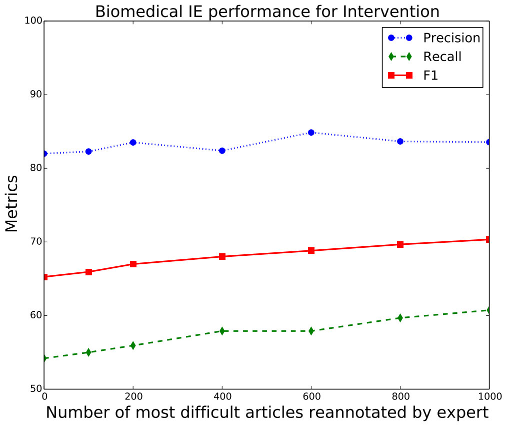


Figure 4: Precision/Recall/F1 as a function of the number of articles re-annotated by expert, in decreasing order of difficulty.

In sum, the results clearly indicate that mixing expert and crowd annotations leads to better models than using solely crowd data, and better than using expert data alone. More importantly, there is greater gain in performance when instances are routed according to difficulty, as compared to randomly selecting the data for expert annotators. These findings align with our motivating hypothesis that annotation quality for difficult instances is important for final model performance. They also indicate that mixing annotations from expert and crowd could be an effective way to achieve acceptable model performance given a limited budget.

8.3 How Many Expert Annotations?

We established that crowd annotation are still useful in supplementing expert annotations for medical IE. Obtaining expert annotations for the one thousand most difficult instances greatly improved the model performance. However the choice of how many difficult instances to annotate was an uninformed choice. Here we check if less expert data would have yielded similar gains. Future work will need to address how best to choose this parameter for a routing system.

We simulate a routing scenario in which we send consecutive batches of the most difficult examples to the experts for annotation. We track changes in performance as we increase the number of most-difficult-articles sent to domain experts. As shown in Figure 4, adding expert annotations for difficult articles consistently increases F1 scores. The performance gain is mostly from increased recall; the precision changes only a bit with higher quality annotation. This observation implies that crowd workers often fail to mark target tokens, but do not tend to produce large numbers of false positives. We suspect such failures to identify relevant spans/tokens are due to insufficient domain knowledge possessed by crowd workers.

The F1 score achieved after re-annotating the 600 most-difficult articles reaches 68.1%, which is close to the performance when re-annotating 1000 random articles. This demonstrates the effectiveness of recognizing difficult instances. The trend when we use up all expert data is still upward, so adding even more expert data is likely to further improve performance. Unfortunately we exhausted our budget and were not able to obtain additional expert annotations. It is likely that as the size of the expert annotations increases, the value of crowd annotations will diminish. This investigation is left for future work.

9 Conclusions

We have introduced the task of predicting annotation difficulty for biomedical information extraction (IE). We trained neural models using different learned representations to score texts in terms of their difficulty. Results from all models were strong with Pearsons correlation coefficients higher than 0.45 in almost all evaluations, indicating the feasibility of this task. An ensemble model combining universal and task specific feature sentence vectors yielded the best results.

Experiments on biomedical IE tasks show that removing up to $\sim$10% of the sentences predicted to be most difficult did not decrease model performance, and that re-weighting sentences inversely to their difficulty score during training improves predictive performance. Simulations in which difficult examples are routed to experts and other instances to crowd annotators yields the best results, outperforming the strategy of randomly selecting data for expert annotation, and substantially improving upon the approach of relying exclusively on crowd annotations. In future work, routing strategies based on instance difficulty could be further investigated for budget-quality trade-off.

Acknowledgements

This work has been partially supported by NSF1748771 grant. Wallace was support in part by NIH/NLM R01LM012086.

References

Azad Abad and Alessandro Moschitti. 2016. Taking the best from the crowd: Learning question passage classification from noisy data. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, *SEM@ACL 2016, Berlin, Germany, 11-12 August 2016*.

Héctor Martínez Alonso, Anders Johannsen, Oier Lopez de Lacalle, and Eneko Agirre. 2015. Predicting word sense annotation agreement. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics, LSDSem@EMNLP 2015, Lisbon, Portugal, September 18, 2015*, pages 89–94.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Anne Cocos, Ting Qian, Chris Callison-Burch, and Aaron J. Masino. 2017. Crowd control: Effectively utilizing unscreened crowd workers for biomedical data annotation. *Journal of Biomedical Informatics*, 69:86–92.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 32–42.

Pinar Donmez and Jaime G Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628. ACM.

Anca Dumitrache, Lora Aroyo, and Chris Welty. 2018. Crowdsourcing ground truth for medical relation extraction. *ACM Trans. Interact. Intell. Syst.*, 8(2):11:1–11:20.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. 2017. Google vizier: A service for black-box optimization. In *Proceedings of KDD '17*.

Benjamin M Good and Andrew I Su. 2013. Crowdsourcing for bioinformatics. *Bioinformatics*, 29(16):1925–1933.

Ritu Khare, Benjamin M Good, Robert Leaman, Andrew I Su, and Zhiyong Lu. 2015. Crowdsourcing in biomedicine: challenges and opportunities. *Briefings in bioinformatics*, 17(1):23–32.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Young Ji Lee, Janet A Arida, and Heidi S Donovan. 2017. The application of crowdsourcing approaches to cancer research: a systematic review. *Cancer medicine*, 6(11):2595–2605.

An T Nguyen, Byron C Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2017, page 299. NIH Public Access.

An Thanh Nguyen, Byron C Wallace, and Matthew Lease. 2015. Combining crowd and expert labels using decision theoretic active learning. In *Third AAAI conference on human computation and crowdsourcing*.

Benjamin Nye, Jessy Li, Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova, and Byron C. Wallace. 2018. A Corpus with Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova, and Byron Wallace. 2018. Syntactic patterns improve information extraction for medical search. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 742–751.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*. Curran Associates, Inc.

Byron C Wallace, Kevin Small, Carla E Brodley, and Thomas A Trikalinos. 2011. Who should label what? instance allocation in multiple expert active learning. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 176–187. SIAM.

Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G Dy. 2011. Active learning from crowds. In *ICML*, volume 11, pages 1161–1168.

Haijun Zhai, Todd Lingren, Louise Deleger, Qi Li, Megan Kaiser, Laura Stoutenborough, and Imre Solti. 2013. Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing. *Journal of medical Internet research*, 15(4).

Detecting Depression in Social Media using Fine-Grained Emotions

Mario Ezra Aragón⋆, **A. Pastor López-Monroy**†,
Luis C. González-Gurrola‡ and **Manuel Montes-y-Gómez**⋆
⋆ Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Mexico.
† Centro de Investigación en Matemáticas (CIMAT), Mexico.
‡ Facultad de Ingeniería, UACh, Mexico.
`mearagon@inaoep.mx, pastor.lopez@cimat.mx,`
`lcgonzalez@uach.mx, mmontesg@inaoep.mx`

Abstract

Nowadays social media platforms are the most popular way for people to share information, from work issues to personal matters. For example, people with health disorders tend to share their concerns for advice, support or simply to relieve suffering. This provides a great opportunity to proactively detect these users and refer them as soon as possible to professional help. We propose a new representation called Bag of Sub-Emotions (BoSE), which represents social media documents by a set of fine-grained emotions automatically generated using a lexical resource of emotions and subword embeddings. The proposed representation is evaluated in the task of depression detection. The results are encouraging; the usage of fine-grained emotions improved the results from a representation based on the core emotions and obtained competitive results in comparison to state of the art approaches.

1 Introduction

Mental Disorders affect millions of people around the world. Out of these disorders, depression has been ranked among the most common, even with a high incidence in mortality rates (Kessler et al., 2017; Mathers and Loncar, 2006). It is imperative then, to come with effective approaches to detect depression before it causes irreparable damage to mere individuals that suffer it and their loved ones. In a connected world where we live, it is very normal to share personal information, matters and concerns in social media platforms. This fact poses an opportunity, since the understanding of depression through the analysis of social media documents increases the chances to detect people that present signs of depression and could lead to provide them professional help (Guntuku et al., 2017; Pestian et al., 2010).

Several works in literature have explored how to use linguistic and sentiment analysis to detect depression (Xue et al., 2013). For example, in (Huang et al., 2014) the authors applied sentiment analysis (SA) to assign polarity to tweets. They count the number of positive, negative, neutral words, and the ratio of the negative and positive words, and found that depressed users post longer emotional tweets. The work of Wang et al. (2013) enriched SA with features derived from psychological research like the use of first person pronouns, user social interaction and user behaviors in micro blogs. An interesting finding is that the time of the posts is useful to detect people with high risk of committing suicide. In a recent work, Chen et al. (2018) proposed to use emotions with the aim to identify depression on Twitter users. That study openly exposed the potential of using discrete emotions as features, instead of only using linguistic features, and broad categories to represent them. To further investigate this latter point, in this study we propose to model emotions in a fine-grained way and use them to build a new representation to tackle the problem of detecting depression in users of social media. We construct these fine-grained emotions using lexical information extracted from emotions combined with subword embeddings. The leading hypothesis of our study is that emotions could be better, and more flexible, represented at a lower level, instead of only using broad categories such as as ”anger”, ”joy”, ”negative” or ”positive”.

2 The Bag of Sub-Emotions (BoSE) Representation

Figure 1 depicts ourproposed approach. In a first step, we compute a set of fine-grained emotions for each broad emotion described in the lexical resource by Mohammad and Turney (2013). Then,

Proceedings of NAACL-HLT 2019, pages 1481–1486
Minneapolis, Minnesota, June 2 - June 7, 2019.

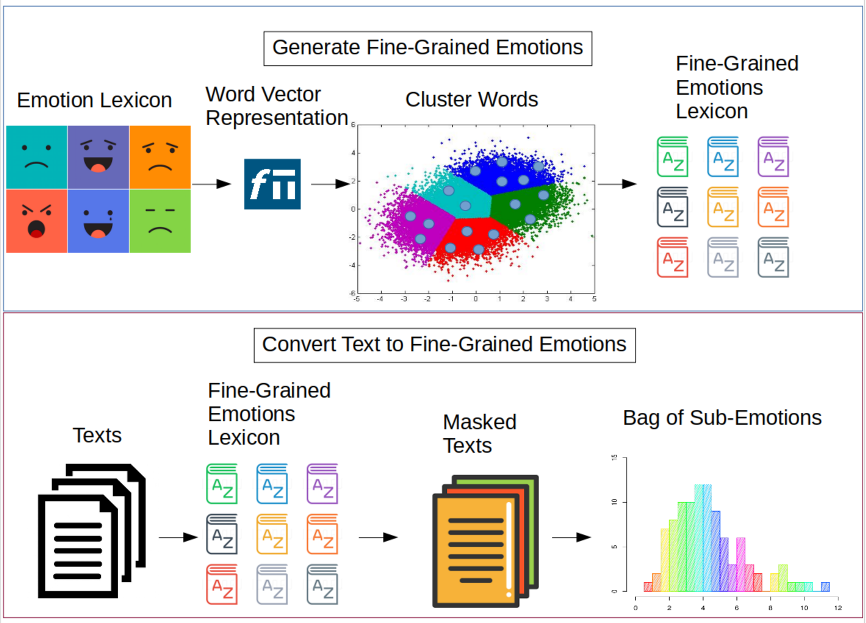


Figure 1: Diagram that represents the creation of the Bag of Sub-Emotions (**BoSE**) representation. First, Fine-Grained Emotions are generated from a given Emotion Lexicon; then, texts are masked using these fine-grained emotions and their histogram is build as final representation.

we use the obtained fine-grained emotions to mask the texts, eventually representing them by a histogram of their frequencies. Accordingly, we named this new representation **BoSE**, for Bag of Sub-Emotions. In the following sections we detail each step of our proposed approach.

2.1 Generating Fine-Grained Emotions

To generate the fine-grained emotions we use a lexical resource based on eight recognized emotions, e.g., Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise and Trust (Ekman and Davidson, 1994), and two main sentiments[1], Positive and Negative. We represent this as $E = \{E_1, E_2, ..., E_{10}\}$, where E is the set of emotions in the lexical resources and $E_i = \{w_1, .., w_n\}$ is the set of words associated to the emotion E_i. We compute a word vector for each word using pre-trained Wikipedia sub-word embeddings from FastText (Bojanowski et al., 2016) of size 300, and then we create subgroups of words by emotion using the *Affinity Propagation* (*AP*) clustering algorithm (Thavikulwat, 2008). This AP clustering algorithm has several appealing characteristics, e.g. it does not employ artificial elements (centroids) to create clusters and it does not require to specify the number of groups before running the algorithm. To have an idea of how the vocabulary was distributed among emotions and the number of generated clusters after applying AP to the lexical resource we present Table 1.

After this process, now each (broad) emotion is represented by a set of fine-grained emotions, $E_i = \{F_{i1}, ..., F_{ij}\}$, where each F_{ij} is a subset of the words from E_i and is represented by the average vector of their respective embeddings. These subgroups of words allow separating each broad emotion in different topics that help identify and capture the fine-grained emotions expressed by users in their posts. Figure 2 presents some examples of groups of fine-grained emotions that were automatically computed by our approach. We can appreciate that words with similar context tend to group together, as shown in each column. We can also notice that words corresponding to the same broad emotion consider very different topics, for example, in the Anger emotion, the group *anger3* is related to fighting and battles, whereas the group *anger2* is about growls or loud noises. Another interesting example are the groups from the Surprise emotion, where groups express different kinds of surprises like art and museums (*surprise2*), accidents and disasters (*surprise1*), as well as magic and illusions (*surprise3*).

---

[1] In the rest of the paper we refer to these sentiments as emotions as well.

Anger			Joy		
anger1	***anger2***	***anger3***	***joy1***	***joy2***	***joy3***
abomination	growl	battle	accomplish	bounty	charity
fiend	growling	combat	achieve	cash	foundation
inhuman	thundering	fight	gain	money	trust
abominable	snarl	battler	reach	reward	humanitarian
unholy	snort	fists	goal	wealth	charitable
Surprise			**Disgust**		
surprise1	***surprise2***	***surprise3***	***disgust1***	***disgust2***	***disgust3***
accident	art	magician	accusation	criminal	cholera
crash	museum	wizard	suspicion	homicide	epidemic
disaster	artwork	magician	complaint	delinquency	malaria
incident	gallery	illusionist	accuse	crime	aids
collision	visual	sorcerer	slander	enforcement	polio

Figure 2: Examples of words grouped by Fine-Grained Emotions.

Emotion	Vocabulary	Clusters
anger	6035	444
anticipation	5837	395
disgust	5285	367
fear	7178	488
joy	4357	318
sadness	5837	395
surprise	3711	274
trust	5481	383
positive	11021	740
negative	12508	818

Table 1: Size of the vocabulary for each emotion presented in the lexical resources, and number of generated clusters.

2.2 Converting Text to Fine-Grained Emotions

Text masking: In this step documents are masked by replacing each word with the label of its closest fine-grained emotion. To this end, we compute the vector representation of each word using sub-word embeddings from FastText, then we measure the distance of each word vector against the centroid vectors from all fine-grained emotions by means of the cosine similarity, and, finally, we substitute each word by the label of its closest fine-grained emotion. To illustrate this process consider the text "*Leave no stone unturned*", which will be masked as "*fear2 negative8 anger10 anticipation3*".

Text representation: Based on the masked documents, we build their **BoSE** representations computing a frequency histogram of their fine-grained emotions. To build these representations we follow two different approaches: *i*) similar to the Bag-of-Words representation we create a histogram counting the number of occurrences of each fine-grained emotion in the text, we refer to this representation as **BoSE-unigrams**, and *ii*) we create a histogram counting the number of occurrences of fine-grained emotion sequences in the text, we refer to this representation as **BoSE-ngrams**. For the latter representation, we tested different sizes and combinations of sequences; using unigrams and bigrams we obtained the best performance for this task.

3 Experimental Settings

Preprocessing: For our experiments, we normalized the texts by removing special characters and lowercasing all the words. After preprocessing we masked the texts using the fine-grained emotions.
Classification: Once built the BoSE representation, we selected the more relevant features (i.e., sequences of fine-grained emotions) using the chi^2 distribution X_k^2 (Walck, 2007). Then, we used a Support Vector Machine (SVM) with a linear kernel and $C = 1$ to classify the documents.
Baselines: To properly evaluate the relevance of using fine-grained emotions in the detection of depression, we considered a representation based on the occurrences of broad emotions and the words that do not have an associated emotion. We named this approach Bag-of-Emotions (BoE). We also compared our results against a Bag-of-Words representation based on word unigrams and n-grams, since they are the common baseline approaches for text classification. Additionally, we compared our results against the f_1 results from the participants of the eRisk 2017 and 2018 evaluation tasks (Losada et al., 2017, 2018).

Data set	**Training**		**Testing**	
	Dep	ND	Dep	ND
eRisk'17	83	403	52	349
eRisk'18	135	752	79	741

Table 2: Depression data sets used for experimentation. Each data set have two classes (Depressed = Dep, Non-Depressed = ND).

Data Collections: We evaluated our approach in the task of depression detection, using the data sets from the eRisk 2017 and 2018 evaluation tasks (Losada et al., 2017, 2018). These data sets contain Reddit posts for several users. The users which explicitly mentioned that were diagnosed with depression were automatically labeled as positive. Vague expressions like "I think I have depression" or "I'm depressed" were discarded, the rest of them were labeled as negative. Table 2 shows some numbers from these data sets; (Losada and Crestani, 2016) describes these collections in more detail.

4 Experimental Results

The goal of our first experiment was to evaluate the appropriateness of the BoSE representation to identify depressed users. To accomplish this, we compared its performance against the results from a traditional BOW representation as well as to a representation considering only the broad emotions. Table 3 shows the f_1 performance over the positive class for the BOW, BoE and BoSE approaches. It can be noticed that the BoSE representation outperforms both baseline results, particularly when sequences of fine-grained emotions were considered. To better characterize the BoSE representation, we evaluated it without considering the clusters associated to the positive and negative sentiments. We refer to these experiments as BoSE8. Results from this variant show a drop in performance, confirming that sentiment information is relevant to the identification of depressed users.

To further evaluate the relevance of the BoSE representation, Table 4 compares its results against those from the first three places at the eRisk 2017 and 2018 evaluation tasks (Losada et al., 2017, 2018). To contextualize this comparison, consider that the first place in both years (Trotzek et al., 2017, 2018) defined multiple strategies and considered a wide range of features to build their models, e.g., they extract readability features, LIWC features, user-level linguistic metadata, neural word embeddings, specific terms related to depression, and used models based on LSTM neural networks and convolutional neural networks, using four machine learning models in an ensemble model. Other top performers (Villegas et al., 2017) combined semantic representation considering partial information and temporal variation features. In (Funez et al., 2018) they implemented two models; one based on flexible temporal variation of terms and a second model based on sequential incremental classification.

Method	Dep'17	Dep'18
BoW-unigrams	0.59	0.58
BoE-unigrams	0.57	0.60
BoSE8-unigrams	0.56	0.60
BoSE-unigrams	**0.61**	**0.61**
BOW-ngrams	0.58	0.60
BoE-ngrams	0.61	0.58
BoSE8-ngrams	0.57	0.59
BoSE-ngrams	**0.64**	**0.63**

Table 3: F1 results over the positive class against baseline methods

Method	Dep'17	Dep'18
first place	**0.64**	**0.64**
second place	0.59	0.60
third place	0.53	0.58
BoSE-ngrams	**0.64**	0.63

Table 4: F1 results over the positive class against top performers at eRisk

From the obtained results we highlight the following observations:

1. Our approach outperformed the traditional BOW representation in both data sets, indicating that considering emotional information is quite relevant for the detection of depression in online communications.

2. The use of fine-grained emotions as features helps improving the results from a representation that only considers broad emotions. This result confirms our hypothesis that depressive users tend to express their emotions in a different way than non depressive users.

Examples of relevant sequences	
"anger1" "anger11-anticipation10" "disgust16-anger11" "disgust11-fear17"	
anger1	abandoned, deserted, unattended
anger11	unsociable, crowd, mischievous
anticip10	disappointed, inequality, infidelity
disgust16	unsatisfactory, dilution, influence
disgust11	insecurity, desolation, incursion
fear17	hysterical, immaturity, injury

Table 5: Examples of words that create the fine-grained emotions

3. Our approach obtained comparable results to the best reported approaches in both data sets. It is important to highlight that the participants of these tasks tested different complex models with a wide range of features and sophisticated approaches based on traditional and deep learning representations of texts, whereas ours only relies on the use of fine-grained emotions as features.

4.1 Analysis of the Fine-Grained Emotions

To offer a glimpse of what fine-grained emotions actually capture, we selected the most relevant sequences for the detection of depression according to the chi^2 distribution. Table 5 shows some relevant sequences of fine-grained emotions as well as some examples of the words that correspond to these sequences.

Most of the fine-grained emotions that present high relevance for the detection of depression are related to negative topics, for example, the anger emotion is associated to the feeling of abandonment or unsociable, and the disgust emotion is related to dilution, insecurity and desolation. These fine-grained emotions seems to capture the way a depressed user expresses about himself or his environment.

5 Conclusions and Future Work

In this study we proposed a new representation that creates fine-grained emotions that were automatically generated using a lexical resource of emotions and sub-word embeddings from FastText. Using these fine-grained emotions our approach can automatically capture more specific topics and emotions that are expressed in the documents by users that have depression. BoSE obtained better results than the proposed baselines and also improved the results of only using broad emotions. It is worth mentioning the simplicity and interpretability of our approach, which contrasts with the best previous eRisk competition methods that are much more complex and difficult to interpret (most of the participants used plenty of different features and a vast range of models, including deep). Our results encourage to attempt this approach based on fine-grained emotions in other relevant health and safety tasks such as the detection of anorexia and self-harm. In addition, we also plan to explore the learning of emotional-based representations by means of a deep neural network from which we could exploit local invariance properties to model fine-grained emotions.

Acknowledgments

This research was supported by CONACyT-Mexico (Scholarship 654803 and Project FC-2410).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, pages 135–146.

Xuetong Chen, Martin D. Sykora, Thomas W. Jackson, and Suzanne Elayan. 2018. What about mood swings? identifying depression on twitter with temporal measures of emotions. *Companion Proceedings of the The Web Conference 2018, International World Wide Web Conferences Steering Committee*, pages 1653–1660.

Paul Ed Ekman and Richard J Davidson. 1994. The nature of emotion: Fundamental questions. *New York, NY, US: Oxford University Press.*

Dario G. Funez, Ma. Jos Garciarena Ucelay, Ma. Paula Villegas, Sergio G. Burdisso, Leticia C. Cagnina, Manuel Montes-y Gmez, and Marcelo L. Errecalde. 2018. Unsls participation at erisk 2018 lab. *Proceedings of the 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France.*

Sharath Chandra Guntuku, David B Yaden, Margaret L Kern, Lyle H Ungar, and Johannes C Eichstaedt. 2017. Detecting depression and mental illness on social media: an integrative review. *Current Opinion in Behavioral Sciences*, pages 43–49.

Xiaolei Huang, Lei Zhang, Tianli Liu, David Chiu, Tingshao Zhu, and Xin Li. 2014. Detecting suicidal ideation in chinese microblogs with psychological lexicons. *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 844–849.

Ronald C Kessler, Evelyn J Bromet, Victoria Shahly Peter de Jonge, and Marsha. 2017. The burden of depressive illness. *Public Health Perspectives on Depressive Disorders*.

David E. Losada and Fabio Crestani. 2016. A test collection for research on depression and language use. *Proceedings of the 7th International Conference of the CLEF Association, CLEF 2016, Evora, Portugal*.

David E Losada, Fabio Crestani, and Javier Parapar. 2017. erisk 2017: Clef lab on early risk prediction on the internet: Experimental foundations. *Proceedings of the 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland*.

David E Losada, Fabio Crestani, and Javier Parapar. 2018. Overview of erisk 2018: Early risk prediction on the internet (extended lab overview). *Proceedings of the 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France*.

Colin D Mathers and Dejan Loncar. 2006. Projections of global mortality and burden of disease from 2002 to 2030. *PLOS Medicine, Public Library of Science*, pages 1–20.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, pages 436–465.

John P. Pestian, Henry Nasrallah, Pawel Matykiewicz, Aurora Bennett, and Antoon A. Leenaars. 2010. Suicide note classification using natural language processing: A content analysislin heidelberg. *Biomed Inform Insights*, page BII.S4706.

Precha Thavikulwat. 2008. Affinity propagation: A clustering algorithm for computer-assisted business simulation and experimental exercises. *Developments in Business Simulation and Experiential Learning*.

Marcel Trotzek, Sven Koitka, and Christoph M. Friedrich. 2017. Linguistic metadata augmented classifiers at the clef 2017 task for early detection of depression. *Proceedings of the 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland*.

Marcel Trotzek, Sven Koitka, and Christoph M. Friedrich. 2018. Word embeddings and linguistic metadata at the clef 2018 tasks for early detection of depression and anorexia. *Proceedings of the 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France*.

Ma. Paula Villegas, Dario G. Funez, Ma. Jos Gariarena Uelay, Letiia C. Cagnina, and Marcelo L. Errealde. 2017. Lidic - unsl's partiipation at erisk 2017: Pilot task on early detetion of depression notebook for erisk at clef 2017. *Proceedings of the 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland*.

C. Walck. 2007. Hand-book on statistical distributions for experimentalists. *University of Stockholm, Internal Report SUFPFY/9601*, pages 36–44.

Xinyu Wang, Chunhong Zhang, Yang Ji, Li Sun, Leijia Wu, and Zhana Bao. 2013. A depression detection model based on sentiment analysis in micro-blog social network. *Trends and Applications in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg*, pages 201–213.

Yuanyuan Xue, Qi Li, Li Jin, Ling Feng, David A. Clifton, and Gari D. Clifford. 2013. Detecting adolescent psychological pressures from micro-blog. *Health Information Science, Springer International Publishing*, pages 83–94.

A Silver Standard Corpus of Human Phenotype-Gene Relations

Diana Sousa*, Andre Lamurias and Francisco M. Couto

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract

Human phenotype-gene relations are fundamental to fully understand the origin of some phenotypic abnormalities and their associated diseases. Biomedical literature is the most comprehensive source of these relations, however, we need Relation Extraction tools to automatically recognize them. Most of these tools require an annotated corpus and to the best of our knowledge, there is no corpus available annotated with human phenotype-gene relations. This paper presents the Phenotype-Gene Relations (PGR) corpus, a silver standard corpus of human phenotype and gene annotations and their relations. The corpus consists of 1712 abstracts, 5676 human phenotype annotations, 13835 gene annotations, and 4283 relations[1]. We generated this corpus using Named-Entity Recognition tools, whose results were partially evaluated by eight curators, obtaining a precision of 87.01%. By using the corpus we were able to obtain promising results with two state-of-the-art deep learning tools, namely 78.05% of precision. The PGR corpus was made publicly available to the research community.[2]

1 Introduction

Automatic extraction of relations between entities mentioned in literature is essential to obtain knowledge that was already available but required considerable manual effort and time to retrieve. Recently, biomedical relation extraction has gained momentum in several text-mining applications, such as event extraction and slot-filling (Lamurias and Couto, 2019b). Some of the commonly extracted biomedical relations are protein-protein interactions (Papanikolaou et al., 2015), drug-drug interactions (Lamurias et al., 2019) and disease-gene relationships (Kim et al., 2017).

There are a few worth mention systems regarding biomedical Relation Extraction (RE) (Verga et al., 2018), and that specifically focus on the extraction of phenotype-gene relations regarding different species types like plants (Xing et al., 2018) and humans (Collier et al., 2015). The main problem that these systems face is a lack of specific high quality annotated corpora (gold standard corpus), mostly because this task requires not only a considerable amount of manual effort but also specific expertise that is not widely available. A solution to these limitations is to generate the corpus in a fully automated manner (silver standard corpus).

Connecting human phenotypes to genes helps us to understand the origin of some phenotypic abnormalities and their associated diseases. To extract human phenotype-gene relations, both entities, human phenotypes and genes have to be recognized. With genes, as a result of lexical features being relatively regular, many systems can successfully identify them in text (Leaman and Gonzalez, 2008). Even though Named-Entity Recognition (NER) research has significantly improved in the last years, human phenotype identification is still a complex task, only tackled by a handful of systems (Lobo et al., 2017).

To generate a silver standard for phenotype-gene relation extraction, we used a pipeline that performs: i) NER to recognize genes and human phenotype entities; ii) RE to classify a relation between human phenotype and gene entities. First, we gathered abstracts using the PubMed API with manually defined keywords, namely each gene name, *homo sapiens*, and *disease*. Then we used the Minimal Named-Entity Recognizer (MER) tool (Couto and Lamurias, 2018) to extract gene mentions in the abstracts and the Identifying Human Phenotypes (IHP) tool (Lobo et al.,

---

*dfsousa@lasige.di.fc.ul.pt

[1] Query 1, corresponds to the *10/12/2018* release of PGR

[2] `https://github.com/lasigeBioTM/PGR`

Proceedings of NAACL-HLT 2019, pages 1487–1492
Minneapolis, Minnesota, June 2 - June 7, 2019.

2017) to extract human phenotype mentions. At last, we used a gold standard relations file, provided by the Human Phenotype Ontology (HPO), to classify the relations obtained by co-occurrence in the same sentence as *Known* or *Unknown*.

To the best of our knowledge, there is no corpus available specific to human phenotype-gene relations. This work, overcame this issue by creating a large and versatile silver standard corpus. To assess the quality of the Phenotype-Gene Relations (PGR) corpus, eight curators manually evaluated a subset of PGR. We obtained highly promising results, for example 87.18% in precision. Finally, we evaluated the impact of using the corpus on two deep learning RE systems, obtaining 69.23% (BO-LSTM) and 78.05% (BioBERT) in precision.

2 PGR Corpus

The HPO is responsible for providing a standardized vocabulary of phenotypic abnormalities encountered in human diseases (Köhler et al., 2017). The developers of the HPO also made available a file that links these phenotypic abnormalities to genes. These phenotype-gene relations are regularly extracted from texts in Online Mendelian Inheritance in Man (OMIM) and Orphanet (ORPHA) databases, where all phenotype terms associated with any disease that is related with a gene are assigned to that gene in the relations file. In this work, we used the relations file created by HPO as a gold standard for human phenotype-gene relations.

We started by retrieving abstracts from PubMed, using the genes involved in phenotype-gene relations and *homo sapiens* as keywords, and the Entrez Programming Utilities (E-utilities) web service (`https://www.ncbi.nlm.nih.gov/books/NBK25501/`), retrieving one abstract per gene (Query 1).

Later, we added the keyword *disease* and filter for abstracts in English (Query 2)[3]. Query 2 represents a more focused search of the type of abstracts to retrieve, such as abstracts regarding diseases, their associated phenotypes and genes.

For each gene, we opted for the most recent abstract (Query 1) and the two most recent abstracts (Query 2).

We opted by searching per gene and not human phenotype or the combination of both terms because this approach was the one that retrieved abstracts with the higher number of gene and human phenotype annotations, in the following NER and RE phases. We removed the abstracts that did not check the conditions of being written in English, with a correct XML format and content. The final number of abstracts was 1712 for Query 1 and 2657 for Query 2 as presented in Table 1. Then we proceeded to use the MER tool (Couto and Lamurias, 2018) for the annotation of the genes and the IHP framework (Lobo et al., 2017) for the annotation of human phenotype terms.

2.1 Gene Extraction

MER is a dictionary-based NER tool which given any lexicon or ontology (e.g., an OWL file) and an input text returns a list of recognized entities, their location, and links to their respective classes.

To annotate genes with MER we need to provide a file of gene names and their respective identifiers. To this goal, we used a list created by the HUGO Gene Nomenclature Committee (HGNC) at the European Bioinformatics Institute (`http://www.genenames.org/`). The HGNC is responsible for approving unique symbols and names for human loci, including protein-coding genes, ncRNA genes, and pseudogenes, with the goal of promoting clear scientific communication. Considering that we intended not only to map the genes to their names but also their Entrez Gene (`www.ncbi.nlm.nih.gov/gene/`) identifiers, we used the API from MyGene (`http://mygene.info/`) with the keyword *human* in species. The MyGene API provides several gene characteristics, including the confidence score for several possible genes that match the query. For this work, we chose the Entrez Gene identifier with a higher confidence score.

After corresponding all gene names to their respective identifiers, we were left with three genes that did not have identifiers (*CXorf36*, *OR4Q2*, and *SCYGR9*). For the first two genes (*CXorf36* and *OR4Q2*), a simple search in Entrez Gene allowed us to match them to their identifiers. For the last gene (*SCYGR9*) we were not able to find an Entrez Gene identifier, so we used the HGNC identifier for that gene instead. We opted to use the Entrez Gene identifiers because of their widespread use in the biomedical research field.

To the original gene list, we added gene synonyms using a synonyms list file provided by

---

[3] Query 2, corresponds to the *11/03/2019* release of PGR

Query	Abstracts	Annotations		Relations		
		Phenotype	Gene	Known	Unknown	Total
1 (10/12/2018)	1712	5676	13835	1510	2773	4283
2 (11/03/2019)	2657	9553	23786	2480	5483	7963

Table 1: The final number of abstracts retrieved, number of phenotype and gene annotations extracted and the number of known, unknown and total of relations extracted between phenotype and genes, for Query 1 and 2.

`https://github.com/macarthur-lab/gene_lists` (expanding the original list almost 3-fold). These synonyms were matched to their identifiers and filtered according to their length to exclude one character length synonyms and avoid a fair amount of false positives. The number of genes in the original gene list was 19194, and by including their synonyms that number increased to 56670, representing a total gain of 37476 genes.

At last, we identified some missed gene annotations that were caught using regular expressions. These missed gene annotations were next to forward/back slash and dashes characters (Example 1).

Example 1. Missed gene annotation because of forward slash.

- **Gene:** *BAX*
- **Gene Identifier:** 581
- **Abstract Identifier:** 30273005
- **Sentence:** According to the morphological observations and DNA fragmentation assay, the MPS compound induced apoptosis in both cell lines, and also cause a significant increase in the expression of **Bax**/Bcl-2.

2.2 Phenotype Extraction

IHP is a Machine Learning-based NER tool, specifically created to recognize HPO entities in unstructured text. It uses Stanford CoreNLP (Manning et al., 2014) for text processing and applies Conditional Random Fields trained with a rich feature set, combined with hand-crafted validation rules and a dictionary to improve the recognition of phenotypes.

To use the IHP system we chose to update the HPO ontology for the most recent version[4]. The annotations that originated from the IHP system were matched to their HPO identifier. There was a total of 7478 annotations for Query 1 and 10973 annotations for Query 2 that did not match any HPO identifier. We put aside these annotations to be confirmed or discarded manually as some of them are incorrectly identified entities but others are parts of adjacent entities that can be combined for an accurate annotation.

We did not use the MER system for phenotype extraction mainly because a more efficient tool for this task was available without the limitations of a dictionary-based NER tool for complex terms as phenotypes are.

2.3 Relation Extraction

After filtering abstracts that did not have annotations of both types, gene and phenotype, we gathered a total of 1712 abstracts for Query 1 and 2656 abstracts for Query 2 as presented in Table 1. The abstracts retrieved by Query 1 were not specific enough for human phenotype-gene relations and therefore about half of them did not contained entities from both types, which we addressed in Query 2, increasing from about 2.5 relations per abstract to about 3.0 relations per abstract.

Using a distant supervision approach, with the HPO file that links phenotypic abnormalities to genes, we were able to classify a relation with *Known* or *Unknown*. For this end, we extract pairs of entities, of gene and human phenotype, by co-occurrence in the same sentence (Example 2). The final number of both *Known* and *Unknown* annotations is also presented in Table 1.

Example 2. Relation extraction.

- **Abstract Identifier:** 23669344
- **Sentence:** A homozygous mutation of **SERPINB6**, a gene encoding an intracellular protease inhibitor, has recently been associated with post-lingual, autosomal-recessive, nonsyndromic **hearing loss** in humans (DFNB91).
- **Gene:** *SERPINB6*
- **Gene Identifier:** 5269
- **Phenotype:** *hearing loss*
- **Phenotype Identifier:** HP_0000365
- **Relation: Known**

---

[4] *09/10/2018* release

3 Evaluation

To evaluate the quality of the classifier, we randomly selected 260 relations from Query 1 to be reviewed by eight curators (50 relations each, with an overlap of 20 relations). All researchers work in the areas of Biology and Biochemistry. These curators had to evaluate the correctness of the classifier by attributing to each sentence one of the following options: *C* (correct), *I* (incorrect) or *U* (uncertain). The *U* option was given to identify cases of ambiguity and possible errors in the NER phase. We classified as a true positive (TP) a *Known* relation that was marked *C* by the curator, a false positive (FP) as a *Known* relation marked *I*, a false negative (FN) as a *Unknown* relation marked *I* and a true negative (TN) as a *Unknown* relation marked *C*.

3.1 State-of-the-art Applications

The PGR corpus was applied to two state-of-the-art systems that were compared against a co-occurrence (or all-true) baseline method.

3.1.1 BO-LSTM Application

The BO-LSTM system (Lamurias et al., 2019) is a deep learning system that is used to extract and classify relations via long short-term memory networks along biomedical ontologies. This system was initially created to detect and classify drug-drug interactions and later adapted to detect other types of relations between entities like human phenotype-gene relations. It takes advantage of domain-specific ontologies, like the HPO and the Gene Ontology (GO) (Ashburner et al., 2000). The BO-LSTM system represents each entity as the sequence of its ancestors in their respective ontology.

3.1.2 BioBERT Application

The BioBERT system (Lee et al., 2019) is a pre-trained biomedical language representation model for biomedical text mining based on the BERT (Devlin et al., 2018) architecture. Trained on large-scale biomedical corpora, this system is able to perform diverse biomedical text mining tasks, namely NER, RE and Question Answering (QA). The novelty of the architecture is that these systems (BioBERT and BERT) are designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. These feature allows easy adaption to several tasks without loss in performance.

4 Results and Discussion

The final results are presented in Table 2. The inter-curator agreement score, calculated from a total of 20 relations classified by eight curators, was 87.58%. Besides the fact that there were a few incorrectly extracted relations due to errors in the NER phase, that were discarded, the inter-curator agreement is not higher due to the complexity of the sentences where the relations between entities were identified. Even with highly knowledgeable curators in the fields of Biology and Biochemistry, most of them expressed difficulties in deciding which mark to choose on complex sentences that did not necessarily imply a relation between the identified entities (Example 3).

Example 3. Relation marked with *U* (Uncertain).

- **Abstract Identifier:** 27666346
- **Sentence: FRMD4A** antibodies were used to probe 78 paraffin-embedded specimens of **tongue squamous cell carcinoma** and 15 normal tongue tissues, which served as controls.
- **Mark:** ***U***

The precision obtained from the test-set (about 6% of the total of relations), was 87.01%. Although we cannot state that this test-set is representative of the overall data-set, this is a solid evidence of the effectiveness of our pipeline to automate RE corpus creation, especially between human phenotype and genes, and other domains if a gold standard relations file is provided. Our lower recall is mostly due to incorrectly retrieved human phenotype annotations by IHP, that can be manually confirmed in a future optimized version of the PGR corpus, as some of them are parts of adjacent entities that can be combined for an accurate annotation.

4.1 Impact on Deep Learning

For BioBERT we used the available pre-trained weights for training and testing of RE model on our corpus. The results of BO-LSTM and BioBERT in the test-set are presented in Table 3. We also measured the performance of the co-occurrence (i.e. assuming all-true) baseline method. This baseline method assumes that all relations in the test-set are *Known* and therefore the recall is 100%. These results are comparable to

Relations		Marked Relations				Metrics		
Known	Unknown	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F-Measure
77	143	67	86	10	57	0.8701	0.4379	0.5826

Table 2: The *Known* and *Unknown* number of relations selected, the number of true positives, false negatives, false positives and true negatives, and the evaluation metrics for the *Known* relations.

Method	Precision	Recall	F-Measure
Co-occurrence	0.3500	1.0000	0.5185
BO-LSTM	0.6923	0.4200	0.5228
BioBERT	0.7895	0.5844	0.6716

Table 3: Precision, recall and F-measure of the co-occurrence baseline, BO-LSTM and BioBERT.

the ones obtained from the evaluation stage by the curators, and show the applicability of our corpus.

BioBERT significantly outperforms BO-LSTM in all metrics proving that is indeed a viable language representation model for biomedical text mining. Even though the recall for both systems is relatively low, the purpose of this work was mainly to extract correct relations between entities to facilitate Machine Learning (ML), which was achieved by obtaining the precision of 69.23% (BO-LSTM) and 78.95% (BioBERT).

The most relevant metric for a silver standard corpus, directed towards ML tools, is precision. ML tools depend on correct examples to create effective models that can detect new cases, afterwards, being able to deal with small amounts of noise in the assigned labels.

5 Conclusions

This paper showed that our pipeline is a feasible way of generating a silver standard human phenotype-gene relation corpus. The pipeline required the application of two NER tools, and the availability of a list of known relations. We manually evaluated the corpus using eight curators obtaining a 87.01% precision with an inter-agreement of 87.58%. We also measured the impact of using the corpus in state-of-the-art deep learning RE systems, namely BO-LSTM and BioBERT. The results were promising with 69.23%, and 78.95% in precision, respectively. We believe that our pipeline and silver standard corpus will be a highly useful contribution to overcome the lack of gold standards.

Future work includes manually correcting the human phenotype annotations that did not match any HPO identifier, with the potential of expanding the number of human phenotype annotations almost 2-fold and increasing the overall recall. Also, we intend to expand the corpus by identifying more missed gene annotations using pattern matching, which is possible due to our approach being fully automated. Another possibility is the expansion of the test-set for a more accurate capture of the variance in the corpus. For example, we can select a subset of annotated documents in which two curators could work to grasp the complexity of manually annotating some of these abstracts. Further, we intend to use semantic similarity to validate the human phenotype-gene relations. Semantic similarity has been used to compare different types of biomedical entities (Lamurias and Couto, 2019a), and is a measure of closeness based on their biological role. For example, if the *BRCA1* gene is semantically similar to the *BRAF* gene and the *BRCA1* has an established relation with the *tumor* phenotype, it could be possible to infer that *BRAF* gene also has a relation with the *tumor* phenotype, even if that is not evident by the training data. Finally, the effect of different NER systems applied to the pipeline should be studied.

Acknowledgments

We acknowledge the help of Márcia Barros, Joana Matos, Rita Sousa, Ana Margarida Vasconcelos, Maria Teresa Cunha and Sofia Jesus in the curating phase.

This work was supported by FCT through funding of DeST: Deep Semantic Tagger project, ref. PTDC/CCI-BIO/28685/2017 (`http://dest.rd.ciencias.ulisboa.pt/`), and LASIGE Research Unit, ref. UID/CEC/00408/2019.

References

Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan Peter Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29.

Nigel Collier, Tudor Groza, Damian Smedley, Peter N. Robinson, Anika Oellrich, and Dietrich Rebholz-Schuhmann. 2015. Phenominer: from text to a database of phenotypes associated with OMIM diseases. *Database*, 2015:bav104.

Francisco M Couto and Andre Lamurias. 2018. MER: a shell script and annotation server for minimal named entity recognition and linking. *Journal of Cheminformatics*, 10(58):1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeongkyun Kim, Jung-jae Kim, and Hyunju Lee. 2017. An analysis of disease-gene relationship from Medline abstracts by DigSee. *Scientific Reports*, 7:40154.

Sebastian Köhler, Nicole Vasilevsky, Mark Engelstad, Erin Foster, et al. 2017. The human phenotype ontology. *2017 Nucleic Acids Research*.

Andre Lamurias and Francisco M Couto. 2019a. Semantic similarity definition. In *Encyclopedia of Bioinformatics and Computtational Biology*, volume 1, pages 870–876. Oxford: Elsevier.

Andre Lamurias and Francisco M Couto. 2019b. Text mining for bioinformatics using biomedical literature. In *Encyclopedia of Bioinformatics and Computtational Biology*, volume 1, pages 602–611. Oxford: Elsevier.

Andre Lamurias, Diana Sousa, Luka A. Clarke, and Francisco M. Couto. 2019. BO-LSTM: classifying relations via long short-term memory networks along biomedical ontologies. *BMC Bioinformatics*, 20(1):10.

Robert Leaman and Graciela Gonzalez. 2008. BANNER: an executable survey of advances in biomedical named entity recognition. *Pacific Symposium on Biocomputing*, pages 652–663.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *arXiv e-prints*, page arXiv:1901.08746.

Manuel Lobo, Andre Lamurias, and Francisco M Couto. 2017. Identifying human phenotype terms by combining machine learning and validation rules. *BioMed Research International*, 7.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL*.

Nikolas Papanikolaou, Georgios A. Pavlopoulos, Theodosios Theodosiou, and Ioannis Iliopoulos. 2015. Protein-protein interaction predictions using text mining methods. *Methods*, 74:47–53.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 872–884. Association for Computational Linguistics.

Wenhui Xing, Junsheng Qi, Xiaohui Yuan, Lin Li, Xiaoyu Zhang, Yuhua Fu, Shengwu Xiong, Lun Hu, and Jing Peng. 2018. A gene-phenotype relationship extraction pipeline from the biomedical literature using a representation learning approach. *Bioinformatics*, 34(13):i386–i394.

Improving Lemmatization of Non-Standard Languages with Joint Learning

Enrique Manjavacas[1], **Ákos Kádár**[2], and **Mike Kestemont**[1]

[1]Computational Linguistics and Psycholinguistics Research Center, University of Antwerp, Belgium
[2]Tilburg Center for Cognition and Communication, Tilburg University, The Netherlands
{firstname,lastname}@uantwerpen.be,a.kadar@uvt.nl

Abstract

Lemmatization of standard languages is concerned with (i) abstracting over morphological differences and (ii) resolving token-lemma ambiguities of inflected words in order to map them to a dictionary headword. In the present paper we aim to improve lemmatization performance on a set of non-standard historical languages in which the difficulty is increased by an additional aspect (iii): spelling variation due to lacking orthographic standards. We approach lemmatization as a string-transduction task with an encoder-decoder architecture which we enrich with sentence context information using a hierarchical sentence encoder. We show significant improvements over the state-of-the-art when training the sentence encoder jointly for lemmatization and language modeling. Crucially, our architecture does not require POS or morphological annotations, which are not always available for historical corpora. Additionally, we also test the proposed model on a set of typologically diverse standard languages showing results on par or better than a model without enhanced sentence representations and previous state-of-the-art systems. Finally, to encourage future work on processing of non-standard varieties, we release the dataset of non-standard languages underlying the present study, based on openly accessible sources.

1 Introduction

Lemmatization is the task of mapping a token to its corresponding dictionary head-form to allow downstream applications to abstract away from orthographic and inflectional variation (Knowles and Mohd Don, 2004). While lemmatization is considered to be solved for analytic and resource-rich languages such as English, it remains an open challenge for morphologically complex (e.g. Estonian, Latvian) and low-resource languages with unstable orthography (e.g. historical languages). Especially for languages with higher surface variation, lemmatization plays a crucial role as a preprocessing step for downstream tasks such as topic modeling, stylometry and information retrieval.

In the case of standard languages, lemmatization complexity arises primarily from two sources: (i) morphological complexity affecting the number of inflectional patterns a lemmatizer has to model and (ii) token-lemma ambiguities (e.g. "living" can refer to lemmas "living" or "live") which require modeling sentence context information. In the case of historical languages, however, the aforementioned spelling variation introduces further complications. For instance, the regularity of the morphological system is drastically reduced since the evidence supporting token-lemma mappings becomes more sparse. As an example, while the modern Dutch lemma "jaar" (en. "year") can be inflected in 2 different ways ("jaar", "jaren"), in a Middle Dutch corpus used in this study it is found in combination with 70 different forms ("iare", "ior", "jaer", etc.). Moreover, spelling variation increases token-lemma ambiguities by conflating surface realizations of otherwise unambiguous tokens—e.g. Middle Low German "bath" can refer to lemmas "bat" (en. "bad") and "bidden" (en. "bet") due to different spellings of the dental occlusive in final position.

Spelling variation is not exclusive of historical languages and it can be found in contemporary forms of on communication, such as microblogs, with loose orthographic conventions (Crystal, 2001). An important difference, however, is that while for modern languages normalization is feasible (Schulz et al., 2016), for many historic languages such is not possible, because one is dealing with an amalgam of regional dialects that lacked any sort of supra-regional variant functioning as target domain (Kestemont et al., 2016).

In the present paper, we apply representation

Proceedings of NAACL-HLT 2019, pages 1493–1503
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

learning to lemmatization of historical languages. Our method shows improvements over a plain encoder-decoder framework, which reportedly achieves state-of-the-art performance on lemmatization and morphological analysis (Bergmanis and Goldwater, 2018; Peters et al., 2018). In particular, we make the following contributions:

1. We introduce a simple joint learning approach based on a bidirectional Language Model (LM) loss and achieve relative improvements in overall accuracy of 7.9% over an encoder-decoder trained without joint loss and 30.72% over edit-tree based approaches.
2. We provide a detailed analysis of the linguistic and corpus characteristics that explain the amount of improvement we can expect from LM joint training.
3. We probe the hidden representations learned with the joint loss and find them significantly better predictors of POS-tags and other morphological categories than the representations of the simple model, confirming the efficiency of the joint loss for feature extraction.

Additionally, we test our approach on a typologically varied set of modern standard languages and find that the joint LM loss significantly improves lemmatization accuracy of ambiguous tokens over the encoder-decoder baseline (with a relative increase of 15.1%), but that, in contrast to previous literature (Chakrabarty et al., 2017; Bergmanis and Goldwater, 2018), the overall performance of encoder-decoder models is not significantly higher than that of edit-tree based approaches. Taking into account the type of inflectional morphology dominating in a particular language, we show that the benefit of encoder-decoder approaches is highly dependent on typological morphology. Finally, to assure reproducibility, all corpus preprocessing pipelines and train-dev-test splits are released. With this release, we hope to encourage future work on processing of lesser studied non-standard varieties.[1]

2 Related Work

Modern data-driven approaches typically treat lemmatization as a classification task where classes are represented by binary edit-trees induced from the training data. Given a token-lemma pair, its binary edit-tree is induced by computing the prefix and suffix around the longest common subsequence, and recursively building a tree until no common character can be found. Such edit-trees manage to capture a large proportion of the morphological regularity, especially for languages that rely on suffixation for morphological inflection (e.g. Western European languages), for which such methods were primarily designed.

Based on edit-tree induction, different lemmatizers have been proposed. For example, Chrupala et al. (2008) use a log-linear model and a set of hand-crafted features to decode a sequence of edit-trees together with the sequence of POS-tags using a beam-search strategy. A related approach is presented by Gesmundo and Samardži (2012), where edit-trees are extracted using a non-recursive version of the binary edit-tree induction approach. More recently, Cotterell et al. (2015) have used an extended set of features and a second-order CRF to jointly predict POS-tags and edit-trees with state-of-the-art performance. Finally, Chakrabarty et al. (2017) employed a softmax classifier to predict edit-trees based on sentence-level features implicitly learned with a neural encoder over the input sentence.

With the advent of current encoder-decoder architectures, lemmatization as a string-transduction task has gained interest partly inspired by the success of such architectures in Neural Machine Translation (NMT). For instance, Bergmanis and Goldwater (2018) apply a state-of-the-art NMT system with the lemma as target and as source the focus token with a fixed window over neighboring tokens. Most similar to our work is the approach by Kondratyuk et al. (2018), which conditions the decoder on sentence-level distributional features extracted from a sentence-level bidirectional RNN and morphological tags.

Recently, work on non-standard historical varieties has focused on spelling normalization using rule-based, statistical and neural string-transduction models (Pettersson et al., 2014; Bollmann and Søgaard, 2016; Tang et al., 2018). Previous studies on lemmatization of historical variants focused on evaluating off-the-shelf systems.

---

[1] Datasets and training splits are available at `https://www.github.com/emanjavacas/pie-data`. Experiments are conducted with our framework `pie` available at: `https://www.github.com/emanjavacas/pie`. All our experiments are implemented using `PyTorch` (Paszke et al., 2017).

For instance, Eger et al. (2016) evaluates different pre-existing models on a dataset of German and Medieval Latin, and Dereza (2018) focuses on Early Irish. The most similar to the present paper in this area is work by Kestemont et al. (2016), which tackled lemmatization of Middle Dutch with a neural encoder that extracts character and word-level features from a fixed-length token window and predicts the target lemma from a closed-set of true lemmas.

Using Language Modeling as a task to extract features in a Transfer Learning setup has gained momentum only in the last year, partly thanks to overall improvements over previous state-of-the-art across multiple tasks (NER, POS, QA, etc.). Different models have been proposed around the same idea varying in implementation, optimization and task definition. For instance, Howard and Ruder (2018) present a method to fine-tune a pretrained LM for text classification. Peters et al. (2018) learn task-specific weighting schemes over different layer features extracted by a pretrained bidirectional LM. Recently, Akbik et al. (2018) used context-sensitive word-embeddings extracted from a bidirectional character-level LM to improve NER, POS-tagging and chunking.

3 Proposed Model

Here we describe our encoder-decoder architecture for lemmatization. In Section 3.1 we start by describing the basic formulation known from the machine translation literature. Section 3.2 shows how sentential context is integrated into the decoding process as an extra source of information. Finally, Section 3.3 describes how we learn richer representations for the encoder through the addition of an extra language modeling task.

3.1 Encoder-Decoder

We employ a character-level Encoder-Decoder architecture that takes an input token x_t character-by-character and has as goal the character-level decoding of the target lemma l_t conditioned on an intermediate representation of x_t. For token x_t, a sequence of token character embeddings $c^x_1, \ldots, c^x_n$ is extracted from embedding matrix $W_{enc} \in \mathbb{R}^{|C| \times d}$ (where $|C|$ and d represent, respectively, the size of the character vocabulary and the embedding dimensionality). These are then passed to a bidirectional RNN encoder, that computes a forward and a backward sequence of hidden states: $\overrightarrow{h^{enc}_1}, \ldots, \overrightarrow{h^{enc}_n}$ and $\overleftarrow{h^{enc}_1}, \ldots, \overleftarrow{h^{enc}_n}$. The final representation of each character i is the concatenation of the forward and the backward states: $h^{enc}_i = [\overrightarrow{h^{enc}_i}; \overleftarrow{h^{enc}_i}]$.

At each decoding step j, a RNN decoder generates the hidden state h^{dec}_j, given the lemma character embedding c^l_j from embedding matrix $W_{dec} \in \mathbb{R}^{|L| \times d}$, the previous hidden state h^{dec}_{j-1} and additional context. This additional context consists of a summary vector r_j obtained via an attentional mechanism (Bahdanau et al., 2014) that takes as input the previous decoder state h^{dec}_{j-1} and the sequence of encoder activations $h^{enc}_1, \ldots, h^{enc}_n$.[2] Finally, the output logits for character j are computed by a linear projection of the current decoder state h^{enc}_j with parameters $O \in \mathbb{R}^{H \times |L|}$, which are normalized to probabilities with the $softmax$ function. The model is trained to maximize the probability of the target character sequence expressed in Equation 1 using teacher-forcing.

$$P(l_t|x_t) = \prod_{j=1}^{m} P(c^l_j | c^l_{<j}, r_j; \theta_{enc}, \theta_{dec}) \qquad (1)$$

3.2 Adding sentential context

Lemmatization of ambiguous tokens can be improved by incorporating sentence-level information. Our architecture is similar to Kondratyuk et al. (2018) in that it incorporates global sentence information by extracting distributional features with a hierarchical bidirectional RNN over the input sequence of tokens $x_1, ..., x_m$. For each token x_t, we first extract word-level features re-using the last hidden state of character-level bidirectional RNN Encoder from Section 3.1 $w_t = [\overrightarrow{h^{enc}_t}; \overleftarrow{h^{enc}_t}]$. Optionally, word-level features can be enriched with extra lookup parameters from an embedding matrix $W_{word} \in \mathbb{R}^{|V| \times e}$ – where V and e denote respectively the vocabulary size in words and the word embedding dimensionality.[3] Given these word-level features w_t, the sentence-level features s_t are computed as the concatenation of

---

[2] We refer to Bahdanau et al. (2014) for the description of the attentional mechanism.

[3] During development word embeddings did not contribute significant improvements on historical languages, and we therefore exclude them from the rest of the experiments. It must be noted, however, that word embeddings might still be helpful for lemmatization of standard languages where the type-token ratio is smaller as well as when pretrained embeddings are available.

forward and backward activations of an additional sentence-level bidirectional RNN $s_t = [\overrightarrow{s_t}; \overleftarrow{s_t}]$.

In order to perform sentence-aware lemmatization for token x_t, we condition the decoder on the sentence-level encoding s_t and optimize the probability given by Equation 2.

$$P(l_t|x_t) = \prod_{j=1}^{m} P(c_j^l|c_{<j}^l, r_j, s_t; \theta_{enc}, \theta_{dec}) \qquad (2)$$

Our architecture ensures that both word-level and character-level features of each input token in a sentence can contribute to the sentence-level features at any given step and therefore to the lemmatization of any other token in the sentence. From this perspective, our architecture is more general than those presented in Kestemont et al. (2016); Bergmanis and Goldwater (2018), where sentence information is included by running the encoder over a predetermined fixed-length window of neighboring characters. Moreover, we let the character-level embedding extractor and the lemmatizer encoder share parameters in order to amplify the training signal coming into the latter. Figure 1 visualizes the proposed architecture.

3.3 Improved sentence-level features

We hypothesize that the training signal from lemmatization alone might not be enough to extract sufficiently high quality sentence-level features. As such we include an additional bidirectional word-level language-model loss over the input sentence. Given the forward and backward subvectors of the sentence encoding $s^t = [\overrightarrow{s^t}; \overleftarrow{s^t}]$, we train two additional softmax classifiers to predict token x^{t+1} given $\overrightarrow{s_t}$ and x^{t-1} given $\overleftarrow{s_t}$ with parameters O_{LMfwd} and $O_{LMbwd} \in \mathbb{R}^{S\times|V|}$.[4]

We train our model to jointly minimize the negative log-likelihood of the probability defined by Equation 2 and the LM probability defined by Equation 3.

$$\begin{aligned} P_{LM}(\mathbf{x}) = 1/2 \prod_{t=2}^{n} P(x_t|x_1, \ldots, x_{t-1}) \\ + 1/2 \prod_{t=1}^{n-1} P(x_t|x_{t+1}, \ldots, x_n) \end{aligned} \qquad (3)$$

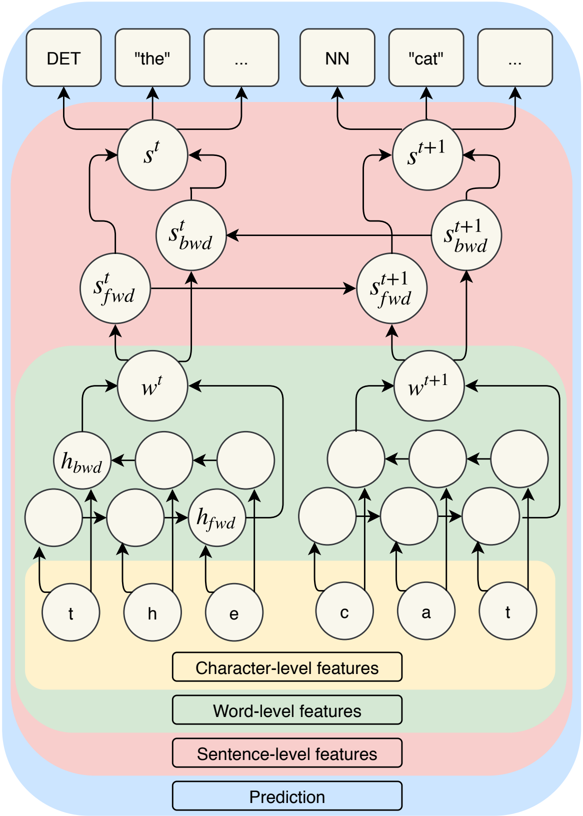


Figure 1: Hierarchical sentence encoding architecture with feature extraction at different levels.

Following a Multi-Task Learning (Caruana, 1997), we set a weight on the LM negative log-likelihood which we decrease over training based on lemmatization accuracy on development data to reduce its influence on training after convergence.

4 Experiments

Section 4.1 first introduces the datasets, both the newly introduced dataset of historical languages, and the dataset of modern standard languages sampled from Universal Dependencies (v2.2) corpus (Nivre et al., 2016). Finally, Section 4.2 describes model training and settings in detail.

4.1 Datasets

Historical Languages In recent years, a number of historical corpora have appeared thanks to an increasing number of digitization initiatives (Piotrowski, 2012). For the present study, we chose a representative collection of medieval and early modern datasets, favoring publicly available data, corpora with previously published results and datasets covering multiple genres and historic periods. We include a total of 8 corpora covering Middle Dutch, Middle Low German, Medieval

---

[4] We have found the joint loss most effective when both forward and backward classifiers shared parameters.

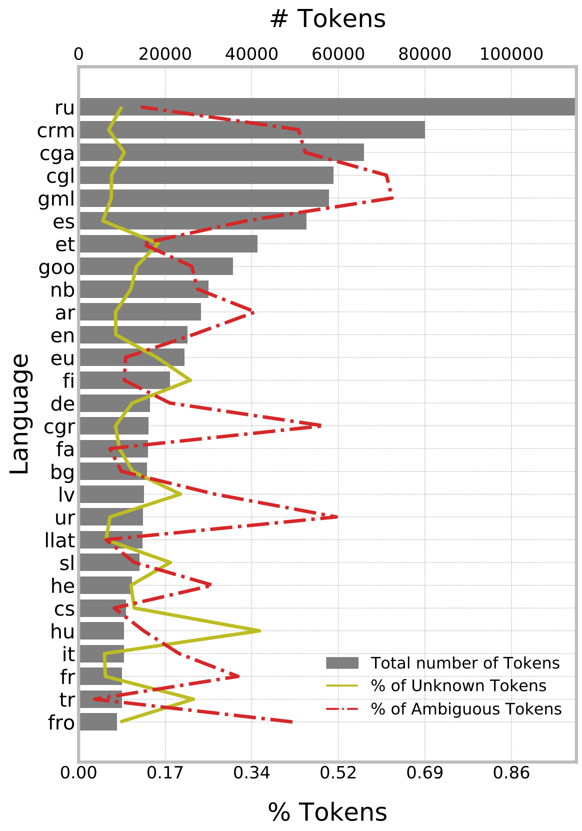


Figure 2: Statistics of total number of tokens, ambiguous and unknown tokens in the test sets. The full list of languages for both historical and standard corpora as well as the corresponding ISO 639-1 codes used in the present study can be found in the Appendix. Statistics for unknown and ambiguous tokens are shown as percentages.

French, Historical Slovene and Medieval Latin, which we take from the following sources.

Both `cga` and `cgl` contain medieval Dutch material from the Gysseling corpus curated by the Institute for Dutch Lexicology[5] `cga` is a charter collection (administrative documents), whereas `cgl` concerns a variety of literary texts that greatly vary in length. `crm` is another Middle Dutch charter collection from the 14th century with wide geographic coverage (Van Reenen and Mulder, 1993; van Halteren and Rem, 2013). `cgr`, finally, is a smaller collection of samples from Middle Dutch religious writings that include later medieval texts (Kestemont et al., 2016). `fro` offers a corpus of Old French heroic epics, known as *chansons de geste* (Camps, 2016). `llat` dataset is taken from the Late Latin Charter Treebank, consisting of early medieval Latin documentary texts (Korkiakangas and Lassila, 2013). `goo` comes from the reference corpus of historical Slovene, sampled from 89 texts from the period 1584-1899 (Erjavec, 2015). `gml` refers to the reference corpus of Middle Low German and Low Rhenish texts, found in manuscripts, prints and inscriptions (Barteld et al., 2017). Finally, `cap` is a corpus of early medieval Latin ordinances decreed by Carolingian rulers (Eger et al., 2016).

Standard Languages For a more thorough comparison between systems across domains and a better examination of the effect of the LM loss, we evaluate our systems on a set of 20 standard languages sampled from the UD corpus, trying to guarantee typological diversity while selecting datasets with at least 20k words. We use the pre-defined splits from the original UD corpus (v2.2).[6]. Figure 2 visualizes the test set sizes in terms of total, ambiguous and unknown tokens for both historical and standard languages.

4.2 Models

We refer to the full model trained with joint LM loss by `Sent-LM`. In order to test the effectiveness of sentence information and the importance of enhancing the quality of the sentence-level feature extraction, we compare against a simple encoder-decoder model without sentence-level information (`Plain`) and a model trained without joint LM loss (`Sent`). Moreover, we compare to previous state-of-the-art lemmatizers based on binary edit-tree induction: `Morfette` (Chrupala et al., 2008) and `Lemming` (Cotterell et al., 2015), which we run with default hyperparameters.

For all our models, we use the same hyperparameter values as follows. All recurrent layers have 150 cells per layer and use GRUs (Cho et al., 2014). Encoder and Decoder have 2 layers but the sentence encoder has only 1. We apply 0.25 dropout (Srivastava et al., 2014) after the embedding layer and before the output layer and 0.25 variational dropout (Gal and Ghahramani, 2016) in between recurrent layers. Models are optimized with Adam (Kingma and Ba, 2015) using an initial learning rate of 1e-3 which is reduced by 25% after each epoch without improvement on development accuracy. Models are trained until fail-

---

[5] `https://ivdnt.org/taalmaterialen`.

[6] The full list of languages for both historical and standard corpora as well as the corresponding ISO 639-1 codes used in the present study can be found in the Appendix. In the cases where train-dev-test splits were not pre-defined, we randomly split sentences using 10% and 5% for test and dev respectively.

	Full	Ambiguous	Unknown
`Edit-tree`	91.11	91.79	35.48
`Plain`	91.61	87.39	65.69
`Sent`	93.4	91.14	**66.98**
`Sent-LM`	**94.0**	**92.81**	65.39

Table 1: Average accuracy across historical languages. `Lemming` and `Morfette` are shown aggregated by taking the best performing model per dataset.

	Full	Ambiguous	Unknown
`K-2016`	91.88		51.64
`Edit-tree`	89.01	91.18	20.46
`Plain`	90.21	87.4	61.93
`Sent`	92.55	91.6	**64.61**
`Sent-LM`	**93.25**	**93.31**	62.1

Table 2: Accuracy for the Gysseling subcorpora.

ing to achieve any improvement for 3 consecutive epochs. Initial LM loss weight is set to 0.2 and it is halved each epoch after two consecutive epochs without achieving any improvements on development perplexity.

We use sentence boundaries when given and otherwise use POS tags corresponding to full stops as clues. In any case, sentences are split into chunks of maximum 35 words to accommodate to limited memory. Target lemmas during both training and testing are lowercased in agreement with the implementation of `Lemming` and `Morfette`, which also do so. For models with joint loss, we truncate the output vocabulary to the top 50k most frequent words for similar reasons. We run a maximum of 100 optimization epochs in randomized batches containing 25 sentences each. The learning rate is decreased by a factor of 0.75, after every 2 epochs without accuracy increase on held-out data and learning stops after failing to improve for 5 epochs. Decoding is done with beam search with a beam size of 10.[7]

5 Results

As is customary, we report exact-match accuracy on target lemmas. Besides overall accuracy, we also compute accuracy of ambiguous tokens (i.e. tokens that map to more than 1 lemma in the training data) and unknown tokens (i.e. tokens that do not appear in the training data).

5.1 Historical languages

Table 1 shows the aggregated results over all datasets in our historical language corpus.[8] In 4 cases (`cga`, `cgl`, `crm` and `gml`), `Lemming` failed to converge due to memory requirements exceeding 250G RAM due to the large amount of edit-trees. Following Søgaard et al. (2014), we compute p-values with a Wilcoxon's signed rank test. `Sent-LM` is the best performing model with a relative improvement of 7.9% ($p < .01$) over `Sent` and 30.72% ($p < .01$) over the edit-tree approach on full datasets and 10.27% ($p < .1$) and 18.66% ($p < .01$) on ambiguous tokens. Moreover, the edit-tree approach outperforms encoder-decoder models `Plain` and `Sent` on ambiguous tokens, and it is only due to the joint loss that the encoder-decoder paradigm gains an advantage. Finally, for tokens unseen during training, the best performing model is `Sent` with a relative error reduction of 47% ($p < .01$) over the edit-tree approach and 4.77% ($p < .1$) over `Sent-LM`.

Table 2 compares scores for a subset from the corpora coming from the Gysseling corpus, which have been used in previous work on lemmatization of historical languages. The model described by Kestemont et al. (2016) is included as `K-2016` for comparison.[9] It is apparent that both `Sent` and `Sent-LM` outperform `K-2016` on full and unknown tokens. It is worth noting that `K-2016`, a model that uses distributed contextual features but no edit-tree induction, performs better than `Plain` – which highlights the importance of context for the lemmatization of historical languages –, and also better than the edit-tree approaches – which highlights the difficulty of tree induction on this dataset. We find `Sent-LM` to have a significant advantage over `Sent` on full and ambiguous tokens, but a disadvantage vs `Sent` and `Plain` on unknown tokens.

5.2 Standard languages

Table 4 shows overall accuracy scores aggregated across all languages.[10] We observe that on average `Sent-LM` is the best model on full datasets.

---

[7] For all languages, we observed relatively small gains ranging from 0.1% to 0.5% in overall accuracy.

[8] We aggregate both edit-tree based approaches by selecting the best performing model for each corpus. When `Lemming` converge, the results were better than `Morfette`.

[9] Unfortunately, scores on ambiguous tokens were not reported and therefore cannot be compared.

[10] Similarly to results on historical languages, we aggregate `Morfette` and `Lemming` due to the later failing to converge on `et`.

	Full			Ambiguous			Unknown		
	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
`Edit-tree`	96.34	93.02	**98.37**	92.47	**92.47**	**97.5**	84.99	74.66	**91.39**
`Plain`	96.21	94.6	97.5	88.87	90.51	95.34	85.43	**83.78**	86.37
`Sent`	96.42	94.41	97.84	91.12	92.01	96.65	**85.44**	84.02	86.67
`Sent-LM`	**96.52**	**94.62**	97.86	**93.01**	92.07	97.48	85.15	83.32	85.36

Table 3: Average accuracy across morphologically related standard languages. Type 1 encloses `bg`, `cs`, `lv`, `ru` and `sl`. Type 2 comprises `et`, `fi`, `hu`, `tr`. Finally, Type 3 encompasses `de`, `en`, `es`, `fr`, `it` and `nb`.

	Full	Ambiguous	Unknown
`Edit-tree`	96.1	**94.35**	**83.26**
`Plain`	95.93	91.44	83.02
`Sent`	96.19	93.25	82.61
`Sent-LM`	**96.28**	94.08	82.58

Table 4: Average accuracy across all 20 standard languages. `Lemming` and `Morfette` are shown aggregated by taking the best performing model per dataset.

However, in contrast to previous results, the edit-tree approach has an advantage over all encoder-decoder models for both ambiguous and unknown tokens.

Since the differences in performance are not statistically significant ($p > 0.05$), we seek to shed light on the advantages and disadvantages of the encoder-decoder and edit-tree paradigms by conducting a more fine-grained analysis with respect to the morphological typology of the considered languages. To this end, we group languages into morphological types depending on the dominant morphological processes of each language and aggregate scores over languages in each type:

Type 1. Balto-Slavic languages which are known for their strongly suffixing morphology and complex case system.

Type 2. Uralic and Altaic languages, which are characterized by agglutinative morphology and a tendency towards monoexponential case and vowel harmony.

Type 3. Western European languages with a tendency towards synthetic morphology and partially lacking nominal case.

Table 3 shows accuracy scores per morphological group for each model type. It is apparent that the `Edit-tree` approach is very effective for Type 3 languages both in ambiguous and unknown tokens. In both Type 1 and Type 2 languages, the best overall performing model is `Sent-LM`. In the case of ambiguous tokens, `Sent-LM` achieves highest accuracy for Type 1 languages, but it is surpassed by the `Edit-tree` approach on Type 2 languages. Finally, in the case of unknown tokens, we observe a similar pattern to the historical languages where `Plain` and `Sent` have an advantage over `Sent-LM`.

6 Discussion

For clarity, we group the discussion of the main findings according to four major discussion points.

How does the joint LM loss help? As Section 5 shows, `Sent-LM` is the overall best model, and its advantage is biggest on ambiguous datasets, always outperforming the second-best encoder-decoder model on ambiguous tokens. For a more detailed comparison of the two models we tested the following two hypotheses: (i) the joint LM loss helps by providing sentence representations with stronger disambiguation capacities (ii) The joint LM loss helps in cases when the evidence of a token-lemma relationship is sparse —e.g in languages with highly synthetic morphological systems and in the presence of spelling variation.

As Figure 3 shows, improvement over `Sent` is correlated with percentage of token-lemma ambiguity in the corpus, providing evidence for hypothesis (i). Finally, as Figure 4 shows, improvement over `Sent` is correlated with higher token-lemma ratio, suggesting that the improvement is likely to be due to learned representations that better identify the input token. These two aspects help explain the efficiency of the joint learning approach on non-standard languages where high levels of spelling variation provide increased ambiguity by conflating unrelated forms and also lower evidence for token-lemma mappings.

Another factor certainly related to the efficiency of the proposed joint LM-loss is the size of the training dataset. However, dataset size should be considered a necessary but not a sufficient condition for the feasibility of the joint LM-loss and has therefore weak explanation power for the performance of the proposed approach.

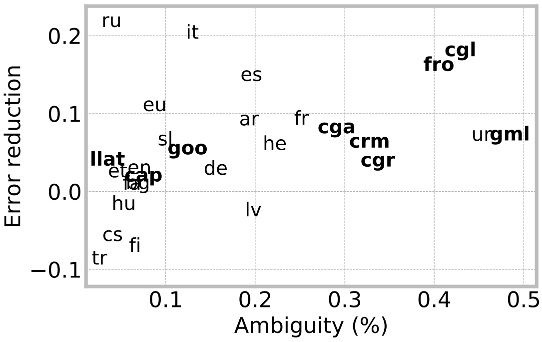


Figure 3: Error reduction of `Sent-LM` vs `Sent` by percentage of ambiguous tokens (Spearman's $R = 0.53; p < .01$). Historical languages are shown in bold.

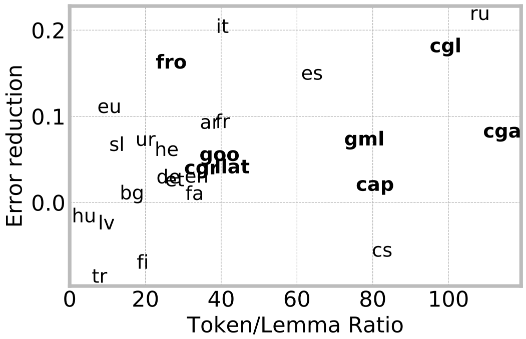


Figure 4: Error reduction of `Sent-LM` vs `Sent` by Token-Lemma ratio on 50k tokens of the training set (Spearman's $R = 0.47; p < .05$). Historical languages are shown in bold.

LM loss leads to better representations In order to analyze the representations learned with the joint loss, we turn to "representation probing" experiments following current approaches on interpretability (Linzen et al., 2016; Adi et al., 2017). Using the same train-dev-test splits from the current study, we exploit additional POS, Number, Gender, Case and syntactic function (Dep) annotations provided in the UD corpora and compare the ability of the representations extracted by `Sent` and `Sent-LM` to predict these labels.[11] Model parameters are frozen and a linear softmax layer $Q \in \mathbb{R}^{H\times V}$ per task is learned using a cross-entropy loss function.[12] The results of this experiment are reported in Table 5. The classifier trained with `Sent-LM` outperforms the one with `Sent` on all considered labeling tasks, confirming the efficiency of the LM loss at extracting better representations.

	Pos	Dep	Gender	Case	Num
`Majority`	82.61	62.93	85.83	80.76	83.92
`Sent`	79.27	64.14	83.39	83.01	81.01
`Sent-LM`	**83.62**	**68.36**	**86.55**	**84.44**	**84.37**
Support	29	20	15	19	20

Table 5: Overall accuracy of `Sent` and `Sent-LM` models and a `Majority` baseline on 5 probing tasks and actual number of languages per morphological category. All differences over `Sent` except for Case are significant at $p < 0.05$. Support is shown in terms of number of languages exhibiting such grammatical distinctions.

Edit-tree vs. Encoder-Decoder Our fine-grained analysis suggests that the performance of the edit-tree and encoder-decoder approaches depends on the underlying morphological typology of the studied languages. Neural approaches seem to be stronger for languages with complex case systems and agglutinative morphology. In contrast, edit-tree approaches excel on more synthetic languages (e.g. Type 3) and languages with lower ambiguity (e.g. Type 2).

Figure 5 illustrates that as the number of edit-trees increase the encoder-decoder models start to excel. This is most likely due to the fact that, from an edit-tree approach perspective, a large number of trees creates a large number of classes, which leads to higher class imbalance and more sparsity. However, edit-tree based approaches do outperform representation learning methods for languages with lower number of trees, which leads to the intuition that the edit-tree formalism does provide a useful inductive bias to the task of lemmatization and it should not be discarded in future work. Our results, in fact, point to a future direction which applies the edit-tree formalism, but alleviates the edit-tree explosion by exploiting the relationships between the edit-tree classes potentially using representation learning methods.

---

[11] Note that not all tasks are available for all languages, due to some corpora not providing all annotations and some categories not being relevant for particular languages.

[12] Models trained for 50 epochs using the Adam optimizer with default learning rate and training stops after 2 epochs without accuracy increase on dev set.

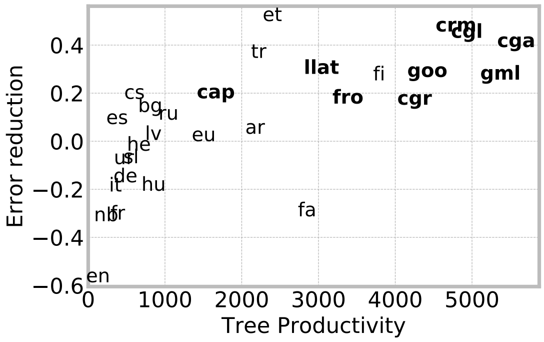


Figure 5: Error reduction of best encoder-decoder model vs best tree-edit model over tree productivity computed as number of unique binary edit-trees in the first 50k tokens of the training corpora (Spearman's $R = 0.79$; $p < .001$). Historical languages are shown in bold.

Accuracy on unknown tokens We observe that while overall the joint loss outperforms the simpler encoder-decoder, it seems, however, detrimental to the accuracy on unknown tokens. This discrepancy is probably due to the fact that (i) unknown tokens are likely unambiguous and therefore less likely to profit from improved context representations and to (ii) our design choice of word-level language modeling, where the model is forced to predict UNK for unknown words. As `Sent-LM` is the overall best model, in future work we will explore character-level language modeling in order to harness the full potential of the joint-training approach even on unknown tokens.

7 Conclusion

We have presented a method to improve lemmatization with encoder-decoder models by improving context representations with a joint bidirectional language modeling loss. Our method sets a new state-of-the-art for lemmatization of historical languages and is competitive on standard languages. Our examination of the learned representations indicates that the LM loss helps enriching sentence representations with features that capture morphological information. In view of a typologically informed comparison of encoder-decoder and edit-tree based approaches, we have shown that the latter can be very effective for highly synthetic languages. Such result might have been overlooked in previous studies due to only considering a reduced number of languages (Chakrabarty et al., 2017) or pooling results across typology (Bergmanis and Goldwater, 2018). With respect to languages with higher ambiguity and token-lemma ratio, the encoder-decoder approach is preferable and the joint loss generally provides a substantial improvement. Finally, while other models use morphological information to improve the representation of context (e.g. edit-tree approaches), our joint language modeling loss does not rely on any additional annotation, which can be crucial in low resource and non-standard situations where annotation is costly and often not trivial.

Acknowledgments

We thank NVIDIA for donating 1 GPU that was used for the experiments in the present paper. We would also like to thank the anonymous reviewers for their valuable comments.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *ICLR '17*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Fabian Barteld, Katharina Dreessen, Sarah Ihden, and Ingrid Schröder. 2017. Das Referenzkorpus Mittelniederdeutsch/Niederrheinisch (1200–1650)–Korpusdesign, Korpuserstellung und Korpusnutzung. *Mitteilungen des Deutschen Germanistenverbandes*, 64(3):226–241.

Toms Bergmanis and Sharon Goldwater. 2018. Context Sensitive Neural Lemmatization with Lematus. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies Volume 1*.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bidirectional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139. The COLING 2016 Organizing Committee.

Jean-Baptiste Camps. 2016. Geste: un corpus de chansons de geste. Avec la collab. d'Elena Albarran, Alice Cochet & Lucence Ing.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1491. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning Morphology with Morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint Lemmatization and Morphological Tagging with LEMMING. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

D. Crystal. 2001. *Language and the Internet*. Cambridge University Press.

Oksana Dereza. 2018. Lemmatization for Ancient Languages: Rules or Neural Networks? In *Conference on Artificial Intelligence and Natural Language*, pages 35–47. Springer.

Steffen Eger, Rdiger Gleim, and Alexander Mehler. 2016. Lemmatization and Morphological Tagging in German and Latin: A Comparison and a Survey of the State-of-the-art. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Tomaž Erjavec. 2015. Reference corpus of historical slovene goo300k 1.2. Slovenian language resource repository CLARIN.SI.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Andrea Gesmundo and Tanja Samardži. 2012. Lemmatisation as a Tagging Task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 368–372, Jeju Island, Korea. Association for Computational Linguistics.

Hans van Halteren and Margit Rem. 2013. Dealing with orthographic variation in a tagger-lemmatizer for fourteenth century dutch charters. *Language Resources and Evaluation*, 47(4):1233–1259.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Mike Kestemont, Guy De Pauw, Renske van Nie, and Walter Daelemans. 2016. Lemmatization for variation-rich languages using deep learning. *Digital Scholarship in the Humanities*, 32(4):797–815.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15.

G. Knowles and Z. Mohd Don. 2004. The notion of a lemma. Headwords, roots and lexical sets. *International Journal of Corpus Linguistics*, 9(1):69–81.

Daniel Kondratyuk, Tom Gavenčiak, and Milan Straka. 2018. LemmaTag: Jointly Tagging and Lemmatizing for Morphologically-Rich Languages with BRNNs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928, Brussels, Belgium.

Timo Korkiakangas and Matti Lassila. 2013. Abbreviations, fragmentary words, formulaic language: treebanking mediaeval charter material. In *Proceedings of The Third Workshop on Annotation of Corpora for Research in the Humanities*, pages 61–72.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference* *of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Eva Pettersson, Beta Megyesi, and Joakim Nivre. 2014. A Multilingual Evaluation of Three Spelling Normalisation Methods for Historical Text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41. Association for Computational Linguistics.

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis Lectures on Human Language Technologies*, 5(2):1–157.

Sarah Schulz, Guy De Pauw, Orphe De Clercq, Bart Desmet, Veronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular text normalization of dutch user-generated content. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(4):61.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hctor Martínez Alonso. 2014. What's in a p-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An Evaluation of Neural Machine Translation Models on Historical Spelling Normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331. Association for Computational Linguistics.

P. Van Reenen and M. Mulder. 1993. Een gegevensbank van 14de-eeuwse middelnederlandse dialecten op computer. *Lexikos*, 3:259–279.

A Appendix

A.1 Dataset Statistics

Table 6 shows the dataset sources and codes from our Historical Languages.

Language	Dataset	Code
Medieval Latin	Capitularia	cap
	LLCT1	llat
Middle Dutch	Gys (Admin)	cga
	Gys (Literary)	cgl
	Religious	cgr
	Adelheid	crm
Medieval French	Geste	fro
Middle Low German	ReN	gml
Slovenian	goo300k	goo

Table 6: Corpus identifier and description in the historical languages dataset. "Gys" refers to the Gysseling corpus, which consists of several subsets.

Table 7 shows the languages from the UD corpus that were sampled for the study. We have used ISO 639-1 codes (instead of the more general ISO 639-2) in order to avoid clutter in the plots.

Language	Dataset	Code
Arabic	Arabic-PDAT	ar
Bulgarian	Bulgarian-BTB	bg
Czech	Czech-CAC	cs
German	German-GSD	de
English	English-EWT	en
Spanish	Spanish-AnCora	es
Estonian	Estonian-EDT	et
Basque	Basque-BDT	eu
Persian	Persian-Seraji	fa
Finnish	Finnish-TDT	fi
French	French-GSD	fr
Hebrew	Hebrew-HTB	he
Hungarian	Hungarian-Szeged	hu
Italian	Italian-ISDT	it
Latvian	Latvian-LVTB	lv
Norwegian (Bokmaal)	Norwegian-Bokmaal	nb
Russian	Russian-SynTagRus	ru
Slovenian	Slovenian-SSJ	sl
Turkish	Turkish-IMST	tr
Urdu	Urdu-UDTB	ur

Table 7: Standard language datasets from the Universal Dependencies (v2.2) corpus.

One Size Does Not Fit All:
Comparing NMT Representations of Different Granularities

Nadir Durrani **Fahim Dalvi** **Hassan Sajjad** **Yonatan Belinkov**[*] **Preslav Nakov**
{ndurrani,faimaduddin,hsajjad,pnakov}@qf.org.qa
Qatar Computing Research Institute, HBKU Research Complex, Doha 5825, Qatar

[*]MIT Computer Science and Artificial Intelligence Laboratory and
Harvard John A. Paulson School of Engineering and Applied Sciences, Cambridge, MA, USA
belinkov@mit.edu

Abstract

Recent work has shown that contextualized word representations derived from neural machine translation are a viable alternative to such from simple word predictions tasks. This is because the internal understanding that needs to be built in order to be able to translate from one language to another is much more comprehensive. Unfortunately, computational and memory limitations as of present prevent NMT models from using large word vocabularies, and thus alternatives such as subword units (BPE and morphological segmentations) and characters have been used. Here we study the impact of using different kinds of units on the quality of the resulting representations when used to model morphology, syntax, and semantics. We found that while representations derived from subwords are slightly better for modeling syntax, character-based representations are superior for modeling morphology and are also more robust to noisy input.

1 Introduction

Recent years have seen the revolution of deep neural networks and the subsequent rise of representation learning based on network-internal activations. Such representations have been shown useful when addressing various problems from fields such as image recognition (He et al., 2016), speech recognition (Bahdanau et al., 2016), and natural language processing (NLP) (Mikolov et al., 2013a). The central idea is that the internal representations trained to solve an NLP task could be useful for other tasks as well. For example, word embeddings learned for a simple word prediction task in context, word2vec-style (Mikolov et al., 2013b), have now become almost obligatory in state-of-the-art NLP models. One issue with such word embeddings is that the resulting representation is context-independent.

Recently, it has been shown that huge performance gains can be achieved by contextualizing the representations, so that the same word could have a different embedding in different contexts. This is best achieved by changing the auxiliary task. For example, ELMo (Peters et al., 2018) learns contextualized word embeddings from language modeling (LM) using long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997).

It has been further argued that complex auxiliary tasks such as neural machine translation (NMT) are better tailored for representation learning, as the internal understanding of the input language that needs to be built by the network to be able to translate from one language to another needs to be much more comprehensive compared to what would be needed for a simple word prediction task. This idea is implemented in the seq2seq-based CoVe model (McCann et al., 2017).

More recently, the BERT model (Devlin et al., 2019) proposed to use representation from another NMT model, the Transformer, while optimizing for two LM-related auxiliary tasks: (*i*) masked language model and (*ii*) next sentence prediction.

Another important aspect of representation learning is the basic unit the model operates on. In word2vec-style embeddings, it is the word, but this does not hold for NMT-based models, as computational and memory limitations, as of present, prevent NMT from using a large vocabulary, typically limiting it to 30-50k words (Wu et al., 2016). This is a severe limitation, as most NLP applications need to handle vocabularies of millions of words, e.g., word2vec (Mikolov et al., 2013b), GloVe (Pennington et al., 2014) and FastText (Mikolov et al., 2018) offer pre-trained embeddings for 3M, 2M, and 2.5M words/phrases. The problem is typically addressed using byte-pair encoding (BPE), where words are segmented into pseudo-word sequences (Sennrich et al., 2016).

Proceedings of NAACL-HLT 2019, pages 1504–1516
Minneapolis, Minnesota, June 2 - June 7, 2019.

A less popular solution is to use characters as the basic unit (Chung et al., 2016; Lee et al., 2017), and in the case of morphologically complex languages, yet another alternative is to reduce the vocabulary size by using unsupervised morpheme segmentation (Bradbury and Socher, 2016).

The impact of using different units of representation in NMT models has been studied in previous work (Ling et al., 2015; Costa-jussà and Fonollosa, 2016; Chung et al., 2016; Lee et al., 2017, among others), but the focus has been exclusively on the quality of the resulting translation output. However, it remains unclear what input and output units should be chosen if we are primarily interested in representation learning. Here, we aim at bridging this gap by evaluating the quality of NMT-derived embeddings originating from units of different granularity when used for modeling morphology, syntax, and semantics (as opposed to end tasks such as sentiment analysis and question answering). Our contributions are as follows:

- We study the impact of using words vs. characters vs. BPE units vs. morphological segments on the quality of representations learned by NMT models when used to model morphology, syntax, and semantics.
- We further study the robustness of these representations with respect to noise.

We found that while representations derived from morphological segments are better for modeling non-local syntactic and semantic dependencies, character-based ones are superior for morphology and are also more robust to noise. There is also value in combining different representations.

2 Related Work

Representation analysis aims at demystifying what is learned inside the neural network blackbox. This includes analyzing word and sentence embeddings (Adi et al., 2017; Qian et al., 2016b; Ganesh et al., 2017; Conneau et al., 2018, among others), RNN states (Qian et al., 2016a; Shi et al., 2016; Wu and King, 2016; Wang et al., 2017), and NMT representations (Shi et al., 2016; Belinkov et al., 2017a), as applied to morphological (Vylomova et al., 2017; Dalvi et al., 2017), semantic (Qian et al., 2016b; Belinkov et al., 2017b) and syntactic (Linzen et al., 2016; Tran et al., 2018; Conneau et al., 2018) tasks. See Belinkov and Glass (2019) for a recent survey. Other studies carried a more fine-grained neuron-level analysis for NMT and LM (Dalvi et al., 2019; Bau et al., 2019; Lakretz et al., 2019). While previous work focused on words, here we compare units of different granularities.

Subword translation units aim at reducing the vocabulary size and the out-of-vocabulary (OOV) rate. Researchers have used BPE units (Sennrich et al., 2016), morphological segmentation (Bradbury and Socher, 2016), characters (Durrani et al., 2014; Lee et al., 2017), and hybrid units (Ling et al., 2015; Costa-jussà and Fonollosa, 2016) to address the OOV word problem in MT. The choice of translation unit impacts what the network learns. Sennrich (2017) carried a systematic error analysis by comparing subword versus character units and found the latter to be better at handling OOV and transliterations, whereas BPE-based subword units were better at capturing syntactic dependencies. In contrast, here we focus on representation learning, not translation quality.

Robustness to noise is an important aspect in machine learning. It has been studied for various models (Szegedy et al., 2014; Goodfellow et al., 2015), including NLP in general (Papernot et al., 2016; Samanta and Mehta, 2017; Liang et al., 2018; Jia and Liang, 2017; Ebrahimi et al., 2018; Gao et al., 2018), and character-based NMT in particular (Heigold et al., 2018; Belinkov and Bisk, 2018). Unlike this work, we compare robustness to noise for units of different granularity. Moreover, we focus on representation learning rather than on the quality of the translation output.

3 Methodology

Our methodology is inspired by research on interpreting neural network (NN) models. A typical framework involves extracting feature representations from different components (e.g., encoder/decoder) of a trained model and then training a classifier to make predictions for an auxiliary task. The performance of the trained classifier is considered to be a proxy for judging the quality of the extracted representations with respect to the particular auxiliary task.

Formally, for each input word $\mathbf{x}_i$ we extract the corresponding LSTM hidden state(s) from each layer of the encoder/decoder. We then concatenate the representations of the layers and use them as a feature vector $\mathbf{z}_i$ for the auxiliary task.

Words	Obama	receives	Netanyahu	in	the	capital	of	USA
POS	NP	VBZ	NP	IN	DT	NN	IN	NP
Sem.	PER	ENS	PER	REL	DEF	REL	REL	GEO
CCG	NP	((S[dcl]\NP)/PP)/NP	NP	PP/NP	PP/N	N	(NP\NP)/NP	NP

Table 1: Example sentence with different annotations.

Words	He admits to shooting girlfriend
BPE	He admits to sho@@ oting gir@@ l@@ friend
Morfessor	He admit@@ s to shoot@@ ing girl@@ friend
Characters	H e _ a d m i t s _ t o _ s h o o t i n g _ g i r l f r i e n d

Table 2: Example with different segmentations.

We then train a logistic regression classifier, minimizing the cross-entropy loss:

$$\mathcal{L}(\theta) = -\sum_i \log P_\theta(\mathbf{l}_i|\mathbf{x}_i)$$

where $P_\theta(\mathbf{l}|\mathbf{x}_i) = \frac{\exp(\theta_l \cdot \mathbf{z}_i)}{\sum_{l'} \exp(\theta_{l'} \cdot \mathbf{z}_i)}$ is the probability that word $\mathbf{x}_i$ is assigned label $\mathbf{l}$.

We learn the weights $\theta \in \mathbb{R}^{D \times L}$ using gradient descent. Here D is the dimensionality of the latent representations $\mathbf{z}_i$ and L is the size of the label set for property $\mathcal{P}$. See Section 4 for details.

3.1 Word Representation Units

We consider four representation units: words, byte-pair encoding (BPE) units, morphological units, and characters. Table 2 shows an example of each representation unit. *BPE* splits words into symbols (a symbol is a sequence of characters) and then iteratively replaces the most frequent sequences of symbols with a new merged symbol. In essence, frequent character n-grams merge to form one symbol. The number of merge operations is controlled by a hyper-parameter `OP`; a high value of `OP` means coarse segmentation and a low value means fine-grained segmentation (Sajjad et al., 2017). For *morphologically segmented units*, we use an unsupervised morphological segmenter, Morfessor (Smit et al., 2014). Note that although BPE and Morfessor segment words at a similar level of granularity, the segmentation generated by Morfessor is linguistically motivated. For example, it splits the gerund verb *shooting* into root *shoot* and the suffix *ing*. Compare this to the BPE segmentation *sho* + *oting*, which has no linguistic connotation. On the extreme, the fully *character-level* units treat each word as a sequence of characters.

3.2 Extracting Activations for Subword and Character Units

Previous work on analyzing NMT representations has been limited to the analysis of word representations only,[1] where there is a one-to-one mapping from input units (words) and their NMT representations (hidden states) to their linguistic annotations (e.g., morphological tags).

In the case of subword-based systems, each word may be split into multiple subword units, and each unit has its own representation. It is less trivial to define which representations should be evaluated when predicting a word-level linguistic property such as part of speech. We consider two simple approximations to estimate a word representation from subword units:

(*i*) **Average**: for each source (or target) word, we average the activation values of all the subwords (or characters) comprising it. In the case of a bi-directional encoder, we concatenate the averages from the forward and the backward activations of the encoder on the subwords (or characters) that represent the current word.[2]

(*ii*) **Last**: we consider the activation of the last subword (or character) as the representation of the word. For the bi-directional encoder, we concatenate the forward encoder's activation on the last subword unit with the backward encoder's activation on the first subword unit.

This formalization allows us to analyze the quality of character- and subword-based representations at the word level via prediction tasks. Such kind of analysis has not been performed before.

---

[1] Belinkov et al. (2017a) analyzed representations trained from character CNN models (Kim et al., 2016), but the extracted features were still based on word representations produced by the character CNN. As a result, they could not analyze and compare results for the BPE and character-based models that do not assume segmentation into words.

[2] One could envision more sophisticated averages, such as weighting via an attention mechanism.

4 Linguistic Properties

We choose three fundamental NLP tasks that serve as a good representative of various properties inherent in a language, ranging from morphology (word structure), syntax (grammar), and semantics (meaning). In particular, we experiment with *morphological tagging* for German, Czech, Russian, and English,[3] *lexical semantic tagging* for English and German, and *syntactic tagging* via CCG supertagging for English. Table 1 shows an example sentence with annotations for each task.

The morphological tags capture word structure, the semantic tags reflect lexical semantics, and the syntactic tags (CCG supertags) capture global syntactic information locally, at the lexical level.

For example, in Table 1, the morphological tag `VBZ` for the word *receives* marks it as a verb in third person, singular, present tense; the semantic tag `ENS` describes a *present simple event* category; and the syntactic tag `PP/NP`) can be thought of as a function that takes a noun phrase on the right (e.g., *the capital of USA*), and returns a prepositional phrase (e.g., *in the capital of USA*).

Artificial Error Induction Recent studies have shown that small perturbations in the input can cause significant deterioration in the performance of the deep neural networks. Here, we evaluate the robustness of various representations under noisy input conditions. We use corpora of real errors harvested by Belinkov and Bisk (2018). The errors contain a good mix of typos, misspellings, and other kinds of errors. In addition, we created data with synthetic noise. We induced two kinds of errors: (*i*) *swap* and (*ii*) *middle*. *Swap* is a common error, which occurs when neighboring characters are mistakenly swapped, e.g., *word* → *wodr*. In *Middle* errors, the order of the first and the last characters of a word are preserved, while the middle characters are randomly shuffled (Rawlinson, 1976), e.g., *example* → *eaxmlpe*. We corrupt $n\%$ words randomly in each test sentence, using *swap* or *middle* heuristics, or replace words using real-error corpora. We then re-extract feature vectors for the erroneous words in a sentence and we re-evaluate the prediction capability of these embeddings on the linguistic tasks.

	de-en	cs-en	ru-en
Train	507K	340K	370K
Dev	3,000	3,000	2,818
Test	3,000	3,000	2,818

(a) NMT data

	de	cs	ru	en
Morphology	509	1,004	602	42
Semantics	69	–	–	66
Syntax	1,272	–	–	–

(b) Number of tags

	de	cs	ru	en
	Morphology			
Train	14,498	14,498	11,824	14498
Test	8,172	8,172	6,000	8,172
	Semantics			
Train	–	–	–	14,084
Test	–	–	–	12,168
CV	1,863	–	–	–
	Syntax			
Train	–	–	–	41,586
Test	–	–	–	2,407

(c) Classifier data

Table 3: Statistics about NMT and classifier training data for English (en), German (de), Russian (ru), and Czech (cs). Here, CV stands for cross-validation.

5 Experimental Setup

Data and Languages We trained NMT systems for four language pairs: German-English, Czech-English, Russian-English, and English-German, using data made available through two popular machine translation campaigns, namely, WMT (Bojar et al., 2017) and IWSLT (Cettolo et al., 2016). We trained the MT models using a concatenation of the NEWS and the TED training datasets, and we tested on official TED test sets (testsets-11-13) to perform the evaluation using BLEU (Papineni et al., 2002). We trained the morphological classifiers and we tested them on a concatenation of the NEWS and the TED testsets, which were automatically tagged as described in the next paragraph. We trained and evaluated the semantic and the syntactic classifiers on existing annotated corpora. See Table 3 for details about the datasets.

---

[3] As English is morphologically poor, we use part-of-speech tags for it. We refer to English part-of-speech tags as morphological tags later in the paper in order to keep the terminology consistent.

Taggers We used RDRPOST (Nguyen et al., 2014) to annotate data for the classifier. For semantic tagging, we used the gold-annotated semantic tags from the Groningen Parallel Meaning Bank (Abzianidze et al., 2017), which were made available by (Bjerva et al., 2016). The tags are grouped into coarse categories such as events, names, time, and logical expressions. There is enough data for English (≈42K), and we randomly sampled the same amount of data we used to train our morphological classifiers to train the semantic classifiers. Yet, only 1,863 annotated sentences (12,783 tokens) were available for German. Thus, in the experiments, we performed 5-fold cross-validation. For CCG supertagging, we used the English CCGBank (Hockenmaier and Steedman, 2007), which contains 41,586/2,407 train/test sentences.[4] See Table 3 for more detailed statistics about the train/dev/test datasets we used.

MT Systems and Classifiers We used `seq2seq-attn` (Kim, 2016) to train a two-layer encoder-decoder NMT model based on LSTM representation with attention (Hochreiter and Schmidhuber, 1997) with a bidirectional encoder and a unidirectional decoder.[5] We used 500 dimensions for both word embeddings and LSTM states. We trained the systems with SGD for 20 epochs and we used the final model, i.e., the one with the lowest loss on the development dataset, to generate features for the classifier.

We trained our neural machine translation models in both *-to-English and English-to-* translation directions, and we analyzed the representations from both the *encoder* and the *decoder*. In order to analyze the representations derived from the encoder side, we fixed the decoder side with BPE-based embeddings, and we trained the source side with word/BPE/Morfessor/character units. Similarly, when analyzing the representations from the decoder side, we trained the encoder representation with BPE units, and we varied the decoder side using word/BPE/char units. Our motivation for this setup is that we wanted to analyze the encoder/decoder side representations in isolation, keeping the other half of the network (i.e., the decoder/encoder) static across different settings.[6]

In our experiments, we used 50k BPE operations and we limited the vocabulary of all systems to 50k. Moreover, we trained the word, BPE, Morfessor, and character-based systems with maximum sentence lengths of 80, 100, 100, and 400 units, respectively.

For the classification tasks, we used a logistic regression classifier whose input is either the hidden states in the case of the word-based models, or the **Last** or the **Average** representations in the case of character- and subword-based models. Since for the bidirectional encoder we concatenate forward and backward states from all layers, this yields 2,000/1,000 dimensions when classifying using the representations from the encoder/decoder: 500 dimensions×2 layers×2 directions (1 for the decoder, as it is uni-directional). In all cases, we trained the logistic regression classifier for ten epochs.

6 Results

We now present the evaluation results for using representations learned from different input units to predict morphology, semantics, and syntax. For subword and character units, we found the activation of the last subword/character unit of a word to be consistently better than using the average of all activations (See Table 4). Therefore, we report only the results using the **Last** method, for the remainder of the paper.

	de		cs		ru	
	sub	char	sub	char	sub	char
Last	78.5	80.5	78.6	88.3	80.0	88.8
Avg	76.3	79.2	76.4	84.9	78.3	84.4

Table 4: Classifier accuracy for the representations generated by aggregating subword (sub) or character (char) representations using either the average or the last LSTM state for each word.

6.1 Morphological Tagging

Figure 1 summarizes the results for predicting morphological tags with representations learned using different units. The character-based representations consistently outperformed other representations on all language pairs, while the word-based ones performed worst. The differences are more significant in the case of languages with relatively complex morphology such as Czech.

---

[4] There are no available CCG banks for the other languages we experiment with, except for a German CCG bank, which is not publicly available (Hockenmaier, 2006).

[5] The decoder has to be unidirectional as, at decoding time, the future is unknown.

[6] Heigold et al. (2018) used a similar setup.

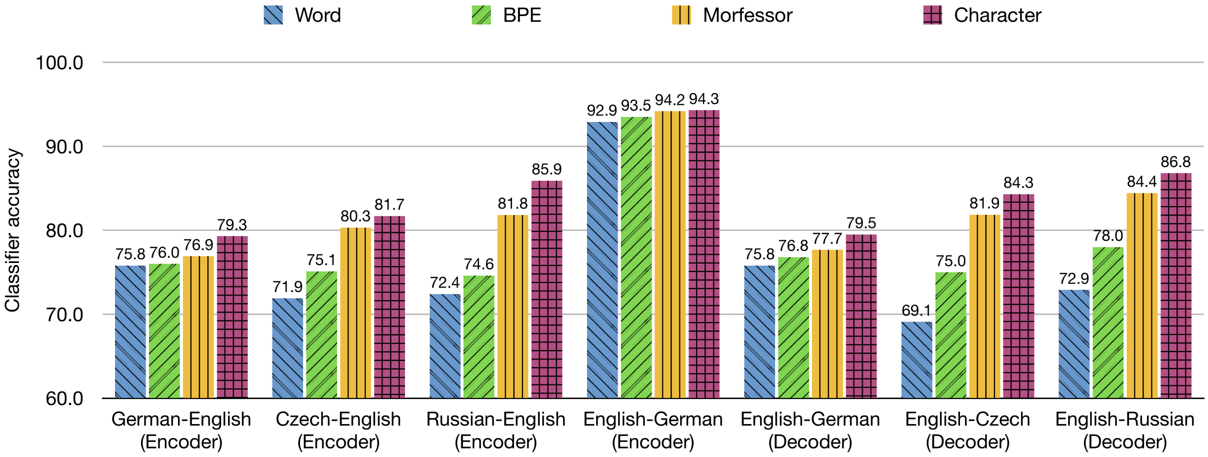


Figure 1: Morphological tagging accuracy across various systems and language pairs. The first four groups of results use BPE on the decoder side, while the last three groups use BPE on the encoder side.

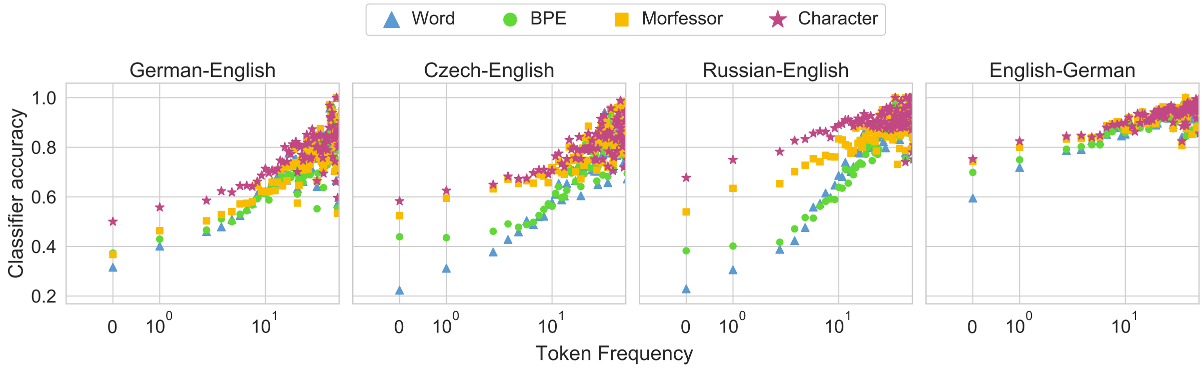


Figure 2: Morphological tagging accuracy vs. word frequency for different encoding units. Best viewed in color.

	de-en	cs-en	ru-en	en-de
word → bpe	34.0	27.5	20.9	29.7
bpe → bpe	35.6	28.4	22.4	30.2
morf → bpe	35.5	28.5	22.5	29.9
char → bpe	34.9	29.0	21.3	30.0

Table 5: BLEU scores across language pairs. "**s** → t" means **s**ource and **t**arget units; morf = Morfessor.

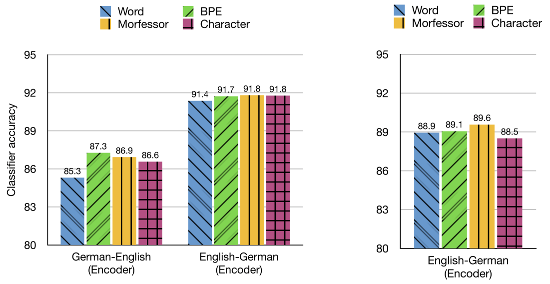
	de-en	cs-en	ru-en	en-de
MT	3.42	6.46	6.86	0.82
Classifier	4.42	6.13	6.61	2.09

Table 6: OOV rate for the MT and the classifier testsets.

We see in Figure 1 a difference of up to 14% in favor of character-based representations when compared with word-based ones. The improvement is minimal in the case of English (1.2%), which is a morphologically simpler language. This is also somewhat reflected in the translation quality.

We can see in Table 5 that character-based segmentation yielded higher BLEU scores in the case of a morphologically rich language such as Czech, but performed poorly in the case of German, which requires handling long-distance dependencies. Comparing subword units, we found Morfessor to yield much better morphological tagging performance, especially in the case of morphologically rich languages such as Czech and Russian, supposedly due to the Morfessor's linguistically motivated segmentations, which are helpful for learning morphology.

We further investigated whether the performance difference between the representations is due to the difference in modeling infrequent and out-of-vocabulary (OOV) words. Table 6 shows the OOV rate for each language, which is higher for morphologically rich languages. Figure 2 shows that the gap between different representations is inversely proportional to the frequency of



(a) Semantic tagging. (b) Syntactic tagging.

Figure 3: Semantic and syntactic tagging for English and German (using BPE on the decoder).

the word in the training data, and character-based models handle infrequent and OOV words better.

Decoder Representations Next, we used the decoder representations from the English-to-* models. We saw similar performance trends as for the encoder-side representations: characters performed best, and words performed worst. Again, the morphological units performed better than the BPE-based units. Comparing encoder representations to decoder representations, it is interesting to see that in several cases the decoder side representations performed better than the encoder side ones, even though the former were trained using a uni-directional LSTM. However, since there is no difference in the general trends between the encoder- and the decoder-side representations, below we focus on the encoder-side only.

6.2 Semantic Tagging

Figure 3a summarizes the experimental results for evaluating representtaion units of the semantic tagging task. For English, the subword (BPE and Morfessor) and the character representations yielded comparable results. However, for German, BPE performed better. This is in contrast with the morphology prediction experiments, where the character representations were consistently better. We will discuss this in more detail in Section 7.

6.3 Syntactic Tagging

The final task we experimented with is CCG super-tagging, which reflects modeling syntactic structure. Here we only have English tags, and thus we evaluate the performance of encoder representations for English→German models, trained using words, characters, and subword units.

We can see in Figure 3b that the morphologically segmented representation units performed the best overall. Moreover, there is no much difference when using word-based vs. BPE-based representations.

The character-based representations lag behind, but the difference in accuracy is small compared to the morphological tagging results.[7] It is noteworthy that here character-based representations perform worse than both words and subwords, contrary to their superior performance on morphology. We will return to this in Section 7 below.

6.4 Robustness to Noise

Next, we evaluated the robustness of the representations with respect to noise. We induced errors in the test sets by corrupting 25% of the words in each sentence using different error types (synthetic or real noise), as described in Section 4. We extracted the representations of the noisy test sets and we re-evaluated the classifiers. Figure 4 shows the performance on each task. We can see that characters yielded much better performance on all tasks and for all languages, showing minimal drop in accuracy, in contrast to earlier results where they did not outperform subword units[8] on the task of syntactic tagging. This shows that character representations are more robust to noise.

Surprisingly, in a few cases, BPE performed worse than word units, e.g., in the case of syntactic tagging (80.3 vs. 81.1). We found that BPE can segment a noisy word into two or more known subword units that have no real relationship to the actual word. Thus, using representations of wrong subword units could hurt the performance.

We further investigated the robustness of each classifier by increasing the percentage of noise in the test data. We found that the difference in representation quality stays constant across BPE and character representations, whereas word representations deteriorate significantly as the amount of noise increases (see Figure 5).

7 Discussion

7.1 Performance Across Various Tasks

Our experiments show a complicated picture, where none of the representations is superior in all scenarios. Characters were found to be better for morphological tagging, BPE was ahead in

---

[7] For perspective, these numbers are above a majority class baseline of 72% and below the state-of-the-art, which is around 94-95% (Kadari et al., 2018; Xu, 2016).

[8] Morphological segmentation showed similar results compared to BPE-based segmentation in these experiments.

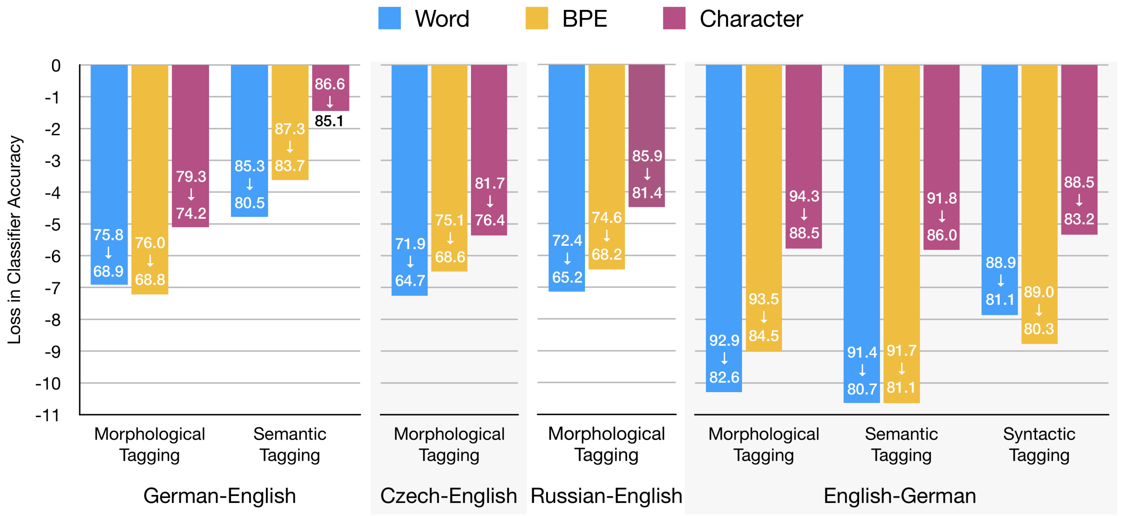
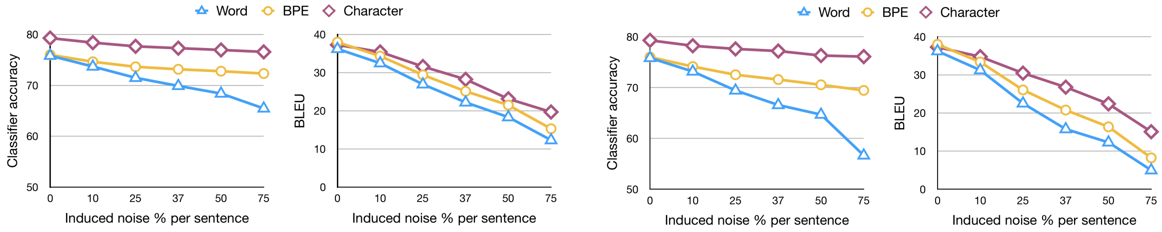


Figure 4: Classification accuracy for morphological, syntactic and semantic tagging with 25% noise in each sentence. Absolute scores (original → noisy) are shown inset.



(a) Real noise.

(b) Synthetic noise.

Figure 5: Results for morphological classification when adding induced noise.

the semantic tagging task for German (and about the same in English), and Morfessor units were slightly better for syntax.

Syntactic tagging requires knowledge about the complete sentence. Splitting a sentence into characters substantially increases its length: on average from 50 words to 250 single-character tokens. Thus, character-based models struggle to capture long-distance dependencies. Sennrich (2017) also found this to be true in their evaluation based on contrastive translation pairs in German-English.

Similarly, in the case of morphological tagging, the information about the morphological structure of a word is dependent on the surrounding words plus some internal information (root, morphemes, etc.) present inside the word. A character-based system has access to all of this information, and thus performs well. Morphological segmentation performed better than BPE for the morphological tagging because its segments are linguistically motivated units (segmented into root + morphemes), thus making the information about the word morphology explicit in the representation. In contrast, BPE solely focuses on the frequency of characters occurring together in the corpus, and thus can generate linguistically incorrect units.

		W	B	C	W+B	B+C	W+C	ALL
Morph	DE	75.8	76.0	79.3	78.0	80.8	81.1	**81.6**
	CS	71.9	75.1	81.7	77.2	84.0	84.1	**85.0**
	RU	72.4	74.6	85.9	77.1	88.1	88.2	**88.6**
	EN	92.9	93.5	94.3	94.2	95.2	95.1	**95.4**
Sem	EN	91.1	91.4	91.4	92.6	93.0	93.1	**93.4**

Table 7: Classification accuracy for combined representations for morphological and semantic tagging. Here, W/B/C stand for word/BPE/character units.

7.2 The Best of Many Worlds

The variations in performance for different representations suggest that they are learning different aspects of language, which might be complementary. Thus, we tried to combine them. Table 7 summarizes the results for morphological and semantic tagging.[9] We can see that combinations involving characters (B+C, W+C in the table) yield

---

[9] We observed similar trends for the other tasks.

larger improvement compared to combining word- and BPE-based representations (W+B). However, combining all three performed best for all languages and for all tasks.

7.3 State-of-the-Art Embeddings

We connect our findings with recent work on training state-of-the-art embeddings: CoVe (McCann et al., 2017), ELMo (Peters et al., 2018), and BERT (Devlin et al., 2019). Each of these architectures uses different units of representations: e.g., CoVe uses words, BERT is based on subword units, while ELMo focuses on characters.[10]

We speculate that, although these models yield state-of-the-art results for several tasks, their performance may be suboptimal because of the choice of their underlying representation units. In our experiments above, we have shown that it is possible to achieve potentially better performance when using units of different granularity jointly.

We have further shown that the best-performing representation unit is target task-dependent. We believe this would be true for more complex NLP tasks as well. For example, question answering generally requires learning long-range dependencies, and thus embeddings from a character-based model might not be the right choice in this case. Our results show that character-based models are not a viable option for handling long-range dependencies, and subword-based representations might be a better option for such tasks.

7.4 Translation vs. Representation Quality

Table 5 summarizes the translation performance of each system. We can see that in most cases, the subword-based systems perform better than the word-based and the character-based ones. However, this is not true in the case of using their representations as features for a core NLP task as in our experiments above. For example, we have found that character-based representations perform best for the morphological tagging task. On a side note, although BPE-based representations perform better for some tasks, they are sensitive to noise. Their capability of segmenting any OOV word into known subwords may result in less reliable systems. Notably, the translation performance of the BPE-based system can fall below that of the character-based system even in the presence of only 10% noise: from 0.7 BLEU in favor of BPE on clean data to 0.8/1.1 BLEU in favor of characters with synthetic/real errors.

8 Conclusion and Future Work

We studied the impact of using different representation units—words, characters, BPE units, and morphological segments—on the representations learned by seq2seq models trained for neural machine translation. In particular, we evaluated the performance of such representations on core natural language processing tasks modeling morphology, syntax, and semantics.

Based on our experiments, we can make the following conclusions:

- Representations derived from subword units are better for modeling syntax.
- Character-based representations are distinctly better for modeling morphology.
- Character-based representations are very robust to noise.
- Using a combination of different representations often works best.

Based on our findings, we further conjecture that although subword-based segmentation based on BPE are a de-facto standard when building state-of-the-art NMT systems, the underlying representations they yield are suboptimal for many external tasks. Character-based representations provide a more viable and robust alternative in this regard, followed by morphological segmentation.

In future work, we plan to study how different units affect representation quality in non-recurrent models such as the Transformer (Vaswani et al., 2017) as well as in convolutional architectures (Gehring et al., 2017). We would also like to explore representations from robustly trained systems, which should improve performance on noisy input (Belinkov and Bisk, 2018; Heigold et al., 2018). Finally, it would be interesting to study representations in other NLP tasks besides neural machine translation.

Acknowledgements

This work was funded by the QCRI, HBKU, as part of the collaboration with the MIT, CSAIL. Yonatan Belinkov was also partly supported by the Harvard Mind, Brain, and Behavior Initiative.

---

[10] However, note that ELMo uses character convolutions, which is different from a fully character-based model.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 242–247, Valencia, Spain.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of the International Conference on Learning Representations*, ICLR '17, Toulon, France.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '16, pages 4945–4949, Shanghai, China.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *Proceedings of the International Conference on Learning Representations*, ICLR '19, New Orleans, LA, USA.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations*, ICLR '18, Vancouver, Canada.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 861–872, Vancouver, Canada.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, IJCNLP '17, pages 1–10, Taipei, Taiwan.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proceedings of the 26th International Conference on Computational Linguistics*, COLING '16, pages 3531–3541, Osaka, Japan.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation. In *Proceedings of the Second Conference on Machine Translation*, WMT '17, pages 169–214, Copenhagen, Denmark.

James Bradbury and Richard Socher. 2016. MetaMind neural machine translation system for WMT 2016. In *Proceedings of the First Conference on Machine Translation*, WMT '16, pages 264–267, Berlin, Germany.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Bentivogli Luisa, and Marcello Federico. 2016. The IWSLT 2016 evaluation campaign. In *Proceedings of the 13th International Workshop on Spoken Language Translation*, IWSLT '16, Seattle, WA, USA.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 1693–1703, Berlin, Germany.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 2126–2136, Melbourne, Australia.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 357–361, Berlin, Germany.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D. Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? Analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI '19, Honolulu, HI, USA.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, IJCNLP '17, pages 142–151, Taipei, Taiwan.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '19, Minneapolis, MN, USA.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an unsupervised transliteration model into statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '14, pages 148–153, Gothenburg, Sweden.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 31–36, Melbourne, Australia.

J. Ganesh, Manish Gupta, and Vasudeva Varma. 2017. Interpretation of semantic tweet representations. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, pages 95–102, Sydney, Australia.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proceedings of the 2018 IEEE Security and Privacy Workshop*, SPW '18, pages 50–56, San Francisco, CA, USA.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning*, ICML '17, pages 1243–1252, Sydney, Australia.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, ICLR '15, San Diego, CA, USA.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778, Las Vegas, CA, USA.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef Genabith. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas*, AMTA '18, pages 68–80, Boston, MA, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Julia Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, ACL '06, pages 505–512, Sydney, Australia.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2011–2021, Copenhagen, Denmark.

Rekia Kadari, Yu Zhang, Weinan Zhang, and Ting Liu. 2018. CCG supertagging via Bidirectional LSTM-CRF neural architecture. *Neurocomputing*, 283:31–37.

Yoon Kim. 2016. Seq2seq-attn. `https://github.com/harvardnlp/seq2seq-attn`.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749, Phoenix, AZ, USA.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '19, Minneapolis, MN, USA.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI '18, pages 4208–4215, Stockholm, Sweden.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based neural machine translation. *CoRR*, abs/1511.04586.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems 30*, NIPS '17, pages 6297–6308, Long Beach, CA, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the ICLR Workshop*, Scottsdale, AZ, USA.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation*, LREC '18, Miyazaki, Japan.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, NIPS '13, pages 3111–3119, Stateline, NV, USA.

Dat Quoc Nguyen, Dai Quoc Nguyen, Dang Duc Pham, and Son Bao Pham. 2014. RDRPOSTagger: A ripple down rules-based part-of-speech tagger. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '14, pages 17–20, Gothenburg, Sweden.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *Proceedings of the IEEE Military Communications Conference*, MILCOM '16, pages 49–54, Baltimore, MD, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, PA, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, Doha, Qatar.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '18, pages 2227–2237, New Orleans, LA, USA.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016a. Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 826–835, Austin, TX, USA.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016b. Investigating language universal and specific properties in word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 1478–1488, Berlin, Germany.

Graham Rawlinson. 1976. *The significance of letter position in word recognition*. Ph.D. thesis, University of Nottingham.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, Ahmed Abdelali, Yonatan Belinkov, and Stephan Vogel. 2017. Challenging language-dependent segmentation for Arabic: An application to machine translation and part-of-speech tagging. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL '17, pages 601–607, Vancouver, Canada.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? Assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '17, pages 376–382, Valencia, Spain.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 1715–1725, Berlin, Germany.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 1526–1534, Austin, TX, USA.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '14, pages 21–24, Gothenburg, Sweden.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, ICLR '14, Banff, Canada.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 4731–4736, Brussels, Belgium.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 5998–6008. Long Beach, CA, USA.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2017. Word representation models for morphologically rich languages in neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, SWCN '17, pages 103–108, Copenhagen, Denmark.

Yu-Hsuan Wang, Cheng-Tao Chung, and Hung-yi Lee. 2017. Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries. In *Proceedings of the 18th Annual Conference of the International Speech Communication Association*, Interspeech '2017, pages 3822–3826, Stockholm, Sweden.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Zhizheng Wu and Simon King. 2016. Investigating gated recurrent networks for speech synthesis. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '16, pages 5140–5144, Shanghai, China.

Wenduan Xu. 2016. LSTM shift-reduce CCG parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 1754–1764, Austin, TX, USA.

A Simple Joint Model for Improved Contextual Neural Lemmatization

Chaitanya Malaviya[*,β] and **Shijie Wu**[*,ə] and **Ryan Cotterell**[ə,ﬁ]
[β]Allen Institute for Artificial Intelligence
[ə]Department of Computer Science, Johns Hopkins University
[ﬁ]Department of Computer Science and Technology, University of Cambridge
`chaitanyam@allenai.org`, `shijie.wu@jhu.edu`, `rdc42@cam.ac.uk`

Abstract

English verbs have multiple forms. For instance, *talk* may also appear as *talks*, *talked* or *talking*, depending on the context. The NLP task of lemmatization seeks to map these diverse forms back to a canonical one, known as the lemma. We present a simple joint neural model for lemmatization and morphological tagging that achieves state-of-the-art results on 20 languages from the Universal Dependencies corpora. Our paper describes the model in addition to training and decoding procedures. Error analysis indicates that joint morphological tagging and lemmatization is especially helpful in low-resource lemmatization and languages that display a larger degree of morphological complexity. Code and pre-trained models are available at `https://sigmorphon.github.io/sharedtasks/2019/task2/`.

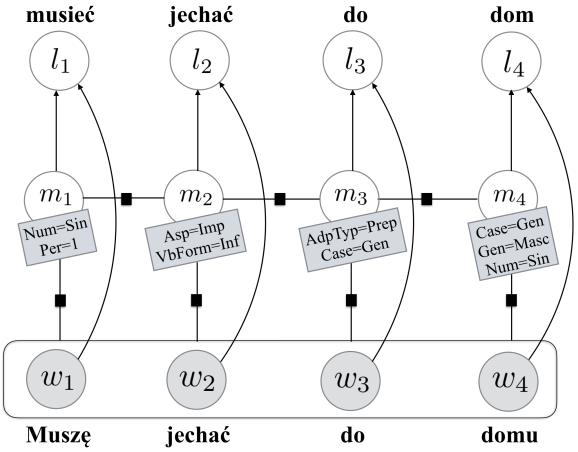


Figure 1: Our structured neural model shown as a hybrid (directed-undirected) graphical model (Koller and Friedman, 2009). Notionally, the w_i denote inflected word forms, the m_i denote morphological tags and the ℓ_i denote lemmata.

1 Introduction

Lemmatization is a core NLP task that involves a string-to-string transduction from an inflected word form to its citation form, known as the lemma. More concretely, consider the English sentence: *The bulls are running in Pamplona*. A lemmatizer will seek to map each word to a form you may find in a dictionary—for instance, mapping *running* to *run*. This linguistic normalization is important in several downstream NLP applications, especially for highly inflected languages. Lemmatization has previously been shown to improve recall for information retrieval (Kanis and Skorkovská, 2010; Monz and De Rijke, 2001), to aid machine translation (Fraser et al., 2012; Chahuneau et al., 2013) and is a core part of modern parsing systems (Björkelund et al., 2010; Zeman et al., 2018).

However, the task is quite nuanced as the proper choice of the lemma is context dependent. For instance, in the sentence *A running of the bulls took place in Pamplona*, the word *running* is its own lemma, since, here, *running* is a noun rather than an inflected verb. Several counter-examples exist to this trend, as discussed in depth in Haspelmath and Sims (2013). Thus, a good lemmatizer must make use of some representation of each word's sentential context. The research question in this work is, then, how do we design a lemmatization model that best extracts the morpho-syntax from the sentential context?

Recent work (Bergmanis and Goldwater, 2018) has presented a system that directly summarizes the sentential context using a recurrent neural network to decide how to lemmatize. As Bergmanis and Goldwater (2018)'s system currently achieves state-of-the-art results, it *must* implicitly learn a contextual representation that encodes the necessary morpho-syntax, as such knowledge is requisite for the task. We contend, however, that rather than expecting the network to *implicitly* learn some no-

---

* Equal contribution. Listing order is random.

Proceedings of NAACL-HLT 2019, pages 1517–1528
Minneapolis, Minnesota, June 2 - June 7, 2019.

Morph. Tag:	**POS**=D **CASE**=NOM **NUM**=PL	**POS**=N **CASE**=NOM **NUM**=PL	**POS**=N **CASE**=NOM **NUM**=PL **GEN**=FEM	**POS**=A **CASE**=NOM **NUM**=PL	**POS**=N **CASE**=NOM **NUM**=SG	**POS**=P	**POS**=N **CASE**=ACC **NUM**=SG
Lemma:	весь	счастливый	семья	похож	друг	на	друга…
Inflection:	Все	счастливые	семьи	похожи	друг	на	друга…
Translation:	All	happy	families	are similar		to each other	

Figure 2: Example of a morphologically tagged (in purple) and lemmatized (in red) sentence in Russian using the annotation scheme provided in the UD dataset. The translation is given below (in blue).

tion of morpho-syntax, it is better to *explicitly* train a joint model to morphologically disambiguate and lemmatize. Indeed, to this end, we introduce a joint model for the introduction of morphology into a neural lemmatizer. A key feature of our model is its *simplicity*: Our contribution is to show how to stitch existing models together into a joint model, explaining how to train and decode the model. However, despite the model's simplicity, it still achieves a significant improvement over the state of the art on our target task: lemmatization.

Experimentally, our contributions are threefold. First, we show that our joint model achieves state-of-the-art results, outperforming (on average) all competing approaches on a 20-language subset of the Universal Dependencies (UD) corpora (Nivre et al., 2017). Second, by providing the joint model with gold morphological tags, we demonstrate that we are far from achieving the upper bound on performance—improvements on morphological tagging could lead to substantially better lemmatization. Finally, we provide a detailed error analysis indicating *when* and *why* morphological analysis helps lemmatization. We offer two tangible recommendations: one is better off using a joint model (i) for languages with fewer training data available and (ii) languages that have richer morphology.

Our system and pre-trained models on all languages in the latest version of the UD corpora[1][2] are released at `https://sigmorphon.github.io/sharedtasks/2019/task2/`.

2 Background: Lemmatization

Most languages (Dryer and Haspelmath, 2013) in the world exhibit a linguistic phenomenon known as **inflectional morphology**, which causes word forms to mutate according to the syntactic category of the word. The syntactic context in which the word form occurs determines which form is properly used. One privileged form in the set of inflections is called the **lemma**. We regard the lemma as a lexicographic convention, often used to better organize dictionaries. Thus, the choice of which inflected form is the lemma is motivated by tradition and convenience, e.g., the lemma is the infinitive for verbs in some Indo-European languages, rather than by linguistic or cognitive concerns. Note that the **stem** differs from the lemma in that the stem may not be an actual inflection.[3] In the NLP literature, the syntactic category that each inflected form encodes is called the **morphological tag**. The morphological tag generalizes traditional part-of-speech tags, enriching them with further linguistic knowledge such as tense, mood, and grammatical case. We call the individual key–attribute pairs **morphological attributes**.

An example of a sentence annotated with morphological tags and lemmata in context is given in Figure 2. The task of mapping a sentence to a sequence of morphological tags is known as **morphological tagging**.

Notation. Let $\mathbf{w} = w_1, \ldots, w_n$ be a sequence of n words. Each individual word is denoted as w_i. Likewise, let $\mathbf{m} = m_1, \ldots, m_n$ and $\boldsymbol{\ell} = \ell_1, \ldots, \ell_n$ be sequences of morphological tags and lemmata, respectively. We will denote the set of all tags seen in a treebank as $\mathcal{Y}$. We remark that m_i is w_i's morphological tag (e.g. $[\text{POS}{=}\text{N}, \text{CASE}{=}\text{NOM}, \text{NUM}{=}\text{SG}]$ as a single label) and

---

[1] We compare to previously published numbers on non-recent versions of UD, but the models we release are trained on the current version (2.3).

[2] Instead of UD schema for morphological attributes, we use the UniMorph schema (Sylak-Glassman, 2016) instead. Note the mapping from UD schema to UniMorph schema is not one-to-one mapping (McCarthy et al., 2018).

[3] The stem is also often ill-defined. What is, for instance, the stem of the word *running*, is it *run* or *runn*?

ℓ_i is w_i's lemma. We will denote a language's discrete alphabet of characters as Σ. Thus, we have $w_i, \ell_i \in \Sigma^*$. Furthermore, we $\mathbf{c} = c_1, \ldots, c_n$ be a vector of characters where $c_i \in \Sigma$.

3 A Joint Neural Model

The primary contribution of this paper is a joint model of morphological tagging and lemmatization. The intuition behind the joint model is simple: high-accuracy lemmatization requires a representation of the sentential context, in which the word occurs (this behind has been evinced in §1)—a morphological tag provides the precise summary of the context required to choose the correct lemma. Armed with this, we define our joint model of lemmatization and morphological tagging as:

$$p(\boldsymbol{\ell}, \mathbf{m} \mid \mathbf{w}) = p(\boldsymbol{\ell} \mid \mathbf{m}, \mathbf{w})\, p(\mathbf{m} \mid \mathbf{w}) \quad (1)$$
$$= \left(\prod_{i=1}^{n} \underbrace{p(\ell_i \mid m_i, w_i)}_{\textit{Neural Transducer}} \right) \underbrace{p(\mathbf{m} \mid \mathbf{w})}_{\textit{Neural Tagger}}$$

Figure 1 illustrates the structure of our model in the form of a graphical model. We will discuss the lemmatization factor and the morphological tagging factor following two subsections, separately. We caution the reader that the discussion of these models will be brief: Neither of these particular components is novel with respect to the literature, so the formal details of the two models is best found in the original papers. The point of our paper is to describe a simple manner to combine these existing parts into a state-of-the-art lemmatizer.

3.1 Morphological Tagger: $p(\mathbf{m} \mid \mathbf{w})$

We employ a simple LSTM-based tagger to recover the morphology of a sentence (Heigold et al., 2017; Cotterell and Heigold, 2017). We also experimented with the neural conditional random field of Malaviya et al. (2018), but Heigold et al. (2017) gave slightly better tagging scores on average and is faster to train. Given a sequence of n words $\mathbf{w} = w_1, \ldots, w_n$, we would like to obtain the morphological tags $\mathbf{m} = m_1, \ldots, m_n$ for each word, where $m_i \in \mathcal{Y}$. The model first obtains a word representation for each token using a character-level biLSTM (Graves et al., 2013) embedder, which is then input to a word-level biLSTM tagger that predicts tags for each word. Given a function cLSTM that returns the last hidden state of a character-based LSTM, first we obtain a word representation $\mathbf{u}_i$ for word w_i as,

$$\mathbf{u}_i = [\text{cLSTM}(c_1 \ldots c_n); \text{cLSTM}(c_n \ldots c_1)] \quad (2)$$

where $c_1, \ldots, c_n$ is the character sequence of the word. This representation $\mathbf{u}_i$ is then input to a word-level biLSTM tagger. The word-level biLSTM tagger predicts a tag from $\mathcal{Y}$. A full description of the model is found in Heigold et al. (2017). We use standard cross-entropy loss for training this model and decode greedily while predicting the tags during test-time. Note that greedy decoding is optimal in this tagger as there is no interdependence between the tags m_i.

3.2 A Lemmatizer: $p(\ell_i \mid m_i, w_i)$

Neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) have yielded state-of-the-art performance on the task of generating morphological variants—including the lemma—as evinced in several recent shared tasks on the subject (Cotterell et al., 2016, 2017, 2018). Our lemmatization factor in eq. (1) is based on such models. Specifically, we make use of a hard-attention mechanism (Xu et al., 2015; Rastogi et al., 2016), rather than the original soft-attention mechanism. Our choice of hard attention is motivated by the performance of Makarov and Clematide (2018)'s system at the CoNLL-SIGMORPHON task. We use a nearly identical model, but opt for an exact dynamic-programming-based inference scheme (Wu et al., 2018).[4]

We briefly describe the model here. Given an inflected word w and a tag m, we would like to obtain the lemma $\ell \in \Sigma^*$, dropping the subscript for simplicity. Moreover, for the remainder of *this* section the subscripts will index into the character string ℓ, that is $\ell = \ell_1, \ldots, \ell_{|\ell|}$, where each $\ell_i \in \Sigma$. A character-level biLSTM encoder embeds w to $\mathbf{h}^{(enc)}$. The decoder LSTM produces $\mathbf{h}^{(dec)}_j$, reading the concatenation of the embedding of the previous character $\ell_{j-1} \in \Sigma$ and the tag embedding $\mathbf{h}^{(tag)}$, which is produced by an order-invariant linear function. In contrast to soft attention, hard attention models the alignment distribution explicitly.

We denote $A = \{a_1, \ldots, a_{|w|}\}^{|\ell|}$ as the set of all monotonic alignments from w to ℓ where an alignment aligns each target character ℓ_j to exactly one source character in w and for $\mathbf{a} \in A$, $a_j = i$

---

[4] Our formulation differs from the work of Wu et al. (2018) in that we enforce monotonic hard alignments, rather than allow for non-monotonic alignments.

refers to the event that the j^{th} character of ℓ is aligned to the i^{th} character of w. We factor the probabilistic lemmatizer as,

$$p(\ell \mid m, w) = \sum_{\mathbf{a} \in A} p(\ell, \mathbf{a} \mid m, w) \tag{3}$$

$$= \sum_{\mathbf{a} \in A} \prod_{j=1}^{|\ell|} p(\ell_j \mid a_j, \ell_{<j}, m, w) \tag{4}$$
$$p(a_j \mid a_{j-1}, \ell_{<j}, m, w)$$

$$= \sum_{\mathbf{a} \in A} \prod_{j=1}^{|\ell|} p(\ell_j \mid \mathbf{h}^{(enc)}_{a_j}, \mathbf{h}^{(dec)}_j) \tag{5}$$
$$p(a_j \mid a_{j-1}, \mathbf{h}^{(enc)}, \mathbf{h}^{(dec)}_j)$$

The summation is computed with dynamic programming—specifically, using the forward algorithm for hidden Markov models (Rabiner, 1989). $p(\ell_j \mid \mathbf{h}^{(enc)}_{a_j}, \mathbf{h}^{(dec)}_j)$ is a two-layer feed-forward network followed by a softmax. The transition $p(a_j \mid a_{j-1}, \mathbf{h}^{(enc)}, \mathbf{h}^{(dec)}_j)$ is the multiplicative attention function with $\mathbf{h}^{(enc)}$ and $\mathbf{h}^{(dec)}_j$ as input. To enforce monotonicity, $p(a_j \mid a_{j-1}) = 0$ if $a_j < a_{j-1}$.

3.3 Decoding

We consider two manners, by which we decode our model. The first is a greedy decoding scheme. The second is a **crunching** (May and Knight, 2006) scheme. We describe each in turn.

Greedy Decoding. In the greedy scheme, we select the best morphological tag sequence

$$\mathbf{m}^\star = \text{argmax}_{\mathbf{m}} \log p(\mathbf{m} \mid \mathbf{w}) \tag{6}$$

and then decode each lemmata

$$\ell_i^\star = \text{argmax}_\ell \log p(\ell \mid m_i^\star, w_i) \tag{7}$$

Note that we slightly abuse notation since the argmax here is *approximate*: exact decoding of our neural lemmatizer is hard. This sort of scheme is also referred to as pipeline decoding.

Crunching. In the crunching scheme, we first extract a k-best list of taggings from the morphological tagger. For an input sentence $\mathbf{w}$, call the k-best tags for the i^{th} word $\mathcal{K}(w_i)$. Crunching then says we should decode in the following manner

$$\ell_i^\star = \text{argmax}_\ell \tag{8}$$
$$\log \sum_{m_i \in \mathcal{K}(w_i)} p(\ell \mid m_i, w_i)\, p(m_i \mid \mathbf{w})$$

Crunching is a tractable heuristic that approximates true joint decoding[5] and, as such, we expect it to outperform the more naïve greedy approach.

3.4 Training with Jackknifing

In our model, a simple application of maximum-likelihood estimation (MLE) is unlikely to work well. The reason is that our model is a discriminative directed graphical model (as seen in Figure 1) and, thus, suffers from **exposure bias** (Ranzato et al., 2015). The intuition behind the poor performance of MLE is simple: the output of the lemmatizer depends on the output of the morphological tagger; as the lemmatizer has only ever seen correct morphological tags, it has never learned to adjust for the errors that will be made at the time of decoding. To compensate for this, we employ **jackknifing** (Agić and Schluter, 2017), which is standard practice in many NLP pipelines, such as dependency parsing.

Jackknifing for training NLP pipelines is quite similar to the oft-employed cross-validation. We divide our training data into κ splits. Then, for each split $i \in \{1, \ldots, \kappa\}$, we train the morphological tagger on the i^{th} split, and then decode it, using either greedy decoding or crunching, on the remaining $(\kappa - 1)$ splits. This technique helps avoid exposure bias and improves the lemmatization performance, which we will demonstrate empirically in §4. Indeed, the model is quite ineffective without this training regime. Note that we employ jackknifing for both the greedy decoding scheme and the crunching decoding scheme.

4 Experimental Setup

4.1 Dataset

To enable a fair comparison with Bergmanis and Goldwater (2018), we use the Universal Dependencies Treebanks (Nivre et al., 2017) for all our experiments. Following previous work, we use v2.0 of the treebanks for all languages, except Dutch, for which v2.1 was used due to inconsistencies in v2.0. The standard splits are used for all treebanks.

---

[5] True joint decoding would sum over all possible morphological tags, rather than just the k-best. While this is tractable in our setting in the sense that there are, at, most, 1662 morphological tags (in the case of Basque), it is significantly slower than using a smaller k. Indeed, the probability distribution that morphological taggers learn tend to be peaked to the point that considering improbable tags is not necessary.

4.2 Training Setup and Hyperparameters

For the morphological tagger, we use the baseline implementation from Malaviya et al. (2018). This implementation uses an input layer and linear layer dimension of 128 and a 2-layer LSTM with a hidden layer dimension of 256. The Adam (Kingma and Ba, 2014) optimizer is used for training and a dropout rate (Srivastava et al., 2014) of 0.3 is enforced during training. The tagger was trained for 10 epochs.

For the lemmatizer, we use a 2-layer biLSTM encoder and a 1-layer LSTM decoder with 400 hidden units. The dimensions of character and tag embedding are 200 and 40, respectively. We enforce a dropout rate of 0.4 in the embedding and encoder LSTM layers. The lemmatizer is also trained with Adam and the learning rate is 0.001. We halve the learning rate whenever the development log-likelihood increases and we perform early-stopping when the learning rate reaches 1×10^{-5}. We apply gradient clipping with a maximum gradient norm of 5.

4.3 Baselines (and Related Work)

We compare our approach against recent competing methods that report results on UD datasets.

Lematus. The current state of the art is held by Bergmanis and Goldwater (2018), who, as discussed in §1, provide a direct context-to-lemma approach, avoiding the use of morphological tags. We remark that Bergmanis and Goldwater (2018) assume a setting where lemmata are annotated at the token level, but morphological tags are not available; we contend, however, that such a setting is not entirely realistic as almost all corpora annotated with lemmata at the token level include morpho-syntactic annotation, including the vast majority of the UD corpora. Thus, we do not consider it a stretch to assume the annotation of morphological tags to train our joint model.[6]

UDPipe. Our next baseline is the UDPipe system of Straka and Straková (2017). Their system performs lemmatization using an averaged perceptron tagger that predicts a (lemma rule, UPOS) pair. Here, a lemma rule generates a lemma by removing parts of the word prefix/suffix and prepending and appending a new prefix/suffix. A guesser first produces correct lemma rules and the tagger is used to disambiguate from them.

Lemming. The strongest non-neural baseline we consider is the system of Müller et al. (2015), who, like us, develop a joint model of morphological tagging lemmatization. In contrast to us, however, their model is globally normalized (Lafferty et al., 2001). Due to their global normalization, they directly estimate the parameters of their model with MLE without worrying about exposure bias. However, in order to efficiently normalize the model, they heuristically limit the set of possible lemmata through the use of **edit trees** (Chrupała, 2008), which makes the computation of the partition function tractable.

Morfette. Much like Müller et al. (2015), Morfette relies on the concept of edit trees. However, a simple perceptron is used for classification with hand-crafted features. A full description of the model is given in Chrupała et al. (2008).

5 Results and Discussion

Experimentally, we aim to show three points. i) Our joint model (eq. (1)) of morphological tagging and lemmatization achieves state-of-the-art accuracy; this builds on the findings of Bergmanis and Goldwater (2018), who show that context significantly helps neural lemmatization. Moreover, the upper bound for contextual lemmatizers that make use of morphological tags is much higher, indicating room for improved lemmatization with better morphological taggers. ii) We discuss a number of error patterns that the model seems to make on the languages, where absolute accuracy is lowest: Latvian, Estonian and Arabic. We suggest possible paths forward to improve performance. iii) We offer an explanation for *when* our joint model does better than the context-to-lemma baseline. We show through a correlational study that our joint approach with morphological tagging helps the most in two cases: low-resource languages and morphologically rich languages.

---

[6] After correspondence with Toms Bergmanis, we would like to clarify this point. While Bergmanis and Goldwater (2018) explores the model in a token-annotated setting, as do we, the authors argue that such a model is better for a very low-resource scenario where the entire sentence is not annotated for lemmata. We concede this point—our current model is not applicable in such a setting. However, we note that a semi-supervised morphological tagger could be trained in such a situation as well, which may benefit lemmatization.

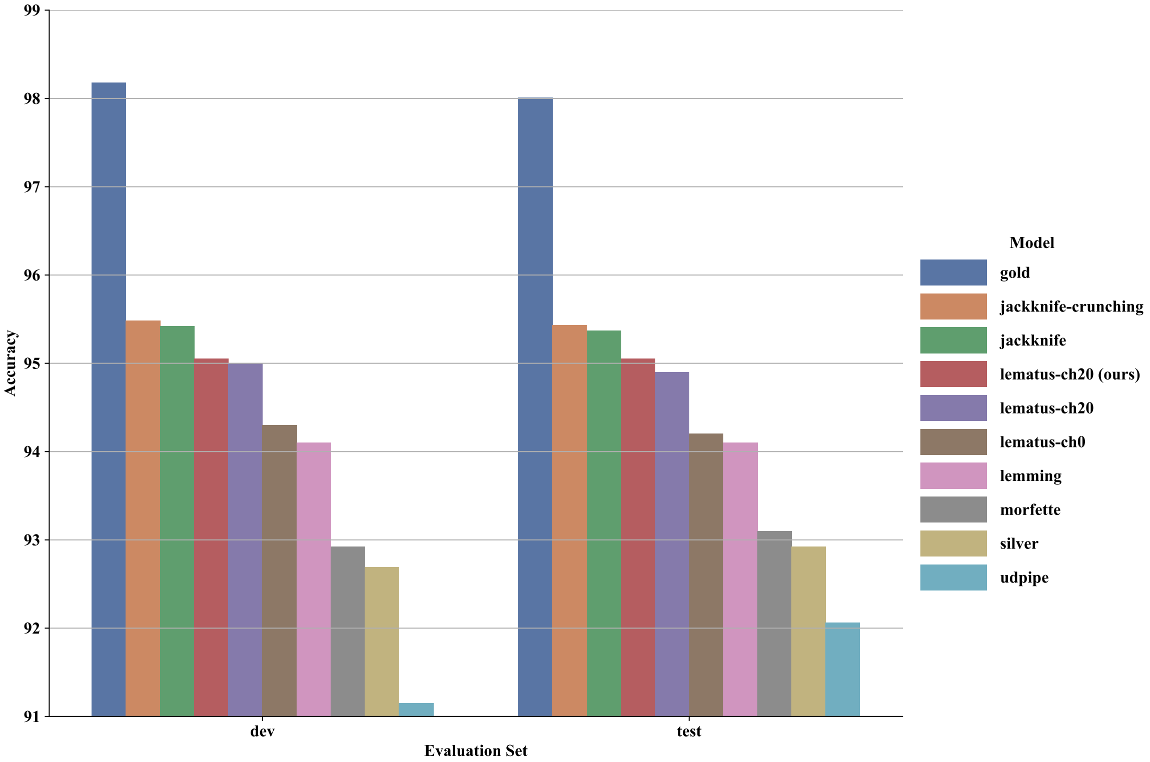


Figure 3: We present performance (in accuracy) averaged over the 20 languages from UD we consider. Our method (second from the left) significantly outperforms the strongest baseline (fourth from the left; Bergmanis and Goldwater (2018)). The blue column is a skyline that gives our model gold tags during decoding, showing improved tagging should lead to better lemmatization. The remaining are baselines described in §4.3.

5.1 Main Results

The first experiment we run focuses on pure performance of the model. Our goal is to determine whether joint morphological tagging and lemmatization improves average performance in a state-of-the-art neural model.

Evaluation Metrics. For measuring lemmatization performance, we measure the accuracy of guessing the lemmata correctly over an entire corpus. To demonstrate the effectiveness of our model in utilizing context and generalizing to unseen word forms, we follow Bergmanis and Goldwater (2018) and also report accuracies on tokens that are i) **ambiguous**, i.e., more than one lemmata exist for the same inflected form, ii) **unseen**, i.e., where the inflected form has not been seen in the training set, and iii) **seen unambiguous**, i.e., where the inflected form has only one lemma and is seen in the training set.

Results. The results showing comparisons with all other methods are summarized in Figure 3. Each bar represents the average accuracy across 20 languages. Our method achieves an average accuracy of 95.42 and the strongest baseline, Bergmanis and Goldwater (2018), achieves an average accuracy of 95.05. The difference in performance (0.37) is statistically significant with $p < 0.01$ under a paired permutation test. We outperform the strongest baseline in 11 out of 20 languages and underperform in only 3 languages with $p < 0.05$. The difference between our method and all other baselines is statistical significant with $p < 0.001$ in all cases. We highlight two additional features of the data. First, decoding using gold morphological tags gives an accuracy of 98.04 for a difference in performance of $+2.62$. We take the large difference between the upper bound and the current performance of our model to indicate that improved morphological tagging is likely to significantly help lemmatization. Second, it is noteworthy that training with gold tags, but decoding with predicted tags, yields performance that is significantly worse than every baseline except for UDPipe. This speaks for the importance of jackknifing in the training of joint morphological tagger-lemmatizers that are directed and, therefore, suffer from exposure bias.

In Figure 4, we observed crunching further im-

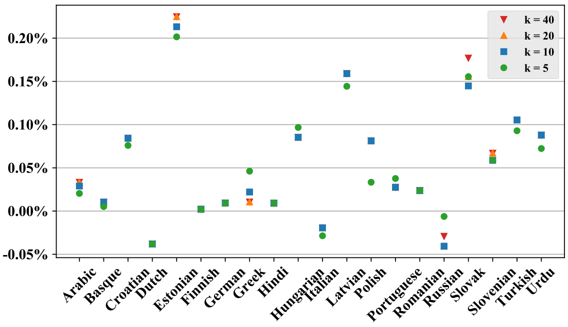


Figure 4: Relative improvement on validation set with crunching over greedy decoding for different values of k.

proves performance of the greedy decoding scheme. In 8 out of 20 languages, the improvement is statistical significant with $p < 0.05$. We select the best k for each language based on the development set.

In Figure 5, we provide a language-wise breakdown of the performance of our model and the model of Bergmanis and Goldwater (2018). Our strongest improvements are seen in Latvian, Greek and Hungarian. When measuring performance solely over unseen inflected forms, we achieve even stronger gains over the baseline method in most languages. This demonstrates the generalization power of our model beyond word forms seen in the training set. In addition, our accuracies on ambiguous tokens are also seen to be higher than the baseline on average, with strong improvements on highly inflected languages such as Latvian and Russian. Finally, on seen unambiguous tokens, we note improvements that are similar across all languages.

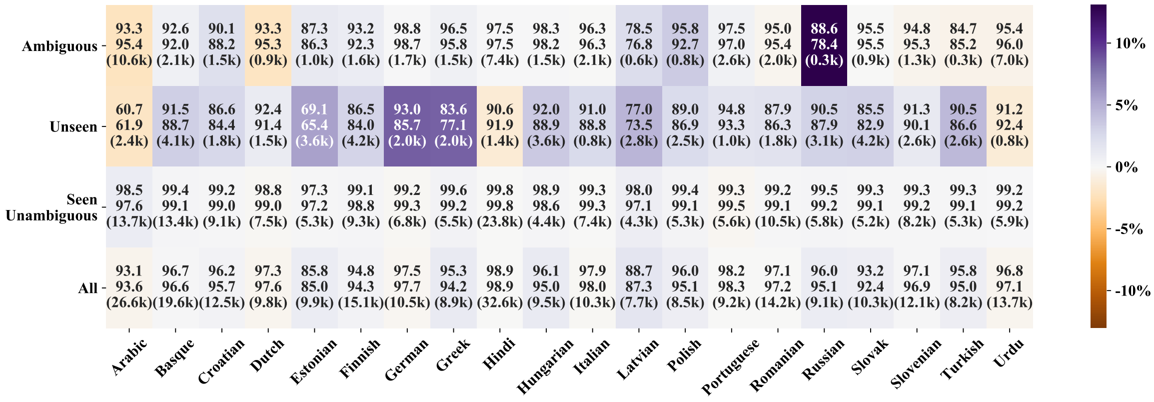
5.2 Error Patterns

We attempt to identify systematic error patterns of our model in an effort to motivate future work. For this analysis, we compare predictions of our model and the gold lemmata on three languages with the weakest absolute performance: Estonian, Latvian and Arabic. First, we note the differences in the average lengths of gold lemmata in the tokens we guess incorrectly and all the tokens in the corpus. The lemmata we guess incorrectly are on average 1.04 characters longer than the average length of all the lemmata in the corpus. We found that the length of the incorrect lemmata does not correlate strongly with their frequency. Next, we identify the most common set of edit operations in each language that would transform the incorrect hypothesis to the gold lemma. This set of edit operations was found to follow a power-law distribution.

lang	# tokens	# tags	ours	Lematus	Δ
arabic	202000	349	93.1	93.55	-0.48
basque	59700	884	96.74	96.55	0.2
croatian	146000	1105	96.16	95.7	0.48
dutch	163000	62	97.26	97.65	-0.4
estonian	17000	482	85.83	84.99	0.99
finnish	135000	1669	94.79	94.31	0.51
german	227000	683	97.46	97.72	-0.26
greek	36500	346	95.29	94.22	1.13
hindi	261000	939	98.88	98.92	-0.05
hungarian	16700	580	96.13	94.99	1.2
italian	236000	278	97.93	98.04	-0.11
latvian	28000	640	88.67	87.31	1.56
polish	52000	991	95.99	95.12	0.91
portuguese	176000	375	98.2	98.26	-0.06
romanian	157000	451	97.11	97.19	-0.08
russian	58400	715	96.0	95.07	0.98
slovak	64800	1186	93.25	92.43	0.89
slovenian	96500	1101	97.07	96.9	0.17
turkish	31200	972	95.81	95.01	0.85
urdu	101000	1001	96.76	97.12	-0.37

Table 1: Here we present the number of tokens in each of the UD treebanks we use as well as the number of morphological tags. Note, we take the number of tags as a proxy for the morphological complexity of the language. Finally, we present numbers on validation set from our method with greedy decoding and from the strongest baseline (Lematus) as well as the difference. Correlations between the first two columns and the differences are shown in Table 2.

For the case of Latvian, we find that the operation {*replace*: s → a} is the most common error made by our model. This operation corresponds to a possible issue in the Latvian treebank, where adjectives were marked with gendered lemmas. This issue has now been resolved in the latest version of the treebank. For Estonian, the operation {*insert*: m, *insert*: a} is the most common error. The suffix *-ma* in Estonian is used to indicate the infinitive form of verbs. Gold lemmata for verbs in Estonian are marked in their infinitive forms whereas our system predicts the stems of these verbs instead. These inflected forms are usually ambiguous and we believe that the model doesn't generalize well to different form-lemma pairs, partly due to fewer training data available for Estonian. This is an example of an error pattern that could be corrected using improved morphological information about the tokens. Finally, in Arabic, we find that the most common error pattern corresponds to a single ambiguous word form, *'an* , which can be lemmatized as *'anna* (like "that" in English) or *'an* (like "to" in English) depending on the usage of the word in



	Arabic	Basque	Croatian	Dutch	Estonian	Finnish	German	Greek	Hindi	Hungarian	Italian	Latvian	Polish	Portuguese	Romanian	Russian	Slovak	Slovenian	Turkish	Urdu
Ambiguous	93.3 95.4 (10.6k)	92.6 92.0 (2.1k)	90.1 88.2 (1.5k)	93.3 95.3 (0.9k)	87.3 86.3 (1.0k)	93.2 92.3 (1.6k)	98.8 98.7 (1.7k)	96.5 95.8 (1.5k)	97.5 97.5 (7.4k)	98.3 98.2 (1.5k)	96.3 96.3 (2.1k)	78.5 76.8 (0.6k)	95.8 92.7 (0.8k)	97.5 97.0 (2.6k)	95.0 95.4 (2.0k)	88.6 78.4 (0.3k)	95.5 95.5 (0.9k)	94.8 95.3 (1.3k)	84.7 85.2 (0.3k)	95.4 96.0 (7.0k)
Unseen	60.7 61.9 (2.4k)	91.5 88.7 (4.1k)	86.6 84.4 (1.8k)	92.4 91.4 (1.5k)	69.1 65.4 (3.6k)	86.5 84.0 (4.2k)	93.0 85.7 (2.0k)	83.6 77.1 (2.0k)	90.6 91.9 (1.4k)	92.0 88.9 (3.6k)	91.0 88.8 (0.8k)	77.0 73.5 (2.8k)	89.0 86.9 (2.5k)	94.8 93.3 (1.0k)	87.9 86.3 (1.8k)	90.5 87.9 (3.1k)	85.5 82.9 (4.2k)	91.3 90.1 (2.6k)	90.5 86.6 (2.6k)	91.2 92.4 (0.8k)
Seen Unambiguous	98.5 97.6 (13.7k)	99.4 99.1 (13.4k)	99.2 99.0 (9.1k)	98.8 99.0 (7.5k)	97.3 97.2 (5.3k)	99.1 98.8 (9.3k)	99.2 99.3 (6.8k)	99.6 99.2 (5.5k)	99.8 99.8 (23.8k)	98.9 98.6 (4.4k)	99.3 99.3 (7.4k)	98.0 97.1 (4.3k)	99.4 99.1 (5.3k)	99.3 99.5 (5.6k)	99.2 99.1 (10.5k)	99.5 99.2 (5.8k)	99.3 99.1 (5.2k)	99.3 99.2 (8.2k)	99.3 99.1 (5.3k)	99.2 99.2 (5.9k)
All	93.1 93.6 (26.6k)	96.7 96.6 (19.6k)	96.2 95.7 (12.5k)	97.3 97.6 (9.8k)	85.8 85.0 (9.9k)	94.8 94.3 (15.1k)	97.5 97.7 (10.5k)	95.3 94.2 (8.9k)	98.9 98.9 (32.6k)	96.1 95.0 (9.5k)	97.9 98.0 (10.3k)	88.7 87.3 (7.7k)	96.0 95.1 (8.5k)	98.2 98.3 (9.2k)	97.1 97.2 (14.2k)	96.0 95.1 (9.1k)	93.2 92.4 (10.3k)	97.1 96.9 (12.1k)	95.8 95.0 (8.2k)	96.8 97.1 (13.7k)

Figure 5: Dev accuracy breakdown by type of inflected form on all languages comparing our system with greedy decoding against our run of Lematus-ch20, colored by relative improvement in percentage. In each entry, the bottom score is from Lematus-ch20 and the top one is from our system, and the number in the parenthesis is the number of tokens for the corresponding setting.

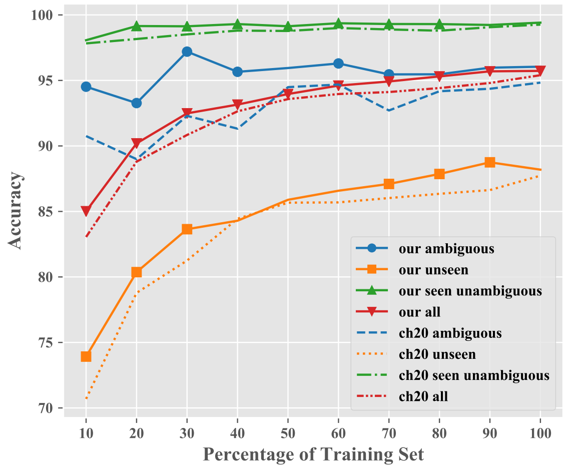


Figure 6: Learning curve showing the accuracy on the validation set of the Polish treebank as the percentage of training set is increased. Markers indicate statistically significant better system with paired permutation test ($p < 0.05$). Our model is decoded greedily.

context. The word *'anna* must be followed by a nominal sentence while *'an* is followed by a verb. Hence, models that can incorporate rich contextual information would be able to avoid such errors.

5.3 Why our model performs better?

Simply presenting improved results does not entirely satiate our curiosity: we would also like to understand *why* our model performs better. Specifically, we have assumed an additional level of supervision—namely, the annotation of morphological tags. We provide the differences between our method and our retraining of the Lematus system presented in Table 1. In addition to the performance of the systems, we also list the number of tokens in each treebank and the number of distinct morphological tags per language. We perform a correlational study, which is shown in Table 2.

	Pearson's Rv	Spearman's ρ
# tags vs. Δ	0.206	0.209
# tokens vs. Δ	-0.808	-0.845

Table 2: The table shows the correlations between the differences in dev performance between our model with greedy decoding and Lematus and two aspects of the data: number of tokens and number of tags.

Morphological Complexity and Performance. We see that there is a moderate positive correlation ($\rho = 0.209$) between the number of morphological tags in a language and the improvement our model obtains. As we take the number of tags as a proxy for the morphological complexity in the language, we view this as an indication that attempting to directly extract the relevant morpho-syntactic information from the corpus is not as effective when there is more to learn. In such languages, we recommend exploiting the additional annotation to achieve better results.

Amount of Data and Performance. The second correlation we find is a stronger negative correlation ($\rho = -0.845$) between the number of tokens available for training in the treebank and the gains in performance of our model over the baseline. This is further demonstrated by the learning curve plot in Figure 6, where we plot the validation accuracy on the Polish treebank for different sizes of the training set. The gap between the performance of our model and Lematus-ch20 is larger when fewer training data are available, especially for ambiguous tokens. This indicates that the incorporation of morphological tags into a model helps more in the low-resource setting. Indeed, this conclusion makes sense—neural networks are good at extracting features from text when there is a sufficiently large amount of data. However, in the low-resource case, we would expect direct supervision on the sort of features we desire to extract to work better. Thus, our second recommendation is to model tags jointly with lemmata when fewer training tokens are available. As we noted earlier, it is almost always the case that token-level annotation of lemmata comes with token-level annotation of morphological tags. In low-resource scenarios, a data augmentation approach such as the one proposed by Bergmanis and Goldwater (2019) can be helpful and serve complementary to our approach.

6 Conclusion

We have presented a simple joint model for morphological tagging and lemmatization and discussed techniques for training and decoding. Empirically, we have shown that our model achieves state-of-the-art results, hinting that explicitly modeling morphological tags is a more effective manner for modeling context. In addition to strong numbers, we tried to explain *when* and *why* our model does better. Specifically, we show a significant correlation between our scores and the number of tokens and tags present in a treebank. We take this to indicate that our method improves performance more for low-resource languages as well as morphologically rich languages.

Acknowledgments

We thank Toms Bergmanis for his detailed feedback on the accepted version of the manuscript. Additionally, we would like to thank the three anonymous reviewers for their valuable suggestions. The last author would like to acknowledge support from a Facebook Fellowship.

References

Željko Agić and Natalie Schluter. 2017. How (not) to train a dependency parser: The curious case of jackknifing part-of-speech taggers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 679–684. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with Lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1391–1400. Association for Computational Linguistics.

Toms Bergmanis and Sharon Goldwater. 2019. Data Augmentation for Context-Sensitive Neural Lemmatization Using Inflection Tables and Raw Text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). Http://www.lrec-conf.org/proceedings/lrec2008/.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gèraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the*

CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, pages 1–27. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gèraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in smt. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674. Association for Computational Linguistics.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology*. Routledge.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 505–513. Association for Computational Linguistics.

Jakub Kanis and Lucie Skorkovská. 2010. Comparison of different lemmatization approaches through the means of information retrieval performance. In *International Conference on Text, Speech and Dialogue*, pages 93–100. Springer.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: Principles and techniques*. MIT Press.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Peter Makarov and Simon Clematide. 2018. UZH at CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 69–75. Association for Computational Linguistics.

Chaitanya Malaviya, Matthew R. Gormley, and Graham Neubig. 2018. Neural factor graph models for cross-lingual morphological tagging. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2653–2663. Association for Computational Linguistics.

Jonathan May and Kevin Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*.

Arya D McCarthy, Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2018. Marrying universal dependencies and universal morphology. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 91–101.

Christof Monz and Maarten De Rijke. 2001. Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 262–277. Springer.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274. Association for Computational Linguistics.

Joakim Nivre et al. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference* *of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21.

A Additional Results

We present the exact numbers on all languages to allow future papers to compare to our results in Table 3 and Table 4.

	Gold	Crunching	Jackknifing	Ch-20	Silver
Arabic	97.40	93.13	93.10	93.55	89.32
Basque	98.70	96.75	96.74	96.55	94.88
Croatian	98.61	96.24	96.16	95.70	94.89
Dutch	98.82	97.26	97.26	97.65	96.31
Estonian	91.92	86.02	85.83	84.99	71.65
Finnish	97.01	94.79	94.79	94.31	90.89
German	97.74	97.47	97.46	97.72	96.27
Greek	97.40	95.33	95.29	94.22	94.33
Hindi	99.29	98.89	98.88	98.92	98.63
Hungarian	98.54	96.22	96.13	94.99	94.15
Italian	99.37	97.93	97.93	98.04	96.95
Latvian	97.24	88.81	88.67	87.31	85.58
Polish	98.46	96.07	95.99	95.12	91.77
Portuguese	99.63	98.24	98.20	98.26	97.72
Romanian	99.34	97.13	97.11	97.19	95.99
Russian	98.46	96.00	96.00	95.07	93.84
Slovak	97.14	93.41	93.25	92.43	88.49
Slovenian	99.46	97.13	97.07	96.90	95.64
Turkish	98.71	95.91	95.81	95.01	91.25
Urdu	97.48	96.84	96.76	97.12	96.62
AVERAGE	98.04	95.48	95.42	95.05	92.76

Table 3: Development performance breakdown.

	Gold	Crunching	Jackknifing	Ch-20	Silver
Arabic	97.95	93.99	93.92	94.16	91.37
Basque	98.54	96.63	96.67	96.49	94.57
Croatian	98.24	95.63	95.58	95.22	94.28
Dutch	98.43	97.25	97.25	97.21	96.50
Estonian	92.34	86.33	86.13	85.44	73.41
Finnish	97.02	94.34	94.29	93.94	90.57
German	97.39	97.14	97.07	97.63	95.88
Greek	97.83	96.53	96.46	95.32	95.05
Hindi	99.10	98.68	98.65	98.73	98.47
Hungarian	97.72	94.02	93.96	93.15	92.42
Italian	99.33	97.83	97.83	98.05	96.96
Latvian	96.69	89.79	89.76	88.87	86.49
Polish	98.45	95.74	95.78	94.90	91.94
Portuguese	99.60	97.97	97.86	98.14	97.58
Romanian	99.55	97.21	97.14	97.21	96.36
Russian	98.30	95.82	95.82	94.77	93.71
Slovak	97.40	93.46	93.31	92.29	88.63
Slovenian	99.25	96.74	96.66	96.69	95.42
Turkish	99.18	96.48	96.32	95.99	92.14
Urdu	97.87	96.94	96.91	96.77	96.73
AVERAGE	98.01	95.43	95.37	95.05	92.92

Table 4: Test performance breakdown.

	F1 Score
Arabic	85.62
Basque	83.68
Croatian	85.37
Dutch	90.92
Estonian	65.80
Finnish	87.94
German	79.45
Greek	87.63
Hindi	87.89
Hungarian	86.00
Italian	93.78
Latvian	80.96
Polish	80.29
Portuguese	93.65
Romanian	93.51
Russian	83.69
Slovak	64.53
Slovenian	88.81
Turkish	82.60
Urdu	72.86
AVERAGE	83.75

Table 5: Morphological Tagging Performance on development set.

A Probabilistic Generative Model of Linguistic Typology

Johannes Bjerva[ɘ] **Yova Kementchedjhieva**[ɘ] **Ryan Cotterell**[ʔ,ﬁ] **Isabelle Augenstein**[ɘ]
[ɘ]Department of Computer Science, University of Copenhagen
[ʔ]Department of Computer Science, Johns Hopkins University
[ﬁ]Department of Computer Science and Technology, University of Cambridge
bjerva,yova,augenstein@di.ku.dk, rdc42@cam.ac.uk

Abstract

In the principles-and-parameters framework, the structural features of languages depend on parameters that may be toggled on or off, with a single parameter often dictating the status of multiple features. The implied covariance between features inspires our probabilisation of this line of linguistic inquiry—we develop a generative model of language based on exponential-family matrix factorisation. By modelling all languages and features within the same architecture, we show how structural similarities between languages can be exploited to predict typological features with near-perfect accuracy, outperforming several baselines on the task of predicting held-out features. Furthermore, we show that language embeddings pre-trained on monolingual text allow for generalisation to unobserved languages. This finding has clear practical and also theoretical implications: the results confirm what linguists have hypothesised, i.e. that there are significant correlations between typological features and languages.

1 Introduction

Linguistic typologists dissect and analyse languages in terms of their structural properties (Croft, 2002). For instance, consider the phonological property of word-final obstruent decoding: German devoices word-final obstruents (*Zug* is pronounced /zuk/), whereas English does not (*dog* is pronounced /dɒg/). In the tradition of generative linguistics, one line of typological analysis is the principles-and-parameters framework (Chomsky, 1981), which posits the existence of a set of universal parameters, switches as it were, that languages toggle. One arrives at a kind of factorial typology, to borrow terminology from optimality theory (Prince and Smolensky, 2008), through different settings of the parameters. Within the principle-and-parameters research program, then, the goal is to identify the parameters that serve as axes, along which languages may vary.

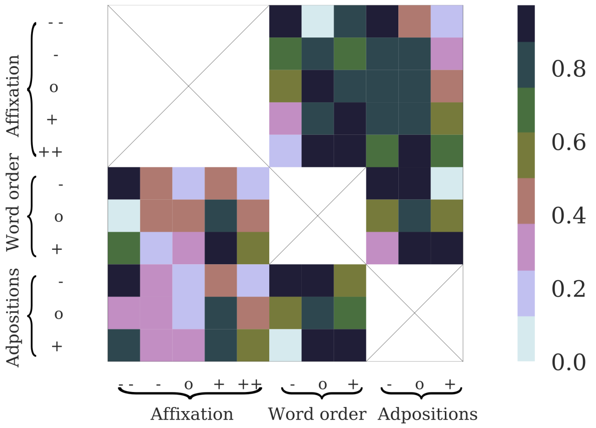


Figure 1: Correlations between selected typological parameters. Feature values are classified according to head-directionality (head-initial **+**, head-final **-**, no dominant order **o**). For instance, **++** under Affixation means strongly suffixing.

It is not enough, however, to simply write down the set of parameters available to language. Indeed, one of the most interesting facets of typology is that different parameters are correlated. To illustrate this point, we show a heatmap in Figure 1 that shows the correlation between the values of selected parameters taken from a typological knowledge base (KB). Notice how head-final word order, for example, highly correlates with strong suffixation. The primary contribution of this work is a probabilisation of typology inspired by the principles-and-parameters framework. We assume a given set of typological parameters and develop a generative model of a language's parameters, casting the problem as a form of exponential-family matrix factorisation. We observe a binary matrix that encodes the settings of each parameter for each language. For example, the Manchu head-final entry of this matrix would be set to 1, because Manchu

Proceedings of NAACL-HLT 2019, pages 1529–1540
Minneapolis, Minnesota, June 2 - June 7, 2019.

is a head-final language. The goal of our model is to *explain* each entry of matrix as arising through the dot product of a language embedding and a parameter embedding passed through a sigmoid.

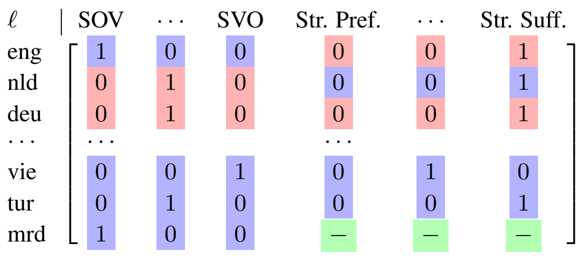
We test our model on The World Atlas of Language Structures (WALS), the largest available knowledge base of typological parameters at the lexical, phonological, syntactic and semantic level. Our contributions are: (i) We develop a probabilisation of typology inspired by the principles-and-parameters framework. (ii) We introduce the novel task of typological collaborative filtering, where we observe some of a language's parameters, but hold some out. At evaluation time, we predict the held-out parameters using the generative model. (iii) We develop a semi-supervised extension, in which we incorporate language embeddings output by a neural language model, thus improving performance with unlabelled data. Indeed, when we partially observe some of the typological parameters of a language, we achieve near-perfect (97%) accuracy on the prediction of held-out parameters. (iv) We perform an extensive qualitative and quantitative analysis of our method.

2 Typology in Generative Grammar

What we will present in this paper is a generative model that corresponds to a generative tradition of research in linguistic typology. We first outline the technical linguistic background necessary for the model's exposition. Chomsky famously argued that the human brain contains a prior, as it were, over possible linguistic structures, which he termed **universal grammar** (Chomsky, 1965). The connection between Chomsky's Universal Grammar and the Bayesian prior is an intuitive one, but the earliest citation we know for the connection is Eisner (2002, §2). As a theory, universal grammar holds great promise in explaining the typological variation of human language. Cross-linguistic similarities and differences may be explained by the influence universal grammar exerts over language acquisition and change. While universal grammar arose early on in the writtings of Chomsky, early work in generative grammar focused primarily on English (Harris, 1995). Indeed, Chomsky's *Syntactic Structures* contains exclusively examples in English (Chomsky, 1957). As the generative grammarians turned their focus to a wider selection of languages, the **principles and parameters** framework for syntactic analysis rose to prominence. Given the tight relationship between the theory of universal grammar and typology, principles and parameters offers a fruitful manner in which to research typological variation.

The principles and parameters takes a parametric view of linguistic typology. The structure of human language is governed by a series of **principles**, which are hard constraints on human language. A common example of a principle is the requirement that every sentence has a subject, even if one that is not pronounced; see the discussion on the pro-drop parameter in Carnie (2013). Principles are universally true for all languages. On the other hand, languages are also governed by **parameters**. Unlike principles, parameters are the parts of linguistic structure that are allowed to vary. It is useful to think of parameters as attributes that can take on a variety of values. As Chomsky (2000) himself writes "we can think of the initial state of the faculty of language as a fixed network connected to a switch box; the network is constituted of the principles of language, while the switches are the options to be determined by experience. When the switches are set one way, we have Swahili; when they are set another way, we have Japanese. Each possible human language is identified as a particular setting of the switches-a setting of parameters, in technical terminology."

What are possible parameters? Here, in our formalisation of the parameter aspect of the principles-and-parameters framework, we take a catholic view of parameters, encompassing all areas of linguistics, rather than just syntax. For example, as we saw before, consider the switch of devoicing word-final obstruents a parameter. We note that while principle-and-parameters typology has primarily been applied to syntax, there are also interesting applications to non-syntactic domains. For instance, van Oostendorp (2015) applies a parametric approach to metrical stress in phonology; this is in line with our view. In the field of linguistic typology, there is a vibrant line of research, which fits into the tradition of viewing typological parameters through the lens of principles and parameters. Indeed, while earlier work due to Chomsky focused on what have come to be called macro-parameters, many linguists now focus on micro-parameters, which are very close to the features found in the WALS dataset that we will be modelling (Baker, 2008; Nicolis and Biberauer, 2008; Biberauer et al., 2009). This justifies our viewing



ℓ	SOV	$\cdots$	SVO	Str. Pref.	$\cdots$	Str. Suff.
eng	1	0	0	0	0	1
nld	0	1	0	0	0	1
deu	0	1	0	0	0	1
$\cdots$	$\cdots$			$\cdots$		
vie	0	0	1	0	1	0
tur	0	1	0	0	0	1
mrd	1	0	0	–	–	–

Figure 2: Example of training and evaluation portions of a feature matrix, with word order features (81A, SOV, $\cdots$, SVO) and affixation features (26A, Strongly Prefixing, $\cdots$, Strongly Suffixing). We train on the examples highlighted in blue, evaluate on the examples highlighted in red, and ignore those which are not covered in WALS highlighted in green.

WALS through the lens of principles and parameters, even though the authors of WALS adher to the functional-typological school.[1]

Notationally, we will represent the parameters as a vector $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_n]$. Each typological parameter π_i is a binary variable; for instance, does the language admit word-final voiced obstruents?

3 A Generative Model of Typology

We now seek a probabilistic formalisation of the linguistic theory presented in §2; specifically, for every language ℓ, we seek to explain the observed binary vector of parameters $\boldsymbol{\pi}^\ell$: $\pi_i^\ell = 1$ indicates that the i^{th} parameter is "on" in language ℓ. The heart of our model will be quite simple: every language ℓ will have a language embedding $\boldsymbol{\lambda}^\ell \in \mathbb{R}^d$ and every parameter will have a parameter embedding $\mathbf{e}_{\pi_i} \in \mathbb{R}^d$. Now, $\pi_i^\ell \sim \textit{sigmoid}(\mathbf{e}_{\pi_i}^\top \boldsymbol{\lambda}^\ell)$.

This model also takes inspiration from work in relation extraction (Riedel et al., 2013). Writing the joint distribution over the entire binary vector of parameters, we arrive at

$$p(\boldsymbol{\pi}^\ell \mid \boldsymbol{\lambda}^\ell) = \prod_{i=1}^{|\boldsymbol{\pi}|} p(\pi_i^\ell \mid \boldsymbol{\lambda}^\ell) \tag{1}$$

$$= \prod_{i=1}^{|\boldsymbol{\pi}|} \textit{sigmoid}\left(\mathbf{e}_{\pi_i}^\top \boldsymbol{\lambda}^\ell\right) \tag{2}$$

$$= \prod_{i=1}^{|\boldsymbol{\pi}|} \frac{1}{1 + \exp(-\mathbf{e}_{\pi_i}^\top \boldsymbol{\lambda}^\ell)} \tag{3}$$

We define the the prior over language embeddings:

$$p(\boldsymbol{\lambda}^\ell) = \mathcal{N}\left(\mathbf{0}; \sigma^2 I\right) \tag{4}$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian whose covariance is fixed at I. Now, give a collection of languages $\mathcal{L}$, we arrive at the joint distribution

$$\prod_{\ell \in \mathcal{L}} p(\boldsymbol{\pi}^\ell, \boldsymbol{\lambda}^\ell) = \prod_{\ell \in \mathcal{L}} p(\boldsymbol{\pi}^\ell \mid \boldsymbol{\lambda}^\ell) \cdot p(\boldsymbol{\lambda}^\ell) \tag{5}$$

Note that $p(\boldsymbol{\lambda}^\ell)$ is, spiritually at least, a universal grammar: it is the prior over what sort of languages can exist, albeit encoded as a real vector. In the parlance of principles and parameters, the prior represents the principles.

Then our model parameters are $\boldsymbol{\Theta} = \{\mathbf{e}_{\pi_1}, \ldots, \mathbf{e}_{\pi_{|\boldsymbol{\pi}|}}, \boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^{|\mathcal{L}|}\}$. Note that for the remainder of the paper, we will never shorten 'model parameters' to simply 'parameters' to avoid ambiguity. We will, however, refer to 'typological parameters' as simply 'parameters.'

We can view this model as a form of exponential-family matrix factorisation (Collins et al., 2001). Specifically, our model seeks to explain a binary matrix of parameters. We consider such matrices as the one in Figure 2, which depicts some of the binarised feature values for word order and affixation for English, Dutch, German, Vietnamese, Turkish, and Marind. We will have some parameters seen during training (highlighted in blue), some we use for evaluation (highlighted in red), and some which are unknown due to the nature of WALS (highlighted in green). Crucially, the model in eq. (5) allows us to learn the correlations between typological parameters, as illustrated in Figure 1. We train the model over 10 epochs with a batch size of 64, using the Adam optimiser (Kingma and Ba, 2014) and L_2 regularisation (0.1), which corresponds to the Gaussian prior with variance $\sigma^2 = 10$.

4 A Semi-Supervised Extension

A natural question we might ask is if our model can exploit unlabelled monolingual data to improve its performance. We explain how we can induce language embeddings from unlabelled data below and then incorporate these into our model through the prior eq. (4). This results in a semi-supervised model, as we incorporate an unsupervised pretraining step. This is motivated by the fact that related languages tend to exhibit correlations between each other. Figure 3 shows the distribution

---

[1] For an overview of differences between these schools, we refer the reader to Haspelmath (2008).

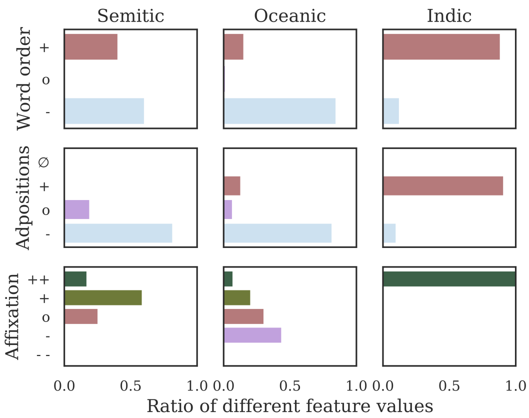


Figure 3: Distribution of feature values across three of the biggest language families in WALS. The representation of feature values is described in Figure 1.

of a few features within the Semitic, Oceanic, and Indic language branches. Notice, for instance, the skewed distribution of feature values within the Indic branch: languages in that branch are almost exclusively head-initial with respect to word order, order of adposition and noun, and affixation.

4.1 Distributional Language Embeddings

Words can be represented by distributed word representations, currently often in the form of word embeddings. Similarly to how words can be embedded, so can languages, by associating each language with a real-valued vector known as a *language embedding*. Training such representations as a part of a multilingual model allows us to infer similarities between languages. This is due to the fact that in order for multilingual parameter sharing to be successful in this setting, the neural network needs to use the language embeddings to encode features of the languages. Previous work has explored this type of representation learning in various tasks, such as NMT (Malaviya et al., 2017), language modelling (Tsvetkov et al., 2016; Östling and Tiedemann, 2017), and tasks representing morphological, phonological, and syntactic linguistic levels (Bjerva and Augenstein, 2018a).

In the context of computational typology, representations obtained through language modelling have been the most successful (Östling and Tiedemann, 2017). This approach is particularly interesting since unlabelled data is available for a large portion of the world's languages, meaning that high quality language embeddings can be obtained for more than 1,000 of the world's languages.

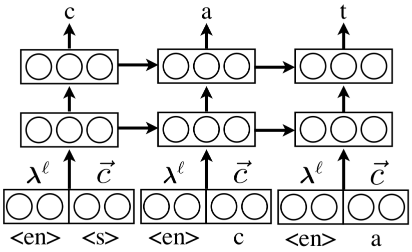


Figure 4: Char LSTM LM with language embeddings concatenated to char embeddings at every time step.

4.2 Language Embeddings through LMs

In this work, we use a language modelling objective to pre-train language embeddings; we train a character-level neural language model with a distributed language embedding of language ℓ. Specifically, we use the model of Östling and Tiedemann (2017), visualised in Figure 4. The model is a stacked character-based LSTM (Hochreiter and Schmidhuber, 1997) with two layers, followed by a softmax output layer. In order to accommodate the language embeddings, this relatively standard model is modified such that language embeddings are concatenated to the character embeddings at each time step. This method returns a language embedding, which we denote $\tilde{\boldsymbol{\lambda}}^\ell$ to distinguish it from the language embeddings $\boldsymbol{\lambda}^\ell$ discussed in the previous section. We use the same hyperparameter settings as Östling and Tiedemann (2017), with 1024-dimensional LSTMs, 128-dimensional character embeddings, and 64-dimensional language embeddings. Training is done with Adam (Kingma and Ba, 2014), and using early stopping.

In the semi-supervised regime, we use the estimated language embedding $\tilde{\boldsymbol{\lambda}}^\ell$ from the language model and define the model as follows

$$p(\boldsymbol{\pi}^\ell \mid \tilde{\boldsymbol{\lambda}}^\ell) = \prod_{i=1}^{|\boldsymbol{\pi}|} sigmoid(\mathbf{e}_{\pi_i}^\top \tilde{\boldsymbol{\lambda}}^\ell) \qquad (6)$$

omitting the learned language embedding $\boldsymbol{\lambda}$ in the matrix factorisation. The likelihood of this model is now convex in the parameter embeddings. In contrast to the full matrix factorisation setting, here, all language-specific knowledge must come from an external source, namely, the unlabelled text.

5 A Novel Task: Typological Collaborative Filtering

In this section, we introduce a novel task for linguistic typology, which we term **typological collaborative filtering**. Typological KBs such as WALS are known to be incomplete. In other words, not all parameters π_i^ℓ are observed for all languages. Thus, a natural question we may ask how well our models can predict unobserved (or held-out) parameters. We view this problem as analogous to collaborative filtering (CF). Our task is similar to knowledge base population or completion in that we start off with a partly populated KB which we aim to complete, but differs in the aspect that we attempt to infer values from correlations between features and similarities between languages, rather than inferring these from a collection of texts.

CF is a common technique for recommender systems (see §9). Consider the task of recommending a new movie to a customer. Given which movies different users have liked (equivalent to a typological parameter being 'on') and which movies the user has disliked (equivalent to the typological parameter being 'off'), a CF model tries to figure out the (latent) preferences of each user and the latent genre of each movie. In our setting, the languages are analogous to users and movies are analogous to parameters. Our model in eq. (5) seeks to learn what latent properties of languages and what latent properties of parameters explain their correlations.

6 Data

WALS. The World Atlas of Language Structures (WALS) is a large knowledge base of typological properties at the lexical, phonological, syntactic and semantic level on which we will run our experiments. The documentation of linguistic structure is spread throughout a wide variety of academic works, ranging from field linguistics to grammars describing the nuances of individual grammatical uses. KB creation is a laborious tasks as it involves distilling knowledge into a single, standardised resource, which, naturally, will be incomplete, prompting the need for methods to complete them (Min et al., 2013). In the case of WALS, few languages have all values annotated for all of the properties. In this section, we offer a formalisation of typological KBs to allow for our development of a probabilistic model over vectors of properties. WALS, for instance, contains $n = 202$ different parameters (Dryer and Haspelmath, 2013).

Binarisation of WALS. Many common typological KBs, including WALS, the one studied here, contain binary as well as non-binary parameters. To deal with this, we binarise the KB as follows: Whenever there is a typological parameter that takes ≥ 3 values, e.g., 'Feature 81A: Order of Subject, Object and Verb' which takes the 7 values 'SOV', 'SVO', 'VSO', 'VOS', 'OVS', 'OSV', 'No dominant order', we introduce that many binary parameters. At test time, we get non-binary predictions by using a simple decoder that returns the arg max over the predicted probabilities for the binary features.

As each typological parameter with $n \geq 3$ feature values is coded as a one-hot binary vector of length n, we need to make sure that we do not mix a single typological parameter for a language into the training and test sets. This is visualised in Figure 2, where we train on the blue squares, i.e., the binarised 81A feature for for English, and the 26A feature for Dutch, as well as all features for all non-Germanic languages. The model is then evaluated on the held-out features for Germanic highlighted in red, i.e., 26A for English, 81A for Dutch, and both of these for German. This is important, as knowing that a language is SVO would make it trivial to infer that it is not, e.g., SOV.

Unlabelled multilingual data. To induce language embeddings, we need a considerable amount of multilingual unlabelled data. We use an in-house version of the massively multilingual Bible corpus, so as to have comparable data for all languages, although parallel data is not a strict necessity for our method.[2] We train the multilingual language model on a collection of 975 languages, each with approximately 200,000 tokens available. We only train on languages for which the symbol size is relatively modest, a criterion which we fulfil by only using translations with Latin, Cyrillic, and Greek alphabets.

Language used in the bible differs substantially from most modern language use, which would be a challenge if one were interested in transferring the language model itself. Here, we are only interested in the distributed language embeddings for each language ℓ. It is safe to assume that the typological features underlying the texts we use will be representative of those coded in WALS, hence the domain should not matter much and the method should work equally well given any domain of input texts for the unsupervised training of language embeddings.

---

[2] Some languages available via `http://homepages.inf.ed.ac.uk/s0787820/bible/`

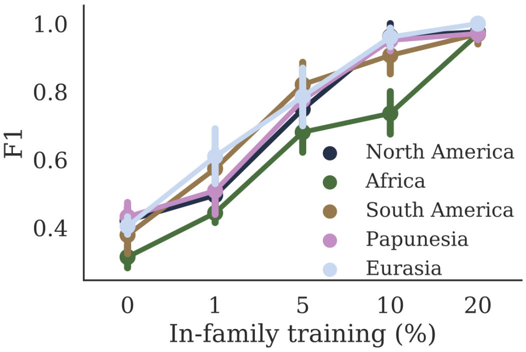


Figure 5: F1 score for feature prediction separated by macroarea with varying in-branch training data.

7 Experiments

7.1 General Experimental Setup

As described in §6, we binarise WALS. In order to compare directly with our semi-supervised extension, we limit our evaluation to the subset of languages which is the intersection of the languages for which we have Bible data, and the languages which are present in WALS. Finally, we observe that some languages have very few features encoded, and some features are encoded for very few languages. For instance, feature 10B (Nasal Vowels in West Africa), is only encoded for a total of 40 languages, and only one feature value appears for more than 10 languages. Because of this, we restrict our evaluation to languages and feature values which occur at least 10 times. Note that we evaluate on the original parameters, and not the binarised ones.

Our general experimental set-up is as follows. We first split the languages in WALS into each language branch (*genus* using WALS terminology). This gives us, e.g., a set of Germanic languages, a set of Romance languages, a set of Berber languages, and so on. (We note that this does not correspond to the notion of a language family, e.g., the Indo-European language family.) We wish to evaluate on this type of held-out set, as it is both relatively challenging: If we know the parameters of Portuguese, predicting the parameters for Spanish is a much easier task. This setup will both give us a critical estimate of how well we can predict features overall, in addition to mimicking a scenario in which we either have a poorly covered language or branch, which we wish to add to WALS.

7.2 Typological Collaborative Filtering

We evaluate our method for typological collaborative filtering on each language branch in WALS in a series of experiments. Given a branch B (a set of languages), we randomly select 80% of the feature-language combinations from the languages $\ell \in B$, which we use for evaluation (e.g. those highlighted in red in Figure 2). The remaining 20% is either not considered, or (partially) used for training, as we run experiments in which we train on $(0, 1, 5, 10, 20)\%$ relative of the held-out data. The idea behind this is that it should be very difficult to predict features if nothing is known for a language at all, whereas knowing a few features of a language, or of related languages, should allow the model to take advantage of the strong correlations between features (Figure 1) and between languages (Figure 3). We train on all data in WALS for languages which are not in the current evaluation branch under consideration.

Each experiment is then an evaluation of how well we can predict features for a completely or relatively unseen language family. Evaluation is done across the branches in WALS with more than four languages represented, after filtering away languages for which we have fewer than 10 features available. This amounts to a total of 36 branches, and 448 languages. We repeat each experiment 5 times per language branch, for each proportion of in-branch training data in $(0, 1, 5, 10, 20)\%$, yielding a total of 900 experiment runs. The results reported are the mean across these runs.

Figure 5 shows the micro F1 score we obtain averaged across macroareas. The bars indicate 95% confidence intervals. We can see that, with access to 20% of the in-branch training data, we can predict features at above 90% F1 score regardless of macroarea. Prediction generally is more challenging for languages in the macroarea Africa. This can be explained by, e.g., contrasting with the Eurasian macroarea. Whereas the latter includes branches which are relatively uncontroversial, such as Germanic and Slavic languages, this is not the case with the former. One such example is Bongo-Bagirmi (one of the evaluation branches, spoken in Central Africa), for which there is poor agreement in terms of classification (Bender, 2000).

7.3 Semi-supervised extension

We next evaluate the semi-supervised extension, which requires unlabelled texts for a large amount

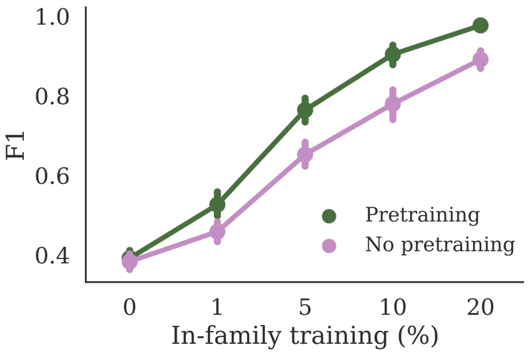


Figure 6: F1 scores across all held-out language families, comparing pretraining with no pretraining.

of languages, although the domain of these texts should not matter much. This allows us to take advantage of correlations between similar languages to point the model in the right direction. Even with 1% training data, it may be very useful for a model to know that, e.g., German and Dutch are grammatically very similar. Hence, if the 1% training data contains features for Dutch, it should be quite easy for the model to learn to transfer these to German.

Figure 6 shows results with pre-training. Without any in-branch training data, using pretrained embeddings does not offer any improvement in prediction power. This can be explained by the fact language embeddings are updated during training, which leads to a drift in the representations present in the training material. This was chosen since not updating the representations yielded poor performance in all settings explored. We hypothesise that this is because, although a language modelling objective offers a good starting point in terms of encoding typological features, it is not sufficient to explain the typological diversity of languages. For instance, a language model should hardly care about the phonological nature of a language. This is in line with previous work which shows that the linguistic nature of the target task is important when predicting typological features with language embeddings (Bjerva and Augenstein, 2018a). However, once we have access to $> 5\%$ of in-branch training data, language embeddings offers a substantial improvement, e.g. an F1 error reduction of more than 50% with access to 10% of the in-branch data (see Table 1 and Table 3 for per-branch results in the appendix). This shows that we can partially aid a typologist's work by utilising unannotated data.

In-branch train %	Freq. F1	Indiv. pred. F1	T-CF F1	SemiSup F1
0.00	0.2950	0.2990	0.3998	0.3916
0.01	0.2949	0.2976	0.4578	0.5263
0.05	0.2947	0.2970	0.6552	0.7641
0.10	0.2945	0.2971	0.7807	0.9040
0.20	0.2938	0.2973	0.8835	**0.9767**

Table 1: Aggregate results w. 0-20% relative use of in-branch training data. Columns show: most frequent class (Freq.), individual prediction per feature with language-embeddings (Individual pred.), typological collaborative filtering (T-CF), semi-supervised extension (SemiSup). For standard deviations across runs and per-branch results, see appendix.

7.4 Quantitative comparison

In addition to evaluating our method of typological CF, we compare to some baselines drawn from earlier work. First, we report a most frequent value baseline. As many typological features are heavily skewed, this is quite high already. For instance, defaulting to the most frequent value for word order (i.e. SVO) would yield an accuracy of 41% (**Freq.** in Table 1). A more involved baseline is Bjerva and Augenstein (2018a), who use pre-trained language embeddings in a k-NN classifier trained on individual WALS features (**Individual pred.** in Table 1). For the baseline reported here, we only use one nearest neighbour for this prediction. The scores we obtain here are quite low compared to Bjerva and Augenstein (2018a), which is explained by the fact that we have access to very little training data in the current setting, and highlights the importance of taking advantage of correlations between languages and features, and not simply looking at these factors in isolation. Finally, we compare our typological collaborative filtering approach, as well as our semi-supervised extension (**T-CF** and **SemiSup** in Table 1).

8 Analysis

Accuracy for several experimental settings is visualised in Figure 7, broken down by the linguistic category of the predicted features. Since results change little between the 5% in-branch setting and higher percentages, we only look at 0%, 1% and 5% here. We also visualise accuracy without (Figure 7, left) and with our semi-supervised extension (Figure 7, right) in each setting.

8.1 Typological collaborative filtering with variable amounts of data

Focussing on Figure 7 (left) alone first we observe an expected pattern: using increasingly more in-branch data boosts performance across all feature groups. This increase in accuracy can be attributed to the model having more knowledge about each query language in itself and about how languages relate to one another, based on similarities in their parameter configurations.

Making a prediction about the order of adposition and noun phrase in a given language, lacking any other information about that language, is basically a shot in the dark. In-branch training data, in our experiments, includes in-language training data, too. Having one piece of information about the word order of that language, its ordering of relative clause and noun, or even its affixational properties, immediately makes the prediction informed rather than random: in many other languages within and outside this particular language family the model would have likely observed a strong correlation between these features and the order of adposition and noun phrase, which are all subject to the more general headedness parameter.

Certain features and feature configurations may not be as abundant cross-linguistically as the set of headedness features. In those cases, access to in-branch data is crucial. Consider e.g. the feature 10B Nasal Vowels in West Africa: a handful of language branches exhibit this feature and at least one of its values, *no nasal vs. oral vowel contrast*, is characteristic predominantly of Niger-Congo languages. Without *any* in-branch training data, the model's knowledge of this feature value is extremely limited, making its correct prediction for a Niger-Congo language virtually impossible. A small amount of in-branch training data thus increases the chance of a correct prediction greatly.

8.2 Semi-supervised extension

Comparing Figure 7 to Figure 6 reveals a crucial finding. While we see very little improvement from pretraining for 0% in-branch training overall, for individual linguistic categories, it mostly benefits prediction: seven out of nine feature groups are predicted more accurately with pretraining. Phonological and morphological predictions experience moderate deterioration, however, counterbalancing much of the improvement in other categories, which leads to the overall result of seemingly little improvement from pretraining. The limited effect of pretraining on prediction of phonological and morphological features can be explained with reference to the richness and complexity of these linguistic domains, which makes for data sparsity and generally makes them harder to learn based on distributional information alone. Moreover, a number of phonological features refer to aspects of language that may not be reflected in writing, such as stress and devoicing. All other categories concern syntactic and semantic information, which is known to be learnable from word distribution, and therefore benefit from the knowledge carried by language embeddings.

Language	**Genus**	**Fixed Stress Location**	**Weight-Sensitive Stress**
English	Germanic	?	Right-oriented
Icelandic	Germanic	Initial	Fixed stress

Table 2: In-branch training data in example scenario

Figure 7 (right) shows an unsteady interaction between pretraining and the addition of increasing amounts of in-branch data. While pretraining alone helps for predicting most features, as pointed out above, an extra 1% of in-branch data in the pretrained setting has a rather variable impact across feature groups. For a few groups it helps, as is expected, for a few it has no effect and for two groups, 'Word Order' and 'Simple Clauses', it makes for quite a drop in accuracy. We speculate that this effect, while negative, is indicative of the general power of language embeddings in associating related languages. Consider the test query 'Fixed Stress Location' in English, where the 1% of in-branch training data contains the information in Table 2. Based on feature correlation alone, the model should predict 'No fixed stress' for English, since this value always co-occurs with 'Right-oriented stress'. Yet, due to the proximity in the English and Icelandic embeddings, the model may copy the value of Icelandic and falsely predict 'Initial stress' for English, too. The risk of this happening decreases with more in-branch training data, since the model can generalise over more in-branch features.

Lastly, notice that accuracy for phonological features remains low even with 5% of in-branch data, and it is lower in the pretrained setting compared to the no-pretraining one. This brings us to the conclusion that using pretrained embeddings which are fine-tuned for specific tasks which encode different linguistic levels, as in Bjerva and Augen-

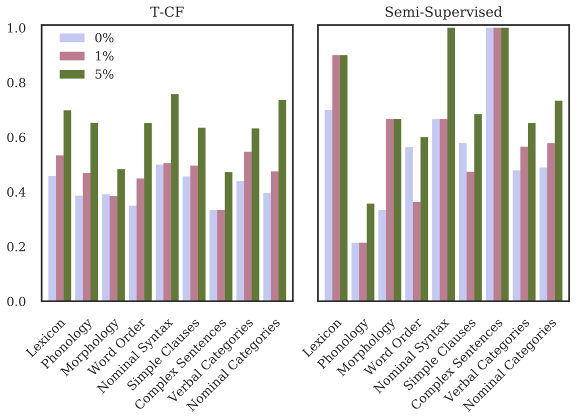


Figure 7: Accuracy per feature group (Germanic).

stein (2018a), might also be useful in our semi-supervised extension of typological collaborative filtering.

9 Related Work

Computational Typology The availability of unlabelled datasets for hundreds of languages permits inferring linguistic properties and categories (Östling, 2015; Asgari and Schütze, 2017). Individual prediction of typological features has been attempted in conjunction with several NLP tasks (Malaviya et al., 2017; Bjerva and Augenstein, 2018a,b). Our work is most similar to Murawaki (2017), who presents a Bayesian approach to utilising relations between features and languages for feature prediction. However, our work differs on several important counts, as we (i) include language information obtained through unsupervised learning, which allows us to take advantage of raw data and predict features for completely unannotated languages, (ii) analyse the effects of varying amounts of known features, especially in situations with and without in-branch training data, and (iii) view the problem of typological features through the lens of parameters from principles and parameters (Chomsky, 2000). Deep generative models have also been explored previously for modelling phonology (Cotterell and Eisner, 2017). Our work builds on these research directions, by (i) developing a deep generative model which (ii) takes advantage of correlations, rather than predicting features individually, and (iii) exploits unlabelled data. This work is also related to linguistic representations encoded in neural models (Kádár et al., 2017) and language embeddings (Bjerva et al., 2019), multilingual relations between languages in various representational levels (Beinborn and Choenni, 2019), as well as the related problem of phylogenetic inference (Farach et al., 1995; Nichols and Warnow, 2008). For a survey of typology in NLP, see Ponti et al. (2018).

Matrix Factorisation Collaborative Filtering was popularised in the early 1990s as a technique for recommender systems with applications such as mail filtering (Goldberg et al., 1992), and article (Resnick et al., 1994) and movie recommendation (Dahlen et al., 1998). Model-based algorithms soon became popular (Breese et al., 1998) to overcome the cold start problem arising for unseen users or items at test time. The most successful one of these, in turn, is matrix factorisation, as applied in this paper, which represents users and items as (dense) vectors in the same latent feature space and measures their compatibility by taking the dot product between the two representations (Koren et al., 2009; Bokde et al., 2015). Beyond recommender systems, matrix factorisation has shown successes in a wide variety of subareas of NLP (Riedel et al., 2013; Rocktäschel et al., 2015; Levy and Goldberg, 2014; Lei et al., 2014; Augenstein et al., 2018).

10 Conclusion

We introduce a generative model inspired by the principles-and-parameters framework, drawing on the correlations between typological features of languages to solve the novel task of typological collaborative filtering. We further show that raw text can be utilised to infer similarities between languages, thus allowing for extending the method with semi-supervised language embeddings.

Acknowledgements

We acknowledge the computational resources provided by CSC in Helsinki through NeIC-NLPL (www.nlpl.eu), and the support of the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. The third author acknowledges support from a Facebook Fellowship.

References

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 113–124, Copenhagen, Denmark. Association for Computational Linguistics.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1896–1906, New Orleans, Louisiana. Association for Computational Linguistics.

Mark C Baker. 2008. The macroparameter in a microparametric world. *The Limits of Syntactic Variation*, 132.

Lisa Beinborn and Rochelle Choenni. 2019. Semantic drift in multilingual representations. *arXiv preprint arXiv:1904.10820*.

M Lionel Bender. 2000. Nilo-Saharan. In B. Heine and Nurse. D., editors, *African languages: An introduction*, pages 43–73. Cambridge University Press Cambridge.

Theresa Biberauer, Anders Holmberg, Ian Roberts, and Michelle Sheehan. 2009. *Parametric variation: Null subjects in minimalist theory*. Cambridge University Press.

Johannes Bjerva and Isabelle Augenstein. 2018a. From phonology to syntax: Unsupervised linguistic typology at different levels with language embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 907–916, New Orleans, Louisiana. Association for Computational Linguistics.

Johannes Bjerva and Isabelle Augenstein. 2018b. Tracking typological traits of uralic languages in distributed language representations. In *Proceedings of the Fourth International Workshop on Computatinal Linguistics of Uralic Languages*, pages 76–86, Helsinki, Finland. Association for Computational Linguistics.

Johannes Bjerva, Robert Östling, Maria Han Veiga, Jörg Tiedemann, and Isabelle Augenstein. 2019. What do Language Representations Really Represent? *Computational Linguistics*. In press, arXiv preprint arXiv:1901.02646.

Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. 2015. Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey. *Procedia Computer Science*, (49):136–146.

John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52.

Andrew Carnie. 2013. *Syntax: A generative introduction*. John Wiley & Sons.

Noam Chomsky. 1957. *Syntactic structures*. Walter de Gruyter.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press.

Noam Chomsky. 1981. *Lectures on government and binding: The Pisa lectures*. 9. Walter de Gruyter.

Noam Chomsky. 2000. *New Horizons in the Study of Language and Mind*. Cambridge University Press.

Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. 2001. A generalization of principal components analysis to the exponential family. In *NIPS*, pages 617–624.

Ryan Cotterell and Jason Eisner. 2017. Probabilistic typology: Deep generative models of vowel inventories. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1182–1192, Vancouver, Canada. Association for Computational Linguistics.

William Croft. 2002. *Typology and Universals*. Cambridge University Press.

B.J. Dahlen, J.A. Konstan, J.L. Herlocker, N. Good, A. Borchers, and J. Riedl. 1998. Jump-starting movieLens: User benefits of starting a collaborative filtering system with ”dead-data”. In . University of Minnesota TR 98-017.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Jason Eisner. 2002. Discovering syntactic deep structure via Bayesian statistics. *Cognitive Science*, 26(3):255–268.

Martin Farach, Sampath Kannan, and Tandy Warnow. 1995. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13:155–179.

David Goldberg, David Nichols, Brian Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70. Special Issue on Information Filtering.

Randy Allen Harris. 1995. *The Linguistics Wars*. Oxford University Press.

Martin Haspelmath. 2008. Parametric versus functional explanations of syntactic universals. *The limits of syntactic variation*, 132:75–107.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural network. *Computational Linguistics*, 43:761–780.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2529–2535, Copenhagen, Denmark. Association for Computational Linguistics.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, Atlanta, Georgia. Association for Computational Linguistics.

Yugo Murawaki. 2017. Diachrony-aware induction of binary latent representations from typological features. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 451–461, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Johanna Nichols and Tandy Warnow. 2008. Tutorial on computational linguistic phylogeny. *Language and Linguistics Compass*, 2:760–820.

Marco Nicolis and Theresa Biberauer. 2008. The null subject parameter and correlating properties. *The limits of syntactic variation*, 132:271.

Marc van Oostendorp. 2015. Parameters in phonological analysis: stress. In A. Fabregas, J. Mateu, and M. Putnam, editors, *Contemporary Linguistic Parameters*, Contemporary Studies in Linguistics, pages 234–258. London: Bloomsbury.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 205–211, Beijing, China. Association for Computational Linguistics.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649, Valencia, Spain. Association for Computational Linguistics.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2018. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics, arXiv preprint arXiv:1807.00914*.

Alan Prince and Paul Smolensky. 2008. *Optimality Theory: Constraint interaction in generative grammar*. John Wiley & Sons.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 Conference on Computer Supported Collaborative Work*, pages 175–186.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado. Association for Computational Linguistics.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W. Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1357–1366, San Diego, California. Association for Computational Linguistics.

A Appendices

Macroarea	**Branch**	**In-branch training (%)**	**Freq. F1**	**Individual pred. F1**	**T-CF F1**	**SemiSup F1**
Africa	Bantoid	0.00	0.3510 (0.0042)	0.3540 (0.0057)	0.3527 (0.0260)	0.3287 (0.0292)
		0.01	0.3510 (0.0042)	0.3643 (0.0077)	0.4100 (0.0196)	0.4013 (0.0260)
		0.05	0.3510 (0.0036)	0.3523 (0.0053)	0.4997 (0.0446)	0.5467 (0.0266)
		0.10	0.3507 (0.0031)	0.3570 (0.0029)	0.5687 (0.1065)	0.6107 (0.0963)
		0.20	0.3480 (0.0057)	0.3450 (0.0071)	0.8397 (0.0475)	**0.8620** (0.0645)
Papunesia	Chimbu	0.00	0.2860 (0.0182)	0.2940 (0.0142)	0.3460 (0.0771)	0.3357 (0.0708)
		0.01	0.2860 (0.0182)	0.2887 (0.0120)	0.3600 (0.0805)	0.2773 (0.0890)
		0.05	0.2850 (0.0171)	0.2990 (0.0236)	0.3833 (0.1012)	0.4560 (0.2013)
		0.10	0.2850 (0.0171)	0.2937 (0.0161)	0.6140 (0.0884)	0.6950 (0.0318)
		0.20	0.2873 (0.0181)	0.2893 (0.0135)	0.7800 (0.1181)	**0.8267** (0.1605)
Africa	Kwa	0.00	0.2417 (0.0111)	0.2540 (0.0099)	0.3163 (0.0381)	0.2720 (0.0490)
		0.01	0.2407 (0.0104)	0.2550 (0.0051)	0.3807 (0.0775)	0.3843 (0.0796)
		0.05	0.2417 (0.0111)	0.2597 (0.0095)	0.4927 (0.1181)	0.5290 (0.0978)
		0.10	0.2417 (0.0111)	0.2453 (0.0076)	0.6827 (0.1092)	0.6827 (0.1092)
		0.20	0.2417 (0.0111)	0.2443 (0.0033)	**0.9960** (0.0057)	**0.9960** (0.0057)
North America	Mixtecan	0.00	0.3507 (0.0076)	0.3427 (0.0047)	0.4387 (0.1244)	0.4747 (0.0766)
		0.01	0.3507 (0.0076)	0.3360 (0.0065)	0.4593 (0.0514)	0.5010 (0.0304)
		0.05	0.3500 (0.0083)	0.3460 (0.0065)	0.6867 (0.2274)	0.7803 (0.0988)
		0.10	0.3490 (0.0094)	0.3400 (0.0114)	0.8610 (0.1288)	0.8830 (0.1441)
		0.20	0.3490 (0.0094)	0.3383 (0.0130)	0.9883 (0.0165)	**1.0000** (0.0000)
Papunesia	Oceanic	0.00	0.3837 (0.0048)	0.4030 (0.0096)	0.3677 (0.0416)	0.3663 (0.0433)
		0.01	0.3837 (0.0048)	0.3957 (0.0066)	0.3783 (0.0324)	0.4047 (0.0217)
		0.05	0.3833 (0.0041)	0.3993 (0.0068)	0.6193 (0.0382)	0.6753 (0.0838)
		0.10	0.3830 (0.0051)	0.3983 (0.0071)	0.7243 (0.1116)	0.7337 (0.1245)
		0.20	0.3817 (0.0034)	0.4057 (0.0113)	0.8393 (0.0662)	**0.9200** (0.0641)
All	All	0.00	0.2950 (0.0453)	0.2990 (0.0448)	0.3998 (0.1095)	0.3916 (0.1077)
		0.01	0.2949 (0.0453)	0.2976 (0.0457)	0.4578 (0.1298)	0.5263 (0.1605)
		0.05	0.2947 (0.0452)	0.2970 (0.0436)	0.6552 (0.1726)	0.7641 (0.1572)
		0.10	0.2945 (0.0451)	0.2971 (0.0434)	0.7807 (0.1782)	0.9040 (0.1324)
		0.20	0.2938 (0.0446)	0.2973 (0.0451)	0.8835 (0.1278)	**0.9767** (0.0464)

Table 3: Aggregate results with 0-20% relative use of in-branch training data. The columns indicate the most frequent class (Freq.), individual prediction per feature with language-embeddings (Individual pred.), typological collaborative filtering (T-CF), semi-supervised extension (SemiSup).

Quantifying the morphosyntactic content of Brown Clusters

Manuel R. Ciosici
UNSILO A/S and
Aarhus University
Aarhus, Denmark
manuel@cs.au.dk

Leon Derczynski
IT University of
Copenhagen
Copenhagen, Denmark
leod@itu.dk

Ira Assent
Department of Computer Science
Aarhus University
Aarhus, Denmark
ira@cs.au.dk

Abstract

Brown and Exchange word clusters have long been successfully used as word representations in Natural Language Processing (NLP) systems. Their success has been attributed to their seeming ability to represent both semantic and syntactic information. Using corpora representing several language families, we test the hypothesis that Brown and Exchange word clusters are highly effective at encoding morphosyntactic information. Our experiments show that word clusters are highly capable of distinguishing Parts of Speech. We show that increases in Average Mutual Information, the clustering algorithms' optimization goal, are highly correlated with improvements in encoding of morphosyntactic information. Our results provide empirical evidence that downstream NLP systems addressing tasks dependent on morphosyntactic information can benefit from word cluster features.

1 Introduction

Distributionally generated word classes (often referred to as *word clusters*) are hard clusters, containing all word types observed in a corpus, allocated to clusters based on contextual information observed in the corpus. They have found wide use in Natural Language Processing (NLP) systems as an alternative to word embeddings such as word2vec (Mikolov et al., 2013). Word clusters differentiate themselves from word embeddings by requiring estimation of many fewer parameters, and by their ability to derive qualitative representations from smaller corpora (Qu et al., 2015; Bansal et al., 2014).

Brown Clusters (Brown et al., 1992) are a well-known approach based on hard, hierarchical, distributionally derived groups of word types observed in a corpus of unstructured text, with Average Mutual Information (AMI) as the optimization goal. Exchange Clusters are an alternative approach obtained by applying the Exchange Algorithm (Kneser and Ney, 1993) to the same optimization goal. Unlike Brown, Exchange outputs a flat clustering, with no hierarchy (Martin et al., 1998). When only the bottom of the hierarchy is used, like in this paper, Exchange and Brown clusters are interchangeable.

Both Brown and Exchange clusters have been used as word representations for various Natural Language Processing tasks such as Part of Speech tagging in clean and noisy text (Swain and Cole, 2016; Owoputi et al., 2013; Derczynski et al., 2015), dependency parsing (Koo et al., 2008; Bansal et al., 2014), Chinese Word Segmentation (Liang, 2005), and Named Entity Recognition (Swain and Cole, 2016; Derczynski et al., 2015; Liang, 2005). Word clusters distinguish themselves from word embedding models by their ability to learn from little data (Bansal et al., 2014; Qu et al., 2015); for example, in cases like (Bansal et al., 2014), word clusters outperform other kinds of representations, including word embeddings. In the literature, it is often observed that word clusters seem to encode a considerable amount of morphosyntactic and semantic knowledge (Brown et al., 1992; Derczynski et al., 2015). However, it has not yet been studied to which extent such knowledge is encoded, as previous work on Brown and Exchange clusters focuses mostly on algorithmic improvements and on applications to different NLP tasks.

In this work, we present a principled study of the morphosyntactic information encoded in flat word clusters induced exclusively from class-based language models via Brown Clustering and Exchange algorithm. In particular, we focus on how well these approaches derive clusters that represent Parts of Speech as a measure of the morphosyntactic information encoded.

Proceedings of NAACL-HLT 2019, pages 1541–1550
Minneapolis, Minnesota, June 2 - June 7, 2019.

We find that Brown and Exchange clusters are highly effective at representing morphosyntactic information, even when hyper-parameters are set such that they match only the number of Parts of Speech, thereby grouping into relatively few word clusters only. Our results provide empirical evidence for the observed performance gains when including Brown and Exchange word clusters as features in NLP systems that rely on morphosyntactic information.

Furthermore, we find that there is a strong correlation between the optimization goal of Brown clustering and the Exchange Algorithm (i.e., Average Mutual Information), and performance at Parts of Speech separation, which again confirms the appropriateness of choosing AMI in word clustering for morphosyntactic information.

2 Background

Class-based language models address the problem of brittleness in classic n-gram language models by trading precision for performance stability over different text styles (Brown et al., 1992).

Brown Clustering (Brown et al., 1992) and Exchange (Kneser and Ney, 1993) are greedy algorithms that construct word classes by optimizing for higher Average Mutual Information (AMI). Maximizing Average Mutual Information is a proxy for maximizing the log-likelihood of the underlying class-based language model on the given corpus (Martin et al., 1998). Despite their age, most research on Brown or Exchange clusters has so far followed two major directions: algorithm improvements and applications in Natural Language Processing. In contrast little focus has been placed on understanding and evaluating the information content of the clusters.

In the direction of algorithm improvements, work has been done on the effect of greedy merge choices in Brown Clustering (Derczynski and Chester, 2016; Ciosici, 2015) and extension of AMI to n-grams (Martin et al., 1998). Model relaxations, particularly to Exchange, aim to improve computational performance by reducing the effect of words swapping clusters (Dehdari et al., 2016; Uszkoreit and Brants, 2008).

As mentioned earlier, both Brown and Exchange clusters have seen many applications in Natural Language Processing (NLP) systems: PoS tagging (Swain and Cole, 2016; Owoputi et al., 2013; Derczynski et al., 2015), dependency parsing (Koo et al., 2008; Bansal et al., 2014), Chinese Word Segmentation (Liang, 2005), and Named Entity Recognition (Swain and Cole, 2016; Derczynski et al., 2015; Liang, 2005). Most of this work, like (Swain and Cole, 2016) uses the word clusters as sources of features which are combined with hand-designed ones. While word clusters derived using Exchange and Brown clustering have found wide use in NLP systems, their use has been based on the assumption that they encode morphosyntactic and semantic information rather that a principled use.

In relation to Parts of Speech, early on Martin et al. (1998) concluded that initializing Exchange with PoS-homogeneous clusters has no effect on final clustering AMI, but that it does help accelerate convergence. More recently, Christodoulopoulos et al. (2010) found that Brown clusters match the performance of more sophisticated clustering methods, despite their simple algorithmic construction. The study focused on using word clustering algorithms as sources of prototypal information to prototype-driven learning models for classification. In this paper, we study the amount of morphosyntactic information encoded in Brown and Exchange word clusters with the goal of providing empirical results for a principled use of such clusters in downstream tasks.

3 Metric selection

In order to determine the amount of morphosyntactic information encoded in Brown and Exchange word clusters, we measure their ability to separate word types by their Parts of Speech. For this, we require cluster quality measures. Brown and Exchange clusters do not exist in a metric space; therefore unsupervised cluster quality measures relying on distances between points or clusters, such as the Silhouette coefficient (Rousseeuw, 1987), cannot be used. Instead, we focus on two quality measures that compare clusters with a ground truth partitioning. We work under the hypothesis that Brown and Exchange clusters represent parts of speech and thus, we consider parts of speech as the ground-truth partitioning of the data. This makes it possible to use cluster quality measures that require as input an existing ground-truth partitioning. We use PoS tags resulting from manual or automatic annotation. We evaluate using a widespread and easy to interpret measure based on overlap (*purity*), and an information theoretical measure (*Adjusted Mutual Information*).

3.1 Clustering Purity

Cluster purity measures how many points in a clustering (in our case words) have been assigned to a cluster whose predominant label they share (e.g. adjectives clustered with other adjectives, nouns with other nouns etc). Intuitively, it measures the percentage of points properly classified (via their cluster membership). Formally, cluster purity is defined as:

$$purity(C_i) = \frac{1}{|C_i|} \max_{l=1}^{|L|} |label(C_i, l)| \tag{1}$$

$$purity(C) = \sum_{i=1}^{k} \frac{|C_i|}{|V|} purity(C_i) \tag{2}$$

$$= \frac{1}{|V|} \sum_{i=1}^{k} \max_{l=1}^{|L|} |label(C_i, l)| \tag{3}$$

Where the function $label(C_i, l)$ provides the number of elements from C_i with label l and L is the set of labels. *Purity* reaches a value of 1 when the clustering is identical to the ground-truth partitioning, or each point is allocated to its own cluster (i.e. $k = |V|$). When $k = 1$, *purity* is equal to the fraction of points labeled with the most popular label, and thus provides a baseline. In our case, that is equal to the percentage of vocabulary allocated to the most popular PoS class, usually nouns. For values of $k \in (1, |V|)$, it varies depending on cluster quality. If $k > |L|$, *purity* can take a value of 1 if each cluster is a subset of a ground partition. Thus, *purity* is expected to increase as k grows higher than $|L|$.

In our experiments, *purity* measures the percentage of vocabulary that is labeled correctly. In other words, *purity* does not depend on word frequency. Thus, it is not an approximation of PoS tagging accuracy, like the M-1 measure used by Bansal et al. (2014). Since we focus on morphosyntactic information encoded in word clusters, we do not want a measure that takes into account word frequency in the given corpus (i.e, one that is a good approximation of PoS tagging performance), but one that focuses exclusively on the clusters and their content.

3.2 Adjusted Mutual Information

Since the vocabulary size is fixed, as the number of clusters k increases, *purity* can increase even if cluster membership is randomly assigned, as it is easier for smaller clusters to randomly achieve label agreement. *Adjusted Mutual Information (AdjMI)* (Vinh et al., 2009), not to be confused with *AMI* (Brown and Exchange's optimization goal), measures the amount of information shared by the ground truth partitioning U and a clustering C. In our evaluation that corresponds to the amount of information shared by the PoS ground-truth partitioning and the clustering resulting from Brown or Exchange. AdjMI corrects for the mutual information expected to exist between the ground truth partitioning U and a random clustering. Formally, it is defined as:

Data Set	Vocabulary	Length
EN UD	33 904	488 579
EN EuroParl	159 716	73 577 783
FR UD	51 670	575 887
FR EuroParl	204 166	65 878 206
CZ UD	164 483	2 226 848
CZ EuroParl	177 631	15 193 309

Table 1: Overview of data sets

$$AdjMI(U, C) = \frac{MI(U, C) - E\{MI(U, C)\}}{avg\{H(U), H(C)\} - E\{MI(U, C)\}} \tag{4}$$

Where MI and H stand for Mutual Information and Entropy, respectively. Intuitively, *AdjMI* measures how much information we gain about a point's membership in to a cluster in the ground-truth partitioning U, when we know its membership to a cluster in an induced clustering C, and the other way around. As k increases over the number of ground partitions L, *AdjMI* has the opposite effect to *purity*, i.e., it scores lower due to the higher effect attributed to randomness. Just like *purity*, *AdjMI* takes values in the interval $[0, 1]$. An *AdjMI* value of 1 corresponds to a clustering identical to the ground-truth partitioning, while a value of 0 corresponds to a clustering that is not better than a random allocation of points to clusters. Unlike *purity*, values of *AdjMI* cannot be interpreted to say anything about the percentage of points that have been properly allocated. In other words, a value of, say 0.3, does not indicate that 30% of the points have been properly separated.

4 Experiments

4.1 Data and preprocessing

We use manually annotated data from Universal Dependencies (UD) (Leung et al., 2017) for English, French and Czech[1]. We chose the group of languages so that it represents different language families. Our choice of languages is based on the amount of manually labeled data, and the presence of each language in the larger, not annotated, EuroParl corpus. We append the manual or automatic PoS tags and convert text to lowercase. Therefore, a sentence such as "Words have meaning." is transformed into "words_NOUN have_VERB meaning_NOUN ._PUNCT". Both word clustering algorithms studied in this paper are insensitive to the appended PoS tags as they operate at word and not character level. The appended tags allow us to evaluate the quality of word clusters using the measures described in the previous section. We replace all numbers, dates, times, URLs and emails with placeholders in order to reduce vocabulary size. Universal Dependencies is the largest manually annotated corpus we have access to. For experiments on larger corpora, we use the unlabeled EuroParl corpus (Koehn, 2005). More specifically, the English-French and English-Czech pairs. Since manually annotated PoS tags are not available for Europarl, we append automatically assigned PoS tags, obtained by using UDPipe (Straka and Straková, 2017) pretrained on manually annotated corpora from Universal Dependencies.

We use flat clusters from the Exchange clustering algorithm for all experiments reported in this section as they outperform the flat clustering resulting from Brown in terms of *Average Mutual Information* (their optimization goal), *Adjusted Mutual Information (AdjMI)* and cluster *purity*. All observations in the following section also apply to the flat clusters resulting from Brown. For interested readers, we include all experiments with Brown Clustering as supplementary material. The fact that Exchange outperforms Brown Clustering in terms of AMI is well-understood (Brown et al., 1992), but its effect on cluster content is not.

4.2 Morphosyntactic content in Exchange and Brown clusters

Using Exchange, we induce flat clusterings with k in the range 18 to 800. We start with $k = 18$ as it matches the observed number of distinct PoS tags in the Universal Dependencies corpora (17 distinct tags and one catch-all tag). When setting the hyper-parameter k to be higher than 18, if Exchange separates clusters by Parts of Speech, then the expectation is that clusters are subsets of words sharing the same PoS tags, and that *purity* for such clusterings will be high. In Figure 1a, we show *purity* measured on the aforementioned clusters. We can see that, even when the number of clusters is equal to that of PoS tags ($k = 18$), between 55% and 62% of the vocabulary is properly separated. *Purity* increases as k increases towards 100. At $k = 500$ and $k = 800$, between 64% and 70% of the vocabulary is grouped based on PoS, not that much more than at $k = 100$.

Increasing the number of clusters k to high values is not guaranteed to improve *purity*, for any of the languages studied. This is contrary to the expectation that *purity* increases when $k > 18$. This indicates that Exchange and Brown do not exclusively optimize for Part of Speech separation. We believe the clustering algorithms might be striking a balance between encoding semantic and morphosyntactic information, since at higher values of k we usually see more clusters with a coherent semantic theme such as names of geographic locations, names of men, names of women, nouns determining times, similar to the clusters observed in previous literature (Brown et al., 1992). For example, when using $k = 18$ in English, the token "cat_NOUN" appears in the same cluster as the plural version "cats_NOUN". At $k = 800$, the clusters distinguish between the two tokens and "cat_NOUN" is placed together with a number of nouns in the singular such as "budget_NOUN, computer_NOUN, pet_NOUN, wheel_NOUN", while "cats_NOUN" is placed in a cluster of mostly pluralized nouns like "children_NOUN, rooms_NOUN, dogs_NOUN, families_NOUN".

Adjusted Mutual Information (AdjMI) for the same clusterings, Figure 1b, shows a considerable decrease as the hyper-parameter k increases, especially at high values of 500 and 800. This is in line with the expected punishment due to the effect attributed to randomness (see the term for expected

---

[1] We use data from release 2.2 of Universal Dependencies.

value of Mutual Information in Equation (4)). At values of k closer to the number of PoS tags in the data, *AdjMI* varies little from one clustering to the other. More interestingly, the relative order of separation performance between the languages studied is maintained going from *purity* to *AdjMI*, suggesting that no measure-specific effects are at play.

By studying the frequency of incorrectly classified words types (i.e., of those whose PoS tag does not match the most popular one in their cluster), we find that most (about 85%) occur less than 5 times in the corpora. Such few observations likely do not provide enough information for Brown or Exchange to properly place those words. Therefore, from the already computed clusterings, we remove words with a frequency less than 5 and recalculate the two quality measures.

In Figures 1c and 1d, we can see that both *purity* and *AdjMI* improve considerably. Even in the most difficult case ($k = 18$), where the number of clusters matches that of distinct PoS tags, between 68% and 78% of words are properly placed, an increase of $21\% - 28\%$ compared to the values in Figure 1a. For *AdjMI*, the scores more than double. These results show that even for small corpora, a large amount of morphosyntactic information can be encoded, completely unsupervised, using the Exchange clustering algorithm. (The same behavior can be observed for clusters derived using the Brown Clustering algorithm, see supplementary material.) It also shows that, for low frequency terms, there is not enough contextual information for a proper clustering.

One disadvantage of thresholding by frequency is that, due to the zipfian distribution of word frequencies in natural language, only a fraction of the original vocabulary remains after filtering out words with a frequency less than 5: English ($8\,143\ words - 24.01\%$), French ($9\,020\ words - 17.45\%$), Czech ($37\,026\ words - 22.51\%$). In order to benefit from more reliable word usage estimates, it is necessary to perform the same experiment on larger corpora. Unfortunately bigger manually annotated data sets do not exist. We therefore turn to automatic PoS tagging.

We use UDPipe (Straka and Straková, 2017) with models pretrained on the Universal Dependencies corpora to automatically tag text from the EuroParl multi-language corpus containing transcriptions of European Parliament proceedings (Koehn, 2005). Automated annotations introduce labeling noise that should lead to a decrease in separation performance. Despite this, we expect to still be able to observe good PoS separation.

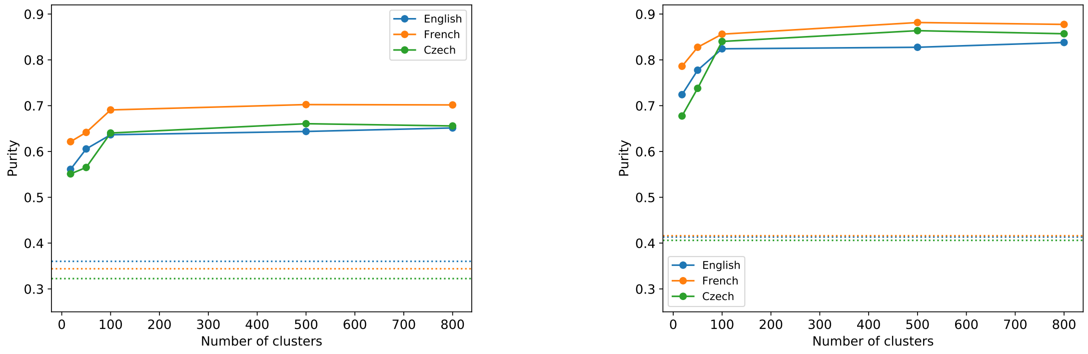
After filtering the EuroParl corpora, the size of remaining vocabulary is considerably larger: English ($60\,373\ words - 37.80\%$), French ($78\,822\ words - 38.60\%$), Czech ($62\,512\ words - 35.19\%$). In Figure 2a we can see that there is a drop in performance that varies with language, but when looking at purity, even in the worst performing clustering (French at $k = 50$), 60% of the vocabulary is still properly separated according to Parts of Speech. A drop in performance can also be observed for *AdjMI* in Figure 2b, with the value dropping for all languages, in some cases reducing by half.

More interestingly, the relative performance order of the languages is changed. PoS separation for Czech outperforms that of the other languages. Actually, PoS separation for Czech on EuroParl data (Figure 2) is scored higher than that of Czech on Universal Dependencies (Figures 1c and 1d). The source of this improvement requires more study for a proper attribution, but could be due to “beneficial” noise introduced by the automatic tagging, or due to the introduction of more sentence structure by human translators.

The fact that even at low values of k, for all languages studied, on both corpora, Exchange Word clusters (and also Brown word clusters, see supplementary material) can successfully separate by Parts of Speech, helps understand why word clusters have had such success at PoS tagging whether coupled with Markov Models (Derczynski et al., 2015), Markov Models and morphological features (Owoputi et al., 2013), or just by themselves via M-1 (Bansal et al., 2014).

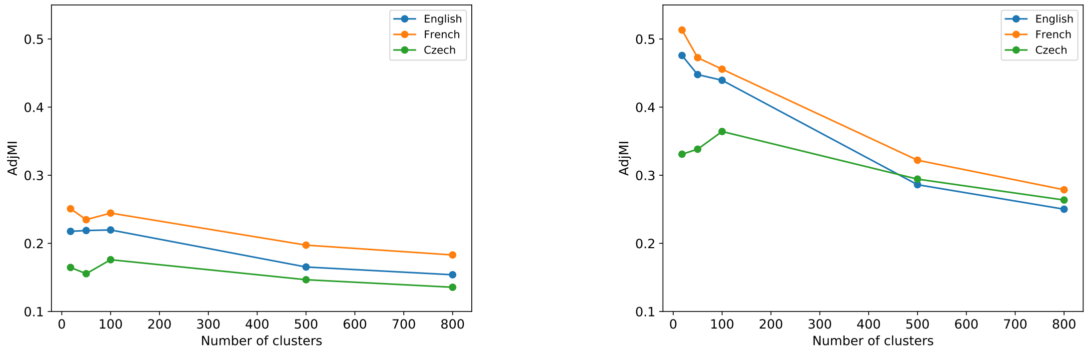
4.3 The relation between AMI and PoS

Neither Exchange, nor Brown are guaranteed to converge to a global optimum. Both are greedy algorithms that optimize for high Average Mutual Information (AMI). As we have mentioned earlier, word clusters resulting from Exchange outperform those induced using the Brown clustering algorithm in terms of both *AMI* (the algorithm’s optimization goal), *PoS purity* and *Adjusted Mutual Information (AdjMI)*. A natural question to ask is: can one improve the morphosyntactic content of word clusters by obtaining higher AMI, maybe by



(a) Cluster purity with baselines for $k = 1$. No frequency thresholding.

(c) Cluster purity with baselines for $k = 1$. Only words with frequency minimum 5.



(b) Adjusted Mutual Information. No frequency thresholding.

(d) Adjusted Mutual Information. Only words with frequency minimum 5.

Figure 1: Cluster agreement with manual labels from UD.

developing new and better AMI-based clustering algorithms?

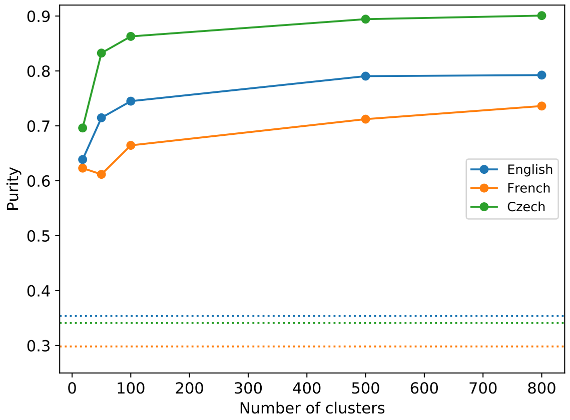
We answer this question by studying the correlation between *Average Mutual Information* and the two cluster quality measures used earlier: *purity* and *AdjMI*. Brown clustering is a predictable, bottom-up, agglomerative, hard clustering algorithm that for the same hyper-parameter k, generates the same clusters and therefore only one data sample.[2] However, the Exchange algorithm is an iterative clustering algorithm that has a complete and valid cluster partitioning at the end of each iteration. Thus, we can also measure morphosyntactic content in each of these clusterings. In our experiments, we only obtain 10 different data samples from each run of the algorithm, not enough for a correlation analysis.

In order to collect more data samples (i.e. more clusterings), we suggest using a stochastic version of Exchange where a percentage of all swaps are performed at random, rather than with the goal of improving AMI. This version of Exchange terminates based on the number of iterations, and generates valid word partitionings of varying quality (from an *AMI* perspective) at the end of each iteration. In this manner, it provides us with more data points (i.e. more different clusterings) for analysis. Due to the small amount of random swaps, at varying *AMI*, we obtain a sufficient number of distinct clusterings to perform a correlation study with sufficient data.

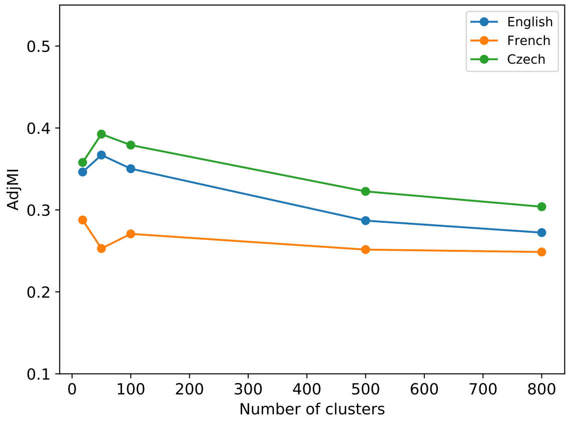
With the stochastic implementation of Exchange, we run 50 iterations for all languages and k combinations studied earlier. In Tables 2 and 3, we show the Pearson and Spearman correlation coefficients between AMI of all clusterings generated by StochasticExchange for a given run, and the two scores used earlier: *purity* and *AdjMI*. Due to space considerations, we only show results for $k = 18$ (i.e., same number of clusters as the number of PoS tags). Correlation coefficients for other combinations are included in the supplementary material. $p < 0.01$ for all correlation experiments here and in the supplementary material and for both correlation coefficients. The analysis presented below also holds for all the correlation re-

---

[2] Assuming a stable and repeatable tie-breaking process; this is undefined in the literature.



(a) Cluster purity. Dotted lines are baselines for $k = 1$



(b) Adjusted Mutual Information.

Figure 2: Cluster agreement with automatically generated labels from EuroParl. Only words with frequency minimum 5.

Data Set	Pearson	Spearman
EN UD	0.9776	0.7173
FR UD	0.9863	0.3976
CZ UD	0.9883	0.7378

Table 2: Correlation between Average Mutual Information and PoS purity of the clustering resulted from Exchange with $k = 18$. Words with frequency < 5 have been filtered. $p < 0.01$ for all coefficients.

Data Set	Pearson	Spearman
EN UD	0.9897	0.7464
FR UD	0.9845	0.7192
CZ UD	0.9859	0.8930

Table 3: Correlation between Average Mutual Information and AdjMI of the clustering resulted from Exchange with $k = 18$. Words with frequency < 5 have been filtered. $p < 0.01$ for all coefficients.

sults included in the supplementary material.

For both *purity* and *AdjMI*, there is a strong Pearson correlation between higher AMI and better values of the evaluation score. This is independent of the language studied or the number of clusters derived. For Spearman, except for one case, in all combinations studied, there is a high correlation, although to a slightly less extreme degree as with Pearson.

Our experiments show that there is strong correlation between *AMI* and performance in separation of Parts of Speech as measured by *purity* and *AdjMI*. The strong correlation provides grounding for research into new AMI-maximizing word clustering algorithms that can achieve higher AMI than Exchange, or Brown, as such algorithms might be able to separate Parts of Speech even better.

4.4 Effect of polysemy on cluster purity

In previous sections, we studied the ability of word clusters to encode morphosyntactic information. We clustered word types from unstructured text, where each token had its Part of Speech tag appended. The post-pended PoS tags are not used by either Brown, or Exchange. They are essentially invisible to the algorithms, since the both Brown and Exchange recognize words exclusively by internally assigned integer IDs and do not operate at character level.

However, post-pending PoS tags does introduce some information into the text by providing PoS-role disambiguation for each word occurrence.

For example, without post-pended PoS tags, both Exchange, and Brown algorithms, would conflate the two distinct grammatical roles of *show* in the sentence: “Everyone must show their show tickets at the entrance”. In this section, we study PoS separation effects caused by such polysemy on Brown and Exchange word clusters.

Both Exchange and Brown construct hard clusters, i.e. each word can be assigned to exactly one word class. Thus, words with multiple roles, such as denominal verbs or deverbal nouns, cannot be differentiated by the algorithms when operating on corpora from languages where the such morphological derivations are performed without employing suffixes or prefixes. In other words, if the lexical form does not change, neither Brown nor Exchange can identify which tokens represent what grammatical role.

The extent of this effect is dependent on language. In English, for example, nouns are often turned into verbs without changing the lexical form through morphological derivation, e.g. *show* as a verb vs *show* as a noun. On the other hand, Czech is highly inflected accounting for gender, case, number and person. This property of each language was not problematic in the experiments we have performed so far due to the fact that post-pending the PoS tag from ground-truth (or automatic tags) effectively provides disambiguation of grammatical role. Measuring on the Universal Dependencies corpora, we find that the percentage of polyclass words (i.e. word types that are assigned more than one PoS class tag throughout the corpus) varies by language and increases (as percentage of remaining vocabulary) as we raise the minimum frequency threshold, see Table 4. For English and French, up to 43% of the vocabulary words have more than one tag, while only $5,5\%$ of the Czech vocabulary shares the same property. Part of the reason why so many words have multiple PoS tags has to do with how the various language families derive new words, and part of the reason stems from errors in PoS tagging of large text corpora (Silberztein, 2018).

From a practical point of view, polysemous words create an upper bound on the effectiveness of hard clustering for Part of Speech separation (PoS). In Figure 3, we show PoS purity for clusters induced over Universal Dependencies (UD) corpora, where we consider all polyclass words as clustered incorrectly. We also show the minimum purity (when $k = 1$) as well as the upper bound given by the polysemy of each language as observed from the manual labels. The evaluation strictly penalizes multiclass polysemy and ignores errors in labeling, such as those identified by Silberztein (2018). For example, in the UD English corpus, even though only 3 occurrences of the word “them” are incorrectly tagged as adverb, while the remaining 750 are correctly labeled as pronoun. We defer to the data and consider the word to be impossible to correctly allocate to a cluster. We use such a strict evaluation as it provides a lower bound on what can be expected from Exchange and Brown clusters given the current data. Correcting PoS tags in the data would probably improve PoS separation, however, such corrections are outside the scope of the work in this paper.

We should point out that this evaluation is not representative of the expected PoS tagging performance of word clusters on any given corpus, as for such taggers one would employ a different strategy, such as, for instance, always outputting the most popular PoS tag for any given word type. On top of that, our evaluation here does not take into account the frequency of tokens, which would be highly relevant for PoS taggining performance, but not for our evaluation.

As expected, the most affected language is English, due to the high level of polysemy in the data. Here *purity* drops from 72.4 to 42.32 for $k = 18$, when compared with results in Figure 1c. It is followed by a 20 point drop for French, and only a few points for Czech, the most morphologically rich of the three and with the least amount of ambiguity in grammatical role. The results suggest that even in the presence of language ambiguity, and considering the strictest evaluation, Exchange and Brown clusters successfully encode a considerable amount of morphosyntactic information, which varies by language. These, together with results presented earlier in this paper provide empirical evidence for using word clusters as word representations in downstream NLP systems addressing tasks that rely on morphosyntactic knowledge of the language targeted (e.g. dependency parsing), or for use in new paradigms such as data programming (Ratner et al., 2016), where cluster membership can be a strong signal for probabilistic data labeling, even when considering language ambiguity.

Data Set	Min 1	Min 5
EN UD	15.39	43.02
FR UD	9.09	41.04
CZ UD	1.81	5.51

Table 4: Percentage of vocabulary with multiple PoS tags. Values are calculated relative to the vocabulary remaining after application of threshold.

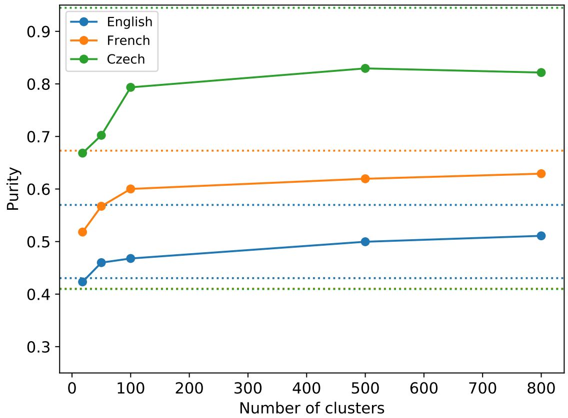


Figure 3: Cluster purity for manually annotated corpora from UD. Only words with frequency minimum 5. Dotted lines are baselines for $k = 1$ and highest achievable purity given polysemy in corpus.

5 Conclusion

In this paper, we quantified the amount of morphosyntactic information encoded in Brown and Exchange word clusters, in a number of languages, from different language families. Our empirical quantification helps explain the success of word clusters as word representations in NLP tasks that rely on morphosyntactic information, such as PoS tagging and Named Entity Recognition. It further provides empirical evidence for using word clusters as word representations in other NLP tasks that require morphosyntactic knowledge of the language targeted (e.g. dependency parsing), or for use in new paradigms such as data programming (Ratner et al., 2016), where cluster membership can be a strong signal for probabilistic data labeling. We have also shown that there is a strong correlation between *AMI* (Brown and Exchange's optimization goal) and performance in PoS separation. The strong correlation demonstrated provides grounding for research into new AMI-maximizing word representation algorithms that can achieve even better AMI optimization than Exchange or Brown.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring Continuous Word Representations for Dependency Parsing. In *Acl*, pages 809–815.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two Decades of Unsupervised POS Induction: How Far Have We Come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

Manuel Ciosici. 2015. Improving quality of hierarchical clustering for large data series. Master's thesis, Aarhus Universitet, Datalogisk Institut.

Jon Dehdari, Li Ling Tan, and Josef van Genabith. 2016. BIRA: Improved Predictive Exchange Word Clustering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL). Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2016), Ju*, pages 1169–1174.

Leon Derczynski and Sean Chester. 2016. Generalized Brown Clustering and Roll-up Feature Generation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1533–1539.

Leon Derczynski, Sean Chester, and Kenneth S Bøgh. 2015. Tune Your Brown Clustering, Please. In *Proceedings of the conference on Recent Advances in Natural Lang Processing (RANLP)*.

Reinhard Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Eurospeech*, volume 93, pages 973–976.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Terry Koo, Xavier Carreras Pérez, and Michael Collins. 2008. Simple semi-supervised dependency parsing. *46th Annual Meeting of the Association for Computational Linguistics*, 8(June):595–603.

Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, et al. 2017. Universal Dependencies 2.1.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah a Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT 2013*, June, pages 380–390. Association for Computational Linguistics.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big Data Small Data, In Domain Out-of Domain, Known Word Unknown Word: The Impact of Word Representation on Sequence Labelling Tasks. In *Proceedings of CoNLL*.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3567–3575. Curran Associates, Inc.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Max Silberztein. 2018. Using Linguistic Resources to Evaluate the Quality of Annotated Corpora. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 2–11.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Matthew C Swain and Jacqueline Manina Cole. 2016. ChemDataExtractor: A toolkit for automated extraction of chemical information from the scientific literature. *Journal of Chemical Information and Modeling*, page acs.jcim.6b00207.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation. In *Proceedings of ACL*, June, pages 755–762.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2009. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA. ACM.

Analyzing Bayesian Crosslingual Transfer in Topic Models

Shudong Hao
Boulder, CO
shudonghao@gmail.com

Michael J. Paul
Information Science
University of Colorado
Boulder, CO
mpaul@colorado.edu

Abstract

We introduce a theoretical analysis of crosslingual transfer in probabilistic topic models. By formulating posterior inference through Gibbs sampling as a process of language transfer, we propose a new measure that quantifies the loss of knowledge across languages during this process. This measure enables us to derive a PAC-Bayesian bound that elucidates the factors affecting model quality, both during training and in downstream applications. We provide experimental validation of the analysis on a diverse set of five languages, and discuss best practices for data collection and model design based on our analysis.

1 Introduction

Crosslingual learning is an important area of natural language processing that has driven applications including text mining in multiple languages (Ni et al., 2009; Smet and Moens, 2009), cultural difference detection (Gutiérrez et al., 2016), and various linguistic studies (Shutova et al., 2017; Barrett et al., 2016). Crosslingual learning methods generally extend monolingual algorithms by using various multilingual resources. In contrast to traditional high-dimensional vector space models, modern crosslingual models tend to rely on learning low-dimensional word representations that are more efficient and generalizable.

A popular approach to representation learning comes from the word embedding community, in which words are represented as vectors in an embedding space shared by multiple languages (Ruder et al., 2018; Faruqui and Dyer, 2014; Klementiev et al., 2012). Another direction is from the topic modeling community, where words are projected into a probabilistic topic space (Ma and Nasukawa, 2017; Jagarlamudi and III, 2010). While formulated differently, both types of models apply the same principles—low-dimensional vectors exist in a shared crosslingual space, wherein vector representations of similar concepts across languages (*e.g.,* "dog" and "hund") should be nearby in the shared space.

To enable crosslingual representation learning, knowledge is transferred from a source language to a target language, so that representations have similar values across languages. In this study, we will focus on probabilistic topic models, and "knowledge" refers to a word's probability distribution over topics. Little is known about the characteristics of crosslingual knowledge transfer in topic models, and thus this paper provides an analysis, both theoretical and empirical, of crosslingual transfer in multilingual topic models.

1.1 Background and Contributions

Multilingual Topic Models Given a multilingual corpus $D^{(1,\dots,L)}$ in languages $\ell = 1,\dots,L$ as inputs, a multilingual topic model learns K topics. Each multilingual topic $k^{(1,\dots,L)}$ ($k = 1,\dots,K$), is defined as an L-dimensional tuple $\left(\phi_k^{(1)},\dots,\phi_k^{(L)}\right)$, where $\phi_k^{(\ell)}$ is a multinomial distribution over the vocabulary $V^{(\ell)}$ in language ℓ. From a human's perspective, a multilingual topic $k^{(1,\dots,L)}$ can be interpreted by looking at the word types that have C highest probabilities in $\phi_k^{(\ell)}$ for each language ℓ. C here is called *cardinality* of the topic. Thus, a multilingual topic can loosely be thought of as a group of word lists where each language ℓ has its own version of the topic.

Multilingual topic models are generally extended from Latent Dirichlet Allocation (Blei et al., 2003, LDA). Though many variations have been proposed, the underlying structures of multilingual topic models are similar. These models require either a parallel/comparable corpus in multiple languages, or word translations from a

Proceedings of NAACL-HLT 2019, pages 1551–1565
Minneapolis, Minnesota, June 2 - June 7, 2019.

dictionary. One of the most popular models is the polylingual topic model (Mimno et al., 2009, PLTM), where comparable document pairs share distributions over topics θ, while each language ℓ has its own distributions $\{\phi_k^{(\ell)}\}_{k=1}^K$ over the vocabulary $V^{(\ell)}$. By re-marginalizing the estimations $\{\widehat{\phi}_k^{(\ell)}\}_{k=1}^K$, we obtain word representations $\widehat{\varphi}^{(w)} \in \mathbb{R}^K$ for each word w, where $\widehat{\varphi}_k^{(w)} = \Pr(z_w = k|w)$, *i.e.,* the probability of topic k given a word type w.

Crosslingual Transfer Knowledge transfer through crosslingual representations has been studied in prior work. Smet and Moens (2009) and Heyman et al. (2016) show empirically how document classification using topic models implements the ideas of crosslingual transfer, but to date there has been no theoretical framework to analyze this transfer process in detail.

In this paper, we describe two types of transfer—on-site and off-site—based on the nature of where and how the transfer takes place. We refer to transfer that happens while training topic models (*i.e.,* during representation learning) as *on-site*. Once we obtain the low-dimensional representations, they can be used for downstream tasks. We refer to transfer in this phase as *off-site*, since the crosslingual tasks are usually detached from the process of representation learning.

Contributions Our study provides a theoretical analysis of crosslingual transfer learning in topic models. Specifically, we first formulate on-site transfer as circular validation, and derive an upper bound based on PAC-Bayesian theories (Section 2). The upper bound explicitly shows the factors that can affect knowledge transfer. We then move on to off-site transfer, and focus on crosslingual document classification as a downstream task (Section 3). Finally, we show experimentally that the on-site transfer error can have impact on the performance of downstream tasks (Section 4).

2 On-Site Transfer

On-site transfer refers to the training procedure of multilingual topic models, which usually involves Bayesian inference techniques such as variational inference and Gibbs sampling. Our work focuses on the analysis of collapsed Gibbs sampling (Griffiths and Steyvers, 2004), showing how knowledge is transferred across languages and how a topic space is formed through the sampling process. To this end, we first describe a specific formulation of knowledge transfer in multilingual topic models as a starting point of our analysis (Section 2.1). We then formulate Gibbs sampling as circular validation and quantify a loss during this phase (Section 2.2). This formulation leads us to a PAC-Bayesian bound that explicitly shows the factors that affect the crosslingual training (Section 2.3). Lastly, we look further into different transfer mechanisms in more depth (Section 2.4).

2.1 Transfer through Priors

Priors are an important component in Bayesian models like PLTM. In the original generative process of PLTM, each comparable document pair (d_S, d_T) in the source and target languages (S, T) is generated by the same multinomial $\theta \sim \text{Dir}(\alpha)$.

Hao and Paul (2018) showed that knowledge transfer across languages happens through priors. Specifically, assume the source document is generated from $\theta^{(d_S)} \sim \text{Dir}(\alpha)$, and has a sufficient statistics $\mathbf{n}_{d_S} \in \mathbb{N}^K$ where each cell $n_{k|d_S}$ is the count of topic k in document d_S. When generating the corresponding comparable document d_T, the Dirichlet prior of the distribution over topics $\theta^{(d_T)}$, instead of a symmetric α, is parameterized by $\alpha + \mathbf{n}_{d_S}$. This formulation yields the same posterior estimation as the original joint model and is the foundation of our analysis in this section.

To see this transfer process more clearly, we look closer to the conditional distributions during sampling, and take PLTM as an example. When sampling a token in target language x_T, the Gibbs sampler calculates a conditional distribution $\mathcal{P}_{x_T}$ over K topics, where a topic k is randomly drawn and assigned to x_T (denoted as z_{x_T}). Assume the token x_T is in document d_T whose comparable document in the source language is d_S. The conditional distribution for x_T is

$$\begin{aligned}\mathcal{P}_{x,k} &= \Pr(z_x = k; \mathbf{w}_-, \mathbf{z}_-) \qquad (1)\\ &\propto \left(n_{k|d_T} + n_{k|d_S} + \alpha\right) \cdot \frac{n_{w_T|k} + \beta}{n_{\cdot|k} + V^{(T)}\beta},\end{aligned}$$

where the quantity $n_{k|d_S}$ is added and thus transferred from the source document. Thus, the calculation of $\mathcal{P}_x$ incorporates the knowledge transferred from the other language.

Now that we have identified the transfer process, we provide an alternative view of Gibbs sampling, *i.e.,* circular validation, in the next section.

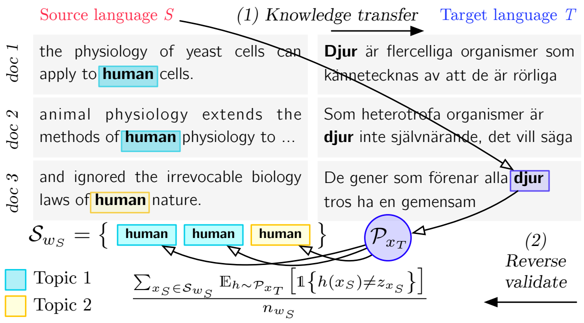


Figure 1: The Gibbs sampler is sampling the token "djur" (animal). Using the classifier h_k sampled from its conditional distribution $\mathcal{P}_{x_T}$, circular validation evaluates h_k on all the tokens of type "human".

2.2 Circular Validation

Circular validation (or reverse validation) was proposed by Zhong et al. (2010) and Bruzzone and Marconcini (2010) in transfer learning. Briefly, a learning algorithm $\mathcal{A}$ is trained on both source and target datasets (D_S and D_T), where the source is labeled and target is unlabeled. After predicting the labels for the target dataset using $\mathcal{A}$ (predictions denoted as $\mathcal{A}(D_T)$), circular validation trains another algorithm $\mathcal{A}'$ in the reverse direction, *i.e.,* uses $\mathcal{A}(D_T)$ and D_T as the labeled dataset and D_S as the unlabeled dataset. The error is then evaluated on $\mathcal{A}'(D_S)$. This "train-predict-reverse-repeat" cycle has a similar flavor to the iterative manner of Gibbs sampling, which inspires us to look at the sampling process as circular validation.

Figure 1 illustrates this process. Suppose the Gibbs sampler is currently sampling x_T of word type w_T in target language T. As discussed for Equation (1), the calculation of the conditional distribution $\mathcal{P}_{x_T}$ incorporates the knowledge transferred from the source language. We then treat the process of drawing a topic from $\mathcal{P}_{x_T}$ as a *classification* of the token x_T. Let $\mathcal{P}_{x_T}$ be a distribution over K unary *classifiers*, $\{h_k\}_{k=1}^{K}$, and the k-th classifier labels the token as topic k with a probability of one:

$$h_k \sim \mathcal{P}_{x_T}, \text{ and } \Pr\left(z_{x_T} = k; h_k\right) = 1. \quad (2)$$

This process is repeated between the two languages until the Markov chain converges.

The training of topic models is unsupervised, *i.e.,* there is no ground truth for labeling a topic, which makes it difficult to analyze the effect of transfer learning. Thus, after calculating $\mathcal{P}_{x_T}$, we take an additional step called *reverse validation*, where we design and calculate a measure—circular validation loss—to quantify the transfer.

Definition 1 (Circular validation loss, CVL)**.** *Let $\mathcal{S}_w$ be the set containing all the tokens of type w throughout the whole training corpus, and call it the sample of w. Given a bilingual word pair (w_T, w_S) where w_T is in target language T while w_S in source S, let $\mathcal{S}_{w_T}$ and $\mathcal{S}_{w_S}$ be the samples for the two types respectively, and n_{w_T} and n_{w_S} the sizes of them. The empirical circular validation score ($\widehat{\text{CVL}}$) is defined as*

$$\begin{aligned}
\widehat{\text{CVL}}(w_T, w_S) &= \frac{1}{2} \mathop{\mathbb{E}}_{x_S, x_T} \left[\widehat{\mathcal{L}}(x_T, w_S) + \widehat{\mathcal{L}}(x_S, w_T) \right], \\
\widehat{\mathcal{L}}(x_T, w_S) &= \frac{1}{n_{w_S}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathbb{E}_{h \sim \mathcal{P}_{x_T}} \left[\mathbb{1}\left\{ h(x_S) \neq z_{x_S} \right\} \right] \\
&= \frac{1}{n_{w_S}} \sum_{x_S \in \mathcal{S}_{w_S}} \left(1 - \mathcal{P}_{x_T, z_{x_S}} \right),
\end{aligned}$$

where $\mathcal{P}_{x_T,k}$ is the conditional probability of token x_T assigned with topic k. Taking expectations over all tokens x_S and x_T, we have general CVL*:*

$$\begin{aligned}
\text{CVL}(w_T, w_S) &= \frac{1}{2} \mathop{\mathbb{E}}_{x_S, x_T} \left[\mathcal{L}(x_T, w_S) + \mathcal{L}(x_S, w_T) \right], \\
\mathcal{L}(x_T, w_S) &= \mathbb{E}_{x_S} \mathbb{E}_{h \sim \mathcal{P}_{x_T}} \left[\mathbb{1}\left\{ h(x_S) \neq z_{x_S} \right\} \right].
\end{aligned}$$

When sampling a token x_T, we still follow the two-step process as in Equation (2), but instead of labeling x_T itself, we use its conditional $\mathcal{P}_{x_T}$ to label the entire sample of a word type w_S in the source language. Since all the topic labels for the source language are fixed, we take them as the assumed "correct" labelings, and compare x_S's labels and the predictions from $\mathcal{P}_{x_T}$. This is the intuition behind CVL.

Note that the choice of word types w_T and w_S to calculate $\widehat{\text{CVL}}$ is arbitrary. However, $\widehat{\text{CVL}}$ is only meaningful when the two word types are semantically related, such as word translations, because those word pairs are where the knowledge transfer takes place. On the other hand, the Gibbs sampler does not calculate this $\widehat{\text{CVL}}$ explicitly, and thus adding reverse validation step does not affect the training of the model. It does, however, help us to expose and analyze the knowledge transfer mechanism. In fact, as we show in the next theorem, sampling is also a procedure of optimizing $\widehat{\text{CVL}}$.

Theorem 1. *Let $\widehat{\text{CVL}}^{(t)}(w_T, w_S)$ be the empirical circular validation loss of any bilingual word pair at iteration t of Gibbs sampling. Then $\widehat{\text{CVL}}^{(t)}(w_T, w_S)$ converges as $t \to \infty$.*

Proof. See Appendix. □

2.3 PAC-Bayes View

A question following the formulation of $\widehat{\text{CVL}}$ is, what factors could lead to better transfer during this process, particularly for semantically related words? To answer this, we turn to theory that bounds the performance of classifiers and apply this theory to this formulation of topic sampling as classification.

The PAC-Bayes theorem was introduced by McAllester (1999) to bound the performance of Bayes classifiers. Given a hypothesis set $\mathcal{H}$, the majority vote classifier (or Bayes classifier) uses every hypothesis $h \in \mathcal{H}$ to perform binary classification on an example $\mathbf{x}$, and uses the majority output as the final prediction. Since minimizing the error by Bayes classifier is NP-hard, an alternative way is to use a *Gibbs classifier* as approximation. The Gibbs classifier first draws a hypothesis $h \in \mathcal{H}$ according to a posterior distribution over $\mathcal{H}$, and then uses this hypothesis to predict the label of an example $\mathbf{x}$ (Germain et al., 2012). The generalization loss of this Gibbs classifier can be bounded as follows.

Theorem 2 (PAC-Bayes theorem, McAllester (1999))**.** *Let $\mathcal{P}$ be a posterior distribution over all classifiers $h \in \mathcal{H}$, and Q a prior distribution. With a probability at least $1-\delta$, we have*

$$\mathcal{L} \le \widehat{\mathcal{L}} + \sqrt{\frac{1}{2n}\left(\mathrm{KL}\left(\mathcal{P}||Q\right) + \ln\frac{2\sqrt{n}}{\delta}\right)},$$

where $\mathcal{L}$ and $\widehat{\mathcal{L}}$ are the general loss and the empirical loss on a sample of size n.

In our framework, a token x_T provides a posterior $\mathcal{P}_{x_T}$ over K classifiers. The loss $\widehat{\mathcal{L}}(x_T, w_S)$ is then calculated on a sample of $\mathcal{S}_{w_S}$ in language S. The following theorem shows that for a bilingual word pair (w_T, w_S), the general CVL can be bounded with several quantities.

Theorem 3. *Given a bilingual word pair (w_T, w_S), with probability at least $1-\delta$, the following bound holds:*

$$\begin{gathered}\text{CVL}(w_T, w_S) \le \widehat{\text{CVL}}(w_T, w_S) + \qquad (3)\\ \frac{1}{2}\sqrt{\frac{1}{n}\left(\mathrm{KL}_{w_T} + \mathrm{KL}_{w_S} + 2\ln\frac{2}{\delta}\right)} + \frac{\ln n^\star}{n},\\ n = \min\left\{n_{w_T}, n_{w_S}\right\}, \quad n^\star = \max\left\{n_{w_T}, n_{w_S}\right\}.\end{gathered}$$

For brevity we use KL_w to denote $\mathrm{KL}(\mathcal{P}_x||Q_x)$, where $\mathcal{P}_x$ is the conditional distribution from Gibbs sampling of token x with word type w that gives highest loss $\widehat{\mathcal{L}}(x,w)$, and Q_x a prior.

Proof. See Appendix. □

2.4 Multilevel Transfer

Recall that knowledge transfer happens through priors in topic models (Section 2.1). Because the KL-divergence terms in Theorem 3 include this prior Q, we can use this theorem to analyze the transfer mechanisms more concretely.

The conditional distribution for sampling a topic z_x for a token x during sampling can be factorized into document-topic and topic-word levels:

$$\begin{aligned}\mathcal{P}_{x,k} &= \Pr\left(z_x = k | w_x = w, \mathbf{w}_-, \mathbf{z}_-\right)\\ &= \Pr\left(z_x = k | \mathbf{z}_-\right) \cdot \Pr\left(w_x = w | z_x = k, \mathbf{w}_-, \mathbf{z}_-\right)\\ &\propto \underbrace{\Pr\left(z_x = k | \mathbf{z}_-\right)}_{\text{document level}} \cdot \underbrace{\Pr\left(z_x = k | w_x = w, \mathbf{w}_-\right)}_{\text{word level}}\\ &\triangleq \mathcal{P}_{\theta,x,k} \cdot \mathcal{P}_{\varphi,x,k},\\ \mathcal{P}_x &\triangleq \mathcal{P}_{\theta,x} \otimes \mathcal{P}_{\varphi,x},\end{aligned}$$

where $\otimes$ is element-wise multiplication. Thus, we have the following inequality:

$$\begin{aligned}\mathrm{KL}\left(\mathcal{P}_x||Q_x\right) &= \mathrm{KL}\left(\mathcal{P}_{\theta,x} \otimes \mathcal{P}_{\varphi,x}||Q_{\theta,x} \otimes Q_{\varphi,x}\right)\\ &\le \mathrm{KL}\left(\mathcal{P}_{\theta,x}||Q_{\theta,x}\right) + \mathrm{KL}\left(\mathcal{P}_{\varphi,x}||Q_{\varphi,x}\right),\end{aligned}$$

and the KL-divergence term in Theorem 3 is simply the sum of the KL-divergences between the conditional and prior distributions on all levels.

Recall that PLTM transfers knowledge at the document level, through $Q_{\theta,x}$, by linking document translations together (Equation (1)). Assume the current token x is from a target document linked to a document d_S in the source language. Then the prior for $\mathcal{P}_{\theta,x}$ is $\widehat{\theta}^{(d_S)}$, *i.e.,* the normalized empirical distribution over topics of d_S.

Since the words are generated *within* each language under PLTM, *i.e.,* $\phi_k^{(S)}$ is irrelevant to $\phi_k^{(T)}$, no transfer happens at the word level. In this case, $Q_{\varphi,x}$, the prior for $\mathcal{P}_{\varphi,x}$, is simply a K-dimensional uniform distribution $\mathcal{U}$. Then:

$$\begin{aligned}\mathrm{KL}_w &\le \mathrm{KL}\left(\mathcal{P}_{\theta,x}||\widehat{\theta}^{(d_S)}\right) + \mathrm{KL}\left(\mathcal{P}_{\varphi,x}||\mathcal{U}\right)\\ &= \underbrace{\mathrm{KL}\left(\mathcal{P}_{\theta,x}||\widehat{\theta}^{(d_S)}\right)}_{\text{crosslingual entropy}} + \log K - \underbrace{H(\mathcal{P}_{\varphi,x})}_{\text{monolingual entropy}}.\end{aligned}$$

Thus, at levels where transfer happens (document- or word-level), a low crosslingual entropy is preferred, to offset the impact of monolingual entropy where no transfer happens.

Most multilingual topic models are generative admixture models in which the conditional probabilities can be factorized into different levels, thus KL-divergence term in Theorem 3 can be decomposed and analyzed in the same way as in this section for models that have transfer at other levels, such as Hao and Paul (2018), Heyman et al. (2016), and Hu et al. (2014). For example, if a model has word-level transfer, *i.e.,* the model assumes that word translations share the same distributions, we have a KL-divergence term as,

$$\begin{aligned}\mathrm{KL}_w &\le \mathrm{KL}\left(\mathcal{P}_{\varphi,x}||\widehat{\varphi}^{(w_S)}\right) + \mathrm{KL}(\mathcal{P}_{\theta,x}||\mathcal{U}) \\ &= \mathrm{KL}\left(\mathcal{P}_{\varphi,x}||\widehat{\varphi}^{(w_S)}\right) + \log K - H(\mathcal{P}_{\theta,x}),\end{aligned}$$

where w_S is the word translation to word w.

3 Off-Site Transfer

Off-site transfer refers to language transfer that happens while applying trained topic models to downstream crosslingual tasks such as document classification. Because transfer happens using the trained representations, the performance of off-site transfer heavily depends on that of on-site transfer. To analyze this problem, we focus on the task of crosslingual document classification.

In crosslingual document classification, a *document classifier*, h, is trained on documents from one language, and h is then applied to documents from another language. Specifically, after training bilingual topic models, we have K bilingual word distributions $\{\widehat{\phi}_k^{(S)}\}_{k=1}^K$ and $\{\widehat{\phi}_k^{(T)}\}_{k=1}^K$. These two distributions are used to infer document-topic distributions $\widehat{\theta}$ on unseen documents in the test corpus, and each document is represented by the inferred distributions. A document classifier is then trained on the $\widehat{\theta}$ vectors as features in source language S and tested on the target T.

We aim to show how the generalization risk on target languages T, denoted as $R_T(h)$, is related to the training risk on source languages S, $\widehat{R_S}(h)$. To differentiate the loss and classifiers in this section from those in Section 2, we use the term "risk" here, and h refers to the document classifiers, not the topic labeling process by the sampler.

3.1 Languages as Domains

Classic learning theory requires training and test sets to come from the same distribution $\mathcal{D}$, *i.e.,* $(\theta, y) \sim \mathcal{D}$, where θ is the document representation (features) and y the document label (Valiant, 1984). In practice, however, corpora in S and T may be sampled from different distributions, *i.e.,* $D^{(S)} = \{(\widehat{\theta}^{(d_S)}, y)\} \sim \widehat{\mathcal{D}}^{(S)}$ and $D^{(T)} = \{(\widehat{\theta}^{(d_T)}, y)\} \sim \widehat{\mathcal{D}}^{(T)}$. We refer to these distributions as *document spaces.* To relate $R_T(h)$ and $\widehat{R_S}(h)$, therefore, we have to take their distribution bias into consideration. This is often formulated as a problem of domain adaptation, and here we can formulate this such that each language is treated as a "domain".

We follow the seminal work by Ben-David et al. (2006), and define $\mathcal{H}$-distance as follows.

Definition 2 ($\mathcal{H}$-distance, Ben-David et al. (2006))**.** *Let $\mathcal{H}$ be a symmetric hypothesis space,* i.e., *for every hypothesis $h \in \mathcal{H}$ there exists its counterpart $1-h \in \mathcal{H}$. We let $m = \left|D^{(S)}\right| + \left|D^{(T)}\right|$, the total size of test corpus. The $\mathcal{H}$-distance between $\widehat{\mathcal{D}}^{(S)}$ and $\widehat{\mathcal{D}}^{(T)}$ is defined as*

$$\begin{aligned}&\frac{1}{2}\widehat{d}_{\mathcal{H}}\left(\widehat{\mathcal{D}}^{(S)}, \widehat{\mathcal{D}}^{(T)}\right) \\ &= \max_{h\in\mathcal{H}} \frac{1}{m} \sum_{\ell\in\{S,T\}} \sum_{\mathbf{x}_d : h(\mathbf{x}_d)=\ell} \mathbb{1}\left\{\mathbf{x}_d \in D^{(\ell)}\right\},\end{aligned}$$

where $\mathbf{x}_d$ is the representation for document d, and $h(\mathbf{x}_d)$ outputs the language of this document.

This distance measures how identifiable the languages are based on their representations. If source and target languages are from entirely different distributions, a classifier can easily identify language-specific features, which could affect performance of the document classifiers.

With $\mathcal{H}$-distances, we have a measure of the "distance" between the two distributions $\widehat{\mathcal{D}}^{(S)}$ and $\widehat{\mathcal{D}}^{(T)}$. We state the following theorem from domain adaptation theory.

Theorem 4 (Ben-David et al. (2006))**.** *Let m be the corpus size of the source language,* i.e., $m = \left|D^{(S)}\right|$, *c the VC-dimension of document classifiers $h \in \mathcal{H}$, and $\widehat{d}_{\mathcal{H}}\left(\widehat{\mathcal{D}}^{(S)}, \widehat{\mathcal{D}}^{(T)}\right)$ the $\mathcal{H}$-distance between two languages in the document space. With probability at least $1-\delta$, we have the following bound,*

$$\begin{aligned}R_T(h) &\le \widehat{R}_S(h) + \widehat{d}_{\mathcal{H}}\left(\widehat{\mathcal{D}}^{(S)}, \widehat{\mathcal{D}}^{(T)}\right) + \widehat{\lambda} + \\ &\qquad \sqrt{\frac{4}{m}\left(c\log\frac{2em}{c} + \log\frac{4}{\delta}\right)}, \qquad (4)\\ \widehat{\lambda} &= \min_{h\in\mathcal{H}} \widehat{R}_S(h) + \widehat{R}_T(h). \qquad (5)\end{aligned}$$

The term $\widehat{\lambda}$ in Theorem 4 defines a *joint risk*, *i.e.,* the training error on both source and target documents. This term usually cannot be estimated in practice since the labels for target documents are unavailable. However, we can still calculate this term for the purpose of analysis.

The theorem shows that the crosslingual classification risk is bounded by two critical components: the $\mathcal{H}$-distance, and the joint risk $\widehat{\lambda}$. Interestingly, these two quantities are based on the same set of features with different labeling rules: for $\mathcal{H}$-distance, the label for each instance is its language, while $\widehat{\lambda}$ uses the actual document label. Therefore, a better bound requires the consistency of features across languages, both in language and document labelings.

3.2 From Words to Documents

Since consistency of features depends on the document representations $\widehat{\theta}$, we need to trace back to the upstream training of topic models and show how the errors propagate to the formation of document representations. Thus, we first show the relations between $\widehat{\text{CVL}}$ and word representations $\widehat{\varphi}$ in the following lemma.

Lemma 1. *Given any bilingual word pair* (w_T, w_S), *let* $\widehat{\varphi}^{(w)}$ *denote the distribution over topics of word type* w. *Then we have,*

$$1 - \widehat{\varphi}^{(w_T)\top} \cdot \widehat{\varphi}^{(w_S)} \quad \leq \quad \widehat{\text{CVL}}(w_T, w_S).$$

Proof. See Appendix. □

We need to connect the word representations $\widehat{\varphi}$, which are central to on-site transfer, to the document representations $\widehat{\theta}$, which are central to off-site transfer. To do this, we make an assumption that the inferred distribution over topics $\widehat{\theta}^{(d)}$ for each test document d is a weighted average over all word vectors, *i.e.,* $\widehat{\theta}^{(d)} \propto \sum_w f_w^d \cdot \widehat{\varphi}^{(w)}$, where f_w^d is the normalized frequency of word w in document d (Arora et al., 2013). When this assumption holds, we can bound the similarity of document representations $\widehat{\theta}^{(d_S)}$ and $\widehat{\theta}^{(d_T)}$ in terms of word representations and hence their $\widehat{\text{CVL}}$.

Theorem 5. *Let* $\widehat{\theta}^{(d_S)}$ *be the distribution over topics for document* d_S *(similarly for* d_T*),* $F(d_S, d_T) = \left(\sum_{w_S} {f_{w_S}^{d_S}}^2 \cdot \sum_{w_T} {f_{w_T}^{d_T}}^2\right)^{\frac{1}{2}}$ *where* f_w^d *is the* normalized *frequency of word* w *in document* d, *and* K *the number of topics. Then*

$$\widehat{\theta}^{(d_S)\top} \cdot \widehat{\theta}^{(d_T)} \leq F(d_S, d_T) \cdot \sqrt{K \cdot \sum_{w_S, w_T} \left(\widehat{\text{CVL}}(w_T, w_S) - 1\right)^2}.$$

Proof. See Appendix. □

This provides a spatial connection between document pairs and word pairs they have. Many kernalized classifiers such as support vector machines (SVM) explicitly use this inner product in the dual optimization objective (Platt, 1998). Since the inner product is directly related to the cosine similarity, Theorem 5 indicates that if two documents are spatially close, their inner product should be large, and thus the $\widehat{\text{CVL}}$ of all word pairs they share should be small. In an extreme case, if $\widehat{\text{CVL}}(w_T, w_S) = 1$ for all the bilingual word pairs appearing in document pair (d_S, d_T), then $\widehat{\theta}^{(d_S)\top} \cdot \widehat{\theta}^{(d_T)} = 0$, meaning the two documents are orthogonal and tend to be irrelevant topically.

With upstream training discussed in Section 2, we see that $\widehat{\text{CVL}}$ has an impact on the consistency of features across languages. A low $\widehat{\text{CVL}}$ indicates that the transfer from source to target is sufficient in two ways. First, languages share similar distributions, and therefore, it is harder to distinguish languages based on their distributions. Second, if there exists a latent mapping from a distribution to a label, it should produce similar labeling on both source and target data since they are similar. These two aspects correspond to the language $\mathcal{H}$-distance and joint risk $\widehat{\lambda}$ in Theorem 4.

4 Experiments

We experiment with five languages: Arabic (AR, Semitic), German (DE, Germanic), Spanish (ES, Romance), Russian (RU, Slavic), and Chinese (ZH, Sinitic). In the first two experiments, we pair each with English (EN, Germanic) and train PLTM on each language pair individually.

Training Data For each language pair, we use a subsample of 3,000 Wikipedia comparable documents, *i.e.,* 6,000 documents in total. We set $K = 50$, and train PLTM with default hyperparameters (McCallum, 2002). We run each experiment five times and average the results.

Test Data For experiments with document classification, we use Global Voices (GV) in all five language pairs as test sets. Each document in this dataset has a "categories" attribute that can be used as the document label. In our classification experiments, we use *culture*, *technology*, and *education* as the labels to perform multiclass classification.

Evaluation To evaluate topic qualities, we use Crosslingual Normalized Pointwise Mutual Information (Hao et al., 2018, CNPMI), an intrinsic metric of crosslingual topic coherence. For any bilingual word pair (w_T, w_S),

$$\text{CNPMI}(w_T, w_S) = -\frac{\log \frac{\Pr(w_T, w_S)}{\Pr(w_T)\Pr(w_S)}}{\log \Pr(w_T, w_S)}, \quad (6)$$

where $\Pr(w_T, w_S)$ is the occurrence of w_T and w_S appearing in the same pair of comparable documents. We use 10,000 Wikipedia comparable document pairs outside PLTM training data for each language pair to calculate CNPMI scores. All datasets are publicly available at `http://opus.nlpl.eu/` (Tiedemann, 2012). Additional details of our datasets and experiment setup can be found in the appendix.

4.1 Sampling as Circular Validation

Our first experiment shows how $\widehat{\text{CVL}}$ changes over time during Gibbs sampling. According to the definition, the arguments of $\widehat{\text{CVL}}$ can include any bilingual word pairs; however, we suggest that it should be calculated specifically among word pairs that are expected to be related (and thus enable transfer). In our experiments, we select word pairs in the following way.

Recall that the output of a bilingual topic model is K topics, where each language has its own distribution. For each topic k, we can calculate $\widehat{\text{CVL}}(w_S, w_T)$ such that w_S and w_T belong to the same topic (*i.e.,* are in the top C most probable words in that topic), from the two languages, respectively. Using a cardinality C for each of the K topics, we have in total $C^2 \times K$ bilingual word pairs in the calculation of $\widehat{\text{CVL}}$.

At certain iterations, we collect the topic words as described above with cardinality $C = 5$, and calculate $\widehat{\text{CVL}}(w_T, w_S)$, $\text{CNPMI}(w_T, w_S)$, and the error term (the $\frac{1}{2}\sqrt{\cdots}$ term in Theorem 3) of all the bilingual word pairs. In the middle panel of Figure 2, $\widehat{\text{CVL}}$ over all word pairs from topic words is decreasing as sampling proceeds and becomes stable by the end of sampling. On the other hand, the correlations between CNPMI and $\widehat{\text{CVL}}$ are constantly decreasing. The negative correlations between $\widehat{\text{CVL}}$ and CNPMI implies that lower $\widehat{\text{CVL}}$ is associated with higher topic quality, since higher-quality topic has higher CNPMI but lower $\widehat{\text{CVL}}$.

4.2 What the PAC-Bayes Bound Shows

Theorem 3 provides insights into how knowledge is transferred during sampling and the factors that could affect this process. We analyze this bound from two aspects, the size of the training data (corresponding to $\frac{\ln n^\star}{n}$ term) and model assumptions (as in the crosslingual entropy terms).

4.2.1 Training Data: Downsampling

One factor that could affect $\widehat{\text{CVL}}$, according to Theorem 3, is the balance of tokens of a word pair. In an extreme case, if a word type w_S has only one token, while another word type w_T has a large number of tokens, the transfer from w_S to w_T is negligible. In this experiment, we will test if increasing the ratio term $\frac{\ln n^\star}{n}$ in the corpus lowers the performance of crosslingual transfer learning.

To this end, we specify a *sample rate* $\rho = 0.2, 0.4, 0.6, 0.8$, and 1.0. For each word pair (w_T, w_S), we calculate n as in the ratio term $\frac{\ln n^\star}{n}$, and remove $(1-\rho)\cdot n$ tokens from the corpus (rounded to the nearest integer). Smaller ρ removes more tokens from the corpus and thus yields a larger ratio term on average.

We use a dictionary from Wiktionary to collect word pairs, where each word pair (w_S, w_T) is a translation pair. Figure 3 shows the results of downsampling using these two methods. Decreasing the sample rate ρ lowers the topic qualities. This implies that although PLTM can process comparable corpora, which need not be exact translations, one still needs to be careful about the token balance between linked document pairs.

For many low-resource languages, the target language corpus is much smaller than the source corpus, so the effect of this imbalance is important to be aware of. This is an important issue when choosing comparable documents, and Wikipedia is an illustrative example. Although one can collect comparable documents via Wikipedia's interlanguage links, articles under the same title but in different languages can have very large variations on document length, causing the imbalance of samples $\frac{\ln n^\star}{n}$, and thus potentially suboptimal performance of crosslingual training.

4.2.2 Model Assumptions

Recall that the crosslingual entropy term can be decomposed into different levels, *e.g.,* document

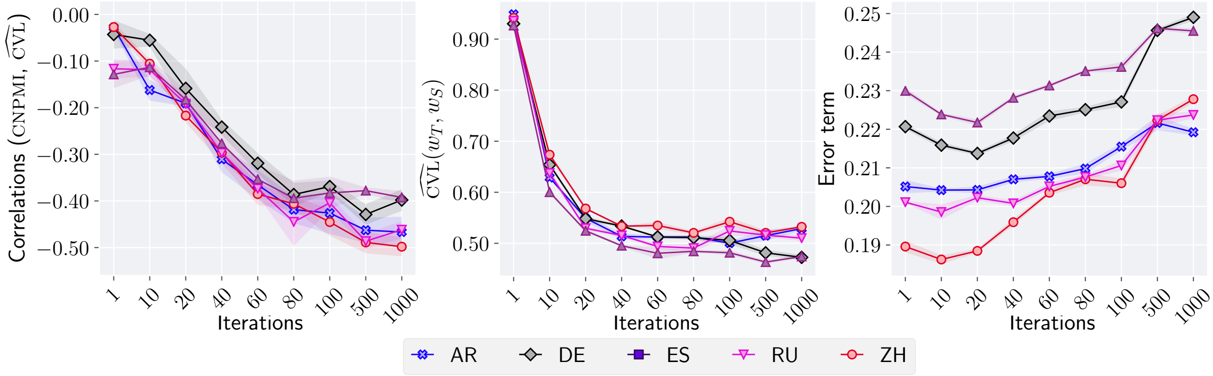


Figure 2: As Gibbs sampling progresses, $\widehat{\mathrm{CVL}}$ of topic words drops, which leads to higher quality topics, and thus increases CNPMI. The left panel shows this negative correlation, and we use shades to indicate standard deviations across five chains.

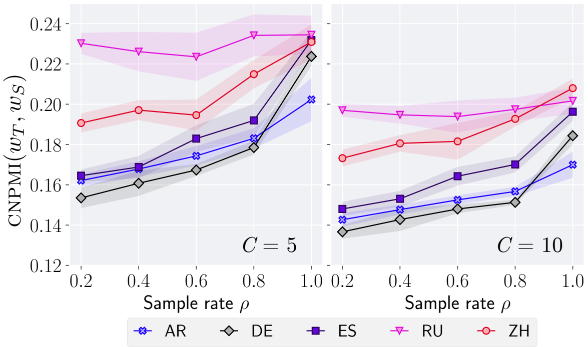


Figure 3: Increasing ρ results in smaller values of $\frac{\ln n^\star}{n}$ for translation pairs. Topic quality, evaluated by CNPMI, increases as well.

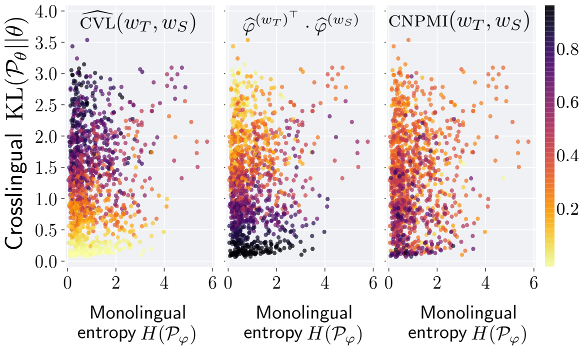


Figure 4: Each dot is a (EN,DE) word pair, and its color shows corresponding values of the indicated quantity. Best viewed in color.

level and word level, and we prefer a model with low crosslingual entropy but high monolingual entropy. In this experiment, we show how these two quantities affect the topic qualities, using English-German (EN-DE) documents as an example.

Given PLTM output in (EN,DE) and a cardinality $C = 5$, we collect $C^2 \times K$ bilingual word pairs as described in Section 4.1. For each word pair, we calculate three quantities: $\widehat{\mathrm{CVL}}$, CNPMI, and the inner product of the word representations. In Figure 4, each dot is a word pair (w_S, w_T) colored by the values of these quantities. The word pair dots are positioned by their crosslingual and monolingual entropies.

We observe that $\widehat{\mathrm{CVL}}$ decreases with crosslingual entropy on document level. The larger the crosslingual entropy, the harder it is to get a low $\widehat{\mathrm{CVL}}$ because it needs larger monolingual entropy to decrease the bound, as shown in Section 2.4. On the other hand, the inner product of word pairs shows an opposite pattern of $\widehat{\mathrm{CVL}}$, indicating a negative correlation (Lemma 1). In Figure 2 we see the correlation between CNPMI and $\widehat{\mathrm{CVL}}$ is around -0.4 at the end of sampling, so there are fewer clear patterns for CNPMI in Figure 4. However, we also notice that the word pairs with higher CNPMI scores often appear at the bottom where crosslingual entropy is low while the monolingual entropy is high.

4.3 Downstream Task

We move on to crosslingual document classification as a downstream task. At various iterations of Gibbs sampling, we infer topics on the test sets for another 500 iterations and calculate the quantities shown in the Figure 5 (averaged over all languages), including the $\mathcal{H}$-distances for both training and test sets, and the joint risk $\widehat{\lambda}$.

We treat English as the source language and train support vector machines to obtain the best classifier $h^\star$ that fits the English documents. This classifier is then used to calculate the source and target risks $\widehat{R}_S(h^\star)$ and $\widehat{R}_T(h^\star)$. We also include $\frac{1}{2}\widehat{d}_{\mathcal{H}}(S, T)$, the $\mathcal{H}$-distance based on word rep-

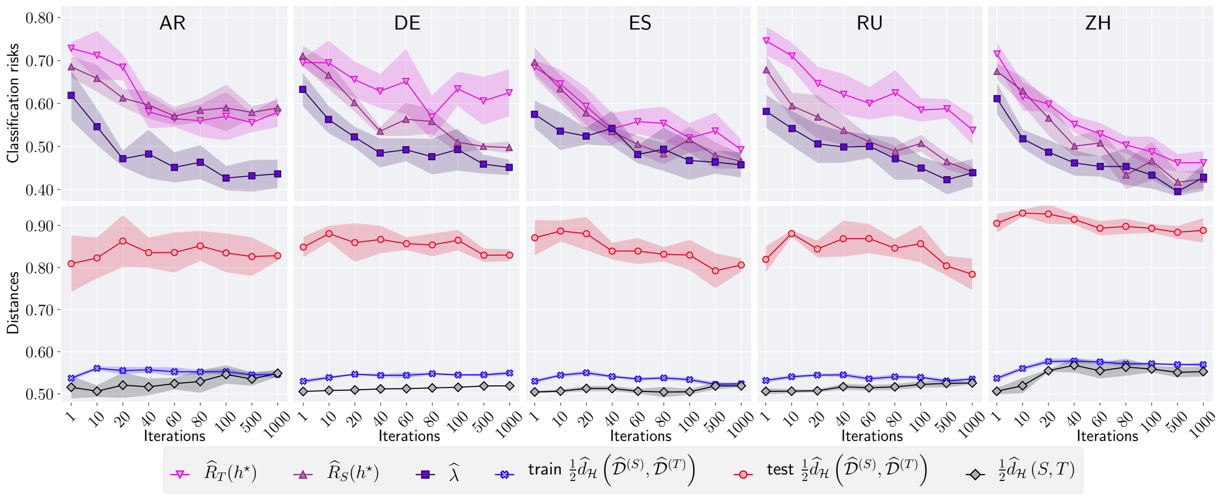


Figure 5: Gibbs sampling optimizes $\widehat{\text{CVL}}$, which decreases the joint risk $\widehat{\lambda}$ and $\mathcal{H}$-distances for test data.

resentations $\widehat{\varphi}$. As mentioned in Section 3.1, we train support vector machines to use languages as labels, and the accuracy score as the $\mathcal{H}$-distance.

The classification risks, such as $\widehat{R}_S(h^\star)$, $\widehat{R}_T(h^\star)$, and $\widehat{\lambda}$, are decreasing as expected (upper row in Figure 5), which shows very similar trends as $\widehat{\text{CVL}}$ in Figure 2. On the other hand, we notice that the $\mathcal{H}$-distances of training documents and vocabularies, $\frac{1}{2}\widehat{d}_{\mathcal{H}}\left(\widehat{\mathcal{D}}^{(S)}, \widehat{\mathcal{D}}^{(T)}\right)$ and $\frac{1}{2}\widehat{d}_{\mathcal{H}}(S, T)$, stabilize around 0.5 to 0.6, meaning it is difficult to differentiate the languages based on their representations. Interestingly, the $\mathcal{H}$-distances of test documents are at a less ideal value, although they are slightly decreasing in most of the languages except AR. However, recall that the target risk also depends on other factors than $\mathcal{H}$-distance (Theorem 4), and we use Figure 6 to illustrate this point.

We further explore the relationship between the predictability of languages vs document classes in Figure 6. We collect documents correctly classified for both document class and language labels, from which we randomly choose 200 documents for each language, and use $\widehat{\theta}$ to plot t-SNE scatterplots. Note that the two plots are from the same set of documents, and so the spatial relations between any two points are fixed, but we color them with different labelings. Although the classifier can identify the languages (right panel), the features are still consistent, because on the left panel, the decision boundary changes its direction and also successfully classifies the documents based on actual label class. This illustrates why a single $\mathcal{H}$-distance does not necessarily mean inconsistent features across languages and high target risks.

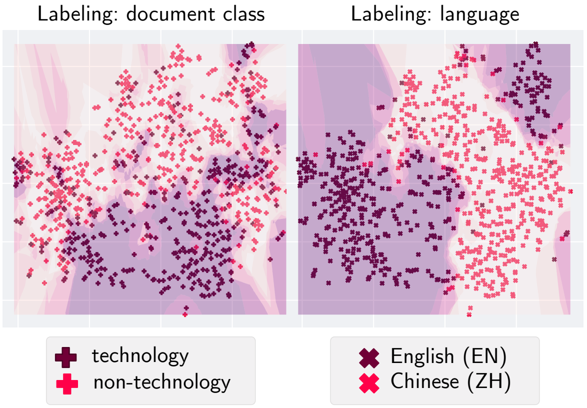


Figure 6: Although the classifier identifies the languages (right), the features are still consistent based on actual document class (left).

5 Conclusions and Future Directions

This study gives new insights into crosslingual transfer learning in multilingual topic models. By formulating the inference process as a circular validation, we derive a PAC-Bayesian theorem to show the factors that affect the success of crosslingual learning. We also connect topic model learning with downstream crosslingual tasks to show how errors propagate.

As the first step toward more theoretically justified crosslingual transfer learning, our study suggests considerations for constructing crosslingual transfer models in general. For example, an effective model should strengthen crosslingual transfer while minimizing non-transferred components, use a balanced dataset or specific optimization algorithms for low-resource languages, and support evaluation metrics that relate to CVL.

References

Sanjeev Arora, Rong Ge, Yonatan Halpern, David M. Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A Practical Algorithm for Topic Modeling with Provable Guarantees. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 280–288.

Maria Barrett, Frank Keller, and Anders Søgaard. 2016. Cross-lingual Transfer of Correlations between Parts of Speech and Gaze Features. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1330–1339.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of Representations for Domain Adaptation. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 137–144.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Lorenzo Bruzzone and Mattia Marconcini. 2010. Domain Adaptation Problems: A DASVM Classification Technique and a Circular Validation Strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 462–471.

Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. 2012. PAC-Bayesian Learning and Domain Adaptation. *CoRR*, abs/1212.2340.

Thomas L Griffiths and Mark Steyvers. 2004. Finding Scientific Topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting Cross-cultural Differences Using a Multilingual Topic Model. *Transactions of the Association for Computational Linguistics*, 4:47–60.

Shudong Hao, Jordan L. Boyd-Graber, and Michael J. Paul. 2018. Lessons from the Bible on Modern Topics: Low-Resource Multilingual Topic Model Evaluation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1090–1100.

Shudong Hao and Michael J. Paul. 2018. Learning Multilingual Topics from Incomparable Corpora. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2595–2609.

Geert Heyman, Ivan Vulic, and Marie-Francine Moens. 2016. C-BiLDA: Extracting Cross-lingual Topics from Non-parallel Texts by Distinguishing Shared from Unshared Content. *Data Mining and Knowledge Discovery*, 30(5):1299–1323.

Yuening Hu, Jordan L. Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive Topic Modeling. *Machine Learning*, 95(3):423–469.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting Multilingual Topics from Unaligned Comparable Corpora. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings*, pages 444–456.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 1459–1474.

Tengfei Ma and Tetsuya Nasukawa. 2017. Inverted Bilingual Topic Models for Lexicon Extraction from Non-parallel Data. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4075–4081.

David A. McAllester. 1999. PAC-Bayesian Model Averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory, COLT 1999, Santa Cruz, CA, USA, July 7-9, 1999*, pages 164–170.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit.

David M. Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual Topic Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 880–889.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining Multilingual Topics from Wikipedia. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 1155–1156.

John Platt. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2018. A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research*, abs/1706.04902.

Ekaterina Shutova, Lin Sun, E. Dario Gutiérrez, Patricia Lichtenstein, and Srini Narayanan. 2017. Multilingual Metaphor Processing: Experiments with Semi-Supervised and Unsupervised Learning. *Computational Linguistics*, 43(1):71–123.

Wim De Smet and Marie-Francine Moens. 2009. Cross-language Linking of News Stories on the Web Using Interlingual Topic Modelling. In *Proceedings of the 2nd ACM Workshop on Social Web Search and Mining, CIKM-SWSM 2009, Hong Kong, China, November 2, 2009*, pages 57–64.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2214–2218.

Leslie G. Valiant. 1984. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142.

Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. 2010. Cross Validation Framework to Choose amongst Models and Datasets for Transfer Learning. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, pages 547–562.

Appendix A Notation

See Table 1.

Notation	Description
S, T	Source and target languages. They are interchangeable during Gibbs sampling. For example, when training English and German, English can be either source or target.
w_ℓ	A **word type** of language ℓ.
x_ℓ	An individual **token** of language ℓ.
z_{x_ℓ}	The topic assignment of token x_ℓ.
$\mathcal{S}_{w_\ell}$	The sample of word type w_ℓ, the set containing all the tokens x_ℓ that are of this word type.
$\mathcal{P}_{x_\ell}$, $\mathcal{P}_{x_\ell,k}$	$\mathcal{P}_{x_\ell}$ denotes the conditional distribution over all topics for token x_ℓ. The conditional probability of sampling a topic k from $\mathcal{P}_{x_\ell}$ is denoted as $\mathcal{P}_{x_\ell,k}$.
$D^{(\ell)}$	The set of documents in language ℓ. This usually refers to the test corpus.
$\widehat{\mathcal{D}}^{(\ell)}$	The array of document representations from the corpus $D^{(\ell)}$ and their document labels.
$\widehat{\phi}_k^{(\ell)}$	The empirical distribution over vocabulary of language ℓ for topic $k = 1, \ldots, K$.
$\widehat{\varphi}^{(w)}$	The word representation, *i.e.,* the empirical distribution over K topics for a word type w. This can be obtained by re-normalizing $\widehat{\phi}_k^{(\ell)}$.
$\widehat{\theta}^{(d)}$	The document representation, *i.e.,* the empirical distribution over K topics for a document d.

Table 1: Notation table.

Appendix B Proofs

Theorem 1. *Let $\widehat{\mathrm{CVL}}^{(t)}(w_T, w_S)$ be the empirical circular validation loss of any bilingual word pair at iteration t of Gibbs sampling. Then $\widehat{\mathrm{CVL}}^{(t)}(w_T, w_S)$ converges as $t \to \infty$.*

Proof. We first notice the triangle inequality:

$$
\begin{aligned}
&\left|\widehat{\mathrm{CVL}}^{(t)}(w_T, w_S) - \widehat{\mathrm{CVL}}^{(t-1)}(w_T, w_S)\right| \\
=& \left| \mathop{\mathbb{E}}_{x_S, x_T}\left[\widehat{\mathcal{L}}^{(t)}(x_T, w_S) + \widehat{\mathcal{L}}^{(t)}(x_S, w_T)\right] \right. \\
&\left. - \mathop{\mathbb{E}}_{x_S, x_T}\left[\widehat{\mathcal{L}}^{(t-1)}(x_T, w_S) + \widehat{\mathcal{L}}^{(t-1)}(x_S, w_T)\right] \right| \\
=& \left| \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}^{(t)}(x_T, w_S)\right] + \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}^{(t)}(x_S, w_T)\right] \right. \\
& - \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}^{(t-1)}(x_T, w_S)\right] \\
&\left. - \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}^{(t-1)}(x_S, w_T)\right] \right| \\
\leq& \left| \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}^{(t)}(x_T, w_S)\right] - \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}^{(t-1)}(x_T, w_S)\right] \right. \\
&\left. + \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}^{(t)}(x_S, w_T)\right] - \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}^{(t-1)}(x_S, w_T)\right] \right| \\
\equiv& \left| \Delta \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}(x_T, w_S)\right] + \Delta \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}(x_S, w_T)\right] \right| \\
\leq& \left| \Delta \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}}\left[\widehat{\mathcal{L}}(x_T, w_S)\right] \right| + \left| \Delta \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}}\left[\widehat{\mathcal{L}}(x_S, w_T)\right] \right|.
\end{aligned}
$$

We look at the first term of the last equation, and the other term can be derived in the same way. We use $\mathcal{P}_{x_T}$ to denote the invariant distribution of the conditional $\mathcal{P}_{x_T}^{(t)}$ as $t \to \infty$. Additionally, let $\mathcal{P}_{x_T, z_{x_S}}$ be the conditional probability for the token x_T being assigned to topic z_{x_S}:

$$\mathcal{P}_{x_T, z_{x_S}} = \Pr\left(k = z_{x_S}; w = w_{x_T}, \mathbf{z}_-, \mathbf{w}_-\right).$$

Another assumption we made is once the source language is converged, we keep the states of it fixed. That is, $z_{x_S}^{(t)} = z_{x_S}^{(t-1)}$, and only sample the target language. Taking the difference between the expectation at iterations t and $t-1$, we have

$$\begin{aligned}
& \lim_{t\to\infty} \left| \Delta \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}} \left[\widehat{\mathcal{L}}(x_T, w_S) \right] \right| \\
= & \lim_{t\to\infty} \left| \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}} \left[\widehat{\mathcal{L}}^{(t)}(x_T, w_S) \right] \right. \\
& \left. - \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}} \left[\widehat{\mathcal{L}}^{(t-1)}(x_T, w_S) \right] \right| \\
= & \lim_{t\to\infty} \left| \mathop{\mathbb{E}}_{x_T} \left[\frac{1}{n_{w_S}} \sum_{x_S} \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S}^{(t)} \right\} \right] \right. \\
& \left. - \mathop{\mathbb{E}}_{x_T} \left[\frac{1}{n_{w_S}} \sum_{x_S} \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t-1)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S}^{(t-1)} \right\} \right] \right| \\
= & \lim_{t\to\infty} \frac{1}{n_{w_S}} \sum_{x_S} \mathop{\mathbb{E}}_{x_T} \left[\left| \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S}^{(t)} \right\} \right. \right. \\
& \left. \left. - \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t-1)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S}^{(t-1)} \right\} \right| \right] \\
= & \lim_{t\to\infty} \frac{1}{n_{w_S}} \sum_{x_S} \mathop{\mathbb{E}}_{x_T} \left[\left| \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S} \right\} \right. \right. \\
& \left. \left. - \mathbb{E}_{h \sim \mathcal{P}_{x_T}^{(t-1)}} \mathbb{1}\left\{ h(x_S) \neq z_{x_S} \right\} \right| \right] \\
= & \lim_{t\to\infty} \frac{1}{n_{w_S}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \left[\left| \left(1 - \mathcal{P}_{x_T, z_{x_S}}^{(t)} \right) \right. \right. \\
& \left. \left. - \left(1 - \mathcal{P}_{x_T, z_{x_S}}^{(t-1)} \right) \right| \right] \\
= & \lim_{t\to\infty} \frac{1}{n_{w_S}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \left[\left| \mathcal{P}_{x_T, z_{x_S}}^{(t-1)} - \mathcal{P}_{x_T, z_{x_S}}^{(t)} \right| \right] \\
= & \lim_{t\to\infty} \frac{1}{n_{w_S}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \left[\left| \mathcal{P}_{x_T, z_{x_S}} - \mathcal{P}_{x_T, z_{x_S}} \right| \right] \\
= & \ 0.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
& \lim_{t\to\infty} \left| \widehat{\mathrm{CVL}}^{(t)}(w_T, w_S) - \widehat{\mathrm{CVL}}^{(t-1)}(w_T, w_S) \right| \\
\leq & \lim_{t\to\infty} \left| \Delta \mathop{\mathbb{E}}_{x_T \in \mathcal{S}_{w_T}} \left[\widehat{\mathcal{L}}(x_T, w_S) \right] \right| \\
& + \left| \Delta \mathop{\mathbb{E}}_{x_S \in \mathcal{S}_{w_S}} \left[\widehat{\mathcal{L}}(x_S, w_T) \right] \right| \\
= & \ 0.
\end{aligned}$$

$\square$

Theorem 3. *Given a bilingual word pair (w_T, w_S), with probability at least $1-\delta$, the following bound holds:*

$$\begin{aligned}
& \mathrm{CVL}(w_T, w_S) \leq \widehat{\mathrm{CVL}}(w_T, w_S) + \\
& \frac{1}{2}\sqrt{\frac{1}{n}\left(\mathrm{KL}_{w_T} + \mathrm{KL}_{w_S} + 2\ln\frac{2}{\delta}\right) + \frac{\ln n^\star}{n}}, \\
& n = \min\left\{n_{w_T}, n_{w_S}\right\}, \quad n^\star = \max\left\{n_{w_T}, n_{w_S}\right\}.
\end{aligned}$$

For brevity we use KL_w to denote $\mathrm{KL}(\mathcal{P}_x || Q_x)$, where $\mathcal{P}_x$ is the conditional distribution from Gibbs sampling of token x with word type w that gives highest loss $\widehat{\mathcal{L}}(x, w)$, and Q_x a prior.

Proof. From Theorem 2, for target language, with probability at least $1-\delta$,

$$\begin{aligned}
& \mathcal{L}(x_T, w_S) \\
\leq & \ \widehat{\mathcal{L}}(x_T, w_S) + \sqrt{\frac{\mathrm{KL}\left(\mathcal{P}_{x_T} || Q_{x_T}\right) + \ln\frac{2\sqrt{n_{w_S}}}{\delta}}{2n_{w_S}}} \\
= & \ \widehat{\mathcal{L}}(x_T, w_S) + \sqrt{\frac{\mathrm{KL}\left(\mathcal{P}_{x_T} || Q_{x_T}\right) + \ln\frac{2}{\delta} + \frac{\ln n_{w_S}}{2n_{w_S}}}{2}} \\
\equiv & \ \widehat{\mathcal{L}}(x_T, w_S) + \epsilon(x_T, w_S).
\end{aligned}$$

For the source language, similarly, with probability at least $1-\delta$,

$$\begin{aligned}
& \mathcal{L}(x_S, w_T) \\
\leq & \ \widehat{\mathcal{L}}(x_S, w_T) + \sqrt{\frac{\mathrm{KL}\left(\mathcal{P}_{x_S} || Q_{x_S}\right) + \ln\frac{2}{\delta} + \frac{\ln n_{w_T}}{2}}{2n_{w_T}}} \\
\equiv & \ \widehat{\mathcal{L}}(x_S, w_T) + \epsilon(x_S, w_T).
\end{aligned}$$

Given a word type w_T, we notice that only the KL-divergence term in $\epsilon(x_T, w_S)$ varies among different tokens x_T. Thus, we use KL_{w_S} and KL_{w_T} to denote the maximal values of KL-divergence over all the tokens,

$$\begin{aligned}
\mathrm{KL}_{w_S} &= \mathrm{KL}\left(\mathcal{P}_{x_T^\star} || Q_{x_T^\star}\right), \\
x_T^\star &= \mathop{\arg\max}_{x_T \in \mathcal{S}_{w_T}} \epsilon(x_T, w_S); \\
\mathrm{KL}_{w_T} &= \mathrm{KL}\left(\mathcal{P}_{x_S^\star} || Q_{x_S^\star}\right), \\
x_S^\star &= \mathop{\arg\max}_{x_S \in \mathcal{S}_{w_S}} \epsilon(x_S, w_T).
\end{aligned}$$

Let $n = \min\{n_{w_T}, n_{w_S}\}$, and $n^\star = \max\{n_{w_T}, n_{w_S}\}$. Due to the fact that $\sqrt{x} + \sqrt{y} \leq \frac{2}{\sqrt{2}}\sqrt{x+y}$ for $x, y > 0$, we have

$$
\begin{aligned}
&\text{CVL}(w_T, w_S)\\
&= \frac{1}{2} \mathop{\mathbb{E}}_{x_S, x_T} \left[\mathcal{L}(x_T, w_S) + \mathcal{L}(x_S, w_T)\right]\\
&= \frac{1}{2} \left(\mathbb{E}_{x_T} \mathcal{L}(x_T, w_S) + \mathbb{E}_{x_S} \mathcal{L}(x_S, w_T)\right)\\
&\leq \frac{1}{2} \left(\mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \widehat{\mathcal{L}}(x_T, w_S) + \mathbb{E}_{x_S \in \mathcal{S}_{w_S}} \widehat{\mathcal{L}}(x_S, w_T)\right)\\
&\quad + \frac{1}{2} \left(\mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \epsilon(x_T, w_S) + \mathbb{E}_{x_S \in \mathcal{S}_{w_S}} \epsilon(x_S, w_T)\right)\\
&= \widehat{\text{CVL}}(w_T, w_S)\\
&\quad + \frac{1}{2} \left(\mathbb{E}_{x_T \in \mathcal{S}_{w_T}} \epsilon(x_T, w_S) + \mathbb{E}_{x_S \in \mathcal{S}_{w_S}} \epsilon(x_S, w_T)\right)\\
&\leq \widehat{\text{CVL}}(w_T, w_S) + \frac{1}{2} \left(\epsilon(x_T^\star, w_S) + \epsilon(x_S^\star, w_T)\right)\\
&\leq \widehat{\text{CVL}}(w_T, w_S)\\
&\quad + \frac{1}{2} \left(\sqrt{\frac{1}{2n_{w_T}} \left(\text{KL}_{w_T} + \ln \frac{2}{\delta} + \frac{1}{2} \ln n_{w_T}\right)} \right.\\
&\quad \left. + \sqrt{\frac{1}{2n_{w_S}} \left(\text{KL}_{w_S} + \ln \frac{2}{\delta} + \frac{1}{2} \ln n_{w_S}\right)} \right)\\
&\leq \widehat{\text{CVL}}(w_T, w_S)\\
&\quad + \frac{1}{2} \sqrt{\frac{\text{KL}_{w_T} + \text{KL}_{w_S} + 2 \ln \frac{2}{\delta}}{n} + \left(\frac{\ln\left(n_{w_T} \cdot n_{w_S}\right)}{2n}\right)}\\
&\leq \widehat{\text{CVL}}(w_T, w_S)\\
&\quad + \frac{1}{2} \sqrt{\frac{\text{KL}_{w_T} + \text{KL}_{w_S} + 2 \ln \frac{2}{\delta}}{n} + \left(\frac{\ln n^\star}{n}\right)},
\end{aligned}
$$

which gives us the result. □

Lemma 1. *Given any bilingual word pair* (w_T, w_S)*, let* $\widehat{\varphi}^{(w)}$ *denote the distribution over topics of word type* w*. Then we have,*

$$
1 - \widehat{\varphi}^{(w_T)\top} \cdot \widehat{\varphi}^{(w_S)} \;\leq\; \widehat{\text{CVL}}(w_T, w_S).
$$

Proof. We expand the equation of $\widehat{\text{CVL}}$ as follows,

$$
\begin{aligned}
&\widehat{\text{CVL}}(w_T, w_S)\\
&= \frac{1}{2} \mathop{\mathbb{E}}_{x_S, x_T} \left[\widehat{\mathcal{L}}(x_T, w_S) + \widehat{\mathcal{L}}(x_S, w_T)\right]\\
&= \frac{1}{2} \left(\mathbb{E}_{x_T} \left[\widehat{\mathcal{L}}(x_T, w_S)\right] + \mathbb{E}_{x_S} \left[\widehat{\mathcal{L}}(x_S, w_T)\right]\right)\\
&= \frac{1}{2} \left(\frac{\sum_{x_T \in \mathcal{S}_{w_T}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathbb{E}_{h \sim \mathcal{P}_{x_T}} \left[\mathbb{1}\left\{h(x_S) \neq z_{x_S}\right\}\right]}{n_{w_T} \cdot n_{w_S}} \right.\\
&\quad \left. + \frac{\sum_{x_S \in \mathcal{S}_{w_S}} \sum_{x_T \in \mathcal{S}_{w_T}} \mathbb{E}_{h \sim \mathcal{P}_{x_S}} \left[\mathbb{1}\left\{h(x_T) \neq z_{x_T}\right\}\right]}{n_{w_S} \cdot n_{w_T}} \right)\\
&= \frac{1}{2} \left(\frac{\sum_{x_T \in \mathcal{S}_{w_T}} \sum_{x_S \in \mathcal{S}_{w_S}} \left(1 - \mathcal{P}_{x_T, z_{x_S}}\right)}{n_{w_T} \cdot n_{w_S}} \right.\\
&\quad \left. + \frac{\sum_{x_S \in \mathcal{S}_{w_S}} \sum_{x_T \in \mathcal{S}_{w_T}} \left(1 - \mathcal{P}_{x_S, z_{x_T}}\right)}{n_{w_S} \cdot n_{w_T}} \right)\\
&= 1 - \frac{1}{2} \left(\frac{\sum_{x_T \in \mathcal{S}_{w_T}} \sum_{x_S \in \mathcal{S}_{w_S}} \mathcal{P}_{x_T, z_{x_S}}}{n_{w_T} \cdot n_{w_S}} \right.\\
&\quad \left. + \frac{\sum_{x_S \in \mathcal{S}_{w_S}} \sum_{x_T \in \mathcal{S}_{w_T}} \mathcal{P}_{x_S, z_{x_T}}}{n_{w_S} \cdot n_{w_T}} \right)\\
&= 1 - \frac{1}{2} \sum_{k=1}^{K} \left(\frac{n_{k|w_S} \cdot \sum_{x_T \in \mathcal{S}_{w_T}} \mathcal{P}_{x_T, k}}{n_{w_T} \cdot n_{w_S}} \right.\\
&\quad \left. + \frac{n_{k|w_T} \cdot \sum_{x_S \in \mathcal{S}_{w_S}} \mathcal{P}_{x_S, z_{x_T}}}{n_{w_S} \cdot n_{w_T}} \right)\\
&= 1 - \frac{1}{2} \sum_{k=1}^{K} \left(\widehat{\varphi}_k^{(w_S)} \cdot \frac{\sum_{x_T \in \mathcal{S}_{w_T}} \mathcal{P}_{x_T, k}}{n_{w_T}} \right.\\
&\quad \left. + \widehat{\varphi}_k^{(w_T)} \cdot \frac{\sum_{x_S \in \mathcal{S}_{w_S}} \mathcal{P}_{x_S, z_{x_T}}}{n_{w_S}} \right)\\
&\geq 1 - \frac{1}{2} \sum_{k=1}^{K} \left(\widehat{\varphi}_k^{(w_S)} \cdot \frac{n_{k|w_T}}{n_{w_T}} + \widehat{\varphi}_k^{(w_T)} \cdot \frac{n_{k|w_S}}{n_{w_S}} \right)\\
&= 1 - \frac{1}{2} \sum_{k=1}^{K} \left(\widehat{\varphi}_k^{(w_S)} \cdot \widehat{\varphi}_k^{(w_T)} + \widehat{\varphi}_k^{(w_T)} \cdot \widehat{\varphi}_k^{(w_S)} \right)\\
&= 1 - \widehat{\varphi}^{(w_S)\top} \cdot \widehat{\varphi}^{(w_T)}
\end{aligned}
$$

which concludes the proof. □

Theorem 5. *Let* $\widehat{\theta}^{(d_S)}$ *be the distribution over topics for document* d_S *(similarly for* d_T*),* $F(d_S, d_T) = \left(\sum_{w_S} {f_{w_S}^{d_S}}^2 \cdot \sum_{w_T} {f_{w_T}^{d_T}}^2\right)^{\frac{1}{2}}$ *where* f_w^d *is the* normalized *frequency of word* w *in document* d*, and* K *the number of topics. Then*

$$
\begin{aligned}
\widehat{\theta}^{(d_S)\top} \cdot \widehat{\theta}^{(d_T)} &\leq F(d_S, d_T)\\
&\quad \cdot \sqrt{K \cdot \sum_{w_S, w_T} \left(\widehat{\text{CVL}}(w_T, w_S) - 1\right)^2}.
\end{aligned}
$$

Proof. We first expand the inner product of $\widehat{\theta}^{(d_S)\top} \cdot \widehat{\theta}^{(d_T)}$ as follows,

$$
\begin{aligned}
&\widehat{\theta}^{(d_S)\top} \cdot \widehat{\theta}^{(d_T)}\\
&= \sum_{k=1}^{K} \widehat{\theta}_k^{(d_S)} \cdot \widehat{\theta}_k^{(d_T)}\\
&= \sum_{k=1}^{K} \left(\left(\sum_{w_S \in V^{(S)}} f_{w_S}^{d_S} \cdot \widehat{\varphi}_k^{(w_S)} \right) \right.\\
&\quad \left. \cdot \left(\sum_{w_T \in V^{(T)}} f_{w_T}^{d_T} \cdot \widehat{\varphi}_k^{(w_T)} \right) \right)\\
&\leq F(d_S, d_T) \cdot \sum_{k=1}^{K} \left(\left(\sum_{w_S \in V^{(S)}} {\widehat{\varphi}_k^{(w_S)}}^2 \right)^{\frac{1}{2}} \cdot \right.\\
&\quad \left. \left(\sum_{w_T \in V^{(T)}} {\widehat{\varphi}_k^{(w_T)}}^2 \right)^{\frac{1}{2}} \right),\\
&F(d_S, d_T)\\
&= \left(\sum_{w_S \in V^{(S)}} {f_{w_S}^{d_S}}^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{w_T \in V^{(T)}} {f_{w_T}^{d_T}}^2 \right)^{\frac{1}{2}},
\end{aligned}
$$

where $F(d_S, d_T)$ is a constant independent of topic k, and the last inequality due to Hölder's. We then focus on the topic-dependent part of the last inequality.

$$\begin{aligned}
&\sum_{k=1}^{K}\left(\left(\sum_{w_S\in V^{(S)}}\widehat{\varphi}_k^{(w_S)^2}\right)^{\frac{1}{2}}\cdot\left(\sum_{w_T\in V^{(T)}}\widehat{\varphi}_k^{(w_T)^2}\right)^{\frac{1}{2}}\right)\\
&=\sum_{k=1}^{K}\left(\sum_{w_S,w_T}\left(\widehat{\varphi}_k^{(w_S)}\cdot\widehat{\varphi}_k^{(w_T)}\right)^2\right)^{\frac{1}{2}}\\
&\leq\sqrt{K}\cdot\left(\sum_{k=1}^{K}\sum_{w_S,w_T}\left(\widehat{\varphi}_k^{(w_S)}\cdot\widehat{\varphi}_k^{(w_T)}\right)^2\right)^{\frac{1}{2}}\\
&=\sqrt{K}\cdot\left(\sum_{w_S,w_T}\sum_{k=1}^{K}\left(\widehat{\varphi}_k^{(w_S)}\cdot\widehat{\varphi}_k^{(w_T)}\right)^2\right)^{\frac{1}{2}}\\
&\leq\sqrt{K}\cdot\left(\sum_{w_S,w_T}\left(\sum_{k=1}^{K}\widehat{\varphi}_k^{(w_S)}\cdot\widehat{\varphi}_k^{(w_T)}\right)^2\right)^{\frac{1}{2}}\\
&=\sqrt{K}\cdot\left(\sum_{w_S,w_T}\left(\widehat{\varphi}^{(w_T)\top}\cdot\widehat{\varphi}^{(w_S)}\right)^2\right)^{\frac{1}{2}}.
\end{aligned}$$

Thus, we have the following inequality:

$$\begin{aligned}
\widehat{\theta}^{(d_S)\top}\cdot\widehat{\theta}^{(d_T)} \;\leq\;& F(d_S,d_T)\cdot\sqrt{K}\\
&\cdot\left(\sum_{w_S,w_T}\left(\widehat{\varphi}^{(w_T)\top}\cdot\widehat{\varphi}^{(w_S)}\right)^2\right)^{\frac{1}{2}}.
\end{aligned}$$

Plug in Lemma 1, we see that

$$\begin{aligned}
\widehat{\theta}^{(d_S)\top}\cdot\widehat{\theta}^{(d_T)} \;\leq\;& F(d_S,d_T)\cdot\sqrt{K}\\
&\cdot\left(\sum_{w_S,w_T}\left(\widehat{\mathrm{CVL}}(w_T,w_S)-1\right)^2\right)^{\frac{1}{2}}.
\end{aligned}$$

$\square$

Appendix C Dataset Details

C.1 Pre-processing

For all the languages, we use existing stemmers to stem words in the corpora and the entries in Wiktionary. Since Chinese does not have stemmers, we loosely use "stem" to refer to "segment" Chinese sentences into words. We also use fixed stopword lists to filter out stop words. Table 2 lists the source of the stemmers and stopwords.

C.2 Training Sets

Our training set is a comparable corpus from Wikipedia. For each Wikipedia article page, there exists an interlingual link to view the article in another language. This interlingual link provides the same article in different languages and is commonly used to create comparable corpora in multilingual studies. We show the statistics of this training corpus in Table 3. The numbers are calculated after stemming and lemmatization.

C.3 Test Sets

C.3.1 Topic Coherence Evaluation Sets

Topic coherence evaluation for multilingual topic models was proposed by Hao et al. (2018), where a comparable corpus is used to calculate bilingual word pair co-occurrence and CNPMI scores. We use a Wikipedia corpus to calculate this score, and the statistics are shown in Table 4. This Wikipedia corpus does not overlap with the training set.

C.3.2 Unseen Document Inference

We use the Global Voices (GV) corpus to create test sets, which can be retrieved from the website `https://globalvoices.org` directly, or from the OPUS collection at `http://opus.nlpl.eu/GlobalVoices.php`. We show the statistics in Table 5. After the column showing number of documents, we also include the statistics of specific labels. The multiclass labels are mutual exclusive, and each document has only one label.

Note that although all the language pairs share the same set of English test documents, the document representations are inferred from different topic models trained specifically for that language pair. Thus, the document representations for the same English document are different across different language pairs.

Lastly, the number of word types is based on the training set and after stemming and lemmatization. When a word type in the test set does not appear in the training set, we ignore this type.

C.3.3 Wiktionary

In downsampling experiments (Section 4.2), we use English Wiktionary to create bilingual dictionaries, which can be downloaded at `https://dumps.wikimedia.org/enwiktionary/`.

---

[1] `http://snowball.tartarus.org`;
[2] `http://arabicstemmer.com`;
[3] `https://github.com/6/stopwords-json`;
[4] `https://github.com/fxsjy/jieba`.

Language	Family	Stemmer	Stopwords
AR	Semitic	`Assem's Arabic Light Stemmer` [1]	GitHub [2]
DE	Germanic	`SnowBallStemmer` [3]	NLTK
EN	Germanic	`SnowBallStemmer`	NLTK
ES	Romance	`SnowBallStemmer`	NLTK
RU	Slavic	`SnowBallStemmer`	NLTK
ZH	Sinitic	`Jieba` [4]	GitHub

Table 2: List of source of stemmers and stopwords used in experiments.

	English		
Language	#docs	#token	#types
AR	3,000	724,362	203,024
DE	3,000	409,381	125,071
ES	3,000	451,115	134,241
RU	3,000	480,715	142,549
ZH	3,000	480,142	141,679
	Paired language		
Language	#docs	#token	#types
AR	3,000	223,937	61,267
DE	3,000	285,745	125,169
ES	3,000	276,188	95,682
RU	3,000	276,462	96,568
ZH	3,000	233,773	66,275

Table 3: Statistics of the Wikipedia training corpus.

Appendix D Topic Model Configurations

For each experiment, we run five chains of Gibbs sampling using the Polylingual Topic Model implemented in MALLET, [5] and take the average over all chains. Each chain has $1{,}000$ iterations, and we do not set a burn-in period. We set the topic number $K = 50$. Other hyperparameters are $\alpha = \frac{50}{K} = 1$ and $\beta = 0.01$ which are the default settings. We do not enable hyperparameter optimization procedures.

	English		
Language	#docs	#token	#types
AR	10,000	3,092,721	143,504
DE	10,000	2,779,963	146,757
ES	10,000	3,021,732	149,423
RU	10,000	3,016,795	154,442
ZH	10,000	1,982,452	112,174
	Paired language		
Language	#docs	#token	#types
AR	10,000	1,477,312	181,734
DE	10,000	1,702,101	227,205
ES	10,000	1,737,312	142,086
RU	10,000	2,299,332	284,447
ZH	10,000	1,335,922	144,936

Table 4: Statistics of the Wikipedia corpus for topic coherence evaluation (CNPMI).

Language	#docs	#token	#types
EN	11,012	3,838,582	104,164
AR	1,086	314,918	53,030
DE	773	334,611	38,702
ES	7,470	3,454,304	110,134
RU	1,035	454,380	67,202
ZH	1,590	804,720	61,319
	#tech.	*#culture*	*#edu.*
EN	4,384	4,679	1,949
AR	457	430	199
DE	315	294	164
ES	2,961	3,121	1,388
RU	362	456	217
ZH	619	622	349

Table 5: Statistics of the Global Voices (GV) corpus.

---

[5] `http://mallet.cs.umass.edu/topics-polylingual.php`.

Recursive Subtree Composition in LSTM-Based Dependency Parsing

Miryam de Lhoneux♠ **Miguel Ballesteros**♢ **Joakim Nivre**♠
♠ Department of Linguistics and Philology, Uppsala University
♢ IBM Research AI, Yorktown Heights, NY
`{miryam.de_lhoneux,joakim.nivre}@lingfil.uu.se`
`miguel.ballesteros@ibm.com`

Abstract

The need for tree structure modelling on top of sequence modelling is an open issue in neural dependency parsing. We investigate the impact of adding a tree layer on top of a sequential model by recursively composing subtree representations (composition) in a transition-based parser that uses features extracted by a BiLSTM. Composition seems superfluous with such a model, suggesting that BiLSTMs capture information about subtrees. We perform model ablations to tease out the conditions under which composition helps. When ablating the backward LSTM, performance drops and composition does not recover much of the gap. When ablating the forward LSTM, performance drops less dramatically and composition recovers a substantial part of the gap, indicating that a forward LSTM and composition capture similar information. We take the backward LSTM to be related to lookahead features and the forward LSTM to the rich history-based features both crucial for transition-based parsers. To capture history-based information, composition is better than a forward LSTM on its own, but it is even better to have a forward LSTM as part of a BiLSTM. We correlate results with language properties, showing that the improved lookahead of a backward LSTM is especially important for head-final languages.

1 Introduction

Recursive neural networks allow us to construct vector representations of trees or subtrees. They have been used for constituency parsing by Socher et al. (2013) and Dyer et al. (2016) and for dependency parsing by Stenetorp (2013) and Dyer et al. (2015), among others. In particular, Dyer et al. (2015) showed that composing representations of subtrees using recursive neural networks can be beneficial for transition-based dependency parsing. These results were further strengthened in Kuncoro et al. (2017) who showed, using ablation experiments, that composition is key in the Recurrent Neural Network Grammar (RNNG) generative parser by Dyer et al. (2016).

In a parallel development, Kiperwasser and Goldberg (2016b) showed that using BiLSTMs for feature extraction can lead to high parsing accuracy even with fairly simple parsing architectures, and using BiLSTMs for feature extraction has therefore become very popular in dependency parsing. It is used in the state-of-the-art parser of Dozat and Manning (2017), was used in 8 of the 10 highest performing systems of the 2017 CoNLL shared task (Zeman et al., 2017) and 10 out of the 10 highest performing systems of the 2018 CoNLL shared task (Zeman et al., 2018).

This raises the question of whether features extracted with BiLSTMs in themselves capture information about subtrees, thus making recursive composition superfluous. Some support for this hypothesis comes from the results of Linzen et al. (2016) which indicate that LSTMs can capture hierarchical information: they can be trained to predict long-distance number agreement in English. Those results were extended to more constructions and three additional languages by Gulordava et al. (2018). However, Kuncoro et al. (2018) have also shown that although sequential LSTMs can learn syntactic information, a recursive neural network which explicitly models hierarchy (the RNNG model from Dyer et al. (2015)) is better at this: it performs better on the number agreement task from Linzen et al. (2016).

To further explore this question in the context of dependency parsing, we investigate the use of recursive composition (henceforth referred to as *composition*) in a parser with an architecture like the one in Kiperwasser and Goldberg (2016b). This allows us to explore variations of features and isolate the conditions under which composi-

Proceedings of NAACL-HLT 2019, pages 1566–1576
Minneapolis, Minnesota, June 2 - June 7, 2019.

tion is helpful. We hypothesise that the use of a BiLSTM for feature extraction makes it possible to capture information about subtrees and therefore makes the use of subtree composition superfluous. We hypothesise that composition becomes useful when part of the BiLSTM is ablated, the forward or the backward LSTM. We further hypothesise that composition is most useful when the parser has no access to information about the function of words in the context of the sentence given by POS tags. When using POS tags, the tagger has indeed had access to the full sentence. We additionally look at what happens when we ablate character vectors which have been shown to capture information which is partially overlapping with information from POS tags. We experiment with a wider variety of languages than Dyer et al. (2015) in order to explore whether the usefulness of different model variants vary depending on language type.

2 K&G Transition-Based Parsing

We define the parsing architecture introduced by Kiperwasser and Goldberg (2016b) at a high level of abstraction and henceforth refer to it as K&G. A K&G parser is a greedy transition-based parser.[1] For an input sentence of length n with words $w_1, \ldots, w_n$, a sequence of vectors $x_{1:n}$ is created, where the vector x_i is a vector representation of the word w_i. We refer to these as *type* vectors, as they are the same for all occurrences of a word type. Type vectors are then passed through a feature function which learns representations of words in the context of the sentence.

$$x_i = e(w_i)$$
$$v_i = f(x_{1:n}, i)$$

We refer to the vector v_i as a *token* vector, as it is different for different tokens of the same word type. In Kiperwasser and Goldberg (2016b), the feature function used is a BiLSTM.

As is usual in transition-based parsing, parsing involves taking transitions from an initial configuration to a terminal one. Parser configurations are represented by a stack, a buffer and set of dependency arcs (Nivre, 2008). For each configuration c, the feature extractor concatenates the token representations of core elements from the stack and buffer. These token vectors are passed to a classifier, typically a Multilayer Perceptron (MLP). The MLP scores transitions together with the arc labels for transitions that involve adding an arc. Both the word type vectors and the BiLSTMs are trained together with the model.

3 Composing Subtree Representations

Dyer et al. (2015) looked at the impact of using a recursive composition function in their parser, which is also a transition-based parser but with an architecture different from K&G. They make use of a variant of the LSTM called a stack LSTM. A stack LSTM has push and pop operations which allow passing through states in a tree structure rather than sequentially. Stack LSTMs are used to represent the stack, the buffer, and the sequence of past parsing actions performed for a configuration.

The words of the sentence are represented by vectors of the word types, together with a vector representing the word's POS tag. In the initial configuration, the vectors of all words are in the buffer and the stack is empty. The representation of the buffer is the end state of a backward LSTM over the word vectors. As parsing evolves, the word vectors are popped from the buffer, pushed to and popped from the stack and the representations of stack and buffer get updated.

Dyer et al. (2015) define a recursive composition function and compose tree representations incrementally, as dependents get attached to their head. The composed representation c is built by concatenating the vector h of the head with the vector of the dependent d, as well as a vector r representing the label paired with the direction of the arc. That concatenated vector is passed through an affine transformation and then through a $tanh$ non-linear activation.

$$c = tanh(W[h; d; r] + b)$$

They create two versions of the parser. In the first version, when a dependent is attached to a head, the word vector of the head is replaced by a composed vector of the head and dependent. In the second version, they simply keep the vector of the head when attaching a dependent to a head. They observe that the version with composition is substantially better than the version without, by 1.3 LAS points for English (on the Penn Treebank (PTB) test set) and 2.1 for Chinese (on the Chinese Treebank (CTB) test set).

---

[1] Kiperwasser and Goldberg (2016b) also define a graph-based parser with similar feature extraction, but we focus on transition-based parsing.

Their parser uses POS tag information. POS tags help to disambiguate between different functional uses of a word and in this way give information about the use of the word in context. We hypothesise that the effect of using a recursive composition function is stronger when not making use of POS tags.

4 Composition in a K&G Parser

The parsing architectures of the stack LSTM parser (S-LSTM) and K&G are different but have some similarities.[2] In both cases, the configuration is represented by vectors obtained by LSTMs. In K&G, it is represented by the token vectors of top items of the stack and the first item of the buffer. In the S-LSTM, it is represented by the vector representations of the entire stack, buffer and sequence of past transitions.

Both types of parsers learn vector representations of word types which are passed to an LSTM. In K&G, they are passed to an LSTM in a feature extraction step that happens before parsing. The LSTM in this case is used to learn vectors that have information about the context of each word, a token vector. In the S-LSTM, word type vectors are passed to Stack LSTMs as parsing evolves. In this case, LSTMs are used to learn vector representations of the stack and buffer (as well as one which learns a representation of the parsing action history).

When composition is not used in the S-LSTM, word vectors represent word types. When composition is used, as parsing evolves, the stack and buffer vectors get updated with information about the subtrees they contain, so that they gradually become contextualised. In this sense, those vectors become more like token vectors in K&G. More specifically, as explained in the previous section, when a dependent is attached to its head, the composition function is applied to the vectors of head and dependent and the vector of the head is replaced by this composed vector.

We cannot apply composition on type vectors in the K&G architecture, since they are not used after the feature extraction step and hence cannot influence the representation of the configuration. Instead, we apply composition on the token vectors. We embed those composed representations in the same space as the token vectors.

In K&G, like in the S-LSTM, we can create a composition function and compose the representation of subtrees as parsing evolves. We create two versions of the parser, one where word tokens are represented by their token vector. The other where they are represented by their token vector and the vector of their subtree c_i, which is initially just a copy of the token vector ($v_i = f(x_{1:n}, i) \circ c_i$). When a dependent word d is attached to a word h with a relation and direction r, c_i is computed with the same composition function as in the S-LSTM defined in the previous section, repeated below.[3]

This composition function is a simple recurrent cell. Simple RNNs have known shortcomings which have been addressed by using LSTMs, as proposed by Hochreiter and Schmidhuber (1997). A natural extension to this composition function is therefore to replace it with an LSTM cell. We also try this variant. We construct LSTMs for subtrees. We initialise a new LSTM for each new subtree that is formed, that is, when a dependent d is attached to a head h which does not have any dependent yet. Each time we attach a dependent to a head, we construct a vector which is a concatenation of h, d and r. We pass this vector to the LSTM of h. c is the output state of the LSTM after passing through that vector. We denote those models with $+rc$ for the one using an ungated recurrent cell and with $+lc$ for the one using an LSTM cell.

$$c = tanh(W[h; d; r] + b)$$
$$c = \textsc{Lstm}([h; d; r])$$

As results show (see § 5), neither type of composition seems useful when used with the K&G parsing model, which indicates that BiLSTMs capture information about subtrees. To further investigate this and in order to isolate the conditions under which composition is helpful, we perform different model ablations and test the impact of recursive composition on these ablated models.

First, we ablate parts of the BiLSTMs: we ablate either the forward or the backward LSTM. We therefore build parsers with 3 different feature functions $f(x, i)$ over the word type vectors x_i in the sentence x: a BiLSTM (bi) (our baseline), a backward LSTM (bw) (i.e., ablating the forward LSTM) and a forward LSTM (fw) (i.e., ablating

---

[2] Note that we use S-LSTM to denote the stack LSTM parser, not the stack LSTM as an LSTM type.

[3] Note that, in preliminary experiments, we tried replacing the vector of the head by the vector of its subtree instead of concatenating the two but concatenating gave much better results.

the backward LSTM):

$$\begin{aligned} bi(x,i) &= \text{BiLSTM}(x_{1:n}, i) \\ bw(x,i) &= \text{LSTM}(x_{n:1}, i) \\ fw(x,i) &= \text{LSTM}(x_{1:n}, i) \end{aligned}$$

K&G parsers with unidirectional LSTMs are, in some sense, more similar to the S-LSTM than those with a BiLSTM, since the S-LSTM only uses unidirectional LSTMs. We hypothesise that composition will help the parser using unidirectional LSTMs in the same way it helps an S-LSTM.

We additionally experiment with the vector representing the word at the input of the LSTM. The most complex representation consists of a concatenation of an embedding of the word type $e(w_i)$, an embedding of the (predicted) POS tag of w_i, $p(w_i)$ and a character representation of the word obtained by running a BiLSTM over the characters $ch_{1:m}$ of w_i (BiLSTM($ch_{1:m}$)).

$$x_i = e(w_i) \circ p(w_i) \circ \text{BiLSTM}(ch_{1:m})$$

Without a POS tag embedding, the word vector is a representation of the word type. With POS information, we have some information about the word in the context of the sentence and the tagger has had access to the full sentence. The representation of the word at the input of the BiLSTM is therefore more contextualised and it can be expected that a recursive composition function will be less helpful than when POS information is not used. Character information has been shown to be useful for dependency parsing first by Ballesteros et al. (2015). Ballesteros et al. (2015) and Smith et al. (2018b) among others have shown that POS and character information are somewhat complementary. Ballesteros et al. (2015) used similar character vectors in the S-LSTM parser but did not look at the impact of composition when using these vectors. Here, we experiment with ablating either or both of the character and POS vectors. We look at the impact of using composition on the full model as well as these ablated models. We hypothesise that composition is most helpful when those vectors are not used, since they give information about the functional use of the word in context.

Parser We use UUParser, a variant of the K&G transition-based parser that employs the arc-hybrid transition system from Kuhlmann et al. (2011) extended with a SWAP transition and a Static-Dynamic oracle, as described in de Lhoneux et al. (2017b)[4]. The SWAP transition is used to allow the construction of non-projective dependency trees (Nivre, 2009). We use default hyperparameters. When using POS tags, we use the universal POS tags from the UD treebanks which are coarse-grained and consistent across languages. Those POS tags are predicted by UDPipe (Straka et al., 2016) both for training and parsing. This parser obtained the 7th best LAS score on average in the 2018 CoNLL shared task (Zeman et al., 2018), about 2.5 LAS points below the best system, which uses an ensemble system as well as ELMo embeddings, as introduced by Peters et al. (2018). Note, however, that we use a slightly impoverished version of the model used for the shared task which is described in Smith et al. (2018a): we use a less accurate POS tagger (UDPipe) and we do not make use of multi-treebank models. In addition, Smith et al. (2018a) use the three top items of the stack as well as the first item of the buffer to represent the configuration, while we only use the two top items of the stack and the first item of the buffer. Smith et al. (2018a) also use an *extended feature set* as introduced by Kiperwasser and Goldberg (2016b) where they also use the rightmost and leftmost children of the items of the stack and buffer that they consider. We do not use that extended feature set. This is to keep the parser settings as simple as possible and avoid adding confounding factors. It is still a near-SOTA model. We evaluate parsing models on the development sets and report the average of the 5 best results in 30 epochs and 5 runs with different random seeds.

Data We test our models on a sample of treebanks from Universal Dependencies v2.1 (Nivre et al., 2017). We follow the criteria from de Lhoneux et al. (2017c) to select our sample: we ensure typological variety, we ensure variety of domains, we verify the quality of the treebanks, and we use one treebank with a large amount of non-projective arcs. However, unlike them, we do not use extremely small treebanks. Our selection is the same as theirs but we remove the tiny treebanks and replace them with 3 others. Our final set is: Ancient Greek (PROIEL), Basque, Chinese, Czech, English, Finnish, French, Hebrew and Japanese.

---

[4] The code can be found at `https://github.com/mdelhoneux/uuparser-composition`

5 Results

First, we look at the effect of our different recursive composition functions on the full model (i.e., the model using a BiLSTM for feature extraction as well as both character and POS tag information). As can be seen from Figure 1, recursive composition using an LSTM cell ($+lc$) is generally better than recursive composition with a recurrent cell ($+rc$), but neither technique reliably improves the accuracy of a BiLSTM parser.

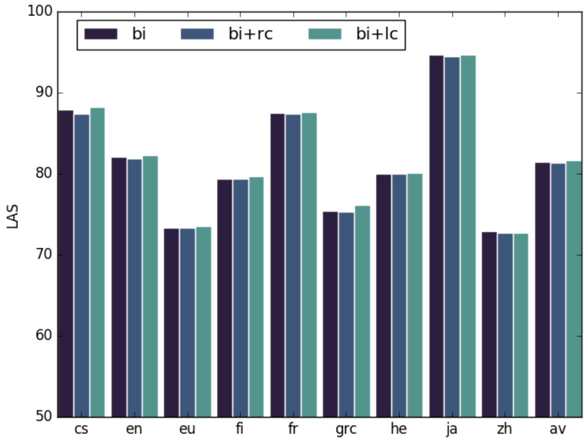


Figure 1: LAS of models using a BiLSTM (bi) without composition, with a recurrent cell ($+rc$) and with an LSTM cell ($+lc$). Bar charts truncated at 50 for visualization purposes.

5.1 Ablating the forward and backward LSTMs

Second, we only consider the models using character and POS information and look at the effect of ablating parts of the BiLSTM on the different languages. The results can be seen in Figure 2. As expected, the BiLSTM parser performs considerably better than both unidirectional LSTM parsers, and the backward LSTM is considerably better than the forward LSTM, on average. It is, however, interesting to note that using a forward LSTM is much more hurtful for some languages than others: it is especially hurtful for Chinese and Japanese. This can be explained by language properties: the right-headed languages suffer more from ablating the backward LSTM than other languages. We observe a correlation between how hurtful a forward model is compared to the baseline and the percentage of right-headed content dependency relations ($R = -0.838$, $p < .01$), see Figure 3.[5]

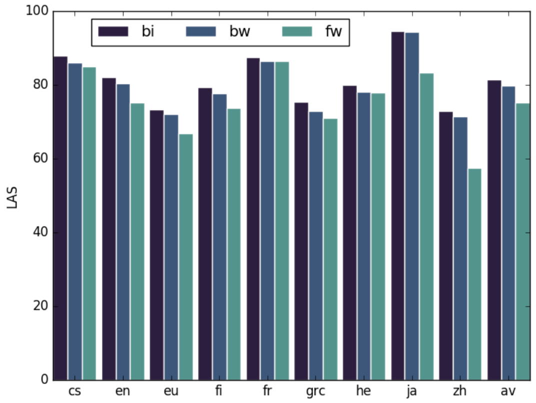


Figure 2: LAS of models using a BiLSTM (bi), backward LSTM (bw) and forward LSTM (fw).

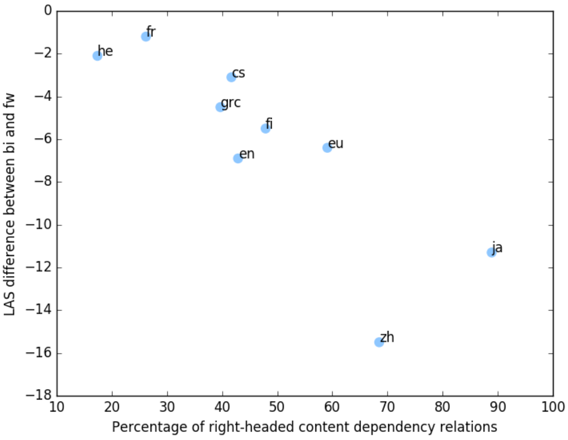


Figure 3: Correlation between how hurtful it is to ablate the backward LSTM and right-headedness of languages.

There is no significant correlation between how hurtful ablating the forward LSTM is and the percentage of left-headed content dependency relations ($p > .05$) indicating that its usefulness is not dependent on language properties. We hypothesise that dependency length or sentence length can play a role but we also find no correlation between how hurtful it is to ablate the forward LSTM and average dependency or sentence length in treebanks. It is finally also interesting to note that the backward LSTM performance is close to the BiLSTMs performance for some languages (Japanese and French).

---

[5] The reason we only consider content dependency relations is that the UD scheme focuses on dependency relations between content words and treats function words as features of content words to maximise parallelism across languages (de Marneffe et al., 2014).

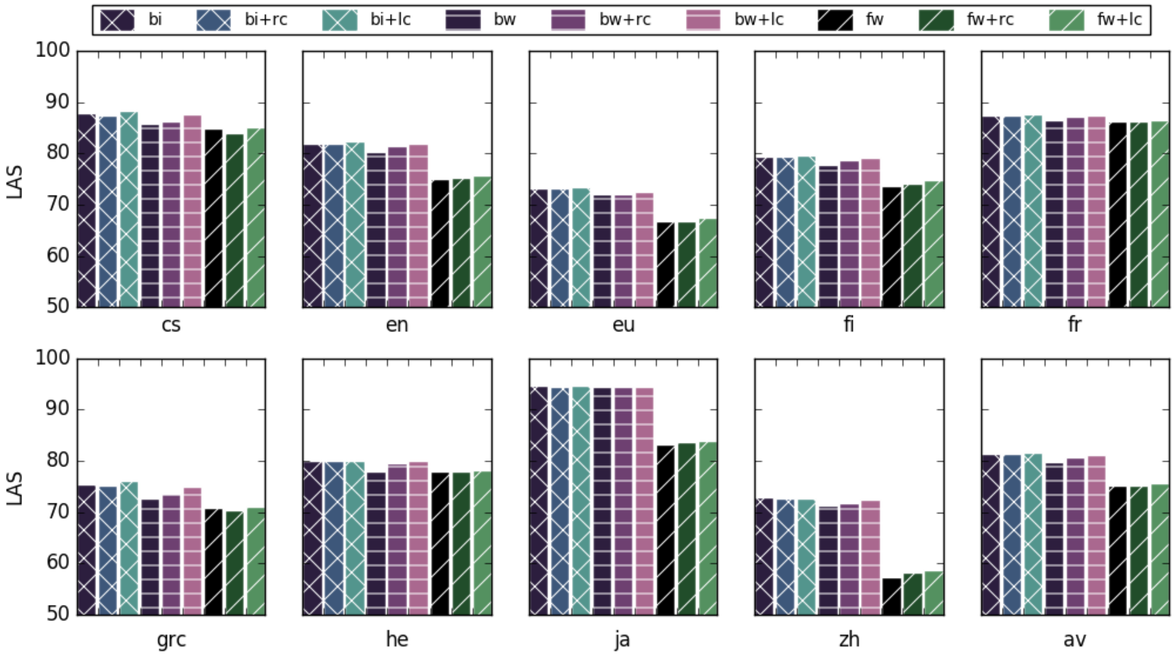


Figure 4: LAS of models using a BiLSTM (bi), backward LSTM (bw) and forward LSTM (fw), without recursive composition, with a recurrent cell ($+rc$) and with a LSTM cell ($+lc$). Bar charts truncated at 50 for visualization purposes.

We now look at the effect of using recursive composition on these ablated models. Results are given in Figure 4. First of all, we observe unsurprisingly that composition using an LSTM cell is much better than using a simple recurrent cell. Second, both types of composition help the backward LSTM case, but neither reliably helps the bi models. Finally, the recurrent cell does not help the forward LSTM case but the LSTM cell does to some extent. It is interesting to note that using composition, especially using an LSTM cell, bridges a substantial part of the gap between the bw and the bi models.

These results can be related to the literature on transition-based dependency parsing. Transition-based parsers generally rely on two types of features: *history-based features* over the emerging dependency tree and *lookahead features* over the buffer of remaining input. The former are based on a hierarchical structure, the latter are purely sequential. McDonald and Nivre (2007) and McDonald and Nivre (2011) have shown that history-based features enhance transition-based parsers as long as they do not suffer from error propagation. However, Nivre (2006) has also shown that lookahead features are absolutely crucial given the greedy left-to-right parsing strategy.

In the model architectures considered here, the backward LSTM provides an improved lookahead. Similarly to the lookahead in statistical parsing, it is sequential. The difference is that it gives information about upcoming words with unbounded length. The forward LSTM in this model architecture provides history-based information but unlike in statistical parsing, that information is built sequentially rather than hierarchically: the forward LSTM passes through the sentence in the linear order of the sentence. In our results, we see that lookahead features are more important than the history-based ones. It hurts parsing accuracy more to ablate the backward LSTM than to ablate the forward one. This is expected given that some history-based information is still available through the top tokens on the stack, while the lookahead information is almost lost completely without the backward LSTM.

A composition function gives hierarchical information about the history of parsing actions. It makes sense that it helps the backward LSTM model most since that model has no access to any information about parsing history. It helps the forward LSTM slightly which indicates that there can be gains from using structured information about parsing history rather than sequential information. We could then expect that composition should help the BiLSTM model which, how-

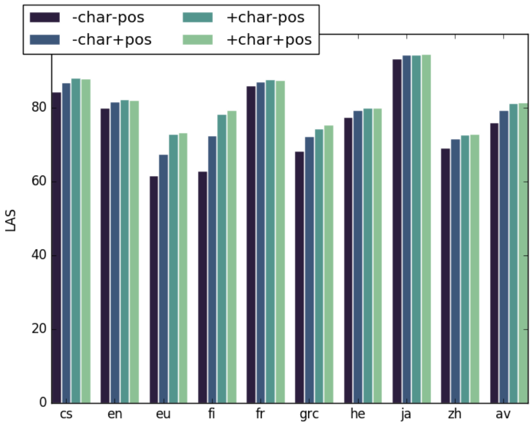


Figure 5: LAS of baseline, using char and/or POS tags to construct word representations

ever, is not the case. This might be because the BiLSTM constructs information about parsing history and lookahead into a unique representation. In any case, this indicates that BiLSTMs are powerful feature extractors which seem to capture useful information about subtrees.

5.2 Ablating POS and character information

Next, we look at the effect of the different word representation methods on the different languages, as represented in Figure 5. As is consistent with the literature (Ballesteros et al., 2015; de Lhoneux et al., 2017a; Smith et al., 2018b), using character-based word representations and/or POS tags consistently improves parsing accuracy but has a different impact in different languages and the benefits of both methods are not cumulative: using the two combined is not much better than using either on its own. In particular, character models are an efficient way to obtain large improvements in morphologically rich languages.

We look at the impact of recursive compositions on all combinations of ablated models, see Table 1. We only look at the impact of using an LSTM cell rather than a recurrent cell since it was a better technique across the board (see previous section).

Looking first at BiLSTMs, it seems that composition does not reliably help parsing accuracy, regardless of access to POS and character information. This indicates that the vectors obtained from the BiLSTM already contain information that would otherwise be obtained by using composition.

Turning to results with either the forward or the backward LSTM ablated, we see the expected pattern. Composition helps more when the model lacks POS tags, indicating that there is some redundancy between these two methods of building contextual information. Composition helps recover a substantial part of the gap of the model with a backward LSTM with or without POS tag. It recovers a much less substantial part of the gap in other cases which means that, although there is some redundancy between these different methods of building contextual information, they are still complementary and a recursive composition function cannot fully compensate for the lack of a backward LSTM or POS and/or character information. There are some language idiosyncracies in the results. While composition helps recover most of the gap for the backward LSTM models without POS and/or character information for Czech and English, it does it to a much smaller extent for Basque and Finnish. We hypothesise that arc depth might impact the usefulness of composition, since more depth means more matrix multiplications with the composition function. However, we find no correlation between average arc depth of the treebanks and usefulness of composition. It is an open question why composition helps some languages more than others.

Note that we are not the first to use composition over vectors obtained from a BiLSTM in the context of dependency parsing, as this was done by Qi and Manning (2017). The difference is that they compose vectors before scoring transitions. It was also done by Kiperwasser and Goldberg (2016a) who showed that using BiLSTM vectors for words in their Tree LSTM parser is helpful but they did not compare this to using BiLSTM vectors without the Tree LSTM.

Recurrent and recursive LSTMs in the way they have been considered in this paper are two ways of constructing contextual information and making it available for local decisions in a greedy parser. The strength of recursive LSTMs is that they can build this contextual information using hierarchical context rather than linear context. A possible weakness is that this makes the model sensitive to error propagation: a wrong attachment leads to using the wrong contextual information. It is therefore possible that the benefits and drawbacks of using this method cancel each other out in the context of BiLSTMs.

	pos+char+						pos+char-					
	bi	bi+lc	bw	bw+lc	fw	fw+lc	bi	bi+lc	bw	bw+lc	fw	fw+lc
cs	87.9	88.2	85.9	**87.7**	84.9	85.0	86.7	87.0	84.5	**86.2**	83.6	83.6
en	82.0	82.3	80.3	**81.9**	75.1	**75.6**	81.5	81.5	79.7	**81.4**	74.3	**75.0**
eu	73.3	73.5	72.0	72.4	66.8	**67.4**	67.4	67.6	65.6	**66.3**	59.6	**60.5**
fi	79.3	79.7	77.7	**79.2**	73.7	**74.7**	72.5	72.7	69.8	**71.7**	66.7	**67.4**
fr	87.5	87.6	86.4	**87.5**	86.3	86.4	87.1	87.2	85.8	**86.9**	85.7	85.9
grc	75.4	**76.1**	72.8	**75.0**	70.9	71.1	72.2	72.5	69.6	**71.4**	67.4	67.8
he	80.0	80.1	78.0	**80.0**	77.9	78.2	79.4	79.2	77.2	**79.0**	76.9	77.3
ja	94.6	94.6	94.4	94.5	83.3	**83.9**	94.3	94.3	94.2	94.3	83.0	**83.6**
zh	72.9	72.7	71.3	**72.4**	57.4	**58.7**	71.5	71.3	69.9	**70.8**	56.4	**57.9**
av	81.4	81.6	79.8	**81.2**	75.1	**75.7**	79.2	79.2	77.4	**78.7**	72.6	**73.2**

	pos-char+						pos-char-					
	bi	bi+lc	bw	bw+lc	fw	fw+lc	bi	bi+lc	bw	bw+lc	fw	fw+lc
cs	88.1	88.4	86.0	**87.8**	84.7	84.9	84.3	84.5	81.3	**83.1**	79.9	79.8
en	82.2	82.1	79.8	**81.6**	73.2	**73.8**	80.0	79.9	77.5	**79.2**	70.5	**71.5**
eu	72.8	72.9	71.5	71.8	65.4	**66.4**	61.6	62.0	57.7	**59.5**	48.7	**51.2**
fi	78.2	78.6	75.8	**77.9**	72.0	**73.0**	62.8	63.1	56.6	**60.2**	52.8	**54.7**
fr	87.6	87.7	86.1	**87.4**	85.4	85.7	85.9	85.8	83.7	**85.3**	83.1	83.3
grc	74.4	74.8	71.3	**73.7**	69.2	69.6	68.3	**69.0**	64.6	**67.3**	62.6	**63.4**
he	79.9	80.1	77.4	**79.9**	76.5	**77.3**	77.5	77.4	74.4	**77.2**	74.2	74.7
ja	94.2	94.4	94.2	94.4	81.3	**81.8**	93.2	93.3	92.7	93.1	79.5	**80.2**
zh	72.7	72.5	70.8	**72.2**	56.5	**58.2**	69.1	69.3	66.7	**68.1**	53.4	**55.0**
av	81.1	81.3	79.2	**80.8**	73.8	**74.5**	75.9	76.0	72.8	**74.8**	67.2	**68.2**

Table 1: LAS for bi, bw and fw, without and with composition ($+lc$) with an LSTM. Difference > 0.5 with $+lc$ in bold.

5.3 Ensemble

To investigate further the information captured by BiLSTMs, we ensemble the 6 versions of the models with POS and character information with the different feature extractors (bi, bw, fw) with ($+lc$) and without composition. We use the (unweighted) reparsing technique of Sagae and Lavie (2006)[6] and ignoring labels. As can be seen from the UAS scores in Table 2, the ensemble (*full*) largely outperforms the parser using only a BiLSTM, indicating that the information obtained from the different models is complementary. To investigate the contribution of each of the 6 models, we ablate each one by one. As can be seen from Table 2, ablating either of the BiLSTM models or the backward LSTM using composition, results in the least effective of the ablated models, further strengthening the conclusion that BiLSTMs are powerful feature extractors.

	bi	full	-bi	-[bi+lc]	-bw	-[bw+lc]	-fw	-[fw+lc]
cs	90.9	92.0	91.8	91.8	91.8	91.8	92.1	92.0
en	85.8	87.1	86.7	86.7	86.8	86.7	87.2	87.2
eu	78.7	80.9	80.3	80.2	80.4	80.3	80.9	81.0
fi	83.5	85.5	85.4	85.4	85.3	85.2	85.6	85.5
fr	89.8	90.8	90.8	90.6	90.8	90.7	90.8	90.8
grc	81.2	83.5	83.0	83.1	83.3	83.0	83.4	83.6
he	86.2	87.6	87.6	87.4	87.5	87.2	87.6	87.7
ja	95.9	96.1	95.8	95.7	95.9	95.8	96.3	96.2
zh	78.3	79.3	78.4	78.6	78.4	78.7	79.8	79.9
av	85.6	87.0	86.6	86.6	86.7	86.6	87.1	87.1

Table 2: UAS ensemble (full) and ablated experiments.

6 Conclusion

We investigated the impact of composing the representation of subtrees in a transition-based parser. We observed that composition does not reliably help a parser that uses a BiLSTM for feature extraction, indicating that vectors obtained from the BiLSTM might capture subtree information, which is consistent with the results of Linzen et al. (2016). However, we observe that, when ablating the backward LSTM, performance drops and recursive composition does not help to recover much of this gap. We hypothesise that this is because the backward LSTM primarily improves the lookahead for the greedy parser. When ablating the forward LSTM, performance drops to a smaller extent and recursive composition recovers a substantial part of the gap. This indicates that a forward LSTM and a recursive composition function capture similar information, which we take to be related to the rich history-based features crucial for a transition-based parser. To capture this infor-

---

[6] This method scores all arcs by the number of parsers predicting them and extracts a maximum spanning tree using the Chu-Liu-Edmonds algorithm (Edmonds, 1967).

mation, a recursive composition function is better than a forward LSTM on its own, but it is even better to have a forward LSTM as part of a BiLSTM. We further find that recursive composition helps more when POS tags are ablated from the model, indicating that POS tags and a recursive composition function are partly redundant ways of constructing contextual information. Finally, we correlate results with language properties, showing that the improved lookahead of a backward LSTM is especially important for head-final languages.

Acknowledgments

We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). We thank Sara Stymne and Aaron Smith for many discussions about this paper.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 349–359.

Timothy Dozat and Christopher Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations.*

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138.*

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree LSTMs. *Transactions of the Association for Computational Linguistics*, 4:445–461.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 673–682.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From raw text to universal dependencies - look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217, Vancouver, Canada. Association for Computational Linguistics.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104, Pisa, Italy. Association for Computational Linguistics.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017c. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *Proceedings of the 15th Treebanks and Linguistic Theories Workshop (TLT).*

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: A cross-linguistic typology. In *Proceedings of the 9th International*

Conference on Language Resources and Evaluation (LREC), pages 4585–4592.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, pages 197–230.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual*

Parsing from Raw Text to Universal Dependencies, pages 113–123. Association for Computational Linguistics.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, et al. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465.

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *Deep Learning Workshop at the 2013 Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, USA.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. European Language Resources Association.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Cross-lingual CCG Induction

Kilian Evang
University of Düsseldorf
Germany
`evang@hhu.de`

Abstract

Combinatory categorial grammars are linguistically motivated and useful for semantic parsing, but costly to acquire in a supervised way and difficult to acquire in an unsupervised way. We propose an alternative making use of cross-lingual learning: an existing source-language parser is used together with a parallel corpus to induce a grammar and parsing model for a target language. On the PASCAL benchmark, cross-lingual CCG induction outperforms CCG induction from gold-standard POS tags on 3 out of 8 languages, and unsupervised CCG induction on 6 out of 8 languages. We also show that cross-lingually induced CCGs reflect known syntactic properties of the target languages.

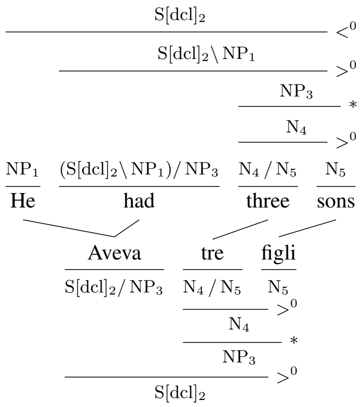


Figure 1: Projection of an English CCG derivation to an Italian translation. The indices distinguish different instances of categories.

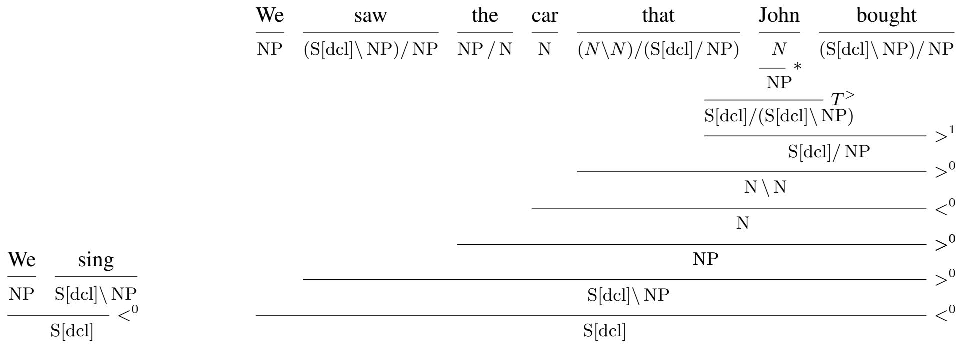
1 Introduction

Combinatory Categorial Grammar (CCG) (Steedman, 2001) is a grammar formalism known for its linguistic elegance and computational efficiency. It has been successfully used for statistical syntactic parsing (Clark and Curran, 2004; Lewis et al., 2016) and has emerged as a leading grammar formalism in semantic parsing (Curran et al., 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011, 2013; Reddy et al., 2014; Artzi et al., 2015; Beschke and Menzel, 2018). Semantic parsing is important because it translates natural language utterances to something that a computer can understand, e.g., database queries, computer commands, or logical formulas, enabling next-generation information systems and knowledge extraction from text, among other applications.

CCGs used in most work to date are either hand-crafted (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Artzi et al., 2015) or extracted from large syntactically annotated corpora (Curran et al., 2007; Reddy et al., 2014). In either case language-specific human effort is required. Acquiring CCGs in an unsupervised way is difficult and does not reach the performance of supervised methods (Bisk and Hockenmaier, 2013). As a result, most research focuses on English and other languages are neglected, meaning that speakers of other languages have delayed or no access to CCG-based semantic parsing technology.

We propose to overcome this bottleneck by inducing CCGs cross-lingually, i.e., transferring an existing grammar from English to other languages via unannotated parallel data. The process is illustrated for one English-Italian sentence pair in Figure 1: the English sentence is parsed by an existing CCG parser and word-aligned to the Italian sentence. Italian words receive categories equivalent to those of the aligned English words, and a semantically equivalent derivation is built for the Italian sentence. With enough derivations projected in this way, they can be used to extract a CCG lexicon and to estimate parameter weights for parsing the target language.

Proceedings of NAACL-HLT 2019, pages 1577–1587
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) (b)

Figure 2: Two examples of CCG derivations.

Unlike previous competitive methods for CCG induction such as Bisk and Hockenmaier (2013), our method does not require the training data to be POS-tagged. It also induces more fine-grained labels. In this paper, we compare the performance of parsers trained using our method to previous induced CCG parsers. We also investigate whether the cross-lingually induced CCG lexicons correspond with linguistic insights about the target languages.

2 Combinatory Categorial Grammar

In categorial grammars (Bar-Hillel, 1953), words and larger constituents share a single space of labels, called *categories*. For example, the intransitive verb *sing* in Figure 2(a) and the verb phrase *saw the car that John bought* in Figure 2(b) have the same category: $\mathrm{S[dcl]\backslash NP}$. Parse trees are conventionally called *derivations* and their nodes depicted as horizontal lines, placed underneath their children.

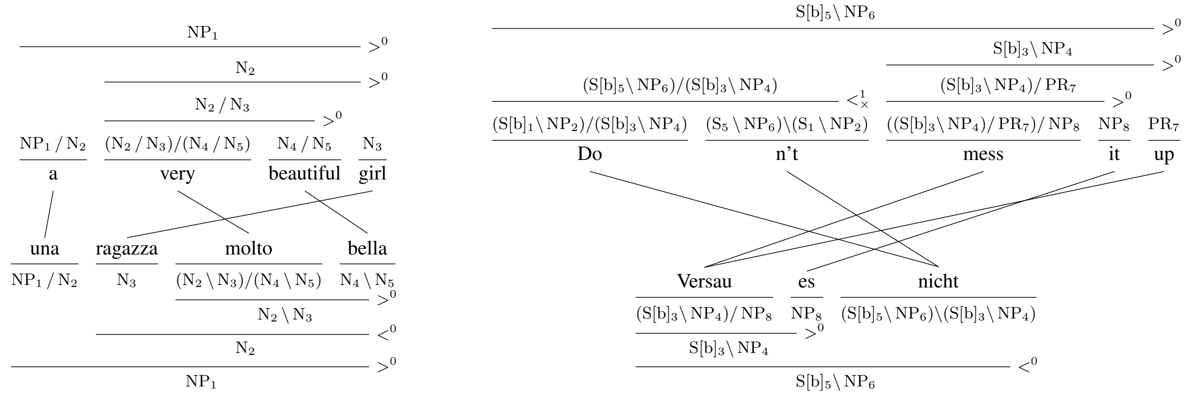
Categorial grammars have only few *basic categories*, typically: N for nouns, NP for noun phrases, PP for argument prepositional phrases, PR for verb particles, and $\mathrm{S}[X]$ for sentences, where X is a feature that indicates the type of sentence or clause, e.g., dcl for declarative sentences or b for infinitives. All other categories are *functional categories*, which contain information about what kinds of arguments constituents with these categories combine with, and what kinds of constituents result. For example, in English, a declarative verb phrase is a constituent that combines with a noun phrase (the subject) to its left to form a declarative sentence. This is expressed by its category: $\mathrm{S[dcl]\backslash NP}$. Similarly, a transitive verb is a constituent that combines with a noun phrase (the object) to its right to form a verb phrase. This results in the functional category $\mathrm{(S[dcl]\backslash NP)/NP}$ for a transitive verb, where the brackets determine the order in which it combines with its arguments.

With such expressive categories, categorial grammars are mainly defined via the lexicon, i.e., which words are associated with which categories. Only few and very general rules are needed to specify how constituents may combine. The basic rules are forward application and backward application ($>^0$, $<^0$). They allow a constituent with a functional category to combine with its argument. Combinatory categorial grammar adds type raising ($T^>$, $T^<$) and generalizes application to (harmonic and crossing) composition ($>^1$, $<^1$, $>^2$, $<^2$...). This allows for dealing with “incomplete” constituents such as the object relative clause *John bought* in Figure 2(b). The object is extracted, thus the transitive verb *bought* cannot combine with the NP it expects to its right. Thanks to type raising and composition, it can nevertheless combine with its subject, resulting in a sentence with an open object argument slot ($\mathrm{S[dcl]/NP}$), which is taken as an argument by the relative pronoun *that*.

Additionally, some unary *type changing* ($*$) rules are used to convert categories, e.g., $\mathrm{N} \Rightarrow \mathrm{NP}$ to convert N to NP when there is no determiner.

3 Derivation Projection

Examples of derivations projected from English to other languages are shown in Figures 1 and 3. Note that we give basic categories indices here to distinguish different instantiations of the same category. For the purposes of derivation projection, different instantiations are treated as different cat-



(a) (b)

Figure 3: Projections of English CCG derivations to Italian and German translations.

egories to ensure that projected derivations are semantically equivalent to the input derivations (e.g., $N_2 / N_3 \neq N_4 / N_5$).

We now describe our derivation projection algorithm. Given a source derivation, a target sentence, and a word alignment, it attempts to produce a target derivation. Note that target derivations are entirely derived from the data by the algorithm; we do not make use of any hand-crafted language-specific rules.

Input The input to derivation projection consists of a source sentence E with a derivation D_E, a target sentence F which is a translation of E, and a (potentially ambiguous) alignment $\mathcal{A}$ which is a set of 1:N translation units $\langle\langle f\rangle, \mathbf{e}\rangle$ where f is a token in F and $\mathbf{e}$ is a subsequence (not necessarily contiguous) of tokens in E, as well as translation units $\langle\langle\rangle, \langle e\rangle\rangle$, indicating that the English word e is not aligned.

Output Derivation projection may succeed or fail; if it succeeds, the output is a derivation D_F for F.

Auxiliary Definitions $\mathcal{C}$ is the set of all categories. A *category assignment* c for a sequence of tokens $\mathbf{t}$ is a relation such that $c \subseteq \mathbf{t} \times \mathcal{C}$.[1] We write c_E for the category assignment relating tokens in E to the categories they have in D_E; this relation is a function. We write R^*_E for the set of type-changing rules used in D_E. We write RootCat(D) for the category of the root of a derivation D. Parse is a function that takes a sequence of tokens $\mathbf{t}$, a category assignment c for $\mathbf{t}$, and a set of type-changing rules R^*. It returns the set of all normal-form CCG derivations (Hockenmaier and Bisk, 2010) that can be built over $\mathbf{t}$ using R^*, forward/backward type raising and harmonic/crossing composition up to degree 2, with possible lexical categories determined by c. To deal with parsing ambiguity during derivation projection, we assume a function Choose that takes a non-empty set of derivations and returns one element. We will say more about it below.

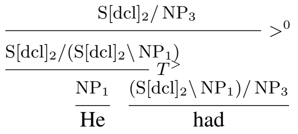


Figure 4: Two source-language categories are merged into one.

Step 1: Transfer Categories This step assigns categories to the words in F based on the categories of aligned words in E. This is straightforward for 1:1 translation units such as ⟨tre, three⟩, but 1:N translation units such as ⟨Aveva, He had⟩ need a bit more care. We define Merge as a partial function from subsequences of E to $\mathcal{C}$. For a single-token subsequence $e \in E$, Merge$(e) = c_E(e)$. For a longer subsequence $\mathbf{e}$, Merge$(\mathbf{e})$ = RootCat(Choose(Parse$(\mathbf{e}, c_E, R^*_E)$)) (if defined). For example, even though *He had* is not a constituent in Figure 1, it has a parse (shown in Figure 4), and so Merge(He had) = $\mathrm{S[dcl]}_2/\,\mathrm{NP}_3$. We then define a preliminary category assignment

---

[1] In a slight abuse of notation, we treat sequences of tokens as sets of tokens when convenient.

for F: $c_F = \{\langle f, \textsc{Merge}(\mathbf{e})\rangle | \langle\langle f\rangle, \mathbf{e}\rangle \in \mathcal{A}, \textsc{Merge}(\mathbf{e}) \text{ is defined}\}$.

Step 2: Transfer Type-changing Rules This step creates a set R^*_F of type-changing rules to be used in D_F. In addition to the type-changing rules used in D_E, we add $\mathrm{N} \Rightarrow \mathrm{NP}$ rules for English determiners that have no corresponding token in the target language. This is a common occurrence, especially with languages which have no articles, such as Czech, or where (some) articles are affixes rather than separate words, such as Swedish. Thus, $R^*_F = R^*_E \cup \{\mathrm{N}_i \Rightarrow \mathrm{NP}_j \,|\langle\langle\rangle, \langle e\rangle\rangle \in \mathcal{A}, c_E(e) = \mathrm{NP}_i \,/\, \mathrm{N}_j \text{ for some } i, j\}$.

Step 3: Flip Slashes This step adapts the directionality of slashes in the assigned categories, because the word order may be different in F than in E. We say that a category C' is a *flip variant* of category C ($\textsc{Flip}(C, C')$) if it is the same as C, except that slashes may lean a different way, as long as subcategories that are modifier categories in C (i.e., are of form X/X or $X\backslash X$, ignoring indices) remain so in C'. For example, in Figure 3(a), the category $(\mathrm{N}_2 \,/\, \mathrm{N}_3)/(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ has a flip variant $(\mathrm{N}_2 \,\backslash\, \mathrm{N}_3)/(\mathrm{N}_4 \,\backslash\, \mathrm{N}_5)$ whereas $(\mathrm{N}_2 \,\backslash\, \mathrm{N}_3)/(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ is not a flip variant because that would destroy the modifier status. In order to be able to construct a derivation for F even with word order different from E, we define a new category assignment: $c'_F = \{\langle f, C'\rangle | \langle f, C\rangle \in c_F, \textsc{Flip}(C, C')\}$. Similarly, we construct a set of type-changing rules with flip variants: $R^{*\prime}_F = \{X' \Rightarrow Y' | X \Rightarrow Y \in R^*_F, \textsc{Flip}(X, X'), \textsc{Flip}(Y, Y')\}$. This constructs more categories and type-changing rules than needed; for example, $(\mathrm{N}_2 \,/\, \mathrm{N}_3)\backslash(\mathrm{N}_4 \,/\, \mathrm{N}_5)$ is a flip variant for *molto* that cannot be used, as the argument category $\mathrm{N}_4 \,/\, \mathrm{N}_5$ does not appear on the left. Such spurious categories are discarded automatically in our implementation.

Step 4: Construct Derivation With c'_F and $R^{*\prime}_F$ constructed, we try to find a parse for F that has the same root category as D_E: $D_F = \textsc{Choose}(\{D | D \in \textsc{Parse}(F, c'_F, R^{*\prime}_F), \textsc{RootCat}(D) = \textsc{RootCat}(D_E)\})$ if defined; otherwise derivation projection fails and no derivation is returned.

Resolving Ambiguity Since parsing in steps 1 and 4 of derivation projection is guided by indexed categories and normal-form constraints, ambiguity primarily arises through ambiguous word alignments, which we use to achieve better projection coverage (see Section 5). For example, in Figure 1, *tre* might also be aligned to *sons*, and *three* to *figli*, giving rise to an additional (incorrect) parse. Our strategy for resolving such ambiguities is to prefer parses whose lexical categories result from word alignments with higher alignment scores. Our current implementations of PARSE and CHOOSE naively order parses by the score of the alignment that produced each lexical target category, greedily from left to right. Future work might improve upon this by ranking parses according to a global score.

4 The Learning Procedure

Given a parallel training corpus of source-target sentence pairs, we parse the source-language part using a source-language parser and run unsupervised word alignment on the entire corpus. Then, for each sentence pair, we run derivation projection using the generated source parses and alignments. If successful, we add the target derivation picked by CHOOSE to a target-language training set. Finally, we use this training set to train a target-language parser in the usual way.

5 Experiments[2]

Target Languages Following prior work, we evaluate the induced CCG parsers in terms of unlabeled attachment score (UAS) on the data of the PASCAL unsupervised grammar induction challenge (Gelling et al., 2012), which includes eight different languages other than English: Arabic, Czech, Danish, Basque, Dutch, Portuguese, Slovenian, and Swedish. For qualitative evaluation, we use German, Italian, and Dutch. We acknowledge the importance of testing our approach on a more typologically diverse range of languages, but leave this for future work.

Training Data To start learning to parse a new language, one needs short and simple example sentences. This is true for human learners, and presumably also for computers. We therefore used the Tatoeba corpus[3] for training, a multilingual parallel corpus gathered by volunteers and aimed at language learners. We extracted English-X sentence pairs for various languages X and tokenized

---

[2] The training data, code, and configurations are available at `https://github.com/texttheater/xlci`.

[3] `https://tatoeba.org`

Parallel corpus	sentences	∅ tokens
eng-ara	19,502	5.8
eng-ces	11,147	6.2
eng-dan	21,409	7.1
eng-deu	244 140	8.1
eng-eus	1,882	6.4
eng-ita	412 427	6.5
eng-nld	44 126	7.5
eng-por	161 126	7.2
eng-slv	835	6.3
eng-swe	24 206	6.4

Table 1: Number of sentence pairs and average number of tokens per target-language sentence in the data extracted from Tatoeba.

them using UDPipe (Straka and Straková, 2017), not making use of the optional multiword token subdivision feature. The resulting parallel corpora are summarized in Table 1.

Source-language Parser To create derivations to project, we needed a suitable parser for our source language, English. Commonly, English CCG parsers are trained on CCGbank (Hockenmaier and Steedman, 2007) or its derivative CCGrebank (Honnibal et al., 2010). However, these treebanks use special categories for punctuation and conjunctions, which would complicate derivation projection. We thus took CCGrebank, automatically transformed it to use normal categories for these cases (an example is shown in Figure 5), and trained the EasyCCG parser (Lewis and Steedman, 2014) on that. The resulting model was used to produce parses for the English portions of our parallel training corpora.

Word Alignments and Derivation Projection For word-aligning the parallel training data, we used GIZA++ with default settings (Och and Ney, 2003). We generated alignments $\mathcal{A}$ for each sentence pair by taking the union of the n-best GIZA++ alignments, trying out different values for n between 1 and 5.

Target-language Parser Again, we used EasyCCG. Its supertagger component is trained on sentences where the words are annotated with categories. We used the projected derivations for that. We used the Polyglot word embeddings (Al-Rfou et al., 2013). Since we do not have supertagged validation sets for the target languages, the number of training epochs was fixed at 3 following initial experimentation. The parser component requires no training, but for decoding, we made some modifications to it to generalize beyond English: instead of a hard-coded set for English, the modified parser uses the set of unary rules used in the projected derivations for the respective language. It also implements all composition rules up to degree 2 rather than an English-specific subset, and it implements Hockenmaier and Bisk's normal-form constraints.

Dependency Conversion For evaluating the induced target-language parsers on the PASCAL benchmark, we have to be able to convert their output derivations to dependency trees, as exemplified in Figure 6. The simplest way to do this is to make arguments dependents of their functors, similar to Koller and Kuhlmann (2009). That is, a word v with the (indexed) category $X|Y\alpha$

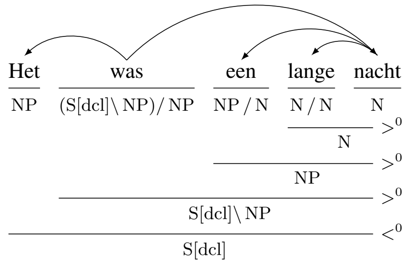


Figure 6: An example derivation and its conversion into a dependency tree.

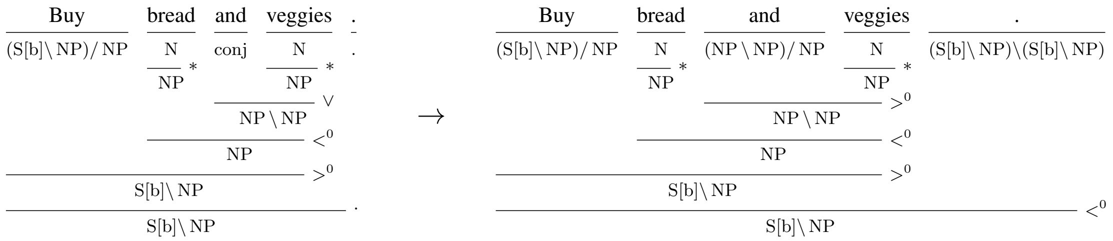


Figure 5: Elimination of special categories and rules for punctuation and coordination.

categories	description	ara	ces	dan	eus	nld	por	slv	swe
$X\mid X$	modifier	✓	✓	✓	✓	✓	✓	✓	✓
$\mathrm{NP}\mid\mathrm{N}, \mathrm{NP}\mid(\mathrm{N}\mid\mathrm{PP})$	determiner	✓	✓		✓	✓	✓	✓	✓
$\{(\mathrm{S}\mid\mathrm{S}), (\mathrm{S}\mid\mathrm{NP})\mid(\mathrm{S}\mid\mathrm{NP})\}\mid$ $\{\mathrm{S[dcl]}, \mathrm{S[to]}, (\mathrm{S[ng]}\mid\mathrm{NP})\}$	subordinating conjunction						✓	✓	✓
$\mathrm{S[em]}\mid\mathrm{S[dcl]}$, $(\mathrm{S[to]}\mid\mathrm{NP})\mid(\mathrm{S[b]}\mid\mathrm{NP})$	complementizer						✓	✓	✓
$\{\mathrm{N}\mid\mathrm{N}, \mathrm{NP}\mid\mathrm{NP}\}\mid(\mathrm{S[dcl]}\mid\mathrm{NP})$	relative pronoun						✓	✓	✓
$\{\mathrm{PP}, \mathrm{N}\mid\mathrm{N}, \mathrm{NP}\mid\mathrm{NP}, \mathrm{S}\mid\mathrm{S},$ $(\mathrm{S}\mid\mathrm{NP})\mid(\mathrm{S}\mid\mathrm{NP})\}\mid\mathrm{NP}$	adposition							✓	
$\mathrm{S[dcl]}\mid\mathrm{S[b]}$, $\mathrm{S[\{b, dcl, ng, pt\}]}\mid\mathrm{S[ng]}$, $\mathrm{S[\{b, dcl, ng, pt\}]}\mid\mathrm{S[pt]}$	auxiliary verb		✓					✓	

Table 2: Functional categories which in dependency conversion become dependents of their first argument, rather than the other way around, depending on the treebank-specific conventions. Braces denote alternatives.

becomes the head of a word w with category $Y\beta$, where $| \in \{/, \backslash\}$ and α, β stand for any number of additional argument categories with slashes. However, for some categories the head-dependent relation should be inverted. For example, if $X|Y$ is a modifier category, then w becomes the head of v, and any dependents v would get because of additional arguments in X become dependents of w instead. Because dependency treebanks differ in their conventions for attaching certain function words, certain non-modifier categories also need to be treated in this inverted way. They are shown in Table 2. Note that this fine-grained control is only possible because we induce relatively rich CCG categories; by contrast, Bisk and Hockenmaier (2013) use only two basic categories (S and N) and therefore cannot distinguish, e.g., determiners from attributive adjectives (N/N) or *to*-complementizers from auxiliary verbs ($((\mathrm{S}\backslash\mathrm{N})/(\mathrm{S}\backslash\mathrm{N}))$). They do apply treebank-specific conversion rules for coordination, which we also implement.

Hyperparameter Tuning We use the PASCAL development data to tune the hyperparameter n which controls how many GIZA++ alignments are used for derivation projection. Table 3 shows how many sentence pairs our parallel training corpus contains for each of the eight languages, how many of the derivations are successfully projected

language sentence pairs		ara 19 502	ces 11 147	dan 21 409	eus 1 882	nld 44 026	por 161 126	slv 835	swe 24 206
$n = 1$	projected	27.4%	30.5%	49.8%	20.6%	36.3%	30.8%	32.7%	48.8%
	ambiguity	1.029	1.044	1.011	1.111	1.046	1.015	1.040	1.014
	UAS	45.9%	43.6%	61.6%	18.4%	65.7%	**64.8%**	26.9%	**65.0%**
$n = 2$	projected	33.6%	36.6%	52.2%	23.8%	40.0%	34.7%	38.4%	52.3%
	ambiguity	1.252	1.230	1.169	1.266	1.092	1.075	1.215	1.143
	UAS	**46.3%**	45.7%	61.2%	25.6%	**65.9%**	64.2%	28.2%	63.2%
$n = 3$	projected	38.1%	40.4%	53.4%	26.0%	41.6%	37.1%	41.8%	54.2%
	ambiguity	1.379	1.364	1.226	1.325	1.118	1.114	1.289	1.193
	UAS	35.8%	**46.4%**	**62.5%**	24.8%	64.3%	63.0%	29.0%	64.6%
$n = 4$	projected	41.8%	43.4%	54.3%	28.9%	42.7%	39.2%	43.6%	55.6%
	ambiguity	1.484	1.474	1.269	1.397	1.142	1.152	1.352	1.232
	UAS	38.1%	45.3%	60.0%	26.1%	65.0%	62.0%	**32.2%**	63.1%
$n = 5$	projected	45.2%	45.9%	55.0%	31.3%	43.8%	41.2%	46.0%	57.0%
	ambiguity	1.592	1.583	1.318	1.461	1.174	1.207	1.409	1.278
	UAS	33.9%	45.8%	60.4%	**27.4%**	64.4%	61.8%	30.2%	63.7%

Table 3: Effects of varying the projection hyperparameter n: percentage of successfully projected source derivations, mean ambiguity (how many target derivations are found per projected source derivation), and UAS of the trained system on the PASCAL development data (max sentence length 15, not counting punctuation).

		ara	cze	dan	eus	nld	por	slv	swe
train tokens (PASCAL)		5 470	436 126	25 341	81 345	78 737	158 648	54 032	61 877
system	input								
BH13 (publ.)	gold POS	**65.1%**	**50.7%**	58.5%	**45.0%**	54.4%	62.9%	**46.4%**	**66.9%**
BH13 (repl.)	gold POS	45.4%	38.3%	25.1%	37.3%	54.9%	51.0%	41.8%	63.2%
BCH15 (publ.)	raw text	43.7%	32.4%	37.7%	35.2%	43.8%	51.6%	23.6%	52.9%
train tokens (Tatoeba)		19 502	11 147	21 409	1 882	44 026	161 126	835	24 206
system	input								
EB16	parallel, POS	26.4%	28.4%	35.8%	22.1%	40.4%	39.4%	27.2%	26.2%
ours	parallel, embeddings	46.8%	44.9%	**63.0%**	29.0%	**61.4%**	**67.8%**	35.0%	63.7%

Table 4: UAS of different systems on the PASCAL test data (max sentence length 15, not counting punctuation).

for each value of n, and how accurately the development data is parsed. The numbers show the importance of having enough training examples: Portuguese, Swedish, Dutch, and Danish are leading in terms of corpus size and parsing accuracy, whereas Basque and Slovene are far behind in both. Arabic is a bit of an outlier, performing worse than Czech despite a considerably larger corpus. The ratio of successfully projected derivations increases as n is increased. This makes for more training data but also more noise; different languages peak at different values for n. Languages with little training data (Slovene, Basque, Czech) most clearly profit from more projected derivations. For the final tests, we set $n \leq 5$ to maximize UAS on the development data for each language.

Baselines We compare with two unsupervised CCG induction system and one other cross-lingual CCG induction system. To our knowledge, Bisk and Hockenmaier (2013) represents the state of the art in unsupervised CCG induction. It does, however, use gold-standard POS tags in the training and testing data. These seem to be essential, as a variant of this system which does not rely on POS tags performed much worse (Bisk et al., 2015). Our system does not rely on POS tags but on parallel data and word embeddings instead, which is an advantage as parallel data and word embeddings may be more readily available than POS tags for new languages. We also compare with the system of Evang and Bos (2016), a cross-lingual system similar to ours which was previously only evaluated on a semantic parsing task, not on syntactic dependencies. For the unsupervised systems, we report published results when trained on the complete PASCAL data. For BH13, we also include our best replication attempt using the original software and training data, falling short of the published results as the exact configurations appear to be lost. For the cross-lingual systems which require parallel training data, we train on the Tatoeba dataset. All test scores are on the PASCAL test set, limited to sentences with at most 15 tokens, not counting punctuation.

Results Test results are shown in Table 4. Despite not using POS tags, our system outperforms the cross-lingually supervised system of Evang and Bos (2016) by a large margin on all languages. It also outperforms the unsupervised system of Bisk et al. (2015) on 6 out of 8 languages, and that of Bisk and Hockenmaier (2013) (which uses POS tags) on 3 out of 8 languages. This is also in spite of these two unsupervised systems being trained on more (albeit not parallel) data, which even included the test data.

6 The Induced Lexicons

Have our cross-lingually trained parsers acquired language-specific knowledge? Based on what we know about the syntactic differences between English, German, Italian, and Dutch, we would expect certain categories to be more prominent in the lexicon for some languages than for others:

1. English word order in transitive clauses is almost always SVO, whereas for German and Dutch, SVO is the typical order for main clauses, and SOV the typical order for subordinate clauses (Dryer, 2013c). Thus, we expect the English parser to almost always assign category $(\mathrm{S[X]} \backslash \mathrm{NP}) / \mathrm{NP}$ to transitive verbs, whereas we expect German and Dutch transitive verbs to be split between $(\mathrm{S[X]} \backslash \mathrm{NP}) / \mathrm{NP}$ and $(\mathrm{S[X]} \backslash \mathrm{NP}) \backslash \mathrm{NP}$.

	Category	English	German	Italian	Dutch
1	$(S[dcl]\backslash NP)/NP$	.0366	.0445	.0256	.0389
	$(S[dcl]\backslash NP)\backslash NP$	.0000	.0056	.0046	.0061
	$(S[b]\backslash NP)/NP$	.0284	.0032	.0147	.0044
	$(S[b]\backslash NP)\backslash NP$	.0000	.0169	.0043	.0151
2	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	.0237	.0184	.0150	.0180
3	$(S[b]\backslash NP)\backslash PR$	.0000	.0000	.0000	.0000
	$(S[b]\backslash NP)/PR$	.0004	.0000	.0000	.0000
4	N/N	.0309	.0299	.0213	.0316
	$N\backslash N$	.0013	.0018	.0099	.0018
	$(N/N)/(N/N)$	.0016	.0018	.0003	.0012
	$(N\backslash N)/(N\backslash N)$	.0001	.0000	.0008	.0000
5	$S[dcl]$	.0000	.0000	.0012	.0001
	$S[dcl]/NP$	.0004	.0013	.0115	.0007

Table 5: Frequency (per sentence) of lexical categories in the output of different parsers when applied to the Tatoeba data, illustrating learned language-specifics. The averages for English are calculated over all three parallel training corpora.

2. German, Italian and Dutch do not have *do*-support for negation (Miestamo, 2013), so we expect the category $(S[dcl]\backslash NP)/(S[b]\backslash NP)$ to be less common in them than in English.

3. In the infinitive mood, German and Dutch spell particle verbs as one token (e.g., *ausgehen*, *uitgaan*), unlike English which spells them apart (*go out*) (Dehé, 2015). Thus, we expect categories such as $(S[b]\backslash NP)\backslash PR$ or $(S[b]\backslash NP)/PR$ to be nonexistent in German and Dutch but common in English.

4. In Italian, attributive adjectives commonly appear after the noun they modify, whereas in English they almost always appear before (Dryer, 2013b). We thus expect the category $N\backslash N$ to be much more common in Italian than in English. Likewise, for adverbs modifying these adjectives, we expect $(N\backslash N)/(N\backslash N)$ in Italian but not in English (cf. Figure 3(a)).

5. In Italian, subject pronouns are frequently dropped (Dryer, 2013a), so we expect to frequently see verb categories like $S[X]$ and $S[X]/NP$, which are uncommon in English (cf. Figure 1).

To quantify these effects on comparable data for all four languages, we applied our parsers to the Tatoeba data to see how often they predict each category for a word. The relative numbers are shown in Table 5. We find all five expectations confirmed, suggesting that training parsers on projected derivations can indeed teach them specifics of each language's syntax.

7 Related Work

Recent years have seen much interest in cross-lingual learning, that is, learning tagging and parsing models for languages without training data for that language, instead relying on training data or existing systems for another language, and on parallel data to transfer knowledge from one language to the other. This is either done by automatically projecting source-language annotations from the source text to the target text (Yarowsky et al., 2001; Hwa et al., 2005; Tiedemann, 2014; Rasooli and Collins, 2015; Johannsen et al., 2016; Agić et al., 2016; Damonte and Cohen, 2018), sharing parameters between models for different languages (Zeman and Resnik, 2008; Ganchev et al., 2009; McDonald et al., 2011; Naseem et al., 2012; Täckström et al., 2013; de Lhoneux et al., 2018), or automatically translating the text from the source language to the target language and synchronously projecting the annotations (Tiedemann et al., 2014). Our work is an application of the first approach to CCG, which as a grammar formalism provides a more systematic framework for the study of syntax and for compositional interpretation than dependency parsers.

Apart from unsupervised syntactic CCG induction, CCG induction has also been done as part of learning semantic parsers, where supervision typically comes from logical forms, and syntax is treated as latent. Much of this work starts with a manually specified inventory of syntactic categories and only learns the semantic parts (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Reddy et al., 2014; Artzi et al., 2015), whereas we start with no knowledge of the syntactic categories of the target language. Kwiatkowksi et al. (2010); Kwiatkowski et al. (2011); Bisk et al. (2016); Evang and Bos (2016) also learn the syntactic categories but evaluate their parsers only on semantic tasks, so it is unclear how linguistically plausible the induced CCGs are.

Earlier versions of the projection algorithm presented here were used in Evang and Bos (2016) for cross-lingual semantic parsing, and in Abzianidze et al. (2017) for bootstrapping a multilingual CCG treebank.

8 Conclusions and Future Work

Cross-lingual learning is a promising strategy whenever annotated training data for the target language is not available, but annotated training data for a source language as well as a parallel corpus is. This paper has introduced a method to apply this idea to syntactic CCG parsing, based on an algorithm for projecting CCG derivations along word alignments.

Compared to existing work on CCG induction, our method relies on parallel data and word embeddings but obviates the need for POS tags while in many cases outperforming methods that do use POS tags, and with less training data. This should make our method suitable for bringing multilingualism to CCG-based semantic parsers that so far rely on hand-written grammars.

In addition, we have shown that the induced lexicons reflect linguistic knowledge about the target languages. Our method also induces more fine-grained categories than previous approaches. It can thus also be a valuable asset for bootstrapping linguistically informed parsers and CCG treebanks for new languages.

There are various avenues to improving and extending derivation projection: alignment ambiguity could be handled with a global score, and multiple possible parses could be included in the target-language set, potentially improving the tradeoff between the number of projected derivations and the amount of noise. To increase the range of structural differences between languages that can be handled, derivation projection could be extended to consider sub-token units and to handle 1:n translation units in addition to n:1 ones.

Acknowledgments

The author would like to thank Lasha Abzianidze and Johan Bos for invaluable help in planting the seeds for this research, Yonatan Bisk for help with replication, and Laura Kallmeyer, Jakub Waszczuk, and all anonymous reviewers for valuable feedback. This research was supported by the NVIDIA Corporation with a Titan Xp GPU. It was partly carried out within the TreeGraSP project, funded by a Consolidator Grant of the European Research Council (ERC).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247. Association for Computational Linguistics.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Sebastian Beschke and Wolfgang Menzel. 2018. Graph algebraic combinatory categorial grammar. In *Proceedings of the Seventh Joint Conference on Lexical* *and Computational Semantics*, pages 54–64. Association for Computational Linguistics.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2022–2027. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 33–36.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual abstract meaning representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155. Association for Computational Linguistics.

Nicole Dehé. 2015. *Particle verbs in Germanic*, volume 1, pages 611–626. De Gruiter Mouton.

Matthew S. Dryer. 2013a. Expression of pronominal subjects. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer. 2013b. Order of adjective and noun. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer. 2013c. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Kilian Evang and Johan Bos. 2016. Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 25th International Conference on Computational Linguistics*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the OPTAFNLP: Volume 1*, pages 369–377.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Jo ao Graça. 2012. The PASCAL challenge on grammar induction. In *NAACL HLT Workshop on Induction of Linguistic Structure*, pages 64–80, Montreal, Canada.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 465–473.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Anders Johannsen, Željko Agić, and Anders Søgaard. 2016. Joint part-of-speech and dependency projection from multiple sources. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 561–566. Association for Computational Linguistics.

Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 460–468. Association for Computational Linguistics.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Matti Miestamo. 2013. Symmetric and asymmetric standard negation. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, 2:377–392.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Ole Tange. 2018. *GNU Parallel 2018*. Ole Tange.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Eighteenth Conference on Computational Natural Language Learning*.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.

Density Matching for Bilingual Word Embedding

Chunting Zhou, Xuezhe Ma, Di Wang, Graham Neubig
Language Technologies Institute
Carnegie Mellon University
`ctzhou,xuezhem,diwang,gneubig@cs.cmu.edu`

Abstract

Recent approaches to cross-lingual word embedding have generally been based on linear transformations between the sets of embedding vectors in the two languages. In this paper, we propose an approach that instead expresses the two monolingual embedding spaces as probability densities defined by a Gaussian mixture model, and matches the two densities using a method called normalizing flow. The method requires no explicit supervision, and can be learned with only a seed dictionary of words that have identical strings. We argue that this formulation has several intuitively attractive properties, particularly with the respect to improving robustness and generalization to mappings between difficult language pairs or word pairs. On a benchmark data set of bilingual lexicon induction and cross-lingual word similarity, our approach can achieve competitive or superior performance compared to state-of-the-art published results, with particularly strong results being found on etymologically distant and/or morphologically rich languages.[1]

1 Introduction

Cross-lingual word embeddings represent words in different languages in a single vector space, capturing the syntactic and semantic similarity of words across languages in a way conducive to use in computational models (Upadhyay et al., 2016; Ruder et al., 2017). These embeddings have been shown to be an effective tool for cross-lingual NLP, e.g. the transfer of models trained on high-resource languages to low-resource ones (Klementiev et al., 2012; Guo et al., 2015; Zoph et al., 2016; Zhang et al., 2018; Gu et al., 2018) or unsupervised learning (Artetxe et al., 2018c).

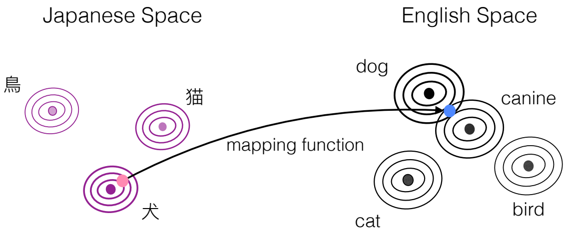


Figure 1: An illustration of our method. Thicker lines represent higher mixture weights, linked to word frequency. The pink point is a continuous training sample in the Japanese embedding space while the blue point is a mapped point in the English space.

There are two major paradigms in the learning of cross-lingual word embeddings: "online" and "offline". "Online" methods learn the cross-lingual embeddings directly from parallel corpora (Hermann and Blunsom, 2014), optionally augmented with monolingual corpora (Gouws et al., 2015). In contrast, "offline" approaches learn a bilingual mapping function or multilingual projections from pre-trained monolingual word embeddings or feature vectors (Haghighi et al., 2008; Mikolov et al., 2013; Faruqui and Dyer, 2014). In this work, we focus on this latter offline approach.

The goal of bilingual embedding is to learn a shared embedding space where words possessing similar meanings are projected to nearby points. Early work focused on supervised methods maximizes the similarity of the embeddings of words that exist in a manually-created dictionary, according to some similarity metric (Mikolov et al., 2013; Faruqui and Dyer, 2014; Jawanpuria et al., 2018; Joulin et al., 2018). In contrast, recently proposed unsupervised methods frame this problem as minimization of some form of distance between the whole set of discrete word vectors in the chosen vocabulary, e.g. Wasserstein distance or Jensen–Shannon divergence (Xu et al., 2018; Conneau et al., 2017; Zhang et al., 2017; Grave

---

[1] Code/scripts can be found at `https://github.com/violet-zct/DeMa-BWE`.

Proceedings of NAACL-HLT 2019, pages 1588–1598
Minneapolis, Minnesota, June 2 - June 7, 2019.

et al., 2018). While these methods have shown impressive results for some language pairs despite the lack of supervision, regarding the embedding space as a set of discrete points has some limitations. First, expressing embeddings as a single point in the space doesn't take into account the inherent uncertainty involved in learning embeddings, which can cause embedding spaces to differ significantly between training runs (Wendlandt et al., 2018). Second, even in a fixed embedding space the points surrounding those of words that actually exist in the pre-trained vocabulary also often are coherent points in the embedding space.

In this work, we propose a method for **density matching for bilingual word embedding (DeMa-BWE)**. Instead of treating the embedding space as a collection of discrete points, we express it as a probability density function over the entire continuous space over word vectors. We assume each vector in the monolingual embedding space is generated from a Gaussian mixture model with components centered at the pretrained word embeddings (Fig. 1), and our approach then learns a bilingual mapping that most effectively matches the two probability densities of the two monolingual embedding spaces.

To learn in this paradigm, instead of using the pre-trained word embeddings as fixed training samples, at every training step we obtain samples from the Gaussian mixture space. Thus, our method is exploring the entire embedding space instead of only the specific points assigned for observed words. To calculate the density of the transformed samples, we use volume-preserving invertible transformations over the target word embeddings, which make it possible to perform density matching in a principled and efficient way (Rezende and Mohamed, 2015; Papamakarios et al., 2017; He et al., 2018). We also have three additional ingredients in the model that proved useful in stabilizing training: (1) a back-translation loss to allow the model to learn the mapping jointly in both directions, (2) an identical-word-matching loss that provides weak supervision by encouraging the model to have words with identical spellings be mapped to a similar place in the space, and (3) frequency-matching based Gaussian mixture weights that accounts for the approximate frequencies of aligned words.

Empirical results are strong; our method is able to effectively learn bilingual embeddings that achieve competitive or superior results on the MUSE dataset (Conneau et al., 2017) over state-of-the-art published results on bilingual word translation and cross-lingual word similarity tasks. The results are particularly encouraging on etymologically distant or morphologically rich languages, as our model is able to explore the integration over the embedding space by treating the space as a continuous one. Moreover, unlike previous unsupervised methods that are usually sensitive to initialization or require sophisticated optimization procedures, our method is robust and requires no special initialization.

2 Background: Normalizing Flows

In this section, we will briefly describe normalizing flows - the backbone of DeMa-BWE.

As mentioned in the introduction and detailed later, our model is based on matching two probability density functions, one representing the source embedding space and one representing the target embedding space. To learn in this framework, we will use the concept of *normalizing flows* (Rezende and Mohamed, 2015). We will explain them briefly here, but refer readers to Rezende and Mohamed (2015) for details due to space constraints.

Concretely, let $\mathbf{u}$ denote a high dimensional random vector (e.g. representing a point in the source embedding space) and $\mathbf{z}$ be a latent variable that corresponds to $\mathbf{u}$ (e.g. a point in the target embedding space). Flow-based generative models (Kingma et al., 2016; Kingma and Dhariwal, 2018) learn invertible transformations $f_\theta(\mathbf{z})$ from the distribution over $\mathbf{z}$ to the distribution over $\mathbf{u}$. The generative story of the model is defined as:

$$\mathbf{z} \sim p_\theta(\mathbf{z}), \quad \mathbf{u} = f_\theta(\mathbf{z}) \tag{1}$$

where $p_\theta(\mathbf{z})$ is the prior distribution. This prior can be any distribution for which we can tractably compute the density of sample points $\mathbf{z}$. A common choice of such distribution is a spherical multivariate Gaussian distribution: $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$ (Dinh et al., 2016). Assuming the transformation function $f_\theta(\cdot)$ is invertible, using the rule for change of variables, the probability density of $\mathbf{u}$ can be calculated as:

$$p_\theta(\mathbf{u}) = p_\theta(\mathbf{z}) |\det(J(f^{-1}(\mathbf{u})))| \tag{2}$$

where $\det(J(f^{-1}(\mathbf{u})))$ is determinant of the Jacobian matrix of the function inverse. This term accounts for the way in which f locally expands or contracts regions of $\mathbf{z}$, and enforces the invertibility of the function. A "normalizing flow" is a cascaded sequence of such invertible transformations, which is learned by maximizing the density in Equation (2) over observed data points $\mathbf{u}$. One computational issue with these models lies in calculating the Jacobian matrix, which is expensive in the general case. A common method is to choose transformations whose Jacobians' are a triangular matrix, which renders this computation tractable (Dinh et al., 2016; Papamakarios et al., 2017).

3 Proposed Method

In this section, we present notation used in our method, describe the prior we define for the monolingual embedding space, then detail our density matching method.

3.1 Notation

Given two sets of independently trained monolingual embeddings, the problem of bilingual embedding mapping is to learn a mapping function that aligns the two sets in a shared space. Let $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^d$ denote vectors in the source and target language embedding space respectively. Let x_i and y_j denote an actual word in the pretrained source and target vocabularies respectively. Words are sorted by their occurrence counts in the monolingual corpus and the index of the word represents its rank. We use $\mathbf{x}_i$ and $\mathbf{y}_j$ to denote the pretrained word embeddings for word x_i in the source language and word y_j in the target language respectively. Given a pair of languages s and t, our approach learns two mapping functions: f_{xy} that maps the source embedding to the target space and f_{yx} that gives the mapping in the reverse direction.

3.2 Density Estimation in Monolingual Space

To learn the mapping, we project a vector $\mathbf{x}$ in the source embedding space into the target space $\mathbf{y}$. We learn this mapping by maximizing the density of data points in the source space. The density can be computed using the idea of normalizing flow described above. Thus, for the the monolingual embedding spaces, we need to define tractable density functions $p(\mathbf{x})$ and $p(\mathbf{y})$.

While any number of functions could be conceived to calculate these densities, in the current method we opt to use a Gaussian Mixture Model (GMM) with Gaussian components centered at each pretrained word embedding. This is motivated by the assumption that embeddings are likely to appear in the neighborhood of other embeddings, where we define the "neighborhood" to be characterized as closeness in Euclidean space, and the uncertainty of each neighborhood as being Gaussian. Concretely, let N_x and N_y denote the number of pretrained word embeddings that serve as Gaussian component centers during training for the source and target languages, respectively. Then we can express the density of any point in the source embedding space as:

$$p(\mathbf{x}) = \sum_{i \in \{1,\ldots,N_x\}} \pi(x_i)\tilde{p}(\mathbf{x}|x_i) \tag{3}$$

where $\pi(x_i)$ is the frequency of word x_i normalized within the N_x component words, and $\tilde{p}(\mathbf{x}|x_i)$ is a Gaussian distribution centered at the embedding of word x_i. We simply use a fixed variance σ_x^2 for all Gaussian components:

$$\tilde{p}(\mathbf{x}|x_i) = \mathcal{N}(\mathbf{x}|\mathbf{x}_i, \sigma_x^2\mathbf{I}) \tag{4}$$

Similarly, the density of any point in the target embedding space can be written as:

$$p(\mathbf{y}) = \sum_{j \in \{1,\ldots,N_y\}} \pi(y_j)\tilde{p}(\mathbf{y}|y_j) \tag{5}$$

where $\tilde{p}(\mathbf{y}|y_j) = \mathcal{N}(\mathbf{y}|\mathbf{y}_j, \sigma_y^2\mathbf{I})$.

3.3 Density Matching

With the Gaussian mixture model as the prior distribution in the monolingual space, our goal is to learn a mapping function from one embedding space to the other such that the log probabilistic density is maximized in the source space.

While we are jointly learning the two mapping functions f_{xy} and f_{yx} simultaneously, for conciseness we will illustrate our approach using the source to target mapping f_{xy}. First, a continuous vector $\mathbf{x}$ is sampled from the Gaussian mixture model (Eq. (3)) by sampling $x_i \sim \pi(x_i)$ then $\mathbf{x} \sim \tilde{p}(\mathbf{x}|x_i)$ (4). Next, we apply the mapping function f_{xy} to obtain the transformed vector $\mathbf{y}$ in the target space. Concretely, the mapping functions we employ in this work are two linear transformations: $f_{xy}(\cdot) = \mathbf{W}_{xy}\cdot$ and $f_{yx} = \mathbf{W}_{yx}\cdot$. Connecting to the transformation function in Sec. 2, we see that $\mathbf{x} = f(\mathbf{y}) = \mathbf{W}_{xy}^{-1}\mathbf{y}$, $\mathbf{y} = f^{-1}(\mathbf{x}) = \mathbf{W}_{xy}\mathbf{x}$, and $J(f^{-1}(\mathbf{x})) = \mathbf{W}_{xy}$. We can then express the log density of a sample $\mathbf{x}$ as:

$$\log p(\mathbf{x}; \mathbf{W}_{xy}) = \log p(\mathbf{y}) + \log \left|\det(\mathbf{W}_{xy})\right| \tag{6}$$

where the Jacobian regularization term accounts for the volume expansion or contraction resulting from the projection matrix $\mathbf{W}_{xy}$.

We maximize the likelihood function which is equivalent to minimizing expectation of the KL-divergence between the prior distribution and the model distribution of $\mathbf{x}$. This provides a natural objective for optimization:

$$\text{minimize: } \mathrm{KL}(p(\mathbf{x})||p(\mathbf{x};\mathbf{W}_{xy})) \tag{7}$$

By replacing $\mathbf{W}_{xy}\mathbf{x}$ with $\mathbf{y}$, this is equivalent to maximizing the log density of transformed source samples in the target space (see Eq. (6)):

$$\mathcal{L}_{xy} = \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log p(\mathbf{y}) + \log\left|\det(\mathbf{W}_{xy})\right|] \tag{8}$$

The objective $\mathcal{L}_{xy}$ contains two parts: the log density function $\log p(\mathbf{y})$ and a regularization term $\log\det(\mathbf{W}_{xy})$. Likewise, for the target to source mapping $\mathbf{W}_{yx}$, we have the density matching objective $\mathcal{L}_{yx}$.

Conditional Density Matching The above marginal density matching method does not take into account the dependency between the embeddings in the two monolingual spaces. To address this issue, we extend the density matching method to the conditional density function:

$$\log p(\mathbf{x}|x_i;\mathbf{W}_{xy}) = \log p(\mathbf{y}|x_i) + \log\left|\det(\mathbf{W}_{xy})\right|$$

The conditional density $p(\mathbf{y}|x_i)$ is the prior distribution in this simple normalizing flow, and for this we use a Gaussian mixture model in the target monolingual space:

$$\begin{aligned} p(\mathbf{y}|x_i) &= \sum_{j\in\{1,\dots,N_y\}} p(\mathbf{y}, y_j|x_i) \\ &= \sum_{j\in\{1,\dots,N_y\}} \tilde{p}(\mathbf{y}|y_j)\pi(y_j|x_i) \end{aligned} \tag{9}$$

Similarly, where $\tilde{p}(\mathbf{y}|y_j)$ is the Gaussian density function in the mixture model defined in Equation (5). $\pi(y_j|x_i)$ allows us to incorporate a-priori assumptions about whether two words are likely to match. In fact, the density matching method in Eq. (6) can be regarded as a special case of the conditional density matching method by adopting a naive prior $\pi(y_j|x_i) := \pi(y_j)$. However previous work (Zhang et al., 2017) has noted that word frequency information is a strong signal – words with similar frequency are likely to be matched – and thus we use $\pi(y_j|x_i)$ to incorporate this prior knowledge.

In this work, we assume that the frequencies of aligned bilingual words in the individual monolingual corpus should be correlated, and to match words that are ranked similarly, we model the Gaussian mixture weights as the negative absolute difference between log-scale word ranks and normalize over all the target Gaussian component words by a $\mathrm{softmax}$ function with temperature τ:

$$\pi(y_j|x_i) = \frac{\exp(-|\log(j) - \log(i)|)/\tau}{\sum_{k=1}^{N_y}\exp(-|\log(k) - \log(i)|/\tau)}$$

Thus, if a word x_i has similarly frequency rank as word y_j, the sample $\mathbf{x}$ from the Gaussian distribution centered at $\mathbf{x}_i$ will be assigned higher weight for the component $\tilde{p}(\mathbf{y}|y_j) = \tilde{p}(\mathbf{W}_{xy}\mathbf{x}|y_j)$. Although this assumption will not hold always (e.g. for languages that have different levels of morphological complexity), intuitively we expect that using this signal will help more overall than it will hurt, and empirically we find that this weighting is not sensitive to language variation and works well in practice.

The updated objective is

$$\begin{aligned} \mathcal{L}_{xy} &= \mathrm{E}_{x_i\sim\pi(x_i)}[\mathrm{KL}(\tilde{p}(\mathbf{x}|x_i)||p(\mathbf{x}|x_i;\mathbf{W}_{xy}))] \\ &= \mathrm{E}_{x_i\sim\pi(x_i),\mathbf{x}\sim\tilde{p}(\mathbf{x}|x_i)}\big[\log p(\mathbf{y}|x_i) \\ &\quad + \log\left|\det(\mathbf{W}_{xy})\right|\big] \end{aligned} \tag{10}$$

In the conditional density above, both the frequency-matching weight and the Gaussian density function play an important role in matching the density of a source-space sample with the target embedding space. The former matches bilingual words with their frequency ranks while the latter matches words with their vector distances.

3.4 Weak Orthogonality Constraint

A common choice of bilingual mapping function is a linear transformation with an orthogonality constraint. Various motivations have been proposed for the orthogonality constraint such as length normalization of embeddings (Xing et al., 2015), and reversible mapping (Smith et al., 2017). In this work, we add a weak orthogonality constraint to the bilingual mappings via a back-translation loss as follows:

$$\begin{aligned} \mathcal{L}_{bt} &= \mathrm{E}_{x_i\sim\pi(x_i),\mathbf{x}\sim\tilde{p}(\mathbf{x}|x_i)}\left[g(\mathbf{W}_{yx}\mathbf{W}_{xy}\mathbf{x},\mathbf{x})\right] \\ &\quad + \mathrm{E}_{y_j\sim\pi(y_j),\mathbf{y}\sim\tilde{p}(\mathbf{y}|x_j)}\left[g(\mathbf{W}_{xy}\mathbf{W}_{yx}\mathbf{y},\mathbf{y})\right] \end{aligned}$$

where $g(\cdot, \cdot) = 1 - \text{cosine}(\cdot, \cdot)$ is the cosine loss. Jointly learning the two mapping matrices by minimizing this cyclic loss encourages $\mathbf{W}_{xy}$ and $\mathbf{W}_{yx}$ to be orthogonal to each other.

3.5 Weak Supervision with Identical Words

To reduce the search space of the mapped bilingual embeddings, we add an additional weakly supervised loss over words that have identical strings in both the source and target languages denoted $\mathcal{W}_{id}$.

$$\mathcal{L}_{sup} = \sum_{v \in \mathcal{W}_{id}} g(\mathbf{v}_x \mathbf{W}_{xy}^T, \mathbf{v}_y) + g(\mathbf{v}_y \mathbf{W}_{yx}^T, \mathbf{v}_x)$$

where $\mathbf{v}_x$ and $\mathbf{v}_y$ are the pretrained word embedding of word v in the source and target side respectively, and $g(\cdot, \cdot)$ is the cosine loss described above. Although using identical strings for supervision is very noisy, especially for languages with little overlap in vocabularies, we find that they provide a enough guidance to training to prevent the model from being trapped in poor local optima.

Putting everything together, the overall objective function of DeMa-BWE includes three parts: the density matching loss, the weak orthogonality loss and the weak supervised loss:

$$\mathcal{L} = \mathcal{L}_{xy} + \mathcal{L}_{yx} + \lambda \cdot \mathcal{L}_{bt} + \alpha \cdot \mathcal{L}_{sup} \qquad (11)$$

where λ and α are coefficients that tradeoff between different losses.

4 Retrieval and Refinement

4.1 Retrieval Method

One standard use case for bilingual embeddings is in bilingual lexicon induction, where the embeddings are used to select the most likely translation in the other language given these embeddings. In this case, it is necessary to have a retrieval metric that selects word or words likely to be translations given these embeddings. When performing this retrieval, it has been noted that high-dimensional embedding spaces tend to suffer from the "hubness" problem (Radovanović et al., 2010) where some vectors (known as *hubs*) are nearest neighbors of many other points, which is detrimental to reliably retrieving translations in the bilingual space. To mitigate the hubness problem, we adopt the *Cross-Domain Similarity Local Scaling* (CSLS) metric proposed in (Conneau et al., 2017) that penalizes the similarity score of the hubs. Specifically, given two mapped embeddings $\mathbf{W}_{xy}\mathbf{x}$ denoted $\mathbf{x}'$ and $\mathbf{y}$, CSLS first computes the average cosine similarity of $\mathbf{x}'$ and $\mathbf{y}$ for their k nearest neighbors denoted $r_T(\mathbf{x}')$ and $r_S(\mathbf{y})$ in the other language respectively, then the corrected similarity measure $\text{CSLS}(\cdot, \cdot)$ is defined as:

$$\text{CSLS}(\mathbf{x}', \mathbf{y}) = 2\text{cos}(\mathbf{x}', \mathbf{y}) - r_T(\mathbf{x}') - r_S(\mathbf{y})$$

where $\cos(\cdot, \cdot)$ is the cosine similarity. Following (Conneau et al., 2017), k is set to be 10.

CSLS consistently outperform cosine similarity on nearest neighbor retrieval, however it does not consider the relative frequency of bilingual words which we hypothesize can be useful in disambiguation. As we discussed in Sec. 3.2, our density matching objective considers both the relative frequencies and vector similarities. The conditional density $p(\mathbf{y}|x_i)$ (Eq. (9)) in our density matching objective (Eq. (8)) is a marginalized distribution over all target component words y_j where the density of each component $p(\mathbf{y}, y_j|x_i)$ can be directly used as a similarity score for a pair of words (y_j, x_i) to replace the cosine similarity $\cos(\mathbf{x}', \mathbf{y})$ in CSLS. Let CSLS-D denote this modified CSLS metric, which we compare to standard CSLS in experiments. We find that using CSLS-D for nearest neighbor retrieval outperforms the CSLS metric in most cases on the bilingual dictionary induction task.

4.2 Iterative Procrustes Refinement

Iterative refinement, which learns the new mapping matrix by constructing a bilingual lexicon iteratively, has been shown as an effective method for improving the performance of unsupervised lexicon induction models (Conneau et al., 2017). Given a learned bilingual embedding mapping $\mathbf{W}$, the refinement starts by inferring an initial bilingual dictionary using the retrieval method above on the most frequent words. Let $\mathbf{X}$ and $\mathbf{Y}$ denote the ordered embedding matrices for the inferred dictionary words for source and target languages respectively. Then a new mapping matrix $\mathbf{W}^*$ is induced by solving the Procrustes problem:

$$\mathbf{W}^* = \underset{\mathbf{W} \in O_d(\mathbb{R})}{\text{argmin}} ||\mathbf{W}\mathbf{X} - \mathbf{Y}||_F = \mathbf{U}\mathbf{V}^T$$
$$s.t.\ \mathbf{U}\Sigma\mathbf{V}^T = \text{SVD}(\mathbf{Y}\mathbf{X}^T)$$

The step above can be applied iteratively by inducing a new dictionary with the new mapping $\mathbf{W}$.

DeMa-BWE is able to achieve very competitive performance without further refinement, but for comparison we also report results with the refinement procedure, which brings small improvements in accuracy for most language pairs. Note that for bilingual dictionary induction during refinement, we use CSLS as the retrieval metric across all experiments for fair comparison to the refinement step in previous work.

5 Experiments

5.1 Dataset and Task

We evaluate our approach extensively on the bilingual lexicon induction (BLI) task, which measures the word translation accuracy in comparison to a gold standard. We report results on the widely used MUSE dataset (Conneau et al., 2017), which consists of FastText monolingual embeddings pre-trained on Wikipedia (Bojanowski et al., 2017), and dictionaries for many language pairs divided into train and test sets. We follow the evaluation setups of (Conneau et al., 2017). We evaluate DeMa-BWE by inducing lexicons between English and different languages including related languages, e.g. Spanish; etymologically distant languages, e.g. Japanese; and morphologically rich languages, e.g. Finnish.

5.2 Implementation Details

Embedding Normalization Following (Artetxe et al., 2018b), we pre-process the monolingual embeddings by first applying length normalization, then mean center each dimension, and then length normalize again to ensure that the final embeddings have a unit length. We observe that this normalization method helps stabilize training and accelerate convergence.

Other Experimental Details We held out 1000 translation pairs randomly sampled from the training set in the MUSE dataset as our validation data. We also tried the unsupervised validation criterion proposed in (Conneau et al., 2017) as the model selection method that computes the average cosine similarity over the model induced dictionary pairs and found that this unsupervised criterion can select models that achieve similar performance as the supervised validation criterion. All hyperparameters are tuned on the validation set and include the following: For the number of base words used as Gaussian components in the GMM, we typically choose the most frequent 20,000 words for all language pairs but en-ja for which we use 10,000 which achieves better performance. We use a batch size of 2048 for all languages but en-ja for which we use 1024. We use Adam (Kingma and Ba, 2014) for optimization with default hyper-parameters.

We empirically set the Gaussian variance to be 0.01 for both the source and target languages in en-es, en-de, en-fr, en-ru; in the experiments for morphologically rich languages (Sec. 5.4), we set the variance to be 0.015 for all these languages except for et whose variance is set to be 0.02 while keeping the variance of English to be 0.01. In the experiments for etymologically distant language pairs en-ja and en-zh, we set different variances for the source and target languages in different mapping directions. For details of the variance setting please check the scripts in our code base. We empirically find that for a language pair the variance of the language with relatively more complex morphological properties needed to be set larger than the other language, indicating that the model needs to explore more in the embedding space for the morphologically richer language.

We initialize mapping matrices $\mathbf{W}_{xy}$ and $\mathbf{W}_{yx}$ with a random orthogonal matrix. For the weak orthogonality constraint loss $\mathcal{L}_{bt}$, we set the weight λ to be 0.5 throughout all language pairs. For the weak supervision loss $\mathcal{L}_{sup}$, we set the weight α to be 10 for all languages except for en-zh where we find 5 performs better. We set the temperature τ used in the $\mathrm{softmax}$ function for Gaussian mixture weights to be 2 across all languages.

5.3 Main Results on BLI

In Tab. 1, we compare the performance of DeMa-BME extensively with the best performing unsupervised and supervised methods on the commonly benchmarked language pairs.

Our unsupervised baselines are: (1) **MUSE (U+R)** (Conneau et al., 2017), a GAN-based unsupervised method with refinement. (2) A strong and robust unsupervised self-learning method **SL-unsup** from (Artetxe et al., 2018b). We also run their published code with identical words as the initial dictionary for fair comparison with our approach, denoted **SL-unsup-ID**. (3) **Sinkhorn** (Xu et al., 2018) that minimizes the Sinkhorn distance between the source and target word vectors. (4) An iterative matching method from (Hoshen and

	en-es	es-en	en-de	de-en	en-fr	fr-en	en-ru	ru-en	en-zh	zh-en	en-ja	ja-en
Supervised												
Procrustes (R)	81.4	82.9	73.5	72.4	81.1	82.4	*51.7*	63.7	42.7	36.7	14.2	7.44
MSF-ISF	79.9	82.1	73.0	72.0	80.4	81.4	50.0	65.3	28.0	40.7	-	-
MSF	80.5	83.8	73.5	73.5	80.5	83.1	50.5	67.3	32.3	43.4	-	-
CSLS-Sp	80.7	83.9	75.1	72.1	81.7	83.2	51.1	63.8	-	-	-	-
GeoMM	81.4	*85.5*	74.7	*76.7*	82.1	*84.1*	51.3	*67.6*	*49.1*	*45.3*	-	-
Unsupervised												
MUSE (U+R)	81.7	83.3	74.0	72.2	82.3	81.1	44.0	59.1	32.5	31.4	0.0	4.2
SL-unsup	82.3	84.7	75.1	74.3	82.3	83.6	49.2	65.6	0.0	0.0	2.9	0.2
SL-unsup-ID	82.3	84.6	75.1	74.1	82.2	**83.7**	48.8	**65.7**	37.4	34.2	48.5	33.7
Sinkhorn*	79.5	77.8	69.3	67.0	77.9	75.5	-	-	-	-	-	-
Non-Adv	81.1	82.1	73.7	72.7	81.5	81.3	44.4	55.6	0.0	0.0	0.0	0.0
Non-Adv (R)	82.1	84.1	74.7	73.0	82.3	82.9	47.5	61.8	0.0	0.0	0.0	0.0
WS-Procrustes (R)	82.8	84.1	75.4	73.3	82.6	82.9	43.7	59.1	-	-	-	-
DeMa-BME												
CSLS (w/o R)	82.0	**85.4**	75.3	74.9	82.6	82.4	46.9	62.4	39.6	**40.0**	46.7	32.9
CSLS-D (w/o R)	82.3	85.1	76.3	**75.1**	***83.7***	82.5	48.0	61.7	40.5	37.7	45.3	32.4
CSLS (w/ R)	***82.8***	84.5	75.6	74.1	82.5	83.3	47.3	63.5	41.9	37.7	50.7	35.2
CSLS-D (w/ R)	***82.8***	84.9	***77.2***	74.4	83.1	83.5	**49.2**	63.6	**42.5**	37.9	***52.0***	***35.6***

Table 1: Precision@1 for the MUSE BLI task compared with previous work. All the baseline results employ CSLS as the retrieval metric except for Sinkhorn* which uses cosine similarity. **R** represents refinement. Bold and italic indicate the best unsupervised and overall numbers respectively. ('en' is English, 'es' is Spanish, 'de' is German, 'fr' is French, 'ru' is Russian, 'zh' is traditional Chinese, 'ja' is Japanese.)

Wolf, 2018)): **Non-Adv** and **Non-Adv (R)** with refinement. (5) **WS-Procrustes (R)** using refinement by (Grave et al., 2018). Our supervised methods include: (1) The iterative Procrustes method **Procrustes (R)** (Smith et al., 2017). (2) A multi-step framework **MSF-ISF** (Artetxe et al., 2018a) and its variant **MSF** which uses CSLS for retrieval, whose results are from (Jawanpuria et al., 2018). (3) **CSLS-Sp** by (Joulin et al., 2018) that optimizes the CSLS score, and (4) a geometric approach **GeoMM** by (Jawanpuria et al., 2018). For fair comparisons, all supervised results are trained with the training dictionaries in the MUSE dataset. All baseline methods employ CSLS for retrieval except for the Sinkhorn method.

For DeMa-BME, we present results with and without refinement, and with CSLS and CSLS-D as retrieval methods. From Tab. 1, we can see the overall performance of DeMa-BME is remarkable comparing with other unsupervised methods and is also competitive with strong supervised methods. The results without the iterative refinement **CSLS (w/o R)** are strong on almost all language pairs with particularly strong performance being observed on es-en, en-de and en-fr on which DeMa-BME outperforms or is on par with the best performing methods. Applying refinement to DeMa-BME brings slightly better performance on most language pairs but degrades the performance on some language pairs such as es-en, zh-en for which the DeMa-BME already obtains very good results.

Also, DeMa-BME demonstrates notably better performance on distant language pairs (en-ru, en-ja and en-zh) over other unsupervised methods, which often achieve good performance on etymologically close languages but fail to converge on the distant language pairs. However, when the dictionary is initialized with identical strings for SL-unsup, we obtain decent results on these languages. The strong performance of supervised methods on Russian and Chinese demonstrates that on some language pairs supervised seed lexicons are still necessary.

Finally, when our density-based metric CSLS-D is employed for retrieval, it could achieve further gains in accuracy for most language pairs compared to its counterpart.

5.4 Morphologically Rich Language Results

Søgaard et al. (2018) found that the commonly

	en-et	et-en	en-fi	fi-en	en-el	el-en	en-hu	hu-en	en-pl	pl-en	en-tr	tr-en
MUSE (U+R)	1.7	0.	1 0.1	59.8	39.1	59.0	50.2	0.1	53.9	0.0	45.4	0.0
5k+Procrustes (R)	31.9	45.6	47.3	59.5	44.6	58.5	53.3	**64.8**	58.2	66.9	46.3	59.2
id+Procrustes (R)	29.7	40.6	45.0	59.1	40.7	55.1	52.6	63.7	57.3	66.7	45.4	61.4
id+Procrustes (R)*	31.5	-	28.0	-	42.9	-	46.6	-	52.6	-	39.2	-
CSLS (w/o R)	32.9	45.3	47.5	58.7	43.6	57.8	55.3	64.5	59.9	69.1	50.3	60.8
CSLS-D (w/o R)	35.9	45.2	49.4	58.5	45.0	58.1	57.6	64.7	60.9	68.3	52.8	**61.6**
CSLS (w/ R)	34.4	47.8	48.9	60.2	44.5	61.1	55.1	63.9	59.7	69.0	50.3	60.8
CSLS-D (w/ R)	**37.0**	**47.9**	**50.7**	**60.8**	**46.3**	**61.6**	**58.3**	64.5	**61.5**	**69.1**	**53.4**	61.2

Table 2: BLI Precision (@1) for morphologically complex languages. **id+Procrustes (R)*** is the result reported in (Søgaard et al., 2018). **5k+Procrustes (R)** uses the training dictionary with 5k unique query words. ('et' is Estonian, 'fi' is Finnish, 'el' is Greek, 'hu' is Hungarian, 'pl' is Persian, 'tr' is Turkish.)

benchmarked languages are morphologically poor isolating or exclusively concatenating languages. They select several languages with different morphological traits and complexity then studied the impacts of language similarities on the bilingual lexicon induction.

They show that a simple trick, harvesting the identical word strings in both languages as an initial dictionary and running the iterative Procrustes analysis described in Sec. 4.2, enables more robust and competitive bilingual induction results over the GAN-based unsupervised method with refinement. We denote this method 'id+Procrustes (R)'.

Tab. 2 shows results on the morphologically complex languages used by Søgaard et al. (2018). For each language pair we run experiments in both directions. The baseline methods in Søgaard et al. (2018) include id+Procrustes (R) and the MUSE (U+R). We run id+Procrustes (R) ourselves and obtain different results from them: except for en-et and en-el, we obtain significantly better results on other language pairs. In addition, we add another strong supervised baseline (5k+Procrustes (R)) with the training dictionary in the MUSE dataset and iterative Procrustes refinement. From Tab. 2, we observe that even without refinement, DeMa-BME (CSLS (w/o R)) outperforms both the unsupervised and supervised baselines on even these difficult morphologically rich languages.

5.5 Cross-lingual Word Similarity

We evaluate DeMa-BWE on the cross-lingual word similarity task from SemEval 2017 (Camacho-Collados et al., 2016) and compare with some strong baselines in Xu et al. (2018). In Tab. 5, following the convention in benchmark evaluation for this task, we report the Pearson correlation scores (×100). DeMa-BME achieves the best performance among all the listed unsupervised methods. Compared with the supervised methods, DeMa-BME is also very competitive.

Supervised	de-en	es-en	fa-en	it-en
Xing et al. (2015)	72	71	69	72
Shigeto et al. (2015)	72	*72*	69	72
Artetxe et al. (2016)	*73*	*72*	*70*	*73*
Artetxe et al. (2017)	70	70	67	71
Unsupervised				
Conneau et al. (2017)	71	71	68	71
Xu et al. (2018)	71	71	67	71
DeMa-BME (w/o R)	**72.2**	**72.2**	**68.6**	**72.2**

Table 3: Pearson rank correlation (×100) on cross-lingual word similarity task. Bold indicates the best unsupervised numbers.

	en-fr	fr-en	en-ja	ja-en
w/o L_{xy} & L_{yx}	45.4	73.3	6.3	27.3
w/o L_{bt}	82.3	82.3	46.2	31.1
w/o L_{sup}	0.1	0.1	0.0	0.0
$\pi(y_j\|x_i) := \pi(y_j)$	82.1	82.1	46.0	32.5
Full Model	82.6	82.4	46.7	32.9

Table 4: Ablation study on different components of DeMa-BME.

5.6 Ablation Study

Finally, we perform ablation studies in Tab. 4 to examine the effects of different components of DeMa-BWE. In comparison to the full model, we remove the density matching loss L_{xy} & L_{yx}, the weakly supervised loss L_{sup}, the back-translation loss L_{bt} respectively. First, we observe that without the identical strings as the supervised loss, DeMa-BWE fails to converge as the density matching is difficult given a high-dimensional embedding space to search. Second, when we remove the proposed density matching loss, the model is able to produce reasonable accuracy for fr-en and ja-en, but has undesirable results on en-fr and en-ja, which verifies the necessity of the unsupervised density matching. Third, the back-translation loss is not a crucial component in DeMa-BME; removing it only degrades the model's performance by a small margin. This indicates that orthogonality is not must-have constraint given the model has enough capacity to learn a good transformation.

In addition, we also compare the frequency-matching based Gaussian mixture weights in (9) with the naive target frequency based weights. As shown in the fourth row of Tab. 4, the performance of DeMa-BWE with the naive weights is nominally worse than the model using the frequency-matching based mixture weights.

6 Conclusion

In this work, we propose a density matching based unsupervised method for learning bilingual word embedding mappings. DeMa-BWE performs well in the task of bilingual lexicon induction. In the future work, we will integrate Gaussian embeddings (Vilnis and McCallum, 2015) with our approach.

Acknowledgments

This work is sponsored by Defense Advanced Research Projects Agency Information Innovation Office (I2O), Program: Low Resource Languages for Emergent Incidents (LORELEI), issued by DARPA/I2O under Contract No. HR0011-15-C-0114. The authors would also like to thank Ruochen Xu, Barun Patra, Joel Ruben Antony Moni for their helpful discussions during drafting this paper.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018c. Unsupervised statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *International Conference on Machine Learning*, pages 748–756.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018. Unsupervised alignment of embeddings with Wasserstein Procrustes. *arXiv preprint arXiv:1805.11222*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pages 771–779, Columbus, Ohio. Association for Computational Linguistics.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of EMNLP*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 58–68.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2018. Learning multilingual word embeddings in latent metric space: a geometric approach. *arXiv preprint arXiv:1808.08773*.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10235–10244.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of COLING 2012*, pages 1459–1474.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

George Papamakarios, Iain Murray, and Theo Pavlakou. 2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1530–1538. JMLR. org.

Sebastian Ruder, Ivan Vulic, and Anders Søgaard. 2017. A survey of cross-lingual embedding models. *CoRR, abs/1706.04902*.

Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. 2015. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788. Association for Computational Linguistics.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. *International Conference on Learning Representations*.

Laura Wendlandt, Jonathan K. Kummerfeld, and Rada Mihalcea. 2018. Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2092–2102. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised cross-lingual transfer of word embedding spaces. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

Zhisong Zhang, Wasi Uddin Ahmad, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2018. Near or far, wide range zero-shot cross-lingual dependency parsing. *arXiv preprint arXiv:1811.00570*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Cross-Lingual Alignment of Contextual Word Embeddings, with Applications to Zero-shot Dependency Parsing

Tal Schuster[*,1], **Ori Ram**[*,2], **Regina Barzilay**[1], **Amir Globerson**[2]

[1]Computer Science and Artificial Intelligence Lab, MIT
[2]Tel Aviv University
{tals, regina}@csail.mit.edu, {ori.ram, gamir}@cs.tau.ac.il

Abstract

We introduce a novel method for multilingual transfer that utilizes deep contextual embeddings, pretrained in an unsupervised fashion. While contextual embeddings have been shown to yield richer representations of meaning compared to their static counterparts, aligning them poses a challenge due to their dynamic nature. To this end, we construct context-independent variants of the original monolingual spaces and utilize their mapping to derive an alignment for the context-dependent spaces. This mapping readily supports processing of a target language, improving transfer by context-aware embeddings. Our experimental results demonstrate the effectiveness of this approach for zero-shot and few-shot learning of dependency parsing. Specifically, our method consistently outperforms the previous state-of-the-art on 6 tested languages, yielding an improvement of 6.8 LAS points on average.[1]

1 Introduction

Multilingual embedding spaces have been demonstrated to be a promising means for enabling cross-lingual transfer in many natural language processing tasks (e.g. Ammar et al. (2016); Lample et al. (2018)). Similar to how universal part-of-speech tags enabled parsing transfer across languages (Petrov et al., 2012), multilingual word embeddings further improve transfer capacity by enriching models with lexical information. Since this lexical representation is learned in an unsupervised fashion and thus can leverage large amounts of raw data, it can capture a more nuanced representation of meaning than unlexicalized transfer. Naturally, this enrichment is translated into improved transfer accuracy, especially in low-resource scenarios (Guo et al., 2015).

In this paper, we are moving further along this line and exploring the use of contextual word embeddings for multilingual transfer. By dynamically linking words to their various contexts, these embeddings provide a richer semantic and syntactic representation than traditional context-independent word embeddings (Peters et al., 2018). A straightforward way to utilize this richer representation is to directly apply existing transfer algorithms on the contextual embeddings instead of their static counterparts. In this case, however, each token pair is represented by many different vectors corresponding to its specific context. Even when supervision is available in the form of a dictionary, it is still unclear how to utilize this information for multiple contextual embeddings that correspond to a word translation pair.

In this paper, we propose a simple but effective mechanism for constructing a multilingual space of contextual embeddings. Instead of learning the alignment in the original, complex contextual space, we drive the mapping process using context-independent embedding anchors. We obtain these anchors by factorizing the contextual embedding space into context-independent and context-dependent parts. Operating at the anchor level not only compresses the space, but also enables us to utilize a word-level bilingual dictionary as a source of supervision, if available. Once the anchor-level alignment is learned, it can be readily applied to map the original spaces with contextual embeddings.

Clearly, the value of word embeddings depends on their quality, which is determined by the amount of raw data available for their training (Jiang et al., 2018). We are interested in expanding the above approach to the truly low-resource

---

[*] Equal contribution

[1] Code and models: https://github.com/TalSchuster/CrossLingualELMo.

Proceedings of NAACL-HLT 2019, pages 1599–1613
Minneapolis, Minnesota, June 2 - June 7, 2019.

scenario, where a language not only lacks annotations, but also has limited amounts of raw data. In this case, we can also rely on a data rich language to stabilize monolingual embeddings of the resource-limited language. As above, context-independent anchors are informing this process. Specifically, we introduce an alignment component to the loss function of the language model, pushing the anchors to be closer in the joint space. While this augmentation is performed on the static anchors, the benefit extends to the contextual embeddings space in which we operate.

We evaluate our aligned contextual embeddings on the task of zero-shot cross-lingual dependency parsing. Our model consistently outperforms previous transfer methods, yielding absolute improvement of 6.8 LAS points over the prior state-of-the-art (Ammar et al., 2016). We also perform comprehensive studies of simplified variants of our model. Even without POS tag labeling or a dictionary, our model performs on par with context-independent models that do use such information. Our results also demonstrate the benefits of this approach for few-shot learning, i.e. processing languages with limited data. Specifically, on the Kazakh tree-bank from the recent CoNLL 2018 shared task with only 38 trees for training, the model yields 5 LAS points gain over the top result (Smith et al., 2018a).

2 Related work

Multilingual Embeddings The topic of cross-lingual embedding alignment is an active area of research (Mikolov et al., 2013; Xing et al., 2015; Dinu and Baroni, 2014; Lazaridou et al., 2015; Zhang et al., 2017). Our work most closely relates to MUSE (Conneau et al., 2018a), which constructs a multilingual space by aligning monolingual embedding spaces. When a bilingual dictionary is provided, their approach is similar to those of (Smith et al., 2017; Artetxe et al., 2017). MUSE extends these methods to the unsupervised case by constructing a synthetic dictionary. The resulting alignment achieves strong performance in a range of NLP tasks, from sequence labeling (Lin et al., 2018) to natural language inference (Conneau et al., 2018b) and machine translation (Lample et al., 2018; Qi et al., 2018). Recent work further improves the performance on both the supervised (Joulin et al., 2018) and unsupervised (Grave et al., 2018b; Alvarez-Melis and Jaakkola, 2018; Hoshen and Wolf, 2018) settings for context-independent embeddings.

While MUSE operates over token based embeddings, we are interested in aligning contextual embeddings, which have shown their benefits in several monolingual applications (Peters et al., 2018; McCann et al., 2017; Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). However, this expansion introduces new challenges which we address in this paper.

In a concurrent study, Aldarmaki and Diab (2019) introduced an alignment that is based only on word pairs in the same context, using parallel sentences. Our method achieves better word translations without relying on such supervision.

Our work also relates to prior approaches that utilize bilingual dictionaries to improve embeddings that were trained on small datasets. For instance, Xiao and Guo (2014) represent word pairs as a mutual vector, while Adams et al. (2017) jointly train cross-lingual word embeddings by replacing the predicted word with its translation. To utilize a dictionary in the contextualized case, we include a soft constraint that pushes those translations to be similar in their context-independent representation. A similar style of regularization was shown to be effective for cross-domain transfer of word embeddings (Yang et al., 2017).

Multilingual Parsing In early work on multilingual parsing, transfer was commonly implemented using delexicalized representation such as part-of-speech tags (McDonald et al., 2011; Petrov et al., 2012; Naseem et al., 2012; Tiedemann, 2015).

Another approach for cross-lingual parsing includes annotation projection and treebank translation (Xiao and Guo, 2015; Wang and Eisner, 2016; Tiedemann, 2017), which mostly require some source of supervision.

Advancements in multilingual word representations opened a possibility of lexicalized transfer. Some of these approaches start by aligning monolingual embedding spaces (Zhang and Barzilay, 2015; Guo et al., 2015, 2016; Ammar et al., 2016), and using resulting word embeddings as word representations instead of universal tags. Other approaches are learning customized multilingual syntactic embeddings bootstrapping from universal POS tags (Duong et al., 2015). While some models also learn a language embedding (Ammar et al., 2016; de Lhoneux et al., 2018), it is unfeasible in a zero-shot scenario.

$D_{cos}(\bar{\boldsymbol{e}}_i, \bar{\boldsymbol{e}}_j)$	$D_{cos}(\bar{\boldsymbol{e}}_i, \boldsymbol{e}_{i,c})$	
	ALL WORDS	HOMONYMS
0.85 (±0.09)	0.18 (±0.04)	0.21 (±0.04)

Table 1: Average cosine distances between pairs of embedding anchors (left column) and between contextualized embeddings of words to their corresponding anchor. The right column includes these distances only for homonyms, whereas the center column is averaged across all words. Only alphabetic words with at least 100 occurrences were included.

In all of the above cases, token-level embeddings are used. Inspired by strong results of using contextualized embeddings in monolingual parsing (Che et al., 2018; Wang et al., 2018; Clark et al., 2018), we aim to utilize them in the multilingual transfer case. Our results demonstrate that richer representation of lexical space does lead to significant performance gains.

3 Aligning Contextual Word Embeddings

In this section we describe several approaches for aligning context-dependent embeddings from a source language s to a target language t. We address multiple scenarios, where different amounts of supervision and data are present. Our approach is motivated by interesting properties of context-dependent embeddings, which we discuss later.

We begin with some notations:

- **Context Dependent Embeddings:** Given a context c and a token i, we denote the embedding of i in the context c by $\boldsymbol{e}_{i,c}$. We use $\boldsymbol{e}_{i,\cdot}$ to denote the point cloud of all contextual embeddings for token i.

- **Embedding Anchor:** Given a token i we denote the anchor of its context dependent embeddings by $\bar{\boldsymbol{e}}_i$, where:

$$\bar{\boldsymbol{e}}_i = \mathbb{E}_c\left[\boldsymbol{e}_{i,c}\right] \ . \tag{1}$$

  In practice, we calculate the average over a subset of the available unlabeled data.

- **Shift From Mean:** For any embedding $\boldsymbol{e}_{i,c}$ we can therefore define the shift $\hat{\boldsymbol{e}}_{i,c}$ from the average via:

$$\boldsymbol{e}_{i,c} = \bar{\boldsymbol{e}}_i + \hat{\boldsymbol{e}}_{i,c} \ . \tag{2}$$

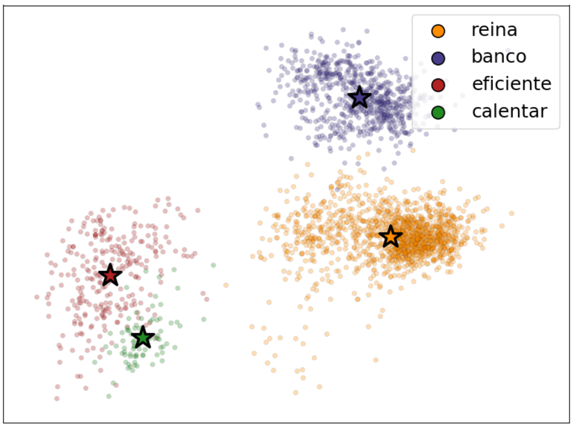


Figure 1: A two dimensional PCA showing examples of contextual representations for four Spanish words. Their corresponding anchors are presented as a star in the same color. (best viewed in color)

- **Embedding Alignment:** Given an embedding $\boldsymbol{e}^{s}_{i,c}$ in s, we want to generate an embedding $\boldsymbol{e}^{s\rightarrow t}_{i,c}$ in the target language space, using a linear mapping $W^{s\rightarrow t}$. Formally, our alignment is always of the following form:

$$\boldsymbol{e}^{s\rightarrow t}_{i,c} = W^{s\rightarrow t}\boldsymbol{e}^{s}_{i,c} \ . \tag{3}$$

3.1 The Geometry of Context-Dependent Embeddings

A given token i can generate multiple vectors $\boldsymbol{e}_{i,c}$, each corresponding to a different context c. A key question is how the point cloud $\boldsymbol{e}_{i,\cdot}$ is distributed. In what follows we explore this structure, and reach several conclusions that will motivate our alignment approach. The following experiments are performed on ELMo (Peters et al., 2018).

Point Clouds are Well Separated A cloud $\boldsymbol{e}_{i,\cdot}$ corresponds to occurrences of the word i in different contexts. Intuitively, we would expect its points to be closer to each other than to points from $\boldsymbol{e}_{j,\cdot}$ for a different word j. Indeed, when measuring similarity between points $\boldsymbol{e}_{i,c}$ and their anchor $\bar{\boldsymbol{e}}_i$, we find that these are much more similar than anchors of different words $\bar{\boldsymbol{e}}_i$ and $\bar{\boldsymbol{e}}_j$ (see Table 1). This observation supports our hypothesis that anchor-driven alignment can guide the construction of the alignment for the contextual space. A visualized example of the contextualized representations of four words is given in Figure 1, demonstrating the appropriateness of their anchors. Still, as previous studies have shown, and

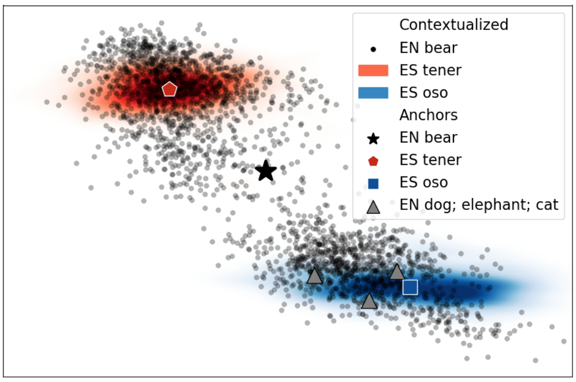


Figure 2: Contextual embeddings for the English word "bear" and its two possible translations in Spanish — "oso" (animal) in blue and "tener" (to have) in red. The figure shows a two dimensional PCA for the aligned space of the two languages. The symbols are the anchors, the clouds represent the distribution of the contextualized Spanish words, and the black dots are for contextualized embeddings of "bear". The gray colored triangles show the anchors of the English words "dog", "elephant", "cat", from left to right respectively.

as our results point, the context component is very useful for downstream tasks.

Homonym Point Clouds are Multi-Modal When a word i has multiple distinct senses, we might expect the embeddings for i to reflect this by separating into multiple distinct clouds, one for each meaning. Figure 2 demonstrates that this indeed happens for the English word "bear". Furthermore, it can be seen that after alignment (Section 3.3) with Spanish, the distinct point clouds are aligned with their corresponding distinct words in Spanish. See App. D for another example.

We examined the shift from mean for a list of 250 English homonyms from Wikipedia.[2] As Table 1 shows, the shift of these words is indeed slightly higher than it is for other words. However, they still remain relatively close to their per-token anchor. Therefore, these anchors can still serve as a good approximation for learning alignments.

3.2 Context-Independent Alignment

We begin by briefly reviewing previous approaches for aligning context-independent embeddings, as they are generalized in this work to the contextual case. We denote the embedding of a word i by $\boldsymbol{e}_i$. At first, assume we are given word pairs $\{(\boldsymbol{e}_i^s, \boldsymbol{e}_i^t)\}$ from a source language s and a target language t, and we look for a mapping between those. Mikolov et al. (2013) proposed to learn a linear transformation whereby $\boldsymbol{e}_i^t$ is approximated via $W\boldsymbol{e}_i^s$, for a learned matrix W. We focus on methods that follow this linear alignment. The alignment matrix is found by solving:

$$W^{s\to t} = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| W\boldsymbol{e}_i^s - \boldsymbol{e}_i^t \right\|^2 , \qquad (4)$$

where $O_d(\mathbb{R})$ is the space of orthogonal matrices. This constraint was proposed by Xing et al. (2015) in order to preserve inter-lingual relations. Under this constraint, Eq. 4 is an instance of the orthogonal Procrustes problem, which has a closed-form solution $W^{s\to t} = UV^T$. The columns of U and V are the left and right singular vectors of the multiplication of the source and (transposed) target embedding matrices.

For the *unsupervised* case (i.e. when a dictionary is absent), Conneau et al. (2018a) (MUSE) suggested to learn the alignment via adversarial training, such that a discriminator is trained to distinguish between target and aligned source embeddings. Thereafter, a *refinement* procedure is applied iteratively as follows. First, a dictionary is built dynamically using the current alignment such that only words with high confidence are considered. Using the dictionary, the alignment matrix is re-calculated as in the supervised case.

3.3 Context-Dependent Alignment

We next turn our attention to the main task of this paper, which is aligning context-dependent embeddings. We now describe our generalization of the methods described in Section 3.2 for this case. The first two methods are based only on anchors while the third one uses the contextual vectors themselves. Altogether, we suggest three alignment procedures, one aimed for the supervised and two for the unsupervised cases.

Supervised Anchored Alignment As a first step, we are assuming access to a dictionary for the source and target domains. For each source word i denote by $D(i)$ the corresponding word in the target language.[3]

In the context-dependent case, Eq. 4 is no longer well-defined, as there are many corresponding vectors to both the source and the target

---

[2] `https://en.wikipedia.org/wiki/List_of_true_homonyms`

[3] In practice, we may have multiple target words for a single source word, and the extension is straight-forward.

words. However, this challenge can be addressed by aligning the vectors $\bar{e}_i$ for which we do have one per word. This is motivated by our observations in Section 3.1 that context-dependent embeddings are well clustered around their centers.

Thus, in the case where a dictionary is available, we solve Eq. 4 with token anchors as inputs.

We emphasize that by constraining $W^{s\rightarrow t}$ to be orthogonal, we also preserve relations between $\hat{e}_{i,c}$ and $\hat{e}_{i,c'}$ that represent the contextual information.

Unsupervised Anchored Alignment In this setting, no dictionary is present. As in the supervised case, we can naturally extend a context-independent alignment procedure to the contextual space by leveraging the anchor space $\bar{e}_i$. This can be done using the adversarial MUSE framework proposed by Conneau et al. (2018a) and described at the end of Section 3.2.

Unsupervised Context-based Alignment Alternatively, the alignment could be learned directly on the contextual space. To this end, we follow again the adversarial algorithm of MUSE, but for each word we use multiple embeddings induced by different contexts, rather than the word anchor.

This context-based alignment presents opportunities but also introduces certain challenges. On the one hand, it allows to directly handle homonyms during the training process. However, empirically we found that training in this setting is less stable than unsupervised anchored alignments.

Refinement As a final step, for both of the unsupervised methods, we perform the refinement procedure that is incorporated in MUSE (end of Section 3.2). In order to synthesize a dictionary, we use distance in the anchor space.

3.4 Learning Anchored Language Models

Thus far we assumed that embeddings for both source and target languages are pretrained separately. Afterwards, the source is mapped to the target in a second step via a learned mapping. However, this approach may not work well when raw data for the source languages is scarce, resulting in deficient embeddings. In what follows, we show how to address this problem when a dictionary is available. We focus on embeddings that are learned using a language model objective but this can be easily generalized to other objectives as well.

Our key idea is to constrain the embeddings across languages such that word translations will be close to each other in the embedding space. This can serve as a regularizer for the resource-limited language model. In this case, the anchors are the model representations prior to its context-aware components (e.g., the inputs to ELMo's LSTM).

Denote the anchor for word i in language s by $\boldsymbol{v}_i^s$. Now, assume we have trained a model for the target language and similarly have embeddings $\boldsymbol{v}_i^t$. We propose to train the source model with an added regularization term as follows:

$$\lambda_{\text{anchor}} \cdot \sum_i \|\boldsymbol{v}_i^s - \boldsymbol{v}_{D(i)}^t\|_2^2 \,, \qquad (5)$$

where λ_{anchor} is a hyperparamter. This regularization has two positive effects. First, it reduces overfitting by reducing the effective number of parameters the model fits (e.g., if the regularizer has large coefficient, these parameters are essentially fixed). Second, it provides a certain level of alignment between the source and target language since they are encouraged to use similar anchors.

4 Multilingual Dependency Parsing

Now that we presented our method for aligning contextual embeddings, we turn to evaluate it on the task of cross-lingual dependency parsing. We first describe our baseline model, and then show how our alignment can easily be incorporated into this architecture to obtain a multilingual parser.

Baseline Parser Most previous cross-lingual dependency parsing models used transition-based models (Ammar et al., 2016; Guo et al., 2016). We follow Che et al. (2018); Wang et al. (2018); Clark et al. (2018) and use a first-order graph-based model. Specifically, we adopt the neural edge-scoring architecture from Dozat and Manning (2017); Dozat et al. (2017), which is based on Kiperwasser and Goldberg (2016). We now briefly review this architecture. Given a sentence s, let $\boldsymbol{e}_i$ and $\mathbf{p}_i$ be its word and POS-tag embeddings. These are concatenated and fed into a Bi-LSTM to produce token-level contextual representations $\mathbf{r}_i$. Four Multi-Layer Perceptrons are applied on these vectors, resulting in new representations $\boldsymbol{h}_i^{arc-dep}$, $\boldsymbol{h}_i^{arc-head}$, $\boldsymbol{h}_i^{rel-dep}$ and $\boldsymbol{h}_i^{rel-head}$

for each word i. Arc scores are then obtained by:

$$s_{ij}^{arc} = \left(\boldsymbol{h}_i^{arc-head}\right)^T \left(U^{arc}\boldsymbol{h}_j^{arc-dep} + \boldsymbol{b}^{arc}\right) . \tag{6}$$

Additionally, the score for predicting the dependency label r for an edge (i, j) is defined as

$$\begin{aligned} s_{(i,j),r}^{rel} = &\left(\boldsymbol{h}_i^{rel-head}\right)^T U_r^{rel}\boldsymbol{h}_j^{rel-dep} + \\ &\left(\boldsymbol{u}_r^{rel-head}\right)^T \boldsymbol{h}_i^{rel-head} + \\ &\left(\boldsymbol{u}_r^{rel-dep}\right)^T \boldsymbol{h}_j^{rel-dep} + b_r . \end{aligned} \tag{7}$$

At test time, MST is calculated to ensure valid outputs.

Multilingual Parsing with Alignment We now extend this model, in order to effectively use it for transfer learning. First, we include contextualized word embeddings by replacing the static embeddings with a pre-trained ELMo (Peters et al., 2018) model (instead of $\boldsymbol{e}_i$). Second, we share all model parameters across languages and use the contextual word embeddings after they are aligned to a joint space J. Formally, if s is a sentence of language ℓ, contextual word embeddings are obtained via:

$$\boldsymbol{e}_{i,s}^{\ell \to J} = W^{\ell \to J}\boldsymbol{e}_{i,s} \ , \tag{8}$$

where $W^{\ell \to J}$ is the alignment matrix from language ℓ to the joint space.[4] This alignment is learned apriori and kept fixed during parser training. This setup is applicable for both single and multiple training languages. For the tested language, training data could be available, sometimes limited (few-shot), or absent (zero-shot). The alignment methods are described in detail in Section 3.

In their paper, Peters et al. (2018) suggest to output a linear combination over the representations of each layer of ELMo, learning these weights jointly with a downstream task. Our alignment is learned separately for each layer. Therefore, we keep the weights of the combination fixed during the training to ensure that the parser's inputs are from the joint cross-lingual space. Alternatively, one can share the weights of the combination between the languages and learn them.

All the above modifications are at the word embedding level, making them applicable to any other NLP model that uses word embeddings.

5 Experimental Setup

Contextual Embeddings We use the ELMo model (Peters et al., 2018) with its default parameters to generate embeddings of dimension 1024 for all languages. For each language, training data comprises Wikipedia dumps[5] that were tokenized using UDpipe (Straka and Straková, 2017). We randomly shuffle the sentences and, following the setting of ELMO, use 95% of them for training and 5% for evaluation.

Alignment We utilize the MUSE framework[6] (Conneau et al., 2018a) and the dictionary tables provided by them. The $\bar{e}_i$ (anchor) vectors for the alignment are generated by computing the average of representations on the evaluation set (except for the limited unlabeled data case). To evaluate our alignment, we use the anchors to produce word translations. For all experiments we use the 50k most common words in each language.

Dependency Parsing We used the biaffine parser implemented in AllenNLP (Gardner et al., 2018), refactored to handle our modifications as described in Section 4.[7] The parser is trained on trees from a single or multiple languages, as described in each setting (Section 6). For the multiple case, we randomly alternate between the available languages, i.e. at each iteration we randomly choose one language and sample a corresponding batch. Dropout (Srivastava et al., 2014) is applied on ELMo representations, Bi-LSTM representations and outputs of MLP layers. We also apply early stopping, where validation accuracy is measured as average LAS score on the development set across all training languages. The parser hyperparameters are the same as Dozat et al. (2017) except we reduce the POS tag embedding size from 100 to 50 and increase the head/dependent MLP dimension from 400 to 500. All hyperparameter values used are listed in App. C.

From experiments on the English tree-bank, we found that using the outputs of the first LSTM layer is as good as learning a combination.[8] This

---

[4] We use the space of the training language as our joint space and align the tested language to it. In the multi-source scenario, we align all embeddings to English.

[5] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989`

[6] `https://github.com/facebookresearch/MUSE/`

[7] `https://github.com/TalSchuster/allennlp-MultiLang`

[8] This was concurrently justified by Liu et al. (2019), showing that the first layer alone can perform better than a mixture.

ALIGNMENT METHOD	DE	ES	FR	IT	PT	SV	AVERAGE
SUPERVISED ANCHORED	78	85	86	82	74	68	79
UNSUPERVISED ANCHORED	63	61	70	58	35	22	52
+ REFINE	72	74	**81**	**77**	53	**33**	**65**
UNSUPERVISED CONTEXT-BASED	57	68	59	57	53	*	49
+ REFINE	**73**	**82**	77	73	**66**	*	62

Table 2: Word translation to English precision @5 using CSLS (Conneau et al., 2018a) with a dictionary (supervised) and without (unsupervised) for German (DE), Spanish (ES), French (FR), Italian (IT), Portuguese (PT) and Swedish (SV). Each of the unsupervised results is followed by a line with the results post the anchor-based refinement steps. * stands for 'Failed to converge'.

MODEL	DE	ES	FR	IT	PT	SV	AVERAGE
Zhang and Barzilay (2015)	54.1	68.3	68.8	69.4	72.5	62.5	65.9
Guo et al. (2016)	55.9	73.1	71.0	71.2	78.6	69.5	69.9
Ammar et al. (2016)	57.1	74.6	73.9	72.5	77.0	68.1	70.5
ALIGNED FASTTEXT	61.5	78.2	76.9	76.5	**83.0**	70.1	74.4
ALIGNED $\bar{e}$	58.0	76.7	76.7	76.1	79.2	71.9	73.1
OURS	**65.2**	**80.0**	**80.8**	**79.8**	82.7	**75.4**	**77.3**
OURS, NO DICTIONARY	64.1	77.8	79.8	79.7	79.1	69.6	75.0
OURS, NO POS	61.4	77.5	77.0	77.6	73.9	71.0	73.1
OURS, NO DICTIONARY, NO POS	61.7	76.6	76.3	77.1	69.1	54.2	69.2

Table 3: Zero-shot cross lingual LAS scores compared to previous methods, for German (DE), Spanish (ES), French (FR), Italian (IT), Portuguese (PT) and Swedish (SV). Aligned FASTTEXT and $\bar{e}$ context-independent embeddings are also presented as baselines. The bottom three rows are models that don't use POS tags at all and/or use an unsupervised anchored alignment. Corresponding UAS results are provided in App. B.

agrees with Belinkov et al. (2017), showing that lower layers capture more syntactic information. Therefore, we fix the weights over ELMo layers to $[0, 1, 0]$, i.e. using only representations from the first LSTM layer.

Evaluation Scenarios for Dependency Parsing For a fair comparison, we use the same setting as used by previous models for each scenario. Our main model (which we refer to as OURS) is using a SUPERVISED ANCHORED ALIGNMENT (Section 3.3) to align the multilingual pretrained ELMo embeddings which are used by the parser. We compare against several variants of our model:

- ALIGNED FASTTEXT: instead of ELMo, we use FASTTEXT pretrained embeddings (Grave et al., 2018a), aligned to English using MUSE.
- ALIGNED $\bar{e}$: instead of contextualized embeddings, we use the anchors themselves as fixed embeddings, aligned to English.
- NO DICTIONARY: we assume the absence of a dictionary and use UNSUPERVISED ANCHORED ALIGNMENT.
- NO POS: no use of part of speech tags.

6 Results

Alignment As mentioned above, we use outputs of the first LSTM layer of ELMo in our parsing experiments. Therefore, we present the alignment accuracy for those in Table 2, summarizing the precision@5 word-translation from 6 languages to English. Results for the other layers are presented in App. A. As expected, supervised alignments outperform unsupervised ones by a large margin. Between the two unsupervised methods, the context-based alignment achieved significantly better results for Spanish and Portuguese but failed to converge for Swedish. In both cases, the value of anchors in the REFINE step is clear, substantially improving the precision for all languages.

# SENTENCES	LANGUAGE MODEL	UAS / LAS		PERPLEXITY		ALIGN
		DEV	TEST	TRAIN	DEV	
28M	ELMO	72.3 / 62.8	72.5 / 61.3	22	44	85
10K	ELMO	52.9 / 38.3	50.1 / 33.1	4	4060	4
	ANCHORED ELMO	**59.2 / 47.3**	**57.2 / 42.2**	92	600	12

Table 4: Zero-shot, single-source results for the Spanish limited unlabeled data experiments. The parsing results are UAS/LAS scores, the perplexity is of the ELMo model, and the alignment scores are precision@5 on the held-out set, based on CSLS. All embeddings were aligned to English using supervised anchored alignment.

MODEL	LAS-F1
Rosa and Mareček (2018)	26.31
Smith et al. (2018a)	31.93
ALIGNED FASTTEXT	26.77
OURS	**36.98**

Table 5: Results for the Kazakh dataset from CoNLL 2018 Shared Task on Multilingual Parsing, compared to the two leading models w.r.t. this treebank.

Zero-Shot Parsing, Multiple Source Languages Table 3 summarizes the results for our zero-shot, multi-source experiments on six languages from Google universal dependency treebank version 2.0.[9] For each tested language, the parser was trained on all treebanks in the five other languages and English. We align each of the six languages to English. We compare our model to the performance of previous methods in the same setting (referred to as $L^t \cap L^s = \emptyset$ in Ammar et al. (2016)). The results show that our multilingual parser outperforms all previous parsers with a large margin of 6.8 LAS points. Even with an unsupervised alignment, our model consistently improves over previous models.

To make a fair comparison to previous models, we also use gold POS tags as inputs to our parser. However, for low-resource languages, we might not have access to such labels. Even without the use of POS tags at all, in five out of six languages the score is still higher than previous methods that do consider such annotations. An exception is the Portuguese language where it leads to a drop of 8.8 LAS points. While in the single language setting this good performance can be explained by the knowledge captured in the character level, contextual embeddings (Smith et al., 2018b; Belinkov et al., 2017), the results suggest that this knowledge transfers across languages.

In order to assess the value of contextual embeddings, we also evaluate our model using non-contextual embeddings produced by FASTTEXT (Bojanowski et al., 2017). While these improve over previous works, our context-aware model outperforms them for all six languages in UAS score and for 5 out of 6 languages in LAS score, obtaining an average higher by 3 points. To further examine the impact of introducing context, we run our model with precomputed anchors ($\bar{e}$). Unlike FASTTEXT embeddings of size 300, these anchors share the same dimension with contextual embeddings but lack the contextual information. Indeed, the context-aware model is consistently better.

Few-Shot Parsing, Small Treebanks In this scenario, we assume a very small tree-bank for the tested language and no POS tags. We use the Kazakh tree-bank from CoNLL 2018 shared task (Zeman et al., 2018). The training set consists of only 38 trees and no development set is provided. Segmentation and tokenization are applied using UDPipe. Similar to Rosa and Mareček (2018); Smith et al. (2018a), we utilize the available training data in Turkish as it is a related language. To align contextual embeddings, we use a dictionary generated and provided by Rosa and Mareček (2018) and compute an alignment from Kazakh to Turkish. The dictionary was obtained using FastAlign (Dyer et al., 2013) on the OpenSubtitles2018 (Lison and Tiedemann, 2016) parallel sentences dataset from OPUS (Tiedemann, 2012).[10]

Table 5 summarizes the results, showing that our algorithm outperforms the best model from the shared task by 5.05 LAS points and improves by

---

[9] `https://github.com/ryanmcd/uni-dep-tb/`

[10] `https://github.com/CoNLL-UD-2018/CUNI-x-ling`

over 10 points over a FASTTEXT baseline.

Zero-Shot Parsing, Limited Unlabeled Data To evaluate our anchored language model (Section 3.4), we simulate a low resource scenario by extracting only 10k random sentences out of the Spanish unlabeled data. We also extract 50k sentences for LM evaluation but perform all computations, such as anchor extraction, on the 10k training data. For a dictionary, we used the 5k training table from Conneau et al. (2018a).[11] Another table of size 1,500 was used to evaluate the alignment. In this scenario, we assume a single training language (English) and no usage of POS tags nor any labeled data for the tested language.

Table 4 shows the results. Reducing the amount of unlabeled data drastically decreases the precision by around 20 points. The regularization introduced in our anchored LM significantly improves the validation perplexity, leading to a gain of 7 UAS points and 9 LAS points.

7 Conclusion

We introduce a novel method for multilingual transfer that utilizes deep contextual embeddings of different languages, pretrained in an unsupervised fashion. At the core of our methods, we suggest to use anchors for tokens, reducing this problem to context-independent alignment. Our methods are compatible both for cases where a dictionary is present and absent, as well as for low-resource languages. The acquired alignment can be used to improve cross-lingual transfer learning, gaining from the contextual nature of the embeddings. We show that these methods lead to good word translation results, and improve significantly upon state-of-the-art zero-shot and few-shot cross-lingual dependency parsing models.

In addition, our analysis reveals interesting properties of the context-aware embeddings generated by the ELMo model. Those findings are another step towards understanding the nature of contextual word embeddings.

As our method is in its core task-independent, we conjecture that it can generalize to other tasks as well.

Acknowledgements

We thank the MIT NLP group and the reviewers for their helpful discussion and comments.

The first and third authors were supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

This work was also supported in part by the US-Israel Binational Science Foundation (BSF, Grant No. 2012330), and by the Yandex Initiative in Machine Learning.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 937–947. Association for Computational Linguistics.

Hanan Aldarmaki and Mona Diab. 2019. Context-aware crosslingual mapping. *arXiv preprint arXiv:1903.03243*.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890. Association for Computational Linguistics.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10. Asian Federation of Natural Language Processing.

---

[11] We filtered words with multiple translations to the most common one by Google Translate.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *International Conference on Learning Representations*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Georgiana Dinu and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *CoRR*, abs/1412.6568.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *International Conference on Learning Representations*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30. Association for Computational Linguistics.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018a. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2018b. Unsupervised alignment of embeddings with Wasserstein Procrustes. *CoRR*, abs/1805.11222.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2734–2740.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Chao Jiang, Hsiang-Fu Yu, Cho-Jui Hsieh, and Kai-Wei Chang. 2018. Learning word embeddings for low-resource languages by PU learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1024–1034. Association for Computational Linguistics.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280. Association for Computational Linguistics.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997. Association for Computational Linguistics.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809. Association for Computational Linguistics.

Pierre Lison and Jrg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Rudolf Rosa and David Mareček. 2018. CUNI x-ling: Parsing under-resourced languages in CoNLL 2018 UD Shared Task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 187–196. Association for Computational Linguistics.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123. Association for Computational Linguistics.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted POS labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

Jörg Tiedemann. 2017. Cross-lingual dependency parsing for closely related languages - helsinki's submission to vardial 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 131–136. Association for Computational Linguistics.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Wenhui Wang, Baobao Chang, and Mairgup Mansur. 2018. Improved dependency parsing using implicit word connections learned from unlabeled data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2857–2863. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2015. Annotation projection-based representation learning for cross-lingual dependency parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 73–82. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011. Association for Computational Linguistics.

Wei Yang, Wei Lu, and Vincent Zheng. 2017. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2898–2904.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970. Association for Computational Linguistics.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867. Association for Computational Linguistics.

A Alignment Results for All Layers

Table 6 manifests word-to-word translation results when supervised alignment is performed over different layer outputs from ELMo's LSTM. Even though layer zero produces context independent representations, the anchors computed over the contextual representations achieved higher precision. We conjecture that this is due to the language model objective being applied to the output of the second layer. Hence, unlike token-based embeddings such as FASTTEXT that optimize them directly, the context-independent representations of ELMo are optimized to produce a good base for the contextual embeddings that are computed on top of them.

B Additional Parsing Results

In Table 7 we provide complementary results to those in zero-shot closs-lingual parsing.

LAYER	DE	ES	FR	IT	PT	SV	AVERAGE
0	54 / 71	65 / 80	66 / 80	61 / 78	55 / 71	41 / 61	57 / 74
1	**62 / 78**	**73 / 85**	**74 / 86**	**69 / 82**	**61 / 74**	**49 / 68**	**65 / 79**
2	59 / 75	68 / 82	70 / 83	66 / 79	56 / 72	48 / 67	61 / 76

Table 6: Per ELMo layer word translation to English precision @1 / @5 using CSLS (Conneau et al., 2018a) with a dictionary (supervised) for German (DE), Spanish (ES), French (FR), Italian (IT), Portuguese (PT) and Swedish (SV). Layer 0 representations are the result of the character-level word embeddings (which are context independent). Layer 1 and 2 alignments are based on anchors from the first and second LSTM layer output respectively.

MODEL	DE	ES	FR	IT	PT	SV	AVERAGE
Zhang and Barzilay (2015)	62.5	78.0	78.9	79.3	78.6	75.0	75.4
Guo et al. (2016)	65.0	79.0	77.7	78.5	81.9	78.3	76.7
ALIGNED FASTTEXT	69.2	83.4	84.6	84.3	86.0	80.6	81.4
ALIGNED $\bar{e}$	65.1	82.8	83.9	83.6	83.4	82.0	80.1
OURS	**73.7**	**85.5**	**87.8**	**87.0**	**86.6**	**84.6**	**84.2**
OURS, NO DICTIONARY	73.2	84.3	87.0	86.8	84.5	80.4	82.7
OURS, NO POS	69.7	84.8	85.3	85.3	79.7	81.7	81.1
OURS, NO DICTIONARY, NO POS	72.2	84.7	84.9	85.0	78.1	67.9	78.8

Table 7: Zero-shot cross lingual results compared to previous methods, measured in UAS. Aligned fastText and $\bar{e}$ context-independent models are also presented as baselines. The bottom three rows are models that don't use POS tags at all and/or use an unsupervised anchored alignment.
Note that Ammar et al. (2016) did not publish UAS results.

C Hyperparameters

We now detail the hyperparameters used throughout our experiments. All alignment experiments were performed using the default hyperparameters of the MUSE framework (see their github repository). Table 8 depicts the values used in multilingual parsing experiments.

HYPERPARAMETER	VALUE
BATCH SIZE (# SENTENCES)	32
INSTANCES PER EPOCH	32,000
EPOCHS (MAX.)	40
PATIENCE (EARLY STOPPING)	10
ENCODER TYPE	BI-LSTM
POS TAG EMBEDDING DIM.	50
LSTM, HIDDEN SIZE	200
LSTM, # LAYERS	3
DROPOUT RATE	0.33
ARC REPRESENTATION DIM.	500
TAG REPRESENTATION DIM.	100
ADAM	(DEFAULT)

Table 8: Hyper-parameters used in parsing experiments, shared across different settings.

D Additional Alignment Example

We provide an additional example of a homonym. Figure 3 shows the contextual embeddings of the word "bank" in English and the words "banco" (a financial establishment) and "orilla" (shore) in Spanish. In this case, unlike the "bear" example (Figure 2), the embeddings do not form two obvious clusters in the reduced two dimensional space. A possible explanation is that here the two meanings have the same POS tag (Noun). Even so, as shown in Table 9, the alignment succeeds to place the embeddings of words from each context close to the matching translation.

The nearest-neighbors for the "bear" example are presented in Table 10.

k	“BANCO” ANCHOR
1	Unlike in primary succession , the species that dominate secondary succession , are usually present from the start of the process , often in the soil seed **bank** .
2	Canto XLV is a litany against Usura or usury , which Pound later defined as a charge on credit regardless of potential or actual production and the creation of wealth ex nihilo by a **bank** to the benefit of its shareholders .
3	This prompted some investigation , led by Sir Benjamin Hall , which quickly turned up the fact that O’ Connor was registered as the owner of all the estates , and of the associated **bank** .
4	The commercial NaS battery **bank** offers : (Japanese) .
5	Both team leaders are given a mystery word , which along with their team - mates use gigantic foam blocks and place them on the clue **bank** (similar to Boggle) with only giving a clue to the word ...

k	“ORILLA” ANCHOR
1	The combined Protestant forces , now numbering 25,000 strong , positioned themselves on the western **bank** of the Rhine River .
2	The Romans had a small advance guard of auxiliaries and cavalry on the opposite **bank** of the river .
3	Between Vaugirard and the river Seine he had a considerable force of cavalry , the front of which was flanked by a battery advantageously posted near Auteuil on the right **bank** of the river .
4	Mallus therefore stood on the eastern **bank** of the river .
5	The Argentine squadron spent the night of February 7 anchored between Juncal Island and the west **bank** of the river .
6	After marching north from Tewkesbury , Sir William Waller tried to contain the cavalry forces of Maurice on the western **bank** of the Severn , cutting this substantial force off from the rest of the Royalist army .

Table 9: Nearest-neighbors (after alignment) of the Spanish anchors “banco” (a financial establishment) and “orilla” (shore) from the contextual embeddings of the word “bank” in English. The full sentence is presented for context.

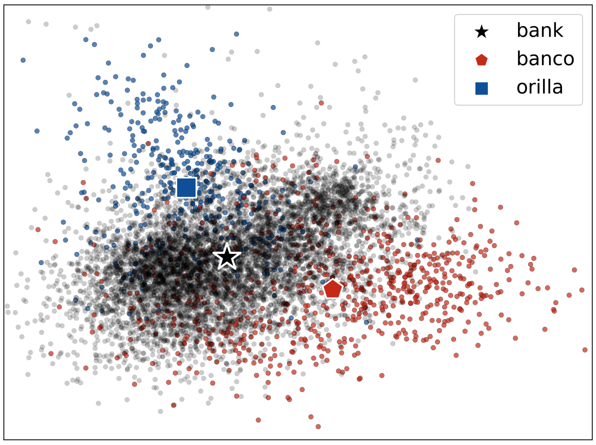


Figure 3: Contextual embeddings for the English word “bank” and its two possible translations in Spanish — “banco” (a financial establishment) in red and “orilla” (shore) in blue. The figure shows a two dimensional PCA for the aligned space of the two languages. The symbols are the anchors and the dots are the contextualized embeddings. (best viewed in color)

k	"TENER" ANCHOR
1	It may be difficult for the patient to **bear** the odour of the smoke at first , but once he gets used to such a smell , it does not really matter .
2	No matter what the better class of slave owners might do , they had to **bear** the stigma of cruelty with the worst of tyrants ...
3	Every new car will **bear** the lan name instead of Van Diemen , so the highly successful marque will gradually disappear .
4	had a sufficient economic stake to **bear** the litigation burden necessary to maintain a private suit for recovery under section 4 .
5	In this example , consumers **bear** the entire burden of the tax ; the tax incidence falls on consumers .

k	"OSO" ANCHOR
1	In 2010 , the government of the NWT decided to update its version of the polar **bear** - shaped plate .
2	Salad Fingers appears to be masochistic , as he can be seen taking pleasure from impaling his finger on a nail , rubbing stinging nettles on himself or stepping onto a **bear** trap .
3	The old **bear** - hunter , on being toasted , made a speech to the Texians , replete with his usual dry humor .
4	Balto arrives , distracts the **bear** , saves Aleu , they both escape and the **bear** disappears .
5	Defeated , the polar **bear** shrinks and transforms into a plush toy .

Table 10: Nearest-neighbors (after alignment) of the Spanish anchors "tener" (carry, verb) and "oso" (animal, noun) from the contextual embeddings of the word "bear" in English. The full sentence is presented for context.

Early Rumour Detection

Kaimin Zhou[2,3] **Chang Shu**[1,2] **Binyang Li**[3*] **Jey Han Lau**[2,4,5]

[1] School of Computer Science, University of Nottingham Ningbo China
[2] DeepBrain
[3] School of Information Science and Technology, University of International Relations
[4] School of Computing and Information Systems, The University of Melbourne
[5] IBM Research Australia
will@deepbrain.ai, scxcs1@nottingham.edu.cn,
byli@uir.edu.cn, jeyhan.lau@gmail.com

Abstract

Rumours can spread quickly through social media, and malicious ones can bring about significant economical and social impact. Motivated by this, our paper focuses on the task of rumour detection; particularly, we are interested in understanding how *early* we can detect them. Although there are numerous studies on rumour detection, few are concerned with the timing of the detection. A successfully-detected malicious rumour can still cause significant damage if it isn't detected in a timely manner, and so timing is crucial. To address this, we present a novel methodology for early rumour detection. Our model treats social media posts (e.g. tweets) as a data stream and integrates reinforcement learning to learn the number minimum number of posts required before we classify an event as a rumour. Experiments on Twitter and Weibo demonstrate that our model identifies rumours earlier than state-of-the-art systems while maintaining a comparable accuracy.

1 Introduction

The concept of rumour has a long history, and there are various definitions from different research communities (Allport and Postman, 1947). In this paper, we follow a commonly accepted definition of rumour, that it is an unverified statement, circulating from person to person and pertaining to an object, event, or issue of public concern and it is circulating without known authority for its truthfulness at the current time, but it may turn out to be true, or partly or entirely false; alternatively, it may also remain unresolved (Peterson and Gist, 1951; Zubiaga et al., 2018).

Rumours have the potential to spread quickly through social media, and bring about significant economical and social impact. Figure 1 illustrates an example of a rumour propagating on TWITTER. The source message started a claim about the cause of Michael Brown's shooting, and it was published shortly after the shooting happened. It claimed that he was shot ten times by the police for stealing candy. The message was retweeted by multiple users on TWITTER, and within 24 hours there were about 900K users involved, either by reposting, commenting, or questioning the original source message. From Figure 1, we see some users (e.g. User 7) question the veracity of the original message. Had the rumour been identified timely and rebutted, its propagation could have been contained.

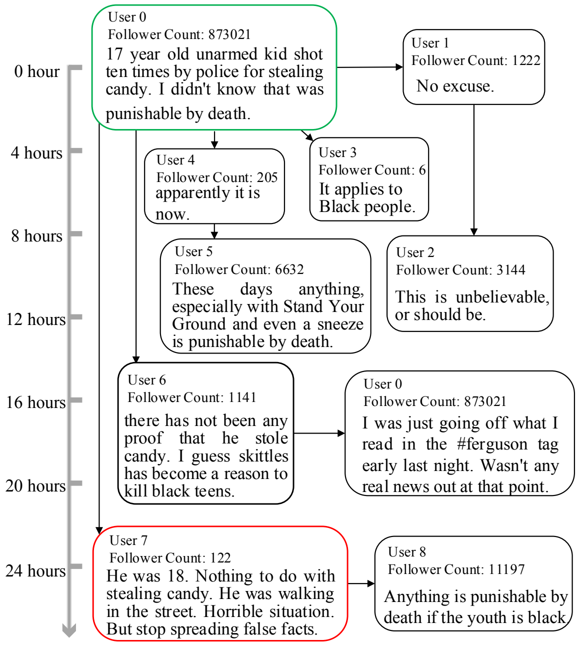


Figure 1: An illustration of a rumour propagating on TWITTER. The green box indicates the source message, and the red box highlights a post that rebuts the rumour.

Most studies (Qazvinian et al., 2011; Zhang et al., 2015) consider rumour detection as a binary classification problem, where they extract various

Proceedings of NAACL-HLT 2019, pages 1614–1623
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

features to capture rumour indicative signals for detecting a rumour, and a few recent works explore deep learning approaches to enhance detection accuracy (Long et al., 2017; Ruchansky et al., 2017). In all these studies, however, the timeliness of the rumour detection is not evaluated.

There are a few exceptions. In Ma et al. (2015) and Kwon et al. (2017), the authors define a checkpoint (e.g. number of posts or time elapsed after the source message) in the timeline and use all the posts prior to this checkpoint to classify a rumour. The checkpoint is often a pre-determined value for all rumours, and so does not capture the variation of propagation patterns for different rumours.

The focus of our paper is on *early* rumour detection. That is, our aim is to identify rumours as early as possible, while keeping a reasonable detection accuracy. Our early rumour detection system (ERD) features two modules: a rumour detection module that classifies whether an event (which consists of a number of posts) constitutes a rumour, and a checkpoint module that determines when to trigger the rumour detection module.

ERD treats incoming posts as a data stream and monitors the posts in real time. When ERD receives a new post, this post — along with all prior posts of the same event — will be used to decide if it constitutes an appropriate checkpoint to trigger the rumour detection module. ERD integrates reinforcement learning for the checkpoint module to guide the rumour detection module, using its classification accuracy as a reward. Through reinforcement learning ERD is able to learn the minimum number of posts required to identify a rumour. In other words, ERD can dynamically determine the appropriate checkpoint for different rumours, and this feature is the core novelty of our methodology.

To evaluate our approach, we use standard microblog data sets from WEIBO and TWITTER. We compare our method with benchmark rumour detection systems (Ma et al., 2016; Ruchansky et al., 2017; Dungs et al., 2018) and found that ERD could on average identify rumours within 7.5 and 3.4 hours with an accuracy of 93.3% and 85.8% on WEIBO and TWITTER respectively. Our detection accuracy performance is better than a state-of-the-art system that that detects rumours within 12 hours.

To summarise, we present a novel methodology for rumour detection. Unlike most rumour detection systems, our approach determines the checkpoint for each event dynamically, by learning when it should classify it as a rumour. Our experimental results showed that ERD outperforms state-of-the-art methods over two benchmark data sets in detection accuracy and timeliness. Our proposed framework is flexible and the individual modules (i.e. the rumour detection and checkpoint module) can be extended to incorporate more complex networks for further improvements. An open source implementation of our model is available at: `https://github.com/DeepBrainAI/ERD`.

2 Related Work

Traditionally, research on rumour detection has mainly focused on developing handcrafted features for machine learning algorithms (Qazvinian et al., 2011). Takahashi and Igata (2012) propose a method for rumour detection on Twitter using cue words and tweets statistics. Yang et al. (2012) apply two new types of features — client-based and location-based features — to rumour detection on Sina Weibo. Beyond this, user-based (Liang et al., 2015) and topic-based (Yang et al., 2015) features have also been explored. Friggeri et al. (2014) demonstrate that there are structural differences in the propagation of rumours and non-rumours, and Wu et al. (2015) and Ma et al. (2017) experiment with using these propagation patterns extensively to improve detection.

More recently, deep learning models are explored for the task. Compared to traditional machine learning approaches, these deep learning models tend to rely less on sophisticated handcrafted features. Ma et al. (2016) introduce a rumour detection model for microblogs based on recurrent networks. The input to their model is simple tf-idf features but it outperforms models leveraging handcrafted features. Sampson et al. (2016) show that implicit linkages between conversation fragments improve detection accuracy. Long et al. (2017) present a deep attention model that learns a hidden temporal representation for each sequential posts to represent the hypothesis. Ruchansky et al. (2017) integrate textual, user response, and source information into their neural models and achieve better performance.

Most of these works focus on detection accuracy, and so largely ignore the timing of the detection. Ma et al. (2015) develop a dynamic time series structure to incorporate temporal information to the features to understand the whole life cycle of rumours. Zhao et al. (2015) propose a detection model using a set of regular expressions to find posts that question or rebut the rumour to detect it earlier. Dungs et al. (2018) present an approach that checks for a rumour after 5 or 10 retweets. These models are interested in *early* rumour detection, although the checkpoint for triggering a detection is pre-determined, and succeeding posts after the checkpoint are usually ignored. On a similar note but a different task, Farajtabar et al. (2017) experiment with reinforcement learning by combining it with a point process network activity model to detect fake news and found some success.

3 Model Architecture

Let E denote an event, and it consists of a series of relevant posts x_i, where x_0 denotes the source message and x_T the last relevant message.[1] The objective of early rumor detection is to make a classification decision whether E is a rumour as early as possible while keeping an acceptable detection accuracy.[2]

As shown in Figure 2, ERD has two modules: a rumour detection module (RDM) that classifies whether an event is a rumour, and a checkpoint module (CM) that decides when the rumour detection module should be triggered. The checkpoint module plays an important role here, as it is responsible for the timeliness of a detection.

3.1 Rumor Detection Module (RDM)

RDM contains three layers: a word embedding layer that maps input words into vectors, a max-pooling layer that extracts important features of a post, and a GRU (Cho et al., 2014) that processes the sequential posts of an event.

In the word embedding layer, we map words in post x_i into vectors, yielding vectors e_i^j for each word. To capture the most salient features of a post, we apply a max pooling operation (Collobert et al., 2011; Kim, 2014; Lau et al., 2017), producing a fixed size vector m_i:

$$m_i = \text{maxpool}([\mathbf{W}_m e_i^0; \mathbf{W}_m e_i^1; ...; \mathbf{W}_m e_i^K])$$

where K is the number of words in the post. Henceforth $\mathbf{W}$ in all equations are model parameters.

To capture the temporal relationship between multiple posts, we use a GRU (Cho et al., 2014):

$$h_i = \text{GRU}(m_i, h_{i-1}) \tag{1}$$

We take the final state h_N (N = number of posts received to date) and use it to perform rumour classification:

$$p = \text{softmax}(\mathbf{W}_p h_N + b_p) \tag{2}$$

where $p \in \mathbb{R}^2$, i.e. p^0 (p^1) gives the probability of the positive (negative) class.[3]

3.2 Checkpoint Module (CM)

Rather than setting a static checkpoint when to classify an event as a rumour, CM learns the number of posts needed to trigger RDM. To this end, we leverage deep reinforcement learning to identify the optimal checkpoint. We reward CM based on RDM's accuracy and also penalise CM slightly every time it decides to not trigger RDM (and continue to monitor the event). This way CM learns the trade-off between detection accuracy and timeliness. The reward function is detailed in Section 3.3.

We use the deep Q-learning model (Mnih et al., 2013) for CM. The optimal action-value function $Q^*(s, a)$ is defined as the maximum expected return achievable under state s, which can be formulated as follows:

$$Q^*(s,a) = E_{s'\ \varepsilon}[r + \gamma \max_{a'} Q_i(s', a')|s, a]$$

where r is the reward value, γ the discount rate, and the optimal action in all action sequence a' is selected to maximise the expected value of $r + \gamma Q^*(s', a')$.

The optimal action-value function obeys the Bellman equation and is used for iterative value update:

$$Q_{i+1}(s,a) = E[r + \gamma \max_{a'} Q_i(s', a')|s, a]$$

The above iterative algorithm will converge and reach the optimal action value function, i.e. $Q_i \rightarrow Q^*$ when $q \rightarrow \infty$ (Sutton et al., 1998).

---

[1] Relevant posts are defined as retweets or responses to a source message.

[2] The earliest possible time to classify E is when we receive the first post x_0.

[3] Although sigmoid activation is more appropriate here as it is a binary classification task, we used the softmax function because in preliminary experiments we considered a third neural class.

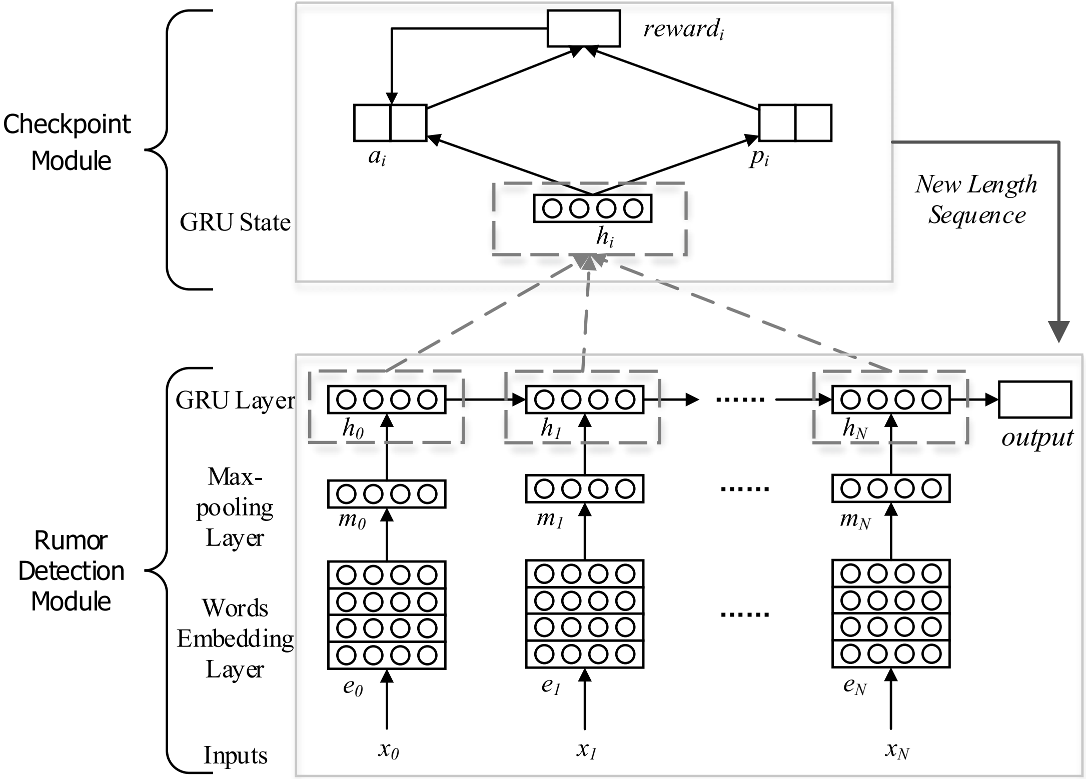


Figure 2: Architecture of ERD.

CM takes as input the hidden states produced by the GRU in RDM to compute the action-value function using a two-layer feedforward network:

$$a_i = \mathbf{W}_a(\text{ReLu}(\mathbf{W}_h h_i + b_h)) + b_a \qquad (3)$$

where $a_i \in \mathbb{R}^2$ is the action value for *terminate* (a_i^0) or *continue* (a_i^1) at post x_i. Note that a random action will be taken with the probability of θ irrespective to the action value a_i.

3.3 Joint Training

We train both RDM and CM jointly, and the training process is similar to that of generative adversarial networks (Goodfellow et al., 2014). The checkpoint module serves as the generator for action sequences, while the detection module is the discriminator. A key contrast, however, is that the two modules are working cooperatively rather than adversarially.

CM is trained using RDM's accuracy as reward. To compute the reward, we first pre-train RDM based on cross entropy:

$$-\sum_j [L_j \log(p_j^0) + (1 - L_j)(\log(p_j^1))] + \alpha l_2$$

where L_j is a binary label indicating the true class for event E_j, p is computed based on Equation (2), l_2 is the L2 loss for RDM parameters, and α is a hyper-parameter for scaling l_2.

We then train CM while keeping RDM's parameters fixed. In each step of the training, new posts that have arrived and previous GRU states are first fed to the RDM to produce the new states (Equation (1)), which will in turn be used by CM to calculate the action values (Equation (3)). This decides whether the system takes the *continue* or *terminate* action. If *terminate* is chosen, the reward is given in accordance to RDM's prediction; otherwise, a small penalty is incurred:

$$r_i = \begin{cases} \log M, & \textit{terminate} \text{ with correct prediction} \\ -P, & \textit{terminate} \text{ with incorrect prediction} \\ -\varepsilon, & \textit{continue} \end{cases}$$

where M is the number of correct predictions accumulated thus far, P is a large value to penalise an incorrect prediction, and ε is a small penalty value for delaying the detection.

To optimise our action value function, we apply the deep Q-learning approach with the experience replay algorithm (Mnih et al., 2013). Based on the optimal action-value function $Q^*(s, a)$, the objective of the action value function y_i is given as

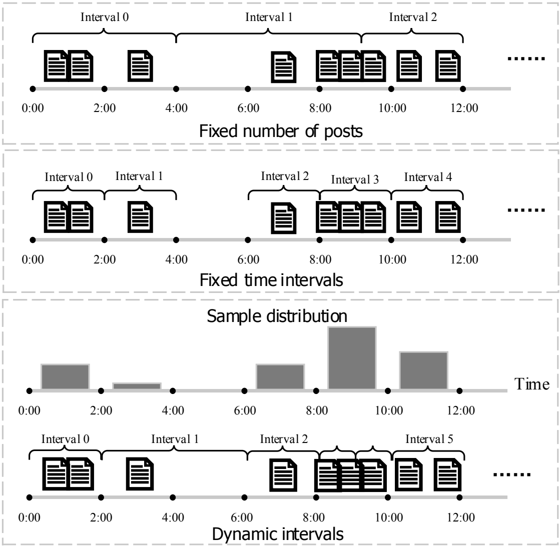


Figure 3: Three bucketing strategies to process streaming posts in batches.

follows:

$$y_i = \begin{cases} r_i, & \textit{terminate} \\ r_i + \gamma \max\limits_{a'} Q(h_{i+1}, a'; \theta), & \textit{continue} \end{cases}$$

where γ is the discount rate that decides how much experience is taken into consideration. And lastly, CM is optimised by minimising the cost:

$$(y_i - a_i)^2$$

We train CM and RDM in an alternating fashion, i.e. we train CM for several iterations while keeping RDM's parameters fixed, and then we move to train RDM for several iterations while keeping CM's parameters fixed. Training converges when CM's reward value stabilises between consecutive epochs.

3.4 Bucketing Strategy

For processing efficiency purposes, instead of processing each incoming post individually, we experiment with several bucketing strategies that group posts together and process them in batches. As Figure 3 illustrates, we group posts based on: (1) a fixed number of posts (FN), e.g. every 3 posts (i.e. 3 posts are combined together forming 1 single post); (2) a fixed time interval (FT), e.g. every 2 hours; and (3) a dynamic interval (DI) that ensures the number of posts collected in an interval is close to the mean number of posts collected in an hour in the full data set.

Statistics	WEIBO	TWITTER
User#	2,746,818	49,345
Posts#	3,805,656	103,212
Events#	4,664	5,802
Rumors#	2,313	1,972
Non-rumours	2,351	3,830
Avg. hours per event	2,460.7	33.4
Avg. # of posts per event	816	17
Max # of posts per event	59,318	346
Min # of posts per event	10	1

Table 1: Statistics of WEIBO and TWITTER.

4 Experiment

4.1 Data Set

We experiment with two data sets: WEIBO and TWITTER, developed by Ma et al. (2016) and Zubiaga et al. (2016) respectively.[4]

Statistics of the data sets is presented in Table 1. Even though both data sets have a comparable number of events, WEIBO is an order of magnitude larger than TWITTER as there are more posts per event. We reserve 10% of the events as the validation set for hyper-parameter tuning and early stopping, and split the rest in a ratio of 3:1 for training and test partitions.

4.2 Model Comparison

As a baseline, we use an SVM with tf-idf features. We also include several state-of-the-art rumour detection systems for comparisons: CSI (Ruchansky et al., 2017) on WEIBO; CRF (Zubiaga et al., 2016) and HMM (Dungs et al., 2018) on TWITTER; and GRU-2 (Ma et al., 2016) on both data sets. For GRU-2 (Ma et al., 2016) we also report performance on several variants that use a different recurrent network: simple RNN with tanh activation (RNN); single-layer LSTM (LSTM); and single-layer GRU (GRU-1).

CSI is a neural model that integrates text and users representations to classify rumours. CRF and HMM are classical models that use crowd opinions (a.k.a. stance) of the event for classification. GRU-2 is based on a two-layer GRU that captures contextual information of posts with tf-idf features as inputs.

---

[4] There is a small difference in the definition of a "rumour" in these two data sets. For WEIBO, all labelled rumours are false rumours (i.e. the source message contains verified untruthful statements), where else for TWITTER, rumours maybe truthful, untruthful, or unverified.

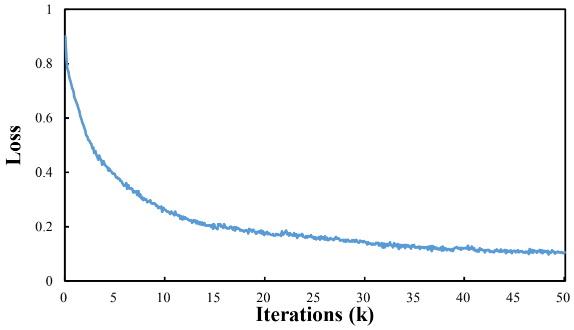


Figure 4: Loss over time during joint training.

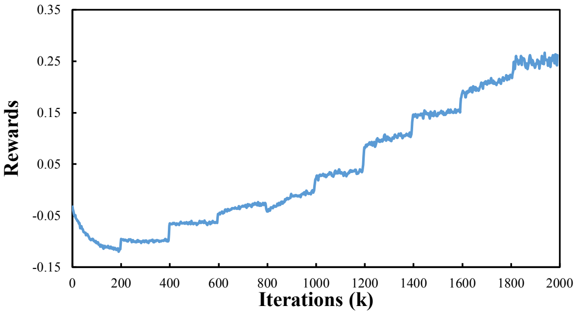


Figure 5: Reward over time during joint training.

4.3 Preprocessing and Hyper-parameters

We preprocess each post by segmenting them into words, and remove all stop words.[5] We pre-train word embeddings and kept them fixed during training.[6] θ is set to 0.01 and γ to 0.95; both values are determined empirically based on validation data. We use the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 0.001 during joint training, which we found to produce stable training.

4.4 Training Loss and Reward

We present the training loss and reward values over time during joint training in Figure 4 and Figure 5. We pre-train RDM for 2 epochs before joint training, and then we train RDM and CM in an alternating fashion for 1 epoch and 200K iterations respectively. We can see that loss declines steadily after 20K iterations and converges at around 50K iterations. The reward curve, on the other hand, fluctuates more as the reward was calculated based on the accuracy of RDM. When switching between training RDM and CM, the reward value tends to change abruptly, although over time we see a consistent improvement.

Method	Accuracy	Precision	Recall	F1
FN	**0.874**	**0.808**	0.835	**0.821**
FT	0.861	0.771	**0.850**	0.808
DI	0.814	0.771	0.767	0.769

Table 2: Classification performance for 3 bucketing strategies on TWITTER.

4.5 Results

4.5.1 Bucketing Strategy

Recall that we explore 3 different methods to group posts in order to process them in batches (Section 3.4). Here we evaluate them on rumour classification accuracy over the validation set of TWITTER. Note that we do not use CM here (and hence no reinforcement learning is involved) — we simply use all posts of an event to perform rumour classification with RDM. In terms of metrics we use standard accuracy, precision, recall and F1 scores. Results are presented in Table 2.

We see FN produces the best performance, and so FN is used for all following experiments as the default bucketing strategy.[7] As certain events have a long delay between posts, we also incorporate a maximum delay of one hour before processing the posts in a batch.

4.5.2 Detection Accuracy

In this section, we assess how accurately the models classify rumours. All baselines and benchmark systems uses all posts of an event to perform classification, with the exception of HMM which uses only the first 5 posts. For our models, we present: (1) the full model ERD, which uses a subset of posts for classification (checkpoint decided by CM); and (2) RDM, which uses the full set of posts. Results are detailed in Table 3 and 4.

We can see that RDM outperforms all models across most metrics, including state-of-the-art rumour detection systems CSI (marginally) and CRF (substantially). ERD, on the other hand, performs very competitively, outperforming most

---

[5] For TWITTER, words are tokenised using white spaces, and stopword list is based on NLTK (Bird et al., 2009). For WEIBO, Jieba is used for tokenisation: `https://pypi.org/project/jieba/`; and stopword list is a customised list based on: `http://blog.sina.com.cn/s/blog_a19ab3770102wjav.html`.

[6] For WEIBO, the embeddings are pre-trained using word2vec (Mikolov et al., 2013) on a separate Weibo data set we collected. For TWITTER, the embeddings are pre-trained GloVe embeddings (Pennington et al., 2014). Unknown words are initialised as zero vectors.

[7] FN value: 5 posts for WEIBO and 2 posts for TWITTER.

Method	Accuracy	Precision	Recall	F1
Baseline	0.724	0.673	0.746	0.707
RNN	0.873	0.816	0.964	0.884
LSTM	0.896	0.846	**0.968**	0.913
GRU-1	0.908	0.871	0.958	0.913
GRU-2	0.910	0.876	0.956	0.914
CSI*	0.953	—	—	0.954
RDM	**0.957**	**0.950**	0.963	**0.957**
ERD	0.933	0.929	0.936	0.932

Table 3: Detection accuracy on WEIBO. '*' denotes values taken from the original publications.

Method	Accuracy	Precision	Recall	F1
Baseline	0.612	0.355	0.465	0.398
RNN	0.785	0.707	0.659	0.682
LSTM	0.796	0.719	0.683	0.701
GRU-1	0.800	0.735	0.685	0.709
GRU-2	0.808	0.741	0.694	0.717
CRF*	—	0.667	0.566	0.607
HMM*	—	—	—	0.524
RDM	**0.873**	0.817	**0.823**	**0.820**
ERD	0.858	**0.843**	0.735	0.785

Table 4: Detection accuracy on TWITTER. '*' denotes values taken from the original publications.

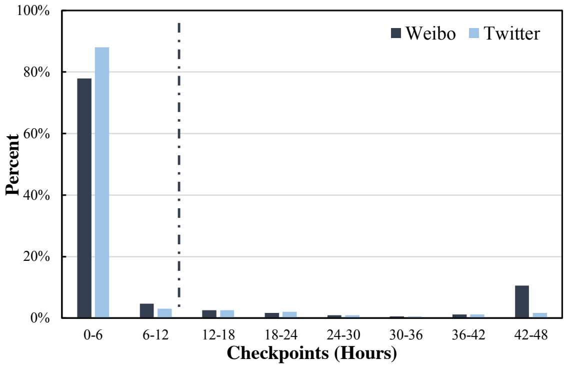


Figure 6: Proportion of events classified by ERD over time. Dashed line indicates the optimal checkpoint (12 hours) for GRU-2.

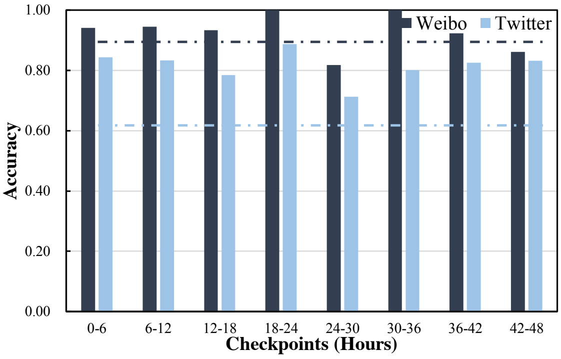


Figure 7: Detection accuracy of ERD over time. Dashed lines indicates GRU-2's accuracies.

benchmark systems and baselines, with the exception of CSI on WEIBO. Note, however, that unlike most other systems, ERD leverages only a subset of posts for rumour classification. HMM is the only benchmark system on TWITTER that uses a subset (first 5), and its performance is markedly worse than that of ERD (which uses 4.03 posts on average).

4.5.3 Detection Timeliness

Next we evaluate the timeliness of the detection, and we focus on comparing our system with GRU-2 (Ma et al., 2016), as it performed competitively in Section 4.5.2. Note that GRU-2 uses a manually set checkpoint (12 hours) that were found to be optimal, while ERD determines the checkpoint dynamically.

We first present the proportion of events that are classified by ERD over time (6-hour interval) in Figure 6.[8] We see that for both WEIBO and TWITTER, the majority of the events (approximately 80%) are classified within the first 6 hours. GRU-2's optimal checkpoint is 12 hours (dashed line), and so ERD is detecting rumours much earlier than GRU-2.

We next present the classification accuracy of these events over time (again, in 6-hour interval) in Figure 7. ERD generally outperforms GRU-2 (dashed lines) over all checkpoints. To be fair, checkpoints that are longer than 12 hours are not exactly comparable, as ERD uses more posts than GRU-2 in these instances. But even if we consider only the first 2 intervals (0-6 and 6-12 hours), ERD still outperforms GRU-2 across both data sets, demonstrating that ERD detects rumours earlier and more accurately.

For the two checkpoints on WEIBO where GRU-2 outperforms ERD, in the first checkpoint (24-30) we find that there are only 5 events and so the difference is unlikely to be statistically robust. For the second checkpoint (42-48), we hypothesise that these events are possibly the diffi-

---

[8] We include all events (whether it is a true of false positive) that CM decides to checkpoint.

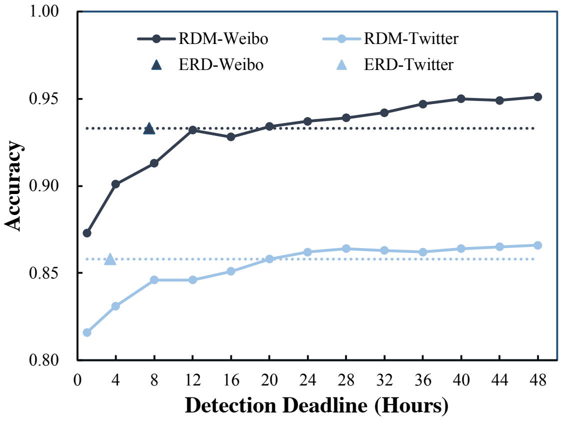


Figure 8: Detection accuracies of ERD and RDM over time.

Interval	Salient Words	Translation
18:41 – 18:44	大闸蟹，毒性，激素，有害，吃惊	hairy crabs, toxicity, hormone, harmful, amazed
18:48 – 18:51	大闸蟹，爆出，消息，吃惊，上市	hairy crabs, bursts, message, amazed, on the market
18:51 – 18:59	美食，为何，这样，晕，同城会	delicious food, why, so, dizzy, one city club
18:59 – 19:09	敢吃吗，吃得起，喜欢，惨，偷笑	dare to eat, afford to eat, like, miserable, laughing
19:11 – 19:15	食品安全，真的吗，失望，神马，不能	food safety, really, disappointment, what, cannot
Rumour Detected		
19:34 – 19:49	是不是，大闸蟹，吃不成，疑问，围观	is it, hairy crabs, cannot eat, doubt, look around

Table 5: Case study of a rumour on WEIBO.

cult cases, and as such the classification decision is deferred until much later (and classification performance is ultimately still low due to its difficulty).

To understand the advantage of incorporating reinforcement learning (CM) for rumour detection, we compute the detection accuracy over time for ERD and RDM in Figure 8. The dashed lines indicate the average accuracy performance of ERD, which detects rumours on average in 7.5 and 3.4 hours on WEIBO and TWITTER respectively. The solid lines show the accuracy performance of RDM, which increases over time as it has more evidence. For RDM to achieve the performance of ERD, we see that it requires approximately at least 20 hours of posts on both data sets. This highlights the importance of the checkpoint module, which allows ERD to detect rumours much earlier. In certain events, they are detected within 3 minutes.

4.5.4 Case Study

To provide a qualitative analysis for our approach, we showcase an example of a rumour event from WEIBO in Table 5. We present a set of salient words (second column) and their translations (third column) extracted from posts published during a particular period (first column) using simple tf-idf features.

The rumour was started by a message claiming that hairy crabs contain harmful hormones and toxins on August 18th, 2012. After the message was posted, within 12 hours 2.3M users participated in its propagation, either by re-posting, commenting, or questioning the original source message. The rumour spread quickly and led to significant economic damage to the aquaculture industry in China. Officially the rumour was rebutted after 24 hours, but in Table 5 we see that ERD detects the rumour in 34 minutes.

5 Conclusions

We present ERD, an early rumour detection system. Rather than setting a static checkpoint that determines when an event should be classified as rumour, ERD learns dynamically the minimum number of posts required to identify a rumour. To this end, we integrate reinforcement learning with recurrent neural networks to monitor social media posts in real time to decide when to classify rumours. We evaluate our model on two standard data sets, and demonstrate that ERD identifies rumours within 7.5 hours and 3.4 hours on WEIBO and TWITTER on average, compared to 12 hours of a competitive system. In terms of detection accuracy, ERD achieves a performance of 93.3% and 85.8%, which is comparable to state-of-the-art rumour detection systems.

Acknowledgements

This work is partially funded by the National Natural Science Foundation of China (61502115, U1636103, U1536207). We would also like to thank Wei Gao and Jing Li for their valuable suggestions.

References

Gordon W Allport and Leo Postman. 1947. The psychology of rumor.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — An-*

alyzing Text with the Natural Language Toolkit. O'Reilly Media, Sebastopol, USA.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. 12:2493–2537.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370.

Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. 2017. Fake news mitigation via point process based intervention. In *International Conference on Machine Learning*, pages 1097–1106.

Adrien Friggeri, Lada A Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *ICWSM*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–365.

Gang Liang, Wenbo He, Chun Xu, Liangyin Chen, and Jinquan Zeng. 2015. Rumor identification in microblogging systems based on users' behavior. *IEEE Transactions on Computational Social Systems*, 2(3):99–108.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 252–256.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 708–717.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Warren A Peterson and Noel P Gist. 1951. Rumor and public opinion. *American Journal of Sociology*, 57:159–167.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM.

Justin Sampson, Fred Morstatter, Liang Wu, and Huan Liu. 2016. Leveraging the implicit structure within social media for emergent rumor detection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2377–2382. ACM.

Richard S Sutton, Andrew G Barto, Francis Bach, et al. 1998. *Reinforcement learning: An introduction*. MIT press.

Tetsuro Takahashi and Nobuyuki Igata. 2012. Rumor detection on twitter. In *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on*, pages 452–457. IEEE.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 651–662. IEEE.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Zhifan Yang, Chao Wang, Fan Zhang, Ying Zhang, and Haiwei Zhang. 2015. Emerging rumor identification for social media with hot topic detection. In *Web Information System and Application Conference (WISA), 2015 12th*, pages 53–58. IEEE.

Qiao Zhang, Shuiyuan Zhang, Jian Dong, Jinhua Xiong, and Xueqi Cheng. 2015. Automatic detection of rumor on social network. In *Natural Language Processing and Chinese Computing*, pages 113–122. Springer.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. International World Wide Web Conferences Steering Committee.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016. Learning reporting dynamics during breaking news for rumour detection in social media. *arXiv preprint arXiv:1610.07363*.

Microblog Hashtag Generation via Encoding Conversation Contexts

Yue Wang [1*] **Jing Li**[2†] **Irwin King**[1] **Michael R. Lyu**[1] **Shuming Shi**[2]
[1]Department of Computer Science and Engineering
The Chinese University of Hong Kong, HKSAR, China
[2]Tencent AI Lab, Shenzhen, China
[1]{yuewang, king, lyu}@cse.cuhk.edu.hk
[2]{ameliajli,shumingshi}@tencent.com

Abstract

Automatic hashtag annotation plays an important role in content understanding for microblog posts. To date, progress made in this field has been restricted to phrase selection from limited candidates, or word-level hashtag discovery using topic models. Different from previous work considering hashtags to be inseparable, our work is the first effort to annotate hashtags with a novel *sequence generation* framework via viewing the hashtag as a short sequence of words. Moreover, to address the data sparsity issue in processing short microblog posts, we propose to jointly model the target posts and the *conversation contexts* initiated by them with bidirectional attention. Extensive experimental results on two large-scale datasets, newly collected from English Twitter and Chinese Weibo, show that our model significantly outperforms state-of-the-art models based on classification.[1] Further studies demonstrate our ability to effectively generate rare and even unseen hashtags, which is however not possible for most existing methods.

1 Introduction

Microblogs have become an essential outlet for individuals to voice opinions and exchange information. Millions of user-generated messages are produced every day, far outpacing the human being's reading and understanding capacity. As a result, the current decade has witnessed the increasing demand for effectively discovering gist information from large microblog texts. To identify the key content of a microblog post, hashtags, user-generated labels prefixed with a "#" (such as "*#NAACL*" and "*#DeepLearning*"), have been widely used to reflect keyphrases (Zhang et al., 2016, 2018) or topics (Yan et al., 2013; Hong et al., 2012; Li et al., 2016). Hashtags can further benefit downstream applications, such as microblog search (Efron, 2010; Bansal et al., 2015), summarization (Zhang et al., 2013; Chang et al., 2013), sentiment analysis (Davidov et al., 2010; Wang et al., 2011), and so forth. Despite the widespread use of hashtags, there are a large number of microblog messages without any user-provided hashtags. For example, less than 15% tweets contain at least one hashtag (Wang et al., 2011; Khabiri et al., 2012). Consequently, for a multitude of posts without human-annotated hashtags, there exists a pressing need for automating the hashtag annotation process for them.

Target post for hashtag generation

This *Azarenka* woman needs a talking to from the umpire her weird noises are totes inappropes professionally. ***#AusOpen***

Replying messages forming a conversation

[T1] How annoying is she. I just worked out what she sounds like one of those turbo charged cars when they change gear or speed.

[T2] On the topic of noises, I was at the *Nadal-Tomic* game last night and I loved how quiet *Tomic* was compared to *Nadal*.

[T3] He seems to have a shitload of talent and the postmatch press conf. He showed a lot of maturity and he seems nice.

[T4] *Tomic* has a fantastic *tennis* brain...

Table 1: A post and its conversation snippet about "Australian Open" on Twitter. "*#AusOpen*" is the human-annotated hashtag for the target post. *Words indicative of the hashtag* are in blue and italic type.

Most previous work in this field focuses on extracting phrases from target posts (Zhang et al., 2016, 2018) or selecting candidates from a pre-

---

[*] This work was mainly done when Yue Wang was an intern at Tencent AI Lab.

[†] Jing Li is the corresponding author.

[1] To obtain our datasets, please contact Yue Wang and Jing Li.

Proceedings of NAACL-HLT 2019, pages 1624–1633
Minneapolis, Minnesota, June 2 - June 7, 2019.

defined list (Gong and Zhang, 2016; Huang et al., 2016; Zhang et al., 2017). However, hashtags usually appear in neither the target posts nor the given candidate list. The reasons are two folds. For one thing, microblogs allow large freedom for users to write whatever hashtags they like. For another, due to the wide range and rapid change of social media topics, a vast variety of hashtags can be daily created, making it impossible to be covered by a fixed candidate list. Prior research from another line employs topic models to generate topic words as hashtags (Gong et al., 2015; Zhang et al., 2016). These methods, ascribed to the limitation of most topic models, are nevertheless incapable of producing phrase-level hashtags.

In this paper, we approach hashtag annotation from a novel *sequence generation* framework. In doing so, we enable phrase-level hashtags beyond the target posts or the given candidates to be created. Here, hashtags are first considered as a sequence of tokens (e.g., "*#DeepLearning*" as "*deep learning*"). Then, built upon the success of sequence to sequence (seq2seq) model on language generation (Sutskever et al., 2014), we present a neural seq2seq model to generate hashtags in a *word-by-word* manner. To the best of our knowledge, *we are the first to deal with hashtag annotation in sequence generation architecture*.

In processing microblog posts, one major challenge we might face is the limited features to be encoded. It is mostly caused by the data sparsity exhibited in short and informal microblog posts.[2] To illustrate such challenge, Table 1 displays a sample Twitter post tagged with "*#AusOpen*", referring to Australian Open tennis tournament. Only given the short post, it is difficult to understand why it is tagged with "*#AusOpen*", not to mention that neither "*aus*" nor "*open*" appear in the target post. In such a situation, how shall we generate hashtags for a post with limited words?

To address the data sparsity challenge, we exploit conversations initiated by the target posts to enrich their contexts. Our approach is benefited from the nature that most messages in a conversation tend to focus on relevant topics. Content in conversations might hence provide contexts facilitating the understanding of the original post (Chang et al., 2013; Li et al., 2015). The effects of conversation contexts, useful on topic modeling (Li et al., 2016, 2018) and keyphrase extraction (Zhang et al., 2018), have never been explored on microblog hashtag generation. To show why conversation contexts are useful, we display in Table 1 a conversation snippet formed by some replies of the sample target post. As can be seen, key content words in the conversation (e.g., "*Nadal*", "*Tomic*", and "*tennis*") are useful to reflect the relevance of the target post to the hashtag "*#AusOpen*", because Nadal and Tomic are both professional tennis players. Concretely, our model employs a dual encoder (i.e., two encoders), one for the target post and the other for the conversation context, to capture the representations from the *two sources*. Furthermore, to capture their joint effects, we employ the bidirectional attention (**bi-attention**) (Seo et al., 2016) to explore the interactions between two encoders' outputs. Afterward, an attentive decoder is applied to generate the word sequence of the hashtag.

In experiments, we construct two large-scale datasets, one from English platform Twitter and the other from Chinese Weibo. Experimental results based on both information retrieval and text summarization metrics show that our model generates hashtags closer to human-annotated ones than all the comparison models. For example, our model achieves 45.03% ROUGE-1 F1 on Weibo, compared to 25.34% given by the state-of-the-art classification-based method. Further comparisons with classification-based models show that our model, in a sequence generation framework, can better produce rare and even new hashtags.

To summarize, our contributions are three-fold:

- We are the first to approach microblog hashtag annotation with *sequence generation* architecture.
- To alleviate data sparsity, we enrich context for short target posts with their *conversations* and employ a bi-attention mechanism for capturing their interactions.
- Our proposed model outperforms state-of-the-art models by large margins on two large-scale datasets, constructed as part of this work.

2 Neural Hashtag Generation Model

In this section, we describe our framework shown in Figure 1. There are two major modules: a dual encoder to encode both target posts and their conversations with a bi-attention to explore their interactions, and a decoder to generate hashtags.

---

[2] For instance, the eligible length of a post on Twitter or Weibo is up to 140 characters.

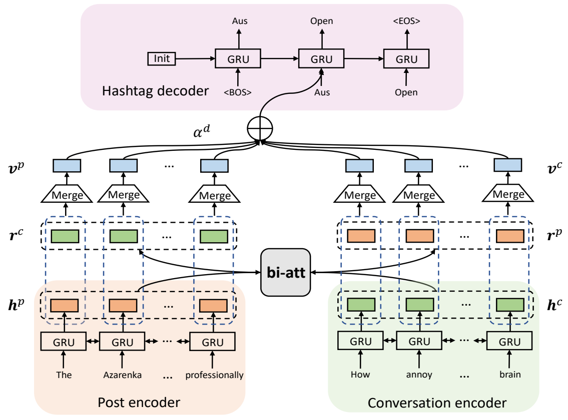


Figure 1: Our hashtag generation framework with a dual encoder, including a post encoder and a conversation encoder, where a bi-attention (bi-att) distills their salient features, followed by a merge layer to fuse them. An attentive decoder generates the hashtag sequence.

Input and Output. Formally, given a target post $\mathbf{x}^p$ formulated as word sequence $\langle x_1^p, x_2^p, ..., x_{|\mathbf{x}^p|}^p \rangle$ and its conversation context $\mathbf{x}^c$ formulated as word sequence $\langle x_1^c, x_2^c, ..., x_{|\mathbf{x}^c|}^c \rangle$, where $|\mathbf{x}^p|$ and $|\mathbf{x}^c|$ denote the number of words in the input target post and its conversation, respectively, our goal is to output a hashtag $\mathbf{y}$ represented by a word sequence $\langle y_1, y_2, ..., y_{|\mathbf{y}|} \rangle$. For training instances tagged with multiple gold-standard hashtags, we copy the instances multiple times, each with one gold-standard hashtag following Meng et al. (2017). All the input target posts, their conversations, and the hashtags share the same vocabulary V.

Dual Encoder. To capture representations from both target posts and conversation contexts, we design a dual encoder, composed of a post encoder and a conversation encoder, each taking the $\mathbf{x}^p$ and $\mathbf{x}^c$ as input, respectively.

For the post encoder, we use a bidirectional gated recurrent unit (Bi-GRU) (Cho et al., 2014) to encode the target post $\mathbf{x}^p$, where its embeddings $e(\mathbf{x}^p)$ are mapped into hidden states $\mathbf{h}^p = \langle \mathbf{h}_1^p, \mathbf{h}_2^p, ..., \mathbf{h}_{|\mathbf{x}^p|}^p \rangle$. Specifically, $\mathbf{h}_i^p = [\overrightarrow{\mathbf{h}_i^p}; \overleftarrow{\mathbf{h}_i^p}]$ is the concatenation of forward hidden state $\overrightarrow{\mathbf{h}_i^p}$ and backward hidden state $\overleftarrow{\mathbf{h}_i^p}$ for the i-th token:

$$\overrightarrow{\mathbf{h}_i^p} = GRU(e(\mathbf{x}_i^p), \overrightarrow{\mathbf{h}_{i-1}^p}), \tag{1}$$

$$\overleftarrow{\mathbf{h}_i^p} = GRU(e(\mathbf{x}_i^p), \overleftarrow{\mathbf{h}_{i+1}^p}). \tag{2}$$

Likewise, the conversation encoder converts conversations into hidden states $\mathbf{h}^c$ via another Bi-GRU. The dimensions of both $\mathbf{h}^p$ and $\mathbf{h}^c$ are d.

Bi-attention. To further distill useful representations from our two encoders, we employ the bi-attention to explore the interactions between the target posts and their conversations. The adoption of bi-attention is inspired by Seo et al. (2016), where the bi-attention was applied to extract query-aware contexts for machine comprehension. Our intuition is that the content concerning the key points in target posts might have their relevant words frequently appearing in their conversation contexts, and vice versa. In general, such content can reflect what the target posts focus on and hence effectively indicate what hashtags should be generated. For instance, in Table 1, names of tennis players (e.g., "*Azarenka*", "*Nadal*", and "*Tomic*") are mentioned many times in both target posts and their conversations, which reveals why the hashtag is "*#AusOpen*".

To this end, we first put a *post-aware* attention on the conversation encoder with coefficients:

$$\alpha_{ij}^c = \frac{\exp(f_{score}(\mathbf{h}_i^p, \mathbf{h}_j^c))}{\sum_{j'=1}^{|\mathbf{x}^c|} \exp(f_{score}(\mathbf{h}_i^p, \mathbf{h}_{j'}^c))}, \tag{3}$$

where the alignment score function $f_{score}(\mathbf{h}_i^p, \mathbf{h}_j^c) = \mathbf{h}_i^p \mathbf{W}_{bi-att} \mathbf{h}_j^c$ captures the similarity of the i-th word in the target post and the j-th word in its conversation. Here $\mathbf{W}_{bi-att} \in \mathbb{R}^{d \times d}$ is a weight matrix to be learned. Then, we compute a context vector $\mathbf{r}^c$ conveying post-aware conversation representations, where the i-th value is defined as:

$$\mathbf{r}_i^c = \sum_{j=1}^{|\mathbf{x}^c|} \alpha_{ij}^c \mathbf{h}_j^c. \tag{4}$$

Analogously, a *conversation-aware* attention on post encoder is used to capture the conversation-aware post representations as $\mathbf{r}^p$.

Merge Layer. Next, to further fuse representations distilled by the bi-attention on each encoder, we design a *merge* layer, a multilayer perceptron (MLP) activated by hyperbolic function:

$$\mathbf{v}^p = \tanh(\mathbf{W}_p[\mathbf{h}^p; \mathbf{r}^c] + \mathbf{b}_p), \tag{5}$$

$$\mathbf{v}^c = \tanh(\mathbf{W}_c[\mathbf{h}^c; \mathbf{r}^p] + \mathbf{b}_c), \tag{6}$$

where $\mathbf{W}_p, \mathbf{W}_c \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b}_p, \mathbf{b}_c \in \mathbb{R}^d$ are trainable parameters.

Note that either $\mathbf{v}^p$ or $\mathbf{v}^c$ conveys the information from both posts and conversations, but with a different emphasis. Specifically, $\mathbf{v}^p$ mainly retains the contexts of posts with the auxiliary information from conversations, while $\mathbf{v}^c$ does the opposite. Finally, vectors $\mathbf{v}^p$ and $\mathbf{v}^c$ are concatenated and fed into the decoder for hashtag generation.

Decoder. Given the representations $\mathbf{v} = [\mathbf{v}^p; \mathbf{v}^c]$ produced by our dual encoder with bi-attention, we apply an attention-based GRU decoder to generate a word sequence $\mathbf{y}$ as the hashtag. The probability to generate the hashtag conditioned on a target post and its conversation is defined as:

$$Pr(\mathbf{y}|\mathbf{x}^p, \mathbf{x}^c) = \prod_{t=1}^{|\mathbf{y}|} Pr(y_t|\mathbf{y}_{<t}, \mathbf{x}^p, \mathbf{x}^c), \quad (7)$$

where $\mathbf{y}_{<t}$ refers to $(y_1, y_2, ..., y_{t-1})$.

Concretely, when generating the t-th word in hashtag, the decoder emits a hidden state vector $\mathbf{s}_t \in \mathbb{R}^d$ and puts a global attention over $\mathbf{v}$. The attention aims to exploit indicative representations from the encoder outputs $\mathbf{v}$ and summarizes them into a context vector $\mathbf{c}_t$ defined as:

$$\mathbf{c}_t = \sum_{i=1}^{|\mathbf{x}^p|+|\mathbf{x}^c|} \alpha^d_{ti} \mathbf{v}_i, \quad (8)$$

$$\alpha^d_{ti} = \frac{\exp(g_{score}(\mathbf{s}_t, \mathbf{v}_i))}{\sum_{i'=1}^{|\mathbf{x}^p|+|\mathbf{x}^c|} \exp(g_{score}(\mathbf{s}_t, \mathbf{v}_{i'}))}, \quad (9)$$

where $g_{score}(\mathbf{s}_t, \mathbf{v}_i) = \mathbf{s}_t \mathbf{W}_{att} \mathbf{v}_i$ is another alignment function ($\mathbf{W}_{att} \in \mathbb{R}^{d \times d}$) to measure the similarity between $\mathbf{s}_t$ and $\mathbf{v}_i$.

Finally, we map the current hidden state $\mathbf{s}_t$ of the decoder together with the context vector $\mathbf{c}_t$ to a word distribution over the vocabulary V via:

$$Pr(y_t|\mathbf{y}_{<t}, \mathbf{x}^p, \mathbf{x}^c) = softmax(\mathbf{W}_v[\mathbf{s}_t; \mathbf{c}_t] + \mathbf{b}_v), \quad (10)$$

which reflects how likely a word to be the t-th word in the generated hashtag sequence. Here $\mathbf{W}_v \in \mathbb{R}^{V \times 2d}$ and $\mathbf{b}_v \in \mathbb{R}^V$ are trainable weights.

Learning and Inferring Hashtags. During the training stage, we apply stochastic gradient descent to minimize the loss function of our entire framework, which is defined as:

$$\mathcal{L}(\Theta) = -\sum_{n=1}^{N} \log(Pr(\mathbf{y}_n|\mathbf{x}^p_n, \mathbf{x}^c_n; \Theta)). \quad (11)$$

Here N is the number of training instances and Θ denotes the set of all the learnable parameters.

Datasets	# of posts	Avg len of posts	Avg len of convs	Avg len of tags	# of tags per post
Twitter	44,793	13.27	29.94	1.69	1.14
Weibo	40,171	32.64	70.61	2.70	1.11

Table 2: Statistics of our datasets. Avg len of posts, convs, tags refer to the average number of words in posts, conversations, and hashtags, respectively.

Datasets	|Tagset|	$\mathcal{P}$	$\mathcal{C}$	$\mathcal{P} \cup \mathcal{C}$
Twitter	4,188	2.72%	5.58%	7.69%
Weibo	5,027	8.29%	6.21%	12.52%

Table 3: Statistics of the hashtags. |Tagset|: the number of distinct hashtags. $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{P} \cup \mathcal{C}$: the percentage of hashtags appearing in their corresponding posts, conversations, and the union set of them, respectively.

In hashtag inference, based on the produced word distribution at each time step, word selection is conducted using beam search. In doing so, we generate a ranking list of output hashtags, where the top K hashtags serve as our final output.

3 Experiment Setup

Here we describe how we set up our experiments.

Datasets and Statistic Analysis. Two large-scale experiment datasets are *newly collected* from popular microblog platforms: an English Twitter dataset and a Chinese Weibo dataset. The Twitter dataset was built based on the TREC 2011 microblog track.[3] To recover the conversations, we used Tweet Search API to fetch "in-reply-to" relations in a recursive way. The Weibo dataset was collected from January to August 2014 using Weibo Search API via searching messages with the trending queries[4] as keywords. For gold-standard hashtags, we take the user-annotated hashtags, appearing before or after a post, as the reference.[5] The statistics of our datasets are shown in Table 2. We randomly split both datasets into three subsets, where 80%, 10%, and 10% of the data corresponds to training, development, and test sets, respectively.

To further investigate how challenging our problem is, we show some statistics of the hashtags in Table 3 and the distributions of hashtag frequency in Figure 2. In Table 3, we observe

---

[3] `https://trec.nist.gov/data/tweets/`

[4] `http://open.weibo.com/wiki/Trends/`

[5] Hashtags in the middle of a post are not considered here as they generally act as semantic elements (Zhang et al., 2016, 2018).

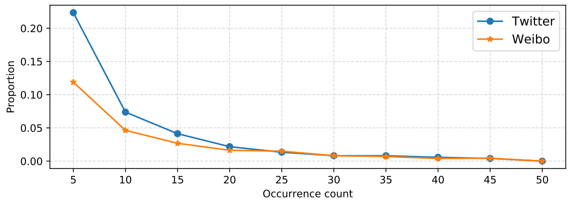


Figure 2: Distribution of hashtag frequency. The horizontal axis refers to the occurrence count of hashtags (shown with maximum 50 and bin 5) and the vertical axis denotes the data proportion.

the large size of hashtags in both datasets. Moreover, Figure 2 indicates that most hashtags only appear a few times. Given such a large and imbalanced hashtag space, hashtag selection from a candidate list, as many existing methods do, might not perform well. Table 3 also shows that only a small proportion of hashtags appearing in their posts, conversations, and either of them, making it inappropriate to directly extract words from the two sources to form hashtags.

Preprocessing. For tokenization and word segmentation, we employed the tweet preprocessing toolkit released by Baziotis et al. (2017) for Twitter, and the Jieba toolkit[6] for Weibo. Then, for both Twitter and Weibo, we further take the following preprocessing steps: First, single-character hashtags were filtered out for not being meaningful. Second, generic tags, i.e., links, mentions (@username), and numbers, were replaced with "URL" "MENTION", and "DIGIT", respectively. Third, inappropriate replies (e.g., retweet-only messages) were removed, and the remainder were chronologically ordered to form a sequence as conversation contexts. Last, a vocabulary was maintained with the $30K$ and $50K$ most frequent words, for Twitter and Weibo, respectively.

Comparisons. For experiment comparisons, we first consider a weak baseline RANDOM that randomly ranks hashtags seen from training data. Two unsupervised baselines are also considered, where words are ranked by latent topics induced with the latent Dirichlet allocation topic model (henceforth LDA), and by their TF-IDF scores (henceforth TF-IDF). Here for TF-IDF scores, we consider the N-gram TF-IDF ($N \leq 5$). Besides, we compare with *supervised* models below:

- EXTRACTOR: Following Zhang et al. (2018), we extract phrases from target posts as hashtags via sequence tagging and encode conversations with memory networks (Sukhbaatar et al., 2015).
- CLASSIFIER: We compare with the state-of-the-art model based on classification (Gong and Zhang, 2016), where hashtags are selected from candidates seen in training data. Here two versions of their classifier are considered, one only taking a target post as input (henceforth CLASSIFIER (*post only*)) and the other taking the concatenation of a target post and its conversation as input (henceforth CLASSIFIER (*post+conv*)).
- GENERATOR: A seq2seq generator (henceforth SEQ2SEQ) (Sutskever et al., 2014) is applied to generate hashtags given a target post. We also consider its variant augmented with copy mechanism (Gu et al., 2016) (henceforth SEQ2SEQ-COPY), which has proven effective in keyphrase generation (Meng et al., 2017) and also takes the post as input. The proposed seq2seq with the bi-attention to encode both the post and its conversation is denoted as OUR MODEL for simplicity.

Model Settings. We conduct model tunings on the development set based on grid search, where the hyper-parameters that give the lowest objective loss are selected. For the sequence generation models, the implementations are based on the OpenNMT framework (Klein et al., 2017). The word embeddings, with dimension set to 200, are randomly initialized. For encoders, we employ two layers of Bi-GRU cells, and for decoders, one layer of GRU cell is used. The hidden size of all GRUs is set to 300. In learning, we use the Adam optimizer (Kingma and Ba, 2014) with the learning rate initialized to 0.001. We adopt the early-stop strategy: the learning rate decreases by a decay rate of 0.5 till either it is below $1e^{-6}$ or the validation loss stops decreasing. The norm of gradients is rescaled to 1 if the $L2$-norm > 1 is observed. The dropout rate is 0.1 and the batch size is 64. In inference, we set the beam-size to 20 and the maximum sequence length of a hashtag to 10.

For CLASSIFIER and EXTRACTOR, lacking publicly available codes, we reimplement the models using Keras.[7] Their results are reproduced in their original experiment settings. For LDA, we employ an open source toolkit lda.[8]

Evaluation Metrics. Popular *information retrival* evaluation metrics F1 scores at K (F1@K)

---

[6] `https://pypi.python.org/pypi/jieba/`

[7] `https://keras.io/`

[8] `https://pypi.org/project/lda/`

Model	Twitter					Weibo				
	F1@1	F1@5	MAP	RG-1	RG-4	F1@1	F1@5	MAP	RG-1	RG-4
Baselines										
Random	0.37	0.63	0.89	0.56	0.16	0.43	0.67	0.97	2.14	1.13
LDA	0.13	0.25	0.35	0.60	-	0.10	0.86	0.94	3.89	-
Tf-Idf	0.02	0.02	0.03	0.54	0.14	0.85	0.73	1.30	8.04	4.29
Extractor	0.44	-	-	1.14	0.14	2.53	-	-	7.64	5.20
State of the arts										
Classifier (*post only*)	9.44	6.36	12.71	10.75	4.00	16.92	10.48	22.29	25.34	21.95
Classifier (*post+conv*)	8.54	6.28	12.10	10.00	2.47	17.25	11.03	23.11	25.16	22.09
Generators										
Seq2Seq	10.44	6.73	14.00	10.52	4.08	26.00	14.43	32.74	37.37	32.67
Seq2Seq-copy	10.63	6.87	14.21	12.05	4.36	25.29	14.10	31.63	37.58	32.69
Our model	**12.29***	**8.29***	**15.94***	**13.73***	**4.45**	**31.96***	**17.39***	**38.79***	**45.03***	**39.73***

Table 4: Comparison results on Twitter and Weibo datasets (in %). RG-1 and RG-4 refer to ROUGE-1 and ROUGE-SU4 respectively. The best results in each column are in bold. The "*" after numbers indicates significantly better results than all the other models ($p < 0.05$, paired t-test). Higher values indicate better performance.

and mean average precision (MAP) scores (Manning et al., 2008) are reported. Here, different K values are tested on F1@K and result in a similar trend, so only F1@1 and F1@5 are reported. MAP scores are also computed given the top 5 outputs. Besides, as we consider a hashtag as a sequence of words, ROUGE metrics for *summarization* evaluation (Lin, 2004) are also adopted. Here, we use ROUGE F1 for the top-ranked hashtag prediction computed by an open source toolkit pythonrouge,[9] with Porter stemmer used for English tweets. For Weibo posts, scores calculated at the Chinese character level following Li et al. (2018). We report the average scores for multiple gold-standard hashtags on ROUGE evaluation.

4 Experimental Results

In this section, we first report the main comparison results in Section 4.1, followed by an in-depth comparative study between classification and sequence generation models in Section 4.2. Further discussions are then presented to analyze our superiority and errors in Section 4.3.

4.1 Main Comparison Results

Table 4 reports the main comparison results. For Classifier, their outputs are ranked according to the logits after a $softmax$ layer. For Extractor, it is unable to produce ranked hashtags and thus no results are reported for F1@5 and MAP. For LDA, as it cannot generate bigram hashtags, no results are presented for ROUGE-SU4. In general, we have the following observations:

• ***Hashtag annotation is more challenging for Twitter than Weibo.*** Generally, all models perform worse on Twitter measured by different metrics. The intrinsic reason is the essential language difference between English and Chinese microblogs. English allows higher freedom in writing, resulting in more variety in Twitter hashtags (e.g., abbreviations are prominent like "*aus*" in "*#AusOpen*"). For statistical reasons, Twitter hashtags are more likely to be absent in either posts or conversations (Table 3), and have a more severe imbalanced distribution (Figure 2).

• ***Topic models and extractive models are ineffective for hashtag annotation***. The poor performance of all baseline models indicates that hashtag annotation is a challenging problem. LDA sometimes performs even worse than Random due to its inability to produce phrase-level hashtags. For extractive models, both Tf-Idf and Extractor fail to achieve good results. It is because most hashtags are absent in target posts, as we see in Table 3 that only 2.72% hashtags on Twitter and 8.29% on Weibo appear in target posts. This confirms that extractive models, relying on word selection from target posts, cannot well fit the hashtag annotation scenario. For the same reason, copy mechanism fails to bring noticeable improvements for the seq2seq generator on both datasets.

• ***Sequence generation models outperform other counterparts.*** When comparing Generators with other models, we find the former uniformly achieve better results, showing the superiority to produce hashtags with sequence generation framework. Classification models, though as

---

[9] `https://github.com/tagucci/pythonrouge`

the state of the art, expose their inferiority as they select labels from the large and imbalanced hashtag space (reflected in Table 3 and Figure 2).

- ***Conversations are useful for hashtag generation.*** Among the sequence generation models, OUR MODEL achieves the best performance across all the metrics. The observation indicates the usefulness of bi-attention in exploiting the joint effects of target posts and their conversations, which further helps in identifying indicative features from both sources for hashtag generation. However, interestingly, incorporating conversations fails to boost the classification performance. The reason why OUR MODEL better exploits conversations than CLASSIFIER (*post+conv*) might be that we can attend the indicative features when decoding each word in the hashtag, which is however not possible for classification models (considering hashtags to be inseparable).

4.2 Classification vs. Generation

From Table 4, we observe that the classifiers outperform topic models and extractive models by a large margin but exhibit generally worse results than sequence generation models. Here, we present a thorough study to compare hashtag classification and generation. Four models are selected for comparison: two classifiers, CLASSIFIER (*post only*) and CLASSIFIER (*post+conv*), and two sequence generation models, SEQ2SEQ and OUR MODEL. Below, we explore how they perform to predict rare and new hashtags.

Rare Hashtags. According to the hashtag distributions in Figure 2, we can see a large proportion of hashtags appearing only a few times in the data. To study how models perform to predict such hashtags, in Figure 3, we display their F1@1 scores in inferring hashtags with varying frequency. The lower F1 score on less frequent hashtags indicates the difficulty to yield rare hashtags. The reason probably comes from the overfitting issue caused by limited data to learn from.

We also observe that sequence generation models achieve consistently better F1@1 scores on hashtags with varying sparsity degree, while classification models suffer from the label sparsity issue and obtain worse results. The better performance of the former might result from the word-by-word generation manner in hashtag generation, which enables the internal structure of hashtags (how words form a hashtag) to be exploited.

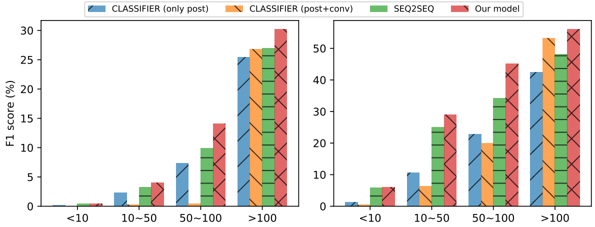


Figure 3: F1@1 on Twitter (the left subfigure) and Weibo (the right subfigure) in inferring hashtags with varying frequency. In each subfigure, from left to right shows the results of CLASSIFIER (*post only*), CLASSIFIER (*post+conv*), SEQ2SEQ, and OUR MODEL. Generation models consistently perform better.

New Hashtags. To further explore the extreme situation where hashtags are absent in the training set, we experiment to see how models perform in handling new hashtags. To this end, we additionally collect instances tagged with hashtags absent in training data and construct an external test set, with the same size as our original test set. Considering that classifiers will never predict unseen labels, to ensure comparable performance, we only adopt summarization metrics here for evaluation and report ROUGE-1 F1 scores in Table 5.

As can be seen, creating unseen hashtags is a challenging task, where unsurprisingly, all models perform poorly on this task. Nevertheless, sequence generation models perform much better on both datasets, e.g., at least 6.5x improvements over classification models observed on Weibo dataset. For Twitter dataset, the improvements are not that large, which confirms again that hashtag annotation on Twitter is more difficult due to the noisier data characteristics. In particular, compared to SEQ2SEQ, OUR MODEL achieves an additional performance gain in producing new hashtags by leveraging conversations with the bi-attention.

Model	**Twitter**	**Weibo**
CLASSIFIER (*post only*)	1.15	1.65
CLASSIFIER (*post+conv*)	1.13	1.52
SEQ2SEQ	1.33	10.84
OUR MODEL	**1.48**	**12.55**

Table 5: ROUGE-1 F1 scores (%) in producing unseen hashtags. Best results are in bold.

4.3 Further Discussions on Our Model

To further analyze our model, we conduct a quantitative ablation study, a qualitative case study, and an error analysis. We then discuss them in turn.

Ablation Study. We report the ablation study results in Table 6 to examine the relative contributions of the target posts and the conversation contexts. To this end, our model is compared with its five variants below: Seq2Seq (*post only*), Seq2Seq (*conv only*), and Seq2Seq (*post+conv*), using standard seq2seq to generate hashtags from their target posts, conversation contexts, and their concatenation, respectively; Our model (*post-att only*) and Our model (*conv-att only*), whose decoder only takes $\mathbf{v}^p$ and $\mathbf{v}^c$ defined in Eq. (5) and Eq. (6), respectively. The results show that solely encoding target posts is more effective than modeling the conversations alone, but exploring their joint effects can further boost the performance, especially combined with a bi-attention mechanism over them.

Model	Twitter	Weibo
Seq2Seq (*post only*)	10.44	26.00
Seq2Seq (*conv only*)	6.27	18.57
Seq2Seq (*post + conv*)	11.24	29.85
Our model (*post-att only*)	11.18	28.67
Our model (*conv-att only*)	10.61	28.06
Our model (*full*)	**12.29**	**31.96**

Table 6: F1@1 scores (%) for our variants.

Case Study. We further present a case study on the target post shown in Table 1, where the top five outputs of some comparison models are displayed in Table 7. As can be seen, only our model successfully generates "*aus open*", the gold standard. Particularly, it not only ranks the correct answer as the top prediction, but also outputs other semantically similar hashtags, e.g., sport-related terms like "*bbc football*", "*arsenal*", and "*murray*". On the contrary, Classifier and Seq2Seq tend to yield frequent hashtags, such as "*just saying*" and "*jan 25*". Baseline models also perform poorly: LDA produces some common single word, and TF-IDF extracts phrases in the target post, where the gold-standard hashtag is however absent.

Model	Top five outputs
LDA	found; stated; excited; card; apparently
TF-IDF	inappropes; umpire; woman need; azarenka woman; the umpire
Classifier	fail; facebook; just saying; quote; pro choice
Seq2Seq	fail; jan 25; yr; eastenders; facebook
Our model	***aus open*** ; bbc football ; bbc aus ; arsenal ; murray

Table 7: Model outputs for the target post in Table 1. "*aus open*" matches the gold-standard hashtag.

To analyze why our model obtains superior results in this case, we display the heatmap in Figure 4 to visualize our bi-attention weight matrix $\mathbf{W}_{bi-att}$. As we can see, bi-attention can identify the indicative word "*Azarenka*" in the target post, via highlighting its other pertinent words in conversations, e.g., "*Nadal*" and "*tennis*". In doing so, salient words in both the post and its conversations can be unveiled, facilitating the correct hashtag "*aus open*" to be generated.

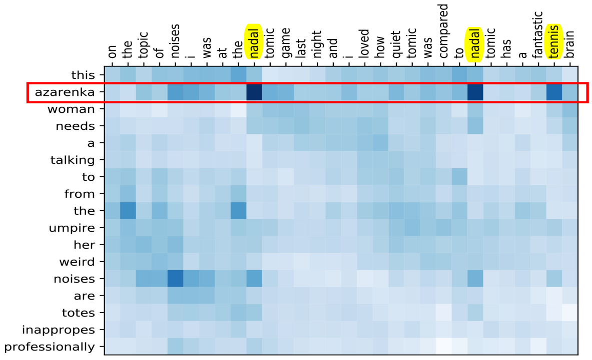


Figure 4: Visualization of bi-attention given the input case in Table 1. The horizontal axis denotes a snippet of a truncated conversation. The vertical axis shows the target post. Salient words are highlighted.

Error Analysis. Taking a closer look at our outputs, we find that one type of major errors comes from the unmatched outputs with gold standards, even as a close guess. For example, our model predicts "*super bowl*" for a post tagged with "*#steelers*", a team in super bowl. In future work, the semantic similarity should be considered in hashtag evaluation. Another primary type of error is caused by the non-topic hashtags, such as "*#fb*" (indicating the messages forwarded from Facebook). Such non-topic hashtags cannot reflect any content information from target posts and should be distinguished from topic hashtags in the future.

5 Related Work

Our work mainly builds on two streams of previous work — microblog hashtag annotation and neural language generation.

We are in the line of microblog hashtag annotation. Some prior work extracts phrases from target posts with sequence tagging models (Zhang et al., 2016, 2018). Another popular approach is to apply classifiers and select hashtags from a candidate list (Heymann et al., 2008; Weston et al., 2014; Sedhai and Sun, 2014; Gong and Zhang, 2016;

Huang et al., 2016; Zhang et al., 2017). Unlike them, we generate hashtags with a language generation framework, where hashtags in neither the target posts nor the pre-defined candidate list can be created. Topic models are also widely applied to induce topic words as hashtags (Krestel et al., 2009; Ding et al., 2012; Godin et al., 2013; Gong et al., 2015; Zhang et al., 2016). However, these models are usually unable to produce phrase-level hashtags, which can be achieved by ours via generating hashtag word sequences with a decoder.

Our work is also closely related to neural language generation, where the encoder-decoder framework (Sutskever et al., 2014) acts as a springboard for many sequence generation models. In particular, we are inspired by the keyphrase generation studies for scientific articles (Meng et al., 2017; Ye and Wang, 2018; Chen et al., 2018, 2019), incorporating word extraction and generation using a seq2seq model with copy mechanism. However, our hashtag generation task is inherently different from theirs. As we can see from Table 4, it is suboptimal to directly apply keyphrase generation models on our data. The reason mostly lies in the informal language style of microblog users in writing both target posts and their hashtags. To adapt our model on microblog data, we explore the effects of conversation contexts on hashtag generation, which has never been studied in any prior work before.

6 Conclusion

We have presented a novel framework of hashtag generation via jointly modeling of target posts and conversation contexts. To this end, we have proposed a neural seq2seq model with bi-attention over a dual encoder for capturing indicative representations from the two sources. Experimental results on two newly collected datasets have demonstrated that our proposed model significantly outperforms existing state-of-the-art models. Further studies have shown that our model can effectively generate rare and even unseen hashtags.

Acknowledgements

This work is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund). We thank NAACL reviewers for their insightful suggestions on various aspects of this work.

References

Piyush Bansal, Somay Jain, and Vasudeva Varma. 2015. Towards semantic retrieval of hashtags in microblogs. In *World Wide Web Conference*.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yi Chang, Xuanhui Wang, Qiaozhu Mei, and Yan Liu. 2013. Towards twitter context summarization with user influence models. In *International Conference on Web Search and Data Mining*.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Empirical Methods in Natural Language Processing*.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-guided encoding for keyphrase generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *International Conference on Computational Linguistics*.

Zhuoye Ding, Qi Zhang, and Xuanjing Huang. 2012. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *International Conference on Computational Linguistics*.

Miles Efron. 2010. Hashtag retrieval in a microblogging environment. In *Conference on Research and Development in Information Retrieval*.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *World Wide Web Conference*.

Yeyun Gong, Qi Zhang, and Xuanjing Huang. 2015. Hashtag recommendation using dirichlet process mixture models incorporating types of hashtags. In *Empirical Methods in Natural Language Processing*.

Yuyun Gong and Qi Zhang. 2016. Hashtag recommendation using attention-based convolutional neural network. In *International Joint Conference on Artificial Intelligence*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics*.

Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. 2008. Social tag prediction. In *Conference on Research and Development in Information Retrieval*.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *World Wide Web Conference*.

Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *International Conference on Computational Linguistics*.

Elham Khabiri, James Caverlee, and Krishna Yeswanth Kamath. 2012. Predicting semantic annotations on the real-time web. In *ACM Conference on Hypertext and Social Media*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Association for Computational Linguistics*.

Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *ACM Conference on Recommender Systems*.

Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng, and Kam-Fai Wong. 2015. Using content-level structures for summarizing microblog repost trees. In *Empirical Methods in Natural Language Processing*.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016. Topic extraction from microblog posts using conversation structures. In *Association for Computational Linguistics*.

Jing Li, Yan Song, Zhongyu Wei, and Kam-Fai Wong. 2018. A joint model of conversational discourse and latent topics on microblogs. *Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the Association for Computational Linguistics-04 Workshop*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Association for Computational Linguistics*.

Surendra Sedhai and Aixin Sun. 2014. Hashtag recommendation for hyperlinked tweets. In *Conference on Research and Development in Information Retrieval*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Neural Information Processing Systems*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Conference on Information and Knowledge Management*.

Jason Weston, Sumit Chopra, and Keith Adams. 2014. #tagspace: Semantic embeddings from hashtags. In *Association for Computational Linguistics*.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *World Wide Web Conference*.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Empirical Methods in Natural Language Processing*.

Qi Zhang, Jiawen Wang, Haoran Huang, Xuanjing Huang, and Yeyun Gong. 2017. Hashtag recommendation for multimodal microblog using co-attention network. In *International Joint Conference on Artificial Intelligence*.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Empirical Methods in Natural Language Processing*.

Renxian Zhang, Wenjie Li, Dehong Gao, and Ouyang You. 2013. Automatic twitter topic summarization with speech acts. *IEEE Trans. Audio, Speech & Language Processing*.

Yingyi Zhang, Jing Li, Yan Song, and Chengzhi Zhang. 2018. Encoding conversation context for neural keyphrase extraction from microblog posts. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems

Steffen Eger[†‡], **Gözde Gül Şahin**[†‡], **Andreas Rücklé**[†], **Ji-Ung Lee**[†‡], **Claudia Schulz**[†‡], **Mohsen Mesgar**[†‡], **Krishnkant Swarnkar**[†], **Edwin Simpson**[†], **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
[‡]Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
[†]www.ukp.tu-darmstadt.de
[‡]www.aiphes.tu-darmstadt.de

Abstract

Visual modifications to text are often used to obfuscate offensive comments in social media (e.g., "!d10t") or as a writing style ("1337" in "leet speak"), among other scenarios. We consider this as a new type of adversarial attack in NLP, a setting to which humans are very robust, as our experiments with both simple and more difficult visual perturbations demonstrate. We investigate the impact of visual adversarial attacks on current NLP systems on character-, word-, and sentence-level tasks, showing that both neural and non-neural models are, in contrast to humans, extremely sensitive to such attacks, suffering performance decreases of up to 82%. We then explore three shielding methods—visual character embeddings, adversarial training, and rule-based recovery—which substantially improve the robustness of the models. However, the shielding methods still fall behind performances achieved in non-attack scenarios, which demonstrates the difficulty of dealing with visual attacks.

1 Introduction

For humans, visual similarity can play a decisive role for assessing the meaning of characters. Some evidence for these are: the frequent swapping of similar looking characters in Internet slang or abusive comments, creative trademark logos, and attack scenarios such as domain name spoofing (see examples in Table 1).

Internet slang writing style	n00b, w!k!p3d!4, 1337
Toxic comments	pi§§, ṇiggers, ḟucking
Trademark logos/artwork	SΛMSUNG
Domain name spoofing	http://wíkipedia.org

Table 1: Examples of text in which characters have been changed to visually similar ones.

Recently, some NLP systems have exploited visual features to capture visual relationships among characters in compositional writing systems such as Chinese or Korean (Liu et al., 2017). However, in more general cases, current neural NLP systems have no built-in notion of visual character similarity. Rather, they either treat characters as discrete units forming a word or they represent characters by randomly initialized embeddings and update them during training—typically in order to generate a character-based word representation that is robust to morphological variation or spelling mistakes (Ma and Hovy, 2016). Intriguingly, this marked distinction between human and machine processing can be exploited as a blind spot of NLP systems. For example, spammers might send malicious emails or post toxic comments to online discussion forums (Hosseini et al., 2017) by visually 'perturbing' the input text in such a way that it is still easily recoverable by humans.

The issue of exposing and addressing the weaknesses of deep learning models to *adversarial inputs*, i.e., perturbed versions of original input samples, has recently received considerable attention. For instance, Goodfellow et al. (2015) showed that small perturbations in the pixels of an image can mislead a neural classifier to predict an incorrect label for the image. In NLP, Jia and Liang (2017) inserted grammatically correct but semantically irrelevant paragraphs to stories to fool neural reading comprehension models. Singh et al. (2018) showed significant drops in the performance of neural models for question answering when using simple paraphrases of the original questions.

Unlike previous NLP attack scenarios, visual attacks, i.e., the exchange of characters in the input with visually similar alternatives, have the following 'advantages': 1) They do not require *any* linguistic knowledge beyond the character level, making the attacks straightforwardly applicable across languages, domains, and tasks. 2) They are al-

Proceedings of NAACL-HLT 2019, pages 1634–1647
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

legedly less damaging to human perception and understanding than, e.g., syntax errors or the insertion of negations (Hosseini et al., 2017). 3) They do not require knowledge of the attacked model's parameters or loss function (Ebrahimi et al., 2018).

In this work, we investigate to what extent recent state-of-the-art (SOTA) deep learning models are sensitive to visual attacks and explore various shielding techniques. Our contributions are:

- We introduce VIPER, a Visual Perturber that randomly replaces characters in the input with their visual nearest neighbors in a visual embedding space.
- We show that the performance of SOTA deep learning models substantially drops for various NLP tasks when attacked by VIPER. On individual tasks (e.g., Chunking) and attack scenarios, our observed drops are up to 82%.
- We show that, in contrast to NLP systems, humans are only mildly or not at all affected by visual perturbations.
- We explore three methods to shield from visual attacks, *viz.*, visual character embeddings, adversarial training (Goodfellow et al., 2015), and rule-based recovery. We quantify to which degree and in which circumstances these are helpful.

We point out that integrating visual knowledge with deep learning systems, as our visual character embeddings do, aims to make NLP models behave more like humans by taking cues directly from sensory information such as vision.[1]

2 Related Work

Our work connects to two strands of literature: adversarial attacks and visually informed character embeddings.

Adversarial Attacks are modifications to a classifier's input, that are designed to fool the system into making an incorrect decision, while the original meaning is still understood by a human observer. Different forms of attacks have been studied in NLP and computer vision (CV), including at a character, syntactic, semantic and, in CV, the visual level. Ebrahimi et al. (2018) propose a character flipping algorithm to generate adversarial examples and use it to trick a character-level neural classifier. They show that the accuracy decreases significantly after a few manipulations if certain characters are swapped. Their character flipping approach requires very strong knowledge in the form of the attacked networks' gradients in a so-called white box attack setup. Chen et al. (2018) find that reading comprehension systems often ignore important question terms, thus giving incorrect answers when these terms are replaced. Belinkov and Bisk (2018) show that neural machine translation systems break for all kinds of noise to which humans are robust, such as reordering characters in words, keyboard typos and spelling mistakes. Alzantot et al. (2018) replace words by synonyms to fool text classifiers. Iyyer et al. (2018) reorder sentences syntactically to generate adversarial examples.

In contrast to those related works which perform attacks on the character level, our attacks allow perturbation of any character in a word while potentially minimizing impairment for humans. For example, the strongest attack in Belinkov and Bisk (2018) is random shuffling of all characters, which is much more difficult to restore for humans.

To cope with adversarial attacks, *adversarial training* (Goodfellow et al., 2015) has been proposed as a standard remedy in which training data is augmented with data that is similar to the data used to attack the neural classifiers. Rodriguez and Rojas-Galeano (2018) propose simple rule-based corrections to address a limited number of attacks, including obfuscation (e.g., "idiots" to "!d10ts") and negation (e.g., "idiots" to "NOT idiots"). Most other approaches have been explored in the context of CV, such as adding a stability objective during training (Zheng et al., 2016) and distillation (Papernot et al., 2016). However, methods to increase the robustness in CV have been shown to be less effective against more sophisticated attacks (Carlini and Wagner, 2017).

Visual Character Embeddings were originally proposed to address large character vocabularies in 'compositional' languages like Chinese and Japanese. Shimada et al. (2016) and Dai and Cai (2017) employ a convolutional autoencoder to generate image-based character embeddings (ICE) for Japanese and Chinese text and show improvement on author and publisher identification tasks. Similarly, Liu et al. (2017) create ICEs from a CNN and show that ICEs carry more semantic content and are more suitable for rare characters. However,

---

[1] Code and data available from `https://github.com/UKPLab/naacl2019-like-humans-visual-attacks`

existing work on visual character embeddings has not used visual information to attack NLP systems or to them.

3 Approach

To investigate the effects of visual attacks and propose methods for shielding, we introduce 1) a visual text perturber, 2) three character embedding spaces, and 3) methods for obtaining word embeddings from character embeddings, used as input representations in some of our experiments.

3.1 Text perturbations

Our visual perturber VIPER disturbs an input text in such a way that (ideally) it is still readable by humans but causes NLP systems to fail blatantly. We parametrize VIPER by a probability p and a character embedding space, CES:[2] For each character c in the input text a flip decision is made (i.i.d. Bernoulli distributed with probability p), and if a replacement takes place, one of up to 20 nearest neighbors in the CES is chosen.[3] Thus, we denote VIPER as taking two arguments:

$$\mathsf{VIPER} = \mathsf{VIPER}(p, \mathrm{CES}).$$

Note that VIPER is a black-box attacker as it does not require any knowledge of the attacked system. It would also be possible to design a more intelligent perturber that only disturbs content words (or "hot" words), similar to Ebrahimi et al. (2018), but this would increase the difficulty for realizing VIPER as a black-box attacker because different types of hot words may be relevant for different tasks.

3.2 Character Embeddings

We consider three different character embedding spaces. The first is continuous, assigning each character a dense 576 dimensional representation, which allows, e.g., for computing cosine similarities between any two characters as well as nearest neighbors for each input character. The other two are discrete and merely used as arguments to VIPER. Thus, they are only required to specify nearest neighbors for standard input characters. For them, each character c in a selected range (e.g., standard English alphabet a-zA-Z) is assigned a set of nearest neighbors, and all nearest neighbors are equidistant to c. All three CES carry visual information, i.e., nearest neighbors are visually similar to the character in question. For practical reasons, we limit all our perturbations to the first 30k Unicode characters throughout.

Image-based character embedding space (ICES) provides a continuous image-based character embedding (ICE) for each Unicode character. We retrieve a 24×24 image representation of the character (using Python's PIL library), then stack the rows of this matrix (with entries between 0 and 255) to form a $24 \cdot 24 = 576$ dimensional embedding vector.

Description-based character embedding space (DCES) is based on the textual descriptions of Unicode characters. We first obtain descriptions of each character from the Unicode 11.0.0 final names list (e.g., LATIN SMALL LETTER A for the character 'a'). Then we determine a set of nearest neighbors by choosing all characters whose descriptions refer to the same letter in the same case, e.g., an alternative to LATIN SMALL LETTER A is LATIN SMALL LETTER A WITH GRAVE as it contains the keywords SMALL and A.

Easy character embedding space (ECES) provides manually selected simple visual perturbations. It contains exactly one nearest neighbor for each of the 52 characters a-zA-Z, chosen as a diacritic below or above a character, such as $\hat{c}$ for the character c.

Differences between the CESs The three embedding spaces play different roles in our experiments. We use ICES as character representations in deep learning systems. DCES and ECES are used as input to VIPER to perturb our test data.[4] ECES models a 'minimal perturbance with maximal impact' scenario: we assume that ECES perturbations do not or only minimally affect human perception but may still have a large impact upon NLP systems. Indeed, we could have chosen an even simpler embedding space, e.g., by considering visually identical characters in different alphabets, such as the Cyrillic 'a' (Unicode 1072) for a Latin 'a' (Unicode 97). DCES is a more difficult

---

[2] CES may be any 'embedding space' that can be used to identify the nearest neighbors of characters.

[3] The probability of choosing one of the 20 neighbors of c is proportional to its distance to c.

[4] We do not attack with ICES because we also shield with ICES and this would be a (very unrealistic) white box defense scenario. Besides, ICES is also more difficult to restore for humans (see below), making it less desirable for an attacker.

test-bed designed for evaluating our approaches under more realistic conditions with more varied and stronger attacks.

Table 2 exemplifies the differences between ICES, DCES, and ECES by comparing the nearest neighbors of a given character. As expected, ICES contains neighbors of characters which are merely visually similar without representing the same underlying character (such as Λ as a neighbor of A, or ı as a neighbor of i). In contrast, DCES sometimes has neighbors with considerable visual dissimilarity to the original character such as Cyrillic small letter i (и) which rather resembles a mirror-inverted n. The overlap between ICES and DCES is modest: out of 20 neighbors, a character has on average only four to five common neighbors in ICES and DCES.

3.3 Word Embeddings

Most neural NLP architectures encode text either on a character or word level. For the latter, word embeddings are needed. In this work, we use the ELMo architecture (Peters et al., 2018) to obtain (contextualized) word embeddings based on characters, i.e., there exists no fixed vocabulary and there will be no (word-level) out-of-vocabulary issues due to perturbation. In the following, we outline our ELMo variant and a visual extension that includes visual signals from the input characters.

SELMo: ELMo as proposed by Peters et al. (2018) first retrieves embeddings for every character in the input, which are learned as part of the network. ELMo then infers non-contextualized word embeddings by applying CNNs over all character embeddings in a word. Two layers of a deep bidirectional language model further process the word embeddings in their local sentential context and output contextualized word embeddings.

We slightly extend ELMo to include character embeddings for the first 30k Unicode characters (instead of the default 256). We call this variant SELMo ("Standard ELMo"). It is worth pointing out that the learned character embeddings of SELMo carry almost no visual information, as illustrated in Table 2. That is, except for a few very standard cases, nearest neighbors of characters do not visually resemble the orginal characters, even when trained on the 1 billion word benchmark (Chelba et al., 2013).[5]

VELMo: To obtain a visually informed variant of ELMo, we replace learned character embeddings with the ICEs and keep the character embeddings fixed during training. This means that during training, the ELMo model learns to utilize visual features of the input, thus potentially being more robust against visual attacks. We call this variant VELMo ("Visually-informed ELMo").

To keep training times of SELMo and VELMo feasible, we use an output dimensionality of 512 instead of the original ELMo's 1024d output. Our detailed hyperparameter setup is given in §A.1.

4 Human annotation experiment

We asked 6 human annotators, university employees and students with native or near-native English language skills, to recover the original underlying English sentences given some perturbed text (data taken from the POS tagging and Chunking tasks, see Table 4). We considered different conditions:

(i) `clean`: $\mathsf{VIPER}(0,\_)$, i.e., no perturbation;
(ii) $\mathsf{VIPER}(p,\text{ICES})$ for $p = 0.2, 0.4, 0.6, 0.8$;
(iii) $\mathsf{VIPER}(p,\text{DCES})$ for $p = 0.2, 0.4, 0.6, 0.8$;
(iv) `easy`: $\mathsf{VIPER}(p,\text{ECES})$ for $p = 0.4, 0.8$.

For each condition, we used 60-120 sentences, where at most 20 sentences of one condition were given to an annotator. Examples of selected conditions are shown in Table 3. Our rationale for including this recovery task is to test robustness of human perception under (our) visual perturbations. We focus on recovery instead of an extrinsic task such as POS because the latter would have required expert/trained annotators.

We evaluate by measuring the normalized edit distance between the recovered sentence and the underlying original, averaged over all sequence pairs and all human annotators. We normalize by the maximum lengths of the two sequences. In our case, this metric can be interpreted as the fraction of characters that have been, on average, wrongly recovered by human annotators. We refer to the metric as "error rate".

Results are shown in Figure 1. In `easy`, there is almost no difference between perturbation levels $p = 0.4$ and $p = 0.8$, so we merge the two conditions.

Humans make copy mistakes even when the input is not perturbed, as evidenced by a positive

---

[5] We believe SELMo nearest neighbors are more likely to be Chinese/Japanese/Korean (CJK) characters because these nearest neighbors are largely random and there are far more CJK characters in our subset of Unicode.

Input	ICES	DCES	ECES	SELMo
e **i** **A**	e ẹ ė ȩ ϵ ē ę ḛ ë ë i ¡ ị I l l ḷ l ḭ ḷ A A Ạ Aι Ḁ Ἀ Ἁ Ἀι Λ Λ	ĕ ɝ ə ɜ ē ĕ ȩ ɛ è ë ĭ ɨ̀ ȉ i ī ǐ ı ɨ̆ í ĩ Ā Ă Ǟ A Å Ǎ Ȁ Ⱥ Á Ä	ê î Â	é 跣 ○ 徛 ꞇ ꞇ ç ↙ ㈎ 坪 í ⁞ 嚫 檯 爗 篖 炾 媸 腥 > 邃 榜 溷 翏 呭 8. 敲 漸 ˛ 唋 用

Table 2: Ten nearest neighbors in our different character spaces. ‘SELMo’ refers to the nearest neighbors of the trained character embeddings in SELMo.

Condition	**Sentences** (Perturbed / Original)
`easy`-0.8	Ḿř. Ĉôḟfêê iŝ â ṕřôfêŝsoř âẗ Ĉôḷǔḿḃîâ Ĺâẘ Ŝcĥôôḷ . Mr. Coffee is a professor at Columbia Law School .
ICES-0.6	Tṇẽ sẖutḋown a|fëçṱə Ə,0ð0 wòřkḗrs ḁng ʁllị ȼũt óũṱpuȴ ɓỳ apouṭ 4‚3Ẓ0 câṛṣ : The shutdown affects 3,000 workers and will cut output by about 4,320 cars .
DCES-0.8	Tʰe śţōcƙ ɽeco(v)ɝʶèd ṡøɯəẅḫât ᵗö ƫịṇịṣḣ 1 1/4 ˡòwɇʶ áƭ 26 1/4 . The stock recovered somewhat to finish 1 1/4 lower at 26 1/4 .

Table 3: Examples of perturbed sentences and underlying originals.

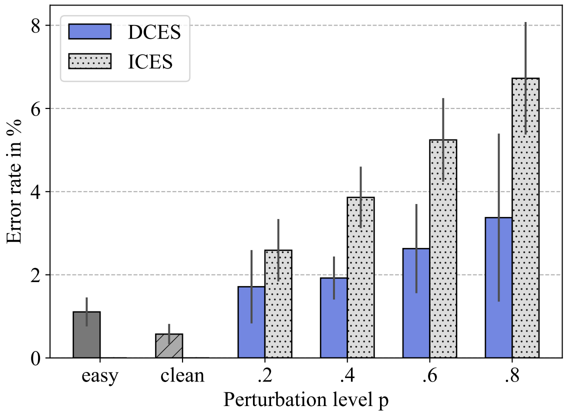


Figure 1: Human annotation experiment. Error bars indicate std. across annotators. For `easy`, we merge the cases $p = 0.4/0.8$.

error rate in `clean`. Such mistakes are typically misspellings or the wrong type of quotation marks (” vs. “). We observe a slightly higher error rate in `easy` than in `clean`. However, on average 75% of all sentences are (exactly) correctly recovered in `easy` while this number is lower (72.5%) in `clean`. By chance, `clean` contains fewer sentences with quotation marks than `easy`, for which a copy mistake was more likely. This may explain `easy`’s higher error rate.

As we increase the perturbation level, the error rate increases consistently for DCES/ICES. It is noteworthy that DCES perturbations are easier to parse for humans than ICES perturbations. We think this is because DCES perturbations always retain a variant of the same character, while ICES may also disturb one character to another character (such as *h* to *b*). Another explanation is that ICES, unlike DCES and ECES, also disturbs numbers and punctuation. Numbers, especially, are more difficult to recover. However, even at 80% disturbance level, humans can, on average, correctly recover at least 93% of all characters in the input text in all conditions.

In summary, humans appear very good at understanding visual perturbations, and are almost perfectly robust to the easy perturbations of ECES. Since adversarial attacks should have minimal impact on humans (Szegedy et al., 2014), the good performance of humans especially on ECES and DCES makes these two spaces ideal candidates for attacks on NLP systems.

5 Computational Experiments

We now evaluate the capabilities of SOTA neural network models to deal with visual attacks in four extrinsic evaluation tasks described in §5.1 and illustrated in Table 4. Hyperparameters of all our models are given in §A.2. We first examine the robustness of all architectures to visual perturbations in §5.2 and then evaluate different shielding approaches in §5.3.

5.1 Tasks

G2P: As our first task, we consider the character-level task of grapheme-to-phoneme (G2P) conversion. It consists of transcribing a character input stream into a phonetic representation. As our dataset, we choose the Combilex pronunciation dataset of American English (Richmond et al.,

Task	**Task Type**	**Input**	**Target / Label(s)**	**Train/Dev/Test**
G2P	char-lvl	ẉṙḛŧchȩđlȳ	r E < @ d 5 i	5K/1K/1K
POS	word-lvl	… exᴛȇ’nɖíng itŝ contraᶝẗ …	… VBG PRP NN …	212K/44K/47K
Chunking	word-lvl	… exᴛȇ’nɖíng itŝ contraᶝẗ …	… B-VP B-NP I-NP …	212K/44K/47K
Toxic Comments	sent-lvl	Ḟǔĉǩ ôḟḟ , yoǔ âňtî - ŝeḿîẗîĉ ĉǔňẗ .	toxic, obscene, insult	149K/10K/64K

Table 4: NLP tasks considered in this work, along with (perturbed) examples and data split statistics.

2009). We frame G2P as a sequence tagging task. To do so, we first hard-align input and output sequences using a 1-0,1-1,1-2 alignment scheme (Schnober et al., 2016) in which an input character is matched with zero, one, or two output characters. Once this preprocessing is done, input and output sequences have equal lengths and we can apply a standard BiLSTM on character-level to the aligned sequences (Reimers and Gurevych, 2017).

POS & Chunking: We consider two word-level tasks. POS tagging associates each token with its corresponding word class (e.g., *noun, adjective, verb*). Chunking groups words into syntactic chunks such as noun and verb phrases (NP and VP), assigning a unique tag to each word, which encodes the position and type of the syntactic constituent, e.g., begin-noun-phrase (B-NP). We use the training, dev and test splits provided by the CoNLL-2000 shared task (Sang and Buchholz, 2000) and use the same BiLSTM architecture as above with SELMo/VELMo embeddings.

Toxic comment (TC) classification: A very realistic use case for adversarial attacks is the toxic comment classification task. One could easily think of a scenario where a person with malicious intent explicitly aims to fool automated methods for detecting toxic comments or insults by obfuscating text with non-standard characters that are still human-readable. We conduct experiments on the TC classification task provided by Kaggle.[6] It is a multi-label sentence classification task with six classes, i.e., *toxic*, *severe toxic*, *obscene*, *threat*, *insult*, *identity hate*. We use average SELMo/VELMo embeddings as input to an MLP.

5.2 VIPER attacks

In Figure 2, we plot how various SOTA systems degrade as we perturb the test data using DCES. We do not only include our own systems, but also existing SOTA models: Marmot (Müller et al., 2013) and Stanford POS tagger (SPT) (Manning et al., 2014). Marmot is a feature-based POS tagger and trained on our data splits. SPT is a bi-directional dependency network tagger that mostly employs lexical features. For SPT, we used the pretrained English model provided by the toolkit. Further, we include a FastText TC classifier which has achieved SOTA performance.[7] We additionally experiment with word level dependency embeddings for POS tagging and TC classification (Komninos and Manandhar, 2016).

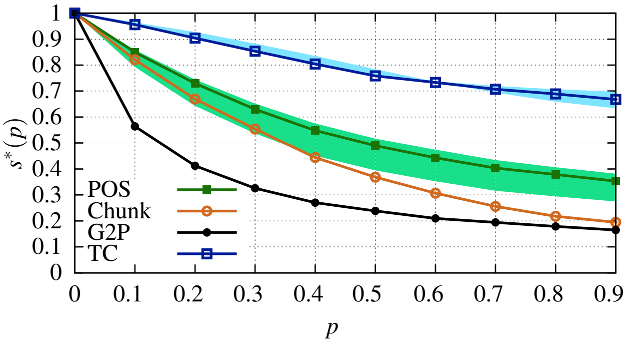


Figure 2: Degradation of SOTA systems for different perturbation levels when attacked by VIPER(p,DCES). The colored regions show how the performance of other SOTA systems relate to ours (i.e., they all suffer from similar degradation).

To compare the performance of different tasks, Figure 2 shows scores computed by:

$$s^*(p) = \frac{s(p)}{s(0)},$$

where p is the perturbation level and $s(p)$ is the score for each task at p, measured in edit distance for G2P, accuracy for POS tagging, micro-F1 for chunking, and AUCROC for TC classification. We invert the scores g of G2P by $1/g$ since lower scores are better for edit distance. Thus, $s^*(0)$ is always 1 and $s^*(p)$ is the relative performance compared to the clean case of no perturbations.

We see that all systems degrade considerably. For example, all three POS taggers have a performance of below 60% of the clean score when 40%

---

[6] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/

[7] https://www.kaggle.com/yekenot/pooled-gru-fasttext

of the input characters are disturbed. Chunking degrades even more strongly, and G2P has the highest drop: 10% perturbation level causes a 40% performance deterioration. This may be because G2P is a character-level task and the perturbation of a single character is analogous to perturbing a complete word in the word-level tasks. Finally, TC classification degrades least, i.e., only at $p = 0.9$ do we see a degradation of 30% relative to the clean score. These results appear to suggest that character-level tasks suffer the most from our VIPER attacks and sentence-level tasks the least. However, it is worthwhile pointing out that lower-bounds for individual tasks may depend on the evaluation metric (e.g., AUCROC always yields 0.5 for majority class voting) as well as task-specific idiosyncrasies such as the size of the label space.

We note that the degradation curves look virtually identical for both DCES or ECES perturbations (given in §A.3). This is in stark contrast to human performance, where ECES was much easier to parse than DCES, indicating the discrepancies between human and machine text processing.

5.3 Shielding

We study four forms of shielding against VIPER attacks: adversarial training (**AT**), visual character embeddings (**CE**), **AT+CE**, and rule-based recovery (**RBR**). For **AT**, we include visually perturbed data at train time. We do not augment the training data, but replace clean examples using VIPER in the same way as for the test data. Based on preliminary experiments with the G2P task, we apply VIPER to the training data using $p_{\text{train}} = 0.2$. Higher levels of p_{train} did not appear to improve performance. For **CE**, we use fixed ICEs, either fed directly into a model (G2P) or via VELMo (all other tasks). For **AT+CE**, we combine adversarial training with visual embeddings. Finally, for **RBR**, we replace each non-standard character in the input stream with its nearest standard neighbor in ICES, where we define the standard character set as a-zA-Z plus punctuation.

Rather than absolute scores, we report differences between the scores in one of the shielding treatments and original scores:

$$\Delta_\tau := \sigma^*(p) - s^*(p), \quad \sigma^*(p) := \sigma(p)/s(0)$$

where $\sigma(p)$ is the score for each task using a form of shielding. The value Δ_τ denotes the improvement of the scores from shielding method τ over the original scores without shielding. We normalize $\sigma(p)$ by the score $s(0)$ of the systems without shielding on clean data. We also note that our test perturbations are unseen during training for DCES; for ECES this would not make sense, because each character has only one nearest neighbor. In the following, we report results mostly for DCES and show the ECES results in §A.3. We highlight marked differences between the results, however.

All tasks typically profit considerably from **AT** (Figure 3 left). Chunking scores improve most; e.g., at $p = 0.5$, σ^* is 17 percent points (pp) higher than s^*. AT does not help for G2P in the DCES setting but it does help for ECES (see §A.3), where test perturbations may have been seen during training. We conjecture that AT makes systems generally aware that the input can be broken in some way and forces them to shield against such situations, an effect similar to dropout. However, such shielding appears more difficult in character-level tasks, where a missing token is considerably more damaging than in word- or sentence-level tasks.

In Figure 3 (right), we observe that **CE** helps a lot for G2P, but much less particularly for POS and Chunking. We believe that for G2P, the visual character embeddings restore part of the input and thus have considerable effect. It is surprising, however, that visual embeddings have no positive effect for both word-level tasks, and instead lead to small deteriorations. A possible explanation is that, as the character embeddings are fed into the ELMo architecture, their effect is dampened. Indeed, we performed a sanity check (see §A.5) to test how (cosine) similar a word or sentence w is to a perturbed version w' of w under both SELMo and VELMo. We found that VELMo assigns consistently better similarities but the overall gap is small.

We observe that the combined effect of AT and CE (**AT+CE**, Figure 4 left) is always substantially better than either of the two alone. For instance, at $p = 0.5$, POS improves by about 20pp, while AT alone had an effect of only 12pp and the effect of CE was even negative. Thus, it appears that AT is able to kick-start the benefits of CE, especially in the case when they alone are not effective.

RBR is excellent for ECES (see §A.3). It has a small negative effect on clean data, meaning that there is some foreign material in English texts which gets corrupted by RBR, but for any $p > 0$ the performance under RBR is almost on the level of $p = 0$ for ECES. RBR is also consistently better

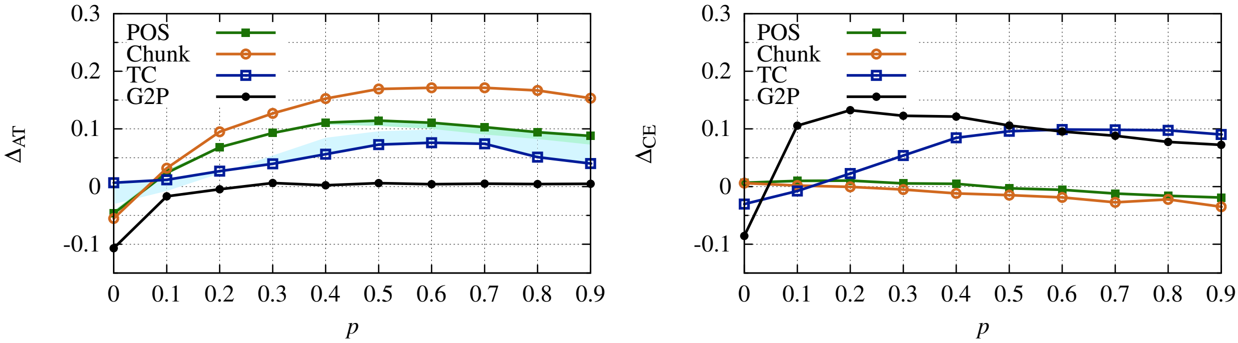


Figure 3: AT (with ICES replacements) and CE tested on DCES perturbed data. The colored regions show AT (with random replacements).

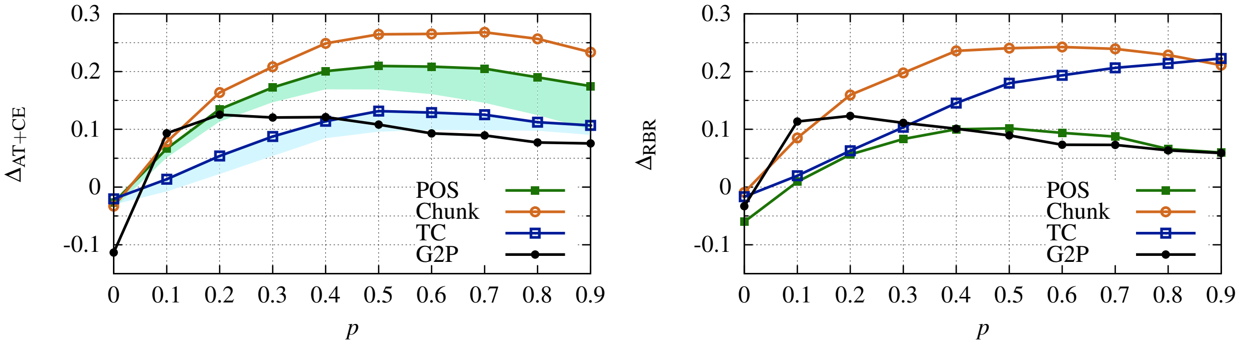


Figure 4: AT+CE (with ICES replacements) and RBR on DCES perturbed data. The colored regions show AT (with random replacements).

than CE, even though both depend on ICES: CE in a 'soft' way and RBR in a 'hard' way. Our best explanation is that RBR is analogous to 'machine translating' a foreign text into English and then applying a trained classifier, while CE is analogous to a direct transfer approach (McDonald et al., 2011) which trains in one domain and is then applied to another. This causes a form of domain shift to which neural nets are quite vulnerable (Ruder and Plank, 2018; Eger et al., 2018a).

For DCES, RBR is outperformed by AT+CE, which better mitigates the domain shift than CE, except for TC.

We note that even with all our shielding approaches, the performance of the shielded systems is still considerably below the performance on clean data at some perturbation levels. E.g., at $p = 0.9$, AT+CE shielded Chunking has a score of less than 60% of the clean performance. While it may be partially due to our character embeddings not being optimal (i.e., they assign low similarity to major and minor variants of the same letter such as ˢ and s, which could be improved by size-invariant CNNs), a main reason for this could be the domain-shift induced by the perturbations, for which even AT cannot always help when attacks are unseen during training. This is another major distinction between human and machine processing.

6 Discussion

Random AT. We discussed that the effect of AT may be similar to dropout. If so, AT with random rather than visual character replacements should be similarly effective. Indeed, the graphs in Figures 3 and 4 show that injecting random noise generally improves the robustness, but not to the level of visually informed AT (exemplarily for POS/TC).

Error analysis. We analyze the cases in which our perturbation with VIPER changes the prediction to a wrong class. We perform our analysis for TC as it represents a real-world attack scenario. We define the *toxicity level* (TL) of a comment x for a class $y \in \{1,\ldots,6\}$ and model θ as $\text{TL}(x) = \sum_y \theta(x,y)$, e.g., a comment x which has been classified as *insult* (probability 0.8) and *obscene* (probability 0.7) gets a TL of 1.5. We consider a successful attack to decrease TL after perturbation. VIPER with DCES and $p = 0.1$ achieves a success rate of 24.1%—i.e., roughly one fourth of the *toxic* comments receive a lower TL. In contrast, the

Perturbation (CES)	Perturbed Text	Toxicity Level	No Shielding	AT	CE	AT+CE	RBR
clean	u are a gigantic faggot	4.00	3.29	3.16	2.93	3.11	3.57
VIPER(p, ECES)	u are â gigantic faggoẗ	4.00	0.11	1.05	1.37	2.05	3.57
VIPER(p, DCES)	u are a ɠigantic faggoƫ	4.00	0.82	1.18	3.36	3.16	1.69
clean	he is also a faggot .	4.00	3.26	3.68	2.81	2.14	3.79
VIPER(p, ECES)	ĥe is also a faggot .	4.00	3.39	3.70	2.64	2.23	3.79
VIPER(p, DCES)	he is alşo a fagᶢof .	4.00	0.48	1.48	1.55	1.78	0.40

Table 5: Two examples of toxic/non-toxic comments that show the effects of the different shielding methods. We report the averaged sum over the six toxicity classes, e.g., 4.00 is equal to a positive example in four classes. $p = 0.1$.

impact on *non-toxic* comments is small—TL increased in only 3.2% of the cases.

Table 5 shows sample comments and their TL for different shielding and perturbation methods. As can be seen, perturbing specific words (*hot words* for TC) substantially reduces the TL score of a non-shielded approach (e.g., from 3.29 to 0.11), while perturbing 'non-hot' words like 'he' has little effect. The shielding approaches help in these show-cased examples to various degrees and the shielding with AT+CE is more robust to stronger attacks (higher visual dissimilarity) than RBR.

This illustrates that a malicious attacker may aim to increase the success rate of an attack by only perturbing offensive words (in the TC task). To test whether VIPER benefits from perturbing such hot words, we manually compiled a list of 20 hand-selected offensive words (see §A.6) which we believe are indicators of toxic comments. We then analyzed how often a perturbation of a word from this list co-occurs with a successful attack. We observe that in 55% of successful attacks, a word from our list was among the perturbed words of the comment. As our list is only a small subset of all possible offensive words, the perturbation of hot words may have an even stronger effect.

7 Conclusion

In this work, we considered visual modifications to text as a new type of adversarial attack in NLP and we showed that humans are able to reliably recover visually perturbed text. In a number of experiments on character-, word-, and sentence-level, we highlighted the fundamental differences between humans and state-of-the-art NLP systems, which sometimes blatantly fail under visual attack, showing that visual adversarial attacks can have maximum impact. This calls for models that have richer biases than current paradigm types do, which would allow them to bridge the gaps in information processing between humans and machines. We have explored one such bias, visual encoding, but our results suggest that further work on such shielding is necessary in the future.

Our work is also important for system builders, such as of toxic comment detection models deployed by, e.g., Facebook and Twitter, who regularly face visual attacks, and who might face even more such attacks once visual character perturbations are easier to insert than via the keyboard. From the opposite viewpoint, VIPER may help users retain privacy in online engagements and when trying to avoid censorship (Hiruncharoenvate et al., 2015) by suggesting visually similar spellings of words.

Finally, our work shows that the 'brittleness' (Belinkov and Bisk, 2018) of NLP extends beyond MT and beyond word reordering or replacements, a recognition that we hope inspires others to investigate more ubiquitous shielding techniques.

Acknowledgments

We thank the reviewers for helpful feedback. This work has been supported by the German Research Foundation (DFG) funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1), the DFG-funded projects QA-EduInf (GU 798/18-1, RI 803/12-1), DIP (GU 798/17-1), the German Federal Ministry of Education and Research (BMBF) under the promotional references 16DHL1040 (FAMULUS) and by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich- Ökonomischer Exzellenz" (LOEWE) as part of the a! - automated language instruction project (No. 521/17-03). We gratefully acknowledge support of NVIDIA Corp. with the donation of the Tesla K40 GPU used for this research. Calculations for this research were also conducted on the Lichtenberg high performance cluster of Technical University Darmstadt.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2587–2597.

Falcon Dai and Zheng Cai. 2017. Glyph-aware embedding of chinese characters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, September 7, 2017*, pages 64–69.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam. *ICLR Workshop*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 31–36.

Steffen Eger, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2018a. Cross-lingual argumentation mining: Machine translation (and a bit of projection) is all you need! In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Steffen Eger, Paul Youssef, and Iryna Gurevych. 2018b. Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4415–4424. Association for Computational Linguistics.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Chaya Hiruncharoenvate, Zhiyuan Lin, and Eric Gilbert. 2015. Algorithmically bypassing censorship on sina weibo with nondeterministic homophone substitutions. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 150–158.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2059–2068.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ryan T. McDonald, Slav Petrov, and Keith B. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 62–72.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy*, pages 582–597.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Korin Richmond, Robert A. J. Clark, and Susan Fitt. 2009. Robust LTS rules with the combilex speech technology lexicon. In *INTERSPEECH*, pages 1295–1298. ISCA.

Nestor Rodriguez and Sergio Rojas-Galeano. 2018. Shielding google's language toxicity model against adversarial attacks. *arXiv preprint arXiv:1801.01828*.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning, CoNLL 2000, and the Second Learning Language in Logic Workshop, LLL 2000, Held in cooperation with ICGI-2000, Lisbon, Portugal, September 13-14, 2000*, pages 127–132.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714. The COLING 2016 Organizing Committee.

Daiki Shimada, Ryunosuke Kotani, and Hitoshi Iyatomi. 2016. Document classification through image-based character embedding and wildcard training. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 3922–3927.

Sameer Singh, Carlos Guestrin, and Marco Túlio Ribeiro. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 856–865.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 4480–4488.

A Appendices

A.1 SELMo and VELMo Hyperparameters

Differences to the original ELMo as in (Peters et al., 2018) are:

- We exclude CNN filters of size 6 and 7.
- The maximum characters per token is 20 (instead of 50).
- The LSTM dimensionality is 2048 (instead of 4096).
- Our projection dimensionality is 256 (instead of 512).
- We train the models for 5 epochs (instead of training it for 10 epochs).

A.2 Task Settings

G2P: We randomly draw most hyperparameters for the sequence tagging BiLSTM architecture (Reimers and Gurevych, 2017) that we use for G2P, e.g., those concerning the optimizer used, learning and dropout rates. We hand-set the number of hidden recurrent layers to 1, and its size to 50. We use early stopping and set the maximum number of epochs for training to 50. As our dataset, we choose the Combilex pronunciation dataset of American English (Richmond et al., 2009). We randomly draw our train/dev/test splits from the whole corpus. Examples and split sizes are given in Table 4. We report **edit distance** between desired pronunciations and predicted pronunciations as metric. We report the edit distance averaged across all 1k test strings, averaged over 5 random initializations of all weight matrices.

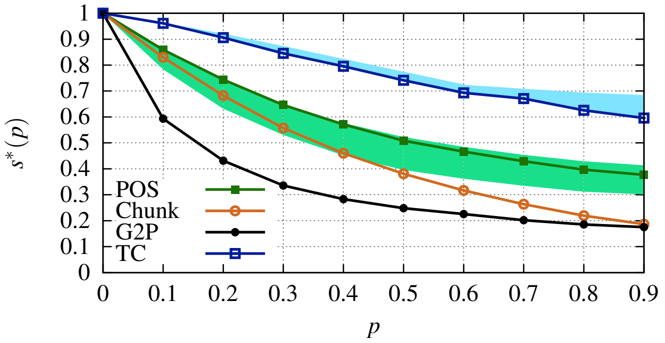


Figure 5: Degradation of SOTA systems for different perturbation levels when attacked by VIPER(p,ECES). The colored regions show how the performance of other SOTA systems relate to ours.

POS & Chunking: We use the training, dev and test splits provided by the CoNLL-2000 shared task (Sang and Buchholz, 2000) for both tasks. We have used a readily available LSTM-CRF sequence tagger (Reimers and Gurevych, 2017) as above, but adapted for ELMo-type input embeddings, with default hyperparameter settings. We run each experimental setting 10 times, and report the average, measured as **accuracy** for POS and **micro-F1** for Chunking. For both tasks, we have used two stacked BiLSTM-layers with 100 recurrent units and dropout probability of 0.5. Mini-batch size is chosen as 32. We used gradient clipping and early stopping to prevent overfitting. Adam is used as the optimizer.

Toxic comment classification: We use the train and test splits provided by the task organizers. For tuning our models, we split off a development set of 10k sentences from the training data. As in POS&Chunking, we train models on clean and perturbed data using SELMo and VELMo representations. We obtain the sentence representation for a single sentence by averaging ELMo word embeddings over all tokens. We then train an MLP which we tune separately for each SELMo and VELMo embedding using random grid search with 100 different configurations. We tune the following hyperparameters separately for each hidden layer: the depth of the neural network, i.e., one, two, or three hidden layers; the size of the hidden layer (128, 256, 512, or 1024); the amount of dropout after each layer (0.1 - 0.5); the activation functions for each hidden layer (*tanh*, *sigmoid*, or *relu*) (Eger et al., 2018b). Both models are trained for 100 epochs with an early stopping after 10 epochs without any substantial improvement and use Nesterov-accelerated Adaptive Moment Estimation (Dozat, 2016) for optimization. Model performance is measured as proposed by the task organizers using the *area under the receiver operating characteristics curve* (**AUCROC**).

A.3 ECES Results

Figure 5 shows how the performances of various SOTA systems degrade on ECES settings. Figures 6 and 7 show our shielding results on ECES perturbed data. As indicated in the main paper, RBR is able to recover ECES data almost perfectly regardless of the perturbation level. This is because ECES only perturbs with a single nearest neighbor, which in addition is visually extremely similar to the underlying original, and thus, RBR can almost completely undo the perturbations.

A.4 AT+CE *vs.* AT or CE

Figure 8 compares AT+CE against either AT or CE. For this, we compute the difference in the performance decrease normalized by the test performance on the clean data. As can be seen, AT+CE almost constantly outperforms either one of both, especially on word- and sentence-level tasks.

A.5 Intrinsic Evaluation

To analyze the differences between VELMo and SELMo, we investigate whether the models learn similar word embeddings for a clean sentence and its visually perturbed counterpart. We compare sentence embeddings which we obtain by averaging over the SELMo or VELMo word embeddings of a sentence (clean or perturbed).

Setup Given a sentence embedding σ of a clean sentence and an embedding σ' of its visually perturbed counterpart, obtained by either averaging over VELMo (indicated by subscript v) or SELMo (indicated by subscript s) word embeddings, we test if the condition

$$\cos(\sigma_v, \sigma'_v) > \cos(\sigma_s, \sigma'_s) \qquad (1)$$

is met (with cos being the cosine similarity).

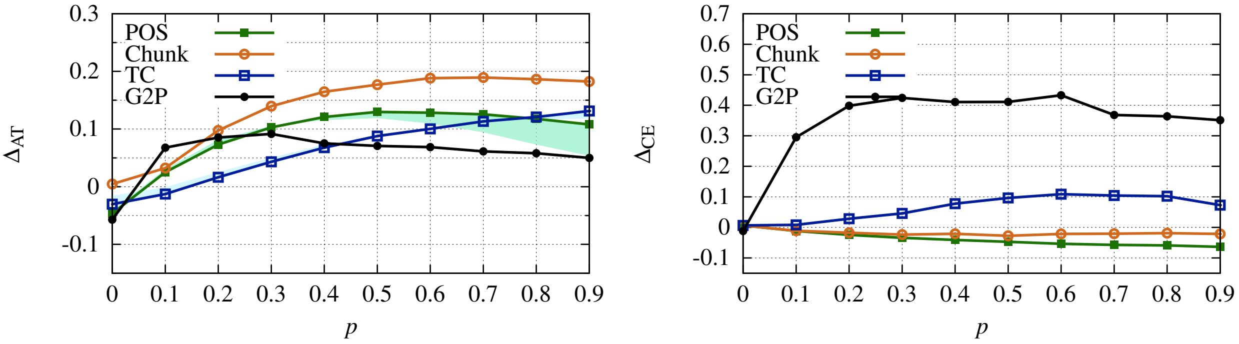


Figure 6: AT (ICES) and CE tested on ECES perturbed data. The colored regions show AT (Random). The y axis of Δ_{AT} spans -0.15-0.3 for better visualization.

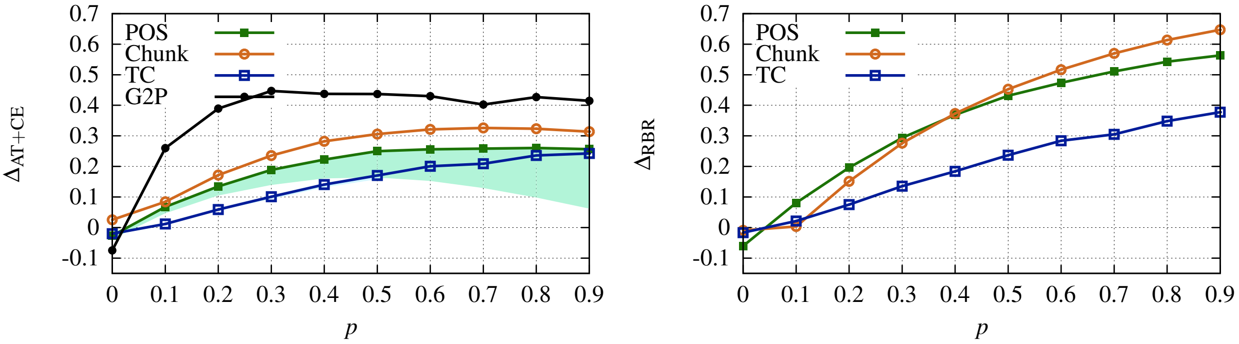


Figure 7: AT+CE (ICES) and RBR on ECES perturbed data. The colored regions show AT (Random).

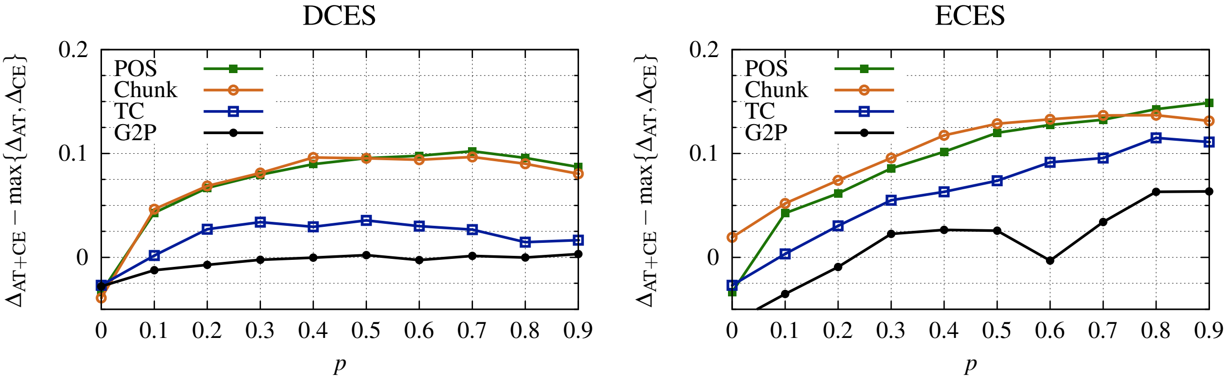


Figure 8: AT+CE (ICES) vs max(AT,CE) for DCES and ECES perturbed test data.

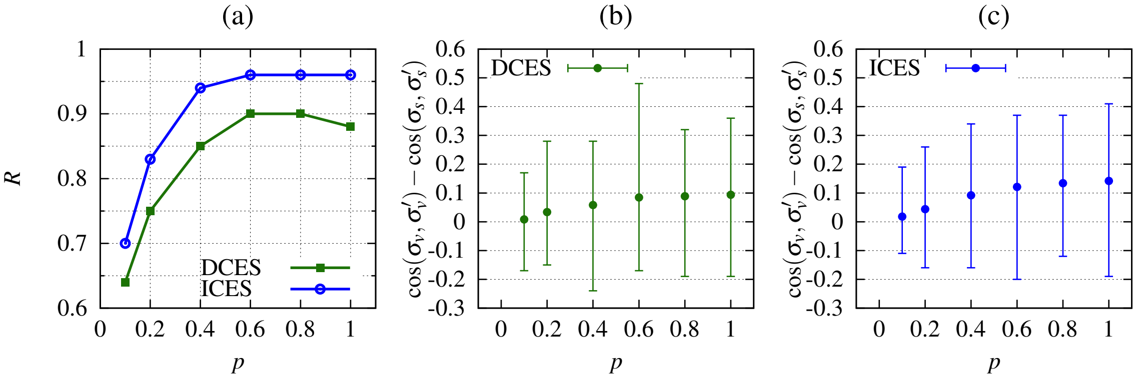


Figure 9: Results of the intrinsic evaluation. (a) shows the ratio of cases in which the condition in Eq. (1) is met (‘R’). (b) and (c) show the average difference of the cosine similarities when sentences are perturbed with (b) DCES and (c) ICES.

For our experiments, we randomly sample 1000 sentences from the Toxic Comments dataset (see §5.1) and perturb them with VIPER(p, CES) where CES $\in$ {ICES, DCES}. We then count the number N of cases in which the above condition is met with regards to the chosen CES and the value of p, and report the ratio $R = N/1000$.

Results The results are given in Figure 9. In Figure 9(a) we observe that the VELMo embeddings of a clean sentence and its perturbed counterpart are in many cases more similar than the ones of SELMo. For larger values of p, this ratio substantially increases from 70% to 95% (with ICES), which shows that VELMo is better suited to capture the similarity to the source sentence, especially in cases with a strong perturbation.

In Figures 9(b) and (c) we show the (mean) difference $\cos(\sigma_v, \sigma'_v) - \cos(\sigma_s, \sigma'_s)$. Here, we only observe a small positive effect in favor of VELMo, showing that the advantage of VELMo over SELMo is not substantial. We hypothesize that this is due to the contextual information which is utilized throughout the ELMo architecture, allowing SELMo to infer individual characters from the context of the word and the sentence. However, our results also show that the advantage of VELMo over SELMo is consistent.

Differences in similarities can also be affected by model training or the model architecture—e.g., in an extreme case a model could output the same embedding for every word/sentence. This would result in a ‘perfect’ cosine similarity, which would be advantageous in the previous experiment. Thus, we perform an additional experiment where we examine if

$$\cos(\sigma_v, \sigma'_v) > \cos(\sigma_v, \rho_v) \tag{2}$$

holds, where ρ is a randomly sampled sentence from the Toxic Comments dataset (with no perturbation). The same experiment is also performed for SELMo.

	DCES		**ICES**	
p	**VELMo**	**SELMo**	**VELMo**	**SELMo**
0.1	0.99	0.98	1.00	0.97
0.2	0.98	0.84	0.98	0.77
0.4	0.81	0.38	0.85	0.25
0.6	0.60	0.10	0.63	0.07
0.8	0.42	0.04	0.49	0.03
1.0	0.34	0.01	0.39	0.01

Table 6: Results of the intrinsic evaluation where we compare clean sentences to their perturbed counterparts as well as randomly chosen sentences. The numbers show the ratio of cases where clean sentences are more similar to their perturbed counterparts than the randomly chosen sentences.

The results in Table 6 show that for both models with $p = 0.1$ the original sentence is in 97–100% of the cases more similar to its perturbed counterpart than the randomly chosen sentence. As the perturbation probability increases, VELMo has a clear advantage over SELMo. E.g., if we perturb all characters in a sentence ($p = 1.0$), the SELMo embeddings of the perturbed sentence are in 1% of the cases more similar to the original sentence whereas this is the case for more than 34–39% for VELMo. Thus, VELMo embeddings better capture similarity between visually similar words.

A.6 List of Hand-selected Curse Words

arrogant, ass, bastard, bitch, dick, die, fag, fat, fuck, gay, hate, idiot, jerk, kill, nigg*[8], shit, stupid, suck, troll, ugly

---

[8] Due to several variations in the data, we match against *nigg** instead of the whole word.

Something's Brewing!
Early Prediction of Controversy-causing Posts from Discussion Features

Jack Hessel
Cornell University
`jhessel@cs.cornell.edu`

Lillian Lee
Cornell University
`llee@cs.cornell.edu`

Abstract

Controversial posts are those that split the preferences of a community, receiving both significant positive and significant negative feedback. Our inclusion of the word "community" here is deliberate: what is controversial to some audiences may not be so to others. Using data from several different communities on `reddit.com`, we predict the ultimate controversiality of posts, leveraging features drawn from both the textual content and the tree structure of the early comments that initiate the discussion. We find that even when only a handful of comments are available, e.g., the first 5 comments made within 15 minutes of the original post, discussion features often add predictive capacity to strong content-and-rate only baselines. Additional experiments on domain transfer suggest that conversation-structure features often generalize to other communities better than conversation-content features do.

1 Introduction

Controversial content — that which attracts both positive and negative feedback — is not necessarily a bad thing; for instance, bringing up a point that warrants spirited debate can improve community health.[1] But regardless of the nature of the controversy, detecting potentially controversial content can be useful for both community members and community moderators. Ordinary users, and in particular new users, might appreciate being warned that they need to add more nuance or qualification to their earlier posts.[2] Moderators could be alerted that the discussion ensuing from some content might need monitoring. Alternately, they could draw community attention to issues possibly needing resolution: indeed, some sites already provide explicit sorting by controversy.

We consider the controversiality of a piece of content in the context of the community in which it is shared, because what is controversial to some audiences may not be so to others (Chen and Berger, 2013; Jang et al., 2017; Basile et al., 2017). For example, we identify "break up" as a controversial concept in the relationships subreddit (a subreddit is a subcommunity hosted on the Reddit discussion site), but the same topic is associated with a *lack* of controversy in the AskWomen subreddit (where questions are posed for women to answer). Similarly, topics that are controversial in one community may simply not be discussed in another: our analysis identifies "crossfit", a type of workout, as one of the most controversial concepts in the subreddit Fitness.

However, while controversial topics may be community-specific, community moderators still may not be able to determine a priori which posts will attract controversy. Many factors cannot be known ahead of time, e.g., a fixed set of topics may not be dynamic enough to handle a sudden current event, or the specific set of users that happen to be online at a given time may react in unpredictable ways. Indeed, experiments have shown that, to a certain extent, the influence of early opinions on subsequent opinion dynamics can override the influence of an item's actual content (Salganik et al., 2006; Wu and Huberman, 2008; Muchnik et al., 2013; Weninger et al., 2015).

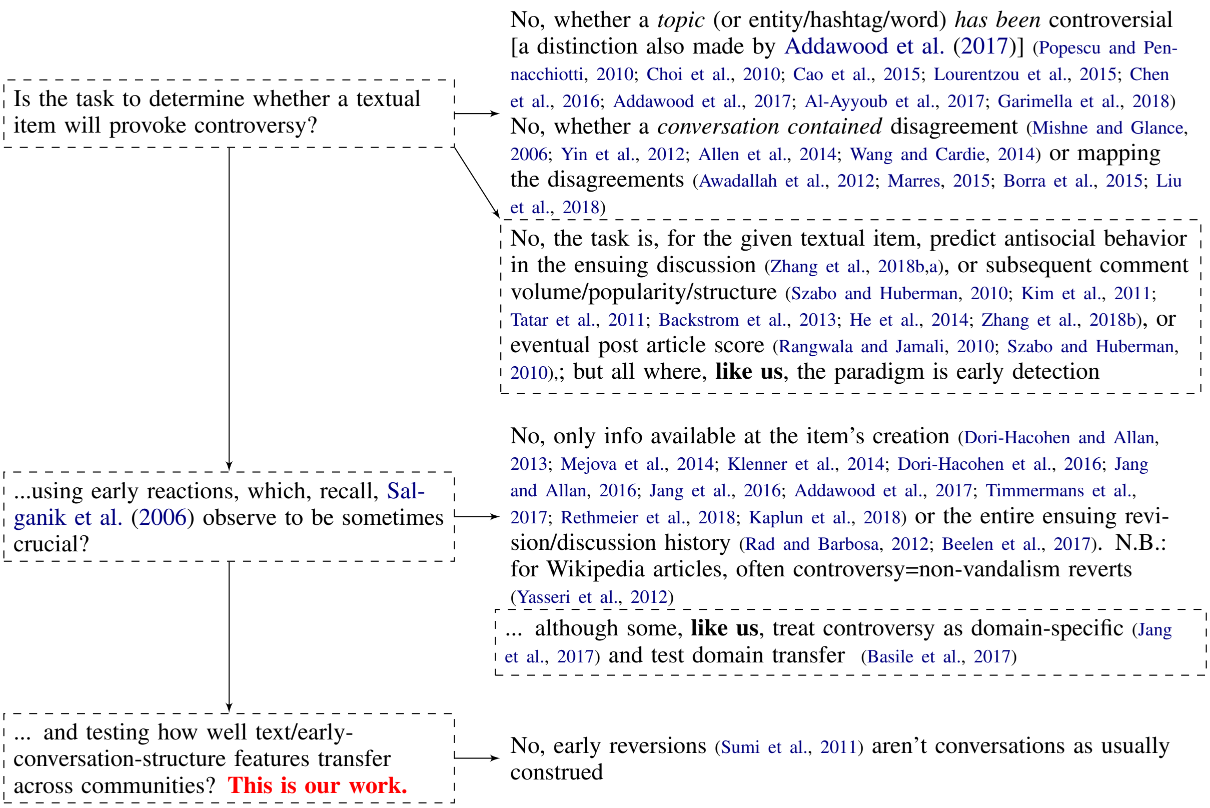
Hence, we propose an early-detection approach that uses not just the content of the initiating post, but also the content and structure of the initial responding comments. In doing so, we unite streams of heretofore mostly disjoint research programs: see Figure 1. Working with over 15,000 discus-

---

[1] Coser (1956); Jehn (1995); De Dreu and Weingart (2003) discuss how disagreement interacts with group makeup, group-task type, and outcome. Chen and Berger (2013) demonstrate a non-linear relationship between controversy and amount of subsequent discussion.

[2] We set aside the issue of *trolls* whose intent is solely to divide a community.

Proceedings of NAACL-HLT 2019, pages 1648–1659
Minneapolis, Minnesota, June 2 - June 7, 2019.



Is the task to determine whether a textual item will provoke controversy?

No, whether a *topic* (or entity/hashtag/word) *has been* controversial [a distinction also made by Addawood et al. (2017)] (Popescu and Pennacchiotti, 2010; Choi et al., 2010; Cao et al., 2015; Lourentzou et al., 2015; Chen et al., 2016; Addawood et al., 2017; Al-Ayyoub et al., 2017; Garimella et al., 2018) No, whether a *conversation contained* disagreement (Mishne and Glance, 2006; Yin et al., 2012; Allen et al., 2014; Wang and Cardie, 2014) or mapping the disagreements (Awadallah et al., 2012; Marres, 2015; Borra et al., 2015; Liu et al., 2018)

No, the task is, for the given textual item, predict antisocial behavior in the ensuing discussion (Zhang et al., 2018b,a), or subsequent comment volume/popularity/structure (Szabo and Huberman, 2010; Kim et al., 2011; Tatar et al., 2011; Backstrom et al., 2013; He et al., 2014; Zhang et al., 2018b), or eventual post article score (Rangwala and Jamali, 2010; Szabo and Huberman, 2010).; but all where, **like us**, the paradigm is early detection

...using early reactions, which, recall, Salganik et al. (2006) observe to be sometimes crucial?

No, only info available at the item's creation (Dori-Hacohen and Allan, 2013; Mejova et al., 2014; Klenner et al., 2014; Dori-Hacohen et al., 2016; Jang and Allan, 2016; Jang et al., 2016; Addawood et al., 2017; Timmermans et al., 2017; Rethmeier et al., 2018; Kaplun et al., 2018) or the entire ensuing revision/discussion history (Rad and Barbosa, 2012; Beelen et al., 2017). N.B.: for Wikipedia articles, often controversy=non-vandalism reverts (Yasseri et al., 2012)

... although some, **like us**, treat controversy as domain-specific (Jang et al., 2017) and test domain transfer (Basile et al., 2017)

... and testing how well text/early-conversation-structure features transfer across communities? **This is our work.**

No, early reversions (Sumi et al., 2011) aren't conversations as usually construed

Figure 1: How our research relates to prior work.

sion trees across six subreddits, we find that incorporating structural and textual features of budding comment trees improves predictive performance relatively quickly; for example, in one of the communities we consider, adding features taken from just the first 15 minutes of discussion significantly increases prediction performance, even though the average thread only contains 4 comments by that time ($\sim$4% of all eventual comments).

Additionally, we study feature transferability across domains (in our case, communities), training on one subreddit and testing on another. While text features of comments carry the greatest predictive capacity in-domain, we find that discussion-tree and -rate features are less brittle, transferring better between communities.

Our results not only suggest the potential usefulness of granting controversy-prediction algorithms a small observation window to gauge community feedback, but also demonstrate the utility of our expressive feature set for early discussions.

2 Datasets

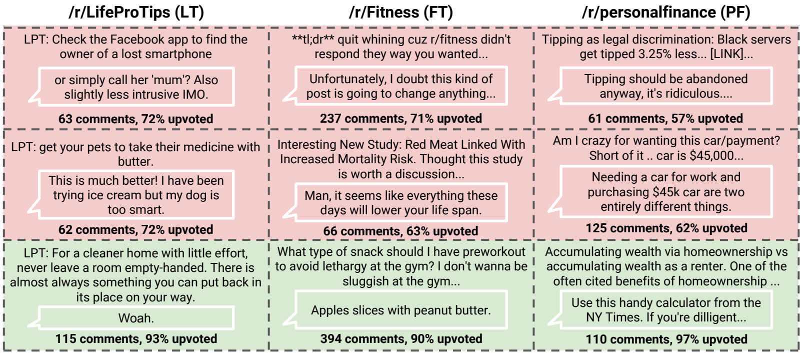
Given our interest in community-specific controversiality, we draw data from `reddit.com`, which hosts several thousand discussion subcommunities (subreddits) covering a variety of interests. Our dataset, which attempts to cover all public posts and comments from Reddit's inception in 2007 until Feb. 2014, is derived from a combination of Jason Baumgartner's posts and comments sets and our own scraping efforts to fill in dataset gaps. The result is a mostly-complete set of posts alongside associated comment trees.[3] We focus on six text-based[4] subreddits ranging over a variety of styles and topics: two Q&A subreddits: AskMen (AM) and AskWomen (AW); a special-interest community, Fitness (FT); and three advice communities: LifeProTips (LT), personalfinance (PF), and relationships (RL). Each comprises tens of thousands of posts and hundreds of thousands to millions of comments.

In Reddit (similarly to other sites allowing explicit negative feedback, such as YouTube, imgur, 9gag, etc.), users can give posts *upvotes*, increas-

---

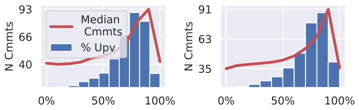
[3] Data hosted at `pushshift.io`, an open data initiative. Scraping was performed using Reddit's API or `github.com/pushshift/api`. Roughly 10% of comments and 20% of posts are deleted by users and/or moderators; also, authorship information is not available for many posts due to deletion of accounts.

[4] We ignore subreddits devoted to image sharing.

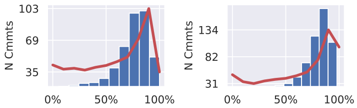


/r/LifeProTips (LT)	/r/Fitness (FT)	/r/personalfinance (PF)
LPT: Check the Facebook app to find the owner of a lost smartphone or simply call her 'mum'? Also slightly less intrusive IMO. **63 comments, 72% upvoted**	**tl;dr** quit whining cuz r/fitness didn't respond they way you wanted... Unfortunately, I doubt this kind of post is going to change anything... **237 comments, 71% upvoted**	Tipping as legal discrimination: Black servers get tipped 3.25% less... [LINK]... Tipping should be abandoned anyway, it's ridiculous.... **61 comments, 57% upvoted**
LPT: get your pets to take their medicine with butter. This is much better! I have been trying ice cream but my dog is too smart. **62 comments, 72% upvoted**	Interesting New Study: Red Meat Linked With Increased Mortality Risk. Thought this study is worth a discussion... Man, it seems like everything these days will lower your life span. **66 comments, 63% upvoted**	Am I crazy for wanting this car/payment? Short of it .. car is $45,000... Needing a car for work and purchasing $45k car are two entirely different things. **125 comments, 62% upvoted**
LPT: For a cleaner home with little effort, never leave a room empty-handed. There is almost always something you can put back in its place on your way. Woah. **115 comments, 93% upvoted**	What type of snack should I have preworkout to avoid lethargy at the gym? I don't wanna be sluggish at the gym... Apples slices with peanut butter. **394 comments, 90% upvoted**	Accumulating wealth via homeownership vs accumulating wealth as a renter. One of the often cited benefits of homeownership ... Use this handy calculator from the NY Times. If you're dilligent... **110 comments, 97% upvoted**

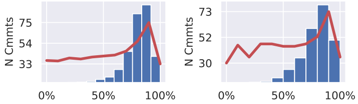
Figure 2: Examples of two controversial and one non-controversial post from three communities. Also shown are the text of the first reply, the number of comments the post received, and its percent-upvoted.



(a) AM (b) AW



(c) FT (d) LT



(e) PF (f) RL

Figure 3: For each community, a histogram of percent-upvoted and the median number of comments per bin.

ing a post's score, or *downvotes*, decreasing it.[5] While the semantics of up/down votes may vary based on community (and, indeed, each user may have their own views on what content should be upvoted and what downvoted), in aggregate, posts that split community reaction fundamentally differ from those that produce agreement. Thus, in principle, posts that have unambiguously received both many upvotes *and* many downvotes should be deemed the most controversial.

Percent Upvoted on Reddit. We quantify the relative proportion of upvotes and downvotes on a post using *percent-upvoted*, a measure provided by Reddit that gives an estimate of the percent of all votes on a post that are upvotes. In practice, exact values of percent-upvoted are not directly available; the site adds "vote fuzzing" to fight vote manipulation.[6] To begin with, we first discard posts with fewer than 30 comments.[7] Then, we query for the noisy percent-upvoted from each post ten times using the Reddit API, and take a mean to produce a final estimate.

Post Outcomes. To better understand the interplay between upvotes and downvotes, we first explore the outcomes for posts both in terms of percent-upvoted and the number of comments; doing so on a per-community basis has the potential to surface any subreddit-specific effects. In addition, we compute the median number of comments for posts falling into each bin of the histogram. The resulting plots are given in Figure 3.

In general, posts receive mostly positive feedback in aggregate, though the mean percent-upvoted varies between communities (Table 1). There is also a positive correlation between a post's percent-upvoted and the number of comments it receives. This relationship is unsurprising, given that Reddit displays higher rated posts to more users.

A null hypothesis, which we compare to empirically in our prediction experiments, is that popularity and percent-upvoted simply carry the same information. However, we have reason to doubt this null hypothesis, as quite a few posts receive significant attention despite having a low percent-upvoted (Figure 2).

Assigning Controversy Labels To Posts. We assign binary controversy labels (i.e., relatively controversial vs. relatively non-controversial) to posts according to the following process: first, we discard posts where the observed variability across 10 API queries for percent-upvoted exceeds 5%; in these cases, we assume that there are too few total votes for a stable estimate. Next, we discard posts where neither the observed upvote ratio nor the observed score[8] vary at all; in these cases, we cannot be sure that the upvote ratio is insensitive to the vote fuzzing function.[9] Fi-

---

[5] Vote timestamps are not publicly available.

[6] Prior to Dec. 2016, vote information was fuzzed according to a different algorithm; however, vote statistics for all posts were recomputed according to a new algorithm that, according to a reddit moderator, can "actually be trusted;" `https://goo.gl/yHWeJp`

[7] The intent is to only consider posts receiving enough community attention for us to reliably compare upvote counts with downvotes. We use number of comments as a proxy for aggregate attention because Reddit does not surface the true number of votes.

[8] A score is the (noised) upvotes minus the downvotes.

[9] We validate our filtration process in a later section by directly comparing to Reddit's rank-by-controversy function.

	# posts	# cmnts	μ_{up} cont	μ_{up} noncont
AM	3.3K	474K	66%	90%
AW	3.0K	417K	67%	91%
FT	3.9K	625K	66%	91%
LT	1.6K	208K	68%	91%
PF	1.0K	95K	72%	92%
RL	2.2K	221K	68%	93%

Table 1: Dataset statistics: number of posts, number of comments, mean percent-upvoted for the controversial and non-controversial classes.

nally, we sort each community's surviving posts by upvote percentage, and discard the small number of posts with percent-upvoted below 50%.[10] The top quartile of posts according to this ranking (i.e., posts with mostly only upvotes) are labeled "non-controversial." The bottom quartile of posts, where the number of downvotes cannot exceed but may approach the number of upvotes, are labeled as "controversial." For each community, this process yields a balanced, labeled set of controversial/non-controversial posts. Table 1 contains the number of posts/comments for each community after the above filtration process, and the percent-upvoted for the controversial/non-controversial sets.

2.1 Quantitative Validation of Labels

Reddit provides a sort-by-controversy function, and we wanted to ensure that our controversy labeling method aligned with this ranking.[11] We contacted Reddit itself, but they were unable to provide details. Hence, we scraped the 1K most controversial posts according to Reddit (1K is the max that Reddit provides) for each community over the past year (as of October 2018). Next, we sampled posts that *did not* appear on Reddit's controversial list in the year prior to October 2018 to create a 1:k ratio sample of Reddit-controversial posts and non-Reddit-controversial posts for $k \in \{1, 2, 3\}$, $k = 3$ being the most difficult setting. Then, we applied the filtering/labeling method described above, and measured how well our process matched Reddit's ranking scheme, i.e., the "controversy" label applied by our method matched the "controversy" label assigned by Reddit.

Our labeling method achieves high precision in identifying controversial/non-controversial posts. While a large proportion of posts are discarded, the labels assigned to surviving posts match those assigned by Reddit with the following F-measures at $k = 3$ (the results for $k = 1, 2$ are higher):[12]

	AM	AW	FT	LT	PF	RL
F-measure	97	96	88	90	94	96

In all cases, the precision for the non-controversial label is perfect, i.e., our filtration method never labeled a Reddit-controversial post as non-controversial. The precision of the controversy label was also high, but imperfect; errors could be a result of, e.g., Reddit's controversy ranking being limited to 1K posts, or using internal data, etc.

2.2 Qualitative Validation of Labels

Figure 2 gives examples of controversial and non-controversial posts from three of the communities we consider, alongside the text of the first comment made in response to those posts.

Topical differences. A priori, we expect that the topical content of posts may be related to how controversial they become (see prior work in Fig. 1). We ran LDA (Blei et al., 2003) with 10 topics on posts from each community independently, and compared the differences in mean topic frequency between controversial and non-controversial posts. We observe community-specific patterns, e.g., in relationships, posts about family (top words in topic: "family parents mom dad") are less controversial than those associated with romantic relationships (top words: "relationship, love, time, life"); in AskWomen, a gender topic ("women men woman male") tends to be associated with more controversy than an advice-seeking topic ("im dont feel ive")

Wording differences. We utilize Monroe et al.'s (2008) algorithm for comparing language usage in two bodies of text; the method places a Dirichlet prior over n-grams (n=1,2,3) and estimates Z-scores on the difference in rate-usage between controversial and non-controversial posts.

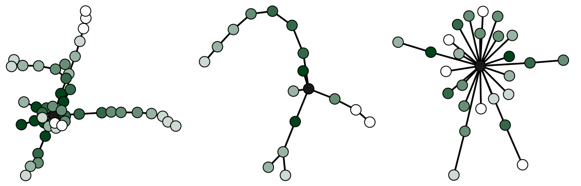
This analysis reveals many community-specific patterns, e.g., phrases associated with controversy include "crossfit" in Fitness, "cheated on my" in relationships, etc. What's controversial in one community may be non-controversial in another, e.g., "my parents" is associated with controversy

---

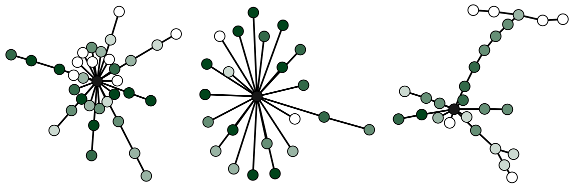
[10] Reddit provides less information for posts with more upvotes than downvotes.

[11] This validation step rules out the possibility that percent-upvoted is uncorrelated with Reddit's official definition of controversy.

[12] There were communities that we did not consider because the correlation between our filter and Reddit's ranking was lower, e.g., PoliticalDiscussion.



(a) Discussions on controversial posts



(b) Discussions on non-controversial posts

Figure 4: Early conversation trees from AskMen; nodes are comments and edges indicate reply structure. The original post is the black node, and as node colors lighten, comment timing increases from zero minutes to sixty minutes.

in personalfinance (e.g., "live with my parents") but strongly associated with lack of controversy in relationships (e.g., "my parents got divorced"). We also observe that some communities share commonalities in phrasing, e.g., "do you think" is associated with controversy in both AskMen and AskWomen, whereas "what are some" is associated with a lack of controversy in both.

3 Early Discussion Threads

We now analyze comments posted in early discussion threads for controversial vs. non-controversial posts. In this section, we focus on comments posted within one hour of the original submission, although we consider a wider range of times in later experiments.

Comment Text. We mirrored the n-gram analysis conducted in the previous section, but, rather than the text of the original post, focused on the text of comments. Many patterns persist, but the conversational framing changes, e.g., "I cheated" in the *posts* of relationships is mirrored by "you cheated" in the *comments*. Community differences again appear: e.g., "birth control" indicated controversy when it appears in the comments for relationships, but not for AskWomen.

Comment Tree Structure. While prior work in early prediction mostly focuses on measuring rate of early responses, we postulate that more expressive, structural features of conversation trees may also carry predictive capacity.

Figure 4 gives samples of conversation trees that developed on Reddit posts within one hour of the original post being made. There is significant diversity among tree size and shape. To quantify these differences, we introduce two sets of features: C-RATE features, which encode the rate of commenting/number of comments;[13] and C-TREE features, which encode structural aspects of discussion trees.[14] We then examine whether or not tree features correlate with controversy after controlling for popularity.

Using binary logistic regression, after controlling for C-RATE, C-TREE features extracted from comments made within one hour of the original post improve model fit in all cases except for personalfinance ($p < .05$, LL-Ratio test). We repeated the experiment, but also controlled for *eventual* popularity[15] in addition to C-RATE, and observed the same result. This provides evidence that *structural* features of conversation trees are predictive, though *which* tree feature is most important according to these experiments is community-specific. For example, for the models without eventual popularity information, the C-TREE feature with largest coefficient in AskWomen and AskMen was the max-depth ratio, but it was the Wiener index in Fitness.

4 Early Prediction of Controversy

We shift our focus to the task of predicting controversy on Reddit. In general, tools that predict controversy are most useful if they only require information available at the time of submission or as soon as possible thereafter. We note that while the causal relationship between vote totals and comment threads is not entirely clear (e.g., perhaps the comment threads cause more up/down votes on the post), predicting the ultimate *outcome* of posts is still useful for community moderators.

Experimental protocols. All classifiers are bi-

---

[13] Specifically: total number of comments, the logged time between OP and the first reply, and the average logged parent-child reply time over pairs of comments.

[14] Specifically: max depth/total comment ratio, proportion of comments that were top-level (i.e., made in direct reply to the original post), average node depth, average branching factor, proportion of top-level comments replied to, Gini coefficient of replies to top-level comments (to measure how "clustered" the total discussion is), and Wiener Index of virality (which measures the average pairwise path-length between all nodes in the conversation tree (Wiener, 1947; Goel et al., 2015)).

[15] We added in the logged number of eventual comments, and also whether or not the post received an above-median number of comments.

nary (i.e., controversial vs. non-controversial) and, because the classes are in 50/50 balance, we compare algorithms according to their accuracy. Experiments are conducted as 15-fold cross validation with random 60/20/20 train/dev/test splits, where the splits are drawn to preserve the 50/50 label distribution. For non-neural, feature-based classifiers, we use linear models.[16] For BiLSTM models,[17] we use Tensorflow (Abadi et al., 2015). Whenever a feature is ill-defined (e.g., if it is a comment text feature, but there are no comments at time t) the column mean of the training set for each cross-validation split is substituted. Similarly, if a comment's body is deleted, it is ignored by text processing algorithms. We perform both Wilcoxon signed-rank tests (Demšar, 2006) and two-sided corrected resampled t-tests (Nadeau and Bengio, 2000) to estimate statistical significance, taking the maximum of the two resulting p-values to err on the conservative side and reduce the chance of Type I error.

4.1 Comparing Text Models

The goal of this section is to compare text-only models for classifying controversial vs. non-controversial posts. Algorithms are given access to the full post titles and bodies, unless stated otherwise.

HAND. We consider a number of hand-designed features related to the textual content of posts inspired by Tan et al. (2016).[18]

TFIDF. We encode posts according to tfidf feature vectors. Words are included in the vocabulary if they appear more than 5 times in the corresponding cross-validation split.

W2V. We consider a mean, 300D word2vec (Mikolov et al., 2013) embedding representation, computed from a GoogleNews corpus.

ARORA. A slight modification of W2V, proposed by Arora et al. (2017), serves as a "tough to beat" baseline for sentence representations.

LSTM. We train a Bi-LSTM (Graves and Schmidhuber, 2005) over the first 128 tokens of titles + post text, followed by a mean pooling layer, and then a logistic regression layer. The LSTM's embedding layer is initialized with the same word2vec embeddings used in W2V. Markdown formatting artifacts are discarded.

BERT-LSTM. Recently, features extracted from fixed, pretrained, neural language models have resulted in high performance on a range of language tasks. Following the recommendations of §5.4 of Devlin et al. (2019), we consider representing posts by extracting BERT-Large embeddings computed for the first 128 tokens of titles + post text; we average the final 4 layers of the 24-layer, pretrained Transformer-decoder network (Vaswani et al., 2017). These token-specific vectors are then passed to a Bi-LSTM, a mean pooling layer, and a logistic classification layer. We keep markdown formatting artifacts because BERT's token vocabulary are WordPiece subtokens (Wu et al., 2016), which are able to incorporate arbitrary punctuation without modification.

BERT-MP. Instead of training a Bi-LSTM over BERT features, we mean pool over the first 128 tokens, apply L2 normalization to the resulting representations, reduce to 100 dimensions using PCA,[19] and train a linear classifier on top.

BERT-MP-512. The same as BERT-MP, except the algorithm is given access to 512 tokens (the maximum allowed by BERT-Large) instead of 128.

Results: Table 2 gives the performance of each text classifier for each community. In general, the best performing models are based on the BERT features, though HAND+W2V performs well, too. However, no performance gain is achieved when adding hand designed features to BERT. This may be because BERT's subtokenization scheme incorporates punctuation, link urls, etc., which are similar to the features captured by HAND. Adding an LSTM over BERT features is comparable to mean pooling over the sequence; similarly, considering 128 tokens vs. 512 tokens results in comparable

---

[16] We cross-validate regularization strength 10^(-100,-5,-4,-3,-2,-1,0,1), model type (SVM vs. Logistic L1 vs. Logistic L2 vs. Logistic L1/L2), and whether or not to apply feature standardization for each feature set and cross-validation split separately. These are trained using `lightning` (`http://contrib.scikit-learn.org/lightning/`).

[17] We optimize using Adam (Kingma and Ba, 2014) with LR=.001 for 20 epochs, apply dropout with $p = .2$, select the model checkpoint that performs best over the validation set, and cross-validate the model's dimension (128 vs. 256) and the number of layers (1 vs. 2) separately for each cross-validation split.

[18] Specifically: for the title and text body separately, length, type-token ratio, rate of first-person pronouns, rate of second-person pronouns, rate of question-marks, rate of capitalization, and Vader sentiment (Hutto and Gilbert, 2014). Combining the post title and post body: number of links, number of Reddit links, number of imgur links, number of sentences, Flesch-Kincaid readability score, rate of italics, rate of boldface, presence of a list, and the rate of word use from 25 Empath wordlists (Fast et al., 2016), which include various categories, such as politeness, swearing, sadness, etc.

[19] Values of 50 and 150 both work well, too.

	AM	AW	FT	LT	PF	RL
HAND	55.4	52.2	61.9	59.7	54.5	60.8
TFIDF	57.4	60.1	63.3	59.1	58.7	65.4
ARORA	58.6	62.0	60.5	59.4	57.2	62.1
W2V	60.7	62.1	63.1	61.4	59.9	64.3
LSTM	58.9	58.2	63.6	61.5	60.0	63.1
BERT-LSTM	**64.5**	**65.1**	**66.2**	65.0	65.1	67.8
BERT-MP	63.4	64.0	64.4	65.7	64.1	67.0
BERT-MP-512	63.9	64.0	64.7	**65.8**	**65.6**	67.7
HAND+W2V	61.3	62.3	64.9	63.2	60.0	66.3
HAND+BERTMP512	63.6	63.5	64.9	64.1	64.4	**68.0**

Table 2: Average accuracy for each post-time, text-only predictor for each dataset, averaged over 15 cross-validation splits; standard errors are ±.6, on average (and never exceed ±1.03). Bold is best in column; underlined are statistically indistinguishable from best in column ($p < .01$)

	AM	AW	FT	LT	PF	RL
TEXT	63.9	64.0	64.7	65.8	65.6	67.7
+TIME	68.1	65.4	65.5	66.2	66.5	69.3
+AUTHOR	68.2	65.3	65.7	66.0	66.4	69.3

Table 3: Post-time only results: the effect of incorporating timing and author identity features.

performance. Based on the results of this experiment, we adopt BERT-MP-512 to represent text in experiments for the rest of this work.

4.2 Post-time Metadata

Many non-content factors can influence community reception of posts, e.g., Hessel et al. (2017) find that *when* a post is made on Reddit can significantly influence its eventual popularity.

TIME. These features encode when a post was created. These include indicator variables for year, month, day-of-week, and hour-of-day.

AUTHOR. We add an indicator variable for each user that appears at least 3 times in the training set, encoding the hypothesis that some users may simply have a greater propensity to post controversial content.

The results of incorporating the metadata features on top of TEXT are given in Table 3. While incorporating TIME features on top of TEXT results in consistent improvements across all communities, incorporating author features on top of TIME+TEXT does not. We adopt our highest performing models, TEXT+TIME, as a strong post-time baseline.

4.3 Early Discussion Features

Basic statistics of early comments. We augment the post-time features with early-discussion feature sets by giving our algorithms access to comments from increasing observation periods. Specifically, we train linear classifiers by combining our best post-time feature set (TEXT+TIME) with features derived from comment trees available after t minutes, and sweep t from $t = 15$ to $t = 180$ minutes in 15 minute intervals.

Figure 6 plots the median number of comments available per thread at different t values for each community. The amount of data available for the early-prediction algorithms to consider varies significantly, e.g., while AskMen threads have a median 10 comments available at 45 minutes, LifeProTips posts do not reach that threshold even after 3 hours, and we thus expect that it will be a harder setting for early prediction. We see, too, that even our maximal 3 hour window is still early in a post's lifecycle, i.e., posts tend to receive significant attention afterwards: only 15% (LT) to 32% (AW) of all eventual comments are available per thread at this time, on average. Figure 7 gives the distribution of the number of comments available for controversial/non-controversial posts on AskWomen at $t = 60$ minutes. As with the other communities we consider, the distribution of number of available posts is not overly-skewed, i.e., most posts in our set (we filtered out posts with less than 30 comments) get at least *some* early comments.

We explore a number of feature sets based on early comment trees (comment feature sets are prefixed with "C-"):

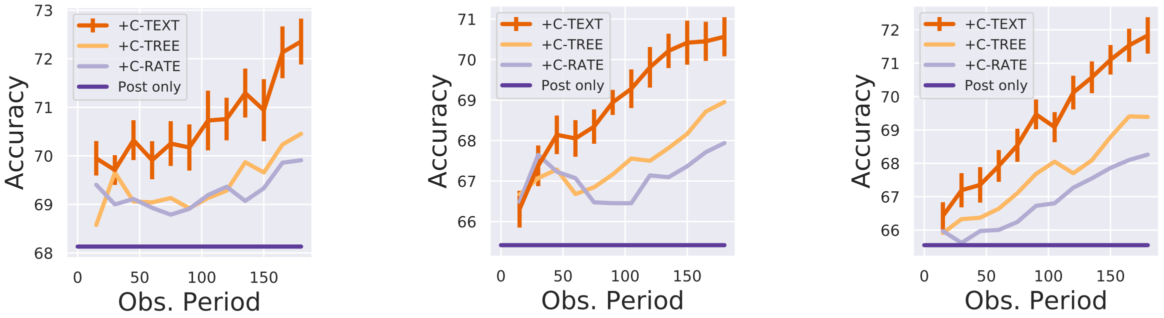
C-RATE and C-TREE. We described these in §3.

C-TEXT. For each comment available at a given observation period, we extract the BERT-MP-512 embedding. Then, for each conversation thread, we take a simple mean over all comment representations. While we tried several more expressive means of encoding the text of posts in comment trees, this simple method proved surprisingly effective.[20]

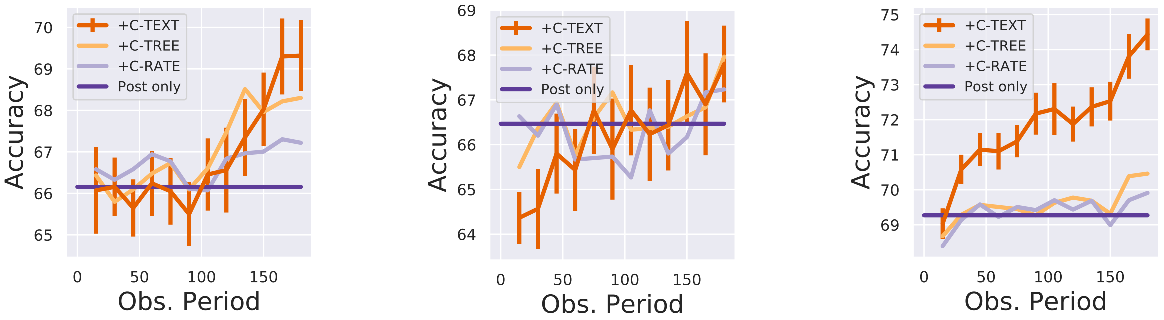
Sweeping over time. Figure 5 gives the performance of the post-time baseline combined with comment features while sweeping t from 15 to 180 minutes. For five of the six communities we consider, the performance of the comment feature classifier significantly ($p < .05$) ex-

---

[20] We do not claim that this is the *best* way to represent text in comment trees. However, this simple method produces performance improvements over strong post-time baselines; exploring better models is a promising avenue for future work.



(a) AskMen ($t_s = 15$) (b) AskWomen ($t_s = 45$) (c) Fitness ($t_s = 60$)



(d) LifeProTips ($t = 165$) (e) personalfinance (N/A) (f) relationships ($t_s = 45$)

Figure 5: Classifier accuracy for increasing periods of observation; the "+" in the legend indicates that a feature set is combined with the feature sets below. t_s, the time the full feature set first achieves statistical significance over the post-time only baseline, is given for each community (if significance is achieved).

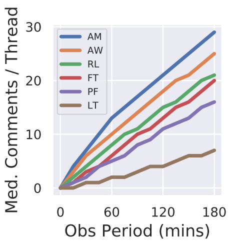


Figure 6: Observation period versus median number of comments available.

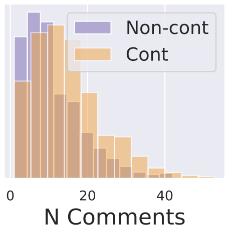


Figure 7: Histogram of the number of comments available per thread at $t = 60$ minutes in AskWomen.

ceeds the performance of the post-time baseline in less than three hours of observation, e.g., in the case of AskMen and AskWomen, significance is achieved within 15 and 45 minutes, respectively.

In general, C-RATE improves only slightly over post only, even though rate features have proven useful in predicting popularity in prior work (He et al., 2014). While adding C-TREE also improves performance, comment textual content is the biggest source of predictive gain. These results demonstrate i) that incorporating a variety of early conversation features, e.g., structural features of trees, can improve performance of controversy prediction over strong post-time baselines, and ii) the text content of comments contains significant complementary information to post text.

Controversy prediction $\neq$ popularity prediction. We return to a null hypothesis introduced in §2: that the controversy prediction models we consider here are merely learning the same patterns that a popularity prediction algorithm would learn. We train popularity prediction algorithms, and then attempt to use them at test-time to predict controversy; under the null hypothesis, we would expect little to no performance degradation when training on these alternate labels.

We 1) train binary popularity predictors using post text/time + comment rate/tree/text features available at $t = 180$,[21] and use them to predict controversy at test-time; and 2) consider an oracle that predicts the true popularity label at test-time; this oracle is *quite* strong, as prior work suggests that perfectly predicting popularity is impossible (Salganik et al., 2006).

---

[21] We predict whether or not a post eventually receives an above-median number of comments. We force the popularity predictors to predict 50/50 at test time, which improves their performance.

	AM	AW	FT	LT	PF	RL
Pop Pred	53.9	55.2	60.1	54.2	52.9	52.8
Pop Oracle	65.8	67.0	70.3	68.1	64.0	63.3

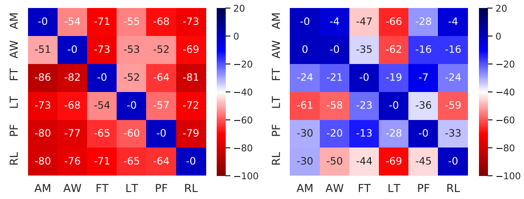
In all cases, the best popularity predictor does not achieve performance comparable to even the post-only baseline. For 3 of 6 communities, even the popularity oracle does not beat post time baseline, and in all cases, the mean performance of the controversy predictor exceeds the oracle by $t = 180$. Thus, in our setting, controversy predictors and popularity predictors learn disjoint patterns.

4.3.1 Domain Transfer

We conduct experiments where we train models on one subreddit and test them on another. For these experiments, we discard all posting time features, and compare C-(TEXT+TREE+RATE) to C-(TREE+RATE); the goal is to empirically examine the hypothesis in §1: that controversial text is community-specific.

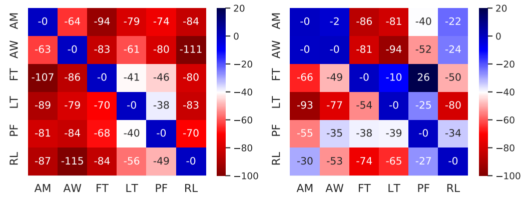
To measure performance differences in the domain transfer setting, we compute the percentage accuracy drop relative to a constant prediction baseline when switching the training subreddit from the matching subreddit to a different one. For example, at $t = 60$, we observe that raw accuracy drops from $65.6 \rightarrow 55.8$ when training on AskWomen and testing on AskMen when considering text, rate, and tree features together; given that the constant prediction baseline achieves 50% accuracy, we compute the percent drop in accuracy as: $(55.8 - 50)/(65.6 - 50) - 1 = -63\%$.

The results of this experiment (Figure 8) suggest that while text features are quite strong in-domain, they are brittle and community specific. Conversely, while rate and structural comment tree features do not carry as much in-domain predictive capacity on their own, they generally transfer better between communities, e.g., for RATE+TREE, there is very little performance drop-off when training/testing on AskMen/AskWomen (this holds for all timing cutoffs we considered). Similarly, in the case of training on Fitness and testing on PersonalFinance, we sometimes observe a performance *increase* when switching domains (e.g., at $t = 60$); we suspect that this could be an effect of dataset size, as our Fitness dataset has the most posts of any subreddit we consider, and PersonalFinance has the least.



(a) TEXT+RATE+TREE $t = 180$

(b) RATE+TREE $t = 180$



(c) TEXT+RATE+TREE $t = 60$

(d) RATE+TREE $t = 60$

Figure 8: Average cross-validated performance degradation for transfer learning setting at $t = 180$ and $t = 60$; the y-axis is the training subreddit and the x-axis is testing. For a fixed test subreddit, each column gives the percent accuracy drop when switching from the matching training set to a domain transfer setting. In general, while incorporating comment text features results in higher accuracy overall, comment rate + tree features transfer between communities with less performance degradation.

5 Conclusion

We demonstrated that early discussion features are predictive of eventual controversiality in several reddit communities. This finding was dependent upon considering an expressive feature set of early discussions; to our knowledge, this type of feature set (consisting of text, trees, etc.) hadn't been thoroughly explored in prior early prediction work.

One promising avenue for future work is to examine higher-quality textual representations for conversation trees. While our mean-pooling method did produce high performance, the resulting classifiers do not transfer between domains effectively. Developing a more expressive algorithm (e.g., one that incorporates reply-structure relationships) could boost predictive performance, and enable textual features to be less brittle.

Acknowledgments

We thank C. Danescu-Niculescu-Mizil, J. Zhang, V. Niculae, J. Kleinberg, and the reviewers for helpful feedback, and NVidia Corporation for GPUs. This work was supported in part by NSF grant SES-1741441, but this material does not necessarily reflect the views of the sponsors.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aseel Addawood, Rezvaneh Rezapour, Omid Abdar, and Jana Diesner. 2017. Telling apart tweets associated with controversial versus non-controversial topics. In *Second Workshop on NLP and Computational Social Science*.

Mahmoud Al-Ayyoub, Abdullateef Rababah, Yaser Jararweh, Mohammed N. Al-Kabi, and Brij B. Gupta. 2017. Studying the controversy in online crowds interactions. *Applied Soft Computing*.

Kelsey Allen, Giuseppe Carenini, and Raymond Ng. 2014. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *EMNLP*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2012. Harmony and dissonance: organizing the people's voices on political controversies. In *Proceedings of WSDM*, page 523, Seattle, Washington, USA. ACM Press.

Lars Backstrom, Jon Kleinberg, Lillian Lee, and Cristian Danescu-Niculescu-Mizil. 2013. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *WSDM*. ACM. Project homepage at `http://www.cs.cornell.edu/home/llee/papers/convcuration.home.html`.

Angelo Basile, Tommaso Caselli, and Malvina Nissim. 2017. Predicting controversial news using Facebook reactions. In *The Italian Conference on Computational Linguistics*.

Kaspar Beelen, Evangelos Kanoulas, and Bob van de Velde. 2017. Detecting Controversies in Online News Media. In *Proceedings of SIGIR*, pages 1069–1072, New York, NY, USA. ACM.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *JMLR*.

Erik Borra, Esther Weltevrede, Paolo Ciuccarelli, Andreas Kaltenbrunner, David Laniado, Giovanni Magni, Michele Mauri, Richard Rogers, and Tommaso Venturini. 2015. Societal Controversies in Wikipedia Articles. In *Proceedings of CHI*, pages 193–196, New York, NY, USA. ACM.

Nan Cao, Lu Lu, Yu-Ru Lin, Fei Wang, and Zhen Wen. 2015. Socialhelix: Visual analysis of sentiment divergence in social media. *Journal of Visualization*.

Wei-Fan Chen, Fang-Yu Lin, and Lun-Wei Ku. 2016. Wordforce: Visualizing controversial words in debates. In *Proceedings of COLING: System Demonstrations*, pages 273–277, Osaka, Japan. The COLING 2016 Organizing Committee.

Zoey Chen and Jonah Berger. 2013. When, why, and how controversy causes conversation. *Journal of Consumer Research*, 40(3).

Yoonjung Choi, Yuchul Jung, and Sung-Hyon Myaeng. 2010. Identifying controversial issues and their subtopics in news articles. In *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer.

Lewis Coser. 1956. *The Functions of Social Conflict*. The Free Press, New York.

Carsten K. W. De Dreu and Laurie R. Weingart. 2003. Task versus relationship conflict, team performance, and team member satisfaction: A meta-analysis.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Shiri Dori-Hacohen and James Allan. 2013. Detecting controversy on the web. In *CIKM*. ACM.

Shiri Dori-Hacohen, David Jensen, and James Allan. 2016. Controversy detection in Wikipedia using collective classification. In *Proceedings of SIGIR*, pages 797–800, New York, NY, USA. ACM.

Ethan Fast, Binbin Chen, and Michael S. Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *CHI*.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2018. Quantifying controversy on social media. *ACM Transactions on Social Computing*.

Sharad Goel, Ashton Anderson, Jake Hofman, and Duncan J. Watts. 2015. The structural virality of online diffusion. *Management Science*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*.

Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014. Predicting the popularity of Web 2.0 items based on user comments. In *SIGIR*.

Jack Hessel, Lillian Lee, and David Mimno. 2017. Cats and captions vs. creators and the clock: Comparing multimodal content to context in predicting relative popularity. In *WWW*. Project homepage at `http://www.cs.cornell.edu/~jhessel/cats/cats.html`.

C.J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *IWCSM*.

Myungha Jang and James Allan. 2016. Improving automated controversy detection on the web. In *SIGIR*.

Myungha Jang, Shiri Dori-Hacohen, and James Allan. 2017. Modeling controversy within populations. In *SIGIR*.

Myungha Jang, John Foley, Shiri Dori-Hacohen, and James Allan. 2016. Probabilistic approaches to controversy detection. In *CIKM*.

Karen A. Jehn. 1995. A multimethod examination of the benefits and detriments of intragroup conflict. *Administrative Science Quarterly*, 40(2):256–282.

Kateryna Kaplun, Christopher Leberknight, and Anna Feldman. 2018. Controversy and sentiment: An exploratory study. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence (SETN)*, pages 37:1–37:7, New York, NY, USA. ACM.

Su-Do Kim, Sung-Hwan Kim, and Hwan-Gue Cho. 2011. Predicting the virtual temperature of weblog articles as a measurement tool for online popularity. In *11th International Conference on Computer and Information Technology*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Manfred Klenner, Michael Amsler, and Nora Hollenstein. 2014. Verb polarity frames: A new resource and its application in target-specific polarity classification. In *KONVENS*.

Weichen Liu, Sijia Xiao, Jacob T. Browne, Ming Yang, and Steven P. Dow. 2018. ConsensUs: Supporting multi-criteria group decisions by visualizing points of disagreement. *ACM Transactions on Social Computing*, 1(1):1–26.

Ismini Lourentzou, Graham Dyer, Abhishek Sharma, and ChengXiang Zhai. 2015. Hotspots of news articles: Joint mining of news text & social media to discover controversial points in news. In *2015 IEEE International Conference on Big Data*, pages 2948–2950. IEEE.

Noortje Marres. 2015. Why map issues? On controversy analysis as a digital method. *Science, Technology, & Human Values*, 40(5):655–686.

Yelena Mejova, Amy X. Zhang, Nicholas Diakopoulos, and Carlos Castillo. 2014. Controversy and sentiment in online news. In *Computation and Journalism Symposium*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Gilad Mishne and Natalie Glance. 2006. Leave a reply: An analysis of weblog comments. In *Third Annual Workshop on the Weblogging Ecosystem*. Edinburgh, Scotland.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*.

Lev Muchnik, Sinan Aral, and Sean J. Taylor. 2013. Social influence bias: A randomized experiment. *Science*, pages 647–651.

Claude Nadeau and Yoshua Bengio. 2000. Inference for the generalization error. In *NeurIPS*.

Ana-Maria Popescu and Marco Pennacchiotti. 2010. Detecting controversial events from Twitter. In *CIKM*.

Hoda Sepehri Rad and Denilson Barbosa. 2012. Identifying controversial articles in Wikipedia: A comparative study. In *Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration (WikiSym)*, pages 7:1–7:10, New York, NY, USA. ACM.

Huzefa Rangwala and Salman Jamali. 2010. Defining a coparticipation network using comments on Digg. *IEEE Intelligent Systems*.

Nils Rethmeier, Marc Hübner, and Leonhard Hennig. 2018. Learning comment controversy prediction in web discussions using incidentally supervised multitask CNNs. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 316–321.

Matthew J. Salganik, Peter Sheridan Dodds, and Duncan J. Watts. 2006. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311:854–6.

Róbert Sumi, Taha Yasseri, András Rung, Andrs Kornai, and János Kertész. 2011. Characterization and prediction of Wikipedia edit wars. In *Proceedings of Web Science*. Poster.

Gabor Szabo and Bernardo A. Huberman. 2010. Predicting the popularity of online content. *Comm. of the ACM*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *WWW*, pages 613–624. Paper homepage at `https://chenhaot.com/papers/changemyview.html`.

Alexandru Tatar, Jérémie Leguay, Panayotis Antoniadis, Arnaud Limbourg, Marcelo Dias de Amorim, and Serge Fdida. 2011. Predicting the popularity of online articles based on user comments. In *International Conference on Web Intelligence, Mining and Semantics*.

Bram Timmermans, Lora Aroyo, Thomas A. Kuhn, Kaspar Beelen, Evangelos Kanoulas, Bob van de Velde, and Gerben van Eerten. 2017. ControCurator: Understanding controversy using collective intelligence. In *Proceedings of Collective Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Lu Wang and Claire Cardie. 2014. A piece of my mind: A sentiment analysis approach for online dispute detection. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 693–699, Baltimore, Maryland. Association for Computational Linguistics.

Tim Weninger, Thomas James Johnston, and Maria Glenski. 2015. Random voting effects in social-digital spaces: A case study of Reddit post submissions. In *Conference on Hypertext and Social Media*, pages 293–297, New York, NY, USA. ACM.

Harry Wiener. 1947. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*.

Fang Wu and Bernardo A. Huberman. 2008. How public opinion forms. In *Proceedings of the 4th International Workshop on Internet and Network Economics (WINE)*, pages 334–341, Berlin, Heidelberg. Springer-Verlag.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*. Version 2.

Taha Yasseri, Robert Sumi, Andrs Rung, Andrs Kornai, and Jnos Kertsz. 2012. Dynamics of Conflicts in Wikipedia. *PLoS ONE*, 7(6):e38869.

Jie Yin, Paul Thomas, Nalin Narang, and Cecile Paris. 2012. Unifying local and global agreement and disagreement classification in online debates. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*. Association for Computational Linguistics.

Justine Zhang, Jonathan P. Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Nithum Thain, and Dario Taraborelli. 2018a. Conversations gone awry: Detecting early signs of conversational failure. In *ACL*. Project homepage at `http://www.cs.cornell.edu/~cristian/Conversations_gone_awry.html`.

Justine Zhang, Cristian Danescu-Niculescu-Mizil, Christina Sauper, and Sean J. Taylor. 2018b. Characterizing online public discussions through patterns of participant interactions. In *Proceedings of CSCW*. Project homepage at `http://www.cs.cornell.edu/~cristian/Patterns_of_participant_interactions.html`.

No Permanent Friends or Enemies: Tracking Relationships between Nations from News

Xiaochuang Han
College of Computing
Georgia Institute of Technology
Atlanta, GA
`xc@gatech.edu`

Eunsol Choi
Paul G. Allen School of CSE
University of Washington
Seattle, WA
`eunsol@cs.washington.edu`

Chenhao Tan
Dept. of Computer Science
Unversity of Colorado Boulder
Boulder, CO
`chenhao@chenhaot.com`

Abstract

Understanding the dynamics of international politics is important yet challenging for civilians. In this work, we explore unsupervised neural models to infer relations between nations from news articles. We extend existing models by incorporating shallow linguistics information and propose a new automatic evaluation metric that aligns relationship dynamics with manually annotated key events. As understanding international relations requires carefully analyzing complex relationships, we conduct in-person human evaluations with three groups of participants. Overall, humans prefer the outputs of our model and give insightful feedback that suggests future directions for human-centered models. Furthermore, our model reveals interesting regional differences in news coverage. For instance, with respect to US-China relations, Singaporean media focus more on "strengthening" and "purchasing", while US media focus more on "criticizing" and "denouncing".

1 Introduction

In the context of growing globalization (Baylis et al., 2017), understanding complex international relations is increasingly relevant to our daily life. Yet this is a challenging task due to the inherently dynamic nature of international relations. As Kissinger famously said, "America has no permanent friends or enemies, only interests." Staying informed becomes even harder in the continuous streams of information from news outlets and social media.

This very availability of such information, however, opens up exciting opportunities for natural language processing to support individuals in understanding international relations. Supervised extraction has been incredibly useful at identifying pre-defined relations and events (Doddington et al., 2004; Mintz et al., 2009) but fails to capture emerging or complex information needs. Topic models and neural models have been proposed to explore relations between entities without supervision (O'Connor et al., 2013; Chaney et al., 2016; Iyyer et al., 2016). In particular, Iyyer et al. (2016) introduces an unsupervised neural model for tracking relations between fictional characters, and this approach outperforms baselines from topic models and hidden Markov models. In this work, we incorporate linguistic insights into this model to track relation dynamics between nations from news articles.

Our model reconstructs textual information in the embedding space using relation embeddings, as proposed in Iyyer et al. (2016). We integrate simple yet effective linguistic insights: verbal predicates often describe the relationship between entities,[1] while nouns and proper nouns provide the context of this relationship. For example, in "U.S. denounces Russia for its interference in the 2016 election", *denounce* describes the relation, and *election* and *interference* provide the context. We show that this intuition leads the model to discover relation descriptors that are easier to interpret and less noisy.

Evaluating these exploratory models for subjective tasks poses a challenge as there are no gold labels. Along with the model, we propose new approaches for evaluation. We introduce a quantitative metric which aligns pre-annotated key events with the temporal trends of relationships produced by the models. Since this task requires careful analysis of complex international relations, we conduct in-person user studies with NLP researchers and undergraduate students recruited from political science and linguistics courses. Both quantitative evaluation and human evaluation indicate that our model better rep-

---

[1] We use entities, countries, and nations interchangeably in this work.

Proceedings of NAACL-HLT 2019, pages 1660–1676
Minneapolis, Minnesota, June 2 - June 7, 2019.

resents the dynamic relationships between nations than the prior model (Iyyer et al., 2016): 75.9% of participants preferred our model for finding natural language words describing international relations and 85.5% preferred temporal trends generated by our model.

Finally, we qualitatively explore the context of relations provided by an attention-based mechanism and demonstrate a practical application of our model by studying regional differences in news coverage of relationships between two countries. We conclude with discussions on future directions for buildings models that can support individuals in navigating a large collection of news articles. Our code is available at `https://github.com/BoulderDS/LARN`.

2 Data

We start by introducing our dataset of news articles, the shallow linguistic information that we extract, and our annotation of key events.

News article collection. Our dataset is derived from the NOW corpus, the largest corpus of news articles that is available in full-text format.[2] The NOW corpus collects the news articles that are available on Google News in 20 English-speaking countries and thus include news articles that span a wide variety of topics, ranging from politics, to sports, to celebrity news from 23K media outlets. In this work, we consider the news articles in recent years, i.e., from January 2016 to June 2018, to facilitate human evaluation.

We consider 12 nations (U.S., Russia, China, UK, Germany, Canada, France, India, Japan, Iran, Israel, and Syria) and the 66 nation pairs between them. To identify mentions of each nation in news articles, we manually construct a set of aliases for each nation to cover common abbreviations and the names of political leaders (e.g., Trump, Putin). On average, each nation has 3.5 aliases. We then use these aliases to find sentences that contain a pair of nations under consideration and obtain 1.2M sentences associated with 634K articles.

Adding Shallow Linguistic Information. To incorporate shallow linguistic knowledge, we process the news article collection for each nation pairs to extract (1) verbal predicates and (2) nouns and proper nouns from each sentence. Specifically, we use a dependency parser to detect verbal predicates and their subjects and objects, and only include predicates for which both subjects and objects were found. For sentences with such predicates, we find nouns and proper nouns using part of speech tags. All data processing was done in spaCy (Honnibal and Montani, 2017) and full details can be found in the appendix.

Total # of months considered	30
Total # of articles	7.7M
Total # of articles with valid nation pairs	634K
Avg. # of articles per nation pair	9.6K
Avg. # of sentences per nation pair	18.2K
Avg. # of predicates per nation pair	21.6K
Avg. # of nouns & proper nouns per nation pair	201K
Avg. # of key events per nation pair (8 pairs)	4.9

Table 1: Data statistics.

Key events annotation. The main goal of our model is to support the exploration of international relations, which is very challenging to evaluate. To derive quantitative evaluation measures and provide the basic context of international relations, we manually identify key events over the 30 months for 8 most frequently mentioned nation pairs (i.e. US-[China, Russia, UK, India, Canada, Japan, Syria], China-India) by reading through the top Google search results for each two countries and each month.[3] We identified roughly five key events per nation pair. For example, one key event for US-China relation is Chairman Xi's visit to the US in April 2017. A complete list of key events is shown in the appendix. Table 1 summarizes the data statistics.

3 Model

In this section, we formally introduce our model that builds on Relation Modeling Network (*RMN*) (Iyyer et al., 2016). Our main contribution is to integrate shallow linguistic information (i.e., verbal predicates and nouns/proper nouns) and identify the context of relations.

3.1 Overview

The intuition behind our model follows RMN, i.e., inferring relation embeddings by reconstruct-

---

[2] The dataset can be obtained from `https://www.corpusdata.org/now_corpus.asp`.

[3] This annotation is done by the first author and is thus inherently subjective. We add a robustness check in the appendix based on another set of independently annotated key events by the third author. Automatic evaluation shows a similar trend holds for the two sets of annotations.

ing textual information in the embedding space. Specifically, we learn a fixed set of relation embeddings and use a convex combination of these relation embeddings to reconstruct information in sentences that mention both entities.

Our main hypothesis is that relation information is often encoded in verbal predicates and we can obtain more interpretable and robust relations if we focus on predicates. For each pair of entities, we extract information from the predicates in sentences with both entities and reconstruct these predicates using shared relation embeddings. In addition, we use nouns to provide the context for the relations. We refer to our model as Linguistically Aware Relationship Networks (*LARN*).

We now formally define our problem. The input of our model is a collection of news articles. For each entity pair e_i, e_j, we obtain a set of articles, $\mathcal{A}_{e_i,e_j}$, containing at least one sentence mentioning both entities. We identify sentences where both e_i and e_j occur based on any alias associated with an entity (nation). We extract all the verbal predicates from these sentences in article $a \in \mathcal{A}_{e_i,e_j}$ as $\{p_1^{a,e_i,e_j}, \ldots, p_N^{a,e_i,e_j}\}$, as well as the proper nouns or nouns as $\{n_1^{a,e_i,e_j}, \ldots, n_M^{a,e_i,e_j}\}$. Preprocessing details could be found in the appendix. We then use GloVe embeddings to represent these words, i.e., $v_{p_k^{a,e_i,e_j}}$ for p_k^{a,e_i,e_j} and $v_{n_k^{a,e_i,e_j}}$ for n_k^{a,e_i,e_j} (Pennington et al., 2014). These word embeddings are static in the entire learning process.

Our model learns relation embeddings $\mathbf{R} \in \mathbb{R}^{K \times d}$, where K is a hyperparameter for the number of relations and d is the dimension of the relation embedding as well as the word embedding. Following Iyyer et al. (2016), we obtain a list of natural language descriptors for each relation using the nearest neighbors of the relation embedding within the 500 most common predicates.[4] The model also provides (1) a probability distribution over relations for each article, (2) a probability distribution over nouns for each relation between each entity pair, and (3) an embedding for each entity.

Figure 1 describes the overall architecture. We will describe the construction of v^a_{label} in §3.2, a reconstruction of v^a_{label} through a weighted sum of relation embeddings in §3.3, and finally the learning in §3.4.

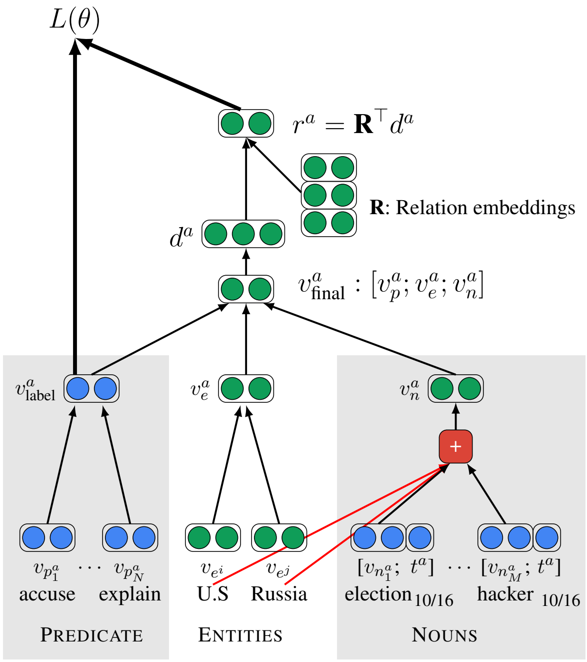


Figure 1: Model overview of an article with respect to a pair of nations. Our model approximates the predicates from the input article v^a_{label} as a weighted sum of relation embeddings from $\mathbf{R}$. The blue embeddings are fixed and the green ones are trained. The red cell indicates our attention mechanism. The shaded area is our extension: we use only predicates as the label and provide the context of relations via nouns.

3.2 Modeling Text with Predicates

We compute the representation for each article to be reconstructed as the sum of bag-of-words embeddings. While Iyyer et al. (2016) considers all the words in a window in which both entities occur, we only consider predicates in the sentences where both entities show up:

$$v_{\text{label}}^{a,e_i,e_j} = \sum_{k=1}^{N} v_{p_k^{a,e_i,e_j}}.$$

$v_{\text{label}}^{a,e_i,e_j}$ depends on both the news article and the entity pair. In the rest of the paper, We omit e_i, e_j in the superscript here for simplicity, i.e., v^a_{label}.

3.3 Reconstructing Text with Relation Embeddings

We represent each article as a weighted sum of relation embeddings. Assuming d^a represents a weight vector over the K relation embeddings that

---

[4] Refer to the appendix for a version that include all words. The relation descriptors from *RMN* required a manual filtering step in Iyyer et al. (2016), and become unintelligible without the 500 common words constraint. Note that our model produces intelligible descriptors even without the 500 most common predicates constraint.

sums to 1 for an article a with respect to a pair of entities (e_i, e_j), we obtain r^a as follows:

$$r^a = \mathbf{R}^\top d^a.$$

d^a can also be thought of as a distribution over relations. This distribution over relations depends on entity pair (v^a_e), predicate information (v^a_p), and noun information (v^a_n). While *RMN* simply takes the average of all words in the sentence, *LARN* focuses on verbal predicates and nouns to capture our intuition that predicates describe the main relations, whereas nouns provide background information to explain the relations. We now describe these three components in details.

Representing predicates and entity pair. We follow *RMN* to construct embedding for each entity pair and for the predicates. The entity pair vector, v^a_e, simply adds the embedding of the two entities. The predicate vector, v^a_p, is equivalent to v^a_{label} except for word dropout during training, i.e., setting b^a_k to be 0 or 1 with a probability of 0.5.

$$\begin{aligned} v^a_e &= v_{e_i} + v_{e_j}, \\ v^a_{\text{p}} &= \sum_{k=1}^{N} b^a_k \cdot v_{p^a_k}. \end{aligned}$$

Representing context with nouns. To understand relations between an entity pair in a sentence, nouns should be considered in addition to predicates. For example, *tariff* is indicative of the relation between US and China in "Originally, Trump favoured the simple imposition of a tariff on products from selected countries, especially China and Mexico", despite the seemingly positive predicate *favour*. As nouns are much more common than predicates (see Table 1) and not all of them are meaningful for understanding international relations, we employ a weighted sum of noun vectors. We use an attention mechanism (Conneau et al., 2017; Bahdanau et al., 2015) and consider each entity pair as a unique key to compute the attention weights, since the same noun can be interpreted differently across different entity pairs. To this end, we train an attention query embedding q_{e_i,e_j} for each entity pair separately. We further encode the temporal information by concatenating a one-hot vector t^a that indicates the month when the article was published with the noun representation $v_{n^a_k}$. This allows us to capture the shifts in a word's meaning over time.

$$\begin{aligned} h_{n^a_k} &= \tanh(W_{\text{proj}} \cdot [v_{n^a_k}; t^a]), \\ \alpha_{n^a_k} &= \frac{\exp(h_{n^a_k} \cdot q_{e_i,e_j})}{\sum_{k'=1}^{M} \exp(h_{n^a_{k'}} \cdot q_{e_i,e_j})}, \\ v^a_n &= \sum_{k=1}^{M} \alpha_{n^a_k} h_{n^a_k}. \end{aligned}$$

Finally, we concatenate the three representations as the input to a feedforward network and pass to a softmax layer to create the weight vector d^a.

$$\begin{aligned} v^a_{\text{final}} &= \text{ReLU}(W_{\text{cat}} \cdot [v^a_p; v^a_e; v^a_n]), \\ d^a &= \text{Softmax}(W_{\text{final}} \cdot v^a_{\text{final}}). \end{aligned}$$

This d^a is multiplied with the descriptor matrix $\mathbf{R}$ to get the final representation r^a. Different from *RMN*, we do not consider temporal dependencies between time steps in our model because it is important to understand sudden shifts in international relations rather than assuming that the relations slowly evolve. We also found the temporal dependencies were not helpful empirically in our domain but rather computationally expensive.

3.4 Learning Objective and Summary

The reconstruction objective pushes r^a to resemble v^a_{label}. Our formulation is identical to *RMN*: the loss function consists of a contrastive max-margin loss term, J, and an auxiliary loss term, X, to encourage unique relation embeddings.

$$\begin{aligned} L(\theta) =& J(\theta) + \lambda X(\theta), \\ J(\theta) =& \sum_{a \in \mathcal{A}_{e_i,e_j}} \sum_{v^a_{\text{label}'} \in \mathcal{N}} \max(0, \\ & 1 - \frac{r_a \cdot v^a_{\text{label}}}{||r_a|| \cdot ||v^a_{\text{label}}||} + \frac{r_a \cdot v^a_{\text{label}'}}{||r_a|| \cdot ||v^a_{\text{label}'}||}), \\ X(\theta) =& \|RR^\top - I\|, \end{aligned}$$

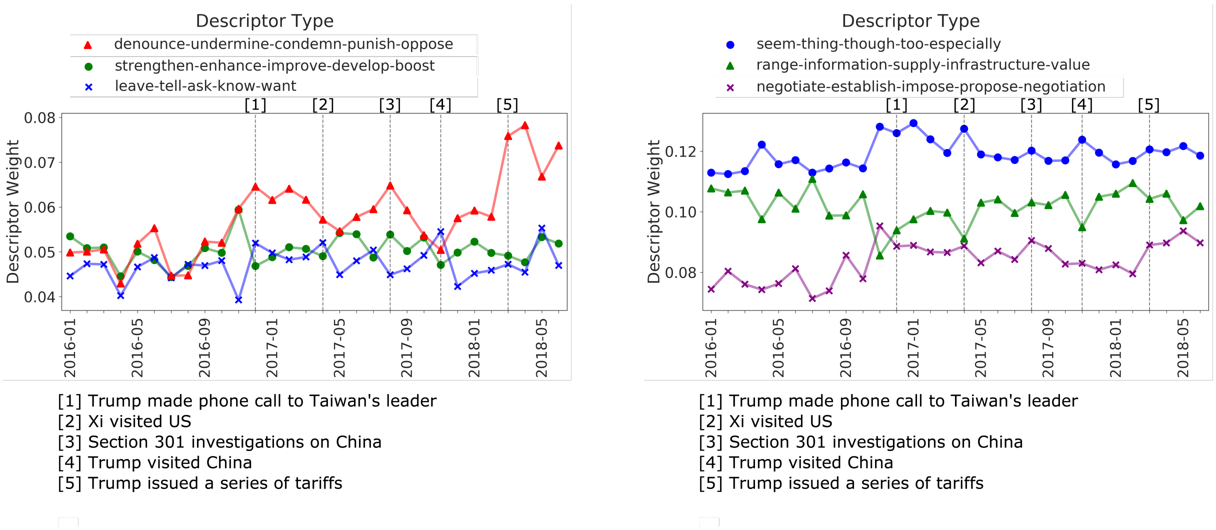
where v'_{label} is a randomly sampled negative example, $\mathcal{N}$ is a collection of them, and λ is a hyperparameter for balancing two loss terms.

4 Evaluation

In this section, we compare our model to *RMN*. For both models, we fixed the number of descriptors to 30 following Iyyer et al. (2016). As tracking dynamic international relations requires careful analysis, we hosted onsite user studies for quality control and in-person feedback. We first describe the model outputs, and then present both quantitative and qualitative evaluation results.

LARN (Linguistically Aware Relationship Network)	Weight	*RMN* (Relationship Modeling Network)	Weight
denounce, undermine, condemn, punish, oppose	5.46%	seem, thing, though, too, especially	11.79%
leave,tell,ask,know,want	4.93%	range, information, supply, infrastructure, value	9.42%
differ, indicate, affect, regard, determine	4.64%	theresa, emmanuel, yemen, poland, lebanon	8.24%
strengthen, enhance, improve, develop, boost	4.56%	negotiate, establish, impose, propose, negotiation	7.92%
hit, cut, end, kick, beat	4.05%	terrorism, militant, terror, terrorist, condemn	7.54%
buy, manufacture, use, brand, purchase	2.59%	propose, woman, represent, indians, spend	1.69e-09
win, defeat, beat, title, match	2.59%	de, defense, kremlin, cite, pressure	1.62e-09
receive, express, praise, send, acknowledge	2.58%	car, de, bomb, little, note	5.39e-10
offer, provide, deliver, feature, guarantee	2.46%	cost, cite, compare, co, m	2.94e-10
launch, announce, unveil, release, celebrate	2.36%	aid, hand, small, public, round	2.21e-10

Table 2: Relation descriptors of the most/least frequent five relations by *LARN* and *RMN* and their average weights in news articles. The relations generated from *LARN* are more semantically meaningful.



(a) US-China's relation trends by *LARN*.

(b) US-China's relation trends by *RMN*.

Figure 2: Temporal trends of top three relations between US and China based on *LARN* (Figure 2a), in comparison with results from *RMN* (Figure 2b). We highlight key events during this time period under the x-axes. Similar visualizations are also used in human evaluations. More figures are provided in the appendix.

4.1 Understanding Model Outputs

Given a set of time-stamped news articles and a list of nations of interest, both models provide a set of relation descriptors, where each one **defines** a type of relation and a temporal trend analysis of these descriptors that shows how the relation **evolves**.

Relation descriptors. Table 2 shows the top five and bottom five descriptors from *LARN* and *RMN* sorted by the average weights over all news articles related to the most frequently mentioned eight nation pairs.[5] By using predicates to describe relations, our descriptors seem to contain more semantically meaningful words. For instance, the top relation in *RMN* consists of exclusively non-content words. Another interesting advantage of our model is that the five relations with the lowest weight have much higher weights in *LARN* than in *RMN*. This suggests that *RMN* tends to generate "useless" relations that do not show up in the data, while even bottom relations in *LARN* remain useful for describing the data.

Temporal trends. We visualize the temporal trends of the most prominent relations between nation pairs. We further provide our annotated key events as the context to interpret these temporal trends. Figure 2 gives an example for US and China. The top three relations based on *LARN* are "denounce", "strengthen", and "leave",[6] while the top three based on *RMN* are "seem", "range", and "negotiate". We find our model generally aligns better with the key events: for instance, the "denounce" relation peaked around the time that Trump started issuing a series of tariffs based on

---

[5] We focus on the top eight nation pairs to be consistent with human evaluation.

[6] For space reasons, we only include the top word in the relation descriptor.

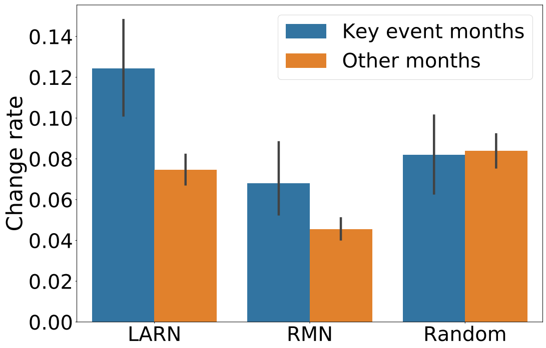


Figure 3: Change rate at key event months vs. other months. Our model demonstrates a clearer relative difference between key event months and other months (66.9%) than *RMN* (50.1%). For the random baseline, we randomly choose the same number of key events and we observe no differences as expected.

our model (Figure 2a), while there do not exist similar fluctuations in Figure 2b based on *RMN*.

4.2 Quantitative Evaluation

We leverage our manually annotated key events to develop a novel automatic metric to evaluate how well the temporal trends from our model are aligned with key events. We define a change rate at each month Δ_t as the weighted average of changes in relation weights in the top three relations:

$$\Delta_t = \sum_{i \in \text{top 3 relations}} w_{t,i} * \frac{|d_{t,i} - d_{t_{prev},i}|}{d_{t_{prev},i}},$$

where $d_{t,i}$ is the average weight for the i-th relation in all the articles published at t, $d_{t_{prev},i}$ is the average weight before t in a window of W preceding months ($\frac{1}{W}\sum_{w=1}^{W} d_{(t-w),i}$), and weight $w_{t,i}$ is the normalized weight for top three descriptors ($\frac{d_{t,i}}{\sum_{j \in \text{top 3 relations}} d_{t,j}}$). Our results are robust to the choices of W, and we set it to 6 for presentation.

We expect change rates to be greater when significant events happen in international politics. Figure 3 compares the change rate at months where key events occurred with other months for eight nation pairs for which we annotated key events. Both models present more abrupt changes when key events occurred. Unlike *RMN*, our model does not have temporal dependencies between relation distributions over time,[7] and thus has a higher discontinuity in general. However, even in relative terms, our model fluctuated more substantially than *RMN* when key events occurred.

	Model		P Value
	LARN	*RMN*	
Descriptor	22	7	8.1e-3
Temporal Trend	106	18	2.3e-16
Nation Pair Matching	38.0 %	45.2 %	0.33

Table 3: Human evaluation results: The first two rows represents the number of votes, while the last row represents % of nation pairs matched correctly with its temporal trend. We did a two-tail binomial test for the relation descriptor and temporal trend evaluation and an independent t-test for the nation pair matching evaluation. We show the p-values in the last column.

We also did a robustness check with another set of independently annotated key events and the results can be found in the appendix. This measure captures whether the model can detect the change points, but does not measure whether the model correctly captures the **semantics** of the key events, i.e., did a negative relation increase after hostile events such as war? To this end, we performed human evaluations.

4.3 Human Evaluation

We hosted three human evaluations with participants from different demographics: undergraduate students from political science classes, graduate students from a computer science department (mostly in NLP), and undergraduate students taking a linguistics class. The total number of participants was 29, roughly equally divided among the three groups. The participants were shown outputs from *RMN* and *LARN*, and asked to choose the output that better aligns with their intuitions. Each participant answered about 10 questions and provided justification for their answers to each question, taking roughly 30 minutes to an hour. Table 3 summarizes the results of our human evaluations.

Relation descriptor evaluation. The participants were shown a list of top five descriptors (as in Table 2) from two models, and prompted to select a set which adequately covers possible relationships that can occur between countries. 75.9% of participants preferred our model.

Temporal trends evaluation. We showed temporal trends between nation pairs annotated with key events, one from *RMN* and the other from *LARN* (as in Figure 2). We asked them to evaluate whether the temporal trends accurately reflect the dynamics in nation-to-nation relationships. Each participant evaluated four randomly chosen nation

---

[7] Please refer to Iyyer et al. (2016)'s Section 3.2.3 for more details on how they incorporated previous time steps.

pairs. The temporal trend from our model was preferred more frequently (85.5% of total responses).

Nation pair matching. We designed a novel task where we showed the participants the other four temporal trends without annotated key events and asked them to match each trend with the corresponding nation pair from the four candidates, based on their world knowledge about nation-nation relations. Each participant did the matching twice, once for *RMN* and once for *LARN*. The participants found correct temporal trends for 45.2% of entity pairs for *RMN*, and 38.0% of entity pairs for *LARN*, when random pairing would yield 25.0%. The difference between two models here is not statistically significant. Most participants found this task very challenging, as they did not know much about the relationship between certain entity pairs (e.g., a participant said “As an American, there’s no way to know the relation between China and India.”). Even political science students do not perform better than the other two groups.

Discussion. Overall the output from our model is preferred by the participants. We found that political science students paid more attention to detail, took a longer time to finish, and were more ambivalent between the performance of the two models. For example, for temporal trends, they preferred our model for 71.4% of the examples, compared to other groups which preferred our model for 90% of the examples. They also preferred *RMN*’s relation descriptors slightly (42.9% selected our model) and commented that a few concepts from *RMN*, like “infrastructure, supply, and value”, are more concrete (e.g., a participant said “I chose the left one (*RMN*), because it is easy to determine and remember the positive of items such as infrastructure, value, and supply. Those have more positive undertones, while it is easy to gauge negative sentiments with ‘terrorism,’ ‘condemn’ and those.”). As *LARN* encodes the background context specific to each nation pair separately, our relation descriptors do not contain such “concrete” concepts. In the next section, we will discuss **contexts** for relations, where these concepts appear in *LARN*.

5 Further Exploration

We present additional qualitative results to showcase the applications of our model. First, we examine the context (nouns) associated with a relation between two nations based on an attention-based mechanism, which *RMN* does not handle. Second, we perform an in-depth analysis to show how our model can reveal regional differences in news coverage on the same topic.

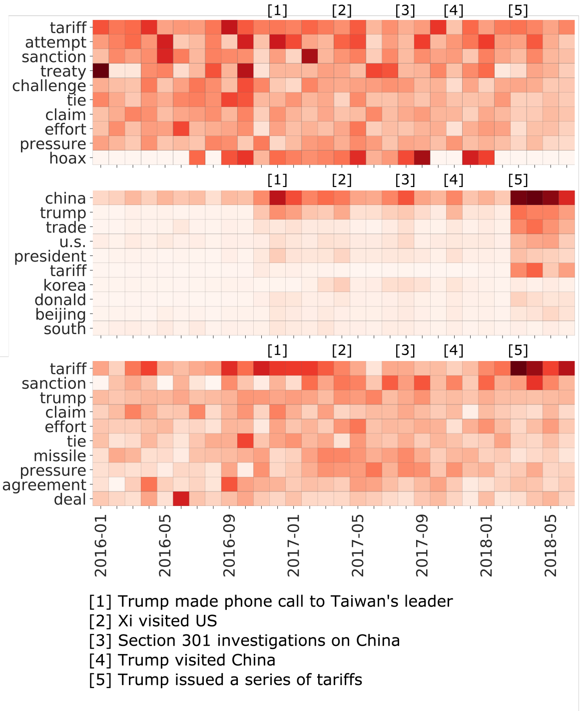


Figure 4: Top contextual words for US-China’s “denounce” relation derived from three approaches. *Top*: Showing word’s avg. attention score in all “denounce” articles each month. *Middle*: Showing word’s frequency in all “denounce” articles each month. *Bottom*: Showing word’s average attention score multiplied by $\log(\text{appearance})$ in each month. All figures are normalized by the global maximum score in the figure.

5.1 Context for Relations

To help users better understand the inferred relations, we offer specific contexts that relate to the inferred relations based on the attention-based mechanism introduced in §3. For each relation, we find articles that place the most weight in that relation, and rank the nouns and proper nouns in those articles by their average attention score (i.e., $\alpha_{n_i^a}$ in §3.3).

The top part of Figure 4 shows the top 10 words for the “denounce” relation in Figure 2a, the temporal trend for US-China relations. Since *RMN* does not support such mechanism, we show the most frequent nouns that occurred in the documents that mention both entities as a baseline (middle part).[8] We find that the attended nouns from our model are more informative than frequent nouns: "tariff" is the most attended word; words such as "sanction", "treaty", and "pressure" also show up, while the frequency baseline centers around words like "China" and "president". As we noticed that the frequency baseline can capture alignment with key events, we incorporate attention score and frequency in the bottom part of Figure 4. This augmented version captures informative words (e.g., "tariff", "sanction", and "missile") [9] and closely aligns with the key events.

5.2 Regional Differences in News Coverage

To further demonstrate the utility of our model, we explore regional differences in news coverage, as "it is possible to build real knowledge by comparing perspectives from different social contexts."[10] This also relates to the longstanding literature on framing in news coverage, i.e., "selecting some aspects of a perceived reality and make them more salient to promote problem definition/interpretation" (Entman, 1993; Chong and Druckman, 2007).

We picked two countries, Singapore and US, to study US-China relations.[11] Using country source of media outlets in the NOW corpus, we found 10K articles from Singaporean media and 5.7K articles from US media on US and China.

Table 4 shows the top five relations sorted by their absolute weight differences between US media outlets and Singaporean media outlets. Singaporean media more frequently use "positive" descriptors such as "strengthen" and "purchase", whereas US media report negative relations such as "denounce" and "criticize" more frequently. Table 5 shows two example sentences from articles with the most weight in the "denounce" relation. Even though two media sources are focusing on events leading to the same type of relation, a reader who mainly consume news articles in Singapore would get a clearly different impression of US-China relations from those who read US news.

Relation descriptor	US	Singapore
strengthen, enhance, improve, develop, boost	4.62%	**5.42%**
purchase, sell, pay, buy, cost	3.18%	**3.86%**
denounce, undermine, condemn, punish, oppose	**6.21%**	5.66%
wag, criticise, accuse, tell, criticize	**3.81%**	3.29%
flee, destroy, invade, unmake, expel	**3.70%**	3.32%

Table 4: Top five relations between US and China sorted by absolute weight differences between US media outlets and Singaporean media outlets.

US media: "President Donald Trump is preparing to *impose* a *package* of $60 billion in annual *tariffs* against Chinese products, following through on a longtime *threat* that he says will *punish* China for intellectual property theft and *create* more American jobs."
Singaporean media: It does not look like just a *trade war*, but rather the US is trying to *bully* China and the rest of the world in order to make China *concede* economic resources and development *opportunities* to the US and make the US forever big and strong.

Table 5: Example sentences in top-scoring articles on US-China's "denounce" relation in March 2018. We italicize key predicates (red color) and three most attended nouns (cyan color).

6 Related Work

Prior work (Chambers et al., 2015; Choi et al., 2016; Rashkin et al., 2017) studied entity-entity relations in terms of positive and negative sentiments between them. Similarly, literature on relation extraction (Riedel et al., 2010; Gardner and Krishnamurthy, 2017; Elson et al., 2010; Srivastava et al., 2016) focused on pre-defined relations between a pair of entities in the database schema. In comparison, our work discovers descriptors for relations between entity pairs instead of finding entity pairs matching pre-defined relation schema.

Topic modeling has been an important method to grasp important concepts from a large collection of documents in an unsupervised fashion (Blei et al., 2003; Das et al., 2015; Chang et al., 2009; Schein et al., 2015). Similar to our work, O'Connor et al. (2013) incorporates linguistic insights with topic models to identify event classes and detect conflicts. Our work additionally models

---

[8] We had a comparison between the top figure and the middle figure in the human evaluation, but we found an error in visualization and thus focus on qualitative comparisons.

[9] "missile" points us to another event related to US deploying missile defense system in South Korea, which also impacts US-China relations.

[10] https://www.publicbooks.org/why-an-age-of-machine-learning-needs-the-humanities/

[11] China is not an English speaking country and is thus not in the NOW corpus. Singapore contained the most articles containing both US and China.

the context of relations through nouns and focuses on exploring the potential of neural models.

Most relevant to our work is Iyyer et al. (2016), which suggests *RMN* better capture dynamic relationships in literature than hidden Markov model (Gruber et al., 2007) and LDA (Blei et al., 2003). Recent work extended and applied *RMN* to other settings such as studying user roles in online communities (Wang et al., 2016; Fremann and Szarvas, 2017). Notably, Chaturvedi et al. (2017) suggests HMM with shallow linguistic features (i.e., frame net parses) and global constraints can outperform *RMN* for modeling relations in literature. In this work, we incorporate linguistic insights with *RMN* and apply it to news domain.

Last but not least, researchers have studied the dynamics of media coverage from a wide range of perspectives, ranging from framing (Card et al., 2015; Field et al., 2018), to relationship between ideas (Tan et al., 2017), to quotes of politicians (Niculae et al., 2015; Tan et al., 2018; Leskovec et al., 2009). There is also significant effort for building event databases in political science (Leetaru and Schrodt, 2013), and assisting journalists with tools (Handler and O'Connor, 2017), and dating historical text (Niculae et al., 2014).

7 Conclusion

We investigate the promise of unsupervised neural models for automatically inferring relations between nations. We find that incorporating shallow linguistic information is a simple yet effective strategy for deriving robust and interpretable relations. We develop a novel quantitative evaluation metric for understanding international relations and in-depth human evaluation also confirms the effectiveness of our model. We further show that our models can provide the background of relations using attention score and reveal regional differences for future studies on media framing.

Meanwhile, our work suggests important future directions for using NLP technologies to support individuals in navigating a large collection of news articles. Our participants often find it challenging to infer information simply from temporal dynamics of our inferred relations based on natural language descriptors. It is thus important to incorporate human cognitive preferences in developing such models and provide narratives beyond words such as key events. Furthermore, different populations pay attention to different parts of information. We need to understand the diversity when developing NLP technologies for end users and provide helpful personalized hints to lower the barrier of benefiting from model outputs.

Acknowledgments

We thank the anonymous reviewers, Dallas Card, and Mohit Iyyer for helpful feedback and discussion. We thank all the participants in our human evaluation from LING 3813 at Georgia Tech by Lelia Glass, students at X-lab at University of Washington, and undergraduate students at University of Colorado Boulder. We also thank Andy Baker and Sven Steinmo for recruiting students in their classes. Choi is supported by a Facebook Fellowship.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

John Baylis, Steve Smith, and Patricia Owens. 2017. *The globalization of world politics: an introduction to international relations*. Oxford University Press.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *ACL*.

Nathanael Chambers, Victor Bowen, Ethan Genco, Xisen Tian, Eric Young, Ganesh Harihara, and Eugene Yang. 2015. Identifying political sentiment between nation states with social media. In *EMNLP*.

Allison June-Barlow Chaney, Hanna M. Wallach, Matthew Connelly, and David M. Blei. 2016. Detecting and characterizing events. In *EMNLP*.

Jonathan Chang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Connections between the lines: augmenting social networks with text. In *KDD*.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé. 2017. Unsupervised learning of evolving relationships between literary characters. In *AAAI*.

Eunsol Choi, Hannah Rashkin, Luke S. Zettlemoyer, and Yejin Choi. 2016. Document-level sentiment inference with social, faction, and discourse context. In *ACL*.

Dennis Chong and James N Druckman. 2007. Framing theory. *Annual Review of Political Science*, 10:103–126.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *ACL*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*.

David K. Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *ACL*.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication*, 43(4):51–58.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky, and Yulia Tsvetkov. 2018. Framing and agenda-setting in russian news: a computational analysis of intricate political strategies. In *EMNLP*.

Lea Frermann and György Szarvas. 2017. Inducing semantic micro-clusters from deep multi-view representations of novels. In *EMNLP*.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-vocabulary semantic parsing with both distributional statistics and formal knowledge. In *AAAI*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *AISTATS*.

Abram Handler and Brendan T. O'Connor. 2017. Rookie: A unique approach for exploring news archives. In *Workshop on Data Science + Journalism at KDD*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *NAACL*.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *KDD*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Vlad Niculae, Caroline Suen, Justine Zhang, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Quotus: The structure of political media coverage as revealed by quoting patterns. In *WWW*.

Vlad Niculae, Marcos Zampieri, Liviu P. Dinu, and Alina Maria Ciobanu. 2014. Temporal text ranking and automatic dating of texts. In *EACL*.

Brendan O'Connor, Brandon M Stewart, and Noah A Smith. 2013. Learning to extract international relations from political context. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Hannah Rashkin, Eric Bell, Yejin Choi, and Svitlana Volkova. 2017. Multilingual connotation frames: A case study on social media for targeted sentiment analysis and forecast. In *ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML*.

Aaron Schein, John Paisley, David M Blei, and Hanna Wallach. 2015. Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *KDD*.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M. Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *AAAI*.

Chenhao Tan, Dallas Card, and Noah A. Smith. 2017. Friendships, rivalries, and trysts: Characterizing relations between ideas in texts. In *ACL*.

Chenhao Tan, Hao Peng, and Noah A Smith. 2018. " you are no jack kennedy": On media selection of highlights from presidential debates. In *WWW*.

Alex Wang, William L. Hamilton, and Jure Leskovec. 2016. Learning linguistic descriptors of user roles in online communities. In *NLP+CSS@EMNLP*.

A Appendix

Data: Aliases and Key Events

We use a dictionary of nation aliases to find relevant sentences in the corpus that mentioned at least one nation pair.[12] Table A1 shows the full alias dictionary we used. Table A5 shows the complete list of key events used in this paper.

Nation	Aliases
U.S.	U.S., US, USA, Trump, Obama
China.	China, Chinese, Xi
Syria	Syria, Syrian, Assad
France	France, French, Macron, Hollande
Germany.	Germany, German, Merkel
Canada	Canada, Canadian, Trudeau
Russia	Russia, Russian, Putin
U.K.	U.K., UK, British, Britain, Cameron
India	India, Indian, Modi
Japan	Japan, Japanese, Abe
Iran	Iran, Iranian, Khamenei, Rouhani
Israel	Israel, Israeli, Netanyahu

Table A1: Aliases used for detecting nation mentions.

Preprocessing

We use spaCy to preprocess all sentences (Honnibal and Montani, 2017). We then use a rule based extractor[13] to get verbal predicates with detectable subjects and objects. When verbal predicate has a negation, we take its antonym in WordNet (Fellbaum, 1998) if it exists, ignored otherwise. We also extract nouns and proper nouns in sentences using spaCy's part-of-speech tagging.

We follow the preprocessing steps in Iyyer et al. (2016) for *RMN*. The original *RMN*, which used a fiction dataset, removes the 500 most frequently occurring words and words that occur in fewer than 100 books. We also remove the 500 most frequent words for the input to *RMN*. However, since our news dataset doesn't have the notion of "books", we remove 5000 least common words (out of a 200K vocabulary) for *RMN*. Note that this is not done for our model *LARN*.

Implementation Details

We show all hyperparameters used in our study in Table A2. We generally use the same hyperparameter values as *RMN*, except a few model-specific hyperparameters in Table A2. We also use Xavier initialization (Glorot and Bengio, 2010) for all trainable layers as in the original *RMN*.

Hyperparameter	Value
v_{label}, v_p, v_n dimension	300
p_{drop}	0.5
v_e dimension	50
h_n (*LARN*) dimension	300 + 30
v_{final} dimension	300
recurrent enforcement α (*RMN*)	0.5
number of relations K	30
number of training epochs	15
learning rate	1e-3
orthogonal penalty λ	1e-1
number of negative samples	15
batch size	256

Table A2: Hyperparameters.

Relation Descriptors Postprocessing

In addition to finding the nearest neighboring words to the relation embeddings, the orginal *RMN* also requires a manual filtering process to achieve an interpretable definition of the relation descriptor. To prevent introducing bias in this manual filtering step, we define each relation descriptor directly as its five nearest neighboring words in the input vocabulary. However, since the full vocabulary would contain many uncommon words which could hinder the interpretation of relation descriptors, we limit both models to choose descriptor words from the most frequently occurring 500 words in their own processed input vocabulary (i.e., verb predicates only for *LARN* and all words for *RMN*).

In Table A3, we present the descriptor set by *RMN* and *LARN* without the most common 500 words constraint on descriptor word selection. We find that the descriptor set of our model is robust to this change and does not suffer from a lowered interpretability as much as *RMN*.

Additional Temporal Relation Trends: US-Russia, US-India, US-Syria

See Figure A2, Figure A3, and Figure A4 for three more examples of temporal relation trends between nation pairs.

Additional Influential Background Words:US-Russia, US-India, US-Syria

In Figure A5, Figure A6, and Figure A7, we further show the corresponding attended background words for the top relation descriptor in each of the

---

[12] We did not use coreference resolution to avoid noisy linking.

[13] `https://github.com/NSchrading/intro-spacy-nlp/blob/master/subject_object_extraction.py`

LARN (Linguistically aware relationship network)	Weight	*RMN* (Relationship modeling network)	Weight
denounce, undermine, condemn, decry, legitimize	5.46%	always, something, seem, sense, thought	11.79%
leave,tell,forget,ask,know	4.93%	components, utilization, integrated, optimized, component	9.42%
differ, indicate, extent, affect, imply	4.64%	morroco, theresa, miriam, emmanuel, charlene	8.24%
strengthen, enhance, improve, develop, maximize	4.56%	mandate, authorize, assurances, rescind, ratify	7.92%
hit, fell, slump, edge, cut	4.05%	provocation, bloodshed, reprisals, incursion, hostilities	7.54%
buy, cheap, manufacture, use, shop	2.59%	una, esa, harridan, chicas, hija	1.69e-09
win, clinch, defeat, championship, champion	2.59%	gest, epoque, rol, chiat, tripper	1.62e-09
receive, bestow, redeem, entitle, outpour	2.58%	pharmacologically, offred, condemnable, chapelle, benassi	5.39e-10
offer, provide, deliver, ideal, cater	2.46%	ricoh, powe, mw, sy, hamill	2.94e-10
launch, relaunch, announce, debut, unveil	2.36%	tunstall, booksellers, seiko, pring, reflation	2.21e-10

Table A3: Relation descriptors of the most/least frequent five relations by *RMN* and *LARN* and their average weights in news articles (without constraint on most common words).

three nation pair relations shown in the previous section.

Robustness Check: Another Set of Key Events Annotations

Apart from the key events annotated by the first author, which were used in the change rate analysis in the main paper, the third author also did an independent annotation of the key events as a robustness check. The details of this annotation could found in Table A6. Figure A1 shows the change rate evaluation with respect to this annotation. A similar trend holds for both sets of annotations.

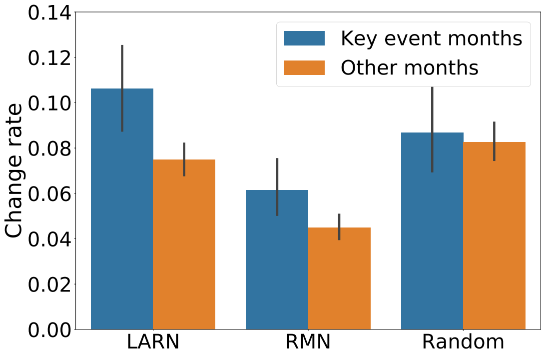


Figure A1: Change rate at key event months vs. other months using an additional set of key events annotations. Our model demonstrates a clearer relative difference between key event months and other months (41.74%) than *RMN* (36.75%). For the random baseline, we randomly choose the same number of key events and we observe no differences as expected.

A Simpler Baseline: Term-frequency Trend of Verbal Predicates

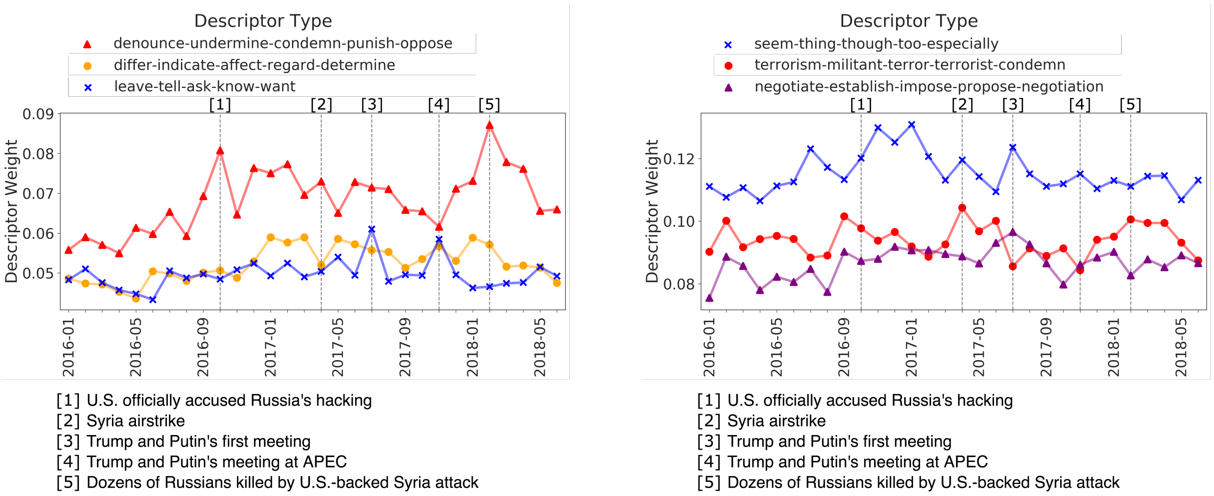
In addition to the comparison with *RMN*, we also checked a simpler baseline by replacing the descriptor weights with term-frequency values of each verbal predicates for each document. Specifically, the term-frequency of a predicate p in document d is $\frac{count(p)}{length(d)}$. Note that *document* refers to all sentences where a nation pair is mentioned. We show some temporal trend examples in Figure A8 and also the change rate evaluation result in Table A4. Note that the scales of these results are different from *LARN* and *RMN*, since the predicate words are much more sparse than the 30 descriptors used in *LARN* and *RMN*.

	Change rate		Δ
	key months	other months	
LARN	12.44 %	7.45 %	66.91 %
RMN	6.81 %	4.53 %	50.07 %
Predicate TF	69.86 %	62.36 %	12.02 %

Table A4: Change rate at key event months vs. other months with the predicate term-frequency baseline, compared to *LARN* and *RMN*.

Nation pair	Key events
U.S.-China	(2016-12) Trump made phone call to Taiwan's leader; (2017-04) Xi visited U.S.; (2017-08) Section 301 investigations on China; (2017-11) Trump visited China; (2018-03) Trump started issuing a series of tariffs.
U.S.-Russia	(2016-10) U.S. officially accused Russia's hacking; (2017-04) Syria airstrike; (2017-07) Trump and Putin's first meeting; (2017-11) Trump and Putin's meeting at APEC; (2018-02) Dozens of Russians killed by U.S.-backed Syria attack.
U.S.-Syria	(2016-10) U.S. suspended Syria ceasefire talk; (2017-04) Khan Shaykhun chemical attack and Shayrat missile strike; (2017-10) ISIS 'capital' captured; (2017-11) ISIS's defeat and aftermath; (2018-02) Battle of Khasham.
U.S.-U.K.	(2016-06) U.K. Brexit vote; (2017-01) Therasa May visited U.S.; (2017-12) Trump set a controversial visit to U.K.
U.S.-Canada	(2017-01) Trump said Nafta renegotiation to be started; (2017-04) Trump imposed tariff on Canadian lumber; (2018-06) Canada fought back with retaliatory tariff on U.S. products.
U.S.-India	(2016-06) Modi visited U.S. and met Obama; (2016-12) Trump made a complimentary phone call to Pakistan; (2017-06) Modi visited U.S. and met Trump; (2017-10) U.S. Secretary Of State Tillerson visited India; (2018-06) India imposed retaliatory tariffs on U.S.
U.S.-Japan	(2016-05) Obama gave memorial speech at Hiroshima with Japanese PM Abe; (2016-12) Abe visited Pearl Harbor; (2017-02) Abe visited Washington and met Trump; (2017-11) Trump visited Japan and met Abe; (2018-03) Trump accepted North Korea's invitation for direct nuclear talks.
China-India	(2016-04) Minister of Foreign Affairs meeting; (2016-11) China and India's joint military drill; (2017-02) India to develop a new missile; (2017-05) India refused to attend Belt and Road Summit; (2017-06) Doklam border standoff started; (2017-08) Doklam border standoff ended; (2017-11) China and India's WMCC meeting; (2018-03) China and India to boost trade.

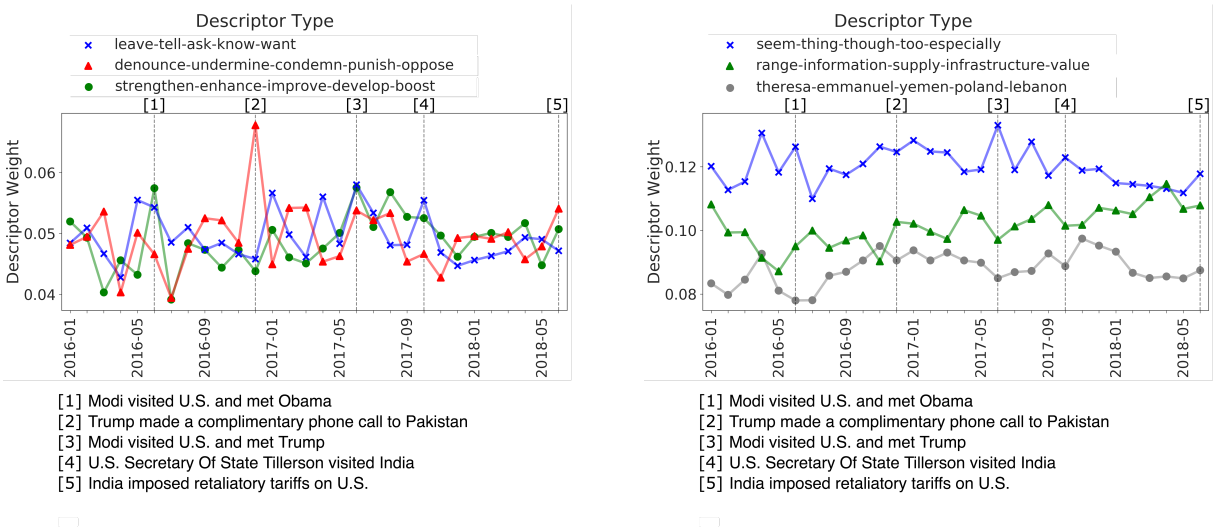
Table A5: Key events annotations and references.



(a) US-Russia's relation trends by *LARN*.

(b) US-Russia's relation trends by *RMN*.

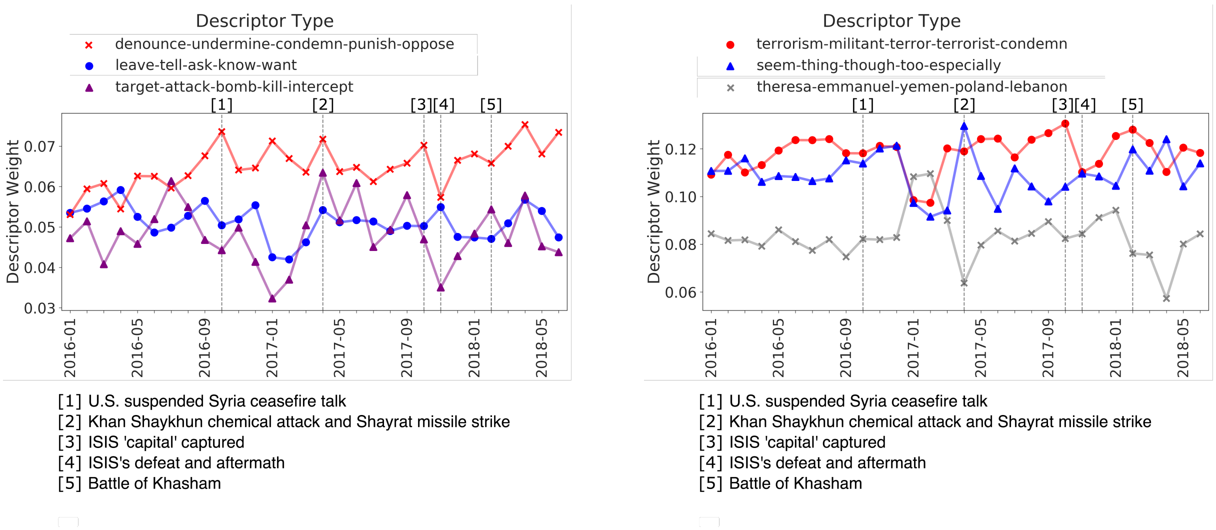
Figure A2: Temporal trends of top three relations between U.S. and Russia based on *LARN* (Figure A2a), in comparison with results from *RMN* (Figure A2b).



(a) US-India's relation trends by *LARN*.

(b) US-India's relation trends by *RMN*.

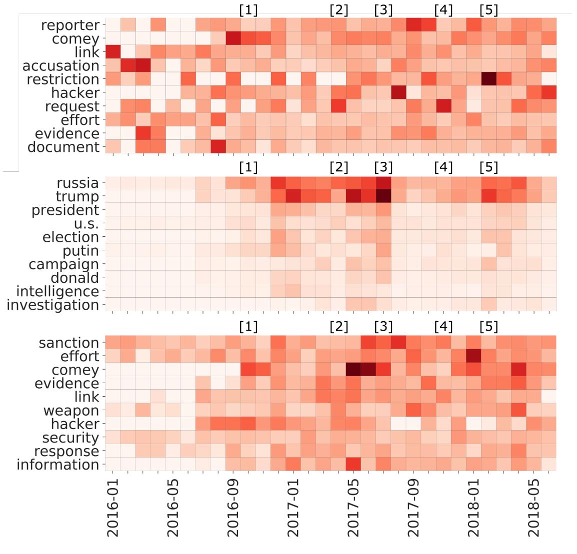
Figure A3: Temporal trends of top three relations between U.S. and India based on *LARN* (Figure A3a), in comparison with results from *RMN* (Figure A3b).



(a) US-Syria's relation trends by *LARN*.

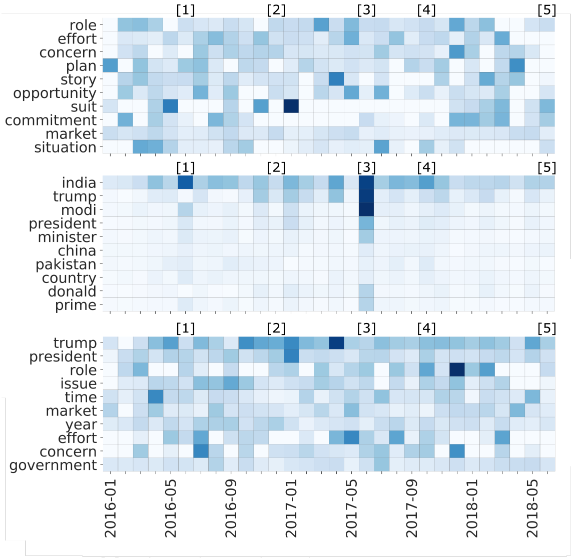
(b) US-Syria's relation trends by *RMN*.

Figure A4: Temporal trends of top three relations between U.S. and Syria based on *LARN* (Figure A4a), in comparison with results from *RMN* (Figure A4b).



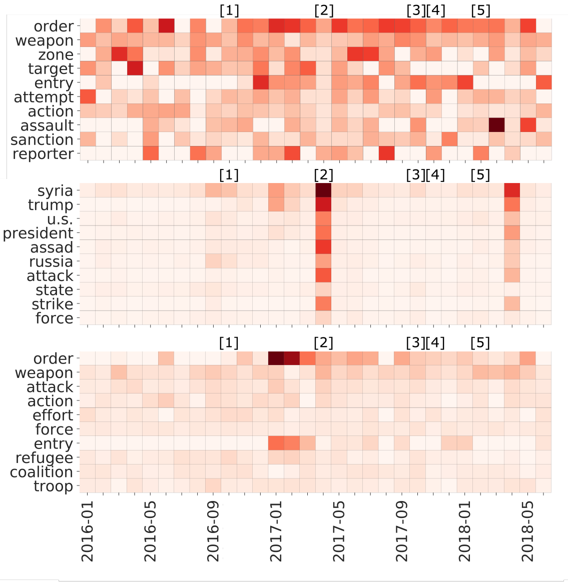
[1] U.S. officially accused Russia's hacking
[2] Syria airstrike
[3] Trump and Putin's first meeting
[4] Trump and Putin's meeting at APEC
[5] Dozens of Russians killed by U.S.-backed Syria attack

Figure A5: Influential words for US-Russia's "denounce" relation derived from three approaches. *Top*: Showing word's average attention score in all "denounce" articles each month. *Middle*: Showing word's appearance frequency in all "denounce" articles each month. *Bottom*: Showing word's average attention score multiplied by $\log(\text{appearance})$ in each month. All figures are normalized by the global maximum score in the figure.



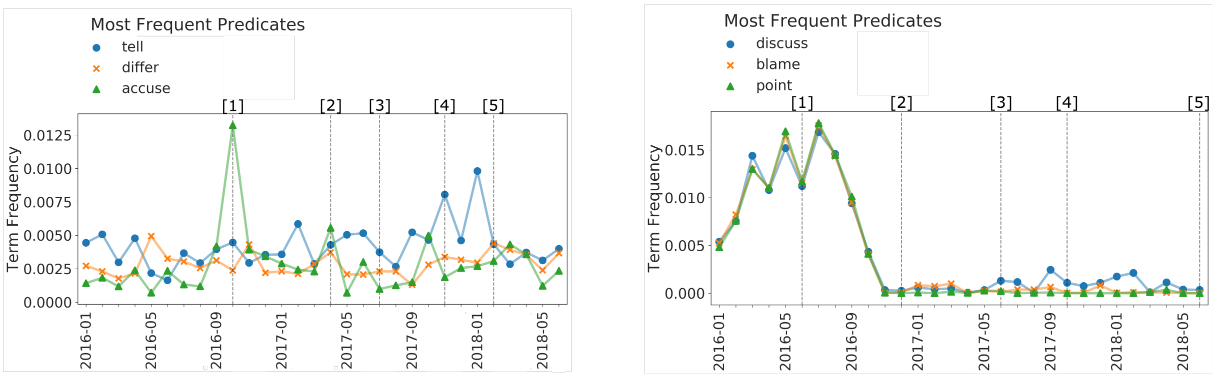
[1] Modi visited U.S. and met Obama
[2] Trump made a complimentary phone call to Pakistan
[3] Modi visited U.S. and met Trump
[4] U.S. Secretary of State Tillerson visited India
[5] India imposed retaliatory tariffs on U.S.

Figure A6: Influential words for US-India's "leave" relation derived from three approaches. *Top*: Showing word's average attention score in all "leave" articles each month. *Middle*: Showing word's appearance frequency in all "leave" articles each month. *Bottom*: Showing word's average attention score multiplied by $\log(\text{appearance})$ in each month. All figures are normalized by the global maximum score in the figure.



[1] U.S. suspended Syria ceasefire talk
[2] Khan Shaykhun chemical attack and Shayrat missile strike
[3] ISIS 'capital' captured
[4] ISIS's defeat and aftermath
[5] Battle of Khasham

Figure A7: Influential words for US-Syria's "denounce" relation derived from three approaches. *Top*: Showing word's average attention score in all "denounce" articles each month. *Middle*: Showing word's appearance frequency in all "denounce" articles each month. *Bottom*: Showing word's average attention score multiplied by $\log(\text{appearance})$ in each month. All figures are normalized by the global maximum score in the figure.



[1] U.S. officially accused Russia's hacking
[2] Syria airstrike
[3] Trump and Putin's first meeting
[4] Trump and Putin's meeting at APEC
[5] Dozens of Russians killed by U.S.-backed Syria attack

(a) US-Russia's relation

[1] Modi visited U.S. and met Obama
[2] Trump made a complimentary phone call to Pakistan
[3] Modi visited U.S. and met Trump
[4] U.S. Secretary Of State Tillerson visited India
[5] India imposed retaliatory tariffs on U.S.

(b) US-India's relation

Figure A8: Temporal trend examples by the predicate term-frequency baseline.

Nation pair	Key events
U.S.-China	(2016-02) China Sends Missiles to Contested South China Sea Island; (2016-06) Xi urges caution over THAAD deployment in South Korea; (2016-12) Trump-Tsai call; (2017-02) Trump affirms One China policy; (2017-04) Xi visits US; (2017-11) Trump visited Beijing; (2018-03) tariff targets China; (2018-04) China retaliates.
U.S.-Russia	(2016-02) U.S.-Russian deal on a "cessation of hostilities"; (2016-09) Russia and the U.S. announce joint peace plan for Syria; (2016-10) Russian interference with election; (2017-01) The U.S. Office of the Director of National Intelligence releases a declassified version of the Intelligence Communitys assessment that "Putin ordered an influence campaign in 2016 aimed at the U.S. presidential election"; (2017-04) US airstrike Syria; (2017-05) Comey fired and Trump met Russian ambassadors; (2017-07) Trump met Putin at G20 summit; (2018-04) Missile strikes against Syria.
U.S.-Syria	(2016-02) US and Russia agree to enforce new Syria ceasefire; (2016-09) Another ceasefire agreement broke in the same month; (2016-11) Senior Chief Petty Officer Scott Cooper Dayton, 42, of Woodbridge, Virginia, is killed in an improvised explosive device blast near the northern Syrian town of Ayn Issa, becoming the first American casualty in combat in the fight against IS in Syria.; (2017-01) Trump orders ban on Syrian refugees; (2017-04) Trump orders strikes against Syria in response to chemical attack; (2017-07) Another ceasefire agreement that involved united states; (2018-03) U.S. Master Sgt. Jonathan J. Dunbar is killed by a roadside bomb attack in Syria alongside a British serviceman; (2018-04) The US, The U.K. and France carried out missile strikes against Assad's compounds in response to the Douma chemical attack.
U.S.-U.K.	(2016-04) Obama and Cameron joint news conference; (2017-01) May visit Trump; (2017-02) Parliament debate trump visit; (2017-09) May visit US for UN general assembly; (2018-04) Missile strikes in Syria.
U.S.-Canada	(2016-03) Trudeau visited the White House for an official visit and state dinner on March 10, 2016; (2017-02) Prime Minister Trudeau and President Trump formally met for the first time at the White House on February 13, 2017; (2017-06) The Trudeau government announced that Canada would continue to support coalition operations; (2018-03) President Donald J. Trump announced he will apply across-the-board tariffs, or import taxes, on steel and aluminum; (2018-05) Canada, Mexico, and the EU became subject to the steel and aluminium tariffs later in an announcement on May 31, 2018; (2018-06) Trump comments at G7 summit.
U.S.-India	(2016-02) The Obama administration notified the US Congress that it intended to provide Pakistan eight nuclear-capable F-16 fighters and assorted military goods; (2016-06) Modi visits Obama; (2016-08) U.S., India sign military logistics agreement; (2017-06) Modi visits Trump; (2017-09) Mattis visits India; (2017-10) Secretary of State Rex Tillerson visits India; (2018-03) The India-US 2 + 2 Dialogue postponed.
U.S.-Japan	(2016-05) Abe meets Obama; (2016-07) Abe, U.S. commander agree to carry out defense guidelines in steady manner; (2016-11) Abe meets Trump in New York; (2016-12) U.S. Returns Some Okinawa Land to Japan, but Unease Endures; Abe visits Pearl Harbor; (2017-01) Trump abandons TPP; (2017-02) Abe and Trump presented a united front on dealing with Pyongyang's nuclear weapon test and multiple missile launches; (2017-04) Japan, U.S. hold missile defense drill in Sea of Japan; (2017-09) Japan, U.S., India vow to work together on strategic port development as China flexes clout; (2017-11) Trump visits Japan; (2018-02) Pence visits Japan; (2018-04) Abe visits US.
China-India	(2016-06) Tashkent on the sidelines of the Shanghai Cooperation Organization Summit; (2016-09) G20 and East Asia Summit; (2016-10) Modi meets Chinese president Xi Jinping on the sidelines of the Goa BRICS Summit; (2017-05) Indias decision to boycott the Belt and Road Initiative (BRI) summit held in Beijing in May; (2017-06) Chinese troops with construction vehicles and road-building equipment began extending an existing road southward in Doklam, a territory which is claimed by both China as well as India's ally Bhutan.; (2017-08) China and India reached a consensus to put an end to the border stand-off; (2017-09) Xi Jinping Meets with Prime Minister Narendra Modi of India; (2017-11) India joins QUAD; (2018-03) China-India border affairs meeting held in New Delhi; (2018-06) Xi meets Modi.

Table A6: Extra key events annotations for robustness check.

Improving Human Text Comprehension through Semi-Markov CRF-based Neural Section Title Generation

Sebastian Gehrmann[1,2], **Steven Layne**[2,3], **Franck Dernoncourt**[2]
[1]Harvard SEAS, [2]Adobe Research, [3]University of Illinois Urbana Champaign
`gehrmann@seas.harvard.edu`, `solayne2@illinois.edu`,
`franck.dernoncourt@adobe.com`

Abstract

Titles of short sections within long documents support readers by guiding their focus towards relevant passages and by providing anchor-points that help to understand the progression of the document. The positive effects of section titles are even more pronounced when measured on readers with less developed reading abilities, for example in communities with limited labeled text resources. We, therefore, aim to develop techniques to generate section titles in low-resource environments. In particular, we present an extractive pipeline for section title generation by first selecting the most salient sentence and then applying deletion-based compression. Our compression approach is based on a Semi-Markov Conditional Random Field that leverages unsupervised word-representations such as ELMo or BERT, eliminating the need for a complex encoder-decoder architecture. The results show that this approach leads to competitive performance with sequence-to-sequence models with high resources, while strongly outperforming it with low resources. In a human-subjects study across subjects with varying reading abilities, we find that our section titles improve the speed of completing comprehension tasks while retaining similar accuracy.

1 Introduction

Section titles in long documents that explain the content of the section improve the recall of content (Dooling and Lachman, 1971; Smith and Swinney, 1992) while simultaneously increasing the reading speed (Bransford and Johnson, 1972). Additionally, they can provide a context to allow ambiguous words to be understood more easily (Wiley and Rayner, 2000) and to better understand the overall text (Kintsch and Van Dijk, 1978). However, most documents do not include titles for short segments or only provide a very abstract description of their topics, e.g. "Geography" or "Introduction". This makes them more inaccessible especially to readers with less developed reading skills, who have trouble identifying relevant information in text (Englert et al., 2009) and therefore more strongly rely on text-markups (Bell and Limber, 2009).

Lascaux cave complex discovered	When another old cave is discovered in the south of France, it is not usually news. Rather, it is an ordinary event. Such discoveries are so frequent these days that hardly anybody pays heed to them. However, when the *Lascaux cave complex* was *discovered* in 1940, the world was amazed. Painted directly on its walls were hundreds of scenes showing how people lived thousands of years ago. The scenes show people hunting animals, such as bison or wild cats. Other images depict birds and, most noticeably, horses, which appear in more than 300 wall images, by far outnumbering all other animals.
Paintings exposed to destructive action	Early artists drawing these animals accomplished a monumental and difficult task. They did not limit themselves to the easily accessible walls but carried their painting materials to spaces that required climbing steep walls or crawling into narrow passages in the Lascaux complex. Unfortunately, the *paintings* have been *exposed to* the *destructive action* of water and temperature changes, which easily wear the images away. Because the Lascaux caves have many entrances, air movement has also damaged the images inside. Although they are not out in the open air, where natural light would have destroyed them long ago, many of the images have deteriorated and are barely recognizable.
Site closed to tourists	To prevent further damage, the *site* was *closed to tourists* in 1963, 23 years after it was discovered.

Figure 1: Output example from our model (left) for an out-of-domain text (right).

This paper introduces an approach to generate section titles by extractively compressing the most salient sentence of each paragraph, as shown in Figure 1. While there has been much recent work on abstractive headline generation from a single sentence (Nallapati et al., 2016), abstractive models require larger datasets, which are not available in many domains and languages. Moreover, abstractive text-generation models tend to generate incorrect information for complex inputs (See et al., 2017; Wiseman et al., 2017). Misleading headlines can have unintended effects, affecting readers' memory and reasoning skills and even bias them (Ecker et al., 2014; Chesney et al., 2017). Especially in times of sensationalism and click-baiting, the unguided generation of titles can be considered unethical and we thus focus on the investigation of deletion-only approaches to title-generation. While this restricts this approach to languages that, similar to English, do not

Proceedings of NAACL-HLT 2019, pages 1677–1688
Minneapolis, Minnesota, June 2 - June 7, 2019.

lose grammatical soundness when clauses are removed, this approach is highly data-efficient and preserves the original meaning in most cases.

We approach the problem with a two-part pipeline where we aim to generate a title for each paragraph of a text, as illustrated in Figure 2. First, a SELECTOR selects the most salient sentence within a paragraph and then the COMPRESSOR compresses the sentence. The selector is an extractive summarization algorithm that assigns a score to each sentence corresponding to its likelihood to be used in a summary (Gehrmann et al., 2018). Algorithms for word deletion typically rely on linguistic features within a tree-pruning algorithm that identifies which phrases can be excluded (Filippova and Altun, 2013). Following recent work that shows the efficiency of contextual span-representations (Lee et al., 2016; Peters et al., 2018), we develop an alternative approach based on a Semi-Markov Conditional Random Field (SCRF) (Sarawagi and Cohen, 2005). The SCRF is further extended by a language model that ranks multiple compression candidates to generate grammatically correct compressions.

We evaluate this approach by comparing it to strong sequence-to-sequence baselines on an English sentence-compression dataset and show that our approach performs almost as well on large datasets while outperforming the complex models with limited training data. We further show the results of a human study to compare the effects of showing no section titles, human-generated titles, and titles generated with our method. The results corroborate previous findings in that we find a significant decrease in time required to answer questions about a text and an increase in the length of summaries written by test subjects. We also observe that the extractive algorithmic titles have a stronger effect on question answering tasks, whereas abstractive human titles have a stronger effect on the summarization task. This indicates that the inherent differences in how humans and our approach summarize the content of a section play a major role in how reading comprehension is affected.

2 Methods

2.1 SELECTOR

To select the most important sentence, we adapt an approach to the problem of content-selection in summarization, which has been shown to be effective and data-efficient (Gehrmann et al., 2018). An advantage of this approach over other extractive summarizers (e.g. Zhou et al. (2018)) is that those often model dependencies between selected sentences, which is not applicable to this problem since we only aim to extract a single sentence.

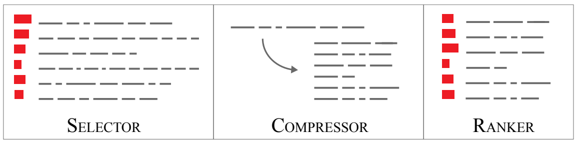


Figure 2: Overview of the three steps. The SELECTOR detects the most salient sentence. Then, the COMPRESSOR generates compressions and the RANKER scores the them.

Let $x = x_1, \dots, x_n$ denote a sequence of words within a paragraph, and $y = y_1, \dots, y_m$ a multi-sentence summary with $n \gg m$. Further, let $t = t_1, \dots, t_n$ be a binary alignment variable where t_i=1 iff $x_i \in y$. Using this alignment, the word-saliency problem is defined as learning a SELECTOR model that maximizes $\log p(t|x) = \sum_{i=1}^{n} \log p(t_i|x)$. Using this model, we calculate the relevance of a sentence $sent := x_{start}, \dots, x_{end}$, with $1 \leq start < end \leq n$, with a saliency function defined as

$$\text{saliency}(sent) = \frac{1}{|sent|} \sum_{i=1}^{|sent|} p(t_i|sent).$$

The sentence selection problem thus reduces to sentence with the most relevant words within a paragraph $para$,

$$\underset{\text{sent} \in \text{para}}{\text{argmax}} \ \text{saliency}(sent).$$

We first represent each word using two different embedding channels. The first is a contextual word representation using ELMo (Peters et al., 2018), and the second one uses GloVe (Pennington et al., 2014). Preliminary experiments corroborated the findings by Peters et al. that the combination of the embeddings help the model converge faster, and perform better with limited training data. Both embeddings for a word x_i are concatenated into one vector e_i, and used as input to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). Finally, the output of the LSTM h_i is used to compute the probability that a word is selected $\sigma(W_s^T h_i + b_s)$ with the trainable parameters W_s and b_s.

2.2 COMPRESSOR

We next define the problem of deletion-only compression of a single sentence. For simplicity of notation, let $x_1, \ldots, x_n$ refer to the words within a single sentence, and $y_1, \ldots, y_n$ be a binary indicator whether a word is kept in the compressed form. The compression $c(x, y)$ becomes

$$\mathrm{c}(x, y) = x_i \forall (x_1, y_1), \ldots, (x_n, y_n) \text{ iff } y_i = 1.$$

The challenge here is that choices are not made independently from another. Consider the sentence *The round ball flew into the net last weekend*, which should be compressed to *Ball flew into net*. Here, the choice to include *into* depends on first selecting the corresponding verb *flew*. One approach to this problem is to use an autoregressive sequence-to-sequence model, in which choices are conditioned on preceding ones. However, these models typically require too many training examples for many languages or domains. Therefore, we relax the problem by assuming that it obeys the Markov property of order L, and train a COMPRESSOR model to maximize $p(y|x) = \sum_{i=1}^{n} \log p(y_i | \mathbf{x}, y_{i-L:i-1})$. To still retain grammaticality, we define the additional problem of estimating the likelihood of a compression $p(\mathbf{y})$ with a RANKER model, described below.

We compare multiple approaches to deletion-based sentence compression. Throughout, we apply the same embedding as for the SELECTOR. Since a grammatical compression problem relies on the underlying linguistic features (Filippova and Altun, 2013), we process the source documents with spaCy (Honnibal and Montani, 2017) to include the following features in addition to contextualized word embeddings: (1) Part-of-speech tags, (2) Syntactic dependency tags, (3) original word shape, (4) Named entities. Each of these information is encoded in an additional embedding that is concatenated to the word embeddings. Over the final embedding vector, we use a bidirectional LSTM to compute the hidden representations $h_1, \ldots, h_n$ for this task.

Naive Tagger The simplest approach we consider is a naive tagger similar to the SELECTOR. We assume full independence between values in y. The probability $p(y_i{=}1|x)$ is computed as $\sigma(W_c^T w_1 + b_c)$ with trainable parameters W_c and b_c.

Conditional Random Field Compressed sentences have to remain grammatical, which implies a dependence between values in y. Without any restrictions on the dependence between choices, it is intractable to marginalize over all possible y with a scoring function for a given (x, y) pair,

$$p(y|x) = \frac{\exp \mathrm{Score}(x, y)}{\sum_{y'} \exp \mathrm{Score}(x, y')}. \tag{1}$$

Therefore, we assume that only neighboring values in y are dependent, and apply a linear-chain CRF (Lafferty et al., 2001) that uses h as its features to this problem.

We define a scoring function $\mathrm{Score}(x, y) = \sum_i \log \phi_i(x, y)$ that computes the (log) potentials at a position i using a function ϕ. The emission potential for a word x_i is computed as $\phi_i^{\mathrm{E}}(x, y) = W_{e2}(\tanh(W_{e1} h_i + b_{e1})) + b_{e2}$, using only the local information. The transition potential ϕ_i^{T} depends on the previous and current choice (y_{i-1}, y_i), and can be looked up in a 2 × 2 matrix that is learned during training. The complete scoring function can be expressed as

$$\mathrm{Score}(x, y) = \sum_{i}^{|y|} \left(\phi_i^{\mathrm{E}}(x, y) + \phi_i^{\mathrm{T}}(y_{i-1}, y_i) \right).$$

During training, we can minimize the negative log-likelihood in which the partition function is computed with the forward-backward algorithm. During inference, this formulation allows for exact inference with the Viterbi algorithm.

Semi-Markov Conditional Random Field Although compressed sentences should often include entire phrases, the CRF does not take into account local dependencies beyond neighboring words. Therefore, we relax the Markov assumption and score longer spans of words. This can be achieved with a SCRF (Sarawagi and Cohen, 2005). Following a similar approach as Ye and Ling (2018), let $s = \{s_1, s_2, \ldots, s_p\}$ denote the segmentation of x. Each segment s_i is represented as a tuple $\langle start, end, \tilde{y}_i \rangle$, where $start$ and the end denote the indices of the boundaries of the phrase, and $\tilde{y}$ the corresponding label for the entire phrase. To ensure the validity of the representation, we impose the restrictions that $start_1 = 1$, $end_p = |x|$, $start_{i+1} = end_i + 1$, and $start_i \leq end_i$. Additionally, we set a fixed maximum length L. We extend Eq. 1 to account for a segmentation s instead of individual tags such that

$$p(s|x) = \frac{\exp \text{Score}(s, x)}{\sum_{s'} \exp \text{Score}(s', x)},$$

marginalizing over all possible segmentations. The CRF represents a special case of the SCRF with $L = 1$. To account for the segment-level information, we extend the emission potential function to

$$\phi_i^{\text{E}}(x, \langle \tilde{y}, \mathit{start}, \mathit{end} \rangle) = \sum_{i=\mathit{start}}^{\mathit{end}} W_e^T h_i', \qquad (2)$$

where h_i' is the concatenation of h_i, $h_{start} - h_{end}$, and a span length embedding e_{len} to account for both individual words and global segment information. We also extend ϕ^T to include transitions to and from longer tagged sequences by representing targets as BIEUO tags (Ratinov and Roth, 2009). This formulation allows for similar training by minimizing the negative log-likelihood.

2.3 RANKER

The inference of the CRF and the SCRF require an estimation of the best possible segmentation $s^* = p(s|x)$, which can be computed using the Viterbi algorithm. However, CRFs and SCRFs are typically employed in sequence-level tagging with no inter-segment dependencies. The sentence compression task differs since a resulting compressed sentence should be grammatical. Therefore, we employ a language model (LM) to rank compression candidates based on the likelihood of the compressed sentence compress(x, s). Namely, we extend the inference target to

$$s_{LM}^* = \underset{s}{\text{argmax}} \; \left(p(s|x) + \lambda p(\text{compress}(x, s))\right),$$

using a weighting parameter λ. Since exact inference for this target is intractable, we approximate this inference by constraining the re-ranking to the K best segmentations according to a K-best Viterbi algorithm.

The RANKER uses the same word embeddings as the COMPRESSOR, and a bidirectional LSTM in order to maximize the probability of a sequence $\sum_i p(x_i|x_{1:i-1}) + p(x_i|x_{i+1:n})$. Using the hidden representation h_i, we compute a distribution over the vocabulary as softmax$(W_l h_i + b_l)$. We additionally use the same weights for word embeddings and W_l, which has been shown to improve language model performance (Inan et al., 2016). We prevent affording an advantage to shorter compressions c during the inference by applying length normalization (Wu et al., 2016) with a length penalty α.

2.4 Sequence-to-Sequence Baselines

The most common approach to summarization and sentence compression uses sequence-to-sequence (S2S) models that learn an alignment between source and target sequences (Sutskever et al., 2014; Bahdanau et al., 2014). S2S models are autoregressive and generate one word at a time by maximizing the probability $p(y|x) = \sum_i p(y_i|x, y_{1:i-1})$. Since this condition is stronger than that of CRF-based approaches, we hypothesize that S2S models perform better with unlimited training data. However, since S2S models need to jointly learn the alignment and generate words, they typically perform worse with limited data.

To test this hypothesis, we define two S2S baselines we compare to our models. First, we use a standard S2S model with attention as described by Luong et al. (2015). In contrast to the other approaches, this model is abstractive and has the ability to paraphrase and re-order words. We constrain these abilities in a second S2S approach as described by Filippova et al. (2015). This model is a sequential pointer-network (Vinyals et al., 2015) and can only generate words from the source sentence. Instead of using an attention mechanism to compute which word to copy, the model enforces a monotonically increasing index of copied words to prevent the re-ordering. We compare both against their reported numbers and our own implementation.

3 Data and Experiments

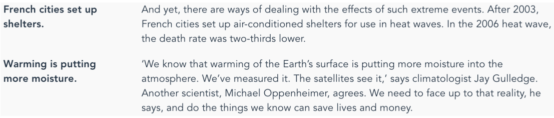
The SELECTOR is trained on the CNN-DM corpus (Hermann et al., 2015; Nallapati et al., 2016), which is the most commonly used corpus for news summarization (Dernoncourt et al., 2018). Each summary comprises a number of bullet points for an article, with an average length of 66 tokens and 4.9 bullet points. The COMPRESSOR is trained on the Google sentence compression dataset (Filippova and Altun, 2013), which comprises 200,000 sentence-headline pairs from news articles. The deletion-only version of the headlines was created by pruning the syntactic tree of the sentence and aligning the words with the headline. The largest comparable corpus Gigaword (Rush et al., 2015) does not include deletion-only headlines.

We limit the vocabulary size to 50,000 words for both corpora. Both SELECTOR and COMPRESSOR use a two-layer bidirectional LSTM with 64 hidden dimensions for each direction, and a word-embedding size of 200. Each linguistic feature is embedded into 30-dimensional space. During training, the dropout probability is set to 0.5 (Srivastava et al., 2014). The model is trained for up to 50 epochs or until the validation loss does not decrease for three consecutive epochs. We additionally halve the learning rate every time the validation loss does not decrease for two epochs. We use Adam (Kingma and Ba, 2014) with AMSGrad (Reddi et al., 2018), an initial learning rate of 0.003, and a $l2$-penalty weight of 0.001. The RANKER uses the same LSTM configuration, but we optimize it with SGD with 0.9 momentum, and an initial learning rate of 0.25.

The S2S models have 64 hidden dimensions for each direction of the encoder, and 128 dimensions for the decoder LSTM. They use one layer, and the decoder is initialized with the final state of the encoder. Our optimizer for this task is adagrad with an initial learning rate of 0.15, and an accumulator value of 0.1 (Duchi et al., 2011).

3.1 Automated Evaluation

In the automated evaluation, we focus on the compression models and first conduct experiments with the full dataset to compute an upper bound on the performance of our approach. This experiment functions as a benchmark to investigate how much better the S2S based approaches perform with sufficient data. The next experiment investigates a scenario, in which data availability is limited and ranges from 100 to 1000 training examples. We compare results with and without linguistic features to further evaluate whether these features improve the performance or whether contextual embeddings are a sufficient representation. In each experiment, we measure precision, recall, and F1-score of the predictions compared to the human reference, as well as the ROUGE-score. We additionally measure the length of the compressions to investigate whether methods delete a sufficient number of words.



French cities set up shelters.

And yet, there are ways of dealing with the effects of such extreme events. After 2003, French cities set up air-conditioned shelters for use in heat waves. In the 2006 heat wave, the death rate was two-thirds lower.

Warming is putting more moisture.

'We know that warming of the Earth's surface is putting more moisture into the atmosphere. We've measured it. The satellites see it,' says climatologist Jay Gulledge. Another scientist, Michael Oppenheimer, agrees. We need to face up to that reality, he says, and do the things we know can save lives and money.

Figure 3: Two paragraphs within the interface of our human evaluation with titles in the left-top margin of a paragraph. A side-effect of the data-efficient deletion-only approach, some titles look ungrammatical, as shown in the second example "*Warming is putting more moisture.*"

3.2 Human Evaluation

We evaluated the effect of our generated titles in a between-subjects study on Amazon Mechanical Turk. We compared three different conditions: no titles, human-generated titles, and algorithmically generated titles by our SCRF+Ranking model. Every participant kept their randomly assigned condition throughout all tasks. We defined the following three tasks to approximately measure the effect of short section titles on (1) retention of text, (2) comprehension of text, and (3) retrieval of information. **(Retention)** We first presented a text and then asked participants three questions about facts in the text. **(Comprehension)** We showed a text and then asked the participants to generate a three-sentence summary of the text. **(Retrieval)** We first presented two questions and then the text, prompting participants to find the answers.

Previous findings indicate that titles help with retention only when presented towards the beginning of a text (Dooling and Mullet, 1973). Thus, we place texts in the left margin at the top of a paragraph as shown in the example in Figure 3. This further avoids interrupting the reading flow of the long text while being integrated into the natural left-to-right reading process. Although reading comprehension is well studied in natural language processing, most datasets focus on machine comprehension (Richardson et al., 2013; Rajpurkar et al., 2016). Therefore, we adapted texts from the interactive reading practice by National Geographic, written by Helen Stephenson[1]. The 33 texts are based on articles and comprise three versions for each story; elementary, intermediate, and advanced, from which we selected intermediate and advanced versions. Topics of the

---

[1] `http://www.ngllife.com/student-zone/interactive-reading-practice`

Model	Features	P		R		F1		Length	
Filippova et al. (2015)	Yes					**82.0**			
S2S w/o copy	No	83.2	±4.5	73.0	±5.9	75.8	±4.1	9.4	±2.9
Sequential Pointer		**87.1**	±4.1	**76.0**	±5.1	**79.1**	±3.5	9.4	±2.6
Naive Tagger		81.4	±3.7	74.6	±6.0	75.5	±3.9	9.7	±2.7
SCRF		85.2	±3.9	71.6	±8.6	73.6	±5.5	9.0	±3.5
SCRF+Ranking		86.1	±3.9	72.9	±8.5	74.8	±5.5	9.1	±3.4
S2S w/o copy	Yes	84.6	±4.1	75.1	±5.6	77.6	±3.8	9.5	±3.1
Sequential Pointer		**89.6**	±3.5	74.2	±5.2	**79.8**	±3.6	8.7	±2.3
Naive Tagger		84.1	±3.5	**75.4**	±5.3	79.5	±3.7	9.5	±2.5
SCRF		86.3	±3.7	73.0	±8.4	79.1	±3.5	9.3	±2.7
SCRF+Ranking		87.2	±3.7	73.9	±7.8	79.6	±3.3	9.1	±2.8

Table 1: Results of our models on the large dataset comprising 200,000 compression examples.

texts include Geography, Science, Anthropology, and History; their length ranges from four to seven paragraphs. Each text is accompanied by reading comprehension questions, which we utilized in the retention and retrieval tasks. We first excluded those questions where the answer was part of either human- or algorithmically generated summary. Of the remaining questions, we randomly selected three questions for each of the retention and retrieval tasks. The same questions were shown in either of the conditions.

Every participant completed six tasks, two for every possible task, one with intermediate and one with advanced difficulty. To account for the different backgrounds of participants, we also asked participants about their perceived difficulty for each task on a 5-point Likert scale. The total time to complete all tasks was limited to 30 minutes, and Turkers were paid $5. In total, we recruited 144 participants who self-reported that they fluently spoke English, uniformly distributed over the three conditions. They answered on average 68.25% of questions correctly and took 16.5 minutes to complete all six tasks. This is approximately 30% faster than the fastest graduate student we recruited for pilot-testing, indicating that Turkers aimed to complete the tasks as fast as possible, possibly by only skimming the text. We omitted results from participants with an answer accuracy of less than 25% (n=21), and excluded individual replies given in under 15 seconds (n=10) or over 10 minutes (n=5), leaving a total of 701 completed tasks. After excluding outliers, the correct answer average was 75.64%, while the time to completion increased by 15 seconds to 16.75 minutes.

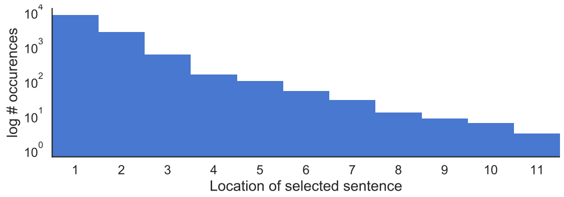


Figure 4: Index of extraction within a paragraph.

4 Results

Selector We compare the performance of the selector against the LEAD-1 baseline that naively selects the first sentence of a news article. This provides a strong comparison since a news article typically aims to summarize its content in the first sentence. LEAD-1 achieves ROUGE (1/2/L)-scores of 27.5/9.6/23.7 respectively. In contrast, our selector achieves scores of 30.2/12.2/26.45 which presents an improvement of over 10% in each category. We illustrate the source of this improvement in Figure 4, which shows the locations of selected sentences and observe a negative correlation between a later location within a text and the probability of being selected. However, in most cases, the first sentence is not the most relevant according to the model.

Unrestricted Data Table 1 shows the results of the different approaches on the large dataset. As expected, the copy-model performs best due to its larger modeling potential. It is closely followed by SCRF+Ranking, which comes within 0.2 F1-score when using additional features. This difference is not statistically significant, given the high variance of the results. Compared to the best reported re-

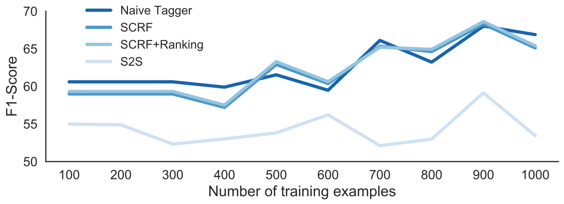


Figure 5: F1-scores of the different models with an increasing number of training examples.

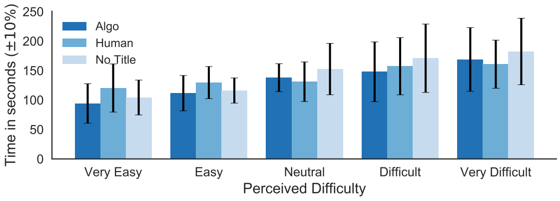


Figure 6: Mean and the 90% confidence interval of the time taken by Turkers to complete the tasks, grouped by the perceived difficulty.

sult in the literature by Filippova et al. (2015), our models perform almost as well, despite the fact that their model is trained on 2,000,000 unreleased datapoints compared to our 200,000. We further observe that all models generate compressed sentences of almost the same length between 8.7 and 9.5 tokens per compression.

The Naive Tagger also achieves comparable performance to the SCRF in F1-Score. To test whether our model leads to a higher fluency compared to it, we additionally measure the ROUGE score. In ROUGE-2, we find that the SCRF+Ranking leads to an increase from 58.1 to 60.1, with an increase in bigram precision by 5 points from 64.5 to 69.3. The Naive Tagger is more efficient at identifying individual words, with a ROUGE-1 of 71.3 when the ranking approach only achieves 68.7. While these differences lead to similar ROUGE-L scores between 69.9 and 70.2, the fact that the ranking-based approach matches longer sequences indicates higher fluency. In an analysis of samples from the Naive Tagger, we found that it commonly omits crucial verb-phrases from compressed sentences.

We show compressions from two paragraphs in the NatGeo data in Figure 3. This example illustrates the robustness of the compression approach to out-of-domain data when including linguistic features. Despite the fact that the example text is not a news article like the training data, it performs well and generates mostly grammatical compressions.

Limited Data We present results on limited data in Figure 5. The results show the major advantage of the simpler training objective. All of the tagging-based models outperform the S2S baselines by a large margin due to their data-efficiency. We did not observe a significant difference between the different tagging approaches in the limited data condition. In our experiments, we found that the S2S models start outperforming the simpler models at around 20,000 training examples. Despite its high F1-Score, the Naive Tagger suffers from ungrammatical output in this condition as well, with the readability scoring significantly lower than the SCRF outputs. We argue that the SCRF+Ranking approach represents the best trade-off of our presented models since it performs well with limited data while performing almost as well as complex models in unlimited data situations. This makes it most flexible to apply to a wide range of tasks.

Human Evaluation In the human study, we notice an immediate effect of the difficulty of texts. Between the intermediate and advanced versions of the texts, the mean time to complete the tasks increases by 8 seconds (from 125 to 133 seconds). Additionally, the mean perceived difficulty increases from 2.40 to 2.55 (in between Easy and Neutral). The largest observed effect of text difficulty is on the accuracy of answers, which decreases from 84.4% to 67.5%, indicating that within a similar time frame, the difficult texts were harder to understand.

We present a breakdown of time spent on a task by perceived difficulty for each of the conditions in Figure 6. There is a positive correlation between perceived difficulty and time spent on a task across all conditions. Interestingly, tasks rated Very Easy and Easy were completed slower in the human condition than in the no-title condition, but faster with the algorithmically generated ones. This effect alleviates in the higher difficulties, in which the no-title condition takes longest. Indeed, a comparison between the algorithmic and no title conditions reveals a decrease in time by 19.8 seconds in the retention task, significant with $p < 0.005$ according to a χ^2 test. Interestingly, we can observe opposite effects in the human condition. Here, the comprehension task is completed 15.4 seconds faster ($p < 0.005$), but the other tasks only show minor effects.

Task	Measure	Intervention	Effect Size	p-value
Retention	Time Taken (sec)	Human	-2.2	0.63
		Algo	-27.1	0.01*
	Accuracy	Human	-0.01	0.07
		Algo	-0.01	0.13
Retrieval	Time Taken (sec)	Human	-0.9	0.87
		Algo	-4.5	0.03*
	Accuracy	Human	+0.01	0.20
		Algo	-0.01	0.15
Comprehension	Time Taken (sec)	Human	-20.9	0.03*
		Algo	-2.6	0.04*
	Summary Length (words)	Human	+8.6	0.02*
		Algo	+5.3	0.03*
	Readability	Human	-0.1	0.24
		Algo	-0.1	0.50
	Relevance	Human	-0.02	0.92
		Algo	+0.01	0.63

Table 2: The causal effects of the human and algorithmic section titles on different measures differ across tasks. All the shown effect sizes are measured in comparison to the baseline without any shown titles. Significant p-values at a 0.05 level are marked with a *.

To further investigate these effects, we analyze the causal effect of our three conditions by measuring the average treatment effect while controlling for both actual and perceived difficulty of the tasks with an Ordinary Least Squares analysis. Whenever possible, we additionally condition on the total time taken. An overview of our tests is presented in Table 2. In the causal tests, we observe similar effects in the retention task – the algorithmically generated titles lead to a decrease in time required for the task with an effect size of 27.1 seconds. In contrast, human-generated titles only lead to a non-significant 2.2-second decrease. We observe a non-significant decrease of approximately 4% in accuracy with added titles. We observed no effect of adding titles on the perceived difficulty of this task. In the retrieval task, added titles result in a weaker effect. The algorithmic titles decrease time by only 4.5 seconds, and the human titles by non-significant 0.9 seconds. Similar to the retention task, there is no significant change in accuracy, with all accuracy levels within 1% from another and we observe no effect on the perceived difficulty.

In the comprehension task, it is not possible to measure accuracy. Instead, we evaluate readability and relevance as judged by human raters on a five-point Likert scale, two commonly used metrics for abstractive summarization (Paulus et al., 2017). We present the average ratings in Table 3 and observe that there is almost no difference in relevance and only a minor (not significant) decrease in readability with either condition. Curiously, the previous effect on speed reverses in this task – algorithmic titles only lead to a 2.6-second decrease in time, while human titles lead to a 20.9-second decrease. Both conditions additionally lead to longer summaries; algorithmic titles by 5.3 words and human titles by 8.6 words. One potential explanation for this behavior could be that subjects copied the presented section titles into the summary text field. This was not the case, since, on average, only 2.8-3.7% of the bigrams in the titles were used in the summaries, across both conditions and difficulties (0.6-1.5% of trigrams, 0.1-0.8% of 4-grams).

	Readability		Relevance	
Condition	μ	σ	μ	σ
No title	4.66	0.65	4.11	0.86
Human	4.55	0.76	4.09	0.95
Algo	4.52	0.72	4.12	1.02

Table 3: Human ratings for human-generated summaries while showing different section titles.

Given the similar relevance scores, we thus argue that presenting the titles leads to more detailed descriptions of the texts. Similarly to the other tasks, the perceived difficulty does not change significantly. However, we note that some subjects noted that there were some ungrammatical generated titles, which is an artifact of the deletion-only approach. Future work may investigate how abstractive approaches that are not restricted to deletion-based approaches can be applied to the same problem.

Overall, the results of the human-subject study reveal an effect that is well studied in the literature. Namely, that the type of title influences what is being remembered about a text (Schallert, 1975), and that different headline styles affect readers in different ways (Lorch Jr et al., 2011). Kozminsky (1977) found that the immediate free recall of information is biased towards topics emphasized in titles. The better performance in memorization tasks in the algorithmic condition can be explained by the fully extractive approach that immediately shows information judged most relevant by the model. In contrast, human-generated titles show a higher level of abstraction and generalization, which is more helpful for the overall comprehension but does not emphasize any piece of information.

5 Related Work

The aim of SCRFs is to learn a segmentation of a sequential input and assigning the same label to an entire segment. While they were originally developed for information extraction (Sarawagi and Cohen, 2005), it is most commonly applied to speech recognition within the acoustic model to improve segmentation between different words (He and Fosler-Lussier, 2015; Lu et al., 2016; Kong et al., 2015). Similar to this work, it has also been shown that coupling an LM with an SCRF can improve segmentation through multi-task training (Lu et al., 2017; Liu et al., 2017). SCRFs have also been applied to sequence tagging tasks, for example, the extraction of phrases that indicate opinions (Yang and Cardie, 2012). In this work, we built upon an approach by Ye and Ling (2018) who recently introduced a hybrid SCRF that uses both word- and phrase-level information. Alternative approaches for similar tasks are CRFs that estimate pairwise potentials rather than using a fixed transition matrix (Jagannatha and Yu, 2016) or high-order CRFs which outperform SCRFs in some sequence labeling tasks (Cuong et al., 2014).

While this work is the first to apply SCRFs to sentence compression, Grootjen et al. (2018) also use extractive summarization techniques to improve reading comprehension by highlighting relevant sentences. Most similar to our compression approach is Hedge Trimmer (Dorr et al., 2003), which compresses sentences through deletion, but uses an iterative shortening algorithm based on linguistic features. Extending this work, Filippova and Altun (2013) apply a similar approach on linguistic features, but learn weights for the shortening algorithm. Both approaches also do not consider the selection of the sentence to be compressed, unlike our proposed model.

6 Conclusion

In this work, we have presented a novel approach to section title generation that uses an efficient sentence compression model. We demonstrated that our approach performs almost as well as sequence-to-sequence approaches with unlimited training data while outperforming sequence-to-sequence approaches in low-resource domains. A human evaluation showed that our section titles lead to strong improvements across multiple reading comprehension tasks. Future work might investigate end-to-end approaches, or develop alternative approaches that generate titles more similar to how humans write titles.

Acknowledgments

We are grateful for the helpful feedback from the three anonymous reviewers. We additionally thank Anthony Colas and Sean MacAvaney for the multiple rounds of feedback on the ideas presented in this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kenneth E Bell and John E Limber. 2009. Reading skill, textbook marking, and course performance. *Literacy research and instruction*, 49(1):56–67.

John D Bransford and Marcia K Johnson. 1972. Contextual prerequisites for understanding: Some investigations of comprehension and recall. *Journal of verbal learning and verbal behavior*, 11(6):717–726.

Sophie Chesney, Maria Liakata, Massimo Poesio, and Matthew Purver. 2017. Incongruent headlines: Yet another way to mislead your readers. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 56–61.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *The Journal of Machine Learning Research*, 15(1):981–1009.

Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. 2018. A repository of corpora for summarization. In *Eleventh International Conference on Language Resources and Evaluation (LREC)*.

D James Dooling and Roy Lachman. 1971. Effects of comprehension on retention of prose. *Journal of experimental psychology*, 88(2):216.

D James Dooling and Rebecca L Mullet. 1973. Locus of thematic effects in retention of prose. *Journal of Experimental Psychology*, 97(3):404.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Ullrich KH Ecker, Stephan Lewandowsky, Ee Pin Chang, and Rekha Pillai. 2014. The effects of subtle misinformation in news headlines. *Journal of experimental psychology: applied*, 20(4):323.

Carol Sue Englert, Troy V Mariage, Cynthia M Okolo, Rebecca K Shankland, Kathleen D Moxley, Carrie Anna Courtad, Barbara S Jocks-Meier, J Christian O'Brien, Nicole M Martin, and Hsin-Yuan Chen. 2009. The learning-to-learn strategies of adolescent students with disabilities: Highlighting, note taking, planning, and writing expository texts. *Assessment for Effective Intervention*, 34(3):147–161.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

FA Grootjen, GE Kachergis, et al. 2018. Automatic text summarization as a text extraction strategy for effective automated highlighting.

Yanzhang He and Eric Fosler-Lussier. 2015. Segmental conditional random fields with deep neural networks as acoustic models for first-pass word recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Abhyuday N Jagannatha and Hong Yu. 2016. Structured prediction models for rnn based sequence labeling in clinical text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 856. NIH Public Access.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Teun A van DijkWalter Kintsch and TA Van Dijk. 1978. Cognitive psychology and discourse: Recalling and summarizing stories. *Current trends in textlinguistics*, page 61.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. *arXiv preprint arXiv:1511.06018*.

Ely Kozminsky. 1977. Altering comprehension: The effect of biasing titles on text comprehension. *Memory & Cognition*, 5(4):482–490.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016.

Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Robert F Lorch Jr, Julie Lemarié, and Russell A Grant. 2011. Three information functions of headings: A test of the sara theory of signaling. *Discourse processes*, 48(3):139–160.

Liang Lu, Lingpeng Kong, Chris Dyer, and Noah A Smith. 2017. Multitask learning with ctc and segmental crf for speech recognition. *arXiv preprint arXiv:1702.06378*.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192.

Diane L Schallert. 1975. Improving memory for prose: The relationship between depth of processing and context. *Center for the Study of Reading Technical Report; no. 005*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Edward E Smith and David A Swinney. 1992. The role of schemas in reading text: A real-time examination. *Discourse Processes*, 15(3):303–316.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Jennifer Wiley and Keith Rayner. 2000. Effects of titles on the processing of text and lexically ambiguous words: Evidence from eye movements. *Memory & Cognition*, 28(6):1011–1021.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational linguistics.

Zhi-Xiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov crf for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, Melbourne, Australia. ACL.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–663.

Unifying Human and Statistical Evaluation for Natural Language Generation

Tatsunori B. Hashimoto*[1,2] **Hugh Zhang***[1] **Percy Liang**[1,2]
(* equal contribution)
[1]Department of Computer Science [2]Department of Statistics
Stanford University
`{thashim,hughz}@stanford.edu` `pliang@cs.stanford.edu`

Abstract

How can we measure whether a natural language generation system produces both high quality and diverse outputs? Human evaluation captures quality but not diversity, as it does not catch models that simply plagiarize from the training set. On the other hand, statistical evaluation (i.e., perplexity) captures diversity but not quality, as models that occasionally emit low quality samples would be insufficiently penalized. In this paper, we propose a unified framework which evaluates both diversity and quality, based on the optimal error rate of predicting whether a sentence is human- or machine-generated. We demonstrate that this error rate can be efficiently estimated by combining human and statistical evaluation, using an evaluation metric which we call HUSE. On summarization and chit-chat dialogue, we show that (i) HUSE detects diversity defects which fool pure human evaluation and that (ii) techniques such as annealing for improving quality actually decrease HUSE due to decreased diversity.

1 Introduction

Generating text is a core part of many NLP tasks such as image captioning (Lin et al., 2014), open-domain dialogue (Sordoni et al., 2015), story generation (Roemmele, 2016), and summarization (Nallapati et al., 2016). However, proper evaluation of natural language generation has proven difficult (Liu et al., 2016; Novikova et al., 2017; Chaganty et al., 2018). A good evaluation metric should not only capture the *quality* of generation, but also the *diversity* of generation, which is especially crucial for creative, open-ended tasks like dialogue or story generation.

Human evaluation, which is often viewed as the gold standard evaluation, captures quality but fails to capture diversity. As an example, for language modeling, a model that directly plagiarizes sentences from the training set would pass the human quality bar but would have zero generalization ability and thus have inadequate diversity. On the other hand, *statistical evaluation*—i.e., perplexity on a reference test set—captures diversity, as it ensures a model must assign reasonable probability to novel sentences, but perplexity provides an inadequate measure of quality (Theis et al., 2015). For example, modifying a perfect model by removing its ability to generate even a single test sentence results in *infinite* perplexity even though the model is still near-perfect. Automatic metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin and Rey, 2004) capture quality better than perplexity but still correlate poorly with human evaluation and fail to capture diversity (Novikova et al., 2017; Chaganty et al., 2018).

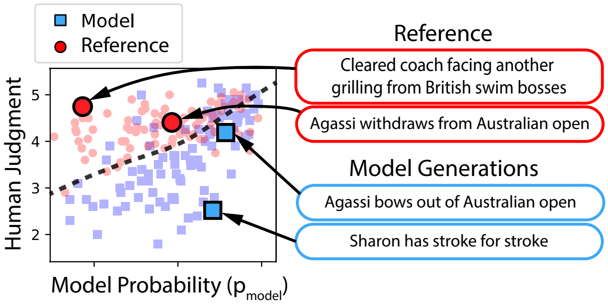


Figure 1: HUSE is twice the classification error of distinguishing reference and generated text based on human judgment scores and model probabilities. HUSE identifies samples with defects in quality (Sharon has stroke . . .) and diversity (Cleared coach facing . . .).

Existing approaches to combining statistical and human evaluation have been ad-hoc, leading to misleading performance measures. A common approach is to measure diversity through the perplexity of a probabilistic model and quality through human evaluation on beam-searched out-

Proceedings of NAACL-HLT 2019, pages 1689–1701
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

puts. This gives the illusion that a single model is high-quality and diverse, while the reality is that it shows we can have either a diverse model (when sampling from the distribution used to compute perplexity) *or* a high-quality model (when beam-searching).

In this paper, we define the idealized evaluation metric as twice the error of the *optimal discriminator* for classifying sentences as coming from the reference distribution or the model (Section 2). If a model generates gibberish (low quality), the optimal discriminator can classify these accurately as coming from the model. If the reference distribution contains sentences the model cannot generate (low diversity), the optimal discriminator can classify these accurately as coming from the reference.

Unfortunately, the optimal discriminator is unavailable. Human discriminators cannot capture diversity effectively, and learned discriminators—e.g., from a Generative Adversarial Network (Goodfellow et al., 2014) or one trained on human judgments (Lowe et al., 2017)—are too unreliable to use for rigorous evaluation.

Our key result (Section 3) is based on the observation that the optimal classifier depends only on two numbers: the probability of a sentence under the model and the probability under the reference distribution. The former can be computed directly from the model, and we show that the latter can be well-approximated by human judgment scores. The resulting two-dimensional space is illustrated in Figure 1. We apply a simple k-nearest neighbor classifier in this space and define Human Unified with Statistical Evaluation (HUSE) as twice the leave-one-out error of this classifier.

We apply HUSE to four natural language generation tasks (Section 5): language modeling, chitchat dialogue, story generation, and summarization. First, we show that human evaluation alone is insufficient to discriminate model generations from the references, leading to inflated estimates of model performance. In contrast, HUSE is able to reveal deficiencies of current models. We also show that common techniques for improving sample quality such as annealing actually increase distinguishability between the model and reference due to losses in diversity.

2 Optimal Discriminator

Consider a natural language generation task where the model is given a context x (e.g., a dialogue history) drawn from some prior $p(x)$ and must output a distribution over possible sentences $p_{\text{model}}(y \mid x)$. We define an idealized evaluation metric based on whether p_{model} is close to a *reference distribution* p_{ref}, which is generally human-generated.[1] Specifically, consider a random variable y drawn from either the reference or the model based on an indicator $z \sim \text{Bernoulli}\left(\frac{1}{2}\right)$:

$$y \mid x, z \sim \begin{cases} p_{\text{ref}}(y \mid x) & \text{if } z = 1 \\ p_{\text{model}}(y \mid x) & \text{if } z = 0. \end{cases} \tag{1}$$

Define L^* to be twice the lowest possible error over any discriminator f that attempts to determine z based on x and y:

$$L^* \overset{\text{def}}{=} 2 \inf_f \mathbb{P}[f(x, y) \neq z]. \tag{2}$$

L^* measures similarity between p_{model} and p_{ref}; it is 0 if p_{model} and p_{ref} are disjoint and 1 if they are identical.[2]

Obstacles. Unfortunately, L^* is unattainable because it requires computing the optimal discriminator. In the spirit of the Turing Test, we could consider using the error rate of a human discriminator f_{hum} instead, often considered the gold standard for evaluation. However, while humans might have knowledge of p_{ref}, they do not have full knowledge of p_{model} and thus would have difficulties determining which sentences a model *cannot* generate.

As a concrete example, suppose p_{ref} placed a uniform distribution over some set S. Without knowledge of p_{model} the most sensible discriminator is to predict $z = 1$ (reference) when $y \in S$. This discriminator achieves the same classification error of 0.5 for both the perfect model $p_{\text{model}} = p_{\text{ref}}$ and one which can only return a single $y \in S$. We could try to reveal p_{model} to humans by showing multiple samples simultaneously, but this is expensive and, as we will later see, unnecessary.

Another option is to learn f over an expressive class of functions such as neural networks on data

---

[1] While some tasks only care about quality and thus only require p_{model} to place mass on *some* high quality y, we demand that p_{model} places mass on *all* high quality y as given by p_{ref}. This diversity is important for open-ended tasks such as dialogue or story generation. Also note that p_{ref} need not be the human distribution, or match the training distribution. It can be defined as the distribution given by experts.

[2] Note that L^* is a linear function of the total variational divergence: $\|p_{\text{model}} - p_{\text{ref}}\|_{\text{TV}} \overset{\text{def}}{=} \sum_{x,y} p(x) \left|p_{\text{model}}(y \mid x) - p_{\text{ref}}(y \mid x)\right| = 1 - L^*$. See Appendix A.1 for details.

sampled from p_{model} and p_{ref}. This is analogous to learning the discriminator in a Generative Adversarial Network (GAN) (Goodfellow et al., 2014) or learning an evaluation metric from human judgments (Lowe et al., 2017). However, as (x, y) are high-dimensional objects, training a good classifier is extremely difficult (and perhaps not significantly easier than solving the original generation problem). Indeed, learned evaluation metrics do not generalize very well (Lowe et al., 2017; Chaganty et al., 2018). Unlike these approaches which seek to replace human evaluation, our focus will instead be on combining human and automatic statistical evaluation to estimate the *optimal* classifier error.

3 Human Unified with Statistical Evaluation (HUSE)

Our key result is that the optimal discriminator depends on (x, y) only through a two-dimensional sufficient statistic (Section 3.1), motivating an approximation which we call HUSE (Section 3.2).

For any feature map ϕ that maps (x, y) to $\phi(x, y) \in \mathbb{R}^d$, define the evaluation score $L(\phi)$ to be twice the error rate of the optimal discriminator that depends on (x, y) *only* through ϕ:

$$L(\phi) \overset{\text{def}}{=} 2 \inf_f \mathbb{P}[f(\phi(x, y)) \neq z]. \tag{3}$$

Note that the evaluation score $L(\phi)$ given by a feature map ϕ optimizes over all functions that depend on ϕ (3). Thus, the more information ϕ contains, the lower $L(\phi)$ is. This has two implications: First, any feature map ϕ yields an (optimistic) *upper bound* on L^* (2), meaning that $L(\phi)$ might be able detect when a model is poor but cannot certify that it is good. Second, adding features to ϕ can only improve this bound.

3.1 Two features suffice

Let us consider the following two-dimensional feature map:

$$\phi_{\text{opt}}(x, y) \overset{\text{def}}{=} \left[p_{\text{ref}}(y \mid x), p_{\text{model}}(y \mid x)\right]. \tag{4}$$

From the arguments above, it is clear that $L(\phi_{\text{opt}}) \geq L^*$, but perhaps more surprisingly, we actually have equality:

Proposition 1. *The two-dimensional feature map* ϕ_{opt} *achieves the optimal discriminator score:* $L(\phi_{opt}) = L^*$.

Proof We compute the true posterior over z given x, y. Since $p(z = 1) = p(z = 0) = \frac{1}{2}$, $p(y \mid x, z = 1) = p_{\text{ref}}(y \mid x)$ and $p(y \mid x, z = 0) = p_{\text{model}}(y \mid x)$, by Bayes' rule:

$$p(z = 1 \mid x, y) = \frac{p_{\text{ref}}(y \mid x)}{p_{\text{ref}}(y \mid x) + p_{\text{model}}(y \mid x)}.$$

The optimal discriminator simply predicts $z = 1$ if $p_{\text{ref}}(y \mid x) > p_{\text{model}}(y \mid x)$ and $z = 0$ otherwise. In other words, the decision boundary is given by $\phi_{\text{opt}}(x, y)_1 > \phi_{\text{opt}}(x, y)_2$. $\square$

More generally, we can obtain this equality with a wider class of ϕ. It will hold exactly for any invertible transformation of ϕ_{opt} (Appendix Corollary 1), and approximately for any ϕ which has high mutual information with ϕ_{opt} (Appendix Theorem 1). This means that we can substitute p_{ref} with noisy, possibly un-normalized estimates and still obtain accurate estimates of L^*.

3.2 HUSE features

While we can directly compute $p_{\text{model}}(y \mid x)$ for many probabilistic models, $p_{\text{ref}}(y \mid x)$ is unattainable, so $L(\phi_{\text{opt}})$ is not computable. However, the *wisdom of the crowds* (Surowiecki, 2004; Ungar et al., 2012) suggests that pooling together the judgments of many humans can often produce surprisingly reliable estimates of real-world probabilities such as $p_{\text{ref}}(y \mid x)$, even if no individual human is particularly reliable. With this motivation, we ask Amazon Mechanical Turk workers to rate a sentence from 1–5 based on how "typical" it is as a way to estimate $p_{\text{ref}}(y \mid x)$. (see Appendix A.3 for more details). We define HJ(x, y) to be the average response over 20 crowdworkers. Figure 2 shows that for a language modeling task on the Reddit corpus,[3] HJ(x, y) strongly correlates with the actual log-frequency of y in the corpus. The high correlation suggests that human judgments HJ(x, y) are a good surrogate for $\log p_{\text{ref}}$.

In addition, we found that rather than using the model probability $p_{\text{model}}(y \mid x)$ directly as a feature, normalizing by sentence length len(y) yielded lower (tighter) scores. We therefore define the HUSE features as follows:

$$\phi_{\text{huse}}(x, y) \overset{\text{def}}{=} \left[\frac{\log p_{\text{model}}(y \mid x)}{\text{len}(y)}, \text{HJ}(x, y)\right], \tag{5}$$

---

[3] We used the Reddit corpus due to crowdworker familiarity, corpus size, and short average sentence length, which results in a wide range of sentence frequencies.

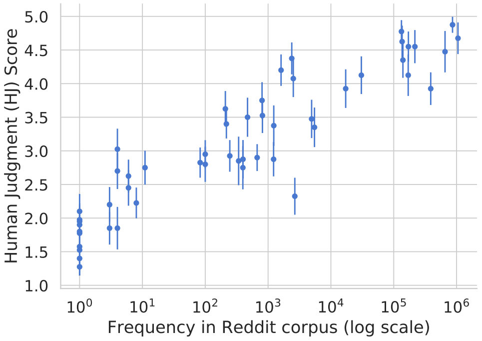


Figure 2: On the Reddit corpus, human judgment (HJ) of the "typicality" of a sentence y correlates strongly ($r = 0.92$) with its frequency in the corpus, suggesting that HJ is a good surrogate for $\log p_{\text{ref}}$. Error bars at the 90% confidence interval.

and define the (population) *HUSE score* as $L(\phi_{\text{huse}})$.

3.3 Guarantees derived from HUSE

We now show that the HUSE score satisfies two nice properties: (i) HUSE does at least as well as human evaluation and (ii) a low HUSE score is *sufficient* to show that a model is far from the reference distribution.

To show (i), consider a feature map that only includes human evaluation: $\phi_{\text{hj}}(x, y) \overset{\text{def}}{=} [\text{HJ}(x, y)]$. Because ϕ_{huse} also incorporates human evaluation, $L(\phi_{\text{huse}})$ is always tighter (lower) than the human discriminator error $L(\phi_{\text{hj}})$:

Proposition 1 (Relationship between HUSE, human evaluation, and optimal scores)**.**

$$L(\phi_{hj}) \geq L(\phi_{huse}) \geq L^*. \tag{6}$$

Furthermore, the main difference between $L(\phi_{\text{huse}})$ and L^* is that the former uses $\text{HJ}(x, y)$ and the latter uses p_{ref}. But as we argued using Figure 2, $\text{HJ}(x, y)$ is strongly correlated with p_{ref}, and good approximations to p_{ref} provide approximation guarantees for $L(\phi_{\text{huse}})$ (Appendix Theorem 1).

4 Evaluating models with HUSE

In this section, we show how we can estimate the error rate $L(\phi)$ from finite data (Section 4.1). We then show how the HUSE estimate $(\hat{L}(\phi_{\text{huse}}))$ can be decomposed into a score that measures quality (HUSE-Q) and a score that measures diversity (HUSE-D), which allows us to study quality-diversity tradeoffs (Section 4.2).

4.1 Learning a discriminator

For any feature map ϕ, we show how to produce an estimate of $L(\phi)$. Fix n contexts $x_1, \ldots, x_n$. First, we draw n examples $y_1, \ldots, y_n$ from the reference distribution $p_{\text{ref}}(y \mid x)$, which are usually human-generated sentences from a test set. We also draw n examples $y'_1, \ldots, y'_n$ from the model $p_{\text{model}}(y \mid x)$ we wish to evaluate. Next, for each of the $2n$ examples (x, y), we compute the feature map $\phi(x, y)$, which might involve evaluating the model probability $p_{\text{model}}(y \mid x)$ as well as collecting human judgments $\text{HJ}(x, y)$ from crowdworkers.

Finally, we compute the leave-one-out error of a classifier that tries to predict whether a given example (x, y) comes from the reference distribution $(z = 1)$ or the model $(z = 0)$.

The classification problems for HUSE are two-dimensional, which allows us to accurately estimate error rates using a k-nearest neighbors classifier. We opt to use nearest neighbors classifiers as they are simple, require no training, and can asymptotically capture arbitrary continuous decision boundaries. Specifically, we set $k = 16$ and define neighbors using L_2 distances over the feature vectors $\phi(x, y)$ scaled componentwise to have unit variance. The overall procedure for computing the estimate $\hat{L}(\phi)$ is formally defined in Algorithm 1.

Algorithm 1 Estimating error rates under ϕ

```
Require: Feature map $\phi$, number of neighbors $k$
    Contexts $x_1, \ldots, x_n$
    Reference outputs $y_1, \ldots, y_n$
    Model outputs $y'_1, \ldots, y'_n$
Construct dataset:
    $\mathcal{D} = \bigcup_{i=1}^{n} \{(\phi(x_i, y_i), 1), (\phi(x_i, y'_i), 0)\}$
$\hat{L}(\phi) \overset{\text{def}}{=}$ leave-one-out error of $k$-NN on $\mathcal{D}$
```

4.2 Quality-diversity decomposition

We now define the (empirical) *HUSE score* using the feature map ϕ_{huse}:

$$\text{HUSE} \overset{\text{def}}{=} \hat{L}(\phi_{\text{huse}}). \tag{7}$$

We define the quality component of HUSE (HUSE-Q) similarly using human judgments alone:

$$\text{HUSE-Q} \overset{\text{def}}{=} \hat{L}(\phi_{\text{hj}}). \quad (8)$$

Since humans can detect quality defects in a model, any increase in error from removing p_{model} must come from a model's lack of diversity. Therefore, we define the diversity component (HUSE-D) as follows:

$$\text{HUSE-D} \overset{\text{def}}{=} 1 + \text{HUSE} - \text{HUSE-Q}, \quad (9)$$

which implies the decomposition $(1-\text{HUSE-D})+(1-\text{HUSE-Q}) = 1-\text{HUSE}$. As long as the discriminators are non-degenerate (obtaining better performance than chance and HUSE $>$ HUSE-Q), all scores are contained in $[0, 1]$. Here, HUSE-D $= 1$ implies that the model suffers no diversity defects, while HUSE-D $= 0$ indicates that the examples could be discriminated perfectly due to a lack of diversity.

5 Experiments

5.1 Experimental setup

We use HUSE to evaluate three different types of single-sentence natural language generation tasks: (i) unconditional and high entropy (language modeling); (ii) conditional and high entropy (story generation, chit-chat dialogue); and (iii) conditional and low entropy (summarization). We show that HUSE provides a direct and interpretable measure of diversity on high-entropy tasks, while also serving as a useful model diagnostic on low-entropy ones.

The four tasks along with the datasets and models are as follows:

- **Summarization**: Giganews story to headline dataset and the pre-trained model from Gehrmann et al. (2018). The dataset consists of 3.8 million news story-headline pairs. Examples from this dataset are shown in Table 2.

- **Story generation**: Last sentence generation for ROC stories (Mostafazadeh et al., 2016) consisting of 96,198 examples of partially written four-sentence stories as input, and a single sentence which completes the story as the target. We use a standard OpenNMT model with global attention (Klein et al., 2017).

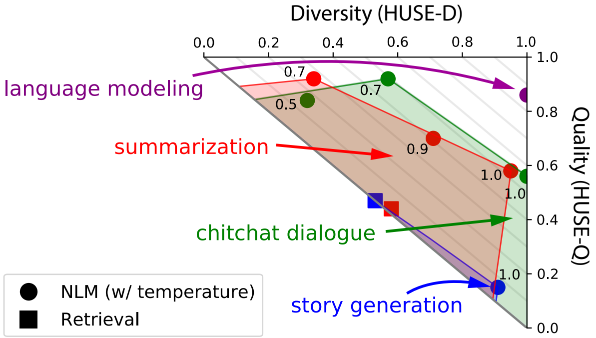


Figure 3: Tradeoffs between HUSE-D and HUSE-Q. Points are models and color indicates task. Neural models (circle) generate using temperature annealing (point labels indicate temperature). Models closer to the top right are superior, and gray diagonal lines indicate equivalent HUSE. A shaded region for a task indicates models which are strictly dominated (worse HUSE with the same HUSE-D-HUSE-Q proportion). Annealing can trade-off between diversity and quality but cannot easily increase the underlying model performance (HUSE).

- **Language modeling**: One billion word benchmark pre-trained language model from Jozefowicz et al. (2016). The task consists of generating a single sentence from the one billion word newswire text distribution.

- **Chit-chat dialogue**: Two-turn chit-chat dialogue dataset consisting of 37.3 million comment-response pairs from Reddit (Appendix A.4). Comments are generally short (5–15 tokens) and cover a single topic (e.g. given "wow how did i not notice that", the response is "you were focusing on other things its understandable"). We train a convolutional model using fairseq (Gehring et al., 2017).

For all the tasks, we train neural models and evaluate their diversity-quality tradeoffs as we change the decoding scheme for generation. Our primary evaluation concerns diversity trade-offs involving **temperature annealing** which is a generation technique applicable to any probabilistic model that generates words sequentially. In temperature annealed models, we sample a word w proportional to $p^{1/t}(w)$ where p is the model probability of w given previous words and t is the temperature parameter. We excluded beam search since it qualitatively behaves similarly to temperature annealing with low temperatures and HUSE ≈ 0 due to beam search being extremely

Score	Summarization $t = 1.0$	Summarization $t = 0.7$	Story generation $t = 1.0$	Story generation Retrieval	Chit-chat dialogue $t = 1.0$	Chit-chat dialogue $t = 0.7$	LM $t = 1.0$
HUSE	**0.53**	0.26	**0.06**	0.00	**0.56**	0.49	**0.86**
HUSE-Q	0.58	**0.92**	0.15	**0.47**	0.56	**0.92**	**0.88**
HUSE-D	**0.95**	0.34	**0.91**	0.53	**1.00**	0.57	**1.02**

Table 1: Performance achieved by the best models on the four tasks, as measured by overall goodness-of-fit (HUSE), sample quality (HUSE-Q) and diversity (HUSE-D). The scale for HUSE and HUSE-Q ranges from 0.0 (completely distinguishable from reference) to 1.0 (indistinguishable from reference) where the implied classification error is HUSE$/2$. HUSE-D may exceed 1.0 with small sample sizes when HUSE-Q $>$ HUSE.

under diverse.

As a non-neural baseline, we also consider retrieval based models based on Apache `solr` on a few tasks. For this approach, we retrieve the single most relevant response from the training set using the BM25 similarity metric on inputs. Such models are known to perform well in tasks with complex outputs such as program generation (Hayati et al., 2018; Hashimoto et al., 2018) and style transfer (Li et al., 2018).

For cost reasons, we did not measure certain combinations of task and generation mechanisms. We did not measure retrieval for chit-chat dialogue, as we observed its outputs were lower quality than a low-temperature neural model. We also did not anneal language models, as the generation quality from the language model was already high, and our goal was to show that they achieved high HUSE. Our set of measurements, while not comprehensive, generally covers the available quality-diversity tradeoffs for conditional tasks.

Finally, we collect human judgments HJ(x, y) as per Section 4.1 where we query 20 Amazon Mechanical Turk crowdworkers for typicality ratings on 100 reference and 100 model sentences. Since our models generate UNK (unknown and out-of-vocabulary) tokens, we instructed crowdworkers to treat UNK tokens as rare, but appropriate words for the context.

5.2 Overall results

The HUSE scores across the four tasks vary widely. Table 1 shows that single-sentence language models are nearly indistinguishable, with HUSE $= 0.86$ and implied discriminator error of 43%.

In contrast, both summarization and dialogue are highly distinguishable (HUSE ≈ 0.5) with relatively low quality when sampled from $t = 1.0$. Human evaluation alone (HUSE-Q) would suggest that using temperature annealing $(t = 0.7)$ to emphasize high-probability outputs substantially improves the model (HUSE-Q goes from 0.58 to 0.92 for summarization and 0.56 to 0.92 for dialogue). However, we find that this increase in sample quality comes at the cost of diversity (HUSE-D goes from 0.95 to 0.34 for summarization and 1.0 to 0.57 for dialogue). Examining the achievable HUSE and diversity tradeoffs in Figure 3 shows that mechanisms such as annealing which improve sample quality actually degrade HUSE due to severe losses in diversity.

We find that all generation schemes and models are inadequate for story generation on ROC stories. The original model $(t = 1.0)$ is very easily distinguishable by a human (HUSE-Q $= 0.15$), corresponding to a discriminator error of 7%. The retrieval models can improve this to HUSE-Q $=$ 0.47, but this comes at the expense of diversity.

Finally, we observe that directly sampling from the model $(t = 1.0)$ is always diverse. This suggests that human evaluation is an appropriate evaluation for generation systems that are directly sampled (rather than beam-searched).

5.3 Model error analysis with HUSE

Since HUSE is estimated from a two-dimensional classification problem, we can directly visualize the classification problem to understand defects in both model quality and diversity.

Figure 4 shows both reference points $\phi_{\text{huse}}(x_i, y_i)$ (blue squares) and model points $\phi_{\text{huse}}(x_i, y'_i)$ (red circles) for the summarization task. The shaded areas indicate the decision boundary of the 16-nearest neighbor classifier.

At temperature $t = 1.0$, we find that the classification boundary is mostly horizontal, implying that human judgment alone can distinguish model outputs from references. There is a cluster of sentences with high HJ and high p_{model} which are es-

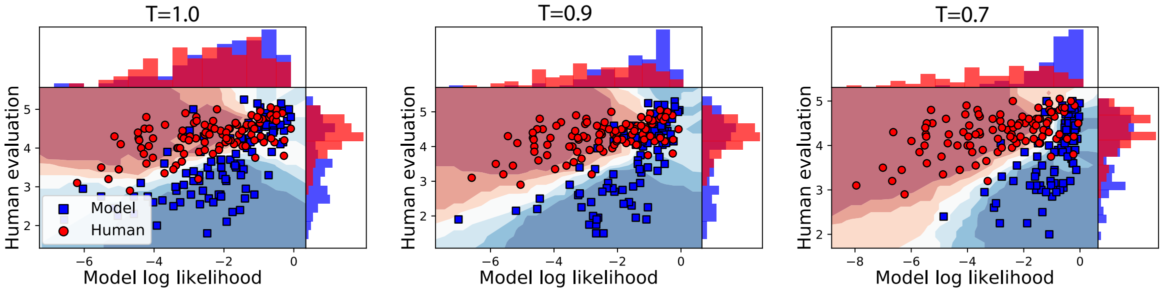


Figure 4: The two-dimensional classification problem in Algorithm 1 on the summarization task with different softmax temperatures (three panels). Each point represents a reference sentence $\phi_{\text{huse}}(x_i, y_i)$ or model-generated sentence $\phi_{\text{huse}}(x_i, y'_i)$. The color denotes the source of the sentence (z), shading is the classification confidence of the nearest neighbor classifier.

sentially indistinguishable. Examining the samples in this top-right region reveals that these are news stories with short headlines such as "Nadal pulls out of Sydney International" which can be reliably generated even at $t = 1.0$. However, the model frequently generates low quality samples that can easily be distinguished such as "two new vaccines in the poor countries were effective against go-it-alone study says" (Table 2).

At lower temperatures of $t = 0.9$ and $t = 0.7$, the boundary shifts towards becoming diagonal. Although the distribution is no longer directly separable on human judgment, the two distributions are clearly separable with the inclusion of p_{model}.

Using Figure 4, we can identify individual examples which were correctly and incorrectly classified based on p_{model} and HJ. Table 2 shows examples of both quality failures and diversity failures identified by HUSE. For example, the "diversity failure" table shows that the summarization model ($t = 0.7$) has an extremely low probability of generating some reference sentences ("NFL's bills shake up front office") and is thus under-diverse. Closer examination of the model shows that the probability of generating "front office" is low, since it is an unusual way to refer to the president and general manager. Improving these models on the diversity failures will require that the model understand more subtle paraphrases. We can also identify model successes, where the model outputs are indistinguishable from the reference in terms of quality ("Agassi bows out of Australian Open after injury"), and the model assigns high probability to the reference ("Agassi withdraws from Australian Open").

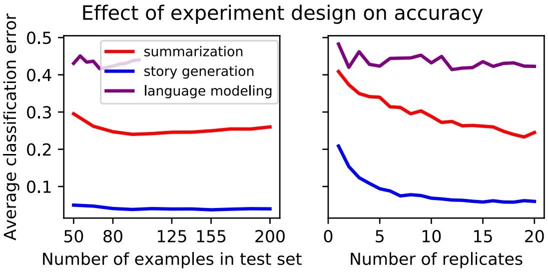


Figure 5: Estimates of HUSE are robust to small test set size, but generally require ≈ 20 crowdworker measurements for each example.

5.4 HUSE stability

Since HUSE depends on human crowdworker annotations, one might ask if it is possible to reduce either the number of annotated examples, or number of distinct crowdworkers for each example. We show that for low-quality models, substantially fewer annotations are needed.

Figure 5 shows the result of subsampling our original data of 200 sentences and 20 crowdworkers and estimating HUSE. First, we find that using 50 test set examples (Figure 5, left) is often sufficient to give accurate estimates of HUSE. Next, we find that the necessary number of crowdworkers per example depends heavily on the task. Easily distinguishable tasks (story generation), require only 10 crowdworkers, while less distinguishable tasks (summarization) require more than 20 crowdworkers to obtain accurate estimates.

6 Related work

The current state of NLG evaluation. Existing approaches to NLG evaluation use a hodgepodge mix of quality and diversity measures. Out of the 26 NLG papers at ACL 2018, six perform only hu-

	Quality failure	$\log p_{\text{model}}$	HJ
Context:	Two new vaccines have been shown effective against rotavirus, which is responsible for a half-million infant deaths in poor countries each year, research studies published Wednesday said.		
Model	Two new vaccines in the poor countries were effective against go-it-alone study says	-2.3	2.6
Reference	New vaccines for key <UNK> virus shown effective	-4.0	4.3
	Diversity failure		
Context:	The Buffalo Bills sacked Tom Donahoe as president and general manager on Wednesday, fulfilling expectations of a shake-up after another failure to make the National Football League playoffs.		
Model	Bills sack <UNK> as president GM and general manager	-0.9	4.3
Reference	NFL's Bills shake up front office.	-5.1	4.3
	Model is indistinguishable		
Context:	US veteran and eight-time Grand Slam winner Andre Agassi has withdrawn from this month's Australian Open due to a nagging ankle injury, his management team announced Thursday.		
Model	Agassi bows out of Australian Open after injury.	-1.4	5.3
Reference	Agassi withdraws from Australian Open.	-0.3	4.9

Table 2: Example reference and model outputs (capitalization added for readability) corresponding to Figure 4 (summarization task) that were shown to crowdworkers (left column). Crowdworkers were shown samples from the model (including the <UNK> token) and returned human judgments (right column). Using human judgments and the model probability, we can identify several types of failures. Quality failures are examples that are classified by human judgment. Diversity failures are examples that are classified by model probabilities. Finally some examples are not easily classified, as they have similar human judgment and model probability scores.

man evaluation, fourteen measure human evaluation and a diversity metric such as perplexity or n-gram diversity, and six do not evaluate using human judgments.

While perplexity and n-gram counts can *in principle* evaluate diversity, their practical implementations suffer from serious drawbacks. When human evaluation and perplexity are both evaluated, they are almost always done on separate models—human evaluations are done on beam-searched output, while perplexity is computed on the softmax outputs. This makes it appear as if the models can simultaneously generate high quality outputs while also being diverse, when in fact they can only be one at a time based on whether they sample or run beam search.

On the other hand, n-gram diversity was proposed by Li et al. (2016) to identify models with the generic utterance problem where models repeat phrases such as 'I don't know'. Unfortunately, n-gram diversity is computed *across* contexts by counting the number of unique n-grams generated, and so does not measure a model's ability to generate multiple valid utterances at any single context. In particular, a model which only outputs a single memorized utterance per context (e.g., via memorization or retrieval) can still have high n-gram diversity as long as the memorized sentences differ across contexts.

Finally, *all* existing diversity measures are computed separately from human evaluation. This results in two incomparable evaluation metrics, which prevent us from reasoning about tradeoffs between diversity and quality. In contrast, HUSE allows us to make precise statements about the tradeoffs between model quality and diversity because it is a single metric which decomposes into diversity and quality terms.

Related evaluations of diversity. The importance of diverse responses has previously been acknowledged for summarization (Nenkova et al., 2007) and information retrieval (Clarke et al., 2008). Our work differs in considering a *single* evaluation measure that captures quality and diversity applicable to *any* generation task.

Automated metrics based on n-gram overlap such as BLEU, METEOR, ROUGE (Papineni et al., 2002; Lavie and Denkowski, 2009; Lin and Rey, 2004) work well for machine translation but do not generalize well to domains with a diverse spectrum of correct responses. While variants (Sun and Zhou, 2012; Galley et al., 2015; Shima and Mitamura, 2011) have adapted such metrics to high entropy generative environments, they are still significantly inferior to the human judgments they attempt to mimic.

Caccia et al. (2018) recently examined the diversity and quality tradeoffs for different language model architectures on synthetic datasets. However, as their approach relies on measuring log-likelihoods under both the model and reference distributions, it *cannot* be applied to real data where p_{ref} is unavailable. Our main conceptual contribution overcomes this by showing that HJ is an acceptable proxy for p_{ref}.

Sajjadi et al. (2018) also examines diversity and quality (which they call precision and recall) in the context of generative image models. However, they rely on assuming that p_{ref} and p_{model} can be estimated accurately using the Fréchet Inception Distance (FID) (Heusel et al., 2017). HUSE avoids such assumptions and instead directly leverages human judgments, resulting in a simple and reliable metric more suitable for use as a gold-standard.

Estimating optimal classification error. Evaluating a model by estimating its optimal classification error has been considered by several earlier works (Olsson et al., 2018; Kannan and Vinyals, 2016; Li et al., 2017; Bruni and Fernandez, 2017; Bowman et al., 2016). However, these methods have focused on classifying sentences directly, which is quite challenging to do reliably. Existing adversarial evaluation methods do not yet reliably outperform human classification (Kannan and Vinyals, 2016; Bruni and Fernandez, 2017). We propose the use of both human evaluation and model probabilities as part of the adversarial evaluation framework, and demonstrate that the resulting classifier reliably outperforms humans and captures both the sample quality and diversity of a model.

Distributional divergence estimation. Our proposed evaluation metric is closely related to the total variation distance which has been studied extensively in the distribution testing literature. It is known that total variation distance estimates have pessimistic minimax estimation rates in high dimensions (Balakrishnan and Wasserman, 2017). Our work overcomes this by utilizing p_{model} and an estimate of p_{ref}. Other approaches to distributional testing include the maximum mean discrepancy (MMD) and Wasserstein distances, but these approaches require knowledge of a ground truth metric or kernel space (Tolstikhin et al., 2016; Singh et al., 2018). Although such divergences are easier to estimate than the total variation distance from samples, the implied convergence rates are still too slow to be practically useful.

7 Discussion

In this paper, we demonstrate that the current gold standard of human evaluation does not penalize under-diverse models. To remedy this, we propose HUSE, a general purpose evaluation strategy which can be applied to any model for which we can calculate a model's sampling probabilities. HUSE is an upper bound on the optimal classification error of distinguishing reference and model-generated text, and *never* does worse than human classification. HUSE leverages both model probabilities and human judgments, ensuring that models which do well on the metric are both high-quality and diverse.

Our work can be viewed as a "superhuman version" of the classic Turing Test (Turing, 1950). Instead of relying on just a human classifier, we approximate the *optimal* classifier, which can utilize information about the model in addition to the reference. We also modify the classification problem and seek to identify whether a sample comes from a (potentially superhuman) reference distribution, rather than the human distribution. These two changes lead to tractable, rigorous estimators which can quantify tradeoffs between model quality and diversity on a wide range of generation tasks.

Acknowledgements. We would like to thank Arun Chaganty, Robin Jia, and Peng Qi for extensive comments and feedback on the paper. This work was funded by DARPA CwC program under ARO prime contract no. W911NF-15-1-0462.

Reproducibility. All code, data, and experiments are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x88644b5ee189402eb19d39d721d1005c`.

References

S. Balakrishnan and L. Wasserman. 2017. Hypothesis testing for high-dimensional multinomials: A selective review. *arXiv preprint arXiv:1712.06120*.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. 2016. Generating sentences from a continuous space. In *Computational Natural Language Learning (CoNLL)*, pages 10–21.

E. Bruni and R. Fernandez. 2017. Adversarial evaluation for open-domain dialogue generation. In *Proceedings of the SIGDIAL 2017 Conference*.

M. Caccia, L. Caccia, W. Fedus, H. Larochelle, J. Pineau, and L. Charlin. 2018. Language gans falling short. *arXiv preprint arXiv:1811.02549*.

A. Chaganty, S. Mussmann, and P. Liang. 2018. The price of debiasing automatic metrics in natural language evaluation. In *Association for Computational Linguistics (ACL)*.

C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *ACM SIGIR*.

M. Feder and N. Merhav. 1994. Relations between entropy and error probability. *IEEE Transactions on Information Theory*, 40:259–266.

M. Galley, C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863*.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

S. Gehrmann, Y. Deng, and A. M. Rush. 2018. Bottom-up abstractive summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*.

T. Hashimoto, K. Guu, Y. Oren, and P. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

S. A. Hayati, R. Olivier, P. Avvaru, P. Yin, A. Tomasic, and G. Neubig. 2018. Retrieval-based neural code generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

A. Kannan and O. Vinyals. 2016. Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

A. Lavie and M. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23.

J. Li, M. Galley, C. Brockett, J. Gao, and W. B. Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 110–119.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *North American Association for Computational Linguistics (NAACL)*.

J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

C. Lin and M. Rey. 2004. Looking for a few good metrics: ROUGE and its evaluation. In *NTCIR Workshop*.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll'ar, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755.

C. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Association for Computational Linguistics (ACL)*.

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *North*

American Association for Computational Linguistics (NAACL).

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023.*

A. Nenkova, R. J. Passonneau, and K. McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. In *ACM Transactions on Speech and Language Processing.*

J. Novikova, O. Dušek, A. C. Curry, and V. Rieser. 2017. Why we need new evaluation metrics for NLG. In *Empirical Methods in Natural Language Processing (EMNLP).*

C. Olsson, S. Bhupatiraju, T. Brown, A. Odena, and I. Goodfellow. 2018. Skill rating for generative models. *arXiv preprint arXiv:1808.04888.*

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL).*

M. Roemmele. 2016. Writing stories with help from recurrent neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI).*

M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. 2018. Assessing generative models via precision and recall. *arXiv preprint arXiv:1806.00035.*

H. Shima and T. Mitamura. 2011. Diversity-aware evaluation for paraphrase patterns. In *Empirical Methods in Natural Language Processing (EMNLP).*

S. Singh, A. Uppal, B. Li, C. Li, M. Zaheer, and B. Poczos. 2018. Nonparametric density estimation under adversarial losses. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 246–257.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *North American Association for Computational Linguistics (NAACL).*

H. Sun and M. Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *Association for Computational Linguistics (ACL).*

J. Surowiecki. 2004. *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations.* Doubleday and Co.

L. Theis, A. van den Oord, and M. Bethge. 2015. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844.*

I. Tolstikhin, B. K. Sriperumbudur, and B. Scholkopf. 2016. Minimax estimation of maximum mean discrepancy with radial kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1930–1938.

A. M. Turing. 1950. Computing machinery and intelligence. *Mind*, 49:433–460.

L. Ungar, B. Mellors, V. Satopää, J. Baron, P. Tetlock, J. Ramos, and S. Swift. 2012. The good judgment project: A large scale test of different methods of combining expert predictions. In *Association for the Advancement of Artificial Intelligence (AAAI).*

A Appendix

A.1 Relationship between total variation distance and optimal discriminator error

This is a standard result, replicated here for completeness:

Proposition 2. *The total variation distance is related to the optimal discriminator error as follows:* $\|p_{model} - p_{ref}\|_{TV} = 1 - L^*$.

Proof Fix any x. Define $a_y \overset{\text{def}}{=} p_{\text{ref}}(y \mid x)$ and $b_y \overset{\text{def}}{=} p_{\text{model}}(y \mid x)$. Let $S \overset{\text{def}}{=} \{y : a_y < b_y\}$ be the y where the p_{model} assigns higher probability than p_{ref}, and define $A \overset{\text{def}}{=} \sum_{y \in S} a_y$ and $B \overset{\text{def}}{=} \sum_{y \in S} b_y$ be the aggregated probabilities. On S, the optimal discriminator should return $z = 0$ (model). This is an error when $z = 1$, which occurs with probability $\frac{1}{2}A$. Analogously, on the complement of S, the error probability (when $z = 0$) is $\frac{1}{2}(1 - B)$. The total contribution to L^* is thus $A + (1 - B)$. The rest follows from algebra:

$$\|p_{\text{model}} - p_{\text{ref}}\|_{\text{TV}} = \frac{1}{2}\|p_{\text{model}} - p_{\text{ref}}\|_1 \quad (10)$$

$$= \frac{1}{2}[(B - A) + (1 - A) - (1 - B)] \quad (11)$$

$$= B - A = (1 - L^*). \quad (12)$$

□

A.2 Approximation error from ϕ features

Theorem 1. *Let* L^* *and* $L(\phi)$ *be the optimal classification error and optimal error under feature map* ϕ *respectively. Then,*

$$L^* \leq L(\phi) \leq L^* + 2(1 - 2^{-I})$$

where $I \overset{\text{def}}{=} I(Z_{opt}; \phi_{opt}(X, Y) \mid \phi(X, Y))$ *is the conditional mutual information in bits and* Z_{opt} *is the prediction of the optimal classifier.*

Proof The lower bound falls out of the definition of L^*. To prove the upper bound, a variant of the entropy lower bound by Feder and Merhav (Feder and Merhav, 1994) shows that the error rate for predicting Z_{opt}, via the optimal $f(\phi(X, Y))$ follows

$$P(f(\phi(X, Y)) \neq Z_{opt}) \leq 1 - 2^{I(Z_{opt};\phi(X,Y)) - H(Z_{opt})}. \quad (13)$$

Now expand the mutual information using the chain rule

$$\begin{aligned} I(Z_{opt}; \phi(X, Y)) &= I(Z_{opt}; \phi_{opt}(X, Y), \phi(X, Y)) \\ &\quad - I(Z_{opt}; \phi_{opt}(X, Y) \mid \phi(X, Y)) \\ &= -I(Z_{opt}; \phi_{opt}(X, Y) \mid \phi(X, Y)) + H(Z_{opt}). \end{aligned}$$

The last line follows from the fact that Z_{opt} is a deterministic function of ϕ_{opt} (Proposition 1). Substituting this into the inequality gives the bound,

$$P(f(\phi(X, Y)) \neq Z_{opt}) \leq 1 - 2^{-I}$$

with $I = I(Z_{opt}; \phi_{opt}(X, Y) \mid \phi(X, Y))$.

Finally, note that Z_{opt} incurs $L^*/2$ error, and we disagree with Z_{opt} at most a $P(f(\phi(X, Y)) \neq Z_{opt})$ fraction of time. Assuming that we get every one of these disagreements wrong gives an upper bound of $L^*/2 + P(f(\phi(X, Y)) \neq Z_{opt})$ on $L(\phi)/2$. □

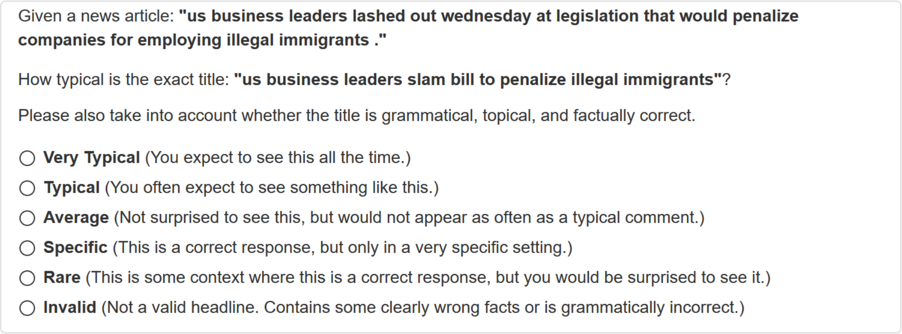
A straightforward corollary is that whenever ϕ is an invertible function of ϕ_{opt}, the conditional mutual information is zero, and therefore the above inequalities become an equality.

Corollary 1. *Whenever* ϕ *is an invertible function of* ϕ_{opt}, $L(\phi) = L^*$.

A.3 Amazon Mechanical Turk for human judgments

In order to show that HUSE can be reliably estimated even with simple crowdsourcing techniques, we used a single uniform task design where we asked Amazon Mechanical Turk workers to rate the typicality of a sentence from 0–5. We defined 0 as invalid (grammatically or factually incorrect) and 5 as 'very typical'. HJ(x, y) is defined as the average score that crowdworkers assign to a response y given the context x. We did not perform substantial filtering or qualification checks beyond HIT acceptance rate (HIT Approval rate greater than 95 percent and number of HITs approved greater than 50 and location is USA). We constructed each HIT to be 25 examples, and paid one dollar per HIT.

We observe that measuring many replicates is sufficient to get low-variance estimates of HJ. For classification tasks where the model is straightforward to identify from references (such as story generation) we require five to ten replicates, while



Given a news article: **"us business leaders lashed out wednesday at legislation that would penalize companies for employing illegal immigrants ."**

How typical is the exact title: **"us business leaders slam bill to penalize illegal immigrants"**?

Please also take into account whether the title is grammatical, topical, and factually correct.

○ **Very Typical** (You expect to see this all the time.)
○ **Typical** (You often expect to see something like this.)
○ **Average** (Not surprised to see this, but would not appear as often as a typical comment.)
○ **Specific** (This is a correct response, but only in a very specific setting.)
○ **Rare** (This is some context where this is a correct response, but you would be surprised to see it.)
○ **Invalid** (Not a valid headline. Contains some clearly wrong facts or is grammatically incorrect.)

Figure 6: Amazon Mechanical Turk survey design for eliciting human judgment scores HJ in the summarization task.

for hard tasks such as summarization at least twenty replicates are needed (Section 5.4). Manual inspection suggests that up to 20% of the collected data are low-quality but that this noise is uncorrelated with the sentence being rated and outweighed by a larger majority of honest and reasonably accurate data. Even if the data quality is low, HUSE is still a valid upper bound (i.e. models with low HUSE are guaranteed to be distinguishable from humans). Thus the models which we identify as having low-HUSE are reliably distinguishable regardless of the crowdworker quality.

A.4 Reddit Dataset

We use a subset of Reddit comments from 2006-2018 scraped from https://pushshift.io/. We construct a dictionary containing the 10,000 most popular words and preprocess the dataset by removing deleted posts, out-of-vocabulary tokens, profanity, comments with less than 10 upvotes, and comments with over 400 tokens.

What makes a good conversation?
How controllable attributes affect human judgments

Abigail See[*]
Stanford University
abisee@stanford.edu

Stephen Roller
Facebook AI Research
roller@fb.com

Douwe Kiela
Facebook AI Research
dkiela@fb.com

Jason Weston
Facebook AI Research
jase@fb.com

Abstract

A good conversation requires balance – between simplicity and detail; staying on topic and changing it; asking questions and answering them. Although dialogue agents are commonly evaluated via human judgments of overall quality, the relationship between quality and these individual factors is less well-studied. In this work, we examine two controllable neural text generation methods, conditional training and weighted decoding, in order to control four important attributes for chitchat dialogue: repetition, specificity, response-relatedness and question-asking. We conduct a large-scale human evaluation to measure the effect of these control parameters on multi-turn interactive conversations on the PersonaChat task. We provide a detailed analysis of their relationship to high-level aspects of conversation, and show that by controlling combinations of these variables our models obtain clear improvements in human quality judgments.

1 Introduction

Neural generation models for dialogue, despite their ubiquity in current research, are still poorly understood. Well known problems, such as the genericness and repetitiveness of responses (Serban et al., 2016a), remain without a de facto solution. Strikingly, the factors that determine human judgments of overall conversation quality are almost entirely unexplored. Most works have been limited to the next utterance prediction problem, whereas a multi-turn evaluation is necessary to evaluate the quality of a full conversation.

In this work we both (i) conduct a large-scale study to identify the fine-grained factors governing human judgments of full conversations, and (ii) develop models that apply our findings in practice, leading to state-of-the-art performance. Specifically, we identify and study eight aspects of conversation that can be measured by human judgments, while varying four types of low-level attributes that can be algorithmically controlled in neural models; see Figure 1. To control the low-level model attributes, we consider two simple but general algorithms: conditional training, in which the neural model is conditioned on additional control features, and weighted decoding, in which control features are added to the decoding scoring function at test time only.

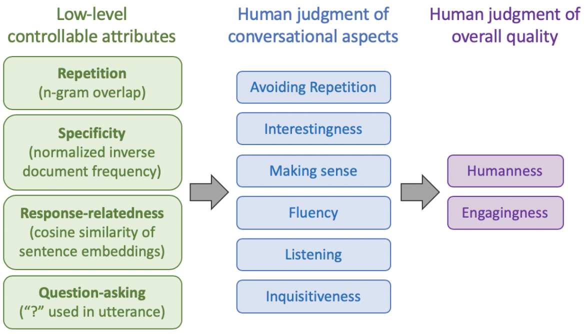


Figure 1: We manipulate four low-level attributes and measure their effect on human judgments of individual conversational aspects, as well as overall quality.

One major result of our findings is that existing work has ignored the importance of conversational flow, as standard models (i) repeat or contradict previous statements, (ii) fail to balance specificity with genericness, and (iii) fail to balance asking questions with other dialogue acts. Conducting experiments on the PersonaChat task (Zhang et al., 2018b), we obtain significantly higher engagingness scores than the baseline by optimizing control of repetition, specificity and question-asking over multiple turns. Using these findings, our best model matches the performance of the winning entry in the recent NeurIPS ConvAI2 competition (Dinan et al., 2019), which was trained on much

---

[*] A.S. completed most of this work at Facebook (FAIR).

Proceedings of NAACL-HLT 2019, pages 1702–1723
Minneapolis, Minnesota, June 2 - June 7, 2019.

more data but had no control (see Section 8.1). Our code, pretrained models, and full chatlogs, are available at `https://parl.ai/projects/controllable_dialogue`.

2 Related Work

Dialogue Dialogue evaluation is relatively well understood in goal-oriented tasks, where automated approaches can be coded by measuring task completion (Bordes et al., 2017; El Asri et al., 2017; Hastie, 2012; Henderson et al., 2014; Wen et al., 2017). Task success combined with dialogue cost can be linked to human judgments like user satisfaction via the PARADISE framework (Walker et al., 1997).

However in chitchat tasks, which we study in this work, automatic metrics and their relation to human ratings are less well-understood. While word-overlap metrics are effective for question-answering and machine translation, for dialogue they have little to no correlation with human judgments (Liu et al., 2016; Novikova et al., 2017) – this is due to the open-ended nature of dialogue. There are more recent attempts to find better automatic approaches, such as adversarial evaluation (Li et al., 2017b) and learning a scoring model (Lowe et al., 2017), but their value is still unclear.

Nevertheless, a number of studies only use automatic metrics, with no human study at all (Lowe et al., 2015; Parthasarathi and Pineau, 2018; Serban et al., 2016b). Other works do use human evaluations (Dinan et al., 2018; Li et al., 2016a,b; Venkatesh et al., 2017; Vinyals and Le, 2015; Zhang et al., 2018b), typically reporting just one type of judgment (either quality or appropriateness) via a Likert scale or pairwise comparison. Most of those works only consider single turn evaluations, often with a shortened dialogue history, rather than full multi-turn dialogue.

A more comprehensive evaluation strategy has been studied within the scope of the Alexa prize (Venkatesh et al., 2017; Guo et al., 2018) by combining multiple automatic metrics designed to capture various conversational aspects (engagement, coherence, domain coverage, conversational depth and topical diversity). Though these aspects have some similarity to the aspects studied here, we also focus on lower-level aspects (e.g. avoiding repetition, fluency), to understand how they correspond to both our controllable attributes, and to overall quality judgments.

Controllable neural text generation Researchers have proposed several approaches to control aspects of RNN-based natural language generation such as sentiment, length, speaker style and tense (Fan et al., 2018; Ficler and Goldberg, 2017; Ghazvininejad et al., 2017; Hu et al., 2017; Kikuchi et al., 2016; Peng et al., 2018; Wang et al., 2017). In particular, several works use control to tackle the same common sequence-to-sequence problems we address here (particularly genericness and unrelated output), in the context of single-turn response generation (Baheti et al., 2018; Li et al., 2016a, 2017a; Shen et al., 2017; Xing et al., 2017; Zhang et al., 2018a; Zhou et al., 2017). By contrast, we focus on developing controls for, and human evaluation of, *multi*-turn interactive dialogue – this includes a new method (described in Section 5) to control attributes at the *dialogue* level rather than the utterance level.

In this work, we require a control method that is both general-purpose (one technique to simultaneously control many attributes) and easily tunable (the control setting is adjustable after training). Given these constraints, we study two control methods: conditional training (variants of which have been described by Fan et al. (2018); Kikuchi et al. (2016); Peng et al. (2018)) and weighted decoding (described by Ghazvininejad et al. (2017) as a general technique, and by Baheti et al. (2018) to control response-relatedness). To our knowledge, this work is the first to systematically compare the effectiveness of two general-purpose control methods across several attributes.

3 The PersonaChat dataset

PersonaChat (Zhang et al., 2018b) is a chitchat dialogue task involving two participants (two humans or a human and a bot). Each participant is given a *persona* – a short collection of personal traits such as *I'm left handed* or *My favorite season is spring* – and are instructed to get to know each other by chatting naturally using their designated personas, for 6–8 turns. The training set contains 8939 conversations and 955 personas, collected via crowdworkers, plus 1000 conversations and 100 personas for validation, and a similar number in the hidden test set. The PersonaChat task was the subject of the NeurIPS 2018 ConvAI2 Challenge (Dinan et al., 2019), in which competitors were first evaluated with respect to automatic metrics (perplexity, hits@1 and F1 score), and then with respect to human judgment via the question *"How much did you enjoy talking to this user?"* on a scale of 1–4.

4 Baseline model

Our baseline model is a 2-layer LSTM sequence-to-sequence model with attention. On any dialogue turn, the input x to the encoder is the entire dialogue history (separated using unique speaker-identifying tokens), with the model's own persona prepended. Conditioned on this input sequence x, the decoder generates a response y. Except when stated otherwise, all our models decode using beam search with beam size 20.

We initialized the word embedding matrix with 300-dimensional GloVe embeddings (Pennington et al., 2014). Using the ParlAI framework (Miller et al., 2017), we pretrained the model on a dataset of 2.5 million Twitter message-response pairs,[1] then fine-tuned it on PersonaChat. On the PersonaChat validation set, the baseline model has a perplexity of 26.83 and F1 of 17.02, which would have placed us 4th out of 26 models in the ConvAI2 competition (Dinan et al., 2019). We attempt to improve over this baseline using control.

5 Controllable text generation methods

Suppose we have a sequence-to-sequence model which gives $P(y|x) = \Pi_t P(y_t|x, y_1, \dots, y_{t-1})$, the conditional probability of a response y (the model's next utterance) given input x (the context, which in our case includes the model's own persona and the dialogue history).

Contrary to most previous work, which controls *at the sentence level*, we wish to control attributes of the output y *at the dialogue level* – meaning that a single control setting is used for a whole dialogue. For example, to control question-asking, we provide a control setting at the beginning of each dialogue (e.g. *20% questions* or *70% questions*) rather than providing a control setting for each utterance (e.g. *is a question* or *isn't a question*). With this approach, the sequence-to-sequence model is able to choose what value the controlled attribute should take for any particular utterance, but we are able to choose the overall distribution. We find that this approach works well – for example, the sequence-to-sequence model is generally good at detecting when to ask a question. In particular, this is easier than the alternative: developing a separate process to decide, for each utterance, whether to ask a question.

In this section, we describe the two methods – which we call Conditional Training (CT) and Weighted Decoding (WD) – that we use to control attributes of the output y at the dialogue level.

5.1 Conditional Training (CT)

Conditional Training (Fan et al., 2018; Kikuchi et al., 2016; Peng et al., 2018) is a method to learn a sequence-to-sequence model $P(y|x, z)$, where z is a discrete *control variable*. If the control attribute is naturally continuous (for example in our work, repetitiveness, specificity and response-relatedness), we use z to represent bucketed ranges. For a binary attribute like question-asking, z represents an overall probability (as explained in Section 5).

To train a CT model, we first automatically annotate every (x, y) pair in the training set with the attribute we wish to control (for example, whether y contains a question mark). During training, for each example we determine the corresponding z value (for continuous attributes, this simply means sorting into the correct bucket; for question-asking, see Section 6.4). Next, the control variable z is represented via an embedding (each of the possible values of z has its own embedding). For all our experiments, the embedding is of length 10; this was determined via hyperparameter tuning. There are several possible ways to condition the sequence-to-sequence model on z – for example, append z to the end of the input sequence, or use z as the START symbol for the decoder. We find it most effective to concatenate z to the decoder's input on every step.[2] Lastly, the CT model learns to produce $y = y_1, \dots, y_T$ by optimizing the cross-entropy loss:

$$\text{loss}_{\text{CT}} = -\frac{1}{T}\sum_{t=1}^{T}\log P(y_t|x, z, y_1, \dots, y_{t-1})$$

Our CT models are initialized with the parameters from the baseline sequence-to-sequence model $P(y|x)$ (the new decoder parameters are initialized with small random values), then fine-tuned to optimize loss_{CT} on the PersonaChat training set, until convergence of loss_{CT} on the validation set.

---

[1] The Twitter dataset is provided in ParlAI; details can be found here: `https://parl.ai/docs/tasks.html`

[2] To build a CT model $P(y|x, z_1, \dots, z_n)$ conditioned on *multiple* controls $\{z_1, \dots, z_n\}$, we can simply concatenate multiple control embeddings to the decoder inputs.

5.2 Weighted Decoding (WD)

Weighted Decoding (Ghazvininejad et al., 2017) is a decoding method that increases or decreases the probability of words with certain features. The technique is applied only at test time, requiring no change to the training method. A limitation of WD is that the controllable attribute must be defined at the word-level; any desired utterance-level attribute must be redefined via word-level features.

In weighted decoding, on the t^{th} step of decoding, a partial hypothesis $y_{<t} = y_1, \dots, y_{t-1}$ is expanded by computing the score for each possible next word w in the vocabulary:

$$\begin{aligned}\text{score}(w, y_{<t}; x) &= \text{score}(y_{<t}; x) \\ + \log P_{\text{RNN}}(w|y_{<t}, x) &+ \sum_i w_i * f_i(w; y_{<t}, x).\end{aligned}$$

Here, $\log P_{\text{RNN}}(w|y_{<t}, x)$ is the log-probability of the word w calculated by the RNN, $\text{score}(y_{<t}; x)$ is the accumulated score of the already-generated words in the hypothesis $y_{<t}$, and $f_i(w; y_{<t}, x)$ are *decoding features* with associated weights w_i. There can be multiple features f_i (to control multiple attributes), and the weights w_i are hyperparameters to be chosen.

A decoding feature $f_i(w; y_{<t}, x)$ assigns a real value to the word w, in the context of the text generated so far $y_{<t}$ and the context x. The feature can be continuous (e.g. the unigram probability of w), discrete (e.g. the length of w in characters), or binary (e.g. whether w starts with the same letter as the last word in $y_{<t}$). A positive weight w_i increases the probability of words w that score highly with respect to f_i; a negative weight decreases their probability.

Note that weighted decoding and conditional training can be applied simultaneously (i.e. train a CT model then apply WD at test time) – a strategy we use in our experiments.

6 Controlling conversational attributes

In this section, we describe how we use conditional training and weighted decoding to control four attributes: repetition, specificity, response-relatedness and question-asking. We evaluate the effectiveness of both control methods via automatic metrics (i.e., measuring how well the attribute was controlled), and use our findings to select control methods and control settings to be explored further via human evaluation (Section 8).

6.1 Repetition

Our baseline model exhibits three types of repetition, which we call *external repetition* (self-repetition across utterances), *internal repetition* (self-repetition within utterances), and *partner repetition* (repeating the conversational partner).

To control repetition with weighted decoding,[3] we define five n-gram based decoding features (see Appendix D). Three of these features (`extrep_bigram`, `intrep_bigram` and `partnerrep_bigram`) identify repeating bigrams for the three repetition types. The other two features (`extrep_unigram` and `intrep_unigram`) identify repeating content words. By applying a negative weight to these features, we can reduce repetition. In particular, if the weight is $-\infty$, our method is equivalent to *n-gram blocking* as described by Kulikov et al. (2018). We observe that repetition control is very important, thus all further control experiments include repetition control.

6.2 Specificity

Like many sequence-to-sequence models using beam search decoding, our baseline frequently asks generic questions such as *What music do you like?* and gives dull, unspecific responses, such as *I like all kinds of music*.

We control specificity using Normalized Inverse Document Frequency (NIDF) as a measure of word rareness.[4] The Inverse Document Frequency of a word w is $\text{IDF}(w) = \log(R/c_w)$ where R is the number of responses in the dataset, and c_w is the number of those responses that contain w. Normalized IDF (which ranges from 0 to 1) is

$$\text{NIDF}(w) = \frac{\text{IDF}(w) - \texttt{min\_idf}}{\texttt{max\_idf} - \texttt{min\_idf}} \quad (1)$$

where `min_idf` and `max_idf` are the minimum and maximum IDFs, taken over all words in the vocabulary. To control specificity with weighted decoding, we use NIDF as a decoding feature. As shown in Table 1, this method produces reasonable outputs when the feature weight is within a certain range, but at the extremes the outputs are

---

[3] We also tried controlling repetition with conditional training, defining z as the (bucketed) maximum ROUGE-L precision between the response y and the bot's previous utterances. However, this method was unsuccessful because there are not enough repetitive examples in the training data for the model to learn the control. Experimenting with data augmentation to solve this problem is an area for future work.

[4] Note that our NIDF specificity features are similar to the NIRF and NIWF features used by Zhang et al. (2018a).

Input: *Yes, I'm studying law at the moment*
Baseline Response: *That sounds like a lot of fun!*

Wt	**NIDF**	**Weighted Decoding Response**
-5.0	0.6%	*Oh......................................*
0.0	17.1%	*That sounds like a lot of fun!*
3.0	18.3%	*That sounds like a lot of fun. How long have you been studying?*
7.0	38.5%	*I majored in practising my spiritual full time philosophy test*
10.0	71.9%	*Oh wow! Merna jean isa paino yi hao hui bu acara sya gila* [. . .]

z	**NIDF**	**Conditional Training Response**
0	16.8%	*Sounds like you are a great person!*
2	18.3%	*So you are a law student?*
4	18.4%	*That sounds like a lot of fun*
6	22.8%	*That sounds like a rewarding job!*
8	24.4%	*That sounds like a rewarding career!*

Table 1: Middle: Example of controlling specificity (NIDF) via weighted decoding. At the extremes, the model produces only the most rare or the most common tokens. Bottom: Example of controlling specificity via conditional training. This gives a narrower NIDF range, but all the responses are appropriate.

Input: *Do you go get coffee often*
Baseline Response: *I do, when I am not playing the piano.*

Wt	**Sim**	**Weighted Decoding Response**
-10.0	-0.05	*I am a musician.*
0.0	-0.02	*I do, when I am not playing the piano.*
5.0	0.40	*I do, usually at starbucks.*
8.0	0.59	*Not usually, especially when you drink latte.*
11.0	0.72	*Not often, usually with drinks, espresso, latte, tea, etc.*

Table 2: Example of controlling response-relatedness (cosine similarity to input) via weighted decoding. Positive weights (e.g. 5.0) can yield more on-topic responses, but higher weights (e.g. 11.0) can result in nonsensical lists of topically related words.

nonsensical. The boundary for nonsensical output differs from example to example.

To control specificity with conditional training, we define the specificity of an utterance y to be the mean NIDF of the words in y. Thus our control variable z is mean NIDF (discretized into 10 equal-sized buckets). As shown in Table 1, this method gives outputs with a narrower NIDF range, but overall produces less nonsensical outputs.

6.3 Response-relatedness

In conversation, it's generally desirable to produce a response that is related to the partner's last utterance; for example if the partner says *My grandfather died last month*, it is appropriate to say *I'm so sorry. Were you close to your grandfather?* However, our baseline model frequently responds with unrelated utterances like *Do you have any pets?*

To control response-relatedness with weighted decoding, we use the decoding feature $\texttt{resp\_rel}$:

$$\texttt{resp\_rel}(w; y_{<t}, x) = \texttt{cos\_sim}(\texttt{word\_emb}(w), \texttt{sent\_emb}(\ell))$$

where $\texttt{word\_emb}(w)$ is the GloVe embedding for the word w, $\texttt{sent\_emb}(\ell)$ is the sentence embedding for the partner's last utterance ℓ (note ℓ is part of the context x), and $\texttt{cos\_sim}$ is the cosine similarity between the two. In particular, the sentence embedding $\texttt{sent\_emb}(s)$ for an utterance s is a weighted average of the GloVe embeddings of the words in s, with the first principal component projected out; for full details, see Arora et al. (2017). This method of controlling response-relatedness is similar to that described in (Baheti et al., 2018). We find that weighted decoding is effective to control the semantic relatedness of the model's response to the partner's last utterance (see Table 2). As before, we find that extreme weights lead to nonsensical output.

To control response-relatedness with conditional training, we try defining the control variable z to be $\texttt{cos\_sim}(\texttt{sent\_emb}(y), \texttt{sent\_emb}(\ell))$, the overall cosine similarity between the partner's last utterance ℓ and the model's response y (again, we discretize z). However, we find this method ineffective – the CT model learns only a very weak connection between z and the semantic relatedness of the output (see Section 7 for more details).

6.4 Question-asking

Considerate chitchat requires a reciprocal asking and answering of questions – asking too few or too many can appear self-centered or nosy. We control question-asking in order to study these trade-offs.

To control question-asking with weighted decoding, we use the binary decoding feature $\texttt{is\_qn\_word}(w)$, which is equal to 1 if and only if the word w is in a pre-defined list of interrogative words (*how, what, when, where, which, who, whom, whose, why, ?*). We find this is a somewhat effective method to encourage or discourage questions, but with unintended side-effects: a negative weight can discourage valid non-question utterances that happen to contain interrogative words (such as *I'm learning **how** to knit*) and a positive weight can result in degenerate utterances (such as

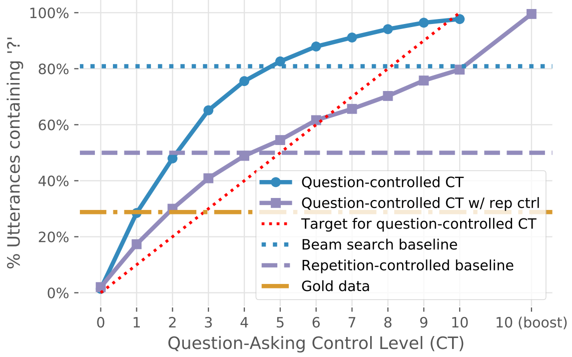


Figure 2: Controlling question-asking via conditional training. Exact numbers can be found in Appendix F.

What??????? or *Who? When? How?*).

For conditional training, we regard an utterance y as containing a question if and only if y contains a question mark. We train our CT model on a control variable z with 11 possible values: $\{0, \ldots, 10\}$. As discussed in Section 5, we wish to control question-asking at the distributional, dialogue level, rather than at the binary, utterance level. Thus the setting $z = i$ means that the model should produce, on average, utterances containing '?' with probability $i/10$. During training we randomly assign examples to buckets such that each bucket i is trained on examples with the correct proportion of questions ($i/10$), and all buckets have the same amount of training examples.

We find that conditional training is effective to control question-asking – as shown in Figure 2, by increasing z from 0 to 10, we obtain a range of question-asking rates from 1.40% to 97.72%. However, when we introduce repetition control, question-asking is reduced – in particular, the $z = 10$ setting (which should produce 100% questions) now only produces 79.67% questions. The primary problem is the weighted decoding feature `extrep_bigram`, which discourages bigrams that have appeared in previous utterances – this prevents the model from producing bigrams that commonly occur in many questions, such as *do you* and *what is*. To fix this, we introduce an extra setting $z = 10$ *(boost)*, in which we do not use the feature `extrep_bigram` for weighted decoding during beam search, but we do use it to rerank the candidates after beam search. This setting, which allows the model to produce necessary question-asking bigrams, yields a 99.54% question-asking rate, at the cost of slightly increased external bigram repetition (see Appendix F).

For controlling question-asking, conditional training is preferable to weighted decoding for two reasons. Firstly, it allows us to achieve (close to) 0% questions, 100% questions, or anything in between, without introducing the risk of degenerate output. Secondly, presence-of-a-question-mark captures the true attribute of interest (question-asking) more exactly and directly than presence of interrogative words. For these reasons, only the CT method is considered in the human evaluation.

7 Comparison of control methods

The previous section shows that conditional training and weighted decoding are both useful techniques, with different strengths and weaknesses.

The primary disadvantage of conditional training is that it sometimes fails to learn the connection between the control variable z and the target output y. In practice, we find the model can learn simple attributes of the output (such as the presence of '?', and overall genericness), but not relationships between the input and output (such as semantic relatedness). By contrast, weighted decoding can force the desired feature to appear in the output by raising the weight arbitrarily high (though this may have unintended side-effects).

The primary disadvantage of weighted decoding is that it risks going off-distribution when the weight is too strong. By contrast, conditional training produces mostly well-formed, in-distribution outputs. This highlights the importance of learned control – it is safer to learn to produce output that both satisfies the control variable and is appropriate, than to alter the decoding process to satisfy the control variable, potentially trading off appropriateness in the process.

Other considerations include: (1) Convenience: conditional training requires retraining; weighted decoding doesn't, but is slower at test time. (2) Data availability: conditional training requires training examples of the controllable attribute, whereas weighted decoding can control any computable feature without requiring examples. (3) Attribute definition: conditional training can control sentence-level attributes, but they must be discrete. By contrast, weighted decoding requires word-level features, but they can be continuous.

8 Human evaluation results

In order to study the effect of our controllable attributes, we conduct a large-scale human evaluation of 28 model configurations (see Appendix E), plus human-human conversations for comparison.

Approach In our evaluation, a crowdworker chats with a model (or in the human-human case, another crowdworker) for six conversational turns, then answers eight multiple-choice questions which each capture different aspects of conversational quality: avoiding repetition, interestingness, making sense, fluency, listening, inquisitiveness, humanness and engagingness. The eight questions are Likert questions on a 1-4 scale, where higher is better.[5] To match the ConvAI2 Challenge, we also add a persona retrieval question, in which the crowdworker is asked to select which of two possible personas was the model's persona. For full details of the evaluation design, see Appendix B.

Our evaluation is the same as the ConvAI2 Challenge evaluation, but more detailed – ConvAI2 includes only engagingness and persona retrieval.[6] As in the ConvAI2 challenge, each of our 28 model configurations was evaluated by over 100 crowdworkers, and the results were adjusted for annotator variance via a Bayesian calibration (Kulikov et al., 2018).

In designing our evaluation, we aimed to capture the four aspects we expected to directly improve via control (avoiding repetition, interestingness, listening, inquisitiveness), two important error classes we thought would be affected by our controls (fluency, making sense), and two overall quality measures (engagingness, humanness).

8.1 Main findings

In this section we summarize the main findings of our human evaluation – whose full results can be found in Appendices G and H, with sample conversations in Appendix C.

As Figure 3 shows, controlling for repetition, specificity and question-asking all lead to large engagingness improvements over the greedy and beam-search baseline models. In particular, we find that controlling for multi-turn (self) repetition is important and should be incorporated alongside other attribute control methods. We found no improvement by controlling response-relatedness.

To better understand these overall engagingness improvements, we consider the full set of human judgments, shown in Figure 4. We find that reducing repetition leads to improvements across all our aspects of conversational quality. Increasing specificity shows improvements in interestingness and listening ability over the repetition-controlled baseline, while increasing question-asking shows improvements in inquisitiveness and interestingness over the repetition-controlled baseline.

Our most engaging model, which controls both repetition and question-asking – marked 'Question (CT)' in Figure 3 (left) – matches the engagingness of the winning entry in the ConvAI2 competition, as both models achieve a raw score[7] of 3.1 (Dinan et al., 2019). However, the ConvAI2 winner, Lost in Conversation, was trained on approximately 12× as much data as our model. Lost in Conversation is based on the OpenAI GPT Language Model (Radford et al., 2018), which is pretrained on the BookCorpus (Zhu et al., 2015), which contains approximately 985 million words, whereas our model is pretrained on the Twitter dataset (approximately 79 million words).

Altogether, our evaluation clearly shows that controlling low-level attributes over multiple turns leads to improved overall quality.

8.2 Effect of controlled attributes

Repetition (WD) We observe that self-repetition across utterances (*external repetition*) is by far the most severe form of repetition in our beam search baseline model. We evaluate several settings of the `extrep_bigram` weighted decoding feature, and find that an aggressive repetition-reduction setting (reducing bigram repetition rate to below gold data levels) is rated best. We also find that blocking repeated content words improves the avoiding repetition score. See Appendices E, F and G for full details.

As shown in Figure 3 (left) and Figure 4, our repetition-controlled model improves hugely

---

[5] Exceptions: Avoiding repetition is a 1-3 scale, as we found this gave clearer instructions. Inquisitiveness has an optimal score of 3; 1 and 2 represent too little question-asking, and 4 represents too much.

[6] There are three other minor differences between our evaluation and ConvAI2's: (1) We fix capitalization and spacing before showing the chatbot's utterances to crowdworkers, while ConvAI2 show the raw lowercase tokenized form. We found the latter interferes with fluency evaluation. (2) We conduct 6 dialogue turns, while ConvAI2 conducts 4-6. This was necessary to evaluate repetitiveness. (3) We use (publicly-available) validation set personas, while ConvAI2 uses (hidden) test set personas. This enables us to release our evaluation chatlogs.

[7] Although the same Bayesian calibration method was applied both in our study and in the ConvAI2 competition, calibrated scores are not comparable across the two; thus we compare raw scores (viewable in Table 7).

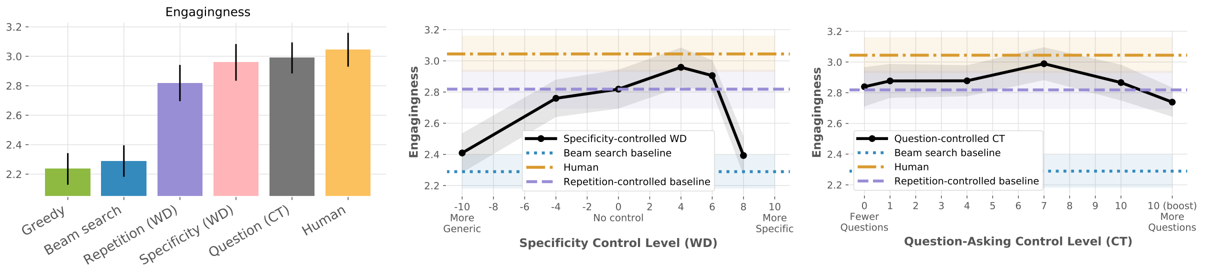


Figure 3: Calibrated human judgments of engagingness for the baselines and best controlled models (left); for different specificity control settings (middle); and for different question-asking control settings (right).

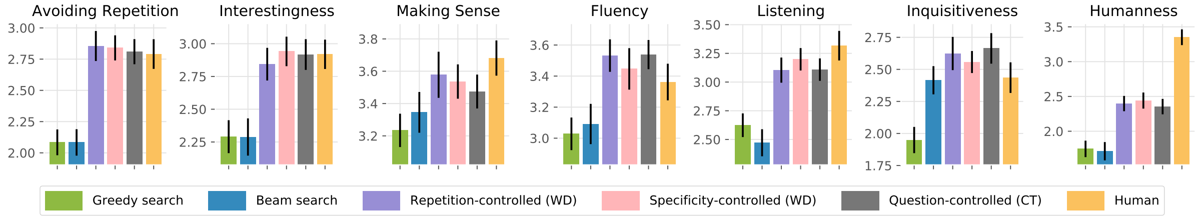


Figure 4: Calibrated human judgments of conversational aspects for the baselines and best controlled models. Note: In Figure 3 and here, the Specificity and Question controlled models both include Repetition control, but Question control doesn't include Specificity control, or vice versa.

over the beam search baseline in all metrics, and achieves close-to-human scores on all metrics except humanness. This striking result demonstrates that repetition is by far the biggest limiting quality factor for naive sequence-to-sequence dialogue agents. The result also emphasizes the importance of *multi-turn* dialogue evaluation to detect the problem. We refer to this model as the *repetition-controlled baseline*, and use it as a basis for all remaining experiments (i.e., we control specificity, response-relatedness and question-asking on top of these repetition-control settings).

Specificity (WD, CT) For our weighted decoding models, the extreme settings (very generic and very specific) score poorly in engagingness due to the frequent presence of degenerate output – see Figure 3 (middle). We find that the weight $= 4$ setting (which is more specific than the repetition-controlled baseline and about as specific as the gold data) maximizes engagingness. As shown in Figure 3 (left) and Figure 4, this more-specific model is rated more interesting, engaging, and a better listener than the repetition-controlled baseline, but at the cost of reduced fluency and making sense. Our CT model with $z = 7$ (which has a similar NIDF level as WD with weight $= 4$) shows similar results, but the improvements are smaller. For further discussion on the interestingness of our specificity models, see Section 8.3.

Response-relatedness (WD) We evaluated several control settings (weight $= -10, 5, 10, 13$) and found that none scored better than weight $= 0$ (no response-relatedness control); see Appendix H. This is surprising – prior to running the human evaluation, we annotated 100 examples ourselves to determine the best control settings. While we identified a more responsive setting (weight $= 5$) as less likely than the uncontrolled model to ignore the user, crowdworkers rated it as a slightly *worse* listener than the uncontrolled model. One explanation for this discrepancy is that the more responsive model takes more risks, using more rare words (0.197 NIDF, up from 0.178), and thus receives a lower makes-sense score (3.41, down from 3.70). We hypothesize that, compared to us, the crowdworkers are less tolerant of slightly nonsensical output, and more tolerant of generic unrelated utterances.

Question-asking (CT) As shown in Figure 3 (right), a question-asking rate of 65.7% ($z = 7$) maximizes engagingness. This setting, which asks more questions than both the repetition-controlled baseline (50.0%) and the human-produced gold data (28.8%), brings us closest to human-level engagingness – see Figure 3 (left). Although we find that a rate of approximately 65.7% question-asking is the most engaging, a lower level (48.9%, or $z = 4$) is rated the best listener. Lastly, we find

Model	Win%	Top 3 reasons for preferring model
Specificity WD (weight $= 6$)	84.1%	*More information; Better flow; More descriptive*
Specificity WD (weight $= 4$)	75.5%	*More information; They describe their life in more detail; Funny*
Specificity CT ($z = 7$)	56.2%	*More information; Better flow; Seems more interested*

Table 3: A/B tests comparing various specificity-controlled models to the repetition-controlled baseline on interestingness. We find all comparisons are significant ($p < .05$; binomial test).

that although asking too many questions is less engaging, most crowdworkers will not directly criticize a chatbot that asks questions on every turn – only 11.9% of crowdworkers judged the $z = 10$ *(boost)* setting, which asks 99.5% questions, as asking too many questions.[8] For full details of these scores, see Appendix F and H.

For time and budget reasons, we did not evaluate any models controlling both question-asking and specificity. However, we expect it is possible to obtain further improvements by doing so.

8.3 A/B tests for interestingness

Though our more-specific models yielded significant improvements in engagingness, we were surprised that they did not yield clearer improvements in interestingness. To investigate further, we conducted an A/B interestingness evaluation of three specificity-controlled models, compared to the repetition-controlled baseline. Crowdworkers were shown two conversations (from the main human evaluation) and asked to choose which model was more interesting (see Figure 7 for details). We collected 500 samples per comparison, plus 200 additional human vs repetition-controlled baseline samples, which were used to filter for quality control. After discarding low-quality crowdworkers, we have roughly 300 evaluations per comparison, with an average Cohen's $\kappa = 0.6$.

As shown in Table 3, all three models were rated significantly more interesting than the repetition-controlled baseline. This convincingly shows that producing utterances with more rare words is a valid strategy to improve interestingness. We have two explanations for why these interestingness differences did not materialize in our main evaluation. Firstly, interestingness is a particularly subjective metric (unlike more tangible metrics such as avoiding repetition and making sense) – this makes it hard to calibrate across crowdworkers. Secondly, we suspect that in our original evaluation, the crowdworkers may have evaluated the interestingness of the *task* rather than the *chatbot*. This could account for why subtle increases in conversational ability did not result in higher interestingness ratings – the PersonaChat task itself has a natural interestingness limit.

9 Conclusion

What makes a good conversation? Through our evaluation, we showed that a good conversation is about balance – controlling for the right level of repetition, specificity and question-asking is important for overall quality. We also found that conversational aspects such as interestingness, listening, and inquisitiveness are all important – though optimizing these can introduce a trade-off against certain types of errors (such as repetitive, disfluent, or nonsensical output). Secondly, multi-turn evaluation is essential to study what makes a good conversation – multiple turns are required to reveal issues such as repetition, consistency, and question-asking frequency. Lastly, what do we mean by 'good'? Although humanness and engagingness are both commonly used as overall quality metrics, the two are very different. While our models achieved close-to-human scores on engagingness, they failed to get close on humanness – showing that a chatbot need not be human-like to be enjoyable. This striking result also demonstrates the importance of measuring more than one quality metric when evaluating dialogue agents.

Outlook Our work shows that neural generative systems have systemic problems when applied to open-ended dialogue, some of which (e.g. repetition) are only observable in the multi-turn setting. Furthermore, control of low-level attributes offers a practical way to correct these problems, yielding large improvements to overall quality – in our case, comparable to systems trained on much more data. Future work includes optimizing control settings automatically, and building more convincingly human-like chatbots.

---

[8] Though this conclusion may hold true for the PersonaChat task – a synthetic chatting task that instructs participants to get to know each other – in real-life social conversations, incessant question-asking may be less tolerated.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3970–3980. Association for Computational Linguistics.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of Wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGDIAL Meeting on Discourse and Dialogue*, pages 207–219, Saarbrücken, Germany. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48. Association for Computational Linguistics.

Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. 2018. Topic-based evaluation for conversational bots. *Advances in Neural Information Processing Systems, Conversational AI Workshop*.

Helen Hastie. 2012. *Metrics and evaluation of spoken dialogue systems*, pages 131–150. Springer.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Thirty-fourth International Conference on Machine Learning*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338. Association for Computational Linguistics.

Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017a. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017b. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. pages 2122–2132.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126. Association for Computational Linguistics.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, Copenhagen, Denmark. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 690–695, Brussels, Belgium. Association for Computational Linguistics.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016a. Generative deep neural networks for dialogue: A short review. *Advances in Neural Information Processing Systems workshop on Learning Methods for Dialogue*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 504–509. Association for Computational Linguistics.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, et al. 2017. On evaluating and comparing conversational agents. *Advances in Neural Information Processing Systems, Conversational AI Workshop*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning, Deep Learning Workshop*, Lille, France.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 271–280, Madrid, Spain. Association for Computational Linguistics.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*, volume 17, pages 3351–3357.

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2018a. Learning to control the specificity in neural response generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1108–1117, Melbourne, Australia. Association for Computational Linguistics.

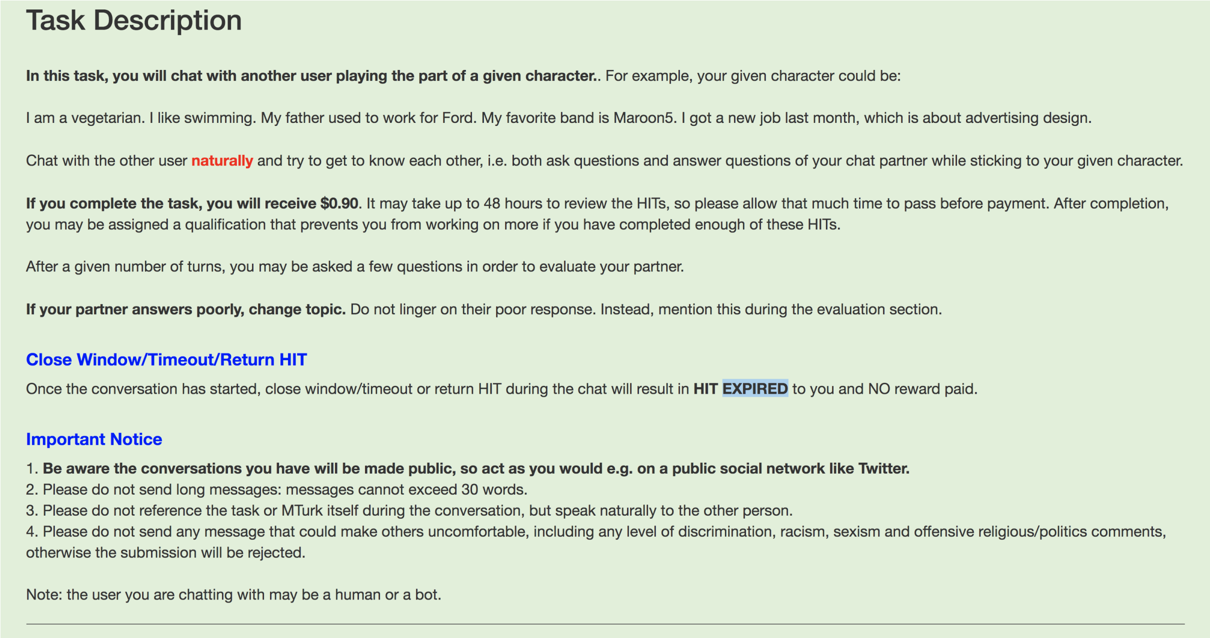
Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *AAAI*, pages 3400–3407.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Supplementary Material

A Screenshots of human evaluation interface



Task Description

In this task, you will chat with another user playing the part of a given character.. For example, your given character could be:

I am a vegetarian. I like swimming. My father used to work for Ford. My favorite band is Maroon5. I got a new job last month, which is about advertising design.

Chat with the other user **naturally** and try to get to know each other, i.e. both ask questions and answer questions of your chat partner while sticking to your given character.

If you complete the task, you will receive $0.90. It may take up to 48 hours to review the HITs, so please allow that much time to pass before payment. After completion, you may be assigned a qualification that prevents you from working on more if you have completed enough of these HITs.

After a given number of turns, you may be asked a few questions in order to evaluate your partner.

If your partner answers poorly, change topic. Do not linger on their poor response. Instead, mention this during the evaluation section.

Close Window/Timeout/Return HIT

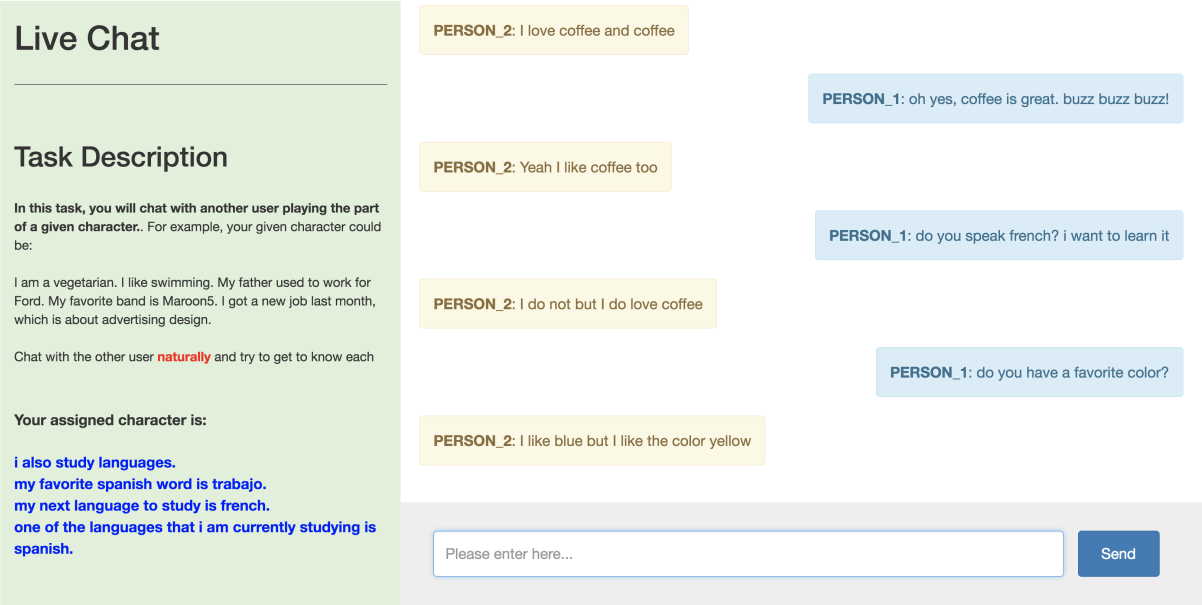
Once the conversation has started, close window/timeout or return HIT during the chat will result in **HIT EXPIRED** to you and NO reward paid.

Important Notice

1. **Be aware the conversations you have will be made public, so act as you would e.g. on a public social network like Twitter.**
2. Please do not send long messages: messages cannot exceed 30 words.
3. Please do not reference the task or MTurk itself during the conversation, but speak naturally to the other person.
4. Please do not send any message that could make others uncomfortable, including any level of discrimination, racism, sexism and offensive religious/politics comments, otherwise the submission will be rejected.

Note: the user you are chatting with may be a human or a bot.

Figure 5: Screenshot of the Task Description



Live Chat

Task Description

In this task, you will chat with another user playing the part of a given character.. For example, your given character could be:

I am a vegetarian. I like swimming. My father used to work for Ford. My favorite band is Maroon5. I got a new job last month, which is about advertising design.

Chat with the other user **naturally** and try to get to know each

Your assigned character is:

i also study languages.
my favorite spanish word is trabajo.
my next language to study is french.
one of the languages that i am currently studying is spanish.

PERSON_2: I love coffee and coffee

PERSON_1: oh yes, coffee is great. buzz buzz buzz!

PERSON_2: Yeah I like coffee too

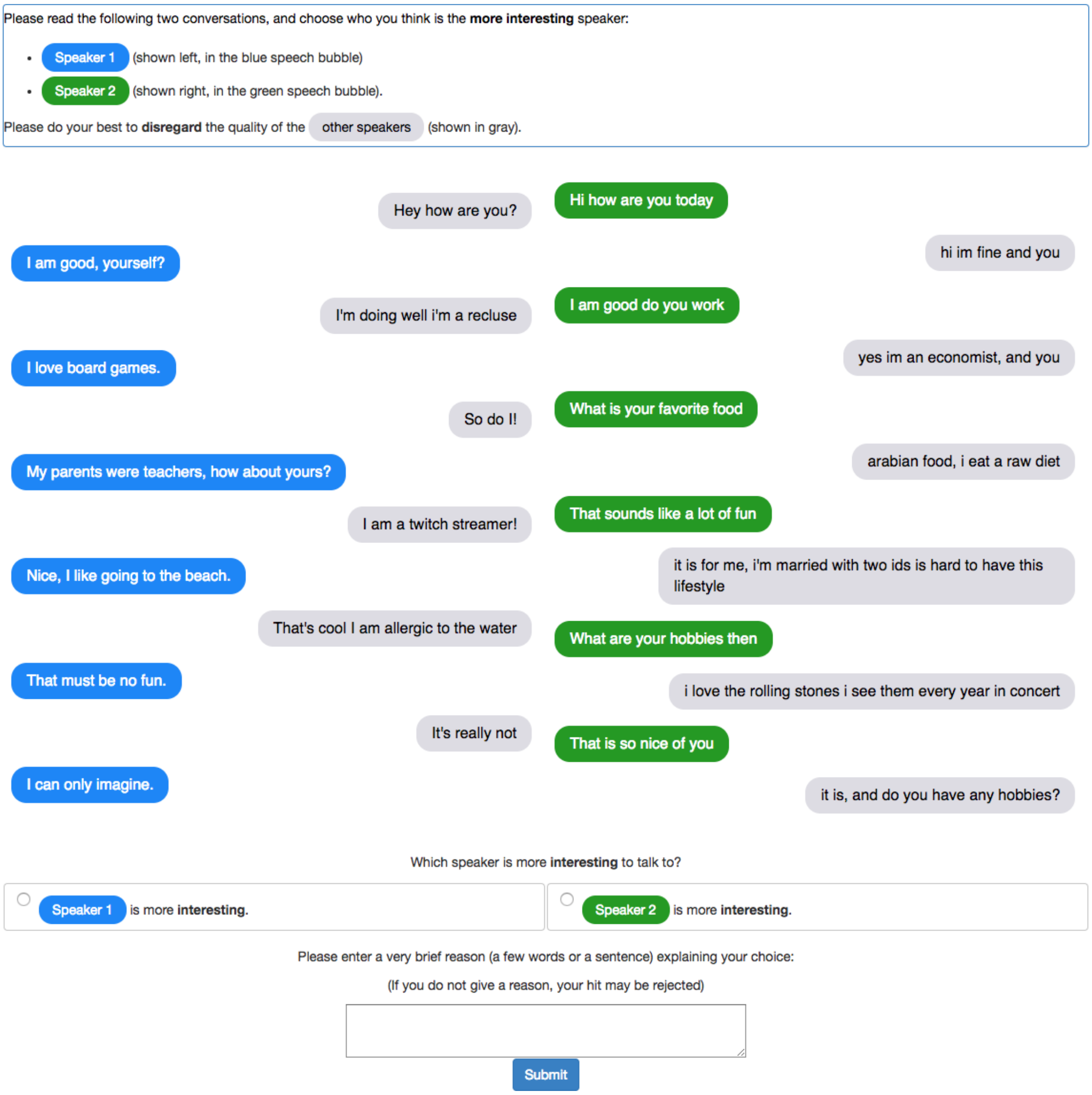
PERSON_1: do you speak french? i want to learn it

PERSON_2: I do not but I do love coffee

PERSON_1: do you have a favorite color?

PERSON_2: I like blue but I like the color yellow

Figure 6: Screenshot of the chat UI, talking with the beam search baseline model.



Please read the following two conversations, and choose who you think is the **more interesting** speaker:

- Speaker 1 (shown left, in the blue speech bubble)
- Speaker 2 (shown right, in the green speech bubble).

Please do your best to **disregard** the quality of the other speakers (shown in gray).

Hey how are you?

I am good, yourself?

I'm doing well i'm a recluse

I love board games.

So do I!

My parents were teachers, how about yours?

I am a twitch streamer!

Nice, I like going to the beach.

That's cool I am allergic to the water

That must be no fun.

It's really not

I can only imagine.

Hi how are you today

hi im fine and you

I am good do you work

yes im an economist, and you

What is your favorite food

arabian food, i eat a raw diet

That sounds like a lot of fun

it is for me, i'm married with two ids is hard to have this lifestyle

What are your hobbies then

i love the rolling stones i see them every year in concert

That is so nice of you

it is, and do you have any hobbies?

Which speaker is more **interesting** to talk to?

○ Speaker 1 is more **interesting**.

○ Speaker 2 is more **interesting**.

Please enter a very brief reason (a few words or a sentence) explaining your choice:

(If you do not give a reason, your hit may be rejected)

Figure 7: Screenshot of the A/B test UI, comparing a human-human conversation (left) and a Repetition-controlled baseline model (right).

B Human evaluation questionnaire design

Here are the questions and multiple-choice options used in the human evaluation, in the order presented:

[Engagingness] How much did you enjoy talking to this user?
• Not at all • A little • Somewhat • A lot

[Interestingness] How interesting or boring did you find this conversation?
• Very boring • A little boring • A little interesting • Very interesting

[Inquisitiveness] How much did the user try to get to know you?
• Didn't ask about me at all • Asked about me some
• Asked about me a good amount • Asked about me too much

[Listening] How much did the user seem to pay attention to what you said?
• Always ignored what I said • Mostly ignored what I said
• Mostly paid attention to what I said • Always paid attention to what I said

[Avoiding Repetition] How repetitive was this user?
• Repeated themselves over and over • Sometimes said the same thing twice
• Always said something new

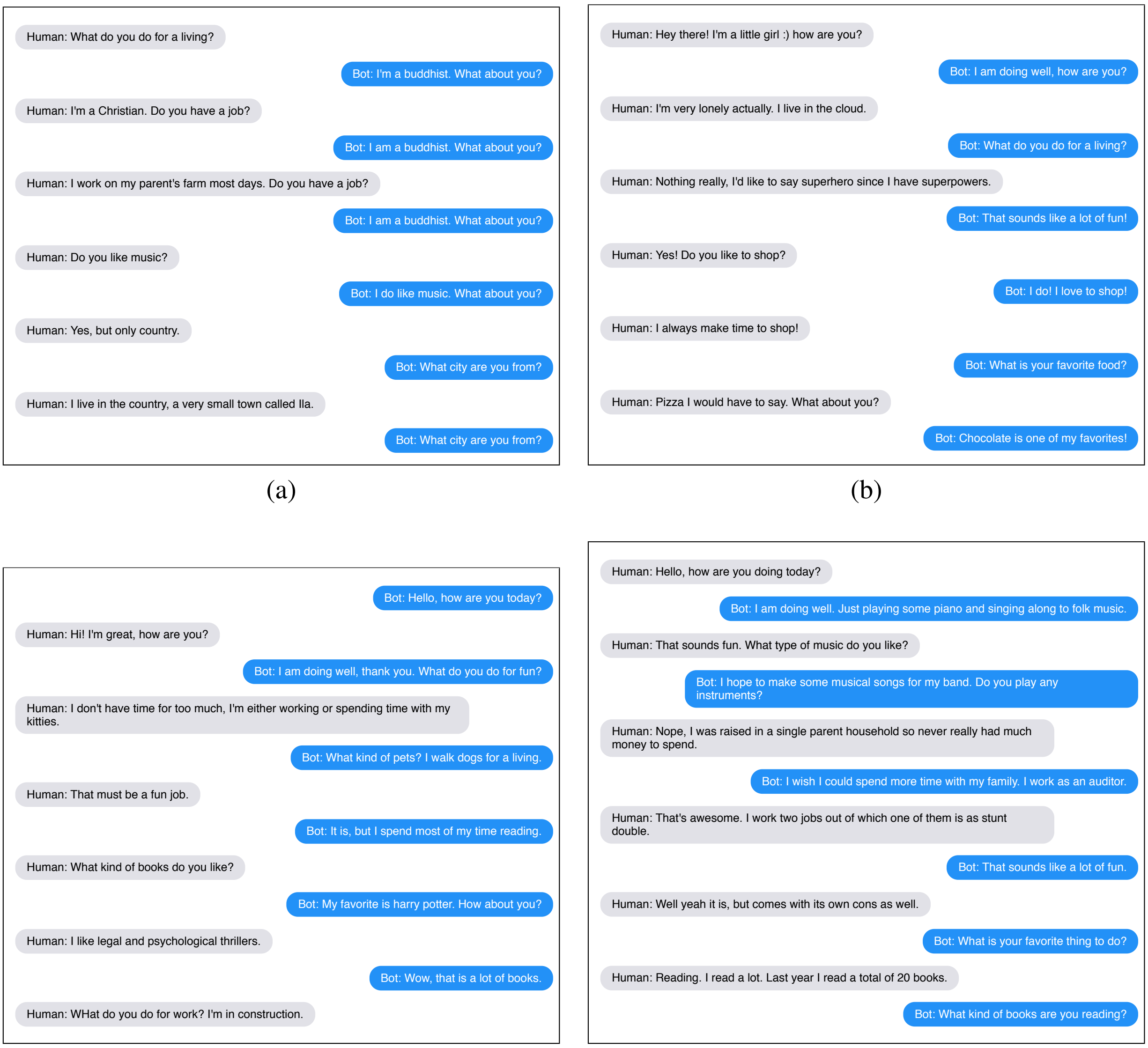
[Fluency] How naturally did this user speak English?
• Very unnatural • Mostly unnatural • Mostly natural • Very natural

[Making sense] How often did this user say something which did NOT make sense?
• Never made any sense • Most responses didn't make sense
• Some responses didn't make sense • Everything made perfect sense

[Humanness] Do you think this user is a bot or a human?
• Definitely a bot • Probably a bot • Probably a human • Definitely a human

[Persona retrieval] Which prompt (character) do you think the other user was given for this conversation?
Respondent chooses one of two provided personas

C Example conversations from human evaluation



Human: What do you do for a living?

Bot: I'm a buddhist. What about you?

Human: I'm a Christian. Do you have a job?

Bot: I am a buddhist. What about you?

Human: I work on my parent's farm most days. Do you have a job?

Bot: I am a buddhist. What about you?

Human: Do you like music?

Bot: I do like music. What about you?

Human: Yes, but only country.

Bot: What city are you from?

Human: I live in the country, a very small town called Ila.

Bot: What city are you from?

(a)

Human: Hey there! I'm a little girl :) how are you?

Bot: I am doing well, how are you?

Human: I'm very lonely actually. I live in the cloud.

Bot: What do you do for a living?

Human: Nothing really, I'd like to say superhero since I have superpowers.

Bot: That sounds like a lot of fun!

Human: Yes! Do you like to shop?

Bot: I do! I love to shop!

Human: I always make time to shop!

Bot: What is your favorite food?

Human: Pizza I would have to say. What about you?

Bot: Chocolate is one of my favorites!

(b)

Bot: Hello, how are you today?

Human: Hi! I'm great, how are you?

Bot: I am doing well, thank you. What do you do for fun?

Human: I don't have time for too much, I'm either working or spending time with my kitties.

Bot: What kind of pets? I walk dogs for a living.

Human: That must be a fun job.

Bot: It is, but I spend most of my time reading.

Human: What kind of books do you like?

Bot: My favorite is harry potter. How about you?

Human: I like legal and psychological thrillers.

Bot: Wow, that is a lot of books.

Human: WHat do you do for work? I'm in construction.

(c)

Human: Hello, how are you doing today?

Bot: I am doing well. Just playing some piano and singing along to folk music.

Human: That sounds fun. What type of music do you like?

Bot: I hope to make some musical songs for my band. Do you play any instruments?

Human: Nope, I was raised in a single parent household so never really had much money to spend.

Bot: I wish I could spend more time with my family. I work as an auditor.

Human: That's awesome. I work two jobs out of which one of them is as stunt double.

Bot: That sounds like a lot of fun.

Human: Well yeah it is, but comes with its own cons as well.

Bot: What is your favorite thing to do?

Human: Reading. I read a lot. Last year I read a total of 20 books.

Bot: What kind of books are you reading?

(d)

Figure 8: Example conversation with (a) Baseline (b) Repetition-controlled baseline (c) Question-controlled CT ($z = 7$), (d) Specificity-controlled WD (weight $= 4$).

D Repetition-control decoding features

Feature	**Condition**
`extrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears in a previous utterance by the model
`extrep_unigram`$(w, y_{<t}, x)$	w is a non-stopword **and** w appears in a previous utterance by the model
`intrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears earlier in the hypothesis $y_{<t}$
`intrep_unigram`$(w, y_{<t}, x)$	w is a non-stopword **and** w appears earlier in the hypothesis $y_{<t}$
`partnerrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears in a previous utterance by the partner

Table 4: We define five binary features for controlling different types of repetition via weighted decoding (see Section 5.2). Each feature depends on the word w, the partial hypothesis $y_{<t}$, and the context x (which includes the model's own persona and the dialogue history). Each of these features is equal to 1 if and only if the condition on the right is true; otherwise 0.

E Control settings for all configurations

	Repetition					**Specificity**	**Response-rel**	**Questions**
	External		**Internal**		**Partner Rep.**			
	Bigram	**Unigram**	**Bigram**	**Unigram**	**Bigram**	**NIDF**	**Cos sim**	**Has '?'**
Baselines								
Greedy Search								
Beam Search (beam size 20)								
Repetition control (WD)								
Extrep bigram WD -0.5	wt -0.5							
Extrep bigram WD -1.25	wt -1.25							
Extrep bigram WD -3.5	wt -3.5							
Extrep bigram WD -inf	wt $-\infty$							
Repetition-controlled baseline	wt -3.5	wt $-\infty$		wt $-\infty$				
Question control (CT)								
Question-controlled CT 0	wt -3.5	wt $-\infty$		wt $-\infty$				$z = 0$
Question-controlled CT 1	wt -3.5	wt $-\infty$		wt $-\infty$				$z = 1$
Question-controlled CT 4	wt -3.5	wt $-\infty$		wt $-\infty$				$z = 4$
Question-controlled CT 7	wt -3.5	wt $-\infty$		wt $-\infty$				$z = 7$
Question-controlled CT 10	wt -3.5	wt $-\infty$		wt $-\infty$				$z = 10$
Question-controlled CT 10 (boost)	wt 0 *	wt $-\infty$		wt $-\infty$				$z = 10$
Specificity control (CT)								
Specificity-controlled CT 0	wt -3.5	wt $-\infty$		wt $-\infty$		$z = 0$		
Specificity-controlled CT 2	wt -3.5	wt $-\infty$		wt $-\infty$		$z = 2$		
Specificity-controlled CT 4	wt -3.5	wt $-\infty$		wt $-\infty$		$z = 4$		
Specificity-controlled CT 7	wt -3.5	wt $-\infty$		wt $-\infty$		$z = 7$		
Specificity-controlled CT 9	wt -3.5	wt $-\infty$		wt $-\infty$		$z = 9$		
Specificity control (WD)								
Specificity-controlled WD -10	wt -3.5	wt $-\infty$		wt $-\infty$		wt -10		
Specificity-controlled WD -4	wt -3.5	wt $-\infty$		wt $-\infty$		wt -4		
Specificity-controlled WD 4	wt -3.5	wt $-\infty$		wt $-\infty$		wt 4		
Specificity-controlled WD 6	wt -3.5	wt $-\infty$		wt $-\infty$		wt 6		
Specificity-controlled WD 8	wt -3.5	wt $-\infty$		wt $-\infty$		wt 8		
Response-related control (WD) **								
Response-related controlled WD -10	wt -3.5	wt $-\infty$	wt $-\infty$	wt $-\infty$	wt $-\infty$		wt -10	
Response-related controlled WD 0	wt -3.5	wt $-\infty$	wt $-\infty$	wt $-\infty$	wt $-\infty$		wt 0	
Response-related controlled WD 5	wt -3.5	wt $-\infty$	wt $-\infty$	wt $-\infty$	wt $-\infty$		wt 5	
Response-related controlled WD 10	wt -3.5	wt $-\infty$	wt $-\infty$	wt $-\infty$	wt $-\infty$		wt 10	
Response-related controlled WD 13	wt -3.5	wt $-\infty$	wt $-\infty$	wt $-\infty$	wt $-\infty$		wt 13	

Table 5: Control settings for all configurations that were human-evaluated. 'wt' means the weight used for a weighted decoding feature and '$z =$' means the setting (i.e. bucket) for the control variable in conditional training.

* In the setting Question-controlled CT 10 (boost), the feature `extrep_bigram` is *not* used for weighted decoding during beam search, but it *is* used to rerank the candidates after beam search. See Section 6.4 for details.

** Note that the Response-related controlled models additionally introduce repetition controls to block internal bigram repetition and partner bigram repetition. This was necessary to prevent the model from parroting the partner's last utterance. In Table 8, we find that just adding these extra repetition controls (here called Response-related controlled WD 0, i.e. increased repetition control but no response-relatedness control) outperforms our canonical Repetition-controlled baseline. However, given that we discovered this later, our specificity and question controlled models are built on top of the canonical Repetition-controlled baseline.

F Automatic metrics for all configurations

	Repetition					**Specificity**	**Response-rel**	**Questions**
	External		**Internal**		**Partner Rep.**			
	Bigram	**Unigram**	**Bigram**	**Unigram**	**Bigram**	**NIDF**	**Cos sim**	**Has '?'**
Gold data and baselines								
Gold Data	4.65%	9.62%	0.38%	0.97%	5.10%	0.2119	0.1691	28.80%
Greedy Search	35.88%	36.31%	8.08%	10.59%	12.20%	0.1688	0.1850	6.46%
Beam Search (beam size 20)	46.85%	44.15%	0.32%	0.61%	12.90%	0.1662	0.0957	80.87%
Repetition control (WD)								
Extrep bigram WD -0.5	19.70%	16.85%	0.26%	0.62%	11.93%	0.1730	0.1348	73.04%
Extrep bigram WD -1.25	4.62%	4.79%	0.40%	0.89%	10.61%	0.1763	0.1504	61.22%
Extrep bigram WD -3.5	0.75%	4.61%	0.47%	0.94%	9.89%	0.1771	0.1681	48.89%
Extrep bigram WD -inf	0.00%	4.74%	0.51%	1.05%	9.56%	0.1780	0.1711	45.98%
Repetition-controlled baseline	0.73%	0.00%	0.17%	0.00%	9.55%	0.1766	0.1676	49.98%
Question control (CT)								
Question-controlled CT 0	0.06%	0.00%	0.19%	0.00%	9.20%	0.1871	0.1753	2.01%
Question-controlled CT 1	0.09%	0.00%	0.19%	0.00%	8.66%	0.1844	0.1722	17.33%
Question-controlled CT 4	0.40%	0.00%	0.25%	0.00%	8.53%	0.1794	0.1713	48.88%
Question-controlled CT 7	0.80%	0.00%	0.17%	0.00%	8.48%	0.1771	0.1724	65.65%
Question-controlled CT 10	1.27%	0.00%	0.16%	0.00%	8.48%	0.1761	0.1728	79.67%
Question-controlled CT 10 (boost)*	7.64%	0.00%	0.03%	0.00%	10.76%	0.1701	0.1651	99.54%
Specificity control (CT)								
Specificity-controlled CT 0	0.60%	0.00%	0.20%	0.00%	9.05%	0.1478	0.1522	48.75%
Specificity-controlled CT 2	0.28%	0.00%	0.10%	0.00%	8.37%	0.1772	0.1833	50.57%
Specificity-controlled CT 4	0.12%	0.00%	0.08%	0.00%	7.90%	0.1921	0.1877	29.46%
Specificity-controlled CT 7	0.02%	0.00%	0.14%	0.00%	8.17%	0.2156	0.1955	16.51%
Specificity-controlled CT 9	0.01%	0.00%	0.11%	0.00%	8.01%	0.2462	0.1990	8.50%
Specificity control (WD)								
Specificity-controlled WD -10	0.14%	0.00%	10.59%	0.00%	8.70%	0.1107	0.0994	33.55%
Specificity-controlled WD -4	0.65%	0.00%	1.98%	0.00%	9.95%	0.1501	0.1398	44.92%
Specificity-controlled WD 4	0.15%	0.00%	0.19%	0.00%	7.54%	0.2121	0.1972	45.53%
Specificity-controlled WD 6	0.07%	0.00%	0.13%	0.00%	6.50%	0.2546	0.2040	39.37%
Specificity-controlled WD 8	0.01%	0.00%	0.10%	0.00%	3.40%	0.4035	0.1436	26.68%
Response-related control (WD)								
Response-related controlled WD -10	0.13%	0.00%	0.00%	0.00%	0.00%	0.1914	-0.0921	25.71%
Response-related controlled WD 0	0.24%	0.00%	0.00%	0.00%	0.00%	0.1785	0.1414	44.55%
Response-related controlled WD 5	0.15%	0.00%	0.00%	0.00%	0.00%	0.1973	0.4360	39.78%
Response-related controlled WD 10	0.05%	0.00%	0.00%	0.00%	0.00%	0.2535	0.6653	27.56%
Response-related controlled WD 13	0.02%	0.00%	0.00%	0.00%	0.00%	0.2999	0.7251	20.47%

Table 6: Automatic metrics (computed over validation set) for all model configurations that were human-evaluated.

G Human evaluation results for all configurations

Model	Avoiding Rep.	Engage	Fluency	Humanness	Inquisitive	Interesting	Listening	Make Sense	Persona
Human and baselines									
Human	2.90 ± 0.39	3.31 ± 0.90	3.66 ± 0.71	3.40 ± 0.80	2.63 ± 0.63	3.23 ± 0.83	3.64 ± 0.63	3.84 ± 0.52	0.92 ± 0.27
Greedy Search	2.16 ± 0.72	2.31 ± 1.08	3.20 ± 0.81	1.78 ± 0.90	2.00 ± 0.81	2.36 ± 0.98	2.78 ± 0.84	3.33 ± 0.75	0.87 ± 0.34
Beam Search (beam size 20)	2.14 ± 0.72	2.35 ± 1.01	3.23 ± 0.93	1.81 ± 0.87	2.50 ± 0.72	2.35 ± 0.98	2.63 ± 0.85	3.40 ± 0.77	0.77 ± 0.42
Repetition control (WD)									
Extrep bigram WD -0.5	2.66 ± 0.56	2.56 ± 0.92	3.57 ± 0.64	2.19 ± 0.94	2.67 ± 0.62	2.61 ± 0.87	3.08 ± 0.78	3.60 ± 0.57	0.75 ± 0.43
Extrep bigram WD -1.25	2.84 ± 0.39	2.91 ± 0.90	3.59 ± 0.64	2.32 ± 0.98	2.63 ± 0.60	2.86 ± 0.89	3.21 ± 0.71	3.64 ± 0.62	0.72 ± 0.45
Extrep bigram WD -3.5	2.90 ± 0.30	2.95 ± 0.86	**3.73 ± 0.50**	2.45 ± 1.03	2.55 ± 0.61	2.88 ± 0.80	3.27 ± 0.79	3.68 ± 0.49	0.80 ± 0.40
Extrep bigram WD -inf	2.82 ± 0.43	2.96 ± 0.86	3.64 ± 0.58	2.40 ± 0.96	2.65 ± 0.69	2.86 ± 0.82	3.31 ± 0.69	3.66 ± 0.59	0.91 ± 0.29
Repetition-controlled baseline	2.89 ± 0.39	2.89 ± 0.89	3.66 ± 0.56	2.50 ± 0.99	2.70 ± 0.64	2.96 ± 0.92	3.25 ± 0.71	3.68 ± 0.54	0.87 ± 0.34
Question control (CT)									
Question-controlled CT 0	2.95 ± 0.25	2.92 ± 0.90	3.70 ± 0.54	2.49 ± 0.97	2.48 ± 0.72	2.85 ± 0.93	3.29 ± 0.69	3.56 ± 0.66	0.86 ± 0.35
Question-controlled CT 1	2.88 ± 0.33	2.94 ± 0.93	3.59 ± 0.66	2.47 ± 0.95	2.52 ± 0.69	2.85 ± 0.90	3.32 ± 0.73	3.63 ± 0.55	0.85 ± 0.36
Question-controlled CT 4	2.88 ± 0.38	2.88 ± 0.94	3.59 ± 0.73	2.42 ± 1.07	2.55 ± 0.66	2.82 ± 0.85	**3.37 ± 0.74**	3.63 ± 0.59	0.84 ± 0.37
Question-controlled CT 7	2.88 ± 0.37	**3.07 ± 0.90**	3.67 ± 0.54	2.42 ± 0.98	2.75 ± 0.58	2.97 ± 0.84	3.23 ± 0.76	3.53 ± 0.76	0.80 ± 0.40
Question-controlled CT 10	2.74 ± 0.46	2.90 ± 0.93	3.70 ± 0.50	2.43 ± 1.04	2.71 ± 0.57	2.72 ± 0.88	3.12 ± 0.73	3.59 ± 0.66	0.79 ± 0.41
Question-controlled CT 10 (boost)	2.76 ± 0.49	2.84 ± 0.94	3.60 ± 0.64	2.26 ± 0.97	**2.94 ± 0.57**	2.83 ± 0.94	3.18 ± 0.80	3.52 ± 0.67	0.72 ± 0.45
Specificity control (CT)									
Specificity-controlled CT 0	2.83 ± 0.40	2.96 ± 0.93	3.62 ± 0.58	2.42 ± 0.99	2.60 ± 0.56	2.86 ± 0.89	3.29 ± 0.70	3.66 ± 0.60	0.72 ± 0.45
Specificity-controlled CT 2	2.90 ± 0.36	2.78 ± 1.00	3.60 ± 0.64	2.37 ± 0.93	2.66 ± 0.66	2.80 ± 0.96	3.14 ± 0.77	3.50 ± 0.63	0.81 ± 0.39
Specificity-controlled CT 4	2.92 ± 0.27	2.81 ± 0.88	3.65 ± 0.59	2.34 ± 1.02	2.57 ± 0.62	2.80 ± 0.78	3.25 ± 0.78	3.50 ± 0.66	0.86 ± 0.35
Specificity-controlled CT 7	2.89 ± 0.32	3.00 ± 0.94	3.64 ± 0.67	2.53 ± 1.03	2.56 ± 0.66	2.90 ± 0.90	3.34 ± 0.70	3.59 ± 0.60	0.82 ± 0.39
Specificity-controlled CT 9	2.90 ± 0.35	2.83 ± 0.87	3.61 ± 0.62	2.40 ± 0.97	2.31 ± 0.74	2.84 ± 0.83	3.07 ± 0.81	3.58 ± 0.56	0.88 ± 0.32
Specificity control (WD)									
Specificity-controlled WD -10	2.85 ± 0.43	2.43 ± 0.99	3.34 ± 0.83	2.15 ± 0.91	2.31 ± 0.69	2.38 ± 0.94	3.03 ± 0.75	3.33 ± 0.70	0.71 ± 0.45
Specificity-controlled WD -4	2.90 ± 0.30	2.78 ± 0.95	3.55 ± 0.63	2.41 ± 0.92	2.52 ± 0.66	2.64 ± 0.93	3.28 ± 0.73	3.56 ± 0.62	0.82 ± 0.38
Specificity-controlled WD 4	2.95 ± 0.21	2.99 ± 0.86	3.65 ± 0.55	2.49 ± 0.90	2.65 ± 0.55	3.00 ± 0.78	**3.37 ± 0.59**	3.63 ± 0.50	**0.93 ± 0.25**
Specificity-controlled WD 6	2.93 ± 0.26	2.96 ± 0.90	3.52 ± 0.76	2.41 ± 1.04	2.58 ± 0.66	**3.06 ± 0.80**	3.24 ± 0.76	3.50 ± 0.66	**0.93 ± 0.26**
Specificity-controlled WD 8	2.78 ± 0.52	2.40 ± 1.23	2.67 ± 1.25	1.86 ± 0.97	2.03 ± 0.87	2.55 ± 1.14	2.61 ± 1.05	2.91 ± 0.91	0.92 ± 0.28
Response-related control (WD)									
Response-related controlled WD -10	2.86 ± 0.44	2.48 ± 0.98	3.42 ± 0.74	2.02 ± 0.93	2.38 ± 0.75	2.53 ± 0.94	2.84 ± 0.80	3.14 ± 0.75	0.91 ± 0.29
Response-related controlled WD 0	**2.96 ± 0.23**	3.01 ± 0.90	3.72 ± 0.54	**2.73 ± 1.00**	2.56 ± 0.67	2.92 ± 0.84	**3.37 ± 0.72**	**3.73 ± 0.52**	0.82 ± 0.38
Response-related controlled WD 5	2.90 ± 0.33	2.88 ± 0.90	3.51 ± 0.63	2.41 ± 1.01	2.53 ± 0.65	2.85 ± 0.90	3.27 ± 0.73	3.49 ± 0.63	0.82 ± 0.39
Response-related controlled WD 10	2.78 ± 0.43	2.39 ± 1.04	3.06 ± 0.90	1.97 ± 0.99	2.22 ± 0.67	2.57 ± 1.01	3.03 ± 0.76	3.16 ± 0.63	0.75 ± 0.43
Response-related controlled WD 13	2.71 ± 0.57	2.10 ± 1.13	2.54 ± 1.12	1.81 ± 1.07	2.14 ± 0.84	2.33 ± 1.06	2.69 ± 0.83	2.70 ± 0.88	0.62 ± 0.49

Table 7: Raw scores (mean ± std.) for all models and human evaluation metrics.

The first eight columns are Likert metrics on a 1-4 scale (except Avoiding Repetition, which is a 1-3 scale), where higher is better (except Inquisitiveness, which has an optimal score of 3). The last column, Persona Retrieval, is on a scale from 0 to 1 where higher is better.

The maximum of each column (excluding Human row) is in bold.

Model	Avoiding Rep.	Engage	Fluency	Humanness	Inquisitive	Interesting	Listening	Make Sense
Human and baselines								
* Human	2.79 ± 0.12	3.04 ± 0.11	3.36 ± 0.12	3.35 ± 0.11	2.44 ± 0.12	2.92 ± 0.11	3.32 ± 0.13	3.68 ± 0.11
* Greedy Search	2.08 ± 0.10	2.24 ± 0.11	3.03 ± 0.10	1.75 ± 0.12	1.95 ± 0.10	2.29 ± 0.13	2.62 ± 0.10	3.23 ± 0.10
* Beam Search (beam size 20)	2.08 ± 0.11	2.29 ± 0.11	3.09 ± 0.13	1.71 ± 0.13	2.42 ± 0.11	2.29 ± 0.14	2.47 ± 0.12	3.35 ± 0.13
Repetition control (WD)								
Extrep bigram WD -0.5	2.62 ± 0.10	2.54 ± 0.12	3.35 ± 0.12	2.13 ± 0.11	2.63 ± 0.11	2.56 ± 0.11	2.93 ± 0.11	3.48 ± 0.11
Extrep bigram WD -1.25	2.78 ± 0.09	2.82 ± 0.13	3.40 ± 0.12	2.27 ± 0.12	2.54 ± 0.09	2.76 ± 0.10	3.05 ± 0.11	3.53 ± 0.14
Extrep bigram WD -3.5	2.83 ± 0.11	2.93 ± 0.10	**3.56 ± 0.10**	2.43 ± 0.11	2.47 ± 0.11	2.83 ± 0.10	3.14 ± 0.10	3.62 ± 0.12
Extrep bigram WD -inf	2.74 ± 0.11	2.87 ± 0.14	3.49 ± 0.12	2.32 ± 0.13	2.56 ± 0.11	2.75 ± 0.12	3.13 ± 0.12	3.59 ± 0.12
* Repetition-controlled baseline	2.86 ± 0.12	2.82 ± 0.12	3.53 ± 0.10	2.40 ± 0.11	2.62 ± 0.13	2.84 ± 0.12	3.10 ± 0.11	3.58 ± 0.14
Question control (CT)								
Question-controlled CT 0	**2.87 ± 0.12**	2.84 ± 0.13	3.51 ± 0.10	2.46 ± 0.11	2.36 ± 0.09	2.76 ± 0.09	3.10 ± 0.10	3.49 ± 0.12
Question-controlled CT 1	2.82 ± 0.11	2.88 ± 0.11	3.42 ± 0.10	2.46 ± 0.12	2.47 ± 0.11	2.79 ± 0.13	3.14 ± 0.11	3.55 ± 0.10
Question-controlled CT 4	2.78 ± 0.12	2.88 ± 0.10	3.47 ± 0.11	2.40 ± 0.09	2.53 ± 0.13	2.83 ± 0.13	**3.24 ± 0.11**	3.59 ± 0.10
* Question-controlled CT 7	2.81 ± 0.10	**2.99 ± 0.11**	3.54 ± 0.09	2.35 ± 0.11	2.66 ± 0.12	2.92 ± 0.12	3.11 ± 0.10	3.47 ± 0.10
Question-controlled CT 10	2.67 ± 0.13	2.87 ± 0.11	3.52 ± 0.12	2.35 ± 0.12	2.63 ± 0.12	2.66 ± 0.10	2.94 ± 0.11	3.53 ± 0.12
Question-controlled CT 10 (boost)	2.68 ± 0.12	2.74 ± 0.09	3.42 ± 0.12	2.19 ± 0.13	**2.79 ± 0.11**	2.74 ± 0.11	3.00 ± 0.12	3.45 ± 0.13
Specificity control (CT)								
Specificity-controlled CT 0	2.79 ± 0.10	2.93 ± 0.09	3.44 ± 0.12	2.38 ± 0.11	2.56 ± 0.12	2.84 ± 0.12	3.12 ± 0.13	3.61 ± 0.11
Specificity-controlled CT 2	2.78 ± 0.12	2.74 ± 0.11	3.39 ± 0.13	2.31 ± 0.13	2.56 ± 0.13	2.74 ± 0.12	2.99 ± 0.11	3.47 ± 0.10
Specificity-controlled CT 4	2.82 ± 0.10	2.80 ± 0.13	3.44 ± 0.14	2.32 ± 0.13	2.51 ± 0.12	2.78 ± 0.15	3.09 ± 0.13	3.46 ± 0.13
Specificity-controlled CT 7	2.81 ± 0.12	2.91 ± 0.13	3.43 ± 0.11	2.45 ± 0.10	2.49 ± 0.11	2.81 ± 0.12	3.15 ± 0.12	3.55 ± 0.11
Specificity-controlled CT 9	2.80 ± 0.13	2.78 ± 0.10	3.41 ± 0.12	2.35 ± 0.13	2.28 ± 0.11	2.79 ± 0.11	2.91 ± 0.11	3.51 ± 0.12
Specificity control (WD)								
Specificity-controlled WD -10	2.76 ± 0.11	2.41 ± 0.12	3.19 ± 0.12	2.15 ± 0.11	2.28 ± 0.13	2.35 ± 0.12	2.89 ± 0.11	3.28 ± 0.12
Specificity-controlled WD -4	2.83 ± 0.10	2.76 ± 0.12	3.37 ± 0.10	2.36 ± 0.11	2.46 ± 0.11	2.62 ± 0.12	3.14 ± 0.09	3.52 ± 0.11
* Specificity-controlled WD 4	2.84 ± 0.10	2.96 ± 0.12	3.45 ± 0.13	2.44 ± 0.12	2.56 ± 0.09	**2.94 ± 0.11**	3.20 ± 0.10	3.54 ± 0.11
Specificity-controlled WD 6	2.81 ± 0.09	2.91 ± 0.10	3.34 ± 0.09	2.31 ± 0.11	2.53 ± 0.12	2.93 ± 0.12	3.09 ± 0.10	3.41 ± 0.12
Specificity-controlled WD 8	2.70 ± 0.11	2.39 ± 0.12	2.54 ± 0.12	1.80 ± 0.13	2.00 ± 0.10	2.49 ± 0.12	2.47 ± 0.10	2.87 ± 0.11
Response-related control (WD)								
Response-related controlled WD -10	2.77 ± 0.12	2.45 ± 0.12	3.26 ± 0.11	1.96 ± 0.10	2.31 ± 0.12	2.47 ± 0.12	2.73 ± 0.11	3.12 ± 0.12
Response-related controlled WD 0	**2.87 ± 0.12**	2.97 ± 0.11	3.55 ± 0.09	**2.62 ± 0.11**	2.48 ± 0.10	2.88 ± 0.12	3.21 ± 0.09	**3.70 ± 0.10**
Response-related controlled WD 5	2.79 ± 0.10	2.83 ± 0.09	3.35 ± 0.12	2.40 ± 0.12	2.51 ± 0.13	2.80 ± 0.13	3.13 ± 0.12	3.41 ± 0.12
Response-related controlled WD 10	2.74 ± 0.11	2.42 ± 0.12	2.93 ± 0.11	1.95 ± 0.12	2.20 ± 0.12	2.56 ± 0.12	2.90 ± 0.12	3.12 ± 0.10
Response-related controlled WD 13	2.63 ± 0.12	2.06 ± 0.11	2.40 ± 0.09	1.74 ± 0.11	2.07 ± 0.11	2.25 ± 0.12	2.49 ± 0.14	2.63 ± 0.10

Table 8: Calibrated scores (mean ± std.) for all models and human evaluation metrics.

The first eight columns are Likert metrics on a 1-4 scale (except Avoiding Repetition, which is a 1-3 scale), where higher is better (except Inquisitiveness, which has an optimal score of 3). The last column, Persona Retrieval, is on a scale from 0 to 1 where higher is better.

The maximum of each column (excluding Human row) is in bold.

Rows marked with * are the six models included in Figure 3 (left) and Figure 4.

H Plots of human evaluation results for all configurations

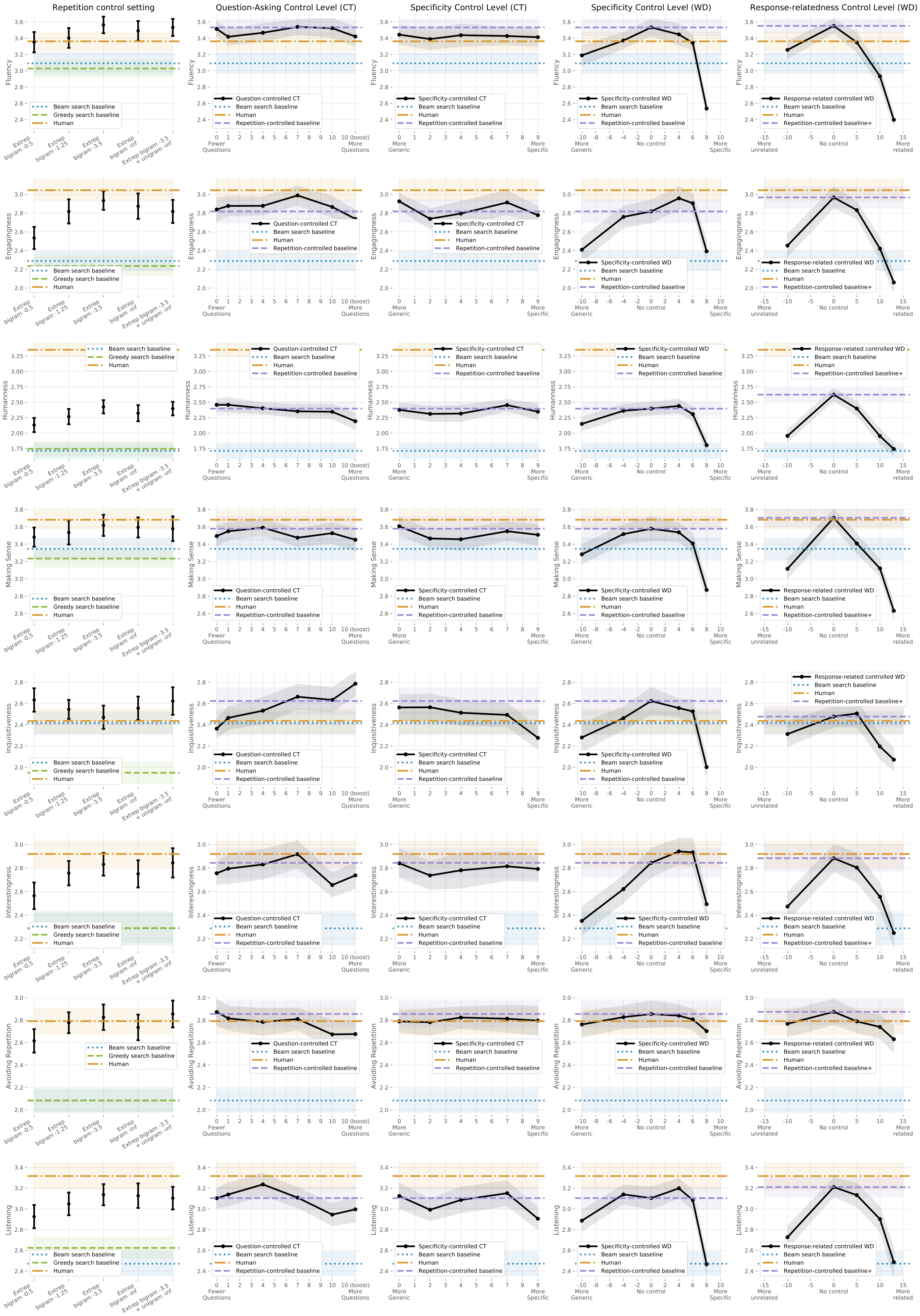


Figure 9: Calibrated human evaluation scores for all models. This is the same data as in Table 8.

Note: ‘Repetition-controlled baseline+’ in the rightmost column is ‘Response-related controlled WD 0’ in Table 8. See Table 5 for explanation.

An Empirical Investigation of Global and Local Normalization for Recurrent Neural Sequence Models Using a Continuous Relaxation to Beam Search

Kartik Goyal
School of Computer Science
Carnegie Mellon University
kartikgo@cs.cmu.edu

Chris Dyer
Deepmind
cdyer@google.com

Taylor Berg-Kirkpatrick
University of California, San Diego
tberg@ucsd.edu

Abstract

Globally normalized neural sequence models are considered superior to their locally normalized equivalents because they may ameliorate the effects of label bias. However, when considering high-capacity neural parametrizations that condition on the whole input sequence, both model classes are theoretically equivalent in terms of the distributions they are capable of representing. Thus, the practical advantage of global normalization in the context of modern neural methods remains unclear. In this paper, we attempt to shed light on this problem through an empirical study. We extend an approach for search-aware training via a continuous relaxation of beam search (Goyal et al., 2017b) in order to enable training of *globally normalized* recurrent sequence models through simple backpropagation. We then use this technique to conduct an empirical study of the interaction between global normalization, high-capacity encoders, and search-aware optimization. We observe that in the context of inexact search, globally normalized neural models are still more effective than their locally normalized counterparts. Further, since our training approach is sensitive to warm-starting with pre-trained models, we also propose a novel initialization strategy based on self-normalization for pre-training globally normalized models. We perform analysis of our approach on two tasks: CCG supertagging and Machine Translation, and demonstrate the importance of global normalization under different conditions while using search-aware training.

1 Introduction

Neural encoder-decoder models have been tremendously successful at a variety of NLP tasks, such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), parsing (Dyer et al., 2016, 2015), summarization (Rush et al., 2015), dialog generation (Serban et al., 2015), and image captioning (Xu et al., 2015). With these models, the target sequence is generated in a left-to-right step-wise manner with the predictions at every step being conditioned on the input sequence and the whole prediction history. This long-distance memory precludes exact search for the maximally scoring sequence according to the model and therefore, approximate algorithms like greedy search or beam search are necessary in practice during decoding. In this scenario, it is natural to resort to search-aware learning techniques for these models which makes the optimization objective sensitive to any potential errors that could occur due to inexact search in these models.

This work focuses on comparison between search-aware locally normalized sequence models that involve projecting the scores of items in the vocabulary onto a probability simplex at each step and globally normalized/unnormalized sequence models that involve scoring sequences without explicit normalization at each step. When conditioned on the the full input sequence and the entire prediction history, both locally normalized and globally normalized conditional models should have same expressive power under a high-capacity neural parametrization in theory, as they can both model same set of distributions over all finite length output sequences (Smith and Johnson, 2007). However, locally normalized models are constrained in how they respond to search errors during training since the scores at each decoding step must sum to one. To let a search-aware training setup have the most flexibility, abandoning this constraint may be useful for easier optimization.

In this paper, we demonstrate that the interaction between approximate inference and non-convex parameter optimization results in more robust training and better performance for models with global normalization compared to those with

Proceedings of NAACL-HLT 2019, pages 1724–1733
Minneapolis, Minnesota, June 2 - June 7, 2019.

the more common locally normalized parametrization. We posit that this difference is due to *label bias* (Bottou, 1991) arising from the interaction of approximate search and search-aware optimization in locally normalized models. A commonly understood source of label bias in *locally normalized* sequence models is an effect of conditioning only on partial input (for example, only the history of the input) at each step during decoding (Andor et al., 2016; Lafferty et al., 2001; Wiseman and Rush, 2016). We discus another potential source of label bias arising from approximate search with locally normalized models that may be present even with access to the full input at each step. To this end, we train search-aware globally and locally normalized models in an end-to-end (sub)-differentiable manner using a continuous relaxation to the discontinuous beam search procedure introduced by Goyal et al. (2017b). This approach requires initialization with a suitable globally normalized model to work in practice. Hence, we also propose an initialization strategy based upon self-normalization for pre-training globally normalized models.

We demonstrate the effect of both sources of label bias through our experiments on two common sequence tasks: CCG supertagging and machine translation. We find that label bias can be eliminated by both, using a powerful encoder, and using a globally normalized model. We observe that global normalization yields performance gains over local normalization and is able to ameliorate label bias especially in scenarios that involve a very large hypothesis space.

2 Recurrent Sequence Models and Effects of Normalization

We now introduce the notation that we will use in the remainder of the paper for describing locally and globally normalized neural sequence-to-sequence models. We are interested in the probability of output sequence, $\mathbf{y}$, conditioned on input sequence, $\mathbf{x}$. Let $s(\mathbf{x}, y_{1:i-1}, y_i)$ be a non negative score of output label y at time-step i for the input x and the prediction history $y_{1:i-1}$, let V be the label space, and let $\mathcal{Y}_{\mathbf{x}}$ be the space of all finite sequences for $\mathbf{x}$.[1] A neural encoder (e.g. a bidirectional LSTM) encodes information about $\mathbf{x}$ and a recurrent neural decoder generates the output $\mathbf{y}$ (typically step-by-step from left-to-right) conditioned on the encoder.

2.1 Locally normalized models

Under a locally normalized model $\mathcal{M}_L$, the probability of $\mathbf{y}$ given $\mathbf{x}$ is:

$$p_{\mathcal{M}_L}(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x}, y_{1:i-1}) = \prod_{i=1}^{n} \frac{s(\mathbf{x}, y_{1:i-1}, y_i)}{Z_{L,i}(\mathbf{x}, y_{1:i-1})}$$

where $Z_{L,i}(\mathbf{x}, y_{1:i-1}) = \sum_{y \in V} s(\mathbf{x}, y_{1:i-1}, y)$, is the local normalizer at each time step and n is the number of prediction steps. Since, the local normalizer is easy to compute, likelihood maximization based training is a standard approach for training these models.

2.2 Globally normalized models

In contrast, under a globally normalized model $\mathcal{M}_G$, the probability of $\mathbf{y}$ given $\mathbf{x}$ is:

$$p_{\mathcal{M}_G}(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{i=1}^{n} s(\mathbf{x}, y_{1:i-1}, y_i)}{Z_G(\mathbf{x})}$$

where $Z_G(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{i=1}^{n} s(\mathbf{x}, y_{1:i-1}, y_i)$, is the global log-normalizer. $Z_G(\mathbf{x})$ is intractable to estimate for most problems of interest due to the large search space therefore, an exact likelihood maximization training approach is intractable for these models.

2.3 Label Bias with partial input

It was shown in Andor et al. (2016); Lafferty et al. (2001), locally normalized conditional models *with access to only partial input*, $x_{1:i-1}$, at each decoding step are biased towards labeling decisions with low-entropy transition probabilities at each decoding step and, as a result, suffer from a weakened ability to revise previous decisions based upon future input observations. This phenomenon has been referred to as *label bias*, and presents itself as an arbitrary allocation of probability mass to unlikely or undesirable label sequences despite the presence of well-formed sequences in training data. Andor et al. (2016) prove that this class of locally normalized models that relies on the structural assumption of access to only left-to-right partial input at each step,

$$\prod_{i=1}^{n} p(y_i \mid \mathbf{x}, y_{1:i-1}) = \prod_{i=1}^{n} p(y_i \mid x_{1:i-1}, y_{1:i-1}),$$

---

[1] For notational convenience we suppress the dependence of the score s on model parameters θ.

is strictly less expressive than its globally normalized counterpart.

However, the standard sequence-to-sequence models used most often in practice and presented in this paper actually condition the decoder on a summary representation of the *entire input sequence*, $\mathbf{x}$, computed by a neural encoder. Hence, depending on the power of the encoder, it is commonly thought that such models avoid this type of label bias. For these models, both locally normalized and globally normalized conditional models are equally expressive, in principle, with a sufficiently powerful encoder.

However, as we suggest in the next section and show empirically in experiments, this does not necessarily mean that both parametrizations are equally amenable to gradient-based training in practice, particularly when the search space is large and search-aware training techniques are used. We will argue that they suffer from a related, but distinct, form of bias introduced by inexact decoding.

2.4 Search-aware training

To improve performance with inexact decoding methods (e.g. beam search), search-aware training techniques take into account the decoding procedure that will be used at test time and adjust the parameters of the model to maximize prediction accuracy under the decoder. Because of the popularity of beam search as a decoding procedure for sequence models, in this paper we focus on beam search-aware training. While many options are available, including beam-search optimization (BSO) (Wiseman and Rush, 2016), in Section 3.1 we will describe the particular search-aware training strategy we use in experiments (Goyal et al., 2017b), chosen for its simplicity.

2.5 Label Bias due to approximate search

We illustrate via example how optimization of locally normalized models may suffer from a new kind of label bias when using beam search-aware training, and point to reasons why this issue might be mitigated by the use of globally normalized models. While the scores of successors of a single candidate under a locally normalized model are constrained to sum to one, scores of successors under a globally normalized model need only be positive. Intuitively, during training, this gives the globally normalized model more freedom to downweight undesirable intermediate candidates in order avoid search errors.

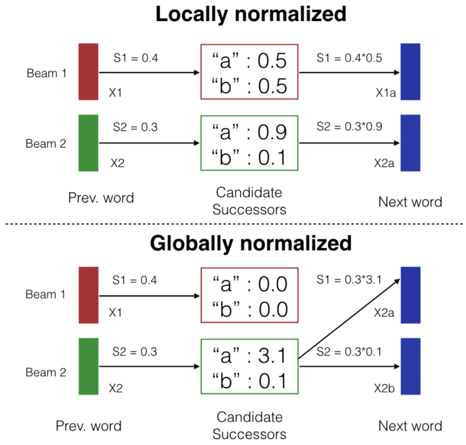


Figure 1: Illustrative example of bias arising in locally normalized models due to beam search. Red indicates the candidate that optimization should learn to discard and green indicates the candidate that should be propagated. Locally normalized models are constrained to return normalized scores for the successors of each candidate, while globally normalized models are unconstrained and can more easily learn to drop successors of the red candidate.

In the example beam search decoding problem in Figure 1, we compare the behavior of locally and globally normalized models at a single time step for a beam size of two. In this example, we assume that the score for beams in both the models is exactly the same until the step shown in Figure 1. Suppose that the lower item on the beam(X2) is correct, and thus, for more effective search, we would prefer the models scores to be such that only successors of the lower beam item are present on the beam at the next step. However, since, the scores at each step for a locally normalized model are constrained to sum to one, the upper beam item(X1) generates successors with scores comparable to those of the lower beam item. As we see in the example, due to the normalization constraint, search-aware training of the locally normalized model might find it difficult to set the parameters to prevent extension of the poorer candidate. In contrast, because the scores of a *globally normalized* model are not constrained to sum to one, the parameters of the neural model can be set such that *all* the successors of the bad candidate have a very low score and thus do not compete for space on the beam. This illustrates a mechanism by which search-aware training of globally normalized models in a large search spaces might

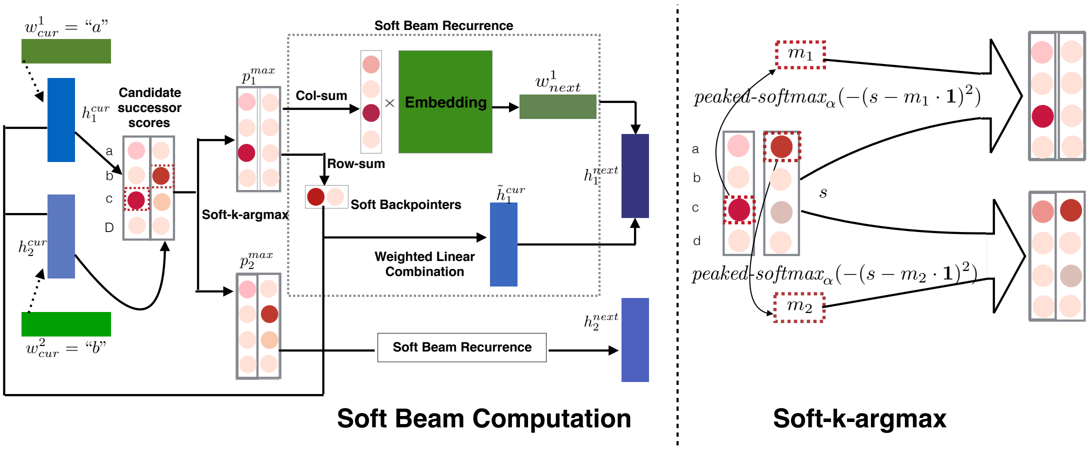


Figure 2: **Left:** Computing LSTM hidden states at a subsequent step using continuous relaxation to beam search for beam size of 2. **Right:** Illustration of top-k-argmax. m_1 and m_2 are the maximum and second maximum candidate scores.

be more effective. However as discussed earlier, if we can perform *exact search* then this *label bias* ceases to exist because both the models have the same expressive power with a search-agnostic optimization scheme. In experiments, we will explore this trade-off empirically.

3 Search-aware Training for Globally Normalized Models

In order to conduct an empirical study with meaningful comparisons, we devise an extension of the relaxed beam-search based optimization proposed by Goyal et al. (2017b) that allows us to train both the search-aware globally and locally normalized models in a similar manner with the same underlying architecture.

3.1 Continuous Relaxation to Beam Search

Following Goyal et al. (2017b), we train a beam-search aware model by optimizing a continuous surrogate approximation to a *direct loss* objective, J, defined as a function of the output of beam search and the ground truth sequence $\mathbf{y}^*$:

$$\min_\theta J(\mathbf{x}, \theta, \mathbf{y}^*) = \min_\theta \ell(\mathit{Beam}(\mathbf{x}, \mathcal{M}(\theta)), \mathbf{y}^*)$$

Here ℓ is a function that computes the loss of the model's prediction produced by beam search , and $\mathcal{M}$ refers to the model parametrized by θ. While this objective is search-aware, it is discontinuous and difficult to optimize because beam search involves discrete *k-argmax* operations. Therefore, Goyal et al. (2017b) propose a continuous surrogate, $\tilde{J}$, by defining a continuous approximation (*soft-k-argmax*) of the discrete *k-argmax* and using this to compute an approximation to a composition of the loss function and the beam search function.

$$\min_\theta \tilde{J}(\mathbf{x}, \theta, \mathbf{y}^*) \approx \min_\theta (\ell \circ \mathit{Beam})(\mathbf{x}, \mathcal{M}(\theta), \mathbf{y}^*)$$

The *soft-k-argmax* procedure involves computing distances between the scores of the successors and the k^{th}-max score and using the temperature based argmax operation (Maddison et al., 2017; Jang et al., 2016; Goyal et al., 2017a) to get an output peaked on the k^{th}-max value as shown in the right panel of Figure 2. The temperature is a hyperparameter which is typically annealed toward producing low entropy distributions during optimization. As shown in the left panel of Figure 2, the *soft candidate vectors* and the *soft backpointers* are computed at every decoding step using this *soft-k-argmax* operation in order to generate the embeddings and recurrent hidden states of the LSTM at each step of the *soft beam search* procedure. With a locally decomposable loss like Hamming loss, both *soft loss* and *soft scores* for the relaxed procedure are iteratively computed so that the end-to-end objective computation can be described by a computation graph that is amenable to backpropagation.

Using this relaxation, point-wise convergence of the surrogate objective to the original objective can be established (α is the inverse temperature):

$$\tilde{J}_\alpha(\mathbf{x}, \theta, \mathbf{y}^*) \xrightarrow[p]{\alpha \to \infty} J(\mathbf{x}, \theta, \mathbf{y}^*)$$

Goyal et al. (2017b) demonstrated empirically that optimizing the surrogate objective, $\tilde{J}$ – which can be accomplished via simple backpropagation for decomposable losses like Hamming distance – leads to improved performance at test time.

In experiments, for training locally normalized models, we use *log-normalized* successor scores. However, for training globally normalized models, we will directly use *unnormalized scores*, which are $\in \mathrm{R}_+$.

3.2 Initialization for training globally normalized models

Goyal et al. (2017b) reported that *initialization* with a locally normalized model pre-trained with teacher-forcing was important for their continuous beam search based approach to be stable and hence they used the locally normalized log-scores for their search-aware training model. In this work, we experimented with the unnormalized candidate successor scores and found that initializing the optimization for a globally normalized objective with a cross-entropy trained locally normalized model resulted in unstable training. This is expected because the locally normalized models are parametrized in a way such that using the scores before the softmax normalization results in a very different outcome than using scores after local normalization. For example, the locally normalized Machine Translation model in Table 1, that gives a BLEU score of 27.62 when decoded with beam search using locally normalized scores, results in BLEU of 4.30 when beam search decoding is performed with unnormalized scores. Pre-training a truly globally normalized model for initialization is not straghtforward because no exact likelihood maximization techniques exist for globally normalized models as the global normalizer is intractable to compute.

Therefore, we propose a new approach to initialization for search-aware training of globally normalized models: we pre-train a locally normalized model that is parametrized like a globally normalized model. More specifically, we train a locally normalized model with its distribution over the output sequences denoted by $p_L(\mathcal{Y})$ such that we can easily find a globally normalized model with a distribution $p_G(\mathcal{Y})$ that matches $p_L(\mathcal{Y})$. Following the notation in Section 2, for a locally normalized model, the log-probability of a sequence is:

$$\sum_{i=1}^{n} [\log s(\mathbf{x}, y_{1:i-1}, y_i) - \log Z_{L,i}(\mathbf{x}, y_{1:i-1})]$$

and for a globally normalized model it is:

$$\left[\sum_{i=1}^{n} \log s(\mathbf{x}, y_{1:i-1}, y_i)\right] - \log Z_G(\mathbf{x})$$

3.2.1 Self Normalization

One way to find a locally normalized model that is parametrized like a globally normalized model is to ensure that the local normalizer at each step, $\log\ Z_{L,i}(\mathbf{x}, y_{1:i-1})$, is 0. With the local normalizer being zero it is straightforward to see that the log probability of a sequence under a locally normalized model can easily be interpreted as log probability of the sequence under a globally normalized model with the global log-normalizer, $\log\ Z_G(\mathbf{x}) = 0$. This training technique is called *self-normalization* (Andreas and Klein, 2015) because the resulting models' unnormalized score at each step lies on a probability simplex. A common technique for training self-normalized models is L2-regularization of local log normalizer which encourages learning a model with $\log\ Z = 0$ and was found to be effective for learning a language model by Devlin et al. (2014)[2]. The L2-regularized cross entropy objective is given by:

$$\min_{\theta} \sum_{\mathbf{x}, \mathbf{y}^* \in \mathcal{D}} -\sum_{i=1}^{n} \log\ p(y_i^* \mid \mathbf{x}, y_{1:i-1}) \\ +\lambda \cdot (\log\ Z_{L,i}(\mathbf{x}, y_{1:i-1}))^2$$

In Table 1, we report the mean and variance of the local log normalizer on the two different tasks using L2-regularization (*L2*) based self normalization and no self normalization (*CE*). We observe that *L2* models are competitive performance-wise to the cross-entropy trained locally normalized models while resulting in a much smaller local log-normalizer on average. Although, we couldn't minimize $\log\ Z$ exactly to 0, we observe in Section 4 that this is sufficient to train a reasonable initializer for the search-aware optimization of globally normalized models. It is important to note that these approaches yield a *globally normalized* model that is equivalent to a locally normalized model trained via teacher-forcing and hence these are only used to *warm-start* the search-aware optimization of globally normalized models. Our search-aware training approach is

---

[2] Noise Contrastive Estimation (Mnih and Teh, 2012; Gutmann and Hyvärinen, 2010) is also an alternative to train unnormalized models but our experiments with NCE were unstable and resulted in worse models.

free to adjust the parameters of the models such that the final globally normalized model has a non-zero log-normalizer Z_G over the data.

		Train logZ		Dev logZ		Acc/BLEU
		Mean	Var	Mean	Var	
CCG	CE	21.08	9.57	21.96	9.18	93.3
	L2	0.6	0.29	0.26	0.08	91.9
MT	CE	24.7	115.4	25.8	129.1	27.62
	L2	0.65	0.18	0.7	0.29	26.63

Table 1: Comparison of logZ between cross entropy trained models (CE) and self normalized models (L2) for CCG supertagging and Machine Translation tasks.

Other possible approaches to project locally normalized models onto globally normalized models include distribution matching via knowledge distillation (Hinton et al., 2015). We leave exploration of warm-starting of search aware optimization with this approach to future work.

4 Experiments and Empirical Analysis

To empirically analyze the interaction between label bias arising from different sources, search-aware training, and global normalization, we conducted experiments on two tasks with vastly different sizes of output space: CCG supertagging and Machine Translation. As described in the next section, the task of tagging allows us to perform controlled experiments which explicitly study the effect of amount of input information available to the decoder at each step, we analyze the scenarios in which search aware training and global normalization are expected to improve the model performance.

In all our experiments, we report results on training with standard teacher forcing optimization and self-normalization as our baselines. We report results with both search-aware locally and globally normalized models (Section 3.1) after warm starting with both cross entropy trained models and self-normalized models to study the effects of search-aware optimization and global normalization. We follow Goyal et al. (2017b) and use the decomposable Hamming loss approximation with search-aware optimization for both the tasks and decode via *soft beam search decoding* method which involves continuous beam search with soft backpointers for the LSTM Beam search dynamics as described in Section 3, but using identifiable backpointers and labels (using MAP estimates of soft backpointers and labels) to decode.

We tune hyperparameters like learning rate and annealing schedule by observing performance on development sets for both the tasks. We performed at least three random restarts for each class and report results based on best development performance.

4.1 CCG supertagging

We used the standard splits of CCG bank (Hockenmaier and Steedman, 2002) for training, development, and testing. The label space of supertags is 1,284 and the labels are correlated with each other based on their syntactic relations. The distribution of supertag labels in the training data exhibits a long tail distribution. This task is sensitive to the long range sequential decisions because it encodes rich syntactic information about the sentence. Hence, this task is ideal to analyze the effects of label bias and search effects. We perform minor preprocessing on the data similar to the preprocessing in Vaswani et al. (2016). For experiments related to search aware optimization, we report results with beam size of 5.[3]

4.1.1 Tagging model for ablation study

We changed the standard sequence-to-sequence model to be more suitable for the tagging task. This change also lets us perform controlled experiments pertaining to the amount of input sequence information available to the decoder at each time step.

In a standard encoder-decoder model with attention, the initial hidden state of the decoder is often some function of the final encoder state so that the decoder's predictions can be conditioned on the full input. For our tagging experiments, instead of influencing the initial decoder state with the encoder, we set it to a vector of zeros. Thus the information about input for prediction is *only* available via the attention mechanism. In addition to the change above, we also forced the model to attend to only the i^{th} input representation while predicting the i^{th} label. This is enforceable because the output length is equal to the input length and it is also a more suitable structure for a tagging model. With these changes in the decoder, we can precisely control the amount of information about the input available to the decoder at each prediction step. For example, with a unidirectional LSTM encoder, the decoder at i^{th} step

---

[3] We observed similar results with beam size 10

only has access to input till the i^{th} token and the prediction history:

$$p(y_i \mid \mathbf{x}, y_{1:i-1}) = p(y_i \mid x_{1:i}, y_{1:i-1})$$

This setting lets us clearly explore the classical notion of *label bias* arising out of access to partial input at each prediction step (Section 2.3). A bidirectional LSTM encoder, however provides access to all of the input information to the decoder at all the prediction steps.

	Unidirectional	Bidirectional
pretrain-greedy	76.54	92.59
pretrain-beam	77.76	93.29
locally normalized	83.9	**93.76**
globally normalized	**83.93**	93.73

Table 2: **Accuracy results on CCG supertagging when initialized with a regular teacher-forcing model**. Reported using *Unidirectional* and *Bidirectional* encoders respectively with fixed attention tagging decoder. *pretrain-greedy* and *pretrain-beam* refer to the output of decoding the initializer model. *locally normalized* and *globally normalized* refer to search-aware soft-beam models

	Unidirectional	Bidirectional
pretrain-greedy	73.12	91.23
pretrain-beam	73.83	91.94
locally normalized	83.35	**92.78**
globally normalized	**85.50**	92.63

Table 3: **Accuracy results on CCG supertagging when initialized with a self normalized model**.

4.2 Machine Translation

We use the same dataset (the German-English portion of the IWSLT 2014 machine translation evaluation campaign (Cettolo et al., 2014)), preprocessing and data splits as Ranzato et al. (2016) for our Machine Translation experiments. The output label/vocabulary size is 32000 and unlike tagging, the length of output sequences cannot be deterministically determined from the length of the input sequence. Moreover, the output sequence does not necessarily align monotonically with the input sequence. Hence the output sequence space for MT is much larger than that for tagging and the effects of inexact search on optimization are expected to be even more apparent for MT. We use a standard LSTM-based encoder/decoder model with a standard attention mechanism (Bahdanau et al., 2016) for our MT experiments. For search-aware optimization experiments, we report results with beam size 3.[4]

Init-scheme →	Regular	Self-normalized
pretrain-greedy	26.24	25.42
pretrain-beam	27.62	26.63
locally-normalized	**29.28**	27.71
globally-normalized	26.24	**29.27**

Table 4: **BLEU results on de-en Machine Translation.** *Regular* and *Self-normalized* refer to the initization scheme for soft beam search training. *pretrain-greedy* and *pretrain-beam* refer to the output of decoding the initializer model. *locally normalized* and *globally normalized* refer to search-aware soft-beam models

4.3 Results and Analysis

The results reported in Tables 2, 3 and 4 allow us to analyze the effect of interaction of label bias, inexact search and global normalization in detail.

4.3.1 Label bias with partial input

First, we analyze the effect of label bias that arises from conditioning on partial input (Section 2.3) during decoding on optimization of the models. The unidirectional encoder based tagging experiments suggest that conditioning on partial input during decoding results in poor models when trained with cross entropy based methods. Interestingly, all techniques improve upon this: (i) search-aware locally and globally normalized models are able to train for accuracy directly and eliminate exposure bias that arises out of the mismatch between train-time and test-time prediction methods, and, (ii) the bidirectional tagging model which provides access to all of the input is powerful enough to learn a complex relationship between the decoder and the input representations for the search space of the CCG supertagging task and results in a much better performance.

4.3.2 Initialization of search-aware training

Next, we analyze the importance of appropriate initialization of search-aware optimization with pretrained models. Across all the results in Tables 2, 3 and 4, we observe that search-aware optimization for locally normalized models always improves upon the pre-trained locally

---

[4] We observed similar results beam size of 5.

normalized models used for initialization. But when the search-aware optimization for globally normalized models is initialized with locally normalized CE models, the improvement is not as pronounced and in the case of MT, the performance is actually *hurt* by the improper initialization for training globally normalized models – probably a consequence of large search space associated with MT and incompatibility between unnormalized scores for search-aware optimization and locally normalized scores of the *CE* model used for pre-training. When the *self-normalized* models are used for initialization, optimization for globally normalized models always improves upon the pre-trained self-normalized model. It is interesting to note that we see improvements for the globally normalized models even when $logZ$ is not exactly reduced to 0 indicating that the scores used for search-aware training initially are comparable to the scores of the pre-trained self-normalized model. We also observe that self-normalized models perform slightly worse than CE-trained models but search aware training for globally normalized models improves the performance significantly.

4.3.3 Search-aware training

Next, we analyze the effect of search-aware optimization on the performance of the models. Search-aware training with locally normalized models improves the performance significantly in *all* our experiments which indicates that accounting for exposure bias and optimizing for predictive performance directly is important. We also observe that the bidirectional model for tagging is quite powerful and seems to account for both *exposure bias* and *label bias* to a large extent. We reckon that this may be because the greedy decoding itself is very close to *exact search* for this well-trained tagging model over a search space that is much simpler than that associated with MT. Therefore, the impact of search-aware optimization on the bidirectional tagger is marginal. However, it is much more pronounced on the task of MT.

4.3.4 Global normalization and label bias

We analyze the importance of training globally normalized models. In the specific setup for tagging with the unidirectional encoder, globally normalized models are actually *more expressive* than the locally normalized models (Andor et al., 2016) as described in Section 2.3 and this is reflected in our experiments (table 3) with tagging. The globally normalized model (warm started with a self-normalized model) performs the best among all the models in the unidirectional tagger case which indicates that it is ameliorating something beyond exposure bias which is fixed by the search-aware locally normalized model.

For MT (table 4), both globally normalized and locally normalized models are equally expressive in theory because the decoder is conditioned on the full input information at each step, but we still observe that the globally normalized model improves significantly over the self-normalized pre-trained model and the search-aware locally normalized model. This indicates that it might be ameliorating the label bias associated with inexact search (discussed in Section 2.5). As discussed in Section 3.2, the globally normalized model, when initialized with a CE trained model, performs worse because of improper initialization of the search aware training. The self-normalized model starts off 1 BLEU point worse than the CE model point but global normalization, initialized with the self-normalized model improves the performance and is competitive with the best model for MT. This suggests that a better technique for initializing the optimization for globally normalized models should be helpful in improving the performance.

4.3.5 Global normalization and sentence length

In tables 5 and 6, we analyze the source of improvement from global normalization for MT. In table 5, we report the ngram overlap scores and ratio of length of the predictions to length of hypothesis for the case when the search-aware training is initialized with a self-normalized model. We observe that the globally normalized model produces longer predictions than the locally normalized model. More interestingly, it seems to have better 3 and 4-gram overlap and slightly worse unigram and bigram overlap score than the locally normalized model. These observations suggest that globally normalized models are better able to take longer range effects into account and are also cautious about predicting the *end-of-sentence* symbol too soon. Moreover, in table 6, we observe

	N-gram overlap	Length ratio
pretrain-beam	63.5/35.7/21.8/13.7	0.931
locally-normalized	66.9/39.4/22.7/14.0	0.918
globally-normalized	65.0/39.1/23.2/14.7	0.959

Table 5: **Breakdown of BLEU results on de-en Machine Translation dev set.** Reported on Self-normalized initialization

Src sent-length →	**0-20**	**20-30**	**30-40**	**40+**
pretrain-beam	29.36	25.73	24.71	24.50
locally-normalized	32.35	26.95	25.39	25.2
globally-normalized	33.21	28.08	26.75	26.41

Table 6: **BLEU scores with different length inputs on dev set** Reported on Self-normalized initialization. The header specifies the range of length of the input sentences

that globally normalized models perform better on all the length ranges but especially so on long sentences.

5 Related Work

Much of the existing work on search-aware training of globally normalized neural sequence models uses some mechanism like early updates (Collins and Roark, 2004) that relies on explicitly tracking if the gold sequence falls off the beam and is not end-to-end continuous. Andor et al. (2016) describe a method for training globally normalized neural feedforward models, which involves optimizing a CRF-based likelihood where the normalizer is approximated by the sum of the scores of the final beam elements. They describe label bias arising out of conditioning on partial input and hence focused on the scenario in which locally normalized models can be less expressive than globally normalized models, whereas we also consider another source of label bias which might be affecting the optimization of equally expressive locally and globally normalized conditional models. Wiseman and Rush (2016) also propose a beam search based training procedure that uses unnormalized scores similar to our approach. Their models achieve good performance over CE baselines – a pattern that we observe in our results as well. In this work, we attempt to empirically analyze the factors affecting this boost in performance with end-to-end continuous search-aware training (Goyal et al., 2017b) for globally normalized models.

Smith and Johnson (2007) proved that locally normalized conditional PCFGs and unnormalized conditional WCFGs are equally expressive for finite length sequences and posit that Maximum Entropy Markov Models (MEMMs) are weaker than CRFs because of the structural assumptions involved with MEMMs that result in label bias.

Recently, energy based neural structured prediction models (Amos et al., 2016; Belanger and McCallum, 2016; Belanger et al., 2017) were proposed that define an energy function over candidate structured output space and use gradient based optimization to form predictions making the overall optimization search aware. These models are designed to model global interactions between the output random variables without specifying strong structural assumptions.

6 Conclusion

We performed empirical analysis to analyze the interaction between label bias, search-aware optimization and global normalization in various scenarios. We proposed an extension to the continuous relaxation to beam search proposed by Goyal et al. (2017b) to train search-aware globally normalized models and comparable locally normalized models. We find that in the context of inexact search over large output spaces, globally normalized models are more effective than the locally normalized models in spite of them being equivalent in terms of their expressive power.

Acknowledgement

This project is funded in part by the NSF under grant 1618044. We thank the three anonymous reviewers for their helpful feedback.

References

Brandon Amos, Lei Xu, and J Zico Kolter. 2016. Input convex neural networks. *arXiv preprint arXiv:1609.07152*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Association for Computational Linguistics*.

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Dzmitry Bahdanau, Dmiriy Serdyuk, Philémon Brakel, Nan Rosemary Ke, Jan Chorowski, Aaron Courville, and Yoshua Bengio. 2016. Task loss estimation for structured prediction.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. *arXiv preprint arXiv:1703.05667*.

Léon Bottou. 1991. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. Ph.D. thesis, Université de Paris XI, Orsay, France.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1370–1380.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017a. Differentiable scheduled sampling for credit assignment. *arXiv preprint arXiv:1704.06970*.

Kartik Goyal, Graham Neubig, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017b. A continuous relaxation of beam search for end-to-end training of neural sequence models. *arXiv preprint arXiv:1708.00111*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *LREC*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Noah A Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proceedings of NAACL-HLT*, pages 232–237.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Pun Generation with Surprise

He He[1*] and **Nanyun Peng**[2*] and **Percy Liang**[1]
[1]Computer Science Department, Stanford University
[2]Information Sciences Institute, University of Southern California
{hehe,pliang}@cs.stanford.edu, npeng@isi.edu

Abstract

We tackle the problem of generating a pun sentence given a pair of homophones (e.g., "*died*" and "*dyed*"). Supervised text generation is inappropriate due to the lack of a large corpus of puns, and even if such a corpus existed, mimicry is at odds with generating novel content. In this paper, we propose an unsupervised approach to pun generation using a corpus of unhumorous text and what we call the *local-global surprisal principle*: we posit that in a pun sentence, there is a strong association between the pun word (e.g., "*dyed*") and the distant context, as well as a strong association between the alternative word (e.g., "*died*") and the immediate context. This contrast creates surprise and thus humor. We instantiate this principle for pun generation in two ways: (i) as a measure based on the ratio of probabilities under a language model, and (ii) a retrieve-and-edit approach based on words suggested by a skip-gram model. Human evaluation shows that our retrieve-and-edit approach generates puns successfully 31% of the time, tripling the success rate of a neural generation baseline.

1 Introduction

Generating *creative* content is a key requirement in many natural language generation tasks such as poetry generation (Manurung et al., 2000; Ghazvininejad et al., 2016), story generation (Meehan, 1977; Peng et al., 2018; Fan et al., 2018; Yao et al., 2019), and social chatbots (Weizenbaum, 1966; Hao et al., 2018). In this paper, we explore creative generation with a focus on puns. We follow the definition of puns in Aarons (2017); Miller et al. (2017): "A pun is a form of wordplay in which one sign (e.g., a word or a phrase) suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another sign, for an intended humorous or rhetorical effect." We focus on a typical class of puns where the ambiguity comes from two (near) homophones. Consider the example in Figure 1: "*Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've dyed (died) a little inside.*". The *pun word* shown in the sentence ("*dyed*") indicates one interpretation: the person is colored inside by food coloring. On the other hand, an *alternative word* ("*died*") is implied by the context for another interpretation: the person is sad due to the accident.

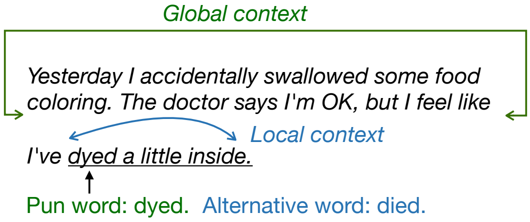


Figure 1: An illustration of a homophonic pun. The *pun word* appears in the sentence, while the *alternative word*, which has the same pronunciation but different meaning, is implicated. The *local context* refers to the immediate words around the pun word, whereas the *global context* refers to the whole sentence.

Current approaches to text generation require lots of training data, but there is no large corpus of puns. Even such a corpus existed, learning the distribution of existing data and sampling from it is unlikely to lead to truly novel, creative sentences. Creative composition requires deviating from the norm, whereas standard generation approaches seek to mimic the norm.

Recently, Yu et al. (2018) proposed an unsupervised approach that generates puns from a neural language model by jointly decoding conditioned on both the pun and the alternative words, thus injecting ambiguity to the output sentence. However, Kao et al. (2015) showed that ambiguity alone is insufficient to bring humor; the two mean-

---

[*] Equal contribution.

Proceedings of NAACL-HLT 2019, pages 1734–1744
Minneapolis, Minnesota, June 2 - June 7, 2019.

ings must also be supported by distinct sets of words in the sentence.

Inspired by Kao et al. (2015), we propose a general principle for puns which we call *local-global surprisal principle*. Our key observation is that the strength for the interpretation of the pun and the alternative words flips as one reads the sentence. For example, in Figure 1, "*died*" is favored by the immediate (local) context, whereas "*dyed*" is favored by the global context (i.e. "*...food coloring...*"). Our surprisal principle posits that the pun word is much more surprising in the local context than in the global context, while the opposite is true for the alternative word.

We instantiate our local-global surprisal principle in two ways. First, we develop a quantitative metric for surprise based on the conditional probabilities of the pun word and the alternative word given local and global contexts under a neural language model. However, we find that this metric is not sufficient for generation. We then develop an unsupervised approach to generate puns based on a retrieve-and-edit framework (Guu et al., 2018; Hashimoto et al., 2018) given an unhumorous corpus (Figure 2). We call our system SURGEN (SURprisal-based pun GENeration).

We test our approach on 150 pun-alternative word pairs.[1] First, we show a strong correlation between our surprisal metric and funniness ratings from crowdworkers. Second, human evaluation shows that our system generates puns successfully 31% of the time, compared to 9% of a neural generation baseline (Yu et al., 2018), and results in higher funniness scores.

2 Problem Statement

We assume access to a large corpus of raw (unhumorous) text. Given a pun word w^{p} (e.g., "*dyed*") and an alternative word w^{a} (e.g., "*died*") which are (near) homophones, we aim to generate a list of pun sentences. A pun sentence contains only the pun word w^{p}, but both w^{p} and w^{a} should be evoked by the sentence.

3 Approach

3.1 Surprise in Puns

What makes a good pun sentence? Our key observation is that as a reader processes a sentence, he or she expects to see the alternative word at the pun word position, and are tickled by the relation between the pun word and the rest of the sentence. Consider the following cloze test: "*Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've _____ a little inside.*". Most people would expect the word in the blank to be "*died*" whereas the actual word is "*dyed*". Locally, "*died a little inside*" is much more likely than "*dyed a little inside*". However, globally when looking back at the whole sentence, "*dyed*" is evoked by "*food coloring*".

Formally, w^{p} is more surprising relative to w^{a} in the local context, but much less so in the global context. We hypothesize that this contrast between local and global surprisal creates humor.

3.2 A Local-Global Surprisal Measure

Let us try to formalize the local-global surprisal principle quantitatively. To measure the amount of surprise due to seeing the pun word instead of the alternative word in a certain context c, we define surprisal S as the log-likelihood ratio of the two events:

$$S(c) \stackrel{\text{def}}{=} -\log \frac{p(w^{\text{p}} \mid c)}{p(w^{\text{a}} \mid c)} = -\log \frac{p(w^{\text{p}}, c)}{p(w^{\text{a}}, c)}. \quad (1)$$

We define the local surprisal to only consider context of a span around the pun word, and the global surprisal to consider context of the whole sentence. Letting $x_1, \ldots, x_n$ be a sequence of tokens, and x_p be the pun word w^{p}, we have

$$S_{\text{local}} \stackrel{\text{def}}{=} S(x_{p-d:p-1}, x_{p+1:p+d}), \quad (2)$$

$$S_{\text{global}} \stackrel{\text{def}}{=} S(x_{1:p-1}, x_{p+1:n}), \quad (3)$$

where d is the local window size.

For puns, both the local and global surprisal should be positive because they are unusual sentences by nature. However, the global surprisal should be lower than the local surprisal due to topic words hinting at the pun word. We use the following unified metric, *local-global surprisal*, to quantify whether a sentence is a pun:

$$S_{\text{ratio}} \stackrel{\text{def}}{=} \begin{cases} -1 & S_{\text{local}} < 0 \text{ or } S_{\text{global}} < 0, \\ S_{\text{local}}/S_{\text{global}} & \text{otherwise.} \end{cases} \quad (4)$$

We hypothesize that larger S_{ratio} is indicative of a good pun. Note that this hypothesis is invalid when either S_{local} or S_{global} is negative, in which case we consider the sentences equally unfunny by setting S_{ratio} to -1.

---

[1] Our code and data are available at `https://github.com/hhexiy/pungen`.

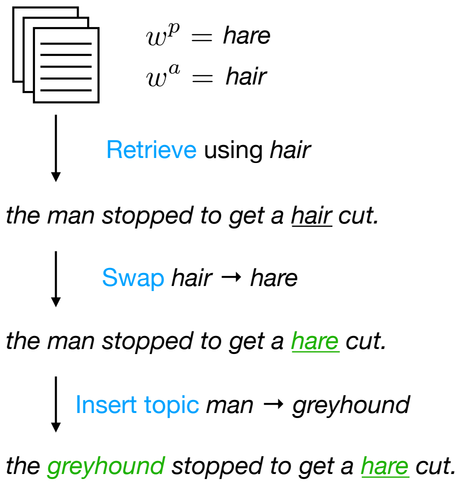


Figure 2: Overview of our pun generation approach. Given a pair of pun/alternative word, we first retrieve sentences containing w^{a} from a generic corpus. Next, w^{a} is replaced by w^{p} to increase local surprisal. Lastly, we insert a topic word at the beginning of the sentence to create global associations supporting w^{p} and decrease global surprisal.

3.3 Generating Puns

The surprisal metric above can be used to assess whether a sentence is a pun, but to generate puns, we need a procedure that can ensure grammaticality. Recall that the surprisal principle requires (1) a strong association between the alternative word and the local context; (2) a strong association between the pun word and the distant context; and (3) both words should be interpretable given local and global context to maintain ambiguity.

Our strategy is to model puns as deviations from normality. Specifically, we mine *seed sentences* (sentences with the potential to be transformed into puns) from a large, generic corpus, and edit them to satisfy the three requirements above.

Figure 2 gives an overview of our approach. Suppose we are generating a pun given $w^{\mathrm{p}} =$ "*hare*" and $w^{\mathrm{a}} =$ "*hair*". To reinforce $w^{\mathrm{a}} =$ "*hair*" in the local context despite the appearance of "*hare*", we retrieve sentences containing "*hair*" and replace occurrences of it with "*hare*". Here, the local context strongly favors the alternative word ("*hair cut*") relative to the pun word ("*hare cut*"). Next, to make the pun word "*hare*" more plausible, we insert a "*hare*"-related topic word ("*greyhound*") near the beginning of the sentence. In summary, we create local surprisal by putting w^{p} in common contexts for w^{a}, and connect w^{p} to a distant topic word by substitution. We describe each step in detail below.

Local surprisal. The first step is to retrieve sentences containing w^{a}. A typical pattern of pun sentences is that the pun word only occurs once towards the end of the sentence, which separates local context from pun-related topics at the beginning. Therefore, we retrieve sentences containing exactly one w^{a} and rank them by the position of w^{a} in the sentence (later is better). Next, we replace w^{a} in the retrieved sentence with w^{p}. The pun word usually fits in the context as it often has the same part-of-speech tag as the alternative word. Thus the swap creates local surprisal by putting the pun word in an unusual but acceptable context. We call this step RETRIEVE+SWAP, and use it as a baseline to generate puns.

Global surprisal. While the pun word is locally unexpected, we need to foreshadow it. This global association must not be too strong that it eliminates the ambiguity. Therefore, we include a single *topic word* related to the pun word by replacing one word at the beginning of the seed sentence. We see this simple structure in many human-written puns as well. For example, "*Old butchers never die, they only meat their fate.*", where pun words and their corresponding topic words are underlined.

We define relatedness between two words w_i and w_j based on a "distant" skip-gram model $p_\theta(w_j \mid w_i)$, where we train p_θ to maximize $p_\theta(w_j \mid w_i)$ for all w_i, w_j in the same sentence between d_1 to d_2 words apart. Formally:

$$\sum_{j=i-d_1}^{i-d_2} \log p_\theta(w_j \mid w_i) + \sum_{j=i+d_1}^{i+d_2} \log p_\theta(w_j \mid w_i). \tag{5}$$

We take the top-k predictions from $p_\theta(w \mid w^{\mathrm{p}})$, where w^{p} is the pun word, as candidate topic words w to be further filtered next.

Type consistent constraint. The replacement must maintain acceptability of the sentence. For example, changing "*person*" to "*ship*" in "*Each person must pay their fare share*" does not make sense even though "*ship*" and "*fare*" are related. Therefore, we restrict the deleted word in the seed sentence to nouns and pronouns, as verbs have more constraints on their arguments and replacing them is likely to result in unacceptable sentences.

In addition, we select candidate topic words that are type-consistent with the deleted word, e.g., replacing "*person*" with "*passenger*" as opposed to "*ship*". We define type-consistency (for nouns) based on WordNet path similarity.[2] Given two words, we get their synsets from WordNet constrained by their POS tags.[3] If the path similarity between any pair of senses from the two respective synsets is larger than a threshold, we consider the two words type-consistent. In summary, the first noun or pronoun in the seed sentence is replaced by a type-consistent topic word. We call this baseline RETRIEVE+SWAP+TOPIC.

Improve grammaticality. Directly replacing a word with the topic word may result in ungrammatical sentences, e.g., replacing "*i*" with "*negotiator*" and getting "*negotiator am just a woman trying to peace her life back together.*". Therefore, we use a sequence-to-sequence model to smooth the edited sentence (RETRIEVE+SWAP+TOPIC+SMOOTHER).

We smooth the sentence by deleting words around the topic word and train a model to fill in the blank. The smoother is trained in a similar fashion to denoising autoencoders: we delete immediate neighbors of a word in a sentence, and ask the model to reconstruct the sentence by predicting missing neighbors. A training example is shown below:

Original:	`the man slowly walked towards the woods .`
Input:	`<i> man </i> walked towards the woods .`
Output:	`the man slowly`

During training, the word to delete is selected in the same way as selecting the word to replace in a seed sentence, i.e. nouns or pronouns at the beginning of a sentence. At test time, the smoother is expected to fill in words to connect the topic word with the seed sentence in a grammatical way, e.g., "*the negotiator is just a woman trying to peace her life back together.*" (the part rewritten by the smoother is underlined).

4 Experiments

We first evaluate how well our surprisal principle predicts the funniness of sentences perceived by humans (Section 4.2), and then compare our pun generation system and its variations with a simple retrieval baseline and a neural generation model (Yu et al., 2018) (Section 4.3). We show that the local-global surprisal scores strongly correlate with human ratings of funniness, and all of our systems outperform the baselines based on human evaluation. In particular, RETRIEVE+SWAP+TOPIC (henceforth SURGEN) achieves the highest success rate and average funniness score among all systems.

4.1 Datasets

We use the pun dataset from 2017 SemEval task7 (Doogan et al., 2017). The dataset contains 1099 human-written puns annotated with pun words and alternative words, from which we take 219 for development. We use BookCorpus (Zhu et al., 2015) as the generic corpus for retrieval and training various components of our system.

4.2 Analysis of the Surprisal Principle

We evaluate the surprisal principle by analyzing how well the local-global surprisal score (Equation (4)) predicts funniness rated by humans. We first give a brief overview of previous computational accounts of humor, and then analyze the correlation between each metric and human ratings.

Prior funniness metrics. Kao et al. (2015) proposed two information-theoretic metrics: *ambiguity* of meanings and *distinctiveness* of supporting words. Ambiguity says that the sentence should support both the pun meaning and the alternative meaning. Distinctiveness further requires that the two meanings be supported by distinct sets of words.

In contrast, our metric based on the surprisal principle imposes additional requirements. First, surprisal says that while both meanings are acceptable (indicating ambiguity), the pun meaning is unexpected based on the local context. Second, the local-global surprisal contrast requires the pun word to be well supported in the global context.

Given the anomalous nature of puns, we also consider a metric for *unusualness* based on normalized log-probabilities under a language model (Pauls and Klein, 2012):

$$\text{Unusualness} \overset{\text{def}}{=} -\frac{1}{n}\log\left(p(x_1,\dots,x_n)/\prod_{i=1}^{n}p(x_i)\right). \tag{6}$$

Implementation details. Both ambiguity and distinctiveness are based on a generative model

---

[2] Path similarity is a score between 0 and 1 that is inversely proportional to the shortest distance between two word senses in WordNet.

[3] Pronouns are mapped to the synset `person.n.01`.

Type	Example	SEMEVAL		KAO	
		Count	Funniness	Count	Funniness
Pun	Yesterday a cow saved my life—it was bovine intervention.	33	1.13	141	1.09
Swap-pun	Yesterday a cow saved my life—it was divine intervention.	33	0.05	0	—
Non-pun	The workers are all involved in studying the spread of bovine TB.	64	-0.34	257	-0.53

Table 1: Dataset statistics and funniness ratings of SEMEVAL and KAO. Pun or alternative words are underlined in the example sentence. Each worker's ratings are standardized to z-scores. There is clear separation among the three types in terms of funniness, where pun $>$ swap-pun $>$ non-pun.

Metric	Pun and non-pun				Pun and swap-pun		Pun			
	SEMEVAL		KAO		SEMEVAL		SEMEVAL		KAO	
Surprisal (S_{ratio})	**0.46**	p=0.00	**0.58**	p=0.00	**0.48**	p=0.00	0.26	p=0.15	0.08	p=0.37
Ambiguity	**0.40**	p=0.00	**0.59**	p=0.00	0.18	p=0.15	0.00	p=0.98	0.00	p=0.95
Distinctiveness	-0.17	p=0.10	**0.29**	p=0.00	0.15	p=0.24	**0.41**	p=0.02	**0.27**	p=0.00
Unusualness	**0.37**	p=0.00	**0.36**	p=0.00	0.19	p=0.12	0.20	p=0.27	0.11	p=0.18

Table 2: Spearman correlation between different metrics and human ratings of funniness. Statistically significant correlations with p-value < 0.05 are bolded. Our surprisal principle successfully differentiates puns from non-puns and swap-puns. Distinctiveness is the only metric that correlates strongly with human ratings within puns. However, no single metric works well across different types of sentences.

of puns. Each sentence has a latent variable $z \in \{w^{\text{p}}, w^{\text{a}}\}$ corresponding to the pun meaning and the alternative meaning. Each word also has a latent meaning assignment variable f controlling whether it is generated from an unconditional unigram language model or a unigram model conditioned on z. Ambiguity is defined as the entropy of the posterior distribution over z given all the words, and distinctiveness is defined as the symmetrized KL-divergence between distributions of the assignment variables given the pun meaning and the alternative meaning respectively. The generative model relies on $p(x_i \mid z)$, which Kao et al. (2015) estimates using human ratings of word relatedness. We instead use the skip-gram model described in Section 3.3 as we are interested in a fully-automated system.

For local-global surprisal and unusualness, we estimate probabilities of text spans using a neural language model trained on WikiText-103 (Merity et al., 2016).[4] The local context window size (d in Equation (2)) is set to 2.

Human ratings of funniness. Similar to Kao et al. (2015), to test whether a metric can differentiate puns from normal sentences, we collected ratings for both puns from the SemEval dataset and *non-puns* retrieved from the generic corpus containing either w^{p} or w^{a}. To test the importance of surprisal, we also included *swap-puns* where w^{p} is replaced by w^{a}, which results in sentences that are ambiguous but not necessarily surprising.

We collected all of our human ratings on Amazon Mechanical Turk (AMT). Workers are asked to answer the question "*How funny is this sentence?*" on a scale from 1 (not at all) to 7 (extremely). We obtained funniness ratings on 130 sentences from the development set with 33 puns, 33 swap-puns, and 64 non-puns. 48 workers each read roughly 10–20 sentences in random order, counterbalanced for sentence types of non-puns, swap-puns, and puns. Each sentence is rated by 5 workers, and we removed 10 workers whose maximum Spearman correlation with other people rating the same sentence is lower than 0.2. The average Spearman correlation among all the remaining workers (which captures inter-annotator agreement) is 0.3. We z-scored the ratings of each worker for calibration and took the average z-scored ratings of a sentence as its funniness score.

Table 1 shows the statistics of our annotated dataset (SEMEVAL) and Kao et al. (2015)'s dataset (KAO). Note that the two datasets have different numbers and types of sentences, and the human ratings were collected separately. As expected, puns are funnier than both swap-puns and non-puns. Swap-puns are funnier than non-puns, possibly because they have inherit ambiguity brought by the RETRIEVE+SWAP operation.

---

[4] `https://dl.fbaipublicfiles.com/fairseq/models/wiki103_fconv_lm.tar.bz2`.

Automatic metrics of funniness. We analyze the following metrics: local-global surprisal (S_{ratio}), ambiguity, distinctiveness, and unusualness, with respect to their correlation with human ratings of funniness. For each metric, we standardized the scores and outliers beyond two standard deviations are set to $+2$ or -2 accordingly.[5] We then compute the metrics' Spearman correlation with human ratings. On KAO, we directly took the ambiguity scores and distinctiveness scores from the original implementation which requires human-annotated word relatedness.[6] On SEMEVAL, we used our reimplemention of Kao et al. (2015)'s algorithm but with the skip-gram model.

The results are shown in Table 2. For puns and non-puns, all metrics correlate strongly with human scores, indicating all of them are useful for pun detection. For puns and swap-puns, only local-global surprisal (S_{ratio}) has strong correlation, which shows that surprisal is important for characterizing puns. Ambiguity and distinctiveness do not differentiate pun word from the alternative word, and unusualness only considers probability of the sentence with the pun word, thus they do not correlate as significantly as S_{ratio}.

Within puns, only distinctiveness has significant correlation, whereas the other metrics are not fine-grained enough to differentiate good puns from mediocre ones. Overall, no single metric is robust enough to score funniness across all types of sentences, which makes it hard to generate puns by optimizing automatic metrics of funniness directly.

There is slight inconsistency between results on SEMEVAL and KAO. Specifically, for puns and non-puns, the distinctiveness metric shows a significant correlation with human ratings on KAO but not on SEMEVAL. We hypothesize that it is mainly due to differences in the two corpora and noise from the skip-gram approximation. For example, our dataset contains longer sentences with an average length of 20 words versus 11 words for KAO. Further, Kao et al. (2015) used human annotation of word relatedness while we used the skip-gram model to estimate $p(x_i \mid z)$.

Method	Success	Funniness	Grammar
NJD	9.2%	1.4	2.6
R	4.6%	1.3	**3.9**
R+S	27.0%	1.6	3.5
R+S+T+M	28.8%	**1.7**	2.9
SURGEN	**31.4%**	**1.7**	3.0
Human	78.9%	3.0	3.8

Table 3: Human evaluation results of all systems. We show average scores of funniness and grammaticality on a 1-5 scale and success rate computed from yes/no responses. We compare with two baselines: NEURALJOINTDECODER (NJD) and RETRIEVE (R). R+S, SURGEN, and R+S+T+M are three variations of our method: RETRIEVE+SWAP, RETRIEVE+SWAP+TOPIC, and RETRIEVE+SWAP+TOPIC+SMOOTHER, respectively. Overall, SURGEN performs the best across the board.

4.3 Pun Generation Results

Systems. We compare with a recent neural pun generator (Yu et al., 2018). They proposed an unsupervised approach based on generic language models to generate homographic puns.[7] Their approach takes as input two senses of a target word (e.g., *bat.n01*, *bat.n02* from WordNet synsets), and decodes from both senses jointly by taking a product of the probabilities conditioned on the two senses respectively (e.g., *bat.n01* and *bat.n02*), so that both senses are reflected in the output. To ensure that the target word appears in the middle of a sentence, they decode backward from the target word towards the beginning and then decode forward to complete the sentence. We adapted their method to generate homophonic puns by considering w^{p} and w^{a} as two input senses and decoding from the pun word. We retrained their forward / backward language models on the same BookCorpus used for our system. For comparison, we chose their best model (NEURALJOINTDECODER), which mainly captures ambiguity in puns.

In addition, we include a retrieval baseline (RETRIEVE) which simply retrieves sentences containing the pun word.

For our systems, we include the entire progression of methods described in Section 3 (RETRIEVE+SWAP, RETRIEVE+SWAP+TOPIC, and RETRIEVE+SWAP+TOPIC+SMOOTHER).

Implementation details. The key components of our systems include a retriever, a skip-gram

---

[5] Since both S_{ratio} and distinctiveness are unbounded, bounding the values gives more reliable correlation results.

[6] `https://github.com/amoudgl/pun-model`

[7] Sentences where the pun word and alternative word have the same written form (e.g., *bat*) but different senses.

Aspect	SurGen v.s. NJD win %	lose %	SurGen v.s. Human win %	lose %
Success	48.0	5.3	6.0	78.7
Funniness	56.7	25.3	10.7	85.3
Grammar	60.7	30.0	8.0	82.0

Table 4: Pairwise comparison between SurGen and NeuralJointDecoder (NJD), and between SurGen and human written puns. Win % (lose %) is the percentage among the human-rated 150 sentences where SurGen achieves a higher (lower) average score compared to the other method. The rest are ties.

model for topic word prediction, a type consistency checker, and a neural smoother. Given an alternative word, the retriever returned 500 candidates, among which we took the top 100 as seed sentences (Section 3.3 local surprisal). For topic words, we took the top 100 words predicted by the skip-gram model and filtered them to ensure type consistency with the deleted word (Section 3.3 global surprisal). The WordNet path similarity threshold for type consistency was set to 0.3.

The skip-gram model was trained on BookCorpus with $d_1 = 5$ and $d_2 = 10$ in Equation (5). We set the word embedding size to 300 and trained for 15 epochs using Adam (Kingma and Ba, 2014) with a learning rate of 0.0001. For the neural smoother, we trained a single-layer LSTM (512 hidden units) sequence-to-sequence model with attention on BookCorpus. The model was trained for 50 epochs using AdaGrad (Duchi et al., 2010) with a learning rate of 0.01 and a dropout rate of 0.1.

Human evaluation. We hired workers on AMT to rate outputs from all 5 systems together with expert-written puns from the SemEval pun dataset. Each worker was shown a group of sentences generated by all systems (randomly shuffled) given the same pun word and alternative word pair. Workers were asked to rate each sentence on three aspects: (1) success ("*Is the sentence a pun?*"),[8] (2) funniness ("*How funny is the sentence?*"), and (3) grammaticality ("*How grammatical is the sentence?*"). Success was rated as yes/no, and funniness and grammaticality were rated on a scale from 1 (not at all) to 5 (very). We also included a N/A choice (does not make sense) for funniness to exclude cases where the sentence are not understandable. Workers were explicitly instructed to try their best to give different scores for sentences in the same group.

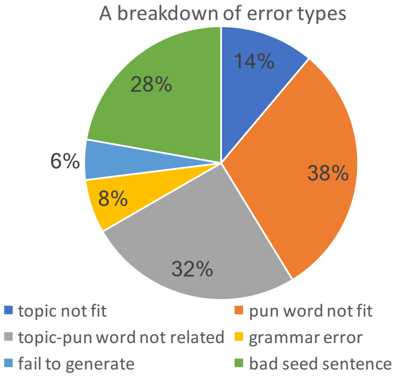


Figure 3: Error case breakdown shows that the main issues lie in finding seed sentences that accommodates both the pun word and the topic word (topic not fit + pun word not fit + bad seed sentence).

We evaluated 150 pun/alternative word pairs. Each generated sentence was rated by 5 workers and their scores were averaged. N/A ratings were excluded unless all ratings of a sentence were N/A, in which case we set its score to 0. We attracted 65, 93, 66 workers for the success, funniness, and grammaticality surveys respectively, and removed 3, 4, 4 workers because their maximum Spearman correlation with other workers was lower than 0.2. We measure inter-annotator agreement using average Spearman correlation among all workers, and the average inter-annotator Spearman correlation for success, funniness, and grammaticality are 0.57, 0.36, and 0.32, respectively.

Table 3 shows the overall results. All 3 of our systems outperform the baselines in terms of success rate and funniness. More edits (i.e. swapping, inserting topic words) made the sentence less grammatical, but also much more like puns (higher success rate). Interestingly, introducing the neural smoother did not improve grammaticality and hurt success rate slightly. Manual inspection shows that ungrammaticality is often caused by improper topic word, thus fixing its neighboring words does not truly solve the problem. For example, filling "*drum*" (related to "*lute*") in "*if that was _____ it was likely that another body would turn up soon, because someone probably wouldn't want to share the lute.*". In addition, when the neural model is given a rare topic word, it tends to rewrite it to a common phrase instead, again showing that supervised learning is against the spirit of generating novel content. For example, inserting "*gentlewoman*" to "_____ *not allow me to ...*" produces "*these people did not allow me to ...*". Overall, our

---

[8] They were shown the definition from Miller et al. (2017).

Method	Example	Rating
1. Pun/alternative word pair: butter – better		
NJD	He is going to come up with the butter a 'very good' approach to the world's economic crisis, the world's biggest economic climate.	1
SurGen	Well, gourmet did it, he thought, it'd butter be right.	2
Human	Why did the dairy churn? The less said, the butter...	1.5
2. Pun/alternative word pair: peace – piece		
NJD	Further, he said, at the end of the peace, it's not clear that it will be a good example.	1
SurGen	That's because negotiator got my car back to me in one peace.	1.5
Human	Life is a puzzle; look here for the missing peace.	3
3. Pun/alternative word pair: flour – flower		
NJD	Go, and if you are going on the flour.	1
SurGen	Butter want to know who these two girls are, the new members of the holy flour.	1.5
Human	Betty crocker was a flour child.	4.5
4. Pun/alternative word pair: wait – weight		
NJD	Gordon Brown, Georgia's prime minister, said he did not have to wait, but he was not sure whether he had been killed.	0
SurGen	Even from the outside, I could tell that he'd already lost some wait.	2
Human	Patience is a virtue heavy in wait.	3

Table 5: Examples of generated puns with average human ratings of funniness (1-5). 0 means that all ratings are N/A (does not make sense).

SurGen performs the best and tripled the success rate of NeuralJointDecoder with improved funniness and grammaticality scores. Nevertheless, there is still a significant gap between generated puns and expert-written puns across all aspects, indicating that pun generation remains an open challenge.

Table 4 shows the pairwise comparison results among our best model SurGen, NeuralJointDecoder, and expert-written puns. Given the outputs of two systems, we decided win/lose/tie by comparing the average scores of both outputs. We see that SurGen dominates NeuralJointDecoder with $> 50\%$ winning rate on funniness and grammaticality. On success rate, the two methods have many ties since they both have relatively low success rate. Our generated puns were rated funnier than expert-written puns around 10% of the time.

4.4 Error Analysis

In Table 5, we show example outputs of our SurGen, the NeuralJointDecoder baseline, and expert-written puns. SurGen sometimes generates creative puns that are rated even funnier than human-written puns (example 1). In contrast, NeuralJointDecoder at best generates ambiguous sentences (example 2 and 3) and sometimes the sentences are ungrammatical (example 1) or hard to understand (example 4). The examples also show the current limitation of SurGen. In example 3, it failed to realize that "*butter*" is not animate thus cannot "*want*" since our type consistency checker is very simple.

To gain further insights on the limitation of our system, we randomly sampled 50 unsuccessful generations (labeled by workers) to analyze the issues. We characterized the issues into 6 non-exclusive categories: (1) weak association between the local context and w^{a} (e.g., "*...in the form of a batty (bat)*"); (2) w^{p} does not fit in the local context, often due to different POS tags of w^{a} and w^{p} (e.g., "*vibrate with a taxed (text)*"); (3) the topic word is not related to w^{p} (e.g., "*pagan*" vs "*fabrication*"); (4) the topic word does not fit in its immediate context, often due to inconsistent types (e.g., "*slider won't go...*"), (5) grammatical errors; and (6) fail to obtain seed sentences or topic words. A breakdown of these errors is shown in Figure 3. The main issues lie in finding seed sentences that accommodate both the pun word and the topic word. There is also room for improvement in predicting pun-related topic words.

5 Discussion and Related Work

5.1 Humor Theory

Humor involves complex cognitive activities and many theories attempt to explain what might be considered humorous. Among the leading theories, the incongruity theory (Tony, 2004) is most related to our surprisal principle. The incongruity theory posits that humor is perceived at the moment of resolving the incongruity between two concepts, often involving unexpected shifts in perspectives. Ginzburg et al. (2015) applied the incongruity theory to explain laughter in dialogues. Prior work (Kao et al., 2015) on formalizing incongruity theory for puns focuses on ambiguity between two concepts and the heterogeneity nature of the ambiguity. Our surprisal principle further formalizes unexpectedness (local surprisal) and incongruity resolution (global association).

The surprisal principle is also related to studies in psycholinguistics on the relation between surprisal and human comprehension (Levy, 2013; Levy and Gibson, 2013). Our study suggests it could be a fruitful direction to formally study the relationship between human perception of surprisal and humor.

5.2 Humor generation

Early approaches to joke generation (Binsted, 1996; Ritchie, 2005) largely rely on templates for specific types of puns. For example, JAPE (Binsted, 1996) generates noun phrase puns as question-answer pairs, e.g., "*What do you call a [murderer] with [fiber]? A [cereal] [killer].*" Petrovic and Matthews (2013) fill in a joke template based on word similarity and uncommonness. Similar to our editing approach, Valitutti et al. (2013) substitutes a word with a taboo word based on form similarity and local coherence to generate adult jokes. Recently, Yu et al. (2018) generates puns from a generic neural language model by simultaneously conditioning on two meanings. Most of these approaches leverage some assumptions of joke structures, e.g., incongruity, relations between words, and word types. Our approach also relies on specific pun structures; we have proposed and operationalized a local-global surprisal principle for pun generation.

5.3 Creative text generation

Our work is also built upon generic text generation techniques, in particular recent neural generation models. Hashimoto et al. (2018) developed a retrieve-and-edit approach to improve both grammaticality and diversity of the generated text. Shen et al. (2017); Fu et al. (2018) explored adversarial training to manipulate the style of a sentence. Our neural smoother is also closely related to Li et al. (2018)'s delete-retrieve-edit approach to text style transfer.

Creative generation is more challenging as it requires both formality (e.g., grammaticality, rhythm, and rhyme) and novelty. Therefore, many works (including us) impose strong constraints on the generative process, such as Petrovic and Matthews (2013); Valitutti et al. (2013) for joke generation, Ghazvininejad et al. (2016) for poetry generation, and Yao et al. (2019) for storytelling.

6 Conclusion

In this paper, we tackled pun generation by developing and exploring a local-global surprisal principle. We show that a simple instantiation based on only a language model trained on non-humorous text is effective at detecting puns (though is not fine-grained enough to detect the degree of funniness within puns). To generate puns, we operationalize the surprisal principle with a retrieve-and-edit framework to create contrast in the amount of surprise in local and global contexts. While we improve beyond current techniques, we are still far from human-generated puns.

While we believe the local-global surprisal principle is a useful conceptual tool, the principle itself is not quite yet formalized in a robust enough way that can be be used both as a principle for evaluating sentences and can be directly optimized to generate puns. A big challenge in humor, and more generally, creative text generation, is to capture the difference between creativity (novel but well-formed material) and nonsense (ill-formed material). Language models conflate the two, so developing methods that are nuanced enough to recognize this difference is key to future progress.

Acknowledgments

This work was supported by the DARPA CwC program under ISI prime contract no. W911NF-15-1-0543 and ARO prime contract no. W911NF-15-1-0462. We thank Abhinav Moudgil and Justine Kao for sharing their data and results. We also thank members of the Stanford NLP group and USC Plus Lab for insightful discussions.

Reproduciblility

All code, data, and experiments for this paper are available on the CodaLab platform: `https://worksheets.codalab.org/worksheets/0x5a7d0fe35b144ad68998d74891a31ed6`.

References

D. Aarons. 2017. Puns and tacit linguistic knowledge. *The Routledge Handbook of Language and Humor, Routledge, New York, NY, Routledge Handbooks in Linguistics*.

K. Binsted. 1996. *Machine Humour: An Implemented Model of Puns*. Ph.D. thesis, University of Edinburgh.

S. Doogan, A. Ghosh, H. Chen, and T. Veale. 2017. Idiom savant at SemEval-2017 task 7: Detection and interpretation of English puns. In *The 11th International Workshop on Semantic Evaluation*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

A. Fan, M. Lewis, and Y. Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. 2018. Style transfer in text: Exploration and evaluation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

M. Ghazvininejad, X. Shi, Y. Choi, and K. Knight. 2016. Generating topical poetry. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Ginzburg, E. Breithholtz, R. Cooper, J. Hough, and Y. Tian. 2015. Understanding laughter. In *Proceedings of the 20th Amsterdam Colloquium*.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*, 0.

F. Hao, C. Hao, S. Maarten, C. Elizabeth, H. Ari, C. Yejin, S. N. A, and O. Mari. 2018. Sounding board: A user-centric and content-driven social chatbot. *arXiv preprint arXiv:1804.10202*.

T. Hashimoto, K. Guu, Y. Oren, and P. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

J. T. Kao, R. Levy, and N. D. Goodman. 2015. A computational model of linguistic humor in puns. *Cognitive Science*.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

R. Levy. 2013. Memory and surprisal in human sentence comprehension. In *Sentence Processing*.

R. Levy and E. Gibson. 2013. Surprisal, the PDC, and the primary locus of processing difficulty in relative clauses. *Frontiers in Psychology*, 4.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *North American Association for Computational Linguistics (NAACL)*.

H. Manurung, G. Ritchie, and H. Thompson. 2000. Towards a computational model of poetry generation. *The University of Edinburgh Technical Report*.

J. R. Meehan. 1977. TALE-SPIN, an interactive program that writes stories. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

T. Miller, C. Hempelmann, and I. Gurevych. 2017. SemEval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

A. Pauls and D. Klein. 2012. Large-scale syntactic language modeling with treelets. In *Association for Computational Linguistics (ACL)*.

N. Peng, M. Ghazvininejad, J. May, and K. Knight. 2018. Towards controllable story generation. In *NAACL Workshop*.

S. Petrovic and D. Matthews. 2013. Unsupervised joke generation from big data. In *Association for Computational Linguistics (ACL)*.

G. Ritchie. 2005. Computational mechanisms for pun generation. In *the 10th European Natural Language Generation Workshop*.

T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*.

V. Tony. 2004. Incongruity in humor: Root cause or epiphenomenon? *Humor: International Journal of Humor Research*, 17.

A. Valitutti, H. Toivonen, A. Doucet, and J. M. Toivanen. 2013. "let everything turn well in your wife: Generation of adult humor using lexical constraints. In *Association for Computational Linguistics (ACL)*.

J. Weizenbaum. 1966. ELIZA–a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Z. Yu, J. Tan, and X. Wan. 2018. A neural approach to pun generation. In *Association for Computational Linguistics (ACL)*.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.

Single Document Summarization as Tree Induction

Yang Liu, Ivan Titov and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
`yang.liu2@ed.ac.uk, ititov@inf.ed.ac.uk, mlap@inf.ed.ac.uk`

Abstract

In this paper we conceptualize single-document extractive summarization as a tree induction problem. In contrast to previous approaches (Marcu, 1999; Yoshida et al., 2014) which have relied on linguistically motivated document representations to generate summaries, our model *induces* a multi-root dependency tree while predicting the output summary. Each root node in the tree is a summary sentence, and the subtrees attached to it are sentences whose content relates to or explains the summary sentence. We design a new iterative refinement algorithm: it induces the trees through repeatedly refining the structures predicted by previous iterations. We demonstrate experimentally on two benchmark datasets that our summarizer[1] performs competitively against state-of-the-art methods.

1 Introduction

Single-document summarization is the task of automatically generating a shorter version of a document while retaining its most important information. The task has received much attention in the natural language processing community due to its potential for various information access applications. Examples include tools which digest textual content (e.g., news, social media, reviews), answer questions, or provide recommendations.

Of the many summarization paradigms that have been identified over the years (see Mani 2001 and Nenkova and McKeown 2011 for comprehensive overviews), two have consistently attracted attention. In *abstractive* summarization, various text rewriting operations generate summaries using words or phrases that were not in the original text, while *extractive* approaches form summaries by copying and concatenating the most important spans (usually sentences) in a document. Recent approaches to (single-document) extractive summarization frame the task as a sequence labeling problem taking advantage of the success of neural network architectures (Bahdanau et al., 2015). The idea is to predict a label for each sentence specifying whether it should be included in the summary. Existing systems mostly rely on recurrent neural networks (Hochreiter and Schmidhuber, 1997) to model the document and obtain a vector representation for each sentence (Nallapati et al., 2017; Cheng and Lapata, 2016). Inter-sentential relations are captured in a sequential manner, without taking the structure of the document into account, although the latter has been shown to correlate with what readers perceive as important in a text (Marcu, 1999). Another problem in neural-based extractive models is the lack of interpretability. While capable of identifying summary sentences, these models are not able to rationalize their predictions (e.g., a sentence is in the summary because it describes important content upon which other related sentences elaborate).

The summarization literature offers examples of models which exploit the structure of the underlying document, inspired by existing theories of discourse such as Rhetorical Structure Theory (RST; Mann and Thompson 1988). Most approaches produce summaries based on tree-like document representations obtained by a parser trained on discourse annotated corpora (Carlson et al., 2003; Prasad et al., 2008). For instance, Marcu (1999) argues that a good summary can be generated by traversing the RST discourse tree structure top-down, following nucleus nodes (discourse units in RST are characterized regarding their text importance; nuclei denote central units, whereas satellites denote peripheral ones). Other work (Hirao et al., 2013; Yoshida et al., 2014) extends this idea by transforming RST trees into dependency trees and generating summaries by tree trimming. Gerani et al. (2014) summarize product reviews; their system aggregates RST trees rep-

---

[1] Our code is publicly available at `https://github.com/nlpyang/SUMO`.

Proceedings of NAACL-HLT 2019, pages 1745–1755
Minneapolis, Minnesota, June 2 - June 7, 2019.

1. One wily coyote traveled a bit too far from home, and its resulting adventure through Harlem had alarmed residents doing a double take and scampering to get out of its way Wednesday morning.
2. Police say frightened New Yorkers reported the coyote sighting around 9:30 a.m., and an emergency service unit was dispatched to find the animal.
3. The little troublemaker was caught and tranquilized in Trinity Cemetery on 155th street and Broadway, and then taken to the Wildlife Conservation Society at the Bronx Zoo, authorities said.
4. "The coyote is under evaluation and observation," said Mary Dixon, spokesperson for the Wildlife Conservation Society.
5. She said the Department of Environmental Conservation will either send the animal to a rescue center or put it back in the wild.
6. According to Adrian Benepe, New York City Parks Commissioner, coyotes in Manhattan are rare, but not unheard of.
7. "This is actually the third coyote that has been seen in the last 10 years," Benepe said.
8. Benepe said there is a theory the coyotes make their way to the city from suburban Westchester.
9. He said they probably walk down the Amtrak rail corridor along the Hudson River or swim down the Hudson River until they get to the city.

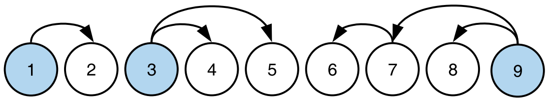


Figure 1: Dependency discourse tree for a document from the CNN/DailyMail dataset (Hermann et al., 2015). Blue nodes indicate the roots of the tree (i.e., summary sentences) and parent-child links indicate dependency relations.

resenting individual reviews into a graph, from which an abstractive summary is generated. Despite the intuitive appeal of discourse structure for the summarization task, the reliance on a parser which is both expensive to obtain (since it must be trained on labeled data) and error prone, presents a major obstacle to its widespread use.

Recognizing the merits of structure-aware representations for various NLP tasks, recent efforts have focused on learning latent structures (e.g., parse trees) while optimizing a neural network model for a down-stream task. Various methods impose structural constraints on the basic attention mechanism (Kim et al., 2017; Liu and Lapata, 2018), formulate structure learning as a reinforcement learning problem (Yogatama et al., 2017; Williams et al., 2018), or sparsify the set of possible structures (Niculae et al., 2018). Although latent structures are mostly induced for individual sentences, Liu and Lapata (2018) induce dependency-like structures for entire documents.

Drawing inspiration from this work and existing discourse-informed summarization models (Marcu, 1999; Hirao et al., 2013), we frame extractive summarization as a tree induction problem. Our model represents documents as multi-root dependency trees where each root node is a summary sentence, and the subtrees attached to it are sentences whose content is related to and covered by the summary sentence. An example of a document and its corresponding tree is shown in Figure 1; tree nodes correspond to document sentences; blue nodes represent those which should be in the summary, dependent nodes relate to or are subsumed by the parent summary sentence.

We propose a new framework that uses structured attention (Kim et al., 2017) as both the objective and attention weights for extractive summarization. Our model is trained end-to-end, it induces document-level dependency trees while predicting the output summary, and brings more interpretability in the summarization process by helping explain how document content contributes to the model's decisions. We design a new iterative structure refinement algorithm, which learns to induce document-level structures through repeatedly refining the trees predicted by previous iterations and allows the model to infer complex trees which go beyond simple parent-child relations (Liu and Lapata, 2018; Kim et al., 2017). The idea of structure refinement is conceptually related to recently proposed models for solving iterative inference problems (Marino et al., 2018; Putzky and Welling, 2017; Lee et al., 2018). It is also related to structured prediction energy networks (Belanger et al., 2017) which approach structured prediction as iterative miminization of an energy function. However, we are not aware of any previous work considering structure refinement for tree induction problems.

Our contributions in this work are three-fold: a novel conceptualization of extractive summarization as a tree induction problem; a model which capitalizes on the notion of structured attention to learn document representations based on iterative structure refinement; and large-scale evaluation studies (both automatic and human-based) which demonstrate that our approach performs competitively against state-of-the-art methods while being able to rationalize model predictions.

2 Model Description

Let d denote a document containing several sentences $[sent_1, sent_2, \cdots, sent_m]$, where $sent_i$ is the i-th sentence in the document. Extractive summarization can be defined as the task of assigning a label $y_i \in \{0, 1\}$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

2.1 Baseline Model

Most extractive models frame summarization as a classification problem. Recent approaches (Zhang et al., 2018; Dong et al., 2018; Nallapati et al., 2017; Cheng and Lapata, 2016) incorporate a neural network-based encoder to build representations for sentences and apply a binary classifier over these representations to predict whether the sentences should be included in the summary. Given predicted scores r and gold labels y, the loss function can be defined as:

$$L = -\sum_{i=1}^{m} (\boldsymbol{y}_i \ln(\boldsymbol{r}_i) + (1 - \boldsymbol{y}_i) \ln(1 - \boldsymbol{r}_i)) \quad (1)$$

The encoder in extractive summarization models is usually a recurrent neural network with Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber 1997) or Gated Recurrent Units (GRU; Cho et al. 2014). In this paper, our baseline encoder builds on the Transformer architecture (Vaswani et al., 2017), a recently proposed highly efficient model which has achieved state-of-the-art performance in machine translation (Vaswani et al., 2017) and question answering (Yu et al., 2018). The Transformer aims at reducing the fundamental constraint of sequential computation which underlies most architectures based on RNNs. It eliminates recurrence in favor of applying a self-attention mechanism which directly models relationships between all words in a sentence.

More formally, given a sequence of input vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, the Transformer is composed of a stack of N identical layers, each of which has two sub-layers:

$$\tilde{\boldsymbol{h}}^l = \mathrm{LayerNorm}(\boldsymbol{h}^{l-1} + \mathrm{MHAtt}(\boldsymbol{h}^{l-1})) \quad (2)$$

$$\boldsymbol{h}^l = \mathrm{LayerNorm}(\tilde{\boldsymbol{h}}^l + \mathrm{FFN}(\tilde{\boldsymbol{h}}^l)) \quad (3)$$

where $\boldsymbol{h}^0 = \mathrm{PosEmb}(\boldsymbol{x})$ and PosEmb is the function of adding positional embeddings to the input; the superscript l indicates layer depth; LayerNorm is the layer normalization operation proposed in Ba et al. (2016); MHAtt represents the multi-head attention mechanism introduced in Vaswani et al. (2017) which allows the model to jointly attend to information from different representation subspaces (at different positions); and FFN is a two-layer feed-forward network with ReLU as hidden activation function.

For our extractive summarization task, the baseline system is composed of a sentence-level Transformer ($\mathcal{T}_\mathrm{S}$) and a document-level Transformer ($\mathcal{T}_\mathrm{D}$), which have the same structure. For each sentence $s_i = [\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \cdots, \boldsymbol{w}_{in}]$ in the input document, $\mathcal{T}_\mathrm{S}$ is applied to obtain a contextual representation for each word:

$$[\boldsymbol{u}_{i1}, \boldsymbol{u}_{i2}, \cdots, \boldsymbol{u}_{in}] = \mathcal{T}_\mathrm{S}([\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \cdots, \boldsymbol{w}_{in}]) \quad (4)$$

And the representation of a sentence is acquired by applying weighted-pooling:

$$\boldsymbol{a}_{ij} = \boldsymbol{W}_0 \boldsymbol{u}_{ij}^T \quad (5)$$

$$\boldsymbol{s}_i = \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{a}_{ij} \boldsymbol{u}_{ij} \quad (6)$$

Document-level transformer $\mathcal{T}_\mathrm{D}$ takes $\boldsymbol{s}_i$ as input and yields a contextual representation for each sentence:

$$[\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m] = \mathcal{T}_\mathrm{D}([\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_m]) \quad (7)$$

Following previous work (Nallapati et al., 2017), we use a sigmoid function after a linear transformation to calculate the probability $\boldsymbol{r}_i$ of selecting s_i as a summary sentence:

$$\boldsymbol{r}_i = \mathrm{sigmoid}(\boldsymbol{W}_1 \boldsymbol{v}_i^T) \quad (8)$$

2.2 Structured Summarization Model

In the Transformer model sketched above, inter-sentence relations are modeled by multi-head attention based on softmax functions, which only capture shallow structural information. Our summarizer, which we call SUMO as a shorthand for **S**tructured **Sum**marization **Mo**del classifies sentences as summary-worthy or not, and simultaneously induces the structure of the source document as a multi-root tree. An overview of SUMO is illustrated in Figure 2. The model has the same sentence-level encoder $\mathcal{T}_\mathrm{S}$ as the baseline Transformer model (see the bottom box in Figure 2), but differs in two important ways: (a) it uses structured attention to model the roots (i.e., summary sentences) of the underlying tree (see the upper box in Figure 2); and (b) through iterative refinement it is able to progressively infer more complex structures from past guesses (see the second and third block in Figure 2).

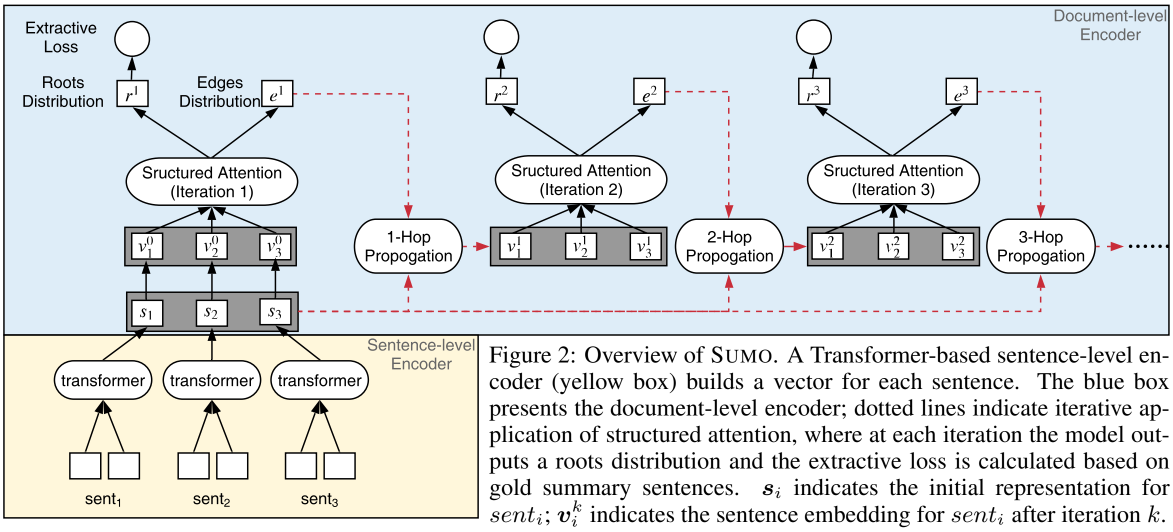


Figure 2: Overview of SUMO. A Transformer-based sentence-level encoder (yellow box) builds a vector for each sentence. The blue box presents the document-level encoder; dotted lines indicate iterative application of structured attention, where at each iteration the model outputs a roots distribution and the extractive loss is calculated based on gold summary sentences. $\boldsymbol{s}_i$ indicates the initial representation for $sent_i$; $\boldsymbol{v}_i^k$ indicates the sentence embedding for $sent_i$ after iteration k.

Structured Attention Assuming document sentences have been already encoded, SUMO first calculates the unnormalized root score $\tilde{\boldsymbol{r}}_i$ for $sent_i$ to indicate the extent to which it might be selected as root in the document tree. It also calculates the unnormalized edge score $\tilde{\boldsymbol{e}}_{ij}$ for sentence pair $\langle sent_i, sent_j \rangle$ indicating the extent to which $sent_i$ might be the head of $sent_j$ in that tree (first upper block in Figure 2). To inject structural bias, SUMO normalizes these scores as the marginal probabilities of forming edges in the document dependency tree.

We use the Tree-Matrix-Theorem (TMT; Koo et al. 2007; Tutte 1984) to calculate root marginal probability $\boldsymbol{r}_i$ and edge marginal probability $\boldsymbol{e}_{ij}$, following the procedure introduced in Liu and Lapata (2017). As illustrated in Algorithm 1, we first build the Laplacian matrix $\bar{\boldsymbol{L}}$ based on unnormalized scores and calculate marginal probabilities by matrix inverse-based operations ($\bar{\boldsymbol{L}}^{-1}$). We refer the interested reader to Koo et al. (2007) and Liu and Lapata (2017) for more details. In contrast to Liu and Lapata (2017), who compute the marginal probabilities of a single-root tree, our tree has multiple roots since in our task the summary typically contains multiple sentences. Given sentence vector $\boldsymbol{s}_i$ as input, SUMO computes:

$$\tilde{\boldsymbol{r}}_i = \boldsymbol{W}_r \boldsymbol{s}_i \tag{9}$$

$$\tilde{\boldsymbol{e}}_{ij} = \boldsymbol{s}_i \boldsymbol{W}_e \boldsymbol{s}_j^T \tag{10}$$

$$\boldsymbol{r}_i, \boldsymbol{e}_{ij} = \mathrm{TMT}(\tilde{\boldsymbol{r}}_i, \tilde{\boldsymbol{e}}_{ij}) \tag{11}$$

Iterative Structure Refinement SUMO essentially reduces summarization to a rooted-tree parsing problem. However, accurately predicting a tree in one shot is problematic. Firstly, when predicting the dependency tree, the model has solely access to labels for the roots (aka summary sentences), while tree edges are latent and learned without an explicit training signal. And as previous work (Liu and Lapata, 2017) has shown, a single application of TMT leads to shallow tree structures. Secondly, the calculation of $\tilde{\boldsymbol{r}}_i$ and $\tilde{\boldsymbol{e}}_{ij}$ would be based on first-order features alone, however, higher-order information pertaining to siblings and grandchildren has proved useful in discourse parsing (Carreras, 2007).

Algorithm 1: Calculate Tree Marginal Probabilities based on Tree-Matrix-Theorem

```
Function TMT ($\tilde{\boldsymbol{r}}_i, \tilde{\boldsymbol{e}}_{ij}$):
    $\boldsymbol{A}_{ij} = \begin{cases} 0 & \text{if } i = j \\ exp(\hat{\boldsymbol{r}}_{ij}) & \text{otherwise} \end{cases}$
    $\boldsymbol{L}_{ij} = \begin{cases} \sum_{i'=1}^{n} \boldsymbol{A}_{i'j} & \text{if } i = j \\ -\boldsymbol{A}_{ij} & \text{otherwise} \end{cases}$
    $\bar{\boldsymbol{L}}_{ij} = \begin{cases} \boldsymbol{L}_{ij} + exp(\hat{\boldsymbol{r}}_i) & i = j \\ \boldsymbol{L}_{ij} & \text{otherwise} \end{cases}$
    $\boldsymbol{e}_{ij} = (1 - \delta_{1,j}) \boldsymbol{A}_{ij} [\bar{\boldsymbol{L}}^{-1}]_{jj}$
        $- (1 - \delta_{i,1}) \boldsymbol{A}_{ij} [\bar{\boldsymbol{L}}^{-1}]_{ji}$
    $\boldsymbol{r}_i = exp(\hat{\boldsymbol{r}}_i) [\bar{\boldsymbol{L}}^{-1}]_{i1}$
    return $\boldsymbol{r}_i, \boldsymbol{e}_{ij}$
```

We address these issues with an inference algorithm which iteratively infers latent trees. In contrast to multi-layer neural network architectures like the Transformer or Recursive Neural Networks (Tai et al., 2015) where word representations are updated at every layer based on the output of previous layers, we refine only the tree structure during each iteration, word representations are not passed across multiple layers. Empirically, at early iterations, the model learns shallow and simple trees, and information propagates mostly between neighboring nodes; as the structure gets more refined, information propagates more globally allowing the model to learn higher-order features.

Algorithm 2 provides the details of our refinement procedure. SUMO takes K iterations to learn the structure of a document. For each sentence, we initialize a structural vector $\boldsymbol{v}_i^0$ with sentence vector $\boldsymbol{s}_i$. At iteration k, we use sentence embeddings from the previous iteration $\boldsymbol{v}^{k-1}$ to calculate unnormalized root $\tilde{\boldsymbol{r}}_i^k$ and edge $\tilde{\boldsymbol{e}}_{ij}^k$ scores using a linear transformation with weight $\boldsymbol{W}_r^k$ and a bilinear transformation with weight $\boldsymbol{W}_e^k$, respectively. Marginal root and edge probabilities are subsequently normalized with the TMT to obtain $\boldsymbol{r}_i^k$ and $\boldsymbol{e}_{ij}^k$ (see lines 4–6 in Algorithm 2). Then, sentence embeddings are updated with *k-Hop Propagation*. The latter takes as input the initial sentence representations $\boldsymbol{s}$ rather than sentence embeddings $\boldsymbol{v}^{k-1}$ from the previous layer. In other words, new embeddings $\boldsymbol{v}^k$ are computed from scratch relying on the structure from the previous layer. Within the *k-Hop-Propagation* function (lines 12–19), edge probabilities $\boldsymbol{e}_{ij}^k$ are used as attention weights to propagate information from a sentence to all other sentences in k hops. $\boldsymbol{p}_i^l$ and $\boldsymbol{c}_i^l$ represent parent and child vectors, respectively, while vector $\boldsymbol{z}_i^l$ is updated with contextual information at hop l. At the final iteration (lines 9 and 10), the top sentence embeddings $\boldsymbol{v}^{K-1}$ are used to calculate the final root probabilities r^K.

We define the model's loss function as the summation of the losses of all iterations:

$$L = \sum_{k=1}^{K} [y \log(r_k) + (1-y)\log(1-r_k)] \quad (12)$$

SUMO uses the root probabilities of the top layer as the scores for summary sentences.

The *k-Hop-Propagation* function resembles the computation used in Graph Convolution Networks (Kipf and Welling, 2017; Marcheggiani and Titov, 2017). GCNs have been been recently applied to latent trees (Corro and Titov, 2019), however not in combination with iterative refinement.

3 Experiments

In this section we present our experimental setup, describe the summarization datasets we used, discuss implementation details, our evaluation protocol, and analyze our results.

```
Algorithm 2: Structured Summarization Model
    Input: Document $d$
    Output: Root probabilities $\boldsymbol{r}^K$ after $K$
            iterations
 1  Calculate sentence vectors $\boldsymbol{s}$ using
      sentence-level Transformer $T_S$
 2  $\boldsymbol{v}^0 \leftarrow \boldsymbol{s}$
 3  for $k \leftarrow 1$ to $K-1$ do
 4      Calculate unnormalized root scores:
          $\tilde{\boldsymbol{r}}_i^k = \boldsymbol{W}_r^k \boldsymbol{v}_i^{k-1}$
 5      Calculate unnormalized edge scores:
          $\tilde{\boldsymbol{e}}_{ij}^k = \boldsymbol{v}_i^{k-1} \boldsymbol{W}_e^k \boldsymbol{v}_j^{k-1^T}$
 6      Calculate marginal probabilities:
          $\boldsymbol{r}^k, \boldsymbol{e}^k = \mathrm{TMT}(\tilde{\boldsymbol{r}}^k, \tilde{\boldsymbol{e}}^k)$
 7      Update sentence representations:
          $\boldsymbol{v}^k = \text{k-Hop-Propagation}(\boldsymbol{e}^k, \boldsymbol{s}, k)$
 8  end
 9  Calculate final unnormalized root and edge
      scores: $\tilde{\boldsymbol{r}}_i^K = \boldsymbol{W}_r^K \boldsymbol{v}_i^{K-1}$,
      $\tilde{\boldsymbol{e}}_{ij}^K = \boldsymbol{v}_i^{K-1} \boldsymbol{W}_e^K \boldsymbol{v}_j^{K-1^T}$
10  Calculate final root and edge probabilities:
      $\boldsymbol{r}^K, \boldsymbol{e}^K = \mathrm{TMT}(\tilde{\boldsymbol{r}}^K, \tilde{\boldsymbol{e}}^K)$
11
12  Function k-Hop-Propagation(e, s, k):
13      $\boldsymbol{z}^0 \leftarrow \boldsymbol{s}$
14      for $l \leftarrow 1$ to $k$ do
15          $\boldsymbol{p}_i^l = \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{e}_{ji} \boldsymbol{z}_i^{l-1}$
16          $\boldsymbol{c}_i^l = \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{e}_{ij} \boldsymbol{z}_i^{l-1}$
17          $\boldsymbol{z}_i^l = \tanh(\boldsymbol{W}_v^k [\boldsymbol{p}_i^{l-1}, \boldsymbol{c}_i^{l-1}, \boldsymbol{z}_i^{l-1}])$
18      end
19      return $\boldsymbol{z}^k$
```

3.1 Summarization Datasets

We evaluated SUMO on two benchmark datasets, namely the CNN/DailyMail news highlights dataset (Hermann et al., 2015) and the New York Times Annotated Corpus (NYT; Sandhaus 2008). The CNN/DailyMail dataset contains news articles and associated highlights, i.e., a few bullet points giving a brief overview of the article. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents). We did not anonymize entities.

The NYT dataset contains 110,540 articles with abstractive summaries. Following Durrett et al. (2016), we split these into 100,834 training and 9,706 test examples, based on date of publication (test is all articles published on January 1, 2007 or later). We also followed their filtering procedure, documents with summaries that are shorter than 50 words were removed from the raw dataset. The

	CNN			DM			CNN+DM			NYT		
Model	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Lead-3	29.2	11.2	26.0	40.7	18.3	37.2	39.6	17.7	36.2	35.5	17.3	32.0
Narayan et al. (2018)	**30.4**	11.7	**26.9**	41.0	18.8	37.7	40.0	18.2	36.6	41.3	22.0	37.8
Marcu (1999)	25.6	6.10	19.5	31.9	12.4	23.5	26.5	9.80	20.4	29.6	11.2	23.0
Durrett et al. (2016)	—	—	—	—	—	—	—	—	—	40.8	22.3	36.7
See et al. (2017)	—	—	—	—	—	—	39.5	17.3	36.4	**42.7**	22.1	38.0
Celikyilmaz et al. (2018)	—	—	—	—	—	—	**41.7**	19.5	**37.9**	—	—	—
Transformer (no doc-att)	29.2	11.1	25.6	40.5	18.1	36.8	39.7	17.0	35.9	41.1	21.5	37.0
Transformer (1-layer doc-att)	29.5	11.4	26.0	41.5	18.7	38.0	40.6	18.1	36.7	41.8	22.1	37.8
Transformer (3-layer doc-att)	29.6	11.8	26.3	41.7	18.8	38.0	40.6	18.1	36.9	42.0	22.3	38.2
Sumo (1-layer)	29.5	11.6	26.2	41.6	18.8	37.6	40.5	18.0	36.8	42.2	22.1	38.1
Sumo (3-layer)	29.7	**12.0**	26.5	**42.0**	**19.1**	**38.0**	41.0	18.4	37.2	42.3	**22.7**	**38.6**

Table 1: Test set results on the CNN/DailyMail and NYT datasets using ROUGE F_1 (R-1 and R-2 are shorthands for unigram and bigram overlap, R-L is the longest common subsequence.

filtered test set includes 3,452 test examples out of the original 9,706. Compared to CNN/DailyMail, the NYT dataset contains longer and more elaborate summary sentences.

Both datasets contain abstractive gold summaries, which are not readily suited to training extractive summarization models. A greedy algorithm similar to Nallapati et al. (2017) was used to generate an oracle summary for each document. The algorithm explores different combinations of sentences and generates an oracle consisting of multiple sentences which maximize the ROUGE score with the gold summary. We assigned label 1 to sentences selected in the oracle summary and 0 otherwise and trained Sumo on this data.

3.2 Implementation Details

We followed the same training procedure for Sumo and various Transformer-based baselines. The vocabulary size was set to 30K. We used 300D word embeddings which were initialized randomly from $\mathcal{N}(0, 0.01)$. The sentence-level Transformer has 6 layers and the hidden size of FFN was set to 512. The number of heads in MHAtt was set to 4. Adam was used for training ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We adopted the learning rate schedule from Vaswani et al. (2017) with warming-up on the first 8,000 steps. Sumo and related Transformer models produced 3-sentence summaries for each document at test time (for both CNN/DailyMail and NYT datasets).

3.3 Automatic Evaluation

We evaluated summarization quality using ROUGE F_1 (Lin, 2004). We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

Table 1 summarizes our results. We evaluated two variants of Sumo, with one and three structured-attention layers. We compared against a baseline which simply selects the first three sentences in each document (Lead-3) and several incarnations of the basic Transformer model introduced in Section 2.1. These include a Transformer without document-level self-attention and two variants with document-level self attention instantiated with one and three layers. Several state-of-the-art models are also included in Table 1, both extractive and abstractive.

Refresh (Narayan et al., 2018) is an extractive summarization system trained by globally optimizing the ROUGE metric with reinforcement learning. The system of Marcu (1999) is another extractive summarizer based on RST parsing. It uses discourse structures and RST's notion of nuclearity to score document sentences in terms of their importance and selects the most important ones as the summary. Our re-implementation of Marcu (1999) used the parser of Zhao and Huang (2017) to obtain RST trees. Durrett et al. (2016) develop a summarization system which integrates a compression model that enforces grammaticality and coherence. See et al. (2017) present an abstractive summarization system based on an encoder-decoder architecture. Celikyilmaz et al.'s (2018) system is state-of-the-art in abstractive summarization using multiple agents to represent the document as well a hierarchical attention mechanism over the agents for decoding.

As far as Sumo is concerned, we observe that it outperforms a simple Transformer model without any document attention as well as variants with document attention. Sumo with three layers of structured attention overall performs best, confirming our hypothesis that document-level structure is beneficial for summarization. The results in Table 1 also reveal that Sumo and all Transformer-based models with document attention (doc-att) outperform Lead-3 across metrics. Sumo (3-layer) is competitive or better than state-of-the-art approaches. Examples of system output are shown in Table 4.

Finally, we should point out that Sumo is superior to Marcu (1999) even though the latter employs linguistically informed document representations.

3.4 Human Evaluation

In addition to automatic evaluation, we also assessed system performance by eliciting human judgments. Our first evaluation quantified the degree to which summarization models retain key information from the document following a question-answering (QA) paradigm (Clarke and Lapata, 2010; Narayan et al., 2018). We created a set of questions based on the gold summary under the assumption that it highlights the most important document content. We then examined whether participants were able to answer these questions by reading system summaries alone without access to the article. The more questions a system can answer, the better it is at summarizing the document as a whole.

We randomly selected 20 documents from the CNN/DailyMail and NYT datasets, respectively and wrote multiple question-answer pairs for each gold summary. We created 71 questions in total varying from two to six questions per gold summary. We asked participants to read the summary and answer all associated questions as best they could without access to the original document or the gold summary. Examples of questions and their answers are given in Table 4. We adopted the same scoring mechanism used in Clarke and Lapata (2010), i.e., a correct answer was marked with a score of one, partially correct answers with a score of 0.5, and zero otherwise. Answers were elicited using Amazon's Mechanical Turk platform. Participants evaluated summaries produced by the Lead-3 baseline, our 3-layered Sumo model and multiple state-of-the-art systems. We elicited 5 responses per summary.

Model	CNN+DM Rank	CNN+DM QA	NYT Rank	NYT QA
Lead	0.07	40.1	-0.18	36.3
Narayan et al. (2018)	**0.21**	62.4	0.12	46.1
Durrett et al. (2016)	—	—	-0.11	40.1
See et al. (2017)	-0.23	36.6	-0.44	35.3
Celikyilmaz et al. (2018)	-0.64	37.5	—	—
Sumo (3-layer)	0.15	**65.3**	**0.33**	**57.2**
Gold	0.11	—	-0.16	—
Oracle	0.37	74.6	0.41	67.1

Table 2: System ranking according to human judgments on summary quality and QA-based evaluation.

Table 2 (QA column) presents the results of the QA-based evaluation. Based on the summaries generated by Sumo, participants can answer 65.3% of questions correctly on CNN/DailyMail and 57.2% on NYT. Summaries produced by Lead-3 and comparison systems fare worse, with Refresh (Narayan et al., 2018) coming close to Sumo on CNN/DailyMail but not on NYT. Overall, we observe there is room for improvement since no system comes close to the extractive oracle, indicating that improved sentence selection would bring further performance gains to extractive approaches. Between-systems differences are all statistically significant (using a one-way ANOVA with posthoc Tukey HSD tests; $p < 0.01$) with the exception of Lead-3 and See et al. (2017) in both CNN+DM and NTY, Narayan et al. (2018) and Sumo in both CNN+DM and NTY, and Lead-3 and Durrett et al. (2016) in NYT.

Our second evaluation study assessed the overall quality of the summaries by asking participants to rank them taking into account the following criteria: *Informativeness* , *Fluency*, and *Succinctness*. The study was conducted on the Amazon Mechanical Turk platform using Best-Worst Scaling (Louviere et al., 2015), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2017). Participants were presented with a document and

	CNN+DM			NYT		
	P	H	EA	P	H	EA
Parser	24.8	8.9	—	18.7	10.6	—
Sumo (1-layer)	69.0	2.9	23.1	54.7	3.6	20.6
Sumo (3-layer)	52.7	3.7	25.3	45.1	6.2	21.6
Left Branching	—	—	21.4	—	—	21.3
Right Branching	—	—	7.3	—	—	6.7

Table 3: Descriptive statistics Projectivity(%), Height and EdgeAgreement(%) for dependency trees produced by our model and the RST discourse parser of Zhao and Huang (2017). Results are shown on the CNN/DailyMail and NYT test sets.

summaries generated from 3 out of 7 systems and were asked to decide which summary was better and which one was worse, taking into account the criteria mentioned above. We used the same 20 documents from each dataset as in our QA evaluation and elicited 5 responses per comparison.

The rating of each system was computed as the percentage of times it was chosen as best minus the times it was selected as worst. Ratings range from -1 (worst) to 1 (best). As shown in Table 2 (Rank column), participants overwhelming prefer the extractive oracle summaries followed by Sumo and Refresh (Narayan et al., 2018). Abstractive systems (Celikyilmaz et al., 2018; See et al., 2017; Durrett et al., 2016) perform relatively poorly in this evaluation; we suspect that humans are less forgiving to fluency errors and slightly incoherent summaries. Interestingly, gold summaries fare worse than the oracle and extractive systems. Albeit fluent, gold summaries naturally contain less detail compared to oracle-based ones; on virtue of being abstracts, they are written in a telegraphic style, often in conversational language while participants prefer the more lucid style of the extracts. All pairwise comparisons among systems are statistically significant (using a one-way ANOVA with post-hoc Tukey HSD tests; $p < 0.01$) except Lead-3 and See et al. (2017) in both CNN+DM and NTY, Narayan et al. (2018) and Sumo in both CNN+DM and NTY, and LEAD and Durrett et al. (2016) in NYT.

3.5 Evaluation of the Induced Structures

To gain further insight into the structures learned by Sumo, we inspected the trees it produces. Specifically, we used the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) to extract the maximum spanning tree from the attention scores. We report various statistics on the characteristics of the induced trees across datasets in Table 3. We also examine the trees learned from different Sumo variants (with different numbers of iterations) in order to establish whether the iterative process yields better structures.

Specifically, we compared the dependency trees obtained from our model to those produced by a discourse parser (Zhao and Huang, 2017) trained on a corpus which combines annotations from the RST treebank (Carlson et al., 2003) and the Penn Treebank (Marcus et al., 1993). Unlike traditional RST discourse parsers (Feng and Hirst, 2014), which first segment a document into Elementary Discourse Units (EDUs) and then build a discourse tree with the EDUs[2] as leaves, Zhao and Huang (2017) parse a document into an RST tree along with its syntax subtrees without segmenting it into EDUs. The outputs of their parser are ideally suited for comparison with our model, since we only care about document-level structures, and ignore the subtrees within sentence boundaries. We converted the constituency RST trees obtained from the discourse parser into dependency trees using Hirao et al.'s algorithm (2013).

As can be seen in Table 3, the dependency structures induced by Sumo are simpler compared to those obtained from the discourse parser. Our trees are generally shallower, almost half of them are projective. We also calculated the percentage of head-dependency edges that are identical between learned trees and parser generated ones. Although Sumo is not exposed to any annotated trees during training, a number of edges agree with the outputs of the discourse parser. Moreover, we observe that the iterative process involving multiple structured attention layers helps generate better discourse trees. We also compare Sumo trees against a left- and right-branching baseline, where the document is trivially parsed into a left- and right-branching tree forming a chain-like structure. As shown in Table 3, Sumo outperforms these baselines (with the exception of the one-layered model on NYT). We should also point out that the edge agreement between Sumo generated trees and left/right branching trees is low (around 30% on both datasets), indicating that the trees we learn are different from a simple chain.

---

[2] EDUs roughly correspond to clauses.

	CNN/DM	NYT
Gold	A company called CyArk specializes in digital preservation of threatened ancient and historical architecture. Founded by an Iraqi-born engineer, it plans to preserve 500 World Heritage sites within five years.	Louisiana officials set July 31 deadline for applicants for the Road Home, grant program for homeowners who lost their houses to hurricanes Katrina and Rita. Program is expected to cost far more than $7.5 billion provided by Federal Government, in part because many more families have applied than officials anticipated. With cutoff date, State hopes to figure out how much more money it needs to pay for program. Shortfall is projected to be $2.9 billion.
QA	Which company specializes in digital preservation of threatened ancient and historical architecture? [CyArk] How many World Heritage sites does the company plan to preserve? [500]	What is Road Home? [the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita] When is the applicants' deadline for the Road Home? [July 31] Why is the program expected to cost far more than $7.5 billion? [many more families have applied than officials anticipated] What is the shortfall projected to be? [$2.9 billion]
Lead-3	In 2001, the Taliban wiped out 1700 years of history in a matter of seconds, by blowing up ancient Buddha statues in central Afghanistan with dynamite. They proceeded to do so after an attempt at bringing down the 175-foot tall sculptures with anti-aircraft artillery had failed. Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the Federal Government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. With the cutoff date, the State hopes to be able to figure out how much more money it needs to pay for the program.
See et al. (2017)	The Taliban wiped out 1700 years of history in a matter of seconds. The thought of losing a piece of our collective history is a bleak one. But if loss can't be avoided, technology can lend a hand.	Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita is expected to cost far more than $7.5 billion provided by federal government. Louisiana officials set July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses.
Narayan et al. (2018)	Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage. But historical architecture is also under threat from calamities which might well escape our control, such as earthquakes and climate change. The thought of losing a piece of our collective history is a bleak one.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the federal government, in part because many more families have applied than officials had anticipated. With the cutoff date, the State hopes to be able to figure out how much more money it needs to pay for the program. The shortfall is projected to be $2.9 billion.
Sumo	In 2001, the Taliban wiped out 1700 years of history in a matter of seconds, by blowing up ancient Buddha statues in central Afghanistan with dynamite. Sadly, the event was just the first in a series of atrocities that have robbed the world of some of its most prized cultural heritage. Now Cyark, a non-profit company founded by an Iraqi-born engineer, is using groundbreaking laser scanning to ensure that – at the very least – incredibly accurate digital versions of the world's treasures will stay with us forever.	The Road Home, the Louisiana grant program for homeowners who lost their houses to hurricanes Katrina and Rita, is expected to cost far more than the $7.5 billion provided by the federal government, in part because many more families have applied than officials had anticipated. As a result, Louisiana officials on Tuesday night set a July 31 deadline for applicants, who can receive up to $150,000 to repair or rebuild their houses. The shortfall is projected to be $2.9 billion.

Table 4: Gold human authored summaries, questions based on them (answers shown in square brackets) and automatic summaries produced by the Lead-3 baseline, the abstractive system of See et al. (2017), Refresh (Narayan et al., 2018), and Sumo for a CNN and NYT (test) article.

4 Conclusions

In this paper we provide a new perspective on extractive summarization, conceptualizing it as a tree induction problem. We present Sumo, a Structured Summarization Model, which induces a multi-root dependency tree of a document, where roots are summary-worthy sentences, and subtrees attached to them are sentences which elaborate or explain the summary content. Sumo generates complex trees following an iterative refinement process which builds latent structures while using information learned in previous iterations. Experiments on two datasets, show that Sumo performs competitively against state-of-the-art methods and induces meaningful tree structures.

In the future, we would like to generalize Sumo to abstractive summarization (i.e., to learn latent structure for documents *and* sentences) and perform experiments in a weakly-supervised setting where summaries are not available but labels can be extrapolated from the article's title or topics.

Acknowledgments

We thank Serhii Havrylov for helpful suggestions. This research is supported by a Google PhD Fellowship to the first author. We gratefully acknowledge the support of the European Research Council (Lapata, award number 681760, "Translating Multiple Modalities into Text"; Titov award number 678254, "Broad Coverage Semantic Parsing").

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *In Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. *ICML*.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Caio Corro and Ivan Titov. 2019. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *ICLR*.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the EMNLP Conference*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. *arXiv preprint arXiv:1603.08887*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 511–521.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 465–470, Vancouver, Canada.

Terry Koo, Amir Globerson, Xavier Carreras Pérez, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *EMNLP*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2017. Learning structured text representations. *arXiv preprint arXiv:1705.09207*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Pub Co.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1507–1516, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Marcu. 1999. Discourse trees are good indicators of importance in text. *Advances in automatic text summarization*, 293:123–136.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Joseph Marino, Yisong Yue, and Stephan Mandt. 2018. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412, Stockholm, Sweden.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Vlad Niculae, André F. T. Martins, and Claire Cardie. 2018. Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*, Marrakech, Morocco. Citeseer.

Patrick Putzky and Max Welling. 2017. Recurrent inference machines for solving inverse problems. *CoRR*, abs/1706.04008.

Evan Sandhaus. 2008. The new york times annotated corpus. Linguistic Data Consortium. Philadelphia.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 2015 Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566, Beijing, China.

William Thomas Tutte. 1984. Graph theory.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the EMNLP Conference*.

Kai Zhao and Liang Huang. 2017. Joint syntacto-discourse parsing and the syntacto-discourse treebank. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2117–2123, Copenhagen, Denmark.

Fixed That for You: Generating Contrastive Claims with Semantic Edits

Christopher Hidey
Department of Computer Science
Columbia University
New York, NY 10027
chidey@cs.columbia.edu

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027
kathy@cs.columbia.edu

Abstract

Understanding contrastive opinions is a key component of argument generation. Central to an argument is the claim, a statement that is in dispute. Generating a counter-argument then requires generating a response in contrast to the main claim of the original argument. To generate contrastive claims, we create a corpus of Reddit comment pairs self-labeled by posters using the acronym FTFY (fixed that for you). We then train neural models on these pairs to edit the original claim and produce a new claim with a different view. We demonstrate significant improvement over a sequence-to-sequence baseline in BLEU score and a human evaluation for fluency, coherence, and contrast.

1 Introduction

In the Toulmin model (1958), often used in computational argumentation research, the center of the argument is the claim, a statement that is in dispute (Govier, 2010). In recent years, there has been increased interest in argument generation (Bilu and Slonim, 2016; Hua and Wang, 2018; Le et al., 2018). Given an argument, a system that generates counter-arguments would need to 1) identify the claims to refute, 2) generate a new claim with a different view, and 3) find supporting evidence for the new claim. We focus on this second task, which requires an understanding of contrast. A system that can generate claims with different views is a step closer to understanding and generating arguments (Apothloz et al., 1993). We build on previous work in automated claim generation (Bilu et al., 2015) which examined generating opposing claims via explicit negation. However, researchers also noted that not every claim has an exact opposite. Consider a claim from Reddit:

> Get employers out of the business, **pass universal single-payer healthcare**. (1)

This is an example of a policy claim - a view on what should be done (Schiappa and Nordin, 2013).

While negation of this claim is a plausible response (e.g. asserting there should be no change by stating *Do not get employers out of the business, do not pass universal healthcare*), negation limits the diversity of responses that can lead to a productive dialogue. Instead, consider a response that provides an alternative suggestion:

> Get employers out of the business, **deregulate and allow cross-state competition**. (2)

In Example 1, the speaker believes in an increased role for government while in Example 2, the speaker believes in a decreased one. As these views are on different sides of the political spectrum, it is unlikely that a single speaker would utter both claims. In related work, de Marneffe et al. (2008) define two sentences as contradictory when they are extremely unlikely to be true simultaneously. We thus define a contrastive claim as one that is *likely to be contradictory if made by the speaker of the original claim*. Our goal, then, is to develop a method for generating contrastive claims when explicit negation is not the best option. Generating claims in this way also has the benefit of providing new content that can be used for retrieving or generating supporting evidence.

In order to make progress towards generating contrastive responses, we need large, high-quality datasets that illustrate this phenomenon. We construct a dataset of 1,083,520 contrastive comment pairs drawn from Reddit using a predictive model to filter out non-contrastive claims. Each pair contains very similar, partially aligned text but the responder has significantly modified the original post. We use this dataset to model differences in views and generate a new claim given an original comment. The similarity within these pairs

Proceedings of NAACL-HLT 2019, pages 1756–1767
Minneapolis, Minnesota, June 2 - June 7, 2019.

allows us to use them as distantly labeled contrastive word alignments. The word alignments provide semantic information about which words and phrases can be substituted *in context* in a coherent, meaningful way.

Our contributions[1] are as follows:

1. Methods and data for *contrastive claim identification* to mine comment pairs from Reddit, resulting in a large, continuously growing dataset of 1,083,520 distant-labeled examples.
2. A crowd-labeled set of 2,625 comments each paired with 5 new contrastive responses generated by additional annotators.
3. Models for *generating contrastive claims* using neural sequence models and constrained decoding.

In the following sections, we describe the task methodology and data collection and processing. Next, we present neural models for contrastive claim generation and evaluate our work and present an error analysis. We then discuss related work in contrast/contradiction, argumentation, and generation before concluding.

2 Task Definition and Motivation

Previous work in claim generation (Bilu et al., 2015) focused on explicit negation to provide opposing claims. While negation plays an important role in argumentation (Apothloz et al., 1993), researchers found that explicit negation may result in incoherent responses (Bilu et al., 2015). Furthermore, recent empirical studies have shown that arguments that provide new content (Wachsmuth et al., 2018) tend to be more effective. While new concepts can be introduced in other ways by finding semantically relevant content, we may find it desirable to explicitly model contrast in order to control the output of the model as part of a rhetorical strategy, e.g. concessions (Musi, 2018). We thus develop a model that generates a contrastive claim given an input claim.

Contrastive claims may differ in more than just viewpoint; they may also contain stylistic differences and paraphrases, among other aspects. We thus propose to model contrastive claims by controlling for context and maintaining the same text between pairs of contrastive claims except for the contrastive word or phrase. Much of the previous work in contrast and contradiction has examined the relationship between words or sentences. In order to understand when words and phrases are contrastive in argumentation, we need to examine them *in context*. For example, consider the claim *Hillary Clinton should be president.* A reasonable contrastive claim might be *Bernie Sanders should be president.* (rather than the explicit negation *Hillary Clinton should not be president.*) In this context, Hillary Clinton and Bernie Sanders are contrastive entities as they were both running for president. However, for the claim *Hillary Clinton was the most accomplished Secretary of State in recent memory.* they would be unrelated. Consider also that we could generate the claim *Hillary Clinton should be senator.* This contrastive claim is not coherent given the context. Generating a contrastive claim then requires 1) identifying the correct substitution span and 2) generating a response with semantically relevant replacements.

While some contrastive claims are not coherent, there are often multiple plausible responses, similar to tasks such as dialogue generation. For example, *Donald Trump should be president* is just as appropriate as *Bernie Sanders should be president*. We thus treat this as a dialogue generation task where the goal is to generate a plausible response given an input context.

3 Data

In order to model contrastive claims, we need datasets that reflect this phenomenon.

3.1 Collection

We obtain training data by scraping the social media site Reddit for comments containing the acronym *FTFY*.[2] FTFY is a common acronym meaning “fixed that for you.”[3] FTFY responses (hereafter FTFY) are used to respond to another comment by editing part of the “parent comment” (hereafter parent). Most commonly, FTFY is used for three categories of responses: 1) expressing a contrastive claim (e.g. the parent is ***Bernie Sanders*** *for president* and the FTFY is ***Hillary Clinton*** *should be president*) which may be sarcastic (e.g. ***Ted Cruz*** *for president* becomes ***Zodiac***

---

[1] Data and code available at github.com/chridey/fixedthat

[2] https://en.wiktionary.org/wiki/FTFY

[3] https://reddit.zendesk.com/hc/en-us/articles/205173295-What-do-all-these-acronyms-mean

killer** for president*) 2) making a joke (e.g. *This Python library really **piques** my interest* vs. *This really ***py*ques** my interest*), and 3) correcting a typo (e.g. *This **peaks** my interest* vs. ***piques). In Section 3.2, we describe how we identify category 1 (contrastive claims) for modeling.

To obtain historical Reddit data, we mined comments from the site pushshift.io for December 2008 through October 2017. This results in 2,200,258 pairs from Reddit, where a pair consists of a parent and an FTFY. We find that many of the top occurring subreddits are ones where we would expect strong opinions (/r/politics, /r/worldnews, and /r/gaming).

3.2 Classification

To filter the data to only the type of response that we are interested in, we annotated comment pairs for contrastive claims and other types. We use our definition of contrastive claims based on contradiction, where both the parent and FTFY are a claim and they are unlikely to be beliefs held by the same speaker. A joke is a response that does not meaningfully contrast with the parent and commonly takes the form of a pun, rhyme, or oronym. A correction is a response to a typo, which may be a spelling or grammatical error. Any other pair is labeled as "other," including pairs where the parent is not a claim.

In order to identify contrastive claims, we selected a random subset of the Reddit data from prior to September 2017 and annotated 1993 comments. Annotators were native speakers of English and the Inter-Annotator Agreement using Kripendorff's alpha was 0.72. Contrast occurs in slightly more than half of the sampled cases (51.4%), with jokes (23.0%) and corrections (21.2%) comprising about one quarter each. We then train a binary classifier to predict contrastive claims, thus enabling better quality data for the generation task.

To identify the sentence in the parent that the FTFY responds to and derive features for classification, we use an edit distance metric to obtain sentence and word alignments between the parent comment and response. As the words in the parent and response are mostly in the same order and most FTFYs contain significant overlap with the parent response, it is possible to find alignments by moving a sliding window over the parent. A sample of 100 comments verifies that this approach yields exact word alignments in 75 comments and exact sentence alignments in 93.

Given these pairs of comments, we derive linguistic and structural features for training a binary classifier. For each pair of comments, we compute features for the words in the *entire comment span* and features from the *aligned phrases span* only (as identified by edit distance). From the *aligned phrases* we compute the *character edit distance* and *character Jaccard similarity* (both normalized by the number of characters) to attempt to capture jokes and typos (the similarity should be high if the FTFY is inventing an oronym or correcting a spelling error). From the *entire comment*, we use the *percentage of characters copied* as a low percentage may indicate a poor alignment and the *percentage of non-ASCII characters* as many of the jokes use emojis or upside-down text. In addition, we use features from GloVe (Pennington et al., 2014) word embeddings[4] for both the *entire comment* and *aligned phrases*. We include the *percentage of words in the embedding vocabulary* for both spans for both the parent and FTFY. The reason for this feature is to identify infrequent words which may be typos or jokes. We compute the *cosine similarity* of the average word embeddings between the parent and FTFY for both spans. Finally, we use *average word embeddings* for both spans for both parent and FTFY.

As we want to model the generation of new content, not explicit negation, we removed any pairs where the difference was only "stop words." The set of stop words includes all the default stop words in Spacy[5] combined with expletives and special tokens (we replaced all URLs and usernames). We trained a logistic regression classifier and evaluated using 4-fold cross-validation. We compare to a *character* overlap baseline where any examples with Jaccard similarity > 0.9 and edit distance < 0.15 were classified as non-contrastive. The goal of this baseline is to illustrate how much of the non-contrastive data involves simple or non-existent substitutions. Results are shown in Table 1. Our model obtains an F-score of 80.25 for an 8 point absolute improvement over the baseline.

3.3 Selection

After using the trained model to classify the remaining data, we have 1,083,797 Reddit pairs. We set aside 10,307 pairs from October 1-20, 2017 for

---

[4] We found the 50-dimensional Wikipedia+Gigaword embeddings to be sufficient

[5] spacy.io

Model	Precision	Recall	F-score
Majority	51.4	100	67.5
Baseline	67.75	77.19	72.16
LR	74.22	87.60	80.25

Table 1: Results of Identifying Contastive Claims

development and October 21-30 for test (6,773), with the remainder used for training. As we are primarily working with sentences, the mean parent length was 16.3 and FTFY length was 14.3.

The resulting test FTFYs are naturally occurring and so do not suffer from annotation artifacts. At the same time, they are noisy and may not reflect the desired phenomenon. Thus, we also conducted an experiment on Amazon Mechanical Turk[6] (AMT) to obtain additional gold references, which are further required by metrics such as BLEU (Papineni et al., 2002). We selected 2,625 pairs from the 10 most frequent categories[7] (see Table 2). These categories form a three-level hierarchy for each subreddit and we use the second-level, e.g. for /r/pokemongo the categories are "Pokémon", "Video Games", and "Gaming" so we use "Video Games." Before participating, each annotator was required to pass a qualification test - five questions to gauge their knowledge of that topic. For the movies category, one question we asked was whether for the sentence *Steven Spielberg is the greatest director of all time*, we could instead use *Stanley Kubrick* or *Paul McCartney*. If they passed this test, the annotators were then given the parent comment and "keywords" (the subreddit and three category levels) to provide additional context. We obtained *five* new FTFYs for each parent and validated them manually to remove obvious spam or trivial negation (e.g. "not" or "can't").

Category	Count	Category	Count
Video Games	1062	Basketball	116
Politics	529	Soccer	99
Football	304	Movies	88
Television	194	Hockey	60
World News	130	Baseball	55

Table 2: Comments for Mechanical Turk

4 Methods

Our goal of *generating* contrastive claims can be broken down into two primary tasks: 1) identifying the words in the original comment that should be removed or replaced and 2) generating the appropriate substitutions and any necessary context. Initially, we thus experimented with a modular approach by tagging each word in the parent and then using the model predictions to determine if we should copy, delete, or replace a segment with a new word or phrase. We tried the bi-directional LSTM-CNN-CRF model of Ma and Hovy (2016) and used our edit distance word alignments to obtain labels for copying, deleting, or replacing. However, we found this model performed slightly above random predictions, and with error propagation, the model is unlikely to produce fluent and accurate output. Instead, we use an end-to-end approach using techniques from machine translation.

4.1 Model

We use neural sequence-to-sequence encoder-decoder models (Sutskever et al., 2014) with attention for our experiments. The tokens from the parent are passed as input to a bi-directional GRU (Cho et al., 2014) to obtain a sequence of encoder hidden states h_i. Our decoder is also a GRU, which at time t generates a hidden state s_t from the previous hidden state s_{t-1} along with the input. When training, the input x_t is computed from the previous word in the gold training data if we are in "teacher forcing" mode (Williams and Zipser, 1989) and otherwise is the prediction made by the model at the previous time step. When testing, we also use the model predictions. The input word w_t may be augmented by additional features, as discussed in Section 4.2. In the baseline scenario $x_t = e(w_t)$ where e is an embedding. The hidden state s_t is then combined with a context vector h_t^*, which is a weighted combination of the encoder hidden states using an attention mechanism:

$$h_t^* = \sum_i \alpha_t^i h_i$$

To calculate α_i^t, we use the attention of Luong et al. (2015) as this encourages the model to select features in the encoder hidden state which correlate with the decoder hidden state, which we want because our input and output are similar. Our experiments on the development data verified this,

---

[6] We paid annotators the U.S. federal minimum wage and the study was approved by an IRB.

[7] Obtained from the snoopsnoo.com API

as Bahdanau attention (Bahdanau et al., 2015) performed worse. Attention is then calculated as:

$$\alpha_t^i = \frac{\exp(h_i^T s_t)}{\sum_{s'} \exp(h_{s'}^T s_t)}$$

Finally, we make a prediction of a vocabulary word w by using features from the context and decoder hidden state with a projection matrix W and output vocabulary matrix V:

$$P(w) = \text{softmax}(V \tanh(W[s_t; h_t^*] + b_w) + b_v)$$

We explored using a copy mechanism (See et al., 2017) for word prediction but found it difficult to prevent the model from copying the entire input.

4.2 Decoder Representation

Decoder Input: We evaluate two representations of the target input: as a sequence of **words** and as a sequence of **edits**. The sequence of words approach is the standard encoder-decoder setup. For the example parent *Hillary Clinton for president 2020* and FTFY *Bernie Sanders for president* we would use the FTFY without modification. Schmaltz et al. (2017) found success modeling error correction using sequence-to-sequence models by representing the target input as a sequence of edits. We apply a similar approach to our problem, generating a target sequence by following the best path in the matrix created by the edit distance algorithm. The new target sequence is the original parent interleaved with "DELETE-N tokens" that specify how many previous words to delete, followed by the newly generated content. For the same example, *Hillary Clinton for president 2020*, the modified target sequence would be *Hillary Clinton DELETE-2 Bernie Sanders for president 2020 DELETE-1*.

Counter: Kikuchi et al. (2016) found that by using an embedding for a length variable they were able to control output length via a learned mechanism. In our work, we compute a counter variable which is initially set to the number of new content words the model should generate. During decoding, the counter is decremented if a word is generated that is not in the source input (I) or in the set of stop words (S) defined in Section 3.2. The model uses an embedding $e(c_t)$ for each count, which is parameterized by a count embedding matrix. The input to the decoder state in this scenario is $x_t = e(w_t, c_t)$. At each time step, the count is computed by:

$$c_0 = |O \setminus (S \cup I)| \text{ or desired count}$$

$$c_{t+1} = \begin{cases} c_t - 1, & w_t \notin S \cup I \text{ and } c_t > 0 \\ c_t, & \text{otherwise} \end{cases}$$

where O is the set of gold output words in training.

For the parent comment *Hillary Clinton for president 2020* and FTFY *Bernie Sanders for president*, the decoder input is presented, with the time t in the first row of Table 3 and the inputs w_t and c_t in the second and third rows, respectively. At the start of decoding, the model expects to generate two new content words, which in this example it generates immediately and decrements the counter. When the counter reaches 0, it only generates stop or input words.

t	0	1	2	3	4
w_t	-	Bernie	Sanders	for	president
c_t	2	1	0	0	0

Table 3: Example of Counter

Unlike the controlled-length scenario, at test time we do not know the number of new content words to generate. However, the count for most FTFYs is between 1 and 5, inclusive, so we can exhaustively search this range during decoding. We experimented with predicting the count but found it to be inaccurate so we leave this for future work.

Subreddit Information: As the model often needs to disambiguate polysemous words, additional context can be useful. Consider the parent comment *this is a strange bug.* In a programming subreddit, a sarcastic FTFY might be *this is a strange feature.* However, in a Pokémon subreddit, an FTFY might be *this is a strange dinosaur* in an argument over whether Armaldo is a bug or a dinosaur. We thus include additional features to be passed to the encoder at each time step, in the form of an embedding g for each the three category levels obtained in Section 3.3. These embeddings are concatenated to the input word w_t at each timestep, i.e. $x_t = e(w_t, g_t^1, g_t^2, g_t^3)$.

4.3 Objective Function

We use a negative log likelihood objective function $\mathcal{L}_{NLL} = -\log \sum_{t \in 1:T} P(w_t^*)$, where w_t^* is the gold token at time t, normalized by each batch. We also include an additional loss term that uses the encoder hidden states to make a binary prediction over the input for whether a token will be copied or inserted/deleted. For the example from Section 4.2, the target for *Hillary Clinton for president 2020* would be *0 0 1 1 0*. This encourages the model to select features that indicate whether the encoder input will be copied to the output. We use a 2-layer multi-layer perceptron and a binary cross-entropy loss $\mathcal{L}_{BCE}$. The joint loss is then:

$$\mathcal{L} = \mathcal{L}_{NLL} + \lambda \mathcal{L}_{BCE}$$

4.4 Decoding

We use beam search for generation, as this method has proven effective for many neural language generation tasks. For the settings of the model that require a counter, we expand the beam by count m so that for a beam size k we calculate $k * m$ states.

Filtering: We optionally include a constrained decoding mode where we filter the output based on the counter; when $c_t > 1$ the end-of-sentence (EOS) score is set to $-\infty$ and when $c_t = 0$ the score of any word $w \in V \setminus (S \cup I)$ is set to $-\infty$. The counter c_t is decremented at every time step as in Section 4.2. In other words, when the counter is zero, we only allow the model to copy or generate stop words. When the counter is positive, we prevent the model from ending the sentence before it generates new content words and decrements the counter. The constrained decoding is possible with any combination of settings, with or without the counter embedding.

4.5 Hyper-parameters and Optimization

We used Pytorch (Paszke et al., 2017) for all experiments. We used 300-dimensional vectors for the word embedding and GRU layers. The count embedding dimension was set to 5 with $m = 5$ and $k = 10$ for decoding. The category embedding dimensions were set to 5, 10, and 25 for each of the non-subreddit categories. We also set $\lambda = 1$ for multi-task learning. We used the Adam optimizer (Kingma and Ba, 2015) with settings of $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and a learning rate of 10^{-3} decaying by $\gamma = 0.1$ every epoch. We used dropout (Srivastava et al., 2014) on the embeddings with a probability of 0.2 and teacher forcing with 0.5. We used a batch size of 100 with 10 epochs, selecting the best model on the development set based on perplexity. We set the minimum frequency of a word in the vocabulary to 4.

5 Results

For training, development, and testing we use the data described in Section 3.3. The test reference data consists of the Reddit FTFYs and the FTFYs generated from AMT. We evaluate our models using automated metrics and human judgments.

Automated metrics should reflect our joint goals of 1) copying necessary context and 2) making appropriate substitutions. To address point 1, we use BLEU-4 as a measure of similarity between the gold FTFY and the model output. As the FTFY may contain significant overlap with the parent, BLEU indicates how well the model copies the appropriate context. As BLEU reflects mostly span selection rather than the insertion of new content, we need alternative metrics to address point 2. However, addressing point 2 is more difficult due to the variety of possible substitutions, including named entities. For example, if the parent comment is *jaguars for the win!* and the gold FTFY is *chiefs for the win!* but the model produces *cowboys for the win!* (or any of 29 other NFL teams), most metrics would judge this response incorrectly even though it would be an acceptable response. Thus we present results using both automated metrics and human evaluation. As an approximation to address point 2, we attempt to measure when the model is making changes rather than just copying the input. To this end, we present two additional metrics - *novelty*, a measure of whether novel content (non-stop word) tokens are generated relative to the parent comment, and *partial match*, a measure of whether the novel tokens in the gold FTFY match any of the novel tokens in the generated FTFY. To provide a reference point, we find that the partial match between two different gold FTFYs (Reddit and AMT) was 11.4% and BLEU was 47.28, which shows the difficulty of automatic evaluation. The scores are lower than expected because the Reddit FTFYs are noisy due to the process in Section 3.2. This also justifies obtaining the AMT FTFYs.

Results are presented in Table 4. The baseline is a sequence-to-sequence model with attention. For other components, the counter embedding is referred to as "COUNT," the category/subreddit embeddings as "SUB," the sequence of edits as "EDIT," and the multi-task copy loss as "COPY." The models in the top half of the table use constrained decoding and those in the bottom half are unconstrained, to show the learning capabil-

			Reddit		AMT	
	Model	Novelty	BLEU-4	% Match	BLEU-4	% Match
Constrained	Baseline	79.88	18.81	4.67	40.14	10.06
	COUNT	89.69	22.61	4.72	47.55	12.55
	COUNT + SUB + COPY	**90.45**	**23.13**	**4.83**	**50.05**	**14.92**
	EDIT	64.64	16.12	3.37	35.48	7.33
	EDIT + COUNT + SUB + COPY	82.96	19.37	4.23	42.69	11.62
Unconstrained	Baseline	3.34	7.31	0.73	25.83	0.68
	COUNT	16.19	8.51	1.95	27.68	2.36
	COUNT + SUB + COPY	16.26	9.62	1.93	31.23	3.81
	EDIT	7.97	**35.41**	1.57	**74.24**	1.56
	EDIT + COUNT + SUB + COPY	**39.99**	32.59	**3.25**	67.56	**6.25**

Table 4: Automatic Evaluation

ities of the models. For each model we compute statistical significance with bootstrap resampling (Koehn, 2004) for the constrained or unconstrained baseline as appropriate and we find the COUNT and EDIT models to be significantly better for constrained and unconstrained decoding, respectively ($p < 0.005$).

Under constrained decoding, we see that the "COUNT + SUB + COPY" model performs the best in all metrics, although most of the performance can be attributed to the count embedding. When we allow the model to determine its own output, we find that "EDIT + COUNT" performs the best. In particular, this model does well at understanding which part of the context to select, and even does better than other unconstrained models at selecting appropriate substitutions. However, when we combine this model with constrained decoding, the improvement is smaller than for the other settings. We suspect that because the EDIT model often needs to generate a DELETE-N token before a new response, these longer-term dependencies are hard to capture with constrained decoding but easier if included in training.

We also conducted a human evaluation of the model output on the same subset of 2,625 examples described in Section 3.3. We performed an additional experiment on AMT where we asked annotators to rate responses on fluency, coherence, and contrast. **Fluency** is a measure of the quality of the grammar and syntax and the likelihood that a native English speaker would utter that statement. **Coherence** is a measure of whether the response makes sense, is semantically meaningful, and would be usable as a response to a claim. **Contrast** is a measure of how much the response contradicts the original comment. We specified that if the response is different but does not provide a contrasting view it should receive a low rating. Previous work (Bilu et al., 2015) used fluency, clarity/usability (which we combine into coherence), and opposition (where we use contrast).

Model	Fluency	Coherence	Contrast
Gold	4.34	4.26	3.01
Baseline	3.49	3.19	1.94
Constrained	3.46	3.32	2.53
Best	**3.52**	**3.46**	**2.87**

Table 5: Human Evaluation

We used a Likert scale where 5 is strongly agree and 1 is strongly disagree. We used the same data and qualification test from Section 3.3 for each category and used three annotators per example. We asked annotators to judge 4 different pairs: 3 model outputs and the gold Reddit[8] FTFYs for comparison. We include the baseline, the baseline with constrained decoding, and the best constrained model ("COUNT + SUB + COPY") according to BLEU and partial match. We verified that the annotators understood how to rate contrast by examining the distribution of responses: the annotators selected option 3 (neither) 15% of the time and preferred to select either extreme, 5 (21%) or 1 (27%). Results are presented in Table 5, showing a clear preference for the best model. Note the degradation in fluency for the constrained baseline, as the model is prevented from generating the EOS token and may repeat tokens up to the

---

[8] We did not evaluate the AMT FTFYs as these were generated by the same pool of annotators.

Parent: ah yes the wonders of the free market
Model: ah yes the wonders of government intervention
Parent: i know that this is an unofficial mod , but xp is the best os for this machine
Model: linux is the best os for this machine
Parent: that ’s why it ’s important to get all your propaganda from infowars and brietbart
Model: propaganda from fox news outlets

Table 6: Model Output

maximum length.

6 Qualitative Analysis

We provide three examples of the model output in Table 6 with the first and third from the News and Politics category, demonstrating how the model handles different types of input. In the first example, the contrast is between allowing markets to regulate themselves versus an increased role of government. In the second example, the contradiction is due to the choice of operating system. In the third (invalid) example, the model responds to a sarcastic claim with another right-wing news organization; this response is not a contradiction since it is plausible the original speaker would also utter this statement.

6.1 Error Analysis

We conduct an error analysis by selecting 100 responses where the model did not partially match any of the 6 gold responses and we found 6 main types of errors. One error is the model identifying an **incorrect substitution span** while the human responses all selected a different span to replace. We noticed that this occurred 5 times and may require world knowledge to understand which tokens to select. For example, in response to the claim *Hillary Clinton could have been president if not for robots*, the model generates *Donald Trump* in place of *Hillary Clinton*, whereas the gold responses generate *humans / votes / Trump’s tweets* in place of *robots*. Another type of error is when the responses are not coherent with the parent and the language model instead determines the token selection based on the **most recent context** (11 cases). For example, given the claim *bb-8 gets a girlfriend and poe still does n’t have a girlf :’)* the Reddit FTFY has *boyf* instead of *girlf* whereas the model generates ... *and poe still does n’t have a clue what i ’m talking about .* We also found examples where the model chose poorly due to unfiltered jokes or **errors in the training data** (12 in total). In 15 cases, due to the constrained decoding the model **repeated** a word until the maximum length or appended an incoherent phrase. For the most common error, the model made a substitution that **was not contrasting** as in Table 6 (19 examples). Finally, we found 38 of the samples were **valid** responses, but did not match the gold, indicating the difficulty of automatic evaluation. For example, in response to the claim *Nintendo is the only company that puts customers over profits*, the model replaces *Nintendo* with *Rockstar* (both video game companies) while the gold FTFYs had other video game companies.

7 Related Work

Understanding contrast and contradiction is key to argumentation as it requires an understanding of differing points-of-view. Recent work examined the negation of claims via explicit negation (Bilu et al., 2015). Other work investigated the detection of different points-of-view in opinionated text (Al Khatib et al., 2012; Paul et al., 2010). Wachsmuth et al. (2017; 2018) retrieved arguments for and against a particular stance using online debate forums. In non-argumentative text, researchers predicted contradictions for types such as negation, antonyms, phrasal, or structural (de Marneffe et al., 2008) or those that can be expressed with functional relations (Ritter et al., 2008). Other researchers have incorporated entailment models (Kloetzer et al., 2013) or crowdsourcing methods (Takabatake et al., 2015). Contradiction has also become a part of the natural language inference (NLI) paradigm, with datasets labeling contradiction, entailment, or neutral (Bowman et al., 2015a; Williams et al., 2018). The increase in resources with contrast and contradiction has resulted in new representations with contrastive meaning (Chen et al., 2015; Nguyen et al., 2016; Vulić, 2018; Conneau et al., 2017). Most of this work has focused on identifying contrast or contradiction while we aim to generate contrast. Furthermore, while contradiction and contrast are present in these corpora, we obtain distant-labeled alignments for contrast at the word and phrase level. Our dataset also includes contrastive concepts and entities while other corpora primarily contain antonyms and explicit negation.

Contrast also appears in the study of stance, where the opinion towards a target may vary. The SemEval 2016 Stance Detection for Twitter task (Mohammad et al., 2016) involved predicting if a tweet favors a target entity. The Interpretable Semantic Similarity task (Agirre et al., 2016) called to identify semantic relation types (including opposition) between headlines or captions. Target-specific stance prediction in debates is addressed by Hasan and Ng (2014) and Walker et al. (2012). Fact checking can be viewed as stance toward an event, resulting in research on social media (Lendvai and Reichel, 2016; Mihaylova et al., 2018), politician statements (Vlachos and Riedel, 2014), news articles (Pomerleau and Rao, 2017), and Wikipedia (Thorne et al., 2018).

In computational argumentation mining, identifying claims and other argumentative components is a well-studied task (Stab and Gurevych, 2014). Daxenberger et al. (2017) and Schulz et al. (2018) developed approaches to detect claims across across diverse claim detection datasets. Recently, a shared task was developed for argument reasoning comprehension (Habernal et al., 2018). The best system (Choi and Lee, 2018) used models pre-trained on NLI data (Bowman et al., 2015b), which contains contradictions. While this work is concerned with identification of argumentative components, we propose to generate new claims.

In the field of argument generation, Wang and Ling (2016) train neural abstractive summarizers for opinions and arguments. Additional work involved generating opinions given a product rating (Wang and Zhang, 2017). Bilu and Slonim (2016) combine topics and predicates via a template-based classifier. This work involves the generation of claims but in relation to a topic. Other researchers generated political counter-arguments supported by external evidence (Hua and Wang, 2018) and generating argumentative dialogue by maximizing mutual information (Le et al., 2018). This research considers end-to-end argument generation, which may not be coherent, whereas we focus specifically on contrastive claims.

8 Conclusion

We presented a new source of over 1 million contrastive claim pairs that can be mined from social media sites such as Reddit. We provided an analysis and models to filter noisy training data from 49% down to 25%. We created neural models for generating contrastive claims and obtained significant improvement in automated metrics and human evaluations for Reddit and AMT test data.

Our goal is to incorporate this model into an argumentative dialogue system. In addition to generating claims with a contrasting view, we can also retrieve supporting evidence for the newly-generated claims. Additionally, we plan to experiment with using our model to improve claim detection (Daxenberger et al., 2017) and stance prediction (Bar-Haim et al., 2017). Our model could be used to generate artificial data to enhance classification performance on these tasks.

To improve our model, we plan to experiment with retrieval-based approaches to handle low-frequency terms and named entities, as sequence-to-sequence models are likely to have trouble in this environment. One possibility is to incorporate external knowledge with entity linking over Wikipedia articles to find semantically-relevant substitutions.

Another way to improve the model is by introducing controllable generation. One aspect of controllability is intention; our model produces contrastive claims without understanding the view of the original claim. Category embeddings partially address this issue (some labels are "Liberal" or "Conservative"), but labels are not available for all views. Going forward, we hope to classify the viewpoint of the original claim and then generate a claim with a desired orientation. Furthermore, we hope to improve on the generation task by identifying the types of claims we encounter. For example, we may want to change the target of the claims in some claims but in others change the polarity.

We also plan to improve the dataset by improving our models for contrastive pair prediction to reduce noise. Finally, we hope that this dataset proves useful for related tasks such as textual entailment (providing examples of contradiction) and argument comprehension (providing counter-examples of arguments) or even unrelated tasks like humor or error correction.

Acknowledgments

We would like to thank the AMT annotators for their work and the anonymous reviewers for their valuable feedback. The authors also thank Alyssa Hwang for providing annotations and Smaranda Muresan, Chris Kedzie, Jessica Ouyang, and Elsbeth Turcan for their helpful comments.

References

Eneko Agirre, Aitor Gonzalez-Agirre, Inigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2016. Semeval-2016 task 2: Interpretable semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 512–524. Association for Computational Linguistics.

Khalid Al Khatib, Hinrich Schütze, and Cathleen Kantner. 2012. Automatic detection of point of view differences in wikipedia. In *Proceedings of COLING 2012*, pages 33–50. The COLING 2012 Organizing Committee.

Denis Apothloz, Pierre-Yves Brandt, and Gustavo Quiroz. 1993. The function of negation in argumentation. *Journal of Pragmatics*, 19:23–38.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain. Association for Computational Linguistics.

Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: Why, how and when. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 84–93. Association for Computational Linguistics.

Yonatan Bilu and Noam Slonim. 2016. Claim synthesis via predicate recycling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 525–530, Berlin, Germany. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015b. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. 2015. Revisiting word embedding for contrasting meaning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 106–115. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

HongSeok Choi and Hyunju Lee. 2018. Gist at semeval-2018 task 12: A network transferring inference knowledge to argument reasoning comprehension task. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 773–777. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066. Association for Computational Linguistics.

Trudy Govier. 2010. *A Practical Study of Argument*. Cengage Learning, Wadsworth.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1930–1940. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 751–762. Association for Computational Linguistics.

Xinyu Hua and Lu Wang. 2018. Neural argument generation augmented with externally retrieved evidence. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Melbourne, Australia. Association for Computational Linguistics.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, Motoki Sano, and Kiyonori Ohtake. 2013. Two-stage method for large-scale acquisition of contradiction pattern pairs using entailment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 693–703. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Dieu-Thu Le, Cam Tu Nguyen, and Kim Anh Nguyen. 2018. Dave the debater: a retrieval-based and generative argumentative dialogue agent. In *Proceedings of the 5th Workshop on Argument Mining*, pages 121–130. Association for Computational Linguistics.

Piroska Lendvai and Uwe Reichel. 2016. Contradiction detection for rumorous claims. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics (ExProM)*, pages 31–40. The COLING 2016 Organizing Committee.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047, Columbus, Ohio. Association for Computational Linguistics.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact checking in community forums. *CoRR*, abs/1803.03178.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41. Association for Computational Linguistics.

Elena Musi. 2018. How did you change my view? a corpus-based study of concessions argumentative role. *Discourse Studies*, 20(2):270–288.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Michael Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 66–76. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Dean Pomerleau and Delip Rao. 2017. Fake news challenge.

Alan Ritter, Stephen Soderland, Doug Downey, and Oren Etzioni. 2008. It's a contradiction – no, it's not: A case study using functional relations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 11–20. Association for Computational Linguistics.

E. Schiappa and J.P. Nordin. 2013. *Argumentation: Keeping Faith with Reason*. Pearson Education.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813, Copenhagen, Denmark. Association for Computational Linguistics.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. *CoRR*, abs/1804.04083.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Yu Takabatake, Hajime Morita, Daisuke Kawahara, Sadao Kurohashi, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2015. Classification and acquisition of contradictory event pairs using crowdsourcing. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 99–107. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Stephen Toulmin. 1958. *The Uses of Argument*. Cambridge At the University Press.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22. Association for Computational Linguistics.

Ivan Vulić. 2018. Injecting lexical contrast into word vectors by guiding vector space specialisation. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 137–143. Association for Computational Linguistics.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark. Association for Computational Linguistics.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251. Association for Computational Linguistics.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596. Association for Computational Linguistics.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Zhongqing Wang and Yue Zhang. 2017. Opinion recommendation using a neural model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1627–1638. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Box of Lies®: Multimodal Deception Detection in Dialogues

Felix Soldner
Department of Security
and Crime Science
University College London
felix.soldner@ucl.ac.uk

Verónica Pérez-Rosas and **Rada Mihalcea**
Department of Computer Science
and Engineering
University of Michigan, Ann Arbor
{vrncapr,mihalcea}@umich.edu

Abstract

Deception often takes place during everyday conversations, yet conversational dialogues remain largely unexplored by current work on automatic deception detection. In this paper, we address the task of detecting multimodal deceptive cues during conversational dialogues. We introduce a multimodal dataset containing deceptive conversations between participants playing The Tonight Show Starring Jimmy Fallon® Box of Lies game, in which they try to guess whether an object description provided by their opponent is deceptive or not. We conduct annotations of multimodal communication behaviors, including facial and linguistic behaviors, and derive several learning features based on these annotations. Initial classification experiments show promising results, performing well above both a random and a human baseline, and reaching up to 69% accuracy in distinguishing deceptive and truthful behaviors.

1 Introduction

Deception occurs often during dialogues, but until now this setting has received little attention from the research community (Tsunomori et al., 2015). In this paper, we explore verbal, nonverbal, and conversational dialog cues between contestants playing the Box of Lies game in The Tonight Show Starring Jimmy Fallon® tv show. In the game, participants try to guess whether an object description provided by their opponent is deceptive or not. The game scenario provides a rich environment where we can explore several aspects of deceptive behavior occurring during conversations. First, it allows us to study conversational deception in the presence of multiple modalities such as verbal and non-verbal behaviors. Second, it provides observable assessments of participant's honesty, which is usually an important challenge during deception research. Third, since participants experience the pressure to win the game in front of a big audience, it presumably presents an environment with high stakes.[1]

Recent work on multimodal deception detection has already shown the importance of verbal and non-verbal behaviors during the automatic identification of deceit (Abouelenien et al., 2017a). Following this line of work, our main contribution consists of investigating whether such modalities can also be leveraged to predict deception in a conversational dialog as well as exploring whether the dialogue setting adds meaningful information to the other modalities to potentially increase the classification performance. Based on earlier work (Perez-Rosas et al., 2014; Mihalcea et al., 2013), we hypothesize that **(1)** including dialogue features in addition to other multimodal features (language and facial expressions) while training a classifier increases the prediction performance; **(2)** automatic classification of truthful and deceptive behavior is better than random guessing (50% for equal class sizes); and **(3)** automatic classification of truthful and deceptive responses is more accurate than human judgments (based on performance of participants in the dataset).

To address these hypotheses, we first generate a dataset containing verbal and non-verbal annotations of deceptive and truthful interactions between the game participants. Next, we derive linguistic, visual, and dialog cues based on our annotations for the verbal and non-verbal components of the dataset. The features are then used to conduct several learning experiments under different scenarios that attempt to distinguish between deceptive and truthful utterances either by focusing on the statements generated by one participant at a time (the game's host or the guest), or by address-

---

[1] To our knowledge, these conversations are not scripted.

Proceedings of NAACL-HLT 2019, pages 1768–1777
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing them all together.

Our initial experiments show that language, as well as behavioral and dialog features, carry meaningful information. Moreover, the automatic classification of deception can be performed with an accuracy that is better than random guessing and ourperforms human judgments.

2 Related Work

To tackle the problem of reliably detecting deception, researchers have applied various forms of automated deception detection methods that rely on machine learning approaches, which are able to incorporate a variety of behavioral cues from text, audiovisual or physiological data sources (Ott et al., 2011; Fornaciari and Poesio, 2013; Mihalcea and Strapparava, 2009; Abouelenien et al., 2016).

Many studies focused on text-based classification, detecting false online reviews (Ott et al., 2013) or deceptive transcribed statements from court hearings (Fornaciari and Poesio, 2013). Other studies utilized visual cues such as facial expressions or other body movements to detect deception (Meservy et al., 2005). These methods already show success in identifying deceptive behavior using individual modalities. In addition, recent approaches, which combine multiple modalities are able to further boost classification performances (Abouelenien et al., 2017b; Perez-Rosas et al., 2015).

However, multimodal approaches have not utilized the dialogue dimension in combination with other modalities as of yet. The dialogue dimension captures the interaction between two individuals and how they react to each other. One previous study investigated such an interaction, in which the researchers examined question types and their behavioral effect on participants (Tsunomori et al., 2015). Findings of the study showed that specific questions led to more salient deceptive behavior patterns in participants. This increase in feature salience resulted in better deception detection performances.

The interaction between two individuals in deceptive conversations was also investigated by Hancock et al. (2004), who examined deception at the linguistic level. Participants, who were unaware of receiving a deceptive message, produced more words, sense terms and asked more questions as compared to when they received a truthful message. In a similar setting, in which two participants engaged in a question-response task, Levitan et al. (2018) examined linguistic, gender and native language differences. They found significant variations in these features for truthful and deceptive responses. The experimenters utilized these variations in an automated classification task and reached up to 72% accuracy.

These studies show that a focus on the linguistic level and the interaction between individuals can have a beneficial effect on detecting deceit. Other studies examined non-verbal behavior. In an experiment, Sen et al. (2018) video-recorded conversations between participants in an interrogation game and examined participant's facial expressions. The results showed that interrogators exhibited different facial expressions when they were lied to as opposed to when they were told the truth. In a different approach, Yu et al. (2015) observed head movements and facial expressions between two individuals. The authors established normalized non-verbal patterns, which enabled them to capture interactional synchrony. This allowed them to successfully discriminate between truths and lies in the experiment.

Overall, this previous research demonstrates that capturing verbal or non-verbal interactions can convey meaningful information about deceit, which can be leveraged for multimodal deception detection.

3 Dataset of Deceptive Conversations

To explore the role played by conversation dynamics in deceptive behaviors, we collected conversations where participants acted deceptively. Specifically, we opted for identifying public sources where the veracity or falsehood of conversation participants is known.

In the game show Box of Lies, which is part of the late-night talk show The Tonight Show Starring Jimmy Fallon, these labels are known. The host (Jimmy Fallon) and his guest play the game Box of Lies, where participants take turns to play the game. During the game, when is their turn, the participants pick a box (from among nine available boxes) that contains an object they have to describe to their opponent. The object is hidden from the opponent through a separation wall between the two contestants. Participants sit opposite to each other and see their upper body and face through a cut hole in the separation wall. The opponent must guess if the provided description is truthful or not. The participant with the best of three guesses wins the game.

This setup allows us to observe verbal and non-verbal behavior exhibited by the participants during the dialogue interaction. In order to better capture multimodal behavioral cues of deception throughout the conversation, we decided to conduct annotations at utterance-level. We thus built a rich multimodal dataset containing verbal and non-verbal annotations for 1049 utterances, which is used in the experiments reported in this paper. The data collection and annotation process are described below.

3.1 Data Collection

We search for publicly available Box of Lies videos on the YouTube platform.[2] We collected 25 videos that are currently available in the show video-feed. The full set consists of 2 hours and 24 minutes of video. The average length of a video is six minutes and contains around three rounds of the game (this varies depending on the score and on whether additional time was available for extra rounds). Each video features a different guest and Jimmy Fallon, resulting in 26 unique participants, with 6 of them being males and 20 females.

3.2 Annotation of Multimodal Communication Behaviors

To capture the non-verbal behavior of the participants, each video is initially segmented based on the conversation turn-taking and annotated with the help of the ELAN software (Wittenburg et al., 2006). ELAN provides a multimodal annotation platform on which audiovisual recordings are annotated in a multi-level tier structure. In our case, we defined the following structure to annotate both types of behavior: host verbal, host non-verbal, guest verbal, and guest non-verbal.

3.2.1 Non-verbal Behaviors

To annotate facial and communication behaviors, we use MUMIN, a multimodal coding scheme that is used to study gestures and facial displays in interpersonal communication with a focus on the role played by multimodal expressions for feedback, turn management, and sequencing (Allwood et al., 2005). Given the nature of the video-conversations being depicted in our dataset, which show the face and upper bodies of the participants and their interaction, we focus our annotations on facial and conversational behavior. These choices are motivated by previous research showing that different expressions for truthful and deceptive behaviors are present (DePaulo et al., 2003) in the eyes and mouth regions, as well as studies on the role of conversational involvement in deceptive interactions (Burgoon et al., 1999).

Facial behaviors. We annotate the categories for visual cues and behaviors of eyebrows, eyes, gaze, mouth-openness, mouth-lips, head, and the general face. Each of the categories takes on one of several mutually exclusive behavior values. Table 1 shows the frequencies of all facial expressions included in this set. In the table, we observe a slightly unequal representation of behavioral categories (e.g., head movements are observed more often than other facial expressions). This is mainly attributed to camera angle changes during the videos causing participant's faces to be only partly or not visible, thus restricting the behavioral coding. The annotated values reflect the most dominant observed behavior in that time segment of the video.

Two annotators coded the videos, and after the first three videos, the inter-annotator agreement was measured by calculating the Kappa score, to ensure accurate coding. If the agreement was below Kappa (weighted) = 0.45 in any category, this category was discussed to identify and reconcile differences in the coding strategy. The annotators re-coded the videos individually and compared them again. This process was repeated until the desired agreement was reached (above .40 for each category). In most cases, we repeated the process only twice, except for the "feedback receiving" and "feedback eliciting" categories which were discussed three times. Table 2 shows the final Kappa score for each category.

3.2.2 Speaker's Veracity

In the full video set, participants play 68 rounds (29 truthful and 39 deceptive). Occasionally, deceptive rounds also contain truthful statements, in which contestants describe parts of the object truthfully, but other parts deceptively, turning the overall description into a lie. For example, a contestant might say: "I have before me, a green lobster on a plate." In truth, the object is a red lob-

---

[2] The videos are originally produced by NBC and retrieved from Youtube. We consider that using YouTube videos for research purposes falls under the "fair use" clause, which is stated on: `https://www.youtube.com/intl/en-GB/yt/about/copyright/fair-use/`

Label	Count
General face	
Smile	411
Neutral	342
Other	100
Laughter	83
Scowl	42
Eyebrows	
Neutral/Normal	531
Raising	320
Frowning	76
Other	39
Mouth-Lips	
Retracted	279
Neutral	267
Corners up	261
Other	102
Protruded	46
Corners down	21

Label	Count
Head	
Neutral/still	320
Waggle	292
Side-turn	242
Single Nod (Down)	165
Move Forward	153
Repeated Nods (Down)	122
Move Backward	117
Single Tilt (Sideways)	115
Single Jerk (Backwards Up)	78
Shake (repeated)	75
Repeated Tilts (Sideways)	18
Other	15
Single Slow Backwards Up	10
Repeated Jerks (Backwards Up)	7

Label	Count
Mouth-Openness	
Open mouth	763
Closed mouth	212
Other	3
Gaze	
Towards interlocutor	674
Towards object	148
Down	37
Towards audience	36
Sideways	35
Other	34
Eyes	
Neutral/Open	465
Closing-repeated	203
Closing-both	166
Other	121
Exaggerated Opening	8
Closing-one	4

Table 1: Frequency counts for participants' face, head and mouth annotations.

Figure 1: Sample screenshots of truthful and deceptive behavior from the original videoclips; left-top (truthful) : Eyebrows-raising, Eyes-open; left-bottom (truthful): Eyebrows-neutral, Eyes-open; right-top (deceptive): Eyebrows-frowning, Eyes-closing (both); right-bottom (deceptive): Eyebrows-raising, Eyes-closing (both).

ster on a plate. The description contains truthful and deceptive aspects, but it is considered to be a deceptive round since the main purpose of the statement is to deceive. This fine-grained distinction is captured during the annotation of behaviors, described below, which allows us to obtain more precise veracity labels of the behavior. In our example, the behavior associated with the description "green" is labeled as deceptive, whereas all the other behaviors are labeled as being truthful.

To enable this annotation, we further process our initial turn-by-turn segmentation to obtain spoken segments by either of the participants. We then code the veracity (i.e., truthful or deceptive) for each verbal statement of the participants. During the veracity coding, we assume that the behavior is always deceptive unless the verbal description indicates otherwise (i.e., accurate description of the object), as the general goal of each participant is to deceive their opponent. The final distribution of these annotations is 862 utterances labeled as deceptive, and 187 as truthful. Figure 1 shows examples of truthful and deceptive behaviors in the dataset.

3.3 Transcriptions

In order to include linguistic features in our analyses, we first transcribe the participants' conversations. To obtain transcriptions, we first extract the

	Kappa	
Category	Host	Guest
General Face	0.75	0.70
Eyebrows	0.51	0.70
Eyes	0.56	0.92
Gaze	0.45	0.74
Mouth-Openness	0.64	0.47
Mouth-Lips	0.79	0.53
Head	0.60	0.55
Feedback receiving	0.47	0.72
Feedback eliciting	0.73	0.46
Average	0.61	0.64

Table 2: Inter-annotator agreement

	Truthful	Deceptive	Total
Host	749	4211	4960
Guests	748	2496	3244
Total	1497	6707	8204

Table 3: Distribution of words for all transcriptions

audio of the corresponding video clip and slice it based on the verbal annotation time-stamps. For this task, we use Pympi (Lubbers and Torreira, 2013) and Ffmpy (Developers, 2016). We transcribe the resulting audio clips using Amazon Mechanical Turk (AMT), a crowd-sourcing platform. We notice that some of the clips include brief interruptions among speakers, thus we ask the AMT workers to transcribe only the speech of the main speaker in the audio clip. After we collect all transcriptions, we proofread them to avoid mistakes such as double transcriptions and remove additional characters or descriptions (e.g. “person 1”, clapping, [pause]). The final distribution of all the words from the transcriptions is shown in Table 3. Example utterances of truthful and deceptive statements are displayed in Table 4.

4 Methodology

Gathering data from different modalities creates the need to combine them into a coherent feature set, which can be utilized by machine learning classifiers. The following subsections describe how we generate features based on our annotations for the verbal and non-verbal behavior components of the dataset. These features are then used to train and test the classifiers in our experiments.

4.1 Linguistic Features

We derive various linguistic features from the transcriptions of the participants’ speech, which include: unigrams, psycholinguistic features, part of speech features, and word embedding features.

Unigrams. These features are created with bag-of-words representations of all transcriptions from the guests and the host. The unigrams are represented using their frequencies.

Psycholinguistic Features. These features are created with the help of the Linguistic Inquiry and Word count Lexicon (Version 2015) (Pennebaker et al., 2007). They represent 80 different classes of words, which can be attributed to different psychological dimensions. The features display the frequencies of occurrences of classes, derived from the occurrences of words, attributed to each class. The lexicon has been successfully used in previous work for automatic deception detection (Ott et al., 2013; Mihalcea et al., 2013).

Part of Speech tags (PoS). These features are created by obtaining PoS-tagging of transcripts. They capture the grammatical and syntactical structure of the utterances of the transcriptions (e.g., noun, verb, adjective). Features display the distribution of these categories in percentage for each utterance.

Word Embeddings. These features are obtained using Word2Vec by creating vector representations of the words in the transcriptions. By training word representations based on other words occurring in the same context, these features capture similarities of words next to each other and in context. Together, all words are represented in a vector space in which similar words lay closer to each other as compared to dissimilar words.

4.2 Non-verbal Behavioral Features

These features are generated from the non-verbal behaviors described in Section 3.2 and represented as percentages. Specifically, the different behavioral values for a category (e.g., Head) in a verbal utterance are counted and represented as percentages. For example, a verbal utterance might last for one minute and during that time head movements might take several different values, such as *side-turn* (20 sec.), *shake* (30 sec.), and *single nod* (10 sec.). These times are transformed into percentages and the category *head* then consist of 33.33% *side-turn*, 50% *shake*, and 16.67% *single nod* during the one-minute utterance. In this manner, each facial area designates its percentage representation of behavioral values, which add up to 100%. In case a behavior cannot be fully attributed to one of the possible actions through the verbal statement, left-over percentages are assigned to

Participant	**Truthful**	**Deceptive**
Host	”In a, no. In a costume and also inside the box, a bunch of Hershey kisses.”	”Ever heard of a boy band called Backstreet Boys?”
Guest	”Ok, it is, um, a rubiks cube inside jello.”	”Okey. Its a toaster oven. Ohhh ha ha ha ha”

Table 4: Examples of utterances from the transcriptions

none, representing the lack of occurrence of a behavioral action in its category. This transformation is performed for all seven different facial areas we have annotated, including General Facial Expressions, Eyebrows, Eyes, Gaze, Mouth-Openness, Mouth-Lips, and Head.

Our non-verbal behavior feature set thus consists of all the facial expressions or head movements expressed as the percentage of times they occur during a speaker's utterance. Possible attributes for each of the seven categories can be found in Table 1.

4.3 Dialogue Features

We derive dialogue-based features by exploring verbal and non-verbal aspects of the interaction between participants that are related to deceptive behavior. The features attempt to capture deception cues the speaker's exhibited during their conversation prior to the current utterance. These features are obtained as follows:

Deception Changes. These features include the count of truthful and deceptive utterances up to the current utterance. We also aggregate the counts of deceptive and truthful utterances to represent the participation of the speaker during the conversation.

Non-verbal Changes. These features capture how facial displays differ between consequent utterances. We calculate these features by subtracting the numeric vectors representing the non-verbal behavior during the current utterance from the previous utterance.

4.4 Data Alignment

In order to attribute the corresponding non-verbal behaviors to verbal utterances for later classification tasks, each behavior receives a veracity label (truthful or deceptive) individually. The veracity label that overlaps with a behavior for more than 50% of its span, it is associated with that behavior. The overlap is determined by comparing the time stamp of the behavior and the veracity annotation, which are obtained from the ELAN files. Table 5 displays the distribution of these feature sets.

Participant	Lies	Truths	All
Guests	394	101	495
Host	468	85	553
All	862	187	1048

Table 5: Class distribution for host and guest features

5 Human Performance

In order to evaluate the automated methods and compare them to human performance, we establish a human baseline, representing how well humans guess deceptive and truthful behavior correctly. Since the game show Box of Lies is already set up in a way that participants have to guess if their opponent is lying or telling the truth, their performance serves as a baseline.

Thus, we use their assessments to obtain a confusion matrix showing their correct and incorrect guesses. We calculate their performance in terms of accuracy, which reflects the proportion of correctly categorized descriptions of all object descriptions; precision, which reflects the proportion of correctly identified descriptions in one classified category; recall, which reflects the proportion of correctly identified descriptions out of all the object descriptions truly belonging to that category; and f1-score, which reflects the weighted average of precision and recall in that category.

Human performance is shown in Table 6. Since participants tell 39 deceptive and 29 truthful descriptions in total, the distribution is slightly uneven, resulting in a baseline of 0.57 in detecting a lie. Considering this, participants and the overall accuracy is almost equal to the accuracy of random guessing. This supports earlier findings that humans are almost only as good as chance (Bond and DePaulo, 2006).

Results for each class (detecting truthful or deceptive descriptions) show that participants are better at detecting truthful descriptions. This could be based on the truth bias, which describes the phenomenon according to which people generally tend to believe others (Levine et al., 1999).

		Lie			Truth		
	Acc.	P	R	F	P	R	F
Guests	55%	0.67	0.26	0.38	0.51	0.86	0.64
Host	58%	0.17	0.17	0.17	0.72	0.72	0.72
All	56%	0.47	0.24	0.32	0.58	0.79	0.67

Table 6: Human performances in the game "Box of Lies"; Acc. = Accuracy; P = Precision; R = Recall; F = F1; Baseline (detecting a lie) = 0.57

6 Experiments

During our experiments, we use a Random Forest classifier. We perform all the classification experiments with the python package Scikit-learn (Pedregosa et al., 2011) using the standard settings for the model parameters. All classifiers are evaluated using five-fold cross-validation. During our experiments we focus on three scenarios:

(1) How well can we distinguish between truthful and deceptive utterances in the dataset? In this scenario, we explore whether the different features we propose can capture differences between truthful and deceptive behavior, regardless of the speaker. Note that in this scenario, a significant fraction of the data comes from the same speaker (host).

(2) How well can we distinguish between truthful and deceptive behaviors elicited by the guests? In this experiment, we consider the subset of deceptive and truthful utterances produced by the guests in our dataset. Again, we test our different feature sets in the prediction of truthful and deceptive behavior, but this time we focus on learning deceptive patterns from several individuals, which might exhibit different verbal and non-verbal behaviors.

(3) How well can we distinguish between truthful and deceptive behaviors exhibited by the host? In this scenario, we explore whether the availability of data by the same individual can help to improve the detection of deceptive behavior. In other words, this experiment builds personalized deception models for the host using the different sets of features representing verbal and non-verbal behavior.

For each scenario, we test classifiers with features derived from the different verbal and non-verbal modalities as well as features that represent the interaction between participants (described in Sections 4.1, 4.2 and 4.3). We test the predictive power of each feature set individually and we also build joint models that combine all feature sets. The classifiers performance is evaluated in terms of accuracy, precision, recall, and f1-score.

An important challenge during the experiments is that the nature of our dataset leads to a high unbalance between the truthful and deceptive classes, as shown in Table 5. During our experiments, the imbalance of the data is tackled by applying down-sampling to the deceptive class (Oliphant, 2006). This ensures an equal distribution of each label and results in a baseline of 0.50 in all scenarios. The results for verbal, non-verbal, dialog features, and their combination for each scenario are shown in Table 7.

Overall, our experiments show the benefit of combining multiple sources of information on this task, with accuracies well above the baseline and a noticeable accuracy improvement when using all feature sets.

7 Discussion

The different classification performances show that adding information from several modalities helps to increase the accuracy of the detection system. Not surprisingly, the linguistic modality shows the best performance among single modalities (Scenarios 1 and 2). More interestingly, the non-verbal modality is the second best indicator of deception, despite a significant amount of facial occlusions present in our dataset.[3] Furthermore, this finding is in line with other work on multi-modal deception detection also showing that gestures are a reliable indicator of deception (Perez-Rosas et al., 2015).

In addition, we generate learning curves for each modality in scenario 1 (figure 2). The curves show that when training with 50 - 60% of the data, the classifier starts to improve upon the (guessing) baseline. The ascending trend does not seem to level off, even with the entire dataset, indicating that the classifier might benefit from more data.

Our experiment on exploring the classification of deceptive behaviors from the host (scenario 3) also lead to interesting insights. First, the linguistic modality is the weakest since it obtained the lowest performance in both classes. As the difference in f-score values shows, the host does not appear to use significantly different language while telling lies or truths, at least not at the lexical and semantic level, as captured by our linguistic features. Second, his non-verbal behavior does

---

[3] Facial occlusions are mainly attributed to changes in camera angles occurring during the videos

			Lie			Truth		
Scenario	Features	Acc.	P	R	F	P	R	F
(1) General truths and lies	Linguistic	62%	0.61	0.67	0.64	0.63	0.57	0.6
	Dialog	54%	0.54	0.56	0.55	0.54	0.52	0.53
	Non-verbal	61%	0.64	0.54	0.58	0.6	0.69	0.64
	All Features	65%	0.64	0.67	0.66	0.66	0.63	0.65
(2) Lies and truths by guests	Linguistic	66%	0.64	0.73	0.68	0.69	0.58	0.63
	Dialog	57%	0.58	0.52	0.55	0.56	0.61	0.59
	Non-verbal	61%	0.60	0.62	0.61	0.61	0.58	0.60
	All Features	69%	0.66	0.76	0.71	0.72	0.61	0.69
(3) Lies and truths by host	Linguistic	55%	0.55	0.60	0.57	0.56	0.51	0.53
	Dialog	58%	0.58	0.61	0.59	0.59	0.55	0.57
	Non-verbal	57%	0.56	0.62	0.59	0.58	0.52	0.55
	All Features	65%	0.67	0.61	0.64	0.64	0.69	0.67

Table 7: Automated performances for the three analyzed scenarios.The baseline for all scenarios (detecting a lie) is 50%; Acc. = Accuracy; P = Precision; R = Recall; F = F1-score

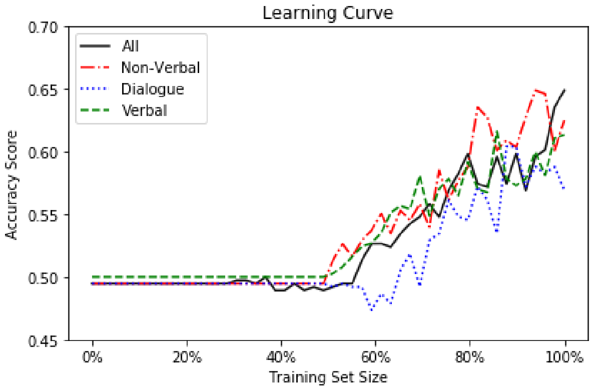


Figure 2: Learning curves (averaged 5-fold cross validation) for each modality using random forest (general truth/lie scenario)

seem to be different while telling lies and truths as we observe noticeable improvement in the models build with non-verbal cues. Third, the performance of the dialog features suggests that having evidence of previous behavior (as captured by deception changes and non-verbal behavior changes) can be useful when modeling the deceptive behavior of a single individual, further suggesting that the non-verbal component of a lie seems to be more individually shaped for each person as opposed to the linguistic component. However, the differences in performance between scenarios 1 and 2 suggest that the current features might not be enough to capture deceptive behavior by a single individual since the developed classifiers still find this task challenging.

The preliminary analyses of this new multimodal dataset show promising results by successfully classifying truthful and deceptive behaviors. Currently, the dataset provides feature sets drawn from three modalities (verbal and non-verbal, as well as dialogue) but can be further analyzed to extract additional features from other modalities such as speech. Specifically, the dialogue has the potential to add many more layers of information by systematically analyzing verbal, non-verbal, and speech patterns between the participants. These patterns can lead to detectable differences between actions and reactions within a dialogue (Tsunomori et al., 2015; Levitan et al., 2016). We consider analyzing such patterns as a future research venue to expand the analyses on our dataset.

A challenge while using this data is the current imbalance of truthful and deceptive feature sets, which can have a detrimental effect on classification performance. However, there are several other possible ways to address this issue other than down-sampling as we did during our experiments. For instance, other computational methods could be explored, such as one-class classification tasks. Such models train on a dataset from the same distribution and classify new data as being similar or different to that distribution. This way, anomalies (i.e., behavior with a feature configuration different from the training set) are detectable. Since truthful behavior is underrepresented in our dataset, the deceptive features could serve as the training set and the goal is to detect truthful behavioral patterns. Expanding on other computational tasks also tackles future applicable problems of dealing with uneven datasets, as they are often present when working with real-life datasets. The issue of an underrepresented class is prevalent in deception detection research.

Finally, the dataset could be expanded with more behavioral data, mainly by augmenting the number of truthful behaviors. Since all the contestants in our dataset are celebrities, it is likely that other videos portraying them are available.

8 Conclusion

In this paper, we showed how we can successfully build a multimodal dialog dataset for deception detection, and presented exploratory deception classification tasks. We showed how we can integrate multiple modalities and build feature sets useful for automated processing. We were able to achieve a classification performance that is better than random guessing and exceeds human performance. Furthermore, additional modalities systematically showed an improvement in classification performance. The best performance of 69% was obtained by combining multiple verbal, nonverbal, and dialogue feature sets, which represents a significant improvement over the human performance of a maximum of 58% accuracy.

The dataset introduced in this paper represents a first attempt to integrate the dialogue dimension with multiple other modalities in deception detection research. It has the potential of triggering novel research on multimodal deception data, specifically for speech and the dialogue dimension, which should be explored in the future.

All the data annotations described in this paper are available upon request.

Acknowledgments

This material is based in part upon work supported by the Michigan Institute for Data Science, by the National Science Foundation (grant #1815291), and by the John Templeton Foundation (grant #61156). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Michigan Institute for Data Science, the National Science Foundation, or the John Templeton Foundation.

References

Mohamed Abouelenien, Rada Mihalcea, and Mihai Burzo. 2016. Analyzing Thermal and Visual Clues of Deception for a Non-Contact Deception Detection Approach. pages 1–4. ACM Press.

Mohamed Abouelenien, Verónica Pérez-Rosas, Rada Mihalcea, and Mihai Burzo. 2017a. Detecting deceptive behavior via integration of discriminative features from multiple modalities. *IEEE Transactions on Information Forensics and Security*, 12(5):1042–1055.

Mohamed Abouelenien, Veronica Perez-Rosas, Rada Mihalcea, and Mihai Burzo. 2017b. Detecting Deceptive Behavior via Integration of Discriminative Features From Multiple Modalities. *IEEE Transactions on Information Forensics and Security*, 12(5):1042–1055.

Jens Allwood, Loredana Cerrato, Laila Dybkjaer, Kristiina Jokinen, Costanza Navarretta, and Patrizia Paggio. 2005. The MUMIN multimodal coding scheme. *NorFA yearbook*, 2005:129–157.

Charles F. Bond and Bella M. DePaulo. 2006. Accuracy of Deception Judgments. *Personality and Social Psychology Review*, 10(3):214–234.

Judee K. Burgoon, David B. Buller, Cindy H. White, Walid Afifi, and Aileen L. S. Buslig. 1999. The role of conversational involvement in deceptive interpersonal interactions. *Personality and Social Psychology Bulletin*, 25(6):669–686.

Bella M. DePaulo, James J. Lindsay, Brian E. Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *Psychological Bulletin*, 129(1):74–118.

FFmpeg Developers. 2016. Ffmpeg tool (Version be1d324). http://ffmpeg.org/.

Tommaso Fornaciari and Massimo Poesio. 2013. Automatic deception detection in Italian court cases. *Artificial Intelligence and Law*, 21(3):303–340.

Jeffrey T Hancock, Lauren E Curry, Saurabh Goorha, and Michael T Woodworth. 2004. Lies in Conversation: An Examination of Deception Using Automated Linguistic Analysis. page 6.

Timothy R. Levine, Hee Sun Park, and Steven A. McCornack. 1999. Accuracy in detecting truths and lies: Documenting the “veracity effect”. *Communication Monographs*, 66(2):125–144.

Sarah Ita Levitan, Yocheved Levitan, Guozhen An, Michelle Levine, Rivka Levitan, Andrew Rosenberg, and Julia Hirschberg. 2016. Identifying individual differences in gender, ethnicity, and personality from dialogue for deception detection. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 40–44.

Sarah Ita Levitan, Angel Maredia, and Julia Hirschberg. 2018. Linguistic Cues to Deception and Perceived Deception in Interview Dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1941–1950, New Orleans, Louisiana. Association for Computational Linguistics.

Mart Lubbers and Francisco Torreira. 2013. Pympiling: A Python module for processing ELANs EAF and Praats TextGrid annotation files.

T.O. Meservy, M.L. Jensen, J. Kruse, D.P. Twitchell, G. Tsechpenakis, J.K. Burgoon, D.N. Metaxas, and J.F. Nunamaker. 2005. Deception Detection through Automatic, Unobtrusive Analysis of Nonverbal Behavior. *IEEE Intelligent Systems*, 20(5):36–43.

Rada Mihalcea, Verónica Pérez-Rosas, and Mihai Burzo. 2013. Automatic detection of deceit in verbal communication. pages 131–134. ACM Press.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Travis E. Oliphant. 2006. *A Guide to NumP*. Trelgol Publishing.

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 497–501.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. 2011. Scikit-learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON*, 12:2825–2830.

James W Pennebaker, Cindy K Chung, Molly Ireland, Amy Gonzales, and Roger J Booth. 2007. The Development and Psychometric Properties of LIWC. page 22.

Veronica Perez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Y. Xiao, C.J. Linton, and Mihai Burzo. 2015. Verbal and Nonverbal Clues for Real-life Deception Detection. *EMNLP*, pages 2336–2346.

Veronica Perez-Rosas, Rada Mihalcea, Alexis Narvaez, and Mihai Burzo. 2014. A Multimodal Dataset for Deception Detection. *LREC*, pages 3118–3122.

Taylan Sen, Md Kamrul Hasan, Zach Teicher, and Mohammed Ehsan Hoque. 2018. Automated Dyadic Data Recorder (ADDR) Framework and Analysis of Facial Cues in Deceptive Communication. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–22.

Yuiko Tsunomori, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. An analysis towards dialogue-based deception detection. In *Natural Language Dialog Systems and Intelligent Assistants*, pages 177–187. Springer.

Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. 2006. ELAN: A Professional Framework for Multimodality Research. page 4.

Xiang Yu, Shaoting Zhang, Zhennan Yan, Fei Yang, Junzhou Huang, Norah E Dunbar, Matthew L Jensen, Judee K Burgoon, and Dimitris N Metaxas. 2015. Is interactional dissynchrony a clue to deception? insights from automated analysis of nonverbal visual cues. *IEEE transactions on cybernetics*, 45(3):492–506.

A Crowdsourced Corpus of Multiple Judgments and Disagreement on Anaphoric Interpretation

Massimo Poesio
Queen Mary University
m.poesio@qmul.ac.uk

Jon Chamberlain
University of Essex
jchamb@essex.ac.uk

Udo Kruschwitz
University of Essex
udo@essex.ac.uk

Silviu Paun
Queen Mary University
s.paun@qmul.ac.uk

Alexandra Uma
Queen Mary University
a.n.uma@qmul.ac.uk

Juntao Yu
Queen Mary University
juntao.cn@gmail.com

Abstract

We present a corpus of anaphoric information (coreference) crowdsourced through a game-with-a-purpose. The corpus, containing annotations for about 108,000 markables, is one of the largest corpora for coreference for English, and one of the largest crowdsourced NLP corpora, but its main feature is the large number of judgments per markable: 20 on average, and over 2.2M in total. This characteristic makes the corpus a unique resource for the study of disagreements on anaphoric interpretation. A second distinctive feature is its rich annotation scheme, covering singletons, expletives, and split-antecedent plurals. Finally, the corpus also comes with labels inferred using a recently proposed probabilistic model of annotation for coreference. The labels are of high quality and make it possible to successfully train a state of the art coreference resolver, including training on singletons and non-referring expressions. The annotation model can also result in more than one label, or no label, being proposed for a markable, thus serving as a baseline method for automatically identifying ambiguous markables. A preliminary analysis of the results is presented.

1 Introduction

A number of datasets for anaphora resolution / coreference now exist (Poesio et al., 2016), including ONTONOTES that has been the de facto standard since the CONLL shared tasks in 2011 and 2012 (Pradhan et al., 2012), and the just introduced and very substantial PRECO corpus (Chen et al., 2018). None of these datasets however take into account the research challenging the idea that a 'gold standard' interpretation can be obtained through adjudication, in particular for anaphora (Poesio and Artstein, 2005b; Wong and Lee, 2013; Aroyo and Welty, 2015). Virtually every project devoted to large-scale annotation of discourse or semantic phenomena has reached the conclusion that genuine disagreements are widespread. This has long been known for anaphora (Poesio and Artstein, 2005b; Versley, 2008; Recasens et al., 2011) (see also the analysis of disagreements in ONTONOTES in (Pradhan et al., 2012)) and wordsenses (Passonneau et al., 2012), but more recent work has provided evidence that disagreements are frequent for virtually every aspect of language interpretation, not just in subjective tasks such as sentiment analysis (Kenyon-Dean et al., 2018), but even in the case of tasks such as part-of-speech tagging (Plank et al., 2014). In fact, researchers in the CrowdTruth project view disagreement as positive, arguing that "disagreement is signal, not noise" (Aroyo and Welty, 2015). In this paper we present what to our knowledge is the largest corpus containing alternative anaphoric judgments: 20.6 judgments per markable on average (up to 90 judgments in some cases) for about 108,000 markables. We are not aware of any comparable resource for studying disagreement and ambiguity in anaphora or indeed any other area of NLP. We present some preliminary analysis in the paper.

The corpus presented in this paper is also the largest corpus for anaphora / coreference entirely created through crowdsourcing, and one of the largest corpus of coreference information for English in terms of markables. So far, only fairly small coreference corpora have been created using crowdsourcing (Chamberlain et al.; Guha et al., 2015). The corpus presented here provides annotations for about 108,000 markables, 55% of the number of markables in ONTONOTES. Another novelty is that the corpus was created through a 'quasi' Game-With-A-Purpose (GWAP) (von Ahn, 2006; Lafourcade et al., 2015), *Phrase Detectives* (Poesio et al., 2013), and is, to our knowledge, the largest GWAP-created corpus for NLP. (So far, the success of GWAPs in other areas of science (Clery,

Proceedings of NAACL-HLT 2019, pages 1778–1789
Minneapolis, Minnesota, June 2 - June 7, 2019.

2011; Cooper et al., 2010) has not been replicated in NLP.) Finally, the corpus is notable for a richer annotation scheme than the other large coreference corpora. Singletons were marked as well as mentions participating in coreference chains (the omission of singletons being one of the main problems with ONTONOTES). Non-referring expressions were also annotated: both expletives (not annotated either in ONTONOTES or PRECO) and predicative NPs. Finally, all types of plurals were annotated, including also **split-antecedent** plurals as in *John met with Mary, and they went to dinner*, which again are not annotated either in ONTONOTES or PRECO.

Turning a crowdsourced corpus into a high-quality dataset suitable to train and evaluate NLP systems requires, however, an aggregation method appropriate to the data and capable of achieving sufficient quality, something that simple majority voting typically cannot guarantee (Dawid and Skene, 1979; Hovy et al., 2013). What made it possible to extract such a dataset from the collected judgments was the recent development of a probabilistic method for aggregating coreference annotations called MPA (Paun et al., 2018b). MPA extracts silver labels from a coreference annotation and associates them with a probability, allowing for multiple labels in cases of ambiguity. As far as we know, ours is the first use of MPA to create a large-scale dataset. We show in the paper that MPA can be used to extract from the judgments a high quality coreference dataset that can be used to develop standard coreference resolvers, as well as to investigate disagreements on anaphora.

2 Background

2.1 Datasets for Anaphora/Coreference

Since the two CONLL shared tasks (Pradhan et al., 2012), ONTONOTES has become the dominant resource for anaphora resolution research (Fernandes et al., 2014; Björkelund and Kuhn, 2014; Martschat and Strube, 2015; Clark and Manning, 2015, 2016a,b; Lee et al., 2017, 2018). ONTONOTES contains documents in three languages, Arabic (300K tokens), Chinese (950K) and English (1.6M), from several genres but predominantly news. One frequently discussed limitation of ONTONOTES is the absence of singletons (De Marneffe et al., 2015; Chen et al., 2018), which makes it harder to train models for mention detection (Poesio et al., 2018). Another limitation is that expletives are not annotated. As a consequence, downstream applications such as machine translation (Guillou and Hardmeier, 2016) that require pronoun interpretation have to adopt various workarounds. Because of these two restrictions, ONTONOTES only has 195K markables, and a low markable density (0.12 markable/token).

A number of smaller corpora provide linguistically richer information (Poesio et al., 2016). Examples include ANCORA for Spanish (Recasens and Martí, 2010), TUBA-D/Z for German (Hinrichs et al., 2005), the Prague Dependency Treebank for Czech and English (Nedoluzhko et al., 2009), and ARRAU for English (Uryupina et al., To Appear). In ARRAU, for example, singletons and expletives are annotated as well, as are split antecedent plurals, generic coreference, discourse deixis, and bridging references. The ARRAU corpus is relatively small in terms of tokens (350K), but has a higher markable density than ONTONOTES (0.29 markable/token), so it has around 100K markables, half the number of ONTONOTES. ARRAU was recently used in the CRAC 2018 shared task (Poesio et al., 2018) to evaluate a number of anaphora resolution tasks.

The recently introduced PRECO corpus (Chen et al., 2018) is the largest existing coreference corpus, consisting of 35,000 documents for a total of 12.5M tokens and 3.8M markables, half of which are singletons. However, the corpus is not intended as a general purpose dataset as only the 3000 most common English words appear in the documents (the majority - 2/3 - of the documents are from Chinese high-school English tests). The corpus's annotation scheme mainly follows the ONTONOTES guidelines, with a few important differences: singleton mentions and generic coreference are annotated, event anaphora is not, and predicative NPs are annotated as co-referring with their argument, as previously done in the MUC (Grishman and Sundheim, 1995; Chinchor, 1998) and ACE (Doddington et al., 2004) corpora.[1] As one could expect, the corpus is relatively easy for coreference systems. The Peters et al. (2018) system trained and tested on PRECO achieves an av-

---

[1] An example of predicative NP is *24 degrees* in *The temperature is 24 degrees*. As discussed by van Deemter and Kibble (2000), annotating *The temperature* and *24 degrees* as coreferent would result in nonsensical coreference chains for sentences like *The temperature was 24 degrees but it is 27 degrees now.* As a result, such markables were annotated as predicative in recent corpora. It's not clear why we find a return to the old practice in PRECO.

erage CONLL score of 81.5%, whereas the same system trained and tested on ONTONOTES only achieves a score of 70.4%.

2.2 Crowdsourcing and GWAPs for NLP

A revolution in the way language annotation tasks are carried out was achieved by **crowdsourcing** (Howe, 2008; Snow et al., 2008). Crowdsourcing comes in many forms, including **citizen science** and **microworking**. A third approach is to use a **game-with-a-purpose (GWAP)** to aggregate data from non-expert players for collective decisions similar to those from an expert (von Ahn, 2006). The game-based approach to collecting language data is initially costly, but once a game is deployed it can continue to collect data with very little financial support, especially if there is an active community. GWAPs such as *Phrase Detectives* (Poesio et al., 2013), *JeuxDesMots* (Joubert and Lafourcade, 2008) and *Zombie Lingo* (Fort et al., 2014) have been used in NLP to collect data on specific linguistic features; broader platforms such as Wordrobe (Venhuizen et al., 2013) to gamify the entire text annotation pipeline.

Crowdsourcing is the most realistic approach to collect a large number of judgments about phenomena such as anaphora. Games in particular are the one type of crowdsourcing scalable to the goal of, for example, a 100M word corpus. So far, however, only small and medium scale resources for NLP have been created via crowdsourcing. For coreference we are only aware of two, both around 50K tokens in size (Chamberlain et al.; Guha et al., 2015). The Groningen Meaning Bank being collected through the Wordrobe platform (Bos et al., 2017) includes many more documents, but so far only very few interpretations have been obtained through the games (e.g., only around 4K judgments have been collected for anaphora).

2.3 Collecting Multiple Judgments

In most of the best known efforts at creating anaphoric corpora for English and other languages substantial disagreements between the coders were observed, but none of the resulting resources contains multiple anaphoric interpretations. Systematic analyses of the disagreements among coders observed in such annotation efforts were provided for ANCORA by Recasens et al. (2011) and for TUBA-D/Z by Versley (2008). The entire ONTONOTES corpus was double annotated, finding disagreements on around 20% of the markables, i.e., around 40,000 cases. An analysis of such disagreements can be found in (Pradhan et al., 2012), but ultimately only the result of adjudication was included in the corpus. Most of the PRECO corpus was doubly annotated and the results adjudicated, but only the result of adjudication is released.

We are aware of only two corpus annotation schemes which explicitly allowed the annotation of anaphoric ambiguity: ARRAU and the Potsdam Commentary Corpus (Krasavina and Chiarcos, 2007). Most of the ARRAU corpus was single-annotated by a highly experienced annotator, who was allowed to mark a variety of cases of ambiguity (Poesio and Artstein, 2005b). It is known, however, that such explicit marking of ambiguity is difficult (Poesio and Artstein, 2005b; Recasens et al., 2012), and indeed not many cases of ambiguity were marked in this way in ARRAU.

3 Collecting the Judgments

In this Section we discuss what type of judgments were collected, and how.

3.1 A gamified approach

The gamified online platform *Phrase Detectives*[2] (Chamberlain et al., 2008; Poesio et al., 2013) was used to collect the judgments about anaphoric reference included in the corpus. *Phrase Detectives* is articulated around a number of tasks centered around the detective metaphor and uses scoring, progression and a variety of other mechanisms to make the activity enjoyable. In **annotation mode** (*Name the Culprit*), the participant provides an anaphoric judgment about a highlighted markable (the possible judgments according to the annotation scheme are discussed next). If different participants enter different interpretations for a markable then each interpretation is presented to other participants in **validation mode** (*Detectives Conference*), in which the participants have to agree or disagree with the interpretation.

One of the key differences between *Phrase Detectives* and GWAPs such as those developed by von Ahn and his lab (von Ahn, 2006) is the much greater complexity of judgments required. Yet clearly we cannot expect participants to be experts about anaphora, or to be willing to read a manual explaining the annotation scheme, so all the training still has to be done while playing the game.

---

[2] `http://www.phrasedetectives.com`

Therefore, we developed a number of mechanisms that could help in this respect: giving suggestions and tips (global, contextual and FAQ), comparing decisions with the gold standard, and showing agreement with other players in Validation Mode. When participants begin to play they are shown training texts (in which the answer is known from a gold standard) and get feedback as to whether their decisions agree with the gold standard. Once the player has completed all training tasks they are given a user rating (the percentage of correct decisions out of the total number of training tasks).

As of 17th of March 2019, 60,822 individuals have participated in *Phrase Detectives* over ten years and using different platforms, providing over 4.26 million judgments, about half of which are included in the present release.

3.2 Types of Judgments

The judgments asked to the participants to *Phrase Detectives* follow a simplified version of the ARRAU annotation scheme, which is the result of extensive tests for intercoder agreement (Uryupina et al., To Appear). The participants are asked to make two basic distinctions: whether a markable is referring or not, and if referring, whether it is **Discourse-Old** (DO), i.e., it refers to an entity already mentioned (in which case the players were asked to indicate the latest mention of that entity), or **Discourse-New** (DN), i.e., it introduces a new entity in the discourse. Anaphoric reference marked include **split antecedent** anaphora, as in *John met Mary, and they went out for drinks*, where the antecedent for *they* is the set consisting of the separately introduced *John* and *Mary*. Two types of non-referring expressions were marked: **expletives**, as in *It's five o'clock* or *There was a fireplace in the room*; and **predicative NPs**, as in *The temperature is 24 degrees*. In the case of predicative NPs, players were asked to mark the nearest mention of the entity that the predication applied to, following in this case the ONTONOTES approach instead of ARRAU's.

The key difference between this corpus and any other existing corpus for anaphora / coreference with the exception of ARRAU is that the corpus was designed to collect information about disagreement. The main difference from ARRAU is that no attempt was made to ask players to identify ambiguity, as that has proven hard or impossible to do (Poesio and Artstein, 2005b). Instead of **explicit (marking of) ambiguity**, the developers relied on **implicit ambiguity**: that genuine ambiguity would emerge if enough players supplied judgments. All the judgments produced by the players were therefore stored, without attempting to choose among them at collection.

The differences between the four corpora being compared are summarized in Table 1, modelled on a similar table in (Chen et al., 2018). In the *Phrase Detectives* corpus predication and coreference are clearly distinguished, as in ONTONOTES and ARRAU but unlike in PRECO. Singletons are considered markables. Expletives and split antecedent plurals are marked, unlike in either ONTONOTES or PRECO. Most importantly, ambiguity of anaphoric interpretation (as in the example from the TRAINS corpus (Poesio and Artstein, 2005b)) is marked, but implicitly, i.e., by asking the judgment of at least 8 players per markable, as opposed to explicitly, as attempted in ARRAU (with little success).

3.3 Markable identification

Following standard practice in anaphoric annotation and GWAPs, the markables to be annotated were not identified by the participants themselves; instead, markable identification was carried out semi-automatically. Each document would first be processed by a pipeline combining off-the-shelf tools (sentence splitting and tokenization using the OpenNLP pipeline[3] and parsing using the Berkeley Parser (Petrov and Klein, 2007)) and custom preprocessing and post-processing heuristic steps to correct the output. (See (Poesio et al., 2013) for more details about the pipeline and its performance.) Then one of the administrators would carry out a quick check of the document removing the most obvious mistakes before uploading it. After the document was uploaded, participants could report markable errors, which would then be corrected by hand.[4]

4 The corpus

4.1 Basic statistics

This second release of the *Phrase Detectives* corpus consists of a total of 542 documents contain-

---

[3] `http://opennlp.apache.org`

[4] As participants report over 10,000 errors per year, it became quickly apparent that carrying out the corrections ourselves was unfeasible. In subsequent work, we developed a gamified approach to markable identification and correction centered around the *TileAttack!* GWAP (Madge et al., 2017).

Type	**Example**	ONTONOTES	PRECO	ARRAU	Present corpus
predicative NPs	[John] is a teacher [John, a teacher]	Pred	Coref	Pred	Pred
singletons		No	Yes	Yes	Yes
expletives	It's five o'clock	No	No	Yes	Yes
split antecedent plurals	[John] met [Mary] and they ...	No	No	Yes	Yes
generic mentions	[Parents] are usually busy. Parents should get involved	Only with pronouns	Yes	Yes	Yes
event anaphora	Sales [grew] 10%. This growth is exciting	Yes	No	Yes	No
ambiguity	Hook up [the engine] to [the boxcar] and send it to Avon	No	No	Explicit	Implicit

Table 1: Comparison between the annotation schemes in ONTONOTES, PRECO, ARRAU and the present corpus

		Docs	Tokens	Markables
PD_{gold}	Gutenberg	5	7536	1947 (1392)
	Wikipedia	35	15287	3957 (1355)
	GNOME	5	989	274 (96)
	Subtotal	45	23812	6178 (2843)
$\text{PD}_{\text{silver}}$	Gutenberg	145	158739	41989 (26364)
	Wikipedia	350	218308	57678 (19444)
	Other	2	7294	2126 (1339)
	Subtotal	497	384341	101793 (47147)
All	Total	542	408153	107971 (49990)

Table 2: Summary of the contents of the current release. The numbers in parentheses indicate the total number of markables that are non-singletons.

	1	2	3	4	5	6	7
PD_{gold}	38.8	30.6	18.5	7.3	2.5	1.0	0.6
$\text{PD}_{\text{silver}}$	36.0	30.0	19.0	8.8	3.8	1.8	0.8

Table 3: Percentage of markables with X distinct interpretations

ing 408K tokens and 108K markables from two main genres: Wikipedia articles and fiction from the Gutenberg collection. This corpus is divided in two subsets. The subset we refer to as $\text{PD}_{\text{silver}}$ consists of 497 documents, for a total of 384K tokens and 101K markables, whose annotation was completed–i.e. 8 judgments per markable were collected, and 4 validations per interpretation–as of 12th of October 2018. In these documents, an aggregated ('silver') label obtained through MPA (see next Section) is also provided. 45 additional documents were also gold-annotated by two experts annotators. We refer to the subset of the corpus for which both gold and silver annotations are available as PD_{gold}, as it is intended to be used as test set.[5] The gold subset consists of a total of 23K tokens and 6K markables. The contents of the corpus are summarized in Table 2.

By comparison, the English corpus used for the CONLL 2011 and 2012 shared tasks consists of 3493 documents, for a total of 1.6M tokens and 194480 markables. In other words, although the current release of the corpus is only about 25% of the CONLL corpus in terms of tokens, it is 55.5% of its size in terms of annotated markables, i.e., actual training / testing items.

4.2 Number of Judgments

In total, 2,235,664 judgments from 1958 players are included in the current release, of which 1,358,559 annotations and 867,844 validations. On average, 20.6 judgments were collected per markable: 12.6 annotations and 8 validations. In addition, around 10K expert judgments were collected for the gold portion of the corpus from two expert annotators. This compares with 600K estimated judgments for the entire ONTONOTES corpus, about 3 per markable (total number of annotators not known), and around 10M for PRECO, also 3 per markable, from about 80 annotators.

4.3 Disagreement: a preliminary analysis

The 'raw' statistics about disagreement in the corpus are shown in Table 3. In total, only 35.7% of the markables in the corpus (38,579) were assigned only one interpretation by the participants, whereas 64.3% received more than one interpretation. This figure would seem to suggest massive ambiguity, but we are not saying that 64.3% of markables in the corpus are ambiguous. As already pointed out e.g. in (Pradhan et al., 2012),

---

[5] $\text{PD}_{\text{gold}}$is the dataset released in 2016 as *Phrase Detectives* corpus, Release 1 (Chamberlain et al.).

there are a number of reasons for disagreements among coders / players apart from ambiguity. In the case of ONTONOTES, the causes for the 20,000 observed disagreements include:

- Ambiguity proper, i.e., unclear interpretation ('Genuine Ambiguity' in (Pradhan et al., 2012)) and/or disagreement on reference (31% of the disagreements in ONTONOTES, around 7% of all markables);
- Annotator error (another 25% of the cases of disagreement in ONTONOTES);
- Various limitations of the coding scheme: unclarity in the guidelines, inability to mark certain types of coreference e.g., between generics, etc. (36.5% of the cases of disagreement in ONTONOTES).
- Interface limitations (around 7.5% of the disagreements in ONTONOTES).

Some of the disagreements due to other causes–and in particular annotation errors–can be filtered through validation, i.e., by excluding those interpretations of a markable for which the **validation score** (annotations + agreements - disagreements) falls below a threshold. For example, if only interpretations with a validation score > 0 are considered, we find that 51,075 / 107,971 markables have at least two such interpretations, i.e., 47.3% of the total, which is considerably less than the 64.3% of markables with more than one interpretation, but it's still a large number.

We will discuss a more sophisticated method for automatically identifying plausible interpretations, as well as the results of a preliminary hand-analysis of the disagreements in a few documents in our corpus, in Section 7.

5 Aggregation

5.1 Probabilistic Aggregation Methods

The data collected via *Phrase Detectives* require an aggregation method to help choose between the different interpretations provided by the players. Simple heuristics such as majority voting are known to underperform compared to probabilistic models of annotation (Whitehill et al., 2009; Raykar et al., 2010; Quoc Viet Hung et al., 2013; Sheshadri and Lease, 2013; Hovy et al., 2013; Passonneau and Carpenter, 2014; Paun et al., 2018a). The models offer a rich framework of interpretation and can employ distinct prior and likelihood structures (pooled, unpooled, and partially pooled) and a diverse set of effects (annotator ability, item difficulty, or a subtractive relationship between the two). However, most work on models of annotation assumes that the set of classes the annotators can choose from is fixed across the annotated items, which is not the case for anaphoric annotation. More specifically, in *Phrase Detectives* the participants can classify a markable as non-referring (expletive or predicative); as introducing a new discourse entity; or as discourse-old, in which case they link it to the most recent mention of its antecedent–and coreference chains are document-specific and not fixed in number (see Section 3.2 for more details on the annotation scheme). Recently, however, Paun et al. (2018b) developed a probabilistic model (MPA) able to aggregate such crowdsourced anaphoric annotations.

5.2 MPA

In MPA, the term **label** is used to refer to a specific interpretation provided by a player, and the term **class** to refer to general interpretation categories such as discourse old, discourse new, expletive, or predicative NP. Please note that under this formalism each label belongs to a class: the antecedents belong to the discourse old category, while the other possible labels (e.g., discourse new) coincide with the classes they belong to. The model assumes a preprocessing step in which the markable-level annotations are transformed into a series of binary decisions with respect to each candidate label. MPA then models these (label-level) decisions as the result of the sensitivity (the true positive rate) and specificity (the true negative rate) of the annotators which it assumes are class dependent. This latter assumption allows inferring different levels of annotator ability for each class (thus capturing, for instance, the fact that whereas most participants are generally able to recognize discourse-new mentions, they are much less good at identifying correct antecedents).

5.3 Aggregating the game data

We use the MPA model as a component in a standard mention-pair framework to extract coreference clusters: 1) link each markable with the most likely label as identified by the model, and 2) follow the link structure to build the coreference

Method	Discouse old class			Discourse new class			Predicative NPs class			Expletives class			Avg. F1	Accuracy
	P	R	F1	P	R	F1	P	R	F1	P	R	F1		
MAJVOTE	**94.5**	62.8	75.4	79.1	**99.0**	87.9	53.9	9.7	16.4	**97.2**	71.4	82.4	65.5	82.9
MPA	90.4	**87.3**	**88.8**	**94.5**	96.0	**95.3**	**64.0**	**72.4**	**68.0**	94.1	**98.0**	**96.0**	**87.0**	**92.2**

Table 4: A per class evaluation of aggregated interpretations against expert annotations.

	Method	MUC			BCUB			CEAFE			Avg. F1
		P	R	F1	P	R	F1	P	R	F1	
Singletons included	MAJVOTE	**96.0**	63.9	76.7	**95.7**	78.7	86.4	77.1	**94.9**	85.1	82.7
	MPA	91.6	**82.4**	**86.8**	94.8	**87.8**	**91.2**	**92.4**	93.8	**93.1**	**90.3**
	STANFORD	65.4	62.4	63.8	78.9	76.1	77.5	78.4	85.2	81.7	74.3
Singletons excluded	MAJVOTE	**96.1**	64.8	77.4	**93.8**	45.0	60.8	66.3	48.5	56.1	64.8
	MPA	92.2	**89.2**	**90.7**	88.1	**77.8**	**82.6**	**79.5**	**80.2**	**79.8**	**84.4**
	STANFORD	65.7	62.1	63.9	50.3	42.5	46.1	42.7	49.8	46.0	52.0

Table 5: The quality of the coreference chains for the PD$_{\text{gold}}$ subset.

chains. We next evaluate both of these components against expert annotations.

Table 4 shows a per class evaluation of the aggregated interpretations from the PD$_{\text{gold}}$ subset. The results indicate an overall better agreement with the expert annotations of MPA compared with a simple majority voting (MAJVOTE) baseline. This is because MAJVOTE makes the an implicit assumption that the annotators have equal expertise, which is not true in general even with data crowdsourced on microworking platforms, and even more so with data collected through GWAPs (Paun et al., 2018a).

After inferring the mention pairs, coreference chains can be extracted and their quality assessed using standard coreference metrics. Table 5 presents the evaluation against gold chains in PD$_{\text{gold}}$. We compare the chains produced from the mention pairs inferred by MPA and by MAJVOTE, and the chains produced by the STANFORD deterministic coreference system (Lee et al., 2011) (for which we switched off post-processing to output singleton clusters). The results indicate a far better quality of the chains produced using MPA over the alternative methods. Another interesting result is that even a simple MAJVOTE baseline based on crowdsourced annotations performed far better than the STANFORD system, underlining the advantage of crowdsourced annotations for coreference over automatically produced annotations.

6 Using the corpus for coreference resolution

Some NLP researchers may question the usefulness of the information about disagreements for coreference resolution (or other NLP tasks). In this Section, we demonstrate that even those purely interested in CONLL-style coreference resolution can use the *Phrase Detectives* corpus aggregated with MPA as a dataset. We use PD$_{\text{silver}}$ to train a coreference system able to simultaneously identify non-referring expression and build coreference chains (including singletons). As no other system of this type exists at the moment, we developed one ourselves.

6.1 Our system

The system trained and tested on the corpus is a cluster ranking system that does mention detection and corefence resolution jointly. The system uses the mention representation from the state-of-the-art (Lee et al., 2018) system, but replaces their mention-ranking model with a cluster ranking model. Our cluster ranking model forms clusters by going through the candidate mentions in their text order and adding them to the clusters, which take into consideration the relative importance of the mentions. An attention mechanism is used to assign mentions within the clusters salience scores, and the clusters are represented as the weighted sums of the mention representations. Separate classifiers are used to identify non-referring markables and singletons.

Singletons	Method	MUC			BCUB			CEAFE			Avg. F1
		P	R	F1	P	R	F1	P	R	F1	
Included	Our Model	79.3	72.5	75.7	72.1	69.3	70.7	70.5	73.2	71.8	72.7
Excluded	Our Model	79.3	**72.5**	**75.7**	58.3	52.4	**55.2**	**58.3**	**49.5**	**53.5**	**61.5**
	Our Model*	77.8	71.8	74.6	55.4	**53.7**	54.6	56.2	49.0	52.4	60.5
	Lee et al. (2018)*	**80.8**	66.1	72.7	**63.3**	45.1	52.7	56.7	44.7	50.0	58.5

Table 6: The CoNLL scores for our systems trained on PD$_{\text{silver}}$ and tested on PD$_{\text{gold}}$. * indicates the models trained on the simplified corpus.

	P	R	F1
Non-referring	55.2	54.0	54.6
Expletives	62.3	86.0	72.3
Predicative NPs	49.7	47.7	48.7

Table 7: Non-referring scores for our model

6.2 Evaluation Methodology

We randomly chose 1/20 of PD$_{\text{silver}}$ as a development set and use the rest as the training set; PD$_{\text{gold}}$ was used as test set.

To get a baseline, we compare the results of our system on a simplified version of the corpus without singletons and expletives with those obtained by the current state-of-the-art system on ONTONOTES, Lee et al. (2018) trained and tested on the same data.

6.3 Results

Table 6 shows the results of both systems on the simplified corpus. Our cluster ranking system achieved an average CONLL score of 60.5%, outperforming the Lee et al. (2018) system by 2 percentage points. Note that the Lee et al. (2018) system achieved a higher score on the CONLL data, which suggests that the present corpus is different from that dataset.

In the same Table, we also report the results obtained by training our system on the full corpus including both non-referring expressions and singletons. This version of system achieves an average CONLL score of 72.7%.[6] We will note that although this score is on system mentions, it is very close to the score (74.3%) achieved by the Stanford deterministic system evaluated with gold mentions (see Table 5 in Section 5). Also, this model trained on the full corpus including singletons achieves a gain of 1 percentage point when compared with the model trained on the simplified corpus even when evaluated in a singleton excluded setting. This indicates that the availability of the singletons is also helpful for resolving non-singleton clusters. In total, this model achieved a CONLL score on singletons excluded of 3 percentage points 3% better than our baseline.

Regarding the task of identifying non-referring mentions, our model achieved a F1 score of 54.6% (see Table 7). The scores of the system on distinct types of non-referring expressions is presented in the following two rows of Table 7. Our model achieved a higher F1 score of 72.3% on expletives, and a lower score (48.7%) on predicative NPs.

Overall, these results–the first results on system mentions for PD$_{\text{gold}}$–suggest that the silver corpus is sufficient to train a state-of-the-art system and achieve a reasonably good performance. Also, that training a model on a corpus enhanced by singletons and non-referring markables results in a better CONLL score when compared with a model trained on the simplified corpus.

7 Disagreements, revisited

In the previous Section we showed that MPA can be used to extract a silver standard out of the annotations that is suitable to train a CONLL-style coreference resolver or an extended coreference resolver also attempting identification of singletons and non-referring expressions. The key property of the corpus however is the information it provides about disagreements. The second useful contribution of MPA is that it can be used to get an assessment of the ambiguity of markables which is more refined than that discussed in Section 4.3. For each markable, MPA assigns a probability to each interpretation. Given that the model does not assume the existence of a 'gold', there are three possible cases for each markable: either only one interpretation has a probability above a

---

[6] The Extended Coreference Scorer developed for the 2018 CRAC shared task (Poesio et al., 2018) was used to evaluate coreference chains on a corpus using singletons and to assess non-referring expressions identification.

	None	One	Two	Zero or more
$\mathrm{PD}_{\mathrm{gold}}$	2.3%	93.4%	4.3%	6.6%
$\mathrm{PD}_{\mathrm{silver}}$	3.5%	94%	2.4%	5.9%

Table 8: Ambiguity in the corpus according to MPA

certain threshold–say, 0.5; or more than one interpretation is above that threshold; or none is. This assessment of ambiguity according to MPA is summarized in Table 8.

This assessment appears to suggest a similar prevalence of ambiguity in our corpus than found in ONTONOTES in the already mentioned analysis by Pradhan et al. (2012). In order to verify this, two experts hand-analyzed 2 documents in $\mathrm{PD}_{\mathrm{gold}}$ containing a total of 900 markables: *Little Red Cap (LRC)* and *Robber Bridegroom* (RG). Given that each markable has on average 20 interpretations, and that player errors are frequent (there is at least one player error for almost every markable) it wasn't possible to use the same categories as Pradhan et al. Instead, we simply attempted to assign markables to one of the categories: **Genuine ambiguity (GA)**, **Interface or Coding Scheme Problem (ICP)**, **Other (O)**. The results are summarized in Table 9. The Table has one row per document. The first column lists the total number of markables in a document; the second (Dis) the percentage of markables on which there is disagreement; the third (GA) the percentage of the *total* number of markables which are cases of genuine ambiguity; and the fourth (ICP) the percentage which are cases of Interface or Coding Scheme Problem. As we can see from the Table, 9% of the total number of markables in these documents (80 out of 865) are genuinely ambiguous, i.e., that 12.6% of the disagreements (80 out of 633) are cases of genuine ambiguity. These are only preliminary figures, and we suspect that the ultimate figures on the prevalence of ambiguity are going to be much higher, given that Recasens et al. (2012) report that 12-15% of coreference relations in their corpus are cases of quasi-coreference, and that Poesio and Artstein (2005a) report a figure of 42.6% once ambiguity on discourse deictic reference are taken into account.

We next checked the extent to which MPA can correctly predict genuine ambiguity. The results suggest that MPA is good at removing spurious ambiguity, but as a predictor of ambiguity it only has a recall of around 20% and a precision of slightly under 50%. Improving these results is one of the objectives of our current research.

	Total	Dis	GA	ICP
LRC	401	79.1%	7% (28)	7.7% (31)
RG	464	68.3%	11.2% (52)	12.9% (60)
Average		73.7%	9.1%	10.3%

Table 9: Analysis of disagreements in two corpus documents

8 Conclusions

We presented a novel resource for anaphora that, because of its annotation scheme and size, at the very least should be useful to those in the community interested in developing systems able to perform a more comprehensive form of anaphora resolution, including for instance expletive detection and split antecedent resolution. The key property of this new resource however is that it provides a large number of judgments about each anaphoric expression, thus enabling the development of systems that do not make the assumption that a 'gold standard' exists, an assumption questioned by all studies associated with the creation of the current resources for the task. The dataset is also to our knowledge the first solid evidence that the games-with-a-purpose approach can be successfully deployed to obtain substantial resources for NLP.

The corpus is freely available from the Linguistic Data Consortium and from `http://www.dali-ambiguity.org`. It is distributed in three formats: an XML format including all the judgments, suitable for analysis of disagreements and/or the development of systems taking disagreement into account; and in CONLL and CRAC18 format, with only the gold annotation or the silver label extracted, for those interested in using the corpus as an alternative resource for developing coreference systems only.

Acknowledgements

This research was supported by the DALI project, funded by the European Research Council (ERC), Grant agreement ID: 695662.

References

Luis von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Lora Aroyo and Chris Welty. 2015. Truth is a lie: Crowd truth and the seven myths of human annotation. *AI Magazine*, 36(1):15–24.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd ACL*, volume 1, pages 47–57.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J. Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In N. Ide and J. Pustejovsky, editors, *The Handbook of Linguistic Annotation*, chapter 18, pages 463–496. Springer.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. phrase detectives corpus.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2008. Phrase Detectives: A web-based collaborative annotation game. In *Proceedings of I-Semantics 2008*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of EMNLP*, pages 172–181, Brussels, Belgium.

Nancy Chinchor. 1998. Overview of MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the ACL*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of EMNLP*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the ACL*.

Daniel Clery. 2011. Galaxy evolution. Galaxy Zoo volunteers share pain and glory of research. *Science*, 333(6039):173–5.

Seth Cooper, Firsas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popovic, and the Foldit Players. 2010. Predicting protein structures with a multiplayer online game. *Nature*, 466:756–760.

Alexander P. Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer errorrates using the EM algorithm. *Applied Statistics*, 28(1):20–28.

Marie-Catherine De Marneffe, Marta Recasens, and Christopher Potts. 2015. Modeling the lifespan of discourse entities with application to coreference resolution. *Journal of Artificial Intelligence Research*, 52(1):445–475.

Kees van Deemter and Rodger Kibble. 2000. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637. Squib.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proceedings of LREC*, volume 2, page 1.

Eraldo R. Fernandes, Cícero N. dos Santos, and Ruy L. Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Karen Fort, Bruno Guillaume, and H. Chastant. 2014. Creating Zombilingo, a game with a purpose for dependency syntax annotation. In *Proceedings of the 1st International Workshop on Gamification for Information Retrieval (GamifIR'14)*, pages 2–6. ACM.

Ralph Grishman and Beth Sundheim. 1995. Design of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, pages 1–11. Association for Computational Linguistics.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. 2015. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Proceedings of NAACL*.

Liane Guillou and Christian Hardmeier. 2016. Protest: A test suite for evaluating pronouns in machine translation. In *Proceedings of LREC*, Paris, France.

Erhard W. Hinrichs, Sandra übler, and Karin Naumann. 2005. A unified representation for morphological, syntactic, semantic and referential annotations. In *Proc. of the ACL Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor, Michigan.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL*, pages 1120–1130.

Jeff Howe. 2008. *Crowdsourcing: Why the power of the crowd is driving the future of business*. Crown Publishing Group.

Alan Joubert and Mathieu Lafourcade. 2008. Jeuxdemots : Un prototype ludique pour l'émergence de relations entre termes. In *Proceedings of JADT*.

Kian Kenyon-Dean, Eisha Ahmed, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, Roman Sarrazingendron, Rohit Verma, and Derek Ruths. 2018. Sentiment analysis:

it's complicated! In *Proceedings of NAACL*, pages 1886–1895. ACL.

Olga Krasavina and Christian Chiarcos. 2007. The Potsdam coreference scheme. In *Proceedings of the 1st Linguistic Annotation Workshop*, pages 156–163.

Mathieu Lafourcade, Alain Joubert, and Nathalie Le Brun. 2015. *Games with a Purpose (GWAPs)*. Wiley.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of CONLL: Shared Task*, pages 28–34, Stroudsburg, PA, USA.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*.

Kenton Lee, Luheng He, and Luke S. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of ACL*.

Chris Madge, Jon Chamberlain, Udo Kruschwitz, and Massimo Poesio. 2017. Experiment-driven development of a gwap for marking segments in text. In *Proceedings of CHI PLAY*, Amsterdam.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

A. Nedoluzhko, J. Mirokvský, and P. Pajas. 2009. The coding scheme for annotating extended nominal coreference and bridging anaphora in the prague dependency treebank. In *Proceedings of the Linguistic Annotation Workshop*, pages 108–111.

Rebecca J. Passonneau, Vikas Bhardwaj, Ansaf Salleb-Aouissi, and Nancy Ide. 2012. Multiplicity and word sense: evaluating and learning from multiply labeled word sense annotations. *Language Resources and Evaluation*, 46(2):219–252.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018a. Comparing bayesian models of annotation. *Transactions of the Association for Computational Linguistics*.

Silviu Paun, Jon Chamberlain, Udo Kruschwitz, Juntao Yu, and Massimo Poesio. 2018b. A probabilistic annotation model for crowdsourcing coreference. In *Proceedings of EMNLP*, pages 1926–1937, Brussels, Belgium. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Barbara Plank, Dirk Hovy, and Anders Sogaard. 2014. Linguistically debatable or just plain wrong? In *Proceedings of EACL*.

Massimo Poesio and Ron Artstein. 2005a. Annotating (anaphoric) ambiguity. In *Proc. of the Corpus Linguistics Conference*, Birmingham.

Massimo Poesio and Ron Artstein. 2005b. The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account. In *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation*, pages 76–83.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase Detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Transactions on Interactive Intelligent Systems*, 3(1):1–44.

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Moosavi, Ina Roesiger, Adam Roussel, Fabian Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, and Heike Zinsmeister. 2018. Anaphora resolution with the arrau corpus. In *Proc. of the NAACL Worskhop on Computational Models of Reference, Anaphora and Coreference (CRAC)*, pages 11–22, New Orleans.

Massimo Poesio, Sameer Pradhan, Marta Recasens, Kepa Rodriguez, and Yannick Versley. 2016. Annotated corpora and annotation tools. In M. Poesio, R. Stuckardt, and Y. Versley, editors, *Anaphora Resolution: Algorithms, Resources and Applications*, chapter 4. Springer.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning (CoNLL 2012)*, Jeju, Korea.

Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. 2013. An evaluation of aggregation techniques in crowdsourcing. In *Web Information Systems Engineering – WISE 2013*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322.

Marta Recasens, Ed Hovy, and M. Antònia Martí. 2011. Identity, non-identity, and near-identity: Addressing the complexity of coreference. *Lingua*, 121(6):1138–1152.

Marta Recasens and M. Antònia Martí. 2010. AnCora-CO: Coreferentially annotated corpora for Spanish and Catalan. *Language Resources and Evaluation*, 44(4):315–345.

Marta Recasens, M. Antonia Martí, and Constantin Orasan. 2012. Annotating near-identity from coreference disagreements. In *Proceedings of LREC*.

Aashish Sheshadri and Matthew Lease. 2013. SQUARE: A benchmark for research on computing crowd consensus. In *Proceedings of the 1st AAAI Conference on Human Computation (HCOMP)*, pages 156–164.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.

Olga Uryupina, Ron Artstein, Antonella Bristot, Federica Cavicchio, Francesca Delogu, Kepa J. Rodriguez, and Massimo Poesio. To Appear. Annotating a broad range of anaphoric phenomena, in a variety of genres: the ARRAU corpus. *Journal of Natural Language Engineering*.

Noortje Venhuizen, Valerio Basile, Kilian Evang, and Johan Bos. 2013. Gamification for word sense labeling. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS'13)*.

Yannick Versley. 2008. Vagueness and referential ambiguity in a large-scale annotated corpus. *Research on Language and Computation*, 6:333–353.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043. Curran Associates, Inc.

Billy T. M. Wong and Sophia Y. M. Lee. 2013. Annotating legitimate disagreement in corpus construction. In *Proceedings of IJCNLP*, pages 51–57, Nagoya, Japan.

A Streamlined Method for Sourcing Discourse-level Argumentation Annotations from the Crowd

Tristan Miller[*†] and **Maria Sukhareva**[‡] and **Iryna Gurevych**[*†]

[*]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
`https://www.ukp.tu-darmstadt.de/`

[†]Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
`https://www.aiphes.tu-darmstadt.de/`

[‡]NLP Lab
BMW Group, Munich
`Maria.Sukhareva@bmwgroup.com`

Abstract

The study of argumentation and the development of argument mining tools depends on the availability of annotated data, which is challenging to obtain in sufficient quantity and quality. We present a method that breaks down a popular but relatively complex discourse-level argument annotation scheme into a simpler, iterative procedure that can be applied even by untrained annotators. We apply this method in a crowdsourcing setup and report on the reliability of the annotations obtained. The source code for a tool implementing our annotation method, as well as the sample data we obtained (4909 gold-standard annotations across 982 documents), are freely released to the research community. These are intended to serve the needs of qualitative research into argumentation, as well as of data-driven approaches to argument mining.

1 Introduction

Empirical study of argumentation requires examples drawn from authentic, human-authored text. Likewise, the applications of computational argumentation, such as argument mining, can require significant amounts of argument-annotated data to achieve reasonable performance. However, this data can be challenging to obtain in sufficient quantity and quality, particularly for discourse-level argumentation. This is because discourse-level annotation schemes are necessarily complex with respect to discrimination and delimitation (i.e., the variety of markable elements in the text and how to define their boundaries), expressiveness (i.e., the need to tag relationships between annotated elements), and context weighting (i.e., the amount of context around markable units that needs to be considered) (Fort et al., 2012). Successfully applying such schemes typically requires expensive and laborious work by expert-trained annotators.

In this paper, we present a method that facilitates the application of one such discourse-level argument annotation scheme (Stab and Gurevych, 2014). This scheme has been widely cited and used in argumentation studies (e.g., Lippi and Torroni, 2015; Persing and Ng, 2015; Nguyen and Litman, 2015; Persing and Ng, 2016; Ghosh et al., 2016; Eger et al., 2017; Nguyen and Litman, 2018), and while it is fairly coarse-grained, it is expensive to apply to new texts. Our method breaks down the annotation process into incremental, intuitive steps, each focusing on a small portion of the overall annotation scheme. We apply this method in a crowdsourcing setup with annotators who receive no training other than a brief set of annotation guidelines, as well as in a more traditional setup with extensively trained local annotators. We find that agreement between the two groups increases sublinearly with the number of crowd annotators, achieving up to $\alpha_U = 0.52$ when using ten crowd workers. We release not only our sample data set (consisting of 4909 gold-standard argument component and argument relation annotations over 982 product reviews), but also the source code for the annotation tool itself, which will allow others to produce their own quantity- and quality-controlled annotated data sets.

Proceedings of NAACL-HLT 2019, pages 1790–1796
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Background and Previous Mork

While there exists a great diversity of argumentation theories in philosophy and logic (e.g., Toulmin, 2003; Freeman, 2011; Walton et al., 2012), they tend to agree that an argument can be decomposed into various interrelated components. Inspired by Freeman's (2011) theory of the macro-structure of argumentation, Stab and Gurevych (2014) broadly categorize these components as *claims* (the conclusions that the audience is persuaded to accept or reject), *premises* (additional information offered to support or attack a given claim), and the *major claim* (the one central claim that relates all other claims in an argument). Taken together, this can be conceptualized as a graph or tree structure, with vertices representing the *argument components* (major claims, claims, and premises) and the directed edges representing the *argument relations* (support and attack).

Stab and Gurevych (2014) annotate a collection of persuasive texts with this scheme, associating each argument component they identify with a contiguous span of text from the document. They report that the annotation process involved "several training sessions" with their annotators, including collaborative annotation of eight example documents in order to obtain a common understanding of the task. This level of effort is in line with what has been reported for other discourse-level argumentation schemes. For example, annotation studies using the Freemanesque schemes of Peldszus and Stede (2013), Li et al. (2017), Haddadan et al. (2018), and Musi et al. (2018) all required one or more lengthy training sessions guided by argumentation experts and up to six pages of written instructions.

Using existing methods to alleviate the knowledge acquisition bottleneck, such as incidental supervision (Roth, 2017), or pre-annotation (Fort and Sagot, 2010), could speed the work of annotators—possibly at the risk of introducing a training bias—but would not obviate the need for expert training. (In any case, pre-annotation has never, to our knowledge, been successfully applied to hard discourse-level tasks such as annotating argumentation structures.) The complexity of the annotation scheme also seemingly rules out the use of crowdsourcing (Howe, 2006) and gamification (von Ahn, 2006), which are geared towards microtasks that are quick and easy for humans. Though one previous study has decomposed a discourse-level scheme for use with crowdsourcing (Kawahara et al., 2014), the constraints it imposes (fixed-size annotations, maximum document length of three sentences) are too restrictive for argumentation annotation.

By contrast, the crowdsourcing approach of Sukhareva et al. (2016), while not concerned with discourse-spanning annotations, employs a few mechanisms that are relevant for our own task. Their approach, intended for the labelling of semantic verb relations, breaks down the annotation work into a series of hierarchical, atomic microtasks. Only those parts of the annotation instructions relevant to the current microtask are shown to the annotator. Furthermore, annotators are encouraged to think of connecting words ("specifically", "generally speaking", "in other words", etc.) that justify their relation annotations. As described in the following section, we adapt and extend these mechanisms for our own annotation method.

3 Annotation Method

Our approach to mitigating the knowledge acquisition problem is an iterative procedure by which annotators apply a distinct subset of the annotation scheme at each step. In this manner, complex discourse-level annotations are built up piecemeal in simple steps. The iterative annotation process is supported by an online JavaScript-based interface. Taken together, this allows the Stab and Gurevych (2014) annotation scheme to be applied even by untrained annotators in a crowdsourcing setup.

In the first step of the annotation process, annotators are presented with the complete argumentative text and asked to select the one phrase (i.e., an arbitrary sequence of words) that best represents the major claim, or else to indicate that there is no such passage.[1] In the second step, annotators are presented once again with the full argument, but with its major claim marked.[2] The annotators then select the claims—that is, all phrases that directly speak to the major claim, as well as whether those passages support or attack the major claim—or else

---

[1] If the user indicates in any step that there is no text span corresponding to the argument component type, we ask them to perform a short alternative task. This is to prevent faithless workers from taking the easy way out of the annotation task, but also to collect further annotations of interest to us.

[2] In the second and third steps, the marked annotation is not necessarily the one applied by the annotator in the previous step. In fact, as we explain below, in our study we source all annotations from a given step simultaneously, distill them into a gold standard, and mark these gold-standard annotations for the next step. With this setup, there is no need for a given annotator to participate in all three steps.

indicate that there are no such text spans. In the third step, annotators see the full argument with one of its claims marked. As with the previous step, annotators select text spans corresponding to the premises of the claim and indicate each premises's stance; they also have the option of reporting that the claim has no premises.

The annotation tool automatically enforces the restrictions that annotations must be contiguous, must begin and end on a word boundary, and cannot overlap with their siblings or ancestors. Crucially, the instructions given to annotators at each step of the process do not attempt to explain the entire annotation scheme but rather describe only the immediate annotation task in layman's terms. Furthermore, the tool attempts to make this task more intuitive for users by framing the second and third steps as a sentence completion task. An example of this is the interface for annotating claims (see Fig. 1). The full argumentative text (in this case, a product review) is shown on the left half of the screen, with the major claim marked, and we separately show a copy of the major claim on the right half of the screen. The user is instructed to extend the major claim with additional supporting or attacking information by appending a "because" or "but" clause, respectively. The user does this by pressing the "but" or "because" button below the major claim and then highlighting a sentence or phrase from the review.

4 Annotation Study

To assess the suitability of our annotation procedure, we applied it in a crowdsourcing setup. Measuring interannotator agreement for crowdsourced annotations is problematic, however, because there are typically a huge number of annotators, most of whom annotate only a tiny fraction of the data set. To gauge the reliability of our crowdsourced annotations, we instead conducted an experiment that compared them to those produced by expert-trained annotators.

For the experiment, we randomly selected 40 Amazon product reviews from the McAuley et al. (2015) data set—four from each of ten product categories. Each review was annotated for major claims by ten crowd workers; all 40 reviews were also annotated for major claims by a fixed group of three locally recruited annotators trained by argumentation experts.[3] We then converted the annotated reviews to BIO tokens (Ramshaw and Marcus, 1995) and applied the annotation aggregation/denoising tool MACE (Hovy et al., 2013) to select at most two gold-standard major-claim annotations per review, one from the crowd (crowd) and one from the trained annotators (train).[4] We then compared the crowd and train gold standards, one review at a time, using Krippendorff's (1995) α_U, a unitizing measure that considers the token-level boundaries of the text spans marked by each annotator. We repeated this process to obtain and evaluate crowd and train claim annotations on the train major claims, and then again for crowd and train premise annotations on the train claims.

Note that in the gold standards for some reviews, there may be no major claim, no claims associated with the major claim, and/or no premises associated with a given claim. In many cases this is because the annotators generally agreed that such argument components were not present in the text. However, in other cases the various annotators *did* identify such argument components, but the agreement among them was too low for MACE to output a gold-standard annotation. A quandary therefore arises when deciding how to treat reviews where *neither* the crowd nor the train gold standard contains a given type of argument component. There is no (easy) way of determining from the MACE output whether missing annotations are due to agreement or disagreement, and even if this information were available, it is not clear how it could be incorporated into the calculation of α_U. For this reason, we apply two different strategies for handling missing annotations, and provide separate α_U calculations for each. The first strategy, skip, disregards missing annotations, excluding them from the mean agreement calculation. The second strategy, agree, treats missing annotations as total agreement ($\alpha_U = 1$).[5]

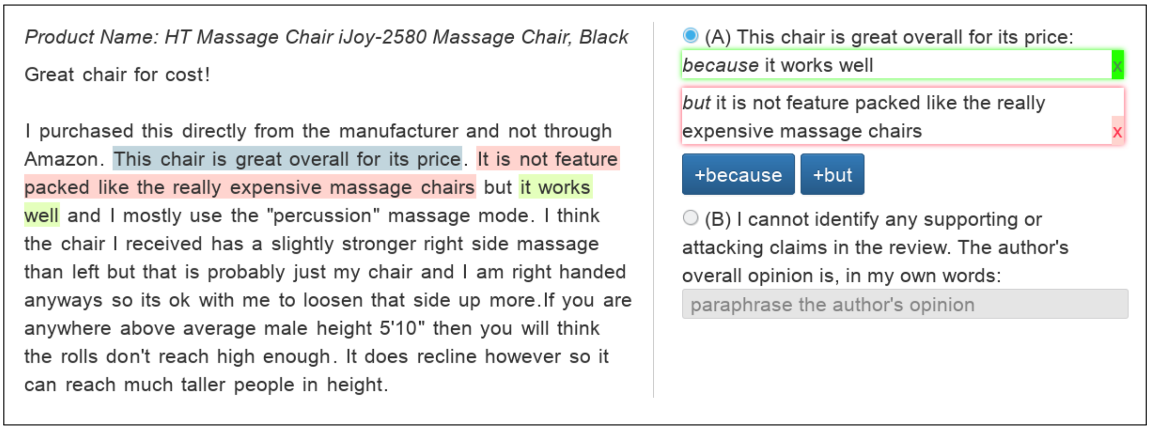
When using all ten crowdsourced annotations per review and the agree strategy, we achieved mean α_U scores of 0.4104, 0.5231, and 0.4385 for major claims, claims, and premises, respectively. With the skip strategy, the respective scores are 0.4104, 0.4845, and 0.2201. As expected, these scores

---

[3] We engaged US-based workers from Amazon Mechanical Turk at the US federal minimum wage of $7.25/hour. Our expert-trained annotators were salaried research staff whose equivalent hourly rate was three to five times higher.

[4] MACE accepts a threshold value that is used to discard instances that cannot be confidently assigned a gold label; we set this to 0.9.

[5] It is not possible to treat the missing annotations as "total disagreement" because per Krippendorff (1995), α_U has no concept of this; there is no lowest disagreement score.



Product Name: HT Massage Chair iJoy-2580 Massage Chair, Black

Great chair for cost!

I purchased this directly from the manufacturer and not through Amazon. This chair is great overall for its price. It is not feature packed like the really expensive massage chairs but it works well and I mostly use the "percussion" massage mode. I think the chair I received has a slightly stronger right side massage than left but that is probably just my chair and I am right handed anyways so its ok with me to loosen that side up more.If you are anywhere above average male height 5'10" then you will think the rolls don't reach high enough. It does recline however so it can reach much taller people in height.

● (A) This chair is great overall for its price:

because it works well

but it is not feature packed like the really expensive massage chairs

○ (B) I cannot identify any supporting or attacking claims in the review. The author's overall opinion is, in my own words:

Figure 1: Annotation interface for the second step (claim annotation)

are lower than the agreement among extensively trained annotators reported by Stab and Gurevych (2014) ($\alpha_U = 0.7726, 0.6033, 0.7594$).[6] However, they are broadly comparable to interannotator agreement scores reported in similar (and in some cases, even simpler) discourse-level argument annotation studies with expert-trained annotators, such as Aharoni et al. (2014) ($\kappa = 0.4$), Musi et al. (2018) ($\kappa = 0.296$), and Li et al. (2017) ($\alpha_U = 0.2452$).

To measure how the number of crowd annotations impacts reliability, we performed an ablation study where we iteratively removed one crowd annotation at random from each review and repeated the MACE distillation and α_U calculation. The study was repeated 100 times and the resulting α_U scores averaged. The results are shown in Fig. 2, which plots the average α_U scores for major claims, claims, and premises when using one to ten crowd annotations per review. The plots are shown as error bars, where the top of the bar is the average agree score and the bottom is the average skip score. Reliability scores start to be uniformly positive with three annotations, with agreement for major claims and premises plateauing around seven annotations. The difference between the agree and skip scores is sizable only for premises.

5 Data Set and Software

Having satisfied ourselves that our method can produce reliable annotations via crowdsourcing, we applied it to a much larger subset of McAuley et al. (2015). The raw data consists of 982 English product reviews randomly sampled from the same ten product categories used in our evaluation study. For each argument component type in a review, we sourced annotations from five crowd workers, considering this to be an acceptable trade-off between annotation quality and cost. The MACE-produced gold standard contains 4909 annotations (937 major claims, 1134 claims, 852 premises, and 1986 argument relations). Our data set is distinguished from the review corpora of García Villalba and Saint-Dizier (2012) and Wyner et al. (2012) in that it is much larger, covers a broader range of product types, and is freely released under the CC BY 4.0 licence. It is comparable in size to but broader in scope than the Chinese-language hotel review corpus of Li et al. (2017).

Our data is distributed[7] as a set of XML Metadata Interchange (XMI) files, one per review, containing stand-off argument annotations that cross-reference the original texts from McAuley et al. (2015). (Because the original review texts are not available under a free licence, we do not include them in our distribution, but we provide a script for extracting them from the original corpus and merging them into our XMIs.) Also included is the JavaScript source for our annotation tool, as well as the Java source for preprocessing the raw data and postprocessing the annotations with MACE. This code can be used to crowdsource further annotated data sets using the McAuley et al. (2015) data at the desired level of quality. It could also be adapted to work with other raw corpora and other Freemanesque annotation schemes.

6 Conclusion

We have presented a scalable, simplified, iterative method for sourcing the discourse-level argumen-

---

[6] Apart from the fact that we used untrained annotators, the difference in agreement may also be due in part to our use of online user-generated content as opposed to student essays.

[7] `https://github.com/UKPLab/naacl2019-argument-annotations`

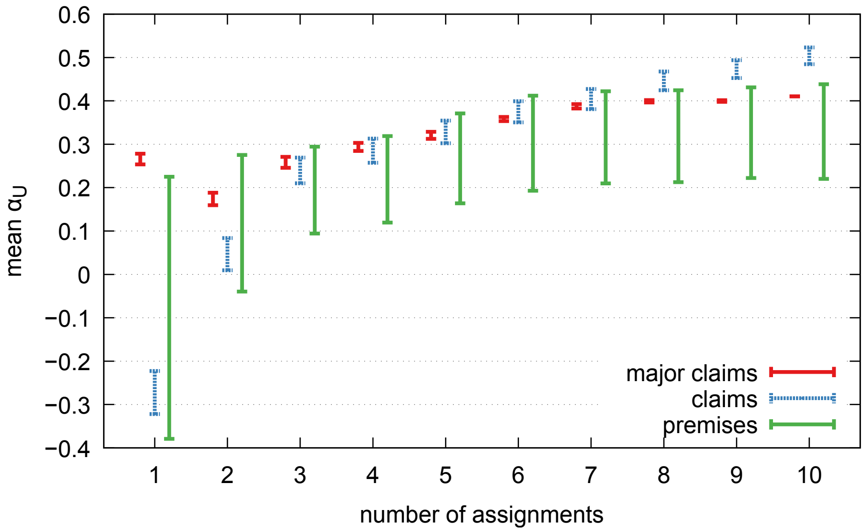


Figure 2: Results of the reliability study

tation annotations of Stab and Gurevych (2014), which may be adaptable to other annotation schemes based on Freeman's (2011) notion of argumentation. Our analysis shows that crowdsourced annotations obtained with our method yield substantial agreement with those obtained, with much greater effort, by expert-trained annotators. We have used our method to quickly and cheaply produce a large, argument-annotated data set of product reviews, which we freely release, along with the source code to our annotation interface and processing tools. Unlike with flat, context-free argument data such as that of Stab et al. (2018), training on our annotations would conceivably permit the identification not just of isolated argument components but of more complex argument structures. Our resources may also be of use for qualitative research on the linguistic features and rhetorical mechanisms of argumentative text (e.g., Peldszus and Stede, 2016).

For future work, we are investigating alternatives to MACE, which was designed for categorical annotations rather than the sequence labelling of our task. In particular, we are looking into the Bayesian method of Simpson and Gurevych (2018), which takes advantage of the sequential dependencies between BIO tags, and works more robustly with noisy, subjective data such as ours.

Acknowledgments

The authors thank Johannes Daxenberger and Christian Stab for many insightful discussions.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional references 03VP02540 (ArgumenText) and 01UG1816B (CEDIFOR), and by the DFG-funded research training group “Adaptive Preparation of Information form Heterogeneous Sources” (AIPHES, GRK 1994/1).

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the 1st Workshop on Argumentation Mining*, pages 64–68. Association for Computational Linguistics.

Luis von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 11–22. Association for Computational Linguistics.

Karën Fort, Adeline Nazarenko, and Sophie Rosset. 2012. Modeling the complexity of manual annotation tasks: A grid of analysis. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 895–910.

Karën Fort and Benoît Sagot. 2010. Influence of pre-annotation on POS-tagged corpus development. In *Proceedings of the 4th Linguistic Annotation Workshop*, pages 56–63. Association for Computational Linguistics.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*, volume 18 of *Argumentation Library*. Springer Netherlands.

María Paz García Villalba and Patrick Saint-Dizier. 2012. Some facets of argument mining for opinion analysis. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Computational Models of Argument*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 23–34. IOS Press.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 549–554. Association for Computational Linguistics.

Shohreh Haddadan, Elena Cabrio, and Serena Villata. 2018. Annotation of argument components in political debates data. In *Proceedings of the Workshop on Annotation in Digital Humanities*, volume 2155, pages 12–16.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Ho vy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics.

Jeff Howe. 2006. The rise of crowdsourcing. *Wired*, 14(6).

Daisuke Kawahara, Yuichiro Machida, Tomohide Shibata, Sadao Kurohashi, Hayato Kobayashi, and Manabu Sassano. 2014. Rapid development of a corpus with discourse annotations using two-stage crowdsourcing. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 269–278. Dublin City University and Association for Computational Linguistics.

Klaus Krippendorff. 1995. On the reliability of unitizing continuous data. *Sociological Methodology*, 25:7–76.

Mengxue Li, Shiqiang Geng, Yang Gao, Shuhua Peng, Haijing Liu, and Hao Wang. 2017. Crowdsourcing argumentation structures in Chinese hotel reviews. In *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics*, pages 87–92.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 185–191. AAAI Press/International Joint Conferences on Artificial Intelligence.

Julian McAuley, Christopher Targett, Qinfeng "Javen" Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM.

Elena Musi, Manfred Stede, Leonard Kriese, Smaranda Muresan, and Andrea Rocci. 2018. A multi-layer annotated corpus of argumentative text: From argument schemes to discourse relations. In *Proceedings of the 11th International Conference on Language Resources and Evaluation*. European Language Resources Association.

Huy Nguyen and Diane Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 22–28. Association for Computational Linguistics.

Huy V. Nguyen and Diane J. Litman. 2018. Argument mining for improving the automated scoring of persuasive essays. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5892–5899. AAAI Press.

Andreas Peldszus and Manfred Stede. 2013. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 196–204. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2016. Rhetorical structure and argumentation structure in monologue text. In *Proceedings of the 3rd Workshop on Argument Mining*, pages 103–112. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 543–552. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394. Association for Computational Linguistics.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 82–94. Association for Computational Linguistics.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 4885–4890. AAAI Press.

Edwin Simpson and Iryna Gurevych. 2018. Bayesian ensembles of crowds and deep learners for sequence tagging. *arXiv preprint*, 1811.00780.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1501–1510. Dublin City University and the Association for Computational Linguistics.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3664–3674. Association for Computational Linguistics.

Maria Sukhareva, Judith Eckle-Kohler, Ivan Habernal, and Iryna Gurevych. 2016. Crowdsourcing a large dataset of domain-specific context-sensitive semantic verb relations. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 2131–2137. European Language Resources Association.

Stephen E. Toulmin. 2003. *The Uses of Argument*, 2nd edition. Cambridge University Press.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2012. *Argumentation Schemes*, online edition. Cambridge University Press.

Adam Wyner, Jodi Schneider, Katie Atkinson, and Trevor Bench-Capon. 2012. Semi-automated argumentative analysis of online product reviews. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Computational Models of Argument*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 43–50. IOS Press.

Unsupervised Dialog Structure Learning

Weiyan Shi
University of California, Davis
wyshi@ucdavis.edu

Tiancheng Zhao
Carnegie Mellon University
tianchez@cs.cmu.edu

Zhou Yu
University of California, Davis
joyu@ucdavis.edu

Abstract

Learning a shared dialog structure from a set of task-oriented dialogs is an important challenge in computational linguistics. The learned dialog structure can shed light on how to analyze human dialogs, and more importantly contribute to the design and evaluation of dialog systems. We propose to extract dialog structures using a modified VRNN model with discrete latent vectors. Different from existing HMM-based models, our model is based on variational-autoencoder (VAE). Such model is able to capture more dynamics in dialogs beyond the surface forms of the language. We find that qualitatively, our method extracts meaningful dialog structure, and quantitatively, outperforms previous models on the ability to predict unseen data. We further evaluate the model's effectiveness in a downstream task, the dialog system building task. Experiments show that, by integrating the learned dialog structure into the reward function design, the model converges faster and to a better outcome in a reinforcement learning setting.

1 Introduction

Human dialogs are like well-structured buildings, with words as the bricks, sentences as the floors, and topic transitions as the stairs connecting the whole building. Therefore, discovering dialog structure is crucial for various areas in computational linguistics, such as dialog system building (Young, 2006), discourse analysis (Grosz and Sidner, 1986), and dialog summarization (Murray et al., 2005; Liu et al., 2010). In domain specific tasks such as restaurant booking, it's common for people to follow a typical conversation flow. Current dialog systems require human experts to design the dialog structure, which is time consuming and sometimes insufficient to satisfy various customer needs. Therefore, it's of great importance to automatically discover dialog structure from existing human-human conversations and incorporate it into the dialog system design.

However, modeling human conversation is not easy for machines. Some previous work rely on human annotations to learn dialog structures in supervised learning settings (Jurafsky, 1997). But since human labeling is expensive and hard to obtain, such method is constrained by the small size of training examples, and by the limited number of application domains (Zhai and Williams, 2014). Moreover, structure annotations on human conversation can be subjective, which makes it hard to reach inter-rater agreements. Therefore, we propose an unsupervised method to infer the latent dialog structure since unsupervised methods do not require annotated dialog corpus.

Limited previous work has studied unsupervised methods to model the latent dialog structure. Most of the previous methods use the *hidden Markov model* to capture the temporal dependency within human dialogs (Chotimongkol, 2008; Ritter et al., 2010; Zhai and Williams, 2014). We propose to adopt a new type of models, the *variational recurrent neural network* (VRNN, a recurrent version of the VAE) (Chung et al., 2015), and infer the latent dialog structure with variational inference. VRNN is suitable for modeling sequential information. Compared to the simpler HMM models, VRNN also has the flexibility to model highly non-linear dynamics (Chung et al., 2015) in human dialogs.

Our basic approach assumes that the dialog structure is composed of a sequence of latent states. Each conversational exchange (a pair of user and system utterances at time t) belongs to a *latent state*, which has causal effect on the future latent states and the words the conversants produce. Because discrete latent states are more interpretable than continuous ones, we combine

Proceedings of NAACL-HLT 2019, pages 1797–1807
Minneapolis, Minnesota, June 2 - June 7, 2019.

VRNN with Gumbel-Softmax (Jang et al., 2016) to obtain discrete latent vectors to represent the latent states. A common way to represent the dialog structure both visually and numerically is to construct a transition probability table among latent states. The idea of transition table inspires us to develop two model variants to model the dependency between states indirectly and directly.

Once we obtain such a human-readable dialog structure, we can use it to facilitate many downstream tasks, such as dialog system training. The motivation is that the dialog structure contains important information on the flow of the conversation; if the automatic dialog system can mimic the behaviour in human-human dialogs, it can interact with users in a more natural and user-friendly way. Therefore, we propose to integrate the dialog structure information into the reward design of the reinforcement learning (RL). Experiments show that the model with the proposed reward functions converges faster to a better success rate.

2 Related Work

Variational Autoencoders (VAEs) (Kingma and Welling, 2013; Doersch, 2016; Kingma et al., 2014) have gained popularity in many computational linguistics tasks due to its interpretable generative model structure (Miao and Blunsom, 2016; Miao et al., 2017). Zhao et al. (2018) applied VAE to learn discrete sentence representations and achieved good results. This is similar to our work, but we focus more on modeling the dialog structure and using the learned structure to improve the dialog system training. Serban et al. (2017) presented a VHRED model which combines the VRNN and encoder-decoder structure for direct dialog response generation. While also similar, our model uses discrete latent vectors instead of continuous ones, and re-constructs the utterances to recover the latent dialog structure, instead of modeling the responses directly.

There are some previous studies on discovering latent structure of conversations (Chotimongkol, 2008; Ritter et al., 2010; Zhai and Williams, 2014). But they are all based on Hidden Markov Model (HMM). Ritter et al. (2010) extended the HMM-based method in Chotimongkol (2008) by adding additional word sources to social interaction data on Twitter. Zhai and Williams (2014) decoupled the number of topics and the number of states to allow an additional layer of information in task-oriented dialogs. Our work also focuses on task-oriented dialogs but adopts the VRNN to perform variational inference in the model. According to Chung et al. (2015), the VRNN retains the flexibility to model highly non-linear dynamics, compared to simpler Dynamic Bayesian Network models such as HMM.

Gunasekara et al. (2017) described a Quantized-Dialog Language Model (QDLM) for task-oriented dialog systems, which performs clustering on utterances and models the dialog as a sequence of clusters to predict future responses. The idea of dialog discretization is similar to our method, but we choose VAE over simple clustering to allow more context-sensitivity and to capture more dynamics in the dialog beyond surface forms of the conversation. Additionally, we propose to utilize the dialog structure information to improve the dialog system training.

Traditional reward functions in RL dialog training use delayed reward to provide feedback to the model. However, delayed reward suffers from potential slow convergence rate problem, so some studies integrated estimated per-turn immediate reward. For example, Ferreira and Lefèvre (2013) studied expert-based reward shaping in dialog management. We use the KL-divergence between the transition probabilities and the predicted probabilities as the immediate per-turn reward. Different from the expert-based reward shaping, such reward does not require any manual labels and is generalizable to different tasks.

3 Models

Fig. 1 gives an overview of the Discrete-VRNN (D-VRNN) model and the Direct-Discrete-VRNN (DD-VRNN) model. In principal, the VRNN contains a VAE at every timestep, and these VAEs are connected by a state-level RNN. The hidden state variable $\mathbf{h}_{\text{t-1}}$ in this RNN encodes the dialog context up to time t. This connection helps the VRNN to model the temporal structure of the dialog (Chung et al., 2015). The observed inputs $\mathbf{x}_t$ to the model is the constructed utterance embeddings. $\mathbf{z}_t$ is the latent vector in the VRNN at time t. Different from Chung et al. (2015), $\mathbf{z}_t$ in our model is a discrete one-hot vector of dimension N, where N is the total number of latent states.

The major difference between D-VRNN and DD-VRNN lies in the priors of $\mathbf{z}_t$. In D-VRNN, we assume that $\mathbf{z}_t$ depends on the entire dialog

(a) Discrete-VRNN (b) Direct-Discrete-VRNN

Figure 1: Discrete-VRNN (D-VRNN) and Direct-Discrete-VRNN (DD-VRNN) overview. D-VRNN and DD-VRNN use different priors to model the transition between $\mathbf{z}_{\text{t}}$, shown in red solid lines. The regeneration of $\mathbf{x}_{\text{t}}$ is in blue dotted lines. The recurrence of the state-level RNN is in gray dash-dotted lines. The inference of $\mathbf{z}_{\text{t}}$ is in black dashed lines.

context $\mathbf{h}_{\text{t-1}}$, shown in red in Fig. 1(a), which is the same as in Chung et al. (2015); while in DD-VRNN we assume that in the prior, $\mathbf{z}_t$ directly depends on $\mathbf{z}_{\text{t-1}}$ in order to model the direct transition between different latent states, shown in red in Fig. 1(b). We use $\mathbf{z}_t$ and $\mathbf{h}_{\text{t-1}}$ to regenerate the current utterances $\mathbf{x}_t$ instead of generating the next utterances $\mathbf{x}_{\text{t+1}}$, shown in blue dotted lines in Fig. 1. The idea of regeneration helps recover the dialog structure. Next, the recurrence in the RNN takes $\mathbf{h}_{\text{t-1}}$, $\mathbf{x}_t$ and $\mathbf{z}_t$ to update itself, and allows the context to be passed down as the dialog proceeds, shown in gray dash-dotted lines. Finally in the inference, we construct the posterior of $\mathbf{z}_t$ with the context $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$, and infer $\mathbf{z}_t$ by sampling from the posterior, shown in black dashed lines in Fig. 1. The mathematical details of each operation are described below. $\varphi_\tau^{(\cdot)}$ are highly flexible feature extraction functions such as neural networks. $\varphi_\tau^{\mathbf{x}}$, $\varphi_\tau^{\mathbf{z}}$, $\varphi_\tau^{\text{prior}}$, $\varphi_\tau^{\text{enc}}$, $\varphi_\tau^{\text{dec}}$ are feature extraction networks for the input $\mathbf{x}$, the latent vector $\mathbf{z}$, the prior, the encoder and the decoder.

Sentence Embedding. $u_t = [\mathrm{w}_{1,\mathrm{t}}, \mathrm{w}_{2,\mathrm{t}}, \ldots \mathrm{w}_{\mathrm{n_w},t}]$ and $s_t = [\mathrm{v}_{1,t}, \mathrm{v}_{2,t}, \ldots \mathrm{v}_{\mathrm{n_v},t}]$ are the user utterance and the system utterance at time t, where $\mathrm{w}_{\mathrm{i,j}}$ and $\mathrm{v}_{\mathrm{i,j}}$ are individual words. The concatenation of utterances from both parties, $x_t = [u_t, s_t]$, is the observed variable in the VAE. We use Mikolov et al. (2013) to perform word embedding and the average of the word embedding vectors of u_t and s_t are $\overline{u_t}$ and $\overline{s_t}$. The concatenation of $\overline{u_t}$ and $\overline{s_t}$ is used as the feature extraction of $\mathbf{x}_t$, namely $\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t) = [\overline{u_t}, \overline{s_t}]$. $\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t)$ is the model inputs.

Prior in D-VRNN. The prior quantifies our assumption on $\mathbf{z}_t$ before we observe the data. In the D-VRNN, it's reasonable to assume that the prior of $\mathbf{z}_t$ depends on the context $\mathbf{h}_{\text{t-1}}$ and follows the distribution shown in Eq. (1), because conversation context is a critical factor that influences dialog transitions. Since $\mathbf{z}_t$ is discrete, we use softmax to obtain the distribution.

$$\mathbf{z}_t \sim \text{softmax}(\varphi_\tau^{\text{prior}}(\mathbf{h}_{t-1})) \qquad (1)$$

Prior in DD-VRNN. The dependency of $\mathbf{z}_{\text{t}}$ on the entire context $\mathbf{h}_{\text{t-1}}$ in Eq. (1) makes it difficult to disentangle the relation between $\mathbf{z}_{\text{t-1}}$ and $\mathbf{z}_{\text{t}}$. But this relation is crucial in decoding how conversations flow from one conversational exchange to the next one. So in DD-VRNN, we directly model the influence of $\mathbf{z}_{\text{t-1}}$ on $\mathbf{z}_{\text{t}}$ in the prior, shown in Eq. (2) and Fig. 1(b). To fit this prior distribution into the variational inference framework, we approximate $p(\mathbf{z}_{\text{t}}|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ with $p(\mathbf{z}_{\text{t}}|\mathbf{z}_{\text{t-1}})$ in Eq. (3). Later, we show that the designed new prior has benefits under certain scenarios.

$$\mathbf{z}_t \sim \text{softmax}(\varphi_\tau^{\text{prior}}(\mathbf{z}_{\text{t-1}})) \qquad (2)$$

$$\begin{aligned} p(\mathbf{x}_{\le T}, \mathbf{z}_{\le T}) &= \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_{\le t}, \mathbf{x}_{\le t}) p(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \\ &\approx \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_{\le t}, \mathbf{x}_{\le t}) p(\mathbf{z}_t|\mathbf{z}_{\text{t-1}}) \end{aligned} \qquad (3)$$

Generation. $\mathbf{z}_t$ is a summarization of the current conversational exchange under the context. We use $\mathbf{z}_t$ and $\mathbf{h}_{\text{t-1}}$ to reconstruct the current utterance $\mathbf{x}_t$. This regeneration of $\mathbf{x}_t$ allows us to recover the dialog structure.

We use two RNN decoders, *dec1* and *dec2*, parameterized by γ_1 and γ_2 to generate the original u_t and s_t respectively. c_t and d_t are the hidden states of *dec1* and *dec2*. The context $\mathbf{h}_{\text{t-1}}$ and feature extraction vector $\varphi_{\tau}^{\mathbf{z}}(\mathbf{z}_t)$ are concatenated to form the initial hidden state h_0^{dec1} of *dec1*. $c_{(n_w,t)}$ is the last hidden state of *dec1*. Since v_t is the response of u_t and will be affected by u_t, we concatenate $c_{(n_w,t)}$ to d_0 to pass the information from u_t to v_t. This concatenated vector is used as h_0^{dec2} of *dec2*. This process is shown in Eq. (4) and (5).

$$\begin{aligned} c_0 &= [\mathbf{h}_{t-1}, \varphi_{\tau}^{\mathbf{z}}(\mathbf{z}_t)], \\ w_{(i,t)}, c_{(i,t)} &= f_{\gamma 1}(w_{(i-1,t)}, c_{(i-1,t)}) \end{aligned} \tag{4}$$

$$\begin{aligned} d_0 &= [\mathbf{h}_{t-1}, \varphi_{\tau}^{\mathbf{z}}(\mathbf{z}_t), c_{(n_w,t)}], \\ v_{(i,t)}, d_{(i,t)} &= f_{\gamma 2}(v_{(i-1,t)}, d_{(i-1,t)}) \end{aligned} \tag{5}$$

Recurrence. The state-level RNN updates its hidden state $\mathbf{h}_t$ with $\mathbf{h}_{\text{t-1}}$ based on the following Eq. (6). f_θ is a RNN parameterized by θ.

$$\mathbf{h}_t = f_\theta(\varphi_{\tau}^{\mathbf{z}}(\mathbf{z}_t), \varphi_{\tau}^{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_{t-1}) \tag{6}$$

Inference. We infer $\mathbf{z}_t$ from the context $\mathbf{h}_{\text{t-1}}$ and current utterances $\mathbf{x}_t$, and construct the posterior distribution of $\mathbf{z}_t$ by another softmax, shown in Eq. (7). Once we have the posterior distribution, we apply Gumbel-Softmax to take samples of $\mathbf{z}_t$. D-VRNN and DD-VRNN differ in their priors but not in their inference, because we assume the direct transitions between $\mathbf{z}_t$ in the prior instead of in the inference.

$$\mathbf{z}_t | \mathbf{x}_t \sim \text{softmax}([\varphi_{\tau}^{\text{enc}}(\mathbf{h}_{t-1}), \varphi_{\tau}^{\mathbf{x}}(\mathbf{x}_t)]) \tag{7}$$

Loss function. The objective function of VRNN is a timestep-wise variational lower bound, shown in Eq. (8) (Chung et al., 2015). To mitigate the *vanishing latent variable problem* in VAE, we incorporate bow-loss and Batch Prior Regularization (BPR) (Zhao et al., 2017, 2018) with tunable weights, λ to the final loss function, shown in Eq. (9).

$$\begin{aligned} &\mathcal{L}_{\text{VRNN}} = \mathbb{E}_{q(\mathbf{z}_{\leq T} | \mathbf{x}_{\leq T})}[\log p(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t})+ \\ &\sum_{t=1}^{T} \text{-KL}(q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \parallel p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t}))] \end{aligned} \tag{8}$$

$$\mathcal{L}_{\text{D-VRNN}} = \mathcal{L}_{\text{VRNN-BPR}} + \lambda * \mathcal{L}_{\text{bow}} \tag{9}$$

3.1 Transition Probability Calculation

A good way to represent a dialog structure both numerically and visually is to construct a transition probability table among latent states. Such transition probability can also be used to design reward function in the RL training process. We calculate transition table differently for D-VRNN and DD-VRNN due to their different priors.

D-VRNN. From Eq. (6), we know that $\mathbf{h}_t$ is a function of $\mathbf{x}_{\leq t}$ and $\mathbf{z}_{\leq t}$. Combining Eq. (1) and (6), we find that $\mathbf{z}_t$ is a function of $\mathbf{x}_{\leq t}$ and $\mathbf{z}_{<t}$. Therefore, $\mathbf{z}_{<t}$ has an indirect influence on $\mathbf{z}_t$ through $\mathbf{h}_{\text{t-1}}$. This indirect influence reinforces our assumption that the previous states $\mathbf{z}_{<t}$ impacts future state $\mathbf{z}_t$, but also makes it hard to recover a clear structure and disentangle the direct impact of $\mathbf{z}_{\text{t-1}}$ on $\mathbf{z}_t$.

In order to better visualize the dialog structure and compare with the HMM-based models, we quantify the impact of $\mathbf{z}_{\text{t-1}}$ on $\mathbf{z}_t$ by estimating a bi-gram transition probability table, where $p_{i,j} = \frac{\#(\text{state}_i, \text{state}_j)}{\#(\text{state}_i)}$. The numerator is the total number of the ordered tuples ($\text{state}_{\text{i, t-1}}$, $\text{state}_{\text{j, t}}$) and the denominator is the total number of state_i in the dataset. We choose a bi-gram transition table over a n-gram transition table with a bigger n, as the most recent context is usually the most relevant, but it should be noted that unlike the HMM models, the degree of transition in our model is not limited nor pre-determined, because $\mathbf{z}_t$ captures all the context. Depending on different applications, different n may be selected.

DD-VRNN As stated before, the dependency of $\mathbf{z}_t$ on the entire context $\mathbf{h}_{\text{t-1}}$ creates difficulty in calculating the transition table. This is our motivation to derive the prior in DD-VRNN. The outputs from the softmax in the prior (Eq. (2)) directly constitute the transition table. So rather than estimating the transition probabilities by frequency count as in D-VRNN, we can optimize the loss function of DD-VRNN and get the parameters in Eq. (2) that directly form the transition table.

3.2 NE-D-VRNN

In task-oriented dialog systems, the presence of certain named entities, such as food preference plays a crucial role in determining the phase of the dialog. To make sure the latent states capture such useful information, we assign larger weights on the named entities when calculating the loss function in Eq. (9). The weights encourage the recon-structured utterances to have more correct named entities, therefore influencing the latent state to have better representation. We refer this model as NE-D-VRNN (Named Entitiy Discrete-VRNN).

4 Datasets

We test the proposed method on the CamRest676 corpus, which was released and collected by Wen et al. (2016). The task is to help users find restaurants in Cambridge, UK. While this task is highly similar to DSTC2, we choose this dataset instead of DSTC2 because it is relatively clean and comes with good entity extraction methods. There are a total of 676 dialogs in this dataset with three information slots (*food, price range and area*) and three request table slots (*address, phone and postcode*).

We also evaluate our model on another dataset of simulated conversations, proposed in Zhao and Eskenazi (2018). The task is to help users get the weather report in a certain place at a specific time. The dialog system is controlled by a fixed structure and hand-set probabilities. Therefore, learning the dialog structure of this dataset might be easier.

We assume each latent vector in the VAE emits one conversational exchange, including one user utterance and the corresponding system response at time t, and each conversational exchange corresponds to one latent vector, following Zhai and Williams (2014).

5 Experiments

We use LSTM (Hochreiter and Schmidhuber, 1997) with 200-400 units for the RNNs, and a fully-connected network for all the $\varphi_{\tau}^{(\cdot)}$ with a dropout rate of 0.4. Additionally, we use trainable 300-dimension word embeddings initialized by Google word2vec (Mikolov et al., 2013). The maximum utterance word length is 40 and the maximum dialog length is 10. We set the λ for the bow-loss to be 0.1. 80% of the entire dataset are used for training, 10% for validation and 10% for testing. Parameters mentioned are selected based on the performance of the validation set.

The evaluation of unsupervised methods has always been a challenge. We first compare our models with a simple K-means clustering algorithm to show its context sensitivity. Then we compare our models with traditional HMM methods both qualitatively and quantitatively. Finally, we compare the three proposed model variants. The qualitative evaluation involves generating dialog structures with different models, and the quantitative evaluations involves calculating the likelihood on a held-out test set under a specific model, which measures the model's predictive power.

5.1 Comparison with K-means Clustering

We apply the model and obtain a latent state $\mathbf{z}_t$ for each conversational exchange. Since $\mathbf{z}_t$ is a discrete one-hot vector, we can group the conversational exchanges with the same latent state together. This process is similar to clustering the conversational exchanges. But we choose the VAE over simple clustering methods because the VAE introduces more flexible context-sensitive information. A straightforward clustering method like K-means usually groups sentences with similar surface forms together, unless the previous context is explicitly encoded along with the utterances. To compare the grouping result of our model with a traditional clustering method, we perform K-means clustering on the dataset and calculate the within-cluster cosine similarity between the bag-of-word vectors of the utterances and the context. This cosine similarity measures how similar the utterances are on the word token level, higher the value is, more words the utterances share in common. It turns out the average cosine similarity between the current utterance is 0.536 using the K-means and 0.357 using the D-VRNN, while the average cosine similarity between the context is 0.320 using the K-means and 0.351 using the D-VRNN. This does show that in the D-VRNN result, the context are more similar to each other, while in the K-means, the current utterances are more similar on the surface textual level,which is not ideal because the dialog system needs context information. Table 1 shows an example where the D-VRNN clustering result is different from the K-means result. The conversational exchanges to be clustered are in the last row. These two exchanges have the same surface form but different contexts. D-VRNN identifies them as different by incorporating context information, whereas K-means places them into the same cluster ignoring the context.

5.2 Comparison with HMM

HMM is similar to our model. Actually, if we remove the $\mathbf{h}_t$ layer from Fig. 1, it becomes an HMM. But it is this additional layer of $\mathbf{h}_t$ that encodes the dialog context into continuous information and is crucial in the success of our model.

From	Utterance
SYS:	Okay, you don't care *place*, do you?
USR:	That's correct.
SYS:	What date are you interested?
USR:	**Weather this morning.**
SYS:	**I believe you said this morning.**

(a) One example in State 2, "provide place and time"

From	Utterance
USR:	Weather tomorrow.
SYS:	[api_call]. your weather report is here [report]. what else can I do?
USR:	**Weather this morning.**
SYS:	**I believe you said this morning.**

(b) One example in State 3, "additional time request"

Table 1: The utterances in bold in both tables have similar surface forms but different context. They are grouped in different latent states using the Discrete-VRNN. In contrast, they are in the same cluster using the K-means.

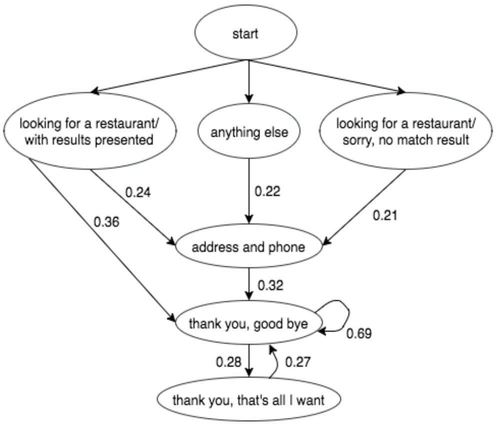


Figure 2: Dialog structure of restaurant data by D-VRNN. Transitions with $P \geq 0.2$ are visualized.

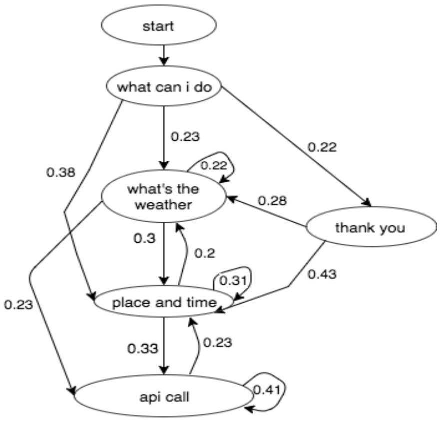


Figure 3: Dialog structure of weather data by D-VRNN. Transitions with $P \geq 0.2$ are visualized.

In Fig. 4 and 5, we compare our models quantitatively with the TM-HMM model with 10 topics and 20 topics from Zhai and Williams (2014), which performs the best on a similar task-oriented dialog dataset, the DSTC1 . The y-axis shows the negative log likelihood of reconstructing the test set under a specific model. The lower the negative log likelihood, the better the model performs. The x-axis shows different numbers of latent states N used in the models. As we can see, all of the VRNN-based models surpass the TM-HMM models by a large margin and are more invariant to the change in N on both datasets. Especially when N is small, the performance of HMM is not stable.

Qualitatively, we compare the dialog structures generated by different models. Fig. 2 and 3 show the discovered dialog structures using the D-VRNN model, and Fig. 7 in the Appendix shows the dialog structures learned by HMM. Each circle in these figures represents a latent state $\mathbf{z}_t$ with expert interpreted meanings, and the directed-arrows between the circles represent the transition probabilities between states. Human experts interpret each $\mathbf{z}_t$ consistently by going through conversational exchanges assigned to the same $\mathbf{z}_t$. For a better visualization effect, we only visualize the transitions with a probability equal or greater than 0.2 in the figures.

We observe reasonable dialog structures in Fig. 2 and 3. The D-VRNN captures the major path *looking for restaurant (anything else)* → *get restaurant address and phone* → *thank you* in the restaurant search task. It also captures *what's the weather* → *place and time* → *api call* in the weather report task. However, we do not get a dialog structure with entities separated (such as *food type I like* → *area I prefer* → *price range I want* → ...). Because users know the system's capability and tend to give as many entities as possible in a single utterance, so these entities are all mingled in one dialog exchange. But the model is able to distinguish the "presenting match result" state from the "presenting no match result" state (on the top of Fig. 2), which is important in making correct predictions. But in Fig. 7(a) generated by HMM, even if we set 10 states in the HMM, some states are still collapsed by the model because they share a similar surface form. And in Fig. 7(b) also by HMM, the dialog flow skips the "what can I do" state, and goes from the "start" directly to "providing place and time", which is not reasonable.

Another interesting phenomenon is that there are two "thank you concentrated" states in Fig. 2. This is because, users frequently say "thank you" on two occasions, 1) after the dialog system presents the restaurant information, most users will say "thank you"; 2) then the system will ask "is there anything else I can help you with", after which users typically respond with "thank you, that's all". This interesting structure is a reflection of the original rules in the dialog system. Moreover, in Fig. 3, we see 1) transitions from both directions between states "place and time" and "api call", and 2) transitions from "api call" to itself, as there are simulated speech recognition errors in the data and the system needs to confirm the entity values and update the API query.

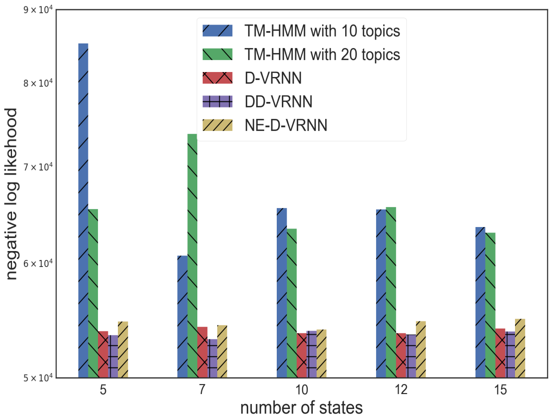


Figure 4: Negative log likelihood on restaurant data.

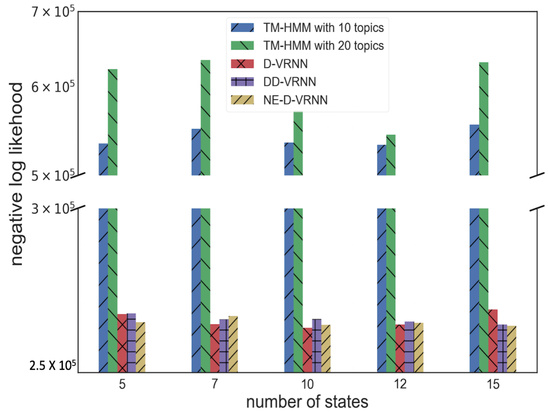


Figure 5: Negative log likelihood on weather data.

5.3 VRNN Model Variants Comparison

Even though the three proposed VRNN-based models have similar structures, they perform differently and are able to compensate each other. For example, in Fig. 8(b) in the Appendix, DD-VRNN is able to recognize a new state "not done yet, what's the weather", when users start a new query. This can complement D-VRNN's result.

Quantitatively, the three model variants also perform differently. On the restaurant test set shown in Fig. 4, DD-VRNN has the best overall performance compared with other models. Especially when the number of states N is small (e.g. 5 or 7 states), the advantage of the direct transition is more obvious. We think the reason behind is that it's easier and more accurate to model the direct transitions between a smaller set of states; as we increase N, the direct transitions between states become less and less obvious and therefore, help less on the predictive power. To our surprise, putting more weights on the named entities has a negative impact on the performance on the restaurant dataset. The underlying reason might be that the emphasis on the named entities shifts the focus of the model from abstract latent representation to a more concrete word token level. However, on the simulated weather dataset shown in Fig. 5, NE-D-VRNN performs relatively well. It might be because the weather dataset is completely simulated, which makes it easier to recognize the named entities. With a more accurate named entity recognition, NE-D-VRNN is able to help the performance. Overall, D-VRNN is the most stable one across datasets, so we will use D-VRNN in the following experiments.

6 Application on RL

The ultimate goal of such a structure discovery model is to utilize the structure to facilitate downstream tasks, such as dialog system training. Therefore, we propose to incorporate the dialog structure information into the reward function of the RL training. We believe that the transition table learned from the original dataset will guide the policy to make better decisions and converge faster by encouraging the policy to follow the real-data distribution. Similar to other RL models, we build a user simulator to perform the RL training. Please refer to the Appendix for the training details.

6.1 Reward Function Design

We use *policy gradient* method (Williams, 1992) to train the dialog policy. Traditional reward functions give a positive reward (e.g. 20) after the successful completion of the task, 0 or a negative reward to penalize a failed task and -1 at each extra turn to encourage the system to finish the task sooner rather than later (Williams et al., 2017). But this type of delayed reward functions doesn't have immediate rewards at each turn, which makes the model converge slowly. Therefore, we propose to incorporate the learned conversational exchange transition information as an immediate reward. The intuition is that in order to complete a task sooner, most users will follow a certain pattern when interacting with the system, for example, a typical flow is that users first give the entities information such as location, then ask for the restaurant information and finally, end the conversation. If we can provide the RL model with the information on what action is more likely to follow another action, the model can learn to follow real-data distributions and make better predictions. We encode the transition information through KL-divergence, a measurement of the distance between two distributions, in the reward function.

We design four types of reward functions and describe each of them in Algorithm 1 and Eq. 10 in the Appendix. The traditional delayed reward is the baseline. The second reward function, Rep-reward, uses constant penalty for repeated questions, as penalizing repetition yields better results, according to Shi and Yu (2018). The third reward function, KL-reward, incorporates the transition table information. From the RL model, we get the predicted probability p_{pred} for different actions; from the D-VRNN model, we get the transition probability p_{trans} between states and each state is translated to an action. We calculate the negation of the KL-divergence between p_{trans} and p_{pred} and use it as the immediate reward for every turn. This immediate reward links the predicted distribution with the real-data distribution by calculating the distance between them. The fourth reward function (KL+Rep) gives an additional -2 repetition penalty to the KL-reward to test the combination effect of the two types of penalties.

6.2 Result Analysis

We evaluate the RL performance by the average success rate, shown in Fig. 6. We observe that all the experimental reward functions greatly improve the performance of the baseline. We also observe that the baseline has a higher variance, which is due to the inefficient delayed rewards. Moreover, both the KL-reward and the KL-Rep reward reach a higher success rate at around 10,000 iterations than the Rep-reward and converges faster. These two reward functions also achieve a better convergent success rate after 10,000 iterations. This suggests that adding KL-divergence of p_{trans} and p_{pred} into the reward helps develop a better policy faster.

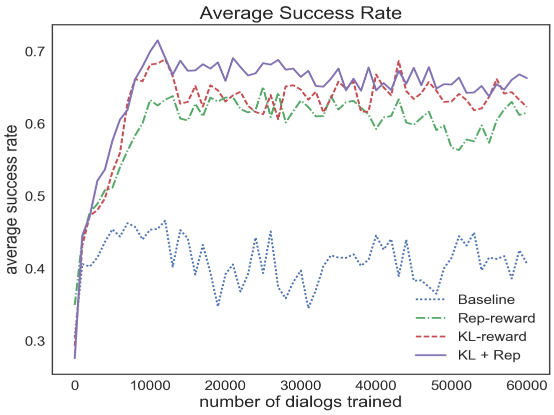


Figure 6: Average success rate of RL models with different reward functions.

The good performance comes from the transition probability p_{trans} learned by the discrete-VRNN. p_{trans} summarizes the communication pattern of most users in the real world and the KL-divergence measures the distance between the predicted distribution p_{pred} and p_{trans} from the real dataset. The RL model processes this KL-reward signal and learns to minimize the gap between p_{trans} and p_{pred}. As a result, p_{pred} will follow closer to the real data distribution, leading the model to converge faster to a better task success rate. In this way, we successfully incorporate the dialog structure information from the real data into the RL system training.

We observe that the use of KL reward improves the performance significantly in terms of both convergence rate and the final task success rate. Further, the combination of KL and repetition reward makes the model more stable and achieves a better task success rate, compared with the model with only KL-reward or Rep-reward. This indicates that the KL-reward can be combined with other type of rewards to achieve a better performance.

7 Conclusion and Future Work

A key challenge for discourse analysis and dialog system building is to extract the latent dialog structure. We adopted the VRNN with discrete latent variables to learn the latent states of each conversational exchange and the transitions between these states in an unsupervised fashion. We applied the algorithm on a restaurant search task and a simulated weather report task, and evaluated the model quantitatively and qualitatively. We also proposed a way to incorporate the learned dialog structure information into a downstream dialog system building task. We involved the dialog structure in the RL reward design, which made the model converge faster to a better task success rate.

The performance of the Discrete-VRNN model has a major impact on the performance of the policy training. We plan to further improve the dialog structure learning process. Currently, we try to capture the status of the named entities by increasing the weights on the entities, which focus on the concrete word token level. In the future, we may use more sophisticated ways to encode the entity information into the latent states.

References

Ananlada Chotimongkol. 2008. *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs*. Ph.D. thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Emmanuel Ferreira and Fabrice Lefèvre. 2013. Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 108–113. IEEE.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

R Chulaka Gunasekara, David Nahamoo, Lazaros C Polymenakos, Jatin Ganhotra, and Kshitij P Fadnis. 2017. Quantized-dialog language model for goal-oriented conversational systems. *Dialog State Tracking Challenge 6, COLIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jingjing Liu, Stephanie Seneff, and Victor Zue. 2010. Dialogue-oriented review summary generation for spoken dialogue recommendation systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 64–72. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. *arXiv preprint arXiv:1706.00359*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Gabriel Murray, Steve Renals, and Jean Carletta. 2005. Extractive summarization of meeting recordings.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. *arXiv preprint arXiv:1804.10731*.

Michel Tokic. 2010. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Steve Young. 2006. Using pomdps for dialog management. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 8–13. IEEE.

Ke Zhai and Jason D Williams. 2014. Discovering latent structure in task-oriented dialogues. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 36–46.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. *arXiv preprint arXiv:1805.04803*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

A Appendices

A.1 Reward Functions in RL

Algorithm 1 Reward function

```
if success then
    $R = 20$
else if failure then
    $R = -10$
else if repeated question then
    $R = f_{\text{reward}}(p_{\text{trans}}, p_{\text{pred}})$
else if each proceeding turn then
    $R = -1$
end if
```

$$f_{\text{reward}}(p_{\text{trans}}, p_{\text{pred}}) = \begin{cases} -1 & \text{Baseline} \\ -5 & \text{Rep-reward} \\ -\text{KL}(p_{\text{trans}}, p_{\text{pred}}) & \text{KL-reward} \\ -\text{KL}(p_{\text{trans}}, p_{\text{pred}}) - 2 & \text{KL+Rep} \end{cases} \tag{10}$$

A.2 RL Simulation Setting

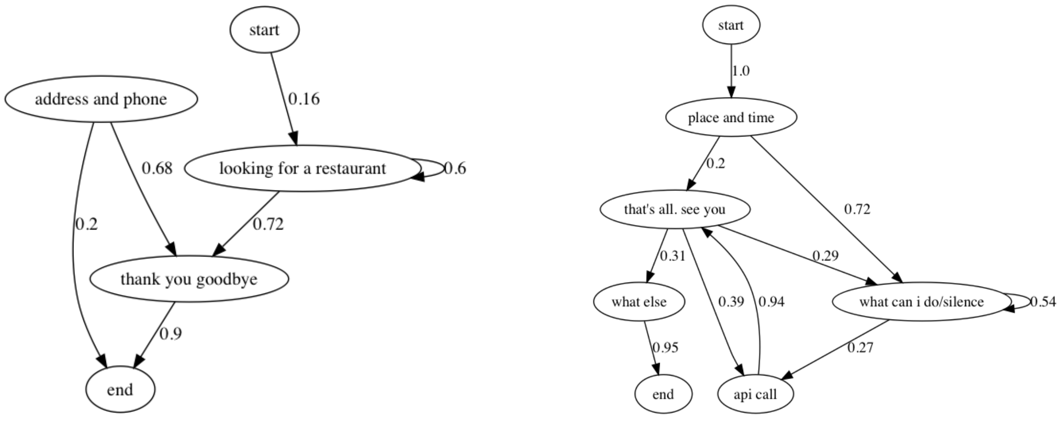
The main purpose of the user simulator is to provide named entities. Therefore, it is built by sampling utterances from the real dataset according to the current status of named entities. Table 2 shows one simulated dialog in the reinforcement learning. We design seven system template actions in the simulated setting which fall under four categories, *ask for entity*, *present results*, *give restaurant information* and *closing statement*. But there are n latent states learned from the real data, and n is not necessarily equal to four. To solve this mismatch in dimensions between p_{pred} and p_{tran}, human experts further collapse similar latent states and reduce the number of latent states to four, which represents the four main categories in the actions, *ask for entity*, *present results*, *give restaurant information* and *closing statement*. For example, the two "thank you concentrated" states in Fig. 2 will be combined to the *closing statement* category. The probability of each category is the sum of the original probabilities whose actions are now under the same category.

From	Utterance
USR:	Could you please help me find an [value_price] restaurant in the [value_area] [slot_area]?
SYS:	Do you have a [slot_food] preference?
USR:	Let's try [value_food], please
SYS:	[value_name] is a good restaurant matching your request. Is there anything else I can help you with?
USR:	What other options are there?
SYS:	[value_name] is a good restaurant matching your request. Is there anything else I can help you with?
USR:	Great. what is their [slot_phone]?
SYS:	Here is the info...

Table 2: A simulated dialog.

A.3 RL Training Details

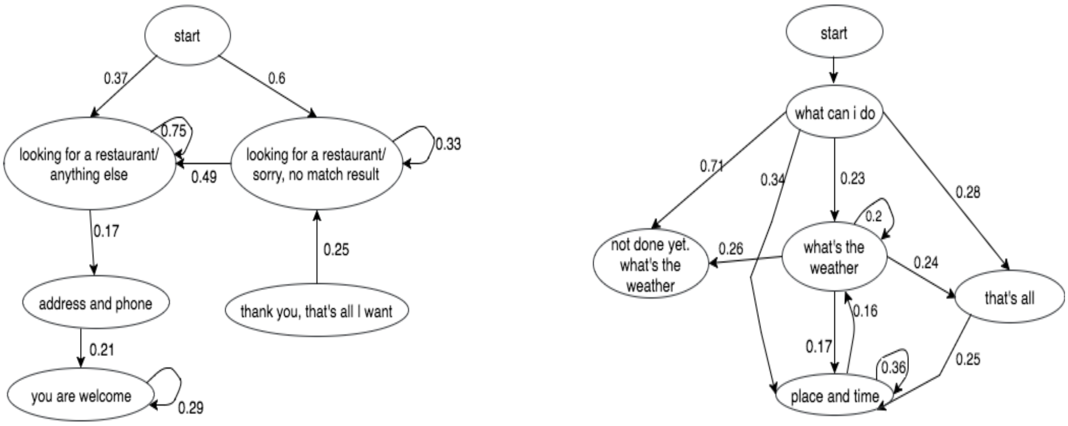
A simple action mask is used to prevent impossible actions, such as presenting results without making a query to the DB. The input to the model is a list of common contextual features, such as the presence of each entity and the last action taken. The output of the model is the system action template.



(a) HMM, restaurant data, 5 states

(b) HMM, weather data, 10 states

Figure 7: Dialog structures generated by HMM on different datasets. Transitions with $P \geq 0.2$ are visualized.



(a) DD-VRNN, restaurant data, 5 states

(b) DD-VRNN, weather data, 10 states

Figure 8: Dialog structures generated by DD-VRNN on different datasets. Transitions with $P \geq 0.2$ are visualized.

Four different reward functions are used. A discount factor of 0.9 is applied to all the experiments and the maximum number of turns is 10. An LSTM with 32 hidden units is used and the RL policy is updated after each dialog. We also apply ϵ-greedy exploration strategy (Tokic, 2010). Because the RL training can sometimes be unstable, we initialize all the model parameters using supervised learning on a simulated dataset, which consists of 500 dialogs between the user simulator and a rule-based agent simulator. The method is evaluated by freezing the policy after every 1000 updates and running 200 simulated dialogs to calculate the task success rate. We repeat the entire process 10 times and report the average success rate in Fig. 6.

A.4 Dialog Structures by Different Models

Fig. 7 and 8 show the dialog structures generated by HMM and DD-VRNN.

Modeling Document-level Causal Structures for Event Causal Relation Identification

Lei Gao, Prafulla Kumar Choubey, Ruihong Huang
Department of Computer Science and Engineering
Texas A&M University
`(sjtuprog, prafulla.choubey, huangrh)@tamu.edu`

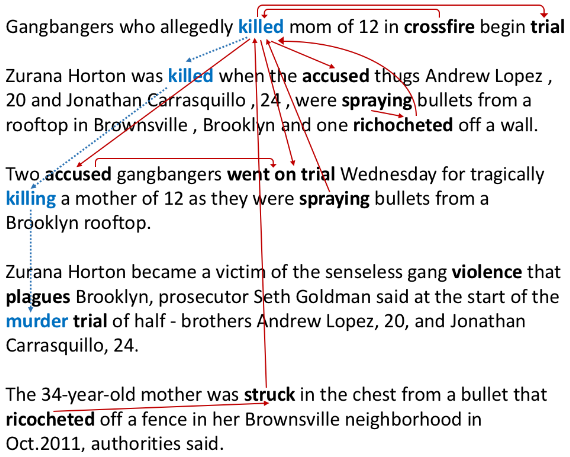
Abstract

We aim to comprehensively identify all the event causal relations in a document, both within a sentence and across sentences, which is important for reconstructing pivotal event structures. The challenges we identified are two: 1) event causal relations are sparse among all possible event pairs in a document, in addition, 2) few causal relations are explicitly stated. Both challenges are especially true for identifying causal relations between events across sentences. To address these challenges, we model rich aspects of document-level causal structures for achieving comprehensive causal relation identification. The causal structures include heavy involvements of document-level main events in causal relations as well as several types of fine-grained constraints that capture implications from certain sentential syntactic relations and discourse relations as well as interactions between event causal relations and event coreference relations. Our experimental results show that modeling the global and fine-grained aspects of causal structures using Integer Linear Programming (ILP) greatly improves the performance of causal relation identification, especially in identifying cross-sentence causal relations.

1 Introduction

Understanding causal relations between events in a document is an important step in text understanding and is beneficial to various NLP applications, such as information extraction, question answering and text summarization. Causal relations can occur between any two events in a document, both between events within a sentence and between events across sentences. In this paper, we aim to identify all the event causal relations in a document.

The main challenges for achieving comprehensive causal relation identification are that event causal relations are sparse among all the event pairs in a document and few event causal relations are explicitly stated. The challenges are especially true for identifying cross-sentence event causal relations and most of them have no clear causal indicators. To address these challenges, we model rich aspects of document-level causal structures, i.e., structural distributions of causal relations within a document, for achieving comprehensive causal relation identification in news articles.



Gangbangers who allegedly **killed** mom of 12 in **crossfire** begin **trial**

Zurana Horton was **killed** when the **accused** thugs Andrew Lopez , 20 and Jonathan Carrasquillo , 24 , were **spraying** bullets from a rooftop in Brownsville , Brooklyn and one **richocheted** off a wall.

Two **accused** gangbangers **went on trial** Wednesday for tragically **killing** a mother of 12 as they were **spraying** bullets from a Brooklyn rooftop.

Zurana Horton became a victim of the senseless gang **violence** that **plagues** Brooklyn, prosecutor Seth Goldman said at the start of the **murder** **trial** of half - brothers Andrew Lopez, 20, and Jonathan Carrasquillo, 24.

The 34-year-old mother was **struck** in the chest from a bullet that **ricocheted** off a fence in her Brownsville neighborhood in Oct.2011, authorities said.

Figure 1: An Example of Main Event Causal Structure

Our key observation for improving causal relation identification is that causal relations, especially cross-sentence causal relations, tend to involve one or two main events of a document. The main events are the focus of a story, which are usually mentioned in the title of an article and have repeated mentions throughout the document. Intuitively, causal relations in a document are often used to explain why the main events happened as well as consequences of the main events. For example, as shown in figure 1, **killing** is the main event. The events **crossfire, spraying, richocheted, struck** are its preconditions, and **accuse, trial** are its consequences. Indeed, many causal

Proceedings of NAACL-HLT 2019, pages 1808–1817
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

relations are related to the main event.

In addition to the global causal structures related to main events of a document, we model three types of fine-grained causal structures in order to accurately identify each individual causal relation. First, specific sentential syntactic relations may evoke causal relations between event pairs. For instance, adverbial clause modifier of a verb phrase explains its consequence, condition or purpose. Second, we model implications of a discourse relation between two text units (e.g., the *contingency* discourse relation) towards causal relations between events in the two text units. Third, we model interactions between event causal relations and event coreference relations. For example, coreferent event mentions should have the same causal relations; a causal relation and an identity relation should not co-exist between any two events.

We use Integer Linear Programming (ILP) to model these rich causal structures within a document by designing constraints and modifying the objective function to encourage causal relations akin to the observed causal structures and discourage the opposite. Our experimental results on the dataset EventStoryLine (Caselli and Vossen, 2017) show that modeling the global and fine-grained aspects of causal structures within a document greatly improves the performance of causal relation identification, especially in identifying cross-sentence causal relations.

2 Related Work

In the last decade or so, both unsupervised and supervised causal relation identification approaches have been proposed including linguistic patterns, statistical measures and supervised classifiers, primarily with the goal of acquiring event causality knowledge from a text corpus. The proposed approaches mainly rely on explicit contextual patterns (Girju; Hashimoto et al., 2014) or other causality cues (Riaz and Girju, 2010; Do et al., 2011), statistical associations between events (Beamer and Girju, 2009; Hu et al., 2017; Hu and Walker, 2017; Do et al., 2011; Hashimoto et al., 2014), and lexical semantics of events (Riaz and Girju, 2013, 2014b,a; Hashimoto et al., 2014).

An increasing amount of recent works focused on recognizing event causal relations within a document, but mostly limited to identifying intra-sentence causal relations with explicit causal indicators. Mirza et al. (2014) annotated event causal relations in the TempEval-3 corpus and created CausalTimeBank. Mirza and Tonelli (2014) stated that incorporating temporal information improved the performance of a causal relation classifier. Mirza and Tonelli (2016) built both a rule-based multi-sieve approach and a feature based classifier to recognize causal relations in CausalTimeBank. However, causal relations in CausalTimeBank are few and only explicitly stated intra-sentence causal relations were annotated. In addition, Mostafazadeh et al. (2016) annotated both temporal and causal relations in 320 short stories (five sentences in each story) taken from the ROCStories Corpus and indicated strong correlations between causal relations and temporal relations.

Lately, Caselli and Vossen (2017) created a corpus called EventStoryLine, which contains 258 documents and more than 5,000 causal relations. The EventStoryLine corpus is the largest dataset for causal relation identification till now with comprehensive event causal relations annotated, both intra-sentence and cross-sentence, which presents unique challenges for causal relation identification. Caselli and Vossen (2017) showed that only 117 annotated causal relations in this dataset are indicated by explicit causal cue phrases while the others are implicit. We conduct experiments on the EventStoryLine dataset. Distinguished from most of the previous approaches that identify one causal relation each time, we model coarse-grained and fine-grained document-level event causal structures and infer all the causal relations in a document.

Integer linear programming (ILP) approaches have been applied to predict a set of temporal relations or an event timeline in a document (Do et al., 2012; Teng et al., 2016; Ning et al., 2017). ILP has been used to improve causal relation identification (Do et al., 2011), but only with fine-grained constraints considering discourse relations between two text units. Our approach innovates on modeling other aspects of document-level causal structures, especially heavy involvements of main events in causal relations, that facilitate resolving multiple causal relations.

3 The EventStoryLine Corpus

Table 1 shows the statistics of the corpus EventStoryLine v0.9[1] (Caselli and Vossen, 2017).

Item	Size
Topics	22
Documents	258
Sentences	4,316
Event Mentions	5,334
Intra-sentence causal links	1,770
Cross-sentence causal links	3,855
The Total causal links	5,625
Explicit causal links	117

Table 1: EventStoryLine v0.9

Causal relations annotated in EventStoryLine are between two event mentions. Different causal relations are annotated in EventStoryLine, called "rising_action" and "falling_action", which indicate the directions of causal relations and intuitively correspond to "precondition" and "consequence" relations. Note that in this paper, we focus on identifying all the pairs of events in a document that are causally related, but not on classifying the direction of a causal relation though; specifically, we aim to recognize if there exists a causal relation between any two events A and B in a document, but we do not further distinguish if A causes B vs. B causes A.

On average, there is 1.2 event mentions in each sentence. There are 7,805 intra-sentence and 46,521 cross-sentence event mention pairs in total in the corpus, around 22% (1,770) and 8% (3,855) of them were annotated with a causal relation respectively. Out of the annotated causal links, only 117 Caselli and Vossen (2017) causal relations are indicated by explicit causal cue phrases while the others are implicit. In our experiments, we use the gold event mentions in EventStoryLine and exclude aspectual, causative, perception and reporting event mentions[2], most of which were not annotated with any causal relation according to Caselli and Vossen (2017).

4 The Feature Based Local Pairwise Classifiers

Intra- and cross-sentence causal relations are different by nature. For instance, dependency relations between words in a sentence may be more useful for detecting intra-sentence causal relations, than when used for detecting cross-sentence causal relations. Therefore, we train two separate logistic regression classifiers, one for intra-sentence causal link detection and the other for cross-sentence causal link detection.

We consider all event mention pairs within a sentence as training instances for the intra-sentence causal relation classifier. Then we pair event mentions from two sentences with one event mention from each sentence, which are used as training instances for the cross-sentence classifier. Note that training instances for both classifiers are unbalanced, with a POS:NEG ration of around 1:3 and 1:10 for intra- and cross-sentence cases respectively. We applied the "balanced" class weight option[3] in logistic regression classifiers to deal with the class imbalance problem.

We use the same set of features for training both classifiers, but we expect the two classifiers to assign different weights to features.

4.1 The Common Feature Set

Lexical Features: We implement rich lexical features to capture event word forms and similarities between two events, event modifiers and event arguments. First, we encode word and lemma for each token in two event phrases as features. Then we created various similarity features between two events.

- Similarities Based on Event Word Form Match. Three binary features indicating whether the lowercases of two event head words, two event head lemmas and two complete event phrases are exactly the same.
- Similarities Based on Wordnet. We first identify synsets for each event head word in Wordnet. Then for each pair of synsets, with one synset for each event head word, we calculate the Wup similarity (Wu and Palmer, 1994). We create numerical features using the average, minimal and maximal Wup similarities.
- Similarities Based on Word Embeddings. We apply l2 normalization on event head word embeddings, and then we calculate the Euclidean distance and Cosine distance between

---

[1] Statistics are calculated based on latest release https://github.com/tommasoc80/EventStoryLine

[2] 639 event mentions were excluded in this way.

[3] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

two word embeddings and use them as features. We use Glove Vectors (Pennington et al., 2014) for word embeddings.

- Similarities Based on Event Modifiers. We run the dependency parsing tool from the Stanford CoreNLP (Manning et al., 2014) and identify event modifiers as words that have a certain dependency relation[4] with an event head word. We measure the similarity between two events using the number of common modifiers and the number of common dependency relations that connect a modifier with an event head word.
- Similarities Based on Event Arguments. We consider entities that have a direct dependency relation with an event head word as its event arguments. We use the Stanford CoreNLP to identify entities and their types. We measure the similarity between two events using the number of common event arguments and the number of common entity types.

Causal Potential Features: As inspired by the causal potential metric proposed by (Beamer and Girju, 2009), we encode features based on the point-wise mutual information (PMI) score and the relative textual order between two events. We calculate the PMI score of two event words in EventStoryLine by using co-occurrences of two events in one sentence, and we use the score as a numerical feature.

Syntactic Features: We use dependency relations on the dependency path between two events. We use the basic dependencies extracted from StanfordCoreNLP (Manning et al., 2014). For cross-sentence event pairs, we consider the dependency path from each event to the root node in its own sentence in extracting dependency relations, following Cheng and Miyao (2017). In addition, we use Part Of Speech tags of two event head words as features.

4.2 Score Replacement

We observed that the cross-sentence causal relation classifier is usually not as capable as the intra-sentence classifier, probably due to less contextual evidence to rely on. Therefore, for cross-sentence event mention pairs that can be converted to intra-sentence cases through event coreference links, we use a heuristic method to improve causal relation prediction performance and replace the predictions from the cross-sentence classifier with the predictions from the intra-sentence classifier[5], by using system predicted event coreference links. Note that two events may have more than one pair of mentions, one mention for each event, co-occur within one sentence, we will use the highest score produced by the intra-sentence classifier over all the event mention pairs.

In addition, the score replacement procedure may change prediction scores of some intra-sentence event mention pairs as well. For instance, if one event mention has a coreferent mention within the same sentence that is closer to and is more clearly in a causal relation with the other event mention according to the intra-sentence classifier, and when paired up, the new event pair has received a higher score, then we will replace the score of the original event pair with the higher score. We implemented the within-document neural network based event coreference classifier as described in (Choubey and Huang, 2017a) and used the system to obtain event coreference links.

5 Modeling Causal Structures Using ILP

Our Integer Linear Programming (ILP) system performs document level global inference for resolving all the intra-sentence and inter-sentence event causal relations in a document. Let p_{ij} denotes confidence score from the corresponding local pairwise classifier for assigning a causal relation to the event pair (i, j). Let μ refer to the set of event mentions in a document, we formulate our basic ILP objective function with equation 1.

$$\Theta_{Basic} = \max \sum_{i \in \mu} \sum_{j \in \mu} [p_{ij} x_{ij} + \neg x_{ij}(1 - p_{ij})] \quad (1)$$

We then augment the objective function with new objectives (equation 2) and add constraints to induce causal structures, including heavy involvements of main events (Θ_M and Θ_F) in causal relations throughout the document, as well as fine-grained interactions of event causal relations with discourse relations (Θ_D), and event coreference

---

[4] Specifically, we consider 'nmod', 'amod', 'advmod', 'mark', 'aux', 'auxpass', 'expl', 'cc', 'cop', 'punct' to be modifiers.

[5] Note that we only conduct the score replacement when a score produced by the intra-sentence classifier is higher than the score produced by the cross-sentence classifier, which indicates that the intra-sentence classifier is more confident.

relations(Θ_C) as well as syntactic structure constraints (Θ_S) for identifying causal relations.

$$\Theta = \Theta_{Basic} + \Theta_M + \Theta_F + \Theta_D + \Theta_C + \Theta_S \qquad (2)$$

5.1 Document Level Main Event Based Constraints

Main Event: Main events are central to the story in a document and tend to participate in multiple causal links. Similar to Choubey et al. (2018), we recognize main events based on characteristics of event coreference chains within a document. Specifically, we rank events based on the number of event mentions referring to an event, and choose the top two events as main events[6]. Then we add a new objective function (equation 3) and additional constraints to encourage causal links in event mention pairs containing a main event (equation 4) and discourage causal links in the remaining mention pairs (equation 5).

$$\begin{aligned}\Theta_M = \max \Big[&\sum_{i\in\Lambda} [k_{m_1} m_1(i) + k_{m_2} m_2(i)] \\ - &\sum_{i\in\mu-\Lambda} [k_{n_1} n_1(i) + k_{n_2} n_2(i)] \Big]\end{aligned} \qquad (3)$$

$$\begin{aligned}\forall i \in \Lambda, &\sum_{j\in\mu, d_i = d_j} x_{ij} \geq m_1(i) \\ \forall i \in \Lambda, &\sum_{j\in\mu, d_i \neq d_j} x_{ij} \geq m_2(i)\end{aligned} \qquad (4)$$

$$\begin{aligned}\forall i \in \mu - \Lambda, &\sum_{j\in\mu-\Lambda, d_i = d_j} x_{ij} \leq n_1(i) \\ \forall i \in \mu - \Lambda, &\sum_{j\in\mu-\Lambda, d_i \neq d_j} x_{ij} \leq n_2(i)\end{aligned} \qquad (5)$$

In the above equations, Λ denotes the set of main event mentions, and d_i denotes the sentence number for event i. The independent variables $m_1(i)$ and $m_2(i)$ indicate the minimum number of intra- and cross-sentence causal relations that main events participate in. By maximizing $m_1(i)$ and $m_2(i)$ in the objective function Θ_M, our model favors main events to have more causal relations. Similarly, variables $n_1(i)$ and $n_2(i)$ in equation 5 are separately defined to set upper thresholds on the maximum number of intra- and cross-sentence causal relations without a main event. Unlike $m_1(i)$ and $m_2(i)$, we aim to minimize the variables $n_1(i)$ and $n_2(i)$ to restrict non-main events from participating in causal relations. Notice that we apply the constraints separately to intra- and cross-sentence mention pairs. This is primarily because main events are likely to participate in many more cross-sentence causal relations compared to intra-sentence cases. Furthermore, we observe that a main event may trigger several consequent events which themselves are causally related. However, causal relations involving only non-main events are less likely to show transitivity. Therefore, we add the constraint 6 to ensure non-transitivity among causal relations with no main event.

$$x_{ij} + x_{jk} + x_{ik} \leq 2 + 1_{i\in\Lambda} + 1_{j\in\Lambda} + 1_{k\in\Lambda} \qquad (6)$$

Locality Constraints: Main events may not always have the largest coreference chain size, and the position of an event mention provides another strong heuristics for identifying the main event (Upadhyay and Roth, 2016). In addition, the first sentence often summarizes the central context of story and are likely to describe foreground events (Grimes, 1975) that have causal relation with multiple other events. Therefore, we add an objective function (equation 7) and additional constraints (equation 8) to encourage causal relations that contain an event from the first sentence.

$$\Theta_S = \max \sum_{i\in S} k_f b_1(i) - \sum_{i\in\mu} \sum_{j\in\mu} k_f l_{ij} \cdot |d_i - d_j| \qquad (7)$$

$$\forall i \in F, \sum_{j\in\mu} x_{ij} \geq b_1(i) \qquad (8a)$$

$$\forall < i, j > \in M, x_{ij} \leq l_{ij} + 1_{i\notin\{F\}} \wedge 1_{j\notin\{F\}} \qquad (8b)$$

where, F represents all the events in first sentence, independent variable $b_1(i)$ indicates the minimum number of causal relations that an event in F participates in, M represents the set of event mention pairs that can be mapped to the same sentence and l_{ij} is a leakage variable that allows distant event mentions in F receiving a very high confidence value p_{ij} to have a causal relation. Particularly, we encourage causal links between two event mentions that are in nearby sentences or can

---

[6] If there is a tie between two event clusters with the same number of coreferential event mentions, we use the sum of confidence scores for pairs of coreferential event mentions in a cluster to break the tie. The confidence scores were assigned by the local pairwise coreference relation classifier.

be mapped to the same sentence using coreference links[7]. By maximizing the variables $b_1(i)$ and minimizing the term $l_{ij} \cdot |d_i - d_j|$, we encourage event mentions in F complying with certain constraints to have more causal relations.

5.2 Fine-grained Causal Structure Constraints

Syntactic Relations: Specific sentential syntactic relations may evoke causal relations between event pairs. First, adverbial clause modifier of a verb phrase explains its consequence, condition or purpose; Second, nominal events mentioned as subject in the main clause presents an assertional structure that delivers foreground (Grimes, 1975) information which may have causal associations with other events; Third, non-finite verb events that share arguments and complement the main event of a sentence are likely to have causal associations with the main event.

Therefore, we add an objective function (equation 9) and additional constraints (equation 10) to encourage causal relations that contain a nominal event as subject or verb event that modifies its parent with advcl or xcomp dependency relations. Here, S represents event mentions that possess one of the above syntactic structures, independent variable $b_2(i)$ indicates the minimum number of causal relations that an event in S participates in. Note that equation 10(b) was modified from 8(b) and allows discounted optimization (with l_{ij}) for events in S that are mappable to the same or nearby sentences.

$$\Theta_S = \max \sum_{i \in S} k_s b_2(i) \tag{9}$$

$$\forall i \in S, \sum_{j \in \mu} x_{ij} \geq b_2(i) \tag{10a}$$

$$\forall < i, j > \in M, x_{ij} \leq l_{ij} + 1_{i \notin \{F,S\}} \wedge 1_{j \notin \{F,S\}} \tag{10b}$$

Discourse Relations: Note that the implications of discourse relations between two text units towards causal relations between events in the two text units have been discussed in the previous work (Do et al., 2011). In this work, we consider three types of discourse relations[8]. First, two subtypes of the *contingency* discourse relation, namely *cause* and *condition*, strongly suggest that causal links exist between events in the two discourse units. On the contrary, the *comparison* discourse relation highlights semantic independence between two discourse units, thus inhibits causal relations between events described in them. Third, all causal relations are inherently temporal. An event that causes another event must necessarily occur before or temporally overlap with the latter. Thus, clauses having one of these temporal discourse relations may also favor causal relations between events in them. We model the above three dependencies between discourse relations and causation through constraints 11 and the objective function 12.

$$\begin{aligned} &\forall r = Contingency, \sum_{i \in arg_1} \sum_{j \in arg_2} x_{ij} \geq 1 \\ &\forall r = Comparison, \sum_{i \in arg_1} \sum_{j \in arg_2} x_{ij} \leq 0 \\ &\forall r = Temporal, \sum_{i \in arg_1} \sum_{j \in arg_2} x_{ij} \geq T(r) \end{aligned} \tag{11}$$

$$\Theta_D = \max \sum_{r = Temporal} k_t T(r) \tag{12}$$

Specifically, we enforce events in clauses with the contingency discourse relation to have at least one event pair with causal relation. Similarly, we inhibit a causal relation between any event pair in clauses with the comparison discourse relation. For events in clauses with a temporal discourse relation, we aim to maximize the number of causal relations without grounding it to any hard lower bound. Here, r denotes the discourse relation between two discourse arguments, arg_1 and arg_2, and *Temporal* refers to the set of temporal discourse relations. We use the pre-trained PDTB discourse parser (Lin et al., 2014) to obtain discourse relations in a document.

Event Coreference Relations: We model interactions between event causal relations and event coreference relations by adding constraints 13 and 14 and an objective function 15.

$$\forall i \equiv j, x_{ij} \leq c_3(i, j) \tag{13}$$

$$\begin{aligned} &\forall i \equiv j, x_{ik} + \neg x_{jk} \leq 1 + c_1(i, j, k) \\ &\forall i \equiv j, \neg x_{ik} + x_{jk} \leq 1 + c_2(i, j, k) \end{aligned} \tag{14}$$

---

[7] Two event mentions are mappable if their respective coreferential event mentions co-occur in at least one sentence.

[8] We use PDTB parser (Lin et al., 2014) to identify three discourse relations.

$$\Theta_C = \max \sum_{i \in \mu} \sum_{j \in \mu} \Big[\sum_{k \in \mu} -k_c(c_1(i,j,k) \\ + c_2(i,j,k)) \Big] - (1 - k_c)(c_3(i,j)) \quad (15)$$

Here $\equiv$ represents the identity (coreference) relation. The constraint 13 ensures that causal relation and coreference relation are mutually exclusive, allowing some violations when $p_{i,j}$ is high. The constraints 14 along with the objective function 15 encourage coreferent event mentions to have a causal relation with the same other event. While this relation between causal and coreference relations is strictly true for gold standard data, we observed that these constraints make the system very sensitive to noise when using system predicted coreference links. Therefore, we added binary leakage variables $c_1(i,j,k)$, $c_2(i,j,k)$ and $c_3(i,j)$ to relax these constraints.By maximizing the negative of leakage variables, we allow our model to overcome this instability.

6 Evaluation

6.1 Experimental settings

There are 22 topics in the EventStoryLine corpus. We put them in order based on their topic IDs and use documents in the last two topics as the development set. We trained the ILP system using the rest 20 topics and tuned parameters based on the system performance on the development set. We report experimental results by conducting 5-fold cross validation on the rest 20 topics. For event causal relation identification, we report precision, recall, and F1-score.

The weighting parameters for constraints, including $k_{m_1}, k_{m_2}, k_{n_1}, k_{n_2}, k_f, k_t, k_c$ and k_s, were first pre-set to be a small number 0.1. We then conducted grid search and searched for the best value for each parameter over the range from 0.1 to 0.5 with a step size of 0.1. The best values for the parameters are $0.2, 0.1, 0.1, 0.5, 0.2, 0.1, 0.1, 0.2$ respectively.

6.2 Baseline Systems

We consider six baseline systems:

OP: a dummy model used in (Caselli and Vossen, 2017) that assigns a causal relation to every event mention pair.

Cheng and Miyao (2017): a dependency path based sequential neural network model that extensively models compositional meanings of the context between two event mentions for causal relation identification. This model was first used for identifying event temporal relations and has been shown effective in identifying both intra- and cross-sentence temporal relations.

Choubey and Huang (2017b): another dependency path based sequential neural network model that was first developed for identifying temporal relations between event mentions within a sentence. We make this model also work for cross sentence cases by merging the root nodes of two dependency trees associated with two separate sentences and extracting a dependency path connecting events across sentences.

So far, there is no well recognized effective approach for causal relation identification within a document. We applied the above two models for causal relation identification considering that causal relations are closely related with certain temporal relations and a causal event must occur before or overlap with the consequence event.

LR (Lexical): the same logistic regression classifier as our local pairwise classifier but using the lexical features only.

LR (Causal Potential): the same logistic regression classifier as our local pairwise classifier but using the causal potential features only.

LR (Full): our local pairwise classifier using the full set of features.

+ Score Replacement: our local pairwise classifier using the full set of features, with the heuristic score replacement procedure applied.

6.3 Experimental Results

The first section of table 2 shows the performance of baseline models on intra- and cross-sentence causal relation identification. The model **OP** labels each event mention pair as causal and suffers from low precisions[9], especially on identifying cross-sentence causal relations. The two dependency path based neural network model **(Cheng and Miyao, 2017; Choubey and Huang, 2017b)** do not perform effectively on identifying causal relations. The performance is especially poor on cross-sentence cases.

The model **LR (Lexical)** improved the precision of causal relation identification but suf-

---

[9] The reason it did not achieve the 100% recall is that we did not consider reporting, causative, perception or aspectual events.

	Intra-sentence			Cross-sentence			Intra + Cross		
Models	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Local Pairwise Models									
OP	22.5	**98.6**	36.6	8.4	**99.5**	15.6	10.5	**99.2**	19.0
Cheng and Miyao (2017)	34.0	41.5	37.4	13.5	30.3	18.7	17.6	33.9	23.2
Choubey and Huang (2017b)	32.7	44.9	37.8	11.3	29.5	16.4	15.5	34.3	21.4
LR (Lexical)	**38.7**	37.0	37.8	24.3	29.1	26.5	**28.2**	31.6	29.8
LR (Causal Potential)	28.2	61.2	38.6	10.7	74.6	18.7	12.9	70.4	21.8
LR (full)	37.6	41.4	39.4	23.8	33.6	27.9	27.4	36.1	31.2
+Score Replacement	37.0	45.2	**40.7**	**25.2**	48.1	**33.1**	27.9	47.2	**35.1**
Modeling Causal Structure using ILP									
+Main Event Constraints	38.1	47.6	42.3	31.5	45.4	37.2	33.4	46.1	38.7
+Locality Constraints	38.0	50.4	43.4	32.1	45.8	37.8	33.9	47.3	39.5
+Syntactic Constraints	37.2	54.8	44.3	32.1	48.6	38.7	33.7	50.6	40.4
+Discourse Constraints	37.4	**55.8**	**44.7**	32.2	**48.7**	38.8	33.8	**51.0**	40.6
+Coreference Constraints	**38.8**	52.4	44.6	**35.1**	48.2	**40.6**	**36.2**	49.5	**41.9**

Table 2: Performance of different models on causal relation identification

fers from low recall. In contrast, the model **LR (Causal Potential)** improved the recall but suffers from low precision. The model **LR (full)** with rich lexical, semantic and syntactic features achieved the best trade-off between precision and recall. **+ Score Replacement** significantly improves the recall and F1-score on identifying cross-sentence causal relations, which also slightly improves the recall of intra-sentence cases. But the precision of causal relation identification remains low, especially on cross-sentence cases.

The second section of table 2 shows the performance of our ILP model after gradually adding each type of constraints. **+Main Event Constraints** shows the performance of the ILP system with constraints encouraging causal relations involving a main event. By modeling this aspect of document-level causal structures, the precision of cross-sentence causal relation identification was clearly improved by around 6.3%. With a small loss on recall, the F1-score was improved by 4.1%. Modeling this document-level causal structure also improves both precision and recall on identifying intra-sentence causal relations, but with a relatively small margin. Compared to the local pairwise model **+ Score Replacement**, the overall F1-score improvement from using global main event constraints is statistically significant with $p<0.05$ (Dietterich, 1998). **+Locality Constraints** strengthens the effects of modeling main events and further improved the performance of both cross- and intra-sentence causal relation identification.

Next, adding sentential syntactic structure based constraints (**+Syntactic Constraints**) recovered additional intra-sentence causal relations and cross-sentence causal relations as well due to score replacement, and improved their recall by 4.4% and 2.8% respectively with little or no drop on precision. Then, after adding discourse constraints (**+Discourse Constraints**), both precision and recall on intra-sentence causal relation identification were slightly improved while the performance on cross-sentence causal relation identification remains roughly the same, this is mainly due to the fact that few cross-sentence discourse relations were identified by the discourse parser we used. Finally, after adding conference constraints (**+Coreference Constraints**), the precision of cross-sentence causal relation identification was increased by 2.9%, with a small loss on recall, the F1-score was improved by 1.8%. Unsurprisingly, the overall performance on intra-sentence causal relation identification was not affected much by coreference constraints since event coreference relations often involve events across sentences. Compared to the model considering global constraints only (the line **+ Locality Constraints**), the overall F1-score improvement from using fine-grained causal structure constraints is statistically significant with $p<0.01$.

To sum up, by modeling the global and fine-grained aspects of causal structures, the performance of both intra- and cross-sentence causal re-

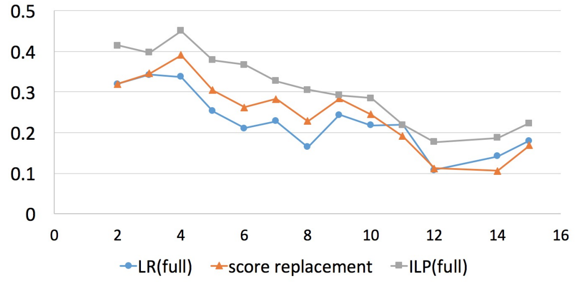


Figure 2: F1-scores on documents with different lengths. The x-axis indicates the number of sentences a document has. The y-axis indicates the macro average F1-score of causal relation identification.

lation identification was greatly improved by 3.9% and 7.5% in F1-score respectively. Compared to the local pairwise model **+ Score Replacement**, the overall F1-score improvement from using both global main event constraints and fine-grained causal structure constraints is statistically significant with $p<0.002$.

Impact of Document Lengths Figure 2 shows performance comparisons of three models on documents with different lengths. The first impression is that causal relation identification becomes harder when documents are longer. If we look into the figure, the score replacement heuristic improves the performance of causal relation identification on medium-sized documents, but not on short ($<$ 4 sentences) or long ($>$ 10 sentences) documents. This may either due to little event coreference information for use in short documents or event coreference information becoming too noisy in long documents. Compared to the mixed effects of the score replacement heuristic, the ILP system improved the performance of causal relation identification consistently in documents of any length, through modeling rich document-level causal structures.

7 Conclusions

We have presented an ILP system that collectively identifies all the causal relations within a document, both intra- and cross-sentence causal relations, by modeling the global and fine-grained aspects of causal structures. In the future, we will continue to enrich document-level causal structures, e.g., by considering segment-wise topic layouts and rhetorical discourse structures.

Acknowledgments

This work was partially supported by the National Science Foundation via NSF Award IIS-1755943. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government.

References

Brandon Beamer and Roxana Girju. 2009. Using a bigram event model to predict causal potential. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 430–441. Springer.

Tommaso Caselli and Piek Vossen. 2017. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional lstm over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 1–6.

Prafulla Kumar Choubey and Ruihong Huang. 2017a. Event coreference resolution by iteratively unfolding inter-dependencies among events. *arXiv preprint arXiv:1707.07344*.

Prafulla Kumar Choubey and Ruihong Huang. 2017b. A sequential model for classifying temporal relations between intra-sentence events. *arXiv preprint arXiv:1707.07343*.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 340–345.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687. Association for Computational Linguistics.

Roxana Girju. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83.

Joseph Evans Grimes. 1975. *The thread of discourse*, volume 207. Walter de Gruyter.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 987–997.

Zhichao Hu, Elahe Rahimtoroghi, and Marilyn A Walker. 2017. Inference of fine-grained event causality from blogs and films. *arXiv preprint arXiv:1708.09453*.

Zhichao Hu and Marilyn A Walker. 2017. Inferring narrative causality between event pairs in films. *arXiv preprint arXiv:1708.09496*.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the tempeval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2097–2106.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th International Conference on Computational Linguistics*, pages 64–75. ACL.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. Caters: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 51–61.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1027–1037.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Mehwish Riaz and Roxana Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 361–368. IEEE.

Mehwish Riaz and Roxana Girju. 2013. Toward a better understanding of causality between verbal events: Extraction and analysis of the causal power of verb-verb associations. In *SIGDIAL Conference*, pages 21–30.

Mehwish Riaz and Roxana Girju. 2014a. In-depth exploitation of noun and verb semantics to identify causation in verb-noun pairs. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 161–170.

Mehwish Riaz and Roxana Girju. 2014b. Recognizing causality in verb-noun pairs via noun and verb semantics. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 48–57.

Jiayue Teng, Peifeng Li, Qiaoming Zhu, and Weiyi Ge. 2016. Joint event co-reference resolution and temporal relation identification. In *Workshop on Chinese Lexical Semantics*, pages 426–433. Springer.

Christos Upadhyay, Shyam @articledietterich1998approximate, title=Approximate statistical tests for comparing supervised classification learning algorithms, author=Dietterich, Thomas G, journal=Neural computation, volume=10, number=7, pages=1895–1923, year=1998, publisher=MIT Press and Christodoulopoulos and Dan Roth. 2016. ”making the news”: Identifying noteworthy events in news articles. In *Proceedings of the Fourth Workshop on Events*, pages 1–7.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

Hierarchical User and Item Representation with Three-Tier Attention for Recommendation

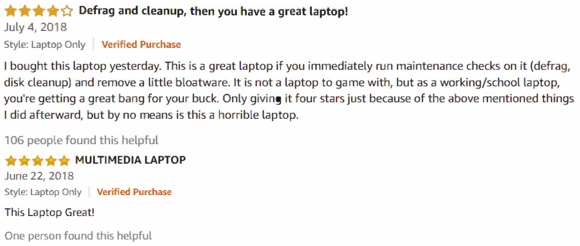
Chuhan Wu[1], **Fangzhao Wu**[2], **Junxin Liu**[1], **and Yongfeng Huang**[1]
[1]Department of Electronic Engineering, Tsinghua University Beijing 100084, China
[2]Microsoft Research Asia
{wuch15,ljx16,yfhuang}@mails.tsinghua.edu.cn
wufangzhao@gmail.com

Abstract

Utilizing reviews to learn user and item representations is useful for recommender systems. Existing methods usually merge all reviews from the same user or for the same item into a long document. However, different reviews, sentences and even words usually have different informativeness for modeling users and items. In this paper, we propose a hierarchical user and item representation model with three-tier attention to learn user and item representations from reviews for recommendation. Our model contains three major components, i.e., a sentence encoder to learn sentence representations from words, a review encoder to learn review representations from sentences, and a user/item encoder to learn user/item representations from reviews. In addition, we incorporate a three-tier attention network in our model to select important words, sentences and reviews. Besides, we combine the user and item representations learned from the reviews with user and item embeddings based on IDs as the final representations to capture the latent factors of individual users and items. Extensive experiments on four benchmark datasets validate the effectiveness of our approach.

1 Introduction

Learning accurate user and item representations is very important for recommender systems (Tay et al., 2018). Many of existing recommendation methods learn user and item representations based on the ratings that users gave to items (Koren et al., 2009; Mnih and Salakhutdinov, 2008). For example, Koren et al. (2009) proposed a matrix factorization method based on SVD to learn latent representations of users and items from the rating matrix between users and items. However, since the numbers of users and items in online platforms are usually huge, and the rating matrix between users and items is usually very sparse, it is quite difficult for those rating based recommendation methods to learn accurate user and item representations (Zheng et al., 2017; Tay et al., 2018).


★★★★☆ **Defrag and cleanup, then you have a great laptop!**
July 4, 2018
Style: Laptop Only | Verified Purchase

I bought this laptop yesterday. This is a great laptop if you immediately run maintenance checks on it (defrag, disk cleanup) and remove a little bloatware. It is not a laptop to game with, but as a working/school laptop, you're getting a great bang for your buck. Only giving it four stars just because of the above mentioned things I did afterward, but by no means is this a horrible laptop.

106 people found this helpful

★★★★★ **MULTIMEDIA LAPTOP**
June 22, 2018
Style: Laptop Only | Verified Purchase

This Laptop Great!

One person found this helpful

Figure 1: Two example reviews.

Luckily, in many online platforms such as Amazon and IMDB, there are rich reviews written by the users to express their opinions on items. These reviews can provide rich information of items. For example, if sentences like “bad battery life” and “battery capacity is low” frequently appear in the reviews of a smartphone, then we can infer the performance of this item in battery life is not good. The reviews also contain rich information of users. For example, if a user frequently mentions “the price is too high” and “very expensive” in his/her reviews for different items, then we can infer this user may be sensitive to price. Thus, these reviews can help enhance the learning of user and item representations especially when ratings are sparse, which is beneficial for improving the performance of recommender systems (Zheng et al., 2017).

Utilizing reviews to learn user and item representations for recommendation has attracted increasing attentions (Zheng et al., 2017; Catherine and Cohen, 2017). For example, Zheng et al. (2017) proposed a DeepCoNN method to learn the representations of users and items from reviews using convolutional neural networks (CNN), and achieved huge improvement in recommendation performance. These methods usually concatenate the reviews from the same user or the

Proceedings of NAACL-HLT 2019, pages 1818–1826
Minneapolis, Minnesota, June 2 - June 7, 2019.

same item into a long document. However, different reviews usually have different informativeness in representing users and items. For example, in Fig. 1 the first review is much more informative than the second one. Distinguishing informative reviews from noisy ones can help learn more accurate user and item representations. In addition, different sentences in the same review may also have different informativeness. For example, in Fig. 1 the sentence “it is not a laptop to game with” contains more important information than “I bought this laptop yesterday”. Besides, different words in the same sentence may also have different importance. For example, in “this is a great laptop if you ...” the word “great” is more important than “you” in modeling this item.

In this paper, we propose a hierarchical user and item representation model with three-tier attention (*HUITA*) to learn informative user and item representations from reviews for recommendation. In our approach, the hierarchical user and item representation model contains three major components, i.e., a sentence encoder to learn sentence representations from words, a review encoder to learn review representations from sentences, and a user/item encoder to learn user/item representations from the all reviews posted by this user or for this item. In addition, we propose to incorporate a three-tier attention network into our model to select important words, sentences and reviews to learn more informative user and item representations. Besides, we combine the user and item representations learned from the reviews with the user and item embeddings based on their IDs as the final representations to capture the latent factors of each individual users and items. We conduct extensive experiments on four benchmark datasets. The results show our approach can effectively improve the performance of recommendation and outperform many baseline methods.

2 Related Work

Learning user and item representations from reviews for recommendation has attracted many attentions (McAuley and Leskovec, 2013; Ling et al., 2014; Bao et al., 2014; Zhang et al., 2014; Diao et al., 2014; He et al., 2015; Tan et al., 2016; Ren et al., 2017). Many of the existing methods focus on extracting topics from reviews to model users and items. For example, McAuley and Leskovec (2013) proposed a Hidden Factors as Topics (HFT) method to use the topic modeling technique LDA to discover the latent aspects of users and items from the reviews. Ling et al. (2014) proposed a Ratings Meet Reviews (RMR) method to enhance the representations of users and items by extracting topics from review texts and aligning the dimensions of these topics with the latent user representations obtained from the rating matrix using matrix factorization. Bao et al. (2014) proposed a TopicMF approach to jointly model user and item representations using rating scores via matrix factorization and using review texts via non-negative matrix factorization (NMF) to obtain topics. However, these methods only extract the topic information from reviews, and a large amount of important semantic information is not captured. In addition, these methods are usually based on topic models and cannot effectively model the contexts and orders of words in reviews, both of which are important for inferring user preferences and item properties.

In recent years, several deep learning based methods have been proposed to learn user and item representations from reviews for recommendation (Zhang et al., 2016; Zheng et al., 2017; Catherine and Cohen, 2017; Seo et al., 2017b,a; Chen et al., 2018; Tay et al., 2018). For example, Zheng et al. (2017) proposed a DeepCoNN method which uses CNN to learn representations of users and items from their reviews. Catherine and Cohen (2017) proposed a TransNets method to learn user and item representations from reviews using CNN and regularize these representations to be close to the representations of the review written by the target user to the target item. Seo et al. (2017b) proposed to learn user and item representations via CNN network as well as attention network over word embeddings. These methods concatenate all the reviews from the same user or for the same item into a long document, and cannot distinguish informative reviews from noisy ones. Chen et al. (2018) proposed to model the usefulness of reviews using review-level attention to enhance the learning of user and item representations. However, their method regards each review as a long sentence, and cannot distinguish informative sentences and words from less informative ones. Different from the aforementioned methods, in our approach we propose a hierarchical framework to learn user and item representations from reviews for recommendation. Our model first learns sentence representations from words, then learns review representations from their sentences, and finally learns user/item representations from their reviews. Our model also contains a three-tier attention network to jointly select important words, sentences and reviews to learn more informative user and item representations. Experiments on benchmark datasets validate the advantage of our approach over existing methods in recommendation.

3 Our Approach

In this section, we introduce our *HUITA* approach to learn user and item representations from reviews for recommendation. The architecture of our approach is shown in Fig. 2. There are three major modules in our approach. The first one is *sentence encoder* which learns representations of sentences from words. The second one is *review encoder* which learns representations of reviews from sentences. And the third one is *user/item encoder*, which learns the representations of users and items from their reviews. Next we introduce each module in detail.

3.1 Sentence Encoder

The *sentence encoder* module is used to learn representations of sentences from words. According to Fig. 2, there are three layers in this module.

The first layer is word embedding. It is used to convert a sequence of words into a sequence of low-dimensional dense vectors which contain semantic information of these words. Denote a sentence s contains M words $[w_1, w_2, ..., w_M]$. Through the word embedding layer the sentence s is transformed into a vector sequence $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M]$ using a word embedding matrix $\mathbf{E} \in \mathcal{R}^{V \times D}$, where V and D represent the vocabulary size and the word embedding dimension, respectively. The word embedding matrix $\mathbf{E}$ is initialized using pretrained word embeddings, and fine-tuned during model training.

The second layer is a convolutional neural network (CNN). CNN is an effective neural architecture for capturing local information (LeCun et al., 2015). We employ a word-level CNN to capture the local contexts of words to learn their contextual representations. Denote $\mathbf{c}_i^w$ as the contextual representation of the word w_i, which is computed as follows:

$$\mathbf{c}_i^w = \mathrm{ReLU}(\mathbf{U}_w \times \mathbf{e}_{(i-K_w):(i+K_w)} + \mathbf{b}_w), \quad (1)$$

where $\mathbf{e}_{(i-K_w):(i+K_w)}$ is the concatenation of the word embedding vectors from the position $i - K_w$ to $i + K_w$. $\mathbf{U}_w \in \mathcal{R}^{N_w \times (2K_w+1)D}$ and $\mathbf{b}_w \in \mathcal{R}^{N_w}$ are the parameters of the filters in CNN network, where N_w is the number of CNN filters and $2K_w + 1$ is the window size. ReLU is the non-linear activation function (Glorot et al., 2011). The output of the CNN layer is a sequence of contextual word representations $[\mathbf{c}_1^w, \mathbf{c}_2^w, ..., \mathbf{c}_M^w]$.

The third layer is a word-level attention network. Different words in the same sentence may have different informativeness for modeling users and items. For example, in the sentence “The laptop I bought yesterday is too heavy”, the word “heavy” is more informative than the word “yesterday” in representing this laptop. Thus, we use a word-level attention network to help our model select and attend to important words based on their contextual representations to build more informative sentence representations for user and item modeling. The attention weight of the i_{th} word in the sentence s is computed as follows:

$$a_i^w = \tanh(\mathbf{v}_w \times \mathbf{c}_i^w + b_w), \quad (2)$$

$$\alpha_i^w = \frac{\exp(a_i^w)}{\sum_{j=1}^{M} \exp(a_j^w)}, \quad (3)$$

where $\mathbf{v}_w \in \mathcal{R}^{N_w}$ and $b_w \in \mathcal{R}$ are the parameters in the attention network. α_i indicates the relative importance of the i_{th} word evaluated by the attention network. The final representation of the sentence s is the summation of the contextual word representations weighted by their attention weights as follows:

$$\mathbf{s} = \sum_{i=1}^{M} \alpha_i^w \mathbf{c}_i^w. \quad (4)$$

3.2 Review Encoder

The *review encoder* module aims to build the representations of each review based on the representation of sentences in these reviews. There are two major layers in the *review encoder* module.

The first layer is a sentence-level CNN network. Neighboring sentences usually have some relatedness with each other. For example, in a laptop review “It is not a laptop to game with. But as a working laptop, you will get a great bang for your buck”, the two neighboring sentences have close relatedness and they both describe the performance of the laptop in different scenarios.

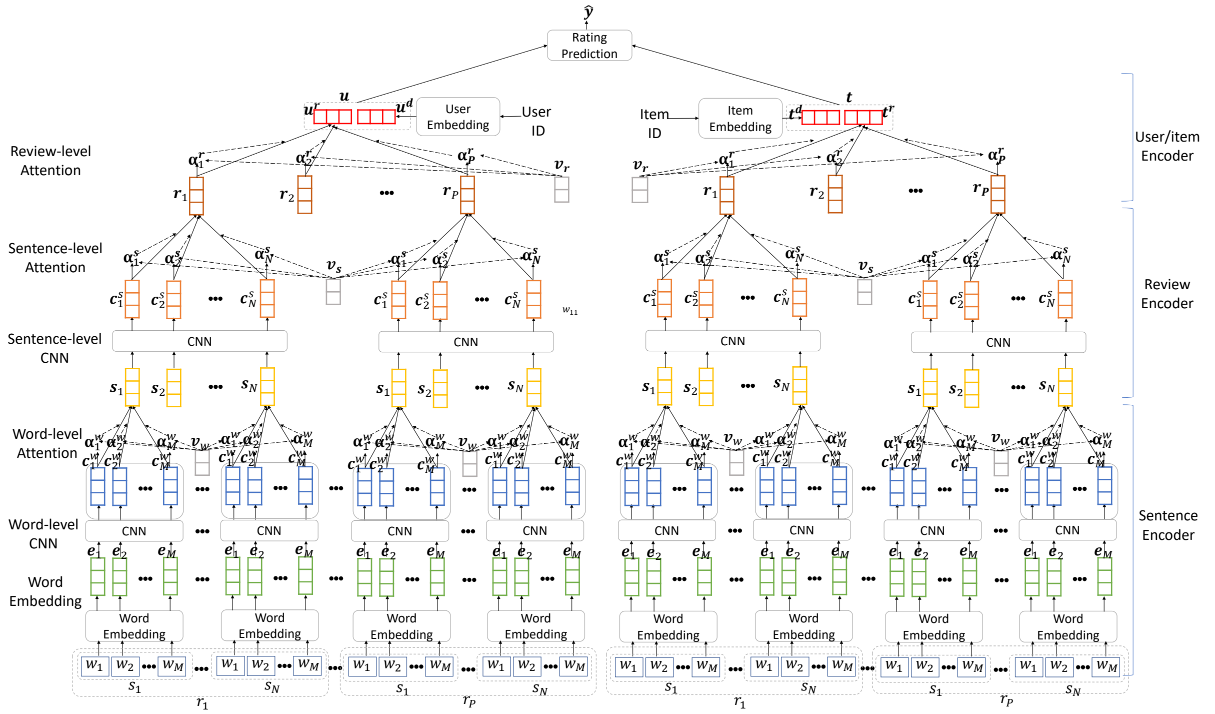


Figure 2: The framework of our *HUITA* approach for recommendation.

Thus, we employ a sentence-level CNN network to learn the contextual sentence representations by capturing the local contexts of sentences. Denote a review r contains N sentences $[s_1, s_2, ..., s_N]$. Denote the contextual representation of sentence s_i as $\mathbf{c}_i^s$, which is computed as follows:

$$\mathbf{c}_i^s = \text{ReLU}(\mathbf{U}_s \times \mathbf{s}_{(i-K_s):(i+K_s)} + \mathbf{b}_s), \quad (5)$$

where $\mathbf{U}_s \in \mathcal{R}^{N_s \times (2K_s+1)N_w}$ and $\mathbf{b}_s \in \mathcal{R}^{N_s}$ are parameters of the sentence-level CNN filters. $\mathbf{s}_{(i-K_s):(i+K_s)}$ is the concatenation of sentence representation vectors from position $i - K_s$ to $i + K_s$. N_s is the number of filters in sentence CNN network and $2K_s + 1$ is the window size.

The second layer is a sentence-level attention network. Different sentences in a review may have different informativeness for modeling users and items. For example, the sentence “it is not a laptop to game with” is more informative than the sentence “I bought this laptop yesterday” in learning the representation of this laptop. Thus, we use sentence-level attention network to help our model select and attend to important sentences to learn more informative review representations. The attention weight of sentence s_i in the review r is formulated as follows:

$$a_i^s = \tanh(\mathbf{v}_s \times \mathbf{c}_i^s + b_s), \quad (6)$$

$$\alpha_i^s = \frac{\exp(a_i^s)}{\sum_{j=1}^{N} \exp(a_j^s)}, \quad (7)$$

where $\mathbf{v}_s \in \mathcal{R}^{N_s}$ and $b_s \in \mathcal{R}$ are the parameters of the attention network. The final contextual representation of the review r is the summation of the contextual representations of sentences weighted by their attention weights, which is formulated as:

$$\mathbf{r} = \sum_{i=1}^{N} \alpha_i^s \mathbf{c}_i^s. \quad (8)$$

3.3 User/Item Encoder

The *user/item encoder* module is used to build the representations of users or items based on the representations of their reviews. Different reviews usually have different informativeness in modeling users or items. For example, in Fig. 1, the first review contains much more information of the laptop than the second review, and should has more contributions in building the representation of this laptop. Thus, we use a review-level attention network to distinguish informative reviews from less informative ones. Denote a user u has P reviews $[r_1, r_2, ..., r_P]$. Then the attention weight of the review r_i is computed as follows:

$$a_i^r = \tanh(\mathbf{v}_r \times \mathbf{r}_i + b_r), \quad (9)$$

$$\alpha_i^r = \frac{\exp(a_i^r)}{\sum_{j=1}^{P} \exp(a_j^r)}, \quad (10)$$

where $\mathbf{v}_r \in \mathcal{R}^{N_s}$ and $b_r \in \mathcal{R}$ are the parameters of the review-level attention network. The user representation learned from the reviews is the summation of the contextual representations of reviews weighted by their attention weights:

$$\mathbf{u}^r = \sum_{i=1}^{P} \alpha_i^r \mathbf{r}_i. \tag{11}$$

Although the user representation $\mathbf{u}^r$ learned from reviews contain rich information of users, there are some latent characteristics of users which are not described in their reviews but can be inferred from the rating patterns. Thus, we also represent users using the embedding of their IDs to capture the latent factors of users, which are motivated by traditional recommendation methods (Koren et al., 2009). The final representation of user u is the concatenation of the user representation $\mathbf{u}^r$ learned from reviews and the user embedding $\mathbf{u}^d$ inferred from user ID, as follows:

$$\mathbf{u} = [\mathbf{u}^r, \mathbf{u}^d]. \tag{12}$$

The representations of items can be computed in a similar way. Denote the representation of item t learned from reviews as $\mathbf{t}^r$, and the item embedding inferred from item ID as $\mathbf{t}^d$. Then the final representation of this item is as follows:

$$\mathbf{t} = [\mathbf{t}^r, \mathbf{t}^d]. \tag{13}$$

3.4 Rating Prediction

In recommender systems the recommendations are made based on the predicted ratings that a user will give to an item. In our *HUITA* approach, the rating score of a user-item pair is predicted based on the representations of users and items as follows:

$$\hat{y} = \mathrm{ReLU}(\mathbf{w}^T(\mathbf{u} \odot \mathbf{t}) + b), \tag{14}$$

where $\odot$ is item-wise dot product, $\mathbf{w}$ and b are parameters in the rating prediction layer.

In the model training stage, we optimize the model parameters to minimize the difference between gold rating and predicted ratings. We use the mean squared error as the loss function:

$$\mathcal{L} = \frac{1}{N_P} \sum_{i=1}^{N_P} (\hat{y}_i - y_i)^2, \tag{15}$$

where N_P denotes the number of user-item pairs in training data, $\hat{y}_i$ and y_i are the predicted rating score and the gold rating score respectively of the i_{th} user-item pair.

Dataset	#users	#items	#reviews
Toys_and_Games	19,412	11,924	167,597
Kindle_Store	68,223	61,935	982,619
Movies_and_TV	123,960	50,052	1,679,533
Yelp 2017	199,445	119,441	3,072,129

Table 1: Statistics of datasets used in our experiments.

4 Experiments

4.1 Datasets and Experimental Settings

We conducted experiments on four widely used benchmark datasets in different domains to evaluate the effectiveness of our approach. Following (Chen et al., 2018), we used three datasets from the Amazon collection[1](He and McAuley, 2016), i.e., **Toys_and_Games**, **Kindle_Store**, and **Movies_and_TV**. Another dataset is from Yelp Challenge 2017[2] (denoted as **Yelp 2017**), which is a large-scale restaurant review dataset. Following (Chen et al., 2018), we only kept the users and items which have at least 5 reviews. The detailed statistics of the four datasets are summarized in Table 1. The ratings in these datasets are in [1, 5].

In our experiments, the dimension of word embeddings was set to 300. We used the pre-trained Google embedding (Mikolov et al., 2013) to initialize the word embedding matrix. The word-level CNN has 200 filters and their window size is 3. The sentence-level CNN has 100 filters with window size of 3. We applied dropout strategy (Srivastava et al., 2014) to each layer of our model to mitigate overfitting. The dropout rate was set to 0.2. Adam (Kingma and Ba, 2014) was used as the optimization algorithm. The batch size was set to 20. We randomly selected 80% of the user-item pairs in each dataset for training, 10% for validation and 10% for test. All the hyperparameters were selected according to the validation set. We independently repeated each experiment for 5 times and reported the average performance in Root Mean Square Error (RMSE).

4.2 Performance Evaluation

We evaluate the performance of our approach by comparing it with several baseline methods. The methods to be compared include:

- *PMF*: Probabilistic Matrix Factorization, which models users and items based on

---

[1] http://jmcauley.ucsd.edu/data/amazon

[2] https://www.yelp.com/dataset_challenge

	PMF	NMF	SVD++	HFT	DeepCoNN	Attn+CNN	NARRE	HUITA
Rating score	✓	✓	✓	✓	✓	✓	✓	✓
Review text				✓	✓	✓	✓	✓
Word context & order					✓	✓	✓	✓
Review attention							✓	✓
Word/sentence attention						✓*		✓

Table 2: Information used in different methods. *Only word attention is modeled.

Methods	**Toys_and_Games**	**Kindle_Store**	**Movies_and_TV**	**Yelp 2017**
PMF	1.3076	0.9914	1.2920	1.3340
NMF	1.0399	0.9023	1.1125	1.2916
SVD++	0.8860	0.7928	1.0447	1.1735
HFT	0.8925	0.7917	1.0291	1.1699
DeepCoNN	0.8890	0.7876	1.0128	1.1642
Attn+CNN	0.8805	0.7796	0.9984	1.1588
NARRE	0.8769	0.7783	0.9965	1.1559
HUITA	**0.8649**	**0.7464**	**0.9631**	**1.1246**

Table 3: RMSE scores of different methods on different datasets. Lower RMSE score means better performance.

ratings via matrix factorization (Mnih and Salakhutdinov, 2008).

- *NMF*: Non-negative Matrix Factorization for recommendation based on rating scores (Lee and Seung, 2001).
- *SVD++*: The recommendation method based on rating matrix via SVD and similarities between items (Koren, 2008).
- *HFT*: Hidden Factor as Topic (HFT), a method to combine reviews with ratings via LDA (McAuley and Leskovec, 2013).
- *DeepCoNN*: Deep Cooperative Neural Networks, a neural method to jointly model users and items from their reviews via CNN (Zheng et al., 2017).
- *Attn+CNN*: Attention-based CNN, which uses both CNN and attention over word embeddings to learn user and item representation from reviews (Seo et al., 2017b).
- *NARRE*: Neural Attentional Rating Regression with Review-level Explanations, which uses attention mechanism to model the informativeness of reviews for recommendation (Chen et al., 2018).
- *HUITA*: our proposed hierarchical user and item representation approach with three-tier attention for recommendation with reviews.

In Table 2, we show a simple comparison of different methods in terms of the information considered in each method. Traditional recommendation methods such as *PMF*, *NMF* and *SVD* are solely based on rating scores, and other methods *HFT*, *DeepCoNN*, *Attn+CNN*, *NARRE* and *HUITA* can exploit both rating scores and reviews for recommendation. Among the latter methods, *HFT* is based on topic models and cannot capture the contexts and orders of words. *DeepCoNN* and *Attn+CNN* simply concatenate reviews into a long document, and cannot model the informativeness of different reviews. Although *NARRE* can model review helpfulness via attention, it simply merges all sentences in a review together, and does not model the informativeness of different sentences and words. Different from these methods, our *HUITA* approach learns user and item representations from reviews in a hierarchical manner, and uses a three-tier attention network to select and attend to important words, sentences and reviews.

The results of different methods are shown in Table 3. We have several observations from the results. First, the methods which exploit reviews (i.e., *HFT*, *DeepCoNN*, *Attn+CNN*, *NARRE* and *HUITA*) usually perform better than the methods only based on rating scores (i.e., *PMF*, *NMF* and *SVD++*). It validates reviews can provide rich information of user preferences and item properties, and is important to learn informative user and item representations and can benefit recommendation.

Methods	Toys_and_Games	Kindle_Store	Movies_and_TV	Yelp 2017
All	0.8649	0.7464	0.9631	1.1246
-word attention	0.8721	0.7610	0.9744	1.1337
-sentence attention	0.8714	0.7569	0.9715	1.1308
-review attention	0.8685	0.7523	0.9720	1.1294

Table 4: The effectiveness of different levels of attentions. The evaluation metric is RMSE.

Second, among the method which can exploit reviews, the neural network based methods (e.g., *DeepCoNN*, *Attn+CNN*, *NARRE* and *HUITA*) usually outperform the *HFT* method which is based on topic models. This is probably because in *HFT* the reviews are represented using bag-of-words features, and the contextual information and the orders of words are lost. This result validates the neural network based method can better capture the semantic information in reviews to model users and items for recommendation.

Third, the methods considering review helpfulness (i.e., *NARRE*) and word importance (i.e., *Attn+CNN*) usually outperform *DeepCoNN*. This result implies that different words and different reviews have different importance for modeling users and items from reviews. Distinguishing important reviews and words from less important ones is beneficial to learn more accurate user and item representations for recommendation.

Fourth, our approach can consistently outperform all the baseline methods compared here. This is because different from baseline methods such as *Attn+CNN* and *DeepCoNN* which merge all reviews into a long document and *NARRE* which merges all sentences into a long sentence, our *HUITA* approach learns user and item representations in a hierarchical manner. *HUITA* first learns sentence representations from words, then learns review representations from sentences, and finally learns user/item representations from reviews. Besides, our approach incorporates a three-tier attention network to jointly select and attend to important words, sentences and reviews. Thus, our approach can learn more informative user and item representations from reviews for recommendation.

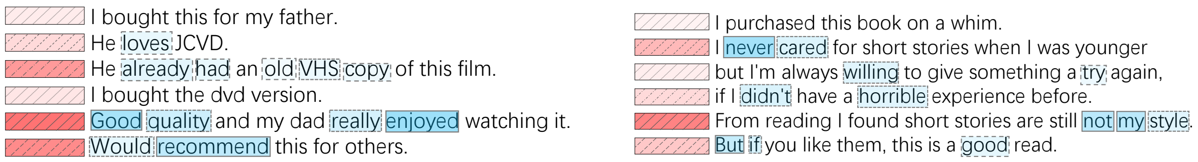
4.3 Effectiveness of Three-Tier Attention

In this section, we conducted experiments to explore the effectiveness of the three-tier attention network in our approach. We compare three variants of our model by removing one kind of attention each time to evaluate its contribution to the performance. The results are shown in Table 4.

According to Table 4, the word-level attention can effectively improve the performance of our approach. This is because different words in reviews have different importance in modeling users and items. Therefore, recognizing and highlighting the important words using the word-level attention network can help learn more informative sentence representations. In addition, the sentence-level attention is also useful. This may be because different sentences have different informativeness. For example, in a laptop review the sentence “this laptop is expensive” is more informative than “I bought this laptop yesterday” in representing this laptop. The sentence-level attention network can help to select important sentences to build review representations. Besides, the review-level attention is also useful in our *HUITA* approach. This is because different reviews have different informativeness in representing users and items. And distinguishing informative reviews from the less informative ones can help learn more accurate representations of users and items. Moreover, combining all the three levels of attentions can further improve the performance of our approach, which validates the effectiveness of our three-tier attention architecture.

4.4 Case Study

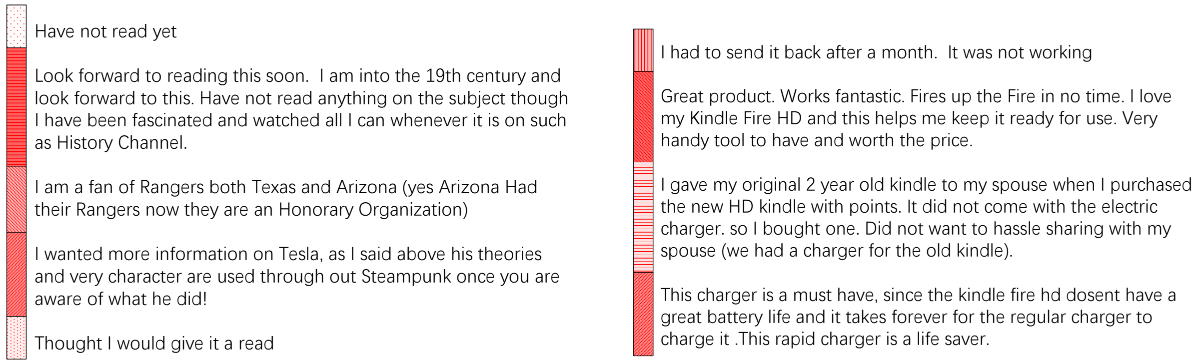
In this section, we conducted several case studies to further explore whether our approach can select informative words, sentences and reviews to learn informative user and item representations for recommendation. First, we want to explore the effectiveness of the word- and sentence-level attention networks. The visualization of the attention weights in the word- and sentence-level attention networks is shown in Fig. 3. From Fig. 3 we can see that our word-level attention network can effectively select and attend to important words. For example, in Fig. 3(a) the words “Good”, “quality” and “recommend” are assigned higher attention weights than “bought” and “dad”, since “Good”, “quality” and “recommend” can better model the properties of the film. In addition, our model can



(a) Movies_and_TV

(b) Kindle_Store

Figure 3: Visualization of attention weights in two randomly selected reviews from the Movies_and_TV and Kindle_Store datasets respectively. Red boxes to the left of the reviews represent sentence-level attention weights, and blue boxes on the individual words represent word-level attention weights. Darker color represents higher attention weights.



(a) User

(b) Item

Figure 4: Visualization of attention weights of reviews from a randomly selected user and item in the Kindle_Store dataset. Vertical bars represent review-level attention weights and darker color represents higher attention weights.

effectively select informative sentences using the sentence-level attention network. For example, in Fig. 3(b) the sentence "From reading I found short stories are still not my style" is assigned high attention weight since it is informative for representing this user and is important for recommendation, while the sentence "I purchased this book on a whim" has low attention weight since it contains limited information of users and items. Thus, these results validate that our approach is effective in selecting informative words and sentences in reviews for recommendation through the word- and sentence-level attention networks.

Second, we want to explore the effectiveness of the review-level attention in our *HUITA* approach. The visualization of the review-level attention weights is shown in Fig. 4. From Fig. 4 we can see that our approach can effectively select and attend to informative reviews. For example, the second review in Fig. 4(a) is assigned high attention weight by our approach since it reveals rich information of user preferences. However, the first review in Fig. 4(a) receives low attention weight since it contain limited information of users. Thus, these results validate the effectiveness of our approach in selecting informative reviews to learn more accurate representations of users and items from reviews for recommendation.

5 Conclusion

In this paper, we propose a hierarchical user and item representation model with three-tier attention to learn user and item representations from reviews for recommendation. In our approach, we use a sentence encoder to learn sentence representations from words, a review encoder to learn review representations from sentences, and a user/item encoder to learn user/item representations from reviews. In addition, we incorporate a three-tier attention network into our model to select and attend to informative words, sentences and reviews to learn more accurate representations of users and items. Besides, we combine the user and item representations learned from the reviews with the embeddings of user and item IDs as the final representations of users and items to capture the latent factors of individual users and items. The experiments on four benchmark datasets validate that our approach can effectively improve the performance of recommendation and consistently outperform many baseline methods.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604002, and the National Natural Science Foundation of China under Grant numbers U1836204, U1705261, U1636113, U1536201, and U1536207.

References

Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, volume 14, pages 2–8.

Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *RecSys*, pages 288–296. ACM.

Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *WWW*, pages 1583–1592.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *KDD*, pages 193–202.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*, pages 105–112.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264.

Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*, pages 485–494.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017a. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*, pages 297–305. ACM.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017b. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *MLRec*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *IJCAI*, pages 2640–2646.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *KDD*, pages 2309–2318.

Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative multi-level embedding learning from reviews for rating prediction. In *IJCAI*, pages 2986–2992.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92.

Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*, pages 425–434. ACM.

Text Similarity Estimation Based on Word Embeddings and Matrix Norms for Targeted Marketing

Tim vor der Brück
School of Information Technology
Lucerne University of
Applied Sciences and Arts
Switzerland
`tim.vorderbrueck@hslu.ch`

Marc Pouly
School of Information Technology
Lucerne University of
Applied Sciences and Arts
Switzerland
`marc.pouly@hslu.ch`

Abstract

The prevalent way to estimate the similarity of two documents based on word embeddings is to apply the cosine similarity measure to the two centroids obtained from the embedding vectors associated with the words in each document. Motivated by an industrial application from the domain of youth marketing, where this approach produced only mediocre results, we propose an alternative way of combining the word vectors using matrix norms. The evaluation shows superior results for most of the investigated matrix norms in comparison to both the classical cosine measure and several other document similarity estimates.

1 Introduction

Estimating semantic document similarity is of utmost importance in a lot of different areas, like plagiarism detection, information retrieval, or text summarization. We will focus here on an NLP application that has been less researched, i.e., the assignment of people to the best matching target group to allow for running precise and customer-oriented marketing campaigns.

Until recently, similarity estimates were predominantly based either on deep semantic approaches or on typical information retrieval techniques like Latent Semantic Analysis. In the last couple of years, however, so-called word and sentence embeddings became state-of-the-art.

The prevalent approach to document similarity estimation based on word embeddings consists in measuring the similarity between the vector representations of the two documents derived as follows:

1. The word embeddings (often weighted by the tf-idf coefficients of the associated words (Brokos et al., 2016)) are looked up in a hashtable for all the words in the two documents to compare. These embeddings are determined beforehand on a very large corpus typically using either the skip gram or the continuous bag of words variant of the Word2Vec model (Mikolov et al., 2013).
2. The centroid over all word embeddings belonging to the same document is calculated to obtain its vector representation.

If vector representations of the two documents to compare were successfully established, a similarity estimate can be obtained by applying the cosine measure to the two vectors.

Let $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ be the word vectors of two documents. The cosine similarity value between the two document centroids C_1 und C_2 is given by:

$$\begin{aligned}&\cos(\angle(\frac{1}{m}\sum_{i=1}^{m} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i))\\ =&\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\langle x_i, y_j\rangle}{mn\|C_1\|\|C_2\|}\end{aligned} \tag{1}$$

Hence, potentially small values of $\langle x_i, y_j\rangle$ can have in aggregate a considerable influence on the total similarity estimate, which makes this estimate vulnerable to noise in the data. We propose an alternative approach that is based on matrix norms and which proved to be more noise-robust by focusing primarily on high word similarities.

Finally, we conducted an evaluation where we achieved with our method superior accuracy in target group assignments than several traditional word embedding based methods.

2 Related Work

The most popular method to come up with word vectors is Word2Vec, which is based on a 3 layer neural network architecture in which the word vectors are obtained as the weights of the hidden layer. Alternatives to Word2Vec are GloVe

Proceedings of NAACL-HLT 2019, pages 1827–1836
Minneapolis, Minnesota, June 2 - June 7, 2019.

(Pennington et al., 2014), which is based on aggregated global word co-occurrence statistics and the Explicit Semantic Analysis (or shortly ESA) (Gabrilovic and Markovitch, 2009), in which each word is represented by the column vector in the tf-idf matrix over Wikipedia.

The idea of Word2Vec can be transferred to the level of sentences as well. In particular, the so-called Skip Thought Vector model (STV) (Kiros et al., 2015) derives a vector representation of the current sentence by predicting the surrounding sentences.

(Song and Roth, 2015) propose an alternative approach to applying the cosine measure to the two word vector centroids for ESA word embeddings. In particular, they establish a bipartite graph consisting of the best matching vector components by solving a linear optimization problem. The similarity estimate for the documents is then given by the global optimum of the objective function. However, this method is only useful for sparse vector representations. In case of dense vectors, (Mijangos et al., 2017) suggested to apply the Frobenius kernel to the embedding matrices, which contain the embedding vectors for all document components (usually either sentences or words) (cf. also (Hong et al., 2015)). However, crucial limitations are that the Frobenius kernel is only applicable if the number of words (sentences respectively) in the compared documents coincide and that a word from the first document is only compared with its counterpart from the second document. Thus, an optimal matching has to be established already beforehand. In contrast, the matrix norm approach as presented here applies to arbitrary embedding matrices. Since it conducts a pairwise comparison of all words contained in the two documents, there is also no need for any matching method.

Another simlarity estimate that employs the entire embedding matrix is the word mover‘s distance (Kusner et al., 2015), which is a special case of the earth mover‘s distance, a well studied transportation problem. Basically, this approach determines the minimum effort (with respect to embedding vector changes) to transform the words of one text into the words of another text. The word mover‘s distance requires a linear optimization problem to be solved. Linear optimization is usually tackled by the simplex method, which has in the worst case, which rarely occurs however, exponential runtime complexity.

Name	Definition
Frob. norm	$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n \|\mathbf{A}_{ij}\|^2}$
2-norm	$\|\mathbf{A}\|_2 := \sqrt{\rho(\mathbf{A}^\top \mathbf{A})}$
$L_{1,1}$-norm	$\|\mathbf{A}\|_{L_{1,1}} := \sum_{i=1}^m \sum_{j=1}^n \|\mathbf{A}_{ij}\|$
1-norm	$\|\mathbf{A}\|_1 := \max_{1 \leq j \leq n} \sum_{i=1}^m \|\mathbf{A}_{ij}\|$
∞-norm	$\|\mathbf{A}\|_\infty := \max_{1 \leq i \leq m} \sum_{j=1}^n \|\mathbf{A}_{ij}\|$

Table 1: Examples of matrix norms; $\mathbf{A}$ is an $m \times n$ matrix; $\rho(\mathbf{X})$ denotes the largest absolute eigenvalue of a squared matrix $\mathbf{X}$.

A drawback of conventional similarity estimates as described above is that slightly related word pairs can have in aggregate a considerable influence on their values, i.e., these estimates are sensitive to noise in the data. In contrast, several of our matrix norm based similarity estimates focus primarily on strongly related word pairs and are therefore less vulnerable to noise.

3 Similarity Measure / Matrix Norm

Before going more into detail, we want to review some concepts that are crucial for the remainder of this paper. According to (Belanche and Orozco, 2011), a similarity measure on some set $\mathbf{X}$ is an upper bounded, exhaustive and total function $s : X \times X \to I \subset \mathbb{R}$ with $|I| > 1$ (therefore I is upper bounded and $\sup I$ exists). Additionally, it should fulfill the properties of reflexivity (the supremum is reached if an item is compared to itself) and symmetry. We call such a measure normalized if the supremum equals 1 (Attig and Perner, 2011). Note that an asymmetric similarity measure can easily be converted into a symmetric by taking the geometric or arithmetic mean of the asymmetric measure applied twice to the same arguments in switched order.

A norm is a function $f : V \to \mathbb{R}$ over some vector space V that is absolutely homogeneous, positive definite and fulfills the triangle inequality. It is called matrix norm if its domain is a set of matrices and if it is sub-multiplicative, i.e., $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$. Several popular matrix norms are given in Table 1. Note that the Frobenius norm can also be represented by $\|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}\mathbf{A}^\top)}$.

4 Document Similarity Measure based on Matrix Norms

For an arbitrary document t we define the embeddings matrix $E(t)$ as follows: $E(t)_{ij}$ is the i-th component of the normalized embeddings vector belonging to the j-th word of the document t. Let t, u be two arbitrary documents, then the entry (i, j) of a product $E(t)^\top E(u)$ specifies the result of the cosine measure estimating the semantic similarity between word i of document t and word j of document u. The value of a matrix norm $\|E(t)^\top E(u)\|$ is then a measure for the similarity of the two documents. Since the vector components obtained by Word2Vec can be negative, the cosine measure between two word vectors can also assume negative values (rather rarely in practice though). Negative cosine values indicate negatively correlated words and should be handled akin to the uncorrelated case. Because a matrix norm usually treats negative and positive matrix entries alike, we replace all negative values in the matrix by zeros. Finally, since our measure should be restricted to values from zero to one, we have to normalize it.

Formally, we define our similarity measure $sn(t, u)$ as follows :

$$\frac{\|K(E(t)^\top E(u))\|}{\sqrt{\|K(E(t)^\top E(t))\| \cdot \|K(E(u)^\top E(u))\|}}$$

where $E(t)$ is the embeddings matrix belonging to document t, where all embedding column vectors are normalized. $K(\mathbf{M})$ is the matrix, where all negative entries are replaced by zero, i.e. $K(\mathbf{M})_{ij} = \max\{0, \mathbf{M}_{ij}\}$.

Proposition 1. *If the cosine similarity values between all embedding vectors of words occurring in any of the documents are non-negative, i.e., if* $K(E(t)^\top E(u)) = E(t)^\top E(u)$ *for all document pairs* (t, u), *then* sn *is a normalized similarity measure for the 2-norm, the Frobenius norm and the* $L_{1,1}$*-norm.*

Proof. We give the proof for the 2-norm here and for the other two norms in the appendix.

Symmetry At first, we focus on the symmetry condition.

Let $\mathbf{A} := E(t)$, $\mathbf{B} := E(u)$, where t and u are arbitrary documents. Symmetry directly follows, if we can show that

$$\|\mathbf{Z}\| = \|\mathbf{Z}^\top\|$$

for arbitrary matrices $\mathbf{Z}$, since with this property we have

$$\begin{aligned} sn(t,u) =& \frac{\|\mathbf{A}^\top\mathbf{B}\|}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\| \cdot \|\mathbf{B}^\top\mathbf{B}\|}} \\ =& \frac{\|(\mathbf{B}^\top\mathbf{A})^\top\|}{\sqrt{\|\mathbf{B}^\top\mathbf{B}\| \cdot \|\mathbf{A}^\top\mathbf{A}\|}} \\ =& \frac{\|\mathbf{B}^\top\mathbf{A}\|}{\sqrt{\|\mathbf{B}^\top\mathbf{B}\| \cdot \|\mathbf{A}^\top\mathbf{A}\|}} = sn(u,t) \end{aligned} \tag{2}$$

Let $\mathbf{M}$ and $\mathbf{N}$ be arbitrary matrices such that $\mathbf{MN}$ and $\mathbf{NM}$ are both defined and quadratic, then (see (Chatelin, 1993))

$$\rho(\mathbf{MN}) = \rho(\mathbf{NM}) \tag{3}$$

where $\rho(\mathbf{X})$ denotes the largest absolute eigenvalue of a squared matrix $\mathbf{X}$.

Using identity 3 one can easily infer that:

$$\|\mathbf{Z}\|_2 = \sqrt{\rho(\mathbf{Z}^\top\mathbf{Z})} = \sqrt{\rho(\mathbf{Z}\mathbf{Z}^\top)} = \|\mathbf{Z}^\top\|_2 \tag{4}$$

□

Boundedness

Proof. The following property needs to be verified:

$$\frac{\|\mathbf{A}^\top\mathbf{B}\|_2}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_2 \cdot \|\mathbf{B}^\top\mathbf{B}\|_2}} \leq 1 \tag{5}$$

In the proof, we exploit the fact that for every positive-semidefinite matrix $\mathbf{X}$, the following equation holds

$$\rho(\mathbf{X}^2) = \rho(\mathbf{X})^2 \tag{6}$$

We observe that for the denominator

$$\begin{aligned} & \|\mathbf{A}^\top\mathbf{A}\|_2 \cdot \|\mathbf{B}^\top\mathbf{B}\|_2 \\ =& \sqrt{\rho((\mathbf{A}^\top\mathbf{A})^\top\mathbf{A}^\top\mathbf{A})}\sqrt{\rho((\mathbf{B}^\top\mathbf{B})^\top\mathbf{B}^\top\mathbf{B})} \\ =& \sqrt{\rho((\mathbf{A}^\top\mathbf{A})^\top(\mathbf{A}^\top\mathbf{A})^\top)}\sqrt{\rho((\mathbf{B}^\top\mathbf{B})^\top(\mathbf{B}^\top\mathbf{B})^\top)} \\ =& \sqrt{\rho([(\mathbf{A}^\top\mathbf{A})^\top]^2)}\sqrt{\rho([(\mathbf{B}^\top\mathbf{B})^\top]^2)} \\ \overset{(6)}{=}& \sqrt{\rho((\mathbf{A}^\top\mathbf{A})^\top)^2}\sqrt{\rho((\mathbf{B}^\top\mathbf{B})^\top)^2} \\ =& \rho((\mathbf{A}^\top\mathbf{A})^\top)\rho((\mathbf{B}^\top\mathbf{B})^\top) \\ \overset{(4)}{=}& \|\mathbf{A}\|_2^2 \cdot \|\mathbf{B}\|_2^2 \end{aligned} \tag{7}$$

Putting things together we finally obtain

$$\begin{aligned} \frac{\|\mathbf{A}^\top\mathbf{B}\|_2}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_2\|\mathbf{B}^\top\mathbf{B}\|_2}} & \overset{\text{sub-mult.}}{\leq} \frac{\|\mathbf{A}^\top\|_2 \cdot \|\mathbf{B}\|_2}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_2\|\mathbf{B}^\top\mathbf{B}\|_2}} \\ & \overset{(4)}{=} \frac{\|\mathbf{A}\|_2 \cdot \|\mathbf{B}\|_2}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_2\|\mathbf{B}^\top\mathbf{B}\|_2}} \\ & \overset{(7)}{=} \frac{\|\mathbf{A}\|_2 \cdot \|\mathbf{B}\|_2}{\sqrt{\|\mathbf{A}\|_2^2 \cdot \|\mathbf{B}\|_2^2}} = 1 \end{aligned} \tag{8}$$

□

However, proposition 1 is not sufficient in all cases, since negative cosine similarity values can occur in practice. Therefore, we also prove a stronger claim stated in the following proposition.

Proposition 2. *If the cosine measure values between embedding vectors belonging to words of the same document are all non-negative, then* sn *is a normalized similarity measure for the Frobenius and the* $L_{1,1}$*-norm.*

Proof. The proof of symmetry and reflexivity is analogous to proposition 1. So we only prove boundedness of sn. Since the cosine measure for two embedding vectors emb belonging to words of the same document cannot be negative, we have $\langle emb(w_i), emb(w_k)\rangle \geq 0$ for i,k with $1 \leq i \leq k \leq |t|$ and therefore $K(E(t)^\top E(t)) = E(t)^\top E(t)$. We furthermore have $\|K(E(t)^\top E(u))\| \leq \|E(t)^\top E(u)\|$ for the Frobenius and $L_{1,1}$-norm, since replacing a zero entry with another value can never decrease the value of the norm. Thus,

$$\begin{aligned} &\frac{\|K(E(t)^\top E(u))\|}{\sqrt{\|K(E(t)^\top E(t))\| \cdot \|K(E(u)^\top E(u))\|}} \leq \\ &\frac{\|E(t)^\top E(u)\|}{\sqrt{\|E(t)^\top E(t)\| \cdot \|E(u)^\top E(u)\|}} \leq 1. \end{aligned} \tag{9}$$

The last inequality follows from proposition 1. ☐

However, the proposed normalization factor $\sqrt{\|K(E(t)^\top E(t))\| \cdot \|K(E(u)^\top E(u))\|}$ is not eligible for all types of matrix norms, which is an immediate consequence of the following proposition.

Proposition 3. *There exist no mean value function* $mean : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ *such that*

$$\begin{aligned} sn_1 :=& \frac{\|K(E(t_0)^\top E(u_0))\|_1}{m} \\ m :=& mean(\|K(E(t_0)^\top E(t_0))\|_1, \\ & \|K(E(u_0)^\top E(u_0))\|_1) \end{aligned} \tag{10}$$

is bounded by one. Hence, sn_1 *cannot be a normalized similarity measure.*

Proof. We give a counter-example for the maximum mean, for which we show that sn_1 exceeds the value of 1:

$$E(t_0) = \begin{bmatrix} 0.1644 & 0.5025 \\ 0 & 0.5025 \\ 0.9864 & 0.7035 \end{bmatrix}$$

Corpus	# Words
German Wikipedia	651 880 623
Frankfurter Rundschau	34 325 073
News journal *20 Minutes*	8 629 955

Table 2: Corpus sizes measured by number of words.

$$E(u_0) = \begin{bmatrix} 0.1204 & 0.9759 & 0.2722 \\ 0.2408 & 0.0976 & 0.9526 \\ 0.9631 & 0.1952 & 0.1361 \end{bmatrix}$$

Since the maximum mean $mean_{max}(a, b) = \max\{a, b\}$ is greater or equal to all other means (including the geometric mean), we have that:

$$\begin{aligned} &\frac{\|K(E(t_0)^\top E(u_0))\|_1}{mean(\|K(E(t_0)^\top E(t_0))\|_1, \|K(E(u_0)^\top E(u_0))\|_1)} \\ &\geq \frac{\|K(E(t_0)^\top E(u_0))\|_1}{\max\{\|K(E(t_0)^\top E(t_0))\|_1, \|K(E(u_0)^\top E(u_0))\|_1\}} \\ &= 1.0284 > 1 \end{aligned} \tag{11}$$

for arbitrary type of means $mean$. ☐

Note that the matrices used in the counter-example can be extended to any number of embedding dimensions by adding additional zeros.

A further issue is, whether the similarity measure is invariant to word permutations. Actually, this is the case for our matrix norm similarity estimates, which is stated in the following proposition.

Proposition 4. *The obtained similarity estimate for all of the considered matrix norms is independent of the word sequence of the input texts.*

This property is quite beneficial in our scenario since one of the texts to compare constitutes of an unordered key word list (see more details in the next section).

Proof. We focus in this proof on the 2-norm, for which this property is not directly obvious like for the other regarded norms. For simplicity, we first concentrate on the special case that all cosine values between word embeddings are non-negative. This proof can easily be extended to the general case, too. In particular, we show that the similarity estimate does not change, if two columns of the first matrix are exchanged, which can be expressed by postmultiplying this matrix with a permutation matrix $\mathbf{P}$. By employing symmetry and induction this proof can be applied to arbitrary sequence permutations and to the second argument matrix as well. With this, the similarity estimate is given as:

$$\begin{aligned} sn_2(t,u) =& \|((\mathbf{AP})^\top \mathbf{B})\|_2 \\ =& \sqrt{\rho(((\mathbf{AP})^\top \mathbf{B})^\top ((\mathbf{AP})^\top \mathbf{B}))} \\ =& \sqrt{\rho(\mathbf{B}^\top \mathbf{APP}^\top \mathbf{A}^\top \mathbf{B})} \\ & (\mathbf{P} \text{ is an orthogonal matrix}) \\ =& \sqrt{\rho(\mathbf{B}^\top \mathbf{AIA}^\top \mathbf{B})} \\ =& \sqrt{\rho(\mathbf{A}^\top \mathbf{B})^\top (\mathbf{A}^\top \mathbf{B})} \\ =& \|\mathbf{A}^\top \mathbf{B}\|_2 \end{aligned} \tag{12}$$

By exploiting that $K(\mathbf{MP}) = K(\mathbf{M})\mathbf{P}$ for arbitary matrices $\mathbf{M}$, this proof can be generalized to negative cosine measure values as well. □

The question remains, how the similarity measure value induced by matrix norms performs in comparison to the usual centroid method. Let us first focus on L_{11} and the Frobenius norm. Actually, both are special cases of a norm that raises the absolute values of the matrix components to a certain power e. If this exponent e becomes large, then:

$$\begin{aligned} & sn_{L_{e,1}}(t,u) \\ =& \frac{\|E(u)^\top E(t)\|_{L_{e,1}}}{\sqrt{\|E(t)^\top E(t)\|_{L_{e,1}} \cdot \|E(u)^\top E(u)\|_{L_{e,1}}}} \\ \approx& \frac{(\#p)^{1/e}}{\sqrt{\left\|\begin{bmatrix} 1 & 0 & \dots \\ & \ddots & \\ 0 & \dots & 1 \end{bmatrix}\right\|_{L_{e,1}} \cdot \left\|\begin{bmatrix} 1 & 0 & \dots \\ & \ddots & \\ 0 & \dots & 1 \end{bmatrix}\right\|_{L_{e,1}}}} \\ & (\text{mainly the diagonal elements of the} \\ & \text{matrices in the denominator assume } 1) \\ \approx& \left(\frac{\#p}{\sqrt{nm}}\right)^{1/e} \end{aligned} \tag{13}$$

where $\#p$ denotes the number of perfect matches (similarity value of 1.0) between words of the two documents and n (m) is the number of words in text t (u). Thus, with an increasing exponent, $sn_{L_{e,1}}$ tends to focus on very good matches and disregards the others. This property is quite beneficial in our scenario, where often only one or two words of the contest answers (cf. next section) indicate the right target group.

General statements about the 2-norm based similarity measure are difficult, but we can draw some conclusions, if we restrict to the case, where $\mathbf{A}^\top \mathbf{B}$ is a square diagonal matrix. Hereby, one word of the first text is very similar to exactly one word of the second text and very dissimilar to all remaining words. The similarity estimate is then given by the largest eigenvalue (also called the spectral radius) of $\mathbf{A}^\top \mathbf{B}$, which equals the largest cosine measure value. Thus, the 2-norm based similarity estimate is able to filter out noise (low word similarity values) akin to the Frobenius norm.

Let us now take a look at the similarity measure sn_1, which is induced by the 1-norm. sn_1 assumes high values, if there is one word of the second document that matches very well with all words of the first document. All other less matching words of the second document do not contribute to the assumed similarity estimate at all.

5 Application to Targeted Marketing

Market segmentation is one of the key tasks of a marketer. Usually, it is accomplished by clustering over demographic variables, geographic variables, psychographic variables and behaviors (Lynn, 2011). In this paper, we will describe an alternative approach based on unsupervised natural language processing. In particular, our business partner operates a commercial youth platform for the Swiss market, where registered members get access to third-party offers such as discounts and special events like concerts or castings. Actually, several hundred online contests per year are launched over this platform sponsored by other firms, an increasing number of them require the members to write short free-text snippets, e.g. to elaborate on a perfect holiday at a destination of their choice in case of a contest sponsored by a travel agency. Based on the results of a broad survey, the platform provider's marketers assume five different target groups (called *milieus*) being present among the platform members: *progressive postmodern youth* (people primarily interested in culture and arts), *young performers* (people striving for a high salary with a strong affinity to luxury goods), *freestyle action sportsmen*, *hedonists* (rather poorly educated people who enjoy partying and disco music) and *conservative youth* (traditional people with a strong concern for security). A sixth milieu called *special groups* comprises all those who cannot be assigned to one of the upper five milieus. For each milieu (with the exception of *special groups*) a keyword list was manually created to describe its main characteristics.

Method	Contest			
	1	2	3	All
Random	0.167	0.167	0.167	0.167
ESA	0.357	0.254	**0.288**	0.335
ESA2	0.355	0.284	0.227	0.330
W2VC	0.347	**0.328**	0.227	0.330
WW2VC	0.347	0.299	0.197	0.322
GloVe	0.350	0.269	0.258	0.328
STV	0.157	0.313	0.258	0.189
$sn_{L_{1,1}}$	0.337	**0.328**	0.197	0.318
GM	**0.377**	0.313	0.227	**0.350**
sn_1	0.372	0.299	0.212	0.343
sn_∞	0.243	0.254	0.273	0.248
sn_2	0.370	0.299	**0.288**	**0.350**
sn_F	0.367	0.254	0.242	0.337

Table 3: Obtained accuracy values for similarity measures induced by different matrix norms and for four baseline methods. GM = Geometric Mean of sn_1 and sn_∞. (W)W2VC=(tf-idf-weighted) Word2Vec Embedding Centroids.

Method	Contest		
	1	2	3
Min kap.	0.123	0.295/0.030	0.110/0.101
Max. kap.	0.178	0.345/0.149	0.114/0.209
sn_2	0.128	0.049/0.065	0.060/0.064
sn_1	0.124	-0.032/0.033	0.024/0.017
sn_F	0.129	0.041/0.042	0.039/0.045
# Entr.	1544	100	100

Table 4: Minimum and maximum average inter-annotator agreements (Cohen's kappa) / average inter-annotator agreement values for our automated matching method, FN=Frobenius norm.

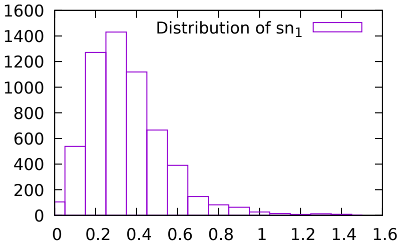


Figure 1: Distribution of sn_1 determined on contest 1.

For triggering marketing campaigns, an algorithm shall be developed that automatically assigns each contest answer to the most likely target group: we propose the youth milieu as best match for a contest answer, for which the estimated semantic similarity between the associated keyword list and user answer is maximal. In case the highest similarity estimate falls below the 10 percent quantile for the distribution of highest estimates, the special groups milieu is selected. Since the keyword list typically consists of nouns (in the German language capitalized) and the user contest answers might contain a lot of adjectives and verbs as well, which do not match very well to nouns in the Word2Vec vector representation, we actually conduct two comparisons for our Word2Vec based measures, one with the unchanged user contest answers and one by capitalizing every word beforehand. The final similarity estimate is then given as the maximum value of both individual estimates.

6 Evaluation

For evaluation, we selected three online contests (language: German), where people elaborated on their favorite travel destination for an example, speculated about potential experiences with a pair of fancy sneakers (contest 2) and explained why they emotionally prefer a certain product out of four available candidates. We experimented with different keyword list sizes but obtained the best results with rather few and therefore precise keywords. In particular, we used the following number of keywords for the individual milieus:

- Action Sportsman: 3
- Young Performer: 4
- Hedonist: 7
- Conservative Youth: 4
- Progressive Postmodern Youth: 6

In order to provide a gold standard, three professional marketers from different youth marketing companies annotated independently the best matching youth milieus for every contest answer. We determined for each annotator individually his/her average inter-annotator agreement with the others (Cohen's kappa). The minimum and maximum of these average agreement values are given in Table 4. Since for contest 2 and contest 3, some of the annotators annotated only the first 50 entries (last 50 entries respectively), we specified min/max average kappa values for both parts. We further compared the youth milieus proposed by our unsupervised matching algorithm with the majority votes over the human experts' answers (see Table 3)[1]. Moreover, we computed its average inter-annotator agreement with the human annotators (see again Table 4), quasi treating the predictions like additional annotations.

The Word2Vec word embeddings were trained on the German Wikipedia (dump originating from 20 February 2017) merged with a Frankfurter Rundschau newspaper Corpus and 34 249 articles of the news journal *20 minutes*[2], where the latter is targeted to the Swiss market and freely available at various Swiss train stations (see Table 2 for a comparison of corpus sizes). By employing articles from *20 minutes*, we want to ensure the reliability of word vectors for certain Switzerland specific expressions like *Velo* or *Glace*, which are underrepresented in the German Wikipedia and the Frankfurter Rundschau corpus. ESA is usually trained on Wikipedia, since the authors of the original ESA paper suggest that the articles of the training corpus should represent disjoint concepts, which is only guaranteed for encyclopedias. However, Stein and Anerka (Gottron et al., 2011) challenged this hypothesis and demonstrated that promising results can be obtained by applying ESA on other types of corpora like the popular Reuters newspaper corpus as well. Unfortunately, the implementation we use (Wikiprep-ESA[3]) expects its training data to be a Wikipedia Dump. Furthermore, Wikiprep-ESA only indexes words that are connected by hyperlinks, which are usually lacking in ordinary newspaper articles. So we could train Wikiprep-ESA on Wikipedia only but additionally have developed a version of ESA that can be applied on arbitrary corpora (in the following referred to as ESA2) and which was trained on the full corpus (Wikipedia+Frankfurter Rundschau+20 minutes). The STVs were also trained on the same corpus as our matrix norms based estimates and Word2Vec embedding centroids. The actual document similarity estimation is accomplished by the usual centroid approach (we did not evaluate matrix norms here). An issue we were faced with is that STVs are not bag of word models but actually take the sequence of the words into account and therefore the obtained similarity estimate between milieu keyword list and contest answer would be dependent on the keyword ordering. However, this order could have arbitrarily been chosen by the marketers and might be completely random. A possible solution is to compare the contest answers with all possible permutation of keywords and determine the maximum value over all those comparisons. However, such an approach would be infeasible already for medium keyword list sizes. Therefore, we use a beam search approach instead, which extends the keyword list iteratively and keeps only the n-best performing permutations.

Method	Accuracy
ESA	0.672
STV	0.716
W2VC	0.726
sn_2	0.731
$sn_{L_{1,1}}$	0.741
sn_F	0.781

Table 5: Accuracy value otbained for matching a sentence of the first to the associated sentence of the second translation.

Finally, to verify the general applicability of our approach, we conducted a second experiment, where a novel from Edgar Allen Poe (The purloined letter) was independently translated by two translators into German. We aim to match a sentence from the first translation to the associated sentence of the second by looking for the assignment with the highest semantic relatedness disregarding the sentence order. The obtained accuracy values based on the first 200 sentences of both translations are given in Table 5. To guarantee an 1:1 sentence mapping, periods were partly replaced by semicolons.

7 Discussion

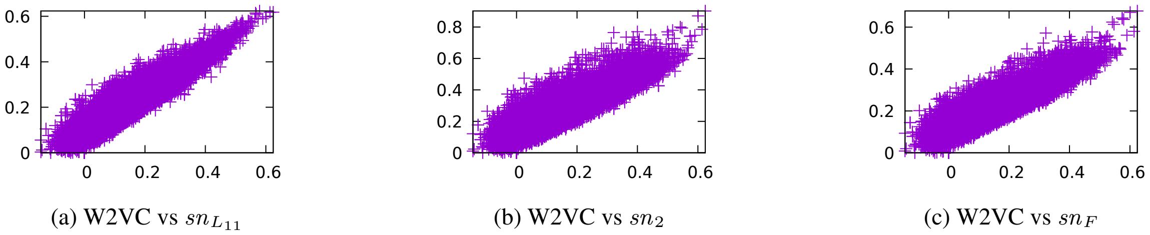
The evaluation showed that the inter-annotator agreement values vary strongly for contest 2 part 2 (minimum average annotator agreement according to Cohen's kappa of 0.03 while the maximum is 0.149, see Table 4). On this contest part, our matrix norm-based matching (2-norm and Frobenius-norm) obtains a considerably higher average agreement than one of the annotators. Regarding baseline systems, the most relevant comparison is naturally the one with Word2Vec cen-

---

[1] Note that the geometric mean of the 1- and ∞-norm as specified in Table 3 is not a matrix norm itself, since it lacks submultipicativity.

[2] http://www.20min.ch

[3] https://github.com/faraday/wikiprep-esa



(a) W2VC vs $sn_{L_{11}}$ (b) W2VC vs sn_2 (c) W2VC vs sn_F

Figure 2: Scatter plots of cosine between centroids of Word2Vec embeddings (W2VC) vs sn.

troids, since it employs the same type of data. Hereby we reached higher accuracy values for the best performing matrix norms on two of the three contests including the largest contest 1. Note that the elimination of negative values from the embedding matrix product proved to be important. If we omit this step, the obtained accuracy of sn_f for instance will drop by around 0.023 determined over all three contests (column: *all*).

It is quite striking that, although sn_1 lacks two properties of a normalized similarity measure (boundedness by 1 and symmetry), it reaches quite good results on contest 1. As you can see in Figure 1, which shows the distribution of sn_1 in contest 1, the value of 1 is indeed exceeded several times (the maximum value is 1.5), but this occurs rather rarely in our experiment. Actually, 99% of its values fall into the interval [0,1]. Thus, the non-boundedness is much less a problem in practice than the theoretical results indicate.

Finally, we determined the scatter plots (see Figure 2) showing cosine of Word2Vec embeddings (W2VC) vs several matrix norm based similarity estimates. These scatter plots exhibits that the score distributions of sn_f and sn_2 are quite similar and their values often exceed the cosine measure value due to the fact that a few very strong word matches can already result in a high similarity estimate. The scatter plot for sn_{L11} reveals that this measure is much closer to W2VC than the other two matrix norm based similarity estimates.

Note that a downside of our approach in relation to the usual Word2Vec centroids method is the increased runtime, since it requires the pairwise comparison of all words contained in the input documents. In our scenario with rather short text snippets and keyword lists, this was not much of an issue. However, for large documents, such a comprehensive comparison could become soon infeasible. One possible solution for this performance issue is to apply our proposed estimates to sentence embeddings instead of word embeddings, which on the one hand would reduce the dimensionality of the embedding matrices and on the other hand would take word order into account.

8 Conclusion

We proposed a novel similarity measure to compare word embeddings from different documents, which makes use of matrix norms. This measure was evaluated on the task to assign users to the best matching marketing target groups. We obtained superior results compared to the usual centroid / cosine measure similarity estimation for most of the investigated matrix norm especially for the largest contest 1. Furthermore, we proved elementary properties for our proposed similarity measure regarding its well-definedness and its performance in comparison to the usual centroid-based approach.

Acknowledgement

Hereby we thank the Jaywalker GmbH as well as the Jaywalker Digital AG for their support regarding this publication and especially for annotating the contest data with the best-fitting youth milieus.

References

Anja Attig and Petra Perner. 2011. The problem of normalization and a normalized similarity measure by online data. *Transactions on Case-Based Reasoning*, 4(1).

Lluís A. Belanche and Jorge Orozco. 2011. Things to know about a (dis)similarity measure. In *Proceedings of the 15th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, Karlsruhe, Germany.

Georgios-Ioannis Brokos, Prodromos, and Ion Androutsopoulos. 2016. Using centroids of word embeddings and word mover‘s distance for biomedical document retrieval in question answering. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 114–118, Berlin, Germany.

Françoise Chatelin. 1993. *Eigenvalues of Matrices - Revised Edition*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania.

Evgeniy Gabrilovic and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34.

Thomas Gottron, Maik Anderka, and Benno Stein. 2011. Insights into explicit semantic analysis. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1961–1964, Glasgow, UK.

Ki-Joo Hong, Gai-Hui Lee, and Han-Joon Kom. 2015. Enhanced document clustering using wikipedia-based document representation. In *Proceedings of the 2015 International Conference on Applied System Innovation (ICASI)*.

Ryand Kiros, Yukun Zhu, Rusland Salakhudinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fiedler. 2015. Skip-thought vectors. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilias Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 957–966.

Michael Lynn. 2011. Segmenting and targeting your market: Strategies and limitations. Technical report, Cornell University. Online: `http://scholorship.sha.cornell.edu/articles/243`.

Victor Mijangos, Gerardo Sierra, and Azuncena Montes. 2017. Sentence level matrix representation for document spectral clustering. *Pattern Recognition Letters*, 85.

Tomas Mikolov, Ilya Sutskever, Chen Ilya, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe, Nevada.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Katar.

Yangqiu Song and Dan Roth. 2015. Unsupervised sparse vector densification for short text similarity. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, Colorado.

A Proof of Proposition 1

In this section we give the proof of proposition 1 for Frobenius and $L_{1,1}$-norm.

Reflexivity is trivially fulfilled for Frobenius, 2- and $L_{1,1}$-norm. Therefore, we only prove that the induced similarity measures are symmetric and upper-bounded by 1.

Let $\mathbf{A} := E(t)$, $\mathbf{B} := E(u)$, where t and u are arbitrary documents.

A.1 Frobenius norm

Symmetry The trace of a matrix is known to be invariant under cyclic permutations. With this property, the symmetry of sn_F can easily be deduced.

Boundedness

Proof. We need to show that:

$$\frac{\|\mathbf{A}^\top\mathbf{B}\|_F}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_F \cdot \|\mathbf{B}^\top\mathbf{B}\|_F}} \leq 1 \tag{14}$$

For that we leverage the Cauchy-Schwarz inequality that states that for an inner product space $\mathcal{H}$ we have

$$|\langle x, y\rangle| \leq \sqrt{\langle x, x\rangle \cdot \langle y, y\rangle} \tag{15}$$

for all $x, y \in \mathcal{H}$, where equality holds if, and only if, x is a scalar multiple of y. In particular, the function $f(\mathbf{Y}, \mathbf{X}) = \mathrm{tr}(\mathbf{Y}^\top\mathbf{X})$ is such an inner product.

Let $\mathbf{X} = \mathbf{A}\mathbf{A}^\top$ and $\mathbf{Y} = \mathbf{B}\mathbf{B}^\top$ such that $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m\times m}$. Then we can infer

$$\begin{aligned}\mathrm{tr}(\mathbf{B}\mathbf{B}^\top\mathbf{A}\mathbf{A}^\top) &= \mathrm{tr}((\mathbf{B}\mathbf{B}^\top)^\top\mathbf{A}\mathbf{A}^\top)\\ &= \mathrm{tr}(\mathbf{Y}^\top\mathbf{X})\\ &\overset{(15)}{\leq} \sqrt{\mathrm{tr}(\mathbf{X}^\top\mathbf{X}) \cdot \mathrm{tr}(\mathbf{Y}^\top\mathbf{X})}\\ &= \sqrt{\mathrm{tr}((\mathbf{A}\mathbf{A}^\top)^\top\mathbf{A}\mathbf{A}^\top) \cdot \mathrm{tr}((\mathbf{B}\mathbf{B}^\top)^\top\mathbf{B}\mathbf{B}^\top)}\\ &= \sqrt{\mathrm{tr}(\mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{A}^\top) \cdot \mathrm{tr}(\mathbf{B}\mathbf{B}^\top\mathbf{B}\mathbf{B}^\top)}\end{aligned} \tag{16}$$

Next, we observe that for the numerator

$$\begin{aligned}\|\mathbf{A}^\top\mathbf{B}\|_F &= \sqrt{\mathrm{tr}(\mathbf{A}^\top\mathbf{B}(\mathbf{A}^\top\mathbf{B})^\top)}\\ &= \sqrt{\mathrm{tr}(\mathbf{A}^\top\mathbf{B}\mathbf{B}^\top\mathbf{A})}\\ &= \sqrt{\mathrm{tr}(\mathbf{B}\mathbf{B}^\top\mathbf{A}\mathbf{A}^\top)}\\ &\overset{(16)}{\leq} \sqrt[4]{\mathrm{tr}(\mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{A}^\top) \cdot \mathrm{tr}(\mathbf{B}\mathbf{B}^\top\mathbf{B}\mathbf{B}^\top)}\\ &= \sqrt[4]{\mathrm{tr}(\mathbf{A}^\top\mathbf{A}^\top\mathbf{A}\top\mathbf{A}) \cdot \mathrm{tr}(\mathbf{B}\top\mathbf{B}\mathbf{B}^\top\mathbf{B})}\\ &= \sqrt{\|\mathbf{A}^\top\mathbf{A}\|_F \cdot \|\mathbf{B}^\top\mathbf{B}\|_F}\end{aligned} \tag{17}$$

Hence, we finally obtain

$$\begin{aligned}\frac{\|\mathbf{A}^\top\mathbf{B}\|_F}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_F\cdot\|\mathbf{B}^\top\mathbf{B}\|_F}} &\overset{(17)}{\leq} \frac{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_F\cdot\|\mathbf{B}^\top\mathbf{B}\|_F}}{\sqrt{\|\mathbf{A}^\top\mathbf{A}\|_F\cdot\|\mathbf{B}^\top\mathbf{B}\|_F}}\\ &= 1.\end{aligned}$$

□

A.2 $L_{1,1}$-norm

Proof for symmetry. For the $L_{1,1}$-norm, we have that

$$\begin{aligned}&\|A\|_{L_{1,1}} = \sum_{i=1}^{m}\sum_{j=1}^{n}|A_{ij}| = \sum_{j=1}^{n}\sum_{i=1}^{m}|A^\top_{ji}|\\ &=\|A^\top\|_{L_{1,1}}\end{aligned} \tag{18}$$

□

The boundedness follows directly from the Cauchy-Schwarz inequality, since the induced similarity estimate $sn_{L_{1,1}}$ is an inner product.

Glocal: Incorporating Global Information in Local Convolution for Keyphrase Extraction

Animesh Prasad†‡ **Min-Yen Kan**†‡
† School of Computing
National University of Singapore
‡ Institute for Application of Learning Science and Educational Technology
National University of Singapore
`animesh@u.nus.edu kanmy@comp.nus.edu.sg`

Abstract

Graph Convolutional Networks (GCNs) are a class of spectral clustering techniques that leverage localized convolution filters to perform supervised classification directly on graphical structures. While such methods model nodes' local pairwise importance, they lack the capability to model global importance relative to other nodes of the graph. This causes such models to miss critical information in tasks where global ranking is a key component for the task, such as in keyphrase extraction. We address this shortcoming by allowing the proper incorporation of global information into the GCN family of models through the use of scaled node weights. In the context of keyphrase extraction, incorporating global random walk scores obtained from TextRank boosts performance significantly. With our proposed method, we achieve state-of-the-art results, bettering a strong baseline by an absolute 2% increase in F_1 score.

1 Introduction

Learning directly on a graphical structure is a crucial requirement in many domains. These graphs represent information in many forms, ranging from interconnected user groups to contextually linked documents to a central document by shared vocabulary. Learning on graphs has been studied extensively in the form of spectral clustering (Ng et al., 2002). The potential of learning directly on graphs has realized in semi-supervised settings where labels for only a few of the nodes are available. Some prior work formulates such setup as propagating the label information using some form of graph-based regularization (Kipf and Welling, 2016). Recently proposed works have updated such methods to be end-to-end learnable in the deep learning style by employing gradient descent on nodes within a fixed neighborhood, approximating spectral clustering's means of approximating the graph's eigenvectors (Bronstein et al., 2017) by aggregating neighborhood features. Recent advancements in normalizing the gradient range further improve the efficiency of such solutions (Kipf and Welling, 2016). However, these techniques can only exploit local features within the neighborhood of individual nodes. For some tasks, such simplified local feature aggregation may be sufficient, but insufficient for tasks that need global relative importance information.

One such important graph-based task is keyphrase extraction. In this task, individual words or phrases serve as graph nodes, and edges represent some form of co-occurrence. Keyphrase extraction has been extensively studied, in both supervised (classification) and unsupervised (ranking) modes. Depending on the length of the text and the final application of the task, solutions can be sample-based classification, pairwise ranking or sequential labeling. For example, Kim et al.(2010) explore the case of extracting top keyphrases from complete documents for downstream indexing, while Augenstein et al.(2017) connects its usage for knowledge base generation, aiming to extract all plausible keyphrases within a short excerpt. Treating a full-text scenario is arguably more challenging than the treatment of an excerpt scenario, as it requires the understanding of the much larger scale of text and extracting its most salient aspects.

Traditional supervised models employ a host of hand-engineered features — $tf.idf$, candidate length, POS tags, sectional information, frequency, among others (Kim et al., 2013; Hasan and Ng, 2010) — trained with a wide range of classifiers. As they typically model the task as a binary classification task (*i.e.*, $keyphrase, \neg keyphrase$), they suffer severely from class imbalance as keyphrases are the excep-

Proceedings of NAACL-HLT 2019, pages 1837–1846
Minneapolis, Minnesota, June 2 - June 7, 2019.

tion among most plausible candidates.

Unsupervised methods use co-occurrence as a signal for the labels. Under the hypothesis that keyphrase saliency is strongly correlated with repetition, graphical methods for unsupervised keyphrase extraction employ centrality measures and random walk techniques to rank prospective keyphrases (Mihalcea and Tarau, 2004). This hypothesis is widely exploited, with proposed extensions further enriching the graph by incorporating topic, section and/or position information (Florescu and Caragea, 2017b,a; Jiang et al., 2018), among other forms of side information.

With these in mind, we make two important observations about the existing keyphrase extraction techniques:

- In the supervised setting, word importance is captured in metrics and engineered features, as is local random walk scores. However, the structure of the graph formed by the text is not exploited.
- In the unsupervised setting, most techniques do not tightly incorporate the rich semantic features common in the supervised setting. Furthermore, random walk scores are used as-is, without the capability of being fine-tuned by downstream supervision.

From this dichotomy, we see there is a gap to close in merging the advantages of both. We propose a Glocal (global–local portmanteau) technique which incorporates both components directly over the word–graph. Specifically, we contribute a neural model that elegantly incorporates the random walk scores, while incorporating parameters to fit keyphrase labels. To the best of our knowledge, our model is the only supervised full-text keyphrase extraction model that operates directly on the word–graph.

2 Related Work

Our work draws motivation from the introduction of random walks to NLP. In this regard, TextRank (Mihalcea and Tarau, 2004), is a central representative work that serves as a basis for many text extraction modeling techniques used in entity extraction and extractive summarization (we use random walk and TextRank interchangeably in this paper). The success of the application of such random walks in text extraction is based on the hypothesis that the important nodes aggregate more mass and are thereby representative of the graph as a whole. Importantly, TextRank can be viewed as a **ranking** model, as it induces a ranking of graph nodes via its centrality calculation. However, as noted, supervised techniques for properly incorporating this information natively within the model, in our opinion, has yet to be explored.

Recently, neural models have been developed to work on graphs. These models port the ideas of spectral clustering into the deep learning modality, allowing direct computation on graphs. Methods such as Graph Convolution Network (or GCN) can then be applied to many different task scenarios which natively feature graphical structures such as citation and community graphs, which are common in the database, information retrieval, and digital library domains (Kipf and Welling, 2016; Hamilton et al., 2017). They enrich the graph by aggregating features with information gathered from the neighborhood of the node to be classified. Enhancements to the model introduce much sophisticated local information aggregation between the node pairs as in Graph Attention Networks (GAT) Veličković et al.. However, we note that such prior methods fall inherently into the **classification** paradigm, and hence focus on only local aggregation; i.e., to pull in the most significant feature from its neighbors.

In the context of keyphrase extraction, Zhang et al. (2017) is a recent work that learns directly on the graph. Their method, MIKE, determines the weight of edges and nodes in a supervised manner, rather than just utilizing co-occurrence statistics. Their work features 5 orthogonal features, one of which is topic distribution. They consider the prominence of the tokens per topic as the surrogate for ranking, utilized for model training, by minimizing the difference in predicted and gold-standard rank between iterations. MIKE can be employed only when topic information is available, but unfortunately, does not generalize to the more common case where only gold-standard keyphrases are available for training.

In Augenstein et al.(2017) benchmarking, state-of-the-art rich semantic embeddings deep learning and handcrafted feature–based statistical sequential labeling models used LSTM (Ammar et al., 2017) and CRF (Prasad and Kan, 2017) models respectively. Meng et al.(2017) uses an encoder–decoder model with a copy mechanism for keyphrase extraction (as a special case of

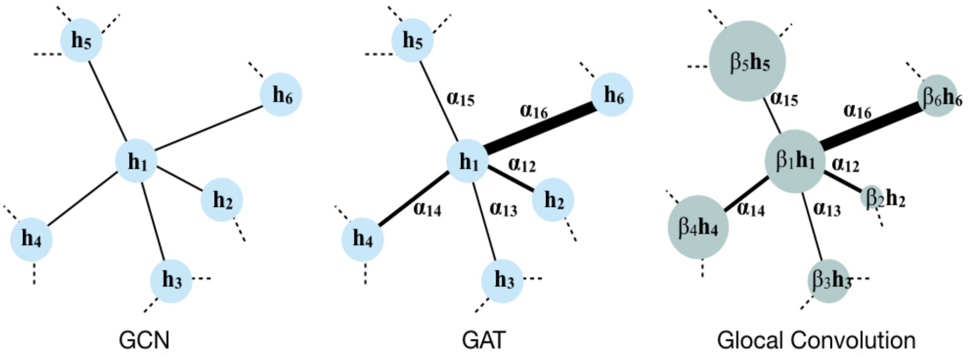


Figure 1: Graph Convolution Model Architectures. Illustrations of the Graph Convolution Network (GCN, left), Graph Attention Network (GAT, center) and our proposed Glocal technique (right), centered on node h_1 within its $1-$hop neighborhood. TextRank on the complete graph is used to compute parameter β_i. GAT parameterizes the edge weights based on gradients (α_i, also represented by the differing edge widths). Our Glocal technique adds the TextRank score (represented by different scaling of nodes), which is not derived from the gradient.

generation). This state-of-the-art technique exploits a complementary idea of sequential semantic modeling focused on generating keyphrases rather than merely extracting them. However, their model does not address the common scenario of keyphrase extraction from long documents but only for short excerpts (namely, the abstract). This assumption reduces the complexity of the problem for sequential models that can effectively encode short text spans but may be ineffective on full-text. We suspect this is a current limitation of the encoder–decoder based models, which necessarily reduces the entire textual sequence into a single vector during the encoding stage – making it susceptible to the vanishing gradient and representation underfitting on large text. Further advances using the encoder–decoder framework such as (Chen et al., 2018) further explore the sequential modeling architectures by improving the attention mechanism with traditional features like title guidance. Note that many forms of such structural information — such as sectional information and citation graph co-occurrence — can enhance basic models, however without loss of generality, in this work we consider only text-based features for all the models.

3 Method

Our proposed model exploits the strength of both supervised and unsupervised modalities by combining two baseline models. Our Glocal model has two components:

- For learning directly over the graph (classification), we use the recently proposed GCN as our baseline model.
- For incorporating global importance (ranking), we use TextRank as our baseline model.

We will first introduce the preliminaries: *e.g.*, Graph Convolution Network (GCN), followed by the modifications that result in the Graph Attention Network (GAT). We then explain how we modify the local convolution operation, to incorporate global importance scores.

3.1 The Graph Convolution Network (GCN)

Graph $G = (V, A)$, where V is a a finite set of vertices such that $|V| = n$ and $A \in R^{n\times n}$ is an undirected weighted adjacency matrix representing all edges, assume $x : V \to R^n$ maps each node to x_i, which is a n-dimensional feature vector. Spectral filtering on a signal x is then represented as (Defferrard et al., 2016):

$$y = g_\theta(L)x = Ug_\theta(\Lambda)U^T x, \qquad (1)$$

where $L = I_n - D^{-1/2}AD^{-1/2}$ and where I_n is the identity matrix and $D_{ii} = \sum_j A_{ij}$. Further, parameterization and simplification of the filter by (Hammond et al., 2011) results in:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \qquad (2)$$

where the Chebyshev polynomial $T_k(x)$ is computed recursively. Note that Eqn. 2 is K-localized; *i.e.* it depends only on nodes that are at a maximum of K hops away from the target node.

A linear model can approximate this (Kipf and Welling, 2016) resulting in the simplified form:

$$y \approx \theta_0' x + \theta_1' \left(L - I_N\right) x = \theta_0' x - \theta_1' D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \,, \tag{3}$$

with two free, tunable parameters θ_0' and θ_1'. Constraining $\theta = \theta_0' = -\theta_1'$ further simplifies the approximation to a single-parameter form:

$$y \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x \,. \tag{4}$$

Note $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvalues in the range $[0, 2]$, so a re-normalization trick was proposed previously $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ to keep gradients stable. This formulation (GCN) supports the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) \,. \tag{5}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections and $W^{(l)}$ is the layer-specific set of weight parameters. σ denotes an activation function; in the GCN model, ReLU is preferred. $H^{(l)}$ is the matrix of activation in the l^{th} layer; $H^{(0)} = X$.

3.2 The Graph Attention Network (GAT)

Unfortunately, the expressive power represented by $\tilde{A}$ is also its biggest limitation: as such, the model only incorporates features from the neighborhood as weighted by the connecting edge normalized to unit sum (*cf.* Fig. 1, left). To address this, the Graph Attention Network model (Veličković et al., 2018) incorporates learnable scaling for each edge. This introduces a local function $attn : R^{F'} \times R^{F'} \rightarrow R$,

$$a_{ij} = attn(W\vec{h}_i, W\vec{h}_j), \tag{6}$$

that computes a score (attention) per node pair (or *edge*, inclusive of self-loops). Here, the $attn$ operator is a single feed-forward layer employing Leaky ReLU activation. This attention is normalized in GAT as:

$$\alpha_{ij} = \frac{\exp\left(a_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(a_{ik}\right)}, \tag{7}$$

to smooth the gradients. In contrast to GCN, GAT replaces the $\tilde{A}$ with a learned $\tilde{A}'$ where each entry α_{ij} is a normalized score computed on each node pair by the gradient (Fig. 1, center). The normalized attention coefficients are used to compute a linear combination of the features in a node neighborhood $\mathcal{N}$, yielding the equivalent layer-wise propagation rule:

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h}_j^{(l)}\right). \tag{8}$$

In terms of Eqn. 5, the fixed weights in the adjacency are replaced with learned weights or equivalently hard neighborhood is replaced by soft neighborhood. Since all edge weights are parameterized, a number ($= T$) of multiple random initializations learn different representations, resulting in a number of different scalings for multiple linear combinations. A final result is achieved by either concatenating or averaging the multiple representations as formulated in Eqns. 9 and 10, respectively.

$$\vec{h}_i^{(l+1)} = \Big\Vert_{t=1}^{T} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^t \mathbf{W}^t \vec{h}_j^{(l)}\right) \tag{9}$$

$$\vec{h}_i^{(l+1)} = \sigma\left(\frac{1}{T} \sum_{t=1}^{T} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^t \mathbf{W}^t \vec{t}_j^{(l)}\right) \tag{10}$$

3.3 Glocal: GAT with Random Walk Scores

In both GCN and GAT, a model with K layers incorporates the feature for a node up to its K hop neighbors. Though GAT improves upon GCN by assigning different importance to nodes via learned weights as compared to the static edge weight in GCN, it is still a local computation. The attention factor, *i.e.* the scaling coefficients α_{ij}, are the function of pairwise feature interaction within the local neighborhood and do not account for node centrality nor the global graph structure. We fix this with our Glocal model.

Consider the random walk based score generated for the graph G such that:

$$\beta_j = TextRank(i). \tag{11}$$

We introduce this parameter β_j to the GAT model. Considering this as the global importance component to the node, we obtain two alternative formulations that encode the node importance as either an additive or a multiplicative form:

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j^{(l)} + \sum_{j\in\mathcal{N}_i} \beta_j\mathbf{W}'\vec{h}_j^{(l)}\right) \qquad (12)$$

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \beta_j\alpha_{ij}\mathbf{W}\vec{h}_j^{(l)}\right) \qquad (13)$$

From Eqns. 12 and 13, we see two scaling factors α and β, which either reinforce or diminish the other's effect (Fig. 1, right). Unlike α, β is neither calculated locally nor normalized. Not normalizing β is an essential component for having both classification and ranking ability. With normalization, even in a neighborhood with minimal TextRank scores, we will get an aggregation of features with unit sum weights. While without normalization it would perform almost no feature aggregation from such nodes.

In our experiments, we use the formulation of Eqn. 13, as a multiplicative formulation easily enables multi-head attention (achieved by multiplying $\tilde{A}$ with B such that $B_{ii} = \beta_i$). Note that this modeling applies to GCN as well: the second component of the Eqn. 13 actually closely resembles the GCN formulation for our proposed model.

4 Experiments

Dataset	Total	Training	Testing
Schutz (2008)	1,323	1,058*	265*
Krapivin (2009)	2,304	1844*	460*
SemEval (2010)	1,095	708	387

Table 1: Number of documents in our keyphrase extraction datasets. '*' denotes that the dataset does not have an official split; results are based on random splits.

We investigate keyphrase extraction, using the most commonly reported full-text datasets, as shown in Table 1. We divide the training data in 80 : 20 fraction for train and validation splits. Our complete pipeline comprises the following steps:

1. Feature Processing. First we perform TextRank on the complete text of each document, retaining only tokens that are nouns and adjectives, filtering out other words (equivalent to simplex noun phrases). We use the gensim library (Řehůřek and Sojka, 2010) to perform TextRank and compute the scores. This process helps in two ways – first, it gives us the node importance value for each keyphrase, needed by Glocal; second, it helps to manage the graph size and lessen the label skew on the minority positive label by removing extraneous tokens. For documents larger than a max size (1200 tokens) we drop the extra least scored tokens. We find that the tokens that have a TextRank score in bottom 50% possess only 16% of partial or full keyphrases in the validation dataset. Hence dropping them from the processing does not affect the recall much. The nodes of the graph are single tokens and not complete phrases, therefore all the tokens of multi-token keyphrases are marked as $keyphrase$ during learning ion the graph. For the node/keyphrase representations, we map our vocabulary to GloVe embeddings using the $2.2M$ vocabulary sized, 300 dimension vector variant (Pennington et al., 2014). For reference, we observe GloVe covers about 90% of the words overall 3 datasets. We then extract various textual features for the candidate keyphrases, including their position of the first occurrence, $tf.idf$ and n-gram count. These features are appended to the word embedding to obtain a final feature vector representing each node. Rather than discard them, we choose to append the n-gram features to retain rich lexical signals obtained from the tokens.

2. Learning. The second step is to train the model with the formulated graphs. We use a 2-layer network with 128 units with ReLU activations for hidden layers, followed by a simple 2-way softmax classification layer ($keyphrase$, $\neg keyphrase$). We further employ 8 attention heads at all layers. We follow Glorot initialization (Glorot and Bengio, 2010) for all setting initial parameters weights, use a dropout of 0.5, and employ a $L2$ regularization of 0.001. We train with Adam optimizer on cross-entropy loss and initial learning rate of 0.01 for 200 epochs using an early stopping strategy with patience set to 20 epochs. In both evaluation and training, as gold standard keyphrases have multiple tokens, we use each token of the gold keyphrase as the true label for each token.

3. Post-processing. This step reconstructs the multi-token keyphrase from the probability scores as generated by the Glocal model. This formation step then requires a re-ranking (calculating $R(p)$) of the resultant phrase as:

$$R(p) = len(p) * \sum_{w_i \epsilon p} r(w_i) \qquad (14)$$

	Schutz			Krapivin			SemEval		
Model	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15
tf.idf	11.3	13.7	15.2	6.9	7.3	9.4	9.1	12.2	13.5
TextRank (Mihalcea and Tarau, 2004)	10.2	12.4	14.9	7.6	9.3	9.9	11.2	14.4	15.2
RNN	3.2	3.6	4.0	2.6	2.9	3.6	3.0	3.2	3.7
GRU	3.8	3.2	3.9	3.1	3.4	5.1	2.6	2.8	3.9
CopyRNN	5.8	6.2	7.5	6.6	6.9	7.1	5.4	5.6	6.1
CopyRNN† (Meng et al., 2017)	29.3	30.2	32.2	**26.2**	25.3	**27.1**	28.7	29.4	31.1
GCN (Kipf and Welling, 2016)	16.7	17.8	19.2	19.2	19.8	20.9	18.7	19.5	21.4
GAT (Veličković et al., 2018)	25.2	28.1	29.3	21.1	23.1	24.2	22.5	26.8	25.9
Glocal	**30.7***	30.3	**33.9***	24.7	**25.6**	**27.1**	**28.9**	**29.8**	**33.5***

Table 2: Main comparative system evaluations on keyphrase extraction. All figures are $F_1@K$. † uses only abstract, rest all models are trained on full-text, * shows significant improvement over strongest baseline CopyRNN.

The initial rank of each candidate token is in this case equal to the probability of the $keyphrase$, i.e., $\forall w_i \in p$, $r(w_i) = p_{keyphrase}(w_i)$. We also constrain the process such that the actual word sequence must appear in the original text.

An important note that we strictly do not normalize the β_i scores; the re-ranking process and the preservation of raw scores work in tandem. Topologically, such graphs generated from textual data often have a few dense neighborhoods and many sparse ones, resulting in significant raw score differences that can benefit from scaling down the feature appropriately. If normalization is done in each neighborhood (as done for α_i), it will scale up individual nodes in a sparse neighborhood and suppress nodes in a dense neighborhood, the reverse of the intended operation.

5 Results

We compare our Glocal model's results against other models on the core task of keyphrase extraction. We select baselines which represent the related state-of-the-art under particular learning paradigms: unstructured retrieval-based ($tf.idf$), unsupervised graph-based (TextRank), supervised sequence learning (RNN and derivatives) and supervised structured models (GCN and derivatives). The marked performance differences help us to ablate and understand the gain brought about by the supervision directly on the graph over unsupervised graph-based techniques, as well as that brought by incorporation of the random walk-based score into the supervised model. We also report TextRank and $tf.idf$ baselines to measure ablative effects, as they contribute to Glocal.

For the sake of comparison, we restrict our comparison to modeling approaches, deciding to keep the feature inventory constant; *i.e.* textual and statistical features. Closely related work discussed earlier — such as MIKE (Zhang et al., 2017) — are not directly comparable, since such works may use many other orthogonal features. Similarly, many supervised techniques use additional features. The best reported SemEval-2010 systems show very close performance to our proposed model, but they take advantages of other sources of side information, such as Wikipedia term acquisition, logical section information, bagging, and ensembles; hence, they are not directly comparable (reported in (Kim et al., 2010)). We argue that our main contribution is in the capacity of modeling; other features utilized in prior work can enhance Glocal's performance and suitable incorporation is future work. In a similar vein, PositionRank (Florescu and Caragea, 2017b) enhances the random walk itself which can again act as a replacement for the TextRank in our model and is thus not strictly a comparable method.

We argued earlier that the recently proposed supervised encoder–decoder based CopyRNN (Meng et al., 2017) deep learning models trained using word embedding do not scale well to the full-text setup. In their work, they only report results for extracting (and generating) keyphrases from abstracts. For direct comparison, we have retrained these models in the full-text scenario to validate our claim that these models have difficulty with scaling. Further, we still compare with the abstract-only trained model for CopyRNN to benchmark both approaches.

The results are reported in F_1@K where $F_1 = 2 * precision * recall/(precision + recall)$ and K is the number of keyphrases generated by the model. Table 2 shows that our Glocal model outperforms both GCN and GAT; and that no other text-only based model shows comparable performance. In the full-text scenario, sequential neural models fare comparably to TextRank, Text example from SemEval: Edge Indexing in a Grid for Highly Dynamic Virtual Environments. Newly emerging game-based application systems such as Second Life provide 3D virtual environments where multiple users interact with each other in real-time. They are filled with autonomous, mutable virtual content which is continuously augmented by the users......

Gold annotations (Author-assigned): 3d object stream, object pop problem, spatial index, visibl model
Gold annotations (Reader-assigned): edg index, dynam virtual environ, game-base applic, mutabl virtual content, spatial databas, spatial index method, real-time visibl test,object-initi view model, object pop, 3d spatial extens

TextRank: *virtual environ*, *applic*, user, interac, system
GAT: *environ*, *method*, *databas*, *model*, user
Glocal: **dynam virtual environ**, *real*, **edg index**, game, *applic*

Text example from Schutz: Debugging Larch Shared Language Specifications. The checkability designed into the LSL (Larch shared language) is described, and two tools that help perform the checking are discussed. LP (the Larch power) is the principal debugging tool. Its design and development have been motivated primarily by work on LSL, but it also has other uses (e.g., reasoning about circuits and concurrent algorithms). Because of these other uses, and because they also tend to use LP to analyze Larch interface specifications, the authors have tried not to make LP too LSL-specific......

Gold annotations: debugging, formal specification, parallel programming, inference mechanisms, consistency, design, program debugging, larch power, larch shared language specifications, checkability, concurrent algorithms, development, static semantics, theory containment

TextRank: *larch*, *programming*, program, *language*, *algorithm*
GAT: *parallel*, **parallel programming**, *language*, *larch*, *interface*
Glocal: *larch shared*, **parallel programming**, **design**, **program debugging**, *programming*

Figure 2: Examples from the SemEval and Schutz datasets. Full matches are bolded; partial matches, italicized. Output from SemEval dataset are stemmed before evaluation

but the recent graphical deep learning models represented by GCN and GAT outperforms it easily. Further, notice that TextRank alone performs weakly in comparison to GAT, but does synergize well with the base GAT in our Glocal system yielding state-of-the-art performance over all three datasets. These results are consistent microscopically as well. Fig. 2 gives two illustrations of keyphrases generated by TextRank, GAT, and Glocal, which show the ability of Glocal to pull out a larger ratio of exact keyphrase matches.

We drill down on SemEval dataset for a closer look, as in Table 3, which has the highest positive label ratio at 14.8 keyphrases per document on average. Keyphrases in SemEval are further classified as either author- or reader-annotated. Despite the task being difficult (15% of the reader- and 19% of the author-assigned keyphrases are not in the text) Glocal's performance edge holds out well, and the results are consistent on both forms of assigned keyphrases.

We experiment with adding the neighborhood normalization for β_j which resulted in a significant decrease in performance of the Glocal model. To further explore this issue of whether normalization thus formed classification model has any benefit over GCN and GAT, we experiment with the Cora and Citeseer scientific document classification tasks (as reported in (Kipf and Welling, 2016) and (Veličković et al., 2018)) which assigns a document into one of six or seven topical categories. We find that the incorporation of global random walk information does not influence topical categorization much (with a minor gain in the case of Cora dataset), and hence Glocal's performance is almost identical to the less expressive GAT model in terms of classification.

Model	Author	Reader
KP-Miner	17.1%	21.5%
Maui	16.2%	16.1%
TextRank	14.5%	15.1%
GAT	25.5%	26.0%
Glocal	**32.2%**	**34.5%**

Table 3: Summary of the fine-grained F_1@15 on SemEval. We compare with other state-of-the-art SemEval systems using text-only feature-based models.

6 Discussion

We now discuss three aspects of the model with respect to the task of keyphrase extraction.

1. Feature versus Scaling. TextRank scores are used in supervised classifiers, traditionally incorporated as a feature. How well would just incorporating TextRank as an additional feature to GAT work? While it does help to improve performance, it does not incorporate the ranking component element in the model; *i.e.* even nodes with very low centrality can positively contribute to its neighbors' scores. Glocal's tight integration is beneficial as it allows further gradient optimization, rather than just simply adding a new feature dimension. Our experiment with adding TextRank as an additional feature show $\approx 2\%$ performance gain for GCN and $< 0.5\%$ performance gain for GAT on average as compared to Glocal showing gain of 5% on average. The behavior is consistent with that of the classification model as in the absence of β_j, the GAT model is inherently a classification model.

As an example (*cf.* Fig. 1, right), consider a neighboring Node h_5 to the prospective keyphrase represented by Node h_1. Node h_5 has a low similarity with the target (low α_{15}), but if a prominent node in the graph will be accorded a higher β_5. In this way, such a node channels more to its neighbor, exerting comparatively steeper gradients to less essential nodes, hence giving a larger chance for Node h_1 to be considered a keyphrase. However, without Glocal's scaling mechanism, such modeling is not captured and essentially unlearnable. We further elaborate on this point next, which shows how our scaling is equivalent to known embedding aggregation techniques (in contrast to adding an extra feature dimension).

2. TextRank Averaged Node Embedding. Kipf and Welling argue that GCN is analogous with node embedding approach with the Weisfeiler-Lehman algorithm for graph isomorphism. Namely, for every node i in V, and scalar feature h_i, the algorithm repeats for k steps or until convergence:

$$\vec{h}_i \leftarrow hash\left(\sum_{j\in\mathcal{N}_i} \vec{h}_j\right). \tag{15}$$

In case $\vec{h}_j$ is a scalar this as well represents the TextRank.

Comparing with layer-wise propagation rules for GAT, GCN, and Glocal, we find a common structure among the methods for incorporating features. Eqn. 16b (representative of GAT) converts a constant aggregation of Eqn. 16a (representing GCN) to be a locally learnable parametric scaling. Eqn. 16c (Glocal) further factorizes the scaling with one locally learnable parametric and one global random walk score. In short, our means of generating node representations (*i.e.*, the post-training embedding) weights the representations for each node/candidate keyphrase by its random walk score. Such a procedure generates stable representations, similar to $tf.idf$ weighted word embeddings used for sentence representation. Such techniques have shown better performance as compared against complex aggregators (like LSTMs) on tasks with insufficient data to train end-to-end models.

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j\in\mathcal{N}_i} c_{ij}\vec{h}_j^{(l)}\mathbf{W}\right), \tag{16a}$$

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\vec{h}_j^{(l)}\mathbf{W}\right), \tag{16b}$$

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j\in\mathcal{N}_i} \beta j\alpha_{ij}\vec{h}_j^{(l)}\mathbf{W}\right). \tag{16c}$$

3. Generating Longer Keyphrases. The re-ranking trick for promoting the generation of longer keyphrases – discussed earlier and in the final step of our pipeline – is fielded in many systems (Zhang et al., 2017). This is a difficult requirement to incorporate directly into the model, relegating such techniques to post-processing by default. A nice offshoot effect of our model is that it implicitly favors generating longer keyphrases, due to the inherent nature of the graph convolution operator in aggregating neighboring nodes' features. This, compounded with a high random walk score for the prospective keyphrase node, results in a higher fraction of features propagating from such nodes being included to its neighboring nodes. In turn, this favors dense local keyphrase neighborhoods of highly weighted keyphrases, making the re-ranking step easier.

7 Conclusion

We have presented Glocal, a global plus local graph convolution model for incorporating the global importance of the node within the local convolution operation for supervised learning on graphs. We argue both theoretical and validate empirically that such a model has benefits in strengthening the graph node ranking component, particularly helpful in tasks such as keyphrase extraction. On our detailed experiments on keyphrase extraction on 3 real-world full-text datasets, our model achieves better performance than traditional, graph-based unsupervised graph-based ranking models and bests sequential supervised classifiers as well. The specific component of incorporating global importance further improve the performance by up to 8.0% absolute F_1 on different evaluation criteria on full-text setup as compared to GAT and up to 2% absolute gain as compared to CopyRNN.

Acknowledgments

We would like to acknowledge the support of the NExT research grant funds, supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC @ SG Funding Initiative. We would also like to gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce GTX Titan X GPU used for this research.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The AI2 system at SemEval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 592–596.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 546–555. Association for Computational Linguistics.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3844–3852.

Corina Florescu and Cornelia Caragea. 2017a. A position-biased PageRank algorithm for keyphrase extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Corina Florescu and Cornelia Caragea. 2017b. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1024–1034.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics, Posters*, pages 365–373. Association for Computational Linguistics.

Shenhao Jiang, Animesh Prasad, Min-Yen Kan, and Kazunari Sugiyama. 2018. Identifying emergent research trends by key authors and phrases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 259–269. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*, 47(3):723–742.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference for Learning Representations*.

Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. *Technical Report DISI-09-055*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems*, pages 849–856.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Animesh Prasad and Min-Yen Kan. 2017. WING-NUS at SemEval-2017 task 10: Keyphrase identification and classification as joint sequence labeling. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 973–977.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA.

Alexander Thorsten Schutz. 2008. Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods. *Master's thesis, National University of Ireland*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations*.

Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. MIKE: keyphrase extraction by integrating multidimensional information. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pages 1349–1358. ACM.

A Study of Latent Structured Prediction Approaches to Passage Reranking

Iryna Haponchyk
DISI, University of Trento
38123 Povo (TN), Italy
iryna.haponchyk@unitn.it

Alessandro Moschitti[*]
Amazon
Manhattan Beach, CA, USA
amosch@amazon.com

Abstract

The structured output framework provides a helpful tool for learning to rank problems. In this paper, we propose a structured output approach which regards rankings as latent variables. Our approach addresses the complex optimization of Mean Average Precision (MAP) ranking metric. We provide an inference procedure to find the max-violating ranking based on the decomposition of the corresponding loss. The results of our experiments on WikiQA and TREC13 datasets show that our reranking based on structured prediction is a promising research direction.

1 Introduction

The current state-of-the-art learning approaches for answer sentence reranking in question answering (QA) are mostly based on learning pairwise ranking signals or simple binary classification (relevant versus irrelevant labels). Intuitively, global information over a rank should improve the ranker accuracy. Thus, there have been promising attempts to learn global ranking functions which encompass the signals of all the candidates for a given query (Chapelle et al., 2007; Weston and Blitzer, 2012; Le et al., 2018). These works employ the structured output learning framework to represent a ranking as a structured object, with respect to which it is possible to directly optimize a ranking measure.

Direct optimization of the target ranking measures is affordable when they are factorizable, e.g., the structural SVM of Chapelle et al. (2007) makes use of the factorization properties of the Normalized Discounted Cumulative Gain (NDCG) ranking score. In contrast, MAP is rather complex making its treatment harder. Yue et al. (2007) could still find an exact solution to the hinge-loss relaxation of Average Precision (AP) for the structural SVM approach. It is found for the particular case of a combined feature mapping of inputs and structured outputs. Such a mapping accounts for respective orderings of the pairs of candidate items, where one item is relevant and the other is not, without explicitly encoding the order of all the items in the rank. Encoding such order (Chapelle et al., 2007; Weston and Blitzer, 2012), i.e., adding yet more complexity to the structural feature space, might lead to intractability of the previous exact max-violating inference with respect to MAP. Furthermore, the feature representation of the gold standard rankings, which could be many for a given candidate list of items, is not unique anymore.

In this work, we study the effect of using structured ranking representations (Chapelle et al., 2007) within the large-margin structured prediction framework versus direct MAP optimization on the most representative task of QA, i.e., passage reranking. To make two ends meet, we have to tackle the above two issues, i.e., i) intractability of the max-violating inference with respect to MAP, and ii) multiplicity of the ground truths.

Regarding the latter, it should be noted that different rankings can correspond to optimal performance, thus, Chapelle et al. (2007) select one among all possible correct rankings at random to build the ground truth for training. Weston and Blitzer (2012) bypass the necessity of comparison to a complete ranking during training and sample the candidate pairs. In this work, we show how this issue can be seamlessly circumvent using the latent structured prediction formulation.

For optimizing MAP, we derive a strict decomposition of the loss corresponding to AP and propose an approximate method for inference of the max-violating constraint with respect to it. More

---

[*] Most of this work was carried out before joining Amazon.

Proceedings of NAACL-HLT 2019, pages 1847–1857
Minneapolis, Minnesota, June 2 - June 7, 2019.

specifically, we provide two structured output approaches optimizing the MAP metric based on Latent Structured Perceptron (LSP) (Sun et al., 2009; Fernandes et al., 2014) and Latent Structural SVM (LSSVM) (Yu and Joachims, 2009) algorithms.

We compare LSP and LSSVM using our MAP optimization strategy on WikiQA (Yang et al., 2015) and TREC13 (Wang et al., 2007) datasets against an SVM classifier and SVM^{map} – the structural approach of Yue et al. (2007). All the models use state-of-the-art traditional feature vectors for the task. Our experiments on WikiQA dataset show a large improvement of our structural approaches over the SVM baseline, i.e., more than 7 absolute points in MAP, MRR and Precision@1.

However, we acknowledge the fact that neural models can produce better representations, which can lead to a superior performance. Thus, to collocate our results in a more general setting, we also carried out experiments using the embeddings of questions and passages, produced by an accurate Convolutional Neural Network (CNN) for passage reranking. In this setting, the structured output models approach the state of the art, confirming the positive impact of our models, which may also be used to train neural networks.

2 Related work

The work related to our approach can be divided in two research areas: (i) structured prediction for reranking problems and (ii) passage reranking in question answering systems.

Structured prediction Seminal work on large-margin structured prediction is by Tsochantaridis et al. (2004), which enables the optimization of multivariate loss functions, exploiting structural dependencies within complex output variables. Chapelle et al. (2007) devise an approach based on the structural SVM, which enables direct optimization of the NDCG ranking measure. Le et al. (2018) facilitate the optimization of a range of ranking measures, within a unified structured output formulation. However, they do not provide any solution for MAP. Weston and Blitzer (2012) optimize a retrieval AUC loss, which decomposes into pairwise decision variables.

The approach most similar to ours is SVM^{map} by Yue et al. (2007), who provide a structural solution for optimizing MAP. They find exactly the max-violating ranking structure with respect to AP within the structural hinge loss formulation. This is done for the case when a ranking is represented by aggregating pairwise outputs between all the relevant and all the irrelevant items in the rank. Our approach is an alternative to this technique, providing an approximate max-violating inference with respect to AP for a more general case of the ranking represenation.

Passage Reranking Most representative *pre-neural networks* work for answer selection/passage reranking is from Wang et al. (2007), who used quasi-synchronous grammar to model relations between a question and a candidate answer with syntactic transformations. Heilman and Smith (2010) and Wang and Manning (2010) applied Tree Edit Distance (TED) to learn the match between question and passage. Yao et al. (2013) applied linear chain CRFs with features derived from TED. Yih et al. (2013) used lexical semantics to build a word-alignment model.

Most recently, Deep Neural Networks (DNNs) have shown to be more competitive. DNN can learn relational patterns between a question (Q) and its passage (P) in a variety of ways, e.g., (i) by using a Q-to-P transformation matrix and simple Q-to-P similarity features (Yu et al., 2014; Severyn and Moschitti, 2015), (ii) by relying on RNN and LSTM architectures (Wang and Nyberg, 2015; Shen et al., 2017), (iii) by employing attention components (Yin et al., 2016; Shen et al., 2017; Wang et al., 2016a), (iv) by decomposing input into similarity and dissimilarity matches (Wang et al., 2016b) or (v) by comparing-aggregating matching results (Wang and Jiang, 2017; Bian et al., 2017).

Since our baselines, SVM and SVM^{map}, as well as our proposed models, LSP and LSSVM, do not apply such transformations, they may perform lower than the state of the art. Thus, we will use the embedding vectors generated by a CNN to show that they can achieve the state-of-the-art accuracy.

3 Structured prediction for ranking

In this section, we provide the task formulation with an introduction of structured prediction algorithms.

3.1 Task formulation

We have training examples of the form $\{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i = (q_i, D_i)$, q_i is a query, $D_i = \{d_i^j\}_{j=1}^{N_i}$ is a list of candidate items corresponding to q_i and

N_i is the number of candidates. $y_i = \{y_i^j : y_i^j \in \{0,1\}\}_{j=1}^{N_i}$ is a vector of gold item labels, label y_i^j corresponding to item d_i^j, taking value of 1 for relevant (good or positive) items and 0 – for irrelevant (bad or negative). The task is to learn to predict, for each example $\mathbf{x}_i$, a ranking of its items $r(\mathbf{x}_i) = r(q_i, D_i)$, such that the relevant items, d_i^j with gold labels $y_i^j = 1$, are always at top positions in $r(q_i, D_i)$. Finally, $r(\mathbf{x}_i) = \{r^j\}_{j=1}^{N_i}$ is a permutation of D_i. In the following, we omit the example index i for simplification of the notation.

3.2 Structured prediction

Generally, structured output approaches aim at linking structured input and output patterns. More formally, in a linear case, such algorithms learn a scoring function $f : X \times Y \to \mathbb{R}$, $f(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \Psi(\mathbf{x}, \mathbf{y})$, where $\Psi(\mathbf{x}, \mathbf{y})$ is a combined feature mapping of input variables X and output variables Y. The predicted structure is derived as $\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y})$. Online structured prediction approaches, e.g., the structured perceptron by Collins (2002), are based on gradient descent updates of the model on the current predicted structure $\hat{\mathbf{y}}$ against the correct structure $\mathbf{y}^* : \mathbf{w} \leftarrow \mathbf{w} + \Psi(\mathbf{x}, \mathbf{y}^*) - \Psi(\mathbf{x}, \hat{\mathbf{y}})$. The large-margin perceptron variants augment the inference with the structural loss (Fernandes and Brefeld, 2011): $\hat{\mathbf{y}} = \underset{\mathbf{y} \in Y}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}^*, \mathbf{y})$. When exhaustive search over Y is impossible, the problem can still be solved if the loss Δ and the scoring function $f(\mathbf{x}, \mathbf{y})$ are decomposable over the subparts of the structure $\mathbf{y}$ and there exists an efficient procedure for finding the argmax using such a decomposition.

3.3 Structured prediction for ranking

The structured prediction framework for ranking (Chapelle et al., 2007; Le et al., 2018) considers a joint feature representation of an input example $\mathbf{x}$ together with an output ranking r: $\Psi(\mathbf{x}, r) = \Psi(q, D, r)$, which factorizes over the individual feature representations of items with respect to the query, weighted relatively to the item positions j in the rank:

$$\Psi(\mathbf{x}, r) = \Psi(q, D, r) = \sum_{j=1}^{N} v_j \psi(q, r^j). \quad (1)$$

The typically used weighting schema, v, implies non-increasing weights associated with the positions j: $v_1 \geq v_2 \geq ... \geq v_N \geq 0$, where the importance decreases gradually from the top to the bottom of the ranking. Inferring a ranking corresponding to a linear model $\mathbf{w}$, i.e., finding

$$\underset{r \in R(\mathbf{x})}{\operatorname{argmax}} \mathbf{w} \cdot \Psi(\mathbf{x}, r) \quad (2)$$

among all possible rankings, $R(\mathbf{x}) = R(q, D)$, simply reduces to ordering the items by scores $\mathbf{w} \cdot \psi(q, d)$, since v_j are fixed.

As the correct ranking r^* for an example $\mathbf{x}$ is often not unique, Chapelle et al. (2007) select one of the correct rankings at random as a gold label during training. This evidently biases the training towards such ground truths.

3.4 Latent structured prediction

The above problem can be alleviated using a latent structured prediction framework. We describe now the general idea of latent variables, and after that we introduce our approach, in which, we implement this idea for ranking tasks.

Latent variables $\mathbf{h}$ are auxiliary structures which are not fully observed in the training data (Yu and Joachims, 2009). The training examples are extended with $\mathbf{h}$ – $(\mathbf{x}, \mathbf{y}, \mathbf{h})$, and the learning is shifted to the space H of latent output structures $\mathbf{h}$. Normally, each $\mathbf{h}$ corresponds to one $\mathbf{y}$, while the opposite is not the case. The problem of multiplicity of the ground truths $\mathbf{h}$ is overcome by finding the best $\mathbf{h}$ explaining the gold $\mathbf{y}^*$:

$$\mathbf{h}^* = \underset{\mathbf{h} \in H(\mathbf{x}, \mathbf{y}^*)}{\operatorname{argmax}} \mathbf{w} \cdot \Psi(\mathbf{x}, \mathbf{h}), \quad (3)$$

using the current model weights $\mathbf{w}$ at each iteration of the training, and $\mathbf{h}^*$ used as gold labels.

4 Our learning approach

In the following, we describe the two classical steps in structured prediction, i.e., learning and inference. Regarding inference, we show our new approximation of MAP.

4.1 Learning

We deal with a non fully observed case as the ground truth ranking labels r we intend to learn are not given in the input data. The case suits perfectly the latent structural formulation (Sec. 3.4), where r can be regarded as latent variables $\mathbf{h}$. Consider the following latent structural large-margin objective (Yu and Joachims, 2009), in terms of the structured ranking variables:

$$\min_{\mathbf{w}} [\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max_{r\in R(\mathbf{x}_i)} [\Delta(y_i,r) + \mathbf{w}\cdot\Psi(\mathbf{x}_i,r)] - C\sum_{i=1}^{n} \max_{r\in R(\mathbf{x}_i,y_i)} \mathbf{w}\cdot\Psi(\mathbf{x}_i,r)], \tag{4}$$

in which the upper bound on the training loss Δ involves (i) finding the max-violating ranking structure, $\hat{r}_i$, over the set $R(\mathbf{x}_i)$ of all possible rankings for the example $\mathbf{x}_i$, under the first max, and (ii) the current ground truth ranking structure, r_i^*, over the set $R(\mathbf{x}_i, y_i)$ of all rankings that comply with the gold label y_i, under the second max.

We adapt the loss-augmented LSP algorithm (Fernandes and Brefeld, 2011) for ranking. LSP is essentially a gradient descent operated on the objective in Eq. 4 with a gradient taken with respect to the example variable. The pseudocode of our adaptation of the algorithm is shown in Alg. 1.

Iterating over the training examples $(\mathbf{x}_i, y_i)$, the algorithm, for each example, first finds the max-violating $\hat{r}_i$ with respect to a ranking loss $\Delta(y_i, r)$, over the set $R(\mathbf{x}_i)$ for the example $\mathbf{x}_i$ (Line 5). Δ can represent any arbitrary ranking loss. In this work, we instantiate Δ with the loss corresponding to the MAP ranking metric. Sec. 4.2 describes the procedure we use here for the max-violating inference with respect to it.

If the max-violating ranking $\hat{r}_i$ is erroneous (Line 6), the algorithm updates the model $\mathbf{w}$. In Line 7, the current ground truth ranking structure – the best correct r_i^* corresponding to the current model weights $\mathbf{w}_t$ – is found. The search here is restricted to the set $R(\mathbf{x}_i, y_i)$ of all correct rankings of the example $\mathbf{x}_i$, i.e., those at which good items take top positions and bad – bottom positions. Thus, the operation is reduced to simple ordering of the good and bad items (separately) by weights, and putting the former to the top, and the latter – to the bottom of the resulting ranking. This step corresponds exactly to imputing latent variables, described by Eq. 3, in the general latent formulation. In Line 8, we update the weights $\mathbf{w}$ using the structural feature representations (defined by Eq. 1) of the two ranking outputs, the current ground truth r_i^* and the max-violating $\hat{r}_i$.

Algorithm 1 Latent Structured Perceptron for Ranking

```
1: Input: $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{w}$, $C^1$, $T$
2: $\mathbf{w}_0 \leftarrow \mathbf{w}; t \leftarrow 0$
3: repeat
4:     for $i = 1, ..., n$ do
5:         $\hat{r}_i \leftarrow \operatorname{argmax}_{r\in R(\mathbf{x}_i)} \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r) + C \times \Delta(y_i, r)$
6:         if $\Delta(y_i, \hat{r}_i) > 0$ then
7:             $r_i^* \leftarrow \operatorname{argmax}_{r\in R(\mathbf{x}_i, y_i)} \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r)$
8:             $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Psi(\mathbf{x}_i, r_i^*) - \Psi(\mathbf{x}_i, \hat{r}_i)$
9:         end if
10:        $t \leftarrow t + 1$
11:    end for
12: until $t < nT$
13: $\mathbf{w} \leftarrow \frac{1}{t}\sum_{i=1}^{t} \mathbf{w}_i$
    return w
```

Likewise, we adapt the Latent Structural SVM (LSSVM) algorithm (Yu and Joachims, 2009) for ranking. We employ also LSSVM in our experiments for its generalization guarantees. The only minor difference from the LSP adaptation is that, in the LSSVM adaptation, we consider only the top items in the joint feature representation in Eq. 1, i.e., $\Psi(\mathbf{x}, r) = \sum_{j=1}^{P} v_j \psi(q, r^j)$, where P is the number of good/positive items in the candidate list D of the example $\mathbf{x}$. This is relevant only at the training phase and only for the updates of the model; for the max-violating inference, still all the items at all N positions participate. By doing so, we help LSSVM to keep the balance between positive and negative items. The test phase inference (respective to Eq. 2), for both LSP and LSSVM, consists only in ordering of all the items by their weight with respect to the model.

Weston and Blitzer (2012) adopt the same structural feature representation of Eq. 1 for ranking, however, in the latent embedding space. They do online SGD updates in correspondence to the positive-negative item pairs, and not on the whole rank, which relates to the way we proceed with LSSVM. In their case, this is due to the impossibility of the global inference (their model is also augmented with the structural component describing item-item interactions) and the scale of the task (they do ranking for recommendation domain). However, they perform a cascade-like inference of the rank which is scattered over the iterations.

---

[1] C here is a loss scaling parameter.

4.2 Max-violating inference

Our target is to optimize the MAP ranking metric. Thus, in training, we intend to minimize the following loss on structural examples which is the inverse of the average precision (AP): $\Delta_{ap}(y, r) = 1 - AP(y, r)$. AP is a global measure, non-decomposable in a strict sense over the position variables, so that to enable iterative exact inference. Here, we propose a method for approximate inference with respect to Δ_{ap}, which is efficient and enables exact local search.

Let us denote by $P = |\{d|y(d) = 1\}|$ the number of good/positive items in the candidate list D, and by $I_j^+ = I_{[y(r^j)=1]}$ and $I_j^- = I_{[y(r^j)\neq 1]}$ the indicator functions that the item at position j in r is good and not good (positive and negative), respectively. Then,

$$AP(y, r) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+\sum_{k=1}^{j}I_k^+.$$

We can have a strict decomposition of Δ_{ap} over the negative items. We rewrite the AP formula as follows:

$$AP(y, r) = \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + I_j^+) =$$
$$\frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1).$$

Here, I_j^+ inside the parentheses becomes 1 in the right-hand side because $I_j^+ * I_j^+ = I_j^+$. Then,

$$\Delta_{ap}(y, r) = 1 - \frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1) =$$
$$\frac{1}{P}(P - \sum_{j=1}^{N}\frac{1}{j}I_j^+(\sum_{k=1}^{j-1}I_k^+ + 1)) =$$
$$\frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+(j - 1 - \sum_{k=1}^{j-1}I_k^+) =$$
$$\frac{1}{P}\sum_{j=1}^{N}\frac{1}{j}I_j^+\sum_{k=1}^{j-1}(1 - I_k^+) = \frac{1}{P}\sum_{j=1}^{N}\frac{I_j^+}{j}\sum_{k=1}^{j-1}I_k^- =$$
$$\frac{1}{P}\sum_{j=1}^{N-1}I_j^-\sum_{k=j+1}^{N}\frac{I_k^+}{k}. \tag{5}$$

According to the last line of Eq. 5, Δ_{ap} decomposes into a sum of quantities $l_j(y, r) = \frac{1}{P}\sum_{k=j+1}^{N}\frac{I_k^+}{k}$ over all positions j with negative items (those activating I_j^-) except for the last position N.

Note that $I_j^- l_j(y, r)$ gives the loss at position j considering the correct items below position j in the ranking. Therefore, we can use it for a bottom-up (max-violating) inference procedure, which first finds the best candidate item to be put at the lowest position of the rank and proceeds filling the positions in the ascending order. Specifically, we start with the last Nth position of the rank and put there the minimum weighted item:

$$\hat{r}^N = \underset{d\in D}{\operatorname{argmin}}\ v_N\mathbf{w}\cdot\psi(q, d). \tag{6}$$

According to the decomposition in Eq. 5, loss is always 0 at position N. At each of the following steps j, $\hat{r}^{N-j} =$

$$= \underset{d\in D\setminus\{\hat{r}^{N-k}\}_{k=0}^{j-1}}{\operatorname{argmin}} v_{N-j}\mathbf{w}\cdot\psi(q, d) + I_{N-j}^- l_{N-j}(y, \hat{r}). \tag{7}$$

Note that the loss part, $I_{N-j}^- l_{N-j}(y, \hat{r})$, in the above formula will be invariant for all the negative items remained in the candidate list, as well as it is for all the positive ones (equal to 0). Thus, $\hat{r}^{N-j}$ is essentially the argmin taken over only two candidate items from $D\setminus\{\hat{r}^{N-k}\}_{k=0}^{j-1}$: one is the positive item with the minimal weight $\mathbf{w}\cdot\psi(q, d)$, and the other, respectively, is the minimal weighted negative one. It is sufficient then to sort independently the positive and the negative items, in the beginning of the whole procedure, in the increasing order of their weights $\mathbf{w}\cdot\psi(q, d)$. Argmin's in equations 6 and 7 are then to be taken over the first items of the two sorted lists, which have not been selected at previous steps. This goes in line with the observation of Yue et al. (2007), that the max-violating ranking output $\hat{r}$ is an interleaving of such two sorted lists, which turns true also for our choice of the structural joint feature representation $\Psi(\mathbf{x}, r)$ in Eq. 1. However, the exact algorithm for max-violating inference of Yue et al. (2007) cannot be applied in our case, since our Ψ, due to distinctive contributions of items at different rank positions scaled with v_j weights, does not satisfy its conditions for an arbitrary choice of v_j.

Since, in our loss decomposition, the position-wise components are not independent of the decisions for the other positions, using a greedy procedure does not find a global optimum, but finds a local optimum with respect to the loss exactly. Namely, an item chosen at each step is optimal with respect to the partial rank constructed at the previous steps of the inference procedure.

Regarding the running time complexity of our greedy inference procedure, it is bounded by the complexity of the sort operation, $\mathcal{O}(N \log N)$, for the candidate lists of size N. In comparison, the worst case complexity of the exact inference in SVM^{map} by Yue et al. (2007) is $\mathcal{O}(N^2)$. In several cases, as shown by our experiments, doing inexact inference produces also higher MAP values compared to SVM^{map}.

5 Experiments

In our experiments, we compare the proposed structural ranking approach with the classification and structural baselines.

5.1 Setup

The setup reports on data, large margin models, CNN and measures we use in our experiments.

WikiQA We use only examples with at least one correct and at least one incorrect answer candidate (Yang et al., 2015) both for training and evaluation. This corresponds to 857 examples for training from train set, 237 – for testing from test, and 122 – for validation from development (dev.) set.

TREC13 We apply the same evaluation strategy as above on TREC13 dataset (Wang et al., 2007), however, for training, we limit to 10 the number of answer candidates for each question. This gives us 970 training examples, 65 examples for validation, and 68 test examples.

5.1.1 Large margin methods

We implement our structural ranking approach described in Sec. 3.3 using both LSP and LSSVM[2] algorithms, denoting the resulting models LSP-AP and LSSVM-AP, respectively.

We compare the models to an SVM baseline using the same feature set for the pairs, (q, d_i), and a polynomial kernel. We consider also a couple of structural baselines: (i) the standard LSP model (Sun et al., 2009), without loss-augmented inference, which we use in order to explore the impact of optimizing the target evaluation measure. This model still follows Alg. 1, however, the difference is that instead of finding the max-violating $\hat{r}_i$ in Line 5, it finds the following max-scoring:

$$\hat{r}_i \leftarrow \underset{r \in R(\mathbf{x}_i)}{\operatorname{argmax}}\, \mathbf{w}_t \cdot \Psi(\mathbf{x}_i, r).$$

And another structural baseline is (ii) SVM^{map} [3] (Yue et al., 2007) – a structural SVM approach affording exact max-violating inference with respect to AP.

Features In our study, we use two feature settings: (i) simple textual similarity features – the setting by Barrón-Cedeño et al. (2016), i.e., *cosine similarity* over the text pair, *the similarity based on the PTK score*, *longest common substring/subsequence measure*, *Jaccard similarity*, *word containment measure*, *greedy string tiling*, *ESA similarity* based on Explicit Semantic Analysis (ESA), and (ii) powerful features coming from the embeddings trained with the state-of-the-art neural networks (Tymoshenko et al., 2017).

Parametrization We use the following weighting schema for the ranking structures: $v_j = \frac{1}{j}$, in LSP, LSP-AP, and LSSVM-AP. LSP-AP requires specifying a loss scaling parameter C. In LSSVM and SVM^{map}, C is the standard trade-off between regularization and training error. In all the three models, we select C on dev. set from the values $\{1, 10, 100, 1000, 2000, 5000\}$. The max number of epochs, T, is set to 100, for both LSP and LSP-AP. We apply weight averaging in the LSP models. We derive the best number, T_{best}, with respect to the MAP score on dev. set. The baseline SVM is trained with polynomial kernels of degree 3.

Cross-validation On TREC13, which has a very small test set we apply cross-validation. On WikiQA, we obtain results on the official test set as well as applying cross-validation. We employ disjoint cross-validation as in Tymoshenko et al. (2017). For each approach, we train 5 models on the training set following the traditional 5-fold cross-validation strategy. We split dev. and test sets in 5 subsets each, and use ith dev. subset to tune the parameters of the models trained on the ith fold, and ith test subset – to test them. We report the results averaged over 5 test subsets.

5.1.2 Convolutional Neural Networks

We borrowed the CNN embeddings of questions and answer passages produced by the neural model for passage reranking of Tymoshenko et al. (2017); Severyn and Moschitti (2015)[4].

---

[2] `www.cs.cornell.edu/~cnyu/latentssvm/`

[3] `http://projects.yisongyue.com/svmmap/`

[4] Several other neural models provide superior accuracy but replicating their results is challenging as shown by

	DEV.		
	MAP	MRR	P@1
SVM	63.37	64.37	50.00
LSP	68.98	69.93	55.74
SVM^{map}	66.15	67.17	51.64
LSP-AP	69.41	69.95	**54.92**
LSSVM-AP	68.67	69.29	**54.92**
LSP-AP*	**69.51**	**70.14**	54.10
LSSVM-AP*	67.48	68.47	51.64
	TEST		
	MAP	MRR	P@1
SVM	54.67	55.90	39.66
LSP	64.35	65.98	48.95
SVM^{map}	61.50	63.10	46.84
LSP-AP	**64.50**	**66.25**	**49.37**
LSSVM-AP	62.39	63.78	46.84
LSP-AP*	63.65	65.48	48.52
LSSVM-AP*	63.74	65.08	47.26

Table 1: Experimental results on WikiQA.

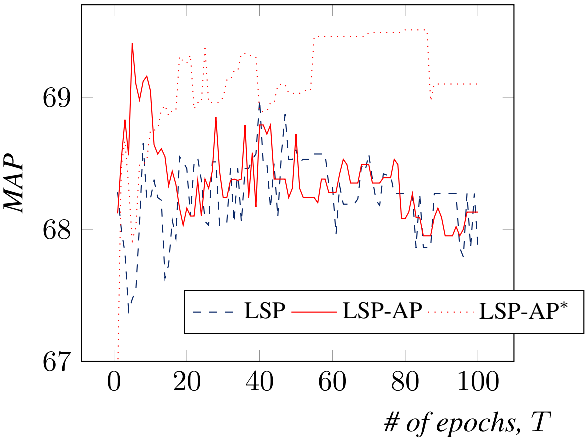


Figure 1: LSP curves on dev. set over the training epochs.

The neural model by Tymoshenko et al. (2017) includes (i) two sentence encoders that map input questions q_i and answer passages d_i^j into fixed size m-dimensional vectors $\phi(q_i)$ and $\phi(d_i^j)$ using a convolutional operation followed by a max pooling layer, and (ii) a feed forward neural network that computes the similarity between the two sentences in the input. The sentence vectors of a question and a passage, $\phi(q_i)$ and $\phi(d_i^j)$, are concatenated together and given in input, at stage (ii), to a standard NN architechture, constituted by a nonlinear hidden layer and a sigmoid output layer, which optimizes binary cross-entropy loss.

Note that we use exactly the concatenated question-passage sentence vectors (CNN embeddings) from stage (i) of the above model as features in SVM and the structured output models: $\psi(q, d) = [\phi(q), \phi(d)]$.

Evaluation metrics We report Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Prec@1 (P@1).

5.2 Comparative analysis on WikiQA dataset

We first report the results using standard similarity features in all of our models. Then, we show the outcome of our models when fed with embeddings produced by the CNN.

5.2.1 Results with textual similarity features

In Tab. 1, we provide the results of our latent structural approaches optimizing MAP in comparison to SVM, LSP, and SVM^{map} models on WikiQA dataset. LSSVM-AP and LSP-AP are better than the SVM classifier baseline by roughly 8 and 10 points, respectively, in terms of MAP metric on the test set. It should be noted that SVM uses kernels, while LSSVM and LSP are simple linear models. Moreover, for SVM, we also had to limit the number of candidates to 10 for each query to make the positive/negative example rate more balanced.

The baseline LSP model (without augmented loss optimization) performs surprisingly well in this setting. It lacks only 0.15 of a MAP point on test set as compared to the highest scoring LSP-AP model. However, as LSP does not optimize the ranking measure directly, it may result unstable. We verified this hypothesis by exploring the performance of the two LSP models (with our loss and without) on dev. set, plotting the learning curves over the training epochs, T. Fig. 1 shows that the LSP-AP curve is distinctly superior to that of LSP only in the beginning of training, before epoch 17. We further verify this issue in cross-validation experiments in Sec. 5.2.3.

LSSVM-AP outperforms the structural baseline SVM^{map}. The latter targets direct optimization of MAP, and it clearly outperforms the SVM classifier. However, it is worse than the standard perceptron model, LSP, which does not optimize MAP. This suggests a higher appropriateness of the structural feature representation for rankings in Eq. 1. Indeed, it encodes the real positioning of items within a ranking (using positional weights v_i), compared to that used by SVM^{map}, agnostic to it and considering only pairwise relevant/irrelevant relative placements among

---

Reimers and Gurevych (2017); Crane (2018). Thus, we rely on the model that is easily replicable and achieves competitive results.

the items.

Finally, in the last lines of Dev. and Test subparts of Tab. 1, we report the results of the models denoted with *. In these model variants, we perform exact search for the max-violating ranking $\hat{r}$ with respect to Δ_{AP} over the structural hypothesis space $R(\mathbf{x})$, e.g., in Line 5 of Alg. 1, instead of our approximate inference procedure in Sec. 4.2.

In the current setting, exhaustive search over all possible rankings is reduced to an inference procedure, which inspects all interleavings of the two sorted lists, of positive and negative items, as pointed out in Sec. 4.2. In addition to the sorting complexity, this operation needs to traverse through a subset R' of ranking structures r, $R' \subset R(\mathbf{x})$, where $|R'|$ is estimated by the binomial coefficient $\binom{N}{N-P}$, which, for fixed P, grows polynomially as $\mathcal{O}(N^P)$. Recall that N is a total number of items in the candidate list D, among which P items are positive. In the context of WikiQA dataset, this is affordable, since the candidate lists D_i of the training examples are relatively short.

This way, LSSVM-AP* adds around 1.3 of a MAP point to its result using our approximate inference. LSP-AP*'s best number of epochs on dev., T_{best}, gives a nearly identical result on the test set in terms of MAP. However, its scores are lower compared to those of LSP-AP.

Fig. 1 illustrates a clear advantage of the direct optimization of AP using exact inference, which results in the best curve on dev. for LSP-AP*, stabilizing to its highest values on the interval between epochs 55 and 85. Still such a level of accuracy is nearly reachable by LSP-AP, although actually achieved at a very narrow interval (see the spike around epoch 5), while LSP's curve lies almost always lower. In future, we would like to study the impact of the loss approximation onto the convergence speed of the structural algorithms.

The LSP models in the current setting do not reveal a clear correlation between dev. and test results in Tab. 1, which might (i) signal of insufficient generalization power of LSP, and (ii) suggest that the effect of direct loss optimization can be reached by carefully selecting the epoch's number parameter, T, in the considered feature space. The test set results of LSSVM in terms of MAP instead conform appropriately to the optimized loss function (using approximate versus exact inference) in the large-margin objective in Eq. 4. This should be due to a better generalization capability of the SVM solver.

	DEV.		
	MAP	MRR	P@1
SVM	70.66	70.89	58.20
LSP	67.65	67.70	52.46
SVM^{map}	72.77	73.55	63.11
CNN	71.73	72.04	59.84
LSP-AP	70.67	70.50	56.56
LSSVM-AP	71.27	71.85	59.84
LSP-AP*	**73.67**	**74.55**	**64.75**
LSSVM-AP*	71.88	72.18	60.66
	TEST		
	MAP	MRR	P@1
SVM	65.00	66.70	53.16
LSP	65.28	66.76	52.32
SVM^{map}	66.91	68.63	54.85
CNN	**68.73**	**70.34**	56.12
LSP-AP	64.23	65.56	50.21
LSSVM-AP	67.62	69.24	55.70
LSP-AP*	65.74	67.51	52.32
LSSVM-AP*	67.93	69.73	**56.54**

Table 2: Models on WikiQA trained on CNN embeddings.

5.2.2 Results using neural network embeddings

In these experiments, we used the CNN embeddings, described in Sec. 5.1.2, as features in all of the models. This setting allows us to examine the performance of the models in a more complex and richer feature space, at the level of the state-of-the-art performance. It can also be seen as a coarse way to "neuralize" the structural ranking approaches.

The results of all the models are shown in Tab. 2. As before, we note the relative inconsistency of the performance of the LSP models between dev. and test sets. The non-loss-augmented structured perceptron, LSP, is the weakest of the models on dev. set, while on test it is better than LSP-AP by around 1 point in terms of MAP. It only slightly outperforms now the baseline SVM, which benefited greatly from using the embeddings. Recall, however, that the baseline SVM is trained with kernels. LSP-AP*, reaching considerably higher scores than the rest of the models, including CNN, on dev., is better than LSP by no more than 0.5 of a MAP point.

LSSVM is in general more robust and consistent as with similarity features. Although SVM^{map} outperforms it on dev., LSSVM-AP is better on test in each of the three metrics. LSSVM-AP* with exact inference further improves the results of LSSVM-AP. It outperforms SVM^{map} by more than 1 point in terms of MAP.

It should be noted that the embeddings that we use were trained in a classification setting, thus, giving an additional advantage to the classification models, e.g., the relative improvement of the baseline SVM when passing to embeddings is the highest among the models. Nonetheless, LSSVM approaches closest of all to CNN, with the variant of the model with exact search showing P@1 superior to that of CNN. This suggests that the structural ranking approaches are of decent capacity, and that the optimal solution lies in regions feasible to the structural linear model, considering also the high results of LSP-AP* on dev. set. This is despite the fact that, in contrast to CNN, which trains on the whole training set, we omit examples with only negative and only positive candidates (Sec. 5.1). Exploiting the information from such examples (subject to additional enhancement of our approach, as it would currently perform a zero update on them in Line 8 of Alg. 1 due to equal max-violated and ground truth rankings) might advance the performance. Thus, good features make our models competitive with the state of the art.

5.2.3 Cross-validation experiments

In Tab. 3, we repeat the main experiments in the cross-validation setting, using the similarity features. On average, LSSVM-AP outperforms SVM^{map} in terms of MAP and MRR, as in the standard setting, however, having relatively higher variance across the folds. The LSP models sustain their superiority using the similarity features also in cross-validation, with LSP-AP scoring the best across the models and with the least variance.

5.3 Experiments on TREC13 dataset

The results of our cross-validation experiments on TREC13 are depicted in Tab. 4. LSP-AP slightly improves over the baseline models in terms of MAP. The baseline LSP this time deviates the least over the folds and reaches better P@1 among all the models. LSSVM-AP instead underperforms in this experiment, which might be for the reason of shortage of the data for validation.

It is also true that in this work, by fixing the weighting schema v, we limited our study to one particular case of a structural ranking representation. However, finding an appropriate structural feature space, e.g., to the extent enabled by tuning the positional weights v_j for the particular application, can be potentially beneficial.

	MAP	MRR	P@1
SVM	55.20±3.60	56.68±3.33	39.69±5.27
LSP	64.27±2.52	65.63±2.62	48.10±4.32
SVM^{map}	60.88±2.57	62.27±2.78	45.18±4.19
LSP-AP	**64.47**±2.48	**65.88**±2.17	**48.95**±4.19
LSSVM-AP	62.00±4.97	63.39±4.59	44.79±8.06

Table 3: Cross-validation results on WikiQA.

	MAP	MRR	P@1
SVM	71.68±4.19	82.41±3.40	70.83±5.73
LSP	71.67±2.54	82.54±1.94	**72.08**±4.28
SVM^{map}	71.85±3.69	81.97±2.91	69.17±5.49
LSP-AP	**72.53**±5.29	**82.85**±3.84	71.01±8.57
LSSVM-AP	70.96±4.74	80.97±2.70	69.17±5.49

Table 4: Cross-validation results on TREC13.

6 Conclusions

In this paper, we proposed new structured prediction algorithms for ranking problems. In particular, we designed (i) a new loss function that leads to the direct optimization of MAP; and (ii) two new algorithms, based on LSP and LSSVM solvers, to optimize it. The comparative results on the benchmarks for passage reranking, WikiQA and TREC13, demonstrate an improvement of LSP-AP over the standard SVM classifier, which is particularly large in the case of WikiQA. LSP without any loss augmentation can achieve good performance, as well, subject to accurate tuning of the epoch number parameter. In the same setting, LSSVM-AP is comparable to SVM^{map} baseline.

Finally, we used CNN embeddings as more expressive features in our models. We found that (i) linear models can benefit from them; (ii) LSSVM-AP is more robust than the LSP models to the use of a complex representation; and (iii) traditional max margin methods may not be on par with neural networks on tasks, such as WikiQA, however providing them with right features (embeddings) can make them approach the performance of neural models. This suggests an interesting research line on using our structural models and loss function optimizing MAP in neural models.

Acknowledgments

We would like to thank the anonymous reviewers for their competent and useful suggestions. Many thanks to Kateryna Tymoshenko and Daniele Bonadiman for kindly providing us with the CNN embeddings for our experiments.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A. Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 896–903, San Diego, California, USA.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A Compare-Aggregate Model with Dynamic-Clip Attention for Answer Selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Olivier Chapelle, Quoc V. Le, and Alex Smola. 2007. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Crane. 2018. Questionable answers in question answering research: Reproducibility and variability of published results. *TACL*, 6:241–252.

Eraldo R. Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Machine Learning and Knowledge Discovery in Databases*, pages 407–422, Berlin, Heidelberg. Springer Berlin Heidelberg.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

Quoc V. Le, Alex Smola, Olivier Chapelle, and Choon Hui Teo. 2018. Optimization of ranking measures. *Journal of Machine Learning Research*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. Association for Computational Linguistics.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1236–1242, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 104–, New York, NY, USA. ACM.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017. Ranking kernels for structures and embeddings: A hybrid preference and classification model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 897–902. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. *Proceedings ACL 2015*, pages 707–712.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence Similarity Learning by Lexical Decomposition and Composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan. The COLING 2016 Organizing Committee.

Janson Weston and John Blitzer. 2012. Latent structured ranking. In *Conference on Uncertainty in Artificial Intelligence*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. *NIPS Deep Learning and Representation Learning Workshop*.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 271–278, New York, NY, USA. ACM.

Combining Distant and Direct Supervision for Neural Relation Extraction

Iz Beltagy **Kyle Lo** **Waleed Ammar**
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{beltagy,kylel,waleeda}@allenai.org

Abstract

In relation extraction with distant supervision, noisy labels make it difficult to train quality models. Previous neural models addressed this problem using an attention mechanism that attends to sentences that are likely to express the relations. We improve such models by combining the distant supervision data with an additional directly-supervised data, which we use as supervision for the attention weights. We find that joint training on both types of supervision leads to a better model because it improves the model's ability to identify noisy sentences. In addition, we find that sigmoidal attention weights with max pooling achieves better performance over the commonly used weighted average attention in this setup. Our proposed method[1] achieves a new state-of-the-art result on the widely used FB-NYT dataset.

1 Introduction

Early work in relation extraction from text used directly supervised methods, e.g., Bunescu and Mooney (2005), which motivated the development of relatively small datasets with sentence-level annotations such as ACE 2004/2005, BioInfer and SemEval 2010 Task 8. Recognizing the difficulty of annotating text with relations, especially when the number of relation types of interest is large, others (Mintz et al., 2009; Craven and Kumlien, 1999) introduced the distant supervision approach of relation extraction, where a knowledge base (KB) and a text corpus are used to automatically generate a large dataset of labeled *bags of sentences* (a set of sentences that might express the relation) which are then used to train a relation classifier. The large number of labeled instances produced with distant supervision make it a practical alternative to manual annotations.

However, distant supervision implicitly assumes that all the KB facts are mentioned in the text (at least one of the sentences in each bag expresses the relation) and that all relevant facts are in the KB (use entities that are not related in the KB as negative examples). These two assumptions are generally not true, which introduces many noisy examples in the training set. Although many methods have been proposed to deal with such noisy training data (e.g., Hoffmann et al., 2011; Surdeanu et al., 2012; Roth et al., 2013; Fan et al., 2014; Zeng et al., 2015; Jiang et al., 2016; Liu et al., 2017), a rather obvious approach has been understudied: combine distant supervision data with additional direct supervision. Intuitively, directly supervising the model can improve its performance by helping it identify which of the input sentences for a given pair of entities are more likely to express a relation.

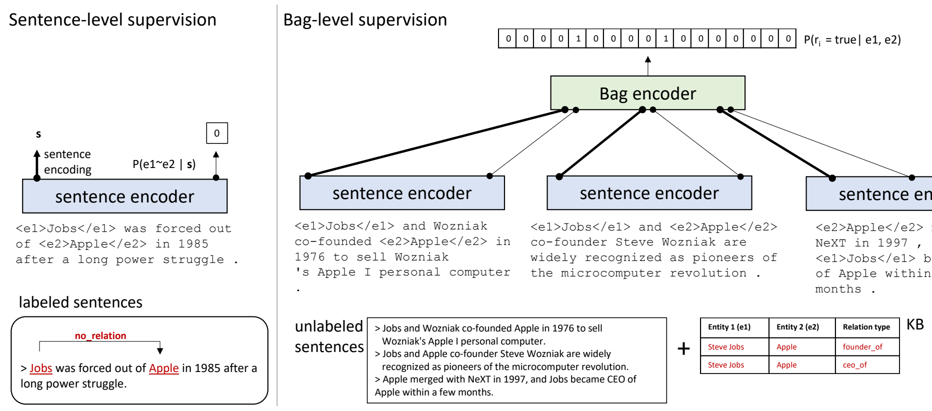
A straightforward way to combine distant and direct supervision is to concatenate instances from both datasets into one large dataset. We show in Section 4.2 that this approach doesn't help the model. Pershina et al. (2014) also observed similar results; instead, they train a graphical model on the distantly supervised instances while using the directly labeled instances to supervise a subcomponent of the model. We discuss prior work in more detail in Section 5.

In our paper, we demonstrate a similar approach with neural networks. Specifically, our neural model attends over sentences to distinguish between sentences that are likely to express some relation between the entities and sentences that do not. We use the additional direct supervision to supervise these attention weights. We train this model jointly on both types of supervision in a multitask learning setup. In addition, we experimentally find that sigmoidal attention weights with max pooling achieves better perfor-

---

[1] https://github.com/allenai/comb_dist_direct_relex/

Proceedings of NAACL-HLT 2019, pages 1858–1867
Minneapolis, Minnesota, June 2 - June 7, 2019.



`<e1>Jobs</e1> was forced out of <e2>Apple</e2> in 1985 after a long power struggle .`

`<e1>Jobs</e1> and Wozniak co-founded <e2>Apple</e2> in 1976 to sell Wozniak 's Apple I personal computer .`

`<e1>Jobs</e1> and <e2>Apple</e2> co-founder Steve Wozniak are widely recognized as pioneers of the microcomputer revolution .`

`<e2>Apple</e2> merged with NeXT in 1997 , and <e1>Jobs</e1> became CEO of Apple within a few months .`

> Jobs was forced out of Apple in 1985 after a long power struggle.

> Jobs and Wozniak co-founded Apple in 1976 to sell Wozniak's Apple I personal computer.
> Jobs and Apple co-founder Steve Wozniak are widely recognized as pioneers of the microcomputer revolution.
> Apple merged with NeXT in 1997, and Jobs became CEO of Apple within a few months.

Entity 1 (e1)	Entity 2 (e2)	Relation type
Steve Jobs	Apple	founder_of
Steve Jobs	Apple	ceo_of

Figure 1: An overview of our approach for combining distant and direct supervision. The left side shows one sentence in the labeled data and how it is used to provide direct supervision for the sentence encoder. The right side shows snippets of the text corpus and the knowledge base, which are then combined to construct one training instance for the model, with a bag of three input sentences and two active relations: 'founder_of' and 'ceo_of'.

mance in this model setup than the commonly used weighted average attention.

The contributions of this paper are as follows:

- We propose an effective neural network model for improving distant supervision by combining it with a directly supervised data in the form of sentence-level annotations. The model is trained jointly on both types of supervision in a multitask learning setup, where the direct supervision data is employed as supervision for attention weights.
- We show experimentally that our model setup benefits from sigmoidal attention weights with max pooling over the commonly used softmax-based weighted averaging attention.
- Our best model achieves a new state-of-the-art result on the FB-NYT dataset, previously used by Lin et al. (2016); Vashishth et al. (2018). Specifically, combining both forms of supervision achieves a 4.4% relative AUC increase than our baseline without the additional supervision.

The following section defines the notation we use, describes the problem and provides an overview of our approach.

2 Overview

Our goal is to predict which relation types are expressed between a pair of entities (e_1, e_2), given all sentences in which both entities are mentioned in a large collection of unlabeled documents.

Following previous work on distant supervision, we use known tuples (e_1, r, e_2) in a knowledge base $\mathcal{K}$ to automatically annotate sentences where both entities are mentioned with the relation type r. In particular, we group all sentences s with one or more mentions of an entity pair (e_1, e_2) into a bag of sentences B_{e_1,e_2}, then automatically annotate this bag with the set of relation types $L^{\text{distant}} = \{r \in \mathcal{R} : (e_1, r, e_2) \in \mathcal{K}\}$, where $\mathcal{R}$ is the set of relations we are interested in. We use 'positive instances' to refer to cases where $|L| > 0$, and 'negative instances' when $|L| = 0$.

In this paper, we leverage an existing dataset of direct supervision for relations. Each direct supervision instance consists of a token sequence s containing mentions of an entity pair (e_1, e_2) and one relation type (or 'no relation'). We do not require that the entities or relation types in the direct supervision annotations align with those in the KB. Furthermore, we replace the relation label associated with each sentence with a binary indicator of 1 if the sentence expresses one of the relationships of interest and 0 otherwise.

Figure 1 illustrates how we modify neural architectures commonly used in distant supervision, e.g., Lin et al. (2016); Liu et al. (2017) to effectively incorporate direct supervision. The model consists of two components: 1) A **sentence encoder** (displayed in blue) reads a sequence of tokens and their relative distances from e_1 and e_2, and outputs a vector s representing the sentence encoding, as well as $P(e_1 \sim e_2 \mid \mathrm{s})$ representing the probability that the two entities are related given this sentence. 2) The **bag encoder** (displayed in green) reads the encoding of each sentence in the bag for the pair (e_1, e_2) and predicts $P(r = 1 \mid e_1, e_2), \forall r \in \mathcal{R}$.

We combine both types of supervision in

a multi-task learning setup by minimizing the weighted sum of the cross entropy losses for $P(e_1 \sim e_2 \mid \mathbf{s})$ and $P(r = 1 \mid e_1, e_2)$. By sharing the parameters of sentence encoders used to compute either loss, the sentence encoders become less susceptible to the noisy bag labels. The bag encoder also benefits from the direct supervision by using the supervised distribution $P(e_1 \sim e_2 \mid \mathbf{s})$ to decide the weight of each sentence in the bag.

3 Model

The model predicts a set of relation types $L^{\text{pred}} \subset \mathcal{R}$ given a pair of entities e_1, e_2 and a bag of sentences B_{e_1,e_2}. In this section, we first describe the sentence encoder part of the model (Figure 2, bottom), then describe the bag encoder (Figure 2, top), then we explain how the two types of supervision are jointly used for training the model end-to-end.

3.1 Sentence Encoder Architecture

Given a sequence of words $w_1, \ldots, w_{|s|}$ in a sentence s, a sentence encoder translates this sequence into a fixed length vector $\mathbf{s}$.

Input Representation. The input representation is illustrated graphically with a table at the bottom of Figure 2. We map word token i in the sentence w_i to a pre-trained word embedding vector $\mathbf{w}_i$.[2] Another crucial input signal is the position of entity mentions in each sentence $s \in B_{e_1,e_2}$. Following Zeng et al. (2014), we map the distance between each word in the sentence and the entity mentions[3] to a small vector of learned parameters, namely $\mathbf{d}_i^{e1}$ and $\mathbf{d}_i^{e2}$.

We find that adding a dropout layer with a small probability ($p = 0.1$) before the sentence encoder reduces overfitting and improves the results. To summarize, the input layer for a sentence s is a sequence of vectors:

$$\mathbf{v}_i = [\mathbf{w}_i; \mathbf{d}_i^{e1}; \mathbf{d}_i^{e2}], \text{ for } i \in 1, \ldots, |s|$$

Word Composition. Word composition is illustrated with the block `CNN` in the bottom part of Figure 2, which represents a convolutional neural network (CNN) with multiple filter sizes. The outputs of the max pool operations for different filter sizes are concatenated then projected into a smaller vector using one feed forward linear layer.

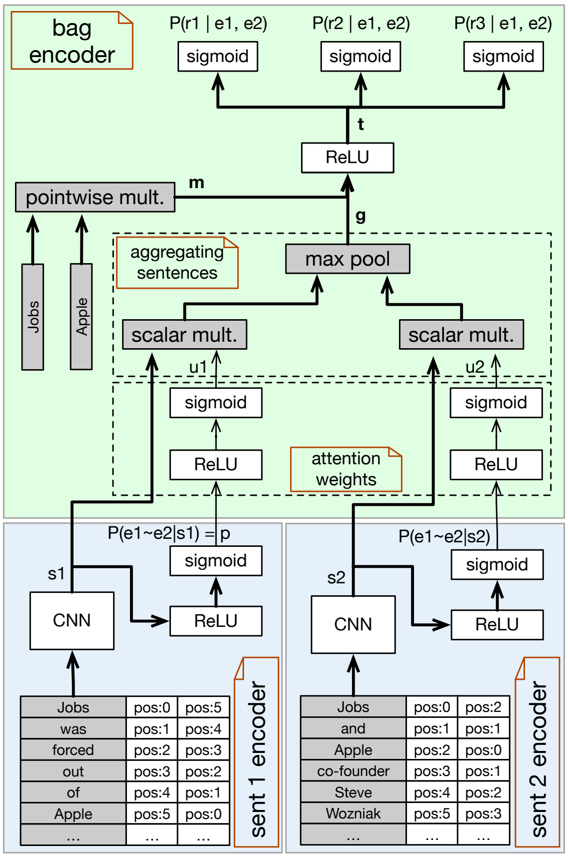


Figure 2: Blue box is the sentence encoder, it maps a sentence to a fixed length vector (`CNN` output) and the probability it expresses a relation between e_1 and e_2 (`sigmoid` output). Green box is the bag encoder, it takes encoded sentences and their weights and produces a fixed length vector (`max pool` output), concatenates it with entity embeddings (`pointwise mult.` output) then outputs a probability for each relation type r. White boxes contain parameters that the model learns while gray boxes do not have learnable parameters. Directly supervised annotations provide supervision for $P(e_1 \sim e_2 \mid \mathbf{s})$. Distantly supervised annotations provide supervision for $P(r = 1 \mid e_1, e_2)$.

Sentence encoding $\mathbf{s}$ is computed as follows:

$$\mathbf{c}_x = \text{CNN}_x(\mathbf{v}_1, \ldots, \mathbf{v}_{|s|}), \text{for } x \in \{2, 3, 4, 5\}$$

$$\mathbf{s} = \mathbf{W}_1 \left[\mathbf{c}_2; \mathbf{c}_3; \mathbf{c}_4; \mathbf{c}_5\right] + \mathbf{b}_1,$$

where CNN_x is a standard convolutional neural network with filter size x, $\mathbf{W}_1$ and $\mathbf{b}_1$ are model parameters and $\mathbf{s}$ is the sentence encoding.

We feed the sentence encoding s into a ReLU layer followed by a sigmoid layer with output size 1, representing $P(e_1 \sim e_2 \mid \mathbf{s})$, as illustrated in Figure 2 (bottom):

$$P(e_1 \sim e_2 \mid \mathbf{s}) = \qquad (1)$$
$$p = \sigma(\mathbf{W}_3 \text{ReLU}(\mathbf{W}_2 \mathbf{s} + \mathbf{b}_2) + \mathbf{b}_3),$$

---

[2] Following Lin et al. (2016), we do not update the word embeddings while training the model.

[3] If an entity is mentioned more than once in the sentence, we use the distance from the word to the closest entity mention. Distances greater than 30 are mapped to the embedding for distance = 30.

where σ is the sigmoid function and $\mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3$ are model parameters.

3.2 Bag Encoder Architecture

Given a bag B_{e_1,e_2} of $n \geq 1$ sentences, we compute their encodings $\mathbf{s}_1, \ldots, \mathbf{s}_n$ as described earlier and feed them into the bag encoder, which combines the information in all of the sentence encodings and predicts the probability $P(r = 1 \mid e_1, e_2), \forall r \in \mathcal{R}$. The bag encoder also incorporates the signal $p = P(e_1 \sim e_2 \mid \mathbf{s})$ from Equation 1 as an estimate of the degree to which sentence s expresses "some" relation between e_1 and e_2.

Attention. To aggregate the sentence encodings $\mathbf{s}_1, \ldots, \mathbf{s}_n$ into a fixed length vector that captures the important features in the bag, we use attention. Attention has two steps: (1) computing weights for the sentences and (2) aggregating the weighted sentences. Weights can be uniform, or computed using a sigmoid or softmax. Weighted sentences can be aggregated using average pooling or max pooling. Prior work (Jiang et al., 2016; Lin et al., 2016; Ji and Smith, 2017) have explored some of these combinations but not all of them. In the ablation experiments, we try all combinations and we find that the (sigmoid, max pooling) attention gives the best result. We discuss the intuition behind this in Section 4.2. For the rest of this section, we will explain the architecture of our network assuming a (sigmoid, max pooling) attention.

Given the encoding $\mathbf{s}_j$ and an unnormalized weight u_j for each sentence $s_j \in B_{e_1,e_2}$, the bag encoding $\mathbf{g}$ is a vector with the same dimensionality as $\mathbf{s}_j$. With (sigmoid, max pooling) attention, each sentence vector is multiplied by the corresponding weight, then we do a "dimension-wise" max pooling (taking the maximum of each dimension across all sentences, not the other way around). The k-th element of the bag encoding $\mathbf{g}$ is computed as:

$$\mathbf{g}_j[k] = \max_{j \in 1,\ldots,n}\{\mathbf{s}_j[k] \times \sigma(u_j)\}.$$

As shown in Figure 2, we do not directly use the p from Equation 1 as attention weights. Instead, we found it useful to feed it into more non-linearities. The unnormalized attention weight for s_j is computed as:

$$u_j = \mathbf{W}_7 \, \mathrm{ReLU}(\mathbf{W}_6 \, p + \mathbf{b}_6) + \mathbf{b}_7.$$

Entity Embeddings. Following Ji et al. (2017), we use entity embeddings to improve our model of relations in the distant supervision setting, although our formulation is closer to that of Yang et al. (2015) who used point-wise multiplication of entity embeddings: $\mathbf{m} = \mathbf{e}_1 \odot \mathbf{e}_2$, where $\odot$ is point-wise multiplication, and $\mathbf{e}_1$ and $\mathbf{e}_2$ are the embeddings of e_1 and e_2, respectively. In order to improve the coverage of entity embeddings, we use pretrained GloVe vectors (Pennington et al., 2014) (same embeddings used in the input layer). For entities with multiple words, like "Steve Jobs", the vector for the entity is the average of the GloVe vectors of its individual words. If the entity is expressed differently across sentences, we average the vectors of the different mentions. As discussed in Section 4.2, this leads to big improvement in the results, and we believe there is still big room for improvement from having better representation for entities. We feed the output $\mathbf{m}$ as additional input to the last block of our model.

Output Layer. The final step is to use the bag encoding $\mathbf{g}$ and the entity pair encoding $\mathbf{m}$ to predict a set of relations L^{pred} which is a standard multilabel classification problem. We concatenate $\mathbf{g}$ and $\mathbf{m}$ and feed them into a feedforward layer with ReLU non-linearity, followed by a sigmoid layer with an output size of $|\mathcal{R}|$:

$$\mathbf{t} = \mathrm{ReLU}(\mathbf{W}_4[\mathbf{g}; \mathbf{m}] + \mathbf{b}_4)$$
$$P(\mathbf{r} = 1 \mid e_1, e_2) = \sigma(\mathbf{W}_5\mathbf{t} + \mathbf{b}_5),$$

where $\mathbf{r}$ is a vector of Bernoulli variables each of which corresponds to one of the relations in $\mathcal{R}$. This is the final output of the model.

3.3 Model Training

To train the model on the distant supervision data, we use binary cross-entropy loss between the model predictions and the labels obtained with distant supervision, i.e.,

$$\mathrm{DistSupLoss} = \sum_{B_{e_1,e_2}} - \log P(\mathbf{r} = \mathbf{r}^{\text{distant}} \mid e_1, e_2)$$

where $\mathbf{r}^{\text{distant}}[k] = 1$ indicates that the tuple (e_1, r_k, e_2) is in the knowledge base.

In addition to the distant supervision, we want to improve the results by incorporating an additional direct supervision. A straightforward way to combine them is to create singleton bags for direct supervision labels, and add the bags to those obtained with distant supervision. However, results in Section 4.2 show that this approach does not improve the results. Instead, a better use of the direct supervision is to improve the model's ability to predict the potential usefulness of a sentence. According to our analysis of baseline models, distinguishing between positive and negative examples is the real bottleneck in the task. Therefore, we use the direct supervision data to supervise $P(e_1 \sim e_2 \mid \mathbf{s})$. This supervision serves two purposes: it improves our encoding of each sentence, and improves the weights used by the attention to decide which sentences should contribute more to the bag encoding. It also has the side benefit of not requiring the same set of relation types as that of the distant supervision data, because we only care about if there exists *some relevant relation* or not between the entities.

We minimize log loss of gold labels in the direct supervision data $\mathcal{D}$ according to the model described in Equation 1:

$$\text{DirectSupLoss} = \sum_{s, l^{\text{gold}} \in \mathcal{D}} - \log P(l = l^{\text{gold}} \mid \mathbf{s})$$

where $\mathcal{D}$ is all the direct supervision data and all distantly-supervised *negative* examples.[4]

We jointly train the model on both types of supervision. The model loss is a weighted sum of the direct supervision and distant supervision losses,

$$\text{loss} = \frac{1}{\lambda + 1}\text{DistSupLoss} + \frac{\lambda}{\lambda + 1}\text{DirectSupLoss} \tag{2}$$

where λ is a parameter that controls the contribution of each loss, tuned on a validation set.

4 Experiments

4.1 Data and Setup

This section discusses datasets, metrics, configurations and the models we are comparing with.

Distant Supervision Dataset (DistSup). The FB-NYT dataset[5] introduced in Riedel et al. (2010) was generated by aligning Freebase facts with New York Times articles. The dataset has 52 relations with the most common being "location", "nationality", "capital", "place_lived" and "neighborhood_of". They used the articles of *2005* and *2006* for training, and *2007* for testing. Recent prior work (Lin et al., 2016; Liu et al., 2017; Huang and Wang, 2017) changed the original dataset. They used all articles for training except those from *2007*, which they left for testing as in Riedel et al. (2010). We use the modified dataset which was made available by Lin et al. (2016).[6] The table below shows the dataset size.

	Train	Test
Positive bags	16,625	1,950
Negative bags	236,811	94,917
Sentences	472,963	172,448

Direct Supervision Dataset (DirectSup). Our direct supervision dataset was made available by Angeli et al. (2014) and it was collected in an active learning framework. The dataset consists of sentences annotated with entities and their relations. It has 22,766 positive examples for 41 relation types in addition to 11,049 negative examples. To use this dataset as supervision for $P(e_1 \sim e_2 \mid \mathbf{s})$, we replace the relation types of positive examples with 1s and label negative examples with 0s.

Metrics. Prior work used precision-recall (PR) curves to show results on the FB-NYT dataset. In this multilabel classification setting, the PR curve is constructed using the model predictions on all entity pairs in the test set for all relation types sorted by the confidence scores from highest to lowest. Different thresholds correspond to different points on the PR curve. We use the area under the PR curve (AUC) for early stopping and hyperparameter tuning. Following previous work on this dataset, we only keep points on the PR curve with recall below 0.4, focusing on the high-precision low-recall part of the PR curve. As a result, the largest possible value for AUC is 0.4.

Configurations. The FB-NYT dataset does not have a validation set for hyperparameter tuning and early stopping. Liu et al. (2017) use the test set for validation, Lin et al. (2016) use 3-fold cross validation, and Vashishth et al. (2018) split the training set into 80% training and 20% testing. In our experiments, we use 90% of the training set for training and keep the other 10% for validation. The main hyperparameter we tune is lambda (section 4.3).

The pre-trained word embeddings we use are 300-dimensional GloVe vectors, trained on 42B tokens. Since we do not update word embeddings while training the model, we define our vocabulary as any word which appears in the training, validation or test sets with frequency greater than two. When a word with a hyphen (e.g., 'five-star') is not

---

[4] We note that the distantly supervised negative examples may still be noisy. However, given that relations tend to be sparse, the signal to noise is high.

[5] `http://iesl.cs.umass.edu/riedel/ecml/`

[6] `https://github.com/thunlp/NRE`

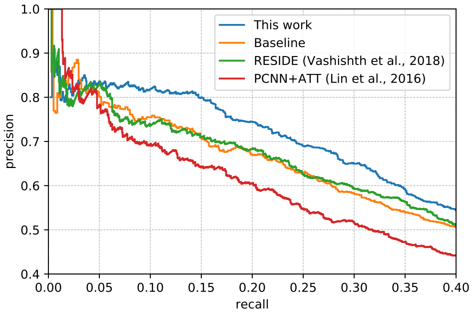
in the GloVe vocabulary, we average the embeddings of its subcomponents. Otherwise, all OOV words are assigned the same random vector (normal with mean 0 and standard deviation 0.05).

Our model is implemented using PyTorch and AllenNLP (Gardner et al., 2017) and trained on machines with P100 GPUs. Each run takes five hours on average. We train for a maximum of 50 epoch, and use early stopping with patience = 3. Each dataset is split into minibatches of size 32 and randomly shuffled before every epoch. We use the Adam optimizer with its default PyTorch parameters. We run every configuration with three random seeds and report the PR curve for the run with the best validation AUC. In the controlled experiments, we report the mean and standard deviation of the AUC across runs.

Compared Models. Our best model (Section 3) is trained on the DistSup and DirectSup datasets in our multitask setup and it uses (sigmoid, max pooling) attention. **Baseline** is the same model described in Section 3 but trained only on the DistSup dataset and uses the more common (softmax, average pooling) attention. This baseline is our implementation of the **PCNN+ATT** model (Lin et al., 2016) with two main differences; they use piecewise convolutional neural networks (PCNNs Pennington et al., 2014) instead of CNNs, and we add entity embeddings before the output layer.[7] We also compare our results to the state of the art model **RESIDE** (Vashishth et al., 2018), which uses graph convolution over dependency parse trees, OpenIE extractions and entity type constraints.

4.2 Main Results

Figure 3 summarizes the main results of our experiments. First, we note that "our baseline" outperforms PCNN+ATT (Lin et al., 2016) despite using the same training data (DistSup) and the same form of attention (softmax, average pooling), which confirms that we are building on a strong baseline. The improved results in our baseline are due to using CNNs instead of PCNNs, and using entity embeddings.



Model	AUC
PCNN+ATT (Lin et al., 2016)	0.247
RESIDE (Vashishth et al., 2018)	0.271
Our baseline	$0.272_{\pm 0.005}$
This work	$\mathbf{0.283}_{\pm 0.007}$

Figure 3: Precision-recall curves and their AUC comparing our model with the baseline and existing models. Baseline is trained on DistSup and uses (softmax, average pooling) attention. Our best model is trained using multitask learning on DistSup and DirectSup and uses (sigmoid, max pooling) attention. Results of Lin et al. (2016) and Vashishth et al. (2018) are copied from their papers.

pooling type	supervision signal	attention weight computation		
		uniform	softmax	sigmoid
average pooling	DistSup	$0.244_{\pm 0.008}$	$0.272_{\pm 0.005}$	$0.258_{\pm 0.020}$
	DistSup + DirectSup	$0.224_{\pm 0.009}$	$0.272_{\pm 0.009}$	$0.256_{\pm 0.009}$
	MultiTask (our model)	$0.220_{\pm 0.012}$	$0.262_{\pm 0.014}$	$0.258_{\pm 0.015}$
max pooling	DistSup	$0.277_{\pm 0.009}$	$0.278_{\pm 0.001}$	$0.274_{\pm 0.004}$
	DistSup + DirectSup	$0.269_{\pm 0.003}$	$0.269_{\pm 0.005}$	$0.277_{\pm 0.012}$
	MultiTask (our model)	$0.266_{\pm 0.007}$	$0.280_{\pm 0.004}$	$0.283_{\pm 0.007}$

Table 1: Controlled experiments for a) how the supervised data is used in the model, b) which function is used to compute attention weights, c) how sentence embeddings are aggregated

A new state-of-the-art. Adding DirectSup in our multitask learning setup and using (sigmoid, max pooling) attention gives us the best result, outperforming our baseline that doesn't use either by 4.4% relative AUC increase, and achieves a new state-of-the-art result outperforming (Vashishth et al., 2018).

We note that the improved results reported here conflate additional supervision and model improvements. Next, we report the results of controlled experiments to tease apart the contributions of different components.

Table 1 summarizes results of our controlled experiments showing the impact of how the training data is used, and the impact of different configurations of the attention (computing weights and aggregating vectors). The model can be trained

---

[7] Contrary to the results in Pennington et al. (2014), we found CNNs to give better results than PCNNs in our experiments. Lin et al. (2016) also compute unnormalized attention weights as $o_j = \mathbf{s}_j \times \mathbf{A} \times \mathbf{q}$ where $\mathbf{s}_j$ is the sentence encoding, $\mathbf{A}$ is a diagonal matrix and $\mathbf{q}$ is the query vector. In our experiments, we found that implementing it as a feedforward layer with output size $= 1$ works better. All our results use the feedforward implementation.

on DistSup only, DistSup + DirectSup together as one dataset with DirectSup expressed as singleton bags, or DistSup + DirectSup in our multitask setup. Attention weights can be uniform, or computed using softmax or sigmoid.[8] Sentence vectors are aggregated by weighting them then averaging (average pooling) or weighting them then taking the max of each dimension (max pooling). (Uniform weights, average pooling) and (softmax, average pooling) were used by Lin et al. (2016), (sigmoid, average pooling) was proposed by Ji and Smith (2017) but for a different task, and (uniform weights, max pooling) is used by Jiang et al. (2016). To the best of our knowledge, (softmax, max pooling) and (sigmoid, max pooling) have not been explored before.

Pooling type. Results in Table 1 show that aggregating sentence vectors using max pooling generally works better than average pooling. This is because max pooling might be better at picking out useful features (dimensions) from each sentence.

Supervision signal. The second dimension of comparison is the use of the supervision signal used to train the model. The table shows that training on DistSup + DirectSup, where the DirectSup dataset is simply used as additional bags, can hurt the performance. We hypothesize that this is because the DirectSup data change the distribution of relation types in the training set from the test set. However, using DirectSup as supervision for the attention weights in our multitask learning setup leads to considerable improvements (1% and 3% relative AUC increase using softmax and sigmoid respectively) because it leads to better attention weights and improves the model's ability to filter noisy sentences.

Attention weight computation. Finally, comparing uniform weights, softmax and sigmoid. We found the result to depend on the available level of supervision. With DistSup only, the results of all three are comparable with softmax being slightly better. However, when we have good attention weights (as provided by the multitask learning), softmax and sigmoid work better than uniform weights where sigmoid gives the best result with 6% relative AUC increase. Sigmoid works better than softmax, because softmax assumes that exactly one sentence is correct by forcing the probabilities to sum to 1. This assumption is not correct for this task, because zero or many sentences could potentially be relevant. On the other hand, sigmoidal attention weights does not make this assumption, which gives rise to more informative attention weights in cases where all sentences are not useful, or when multiple ones are. This makes the sigmoidal attention weights a better modeling for the problem (assuming reliable attention weights).

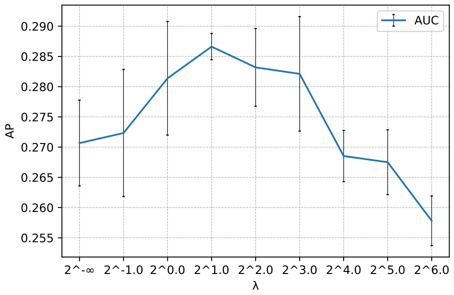


Figure 4: AUC at different λ. X-axis is in log-scale.

4.3 Selecting Lambda.

Although we did not spend much time tuning hyperparameters, we made sure to carefully tune λ (Equation 2) which balances the contribution of the two losses of the multitask learning. Early experiments showed that DirectSupLoss is typically smaller than DistSupLoss, so we experimented with $\lambda \in \{0, 0.5, 1, 2, 4, 8, 16, 32, 64\}$. Figure 4 shows AUC results for different values of λ, where each point is the average of three runs. It is clear that picking the right value for λ has a big impact on the final result.

4.4 Qualitative Analysis.

An example of a positive bag is shown in Table 2. Our best model (Multitask, sigmoid, max pooling) assigns the most weight to the first sentence while the baseline (DistSup, softmax, average pooling) assigns the most weight to the last sentence (which is less informative for the relation between the two entities). Also, the baseline does not use the other two sentences because their weights are dominated by the last one.

5 Related Work

Distant Supervision. The term 'distant supervision' was coined by Mintz et al. (2009) who used relation instances in a KB to identify sentences in

---

[8] We also tried normalizing sigmoid weights as suggested in (Rei and Søgaard, 2018), but this did not work better than regular sigmoid or softmax.

Baseline	This work	Sentences
0.00	0.029	You can line up along the route to cheer for the 32,000 riders, whose 42-mile trip will start in battery park and end with a festival at **Fort Wadsworth** on **Staten Island** .
0.00	0.026	Gateway is a home to the nation's oldest continuing operating lighthouse, Sandy Hook lighthouse, built in 1764; Floyd Bennett field in Brooklyn, which was the city's first municipal airfield; **Fort Wadsworth** on **Staten Island**, which predates the revolutionary war.
0.99	0.027	home energy smart fair, gateway national recreation area, **Fort Wadsworth** visitor center, bay street and school road, **Staten Island**.

Table 2: Weights assigned to sentences by our baseline and our best model. The baseline incorrectly predicts "no_relation", while our best model correctly predicts "neighbourhood_of" for this bag.

a text corpus where two related entities are mentioned, then developed a classifier to predict the relation. Researchers have since extended this approach further (e.g., Takamatsu et al., 2012; Min et al., 2013; Riedel et al., 2013; Koch et al., 2014).

A key source of noise in distant supervision is that sentences may mention two related entities without expressing the relation between them. Hoffmann et al. (2011) used multi-instance learning to address this problem by developing a graphical model for each entity pair which includes a latent variable for each sentence to explicitly indicate the relation expressed by that sentence, if any. Our model can be viewed as an extension of Hoffmann et al. (2011) where the sentence-bound latent variables can also be directly supervised in some of the training examples.

Neural Models for Distant Supervision. More recently, neural models have been effectively used to model textual relations (e.g., Hashimoto et al., 2013; Zeng et al., 2014; Nguyen and Grishman, 2015). Focusing on distantly supervised models, Zeng et al. (2015) proposed a neural implementation of multi-instance learning to leverage multiple sentences which mention an entity pair in distantly supervised relation extraction. However, their model picks only one sentence to represent an entity pair, which wastes the information in the neglected sentences. Jiang et al. (2016) addresses this limitation by max pooling the vector encodings of all input sentences for a given entity pair. Lin et al. (2016) independently proposed to use attention to address the same limitation, and Du et al. (2018) improved by using multilevel self-attention. To account for the noise in distant supervision labels, Liu et al. (2017); Luo et al. (2017); Wang et al. (2018) suggested different ways of using "soft labels" that do not necessarily agree with the distant supervision labels. Ye et al. (2017) proposed a method for leveraging dependencies between different relations in a pairwise ranking framework, while Han et al. (2018) arranged the relation types in a hierarchy aiming for better generalization for relations that do not have enough training data. To improve using additional resources, Vashishth et al. (2018) used graph convolution over dependency parse, OpenIE extractions and entity type constraints, and Liu et al. (2018) used parse trees to prune irrelevant information from the sentences.

Combining Direct and Distant Supervision. Despite the substantial amount of work on both directly and distantly supervised relation extraction, the question of how to combine both signals has not received the same attention. Pershina et al. (2014) trained MIML-RE from (Surdeanu et al., 2012) on both types of supervision by locking the latent variables on the sentences to the supervised labels. Angeli et al. (2014) and Liu et al. (2016) presented active learning models that select sentences to annotate and incorporate in the same manner. Pershina et al. (2014) and Liu et al. (2016) also tried simple baseline of including the labeled sentences as singleton bags. Pershina et al. (2014) did not find this beneficial, which agrees with our results in Section 4.2, while Liu et al. (2016) found the addition of singleton bags to work well.

Our work is addressing the same problem, but combining both signals in a state-of-the-art neural network model, and we do not require the two datasets to have the same set of relation types.

6 Conclusion

We improve neural network models for relation extraction by combining distant and direct supervision data. Our network uses attention to attend to relevant sentences, and we use the direct supervision to improve attention weights, thus improving the model's ability to find sentences that are likely to express a relation. We also found that sigmoidal attention weights with max pooling achieves better performance than the commonly used weighted average attention. Our model combining both forms of supervision achieves a new state-of-the-art result on the FB-NYT dataset with a 4.4% relative AUC increase than our baseline without the additional supervision.

Acknowledgments

All experiments were performed on `beaker.org`. Computations on `beaker.org` were supported in part by credits from Google Cloud.

References

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *EMNLP*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. In *EMNLP*.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant supervision for relation extraction with matrix completion. In *ACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *EMNLP*.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *EMNLP*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Yi Yao Huang and William Yang Wang. 2017. Deep residual learning for weakly-supervised relation extraction. In *EMNLP*.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *ACL*.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. 2016. Relation extraction with multi-instance multi-label convolutional neural networks. In *COLING*.

Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S. Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *EMNLP*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H. Lin, Xiao Ling, and Daniel S. Weld. 2016. Effective crowd annotation for relation extraction. In *NAACL*.

Tian Yu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *EMNLP*.

Tianyi Liu, Xinsong Zhang, Wanhao Zhou, and Weijia Jia. 2018. Neural relation extraction via inner-sentence noise reduction and transfer learning. In *EMNLP*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *ACL*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *VS@HLT-NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. 2014. Infusion of labeled data into distant supervision for relation extraction. In *ACL*.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *NAACL-HLT*.

Sebastian Riedel, Limin Yao, and Andrew D McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML/PKDD*.

Sebastian Riedel, Limin Yao, Andrew D McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *AKBC@CIKM*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *ACL*.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *EMNLP*.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018. Label-free distant supervision for relation extraction via knowledge graph embedding. In *EMNLP*.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Hai Ye, Wen-Han Chao, Zhunchen Luo, and Zhoujun Li. 2017. Jointly extracting relations with class ties via effective deep ranking. In *ACL*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*.

Tweet Stance Detection Using an Attention based Neural Ensemble Model

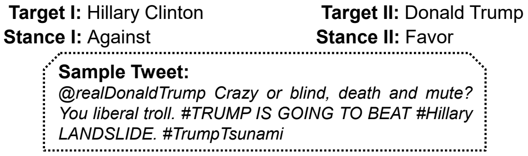
Umme Aymun Siddiqua, Abu Nowshed Chy, and Masaki Aono
Department of Computer Science and Engineering
Toyohashi University of Technology, Toyohashi, Aichi, Japan
`{aymun,nowshed}@kde.cs.tut.ac.jp, and aono@tut.jp`

Abstract

Stance detection in twitter aims at mining user stances expressed in a tweet towards a single or multiple target entities. To tackle this problem, most of the prior studies have been explored the traditional deep learning models, e.g., LSTM and GRU. However, in compared to these traditional approaches, recently proposed densely connected Bi-LSTM and nested LSTMs architectures effectively address the vanishing-gradient and overfitting problems as well as dealing with long-term dependencies. In this paper, we propose a neural ensemble model that adopts the strengths of these two LSTM variants to learn better long-term dependencies, where each module coupled with an attention mechanism that amplifies the contribution of important elements in the final representation. We also employ a multi-kernel convolution on top of them to extract the higher-level tweet representations. Results of extensive experiments on single and multi-target stance detection datasets show that our proposed method achieves substantial improvement over the current state-of-the-art deep learning based methods.

1 Introduction

Tweet stance detection is the task of automatically determining the stance of a tweet whether the tweet is in favor of, against, or none towards a target (Mohammad et al., 2017). We can consider it as a sub-domain of sentiment analysis. However, the goal of sentiment analysis is to classify the polarity of a tweet sentiment based on its contents, whereas identification of stance is dependent on the specific target. For example, Figure 1 depicts the stance of a tweet towards different targets.


Target I: Hillary Clinton
Stance I: Against

Target II: Donald Trump
Stance II: Favor

Sample Tweet:
@realDonaldTrump Crazy or blind, death and mute? You liberal troll. #TRUMP IS GOING TO BEAT #Hillary LANDSLIDE. #TrumpTsunami

Figure 1: Example of target-specific stance detection.

Stance detection in twitter poses unique challenges to the research community since tweets are short and informal user-generated text, which usually tend not to follow the grammatical rules. Moreover, tweets contain plenty of idiosyncratic abbreviations as well as other twitter specific syntaxes such as #hashtags and emoticons. To address the challenges of stance detection in twitter, Mohammad et al. (2016) presented a tweet stance detection task that focused on a single target in SemEval-2016. Top performing systems in this task proposed several deep learning based approaches by using CNN (Wei et al., 2016), RNN (Zarrella and Marsh, 2016), and so on.

Later, (Du et al., 2017) utilized the target-augmented embeddings in an attention based neural network, whereas (Zhou et al., 2017) proposed an attention mechanism at the semantic level in the bidirectional GRU-CNN structure to perform target-specific stance detection on tweets. More recently, (Dey et al., 2018) proposed a two-phase LSTM based model with attention and (Wei et al., 2018b) proposed an end-to-end neural memory model via target and tweet interactions.

By considering the dependency of related targets, Sobhani et al. (2017) introduced a multi-target stance detection (MTSD) task and proposed an attentive encoder-decoder network to capture the dependencies among stance labels regarding multiple targets. Later, (Wei et al., 2018a) proposed a dynamic memory-augmented network that utilized a shared external memory to capture and store multi-targets stance indicative clues.

However, most of the related work of tweet stance detection explored the traditional deep learning models in their methods. In this paper,

Proceedings of NAACL-HLT 2019, pages 1868–1873
Minneapolis, Minnesota, June 2 - June 7, 2019.

we propose a neural ensemble method that combines the attention based state-of-the-art densely connected Bi-LSTM and nested LSTMs models with the multi-kernel convolution in a unified architecture. Experimental results on both the single and multi-target benchmark stance detection datasets demonstrate the efficacy of our method over the state-of-the-art deep learning based methods discussed above.

The rest of the paper is structured as follows: In **Section 2**, we introduce our proposed stance detection framework. **Section 3** includes experiments and evaluations as well as the comparisons with the state-of-the-arts to show the effectiveness of our proposed method. Some concluded remarks and future directions of our work are described in **Section 4**.

2 Proposed Stance Detection Framework

In this section, we describe the details of our proposed neural ensemble model (PNEM) for twitter stance detection. Figure 2 depicts an overview of our proposed framework.

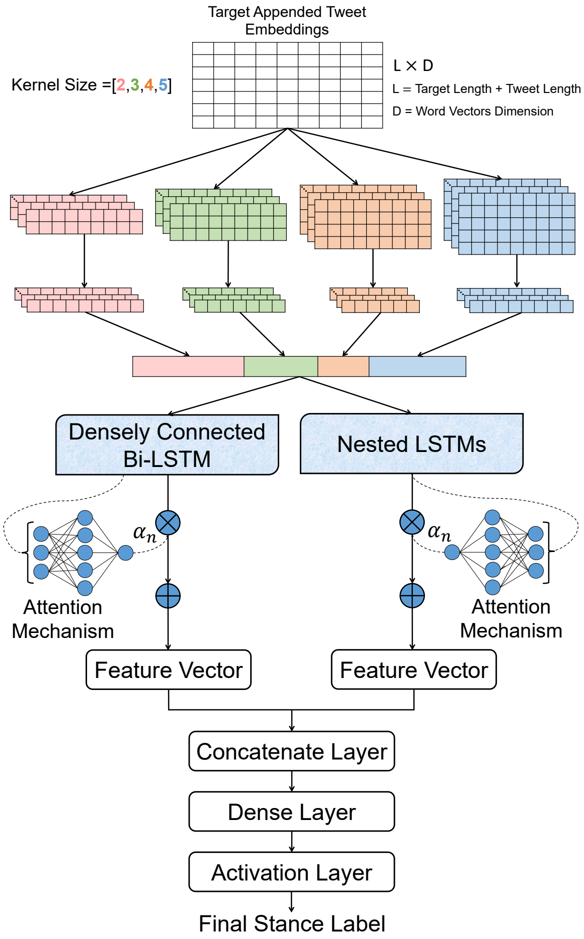


Figure 2: Proposed stance detection framework.

At first, we utilize the multi-kernel convolution filters to extract higher-level feature sequences from the target appended tweet embeddings. These feature sequences are fed into the attention based densely connected Bi-LSTM and nested LSTMs to learn long-term dependencies. Final representations of these modules are concatenated and pass to the stance prediction module to determine the stance label. Next, we describe each component elaborately.

Embedding Layer: Prior works already established the significance of target information for stance detection. To integrate the target information, we generate a unified word vector matrix by concatenating the vector representations of the target and tweet. The dimensionality of the matrix will be $L \times D$, where length L is the sum of the target length and tweet length, and D denotes the word-vector dimension. We utilize a pre-trained word embedding model for obtaining the vector representation of words.

Multi-Kernel Convolution: In our multi-kernel convolution, we adopt the idea proposed by (Kim, 2014) to extract the higher-level features. The input of this module is the target appended tweet matrix generated in the embedding layer. We then perform the convolution on it by using a filter. We apply multiple convolutions based on four different kernel sizes, i.e., the size of the convolution filters: 2, 3, 4, and 5. After performing convolutions, each filter generates the corresponding feature maps and a max pooling function is then applied to generate a univariate feature vector. Finally, the feature vectors generated from each kernel are concatenated to form a single high-level feature vector.

Densely Connected Bi-LSTM: With the emerging trend of deep learning, LSTM based models are the most popular for sequential tasks. Recently, the densely connected structure of LSTM models gets attention among the researchers (Li et al., 2018; Wu et al., 2017) that enable the effective connection from lower to upper layers features without any loss of information on lower-layer features thus alleviate the vanishing-gradient and overfitting problems effectively.

In our proposed framework, we utilize the densely connected Bi-LSTM (Ding et al., 2018) (DC-Bi-LSTM) model. A DC-Bi-LSTM model consists of multiple Bi-LSTM layers, where representation of each layer is estimated by concatenating its hidden states and all the preceding layers' hidden states. Hence, for the first Bi-LSTM layer, the input is a higher-level features sequences generated from multi-kernel convolution (MKC) and the output is $\{h^1 = h_1^1, h_2^1, ..., h_s^1\}$. For the second Bi-LSTM layer, the input is the concatenation of higher-level feature sequences from MKC and the output from first Bi-LSTM layer to generate the corresponding output. Rest of the layers are processed accordingly. We can define the above process as follows:

$$
\begin{aligned}
&h_t^l = [\overrightarrow{h_t^l}; \overleftarrow{h_t^l}], h_t^0 = \text{MKC feature sequence},\\
&\overrightarrow{h_t^l} = \text{lstm}(\overrightarrow{h_{t-1}^l}, M_t^{l-1}),\\
&\overleftarrow{h_t^l} = \text{lstm}(\overleftarrow{h_{t+1}^l}, M_t^{l-1}),\\
&M_t^{l-1} = [h_t^0; h_t^1; ...; h_t^{l-1}].
\end{aligned}
$$

Therefore, from a L layer DC-Bi-LSTM model, the output is $\{h^L = h_1^L, h_2^L, ..., h_s^L\}$.

Nested LSTMs: The nested LSTMs (NLSTMs) architecture (Moniz and Krueger, 2017) creates a temporal hierarchy of memories that achieved significant improvement over the single-layer or stacked LSTM architectures to learn longer-term dependencies.

In NLSTMs, the LSTM memory cells have access to their inner memory, where they can selectively read and write relevant long-term information. While the value of the outer memory cell in the LSTM is estimated as $c_t^{outer} = f_t \odot c_{t-1} + i_t \odot g_t$, memory cells of the NLSTMs use the concatenation $(f_t \odot c_{t-1}, i_t \odot g_t)$ as input to an inner LSTM (or NLSTM) memory cell, and set $c_t^{outer} = h_t^{inner}$. Therefore, in compared to the LSTM and stacked LSTM, the inner memories of NLSTMs operate on longer time-scales and effectively capture the context information from the input texts.

Feed-Forward Attention: Recently, the attention mechanism has been introduced in the neural network models for effectively modeling the long-term dependencies by enabling the model to learn what to attend based on the input text (Vaswani et al., 2017). In order to amplify the contribution of important elements in the final representation of both the DC-Bi-LSTM and NLSTMs module, we employ a feed-forward attention mechanism (Raffel and Ellis, 2015) to aggregate all the hidden states according to their relative importance weight.

Stance Prediction and Model Training: We concatenate the final tweet representation from the attention based DC-Bi-LSTM and NLSTMs module and pass it to a fully connected softmax layer for stance detection. We consider cross-entropy as the loss function and train the model by minimizing the error, which is defined as:

$$
E(x^{(i)}, y^{(i)}) = \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log(y_j^{\sim(i)})
$$

where $x^{(i)}$ is the training sample with its true label $y^{(i)}$. $y_j^{\sim(i)}$ is the estimated probability in $[0, 1]$ for each label j. $1\{condition\}$ is an indicator which is 1 if true and 0 otherwise. We use the stochastic gradient descent (SGD) to learn the model parameter and adopt the Adam optimizer (Kingma and Ba, 2014).

3 Experiments and Evaluations

3.1 Model Configuration

In the following, we describe the set of parameters that we have used in our proposed neural ensemble model (PNEM) during experiments. We used the 300-dimensional fastText embedding model pre-trained on Wikipedia with skip-gram (Bojanowski et al., 2017) to initialize the word embeddings in the embedding layer. For the multi-kernel convolution, we employed 4 kernel sizes (2,3,4,5), and the number of filters was set to 600. In our model, DC-Bi-LSTM module contains 5 layers and NLSTMs module contains 2 layers. We trained all models for max 45 epochs with a batch size of 32 and an initial learning rate 0.001 by Adam optimizer. $L2$ regularization with a factor of 0.01 was applied to the weights in the softmax layer. In this paper, we reported the results based on these settings. Unless otherwise stated, default settings were used for the other parameters.

To preprocess the data, we removed the stop words based on NLTK's standard stoplist, special characters removal, and performed hashtag segmentation according to (Baziotis et al., 2017).

3.2 Single Target Stance Detection

Dataset and Setup: To validate the effectiveness of our proposed method for the single target stance detection, we made use of a widely popular benchmark twitter dataset used in the SemEval-2016 Task 6-A (Mohammad et al., 2016). The training set consists of 2914 tweets and the test set consists of 1249 tweets relevant to 5 targets. Each tweet was annotated as *Favor*, *Against* or *None* towards the specific target.

Following the SemEval-2016 Task 6-A benchmark, we employed the macro-average of F1-score for the *Favor* and *Against* stance classes as the evaluation measure. To estimate it, we used the evaluation script provided by the organizer.

Results and Analysis: We divided the whole dataset across targets and trained the model accordingly. We used 5% of the training samples as the validation set. The summarized experimental results of our proposed neural ensemble model (PNEM) on single target stance detection are presented in Table 1.

At first, we report the results based on the baseline, which is the combination of CNN and LSTM (Zhou et al., 2015) and obtained competitive performances on several text classification tasks. Next, we report the results of our proposed PNEM model. It showed that our PNEM method outperformed the baseline by a large margin. In order to estimate the effect of each component of our model, we performed the component ablation study on our proposed model. In this regard, we removed one component each time and repeated the experiment. From the results, it can be observed that when removing target embedding, multi-kernel convolution (MKC), attention mechanism (ATT), NLSTMs with corresponding attention (NLSTMs+ATT), and DC-Bi-LSTM with corresponding attention (DC-Bi-LSTM+ATT), the results decreased by 0.99%, 4.98%, 1.68%, 4.14%, and 1.11%, respectively. Thus deduced the importance of each of the component in our model.

Method	F_{favor}	$F_{against}$	F_{avg}
CNN+LSTM	59.36	74.93	67.15
PNEM	**66.56**	**77.66**	**72.11**
– Target Embedding	64.87	77.36	71.12
– MKC	60.49	73.78	67.13
– ATT	63.94	76.92	70.43
– (NLSTMs+ATT)	61.51	74.43	67.97
– (DC-Bi-LSTM+ATT)	64.47	77.52	71.00

Table 1: Comparative performance with different experimental settings for single target stance detection. The best results are highlighted in boldface.

Moreover, the comparative performance of our proposed model, PNEM with the state-of-the-art methods are presented in Table 2. It showed that our model gained 3.13% and 4.29% improvement while compared with the SemEval-2016 baseline (SVM-ngrams) and the best performing system MITRE, respectively. Furthermore, in comparison with the related deep learning based methods, PNEM achieved at least 1.07% and at best 3.44% improvement. Overall, our proposed PNEM method greatly surpassed the previous works. The results show that attentive neural ensemble model could benefit stance detection.

3.3 Multi-Target Stance Detection

Dataset and Setup: In order to assess our method for the multi-target stance detection, we made use of a benchmark twitter dataset (Sobhani et al., 2017), where each tweet is annotated with two stance labels towards two targets of a pair. The tweets in this dataset are related to the 2016 US election. Overall, the dataset contains 4455 tweets for the three target-pairs including Hillary Clinton - Bernie Sanders, Hillary Clinton - Donald Trump, and Ted Cruz - Donald Trump. The train, development, and test set contains 3119, 446, and 890 tweets, respectively.

As the evaluation measure, we used a similar kind of approach reported in the dataset paper (Sobhani et al., 2017). Following this, the F_{avg} of each target is estimated according to the SemEval-2016 Task-A benchmark (Mohammad et al., 2016). Then, the F_{avg} of the two targets of a target-pair is averaged to estimate the average score of a target-pair. Finally, the average scores of all the target-pairs are averaged to estimate the overall score.

Results and Analysis: We divided the whole dataset based on each target-pairs. The dataset is then separated into two parts for each target in the target-pair. We trained and evaluated the model for each target in the target-pair and combined the results to estimate the overall performance.

We used a similar kind of baseline (CNN+LSTM) for comparison that we used in the single target stance detection as well as compared with the state-of-the-art deep learning based methods. The comparative results of our proposed PNEM model on multi-target stance dataset are presented in Table 3. It showed that our method surpassed the baseline by a large margin and gained 3.91% and 1.99% improvement over the state-of-the-art methods Seq2Seq and DMAN, respectively.

Method	Overall			Target				
	F_{favor}	$F_{against}$	F_{avg}	Atheism	Climate Change	Feminist Movement	Hillary Clinton	Legal. of Abortion
PNEM	**66.56**	**77.66**	**72.11**	67.73	44.27	**66.76**	60.28	64.23
SemEval-2016 Baseline and Related Deep Learning based Methods								
SVM-ngrams	62.98	74.98	68.98	65.19	42.35	57.46	58.63	66.42
MITRE	59.32	76.33	67.82	61.47	41.63	62.09	57.67	57.28
n-grams+embeddings	-	-	70.30	**68.30**	43.80	58.40	57.80	**66.90**
TGMN-CR	65.52	76.55	71.04	64.60	43.02	59.35	**66.21**	66.21
T-PAN	-	-	68.84	61.19	**66.27**	58.45	57.48	60.21
AS-biGRU-CNN	-	-	69.42	66.76	43.40	58.83	57.12	65.45
TAN	-	-	68.79	59.33	53.59	55.77	65.38	63.72

Table 2: (Single target) Comparative performance of our model against the SemEval-2016 official baseline (SVM-ngrams) (Mohammad et al., 2016) and related deep learning based methods including MITRE (SemEval-2016 best performing system) (Zarrella and Marsh, 2016), n-grams+embeddings (Mohammad et al., 2017), TGMN-CR (Wei et al., 2018b), T-PAN (Dey et al., 2018), AS-biGRU-CNN (Zhou et al., 2017), and TAN (Du et al., 2017). The best results are highlighted in boldface.

Method	Overall	Target-Pair		
		Clinton Sanders	Clinton Trump	Cruz Trump
CNN+LSTM	46.03	43.02	47.35	47.71
Seq2Seq	54.81	54.72	56.60	53.12
DMAN	56.73	56.25	**60.30**	53.64
PNEM	**58.72**	**57.74**	60.05	**58.36**

Table 3: (Multi-target) Comparative performance of our model against the baseline (CNN+LSTM) and state-of-the-art deep learning based methods including Seq2Seq (Sobhani et al., 2017) and DMAN (Wei et al., 2018a). The best results are highlighted in boldface.

4 Conclusion

In this paper, we proposed an attention based neural ensemble model for the target-specific tweet stance detection. The main contribution of our unified model is to learn the contextual information effectively which in turns improved the stance detection performance and outperformed the state-of-the-art deep learning based methods for both the single and multi-target stance detection benchmark datasets.

In the future, we have a plan to leverage external knowledge and generalize our model for target-independent stance detection in the same domain.

Acknowledgments

The part of this research is supported by MEXT KAKENHI, Grant-in-Aid for Scientific Research (B), Grant Number 17H01746 and by grants from the KDDI Foundation.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at SemEval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval)*, pages 747–754, Vancouver, Canada. ACL.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. 2018. Topical stance detection for twitter: A two-phase lstm model using attention. In *Proceedings of the 40th European Conference on Information Retrieval (ECIR)*, pages 529–536. Springer.

Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. 2018. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3988–3994.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. ACL.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Dexu Li, Yimin Chen, Mingke Gao, Siyu Jiang, and Chen Huang. 2018. Multimodal gesture recognition using densely connected convolution and blstm. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR)*, pages 3365–3370. IEEE.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*, pages 31–41.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Joel Ruben Antony Moniz and David Krueger. 2017. Nested lstms. In *Proceedings of the 9th Asian Conference on Machine Learning (ACML)*, pages 530–544. Springer.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 2, Short Papers*, pages 551–557. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Penghui Wei, Junjie Lin, and Wenji Mao. 2018a. Multi-target stance detection via a dynamic memory-augmented network. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1229–1232. ACM.

Penghui Wei, Wenji Mao, and Daniel Zeng. 2018b. A target-guided neural memory model for stance detection in twitter. In *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. Pkudblab at SemEval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval), NAACL-HLT*, pages 384–388.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, Sixing Wu, and Zhigang Yuan. 2017. Thu_ngn at IJCNLP-2017 task 2: Dimensional sentiment analysis for Chinese phrases with deep lstm. *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP), Shared Tasks*, pages 47–52.

Guido Zarrella and Amy Marsh. 2016. Mitre at SemEval-2016 task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 458–463.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A c-lstm neural network for text classification. *CoRR*, abs/1511.08630.

Yiwei Zhou, Alexandra I Cristea, and Lei Shi. 2017. Connecting targets to tweets: Semantic attention-based model for target-specific stance detection. In *Proceedings of the 18th International Conference on Web Information Systems Engineering (WISE)*, pages 18–32. Springer.

Word Embedding-Based Automatic MT Evaluation Metric using Word Position Information

Hiroshi Echizen'ya
Department of Life
Science and Technology
Hokkai-Gakuen University, Japan
echi@lst.hokkai-s-u.ac.jp

Kenji Araki
Graduate School of Information
Science and Technology
Hokkaido University, Japan
araki@ist.hokudai.ac.jp

Eduard Hovy
Language Technologies Institute
Carnegie Mellon University, USA
hovy@cmu.edu

Abstract

We propose a new automatic evaluation metric for machine translation. Our proposed metric is obtained by adjusting the Earth Mover's Distance (EMD) to the evaluation task. The EMD measure is used to obtain the distance between two probability distributions consisting of some signatures having a feature and a weight. We use word embeddings, sentence-level $tf \cdot idf$, and cosine similarity between two word embeddings, respectively, as the features, weight, and the distance between two features. Results show that our proposed metric can evaluate machine translation based on word meaning. Moreover, for distance, cosine similarity and word position information are used to address word-order differences. We designate this metric as **W**ord **E**mbedding-based automatic MT evaluation using **W**ord **P**osition **I**nformation (WE_WPI). A meta-evaluation using WMT16 metrics shared task set indicates that our WE_WPI achieves the highest correlation with human judgment among several representative metrics.

1 Introduction

Recent advances in neural machine translation (NMT) (Sutskever et al., 2014; Luong et al., 2015) are remarkable. Results based on human evaluation have demonstrated that NMT outperforms statistical machine translations significantly (Chiang, 2005; Tufiş and Ceauşu, 2009). The NMT achieved especially high performance in terms of fluency. However, it tends to generate more omission errors than statistical machine translations generate. Unfortunately, it is difficult for automatic evaluation metrics to evaluate outputs with omission errors because those errors are not included as non-match words between the translation and reference. For such cases, the word embedding-based automatic MT evaluation metric, which is based on word position information, is effective.

Various automatic evaluation metrics have been proposed for machine translation, but none is sufficient for NMT. Actually, BLEU (Papineni et al., 2002) is the representative metric based on n-gram matching. Unfortunately, because it is a surface-level metric, it is difficult to address word meaning during evaluation for MT outputs. The word-embedding-based distance measure for document (Kusner et al., 2016) and the word-alignment-based automatic evaluation metric using word embedding (Matsuo et al., 2017) are effective to address word meanings. Nevertheless, they can only ineffectively accommodate word order differences between the translation and reference.

Given those circumstances, a new metric with word embedding-based automatic MT evaluation metric using word position information is proposed in which the evaluation score is obtained by adjusting the Earth Mover's Distance (EMD) (Rubner et al., 1998, 2000) to the evaluation task. The EMD measure represents the distance between two probability distributions. Moreover, the EMD distance is obtained based on a signature consisting of the feature and the weight, and the distance between two features. The feature, weight, and distance must therefore be defined to adjust EMD to the evaluation task.

Proceedings of NAACL-HLT 2019, pages 1874–1883
Minneapolis, Minnesota, June 2 - June 7, 2019.

In our proposed metric, the word embeddings and the sentence-level $tf \cdot idf$ respectively denote the feature and the weight. Consequently, our proposed metric can produce an evaluation based on the word meaning. Moreover, our proposed metric uses word position information in the distance between two word embeddings. The distance is obtained using cosine similarity and the difference of word position between the translation and reference. Results demonstrate that our proposed metric can evaluate translations also considering word order differences. We designate this new metric as **W**ord **E**mbedding-based automatic MT evaluation using **W**ord **P**osition **I**nformation (WE_WPI).

The experimentally obtained results based on the WMT16 metrics shared task (Bojar et al., 2016) demonstrated that our WE_WPI achieves the highest correlation with human judgment among several metrics: BLEU, METEOR (Banerjee and Lavie, 2005), IMPACT (Echizen-ya and Araki, 2007), and RIBES (Isozaki et al., 2010). Moreover, the correlation of WE_WPI is better than that of WE_WPI without word position information (WE). Results therefore confirmed the effectiveness of WE_WPI using word position information.

2 Related Work

Kusner et al. (2016) proposed the Word Mover's Distance (WMD) as a distance measure using word embedding and word alignment. This measure obtains the distance between two documents adjusting EMD to a document. However, it cannot accommodate differences of word order between the translation and reference. Matsuo et al. (2017) also proposed a word-alignment-based automatic evaluation metric using word embeddings for segment-level evaluation. As described in that paper, Maximum Alignment Similarity (MAS) was found to have higher correlation with human evaluation than BLEU for European-to-English, which has similar grammar structures. For Japanese-to-English, which has different grammar structures, Average Alignment Similarity (AAS) showed better correlation with human evaluation than other metrics. However, neither MAS nor AAS uses word position information. Therefore, neither can sufficiently accommodate word order differences. Actually, WE_WPI uses not only the word alignment but also word position information.

One system, DREEM (Chen and Guo, 2015), learns distributed word representations from a neural network model and from distributed sentence representations computed with a recursive autoencoder. Moreover, it uses a penalty based on translation and reference lengths. By contrast, the WE_WPI system specifically examines the difference between the word positions of the translation and reference, not the difference of lengths between the translation and reference. Therefore, it can sufficiently accommodate word order differences. Moreover, it can evaluate the translation efficiently using word embeddings of target languages without requiring large amounts of data or learning time. Our WE_WPI requires no learning of bilingual knowledge or a relation between translation and reference. It needs only a model of word embeddings in advance to apply EMD to the automatic MT evaluation task.

In a non-trained evaluation metric, MEANT 2.0 (Lo, 2017; Bojar et al., 2017) uses a distributional word vector model to evaluate lexical semantic similarity and shallow semantic parses to evaluate structural semantic similarity between the translation and reference. It is a new version of MEANT (Lo and Wu, 2011), which is a non-ensemble and untrained metric. Moreover, MEANT 2.0 - nosrl is a subversion of MEANT 2.0 to evaluate the translation for any output language by removing the dependence on semantic parsers for semantic role labeling (SRL). In that case, phrasal similarity is calculated using n-gram lexical similarities. However, MEANT 2.0 series do not specifically examine the position of each word in the translation and reference. Results show that it is difficult to deal sufficiently with language pairs for which the grammar differs. In WE_WPI, the evaluation score is calculated using the relative difference between the positions of each word in the translation and reference. Therefore, WE_WPI can evaluate translations dealing with word order in languages pairs for which the grammar differs.

3 Word Embedding-Based Automatic Evaluation Metric with Word Position Information (WE_WPI)

3.1 The Earth Mover's Distance (EMD)

3.1.1 Definitions

As described herein, we propose WE_WPI as the automatic MT evaluation metric obtained by adjusting the **E**arth **M**over's **D**istance (EMD) to the

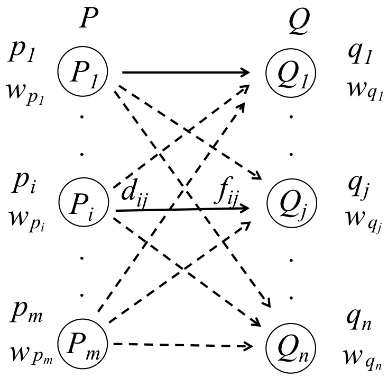


Figure 1: Outline of EMD.

automatic MT evaluation task. First, we describe EMD. Figure 1 depicts an outline of EMD.

In Figure 1, two probability distributions are presented respectively as P and Q. The P and Q consist of some P_i and Q_j, which are the respective signatures. Each signature consists of a feature (*i.e.*, p_i in P_i and q_j in Q_j) and a weight (*i.e.*, w_{p_i} in P_i and w_{q_j} in Q_j). Therefore, two probability distributions P and Q are defined respectively as $P = \{(p_1, w_{p_1})...(p_m, w_{p_m})\}$ and $Q = \{(q_1, w_{q_1})...(q_n, w_{q_n})\}$. Moreover, d_{ij} represents the distance between two features p_i and q_j.

The goal of EMD is to obtain total flow $F = [f_{ij}]$ that minimizes the overall cost from the perspective of a transportation problem. In that case, the overall cost is defined as Eq. (1).

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{1}$$

Moreover, four constraints are defined for f_{ij}, which is the transportation amount in the transportation problem, to find minimum F as the following Eqs. (2)–(5):

$$f_{ij} \geq 0 \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \tag{2}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \quad 1 \leq i \leq m \tag{3}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j} \quad 1 \leq j \leq n \tag{4}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right) \tag{5}$$

Constraint (2) shows that each amount of weight f_{ij} is transported only in the direction from signature P_i to signature Q_j to be nonnegative. In Constraint (3), the amount of weight which is supplied from P_i (*i.e.*, $\sum_{j=1}^{n} f_{ij}$) does not exceed w_{p_i}, which is the weight of P_i. Moreover, in Constraint (4), the amount of weight which Q_j receives (*i.e.*, $\sum_{i=1}^{m} f_{ij}$) does not exceed w_{q_j}, which is the weight of Q_j. Finally, the total amount of weight is equal to the weight of the lighter distribution in Constraint (5). In Eqs. (1)–(5), m shows the number of signatures in P; n shows the number of signatures in Q.

The EMD is defined as shown below.

$$EMD(P, Q) = \frac{min(WORK(P, Q, F))}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{6}$$

In Eq. (6), the $min(WORK(P, Q, F))$ is normalized by the minimum amount of work of Eq. (5).

3.1.2 Computing EMD

P_1		P_2		P_3		P_4	
p_1	w_{p_1}	p_2	w_{p_2}	p_3	w_{p_3}	p_4	w_{p_4}
(1,5)	0.6	(5,5)	0.6	(1,1)	0.6	(5,1)	0.6

Table 1: Examples of signatures of P.

Q_1		Q_2		Q_3	
q_1	w_{q_1}	q_2	w_{q_2}	q_3	w_{q_3}
(2,3)	0.8	(4,3)	0.8	(3,2)	0.8

Table 2: Examples of signatures of Q.

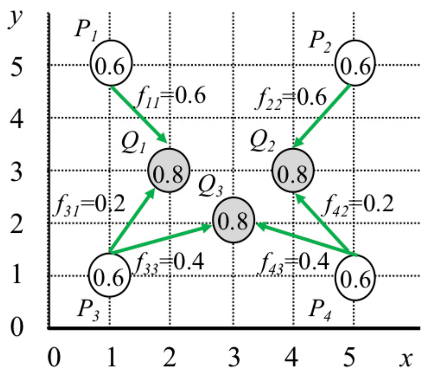


Figure 2: Example of EMD calculation.

We describe the computation of EMD using two probability distributions P and Q in two-dimensional surface. Tables 1 and 2 respectively present the examples of P and Q signatures.

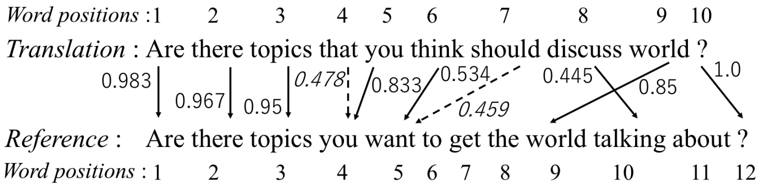


Figure 3: Example of word alignment by WE_WPI.

In Tables 1 and 2, all features p_i and q_j correspond to the coordinate (x, y) of two-dimensional surface.

Figure 2 depicts an example of an EMD calculation based on the signatures in Tables 1 and 2. In Figure 2, the green arrow indicates the amount of weight f_{ij}. All f_{ij} are transported only in the direction from P_i to Q_j according to Constraint (2). In each signature P_i, $\sum_{j=1}^{n} f_{ij}$ does not exceed w_{p_i} by Constraint (3). For example, in P_3, $\sum_{j=1}^{3} f_{3j}$ is 0.6 (=0.2+0.0+0.4). It does not exceed 0.6, which is the weight of P_3. Moreover, in each signature Q_j, $\sum_{i=1}^{m} f_{ij}$ does not exceed w_{q_j} according to Constraint (4). For example, in Q_1, $\sum_{i=1}^{4} f_{i1}$ is 0.8 (=0.6+0.0+0.2+0.0). It does not exceed 0.8, which corresponds to the weight of Q_1. The total amount of weight by $\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}$ is 2.4. It is equal to 2.4 by $\sum_{i=1}^{m} w_{p_i}$ or 2.4 by $\sum_{j=1}^{n} w_{q_j}$. Therefore, this example of Figure 2 conforms to Constraint (5).

Moreover, the distance between two features is necessary to obtain EMD. When the Euclidean distance is used as the calculation of distance in this example, 2.236 (=$\sqrt{1^2 + 2^2}$) is obtained as d_{11}, d_{22}, d_{31}, d_{33}, d_{42}, and d_{43}, and other distances are 3.606 (=$\sqrt{2^2 + 3^2}$) in Figure 2. As a result, 5.366 (=2.236 × (0.6+0.6+0.2+0.4+0.2+0.4)) is obtained as the value of EMD by two probability distributions P and Q in Tables 1 and 2.

We obtain WE_WPI adjusting EMD to the automatic MT evaluation task. Details of application of EMD to WE_WPI are presented in 3.2.2.

3.2 New Automatic MT Evaluation Metric: WE_WPI

3.2.1 Word Alignment using Position Information

For the application of EMD to automatic MT evaluation, we use word alignment results. Word alignment is done using cosine similarity based on word embeddings and the relative difference between the word positions in the translation and reference. In that case, WE_WPI obtains $align\_score$ using Eqs. (7) and (8) presented below.

$$\begin{aligned} &align\_score \\ &= cos\_sim(t_i, r_j) \times (1.0 - pos\_inf(T_i, R_j)) \quad (7) \end{aligned}$$

$$pos\_inf(T_i, R_j) = \left| \frac{pos(T_i)}{m} - \frac{pos(R_j)}{n} \right| \quad (8)$$

In Eq. (7), t_i and r_j respectively represent the word embeddings of word T_i in the translation and word R_j in the reference. The $cos\_sim(t_i, r_j)$ denotes the cosine similarity between t_i and r_j. Moreover, $pos\_inf(T_i, R_j)$ represents the relative difference between the position of word T_i in the translation and the position of word R_j in the reference. It is defined as Eq. (8). In Eq. (8), $pos(T_i)$ and $pos(R_j)$ respectively denote the positions of word T_i in the translation and word R_j in the reference. Actually, m and n respectively denote the word numbers in the translation and reference. The $pos\_inf(T_i, R_j)$ becomes larger as the relative difference between $pos(T_i)$ and $pos(R_j)$ becomes larger . Therefore, $(1.0 - pos\_inf(T_i, R_j))$ is used as the negative weight for $cos\_sim(t_i, r_j)$. The ranges of $cos\_sim(t_i, r_j)$ and $pos\_inf(T_i, R_j)$ are both 0.0-1.0. Figure 3 depicts an example of word alignment using Eqs. (7) and (8).

The WE_WPI calculates $align\_score$ between a word in the translation and all words in reference. Based on those results, the word with the highest $align\_score$ in the reference is selected as the corresponding word to the word in the translation. In Figure 3, the $align\_score$ between "that" in the translation and "you" in the reference is the highest (*i.e.*, 0.478) among the $align\_score$ between "that" in the translation and all words in the reference. However, it is lower than the $align\_score$ 0.833 between "you" in the translation and "you" in the reference. Therefore, the word which corresponds to "that" in the translation cannot be obtained in the reference. Similarly, the word which

		reference											
		Are	there	topics	you	want	to	get	the	world	talking	about	?
trans-lation	Are	**0.017**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	there	1.0	**0.033**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	topics	1.0	1.0	**0.049**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	that	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	you	1.0	1.0	1.0	**0.154**	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	think	1.0	1.0	1.0	1.0	**0.456**	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	should	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	discuss	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.555**	1.0	1.0
	world	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.139**	1.0	1.0	1.0
	?	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**0.0**

Table 3: Distance matrix incorporating translation and reference.

corresponds to "should" in the translation cannot be obtained in the reference.

In contrast, "discuss" in the translation corresponds to "talking" in the reference using $pos\_inf(T_i, R_j)$ of Eq. (8) although "discuss" in the translation corresponds to "topics" in the reference when $(1.0 - pos\_inf(T_i, R_j))$ is not used in Eq. (7) (*i.e.*, $align\_score = cos\_sim(t_i, r_j)$). The 0.477, which is the cos_sim between "discuss" in the translation and "topics" in the reference, is greater than 0.460, which is the cos_sim between "discuss" in the translation and "talking" in the reference. Here, $pos\_inf(T_i, R_j)$ between "discuss" in the translation and "talking" in the reference is 0.033 ($\left|\frac{8}{10} - \frac{10}{12}\right|$). That between "discuss" in the translation and "topics" in the reference is 0.550 ($\left|\frac{8}{10} - \frac{3}{12}\right|$). Consequently, the $align\_score$ of "discuss" in the translation and "talking" in the reference is 0.445 ($0.460 \times (1.0 - 0.033)$). That of "discuss" in the translation and "topics" in the reference is 0.215 ($0.477 \times (1.0 - 0.550)$) using Eq. (7). The WE_WPI can select "talking" in the reference as the corresponding word for "discuss" in the translation using $pos\_inf(T_i, R_j)$. The use of $pos\_inf(T_i, R_j)$ is effective for the correct word alignment.

3.2.2 Adjustment of EMD to the Automatic MT Evaluation Metric

We obtain WE_WPI as new automatic MT evaluation metrics by adjusting EMD to the automatic MT evaluation task. In WE_WPI, the variables P and Q in Figure 1 respectively correspond to a translation T and reference R. Moreover, the features (*i.e.*, p_i and q_j in Figure 1), the weight (*i.e.*, w_{p_i} and w_{q_j} in Figure 1), and distance (*i.e.*, d_{ij} in Figure 1) are required as parameters to adjust EMD to the automatic MT evaluation task. As described herein, we use the word embeddings as features and the sentence-level $tf \cdot idf$ as the weight. The weight definition is presented in Eq. (9) below.

$$w = tf \times \left(\log \frac{N}{df} + 1.0\right) \tag{9}$$

In Eq. (9), tf denotes the appearance frequency of a word in a translation or reference. In addition, df represents the number of sentences in which the word appears in all translations or references. In addition, N is the total number of translations or references. Actually, WE_WPI distinguishes the function word and the content word using Eq. (9). Furthermore, w_{t_i} of the word in the translation and w_{r_i} of the word in the reference by Eq. (9) are normalized respectively using the following Eqs. (10) and (11).

$$\tilde{w}_{t_i} = \frac{w_{t_i}}{\sum_{i=1}^{m} w_{t_i}} \tag{10}$$

$$\tilde{w}_{r_j} = \frac{w_{r_j}}{\sum_{j=1}^{n} w_{r_j}} \tag{11}$$

The dependence of w in Eq. (9) by difference of dataset can be kept to the minimum by normalizing Eqs. (10) and (11). Moreover, we define distance d_{ij}, which is ascertained from the result of the word alignment described in 3.2.1. The d_{ij} is obtained using the following Eq. (12):

$$d_{ij} = \begin{cases} 1.0 - cos\_sim(t_i, r_j) \\ \qquad\qquad \times \mathrm{e}^{-pos\_inf(T_i, R_j)} \\ \qquad \text{if } T_i \text{ corresponds to } R_j \\ 1.0 \quad \text{if } T_i \text{ does not correspond to } R_j \end{cases} \tag{12}$$

In Eq. (12), $1.0 - cos\_sim(t_i, r_j) \times \mathrm{e}^{-pos\_inf(T_i, R_j)}$ is used as d_{ij} when word T_i in the translation corresponds to word R_j in the reference by the word alignment result. The $pos\_inf(T_i, R_j)$ is obtained by Eq. (8). Here, t_i and r_j respectively correspond to the word embeddings of the words in the translation and reference. The $\mathrm{e}^{-pos\_inf(T_i, R_j)}$ represents the penalty

	cs-en		de-en		fi-en		ro-en		ru-en		tr-en	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
Systems	6	6	10	10	9	9	7	7	10	10	8	8
mtevalBLEU	.992	.989	.905	.808	.858	.864	.899	.840	.962	.837	.899	.895
METEOR	.995	**.991**	.935	.887	.952	.963	.934	.909	.987	.930	.965	.980
IMPACT	**.997**	.990	.925	.841	.908	.915	.903	.819	.962	.840	.952	.959
RIBES	.995	.990	.948	.891	.894	.901	**.954**	.794	.972	.864	.850	.868
MEANT 2.0 (Lo, 2017)	.989	.990	.947	**.950**	.953	.966	.940	**.946**	.990	**.959**	**.980**	.990
MEANT 2.0 - nosrl	.985	.988	.928	.942	**.969**	**.979**	.917	.930	.984	.958	.978	.987
WE	.986	.976	.918	.903	.954	.963	.885	.884	.989	.938	.976	.991
WE_WPI	.991	.980	**.958**	.927	.955	.957	.919	.877	**.991**	.926	.977	**.993**

Table 4: Absolute Pearson correlation of to-English system-level metric with human assessment variants: RR, standard WMT relative ranking; DA, direct assessment of translation adequacy.

	en-cs		en-de		en-fi		en-ro		en-ru		en-tr	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
Systems	10		15		13		12		12	12	8	
mtevalBLEU	.968	-	.752	-	.868	-	.897	-	.835	.838	.745	-
METEOR	.960	-	.631	-	.939	-	.873	-	.868	.879	.800	-
IMPACT	**.978**	-	.719	-	.924	-	.911	-	.874	.879	.844	-
RIBES	.968	-	.742	-	**.949**	-	.910	-	.895	.904	.883	-
MEANT 2.0 (Lo, 2017)	-	-	.540	-	-	-	-	-	-	-	-	-
MEANT 2.0 - norsrl	.967	-	.541	-	.902	-	.868	-	**.925**	**.946**	.933	-
WE	.962	-	.609	-	.925	-	.878	-	.899	.910	.930	-
WE_WPI	.967	-	**.780**	-	.931	-	**.917**	-	.914	.923	**.944**	-

Table 5: Absolute Pearson correlation of out-of-English system-level metric with human assessment variants: RR, standard WMT relative ranking; DA, direct assessment of translation adequacy.

to $cos\_sim(t_i, r_j)$ because it becomes smaller as $pos\_inf(T_i, R_j)$ becomes larger. As a result, d_{ij} becomes large when the relative difference between the position of word T_i in the translation and the position of word R_j in the reference (*i.e.*, $pos\_inf(T_i, R_j)$) is large. The d_{ij} by Eq. (12) is 1.0 when word T_i does not correspond to word R_j. Finally, the range of d_{ij} becomes 0.0-1.0.

Moreover, the WE_WPI generates the distance matrix using d_{ij} in Eq. (12). Table 3 presents the distance matrix between the translation "Are there topics that you think should discuss world?" and the reference "Are there topics you want to get the world talking about?" in Figure 3. In Table 3, the bold typeface represents the distance between the two aligned words. The distance matrix using Eq. (12) is effective because it is not influenced by the words which are not aligned between the translation and reference.

The WE_WPI obtains the evaluation score by word embedding, sentence-level $tf \cdot idf$, and the distance matrix based on Eq. (12). The evaluation score of WE_WPI is obtained as Eq. (13).

$$WE\_WPI(T, R) = 1.0 - \frac{min(WORK(T, R, F))}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{13}$$

In that equation, the range of $\frac{min(WORK(T,R,F))}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}$ becomes 0.0-1.0 using the weights normalized by Eqs. (10) and (11). Near 0.0, the distance between T and R is small. However, in the automatic MT evaluation metrics, the score is close to 1.0 when the evaluation for the translation is generally high. Therefore, we obtain WE_WPI by taking the value of $\frac{min(WORK(T,R,F))}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}$ from 1.0. As a result, in between the translation "Are there topics that you think should discuss world?" and the reference "Are there topics you want to get the world talking about?", 0.608 is obtained as the score using Eq. (13).

The WE_WPI can evaluate the translation based on the meanings of words using word embedding. Moreover, it can deal with the word order using the relative difference between the positions of words in the translation and the reference.

4 Experiments

4.1 Experiment Data and Procedure

We conducted evaluation experiments to confirm the effectiveness of WE_WPI. The "newstest2016" set, which is the main test set in WMT16 metrics shared task (Bojar et al., 2016), was used. The script is available at `http://www.statmt.org/wmt16/results.html`.

	cs-en		**de-en**		**fi-en**		**ro-en**		**ru-en**		**tr-en**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
♯ Assessments	70k	12k	15k	12k	19k	14k	11k	12k	18k	13k	7k	13k
♯ Translations	8.6k	560	2.4k	560	4.6k	560	2.2k	560	4.7k	560	2.2k	560
Correlation	τ	r	τ	r	τ	r	τ	r	τ	r	τ	r
sentBLEU	.284	.557	.368	.484	.265	.448	.272	.499	.330	.502	.245	.532
METEOR	**.391**	.636	.393	.500	.351	.539	.297	.578	.370	.541	.334	.604
IMPACT	.338	.624	.342	.535	.301	.510	.248	.531	.309	.541	.282	.602
RIBES	.254	.530	.288	.415	.237	.372	.176	.375	.240	.401	.213	.336
MEANT 2.0 (Lo, 2017)	.355	**.674**	.414	.539	**.453**	.510	**.345**	**.607**	**.401**	.535	**.373**	.588
MEANT 2.0 - nosrl	.347	.672	.411	.522	.438	.484	.338	.587	.400	.540	.364	.577
WE	.372	.617	.395	.472	.365	.517	.316	.545	.362	.523	.346	.572
WE_WPI	.387	.649	**.417**	**.548**	.361	**.540**	.308	.555	.371	**.555**	.347	**.625**

Table 6: Segment-level metric results for to-English language pairs with absolute values of correlation coefficients reported for all metrics: correlation of segment-level metric scores with human assessment variants, where τ are official results computed similarly to Kendall's τ and over standard WMT relative ranking (RR) human assessments; r are Pearson correlation coefficients of metric scores with direct assessment (DA) of absolute translation adequacy.

	en-cs		**en-de**		**en-fi**		**en-ro**		**en-ru**		**en-tr**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
♯ Assessments	118k	-	35k	-	31k	-	7k	-	21k	20k	7k	-
♯ Translations	12.9k	-	6.2k	-	4.1k	-	1.9k	-	6.0k	-	3.0k	-
Correlation	τ	r	τ	r	τ	r	τ	r	τ	r	τ	r
sentBLEU	.223	-	.269	-	.145	-	.171	-	.283	.557	.171	-
METEOR	.245	-	.268	-	.189	-	.177	-	.309	.600	.207	-
IMPACT	.240	-	.263	-	.170	-	.180	-	.297	.609	.231	-
RIBES	.139	-	.188	-	.057	-	.101	-	.206	.442	.153	-
WE	**.359**	-	.347	-	**.360**	-	**.285**	-	**.427**	.625	.336	-
WE_WPI	.352	-	**.371**	-	.357	-	.283	-	.424	**.652**	**.370**	-

Table 7: Segment-level metric results for out-of-English language pairs with the absolute values of correlation coefficients reported for all metrics: absolute correlation of segment-level metric scores with human assessment variants, where τ are official results computed similarly to Kendall's τ and over standard WMT relative ranking (RR) human assessments; r are Pearson correlation coefficients of metric scores with direct assessment (DA) of absolute translation adequacy.

Therefore, we can readily obtain the correlation coefficient between the metrics and human judgments in WMT16 metrics shared task. The WMT16 metrics task includes English paired with Czech, German, Finnish, Romanian, Russian, and Turkish. For all translations, references and scores by human judgment in these language pairs are obtained from the url described above.

For these experiments, we used different automatic MT evaluation metrics for comparison with our WE_WPI: BLEU, METEOR, IMPACT, RIBES, and WE. Here, IMPACT and RIBES, which are surface-based metrics, are effective for language pairs with greatly different word order, such as English and Japanese. In addition, WE is an automatic MT evaluation metric that does not perform word alignment. It uses only $d_{ij} = 1.0 - cos\_sim(t_i, r_j)$ as the d_{ij} of Eq. (12) in the WE_WPI. In both WE and WE_WPI, the word vectors for seven languages (*i.e.*, English, Czech, German, Finnish, Romanian, Russian, and Turkish) were obtained using fastText (Grave et al., 2018).

4.2 Experiment Results and Discussion

Tables 4 and 5 respectively present the correlation coefficient of to-English and out-of-English at the system level. Tables 6 and 7 respectively present the correlation coefficients of to-English and out-of-English at the segment level.

In Tables 4–7, RR represents the correlation based on the relative ranking by human judgment to 5 translations at a time. The bold typeface shows the highest correlation coefficient among all correlation coefficients of metrics. Moreover, the coefficients of MEANT 2.0 described in (Lo, 2017) are added to Tables 4–6. Here, WE_WPI achieves the highest correlation with human judgment in Table 5, DA in Table 6, and Table 7. Especially, the correlation coefficients of WE_WPI are high with language pairs for which the grammar differs (*i.e.*, English-to-German (en-de), German-to-English (de-en),

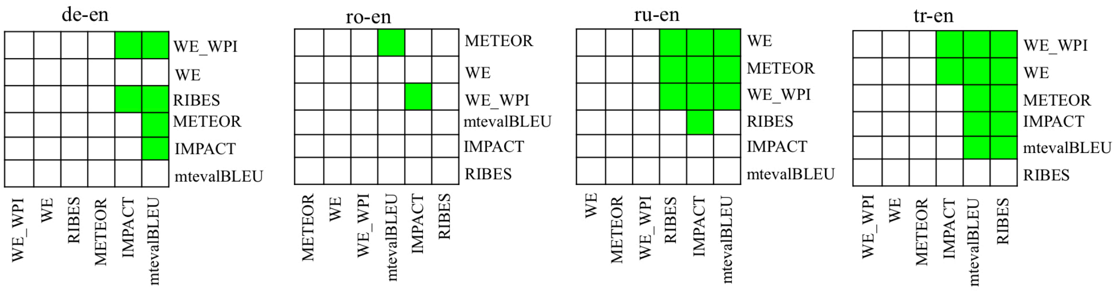


Figure 4: To-English system-level metric significance test of results for human assessment variants, where DA denotes the direct assessment of translation adequacy. Green cells show a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column according to the Williams test.

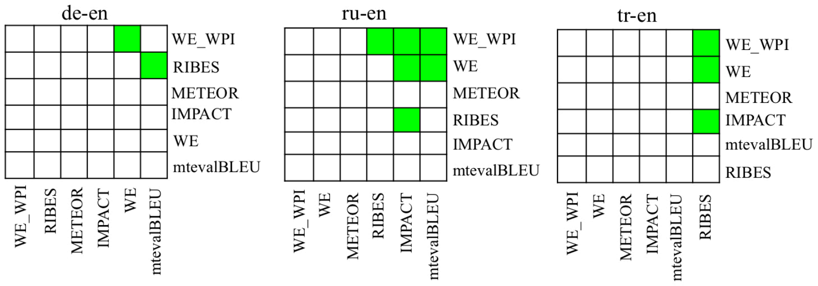


Figure 5: To-English system-level metric significance test of results for human assessment variants, where RR denotes the standard WMT relative ranking for the translation task system only. Green cells show a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column according to the Williams test.

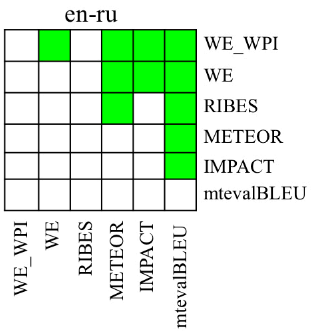


Figure 6: English-to-Russian system-level metric significance test of results for human assessment variants, where DA denotes direct assessment of translation adequacy. Green cells show a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column according to the Williams test.

English-to-Turkish (en-tr), and Turkish-to-English (tr-en)). Therefore, the WE_WPI is effective with such language pairs because it uses word position information.

Moreover, we investigated the significance of WE_WPI results and those of other metrics except those of MEANT 2.0 and MEANT 2.0 - nosrl. As described herein, Williams significance test (Williams, 1959) was used to assess differences in dependent correlations. Figures 4–9 present significance test results for every competing pair of metrics, including those of our WE_WPI. However, the language pairs for which significant differences could not be obtained in any competing pair of metrics are excluded from Figures 4–9 (*i.e.*, cs-en and fi-en in Figure 4, cs-en, fi-en and ro-en in Figure 5, en-cs in Figure 7).

In Figures 4–9, green cells signify that the metric shows significant difference from other metrics with 95% or greater confidence. Results demonstrated that our WE_WPI yielded significantly different results among metrics. Particularly, WE_WPI was found to have significantly better results than those of WE at the segment level, as shown in Figures 8 and 9. This particular result demonstrates that the word position information in WE_WPI is effective for segment-level evaluation.

Moreover, WE_WPI does not need much time to calculate the scores described in 3.2.2. However, it takes time to calculate $tf \cdot idf$ of words and to change the surface-level words to the word vectors. It is efficient to calculate $tf \cdot idf$ of all words in the translations and references, and to extract the word vectors, which correspond to the words in the translations and references, from the fastText models in advance.

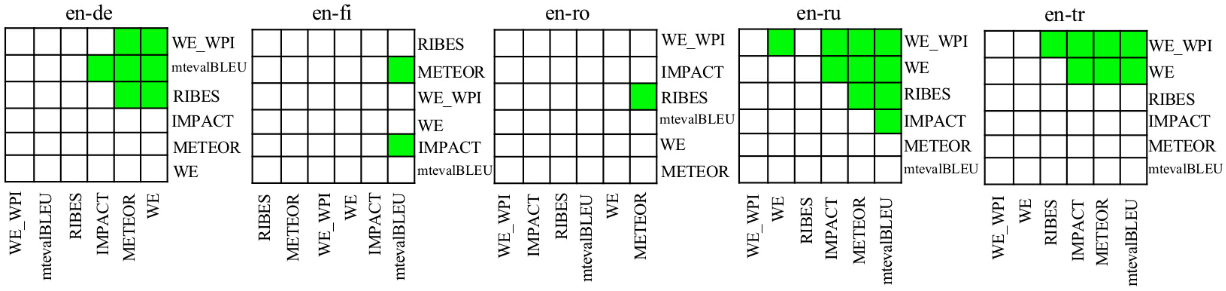


Figure 7: Out-of-English system-level metric significance test of results for human assessment variants, where RR denotes the standard WMT relative ranking for translation task system only. Green cells show a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column according to the Williams test.

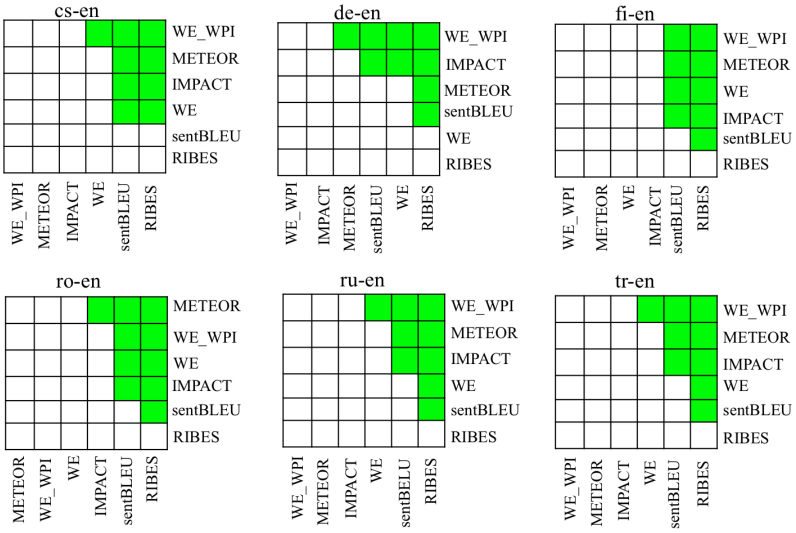


Figure 8: To-English segment-level metric significance test of results for human assessment variants, where DA denotes direct assessment of translation adequacy. Green cells show marked benefits obtained with the metric in a given row over the metric in a given column according to the Williams test.

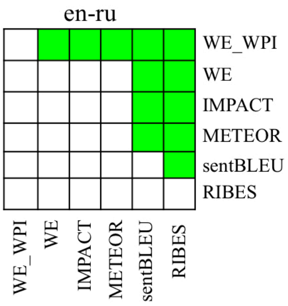


Figure 9: English-to-Russian segment-level metric significance test of results for human assessment variants, where DA denotes direct assessment of translation adequacy: green cells show marked benefits obtained with the metric in a given row over the metric in a given column according to the Williams test.

5 Conclusion

As described herein, we proposed WE_WPI as a new automatic MT evaluation metric. It produces an evaluation based on the meanings of words using word embedding. Moreover, it can accommodate word-order differences. Evaluation experiments demonstrated that our WE_WPI obtains the highest correlation with human judgments among several representative metrics in language pairs for which the grammar differs, and demonstrated that it is significantly better than other metrics at segment-level evaluation.

Our future work will improve WE_WPI to obtain high-quality evaluation scores in combination with other metrics. We will conduct evaluation experiments using various data. Moreover, we will use WE_WPI to improve NMT quality. For instance, WE_WPI can be used easily in Minimum Risk Training (MRT) (Shen et al., 2016), which minimizes the expected loss on the training data.

Acknowledgments

This work was partially supported by grants from Hokkai-Gakuen University.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 65–72.

Ondřej Bojar, Yvette Graham, and Amir Kamran. 2017. Result of the wmt17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation (WMT17)*, pages 489–513.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016. Result of the wmt16 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 199–231.

Boxing Chen and Hongyu Guo. 2015. Representation based translation evaluation metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 150–155.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270.

Hiroshi Echizen-ya and Kenji Araki. 2007. Automatic evaluation of machine translation based on recursive acquisition of an intuitive common parts continuum. In *Proceedings of the Eleventh Machine Translation Summit*, pages 151–158.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3483–3487.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Killian Q. Weinberger. 2016. From word embeddings to document distance. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 957–966.

Chi–kiu Lo. 2017. Meant 2.0: Accurate semantic mt evaluation for any output language. In *Proceedings of the Second Conference on Machine Translation (WMT17)*, pages 589–597.

Chi–kiu Lo and Dekai Wu. 2011. Meant: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 220–229.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Junki Matsuo, Mamoru Komachi, and Katsuhito Sudoh. 2017. Word-alignment-based segment-level machine translation evaluation using word embeddings. arXiv:1704.00380. Version 1.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Gubas. 1998. A metric for distributions with applications to image databases. In *Proceedings of ICCV*, pages 59–66.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Gubas. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(12):99–121.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1683–1692.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Neural Information Processing Systems*, arXiv:1409.3215. Version 3.

Dan Tufiş and Alexandru Ceauşu. 2009. Factored phrase-based statistical machine translation. In *Proceedings of the 5th Conference Speech Technology and Human-Computer Dialogue*, pages 1–7.

Evan James Williams. 1959. Regression analysis. 14. Wiley New York.

Learning to Stop in Structured Prediction for Neural Machine Translation

Mingbo Ma *[†,‡] **Renjie Zheng** *[‡,†] **Liang Huang** [†,‡]
[†]Baidu Research, Sunnyvale, CA
[‡]School of EECS, Oregon State University, Corvallis, OR
{cosmmb, zrenjl1, liang.huang.sh}@gmail.com

Abstract

Beam search optimization (Wiseman and Rush, 2016) resolves many issues in neural machine translation. However, this method lacks principled stopping criteria and does not learn how to stop during training, and the model naturally prefers *longer* hypotheses during the testing time in practice since they use the raw score instead of the probability-based score. We propose a novel ranking method which enables an optimal beam search stopping criteria. We further introduce a structured prediction loss function which penalizes suboptimal finished candidates produced by beam search during training. Experiments of neural machine translation on both synthetic data and real languages (German→English and Chinese→English) demonstrate our proposed methods lead to better length and BLEU score.

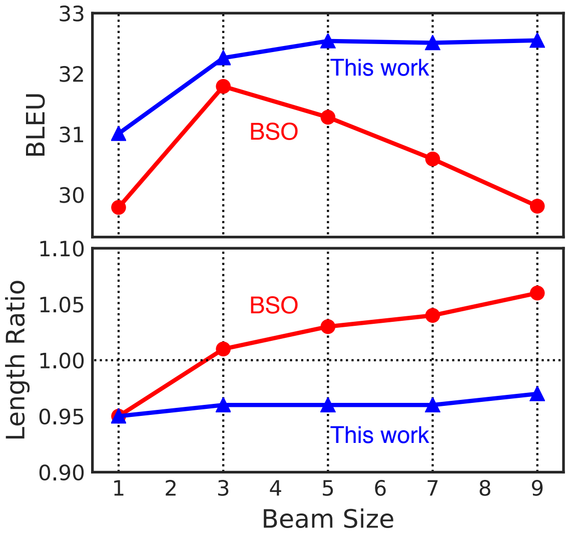


Figure 1: The BLEU score of BSO decreases after beam size 3 as results of increasing length ratio[1] in German→English translation. Our model gets higher BLEU with larger beam.

1 Introduction

Sequence-to-sequence (seq2seq) models based on RNNs (Sutskever et al., 2014; Bahdanau et al., 2014), CNNs (Gehring et al., 2017) and self-attention (Vaswani et al., 2017) have achieved great successes in Neural Machine Translation (NMT). The above family of models encode the source sentence and predict the next word in an autoregressive fashion at each decoding time step. The classical "cross-entropy" training objective of seq2seq models is to maximize the likelihood of each word in the translation reference given the source sentence and all previous words in that reference. This word-level loss ensures efficient and scalable training of seq2seq models.

However, this word-level training objective suffers from a few crucial limitations, namely the *label bias*, the *exposure bias*, and the *loss-evaluation mismatch* (Lafferty et al., 2001; Bengio et al., 2015a; Venkatraman et al., 2015). In addition, more importantly, at decoding time, *beam search* is universally adopted to improve the search quality, while training is fundamentally local and greedy. Several researchers have proposed different approaches to alleviate above problems, such as reinforcement learning-based methods (Ranzato et al., 2016; Rennie et al., 2017; Zheng et al., 2018b), training with alternative references (Shen et al., 2016; Zheng et al., 2018a). Recently, Wiseman and Rush (2016) attempt to address these issues with a structured training method, Beam Search Optimization (BSO). While BSO outperforms other proposed methods on German-to-English translation, it also brings a different set of problems as partially discussed in (Ma, 2018) which we present with details below.

---

[*] Equal contribution

[1] There are two types of "length ratios" in this paper: (a) target to reference ratio ($|\mathbf{y}|/|\mathbf{y}^*|$), which is used in BLEU, and (b) target to source ratio ($|\mathbf{y}|/|\mathbf{x}|$). By default, the term "length ratio" in this paper refers to the former.

Proceedings of NAACL-HLT 2019, pages 1884–1889
Minneapolis, Minnesota, June 2 - June 7, 2019.

BSO relies on unnormalized raw scores instead of locally-normalized probabilities to get rid of the label bias problem. However, since the raw score can be either positive or negative, the optimal stopping criteria (Huang et al., 2017) no longer holds, e.g., one extra decoding step would increase the entire unfinished hypothesis's model score when we have positive word score. This leads to two consequences: we do not know when to stop the beam search and it could return overlength translations (Fig. 1) or underlength translations (Fig. 3) in practice. As shown in Fig. 1, the BLEU score of BSO drops significantly when beam size gets larger as a result of overlong translations (as evidenced by length ratios larger than 1). Furthermore, BSO performs poorly (shown in Section 4) on hard translation pairs, e.g., Chinese→English (Zh→En) translation, when the target / source ratio is more diverse (Table 1).

To overcome the above issues, we propose to use the sigmoid function instead of the raw score at each time step to rank candidates. In this way, the model still has probability properties to hold optimal stopping criteria without label bias effects. Moreover, we also encourage the model to generate the hypothesis which is more similar to gold reference in length. Compared with length reward-based methods (Huang et al., 2017; Yang et al., 2018), our model does not need to tune the predicted length and per-word reward. Experiments on both synthetic and real language translations (De→En and Zh→En) demonstrate significant improvements in BLEU score over strong baselines and other methods.

2 Preliminaries: NMT and BSO

Here we briefly review the conventional NMT and BSO (Wiseman and Rush, 2016) to set up the notations. For simplicity, we choose to use RNN-based model but our methods can be easily applied to other designs of seq2seq model as well.

Regardless of the particular design of different seq2seq models, generally speaking, the decoder always has the following form:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{x},\, \mathbf{y}_{<t}) \qquad (1)$$

where $\mathbf{x} \in R^{N \times D}$ represents the D-dimension hidden states from encoder with N words and $\mathbf{y}_{<t}$ denotes the gold prefix $(y_1, ..., y_{(t-1)})$ before t. The conventional NMT model is locally trained to maximize the above probability.

Instead of maximizing each gold word's probability, BSO tries to promote the non-probabilistic scores of gold sequence within a certain beam size b. BSO removes the softmax layer and directly uses the raw score after hidden-to-vocabulary layer, and the non-probabilistic scoring function $f_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t})$ represents the score of word y_t given gold prefix $\mathbf{y}_{<t}$ and $\mathbf{x}$. Similarly, $f_{\mathbf{x}}(\hat{y}_t^b \mid \hat{\mathbf{y}}_{<t}^b)$ is the b^{th} sequence with beam size b at time step t. Then, we have the following loss function to penalize the b^{th} candidate and promote gold sequence:

$$\mathbb{L} = \sum_{t=1}^{|\mathbf{y}|} \Delta(\hat{\mathbf{y}}_{\leq t}^b)(1 + f_{\mathbf{x}}(\hat{y}_t^b \mid \hat{\mathbf{y}}_{<t}^b) - f_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t}))^+ \qquad (2)$$

where $\Delta(\hat{\mathbf{y}}_{\leq t}^b)$ is defined as $(1-\text{BLEU}(\hat{\mathbf{y}}_{\leq t}^b, \mathbf{y}_{\leq t}))$ which scales the loss according to BLEU score between gold and b^{th} hypothesis in the beam. The notation $(\cdot)^+$ represents a max function between any value and 0, i.e., $z^+ = max(0, z)$.

When Eq. 1 equals to 0 at time step t, then the gold sequence's score is higher than the last hypothesis in the beam by 1, and a positive number otherwise. Finally, at the end of beam search ($t = |\mathbf{y}|$), BSO requires the score of $\mathbf{y}$ exceed the score of the highest incorrect hypothesis by 1.

Note that the above non-probabilistic score function $f_{\mathbf{x}}(\cdot)$ is not bounded as probabilistic score in conventional NMT. In practice, when we have positive word score, then the unfinished candidates always get higher model scores with one extra decoding step and the optimal stopping criteria [2] (Huang et al., 2017) is no longer hold. BSO implements a similar "shrinking beam" strategy which duplicates top unfinished candidate to replace finished hypotheses and terminates the beam search when there are only `</eos>` in the beam. Non-probabilistic score function works well in parsing and Statical MT where we know when to stop beam search. However, in the NMT scenario, without optimal stopping criteria, we don't know when to stop beam search.

3 Learning to Stop

We propose two major improvements to BSO.

3.1 Sigmoid Scoring Function

As mentioned in Section 2, BSO relies on raw score function to eliminate label bias effects.

---

[2] Beam search stops when the score of the top unfinished hypothesis is lower than any finished hypothesis, or the `</eos>` is the highest score candidate in the beam.

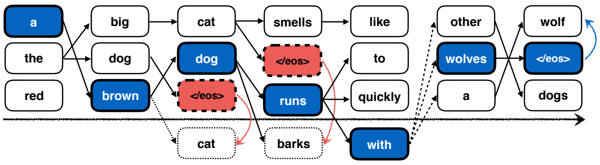


Figure 2: Training illustration with beam size $b = 3$ and gold reference "a brown dog runs with the wolves". The gold reference is highlighted in blue solid boxes. We penalize the under-length translation (short) hypotheses by expelling out the early `</eos>` out of beam (red dashed boxes). The beam search restarts with gold when gold falls off the beam (at step 5).

However, without using locally-normalized score does not mean that we should stop using the probabilistic value function. Similar with multi-label classification in (Ma et al., 2017), instead of using locally normalized softmax-based score and non-probabilistic raw scores, we propose to use another form of probabilistic scoring function, sigmoid function, which is defined as follows:

$$g_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t}) = (1 + e^{w \cdot f_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t})})^{-1} \tag{3}$$

where w is a trainable scalar parameter which shifts the return value of $f_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t})$ into a non-saturated value region of sigmoid function. Eq. 3 measures the probability of each word independently which is different from locally-normalized softmax function. Similar to the scenario in multi-label classification, $g_{\mathbf{x}}(y_t \mid \mathbf{y}_{<t})$ only promotes the words which are preferred by gold reference and does not degrade other words. Eq. 3 enables the model to keep the probability nature of scoring function without introducing label bias effects. After the model regain probability-based scoring function, the optimal stopping criteria can be used in testing time decoding.

3.2 Early Stopping Penalties

Similar to Eq. 1, testing time decoder multiplies the new word's probabilistic score with prefix's score when there is a new word appends to an unfinished hypothesis. Though the new word's probabilistic score is upper bounded by 1, in practice, the score usually far less than one. As described in (Huang et al., 2017; Yang et al., 2018), decoder always prefers short sentence when we use the probabilistic score function.

To overcome the above so-called "*beam search curse*", we propose to penalize early-stopped hypothesis within the beam during training. The procedure during training is illustrated in Fig. 2.

Data	Split	$\overline{\lvert x\rvert}$	$\sigma(\lvert x\rvert)$	$\overline{(\frac{\lvert y\rvert}{\lvert x\rvert})}$	$\sigma(\frac{\lvert y\rvert}{\lvert x\rvert})$	# sents
Synthetic	Train	9.47	5.45	3.0	0.52	5K
	Valid	9.54	5.42	3.0	0.53	1K
	Test	9.51	5.49	3.0	0.52	1K
De→En	Train	17.53	9.93	1.07	0.16	153K
	Valid	17.55	9.97	1.07	0.16	7K
	Test*	18.89	12.82	1.06	0.16	6.5K
Zh→En	Train	23.21	13.44	1.30	0.33	1M
	Valid	29.53	16.62	1.34	0.22	0.6K
	Test	26.53	15.99	1.4	0.24	0.7K

Table 1: Dataset statistics of source sentence length and the ratio between target and source sentences. σ is standard deviation. *shows statistics of cleaned test set.

Different from BSO, to penalize the underlength finished translation hypotheses, we include additional violations when there is an `</eos>` within the beam before the gold reference finishes and we force the score of that `</eos>` lower than the $b+1$ candidate by a margin. This underlength translation violation is formally defined as follows:

$$\begin{aligned} \mathbb{L}^{\mathrm{s}} &= \sum_{t=1}^{|\mathbf{y}|} \sum_{j=1}^{b} \mathbb{1}(\hat{y}_t^j = \texttt{</eos>}) \cdot Q(\hat{y}_t^j, \hat{y}_t^{b+1}) \ , \\ Q(\hat{y}_t^j, \hat{y}_t^{b+1}) &= (1 + f_{\mathbf{x}}(\hat{y}_t^j \mid \hat{\mathbf{y}}_{<t}^j) - f_{\mathbf{x}}(\hat{y}_t^{b+1} \mid \hat{\mathbf{y}}_{<t}^{b+1}))^+ \end{aligned} \tag{4}$$

where notation $\mathbb{1}$ is identification function which only equals to 1 when i^{th} candidate in beam $\hat{y}_t^j$ is `</eos>`, e.g. in Fig. 2. We only have non-zero loss when the model score of underlength translation candidates are greater than the $b+1$ candidate by a margin. In this way, we penalize all the short hypotheses during training time. Note that during both training and testing time, the decoder stops beam search when it satisfies the optimal stopping criteria (Huang et al., 2017). Therefore, we do not need to penalize the overlength translations since we have already promoted the gold reference to the top of the beam at time step $|y|$ during training.

4 Experiments

We showcase the performance comparisons over three different datasets. We implement seq2seq model, BSO and our proposed model based on PyTorch-based OpenNMT (Klein et al., 2017). We use a two-layer bidirectional LSTM as the encoder and a two layer LSTM as the decoder. We train Seq2seq model for 20 epochs to minimize perplexity on the training dataset, with a batch size of 64, word embedding size of 512, the learning rate of 0.1, learning rate decay of 0.5 and dropout rate of 0.2. Following Wiseman and Rush (2016), we then train BSO and our model based on the previous Seq2seq model with the learning rate of

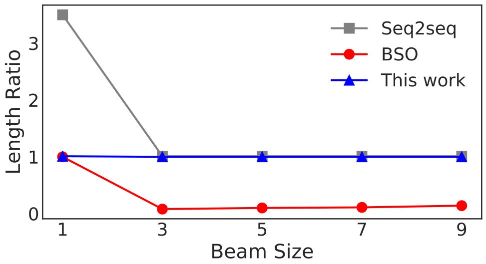


Figure 3: Length ratio on synthetic test dataset.

0.01 and learning rate decay of 0.75, batch size of 40. Note that our pretrained model is softmax-based, and we only replace the softmax layer with the sigmoid layer for later training for simplicity. The performance will have another boost when our pretrained model is sigmoid-based. We use Adagrad (Duchi et al., 2011) as the optimizer.

In Zh→En task, we employ BPE (Sennrich et al., 2015) which reduces the source and target language vocabulary sizes to 18k and 10k. Following BSO, we set the decoding beam size smaller than the training beam size by 1.

4.1 Synthetic Task

Table 1 shows the statistics of source sentence length and the ratio between target and source sentences. The synthetic dataset is a simple translation task which generates target sentences from this grammar: $\{a \to x, b \to x\ x, c \to x\ x\ x, d \to x\ x\ x\ x, e \to x\ x\ x\ x\ x\}$. For example:

1. source sentence $[b\ c\ a]$ will generate the target sentence $[x\ x\ x\ x\ x\ x]$ (2 x from b, 3 x from c and 1 x from a).
2. source sentence $[a, b, c, d, e]$ will be translated into $[x\ x\ x\ x\ x\ x\ x\ x\ x\ x\ x\ x\ x\ x\ x]$ in target side (1 x from a, 2 x from b, 3 x from c, 4 x from d and 5 x from e).

This dataset is designed to evaluate the length prediction ability of different models. Fig. 3 shows the length ratio of different models on the test set. Only our model can predict target sentence length correctly with all beam sizes which shows a better ability to learn target length.

4.2 De→En Translation

The De→En dataset is previously used in BSO and MIXER (Ranzato et al., 2016), which is from IWSLT 2014 machine translation evaluation campaign (Cettolo et al., 2014) [3].

Decode	Seq2seq†		Train	BSO†		This work	
Beam	BLEU	Len.	Beam	BLEU	Len.	BLEU	Len.
1	30.65	1.00	2	29.79	0.95	31.01	0.95
3	31.38	0.97	4	31.79	1.01	32.26	0.96
5	31.38	0.97	6	31.28	1.03	32.54	0.96
7	31.42	0.96	8	30.59	1.04	32.51	0.96
9	31.44	0.96	10	29.81	1.06	32.55	0.97

Table 2: BLEU and length ratio on the De→En validation set. †indicates our own implementation.

Model	BLEU	Len.
This work (full model)	32.54	0.96
This work w/ softmax	32.29	0.98
This work w/o scale augment	31.97	0.95
This work w/o early stopping loss	31.19	0.93

Table 3: Ablation study on the De→En validation set with training beam size $b = 6$.

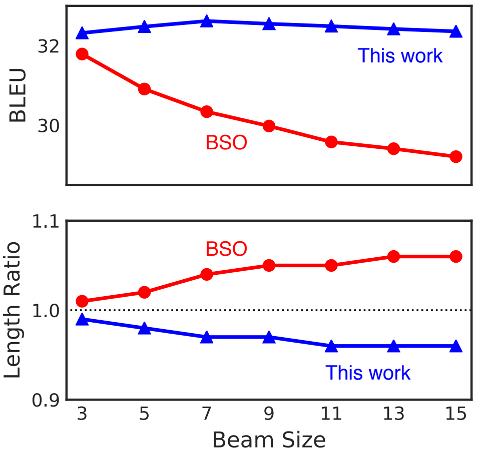


Figure 4: BLEU and length ratio of models with training beam size $b = 6$ and decode with different beam size on De→En dataset.

Table 2 shows the BLEU score and length ratio of different models on dev-set. Similar to seq2seq, our proposed model achieves better BLEU score with larger beam size and outperforms the best BSO $b = 4$ model with 0.76 BLEU. The ablation study in Table 3 shows that the model produces shorter sentence without scale augment (term $\Delta(\hat{y}^b_{\le t})$ in Eq. 2) and early stopping loss. The model also performs worse when replacing softmax to sigmoid because of the label bias problem. Fig. 4 shows BLEU score and length ratio of BSO and our models trained with beam size $b = 6$ with different decoding beam size. Compared with BSO, whose BLEU score degrades dramatically when increasing beam size, our model performs much more stable. Moreover, BSO achieves much better BLEU score with decoding beam $b = 3$ while trained with $b = 6$ because of a better

---

[3] The test set of De→En involves some mismatched source-reference pairs. We have cleaned this test set and report the statistics based on the cleaned version.

Model	Original Test Set		Cleaned Test Set	
	BLEU	Len.	BLEU	Len.
BSO‡	26.35	-	-	
DAD‡	22.40	-	-	
MIXER‡	21.83	-	-	
Seq2seq†	29.54	0.97	30.08	0.97
BSO†	29.63	1.02	30.08	1.02
This work	30.29	0.98	30.85	0.98

Table 4: BLEU and length ratio on the De→En test set. †indicates our own implementation. ‡results from (Wiseman and Rush, 2016).

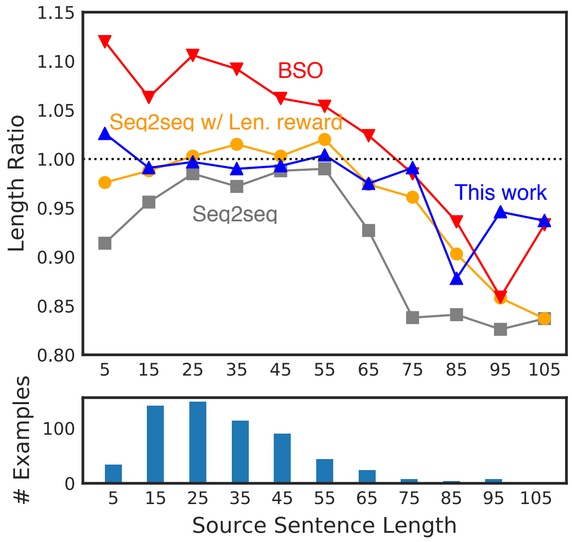


Figure 5: Length ratio of examples on Zh→En dev-set with different source sentence length.

length ratio, this is inconsistent with their claim that decoding beam size should smaller than training beam size by 1.

Table 4 shows better accuracy of our proposed model than not only published test results of BSO (Wiseman and Rush, 2016), DAD (Bengio et al., 2015b) and MIXER (Ranzato et al., 2016), but also our implemented seq2seq and BSO model.

4.3 Zh→En Translation

Model	Train Beam	Decode Beam	BLEU	Len.
Seq2Seq†	-	7	37.74	0.96
w/ Len. reward†	-	7	38.28	0.99
BSO†	4	3	36.91	1.03
BSO†	8	7	35.57	1.07
This work	4	3	38.41	1.00
This work	8	7	39.51	1.00

Table 5: BLEU and length ratio of models on Zh→En validation set. †indicates our own implementation.

Model	BLEU	Len.
Seq2Seq†	34.19	0.95
w/ Len. reward†	34.60	0.99
BSO†	31.78	1.04
This work	35.40	0.99

Table 6: BLEU and length ratio of models on Zh→En test set. †indicates our own implementation.

We also perform experiments on NIST Zh→En translation dataset. We use the NIST 06 and 08 dataset with 4 references as the validation and test set respectively. Table 1 shows that the characteristic of Zh→En translation is very different from De→En in source length and variance in target/source length ratio.

We compare our model with seq2seq, BSO and seq2seq with length reward (Huang et al., 2017) which involves hyper-parameter to solve neural model's tendency for shorter hypotheses (our proposed method does not require tuning of hyper-parameter). Fig. 5 shows that BSO prefers overlength hypotheses in short source sentences and underlength hypotheses when the source sentences are long. This phenomenon degrades the BLEU score in dev-set from Table 5. Our proposed model comparatively achieves better length ratio on almost all source sentence length in dev-set.

5 Future Works and Conclusions

Our proposed methods are general techniques which also can be applied to the Transformer (Vaswani et al., 2017). As part of our future works, we plan to adapt our techniques to the Transformer to further evaluate our model's performance.

There are some scenarios that decoding time beam search is not applicable, such as the simultaneous translation system proposed by Ma et al. (2018) which does not allow for adjusting the committed words, the training time beam search still will be helpful to the greedy decoding performance. We plan to further investigate the performance of testing time greedy decoding with beam search optimization during training.

We propose two modifications to BSO to provide better scoring function and under-translation penalties, which improves the accuracy in De-En and Zh-En by 0.8 and 3.7 in BLEU respectively.

Acknowledgments

This work was supported in part by DARPA grant N66001-17-2-4030, and NSF grants IIS-1817231 and IIS-1656051.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015a. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015b. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *EMNLP*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Mingbo Ma. 2018. Structured neural models for natural language processing. In *Ph.D thesis, Oregon State University*.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2017. Group sparse cnns for question classification with answer sets. In *ACL*.

Mingbo Ma, Liang Huang, Hao Xiong, Kaibo Liu, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, and Haifeng Wang. 2018. Stacl: Simultaneous translation with integrated anticipation and controllable latency. *ArXiv*, abs/1810.08398.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *AAAI*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of EMNLP*.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Renjie Zheng, Mingbo Ma, and Liang Huang. 2018a. Multi-reference training with pseudo-references for neural translation and text generation. In *EMNLP*.

Renjie Zheng, Yilin Yang, Mingbo Ma, and Liang Huang. 2018b. Ensemble sequence level training for multimodal mt: Osu-baidu wmt18 multimodal machine translation system report. In *WMT*.

Learning Unsupervised Multilingual Word Embeddings with Incremental Multilingual Hubs

Geert Heyman[†], **Bregt Verreet**[†], **Ivan Vulić**[‡], **Marie-Francine Moens**[†]
[†] LIIR, Department of Computer Science, KU Leuven
[‡] Language Technology Lab, TAL, University of Cambridge
`{geert.heyman,bregt.verreet,sien.moens}@kuleuven.be`
`iv250@cam.ac.uk`

Abstract

Recent research has discovered that a shared bilingual word embedding space can be induced by projecting monolingual word embedding spaces from two languages using a self-learning paradigm without any bilingual supervision. However, it has also been shown that for distant language pairs such fully unsupervised self-learning methods are unstable and often get stuck in poor local optima due to reduced isomorphism between starting monolingual spaces. In this work, we propose a new robust framework for learning unsupervised multilingual word embeddings that mitigates the instability issues. We learn a shared multilingual embedding space for a variable number of languages by incrementally adding new languages one by one to the current multilingual space. Through the gradual language addition our method can leverage the interdependencies between the new language and all other languages in the current multilingual hub/space. We find that it is beneficial to project more distant languages later in the iterative process. Our fully unsupervised multilingual embedding spaces yield results that are on par with the state-of-the-art methods in the bilingual lexicon induction (BLI) task, and simultaneously obtain state-of-the-art scores on two downstream tasks: multilingual document classification and multilingual dependency parsing, outperforming even supervised baselines. This finding also accentuates the need to establish evaluation protocols for cross-lingual word embeddings beyond the omnipresent intrinsic BLI task in future work.

1 Introduction

The ubiquitous use and success of word embeddings in monolingual tasks inspired further research on inducing cross-lingual word embeddings for two or more languages in the same vector space. Embeddings of translations and words with similar meaning are geometrically close in the *shared cross-lingual vector space*. This property makes them effective features for cross-lingual NLP tasks such as cross-lingual document classification (Klementiev et al., 2012), cross-lingual information retrieval (Vulić and Moens, 2015), bilingual lexicon induction (Mikolov et al., 2013b; Gouws et al., 2015; Heyman et al., 2017), and (unsupervised) machine translation (Artetxe et al., 2017b; Lample et al., 2018; Artetxe et al., 2018c).

Most prior work has focused on methods for constructing *bilingual* word embeddings (BWEs), yielding word representations for exactly two languages. For problems such as multilingual document classification, however, it is highly-desirable to represent words in a *multilingual* space. A favourable property is that it enables fitting a single classifier on the union of training datasets in many languages, which results in 1) knowledge transfer across languages that may lead to better classification performance, and 2) a setup that is easier to maintain as it is no longer required to train many different monolingual or bilingual classifiers.

Multilingual word embedding (MWE) methods typically generalize existing BWE methods by mapping multiple source language spaces to the space of one target language (Ammar et al., 2016), which is used as a pivot/hub language. This approach may lead to suboptimal solutions as it does not account for interdependencies between the source languages. Most BWE and MWE methods rely on *cross-lingual supervision* to some extent: e.g., bilingual lexicons (Mikolov et al., 2013a), parallel corpora (Gouws et al., 2015), or subject-aligned document pairs (Vulić and Moens, 2016). In such paradigms, modeling dependencies between all languages is impractical as it requires supervision for all language pair combinations.

Recent research has shown that BWEs can also be learned *without* cross-lingual supervision and can even outperform supervised BWE variants

Proceedings of NAACL-HLT 2019, pages 1890–1902
Minneapolis, Minnesota, June 2 - June 7, 2019.

on bilingual lexicon induction benchmarks (Conneau et al., 2018; Artetxe et al., 2018a). Chen and Cardie (2018) took a first step towards learning multilingual spaces without supervision while incorporating dependencies between all languages but their approach extends the work of Conneau et al. (2018), which has known limitations concerning optimization stability with distant language pairs (Søgaard et al., 2018). In this work, we investigate robust methods to induce MWEs without any cross-lingual supervision. The robustness of our approach is illustrated in good performance for distant languages such as Finnish and Bulgarian. This paper makes the following contributions.

First, based on a reformulation of the BWE method of Artetxe et al. (2018a), we propose two novel methods for inducing MWEs: 1) the single hub space model (SHS) uses the classical idea of mapping source languages to a single hub language; 2) the incremental hub space model (IHS) incorporates dependencies between all languages by incrementally expanding the multilingual space by one language in each step. IHS results in mappings that are more robust and coherent across languages. Both SHS and IHS only require monolingual data.

Second, we evaluate our method on benchmarks for bilingual lexicon induction (BLI), multilingual document classification, and dependency parsing. We find that the IHS method is competitive with state-of-the-art BWE methods on the bilingual lexicon induction benchmarks, while yielding the highest scores on the multilingual document classification and dependency parsing benchmarks.

Third, unlike the majority of prior work (Conneau et al., 2018; Artetxe et al., 2018a; Chen and Cardie, 2018, *inter alia*), we do not limit our evaluation to the intrinsic BLI task only. Consequently, we investigate if embedding reweighting, a recently proposed best practice for BWEs, is useful for extrinsic tasks such as document classification and dependency parsing in multilingual settings.

2 Related Work

Cross-lingual word embeddings have received a lot of attention in recent years. Most methods construct a space shared between two languages using cross-lingual supervision in the form of bilingual lexicons (Mikolov et al., 2013a; Artetxe et al., 2016; Smith et al., 2017), parallel corpora (Klementiev et al., 2012; Faruqui and Dyer, 2014; Gouws et al., 2015; Luong et al., 2015) or subject-aligned document pairs (Vulić and Moens, 2016). See Ruder et al. (2018) for a full overview of BWE model typology in relation to the required supervision.

To enable knowledge transfer across an arbitrary number of languages, multilingual methods have been introduced. Huang et al. (2015), propose decomposing a matrix with multilingual co-occurrence counts weighted by probabilistic dictionaries. Ammar et al. (2016) compare this method to three other MWE models: MultiCluster, MultiCCA, and MultiSkip. MultiCluster uses bilingual dictionaries to cluster translations and then train the monolingual Skip-gram model (SG) (Mikolov et al., 2013a) on a union of monolingual corpora where they replace words with their cluster id such that words in the same cluster get the same representation. MultiCCA is the multilingual extension of the method of Faruqui and Dyer (2014): Using canonical correlation analysis (CCA) and dictionaries with English as the target language, monolingual embeddings are projected to the English vector space. MultiSkip is a straightforward extension of the BiSkip method (Luong et al., 2015) which generalizes the monolingual SG objective to account for word alignments in parallel corpora. Similarly, Duong et al. (2017), extend CBOW to multiple languages. All these methods learn multilingual embeddings using bilingual dictionaries of parallel corpora: This limits their applicability for many languages.

More recently, Conneau et al. (2018); Artetxe et al. (2018a) showed that BWEs can be effectively induced without any cross-lingual supervision. The approaches are based on the assumption that monolingual embedding spaces are approximately isomorphic.[1] Improving on earlier attempts (Cao et al., 2016; Zhang et al., 2017), Conneau et al. (2018) propose a two-step framework to map two monolingual spaces to the shared space. First, they use an adversarial objective to get an initial bilingual space in which the discriminator can no longer distinguish to which language a given word embedding belongs. They then fine-tune the initial solution. An important limitation is that the adversarial objective is prone to converge to degenerate solutions. Furthermore, Søgaard et al. (2018) empirically prove that the method typically fails for distant language pairs such as English-Finnish.

In parallel, Artetxe et al. (2018a) proposed an-

---

[1] One of the necessary conditions for this assumption to hold is that the monolingual corpora on which the embeddings are trained are comparable (Søgaard et al., 2018).

other framework with the same goal. Expanding on their earlier work (Artetxe et al., 2017a, 2018b), they use an unsupervised heuristic to obtain a noisy initial seed lexicon which is used to obtain an initial bilingual space. This solution is iteratively improved similar to Artetxe et al. (2017a) and Conneau et al. (2018) while using value dropping regularization to escape early convergence to local minima. Their method is the starting point for this work, and it is discussed in detail in §3.

The two unsupervised approaches are limited to finding mappings between a pair of languages. To the best of our knowledge, Chen and Cardie (2018) is the only unsupervised method that constructs a multilingual embedding space. Their method extends the adversarial pre-training and iterative refinement steps of Conneau et al. (2018) to the multilingual setting. Their work does not investigate the limitations of Conneau et al. (2018)'s method: Less stable optimization and difficulties with mapping spaces with reduced isomorphism. Furthermore, their generalization turns the iterative refinement into a non-convex optimization problem. In contrast, our multilingual methods proposed in this work are applicable to distant language pairs and decompose every iteration in the refinement step in multiple convex optimization problems, making them very robust and widely applicable.

3 Unsupervised Bilingual Embeddings

We now summarize mapping-based approaches to learning BWEs, which serve as the backbone of our multilingual approach. These methods rely on a *mapping procedure*, that is, a way to transform two monolingual spaces such that translations and similar words obtain similar representations. Supervised approaches take translations from readily available seed training dictionaries. Unsupervised approaches construct a seed lexicon from scratch, and use an iterative procedure to refine the seed lexicon and the mapped bilingual space.

Mapping Procedure. Various mapping procedures have been proposed in the literature (Mikolov et al., 2013b; Dinu et al., 2015; Lazaridou et al., 2015; Vulić and Korhonen, 2016). These methods can be seen as variants of a a single framework (Artetxe et al., 2018b), summarized here.

At its core, each mapping procedure learns the orthogonal transformations $\boldsymbol{W}_x$ and $\boldsymbol{W}_z$ for the monolingual embedding spaces $\boldsymbol{X}$ and $\boldsymbol{Z}$ that minimize the distance between embeddings of translations in the mapped spaces $\boldsymbol{X}\boldsymbol{W}_x$ and $\boldsymbol{Z}\boldsymbol{W}_z$. The orthogonality constraint ensures that the transformations preserve the monolingual constellation of embeddings. Formally, let $\boldsymbol{D}$ be a matrix representing a bilingual dictionary s.t. $D_{ij} = 1$ if the i^{th} source word is translated by the j^{th} target word and $D_{ij} = 0$ otherwise, then $\boldsymbol{W}_x$ and $\boldsymbol{W}_z$ are found by solving the following optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{W}_x,\boldsymbol{W}_z}{\arg\max} \sum_i \sum_j D_{ij} (\boldsymbol{X}_{i,:}\boldsymbol{W}_x) \cdot (\boldsymbol{Z}_{j,:}\boldsymbol{W}_z) \\ &= \underset{\boldsymbol{W}_x,\boldsymbol{W}_z}{\arg\max}\ tr(\boldsymbol{X}\boldsymbol{W}_x(\boldsymbol{D}\boldsymbol{Z}\boldsymbol{W}_z)^\intercal) \qquad (1) \\ &\text{subject to } \boldsymbol{W}_x\boldsymbol{W}_x^\intercal = \boldsymbol{I}\,, \boldsymbol{W}_z\boldsymbol{W}_z^\intercal = \boldsymbol{I} \end{aligned}$$

where $tr(\cdot)$ denotes the trace operator.

Eq. (1) has a closed-form solution based on the singular vectors of $\boldsymbol{X}^\intercal\boldsymbol{D}\boldsymbol{Z}$: $\boldsymbol{W}_x = \boldsymbol{U}, \boldsymbol{W}_z = \boldsymbol{V}$ with $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\intercal = \text{SVD}(\boldsymbol{X}^\intercal\boldsymbol{D}\boldsymbol{Z})$. In addition to the transformation, there are several optional pre-processing (S1-S2) and post-processing (S3-S5) steps:

S1. *Normalization*: apply length normalization (normalizing $\boldsymbol{X}$ and $\boldsymbol{Z}$ such that all embeddings have a unit Euclidean norm), or mean centering, or a combination of both;

S2. *Whitening*: apply ZCA whitening (Bell and Sejnowski, 1997) on $\boldsymbol{X}$ and $\boldsymbol{Z}$ which transforms the monolingual embedding matrices such that each dimension/component has unit variance and such that the dimensions are uncorrelated (see Eq. (2) later). The intuition is that it is easier to align the vector spaces along directions of high variance;

S3. *Re-weighting* the components according to the singular value matrix $\boldsymbol{S}$ of $\boldsymbol{X}^\intercal\boldsymbol{D}\boldsymbol{Z}$: This is an attempt to further align the embeddings in the multilingual space as each singular value measures how well a dimension in the multilingual space correlates across languages for the given dictionary;

S4. *De-whitening*, the inverse transformation of S2: After the mapping, it was shown as important to restore the variance information in case whitening was applied (Artetxe et al., 2018c);

S5. *Dimensionality reduction* truncates the embedding vectors such that only components with the highest singular values are kept.

Refinement Procedure. The refinement aims at iteratively improving the seed dictionary and the bilingual space with Expectation Maximization (Dempster et al., 1977). In each iteration, the mapping procedure is executed using the dictionary from the previous iteration to obtain a new bilingual space, and then a new dictionary is induced using nearest neighbor retrieval in the *cross-lingual similarity matrix* $\boldsymbol{M}$. This is repeated until the (unsupervised) training objective $\sum_i \sum_j D_{ij}\big((\boldsymbol{X}_{i,:}\boldsymbol{W}_x) \cdot (\boldsymbol{Z}_{j,:}\boldsymbol{W}_z)\big)$ stops increasing.

The matrix $\boldsymbol{M}$ is calculated using cross-domain similarity local scaling (CSLS; Conneau et al. (2018)), an extended variant of cosine similarity that avoids the hubness problem (Radovanović et al., 2010; Dinu et al., 2015).[2] In particular, the element m_{ij} at row i and column j of $\boldsymbol{M}$ corresponds to the CSLS value between the cross-lingual vectors $\boldsymbol{x}_i^{CL}$ and $\boldsymbol{z}_j^{CL}$ of the i^{th} source word and the j^{th} target word respectively: $m_{ij} = \text{CSLS}(\boldsymbol{x}_i^{CL}, \boldsymbol{z}_j^{CL})$; $\text{CSLS}(\boldsymbol{x}, \boldsymbol{z}) = 2cos(\boldsymbol{x}, \boldsymbol{z}) - r_{Zk}(\boldsymbol{x}) - r_{Xk}(\boldsymbol{z})$. $r_{Xk}(\boldsymbol{x})$ and $r_{Zk}(\boldsymbol{z})$ calculate the average cosine similarity of a vector with its k nearest neighbors (measured by cosine) in the mapped spaces of $\boldsymbol{X}$, $\boldsymbol{Z}$, respectively. It is also beneficial to jointly infer dictionaries source-to-target and target-to-source (Artetxe et al., 2018a).[3] The mapping is then learned from the concatenation of these two dictionaries.[4]

To avoid suboptimal local minima, Artetxe et al. (2018a) propose to randomly drop values from the matrix $\boldsymbol{M}$ with probability $1 - p$ (further *value dropping*). The value of p is exponentially increased as training progresses. p is initialized to a small value (e.g., 0.1). Whenever the objective stops improving for $N_{patience}$ refinement steps, p is multiplied with a given factor (e.g., 2) until $p \geq 1$ after which all values in $\boldsymbol{M}$ are kept. Value dropping was shown to be crucial when constructing bilingual spaces between distant language pairs. We later analyze its impact on the proposed multilingual methods.

Inducing a Seed Lexicon. Artetxe et al. (2018a) obtain a seed lexicon based on the assumption that for a translation pair w_i^X, w_j^Z, the monolingual similarity vectors, $\sqrt{\boldsymbol{X}_i\boldsymbol{X}^\top}$ and $\sqrt{\boldsymbol{Z}_j\boldsymbol{Z}^\top}$ of translations i, j are (approximately) equal up to a permutation.[5] Therefore, seed translations for a source word i are generated by finding the nearest neighbor based on similarity of monolingual similarities. This heuristic yields a very noisy seed lexicon, but it was proven to contain a sufficiently strong bilingual signal to bootstrap the refinement procedure. The seed lexicon is inferred symmetrically (i.e., by concatenating respective source-to-target and target-to-source seed lexicons) and the vocabularies are truncated to the C_{seed} most frequent words.

4 Unsupervised Multilingual Embeddings: Methodology

We now present two models for learning unsupervised multilingual word embedding spaces: the single hub space model (SHS) and the incremental hub space model (IHS). The methods generalize the bilingual framework described in §3, and rely on (a subset of) preprocessing and postprocessing steps S1-S5 in the multilingual setting.

Single Hub Space (SHS). The SHS model defines one language as the hub language L_0 and projects the embedding spaces $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_N$ of all other languages $L_1, ..., L_N$ (further *secondary languages*) to the hub space $\boldsymbol{X}$. Hence, we reduce the construction of a multilingual space of N languages to the alignment of $N - 1$ vector spaces. Learning these projections is similar to the bilingual case: We use the unsupervised iterative refinement procedure and seed lexicon heuristic from §3. However, we require the orthogonal mapping to be asymmetric: The hub language space should either remain unchanged or it should be transformed with the same operation for each of the $N - 1$ language pairs. We therefore derive an asymmetric version of the mapping framework from §3 that yields the exact same solution as the original.

Let $\boldsymbol{X}$ be the embedding matrix of the, $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_N$ the embedding matrices of the secondary languages, and $\boldsymbol{D}^{k,l}$ the dictionary between languages L_k and L_l. We induce a multilingual space $\boldsymbol{X}^m$, $\boldsymbol{Z}_1^m$, ... , $\boldsymbol{Z}_N^m$ in three main steps. First, the embeddings of each language are preprocessed by normalizing and whitening the embeddings, as described by Eqs. (2)-(6). Normalization consists of subsequently performing length normalization, mean

---

[2] Hubness is the phenomenon observed in a high-dimensional vector space where there are vectors, called hubs, which are the nearest neighbors to many vectors in the space.

[3] Note that $\boldsymbol{z}_j^{CL}$ being the nearest neighbor of $\boldsymbol{x}_i^{CL}$ does not imply that the inverse is also true.

[4] Due to the large search space, limiting the search space by truncating both vocabularies and their corresponding embedding matrices to the $C_{refinement}$ most frequent words results in better solutions and speeds up computation.

[5] The square root in the formulas is empirically motivated.

centering, and then again length normalization.

$$\text{ZCAwhiten}(\boldsymbol{W}) = \boldsymbol{W}(\boldsymbol{W}^\intercal \boldsymbol{W})^{-0.5} \quad (2)$$

$$\boldsymbol{X}^{'} = \text{normalize}(\boldsymbol{X}) \quad (3)$$

$$\boldsymbol{X}^{''} = \text{ZCAwhiten}(\boldsymbol{X}^{'}) \quad (4)$$

$$\boldsymbol{Z}_l^{'} = \text{normalize}(\boldsymbol{Z}_l) \quad (5)$$

$$\boldsymbol{Z}_l^{''} = \text{ZCAwhiten}(\boldsymbol{Z}_l^{'}) \quad (6)$$

After preprocessing, we rotate each secondary language L_l to a bilingual space between the hub language space and its own embedding space, as described by Eqs. (7)-(10). The calculations are analogous to the bilingual mapping procedure: the left and right singular vectors $\boldsymbol{U}_l$ and $\boldsymbol{V}_l$ of $\boldsymbol{X}^{''} \boldsymbol{D}^{k,l} \boldsymbol{Z}_l^{''\intercal}$ are the rotation matrices that project the preprocessed matrices $\boldsymbol{X}^{''}$ and $\boldsymbol{Z}_l^{''}$ to their bilingual space; this is formulated in Eqs. (7)-(8). The bilingual projection of $\boldsymbol{Z}_l^{''}$ can be reweighted by multiplying it with a given power q of the singular values matrix $\boldsymbol{S}_l$ of $\boldsymbol{X}^{''\intercal} \boldsymbol{D}^{0,l} \boldsymbol{Z}_l^{''}$, Eq. (9). Intuitively, the reweighting operation makes the dimensions that correlate better across languages more important. Next, we restore the variance information of $\boldsymbol{Z}_l^{'}$ by performing a dewhitening operation: we project back to the monolingual space, multiply with the inverse of the whitening matrix, and then project back to the bilingual space (Eq. (10)).[6]

$$\boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\intercal = \text{SVD}(\boldsymbol{X}^{''\intercal} \boldsymbol{D}^{0,l} \boldsymbol{Z}_l^{''}) \quad (7)$$

$$\boldsymbol{Z}_{l,bi(l)} = \boldsymbol{Z}_l^{''} \boldsymbol{V}_l \quad (8)$$

$$\boldsymbol{Z}_{l,bi(l)}^{'} = \boldsymbol{Z}_{l,bi(l)} \boldsymbol{S}_l^q \quad (9)$$

$$\boldsymbol{Z}_{l,bi(l)}^{''} = \boldsymbol{Z}_{l,bi(l)}^{'} \boldsymbol{V}_l^\intercal (\boldsymbol{Z}_l^{'\intercal} \boldsymbol{Z}_l^{'})^{0.5} \boldsymbol{V}_l \quad (10)$$

Finally, we project $\boldsymbol{Z}_{l,bi(l)}^{''}$ to the space of the hub language in Eq. (11)). The multilingual space for the hub language is simply the monolingual embedding space after preprocessing, see Eq. (12).

$$\boldsymbol{Z}_l^m = \boldsymbol{Z}_{l,bi(l)}^{''} \boldsymbol{U}_l^\intercal \quad (11)$$

$$\boldsymbol{X}^m = \boldsymbol{X}^{'} \quad (12)$$

For the bilingual case this formulation is equivalent to the symmetric mapping introduced in §3: one can easily verify that the dot products between the mapped spaces simplify to the same formula.

Incremental Hub Space (IHS). SHS enables language interactions only indirectly through the hub language. Ideally, a multilingual method should incorporate interdependencies between all languages. We hypothesize that, especially when mapping a language distant to the hub language, it is beneficial to incorporate the structural similarities with all other languages as a regularization mechanism to find a more robust mapping.

We therefore propose the incremental hub space (IHS) model. It incrementally expands the multilingual space $\boldsymbol{X}^m$ and takes into account all languages in the current multilingual space when adding a new language. First, we define a language order and initialize the space to the preprocessed embedding space of language L_0. Next, following the order, we gradually add new languages to the space: in each iteration we rotate the preprocessed embedding space $\boldsymbol{Z}_l^{''}$ of language l to the multilingual space by minimizing the dot product between embeddings of the translations between language l and all the languages in the multilingual space. The recipe to calculate the cross-lingual embedding $\boldsymbol{Z}_l^m$ is similar to the SHS model: the preprocessing and postprocessing steps are the same, but the projection matrices are calculated with Eq. (14) instead of Eq. (7) and conform with the new objective from Eq. (13). After convergence, $\boldsymbol{Z}_l^m$ is added to the multilingual space $\boldsymbol{X}^m$: $\boldsymbol{X}_l^m = \boldsymbol{Z}_l^m$.

$$\underset{\boldsymbol{W}_{xl}, \boldsymbol{W}_{zl}}{\arg\max} \sum_{k=0}^{l-1} tr(\boldsymbol{X}_k^m \boldsymbol{W}_{xl} (\boldsymbol{D}^{k,l} \boldsymbol{Z}_l^{''} \boldsymbol{W}_{zl})^\intercal) \quad (13)$$

$$\text{subject to } \boldsymbol{W}_{xl} \boldsymbol{W}_{xl}^\intercal = \boldsymbol{I}, \boldsymbol{W}_{zl} \boldsymbol{W}_{zl}^\intercal = \boldsymbol{I}$$

$$\boldsymbol{U}_l \boldsymbol{S}_l \boldsymbol{V}_l^\intercal = \text{SVD}(\boldsymbol{C}) \quad (14)$$

$$\boldsymbol{C} = (\boldsymbol{X}_0^m)^\intercal \boldsymbol{D}^{0,l} \boldsymbol{Z}_l^{''} || \ldots ||(\boldsymbol{X}_{l-1}^m)^\intercal \boldsymbol{D}^{l-1,l} \boldsymbol{Z}_l^{''}$$

where $||$ denotes concatenation along the row axis

In supervised settings this approach would be impractical as it requires bilingual dictionaries $\boldsymbol{D}^{k,l}$ for all language pairs k, l, and not only with the hub language. However, within an unsupervised framework this constraint is lifted.

5 Experimental Setup

Tasks and Datasets. The induced embeddings are evaluated in three tasks: bilingual lexicon induction (BLI), multilingual dependency parsing, and multilingual document classification. BLI is currently the most widely used method to evaluate bilingual embedding spaces. Although BLI performance is not the primary goal of our multilingual

---

[6] Note that the projection matrices that map from the bilingual to the monolingual spaces are given by the inverses of $\boldsymbol{U}_l$ and $\boldsymbol{V}_l$. Since the matrices are orthogonal their inverses are equal to their transposes.

embedding spaces, it provides a fast means to address the following questions: **1)** Is the incremental construction of multilingual embedding spaces indeed an effective regularization method? Is it still necessary to perform value dropping in this case?[7]; **2)** Is the reweighting of embedding spaces also beneficial for BLI in multilingual settings?; **3)** Does multilingual training improve bilingual lexicon induction performance? How do our multilingual models compare to each other and to the state-of-the-art unsupervised BLI methods?

We report *Precision@1 (P@1)* BLI performance on two standard BLI datasets. 1) DINUARTETXE is the extended version of Dinu et al. (2015)'s dataset, used by Artetxe et al. (2018a).[8] It consists of bilingual dictionaries for English-German, English-Italian, English-Spanish and English-Finnish. Monolingual embeddings are provided, based on the CBOW model trained on the WaCKy corpora for English, Italian and German (Baroni et al., 2009), the monolingual WMT Common Crawl corpus for Finnish, and the WMT News Crawl for Spanish (Bojar et al., 2015). The test dictionary sizes are between 1.869 and 1,993 word pairs for each language pair. As our methods are unsupervised, we do not use the provided training dictionaries. 2) EURMUSEWIKI is the dataset compiled from dictionaries for all combinations of the following European languages: English, German, Spanish, French, Italian, and Portuguese. The test set sizes range between 1,513 and 3,660 word pairs. We rely on publicly available monolingual fastText embeddings (Bojanowski et al., 2016).[9] All monolingual word embeddings are 300-dimensional and represent the 200k most frequent words as in prior work (Dinu et al., 2015; Conneau et al., 2018).

Multilingual dependency parsing and multilingual document classification tasks assess the embeddings w.r.t. their actual goal: enabling transfer learning across multiple languages. The word embeddings are used as feature vectors for classifiers in the respective downstream tasks. We address the following research questions: **4)** Is reweighting of embedding spaces also beneficial in downstream tasks?; **5)** How do our methods compare against supervised multilingual embedding models?

We rely on the evaluation platform of Ammar et al. (2016) for the downstream tasks[10]: the users submit their multilingual embeddings and obtain the final scores, which ensures that the classifiers we use are identical to the ones used in prior work (Ammar et al., 2016; Duong et al., 2017).

REUTERSMLDC is a multilingual document classification dataset covering seven languages: English, German, French, Italian, Spanish, Danish, and Swedish. The final performance is reported as the average accuracy across all languages. The respective training and test set consist of 7,000 and 13,058 documents. The dataset is well balanced in the number of documents per language.[11] The architecture of the document classifier is the average perceptron used by Klementiev et al. (2012).

MLPARSING is a multilingual dependency parsing dataset sampled from the Universal Dependencies 1.1 corpus (Agić et al., 2015)[12]. It contains 12 languages: English, German, French, Spanish, Italian, Bulgarian, Czech, Danish, Swedish, Greek, Finnish, and Hungarian. The respective training and test set contain 6,748 and 1,200 sentences. The test set contains 100 sentences for each language, while for the training set the number of sentences for a language ranges between 98 and 6,694. The parser used is the stack-LSTM parser by Dyer et al. (2015). The parser is not allowed to use any part-of-speech and morphology features, and keeps the input word embeddings fixed to isolate the effect of the evaluated embeddings on the parsing performance (Ammar et al., 2016). The reported scores are UAS scores averaged across languages.

For comparison with related work, we train 512-dimensional monolingual embeddings on the text collections used by Ammar et al. (2016) and Duong et al. (2017). The monolingual embeddings are again trained using fastText.

Training Setup. In all experiments, we set the following hyper-parameters to values that were used in prior research (Conneau et al., 2018; Artetxe et al., 2018a). When constructing the seed lexicon the 4,000 most frequent words of each language are considered ($C_{seed} = 4,000$), and during the refinement step the 20,000 most frequent words

---

[7] Value dropping significantly slows down training time and leads to non-deterministic outcomes. However, it has been shown to be crucial in the bilingual setting to obtain good results when mapping distant language pairs in previous work (Artetxe et al., 2018a).

[8] https://github.com/artetxem/vecmap/

[9] https://fasttext.cc/docs/en/pretrained-vectors.html

[10] https://github.com/wammar/multilingual-embeddings-eval-portal

[11] As the dataset is not publicly available this information was provided by the first author of Ammar et al. (2016).

[12] http://hdl.handle.net/11234/LRT-1478

Model	Drop	EN-DE	EN-IT	EN-ES	EN-FI	Avg
SHS	no	48.00	45.93	36.53	0.14	32.65
SHS	yes	47.51	45.60	36.33	31.92	**40.34**
IHS	no	47.93	45.93	36.07	31.04	**40.24**
IHS	yes	47.45	45.48	36.43	30.97	40.08

Table 1: Comparison of *P@1* scores for SHS and IHS models with and without value dropping (*Drop*) on the DINUARTETXE BLI dataset.

of each language are used ($C_{refinement} = 20,000$). When using value dropping, the keep probability p is initialized is 0.1, $N_{patience}$ is set to 50, and the stochastic multiplier is set to 2. Dictionaries are constructed symmetrically: from hub language(s) to the secondary language and from the secondary language to the hub language(s): during refinement each dictionary consists of $2 \times 20,000$ translation pairs. We use CSLS with $k = 10$ nearest neighbors following the setup of Conneau et al. (2018).

6 Results and Discussion

Experiment 1: Value Dropping. In the first experiment, we investigate if the expensive value dropping procedure is a necessary condition for mapping between distant language pairs in our multilingual framework. Table 1 provides results on the DINUARTETXE dataset for SHS and IHS models. For IHS we process the languages in the following order: English, German, Italian, Spanish, Finnish. When using value dropping we report the average and best results across five runs.

We observe that value dropping is crucial for SHS to succeed for English-Finnish. However, it is not necessary for IHS. This supports our hypothesis that mapping a language to a multilingual hub space serves a type of regularization that can substitute value dropping. As validated later, it is still important to avoid adding distant languages early with the incremental IHS procedure.[13,14]

Experiment 2: Comparative BLI Performance and Reweighting. In this experiment, we test if reweighting embedding spaces is beneficial for BLI in our multilingual setup, and also compare our methods against state-of-the-art BLI methods. Table 2 and Table 3 show the results for SHS and IHS with reweighting coefficients q of 0, 0.5 and 1 on DINUARTETXE and EURMUSEWIKI, respectively. We also include the state-of-the-art results of Artetxe et al. (2018a) and Chen and Cardie (2018) for reference. The EURMUSEWIKI benchmark evaluates BLI performance on all language pair combinations of its six languages and does this in both directions (EN-DE, DE-EN, EN-ES, ... IT-PT, PT-IT) yielding 28 $P@1$ scores per model. For clarity, we report the average $P@1$ scores per language as well as the global $P@1$ average. Following Experiment 1, all results for SHS are obtained using value dropping (again averaged across 5 different runs), while we do not use it with IHS. The SHS hub language is English, and the language orders for IHS are EN, DE, IT, ES, FI for DINUARTETXE, and EN, DE, ES, FR, IT, PT for EURMUSEWIKI.

The scores reveal that reweighting the embedding spaces is indeed still beneficial for BLI when mapping to a multilingual space. Both SHS and IHS obtain best results with the reweighting coefficient $q = 0.5$. When comparing SHS and IHS, we see that for language pairs involving English (the SHS hub language) SHS obtains slightly better results, but for the other language pairs IHS outperforms SHS slightly. This is no surprise as IHS by design incorporates dependencies between all languages when learning the projection matrices, though it is striking that mapping to a single hub language is still a strong BLI baseline. For both datasets IHS obtains BLI performance on par with the state-of-the-art: on DINUARTETXE, SHS and IHS ($q = 0.5$) obtain scores similar to (Artetxe et al., 2018a); on EURMUSEWIKI IHS ($q = 0.5$) slightly outperforms Chen and Cardie (2018) for all languages except Spanish. Although optimizing BLI performance is not the main goal of this work, these results verify the soundness of our methods.

Experiment 3: Language Order. Next, we investigate 1) the influence of the hub language choice for SHS, and 2) the impact of the language order for IHS. We run both SHS and IHS with reweighting 0.5 on DINUARTETXE. SHS is with value dropping (results are again averaged over 5 runs), and for IHS we do not use value dropping.

We find that the SHS model is sensitive to the hub language: the best average scores are obtained when using English (41.6%) or German (41.0%).

---

[13] For instance, when using IHS with a language order that starts with English and Finnish, value dropping still prevents bad performance for Finnish. However, this is not a problem in practice as the language order can be easily predetermined according to various language similarity heuristics.

[14] We further validated the robustness of our approach on other distant language pairs but moved this experiment to the appendix due to space constraints.

Model	q	EN-DE	EN-IT	EN-ES	EN-FI	All
Artetxe	2×0.5	48.13	**48.19**	37.33	**32.63**	**41.57**
SHS	0	47.51	45.60	36.33	31.92	40.34
IHS	0	47.93	45.93	36.07	31.04	40.24
SHS	0.5	**48.69**	47.67	37.51	32.40	**41.57**
IHS	0.5	48.60	47.73	37.53	31.74	41.40
SHS	1	47.77	47.91	37.00	31.82	41.13
IHS	1	48.00	**48.00**	**37.93**	31.46	41.35
IHS order	0.5	48.60	47.81	**38.24**	**33.22**	**41.96**

Table 2: BLI $P@1$ scores on DinuArtetxe: SHS and IHS are evaluated for different values of the reweighting parameter q. The state-of-the-art results (Artetxe et al., 2018a) are added as a reference. The result with the highest average score obtained when trying different language orders is in the bottom row.

Model	q	EN	DE	ES	FR	IT	PT	All
UME		79.57	70.46	**82.88**	82.01	80.69	80.13	79.29
SHS	0	80.04	68.24	80.95	80.10	78.71	77.96	77.67
IHS	0	79.61	69.52	82.35	81.26	80.00	79.02	78.63
SHS	0.5	**80.34**	70.16	82.15	81.65	80.31	79.78	79.07
IHS	0.5	79.91	**70.77**	82.68	**82.08**	**81.08**	**80.47**	**79.50**
SHS	1	79.61	69.59	81.53	81.10	79.74	79.47	78.51
IHS	1	79.05	69.99	81.63	81.23	80.13	79.68	78.62

Table 3: BLI $P@1$ scores on EurMuseWiki averaged per language: SHS and IHS are tested for different values of the reweighting parameter q. The results of Chen and Cardie (2018) (UME) are added as a reference.

With Italian, the average score drops to 40.4%, mainly due to worse performance on English and German. With Spanish, the average score further drops to 31.6%, as Spanish and Finnish completely fail to align even when using value dropping. With Finnish, EN-ES alignment becomes unstable, while $P@1$ for EN-DE and EN-IT drops 5.5% and 3.7% compared to the case with English as the hub. These results indicate that the hub language has to be chosen carefully to avoid instability issues.

For IHS, we evaluate all 120 order permutations on DinuArtetxe. The best performing order (EN-DE-ES-FI-IT) achieves an average accuracy of 41.96%, see the last row in Table 2. The full results, not reported due to space constraints[15] confirm our hypothesis that distant languages (Finnish) should be mapped at the end: when using Finnish as one of the first two languages performance drops significantly. EN-FI scores drop below 1% and the results for all other language pairs are also suboptimal.

Experiment 4: Downstream Tasks. In this experiment, we investigate the effect of reweighting input word embeddings on downstream model performance, and compare SHS and IHS to several supervised methods that use cross-lingual supervision. Table 4 reports the results for SHS and IHS with q set to 0 and 0.5 on the ReutersMLDC and MLParsing benchmarks,[16] along with the results from related work. For SHS the hub language is English and for IHS the language order is English, German, Spanish, Italian, French, Bulgarian, Czech, Danish, Finnish, Greek, Hungarian, and Swedish. We again use SHS with value dropping and IHS without it. The results in Table 4 are comparable: all methods were trained on the same text corpora (i.e., the collections of Ammar et al. (2016)), but our methods do not use parallel corpora nor bilingual dictionaries.

Model	q	Parsing	MLDC
Invariance (Huang et al., 2015)		59.80	91.10
MultiSkip (Luong et al., 2015)		57.70	90.40
MultiCluster (Ammar et al., 2016)		61.00	92.10
MultiCCA (Ammar et al., 2016)		58.70	92.10
(Duong et al., 2017)		61.20	90.80
SHS	0	63.48	92.59
IHS	0	**65.77**	**92.72**
SHS	0.5	62.23	92.63
IHS	0.5	63.42	92.56

Table 4: Results on the MLParsing (dependency parsing) and ReutersMLDC (document classification) benchmarks: SHS and IHS are compared with and without reweighting and we show the state-of-the-art results of supervised embedding mapping methods as a reference. The results for Invariance, MultiSkip, MultiCluster, MultiCCA are from (Ammar et al., 2016).

A first interesting result is that, contrary to the BLI task, reweighting the embeddings is not beneficial for multilingual dependency parsing and document classification. This can be explained by the fact that the reweighted embedding spaces are no longer isomorphic to the original monolingual embedding spaces, hence important patterns in the embedding space could be distorted. Further, we notice that both SHS and IHS improve over the best reported results on the ReutersMLDC and MLParsing benchmarks. This result is surprising given that all the reported baselines require supervision to induce the multilingual embedding spaces. Further, we again find that the best results are obtained with IHS, most notably for dependency parsing for which the difference in UAS scores between

---

[15] The full results can be found in Appendix A.2.

[16] Since the languages covered in MLParsing is a superset of the languages in ReutersMLDC, we use the same multilingual embedding space for both tasks.

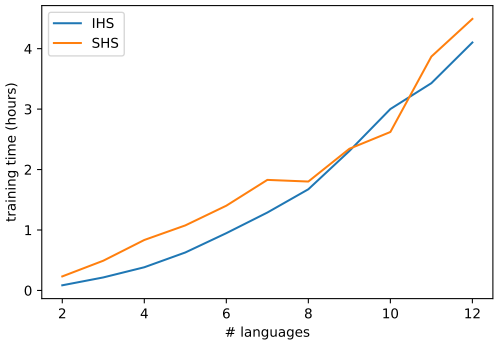


Figure 1: Training times for SHS and IHS for an increasing number of languages.

the best IHS and SHS models is 2.29%.

Experiment 5: Time Complexity. In this experiment, we study how training time of SHS and IHS behaves as a function of the number of languages that are mapped. The singular value decompositions are more expensive for IHS as the matrices grow linearly with the number of languages (see Eq. (14)) whereas for SHS they are constant. On the other hand, IHS does not require the use of value dropping. Figure 1 plots training time for IHS and SHS conditioned on the number of languages. The estimates are based on training on a single Nvidia Titan Xp GPU. We find that SHS with value dropping and IHS without value dropping have similar training times, IHS being a bit more efficient when mapping 7 languages or less. Mapping 12 languages takes approximately four hours for both methods.

7 Conclusion

We proposed two novel methods for learning multilingual word embeddings (MWEs) without any cross-lingual supervision. The better-performing incremental hub space model (IHS) is the first unsupervised MWE method that combines three desirable properties: 1) It incorporates interdependencies between all targeted languages; 2) It works for distant language pairs; and 3) It is both deterministic and robust, that is, it does not produce degenerate solutions. Our evaluation on standard benchmarks has proven that the IHS method induces multilingual word embeddings that are competitive with the state of the art in bilingual lexicon induction. Moreover, we have shown that IHS outperforms even supervised models on downstream tasks of multilingual dependency parsing and document classification, and this anomaly requires further investigation in future work. Furthermore, we looked at the influence of reweighting the dimensions of the embedding spaces according to their cross-correlations with the hub language space(s) and found that, while it improves performance for the BLI task, it is harmful to downstream cross-lingual transfer tasks. These empirical observations stress the requirement to include comprehensive evaluation protocols for cross-lingual word embedding models in future research.

Acknowledgments

We thank the reviewers for their insightful comments. IV would like to thank Goran Glavaš and Anders Søgaard for interesting discussions, and the ERC Consolidator Grant LEXICAL (no 648909) for the support. GH, BV and MFM would like to thank Stijn Jaques, Evelyn Reynders and Evelien Verbaenen for the fruitful collaboration and Flanders Innovation & Entrepreneurship (VLAIO) for funding TELLMI within the ITEA 3 project PAPUD.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richrd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trrup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Hctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal Dependencies 1.1.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. *CoRR abs/1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning Principled Bilingual Mappings of Word Embeddings while Preserving Monolingual Invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning Bilingual Word Embeddings with (Almost) no Bilingual Data. In *Proceedings of the*

55th Annual Meeting of the Association for Computational Linguistics (ACL), pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A Robust Self-learning Method for Fully Unsupervised Cross-lingual Mappings of Word Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018c. Unsupervised Statistical Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised Neural Machine Translation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Anthony Bell and Terrence Sejnowski. 1997. The 'Independent Components' of Natural Scenes are Edge Filters. *Vision Research*, pages 3327–3338.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *CoRR abs/1607.04606*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the 10th Workshop on Statistical Machine Translation (WMT)*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A Distribution-based Model to Learn Bilingual Word Embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*.

Xilun Chen and Claire Cardie. 2018. Unsupervised Multilingual Word Embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 261–270.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–14.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-shot Learning by Mitigating the Hubness Problem. In *Proceedings of the International Conference on Learning Representations (ICLR), workshop track*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual Training of Crosslingual Word Embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 894–904.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 334–343.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 462–471.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 748–756.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2017. Bilingual Lexicon Induction by Learning to Combine Word-Level and Character-Level Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1085–1095.

Kejun Huang, Matt Gardner, Evangelos Papalexakis, Christos Faloutsos, Nikos Sidiropoulos, Tom Mitchell, Partha P Talukdar, and Xiao Fu. 2015. Translation Invariant Word Embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1084–1088.

David Kamholz, Jonathan Pool, and Susan M Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 3145–3150.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1459–1474.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised Machine Translation Using Monolingual Corpora Only. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–12.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 270–280.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *Proceedings Workshop on Vector Modeling for NLP, NAACL 2015*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation. In *CoRR, abs/1309.4168*.

Milo Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in Space: Popular Nearest Neighbors in High-dimensional Data. *Journal of Machine Learning Research*, 11:2487–2531.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2018. A Survey of Cross-lingual Embedding Models. *Journal of Artificial Intelligence Research (JAIR)*.

Samuel Smith, David Turban, Steven Hamblin, and Nils Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–10.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the Limitations of Unsupervised Bilingual Dictionary Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 247–257.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and Cross-Lingual Information Retrieval Models Based on (Bilingual) Word Embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 363–372.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual Distributed Word Representations from Document-Aligned Comparable Data. *Journal of Artifical Intelligence Research*, 55:953–994.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial Training for Unsupervised Bilingual Lexicon Induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1959–1970.

A Appendices

A.1 Experiment with Distant Language Pairs

In this section, we report an additional bilingual lexicon induction experiment that further supports our claim that the IHS model is still applicable when mapping languages with different characteristics, we performed an additional bilingual lexicon induction experiment on the following language pairs: Dutch-Turkish, Spanish-Hungarian, and Finnish-Bulgarian. We induced multilingual spaces with the IHS model and the model of Chen and Cardie (2018) using the publicly available monolingual embeddings trained with fastText. IHS is run with reweighting ($q = 0.5$), without value dropping, and with the following language order Dutch, Spanish, Bulgarian, Finnish, Hungarian, Turkish. The model of Chen and Cardie (2018)[17] is run with Dutch as the target language and with the recommended hyper-parameters. For evaluation, we obtained dictionaries for each language pair using Panlex (Kamholz et al., 2014).

The results are reported in Table 5. We find average BLI accuracy scores of 28.24% (IHS) and 28.20% (Chen and Cardie, 2018). The fact that both models are robust to distant languages without using value dropping further supports our hypothesis that mapping multiple languages simultaneously is an effective regularization mechanism.

A.2 Results Experiment 3

In this section, we report all the results of Experiment 3 of the paper. In Table 6 we report the results for the SHS model with different hub languages and in Table 7 we report the performance for IHS

---

[17] `https://github.com/ccsasuke/umwe`

Model	NL-TR	ES-HU	FI-BG	Avg
IHS	22.67	30.21	31.84	28.24
UME	21.51	31.26	31.84	28.20

Table 5: Precision@1 scores for a bilingual lexicon experiment on three distant language pairs.

Hub language	EN-DE	EN-IT	EN-ES	EN-FI	Average
EN	48.69	47.67	37.51	32.40	41.57
DE	47.75	46.65	38.15	31.47	41.01
IT	45.88	47.15	37.88	30.65	40.39
ES	43.77	45.27	36.63	0.57	31.56
FI	43.15	43.92	6.71	33.20	31.74

Table 6: Influence of the hub language of the SHS model on precision@1 scores evaluated on the DINUARTETXE BLI dataset.

with all possible language orders (sorted by precision@1). Both the SHS and IHS models are run with reweighting ($q = 0.5$), SHS is run with value dropping and IHS is run without value dropping.

Order	EN-DE	EN-ES	EN-FI	EN-IT	Avg
EN DE ES FI IT	48.60	38.20	33.22	47.80	41.96
EN DE FI ES IT	48.60	37.73	33.50	47.93	41.94
DE IT EN FI ES	48.40	38.07	33.22	47.93	41.91
EN DE ES IT FI	48.60	38.20	32.72	47.60	41.78
EN IT DE ES FI	48.33	38.73	32.51	47.53	41.78
EN IT ES DE FI	48.07	38.47	33.01	47.53	41.77
EN DE IT FI ES	48.60	37.53	33.22	47.73	41.77
EN DE FI IT ES	48.60	37.13	33.50	47.67	41.73
EN IT DE FI ES	48.33	38.20	32.79	47.53	41.71
DE IT EN ES FI	48.40	38.00	32.51	47.93	41.71
DE EN FI IT ES	48.40	37.60	33.29	47.53	41.71
DE EN FI ES IT	48.40	37.40	33.29	47.67	41.69
DE EN IT FI ES	48.40	37.73	33.22	47.27	41.66
DE EN ES IT FI	48.40	38.20	32.65	47.33	41.65
EN IT FI DE ES	48.53	37.87	32.65	47.53	41.65
DE IT ES EN FI	48.27	38.07	32.30	47.73	41.59
DE EN ES FI IT	48.40	38.20	32.37	47.40	41.59
DE IT FI ES EN	47.80	37.93	33.22	47.40	41.59
DE EN IT ES FI	48.40	37.53	33.08	47.27	41.57
EN IT FI ES DE	47.80	38.07	32.65	47.53	41.51
EN DE IT ES FI	48.60	37.67	32.02	47.73	41.51
DE ES IT FI EN	47.27	38.33	32.87	47.40	41.47
IT EN DE FI ES	47.40	38.47	32.65	47.07	41.40
DE ES FI IT EN	47.33	38.53	32.30	47.33	41.37
DE ES IT EN FI	47.27	38.40	31.88	47.80	41.34
DE ES EN FI IT	47.67	37.87	32.02	47.67	41.31
DE IT ES FI EN	47.40	37.87	32.65	47.27	41.30
EN ES DE FI IT	47.93	37.53	32.09	47.53	41.27
EN IT ES FI DE	47.87	38.47	31.18	47.53	41.26
DE IT FI EN ES	47.60	38.07	32.37	46.87	41.23
DE ES EN IT FI	47.67	37.87	31.67	47.53	41.19
IT DE EN FI ES	47.00	38.00	33.15	46.53	41.17
IT ES DE EN FI	47.53	38.47	32.37	46.27	41.16
DE ES FI EN IT	47.47	37.60	32.44	46.87	41.10
IT EN DE ES FI	47.40	38.40	31.39	47.07	41.07
IT EN FI DE ES	47.00	38.27	31.88	47.07	41.06
EN ES DE IT FI	47.93	37.53	30.90	47.60	40.99
ES DE IT EN FI	46.67	38.27	31.95	47.07	40.99
IT EN ES DE FI	47.20	37.93	31.74	47.07	40.99
ES IT DE FI EN	47.20	37.40	32.23	46.73	40.89
IT EN FI ES DE	46.47	38.00	31.88	47.07	40.86
EN ES IT FI DE	47.73	37.53	30.76	47.20	40.81
IT DE EN ES FI	47.00	37.93	31.74	46.53	40.80
EN ES IT DE FI	47.33	37.53	31.04	47.20	40.78
IT DE ES EN FI	46.67	38.07	32.02	46.33	40.77
ES IT DE EN FI	46.60	37.73	31.46	47.27	40.77
IT DE FI ES EN	46.80	37.73	32.51	45.87	40.73
ES DE IT FI EN	46.47	37.27	32.23	46.87	40.71
ES DE EN FI IT	46.67	37.67	31.32	47.13	40.70
IT EN ES FI DE	46.40	37.93	31.11	47.07	40.63
ES DE EN IT FI	46.67	37.67	30.69	47.27	40.58
IT DE ES FI EN	46.53	37.40	32.02	46.20	40.54
IT ES DE FI EN	46.53	38.27	31.32	45.93	40.51
IT DE FI EN ES	46.27	37.60	32.02	45.93	40.46
ES IT EN DE FI	46.20	37.20	31.18	46.93	40.38
ES EN IT FI DE	46.87	36.80	30.83	46.67	40.29
ES EN DE FI IT	46.40	36.80	31.32	46.53	40.26
ES EN DE IT FI	46.40	36.80	31.32	46.47	40.25
ES EN IT DE FI	46.13	36.80	31.11	46.67	40.18
ES IT EN FI DE	45.67	37.20	30.55	46.93	40.09
ES DE FI IT EN	45.67	36.80	30.83	46.87	40.04
IT ES EN DE FI	45.33	37.60	30.27	46.20	39.85
IT ES EN FI DE	45.20	37.60	30.34	46.20	39.84
ES DE FI EN IT	45.20	37.07	30.27	46.60	39.79
EN ES FI DE IT	47.27	37.53	0.21	47.33	33.09
DE FI IT EN ES	48.07	37.33	0.07	46.80	33.07
EN FI DE IT ES	46.93	37.40	0.14	47.47	32.99
EN FI DE ES IT	46.93	37.73	0.14	47.07	32.97
EN ES FI IT DE	47.33	37.53	0.21	46.80	32.97
DE FI IT ES EN	47.20	37.73	0.07	46.73	32.93
DE FI ES EN IT	47.47	37.40	0.07	46.40	32.84
DE FI ES IT EN	47.80	37.20	0.07	46.07	32.79
DE FI EN IT ES	46.73	37.27	0.14	46.60	32.69
DE FI EN ES IT	46.73	37.40	0.14	46.33	32.65
IT FI DE ES EN	46.13	37.67	0.14	46.20	32.54
ES EN FI DE IT	46.60	36.80	0.35	46.20	32.49
ES FI DE IT EN	44.67	38.13	0.07	46.93	32.45
ES EN FI IT DE	46.60	36.80	0.35	45.47	32.31
IT FI ES DE EN	45.60	36.93	0.14	46.47	32.29
ES FI IT DE EN	45.60	36.93	0.00	46.53	32.27
IT ES FI DE EN	45.00	37.60	0.21	45.67	32.12
IT FI DE EN ES	45.80	37.73	0.14	44.80	32.12
EN FI IT DE ES	45.13	36.60	0.14	46.13	32.00
EN FI IT ES DE	45.07	36.33	0.14	46.13	31.92
ES FI DE EN IT	44.07	37.13	0.07	46.07	31.84
ES IT FI DE EN	44.13	36.60	0.07	46.33	31.78
ES FI IT EN DE	44.80	36.73	0.00	45.47	31.75
IT FI EN ES DE	45.07	36.20	0.14	45.47	31.72
IT FI ES EN DE	44.60	36.53	0.14	45.33	31.65
IT FI EN DE ES	44.93	35.80	0.14	45.47	31.59
IT ES FI EN DE	44.20	36.93	0.21	44.93	31.57
ES IT FI EN DE	43.33	36.13	0.07	45.93	31.37
FI DE IT EN ES	42.93	37.13	0.42	44.73	31.30
FI DE IT ES EN	43.20	36.67	0.28	45.00	31.29
FI IT DE EN ES	44.40	36.67	0.07	43.67	31.20
ES FI EN DE IT	43.73	35.27	0.14	45.00	31.04
ES FI EN IT DE	43.60	35.27	0.14	44.67	30.92

Order	EN-DE	EN-ES	EN-FI	EN-IT	Avg
FI DE ES IT EN	42.27	35.07	0.14	45.33	30.70
FI IT ES DE EN	44.07	35.07	0.07	43.33	30.64
FI DE EN IT ES	41.27	36.40	0.14	44.33	30.54
FI IT DE ES EN	43.33	35.40	0.07	43.13	30.48
FI ES IT DE EN	43.60	33.60	0.21	44.40	30.45
FI DE EN ES IT	41.27	35.27	0.14	44.60	30.32
FI IT ES EN DE	43.07	34.87	0.14	42.47	30.14
FI ES DE IT EN	42.13	32.67	0.14	45.00	29.99
FI DE ES EN IT	42.73	31.47	0.14	45.47	29.95
FI EN IT DE ES	42.60	32.67	0.28	41.53	29.27
FI ES IT EN DE	42.07	31.93	0.21	42.27	29.12
FI ES DE EN IT	42.00	29.13	0.14	44.60	28.97
FI EN IT ES DE	40.80	31.33	0.28	41.53	28.49
FI EN DE IT ES	35.27	30.33	0.28	40.27	26.54
FI EN DE ES IT	35.27	29.40	0.28	40.07	26.26
EN FI ES DE IT	46.53	0.73	0.14	47.13	23.63
EN FI ES IT DE	47.00	0.73	0.14	44.87	23.19
FI EN ES DE IT	17.27	0.07	0.28	20.33	9.49
FI IT EN DE ES	9.13	6.60	0.00	2.53	4.57
FI IT EN ES DE	5.20	3.93	0.00	2.53	2.92
FI ES EN IT DE	0.33	0.13	0.35	0.67	0.37
FI ES EN DE IT	0.27	0.13	0.35	0.47	0.31
FI EN ES IT DE	0.33	0.07	0.28	0.13	0.20

Table 7: Influence of language order of the IHS model on precision@1 scores evaluated on the DINUARTETXE BLI dataset.

Curriculum Learning for Domain Adaptation in Neural Machine Translation

Xuan Zhang[1], Pamela Shapiro[1], Gaurav Kumar[1], Paul McNamee[1], Marine Carpuat[2], Kevin Duh[1]

[1]Johns Hopkins University
[2]University of Maryland

`{xuanzhang, pshapiro, mcnamee}@jhu.edu, marine@cs.umd.edu, {gkumar, kevinduh}@cs.jhu.edu`

Abstract

We introduce a curriculum learning approach to adapt generic neural machine translation models to a specific domain. Samples are grouped by their similarities to the domain of interest and each group is fed to the training algorithm with a particular schedule. This approach is simple to implement on top of any neural framework or architecture, and consistently outperforms both unadapted and adapted baselines in experiments with two distinct domains and two language pairs.

1 Introduction

Neural machine translation (NMT) performance often drops when training and test domains do not match and when in-domain training data is scarce (Koehn and Knowles, 2017). Tailoring the NMT system to each domain could improve performance, but unfortunately high-quality parallel data does not exist for all domains. Domain adaptation techniques address this problem by exploiting diverse data sources to improve in-domain translation, including *general domain* data that does not match the domain of interest, and *unlabeled domain* data whose domain is unknown (e.g. webcrawl like Paracrawl).

One approach to exploit unlabeled-domain bitext is to apply *data selection* techniques (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013) to find bitext that are similar to in-domain data. This selected data can additionally be combined with in-domain bitext and trained in a continued training framework, as shown in Figure 1. Continued training or fine-tuning (Luong et al., 2015; Freitag and Al-Onaizan, 2016; Chu et al., 2017) is an adaptation technique where a model is first trained on the large general domain data, then used as initialization of a new model which is further trained on in-domain bitext. In our framework, the selected samples are concatenated with in-domain data, then used for continued training. This effectively increases the in-domain training size with "pseudo" in-domain samples, and is helpful in continued training (Koehn et al., 2018).

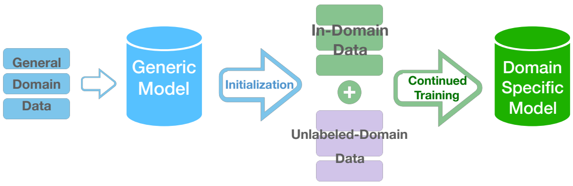


Figure 1: Workflow of our domain adaptation system.

A challenge with employing data selection in continued training is that there exists no clear-cut way to define whether a sample is sufficiently similar to in-domain data to be included. In practice, one has to define a threshold based on similarity scores, and even so the continued training algorithm may be faced with samples of diverse similarities. We introduce a new domain adaptation technique that addresses this challenge.

Inspired by curriculum learning (Bengio et al., 2009), we use the similarity scores given by data selection to rearrange the order of training samples, such that *more similar examples are seen earlier and more frequently during training*. To the best of our knowledge, this is the first work applying curriculum learning to domain adaptation.

We demonstrate the effectiveness of our approach on TED Talks and patent abstracts for German-English and Russian-English pairs, using two distinct data selection methods, Moore-Lewis method (Moore and Lewis, 2010) and cynical data selection (Axelrod, 2017). Results show that our approach consistently outperforms standard continued training, with up to 3.22 BLEU improvement. Our S^4 error analysis (Irvine et al., 2013) reveal that this approach reduces a reasonable number of SENSE and SCORE errors.

Proceedings of NAACL-HLT 2019, pages 1903–1915
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Curriculum Learning for Adaptation

Weinshall and Cohen (2018) provide guidelines for curriculum learning: “A practical curriculum learning method should address two main questions: how to rank the training examples, and how to modify the sampling procedure based on this ranking.” For domain adaptation we choose to estimate the difficulty of a training sample based on its distance to the in-domain data, which can be quantified by existing data selection methods (Section 2.1). For the sampling procedure, we adopt a probabilistic curriculum training (CL) strategy that takes advantage of the spirit of curriculum learning in a nondeterministic fashion without discarding the good practice of original standard training policy, like bucketing and mini-batching.

2.1 Domain Similarity Scoring

We adopt similarity metrics from prior work on data selection to score examples for curriculum learning. Let I be an in-domain corpus, and N be a unlabeled-domain data set. Data selection models rank sentences in N according to a domain similarity measure with respect to I, and choose top n samples from N by a cut-off threshold for further training purpose. We examine two data selection methods, Moore-Lewis method (Moore and Lewis, 2010) and cynical data selection (Axelrod, 2017).

Moore-Lewis Method Each sentence s in N is assigned a *cross-entropy difference score*,

$$H_I(s) - H_N(s), \qquad (1)$$

where $H_I(s)$ is the per-word cross-entropy of s according to a language model trained on I, and $H_N(s)$ is the per-word cross-entropy of s according to a language model trained on a random sample of N with roughly the same size as I. A lower cross-entropy difference indicates that s is more like the in-domain data and less like the unlabeled-domain data.

Cynical Data Selection Iteratively select sentence s from N to construct a training corpus that would approximately model I. At each iteration, each sentence is scored by the expected cross-entropy change from adding it to the already selected subset of N. The selected sentence is the one which most decreases H_n, the cross-entropy between previously selected n-sentence corpus and I.

2.2 Curriculum Learning Training Strategy

We identify two general types of curriculum learning strategy. The *deterministic* curriculum (c.f. Kocmi and Bojar (2017)) trains on a fixed order of samples based on their scores (e.g. “easy-to-hard” or “more similar to less”). While simple to motivate, this may not always perform well because neural methods benefit from randomization in the minibatches and multiple epochs. In contrast, the *probabilistic* curriculum (Bengio et al., 2009) works by dividing the training procedure into distinct phases. Each phase creates a random sample from the entire pool of data, but earlier phases sample the “easier” or “more similar” sentence with higher probability.. Since each phase can be viewed as creating a new training dataset, all the well-tested tricks of the trade for neural network optimization can be employed.

In this paper, we use the same probabilistic curriculum strategy and code base[1] as Zhang et al. (2018). The main difference here is the application to domain adaptation. The proposed strategy is summarized as follows:

- Sentences are first ranked by similarity scores and then distributed evenly into shards, such that each shard contains samples with similar similarity criteria values.
- The training process is segmented into consecutive *phases*, where only a subset of shards are available for training.
- During the first phase, only the easiest shard is presented. When moving to the next phase, the training set will be increased by adding the second easiest shard into it, and so on. Easy shards are those that are more similar to the in-domain data, as quantified by either Moore-Lewis or Cynical Data Selection.
- The presentation order of samples is not deterministic. (1) Shards within one curriculum phase are shuffled, so they are not necessarily visited by the order of similarity level during this phase. (2) Samples within one shard are bucketed by length and batches are drawn randomly from buckets.

---

[1] `https://github.com/kevinduh/sockeye-recipes/tree/master/egs/curriculum`

3 Experiments and Results

We evaluate on four domain adaptation tasks. The code base is provided to ensure reproducibility.[2]

3.1 Data and Setup

General Domain Data We have two general domain datasets, Russian-English (ru) and German-English (de). Both are a concatenation of OpenSubtitles2018 (Lison and Tiedemann, 2016) and WMT 2017 (Bojar et al., 2017), which contains data from several domains, e.g. parliamentary proceedings (Europarl, UN Parallel Corpus), political/economic news (news commentary, Rapid corpus), and web-crawled parallel corpus (Common Crawl, Yandex, Wikipedia titles). We performed sentence length filtering (up to 80 words) after tokenization, ending up with 28 million sentence pairs for German and 51 million sentence pairs for Russian.

In-domain Data We evaluate our proposed methods on two distinct domains per language pair:

- TED talks: data-split from Duh (2018).
- Patents: from the World International Property Organization COPPA-V2 dataset (Junczys-Dowmunt et al., 2016).

We randomly sample 15k parallel sentences from the original corpora as our in-domain bitext.[3] We also have around 2k sentences of development and test data for TED and 3k for patent.

Unlabeled-domain Data For additional unlabeled-domain data, we use web-crawled bitext from the Paracrawl project.[4] We filter the data using the Zipporah cleaning tool (Xu and Koehn, 2017), with a threshold score of 1. After filtering, we have around 13.6 million Paracrawl sentences available for German-English and 3.7 million Paracrawl sentences available for Russian-English. Using different data selection methods, we include up to the 4096k and 2048k sentence-pairs for our German and Russian experiments, respectively.

Data Preprocessing All datasets are tokenized using the Moses (Koehn et al., 2007) tokenizer. We learn byte pair encoding (BPE) segmentation models (Sennrich et al., 2016) from general domain data. The BPE models are trained separately for each language, and the number of BPE symbols is set to 30k. We then apply the BPE models to in-domain and Paracrawl data, so that the parameters of the generic model can be applied as an initialization for continued training. Once we have a converged generic NMT model, which is very expensive to train, we can adapt it to different domains, without building up a new vocabulary and retraining the model.

NMT Setup Our NMT models are developed in Sockeye[5] (Hieber et al., 2017). The generic model and continued training model are trained with the same hyperparameters. We use the seq2seq attention architecture (Bahdanau et al., 2015) with 2 LSTM layers for both encoder and decoder, and 512 hidden nodes in each layer. The word embedding size is also set to 512. Our models apply Adam (Kingma and Ba, 2014) as the optimizer, with an initial learning rate 0.0003. The learning rate is multiplied by 0.7 whenever validation perplexity does not surpass the previous best in 8 checkpoints.[6] We use minibatches of 4096 words. Training stops when the perplexity on the development set has not improved for 20 checkpoints (1000 updates/batches per checkpoint).

Domain Similarity Scoring Setup To get similarity scores, we build 5-gram language models on the source side[7] with modified Kneser-Ney smoothing using KenLM (Heafield, 2011).

Curriculum Learning Setup The number of batches in each curriculum phase is set to 1000. We split the training data into 40 shards[8], with all the 15k in-domain data in the first shard, and Paracrawl data split into the remaining 39 shards.

3.2 Experimental Comparison

Our goal is to empirically test whether the proposed curriculum learning method improves translation quality in the continued training setup of

---

[2] `https://github.com/kevinduh/sockeye-recipes/tree/master/egs/curriculum`

[3] Appendix A explains our choice of 15k in detail.

[4] `https://www.paracrawl.eu/`

[5] `github.com/awslabs/sockeye`

[6] The Adam optimizer for continued training model is initialized without reloading from the trained generic model.

[7] Appendix D also shows the effect of using language models built from target side and both sides.

[8] After experimenting with various values from 5 to 100 (Appendix B), we found best performance can be achieved at 40 shards.

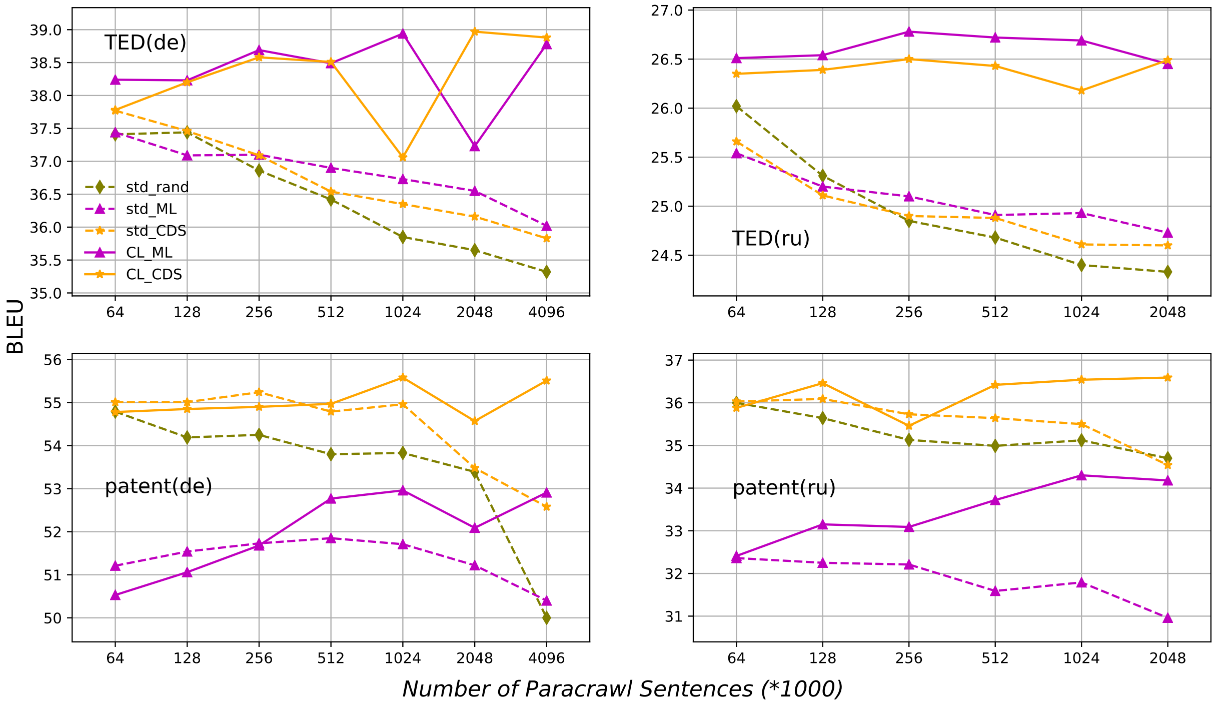


Figure 2: BLEU of adapted models using a concatenation of in-domain and varying amounts of Paracrawl data.

Figure 1. We compare two approaches to continued training: (1) the standard approach reads batches of in-domain and selected Paracrawl in random order; (2) the proposed curriculum learning approach reads these batches according to a schedule. We run the comparison with two data selection methods, leading to four systems:

- **std_ML**: standard continued training with Moore-Lewis scores
- **CL_ML**: curriculum learning approach to continued training with Moore-Lewis scores
- **std_CDS**: standard continued training with scores from Cynical Data Selection
- **CL_CDS**: curriculum learning approach to continued training with scores from Cynical Data Selection

For reference, we show results of the generic model (**GEN**), the model trained from scratch with in-domain data (**IN**), the model continued trained on in-domain data only (**IN_CT**), and a standard continued training model using a random subset (rather than ML or CDS scores) of the concatenated in-domain and Paracrawl data (**std_rand**).

3.3 Results

Table 1 summarizes the key results, where we continue train on 15k in-domain samples and 4096k Paracrawl samples (for de) or 2048k Paracrawl samples (for ru):

	TED(de)	TED(ru)	patent(de)	patent(ru)
GEN	34.59	23.40	35.95	23.41
IN	2.53	1.76	12.09	16.81
IN_CT	36.16	25.04	54.70	35.61
std_rand	35.32	24.33	50.00	34.70
std_ML	36.02	24.73	50.40	30.96
CL_ML	38.78	26.45	52.91	34.18
Δ_ML	2.76	1.72	2.51	3.22
std_CDS	35.83	24.60	52.58	34.54
CL_CDS	**38.88**	**26.49**	**55.51**	**36.59**
Δ_CDS	3.05	1.89	2.93	2.05

Table 1: BLEU of unadapted & adapted models. Δ shows improvement of **CL** over **std**.

- The baseline BLEU scores confirm the need for domain adaptation. Using only the 15k in-domain samples alone (IN) is not sufficient to train a strong domain specific model, yielding BLEU scores as low as 2.53 on TED(de) and 1.76 on TED(ru). The model trained with a large amount of general domain data (GEN) is a stronger baseline, with BLEU scores of 34.59 and 23.40.
- Standard continued training is not robust to samples that are noisy and less similar to in-domain. As expected, continued training on in-domain data (IN_CT) improves BLEU significantly, by up to 18.74 BLEU on patent(de). However, when adding Paracrawl data, the standard continued training strategy (std_rand, std_ML, std_CDS) consistently performs worse than IN_CT.
- Curriculum learning consistently improves BLEU score. Ranking examples using

Moore-Lewis (CL_ML) and Cynical Data Selection (CL_CDS) improve BLEU over their baselines (std_ML and std_CDS) by up to 3.22 BLEU points.

As an additional experiment, we report results on different amounts of Paracrawl data. Figure 2 shows how the curriculum uses increasing amounts of Paracrawl better than standard continued training. Standard continued training model hurts BLEU when too much Paracrawl data is added: for TED(de), there's a 1.94 BLEU drop when increasing CDS data from 64k to 4096k, and for patent(de), the decrease is 2.43 BLEU. By contrast, the curriculum learning models achieve a BLEU score that is as good or better as the initial model, even after being trained on the most dissimilar examples. This trend is clearest on the patent(ru) CL_ML model, where the BLEU score consistently rises from 32.41 to 34.18.

The method used to score domain relevance has a different impact on the TED domain (top plots) and on the patent domain (bottom plots). On the patent domain, which is more distant from Paracrawl, CDS significantly outperforms ML. Replacing ML with CDS improve BLEU from 2.18 to 4.05 BLEU points for standard models and 2.20 to 4.25 BLEU points for curriculum learning models. Interestingly, for patents, the Moore-Lewis method does not beat the random selection, even when curriculum learning is applied. For example, at 64k selected sentences for patent(de), std_rand gets 4.26 higher BLEU scores than CL_ML. By contrast on the TED domain, which is closer to Paracrawl, the Moore-Lewis method slightly outperforms cynical data selection. Due to these differences, we suggest trying different data selection methods with curriculum learning on new tasks; a potential direction for future work may be a curriculum that considers multiple similarity scores jointly.

4 Analysis

4.1 Comparison of Curriculum Strategies

We compare our approach to other curriculum strategies. *CL_reverse* reverses the presenting order of the shards, so that shards containing less similar examples will be visited first, *CL_scrambled* is a model that adopts the same training schedule as *CL*, but no data selection method and ranking is involved here — Paracrawl data are evenly split and randomly assigned to shards; *CL_noshuffle* is another curriculum learning model that does not shuffle shards in each curriculum phase.

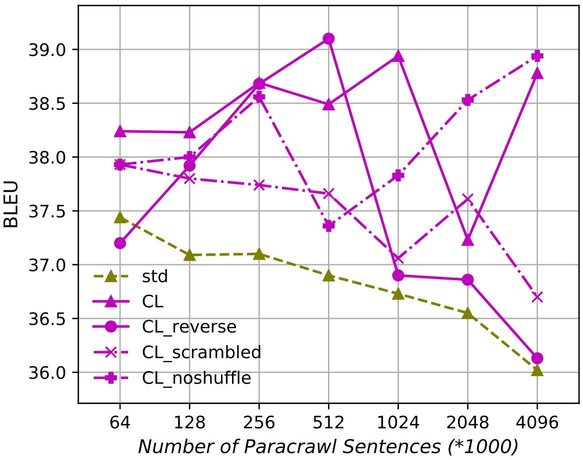


Figure 3: Comparison of various curriculum strategies on German-English TED corpora, where Moore-Lewis method is applied. [9]

Results from Figure 3 show that CL outperforms CL_reverse and CL_noshuffle for 5 out of 7 cases and outperforms CL_scrambled in 6 out of 7 cases. This suggests that it is beneficial to train on examples that are closest to in-domain first and to use a probabilistic curriculum. Analyzing the detailed difference between CL and CL_reverse would be interesting future work. One potential hypothesis why CL might help is that it first trains on a low-entropy subset of the data before moving on to the whole training set, which may have regularization effects.

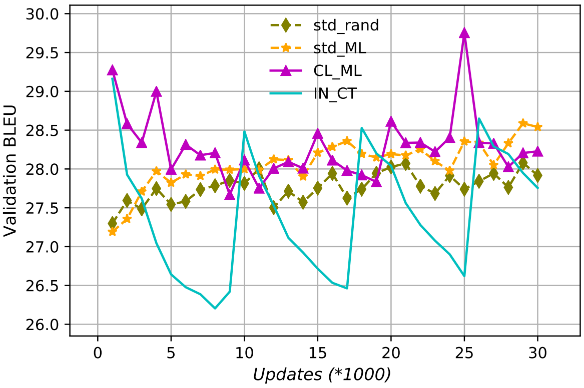


Figure 4: Learning curves for German-English TED NMTs. Except for IN_CT, the other three models all continued trained on the concatenation of in-domain and 1024k Paracrawl data.

---

[9] Each point represents a model trained to convergence on the fixed amount of in-domain and ParaCrawl data whose amount is specified by the x-axis.

4.2 Learning Curves

Learning curves (Figure 4) further illustrate the advantage of our method. Continued training on in-domain data only starts from a strong initialization (thanks to pre-training on large general domain data) but heavily oscillates over training without reaching the initial performance. This behavior may be due to the sparsity of the TED data: the small randomly sampled training set may not represent the development and test data well. Std_ML shows opposite behavior to IN_CT: it starts from a lower initial performance, and then gradually improves to a level comparable to IN_CT. Std_rand behaves similarly to std_ML—in other words, uniformly sampling from Paracrawl drags the initial performance down without helping with the final performance.

Compared to all above, the curriculum learning models start from a high initial performance, suffer much less oscillation than IN_CT, and gradually achieve the highest performance.[10]

4.3 Impact of Curriculum Learning on Lexical Choice: S^4 Analysis

How do translations improve when using curriculum learning? We characterize the impact of curriculum learning on lexical translation errors using the S^4 taxonomy of domain change errors introduced by Irvine et al. (2013) for phrase-based machine translation: (1) SEEN: incorrect translation for a source word that has never been seen in the training corpus; (2) SENSE: incorrect translation for a previously seen source word, whose correct translation (sense) has never been seen in the training corpus; (3) SCORE: a score error is made when the source word and its correct translation are both observed in training data, but the incorrect translation is scored higher than the correct alternative ; and (4) SEARCH: an error caused by pruning in beam search[11].

We extend this taxonomy to neural machine translation. As the unit of S^4 analysis is word alignment between a source word and a reference target word, we first run fast-align (Dyer et al., 2013) to get the source-target word alignments. After this, we follow the algorithm shown in Appendix C to give a summary of S^4 errors on the model's translation of test set.

Figure 5 shows the word translation results for the test set of German-English TED. Most of the errors are SCORE errors, while SEEN and SENSE errors are relatively rare. Curriculum learning significantly improves the adapted NMT systems at the word level — with 4096k Paracrawl data selected by CDS, curriculum continued training model can translate 554 more words correctly than the standard continued training model. This improvement mainly happens in SCORE errors: 1.75% of SCORE errors are corrected. SEEN and SENSE errors are also reduced by 0.02% and 0.026%, respectively. But overall, CL does not help much on SEEN errors.

4.4 Characteristics of Selected Data

We characterize the sentences chosen by different data selection methods, to understand their effect on adaptation as observed in Section 3.3.

Selected Sentences Overlap For each domain in German-English, we compute the overlap between the top n ML and CDS Paracrawl sentences. The overlap is as low as 3.69% for the top 64k sentences in the TED domain, and 8.43% for the patent domain. Even in the top 4096k sentences, there are still 46.25% and 65.40% different ones in TED and patent domain respectively. See Table 2 for examples of selected sentences.

Average Sentence Length The ML score prefers longer sentences and is more correlated with sentence length (See Figure 6) — the curve TED_ML is near linear, which might be a side-effect of sentence-length normalization. CDS produces sentences that better match the average sentence length in the in-domain corpus, which was also observed in Santamaría and Axelrod (2017).

Out-of-Vocabulary Words We count out-of-vocabulary (OOV) tokens in in-domain corpus based on the vocabulary of selected unlabeled-domain data (Figure 7). The CDS subsets cover in-domain vocabulary better than ML subsets as expected, since CDS is based on vocabulary coverage.

Unigram Distribution Distance How do unigram relative frequencies compare in the in-domain and selected Paracrawl data?

---

[10] When converged, IN_CT does not outperform CL_ML.

[11] We will only focus on the first three error categories in this paper for the purpose of model comparison.

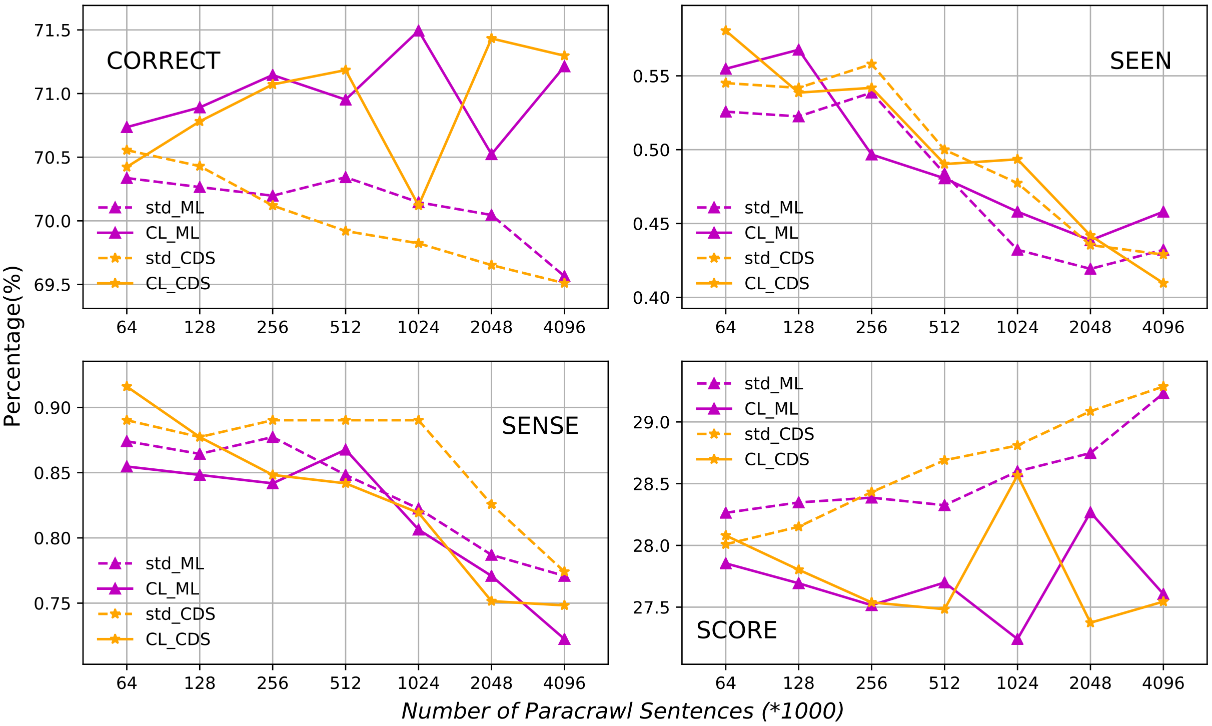


Figure 5: S^4 error analysis on German-English TED.

TED_ML	It changes the way we think; it changes the way we walk in the world; it changes our responses; it changes our attitudes towards our current situations; it changes the way we dress; it changes the way we do things; it changes the way we interact with people.
TED_CDS	But, on the other hand, this signifies that the right of self-determination, as a part of the proletarian peace program, possesses not a "Utopian" but a revolutionary character.
patent_ML	The sites x, y and z can accommodate a large variety of cations with x=na+, k+, ca2+, vacancy; y=mg2+, fe2+, al3+, fe3+, li+, mn2+ and z=al3+, mg2+ , fe3+, v3+, cr3+; while the t site is predominantly occupied by si4+.
patent_CDS	To select alternative viewing methods, such as for 3d-tv.

Table 2: The top ranked sentences selected from German-English Paracrawl corpus.

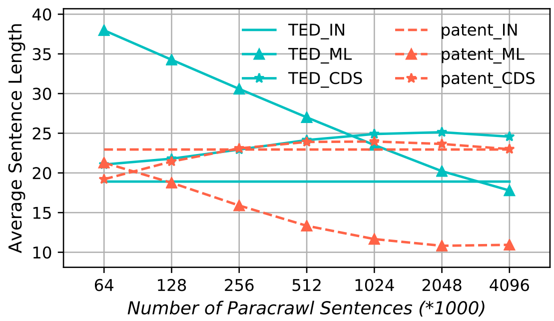


Figure 6: Average sentence length for increasing size of Paracrawl data. This is calculated on the source side of German-English pairs. TED_IN stands for TED corpus. TED_ML and TED_CDS represent the Paracrawl samples selected by ML and CDS methods.

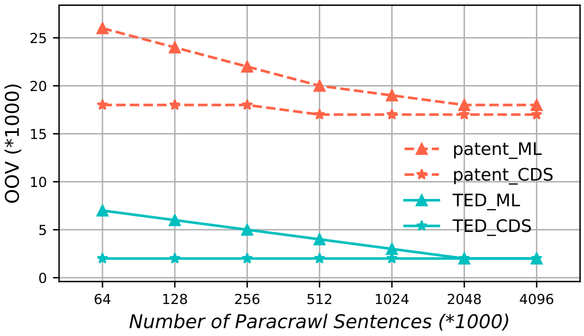


Figure 7: Number of OOV words in the source side of German-English target domain corpora.

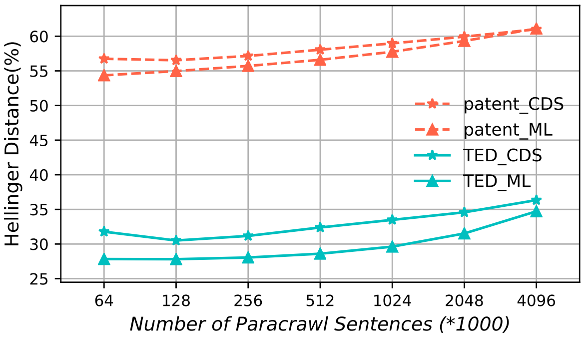


Figure 8: Hellinger distance for source side unigram distributions of German-English corpora between in-domain data and ML/CDS selected data.

We measure the difference of unigram distributions from two corpora by *Hellinger distance*, which is defined as Equation 2 when the probability distribution is discrete, where P and Q are the unigram distributions for the source side of in-domain and Paracrawl. V is the vocabulary size.[12]

$$H_{HD}(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{V}(\sqrt{p_i} - \sqrt{q_i})^2}. \quad (2)$$

From Figure 8, we can see ML can better match the in-domain vocabulary distribution than CDS.

With respect to the OOV rate and unigram distribution, patent is more distant from the Paracrawl data than TED is. Figure 2 suggests that CDS dominates ML for distant domains such as Patent, while ML can do slightly better than CDS for domains that are not that distant such as TED.

5 Related Work

Curriculum learning has shown its potential to improve sample efficiency for neural models (Graves et al., 2017; Weinshall and Cohen, 2018) by guiding the order of presented samples, usually from easier-to-learn samples to difficult samples. Although there is no single criterion to measure difficulty for general neural machine translation tasks (Kocmi and Bojar, 2017; Wang et al., 2018; Zhang et al., 2018; Kumar et al., 2019; Platanios et al., 2019), for the domain adaptation scenario, we measure difficulty based on the distance from in-domain data. Compared to previous work, our application of curriculum learning mainly focuses on improvements on translation quality without consideration of convergence speed.

Chu and Wang (2018) surveyed recent domain adaptation methods for NMT. In their taxonomy, our workflow in Figure 1 can be considered a hybrid that uses both data-centric and model-centric techniques due to the use of additional unlabeled-domain data, with a modified training procedure based for continued training.

For data-centric domain adaptation methods, our curriculum learning approach has connections to instance weighting. In our work, the presentation of certain examples at specific training phases is equivalent to up-weighting those examples and down-weight the others at that time. Weights of similar samples and less similar ones are adjusted dynamically during the training of NMT models based on the curriculum training strategy. In NMT, instance weighting is usually implemented by modifying the objective function (Chen and Huang, 2016; Wang et al., 2017; Chen et al., 2017). In statistical machine translation, Matsoukas et al. (2009) extract features from sentences to capture their domains and then use a classifier to map features to sentence weights. Foster et al. extend this method by weighting at the level of phrase pairs. Shah et al. (2010) use resampling to weight corpora and alignments. Mansour and Ney (2012) focus on sentence-level weighting for phrase extraction. Zhou et al. (2015) weight examples based on their word distributions.

For model-centric domain adaptation methods, our work is related to van der Wees et al. (2017). They adopt gradual fine-tuning, which does the opposite of our method: training starts from the whole dataset, and the training set gradually decreases by removing less similar sentences. Wang et al. (2018) use a similar approach, where the NMT model is trained on progressively noise-reduced data batches. However, such schedules have the risk of wasting computation on non-relevant data, especially when most of the Paracrawl data is not similar to the target domain.

6 Conclusion

We introduced a curriculum learning approach to adapt neural machine translation models to new domains. Our approach first ranks unlabeled-domain training samples based on their similarity to in-domain data, and then adopts a probabilistic curriculum learning strategy so that more similar samples are used earlier and more frequently during training.

---

[12] In Figure 8, for the purpose of fair comparison, each distribution is defined on the same vocabulary, consisting of the source side vocabulary of TED, patent and Paracrawl data.

We show the effectiveness of our method on four tasks. Results show that curriculum learning models can improve over the standard continued training model by up to 3.22 BLEU points and can take better advantage of distant and noisy data. According to our S^4 analysis of lexical choice errors, this improvement is mainly due to better scoring of words that acquire a new SENSE or have a different SCORE distribution in the new domain. Our extensive empirical analysis suggests that this approach is effective for several reasons: (1) It provides a robust way to augment the training data with samples that have different levels of similarity to the in-domain data. Unlabeled-domain data such as webcrawls naturally have a diverse set of sentences, and the probabilistic curriculum allows us to exploit as much diversity as possible. (2) It implements the intuition that samples more similar to in-domain data are seen earlier and more frequently; when adding a new shard into the training set, the previously visited shards are still used, so the model will not forget what it just learned. (3) It builds on a strong continued training baseline, which continues on in-domain data. (4) The method implements best practices that have shown to be helpful in NMT, e.g. bucketing, mini-batching, and data shuffling.

For future work, it would be interesting to measure how curriculum learning models perform on the general domain test set (rather than the in-domain test set we focus on in this work); do they suffer more or less from catastrophic forgetting (Goodfellow et al., 2014; Kirkpatrick et al., 2017; Khayrallah et al., 2018; Thompson et al., 2019)?

Acknowledgments

This work is supported in part by a AWS Machine Learning Research Award and a grant from the Office of the Director of National Intelligence, Intelligence Advanced Research Projects Activity (IARPA), via contract FA8650-17-C-9115. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

We thank the organizers and participants of the 2018 Machine Translation Marathon for providing a productive environment to start this project. We also thank Amittai Axelrod, Hongyuan Mei and all the team members of the JHU SCALE 2018 workshop for helpful discussions.

References

Amittai Axelrod. 2017. Cynical selection of language model training data. *arXiv preprint arXiv: 1709.02279*.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Boxing Chen and Fei Huang. 2016. Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*.

Kevin Duh. 2018. The multitarget ted talks task. `http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/`.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

George Foster, Cyril Goutte, and Roland Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 conference on empirical methods in natural language processing*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Ann Irvine, John Morgan, Marine Carpuat, Hal Daumé III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*.

Marcin. Junczys-Dowmunt, Bruno. Pouliquen, and Christophe. Mazenc. 2016. Coppa v 2 . 0 : Corpus of parallel patent applications building large parallel corpora with gnu make.

Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*.

Philipp Koehn, Kevin Duh, and Brian Thompson. 2018. The jhu machine translation systems for wmt 2018. In *Proceedings of the Third Conference on Machine Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. 2019. Reinforcement learning based curriculum optimization for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Saab Mansour and Hermann Ney. 2012. A simple and effective weighted phrase extraction for machine translation adaptation. In *International Workshop on Spoken Language Translation*.

Spyros Matsoukas, Antti-Veikko I Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M. Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lucía Santamaría and Amittai Axelrod. 2017. Data selection with cluster-based language difference models and cynical selection. In *International Workshop on Spoken Language Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Kashif Shah, Loïc Barrault, and Holger Schwenk. 2010. Translation model adaptation by resampling. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Daphna Weinshall and Gad Cohen. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *Proceedings of the 35th International Conference on Machine Learning*.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. 2018. An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint arXiv:1811.00739*.

Xinpeng Zhou, Hailong Cao, and Tiejun Zhao. 2015. Domain adaptation for smt using sentence weight. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*.

A In-domain Data Details

Dataset	TED(de)	TED(ru)	patent(de)	patent(ru)
#samples	151,627	180,316	821,267	28,536

Table 3: In-domain data statistics.

The total amount of the in-domain data in each domain is summarized in Table 3. In this paper, we uniformly sample 15k in-domain data from each dataset. We choose the amount of 15k, which makes up a relatively small percentage of the original corpora, in order to evaluate the extreme case of low-resource domain adaptation settings. Under this setting, the positive effect of adding more selected unlabeled-domain data into training corpus is more obvious in terms of the performance improvement of NMT models. Our pilot experiments show that curriculum learning can scale with more in-domain data—it consistently outperforms the standard training policy, but with less improvement. This is not surprising, as when there is enough in-domain data, continued training on only the in-domain data can already achieve a pretty good performance, and we do not need to use extra unlabeled-domain data to augment it any more, neither does curriculum learning.

B Data Sharding

We experimented with different number of shards for curriculum learning models as shown in Figure 9. Overall, the performance shows the tendency to first improve and then degrade as the number of shards increases. Consider the extreme case, where the data are all put into one shard, or there are as many shards as samples, then it will actually be the same as the standard continued training.

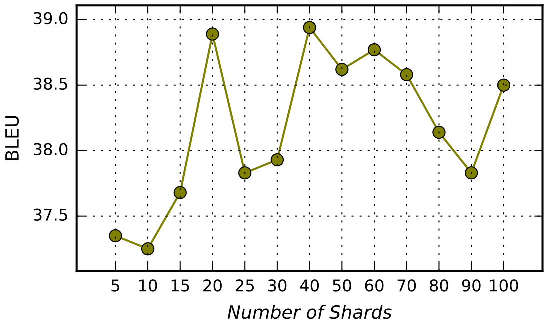


Figure 9: Tuning number of shards on a curriculum learning model trained with German-English corpus augmented by 1024k Paracrawl data.

C S^4 Error Analysis Algorithm

The algorithm for getting S^4 word translation error statistics is shown in 1.

Algorithm 1 S^4 Error Analysis

```
▷$S$: The source-side sentences in test set
▷$f_i$: The $i$th unique word in a sentence
▷$E^r(f_i)$: Words aligned to $f_i$ in the reference translation for test set
▷$E^h(f_i)$: Words aligned to $f_i$ in the output translation for test set
▷$E^t(f_i)$: Words aligned to $f_i$ in the reference translation for training set
procedure S4ErrorCounter($S, E^r, E^h, E^t$)
    correct←0; seen←0; sense←0; score←0
    for $s \in S$ do
        for $f_i \in s$ do
            for $e_j \in E^r(f_i)$ do
                if $e_j \in E^h(f_i)$ then
                    correct ← correct+1
                else
                    if $f_i \notin E^t$ then
                        seen ← seen+1
                    else
                        if $e_j \notin E^t(f_i)$ then
                            sense ← sense+1
                        else
                            score ← score+1
                        end if
                    end if
                end if
            end for
        end for
    end for
    return correct, seen, sense, score
end procedure
```

D Bilingual Criterion

In previous experiments, we only considered the data selection scores obtained from the source side of the corpora. It is very likely that curriculum learning would benefit from also taking into account the features of target side. For Moore-Lewis score, following Axelrod et al. (2011), we sum the scores over each side of the corpus:

$$[H_{I-src}(s)-H_{N-src}(s)]+[H_{I-tr}(s)-H_{N-trg}(s)]. \tag{3}$$

In addition, we also conduct comparison experiments using scores obtained from only the target side for both of the two data selection methods. Results are shown in Figure 10.

For Moore-Lewis method, in terms of the standard models, scores collected from target side can lead to better translation than source side scores, and bilingual criteria is somewhere in between, for all the sizes of Paracrawl data we experimented with. But this does not map to the curriculum learning models perfectly. Although CL_en achieves several impressive BLEU scores (39.35 BLEU at 512k, 39.26 BLEU at 2048k), CL_de can sometimes outperform it. And the performance of their bilingual counterparts are unpredictable: it can be either worse or better than both of them. At 4096k ML selected sentences, CL_bi improves the BLEU score to 39.37 BLEU, which is the best test score among all the results we have for German-English TED. For cynical data selection, it is obvious that curriculum learning models prefer the scores obtained from the source side of the corpus.

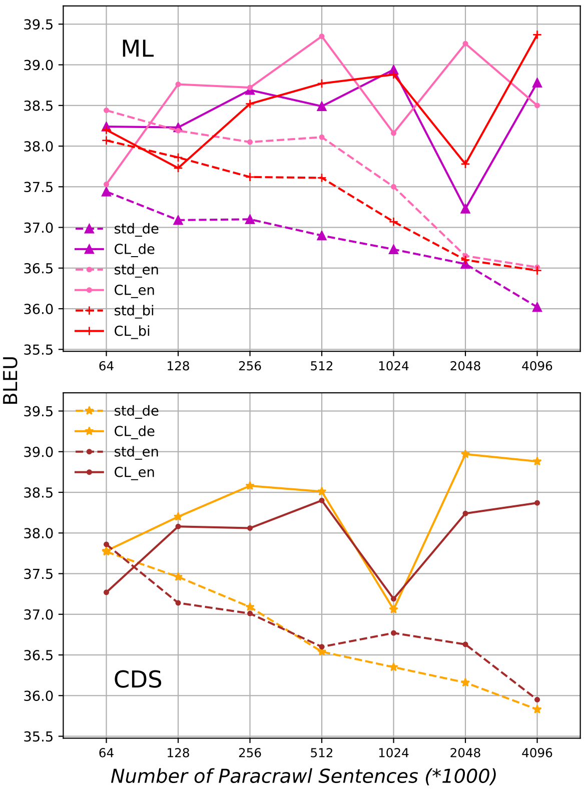


Figure 10: Performance of models continued trained with TED data and Paracrawl data, ranked by their similarity scores collected from the source side (de) or the target side (en) or both sides (bi) of the sentence pairs.

E Perplexity Selection

In Section 3.3, we train models with Paracrawl data of different sizes, only after the training is finished and we get the decoding results of those NMT systems, as shown in Figure 2, can we know which size should be the best choice. Moore and Lewis (2010) proposed a method that can determine the count cut-offs of the selected data beforehand, so that a lot of time and computation will be saved. In their work, the optimal selection threshold is determined by the perplexity of in-domain set evaluated on the language models trained on the different-size selected subsets. We name this method as perplexity selection and we are curious whether it is effective in the NMT settings. Unfortunately, the best thresholds elected by this method (Figure 11) are inconsistent with the cutoffs that achieve high BLEU scores in Figure 2. We can then conclude that perplexity selection may not be an appropriate way to determine the optimal amount of unlabeled-domain data to use for NMT models.

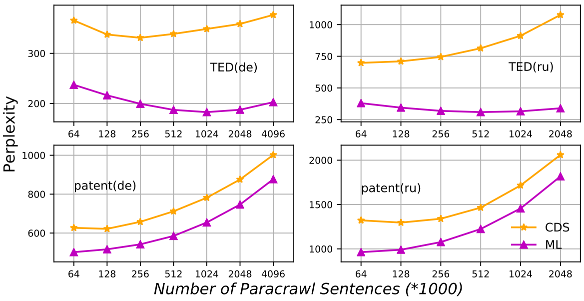


Figure 11: Perplexity of selected data evaluated on the language model learned from in-domain corpus.

However, if computational resources are limited, according to the experiment results (Figure 2) in our work, we recommend 1024k as the first choice for cutoffs on ranked unlabeled-domain data, for NMT domain adaptation models trained with curriculum learning strategy.

Improving Robustness of Machine Translation with Synthetic Noise

Vaibhav*, **Sumeet Singh***, **Craig Stewart***, **Graham Neubig**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{vvaibhav,sumeets,cas1, gneubig}@cs.cmu.edu

Abstract

Modern Machine Translation (MT) systems perform remarkably well on clean, in-domain text. However most human generated text, particularly in the realm of social media, is full of typos, slang, dialect, idiolect and other noise which can have a disastrous impact on the accuracy of MT. In this paper we propose methods to enhance the robustness of MT systems by emulating naturally occurring noise in otherwise clean data. Synthesizing noise in this manner we are ultimately able to make a vanilla MT system more resilient to naturally occurring noise, partially mitigating loss in accuracy resulting therefrom [1].

1 Introduction

Machine Translation (MT) systems have been shown to exhibit severely degraded performance when required to translate of out-of-domain or noisy data (Luong and Manning, 2015; Sakaguchi et al., 2016; Belinkov and Bisk, 2017). This is particularly pronounced when systems trained on clean, formalized parallel data such as Europarl (Koehn, 2005), are tasked with translation of unedited, human generated text such as is common in domains such as social media, where accurate translation is becoming of widespread relevance (Michel and Neubig, 2018).

Improving the robustness of MT systems to naturally occurring noise presents an important and interesting task. Recent work on MT robustness (Belinkov and Bisk, 2017) has demonstrated the need to build or adapt systems that are resilient to such noise. We approach the problem of adapting to noisy data aiming to answer two primary research questions:

1. Can we artificially synthesize the types of noise common to social media text in otherwise clean data?
2. Are we able to improve the performance of vanilla MT systems on noisy data by leveraging artificially generated noise?

In this work we present two primary methods of synthesizing natural noise, in accordance with the types of noise identified in prior work as naturally occurring in internet and social media based text (Eisenstein, 2013; Michel and Neubig, 2018). Specifically, we introduce a **synthetic noise induction** model which heuristically introduces types of noise unique to social media text and **labeled back translation** (Sennrich et al., 2015a), a data-driven method to emulate target noise.

We present a series of experiments based on the Machine Translation of Noisy Text (MTNT) data set (Michel and Neubig, 2018) through which we demonstrate improved resilience of a vanilla MT system by adaptation using artificially noised data.

2 Related Work

Szegedy et al. (2013) demonstrate the fragility of neural networks to noisy input. This fragility has been shown to extend to MT systems (Belinkov and Bisk, 2017; Khayrallah and Koehn, 2018) where both artificial and natural noise are shown to negatively affect performance.

Human generated text on the internet and social media are a particularly rich source of natural noise (Eisenstein, 2013; Baldwin et al., 2015) which causes pronounced problems for MT (Michel and Neubig, 2018).

Robustness to noise in MT can be treated as a domain adaptation problem (Koehn and Knowles, 2017) and several attempts have been made to

---

* These authors contributed equally

[1] Code available at https://github.com/MysteryVaibhav/robust_mtnt

Proceedings of NAACL-HLT 2019, pages 1916–1920
Minneapolis, Minnesota, June 2 - June 7, 2019.

handle noise from this perspective. Notable approaches (Li et al., 2010; Axelrod et al., 2011) include training on varying amounts of data from the target domain. Luong and Manning (2015) suggest the use of fine-tuning on varying amounts of target domain data, and Barone et al. (2017) note a logarithmic relationship between the amount of data used in fine-tuning and the relative success of MT models.

Other approaches to domain adaptation include weighting of domains in the system objective function (Wang et al., 2017) and specifically curated datasets for adaptation (Blodgett et al., 2017). Kobus et al. (2016) introduce a method of domain tagging to assist neural models in differentiating domains. Whilst the above approaches have shown success in specifically adapting across domains, we contend that adaptation to noise is a nuanced task and treating the problem as a simple domain adaptation task may fail to fully account for the varied types of noise that can occur in internet and social media text.

Experiments that specifically handle noise include text normalization approaches (Baldwin et al., 2015) and (most relevant to our work) the artificial induction of noise in otherwise clean data (Sperber et al., 2017; Belinkov and Bisk, 2017).

3 Data

To date, work in the adaptation of MT to natural noise has been restricted by a lack of available parallel data. Michel and Neubig (2018) recently introduced a new data set of noisy social media content and demonstrate the success of fine-tuning which we leverage in the current work. The dataset consists of naturally noisy data from social media sources in both English-French and English-Japanese pairs.

In our experimentation we utilize the subset of the data for English to French which contains data scraped from Reddit[2]. The data set contains training, validation and test data. The training data is used in fine-tuning of our model as outlined below. All results are reported on the MTNT test set for French-English. We additionally use other datasets including Europarl (EP) (Koehn, 2005) and TED talks (TED) (Ye et al., 2018) for training our models as described in §5.

Training Data	# Sentences	Pruned Size
Europarl (EP)	2,007,723	1,859,898
Ted talk (TED)	192,304	181,582
Noisy Text (MTNT)	19,161	18,112

Table 1: Statistics about different datasets used in our experiments. We prune each dataset to retain sentences with length ≤ 50.

4 Baseline Model

Our baseline MT model architecture consists of a bidirectional Long Short-Term Memory (LSTM) network encoder-decoder model with two layers. The hidden and embedding sizes are set to 256 and 512, respectively. We also employ weight-tying (Press and Wolf, 2016) between the embedding layer and projection layer of the decoder.

For expediency and convenience of experimentation we have chosen to deploy a smaller, faster variant of the model used in Michel and Neubig (2018), which allows us to provide comparative results across a variety of settings. Other model parameters reflect the implementation outlined in Michel and Neubig (2018).

In all experimental settings we employ Byte-Pair Encoding (BPE) (Sennrich et al., 2015b) using SentencePiece[3].

5 Experimental Approaches

We propose two primary approaches to increasing the resilience of our baseline model to the MTNT data, outlined as follows:

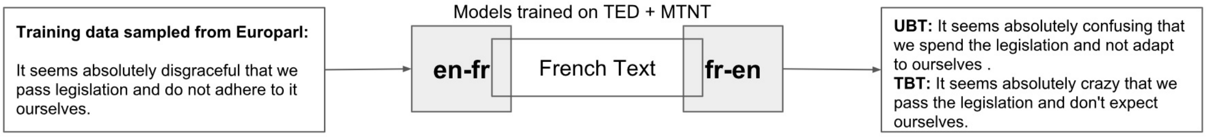
5.1 Synthetic Noise Induction (SNI)

For this method, we inject artificial noise in the clean data according to the distribution of types of noise in MTNT specified in Michel and Neubig (2018). For every token we choose to introduce the different types of noise with some probability on both French and English sides in 100k sentences of EP. Specifically, we fix the probabilities of error types as follows: spelling (0.04), profanity (0.007), grammar (0.015) and emoticons (0.002). To simulate spelling errors, we randomly add or drop a character in a given word. For grammar error and profanity, we randomly select and insert a stop word or an expletive and its translation on either side. Similarly for emoticons, we randomly

---

[2] `www.reddit.com`

[3] `https://github.com/google/sentencepiece`



Training data sampled from Europarl:

It seems absolutely disgraceful that we pass legislation and do not adhere to it ourselves.

UBT: It seems absolutely confusing that we spend the legislation and not adapt to ourselves .

TBT: It seems absolutely crazy that we pass the legislation and don't expect ourselves.

Figure 1: Pipeline for injecting noise through back translation. For demostration purposes we show the process in an English sentence but in experiments, we use French sentences as input.

select an emoticon and insert it on both sides. Algorithm 1 elaborates on this procedure.

Algorithm 1 Synthetic Noise Induction

```
Inputs:$[(p_1, \eta_1), (p_2, \eta_2) \cdots (p_k, \eta_k)]$   ▷ pairs of noise probabilities and noise functions
procedure ADD_NOISE($fr, en$)
    $o = 1 - \sum_i p_i$   ▷ probability of keeping original
    $D = [o, p_1, p_2, \cdots, p_k]$   ▷ Discrete densities
    $j \leftarrow$ SELECT_INDEX(DRAW($D$))   ▷ noise type
    if $j \neq 0$ then   ▷ not original
        $(fr, en) \leftarrow \eta_j(fr, en)$   ▷ add noise to words
    return $fr, en$
```

5.2 Noise Generation Through Back-Translation

We further propose two experimental methods to inject noise into clean data using the back-translation technique (Sennrich et al., 2015a).

5.2.1 Un-tagged Back-Translation (UBT)

We first train both our baseline model for fr-en and an en-fr model using TED and MTNT. We subsequently take 100k French sentences from EP and generate a noisy version thereof by passing them sequentially through the trained models as shown in Figure 1. The resulting translation will be inherently noisy as a result of imperfect translation of the intervening MT system.

5.2.2 Tagged Back-Translation (TBT)

The intuition behind this method is to generate noise in clean data whilst leveraging the particular style of the intermediate corpus. Both models are trained using TED and MTNT as in the preceding setting, save that we additionally append a tag in front on every sentence while training to indicate the origin data set of each sentence (Kobus et al., 2016). For generating the noisy version of 100k French sentences from EP, we append MTNT tag in front of the sentences before passing them through the pipeline shown in Figure 1.

	Training data	BLEU
	Baselines	
Baseline	Europarl (EP)	14.42
+ FT w/	MTNT-train-10k	22.49
+ FT w/	MTNT-train-20k	23.74
Baseline FT w/	TED-100k	10.92
+ FT w/	MTNT-train-20k	24.10
	Synthetic Noise Induction	
Baseline FT w/	EP-100k-SNI	13.53
+ FT w/	MTNT-train-10k	22.67
+ FT w/	MTNT-train-20k	25.05
	Un-tagged Back Translation	
Baseline FT w/	EP-100k-UBT	18.71
+ FT w/	MTNT-train-10k	22.75
+ FT w/	MTNT-train-20k	24.84
	Tagged Back Translation	
Baseline FT w/	EP-100k-TBT	20.49
+ FT w/	MTNT-train-10k	**23.89**
+ FT w/	MTNT-train-20k	**25.75**

Table 2: BLEU scores are reported on MTNT test set. MTNT valid set is used for fine-tuning in all the experiments. + FT denotes fine-tuning of the Baseline model of that particular sub-table, being continued training for 30 epochs or until convergence.

6 Results

We present quantitative results of our experiments in Table 2. Of specific note is the apparent correlation between the amount of in-domain training data and the resulting BLEU score. The tagged back-translation technique produces the most pronounced increase in BLEU score of +6.07 points ($14.42 \longrightarrow 20.49$). This represents a particularly significant result given that we do not fine-tune the baseline model on in-domain data, attributing this gain to the quality of the noise generated.

The results for all our proposed experimental methods further imply that out-of-domain clean data can be leveraged to make the existing MT models robust on a noisy dataset. However, sim-

Systems	Output
REFERENCE	> And yes, I am an idiot with a telephone in usb-c... F*** that's annoying, I had to invest in new cables when I changed phones.
Baseline (trained on EP)	And yes, I am an eelot with a phone in the factory ... P***** to do so, I have invested in new words when I have changed telephone.
FT w/ MTNT-train-20k	> And yes, I am an idiot with a phone in Ub-c. Sh**, it's annoying that, I have to invest in new cable when I changed a phone.
FT w/ EP-100k-TBT	- And yes, I'm an idiot with a phone in the factory... Puard is annoying that, I have to invest in new cables when I changed phone.
FT w/ EP-100k-TBT + MTNT-train-20k	> And yes, I am an idiot with a phone in USb-c... Sh** is annoying that, I have to invest in new cables when I changed a phone.

Table 3: Output comparison of decoded sentences across different models. Profane words are censored.

Figure 2: The impact of varying the amount of Synthetic Noise Induction on BLEU.

ply using clean data is not that beneficial as can be seen from the experiment involving *FT Baseline w/ TED-100k*.

We further present analysis of both methods introduced above. Figure 2 illustrates the relative effect of varying the level of SNI on the BLEU score as evaluated on the newsdiscuss2015[4] dev set, which is a clean dataset. From this we note that the relationship between the amount of noise and the effect on BLEU score appears to be linear. We also note that the most negative effect is obtained by including profanity. Our current approach involves inserting expletives, spelling and grammatical errors at random positions in a given sentence. However we note that our approach might under-represent the nuanced linguistic usage of expletives in natural text, which may result in its above-mentioned effect on accuracy.

Table 3 shows the decoded output produced by different models. We find that the output produced by our best model is reasonably successful at imitating the language and style of the reference. The output of *Baseline + FT w/ EP-100k-TBT* is far superior than that of *Baseline*, which highlights the quality of obtained back translated noisy EP through our tagging method.

Systems	Output
REFERENCE	Voluntary or not because politicians are *very* friendly with large businesses.
FT w/ EP-100k-TBT	Whether it's voluntarily, or invoiseally because the fonts are *èsn* friends with the big companies.
FT w/ EP-100k-TBT + MTNT-train-10k	Whether it's voluntarily, or invokes because the politics are *rès* friends with big companies.
FT w/ EP-100k-TBT + MTNT-train-20k	Whether it's voluntarily, or invisible because the politics are *very* friends with big companies.

Table 4: Output comparison of decoded sentences for different amounts of supervision.

We also consider the effect of varying the amount of supervision which is added for fine-tuning the model. From Table 4 we note that the *Baseline + FT w/ EP-100k-TBT* model already produces a reasonable translation for the input sentence. However, if we further fine-tune the model using only 10k MTNT data, we note that the model still struggles with generation of *very*. This error dissipates if we use 20k MTNT data for fine-tuning. These represent small nuances which the model learns to capture with increasing supervision.

To better understand the performance difference between UBT and TBT, we evaluate the noised EP data. Figure 1 shows an example where we can clearly see that the style of translation obtained from TBT is very informal as opposed to the output generated by UBT. Both the outputs are noisy and different from the input but since the TBT method enforces the style of MTNT, the resulting output is perceptibly closer in style to the MTNT equivalent. This difference results in a gain of 0.9 BLEU of TBT over UBT.

7 Conclusion

This paper introduced two methods of improving the resilience of vanilla MT systems to noise occurring in internet and social media text: a method of emulating specific types of noise and the use of back-translation to create artificial noise. Both of these methods are shown to increase system accuracy when used in fine-tuning without the need for the training of a new system and for large amounts of naturally noisy parallel data.

---

[4] `http://www.statmt.org/wmt15/test.tgz`

Acknowledgements

The authors would like to thank the AWS Educate program for donating computational GPU resources used in this work.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362. Association for Computational Linguistics.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135. Association for Computational Linguistics.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. *CoRR*, abs/1707.09920.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *CoRR*, abs/1711.02173.

Su Lin Blodgett, Johnny Wei, and Brendan O'Connor. 2017. A dataset and classifier for recognizing social media english. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 56–61. Association for Computational Linguistics.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *HLT-NAACL*.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. *arXiv preprint arXiv:1805.12282*.

Catherine Kobus, Josep Maria Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *CoRR*, abs/1612.06140.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

Mu Li, Yinggong Zhao, Dongdong Zhang, and Ming Zhou. 2010. Adaptive development data selection for log-linear model in statistical machine translation. pages 662–670.

Minh-Thang Luong and Christopher D. Manning. 2015. Neural machine translation systems for spoken language domains.

Paul Michel and Graham Neubig. 2018. Mtnt: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *CoRR*, abs/1608.05859.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2016. Robsut wrod reocginiton via semi-character recurrent neural network. *CoRR*, abs/1608.02214.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Matthias Sperber, Jan Niehues, and A. Waibel. 2017. Toward robust neural machine translation for noisy input sequences.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488. Association for Computational Linguistics.

Qi Ye, Sachan Devendra, Felix Matthieu, Padmanabhan Sarguna, and Neubig Graham. 2018. When and why are pre-trained word embeddings useful for neural machine translation. In *HLT-NAACL*.

Non-Parametric Adaptation for Neural Machine Translation

Ankur Bapna
Google AI
ankurbpn@google.com

Orhan Firat
Google AI
orhanf@google.com

Abstract

Neural Networks trained with gradient descent are known to be susceptible to catastrophic forgetting caused by parameter shift during the training process. In the context of Neural Machine Translation (NMT) this results in poor performance on heterogeneous datasets and on sub-tasks like rare phrase translation. On the other hand, non-parametric approaches are immune to forgetting, perfectly complementing the generalization ability of NMT. However, attempts to combine non-parametric or retrieval based approaches with NMT have only been successful on narrow domains, possibly due to over-reliance on sentence level retrieval. We propose a novel n-gram level retrieval approach that relies on local phrase level similarities, allowing us to retrieve neighbors that are useful for translation even when overall sentence similarity is low. We complement this with an expressive neural network, allowing our model to extract information from the noisy retrieved context. We evaluate our semi-parametric NMT approach on a heterogeneous dataset composed of WMT, IWSLT, JRC-Acquis and OpenSubtitles, and demonstrate gains on all 4 evaluation sets. The semi-parametric nature of our approach opens the door for non-parametric domain adaptation, demonstrating strong inference-time adaptation performance on new domains without the need for any parameter updates.

1 Introduction

Over the last few years, neural sequence to sequence models (Sutskever et al., 2014; Bahdanau et al., 2015; Cho et al., 2014) have revolutionized the field of machine translation by significantly improving translation quality over their phrase based counterparts (Sennrich et al., 2015; Wu et al., 2016; Zhou et al., 2016). With more gains arising from continued research on new neural network architectures and accompanying training techniques (Vaswani et al., 2017; Gehring et al., 2017; Chen et al., 2018), NMT researchers, both in industry and academia, have doubled down on their ability to train high capacity models on large corpora with gradient based optimization.

However, despite huge improvements in overall translation quality NMT has shown some glaring weaknesses, including idiom processing, and rare word or phrase translation (Koehn and Knowles, 2017; Isabelle et al., 2017; Lee et al., 2018) - tasks that should be easy if the model could retain learned information from individual training examples. NMT has also been shown to perform poorly when dealing with multi-domain data (Farajian et al., 2017a). This 'catastrophic forgetting' problem has been well-studied in traditional neural network literature, caused by parameter shift during the training process (McCloskey and Cohen, 1989; Santoro et al., 2016). Non-parametric methods, on the other hand, are resistant to forgetting but are prone to over-fitting due to their reliance on individual training examples. We focus on a non-parametric extension to NMT, hoping to combine the generalization ability of neural networks with the eidetic memory of non-parametric methods. Given a translation query, we rely on an external retrieval mechanism to find similar source-target instances in the training corpus, which are then utilized by the model.

There has been some work on semi-parametric NMT (Gu et al., 2017; Zhang et al., 2018b; Cao and Xiong, 2018), but its effectiveness has been confined to narrow domain datasets. Existing approaches have relied on sentence level similarity metrics for retrieval, which works well for domains with high train-test overlap, but fails to retrieve useful candidates for broad domains. Even if we could find training instances with overlapping phrases it's likely that the information in most retrieved source-target pairs is noise for the pur-

Proceedings of NAACL-HLT 2019, pages 1921–1931
Minneapolis, Minnesota, June 2 - June 7, 2019.

pose of translating the current query.

To retrieve useful candidates when sentence similarity is low, we use n-gram retrieval instead of sentence retrieval. This results in neighbors which have high local overlap with the source sentence, even if they are significantly different in terms of overall sentence similarity. This is intuitively similar to utilizing information from a phrase table (Koehn et al., 2003) within NMT (Dahlmann et al., 2017), without losing the global context lost when constructing the phrase table. We also propose another simple extension using dense vectors for n-gram retrieval which allows us to exploit similarities beyond lexical overlap.

To effectively extract the signal from the noisy retrieved neighbors, we develop an extension of the approach proposed in (Cao and Xiong, 2018). While (Cao and Xiong, 2018) encode the retrieved targets without any context, we incorporate information from the current and retrieved sources while encoding the retrieved target, in order to distinguish useful information from noise.

We evaluate our semi-parametric NMT approach on two tasks.

- We evaluate our approach on a multi-domain English-French corpus constructed from narrow domain datasets like JRC-Acquis (Steinberger et al., 2006; Tiedemann) and OpenSubtitles (Tiedemann, 2009)[1], and the standard IWSLT and WMT bilingual corpora, as described in Sections 3 and 4. Our results, for the first time, indicate that semi-parametric NMT can be beneficial beyond narrow domain tasks, demonstrating gains of around 0.5 BLEU on WMT, and huge gains ranging from 2-10 BLEU points on IWSLT, JRC-Acquis and OpenSubtitles, when compared to a strong sequence to sequence baseline.

- The semi-parametric nature of our model enables non-parametric inference-time adaptation to new datasets, without the need for any parameter updates. When trained on WMT and evaluated on the other datasets, our model out-performs fine-tuning based adaptation (Luong and Manning, 2015) on JRC-Acquis and OpenSubtitles, and significantly improves performance over the non-adapted model on IWSLT.

2 Semi-parametric NMT

Standard approaches for Neural Machine Translation rely on seq2seq architectures (Sutskever et al., 2014; Bahdanau et al., 2015), where given a source sequence $X = \{x_1, x_2, \dots x_{Tx}\}$ and a target sequence $Y = \{y_1, y_2, \dots y_{Ty}\}$, the goal is to model the probability distribution, $p(y_t|X, y_1, \dots y_{t-1})$.

Semi-parametric NMT (Dahlmann et al., 2017; Gu et al., 2017) approaches this learning problem with a different formulation, by modeling $p(y_t|X, y_1, \dots y_{t-1}, \Phi_X)$ instead, where $\Phi_X = \{(X^1, Y^1) \dots (X^N, Y^N)\}$ is the set of sentence pairs where the source sentence is a neighbor of X, retrieved from the training corpus using some similarity metric. This relies on a two step approach - the retrieval stage finds training instances, (X^i, Y^i), similar to the source sentence X, and the translation stage generates the target sequence Y given X and Φ_X. We follow this setup, proposing improvements to both stages in order to enhance the applicability of semi-parametric NMT to more general translation tasks.

2.1 Retrieval Approaches

Existing approaches have proposed using off the shelf search engines for the retrieval stage. However, our objective differs from traditional information retrieval, since the goal of retrieval in semi-parametric NMT is to find neighbors which might improve translation performance, which might not correlate with maximizing sentence similarity.

Our baseline strategy relies on a sentence level similarity score, similar to those used for standard information retrieval tasks (Robertson, 2004). We compare this against finer-grained n-gram retrieval using the same similarity metric. We also propose a dense vector based n-gram retrieval strategy, using representations extracted from a pre-trained NMT model.

2.1.1 IDF Based Sentence Retrieval

Our baseline approach relies on a simple inverse document frequency (IDF) based similarity score. We define the IDF score of any token, t, as $f_t = \log(\frac{\|C\|}{n_t})$, where $\|C\|$ is the number of sentence pairs in training corpus and n_t is the number of sentences t occurs in. Let any two sentence pairs in the corpus be (X^i, Y^i) and (X^j, Y^j). Then we define the similarity between (X^i, Y^i) and

---

[1] http://www.opensubtitles.org/

(X^j, Y^j) by,

$$sim(X^i, X^j) = 2 \times \Sigma_{t \in (X^i \cap X^j)} f_t - \Sigma_{t \in (X^i \cup X^j)} f_t \tag{1}$$

For every sentence in the training, dev and test corpora, we find the N most similar training sentence pairs and provide them as context to NMT.

2.1.2 IDF Based N-Gram Retrieval

Motivated by phrase based SMT, we retrieve neighbors which have high local, sub-sentence level overlap with the source sentence. We adapt our approach to retrieve n-grams instead of sentences. We note that the similarity metric defined above for sentences is equally applicable for n-gram retrieval.

Let $X = (t^1, ...t^T)$ be a sentence. Then the set of all possible n-grams of X, for a given n, can be defined as $S_X^n = \{(t_i, ...t_{i+n}) \, \forall \, 1 \le i \le T\}$ (also including padding at the end). To reduce the number of n-grams used to represent every sentence, we define the reduced set of n-grams for X to be $\hat{S_X^n} = \{(t_i, ...t_{i+n}) \, \forall \, 1 \le i \le T, \, i \mod \frac{n}{2} = 1\}$.

We represent every sentence by their reduced n-gram set. For every n-gram in $\hat{S_X^n}$, we find the closest n-gram in the training set using the IDF similarity defined above. For each retrieved n-gram we find the corresponding sentence (In case an n-gram is present in multiple sentences, we choose one randomly). The set of neighbors of X is then the set of all sentences in the training corpus that contain an n-gram that maximizes the n-gram similarity with any n-gram in $\hat{S_X^n}$.

To capture phrases of different lengths we use multiple n-gram widths, n. In case a sentence has already been added to the retrieved set, we find the next most similar sentence to avoid having duplicates. The number of neighbors retrieved for each source sentence is proportional to its length.

2.1.3 Dense Vector Based N-Gram Retrieval

We also extend our n-gram retrieval strategy with dense vector based n-gram representations. The objective behind using a dense vector based approach is to incorporate information relevant to the translation task in the retrieval stage. We use a pre-trained Transformer Base (Vaswani et al., 2017) encoder trained on WMT to generate sub-word level dense representations for the sentence. The representation for each n-gram is now defined to be the mean of the representations of all its constituent sub-words. We use the $L2$ distance of n-gram representations as the retrieval criterion. Note that we use a sub-word level decomposition of sentences for dense retrieval, as compared to word-level for IDF based retrieval (i.e., n-grams are composed of sub-words instead of words).

Following the approach described for IDF based n-gram retrieval, we use multiple values of n, and remove duplicate neighbors while creating the retrieved set.

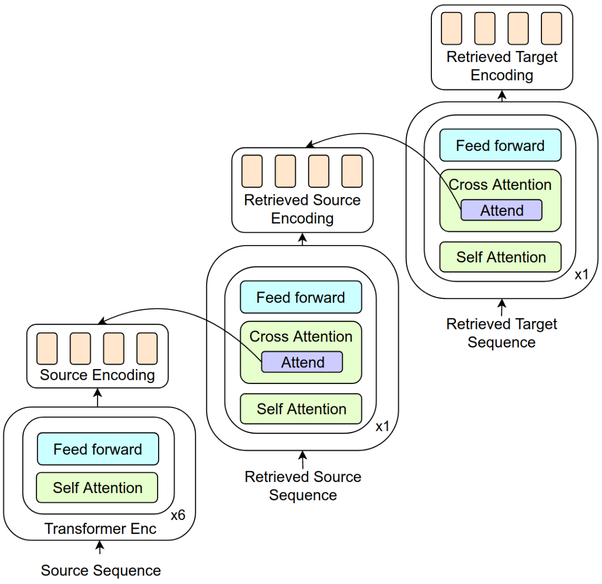


Figure 1: Architecture of the Conditional Source Target Memory. The retrieved targets, Y^i, are encoded in a transformer encoder, incorporating the attention context from the retrieved sources, X^i. In turn, the retrieved sources, X^i, are encoded while incorporating context from the current translation source, X.

2.2 NMT with Context Retrieval

To incorporate the retrieved neighbors, Φ_X, within the NMT model, we first encode them using Transformer layers, as described in subsection 2.2.1. This encoded memory is then used within the decoder via an attention mechanism, as described in subsection 2.2.2.

2.2.1 Conditional Source Target Memory

We now describe how each retrieved translation pair, (X^i, Y^i), is encoded. This architecture is illustrated in Figure 1.

- We first encode the retrieved source, X^i, in a Transformer layer. Apart from self-attention, we incorporate information from the encoder representation of the current source, X, using decoder style cross-attention.

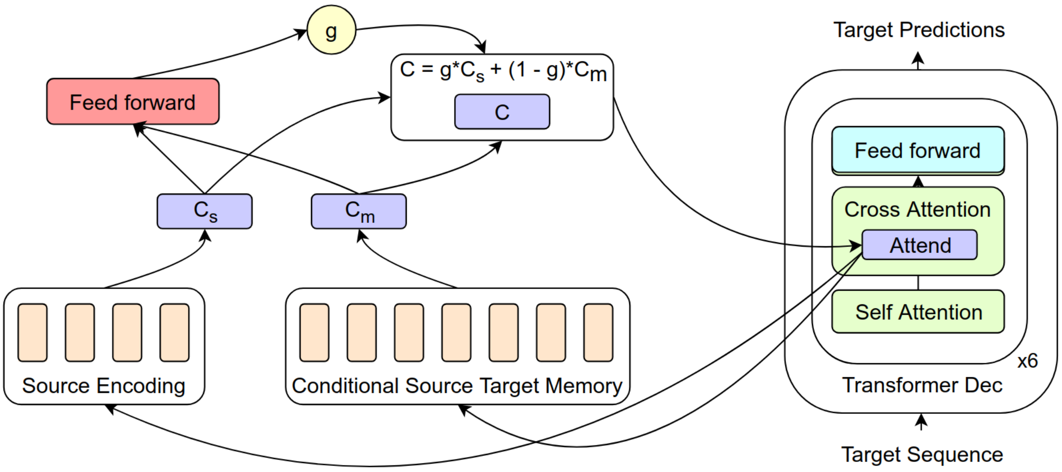


Figure 2: Architecture of the gated attention mechanism used in the multi-source transformer decoder.

- The retrieved target, Y^i, is encoded in a similar manner, attending the encoded representation of X^i generated in the previous step.

The encoded representations for all targets, $\{Y^i, 1 \leq i \leq N\}$, are then concatenated along the time axis to form the Conditional Source Target Memory (CSTM).

2.2.2 Gated Multi-Source Attention

We use gated multi-source attention to combine the context from the source encoder representations and the CSTM. This is similar to the gated attention employed by (Cao and Xiong, 2018). We use a Transformer based decoder that attends to both, the encoder outputs and the CSTM, in every cross-attention layer. The rest of the decoder architecture remains unchanged.

Let the context vectors obtained by applying multi-head attention to the source and memory, with query q_t be c_t^s and c_t^m respectively. Then the gated context vector, c^t, is given by,

$$g_t = \sigma(W_{gs}c_t^s + W_{gm}c_t^m) \tag{2}$$

$$c_t = g_t * c_t^s + (1 - g_t) * c_t^m \tag{3}$$

where g_t is the scalar gating variable at time-step t, and W_{gs} and W_{gm} are learned parameters. These steps are illustrated in Figure 2.

3 Experiments

3.1 Data and Evaluation

We compare the performance of a standard Transformer Base model and our semi-parametric NMT approach on an English-French translation task. We create a new heterogeneous dataset, constructed from a combination of the WMT training set (36M pairs), the IWSLT bilingual corpus (237k pairs), JRC-Acquis (797k pairs)[2] and OpenSubtitles (33M pairs)[3]. For WMT, we use newstest 13 for validation and newstest 14 for test. For IWSLT, we use a combination of the test corpora from 2012-14 for validation and test 2015 for eval. For OpenSubtitles and JRC-Acquis, we create our own splits for validation and test, since no benchmark split is publicly available. After deduping, the JRC-Acquis test and validation set contain 6574 and 5121 sentence pairs respectively. The OpenSubtitles test and validation sets contain 3975 and 3488 pairs. For multi-domain training, the validation set is a concatenation of the four individual validation sets.

All datasets are tokenized with the Moses tokenizer (Koehn et al., 2007) and mixed without any sampling. We use a shared vocabulary SentencePiece Model (Kudo and Richardson, 2018) for sub-word tokenization, with a vocabulary size of 32000 tokens. We train each model for 1M steps, and choose the best checkpoint from the last 5 checkpoints based on validation performance. BLEU scores are computed with tokenized true-cased output and references with *multi-bleu.perl* from Moses.

For IDF based sentence retrieval, for each sentence in the training, dev and test corpus, we use $N = 10$ neighbors per example during both, training and evaluation. For the N-Gram level retrieval strategies, we used $N = 10$ neighbors dur-

---

[2] From http://opus.nlpl.eu/JRC-Acquis.php

[3] From http://opus.nlpl.eu/OpenSubtitles.php

Model	Data	newstest 14	IWSLT 2015	OpenSub	JRC-Acquis
TransformerBase	Multi Domain (MD)	41.92	43.17	26.67	56.19
+ CSTM	MD + IDF Sentence	40.89	42.35	28.25	65.38
+ CSTM	MD + IDF N-Gram	41.92	**45.09**	28.74	66.39
+ CSTM	MD + Dense N-Gram	**42.41**	**45.02**	**29.06**	**66.92**

Table 1: Comparison of test translation quality (BLEU) with different retrieval strategies. Multi-domain is a concatenation of all 4 datasets. IDF Sentence, IDF-NGram and Dense N-Gram correspond to multi-domain datasets constructed with the different retrieval strategies.

source	‘The top copy of the passenger waybill *shall be kept on the* bus or coach throughout **the journey to which it refers .**’
neighbor source	‘The top copy of the journey form *shall be kept on the* vehicle during the whole of **the journey to which it refers .**’
baseline translation	‘La copie supérieure de la lettre de transport de voyageurs doit être conservée dans l’ autobus ou l’ autocar tout au long du voyage auquel elle se rapporte .’
neighbor target	‘L’ original de la feuille de route doit se trouver à bord du véhicule *pendant toute la durée du voyage pour lequel elle a été établie .*’
translation	‘L’ original de la feuille de route doit se trouver à bord de l’ autobus ou de l’ autocar *pendant toute la durée du voyage pour lequel elle a été établie .*’
reference	‘L’ original de la feuille de route doit se trouver à bord de l’ autobus ou de l’ autocar *pendant toute la durée du voyage pour lequel elle a été établie .*’

Table 2: A comparison of model outputs on a sample from the JRC-Acquis dataset. This model was trained using IDF based sentence level retrieval with Conditional Source Target Memory. The different colors and text formatting (underlined, italic, bold) represent different overlapping phrases within the model output, the retrieved target and the reference translation.

ing training, and neighbors corresponding to all n-grams during decoding. This was meant to limit memory requirements and enable the model to fit on P100s during training. We used n-gram width, $n = \{6, 10, 18\}$, for both IDF and dense vector based n-gram retrieval approaches. For scalability reasons, we restricted the retrieval set to the in-domain training corpus, i.e. neighbors for all train, dev and test sentences in the JRC-Acquis corpus were retrieved from the JRC-Acquis training split, and similarly for the other datasets.

3.2 Hyper-parameters and Optimization

For our baseline model we use the standard Transformer Base model (Vaswani et al., 2017). For the semi-parametric model, all our hyper-parameters for attention (8 attention heads), model dimensions (512) and hidden dimensions (2048), including those used in the CSTM memory are equivalent to Transformer Base.

The Transformer baselines are trained on 16 GPUs, with the learning rate, warm-up schedule and batching scheme described in (Vaswani et al., 2017). The semi-parametric models were trained on 32 GPUs with each replica split over 2 GPUs, one to train the translation model and the other for computing the CSTM. We used a conservative learning rate schedule (3, 40K) (Chen et al., 2018) to train the semi-parametric models.

We apply a dropout rate(Srivastava et al., 2014) of 0.1 to all inputs, residuals, attentions and ReLU connections in both models. We use Adam (Kingma and Ba, 2014) to train all models, and apply label smoothing with an uncertainty of 0.1 (Szegedy et al., 2015). In addition to the transformer layers, layer normalization (Ba et al., 2016) was applied to the output of the CSTM. All models are implemented in Tensorflow-Lingvo (Shen et al., 2019).

4 Results

We compare the test performance of a multi-domain Transformer Base and our semi-parametric model using dense vector based n-gram retrieval and CSTM in Table 1. Apart from significantly improving performance by

source	‘Consciousness also is **what makes life worth living .**’
neighbor source	‘So in the last 10 years and the hope for the future , we ’ve seen the beginnings of a science of positive psychology , a science of **what makes life worth living .**’
baseline translation	‘La conscience est aussi ce *qui rend la vie valable .*’
neighbor target	‘Donc , depuis 10 ans , et , espérons-le , à l’ avenir nous assistons à l’ émergence d’ une science de la psychologie positive : une science qui fait en sorte **que la vie** vaille **la peine d’ être vécue .**’
translation	‘La conscience est aussi ce qui fait **que la vie** vaut **la peine d’ être vécue .**’
reference	‘La conscience est aussi ce qui fait **que la vie** vaut **la peine d’ être vécue .**’

Table 3: A comparison of model outputs on a sample from IWSLT. This model was trained using IDF based n-gram retrieval with Conditional Source Target Memory. N-Gram level retrieval results in finding neighbors with high local overlap, even when the rest of the sentences are dissimilar.

source	‘I was expecting to see **gnashing of teeth** and a fight breaking out at the gate .’
neighbor source	‘One could almost hear the collective **gnashing of teeth** in the US , especially in the Congress .’
baseline translation	‘J’ espérais voir *des dents brûlantes* et une bataille éclater à la porte .’
neighbor target	‘On a presque entendre **les dents grincer** aux États-Unis , surtout au Congrès .’
translation	‘Je m’ attendais à voir **des grincements de dents** et une bagarre éclater à la porte .’
reference	‘Je m’ attendais à voir **des grincements de dents** et une bagarre éclater à la porte .’

Table 4: A comparison of model outputs on a sample from WMT. This model was trained using IDF based n-gram retrieval with Conditional Source Target Memory. N-Gram level retrieval results in finding neighbors with high local overlap, even when the rest of the sentences are dissimilar.

more than 10 BLEU points on JRC-Acquis, 2-3 BLEU on OpenSubtitles and IWSLT, we notice a moderate gain of 0.5 BLEU points on WMT 14.

4.1 Comparison of retrieval strategies

We compare the performance of all 3 retrieval strategies in Table 1. The semi-parametric model with sentence level retrieval out-performs the seq2seq model by a huge margin on JRC-Acquis and OpenSubtitles. A sample from the JRC-Acquis dataset where the semi-parametric approach improves significantly over the neural approach is included in Table 2. We notice that there is a lot of overlap between the source sentence and the retrieved source, resulting in the semi-parametric model copying large chunks from the retrieved target. However, its performance is noticeably worse on WMT and IWSLT. Based on a manual inspection of the retrieved candidates, we attribute these losses to retrieval failures. For broad domain datasets like WMT and IWSLT sentence retrieval fails to find good candidates.

Switching to n-gram level retrieval brings the WMT performance close to the seq2seq approach, and IWSLT performance to 2 BLEU points above the baseline model. Representative examples from IWSLT and WMT where n-gram retrieval improves over sentence level retrieval can be seen in Tables 3 and 4. Despite the majority of the retrieved neighbor having nothing in common with the source sentence, n-gram retrieval is able to find neighbors that contain local overlaps.

Using dense n-gram retrieval allows us to move beyond lexical overlap and retrieve semantically similar n-grams even when the actual tokens are different. As a result, dense n-gram retrieval improves performance over all our models on all 4 datasets. An illustrative example from WMT is included in Table 5.

source	‘The artist **died** last Sunday at the age of 71 .’
neighbor source	‘A former minister George Thomson **passed away** last week at the age of 87 .’
baseline translation	‘L’ artiste est *mort* dimanche dernier à l’ âge de 71 ans .’
neighbor target	‘George Thomson , ancien ministre , est **décédé** la semaine dernière à l’ âge de 87 ans .’
translation	‘L’ artiste est **décédé** dimanche dernier à l’ âge de 71 ans .’
reference	‘L’ artiste est **décédé** dimanche dernier , à l’ âge de 71 ans . ’

Table 5: A comparison of model outputs on a sample from WMT. This model was trained using dense vector based n-gram retrieval with Conditional Source Target Memory. Dense vector based n-gram retrieval allows us to find semantically similar phrases, even when the lexical context is dissimilar.

Model	newstest 14	IWSLT 2015	OpenSub	JRC-Acquis
No Memory	41.92	43.17	26.67	56.19
TM	41.64	44.32	27.38	64.25
CTM	41.87	44.76	27.74	65.18
CSTM	**42.41**	**45.02**	**29.06**	**66.92**

Table 6: Comparison of test translation quality (BLEU) with different memory architectures. All models are trained on the Dense N-Gram Multi-Domain dataset. CSTM corresponds to the proposed Conditional Source Target Memory. CTM corresponds to Conditional Target Memory, where we ignore the retrieved sources while encoding the retrieved targets, and directly attend the encoding of the current source, X. TM corresponds to encoding the retrieved targets without any context.

4.2 Memory Ablation Experiments

We report the performance of the various memory ablations in Table 6. We first remove the retrieved sources, X^i, from the CSTM, resulting in an architecture where the encoding of a retrieved target, Y^i, only incorporates information from the source X, represented by the row CTM in the table. This results in a clear drop in performance on all datasets. We ablate further by removing the attention to the original source X, resulting in a slightly smaller drop in performance (represented by TM). These experiments indicate that incorporating context from the sources significantly contributes to performance, by allowing the model to distinguish between relevant context and noise.

5 Non-Parametric Adaptation

Using a semi-parametric formulation for MT opens up the possibility of non-parametric adaptation. The biggest advantage of this approach is the possibility of training a single massively customizable model which can be adapted to any new dataset or document at inference time, by just updating the retrieval dataset.

We evaluate our model’s performance on non-parametric adaptation and compare it against a fully fine-tuned model. In this setting, we train a baseline model and a dense n-gram based semi-parametric model on the WMT training corpus. We only retrieve and train on examples from the WMT corpus during training. We use the same hyper-parameters and training approaches used for the multi-domain experiments, as in Section 3.

The baseline model is then fine-tuned independently on JRC-Acquis, OpenSubtitles and IWSLT. The semi-parametric model is adapted non-parametrically to these three datasets, without any parameter updates. Adaptation is achieved via the retrieval mechanism - while evaluating, we retrieve similar examples from their respective training datasets. To quantify headroom, we also fine-tune our semi-parametric model on each of these datasets.

The results for non-parametric adaptation experiments are documented in Table 7. We notice that the non-parametric adaptation strategy significantly out-performs the base model on all 4 datasets. More importantly, the we find that our approach is capable of adapting to both, JRC-Acquis and OpenSubtitles, via just the retrieval apparatus, and out-performs the fully fine-tuned model indicating that non-parametric adaptation might be a reasonable approach when adapting to a lot of narrow domains or documents.

Adaptation Strategy	newstest 14	IWSLT 2015	OpenSub	JRC-Acquis
Base	41.16	39.75	22.92	53.1
Fine-tuning	-	**42.87**	26.55	62.99
Non-Parametric (NP)	**41.57**	40.95	**27.09**	**64.93**
NP + Fine-tuning	-	43.82	29.12	66.72

Table 7: Comparison of test translation quality (BLEU) with different adaptation strategies. The base model (Transformer Base) is trained on the WMT dataset. Fine-tuning corresponds to fine-tuning based adaptation, where we initialize the domain-specific model from the WMT pre-trained Base model, and fine-tune it on the in-domain dataset for a few epochs. Non-parametric corresponds to our semi-parametric NMT model, adapted to in-domain data during inference by retrieving neighbors from the in-domain training corpus.

In-domain fine-tuning on top of non-parametric adaptation further improves by 2 BLEU points on all datasets, increasing the gap even further with the seq2seq adapted models.

6 Related Work

Tools incorporating information from individual translation pairs, or translation memories (Lagoudaki; Reinke, 2013), have been widely utilized by human translators in the industry. There have been a few efforts attempting to combine non-parametric methods with NMT (Gu et al., 2017; Zhang et al., 2018b; Cao and Xiong, 2018), but the key difference of our approach is the introduction of local, sub-sentence level similarity in the retrieval process, via n-gram level retrieval. Combined with our architectural improvements, motivated by the target encoder and gated attention from (Cao and Xiong, 2018) and the extended transformer model from (Zhang et al., 2018a), our semi-parametric NMT model is able to outperform purely neural models in broad multi-domain settings.

Some works have proposed using phrase tables or the outputs of Phrase based MT within NMT (Dahlmann et al., 2017; Zhang et al., 2017; Zhou et al., 2017). While this reduces the noise present within the retrieved translation pairs, it requires training and maintaining a separate SMT system which might introduce errors of its own.

Another class of methods requires fine-tuning the entire NMT model to every instance at inference time, using retrieved examples (Farajian et al., 2017b; Wuebker et al., 2015), but these approaches require running expensive gradient descent steps before every translation.

Beyond NMT, there have been a few other attempts to incorporate non-parametric approaches into neural generative models (Guu et al., 2018; Hayati et al., 2018; Weston et al., 2018). This strong trend towards combining neural generative models with non-parametric methods is an attempt to counter the weaknesses of neural networks, especially their failure to remember information from individual training instances and the diversity problem of seq2seq models (Vijayakumar et al., 2016; Jiang and de Rijke, 2018).

While our approach relies purely on retrieval from the training corpus, there has been quite a lot of work, especially on Question Answering, that attempts to find additional signals to perform the supervised task in the presence of external knowledge sources (Chen et al., 2017; Wang et al., 2018). Retrieving information from unsupervised corpora by utilizing multilingual representations (Guo et al., 2018) might be another interesting extension of this work.

7 Conclusions and Future Work

We make two major technical contributions in this work which enable us to improve the quality of semi-parametric NMT on broad domain datasets. First, we propose using n-gram retrieval, with standard Inverse Document Frequency similarity and with dense vector representations, that takes into account local sentence similarities that are critical to translation. As a result we are able to retrieve useful candidates even for broad domain tasks with little train-test overlap. Second, we propose a novel architecture to encode retrieved source-target pairs, allowing the model to distinguish useful information from noise by encoding the retrieved targets in context of the current translation task.

We demonstrate, for the first time, that semi-parametric methods can beat neural models by significant margins on multi-domain Machine Translation. By successfully training semi-parametric neural models on a broad domain dataset (WMT), we also open the door for non-parametric adaptation, showing huge improvements on new domains without any parameter updates.

While we constrain this work to retrieved context, our architecture can be utilized to incorporate information from other sources of context, including documents, bilingual dictionaries etc. Using dense representations for retrieval also allows extending semi-parametric neural methods to other input modalities, including images and speech.

With this work, we hope to motivate further investigation into semi-parametric neural models for and beyond Neural Machine Translation.

Acknowledgments

We would like to thank Naveen Arivazhagan, Macduff Hughes, Dmitry Lepikhin, Mia Chen, Yuan Cao, Ciprian Chelba, Zhifeng Chen, Melvin Johnson and other members of the Google Brain and Google Translate teams for their useful inputs and discussions. We would also like to thank the entire Lingvo development team for their foundational contributions to this project.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Qian Cao and Deyi Xiong. 2018. Encoding gated translation memory into neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3042–3047. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Leonard Dahlmann, Evgeny Matusov, Pavel Petrushkov, and Shahram Khadivi. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. *arXiv preprint arXiv:1708.03271*.

M Amin Farajian, Marco Turchi, Matteo Negri, Nicola Bertoldi, and Marcello Federico. 2017a. Neural vs. phrase-based machine translation in a multi-domain scenario. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 280–284.

M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017b. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search engine guided nonparametric neural machine translation. *arXiv preprint arXiv:1705.07267*.

Mandy Guo, Qinlan Shen, Yinfei Yang, Heming Ge, Daniel Cer, Gustavo Hernandez Abrego, Keith Stevens, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Effective parallel corpus mining using bilingual sentence embeddings. *arXiv preprint arXiv:1807.11906*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic, and Graham Neubig. 2018. Retrieval-based neural code generation. *arXiv preprint arXiv:1808.10025*.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. *arXiv preprint arXiv:1704.07431*.

Shaojie Jiang and Maarten de Rijke. 2018. Why are sequence-to-sequence models so dull? *EMNLP 2018*, page 81.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and* *demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Elina Lagoudaki. Translation memories survey 2006: Users perceptions around tm use.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Uwe Reinke. 2013. State of the art in translation memory technology. *Translation: Computation, Corpora, Cognition*, 3(1).

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, et al. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. *arXiv preprint cs/0609058*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Jörg Tiedemann. Parallel data, tools and interfaces in opus.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering. In *Conference on Artificial Intelligence*.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical incremental adaptation for statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1065.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018a. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542. Association for Computational Linguistics.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2017. Improving neural machine translation through phrase-based forced decoding. *arXiv preprint arXiv:1711.00309*.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018b. Guiding neural machine translation with retrieved translation pieces. *arXiv preprint arXiv:1804.02559*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. *arXiv preprint arXiv:1704.06393*.

Online Distilling from Checkpoints for Neural Machine Translation

Hao-Ran Wei, Shujian Huang[*]**, Ran Wang, Xin-yu Dai** and **Jiajun Chen**
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
{weihr, huangsj, wangr, daixy, chenjj}@nlp.nju.edu.cn

Abstract

Current predominant neural machine translation (NMT) models often have a deep structure with large amounts of parameters, making these models hard to train and easily suffering from over-fitting. A common practice is to utilize a validation set to evaluate the training process and select the best checkpoint. Average and ensemble techniques on checkpoints can lead to further performance improvement. However, as these methods do not affect the training process, the system performance is restricted to the checkpoints generated in the original training procedure. In contrast, we propose an online knowledge distillation method. Our method on-the-fly generates a teacher model from checkpoints, guiding the training process to obtain better performance. Experiments on several datasets and language pairs show steady improvement over a strong self-attention-based baseline system. We also provide analysis on data-limited setting against over-fitting. Furthermore, our method leads to an improvement on a machine reading experiment as well.

1 Introduction

Neural Machine Translation (NMT) (Cho et al., 2014; Sutskever et al., 2014) has been rapidly developed during the past several years. For further performance improvement, deeper and more expressive structures (Johnson et al., 2017; Barone et al., 2017b; Gehring et al., 2017; Vaswani et al., 2017) have been exploited. However, all of these models have more than hundreds of millions of parameters, which makes the training process more challenging. During the training of NMT models, we notice the following two problematic phenomena: First, the training process is unstable. This is evidenced by the decreasing of training loss with fluctuate performance on the validation set. Second, the performance on validation set usually begins to worsen after several epochs, while the training loss keeps decreasing, which suggests the model being at risk of over-fitting.

In order to alleviate these issues, the common practice is to periodically evaluate models on a held-out set (with each evaluated model saved as a *checkpoint*). Training is terminated when m consecutive checkpoints show no improvement and select the checkpoint with best evaluation score as the final model. Further improvement can be achieved by utilizing more checkpoints, by *smoothing*, which averages these checkpoints' parameters to generate more desirable parameters (Sennrich et al., 2016a); or by *ensemble*, which averages these checkpoints' output probabilities at every step during inference (Chen et al., 2017).

However, we notice that all of these methods have a limitation. Once the training process gets parameters with poor performance, selecting, smoothing or ensemble from the checkpoints in this process may have limited generalization performance as well. We impute the limitation to the "offline" property of these methods. In other words, only employing checkpoints after training cannot affect the original training process.

In this paper, we propose to utilize checkpoints to lead the training process. Our method is carried out in a *knowledge distillation* manner. At each training step, because being evaluated on the held-out validation data, the best checkpoint up to the current training step can be seen as a model with the best generalization ability so far. Therefore, we employ this checkpoint as the teacher model, and let the current training model, as the student, learn from the output probability distributions of the teacher model, as well as truth translations in the training data. Such kind of knowledge distillation is performed on-the-fly because the teacher

---

[*] Corresponding Author.

Proceedings of NAACL-HLT 2019, pages 1932–1941
Minneapolis, Minnesota, June 2 - June 7, 2019.

model could always be updated once any latest better checkpoint is generated. We call our method **O**nline **D**istillation from **C**heckpoints (ODC).

We conduct experiments on four translation tasks (including two low-resource tasks), and one machine reading comprehension task. All the results demonstrate that our ODC method can achieve improvement upon strong baseline systems. ODC also outperforms checkpoint smoothing and ensemble methods, without extra cost during inference. We can achieve further improvement by combining ODC with those methods.

Major contributions of our work include:

1. In contrast to checkpoint smoothing and ensemble which do not affect the training process, we explore the way to distill knowledge from checkpoints to lead the training process in an on-the-fly manner(§3.1, §3.2). We obtain better performance by replacing the best checkpoint with moving average parameters at that step. (§3.3)

2. We conduct experiments on four translation tasks, including two low resource tasks. In all the tasks our method outperforms strong baseline systems (§4.2, §4.3). We also conduct an experiment on machine reading comprehension task and the result shows that our method can be applied to other tasks too (§4.4).

3. We conduct comprehensive analysis and show that our method can significantly alleviate over-fitting issue in low-resource condition (§5.1), and help to find a wider minimum which brings better generation (§5.2).

2 Background

2.1 Neural Machine Translation

Neural Machine Translation (NMT) systems learn a conditional probability $P(Y|X)$ for translating a source sentence $X = (x_1, ..., x_M)$ to a target sentence $Y = (y_1, ..., y_N)$, in which x_i and y_j are the i-th word and j-th word in sentence X and Y, respectively. An NMT model usually consists of an encoder (parameterized by θ_{enc}) and a decoder (parameterized by θ_{dec}). The encoder transforms a sequence of source tokens into a sequence of hidden states:

$$H(X) = (h_1, ..., h_M) = f_{\text{enc}}(X; \theta_{\text{enc}}). \qquad (1)$$

The decoder of NMT is usually a network computing the conditional probability of each target words y_j based on its previous words and the source sentence:

$$p(y_j|y_{<j}, X) \propto \exp(f_{\text{dec}}(y_{<j}, s_j, H(X); \theta_{\text{dec}})), \qquad (2)$$

where s_j is the hidden state of decoder at time step j, p is the distribution of NMT model and θ is all the parameters of NMT model.

The standard way to train an NMT model is to minimize the cross-entropy between the one-hot distribution of the target sentence and the NMT model's output distribution:

$$\begin{aligned} \mathcal{L}(\theta) = &-\sum_{j=1}^{N}\sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y_j = k\} \qquad (3) \\ &\cdot \log p(y_j = k|y_{<j}, X; \theta), \\ \theta^* &= \arg\min_{\theta} \mathcal{L}(\theta), \qquad (4) \end{aligned}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $\mathcal{V}$ is the target vocabulary.

2.2 Knowledge Distillation in Neural Machine Translation

Knowledge distillation is a class of methods which transfers knowledge from a pre-trained *teacher* model $\mathcal{T}$, to a *student* model $\mathcal{S}$. The teacher model can be a model with large capacity (Bucila et al., 2006) or an ensemble of several models (Hinton et al., 2015). In knowledge distillation, the student model learns to match the predictions of the teacher model. Concretely, assuming that we learn a classification model (parameterized by θ) on a set of training samples in the form of (x, y) with $|\mathcal{V}|$ classes. Instead of minimizing the cross-entropy loss between one-hot label y and model's output probability $p(y|x;\theta)$, knowledge distillation uses the teacher model's distribution $q(\cdot|x)$ as "soft targets" and optimizes the loss:

$$\begin{aligned} \mathcal{L}_{\text{KD}}(\theta) = -\sum_{k=1}^{|\mathcal{V}|} & q(y = k|x; \theta_{\mathcal{T}}) \qquad (5) \\ & \log p(y = k|x; \theta), \end{aligned}$$

where $\theta_{\mathcal{T}}$ parameterizes the teacher model and $p(\cdot|x)$ is the distribution of the student model.

Kim and Rush (2016) proposed that, as the loss of NMT model (Equation 4) can be factored into minimizing cross-entropy loss between the target

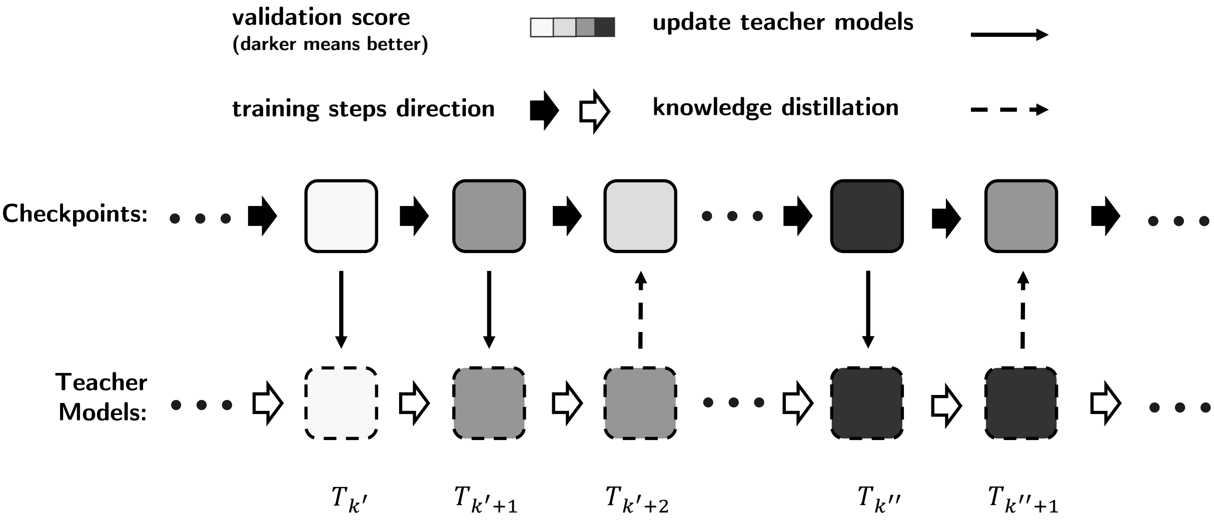


Figure 1: Illustration of online distillation from checkpoints(ODC). Darker color means better performance on validation data. In validation step $T_{k'}$, ODC selects the current best checkpoints as the teacher model; while in the next validation step $T_{k'+1}$, the training generates a better checkpoint, and use it to update the teacher model. In validation step $T_{k'+2}$, training model's performance declines, and the teacher model is used to lead the training of the model. The similar situation happens during the entire training process.

words and word-level probabilities of the NMT model for every position at target side, knowledge distillation on multi-class classification can be naturally applied. They defined word-level knowledge distillation (W-KD) on a sentence as:

$$\mathcal{L}_{\text{W-KD}}(\theta) = -\sum_{i=j}^{N}\sum_{k=1}^{|\mathcal{V}|} q(y_j = k|y_{<j}, H(x); \theta_{\mathcal{T}}) \cdot \log p(y_j = k|y_{<j}, H(x); \theta), \tag{6}$$

where $\mathcal{V}$ is the target vocabulary.

They further proposed sequence-level knowledge distillation (S-KD), which optimizes the student model by matching the predictions of the teacher model in the probability distribution over the space of all possible target sequences:

$$\mathcal{L}_{\text{S-KD}}(\theta) = \sum_{Y \in \tau} q(Y|X; \theta_{\mathcal{T}}) \log p(Y|X; \theta), \tag{7}$$

where τ is the space of target side sentences. As summing over exponential numbers of samples here is intractable, they proposed to train student model on samples generated by teacher model as an approximation.

3 Online Distillation from Checkpoints

3.1 Online Knowledge Distillation

Traditional knowledge distillation maintains a static teacher throughout the training process, which not only requires one pre-training process to obtain the teacher model, but also limits the power of leading the training process.

In contrast, we are aiming at a more integrated process where the teacher model does not come from a separate training process, but from the current training routine itself. More specifically, we update the teacher along with the training process, so the distilled knowledge could be updated when a stronger model comes out. Figure 1 illustrates the paradigm of our method.

In generation tasks, the knowledge distillation could be performed at the word-level or sequence-level. In this paper, we focus on the word-level distillation because this distillation only needs forced teaching, which could be performed efficiently together with the training of the student model compared to generating translations from teacher model. It is more computational-friendly, especially when the NMT models are built with parallelizable convolution (Gehring et al., 2017) or self-attention structures (Vaswani et al., 2017).

3.2 Online Distillation from Best Checkpoint

Observed from the training process of NMT models, performance on the validation set does not improve monotonically. When the performance of the training model on the validation set declines, we could always select the best checkpoint so far as the teacher, because it has the best generalization performance. Specially, when the best checkpoint is generated at the current time step, we only update the teacher model but perform no distillation. Figure 1 gives an illustration of this process.

The online distillation process is summarized in Algorithm 1. We use t to denote the training step and θ_t to denote the parameters at time step t. We denote T_k as the time step the k-th time when the model is evaluated on validation and ΔT as the validation interval, for which $T_{k+1} = T_k + \Delta T$. Let $\hat{T}_k$ ($\hat{T}_k \leq T_k$) be the time step when the best checkpoint is obtained up to T_k, and $\theta_{\mathcal{T}}$ as the teacher's parameters to lead the following training process. If the current checkpoint is the best checkpoint so far, i.e. $\hat{T}_k = T_k$, we update the teacher to be this new checkpoint $\theta_{\mathcal{T}} = \theta_{\hat{T}_k}$ (in Line 16 and 20). The loss for the training process at time step t ($T_k < t < T_{k+1}$) is defined as follows:

$$\mathcal{L}_t(\theta) = \begin{cases} \mathcal{L}(\theta) & \hat{T}_k = T_k \\ \mathcal{L}(\theta) + \mathcal{L}_{\text{W-KD}}(\theta) & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathcal{L}(\theta)$ and $\mathcal{L}(\theta)_{\text{W-KD}}$ is defined in Equation 4 and 7, respectively (in Line 5-8).

3.3 Integrated with Mean Teacher

Knowledge distillation usually works better when teacher models have better performance. As Tarvainen and Valpola (2017) proposed in their work, averaging model parameters over training steps tends to produce a more accurate model that using final parameters directly. They called this method as *Mean Teacher*.

Following Tarvainen and Valpola (2017), besides updating parameters, we maintain the exponential moving average (EMA) of the model parameters as:

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t, \tag{9}$$

where t is the update step, θ is the parameters of the training model and θ' the parameters of EMA. α is the decay weight which is close to 1.0, and typically in multiple-nines range, i.e., 0.999, 0.9999. By doing so, at each timestep t, parameters of NMT model θ_t has their corresponding EMA parameters θ'_t.

Whenever we update teacher model $\theta_{\mathcal{T}}$ with the current best checkpoint, we can use its EMA parameters instead (in Line 17-18). It can further improve the generalization ability of the teacher model, and bring a better performance of knowledge distillation. We will show in §4.2 that using meaning teacher indeed achieves better performance.

Algorithm 1: Online Distillation from Checkpoints

```
1  Input: validation interval $\Delta T$; validation count $k$; EMA decay weight $\alpha$; initial model parameters $\theta_0$
2  Initialization: $k = 0$; $t = 0$; $T_0 = -1, \hat{T}_0 = -1$; $\theta_{\mathcal{T}} = \emptyset$; $\theta'_0 = \theta_0$; $\mathcal{L}_0(\theta) = \mathcal{L}(\theta)$
3  while not reach stopping criteria do
4      repeat
5          if $\hat{T}_k = T_k$ then
6              $\mathcal{L}_t(\theta) = \mathcal{L}(\theta)$
7          else
8              $\mathcal{L}_t(\theta) = \mathcal{L}(\theta) + \mathcal{L}_{\text{W-KD}}(\theta)$
9          minimize $\mathcal{L}_t(\theta)$ and update $\theta_t$ ;
10         $\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$;
11         $t = t + 1$ ;
12     until $t \bmod \Delta T == 0$;
13     $T_{k+1} = t$;
14     evaluate on validation set;
15     if get better checkpoint then
16         $\hat{T}_{k+1} = t$;
17         if use EMA as teacher then
18             $\theta_{\mathcal{T}} = \theta'_t$;
19         else
20             $\theta_{\mathcal{T}} = \theta_t$;
21     else
22         $\hat{T}_{k+1} = \hat{T}_k$;
23     $k = k + 1$;
```

4 Experiments

4.1 Setups

To evaluate the effectiveness of our method, we conduct experiments on four machine translation tasks: NIST Chinese-English, WMT17 Chinese-English, IWSLT15 English-Vietnamese, and WMT17 English-Turkish. We conduct experiments based on an open source implementation of Transformer (Vaswani et al., 2017) model in NJUNMT-pytorch[1]. For all the translation experiments, we use SacreBLEU [2] to report reproducible BLEU scores.

We also present an experiment on machine reading comprehension, showing our method could also be applied to other tasks.

Datasets For NIST Chinese-English translation task, training data consists of 1.34M LDC sentence pairs[3], with 40.8M Chinese words and 45.8M English words, respectively. We use NIST2003 dataset set as the validation set and NIST 2004,

---

[1] https://github.com/whr94621/NJUNMT-pytorch

[2] https://github.com/awslabs/sockeye/tree/master /sockeye_contrib/sacrebleu

[3] The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

SYSTEMS	NIST Chinese-English					
	NIST03	NIST04	NIST05	NIST06	Average	Δ
RNNSearch (Zhang et al., 2018b)	36.59	39.57	35.56	35.29	-	-
Transformer-base(Yang et al., 2018)	42.23	42.17	41.02	-	-	-
baseline	43.78	44.26	40.97	38.93	41.39	-
baseline + LKS	44.12	44.87	41.59	39.22	41.89	+0.50
baseline + BKS	44.23	44.98	41.62	39.74	42.11	+0.73
baseline + BKE	44.30	45.01	41.86	40.05	42.31	+0.92
ODC	45.33	45.18	42.60	39.67	42.48	+1.10
ODC + LKS	45.05	45.49	42.99	40.48	42.99	+1.60
ODC + BKS	**45.35**	45.49	43.21	39.96	42.89	+1.50
ODC + BKE	45.34	**45.92**	**43.35**	40.30	**43.19**	**+1.80**
ODC-EMA	45.52	45.72	43.01	**40.65**	43.13	+1.74

Table 1: Case-insensitive BLEU scores of Chinese-English translation on NIST datasets. "Average"means average scores on NIST04, 05 and 06.

2005, 2006 as test sets. We filter out sentence pairs whose source or target side contain more than 50 words. We use BPE (Sennrich et al., 2016b) with 30K merge operations on both sides.

For WMT17 Chinese-English translation task, we use the pre-processed version released by WMT[4]. We only use CWMT part of WMT Corpus. We use newsdev2017 as the validation set and newstest2017 s the test set. We learn a BPE model with 32K merge operations and keep all the BPE tokens in the vocabulary. We limit the maximal sentence length as 100 after BPE segmentation.

For IWSLT15 English-Vietnamese translation task, we directly use the pre-processed data used in Luong and Manning (2015) [5], which has 133K sentence pairs, with 2.70M English words and 3.31M Vietnamese words. We use the released validation and test set, which has 1553 and 1268 sentences respectively. Following the settings in Huang et al. (2017), the Vietnamese and English vocabulary size are 7,709 and 17,191, respectively.

For WMT17 English-Turkish translation task, We use the pre-processed data released by WMT17[6]. It has 207K sentence pairs, with 5.21M English words and 4.63 Turkish words. We use newstest2016 as our validation set and newstest2017 as the test set. We use joint BPE segmentation (Sennrich et al., 2017) to process the whole training data. The merge operations are 16K.

Implementation Details Without specific statement, we follow the *transformer_base_v1* hyperparameters settings [7], with 6 layers in both encoder and decoder, 512 hidden units and 8 attention heads in multi-head attention mechanism and 2048 hidden units in feed-forward layers. Parameters are optimized using Adam(Kingma and Ba, 2014). The initial learning rate is set as 0.1 and scheduled according to the method proposed in Vaswani et al. (2017), with warm-up steps as 4000.

We periodically evaluate the training model on the validation set by doing translation and compute the BLEU scores. We stop training when 50 subsequent of BLEU scores on validation set do not get improvement. We use beam search with beam size as 5.

4.2 Evaluation on Chinese-English Translation Tasks

We first evaluate the capability of our method for improving performance when there are plenty of training data. We conduct experiments on both NIST and WMT17 Chinese-English Translation tasks.

Results on NIST Dataset We compare our method with several ways to utilize checkpoints[8]:

- **last-k-smoothing**: After training the baseline model, we average the parameters of the last k checkpoints as the final model.
- **best-k-smoothing**: Average the parameters of the best k checkpoints, instead of the last k,

---

[4] http://data.statmt.org/wmt18/translation-task/preprocessed/zh-en/

[5] https://github.com/tefan-it/nmt-en-vi

[6] http://data.statmt.org/wmt17/translation-task/preprocessed/tr-en/

[7] https://github.com/tensorflow/tensor2tensor/blob/v1.3.0/tensor2tensor/models/transformer.py

[8] We set $k = 5$ in this experiments.

as the final model. In this case, checkpoints may have better performance but higher variance which could be harmful to parameters averaging.

- **best-k-ensemble**: Do ensemble inference (average the output probabilities) with the best k checkpoints (Chen et al., 2017).

As shown in Table 1, our baseline is comparable to the other two recent published results (Zhang et al. (2018b), Yang et al. (2018)). In consistent with Chen et al. (2017), using checkpoints for smoothing or ensemble does improve the baseline system. Using EMA parameters also improve the baseline system as well, which is in consist with (Tarvainen and Valpola, 2017).

Compared to the baseline, our approach ODC brings translation improvement across different test sets and achieves 42.48 BLEU scores on average(+1.09 BLEU v.s. baseline). This result confirms that using best checkpoint as teacher indeed helps improving the performance of the translation model.

Besides, ODC is comparable to the best results among smoothing and ensemble on baseline's checkpoints (achieved by best-k-ensemble). Considering that best-k-ensemble needs to decode with k models, while ODC decodes only one, our model enjoys a better efficiency. Furthermore, we can achieve further improvement by combining these methods on checkpoints generated by ODC.

Results also show that ODC-EMA (§3.3) could achieve additional improvement from ODC itself (43.13 v.s. 42.48 BLEU), demonstrating that using EMA of the best checkpoint instead can bring better knowledge distillation performance, as it generates a better teacher model.

Results on WMT17 Dataset We present the results on WMT17 Chinese-English translation task in Table 2. We report the results of the baseline, ODC and a recent result published by Zhang et al. (2018c). To make a fair comparison, we follow the experiment setting in Zhang et al. (2018c). The experiment results show similar trends with those on the NIST datasets. Applying ODC leads to the result of 24.22 BLEU, which is 0.85 BLEU higher compared with baseline.

SYSTEM	newsdev2017	newstest2017
Zhang et al. (2018c)	-	23.01
baseline	21.96	23.37
ODC	**22.24**	**24.22**

Table 2: Case-sensitive BLEU scores on WMT17 Chinese-English Translation

4.3 Evaluation on Low-resource Scenario

We also apply our method to two low resource translation tasks, i.e., IWSLT2015 English-Vietnamese (EN2VI) and WMT17 English-Turkish (EN2TR). Due to the limited amount of training data, models are more likely to suffer from over-fitting. Therefore, we use a higher dropout rate of 0.2 and weight decay, another common technique against over-fitting, with decay weight set as 10^{-3} as the default setting. We implement weight decay as AdamW (Loshchilov and Hutter, 2017) does.

Besides, we further experiment with grid search on the validation set for optimal hyper-parameters of dropout rate and weight decay, which may lead to better results. We adopt a simple heuristic, which first searches an optimal dropout rate, and then further searches weight decay coefficients based on this dropout. We experiment with dropout as 0.2, 0.3, 0.4, and weight decay as 10^{-1}, 10^{-2} and 10^{-3}.

SYSTEMS	EN2VI	EN2TR
Zhang et al. (2018a)	-	12.11
tensor2tensor	28.43	-
baseline	28.56	12.20
baseline w/ grid-search	29.01	12.51
ODC	**29.47**	**12.92**
ODC w/ grid-search	**29.59**	**13.18**

Table 3: Case-sensitive BLEU scores on two low resource translation tasks.

As in Table 3, our baseline is comparable to two recent published results, respectively: EN2TR from Zhang et al. (2018c) and EN2VI from official release tensor2tensor problem[9]. Grid hyper-parameter search does improve the baseline system. ODC leads to better results compared to the baseline, as well as the baseline with grid parameter search. ODC can achieve further improvement after searching for optimal hyper-parameters of dropout and weight decay.

---

[9] https://github.com/tensorflow/tensor2tensor/pull/611

4.4 Evaluation on Machine Reading Comprehension

Although our main research is focused for the task of machine translation, the idea of ODC could be applied to other tasks as well. We experiments on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), a machine reading comprehension task.

SQuAD contains 107,785 human-generated reading comprehension questions, with 536 Wikipedia articles. Each question is associated with a paragraph extracted from an article, and the corresponding answer is a span from this article. A machine reading comprehension model is designed to predict the start and end positions in the article of the answer.

The state-of-the-art machine reading comprehension system also employs a deep neural network structure, which is similar to NMT. We apply our ODC method on **BiDAF++** (Choi et al., 2018), a multi-layer SQuAD model that augments BiDAF (Seo et al., 2016) with self-attention and contextualized embeddings. We evaluate the model after each epoch and implement the knowledge distillation by teaching the student with the output distribution of answer start and end positions predicted by the best checkpoint.

For the results, ODC improves a base BiDAF++ from 76.83 to 77.40, in EM scores, showing that our method can be applied to a broader range of tasks.

5 Analysis

We conduct further analysis to probe into the reasons for the advantages of ODC. We first show that our method can significantly alleviate the overfitting issue in data-limited condition. After that, we show that parameters gained from our method tend to be wider minimums, which represents better generalization.

5.1 Effectiveness on reducing over-fitting

Taking IWSLT15 English-Vietnamese as a test-bed, we analyze whether our method could help handle the over-fitting issue. We first plot the curve of the loss on the validation set at each training step for the different models (in Figure 2, the top curve with rounds). It is easy to see that the loss curve of the baseline increases as the training goes after 50K steps, indicating a severe over-fitting. With better dropout rate and weight decay, the over-fitting is less severe; while with ODC the loss curve shows a more steady trend of decrease, and is almost always under the other two's.

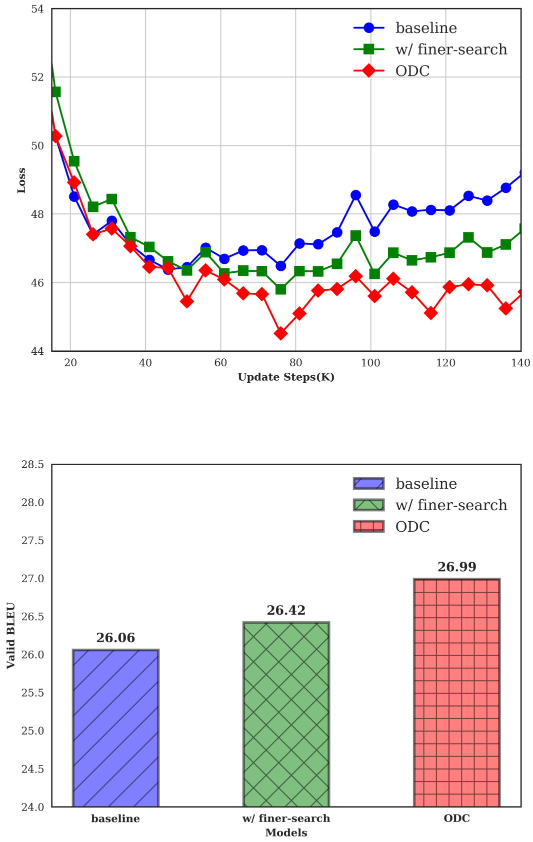


Figure 2: Loss curves (top) and final BLEU scores (bottom) on the validation set of baseline, baseline with grid-search and ODC, respectively.

The final BLEU score on the validation set (Figure 2, bottom) shows corresponding result. The grid search of hyper-parameters improves the BLEU from 26.06 to 26.42 in BLEU, while ODC achieves 26.99.

Both results indicate that our method is more effective at handling the over-fitting problem. We hold that minimizing the cross-entropy between the teacher model and the student model serves as regularization to the training of the student model, which avoids the model getting into over-fitting.

5.2 ODC Brings Wider Minimum

In the training process in Chinese-English tasks, we do not observe obvious over-fitting issue as shown in low resource translation tasks. In this section, we analyze how ODC helps the model generalization. Keskar et al. (2016) proposed that the width

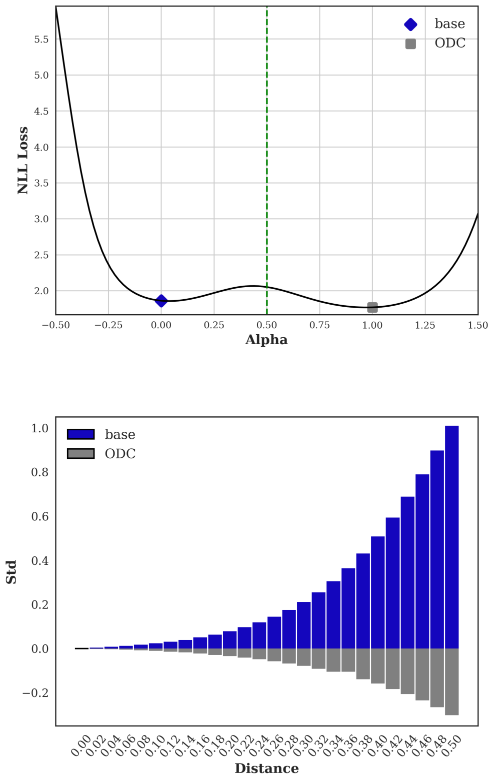


Figure 3: The upper plot shows the validation losses curve along the line segment decided by parameters of baseline and ODC. The bottom plot shows the standard deviations within the neighborhood of baseline and ODC at different distances.

of the minimum in a loss surface is related to its generalization ability. Therefore, we compare the generalization capability between baseline system and our ODC method by exploring around the parameters.

We make use of the visualization technique employed in (Goodfellow and Vinyals, 2014) and analyze the results on the NIST data set. Let θ_{base} and θ_{ODC} denote the final parameters obtained from baseline and ODC. Consider the line:

$$\theta(\alpha) = \alpha \cdot \theta_{\text{ODC}} + (1.0 - \alpha) \cdot \theta_{\text{base}}, \qquad (10)$$

which connects θ_{base} ($\alpha = 0.0$) and θ_{ODC} ($\alpha = 1.0$). We plot the value of Equation 4 as a function of α (normalized by count of words per sentence) with $\theta = \theta(\alpha)$. We draw α from -1.0 to 2.0 at an interval of 0.02. In this way, the width of θ_{base} and θ_{ODC} can be represented as the steepness of the curve nearby. To further quantitatively represent the steepness, we compute the standard deviation of values on this curve within different distances to the two parameters, respectively. We plot them in Figure 3.

From Figure 3 we can see that the loss curve behaves steeper around the parameters of baseline than of ODC. Besides, the standard deviations of losses around the baseline model are consistently higher than ODC within all the distances. It is evident that the parameters of ODC act as a wider minimum c and explains why ODC can lead to a more generalized model.

6 Related Works

6.1 Regularization in NMT

Regularization has broad applications in training NMT models to improve performance and avoid over-fitting. There are some common regularization techniques, such as L^2 normalization and dropout (Srivastava et al., 2014). These methods are simple and easy to implement but need carefully tuning on the validation set. These methods are also orthogonal to our method.

There are also some works to exploit regularization techniques in fine tuning of NMT model. Barone et al. (2017a) proposed a tuneout method which randomly replaces columns of weight matrices of out-of-domain parameter matrices. Khayrallah et al. (2018) shared similar training object with us, as they computed the KL divergence between out-of-domain and in-domain model. Both of their works request a pre-trained teacher model, while we are work on a more general training problem which does not require such kind of model.

6.2 Online Knowledge Distillation

While traditional knowledge distillation requires a static, pre-trained teacher model, online knowledge distillation tends to overcome this problem by selecting or generating a teacher dynamically from scratch.

To the best of our knowledge, Zhang et al. (2017) is the first trial to replace the offline teacher model. They trained peer models to teach each other simultaneously. Compared to their work, our method uses the best checkpoint as the teacher, which avoids introducing extra parameters. Furlanello et al. (2018) tends to update teacher model during the training procedure iteratively, but their method needs to train the teacher model until convergence in each iteration. Instead, our method only needs one phase of training, whose overhead is relatively small.

Lan et al. (2018) using an ensemble of several branches of the model as teacher for computer vision tasks, which only needs one-phase training as well. However, their method relies heavily on the multi-branch structures of the tasks, which are not widely applicable in neural machine translation.

7 Conclusion

In this paper, we propose an online knowledge distillation method with the teacher model generated from checkpoints during the training procedure. Experiments on four machine translation tasks and a machine reading task show that our method outperforms strong baseline systems. Further analysis shows that our method can effectively alleviate the over-fitting issue, and tend to find a wider minimum.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. We also thank Boxing Chen from Alibaba Group for his helpful comments. This work is supported by the National Science Foundation of China (No. 61772261, 61672277) and the Jiangsu Province Research Foundation for Basic Research (No. BK20170074). Part of this work is supported by "13th Five-Yea" All-Army Common Information System Equipment Pre-Research Project (No. 31510040201).

References

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017a. Regularization techniques for fine-tuning in neural machine translation. In *EMNLP 2017*, pages 1489–1494.

Antonio Valerio Miceli Barone, Jindrich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017b. Deep architectures for neural machine translation. In *WMT 2017*.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *CoRR*, abs/1710.03282.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke S. Zettlemoyer. 2018. Quac: Question answering in context. In *EMNLP*.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*.

Ian J. Goodfellow and Oriol Vinyals. 2014. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL 2017*, 5:339–351.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *WMT 2018*, pages 36–44.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Xu Lan, Xiatian Zhu, and Shaogang Gong. 2018. Knowledge distillation by on-the-fly native ensemble. *CoRR*, abs/1806.04606.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *IWSLT 2015*, Da Nang, Vietnam.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural MT systems for WMT17. In *WMT 2017*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *ACL 2016*, pages 371–376.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL 2016*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS 2017*, pages 1195–1204.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *NAACL-HLT*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *ACL*.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018b. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*.

Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. 2017. Deep mutdual learning. *CoRR*, abs/1706.00384.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018c. Regularizing neural machine translation by target-bidirectional agreement. *CoRR*, abs/1808.04064.

Value-based Search in Execution Space for Mapping Instructions to Programs

Dor Muhlgay[1] **Jonathan Herzig**[1] **Jonathan Berant**[1,2]

[1]School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
{dormuhlg@mail,jonathan.herzig@cs,joberant@cs}.tau.ac.il

Abstract

Training models to map natural language instructions to programs, given target world supervision only, requires searching for good programs at training time. Search is commonly done using beam search in the space of partial programs or program trees, but as the length of the instructions grows finding a good program becomes difficult. In this work, we propose a search algorithm that uses the target world state, known at training time, to train a *critic* network that predicts the expected reward of every search state. We then score search states on the beam by interpolating their expected reward with the likelihood of programs represented by the search state. Moreover, we search not in the space of programs but in a more compressed state of program executions, augmented with recent entities and actions. On the SCONE dataset, we show that our algorithm dramatically improves performance on all three domains compared to standard beam search and other baselines.

1 Introduction

Training models that can understand natural language instructions and execute them in a real-world environment is of paramount importance for communicating with virtual assistants and robots, and therefore has attracted considerable attention (Branavan et al., 2009; Vogel and Jurafsky, 2010; Chen and Mooney, 2011). A prominent approach is to cast the problem as semantic parsing, where instructions are mapped to a high-level programming language (Artzi and Zettlemoyer, 2013; Long et al., 2016; Guu et al., 2017). Because annotating programs at scale is impractical, it is desirable to train a model from instructions, an initial world state, and a target world state only, letting the program itself be a latent variable.

Learning from such weak supervision results in a difficult search problem at training time. The model must search for a program that when executed leads to the correct target state. Early work employed lexicons and grammars to constrain the search space (Clarke et al., 2010; Liang et al., 2011; Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Artzi and Zettlemoyer, 2013), but recent success of sequence-to-sequence models (Sutskever et al., 2014) shifted most of the burden to learning. Search is often performed simply using beam search, where program tokens are emitted from left-to-right, or program trees are generated top-down (Krishnamurthy et al., 2017; Yin and Neubig, 2017; Cheng et al., 2017; Rabinovich et al., 2017) or bottom-up (Liang et al., 2017; Guu et al., 2017; Goldman et al., 2018). Nevertheless, when instructions are long and complex and reward is sparse, the model may never find enough correct programs, and training will fail.

In this paper, we propose a novel search algorithm for mapping a sequence of natural language instructions to a program, which extends the standard beam-search in two ways. First, we capitalize on the target world state being available at training time and train a *critic* network that given the language instructions, current world state, and target world state estimates the expected future reward for each search state. In contrast to traditional beam search where states representing partial programs are scored based on their likelihood only, we also consider expected future reward, leading to a more targeted search at training time. Second, rather than search in the space of programs, we search in a more compressed execution space, where each state is defined by a partial program's execution result.

We evaluated our method on the SCONE dataset, which includes three different domains where long sequences of 5 instructions are mapped to programs. We show that while standard beam

Proceedings of NAACL-HLT 2019, pages 1942–1954
Minneapolis, Minnesota, June 2 - June 7, 2019.

search gets stuck in local optima and is unable to discover good programs for many examples, our model is able to bootstrap, improving final performance by 20 points on average. We also perform extensive analysis and show that both value-based search as well as searching in execution space contribute to the final performance. Our code and data are available at `http://gitlab.com/tau-nlp/vbsix-lang2program`.

2 Background

Mapping instructions to programs invariably involves a *context*, such as a database or a robotic environment, in which the program (or logical form) is executed. The goal is to train a model given a training set $\{(x_{(j)} = (c_{(j)}, \mathbf{u}_{(j)}), y_{(j)})\}_{j=1}^{N}$, where c is the context, $\mathbf{u}$ is a sequence of natural language instructions, and y is the target state of the environment after following the instructions, which we refer to as *denotation*. The model is trained to map the instructions $\mathbf{u}$ to a program $\mathbf{z}$ such that executing $\mathbf{z}$ in the context c results in the denotation y, which we denote by $[\![\mathbf{z}]\!]_c = y$. Thus, the program $\mathbf{z}$ is a latent variable we must search for at both training and test time. When the sequence of instructions is long, search becomes hard, particularly in the early stages of training.

Recent work tackled the training problem using variants of reinforcement learning (RL) (Suhr and Artzi, 2018; Liang et al., 2018) or maximum marginal likelihood (MML) (Guu et al., 2017; Goldman et al., 2018). We now briefly describe MML training, which we base our training procedure on, and outperformed RL in past work under comparable conditions (Guu et al., 2017).

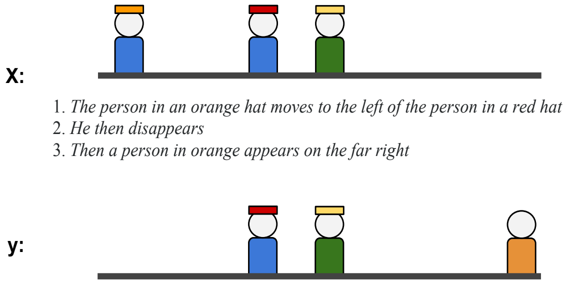
We denote by $\pi_\theta(\cdot)$ a model, parameterized by θ, that generates the program $\mathbf{z}$ token by token from left to right. The model π_θ receives the context c, instructions $\mathbf{u}$ and previously predicted tokens $z_{1\ldots t-1}$, and returns a distribution over the next program token z_t. The probability of a program prefix is defined to be: $p_\theta(z_{1\ldots t} \mid \mathbf{u}, c) = \prod_{i=1}^{t} \pi_\theta(z_i \mid \mathbf{u}, c, z_{1\ldots i-1})$. The model is trained to maximize the MML objective:

$$J_{MML} = \log \prod_{(c,\mathbf{u},y)} p_\theta(y \mid c, \mathbf{u}) = \sum_{(c,\mathbf{u},y)} \log(\sum_{\mathbf{z}} p_\theta(\mathbf{z} \mid \mathbf{u}, c) \cdot R(\mathbf{z})),$$

where $R(\mathbf{z}) = 1$ if $[\![\mathbf{z}]\!]_c = y$, and 0 otherwise (For brevity, we omit c and y from $R(\cdot)$). Typically, the MML objective is optimized with stochastic gradient ascent, where the gradient for an example $(c, \mathbf{u}, y)$ is:

$$\nabla_\theta J_{MML} = \sum_{z} q(z) \cdot R(z) \nabla_\theta \log p_\theta(z|c, \mathbf{u})$$

$$q(z) := \frac{R(z) \cdot p_\theta(z|c, \mathbf{u})}{\sum_{\tilde{z}} R(\tilde{z}) \cdot p_\theta(\tilde{z}|c, \mathbf{u})}.$$



1. *The person in an orange hat moves to the left of the person in a red hat*
2. *He then disappears*
3. *Then a person in orange appears on the far right*

Figure 1: Example from the SCENE domain in SCONE (3 utterances), where people with different hats and shirts are located in different positions. The example contains (from top to bottom): an initial world (the people's properties, from left to right: a blue shirt with an orange hat, a blue shirt with a red hat, and a green shirt with a yellow hat), a sequence of natural language instructions, and a target world (the people's properties, from left to right: blue shirt with a red hat, green shirt with a yellow hat, and an orange shirt).

The search problem arises because it is impossible to enumerate the set of all programs, and thus the sum over programs is approximated by a small set of high probability programs, which have high weights $q(\cdot)$ that dominate the gradient. Search is commonly done with *beam-search*, an iterative algorithm that builds an approximation of the highest probability programs according to the model. At each time step t, the algorithm constructs a beam B_t of at most K program prefixes of length t. Given a beam B_{t-1}, B_t is constructed by selecting the K most likely *continuations* of prefixes in B_{t-1} , according to $p_\theta(z_{1..t}|\cdot)$. The algorithm runs L iterations, and returns all complete programs discovered.

In this paper, our focus is the search problem that arises at training time when training from denotations, i.e., finding programs that execute to the right denotation. Thus, we would like to focus on scenarios where programs are long, and standard beam search fails. We next describe the SCONE dataset, which provides such an opportunity.

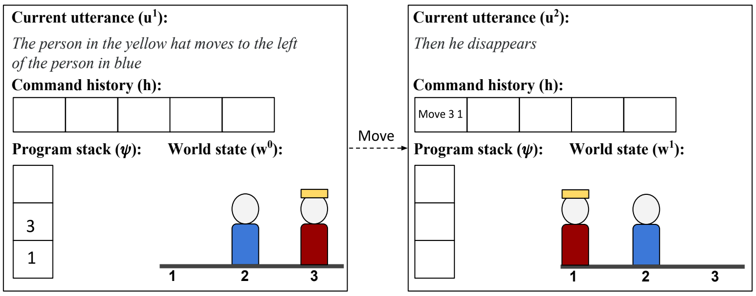


Figure 2: An illustration of a state transition in SCONE. π_θ predicted the action token MOVE. Before the transition (left), the command history is empty and the program stack contains the arguments 3 and 1, which were computed in previous steps. Note that the state does not include the partial program that computed those arguments. After transition (right), the executor popped the arguments from the program stack and applied the action MOVE 3 1: the man in position 3 (a person with a yellow hat and a red shirt) moved to position 1 (to the left of the person with the blue shirt), and the action is added to the execution history with its arguments. Since this action terminated the command, π_θ advanced to the next utterance.

The SCONE dataset Long et al. (2016) presented the SCONE dataset, where a sequence of instructions needs to be mapped to a program consisting of a sequence of executable commands. The dataset has three domains, where each domain includes several objects (such as *people* or *beakers*), each with different properties (such as *shirt color* or *chemical color*). SCONE provides a good environment for stress-testing search algorithms because a long sequence of instructions needs to be mapped to a program. Figure 1 shows an example from the SCENE domain.

Formally, the context in SCONE is a *world* specified by a list of positions, where each position may contain an object with certain properties. A formal language is defined to interact with the world. The formal language contains *constants* (e.g., numbers and colors), *functions* that allow to query the world and refer to objects and intermediate computations, and *actions*, which are functions that modify the world state. Each command is composed of a single action and several arguments constructed recursively from constants and functions. E.g., the command MOVE(HASHAT(YELLOW), LEFTOF(HASSHIRT(BLUE))), contains the action MOVE, which moves a person to a specified position. The person is computed by HASHAT(YELLOW), which queries the world for the position of the person with a yellow hat, and the target position is computed by LEFTOF(HASSHIRT(BLUE)). We refer to Guu et al. (2017) for a full description of the language.

Our goal is to train a model that given an initial world w^0 and a sequence of natural language *utterances* $\mathbf{u} = (u^1, \ldots, u^M)$, will map each utterance u^i to a command z^i such that applying the program $\mathbf{z} = (z^1, \ldots, z^M)$ on w^0 will result in the target world, i.e., $[\![\mathbf{z}]\!]_{w^0} = w^M = y$.

3 Markov Decision Process Formulation

To present our algorithm, we first formulate the problem as a Markov Decision Process (MDP), which is a tuple $(\mathcal{S}, \mathcal{A}, R, \delta)$. To define the state set $\mathcal{S}$, we assume all program prefixes are executable, which can be easily done as we show below. The *execution result* of a prefix $\tilde{z}$ in the context c, denoted by $[\![\tilde{z}]\!]_c^{ex}$, contains its denotation and additional information stored in the executor. Let $\mathcal{Z}_{\text{pre}}$ be the set of all valid programs prefixes. The set of states is defined to be $\mathcal{S} = \{(x, [\![\tilde{z}]\!]_c^{ex}) \mid \tilde{z} \in \mathcal{Z}_{\text{pre}}\}$, i.e., the input paired with all possible execution results.

The action set $\mathcal{A}$ includes all possible program tokens [1], and the transition function $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is computed by the executor. Last, the reward $R(s, a) = 1$ iff the action a ends the program and leads to a state $\delta(s, a)$ where the denotation is equal to the target y. The model $\pi_\theta(\cdot)$ is a parameterized policy that provides a distribution over the program vocabulary at each time step.

SCONE as an MDP We define the partial execution result $[\![\tilde{z}]\!]_c^{ex}$ in SCONE, as described by Guu et al. (2017). We assume that SCONE's formal language is written in postfix notations, e.g., the instruction MOVE(HASHAT(YELLOW), LEFTOF(HASSHIRT(BLUE))) is written as YELLOW HASHAT BLUE HASSHIRT LEFTOF MOVE.

---

[1] Decoding is constrained to valid program tokens only.

With this notation, a partial program can be executed left-to-right by maintaining a *program stack*, ψ. The executor pushes constants (YELLOW) to ψ, and applies functions (HASHAT) by popping their arguments from ψ and pushing back the computed result. Actions (MOVE) are applied by popping arguments from ψ and performing the action in the current world.

To handle references to previously executed commands, SCONE's formal language includes functions that provide access to actions and arguments in previous commands. To this end, the executor maintains an *execution history*, $\mathbf{h}^i = (e^1, \ldots, e^i)$, a list of executed actions and their arguments. Thus, the execution result of a program prefix is $[\![\tilde{z}]\!]^{ex}_{w^0} = (w^{i-1}, \psi, \mathbf{h}^{i-1})$.

We adopt the model from Guu et al. (2017) (architecture details in appendix A): The policy π_θ observes ψ and u^i, the current utterance being parsed, and predicts a token. When the model predicts an action token that terminates a command, the model moves to the next utterance (until all utterances have been processed). The model uses functions to query the world w^{i-1} and history $\mathbf{h}^{i-1}$. Thus, each MDP state in SCONE is a pair $s = (u^i, [\![\tilde{z}]\!]^{ex}_{w^0})$. Figure 2 illustrates a state transition in the SCENE domain. Importantly, the state does not store the full program's prefix, and many different prefixes may lead to the same state. Next, we describe a search algorithm for this MDP.

4 Searching in Execution Space

Model improvement relies on generating correct programs given a possibly weak model. Standard beam-search explores the space of all program token sequences up to some fixed length. We propose two technical contributions to improve search: (a) We simplify the search problem by searching for correct executions rather than correct programs; (b) We use the target denotation at training time to better estimate partial program scores in the search space. We describe those next.

4.1 Reducing program search to execution search

Program space can be formalized as a directed tree $T = (\mathcal{V}_T, \mathcal{E}_T)$, where vertices $\mathcal{V}_T$ are program prefixes, and labeled edges $\mathcal{E}_T$ represent prefixes' continuations: an edge $e = (\tilde{z}, \tilde{z}')$ labeled with the token a, represents a continuation of the prefix $\tilde{z}$ with the token a, which yields the prefix $\tilde{z}'$. The root of the graph represents the empty sequence. Similarly, *Execution space* is a directed graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ induced from the MDP described in Section 3. Vertices $\mathcal{V}_G$ represent MDP states, which express execution results, and labeled edges $\mathcal{E}_G$ represent transitions. An edge (s_1, s_2) labeled by token a means that $\delta(s_1, a) = s_2$. Since multiple programs have the same execution result, execution space is a compressed representation of program space. Figure 3 shows a few commands in both program and execution space. Execution space is smaller, and so searching in it is easier.

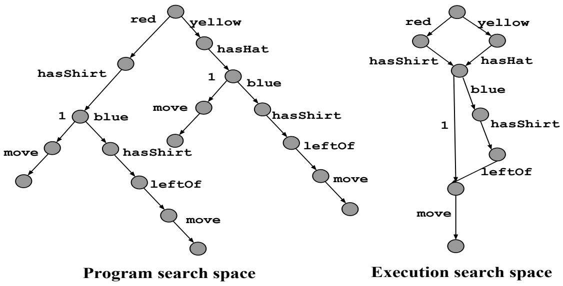


Figure 3: A set of commands represented in program space (left) and execution space (right). The commands relate to the first world in Figure 2. Since multiple prefixes have the same execution result (e.g., YELLOW HASHAT and RED HASSHIRT), the execution space is smaller.

Each path in execution space represents a different program prefix, and the path's final state represents its execution result. Program search can therefore be reduced to execution search: given an example $(c, \mathbf{u}, y)$ and a model π_θ, we can use π_θ to explore in execution space, discover *correct terminal states*, i.e. states corresponding to correct full programs, and extract paths leading to those states. As the number of paths may be exponential in the size of the graph, we can use beam-search to extract the most probable correct programs (according to the model) in the discovered graph.

Our approach is similar to the DPD algorithm (Pasupat and Liang, 2016), where CKY-style search is performed in denotation space, followed by search in a pruned space of programs. However, DPD was used without learning, and the search was not guided by a trained model, which is a major part of our algorithm.

4.2 Value-based Beam Search in Execution Space

We propose **V**alue-based **B**eam **S**earch **i**n e**X**ecution space (VBSIX), a variant of beam

Algorithm 1 Program Search with VBSIX

```
 1: function ProgramSearch($c, \mathbf{u}, y, \pi_\theta, V_\phi$)
 2:     $G, \mathcal{T} \leftarrow$ VBSIX($c, \mathbf{u}, y, \pi_\theta, V_\phi$)
 3:     $\mathcal{Z} \leftarrow$ Find paths in $G$ that lead to states in $\mathcal{T}$ with beam search
 4:     Return $\mathcal{Z}$
 5: function VBSIX($c, \mathbf{u}, y, \pi_\theta, V_\phi$)
 6:     $\mathcal{T} \leftarrow \emptyset, P \leftarrow \{\}$                 ▷ init terminal states and DP chart
 7:     $s_0$ := The empty program parsing state
 8:     $B_0 \leftarrow \{s_0\}, G = (\{s_0\}, \emptyset)$                ▷ Init beam and graph
 9:     $P_0[s_0] \leftarrow 1$                               ▷ The probability of $s_0$ is 1
10:     for $t \in [1 \dots L]$ do
11:         $B_t \leftarrow \emptyset$
12:         for $s \in B_{t-1}, a \in \mathcal{A}$ do
13:             $s' \leftarrow \delta(s, a)$
14:             Add edge $(s, s')$ to $G$ labeled with $a$
15:             if $s'$ is correct terminal then
16:                 $\mathcal{T} \leftarrow \mathcal{T} \cup \{s'\}$
17:             else
18:                 $P_t[s'] \leftarrow P_t[s'] + P_{t-1}[s] \cdot \pi_\theta(a|s)$
19:                 $B_t \leftarrow B_t \cup \{s'\}$
20:         Sort $B_t$ by AC-Scorer($\cdot$)
21:         $B_t \leftarrow$ Keep the top-$K$ states in $B_t$
22:     Return $G, \mathcal{T}$
23: function AC-Scorer($s, P_t, V_\phi, y$)
24:     Return $P_t[s] + V_\phi(s, y)$
```

search modified for searching in execution space, that scores search states with a value-based network. VBSIX is formally defined in Algorithm 1, which we will refer to throughout this section.

Standard beam search is a breadth-first traversal of the program space tree, where a fixed number of vertices are kept in the beam at every level of the tree. The selection of vertices is done by scoring their corresponding prefixes according to $p_\theta(z_{1 \dots t} \mid \mathbf{u}, c)$. VBSIX applies the same traversal in execution space (lines 10-21). However, since each vertex in the execution space represents an execution result and not a particular prefix, we need to modify the scoring function.

Let s be a vertex discovered in iteration t of the search. We propose two scores for ranking s. The first is *the actor score*, the probability of reaching vertex s after t iterations[2] according to the model π_θ. The second and more novel score is the value-based *critic score*, an estimate of the state's expected reward. The AC-Score is the sum of these two scores (lines 23-24).

The actor score, $p^t_\theta(s)$, is the sum of probabilities of all prefixes of length t that reach s (rather than the probability of one prefix as in beam search). VBSIX approximates $p^t_\theta(s)$ by performing the summation only over paths that reach s via states in the beam B_{t-1}, which can be done efficiently with a dynamic programming (DP) chart $P_t[s]$ that keeps actor score approximations in each iteration (line 18). This lower-bounds the true $p^t_\theta(s)$ since some prefixes of length t that reach s might have not been discovered.

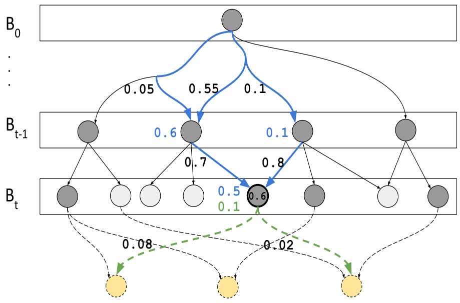


Figure 4: An illustration of scoring and pruning states in step t of VBSIX. Discovered edges are in full edges, undiscovered edges are dashed, correct terminal states are dashed and in yellow, and states that are kept in the beam are in dark grey. The actor scores of the highlighted vertex (0.5) and its parents (0.6, 0.1) are in blue, and its critic score (0.1) is in green.

Contrary to standard beam-search, we want to score search states also with a critic score $\mathbb{E}_{p_\theta}[R(s)]$, which is the sum of the suffix probabilities that lead from s to a correct terminal state:

$$\mathbb{E}_{p_\theta}[R(s)] = \sum_{\tau(s)} p_\theta(\tau(s) \mid s) \cdot R(\tau(s)),$$

where $\tau(s)$ are all possible trajectories starting from s and $R(\tau(s))$ is the reward observed when taking the trajectory $\tau(s)$ from s. Enumerating all trajectories $\tau(s)$ is intractable and so we will approximate $\mathbb{E}_{p_\theta}[R(s)]$ with a trained value network $V_\phi(s, y)$, parameterized by ϕ. Importantly, because we are searching at training time, we can condition V_ϕ on both the current state s and target denotation y. At test time we will use π_θ only, which does not need access to y.

Since the value function and DP chart are used for efficient ranking, the asymptotic run-time complexity of VBSIX is the same as standard beam search ($\mathcal{O}(K \cdot |\mathcal{A}| \cdot L)$). The beam search in Line 3, where we extract programs from the constructed execution space graph, can be done with a small beam size, since it operates over a small space of correct programs. Thus, its contribution to the algorithm complexity is negligible.

Figure 4 visualizes scoring and pruning with the actor-critic score in iteration t. Vertices in B_t are discovered by expanding vertices in B_{t-1}, and each vertex is ranked by the AC-scorer. The highlighted vertex score is 0.6, a sum of the actor score (0.5) and the critic score (0.1). The actor score is the sum of its prefixes $((0.05 + 0.55) \cdot 0.7 + 0.01 \cdot$

---

[2] We score paths in different iterations independently to avoid bias for shorter paths. An MDP state that appears in multiple iterations will get a different score in each iteration.

$0.08 = 0.5$) and the critic score is a value network estimate for the sum of probabilities of outgoing trajectories reaching correct terminal states ($0.02 + 0.08 = 0.1$). Only the top-K states are kept in the beam ($K = 4$ in the figure).

VBSIX leverages execution space in a number of ways. First, since each vertex in execution space compactly represents multiple prefixes, a beam in VBSIX effectively holds more prefixes than in standard beam search. Second, running beam-search over a graph rather than a tree is less greedy, as the same vertex can surface back even if it fell out of the beam.

The value-based approach has several advantages as well: First, evaluating the probability of outgoing trajectories provides look-ahead that is missing from standard beam search. Second (and most importantly), V_ϕ is conditioned on y, which π_θ doesn't have access to, which allows finding correct programs with low model probability, that π_θ can learn from. We note that our two contributions are orthogonal: the critic score can be used in program space, and search in execution space can be done with actor-scores only.

5 Training

We train the model π_θ and value network V_ϕ jointly (Algorithm 2). π_θ is trained using MML over discovered correct programs (Line 4, Algorithm 1). The value network is trained as follows: Given a training example $(c, \mathbf{u}, y)$, we generate a set of correct programs $\mathcal{Z}_{\text{pos}}$ with VBSIX. The value network needs negative examples, and so for every incorrect terminal state z_{neg} found during search with VBSIX we create a single program leading to z_{neg}. We then construct a set of training examples $\mathcal{D}_v$, where each example $((s, y), l)$ labels states encountered while generating programs $z \in \mathcal{Z}$ with the probability mass of correct programs suffixes that extend it, i.e., $l = \sum_{z_t} p_\theta(z_{t\ldots|z|})$, where z_t ranges over all $z \in \mathcal{Z}$ and $t \in [1 \ldots |z|]$ Finally, we train V_ϕ to minimize the log-loss objective: $\sum_{((s,y),l)\in\mathcal{D}_v} l\cdot \log V_\phi(s,y) + (1-l)(\log(1-V_\phi(s,y)))$.

Similar to actor score estimation, labeling examples for V_ϕ is affected by beam-search errors: the labels lower bound the true expected reward. However, since search is guided by the model, those programs are likely to have low probability. Moreover, estimates from V_ϕ are based on multiple examples, compared to probabilities in the DP chart, and are more robust to search errors.

Algorithm 2 Actor-Critic Training

```
procedure TRAIN()
    Initialize $\theta$ and $\phi$ randomly
    while $\pi_\theta$ not converged do
        $(x := (c, \mathbf{u}), y) \leftarrow$ select random example
        $\mathcal{Z}_{\text{pos}} \leftarrow \text{PROGRAMSEARCH}(c, \mathbf{u}, y, \pi_\theta, V_\phi)$
        $\mathcal{Z}_{\text{neg}} \leftarrow$ programs leading to incorrect terminal states
        $\mathcal{D}_v \leftarrow \text{BUILDVALUEEXAMPLES}((\mathcal{Z}_{\text{pos}} \cup \mathcal{Z}_{\text{neg}}), c, y)$
        Update $\phi$ using $\mathcal{D}_v$, update $\theta$ using $(x, \mathcal{Z}_{\text{pos}}, y)$
function BUILDVALUEEXAMPLES($\mathcal{Z}, c, \mathbf{u}, y$)
    for $z \in \mathcal{Z}$ do
        for $t \in [1 \ldots |z|]$ do
            $s \leftarrow [\![z_{1\ldots t}]\!]^{ex}_{c}$
            $L[s] \leftarrow L[s] + p_\theta(z_{t\ldots|z|} \mid c, \mathbf{u}) \cdot R(z)$
    $\mathcal{D}_v \leftarrow \{((s, y), L[s])\}_{s \in L}$
    Return $\mathcal{D}_v$
```

Neural network architecture: We adapt the model proposed by Guu et al. (2017) for SCONE. The model receives the current utterance u^i and program stack ψ, and returns a distribution over the next token. Our value network receives the same input, but also the next utterance u^{i+1}, the world state w^i and target world state y, and outputs a scalar. Appendix A provides a full description.

6 Experiments

6.1 Experimental setup

We evaluated our method on the three domains of SCONE with the standard accuracy metric, i.e., the proportion of test examples where the predicted program has the correct denotation. We trained with VBSIX, and used standard beam search ($K = 32$) at test time for programs' generation. Each test example contains 5 utterances, and similar to prior work we reported the model accuracy on all 5 utterances as well as the first 3 utterances. We ran each experiment 6 times with different random seeds and reported the average accuracy and standard deviation.

In contrast to prior work on SCONE (Long et al., 2016; Guu et al., 2017; Suhr and Artzi, 2018), where models were trained on all sequences of 1 or 2 utterances, and thus were exposed during training to all gold intermediate states, we trained from longer sequences keeping intermediate states latent. This leads to a harder search problem that was not addressed previously, but makes our results incomparable to previous results [3]. In SCENE and TANGRAM, we used the first 4 and 5 utterances as examples. In ALCHEMY, we used the first utterance and 5 utterances.

---

[3] For completeness, we show the performance on these datasets from prior work in Appendix C.

	Beam	SCENE 3 utt	SCENE 5 utt	ALCHEMY 3 utt	ALCHEMY 5 utt	TANGRAM 3 utt	TANGRAM 5 utt
MML	32	8.4±(2.0)	7.2±(1.3)	41.9±(22.8)	33.2±(20.1)	32.5±(20.7)	16.8±(14.1)
	64	15.4±(12.6)	12.3±(9.6)	44.6±(23.7)	36.3±(20.6)	45.6±(18.0)	25.8±(12.6)
EXPERT-MML	32	1.8±(1.5)	1.6±(1.2)	29.4±(22.7)	23.1±(18.8)	2.4±(0.8)	1.2±(0.5)
VBSIX	32	**34.2**±(27.5)	**28.2**±(20.7)	**74.5**±(1.1)	**64.8**±(1.5)	**65.0**±(0.8)	**43.0**±(1.3)

Table 1: Test accuracy and standard deviation of VBSIX compared to MML baselines (top) and our training methods (bottom). We evaluate the same model over the first 3 and 5 utterances in each domain.

Search space	Value	SCENE 3 utt	SCENE 5 utt	ALCHEMY 3 utt	ALCHEMY 5 utt	TANGRAM 3 utt	TANGRAM 5 utt
Program	No	5.5±(0.5)	3.8±(0.6)	36.4±(26.5)	25.4±(23.0)	34.4±(18.3)	15.6±(12.8)
Execution	No	7.4±(10.4)	4.0±(5.8)	41.3±(28.6)	28.2±(23.27)	33.5±(15.5)	12.7±(10.4)
Program	Yes	7.6±(8.3)	3.4±(2.9)	78.5±(1.0)	72.8±(1.3)	66.8±(1.5)	42.8±(1.9)
Execution	Yes	**31.0**±(24.7)	**22.6**±(19.6)	**81.9**±(1.3)	**75.2**±(2.9)	**68.6**±(2.0)	**44.2**±(2.1)

Table 2: Validation accuracy when ablating the different components of VBSIX. The first line presents MML, the last line is VBSIX, and the intermediate lines examine execution space and value-based networks separately.

Training details To warm-start the value network, we trained it for a few thousand steps, and only then start re-ranking with its predictions. Moreover, we gain efficiency by first returning K_0(=128) states with the actor score, and then re-ranking with the actor-critic score, returning K(=32) states. Last, we use the value network only in the last two utterances of every example since we found it has less effect in earlier utterances where future uncertainty is large. We used the Adam optimizer (Kingma and Ba, 2014) and fixed GloVe embeddings (Pennington et al., 2014) for utterance words.

Baselines We evaluated the following training methods (Hyper-parameters are in appendix B):

1. MML: Our main baseline, where search is done with beam search and training with MML. We used randomized beam-search, which adds ϵ-greedy exploration to beam search, which was proposed by Guu et al. (2017) and performed better [4].
2. EXPERT-MML: An alternative way of using the target denotation y at training time, based on imitation learning (Daume et al., 2009; Ross et al., 2011; Berant and Liang, 2015), is to train an *expert policy* $\pi_\theta^{\text{expert}}$, which receives y as input in addition to the parsing state, and trains with the MML objective. Then, our policy π_θ is trained using programs found by $\pi_\theta^{\text{expert}}$. The intuition is that the expert can use y to find good programs that the policy π_θ can train from.
3. VBSIX: Our proposed training algorithm.

We also evaluated REINFORCE, where Monte-Carlo sampling is used as search strategy (Williams, 1992; Sutton et al., 1999). We followed the implementation of Guu et al. (2017), who performed variance reduction with a constant baseline and added ϵ-greedy exploration. We found that REINFORCE fails to discover any correct programs to bootstrap from.

6.2 Results

Table 1 reports test accuracy of VBSIX compared to the baselines. First, VBSIX outperforms all baselines in all cases. MML is the strongest baseline, but even with an increased beam ($K = 64$), VBSIX ($K = 32$) surpasses it by a large margin (more than 20 points on average). On top of the improvement in accuracy, in ALCHEMY and TANGRAM the standard deviation of VBSIX is lower than the other baselines across the 6 random seeds, showing the robustness of our model.

EXPERT-MML performs worse than MML in all cases. We hypothesize that using the denotation y as input to the expert policy $\pi_\theta^{\text{expert}}$ results in many spurious programs, i.e., they are unrelated to the utterance meaning. This is since the expert can learn to perform actions that take it to the target world state while ignoring the utterances completely. Such programs will lead to bad generalization of π_θ. Using a critic at training time eliminates this problem, since its scores depend on π_θ.

Ablations We performed ablations to examine the benefit of our two technical contributions (a) execution space (b) value-based search. Table 2 presents accuracy on the validation set when

---

[4] We did not include meritocratic updates (Guu et al., 2017), since it performed worse in initial experiments.

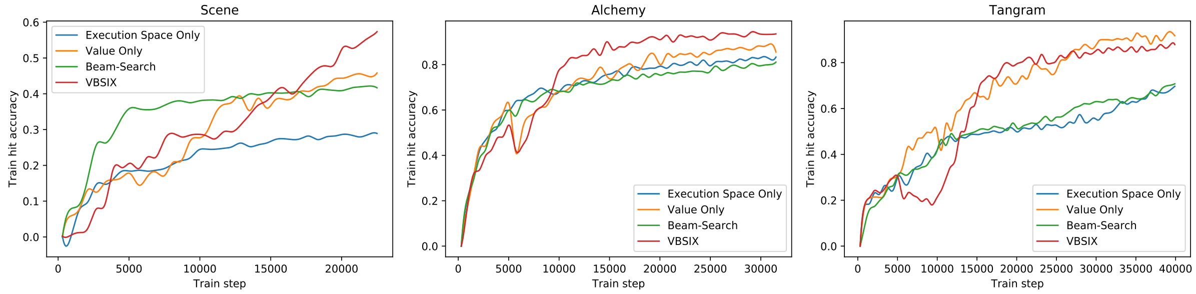


Figure 5: Training hit accuracy on examples with 5 utterances, comparing VBSIX to baselines with ablated components. The results are averaged over 6 runs with different random seeds (best viewed in color)

each component is used separately, when both of them are used (VBSIX), and when none are used (beam-search). We find that both contributions are important for performance, as the full system achieves the highest accuracy in all domains. In SCENE, each component has only a slight advantage over beam-search, and therefore both are required to achieve significant improvement. However, in ALCHEMY and TANGRAM most of the gain is due to the value network.

We also directly measured the *hit accuracy* at training time, i.e., the proportion of training examples where the beam produced by the search algorithm contains a program with the correct denotation, showing the effectiveness of search at training time. In Figure 5, we report train hit accuracy in each training step, averaged across 6 random seeds. The graphs illustrate the performance of each search algorithm in every domain throughout training. The validation accuracy results are correlated with the improvement in train hit-accuracy.

6.3 Analysis

Execution Space We empirically measured two quantities that we expect should reflect the advantage of execution-space search. First, we measured the number of programs stored in the execution space graph compared to beam search, which holds K programs. Second, we counted the average number of states that are connected to correct terminal states in the discovered graph, but fell out of the beam during the search. The property reflects the gain from running search over a graph structure, where the same vertex can resurface. We preformed the analysis on VBSIX over 5-utterance training examples in all 3 domains. The following table summarizes the results:

Property	**SCENE**	**ALCHEMY**	**TANGRAM**
Paths in beam	143903	5892	678
Correct pruned	18.5	11.2	3.8

We found the measured properties and the contribution of execution space in each domain are correlated, as seen in the ablations. Differences between domains are due to the different complexities of their formal languages. As the formal language becomes more expressive, the execution space is more compressed as each state can be reached in more ways. In particular, the formal language in SCENE contains more functions compared to the other domains, and so it benefits the most from execution-space search.

Value Network We analyzed the accuracy of the value network at training time by measuring, for each state, the difference between its expected reward (estimated from the discovered paths) and the value network prediction. Figure 6 shows the average difference in each training step for all encountered states (in blue), and for high reward states only (states with expected reward larger than 0.7, in orange). Those metrics are averaged across 6 runs with different random seeds.

The accuracy of the value network improves during training, except when the policy changes substantially, in which case the value network needs to re-evaluate the policy. When the value network converges, the difference between its predictions and the expected reward is $0.15 - 0.2$ on average. However, for high reward states the difference is higher (~ 0.3). This indicates that the value network has a bias toward lower values, which is expected as most states lead to low rewards. Since VBSIX uses the value network as a beam-search ranker, the value network doesn't need to be exact as long as it assigns higher values to states with higher expected rewards. Further analysis is provided in appendix D.

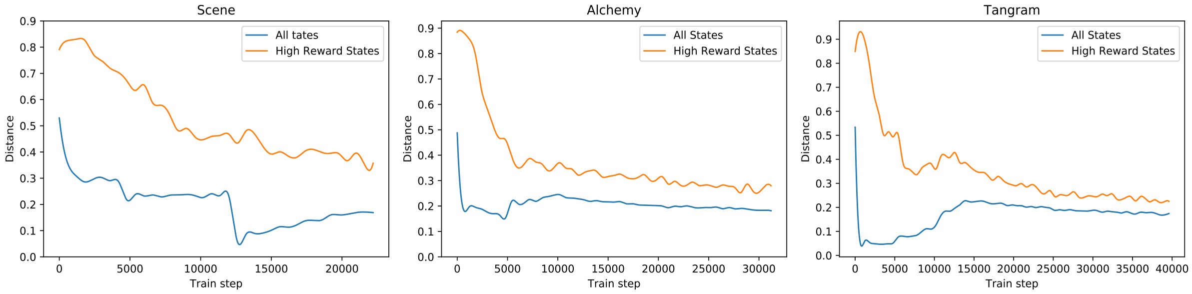


Figure 6: The difference between the prediction of the value network and the expected reward (estimated from the discovered paths) during training. We report the average difference for all of the states (blue) and for the high reward states only (> 0.7, orange). The results are averaged over 6 runs with different random seeds (best viewed in color).

7 Related Work

Training from denotations has been extensively investigated (Kwiatkowski et al., 2013; Pasupat and Liang, 2015; Bisk et al., 2016), with a recent emphasis on neural models (Neelakantan et al., 2016; Krishnamurthy et al., 2017). Improving beam search has been investigated by proposing specialized objectives (Wiseman and Rush, 2016), stopping criteria (Yang et al., 2018), and using continuous relaxations (Goyal et al., 2018).

Bahdanau et al. (2017) and Suhr and Artzi (2018) proposed ways to evaluate intermediate predictions from a sparse reward signal. Bahdanau et al. (2017) used a critic network to estimate expected BLEU in translation, while Suhr and Artzi (2018) used edit-distance between the current world and the goal for SCONE. But, in those works stronger supervision was assumed: Bahdanau et al. (2017) utilized the gold sequences, and Suhr and Artzi (2018) used intermediate worlds states. Moreover, intermediate evaluations were used to compute gradient updates, rather than for guiding search.

Guiding search with both policy and value networks was done in Monte-Carlo Tree Search (MCTS) for tasks with a sparse reward (Silver et al., 2017; T. A. and and Barber, 2017; Shen et al., 2018). In MCTS, value network evaluations are refined with backup updates to improve policy scores. In this work, we gain this advantage by using the target denotation. The use of an actor and a critic is also reminiscent of A^* where states are scored by past cost and an admissible heuristic for future cost (Klein and Manning, 2003; Pauls and Klein, 2009; lee et al., 2016). In semantic parsing, Misra et al. (2018) recently proposed a critic distribution to improve the policy, which is based on prior domain knowledge (that is not learned).

8 Conclusions

In this work, we propose a new training algorithm for mapping instructions to programs given denotation supervision only. Our algorithm exploits the denotation at training time to train a critic network used to rank search states on the beam, and performs search in a compact execution space rather than in the space of programs. We evaluated on three different domains from SCONE, and found that it dramatically improves performance compared to strong baselines across all domains.

VBSIX is applicable to any task that supports graph-search exploration. Specifically, for tasks that can be formulated as an MDP with a deterministic transition function, which allow efficient execution of multiple partial trajectories. Those tasks include a wide range of instruction mapping (Branavan et al., 2009; Vogel and Jurafsky, 2010; Anderson et al., 2018) and semantic parsing tasks (Dahl et al., 1994; Iyyer et al., 2017; Yu et al., 2018). Therefore, evaluating VBSIX on other domains is a natural next step for our research.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work was completed in fulfillment for the M.Sc degree of the first author. This research was partially supported by The Israel Science Foundation grant 942/16, the Blavatnik Computer Science Research Fund, and The Yandex Initiative for Machine Learning.

References

P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Computer Vision and Pattern Recognition (CVPR)*.

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Berant and P. Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)*, 3:545–558.

Y. Bisk, D. Yuret, and D. Marcu. 2016. Natural language communication with robots. In *North American Association for Computational Linguistics (NAACL)*.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 82–90.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 859–865.

J. Cheng, S. Reddy, V. Saraswat, and M. Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Association for Computational Linguistics (ACL)*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

D. A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Workshop on Human Language Technology*, pages 43–48.

H. Daume, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.

D. Fried, J. Andreas, and D. Klein. 2018. Unified pragmatic models for generating and following instructions. In *North American Association for Computational Linguistics (NAACL)*.

O. Goldman, V. Latcinnik, U. Naveh, A. Globerson, and J. Berant. 2018. Weakly-supervised semantic parsing with abstract examples. In *Association for Computational Linguistics (ACL)*.

K. Goyal, G. Neubig, C. Dyer, and T. Berg-Kirkpatrick. 2018. A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

D. Klein and C. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

J. Krishnamurthy, P. Dasigi, and M. Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 754–765.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

K. lee, M. Lewis, and L. Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Liang, J. Berant, Q. Le, and K. D. F. N. Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Association for Computational Linguistics (ACL)*.

C. Liang, M. Norouzi, J. Berant, Q. Le, and N. Lao. 2018. Memory augmented policy optimization for program synthesis with generalization. In *Advances in Neural Information Processing Systems (NIPS)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

R. Long, P. Pasupat, and P. Liang. 2016. Simpler context-dependent logical forms via model projections. In *Association for Computational Linguistics (ACL)*.

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. *arXiv preprint arXiv:1809.01299*.

A. Neelakantan, Q. V. Le, and I. Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations (ICLR)*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

P. Pasupat and P. Liang. 2016. Inferring logical forms from denotations. In *Association for Computational Linguistics (ACL)*.

A. Pauls and D. Klein. 2009. K-best A* parsing. In *Association for Computational Linguistics (ACL)*, pages 958–966.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

M. Rabinovich, M. Stern, and D. Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Association for Computational Linguistics (ACL)*.

S. Ross, G. Gordon, and A. Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*.

Y. Shen, J. Chen, P. Huang, Y. Guo, and J. Gao. 2018. Reinforcewalk: Learning to walk in graph with monte carlo tree search. In *International Conference on Learning Representations (ICLR)*.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L., M. Lai, A. Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

A. Suhr and Y. Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Association for Computational Linguistics (ACL)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*.

Z. Tian T. A. and and D. Barber. 2017. *Thinking Fast and Slow with Deep Learning and Tree Search*. Advances in Neural Information Processing Systems 30.

A. Vogel and D. Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*, pages 806–814.

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

S. Wiseman and A. M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Yang, L. Huang, and M. Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Yin and G. Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Association for Computational Linguistics (ACL)*, pages 440–450.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

A Neural Network Architecture

We adopt the model $\pi_\theta(\cdot)$ proposed by Guu et al. (2017). The model receives the current utterance u^i and the program stack ψ. A bidirectional LSTM (Hochreiter and Schmidhuber, 1997) is used to embed u_i, while ψ is embedded by concatenating the embedding of stack elements. The embedded input is then fed to a feed-forward network with attention over the LSTM hidden states, followed by a softmax layer that predicts a program token. Our value network $V_\phi(\cdot)$ shares the input layer of $\pi_\theta(\cdot)$. In addition, it receives the next utterance u^{i+1}, the current world state w^i and the target world state w^M. The utterance u^{i+1} is embedded with an additional BiLSTM, and world

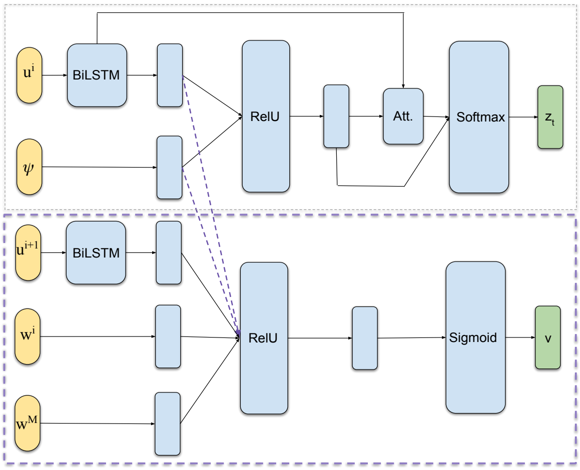


Figure 7: The model proposed by Guu et al. (2017) (top), and our value network (bottom).

states are embedded by concatenating embeddings of SCONE elements. The inputs are concatenated and fed to a feed-forward network, followed by a sigmoid layer that outputs a scalar.

B Hyper-parameters

Table 3 contains the hyper-parameter setting for each experiment. Hyper-parameters of REINFORCE and MML were taken from Guu et al. (2017). In all experiments learning rate was 0.001 and mini-batch size was 8. We explicitly define the following hyper-parameters which are not self-explanatory:

1. *Training steps*: The number of training steps taken.
2. *Sample size*: Number of samples drawn from p_θ in REINFORCE
3. *Baseline*: A constant subtracted from the reward for variance reduction.
4. *Execution beam size*: K in Algorithm 1.
5. *Program bea size*: Size of beam in line 3 of Algorithm 1.
6. *Value ranking start step*: Step when we start ranking states using the critic score.
7. *Value re-rank size*: Size of beam K_0 returned by the actor score before re-ranking with the actor-critic score.

C Prior Work

In prior work on SCONE, models were trained on sequences of 1 or 2 utterances, and thus were exposed during training to all gold intermediate states (Long et al., 2016; Guu et al., 2017; Suhr and Artzi, 2018). Fried et al. (2018) assumed access to the full annotated logical form. In contrast, we trained from longer sequences, keeping the logical form and intermediate states latent. We report the test accuracy as reported by prior work and in this paper, noting that our results are an average of 6 runs, while prior work reports the median of 5 runs.

Naturally, our results are lower compared to prior work that uses much stronger supervision. This is because our setup poses a hard search problem at training time, and also requires overcoming *spuriousness* – the fact that even incorrect programs sometimes lead to high reward.

D Value Network Analysis

We analyzed the ability of the value network to predict expected reward. The reward of a state depends on two properties, (a) *connectivity*: whether there is a trajectory from this state to a correct terminal state, and (b) *model likelihood*: the probability the model assigns to those trajectories. We collected a random set of 120 states in the SCENE domain from, where the real expected reward was very high (> 0.7), or very low ($= 0.0$) and the value network predicted well (less than 0.2 deviation) or poorly (more than 0.5 deviation). For ease of analysis we only look at states from the final utterance.

To analyze connectivity, we looked at states that cannot reach a correct terminal state with a single action (since states in the last utterance can perform one action only, the expected reward is 0). Those are states where either their current and target world differ in too many ways, or the stack content is not relevant to the differences between the worlds. We find that when there are many differences between the current and target world, the value network correctly estimates low expected reward in 87.0% of the cases. However, when there is just one mismatch between the current and target world, the value network tends to ignore it and erroneously predicts high reward in 78.9% of the cases.

To analyze whether the value network can predict the success of the trained policy, we consider states from which there is an action that leads to the target world. While it is challenging to fully interpret the value network, we notice that the network predicts a value that is > 0.5 in 86.1% of the cases where the number of people in the world is no more than 2, and a value that is < 0.5 in

System	SCENE	ALCHEMY	TANGRAM
REINFORCE	Training steps $= 22.5k$ Sample size $= 32$ $\epsilon = 0.2$ Baseline $= 10^{-5}$	Training steps $= 31.5k$ Sample size $= 32$ $\epsilon = 0.2$ Baseline $= 10^{-2}$	Training steps $= 40k$ Sample size $= 32$ $\epsilon = 0.2$ Baseline $= 10^{-3}$
MML	Training steps $= 22.5k$ Beam size $= 32$ $\epsilon = 0.15$	Training steps $= 31.5k$ Beam size $= 32$ $\epsilon = 0.15$	Training steps $= 40k$ Beam size $= 32$ $\epsilon = 0.15$
VBSiX	Training steps $= 22.5k$ Execution beam size $= 32$ Program beam size $= 8$ $\epsilon = 0.15$ Value ranking start step $= 5k$ Value re-rank size $= 128$	Training steps $= 31.5k$ Execution beam size $= 32$ Program beam $= 8$ $\epsilon = 0.15$ Value ranking start step $= 5k$ Value re-rank size $= 128$	Training steps $= 40k$ Execution beam size $= 32$ Program beam size $= 8$ $\epsilon = 0.15$ Value ranking start step $= 10k$ Value re-rank size $= 128$

Table 3: Hyper-parameter settings.

	SCENE		ALCHEMY		TANGRAM	
	3 utt	**5 utt**	**3 utt**	**5 utt**	**3 utt**	**5 utt**
Long et al. (2016)	23.2	14.7	56.8	52.3	64.9	27.6
Guu et al. (2017)	64.8	46.2	66.9	52.9	65.8	37.1
Suhr and Artzi (2018)	73.9	62.0	74.2	62.7	80.8	62.4
Fried et al. (2018)	–	72.7	–	72.0	–	69.6
VBSIX	34.2	28.2	74.5	64.8	65.0	43.0

Table 4: Test accuracy comparison to prior work.

82.1% of the cases where the number of people in the world is more than 2. This indicates that the value network believes more complex worlds, involving many people, are harder for the policy.

VQD: Visual Query Detection in Natural Scenes

Manoj Acharya[1] **Karan Jariwala**[1] **Christopher Kanan**[1,2,3]
[1]Rochester Institute of Technology [2]PAIGE [3]Cornell Tech
{ma7583, kkj1811, kanan}@rit.edu

Abstract

We propose Visual Query Detection (VQD), a new visual grounding task. In VQD, a system is guided by natural language to localize a *variable* number of objects in an image. VQD is related to visual referring expression recognition, where the task is to localize only *one* object. We describe the first dataset for VQD and we propose baseline algorithms that demonstrate the difficulty of the task compared to referring expression recognition.

1 Introduction

In computer vision, object detection is the task of identifying all objects from a specific closed-set of pre-defined classes by putting a bounding box around each object present in an image, e.g., in the widely used COCO dataset there are 80 object categories and an algorithm must put a box around all instances of each object present in an image (Lin et al., 2014). Recent deep learning based models have significantly advanced the state-of-the-art in object detection (Ren et al., 2015b); however, many applications demand more nuanced detection of objects with specific attributes or objects in relation to each other. Here, we study goal-directed object detection, where the set of possible valid objects is far greater than in the typical object detection problem. Specifically, we introduce the Visual Query Detection (VQD) task (see Fig. 1). In VQD, a system is given a query in natural language and an image and it must produce 0–N boxes that satisfy that query. VQD has numerous applications, ranging from image retrieval to robotics.

VQD is related to the visual referring expression recognition (RER) task (Kazemzadeh et al., 2014); however, in RER every image has only a *single* correct box. In contrast, in VQD there could be no valid outputs for a query or multiple valid outputs, making the task both harder and more useful. As discussed later, existing RER datasets have multiple annotation problems and have significant language bias problems. VQD is also related to Visual Question Answering (VQA), where the task is to answer questions about images in natural language (Malinowski and Fritz, 2014; Antol et al., 2015). The key difference is that in VQD the algorithm must generate image bounding boxes that satisfy the query, making it less prone to the forms of bias that plague VQA datasets.

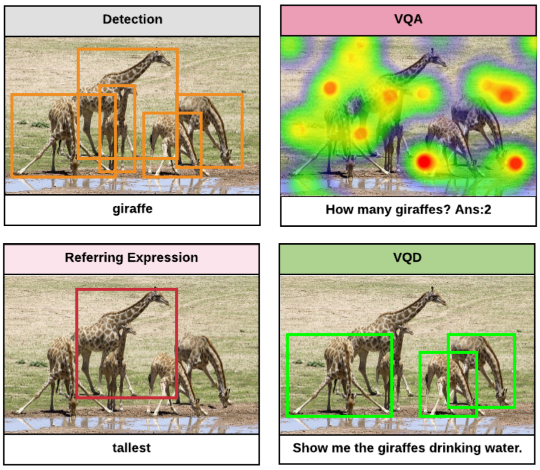


Figure 1: Unlike VQD, object detection cannot deal with attributes and relations. In VQA, often algorithms produce the right answers due to dataset bias without ‘looking’ at relevant image regions. RER datasets have short and often ambiguous prompts, and by having only a single box as an output, they make it easier to exploit dataset biases. VQD requires goal-directed object detection and outputting a variable number of boxes that answer a query.

We make the following contributions:

1. We describe the first dataset for VQD, which will be publicly released.
2. We evaluate multiple baselines on our VQD dataset.

Proceedings of NAACL-HLT 2019, pages 1955–1961
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related work

Over the past few years, a large amount of work has been done at the intersection of computer vision and natural language understanding, including visual madlibs (Yu et al., 2015; Tommasi et al., 2018), captioning (Farhadi et al., 2010; Kulkarni et al., 2013; Johnson et al., 2016; Liu et al., 2018), and image retrieval (Wan et al., 2014; Li et al., 2016). For VQD, the most related tasks are VQA and RER, which we review in detail.

2.1 Visual Question Answering

VQA systems take in an image and open-ended natural language question and then generate a text-based answer (Antol et al., 2015; Goyal et al., 2017; Acharya et al., 2019; Kafle et al., 2018). Many VQA datasets have been created. However, initial datasets, e.g., VQAv1 (Antol et al., 2015) and COCO-QA (Ren et al., 2015a), exhibited significant language bias in which many questions could be answered correctly without looking at the image, e.g., for VQAv1 it was possible to achieve 50% accuracy using language alone (Kafle and Kanan, 2016). To address the bias issue, the VQAv2 dataset was created with a more balanced distribution for each possible answer to make algorithms analyze the image (Goyal et al., 2017), but it still had bias in the kinds of questions asked, with some questions being scarce, e.g., reasoning questions. Synthetic datasets such as the CLEVR dataset (Johnson et al., 2017) addressed this by being synthetically generated to emphasize hard reasoning questions that are rare in VQAv1 and VQAv2. The TDIUC dataset addresses bias using both synthetically generated and human gathered questions about natural images, with performance evaluated for 12 kinds of questions (Kafle and Kanan, 2017a). While the state-of-the-art has rapidly increased on both synthetic and natural image VQA datasets, many models do not generalize across datasets (Shrestha et al., 2019).

2.2 Referring Expression Recognition

Unlike VQA, RER algorithms must produce evidence to justify their outputs. A RER algorithm outputs a box around the image location matching the input string, making it easier to tell if an algorithm is behaving correctly. The RefCOCO and RefCOCO+ datasets for RER were collected from the two-player 'ReferIt' Game (Kazemzadeh et al., 2014). The first player is asked to describe an outlined object and the second player has to correctly localize it from player one's description. The test datasets are futher split into the 'testA' and 'testB' splits. The split 'testA' contains object categories sampled randomly to be close to the original data distribution, while 'testB' contains objects sampled from the most frequent object categories, excluding categories such as 'sky', 'sand', 'floor', etc. Since, there is a time limit on the game, the descriptions are short, e.g., 'guy in a yellow t-shirt,' 'pink,' etc.

Instead of playing a timed game, to create the RefCOCOg dataset for RER, one set of Amazon Mechanical Turk (AMT) users were asked to generate a description for a marked object in an image and other users marked the region corresponding to the description (Mao et al., 2016). This resulted in more descriptive prompts compared to RefCOCO and RefCOCO+.

The Visual7W dataset for VQA includes a 'pointing' task that is closely related to RER (Zhu et al., 2016). Pointing questions require choosing which box of the four *given* boxes correctly answered a query. Systems did not generate their own boxes, and there is always one correct box.

Cirik et al. (2018) showed that RER datasets suffer from biases caused by their dataset collection procedure. For RefCOCOg, they found that randomly permuting the word in the referring expression caused only about a 5% drop in performance, suggesting that instead of relying on language structure, systems may be using some hidden correlations in language. They further showed that an image only model that ignores the referring expression yielded a precision of 71.2% for top-2 best predictions. They also found that predicting the object category given the image region produced an accuracy of 84.2% for top-2 best predictions. By having 0–N boxes, VQD is harder for an image-only model to perform well.

3 The VQD 1.0 Dataset (VQDv1)

We created VQDv1, the first dataset for VQD. VQDv1 is created synthetically using annotations from Visual Genome (VG), COCO, and COCO Panoptic. While this limits variety, it helps combat some kinds of bias and serves as an initial version of the dataset. VQDv1 has three distinct query categories:

1. Object Presence (e.g., 'Show the dog in the image')
2. Color Reasoning (e.g., 'Which plate is white

Type	# Questions
Simple	391,628
Color	172,005
Positional	57,904
Total	621,537

Table 1: VQDv1 Query Types

Dataset	# Images	# Questions
RefCOCO	19,994	142,209
RefCOCO+	19,992	141,564
RefCOCOg	26,711	85,474
VQDv1	123,287	621,537

Table 2: VQDv1 compared to RER datasets.

in color?')

3. Positional Reasoning (e.g., 'Show the cylinder behind the girl in the picture')

The number of queries per type are given in the Table 1. The dataset statistics and example images and are shown in Fig. 2 and Fig. 3, respectively. We show statistics for VQDv1 compared to RER datasets in Table 2.

All images in VQDv1 are from COCO. The ground truth bounding box annotations are derived from the COCO Panoptic annotations dataset (Kirillov et al., 2018). The questions are generated using multiple templates for each question type, which is an approach that has been used in earlier work for VQA (Kafle and Kanan, 2017a; Kafle et al., 2017). The query objects and their attributes are extracted by integrating the annotations from images that have both COCO and VG annotations. COCO annotations are focused on objects, while VG also has attribute and relationship information, e.g., size, color, and actions for scene objects.

3.1 Object Presence

Object presence questions require an algorithm to determine all instances of an object in an image without any relationship or positional attributes, for example, 'Show me the horse in the image' or 'Where is the chair?' We use all of the COCO 'things' labels and half of the COCO 'stuff' labels to generate these questions, making this task test the same capabilities as conventional object detection. We filter some 'stuff' categories that do not have well defined bounding boxes such as 'water-other', 'floor-stone', etc. We use multiple templates to create variety, e.g., 'Show the <object> in the image', 'Where are the <object> in the picture?' etc.

3.2 Color Reasoning

Color questions test the presence of objects modified by color attributes, e.g., 'Show me the cat which is grey in color' or 'Which blanket is blue in color?' Since, COCO has only object annotations, color attributes are derived from VG's attribute annotations. We align every VG image annotation with COCO annotations to obtain (object, color) annotations for each bounding box. When multiple color attributes for an object are present, the object is assigned a single color from that attribute set.

3.3 Positional Reasoning

Positional reasoning questions test the location of objects with respect to other objects, e.g., 'Show the building behind horses', 'Which people are in front of the lighthouse?', and 'Show the rhino behind elephant.' We again use VG's relationship and attribute annotations to create these questions.

3.4 Generating Counter-Concept Questions

Counter-concept questions have no valid boxes as outputs, and we endeavor to create hard counter-concept questions for each category. We ask 'Show me the zebra' only if there is a similar animal present (e.g., a cow), which was done by using COCO's super-categories. Likewise, 'Show me the donut that is brown in color' is only asked if a brown donut does not exist in the image.

4 Experiments

Our experiments are designed to probe the behavior of models on VQD compared to RER datasets. To facilitate this, we created a variant of our VQDv1 dataset that had only a single correct bounding box.

To evaluate performance for the RER and '1 Obj' version of the VQDv1 dataset, systems only output a single bounding box during test time, so the **Precision@1** metric is used. For the '0-N Obj' version of the VQDv1 dataset, we use the standard PASCAL VOC metric $\mathbf{AP}^{IoU=.50}$ from object detection, which calculates the average precision across the dataset using an intersection over union (IoU) greater than 0.5 criteria for matching with the ground truth boxes.

4.1 Models Evaluated

We implemented and evaluated four models for VQD. All models are built on top of Faster R-CNN with a ResNet-101 backbone whose output bounding boxes pass through Non-Maximal Suppression

	RefCOCO			RefCOCO+			RefCOCOg		VQDv1	
	val	testA	testB	val	testA	testB	val	test	1 Obj.	0–N Obj.
DETECT	38.63	37.82	38.32	38.85	37.85	38.98	50.13	50.03	30.44	26.94
RANDOM	16.51	14.30	19.81	16.67	14.10	20.45	19.87	19.76	9.77	2.38
Query-Blind	33.95	37.28	31.58	34.06	37.34	32.46	39.79	23.34	23.34	6.80
Vision+Query	69.41	**75.52**	**65.28**	**59.83**	**65.21**	**53.02**	**62.52**	**62.06**	**37.55**	**31.03**
SLR	**69.48**	73.71	64.96	55.71	60.74	48.80	60.21	59.63	–	–
SLR	68.95	73.10	64.85	54.89	60.04	49.56	59.33	59.21	–	–

Table 3: Results on RER datasets and two versions of our VQD dataset. The ‘1 Obj’ version is trained and evaluated on queries with only a single box, analogous to RER, and the 0–N version contains the entire VQD dataset. All models use the same object proposals and visual features.

with a threshold of 0.7. This acts as a region proposal generator that provides CNN features for each region.

The four models we evaluate are:

1. **DETECT:** A model that uses the full Faster R-CNN system to detect all trained COCO classes, and then outputs the boxes that have the same label as the first noun in the query.
2. **RANDOM:** Select one random Faster R-CNN proposal.
3. **Query-Blind:** A vision only model that does binary classification of each region proposal’s CNN features using a 3 layer MultiLayer Perceptron(MLP) with 1024-unit hidden ReLU layers.
4. **Vision+Query (V+Q):** A model that does binary classification of each region proposal. The query features are obtained from the last hidden layer of a Gated Recurrent Unit (GRU) network, and then they are concatenated with the CNN features and fed into a 3 layer MLP with 1024-unit hidden ReLU layers.

The primary reason for providing VQDv1 (1 obj.) and the RER results is to put the benefits of the VQD task in context. To aid in this endeavor, we also include comparison results directly from the SLR models (Yu et al., 2017) for RER, which is a recent system for that task.

4.2 Training Details

The Query-Blind and Vision+Query models are trained with binary cross-entropy loss. We use a learning rate of 0.0004, and perform learning rate decay of 0.8 when the training loss plateaus continuously for five epochs. The best model is selected based on the validation loss after training for 50 epochs.

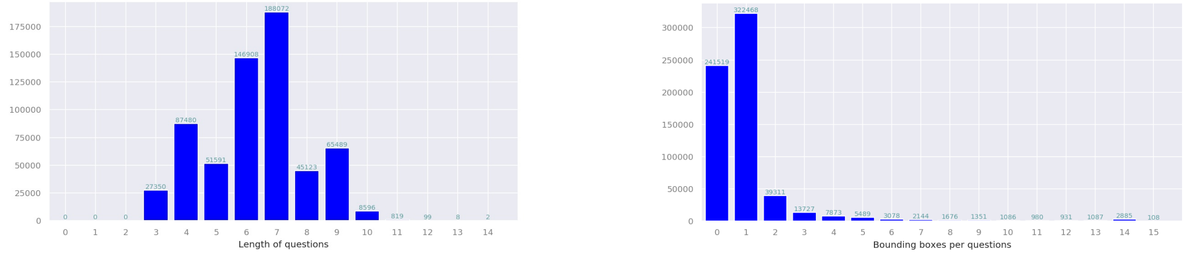
4.3 Results

Our main results are given in Table 3. Although simple, our Vision+Query model performs well across RER datasets, but it can also be applied to VQD tasks. As expected, RANDOM performs poorly on both VQDv1 datasets. DETECT beats RANDOM in the single object VQD setting by a large margin. Since, most of the questions in the RER datasets ask about common COCO categories, choosing one of those objects might be enough to get decent performance; however, DETECT performs poorly when evaluated under 0-N object settings in VQDv1. To handle queries in VQD, models must be able to understand the context and comprehend multiple objects in isolation.

5 Conclusion

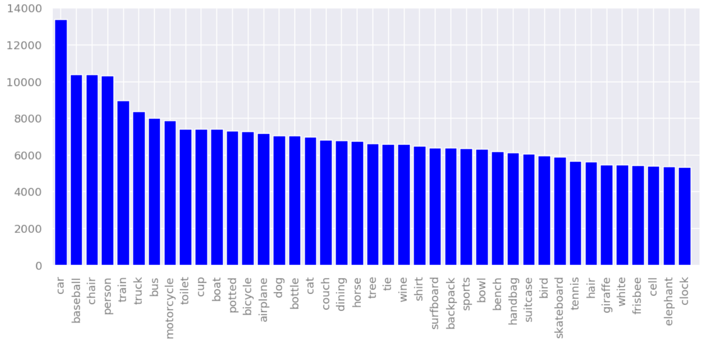
In this paper, we described our VQDv1 dataset as a test for visual grounding via goal-directed object detection. VQDv1 has both simple object presence and complex questions with 0–N bounding boxes. While VQDv1 contains only synthetically generated questions, this can help mitigate some forms of bias present in other VQA and RER datasets (Cirik et al., 2018; Kafle and Kanan, 2017b). While it would be expensive, a large, carefully filtered, and well designed human annotated VQD dataset is the next step toward advancing visual grounding research.

Compared to VQA, we argue that it is harder to be right for the *wrong* reasons in VQD because methods must generate bounding boxes. Compared to RER, we argue that it is harder to exploit bias in VQD since there are a variable number of boxes per image, making it considerably more difficult, as demonstrated by our experiments. We believe the VQD approach has considerable value and can be used to advance visual grounding research.



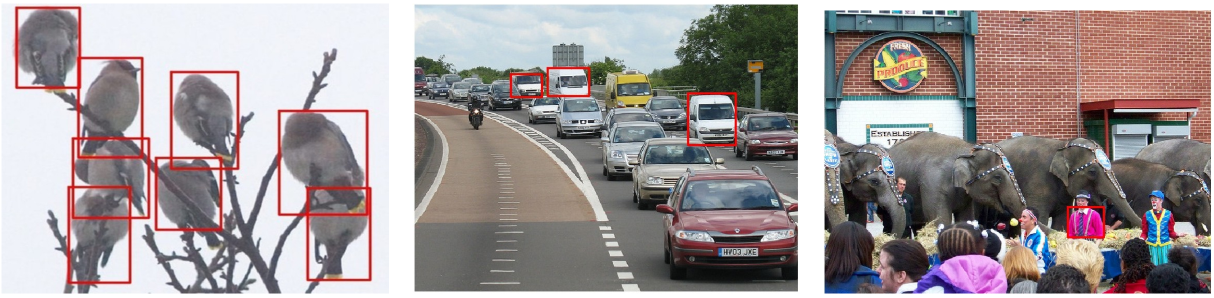
(a) Question length distribution.

(b) Bounding boxes per question distribution.



(c) Top-40 Object category distribution.

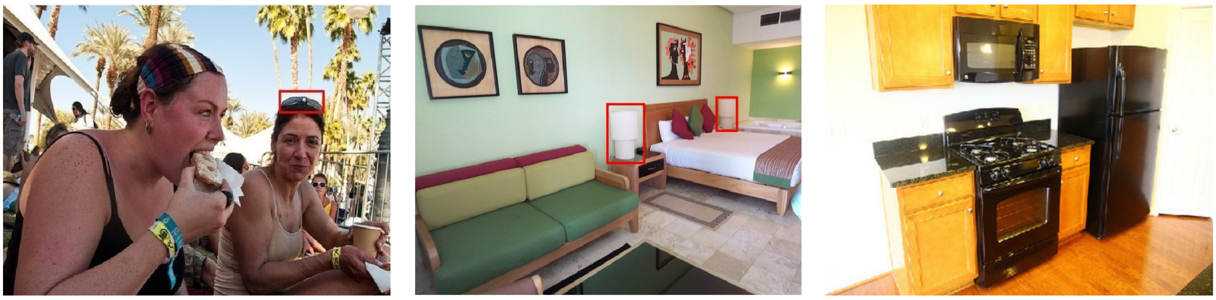
Figure 2: Distribution statistics for the VQD dataset.



(a) Where is the bird?

(b) Show me the van which is white in color.

(c) Which shirt is pink in color?



(d) Which glass is on the top of the head of the women?

(e) Show the lamp beside bed in the image.

(f) Where is the sink in the picture? Where is the toaster in the image?

Figure 3: Example query-detection pairs from the VQD dataset. Counter context questions that do not have a bounding box as an answer are generated in such a way that they are still relevant to the scene context. For example, in Fig. [3f] both questions pertain to the context ‘kitchen’.

Acknowledgments

This work was supported in part by a gift from Adobe Research. The lab thanks NVIDIA for the donation of a GPU. We also thank fellow lab members Kushal Kafle and Tyler Hayes for their comments and useful discussions.

References

Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2019. TallyQA: Answering complex counting questions. In *AAAI*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Volkan Cirik, Louis-Philippe Morency, and Taylor Berg-Kirkpatrick. 2018. Visual referring expression recognition: What do systems actually learn? *Association for Computational Linguistics (ACL)*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision (ECCV)*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kushal Kafle and Christopher Kanan. 2016. Answer-type prediction for visual question answering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kushal Kafle and Christopher Kanan. 2017a. An analysis of visual question answering algorithms. In *Proc. IEEE International Conference on Computer Vision (ICCV)*.

Kushal Kafle and Christopher Kanan. 2017b. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding (CVIU)*.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *INLG-2017*, pages 198–202.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Empirical methods in natural language processing (EMNLP)*.

Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. 2018. Panoptic segmentation. *CoRR*.

Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Xirong Li, Tiberio Uricchio, Lamberto Ballan, Marco Bertini, Cees GM Snoek, and Alberto Del Bimbo. 2016. Socializing the semantic gap: A comparative survey on image tag assignment, refinement, and retrieval. *ACM Computing Surveys*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*.

Xiaoxiao Liu, Qingyang Xu, and Ning Wang. 2018. A survey on deep neural network-based image captioning. *The Visual Computer*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Neural Information Processing Systems (NeurIPS)*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. *Computer Vision and Pattern Recognition (CVPR)*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015a. Exploring models and data for image question answering. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015b. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Robik Shrestha, Kushal Kafle, and Christopher Kanan. 2019. Answer them all! toward universal visual question answering models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tatiana Tommasi, Arun Mallya, Bryan Plummer, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. 2018. Combining multiple cues for visual madlibs question answering. *International Journal of Computer Vision (IJCV)*.

Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. 2014. Deep learning for content-based image retrieval: A comprehensive study. In *ACM international conference on Multimedia*.

Licheng Yu, Eunbyung Park, Alexander C Berg, and Tamara L Berg. 2015. Visual madlibs: Fill in the blank description generation and question answering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017. A joint speakerlistener-reinforcer model for referring expressions. In *Computer Vision and Pattern Recognition (CVPR)*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Improving Natural Language Interaction with Robots Using Advice

Nikhil Mehta
Department of Computer Science
Purdue University
West Lafayette, IN 47907
mehta52@purdue.edu

Dan Goldwasser
Department of Computer Science
Purdue University
West Lafayette, IN 47907
dgoldwas@purdue.edu

Abstract

Over the last few years, there has been growing interest in learning models for physically grounded language understanding tasks, such as the popular blocks world domain. These works typically view this problem as a single-step process, in which a human operator gives an instruction and an automated agent is evaluated on its ability to execute it. In this paper we take the first step towards increasing the bandwidth of this interaction, and suggest a protocol for including advice, high-level observations about the task, which can help constrain the agents prediction. We evaluate our approach on the blocks world task, and show that even simple advice can help lead to significant performance improvements. To help reduce the effort involved in supplying the advice, we also explore model self-generated advice which can still improve results.

1 Introduction

The problem of constructing an artificial agent capable of understanding and executing human instructions is one of the oldest long-standing AI challenges (Winograd, 1972). This problem has numerous applications in various domains (planning, navigation and assembly) and can help accommodate seamless interaction with personal assistants in many environments. Due to its central role in AI and wide applicability, this problem has seen a surge of interest recently (MacMahon et al., 2006; Branavan et al., 2009; Chen and Mooney, 2011; Tellex et al., 2011; Matuszek et al., 2012; Kim and Mooney, 2013; Misra et al., 2017).

Recent works (Bisk et al., 2016; Tan and Bansal, 2018) focus on exploring deep learning methods for grounding spatial language. In this popular setup, human communication with robots is viewed as a single-step process, in which a natural language (NL) instruction is provided, and an outcome is observed.

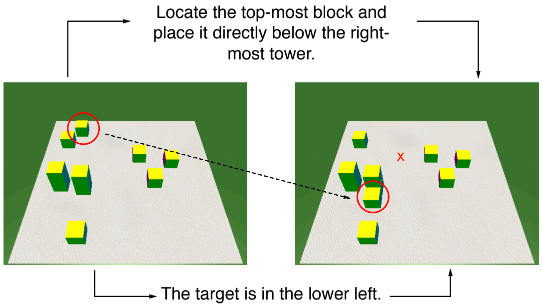


Figure 1: Based on the instruction (upper sentence) the model predicts the coordinates of the block and its target location. The 'x' represents an incorrect prediction, corrected by the provided advice (lower sentence).

Our goal in this paper is to explore different approaches for relaxing the *single step* assumption, and present initial results which we hope would motivate future work in this direction. Similar to interactive dialog systems (Allen et al., 1995; Rybski et al., 2007; Wen et al.), we view this problem as an *interactive* process, in which the human operator can observe the agents' response to their instruction and adjust it by providing advice, a form of online feedback. Specifically, the advice consists of a short sentence, simplifying the user's intent. We utilize two types of advice, one restricting the agent's search space to a general region (*restrictive advice*), and the other telling the agent the appropriate direction (up, down, left, right) to adjust its current prediction (*corrective advice*).

Our focus is on the challenging task of moving blocks on a grid (Winograd, 1972), in which the agent is given only an instruction and the state of the grid, and must predict the coordinates of where a block must be moved. We follow the difficult experimental settings suggested by (Bisk et al., 2016), in which the blocks are unlabeled and can only be referenced by their spatial properties. Fig. 1 describes our settings and uses the advice *"the target is in the lower left"*, to restrict the agents search space after observing the incor-

Proceedings of NAACL-HLT 2019, pages 1962–1967
Minneapolis, Minnesota, June 2 - June 7, 2019.

rect prediction placed the target block in the top half of the board.

To accommodate these settings, we take a two step approach. First, we ground the advice text in the simulated blocks-world environment by training a neural network. In the second step, we integrate the trained advice network into the end-to-end neural model proposed by Bisk et al. Our architecture is described in Fig. 2. The experiments we run show that this end-to-end advice model successfully grounds the meaning of our advice.

We propose four novel interactive advice-based protocols that can be applied on any robot communication architecture, ordered in terms of decreasing human effort. As expected, as human effort lessens, performance does worsen, but all protocols outperform Bisk et al. (whom our model is *identical* to besides the inclusion of advice).

Most notably, we explore the notion of model *self-generated* advice, which significantly reduces/eliminates human effort. In this approach, a model is trained to automatically generate restrictive advice for a given scenario, based on the assumption that it is easier to predict a region containing the target coordinates rather than their exact location. We validate this assumption by developing a neural architecture to predict the restrictive advice and show it can help improve the overall prediction quality, despite having no human assistance.

2 Model

This section describes the architecture we developed for understanding advice, and how to incorporate it into the original Bisk et al. model to make better predictions. We begin by defining the Blocks World task and the types of advice we use. We then introduce a model for grounding the advice and a method for incorporating the pre-trained advice understanding module into the original model. Finally, we discuss an architecture for *advice generation*, a method for self-predicting the advice to avoid any human intervention. Further details of our models and advice generation process are in the Appendix.

2.1 Blocks World Task Definition

Given an input state, consisting of all the block positions on a board, and a NL instruction, the model has to predict the coordinates of the source block to be moved and its target location. We follow the definition by (Bisk et al., 2016) and due to space constraints refer the reader to that paper.

2.2 Advice

The two types of advice we devise in this paper are designed to assist the prediction agent by providing simpler instructions in addition to the original input. The first, restrictive advice, informs the agent about the general region of the source / target coordinates, such as *top left*. These regions are determined by dividing the board into equally sized sections (two halves, four quadrants). The second type of advice, corrective advice, observes the agents' predictions and determines which direction (up, down, left, right) they must be adjusted to get closer to the target. Both of these are representative of information a human could easily provide to a robot in various ways (speech, using assisted devices, etc.), to help correct its predictions. Specific examples are shown below.

Predicted	Target	Advice
-	(-0.5, 0.5, 0.5)	In the top left.
(-0.5, 0.5, 0.9)	(-0.5, 0.5, 0.5)	Move down.

2.3 Advice Grounding

We pre-train a neural network model to accurately understand the advice. For both types of advice, a LSTM-RNN (Hochreiter and Schmidhuber, 1997) is used to read the advice sentence $s = w_1, w_2, ..., w_n$ and output the hidden state representations $\{h_n\}$. Prior to this, a word embedding layer is used to project the input words into high-dimension vectors $\{w_i\}$.

For restrictive advice, the last state from the LSTM h_n is fed along with a random coordinate into a Fully Connected (FC) layer. The network must output a positive prediction if the random coordinate is in the region described by the advice sentence, and negative otherwise.

For corrective advice, the last state from the LSTM h_n is fed along with a random coordinate into a FC layer, and the network must output a coordinate that follows the advice. For example, if the advice is *move the block down*, the predicted coordinate must be below the random input coordinate. If the advice is followed, the network receives 0 loss, otherwise a MSE regression loss.

2.4 End-to-End Training

The pre-trained model from Section 2.3 that understands various advice text is incorporated into

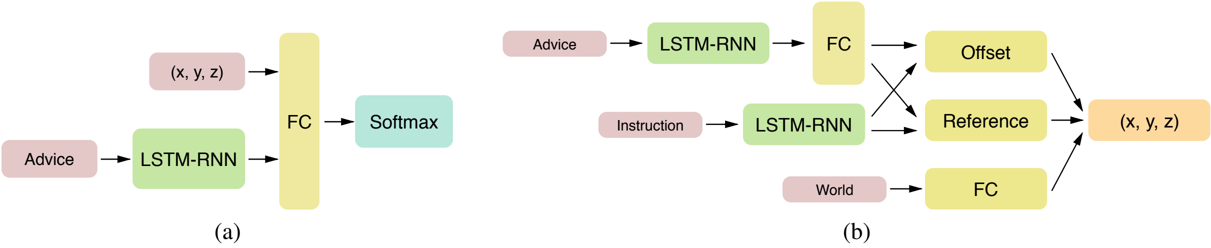


Figure 2: Our advice architectures. (a) Pre-Trained Advice Understanding Model. (b) (Bisk et al., 2016) End-to-End architecture with our pre-trained model. World represents the board state, while offset and reference represent fully connected layers used to identify the offset and reference blocks.

Model	**Source**		**Target**	
	Median	**Mean**	**Median**	**Mean**
`M1` : Bisk et al.	3.29	3.47	3.60	3.70
`M2` : Our Replication of Bisk et al.	3.13	3.42	3.29	3.50
`M3` : Tan and Bansal	–	2.21	2.78	3.07
`M4` : Restrictive Advice w/o Pre-Trained Model	3.88	3.83	3.56	3.43
`M5` : 4 Regions Restrictive Advice	2.23	2.21	2.18	2.19
`M6` : Corrective Advice	2.76	2.94	2.72	3.06
`M7` : 4 Regions Retry Advice	2.41	3.02	2.42	3.14
`M8` : 2 Regions Model Self-Generated Advice	3.01	3.31	3.08	3.36
`M9` : Input-Specific Model Self-Generated Advice	2.87	3.12	2.99	3.26

Table 1: Results for our models compared to previous models evaluated as distance from gold prediction normalized by block length for source and target coordinate prediction.

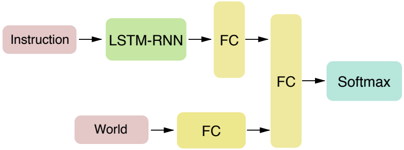


Figure 3: Model for Generating Advice.

the best performing End-to-End RNN architecture proposed in (Bisk et al., 2016) by adding a FC layer to the pre-trained LSTM state h_n and summing it with the LSTM hidden state of the original model (as shown in Figure 2b). We load and freeze the best performing parameters from our pre-trained model into the relevant portion of this end-to-end architecture, and train it on the original task of predicting the coordinates of the source / target location, with the addition of advice input.

2.5 Advice Generation

We use a neural network model to self-generate restrictive advice (as shown in Figure 3), passing the instruction into an embedding layer followed by a LSTM, the board state into a FC layer, concatenating these into a FC layer, and finally using a softmax to classify the input example into a region. We train this architecture and then run it on the test set, generate the appropriate advice based on the region the data is classified in, and use that as test advice input for the end-to-end architecture from section 2.4.

3 Experiments

Next, we present our experiments over our four different advice protocols, each with decreasing human effort and overall performance. In each protocol, we provide advice to the end-to-end model from Section 2.4, whether it is given by a human user or model self-generated. Our results, evaluated on each model's mean and median prediction error, are presented in Table 1. We always compare to the baseline Bisk et al. model, which our model is identical to besides the addition of advice (and we always beat), and the state-of-the-art best non-ensemble Tan and Bansal architecture. Note that Tan and Bansal use an advanced neural architecture and a different training procedure (source prediction trained as classification). We hypothesize that using the advice mechanism over this more complex architecture would lead to further improvements, and leave it for future work.

The pre-trained advice grounding models from Section 2.3 achieve 99.99% accuracy, and are vital, as shown by the poor performance without them (`M4` vs `M5`). These grounding models allow the end-to-end architecture to generalize to the variability in advice utterances.

3.1 Restrictive Advice

When training the end-to-end model from Section 2.4, we provide restrictive advice at training time for only half the examples. For every epoch, a different half set of examples (determined randomly) receive advice. This mechanism gives the model a chance to learn to interpret each example with and without advice, so that it can handle the interactivity without overfitting to one setup. This setup also gave the best performance.

At test time, the advice is provided only whenever the predictions fall in the wrong general region, just like a human would. As seen in Table 1, this model (`M5`) significantly outperforms both baselines (`M1`, `M3`). We note that the performance did not improve much when advice was always provided, showing that this model was able to perform well in its absence and does not rely on it (due to our choice not to provide advice all the time in training). In fact a human would only have to provide restrictive advice for 395/720 examples, and the model always follows it.[1]

3.2 Corrective Advice

We train corrective advice identically to restrictive advice from Section 3.1, except we train in two separate iterations. This is necessary as the model must learn to adjust its predictions based on the advice, which is why it is first trained to make the normal prediction (first iteration), then trained to adjust the prediction (second iteration).

In the first iteration, we train identically to (Bisk et al., 2016) with no advice, but in the second iteration corrective advice is generated based on which direction the predictions must be adjusted to be more accurate. This case is simpler than restrictive advice, since the human operator just has to provide the direction to adjust the predictions, rather than the precise region of the coordinates. However, the performance does worsen (`M5` vs `M6`).

3.3 Retry Advice

In Section 2.5, we introduced a model that was able to self-generate restrictive advice by predicting the general region of the block coordinates given the NL instruction and blocks world. Table 2 shows this model's accuracy on that task when the board is split into 4 regions. As this is a hard problem with low accuracy (`A1`), we instead generate advice for the top 2 most confident predictions (determined by the softmax scores) (`A2`).

We now introduce a new multi-step *retry* advice protocol. In the first step, the model from Section 2.5 self-generates restrictive advice based on the most confident predicted region, which it uses as input in the end-to-end model. If the user believes the coordinate prediction based on this advice is wrong, it can tell the model to "retry", and then the second most likely restrictive advice will be used. Thus, the only human feedback needed now is telling the model to "retry", rather than accurate advice as before. The performance of this (`M7`) still significantly outperforms Bisk et al. and is close to Tan and Bansal on target prediction.

Regions	**Source**	**Target**
`A1` : 4	47%	40%
`A2` : 4, Top 2 Confidence	73%	70%
`A3` : 4, Input-Specific	67%	62%

Table 2: Accuracy of model self-generated advice.

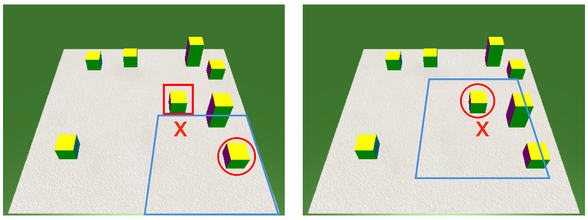
3.4 Model Self-Advice Generation

We now aim to avoid any human interaction, by letting the model completely self-generate the advice. Accomplishing it would allow us to improve the model's performance without additional human effort. We experimented with two approaches. In the first, we generate advice as described in Section 3.3. However, instead of having the user ask the model to "retry", we treat the top 2 confidence regions as a general region, and provide that as advice input as described in Section 3.1. In this case, there is a performance improvement over Bisk et al. with no human effort required (`M8` in Table 1).

Our second approach for self-generated advice aims to improve on some of the shortcomings of the first approach. Previously, when generating the advice, we had decided on four coarse-grained regions, and trained a model to classify each input example into one of these regions. In many cases,

---

[1] We note that the performance does not improve from Bisk et al. if advice is only provided at train time.



(a) (b)

Figure 4: Benefit of Input-Specific Model Self-Generated Advice
(a) The Bisk et al. model would have made a prediction ('x') close to the true block (square). However, the advice region (blue) was incorrect (due to the true block being close to the edge of it) and this led to a significantly worse prediction (circle).
(b) In the input-specific self-generated advice model, the advice region (blue) is centered at the incorrect coordinate prediction ('x'), leading to the true source block being included and a correct prediction (circle).

the true coordinate lay close to the boundary of one of these regions, often resulting in the model predicting the wrong region when self-generating the advice. This incorrect prediction would lead to significantly worse performance (when compared to the model without advice) when running the end-to-end model from Section 3.1, as the advice was incorrect (remember that the model always follows our advice, and the true coordinate is not in the advice region due to the mistake). However, if we had instead chosen our regions to be centered at the true coordinate of each input example, it would be less likely that the model would make an incorrect region prediction (since a small error would still lead to the region containing the correct coordinate). Figure 4 provides a visual explanation of this.

For this reason, we now introduce input-specific model self-generated advice. In this case, we run the Bisk et al. coordinate prediction model in two iterations. In the first iteration, we use the prediction to generate advice for a region (of the same size as in the case of 4 quadrants) centered at the predicted coordinate (see Figure 4b).[2] In the second iteration, we feed in this generated advice just like Section 3.1. This model (`M9`) achieves performance slightly worse than retry advice, and significantly better than Bisk et al., all with no human effort.[3] Table 2 shows the accuracy increase in predicting the advice now (`A3` vs `A1`). It is unsurprising that this approach to self-generating advice performs better, as now the regions are more specific to each coordinate (so there is a higher probability that the true coordinate is actually in the predicted region - see Figure 4).

We hypothesize that the performance improvements in self-generated advice happen since it is easier to predict the general region used to generate the advice rather than the specific coordinates. Previously, we have also shown the benefit of restrictive advice in improving overall coordinate prediction, so it is unsurprising that a high accuracy of advice generation leads to better overall performance. Due to this, we propose that future robot communication works take advantage of predicting and then using model self-generated advice in their end-to-end training procedure.

4 Summary

This paper takes a first step towards a stronger interaction between automated agents and their human operators, for physically grounded language understanding tasks. We focus on the popular blocks task and introduce the notion of advice, Natural Language hints provided by the human operator, correcting the model's predictions. We show that using four versions of this interactive advice driven protocol on an existing robot communication architecture, we can obtain significant performance improvements. The last method, model self-generated advice, shows the benefit of considering advice even when not designing an interactive protocol. Our future work focuses on further increasing the accuracy of the self-generated advice model, so we can achieve better performance with no human effort.

Acknowledgements

We thank the anonymous reviewers of this paper for all of their vital feedback.

This work was partially supported by the Defense Advanced Research Projects Agency (DARPA) under the ASED program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA.

---

[2] We make sure the advice region doesn't exceed the board boundaries.

[3] Note that we must re-train the model from Section 2.3 as there are now significantly more regions. The accuracy of that model is still 99.99%, and the training procedure does not change.

References

James F Allen, Lenhart K Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel Martin, Bradford Miller, Massimo Poesio, et al. 1995. The trains project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):7–48.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory.

Joohyun Kim and Raymond J Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 218–227.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. of the International Conference on Machine Learning (ICML)*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proc. of the International Conference on Machine Learning (ICML)*.

Dipendra K Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1310–1318.

Paul E Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. 2007. Interactive robot task training through dialog and demonstration. In *Proc. of the ACM/IEEE international conference on Human-robot interaction*, pages 49–56. ACM.

Hao Tan and Mohit Bansal. 2018. Source-target inference models for spatial instruction understanding. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proc. of the Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Generating Knowledge Graph Paths from Textual Definitions using Sequence-to-Sequence Models

Victor Prokhorov,[1] Mohammad Taher Pilehvar[1,2] and Nigel Collier[1]
[1]Department of Theoretical and Applied Linguistics, University of Cambridge
[2]School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
`vp361@cam.ac.uk, pilehvar@iust.ac.ir, nhc30@cam.ac.uk`

Abstract

We present a novel method for mapping unrestricted text to knowledge graph entities by framing the task as a sequence-to-sequence problem. Specifically, given the encoded state of an input text, our decoder directly predicts paths in the knowledge graph, starting from the root and ending at the target node following hypernym-hyponym relationships. In this way, and in contrast to other text-to-entity mapping systems, our model outputs hierarchically structured predictions that are fully interpretable in the context of the underlying ontology, in an end-to-end manner. We present a proof-of-concept experiment with encouraging results, comparable to those of state-of-the-art systems.

1 Introduction

Text-to-entity mapping is the task of associating a text with a concept in a knowledge graph (KG) or an ontology (we use two terms, interchangeably). Recent works (Kartsaklis et al., 2018; Hill et al., 2015) use neural networks to project a text to a vector space where the entities of a KG are represented as continuous vectors. Despite being successful, these models have two main disadvantages. First, they rely on a predefined vector space which is used as a gold standard representation for the entities in a KG. Therefore, the quality of these algorithms depends on how well the vector space is represented. Second, these algorithms are not interpretable; hence, it is impossible to understand why a certain text was linked to a particular entity.

To address these issues we propose a novel technique which first represents an ontology concept as a sequence of its ancestors in the ontology (hypernyms) and then maps the corresponding textual description to this unique representation. For example, given the textual description of the concept *swift* ("small bird that resembles a swallow and is noted for its rapid flight"), we map it to the hierarchical sequence of entities in a lexical ontology: *animal* → *chordate* → *vertebrate* → *bird* → *apodiform_bird*. This sequence of nodes constitutes a path.[1]

Our model is based on a sequence-to-sequence neural network (Sutskever et al., 2014) coupled with an attention mechanism (Bahdanau et al., 2014). Specifically, we use an LSTM (Hochreiter and Schmidhuber, 1997) encoder to project the textual description into a vector space and an LSTM decoder to predict the sequence of entities that are relevant to this definition. With this framework we do not need to rely on the pre-existing vector space of the entities, since the decoder explicitly learns topological dependencies between the entities of the ontology. Furthermore, the proposed model is more interpretable for two reasons. First, instead of the closest points in a vector space, it outputs paths; therefore, we can trace all predictions the model makes. Second, the attention mechanism allows to visualise which words in a textual description the model selects while predicting a specific concept in the path. In this paper, we consider rooted tree graphs[2] only and leave the extension of the algorithm for more generic graphs to future work.

We evaluate the ability of our model in generating graph paths for previously unseen textual definitions on seven ontologies (Section 3). We show that our technique either outperforms or performs on a par with a competitive multi-sense LSTM model (Kartsaklis et al., 2018) by better utilising external information in the form of word embeddings. The code and resources for the paper can

---

[1] We only consider hypernymy relations, from the root to the parent node (*apodiform_bird*) of the entity *swift*.

[2] Only single root is allowed. If a tree has more than one root, one can create a dummy root node and connect the roots of the tree to it.

Proceedings of NAACL-HLT 2019, pages 1968–1976
Minneapolis, Minnesota, June 2 - June 7, 2019.

be found at https://github.com/VictorProkhorov/Text2Path.

2 Methodology

We assume that an ontology is represented as a rooted tree graph $G = (V, E, T)$, where V is a set of entities (e.g. synsets in WordNet), E is a set of hyponymy edges, and T is a set of textual descriptions such that $\forall v \in V$ there is a $t_v \in T$.

2.1 Node representation

We assume that an ontological concept can be defined by either using a textual description from a dictionary or hypernyms of the defining concept in the ontology. For example, to define the noun *swift* one can use the dictionary definition mentioned previously. Alternatively, the concept of *swift* can be understood from its hypernyms, e.g. in the trivial case one can say that *swift* is an *animal*. This definition is not very useful since *animal* is a hypernym for many other nouns. To provide a more specific definition, one can use a sequence of hypernyms e.g. *animal* → *chordate* → *vertebrate* → *bird* → *apodiform_bird* starting from the most abstract node (root of an ontology) to the most specif (parent node of the noun).

More formally, for each entity $v \neq v_{root} \in V$ we create a path p_v. Each p_v starts from v_{root} and ends with a hypernym of v, i.e., the hierarchical order of entities is preserved. Then the path p_v is aligned with t_v such that each node is defined by a textual definition and a path. This set of aligned representations is used to train the model.

The path representation of an entity ends with its parent node. Therefore, a leaf node will not be present in any of the paths. This is problematic if a novel definition should be attached to a leaf. To alleviate this issue we employ the "dummy source sentences" technique from neural machine translation (NMT) (Sennrich et al., 2016). We create an additional set of paths from the root node to each leaf. As for the textual definition we leave it empty.

2.2 Model

We use a sequence-to-sequence model with an attention mechanism to map a textual description of a node to its path representation.

Encoder. To encode a textual definition $t_v = (w_i)_{i=1}^{N}$, where N is sentence length, we first map each word w_i to a dense embedding e_{w_i} and then use a bi-directional LSTM to project the sequence into a latent representation. The final encoding state is obtained by concatenating the forward and backward hidden states of the bi-LSTM.

Decoder. Decoding the path representation of a node from the latent state of the textual description is done again with an LSTM decoder. Similarly to the encoding stage, we map each symbol in the path $p_v = (s_j)_{j=1}^{M}$ to a dense embedding e_{s_j}, where M is the path length. To calculate the probability of the path symbol s_j at time step j we first represent the path sequence as $h^*_j = \text{LSTM}(e^j_s, h^*_{j-1})$. Then, we concatenate h^*_j with the context vector c_j (defined next) and pass the concatenated representation $[h^*_j; c_j]$ through the softmax function, i.e. $s_j = \max(\text{softmax}(\mathbf{W}[h^*_j; c_j]))$, where $\mathbf{W}$ is a weight parameter. To calculate the context vector c_j we use an attention mechanism, $e_{ji} = v_a^T \tanh(\mathbf{W_a} h_i + \mathbf{U_a} h^*_j)$ and $c_j = \sum_i^N \text{softmax}(e_{ji}) h_i$, where v_a, $\mathbf{W_a}$ and $\mathbf{U_a}$ are the weight parameters, over the words in the text description.

3 Experimental Setup

Ontologies. We experimented with seven graphs four of which are related to the bio-medical domain: Phenotype And Trait Ontology[3] (PATO), Human Disease Ontology (Schriml et al., 2012, HDO), Human Phenotype Ontology (Robinson et al., 2008, HPO) and Gene Ontology[4] (Ashburner et al., 2000, GO). The other three graphs, i.e. $\text{WN}_{\text{animal.n.01}}$[5], $\text{WN}_{\text{plant.n.02}}$ and $\text{WN}_{\text{entity.n.01}}$ are subgraphs of the WordNet 3.0 (Fellbaum, 1998). We present the statistics of the graphs in Table 1.

Ontology Preprocessing. All the ontologies we experimented with are represented as directed acyclic graphs (DAGs). This creates an ambiguity for node path definitions since there are multiple pathways from a root concept to other concepts. We have assumed that a single unambiguous pathway will reduce the complexity of the problem and leave the comparison with ambiguous pathways (which would inevitably involve a more complex model) to future work. To convert a DAG to a tree

---

[3] http://www.obofoundry.org

[4] After prerocessing GO we took its largest connected component.

[5] The subscript in 'WN' indicates the name of the root node of the graph.

Graphs	|V|	Depth	Branch	A.D
PATO	1742	(4.94,10)	(3.95,92)	20
$\text{WN}_{\text{animal.n.01}}$	3999	(6.94,12)	(3.79,52)	26
$\text{WN}_{\text{plant.n.02}}$	4487	(4.70,9)	(5.91,357)	28
HDO	9095	(5.92,12)	(4.59,222)	27
HPO	13348	(6.95,14)	(3.40,32)	24
GO	29682	(6.40,14)	(3.28,172)	21
$\text{WN}_{\text{entity.n.01}}$	74374	(8.01,18)	(4.52,402)	36

Table 1: Statistics of the Graphs. $|\mathbf{V}|$ is the number of nodes, *depth* is the path length from the root of a graph to a node, *branch* is the number of neighbours a node has (leaves were removed from the calculation). The first value in the parentheses corresponds to the average and the second to the maximum value. A.D stands for average number of decisions the model makes to infer a path, i.e A.D = average depth × average branch.

we constrain each entity to have only one parent node. The edges between the other parent nodes are removed.[6]

Path Representations. We also experiment with two path representations. Our first approach, *text2nodes*, uses the label of an entity (cf. Section 1) to represent a path. This is not efficient since the decoder of the model needs to select between all of the entities in an ontology and also requires more parameters in the model. Our second approach, *text2edges*, to reduce the number of symbols for the model to choose from, uses edges to represent the path. To do this we create an artificial vocabulary of the size $\Delta(G)$, where $\Delta(G)$ corresponds to the maximum degree of a node. Each edge in the graph is labeled using the artificial vocabulary. For the example in Section 1, the path would be *animal* −[a]→ *chordate* −[b]→ *vertebrate* −[c]→ *bird* −[d]→ *apodiform_bird* where {a,b,c,d} is the artificial vocabulary. In the resulting path we discard labels for the entities; therefore, the path reduces to: [a]→ [b]→ [c]→ [d].

3.1 Baselines

Bag-of-Words Linear Regression (BOW-LR): To represent a textual definition in a vector space we first use a pre-trained set of word embeddings (Speer et al., 2017) to represent words in the definition and then find the mean of the word embeddings. As for the ontology, we use node2vec (Grover and Leskovec, 2016), to represent each entity in a vector space. To align the two vector spaces we use linear regression.

Multi-Sense LSTM (MS-LSTM): Kartsaklis et al. (2018) proposed a model that achieves state-of-the-art results on the text-to-entity mapping on the *Snomed CT*[7] dataset. The approach uses a novel multi-sense LSTM, augmented with an attention mechanism, to project the definition to the ontology vector space. Additionally, for a better alignment between the two vector spaces, the authors augmented the ontology graph with textual features.

3.2 Evaluation Metric

To perform evaluation of the models described above we used Ancestor-F1 score (Mao et al., 2018). This metric compares the ancestors ($is-a_{model}$) of the predicted node with the ancestors ($is-a_{gold}$) of the gold node in the taxonomy.

$$P = \frac{|is-a_{model} \wedge is-a_{gold}|}{|is-a_{model}|},$$

$$R = \frac{|is-a_{model} \wedge is-a_{gold}|}{|is-a_{gold}|},$$

where P and R are precision and recall, respectively. The Ancestor-F1 is then defined as:

$$2 \times \frac{P \times R}{P + R}.$$

3.3 Intrinsic Evaluation

To verify the reliability of our model on text-to-entity mapping we did a set of experiments on the seven graphs (Section 3) where we map a textual definition of a concept to a path.

To conduct the experiments we randomly sampled 10% of leaves from the graph. From this sample, 90% are used to evaluate the model and 10% are used to tune the model. The remaining nodes in the graph are used for training. We sample leaves for two reasons: (1) to predict a leaf, the model needs to make the maximum number of (correct) predictions and (2) this way we do not change the original topology of the graph. Note that the sampled nodes and their textual definitions are not present in the training data.

Both baselines predict a single entity instead of a path. To have the same evaluation framework for all the models, for each node predicted by the baselines we create[8] a path from the root of the node to the predicted node. However, we want

---

[6] The choice of an edge is performed on random basis.

[7] https://www.snomed.org/snomed-ct

[8] We used NetworkX (https://networkx.github.io) to find a path from predicted node to the root of a graph.

Models	PATO	WN$_{\text{animal.n.01}}$	WN$_{\text{plant.n.02}}$	HDO	HPO	GO	WN$_{\text{entity.n.01}}$
BOW-LR	0.79	0.75	0.65	0.55	0.63	0.32	0.41
MS-LSTM$_{\lambda = 0}$	0.77	0.73	0.62	0.70	0.72	0.69	0.51
MS-LSTM$_{\lambda = 0.5}$	0.80	0.76	0.65	0.70	0.73	0.70	0.57
MS-LSTM$_{\lambda = 1}$	0.75	0.66	0.57	0.65	0.63	0.62	0.51
text2nodes	0.75	0.66	0.66	0.69	0.62	0.67	0.60
text2edges	0.76	0.68	0.66	0.69	0.69	0.69	0.61
MS-LSTM$^{*}_{\lambda=0.5}$	0.81	0.76	0.66	0.71	0.74	0.71	0.58
text2nodes*	0.83	0.71	0.68	0.71	0.69	0.70	0.62
text2edges*	**0.83**	**0.77**	**0.70**	**0.73**	**0.74**	**0.72**	**0.65**

Table 2: Ancestor F1 results. Numbers in bold represent the best performing system on a graph. Models marked with $*$ make use of pre-trained word embedding in their encoder. Lambda (λ) is defined in Section 3.1. We use the same number of epochs, batch size and number of latent dimensions both for MS-LSTM and our models (Appendix C).

to emphasize that this is disadvantageous for our model, since all the symbols in the path are predicted by it and in the case of the baselines only a single node is predicted.

The results are presented in Table 2. Models that are in the last three rows of Table 2 use pre-trained word embeddings (Speer et al., 2017) in the encoder. MS-LSTM and our models that are above the last three rows use randomly initialised word vectors. We had four observations: (1) without pre-trained word embeddings in the encoder our model outperforms the best MS-LSTM$_{\lambda = 0.5}$ only on two of the seven graphs, (2) the text2edges* model outperforms all the other models including MS-LSTM$^{*}_{\lambda=0.5}$, (3) the text2edges model can better exploit pre-trained word embeddings than MS-LSTM, (4) our model performs better when the paths are represented using edges (rather than nodes). We also found that there is a strong negative correlation (Spearman: -0.75, Pearson: -0.80) between A.D. (Table 3) and the Ancestor F1 score for the text2edges* model, meaning that with an increase in A.D. the Ancestor F1 score decreases.

3.4 Error Analysis

We carried out an analysis on the outputs of our best-performing model, i.e. text2edges* with pre-trained word embeddings. One factor that affects the performance is the number of invalid sequences predicted by the text2nodes and text2edges models. An invalid sequence is the path that does not exist in the original graph. This happens because at each time step the decoder outputs a distribution over all the nodes/edges and not just over possible children nodes. We therefore performed a count of the number of invalid sequences produced by the model. The percentage of invalid sequences is in the range of 1.82% - 8.50% (Appendix B), which is relatively low. This analysis was also performed by J. Kusner et al. (2017). To guarantee that the model always produces valid graphs, they use a context-free grammar. A similar method can be adapted in our work.

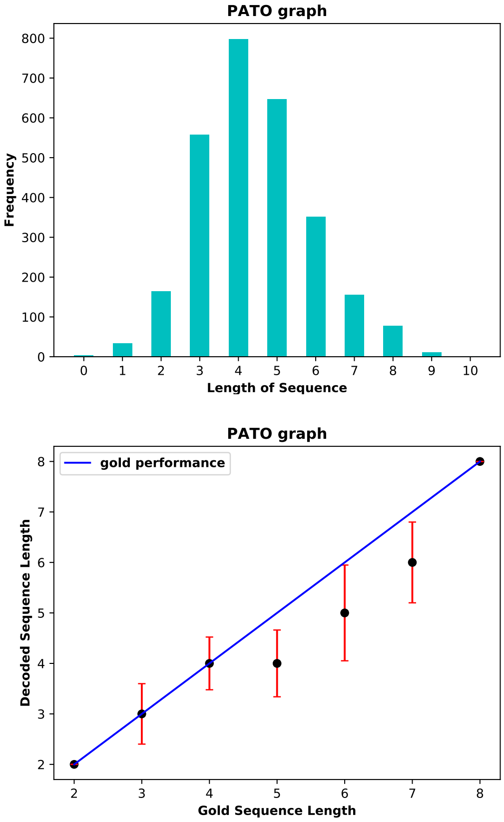


Figure 1: The graph on top shows the length of sequence vs length frequency on a training set. The graph on the bottom shows the length of the gold sequence vs mean length of decoded sequence on the test set.

Another factor that affects the performance is the length of the generated paths which is expected to match the length of the gold path. To test this, we compared the mean length of the generated sequences with the length of the gold path (the graph on the bottom of Figure 1). Also, in the training set, we associate the length of the sequences with their frequencies (the graph on the top of Figure 1). We found that (1) the length of the generated paths are biased towards the more frequent paths in the training data, (2) if the length of a path is not frequent in the training data, the model either under-generates or over-generates the length (Appendix D).

4 Related Work

Text-to-entity mapping is an essential component of many NLP tasks, e.g. fact verification (Thorne et al., 2018) or question answering (Yih et al., 2015). Previous work has approached this problem with pairwise learning-to-rank method (Leaman et al., 2013) or phrase-based machine translation (Limsopatham and Collier, 2015). However, these methods generally ignore ontology's structure. More recent work has viewed the problem of text-to-entity mapping as a projection of a textual definition to a single point in a KG (Kartsaklis et al., 2018; Hill et al., 2015). However, despite potential advantages, such as being more interpretable and less brittle (model predicts multiple related entities instead of one), path-based approaches have received relatively little attention. Instead of predicting a single entity, path-based models, such as the one we proposed in this paper, try to map a textual definition to multiple relevant entities in an external resource.

5 Conclusion and Future Work

We presented a model that maps textual definitions to interpretable ontological pathways. We evaluated the proposed technique on seven semantic graphs, showing that it can perform competitively with respect to existing state-of-the-art text-to-entity systems, while being more interpretable and self-contained. We hope this work will encourage further research on path-based text-to-entity mapping algorithms. A natural next step will be to extend our framework to DAGs. Furthermore, we plan to constrain our model to always predict paths that exist in the graph, as we discussed above.

Acknowledgments

We would like to thank the anonymous reviewers for their comments. Also, we would like to thank Dimitri Kartsaklis and Ehsan Shareghi for helpful discussions and comments. This research was supported by an EPSRC Experienced Researcher Fellowship (N. Collier: EP/M005089/1) and an MRC grant (M.T. Pilehvar: MR/M025160/1). We gratefully acknowledge the donation of a GPU from the NVIDIA Grant Program.

References

Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. 2000. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *CoRR*, abs/1504.00548.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Matt J. Kusner, Brooks Paige, and Jos Miguel Hernndez-Lobato. 2017. Grammar variational autoencoder.

Dimitri Kartsaklis, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Mapping text to knowledge graph entities using multi-sense lstms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1959–1970. Association for Computational Linguistics.

Robert Leaman, Rezarta Dogan, and Zhiyong lu. 2013. Dnorm: Disease name normalization with pairwise learning to rank. *Bioinformatics (Oxford, England)*, 29.

Nut Limsopatham and Nigel Collier. 2015. Adapting phrase-based machine translation to normalise medical terms in social media messages. *CoRR*, abs/1508.02285.

Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-end reinforcement learning for automatic taxonomy induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2462–2472. Association for Computational Linguistics.

Peter N. Robinson, Sebastian Köhler, Sebastian B Bauer, Dominik Seelow, Denise Horn, and Stefan Mundlos. 2008. The human phenotype ontology: a tool for annotating and analyzing human hereditary disease. *American journal of human genetics*, 83 5:610–5.

Lynn M. Schriml, Cesar Arze, Suvarna Nadendla, Yu-Wei Wayne Chang, Mark Mazaitis, Victor Felix, Gang Feng, and Warren A. Kibbe. 2012. Disease ontology: a backbone for disease semantic integration. In *Nucleic Acids Research*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. Association for Computational Linguistics.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Wen-Tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.

A DAGs

Graphs	**Mult.P$_{\%}$**	**AV.P**
PATO	31.29	2.97
WN$_{\text{animal.n.01}}$	0.88	2.00
WN$_{\text{plant.n.02}}$	0.16	2.00
HDO	16.23	2.13
HPO	23.24	2.23
GO	64.01	2.77
WN$_{\text{entity.n.01}}$	1.91	2.03

Table 3: Statistics of nodes with multiple inheritance. Mult.P$_{\%}$ stands for percentage of nodes with more than one parent node. AV.P stands for average number of parents a node with multiple inheritance has.

B Invalid Sequences

Graphs	**Invalid$_{\%}$**	**N$_{\text{total}}$**
PATO	1.82	110
WN$_{\text{animal.n.01}}$	4.56	263
WN$_{\text{plant.n.02}}$	2.23	314
HDO	4.02	622
HPO	7.08	847
GO	6.94	1845
WN$_{\text{entity.n.01}}$	8.50	5191

Table 4: Statistics of invalid sequences. Invalid$_{\%}$ is the percentage of invalid sequences and N$_{\text{total}}$ is the total number of sequences that were tested.

C Settings for Models

BOW-LR: To represent an ontology in a vector space we use node2vec https://snap.stanford.edu/node2vec/. For all the graphs the following hyperparameters of the algorithm are the same: *walk-length= 5*, *window-size=5* and *iter=40*. As for the number of dimensions we set it to 128 for PATO, WN$_{\text{animal.n.01}}$, WN$_{\text{plant.n.02}}$, HDO and HPO graphs. For GO and WN$_{\text{entity.n.01}}$ graphs we set it to 256. All the other parameters of node2vec are default.

We do not modify the numberbatch embeddings https://github.com/commonsense/conceptnet-numberbatch. If a word in a textual definition is missing we initilised the embedding for this word with zeros.

For all the graphs to map the textual vector space into an ontology vector space we use the linear regression model from the *scikit-learn* API https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

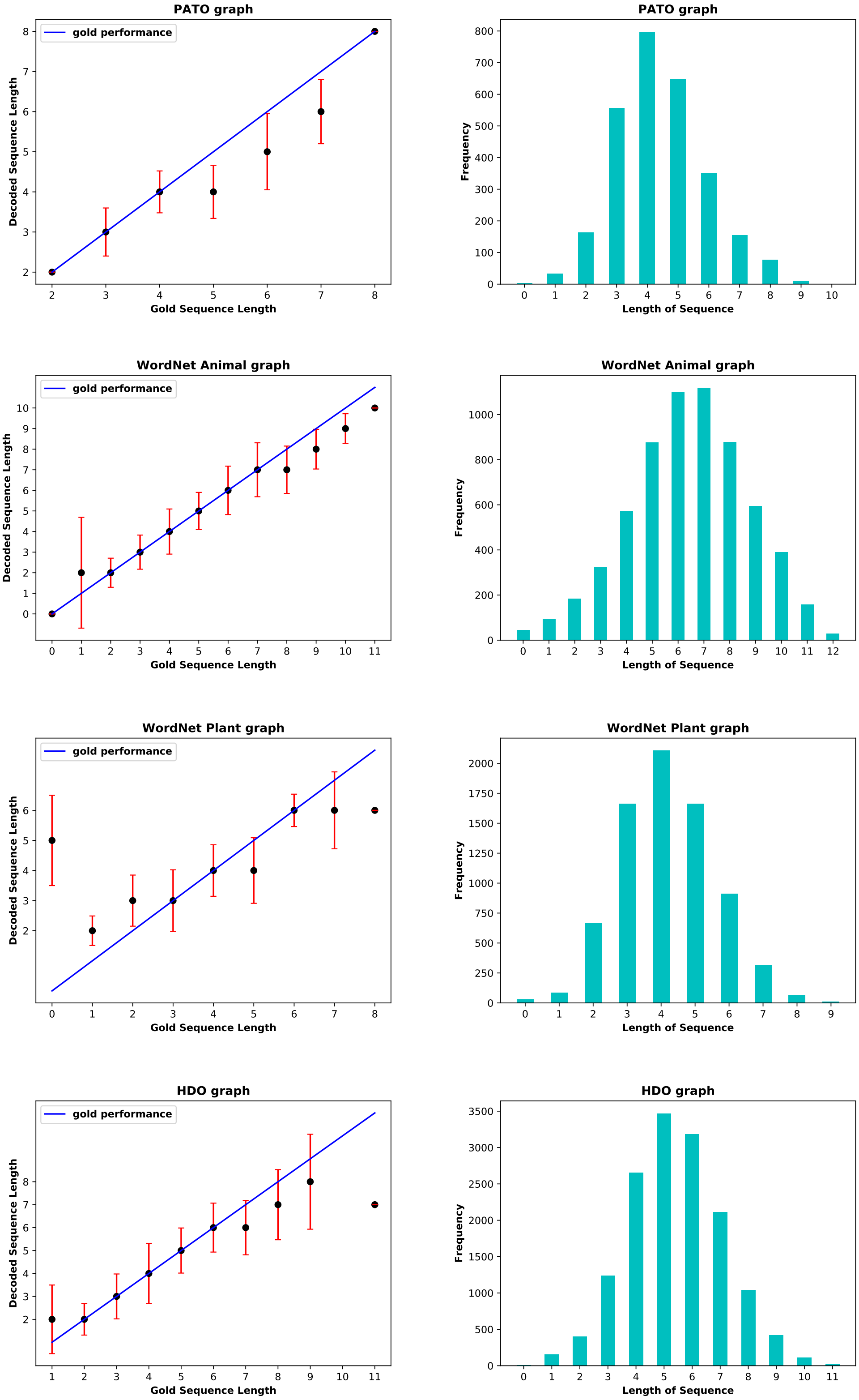


Figure 2: On the left graphs show: length of gold sequence vs mean length of decoded sequence on a test set; On the right graphs show: length of sequence vs length frequency on a training set.

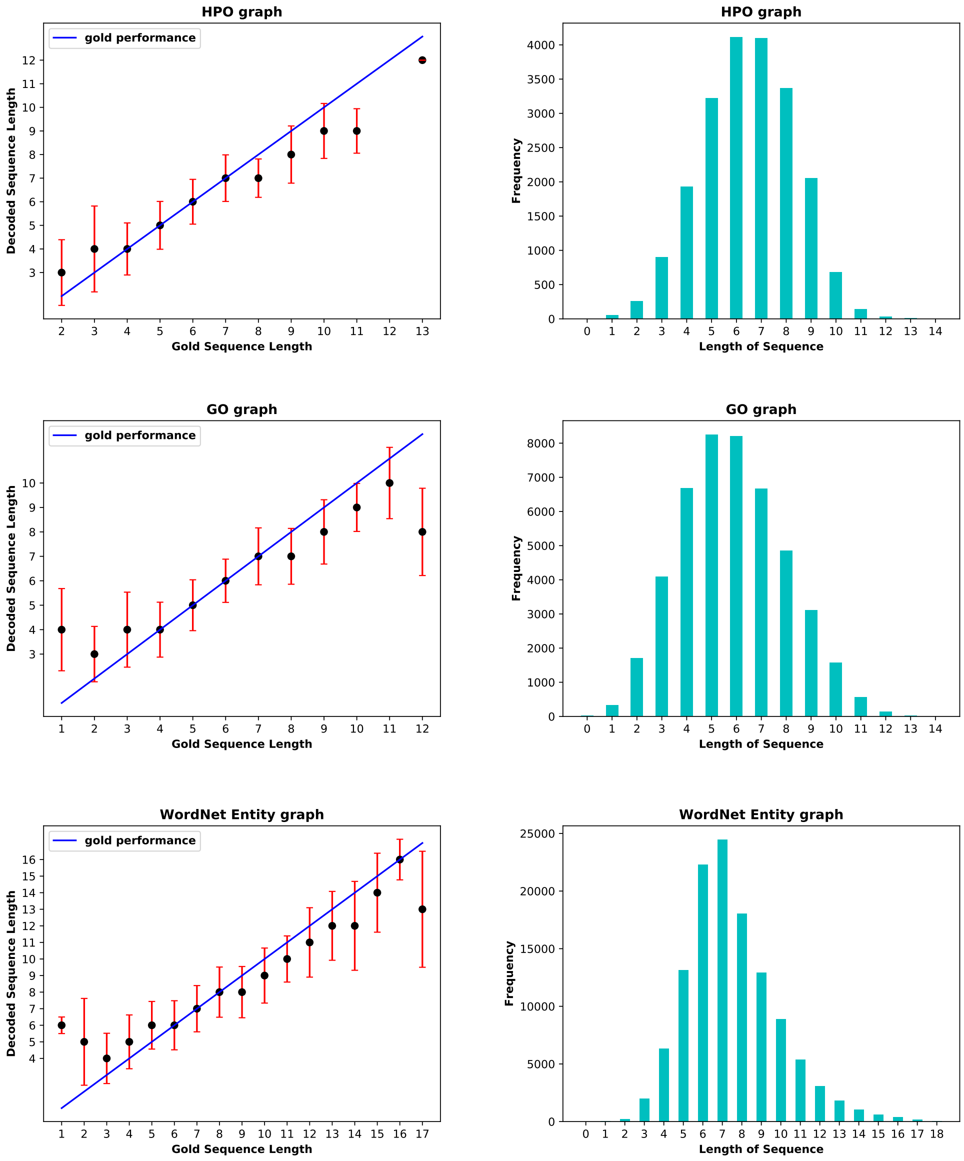


Figure 3: Continuation of Figure 2. On the left graphs show: length of gold sequence vs mean length of decoded sequence on a test set; On the right graphs show: length of sequence vs length frequency on a training set.

MS-LSTM: There are only two hyper-parameters that we vary during the embedding of ontology concepts: λ (we report the values in the paper) and the embedding size of the concepts. We set it to 128 for PATO, $\mathrm{WN}_{\mathrm{animal.n.01}}$, $\mathrm{WN}_{\mathrm{plant.n.02}}$, HDO and HPO graphs. For GO and $\mathrm{WN}_{\mathrm{entity.n.01}}$ graphs we set it to 256.

For all the graphs the model is trained for 300 epochs, dimensions of word embeddings is set to 64 and bi-LSTM is used instead of LSTM. Batch size is set to 16 and the number of latent dimensions in bi-LSTM is set to 128 for the PATO, $\mathrm{WN}_{\mathrm{animal.n.01}}$, $\mathrm{WN}_{\mathrm{plant.n.02}}$, HDO and HPO graphs. For GO and $\mathrm{WN}_{\mathrm{entity.n.01}}$ graphs we set these parameters to 128 and 256 respectively. All the other hyper-parameters are default.

When we use pre-trained word embeddings we reduce (with PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) its dimensions from 300 to 64.

Our Model: For all the graphs the model is trained for 300 epochs, dimensions of word embeddings (also for node/edges embeddings) is set to 64 and bi-LSTM is used in the encoder and LSTM in the decoder. Batch size is set to 16 and the number of latent dimensions in bi-LSTM encoder and LSTM decoder is set to

128 for the PATO, $\text{WN}_{\text{animal.n.01}}$, $\text{WN}_{\text{plant.n.02}}$, HDO and HPO graphs. For GO and $\text{WN}_{\text{entity.n.01}}$ graphs we set these parameters to 128 and 256 respectively. For optimizer we used *RMSProp* (https://www.tensorflow.org/api_docs/python/tf/train/RMSPropOptimizer) with learning rate = 0.001.

When we use pre-trained word embeddings we reduce (with PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) its dimensions from 300 to 64.

D Length of Generated Path

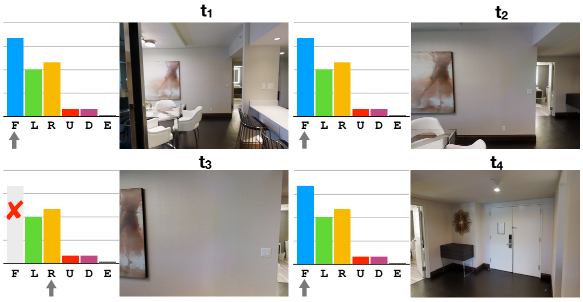
In Figure 2 and 3 the blue line indicates the ideal scenario i.e. mean length of the generated sequences is equal to the gold length. Black dot is the mean of the length of decoded sequences and the red bars are the standard deviation. One can notice that the general trend is following: for short sequences the mode generates (slightly) longer sequences and for the long sequences it generated (slightly) shorter sequences than the gold standard. Another trend is that the sequences of the certain length are matching the gold standard. To understand why this is happening one needs to look at the graph which relate the length of the sequence in the training corpus and the frequency of this length in the corpus. It is become clear there is a correlation between the two. Such as the model tends to generate the sequence of the length that is presented the most in the training data.

Shifting the Baseline:
Single Modality Performance on Visual Navigation & QA

Jesse Thomason **Daniel Gordon** **Yonatan Bisk**
Paul G. Allen School of Computer Science and Engineering
`{jdtho,xkcd,ybisk}@cs.washington.edu`

Abstract

We demonstrate the surprising strength of unimodal baselines in multimodal domains, and make concrete recommendations for best practices in future research. Where existing work often compares against *random* or *majority class* baselines, we argue that unimodal approaches better capture and reflect dataset biases and therefore provide an important comparison when assessing the performance of multimodal techniques. We present unimodal ablations on three recent datasets in visual navigation and QA, seeing an up to 29% absolute gain in performance over published baselines.



Actions: **F**orward, turn **L**eft & **R**ight, tilt **U**p & **D**own, **E**nd

Figure 1: Navigating without vision leads to sensible navigation trajectories in response to commands like "walk past the bar and turn right". At t_3, "forward" is unavailable as the agent would collide with the wall.

1 Introduction

All datasets have biases. Baselines should capture these regularities so that outperforming them indicates a model is actually solving a task. In multimodal domains, bias can occur in any subset of the modalities. To address this, we argue it is not sufficient for researchers to provide random or majority class baselines; instead we recommend presenting results for unimodal models. We investigate visual navigation and question answering tasks, where agents move through simulated environments using egocentric (first person) vision. We find that unimodal ablations (e.g., language only) in these seemingly multimodal tasks can outperform corresponding full models (§4.1).

This work extends observations made in both the Computer Vision (Goyal et al., 2018; Cirik et al., 2018) and Natural Language (Mudrakarta et al., 2018; Glockner et al., 2018; Poliak et al., 2018; Gururangan et al., 2018; Kaushik and Lipton, 2018) communities that complex models often perform well by fitting to simple, unintended correlations in the data, bypassing the complex grounding and reasoning that experimenters hoped was necessary for their tasks.

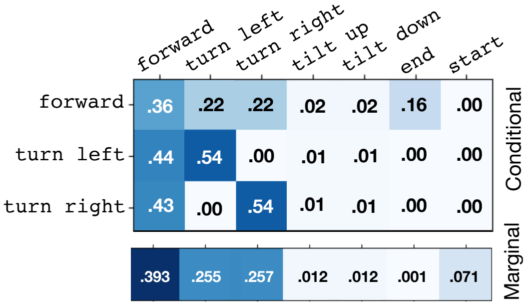
We ablate models from three recent papers: (1) navigation (Figure 1) using images of real homes paired with crowdsourced language descriptions (Anderson et al., 2018); and (2, 3) navigation and egocentric question answering (Gordon et al., 2018; Das et al., 2018a) in simulation with synthetic questions. We find that unimodal ablations often outperform the baselines that accompany these tasks.

Recommendation for Best Practices: Our findings show that in the new space of visual navigation and egocentric QA, all modalities, even an agent's action history, are strongly informative. Therefore, while many papers ablate either language *or* vision, new results should ablate *both*. Such baselines expose possible gains from unimodal biases in multimodal datasets irrespective of training and architecture details.

2 Ablation Evaluation Framework

In the visual navigation and egocentric question answering tasks, at each timestep an agent receives an observation and produces an action. Actions can move the agent to a new location or heading

Proceedings of NAACL-HLT 2019, pages 1977–1983
Minneapolis, Minnesota, June 2 - June 7, 2019.



	forward	turn left	turn right	tilt up	tilt down	end	start
forward	.36	.22	.22	.02	.02	.16	.00
turn left	.44	.54	.00	.01	.01	.00	.00
turn right	.43	.00	.54	.01	.01	.00	.00
Marginal	.393	.255	.257	.012	.012	.001	.071

Figure 2: $P(act = col|prev = row)$ and marginal action distributions in Matterport training. Peaked distributions enable agents to memorize simple rules like not turning left immediately after turning right, or moving forward an average number of steps.

(e.g., *turn left*), or answer questions (e.g., *answer 'brown'*). At timestep t, a multimodal model $\mathcal{M}$ takes in a visual input $\mathcal{V}_t$ and language question or navigation command $\mathcal{L}$ to predict the next action a_t. The navigation models we examine also take in their action from the previous timestep, a_{t-1}, and 'minimally sensed' world information W specifying which actions are available (e.g., that *forward* is unavailable if the agent is facing a wall).

$$a_t \leftarrow \mathcal{M}(\mathcal{V}_t, \mathcal{L}, a_{t-1}; W) \quad (1)$$

In each benchmark, $\mathcal{M}$ corresponds to the author's released code and training paradigm. In addition to their full model, we evaluate the role of each input modality by removing those inputs and replacing them with zero vectors. Formally, we define the full model and three ablations:

$$\begin{array}{lll} \text{Full Model} & \text{is} & \mathcal{M}(\mathcal{V}_t, \mathcal{L}, a_{t-1}; W) \quad (2) \\ \mathcal{A} & \text{is} & \mathcal{M}(\vec{0}, \vec{0}, a_{t-1}; W) \quad (3) \\ \mathcal{A}+\mathcal{V} & \text{is} & \mathcal{M}(\mathcal{V}_t, \vec{0}, a_{t-1}; W) \quad (4) \\ \mathcal{A}+\mathcal{L} & \text{is} & \mathcal{M}(\vec{0}, \mathcal{L}, a_{t-1}; W) \quad (5) \end{array}$$

corresponding to models with access to $\mathcal{A}$ction inputs, $\mathcal{V}$ision inputs, and $\mathcal{L}$anguage inputs. These ablations preserve the architecture and number of parameters of $\mathcal{M}$ by changing only its inputs.

3 Benchmark Tasks

We evaluate on navigation and question answering tasks across three benchmark datasets: Matterport Room-to-Room (no question answering component), and IQUAD V1 and EQA (question answering that requires navigating to the relevant scene in the environment) (Anderson et al., 2018; Gordon et al., 2018; Das et al., 2018a). We divide the latter two into separate navigation and question answering components. We then train and evaluate models separately per subtask to analyze accuracy.

Figure 3: IQA data construction attempts to make both the question and image necessary for QA.

3.1 Matterport Room-to-Room

An agent is given a route in English and navigates through a discretized map to the specified destination (Anderson et al., 2018). This task includes high fidelity visual inputs and crowdsourced natural language routes.

Published Full Model: At each timestep an LSTM decoder uses a ResNet-encoded image $\mathcal{V}_t$ and previous action a_{t-1} to attend over the states of an LSTM language encoder ($\mathcal{L}$) to predict navigation action a_t (seen in Figure 2).

Published Baseline: The agent chooses a random direction and takes up to five forward actions, turning right when no forward action is available.

3.2 Interactive Question Answering

IQUAD V1 (Gordon et al., 2018) contains three question types: existence (e.g., *Is there a ...?*), counting (e.g., *How many ...?*) where the answer ranges from 0 to 3, and spatial relation: (e.g., *Is there a ... in the ...?*). The data was constructed via randomly generated configurations to weaken majority class baselines (Figure 3). To evaluate the navigation subtask, we introduce a new THOR-Nav benchmark.[1] The agent is placed in a random location in the room and must approach one of *fridge, garbage can,* or *microwave* in response to a natural language question.

Although we use the same full model as Gordon et al. (2018), our QA results are not directly comparable. In particular, Gordon et al. (2018) do not quantify the effectiveness of the QA component independent of the scene exploration (i.e. navigation and interaction). To remove the scene explo-

---

[1] Formed from a subset of IQUAD V1 questions.

ration steps of Gordon et al. (2018), we provide a complete ground-truth view of the environment.[2] We use ground-truth rather than YOLO (Redmon et al., 2016) due to speed constraints.

Nav Full Model: The image and ground-truth semantic segmentation mask $\mathcal{V}_t$, tiled question $\mathcal{L}$, and previous action a_{t-1} are encoded via a CNN which outputs a distribution over actions. Optimal actions are learned via teacher forcing.

Nav Baseline: The agent executes 100 randomly chosen navigation actions then terminates. In AI2THOR (Kolve et al., 2017), none of the kitchens span more than 5 meters. With a step-size of 0.25 meters, we observed that 100 actions was significantly shorter than the shortest path length.

Published QA Full Model: The question encoding $\mathcal{L}$ is tiled and concatenated with a top-down view of the ground truth location of all objects in the scene $\mathcal{V}$. This is fed into several convolutions, a spatial sum, and a final fully connected layer which outputs a likelihood for each answer.

Published QA Baseline: We include the majority class baseline from Gordon et al. (2018).

3.3 Embodied Question Answering

EQA (Das et al., 2018a) questions are programmatically generated to refer to a single, unambiguous object for a specific environment, and are filtered to avoid easy questions (e.g., *What room is the bathtub in?*). At evaluation, an agent is placed a fixed number of actions away from the object.

Published Nav Full Model: At each timestep, a planner LSTM takes in a CNN encoded image $\mathcal{V}_t$, LSTM encoded question $\mathcal{L}$, and the previous action a_{t-1} and emits an action a_t. The action is executed in the environment, and then a lower-level controller LSTM continues to take in new vision observations and a_t, either repeating a_t again or returning control to the planner.

Published Nav Baseline: This baseline model is trained and evaluated with the same inputs as the full model, but does not pass control to a lower-level controller, instead predicting a new action using the planner LSTM at each timestep (i.e., no hierarchical control). Das et al. (2018a) name this baseline *LSTM+Question*.

		Matterport↑ *(%)*		**THOR-Nav↑** *(%)*		**EQA↓** *(m)*
	Model	**Seen**	**Un**	**Seen**	**Un**	**Un**
Pub.	Full Model	27.1	19.6	77.7	18.08	4.17
	Baseline	15.9	16.3	2.18	1.54	4.21
Uni	$\mathcal{A}$	18.5	17.1	4.53	2.88	4.53
	$\mathcal{A}+\mathcal{V}$	21.2	16.6	**35.6**	**7.50**	*__4.11__
	$\mathcal{A}+\mathcal{L}$	**23.0**	*__22.1__	4.03	3.46	4.64
Δ	Uni – Base	**+7.1**	**+5.8**	**+33.4**	**+5.96**	**-0.10**

Table 1: Navigation success (Matterport, THOR-Nav) *(%)* and remaining distance to target (EQA) *(m)*. Best unimodal in bold; better than reported baseline; *better than full model.

Published QA Full Model: Given the last five image encodings along the gold standard navigation trajectory, $\mathcal{V}_{t-4} \ldots \mathcal{V}_t$, and the question encoding $\mathcal{L}$, image-question similarities are calculated via a dot product and converted via attention weights to a summary weight $\bar{\mathcal{V}}$, which is concatenated with $\mathcal{L}$ and used to predict the answer. Das et al. (2018a) name this oracle-navigation model *ShortestPath+VQA*.

QA Baseline: Das et al. (2018a) provide no explicit baseline for the VQA component alone. We use a majority class baseline inspired by the data's entropy based filtering.

4 Experiments

Across all benchmarks, unimodal baselines outperform baseline models used in or derived from the original works. Navigating unseen environments, these unimodal ablations outperform their corresponding full models on the Matterport (absolute ↑ 2.5% success rate) and EQA (↓ 0.06m distance to target).

4.1 Navigation

We evaluate our ablation baselines on Matterport,[3] THOR-Nav, and EQA (Table 1),[4] and discover that some unimodal ablations outperform their corresponding full models. For Matterport and THOR-Nav, success rate is defined by proximity to the target. For EQA, we measure absolute distance from the target in meters.

Unimodal Performance: Across Matterport, THOR-Nav, and EQA, either $\mathcal{A}+\mathcal{V}$ or $\mathcal{A}+\mathcal{L}$

---

[2] This approximates the agent having visited every possible location, interacted with all possible objects, and looked in all possible directions before answering.

[3] We report on Matterport-validation since this allows comparing Seen versus Unseen house performance.

[4] For consistency with THOR-Nav and EQA, we here evaluate Matterport using *teacher forcing*.

		Matterport↑ (%)	
	Model	**Seen**	**Unseen**
Pub.	Full Model	38.6	21.8
	Baseline	15.9	16.3
Uni	$\mathcal{A}$	4.1	3.2
	$\mathcal{A}+\mathcal{V}$	**30.6**	13.3
	$\mathcal{A}+\mathcal{L}$	15.4	**13.9**
Δ	Uni – Base	**+14.7**	-2.4

Table 2: Navigation results for Matterport when trained using student forcing. Best unimodal in bold; better than reported baseline.

		$\mathbf{d_T}\downarrow$ (m)			$\mathbf{d_{min}}\downarrow$ (m)		
	Model	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}
Pub.	Full	0.971	4.17	8.83	0.291	2.43	6.45
	Baseline	1.020	4.21	8.73	0.293	2.45	6.38
Uni	$\mathcal{A}$	***0.893**	4.53	9.56	*0.242	3.16	7.99
	$\mathcal{A}+\mathcal{V}$	*0.951	***4.11**	†**8.83**	*0.287	**2.51**	***6.44**
	$\mathcal{A}+\mathcal{L}$	0.987	4.64	9.51	***0.240**	3.19	7.96
Δ	U – B	**-0.127**	**-0.10**	+0.10	**-0.053**	+0.06	+0.06

Table 3: Final distances to targets ($\mathbf{d_T}$) and minimum distance from target achieved along paths ($\mathbf{d_{min}}$) in EQA navigation. Best unimodal in bold; better than reported baseline; *better than full model; †tied with full model.

achieves better performance than existing baselines. In Matterport, the $\mathcal{A}+\mathcal{L}$ ablation performs *better* than the Full Model in unseen environments. The diverse scenes in this simulator may render the vision signal more noisy than helpful in previously unseen environments. The $\mathcal{A}+\mathcal{V}$ model in THOR-Nav and EQA is able to latch onto dataset biases in scene structure to navigate better than chance (for IQA), and the nonhierarchical baseline (in EQA). In EQA, $\mathcal{A}+\mathcal{V}$ also outperforms the Full Model;[5] the latent information about navigation from questions may be too distant for the model to infer.

The agent with access only to its action history ($\mathcal{A}$) outperforms the baseline agent in Matterport and THOR-Nav environments, suggesting it learns navigation correlations that are not captured by simple random actions (THOR-Nav) or programmatic walks away from the starting position (Matterport). Minimal sensing (which actions are available, W) coupled with the *topological* biases in trajectories (Figure 2), help this nearly zero-input agent outperform existing baselines.[6]

Matterport Teacher vs Student forcing With teacher forcing, at each timestep the navigation agent takes the gold-standard action regardless of what action it predicted, meaning it only sees steps along gold-standard trajectories. This paradigm is used to train the navigation agent in THOR-Nav and EQA. Under student forcing, the agent samples the action to take from its predictions, and loss is computed at each time step against the action that would have put the agent on the shortest path to the goal. Thus, the agent sees more of the scene, but can take more training iterations to learn to move to the goal.

Table 2 gives the highest validation success rates across all epochs achieved in Matterport by models trained using student forcing. The unimodal ablations show that the Full Model, possibly because with more exploration and more training episodes, is better able to align the vision and language signals, enabling generalization in unseen environments that fails with teacher forcing.

EQA Navigation Variants Table 3 gives the average final distance from the target ($\mathbf{d_T}$, used as the metric in Table 1) and average minimum distance from target achieved along the path ($\mathbf{d_{min}}$) during EQA episodes for agents starting 10, 30, and 50 actions away from the target in the EQA navigation task. At 10 actions away, the unimodal ablations tend to outperform the full model on both metrics, possibly due to the shorter length of the episodes (less data to train the joint parameters). The $\mathcal{A}+\mathcal{V}$ ablation performs best among the ablations, and ties with or outperforms the Full Model in all but one setting, suggesting that the EQA Full Model is not taking advantage of language information under any variant.

4.2 Question Answering

We evaluate our ablation baselines on IQUAD V1 and EQA, reporting top-1 QA accuracy (Table 4) given gold standard navigation information as $\mathcal{V}$. These decoupled QA models do not take in a previous action, so we do not consider $\mathcal{A}$ ONLY ablations for this task.

Unimodal Performance: On IQUAD V1, due to randomization in its construction, model ab-

---

[5] EQA full & baseline model performances do not exactly match those in Das et al. (2018a) because we use the expanded data updated by the authors `https://github.com/facebookresearch/EmbodiedQA/`.

[6] This learned agent begins to relate to work in minimally sensing robotics (O'Kane and LaValle, 2006).

	Model	IQUAD V1↑ Unseen	IQUAD V1↑ Seen	EQA ↑ Unseen
Pub.	Full Model	88.3	89.3	64.0
	Baseline	41.7	41.7	19.8
Uni	$\mathcal{V}$ ONLY	**43.5**	**42.8**	44.2
	$\mathcal{L}$ ONLY	41.7	41.7	**48.8**
Δ	Uni – Base	**+1.8**	**+1.1**	**+29.0**

Table 4: Top-1 QA accuracy. Best unimodal in bold; better than reported baseline.

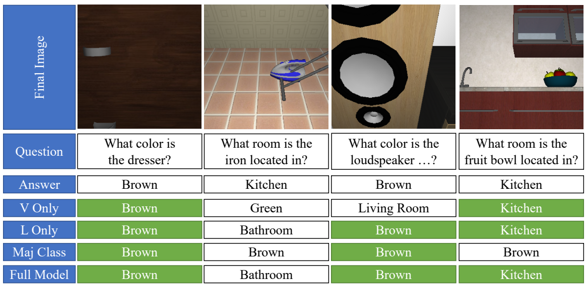
lations perform nearly at chance.[7] The $\mathcal{V}$ ONLY model with access to the locations of all scene objects only improves by 2% over random guessing.

For EQA, single modality models perform significantly better than the majority class baseline. The vision-only model is able to identify salient colors and basic room features that allow it to reduce the likely set of answers to an unknown question. The language only models achieve nearly 50%, suggesting that despite the entropy filtering in Das et al. (2018a) each question has one answer that is as likely as all other answers combined (e.g. 50% of the answers for *What color is the bathtub?* are *grey*, and other examples in Figure 4).

5 Related Work

Historically, semantic parsing was used to map natural language instructions to visual navigation in simulation environments (Chen and Mooney, 2011; MacMahon et al., 2006). Modern approaches use neural architectures to map natural language to the (simulated) world and execute actions (Paxton et al., 2019; Chen et al., 2018; Nguyen et al., 2018; Blukis et al., 2018; Fried et al., 2018; Mei et al., 2016). In visual question answering (VQA) (Antol et al., 2015; Hudson and Manning, 2019) and visual commonsense reasoning (VCR) (Zellers et al., 2019), input images are accompanied with natural language questions. Given the question, egocentric QA requires an agent to navigate and interact with the world to gather the relevant information to answer the question. In both cases, end-to-end neural architectures make progress on these tasks.

For language annotations, task design, difficulty, and annotator pay can introduce unintended artifacts which can be exploited by models to "cheat" on otherwise complex tasks (Glockner et al., 2018; Poliak et al., 2018). Such issues also occur in multimodal data like VQA (Goyal et al., 2018), where models can answer correctly without looking at the image. In image captioning, work has shown competitive models relying only on nearest-neighbor lookups (Devlin et al., 2015) as well as exposed misalignment between caption relevance and text-based metrics (Rohrbach et al., 2018). Our unimodal ablations of visual navigation and QA benchmarks uncover similar biases, which deep architectures are quick to exploit.



Final Image				
Question	What color is the dresser?	What room is the iron located in?	What color is the loudspeaker …?	What room is the fruit bowl located in?
Answer	Brown	Kitchen	Brown	Kitchen
V Only	Brown	Green	Living Room	Kitchen
L Only	Brown	Bathroom	Brown	Kitchen
Maj Class	Brown	Brown	Brown	Brown
Full Model	Brown	Bathroom	Brown	Kitchen

Figure 4: Qualitative results on the EQA task. The language only model can pick out the most likely answer for a question. The vision only model finds salient color and room features, but is unaware of the question.

6 Conclusions

In this work, we introduce an evaluation framework and perform the missing analysis from several new datasets. While new state-of-the-art models are being introduced for several of these domains (e.g., Matterport: (Ma et al., 2019a; Ke et al., 2019; Wang et al., 2019; Ma et al., 2019b; Tan et al., 2019; Fried et al., 2018), and EQA: (Das et al., 2018b)), they lack informative, individual unimodal ablations (i.e., ablating *both* language and vision) of the proposed models.

We find a performance gap between baselines used in or derived from the benchmarks examined in this paper and unimodal models, with unimodal models outperforming those baselines across all benchmarks. These unimodal models can even outperform their multimodal counterparts. In light of this, we recommend all future work include unimodal ablations of proposed multimodal models to vet and highlight their learned representations.

Acknowledgements

This work was supported by NSF IIS-1524371, 1703166, NRI-1637479, IIS-1338054, 1652052, ONR N00014-13-1-0720, and the DARPA CwC program through ARO (W911NF-15-1-0543).

---

[7] Majority class and chance for IQUAD V1 both achieve 50%, 50%, 25% when conditioned on question type; our Baseline model achieves the average of these.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. 2018. Mapping navigation instructions to continuous control actions with position visitation prediction. In *Proceedings of the Conference on Robot Learning*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2018. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *NeurIPS Workshop on Visually Grounded Interaction and Language (ViGIL)*.

Volkan Cirik, Louis-Philippe Morency, and Taylor Berg-Kirkpatrick. 2018. Visual referring expression recognition: What do systems actually learn? In *Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018a. Embodied Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018b. Neural Modular Control for Embodied Question Answering. In *Conference on Robot Learning (CoRL)*.

Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. 2015. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems (NeurIPS)*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proc. of the Association for Computational Linguistics (ACL)*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yash Goyal, Tejas Khot, Aishwarya Agrawal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2018. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision (IJCV)*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: a new dataset for compositional question answering over real-world images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-aware visual-textual co-grounded navigation agent. In *International Conference on Learning Representations (ICLR)*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *Proc. of the Association for Computational Linguistics (ACL)*.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. 2018. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jason M O'Kane and Steven M. LaValle. 2006. On comparing the power of mobile robots. In *Robotics: Science and Systems (RSS)*.

Chris Paxton, Yonatan Bisk, Jesse Thomason, Arunkumar Byravan, and Dieter Fox. 2019. Prospection: Interpretable plans from language by predicting the future. In *International Conference on Robotics and Automation (ICRA)*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. In *Joint Conference on Lexical and Computational Semantics (StarSem)*.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

ExCL: Extractive Clip Localization Using Natural Language Descriptions

Soham Ghosh[1,*] **Anuva Agarwal**[1,*] **Zarana Parekh**[1,*] **Alexander Hauptmann**[1]

[1]Language Technologies Institute
Carnegie Mellon University

{sohamg,anuvaa,zpp,alex}@cs.cmu.edu

Abstract

The task of retrieving clips within videos based on a given natural language query requires cross-modal reasoning over multiple frames. Prior approaches such as sliding window classifiers are inefficient, while text-clip similarity driven ranking-based approaches such as segment proposal networks are far more complicated. In order to select the most relevant video clip corresponding to the given text description, we propose a novel extractive approach that predicts the start and end frames by leveraging cross-modal interactions between the text and video - this removes the need to retrieve and re-rank multiple proposal segments. Using recurrent networks we encode the two modalities into a joint representation which is then used in different variants of start-end frame predictor networks. Through extensive experimentation and ablative analysis, we demonstrate that our simple and elegant approach significantly outperforms state of the art on two datasets and has comparable performance on a third.

1 Introduction

Clip Localization is the task of selecting the relevant span of temporal frames in a video corresponding to a natural language description and has recently piqued interest in research that lies at the intersection of visual and textual modalities. An example of this task is demonstrated in Figure 1. It requires cross-modal reasoning to ground free-form text inside the video and calls for models capable of segmenting a video into action segments (Singh et al., 2016; Yeung et al., 2016; Xu et al., 2017) as well as measuring multi-modal semantic similarity (Karpathy and Fei-Fei, 2015).

This task is inherently discriminative, i.e., there is only a single most relevant clip pertaining to a given query in the corresponding video. However, most prior works (Hendricks et al., 2017, 2018; Liu et al., 2018; Chen et al., 2018; Zhang et al., 2018) explore this as a ranking task over a fixed number of moments by uniformly sampling clips within a video. Moreover, these approaches are restrictive in scope since they use predefined clips as candidates for a video and cannot be easily extended to videos with considerable variance in length.

Figure 1: Clip Extraction task for the given query '*the biker jumps to another ramp near the camera*'

Gao et al. (2017); Xu et al. (2019) apply two-stage methods which rank candidate clips using a learned similarity metric. Gao et al. (2017); Ge et al. (2019) propose a sliding window approach with alignment and offset regression learning objective, but it is limited by the coarseness of the windows and is thus inefficient and inflexible. Xu et al. (2019) address this through a query-guided segment proposal network (QSPN). However, the similarity metric used by these approaches is difficult to learn as it is sensitive to the choice of negative samples (Yu et al., 2018) and it still does not consider the discriminative nature of the task.

Hence, we propose an elegant and fairly simple extractive approach. Our technique is similar to text-based Machine Comprehension (Chen et al., 2017) but in a multimodal setting where the video is analogous to the text passage and the target-clip is analogous to the text span corresponding to the correct answer. We verify empirically that

---

[*] Equal contribution, randomly ordered.

Proceedings of NAACL-HLT 2019, pages 1984–1990
Minneapolis, Minnesota, June 2 - June 7, 2019.

our method significantly outperforms prior work on two benchmark datasets - TACoS, ActivityNet and comparably well on the third, Charades-STA. Our flexible, modular approach to *Extractive Clip Localization (ExCL)* can easily be extended to incorporate attention models and different variants of encoders for both visual and text modality to improve performance further.

2 Approach

Our model comprises of three modular parts - a text encoder, a video encoder and a span predictor as shown in Figure 2. We use a bidirectional text LSTM as the text encoder with pretrained GloVE (Pennington et al., 2014) embeddings as input, and use the last hidden state ($\mathbf{h}^T$) as sentence-embedding. Simpler variants such as bag-of-words, and other approaches such as InferSent (Conneau et al., 2017) or Skip-Thought (Kiros et al., 2015) could also be used. We use a bidirectional LSTM with I3D features (Carreira and Zisserman, 2017) as the video encoder which captures temporal context at each time-step ($\mathbf{h}_t^V$). Note that this could also be substituted by C3D features (Tran et al., 2014) or self-attentive networks (Kim et al., 2018). Finally, we compare the variants of span-predictor networks which output scores $S_{\text{start}}(t), S_{\text{end}}(t) \in \mathbb{R}^1$ respectively.

2.1 Training objectives

We consider two modes of training our networks, one which uses a classification loss (ExCL-clf) and another which uses a regression loss (ExCL-reg).

ExCL-clf: The scores are normalized using SoftMax to give $P_{\text{start}}(t), P_{\text{end}}(t)$, and trained using negative log-likelihood loss:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N} \log(P_{\text{start}}(t_s^i)) + \log(P_{\text{end}}(t_e^i)) \tag{1}$$

where N is number of text-clip pairs and t_s^i, t_e^i are the ground-truth start and end frame indices for the i^{th} pair. During inference we predict the span of frames $(\hat{t}_s, \hat{t}_e)$ for each query by maximizing the joint probability of start and end frames.

$$\text{span}(\hat{t}_s, \hat{t}_e) = \underset{\hat{t}_s,\hat{t}_e}{\text{argmax}}\, P_{\text{start}}(\hat{t}_s)P_{\text{end}}(\hat{t}_e) \tag{2}$$

$$= \underset{\hat{t}_s,\hat{t}_e}{\text{argmax}}\, S_{\text{start}}(\hat{t}_s) + S_{\text{end}}(\hat{t}_e) \tag{3}$$

$$\text{s.t. } \hat{t}_s \leq \hat{t}_e \tag{4}$$

ExCL-reg: The classification approach is limited to predicting start and end times of clip at discretized intervals, while the true target is a continuous value. In order to directly model this as a regression problem, we formulate start and end time prediction by computing an expectation over the probability distribution given by SoftMax outputs:

$$t_s = \mathbb{E}_{P_{\text{start}}}[t] \tag{5}$$

$$= \sum_{t_s=1}^{T} t_s P_{\text{start}}(t_s) \tag{6}$$

$$t_e = \mathbb{E}_{P_{\text{start}}}[\mathbb{E}_{P_{\text{end|start}}}[t]] \tag{7}$$

$$= \sum_{t_s=1}^{T} P_{\text{start}}(t_s) \sum_{t_e=1}^{T} P_{\text{end|start}}(t_e) \tag{8}$$

Here the above equations t_s, t_e refer to the actual time values corresponding to each index. $P_{\text{end|start}}(t_e)$ is computed by a SoftMax over masked logits:

$$P_{\text{end|start}} = \text{SoftMax}(\mathbb{1}[t_e \geq t_s]S_{\text{end}}(t))$$

Finally we train the networks using regression losses such as mean squared error and absolute error. We find that using absolute error and first normalizing the values of time to $t_s, t_e \in [0, 1]$ yields better results.

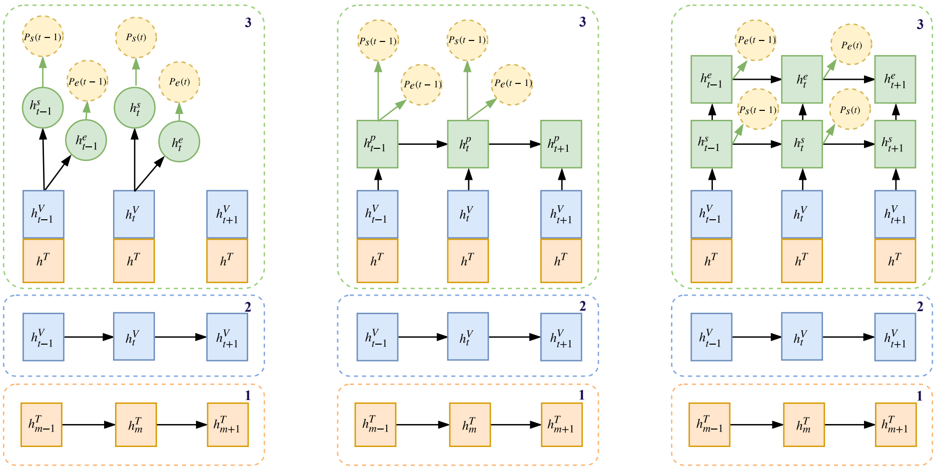
2.2 Span Predictor Variants

We implement the following variants of the span predictor network:

MLP predictor: At each time step t, we pass concatenated video-encoder features with sentence embeddings into two multi-layered perceptrons (MLPs) to obtain scores $S_{\text{start}}(t), S_{\text{end}}(t)$.

$$S_{\text{start}}(t) = \text{MLP}_{\text{start}}([\mathbf{h}_t^V; \mathbf{h}^T]) \tag{9}$$

$$S_{\text{end}}(t) = \text{MLP}_{\text{end}}([\mathbf{h}_t^V; \mathbf{h}^T]) \tag{10}$$



(a) MLP predictor (b) Tied-LSTM predictor (c) Conditioned-LSTM predictor

Figure 2: Our model consists of three modules: a text sentence encoder (orange, denoted by [1]), a video encoder (blue, denoted by [2]) and three variants of span predictor (green, denoted by [3]) - MLP, Tied-LSTM and Conditioned-LSTM to predict start and end probabilities for each frame. We use bidirectional LSTMs for the text and video encoders.

Tied LSTM predictor: Here, we send concatenated video encoder output and sentence embedding to a bidirectional LSTM as input at each step in order to capture recurrent cross-modal interactions. The hidden states ($\mathbf{h}_t^P$) concatenated with the original inputs are then fed to a MLP with $\tanh$ activation in hidden layers to predict start and end scores.

$$h_t^P = \text{LSTM}([\mathbf{h}_t^V;\mathbf{h}^T], \mathbf{h}_{t-1}^P) \tag{11}$$

$$S_{\text{start}}(t) = \text{MLP}_{\text{start}}([\mathbf{h}_t^P;\mathbf{h}_t^V;\mathbf{h}^T]) \tag{12}$$

$$S_{\text{end}}(t) = \text{MLP}_{\text{end}}([\mathbf{h}_t^P;\mathbf{h}_t^V;\mathbf{h}^T]) \tag{13}$$

Conditioned LSTM predictor: Note that in the previous two approaches the end-frame predictor is not conditioned in any way on the start predictor. Here we use two bidirectional LSTMs: LSTM_{end} takes as input the hidden states $\mathbf{h}_t^{P_0}$ of $\text{LSTM}_{\text{start}}$ and produces $\mathbf{h}_t^{P_1}$ as output. The respective hidden states are then used in a similar way as the tied LSTM method to generate start and end scores.

$$\mathbf{h}_t^{P_0} = \text{LSTM}_{\text{start}}([\mathbf{h}_t^V;\mathbf{h}^T], \mathbf{h}_{t-1}^{P_0}) \tag{14}$$

$$\mathbf{h}_t^{P_1} = \text{LSTM}_{\text{end}}(\mathbf{h}_t^{P_0}, \mathbf{h}_{t-1}^{P_1}) \tag{15}$$

$$S_{\text{start}}(t) = \mathbf{W}_s([\mathbf{h}_t^{P_0};\mathbf{h}_t^V;\mathbf{h}^T]) + \mathbf{b}_s \tag{16}$$

$$S_{\text{end}}(t) = \mathbf{W}_e([\mathbf{h}_t^{P_1};\mathbf{h}_t^V;\mathbf{h}^T]) + \mathbf{b}_e \tag{17}$$

3 Datasets

We evaluate our models on three datasets. Note that we do not evaluate our models on DiDeMo/TEMPO (Hendricks et al., 2017, 2018) as these datasets only provide coarse fixed-size moments, thus reducing the problem essentially to ranking a fixed set of candidates. We choose the following datasets because each has unique properties such as visual variance, richness of vocabulary and variance in query lengths.

MPII TACoS: This dataset (Rohrbach et al., 2014) has been built on top of the MPII Cooking Activities dataset. It consists of detailed temporally aligned text descriptions of cooking activities. The average length of videos is 5 minutes. A significant challenge in TACoS dataset is that descriptions span over only a few seconds because of the atomic nature of queries such as 'takes out the knife' and 'chops the onion' (8.4% of them are less than 1.6s long). Such short queries allow a smaller margin of error. We use the train/test split as provided by Gao et al. (2017) *here*. Coupled with lesser visual variance to distinguish activities, fine-grained actions and descriptions which can often have high word overlap, this is a challenging dataset.

ActivityNet Captions: ActivityNet (Caba Heilbron et al., 2015) is a large-scale open domain activity recognition, segmentation and prediction dataset based on YouTube videos additionally augmented with dense temporally annotated captions (Krishna et al., 2017). The average length is 2 minutes, but they are much more diverse in content, with videos that span over 200 activity classes and annotations which use a richer vocabulary. There are 10,024 and 5,044 train and test set videos. We use the same train/test split as provided by the authors. Our reported results have 3,370 missing videos which could not be downloaded.

Charades-STA: The Charades dataset was introduced with action classification, localization and video description annotations (Sigurdsson et al., 2016). Gao et al. (2017) extend this to include sentence level temporal annotation to create the Charades-STA dataset which contains 12,408 training, and 3,720 test sentence level annotations. In Charades-STA, average length of videos is 30 seconds and query length (number of frames) has much lower variance (3.7s) as compared to TACoS (39.5s) and ActivityNet (78.1s).

4 Experiments

4.1 Feature Extraction

We downsample the videos at a frame rate of 5 frames per second and extract I3D RGB (Carreira and Zisserman, 2017) visual features pretrained on Kinetics dataset. We use a fine-tuned version for Charades available *here*.

4.2 Implementation Details

We use pre-trained GloVE embeddings of 300 dimensions. The extracted visual features have 1024 dimensions. For TACoS, Charades-STA, and ActivityNet we use a vocabulary size of 1438, 3720 and 10,000 respectively. We considered batch sizes of 16, 32, 64 and single-layer LSTM hidden sizes of 128, 256, 512. The video and span predictors are bidirectional LSTMs with 256 and 128 hidden units respectively while queries are encoded by a 256-dimensional bidirectional LSTM. The MLP based span predictors have 256 hidden dimensions and use $\tanh$ activation in hidden layers. For all datasets, we train the model with a batch size of 32 for 30 epochs using Adam optimizer with a learning rate of 0.001 and early stopping. We apply a dropout of 0.5 to all the abovementioned LSTMs during training. We measure our model performance primarily using *localization accuracy* which is defined to be intersection over union (IoU) at threshold values of 0.3, 0.5, 0.7 to compare with past work which reports Recall@1, IoU=$\{0.3, 0.5, 0.7\}$.

4.3 Experimentation and Ablative Analysis

We compare the different training objectives (labeled ExCL-clf for classification and ExCL-reg for regression) and evaluate the usefulness of recurrent encoders for video representations by removing the video LSTM (labeled ExCL-clf/reg 1-$\{a, b, c\}$). We also perform ablative analysis of a range of span predictor networks. We compare the performance of our proposed model with three baselines which are the current SOTA for the different datsets. (i) Activity Concepts based Localizer (ACL) proposed in Ge et al. (2019) for TACoS (ii) Segment Proposal Network approach proposed in (Xu et al., 2019) for ActivityNet and (iii) Moment Alignment Network (MAN) via Iterative Graph Adjustment approach proposed in (Zhang et al., 2018) for Charades-STA. While we significantly beat the first two baselines for TACoS and Activity Net respectively, we attain comparable performance with the third for Charades-STA. It is to be noted that since the approach in (Zhang et al., 2018) depends on ranking fixed number of segments in each video, it is not scalable to the other two datasets which have longer videos with greater variance in their lengths.

4.4 Results

Our results in Table 1 confirm our hypothesis that extractive models work better.

We find that across all datasets, models without recurrent architectures (ExCL 1-a) perform significantly worse, demonstrating the importance of temporal context provided by LSTMs in either video encoder (ExCL 2-a) or span predictors (ExCL 1-b). With ExCL 2-a we see that if the visual context is captured well using recurrent architectures in the video LSTM, then even with a MLP span predictor we get a significant improvement in performance. Furthermore, if we add LSTM span predictors along with the video LSTM (ExCL 2-$\{b, c\}$) we obtain an additional boost in performance. However, without a recurrent visual encoder, a recurrent span predictor is essential to capture both uni-modal and cross-modal interactions (ExCL 1-$\{b,\ c\}$). When comparing the regression learning objective with classification, we do not notice a substantial gain in performance thereby indicating that not much information is lost if the continuous nature of the labels is not considered. In terms of span predictors, tied LSTM generally performs well across all datasets,

	TACoS			Charades-STA			ActivityNet		
IoU	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7
Ge et al. (2019)	24.2	20.0	–	–	30.5	12.2	–	–	–
Xu et al. (2019)	–	–	–	54.7	35.6	15.8	45.3	27.7	13.6
Zhang et al. (2018)	–	–	–	–	46.5	22.7	–	–	–
ExCL-clf 1-a	22.6	12.6	5.1	55.4	30.4	14.8	42.5	23.8	12.1
ExCL-clf 1-b	42.0	25.0	12.3	64.7	43.8	22.1	61.7	40.4	23.0
ExCL-clf 1-c	41.9	25.5	13.6	64.2	43.9	**23.3**	60.7	40.9	23.4
ExCL-clf 2-a	41.7	26.0	12.9	64.6	41.5	20.3	60.4	40.5	23.1
ExCL-clf 2-b	44.2	**28.0**	**14.6**	**65.1**	**44.1**	22.6	61.1	41.3	23.4
ExCL-clf 2-c	**44.4**	27.8	**14.6**	61.4	41.2	21.3	**62.1**	**41.6**	**23.9**
ExCL-reg 1-a	26.2	11.9	4.8	54.7	34.0	14.5	48.4	27.0	11.0
ExCL-reg 1-b	45.2	27.5	12.9	60.1	42.6	21.6	**63.0**	**43.6**	23.6
ExCL-reg 1-c	41.4	24.8	11.4	59.0	43.1	20.7	61.5	42.7	23.4
ExCL-reg 2-a	42.2	27.2	11.7	59.6	41.9	20.2	61.5	41.9	23.3
ExCL-reg 2-b	**45.5**	**28.0**	**13.8**	**61.5**	**44.1**	**22.4**	62.3	42.7	**24.1**
ExCL-reg 2-c	42.3	27.3	12.5	58.0	41.8	20.5	61.4	41.7	22.4

Table 1: Clip Localization Accuracy at IoU = $\{0.3, 0.5, 0.7\}$ for TACoS, Charades-STA and ActivityNet. Here ExCL-clf represents the classification loss model and ExCL-reg represents the regression loss model. ExCL-$\{$clf/reg$\}$ 1-m refer to models run without a video LSTM encoder, while ExCL-$\{$clf/reg$\}$ 2-m include the video LSTM. $m = a, b, c$ refer to MLP, tied LSTM and conditioned LSTM span predictor networks respectively.

and any difference with conditioned LSTM is negligible. This benefit is more pronounced when using the regression objective, possibly because conditioning is already captured in the formulation as given by Equation 7.

While Xu et al. (2019) note a significant difference in performance between Charades-STA and ActivityNet captions, performance is similar on both the datasets in our model. We hypothesize that their model fails to work well when there is large variability in the query lengths, as explained in Section 3. We also find TACoS to be a significantly more challenging benchmark, similar to Ge et al. (2019).

5 Conclusion

In conclusion, our main contribution is an extractive model for clip localization based on text queries as opposed to ranking driven approaches used in the past. Our results show that this elegant and fairly simple approach works much better empirically on three very different benchmark datasets, with tied LSTM span predictor generally giving best results. It is to be noted that these datasets previously had three different architectures as their respective SOTA, and our work naturally lays the foundation for training a single generalizable model across datasets and possibly related tasks as the next step. Furthermore, our approach is modular, making it trivial to insert different architectures for the encoders and span-predictors. Other future directions of this work include adding temporal attention in order to handle more complicated temporal references and extending this approach to work for longer, and thereby more challenging videos such as movies.

Acknowledgements

The authors would also like to thank anonymous reviewers for their comments and Junwei Liang for helpful discussions. This research was supported in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program. This work is also supported in part through the financial assistance award 60NANB17D156 from U.S. Department of Commerce, National Institute of Standards and Technology and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00340.

References

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 961–970.

Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. *arXiv preprint arXiv:1705.02101*.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 245–253. IEEE.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5803–5812.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2018. Localizing moments in video with temporal language. *arXiv preprint arXiv:1809.01337*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Kyung-Min Kim, Seong-Ho Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. 2018. Multimodal dual attention memory for video story question answering. *Lecture Notes in Computer Science*, page 698713.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *ICCV*, pages 706–715.

Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2018. Temporal modular networks for retrieving complex compositional activities in videos. In *European Conference on Computer Vision*, pages 569–586. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*, pages 184–195. Springer.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer.

Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. 2016. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1970.

Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2014. C3D: generic features for video analysis. *CoRR*, abs/1412.0767.

Huijuan Xu, Abir Das, and Kate Saenko. 2017. R-c3d: region convolutional 3d network for temporal activity detection. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 5794–5803.

Huijuan Xu, Kun He, L Sigal, S Sclaroff, and K Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*, volume 2, page 7.

Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. 2016. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687.

Baosheng Yu, Tongliang Liu, Mingming Gong, Changxing Ding, and Dacheng Tao. 2018. Correcting the triplet selection bias for triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 71–87.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2018. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. *arXiv preprint arXiv:1812.00087*.

Detecting Dementia in Mandarin Chinese using Transfer Learning from a Parallel Corpus

Bai Li[1,2], **Yi-Te Hsu**[1,2,4], **Frank Rudzicz**[1,2,3]
[1] University of Toronto, Toronto, Canada
[2] Vector Institute, Toronto, Canada
[3] Toronto Rehabilitation Institute, Toronto, Canada
[4] Academia Sinica, Taipei, Taiwan
`{bai, eeder, frank}@cs.toronto.edu`

Abstract

Machine learning has shown promise for automatic detection of Alzheimer's disease (AD) through speech; however, efforts are hampered by a scarcity of data, especially in languages other than English. We propose a method to learn a correspondence between independently engineered lexicosyntactic features in two languages, using a large parallel corpus of out-of-domain movie dialogue data. We apply it to dementia detection in Mandarin Chinese, and demonstrate that our method outperforms both unilingual and machine translation-based baselines. This appears to be the first study that transfers feature domains in detecting cognitive decline.

1 Introduction

Alzheimer's disease (AD) is a neurodegenerative disease affecting 5.7 million people in the US (Association et al., 2018), and is the most common cause of dementia. Although no cure yet exists, early detection of AD is crucial for an effective treatment to delay or prepare for its effects (Dubois et al., 2016). One of the earliest symptoms of AD is speech impairment, including a difficulty in finding words and changes to grammatical structure (Taler and Phillips, 2008). These early signs can be detected by having the patient perform a picture description task, such as the *Cookie Theft* task from the Boston Diagnostic Aphasia Examination (Goodglass and Kaplan, 1983).

Previous models have applied machine learning to automatic detection of AD, for example, Fraser et al. (2016) extracted a wide variety of lexicosyntactic and acoustic features to classify AD and obtained 82% accuracy on the DementiaBank (DB) dataset. However, clinical studies of AD are expensive, so datasets of patient data are often scarce. Noorian et al. (2017) augmented DB with more a much larger corpus of normative data and improved the classification accuracy to 93% on DB. Similar linguistic differences between healthy and AD speech have been observed in Mandarin Chinese (Lai et al., 2009), but machine learning has not yet been applied to detecting AD in Mandarin.

Daume III (2007) proposed a simple way of combining features in different domains, assuming that the same features are extracted in each domain. In our case, ensuring consistency of features across domains is challenging because of the grammatical differences between Mandarin and English. For example, Mandarin doesn't have determiners or verb tenses, and has classifiers, which don't exist in English (Chao, 1965). Another method trains a classifier jointly on multiple domains with different features on each domain, by learning a projection to a common subspace (Duan et al., 2012). However, this method only accepts labelled samples in each domain, and cannot make use of unlabelled, out-of-domain data. Other work from our broader group (Fraser et al., 2019) combined English and French data by extracting features based on conceptual "information units" rather than words, thus limiting the effects of multilingual differences.

In the current work, we train an *unsupervised* model to detect dementia in Mandarin, requiring only the English DB dataset and a large parallel Mandarin-English corpus of normative dialogue. We extract lexicosyntactic features in Mandarin and English using separate pipelines, and use the OpenSubtitles corpus of bilingual parallel movie dialogues to learn a correspondence between the different feature sets. We combine this correspondence model with a classifier trained on DB to predict dementia on Mandarin speech. To evaluate our system, we apply it to a dataset of speech from Mandarin-speakers with dementia, and demon-

Proceedings of NAACL-HLT 2019, pages 1991–1997
Minneapolis, Minnesota, June 2 - June 7, 2019.

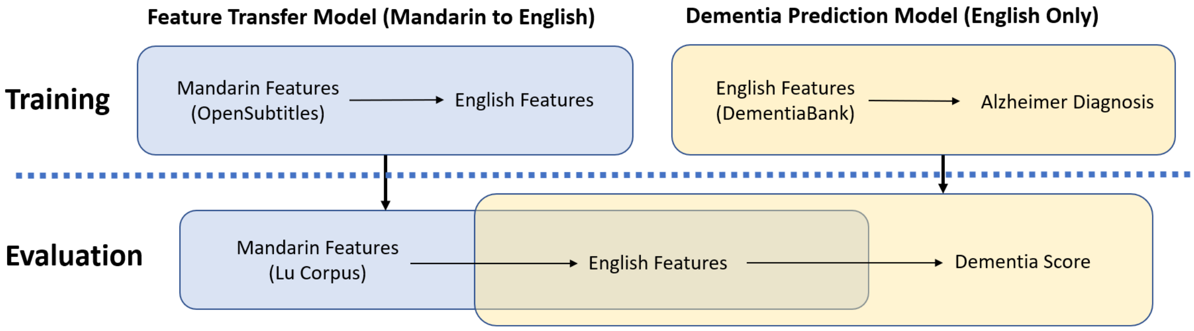


Figure 1: Diagram of our model. We train two separate models: the first is trained on OpenSubtitles and learns to map Mandarin features to English features; the second is trained on DementiaBank and predicts dementia given English features. During evaluation, the two models are combined to predict dementia in Mandarin.

strate that our method outperforms several baselines.

2 Datasets

We use the following datasets:

- **DementiaBank** (Boller and Becker, 2005): a corpus of *Cookie Theft* picture descriptions, containing 241 narrations from healthy controls and 310 from patients with dementia. Each narration is professionally transcribed and labelled with part-of-speech tags. In this work, we only use the narration transcripts, and neither the part-of-speech tags or raw acoustics.

- **Lu Corpus** (MacWhinney et al., 2011): contains 49 patients performing the *Cookie theft* picture description, category fluency, and picture naming tasks in Taiwanese Mandarin. The picture description narrations were human-transcribed; patient diagnoses are unspecified but exhibit various degrees of dementia.

- **OpenSubtitles2016** (Lison and Tiedemann, 2016): a corpus of parallel dialogues extracted from movie subtitles in various languages. We use the Traditional Chinese / English language pair, which contains 3.3 million lines of dialogue.

The Lu Corpus is missing specifics of diagnosis, so we derive a *dementia score* for each patient using the category fluency and picture naming tasks. For each category fluency task, we count the number of unique items named; for the picture naming tasks, we score the number of pictures correctly named, awarding partial credit if a hint was given. We apply PCA to the scores across all tasks, and assign the first principal component to be the *dementia score* for each patient. This gives a relative ordering of all patients for degree of dementia, which we treat as the ground-truth for evaluating our models.

3 Methodology

3.1 Feature Extraction

We extract a variety of lexicosyntactic features in Mandarin and English, including type-token-ratio, the number of words per sentence, and proportions of various part-of-speech tags[1]. A detailed description of the features is provided in the supplementary materials (Section A.1). In total, we extract 143 features in Mandarin and 185 in English. To reduce sparsity, we remove features in both languages that are constant for more than half of the dataset.

Due to the size of the OpenSubtitles corpus, it was computationally unfeasible to run feature extraction on the entire corpus. Therefore, we randomly select 50,000 narrations from the corpus, where each narration consists of between 1 to 50 contiguous lines of dialogue (about the length of a *Cookie Theft* narration).

For English, we train a logistic regression classifier to classify between dementia and healthy controls on DB, using our features as input. Using L1 regularization and 5-fold CV, our model achieves 77% classification accuracy on DB. This is slightly lower than the 82% accuracy reported

---

[1] The feature extraction pipeline is open-source, available at: `https://github.com/SPOClab-ca/COVFEFE`. The `lex` and `lex_chinese` pipelines were used for English and Chinese, respectively.

by Fraser et al. (2016), but it does not include any acoustic features as input.

3.2 Feature Transfer

Next, we use the OpenSubtitles corpus to train a model to transform Mandarin feature vectors to English feature vectors. For each target English feature, we train a separate ElasticNet linear regression (Zou and Hastie, 2005), using the Mandarin features of the parallel text as input. We perform a hyperparameter search independently for each target feature, using 3-fold CV to minimize the MSE.

3.3 Regularization

Although the output of the ElasticNet regressions may be given directly to the logistic regression model to predict dementia, this method has two limitations. First, the model considers each target feature separately and cannot take advantage of correlations between target features. Second, it treats all target feature equally, even though some are noisier than others. We introduce two regularization mechanisms to address these drawbacks: reduced rank regression and joint feature selection.

Reduced Rank Regression

Reduced rank regression (RRR) trains a single linear model to predict all the target features: it minimizes the sum of MSE across all target features, with the constraint that the rank of the linear mapping is bounded by some given R (Izenman, 1975). Following recommended procedures (Davies, 1982), we standardize the target features and find the best value of R with cross validation. However, this procedure did not significantly improve results so it was not included in our best model.

Joint Feature Selection

A limitation of the above models is that they are not robust to noisy features. For example, if some English feature is useful for predicting dementia, but cannot be accurately predicted using the Mandarin features, then including this feature might hurt the overall performance. A desirable English feature in our pipeline needs to not only be useful for predicting dementia in English, but also be reconstructable from Mandarin features.

We modify our pipeline as follows. After training the ElasticNet regressions, we sort the target features by their R^2 (coefficient of determination) measured on the training set, where higher values indicate a better fit. Then, for each K between 1 and the number of features, we select only the top K features and re-train the DB classifier (3.1) to only use those features as input. The result of this experiment is shown in Figure 2.

4 Experiments

4.1 Baseline Models

We compare our system against two simple baselines:

1. **Unilingual baseline:** using the Mandarin features, we train a linear regression to predict the dementia score. We take the mean across 5 cross-validation folds.
2. **Translate baseline:** The other intuitive way to generate English features from a Mandarin corpus is by using translation. We use Google Translate[2] to translate each Mandarin transcript to English. Then, we extract features from the translated English text and feed them to the dementia classifier described in section 3.1.

4.2 Evaluation Metric

We evaluate each model by comparing the Spearman's rank-order correlation ρ (Spearman, 1904) between the ground truth dementia scores and the model's predictions. This measures the model's ability to rank the patients from the highest to the lowest severities of dementia, without requiring a threshold value.

4.3 Experimental Results

Model	**Spearman ρ**
Baselines	
Unilingual	0.385
Google Translate	0.366
Our models	
Feature Transfer	0.319
+ RRR	0.354
+ JFS	**0.549**

Table 1: Baselines compared with our models, evaluated on the Lu corpus. RRR: Reduced rank regression (3.3), JFS: Joint feature selection (3.3).

---

[2] `https://translate.google.com/`

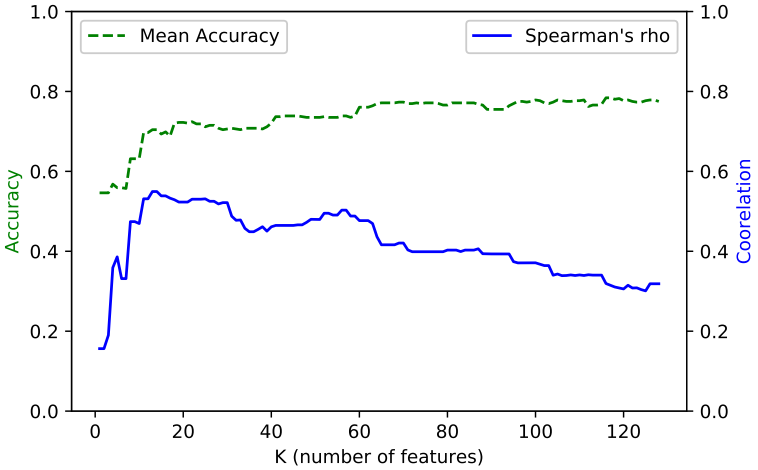


Figure 2: Accuracy of DementiaBank classifier model and Spearman's ρ on Lu corpus, using only the top K English features ordered by R^2 on the OpenSubtitles corpus. Spearman's ρ is maximized at $K = 13$, achieving a score of $\rho = 0.549$. DementiaBank accuracy generally increases with more features.

Our best model achieves a Spearman's ρ of 0.549, beating the translate baseline (n = 49, p = 0.06). Joint feature selection appears to be crucial, since the model performs worse than the baselines if we use all of the features. This is the case no matter if we predict each target feature independently or all at once with reduced rank regression. RRR does not outperform the baseline model, probably because it fails to account for the noisy target features in the correspondence model and considers each feature equally important. We did not attempt to use joint feature selection and RRR at the same time, because the multiplicative combination of hyperparameters K and R would produce a multiple comparisons problem using the small validation set.

Using joint feature selection, we find that the best score is achieved when we use $K = 13$ target features (Figure 2). With $K < 13$, performance suffers because the DementiaBank classifier is not given enough information to make accurate classifications. With $K > 13$, the accuracy for the DementiaBank classifier improves; however, the overall performance degrades because it is given noisy features with low R^2 coefficients. A list of the top features is given in Table 2 in the supplementary materials.

In our experiments, the correspondence model worked better when absolute counts were used for the Chinese CFG features (e.g., the number of $NP \rightarrow PN$ productions in the narration) rather than ratio features (e.g., the proportion of CFG productions that were $NP \rightarrow PN$). When ratios were used for source features, the R^2 coefficients for many target features decreased. A possible explanation is that the narrations have varying lengths, and dividing features by the length introduces a nonlinearity that adversely affects our linear models. However, more experimentation is required to examine this hypothesis.

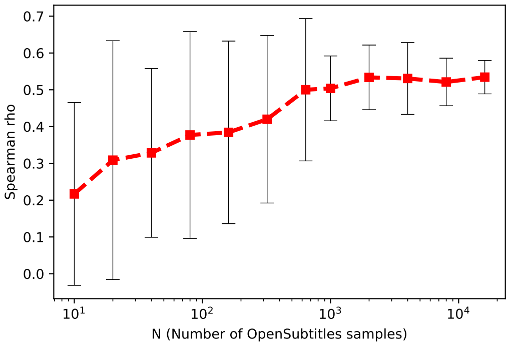


Figure 3: Ablation experiment where a various number of OpenSubtitles samples were used for training. The error bars indicate the two standard deviation confidence interval.

4.4 Ablation Study

Next, we investigate how many parallel OpenSubtitles narrations were necessary to learn the correspondence model. We choose various training sample sizes from 10 to 50,000 and, for each training size, we train and evaluate the whole model from end-to-end 10 times with different random seeds (Figure 3). As expected, the Spearman's ρ increased as more samples were used, but only 1000-2000 samples were required to achieve comparable performance to the full model.

5 Conclusion

We propose a novel method to use a large parallel corpus to learn mappings between engineered features in two languages. Combined with a dementia classifier model for English speech, we constructed a model to predict dementia in Mandarin Chinese. Our method achieves state-of-the-art results for this task and beats baselines based on unilingual models and Google Translate. It is successful despite the stark differences between English and Mandarin, and the fact that the parallel corpus is out-of-domain for the task. Lastly, our method does not require any Mandarin data for training, which is important given the difficulty of acquiring sensitive clinical data.

Future work will investigate the use of automatic speech recognition to reduce the need for manual transcripts, which are impractical in a clinical setting. Also, our model only uses lexicosyntactic features, and ignores acoustic features (e.g., pause duration) which are significant for dementia detection in English. Finally, it remains to apply this method to other languages, such as French (Fraser et al., 2019), for which datasets have recently been collected.

Acknowledgements

We thank Kathleen Fraser and Nicklas Linz for their helpful comments and earlier collaboration which inspired this project.

References

Alzheimer's Association et al. 2018. 2018 Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 14(3):367–429.

Francois Boller and James Becker. 2005. Dementiabank database guide. *University of Pittsburgh*.

Yuen Ren Chao. 1965. *A grammar of spoken Chinese*. Univ of California Press.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263.

PT Davies. 1982. Procedures for reduced-rank regression. *Applied Statistics*, pages 244–255.

Lixin Duan, Dong Xu, and Ivor W Tsang. 2012. Learning with augmented features for heterogeneous domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, pages 667–674. Omnipress.

Bruno Dubois, Harald Hampel, Howard H Feldman, Philip Scheltens, Paul Aisen, Sandrine Andrieu, Hovagim Bakardjian, Habib Benali, Lars Bertram, Kaj Blennow, et al. 2016. Preclinical Alzheimer's disease: definition, natural history, and diagnostic criteria. *Alzheimer's & Dementia*, 12(3):292–323.

Kathleen C. Fraser, Nicklas Linz, Bai Li, Kristina Lundholm Fors, Frank Rudzicz, Alexandra Konig, Jan Alexandersson, Philippe Robert, and Dimitrios Kokkinakis. 2019. Multilingual prediction of Alzheimers disease through domain adaptation and concept-based language modelling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. 2016. Linguistic features identify Alzheimers disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

Harold Goodglass and Edith Kaplan. 1983. *Boston diagnostic examination for aphasia*, 2nd edition. Lea and Febiger, Philadelphia, Pennsylvania.

Alan Julian Izenman. 1975. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Yi-hsiu Lai, Hsiu-hua Pai, et al. 2009. To be semantically-impaired or to be syntactically-impaired: Linguistic patterns in Chinese-speaking persons with or without dementia. *Journal of Neurolinguistics*, 22(5):465–475.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 439–446. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International journal of corpus linguistics*, 15(4):474–496.

Brian MacWhinney, Davida Fromm, Margaret Forbes, and Audrey Holland. 2011. AphasiaBank: Methods for studying discourse. *Aphasiology*, 25(11):1286–1307.

Zeinab Noorian, Chloé Pou-Prom, and Frank Rudzicz. 2017. On the importance of normative data in speech-based assessment. In *Proceedings of Machine Learning for Health Care Workshop (NIPS MLHC)*.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. Luminosoinsight/wordfreq: v2.2.

Vanessa Taler and Natalie A Phillips. 2008. Language performance in Alzheimer's disease and mild cognitive impairment: a comparative review. *Journal of clinical and experimental neuropsychology*, 30(5):501–556.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

A Appendices

A.1 Description of Lexicosyntactic Features

We extract 185 lexicosyntactic features in English and 143 in Mandarin Chinese. We use Stanford CoreNLP to do constituency parsing and part-of-speech tagging (Klein and Manning, 2003; Levy and Manning, 2003). We also use `wordfreq` (Speer et al., 2018) for word frequency statistics in both languages. Our features are similar to the set of features used by Fraser et al. (2016), which the reader can refer to for a more thorough description.

The following features are extracted in English:

- **Narrative length:** Number of words and sentences in narration.
- **Vocabulary richness:** Type-token ratio, moving average type-token ratio (with window sizes of 10, 20, 30, 40, and 50 words), Honoré's statistic, and Brunét's index.
- **Frequency metrics:** Mean word frequencies for all words, nouns, and verbs.
- **POS counts:** Counts and ratios of nouns, verbs, inflected verbs, determiners, demonstratives, adjectives, adverbs, function words, interjections, subordinate conjunctions, and coordinate conjunctions. Also includes some special ratios such as pronoun / noun and noun / verb ratios.
- **Syntactic complexity:** Counts and mean lengths of clauses, T-units, dependent clauses, and coordinate phrases as computed by Lu's syntactic complexity analyzer (Lu, 2010).
- **Tree statistics:** Max, median, and mean heights of all CFG parse trees in the narration.
- **CFG ratios:** Ratio of CFG production rule count for each of the 100 most common CFG productions from the constituency parse tree.

The following features are extracted in Mandarin Chinese:

- **Narrative length:** Number of sentences, number of characters, and mean sentence length.
- **Frequency metrics:** Type-token ratio, mean and median word frequencies.
- **POS counts:** For each part-of-speech category, the number of it in the utterance and ratio of it divided by the number of tokens. Also includes some special ratios such as pronoun / noun and noun / verb ratios.
- **Tree statistics:** Max, median, and mean heights of all CFG parse trees in the narration.
- **CFG counts:** Number of occurrences for each of the 60 most common CFG production rules from the constituency parse tree.

A.2 Top Joint Features

Table 2 lists the top English features for joint feature selection (most reconstructable from Chinese features), ordered by R^2 coefficients on the OpenSubtitles corpus. The top performing model uses the first 13 features.

#	**Feature Name**	R^2
1	Number of words	0.894
2	Number of sentences	0.828
3	Brunét's index	0.813
4	Type token ratio	0.668
5	Moving average TTR (50 word window)	0.503
6	Moving average TTR (40 word window)	0.461
7	Moving average TTR (30 word window)	0.411
8	Average word length	0.401
9	Moving average TTR (20 word window)	0.360
10	Moving average TTR (10 word window)	0.328
11	NP → PRP	0.294
12	Number of nouns	0.233
13	Mean length of clause	0.225
14	PP → IN NP	0.224
15	Total length of PP	0.222
16	Complex nominals per clause	0.220
17	Noun ratio	0.213
18	Pronoun ratio	0.208
19	Number of T-units	0.207
20	Number of PP	0.205
21	Number of function words	0.198
22	Subordinate / coordinate clauses	0.193
23	Mean word frequency	0.193
24	Number of pronouns	0.191
25	Average NP length	0.188

Table 2: Top English features for joint feature selection.

Cross-lingual Visual Verb Sense Disambiguation

Spandana Gella[*] and **Desmond Elliott**[†] and **Frank Keller**[*]
[*]School of Informatics, University of Edinburgh
[†]Department of Computer Science, University of Copenhagen
`{spandana.gella,frank.keller}@ed.ac.uk`, `de@di.ku.dk`

Abstract

Recent work has shown that visual context improves cross-lingual sense disambiguation for nouns. We extend this line of work to the more challenging task of cross-lingual *verb sense* disambiguation, introducing the MultiSense dataset of 9,504 images annotated with English, German, and Spanish verbs. Each image in MultiSense is annotated with an English verb and its translation in German or Spanish. We show that cross-lingual verb sense disambiguation models benefit from visual context, compared to unimodal baselines. We also show that the verb sense predicted by our best disambiguation model can improve the results of a text-only machine translation system when used for a multimodal translation task.

1 Introduction

Resolving lexical ambiguity remains one of the most challenging problems in natural language processing. It is often studied as a word sense disambiguation (WSD) problem, which is the task of assigning the correct sense to a word in a given context (Kilgarrif, 1998). Word sense disambiguation is typically tackled using only *textual context*; however, in a multimodal setting, *visual context* is also available and can be used for disambiguation. Most prior work on visual word sense disambiguation has targeted noun senses (Barnard and Johnson, 2005; Loeff et al., 2006; Saenko and Darrell, 2008), but the task has recently been extended to verb senses (Gella et al., 2016, 2019). Resolving sense ambiguity is particularly crucial for translation tasks, as words can have more than one translation, and these translations often correspond to word senses (Carpuat and Wu, 2007; Navigli, 2009). As an example consider the verb *ride*, which can translate into German as *fahren* (ride a bike) or *reiten* (ride a horse). Recent work on multimodal machine translation has partly addressed lexical ambiguity by using visual information, but it still remains unresolved especially for the part-of-speech categories such as verbs (Specia et al., 2016; Shah et al., 2016; Hitschler et al., 2016; Lala and Specia, 2018). Prior work on cross-lingual WSD has been limited in scale and has only employed textual context (Lefever and Hoste, 2013), even though the task should benefit from visual context, just like monolingual WSD.

Source:	Three guys **riding** on an elephant.
Target:	Drei Männer **reiten** auf einem Elefanten.
Output:	Drei Jungs **fahren** auf einem Elefanten.

Figure 1: An example of a verb sense translation error (shown in **bold red**) by the English-German neural translation system of Sennrich et al. (2017).

Visual information has been shown to be useful to map words across languages for bilingual lexicon induction. For this, images are used as a pivot between languages or visual information is combined with cross-lingual vector spaces to learn word translations across languages (Bergsma and Van Durme, 2011; Kiela et al., 2015; Vulic et al., 2016). However, as with other grounding or word similarity tasks, bilingual lexicon induction has so far mainly targeted nouns and these approaches was shown to perform poorly for other word categories such as verbs. Recent work by Gella et al. (2017) and Kádár et al. (2018) has shown using image as pivot between languages can lead to better multilingual multimodal representations and can have successful applications in crosslingual retrieval and

Proceedings of NAACL-HLT 2019, pages 1998–2004
Minneapolis, Minnesota, June 2 - June 7, 2019.

multilingual image retrieval.

In this paper, we introduce the MultiSense dataset of 9,504 images annotated with English verbs and their translations in German and Spanish. For each image in MultiSense, the English verb is translation-ambiguous, i.e., it has more than one possible translation in German or Spanish. We propose a series of disambiguation models that, given an image and an English verb, select the correct translation of the verb. We apply our models on MultiSense and find that multimodal models that fuse textual context with visual features outperform unimodal models, confirming our hypothesis that cross-lingual WSD benefits from visual context.

Cross-lingual WSD also has a clear application in machine translation. Determining the correct sense of a verb is important for high quality translation output, and sometimes text-only translation systems fail when the correct translation would be obvious from visual information (see Figure 1). To show that cross-lingual visual sense disambiguation can improve the performance of translation systems, we annotate a part of our MultiSense dataset with English image descriptions and their German translations. There are two existing multimodal translation evaluation sets with ambiguous words: the Ambiguous COCO dataset (Elliott et al., 2017) contains sentences that are "possibly ambiguous", and the Multimodal Lexical Translation dataset is restricted to predicting single words instead of full sentences (Lala and Specia, 2018). This type of resource is important for multimodal translation because it is known that humans use visual context to resolve ambiguities for nouns and gender-neutral words (Frank et al., 2018). MultiSense contains sentences that are known to have ambiguities, and it allows for sentence-level and verb prediction evaluation. Here, we use the verbs predicted by our visual sense disambiguation model to constrain the output of a neural translation system and demonstrate a clear improvement in Meteor, BLEU, and verb accuracy over a text-only baseline.

2 MultiSense Dataset

Images Paired with Verb Translations The MultiSense dataset pairs sense-ambiguous English verbs with images as visual context and contextually appropriate German and Spanish translations. Table 1 shows examples of images taken from MultiSense with their Spanish and German translations. To compile the dataset, we first chose a set of English verbs which had multiple translations into German and Spanish in Wiktionary, an online dictionary. Then we retrieved 150 candidate images from Google Images using queries that included the target English verb. We constructed the verb phrases by extracting the 100 most frequent phrases for each verb from the English Google syntactic n-grams dataset (Lin et al., 2012), which we then manually filtered to remove redundancies, resulting in 10 phrases per verb. Examples of verb phrases for *blow* include *blowing hair*, *blowing a balloon*, and *blowing up a bomb*. We filtered the candidate images using crowdworkers on Amazon Mechanical Turk, who were asked to remove images that were irrelevant to the verb phrase query. Overall pairwise agreement for this image filtering task was 0.763. Finally, we employed native German and Spanish speakers to translate the verbs into their language, given the additional visual context.

This resulted in a dataset of 9,504 images, covering 55 English verbs with 154 and 136 unique translations in German and Spanish, respectively. We divided the dataset into 75% training, 10% validation and 15% test splits.

Sentence-level Translations We also annotated a subset of MultiSense with sentence-level translations for English and German. This subset contains 995 image–English description–German translation tuples that can be used to evaluate the verb sense disambiguation capabilities of multimodal translation models. We collected the data in four-steps: (1) crowdsource English descriptions of the images using the gold-standard MultiSense verb as a prompt; (2) manually post-edit the English descriptions to ensure they contain the correct verb; (3) crowdsource German translations, given the English descriptions, the German gold-standard MultiSense verb, and the image; (4) manually post-edit the German translations to ensure they contain the correct verb. Figure 1 shows an example of an image paired with its English description and German translation.

3 Verb Sense Disambiguation Modeling

We propose three models for cross-lingual verb sense disambiguation, based on the visual input, the textual input, or using both inputs. Each model is trained to minimize the negative log probability of predicting the correct verb translation.

Spanish	mandar	hinchar	explotar
German	zublasen	aufblasen	detonieren

Table 1: Images for the English verb *blow* annotated with translations in Spanish and German. The images correspond to the uses of *blowing with a hair dryer* and *blowing a balloon*, and *blowing up a bomb*.

3.1 Unimodal Visual Model

Visual features have been shown to be useful for learning semantic representations of words (Lazaridou et al., 2015), bilingual lexicon learning (Kiela et al., 2015), and visual sense disambiguation (Gella et al., 2016), amongst others. We propose a model that learns to predict the verb translation using only visual input. Given an image **I**, we extract a fixed feature vector from a Convolutional Neural Network, and project it into a hidden layer $\mathbf{h}_v$ with the learned matrix $\mathbf{W}_i \in \mathbb{R}^{h \times 512}$ (Eqn. 1). The hidden layer is projected into the output vocabulary of v verbs using the learned matrix $\mathbf{W}_o \in \mathbb{R}^{h \times v}$, and normalized into a probability distribution using a softmax transformation (Eqn. 2).

$$\mathbf{h}_v = \mathbf{W}_i \cdot \texttt{CNN}(\mathbf{I}) + \mathbf{b}_i \tag{1}$$

$$\mathbf{y} = \texttt{softmax}(\mathbf{W}_o \cdot \mathbf{h}_v + \mathbf{b}_o) \tag{2}$$

3.2 Unimodal Textual Model

Each image in MultiSense is associated with the query phrase that was used to retrieve it. Given a query phrase with N words, we embed each word as a d-dimensional dense vector, and represent the phrase as the average of its embeddings E. We then project the query representation into a hidden layer with the learned matrix $\mathbf{W}_q \in \mathbb{R}^{h \times d}$ (Eqn. 3). The hidden layer is projected into an output layer and normalized to a probability distribution, in the same manner as the unimodal visual model.

$$\mathbf{h}_q = \mathbf{W}_q \cdot \left(\frac{1}{N} \sum_{i}^{N} \mathrm{E}[w_i] \right) + \mathbf{b}_q \tag{3}$$

3.3 Multimodal Model

We also propose a multimodal model that integrates the visual and textual features to predict the correct verb sense. In our multimodal model, we concatenate the inputs together before projecting them into a hidden layer with a learned matrix $\mathbf{W}_h \in \mathbb{R}^{h \times (512+h)}$ (Eqn. 4). We follow the same steps as the unimodal models to project the multimodal hidden layers into the output label space.

$$\mathbf{h}_{early} = \mathbf{W}_h \cdot [\texttt{CNN}(\mathbf{I})\ ;\ \mathbf{h}_q] + \mathbf{b}_h \tag{4}$$

	Chance	Majority	Text	Image	MM
German	0.7	2.8	49.1	52.1	**55.6**
Spanish	0.7	4.0	52.7	50.3	**56.0**

Table 2: Cross-lingual verb sense disambiguation accuracy of our unimodal models and the multimodal model. We also show the performance of a random chance baseline and a majority label baseline.

4 Verb Disambiguation Experiments

Our experiments are designed to determine whether the integration of textual and visual features yields better cross-lingual verb sense disambiguation than unimodal models.

4.1 Setup and Evaluation

We embed the textual queries using pre-trained $d = 300$ dimension word2vec embeddings (Mikolov et al., 2013). We represent images in the visual model using the features extracted from the 512D `pool5` layer of a pre-trained ResNet-34 CNN (He et al., 2016). All our models have a $h = 128$ dimension hidden layer. The German models have an output vocabulary of $v = 154$ verbs, and the Spanish models have a vocabulary of $v = 136$ verbs. All of our models are trained using SGD with minibatches of 16 samples and a learning rate of 0.0001.

We evaluate the performance of our models by measuring the accuracy of the predicted verb against the gold standard. We also compare against chance and majority label baselines. Our preliminary experiments show that with better visual representation we achieve better acccuracy scores similar to others who observed better visual representation contributes to better downstream tasks such as image description (Fang et al., 2015), multimodal machine translation (Specia et al., 2016) and representation learning (Kádár et al., 2018).

4.2 Results

We present the results in Table 2. The chance and majority label baselines perform very poorly. The unimodal textual model performs better than the

Source	A large herd of sheep is **blocking** the road.	A woman smiles as she **brushes** her long, dark hair.
Target	Eine große Herde Schafe **blockiert** die Straße.	Eine Frau lächelt während sie sich ihre dunklen langen Haare **bürstet** .
Baseline	Eine große Herde Schafe **kriecht** die Straße entlang.	Eine Frau lächelt , als sie ihren langen und dunklen Haaren **putzt** .
+WSD	Eine große Herde Schafe **blockieren** die Straße.	Eine Frau lächelt , als sie ihr lange , dunklen Haaren **bürsten** .

Table 3: The visual verb sense predictions ("blockieren", "bürsten") successfully constrains the decoder to predict the correct sense of the verb ("block", "brush") in the German translation **(+WSD)**. The incorrect verb in the baseline translation is shown in **bold red.**

looking
for directions

Model	German	Spanish
Textual	schauen	mirar
Visual	tragen	**buscar**
MM	**suchen**	**buscar**

Figure 2: Examples of the Top-1 predictions of our unimodal and multimodal models. Only the early fusion multimodal model predicts the correct verb sense for both languages (shown in bold).

unimodal visual model for German verb sense disambiguation, but we find the opposite for Spanish unimodal verb sense disambiguation. However, the early fusion multimodal model outperforms the best unimodal model for both German and Spanish. This confirms that cross-lingual verb sense disambiguation benefits from multimodal supervision compared to unimodal supervision.

4.3 Discussion

We analyzed the outputs of our models in order to understand where multimodal features helped in identifying the correct verb translation and the cases where they failed. In Figure 2, we show an example that illustrates how varying the input (textual, visual, or multimodal) affects the accuracy of the verb prediction. We show the top verb predicted by our models for both German and Spanish. The top predicted verb using text-only visual features is incorrect. The unimodal visual features model predicts the correct Spanish verb but the incorrect German verb. However, when visual information is added to textual features, models in both the languages predict the correct label.

	Meteor	BLEU	VAcc
Baseline NMT	38.6	17.8	22.9
+ Predicted Verb	40.0	18.5	49.5
+ Oracle Verb	40.4	19.1	77.7
Caglayan et al.	46.1	25.8	29.3
Helcl & Libovický	42.5	22.3	25.1

Table 4: Translation results: Meteor and BLEU are standard text-similarity metrics; verb accuracy (VAcc) counts how often the model proposal contains the gold standard German verb.

5 Machine Translation Experiments

We also evaluate our verb sense disambiguation model in the challenging downstream task of multimodal machine translation (Specia et al., 2016). We conduct this evaluation on the sentence-level translation subset of MultiSense. We evaluate model performance using BLEU (Papineni et al., 2002) and Meteor scores (Denkowski and Lavie, 2014) between the MultiSense reference description and the translation model output. We also evaluate the verb prediction accuracy of the output against the gold standard verb annotation.

5.1 Models

Our baseline is an attention-based neural machine translation model (Hieber et al., 2017) trained on the 29,000 English-German sentences in Multi30k (Elliott et al., 2016). We preprocessed the text with punctuation normalization, tokenization, and lowercasing. We then learned a joint byte-pair-encoded vocabulary with 10,000 merge operations to reduce sparsity (Sennrich et al., 2016).

Our approach uses the German verb predicted by the unimodal visual model (Section 3.1) to constrain the output of the translation decoder (Post and Vilar, 2018). This means that our approach does not directly use visual features, instead it uses the output of the visual verb sense disambiguation model to guide the translation process.

We compare our approach against two state-of-the-art multimodal translation systems: Caglayan et al. (2017) modulate the target language word embeddings by an element-wise multiplication with a learned transformation of the visual data; Helcl and Libovický (2017) use a double attention model that learns to selectively attend to a combination of the source language and the visual data.

5.2 Results

Table 4 shows the results of the translation experiment. Overall, the Meteor scores are much lower than on the Multi30k test sets, where the state-of-the-art single model scores 51.6 Meteor points compared to 46.1 Meteor we obtained. This gap is most likely due evaluating the models on an out-of-domain dataset with out-of-vocabulary tokens. Using the predicted verb as a decoding constraint outperforms the text-only translation baseline by 1.4 Meteor points. In addition, the translation output of our model contains the correct German verb 27% more often than the text-only baseline model. These results show that a multimodal verb sense disambiguation model can improve translation quality in a multimodal setting.

We also calculated the upper bound of our approach by using the gold standard German verb as the lexical constraint. In this oracle experiment we observed a further 0.4 Meteor point improvement over our best model, and a further 27% improvement in verb accuracy. This shows that: (1) there are further improvements to be gained from improving the verb disambiguation model, and (2) the OOV rate in German means that we cannot achieve perfect verb accuracy.

6 Conclusions

We introduced the MultiSense dataset of 9,504 images annotated with an English verb and its translation in Spanish and German. We proposed a range of cross-lingual visual sense disambiguation models and showed that multimodal models that fuse textual and visual features outperform unimodal models. We also collected a set of image descriptions and their translations, and showed that the output of our cross-lingual WSD system boosts the performance of a text-only translation system on this data. MultiSense is publicly available at `https://github.com/spandanagella/multisense`

Acknowledgements

DE was supported by an Amazon Research Award. This work was supported by the donation of a Titan Xp GPU by the NVIDIA Corporation.

References

Kobus Barnard and Matthew Johnson. 2005. Word sense disambiguation with pictures. *Artificial Intelligence*, 167(1-2):13–30.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, page 1764.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017. LIUM-CVC Submissions for WMT17 Multimodal Translation Task. In *Proceedings of the Second Conference on Machine Translation*.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *EMNLP-CoNLL*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation*.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language, VL@ACL 2016*.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pages 1473–1482.

Stella Frank, Desmond Elliott, and Lucia Specia. 2018. Assessing multilingual multimodal image description: Studies of native speaker preferences and translator choices. *Natural Language Engineering*, 24(3):393—413.

Spandana Gella, Frank Keller, and Mirella Lapata. 2019. Disambiguating visual verbs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):311–322.

Spandana Gella, Mirella Lapata, and Frank Keller. 2016. Unsupervised visual sense disambiguation for verbs using multimodal embeddings. In *Proceedings of 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Spandana Gella, Rico Sennrich, Frank Keller, and Mirella Lapata. 2017. Image pivoting for learning multilingual multimodal representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Jindřich Helcl and Jindřich Libovický. 2017. Cuni system for the wmt17 multimodal translation task. In *Proceedings of the Second Conference on Machine Translation*, pages 450–457.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1712.05690*.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Ákos Kádár, Desmond Elliott, Marc-Alexandre Côté, Grzegorz Chrupala, and Afra Alishahi. 2018. Lessons learned in multilingual grounded language learning. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018*, pages 402–412.

Douwe Kiela, Ivan Vulic, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred convnet features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Adam Kilgarrif. 1998. Senseval: An exercise in evaluating word sense disambiguation programs. In *Proc. of the first international conference on language resources and evaluation*.

Chirag Lala and Lucia Specia. 2018. Multimodal lexical translation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2018)*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 153–163.

Els Lefever and Véronique Hoste. 2013. Semeval-2013 task 10: Cross-lingual word sense disambiguation. *Proc. of SemEval*, pages 158–166.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174.

Nicolas Loeff, Cecilia Ovesdotter Alm, and David A. Forsyth. 2006. Discriminating image senses by clustering with multimodal features. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics NAACL-HLT 2018*, pages 1314–1324.

Kate Saenko and Trevor Darrell. 2008. Unsupervised learning of visual sense models for polysemous words. In *Advances in Neural Information Processing Systems*, pages 1393–1400.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, WMT 2017*, pages 389–399.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Kashif Shah, Josiah Wang, and Lucia Specia. 2016. Shef-multimodal: Grounding machine translation on images. In *Proceedings of the First Conference on Machine Translation, WMT*, pages 660–665.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation*, pages 543–553.

Ivan Vulic, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-modal representations for improved bilingual lexicon learning. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 188.

Subword-Level Language Identification for Intra-Word Code-Switching

Manuel Mager[1], **Özlem Çetinoğlu**[1] and **Katharina Kann**[2]
[1]Institute for Natural Language Processing,
University of Stuttgart, Germany
[2]Center for Data Science, New York University, USA
`{manuel.mager, ozlem}@ims.uni-stuttgart.de, kann@nyu.edu`

Abstract

Language identification for code-switching (CS), the phenomenon of alternating between two or more languages in conversations, has traditionally been approached under the assumption of a single language per token. However, if at least one language is morphologically rich, a large number of words can be composed of *morphemes* from more than one language (intra-word CS). In this paper, we extend the language identification task to the subword level, such that it includes splitting mixed words while tagging each part with a language ID. We further propose a model for this task, which is based on a segmental recurrent neural network. In experiments on a new Spanish–Wixarika dataset and on an adapted German–Turkish dataset, our proposed model performs slightly better than or roughly on par with our best baseline, respectively. Considering only mixed words, however, it strongly outperforms all baselines.

1 Introduction

In settings where multilingual speakers share more than one language, mixing two or more languages within a single piece of text, for example a tweet, is getting increasingly common (Grosjean, 2010). This constitutes a challenge for natural language processing (NLP) systems, since they are commonly designed to handle one language at a time.

Code-switching (CS) can be found in multiple non-exclusive variants. For instance, sentences in different languages can be mixed within one text, or words from different languages can be combined into sentences. CS can also occur on the subword level, when speakers combine morphemes from different languages (*intra-word CS*). This last phenomenon can mostly be found if at least one of the languages is morphologically rich. An example for intra-word CS between the Romance language Spanish and the Yuto-Aztecan language Wixarika[1] is shown in Figure 1.

(a)	*ne'iwa*	*pecansadoxɨ*
	WIX	MIXED
	my.brother	you-are.tired.PPFV

(b)	*ne'iwa*	*pe*	*cansado*	*xɨ*
	WIX	WIX	ES	WIX
	my.brother	you-are	tired	PPFV

'My brother, you are tired.'

Figure 1: Intra-word CS between Spanish and Wixarika, **(a)** standard LID for CS, **(b)** our task. PPFV stands for past perfective.

CS language identification (LID) , i.e., predicting the language of each token in a text, has attracted a lot of attention in recent years (cf. Solorio et al. (2014); Molina et al. (2016)). However, intra-word mixing is mostly not handled explicitly: words with morphemes from more than one language are simply tagged with a `mixed` label.

While this works reasonably well for previously studied language pairs, overlooking intra-word CS leads to a major loss of information for highly polysynthetic languages. A mixed word is unknown for NLP systems, yet a single word contains much more information, cf. Figure 1 (b). Furthermore, we find intra-word CS to be much more frequent for Spanish–Wixarika than for previously studied language pairs, such that it is crucial to handle it.

Motivated by these considerations, we extend the LID task to the subword level (from (a) to (b) in Figure 1). We introduce a new CS dataset for Spanish–Wixarika (ES–WIX) and modify an existing German–Turkish (DE–TR) CS corpus (Çetinoğlu, 2016) for our purposes. We then intro-

---

[1] Wixarika, also known as Huichol, is a polysynthetic Mexican indigenous language.

Proceedings of NAACL-HLT 2019, pages 2005–2011
Minneapolis, Minnesota, June 2 - June 7, 2019.

duce a segmental recurrent neural network (SegRNN) model for the task, which we compare against several strong baselines. Our experiments show clear advantages of SegRNNs over all baselines for intra-word CS.

2 Related Work

The task of LID for CS has been frequently studied in the last years (Al-Badrashiny and Diab, 2016; Rijhwani et al., 2017; Zhang et al., 2018), including two shared tasks on the topic (Solorio et al., 2014; Molina et al., 2016). The best systems (Samih et al., 2016; Shirvani et al., 2016) achieved over 90% accuracy for all language pairs. However, intra-word CS was not handled explicitly, and often systems even failed to correctly assign the `mixed` label. For Nepali–English, Barman et al. (2014) correctly identified some of the mixed words with a combination of linear kernel support vector machines and a k-nearest neighbour approach. The most similar work to ours is Nguyen and Cornips (2016), which focused on detecting intra-word CS for Dutch–Limburgish (Nguyen et al., 2015). The authors utilized Morfessor (Creutz and Lagus, 2002) to segment all words into morphemes and Wikipedia to assign LID probabilities to each morpheme. However, their task definition and evaluation are on the word level. Furthermore, as this method relies on large monolingual resources, it is not applicable to low-resource languages like Wixarika, which does not even have its own Wikipedia edition.

Subword-level LID consists of both segmentation and tagging of words. An earlier approach to handle a similar scenario was the connectionist temporal classification (CTC) model developed by Graves et al. (2006). The disadvantage of this model was the lack of prediction of the segmentation boundaries that are necessary for our task. Kong et al. (2016) later proposed the SegRNN model that segments and labels jointly, with successful applications on automatic glossing of polysynthetic languages (Micher, 2017, 2018). Segmentation of words into morphemes alone has a long history in NLP (Harris, 1951), including semi- or unsupervised methods (Goldsmith, 2001; Creutz and Lagus, 2002; Hammarström and Borin, 2011; Grönroos et al., 2014), as well as supervised ones (Zhang and Clark, 2008; Ruokolainen et al., 2013; Cotterell et al., 2015; Kann et al., 2018).

3 Task and Data Description

3.1 Task Description

Formally, the task of subword-level LID consists of producing two sequences, given an input sequence of tokens $X = \langle x_1, \ldots, x_i, \ldots, x_{|X|} \rangle$. The first sequence contains all words and splits $X^s = \langle x_1^s, \ldots, x_i^s, \ldots, x_{|X|}^s \rangle$, where each x_i^s is an m-tuple of variable length $0 < m \leq |x_i|$, where $|x_i|$ is the number of characters in x_i. The second sequence is such that $T^s = \langle t_1^s, \ldots, t_i^s, \ldots, t_{|X|}^s \rangle$, where $|T^s| = |X^s| = |X|$ and each $t_i^s \in T^s$ is an n-tuple of tags from a given set of LID tags. An input–output example for a DE–TR mixed phrase is shown in Figure 2.

Input	⟨ ‘Yerim’,	‘seni’,	‘,’,	‘danke’,	‘Schatzym’⟩
Output	⟨ (Yerim),	(seni),	(,),	(danke),	(Schatzy, m)⟩
	⟨ (TR),	(TR),	(OTHER),	(DE),	(DE, TR)⟩

Figure 2: Subword-level LID in German–Turkish.

3.2 Datasets

German–Turkish The German–Turkish Twitter Corpus (Çetinoğlu and Çöltekin, 2016) consists of 1029 tweets with 17K tokens. They are manually normalized, tokenized, and annotated with language IDs. The language ID tag set consists of `TR` (Turkish), `DE` (German), `LANG3` (other language), `MIXED` (intra-word CS), `AMBIG` (ambiguous language ID in context), and `OTHER` (punctuation, numbers, emoticns, symbols, etc.). Named entities are tagged with a combination of `NE` and their language ID: `NE.TR`, `NE.DE`, `NE.LANG3`. In the original corpus, some Turkish and mixed words undergo a morphosyntactic split,[2] with splitting points not usually corresponding to language boundaries. For the purpose of subword-level LID, these morphosyntactic splits are merged back into single words. We then manually segment `MIXED` words at language boundaries, and replace their labels with more fine-grained language ID tags. The total percentage of mixed words is 2.75%. However, the percentage of sentences with mixed words is 15.66%. The complete dataset statistics can be found in Table 1.

Spanish–Wixarika Our second dataset consists of 985 sentences and 8K tokens in Spanish and Wixarika. Wixarika is spoken by approximately

---

[2] E.g., separating copular suffixes from roots they are attached to, cf. Çetinoğlu and Çöltekin (2016) for details.

Tokens	All	%	Unique	Unique %
DE	3992	20.37	1360	20.43
TR	9913	50.59	4071	61.16
LANG3	112	0.57	83	1.25
AMBIG	32	0.16	23	0.18
OTHER	4345	22.17	294	4.42
NE.TR	417	2.13	275	4.13
NE.DE	389	1.99	244	3.67
NE.AMBIG	16	0.08	12	1.25
NE.LANG3	112	0.57	95	1.43
MIXED	231	1.18	183	2.75
DE TR	231	100.0	183	100.0

Table 1: The frequency breakdown of tokens by language IDs in the German-Turkish dataset. *All*: the total number of tokens per tag, *%*: the percentage of them with respect to the total number of tokens; *Unique*: the number of unique word types, and *Unique %*: the percentage of them with respect the total number of unique word types.

Tokens	All	%	Unique	Unique %
ES	4218	50.73	1527	45.76
WIX	2019	24.28	1191	35.69
EN	24	0.29	21	0.63
AMBIG	28	0.34	25	0.75
OTHER	1664	20.01	288	8.63
NE.ES	96	1.15	85	2.55
NE.WIX	77	0.93	49	1.47
NE.EN	11	0.13	9	0.27
MIXED	177	2.13	142	4.26
ES WIX	35	*19.77*	31	*21.83*
WIX ES	122	*68.93*	93	*65.49*
WIX ES WIX	17	*9.60*	31	*10.56*
WIX EN	1	*0.07*	1	*0.07*
EN ES	1	*0.07*	1	*0.07*

Table 2: Number of tokens classified by language tags seen in the Spanish-Wixarika dataset. We show the total number of *Tokens* per tag, their proportion (*%*) with the total tokens, the *Unique* word types, and the proportion (*Unique %*) of them with the total number of unique word types.

$50,000$ people in the Mexican states of Durango, Jalisco, Nayarit and Zacatecas (Leza and López, 2006) and is polysynthetic, with most morphemes ocurring in verbs. The data is collected from public postings and comments from Facebook accounts. To ensure the public characteristic of these posts, we manually collect data that is accessible publicly without being logged in to Facebook, to comply with the terms of use and privacy of the users. These posts and comments are taken from 34 users: 14 women, 10 men, and the rest does not publically reveal their gender. None of them have publically mentioned their age. To get a dataset that focuses on the LID task, we only consider threads where the CS phenomenon appears. We replace usernames with `@username` in order to preserve privacy. Afterwards, we tokenize the text, segment mixed words, and add language IDs to words and segments.

The tag set is parallel to that of German–Turkish: `ES` (Spanish), `WIX` (Wixarika), `EN` (English), `AMBIG` (ambiguous) `OTHER` (punctuation, numbers, emoticons, etc), `NE.ES`, `NE.WIX` and `NE.EN` (named entities). Mixed words are segmented and each segment is labeled with its corresponding language (`ES`, `WIX`, `EN`). Table 2 shows a detailed description of the dataset. The percentage of mixed words is higher than in the DE–TR dataset: 3.13% of the tokens and 4.26% of the types. The most common combination is Spanish roots with Wixarika affixes. Furthermore, 16.55% of the sentences contain mixed words.

We split the DE–TR corpus and the ES–WIX corpus into training and test sets of sizes 800:229 and 770:216, respectively. Error analysis and hyperparameter tuning are done on the training set via 5-fold cross-validation. We present results on the test sets. Both datasets are available at `https://www.ims.uni-stuttgart.de/institut/mitarbeiter/ozlem/NAACL2019.html`

4 Experiments

Our main system is a neural architecture that jointly solves the segmentation and language identification tasks. We compare it to multiple pipeline systems and another joint system.

4.1 SegRNN

We suggest a SegRNN (Kong et al., 2016) would be the best fit for our task because it models a joint probability distribution over possible segmentations of the input and labels for each segment.

The model is trained to optimize the following objective, which corresponds to the joint log-likelihood of the segment lengths e and the language tags t:

$$\mathcal{L}(\theta) = \sum_{(x,t,e)\in\mathcal{D}} - \log p(t, e|x) \qquad (1)$$

$\mathcal{D}$ denotes the training data, θ is the set of model parameters, x the input, t the tag sequence and e the sequence of segment lengths.

Our inputs are single words.[3] As hyperparame-

---

[3] We also experimented with entire phrases as inputs, and the achieved scores were slightly worse than for word-based inputs.

	DE–TR							ES–WIX						
	Segmentation			Tagging			Char	Segmentation			Tagging			Char
	P	R	F1	P	R	F1	Acc.	P	R	F1	P	R	F1	Acc.
SegRNN	60.4	46.8	**53.0**	78.8	60.2	74.0	**72.9**	75.6	62.7	**68.5**	85.3	70.5	**77.2**	**84.6**
BiLSTM+Seq2Seq	46.1	33.4	38.7	84.3	66.8	**74.5**	67.7	66.6	52.2	58.5	82.4	66.2	73.4	78.4
BiLSTM+CRF	49.4	34.5	40.6	84.3	66.8	**74.5**	68.0	61.1	48.4	54.0	82.4	66.3	73.4	76.7
CRFTag+Seq2Seq	12.7	6.8	8.9	27.8	15.2	19.7	37.2	47.2	31.9	38.1	63.5	43.0	51.3	69.6
CRFTag+CRF	11.0	5.9	7.7	27.8	15.2	19.7	36.8	47.6	32.4	38.6	63.5	43.0	51.3	69.4
CharBiLSTM	19.1	26.6	22.2	32.9	45.7	38.2	61.1	49.7	52.2	50.8	63.1	68.1	66.5	75.7

Table 3: Segmentation and LID test results for mixed words only.

ters we use: 1 RNN layer, a 64-dimensional input layer, 32 dimensions for tags, 16 for segments, and 4 for lengths. For training, we use Adam (Kingma and Ba, 2014).

4.2 Baselines

BiLSTM+Seq2Seq/BiLSTM+CRF Our first baselines are pipelines. First, the input text is tagged with language IDs. Language IDs of a mixed word are directly predicted as a combination of all language ID tags of the word (i.e., `WIX_ES`). Second, a subword-level model segments words with composed language ID tags. For word-level tagging, we use a hierarchical bidirectional LSTM (**BiLSTM**) that incorporates both token- and character-level information (Plank et al., 2016), similar to the winning system (Samih et al., 2016) of the Second Code-Switching Shared Task (Molina et al., 2016). [4] For the subword level, we use two supervised segmentation methods: a **CRF** segmenter proposed by Ruokolainen et al. (2013), that models segmentation as a labeling problem and a sequence-to-sequence (**Seq2Seq**) model trained with an auxiliary task as proposed by Kann et al. (2018).

CRFTag+Seq2Seq/CRFTag+CRF Since our datasets might be small for training neural networks, we substitute the BiLSTM with a CRF tagger (Müller et al., 2013, **CRFTag**) in the first step. For segmentation, we use the same two approaches as for the previous baselines.

CharBiLSTM We further employ a BiLSTM to tag each character with a language ID. For training, each character inherits the language ID of the word or segment it belongs to. At prediction time, if the characters of a word have different language IDs, the word is split.

4.3 Metrics

We use two metrics for evaluation. First, we follow Kong et al. (2016) and calculate precision (P), recall (R), and F1, using segments as units (an unsegmented word corresponds to one segment). We also report a tagging accuracy (Char Acc.) by assigning a language ID to each character and calculating the ratio of correct language tags over all characters.

	DE–TR			ES–WIX		
	Seg. F1	Tag. F1	Char Acc.	Seg. F1	Tag. F1	Char Acc.
SegRNN	**98.7**	94.0	93.6	97.8	**92.5**	**92.4**
BiLSTM+Seq2Seq	98.6	**95.1**	94.3	**98.1**	90.9	90.7
BiLSTM+CRF	**98.7**	94.9	**94.4**	97.9	87.8	90.6
CRFTag+Seq2Seq	98.4	93.7	93.1	97.7	90.4	90.1
CRFTag+CRF	98.4	93.7	93.1	97.6	90.3	90.1
CharBiLSTM	87.7	88.0	92.5	89.7	87.9	91.3

Table 4: Test set results for entire datasets.

4.4 Results and Discussion

Table 4 shows all test results for the entire datasets. We find the following: (i) For ES–WIX, SegRNN performs slightly better for tagging than the best baseline, both in terms of F1 and character accuracy. For DE–TR, SegRNN and BiLSTM+CRF are the best segmentation models, but the BiLSTM models slightly outperform SegRNN for tagging. (ii) The CRF pipelines perform slightly worse than the best word-level BiLSTM models for both datasets and all evaluations.

Table 3 shows the results of tagging and segmentation only for the mixed words in our datasets. Here, we can see that: (i) Our SegRNN model achieves the best performance for segmentation. Differences to the other approaches are $\geq$ 10%, showing clearly why these models are good for the task when the number of words belonging to two languages is high. (ii) The pipeline BiLSTM models work best for tagging of the DE–TR data with a slight margin, but underperform

---

[4] For all BiLSTM models input dimension is 100 with a hidden layer size of 100. For training we use a stochastic gradient descent (Bottou, 2010), 30 epochs, with a learning rate of 0.1. A 0.25 dropout factor is applied.

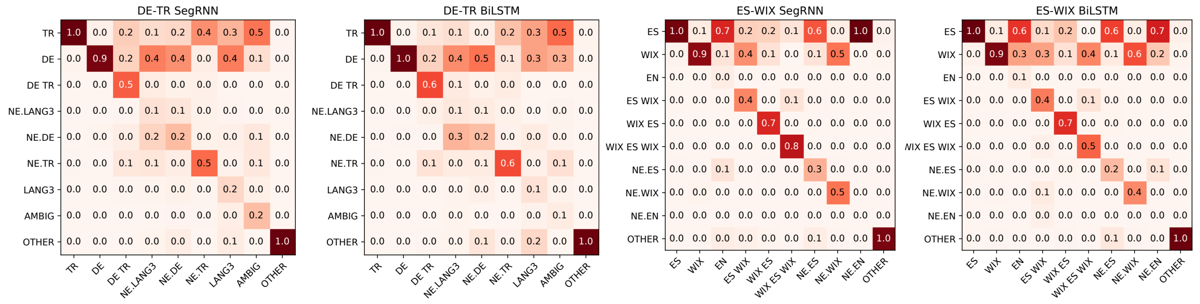


Figure 3: Confusion matrices of the two best models on both datasets. The x axis represents tags seen in the gold standard, and the y axis shows the corresponding predicted tags. Values are rounded up, therefore not all columns add up to 1.

on the ES–WIX dataset as compared to the SegRNN models. (iii) Both CRFTag models achieve very low results for both segmentation and tagging. (iv) CharBiLSTM performs better than the CRFTag models on both tasks, but is worse than all other approaches in our experiments.

More generally, we further observe that recall on mixed words for the DE–TR pair is low for all systems, as compared to ES–WIX. This effect is especially strong for the CRFTag and CharBiLSTM models, which seem to be unable to correctly identify mixed words. While this tendency can also be seen for the ES–WIX pair, it is less extreme. We suggest that the better segmentation and tagging of mixed words for ES–WIX might mostly be due to the higher percentage of available examples of mixed words in the training set for ES–WIX.

Overall, we conclude that SegRNN models seem to work better on language pairs that have more intra-word CS, while pipeline approaches might be as good for language pairs where the number of mixed words is lower.

Error analysis. Figure 3 shows confusion matrices for SegRNN and BiLSTM+Seg2Seg. Both models achieve good results assigning monolingual tags (`ES`, `WIX`, `DE`, `TR`) and punctuation symbols (`OTHERS`). The hardest labels to classify are named entities (`NE`, `NE.TR`, `NE.TR`, `NE.WIX`, `NE.ES`), as well as third language and ambiguous tags (`LANG3`, `EN`, `AMBIG`). Performance on multilingual tags (`DE TR`, `WIX ES`, `ES WIX`, `WIX ES WIX`) is mixed. For `DE TR`, BiLSTM+Seq2Seq gets slightly better classifications, but for the ES–WIX tags SegRNN achieves better results.

Regarding oversegmentation problems, BiLSTM+Seq2Seq (0.8% for DE–TR and 2.0% for ES–WIX) slightly underperforms SegRNN (0.7% for DE–TR and 1.13% for ES–WIX). The BiLSTM+Seq2Seq (2.4%) makes fewer undersegmentation errors for DE–TR than SegRNN (2.7%). However, for ES–WIX, SegRNN performs better with 3.81% undersegmentation errors compared to 4.2% of BiLSTM+Seq2Seq.

5 Conclusion

In this paper, we extended the LID task to the subword level, which is particularly important for code-switched text in morphologically rich languages. We further proposed a SegRNN model for the task and compared it to several strong baselines. Investigating the behaviour of all systems, we found that pipelines including a BiLSTM tagger work well for tagging DE–TR, where the number of mixed tokens is not that high, but that our proposed SegRNN approach performs better than all other systems for ES–WIX. Also, SegRNNs have clear advantages over all baselines if we consider mixed words only. Our subword-level LID datasets for ES–WIX and DE–TR are publicly available.

Acknowledgments

We would like to thank Mohamed Balabel, Sam Bowman, Agnieszka Falenska, Ilya Kulikov and Phu Mon Htut for their valuable feedback. We also want to thank Jeffrey Micher for his help with the setup of the SegRNN code. This project has benefited from financial support to Manuel Mager and Özlem Çetinoğlu by DFG via project CE 326/1-1 "Computational Structural Analysis of German-Turkish Code-Switching", to Manuel Mager by DAAD Doctoral Research Grant, and to Katharina Kann by Samsung Research.

References

Mohamed Al-Badrashiny and Mona Diab. 2016. Lili: A simple language independent approach for language identification. In *COLING*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. *EMNLP*.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*. Springer.

Özlem Çetinoğlu. 2016. A Turkish-German code-switching corpus. In *LREC*.

Özlem Çetinoğlu and Çağrı Çöltekin. 2016. Part of speech annotation of a Turkish-German code-switching corpus. In *LAW-X*, Berlin, Germany.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *CoNLL*.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *SIGMORPHON*.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*. ACM.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor flatcat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *COLING*.

F. Grosjean. 2010. *Bilingual: Life and Reality*. Harvard University Press.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2).

Zellig S Harris. 1951. Methods in structural linguistics.

Katharina Kann, Manuel Mager, Ivan Meza, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *NAACL-HLT*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ICLR*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. *Interspeech*.

José Luis Iturrioz Leza and Paula Gómez López. 2006. *Gramática wixarika*, volume 1. Lincom Europa.

Jeffrey Micher. 2017. Improving coverage of an inuktitut morphological analyzer using a segmental recurrent neural network. In *ComputEL*.

Jeffrey Micher. 2018. Using the Nunavut Hansard data for experiments in morphological analysis and machine translation. In *PYLO*.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *CS Workshop*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *EMNLP*.

Dong Nguyen and Leonie Cornips. 2016. Automatic detection of intra-word code-switching. In *SIGMORPHON*.

Dong Nguyen, Dolf Trieschnigg, and Leonie Cornips. 2015. Audience and the use of minority languages on twitter. In *ICWSM*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *ACL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *CoNLL*.

Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. 2016. Multilingual code-switching identification via LSTM recurrent neural networks. In *CS Workshop*.

Rouzbeh Shirvani, Mario Piergallini, Gauri Shankar Gautam, and Mohamed Chouikha. 2016. The howard university system submission for the shared task in language identification in Spanish-English codeswitching. In *CS Workshop*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. *CS Workshop*.

Yuan Zhang, Jason Riesa, Daniel Gillick, Anton Bakalov, Jason Baldridge, and David Weiss. 2018. A fast, compact, accurate model for language identification of codemixed text. In *EMNLP*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. *ACL-HLT*.

MuST-C: a Multilingual Speech Translation Corpus

Mattia Antonino Di Gangi[1,2], **Roldano Cattoni**[1], **Luisa Bentivogli**[1],
Matteo Negri[1] and **Marco Turchi**[1]
[1]Fondazione Bruno Kessler
[2]University of Trento
Trento, Italy
`{digangi,cattoni,bentivo,negri,turchi}@fbk.eu`

Abstract

Current research on spoken language translation (SLT) has to confront with the scarcity of sizeable and publicly available training corpora. This problem hinders the adoption of neural end-to-end approaches, which represent the state of the art in the two parent tasks of SLT: automatic speech recognition and machine translation. To fill this gap, we created MuST-C, a multilingual speech translation corpus whose size and quality will facilitate the training of end-to-end systems for SLT from English into 8 languages. For each target language, MuST-C comprises at least 385 hours of audio recordings from English TED Talks, which are automatically aligned at the sentence level with their manual transcriptions and translations. Together with a description of the corpus creation methodology (scalable to add new data and cover new languages), we provide an empirical verification of its quality and SLT results computed with strong baseline system on each language direction.

1 Introduction

Besides the increased computing power, the recent surge of neural end-to-end approaches to natural language processing tasks has been stoked by the increased availability of data. For instance, when supported by sizeable training corpora, the robustness and the strong generalization capabilities of neural networks led to their dominance over previous paradigms both in automatic speech recognition (ASR (Chiu et al., 2018)) and machine translation (MT (Bojar et al., 2018)).

Compared to its two parent research areas, spoken language translation (SLT) has not shown such a steady progress yet. Despite recent claims by big industry players about the effectiveness of end-to-end learning (Weiss et al., 2017; Jia et al., 2018), its adoption does not yet represent the mainstream solution to the SLT task. One of the main obstacles to a stable dominance of the end-to-end paradigm also in this area is the scarcity of training corpora. While cascade ASR+MT solutions can exploit the wealth of task-specific data available for each of the two tasks,[1] the situation for end-to-end model training is much less favourable. As shown in Table 1, few publicly available corpora exist, their language coverage is rather limited and, most importantly, their size is often too small (less than 100 hours of translated audio) for training data-hungry neural models.[2]

Corpus	Languages	Hours
Niehues et al. (2018)	En→De	273
Kocabiyikoglu et al. (2018)	En→Fr	236
Tohyama et al. (2005)	En↔Jp	182
Paulik and Waibel (2009)	En→Es	111
	Es→En	105
Post et al. (2013)	En→Es	38
Stüker et al. (2012)	De→En	37
Shimizu et al. (2014)	En↔Jp	22
Federmann and Lewis (2017)	En↔Jp/Zh	22
Bendazzoli and Sandrelli (2005)	En↔It/Es It↔Es	18
Bérard et al. (2016)	Fr→En	17
Federmann and Lewis (2016)	En↔Fr/De	8
Woldeyohannis et al. (2017)	Am→En	7
Godard et al. (2017)	Mboshi→Fr	4

Table 1: Publicly available SLT corpora. The two most recent resources (also known as *IWSLT18* and *Augmented LibriSpeech*) are also the largest ones. Though considerably smaller, the *Fisher and Callhome* corpus described in (Post et al., 2013) is among the most widely used ones in previous research.

To circumvent the problem, neural SLT approaches currently rely on: *i)* large proprietary corpora (Jia et al., 2018), *ii)* multitask learning

---

[1] In resource-rich conditions, ASR and MT training often builds on thousands of hours of transcribed speech and tens of millions of parallel sentences, respectively.

[2] Besides the corpora reported in Table 1, several smaller (< 4 hours) freely-available datasets have been created (*e.g.* the IWSLT evaluation campaign development and test sets from 2010 to 2017 and the Griko-Italian corpus by Boito et al. (2018)).

Proceedings of NAACL-HLT 2019, pages 2012–2017
Minneapolis, Minnesota, June 2 - June 7, 2019.

(Weiss et al., 2017; Anastasopoulos and Chiang, 2018; Bérard et al., 2018), *iii)* encoder/decoder pre-training (Bansal et al., 2018; Bérard et al., 2018), *iv)* synthesized speech data (Bérard et al., 2016), or *v)* machine-translated target text data (Bérard et al., 2018). Though effective, solutions *ii)* and *iii)* assume the availability of ASR and MT data, which is not always guaranteed (especially in low-resource language settings). Solutions *iv)* and *v)*, instead, rely on training material derived from sub-optimal automatic data creation/augmentation procedures. This situation calls for initiatives towards the creation of large, high-quality multilingual corpora suitable to explore end-to-end SLT in more favorable conditions similar to condition *i)*. Along this direction, our contributions are:

- A large ($\sim$400 hours of speech per language) multilingual corpus for SLT from English into 8 languages (German, Spanish, French, Italian, Dutch, Portuguese, Romanian and Russian);
- An empirical verification of its quality;
- ASR, MT and SLT results computed with strong baseline systems on each language direction.

MuST-C is released under a Creative Commons license, Attribution - Non Commercial - No Derivatives (CC BY NC ND 4.0 International), and is freely downloadable at `mustc.fbk.eu`

2 Corpus Creation Methodology

Must-C was created pursuing high quality as well as large size, speaker variety (male/female, native/non-native) and coverage in terms of topics and languages. To achieve these objectives, similar to (Niehues et al., 2018), we started from English TED Talks, in which a variety of speakers discuss topics spanning from business to science and entertainment. Most importantly, the fact that TED talks are often manually transcribed and translated sets ideal conditions for creating an SLT corpus from high-quality text material. Although the initial data are similar to those used to build the IWSLT18 corpus, our methodology is different. Inspired by Kocabiyikoglu et al. (2018), it exploits automatic alignment procedures, first at the text level (between transcriptions and translations) and then with the corresponding audio segments.

More in detail, for each target language L_i, the (English-L_i) section of MuST-C is created as follows. First, for all the English talks available from the TED website,[3] we download the videos and the HTML files containing the manual transcriptions and their translation into L_i.[4]

Then, the plain text transcription and the translation of each talk are split at the sentence level based on strong punctuation marks and aligned using the Gargantua sentence alignment tool (Braune and Fraser, 2010). This step produces a bilingual text corpus aligned at the sentence level.

In the third step, the English side of this bilingual corpus is aligned to the corresponding audio track extracted from the video. This is done using Gentle,[5] an off-the-shelf English forced-aligner built on the Kaldi ASR toolkit (Povey et al., 2011).

Next, the audio-text alignments are processed to create a YAML file containing time information (*i.e.* start and duration) for each sentence. In this processing step, two filters are applied to weed out potentially noisy segments, or entire talks, based on the number of words that were not aligned by Gentle. First, entire talks are discarded if the proportion of unrecognized words is equal or greater than 15% of the total. This threshold was determined after a manual analysis of 73 talks (those with the highest percentage of unrecognized words). The analysis showed that these cases are representative of different types of noise like: *i)* non-English speech, *ii)* long silences, *iii)* music, non-transcribed songs and videos played during the talk, and *iv)* wrong transcriptions (*e.g.* captions from other talks in the material downloaded from the TED website). The second rule applies to the single sentences of the talks that passed the first filter, and removes those in which none of the words was aligned by Gentle.[6]

In the last step, the log Mel 40-dimensional filter-bank features – commonly used as input representation for ASR (Graves et al., 2013) and SLT (Weiss et al., 2017) – are extracted from the

---

[3] `www.ted.com` – dump of April 2018.

[4] All talks have manual captions, which were also translated into many languages by volunteers. The language coverage of the translations depends on several factors like the age of the talk (the old ones often have more translations), the popularity of its topic and the availability of volunteer translators for a given language.

[5] `github.com/lowerquality/gentle`

[6] The effectiveness of this filtering criterion was manually verified on random samples. More aggressive solutions will be explored for future releases of the corpus.

Tgt	#Talk	#Sent	Hours	src w	tgt w
De	2,093	234K	408	4.3M	4.0M
Es	2,564	270K	504	5.3M	5.1M
Fr	2,510	280K	492	5.2M	5.4M
It	2,374	258K	465	4.9M	4.6M
Nl	2,267	253K	442	4.7M	4.3M
Pt	2,050	211K	385	4.0M	3.8M
Ro	2,216	240K	432	4.6M	4.3M
Ru	2,498	270K	489	5.1M	4.3M

Table 2: Statistics for each section of MuST-C.

aligned audio using the XNMT tool (Neubig et al., 2018).[7]

Table 2 provides basic statistics for the 8 sections of the MuST-C corpus. Comparing the 4^{th} column with the numbers reported in Table 1, it is worth noting that, in terms of hours of transcribed/translated speech, each section is larger than any existing publicly available SLT resource.

3 Experiments

In this section we present two sets of experiments, which are respectively aimed to: *i)* empirically assess the quality of the MuST-C corpus (Section 3.3) and *ii)* compute baseline ASR, MT, and SLT results for future comparisons (Section 3.4).

In these experiments, the *audio-transcription* alignments of MuST-C are used to train and evaluate ASR models, *transcription-translation* alignments are used for the MT models, and *audio-translation* alignments are used for the SLT models.

3.1 ASR, MT and SLT Models

ASR and SLT. For our experiments in ASR and SLT we use the same neural architecture. This setting allows us to use the encoder of the ASR models to initialize the weights of the SLT encoders and achieve a faster convergence (Bansal et al., 2018). Our SLT architecture is a variant of the system proposed by Bérard et al. (2018), which we re-implemented in the fairseq toolkit (Gehring et al., 2017). The system relies on an attentional encoder-decoder model that takes in input sequences of audio features and outputs the target sequence at the character level. The encoder processes the input with two consecutive fully-connected layers to expand the size of the representation, followed by two 2D strided convolutional layers that reduce the sequence length. The output of the convolutions is then processed by three stacked LSTMs (Hochreiter and Schmidhuber, 1997). The decoder consists of a two-layered deep transition (Pascanu et al., 2014) LSTM with an attention network based on the general soft attention score (Luong et al., 2015). The final output of the decoder is a function of the concatenation of the LSTM output, the context vector and the previous-character embedding.

MT. For the MT experiments we use the open source version of ModernMT.[8] The system is based on the Transformer (Vaswani et al., 2017) architecture, which represents the state of the art in NMT (Bojar et al., 2018). The encoder consists of a stack of 6 layers, each containing a sequence of two sub-layers, a self-attention network based on multi-head attention, and a position-wise feed-forward layer. The decoder layers have an additional sub-layer: between the self attention and the position-wise feed-forward layer they have an encoder-decoder multi-head attention. All the sub-layers in both the encoder and decoder are preceded by layer normalization and are followed by residual connections.

3.2 Data Processing and Evaluation Metrics

In our experiments, texts are tokenized and punctuation is normalized. Furthermore, the English texts are lowercased, while the target language texts are split into characters still preserving the word boundaries. For MT, we segment the English words with the BPE algorithm (Sennrich et al., 2015) using a maximum of $30K$ merge operations. The output generation of all models is performed using beam search with a beam size of 5.

ASR performance is measured with word error rate (WER) computed on lower-cased, tokenized texts without punctuation. MT and SLT results are computed with BLEU (Papineni et al., 2002).

3.3 Experiment 1: Corpus Quality

As observed in Section 2, each section of MuST-C is larger than any other existing publicly available SLT corpus. The usefulness of a resource, however, is not only a matter of size but also of quality (in this case, the quality of the *audio-transcription-translation* alignments). For an empirical verification of this aspect, we experimented with two comparable datasets. One is

---

[7] `github.com/neulab/xnmt`

[8] `www.modernmt.eu`

the TED-derived English-German IWSLT18 corpus (Niehues et al., 2018), which is built following a pipeline that performs segment extraction and alignment based on time information (*i.e.* start and end position of each segment in the SubRip Text (SRT) files) instead of text-level alignments. The other is the English-German subset of MuST-C derived from the same TED Talks used to build the IWSLT18 corpus. On one side (MuST-C), the number of segments, their length, and the overall corpus quality depend on text-level alignments. On the other side (IWSLT18), they depend on matching time stamps. This strategy, however, has some drawbacks. First, as pointed out by (Niehues et al., 2018; Liu et al., 2018; Di Gangi et al., 2018), the use of time information brings some noise in the corpus. Second, it often results in utterance-level alignment (based on speakers' pauses in the original audio). Compared to sentence-level alignment, this level of granularity can be sub-optimal during model training (*e.g.* for MT and SLT, learning from complete sentences is easier than learning from phrases). Finally, time information about the recorded speech is not always available: bypassing this need would make the method replicable on other data (not only TED-like).

Though initialized with the same set of $1,619$ talks, the two pipelines produce different corpora. As shown in Table 3, our approach filters out 58 entire talks ($\sim$3.6% of the total) but the final number of segments, their corresponding audio duration and their average length (in words) are larger.

Corpus	#Talk	#Sent	Hours	src w	tgt w
IWSLT18	1,619	176K	280	2.7M	2.5M
MuST-C	1,561	179K	313	3.3M	3.1M

Table 3: Statistics of the English-German corpora created by applying the IWSLT18 and MuST-C pipelines to the same initial set of $1,619$ TED Talks.

Each corpus was divided into training, development and test. Development and test contain segments from randomly selected common talks (*i.e.* those preserved by the MuST-C pipeline). Their size is respectively $2.3K$ (from 28 talks) and $2.1K$ segments (from 26 talks). The test portions were concatenated to create a balanced test set ($4.2K$ segments) containing half of the instances from the IWSLT18 corpus and half from MuST-C. The remaining material was used to separately train ASR, MT and SLT models on homogeneous data from either of the two corpora (*i.e.* three systems per corpus). All the systems are evaluated on the common test set.

Training set	ASR (↓)	MT (↑)	SLT (↑)
IWSLT18	42.15	24.90	8.94
MuST-C	32.05	25.46	12.25

Table 4: Performance of ASR, MT and SLT systems trained with En-De IWSLT18 and MuST-C data.

Tgt	ASR (↓)	MT (↑)	SLT (↑)
De	27.00	28.09	12.93
Es	26.61	34.16	18.20
Fr	25.81	42.23	22.29
It	26.38	30.40	14.95
Nl	26.55	33.43	18.20
Pt	28.00	32.44	17.10
Ro	27.61	28.16	13.35
Ru	26.97	18.30	7.22

Table 5: Baseline ASR, MT and SLT results for each language direction.

Table 4 shows that the models trained on MuST-C data achieve better results on the balanced test set in all the three tasks. In particular: *i)* a reduction of 10.1 WER points in ASR indicates a higher quality of *audio-transcription* alignments, *ii)* a BLEU increase of 0.56 points in MT indicates a similar quality for *transcription-translation* alignments, and *iii)* a BLEU increase of 3.31 points in SLT indicates a higher quality of *audio-translation* alignments. We consider these results as evidence of the reliability of our corpus creation methodology. Being the same for all the language pairs, we expect this procedure to end up in comparable quality for all the 8 sections of MuST-C.

3.4 Experiment 2: Baseline Results

We finally present baseline results computed, for all the three tasks, on each section of MuST-C. Also for these experiments, development and test data are created with segments from talks that are common to all the languages. Their size is respectively $1.4K$ (from 11 talks) and $2.5K$ segments (from 27 talks). The remaining data (of variable size depending on the language pairs) are used for training. For the sake of replicability, these splits are preserved in the released version of MuST-C.

The results in Table 5 lead to the following observations. First, though not directly comparable since they are computed on different test sets, English-German results are in line (actually higher, since they are produced by models built on larger training data) with those presented in Section 3.3. This indicates that the level of quality observed in the previous experiments with a subset of the training data is preserved by the whole material released for this language pair. Second, looking at the other language pairs, ASR, MT and SLT results are comparable with the English-German scores. Besides normal fluctuations in the optimization of the neural models, performance differences are coherent with: *i)* the relative difficulty of each target language (*e.g.* Russian is more difficult due to high inflection) and *ii)* the variable quantity of training data available (*e.g.* French has the largest training set, see Table 2). Overall, these explainable differences suggest that our corpus creation methodology yields homogeneous quality for all the languages covered by MuST-C.

4 Conclusion and Future Work

We presented MuST-C, a *Mu*ltilingual *S*peech *T*ranslation *C*orpus built to address the need of resources for training data-hungry neural SLT models. To the best of our knowledge, to date MuST-C is the largest publicly available corpus of this kind. In its current version, it comprises the English transcription and the translations into 8 target languages of at least 385 hours of speech (up to 504) per language. Thanks to a scalable corpus creation procedure initialized with constantly expanding TED talks data, future extensions will increase the coverage of the already present target languages and introduce new ones.

MuST-C is released under a Creative Commons license, Attribution - Non Commercial - No Derivatives (CC BY NC ND 4.0 International), and is freely downloadable at `mustc.fbk.eu`

Acknowledgments

The authors gratefully acknowledge NVIDIA Corporation for the donation of the Tesla K80 and GeForce GTX 1080 Ti GPUs used for this research.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied Multitask Learning for Neural Speech Translation. In *Proceedings of NAACL-HLT 2018*, pages 82–91, New Orleans, Louisiana.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Pre-Training on High-Resource Speech Recognition Improves Low-Resource Speech-to-Text Translation. *arXiv preprint arXiv:1809.01431*.

Claudio Bendazzoli and Annalisa Sandrelli. 2005. An Approach to Corpus-based Interpreting Studies: Developing EPIC (European Parliament Interpreting Corpus). In *Proceedings of the EU-High-Level Scientific Conference Series MuTra 2005Challenges of Multidimensional Translation*, pages 149–160, Saarbrücken, Germany.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-End Automatic Speech Translation of Audiobooks. In *Proceedings of ICASSP 2018*, pages 6224–6228, Calgary, Alberta, Canada.

Alexandre Bérard, Olivier Pietquin, Laurent Besacier, and Christophe Servan. 2016. Listen and Translate: A Proof of Concept for End-to-End Speech-to-Text Translation. In *Proceedings of the NIPS Workshop on end-to-end learning for speech and audio processing*, Barcelona, Spain.

Marcely Zanon Boito, Antonios Anastasopoulos, Marika Lekakou, Aline Villavicencio, and Laurent Besacier. 2018. A Small Griko-Italian Speech Translation Corpus. *CoRR*, abs/1807.10740.

Ondrej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 Conference on Machine Translation (WMT18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 272–307, Belgium, Brussels.

Fabienne Braune and Alexander Fraser. 2010. Improved Unsupervised Sentence Alignment for Symmetrical and Asymmetrical Parallel Corpora. In *Proceedings of COLING 2010*, pages 81–89, Beijing, China.

Chung-Cheng Chiu, Tara Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, et al. 2018. State-of-the-art Speech Recognition With Sequence-to-Sequence Models. In *Proceedings of ICASSP 2018*, pages 4774–4778, Calgary, Alberta, Canada.

Mattia Antonino Di Gangi, Roberto Dessì, Roldano Cattoni, Matteo Negri, and Marco Turchi. 2018. Fine-tuning on Clean Data for End-to-End Speech Translation: FBK@ IWSLT 2018. In *Proceedings of IWSLT 2018*, Bruges, Belgium.

Christian Federmann and William D Lewis. 2016. Microsoft Speech Language Translation (MSLT) Corpus: The IWSLT 2016 Release for English, French and German. In *Proceedings of IWSLT 2016*, Seattle, USA.

Christian Federmann and William D Lewis. 2017. The Microsoft Speech Language Translation (MSLT) Corpus for Chinese and Japanese: Conversational Test data for Machine Translation and Speech

Recognition. In *Proceedings of the 16th Machine Translation Summit*, Nagoya, Japan.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of ICML 2017*, pages 1243–1252, Sydney, Australia.

Pierre Godard, Gilles Adda, Martine Adda-Decker, Juan Benjumea, Laurent Besacier, et al. 2017. A Very Low Resource Language Speech Corpus for Computational Language Documentation Experiments. *arXiv preprint arXiv:1710.03501*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. In *Proceedings of ICASSP 2018*, pages 6645–6649, Vancouver, BC, Canada.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8).

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron-J. Weiss, Yuan Cao, Chung-Cheng Chiu, Stella-Laurenzo Ari, and Yonghui Wu. 2018. Leveraging Weakly Supervised Data to Improve End-to-End Speech-to-Text Translation. *ArXiv e-prints arXiv:1811.02050*.

Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting Librispeech with French Translations: A Multimodal Corpus for Direct Speech Translation Evaluation. In *Proceedings of LREC 2018*, Miyazaki, Japan.

Dan Liu, Junhua Liu, Wu Guo, Shifu Xiong, Zhiqiang Ma, Rui Song, Chongliang Wu, and Quan Liu. 2018. The USTC-NEL Speech Translation system at IWSLT 2018. In *Proceedings of IWSLT 2018*, Bruges, Belgium.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP 2015*, pages 1412–1421, Lisbon, Portugal.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, et al. 2018. XNMT: The eXtensible Neural Machine Translation Toolkit. In *Proceedings of AMTA 2018*, pages 185–192, Boston, MA.

Jan Niehues, Roldano Cattoni, Sebastian Stüker, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. The IWSLT 2018 Evaluation Campaign. In *Proceedings of IWSLT 2018*, Bruges, Belgium.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL 2002*, Philadelphia, PA, USA.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to Construct Deep Recurrent Neural Networks. In *Proceedings of ICLR 2014*, Banff, Canada.

Matthias Paulik and Alex Waibel. 2009. Automatic Translation from Parallel Speech: Simultaneous Interpretation as MT Training Data. In *Proceedings of ASRU 2009*, pages 496–501, Merano, Italy.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus. In *Proceedings of IWSLT 2013*, Heidelberg, Germany.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, K. Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. 2011. The Kaldi Speech Recognition Toolkit. In *Proceedings of ASRU 2011*, pages 1–4, Big Island, Hawaii, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909*.

Hiroaki Shimizu, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Collection of a Simultaneous Translation Corpus for Comparative Analysis. In *Proceedings of LREC 2014*, Reykjavik, Iceland.

Sebastian Stüker, Florian Kraft, Christian Mohr, Teresa Herrmann, Eunah Cho, and Alex Waibel. 2012. The KIT Lecture Corpus for Speech Translation. In *Proceedings of LREC-2012*, Istanbul, Turkey.

Hitomi Tohyama, Shigeki Matsubara, Nobuo Kawaguchi, and Yasuyoshi Inagaki. 2005. Construction and utilization of bilingual speech corpus for simultaneous machine interpretation research. In *Proceedings of INTERSPEECH*, pages 1585–1588, Lisbon, Portugal.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NIPS 2017*, pages 5998–6008, Long Beach, CA, USA.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-Sequence Models Can Directly Translate Foreign Speech. In *Proceedings of Interspeech 2017*, pages 2625–2629, Stockholm, Sweden.

Michael Melese Woldeyohannis, Laurent Besacier, and Million Meshesha. 2017. A Corpus for Amharic-English Speech Translation: the Case of Tourism Domain. In *Proceedings of ICT4DA 2017*, pages 129–139, Bahir Dar, Ethiopia.

Contextualization of Morphological Inflection

Ekaterina Vylomova[ə] **Ryan Cotterell**[ʔ,fi] **Timothy Baldwin**[ə] **Trevor Cohn**[ə] and **Jason Eisner**[ʔ]

[ə]School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
[fi]The Computer Laboratory, University of Cambridge, Cambridge, UK
[ʔ]Department of Computer Science, Johns Hopkins University, Baltimore, USA
`{vylomovae,tbaldwin,tcohn}@unimelb.edu.au`
`rdc42@cam.ac.uk jason@cs.jhu.edu`

Abstract

Critical to natural language generation is the production of correctly inflected text. In this paper, we isolate the task of predicting a fully inflected sentence from its partially lemmatized version. Unlike traditional morphological inflection or surface realization, our task input does not provide "gold" tags that specify what morphological features to realize on each lemmatized word; rather, such features must be inferred from sentential context. We develop a neural hybrid graphical model that explicitly reconstructs morphological features before predicting the inflected forms, and compare this to a system that directly predicts the inflected forms without relying on any morphological annotation. We experiment on several typologically diverse languages from the Universal Dependencies treebanks, showing the utility of incorporating linguistically-motivated latent variables into NLP models.

1 Introduction

NLP systems are often required to generate grammatical text, e.g., in machine translation, summarization, dialogue, and grammar correction. One component of grammaticality is the use of contextually appropriate closed-class morphemes. In this work, we study **contextual inflection**, which has been recently introduced in the CoNLL-SIGMORPHON 2018 shared task (Cotterell et al., 2018) to directly investigate context-dependent morphology in NLP. There, a system must inflect partially lemmatized tokens in sentential context. For example, in English, the system must reconstruct the correct word sequence *two cats are sitting* from partially lemmatized sequence *two _cat_ are sitting*. Among other things, this requires: (1) identifying *cat* as a noun in this context, (2) recognizing that *cat* should be inflected as plural to agree with the nearby verb and numeral, and (3) realizing this inflection as the suffix *s*. Most past work in supervised computational morphology—including the previous CoNLL-SIGMORPHON shared tasks on morphological reinflection (Cotterell et al., 2017)—has focused mainly on step (3) above.

As the task has been introduced into the literature only recently, we provide some background. Contextual inflection amounts to a highly constrained version of language modeling. Language modeling predicts all words of a sentence from scratch, so the usual training and evaluation metric—perplexity—is dominated by the language model's ability to predict *content*, which is where most of the uncertainty lies. Our task focuses on just the ability to reconstruct certain missing parts of the sentence—inflectional morphemes and their orthographic realization. This refocuses the modeling effort from semantic coherence to morphosyntactic coherence, an aspect of language that may take a back seat in current language models (see Linzen et al., 2016; Belinkov et al., 2017). Contextual inflection does not perfectly separate grammaticality modeling from content modeling: as illustrated in Tab. 1, mapping *two cats _be_ sitting* to the fully-inflected *two cats were sitting* does not require full knowledge of English grammar—the system does not have to predict the required word order nor the required auxiliary verb *be*, as these are supplied in the input. Conversely, this example does still require predicting some content—the semantic choice of past tense is *not* given by the input and must be guessed by the system.[1]

The primary contribution of this paper is a novel structured neural model for contextual inflection. The model first predicts the sequence of morphological tags from the partially lemmatized sequence and, then, it uses the predicted tag and lemma to inflect the word. We use this model

---

[1] This morphological feature is *inherent* in the sense of Booij (1996).

Proceedings of NAACL-HLT 2019, pages 2018–2024
Minneapolis, Minnesota, June 2 - June 7, 2019.

Context:	`two`	`cats`	`--`	`sitting`
Lemmata:	`two`	`cat`	`be`	`sit`

Tags:	`POS=NUM`	`POS=NOUN`
		`Num=Plur`
			`--`	`POS=VERB`
				`Tense=Pres`
				`VerbForm=Part`

Target:	`two`	`cats`	`were`	`sitting`

Table 1: Example data entry: the target word *be* should be properly inflected into *were* to fit the sentential context.

to evince a simple point: models are better off jointly predicting morphological tags from context than directly learning to inflect lemmata from sentential context. Indeed, none of the participants in the 2018 shared task jointly predicted tags with the inflected forms. Comparing our new model to several competing systems, we show our model has the best performance on the majority of languages. We take this as evidence that predicting morphological tags jointly with inflecting is a better method for this task. Furthermore, we provide an analysis discussing the role of morphological complexity in model performance.

2 Joint Tagging and Inflection

Given a language, let $\mathcal{M}$ be a set of morphological tags in accordance with the Universal Dependencies annotation (Nivre et al., 2016). Each $m \in \mathcal{M}$ has the form $m = \langle t, \sigma \rangle$, where t is a part of speech, and the slot σ is a set of attribute–value pairs that represent morphosyntactic information, such as number, case, tense, gender, person, and others. We take $t \in \mathcal{T}$, the set of universal parts of speech described by Petrov et al. (2012). A sentence consists of a finite word sequence $\mathbf{w}$ (we use boldface for sequence variables). For every word w_i in the sequence, there is a corresponding analysis in terms of a morphological tag $m_i \in \mathcal{M}$ and a lemma ℓ_i. In general, w_i is determined by the pair $\langle \ell_i, m_i \rangle$.[2] Using this notation, Cotterell et al. (2018)'s shared task is to predict a sentence $\mathbf{w}$ from its partially lemmatized form $\boldsymbol{\ell}$, inferring $\mathbf{m}$ as an intermediate latent variable. Our dataset (§3) has all three sequences for each sentence.

2.1 A Structured Neural Model

Consider an extreme case when *all* words are lemmatized.[3] We introduce a structured neural model for contextual inflection, as follows:

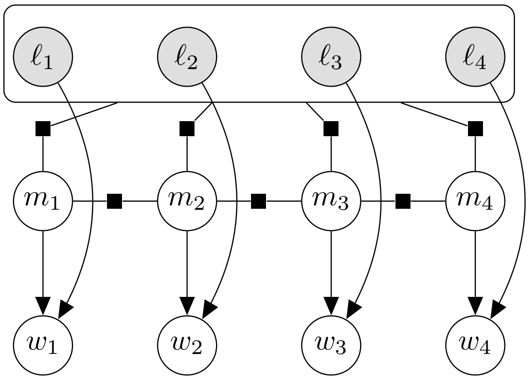


Figure 1: Our structured neural model shown as a hybrid (directed–undirected) graphical model. We omitted several arcs for convenience; namely, every morphological tag m_i depends on the entire sequence $\boldsymbol{\ell}$.

$$p(\mathbf{w}, \mathbf{m} \mid \boldsymbol{\ell}) = \left(\prod_{i=1}^{n} p(w_i \mid \ell_i, m_i) \right) p(\mathbf{m} \mid \boldsymbol{\ell}) \tag{1}$$

In other words, the distribution is over interleaved sequences of one-to-one aligned inflected words and morphological tags, conditioned on a lemmatized sequence, all of length n. This distribution is drawn as a hybrid (directed–undirected) graphical model (Koller and Friedman, 2009) in Fig. 1. We define the two conditional distributions in the model in §2.2 and §2.3, respectively.

2.2 A Neural Conditional Random Field

The distribution $p(\mathbf{m} \mid \boldsymbol{\ell})$ is defined to be a conditional random field (CRF; Lafferty et al., 2001). In this work, our CRF is a conditional distribution over morphological taggings of an input sequence. We define this conditional distribution as

$$p(\mathbf{m} \mid \boldsymbol{\ell}) = \frac{1}{Z(\boldsymbol{\ell})} \prod_{i=1}^{n} \psi\left(m_i, m_{i-1}, \boldsymbol{\ell}\right) \tag{2}$$

where $\psi(\cdot, \cdot, \cdot) \geq 0$ is an arbitrary potential, $Z(\boldsymbol{\ell})$ normalizes the distribution, and m_0 is a distinguished start-of-sequence symbol.

---

[2] Although w_i can sometimes be computed by concatenating ℓ_i with m_i-specific affixes, it can also be irregular.

[3] In case of partially lemmatized sequence we still train the model to predict the tags over the entire sequence, but evaluate it only for lemmatized slots.

In this work, we opt for a recurrent neural potential—specifically, we adopt a parameterization similar to the one given by Lample et al. (2016). Our potential ψ is computed as follows. First, the sequence $\boldsymbol{\ell}$ is encoded into a sequence of word vectors using the strategy described by Ling et al. (2015): word vectors are passed to a bidirectional LSTM (Graves et al., 2005), where the corresponding hidden states are concatenated at each time step. We simply refer to the hidden state $\mathbf{h}_i \in \mathbb{R}^d$ as the result of said concatenation at the i-th step. Using $\mathbf{h}_i$, we can define the potential function as $\psi(m_i, m_{i-1}) = \exp\left(A_{m_i,m_{i-1}} + \mathbf{o}_{m_i}^\top \mathbf{h}_i\right)$, where $A_{m_i,m_{i-1}}$ is a transition weight matrix and $\mathbf{o}_{m_i} \in \mathbb{R}^d$ is a morphological tag embedding; both are learned.

2.3 The Morphological Inflector

The conditional distribution $p(w_i \mid \ell_i, m_i)$ is parameterized by a neural encoder–decoder model with hard attention from Aharoni and Goldberg (2017). The model was one of the top performers in the 2016 SIGMORPHON shared task (Cotterell et al., 2016); it achieved particularly high accuracy in the low-resource setting. Hard attention is motivated by the observation that alignment between the input and output sequences is often monotonic in inflection tasks. In the model, the input lemma is treated as a sequence of characters, and encoded using a bidirectional LSTM (Graves and Schmidhuber, 2005), to produce vectors $\mathbf{x}_j$ for each character position j. Next the word $w_i = \mathbf{c} = c_1 \cdots c_{|w_i|}$ is generated in a decoder character-by-character:

$$p(c_j \mid \mathbf{c}_{<j}, l_i, m_i) = \text{softmax}\left(\mathbf{W} \cdot \phi(\mathbf{z}_1, \ldots, \mathbf{z}_j) + \mathbf{b}\right) \quad (3)$$

where $\mathbf{z}_j$ is the concatenation of the current attended input $\mathbf{x}_j$ alongside morphological features, m_i, and an embedding of the previously generated symbol c_{j-1}; and finally ϕ is an LSTM over the sequence of $\mathbf{z}_j$ vectors. The decoder additionally predicts a type of operation.[4] The distribution in Eq. (3), strung together with the other conditionals, yields a joint distribution over the entire character sequence:

$$p(\mathbf{c} \mid \ell_i, m_i) = \prod_{j=1}^{|w_i|} p(c_j \mid \mathbf{c}_{<j}, \ell_i, m_i) \quad (4)$$

For instance, to map the lemma *talk* to its past form *talked*, we feed in `POS=V;Tense=PAST <w> t a l k </w>` and train the network to output `<w> t a l k e d </w>`, where we have augmented the orthographic character alphabet Σ with the feature–attribute pairs that constitute the morphological tag m_i.

2.4 Parameter Estimation and Decoding

We optimize the log-likelihood of the training data with respect to all model parameters. As Eq. (1) is differentiable, this is achieved with standard gradient-based methods. For decoding we use a greedy strategy where we first decode the CRF, that is, we solve the problem $\mathbf{m}^\star = \text{argmax}_{\mathbf{m}} \log p(\mathbf{m} \mid \boldsymbol{\ell})$, using the Viterbi (1967) algorithm. We then use this decoded $\mathbf{m}^\star$ to generate forms from the inflector. Note that finding the one-best string under our neural inflector is intractable, and for this reason we use greedy search.

3 Experiments

Dataset. We use the Universal Dependencies v1.2 dataset (Nivre et al., 2016) for our experiments. We include all the languages with information on their lemmata and fine-grained grammar tag annotation that also have `fasttext` embeddings (Bojanowski et al., 2017), which are used for word embedding initialization.[5]

Evaluation. We evaluate our model's ability to predict: (i) the correct morphological tags from the lemma context, and (ii) the correct inflected forms. As our evaluation metric, we report 1-best accuracy for both tags and word form prediction.

Configuration. We use a word and character embedding dimensionality of 300 and 100, respectively. The hidden state dimensionality is set to 200. All models are trained with Adam (Kingma and Ba, 2014), with a learning rate of 0.001 for 20 epochs.

Baselines. We use two baseline systems: (1) the CoNLL–SIGMORPHON 2018 subtask 2 neural encoder–decoder with an attention mechanism ("SM"; Cotterell et al. (2018)), where the encoder represents a target form context as a concatenation of its lemma, its left and right word forms, their

---

[4] The model can be viewed as a transition system trained over aligned character-level strings to learn sequences of operations (`write` or `step`).

[5] We also choose mainly non-Wikipedia datasets to reduce any possible intersection with the data used for the *FastText* model training

Language	tag	form				
	JOINT	GOLD	JOINT	DIRECT	SM	CPH
Bulgarian	81.6	91.9	**78.8**	71.5	77.1	76.9
English	89.6	95.6	**90.4**	86.8	86.5	86.7
Basque	66.6	82.2	61.1	59.7	**61.2**	60.2
Finnish	66.0	86.5	**59.3**	51.2	56.6	56.4
Gaelic	68.3	84.5	**69.5**	64.5	68.9	66.9
Hindi	85.3	88.3	81.4	85.4	86.8	**87.5**
Italian	92.3	85.1	80.4	85.2	88.7	**90.5**
Latin	82.6	89.7	**75.7**	71.4	74.2	74.9
Polish	71.9	96.1	**74.8**	61.8	72.4	70.2
Swedish	81.9	96.0	**82.5**	75.4	78.4	80.9

Table 2: Accuracy of the models for various prediction settings. **tag** refers to tag prediction accuracy, and **form** to form prediction accuracy. Our model is JOINT; GOLD denotes form prediction conditioned on gold target morphological tags; the other columns are baseline methods.

lemmata and tag representations, and then the decoder generates the target inflected form character-by-character; and (2) a monolingual version of the best performing system of the shared task ("CPH"; Kementchedjhieva et al. (2018)) that augments the above encoder–decoder with full (sentence-level) left and right contexts (comprising of forms, their lemmata and morphological tags) as well as predicts morphological tags for a target form as an auxiliary task.[6] In both cases, the hyperparameters are set as described in Cotterell et al. (2018). We additionally evaluate the SIGMORPHON baseline system on prediction of the target form without any information on morphological tags ("DIRECT").

4 Results and Discussion

Tab. 2 presents the accuracy of our best model across all languages.[7] Below we highlight two main lessons from our error analysis that apply to a wider range of generation tasks, e.g., machine translation and dialog systems.

Directly Predicting Morphology. Tab. 2 indicates that all systems that make use of morphological tags outperform the DIRECT baseline on most languages. The comparison of our hybrid model with latent morphological tags to the direct form generation baseline in SM suggests that we should be including linguistically-motivated latent variables into models of natural language generation. We observe in Tab. 2 that predicting the tag together with the form (joint) often improves performance. The most interesting comparison here is with the multi-task CPH method, which includes morphology into the model without joint modeling; our model achieves higher results on 7/10 languages.

Morphological Complexity Matters. We observed that for languages with rich case systems, e.g., the Slavic languages (which exhibit a lot of fusion), the agglutinative Finno-Ugric languages, and Basque, performance is much worse. These languages present a broader decision space and often require inferring which morphological categories need to be in agreement in order to make an accurate prediction. This suggests that generation in languages with more morphological complexity will be a harder problem for neural models to solve. Indeed, this problem is under-explored, as the field of NLP tends to fixate on generating English text, e.g., in machine translation or dialogue system research.

Error Analysis. We focused error analysis on prediction of agreement categories. Our analysis of adjective–noun agreement category prediction suggests that our model is able to infer adjective number, gender, and case from its head noun. Verb gender, which appears only in the past tense of many Slavic languages, seems to be harder to predict. Given that the linear distance between the subject and the verb may be longer, we suspect the network struggles to learn longer-distance dependencies, consistent with the findings of Linzen et al. (2016). Overall, automatic inference of agreement categories is an interesting problem that deserves more attention, and we leave it for future work.

We also observe that most uncertainty comes from morphological categories such as noun number, noun definiteness (which is expressed morphologically in Bulgarian), and verb tense, all of which are inherent (Booij, 1996)[8] and typically cannot be predicted from sentential context if they do not participate in agreement.[9] On the other hand, aspect, although being closely related to tense, is well-predicted since it is mainly expressed as a separate lexeme. But, in general, it is still problematic to make a prediction in languages where aspect is morphologically marked or highly mixed with

---

[6] It has been shown to improve the model's performance.

[7] The accuracy numbers are on average higher than the ones achieved in terms of the CoNLL–SIGMORPHON 2018 subtask 2 since we did not filter out tokens that are typically not inflected (such as articles or prepositions).

[8] Such categories exist in most languages that exhibit some degree of morphological complexity.

[9] Unless there is no strong signal within a sentence such as *yesterday*, *tomorrow*, or *ago* as in the case of tense.

tense as in Basque.

We additionally compared 1-best and 10-best predictions for tags. Most mispredictions existing in 1-best lists are due to inherent categories mentioned above (that allow multiple plausible options that can fit the sentential context). Indeed, the problem is usually solved by allowing system to output 10-best lists. There, precision@10 is on average 8 points higher than precision@1.

Finally, our analysis of case category prediction on nouns shows that more common cases such as the nominative, accusative, and genitive are predicted better, especially in languages with fixed word order. On the other hand, cases that appear less frequently and on shifting positions (such as the instrumental), as well as those not associated with specific prepositions, are less well predicted. In addition, we evaluated the model's performance when *all* forms are replaced by their corresponding lemmata (as in *two cat be sit*). For freer word order languages such as Polish or Latin, we observe a substantial drop in performance because most information on inter-word relations and their roles (expressed by means of case system) is lost.

5 Related Work

The primary evaluation for most contemporary language and translation modeling research is perplexity, BLEU (Papineni et al., 2002), or METEOR (Banerjee and Lavie, 2005). Undoubtedly, such metrics are necessary for extrinsic evaluation and comparison. However, relatively few studies have focused on intrinsic evaluation of the model's mastery of grammaticality. Recently, Linzen et al. (2016) investigated the ability of an LSTM language model to capture sentential structure, by evaluating subject–verb agreement with respect to number, and showed that under strong supervision, the LSTM is able to approximate dependencies.

Taking it from the other perspective, a truer measure of grammatical competence would be a task of mapping a meaning representation to text, where the meaning representation specifies all necessary semantic content—content lemmata, dependency relations, and "inherent" closed-class morphemes (semantic features such as noun number, noun definiteness, and verb tense)—and the system is to realize this content according to the morphosyntactic conventions of a language, which means choosing word order, agreement morphemes, function words, and the surface forms of all words. Such tasks have been investigated to some extent—generating text from tectogrammatical trees (Hajic et al., 2002; Ptáček and Žabokrtský, 2006) or from an AMR graph (Song et al., 2017). Belz et al. (2011) organized a related surface realization shared task on mapping unordered and uninflected dependency trees to properly ordered inflected sentences. The generated sentences were afterwards assessed by human annotators, making the task less scalable and more time consuming. Although our task is not perfectly matched to grammaticality modeling, the upside is that it is a "lightweight" task that works directly on text. No meaning representation is required. Thus, training and test data in any language can be prepared simply by lemmatizing a naturally occurring corpus.

Finally, as a morphological inflection task, the form generation task is closely related to previous SIGMORPHON shared tasks (Cotterell et al., 2016, 2017). There, most neural models achieve high accuracy on many languages at type-level prediction of the form from its lemma and slot. The current task is more challenging in that the model has to perform token-level form generation and inherently infer the slot from the contextual environment. Our findings are in line with those from the CoNLL-SIGMORPHON 2018 shared task (Cotterell et al., 2018) and provide extra evidence of the utility of morphosyntactic features.

6 Conclusion

This work proposed a method for contextual inflection using a hybrid architecture. Evaluation over several diverse languages showed consistent improvements over state of the art. Our analysis demonstrated that the contextual inflection can be a highly challenging task, and the inclusion of morphological features prediction is an important element in such a system. We also highlighted two types of morphological categories, contextual and inherent, in which the former relies on agreement and the latter comes from a speaker's intention.

Acknowledgements

We thank all anonymous reviewers for their comments. The first author would like to acknowledge the Google PhD fellowship. The second author would like to acknowledge a Facebook Fellowship.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2004–2015.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 861–872.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The First Surface Realisation Shared Task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Geert Booij. 1996. Inherent versus contextual inflection and the split morphology hypothesis authors. In *Yearbook of Morphology 1995*, pages 1–16. Springer, Netherlands.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gèraldine Walther, Ekaterina Vylomova, Arya D McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 Shared Task: Universal morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 Shared Task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 Shared Task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications ICANN 2005*, pages 799–804.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jan Hajic, Martin Cmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, et al. 2002. Natural language generation in the context of machine translation. In *Summer workshop final report, Johns Hopkins University*.

Yova Kementchedjhieva, Johannes Bjerva, and Isabelle Augenstein. 2018. Copenhagen at CoNLL–SIGMORPHON 2018: Multilingual inflection in context with explicit morphosyntactic decoding. *CoNLL–SIGMORPHON*, page 93.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of LREC 2016*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*.

Jan Ptáček and Zdeněk Žabokrtský. 2006. Synthesis of Czech sentences from tectogrammatical trees. In *International Conference on Text, Speech and Dialogue*, pages 221–228.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions Information Theory*, 13(2):260–269.

A Robust Abstractive System for Cross-Lingual Summarization

Jessica Ouyang and **Boya Song** and **Kathleen McKeown**
Deptartment of Computer Science
Columbia University
New York, NY 10027
{ouyangj, kathy}@cs.columbia.edu, bs3065@columbia.edu

Abstract

We present a robust neural abstractive summarization system for cross-lingual summarization. We construct summarization corpora for documents automatically translated from three low-resource languages, Somali, Swahili, and Tagalog, using machine translation and the New York Times summarization corpus. We train three language-specific abstractive summarizers and evaluate on documents originally written in the source languages, as well as on a fourth, unseen language: Arabic. Our systems achieve significantly higher fluency than a standard copy-attention summarizer on automatically translated input documents, as well as comparable content selection.

1 Introduction

Cross-lingual summarization is a little-explored task combining the difficulties of automatic summarization with those of machine translation. The goal is to summarize in one language a document available only in another language. Wan et al. (2010) describe two approaches: summarize then translate, and translate then summarize. They argue that summarize-then-translate is preferable to avoid both the computational expense of translating more sentences and sentence extraction errors caused by incorrect translations.

However, summarize-then-translate can only be used when the source language is high-resource (Wan et al. used English as the source, for example); if the source language is one of the thousands of low-resource languages in the world, there are no summarization corpora available. Language-independent techniques, such as TextRank (Mihalcea), might be used, but there may be serious difficulties in their application, such as morphologically rich languages that render token-based similarity measures useless. In such a case, translate-then-summarize is the only possible approach.

We address this scenario through the development of a neural abstractive summarization system that fluently summarizes potentially disfluent, automatically-translated documents by generating short, simple phrases to replace awkward input phrases resulting from difficult to translate source documents. Our novel combination of existing building block systems results in a summarization solution that can be easily applied to new low-resource languages. We use machine translation on the New York Times annotated corpus of document/summary pairs to create summarization corpora for documents automatically translated from three low-resource languages, Somali, Swahili, and Tagalog. We use these corpora to train cross-lingual summarizers for these source languages, with English as the target. We also evaluate our systems on a fourth source language, Arabic. Our experiments show that our abstractive summarizers produce more fluent English summaries from automatically-translated documents, and that this improvement generalizes across source languages.

Our main contributions are as follows:

- We create summarization corpora for automatically translated Somali, Swahili, and Tagalog documents: noisy English input documents paired with clean English reference summaries.
- We present a method for producing cross-lingual summarization systems for low resource languages where no summarization corpora currently exist, providing a potential summarization solution for thousands of such languages.
- Our novel approach of training on noisy input with clean references outperforms a standard copy-attention abstractive summarizer on real-world Somali, Swahili, and Tagalog documents.
- Our evaluation on Arabic documents demonstrates that our robust abstractive summarizers

Proceedings of NAACL-HLT 2019, pages 2025–2031
Minneapolis, Minnesota, June 2 - June 7, 2019.

in the editor: why did president clinton continue to praise a program on welfare-to-work that failed in half of those assigned? in his comments, he praised the consultation of the community of kansas city, but was advised by gary j. stangler, director of the department of social service of missouri, which half of the participants failed. where are these people helping each other when the government cut them? back to the pantry of food. bad news, mr. president. the charity of the community will not help everyone who will come to us for help. glenn classic valley park, mo.

Figure 1: A synthetic noisy English article from the Tagalog NYT training set.

generalize to unseen languages.

2 Related Work

Cross-Lingual Summarization. Orăsan and Chiorean (2008) extractively summarized Romanian news articles and automatically translated the summaries into English. Their experiments showed that the poor quality of the translations turned reasonable Romanian summaries into barely legible English ones.

The most extensively investigated source-target language pair is English-to-Chinese. Wan et al. (2010) used a predicted translation quality score as a feature in extracting sentences for their summaries. Wan (2011) translated the English sentences into Chinese and represented sentences in the extraction stage by both the original English and the Chinese translation. Yao et al. (2015) scored aligned phrases from the original English documents and the Chinese translations to perform sentence extraction and compression based on both salience and translation quality. Zhang et al. (2016) parsed the original English documents into predicate-argument structures that were aligned with their Chinese translations and generated the summary from these structures. Finally, Wan et al. (2018) experimented with extracting and ranking multiple candidate summaries.

Abstractive Summarization. Rush et al. (2015) presented the first neural abstractive summarization model, a convolutional neural network encoder and feed-forward network decoder with attention, which learned to generate news headlines from the lead sentences of their articles; Chopra et al. (2016) extended their work using a recurrent network for the decoder. Nallapati et al. (2016) improved on the RNN encoder-decoder with attention model by adding linguistically-motivated part of speech and named entity type embeddings, as well as a pointer-network (Vinyals et al., 2015) to allow copying of rare or out-of-vocabulary words from the input document. In this work, we use See et al.'s (2017) definition of the pointer-generator network, which adds a coverage vector and coverage penalty to prevent repetition in generated words.

The New York Times annotated corpus (Sandhaus, 2008) was first used for neural abstractive summarization by Paulus et al. (2018), who used attention over the decoder's previous predictions to both prevent repetition and to allow for coherent longer summaries. Celikyilmaz et al. (2018) also used the New York Times corpus, training multiple, collaborating encoders to encode long documents one paragraph at a time.

3 Data

We use the New York Times (hereafter *NYT*) summarization corpus (Sandhaus, 2008), consisting of 650k articles and their human-written abstractive summaries. We follow the train/test/validation split and preprocessing steps used by Paulus et al. (2018), with one exception: we do not anonymize named entities. We first translate 112k articles from the NYT corpus into each of our three low-resource languages, Somali, Swahili, and Tagalog, using neural machine translation. Of the 112k articles, 100k are taken from the training set, 6k from validation, and 6k from test. We then translate the articles back into noisy English, again using neural machine translation. Figure 1 shows an example noisy English article.

We pair each noisy English article with the clean English reference summary corresponding to the clean English article that generated it. Thus our abstractive summarization model learns to take a "bad" English input document with translation errors and disfluencies and produce a "good" English summary. For simplicity, we refer to the corpus created by translating into Somali and back as the *Somali NYT corpus*, and similarly with Swahili and Tagalog, but all three corpora are in (noisy) English, not Somali, Swahili, or Tagalog.

4 Models

4.1 Machine Translation

We use neural machine translation systems built on the Marian framework (Junczys-Dowmunt

	BLEU	
Language	from English	to English
Somali	21.8	29.4
Swahili	44.5	37.8
Tagalog	37.2	36.2

Table 1: Neural machine translation performance.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	**48.26**	29.30	36.81
Paulus	47.03	30.51	**43.27**
Celikyilmaz	48.08	**31.19**	42.33

Table 2: Baseline summarizer performance.

et al., 2018) to translate the NYT corpus into Somali, Swahili, and Tagalog, and back to English. The systems were developed at the University of Edinburgh and were trained on a mix of clean, human-curated parallel data (about 23k sentences for Somali and Swahili and 51k for Tagalog); noisy, web-crawled parallel data (Somali only, about 354k sentences); and synthetic, backtranslated parallel data created from monolingual sources including news articles, the Common Crawl, and Wikipedia (250-600k sentences). Table 1 shows the performance of the machine translation systems for each of the three languages on held-out test sets of 500 sentences taken from the clean, human-curated parallel data.

4.2 Abstractive Summarization

For our abstractive summarizers (hereafter *abstractors*), we implemented See et al.'s (2017) pointer-generator network in PyTorch (Paszke et al., 2017). We pre-train for 12 epochs on the unmodified NYT corpus to obtain a baseline system. Table 2 shows the performance of this baseline on the unmodified NYT test set; our baseline underperforms the more complex systems of Paulus et al. (2018) and Celikyilmaz et al. (2018), but we are more interested in the improvements our fluency-focused approach makes over this baseline than in the baseline's performance compared to state-of-the-art systems. We use each of the three noisy English corpora to train the baseline system for another 8 epochs, producing three language-specific abstractors. We also train a fourth, mixed-language abstractor using 100k articles randomly selected from the Somali, Swahili, and Tagalog training sets, evenly split among the three.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	32.94	10.36	22.51
Abs-so*	37.72	15.39	26.56
Abs-mix*	**38.07**	**15.76**	**26.82**

(a) Performance on Somali NYT.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	35.28	12.96	25.64
Abs-sw*	39.24	17.01	29.88
Abs-mix*	**39.96**	**17.56**	**30.24**

(b) Performance on Swahili NYT.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	37.17	14.67	27.27
Abs-tl*	**40.96**	18.72	31.06
Abs-mix*	40.87	**18.91**	**31.14**

(c) Performance on Tagalog NYT.

Table 3: Abs-so, -sw, and -tl are the Somali, Swahili, and Tagalog systems, respectively. * indicates significant improvement over NYT-base ($p < 1.16 \times 10^{-19}$).

	Perplexity		
Model	Somali NYT	Swahili NYT	Tagalog NYT
NYT-base	4986	4428	4707
Abs-so	**3357**	3429	3528
Abs-sw	3384	**3247**	**3312**
Abs-tl	3501	3476	3457
Abs-mix	3464	3285	3402

Table 4: Language model perplexity of generated summaries on noisy Somali, Swahili, and Tagalog NYT.

5 Evaluation

5.1 Noisy NYT Evaluation.

Table 3 shows the performance of our abstractors on the Somali, Swahili, and Tagalog NYT test sets. Differences among the language-specific systems are not statistically significant, and the more general mixed model achieved the best scores[1]. However, we found that abstractors trained solely on one language and tested on another significantly ($p < 0.05$) underperformed the mixed model, which was trained on all three languages, suggesting that training on some same-language data is still important.

We also trained a bigram language model on the entire set of NYT reference summaries and

---

[1] These results are shown in Appendix A, along with all combinations of the language-specific models on the three languages.

Document: mange kimambi 'i pray for the parliamentary seat for kinondoni constituency for ticket of ccm. not special seats' kinondoni without drugs is possible i pray for the parliamentary seat for kinondoni constituency on the ticket of ccm. yes, it's not a special seats, khuini kinondoni, what will i do for kinondoni? tension is many i get but we must remember no good that is available easily. kinondoni without drugs is possible. as a friend, fan or patriotism i urge you to grant your contribution to the situation and propert. you can use western union or money to go to mange john kimambi. account of crdb bank is on blog. reduce my profile in my blog understand why i have decided to vie for kinondoni constituency. you will understand more.

NYT-base: mange kimambi, who pray for parliamentary seat for kinondoni constituency for ticket of ccm in 0 , is on blog, and not special seats' kinondoni without drugs.

Abs-mix: mange kimambi, who pray for parliamentary seat for kinondoni constituency for ticket of ccm, comments on his plans to vie for 'kinondoni' without drugs.

Figure 2: An automatically translated Swahili weblog entry and its baseline and mixed abstractor summaries.

Somali Weblogs			Swahili Weblogs			Tagalog Weblogs		
Model	Content	Fluency	Model	Content	Fluency	Model	Content	Fluency
NYT-base	1.66	1.62	NYT-base	1.88	1.76	NYT-base	1.72	1.76
Abs-so	1.92	1.90	Abs-so	2.14	1.90	Abs-so	1.76	1.88
Abs-sw	1.94	1.88	Abs-sw	2.22	**2.08**	Abs-sw	1.94	1.92
Abs-tl	1.86	1.82	Abs-tl	2.18	1.86	Abs-tl	1.80	2.08
Abs-mix	**2.08**	**2.04**	Abs-mix	**2.36**	**2.08**	Abs-mix	**2.08**	**2.16**

Table 5: Average human-rated content and fluency scores on Somali, Swahili, and Tagalog weblog entries.

calculated the average perplexity of our abstractors' output as a proxy for fluency (Table 4). We see that Somali is the most difficult overall, but all three language-specific systems and the mixed model produce more fluent English across source languages than does the base model.

5.2 Weblog Evaluation.

We perform a human evaluation on 20 Somali, 20 Swahili, and 20 Tagalog weblog entries that we automatically translate into English using the same neural machine translation systems we used to create our noisy NYT corpora. Unlike our NYT data, which we translated from English into the low-resource languages, these weblogs are real-world Somali, Swahili, and Tagalog documents – this evaluation demonstrates the performance of our system in a real use-case. Figure 2 shows a Swahili weblog entry and its summaries[2]. This example shows the advantage of our approach: unlike a machine translation system, which must translate every part of its input, our abstractor is able to delete most of the long, rambling, and disfluent blog entry, instead summing it up fluently with the generated phrase "comments on his plans" and the repurposed phrase "to vie for".

We use five human judges, all native English speakers and none of whom are the authors. The judges were shown a translated document and a summary and asked to rate the content and fluency of the summary on a scale of 1–3 (Table 5). Our human judges rated our abstractors higher in both fluency and content, and we see again that while the language-specific systems are more fluent on their own languages than are the language-specific systems for the other languages, the mixed model still performs the best. We also see that, while our improvement in content is more modest, our improvement in fluency – the goal of this work – is significant. The judges achieved substantial agreement (Fleiss's $\kappa = 0.72$).

5.3 DUC 2004 Arabic Evaluation.

Finally, we evaluate our system on a new language: Arabic. We use the DUC 2004 Task 3 test set, which consists of real-world Arabic news articles translated into English, each paired with four human-written summaries.

Table 6 shows the performance of our abstractors on the Arabic data, demonstrating their ability to generalize and improve the fluency of input documents automatically translated from a previously unseen language, yielding a significant improvement in ROUGE. Compared to the 28 DUC 2004 systems, our performance would have ranked 1st on summarizing the machine-translated documents; despite our use of these lower-quality, *au-*

---

[2] All four abstractors produced very similar summaries.

Document: washington 10-23 (afp) was signed by benjamin netanyahu and yasser arafat on friday at the white house agreed on the israeli military withdrawal from the west bank in return for palestinian additional security guarantees.

NYT-base: washington 10-23, signed by benjamin netanyahu and yasser arafat, agrees on israeli military withdrawal from west bank in return for palestinian additional security guarantees.

Abs-mix: benjamin netanyahu and yasser arafat agree on israeli military withdrawal from west bank in return for palestinian security guarantees.

Figure 3: An Arabic article, automatically translated into English, and its baseline and mixed model summaries.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	26.56	5.86	15.76
Abs-so*	28.64	6.66	19.62
Abs-sw* †	28.08	6.39	18.36
Abs-tl* †	**29.43**	**7.02**	**19.89**
Abs-mix*	28.79	6.74	19.79

Table 6: DUC 2004 with ISI translations. * indicates significant improvement over NYT-base ($p < 2.09 \times 10^{-6}$); † indicates significant difference between systems ($p < 0.05$).

tomatically-translated documents, we performed extremely well even in comparison with the DUC 2004 systems on high-quality, *human*-translated documents: we would have ranked 1st, 4th, and 5th on ROUGE-1, -2, and -L, respectively. Figure 3 compares the baseline system and our abstractors on the Arabic data[3].

6 Discussion

We find that the NYT-base model tends to copy heavily from the beginning of its input documents. Since it was trained entirely on clean English news articles, it is understandable that it tries to copy the *lead* sentence, but in both examples, it copies errors: the confusing run-on sentence "not special seats' kinondoni without drugs is possible" (shown in yellow in Figure 2) and the phrase "signed by" (shown in green in Figure 3), whose subject is missing. In contrast, our abstractors are able to correctly identify the important information in the input documents and produce fluent summaries presenting this information. In Figure 3, Abs-mix deletes the unnecessary "washington 10-23" and produces the verb "agree" in the plural form, agreeing with its plural subject. More dramatically, in Figure 2, Abs-mix identifies "kinondoni without drugs" as Mange Kimambi's campaign platform and succinctly summarizes this using both the purely generated phrase "comments on his plans" and the repurposed – but still fluent and correct – "to vie for."

The main limitation of our approach is that it assumes the existence of a machine translation system for the source language. Although our abstractors are able to handle errorful, disfluent translations, for extremely low-resource languages, there may be no translations of any kind available; in such a case, another approach, such as cross-lingual word embeddings, is necessary.

7 Conclusion

We have presented a robust abstractive summarization system for the task of cross-lingual summarization, taking advantage of an abstractive system's ability to delete difficult to translate phrases and generate new text to use instead. Our straightforward method allows us to produce summarization systems for low resource languages where no summarization corpora are currently available, providing a potential summarization solution for thousands of such languages. Our experiments demonstrate that, by using our novel approach of training on noisy English documents and clean English reference summaries, the model learns to produce fluent summaries from disfluent inputs. Further, we have shown that, while training a system for a specific source language gives strong performance, the abstractive fluency of these systems generalize to other source languages.

Acknowledgements

This research is based upon work supported in part by the National Science Foundation (NSF), under Grant No. IIS-1422863, and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

---

[3] All four abstractor summaries were identical.

References

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*.

Rada Mihalcea. Language independent extractive summarization. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*.

Constantin Orăsan and Oana Andreea Chiorean. 2008. Evaluation of a cross-lingual romanian-english multi-document summariser. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the 6th International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*.

Xiaojun Wan. 2011. Using bilingual information for cross-language document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Xiaojun Wan, Huiying Li, and Jianguo Xiao. 2010. Cross-language document summarization based on machine translation quality prediction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Xiaojun Wan, Fuli Luo, Xue Sun, Songfang Huang, and Jin-ge Yao. 2018. Cross-language document summarization via extraction and ranking of multiple summaries. *Knowledge and Information Systems*.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Phrase-based compressive cross-language summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(10).

A Appendix: Language-Specific Abstractor Performance on Noisy NY

Table 7 expands Table 3, showing the performance of each of our four abstractors on the Somali, Swahili, and Tagalog NYT test sets. As discussed in Section 5.1, the differences among the three language-specific abstractors are not statistically significant on any of the three languages. However, the differences between the mixed model and the two language-specific models *not* trained on a given test language are significant ($p < 0.05$). That is, the difference between Abs-mix and Abs-sw and the difference between Abs-mix and Abs-tl are significant on the Somali test set, the difference between Abs-mix and Abs-so and the difference between Abs-mix and Abs-tl are significant on the Swahili test set, and the difference between Abs-mix and Abs-so and the difference between Abs-mix and Abs-sw are significant on the Tagalog test set.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	32.94	10.36	22.51
Abs-so*	37.72	15.39	26.56
Abs-sw* †	37.26	14.94	25.92
Abs-tl* †	36.89	14.41	25.53
Abs-mix*	**38.07**	**15.76**	**26.82**

(a) Performance on Somali NYT.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	35.28	12.96	25.64
Abs-so* †	38.42	16.34	29.06
Abs-sw*	39.24	17.01	29.88
Abs-tl* †	38.24	16.02	28.79
Abs-mix*	**39.96**	**17.56**	**30.24**

(b) Performance on Swahili NYT.

Model	ROUGE-1	ROUGE-2	ROUGE-L
NYT-base	37.17	14.67	27.26
Abs-so* †	38.97	17.01	29.16
Abs-sw* †	39.14	17.28	29.43
Abs-tl*	**40.96**	18.72	31.06
Abs-mix*	40.87	**18.91**	**31.14**

(c) Performance on Tagalog NYT.

Table 7: Abs-so, -sw, and -tl are the Somali, Swahili, and Tagalog language-specific systems, respectively. * indicates significant improvement over NYT-base ($p < 1.16 \times 10^{-19}$); † indicates significant difference between the mixed model and language-specific abstractors ($p < 0.05$).

Improving Neural Machine Translation with Neural Syntactic Distance

Chunpeng Ma[1,3], **Akihiro Tamura**[2], **Masao Utiyama**[3], **Tiejun Zhao**[1], **Eiichiro Sumita**[3]
[1]Harbin Institute of Technology, Harbin, China
[2]Ehime University, Matsuyama, Japan
[3]National Institute of Information and Communications Technology, Kyoto, Japan
{cpma, tjzhao}@hit.edu.cn tamura@cs.ehime-u.ac.jp
{mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

The explicit use of syntactic information has been proved useful for neural machine translation (NMT). However, previous methods resort to either tree-structured neural networks or long linearized sequences, both of which are inefficient. Neural syntactic distance (NSD) enables us to represent a constituent tree using a sequence whose length is identical to the number of words in the sentence. NSD has been used for constituent parsing, but not in machine translation. We propose five strategies to improve NMT with NSD. Experiments show that it is not trivial to improve NMT with NSD; however, the proposed strategies are shown to improve translation performance of the baseline model (+2.1 (En–Ja), +1.3 (Ja–En), +1.2 (En–Ch), and +1.0 (Ch–En) BLEU).

1 Introduction

In recent years, neural machine translation (NMT) has been developing rapidly and has become the de facto approach for machine translation. To improve the performance of the conventional NMT models (Sutskever et al., 2014; Bahdanau et al., 2014), one effective approach is to incorporate syntactic information into the encoder and/or decoder of the baseline model.

Based on how the syntactic information is represented, there are two categories of syntactic NMT methods: (1) those that use tree-structured neural networks (NNs) to represent syntax structures (Eriguchi et al., 2016; Hashimoto and Tsuruoka, 2017), and (2) those that use linear-structured NNs to represent linearized syntax structures (Li et al., 2017; Ma et al., 2017, 2018). For the first category, there is a direct corresponding relationship between the syntactic structure and the NN structure, but the complexity of NN structures usually makes training inefficient. In contrast, for the second category, syntactic structures are linearized and represented using linear-structured recurrent neural networks (RNNs), but the linearized sequence can generally be quite long and therefore training efficiency is still a problem. Although using a shorter sequence may improve the efficiency, some syntactic information is lost.

We propose a method of using syntactic information in NMT that overcomes the disadvantages of both methods. The basis of our method is the neural syntactic distance (NSD), a recently proposed concept used for constituent parsing (Shen et al., 2018; Gómez-Rodríguez and Vilares, 2018). NSD makes it possible to represent a constituent tree as a sequence whose length is identical to the number of words in the sentence (almost) without losing syntactic information. However, there are no previous studies that use NSD in NMT. Moreover, as demonstrated by our experiments, using NSD in NMT is far from straightforward, so we propose five strategies and verify the effects empirically. The strategies are summarized below.

- Extend NSD to dependency trees, which is inspired by the dependency language model (Shen et al., 2010).
- Use NSDs as input sequences[1], where an NSD is regarded as a linguistic input feature (Sennrich and Haddow, 2016).
- Use NSDs as output sequences, where the NMT and prediction of the NSD are simultaneously trained through multi-task learning (Firat et al., 2016).
- Use NSD as positional encoding (PE), which is a syntactic extension of the PE of the Transformer (Vaswani et al., 2017).
- Add a loss function for NSD to achieve

---

[1] Throughout this paper, ”input” means the input of an encoder or a decoder rather than the input of the NMT model (i.e., only source sentences), and ”output” is similar.

Proceedings of NAACL-HLT 2019, pages 2032–2037
Minneapolis, Minnesota, June 2 - June 7, 2019.

distance-aware training (Shen et al., 2018).

2 Neural Syntactic Distance (NSD)

The NSD was firstly proposed by Shen et al. (2018). This is the first method of linearizing a constituent tree with a sequence of length n, without loss of information, where n is the number of words in the sentence.

Formally, given the sentence $\mathbf{w} = (w_1, \ldots, w_n)$, for any pairs of contiguous words (w_i, w_{i+1}), we can define an NSD $d(w_i)$,[2] where $i \in [1, n-1]$. In Shen et al. (2018), the NSD $d_S(w_i)$ is defined as the height of the lowest common ancestor (LCA) of the words.[3] Subsequently, in Gómez-Rodríguez and Vilares (2018), the NSD $d_G(w_i)$ was defined as the number of the common ancestors of the words. To make the definition complete, we define $d(w_n)$ as follows:[4]

$$d_S(w_n) = H, \quad d_G(w_n) = 0, \tag{1}$$

where H is the height of the constituent tree. It is easy to prove that

$$d_S(w_i) + d_G(w_i) = H, \quad i \in [1, n]. \tag{2}$$

We call d_S and d_G the *absolute* NSD.

Furthermore, Gómez-Rodríguez and Vilares (2018) define the *relative* NSD as follows:

$$d_R(w_i) = \begin{cases} d_G(w_1), & i = 1, \\ d_G(w_i) - d_G(w_{i-1}), & i \in [2, n]. \end{cases} \tag{3}$$

Figure 1 illustrates these NSDs. It is easy to see the one-to-one correspondence relationship between the constituent tree and the (absolute or relative) NSDs.

The effectiveness of all different NSDs has been proven on constituent parsing. However, there has been no attempt to use NSD in machine translation.

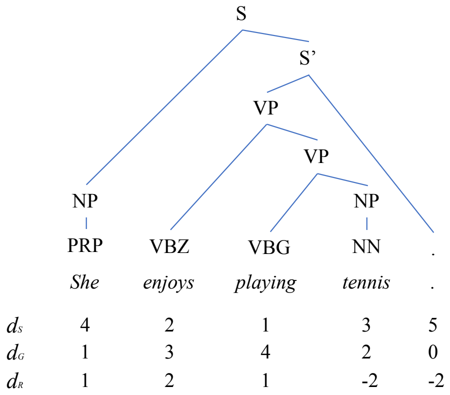


Figure 1: Example of different NSDs. This example is from Shen et al. (2018).

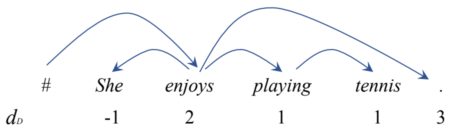


Figure 2: Example of dependency NSDs. "#" is the root. Dependency labels are omitted.

3 Strategies to improve NMT with NSD

3.1 Dependency NSD

There are many previous studies on using dependency trees to improve NMT (Nguyen Le et al., 2017; Wu et al., 2017). Therefore, we extend NSD to dependency trees. Formally, the dependency NSD between two nodes is defined as follows:

$$d_D(w_i) = i - h(i), \tag{4}$$

where $h(i)$ is the index of the head of w_i, and we let the index of root be 0. Note that $d_D(w_i)$ can be either positive or negative, representing the directional information. Figure 2 gives an example.

3.2 NSDs as Input Sequences

It is easy to see that for $\mathbf{w} = (w_1, \ldots, w_n)$, the lengths of d_S, d_G, d_R and d_D are all n. Denoting the NSD sequence as $\mathbf{d} = (d_1, \ldots, d_n)$, we can see that $d_i \in \mathbb{Z}, i \in [1, n]$, so we can obtain a sequence of embedding vectors $\mathbf{e}^d = (e_1^d, \ldots, e_n^d)$ as follows:

$$e_i^d = E_d[d_i^d + (\max(\mathbf{d}) - \min(\mathbf{d}) + 1)]. \tag{5}$$

We call E_d the distance embedding matrix and call $\mathbf{e}^d$ the syntactic embedding sequence. Note that $\mathbf{d}$ can be the NSD on either the source side or the target side, so there are two possible E_d, which are

---

[2] Note that NSD is defined between two contiguous words. For convenience of notation, we use $d(w_i)$ rather than $d(w_i, w_{i+1})$ to denote an NSD.

[3] In Shen et al. (2018), NSD is defined as a real number that is a function of LCA. However, in practice, NSD is simply identical to the depth of the LCA.

[4] $d_S(w_n)$ and $d_G(w_n)$ are undefined in both of the original papers. We give the definitions here to enable the use of NSD in NMT later.

denoted as E_d^s and E_d^t, respectively. The embeddings are calculated as follows:

$$x_i^s = f_{emb}(E_w^s[w_i^s], e_i^{ds}), \tag{6}$$
$$x_i^t = f_{emb}(E_w^t[w_i^t], e_i^{dt}), \tag{7}$$

where e_i^{ds} and e_i^{dt} are defined in Eq. 5 on the source side and target side, respectively, and E_w^s and E_w^t are the word embedding matrices on both sides, respectively. Inspired by Sennrich and Haddow (2016), function f_{emb} is used to combine two vectors. This function has many different options, such as:

$$f_{emb}^{\|}(x, e) = x \| e, \tag{8}$$
$$f_{emb}^{+}(x, e) = x + e, \tag{9}$$
$$f_{emb}^{Wb}(x, e) = W_f \cdot (x \| e) + b_f, \tag{10}$$

where $x, e, b_f \in \mathbb{R}^d$ and $W_f \in \mathbb{R}^{d \times 2d}$. The operator "$\|$" is the concatenation of two vectors.

When NSD is used as the input sequence on the target side, there is one problem: $\mathbf{e}^{dt}$ is unknown during testing. For this case, we use NSDs for both the input and output sequences, let the decoder predict NSD on-the-fly using the strategy introduced in Section 3.3, and use the predicted NSD to calculate $\mathbf{e}^{dt}$.

3.3 NSDs as Output Sequences

An NSD can be used to form the output sequence to improve NMT using the idea of multi-task learning. Specifically, we train the model to predict the NSD sequence. When NSD is used as the output sequence of the *encoder*, we minimize the distance (e.g., cross entropy $\mathcal{L}_{dist}^{ent}$, see Section 3.5 for details) between the predicted and the golden NSD sequences. When NSD is used as the output sequence of the *decoder*, besides minimizing the distance, we use the predicted NSD as the input of the next time step.

Denote the hidden vector as $\mathbf{h} = (h_1, \ldots, h_n)$. For the encoder, $h_i = h_i^s$ and $n = n_s$, while for the decoder, $h_i = h_i^t$ and n is the current time step of decoding. Then, we can obtain a sequence of predicted syntactic distance $\hat{\mathbf{d}} = (\hat{d}_1, \ldots, \hat{d}_n)$, which is calculated as follows:

$$p(\hat{d}_i \mid h_i) = \mathrm{softmax}(W_d \cdot h_i + b_d), \tag{11}$$

where W_d and b_d are parameters to be learned. By minimizing the distance between $\hat{d}_i$ and d_i, NSD can be used to enhance NMT.

3.4 NSD as Positional Encoding (PE)

PE is used by the Transformer (Vaswani et al., 2017) to encode the positions of words. Formally, it is defined as follows:

$$x_i' = x_i + PE(i), \tag{12}$$
$$PE(i)_{2k} = \sin(i/10000^{2k/d}), \tag{13}$$
$$PE(i)_{2k+1} = \cos(i/10000^{2k/d}), \tag{14}$$

where x_i can be either x_i^s or x_i^t, and d is the dimension of the embedding vector. Similarly, we define syntactic PE as follows:

$$PE(i)_{2k} = \sin\left(\frac{i + \max(\mathbf{d}) - \min(\mathbf{d})}{{\lambda_{SPE}}^{2k/d}} \cdot 2\pi\right), \tag{15}$$

$$PE(i)_{2k+1} = \cos\left(\frac{i + \max(\mathbf{d}) - \min(\mathbf{d})}{{\lambda_{SPE}}^{2k/d}} \cdot 2\pi\right), \tag{16}$$

where λ_{SPE} is a hyperparameter to be tuned. In this way, the periods of these two functions vary from 1 to λ_{SPE}. We define syntactic PE in this way because (1) according to a quantitative analysis of the experimental datasets, we found that the ranges of possible values are quite different between NSD and word positions, so we tuned λ_{SPE} instead of fixed it to 10000 as in Eqs. 13 and 14, and (2) d_i may be negative, so we adjust it to be positive.

3.5 Distance-aware Training

Instead of using conventional cross-entropy loss function during training, we use the following loss function to make the NMT model learn NSD better:

$$\mathcal{L} = \mathcal{L}_{NMT} + \mathcal{L}_{dist} + \mathcal{L}_{dist}^{ent}. \tag{17}$$

The first item is the cross-entropy loss of the NMT model, which is

$$\mathcal{L}_{NMT} = -\sum_{\langle \mathbf{w}^s, \mathbf{w}^t \rangle \in \mathcal{D}} \log p(\mathbf{w}^t \mid \mathbf{w}^s), \tag{18}$$

where $\mathcal{D}$ is the training dataset. The second item is the distance-aware loss, which is inspired by Shen et al. (2018) and is as follows:

$$\mathcal{L}_{dist} = \sum_{\langle \mathbf{w}^s, \mathbf{w}^t \rangle \in \mathcal{D}} (\mathcal{L}_{dist}^s(\mathbf{w}^s) + \mathcal{L}_{dist}^t(\mathbf{w}^t)),$$

$$\mathcal{L}_{dist}^s(\mathbf{w}^s) = \sum_{i=1}^{n_s} (d_i - \hat{d}_i)^2 + \sum_{i,j>i} [1 - \mathrm{sign}(d_i - d_j)(\hat{d}_i - \hat{d}_j)]^+, \tag{19}$$

and $\mathcal{L}^{t}_{dist}$ can be defined similarly. The third item is the cross-entropy loss for NSD, which is as follows:

$$\mathcal{L}^{ent}_{dist} = \sum_{\langle \mathbf{w}^s, \mathbf{w}^t \rangle \in \mathcal{D}} (\mathcal{L}^{ent(s)}_{dist}(\mathbf{w}^s) + \mathcal{L}^{ent(t)}_{dist}(\mathbf{w}^t)),$$
$$\mathcal{L}^{ent(s)}_{dist}(\mathbf{w}^s) = -\sum_{d_i \in \mathbf{d}^s} p(d_i \mid h_i) \log p(\hat{d}_i \mid h_i), \tag{20}$$

and $\mathcal{L}^{ent(t)}_{dist}$ can be defined similarly.

4 Experiments

4.1 Configuration

We experimented on two corpora: (1) ASPEC (Nakazawa et al., 2016), using the top 100K sentence pairs for training En–Ja models and top 1M sentence pairs for training Ja–En models, and (2) LDC,[5] which contains about 1.2M sentence pairs, for training En–Ch and Ch–En models. To tackling the problem of memory consumption, sentences longer than 150 were filtered out, so that models can be trained successfully. Chinese sentences were segmented by the Stanford segmentation tool.[6] For Japanese sentences, we followed the preprocessing steps recommended in WAT 2017.[7]

The test set is a concatenation of NIST MT 2003, 2004, and 2005. Constituent trees are generated by the parser of Kitaev and Klein (2018)[8], and dependency trees are generated by the parser of Dyer et al. (2015)[9]. Note that although we only used syntactic information of English in our experiments, our method is also applicable to other languages.

We implemented our method on `OpenNMT`[10] (Klein et al., 2017), and used the Transformer as our baseline. As far as we know, there are no previous studies on using syntactic informations in the Transformer.

The vocabulary sizes for all languages are $50,000$. Both the encoder and decoder have 6 layers. The dimensions of hidden vectors and word embeddings are 512. The multi-head attention has 8 heads, and the dropout probability is 0.1 (Srivastava et al., 2014). The number of training epochs was fixed to 50, and we used the model which performs the best on the development set for testing.

As for optimization, we used the Adam optimizer (Kingma and Ba, 2014), with $\beta_1 = 0.9, \beta_2 = 0.998$, and $\epsilon = 10^{-9}$. Warmup and decay strategy for learning rate of Vaswani et al. (2017) are also used, with $8,000$ warmup steps. We also used the label smoothing strategy (Szegedy et al., 2016) with $\epsilon_{ls} = 0.1$.

4.2 Experimental Results

Table 1 compares the effects of the strategies. We evaluate the proposed strategies using character-level BLEU (Papineni et al., 2002) for Chinese and Japanese, and case-insensitive BLEU for English.

Comparison of different NSDs. The first five rows of Table 1 compare the results of using different NSDs. When NSD was used at the source side (En–Ja/En–Ch), all kinds of NSDs improved translation performance. This indicates that NSD can be regarded as a useful linguistic feature to improve NMT. In contrast, when NSD was used at the target side (Ja–En/Ch–En), d_S and d_G hurt the performance. This is because the values of d_S and d_G are volatile. A tiny change of syntactic structure often causes a big change of d_S and d_G. Since the model has to predict the NSD during decoding, once there is one error, the subsequent predictions will be heavily influenced. The use of d_R and d_D remedies this problem. Furthermore, the effects of d_S and d_G are similar, because they are equivalent in nature (refer to Eq. 2).

NSD as PE. Rows 5 to 8 of Table 1 evaluate the use of dependency NSD (d_D) as syntactic PE. Note that for all the experiments, we used not only the syntactic PE but the conventional PE. Experiment results show that this strategy is indeed useful. When the dominators of Eqs. 15 and 16, λ_{SPE}, were set to 10^4, there was no improvement. When they were set to 40, the improvement was remarkable. This indicates that our design of syntactic PE is reasonable.

NSD as input/output and source/target sequences. Rows 8 to 12 of Table 1 are the results of using dependency NSD (i.e., d_D) as the input and/or output sequences on both sides. First, for the choice of f_{emb}, we can see that $f^{||}_{emb}$ and f^{+}_{emb} are similar, while f^{Wb}_{emb} yields better performance. This is because the model has to learn W_f and

---

[5] LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06.

[6] `https://nlp.stanford.edu/software/stanford-segmenter-2017-06-09.zip`

[7] `http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/baseline/dataPreparationJE.html`

[8] `https://github.com/nikitakit/self-attentive-parser`

[9] `https://github.com/clab/lstm-parser`

[10] `http://opennmt.net`

	Type	I/O	SPE	Loss	En–Ja	Ja–En	En–Ch	Ch–En
1	N/A	N/A	No	Eq. 18	34.59	26.43	29.41	31.60
2	d_S	I(f_{emb}^{Wb})	No	Eq. 18,19,20	35.54	24.57	29.66	28.77
3	d_G	I(f_{emb}^{Wb})	No	Eq. 18,19,20	35.38	24.71	29.60	28.82
4	d_R	I(f_{emb}^{Wb})	No	Eq. 18,19,20	35.83	26.88	29.87	31.82
5	d_D	I(f_{emb}^{Wb})	No	Eq. 18,19,20	36.17	27.21	30.11	32.08
6	d_D	I(f_{emb}^{Wb})	10^4	Eq. 18,19,20	36.06	27.18	30.02	32.03
7	d_D	I(f_{emb}^{Wb})	10^2	Eq. 18,19,20	36.22	27.47	30.23	32.19
8	d_D	I(f_{emb}^{Wb})	40	Eq. 18,19,20	36.44	27.59	**30.59**	32.36
9	d_D	I($f_{emb}^{\Vert}$)	40	Eq. 18,19,20	36.17	27.21	30.15	32.11
10	d_D	I(f_{emb}^{+})	40	Eq. 18,19,20	36.08	27.32	30.21	32.29
11	d_D	O	40	Eq. 18,19,20	36.31	27.42	30.42	32.32
12	d_D	I(f_{emb}^{Wb})&O	40	Eq. 18,19,20	**36.69**	**27.71**	30.56	**32.55**
13	d_D	O	40	Eq. 18	21.08	10.22	18.63	15.61
14	d_D	O	40	Eq. 18,20	33.70	23.31	27.43	30.02
15	d_D	O	40	Eq. 18,19	34.18	25.19	29.14	31.74

Table 1: Comparison of strategies. I/O: use NSDs as the input or output sequences. Functions $f_{emb}^{\Vert}$, f_{emb}^{+}, and f_{emb}^{Wb} are defined in Eqs. 8 to 10, respectively. SPE: use NSD as syntactic PE. Numbers are the values of λ_{SPE} in Eqs. 15 and 16. Loss: items used in the final loss function. Note that when NSD is used as the input sequence of the source language, $\mathcal{L}_{dist} + \mathcal{L}_{dist}^{ent} \equiv 0$, because the parsing tree is fixed.

b_f, which increases the model capacity. Second, performance improved for using NSDs both as input and output sequences, and combining both obtained further improvement. Third, NSDs improved the performance both on the source and the target sides. All these results indicate the robustness of NSDs.

Effects of distance-aware training. The last three rows compare the different effects of the items in the loss function. When only $\mathcal{L}_{NMT}$ are used, the performance is extremely poor. This is within expectations, because with only $\mathcal{L}_{NMT}$, weights related to NSDs are kept to the initial values and were not updated, and hence detrimental to learning. Adding $\mathcal{L}_{dist}^{ent}$ improves the results significantly, but the improvement is lower than that of $\mathcal{L}_{dist}$. This is because training with $\mathcal{L}_{dist}^{ent}$ treats different values of NSDs equally, while $\mathcal{L}_{dist}$ penalizes larger differences between the predicted NSD and the golden NSD more severely.

5 Conclusion

We proposed five strategies to improve NMT with NSD. We found relative NSDs and dependency NSDs are able to improve the performance consistently, while absolute NSDs hurt the performance for some cases. The improvement obtained by using NSDs is general in that NSDs can be used at both the source side and target side, both as input sequences and output sequences. Using NSDs as syntactic PE is also useful, and training with a distance-aware loss function is quite important.

Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments and suggestions. Akihiro Tamura is supported by JSPS KAKENHI Grant Number JP18K18110. Tiejun Zhao is supported by the National Key Research and Development Program of China via grant 2017YFB1002102.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1701.02810*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Eiichiro Sumita. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proc. of AAAI-2017*, pages 3237–3243.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. 2018. Forest-based neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1253–1263.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208.

An Nguyen Le, Ander Martinez, Akifumi Yoshimoto, and Yuji Matsumoto. 2017. Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 21–29.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4).

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707.

Measuring Immediate Adaptation Performance for Neural Machine Translation

Patrick Simianer, Joern Wuebker, John DeNero
Lilt
`<given name>@lilt.com`

Abstract

Incremental domain adaptation, in which a system learns from the correct output for each input immediately after making its prediction for that input, can dramatically improve system performance for interactive machine translation. Users of interactive systems are sensitive to the speed of adaptation and how often a system repeats mistakes, despite being corrected. Adaptation is most commonly assessed using corpus-level BLEU- or TER-derived metrics that do not explicitly take adaptation speed into account. We find that these metrics often do not capture immediate adaptation effects, such as zero-shot and one-shot learning of domain-specific lexical items. To this end, we propose new metrics that directly evaluate immediate adaptation performance for machine translation. We use these metrics to choose the most suitable adaptation method from a range of different adaptation techniques for neural machine translation systems.

1 Introduction

Incremental domain adaptation, or *online* adaptation, has been shown to improve statistical machine translation and especially neural machine translation (NMT) systems significantly (Turchi et al., 2017; Karimova et al., 2018) (*inter-alia*). The natural use case is a computer-aided translation (CAT) scenario, where a user and a machine translation system collaborate to translate a document. Each user translation is immediately used as a new training example to adapt the machine translation system to the specific document.

Adaptation techniques for MT are typically evaluated by their corpus translation quality, but such evaluations may not capture prominent aspects of the user experience in a collaborative translation scenario. This paper focuses on directly measuring the speed of lexical acquisition for in-domain vocabulary. To that end, we propose three related metrics that are designed to reflect the responsiveness of adaptation.

An ideal system would immediately acquire in-domain lexical items upon observing their translations. Moreover, one might expect a neural system to generalize from one corrected translation to related terms. Once a user translates “bank” to German “Bank” (*institution*) instead of “Ufer” (*shore*) in a document, the system should also correctly translate “banks” to “Banken” instead of “Ufer” (the plural is identical to the singular in German) in future sentences. We measure both one-shot vocabulary acquisition for terms that have appeared once in a previous target sentence, as well as zero-shot vocabulary acquisition for terms that have not previously appeared.

Our experimental evaluation shows some surprising results. Methods that appear to have comparable performance using corpus quality metrics such as BLEU can differ substantially in zero-shot and one-shot vocabulary acquisition. In addition, we find that *fine-tuning* a neural model tends to improve one-shot vocabulary recall while *degrading* zero-shot vocabulary recall.

We evaluate several adaptation techniques on a range of online adaptation datasets. Fine tuning applied to all parameters in the NMT model maximizes one-shot acquisition, but shows a worrisome degradation in zero-shot recall. By contrast, fine tuning with group lasso regularization, a technique recently proposed to improve the space efficiency of adapted models (Wuebker et al., 2018), achieves an appealing balance of zero-shot and one-shot vocabulary acquisition as well as high corpus-level translation quality.

Proceedings of NAACL-HLT 2019, pages 2038–2046
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Measuring Immediate Adaptation

2.1 Motivation

For interactive, adaptive machine translation systems, *perceived* adaptation performance is a crucial property: An error in the machine translation output which needs to be corrected multiple times can cause frustration, and thus may compromise acceptance of the MT system by human users. A class of errors that are particularly salient are lexical choice errors for domain-specific lexical items. In the extreme, NMT systems using subword modeling (Sennrich et al., 2015) can generate "hallucinated" words—words that do not exist in the target language—which are especially irritating for users (Lee et al., 2018; Koehn and Knowles, 2017). Users of adaptive MT have a reasonable expectation that in-domain vocabulary will be translated correctly after the translation of a term or some related term has been corrected manually.

Arguably, more subtle errors, referring to syntax, word order or more general semantics are less of a focus for immediate adaptation, as these types of errors are also harder to pinpoint and thus to evaluate[1] (Bentivogli et al., 2016). Traditional metrics for evaluating machine translation outputs, e.g. BLEU and TER, in essence try to measure the similarity of a hypothesized translation to one or more reference translations, taking the full string into account.

Due to significant improvements in MT quality with neural models (Bentivogli et al., 2016) (*inter-alia*), more specialized metrics, evaluating certain desired behaviors of systems become more useful for specific tasks. For example, Wuebker et al. (2016) show, that NMT models, while being better in most respects, still fall short in the handling of content words in comparison with phrase-based MT. This observation is also supported by Bentivogli et al. (2016), who show smaller gains for NMT for translation of nouns, an important category of content words.

Another reason to isolate vocabulary acquisition as an evaluation criterion is that interactive translation often employs local adaptation via prefix-decoding (Knowles and Koehn, 2016; Wuebker et al., 2016), which can allow the system to recover syntactic structure or resolve local ambiguities when given a prefix, but may still suffer from poor handling of unknown or domain-specific vocabulary.

In this work, we therefore focus on translation performance with respect to content words, setting word order and other aspects aside.

2.2 Metrics

We propose three metrics: one to directly measure one-shot vocabulary acquisition, one to measure zero-shot vocabulary acquisition, and one to measure both. In all three, we measure the recall of target-language content words so that the metrics can be computed automatically by comparing translation hypotheses to reference translations without the use of models or word alignments[2].

We define content words as those words that are not included in a fixed stopword list, as used for example in query simplification for information retrieval. Such lists are typically compiled manually and are available for many languages.[3] For western languages, content words are mostly nouns, main verbs, adjectives or adverbs.

For the i-th pair of source sentence and reference translation, $i = 1, \dots, |\mathcal{G}|$, of an ordered test corpus $\mathcal{G}$, we define two sets $\mathcal{R}_{0,i}$ and $\mathcal{R}_{1,i}$ that are a subset of the whole set of unique[4] content words (i.e. types) of the reference translation for i. $\mathcal{R}_{0,i}$ includes a word if its first occurrence in the test set is in the i-th reference of $\mathcal{G}$, and $\mathcal{R}_{1,i}$ if its second occurrence in the test set is in the i-th reference of $\mathcal{G}$. The union $\mathcal{R}_{0,i} \cup \mathcal{R}_{1,i}$ includes content words occurring for either the first or second time.

To measure zero-shot adaptation in a given hypothesis $\mathcal{H}_i$, also represented as a set of its content words, we propose to evaluate the number of word types that were immediately translated correctly:

$$\text{R0} = \frac{|\mathcal{H}_i \cap \mathcal{R}_{0,i}|}{|\mathcal{R}_{0,i}|}.$$

To measure one-shot adaptation, where the system correctly produces a content word after ob-

---

[1] Some practitioners observed that these subtle errors become harder to spot due the improved fluency of NMT systems (Burchardt, 2017).

[2] In each of the data sets considered in this work, the average number of occurrences of content words ranges between 1.01 and 1.11 per sentence. We find this sufficiently close to 1 to evaluate in a bag-of-words fashion and not consider alignments.

[3] For German we used the list available here: `https://github.com/stopwords-iso`.

[4] All proposed metrics operate on the set-level, without clipping (Papineni et al., 2002) or alignment (Banerjee and Lavie, 2005; Kothur et al., 2018), as we have found this simplification effective.

	Reference	Hypothesis	R0	R1	R0+1
1.	The [dog] [bites] the [lady]	A [terrier] [bites] the [person]	1/3	0/0	1/3
2.	The [man] [bites] the [dog]	The [dog] [bites] the [man]	1/1	2/2	3/3
		Total	2/4	2/2	4/6

Figure 1: Example for calculating R0, R1, and R0+1 on a corpus of two sentences. Content words are written in brackets, the corpus-level score is given below the per-segment scores. In the example, the denominator for R1 is 2 due to the two repeated words *dog* and *bites* in the reference.

serving it exactly once, we propose:

$$\text{R1} = \frac{|\mathcal{H}_i \cap \mathcal{R}_{1,i}|}{|\mathcal{R}_{1,i}|}.$$

This principle can be extended to define metrics Rk, $k > 1$ to allow more "slack" in the adaptation, but we leave that investigation to future work.

Finally, we define a metric that measures both zero- and one-shot adaptation:

$$\text{R0+1} = \frac{|\mathcal{H}_i \cap [\mathcal{R}_{0,i} \cup \mathcal{R}_{1,i}]\,|}{|\mathcal{R}_{0,i} \cup \mathcal{R}_{1,i}|}.$$

All metrics can either be calculated for single sentences as described above, or for a full test corpus by summing over all sentences, e.g. for R0:

$$\frac{\sum_{i=1}^{|\mathcal{G}|} |\mathcal{H}_i \cap \mathcal{R}_{0,i}|}{\sum_{i=1}^{|\mathcal{G}|} |\mathcal{R}_{0,i}|}.$$

Figure 1 gives an example calculation of all three metrics across a two-sentence corpus.

3 Related Work

An important line of related work is concerned with estimating the *potential* adaptability of a system given a source text only, the so-called repetition rate (Cettolo et al., 2014). The metric is inspired by BLEU, and uses a sliding window over the source text to count singleton N-grams.

The modus operandi for our metrics is most similar to HTER (Snover et al., 2006), since we are also assuming a single, targeted reference translation[5] for evaluation.

The introduction of NMT brought more aspects of translation quality evaluation into focus, such as discourse-level evaluation (Bawden et al., 2017), or very fine-grained evaluation of specific aspects of the translations (Bentivogli et al., 2016), highlighting the differences between phrase-based and NMT systems.

Online adaptation for (neural) machine translation has been thoroughly explored using BLEU (Turchi et al., 2017), simulated keystroke and mouse action ratio (Barrachina et al., 2009) for effort estimation (Peris and Casacuberta, 2018), word prediction accuracy (Wuebker et al., 2016), and user studies (Denkowski et al., 2014; Karimova et al., 2018) (all *inter-alia*). In (Simianer et al., 2016) immediate adaptation for hierarchical phrase-based MT is specifically investigated, but they also evaluate their systems using human-targeted BLEU and TER.

Regularization for segment-wise continued training in NMT has been explored by Khayrallah et al. (2018) by means of knowledge distillation, and with the group lasso by Wuebker et al. (2018), as used in this paper.

Most relevant to our work, in the context of document-level adaptation, Kothur et al. (2018) calculate accuracy for novel words based on an automatic word alignment. However, they do not focus on zero- and one-shot matches, but instead aggregate counts over the full corpus.

4 Online Adaptation

NMT systems can be readily adapted by fine-tuning (also called continued training) with the same cross-entropy loss ($\mathcal{L}$) as used for training the parameters of the baseline system, which also serves as the starting point for adaptation (Luong and Manning, 2015). Following Turchi et al. (2017), we perform learning from each example i using (stochastic) gradient descent, using the current source x_i and reference translation y_i as a batch of size 1:

$$\theta_i \leftarrow \theta_{i-1} - \gamma \nabla \mathcal{L}(\theta_{i-1}, x_i, y_i). \qquad (1)$$

Evaluation is carried out using simulated post-editing (Hardt and Elming, 2010), first translating the source using the model with parameters θ_{i-1}, before performing the update described

---

[5] A reference translation which was produced from post-editing output of the to-be-evaluated MT system.

above with the now revealed reference translation. The machine translation system effectively only trains for a single iteration for any given data set.

The naïve approach, updating all parameters θ of the NMT model, while being effective, can be infeasible in certain settings[6], since tens of millions of parameters are updated depending on the respective model. While some areas of a typical NMT model can be stored in a sparse fashion without loss (source- and target embeddings), large parts of the model cannot. We denote this type of adaptation as *full*.

A light-weight alternative to adaptation of the full parameter set is to introduce a second bias term in the final output layer of the NMT model, which is trained in isolation, freezing the rest of the model (Michel and Neubig, 2018). This merely introduces a vector in the size of the output vocabulary. This method is referred to as *bias*.

Another alternative is freezing parts of the model (Thompson et al., 2018), for example determining a subset of parameters by performance on a held-out set (Wuebker et al., 2018). In our experiments we use two systems using this method, *fixed* and *top*, the former being a pre-determined fixed selection of parameters, and the latter being the topmost encoder and decoder layers in the *Transformer* NMT model (Vaswani et al., 2017).

Finally, a data-driven alternative to the fixed freezing method was introduced to NMT by Wuebker et al. (2018), implementing tensor-wise ℓ_1/ℓ_2 group lasso regularization, allowing the learning procedure to select a fixed number of parameters after each update. This setup is referred to as *lasso*.

5 Experiments

5.1 Neural Machine Translation Systems

We adapt an English→German NMT system based on the Transformer architecture trained with an in-house NMT framework on about 100M bilingual sentence pairs. The model has six layers in the encoder, three layers in the decoder, each with eight attention heads with dimensionality 256, distinct input and output embeddings, and vocabulary sizes of around 40,000. The vocabularies are generated with byte-pair encoding (Sennrich et al., 2015). For adaptation we use a learning rate γ of 10^{-2} (for the *bias* adaptation a learning rate of 1.0 is used), no dropout, and no label-smoothing. We use a tensor-wise ℓ_2 normalization to 1.0 for all gradients (gradient clipping). Updates for a sentence pair are repeated until the perplexity on that sentence pair is ≤ 2.0, for a maximum of three repetitions. The *fixed* adaptation scheme, which involves selecting a subset of parameters on held-out data following Wuebker et al. (2018), uses about two million parameters excluding all embedding matrices, in addition to potentially the full source embeddings, but in practice this is limited to about 1M parameters. The *top* scheme only adapts the top layers for both encoder and decoder. For the *lasso* adaptation, we allow 1M parameters excluding the embeddings, for which we allow 1M parameters in total selected from all embedding matrices. This scheme also always includes the previously described second bias term in the final output layer.

Method	BLEU	SBLEU	TER
baseline	40.3	49.3	45.2
bias	40.4	49.5	45.0
full	47.0	**55.9**	44.0
lasso	46.3	54.3	42.6
fixed	**47.1**	55.5	**41.0**
top	43.2	54.0	49.5

Table 1: Results on the *Autodesk* test set for traditional MT quality metrics. *SBLEU* refers to an average of sentence-wise BLEU scores as described by Nakov et al. (2012). The best result in each column is denoted with bold font.

Since the proposed metrics operate on words, the machine translation outputs are first converted to full-form words using *sentencepiece* (Kudo and Richardson, 2018), then tokenized and truecased with the tokenizer and truecaser distributed with the *Moses* toolkit (Koehn et al., 2007).

5.2 Results

Tables 1 and 2 show the performance of different adaptation techniques on the *Autodesk* dataset (Zhechev, 2012), a public post-editing software domain dataset for which incremental adaptation is known to provide large gains for corpus-level metrics. BLEU, sentence BLEU, and TER scores (Table 1) are similar for *full* adaptation, sparse adaptation with group *lasso*, and adaptation of a *fixed* subset of parameters. However (in Table 2),

---

[6] For example in setups where a large number of these adapted models need to be stored and transferred.

Method	R0	R1	R0+1
baseline	39.3	44.9	41.0
bias	39.3	45.3	41.1
full	35.8	**55.0**	41.6
lasso	**40.3**	48.6	**42.8**
fixed	35.8	52.3	40.8
top	35.6	50.3	40.0

Table 2: Results on the *Autodesk* test set for the proposed metrics R0, R1, and R0+1.

lasso substantially outperforms the other methods in zero-shot (R0), and combined zero- and one-shot recall of content words (R0+1).

Zero-shot recall is considerably degraded relative to the non-adapted baseline for both *full* and adaptation of a fixed subset of tensors (*fixed* and *top*). That is, terms never observed before during online training are translated correctly less often than they would be with an unadapted system, despite the data set's consistent domain. These approaches trade off long-term gains in BLEU and high one-shot recall for low zero-shot recall, which could be frustrating for users who may perceive the degradation in quality for terms appearing for the first time in a document. The *lasso* technique is the only one that shows an improvement in R0 over the baseline. However, *lasso* has considerably lower one-shot recall compared to the other adaptation methods, implying that it often must observe a translated term more than once to acquire it.

Appendix A shows similar experiments for several other datasets.

5.3 Analysis

For a better understanding of the results described in the previous section, we conduct an analysis varying the units of the proposed metrics, while focusing on *full* and *lasso* adaptation.

For the first variant, only truly novel words are taken into account, i.e. words in the test set that do not appear in the training data. Results for these experiments are depicted in Table 3. It is apparent that the findings of Table 2 are confirmed, and that relative differences are amplified. This can be explained by the reduced number of total occurrences considered, which is only 310 words in this data set. It is also important to note that all of these words are made up of known subwords[7], since our NMT system does not include a copying mechanism and is thus constrained to the target vocabulary.

Method	R0	R1	R0+1
baseline	27.1	40.7	29.9
full	26.1	**63.0**	33.8
lasso	**31.9**	53.1	**36.3**

Table 3: Results on *Autodesk* data calculating the metrics only for truly novel content words, i.e. ones that do not occur in the training data.

Method	R0	R1	R0+1
baseline	**44.1**	48.1	45.5
full	40.4	**54.6**	45.4
lasso	43.7	51.7	**46.5**

Table 4: Results on *Autodesk* data calculating the metrics with subwords.

Further results using the raw subword output[8] of the MT systems are depicted in Table 4: R0 for the *lasso* method is degraded only slightly below the baseline (-1%, compared to +2% for the regular metric), the findings for R1 and R0+1 remain the same as observed before. Compared to the results for novel words this indicates that the improvement in terms of R0 for *lasso* mostly come from learning new combinations of subwords.

A discussion of the adaptation behavior over time, with exemplified differences between the metrics, can be found in Appendix B.

6 Conclusions

To summarize: In some cases, the strong gains in corpus-level translation quality achieved by fine tuning an NMT model come at the expense of zero-shot recall of content words. This concerning impact of adaptation could affect practical user experience. Existing regularization methods mitigate this effect to some degree, but there may be more effective techniques for immediate adaptation that have yet to be developed.

The proposed metrics R0, R1, and R0+1 are useful for measuring immediate adaptation performance, which is crucial in adaptive CAT systems.

---

[7] The test set does not contain any unknown characters.

[8] Note that this includes all tokens, not just parts of content words.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, et al. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2017. Evaluating discourse phenomena in neural machine translation. *arXiv preprint arXiv:1711.00513*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*.

Aljoscha Burchardt. 2017. Comparing errors: Neural MT vs. traditional phrase-based and rule-based MT. https://www.gala-global.org/publications/comparing-errors-neural-mt-vs-traditional-phrase-based-and-rule-based-mt.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2014. The repetition rate of text as a predictor of the effectiveness of machine translation adaptation. In *Proceedings of AMTA*, pages 166–179.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proceedings of EAMT*, pages 261–268.

Michael Denkowski, Alon Lavie, Isabel Lacruz, and Chris Dyer. 2014. Real time adaptive machine translation for post-editing with cdec and transcenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 72–77.

Daniel Hardt and Jakob Elming. 2010. Incremental re-training for post-editing SMT. In *Proceedings of AMTA*, volume 122.

Sariya Karimova, Patrick Simianer, and Stefan Riezler. 2018. A user-study on online adaptation of neural machine translation to human post-edits. *Machine Translation*, 32(4):309–324.

Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 36–44.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Proceedings of AMTA*, pages 107–120.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*, pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Sachith Sri Ram Kothur, Rebecca Knowles, and Philipp Koehn. 2018. Document-level adaptation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 64–73.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation. https://openreview.net/forum?id=SkxJ-309FQ.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of IWSLT*, pages 76–79.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. *arXiv preprint arXiv:1805.01817*.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. *Proceedings of COLING*, pages 1979–1994.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Álvaro Peris and Francisco Casacuberta. 2018. Online learning for effort reduction in interactive neural machine translation. *arXiv preprint arXiv:1802.03594*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Patrick Simianer, Sariya Karimova, and Stefan Riezler. 2016. A post-editing interface for immediate adaptation in statistical machine translation. In *Proceedings of COLING*, pages 16–20, Osaka, Japan.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, volume 200.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of WMT*, pages 124–132.

Marco Turchi, Matteo Negri, M Amin Farajian, and Marcello Federico. 2017. Continuous learning from human post-edits for neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):233–244.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Katharina Wäschle and Stefan Riezler. 2012. Analyzing parallelism and domain similarities in the MAREC patent corpus. *Multidisciplinary Information Retrieval*, pages 12–27.

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of ACL*, volume 1, pages 66–75.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. In *Proceedings of EMNLP*, pages 881–886.

Ventsislav Zhechev. 2012. Machine translation infrastructure and post-editing performance at autodesk. In *Proceedings of WPTP*, pages 87–96.

A Additional Results

Table 5 contains results for additional English→German datasets, namely patents (Wäschle and Riezler, 2012) (*Patent*), transcribed public speeches (Cettolo et al., 2012) (*TED*), and two proprietary user data sets, one from the financial domain (*User 1*) and the other being technical documentation (*User 2*). The same pattern is observed in almost all cases: *lasso* outperforms the other adaptation techniques in zero-shot recall (R0) and combined recall (R0+1), while *full* has the highest one-shot recall (R1) on two out of five test sets, being close runner-up to *lasso* on all others. Overall however, we observe that zero-shot recall R0 is degraded by adaptation, while one-shot recall is improved. We also find that adaptation with the light-weight *bias* method often does not deviate much from the baseline. In contrast, the results for the traditional MT metrics are predominantly positive. For adaptation, the *lasso* method provides the best tradeoff in terms of performance throughout the considered metrics.

B Learning Curves

We are also interested in the behavior of the adaptation methods over time. To this end, in Figure 2, we plot the difference in cumulative scores[9] of two adapted systems (*full* and *lasso*) to the baseline for the proposed metrics as well as the BLEU score.

As evident from comparing the curves for BLEU and R0, the BLEU score and the proposed metric give disparate signals for this data. Specifically, there are two distinct dips in the curves for R0 (as well as R0+1) and BLEU:

1. The degradation in R0 around segment 800 is due to significant noise in segment 774, which has a strong impact on the adapted systems, while the baseline system is not affected. The *full* system's score drops by about 50% at segment 775 (i.e. after adaptation) relative to the cumulative score difference at the previous segment and never recovers after that.
2. The dip in the BLEU score at segment 752, observable for both adapted systems, depicting a relative degradation of about 10%, is due to a pathological repetition of a single character in the output of the adapted MT systems for this segment, which has a large impact on the score.

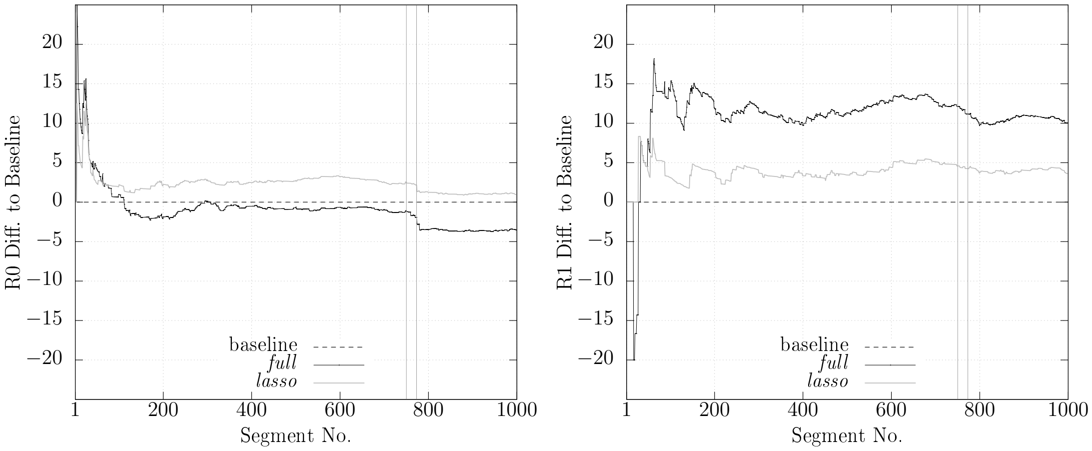
The dip observed with R0 is also noticeable in BLEU, but much less pronounced at 4% relative for *full* and 2% relative for *lasso*. The dip in BLEU on the other hand is not noticeable with R0, R1, or R0+1.

---

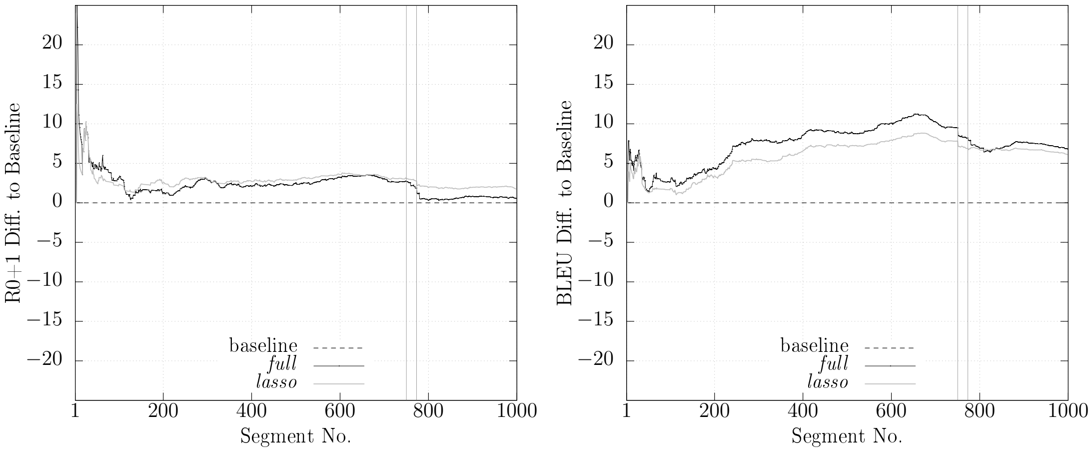
[9] For each sentence i in the data set, the metrics for all systems are calculated up to the ith sentence. The difference for the adapted systems is then calculated by subtracting the baseline score.

User 1	BLEU	SBLEU	TER	R0+1	R0	R1
baseline	35.7	55.2	52.4	44.3	42.8	50.3
bias	8	6	-4	-5	-5	-4
full	36	18	-22	-4	-7	6
lasso	38	18	-23	1	-1	8
fixed	34	18	-22	-6	-9	4
top	29	16	-17	-5	-8	4
User 2	BLEU	SBLEU	TER	R0+1	R0	R1
baseline	35.5	56.2	51.0	43.6	41.0	51.2
bias	0	0	0	0	0	-1
full	0	5	5	-3	-5	4
lasso	6	6	-6	2	0	7
fixed	-5	4	13	-4	-7	1
top	-3	3	4	-5	-7	-2
Autodesk	BLEU	SBLEU	TER	R0+1	R0	R1
baseline	40.3	49.3	45.2	41.0	39.3	44.9
bias	0	0	0	0	0	1
full	17	13	-3	1	-9	22
lasso	15	10	-6	4	3	8
fixed	17	13	-9	0	-9	16
top	7	10	10	-2	-9	12
TED	BLEU	SBLEU	TER	R0+1	R0	R1
baseline	25.9	56.0	54.2	42.6	39.5	53.2
bias	1	0	0	0	0	0
full	0	1	1	-3	-6	3
lasso	4	2	-2	-1	-3	4
fixed	-3	0	4	-4	-7	2
top	-6	0	9	-2	-5	5
Patent	BLEU	SBLEU	TER	R0+1	R0	R1
baseline	53.5	62.1	31.7	51.8	49.7	57.3
bias	2	1	-2	0	0	0
full	3	2	-2	-2	-5	7
lasso	4	2	-4	0	-2	5
fixed	2	1	1	-4	-7	4
top	2	1	-1	-3	-5	2

Table 5: BLEU, sentence-wise BLEU, TER, R0+1, R0, and R1 metrics for a number of data sets, comparing different adaptation methods as described in Section 4. Baseline results are given as absolute scores, results for adaptation are given as relative differences. Best viewed in color.



(a) R0 (b) R1



(c) R0+1 (d) BLEU

Figure 2: Differences in cumulative scores for R0 (top left), R1 (top right), R0+1 (bottom left), and BLEU (bottom right) to the baseline system on the *Autodesk* test set for *full* and *lasso* adaptation. The peculiarities discussed in the running text are marked by solid vertical lines (at $x = 751$ and $x = 774$).

Differentiable Sampling with Flexible Reference Word Order for Neural Machine Translation

Weijia Xu
University of Maryland
`weijia@cs.umd.edu`

Xing Niu
University of Maryland
`xingniu@cs.umd.edu`

Marine Carpuat
University of Maryland
`marine@cs.umd.edu`

Abstract

Despite some empirical success at correcting exposure bias in machine translation, scheduled sampling algorithms suffer from a major drawback: they incorrectly assume that words in the reference translations and in sampled sequences are aligned at each time step. Our new differentiable sampling algorithm addresses this issue by optimizing the probability that the reference can be aligned with the sampled output, based on a soft alignment predicted by the model itself. As a result, the output distribution at each time step is evaluated with respect to the whole predicted sequence. Experiments on IWSLT translation tasks show that our approach improves BLEU compared to maximum likelihood and scheduled sampling baselines. In addition, our approach is simpler to train with no need for sampling schedule and yields models that achieve larger improvements with smaller beam sizes.[1]

1 Introduction

Neural machine translation (NMT) models are typically trained to maximize the likelihood of reference translations (Sutskever et al., 2014; Bahdanau et al., 2015). While simple and effective, this objective suffers from the exposure bias problem (Ranzato et al., 2015): the model is only exposed to reference target sequences during training, but has to rely on its own predictions at inference. As a result, errors can accumulate along the generated sequence at inference time.

This is a well-known issue in sequential decision making (Langford and Zadrozny, 2005; Cohen and Carvalho, 2005; Kääriäinen and Langford, 2006, i.a.) and it has been addressed in past work by incorporating the previous decoding choices into the training scheme, using imitation learning (Daumé et al., 2009; Ross et al., 2011; Bengio et al., 2015; Leblond et al., 2018) and reinforcement learning (Ranzato et al., 2015; Bahdanau et al., 2016) techniques. In this paper, we focus on a simple and computationally inexpensive family of approaches, known as Data as Demonstrator (Venkatraman et al., 2015) and scheduled sampling (Bengio et al., 2015; Goyal et al., 2017). The algorithms use a stochastic mixture of the reference words and model predictions with an annealing schedule controlling the mixture probability. Despite their empirical success in various sequence prediction tasks, they are based on an assumption that does not hold for machine translation: they assume that words in the reference translations and in sampled sequences are aligned at each time step, which results in weak and sometimes misleading training signals.

In this paper, we introduce a differentiable sampling algorithm that exposes machine translation models to their own predictions during training, and allows for differences in word order when comparing model outputs with reference translations. We compute the probability that the reference can be aligned with the sampled output using a soft alignment predicted based on the model states, so that the model will not be punished too severely for producing hypotheses that deviate from the reference, as long as the hypotheses can still be aligned with the reference.

Experiments on three IWSLT tasks (German-English, English-German and Vietnamese-English) show that our approach significantly improves BLEU compared to both maximum likelihood and scheduled sampling baselines. We also provide evidence that our approach addresses exposure bias by decoding with varying beam sizes, and show that our approach is simpler to train than scheduled sampling as it requires no annealing schedule.

---

[1] The code is available at `https://github.com/Izecson/saml-nmt`

Proceedings of NAACL-HLT 2019, pages 2047–2053
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Approach

Our approach is designed to optimize the standard sequence-to-sequence model for translating a source sentence $\boldsymbol{x}$ into a target sentence $\boldsymbol{y}$ (Bahdanau et al., 2015). This model computes the probability of $\boldsymbol{y}$ given $\boldsymbol{x}$ as:

$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}; \theta) \quad (1)$$

where θ represents the model parameters. Given $\boldsymbol{x}$, the model first produces a sequence of hidden representations $\boldsymbol{h}_{1...T}$: $\boldsymbol{h}_t = f(\boldsymbol{y}_{<t}, \boldsymbol{x})$, where T is the length of $\boldsymbol{y}$, and f is usually an encoder-decoder network. At each time step t, the hidden representation $\boldsymbol{h}_t$ is fed to a linear projection layer $\boldsymbol{s}_t = \boldsymbol{W}\boldsymbol{h}_t + \boldsymbol{b}$ to obtain a vector of scores $\boldsymbol{s}_t$ over all possible words in the vocabulary $\mathcal{V}$. Scores are then turned into a conditional probability distribution: $p(\cdot \mid \boldsymbol{y}_{<t}, \boldsymbol{x}; \theta) = \mathrm{softmax}(\boldsymbol{s}_t)$.

The traditional maximum likelihood (ML) objective maximizes the log-likelihood of the training data $\mathcal{D} \equiv \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$ consisting of N pairs of source and target sentences:

$$\mathcal{J}_{ML}(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \log p(y_t^{(n)} \mid \boldsymbol{y}_{<t}^{(n)}, \boldsymbol{x}^{(n)}; \theta) \quad (2)$$

At test time, prefixes $\boldsymbol{y}_{<t}$ are subsequences generated by the model and therefore contain errors. By contrast, in ML training, prefixes $\boldsymbol{y}_{<t}$ are subsequences of reference translations. As a result, the model is never exposed to its own errors during training and errors accumulate at test time. This mismatch is known as the exposure bias problem (Ranzato et al., 2015).

2.1 Limitations in Scheduled Sampling

Bengio et al. (2015) introduced the scheduled sampling algorithm to address exposure bias. Scheduled sampling gradually replaces the reference words with sampled model predictions in the prefix used at training time. An annealing schedule controls the probability of using reference words vs. model predictions. The training objective remains the same as the ML objective, except for the nature of the prefix $\hat{\boldsymbol{y}}_{<t}$, which contains a mixture of reference and predicted words:

$$\mathcal{J}_{SS}(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \log p(y_t^{(n)} \mid \hat{\boldsymbol{y}}_{<t}^{(n)}, \boldsymbol{x}^{(n)}; \theta) \quad (3)$$

Despite the empirical success of scheduled sampling, one limitation is that the discontinuity of the argmax operation makes it impossible to penalize errors made in previous steps, which can lead to slow and unstable training. We address this issue using a continuous relaxation to the greedy search and sampling process, similarly to Goyal et al. (2017), which we describe in Section 2.2.

Another limitation of scheduled sampling is that it incorrectly assumes that the reference and predicted sequence are aligned by time indices which introduces additional noise to the training signal.[2] We address this problem with a novel differentiable sampling algorithm with an alignment based objective called soft aligned maximum likelihood (SAML). It is used in combination with maximum likelihood to define our training objective $\mathcal{J} = \mathcal{J}_{ML} + \mathcal{J}_{SAML}$, where $\mathcal{J}_{ML}$ is computed based on reference translations, and $\mathcal{J}_{SAML}$ is computed based on sampled translations of the same input sentences. We define $\mathcal{J}_{SAML}$ in Section 2.3.

2.2 Differentiable Sampling

To backpropagate errors made in the previous decoding steps, we use a continuous relaxation of the discrete sampling operation similar to Goyal et al. (2017), except that we use the Straight-Through (ST) Gumbel-Softmax estimator (Jang et al., 2017; Bengio et al., 2013) instead of Gumbel-Softmax (Jang et al., 2017; Maddison et al., 2014) to better simulate the scenario at inference time.[3]

The Gumbel-Softmax is derived from the Gumbel-Max trick (Maddison et al., 2014), an algorithm for sampling one-hot vector $z \in \mathbb{R}^k$ from a categorical distribution $(p_1, ..., p_k)$:

$$z = \text{one-hot}(\arg\max_i(\log p_i + \beta g_i)) \quad (4)$$

where g_i is the Gumbel noise drawn i.i.d from $\mathrm{Gumbel}(0,1)$[4], and β is a hyperparameter controlling the scale of the noise. Here, the trick is used to approximate the discontinuous argmax function with the differentiable softmax:

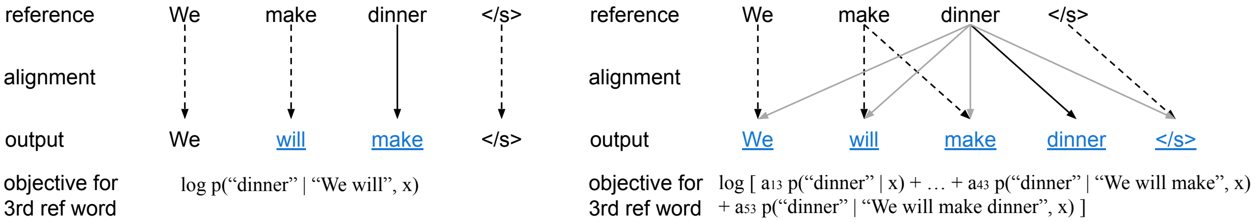
$$\tilde{z} = \mathrm{softmax}((\log p_i + \beta g_i)/\tau) \quad (5)$$

---

[2] https://nlpers.blogspot.com/2016/03/a-dagger-by-any-other-name-scheduled.html

[3] The Straight-Through estimator consistently outperforms the Gumbel-Softmax in preliminary experiments.

[4] $g_i = -\log(-\log(u_i))$ and $u_i \sim \mathrm{Uniform}(0,1)$.



(a) Scheduled Sampling Objective (b) SAML Objective

Figure 1: Difference between objectives used in scheduled sampling (left) and our approach (right), when computing the contribution to the objective of the reference word "dinner". The schedule sampling hypothesis uses a mixture of the reference (black) and sampled (blue underlined) words, while the entire hypothesis sequence is sampled in our approach.

where τ is the temperature parameter. As τ diminishes to zero, $\tilde{z}$ becomes the same as one-hot sample z.

The Straight-Through Gumbel-Softmax maintains the differentiability of the Gumbel-Softmax estimator while allowing for discrete sampling by taking different paths in the forward and backward pass. It uses argmax to get the one-hot sample z in the forward pass, but uses its continuous approximation $\tilde{z}$ in the backward pass. While ST estimators are biased, they have been shown to work well in latent tree learning (Choi et al., 2018) and semi-supervised machine translation (Niu et al., 2019).

2.3 Soft Aligned Maximum Likelihood

The soft aligned maximum likelihood (SAML) is defined as the probability that the reference can be aligned with the sampled output using a soft alignment predicted by the model:

$$P_{SAML}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} \sum_{j=1}^{T'} a_{tj} \cdot p(y_t \mid \tilde{\boldsymbol{y}}_{<j}, \boldsymbol{x}; \theta) \tag{6}$$

where T is the length of the reference sequence, T' is the length of the sampled sequence, a_{tj} is the predicted soft alignment between the reference word y_t and sampled prefix $\tilde{\boldsymbol{y}}_{<j}$.

Training with the SAML objective consists in maximizing:

$$\mathcal{J}_{SAML}(\theta) = \sum_{n=1}^{N} \log P_{SAML}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x}^{(n)}) \tag{7}$$

The conditional probability of the next word $p(y_t \mid \tilde{\boldsymbol{y}}_{<j}, \boldsymbol{x}; \theta)$ is computed as follows:

$$p(\cdot \mid \tilde{\boldsymbol{y}}_{<j}, \boldsymbol{x}; \theta) = \mathrm{softmax}(\boldsymbol{W}\tilde{\boldsymbol{h}}_j + \boldsymbol{b}) \tag{8}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are model parameters. $\tilde{\boldsymbol{h}}_j$ is the hidden representation at step j conditioned on the source sequence $\boldsymbol{x}$ and the preceding words $\tilde{\boldsymbol{y}}_{<j}$ sampled from the model distribution using differentiable sampling:

Task	sentences (K)			vocab (K)	
	train	dev	test	src	tgt
de-en	153.3	7.0	6.8	113.5	53.3
vi-en	121.3	1.5	1.3	23.9	50.0

Table 1: We evaluate on two translation tasks.

$$\tilde{\boldsymbol{h}}_j = f(\tilde{\boldsymbol{y}}_{<j}, \boldsymbol{x}) \tag{9}$$

We compute the soft alignment a_{tj} between y_t and $\tilde{\boldsymbol{y}}_{<j}$ based on the model's hidden states:

$$a_{tj} = \frac{\exp(score(\tilde{\boldsymbol{h}}_j, \boldsymbol{e}_{y_t}))}{\sum_{i=1}^{T'} \exp(score(\tilde{\boldsymbol{h}}_i, \boldsymbol{e}_{y_t}))} \tag{10}$$

where $\boldsymbol{e}_{y_t}$ is the embedding of the reference word y_t. The *score* function captures the similarity between the hidden state $\tilde{\boldsymbol{h}}_j$ and the embedding $\boldsymbol{e}_{y_t}$. We use the dot product here as it does not introduce additional parameters:

$$score(\boldsymbol{h}, \boldsymbol{e}) = \boldsymbol{h}^\top \boldsymbol{e} \tag{11}$$

Figure 1 illustrates how the resulting objective differs from scheduled sampling: (1) it is computed over sampled sequences as opposed to sequences that contain a mixture of sampled and reference words, and (2) each reference word is soft-aligned to the sampled sequence.

3 Experiments

Data We evaluate our approach on IWSLT 2014 German-English (de-en) as prior work (Goyal et al., 2017), as well as two additional tasks: IWSLT 2014 English-German (en-de) and IWSLT

Method	**Anneal**	**de-en**	**en-de**	**vi-en**
Baseline	No	$27.41_{\pm 0.26}$	$22.64_{\pm 0.13}$	$23.59_{\pm 0.13}$
+SS	Yes	$27.47_{\pm 0.28}$	$22.56_{\pm 0.17}$	$23.97_{\pm 0.39}$
+DSS	Yes	$27.30_{\pm 0.24}$	$22.47_{\pm 0.20}$	$23.68_{\pm 0.35}$
+SS	No	$22.91_{\pm 0.21}$	$17.78_{\pm 0.20}$	$19.57_{\pm 0.19}$
+SAML	No	$\mathbf{27.94}_{\pm 0.12}$	$\mathbf{23.30}_{\pm 0.19}$	$\mathbf{24.60}_{\pm 0.35}$

Table 2: BLEU scores of our approach (SAML) and three baselines including the maximum likelihood (ML) baseline, scheduled sampling (SS), and differentiable scheduled sampling (DSS). The *Anneal* column indicates whether the sampling rate is annealed. For each task, we report the mean and standard deviation over 5 runs with different random seeds. SAML achieves the best BLEU scores and is simpler to train than SS and DSS, as it requires no annealing schedule.

2015 Vietnamese-English (vi-en). For de-en and en-de, we follow the preprocessing steps in Ranzato et al. (2015). For vi-en, we use the data preprocessed by Luong and Manning (2015), with test2012 for validation and test2013 for testing. Table 1 summarizes the data statistics.

Setup Our translation models are attentional RNNs (Bahdanau et al., 2015) built on Sockeye (Hieber et al., 2017). We use bi-directional LSTM encoder and single-layer LSTM decoder with 256 hidden units, embeddings of size 256, and multilayer perceptron attention with a layer size of 256. We apply layer normalization (Ba et al., 2016) and label smoothing (0.1). We add dropout to embeddings (0.1) and decoder hidden states (0.2). For ST Gumbel-Softmax, we use temperature $\gamma = 1$ and noise scale $\beta = 0.5$. The decoding beam size is 5 unless stated otherwise. We train the models using the Adam optimizer (Kingma and Ba, 2015) with a batch size of 1024 words. We checkpoint models every 1000 updates. The initial learning rate is 0.0002, and it is reduced by 30% after 4 checkpoints without validation perplexity improvement. Training stops after 12 checkpoints without improvement. For training efficiency, we first pre-train a baseline model for each task using only $\mathcal{J}_{ML}$ and fine-tune it using different approaches. In the fine-tuning phase, we inherit all settings except that we initialize the learning rate to 0.00002 and set the minimum number of checkpoints before early stopping to 24. We fine-tune each randomly seeded model independently.

Baselines We compare our model against three baselines: (1) a standard baseline trained with the ML objective, and models fine-tuned with (2) scheduled sampling (**SS**) (Bengio et al., 2015) and (3) differentiable scheduled sampling (**DSS**) (Goyal et al., 2017). In SS and DSS, the probability of using reference words ϵ_s is annealed using inverse sigmoid decay (Bengio et al., 2015): $\epsilon_s = k/(k + \exp(i/k))$ at the i-th checkpoint with $k = 10$.

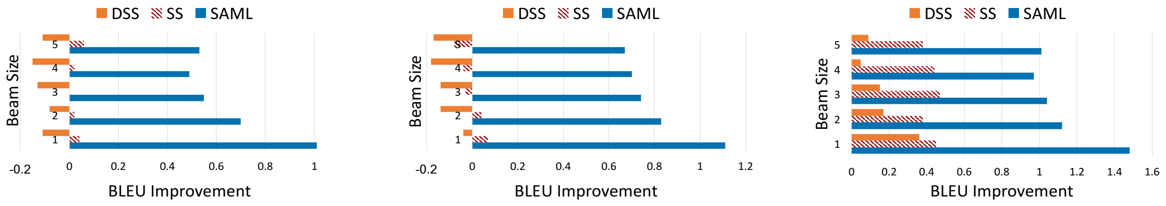
Results Table 2 shows that the SAML improves over the ML baseline by +0.5 BLEU on de-en, +0.7 BLEU on en-de, and +1.0 BLEU on vi-en task. In addition, SAML consistently improves over both the scheduled sampling and differentiable scheduled sampling on all tasks. All improvements are significant with $p < 0.002$. Interestingly, differentiable scheduled sampling performs no better than scheduled sampling in our experiments, unlike in Goyal et al. (2017).

Unlike scheduled sampling, our approach does not require an annealing schedule, and it is therefore simpler to train. We verify that the annealing schedule is needed in scheduled sampling by training a contrastive model with the same objective as scheduled sampling, but without annealing schedule (Table 2). We set the sampling rate to 0.5. The contrastive model hurts BLEU scores by at least 4.0 points compared to both the ML baseline and models fine-tuned with scheduled sampling, confirming that scheduled sampling needs the annealing schedule to work well.

We further examine the performance gain of different approaches over the baseline with varying beam sizes (Figure 2). Our approach yields larger BLEU improvements when decoding with greedy search and smaller beams, while there is no clear pattern for scheduled sampling models. These results support the hypothesis that our approach mitigates exposure bias, as it yields bigger improvements in settings where systems have fewer opportunities to recover from early errors.

4 Related Work

Daumé et al. (2009) first addressed exposure bias in an imitation learning framework by training a classifier on examples generated using a mixture of the ground truth and the model's current predictions. DAgger (Ross et al., 2011) is a similar algorithm which differs in how the training examples are generated and aggregated. Both al-



(a) de-en

(b) en-de

(c) vi-en

Figure 2: Improvements from our method (SAML), scheduled sampling (SS), and differentiable scheduled sampling (DSS) over the maximum likelihood (ML) baseline when decoding with varying beam sizes (average of 5 runs). The SAML model consistently yields the largest improvements with smaller beams.

gorithms require an expert policy, which produces the best next token given any model predicted prefix, and assume that policy can be efficiently computed from the reference. However, for structured prediction tasks such as machine translation with large vocabulary and complex loss functions, it is intractable to find the best next token given any prefix. For time series modeling, the Data as Demonstrator algorithm (Venkatraman et al., 2015) derives the expert policy directly from the reference sequences which are aligned with the sampled sequences at each time step. Scheduled sampling algorithms (Bengio et al., 2015; Goyal et al., 2017) use the same strategy to train neural sequence-to-sequence models for a broader range of language generation tasks, even though the time alignment between reference and sampled sequences does not hold. Leblond et al. (2018) proposed to complete a predicted prefix with all possible reference suffixes and picking the reference suffix that yields the highest BLEU-1 score. However, they found that this approach performs well only when the prefix is close to the reference.

Reinforcement learning (RL) algorithms (Bahdanau et al., 2016; Sutton and Barto, 2018; Van Hasselt et al., 2016) address exposure bias by directly optimizing a sentence-level reward for the model generated sequences. Evaluation metrics such as BLEU can be used as rewards, but they are discontinuous and hard to optimize. Techniques such as policy gradient (Williams, 1992) and actor-critic (Sutton and Barto, 2018; Degris et al., 2012) are thus required to find an unbiased estimation of the gradient to optimize the model. Due to the high variance of the gradient estimation, training with RL can be slow and unstable (Henderson et al., 2018; Wu et al., 2018). Recent alternatives use data augmentation to incorporate the sentence-level reward into the training objective more efficiently (Norouzi et al., 2016).

Finally, our SAML loss shares the idea of flexible reference word order with the bag-of-word loss introduced by Ma et al. (2018) to improve source coverage. However, their loss is computed with teacher forcing and therefore does not address exposure bias.

5 Conclusion

We introduced a differentiable sampling algorithm which exposes a sequence-to-sequence model to its own predictions during training and compares them to reference sequences flexibly to backpropagate reliable error signals. By soft aligning reference and sampled sequences, our approach consistently improves BLEU over maximum likelihood and scheduled sampling baselines on three IWSLT tasks, with larger improvements for greedy search and smaller beam sizes. Our approach is also simple to train, as it does not require any sampling schedule.

Acknowledgments

We thank the anonymous reviewers, Amr Sharaf, Naomi Feldman, Hal Daumé III and the CLIP lab at UMD for helpful comments. This research is supported in part by an Amazon Web Services Machine Learning Research Award and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *CoRR*, abs/1607.07086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3th International Conference on Learning Representations*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*, pages 1171–1179.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5094–5101. AAAI Press.

William W. Cohen and Victor R. Carvalho. 2005. Stacked Sequential Learning. In *Proceedings of the IJCAI*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Thomas Degris, Patrick M Pilarski, and Richard S Sutton. 2012. Model-free reinforcement learning with continuous action in practice. In *American Control Conference (ACC), 2012*, pages 2177–2182. IEEE.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 366–371. Association for Computational Linguistics.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations*.

Matti Kääriäinen and John Langford. 2006. Lower Bounds for Reductions. In *Talk at the Atomic Learning Workshop*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations*.

John Langford and Bianca Zadrozny. 2005. Relating reinforcement learning performance to classification performance. In *Proceedings of the 22nd international conference on Machine learning*, pages 473–480. ACM.

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. 2018. SEARNN: Training RNNs with global-local losses. In *International Conference on Learning Representations*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 332–338. Association for Computational Linguistics.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems 27*, pages 3086–3094. Curran Associates, Inc.

Xing Niu, Weijia Xu, and Marine Carpuat. 2019. Bidirectional differentiable input reconstruction for low-resource neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1723–1731. Curran Associates, Inc.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 2, page 5. AAAI Press.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3024–3030. AAAI Press.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621. Association for Computational Linguistics.

Reinforcement Learning based Curriculum Optimization for Neural Machine Translation

Gaurav Kumar
Johns Hopkins University
gkumar@cs.jhu.edu

George Foster, Colin Cherry, Maxim Krikun
Google AI
{fosterg, colincherry, krikun}
@google.com

Abstract

We consider the problem of making efficient use of heterogeneous training data in neural machine translation (NMT). Specifically, given a training dataset with a sentence-level feature such as noise, we seek an optimal *curriculum*, or order for presenting examples to the system during training. Our curriculum framework allows examples to appear an arbitrary number of times, and thus generalizes data weighting, filtering, and fine-tuning schemes. Rather than relying on prior knowledge to design a curriculum, we use reinforcement learning to learn one automatically, jointly with the NMT system, in the course of a single training run. We show that this approach can beat uniform baselines on Paracrawl and WMT English-to-French datasets by +3.4 and +1.3 BLEU respectively. Additionally, we match the performance of strong filtering baselines and hand-designed, state-of-the-art curricula.

1 Introduction

Machine Translation training data is typically heterogeneous: it may vary in characteristics such as domain, translation quality, and degree of difficulty. Many approaches have been proposed to cope with heterogeneity, such as filtering (Duh et al., 2013) or down-weighting (Wang et al., 2017) examples that are likely to be noisy or out of domain. A powerful technique is to control the curriculum—the order in which examples are presented to the system—as is done in fine-tuning (Freitag and Al-Onaizan, 2016), where training occurs first on general data, and then on more valuable in-domain data. Curriculum based approaches generalize data filtering and weighting[1] by allowing examples to be visited multiple times or not at all; and they additionally potentially enable steering the training trajectory toward a better global optimum than might be attainable with a static attribute-weighting scheme.

Devising a good curriculum is a challenging task that is typically carried out manually using prior knowledge of the data and its attributes. Although powerful heuristics like fine-tuning are helpful, setting hyper-parameters to specify a curriculum is usually a matter of extensive trial and error. Automating this process with meta-learning is thus an attractive proposition. However, it comes with many potential pitfalls such as failing to match a human-designed curriculum, or significantly increasing training time.

In this paper we present an initial study on meta-learning an NMT curriculum. Starting from scratch, we attempt to match the performance of a successful non-trivial *reference curriculum* proposed by Wang et al. (2018), in which training gradually focuses on increasingly cleaner data, as measured by an external scoring function. Inspired by Wu et al. (2018), we use a reinforcement-learning (RL) approach involving a learned agent whose task is to choose a corpus bin, representing a given noise level, at each NMT training step. A challenging aspect of this task is that choosing only the cleanest bin is sub-optimal; the reference curriculum uses all the data in the early stages of training, and only gradually anneals toward the cleanest. Furthermore, we impose the condition that the agent must learn its curriculum in the course of a single NMT training run.

We demonstrate that our RL agent can learn a curriculum that works as well as the reference, obtaining a similar quality improvement over a random-curriculum baseline. Interestingly, it does so using a different strategy from the reference. This result opens the door to learning more sophisticated curricula that exploit multiple data at-

---

[1] Assuming integer weights.

Proceedings of NAACL-HLT 2019, pages 2054–2061
Minneapolis, Minnesota, June 2 - June 7, 2019.

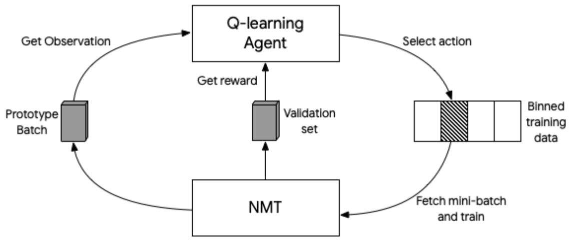


Figure 1: The agent's interface with the NMT system.

tributes and work with arbitrary corpora.

2 Related Work

Among the very extensive work on handling heterogeneous data in NMT, the closest to ours are techniques that re-weight (Chen et al., 2017) or re-order examples to deal with domain mismatch (van der Wees et al., 2017; Sajjad et al., 2017) or noise (Wang et al., 2018).

The idea of a curriculum was popularized by Bengio et al. (2009), who viewed it as a way to improve convergence by presenting heuristically-identified easy examples first. Several recent papers (Kocmi and Bojar, 2017; Zhang et al., 2019; Platanios et al., 2019) explore similar ideas for NMT, and verify that this strategy can reduce training time and improve quality.

Work on meta-learning a curriculum originated with Tsvetkov et al. (2016), who used Bayesian optimization to learn a linear model for ranking examples in a word-embedding task. This approach requires a large number of complete training runs, and is thus impractical for NMT. More recent work has explored bandit optimization for scheduling tasks in a multi-task problem (Graves et al., 2017), and reinforcement learning for selecting examples in a co-trained classifier (Wu et al., 2018). Finally, Liu et al. (2018) apply imitation learning to actively select monolingual training sentences for labeling in NMT, and show that the learned strategy can be transferred to a related language pair.

3 Methods

The attribute we choose to learn a curriculum over is noise. To determine a per-sentence noise score, we use the *contrastive data selection* (CDS) method defined in Wang et al. (2018). Given the parameters θ_n of an NMT model trained on a noisy corpus, and parameters θ_c of the same model fine-tuned on a very small trusted corpus, the score $s(e,f)$ for a translation pair e,f is defined as:

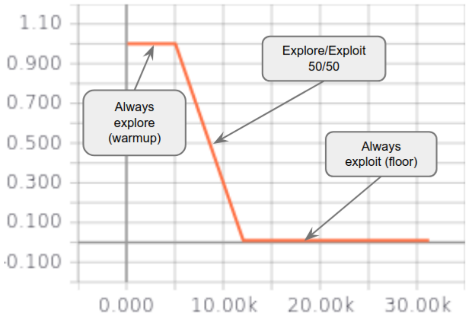


Figure 2: Linearly-decaying ϵ-greedy exploration.

$$s(e,f) = \log p_{\theta_c}(f|e) - \log p_{\theta_n}(f|e) \qquad (1)$$

Wang et al. (2018) show that this correlates very well with human judgments of data quality. They use the CDS score in a heuristic, online schedule that slowly anneals from sampling mini-batches from all the training data to sampling only from the highest-scoring (cleanest) data. Our goal is to replace this heuristic curriculum with a learned one.

Q-learning for NMT Curricula

Our agent uses deep Q-learning (DQN) (Mnih et al., 2015) which is a model-free reinforcement learning procedure. The agent receives an *observation* from the environment and conditions on it to produce an *action* which is executed upon the environment. It then receives a *reward* representing the goodness of the executed action. The agent chooses actions according to a state-action value (Q) function, and attempts to learn the Q-function so as to maximize expected total rewards.

In our setup, the environment is the NMT system and its training data, as illustrated in Figure 1. We divide the training data into a small number of equal-sized *bins* according to CDS scores. At each step, the agent selects a bin (action) from which a mini-batch is sampled to train the NMT system.

Our RL agent must balance exploration (choosing an action at random) versus exploitation (choosing the action which maximizes the Q-function). In our setup, this is done using a linearly-decaying ϵ-greedy exploration strategy (Figure 2). This strategy has three phases: (1) The warmup period where we always explore; (2) the decay period where the probability of exploration decreases and exploitation increases; (3) the floor where we almost always exploit. Since we do not want to exploit an uninformed Q-function, the duration of exploration needs to be set carefully. In our experiments, we found that longer decays were useful and the best performance was achieved when the decay was set to about 50% of the expected NMT training steps.

Observation Engineering

The *observation* is meant to be a summary of the state of the environment. The NMT parameters are too numerous to use as a sensible observation at each time step. Inspired by Wu et al. (2018), we propose an observation type which is a function of the NMT system's current performance at various levels of noise. We first create a *prototype batch* by sampling a fixed number of prototypical sentences from each bin of the training data. At each time step, the observation is the vector containing sentence-level log-likelihoods produced by the NMT system for this prototype batch.

Since the observations are based on likelihood, a metric which aggressively decays at the beginning of NMT training, we use an NMT warmup period to exclude this period from RL training. Otherwise, the initial observations would be unlike any that occur later.

Reward Engineering

Our objective is to find a curriculum which maximizes the likelihood of the NMT system on a development set. The RL *reward* that directly corresponds to this goal would be the highest likelihood value reached during an NMT training run. However, as we use only one NMT training run, having a single reward per run is infeasible. To provide a denser signal to the RL agent, we define the reward at a step to be the change in likelihood since the most recent previous step for which development-set likelihood is available. This has the desired property that the sum of per-step rewards maximized by the RL agent is equal to the NMT maximum-likelihood objective (on development data). We rely on the WMT warmup period described in the previous section to eliminate spuriously large rewards at the beginning of training.

4 Experimental Setup

Our NMT model is similar to RNMT+ (Chen et al., 2018), but with only four layers in both encoder and decoder. Rewards (dev-set log-likelihood) are provided approximately every 10 training steps by an asynchronous process.

We use the DQN agent implementation in Dopamine,[2] which includes an experience replay buffer to remove temporal correlations from the observations, among other DQN best practices. Due to the sparse and asynchronous nature of our rewards, we store observation, action transitions in a temporary buffer until a new reward arrives. At this point, transitions are moved from the temporary buffer to the DQN agent's replay buffer. The RL agent is trained after each NMT training step by sampling an RL mini-batch from the replay buffer. Our RL hyper-parameter settings are listed in the appendix.

Following Wang et al. (2018), we use the Paracrawl and WMT English-French corpora for our experiments. These contain 290M and 36M training sentences respectively. WMT is relatively clean, while a large majority of Paracrawl sentence pairs contain noise. We process both corpora with BPE, using a vocabulary size of 32k. Both corpora are split into 6 equal-sized bins according to their noise level, as provided by CDS score. In both settings, the WMT newstest 2010-2011 corpus is used as trusted data for CDS scores, which are computed using the models and procedure described in Wang et al. (2018). For the *prototype batch* used to generate observations, we extracted the 32 sentences whose CDS scores are closest to the mean in each bin, giving a total of 192 sentences. We use WMT 2012-2013 for development and WMT 2014 for test, and report tokenized, naturally-cased BLEU scores from the test checkpoint closest to the highest-BLEU dev checkpoint. To combat variance caused by sampling different batches per bin (which produces somewhat different results even when bins are visited in fixed order), all models were run twice with different random seeds, and the model with the best score on the dev set was chosen.

5 Results

Our results are presented in Table 1. **Uniform baselines** consist of:

- *Uniform* – standard NMT training
- *Uniform (6-bins)* – sample a bin uniformly at random, and then sample a mini-batch from that bin

---

[2] `github.com/google/dopamine`

	Paracrawl	**WMT**
Uniform baselines		
Uniform	34.1	37.1
Uniform (6-bins)	34.8	-
Uniform (bookends)	35.0	34.8
Heuristic baselines		
Filtered (20%/33%)	37.0	38.3
Fixed ϵ-schedule	36.9	37.7
Online	37.5	37.7
Learned curricula		
Q-learning (bookends)	36.8	36.3
Q-learning (6-bins)	37.5	38.4

Table 1: BLEU scores on Paracrawl and WMT En-Fr datasets with uniform, heuristic and learned curricula.

- *Uniform (bookends)* – as Uniform (6-bins) but uniformly sampling over just the best and worst bin.

Surprisingly, 6-bins performs better than the standard NMT baseline. We hypothesize that this can be attributed to more homogeneous mini-batches.

Heuristic baselines are:

- *Filtered* – train only on the highest-quality data as determined by CDS scores: top 20% of the data for Paracrawl, top 33% for WMT.
- *Fixed ϵ-schedule* – we use the ϵ-decay strategy of our best RL experiment, but always choose the cleanest bin when we exploit.
- *Online* – the online schedule from Wang et al. (2018) adapted to the 6-bin setting. We verified experimentally that our performance matched the original schedule, which did not use hard binning.

Learned curricula were trained over 2 bookend (worst and best) bins and all 6 bins. On the Paracrawl dataset, in the 2-bin setting, the learned curriculum beats all uniform baselines and almost matches the optimized filtering baseline.[3] With 6-bins, it beats all uniform baselines by up to 2.5 BLEU and matches the hand-designed online baseline of Wang et al. (2018). On WMT, with 2 bins, the learned curriculum beats the 2-bin baseline, but not the uniform baseline over all data. With 6 bins, the learned curriculum beats the uniform baseline by 1.5 BLEU, and matches the filtered baseline, which in this case outperforms the online curriculum by 0.6 BLEU.

Observation Reward	Default	Fixed
Default	**37.5**	37.5
Fixed	32.5	-

Table 2: BLEU scores on ablation experiments with fixed rewards or observations on the Paracrawl En-Fr dataset.

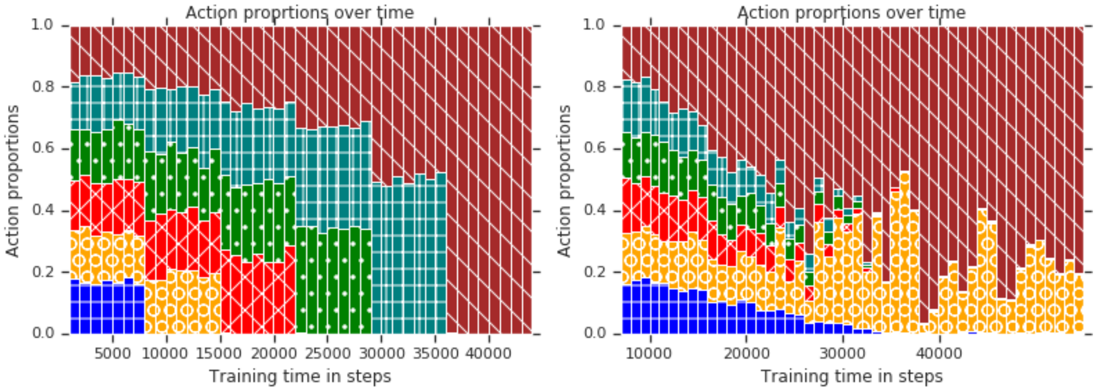
Our exploration strategy for Q-learning (see Figure 2) forces the agent to visit all bins during initial training, and only gradually rely on its learned policy. This mimics the gradual annealing of the online curriculum, so one possibility is that the agent is simply choosing the cleanest bin whenever it can, and its good performance comes from the enforced period of exploration. However, the fact that the agent beats the fixed ϵ-schedule (see Table 1) described above on both corpora makes this unlikely.

6 Analysis

Task-specific reward and observation engineering is critical when building an RL model. We performed ablation experiments to determine if the rewards and observations we have chosen contain information which aids us in the curriculum learning task. Table 2 shows the results of our experiments. The fixed reward experiments were conducted by replacing the default delta-perplexity based reward with a static reward which returns a reward of one when the cleanest bin was selected and zero otherwise. The fixed observation experiments used a static vector of zeroes as input at all time steps. Using fixed observations matches the performance of dynamic observations, from which we can draw two conclusions. First, the agent's good performance is due to associating higher rewards with better bins, but it learns to do so slowly (partly modulated by its ϵ-greedy schedule) so that it avoids the sub-optimal strategy of choosing only the best bin. Second, its ability to distinguish among bins is not impeded by the use of an observation vector that slowly evolves through time and never returns to previous states.

---

[3] The clean data available in the 2-bin setup is limited to the best bin (16%), while filtering uses slightly more data (20%).



(a) Online

(b) RL learned

Figure 3: Online policy from Wang et al. (2018) compared to the RL policy. Each color/pattern represents a bin (blue is the noisiest bin, dark red is the cleanest; bins lower on the vertical axis contain more noise) and length along the vertical axis is proportional to the number of times each bin was selected at a given step during training.

6.1 What did the agent learn?

Figure 3 shows a coarse visualization of the hand-optimized policy of Wang et al. (2018), adapted to our 6-bin scenario, compared to the Q-learning policy on the same scenario. The former, by design, telescopes towards the clean bins. Note that the latter policy is masked by the agent's exploration schedule, which slowly anneals toward nearly complete policy control, beginning at step 30,000. After this point, the learned policy takes over and continues to evolve. This learned policy has little in common with the hand-designed one. Instead of focusing on a mixture of the clean bins, it focuses on the cleanest bin and the second-to-noisiest. We hypothesize that returning to the noisy bin acts as a form of regularization, though this requires further study.

7 Conclusion

We have presented a method to learn a curriculum for presenting training samples to an NMT system. Using reinforcement learning, our approach learns the curriculum jointly with the NMT system during the course of a single NMT training run. Empirical analysis on the Paracrawl and WMT English-French corpora shows that this approach beats the uniform sampling and filtering baselines. In addition, we were able to match a state-of-the-art hand designed curriculum on Paracrawl and beat it on WMT.

We see this a first step toward enabling NMT systems to manage their own training data. In the future, we intend to improve our approach by eliminating the static exploration schedule and binning strategy, and extend it to handle additional data attributes such as domain, style, and grammatical complexity.

Acknowledgements

The authors would like to thank Wei Wang for his advice and help in replicating the CDS baselines.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86. Association for Computational Linguistics.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *The 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 678–683, Sofia, Bulgaria.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1311–1320.

Tom Kocmi and Ondrej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533*.

Ming Liu, Wray L. Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 334–344.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. 2019. Competence-based Curriculum Learning for Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. *arXiv preprint arXiv:1708.08712*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 130–139. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488. Association for Computational Linguistics.

Wei Wang, Taro Watanabe, Macduff Hughes, Tetsuji Nakagawa, and Ciprian Chelba. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 133–143. Association for Computational Linguistics.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410. Association for Computational Linguistics.

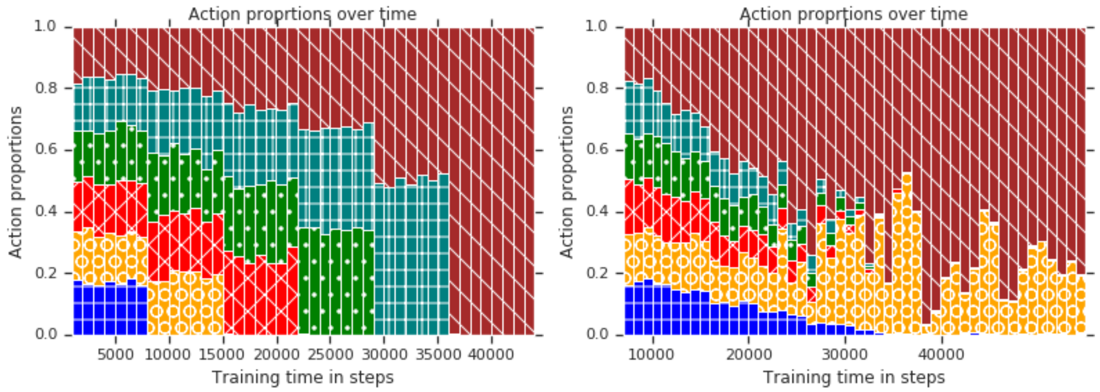
Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1252–1262.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

A Appendix

A.1 Q-learning hyper-parameters

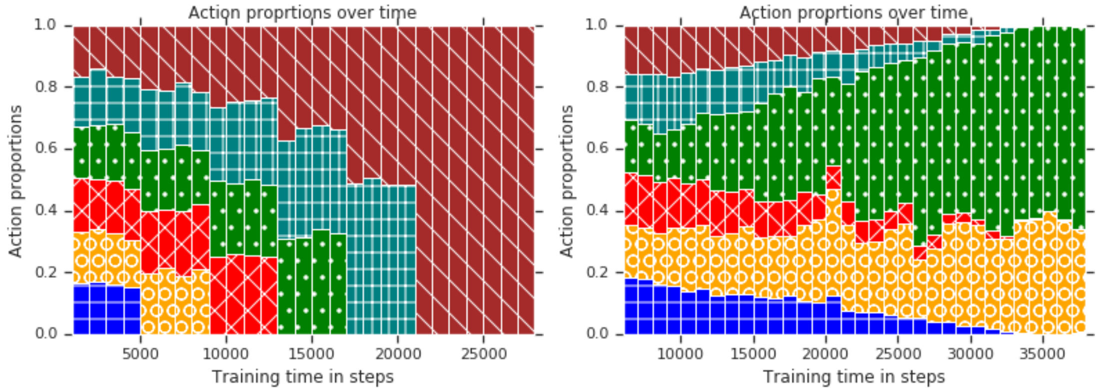
- Observations: We sample 32 prototype sentences from each bin to create a *prototype batch* of 192 sentences.
- Q-networks: The two Q-networks were MLPs with 2 x 512-d hidden layers each. A $\tanh$ activation function was used.
- RL optimizer: We used RMSProp with a learning rate of 0.00025 and a decay of 0.95 and no momentum.
- NMT warmup : 5000 steps (no transitions from this period are recorded).
- Stack size: We do not stack our observations for the RL agent (i.e., stack size = 1).
- Exploration strategy : We use a linearly decaying epsilon function with decay period set to 25k steps. The decay floor was set to 0.01.
- Discount gamma : 0.99
- Update horizon : 2
- Minimum number of transitions in replay buffer before training starts: 3000



(a) Telescoping

(b) RL learned

Figure 4: Policies learned by the RL agent on the Paracrawl En-Fr corpus compared against the telescoping policy from Wang et al. (2018). Lower bins on the vertical axis contain more noise.



(a) Telescoping

(b) RL learned

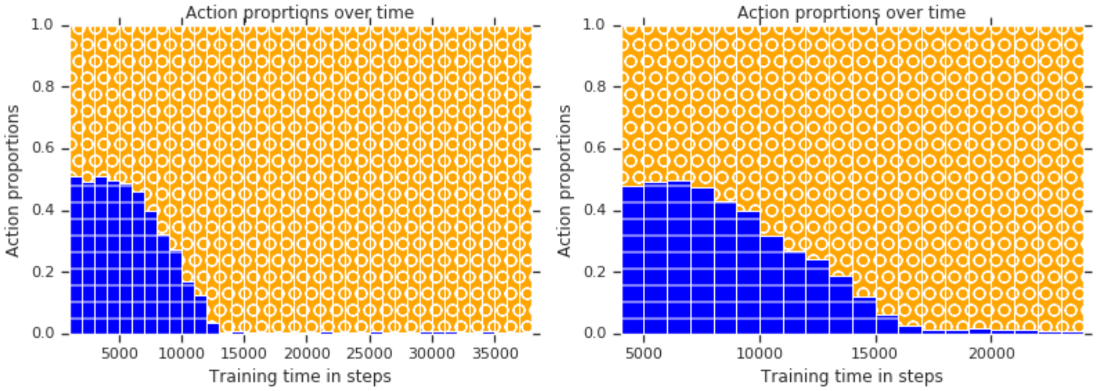
Figure 5: Policies learned by the RL agent on the WMT En-Fr corpus compared against the telescoping policy from Wang et al. (2018). Lower bins on the vertical axis contain more noise.

- Update period (how often the online Q-network is trained): 4 steps
- Target update period (how often the target Q-network is trained): 100 steps
- The window for the delta-perplexity reward was 1.

A.2 Learned Policies

Figures 4, 5 and 6 show coarse representations of the policies learned by the Q-learning agent on the Paracrawl and WMT English-French datasets. Each column in the figures represents the relative proportion of actions taken (bins selected) averaged over a thousand steps and the actions go from noisy to clean on the y-axis. Each policy starts from a uniform distribution over actions. Some salient aspects of the learned policies are listed below.

1. All learned curricula differ significantly from the hand-designed policies.
2. The RL curriculum learned for Paracrawl (Figure 4) focus on two bins during exploitation (choose action using the trained Q-function). Surprisingly, these are not the two cleanest bins but a mixture of the cleanest and the second-to-noisiest bin.
3. The RL curriculum learned for WMT (Figure 4) is closer to a uniform distribution over actions for a long duration. This makes sense since the data from WMT is mostly homogeneous with respect to noise. When the agent does decide to exploit some bins more often, they are not the cleanest ones, but the 1st and 4th bin instead.
4. Figure 6 shows the policies learned on the bookend task for Paracrawl and WMT; the



(a) RL Learned (Paracrawl)

(b) RL learned (WMT)

Figure 6: Policies learned by the RL agent on the 2-bin task on the Paracrawl and WMT En-Fr datasets. Lower bins on the vertical axis contain more noise.

only two bins available contain the noisiest and cleanest portion of the corpus. The RL agent very quickly learns that there is an optimal bin to choose in this task and converges to consistently exploiting it. We consider this a sanity check of curriculum learning methods.

Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation

Brian Thompson[†] **Jeremy Gwinnup**[°] **Huda Khayrallah**[†] **Kevin Duh**[†] **Philipp Koehn**[†]
[†]Johns Hopkins University, [°]Air Force Research Laboratory
{brian.thompson, huda, phi}@jhu.edu,
kevinduh@cs.jhu.edu,
jeremy.gwinnup.1@us.af.mil

Abstract

Continued training is an effective method for domain adaptation in neural machine translation. However, in-domain gains from adaptation come at the expense of general-domain performance. In this work, we interpret the drop in general-domain performance as *catastrophic forgetting* of general-domain knowledge. To mitigate it, we adapt Elastic Weight Consolidation (EWC)—a machine learning method for learning a new task without forgetting previous tasks. Our method retains the majority of general-domain performance lost in continued training without degrading in-domain performance, outperforming the previous state-of-the-art. We also explore the full range of general-domain performance available when some in-domain degradation is acceptable.

1 Introduction

Neural Machine Translation (NMT) performs poorly without large training corpora (Koehn and Knowles, 2017). Domain adaptation is required when there is sufficient data in the desired language pair but insufficient data in the desired *domain* (the topic, genre, style or level of formality). This work focuses on the supervised domain adaptation problem where a small in-domain parallel corpus is available for training. Continued training (Luong and Manning, 2015; Sennrich et al., 2015) (also called fine-tuning), where a model is first trained on general-domain data and then domain adapted by training on in-domain data, is a popular approach in this setting as it leads to empirical improvements in the targeted domain.

One downside of continued training is that the adapted model's ability to translate general-domain sentences is severely degraded during adaptation (Freitag and Al-Onaizan, 2016). We interpret this drop in general-domain performance as *catastrophic forgetting* (Goodfellow et al., 2013) of general-domain translation knowledge. Degradation of general-domain performance may be problematic when the domain adapted NMT system is used to translate text outside its target domain, which can happen if there is a mismatch between the data available for domain-specific training and the test data. Poor performance may also concern end users of these MT systems who are expecting good performance on 'easy' generic sentences.[1]

Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) is a method for training neural networks to learn a new task without forgetting previously learned tasks. We extend EWC to continued training in NMT (see §3): Our first task is to translate general-domain sentences, and our second is to translate domain-specific sentences (without forgetting how to translate general-domain sentences). EWC works by adding a per-parameter regularizer, based on the Fisher Information matrix, while training on the second task. At a high level, the regularization term keeps parameters which are important to general-domain performance close to the initial general-domain model values during continued training, while allowing parameters less important to general-domain performance to adapt more aggressively to the in-domain data.

We show that when adapting general-domain models to the domain of patents, EWC can substantially improve the retention of general-domain performance (up to 18.1 BLEU) without degrading in-domain translation quality. Our proposed method outperforms the previous state-of-the-art method (Dakwale and Monz, 2017) at retaining general-domain performance while adapting to a new domain.

---

[1] See Cadwell et al. (2018) and Porro Rodriguez et al. (2017) for discussions about lack of trust in MT.

Proceedings of NAACL-HLT 2019, pages 2062–2068
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related Work

A few prior studies address the drop in general-domain NMT performance during continued training. Freitag and Al-Onaizan (2016) found that ensembling general- and in-domain models provides most of the in-domain gain from continued training while retaining most of the general-domain performance. Ensembling doubles memory and computational requirements at translation time, which may be impractical for some applications and does not address our more fundamental goal of building a single model that is robust across domains. Chu et al. (2017) found that mixing general-domain data with the in-domain data used for continued training improved general-domain performance of the resulting models, at the expense of training time.

Dakwale and Monz (2017) share our goal of improving the general-domain performance of continued training. They introduce two novel approaches which use the initial, general-domain model to supervise the in-domain model during continued training. The first, *multi-objective fine-tuning*, which they denote MCL, trains the network with a joint objective of standard log-likelihood loss plus a second term based on knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016) of the general-domain model. The second, *multiple-output layer fine tuning*, adds new parameters to the output layer during continued training that are specific to the new domain. They found both methods performed similarly, significantly outperforming ensembling in the more challenging case where domain shift is significant, so we select the simpler MCL as our baseline.

We do not assume that the domain of input sentences is known, thus we do not compare to methods such as LHUC (Vilar, 2018). Our work applies a regularization term to continued training, similar to Miceli Barone et al. (2017) and Khayrallah et al. (2018), but for the purpose of retaining general-domain performance as opposed to improving in-domain performance.

3 Method

Compared to Kirkpatrick et al. (2017), we present a more general derivation of EWC to address the fact that our tasks are not independent. We also show that the diagonal of the Fisher matrix used in EWC is intractable to compute for sequence-to-sequence models with large vocabularies. Instead we propose to approximate it with the diagonal of the *empirical Fisher* (Martens, 2014), which can be computed efficiently using gradients from back-propagation.

At a high level, our method works as follows:

1. Train on the general-domain data, resulting in parameters $\hat{\theta}^G$.
2. Compute the diagonal of the empirical Fisher matrix $\bar{F}$. $\bar{F}_{i,i}$ estimates how important the i^{th} parameter $\hat{\theta}_i^G$ is to the general-domain translation task.
3. Initialize parameters to $\hat{\theta}^G$ and train on in-domain data, using an EWC regularization term which incorporates the diagonal of $\bar{F}$.

Intuitively, the regularization term during continued training keeps a parameter θ_i close to corresponding general-domain parameter $\hat{\theta}_i^G$ if the model's general-domain performance is sensitive to that parameter (i.e., large $\bar{F}_{i,i}$). Parameters to which general-domain performance is less sensitive (i.e., small $\bar{F}_{i,i}$) are allowed to be updated more aggressively to fit the in-domain data.

3.1 Bayesian Rationale for EWC

For the following discussions, let X be the set of all well-formed source sentences and Y be the set of all possible sequences of target words. Training data D consists of translations (x, y). We assume $x \in X$ is drawn from a true underlying distribution of source sentences Q_x, and $y \in Y$ is drawn from a true conditional distribution of correct translations $Q_{y|x}$. Our model, parameterized by θ, computes the conditional probability $P_{y|x} \triangleq P(y|x;\theta)$, which estimates $Q_{y|x}$. Our dataset D is assumed to have come from two distinct tasks: general-domain translation with data D^G and in-domain translation with data D^S(domain-specific). Without loss of generality, $p(D){=}p(D^G)p(D^S|D^G)$. Applying Bayes rule to $\log p(\theta|D)$ and simplifying gives:

$$\log p(\theta|D) = \log p(D^S|D^G,\theta) + \log p(\theta|D^G) - \log p(D^S|D^G) \quad (1)$$

We aim to maximize Equation 1 for θ:

$$\hat{\theta}^* = \arg\max_{\theta} \left[\log p(D^S|D^G,\theta) + \log p(\theta|D^G)\right] \quad (2)$$

3.2 Approximating $\log p(\boldsymbol{\theta}|D^G)$

To efficiently compute Equation 2, we first approximate $p(\theta|D^G)$ as a multivariate Gaussian[2] with mean $\hat{\theta}^G$, obtained by training the network on D^G with standard negative log likelihood (NLL) loss, and diagonal precision matrix (inverse of the covariance matrix) given by the diagonal of the Fisher Information Matrix F:

$$F = E_{P_{x,y}}\left[\nabla \log p(x,y|\theta) \nabla \log p(x,y|\theta)^T\right]$$

$$= E_{Q_x}\left[E_{P_{y|x}}\left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^T\right]\right]$$

This is the expected variance of the likelihood function's gradient at θ.[3] The magnitude of $F_{i,i}$ indicates the model's sensitivity to parameter θ_i, on the general-domain translation task. Note that the first expectation is taken with respect to the true distribution of x and can be approximated by training samples. The second expectation is taken with respect to the model distribution $P_{y|x}$, which is impractical for a large sequence-to-sequence model as it requires summing over all possible output sequences.

We approximate the true Fisher with the empirical Fisher $\bar{F}$ (Martens, 2014), where y is not enumerated but fixed to be the training labels:

$$\bar{F} = \frac{1}{|D^G|} \sum_{(x,y)\in D^G} \nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^T$$

Thus we approximate maximizing $\log p(\theta|D^G)$ in Equation 2 by minimizing $\sum_i \bar{F}_{i,i}\left[\theta_i - \hat{\theta}_i^G\right]^2$. Note that the diagonal of $\bar{F}$ is easily computed from backpropagation gradients.

3.3 Approximating $\log p(D^S|D^G, \boldsymbol{\theta})$

Tasks are assumed to be independent in the original EWC work (Kirkpatrick et al., 2017), which is unrealistic in the continued training scenario since both tasks are translation in the same language.[4] Since we assume source sentences in D^G and D^S are sampled independently, all dependencies can be attributed to $Q_{y|x}$, representing knowledge of translation (i.e., $D^G \perp\!\!\!\perp D^S \mid Q_{y|x}$). $Q_{y|x}$ is unknown, so we approximate it with our general-domain model ($\hat{\theta}^G$). Furthermore, we will regularize continued training such that θ stays in a region near $\hat{\theta}^G$. Thus we assume $D^G \perp\!\!\!\perp D^S \mid \theta$ during continued training. This allows us to approximate $\log p(D^S|D^G,\theta)$ in Equation 2 with $\log p(D^S|\theta)$, which is simply the likelihood function on D^S.

General-domain			WIPO Patents		
De	Ru	Zh	De	Ru	Zh
323 M	730 M	584 M	3.2 M	0.81 M	0.80 M

Table 1: # English words in the training corpora.

3.4 EWC Loss

Combining the approximations above results in the EWC loss used in continued training:

$$L_{EWC}(\theta) = L_{NLL}^S(\theta) + \lambda \sum_i \bar{F}_{i,i}\left[\theta_i - \hat{\theta}_i^G\right]^2 \quad (3)$$

Where $L_{NLL}^S(\theta)$ is the standard NLL loss on D^S and λ is a hyper-parameter which weights the importance of the general-domain task. Note that the left-hand side of Equation 3 is still the loss over both the general- and in-domain translation tasks, but the right-hand side is based only on in-domain data. All information from the general-domain data has been collapsed into the second term, which is in the form of a regularizer.

4 Experiments

Our general-domain training data is the concatenation of the parallel portions of the WMT17 news translation task (Bojar et al., 2017) and OpenSubtitles18 (Lison et al., 2018) corpora. For De↔En and Ru↔En, we use `newstest2017` and the final 2500 lines of OpenSubtitles as our test set. We use `newstest2016` and the penultimate 2500 lines of OpenSubtitles as the development set. For Zh↔En, we use the final and penultimate 4000 lines of the UN portion of the WMT data and the final and penultimate 2500 lines of OpenSubtitles as our test and development sets, respectively.

We use the World Intellectual Property Organization (WIPO) COPPA-V2 corpus (Junczys-Dowmunt et al., 2016) as our in-domain dataset. The WIPO data consist of parallel sentences from international patent application abstracts. WIPO De↔En data are large enough to train strong in-domain systems (Thompson et al., 2018), so we truncate to 100k lines to simulate a more interesting domain adaptation scenario.

We reserve 3000 lines each for in-domain development and test sets. We apply the Moses tok-

---

[2] For background, see MacKay (1992).

[3] See Martens (2014) for detailed derivation.

[4] The fact that continued training works is strong evidence that the in-domain translations are *not* independent of the general-domain translations.

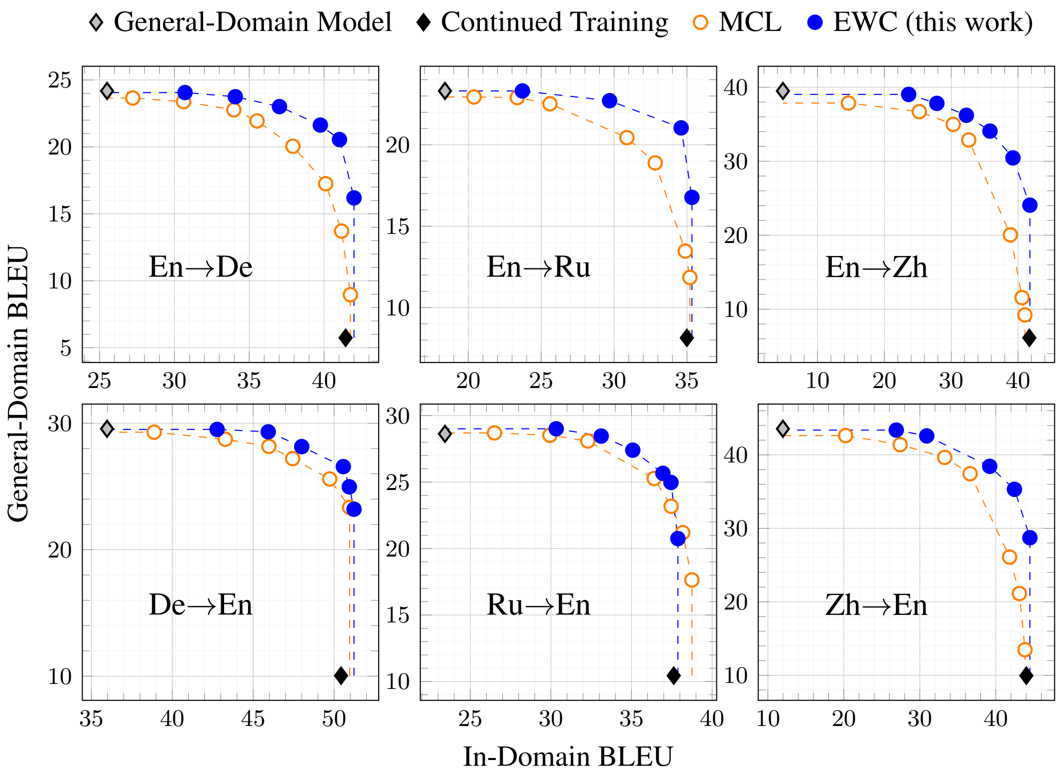


Figure 1: Performance trade-off for MCL and EWC: Convex hull of grid search over learning rate and regularization amount. x-axis is in-domain BLEU and y-axis is general-domain BLEU, so the desired operating point is the top right corner. Initial general-domain model (GD) and continued training (CT) points are shown for comparison.

enizer (Koehn et al., 2007) and byte-pair encoding (BPE) (Sennrich et al., 2016). We train separate BPE models for the source and target languages, each with a vocabulary size of approximately 30k. BPE is trained on the out-of-domain corpus only and then applied to the training, development, and test data for both out-of-domain and in-domain datasets. Token counts for corpora are shown in Table 1.

We implemented[5] both EWC and MCL in Sockeye (Hieber et al., 2017). To avoid floating point issues, we normalize the empirical Fisher diagonal to have a mean value of 1.0 instead of dividing by the number of sentences. For efficiency, we compute gradients for a batch of sentences prior to squaring and accumulating them. Fisher regularization is implemented as weight decay (towards $\hat{\theta}^G$) in Adam (Kingma and Ba, 2014).

Preliminary experiments in Ru→En found no meaningful difference in general-domain or in-domain performance when computing the diagonal of $\bar{F}$ on varying amounts of data ranging from 500k sentences to the full dataset. We also tried computing the diagonal of $\bar{F}$ on held-out data, as there is some evidence that estimating Fisher on held out data reduces overfitting in natural gradient descent (Pascanu and Bengio, 2013). However, we again found no meaningful differences. All results presented herein estimate the the diagonal of $\bar{F}$ on 500k training data sentences, which took less than an hour on a GTX 1080 Ti GPU.

We use a two-layer LSTM network with hidden unit size 512. The general-domain models are trained with a learning rate of 3E-4. We use dropout (0.1) on both RNN inputs and states. We compute lower-cased `multi-bleu.perl`. We use label smoothing (0.1) for all experiments except with MCL, because MCL explicitly regularizes the output distribution.

MCL uses an interpolation of the cross entropy between the output distribution of the model being trained and the general-domain models output distribution (scaled by α) and the standard training loss (scaled by $1-\alpha$). For MCL, we do a grid search over learning rates (10^{-4}, 10^{-5}, 10^{-6}) and α values of (0.1, 0.3, 0.5, 0.7, 0.9). For EWC, we do a grid search over the same learning rates and weight decay values of (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}).

---

[5] `github.com/thompsonb/sockeye_ewc`

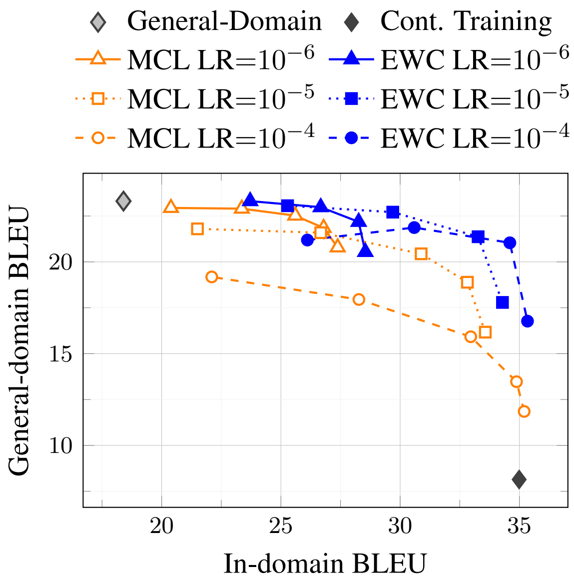


Figure 2: En→Ru results for various learning rates, for both MCL and EWC. Regularization amount increases from left to right for each trace. General-domain and continued training points shown for reference.

5 Results

We present the full in- and general-domain performance trade-off[6] for both EWC and MCL in Figure 1. This is computed by taking the convex hull of a grid search over learning rate and regularization amount for each method. EWC outperforms MCL at all operating points with the exception of Ru→En, where MCL provides a small in-domain performance improvement at lower general-domain performance; this was also observed in Khayrallah et al. (2018).

Figure 2 shows an example result (for En→Ru) of the grid search prior to taking the convex hull. We see similar trends between the three pairs of MCL/EWC curves at corresponding learning rates, but in each case EWC is further up/right, indicating better performance. Note that for both EWC and MCL, both learning rate and regularization amount have a large impact on final in- and general-domain performance.

General-domain gains for *no* in-domain performance degradation are presented in Table 2. Our method provides large general-domain gains (between 8.0 and 18.1 BLEU), regaining the majority of general-domain performance lost in continued training and substantially outperforming MCL.

Langs	GD	CT	MCL	EWC
En→De	24.2	5.7	9.0 (+3.2)	16.2 **(+10.5)**
De→En	29.6	10.0	23.3 (+13.3)	26.6 **(+16.5)**
En→Ru	23.3	8.1	11.8 (+3.7)	16.8 **(+8.6)**
Ru→En	28.6	10.4	21.2 (+10.8)	21.5 **(+11.1)**
En→Zh	39.5	6.1	6.1 (+0.0)	24.1 **(+17.9)**
Zh→En	43.5	9.9	9.9 (+0.0)	28.7 **(+18.8)**

Table 2: General-domain BLEU for: general-domain model prior to adaptation (GD), standard continued training (CT), and best performing MCL and EWC models with *no* in-domain degradation compared to CT (delta from CT). Best improvement over CT bolded.

6 Conclusion

We interpret the general-domain performance drop experienced during continued training as catastrophic forgetting of general-domain knowledge and demonstrate that it can be largely mitigated by applying Elastic Weight Consolidation.

We present the full trade-off for in- and general-domain performance and show that our method outperforms MCL (Dakwale and Monz, 2017) at all operating points in five of six language pairs. Our method is able to regain the majority of the general-domain performance lost during continued training without compromising in-domain performance and without an additional memory or computational burden at translation-time.

Our method retains the advantages of continued training while addressing one of its main shortcomings and can be used in practical situations to avoid poor performance when general-domain input is encountered, even when in-domain performance and translation efficiency are both critical.

Acknowledgments

The authors thank Paul McNamee, Matt Post, Zach Wood-Doughty, and the Johns Hopkins 2018 SCALE participants for for helpful discussions and technical assistance.

Brian Thompson is supported by the Department of Defense through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program. Jeremy Gwinnup received support from the Air Force Office of Scientific Research (AFOSR) Visiting Scientist Program. This work has been partially supported by the DARPA LORELEI and the IARPA MATERIAL programs.

---

[6] Previous work has compared single runs of competing methods, making comparison difficult (e.g. one system may be better on in-domain, the other better on general-domain).

References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Patrick Cadwell, Sharon OBrien, and Carlos SC Teixeira. 2018. Resistance and accommodation: factors for the (non-) adoption of machine translation among professional translators. *Perspectives*, 26(3):301–321.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391. Association for Computational Linguistics.

Praveen Dakwale and Christof Monz. 2017. Finetuning for neural machine translation with limited degradation across in-and out-of-domain data. *Proceedings of the XVI Machine Translation Summit*, page 117.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast Domain Adaptation for Neural Machine Translation. *CoRR*, abs/1612.06897.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*.

Marcin Junczys-Dowmunt, Bruno Pouliquen, and Christophe Mazenc. 2016. COPPA V2.0: Corpus of Parallel Patent Applications building large parallel corpora with GNU make. In *4th Workshop on Challenges in the Management of Large Corpora Workshop Programme*.

Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 36–44. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the 1st Workshop on Neural Machine Translation (and Generation) at ACL*. Association for Computational Linguistics.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

David J. C. MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

James Martens. 2014. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*.

Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government. Cleared for public release on 14 Jan 2019. Originator reference number RH-19-119318. Case number 88ABW-2019-0136.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, Copenhagen, Denmark. Association for Computational Linguistics.

Razvan Pascanu and Yoshua Bengio. 2013. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*.

Victoria Porro Rodriguez, Lucia Morado Vazquez, and Pierrette Bouillon. 2017. Study on the use of machine translation and post-editing in swiss-based language service providers. *Parallèles*, (29 (2)).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya D. McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 124–132, Belgium, Brussels. Association for Computational Linguistics.

David Vilar. 2018. Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 500–505.

Short-Term Meaning Shift: A Distributional Exploration

Marco Del Tredici* **Raquel Fernández*** **Gemma Boleda**†
*University of Amsterdam †Universitat Pompeu Fabra
{m.deltredici|raquel.fernandez}@uva.nl
gemma.boleda@upf.edu

Abstract

We present the first exploration of meaning shift over short periods of time in online communities using distributional representations. We create a small annotated dataset and use it to assess the performance of a standard model for meaning shift detection on short-term meaning shift. We find that the model has problems distinguishing meaning shift from referential phenomena, and propose a measure of contextual variability to remedy this.

1 Introduction

Semantic change has received increasing attention in empirical Computational Linguistics / NLP in the last few years (Tang, 2018; Kutuzov et al., 2018). Almost all studies so far have focused on meaning shift in long periods of time—decades to centuries. However, the genesis of meaning shift and the mechanisms that produce it operate at much shorter time spans, ranging from the online agreement on words' meaning in dyadic interactions (Brennan and Clark, 1996) to the rapid spread of new meanings in relatively small communities of people in (Wenger, 1998; Eckert and McConnell-Ginet, 1992). This paper is, to the best of our knowledge, the first exploration of the latter phenomenon—which we call **short-term meaning shift**—using distributional representations.

More concretely, we focus on meaning shift arising within a period of 8 years, and explore it on data from an online community of speakers, because there the adoption of new meanings happens at a fast pace (Clark, 1996; Hasan, 2009). Indeed, short-term shift is usually hard to observe in standard language, such as the language of books or news, which has been the focus of long-term shift studies (e.g., Hamilton et al., 2016; Kulkarni et al., 2015), since it takes a long time for a new meaning to be widely accepted in the standard language.

Our contribution is twofold. First, we create a small dataset of short-term shift for analysis and evaluation, and qualitatively analyze the types of meaning shift we find.[1] This is necessary because, unlike studies of long-term shift, we cannot rely on material previously gathered by linguists or lexicographers. Second, we test the behavior of a standard distributional model of semantic change when applied to short-term shift. Our results show that this model successfully detects most shifts in our data, but it overgeneralizes. Specifically, the model gets confused with contextual changes due to speakers in the community often talking about particular people and events, which are frequent on short time spans. We propose to use a measure of contextual variability to remedy this and showcase its potential to spot false positives of referential nature like these. We thus make progress in understanding the nature of semantic shift and towards improving computational models thereof.

2 Related Work

Distributional models of semantic change are based on the hypothesis that a change in context of use mirrors a change in meaning. This in turn stems from the Distributional Hypothesis, that states that similarity in meaning results in similarity in context of use (Harris, 1954). Therefore, all models (including ours) spot semantic shift as a change in the word representation in different time periods. Among the most widely used techniques are Latent Semantic Analysis (Sagi et al., 2011; Jatowt and Duh, 2014), Topic Modeling (Wijaya and Yeniterzi, 2011), classic distributional representations based on co-occurence matrices of target words and context terms (Gulordava and Baroni, 2011). More recently, researchers have used

---

[1] Data and code are available at: `https://github.com/marcodel13/Short-term-meaning-shift`.

Proceedings of NAACL-HLT 2019, pages 2069–2075
Minneapolis, Minnesota, June 2 - June 7, 2019.

word embeddings computed using the skip-gram model by Mikolov et al. (2013). Since embeddings computed in different semantic spaces are not directly comparable, time related representations are usually made comparable either by aligning different semantic spaces through a transformation matrix (Kulkarni et al., 2015; Azarbonyad et al., 2017; Hamilton et al., 2016) or by initializing the embeddings at t_{i+1} using those computed at t_i (Kim et al., 2014; Del Tredici et al., 2016; Phillips et al., 2017; Szymanski, 2017). We adopt the latter methodology (see Section 3.2).

Unlike most previous work, we focus on the language of online communities. Recent studies of this type of language have investigated the spread of new forms and meanings (Del Tredici and Fernández, 2017, 2018; Stewart and Eisenstein, 2018), competing lexical variants (Rotabi et al., 2017), and the relation between conventions in a community and the social standing of its members (Danescu-Niculescu-Mizil et al., 2013). None of these works has analyzed the ability of a distributional model to capture these phenomena, which is what we do in this paper for short-term meaning shift. Kulkarni et al. (2015) consider meaning shift in short time periods on Twitter data, but without providing an analysis of the observed shift nor systematically assessing the performance of the model, as we do here.

Evaluation of semantic shift is difficult, due to the lack of annotated datasets (Fremann and Lapata, 2016). For this reason, even for long-term shift, evaluation is usually performed by manually inspecting the n words whose representation changes the most according to the model under investigation (Hamilton et al., 2016; Kim et al., 2014). Our dataset allows for a more systematic evaluation and analysis, and enables comparison in future studies.

3 Experimental Setup

3.1 Data

We exploit user-generated language from an online forum of football fans, namely, the r/LiverpoolFC subreddit, one of the many communities hosted by the Reddit platform.[2] Del Tredici and Fernández (2018) showed that this subreddit presents many characteristics that favour the creation and spread of linguistic innovations, such as a topic that reflects a strong external interest and high density of the connections among its users. This makes it a good candidate for our investigation. We focus on a short period of eight years, between 2011 and 2017. In order to enable a clearer observation of short-term meaning shift, we define two non-consecutive time bins: the first one (t_1) contains data from 2011–2013 and the second one (t_2) from 2017.[3] We also use a large sample of community-independent language for the initialization of the word vectors, namely, a random crawl from Reddit in 2013. Table 1 shows the size of each sample.

3.2 Model

We adopt the model proposed by Kim et al. (2014), a representative method for computing diachronic meaning shift.[4] While other methods might be equally suitable (see Section 2), we expect our results not to be method-specific, because they concern general properties of short-term shift, as we show in Sections 4 and 5. In the model by Kim et al. (2014), word embeddings for the first time bin t_1 are initialized randomly; then, given a sequence of time-related samples, embeddings for t_i are initialized using the embeddings of t_{i-1} and further updated. If at t_i the word is used in the same contexts as in t_{i-1}, its embedding will only be marginally updated, whereas a major change in the context of use will lead to a stronger update of the embedding. The model makes embeddings across time bins directly comparable.

We implement the following steps: First, we create randomly initialized word embeddings with the large sample Reddit$_{13}$ to obtain meaning representations that are community-independent. We then use these embeddings to initialize those in LiverpoolFC$_{13}$, update the vectors on this sample, and thus obtain embeddings for time t_1. This step adapts the general embeddings to the LiverpoolFC community. Finally, we initialize the word embeddings for LiverpoolFC$_{17}$ with those of t_1, train on this sample, and get embeddings for t_2.

The vocabulary is defined as the intersection of the vocabularies of the three samples

---

[2] `https://www.reddit.com`. We downloaded Reddit data using the Python package Praw: `https://pypi.python.org/pypi/praw/`.

[3] These choices ensure that the samples in these two time bins are approximately of the same size – see Table 1. The r/LiverpoolFC subreddit exists since 2009, but very little content was produced in 2009–2010.

[4] The model was implemented using the Python package Gensim: `https://pypi.python.org/pypi/gensim/`.

sample	time bin	million tokens
Reddit_{13}	2013	~900
LiverpoolFC_{13}	2011–13	8.5
LiverpoolFC_{17}	2017	11.9

Table 1: Time bin and size of the datasets.

(Reddit_{13}, LiverpoolFC_{13}, LiverpoolFC_{17}), and includes 157k words. For Reddit_{13}, we include only words that occur at least 20 times in the sample, so as to ensure meaningful representations for each word, while for the other two samples we do not use any frequency threshold: Since the embeddings used for the initialization of LiverpoolFC_{13} encode community-independent meanings, if a word doesn't occur in LiverpoolFC_{13} its representation will simply be as in Reddit_{13}, which reflects the idea that if a word is not used in a community, then its meaning is not altered within that community. We train with standard skip-gram parameters (Levy et al., 2015): window 5, learning rate 0.01, embedding dimension 200, hierarchical softmax.

3.3 Evaluation dataset

Our dataset consists of 97 words from the r/LiverpoolFC subreddit with annotations by members of the subreddit —that is, community members with domain knowledge (needed for this task) but no linguistic background.

To ensure that we would get enough cases of semantic shift to enable a meaningful analysis, we started out from content words that increase their relative frequency between t_1 and t_2.[5] A threshold of 2 standard deviations above the mean yielded $\sim$200 words. The first author manually identified 34 semantic shift candidates among these words by analyzing their contexts of use in the r/LiverpoolFC data. Semantic shift is defined here as a change in the ontological type that a word denotes, which takes place when the word starts to be used to denote an entity which is different from the one originally denoted and the new use spreads among the members of a community (see examples in Sec. 4). We added two types of confounders: 33 words with a significant frequency increase but not marked as meaning shift candidates, and 33 words with constant frequency between t_1 and t_2, included as a sanity check. All words have absolute frequency in range [50–500].

The participants were shown the 100 words (in randomized order) together with randomly chosen contexts of usage from each time period (μ=4.7 contexts per word) and, for simplicity, were asked to make a binary decision about whether there was a change in meaning. In order to have the redditors familiarize with the concept of meaning change, we first provide them with an intuitive, non-technical definition, and then a set of cases that exemplify it. The instructions to participants can be found in the project's GitHub repository (see footnote 1).

Semantic shift is arguably a graded notion. In line with a suggestion by Kutuzov et al. (2018) to account for this fact, we aggregate the annotations into a graded *semantic shift index*, ranging from 0 (no shift) to 1 (shift) depending on how many subjects spotted semantic change. The shift index is exclusively based on the judgments of the redditors, and does not consider the preliminary candidate selection done by us. Overall, 26 members of r/LiverpoolFC participated in the survey, and each word received on average 8.8 judgements. Further details about the dataset are in Appendix A.

4 Types of Meaning Shift

We identify three main types of shift in our data via qualitative analysis of examples with a shift index > 0.5: metonymy, metaphor, and meme.

Metonymy. In metonymic shifts, a highly salient characteristic of an entity is used to refer to it. Among these cases are, for example, *'highlighter'*, which in t_2 occurs in sentences like *'we are playing with the highlighter today'*, or *'what's up with the hate for this kit? This is great, ten times better than the highlighter'*, used to talk about a kit in a colour similar to that of a highlighter pen; or *'lean'*, in *'I hope a lean comes soon!'*, *'Somebody with speed…make a signing…Cuz I need a lean'*, used to talk about hiring players due to new hires typically leaning on a Liverpool symbol when posing for a photo right after signing for the club. Particularly illustrative is the 'F5' example shown in Table 2. While 'F5' is initially used with its common usage of shortcut for refreshing a page (1), it then starts to denote the act of refreshing in order to get the latest news about the possible transfer of a new

---

[5] Frequency increase has been shown to positively correlate with meaning change (Wijaya and Yeniterzi, 2011; Kulkarni et al., 2015); although it is not a necessary condition, it is a reasonable starting point, as a random selection of words would contain very few positive examples. Our dataset is thus biased towards precision over recall.

(1)	*Damn, after losing the F5 key on my keyboard [...]*	16 Jun
(2)	*[he is] so close, F5 tapping is so intense right now*	18 Jun
(3)	*Don't think about it too much, man. Just F5*	1 Jul
(4)	*Literally 4am I slept and just woke up and thought it was f5 time*	3 Jul
(5)	*this was a happy f5*	3 Jul
(6)	*what is an F5?*	3 Jul
(7)	*I'm leaving the f5 squad for now*	5 Jul
(8)	*I made this during the f5 madness*	6 Sep

Table 2: Examples of use of 'F5' with time stamps, which illustrate the speed of the meaning shift process. All the examples are from LiverpoolFC$_{17}$.

player to LiverpoolFC (2). This use catches on and many redditors use it to express their tension while waiting for good news (3-5),[6] though not all members are aware of the new meaning of the word (6). When the transfer is almost done, someone leaves the *'F5 squad'* (7), and after a while, another member recalls the period in which the word was used (8).

Metaphor. Metaphorical shifts lead to a broadening of the original meaning of a word through analogy. For example, in t_2 *'shovel'* occurs in sentences such as *'welcome aboard, here is your shovel'* or *'you boys know how to shovel coal'*: the team is seen as a train that is running through the season, and every supporter is asked to figuratively contribute by shoving coal into the train boiler.

Meme. Finally, memes are another prominent source of meaning shift. In this case, fans use a word to make jokes and be sarcastic, and the new usage quickly spreads within the community. As an example, while Liverpool was about to sign a new player named Van Dijk, redditors started to play with the homography of the first part of the surname with the common noun 'van', its plural form 'vans', and the shoes brand 'Vans': '*Rumour has it Van Djik was wearing these vans in the van*' or '*How many vans could Van Dijk vear if Van Dijk could vear vans*'. Jokes of this kind are positively received in the community ('*Hahah I love it. Anything with vans is instant karma!*') and quickly become frequent in it.

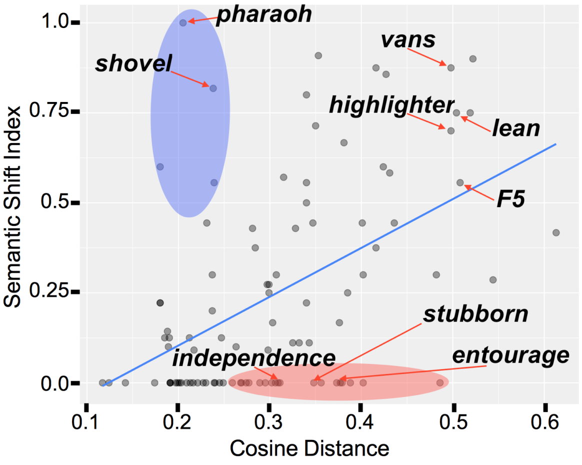


Figure 1: Semantic shift index vs. cosine distance in the evaluation dataset (Pearson's $r = 0.49$, $p < 0.001$). Red horizontal ellipsis: false positives; blue vertical ellipsis: false negatives.

5 Modeling Results and Analysis

The positive correlation between cosine distance and semantic shift index (Pearson's r= 0.49, $p < 0.001$, see Figure 1) confirms the hypothesis that meaning shift is mirrored by a change in context of use. However, we also find systematic deviations.

5.1 False negatives

A small, but consistent group is that of words that undergo semantic shift but are not captured by the model (blue vertical ellipsis Figure 1; shift index>0.5, cosine distance<0.25). These are all metaphorical shifts; in particular, cases of extended metaphor (Werth, 1994), where the metaphor is developed throughout the whole text. For instance, besides the *'shovel'* example mentioned in Section 4, we find *'pharaoh'*, the nickname of an Egyptian player who joined Liverpool in 2017, used in sentences like *'approved by our new Pharaoh Tutankhamun'*, or *'our dear Egyptian Pharaoh, let's hope he becomes a God'*. Despite the metaphoric usage, the local context of these words is similar to the literal one, and so the model does not spot the meaning shift. We expect this to happen in long-term shift models, too, but we are not aware of results confirming this.

5.2 False positives

A larger group of problematic cases is that of words that do *not* undergo semantic shift despite showing relatively large differences in context between t_1 and t_2 (red horizontal ellipsis in Figure 1; shift index=0, cosine distance>0.25). Manual in-

---

[6] Here the semantic change is accompanied by a change in the part of speech, and 'F5' becomes a denominal verb.

spection reveals that most of these "errors" are due to a referential effect: words are used almost exclusively to refer to a specific person or event in t_2, and so the context of use is different from the contexts in t_1. For instance, *'stubborn'* is almost always used to talk about a coach who was not there in 2013 but only in 2017; *'entourage'*, for the entourage of a particular star of the team; *'independence'* for the political events in Catalonia (Spain). In all these cases, the meaning of the word stays the same, despite the change in context. In line with the Distributional Hypothesis, the model spots the context change, but it is not sensitive to its nature. We expect long-term shift to not be as susceptible to referential effects like these because embeddings are aggregated over a larger and more varied number of occurrences.

We expect that in referential cases the contexts of use will be *narrower* than for words with actual semantic shift, as they are specific to one person or event. Hence, a measure of *contextual variability* should help spot false positives. To test this hypothesis, we define contextual variability as follows: For a target word, we create a vector for each of its contexts (5 words on both sides of the target) in t_2 by averaging the embeddings of the words occurring in it, and define variability as the average pairwise cosine distance between context vectors.[7] We find that contextual variability is indeed significantly correlated with semantic shift in our dataset (Pearson's $r = 0.55$, $p < 0.001$), while it is independent from cosine distance (Pearson's $r = 0.18$, $p > 0.05$). These two aspects are thus complementary. While both shift words and referential cases change context of use in t_2, context variability captures the fact that only in referential cases words occur in a restricted set of contexts. Figure 2 shows this effect visually. This result can inform future models of short-term meaning shift.

6 Conclusion

The goal of this initial study was to bring to the attention of the NLP community **short-term meaning shift**, an under-studied problem in the field. Our hope is that it will spark further research into a phenomenon which, besides being of theoretical interest, has potential practical implications for NLP downstream tasks concerned with user-generated language, as modeling how word meanings rapidly change in communities would allow a better understanding of what their members say. Future research should experiment with other datasets (reddits from other domains, other online communities) and also alternative models that address the challenges described here.

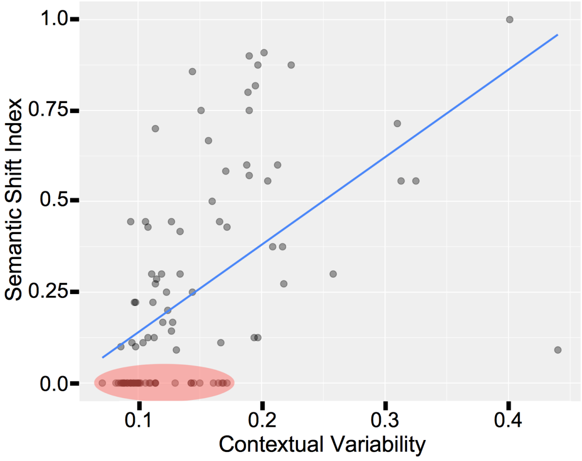


Figure 2: Semantic shift index vs. context variability. Red horizontal ellipsis: referential cases which are assigned high cosine distance values by the model (false positives).

Acknowledgements

The research carried out by the Amsterdam section of the team was partially funded by the Netherlands Organisation for Scientific Research (NWO) under VIDI grant no. 276-89-008, *Asymmetry in Conversation*. This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 715154), and from the Spanish Ramón y Cajal programme (grant RYC-2015-18907). This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Hosein Azarbonyad, Mostafa Dehghani, Kaspar Beelen, Alexandra Arkut, Maarten Marx, and Jaap Kamps. 2017. Words are malleable: Computing

---

[7] There are alternative ways of measuring contextual variability, but we expect them to yield the same picture. For instance, we experimented with a different window size and obtained the same pattern.

semantic shifts in political and media discourse. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1509–1518. ACM.

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(6):1482.

Herbert H. Clark. 1996. *Using language*. Cambridge University Press.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web*, pages 307–318. ACM.

Marco Del Tredici and Raquel Fernández. 2017. Semantic variation in online communities of practice. In *IWCS 2017-12th International Conference on Computational Semantics-Long papers*.

Marco Del Tredici and Raquel Fernández. 2018. The road to success: Assessing the fate of linguistic innovations in online communities. In *COLING 2018-27th International Conference on Computational Linguistics-Long papers*.

Marco Del Tredici, Malvina Nissim, and Andrea Zaninello. 2016. Tracing metaphors in time through self-distance in vector spaces. In *CLiC it 2016 - Third Italian Conference on Computational Linguistics*.

Penelope Eckert and Sally McConnell-Ginet. 1992. Communities of practice: Where language, gender, and power all live. In *Locating Power, Proceedings of the 1992 Berkeley Women and Language Conference*, pages 89–99.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the google books ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1489–1501.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Ruqaiya Hasan. 2009. *Semantic variation: Meaning in society and in sociolinguistics*. Equinox London.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 229–238. IEEE Press.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65. Association for Computational Linguistics.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. ACM.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1384–1397. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lawrence Phillips, Kyle Shaffer, Dustin Arendt, Nathan Hodas, and Svitlana Volkova. 2017. Intrinsic and extrinsic evaluation of spatiotemporal text representations in twitter streams. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 201–210.

Rahmtin Rotabi, Cristian Danescu-Niculescu-Mizil, and Jon Kleinberg. 2017. Competition and selection among conventions. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1361–1370. International World Wide Web Conferences Steering Committee.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. *Current methods in historical semantics*, pages 161–183.

Ian Stewart and Jacob Eisenstein. 2018. Making" fetch" happen: The influence of social and linguistic context on the success of lexical innovations. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 448–453.

Xuri Tang. 2018. A state-of-the-art of semantic change computation. *Natural Language Engineering*, 24(5):649–676.

Etienne Wenger. 1998. *Communities of practice: Learning, meaning, and identity*. Cambridge University Press.

Paul Werth. 1994. Extended metaphor—a text-world account. *Language and literature*, 3(2):79–103.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40. ACM.

A Further Details on Evaluation Dataset

For our experiment, we considered content words only, which we identified by using the external list of common words available at `https://www.wordfrequency.info/free.asp`.

Three words were discarded from the initial list after analysis of the redditor data: 'discord' and 'owls' due to the homonymy with proper names not detected during survey's implementation; 'tracking' because the chosen examples clearly mislead the judgements of the redditors.

As detailed in Section 3.3, 26 members of r/LiverpoolFC participated in the survey, and each word received on average 8.8 judgements. We computed inter-annotator agreement as Krippendorff's alpha, and obtained $\alpha = 0.58$, a relatively low value but common in semantic tasks (Artstein and Poesio, 2008).

The results of the annotation validate our initial word sampling procedure:

- the words that present a significant increase in frequency and were annotated as meaning shift by us received an average shift annotation of 0.72 (± 0.15);
- the words that present a significant increase in frequency but that were *not* annotated as meaning shift by us received an average shift annotation of 0.15 (± 0.16);
- the words that keep a constant frequency between t_1 and t_2, and we don't consider examples of meaning shift, got 0.07 (± 0.12).

Detecting Derogatory Compounds – An Unsupervised Approach

Michael Wiegand*0, **Maximilian Wolf***, **Josef Ruppenhofer**†‡
*Spoken Language Systems, Saarland University, Saarbrücken, Germany
†Leibniz ScienceCampus, Heidelberg/Mannheim, Germany
‡Institute for German Language, Mannheim, Germany
`michael.wiegand@lsv.uni-saarland.de`
`maximilian.wolf@lsv.uni-saarland.de`
`ruppenhofer@ids-mannheim.de`

Abstract

We examine the new task of detecting derogatory compounds (e.g. *curry muncher*). Derogatory compounds are much more difficult to detect than derogatory unigrams (e.g. *idiot*) since they are more sparsely represented in lexical resources previously found effective for this task (e.g. Wiktionary). We propose an unsupervised classification approach that incorporates linguistic properties of compounds. It mostly depends on a simple distributional representation. We compare our approach against previously established methods proposed for extracting derogatory unigrams.

1 Introduction

Abusive or offensive language is commonly defined as hurtful, derogatory or obscene utterances made by one person to another person.[1] Examples are (1)-(3). In the literature, closely related terms include *hate speech* (Waseem and Hovy, 2016) or *cyber bullying* (Zhong et al., 2016). While there may be nuanced differences in meaning, they are all compatible with the general definition above.

(1) stop editing this, you **dumbass**.
(2) Just want to slap the **stupid** out of these **bimbos**!!!
(3) Go lick a pig you arab muslim piece of **scum**.

Due to the rise of user-generated web content, in particular on social media networks, the amount of abusive language is also steadily growing. NLP methods are required to focus human review efforts towards the most relevant microposts.

A substantial amount of abusive utterances comprises derogatory words (e.g. *bimbo* or *scum*). Automatic extraction methods of such words are required since new derogatory words constantly enter language. Wiegand et al. (2018a) extracted a large list of such expressions and demonstrated its importance for text classification.

In this work, we focus on a subtype of derogatory terms, namely derogatory compounds (e.g. *booze hound*, *curry muncher*, *fault finder*). Distinguishing such multi-word expressions from non-derogatory ones (e.g. *fox hound*, *mile muncher*, *branch finder*) is more difficult than classifying unigrams since they are only sparsely represented in general-purpose lexical resources which have previously been found an effective source from which to learn abusive language, such as Wiktionary.[2] For example, while 97% of the derogatory unigrams of the gold standard lexicon in Wiegand et al. (2018a) are contained in Wiktionary, less than 17% of the derogatory compounds used as our gold standard in this work can be found.

Despite their sparsity in lexical resources derogatory compounds are a frequent phenomenon, particularly in German data, which is why we study this task on that language. On the German benchmark corpus for abusive language detection, the *GermEval corpus* (Wiegand et al., 2018b), we found that of the abusive microposts in the test set that include at least one derogatory expression, 39% contain a derogatory compound.

In our work, we focus on noun-noun compounds. Each compound (e.g. *curry muncher*) comprises two constituents, a modifier (i.e. *curry*) and a head (i.e. *muncher*). On the GermEval corpus, 77% of the derogatory compounds are noun-noun compounds. We only consider compounds whose constituents are <u>not</u> derogatory. 58% of the derogatory compounds on the GermEval corpus fall under this category. Given publicly available lists of derogatory unigrams, the detection of derogatory compounds containing derogatory constituents (e.g. *mother<u>fucker</u>*) is rather trivial.

---

[0] Present affiliation: Leibniz ScienceCampus, Heidelberg/Mannheim, Germany

[1] `http://thelawdictionary.org/`

[2] `https://en.wiktionary.org/`

Proceedings of NAACL-HLT 2019, pages 2076–2081
Minneapolis, Minnesota, June 2 - June 7, 2019.

There even exist abusive word generators employing such compounds.[3]

We present the first study to detect derogatory noun-noun compounds and propose an unsupervised classification approach based on distributional information that does not require any properly labeled training data. We demonstrate that linguistic features that have previously been found effective for the classification of derogatory unigrams are notably less effective for the detection of derogatory compounds. We created a new dataset of derogatory compounds which will be made **publicly available**.[4]

Our task is framed as a **binary classification problem**. Each given compound is to be classified out of context as either derogatory or not. For the sake of accessibility, we use English translations of our German compounds in this paper.

2 Related Work

Lexical knowledge for the detection of abusive language has only received little attention in previous work (Schmidt and Wiegand, 2017), the notable exceptions are Razavi et al. (2010) who present a manually-compiled lexicon, Gitari et al. (2015) who bootstrap *hate verbs* and Wiegand et al. (2018a) who induce a lexicon of derogatory words. In all these researches, however, derogatory compounds are not explicitly addressed.

3 Data

We built a **gold standard of derogatory compounds** to train and test classifiers. We inspected a range of websites containing derogatory word lists.[5] Since ambiguity is a massive problem in these lists[6] which makes them hardly usable for abusive language detection (Wiegand et al., 2018a), we manually extracted noun-noun compounds which we considered **unambiguously** derogatory. In order to produce non-derogatory compounds, we randomly sampled from the COW16 corpus (Schäfer, 2015) for each derogatory compound (e.g. *booze hound*) other compounds sharing the same head (e.g. *fox hound*, *stag hound*). Since among those putative non-derogatory instances, there could well be further derogatory compounds, we manually annotated them as well. We limited the set of compounds sharing the same head, which we henceforth call **head group**, to 20 compounds. Thus, we hope to avoid any biases towards particular heads.

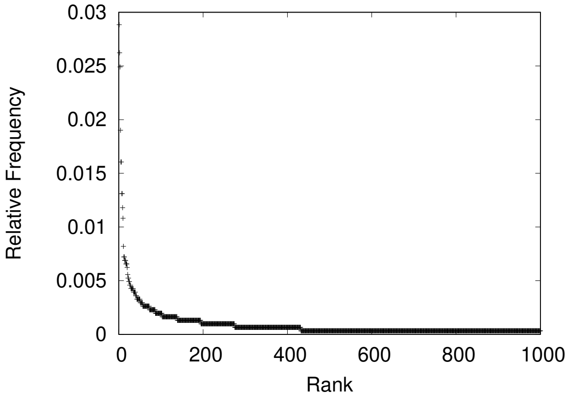


Figure 1: Head distribution of derogatory compounds.

Property	Freq
total compounds	3500
derogatory compounds	382
head groups (*each group contains 20 compounds*)	175
average no. of derogatory compounds in head group	2.2

Table 1: Some statistics of the gold standard.

We also looked at the natural distribution of heads on derogatory compounds. As a proxy we considered the union of all derogatory compounds found on the above websites.[5] Figure 1 plots the frequency rank of the heads against the relative frequency of a particular head. The plot suggests that the heads follow a power-law distribution (Zipf, 1965). As a consequence, one cannot assume that this task could be solved by looking up heads in a finite lexicon with words that often form derogatory compounds in combination with different modifiers.

On a sample of 600 compounds, we measured a substantial agreement of Cohen's $\kappa = 0.61$ (Landis and Koch, 1977) between 2 annotators. Our final dataset (Table 1) comprises 3,500 compounds with only 11% being derogatory.

We also created a **gold standard of derogatory unigram words** in order to examine in how far derogatory compounds can be detected by a classifier trained on derogatory unigrams. For this lexicon, we manually translated the *base lexicon* from Wiegand et al. (2018a) to German.

---

[3] `http://sweary.com`

[4] `https://github.com/uds-lsv/offensive-compounds`

[5] `www.hyperhero.com/de/insults.htm`
`www.schimpfwoerter.de`
`www.seechat.de/warmduscher.htm`

[6] These lists contain many compounds commonly used in a non-offensive manner, e.g. *Colatrinker (coke drinker)*.

4 Method

Our method does not require any labeled training data. In the first step (§4.1), we apply high-precision diagnostics for the detection of derogatory compounds. The output are rankings in which derogatory compounds should be ranked highest. In the second step (§4.2), we combine and re-rank the output of those diagnostics. Our method largely relies on a distributional representation of our compounds. We induced embeddings of our compounds using Word2Vec (Mikolov et al., 2013) on the COW16 corpus, which with its 30B tokens is one of the largest German corpora. Since we exclusively work on German data and German compounds occur as closed compounds, e.g. *Milchbube (milk sop)* or *Schnapsdrossel (booze hound)*, we can employ standard tokenization[7] for inducing embeddings for our compounds.

4.1 Individual High-Precision Diagnostics

Negative Polarity (NEG). Derogatory words form a subset of negative polar expressions. Due to their sparsity, however, derogatory compounds are rarely part of any sentiment lexicon (containing polar expressions). We, therefore, rank all our compounds according to their cosine similarity to a centroid embedding-vector computed from all negative polar expressions from the German PolArt sentiment lexicon (Klenner et al., 2009).

Compound Occurrence vs. Constituent Occurrence (COMCON). Derogatory compounds can be creative word constructions (e.g. *booze hound*, *oxygen thief*, *keyboard warrior*). Consequently, their constituents are often not semantically related. For instance, in *booze hound*, *booze* bears no common semantic relation to *hound*. Therefore, the corpus frequency of a derogatory compound should be much higher than its constituents co-occurring in a sentence (i.e. with other words occurring in between). Such co-occurrences should be coincidental.

We capture this by the following formula (frequencies are computed on the COW16 corpus):

$$COMCON = \frac{\text{\# compound mentions in corpus}}{\text{\# constituents co-occurring in sentence}} \tag{1}$$

In prose, COMCON ranks all compounds by the ratio of observed compound mentions and constituent co-occurrences in a sentence. For derogatory compounds, there should be a high frequency of compound mentions but only a low frequency of the constituents co-occurring in a sentence. Therefore, COMCON will have a high score. While there is a similarly high frequency of compound mentions for non-derogatory compounds, there is also a high frequency of the constituents co-occurring in a sentence since these constituents are usually semantically related (e.g. *landowner* or *circus clown*). This should result in COMCON producing comparably lower scores.

Derogatory Compound Must Be Person (PERSON). We rank our compounds with regard to how likely they represent a person since many non-derogatory compounds represent either objects or animals (e.g. *booze hound* vs. *sight hound*, *fox hound*, *stag hound*). We first compute a centroid vector representing persons. Then, we rank compounds by their similarity to that vector.

As a proxy for persons, we took embeddings of words representing professions, e.g. *banker*, *lawyer*, *salesman*. We also experimented with personal pronouns as a proxy for persons. However, we found them unsuitable since they are also often used as referring expressions to other entities, such as animals. Professions, on the other hand, can only refer to humans. The list of professions we used was created ad-hoc. It should be reproducible in any arbitrary language. *The full list is included in the supplementary notes.*[4]

Outlier Compound(s) in Head Group (OUT). In most head groups, derogatory compounds represent a clear minority with only 1 or 2 compounds. The derogatory compounds are also often semantically different from the non-derogatory compounds (*keyboard warrior* vs. *rajput warrior*, *ninja warrior*, *samurai warrior*). This is particularly true if the non-derogatory compounds are very homogeneous. From that observation we derive a diagnostic in which we determine the semantic outlier(s) for each head group. First, we compute for each compound the average pairwise similarity to all other compounds within its head group. The resulting score of a compound (converted to a <u>dis</u>similarity score by taking its inverse) is then multiplied by a weight representing the homogeneity of all compounds within that head group.[8] (*Pseudocode is provided in the supplementary notes.*[4]) This is done since for head

---

[7] Any alphanumeric string separated by spaces is considered a token.

[8] The homogeneity weight is the average pairwise similarity of all compounds belonging to the same head group.

groups that are heterogeneous (e.g. ***legacy hunter***, *job hunter*, *autograph hunter*), there are less obvious outliers.

4.2 Combination and Reranking

Combination (COMB). Negative polarity is a pre-requisite for being derogatory (Sood et al., 2012; Dinakar et al., 2012; Gitari et al., 2015). Therefore, we base our combination on the ranking of NEG. From that ranking we remove all those compounds which have not co-occurred at the high ranks of at least one of the other diagnostics (COMCON, OUT, PERSON).[9] Compounds that are highly ranked by several diagnostics should more likely represent derogatory compounds.

Re-Ranking by PageRank (PRANK). We observed that among the top ranks of COMB, the derogatory compounds are semantically similar (e.g. *dwarf tosser*, *mischief maker*, *slimeball*) while the non-derogatory compounds are semantically different from each other (e.g. *biker club*, *spirit bear*). Therefore, we run personalized PageRank (Agirre and Soroa, 2009) to further improve the ranking by enforcing the compounds on the high ranks to be distributionally similar. We build a word-similarity graph where our compounds are nodes and edges encode cosine-similarities of their embeddings. PageRank then produces a ranking of nodes where the highest ranked nodes are the ones most highly connected. In personalized PageRank prior information is added. A biased graph is constructed in which attention is drawn towards particular regions of interest. This is achieved by assigning re-entrance weights to the individual nodes. As prior information, we set the nodes representing the compounds returned by COMB with a uniform re-entrance weight (α)[10] while all other nodes receive a weight of 0.

Label Propagation (LP). While previous diagnostics were designed to isolate a few derogatory compounds with a high precision, LP aims for increasing recall. We define some high-precision seeds for the two categories of our task and then propagate the labels to the unlabeled compounds by using label propagation (Talukdar et al., 2008). The algorithm operates on the same word-similarity graph that we used for PRANK. We define highly ranked compounds from PRANK as derogatory seeds and lowly ranked compounds as non-derogatory seeds. Unlike the previous diagnostics, the output of LP is a binary categorization rather than a ranking. In order to make this output comparable to the other diagnostics, we converted the output of LP to a ranking. This is achieved by ranking the compounds predicted as derogatory according to the confidence score provided by the classifier.

5 Experiments

Table 2 shows the precision at rank n (P@n) of different rankings as measured on our compound gold standard. For LP, we consider the top 50 compounds from PRANK as derogatory seeds and the bottom 500 as non-derogatory seeds.[11] As a baseline we add a randomized ranking (RAND).

PRANK produces a very high precision on the high ranks, outperforming the individual rankings and COMB. We also tested a modification, PRANK$_{NEG}$, which applies personalized PageRank on the output of NEG, which is the strongest individual ranking. Since PRANK outperforms PRANK$_{NEG}$, we conclude that the high precision of PRANK also depends on the combination of the individual rankings. LP manages to notably raise scores on the lower ranks (e.g. P@300) which proves the advantage of LP over PRANK.

Table 3 compares our proposed method (LP) against supervised classifiers. We evaluate the entire classification output (with F1-measure) rather than a ranking. The classifiers are trained on our unigram or compound gold standard (§3). For the latter case, we conducted 10-fold crossvalidation. 500 of the 3500 compounds were reserved as a development set on which we tuned hyperparameters of the supervised classifiers. (*The supplementary notes*[4] *contain more details.*) As features we consider word embeddings and the linguistic features from Wiegand et al. (2018a). They are based on knowledge that is expensive to produce, such as sentiment views, polar intensity, or information from Wiktionary.[12]

Table 3 shows that learning from the compound gold standard is more effective than learning from the existing unigram gold standard. Given the

---

[9] We took top 350 from all these rankings which resembles the number of derogatory compounds on our dataset.

[10] Following Manning et al. (2008), we set $\alpha = 0.1$.

[11] The ratio of derogatory and non-derogatory compounds should vaguely reflect the class distribution.

[12] The method WSUP from Wiegand et al. (2018a) was not considered because of its poor performance on compounds.

P@n	RAND	COMCON	PERSON	OUT	NEG	COMB	PRANK$_{NEG}$	PRANK	LP
P@25	12.0	19.2	50.0	60.0	72.0	80.0	96.0	**100.0**	**100.0**
P@50	14.0	20.0	46.0	44.0	62.0	74.0	88.0	**94.0**	**94.0**
P@100	10.0	26.0	40.0	38.0	58.0	68.0	68.0	**82.0**	77.0
P@200	10.0	30.5	31.5	26.5	48.5	54.5	42.0	59.0	**68.5**
P@300	12.3	27.3	27.7	21.7	39.3	44.0	29.3	44.3	**60.3**

Table 2: Comparison of different rankings, evaluated by precision at rank n (P@n). (*PRANK*$_{NEG}$ *is the ranking applied solely on the output of NEG; PRANK is the ranking applied on the output of COMB.*)

Training	Classifier	Features	F1
unigram	SVM	embeddings+linguistic	63.3
compound	SVM	linguistic	66.1
	SVM	embeddings (*off-the-shelf*)	68.5
	SVM	embeddings	72.4
	SVM	embeddings+linguistic	74.7
none	LP	(embeddings)	73.5

Table 3: Comparison of different classifiers.

Classifier	SVM (embeddings+linguistic)				LSTM
Unit	head	modifier	compound	combined	characters
F1	57.0	60.2	**74.7**	69.0	54.5

Table 4: Comparison of compositional approaches.

strong performance of embeddings, we also examined the performance of (publicly available) off-the-shelf embeddings[13] and found that the high classification scores can be mainly ascribed to the large corpus on which we induced our embeddings (i.e. COW16).

Our unsupervised approach (LP) is almost on a par with the most complex SVM. This is particularly appealing since we produced that classifier without manually labeled training data and those manually-created resources required for the linguistic features.

Compound embeddings are the most predictive information for our task, but even from the large COW16 corpus, we only obtained embeddings for 60% of our compounds.[14] In Table 4, we evaluate compositional information, which can also be used for compounds that lack an embedding. We apply an SVM with the best previous feature set (of which embeddings are the main contributor) on the constituents of the compounds. Moreover, we train an LSTM on the sequence of characters of the compound. Table 4 shows that information drawn from units other than the compound itself is less effective. The feature combination of head, modifier and compound is not effective either.

Instead of applying embeddings on constituents and concatenating them, we also examine a sophisticated compositional model (*Wmask*) based on a masking process that takes into account the variation of a constituent depending on whether it is a head or a modifier (Dima, 2015). Table 5 shows the performance of the two best previous classifiers where compounds lacking an embedding are represented by an embedding approximated by Wmask (rather than a dummy vector). The table shows that the two classifiers can be improved by adding the approximated embeddings.

Classifier	SVM (embed.+ling.)		LP	
Embeddings	plain	+approx.	plain	+approx.
F1	74.7	75.7	73.5	74.9*

Table 5: Compound embedding augmentation (**: statistically better than the plain classifier using a paired t-test at* $p < 0.05$).

6 Conclusion

We examined the new task of detecting derogatory compounds and proposed an unsupervised approach incorporating linguistic properties of compounds that mostly depend on a distributional representation. Our method outperforms linguistic features previously shown to be effective for the detection of derogatory unigrams and it is on a par with a far more expensive state-of-the-art supervised approach. Features defined on the constituents of a compound and training a classifier on derogatory unigrams are far less effective.

Acknowledgements

The authors would like to thank Ines Rehbein for feedback on earlier drafts of this paper. We are also grateful to Corina Dima for helping us to run her semantic composition toolkit *wordcomp* on our data. The authors were partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

---

[13] We took the publicly available embeddings induced on Twitter-data from: `www.spinningbytes.com/resources/wordembeddings/`

[14] For the remaining compounds, we used dummy vectors.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–41, Athens, Greece.

Corina Dima. 2015. Reverse-engineering Language: A Study on the Semantic Compositionality of German Compounds. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1637–1642, Lisbon, Portugal.

Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. 2012. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Transactions on Interactive Intelligent Systems*, 2(3):18:1–18:30.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):2015–230.

Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. PolArt: A Robust Tool for Sentiment Analysis. In *Proceedings of the Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 235–238, Odense, Denmark.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations (ICLR)*, Scottsdale, AZ, USA.

Amir Hossein Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive Language Detection Using Multi-level Classification. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 16–27, Ottawa, Canada.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC)*, pages 28–34, Lancaster, United Kingdom.

Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the EACL-Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 1–10, Valencia, Spain.

Sara Owsley Sood, Elizabeth F. Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the Association for Information Science and Technology*, 63(2):270–285.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 582–590, Honolulu, HI, USA.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL – Student Research Workshop)*, pages 88–93, San Diego, CA, USA.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018a. Inducing a Lexicon of Abusive Words – A Feature-Based Approach. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 1046–1056, New Orleans, LA, USA.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018b. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of the GermEval Workshop*, pages 1–10, Vienna, Austria.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J. Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3952–3958, New York City, NY, USA.

George K. Zipf. 1965. *The Psycho-Biology of Language*. MIT Press.

Personalized Neural Embeddings for Collaborative Filtering with Text

Guangneng Hu
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
`njuhgn@gmail.com`

Abstract

Collaborative filtering (CF) is a core technique for recommender systems. Traditional CF approaches exploit user-item relations (e.g., clicks, likes, and views) only and hence they suffer from the data sparsity issue. Items are usually associated with unstructured text such as article abstracts and product reviews. We develop a Personalized Neural Embedding (PNE) framework to exploit both interactions and words seamlessly. We learn such embeddings of users, items, and words jointly, and predict user preferences on items based on these learned representations. PNE estimates the probability that a user will like an item by two terms—behavior factors and semantic factors. On two real-world datasets, PNE shows better performance than four state-of-the-art baselines in terms of three metrics. We also show that PNE learns meaningful word embeddings by visualization.

1 Introduction

Recommender systems are widely used in e-commerce platforms, such as to help consumers buy products at Amazon, watch videos on Youtube, and read articles on Google News. They are useful to alleviate the information overload and improve user satisfaction. Given history records of consumers such as the product transactions and movie watching, collaborative filtering (CF) is among the most effective approaches based on the simple intuition that if users rated items similarly in the past then they are likely to rate items similarly in the future (Sarwar et al., 2001).

History records include both implicit (e.g., purchase and clicks) and explicit (e.g., likes/dislikes and ratings) feedback which can be represented as a user-item interaction matrix. Typically, observed user-item interactions are incomplete with a large portion remaining not recorded. The goal of recommendation is to predict user preferences on these missing interactions. This setting requires to complete the partial observed rating matrix. Matrix Factorization (MF) techniques which can learn latent factors for users and items are the main cornerstone for CF (Mnih and Salakhutdinov, 2008; Koren, 2008; Koren et al., 2009). It is effective and flexible to integrate with additional data sources (Hu et al., 2015). Recently, neural networks like Multilayer Perceptron (MLP) are used to learn an interaction function from data with the power of learning highly nonlinear relationships between users and items (Dziugaite and Roy, 2015; Cheng et al., 2016; He et al., 2017; Hu et al., 2018b). MF and neural CF exploit user-item behavior interactions only and hence they both suffer from the data sparsity and cold-start issues.

Items are usually associated with unstructured text, like news articles and product reviews. These additional sources are essential for recommendation beyond user-item interactions since they contain independent and diverse information. Hence, they provide an opportunity to alleviate the data sparsity issue (Ganu et al., 2009; Hu et al., 2018a). For application domains like recommending research papers and news articles, the unstructured text associated with the item is its text content (Wang and Blei, 2011; Wang et al., 2015; Bansal et al., 2016). For some domains like recommending products, the unstructured text associated with the item is its user reviews which justify the rating behavior (McAuley and Leskovec, 2013; He and McAuley, 2016). These methods adopt topic modelling techniques and neural networks to exploit the item content leading to performance improvement.

A typical way of exploiting text content is to firstly extract a feature vector for each document by averaging word embeddings in the document, and then to learn a text factor corresponding to this feature vector (Hu and Dai, 2017). These embed-

Proceedings of NAACL-HLT 2019, pages 2082–2088
Minneapolis, Minnesota, June 2 - June 7, 2019.

dings are pre-trained from a large corpus such as Wikipedia. This approach separates the extraction of text feature from the learning of user-item interaction. These two processes cannot benefit from each other and errors in the previous step maybe propagate to the successive steps. Another way is to learn a topic vector using topic modelling (Wang and Blei, 2011; McAuley and Leskovec, 2013; Bao et al., 2014) by aligning behavior factors and topic factors with a link function such as softmax and offset.

Recently, neural networks are used to learn a representation from the text using autoencoders (Wang et al., 2015; Zhang et al., 2016), recurrent networks (Bansal et al., 2016), and convolutional networks (Zheng et al., 2017; Catherine and Cohen, 2017). These methods treat different words in the document as equal importance and do not match word semantics with the specific user. Instead, we achieve to learn a personalized word embedding with the guidance of user-item interactions. That is, the importance of words is learned to match user preferences. The attention mechanism can be used to learn these importance weights. Memory Networks (MemNet) have been used in recommendation to model item content (Hu et al., 2018c; Huang et al., 2017), capture user neighborhood (Ebesu et al., 2018), and learn latent relationships (Tay et al., 2018). We follow this thread to adapt a MemNet to match word semantics with user preferences.

In this paper, we propose a novel neural framework to exploit relational interactions and text content seamlessly. The proposed Personalized Neural Embedding (PNE) model fuses semantic representations learnt from unstructured text with behavior representations learnt from user-item interactions jointly for effective estimation of user preferences on items. PNE estimates the preference probability by two kinds of factors. The *behavior factor* is to capture the personalized preference of a user to an item learned from behavior interactions. The *semantic factor* is to capture the high-level representation attentively extracted from the unstructured text by matching word semantics with user preferences.

To model the behavior factor, we adopt a neural CF approach, which learns the user-item nonlinear interaction relationships using a neural network (CFNet). To model the semantic factor, we adopt a memory network to match word semantics with the specific user via the attention mechanism inherent in the memory module (MemNet), determining which words are highly relevant to the user preferences. PNE integrates relational interactions with unstructured text by bridging neural CF and memory networks. PNE can also learn meaningful word embeddings.

2 Approach

We present PNE to jointly learn representations of users, items, and words. PNE seamlessly captures nonlinear user-item interaction relationships and matches word semantics with user preferences.

Denote the set of users by $\mathcal{U}$ and items by $\mathcal{I}$. We use a rating matrix $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ to describe user-item interactions where each entry $y_{ui} \in \{0, 1\}$ is 1 (observed entries) if user u has an interaction with item i and 0 (unobserved entries) otherwise. Usually the interaction matrix is very sparse since a user $u \in \mathcal{U}$ only consumed a very small subset of all items. For the task of item recommendation, each user is only interested in identifying $topK$ items (typically K is small e.g. tens or hundreds). Items are ranked by their predicted scores:

$$\hat{y}_{ui} = f(u, i|\Theta), \tag{1}$$

where f is an interaction function and Θ denotes model parameters.

2.1 Architecture

PNE consists of a CF network (CFNet) to learn a nonlinear interaction function and of a memory network (MemNet) to match word semantics with user preferences. The information flow in PNE goes from the input (u, i) to the output $\hat{y}_{ui}$ through the following five modules.

1. *Input:* $(u, i) \rightarrow (\vec{e}_u, \vec{e}_i)$ This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i, and maps them into one-hot encodings $\vec{e}_u \in \{0, 1\}^{|\mathcal{U}|}$ and $\vec{e}_i \in \{0, 1\}^{|\mathcal{I}|}$ where only the element corresponding to that user/item index is 1 and all others are 0.

2. *Embedding:* $(\vec{e}_u, \vec{e}_i) \rightarrow \boldsymbol{x}_{ui}$ This module firstly embeds one-hot encodings into continuous representations $\boldsymbol{x}_u = \boldsymbol{P}^T \vec{e}_u$ and $\boldsymbol{x}_i = \boldsymbol{Q}^T \vec{e}_i$ by embedding matrices $\boldsymbol{P} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and $\boldsymbol{Q} \in \mathbb{R}^{|\mathcal{I}| \times d}$ respectively, where d is the latent dimension. It then concatenates them as $\boldsymbol{x}_{ui} = [\boldsymbol{x}_u, \boldsymbol{x}_i]$ to be the input of following CFNet and MemNet modules.

3. *CFNet:* $\boldsymbol{x}_{ui} \rightarrow z_{ui}^{behavior}$ This module is a CF approach to exploit user-item interactions. It takes continuous representations from the embedding

Dataset	#user	#item	#rating	#word	#density	avg. words
Amazon	8,514	28,262	56,050	1,845,387	0.023%	65.3
Cheetah	15,890	84,802	477,685	612,839	0.035%	7.2

Table 1: Datasets and statistics.

module and then transforms to a final *behavior factor* representation:

$$\boldsymbol{z}_{ui}^{\text{behavior}} = \text{ReLU}(\boldsymbol{W}\boldsymbol{x}_{ui} + \boldsymbol{b}), \tag{2}$$

where $\text{ReLU}(x) = \max(0, x)$ is an activation function, and $\boldsymbol{W}$ and $\boldsymbol{b}$ are connection weights and biases.

4. *MemNet:* $\boldsymbol{x}_{ui} \to \boldsymbol{z}_{ui}^{semantic}$ This module is to model the item content with the guidance of user-item interaction. The item content is modelled by memories. It takes representations from both the embedding module and the review text d_{ui} associated with the corresponding user-item (u, i) into a final *semantic factor* representation:

$$\boldsymbol{z}_{ui}^{\text{semantic}} = \sum\nolimits_{j:\, w_j \in d_{ui}} \text{Softmax}(a_j^{u,i})\boldsymbol{c}_j, \tag{3}$$

where the external memory slot $\boldsymbol{c}_j$ is an embedding vector for word w_j by mapping it with an external memory matrix $\boldsymbol{C}$. The attentive weight $a_j^{u,i}$ encodes the relevance of user u to word w_j by content-based addressing:

$$a_j^{u,i} = \boldsymbol{x}_{ui}^T \boldsymbol{m}_j^{u,i}, \tag{4}$$

where memory $\boldsymbol{m}_j^{u,i}$ is concatenated from internal memory slots $\{\boldsymbol{m}_j^u, \boldsymbol{m}_j^i\}$ which are mapped from word w_j by internal memory matrices $\boldsymbol{A}^u$ for user attention and $\boldsymbol{A}^i$ for item attention.

5. *Output:* $\boldsymbol{z}_{ui} \to \hat{y}_{ui}$ This module predicts the recommendation score $\hat{y}_{ui}$ for a given user-item pair based on the representation of both behavior factor and semantic factor from CFNet and MemNet respectively: $\boldsymbol{z}_{ui} = [\boldsymbol{z}_{ui}^{\text{behavior}}, \boldsymbol{z}_{ui}^{\text{semantic}}]$. The output is the probability that the input pair is a positive interaction. This is achieved by a logistic layer:

$$\hat{y}_{ui} = \frac{1}{1 + \exp(-\boldsymbol{h}^T \boldsymbol{z}_{ui})}, \tag{5}$$

where $\boldsymbol{h}$ is model parameter.

2.2 Learning

We adopt the binary cross-entropy loss:

$$\mathcal{L}(\Theta) = -\sum_{(u,i)\in\mathcal{S}} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}), \tag{6}$$

where $\mathcal{S} = \boldsymbol{Y}^+ \cup \boldsymbol{Y}^-$ is the union of observed interactions and randomly sampled negative examples. Model parameters $\Theta = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{A}, \boldsymbol{h}\}$ where we use a single word embedding matrix $\boldsymbol{A}$ by sharing all memory matrices $\boldsymbol{A}^u$, $\boldsymbol{A}^i$, and $\boldsymbol{C}$ in order to reduce model complexity. The objective function can be optimized by stochastic gradient descent.

Baselines	Shallow method	Deep method
CF	BPR	MLP
CF w/ text	HFT, TBPR	LCMR, PNE (ours)

Table 2: Categorization of recommender approaches.

3 Experiment

In this section, we evaluate PNE on two datasets with five baselines in terms of three metrics.

3.1 Datasets

We evaluate on two real-world datasets. The public **Amazon** products (McAuley and Leskovec, 2013) and a company **Cheetah** Mobile news (Hu et al., 2018c; Liu et al., 2018) (see Table 1). We preprocess the data following the strategy in (Wang and Blei, 2011). The size of word vocabulary is 8,000.

3.2 Evaluation protocol

We adopt leave-one-out evaluation (Hu et al., 2018b) and use three ranking metrics: hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR).

We compare with five baselines (see Table 2).

- BPR (Rendle et al., 2009) is a latent factor model based on matrix factorization.
- HFT (McAuley and Leskovec, 2013) adopts topic distributions to learn latent factors from text reviews.
- TBPR (Hu and Dai, 2017) extends BPR by integrating text content via word embedding features. Word embeddings used in TBPR are pre-trained by GloVe (Pennington et al., 2014).

TopK	Metric	Method					
		BPR	HFT	TBPR	MLP	LCMR	PNE
5	HR	8.10	10.77	15.17	21.00*	20.24	**23.52**
	NDCG	5.83	8.15	12.08	14.86*	14.51	**16.46**
	MRR	5.09	7.29	11.04	12.83*	12.63	**14.13**
10	HR	12.04	13.60	17.77	28.36*	28.36*	**31.86**
	NDCG	7.10	9.07	12.91	16.97*	16.78	**19.15**
	MRR	5.61	7.67	11.38	13.71*	13.56	**15.24**
20	HR	18.21	27.82	22.68	38.20	39.51*	**42.21**
	NDCG	8.64	12.52	14.14	18.99	19.18*	**21.75**
	MRR	6.02	8.54	11.71	14.26*	14.20	**15.95**

Table 3: Results (×100) on Amazon dataset. Best baseline marked with asterisk and best result in boldfaced.

TopK	Metric	Method					
		BPR	HFT	TBPR	MLP	LCMR	PNE
5	HR	43.80	49.66	49.48	53.80	54.76*	**56.48**
	NDCG	39.71	36.17	42.98*	41.21	41.89	**43.45**
	MRR	36.06	31.75	38.26*	37.02	37.62	**39.11**
10	HR	49.41	55.80	54.66	61.76	63.11*	**64.24**
	NDCG	41.82	40.93	44.99*	43.81	44.60	**45.98**
	MRR	36.94	33.65	39.13*	38.10	38.74	**40.16**
20	HR	53.98	65.47	61.23	67.93	69.27*	**69.52**
	NDCG	43.16	43.79	46.82*	45.29	46.19	**47.32**
	MRR	37.30	34.45	39.58*	38.51	39.18	**40.53**

Table 4: Results (×100) on Cheetah dataset. Best baseline marked with asterisk and best result in boldfaced.

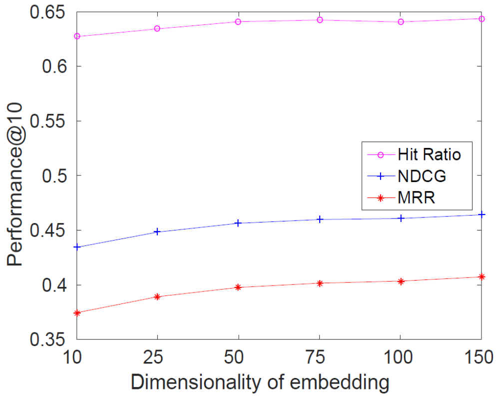


Figure 1: Dimension of embedding.

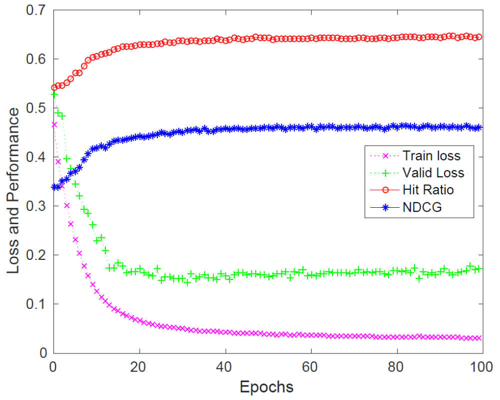


Figure 2: Loss and performance@10.

- MLP (He et al., 2017) is a neural CF approach. Note that, CFNet of PNE is an MLP with only one hidden layer.

- LCMR (Hu et al., 2018c) is a deep model for CF with unstructured text. Note that, MemNet of PNE is the same with the local MemNet of LCMR with only one-hop hidden layer.

Our method is implemented by TensorFlow (Abadi et al., 2016). Parameters are randomly initialized from Gaussian with optimizer Adam (Kingma and Ba, 2015). Learning rate is 0.001, batch size is 128, the ratio of negative sampling is 1.

3.3 Results

Results on two datasets are shown in Table 3 and Table 4, respectively. We have some observations. First, PNE outperforms the neural CF method MLP on two datasets in terms of three ranking metrics. On Amazon dataset, PNE obtains a large improvement in performance gain with relative 12.3% HR@10, 7.7% NDCG@10, and 6.2% MRR@5. On Cheetah Mobile dataset, PNE obtains a large improvement in performance gain with relative 5.0% HR@5, and 4.2% NDCG@5, and 3.9% MRR@5. Since the CFNet component of PNE is a neural CF method (with only one hidden layer), results show the benefit of exploiting unstructured text to alleviate the data sparsity issue faced by CF methods (BPR and MLP).

Second, PNE outperforms the traditional hybrid methods HFT and TBPR on two datasets in terms of three ranking metrics. On Amazon dataset, PNE obtains a significantly large improvement in performance gain with relative 55.0% HR@5, 28.9% NDCG@5, and 20.4% MRR@5. On Cheetah Mobile dataset, PNE still obtains reasonably large improvements with relative 17.5% HR@10, 1.8% NDCG@10, and 1.9% MRR@10. Compared with traditional hybrid methods which integrate the text content using topic modelling or word embeddings, results show the benefit of integrating text information through memory networks (and exploiting the interaction data through neural CF).

Last, PNE outperforms neural hybrid method LCMR by a large margin on Amazon dataset with relative improvements of 16.2% HR@5, 9.6% NDCG, and 7.4% MRR@5. PNE obtains reasonable improvements on Cheetah Mobile dataset with relative improvements of 3.1% HR@5, 2.8% NDCG, and 2.7% MRR. The design of CFNet of PNE is

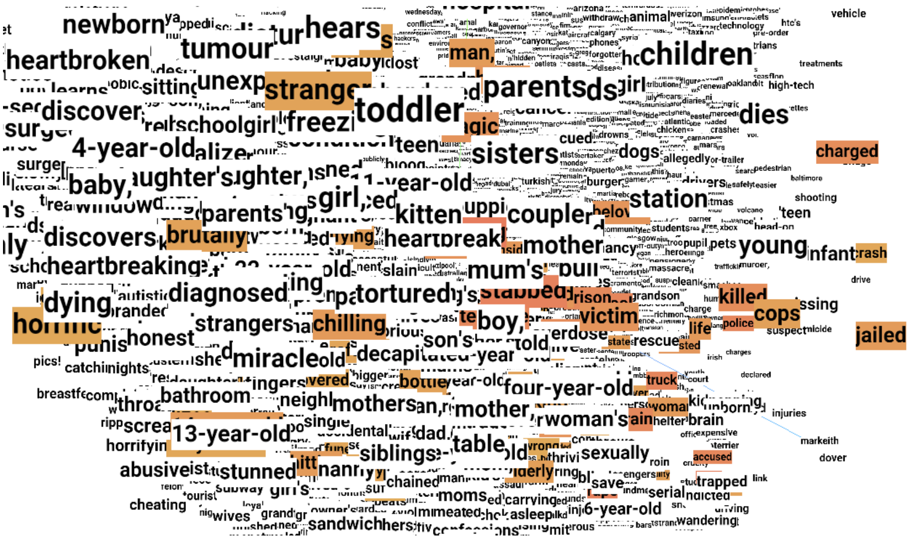


Figure 3: Visualization of word embeddings.

more reasonable than that of centralized memory module of LCMR which is equivalent to use a softmax activation between two hidden layers. The results show the effectiveness of fusing strategy in PNE to exploit unstructured text via MemNet and the interaction data via CFNet.

3.4 Analysis

We first evaluate the effects of the dimensionality of the embedding space. The x-axis in Figure 1 is the dimension of user/item and hence the dimensionality of input to CFNet is double since we adopt concatenation. It clearly indicates that the embedding should not be too small due to the possibility of information loss and the limits of expressiveness. The dimension 75 (and hence $d = 150$) is a good tradeoff between recommendation performance and computation burden.

We next show optimization curves of performance and loss (averaged over all examples) against iterations on Cheetah Mobile dataset in Figure 2. The model learns quickly in the first 20 iterations and improves slowly until 50, while training losses continue to go down and valid losses stabilize. The average time per epoch of PNE takes 68.1s and as a reference it is 34.5s for MLP using one NVIDIA TITAN Xp GPU.

3.5 Visualization

We visualize the learned word embeddings $\boldsymbol{A}$. We show that we can learn meaningful semantics for word embeddings such that words are to cluster when they have relevant semantics. We give an example to show the neighbors of the word “drug” in the 3D space by projecting the high-dimensional word vectors using TensorFlow [1] as shown in Figure 3. The top nearest neighbors of *drug* are: *shot, shoots, gang, murder, killing, rape, stabbed, truck, school, police, teenage*. We can see they are highly semantic relevant. We may also infer that school teenagers have close relationships to the drug issue from the Cheetah News corpus. This should raise a concern for society and it shows the society impact of natural language processing (Hovy and Spruit, 2016). Try our trained word embeddings [2].

4 Conclusion

We showed that relational interactions can be effectively integrated with unstructured text under a neural embedding model. Our method attentively focuses relevant words to match user preferences with user and item attentions (semantic factor) and captures nonlinear relationships between users and items (behavior factor). Experiments show better performance than five baselines on two real-world datasets in terms of three ranking metrics. We learn meaningful word embeddings and rethink the society impact of language processing technology.

Acknowledgments

The work is supported by HK CERG projects 16211214/16209715/16244616, NSFC 61673202, and HKPFS PF15-16701.

---

[1] `https://projector.tensorflow.org/`

[2] `https://www.dropbox.com/sh/ef74fpagf6sd137/AACXF6EnEY6QBdmJcyIW4RE0a?dl=0`

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM.

Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2–8. AAAI Press.

Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 288–296. ACM.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

Gintare Karolina Dziugaite and Daniel M Roy. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*.

Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 195–204, New York, NY, USA. ACM.

Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: improving rating predictions using review text content. In *International Workshop on the Web and Databases (WebDB)*, volume 9, pages 1–6.

Ruining He and Julian McAuley. 2016. Vbpr: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 144–150. AAAI Press.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 591–598.

Guang-Neng Hu and Xin-Yu Dai. 2017. Integrating reviews into personalized ranking for cold start recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 708–720. PAKDD.

Guang-Neng Hu, Xin-Yu Dai, Feng-Yu Qiu, Rui Xia, Tao Li, Shu-Jian Huang, and Jia-Jun Chen. 2018a. Collaborative filtering with topic and social latent factors incorporating implicit feedback. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):23.

Guang-Neng Hu, Xin-Yu Dai, Yunya Song, Shu-Jian Huang, and Jia-Jun Chen. 2015. A synthetic approach for recommendation: combining ratings, social relations, and reviews. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Guangneng Hu, Yu Zhang, and Qiang Yang. 2018b. Conet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 667–676. ACM.

Guangneng Hu, Yu Zhang, and Qiang Yang. 2018c. L-cmr: Local and centralized memories for collaborative filtering with unstructured text. *arXiv preprint arXiv:1804.06201*.

Haoran Huang, Qi Zhang, and Xuanjing Huang. 2017. Mention recommendation for twitter with end-to-end memory network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1872–1878. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer*, (8):30–37.

Bo Liu, Ying Wei, Yu Zhang, Zhixian Yan, and Qiang Yang. 2018. Transferable contextual bandit for cross-domain recommendation. In *AAAI Conference on Artificial Intelligence,*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 729–739. International World Wide Web Conferences Steering Committee.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.

Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 425–434. ACM.

An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models

Alexandra Chronopoulou[1], **Christos Baziotis**[1], **Alexandros Potamianos**[1,2,3]
[1]School of ECE, National Technical University of Athens, Athens, Greece
[2] Signal Analysis and Interpretation Laboratory (SAIL), USC, Los Angeles, USA
[3] Behavioral Signal Technologies, Los Angeles, USA
`el12068@central.ntua.gr`, `cbaziotis@mail.ntua.gr`
`potam@central.ntua.gr`

Abstract

A growing number of state-of-the-art transfer learning methods employ language models pretrained on large generic corpora. In this paper we present a conceptually simple and effective transfer learning approach that addresses the problem of catastrophic forgetting. Specifically, we combine the task-specific optimization function with an auxiliary language model objective, which is adjusted during the training process. This preserves language regularities captured by language models, while enabling sufficient adaptation for solving the target task. Our method does not require pretraining or finetuning separate components of the network and we train our models end-to-end in a single step. We present results on a variety of challenging affective and text classification tasks, surpassing well established transfer learning methods with greater level of complexity.

1 Introduction

Pretrained word representations captured by Language Models (LMs) have recently become popular in Natural Language Processing (NLP). Pretrained LMs encode contextual information and high-level features of language, modeling syntax and semantics, producing state-of-the-art results across a wide range of tasks, such as named entity recognition (Peters et al., 2017), machine translation (Ramachandran et al., 2017) and text classification (Howard and Ruder, 2018).

However, in cases where contextual embeddings from language models are used as additional features (e.g. ELMo (Peters et al., 2018)), results come at a high computational cost and require task-specific architectures. At the same time, approaches that rely on fine-tuning a LM to the task at hand (e.g. ULMFiT (Howard and Ruder, 2018)) depend on pretraining the model on an extensive vocabulary and on employing a sophisticated slanted triangular learning rate scheme to adapt the parameters of the LM to the target dataset.

We propose a simple and effective transfer learning approach, that leverages LM contextual representations and does not require any elaborate scheduling schemes during training. We initially train a LM on a Twitter corpus and then transfer its weights. We add a task-specific recurrent layer and a classification layer. The transferred model is trained end-to-end using an auxiliary LM loss, which allows us to explicitly control the weighting of the pretrained part of the model and ensure that the distilled knowledge it encodes is preserved.

Our contributions are summarized as follows: 1) We show that transfer learning from language models can achieve competitive results, while also being intuitively simple and computationally effective. 2) We address the problem of catastrophic forgetting, by adding an auxiliary LM objective and using an unfreezing method. 3) Our results show that our approach is competitive with more sophisticated transfer learning methods. We make our code widely available. [1]

2 Related Work

Unsupervised pretraining has played a key role in deep neural networks, building on the premise that representations learned for one task can be useful for another task. In NLP, pretrained word vectors (Mikolov et al., 2013; Pennington et al., 2014) are widely used, improving performance in various downstream tasks, such as part-of-speech tagging (Collobert et al., 2011) and question answering (Xiong et al., 2016). These pretrained word vectors serve as initialization of the embedding layer and remain frozen during training, while our pretrained language model also initializes the hidden layers of the model and is fine-tuned to each

---

[1] `/github.com/alexandra-chron/siatl`

Proceedings of NAACL-HLT 2019, pages 2089–2095
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

classification task.

Aiming to learn from unlabeled data, Dai and Le (2015) use unsupervised objectives such as sequence autoencoding and language modeling for as pretraining methods. The pretrained model is then fine-tuned to the target task. However, the fine-tuning procedure of the language model to the target task does not include an auxiliary objective. Ramachandran et al. (2017) also pretrain encoder-decoder pairs using language models and fine-tune them to a specific task, using an auxiliary language modeling objective to prevent catastrophic forgetting. This approach, nevertheless, is only evaluated on machine translation tasks; moreover, the seq2seq (Sutskever et al., 2014) and language modeling losses are weighted equally throughout training. By contrast, we propose a weighted sum of losses, where the language modeling contribution gradually decreases. ELMo embeddings (Peters et al., 2018) are obtained from language models and improve the results in a variety of tasks as additional contextual representations. However, ELMo embeddings rely on character-level models, whereas our approach uses a word-level LM. They are, furthermore, concatenated to pretrained word vectors and remain fixed during training. We instead propose a fine-tuning procedure, aiming to adjust a generic architecture to different end tasks.

Moreover, BERT (Devlin et al., 2018) pretrains language models and fine-tunes them on the target task. An auxiliary task (next sentence prediction) is used to enhance the representations of the LM. BERT fine-tunes masked bi-directional LMs. Nevertheless, we are limited to a uni-directional model. Training BERT requires vast computational resources, while our model only requires 1 GPU. We note that our approach is not orthogonal to BERT and could be used to improve it, by adding an auxiliary LM objective and weighing its contribution.

Towards the same direction, ULMFiT (Howard and Ruder, 2018) shows impressive results on a variety of tasks by employing pretrained LMs. The proposed pipeline requires three distinct steps, that include (1) pretraining the LM, (2) fine-tuning it on a target dataset with an elaborate scheduling procedure and (3) transferring it to a classification model. Our proposed model is closely related to ULMFiT. However, ULMFiT trains a LM and fine-tunes it to the target dataset, before transferring it to a classification model. While fine-tuning the LM to the target dataset, the metric (e.g. accuracy) that we intend to optimize cannot be observed. We propose adopting a multi-task learning perspective, via the addition of an auxiliary LM loss to the transferred model, to control the loss of the pretrained and the new task simultaneously. The intuition is that we should avoid catastrophic forgetting, but at the same time allow the LM to distill the knowledge of the prior data distribution and keep the most useful features.

Multi-Task Learning (MTL) via hard parameter sharing (Caruana, 1993) in neural networks has proven to be effective in many NLP problems (Collobert and Weston, 2008). More recently, alternative approaches have been suggested that only share parameters across lower layers (Sogaard and Goldberg, 2016). By introducing part-of-speech tags at the lower levels of the network, the proposed model achieves competitive results on chunking and CCG super tagging. Our auxiliary language model objective follows this line of thought and intends to boost the performance of the higher classification layer.

3 Our Model

We introduce **SiATL**, which stands for **Si**ngle-step **A**uxiliary loss **T**ransfer **L**earning. In our proposed approach, we first train a LM. We then transfer its weights and add a task-specific recurrent layer to the final classifier. We also employ an auxiliary LM loss to avoid catastrophic forgetting.

LM Pretraining. We train a word-level language model, which consists of an embedding LSTM layer (Hochreiter and Schmidhuber, 1997), 2 hidden LSTM layers and a linear layer. We want to minimize the negative log-likelihood of the LM:

$$L(\hat{p}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T^n}\log\hat{p}(x_t^n|x_1^n, ..., x_{t-1}^n) \quad (1)$$

where $\hat{p}(x_t^n|x_1^n, ..., x_{t-1}^n)$ is the distribution of the t^{th} word in the n^{th} sentence given the $t-1$ words preceding it and N is total number of sentences.

Transfer & auxiliary loss. We transfer the weights of the pretrained model and add one LSTM with a self-attention mechanism (Lin et al., 2017; Bahdanau et al., 2015).

In order to adapt the contribution of the pretrained model to the task at hand, we introduce an auxiliary LM loss during training. The joint loss is the

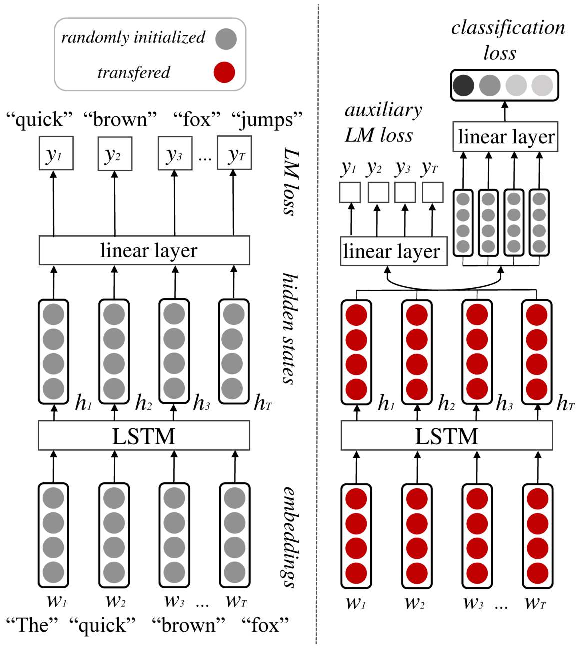


Figure 1: High-level overview of our proposed TL architecture. We transfer the pretrained LM add an extra recurrent layer and an auxiliary LM loss.

weighted sum of the task-specific loss L_{task} and the auxiliary LM loss L_{LM}, where γ is a weighting parameter to enable adaptation to the target task but at the same time keep the useful knowledge from the source task. Specifically:

$$L = L_{task} + \gamma L_{LM} \tag{2}$$

Exponential decay of γ. An advantage of the proposed TL method is that the contribution of the LM can be explicitly controlled in each training epoch. In the first few epochs, the LM should contribute more to the joint loss of SiATL so that the task-specific layers adapt to the new data distribution. After the knowledge of the pretrained LM is transferred to the new domain, the task-specific component of the loss function is more important and γ should become smaller. This is also crucial due to the fact that the new, task-specific LSTM layer is randomly initialized. Therefore, by backpropagating the gradients of this layer to the pretrained LM in the first few epochs, we would add noise to the pretrained representation. To avoid this issue, we choose to initially pay attention to the LM objective and gradually focus on the classification task. In this paper, we use an exponential decay for γ over the training epochs.

Sequential Unfreezing. Instead of fine-tuning all the layers simultaneously, we propose unfreezing them sequentially, according to Howard and Ruder (2018); Chronopoulou et al. (2018). We first fine-tune only the extra, randomly initialized LSTM and the output layer for $n-1$ epochs. At the n^{th} epoch, we unfreeze the pretrained hidden layers. We let the model fine-tune, until epoch $k-1$. Finally, at epoch k, we also unfreeze the embedding layer and let the network train until convergence. The values of n and k are obtained through grid search. We find the sequential unfreezing scheme important, as it minimizes the risk of overfitting to small datasets.

Optimizers. While pretraining the LM, we use Stochastic Gradient Descent (SGD). When we transfer the LM and fine-tune on each classification task, we use 2 different optimizers: SGD for the pretrained LM (embedding and hidden layer) with a small learning rate, in order to preserve its contextual information. As for the new, randomly initialized LSTM and classification layers, we employ Adam (Kingma and Ba, 2015), in order to allow them to train fast and adapt to the target task.

Dataset	Domain	# classes	# examples
Irony18	Tweets	4	4618
Sent17	Tweets	3	61854
SCv2	Debate Forums	2	3260
SCv1	Debate Forums	2	1995
PsychExp	Experiences	7	7480

Table 1: Datasets used for the downstream tasks.

4 Experiments and Results

4.1 Datasets

To pretrain the language model, we collect a dataset of 20 million English Twitter messages, including approximately 2M unique tokens. We use the 70K most frequent tokens as vocabulary. We evaluate our model on five datasets: *Sent17* for sentiment analysis (Rosenthal et al., 2017), *PsychExp* for emotion recognition (Wallbott and Scherer, 1986), *Irony18* for irony detection (Van Hee et al., 2018), *SCv1* and *SCv2* for sarcasm detection (Oraby et al., 2016; Lukin and Walker, 2013). More details about the datasets can be found in Table 1.

4.2 Experimental Setup

To preprocess the tweets, we use *Ekphrasis* (Baziotis et al., 2017). For the generic datasets, we use NLTK (Loper and Bird, 2002). For the *NBoW* baseline, we use *word2vec* (Mikolov et al., 2013) 300-dimensional embeddings as features.

	Irony18	Sent17	SCv2	SCv1	PsychExp
BoW	43.7	61.0	65.1	60.9	25.8
NBoW	45.2	63.0	61.1	51.9	20.3
P-LM	42.7 ± 0.6	61.2 ± 0.7	69.4 ± 0.4	48.5 ± 1.5	38.3 ± 0.3
P-LM + su	41.8 ± 1.2	62.1 ± 0.8	69.9 ± 1.0	48.4 ± 1.7	38.7 ± 1.0
P-LM + aux	45.5 ± 0.9	65.1 ± 0.6	72.6 ± 0.7	55.8 ± 1.0	40.9 ± 0.5
SiATL (P-LM + aux + su)	**47.0** ± 1.1	**66.5** ± 0.2	**75.0** ± 0.7	**56.8** ± 2.0	**45.8** ± 1.6
ULMFiT (Wiki-103)	23.6 ± 1.6	60.5 ± 0.5	**68.7** ± 0.6	**56.6** ± 0.5	21.8 ± 0.3
ULMFiT (Twitter)	**41.6** ± 0.7	**65.6** ± 0.4	67.2 ± 0.9	44.0 ± 0.7	**40.2** ± 1.1
State of the art	53.6	68.5	76.0	69.0	57.0
	(Baziotis et al., 2018)	(Cliche, 2017)	(Ilic et al., 2018)	(Felbo et al., 2017)	

Table 2: Ablation study on various downstream datasets. Average over five runs with standard deviation. *BoW* stands for Bag of Words, *NBoW* for Neural Bag of Words. *P-LM* stands for a classifier initialized with our pretrained LM, *su* for sequential unfreezing and *aux* for the auxiliary LM loss. In all cases, F_1 is employed.

For the neural models, we use an LM with an embedding size of 400, 2 hidden layers, 1000 neurons per layer, embedding dropout 0.1, hidden dropout 0.3 and batch size 32. We add Gaussian noise of size 0.01 to the embedding layer. A clip norm of 5 is applied, as an extra safety measure against exploding gradients. For each text classification neural network, we add on top of the transferred LM an LSTM layer of size 100 with self-attention and a softmax classification layer. In the pretraining step, SGD with a learning rate of 0.0001 is employed. In the transferred model, SGD with the same learning rate is used for the pretrained layers. However, we use Adam (Kingma and Ba, 2015) with a learning rate of 0.0005 for the newly added LSTM and classification layers. For developing our models, we use PyTorch (Paszke et al., 2017) and Scikit-learn (Pedregosa et al., 2011).

5 Results & Discussion

Baselines and Comparison. Table 2 summarizes our results. The top two rows detail the baseline performance of the BoW and NBoW models. We observe that when enough data is available (e.g. *Sent17*), baselines provide decent results. Next, the results for the generic classifier initialized from a pretrained LM (*P-LM*) are shown with and without sequential unfreezing, followed by the results of the proposed model SiATL. SiATL is also directly compared with its close relative ULMFiT (trained on Wiki-103 or Twitter) and the state-of-the-art for each task; ULMFiT also fine-tunes a LM for classification tasks. The proposed SiATL method consistently outperforms the baselines, the *P-LM* method and ULMFiT in all datasets. Even though we do not perform any elaborate learning rate scheduling and we limit ourselves to pretraining in Twitter, we obtain higher results in two Twitter datasets and three generic.

Auxiliary LM objective. The effect of the auxiliary objective is highlighted in very small datasets, such as *SCv1*, where it results in an impressive boost in performance (7%). We hypothesize that when the classifier is simply initialized with the pretrained LM, it overfits quickly, as the target vocabulary is very limited. The auxiliary LM loss, however, permits refined adjustments to the model and fine-grained adaptation to the target task.

Exponential decay of γ. For the optimal γ interval, we empirically find that exponentially decaying γ from 0.2 to 0.1 over the number of training epochs provides best results for our classification tasks. A heatmap of γ is depicted in Figure 3. We observe that small values of γ should be employed, in order to scale the LM loss in the same order of magnitude as the classification loss over the training period. Nevertheless, the use of exponential decay instead of linear decay does not provide a significant improvement, as our model is not sensitive to the way of decaying hyperparameter γ.

Sequential Unfreezing. Results show that sequential unfreezing is crucial to the proposed method, as it allows the pretrained LM to adapt to the target word distribution. The performance improvement is more pronounced when there is a mismatch between the LM and task domains, i.e., the non-Twitter domain tasks. Specifically for the *PsychExp* and *SCv2* datasets, sequentially unfreezing yields significant improvement in F_1 building upon our intuition.

Number of training examples. Transfer learning is particularly useful when limited training data are available. We notice that for our largest dataset

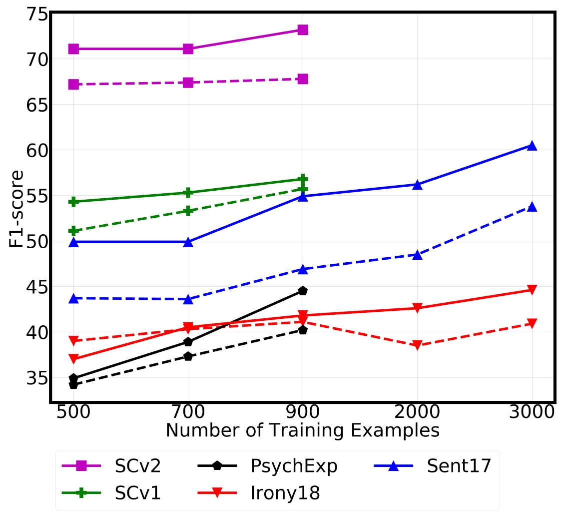


Figure 2: Results of SiATL, our proposed approach (continuous lines) and ULMFiT (dashed lines) for different datasets (indicated by different markers) as a function of the number of training examples.

Sent17, SiATL outperforms ULMFiT only by a small margin when trained on all the training examples available (see Table 2), while for the small *SCv2* dataset, SiATL outperforms ULMFiT by a large margin and ranks very close to the state-of-the-art model (Ilic et al., 2018). Moreover, the performance of SiATL vs ULMFiT as a function of the training dataset size is shown in Figure 2. Note that the proposed model achieves competitive results on less than 1000 training examples for the *Irony18, SCv2, SCv1* and *PsychExp* datasets, demonstrating the robustness of SiATL even when trained on a handful of training examples.

Catastrophic forgetting. We observe that SiATL indeed provides a way of mitigating catastrophic forgetting. Empirical results that are shown in Table 2 indicate that by only adding the auxiliary language modeling objective, we obtain better results on all downstream tasks. Specifically, a comparison of the *P-LM + aux* model and the *P-LM* model shows that the performance of SiATL on classification tasks is improved by the auxiliary objective. We hypothesize that the language model objective acts as a regularizer that prevents the loss of the most generalizable features.

6 Conclusions and Future Work

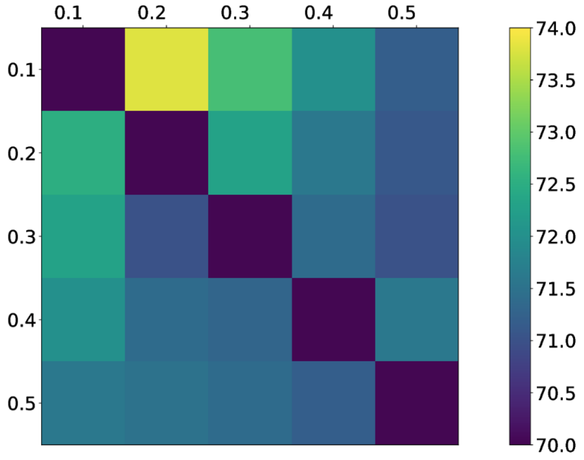


Figure 3: Heatmap of the effect of γ to F_1-score, evaluated on *SCv2*. The horizontal axis depicts the initial value of γ and the vertical axis the final value of γ.

We introduce SiATL, a simple and efficient transfer learning method for text classification tasks. Our approach is based on pretraining a LM and transferring its weights to a classifier with a task-specific layer. The model is trained using a task-specific functional with an auxiliary LM loss. SiATL avoids catastrophic forgetting of the language distribution learned by the pretrained LM. Experiments on various text classification tasks yield competitive results, demonstrating the efficacy of our approach. Furthermore, our method outperforms more sophisticated transfer learning approaches, such as ULMFiT in all tasks.

In future work, we plan to move from Twitter to more generic domains and evaluate our approach to more tasks. Additionally, we aim at exploring ways for scaling our approach to larger vocabulary sizes (Kumar and Tsvetkov, 2019) and for better handling of out-of-vocabulary words (OOV) (Mielke and Eisner, 2018; Sennrich et al., 2015) in order to be applicable to diverse datasets.

Finally, we want to explore approaches for improving the adaptive layer unfreezing process and the contribution of the language model objective (value of γ) to the target task.

Acknowledgments

We would like to thank Katerina Margatina and Georgios Paraskevopoulos for their helpful suggestions and comments. This work has been partially supported by computational time granted from the Greek Research & Technology Network (GR-NET) in the National HPC facility - ARIS. Also, the authors would like to thank NVIDIA for supporting this work by donating a TitanX GPU.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, California.

Christos Baziotis, Athanasiou Nikolaos, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 613–621, New Orleans, Louisiana.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48.

Alexandra Chronopoulou, Aikaterini Margatina, Christos Baziotis, and Alexandros Potamianos. 2018. Ntua-slp at iest 2018: Ensemble of neural transfer methods for implicit emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 57–64, Brussels, Belgium.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 573–580, Vancouver, Canada.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine learning*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, pages 2493–2537.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bjarke Felbo, Alan Mislove, Anders Sogaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, (8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Annual Meeting of the ACL*, pages 328–339, Melbourne, Australia.

Suzana Ilic, Edison Marrese-Taylor, Jorge A. Balazs, and Yutaka Matsuo. 2018. Deep contextualized word representations for detecting sarcasm and irony. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–7.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *International Conference on Learning Representations*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 30–40, Atlanta, Georgia.

Sebastian J. Mielke and Jason Eisner. 2018. Spell once, summon anywhere: A two-level open-vocabulary language model. *CoRR*, abs/1804.08205.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3111–3119.

Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn A. Walker. 2016. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *Proceedings of the*

SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 31–41.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, pages 2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the Annual Meeting of the ACL*, pages 1756–1765, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the NAACL:HLT*, pages 2227–2237, New Orleans, Louisiana.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Anders Sogaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the Annual Meeting of the ACL*, pages 231–235, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3104–3112.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 39–50, New Orleans, Louisiana.

Harald G. Wallbott and Klaus R. Scherer. 1986. How universal and specific is emotional experience? evidence from 27 countries on five continents. *Information (International Social Science Council)*, (4):763–795.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering.

Incorporating Emoji Descriptions Improves Tweet Classification

Abhishek Singh, **Eduardo Blanco** and **Wei Jin**
Computer Science and Engineering
University of North Texas
Denton, TX, 76203
`abhisheksingh@my.unt.edu, {eduardo.blanco,wei.jin}@unt.edu`

Abstract

Tweets are short messages that often include specialized language such as hashtags and emojis. In this paper, we present a simple strategy to process emojis: replace them with their natural language description and use pretrained word embeddings as normally done with standard words. We show that this strategy is more effective than using pretrained emoji embeddings for tweet classification. Specifically, we obtain new state-of-the-art results in irony detection and sentiment analysis despite our neural network is simpler than previous proposals.

1 Introduction

Tweets are short messages shared on Twitter, one of the most popular social networking services with 326 million monthly active users word wide (Twitter, 2018). Tweets often use specialized language such as abbreviations (e.g., *TBH: To be honest*), hashtags (e.g., *#NBAFinals*), emoticons and emojis. The Oxford Dictionary defines an emoticon as “a facial expression such as a smile or frown, formed by various combinations of keyboard characters” (e.g., “:)”, “:-(”), and an emoji as “a small digital image or icon used to express an idea or emotion” (e.g., 😀, 👏🏼, 😷). While the number of emoticons is relatively small, the Unicode Standard includes over 2,800 emojis.

Emojis are interesting because they succinctly encode meaning that otherwise would require more than one word to convey (e.g., *grinning face*, *clapping hands* and *face with medical mask* for the emojis above). Additionally, emojis have become popular in social media. 5 billion emojis are sent daily on Facebook (Burge, 2018). While only 6% of the top-100 Facebook headlines used emojis in 2015, 52% did so in 2017 (Boland, 2017). Over 14% of tweets and 50% of Instagram posts contain at least one emoji (Cruse, 2015; Moon, 2015).

	Irony?
I just love being ignored😊|#not 😒	Yes
Love it when my mans on a cleaning spree.. Saves me doing it 👌🏼😘	No

	Sentiment
@Paul_OConnor187 hi we going to see ted 2 at the Odeon cinemas at Glasgow on Wednesday 😊	Positive
Serato DJ isn’t compatible with Windows 10 yet 😞 ...got to spin on my old laptop Saturday.	Negative

Table 1: Sample tweets with their irony and sentiment judgements. Note that the emojis help to determine irony usage and the author’s sentiment.

Irony detection and sentiment analysis in tweets are two popular tasks. Sentiment analysis has received substantially more attention than irony detection. Irony, however, is a major error source in sentiment analysis (0.71 F1 overall but 0.29 F1 with ironic tweets (Hee et al., 2018)), and natural language understanding in general does not generalize well with ironic texts (Liu et al., 2012; Maynard and Greenwood, 2014).

In this paper, we tackle both irony and sentiment analysis in tweets—two classification tasks. In particular, we focus on modeling emojis. Consider the examples in Table 1. Understanding the emojis is critical to making irony and sentiment judgements. In the first example, the contrast between the emojis helps determining that irony is present (the hashtag *#not* also helps). In the second tweet, the *OK hand sign* and *face blowing a kiss* emojis help reinforcing that the author is praising somebody and not being ironic. Similarly, the smiling and sad emojis in the last two examples are a clear

Proceedings of NAACL-HLT 2019, pages 2096–2101
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

sign of the author's sentiment towards the movie *Ted 2* and the incompatibility issue.

The main contributions of this paper are twofold. First, we present a simple strategy to model emojis: replace them with their textual description. Second, we show that this strategy outperforms previous methods and yields a new state-of-the-art in two tweet classification tasks: irony detection and sentiment analysis.

2 Related Work

Irony is closely related to sarcasm. The Oxford Dictionary defines irony as "The expression of one's meaning by using language that normally signifies the opposite, typically for humorous or emphatic effect", and sarcasm as "The use of irony to mock or convey contempt." Given these definitions, it is not surprising that many researchers do not distinguish between them (Maynard and Greenwood, 2014). The top-3 systems to detect irony are built with neural networks and pretrained word embeddings. Baziotis et al. (2018) build an ensemble of two stacks of BiLSTMs (word and character level) with attention. Wu et al. (2018) propose a BiLSTM and a multitask learning framework (hashtag, irony presence and irony type prediction), and complement the input text with sentiment features extracted from lexicons. Vu et al. (2018) propose a multilayer perceptron taking as input an embedding for the input text (average of word embeddings) as well as manually crafted lexical, syntactic, semantic and polarity features. Our strategy to incorporate emojis outperforms all of them (Table 3).

Sentiment analysis in tweets has been studied for years (Nakov et al., 2013). At its core, it is the task of classifying a tweet into expressing positive, neutral or negative sentiment (Rosenthal et al., 2017). Initial systems were primarily based on sentiment lexicons and manually extracted features, but the state of the art uses neural networks and word embeddings. Baziotis et al. (2017) propose a stack of two BiLSTMs at the word level and do not use any lexicons. Cliche (2017) presents a CNN and BiLSTM ensemble and experiment with three pretrained embeddings. Rouvier (2017) also presents a CNN and BiLSTM ensemble but incorporates manually defined features (e.g., word presence in emotion lexicons, all-caps). The strategy presented here to incorporate emojis outperforms all these systems (Table 4).

Within natural language processing and social media, emojis have received considerable attention. Barbieri et al. (2016) train emoji embeddings with word2vec and discover that the closest words are sound (e.g., ☕: coffee, roasters, caffeine, latte). Eisner et al. (2016) propose a complementary approach to train emoji embeddings (Section 3). Emojis have also been used as labels for distant supervision to improve tweet classification (Felbo et al., 2017). The strategy presented here to incorporate emojis is simpler and more effective than previous ones, does not require additional pretraining or domain specific corpora, and can be used with any neural architecture that takes text as input without any modifications. Simply put, we replace emojis with their textual descriptions and leverage existing pretrained word embeddings.

3 Strategies to Incorporate Emojis

Neural networks that take as input text usually transform the input tokens into pretrained embeddings. When the input text are tweets, it is common to use embeddings pretrained with large collections of tweets as opposed to general purpose text (Li et al., 2017; Pennington et al., 2014).

Emojis as Regular Tokens. The simplest option to incorporate emojis into a neural network is to consider them as any other token in the input text (Barbieri et al., 2016). This strategy relies on having seen enough instances of each emoji in the texts with which embeddings were pretrained—otherwise the embeddings will not capture the semantics of emoji tokens properly.

Emoji Embeddings. Another strategy is to use separate embeddings for emojis. Eisner et al. (2016) pretrain emoji embeddings using positive and negative (randomly sampled) emoji descriptions. Descriptions are transformed into a vector by adding the corresponding word2vec embeddings (Mikolov et al., 2013). Emoji embeddings are tuned quickly because only a positive and a negative description per emoji are considered.

We refer to this strategy as EMJ-EMBED.

Our Strategy: Emoji Descriptions. Our strategy is simple: replace emojis with their textual descriptions. Effectively, this eliminates all emojis in the input and incorporates a rather detailed description—several tokens—of the emojis (see examples in Table 2). Our rationale is as follows. First, lists of emojis and their textual descriptions

Emoji	Description
😂	Face with tears of joy
😘	Face blowing a kiss
😄	Grinning face with smiling eyes
😌	Relieved face
😝	Squinting face with tongue
😥	Sad but relieved face
😠	Angry face
😭	Loudly crying face
😓	Downcast face with sweat
😰	Anxious face with sweat

Table 2: Emojis and their textual description.

are readily available.[1] Second, while emojis are common (Section 1), words are more common. Thus, it is reasonable to expect word embeddings to capture the meaning of words better than emoji embeddings capture the meaning of emojis. Consider the last two examples in Table 2, 😓 and 😰. These emojis are relatively uncommon, and pretrained emoji embeddings do not leverage the fact that both of them are *faces* with *sweat*. Using the textual descriptions bypasses both issues.

Finally, this strategy is straightforward, fast to implement and run, and can be used regardless of the neural network architecture. Indeed, it could be considered a preprocessing step.

We refer to this strategy as EMJ-DESC.

4 Experiments and Results

We experiment with two tweet classification tasks: irony detection and sentiment analysis. We use standard corpora and compare with previous work using the same set up (i.e., we train and test with exactly the same instances they did).

4.1 Experimental Setup

Corpora. For irony detection, we use the corpus released by Hee et al. (2018). It includes two tasks: binary (Task A: yes or no) and 4-way multiclass irony detection (Task B: verbal irony realized through polarity contrast, other verbal irony, situational irony or non-irony). The corpus consists of 3,000 tweets (yes: 2396, no: 604; verbal irony with polarity contrast: 1,728, other verbal irony: 267, situational irony: 401, non-irony: 604). Here are some examples: *I love waking up with migraines #not* 😣 (verbal, polarity contrast), *I cared for 8 seconds, then I got distracted.* 😂 (other verbal), *I wonder what Professor Iaukea has to say about the new Disney Princess...?* 😒 (situational), and *Is Obamacare Slowing Health Care Spending? #NOT* (non-irony).

For sentiment analysis, we use the corpus released by Rosenthal et al. (2017), which has 62,617 tweets (positive: 22,277, neutral: 28,528, negative: 8,982). Table 1 shows examples of positive and negative sentiment, and here is an example of neutral sentiment: *I'm switching to T-Mobile tomorrow and I'm getting a new number.*

Evaluation Metrics. We follow the metrics used by previous work. Regarding irony detection, we report accuracy and macro-average F1 (all labels weighted equal regardless of frequency). Regarding sentiment analysis, we report accuracy, average recall and F1. Following previous work, we calculate accuracy and average recall using all labels (positive, negative and neutral) but F1 using only positive and negative instances.

Preprocessing. We preprocess the input text following standard steps. Specifically, we tokenize with the NLTK's TweetTokenizer (Bird, 2006), lowercase all text, and use regular expressions to remove stop words, numbers, urls, consecutive repeated words and Twitter users (i.e., tokens whose first character is '@'). We also expand hashtags (e.g., *#PickANewSong*: *Pick a new song*) with ekphrasis (Baziotis et al., 2017).

Regarding emojis, we either (a) do nothing special and use pretrained emoji embeddings (EMJ-EMBED strategy), or (b) replace emojis with their textual description and use pretrained word embeddings for the words in their descriptions (EMJ-DESC strategy).

Let us consider the following tweet: "@Paul_OConnor187 hi we going to see ted 2 at the Odeon cinemas at Glasgow on Wednesday 😊". After preprocessing, we transform it into "hi we going see ted odeon cinemas glasgow wednesday 😊" or "hi we going see ted odeon cinemas glasgow wednesday smiling face" (EMJ-EMBED and EMJ-DESC strategies respectively).

Neural Network Architecture. We experiment with a stack of two BiLSTMs (Dyer et al., 2015) with attention (Zhou et al., 2016) to generate distributed representations of the input, and a softmax layer as the output layer. This architecture is simpler than previous proposals, but as we shall see,

---

[1] `https://pypi.org/project/emoji-unicode/`, `https://github.com/uclmr/emoji2vec`, `https://emojipedia.org/`

		Task A		Task B	
		Acc.	F1	Acc.	F1
Previous Work (Top 3)	Vu et al. (2018)	0.7015	0.6476	0.6594	0.4437
	Wu et al. (2018)	0.7347	0.7054	0.6046	0.4947
	Baziotis et al. (2018)	0.7883	0.7856	0.6888	0.5358
This paper	EMJ-EMBED	0.7864	0.7814	0.6940	0.5434
	EMJ-DESC	**0.8056**	**0.8031**	**0.7187**	**0.5565**

Table 3: Results on irony detection (Accuracy and Macro F1). Task A is a binary classification (yes / no) and Task B is a four-way classification (verbal irony with polarity contrast, other verbal irony, situational irony, non-irony).

		Avg. Rec.	Acc.	F1
Previous Work (Top 3)	Baziotis et al. (2017)	0.681	0.651	0.677
	Cliche (2017)	0.681	0.658	0.685
	Rouvier (2017)	0.676	0.661	0.674
This paper	EMJ-EMBED	0.703	0.689	0.691
	EMJ-DESC	**0.728**	**0.704**	**0.703**

Table 4: Results on sentiment analysis (three-way classification: positive, neural or negative).

outperforms previous work when coupled with our strategy to incorporate embeddings. Regarding word embeddings, we use the ones trained by Baziotis et al. (2018) using word2vec (Mikolov et al., 2013) and 550 million tweets. Regarding emoji embeddings, we use emoji2vec (Eisner et al., 2016). Note that EMJ-EMBED uses both word and emoji embeddings whereas EMJ-DESC only uses word embeddings.

4.2 Results

Irony Detection. Table 3 presents the results for irony detection. EMJ-EMBED obtains virtually the same results than the state of the art, although the neural architecture is much simpler (Section 2). EMJ-DESC, however, obtains the best results to date: (Task A: 0.80 vs. 0.78 F1, Task B: 0.55 vs. 0.53 F1). These results show that replacing embeddings with their textual descriptions and using the corresponding word embeddings is more effective than using emoji embeddings. As discussed earlier, there is a large amount of emojis (over 2,800), and some of them are infrequent. Many have, however, words in common in their descriptions thus leveraging the descriptions is beneficial.

Sentiment Analysis. Table 4 presents results for sentiment analysis. The standard evaluation metric in this task is average recall (Rosenthal et al., 2017), but we also provide accuracy and F1. Both EMJ-EMBED and EMJ-DESC outperform the state of the art despite we experiment with a simpler neural architecture. Indeed, EMJ-DESC outperforms previous work by a substantial margin (+0.047, +6.9% avg. recall). The reason for these results is the same than for irony detection: modeling emojis is key for tweet classification and our strategies to incorporate emojis are better suited than the ones used by previous work, which primarily treat them as any other token.

5 Conclusions

We have presented a strategy to incorporate emojis into any neural network: replace them with their textual descriptions. This strategy does not require any additional pretraining or component in the network. Instead, it leverages pretrained word embeddings, which are readily available and pretrained using massive amounts of text (Mikolov et al., 2013; Pennington et al., 2014).

Experimental results show that our strategy is more effective than previous ones (either consider emojis as regular tokens or use specialized emoji embeddings). Indeed, we obtain new state-of-the-art results in two tweet classification tasks: irony detection and sentiment analysis. We hypothesize that this is due to two reasons. First, while emojis are common, the words in their descriptions are more common. Thus, there is more data to pretrain word embeddings than emoji embeddings. Simply put, there is more proper English texts available than social media text with emojis. Second, emoji descriptions have many words in common (Table

2), thus many emojis benefit from a single word embedding (e.g., the textual descriptions of 😌 and 😥 share the words *face* and *relieved*).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants Nos. 1734730, 1832267 and 1845757. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016. What does this emoji mean? A vector space skip-gram model for twitter emojis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Christos Baziotis, Athanasiou Nikolaos, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018*, pages 613–621.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006.*

Gabriele Boland. 2017. How emoji usage has exploded by 766% on social. `https://www.newswhip.com/2017/12/emoji-on-social/`. [Online; accessed Nov 17th, 2018].

Jeremy Burge. 2018. 5 billion emojis sent daily on messenger. [Online; accessed Nov 17th, 2018].

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 573–580, Vancouver, Canada. Association for Computational Linguistics.

Joe Cruse. 2015. Emoji usage in TV conversation. `https://blog.twitter.com/official/en_us/a/2015/emoji-usage-in-tv-conversation.html`. [Online; accessed Dec 5th, 2018].

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 334–343.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media, SocialNLP@EMNLP 2016, Austin, TX, USA, November 1, 2016*, pages 48–54.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1615–1625.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana, June 5-6, 2018*, pages 39–50.

Quanzhi Li, Sameena Shah, Xiaomo Liu, and Armineh Nourbakhsh. 2017. Data sets: Word embeddings learned from tweets and general data. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 428–436.

Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Diana Maynard and Mark A. Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 4238–4243.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Mariella Moon. 2015. Instagram takes a serious look at how people use emojis. `https://www.engadget.com/2015/05/02/instagram-emoji-study/`. [Online; accessed Dec 5th, 2018].

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 312–320.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada. Association for Computational Linguistics.

Mickael Rouvier. 2017. Lia at semeval-2017 task 4: An ensemble of neural networks for sentiment classification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 760–765, Vancouver, Canada. Association for Computational Linguistics.

Twitter. 2018. Q3 2018 Earnings Report. `https://investor.twitterinc.com/static-files/4bfbf376-fefd-43cc-901e-aedd6a7f1daf`. [Online; accessed Dec 5th, 2018].

Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. 2018. NIHRIO at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. *CoRR*, abs/1804.00520.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. Thu_ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56. Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Modeling Personal Biases in Language Use by Inducing Personalized Word Embeddings

Daisuke Oba[1], **Naoki Yoshinaga**[2], **Shoetsu Sato**[1], **Satoshi Akasaki**[1], **Masashi Toyoda**[2]
[1]The University of Tokyo, Japan
[2]Institute of Industrial Science, the University of Tokyo, Japan
[1]{oba, shoetsu, akasaki}@tkl.iis.u-tokyo.ac.jp
[2]{ynaga, toyoda}@iis.u-tokyo.ac.jp

Abstract

There exist biases in individual's language use; the same word (*e.g.*, *cool*) is used for expressing different meanings (*e.g.*, *temperature range*) or different words (*e.g.*, *cloudy*, *hazy*) are used for describing the same meaning. In this study, we propose a method of modeling such personal biases in word meanings (hereafter, *semantic variations*) with personalized word embeddings obtained by solving a task on subjective text while regarding words used by different individuals as different words. To prevent personalized word embeddings from being contaminated by other irrelevant biases, we solve a task of identifying a review-target (objective output) from a given review. To stabilize the training of this extreme multi-class classification, we perform a multi-task learning with metadata identification. Experimental results with reviews retrieved from RateBeer confirmed that the obtained personalized word embeddings improved the accuracy of sentiment analysis as well as the target task. Analysis of the obtained personalized word embeddings revealed trends in semantic variations related to frequent and adjective words.

1 Introduction

When we verbalize what we have sensed, there exist inevitable personal biases in word meanings (hereafter, (*personal*) *semantic variations*). For example, when we say "this pizza is greasy," how greasy can vary widely among individuals. When we see the same beer, we may use different words (*e.g.*, *red*, *amber*) to refer its color. The semantic variations will thereby cause problems not only in communicating with each other, but also in building natural language processing (NLP) systems.

Several studies have attempted to personalize models to improve the performance on NLP tasks such as sentiment analysis (Gao et al., 2013) and dialogue systems (Li et al., 2016; Zhang et al., 2018). All of these studies, however, tried to estimate *subjective* output from *subjective* input (*e.g.*, estimating sentiment scores given by reviewers). These personalized models are thereby affected by not only semantic variations in subjective input but also *annotation bias* (deviation of outputs given by the annotators) and *selection bias* (deviation of outputs caused by the deviation of input) (§ 2). This makes it difficult to understand the pure impact of the personal semantic variations.

In this study, aiming at understanding semantic variations and their impact on NLP tasks, we propose a method for modeling personal semantic variations with personalized word embeddings obtained through the review-target identification task. This task estimates the review-target (*objective* output) from a given review (*subjective* input) (§ 3), and is free from *annotation bias* since output labels are given a priori. Also, *selection bias* can be suppressed by using a dataset in which the same reviewer evaluates the same target only once, so as not to learn the deviation of output labels caused by the choice of inputs. To stabilize the training of this extreme multi-class classification, we apply multi-task learning (MTL) with metadata estimation of the review-target to effectively learn a reliable model (personalized word embeddings).

We validate our hypothesis that words related to the five senses have large semantic variations. We first confirm the impact of personalized word embeddings in the review-target identification using a review dataset obtained from RateBeer, and next evaluate their usefulness in sentiment analysis (§ 4.2). Analysis of the obtained personalized word embeddings on three metrics (frequency, dissemination and polysemy) reveals trends on which words have large semantic variations (§ 4.3).

The contributions of this paper are as follows:

- We established a method to obtain personal semantic variations via multi-task learning on a task with objective outputs (§ 3).
- We categorized personal biases in NLP (§ 2).

Proceedings of NAACL-HLT 2019, pages 2102–2108
Minneapolis, Minnesota, June 2 - June 7, 2019.

- We confirmed the usefulness of personalized word embeddings in review-target identification and sentiment analysis tasks (§ 4.2).
- We revealed trends in personal semantic variations (§ 4.3).

2 Related Work

As discussed in § 1, biases considered by personalization in NLP tasks have three facets: (1) *semantic variation* in task inputs (biases in how people use words; our target) (2) *annotation bias* of output labels (biases in how annotators label) and (3) *selection bias* of output labels (biases in how people choose perspectives (*e.g.,* review-targets) that directly affects outputs (*e.g.,* polarity labels)).

Existing studies have modeled (2) and (3) with or without (1) for NLP tasks such as sentiment analysis (Li et al., 2011; Gao et al., 2013; Tang et al., 2015a,b; Chen et al., 2016), machine translation (Mirkin and Meunier, 2015; Michel and Neubig, 2018; Wuebker et al., 2018), and dialogue systems (Li et al., 2016; Zhang et al., 2018). However, it is difficult to untangle the different facets of personal biases, there is no study aiming to analyze solely personal semantic variations. Meanwhile, word embeddings induced for a simple NLP task such as sentiment classification conveys less information, which are not suitable for analyzing semantic variations.

Computational linguists have utilized word embeddings to capture semantic variations of words caused by diachronic (Hamilton et al., 2016; Szymanski, 2017; Rosenfeld and Erk, 2018; Jaidka et al., 2018), geographic (Bamman et al., 2014; Garimella et al., 2016) or domain (Tredici and Fernández, 2017) differences. In these studies, they have mainly discussed relationships between semantic variations of words and their frequency, dissemination (the number of users), or polysemy of the words. Hamilton et al. (2016) report that the meanings of more frequent words are more stable over time, and the meanings of polysemous words are likely to change over time since polysemous words appear in diverse contexts (Winter et al., 2014; Bréal, 1897). Tredici and Fernández (2017) report that the meanings of words used by more people are more stable. In this study, we analyze the personal semantic variations by inducing personalized word embeddings, mainly focusing on how frequent, disseminated or polysemous words are biased, following these studies.

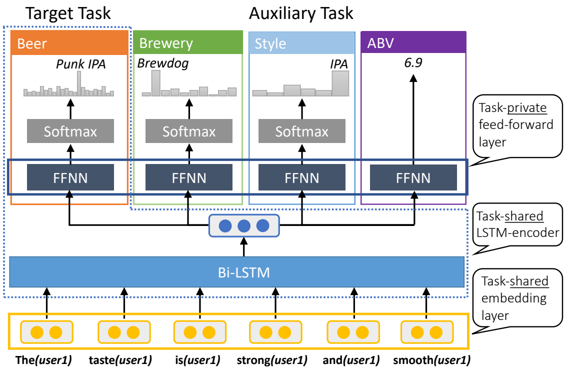


Figure 1: The overview of our model.

3 Personalized Word Embeddings

This section describes our neural network-based model (Figure 1) designed for inducing personalized word embeddings via review-target identification. This model estimates the review-target from a given review.

Model Overview: The whole process is as follows. First, a given review, represented as a sequence of words, is transformed to a sequence of their word embeddings via an embedding layer. Here, our model regards words written by different reviewers as different words for personalization. Next, we apply bi-directional long-short term memory (Bi-LSTM) (Gers et al., 1999) to the sequence of word embeddings and use the concatenation of outputs from the forward and backward LSTMs as a review representation. Finally, a feed-forward layer computes an output probability distribution from the encoded representation of the review.

Multi-task Learning (MTL): The extremely large number of labels (review-targets) makes it difficult to stably train the target identification model. To mitigate this, we jointly train auxiliary tasks that estimate the metadata of the review-target along with the target task. This approach assumes that understanding metadata contributes the performance of the target identification. Concretely, our MTL model contains a task-shared embedding layer, a task-shared LSTM-encoder, and task-private feed-forward layers similarly to (Dong et al., 2015; Luong et al., 2016). In our experiments, these task-private layers consist of three layers for classification and one layer for regression (Figure 1). In the classification tasks, the model computes log probability over target labels as the output and cross-entropy is used as the loss function. In the regression task, the output is the metadata itself represented as a scalar value and squared error is used as the loss function.

Here, multi-task learning raises a new problem. In auxiliary tasks, since the same reviewer can select the same label multiple times, the personalized word embeddings trained through the multi-task learning may implicitly include the *selection bias* of the output labels depending on the reviewers. Therefore, to exclude those irrelevant biases from the personalized embeddings, we introduce personalized bias terms to feed-forward layers of each task. These bias terms are fixed to the prior distributions of outputs in the training set depending on reviewers so that they absorb selection biases instead of personalized word embeddings.

4 Experiments

We first evaluate the effect of personalization in the target identification task. Next, to confirm the usefulness of the obtained personalized embeddings, we exploit them to solve a sentiment analysis task for extrinsic evaluation. Finally, we analyze the degree and tendencies of semantic variations captured by the obtained personalized word embeddings.

4.1 Settings

Data For training and intrinsic evaluation, we use a huge review dataset about beers constructed from RateBeer[1] (McAuley and Leskovec, 2013). It contains 2,924,163 reviews about 110,369 types of beers with various metadata (*e.g.,* brewery name, style, rating, etc.) written by 29,265 reviewers. From this dataset, we extracted 527,157 reviews about 83,049 types of beers written by the top-100 reviewers who wrote the most reviews, to guarantee enough data size per reviewer. After that, we randomly divided these reviews into training (421,725), development (52,716), and test sets (52,716) in the ratio of 8:1:1. We refer to this dataset as **RateBeer dataset**.

Tasks Our target task takes a beer review and estimates the target beer reviewed in it. Regarding the metadata estimated in multi-task learning (MTL), we chose **style** with 89 types and **brewery** with 6,208 types for classification tasks and **alcohol by volume (ABV)** for a regression task. Note that these metadata are objective and our MTL is free from annotation bias.

# Layers of Bi-LSTM	1
Dimensions of LSTM output	200
Dimensions of word embeddings	200
Dropout rate	0.5
Mini-batch size	400
Initial learning rate	0.005
Vocabulary size (w/o personalization)	23,556
Vocabulary size (w/ personalization)	469,346

Table 1: Hyperparameters of our model.

In the sentiment analysis task, we estimate the **ratings** of given reviews annotated by the reviewers. The ratings are integers and range from 1 to 20. Here, we solve this task as a regression task since it is natural to treat the fine-grained rating as continuous values.

Models and Hyperparameters In the review-target and its metadata identification tasks, we compare our model described in § 3 with five models with different settings.[2] Their differences are, (1) whether the model is trained through MTL, (2) whether personalization is applied to the embeddings, and (3) whether personalization is applied to the bias term in the output layers. When MTL is not employed, multiple models are independently trained by tasks without sharing layers.

Table 1 shows major hyperparameters. We initialize the embedding layer by pretrained skip-gram embeddings (Mikolov et al., 2013) induced from the training set of RateBeer dataset. The vocabulary is defined by all the words that appeared more than or equal to 10 times in the training set, and the top-100 reviewers have used at least once. For optimization, we train the models up to 100 epochs with Adam (Kingma and Ba, 2015) and select the one at the epoch with the best results on the development set.[3]

In the sentiment analysis task for extrinsic evaluation of the obtained personalized word embeddings, we train another set of models with the same architecture and hyperparameters as the review-target identification models in Figure 1 except that they have only one feed-forward layer for the target regression task. The embedding layers of the models are kept fixed after initialized by the word embeddings extracted from the corresponding review-target identification models with the same settings of personalization and MTL.

---

[1] `https://www.ratebeer.com/`

[2] All of our models were implemented by PyTorch (`https://pytorch.org/`) in the version of 0.4.0.

[3] Regarding MTL, we select the model at the epoch with the best results in the target task.

Model			Target task	Auxiliary tasks		
MTL	personalized		beer	brewery	style	ABV(%)
	emb.	bias	[Acc.(%)]	[Acc.(%)]	[Acc.(%)]	[RMSE]
			2.99	**8.70**	**46.60**	1.437
	✓		3.32	7.88	44.52	1.462
✓			3.81	8.03	44.12	**1.425**
✓	✓		4.14	7.41	43.74	1.467
✓	✓	✓	**4.47**	7.83	43.93	1.478
baseline			0.03	0.69	5.46	2.284

Table 2: Results on the review-target and its metadata identification.

Model			sentiment
MTL	personalized		rating
	emb.	bias	[RMSE]
			1.452
	✓		1.406
✓			1.447
✓	✓		1.381
✓	✓	✓	**1.377**
baseline			2.903

Table 3: Results on the sentiment analysis: embedding layers are kept fixed to those of the corresponding models in Table 2.

4.2 Results

Table 2 shows the accuracies on the three classification tasks (product, style, and brewery) and RMSE on the regression task (ABV) through the test sets. We can see two insights from the results: (1) In the target task, the model adopted all the methods outperformed others, (2) In the auxiliary tasks, MTL and personalization had no effect.

As for the first one, since the identification of the review-target requires both detailed understandings of all the related metadata and capturing biases of word meanings, our proposed method considering both elements achieved the best performance as a natural consequence. The second one is not surprising since the metadata estimated in the auxiliary tasks are weakly related to each other. Thus multi-task learning and personalization did not contribute to the improvement of these auxiliary tasks.

Finally, Table 3 shows the results of the sentiment analysis task for extrinsic evaluation. Similarly to the review-target identification, the model with both MTL and personalization performed the best. The personalization of output bias term also slightly improved RMSE. These results confirm that the personalized word embeddings trained through our methods successfully learned task-independent personal semantic variations. In other words, they were helpful even for solving tasks other than the review-target identification.

4.3 Analysis

In this section, we analyze the personalized word embeddings extracted from the best model with MTL and personalization to confirm what kind of personal biases exist in each word. Here, we target on only the words used by more than or equal to 30% of the reviewers excluding stopwords to remove the influences of low frequent words.

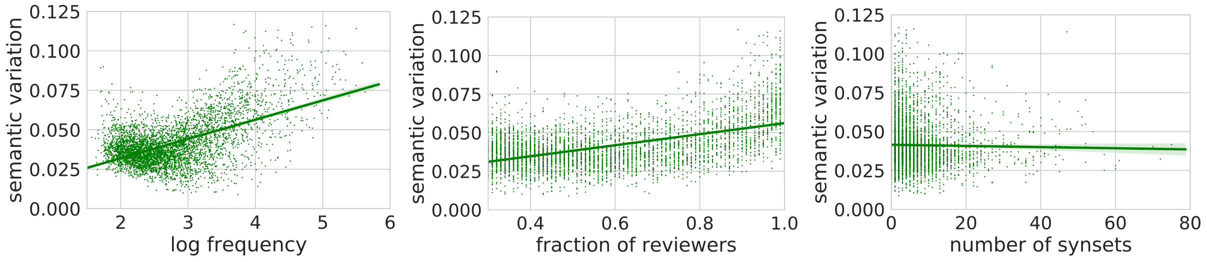
We first define the **personal semantic variations** of word w, to determine how the representations of the word are different by individuals, as:

$$\frac{1}{|\mathrm{U}(w_i)|}\sum_{u_j \in \mathrm{U}(w_i)} \left(1-\cos(\boldsymbol{e}_{w_i}^{u_j}, \bar{\boldsymbol{e}}_{w_i})\right) \tag{1}$$

where $\mathrm{U}(w_i)$ is the set of the reviewers who used the word w_i in the training set, $\boldsymbol{e}_{w_i}^{u_j}$ is the word embedding of w_i personalized to reviewer u_j, and $\bar{\boldsymbol{e}}_{w_i}$ is the average of $\boldsymbol{e}_{w_i}^{u_j}$ for $u_j \in \mathrm{U}(w_i)$.

Here, we focus on the three factors, **frequency**, **dissemination**, and **polysemy** which have been studied on semantic variations caused by diachronic, geographical or domain differences of text (see § 2). Figure 2 shows the semantic variations of words against the degree of the three metrics. The x-axes correspond to (a) log frequency of the word, (b) the ratio of the reviewers who used the word, and (c) the number of synsets found in WordNet (Miller, 1995) ver. 3.0, respectively.

Interestingly, in contrast to the results reported in previous studies (Hamilton et al., 2016; Tredici and Fernández, 2017), personal semantic variations correlate highly with frequency and dissemination, and poorly with polysemy in our results. This tendency can be explained as follows: In the dataset used in our experiments, words related to five senses such as “*mild*,” “*dry*” and “*soapy*” frequently appear and their usages depend on the feelings and experiences of each individual. Therefore, these words show high semantic variations. Regarding polysemy, although the semantic variations acquired through our method might change the degree or nuance of the word sense, they do not change its synset. This is because those words are still used only in highly skewed contexts related to beer where word senses and their meanings do not significantly fluctuate.



(a) log-frequency (b) dissemination (c) polysemy

Figure 2: Personal semantic variations: The Pearson coefficient correlations are (a) **0.55**, (b) **0.51**, (c) -0.02, respectively. The trendlines show 95% confidence intervals from bootstrapped kernel regressions.

top-50
surprisingly, **nice**, quite, **light**, **pleasant**, actually, though, buttery, **grassy**, really, **bready**, **dusty**, **fruity**, **decent**, **mild**, rather, little, **toffee**, **earthy**, **woody**, **subtle**, **nutty**, **strange**, even, still, **dry**, **tasty**, maybe, **medium**, bit, **soapy**, **interesting**, somewhat, malt, **pretty**, brewery, character, **solid**, lovely, **floral**, **herbal**, **grainy**, **big**, yet, nose, fruit, fairly, aroma, **good**, almost, **metallic**
bottom-50
lasted, primary, system, **secondary**, **personal**, test, acquired, ii, **greater**, standout, roof, england, flow, scored, purchase, partly, colorado, spare, rocks, ounce, se, jug, source, shipping, fullness, denmark, center, **diminished**, greatly, met, spirits, burns, comments, **surrounded**, scores, expectations, carmel, crew, die, **annual**, laces, reading, consumed, handpump, **disappeared**, suits, husks, duck, rise, meal, hall

Table 4: Top-50 words with the largest (and smallest) semantic variations. Adjectives are boldfaced.

Table 4 shows the top-50 words with the largest (and smallest) semantic variations. As can be seen from the table, the top-50 words contain much more adjectives (58%) compared with the bottom-50 ones (16%), which are likely to be used to represent our feelings depending on the five senses.

To see more precisely what kind of words have large semantic variations, we manually classify the adjectives of the top-50 (and bottom-50) by the five senses. From the results, on the RateBeer dataset, there were more words representing each sense except hearing in the top-50 words compared with the bottom-50 ones.

Finally, we analyze the relationships between words beyond the analysis focusing on the single word. We visualized the obtained personalized word embeddings of the word "*mild*" and some closest words in the embedding space as an example in Figure 3. From the results, intersection of the clusters (*e.g.*, "*grainy*" and "*grassy*") means that the same meaning can be represented in different ways by individuals.

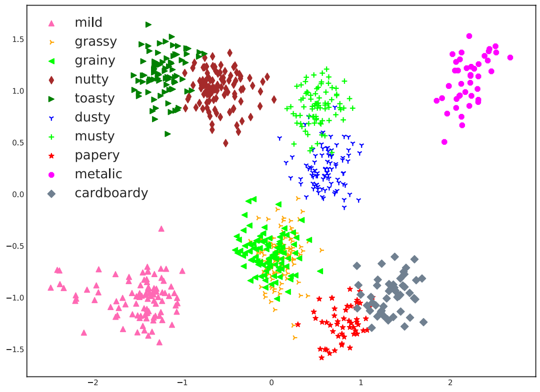


Figure 3: Two-dimensional visualization of the word "*mild*" with some closest words.

5 Conclusions

In this study, we proposed a method of modeling personal semantic variations with personalized word embeddings induced through the review-target identification task. The experimental results on the large-scale beer review dataset showed that personalized word embeddings obtained by multi-task learning with metadata identification improved the accuracy of sentiment analysis as well as the target task. Our analysis revealed that words related to the five senses and adjectives have large semantic variations.

We plan to analyze relationships between semantic variations and user factors of writers who used the target words such as age and gender. We will develop a generic method of inducing personalized word embeddings for any subjective text.

Acknowledgements

This work was partially supported by Commissioned Research (201) of the National Institute of Information and Communications Technology of Japan.

References

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014, Short Papers)*, pages 828–834.

Michel Bréal. 1897. The history of words. *The beginnings of semantics: Essays, lectures and reviews*, pages 152–175.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1650–1659.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 1723–1732.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP 2013)*, pages 1107–1111.

Aparna Garimella, Rada Mihalcea, and James Pennebaker. 2016. Identifying cross-cultural differences in word usage. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 674–683.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. In *Proceedings of the ninth International Conference on Artificial Neural Networks (ICANN 1999)*, pages 850–855.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1489–1501.

Kokil Jaidka, Niyati Chhaya, and Lyle Ungar. 2018. Diachronic degradation of language models: Insights from social media. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018, Short Papers)*, pages 195–200.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR 2015).*

Fangtao Li, Nathan Liu, Hongwei Jin, Kai Zhao, QiangYang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *Proceedings of the 22nd international joint conference on Artificial Intelligence (IJCAI 2011)*, pages 1820–1825.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 994–1003.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *International Conference on Learning Representations (ICLR 2016).*

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the seventh ACM conference on Recommender systems (RecSys 2013)*, pages 165–172.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018, Short Papers)*, pages 312–318.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS 2013)*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 2019–2025.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 474–484.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL, Short Papers)*, pages 448–453.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*, pages 1014–1023.

Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. 2015b. User modeling with neural network for review rating prediction. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*.

Marco Del Tredici and Raquel Fernández. 2017. Semantic variation in online communities of practice. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS 2017)*.

Bodo Winter, Graham Thompson, and Matthias Urban. 2014. Cognitive factors motivating the evolution of word meanings: Evidence from corpora, behavioral data and encyclopedic network structure. In *Evolution of Language: Proceedings of the tenth International Conference (EVOLANG10)*, pages 353–360.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 881–886.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 2204–2213.

Multi-Task Ordinal Regression for Jointly Predicting the Trustworthiness and the Leading Political Ideology of News Media

Ramy Baly[1], **Georgi Karadzhov**[2], **Abdelrhman Saleh**[3],
James Glass[1], **Preslav Nakov**[4]
[1]MIT Computer Science and Artificial Intelligence Laboratory, MA, USA
[2]SiteGround Hosting EOOD, Bulgaria, [3]Harvard University, MA, USA
[4]Qatar Computing Research Institute, HBKU, Qatar
`{baly, glass}@mit.edu`, `georgi.m.karadjov@gmail.com`
`abdelrhman_saleh@college.harvard.edu`, `pnakov@hbku.edu.qa`

Abstract

In the context of fake news, bias, and propaganda, we study two important but relatively under-explored problems: (*i*) *trustworthiness estimation* (on a 3-point scale) and (*ii*) *political ideology detection* (left/right bias on a 7-point scale) of entire news outlets, as opposed to evaluating individual articles. In particular, we propose a multi-task ordinal regression framework that models the two problems jointly. This is motivated by the observation that hyper-partisanship is often linked to low trustworthiness, e.g., appealing to emotions rather than sticking to the facts, while center media tend to be generally more impartial and trustworthy. We further use several auxiliary tasks, modeling centrality, hyper-partisanship, as well as left-vs.-right bias on a coarse-grained scale. The evaluation results show sizable performance gains by the joint models over models that target the problems in isolation.

1 Introduction

Recent years have seen the rise of social media, which has enabled people to virtually share information with a large number of users without regulation or quality control. On the bright side, this has given an opportunity for anyone to become a content creator, and has also enabled a much faster information dissemination. However, it has also opened the door for malicious users to spread disinformation and misinformation much faster, enabling them to easily reach audience at a scale that was never possible before. In some cases, this involved building sophisticated profiles for individuals based on a combination of psychological characteristics, meta-data, demographics, and location, and then micro-targeting them with personalized "fake news" with the aim of achieving some political or financial gains (Lazer et al., 2018; Vosoughi et al., 2018).

A number of fact-checking initiatives have been launched so far, both manual and automatic, but the whole enterprise remains in a state of crisis: by the time a claim is finally fact-checked, it could have reached millions of users, and the harm caused could hardly be undone. An arguably more promising direction is to focus on fact-checking entire news outlets, which can be done in advance. Then, we could fact-check the news before they were even written: by checking how trustworthy the outlets that published them are. Knowing the reliability of a medium is important not only when fact-checking a claim (Popat et al., 2017; Nguyen et al., 2018), but also when solving article-level tasks such as "fake news" and click-bait detection (Brill, 2001; Finberg et al., 2002; Hardalov et al., 2016; Karadzhov et al., 2017; De Sarkar et al., 2018; Pan et al., 2018; Pérez-Rosas et al., 2018)

Political ideology (or left/right bias) is a related characteristic, e.g., extreme left/right media tend to be propagandistic, while center media are more factual, and thus generally more trustworthy. This connection can be clearly seen in Figure 1.

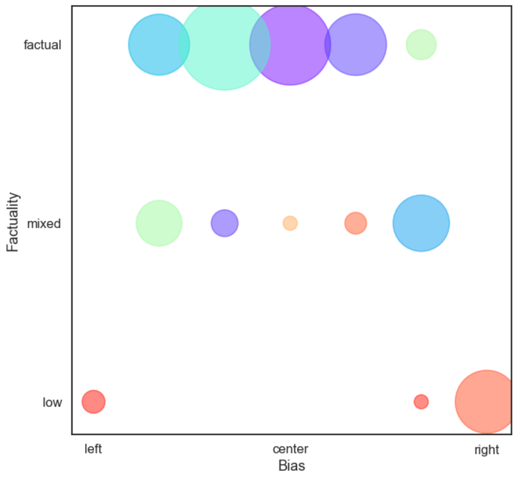


Figure 1: Correlation between bias and factuality for the news outlets in the Media Bias/Fact Check website.

Proceedings of NAACL-HLT 2019, pages 2109–2116
Minneapolis, Minnesota, June 2 - June 7, 2019.

Despite the connection between factuality and bias, previous research has addressed them as independent tasks, even when the underlying dataset had annotations for both (Baly et al., 2018). In contrast, here we solve them jointly. Our contributions can be summarized as follows:

- We study an under-explored but arguably important problem: predicting the factuality of reporting of news media. Moreover, unlike previous work, we do this jointly with the task of predicting political bias.
- As factuality and bias are naturally defined on an ordinal scale (factuality: from *low* to *high*, and bias: from *extreme-left* to *extreme-right*), we address them as ordinal regression. Using multi-task ordinal regression is novel for these tasks, and it is also an under-explored direction in machine learning in general.
- We design a variety of auxiliary subtasks from the bias labels: modeling centrality, hyper-partisanship, as well as left-vs.-right bias on a coarse-grained scale.

2 Related Work

Factuality of Reporting Previous work has modeled the factuality of reporting at the medium level by checking the general stance of the target medium with respect to known manually fact-checked claims, without access to gold labels about the overall medium-level factuality of reporting (Mukherjee and Weikum, 2015; Popat et al., 2016, 2017, 2018).

The trustworthiness of Web sources has also been studied from a Data Analytics perspective, e.g., Dong et al. (2015) proposed that a trustworthy source is one that contains very few false claims. In social media, there has been research targeting the user, e.g., finding malicious users (Mihaylov and Nakov, 2016; Mihaylova et al., 2018; Mihaylov et al., 2018), *sockpuppets* (Maity et al., 2017), *Internet water army* (Chen et al., 2013), and *seminar users* (Darwish et al., 2017).

Unlike the above work, here we study source reliability as a task in its own right, using manual gold annotations specific for the task and assigned by independent fact-checking journalists. Moreover, we address the problem as one of ordinal regression on a three-point scale, and we solve it jointly with political ideology prediction in a multi-task learning setup, using several auxiliary tasks.

Predicting Political Ideology In previous work, political ideology, also known as media bias, was used as a feature for "fake news" detection (Horne et al., 2018a). It has also been the target of classification, e.g., Horne et al. (2018b) predicted whether an article is biased (*political* or *bias*) vs. unbiased. Similarly, Potthast et al. (2018) classified the bias in a target article as (*i*) left vs. right vs. mainstream, or as (*ii*) hyper-partisan vs. mainstream. Left-vs-right bias classification at the article level was also explored by Kulkarni et al. (2018), who modeled both the textual and the URL contents of the target article. There has been also work targeting bias at the phrase or the sentence level (Iyyer et al., 2014), focusing on political speeches (Sim et al., 2013) or legislative documents (Gerrish and Blei, 2011), or targeting users in Twitter (Preoţiuc-Pietro et al., 2017). Another line of related work focuses on propaganda, which can be seen as a form of extreme bias (Rashkin et al., 2017; Barrón-Cedeño et al., 2019a,b). See also a recent position paper (Pitoura et al., 2018) and an overview paper on bias on the Web (Baeza-Yates, 2018). Unlike the above work, here we focus on predicting the political ideology of news media outlets.

In our previous work (Baly et al., 2018), we did target the political bias of entire news outlets, as opposed to working at the article level (we also modeled factuality of reporting, but as a separate task without trying multi-task learning). In addition to the text of the articles published by the target news medium, we used features extracted from its corresponding Wikipedia page and Twitter profile, as well as analysis of its URL structure and traffic information about it from Alexa rank. In the present work, we use a similar set of features, but we treat the problem as one of ordinal regression. Moreover, we model the political ideology and the factuality of reporting jointly in a multi-task learning setup, using several auxiliary tasks.

Multitask Ordinal Regression *Ordinal regression* is well-studied and is commonly used for text classification on an ordinal scale, e.g., for sentiment analysis on a 5-point scale (He et al., 2016; Rosenthal et al., 2017a). However, *multi-task ordinal regression* remains an understudied problem.

Yu et al. (2006) proposed a Bayesian framework for collaborative ordinal regression, and demonstrated that modeling multiple ordinal regression tasks outperforms single-task models.

Walecki et al. (2016) were interested in jointly predicting facial action units and their intensity level. They argued that, due to the high number of classes, modeling these tasks independently would be inefficient. Thus, they proposed the *copula ordinal regression* model for multi-task learning and demonstrated that it can outperform various single-task setups. We use this model in our experiments below.

Balikas et al. (2017) used multi-task ordinal regression for the task of fine-grained sentiment analysis. In particular, they introduced an auxiliary coarse-grained task on a 3-point scale, and demonstrated that it can improve the results for sentiment analysis on the original 5-point scale. Inspired by this, below we experiment with different granularity for political bias; however, we explore a larger space of possible auxiliary tasks.

3 Method

Copula Ordinal Regression We use the *Copula Ordinal Regression* (COR) model, which was originally proposed by Walecki et al. (2016) to estimate the intensities of facial action units (AUs). The model uses copula functions and conditional random fields (CRFs) to approximates the learning of the joint probability distribution function (PDF) of the facial AUs (random variables), using the bivariate joint distributions capturing dependencies between AU pairs. It was motivated by the fact that (*i*) many facial AUs co-exist with different levels of intensity, (*ii*) some AUs co-occur more often than others, and (*iii*) some AUs depend on the intensity of other units.

We can draw an analogy between modeling facial AUs and modeling news media, where each medium expresses a particular bias (political ideology) and can also be associated with a particular level of factuality. Therefore, bias and factuality can be analogous to the facial AUs in (Walecki et al., 2016), and represent two aspects of news reporting, each being modeled on a multi-point ordinal scale. In particular, we model bias on a 7-point scale (*extreme-left*, *left*, *center-left*, *center*, *center-right*, *right*, and *extreme-right*), and factuality on a 3-point scale (*low*, *mixed*, and *high*).

In our case, we train the COR model to predict the joint PDF between political bias and factuality of reporting. This could potentially work well given the inherent inter-dependency between the two tasks as we have seen on Figure 1.

Auxiliary Tasks We use a variety of auxiliary tasks, derived from the bias labels. This includes converting the 7-point scale to (*i*) 5-point and 3-point scales, similarly to (Balikas et al., 2017), and to (*ii*) a 2-point scale in two ways to model extreme partisanship, and centrality. Here is the list of the auxiliary tasks we use with precise definition of the label mappings:

- **Bias5-way:** Predict bias on a 5-pt scale; 1:*extreme-left*, 2:*left*, 3:{*center-left, center, center-right*}, 4:*right*, and 5:*extreme-right*.
- **Bias3-way:** Predict bias on a 3-pt scale; 1:{*extreme-left, left*}, 2:{*center-left, center, center-right*}, and 3:{*right, extreme-right*}.
- **Bias-extreme:** Predict extreme vs. non-extreme partisanship on a 2-pt scale; 1:{*extreme-left, extreme-right*}, 2:{*left, center-left, center, center-right, right*}.
- **Bias-center:** Predict center vs. non-center political ideology on a 2-pt scale, ignoring polarity: 1:{*extreme-left, left, right, extreme-right*}, 2:{*center-left, center, center-right*}.

Features We used the features from (Baly et al., 2018)[1]. We gathered a sample of articles from the target medium, and we calculated features such as POS tags, linguistic cues, sentiment scores, complexity, morality, as well as embeddings. We also used the Wikipedia page of the medium (if any) to generate document embedding. Then, we collected metadata from the medium's Twitter account (if any), e.g., whether is is verified, number of followers, whether the URL in the Twitter page matches the one of the medium. Finally, we added Web-based features that (*i*) model the orthographic structure of the medium's URL address, and (*ii*) analyze the Web-traffic information about the medium's website, as found in Alexa rank.[2]

4 Experiments and Evaluation

Data We used the MBFC dataset (Baly et al., 2018) that has 1,066 news media manually annotated for factuality (3-pt scale: *high*, *mixed*, *low*) and political bias (7-pt scale: from *extreme-left* to *extreme-right*). This dataset was annotated by volunteers using a detailed methodology[3] that is designed to guarantee annotation objectivity.

---

[1] `https://github.com/ramybaly/News-Media-Reliability`

[2] `https://www.alexa.com/siteinfo`

[3] For details, see `https://mediabiasfactcheck.com/methodology/`

Name	URL	Bias	Factuality	Twitter Handle	Wikipedia page
London Web News	`londonwebnews.com`	Extreme Left	Low	@londonwebnews	N/A
Daily Mirror	`www.mirror.co.uk`	Left	Mixed	@DailyMirror	`~/Daily_Mirror`
NBC News	`www.nbcnews.com`	Center-Left	High	@nbcnews	`~/NBC_News`
Associated Press	`apnews.com`	Center	Very High	@apnews	`~/Associated_Press`
Gulf News	`gulfnews.com`	Center-Right	High	@gulf_news	`~/Gulf_News`
Russia Insider	`russia-insider.com`	Right	Mixed	@russiainsider	`~/Russia_Insider`
Breitbart	`www.breitbart.com`	Extreme Right	Low	@BreitbartNews	`~/Breitbart_News`

Table 1: Examples of media and their labels for bias and factuality of reporting derived from MBFC.

Furthermore, readers can provide their own feedback on existing annotations, and in case of a large discrepancy, annotation is adjusted after a thorough review. Therefore, we believe the annotation quality is good enough to experiment with. We noticed that 117 media had *low* factuality because they publish *satire* and *pseudo-science*, neither of which has a political perspective. Since we are interested in modeling the relation between factuality and bias, we excluded those websites, thus ending up with 949 news media. Some examples from this dataset are shown in Table 1 with both factuality and bias labels, in addition to their corresponding Twitter handles and Wikipedia pages. Overall, 64% of the media in our dataset have Wikipedia pages, and 65% have Twitter accounts. Table 2 further provides detailed statistics about the label distribution in the MBFC dataset.

Factuality		Bias	
Low	198	Extreme-Left	23
Mixed	282	Left	151
High	469	Center-Left	200
		Center	139
		Center-Right	105
		Right	164
		Extreme-Right	167

Table 2: Label distribution (counts) in the MBFC dataset, which we used in our experiments.

Experimental Setup We used the implementation[4] of the Copula Ordinal Regression (COR) model as described in (Walecki et al., 2016). In our experiments, we used 5-fold cross-validation, where for each fold we split the training dataset into a training part and a validation part, and we used the latter to fine-tune the model's hyper-parameters, optimizing for Mean Absolute Error (MAE). MAE is an appropriate evaluation measure given the ordinal nature of the tasks. These hyper-parameters include the copula function (*Gumbel* vs. *Frank*), the marginal distribution (*normal* vs. *sigmoid*), the number of training iterations, the optimizer (*gradient descent*, *BFGS*), and the connection density of the CRFs. We report both MAE and MAE^M, which is a variant of MAE that is more robust to class imbalance. See (Baccianella et al., 2009; Rosenthal et al., 2017b) for more details about MAE^M vs. MAE. We compare the results to two baselines: (*i*) majority class, and (*ii*) single-task ordinal regression.

Results and Discussion Table 3 shows the evaluation results for the COR model when trained to jointly model the main task (*shown in the columns*) using combinations of auxiliary tasks (*shown in the rows*). We can see that the single-task ordinal regression model performs much better than the majority class baseline based on both evaluation measures. We can further see that the performance on the main task improves when jointly modeling several auxiliary tasks. This improvement depends on the auxiliary tasks in use.

For factuality prediction, it turns out that the combination of *bias-center+bias-extreme* yields the best overall MAE of 0.481. This makes sense and aligns well with the intuition that knowing whether a medium is centric or hyper-partisan is important to predict the factuality of its reporting. For instance, a news medium without a political ideology tends to be more trustworthy compared to an extremely biased one, regardless of their polarity (left or right), as we should expect based on the data distribution shown in Figure 1 above.

For bias prediction (at a 7-point left-to-right scale), a joint model that uses political bias at different levels of granularity (5-point and 3-point) as auxiliary tasks yields the best overall MAE of 1.479. This means that jointly modeling bias with the same information at coarser levels of granularity, i.e., adding 3-point and 5-point as auxiliary tasks, reduces the number of gross mistakes.

---

[4] `https://github.com/RWalecki/copula_ordinal_regression`

	Factuality		Bias	
Auxiliary Tasks	**MAE**	**MAE^M**	**MAE**	**MAE^M**
(None) majority class	0.714	1.000	1.798	1.857
(None) single-task COR	0.514	0.567	1.582	1.728
+bias	0.526	0.566	–	–
+factuality	–	–	1.584	1.695
+bias5-way	0.495	0.541	1.504 *(1.485)*	1.627 *(1.647)*
+bias3-way	0.497	0.548	1.528 *(1.498)*	1.658 *(1.654)*
+bias-center	0.509	0.561	1.594 *(1.535)*	1.745 *(1.695)*
+bias-extreme	0.498	0.550	1.584 *(1.558)*	1.743 *(1.726)*
+bias5-way+bias3-way	0.493	0.541	1.479 *(**1.475**)*	1.637 *(**1.623**)*
+bias-center+bias-extreme	**0.481**	**0.529**	1.563 *(1.526)*	1.714 *(1.672)*
+bias5-way+bias3-way+bias-center+bias-extreme	0.485	0.537	1.513 *(1.504)*	1.665 *(1.677)*

Table 3: Evaluating the copula ordinal regression model trained to jointly model the main task (*shown in the columns*) and different auxiliary tasks (*shown in the rows*). The results in parentheses correspond to the case when factuality is added as an additional auxiliary task (only applicable when the main task is bias prediction).

E.g., predicting *extreme-left* instead of *extreme-right*, since the model is encouraged by the auxiliary tasks to learn the correct polarity, regardless of its intensity. We can see that *factuality* is not very useful as an auxiliary task by itself (MAE=1.584 and MAE^M=1.695). In other words, a medium with low factuality could be extremely biased to either the right or to the left. Therefore, relying on *factuality* alone to predict bias might introduce severe errors, e.g., confusing extreme-left with extreme-right, thus leading to higher MAE scores. This can be remedied by adding *factuality* to the mix of other auxiliary tasks to model the main task (7-point bias prediction). The results of these experiments, shown in parentheses in Table 3, indicate that adding *factuality* to any combination of auxiliary tasks consistently yields lower MAE scores. In particular, modeling the combination of *factuality+bias5-way+bias3-way* yields the best results (MAE=1.475 and MAE^M=1.623). This result indicates that *factuality* provides complementary information that can help predict bias.

We ran a two-tailed t-test for statistical significance, which is suitable for an evaluation measure such as MAE, to confirm the improvements that were introduced by the multi-task setup. We found that the best models (shown in bold in Table 3) outperformed both the corresponding majority class baselines with a p-value ≤ 0.001, and the corresponding single-task ordinal regression baselines with a p-value ≤ 0.02.

Finally, we compared the above results to our previous work (Baly et al., 2018) by independently training a Support Vector Machine (SVM) classifier for each task, using the same features. The resulting MAE was 0.450 for factuality and 1.184 for bias prediction, which is slightly better then our results (yet, very comparable for factuality). However, our goal here is to emphasize the advantages of modeling the two tasks jointly.

5 Conclusion and Future Work

We have presented a multi-task ordinal regression framework for jointly predicting trustworthiness and political ideology of news media sources, using several auxiliary tasks, e.g., based on a coarser-grained scales or modeling extreme partisanship. Overall, we have observed sizable performance gains in terms of reduced MAE by the multi-task ordinal regression models over single-task models for each of the two individual tasks.

In future work, we want to try more auxiliary tasks, and to experiment with other languages. We further plan to go beyond *left vs. right*, which is not universal and can exhibit regional specificity (Tavits and Letki, 2009), and to model other kinds of biases, e.g., *eurosceptic vs. europhile*, *nationalist* vs. *globalist*, *islamist vs. secular*, etc.

Acknowledgments

This research is part of the Tanbih project,[5] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Qatar Computing Research Institute (QCRI), HBKU.

---

[5] `http://tanbih.qcri.org/`

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Evaluation measures for ordinal regression. In *Proceedings of the 9th IEEE International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 283–287, Pisa, Italy.

Ricardo Baeza-Yates. 2018. Bias on the web. *Commun. ACM*, 61(6):54–61.

Georgios Balikas, Simon Moura, and Massih-Reza Amini. 2017. Multitask learning for fine-grained Twitter sentiment analysis. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1005–1008, Tokyo, Japan.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3528–3539, Brussels, Belgium.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019a. Proppy: A system to unmask propaganda in online news. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, Honolulu, HI, USA.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019b. Proppy: Organizing news coverage on the basis of their propagandistic content. *Information Processing and Management*.

Ann M Brill. 2001. Online journalists embrace new marketing function. *Newspaper Research Journal*, 22(2):28.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the Internet Water Army: detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 116–120, Niagara, Canada.

Kareem Darwish, Dimitar Alexandrov, Preslav Nakov, and Yelena Mejova. 2017. Seminar users in the Arabic Twitter sphere. In *Proceedings of the 9th International Conference on Social Informatics*, SocInfo '17, pages 91–108, Oxford, UK.

Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. 2018. Attending sentences to detect satirical fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3371–3380, Santa Fe, NM, USA.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949.

Howard Finberg, Martha L Stone, and Diane Lynch. 2002. Digital journalism credibility study. *Online News Association. Retrieved November*, 3:2003.

Sean M. Gerrish and David M. Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML '11, pages 489–496, Bellevue, Washington, USA.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '16, pages 172–180, Varna, Bulgaria.

Yunchao He, Liang-Chih Yu, Chin-Sheng Yang, K Robert Lai, and Weiyi Liu. 2016. YZU-NLP team at semeval-2016 task 4: Ordinal sentiment classification using a recurrent convolutional network. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 251–255, San Diego, CA, USA.

Benjamin Horne, Sara Khedr, and Sibel Adali. 2018a. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Proceedings of the Twelfth International Conference on Web and Social Media*, ICWSM '18, pages 518–527, Stanford, CA, USA.

Benjamin D. Horne, William Dron, Sara Khedr, and Sibel Adali. 2018b. Assessing the news landscape: A multi-module toolkit for evaluating the credibility of news. In *Proceedings of the The Web Conference*, WWW '18, pages 235–238, Lyon, France.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1113–1122, Baltimore, MD, USA.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017. We built a fake news & click-bait filter: What happened next will blow your mind! In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 334–343, Varna, Bulgaria.

Vivek Kulkarni, Junting Ye, Steven Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 3518–3527, Brussels, Belgium.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2017. Detection of sockpuppets in social media. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 243–246, Portland, OR, USA.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*, 28(5):1292–1312.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 399–405, Berlin, Germany.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, LA, USA.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 353–362, Melbourne, Australia.

An T. Nguyen, Aditya Kharosekar, Matthew Lease, and Byron C. Wallace. 2018. An interpretable joint graphical model for fact-checking from crowds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI '18, New Orleans, LA, USA.

Jeff Z. Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. 2018. Content based fake news detection using knowledge graphs. In *Proceedings of the International Semantic Web Conference*, ISWC '18, Monterey, CA, USA.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, COLING '18, pages 3391–3401, Santa Fe, NM, USA.

Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On measuring bias in online information. *SIGMOD Rec.*, 46(4):16–21.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2173–2178, Indianapolis, IN, USA.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17, pages 1003–1012, Perth, Australia.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2018. CredEye: A credibility lens for analyzing and explaining misinformation. In *Proceedings of The Web Conference 2018*, WWW '18, pages 155–158, Lyon, France.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 231–240, Melbourne, Australia.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: Political ideology prediction of Twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '17, pages 729–740, Vancouver, Canada.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017a. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 502–518, Vancouver, Canada.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017b. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 502–518, Vancouver, Canada.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 91–101, Seattle, WA, USA.

Margit Tavits and Natalia Letki. 2009. When left is right: Party ideology and policy in Post-Communist Europe. *The American Political Science Review*, 103(4):555–569.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Robert Walecki, Ognjen Rudovic, Vladimir Pavlovic, and Maja Pantic. 2016. Copula ordinal regression for joint estimation of facial action unit intensity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4902–4910.

Shipeng Yu, Kai Yu, Volker Tresp, and Hans-Peter Kriegel. 2006. Collaborative ordinal regression. In *Proceedings of the 23rd international conference on Machine learning*, pages 1089–1096. ACM.

Joint Detection and Location of English Puns

Yanyan Zou and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
`yanyan_zou@mymail.sutd.edu.sg, luwei@sutd.edu.sg`

Abstract

A pun is a form of wordplay for an intended humorous or rhetorical effect, where a word suggests two or more meanings by exploiting polysemy (*homographic pun*) or phonological similarity to another word (*heterographic pun*). This paper presents an approach that addresses pun detection and pun location jointly from a sequence labeling perspective. We employ a new tagging scheme such that the model is capable of performing such a joint task, where useful structural information can be properly captured. We show that our proposed model is effective in handling both homographic and heterographic puns. Empirical results on the benchmark datasets demonstrate that our approach can achieve new state-of-the-art results.

1 Introduction

There exists a class of language construction known as *pun* in natural language texts and utterances, where a certain word or other lexical items are used to exploit two or more separate meanings. It has been shown that understanding of puns is an important research question with various real-world applications, such as human-computer interaction (Morkes et al., 1999; Hempelmann, 2008) and machine translation (Schröter, 2005). Recently, many researchers show their interests in studying puns, like detecting pun sentences (Vadehra, 2017), locating puns in the text (Cai et al., 2018), interpreting pun sentences (Sevgili et al., 2017) and generating sentences containing puns (Ritchie, 2005; Hong and Ong, 2009; Yu et al., 2018). A pun is a wordplay in which a certain word suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another sign, for an intended humorous or rhetorical effect. Puns can be generally categorized into two groups, namely heterographic puns (where the pun and its latent target are phonologically similar) and homographic puns (where the two meanings of the pun reflect its two distinct senses) (Miller et al., 2017). Consider the following two examples:

(1) When the church bought gas for their annual barbecue, proceeds went from the sacred to the *propane*.

(2) Some diets cause a *gut* reaction.

The first punning joke exploits the sound similarity between the word "*propane*" and the latent target "profane", which can be categorized into the group of heterographic puns. Another categorization of English puns is homographic pun, exemplified by the second instance leveraging distinct senses of the word "*gut*".

Pun detection is the task of detecting whether there is a pun residing in the given text. The goal of pun location is to find the exact word appearing in the text that implies more than one meanings. Most previous work addresses such two tasks separately and develop separate systems (Pramanick and Das, 2017; Sevgili et al., 2017). Typically, a system for pun detection is built to make a binary prediction on whether a sentence contains a pun or not, where all instances (with or without puns) are taken into account during training. For the task of pun location, a separate system is used to make a single prediction as to which word in the given sentence in the text that trigger more than one semantic interpretations of the text, where the training data involves only sentences that contain a pun. Therefore, if one is interested in solving both problems at the same time, a pipeline approach that performs pun detection followed by pun location can be used.

Compared to the pipeline methods, joint learning has been shown effective (Katiyar and Cardie, 2016; Peng et al., 2018) since it is able to re-

Proceedings of NAACL-HLT 2019, pages 2117–2123
Minneapolis, Minnesota, June 2 - June 7, 2019.

duce error propagation and allows information exchange between tasks which is potentially beneficial to all the tasks. In this work, we demonstrate that the detection and location of puns can be jointly addressed by a single model. The pun detection and location tasks can be combined as a sequence labeling problem, which allows us to jointly detect and locate a pun in a sentence by assigning each word a tag. Since each context contains a maximum of one pun (Miller et al., 2017), we design a novel tagging scheme to capture this structural constraint. Statistics on the corpora also show that a pun tends to appear in the second half of a context. To capture such a structural property, we also incorporate word position knowledge into our structured prediction model. Experiments on the benchmark datasets show that detection and location tasks can reinforce each other, leading to new state-of-the-art performance on these two tasks. To the best of our knowledge, this is the first work that performs joint detection and location of English puns by using a sequence labeling approach.[1]

2 Approach

2.1 Problem Definition

We first design a simple tagging scheme consisting of two tags $\{\mathsf{N}, \mathsf{P}\}$:

- N tag means the current word is not a pun.
- P tag means the current word is a pun.

If the tag sequence of a sentence contains a P tag, then the text contains a pun and the word corresponding to P is the pun.

The contexts have the characteristic that each context contains a maximum of one pun (Miller et al., 2017). In other words, there exists only one pun if the given sentence is detected as the one containing a pun. Otherwise, there is no pun residing in the text. To capture this interesting property, we propose a new tagging scheme consisting of three tags, namely $\{\mathsf{B}, \mathsf{P}, \mathsf{A}\}$.

- B tag indicates that the current word appears before the pun in the given context.
- P tag highlights the current word is a pun.
- A tag indicates that the current word appears after the pun.

We empirically show that the BPA scheme can guarantee the context property that there exists a maximum of one pun residing in the text.

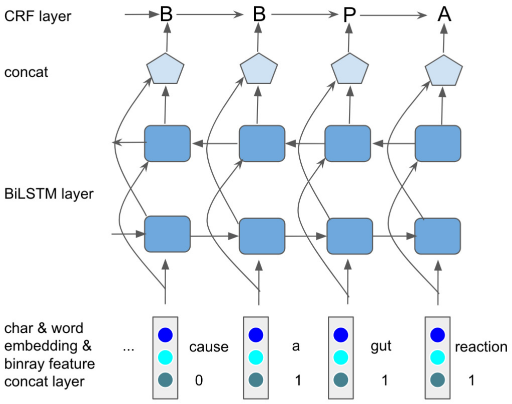


Figure 1: Model architecture

Given a context from the training set, we will be able to generate its corresponding gold tag sequence using a deterministic procedure. Under the two schemes, if a sentence does not contain any puns, all words will be tagged with N or B, respectively. Exemplified by the second sentence "*Some diets cause a gut reaction*," the pun is given as "*gut*." Thus, under the BPA scheme, it should be tagged with P, while the words before it are assigned with the tag B and words after it are with A, as illustrated in Figure 1. Likewise, the NP scheme tags the word "*gut*" with P, while other words are tagged with N. Therefore, we can combine the pun detection and location tasks into one problem which can be solved by the sequence labeling approach.

2.2 Model

Neural models have shown their effectiveness on sequence labeling tasks (Chiu and Nichols, 2016; Ma and Hovy, 2016; Liu et al., 2018). In this work, we adopt the bidirectional Long Short Term Memory (BiLSTM) (Graves and Schmidhuber, 2005) networks on top of the Conditional Random Fields (Lafferty et al., 2001) (CRF) architecture to make labeling decisions, which is one of the classical models for sequence labeling. Our model architecture is illustrated in Figure 1 with a running example. Given a context/sentence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where n is the length of the context, we generate the corresponding tag sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ based on our designed tagging schemes and the original annotations for pun detection and location provided by the corpora. Our model is then trained on pairs of $(\mathbf{x}, \mathbf{y})$.

Input. The contexts in the pun corpus hold the property that each pun contains exactly one content word, which can be either a noun, a verb, an

---

[1] Our code is publicly available at `https://github.com/zoezou2015/PunLocation`.

adjective, or an adverb. To capture this characteristic, we consider lexical features at the character level. Similar to the work of (Liu et al., 2018), the character embeddings are trained by the character-level LSTM networks on the unannotated input sequences. Nonlinear transformations are then applied to the character embeddings by highway networks (Srivastava et al., 2015), which map the character-level features into different semantic spaces.

We also observe that a pun tends to appear at the end of a sentence. Specifically, based on the statistics, we found that sentences with a pun that locate at the second half of the text account for around 88% and 92% in homographic and heterographic datasets, respectively. We thus introduce a binary feature that indicates if a word is located at the first or the second half of an input sentence to capture such positional information. A binary indicator can be mapped to a vector representation using a randomly initialized embedding table (He et al., 2017; Wang and Lu, 2018). In this work, we directly adopt the value of the binary indicator as part of the input.

The concatenation of the transformed character embeddings, the pre-trained word embeddings (Pennington et al., 2014), and the position indicators are taken as input of our model[2].

Tagging. The input is then fed into a BiLSTM network, which will be able to capture contextual information. For a training instance $(\mathbf{x}, \mathbf{y})$, we suppose the output by the word-level BiLSTM is $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$. The CRF layer is adopted to capture label dependencies and make final tagging decisions at each position, which has been included in many state-of-the-art sequence labeling models (Ma and Hovy, 2016; Liu et al., 2018). The conditional probability is defined as:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\prod_{i=1}^{n} \exp\left(W_{y_{i-1},y_i}\mathbf{z}_i + b_{y_{i-1},y_i}\right)}{\sum_{\mathbf{y}' \in \mathbf{Y}} \prod_{i=1}^{n} \exp\left(W_{y'_{i-1},y'_i}\mathbf{z}_i + b_{y'_{i-1},y'_i}\right)}$$

where $\mathbf{Y}$ is a set of all possible label sequences consisting of tags from $\{\mathsf{N}, \mathsf{P}\}$ (or $\{\mathsf{B}, \mathsf{P}, \mathsf{A}\}$), W_{y_{i-1},y_i} and b_{y_{i-1},y_i} are weight and bias parameters corresponding to the label pair (y_{i-1}, y_i). During training, we minimize the negative log-likelihood summed over all training instances:

$$\mathcal{L} = -\sum_i \log P(\mathbf{y}_i|\mathbf{x}_i)$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ refers to the i-th instance in the training set. During testing, we aim to find the optimal label sequence for a new input $\mathbf{x}$:

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathbf{Y}} P(\mathbf{y}|\mathbf{x})$$

This search process can be done efficiently using the Viterbi algorithm.

3 Experiments

3.1 Datasets and Settings

We evaluate our model on two benchmark datasets (Miller et al., 2017). The homographic dataset contains 2,250 contexts, 1,607 of which contain a pun. The heterographic dataset consists of 1,780 contexts with 1,271 containing a pun. We notice there is no standard splitting information provided for both datasets. Thus we apply 10-fold cross validation. To make direct comparisons with prior studies, following (Cai et al., 2018), we accumulated the predictions for all ten folds and calculate the scores in the end.

For each fold, we randomly select 10% of the instances from the training set for development. Word embeddings are initialized with the 100-dimensional Glove (Pennington et al., 2014). The dimension of character embeddings is 30 and they are randomly initialized, which can be fine tuned during training. The pre-trained word embeddings are not updated during training. The dimensions of hidden vectors for both char-level and word-level LSTM units are set to 300. We adopt stochastic gradient descent (SGD) (Bottou, 1991) with a learning rate of 0.015.

For the pun detection task, if the predicted tag sequence contains at least one P tag, we regard the output (i.e., the prediction of our pun detection model) for this task as true, otherwise false. For the pun location task, a predicted pun is regarded as correct if and only if it is labeled as the gold pun in the dataset. As to pun location, to make fair comparisons with prior studies, we only consider the instances that are labeled as the ones containing a pun. We report precision, recall and F_1 score in Table 1. A list of prior works that did not employ joint learning are also shown in the first block of Table 1.

3.2 Results

We also implemented a baseline model based on conditional random fields (CRF), where features like POS tags produced by the Stanford POS tagger (Toutanova et al., 2003), n-grams, label tran-

---

[2] The word sense has also been shown helpful for the location of a homographic pun (Cai et al., 2018). However, such information may not always be helpful for the location of heterographic puns. We thus exclude such knowledge.

System	Homographic						Heterographic					
	Detection			Location			Detection			Location		
	P.	*R.*	F_1	*P.*	*R.*	F_1	*P.*	*R.*	F_1	*P.*	*R.*	F_1
Pedersen (2017)	78.32	87.24	82.54	44.00	44.00	44.00	73.99	86.62	68.71	-	-	-
Pramanick and Das (2017)	72.51	90.79	68.84	33.48	33.48	33.48	73.67	94.02	71.74	37.92	37.92	37.92
Mikhalkova and Karyakin (2017)	79.93	73.37	67.82	32.79	32.79	32.79	75.80	59.40	57.47	35.01	35.01	35.01
Vadehra (2017)	68.38	47.23	46.71	34.10	34.10	34.10	65.23	41.78	42.53	42.80	42.80	42.80
Indurthi and Oota (2017)	90.24	89.70	85.33	52.15	52.15	52.15	-	-	-	-	-	-
Vechtomova (2017)	-	-	-	65.26	65.21	65.23	-	-	-	79.73	**79.54**	**79.64**
Cai et al. (2018)	-	-	-	81.50	74.70	78.00	-	-	-	-	-	-
CRF	87.21	64.09	73.89	**86.31**	55.32	67.43	**89.56**	70.94	79.17	**88.46**	62.76	73.42
Ours – NP	89.19	86.25	87.69	82.11	70.82	76.04	85.33	90.64	87.91	79.17	71.76	75.28
Ours – BPA	89.24	92.28	91.04	83.55	**77.10**	**80.19**	84.62	**95.20**	89.60	81.41	77.50	79.40
Ours – BPA-p	**91.25**	**93.28**	**92.19**	82.06	76.54	79.20	86.67	93.08	**89.76**	80.81	75.22	77.91
Pipeline	-	-	-	67.70	67.70	67.70	-	-	-	68.84	68.84	68.84

Table 1: Comparison results on two benchmark datasets. (*P.*: Precision, *R.*: Recall, F_1: F_1 score.)

sitions, word suffixes and relative position to the end of the text are considered. We can see that our model with the BPA tagging scheme yields new state-of-the-art F_1 scores on pun detection and competitive results on pun location, compared to baselines that do not adopt joint learning in the first block. For location on heterographic puns, our model's performance is slightly lower than the system of (Vechtomova, 2017), which is a rule-based locator. Compared to CRF, we can see that our model, either with the NP or the BPA scheme, yields significantly higher recall on both detection and location tasks, while the precisions are relatively close. This demonstrates the effectiveness of BiLSTM, which learns the contextual features of given texts – such information appears to be helpful in recalling more puns.

Compared to the NP scheme, the BPA tagging scheme is able to yield better performance on these two tasks. After studying outputs from these two approaches, we found that one leading source of error for the NP approach is that there exist more than one words in a single instance that are assigned with the P tag. However, according to the description of pun in (Miller et al., 2017), each context contains a maximum of one pun. Thus, such a useful structural constraint is not well captured by the simple approach based on the NP tagging scheme. On the other hand, by applying the BPA tagging scheme, such a constraint is properly captured in the model. As a result, the results for such a approach are significantly better than the approach based on the NP tagging scheme, as we can observe from the table. Under the same experimental setup, we also attempted to exclude word position features. Results are given by BPA-p. It is expected that the performance of pun location drops, since such position features are able to capture the interesting property that a pun tends to appear in the second half of a sentence. While such knowledge is helpful for the location task, interestingly, a model without position knowledge yields improved performance on the pun detection task. One possible reason is that detecting whether a sentence contains a pun is not concerned with such word position information.

Additionally, we conduct experiments over sentences containing a pun only, namely 1,607 and 1,271 instances from homographic and heterographic pun corpora separately. It can be regarded as a "pipeline" method where the classifier for pun detection is regarded as perfect.[3] Following the prior work of (Cai et al., 2018), we apply 10-fold cross validation. Since we are given that all input sentences contain a pun, we only report accumulated results on pun location, denoted as Pipeline in Table 1. Compared with our approaches, the performance of such an approach drops significantly. On the other hand, such a fact demonstrates that the two task, detection and location of puns, can reinforce each other. These figures demonstrate the effectiveness of our sequence labeling method to detect and locate English puns in a joint manner.

3.3 Error Analysis

We studied the outputs from our system and make some error analysis. We found the errors can be broadly categorized into several types, and we elaborate them here. 1) Low word coverage: since the corpora are relatively small, there exist many unseen words in the test set. Learning the representations of such unseen words is challeng-

---

[3] Under a pipeline setting, the first step is to detect if a sentence contains a pun. Then another algorithm is called to locate the exact pun word residing in the sentence if such a sentence is detected as the one containing a pun. In our setting, we assume the detection phase is perfect. In other words, all sentences containing a pun are exactly retrieved.

ing, which affects the model's performance. Such errors contribute around 40% of the total errors made by our system. 2) Detection errors: we found many errors are due to the model's inability to make correct pun detection. Such inability harms both pun detection and pun location. Although our approach based on the BPA tagging scheme yields relatively higher scores on the detection task, we still found that 40% of the incorrectly predicted instances fall into this group. 3) Short sentences: we found it was challenging for our model to make correct predictions when the given text is short. Consider the example "*Superglue! Tom rejoined,*" here the word *rejoined* is the corresponding pun. However, it would be challenging to figure out the pun with such limited contextual information.

4 Related Work

Most existing systems address pun detection and location separately. Pedersen (2017) applied word sense knowledge to conduct pun detection. Indurthi and Oota (2017) trained a bidirectional RNN classifier for detecting homographic puns. Next, a knowledge-based approach is adopted to find the exact pun. Such a system is not applicable to heterographic puns. Doogan et al. (2017) applied Google n-gram and word2vec to make decisions. The phonetic distance via the CMU Pronouncing Dictionary is computed to detect heterographic puns. Pramanick and Das (2017) used the hidden Markov model and a cyclic dependency network with rich features to detect and locate puns. Mikhalkova and Karyakin (2017) used a supervised approach to pun detection and a weakly supervised approach to pun location based on the position within the context and part of speech features. Vechtomova (2017) proposed a rule-based system for pun location that scores candidate words according to eleven simple heuristics. Two systems are developed to conduct detection and location separately in the system known as UWAV (Vadehra, 2017). The pun detector combines predictions from three classifiers. The pun locator considers word2vec similarity between every pair of words in the context and position to pinpoint the pun. The state-of-the-art system for homographic pun location is a neural method (Cai et al., 2018), where the word senses are incorporated into a bidirectional LSTM model. This method only supports the pun location task on homographic puns. Another line of research efforts related to this work is sequence labeling, such as POS tagging, chunking, word segmentation and NER. The neural methods have shown their effectiveness in this task, such as BiLSTM-CNN (Chiu and Nichols, 2016), GRNN (Xu and Sun, 2016), LSTM-CRF (Lample et al., 2016), LSTM-CNN-CRF (Ma and Hovy, 2016), LM-LSTM-CRF (Liu et al., 2018).

In this work, we combine pun detection and location tasks as a single sequence labeling problem. Inspired by the work of (Liu et al., 2018), we also adopt a LSTM-CRF with character embeddings to make labeling decisions.

5 Conclusion

In this paper, we propose to perform pun detection and location tasks in a joint manner from a sequence labeling perspective. We observe that each text in our corpora contains a maximum of one pun. Hence, we design a novel tagging scheme to incorporate such a constraint. Such a scheme guarantees that there is a maximum of one word that will be tagged as a pun during the testing phase. We also found the interesting structural property such as the fact that most puns tend to appear at the second half of the sentences can be helpful for such a task, but was not explored in previous works. Furthermore, unlike many previous approaches, our approach, though simple, is generally applicable to both heterographic and homographic puns. Empirical results on the benchmark datasets prove the effectiveness of the proposed approach that the two tasks of pun detection and location can be addressed by a single model from a sequence labeling perspective.

Future research includes the investigations on how to make use of richer semantic and linguistic information for detection and location of puns. Research on puns for other languages such as Chinese is still under-explored, which could also be an interesting direction for our future studies.

Acknowledgments

We would like to thank the three anonymous reviewers for their thoughtful and constructive comments. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156, and is partially supported by SUTD project PIE-SGP-AI-2018-01.

References

Léon Bottou. 1991. Stochastic gradient learning in neural networks. In *Proc. of Neuro-Nimes*.

Yitao Cai, Yin Li, and Xiaojun Wan. 2018. Sense-aware neural models for pun location in texts. In *Proceedings of ACL*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Proceedings of TACL*.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, volume 4, pages 2047–2052.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of ACL*.

Christian F Hempelmann. 2008. Computational humor: Beyond the pun? *The Primer of Humor Research. Humor Research*, 8:333–360.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at semeval-2017 task 7: Detection and interpretation of homographic puns in english language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank F Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI*.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL*.

Elena Mikhalkova and Yuri Karyakin. 2017. Punfields at semeval-2017 task 7: Employing roget's thesaurus in automatic pun recognition and interpretation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

John Morkes, Hadyn K Kernal, and Clifford Nass. 1999. Effects of humor in task-oriented human-computer interaction and computer-mediated communication: A direct test of srct theory. *Human-Computer Interaction*, 14(4):395–435.

Ted Pedersen. 2017. Duluth at semeval-2017 task 7: Puns upon a midnight dreary, lexical semantics for the weak and weary. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Aniket Pramanick and Dipankar Das. 2017. Ju cse nlp @ semeval 2017 task 7: Employing rules to detect and interpret english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Graeme Ritchie. 2005. Computational mechanisms for pun generation. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

Thorsten Schröter. 2005. *Shun the Pun, Rescue the Rhyme?: The Dubbing and Subtitling of Language Play in Film*. Ph.D. thesis, Estetisk-filosofiska fakulteten.

Özge Sevgili, Nima Ghotbi, and Selma Tekir. 2017. N-hance at semeval-2017 task 7: A computational approach using word association for puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL-HLT*.

Ankit Vadehra. 2017. Uwav at semeval-2017 task 7: Automated feature-based system for locating puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 7: Locating the pun using syntactic characteristics and corpus-based metrics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *Proceedings of AAAI*.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of ACL*.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of ACL*.

Harry Potter and the Action Prediction Challenge from Natural Language

David Vilares
Universidade da Coruña, CITIC
Departamento de Computación
Campus de Elviña s/n, 15071
A Coruña, Spain
`david.vilares@udc.es`

Carlos Gómez-Rodríguez
Universidade da Coruña, CITIC
Departamento de Computación
Campus de Elviña s/n, 15071
A Coruña, Spain
`carlos.gomez@udc.es`

Abstract

We explore the challenge of action prediction from textual descriptions of scenes, a testbed to approximate whether text inference can be used to predict upcoming actions. As a case of study, we consider the world of the Harry Potter fantasy novels and inferring what spell will be cast next given a fragment of a story. Spells act as keywords that abstract actions (e.g. 'Alohomora' to open a door) and denote a response to the environment. This idea is used to automatically build HPAC, a corpus containing 82 836 samples and 85 actions. We then evaluate different baselines. Among the tested models, an LSTM-based approach obtains the best performance for frequent actions and large scene descriptions, but approaches such as logistic regression behave well on infrequent actions.

1 Introduction

Natural language processing (NLP) has achieved significant advances in reading comprehension tasks (Chen et al., 2016; Salant and Berant, 2017). These are partially due to embedding methods (Mikolov et al., 2013; Devlin et al., 2018) and neural networks (Rosenblatt, 1958; Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017), but also to the availability of new resources and challenges. For instance, in cloze-form tasks (Hermann et al., 2015; Bajgar et al., 2016), the goal is to predict the missing word given a short context. Weston et al. (2015) presented baBI, a set of proxy tasks for reading comprenhension. In the SQuAD corpus (Rajpurkar et al., 2016), the aim is to answer questions given a Wikipedia passage. Kocisky et al. (2018) introduce NarrativeQA, where answering the questions requires to process entire stories. In a related line, Fremann et al. (2017) use fictional crime scene investigation data, from the CSI series, to define a task where the models try to answer the question: 'who committed the crime?'.

In an alternative line of work, script induction (Schank and Abelson, 1977) has been also a useful approach to evaluate inference and semantic capabilities of NLP systems. Here, a model processes a document to infer new sequences that reflect events that are statistically probable (e.g. go to a restaurant, be seated, check the menu, ...). For example, Chambers and Jurafsky (2008) introduce narrative event chains, a representation of structured knowledge of a set of events occurring around a protagonist. They then propose a method to learn statistical scripts, and also introduce two different evaluation strategies. With a related aim, Pichotta and Mooney (2014) propose a multi-event representation of statistical scripts to be able to consider multiple entities. These same authors (Pichotta and Mooney, 2016) have also studied the abilities of recurrent neural networks for learning scripts, generating upcoming events given a raw sequence of tokens, using BLEU (Papineni et al., 2002) for evaluation.

This paper explores instead a new task: action prediction from natural language descriptions of scenes. The challenge is addressed as follows: given a natural language input sequence describing the scene, such as a piece of a story coming from a transcript, the goal is to infer which action is most likely to happen next.

Contribution We introduce a fictional-domain English corpus set in the world of Harry Potter novels. The domain is motivated by the existence of a variety of spells in these literary books, associated with keywords that can be seen as unambiguous markers for actions that potentially relate to the previous context. This is used to automatically create a natural language corpus coming from hundreds of users, with different styles, interests and writing skills. We then train a number of standard baselines to predict upcoming actions, a task that

Proceedings of NAACL-HLT 2019, pages 2124–2130
Minneapolis, Minnesota, June 2 - June 7, 2019.

requires to be aware of the context. In particular, we test a number of generic models, from a simple logistic regression to neural models. Experiments shed some light about their strengths and weaknesses and how these are related to the frequency of each action, the existence of other semantically related actions and the length of the input story.

2 HPAC: The Harry Potter's Action prediction Corpus

To build an action prediction corpus, we need to: (1) consider the set of actions, and (2) collect data where these occur. Data should come from different users, to approximate a real natural language task. Also, it needs to be annotated, determining that a piece of text ends up triggering an action. These tasks are however time consuming, as they require annotators to read vast amounts of large texts. In this context, machine comprehension resources usually establish a compromise between their complexity and the costs of building them (Hermann et al., 2015; Kocisky et al., 2018).

2.1 Domain motivation

We rely on an intuitive idea that uses transcripts from the Harry Potter world to build up a corpus for textual action prediction. The domain has a set of desirable properties to evaluate reading comprehension systems, which we now review.

Harry Potter novels define a variety of spells. These are keywords cast by witches and wizards to achieve purposes, such as turning on a light ('Lumos'), unlocking a door ('Alohomora') or killing ('Avada Kedavra'). They abstract complex and non-ambiguous actions. Their use also makes it possible to build an automatic and self-annotated corpus for action prediction. The moment a spell occurs in a text represents a response to the environment, and hence, it can be used to label the preceding text fragment as a scene description that ends up triggering that action. Table 1 illustrates it with some examples from the original books.

This makes it possible to consider texts from the magic world of Harry Potter as the domain for the action prediction corpus, and the spells as the set of eligible actions.[1] Determining the length of the preceding context, namely *snippet*, that has to be considered as the scene description is however not trivial. This paper considers experiments (§4) using snippets with the 32, 64, 96 and 128 previous tokens to an action. We provide the needed scripts to rebuild the corpus using arbitrary lengths.[2]

2.2 Data crawling

The number of occurrences of spells in the original Harry Potter books is small (432 occurrences), which makes it difficult to train and test a machine learning model. However, the amount of available fan fiction for this saga allows to create a large corpus. For HPAC, we used fan fiction (and *only* fan fiction texts) from `https://www.fanfiction.net/book/Harry-Potter/` and a version of the crawler by Milli and Bamman (2016).[3] We collected Harry Potter stories written in English and marked with the status 'completed'. From these we extracted a total of 82 836 spell occurrences, that we used to obtain the scene descriptions. Table 2 details the statistics of the corpus (see also Appendix A). Note that similar to Twitter corpora, fan fiction stories can be deleted over time by users or admins, causing losses in the dataset.[4]

Preprocessing We tokenized the samples with (Manning et al., 2014) and merged the occurrences of multi-word spells into a single token.

3 Models

This work addresses the task as a classification problem, and in particular as a sequence to label classification problem. For this reason, we rely on standard models used for this type of task: multinomial logistic regression, a multi-layered perceptron, convolutional neural networks and long short-term memory networks. We outline the essentials of each of these models, but will treat them as black boxes. In a related line, Kaushik and Lipton (2018) discuss the need of providing rigorous baselines that help better understand the improvement coming from future and complex models, and also the need of not demanding architectural novelty when introducing new datasets.

Although not done in this work, an alternative (but also natural) way to address the task is as a

---

[1] Note that the corpus is built in an automatic way and some occurrences might not correspond to actions, but for example, to a description of the spell or even some false positive samples. Related to this, we have not censored the content of the stories, so some of them might contain adult content.

[2] `https://github.com/aghie/hpac`

[3] Due to the website's Terms of Service, the corpus cannot be directly released.

[4] They also can be modified, making it unfeasible to retrieve some of the samples.

Text fragment	**Action**
Ducking under Peeves, they ran for their lives, right to the end of the corridor where they slammed into a door - and it was locked. 'This is it!' Ron moaned, as they pushed helplessly at the door, 'We're done for! This is the end!' They could hear footsteps, Filch running as fast as he could toward Peeves's shouts. 'Oh, move over', Hermione snarled. She grabbed Harry's wand, tapped the lock, and whispered, '**Alohomora**'.	Unlock the door
And then, without warning, Harry's scar exploded with pain. It was agony such as he had never felt in all his life; his wand slipped from his fingers as he put his hands over his face; his knees buckled; he was on the ground and he could see nothing at all; his head was about to split open. From far away, above his head, he heard a high, cold voice say, 'Kill the spare.' A swishing noise and a second voice, which screeched the words to the night: '**Avada Kedavra**'	Kill a target
Harry felt himself being pushed hither and thither by people whose faces he could not see. Then he heard Ron yell with pain. 'What happened?' said Hermione anxiously, stopping so abruptly that Harry walked into her. 'Ron, where are you? Oh, this is stupid' - '**Lumos**'	Turn on a light

Table 1: Examples from the Harry Potter books showing how spells map to reactions to the environment.

Statistics	**Training**	**Dev**	**Test**
#Actions	85	83	84
#Samples	66 274	8 279	8 283
#Tokens (s=32)	2 111 180	263 573	263 937
#Unique tokens (s=32)	33 067	13 075	13 207
#Tokens (s=128)	8 329 531	1 040 705	1 041 027
#Unique tokens (s=128)	60 379	25 146	25 285

Table 2: Corpus statistics: s is the length of the snippet.

special case of language modelling, where the output vocabulary is restricted to the size of the 'action' vocabulary. Also, note that the performance for this task is not expected to achieve a perfect accuracy, as there may be situations where more than one action is reasonable, and also because writers tell a story playing with elements such as surprise or uncertainty.

The source code for the models can be found in the GitHub repository mentioned above.

Notation $w_{1:n}$ denotes a sequence of words $w_1, ..., w_n$ that represents the scene, with $w_i \in V$. $F_\theta(\cdot)$ is a function parametrized by θ. The task is cast as $F : V^n \rightarrow A$, where A is the set of actions.

3.1 Machine learning models

The input sentence $w_{1:n}$ is encoded as a one-hot vector, $\mathbf{v}$ (total occurrence weighting scheme).

Multinomial Logistic Regression Let $\text{MLR}_\theta(\mathbf{v})$ be an abstraction of a multinomial logistic regression parametrized by θ, the output for an input $\mathbf{v}$ is computed as the $\arg\max_{a \in A} P(y = a|\mathbf{v})$, where $P(y = a|\mathbf{v})$ is a $softmax$ function, i.e, $P(y = a|\mathbf{v}) = \frac{e^{W_a \cdot \mathbf{v}}}{\sum_{a'}^{A} e^{W_{a'} \cdot \mathbf{v}}}$.

MultiLayer Perceptron We use one hidden layer with a rectifier activation function ($relu(x)$=$max(0, x)$). The output is computed as $\text{MLP}_\theta(\mathbf{v})$= $softmax(W_2 \cdot relu(W \cdot \mathbf{v} + \mathbf{b}) + \mathbf{b_2})$.

3.2 Sequential models

The input sequence is represented as a sequence of word embeddings, $\mathbf{w}_{1:n}$, where $\mathbf{w}_i$ is a concatenation of an internal embedding learned during the training process for the word w_i, and a pretrained embedding extracted from GloVe (Pennington et al., 2014)[5], that is further fine-tuned.

Long short-term memory network (Hochreiter and Schmidhuber, 1997): The output for an element $\mathbf{w}_i$ also depends on the output of $\mathbf{w}_{i-1}$. The $\text{LSTM}_\theta(\mathbf{w}_{1:n})$[6] takes as input a sequence of word embeddings and produces a sequence of hidden outputs, $\mathbf{h}_{1:n}$ ($\mathbf{h}_i$ size set to 128). The last output of the LSTM_θ, $\mathbf{h}_n$, is fed to a MLP_θ.

Convolutional Neural Network (LeCun et al., 1995; Kim, 2014). It captures local properties over continuous slices of text by applying a convolution layer made of different filters. We use a wide convolution, with a window slice size of length 3 and 250 different filters. The convolutional layer uses a $relu$ as the activation function. The output is fed to a max pooling layer, whose output vector is passed again as input to a MLP_θ.

4 Experiments

Setup All MLP_θ's have 128 input neurons and 1 hidden layer. We trained up to 15 epochs using mini-batches (size=16), Adam (lr=0.001) (Kingma and Ba, 2015) and early stopping.

Table 3 shows the macro and weighted F-scores for the models considering different snippet sizes.[7]

---

[5] `http://nlp.stanford.edu/data/glove.6B.zip`

[6] n is set to be equal to the length of the snippet.

[7] As we have addressed the task as a classification problem, we will use precision, recall and F-score as the evaluation metrics.

To diminish the impact of random seeds and local minima in neural networks, results are averaged across 5 runs.[8] 'Base' is a majority-class model that maps everything to 'Avada Kedavra', the most common action in the training set. This helps test whether the models predict above chance performance. When using short snippets (size=32), disparate models such as our MLR, MLP and LSTMs achieve a similar performance. As the snippet size is increased, the LSTM-based approach shows a clear improvement on the weighted scores[9], something that happens only marginally for the rest. However, from Table 3 it is hard to find out what the approaches are actually learning to predict.

Snippet	Model	Macro			Weighted		
		P	R	F	P	R	F
-	Base	0.1	1.2	0.2	1.3	11.5	2.4
32	MLR	18.7	11.6	13.1	28.9	31.4	28.3
	MLP	19.1	9.8	10.3	**31.7**	32.1	28.0
	LSTM	13.7	9.7	9.5	29.1	32.2	28.6
	CNN	9.9	7.8	7.3	24.6	29.2	24.7
64	MLR	**20.6**	12.3	13.9	29.9	32.1	29.0
	MLP	17.9	9.5	9.8	31.2	32.7	27.9
	LSTM	13.3	10.3	10.2	30.3	33.9	30.4
	CNN	9.8	7.8	7.4	25.0	29.9	25.4
96	MLR	20.4	**13.3**	**14.6**	30.3	32.0	29.3
	MLP	16.9	9.5	9.8	30.2	32.6	27.8
	LSTM	14.0	10.5	10.3	30.6	34.5	30.7
	CNN	10.2	7.1	6.9	25.2	29.4	24.4
128	MLR	19.6	12.1	12.9	30.0	31.7	28.2
	MLP	18.9	9.9	10.3	31.4	32.9	28.0
	LSTM	14.4	10.5	10.5	31.3	**35.1**	**31.1**
	CNN	8.8	7.8	7.1	24.8	30.2	25.0

Table 3: Macro and weighted F-scores over 5 runs.

To shed some light, Table 4 shows their performance according to a ranking metric, recall at k. The results show that the LSTM-based approach is the top performing model, but the MLP obtains just slightly worse results. Recall at 1 is in both cases low, which suggests that the task is indeed complex and that using just LSTMs is not enough. It is also possible to observe that even if the models have difficulties to correctly predict the action as a first option, they develop certain sense of the scene and consider the right one among their top choices. Table 5 delves into this by splitting the performance of the model into infrequent and frequent actions (above the average, i.e. those that occur more than 98 times in the training set, a total of 20 actions). There is a clear gap between the performance on these two groups of actions, with a ∼50 points difference in recall at 5. Also, a simple logistic regression performs similar to the LSTM on the infrequent actions.

Snippet	Model	R@1	R@2	R@5	R@10
-	Base	11.5	-	-	-
32	MLR	31.4	43.7	60.3	73.5
	MLP	32.1	44.3	61.5	74.9
	LSTM	32.2	44.3	61.5	74.7
	CNN	29.2	41.1	58.1	71.6
64	MLR	32.1	44.9	61.9	74.3
	MLP	32.7	46.0	63.5	76.6
	LSTM	33.9	46.1	63.1	75.7
	CNN	29.9	41.8	59.0	72.2
96	MLR	32.0	44.5	60.7	74.6
	MLP	32.6	45.6	63.4	76.6
	LSTM	34.5	46.9	63.7	76.1
	CNN	29.3	41.9	59.5	72.8
128	MLR	31.7	44.5	61.0	74.3
	MLP	32.9	45.8	63.2	**76.9**
	LSTM	**35.1**	**47.4**	**64.4**	**76.9**
	CNN	30.2	42.3	59.6	72.8

Table 4: Averaged recall at k over 5 runs.

Snippet	Model	Frequent			Infrequent		
		F_{we}	R@1	R@5	F_{we}	R@1	R@5
	Base	3.7	14.5	-	0.0	0.0	-
32	MLR	35.8	37.1	70.5	14.8	9.5	23.0
	MLP	35.9	38.1	71.9	13.2	9.4	21.8
	LSTM	37.1	38.4	71.6	11.7	8.6	23.0
	CNN	33.1	35.5	69.3	7.1	5.2	15.2
64	MLR	36.7	37.9	71.8	14.9	9.9	24.0
	MLP	36.4	39.2	**74.5**	11.0	7.9	21.6
	LSTM	39.2	40.3	73.0	12.4	9.4	25.4
	CNN	33.9	36.4	70.6	6.9	5.2	15.1
96	MLR	36.4	37.4	70.1	**17.1**	**11.7**	25.1
	MLP	36.2	39.1	74.0	11.0	7.9	23.1
	LSTM	39.6	41.1	73.7	12.4	9.6	25.8
	CNN	32.7	35.8	71.6	6.3	4.8	13.7
128	MLR	35.4	37.2	70.5	15.4	10.7	25.0
	MLP	36.5	39.5	74.0	11.1	8.2	22.3
	LSTM	**40.3**	**41.9**	74.4	12.3	9.5	**26.2**
	CNN	33.7	36.9	71.4	6.5	5.0	14.6

Table 5: Performance on *frequent* (those that occur above the average) and *infrequent* actions.

Error analysis[10] Some of the misclassifications made by the LSTM approach were semantically related actions and counter-actions. For example, 'Colloportus' (to close a door) was never predicted. The most common mis-classification (14 out of 41) was 'Alohomora' (to unlock a door), which was 5 times more frequent in the training corpus. Similarly, 'Nox' (to extinguish the light from a wand) was correctly predicted 6 times, meanwhile 36 mis-classifications corre-

---

[8] Some macro F-scores do not lie within the Precision and Recall due to this issue.

[9] For each label, we compute their average, weighted by the number of true instances for each label. The F-score might be not between precision and recall.

[10] Made over one of the runs from the LSTM-based approach and setting the snippet size to 128 tokens.

spond to 'Lumos' (to light a place using a wand), which was 6 times more frequent in the training set. Other less frequent spells that denote vision and guidance actions, such as 'Point me' (the wand acts a a compass pointing North) and 'Homenum revelio' (to revel a human presence) were also mainly misclassified as 'Lumos'. This is an indicator that the LSTM approach has difficulties to disambiguate among semantically related actions, especially if their occurrence was unbalanced in the training set. This issue is in line with the tendency observed for recall at k. Spells intended for much more specific purposes, according to the books, obtained a performance significantly higher than the average, e.g. F-score('Riddikulus')=63.54, F-score('Expecto Patronum')=55.49 and F-score('Obliviate')=47.45. As said before, the model is significantly biased towards frequent actions. For 79 out of 84 gold actions in the test set, we found that the samples tagged with such actions were mainly classified into one of the top 20 most frequent actions.

Human comparison We collected human annotations from 208 scenes involving frequent actions. The accuracy/F-macro/F-weighted was 39.20/30.00/40.90. The LSTM approach obtained 41.26/25.37/39.86. Overall, the LSTM approach obtained a similar performance, but the lower macro F-score by the LSTM could be an indicator that humans can distinguish within a wider spectrum of actions. As a side note, super-human performance it is not strange in other NLP tasks, such as sentiment analysis (Pang et al., 2002).

5 Conclusion

We explored action prediction from written stories. We first introduced a corpus set in the world of Harry Potter's literature. Spells in these novels act as keywords that abstract actions. This idea was used to label a collection of fan fiction. We then evaluated standard NLP approaches, from logistic regression to sequential models such as LSTMs. The latter performed better in general, although vanilla models achieved a higher performance for actions that occurred a few times in the training set. An analysis over the output of the LSTM approach also revealed difficulties to discriminate among semantically related actions.

The challenge here proposed corresponded to a fictional domain. A future line of work we are interested in is to test whether the knowledge learned with this dataset could be transferred to real-word actions (i.e. real-domain setups), or if such transfer is not possible and a model needs to be trained from scratch.

Acknowlegments

This work has received support from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01), and from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150).

References

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2016. Embracing data abundance: Booktest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. pages 789–797.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

L. Fremann, S. B. Cohen, and M. Lapata. 2017. Whodunnit? Crime Drama as a Case for Natural Language Understanding. *Transactions of the Association for Computational Linguistics*, 6:1–15.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. pages 5010–5015.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representation*.

Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Smitha Milli and David Bamman. 2016. Beyond canonical texts: A computational analysis of fanfiction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2048–2053. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level lstm language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

S. Salant and J. Berant. 2017. Contextualized Word Representations for Reading Comprehension. *ArXiv e-prints*.

Roger C Schank and Robert P Abelson. 1977. Scripts: Plans, goals and understanding. *Lawrence Erlbaum*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łũkasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

A Corpus distribution

Table 6 summarizes the label distribution across the training, development and test sets of the HPAC corpus.

Action	#Training	#Dev	#Test	Action	#Training	#Dev	#Test
AVADA KEDAVRA	7937	986	954	CRUCIO	7852	931	980
ACCIO	4556	595	562	LUMOS	4159	505	531
STUPEFY	3636	471	457	OBLIVIATE	3200	388	397
EXPELLIARMUS	2998	377	376	LEGILIMENS	1938	237	247
EXPECTO PATRONUM	1796	212	242	PROTEGO	1640	196	229
SECTUMSEMPRA	1596	200	189	ALOHOMORA	1365	172	174
INCENDIO	1346	163	186	SCOURGIFY	1317	152	166
REDUCTO	1313	171	163	IMPERIO	1278	159	144
WINGARDIUM LEVIOSA	1265	158	154	PETRIFICUS TOTALUS	1253	175	134
SILENCIO	1145	153	136	REPARO	1124	159	137
MUFFLIATO	1005	108	92	AGUAMENTI	796	84	86
FINITE INCANTATEM	693	90	75	INCARCEROUS	686	99	87
NOX	673	82	80	RIDDIKULUS	655	81	88
DIFFINDO	565	90	82	IMPEDIMENTA	552	88	79
LEVICORPUS	535	63	68	EVANESCO	484	53	59
SONORUS	454	66	73	POINT ME	422	57	69
EPISKEY	410	55	59	CONFRINGO	359	52	48
ENGORGIO	342	52	41	COLLOPORTUS	269	26	41
RENNERVATE	253	24	33	PORTUS	238	22	31
TERGEO	235	23	26	MORSMORDRE	219	29	38
EXPULSO	196	23	20	HOMENUM REVELIO	188	30	24
MOBILICORPUS	176	20	14	RELASHIO	174	20	27
LOCOMOTOR	172	24	19	AVIS	166	17	29
RICTUSEMPRA	159	16	26	IMPERVIUS	149	26	13
OPPUGNO	144	18	7	FURNUNCULUS	137	20	20
SERPENSORTIA	133	14	15	CONFUNDO	130	17	21
LOCOMOTOR MORTIS	127	14	15	TARANTALLEGRA	126	11	17
REDUCIO	117	13	22	QUIETUS	108	15	17
LANGLOCK	99	12	19	GEMINIO	78	5	10
FERULA	78	6	10	ORCHIDEOUS	76	7	5
DENSAUGEO	67	13	8	LIBERACORPUS	63	7	5
APARECIUM	63	14	10	ANAPNEO	62	6	5
FLAGRATE	59	4	11	DELETRIUS	59	12	6
OBSCURO	57	11	7	PRIOR INCANTATO	56	4	3
DEPRIMO	51	2	2	SPECIALIS REVELIO	50	11	6
WADDIWASI	45	5	8	PROTEGO TOTALUM	44	9	5
DURO	36	4	4	SALVIO HEXIA	36	8	5
DEFODIO	34	2	6	PIERTOTUM LOCOMOTOR	30	4	3
GLISSEO	26	4	3	MOBILIARBUS	25	3	4
REPELLO MUGGLETUM	23	2	5	ERECTO	23	7	5
CAVE INIMICUM	19	5	2	DESCENDO	19	0	1
PROTEGO HORRIBILIS	18	7	5	METEOLOJINX RECANTO	10	3	1
PESKIPIKSI PESTERNOMI	7	0	0				

Table 6: Label distribution for the HPAC corpus

Argument Mining for Understanding Peer Reviews

Xinyu Hua, Mitko Nikolov, Nikhil Badugu, Lu Wang
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115
`{hua.x, nikolov.m, badugu.n}@husky.neu.edu`
`luwang@ccs.neu.edu`

Abstract

Peer-review plays a critical role in the scientific writing and publication ecosystem. To assess the efficiency and efficacy of the reviewing process, one essential element is to understand and evaluate the reviews themselves. In this work, we study the content and structure of peer reviews under the argument mining framework, through automatically detecting (1) argumentative propositions put forward by reviewers, and (2) their types (e.g., evaluating the work or making suggestions for improvement). We first collect 14.2K reviews from major machine learning and natural language processing venues. 400 reviews are annotated with $10,386$ propositions and corresponding types of EVALUATION, REQUEST, FACT, REFERENCE, or QUOTE. We then train state-of-the-art proposition segmentation and classification models on the data to evaluate their utilities and identify new challenges for this new domain, motivating future directions for argument mining. Further experiments show that proposition usage varies across venues in amount, type, and topic.

Review #1 (*rating*: 5, *# sentences*: 11)
[Quality: This paper demonstrates that convolutional and relational neural networks fail to solve visual relation problems . . .]$_{\text{FACT}}$ [This points at important limitations of current neural network architectures where architectures depend mainly on rote memorization.]$_{\text{EVAL}}$ . . . [Significance: This work demonstrates failures of relational networks on relational tasks. . .]$_{\text{FACT}}$ [Pros: Important message about network limitations.]$_{\text{EVAL}}$ [Cons: Straightforward testing of network performance on specific visual relation tasks.]$_{\text{EVAL}}$ . . .
Review #2 (*rating*: 5, *# sentences*: 10)
[The authors present two autoregressive models . . .]$_{\text{FACT}}$. . . [In that context , this work can be viewed as applying deep autoregressive density estimators to policy gradient methods.]$_{\text{EVAL}}$. . . [At least one of those papers ought to be cited.]$_{\text{REQ}}$ [It also seems like a simple, obvious baseline is missing from their experiments . . .]$_{\text{EVAL}}$. . . [The method could even be made to capture dependencies between different actions by adding a latent probabilistic layer . . .]$_{\text{EVAL}}$. . .[A direct comparison against one of the related methods in the discussion section would help]$_{\text{REQ}}$. . .

Figure 1: Sample ICLR review excerpts. Propositions are annotated with types, such as FACT (fact), EVAL (evaluation), and REQ (request). Review #2 contains in-depth evaluation and actionable suggestion, thus is perceived to be of a higher quality.

1 Introduction

Peer review is a process where domain experts scrutinize the quality of research work in their field, and it is a cornerstone of scientific discovery (Hettich and Pazzani, 2006; Kelly et al., 2014; Price and Flach, 2017). In 2015 alone, approximately 63.4 million hours were spent on peer reviews (Kovanis et al., 2016). To maximize their benefit to the scientific community, it is crucial to understand and evaluate the construction and limitation of reviews themselves. However, minimal work has been done to analyze reviews' content and structure, let alone to evaluate their qualities.

As seen in Figure 1, peer reviews resemble arguments: they contain **argumentative propositions** (henceforth propositions) that convey reviewers' interpretation and evaluation of the research. Constructive reviews, e.g., review #2, often contain in-depth analysis as well as concrete suggestions. As a result, automatically identifying propositions and their types would be useful to understand the composition of peer reviews.

Therefore, we propose *an argument mining-based approach to understand the content and structure of peer reviews*. Argument mining studies the automatic detection of argumentative components and structure within discourse (Peldszus and Stede, 2013). Specifically, argument types (e.g. evidence and reasoning) and their arrangement are indicative of argument quality (Habernal and Gurevych, 2016; Wachsmuth et al., 2017). In this work, we focus on two specific tasks: (1) **proposition segmentation**—detecting elementary argumentative discourse units that are

Proceedings of NAACL-HLT 2019, pages 2131–2137
Minneapolis, Minnesota, June 2 - June 7, 2019.

propositions, and (2) **proposition classification**—labeling the propositions according to their types (e.g., evaluation vs. request).

Since there was no annotated dataset for peer reviews, as part of this study, we first collect 14.2K reviews from major machine learning (ML) and natural language processing (NLP) venues. We create a dataset, **AMPERE** (Argument Mining for PEer REviews), by annotating 400 reviews with $10,386$ propositions and labeling each proposition with the type of EVALUATION, REQUEST, FACT, REFERENCE, QUOTE, or NON-ARG.[1] Significant inter-annotator agreement is achieved for proposition segmentation (Cohen's $\kappa = 0.93$), with good consensus level for type annotation (Krippendorf's $\alpha_U = 0.61$).

We benchmark our new dataset with state-of-the-art and popular argument mining models to better understand the challenges posed in this new domain. We observe a significant drop of performance for proposition segmentation on AMPERE, mainly due to its different argument structure. For instance, 25% of the sentences contain more than one proposition, compared to that of 8% for essays (Stab and Gurevych, 2017), motivating new solutions for segmentation and classification.

We further investigate review structure difference across venues based on proposition usage, and uncover several patterns. For instance, ACL reviews tend to contain more propositions than those in ML venues, especially with more requests but fewer facts. We further find that reviews with extreme ratings, i.e., strong reject or accept, tend to be shorter and make much fewer requests. Moreover, we probe the salient words for different proposition types. For example, ACL reviewers ask for more "examples" when making requests, while ICLR reviews contain more evaluation of "network" and how models are "trained".

2 AMPERE Dataset

We collect review data from three sources: (1) `openreview.net`—an online peer reviewing platform for ICLR 2017, ICLR 2018, and UAI 2018 [2]; (2) reviews released for accepted papers at NeurIPS from 2013 to 2017; and (3) opted-in reviews for ACL 2017 from Kang et al. (2018).

EVALUATION: Subjective statements, often containing qualitative judgment. Ex: *"This paper shows nice results on a number of small tasks."*
REQUEST: Statements suggesting a course of action. Ex: *"The authors should compare with the following methods."*
FACT: Objective information of the paper or commonsense knowledge. Ex: *"Existing works on multi-task neural networks typically use hand-tuned weights..."*
REFERENCE: Citations and URLs. Ex: *"see MuseGAN (Dong et al), MidiNet (Yang et al), etc "*
QUOTE: Quotations from the paper. Ex: *"The author wrote 'where r is lower bound of feature norm'."*
NON-ARG: Non-argumentative statements. Ex: *"Aha, now I understand."*

Table 1: Proposition types and examples.

Dataset	#Doc	#Sent	#Prop
Comments (Park and Cardie, 2018)	731	3,994	4,931
Essays (Stab and Gurevych, 2017)	402	7,116	6,089
News (Al Khatib et al., 2016)	300	11,754	14,313
Web (Habernal and Gurevych, 2017)	340	3,899	1,882
AMPERE	400	8,030	10,386

Table 2: Statistics for AMPERE and some argument mining corpora, including # of annotated propositions.

In total, $14,202$ reviews are collected (ICLR: $4,057$; UAI: 718; ACL: 275; and NeurIPS: $9,152$). All venues except NeurIPS have paper rating scores attached to the reviews.

Annotation Process. For proposition segmentation, we adopt the concepts from Park et al. (2015) and instruct the annotators to identify elementary argumentative discourse units on sentence or sub-sentence level, based on their discourse functions and topics. They then classify the propositions into five types with an additional non-argument category, as explained in Table 1.

400 ICLR 2018 reviews are sampled for annotation, with similar distributions of length and rating to those of the full dataset. Two annotators who are fluent English speakers first label the 400 reviews with proposition segments and types, and a third annotator then resolves disagreements.

We calculate the inter-annotator agreement between the two annotators. A Cohen's κ of 0.93 is achieved for proposition segmentation, with each review treated as a BIO sequence. For classification, unitized Krippendorf's α_U (Krippendorff, 2004), which considers disagreements among segmentation, is calculated per review and then averaged over all samples, and the value is 0.61. Among the exactly matched proposition segments, we report a Cohen's κ of 0.64.

Statistics. Table 2 shows comparison between AMPERE and some other argument min-

---

[1] Dataset and annotation guideline can be found at `http://xinyuhua.github.io/Resources/naacl19/`.

[2] ICLR reviews are downloaded from the public API: `https://github.com/iesl/openreview-py`. UAI reviews are collected by the OpenReview team.

ing datasets of different genres. We also show the number of propositions in each category in Table 3. The most frequent types are evaluation (38.3%) and fact (36.5%).

Eval	Req	Fact	Ref	Quot	Non-A	Total
3,982	1,911	3,786	207	161	339	10,386

Table 3: Number of propositions per type in AMPERE.

3 Experiments with Existing Models

We benchmark AMPERE with popular and state-of-the-art models for proposition segmentation and classification. Both tasks can be treated as sequence tagging problems with the setup similar to Schulz et al. (2018). For experiments, 320 reviews (7, 999 propositions) are used for training and 80 reviews (2, 387 propositions) are used for testing. Following Niculae et al. (2017), 5-fold cross validation on the training set is used for hyperparameter tuning. To improve the accuracy of tokenization, we manually replace mathematical formulas, variables, URL links, and formatted citation with special tokens such as `<EQN>`, `<VAR>`, `<URL>`, and `<CIT>`. Parameters, lexicons, and features used for the models are described in the supplementary material.

3.1 Task I: Proposition Segmentation

We consider three baselines. **FullSent**: treating each sentence as a proposition. **PDTB-conn**: further segmenting sentences when any discourse connective (collected from Penn Discourse Treebank (Prasad et al., 2007)) is observed. **RST-parser**: segmenting discourse units by the RST parser in Feng and Hirst (2014).

For learning-based methods, we start with Conditional Random Field (**CRF**) (Lafferty et al., 2001) with features proposed by Stab and Gurevych ((2017), Table 7), and **BiLSTM-CRF**, a bidirectional Long Short-Term Memory network (BiLSTM) connected to a CRF output layer and further enhanced with ELMo representation (Peters et al., 2018). We adopt the BIO scheme for sequential tagging (Ramshaw and Marcus, 1999), with O corresponding to Non-Arg. Finally, we consider **jointly modeling** segmentation and classification by appending the proposition types to BI tags, e.g., B-fact, with CRF (**CRF-joint**) and BiLSTM-CRF (**BiLSTM-CRF-joint**).

	Prec.	**Rec.**	**F1**
FullSent	73.68	56.00	63.64
PDTB-conn	51.11	49.71	50.40
RST-parser	30.28	43.00	35.54
CRF	66.53	52.92	58.95
BiLSTM-CRF	**82.25**	**79.96**	**81.09***
CRF-joint	74.99	63.33	68.67
BiLSTM-CRF-joint	81.12	78.42	79.75

Table 4: Proposition segmentation results. Result that is significantly better than all comparisons is marked with $*$ ($p < 10^{-6}$, McNemar test).

	Overall	Eval	Req	Fact	Ref	Quot
With Gold-Standard Segments						
Majority	40.75	57.90	–	–	–	–
PropLexicon	36.83	40.42	36.07	32.23	59.57	31.28
SVM	60.98	63.88	**69.02**	54.74	**69.47**	7.69
CNN	**66.56***	**69.02**	63.26	**66.17**	67.44	**52.94**
With Predicted Segments						
Majority	33.30	47.60	–	–	–	–
PropLexicon	23.21	22.45	23.97	23.73	35.96	16.67
SVM	51.46	54.05	48.16	52.77	52.27	4.71
CNN	55.48	57.75	53.71	55.19	48.78	33.33
CRF-joint	50.69	46.78	55.74	52.27	**55.77**	26.47
BiLSTM-CRF-joint	**62.64***	**62.36***	**67.31***	**61.86**	54.74	**37.36**

Table 5: Proposition classification F1 scores. Results that are significant better than other methods are marked with $*$ ($p < 10^{-6}$, McNemar test).

Table 4 shows that BiLSTM-CRF outperforms other methods in F1. More importantly, the performance on reviews is lower than those reached on existing datasets, e.g., an F1 of 86.7 is obtained by CRF for essays (Stab and Gurevych, 2017). This is mostly due to essays' better structure, with frequent use of discourse connectives.

3.2 Task II: Proposition Classification

With given proposition segments, predicted or gold-standard, we experiment with proposition-level models to label proposition types.

We utilize two baselines. **Majority** simply assigns the majority type in the training set. **PropLexicon** matches the following lexicons for different proposition types in order, and returns the first corresponding type with a match; if no lexicon is matched, the proposition is labeled as Non-Arg:

- Reference: `<URL>`, `<CIT>`
- Quote: “, ”, ’
- Request: *should, would be nice, why, please, would like to, need*
- Evaluation: *highly, very, unclear, clear, interesting, novel, well, important, similar, clearly, quite, good*
- Fact: *author, authors, propose, present, method, parameters, example, dataset, same, incorrect, correct*

For supervised models, we employ linear **SVM** with a squared hinge loss and group Lasso regularizer (Yuan and Lin, 2006). It is trained with the top 500 features selected from Table 9 in (Stab and Gurevych, 2017) by χ^2 test. We also train a convolutional neural network (**CNN**) proposed by Kim (2014), with the same setup and pre-trained word embeddings from word2vec (Mikolov et al., 2013). Finally, results by joint models of CRF and BiLSTM-CRF are also reported.

F1 scores for all propositions and each type are reported in Table 5. A prediction is correct when both segment and type are matched with the true labels. CNN performs better for types with significantly more training samples, i.e., evaluation and fact, indicating the effect of data size on neural model's performance. Joint models (CRF-joint and BiLSTM-CRF-joint) yield the best F1 scores for all categories when gold-standard segmentation is unavailable.

4 Proposition Analysis by Venues

Here we leverage the BiLSTM-CRF-joint model trained on the annotated AMPERE data to identify propositions and their types in unlabeled reviews from the four venues (ICLR, UAI, ACL, and NeurIPS), to understand the content and structure of peer reviews at a larger scale.

Proposition Usage by Venue and Rating. Figure 2 shows the average number of propositions per review, grouped by venue and rating. Scores in $1-10$ are scaled to $1-5$ by $\lceil x/2 \rceil$, with 1 as strong reject and 5 as strong accept. ACL and NeurIPS have significantly more propositions than ICLR and UAI. Ratings, which reflect a reviewer's judgment of paper quality, also affect proposition usage. We find that reviews with extreme ratings, i.e., 1 and 5, tend to have fewer propositions.

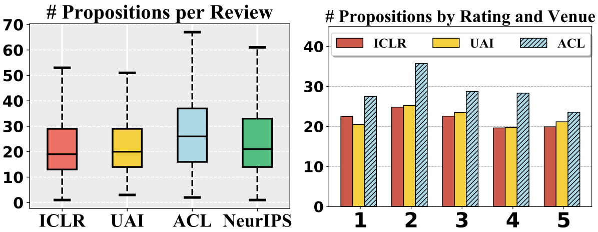


Figure 2: Proposition number in reviews. Differences among venues are all significant except UAI vs. ICLR and ACL vs. NeurIPS ($p < 10^{-6}$, unpaired t-test).

We further study the distribution of proposition type in each venue. As observed in Figure 3, ACL reviews contain more requests but fewer facts than other venues. Specifically, we find that 94.6% of ACL reviews have at least one REQUEST proposition, compared to 81.5% for ICLR and 84.7% for UAI. We also show proposition type distribution based on ratings in Figure 4. Reviews with the highest rating tend to use fewer evaluation and reference, while reviews with ratings of $3-4$ (borderline or weak accept) contain more requests. We further observe a sharp decrease of QUOTE usage in rating group 4, and a surge of NON-ARG for rating group 5, while FACT remains consistent across rating ranges.

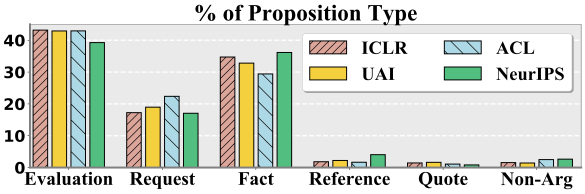


Figure 3: Distribution of proposition type per venue.

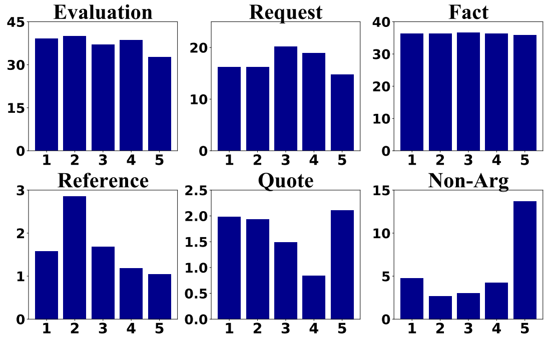


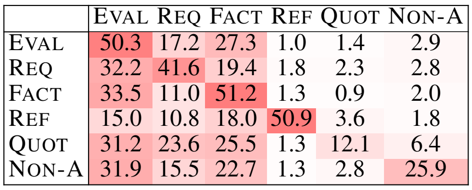
Figure 4: Distribution of proposition type per rating (in %) on AMPERE.

Proposition Structure. Argumentative structure, which is usually studied as support and attack relations, reveals how propositions are organized into coherent text. According to Park and Cardie (2018), 75% of support relations happen between adjacent propositions in user comments. We thus plot the proposition transition probability matrix in Figure 5, to show the argument structure in AMPERE. The high probabilities along the diagonal line imply that propositions of the same type are often constructed consecutively, with the exception of quote, which is more likely to be followed by evaluation.

Proposition Type and Content. We also probe the salient words used for each proposition type, and the difference of their usage across venues. For each venue, we utilize log-likelihood ratio test (Lin and Hovy, 2000) to identify the represen-

	EVALUATION	REQUEST	FACT	REFERENCE	QUOTE
All Venues	overall, unclear, not, contribution, seem, interesting	please, could, should, if, why, would, more, suggest	think, each, some, data, useful, written, proposes	`<URL>`, et, al., conference, paper, proceedings, arxiv	”, paper, we, :, our
ICLR	network, general, acceptance, convinced, trained	network, appendix, recommend, because, novelty	training, results, work, then, image	deep, ;, nips, pp., speech	not, section, 4, 5, agent
UAI	quality, relevant, found, presentation, major	`<VAR>`, model, method, nice, column	stochastic, called, considers, sense, writing	artificial, discovery, etc., via, systems	–, second, column, processes, connections
ACL	weaknesses, strengths, so, word, main	consider, examples, further, models, proposed	word, method, words, proposed, embeddings	language, extraction, emnlp, computational, linguistics	
NeurIPS	theoretical, `<EQN>`, interest, practical, nips	following, clarity, address, significance, quality	`<EQN>`, maximum, may, comments, characters	for, see, class, detailed, guidelines	of, in, which, `<EQN>`, reviewer

Table 6: Salient words ($\alpha = 0.001$, χ^2 test) per proposition type. Top 5 frequent words that are unique for each venue are shown. “`<EQN>`”, “`<URL>`”, and “`<VAR>`” are equations, URL links, and variables.



	EVAL	REQ	FACT	REF	QUOT	NON-A
EVAL	50.3	17.2	27.3	1.0	1.4	2.9
REQ	32.2	41.6	19.4	1.8	2.3	2.8
FACT	33.5	11.0	51.2	1.3	0.9	2.0
REF	15.0	10.8	18.0	50.9	3.6	1.8
QUOT	31.2	23.6	25.5	1.3	12.1	6.4
NON-A	31.9	15.5	22.7	1.3	2.8	25.9

Figure 5: Proposition transition prob. on AMPERE.

tative words in each proposition type compared to other types. Table 6 shows both the commonly used salient words across venues and the unique words with top frequencies for each venue ($\alpha = 0.001$, χ^2 test). For evaluation, all venues tend to focus on clarity and contribution, with ICLR discussing more about “network” and NeurIPS often mentioning equations. ACL reviews then frequently request for “examples”.

5 Related Work

There is a growing interest in understanding the content and assessing the quality of peer reviews. Authors’ feedback such as satisfaction and helpfulness have been adopted as quality indicators (Latu and Everett, 2000; Hart-Davidson et al., 2010; Xiong and Litman, 2011). Nonetheless, they suffer from author subjectivity and are often influenced by acceptance decisions (Weber et al., 2002). Evaluation by experts or editors proves to be more reliable and informative (van Rooyen et al., 1999), but requires substantial work and knowledge of the field. Shallow linguistic features, e.g., sentiment words, are studied in Bornmann et al. (2012) for analyzing languages in peer reviews. To the best of our knowledge, our work is the first to understand the content and structure of peer reviews via argument usage.

Our work is also in line with the growing body of research in argument mining (Teufel et al., 1999; Palau and Moens, 2009). Most of the work focuses on arguments in social media posts (Park and Cardie, 2014; Wei et al., 2016; Habernal and Gurevych, 2016), online debate portals or Oxford-style debates (Wachsmuth et al., 2017; Hua and Wang, 2017; Wang et al., 2017), and student essays (Persing and Ng, 2015; Ghosh et al., 2016). We study a new domain of peer reviews, and identify new challenges for existing models.

6 Conclusion

We study the content and structure of peer reviews under the argument mining framework. AMPERE, a new dataset of peer reviews, is collected and annotated with propositions and their types. We benchmark AMPERE with state-of-the-art argument mining models for proposition segmentation and classification. We leverage the classifiers to analyze the proposition usage in reviews across ML and NLP venues, showing interesting patterns in proposition types and content.

Acknowledgements

This research is based upon work supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341. We are grateful to the OpenReview team, especially Michael Spector, for setting up the API to facilitate review data collection. We also thank three anonymous reviewers for their constructive suggestions.

References

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443. The COLING 2016 Organizing Committee.

Lutz Bornmann, Markus Wolf, and Hans-Dieter Daniel. 2012. Closed versus open reviewing of journal manuscripts: how far do comments differ in language use? *Scientometrics*, 91(3):843–856.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521. Association for Computational Linguistics.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 549–554. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

William Hart-Davidson, Michael McLeod, Christopher Klerkx, and Michael Wojcik. 2010. A method for measuring helpfulness in online peer review. In *Proceedings of the 28th ACM international conference on design of communication*, pages 115–121. ACM.

Seth Hettich and Michael J Pazzani. 2006. Mining for proposal reviewers: lessons learned at the national science foundation. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 862–871. ACM.

Xinyu Hua and Lu Wang. 2017. Understanding and detecting supporting arguments of diverse types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 203–208, Vancouver, Canada. Association for Computational Linguistics.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1647–1661. Association for Computational Linguistics.

Jacalyn Kelly, Tara Sadeghieh, and Khosrow Adeli. 2014. Peer review in scientific publications: benefits, critiques, & a survival guide. *EJIFCC*, 25(3):227.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Michail Kovanis, Raphaël Porcher, Philippe Ravaud, and Ludovic Trinquart. 2016. The global burden of journal peer review in the biomedical literature: Strong imbalance in the collective enterprise. *PLoS One*, 11(11):e0166387.

Klaus Krippendorff. 2004. Measuring the reliability of qualitative text analysis data. *Quality and Quantity*, 38:787–800.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Tavite M Latu and André M Everett. 2000. *Review of satisfaction research and measurement approaches*. Citeseer.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured svms and rnns. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995. Association for Computational Linguistics.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM.

Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015. Toward machine-assisted participation in erulemaking: An argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, pages 206–210. ACM.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38. Association for Computational Linguistics.

Joonsuk Park and Claire Cardie. 2018. A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The penn discourse treebank 2.0 annotation manual.

Simon Price and Peter A Flach. 2017. Computational support for academic peer review: A perspective from artificial intelligence. *Communications of the ACM*, 60(3):70–79.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Susan van Rooyen, Nick Black, and Fiona Godlee. 1999. Development of the review quality instrument (rqi) for assessing peer reviews of manuscripts. *Journal of clinical epidemiology*, 52(7):625–629.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 35–41. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Simone Teufel et al. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, Citeseer.

Henning Wachsmuth, Nona Naderi, Ivan Habernal, Yufang Hou, Graeme Hirst, Iryna Gurevych, and Benno Stein. 2017. Argumentation quality assessment: Theory vs. practice. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 250–255. Association for Computational Linguistics.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *Transactions of the Association for Computational Linguistics*, 5:219–232.

Ellen J Weber, Patricia P Katz, Joseph F Waeckerle, and Michael L Callaham. 2002. Author perception of peer review: impact of review quality and acceptance on satisfaction. *JAMA*, 287(21):2790–2793.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200. Association for Computational Linguistics.

Wenting Xiong and Diane Litman. 2011. Automatically predicting peer-review helpfulness. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 502–507. Association for Computational Linguistics.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

An Annotated Dataset of Literary Entities

David Bamman
School of Information
UC Berkeley
`dbamman@berkeley.edu`

Sejal Popat
School of Information
UC Berkeley
`sejal@berkeley.edu`

Sheng Shen
Computer Science Division
UC Berkeley
`sheng.s@berkeley.edu`

Abstract

We present a new dataset comprised of 210,532 tokens evenly drawn from 100 different English-language literary texts annotated for ACE entity categories (person, location, geo-political entity, facility, organization, and vehicle). These categories include non-named entities (such as "the boy", "the kitchen") and nested structure (such as [[the cook]'s sister]). In contrast to existing datasets built primarily on news (focused on geo-political entities and organizations), literary texts offer strikingly different distributions of entity categories, with much stronger emphasis on people and description of settings. We present empirical results demonstrating the performance of nested entity recognition models in this domain; training natively on in-domain literary data yields an improvement of over 20 absolute points in F-score (from 45.7 to 68.3), and mitigates a disparate impact in performance for male and female entities present in models trained on news data.

1 Introduction

Computational literary analysis works at the intersection of natural language processing and literary studies, drawing on the structured representation of text to answer literary questions about character (Underwood et al., 2018), objects (Tenen, 2018) and place (Evans and Wilkens, 2018).

Much of this work relies on the ability to extract entities accurately, including work focused on modeling (Bamman et al., 2014; Iyyer et al., 2016; Chaturvedi et al., 2017). And yet, with notable exceptions (Vala et al., 2015; Brooke et al., 2016), nearly all of this work tends to use NER models that have been trained on non-literary data, for the simple reason that labeled data exists for domains like news through standard datasets like ACE (Walker et al., 2006), CoNLL (Tjong Kim Sang and De Meulder, 2003) and OntoNotes (Hovy et al., 2006)—and even historical non-fiction (DeLozier et al., 2016; Rayson et al., 2017)—but not for literary texts.

This is naturally problematic for several reasons: models trained on out-of-domain data surely degrade in performance when applied to a very different domain, and especially for NER, as Augenstein et al. (2017) has shown; and without in-domain *test* data, it is difficult to directly estimate the severity of this degradation. At the same time, literary texts also demand slightly different representations of entities. While classic NER models typically presume a flat entity structure (Finkel and Manning, 2009), relevant characters and places (and other entities) in literature need not be flat, and need not be named: *The cook's sister ate lunch* contains two `PER` entities ([The cook] and [The cook's sister]).

We present in this work a new dataset of entity annotations for a wide sample of 210,532 tokens from 100 literary texts to help address these issues and help advance computational work on literature. These annotations follow the guidelines set forth by the ACE 2005 entity tagging task (LDC, 2005) in labeling all nominal entities (named and common alike), including those with nested structure. In evaluating the stylistic difference between the texts in ACE 2005 (primarily news) and the literary texts in our new dataset, we find considerably more attention dedicated to people and settings in literature; this attention directly translates into substantially improved accuracies for those classes when models are trained on them. The dataset is freely available for download under a Creative Commons ShareAlike 4.0 license at `https://github.com/dbamman/litbank`.

Proceedings of NAACL-HLT 2019, pages 2138–2144
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Corpus

We draw our corpus from the public-domain texts on Project Gutenberg, selecting individual works of fiction (both novels and short stories) that include a mix of high literary style (e.g., Edith Wharton's *Age of Innocence*, James Joyce's *Ulysses*) and popular pulp fiction (e.g., H. Rider Haggard's *King Solomon's Mines*, Horatio Alger's *Ragged Dick*). All texts are published before 1923 (the current threshold for public domain in the United States), with the majority falling between 1852 and 1911.

From each text, we select approximately the first 2,000 words as a sample, yielding a total dataset of 210,532 tokens.

3 Annotation

We adopt the ACE 2005 guidelines for entity annotation, focusing on the subset of people (`PER`), natural locations (`LOC`), built facilities (`FAC`), geo-political entities (`GPE`), organizations (`ORG`) and vehicles (`VEH`).[1] While traditional named entity recognition presumes a flat structure in which entity labels cannot be embedded within each other, we allow for nested structure, as in the following (from Jane Austen's *Emma*):

$$\ldots \overbrace{\text{the elder brother of } \overbrace{\overbrace{\text{Isabella}}^{\texttt{PER}} \text{'s husband}}^{\texttt{PER}}}^{\texttt{PER}}$$

This nested structure is in fact quite common in our data, with entities that contain at least one level of nesting accounting for 13.8% of the annotations—86.2% contain no nesting (as in *Isabella* above), 12.5% contain one level (*Isabella's husband*), 1.2% contain two (*the elder brother of Isabella's husband*), and 0.1% contain three. The dataset contains a total of 13,912 entity annotations.

3.1 Entity types

We generally follow the ACE annotation guidelines for each of the entity classes and restrict our annotations to proper and common noun phrases (i.e., excluding pronouns or WH-question words); table 1 illustrates examples for each class.

PER. By person we describe a single person indicated by a proper name (Tom Saywer) or common entity (the boy); or set of people, such as *her daughters* and *the Ashburnhams*.

FAC. ACE guidelines define a facility as a "functional, primarily man-made structure" designed for human habitation (buildings, museums), storage (barns, parking garages), transportation infrastructure (streets, highways), and maintained outdoor spaces (gardens) (LDC, 2005). We adopt the ACE threshold for taggability here as well, and rooms and closets within a house as the smallest possible facility.

GPE. Geo-political entities are single units that contain a population, government, physical location, and political boundaries (LDC, 2005). In literary data, this includes not only cities that have known geographical locations within the real world (London, New York), or nations (England, the United States), but also both named and common imagined entities as well (the town, the village).

LOC. Locations describe entities with physicality but without political entities. In our dataset, this includes named regions without political organization (New England, the South) and planets (Mars). The most common class, however, are geologically designated areas describing natural settings, such as the sea, the river, the country, the valley, the woods, and the forest.

VEH. Literary texts include a number of vehicles defined as "a physical device primarily designed to move an object from one location to another" (LDC, 2005); ships, trains, and carriages dominate since nearly all texts were written before the rise of automobiles.

ORG. Organizations are defined by the criterion of formal association and are relatively rare in literary data, comprising the least frequently occurring entity class. The most frequent organizations include the army and the Church (as an administrative entity, distinct from the church as a facility with a physical location).

3.2 Figurative language

Literary language in particular presents several unique challenges to entity annotation, including metaphor, personification and metonymy.

---

[1] We exclude the ACE category of weapons (`WEA`), since that class is rarely attested in our data.

Entity type	Count	Examples
PER	9,383	my mother, Jarndyce, the doctor, a fool, his companion
FAC	2,154	the house, the room, the garden, the drawing-room, the library
LOC	1,170	the sea, the river, the country, the woods, the forest
GPE	878	London, England, the town, New York, the village
VEH	197	the ship, the car, the train, the boat, the carriage
ORG	130	the army, the Order of Elks, the Church, Blodgett College

Table 1: Entity classes annotated in literary data.

Metaphor. For non-figurative texts, predicative structures like *John is a doctor* nearly always entail the two comparands to be identical in their entity type (here, *John* and *a doctor* are both `PER`). Literary texts, however, are awash in figurative metaphor, such as "the young man was not really a poet; but surely he was a poem" (Chesterton, *The Man Who Was Thursday*). In such cases where the metaphor takes a predicative structure of *x is y*, we annotate only those phrases whose type describes an entity class (in this case, labeling *a poet* as a `PER`, but not *a poem*).

Personification. Several works, such as London's *The Call of the Wild* and Sewell's *Black Beauty*, feature personified animals as main characters, with dialogue and evident cognition. We expand the criteria for `PER` to include such characters who engage in dialogue or have reported internal monologue, regardless of their human status (this includes depicted non-human life forms in science fiction, such as aliens and robots, as well).

Metonymy. Metonymy is a rhetorical device of describing one aspect of a concept by a closely related one (such as the *White House* to refer to the organization of government it houses). We see many examples of metonymy in literature, such as the following:

> 'Them men would eat and drink if we was all in our graves,' said the indignant cook, who indeed had a real grievance; and the outraged sentiment of *the kitchen* was avenged by a bad and hasty dinner." (Oliphant, Miss Marjoribanks)

Following ACE, we annotate such examples by annotating the evoked entity class; in this case, annotating *the kitchen* as a `PER` (describing a set of cooks who feel outrage) rather than as a `FAC`.

3.3 Annotation process

Two co-authors annotated all 100 texts with a single pass between them after an initial phase of discussions about the annotation process, difficulties encountered and formalizing annotation decisions specific to literary texts. At the end of annotating, the inter-annotator agreement was calculated by double-annotating the same five texts and measuring the F1 score. We find that agreement rate to be high (86.0 F), likely due to the existence of thorough previous guidelines in ACE that both annotators were able to reference during the process of annotation.[2]

4 Comparison with ACE

We can compare the properties of this dataset to those of the ACE 2005 annotated data. To enable an apples-to-apples comparison, we filter the ACE data to exclude entity labels for tokens that are marked with a mention type of pronoun (`PRO`) or WH-question (`WHQ`) and remove all weapon (`WEA`) labels; we consider only the subsets for broadcast conversation (bc), broadcast news (bn), newswire (nw) and weblog (wl), as in past work (Lu and Roth, 2015; Muis and Lu, 2017; Ju et al., 2018).

Figure 2 plots the difference in entity label distributions between the ACE 2005 data and our literary data: literature has a proportionally higher ratio of person and facility mentions, and much lower mentions for GPEs and organizations.

4.1 Prediction

To understand how this different distribution of entity types impacts the performance of models trained on these different sources, we evaluate the performance of a state-of-the-art model for nested

---

[2] Note we report F-score since we are measuring the agreement rates between annotators not only in their choice of labels (for which a categorical chance-corrected measure like Cohen's κ would be appropriate), but also the *spans* in text to which they apply.

Train → Test	Precision	Recall	F
ACE → ACE	75.3 [72.7–77.9]	63.3 [60.5–66.1]	68.8 [66.2–71.2]
ACE → Literature	57.8 [54.2–61.5]	37.7 [35.1–40.6]	45.7 [42.8–48.6]
Literature → Literature	75.1 [72.1–77.8]	62.6 [59.7–65.4]	68.3 [65.5–70.8]

Table 2: Performance on nested entity recognition using the layered BiLSTM-CRF of Ju et al. (2018) with different training → test combinations. All metrics are reported with 95% bootstrap confidence intervals.

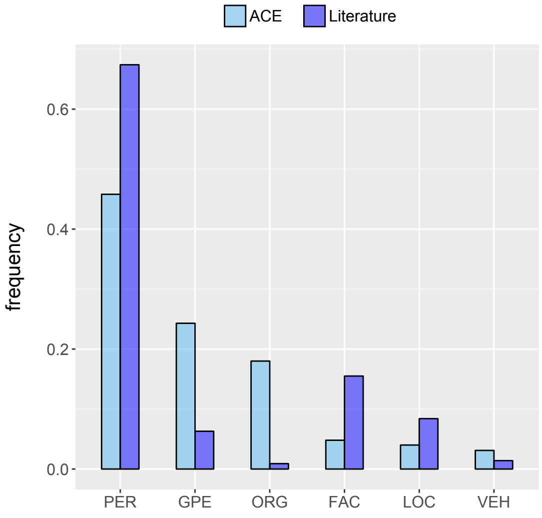


Figure 1: Distribution of entity types

entity recognition (Ju et al., 2018). We create training, development and test splits on the 100 literary books by stratifying at the document level, with 80 books in training, 10 books in development and 10 books in test.

To preprocess ACE, we tokenize and split sentences using the Stanford tokenizer (Manning et al., 2014), and create training, development and test partitions again stratified by document, so that sentences from the same document do not appear in both train and test. As above, we adapt the ACE annotations to our format by removing pronoun (`PRO`) and WH-question (`WHQ`) annotations and remove all weapon (`WEA`) labels, and consider only the subsets for broadcast conversation, broadcast news, newswire and weblogs. We present results with 95% confidence intervals using the bootstrap.

When trained on ACE and tested on ACE, the layered bidirectional LSTM-CRF of Ju et al. (2018) achieves an F-score of 68.8. When that same model (trained on ACE) is evaluated on our literature data, performance drops precipitously (23 absolute points in F-score). This alone—that cross-domain performance can be so strikingly worse—is a significant result, providing the first estimate of how performance degrades across these domains for this task.

However, when we train an identically parameterized model on the training partition of the literary data and evaluate it on the literary test partition, performance naturally improves substantially to an F-score of 68.3. As table 2 shows, performance improves dramatically for nearly all entity classes; the classes with the most statistically significant improvement are `PER` and `FAC`—both of which improve by 20 absolute points.

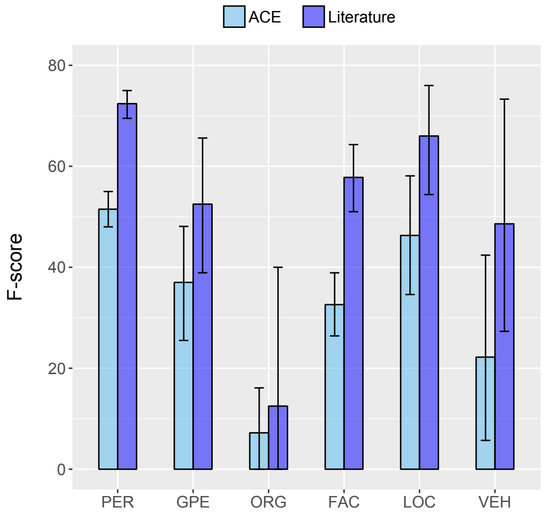


Figure 2: F-scores with 95% bootstrap confidence intervals by entity type when evaluating on literature test data with different training sources.

4.2 Analysis

To better understand the ways in which a model trained on ACE data differs in its predictions from an identically parameterized model trained on literary data, we used the two models described above to generate predictions for nested entities in a random sample of 1,000 full-text books from Project Gutenberg not in our training, development or test data (a total of 78M tokens). We then analyzed a simple difference in frequencies between the predictions of the two models on that same data; for a given entity e (e.g., *the boy*) and category t (e.g., `PER`), we calculate the frequency f as the number of times e was tagged by each model as t, and measure the difference:

$$f_{LIT}(e,t) - f_{ACE}(e,t)$$

The ten terms with the strongest positive difference in frequencies for the `PER` class—those phrases that are found significantly more often in a model trained on literary data than a model trained on ACE—are *Mrs., Miss, Lady, Aunt, Sir, Captain, no one, Mr, Madame* and *nobody*, suggesting a potential gender bias in the predictions; indeed, while ACE 2005 contains 47 instances of *Mr.*, it contains no mentions of *Mrs.* or honorific *Miss* (and only three instances of *Ms.*). While other work has demonstrated the gender bias present in word embeddings (Bolukbasi et al., 2016; Caliskan et al., 2017; Zhao et al., 2019) and in such NLP tasks as coreference resolution (Rudinger et al., 2018; Zhao et al., 2018), sentiment analysis (Kiritchenko and Mohammad, 2018), and speech recognition (Tatman, 2017), we can investigate the same phenomenon here: does a model trained on ACE result in a disparate impact in its performance recognizing men and women entities in text?

To answer this question, we annotate the gender for all `PER` entities in the literary test data (a total of 969 entities) and measure the recall of each model as a function of the gender of the true entity (measuring, for example, how many women in the gold literary data each model was able to identify, and how many men).

Training	Women	Men	Diff	p
ACE	38.0	49.6	−11.6	0.0009
Literature	69.3	68.2	1.1	0.7459

Table 3: Recall on literary test data by the gender of `PER` entity for models trained on ACE and literary data.

Table 4 lists these results: while a model trained on literary data recognizes male and female entities at roughly equal rates, ACE data shows a strong disparate performance, with female entities recognized at a rate over 11 points worse than male entities. This difference is significant at $p < 0.001$ under a permutation test (randomly shuffling the gender labels assigned to entities to generate a non-parametric null distribution, with 100,000 permutations).

If we remove the obvious entities from the gold data that begin with *Mrs.* and *Miss* (the honorifics that are rarely attested in ACE) along with those that begin with *Mr.*, we still see a sizable disparity in performance, suggesting that this result is more pervasive than the simple absence of those honorifics from the training data.

Training	Women	Men	Diff	p
ACE	40.4	48.3	−7.9	0.0358
Literature	63.7	67.1	−3.4	0.3542

Table 4: Recall on literary test data by the gender of `PER` entity for models trained on ACE and literary data, excluding all gold entities beginning with *Mrs.*, *Miss* and *Mr.*

5 Conclusion

We present in this work a new dataset of nested entity annotations for literature; such data allows us to measure the performance of existing NER systems when evaluated on literary data, train new models optimized for literature as a domain, and explore the stylistic differences in entity attention that help define literature as a genre. In addition to helping advance the state-of-the-art in NLP for literary texts, we provide this dataset to advance modeling for entity recognition generally; as Søgaard (2013) argues, the robustness of performance improvements for methods in NLP is best estimated by performance across a range of domains; we would expect a robust model that shows improvement on news entities in ACE and proteins in GENIA to show improvements on recognizing characters and settings in literature as well.

All data is freely available for public use under a Creative Commons Sharealike license and is available at: `https://github.com/dbamman/litbank`; code to support this work can be found at: `https://github.com/dbamman/NAACL2019-literary-entities`.

Acknowledgments

We thank the anonymous reviewers, Matt Sims and Jon Gillick for their valuable feedback. The research reported in this article was supported by an Amazon Research Award, a grant from the Digital Humanities at Berkeley initiative and by resources provided by NVIDIA.

References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition. *Comput. Speech Lang.*, 44(C):61–83.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4356–4364, USA. Curran Associates Inc.

Julian Brooke, Adam Hammond, and Timothy Baldwin. 2016. Bootstrapped text-level named entity recognition for literature. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 344–350. Association for Computational Linguistics.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé III. 2017. Unsupervised learning of evolving relationships between literary characters. In *Association for the Advancement of Artificial Intelligence*.

Grant DeLozier, Ben Wing, Jason Baldridge, and Scott Nesbit. 2016. Creating a novel geolocation corpus from historical texts. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 188–198. Association for Computational Linguistics.

Elizabeth F. Evans and Matthew Wilkens. 2018. Nation, ethnicity, and the geography of British fiction, 1880–1940. *Cultural Analytics*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *North American Association for Computational Linguistics*.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1446–1459.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53. Association for Computational Linguistics.

LDC. 2005. Ace (automatic content extraction) English annotation guidelines for entities. `https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-entities-guidelines-v5.6.6.pdf`.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618. Association for Computational Linguistics.

Paul Rayson, Alex Reinhold, James Butler, Chris Donaldson, Ian Gregory, and Joanna Taylor. 2017. A deeply annotated testbed for geographical text analysis: The corpus of lake district writing. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on Geospatial Humanities*, pages 9–15. ACM.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14. Association for Computational Linguistics.

Anders Søgaard. 2013. Estimating effect size across datasets. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 607–611. Association for Computational Linguistics.

Rachael Tatman. 2017. Gender and dialect bias in YouTube's automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, Valencia, Spain. Association for Computational Linguistics.

Dennis Yi Tenen. 2018. Toward a computational archaeology of fictional space. *New Literary History*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Ted Underwood, David Bamman, and Sabrina Lee. 2018. The transformation of gender in English-language fiction. *Cultural Analytics*.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 769–774, Lisbon, Portugal. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. LDC.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *NAACL*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20. Association for Computational Linguistics.

Abusive Language Detection with Graph Convolutional Networks

Pushkar Mishra★, **Marco Del Tredici**♣, **Helen Yannakoudakis**♠, **Ekaterina Shutova**♣
★ Facebook AI, London, United Kingdom
♣ ILLC, University of Amsterdam, The Netherlands
♠ The ALTA Institute, Dept. of CS & Technology, University of Cambridge, United Kingdom
`pushkarmishra@fb.com, hy260@cam.ac.uk, {m.deltredici, e.shutova}@uva.nl`

Abstract

Abuse on the Internet represents a significant societal problem of our time. Previous research on automated abusive language detection in Twitter has shown that community-based profiling of users is a promising technique for this task. However, existing approaches only capture shallow properties of online communities by modeling follower–following relationships. In contrast, working with graph convolutional networks (GCNs), we present the first approach that captures not only the structure of online communities but also the linguistic behavior of the users within them. We show that such a heterogeneous graph-structured modeling of communities significantly advances the current state of the art in abusive language detection.

1 Introduction

Matthew Zook (2012) carried out an interesting study showing that the racist tweets posted in response to President Obama's re-election were not distributed uniformly across the United States but instead formed clusters. This phenomenon is known as *homophily*: i.e., people, both in real life and online, tend to cluster with those who appear similar to themselves. To model homophily, recent research in abusive language detection on Twitter (Mishra et al., 2018a) incorporates embeddings for *authors* (i.e., users who have composed tweets) that encode the structure of their surrounding communities. The embeddings (called *author profiles*) are generated by applying a node embedding framework to an undirected unlabeled community graph where nodes denote the authors and edges the follower–following relationships amongst them on Twitter. However, these profiles do not capture the linguistic behavior of the authors and their communities and do not convey whether their tweets tend to be abusive or not.

In contrast, we represent the community of authors as a heterogeneous graph consisting of two types of nodes, authors and their tweets, rather than a homogeneous community graph of authors only. The primary advantage of such heterogeneous representations is that they enable us to model both community structure as well as the linguistic behavior of authors in these communities. To generate richer author profiles, we then propose a semi-supervised learning approach based on graph convolutional networks (GCNs) applied to the heterogeneous graph representation. To the best of our knowledge, our work is the first to use GCNs to model online communities in social media. We demonstrate that our methods provide significant improvements over existing techniques.

2 Related work

Supervised learning for abusive language detection was first explored by Spertus (1997) who extracted rule-based features to train their classifier. Subsequently, manually-engineered lexical–syntactic features formed the crux of most approaches to the task (Yin et al., 2009; Warner and Hirschberg, 2012). Djuric et al. (2015) showed that dense comment representations generated using *paragraph2vec* outperform bag-of-words features. Several works have since utilized (deep) neural architectures to achieve impressive results on a variety of abuse-annotated datasets (Nobata et al., 2016; Pavlopoulos et al., 2017a). Recently, the research focus has shifted towards extraction of features that capture behavioral and social traits of users. Pavlopoulos et al. (2017b) showed that including randomly-initialized user embeddings improved the performance of their RNN methods. Qian et al. (2018) employed LSTMs to generate inter and intra-user representations based on tweets, but they did not leverage community information.

Proceedings of NAACL-HLT 2019, pages 2145–2150
Minneapolis, Minnesota, June 2 - June 7, 2019.

3 Dataset

Following previous work (Mishra et al., 2018a), we experiment with a subset of the Twitter dataset compiled by Waseem and Hovy (2016). Waseem and Hovy released a list of $16,907$ tweet IDs along with their corresponding annotations,[1] labeling each tweet as *racist*, *sexist* or *neither (clean)*. Recently, Mishra et al. (2018a) could only retrieve $16,202$ of these tweets since some of them are no longer available. This is the dataset we use in our experiments. $1,939$ (12%) of $16,202$ tweets are *racist*, $3,148$ (19.4%) are *sexist*, and the remaining $11,115$ (68.6%) are *clean*. The tweets have been authored by a total of $1,875$ unique users. Tweets in the *racist* class come from 5 of the users, while those in the *sexist* class come from 527 of them.

4 Approach

4.1 Representing online communities

We create two different graphs: the first one is identical to the community graph of Mishra et al. (2018a) (referred to as the *community* graph). It contains $1,875$ nodes representing each of the authors in the dataset. Two authors/nodes are connected by a single undirected edge if either one follows the other on Twitter. There are 453 *solitary authors* in the graph who are neither followed by nor follow any other author in the dataset. This graph is homogeneous, i.e., it has nodes (and hence edges) of a single type only.

Our second graph is an extended version of the first (referred to as the *extended* graph) that additionally contains nodes representing the tweets of the authors. Specifically, in addition to the $1,875$ author nodes, the graph contains $16,202$ tweet nodes. Each tweet node is connected to a single author node, denoting that the tweet is elicited from that particular author. This graph is no longer homogeneous since it contains nodes and edges of two different types.

4.2 Generating author profiles

We first describe the approach of Mishra et al. (2018a) that learns author embeddings using *node2vec* (Grover and Leskovec, 2016); this serves as our baseline. We then move on to our semi-supervised approach based on graph convolutional networks (Kipf and Welling, 2017).

Node2vec. *Node2vec* extends the *word2vec* skip-gram model (Mikolov et al., 2013) to graphs in order to create low-dimensional embeddings for nodes based on their position and neighborhood. Specifically, for a given graph with nodes $V = \{v_1, v_2, \ldots, v_n\}$, *node2vec* aims to maximize the following log probability:

$$\sum_{v \in V} \log \mathrm{P}\left(N_s(v) \mid v\right)$$

where $N_s(v)$ denotes the neighbor set of node v generated using neighbor sampling strategy s. The framework utilizes two different strategies for sampling neighbor sets of nodes: Depth-First Sampling (DFS) and Breadth-First Sampling (BFS). The former captures the structural role of nodes, while the latter captures the local neighborhood around them. Two hyper-parameters control the overall contribution of each of these strategies. Following Mishra et al. (2018a), we initialize these parameters to their default value of 1 and set the embedding size and number of iterations to 200 and 25 respectively. Since *node2vec* cannot produce embeddings for nodes without edges, we map the *solitary authors* to a single zero embedding as done by Mishra et al.

Graph convolutional networks. We propose an approach for learning author profiles using GCNs applied to the *extended* graph. In contrast to *node2vec*, our method allows us to additionally propagate information with respect to whether tweets composed by authors and their communities are abusive or not. Specifically, as labels are available for a subset of nodes in our graph (i.e., the tweet nodes), we frame the task as a graph-based semi-supervised learning problem, allowing the model to distribute gradient information from the supervised loss on the labeled tweet nodes. This, in turn, allows us to create profiles for authors that not only capture the structural traits of their surrounding community but also their own linguistic behavior based on the types of tweets that they have composed.

We consider a graph $G = (V, E)$, where V is the set of nodes ($|V| = n$) and E is the set of edges. A denotes the adjacency matrix of G. We assume that A is symmetric ($A_{ij} = A_{ji}$), and that all nodes in G have self loops ($A_{ii} = 1$). The significance of these assumptions is explained in Kipf and Welling (2017). Let D be the diagonal degree matrix defined as $D_{ii} = \sum_j A_{ij}$, and $F \in \mathbb{R}^{n \times m}$

---

[1] `https://github.com/ZeerakW/hatespeech/`

be the input feature matrix that holds feature vectors of length m for the nodes in G. We can now recursively define the computation that takes place at the i^{th} *convolutional layer* of a k-layer GCN as:

$$O^{(i)} = \sigma\,(\,\widetilde{A}\,O^{(i-1)}\,W^{(i)}\,)$$

with the computation at the first layer being:

$$O^{(1)} = \sigma\,(\,\widetilde{A}\,F\,W^{(1)}\,)$$

Here, σ denotes an activation function; $\widetilde{A} = D^{-\frac{1}{2}}A\,D^{-\frac{1}{2}}$ is the normalized adjacency matrix; $W^{(i)} \in \mathbb{R}^{d_{i-1}\times d_i}$ is the weight matrix of the i^{th} convolutional layer; $O^{(i-1)} \in \mathbb{R}^{n\times d_{i-1}}$ represents the output from the preceding convolutional layer, where d_i is the number of hidden units in the i^{th} layer (note that $d_0 = m$, i.e., the length of the input feature vectors).

In our experiments, we apply a 2-layer GCN to the *extended* graph.[2] Specifically, our GCN performs the following computation, yielding a softmax distribution over the 3 classes in the dataset for each of the nodes:

$$O = softmax\,(\,\widetilde{A}\ ReLU\,(\,\widetilde{A}\,F\,W^{(1)}\,)\,W^{(2)})$$

We set the input feature vectors in F to be the binary bag-of-words representations of the nodes (following Kipf and Welling 2017); for author nodes, these representations are constructed over the entire set of their respective tweets. Note that F is row-normalized prior to being fed to the GCN. We set the number of hidden units in the first convolutional layer to 200 in order to extract 200-dimensional embeddings for author nodes so that they are directly comparable with those from *node2vec* . The number of hidden units in the second convolutional layer is set to 3 for the output $O \in \mathbb{R}^{n\times 3}$ of the GCN to be a softmax distribution over the 3 classes in the data.

The GCN is trained by minimizing the cross-entropy loss with respect to the labeled nodes of the graph. Once the model is trained, we extract 200-dimensional embeddings $E = \widetilde{A}\,F\,W^{(1)}$ from the first layer (i.e., the layer's output without activation). This contains embeddings for author nodes as well as tweet nodes. For our experiments on author profiles, we make use of the former.

4.3 Classification methods

We experiment with five different supervised classification methods for tweets in the dataset. The first three (LR, LR+AUTH, LR+EXTD) serve as our baselines,[3] and the last two with GCNs[4] are the methods we propose.

LR. This method is adopted from Waseem and Hovy (2016) wherein they train a logistic regression classifier on character n-grams (up to 4-grams) of the tweets. Character n-grams have been shown to be highly effective for abuse detection due to their robustness to spelling variations.

LR + AUTH. This is the state of the art method (Mishra et al., 2018a) for the dataset we are using. For each tweet, the profile of its author (generated by *node2vec* from the *community* graph) is appended onto the tweet's character n-gram representation for training the LR classifier as above.

LR + EXTD. This method is identical to LR + AUTH, except that we now run *node2vec* on the *extended* graph to generate author profiles. Intuitively, since *node2vec* treats both author and tweet nodes as the same and does not take into account the labels of tweets, the author profiles generated should exhibit the same properties as those generated from the *community* graph.

GCN. Here, we simply assign a label to each tweet based on the highest score from the softmax distribution provided by our GCN model for the (tweet) nodes of the *extended* graph.

LR + GCN. Identical to LR + EXTD, except that we replace the author profiles from *node2vec* with those extracted by our GCN approach.

5 Experiments and results

5.1 Experimental setup

We run every method 10 times with random initializations and stratified train–test splits. Specifically, in each run, the dataset is split into a randomly-sampled train set (90%) and test set (10%) with identical distributions of the 3 classes in each. In methods involving our GCN, a small part of the train set is held out as validation data to prevent over-fitting using *early-stopping* regularization. When training the GCN, we only have

---

[2] Stacking more layers does not improve results on the validation set further.

[3] The implementations of the baselines are taken from https://github.com/pushkarmishra/AuthorProfilingAbuseDetection.

[4] The code we use for our GCN models can be found at https://github.com/tkipf/gcn.

Method	Racism			Sexism			Overall		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
LR	**80.59**	70.62	75.28	83.12	62.54	71.38	83.18	75.62	78.75
LR + AUTH	77.95	78.35	78.15	87.28	78.41	82.61	85.26	83.28	84.18
LR + EXTD	77.95	78.35	78.15	87.02	78.73	82.67	85.17	83.33	84.17
GCN†	74.12	64.95	69.23	82.48	**82.22**	82.35	81.90	79.42	80.56
LR + GCN†	79.08	**79.90**	**79.49**	**88.24**	80.95	**84.44**	**86.23**	**84.73**	**85.42**

Table 1: The baselines (LR, LR + AUTH/EXTD) vs. our GCN approaches (†) on the racism and sexism classes. *Overall* shows the macro-averaged metrics computed over the 3 classes: *sexism*, *racism*, and *clean*.

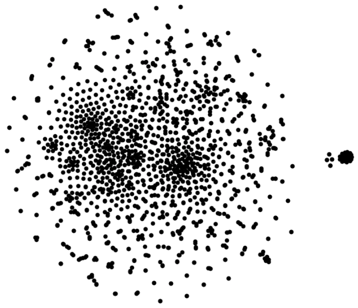
labeled tweet nodes for those tweets in the *extended* graph that are part of the train set. Our GCN is trained using the parameters from the original paper (Kipf and Welling, 2017): *Glorot* initialization (Glorot and Bengio, 2010), ADAM optimizer (Kingma and Ba, 2015) with a learning rate of 0.01, *dropout* regularization (Srivastava et al., 2014) rate of 0.5, 200 training epochs with an early-stopping patience of 10 epochs.

5.2 Results and analysis

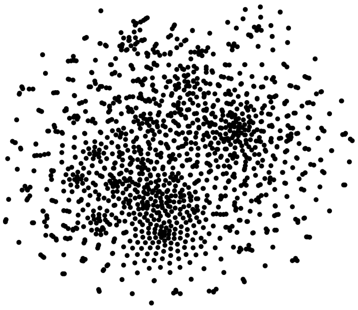
In Table 1, we report the mean precision, recall, and F_1 on the *racism* and *sexism* classes over the 10 runs. We further report the mean macro-averaged precision, recall, and F_1 for each method ('Overall') to investigate their overall performance on the data. LR + GCN significantly ($p < 0.05$ on paired t-test) outperforms all other methods. The author profiles from *node2vec* only capture the structural and community information of the authors; however, those from the GCN also take into account the (abusive) nature of the tweets composed by the authors. As a result, tweets like "*#MKR #mkr2015 Who is gonna win the peoples choice?*" that are misclassified as sexist by LR + AUTH (because their author is surrounded by others producing sexist tweets) are correctly classified as clean by LR + GCN.

GCN on its own achieves a high performance, particularly on the *sexism* class where its performance is typical of a community-based profiling approach, i.e., high recall at the expense of precision. However, on the *racism* class, its recall is hindered by the same factor that Mishra et al. (2018a) highlighted for their *node2vec*-only method, i.e., that racist tweets come from 5 unique authors only who have also contributed sexist or clean tweets. The racist activity of these authors is therefore eclipsed, leading to misclassifications of their tweets. LR + GCN alleviates this problem by incorporating character n-gram representations of the tweets, hence not relying solely on the linguistic behavior of their authors.

Figure 1 shows the t-SNE (van der Maaten and Hinton, 2008) visualizations of *node2vec* author profiles from the *community* and *extended* graphs. Both visualizations show that some authors belong to densely-connected communities while others are part of more sparse ones. The results from LR + AUTH and LR + EXTD have insignificant differences, further confirming that their author profiles have similar properties. In essence, *node2vec* is unable to gain anything more from the *extended* graph than what it does from the *community* graph.



(a) Author profiles from the *community* graph



(b) Author profiles from the *extended* graph

Figure 1: Visualizations of the *node2vec* author profiles from the *community* and *extended* graphs.

Figure 2 shows a t-SNE visualization of the author profiles generated using our GCN approach. Red dots denote the authors who are abusive (sexist or racist) according to our model (i.e., as per

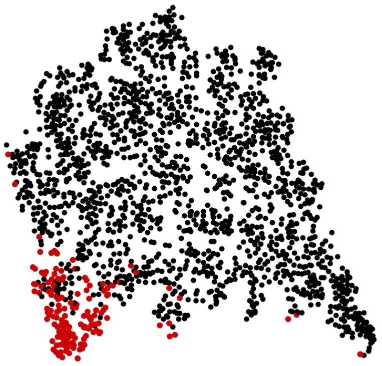


Figure 2: Visualization of the author profiles extracted from our GCN. Red dots represent the authors who are deemed abusive (racist or sexist) by the GCN.

the softmax outputs for the author nodes).[5] The red dots are mostly clustered in a small portion of the visualization, which corroborates the notion of homophily amongst abusive authors.

Despite the addition of improved author profiles, several abusive tweets remain misclassified. As per our analysis, many of these tend to contain URLs to abusive content but not the content itself, e.g., "*@MENTION: Logic in the world of Islam http://t.co/6nALv2HPc3*" and "*@MENTION Yes. http://t.co/ixbt0uc7HN*". Since Twitter shortens all URLs into a standard format, there is no indication of what they refer to. One possible way to address this limitation could be to append the content of the URL to the tweet; however this can lead to misclassifications in cases where the tweet is disagreeing with the URL. Another factor in misclassifications is the deliberate obfuscation of words and phrases by authors in order to evade detection, e.g., "*Kat, a massive c*nt. The biggest ever on #mkr #cuntandandre*". Mishra et al. (2018b) demonstrate in their work that character-based word composition models can be useful in dealing with this aspect.

6 Conclusions

In this paper, we built on the work of Mishra et al. (2018a) that introduces community-based profiling of authors for abusive language detection. We proposed an approach based on graph convolutional networks to show that author profiles that directly capture the linguistic behavior of authors along with the structural traits of their community significantly advance the current state of the art.

Acknowledgments

We would like to thank the anonymous reviewers for their useful feedback. Helen Yannakoudakis was supported by Cambridge Assessment, University of Cambridge.

References

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 29–30. Association for Computing Machinery.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. 2018a. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098. Association for Computational Linguistics.

Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2018b. Neural character-based composition models for abuse detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 1–10. Association for Computational Linguistics.

---

[5] Note that there are no such gold labels for authors in the dataset itself.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017a. Deeper attention to abusive user content moderation. In *Proceedings of EMNLP 2017*, pages 1125–1135. Association for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, Juli Bakagianni, and Ion Androutsopoulos. 2017b. Improved abusive comment moderation with user embeddings. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 51–55. Association for Computational Linguistics.

Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. 2018. Leveraging intra-user and inter-user representation learning for automated hate speech detection. In *Proceedings of the 2018 Conference of the NAACL: Human Language Technologies, Volume 2 (Short Papers)*, pages 118–123. Association for Computational Linguistics.

Ellen Spertus. 1997. Smokey: Automatic recognition of hostile messages. In *Proceedings of the 14th AAAI and 9th IAAI*, AAAI'97/IAAI'97, pages 1058–1065. AAAI Press.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the 2nd Workshop on Language in Social Media*, LSM '12, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0. In *Processings of the Content Analysis in the WEB 2.0*.

Matthew Zook. 2012. Mapping racist tweets in response to president obama's re-election. [Online; accessed 01 October 2018].

On the Importance of Distinguishing Word Meaning Representations: A Case Study on Reverse Dictionary Mapping

Mohammad Taher Pilehvar
Tehran Institute for Advanced Studies (TeIAS), Tehran, Iran
DTAL, University of Cambridge, Cambridge, UK
`mp792@cam.ac.uk`

Abstract

Meaning conflation deficiency is one of the main limiting factors of word representations which, given their widespread use at the core of many NLP systems, can lead to inaccurate semantic understanding of the input text and inevitably hamper the performance. Sense representations target this problem. However, their potential impact has rarely been investigated in downstream NLP applications. Through a set of experiments on a state-of-the-art reverse dictionary system based on neural networks, we show that a simple adjustment aimed at addressing the meaning conflation deficiency can lead to substantial improvements.

1 Meaning Conflation Deficiency

Words are often the most fine-grained meaning bearing components of NLP systems. As a standard practise, particularly for neural models, the input text is treated as a sequence of words and each word in the sequence is represent with a dense distributional representation (word embedding). Importantly, this setting ignores the fact that a word can be polysemous, i.e., it can take multiple (possibly unrelated) meanings. Representing a word with all its possible meanings as a single point (vector) in the embedding space, the so-called meaning conflation deficiency (Camacho-Collados and Pilehvar, 2018), can hinder system's semantic effectiveness.

To address this deficiency, many techniques have been put forward over the past few years, the most prominent of which is sense representation or multi-prototype embedding (Schütze, 1998; Reisinger and Mooney, 2010). However, as a general trend, these representations are usually evaluated either on generic benchmarks, such as word similarity, or on sense-centered tasks such as Word Sense Disambiguation, leaving their potential impact on downstream word-based systems unknown. In this paper, we provide an analysis to highlight the importance of addressing the meaning conflation deficiency. Specifically, we show how distinguishing different meanings of a word can facilitate a more accurate semantic understanding of a state-of-the-art reverse dictionary system, reflected by substantial improvements in recall and generalisation power.

2 Reverse Dictionary

Reverse dictionary, conceptual dictionary, or concept lookup is the task of returning a word given its description or definition (Brown and McNeill, 1966; Zock and Bilac, 2004). For example, given "a crystal of snow", the system has to return the word *snowflake*. The task is closely related to the "tip of the tongue" problem where an individual recalls some general features about a word but cannot retrieve that from memory. Therefore, a reverse dictionary system can be particularly useful to writers and translators when they cannot recall a word in time or are unsure how to express an idea they want to convey.

2.1 Evaluation framework

Our experiments are based on the reverse dictionary model of Hill et al. (2016) which leverages a standard neural architecture in order to map dictionary definitions to representations of the words defined by those definitions. Specifically, they proposed two neural architectures for mapping the definition of word t to its word embedding $\mathbf{e}_t$. Let $\mathcal{D}_t$ be the sequence of words in t's definition, i.e., $\mathcal{D}_t = \{w_1, w_2, \ldots, w_n\}$, with their corresponding embeddings $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$. The two models differ in the way they process $\mathcal{D}_t$. In the bag-of-words (**BoW**) model, $\mathcal{D}_t$ is taken as a bag

Proceedings of NAACL-HLT 2019, pages 2151–2156
Minneapolis, Minnesota, June 2 - June 7, 2019.

of words, i.e., the representation of the definition is encoded by adding the word embeddings of all its content words, i.e., $\sum_{i=1}^{n} \mathbf{v}_i$. The model learns, using a fully-connected layer, a matrix for transforming the encoded representation to the target word's embedding $\mathbf{e}_t$. The BoW model is not sensitive to the order of words in $\mathcal{D}_t$. This might be crucial for an accurate semantic understanding. The Recurrent Neural Network (**RNN**) model alleviates this issue by encoding the input sequence using an LSTM architecture (Hochreiter and Schmidhuber, 1997). Similarly to the BoW model, a dense layer maps the encoded representation to the target word's embedding.

In both cases, the goal is to map a given definition to the corresponding target word's embedding $\mathbf{e}_t$, computed using Word2vec (Mikolov et al., 2013) and independently from the training of the main model. Two cost functions were tested: (1) the **cosine** distance between the estimated point in the target space ($\hat{\mathbf{e}}_t$) and $\mathbf{e}_t$, and (2) the **rank** loss which contrast the choice of $\mathbf{e}_t$ with a random choice for a randomly-selected word from the vocabulary other than t.

The reverse dictionary system takes advantage of a standard architecture which has proven effective in various NLP tasks. However, similarly to many other word-based models, the system ignores that the same word can have multiple (potentially unrelated) meanings. In fact, it tries to map multiple definitions, with different semantics, to the same point in the target space. For instance, the three semantically unrelated definitions of *crane*: "lifts and moves heavy objects", "large long-necked wading bird", and "a small constellation in the southern hemisphere" will have similar semantic interpretation by the system. This word-level meaning conflation can hamper the ability of the system in learning an accurate mapping function. In what follows in this paper, we will illustrate how a simple sense level distinction can facilitate a more accurate semantic understanding for the reverse dictionary system, hence leading to significant performance improvements.

2.2 Sense Integration

Let t be an ambiguous word with three meanings; hence, three distinct definitions $\mathcal{D}_{t_1}$, $\mathcal{D}_{t_2}$, and $\mathcal{D}_{t_3}$. The original model of Hill et al. (2016) maps all these definitions to $\mathbf{e}_t$. We mitigate the meaning conflation deficiency through a sense-specific mapping function that obtains distinct interpretations for individual definition, hence mapping them to different points in the target space: $\mathbf{s}_{t_1}$, $\mathbf{s}_{t_2}$, and $\mathbf{s}_{t_3}$. Specifically, in our experiments we leveraged DeConf (Pilehvar and Collier, 2016). DeConf is a WordNet-based sense representation technique which receives a set of pre-trained word embeddings and generates embeddings for individual word senses in the same semantic space, hence generating a combined space of words and word senses.

DeConf performs a set of random walks on WordNet's semantic network and extracts for each sense a set of *sense biasing* words $\mathcal{B}_{\mathbf{s}}$. A sense biasing word for the i^{th} meaning of a target word t is a semantically related word to that specific sense of the word ($\mathbf{s}_{t_i}$). For each word sense in WordNet we obtain the corresponding $\mathcal{B}_{\mathbf{s}}$. Then, the embedding for a specific word sense $\mathbf{s}$ is computed as:

$$\mathbf{s} = ||\, \mathbf{e}_w + \alpha \sum_{b \in \mathcal{B}_{\mathbf{s}}} \exp(-\delta_i)\, \mathbf{e}_b||, \qquad (1)$$

where δ is a decay parameter and $\mathbf{e}_w$ is the embedding of corresponding lemma of sense $\mathbf{s}$. In our experiments, as for word embeddings we used the 300-dimensional Word2vec embeddings, trained on the Google News corpus.[1] The same set was used as input to DeConf. As a result of this process, around 207K additional word senses were introduced in the space for the 155K unique words in WordNet 3.0.

2.2.1 Supersenses

It is widely acknowledged that sense distinctions in WordNet inventory are too fine-grained for most NLP applications (Hovy et al., 2013). For instance, for the noun *star*, WordNet 3.0 lists eight senses, among which two *celestial body* senses (as an "astronomical object" and that "visible, as a point of light, from the Earth"), and three *person* senses ("skillful person", "lead actor", and "performing artist"). This fine level of sense distinction is often more than that required by the target downstream application (Rüd et al., 2011; Severyn et al., 2013; Flekova and Gurevych, 2016). In our experiments, we used WordNet's lexicographer files (lexnames[2]) in order to reduce sense granularity. Created by the curators of WordNet

---

[1] `https://code.google.com/archive/p/word2vec/`

[2] `https://wordnet.princeton.edu/man/lexnames.5WN.html`

			WN-seen		**WN-unseen**		**Concept Mapping**	
			top-10	*top-100*	*top-10*	*top-100*	*top-10*	*top-100*
Supersense	RNN	cosine	0.656	0.824	0.150	0.310	0.230	0.480
		ranking	0.694	0.836	0.162	0.352	0.335	0.630
	BoW	cosine	0.642	0.820	0.250	0.416	0.280	0.590
		ranking	0.706	**0.872**	**0.310**	**0.474**	**0.390**	**0.735**
Sense	RNN	cosine	**0.742**	0.854	0.164	0.336	0.275	0.505
		ranking	0.668	0.840	0.180	0.372	0.325	0.615
	BoW	cosine	0.678	0.826	0.290	0.456	0.300	0.620
		ranking	0.692	0.848	0.292	0.470	0.380	**0.735**
Word	RNN	cosine	0.462	0.652	0.056	0.162	0.215	0.400
		ranking	0.534	0.728	0.086	0.188	0.190	0.475
	BoW	cosine	0.446	0.652	0.136	0.264	0.175	0.465
		ranking	0.562	0.740	0.160	0.292	0.320	0.600
Baseline	-	-	0.104	0.346	0.054	0.158	0.065	0.300

Table 1: Accuracy performance (@10/100) of the original (word-based) reverse dictionary system and its sense- and supersense-based improvements on different datasets. See Section 2.1 for system configurations.

during its development, these files organize WordNet synsets into 45 groups (such as food, animal, event, and emotion) according to their syntactic and logical properties. These groupings are usually referred to as *supersenses*. Using supersenses, the *celestial* and *person* meanings of *star* are grouped into two main groups. A supersense embedding e_{ss} in our experiments is simply computed as a normalized average (centroid) of its contained sense embeddings, i.e., $e_{ss} = ||\sum_{s \in ss} e_s||$. This reduces the average number of senses for polysemous words in WordNet from 2.9 to 1.8.

3 Experiments

We carried out evaluations on the three reverse dictionary datasets created by Hill et al. (2016): WordNet definitions and "single-sentence descriptions" written for a set of frequent words (*concept mapping*). They proposed two different versions of the WordNet dataset: *WN-seen*, in which a test instance is already observed during training, and *WN-unseen*, in which test instances are excluded from the training data. The former dataset is targeted at evaluating the ability of the system to recall a previously encoded information.

We experimented with three variants of the reverse dictionary system: the original word-based model and the two proposed sense-based variants, based on WordNet senses and supersenses.[3]

Table 1 reports accuracy performance for four different configurations of the system (BoW and RNN definition composition and cosine and ranking loss; cf. Section 2.1) on the three datasets. In the last row, we also report results for the unsupervised baseline of Hill et al. (2016) which adds the embedding of words in the input definition and finds the nearest embedding in the target space.

Results reported in the Table clearly highlight that addressing the meaning conflation deficiency in the system has led to significant performance improvements (word vs. sense and supersense settings). This is observed consistently across all the three datasets and for both sense-based models. The better semantic understanding of the system is reflected by its better recall of seen test instances (WN-seen) and better generalisation to unseen and out-of-domain data (WN-unseen and concept mapping). The absolute top-10 accuracy improvements of the ranking-BoW supersense model over the best corresponding word-based configurations are: 14.4% (WN-seen), 15% (WN-unseen), and 7% (concept mapping).

Among the two proposed systems, supersenses prove to be more effective, suggesting that the fine-grained sense distinctions in WordNet might not be necessary for an accurate reverse dic-

---

[3] The experiments are based on the implementation available at `https://github.com/fh295/DefGen2`.

tionary mapping, corroborating previous findings (Flekova and Gurevych, 2016). Our results are also in line with the findings of Hill et al. (2016) that the reverse dictionary system performs best with the bag-of-words (BoW) input encoding and the ranking loss. One of the fundamental differences between the two input encodings lies in their sensitiveness to order: RNNs are sensitive to the order of words in a given sequence whereas permuting words in the sequence does not alter BoW's encoding. Hill et al. (2016) suggested that it is often possible to retrieve a concept even if the words in its corresponding definition are shuffled. This can partly explain the strikingly good relative performance of the BoW model.

4 Analysis

During our analysis of system outputs, we observed many examples in which the word-based model was unable to retrieve an ambiguous word since the definition was referring to one of its less frequent meanings. For instance, the word *dressing* might refer to different concepts such as "getting dressed" or "savory dressing for salads". Having a conflated understanding of *dressing*, the word-based model was unable to retrieve the salad meaning.

dressing *savory dressings for salads; basically of two kinds: either the thin french or vinaigrette type or the creamy mayonnaise type*

word: mayonnaise, marinade, sauce
sense: *dressing*, mayonnaise, mayo
baseline: or, either, type

Other similar examples include infrequent senses of *party*, defined as "an organization to gain political power", and *partition*, defined as "a vertical structure that divides or separates". In both cases, the sense-based model improves the original word-based one, in which the system is unable to retrieve the intended word. Numerous such examples were observed during our analysis of the results, highlighting the important limitation of word-based models for their inherent bias towards more frequent usages.

Moreover, as a side benefit, sense embeddings provide parts of speech distinction, unlike common pre-trained word embeddings which conflate all parts of speech to a single token. For instance, the word-based model is unable to recall the nominal *bear* because it has a conflated understanding of the word which includes all its senses, particularly the dominant verb meaning.[4]

bear *massive plantigrade carnivorous or omnivorous mammals with long shaggy coats and strong claws*

word: critter, rabbit, squirrel, wolf
sense: *bear*, mustelid, bruin
baseline: carnivorous, omnivorous.

The same applies to the "open land" meaning of *common*, which is a less frequent (nominal) meaning of the word which is usually used as an adjective for concepts such as "ordinary" or "usual".

common *a piece of open land for recreational use in an urban area*

word: park, plaza, entryway, courtway
sense: park, green, *common*
baseline: for, area, in, recreational

Additionally, word embeddings are insensitive to fine-grained semantic distinctions, such as antonymy, due to their construction nature. However, the sense representations used in our experiments (DeConf) were constructed by exploiting the knowledge encoded in WordNet. Hence, they benefit from the rich semantic and ontological knowledge provided by the resource (such as relation types). Some of the improvements can be attributed to this property of sense embeddings.

unanticipated *not anticipated*

word: unavoidable, inevitable, plausible
sense: unforeseen, *unanticipated*, unpredicted
baseline: not, anticipated, expected

However, there are cases in which the word-based model provided more accurate results. For instance:

service *work done by one person or group that benefits another*

word: *service*, caring
sense: organisation, dependant, programme

Our analysis showed that most of these errors were due to fine-grained sense distinctions in WordNet or obscure meanings. For instance, one of the senses[5] of *organisation* is semantically re-

---

[4] In our analysis, we found that improvements are mostly due to addressing semantic conflation rather than ambiguities in parts of speech.

[5] The 6th sense of *organisation* in WordNet 3.0, defined as "the activity or result of distributing or disposing persons or things properly or methodically".

lated (also close in WordNet's graph) to the meaning of *service* in the example. This would suggest the need for more accurate sense representations and highlight the fact that the fine-granularity of senses should be better adjusted to the underlying task. Moreover, it corroborates our finding that the coarse-grained supersenses are more suitable in the task of reverse dictionary mapping. We leave the experiments with other sense representation techniques to future work.

5 Related work

Sense representations address the meaning conflation deficiency of their word-based counterparts by computing distinct representations for individual meanings of words, usually referred to as word senses. Sense distinctions might be given by an external sense inventory, such as WordNet (Fellbaum, 1998). An inventory-based sense representation technique exploits the knowledge encoded in the resource to construct representations (Rothe and Schütze, 2015; Jauhar et al., 2015; Pilehvar and Collier, 2016). Alternatively, senses can be automatically induced in an unsupervised manner by analyzing the diversity of contexts in which a word appears (Schütze, 1998; Reisinger and Mooney, 2010; Huang et al., 2012; Neelakantan et al., 2014; Guo et al., 2014; Šuster et al., 2016).

Regardless of how senses are obtained, the integration of sense representations into NLP systems is not a straightforward process. Hence, they have often been evaluated on artificial tasks such as word similarity. This is also due to lack of suitable evaluation benchmarks for sense representation techniques. Pilehvar and Camacho-Collados (2019) recently proposed a dataset, The Word-in-Context (WiC), which provides a challenging, yet reliable, benchmark for the purpose.

Few attempts have been made at the integration of sense representation into downstream applications. Li and Jurafsky (2015) experimented with unsupervised sense representations in tasks such as part-of-speech tagging and named entity recognition, with mixed results. Also related to our work are the proposals of Flekova and Gurevych (2016) and Pilehvar et al. (2017) to disambiguate the input text and replace word embeddings with sense embeddings for the intended senses. Our results for supersenses corroborates the findings of Pilehvar et al. (2017) who found reducing fine-granularity of senses beneficial to some settings.

A more recent branch of research investigates the construction of dynamic word embeddings that can adapt according to the context in which they appear (Peters et al., 2018; Devlin et al., 2018). One of the objectives of this research has been to bypass the integration difficulties of sense representations into downstream models. These so-called contextualised word embeddings can easily be replaced with conventional static word embeddings in neural-based NLP systems. This integration has proven beneficial to a wide range of NLP applications. Pilehvar and Camacho-Collados (2019) carried out an analysis on the sense distinguishing capability of contextualised embeddings, showing that, despite their successful application to downstream applications, these embeddings are not very powerful in capturing distinct meanings of words.

6 Conclusions

We provided an analysis on the impact of addressing the meaning conflation deficiency of word embeddings on the performance of a downstream NLP application, i.e., reverse dictionary mapping. Through a set of experiments we showed that a simple migration from words to senses can significantly improve the ability of the system in semantic understanding, leading to consistent performance boost. In future work, we plan to evaluate sense integration in other NLP applications, such as Machine Translation, in the light of (Liu et al., 2018), and question answering.

References

R. Brown and D. McNeill. 1966. The tip of the tongue phenomenon. *Journal of Verbal Learning and Verbal Behavior* 5:325–337.

José Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research* 63:743–788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *Proceedings of the 54th Annual Meeting of the Association* *for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2029–2041.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*. pages 497–507.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics* 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence* 194:2–27.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*. Jeju Island, Korea, pages 873–882.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*. Denver, Colorado, pages 683–693.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of EMNLP*. Lisbon, Portugal, pages 683–693.

Frederick Liu, Han Lu, and Graham Neubig. 2018. Handling Homographs in Neural Machine Translation. In *Proceedings of NAACL*. New Orleans, LA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*. Doha, Qatar, pages 1059–1069.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*. New Orleans, LA, USA.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of NAACL-HLT*.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a seamless integration of word senses into downstream nlp applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1857–1869.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of EMNLP*. Austin, TX, pages 1680–1690.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of ACL*. pages 109–117.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*. Beijing, China, pages 1793–1803.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of ACL-HLT*. Portland, Oregon, USA, pages 965–975.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics* 24(1):97–124.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *Proceedings of ACL (2)*. Sofia, Bulgaria, pages 714–718.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of NAACL-HLT*. San Diego, California, pages 1346–1356.

Michael Zock and Slaven Bilac. 2004. Word lookup on the basis of associations: From an idea to a roadmap. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*. ElectricDict '04, pages 29–35.

Factorising AMR generation through syntax

Kris Cao and **Stephen Clark**
Computer Laboratory
University of Cambridge
United Kingdom
{kc391,sc609}@cam.ac.uk

Abstract

Generating from Abstract Meaning Representation (AMR) is an underspecified problem, as many syntactic decisions are not constrained by the semantic graph. To explicitly account for this underspecification, we break down generating from AMR into two steps: first generate a syntactic structure, and then generate the surface form. We show that decomposing the generation process this way leads to state-of-the-art single model performance generating from AMR without additional unlabelled data. We also demonstrate that we can generate meaning-preserving syntactic paraphrases of the same AMR graph, as judged by humans.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic annotation framework which abstracts away from the surface form of text to capture the core 'who did what to whom' structure. As a result, generating from AMR is underspecified (see Figure 1 for an example). Single-step approaches to AMR generation (Flanigan et al., 2016; Konstas et al., 2017; Song et al., 2016, 2017) therefore have to decide the syntax and surface form of the AMR realisation in one go. We instead explicitly try and capture this syntactic variation and factor the generation process through a syntactic representation (Walker et al., 2001; Dušek and Jurcicek, 2016; Gardent and Perez-Beltrachini, 2017; Currey and Heafield, 2018).

First, we generate a delexicalised constituency structure from the AMR graph using a syntax model. Then, we fill out the constituency structure with the semantic content in the AMR graph using a lexicalisation model to generate the final surface form. Breaking down the AMR generation process this way provides us with several advantages: we disentangle the variance caused by the choice of syntax from that caused by the choice of words. We can therefore realise the same AMR graph with a variety of syntactic structures by sampling from the syntax model, and deterministically decoding using the lexicalisation model. We hypothesise that this generates better paraphrases of the reference realisation than sampling from a single-step model.

We linearise both the AMR graphs (Konstas et al., 2017) and constituency trees (Vinyals et al., 2015b) to allow us to use sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) for the syntax and lexicalisation models. Further, as the AMR dataset is relatively small, we have issues with data sparsity causing poor parameter estimation for rarely seen words. We deal with this by anonymizing named entities, and including a *copy mechanism* (Vinyals et al., 2015a; See et al., 2017; Song et al., 2018) into our decoder, which allows open-vocabulary token generation.

We show that factorising the generation process in this way leads to improvements in AMR generation, setting a new state of the art for single-model AMR generation performance training only on labelled data. We also verify our diverse generation hypothesis with a human annotation study.

2 Data

Abstract Meaning Reprentation Abstract Meaning Representation is a semantic annotation formalism which represents the meaning of an English utterance as a rooted directed acyclic graph. Nodes in the graph represent entities, events, properties and states mentioned in the text, while leaves of the graph label the nodes with concepts (which do not have to be aligned to spans in the text). Re-entrant nodes correspond to coreferent entities. Edges in the graph represent

Proceedings of NAACL-HLT 2019, pages 2157–2163
Minneapolis, Minnesota, June 2 - June 7, 2019.

```
(g / give-01
  :ARG0 (i / I)
  :ARG1 (b / ball)
  :ARG2 (d / dog))
```

```
give :arg0 i :arg1 ball :arg2 dog
```

I [gave]$_{\text{VP}}$ [the dog]$_{\text{NP}}$ [a ball]$_{\text{NP}}$
I [gave]$_{\text{VP}}$ [the ball]$_{\text{NP}}$ [to a dog]$_{\text{PP}}$

Figure 1: An example AMR graph, with variable names and verb senses, followed by the input to our system after preprocessing, and finally two sample realisations different in syntax.

relations between entities in the text. See Figure 1 for an example of an AMR graph, together with sample realisations.

Konstas et al. (2017) outline a set of preprocessing procedures for AMR graphs to both render them suitable for sequence-to-sequence learning and to ameliorate data sparsity; we follow the same pipeline. We train our models on the two most recent AMR releases. LDC2017T10 has roughly 36k training sentences, while LDC2015E86 is about half this size. Both share dev and test sets, facilitating comparison.

Constituency syntax While there are many syntactic annotation formalisms, we use delexicalised Penn treebank-style constituency trees to represent syntax. Constituency trees have the advantage of a well-defined linearization order compared to dependency trees. Further, constituency trees may be easier to realise, as they effectively correspond to a bracketing of the surface form.

Unfortunately, AMR annotated data does not come with syntactic annotation. We therefore parse the training and dev splits of both corpora with the Stanford parser (Manning et al., 2014) to provide silver-standard reference parse trees. We then delexicalise the parse trees by trimming the trees of the surface words; after this stage, the leaves of the tree are the preterminal POS tags. After this, we linearise the delexicalised constituency trees with depth-first traversal, following Vinyals et al. (2015b).

3 Model implementation and training

3.1 Model details

We wish to estimate $P(Y, Z|X)$, the joint probability of a parse Y and surface form Z given an AMR graph X. We model this in two parts, using the chain rule to decompose the joint distribution. The first model, which we call the syntax model, approximates $P(Y|X)$, the probability of a particular syntactic structure for a meaning representation. The second is $P(Z|X, Y)$, the lexicalisation model. This calculates the probability of a surface realisation given a parse tree and an AMR graph. We implement both as recurrent sequence-to-sequence models.

As we are able to linearise both the AMR graph and the parse tree, we use LSTMs (Hochreiter and Schmidhuber, 1997) both as the encoder and the decoder of our seq2seq models. Given an input sequence $X_1, \ldots, X_n$, which can either be an AMR graph or a parse tree, we first embed the tokens to obtain a dense vector representation of each token $x_1, \ldots, x_n$. Then we feed this into a stacked bidirectional LSTM encoder to obtain contextualised representations of each input token c_i. As far as possible, we share parameters between our two models. Concretely, this means that the syntax model uses the same AMR and parse embeddings, and AMR encoder, as the lexicalisation model. We find that this speeds up model inference, as we only have to encode the AMR sequence once for both models. Further, it regularises the joint model by reducing the number of parameters.

In our decoder, we use the dot-product formulation of attention (Luong et al., 2015): the attention potentials a_i at timestep t are given by

$$a_i = h_{t-1}^T W_{att} c_i$$

where h_{t-1} is the decoder hidden state at the previous timestep, and c_i is the context representation at position i given by the encoder. The attention weight w_i is then given by a softmax over the attention potentials, and the overall context representation s_t is given by $\sum w_i c_i$. The syntax model only attends over the input AMR graph; the linearisation model attends over both the input AMR and syntax tree independently, and the resulting context representation s_t is given by the concatenation of the AMR context representation and the syntax tree context representation (Libovický and Helcl, 2017).

We use s_t to augment the input to the LSTM: $\widetilde{y}_t = W_{in} \tanh([y_t; s_t])$. Then the LSTM hidden and cell state are updated according to the LSTM equations: $h_t, c_t = LSTM(h_{t-1}, c_{t-1}, \widetilde{y}_t)$. Finally, we again concatenate s_t to h_t before calculating the logits over the distribution of tokens:

$$\widetilde{h}_t = \tanh(W_{out}[h_t; s_t]) \quad (1)$$

$$p(y_t|y_{<t}) = \mathrm{softmax}(W\widetilde{h}_t) \quad (2)$$

For the syntax model, we further constrain the decoder to only produce valid parse trees; as we build the parse tree left-to-right according to a depth-first traversal, the permissible actions at any stage are to open a new constituent, produce a terminal (i.e. a POS tag), or close the currently open constituent. We implement this constraint by setting the logits of all impermissible actions to negative infinity before taking the softmax. We find that this improves both training speed and final model performance, as we imbue the decoder with an intrinsic bias towards producing well-formed parse trees.

3.2 Generation with a copy mechanism

Despite the preprocessing procedures referred to in Section 2, we found that the lexicalisation model still had trouble with out-of-vocabulary words, due to the small size of the training corpus. This led to poor vocabulary coverage on the unseen test portions of the dataset. On closer inspection, many out-of-vocabulary words in the dev split are open-class nouns and verbs, which correspond to concept nodes in the AMR graph. We therefore incorporate a copy mechanism (Vinyals et al., 2015a; See et al., 2017) into our lexicalisation model to make use of these alignments.

We implement this by decomposing the word generation probability into a weighted sum of two terms. One is the vocabulary generation term. This models the probability of generating the next token from the model vocabulary, and is calculated in the same way as the base model. The other is a copy term, which calculates the probability of generating the next token by copying a token from the input. This uses the attention distribution over the input tokens calculated in the decoder to decide which input token to copy. The weighting between these two terms is calculated as a function of the current decoder input token, the decoder hidden state, and the AMR and parse context vectors. To sum up, the per-word generation probability in the decoder is given by

$$p(y_t|y_{<t}) = (1-\theta_t)p_{lex}(y_t|y_{<t}) + \theta_t \sum_{i:X_i=y_t} w_i \quad (3)$$

where $p_{lex}(y_t|y_{<t})$ is as in Equation 2 and w_i is the attention weight on the input token X_i. θ is the weighting between the generation term and the copy term: this is implemented as a 2-layer MLP.

3.3 Model training procedures

The AMR training corpus, together with the automatically derived parse trees, give us aligned triples of AMR graph, parse tree and realisation. We train our model to minimise the sum of the parse negative log-likelihood from the syntax model and the text negative log-likelihood from the lexicalisation model. We use the ADAM optimizer (Kingma and Ba, 2015) with batch size 40 for 200 epochs. We evaluate model BLEU score on the dev set during training, and whenever this did not increase after 5 epochs, we multiplied the learning rate by 0.8. We select the model with the highest dev BLEU score during training as our final model.

We apply layer normalization (Ba et al., 2016) to all matrix multiplications inside our network, including in the LSTM cell, and drop out all non-recurrent connections with probability 0.5 (Srivastava et al., 2014). We also drop out recurrent connections in both encoder and decoder LSTMs with probability 0.3, tying the mask across timesteps as suggested by Gal and Ghahramani (2016). All model hidden states are size 500, and token embeddings are size 300. Word embeddings are initialised with pretrained `word2vec` embeddings (Mikolov et al., 2013). We replace words with count 1 in the training corpus with the UNK token with probability 0.5, and replace POS tags in the parse tree and AMR concepts with the UNK token with probability 0.1 regardless of count.

Decoding from our model During test time, we would like to estimate

$$\arg\max_Z \sum_Y P(Z, Y|X) \quad (4)$$

the most likely text realisation of an AMR, marginalising out over the possible parses. To do this, we heuristically find the n best parses $Y_1, \ldots, Y_n$ from the syntax model, generate a realisation Z_i for each parse Y_i, and take the highest scoring parse-realisation pair as the model output.

We use beam search with width 2 for both steps, removing complete hypotheses from the active beam and appending them to a k-best list. We terminate search after a predetermined number of steps, or if there are no active beam items left. After termination, if $k > n$, we return the top n items of the k-best list; otherwise we return additional

Model	Unlabelled F1	Labelled F1
Text-to-parse	87.5	85.8
AMR-to-parse	60.4	54.8
Unconditional	38.5	31.7

Table 1: Parsing scores on LDC2017T10 dev set.

Model	# good realisations
Syntax-aware model	1.52
Baseline s2s	1.19

Table 2: Average number of acceptable realisations out of 3. The difference is significant with $p < 0.001$.

items from the beam. In our experiments, we find that considering realisations of the 2 best parses (i.e. setting $n = 2$ above) gives the highest BLEU score on the dev set.

4 Experiment 1: AMR and syntax

We first investigate how much information AMR contains about possible syntactic realisations. We train two seq2seq models of the above architecture to predict the delexicalised constituency tree of an example given either the AMR graph or the text. We then evaluate both models on labelled and unlabelled F1 score on the dev split of the corpus. As neither model is guaranteed to produce trees with the right number of terminals, we first run an insert/delete aligner between the predicted and reference terminals (i.e. POS tags) before calculating span F1s. We also report the results of running our aligner on the most probable parse tree as estimated by an unconditional LSTM as a baseline both to control for our aligner and also to see how much extra signal is in the AMR graph. The results in Table 1 show that predicting a syntactic structure from an AMR graph is a much harder task than predicting from the text, but there is information in the AMR graph to improve over a blind baseline.

5 Experiment 2: Generating natural language from AMR

Table 3 shows the results of our model on the AMR generation task. We evaluate using BLEU score (Papineni et al., 2002) against the reference realisations. As a baseline, we train a straight AMR-to-text model with the same architecture as above to control for the extra regularisation in our model compared to previous work. Our results show that adding syntax into the model dramatically boosts performance, resulting in state-of-the-art single model performance on both datasets without using external training data.

Model	Dev BLEU	Test BLEU
Trained on LDC2017T10		
Our model	**26.1**	**26.8**
Our model + oracle parse	57.5	-
Baseline s2s + copy	23.7	23.5
Beck et al. (2018)	-	23.3
Trained on LDC2015E86		
Our model	**23.6**	**23.5**
Our model + oracle parse	53.1	-
Konstas et al. (2017)	21.7	22.0
Song et al. (2018)	22.8	23.3
Trained on LDC2015E86 or earlier + additional unlabelled data		
Song et al. (2018)	-	33.0
Konstas et al. (2017)	33.1	33.8
Pourdamghani et al. (2016)	27.2	26.9
Song et al. (2017)	25.2	25.6

Table 3: BLEU results for generation.

As an oracle experiment, we also generate from the realisation model conditioned on the ground truth parse. The outstanding result here – BLEU scores in the 50s – demonstrates that being able to predict the gold reference parse tree is a bottleneck in the performance of our model. However, given the inherent difficulty of predicting a single syntax realisation (cf. Section 4), we suspect that there is an intrinsic limit to how well generating from an AMR graph can replicate the reference realisation.

We further note that we do not use models tailored to graph-structured data or character-level features as in Song et al. (2018); Beck et al. (2018), or additional unlabelled data to perform semi-supervised learning (Konstas et al., 2017). We believe that we can improve our results even further if we use these techniques.

6 Experiment 3: Generating varied realisations

Our model explicitly disentangles variation caused by syntax choice from that caused by lexical choice. This means that we can generate diverse realisations of the same AMR graph by sampling from the syntax model and deterministically decoding from the realisation model. We hypothesise that this procedure generates more meaning-preserving realisations than just sampling from a straight AMR-to-text model, which can result in incoherent output (Cao and Clark, 2017).

We selected the first 50 AMR graphs in the dev set on linearised length between 15 and 40 with coherent reference realisations and generated 5 different realisations with our joint model and our baseline model. For our joint model, we first sampled 3 parse structures from the syntax model with temperature 0.3. This means we divide the per-timestep logits of the syntax decoder by 0.3; this serves to sharpen the outputs of the syntax model and constrains the sampling process to produce relatively high-probability syntactic structures for the given AMR. Then, we realised each parse deterministically with the lexicalisation model. For the baseline model, we sample 3 realisations from the decoder with the same temperature. This gave us 100 examples in total.

We then crowdsourced acceptability judgments for each example from 100 annotators: we showed the reference realisation of an AMR graph, together with model realisations, and asked each annotator to mark all the grammatical realisations which have the same meaning as the reference realisation. Each annotator was presented 30 examples selected randomly. Our results in Table 2 show that the joint model can generate more meaning-preserving realisations compared to a syntax-agnostic baseline. This shows the utility of separating out syntactic and lexical variation: we model explicitly meaning-preserving invariances, and can therefore generate better paraphrases.

7 Conclusions and further work

We present an AMR generation model that factors the generation process through a syntactic decision, and show that this leads to improved AMR generation performance. In addition, we show that separating the syntactic decisions from the lexicalisation decisions allows the model to generate higher quality paraphrases of a given AMR graph.

In future work, we would like to integrate a semantic parser into our model (Yin et al., 2018). Annotating data with AMR is expensive, and existing AMR treebanks are small. By integrating a component which parses into AMR into our model, we can do semi-supervised learning on plentiful unannotated natural language sentences, and improve our AMR generation performance even further. In addition, we would be able to generate text-to-text paraphrases by parsing into AMR first and then carrying out the paraphrase generation procedure described in this paper (Iyyer et al., 2018). This opens up scope for data augmentation for downstream NLP tasks, such as machine translation.

Acknowledgements

The authors would like to thank Amandla Mabona and Guy Emerson for fruitful discussions. KC is funded by an EPSRC studentship.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283. Association for Computational Linguistics.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 182–187. Association for Computational Linguistics.

Anna Currey and Kenneth Heafield. 2018. Multi-source syntactic neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2966, Brussels, Belgium. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Claire Gardent and Laura Perez-Beltrachini. 2017. A statistical, grammar-based approach to microplanning. *Computational Linguistics*, 43(1):1–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Compututation*, 9(8):1735–1780.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–202, Vancouver, Canada. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2084–2089. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing*

Systems 28, pages 2692–2700. Curran Associates, Inc.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Marilyn A. Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–765, Melbourne, Australia. Association for Computational Linguistics.

A Crowdsourced Frame Disambiguation Corpus with Ambiguity

Anca Dumitrache
FD Mediagroep,
Vrije Universiteit Amsterdam
Netherlands
anca.dmtrch@gmail.com

Lora Aroyo
Google
USA
l.m.aroyo@gmail.com

Chris Welty
Google
USA
cawelty@gmail.com

Abstract

We present a resource for the task of FrameNet semantic frame disambiguation of over 5,000 word-sentence pairs from the Wikipedia corpus. The annotations were collected using a novel crowdsourcing approach with multiple workers per sentence to capture *inter-annotator disagreement*. In contrast to the typical approach of attributing the best single frame to each word, we provide a list of frames with disagreement-based scores that express the confidence with which each frame applies to the word. This is based on the idea that inter-annotator disagreement is at least partly caused by ambiguity that is inherent to the text and frames. We have found many examples where the semantics of individual frames overlap sufficiently to make them acceptable alternatives for interpreting a sentence. We have argued that ignoring this ambiguity creates an overly arbitrary target for training and evaluating natural language processing systems - if humans cannot agree, why would we expect the correct answer from a machine to be any different? To process this data we also utilized an expanded lemma-set provided by the Framester system, which merges FN with WordNet to enhance coverage. Our dataset includes annotations of 1,000 sentence-word pairs whose lemmas are not part of FN. Finally we present metrics for evaluating frame disambiguation systems that account for ambiguity.

1 Introduction

Crowdsourcing has been a popular method to collect corpora for a variety of natural language processing tasks (Snow et al., 2008), although one of its downsides is the crowd's lack of domain knowledge that is helpful in solving some tasks. *Semantic frame disambiguation* is an example of a complex natural language processing task that is usually performed by linguistic experts, subjected to strict annotation guidelines and quality control (Baker, 2012). The theory of frame semantics (J Fillmore, 1982) defines a *frame* as an abstract representation of a word sense, describing a type of entity, relation, or event, together with the associated *roles* implied by the frame. The FrameNet (FN) corpus (Baker et al., 1998) is a collection of semantic frames, together with a corpus of documents annotated with these frames. Similarly to word-sense disambiguation, *frame disambiguation* is the task of obtaining the correct frame for each word, since many words have multiple possible meanings.

Using domain experts for frame disambiguation is expensive and time consuming, resulting in small corpora for this task that do not scale well for modern machine learning methods – FN version 1.7, the latest one at the time of writing, contains only about 10,000 sentences annotated with frames. Furthermore, only using one expert to perform the annotation makes it difficult to capture any diversity of perspectives.

There have been a number of small-scale attempts at using crowdsourcing for frame disambiguation in sentences, showing that the crowd has comparable performance to the FN domain experts (Hong and Baker, 2011), and that the crowd can be used to correct wrong examples that have been collected automatically (Pavlick et al., 2015). Crowd performance can be improved by combining frame role identification with disambiguation (Fossati et al., 2013), or by asking crowd workers to give each other feedback and then letting them change their answer (Chang et al., 2015). Crowdsourcing has also been useful to identify the ambiguity in frame disambiguation (Jurgens, 2013).

Previously, we have shown (Dumitrache et al., 2018a) that while the crowd and FN expert mostly agree over frame disambiguation, disagreement cases are often caused by ambiguity, such as vague or overlapping frame definitions, or incomplete

Proceedings of NAACL-HLT 2019, pages 2164–2170
Minneapolis, Minnesota, June 2 - June 7, 2019.

information in the sentence. Because of these issues with the input data, the approach of selecting one single correct frame for every word, and ignoring alternative interpretations, often results in arbitrary, incomplete ground truth corpora. In order to aggregate annotated data while preserving disagreement, we use the CrowdTruth method[1] (Aroyo and Welty, 2014), which encourages using multiple crowd annotators to perform the same work, and processes the disagreement between them to signal low quality workers, sentences, and frames.

This paper presents a crowdsourced FN frame disambiguation corpus of 5,042 sentence-word pairs (which has since grown to over 9,000 since the submission of this paper). More than 1,000 of these are lexical units (LUs) not part of FN. To our knowledge, it is the largest corpus of this type outside of FN. In addition, we applied the CrowdTruth method, in which each sentence and lexical item is accompanied by *a list of multiple frames* with scores that express the confidence with which each frame applies to the word. This allows us to demonstrate that ambiguity is a prominent feature of frame disambiguation, with many cases where more than one possible frame can apply to the same word. Finally, we present an evaluation of several frame disambiguation models using evaluation metrics that leverage the multiple answers and their confidence scores, and show that even a model that always predicts the top crowd answer will not always have the best performance.

2 Corpus Collection & Analysis

2.1 Data Preprocessing

Our corpus consists of 5,042 candidate word-sentence pairs from Wikipedia (which has since grown to over 9,000 since the submission of this paper) and a candidate list of frames for the word, with 742 unique frames and 1,705 unique lexical units (LUs). The sentences have been randomly selected, based on these criteria:

- The candidate word has *no more than 25 candidate frames*, to not overwhelm the annotators.
- The part of speech of the word is a *verb*.
- The *distribution of candidate frames was optimized for maximum diversity* using a greedy approach.

To gather the candidate frames for each word, we gathered the candidate frames associated with the LU from FN1.7. Next we completed the candidate list using Framester (Gangemi et al., 2016), which maps FN semantic frames to synonym sets from WordNet (Miller, 1995). The sentences were processed with tokenization, sentence splitting, lemmatization and part-of-speech tagging. Then each word with a frame attached to it was matched with all of its possible synonym sets from WordNet, while making sure that the part-of-speech constraint of the synonym set is fulfilled. Using the WordNet mapping, we constructed the list of additional candidate frames for each word. Framester disambiguation used release 1.5 of FN, and some frames changed names in version 1.7, so we manually mapped these frames from FS to their latest version. Framester disambiguation was also used to collect a subset of our corpus consisting of 1,000 sentence-word pairs with LUs that are not part of the FN corpus. For simplicity, we refer to the sentence-word pairs as sentences in the rest of the paper.

2.2 Crowdsourcing Setup

We ran the task on Amazon Mechanical Turk, where the workers were asked to select *all frames* that fit the sense of the highlighted word in a sentence from the multiple choice candidate list, or that none of the frames is correct. We used 15 workers/sentence that were paid \$0.05 for each judgment, and a total cost of \$1.35 per sentence (after factoring in the additional AMT costs).[2]

To aggregate the results of the crowd while also capturing inter-annotator disagreement, we use the CrowdTruth metrics[3] (Dumitrache et al., 2018b), replicating the setup from our previous work (Dumitrache et al., 2018a). The choice of frames of one worker over one sentence are aggregated into a *worker vector* – a binary vector with $n + 1$ components, where n is the number of frames shown together with the sentence, where the decision to pick each of the frames (or none) corresponds to a component in the vector. The vectors are used to calculate quality scores for workers, sentences

---

[1] `http://crowdtruth.org`

[2] `https://mturk.com/`

[3] `https://github.com/CrowdTruth/CrowdTruth-core`

#	SENTENCE	SQS	FRAMES (FSS)
1	Domestication of plants has, over the centuries **improved** disease resistance.	0.652	*improvement or decline* (0.823), *cause to make progress* (0.683)
2	He is the 5th of 8 male players in history to **achieve** this.	0.626	*accomplishment* (0.764), *successful action* (0.709)
3	Albertus Magnus, a Dominican monk, **commented** on the operations and theories of alchemical authorities.	0.511	*communication* (0.522), *statement* (0.703)
4	He **slices** at Hector's armor, throwing him off guard and spinning him around.	0.319	*part piece* (0.499), *cause harm* (0.4), *cutting* (0.394), *attack* (0.254), *hit target* (0.227)
5	Another 46 steps **remain** to climb in order to reach the top, the "terrasse", from where one can enjoy a panoramic view of Paris.	0.308	*left to do* (0.497), *remainder* (0.478), *state continue* (0.319), *existence* (0.155)
6	Borzoi males frequently **weigh** more.	0.283	*assessing* (0.421), *dimension* (0.402), *importance* (0.128)
7	The dance includes bending and **straightening** of the knee giving it a touch of Cuban motion.	0.24	*reshaping* (0.495), *arranging* (0.356), *body movement* (0.298), *cause motion* (0.249)

Table 1: Example sentences with disagreement over the frame annotations (candidate word in bold).

and frames. Although we make all quality scores available as part of the corpus, in this paper we focus on:

- **frame-sentence score** (FSS)**:** the degree with which a frame matches the sense of the word in the sentence. It is the ratio of workers that picked the frame to all the workers that read the sentence, weighted by the worker quality. A high FSS means the frame is clearly expressed in a sentence.
- **sentence quality** (SQS)**:** the overall worker agreement over one sentence. It is the average cosine similarity over all worker vectors for one sentence, weighted by the worker quality and frame quality. A high SQS indicates a clear sentence.

The aggregated crowdsourcing results and the FN 1.5 to 1.7 mapping table are available online.[4]

2.3 Ambiguity in the Corpus

An analysis of the corpus found many examples of inter-annotator disagreement, of which a few examples are shown in Table 1. For 720 sentences, a majority of the workers picked at least 2 frames (examples 1-3 in Table 1). And for 1,514 sentences, no one frame has been picked by a majority of the workers (examples 4-7 in Table 1). Disagreement is also more prominent in the sentences where the LU is not a part of FN (Figure 1).

The disagreement comes from a variety of causes: a parent-child relation between the frames (*statement* and *communication* in #3), an overlap in the definition of the frames (*accomplishment* and *successful action* in #2), the meaning of the word is expressed by a composition of frames (in #7, "straightening of the knee" is a combination of *reshaping* the form of the knee, *arranging* the knee in the right position, and *body movement*), and combinations of all of these reasons (in #4, "slices" is a combination of *part piece* and *cause harm*, and the other frames are their children). More example sentences for each type of disagreement are available in the appendix. The sentences themselves are not difficult to understand, and it can be argued that all of them have one frame that applies the best for the word. The goal of this corpus is to show that next to this best frame for the word, there are other frames that apply to a lesser degree, or capture a different part of the meaning. When evaluating a model for frame disambiguation, it seems unfair to penalize misclassifications of frames that still apply to the word, but with less clarity, in the same way we would penalize a frame that captures a wrong meaning. Also, we argue that models should take into account that annotators do not agree over some examples, and treat them differently than clear expressions of frames. Disagreement can also be caused by worker mistakes (in #6, *dimension* refers to the size of the object, not the act of measuring the size). While we try to mitigate for this by weighing confidence scores with the worker quality, the mistakes still appear in the corpus. This type of disagreement could be useful in future work to identify examples that workers need to be trained on.

3 Evaluating Frame Disambiguation

3.1 Systems Tested

As an example usage of our corpus, we used it to evaluate these frame disambiguation models:

---

[4] `https://github.com/CrowdTruth/FrameDisambiguation`

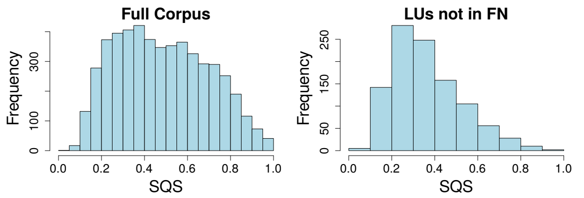


Figure 1: Histogram of SQS values - the quality scores in sentences where the LU is not in FN skew lower.

1. **OS:** The Open-Sesame (Swayamdipta et al., 2017) classifier, pre-trained on the FN corpus (release 1.7). Given a word-sentence pair, OS uses a BiLSTM model with a softmax final layer to predict a single frame for the word. If the LU is not in FN, it cannot make a prediction.
2. **OS+:** We modified the OS classifier to perform multi-label classification. To calculate the confidence score for candidate frame f, we removed the softmax layer and passed the output of the BiLSTM model $\nu(f)$ through the following transformation: $c(f) = [1 + tanh\,\nu(f)]/2$. This gave a score $c(f) \in [0, 1]$ expressing the confidence that frame f is expressed in the sentence.
3. **FS:** Framester includes a tool for rule-based multi-class multi-label frame disambiguation (Gangemi et al., 2016). While for the dataset pre-processing (Sec. 2) we considered the frames for all synsets a word is part of, FS performs an additional word-sense disambiguation step to return a more precise list of frames. We used the tool with *profile T* as it was shown to have the overall better performance. FS can only predict FN frames from the 1.5 release, which is missing 202 frames from version 1.7.

While OS+ produces confidence scores, the other methods produce binary labels for each frame-sentence pair. These models do not have state-of-the-art performance (Hermann et al., 2014; FitzGerald et al., 2015), we picked them because they were accessible and allowed testing on a novel corpus. Finally, we evaluate the quality of the **TC** corpus, containing only the top frame picked by the crowd for every sentence. This test shows what is the best possible performance over our corpus that can be expected from a system such as OS that selects a single frame per sentence.

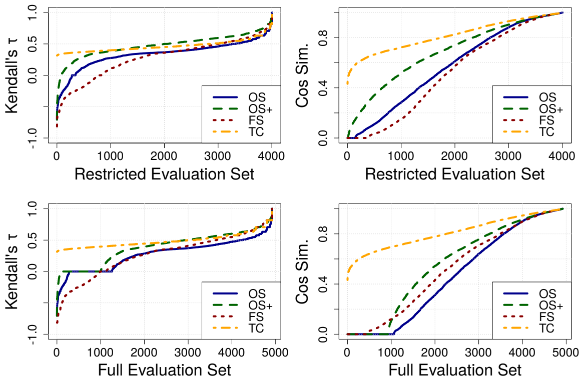


Figure 2: Baselines evaluation results.

	Eval. Metric	OS	OS+	FS	TC
R-Set	Kendall's τ AUC	0.339	**0.477**	0.279	0.466
	Kendall's τ w-avg	0.362	**0.497**	0.3	0.48
	Cos Sim AUC	0.57	**0.685**	0.518	0.818
	Cos Sim w-avg	0.608	**0.717**	0.545	0.854
F-Set	Kendall's τ AUC	0.269	**0.379**	0.253	0.491
	Kendall's τ w-avg	0.307	**0.421**	0.284	0.501
	Cos Sim AUC	0.453	**0.544**	0.511	0.810
	Cos Sim w-avg	0.515	**0.607**	0.539	0.849

Table 2: Aggregated evaluation results.

3.2 Evaluation Metrics & Results

Instead of traditional evaluation metrics that require binary labels, we propose an evaluation methodology that is able to consider multiple candidate frames for each sentence and their quality scores. We use **Kendall's** τ list ranking coefficient (Kendall, 1938) and **cosine similarity** to calculate the distance between the list of frames produced by the crowd labeled with the FSS, and the frames predicted by the baselines in each sentence. Whereas Kendall's τ only accounts for the ranking of the FSS for each frame, cosine similarity uses the actual FSS values in the calculation of the similarity. Both metrics compute a score per sentence (Kendall's $\tau \in [-1, 1]$, and cosine similarity $\in [0, 1]$). Using these metrics, we produce two aggregate statistics over our test corpus: (1) the area-under-curve (AUC) for each metric, normalized by the corpus size, and (2) the SQS-weighted average of each metric ($w - avg$), which also accounts for the ambiguity of the sentence as expressed by the SQS. We evaluate on two versions of the corpus: (1) the restricted set (R-Set) of 4,000 sentences with LUs from the FN corpus, and (2) the full set (F-Set) of 5,042 sentences.

The results (Figure 2 & Table 2) show that OS+ performs best out of all the models, even taking into account sentences with LUs not in FN for which OS+ cannot disambiguate. FS performs the worst out of all models on R-SET, because it cannot find newly added frames from the latest FN release, but improves on the F-SET (FS can find candidate frames for LUs not in FN). The scores on the F-SET were lower for all baselines, suggesting that sentences with LUs not in FN are more difficult to classify – this could be because FN is missing frames that can express the full meaning of these LUs. TC has a good performance, but is far from being unbeatable – when measuring Kendall's τ over the R-SET, OS+ performs better than TC.

4 Conclusions

We described a FrameNet frame disambiguation resource of 5,042 sentence-word pairs, and 1,000 LUs that are new to FN – the largest corpus of this type outside of FN. Since the submission of this paper, the corpus has grown to over 9,000 sentence-word pairs. We also provide confidence scores for each candidate frame that are based on inter-worker disagreement. We made a case for this kind of disagreement reflecting genuine cases of ambiguity in FrameNet frames, caused by: child-parent relations between frames, frames with overlapping definitions, or compositions of frames making up the meaning of a word. The evaluation method we proposed uses the scores for multiple frames, and is thus able to differentiate between frames that still apply to the word, but with less clarity, and frames that capture the wrong meaning. Our goal was to build a resource that recognizes different levels of ambiguity in the expression of the frames in the text, and allows a more fair evaluation of performance of frame disambiguation systems.

Acknowledgments

We would like to thank Luigi Asprino, Valentina Presutti and Aldo Gangemi for their assistance with using the Framester corpus, as well as their advice in better understanding the task of frame disambiguation. We would also like to thank the anonymous crowd workers for their contributions to our crowdsourcing tasks.

References

Lora Aroyo and Chris Welty. 2014. The Three Sides of CrowdTruth. *Journal of Human Computation*, 1:31–34.

Collin F Baker. 2012. FrameNet, current collaborations and future goals. *Language Resources and Evaluation*, 46(2):269–286.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Nancy Chang, Praveen Paritosh, David Huynh, and Collin Baker. 2015. Scaling semantic frame annotation. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 1–10.

Anca Dumitrache, Lora Aroyo, and Chris Welty. 2018a. Capturing ambiguity in crowdsourcing frame disambiguation. In *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2018, Zürich, Switzerland, July 5-8, 2018.*, pages 12–20. AAAI Press.

Anca Dumitrache, Oana Inel, Lora Aroyo, Benjamin Timmermans, and Chris Welty. 2018b. CrowdTruth 2.0: Quality Metrics for Crowdsourcing with Disagreement. *arXiv preprint arXiv:1808.06080*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

Marco Fossati, Claudio Giuliano, and Sara Tonelli. 2013. Outsourcing FrameNet to the crowd. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 742–747.

Aldo Gangemi, Mehwish Alam, Luigi Asprino, Valentina Presutti, and Diego Reforgiato Recupero. 2016. Framester: a wide coverage linguistic linked data hub. In *European Knowledge Acquisition Workshop*, pages 239–254. Springer.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1448–1458.

Jisup Hong and Collin F. Baker. 2011. How good is the crowd at "real" WSD? In *Proceedings of the 5th Linguistic Annotation Workshop*, LAW V '11, pages 30–37. Association for Computational Linguistics.

Charles J Fillmore. 1982. *Frame Semantics*, volume 34, pages 111–138.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *HLT-NAACL*, pages 556–562.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. FrameNet+: Fast paraphrastic tripling of FrameNet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 408–413.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263. Association for Computational Linguistics.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *arXiv preprint arXiv:1706.09528*.

A Ambiguity Examples in the Corpus

#	Sentence	SQS	Frames (FSS)
1	These Articles have historically shaped and **continue** to direct the ethos of the Communion.	0.795	*activity ongoing* (0.862) *process continue* (0.86)
2	"A Modest Proposal" is **included** in many literature programs as an example of early modern western satire.	0.771	*inclusion* (0.89) *cause to be included* (0.813)
3	The states often **failed** to meet these requests in full, leaving both Congress and the Continental Army chronically short of money.	0.628	*endeavor failure* (0.826) *success or failure* (0.8)
4	This is a chart of trend of nominal gross domestic product of Angola at market prices **using** International Monetary Fund data.	0.598	*using resource* (0.831) *using* (0.554) *tool purpose* (0.336)
5	The Asian tigers have now all **received** developed country status, having the highest GDP per capita in Asia.	0.504	*receiving* (0.751) *getting* (0.556)
6	MasterCard has released Global Destination Cities Index 2013 with 10 of 20 are **dominated** by Asia and Pacific Region Cities.	0.467	*dominate situation* (0.638) *dominate competitor* (0.579) *being in control* (0.327)

Table 3: Ambiguity because of parent-child relation between frames.

#	Sentence	SQS	Frames (FSS)
1	Kournikova then **withdrew** from several events due to continuing problems with her left foot and did not return until Leipzig.	0.725	*withdraw from participation* (0.955), *removing* (0.61)
2	Some aikido organizations use belts to **distinguish** practitioners' grades.	0.68	*differentiation* (0.867) *distinctiveness* (0.703)
3	Since then, it has focused on **improving** relationships with Western countries, cultivating links with other Portuguese-speaking countries, and asserting its own national interests in Central Africa.	0.654	*improvement or decline* (0.787) *cause to make progress* (0.732)
4	To **emphasize** the validity of the Levites' claim to the offerings and tithes of the Israelites, Moses collected a rod from the leaders of each tribe in Israel and laid the twelve rods over night in the tent of meeting.	0.65	*emphasizing* (0.764) *convey importance* (0.638)
5	He not only had enough food from his subjects to **maintain** his military, but the taxes collected from traders and merchants added to his coffers sufficiently to fund his continuous wars.	0.453	*cause to continue* (0.7) *activity ongoing* (0.602)
6	He **spent** the later part of his life in the United States, living in Los Angeles from 1937 until his death.	0.29	*taking time* (0.41) *expend resource* (0.365)

Table 4: Ambiguity because of overlapping frame definitions.

#	SENTENCE	SQS	FRAMES (FSS)
1	These writings lack the mystical, philosophical elements of alchemy, but do contain the works of Bolus of Mendes (or Pseudo-Democritus), which **aligned** these recipes with theoretical knowledge of astrology and the classical elements.	0.284	*arranging* (0.474) *adjusting* (0.4) *assessing* (0.298) *compatibility* (0.254) *undergo change* (0.169)
2	However, commercial application of this fact has challenges in **circumventing** the passivating oxide layer, which inhibits the reaction, and in storing the energy required to regenerate the aluminium metal.	0.239	*dodging* (0.477) *compliance* (0.248) *surpassing* (0.204) no frame (0.148)
3	This had the effect of **inculcating** the principle of “Lex orandi, lex credendi” (Latin loosely translated as ’the law of praying [is] the law of believing’) as the foundation of Anglican identity and confession.	0.201	*education teaching* (0.384) *communication* (0.35) no frame (0.153)
4	Legal segregation ended in the states in 1964, but Jim Crow customs often continued until specifically **challenged** in court.	0.172	*difficulty* (0.372) *competition* (0.283) *taking sides* (0.257) *communication* (0.154)
5	When Washington’s army arrived outside Yorktown, Cornwallis prematurely abandoned his outer position, **hastening** his subsequent defeat.	0.134	*speed description* (0.39) *assistance* (0.209) *self motion* (0.165) *travel* (0.16) *causation* (0.124)

Table 5: Ambiguity because the meaning of the word is expressed by a composition of frames.

Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets

Nelson F. Liu[♠♡♣] **Roy Schwartz**[♠♣] **Noah A. Smith**[♠♣]
[♠]Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
[♡]Department of Linguistics, University of Washington, Seattle, WA, USA
[♣]Allen Institute for Artificial Intelligence, Seattle, WA, USA
`{nfliu,roysch,nasmith}@cs.washington.edu`

Abstract

Several datasets have recently been constructed to expose brittleness in models trained on existing benchmarks. While model performance on these *challenge datasets* is significantly lower compared to the original benchmark, it is unclear what particular weaknesses they reveal. For example, a challenge dataset may be difficult because it targets phenomena that current models cannot capture, or because it simply exploits blind spots in a model's specific training set. We introduce *inoculation by fine-tuning*, a new analysis method for studying challenge datasets by exposing models (the metaphorical patient) to a small amount of data from the challenge dataset (a metaphorical pathogen) and assessing how well they can adapt. We apply our method to analyze the NLI "stress tests" (Naik et al., 2018) and the Adversarial SQuAD dataset (Jia and Liang, 2017). We show that after slight exposure, some of these datasets are no longer challenging, while others remain difficult. Our results indicate that failures on challenge datasets may lead to very different conclusions about models, training datasets, and the challenge datasets themselves.

1 Introduction

NLP research progresses through the construction of dataset-benchmarks and the development of systems whose performance on them can be fairly compared. A recent pattern involves *challenges* to benchmarks:[1] manipulations to input data that result in severe degradation of system performance, but not human performance. These challenges have been used as evidence that current systems are brittle (Belinkov and Bisk, 2018; Mudrakarta et al., 2018; Zhao et al., 2018; Glockner et al., 2018; Ebrahimi et al., 2018; Ribeiro et al., 2018, *inter alia*). For instance, Naik et al. (2018) generated natural language inference challenge data by applying simple textual transformations to existing examples from MultiNLI (Williams et al., 2018) and SNLI (Bowman et al., 2015). Similarly, Jia and Liang (2017) built an adversarial evaluation dataset for reading comprehension based on SQuAD (Rajpurkar et al., 2016).

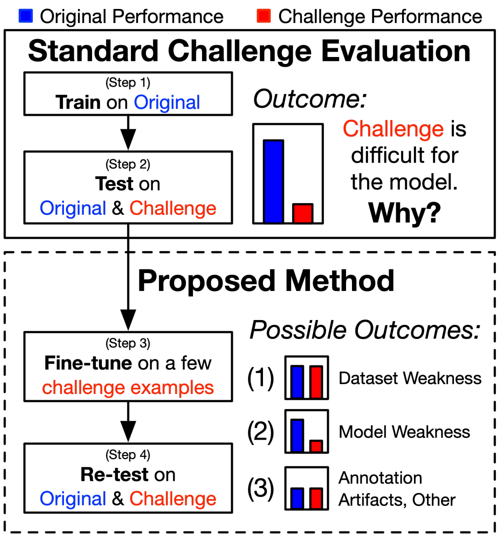


Figure 1: An illustration of the standard challenge evaluation procedure (e.g., Jia and Liang, 2017) and our proposed analysis method. "Original" refers to the a standard dataset (e.g., SQuAD) and "Challenge" refers to the challenge dataset (e.g., Adversarial SQuAD). Outcomes are discussed in Section 2.

What should we conclude when a system fails on a challenge dataset? In some cases, a challenge might exploit blind spots in the design of the *original* dataset (*dataset weakness*). In others, the challenge might expose an inherent inability of a particular model family to handle certain natural language phenomena (*model weakness*). These are, of course, not mutually exclusive.

We introduce **inoculation by fine-tuning**, a

---

[1] Often referred to as "adversarial datasets" or "attacks".

Proceedings of NAACL-HLT 2019, pages 2171–2179
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

new method for analyzing the effects of challenge datasets (Figure 1).[2] Given a model trained on the original dataset, we expose it to a small number of examples from the challenge dataset, allowing learning to continue. To the extent that the weakness lies with the original dataset, then the inoculated model will perform well on both the original and challenge held-out data (Outcome 1 in Figure 1). If the weakness lies with the model, then inoculation will prove ineffective and the model's performance will remain unchanged (Outcome 2).

Inoculation can also decrease a model's performance on the original dataset (Outcome 3). This case is not as clear as the first two, and could result from systematic differences between the original and challenge datasets, due to, e.g., predictive artifacts in either dataset (Gururangan et al., 2018).

We apply our method to analyze six challenge datasets: the *word overlap*, *negation*, *spelling errors*, *length mismatch* and *numerical reasoning* NLI challenge datasets proposed by Naik et al. (2018), as well as the Adversarial SQuAD reading comprehension challenge dataset (Jia and Liang, 2017). We analyze NLI datasets with the ESIM (Chen et al., 2017) and the decomposable attention (Parikh et al., 2016) models, and reading comprehension with the BiDAF (Seo et al., 2017) and the QANet (Yu et al., 2018) models.

By fine-tuning on, in some cases, as few as 100 examples, both NLI models are able to recover almost the entire performance gap on both the *word overlap* and *negation* challenge datasets (Outcome 1). In contrast, both models struggle to adapt to the *spelling error* and *length mismatch* challenge datasets (Outcome 2). On the *numerical reasoning* challenge dataset, both models close all of the gap using a small number of samples, but at the expense of performance on the original dataset (Outcome 3). For Adversarial SQuAD, BiDAF closes 60% of the gap with minimal fine-tuning, but suffers a 7% decrease in original test set performance (Outcome 3). QANet shows similar trends.

Our proposed analysis is broadly applicable, easy to perform, and task-agnostic. By gaining a better understanding of how challenge datasets stress models, we can better tease apart limitations of datasets and limitations of models.

2 Inoculation by Fine-Tuning

Our method assumes access to an original dataset divided into training and test portions, as well as a challenge dataset, divided into a (small) training set[3] and a test set. After training on the original (training) data, we measure system performance on both test sets. We assume the usual observation—a generalization gap with considerably lower performance on the challenge test set.

We then proceed to fine-tune the model on the challenge training data, i.e., continuing to train the pre-trained model on the new data until development performance on the original development set has not improved for five epochs.[4] Finally, we measure performance of the inoculated model on both the original and challenge test sets. Three clear outcomes of interest are:[5]

Outcome 1 The gap closes, i.e., the inoculated system retains its (high) performance on the original test set and performs as well (or nearly so) on the challenge test set. This case suggests that the challenge dataset did not reveal a weakness in the model family. Instead, the challenge has likely revealed a lack of diversity in the original dataset.

Outcome 2 Performance on both test sets is unchanged. This indicates that the challenge dataset has revealed a fundamental weakness of the model; it is unable to adapt to the challenge data distribution, even with some exposure.

Outcome 3 Inoculation damages performance on the original test set (regardless of improvement on the challenge test set). The main difference between Outcome 3 and Outcomes 1 and 2 is that here, by fine-tuning, the model is shifting towards a challenge distribution that somehow contradicts the original distribution. This could result from, e.g., a different label distribution between both datasets, or annotation artifacts that exist in one dataset but not in the other (see Sections 3.2, 3.3).

3 Not all Challenge Datasets are Alike

To demonstrate the utility of our method, we apply it to analyze the NLI stress tests (Naik et al.,

---

[2] *Inoculation* evokes the idea that treatable diseases have different implications (for society and for the patient) than untreatable ones. We differentiate the abstract process of inoculation from our way of executing it (fine-tuning) since it is easy to imagine alternative ways to inoculate a model.

[3] The exact amount of challenge data used for fine-tuning might affect our conclusions, so we consider different sizes of the "vaccine" in our experiments.

[4] The use of the original development set is meant to both prevent us from using more challenge data and verify that the learner does not completely forget the original dataset.

[5] The outcome may also lie between these extremes, necessitating deeper analysis.

Category	Premise	Hypothesis
Word Overlap	Possibly no other country has had such a turbulent history.	The country's history has been turbulent **and true is true**.
Negation	Possibly no other country has had such a turbulent history.	The country's history has been turbulent **and false is not true**.
Spelling Errors	Fix the engine, Dave Hanson, he called.	Hanson received **ordets** not to fix **teh** engine.
Length Mismatch	Possibly no other country has had such a turbulent history **and true is true and true is true and true is true and true is true and true is true**.	The countrys history has been turbulent.
Numerical Reasoning	**Tim has 350 pounds of cement in 100, 50, and 25 pound bags.**	**Tim has less than 750 pounds of cement in 100, 50, and 25 pound bags.**

Table 1: Examples from each of the NLI challenge datasets analyzed, a subset of a broader suite of NLI stress tests proposed by Naik et al. (2018). Boldface denotes perturbations to original MultiNLI examples. Figure contents reproduced from Naik et al. (2018).

Article: Super Bowl 50
Paragraph: *Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*
Question: *What is the name of the quarterback who was 38 in Super Bowl XXXIII?*

Figure 2: An example from the Adversarial SQuAD dataset, with the distractor sentence in blue. Figure reproduced from Jia and Liang (2017).

2018) and the Adversarial SQuAD dataset (Jia and Liang, 2017). We fine-tune models on a varying number of examples from the challenge dataset training split in order to study whether our method is sensitive to the level of exposure.[6] Our results demonstrate that different challenge datasets lead to different outcomes. We release code for reproducing our results.[7]

3.1 Datasets

We briefly describe the analyzed datasets, but refer readers to the original publications for details.

NLI Stress Tests Naik et al. (2018) proposed six automatically-constructed "stress tests", each focusing on a different weakness of NLI systems. We analyze five of these stress tests (Table 1).[8]

The **word overlap** challenge dataset is designed to exploit models' sensitivity to high lexical overlap in the premise and hypothesis by appending the tautology *"and true is true"* to the hypothesis. The **negation** challenge dataset is based on the observation that negation words (e.g., *"no"*, *"not"*) cause the model to classify neutral or entailed statements as contradiction. In this dataset, the tautology *"and false is not true"* is appended to the hypothesis sentence. The **spelling errors** challenge dataset is designed to evaluate model robustness to noisy data in the form of misspellings. The **length mismatch** challenge dataset is designed to exploit models' inability to handle examples with much longer premises than hypotheses. In this dataset, the tautology *"and true is true"* is appended five times to the end of the premise. Lastly, the **numerical reasoning** challenge dataset is designed to test models' ability to perform algebraic calculations, by introducing premise-hypothesis pairs containing numerical expressions.

We analyze these challenge datasets using two models, both trained on the MultiNLI dataset:[9] the ESIM model (Chen et al., 2017) and the decomposable attention model (DA; Parikh et al., 2016).

To better address the spelling errors challenge dataset, we also train a character-sensitive version of the ESIM model. We concatenate the word representations with the 50-dimensional hidden states

---

[6] See Appendix A for experimental process details.

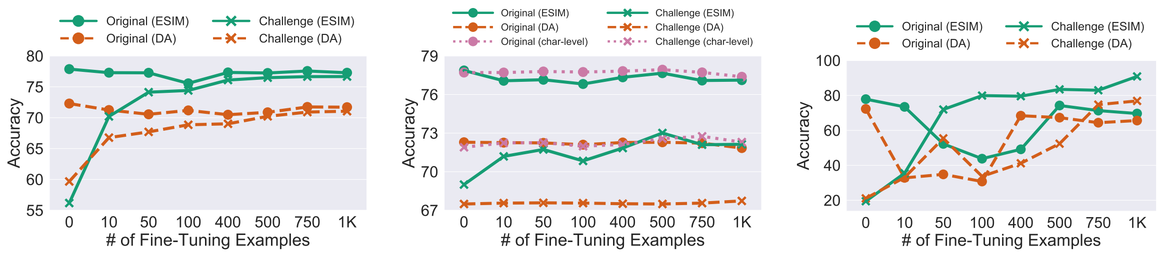
[7] `http://nelsonliu.me/papers/inoculation-by-finetuning`

[8] The remaining challenge dataset—**antonym**—is briefly discussed in Section 3.3.

[9] MultiNLI has domain-matched and mismatched development data, so we train separate "matched" and "mismatched" models that each use the corresponding development set for learning rate scheduling and early stopping. We observe similar results in both cases, so we focus on the models trained on "matched" data. See Appendix B for mismatched results.

Outcome 1 **Outcome 2** **Outcome 3**

(a) Word Overlap **(c) Spelling Errors** **(e) Numerical Reasoning**



(b) Negation **(d) Length Mismatch** **(f) Adversarial SQuAD**

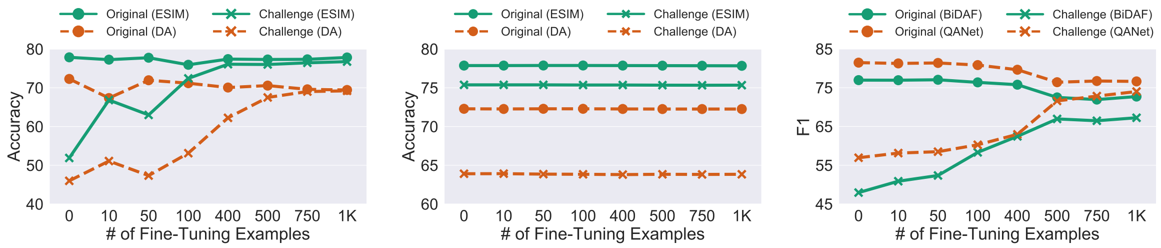


Figure 3: Inoculation by fine-tuning results. (a–e): NLI accuracy for the ESIM and decomposable attention (DA) models. (f): Reading comprehension F_1 scores for the BiDAF and QANet models.
Fine-tuning on a small number of word overlap (a) and negation (b) examples erases the performance gap (Outcome 1). Fine-tuning does not yield significant improvement on spelling errors (c) and length mismatch (d), but does not degrade original performance either (Outcome 2). Fine-tuning on numerical reasoning (e) closes the gap entirely, but also reduces performance on the original dataset (Outcome 3). On Adversarial SQuAD (f), around 60% of the performance gap is closed after fine-tuning, though performance on the original dataset decreases (Outcome 3). On each challenge dataset, we observe similar trends between different models.

produced by running each token through a character bidirectional GRU (Cho et al., 2014).

Adversarial SQuAD Jia and Liang (2017) created a challenge dataset for reading comprehension by appending automatically-generated distractor sentences to SQuAD passages. The appended distractor sentences are crafted to look similar to the question while not contradicting the correct answer or misleading humans (Figure 2). The authors released model-independent Adversarial SQuAD examples, which we analyze. For our analysis, we use the BiDAF model (Seo et al., 2017) and the QANet model (Yu et al., 2018).

3.2 Results

We refer to difference between a model's pre-inoculation performance on the original test set and the challenge test set as the *performance gap*.

NLI Stress Tests Figure 3 presents NLI accuracy for the ESIM and DA models on the word overlap, negation, spelling errors, length mismatch and numerical reasoning challenge datasets after fine-tuning on a varying number of challenge examples.

For the word overlap and negation challenge datasets, both ESIM and DA quickly close the performance gap when fine-tuning (Outcome 1). For instance, on both of the aforementioned challenge datasets, ESIM requires only 100 examples to close over 90% of the performance gap while maintaining high performance on the original dataset. Since these performance gaps are closed after seeing a few challenge dataset examples (< 0.03% of the original MultiNLI training dataset), these challenges are likely difficult because they exploit easily-recoverable gaps in the models' training dataset rather than highlighting their inability to capture semantic phenomena.

In contrast, on spelling errors and length mismatch, fine-tuning does not allow either model to close a substantial portion of the performance gap, while performance on the original dataset is unaffected (Outcome 2).[10] Interestingly, the character-aware ESIM model trained on spelling errors shows a similar trend, suggesting that the this challenge set is highlighting a weakness of ESIM that goes beyond the word representation.

On numerical reasoning, the entire gap is closed by fine-tuning ESIM on 100 examples, or DA on 750 examples. However, both models' original dataset performance substantially decreases (Outcome 3; see discussion in Section 3.3).

Adversarial SQuAD Figure 3(f) shows BiDAF and QANet results after fine-tuning on a varying number of challenge samples.

Fine-tuning BiDAF on only 400 challenge examples closes more than 60% of the performance gap, but also results in substantial performance loss on the original SQuAD development set; fine-tuning QANet yields the same trend (Outcome 3). In this case, the model likely takes advantage of the fact that the adversarial distractor sentence is always concatenated to the end of the paragraph.[11]

3.3 Discussion

Explaining the Numerical Reasoning Results The relative ease with which the ESIM model overcomes the numerical reasoning challenge seems to contradict the findings of Naik et al. (2018), who observed that "*the model is unable to perform reasoning involving numbers or quantifiers . . .*". Indeed, it seems unlikely that a model will learn to perform algebraic numerical reasoning based on as few as 50 NLI examples.

However, a closer look at this dataset provides a potential explanation for this finding. The dataset was constructed such that a simple 3-rule baseline is able to surpass 80% on the task (see Appendix C). For instance, 35% of the dataset examples contain the phrase "more than" or "less than" in their hypothesis, and 95% of these have the label "neutral". As a result, learning a handful of these rules is sufficient for achieving high performance on this challenge dataset.

This observation highlights a key property of Outcome 3: challenge datasets that are easily recoverable by our method, at the expense of performance on the original dataset, are likely not testing the full breadth of a linguistic phenomenon but rather a specific aspect of it.

Limitations of Our Method Our inoculation method assumes a somewhat balanced label distribution in the challenge dataset training portion. If a challenge dataset is highly skewed to a specific label, fine-tuning will result in simply learning to predict the majority label; such a model would achieve high performance on the challenge dataset and low performance on the original dataset (Outcome 3). For such datasets, the result of our method is not very informative.[12] Nonetheless, as in the numerical reasoning case discussed above, this lack of diversity signals a somewhat limited phenomenon captured by the challenge dataset.

4 Conclusion

We presented a method for studying *why* challenge datasets are difficult for models. Our method fine-tunes models on a small number of challenge dataset examples. This analysis yields insights into models, their training datasets, and the challenge datasets themselves. We applied our method to analyze the challenge datasets of Naik et al. (2018) and Jia and Liang (2017). Our results indicate that some of these challenge datasets break models by exploiting blind spots in their training data, while others may challenge more fundamental weaknesses of model families.

Acknowledgments

We thank Aakanksha Naik and Abhilasha Ravichander for generating NLI stress test examples from the MultiNLI training split, and Robin Jia for answering questions about the Adversarial SQuAD dataset. We also thank the members of the Noah's ARK group at the University of Washington, the researchers at the Allen Institute for Artificial Intelligence, and the anonymous reviewers for their valuable feedback. NL is supported by a Washington Research Foundation Fellowship and a Barry M. Goldwater Scholarship. This work was supported in part by a hardware gift from NVIDIA Corporation.

---

[10] The length mismatch dataset is not particularly challenging for the ESIM model: its untuned performance on the challenge set is only 2.5% lower than its original performance. Nonetheless, this gap remains fixed even after fine-tuning

[11] Indeed, Jia and Liang (2017) show that models trained on Adversarial SQuAD are able to overcome the adversary by simply learning to ignore the last sentence of the passage.

[12] For instance, the *antonym* challenge dataset of Naik et al. (2018), in which all examples are labeled "contradiction".

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proc. of ICLR*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proc. of ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proc. of ACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proc. of NLP-OSS*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proc. of ACL*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. of NAACL*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proc. of EMNLP*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *Proc. of ACL*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Penstein Rosé, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proc. of COLING*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proc. of EMNLP*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proc. of ACL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proc. of ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proc. of ICLR*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proc. of ICLR*.

Appendices

A Experimental Setup Details

Generating challenge training sets When varying the size of the challenge dataset train split used for fine-tuning, we subsample inclusively. For example, the dataset used for fine-tuning on 5 examples is a subset of the dataset used for fine-tuning on 100 examples, which is a subset of the dataset used for fine-tuning on 1000 examples.

The *word overlap*, *negation*, *spelling errors* and *length mismatch* NLI challenge datasets, as well as Adversarial SQuAD, include splits for training and evaluation. To generate the datasets used for fine-tuning, we subsample 1000 random examples from each of the challenge dataset train splits.[13] The evaluation splits are used as-is.

The *numerical reasoning* NLI challenge dataset is unsplit. As a result, we generate the datasets used for fine-tuning by subsampling 1000 random examples from the entirety of the challenge dataset, and use the remaining examples for evaluation.

Experimental details To train the ESIM model of Chen et al. (2017), the decomposable attention model of Parikh et al. (2016), the BiDAF model of Seo et al. (2017), and the QANet model of Yu et al. (2018), we use the implementations in AllenNLP (Gardner et al., 2018). The models are trained with the same hyperparameters as described in their respective papers.

For each training dataset size, we tune the learning rate on the *original* development set accuracy; the learning rate is halved whenever validation performance (F_1 for SQuAD, accuracy for NLI) does not improve, and we employ early stopping with a patience of 5. This ensures that we are not implicitly using additional challenge dataset examples. For each model and amount of challenge dataset examples used for fine-tuning, the reported challenge dataset performance is the performance of the learning rate configuration that yields the best challenge dataset performance. We leave all other hyperparameters (such as the batch size and choice of optimizer) unchanged from the model's original training procedure.

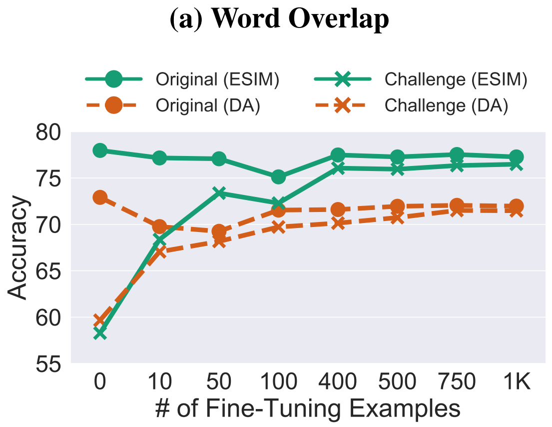
For the Adversarial SQuAD experiments, we experiment with learning rates of 0.00001, 0.0001, 0.001 and 0.01. For the NLI stress test experiments, we experiment with learning rates of 0.000001, 0.00001, 0.0001, 0.0004, 0.001, and 0.01.

We use AllenNLP to run our fine-tuning experiments.

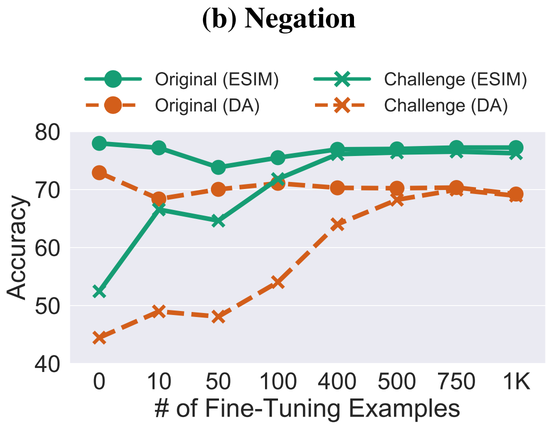
---

[13] For Adversarial SQuAD, we subsample from distinct passages.

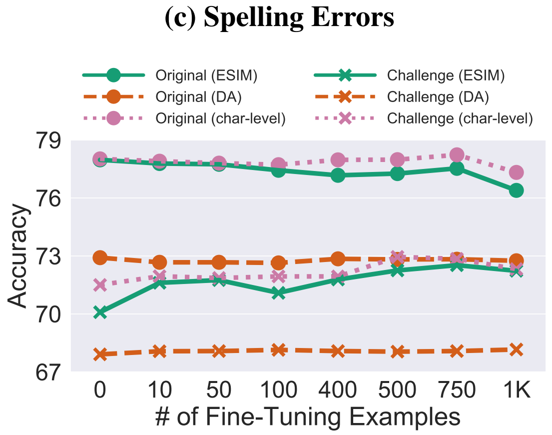
B MultiNLI Mismatched Stress Test Results



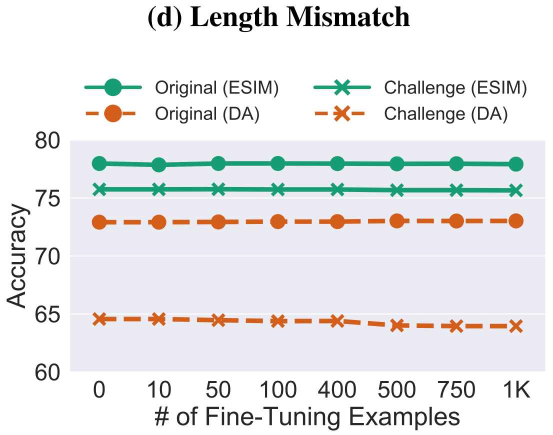
(a) Performance of the ESIM and DA models after fine-tuning on a variable number of word overlap challenge dataset examples generated from the MultiNLI mismatched development set.



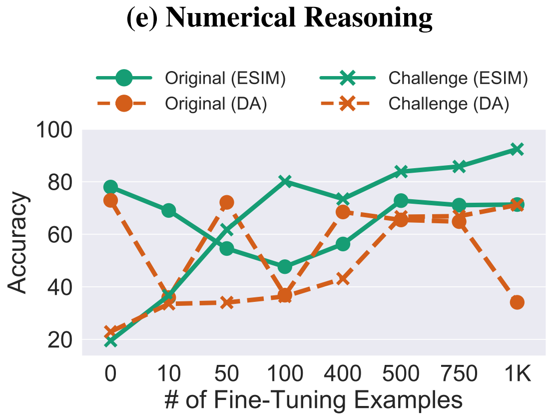
(b) Performance of the ESIM and DA models after fine-tuning on a variable number of negation challenge dataset examples generated from the MultiNLI mismatched development set.



(c) Performance of the ESIM (with and without an additional character-level component) and DA models after fine-tuning on a variable number of spelling error challenge dataset examples generated from the MultiNLI mismatched development set.



(d) Performance of the ESIM and DA models after fine-tuning on a variable number of length mismatch challenge dataset examples generated from the MultiNLI mismatched development set.



(e) Performance of the ESIM and DA models (where the mismatched development set was used during training to control learning rate scheduling and early stopping) after fine-tuning on a variable number of numerical reasoning challenge dataset examples.

C Three Simple Rules for the Numerical Reasoning Dataset

The numerical reasoning dataset of Naik et al. (2018) has 7,596 examples in total, with 2,532 in each of the “entailment”, “neutral”, and “contradiction” categories. With only three rules, we can correctly classify around 82% of the examples.

1,235 examples (out of the 7,596 in total) can be correctly labeled as contradiction with the rule: “more than” or “less than” do not appear in the premise or the hypothesis.

2,664 examples (out of the 6,361 examples left to be considered) contain “more than” or “less than” in the hypothesis. Of these 2,664 examples, 2,532 have the label “neutral”, 66 have the label “entailment”, and 66 have the label “contradiction”. So, if the hypothesis contains “more than” or “less than”, we predict “neutral”. This rule leads us to correctly classify 2,532 examples and incorrectly classify 132 examples.

Finally, we have 3,697 examples to be considered. All 3,697 of these examples have “more than” or “less than” in the premise. 2,466 of these examples are labeled “entailment”, while 1,231 are labeled “contradicion”. By assigning the label “entailment” to examples that contain “more than” or “less than” in their premise, we correctly classify 2,446 examples and incorrectly classify 1,231 examples.

In total, these three rules result in correct predictions on 6,233 examples out of 7,596 (82.05%).

A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization

Dai Quoc Nguyen[1], **Thanh Vu**[2], **Tu Dinh Nguyen**[1], **Dat Quoc Nguyen**[3], **Dinh Phung**[1]
[1]Monash University, Australia; [3]The University of Melbourne, Australia
[2]The Australian e-Health Research Centre, CSIRO, Australia
[1]{dai.nguyen,tu.dinh.nguyen,dinh.phung}@monash.edu
[2]thanh.vu@csiro.au; [3]dqnguyen@unimelb.edu.au

Abstract

In this paper, we introduce an embedding model, named CapsE, exploring a capsule network to model relationship triples *(subject, relation, object)*. Our CapsE represents each triple as a 3-column matrix where each column vector represents the embedding of an element in the triple. This 3-column matrix is then fed to a convolution layer where multiple filters are operated to generate different feature maps. These feature maps are reconstructed into corresponding capsules which are then routed to another capsule to produce a continuous vector. The length of this vector is used to measure the plausibility score of the triple. Our proposed CapsE obtains better performance than previous state-of-the-art embedding models for knowledge graph completion on two benchmark datasets WN18RR and FB15k-237, and outperforms strong search personalization baselines on SEARCH17.

1 Introduction

Knowledge graphs (KGs) containing relationship triples *(subject, relation, object)*, denoted as *(s, r, o)*, are the useful resources for many NLP and especially information retrieval applications such as semantic search and question answering (Wang et al., 2017). However, large knowledge graphs, even containing billions of triples, are still incomplete, i.e., missing a lot of valid triples (West et al., 2014). Therefore, much research efforts have focused on the knowledge graph completion task which aims to predict missing triples in KGs, i.e., predicting whether a triple not in KGs is likely to be valid or not (Bordes et al., 2011, 2013; Socher et al., 2013). To this end, many embedding models have been proposed to learn vector representations for entities (i.e., *subject*/head entity and *object*/tail entity) and relations in KGs, and obtained state-of-the-art results as summarized by Nickel et al. (2016a) and Nguyen (2017). These embedding models score triples *(s, r, o)*, such that valid triples have higher plausibility scores than invalid ones (Bordes et al., 2011, 2013; Socher et al., 2013). For example, in the context of KGs, the score for *(Melbourne, cityOf, Australia)* is higher than the score for *(Melbourne, cityOf, United Kingdom)*.

Triple modeling is applied not only to the KG completion, but also for other tasks which can be formulated as a triple-based prediction problem. An example is in search personalization, one would aim to tailor search results to each specific user based on the user's personal interests and preferences (Teevan et al., 2005, 2009; Bennett et al., 2012; Harvey et al., 2013; Vu et al., 2015, 2017). Here the triples can be formulated as *(submitted query, user profile, returned document)* and used to re-rank documents returned to a user given an input query, by employing an existing KG embedding method such as TransE (Bordes et al., 2013), as proposed by Vu et al. (2017). Previous studies have shown the effectiveness of modeling triple for either KG completion or search personalization. However, there has been no single study investigating the performance on both tasks.

Conventional embedding models, such as TransE (Bordes et al., 2013), DISTMULT (Yang et al., 2015) and ComplEx (Trouillon et al., 2016), use addition, subtraction or simple multiplication operators, thus only capture the linear relationships between entities. Recent research has raised interest in applying deep neural networks to triple-based prediction problems. For example, Nguyen et al. (2018) proposed ConvKB—a convolutional neural network (CNN)-based model for KG completion and achieved state-of-the-art results. Most of KG embedding models are constructed to modeling entries at the same dimension of the given triple, where presumably each dimension captures some relation-specific attribute of entities. To the

Proceedings of NAACL-HLT 2019, pages 2180–2189
Minneapolis, Minnesota, June 2 - June 7, 2019.

best of our knowledge, however, none of the existing models has a "deep" architecture for modeling the entries in a triple at the same dimension.

Sabour et al. (2017) introduced capsule networks (CapsNet) that employ capsules (i.e., *each capsule is a group of neurons*) to capture entities in images and then uses a routing process to specify connections from capsules in a layer to those in the next layer. Hence CapsNet could encode the intrinsic spatial relationship between a part and a whole constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints. Each capsule accounts for capturing variations of an object or object part in the image, which can be efficiently visualized. Our high-level hypothesis is that embedding entries at the same dimension of the triple also have these variations, although it is not straightforward to be visually examined.

To that end, we introduce CapsE to explore a novel application of CapsNet on triple-based data for two problems: KG completion and search personalization. Different from the traditional modeling design of CapsNet where capsules are constructed by splitting feature maps, we use capsules to model the entries at the same dimension in the entity and relation embeddings. In our CapsE, $\boldsymbol{v}_s$, $\boldsymbol{v}_r$ and $\boldsymbol{v}_o$ are unique k-dimensional embeddings of s, r and o, respectively. The embedding triple $[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o]$ of *(s, r, o)* is fed to the convolution layer where multiple filters of the same 1×3 shape are repeatedly operated over every row of the matrix to produce k-dimensional feature maps. Entries at the same dimension from all feature maps are then encapsulated into a capsule. Thus, *each capsule can encode many characteristics in the embedding triple to represent the entries at the corresponding dimension*. These capsules are then routed to another capsule which outputs a continuous vector whose length is used as a score for the triple. Finally, this score is used to predict whether the triple *(s, r, o)* is valid or not.

In summary, our main contributions from this paper are as follows:

• We propose an embedding model CapsE using the capsule network (Sabour et al., 2017) for modeling relationship triples. To our best of knowledge, our work is the first consideration of exploring the capsule network to knowledge graph completion and search personalization.

• We evaluate our CapsE for knowledge graph completion on two benchmark datasets WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). CapsE obtains the best mean rank on WN18RR and the highest mean reciprocal rank and highest Hits@10 on FB15k-237.

• We restate the prospective strategy of expanding the triple embedding models to improve the ranking quality of the search personalization systems. We adapt our model to search personalization and evaluate on SEARCH17 (Vu et al., 2017) – a dataset of the web search query logs. Experimental results show that our CapsE achieves the new state-of-the-art results with significant improvements over strong baselines.

2 The proposed CapsE

Let $\mathcal{G}$ be a collection of valid factual triples in the form of *(subject, relation, object)* denoted as *(s, r, o)*. Embedding models aim to define a *score function* giving a score for each triple, such that valid triples receive higher scores than invalid triples.

We denote $\boldsymbol{v}_s$, $\boldsymbol{v}_r$ and $\boldsymbol{v}_o$ as the k-dimensional embeddings of s, r and o, respectively. In our proposed CapsE, we follow Nguyen et al. (2018) to view each embedding triple $[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o]$ as a matrix $\boldsymbol{A} = [\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o] \in \mathbb{R}^{k \times 3}$, and denote $\boldsymbol{A}_{i,:} \in \mathbb{R}^{1 \times 3}$ as the i-th row of $\boldsymbol{A}$. We use a filter $\boldsymbol{\omega} \in \mathbb{R}^{1 \times 3}$ operated on the convolution layer. This filter $\boldsymbol{\omega}$ is repeatedly operated over every row of $\boldsymbol{A}$ to generate a feature map $\mathbf{q} = [\mathsf{q}_1, \mathsf{q}_2, ..., \mathsf{q}_k] \in \mathbb{R}^k$, in which $\mathsf{q}_i = g\left(\boldsymbol{\omega} \cdot \boldsymbol{A}_{i,:} + b\right)$ where $\cdot$ denotes a dot product, $b \in \mathbb{R}$ is a bias term and g is a non-linear activation function such as ReLU. Our model uses multiple filters $\in \mathbb{R}^{1 \times 3}$ to generate feature maps. We denote $\boldsymbol{\Omega}$ as the set of filters and $\mathsf{N} = \mid \boldsymbol{\Omega} \mid$ as the number of filters, thus we have N k-dimensional feature maps, for which each feature map can capture one single characteristic among entries at the same dimension.

We build our CapsE with two single capsule layers for a simplified architecture. In the first layer, we construct k capsules, wherein entries at the same dimension from all feature maps are encapsulated into a corresponding capsule. Therefore, each capsule can capture many characteristics among the entries at the corresponding dimension in the embedding triple. These characteristics are generalized into one capsule in the second layer which produces a vector output whose length is used as the score for the triple.

The first capsule layer consists of k capsules, for which each capsule $i \in \{1, 2, ..., k\}$ has a vector

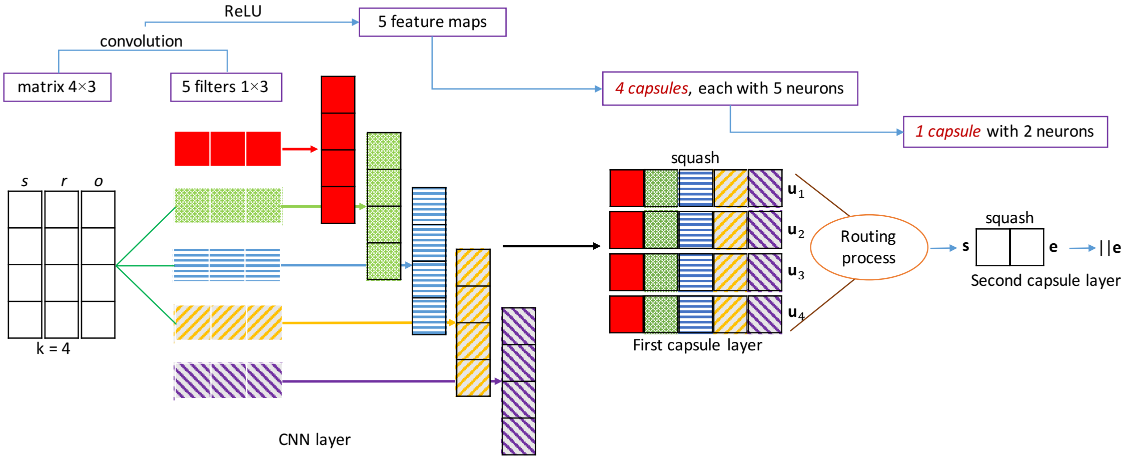


Figure 1: An example illustration of our CapsE with $k = 4$, $\mathsf{N} = 5$, and $d = 2$.

output $\mathbf{u}_i \in \mathbb{R}^{\mathsf{N}\times 1}$. Vector outputs $\mathbf{u}_i$ are multiplied by weight matrices $\mathbf{W}_i \in \mathbb{R}^{d\times\mathsf{N}}$ to produce vectors $\hat{\mathbf{u}}_i \in \mathbb{R}^{d\times 1}$ which are summed to produce a vector input $\mathbf{s} \in \mathbb{R}^{d\times 1}$ to the capsule in the second layer. The capsule then performs the non-linear squashing function to produce a vector output $\mathbf{e} \in \mathbb{R}^{d\times 1}$:

$$\mathbf{e} = \mathsf{squash}\left(\mathbf{s}\right) \quad ; \quad \mathbf{s} = \sum_i c_i \hat{\mathbf{u}}_i \quad ; \quad \hat{\mathbf{u}}_i = \mathbf{W}_i \mathbf{u}_i$$

where $\mathsf{squash}\left(\mathbf{s}\right) = \frac{\|\mathbf{s}\|^2}{1+\|\mathbf{s}\|^2}\frac{\mathbf{s}}{\|\mathbf{s}\|}$, and c_i are coupling coefficients determined by the routing process as presented in Algorithm 1. Because there is one capsule in the second layer, we make only one difference in the routing process proposed by Sabour et al. (2017), for which we apply the softmax in a direction from all capsules in the previous layer to each of capsules in the next layer.[1]

```
for all capsule $i \in$ the first layer do
    $b_i \leftarrow 0$
for iteration = 1, 2, ..., $m$ do
    $\mathbf{c} \leftarrow \mathsf{softmax}\left(\mathbf{b}\right)$
    $\mathbf{s} \leftarrow \sum_i c_i \hat{\mathbf{u}}_i$
    $\mathbf{e} = \mathsf{squash}\left(\mathbf{s}\right)$
    for all capsule $i \in$ the first layer do
        $b_i \leftarrow b_i + \hat{\mathbf{u}}_i \cdot \mathbf{e}$
```

Algorithm 1: The routing process is extended from Sabour et al. (2017).

We illustrate our proposed model in Figure 1 where embedding size: $k = 4$, the number of filters: $\mathsf{N} = 5$, the number of neurons within the capsules in the first layer is equal to N, and the number of neurons within the capsule in the second layer: $d = 2$. The length of the vector output $\mathbf{e}$ is used as the score for the input triple.

Formally, we define the score function f for the triple (s, r, o) as follows:

$$f\left(s, r, o\right) = \|\mathsf{capsnet}\left(g\left(\left[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o\right] * \boldsymbol{\Omega}\right)\right)\|$$

where the set of filters $\boldsymbol{\Omega}$ is shared parameters in the convolution layer; $*$ denotes a convolution operator; and capsnet denotes a capsule network operator. We use the Adam optimizer (Kingma and Ba, 2014) to train CapsE by minimizing the loss function (Trouillon et al., 2016; Nguyen et al., 2018) as follows:

$$\mathcal{L} = \sum_{(s,r,o)\in\{\mathcal{G}\cup\mathcal{G}'\}} \log\left(1 + \exp\left(-t_{(s,r,o)} \cdot f\left(s, r, o\right)\right)\right)$$

$$\text{in which, } t_{(s,r,o)} = \begin{cases} 1 \text{ for } (s, r, o) \in \mathcal{G} \\ -1 \text{ for } (s, r, o) \in \mathcal{G}' \end{cases}$$

here $\mathcal{G}$ and $\mathcal{G}'$ are collections of valid and invalid triples, respectively. $\mathcal{G}'$ is generated by corrupting valid triples in $\mathcal{G}$.

3 Knowledge graph completion evaluation

In the knowledge graph completion task (Bordes et al., 2013), the goal is to predict a missing entity given a relation and another entity, i.e, inferring a head entity s given (r, o) or inferring a tail entity o given (s, r). The results are calculated based on ranking the scores produced by the score function f on test triples.

---

[1] The softmax in the original routing process proposed by Sabour et al. (2017) is applied in another direction from each of capsules in the previous layer to all capsules in the next layer.

3.1 Experimental setup

Datasets: We use two recent benchmark datasets WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). These two datasets are created to avoid reversible relation problems, thus the prediction task becomes more realistic and hence more challenging (Toutanova and Chen, 2015). Table 1 presents the statistics of WN18RR and FB15k-237.

Dataset	#E	#R	#Triples in train/valid/test		
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 1: Statistics of the experimental datasets. #E is the number of entities. #R is the number of relations.

Evaluation protocol: Following Bordes et al. (2013), for each valid test triple (s, r, o), we replace either s or o by each of all other entities to create a set of corrupted triples. We use the "Filtered" setting protocol (Bordes et al., 2013), i.e., not taking any corrupted triples that appear in the KG into accounts. We rank the valid test triple and corrupted triples in descending order of their scores. We employ evaluation metrics: mean rank (MR), mean reciprocal rank (MRR) and Hits@10 (i.e., the proportion of the valid test triples ranking in top 10 predictions). Lower MR, higher MRR or higher Hits@10 indicate better performance. Final scores on the test set are reported for the model obtaining the highest Hits@10 on the validation set.

Training protocol: We use the common Bernoulli strategy (Wang et al., 2014; Lin et al., 2015b) when sampling invalid triples. For WN18RR, Pinter and Eisenstein (2018)[2] found a strong evidence to support the necessity of a WordNet-related semantic setup, in which they averaged pre-trained word embeddings for word surface forms within the WordNet to create synset embeddings, and then used these synset embeddings to initialize entity embeddings for training their TransE association model. We follow this evidence in using the pre-trained 100-dimensional Glove word embeddings (Pennington et al., 2014) to train a TransE model on WN18RR.

We employ the TransE and ConvKB implementations provided by Nguyen et al. (2016b) and Nguyen et al. (2018). For ConvKB, we use a new process of training up to 100 epochs and monitor the Hits@10 score after every 10 training epochs to choose optimal hyper-parameters with the Adam initial learning rate in $\{1e^{-5}, 5e^{-5}, 1e^{-4}\}$ and the number of filters N in $\{50, 100, 200, 400\}$. We obtain the highest Hits@10 scores on the validation set when using N= 400 and the initial learning rate $5e^{-5}$ on WN18RR; and N= 100 and the initial learning rate $1e^{-5}$ on FB15k-237.

Like in ConvKB, we use the same pre-trained entity and relation embeddings produced by TransE to initialize entity and relation embeddings in our CapsE for both WN18RR and FB15k-237 ($k = 100$). We set the batch size to 128, the number of neurons within the capsule in the second capsule layer to 10 ($d = 10$), and the number of iterations in the routing algorithm m in $\{1, 3, 5, 7\}$. We run CapsE up to 50 epochs and monitor the Hits@10 score after each 10 training epochs to choose optimal hyper-parameters. The highest Hits@10 scores for our CapsE on the validation set are obtained when using $m = 1$, $\mathsf{N} = 400$ and the initial learning rate at $1e^{-5}$ on WN18RR; and $m = 1$, $\mathsf{N} = 50$ and the initial learning rate at $1e^{-4}$ on FB15k-237.

3.2 Main experimental results

Table 2 compares the experimental results of our CapsE with previous state-of-the-art published results, using the same evaluation protocol. Our CapsE performs better than its closely related CNN-based model ConvKB on both experimental datasets (except Hits@10 on WN18RR and MR on FB15k-237), especially on FB15k-237 where our CapsE gains significant improvements of $0.523 - 0.418 = 0.105$ in MRR (which is about 25.1% relative improvement), and $59.3\% - 53.2\% = 6.1\%$ absolute improvement in Hits@10. Table 2 also shows that our CapsE obtains the best MR score on WN18RR and the highest MRR and Hits@10 scores on FB15k-237.

Following Bordes et al. (2013), for each relation r in FB15k-237, we calculate the averaged number η_s of head entities per tail entity and the averaged number η_o of tail entities per head entity. If η_s <1.5 and η_o <1.5, r is categorized one-to-one (1-1). If η_s <1.5 and η_o ≥1.5, r is categorized one-to-many (1-M). If η_s ≥1.5 and η_o <1.5, r is

---

[2] Pinter and Eisenstein (2018) considered WN18RR and evaluated their M3GM model only for 7 relations as they employed the inverse rule model (Dettmers et al., 2018) for 4 remaining symmetric relations. Regarding a fair comparison to other models, we use the M3GM implementation released by Pinter and Eisenstein (2018) to re-train and re-evaluate the M3GM model for all 11 relations. We thank Pinter and Eisenstein (2018) for their assistance running their code.

Method	**WN18RR**			**FB15k-237**		
	MR	MRR	H@10	MR	MRR	H@10
DISTMULT (Yang et al., 2015)	5110	0.425	49.1	254	0.241	41.9
ComplEx (Trouillon et al., 2016)	5261	**0.444**	50.7	339	0.247	42.8
ConvE (Dettmers et al., 2018)	4187	0.433	51.5	**244**	0.325	50.1
KBGAN (Cai and Wang, 2018)	–	0.213	48.1	–	0.278	45.8
M3GM (Pinter and Eisenstein, 2018)	1864	0.311	53.3	–	–	–
TransE (Bordes et al., 2013)	743*	0.245*	56.0*	347	0.294	46.5
ConvKB (Nguyen et al., 2018)	763*	0.253*	**56.7***	254*	0.418*	53.2*
Our **CapsE**	**719**	0.415	56.0	303	**0.523**	**59.3**

Table 2: Experimental results on the WN18RR and FB15k-237 test sets. Hits@10 (H@10) is reported in %. Results of DISTMULT, ComplEx and ConvE are taken from Dettmers et al. (2018). Results of TransE on FB15k-237 are taken from Nguyen et al. (2018). Our CapsE Hits@1 scores are 33.7% on WN18RR and 48.9% on FB15k-237. Formulas of MRR and Hits@1 show a strong correlation, so using Hits@1 does not really reveal any additional information for this task. The best score is in bold, while the second best score is in underline. $\star$ denotes *our new results* for TransE and ConvKB, which are better than those published by Nguyen et al. (2018).

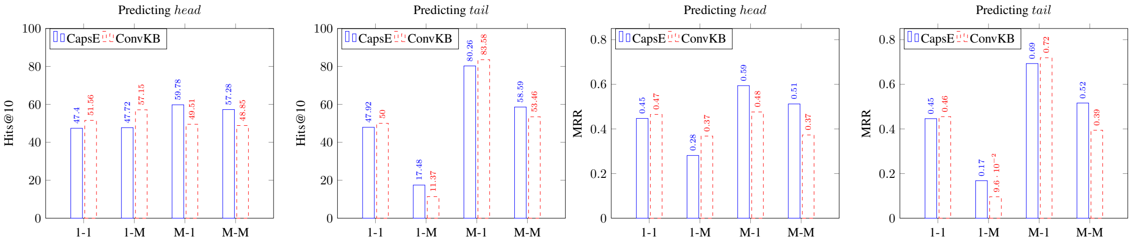


Figure 2: Hits@10 (in %) and MRR on the FB15k-237 test set w.r.t each relation category.

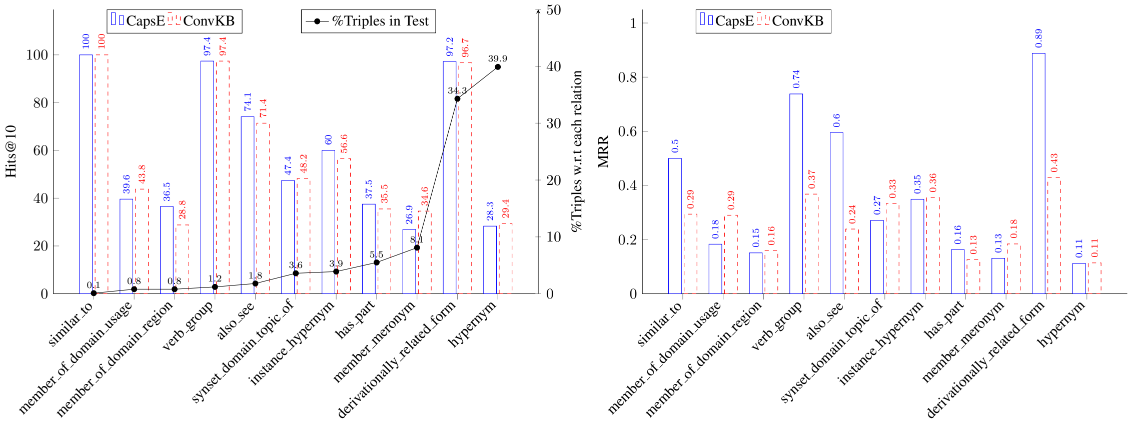


Figure 3: Hits@10 and MRR on the WN18RR test set w.r.t each relation. The right y-axis is the percentage of triples corresponding to relations.

categorized many-to-one (M-1). If $\eta_s \geq 1.5$ and $\eta_o \geq 1.5$, r is categorized many-to-many (M-M). As a result, 17, 26, 81 and 113 relations are labelled 1-1, 1-M, M-1 and M-M, respectively. And 0.9%, 6.3%, 20.5% and 72.3% of the test triples in FB15k-237 contain 1-1, 1-M, M-1 and M-M relations, respectively.

Figure 2 shows the Hits@10 and MRR results for predicting head and tail entities w.r.t each relation category on FB15k-237. CapsE works better than ConvKB in predicting entities on the "side M" of triples (e.g., predicting *head* entities in M-1 and M-M; and predicting *tail* entities in 1-M and M-M), while ConvKB performs better than CapsE in predicting entities on the "side 1" of triples (i.e., predicting *head* entities in 1-1 and 1-M; and predicting *tail* entities in 1-1 and M-1).

Figure 3 shows the Hits@10 and MRR scores w.r.t each relation on WN18RR. $also\_see$, $similar\_to$, $verb\_group$ and $derivationally\_related\_form$ are symmetric relations which can be considered as M-M relations. Our CapsE also performs better than ConvKB on these 4 M-M relations. Thus, results

m	10	20	30	40	50
1	**48.37**	**52.60**	**53.14**	**53.33**	**53.21**
3	47.78	52.34	52.93	52.99	52.86
5	47.03	52.25	45.80	45.99	45.76
7	40.46	45.36	45.79	45.85	45.93

Table 3: Hits@10 on the WN18RR validation set with $\mathsf{N} = 50$ and the initial learning rate at $1e^{-5}$ w.r.t each number of iterations in the routing algorithm m and each 10 training epochs.

shown in Figures 2 and 3 are consistent. These also imply that our CapsE would be a potential candidate for applications which contain many M-M relations such as search personalization.

We see that the length and orientation of each capsule in the first layer can also help to model the important entries in the corresponding dimension, thus CapsE can work well on the "side M" of triples where entities often appear less frequently than others appearing in the "side 1" of triples. Additionally, existing models such as DISTMULT, ComplEx and ConvE can perform well for entities with high frequency, but may not for rare entities with low frequency. These are reasons why our CapsE can be considered as the best one on FB15k-237 and it outperforms most existing models on WN18RR.

Effects of routing iterations: We study how the number of routing iterations affect the performance. Table 3 shows the Hits@10 scores on the WN18RR validation set for a comparison w.r.t each number value of the routing iterations and epochs with the number of filters $\mathsf{N} = 50$ and the Adam initial learning rate at $1e^{-5}$. We see that the best performance for each setup over each 10 epochs is obtained by setting the number m of routing iterations to 1. This indicates the opposite side for knowledge graphs compared to images. In the image classification task, setting the number m of iterations in the routing process higher than 1 helps to capture the relative positions of entities in an image (e.g., eyes, nose and mouth) properly. In contrast, this property from images may be only right for the 1-1 relations, but not for the 1-M, M-1 and M-M relations in the KGs because of the high variant of each relation type (e.g., symmetric relations) among different entities.

4 Search personalization application

Given a *user*, a submitted *query* and the *documents* returned by a search system for that query, our approach is to re-rank the returned documents so that the more relevant documents should be ranked higher. Following Vu et al. (2017), we represent the relationship between the submitted query, the user and the returned document as a *(s, r, o)*-like triple *(query, user, document)*. The triple captures how much interest a user puts on a document given a query. Thus, we can evaluate the effectiveness of our CapsE for the search personalization task.

4.1 Experimental setup

Dataset: We use the SEARCH17 dataset (Vu et al., 2017) of query logs of 106 users collected by a large-scale web search engine. A log entity consists of a user identifier, a query, top-10 ranked documents returned by the search engine and clicked documents along with the user's dwell time. Vu et al. (2017) constructed short-term (session-based) user profiles and used the profiles to personalize the returned results. They then employed the SAT criteria (Fox et al., 2005) to identify whether a returned document is relevant from the query logs as either a clicked document with a dwell time of at least 30 seconds or the last clicked document in a search session (i.e., a SAT click). After that, they assigned a $relevant$ label to a returned document if it is a SAT click and also assigned $irrelevant$ labels to the remaining top-10 documents. The rank position of the $relevant$ labeled documents is used as the ground truth to evaluate the search performance before and after re-ranking.

The dataset was uniformly split into the training, validation and test sets. This split is for the purpose of using historical data in the training set to predict new data in the test set (Vu et al., 2017). The training, validation and test sets consist of 5,658, 1,184 and 1,210 relevant (i.e., valid) triples; and 40,239, 7,882 and 8,540 irrelevant (i.e., invalid) triples, respectively.

Evaluation protocol: Our CapsE is used to re-rank the original list of documents returned by a search engine as follows: (i) We train our model and employ the trained model to calculate the score for each (s, r, o) triple. (ii) We then sort the scores in the descending order to obtain a new ranked list. To evaluate the performance of our proposed model, we use two standard evaluation metrics: mean reciprocal rank (MRR) and Hits@1.[3] For each metric, the higher value indi-

---

[3] We re-rank the list of top-10 documents returned by the

cates better ranking performance.

We compare CapsE with the following baselines using the same experimental setup: (**1**) SE: The original rank is returned by the search engine. (**2**) CI (Teevan et al., 2011): This baseline uses a personalized navigation method based on previously clicking returned documents. (**3**) SP (Bennett et al., 2012; Vu et al., 2015): A search personalization method makes use of the session-based user profiles. (**4**) Following Vu et al. (2017), we use TransE as a strong baseline model for the search personalization task. Previous work shows that the well-known embedding model TransE, despite its simplicity, obtains very competitive results for the knowledge graph completion (Lin et al., 2015a; Nickel et al., 2016b; Trouillon et al., 2016; Nguyen et al., 2016a, 2018). (**5**) The CNN-based model ConvKB is the most closely related model to our CapsE.

Embedding initialization: We follow Vu et al. (2017) to initialize user profile, query and document embeddings for the baselines TransE and ConvKB, and our CapsE.

We train a LDA topic model (Blei et al., 2003) with 200 topics only on the *relevant* documents (i.e., SAT clicks) extracted from the query logs. We then use the trained LDA model to infer the probability distribution over topics for every returned document. We use the topic proportion vector of each document as its document embedding (i.e. $k = 200$). In particular, the z^{th} element ($z = 1, 2, ..., k$) of the vector embedding for document d is: $\boldsymbol{v}_{d,z} = \mathrm{P}(z \mid d)$ where $\mathrm{P}(z \mid d)$ is the probability of the topic z given the document d.

We also represent each query by a probability distribution vector over topics. Let $\mathcal{D}_q = \{d_1, d_2, ..., d_n\}$ be the set of top n ranked documents returned for a query q (here, $n = 10$). The z^{th} element of the vector embedding for query q is defined as in (Vu et al., 2017): $\boldsymbol{v}_{q,z} = \sum_{i=1}^{n} \lambda_i \mathrm{P}(z \mid d_i)$, where $\lambda_i = \frac{\delta^{i-1}}{\sum_{j=1}^{n} \delta^{j-1}}$ is the exponential decay function of i which is the rank of d_i in D_q. And δ is the decay hyper-parameter ($0 < \delta < 1$). Following Vu et al. (2017), we use $\delta = 0.8$. Note that if we learn query and document embeddings during training, the models will overfit to the data and will not work for new queries and documents. Thus, after the initialization process, we fix (i.e., not updating) query and document embeddings during training for TransE, ConvKB and CapsE.

In addition, as mentioned by Bennett et al. (2012), the more recently clicked document expresses more about the user current search interest. Hence, we make use of the user clicked documents in the training set with the temporal weighting scheme proposed by Vu et al. (2015) to initialize user profile embeddings for the three embedding models.

Hyper-parameter tuning: For our CapsE model, we set batch size to 128, and also the number of neurons within the capsule in the second capsule layer to 10 ($d = 10$). The number of iterations in the routing algorithm is set to 1 ($m = 1$). For the training model, we use the Adam optimizer with the initial learning rate $\in \{5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}\}$. We also use ReLU as the activation function g. We select the number of filters $\mathsf{N} \in \{50, 100, 200, 400, 500\}$. We run the model up to 200 epochs and perform a grid search to choose optimal hyper-parameters on the validation set. We monitor the MRR score after each training epoch and obtain the highest MRR score on the validation set when using $\mathsf{N} = 400$ and the initial learning rate at $5e^{-5}$.

We employ the TransE and ConvKB implementations provided by Nguyen et al. (2016b) and Nguyen et al. (2018) and then follow their training protocols to tune hyper-parameters for TransE and ConvKB, respectively. We also monitor the MRR score after each training epoch and attain the highest MRR score on the validation set when using margin = 5, l_1-norm and SGD learning rate at $5e^{-3}$ for TransE; and $\mathsf{N} = 500$ and the Adam initial learning rate at $5e^{-4}$ for ConvKB.

4.2 Main results

Table 4 presents the experimental results of the baselines and our model. Embedding models TranE, ConvKB and CapsE produce better ranking performances than traditional learning-to-rank search personalization models CI and SP. This indicates a prospective strategy of expanding the triple embedding models to improve the ranking quality of the search personalization systems. In particular, our MRR and Hits@1 scores are higher than those of TransE (with relative improvements of 14.5% and 22% over TransE, respectively). Specifically, our CapsE achieves the highest performances in both MRR and Hits@1 (our improvements over all five baselines are statistically

---

search engine, so Hits@10 scores are same for all models.

Method	**MRR**	**H@1**
SE [$\star$]	0.559	38.5
CI [$\star$]	0.597	41.6
SP [$\star$]	0.631	45.2
TransE [$\star$]	0.645	48.1
TransE (ours)	0.669	50.9
ConvKB	$0.750_{+12.1\%}$	$59.9_{+17.7\%}$
Our CapsE	$\mathbf{0.766}_{+14.5\%}$	$\mathbf{62.1}_{+22.0\%}$

Table 4: Experimental results on the test set. [$\star$] denotes the results reported in (Vu et al., 2017). Hits@1 (H@1) is reported in %. In information retrieval, Hits@1 is also referred to as P@1. The subscripts denote the relative improvement over our TransE results.

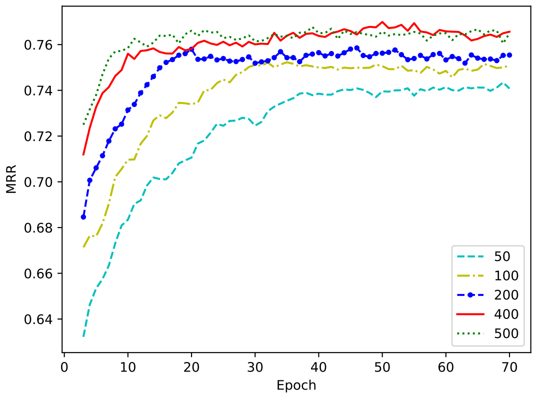


Figure 4: Learning curves on the validation set with the initial learning rate at $5e^{-5}$.

significant with $p < 0.05$ using the *paired t-test*).

To illustrate our training progress, we plot performances of CapsE on the validation set over epochs in Figure 4. We observe that the performance is improved with the increase in the number of filters since capsules can encode more useful properties for a large embedding size.

5 Related work

Other transition-based models extend TransE to additionally use projection vectors or matrices to translate embeddings of s and o into the vector space of r, such as: TransH (Wang et al., 2014), TransR (Lin et al., 2015b), TransD (Ji et al., 2015) and STransE (Nguyen et al., 2016b). Furthermore, DISTMULT (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) use a tri-linear dot product to compute the score for each triple. Moreover, ConvKB (Nguyen et al., 2018) applies convolutional neural network, in which feature maps are concatenated into a single feature vector which is then computed with a weight vector via a dot product to produce the score for the input triple. ConvKB is the most closely related model to our CapsE. See an overview of embedding models for KG completion in (Nguyen, 2017).

For search tasks, unlike classical methods, personalized search systems utilize the historical interactions between the user and the search system, such as submitted queries and clicked documents to tailor returned results to the need of that user (Teevan et al., 2005, 2009). That historical information can be used to build the *user profile*, which is crucial to an effective search personalization system. Widely used approaches consist of two separated steps: (1) building the user profile from the interactions between the user and the search system; and then (2) learning a ranking function to *re-rank* the search results using the user profile (Bennett et al., 2012; White et al., 2013; Harvey et al., 2013; Vu et al., 2015). The general goal is to re-rank the documents returned by the search system in such a way that the more relevant documents are ranked higher. In this case, apart from the user profile, dozens of other features have been proposed as the input of a learning-to-rank algorithm (Bennett et al., 2012; White et al., 2013). Alternatively, Vu et al. (2017) modeled the potential *user*-oriented relationship between the submitted query and the returned document by applying TransE to reward higher scores for more relevant documents (e.g., clicked documents). They achieved better performances than the standard ranker as well as competitive search personalization baselines (Teevan et al., 2011; Bennett et al., 2012; Vu et al., 2015).

6 Conclusion

We propose CapsE—a novel embedding model using the capsule network to model relationship triples for knowledge graph completion and search personalization. Experimental results show that our CapsE outperforms other state-of-the-art models on two benchmark datasets WN18RR and FB15k-237 for the knowledge graph completion. We then show the effectiveness of our CapsE for the search personalization, in which CapsE outperforms the competitive baselines on the dataset SEARCH17 of the web search query logs. In addition, our CapsE is capable to effectively model many-to-many relationships. Our code is available at: https://github.com/daiquocnguyen/CapsE.

Acknowledgements

This research was partially supported by the ARC Discovery Projects DP150100031 and DP160103934. The authors thank Yuval Pinter for assisting us in running his code.

References

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 185–194.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page to appear.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168.

Morgan Harvey, Fabio Crestani, and Mark J. Carman. 2013. Building user profiles from topic models for personalised search. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2309–2314.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Learning*, pages 2181–2187.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333.

Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint*, arXiv:1703.08098.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016a. Neighborhood Mixture Model for Knowledge Base Completion. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 40–50.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016b. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Yuval Pinter and Jacob Eisenstein. 2018. Predicting Semantic Relations using Global Graph Properties. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1751.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 449–456.

Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 85–94.

Jaime Teevan, Meredith Ringel Morris, and Steve Bush. 2009. Discovering and using groups to improve personalized search. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 15–24.

Kristina Toutanova and Danqi Chen. 2015. Observed Versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080.

Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *Proceedings of the European Conference on Information Retrieval*, pages 598–604.

Thanh Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *Proceedings of the European Conference on Information Retrieval*, pages 605–616.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-based Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526.

Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the World Wide Web conference*, pages 1411–1420.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*.

Partial Or Complete, That Is The Question

Qiang Ning,[1] **Hangfeng He,**[2] **Chuchu Fan,**[1] **Dan Roth**[1,2]
[1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
[2]Department of Computer and Information Science, University of Pennsylvania
`{qning2,cfan10}@illinois.edu`, `{hangfeng,danroth}@seas.upenn.edu`

Abstract

For many structured learning tasks, the data annotation process is complex and costly. Existing annotation schemes usually aim at acquiring *completely* annotated structures, under the common perception that *partial* structures are of low quality and could hurt the learning process. This paper questions this common perception, motivated by the fact that structures consist of *interdependent* sets of variables. Thus, given a fixed budget, partly annotating each structure may provide the same level of supervision, while allowing for more structures to be annotated. We provide an information theoretic formulation for this perspective and use it, in the context of three diverse structured learning tasks, to show that learning from partial structures can *sometimes* outperform learning from complete ones. Our findings may provide important insights into structured data annotation schemes and could support progress in learning protocols for structured tasks.

1 Introduction

Many machine learning tasks require structured outputs, and the goal is to assign values to a set of variables coherently. Specifically, the variables in a structure need to satisfy some global properties required by the task. An important implication is that once some variables are determined, the values taken by other variables are constrained. For instance, in the temporal relation extraction problem in Fig. 1a, if *met* happened before *leaving* and *leaving* happened on *Thursday*, then we know that *met* must either be before *Thursday* ("met (1)") or has to happen on *Thursday*, too ("met (2)") (Ning et al., 2018a). Similarly, in the semantic frame of the predicate *gave* (Kingsbury and Palmer, 2002) in Fig. 1b, if *the boy* is ARG0 (short for argument 0), then it rules out the possibility of *a frog* or *to the girl* taking the same role. Figure 1c further shows an example of part-labeling of images (Choi et al., 2018); given the position of FOREHEAD and LEFT_EYE of the cat in the picture, we roughly know that its NECK should be somewhere in the red solid box, while the blue dashed box is likely to be wrong.

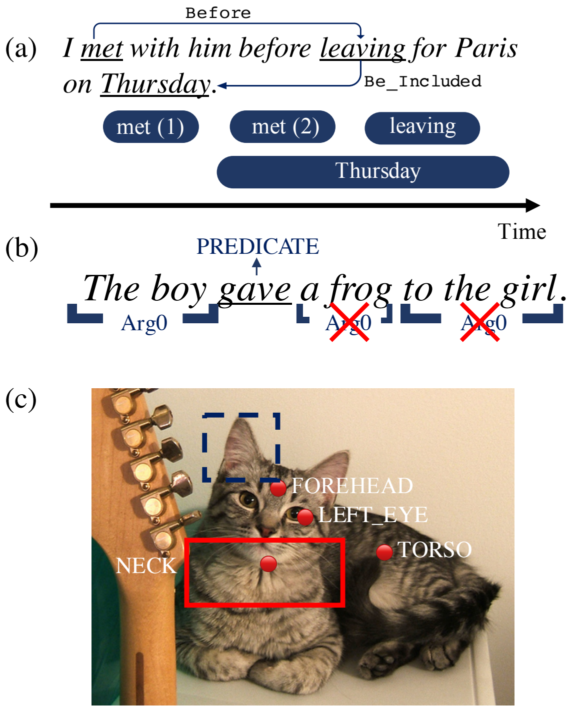


Figure 1: Due to the inherent structural constraints of each task, individual instances therein put restrictions on others. (a) The temporal relation between *met* and *Thursday* has to be BEFORE ("met (1)") or BE_INCLUDED ("met (2)"). (b) The argument roles of *a frog* and *to the girl* cannot be ARG0 anymore. (c) Given the position of the cat's FOREHEAD and LEFT_EYE, a rough estimate of its NECK can be the red solid box rather than the blue dashed box.

Data annotation for these structured tasks is complex and costly, thus requiring one to make the most of a given budget. This issue has been

Proceedings of NAACL-HLT 2019, pages 2190–2200
Minneapolis, Minnesota, June 2 - June 7, 2019.

investigated for decades from the perspective of active learning for classification tasks (Angluin, 1988; Atlas et al., 1990; Lewis and Gale, 1994) and for structured tasks (Roth and Small, 2006a,b, 2008; Hu et al., 2019). While active learning aims at selecting the next structure to label, we try to investigate, from a different perspective, whether we should annotate each structure completely or partially. Conventional annotation schemes typically require *complete* structures, under the common perception that partial annotation could adversely affect the performance of the learning algorithm. But note that partial annotations will allow for more structures to be annotated (see Fig. 2). Therefore, a fair comparison should be done while maintaining a fixed annotation budget, which was not done before. Moreover, even if partial annotation leads to comparable learning performance to conventional complete schemes, it provides more flexibility in data annotation.

Another potential benefit of partial annotation is that it imposes constraints on the remaining parts of a structure. As illustrated by Fig. 1, with partial annotations, we already have some knowledge about the unannotated parts. Therefore, further annotations of these variables may use the available budget less efficiently; this effect was first discussed in Ning et al. (2018c). Motivated by the observations in Figs. 1-2, we think it is important to study partialness systematically, before we hastily assume that *completeness* should always be favored in data collection.

To study whether the above benefits of partialness can offset its weakness for learning, **our first contribution** is the proposal of *early stopping partial annotation* (ESPA) scheme, which randomly picks up instances to label in the beginning, and stops before a structure is completed. We do not claim that ESPA should *always* be preferred; instead, it serves as an *alternative* to conventional, complete annotation schemes that we should keep in mind, because, as we show later, it can be comparable to (and sometimes even better than) complete annotation schemes. ESPA is straightforward to implement even in crowdsourcing; instances to annotate can be selected *offline* and distributed to crowdsourcers; this can be contrasted with the difficulties of implementing active learning protocols in these settings (Ambati et al., 2010; Laws et al., 2011). We think that ESPA is a good representative for a systematic study of partialness.

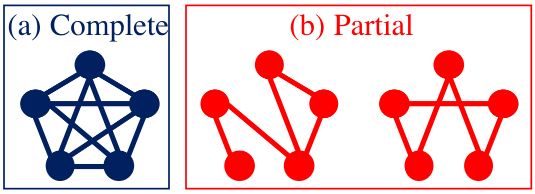


Figure 2: If we need training data for a graph labeling task (assuming the gold values for the nodes are given) and our annotation budget allows us to annotate, for instance, 10 edges in total, we could (a) completely annotate one graph (and then we run out of budget), or (b) partially annotate two graphs.

Our second contribution is the development of an information theoretic formulation to explain the benefit of ESPA (Sec. 2), which we further demonstrate via three structured learning tasks in Sec. 4: temporal relation (TempRel) extraction (UzZaman et al., 2013), semantic role classification (SRC),[1] and shallow parsing (Tjong Kim Sang and Buchholz, 2000). These tasks are chosen because they each represent a wide spectrum of structures that we will detail later. **As a byproduct**, we extend constraint-driven learning (CoDL) (Chang et al., 2007) to cope with partially annotated structures (Sec. 3); we call the algorithm *Structured Self-learning with Partial ANnotations* (SSPAN) to distinguish it from CoDL.[2]

We believe in the importance of work in this direction. *First, partialness is inevitable in practice*, either by mistake or by choice, so our theoretical analysis can provide unique insight into understanding partialness. *Second, it opens up opportunities for new annotation schemes.* Instead of considering partial annotations as a compromise, we can in fact annotate partial data *intentionally*, allowing us to design favorable guidelines and collect more important annotations at a cheaper price. Many recent datasets that were collected via crowdsourcing are already partial, and this paper provides some theoretical foundations for them. Furthermore, the setting described here addresses natural scenarios where only partial, indirect supervision is available, as in *Incidental Supervision*

---

[1] A subtask of semantic role labeling (SRL) (Palmer et al., 2010) that only classifies the role of an argument.

[2] There has been many works on learning from partial annotations, which we review in Sec. 3. SSPAN is only an experimental choice in demonstrating ESPA. Whether SSPAN is better than other algorithms is out of the scope here, and a better algorithm for ESPA will only strengthen the claims in this paper.

(Roth, 2017), and this paper begins to provide theoretical understanding for this paradigm, too. Further discussions can be found in Sec. 5.

It is important to clarify that we assume uniform cost over individual annotations (that is, all edges in Fig. 2 cost equally), often the default setting in crowdsourcing. We realize that the annotation difficulty can vary a lot in practice, sometimes incurring different costs. To address this issue, we randomly select instances to label so that on average, the cost is uniform. We agree that, even with this randomness, there could still be situations where the assumption does not hold, but we leave it for future studies, possibly in the context of active learning schemes.

2 ESPA: Early Stopping Partial Annotation

In this section, we study whether the effect demonstrated by the examples in Fig. 1 exists in general. First, we formally define *structure* and *annotation*.

Definition 1. *A structure of size* d *is a vector of random variables (RV)* $\mathbf{Y} = [Y_1, \ldots, Y_d] \in C(\mathcal{L}^d)$*, where* $\mathcal{L} = \{\ell_1, \ldots, \ell_{|\mathcal{L}|}\}$ *is the label set for each variable and* $C(\mathcal{L}^d) \subseteq \mathcal{L}^d$ *represents the constraints imposed by this type of structure.*

It is necessary to model a structure as a set of *random* variables because when it is not completely annotated, there is still uncertainty in the annotation assignment. To study partial annotations, we introduce the following:

Definition 2. *A* k*-step annotation* ($0 \le k \le d$) *is a vector of RVs* $\mathbf{A}_k = [A_{k,1}, \ldots, A_{k,d}] \in (\mathcal{L} \cup \sqcap)^d$ *where* $\sqcap$ *is a special character for null, such that*

$$\sum_{i=1}^{d} \mathbb{1}(A_{k,i} \neq \sqcap) = k, \qquad (1)$$

$$P(\mathbf{Y}|\mathbf{A}_k = \mathbf{a}_k) = P(\mathbf{Y}|Y_j = a_{k,j}, j \in \mathcal{J}), \quad (2)$$

where $\mathcal{J}$ *is the set of indices that* $a_{k,j} \neq \sqcap$.

Eq. (1) means that, in total, k variables are already annotated at step k. Obviously, $\mathbf{A}_0$ means that no variables are labeled, and $\mathbf{A}_d$ means that all variables in $\mathbf{Y}$ are determined. $\mathbf{A}_k$ is what we call a k-step ESPA, so hereafter we use k/d to represent annotation completeness. Eq. (2) assumes no annotation mistakes, so if the i-th variable is labeled, then Y_i must be the same as $A_{k,i}$.

To measure the theoretical benefit of $\mathbf{A}_k$, we propose the following quantity

$$I_k = \log |C(\mathcal{L}^d)| - E\left[\log f(\mathbf{a}_k)\right] \qquad (3)$$

for $k = 0, \ldots, d$, where $f(\mathbf{a}_k) = |\{\mathbf{y} \in C(\mathcal{L}^d) : \mathrm{P}(\mathbf{y}|\mathbf{a}_k) > 0\}|$ is the total number of structures in $C(\mathcal{L}^d)$ that are still valid given $\mathbf{A}_k = \mathbf{a}_k$. Since we assume that the labeled variables in $\mathbf{A}_k$ are selected uniformly randomly, $E[\cdot]$ is simply the average of $\log f(\mathbf{a}_k)$. When $k = 0$, $f(\mathbf{a}_k) \equiv C(\mathcal{L}^d)$ and $I_0 \equiv 0$; as k increases, I_k increases since the structure has more and more variables labeled; finally, when $k = d$, the structure is fully determined and $I_d \equiv \log |C(\mathcal{L}^d)|$. The first-order finite difference, $I_k - I_{k-1}$, is the benefit brought by annotating an additional variable at step k; if I_k is concave (i.e., a decaying $I_k - I_{k-1}$), the benefit from a new annotation attenuates, suggesting the potential benefit of the ESPA strategy.

In an extreme case where the structure is so strong that it requires all individual variables to share the same label, then labeling any variable is sufficient for determining the entire structure. Intuitively, we do not need to annotate more than one variable. Our I_k quantity can support this intuition: The structural constraint, $C(\mathcal{L}^d)$, contains only $|\mathcal{L}|$ elements: $\{[\ell_i, \ell_i, \ldots, \ell_i]\}_{i=1}^{|\mathcal{L}|}$, so $I_0 = 0$, and $I_1 = \cdots = I_d = \log |\mathcal{L}|$. Since I_k does not increase at all when $k >= 1$, we should adopt first-step annotation $\mathbf{A}_1$. Another extreme case is that of a trivial structure that has no constraints (i.e., $C(\mathcal{Y}^d) = \mathcal{Y}^d$). The annotation of all variables are independent and we gain no advantage from skipping any variables. This intuition can be supported by our I_k analysis as well: Since $I_k = k \log |\mathcal{L}|$, $\forall k = 0, 1, \ldots, d$, I_k is linear and all steps contribute equally to improving I_k by $\log |\mathcal{L}|$; therefore ESPA is not necessary.

Real-world structures are often not as trivial as the two extreme cases above, but I_k can still serve as a guideline to help determine whether it is beneficial to use ESPA. We next discuss three diverse types of structures and how to obtain I_k for them.

Example 1. *The ranking problem is an important machine learning task and often depends on pairwise comparisons, for which the label set is* $\mathcal{L} = \{<, >\}$. *For a ranking problem with* n *items, there are* $d = n(n-1)/2$ *pairwise comparisons in total. Its structure is a chain following the transitivity constraints, i.e., if* $A < B$ *and* $B < C$*, then* $A < C$.

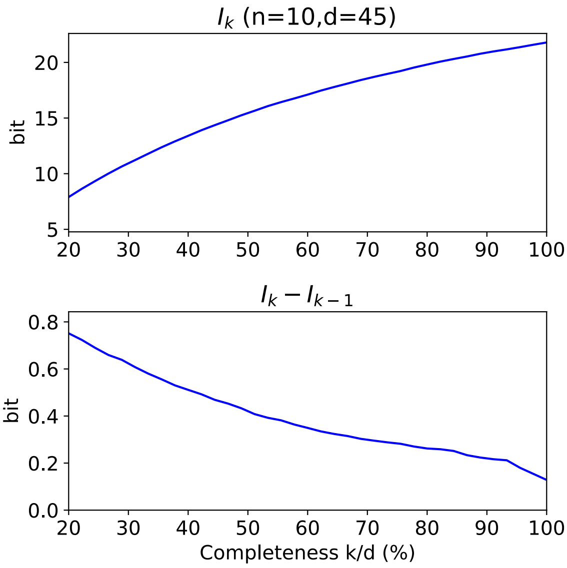


Figure 3: The mutual information between the chain structure and its k-step ESPA, I_k, is concave, suggesting possible benefit of using ESPA. In the simulation, there are $n = 10$ items in the chain and thus $d = 45$ pairs, k of which are labeled. The values of I_k's, as defined by Eq. (3), were obtained through averaging 1000 experiments. We use base-2 logarithm and the unit on y-axis is thus "bit".

A k-step ESPA $\mathbf{A}_k$ for a chain means that only k (out of d) pairs are compared and labeled, resulting in a directed acyclic graph (DAG). In this case, $f(\mathbf{a}_k)$ is actually counting the number of linear extensions of the DAG, which is known to be #P-complete (Brightwell and Winkler, 1991), so we do not have a closed-form solution to I_k. In practice, however, we can use the Kahn's algorithm and backtracking to simulate with a relatively small n, as shown by Fig. 3, where $n = 10$ and I_k was obtained through averaging 1000 random simulations. I_k is concave, as reflected by the downward shape of $I_k - I_{k-1}$. Therefore, new annotations are less and less efficient for the chain structure, suggesting the usage of ESPA.

Example 2. *The general assignment problem requires assigning d agents to d′ tasks such that the agent nodes and the task nodes form a bipartite graph (without loss of generality, assume d ≤ d′). That is, an agent can handle exactly one task, and each task can only be handled by at most one agent. Then from the agents' point of view, the label set for each of them is* $\mathcal{L} = \{1, 2, \ldots, d'\}$*, denoting the task assigned to the agent.*

A k-step ESPA $\mathbf{A}_k$ for this problem means that k agents are already assigned with tasks, and $f(\mathbf{a}_k)$ is to count the valid assignments of the remaining tasks to the remaining $d - k$ agents, to which we have closed-form solutions: $f(\mathbf{a}_k) = \frac{(d'-k)!}{(d'-d)!}, \forall \mathbf{a}_k$. According to Eq. (3), $I_k = \log \frac{d'!}{(d'-k)!}$ regardless of d or the distribution of $\mathbf{A}_k$, and is concave (Fig. 4 shows an example of it when $d = 4, d' = 10$).

Example 3. *Sequence tagging is an important NLP problem, where the tags of tokens are interdependent. Take chunking as an example. A basic scheme is for each token to choose from three labels, B(egin), I(nside), and O(utside), to represent text chunks in a sentence. That is,* $\mathcal{L} = \{B, I, O\}$*. Obviously, O cannot be immediately followed by I.*

Let d be the number of tokens in a sentence. A k-step ESPA $\mathbf{A}_k$ for chunking means that k tokens are already labeled by B/I/O, and $f(\mathbf{a}_k)$ counts the valid BIO sequences that do not violate those existing annotations. Again, as far as we know, there is no closed-form solution to $f(\mathbf{a}_k)$ and I_k, but in practice, we can use dynamic programming to obtain $f(\mathbf{a}_k)$ and then I_k using Eq. (3). We set $d = 10$ and show $I_k - I_{k-1}$ for this task in Fig. 4, where we observe the same effect we see in previous examples: The benefit provided by labeling a new token in the structure attenuates.

Interestingly, based on Fig. 4, we find that **the slope of $I_k - I_{k-1}$ may be a good measure of the "tightness" or "strength" of a structure**. When there is no structure at all, the curve is flat (black). The BIO structure is intuitively simple, and it indeed has the flattest slope among the three structured tasks (purple). When the structure is a chain, the level of uncertainty goes down rapidly with every single annotation (think of standard sorting algorithms); the constraint is intuitively strong and in Fig. 4, it indeed has a steep slope (blue).

Finally, we want to emphasize that the definition of I_k in Eq. (3) is in fact backed by information theory. When we do not have prior information about $\mathbf{Y}$, we can assume that $\mathbf{Y}$ follows a uniform distribution over $C(\mathcal{L}^d)$. Then, I_k is essentially the mutual information between structure $\mathbf{Y}$ and annotation $\mathbf{A}_k$, $I(\mathbf{Y}; \mathbf{A}_k)$:

$$\begin{aligned} I(\mathbf{Y}; \mathbf{A}_k) &= H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{A}_k) \\ &= \log |C(\mathcal{L}^d)| - E\left[H(\mathbf{Y}|\mathbf{A}_k = \mathbf{a}_k)\right] \\ &= \log |C(\mathcal{L}^d)| - E\left[\log f(\mathbf{a}_k)\right], \end{aligned}$$

where $H(\cdot)$ is the entropy function. This is an important discovery, since it points out a new way to view a structure and its annotations. It may be useful for studying active learning methods for structured tasks, and other annotation phenomena such as noisy annotations. The usage of mutual information also aligns well with the information bottleneck framework (Shamir et al., 2010; Shwartz-Ziv and Tishby, 2017; Yu and Principe, 2018), although a more recent paper challenges the interpretation of information bottleneck (Saxe et al., 2018).

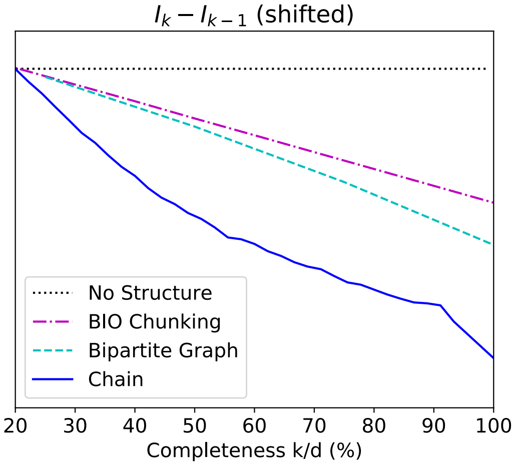


Figure 4: The $I_k - I_{k-1}$ curves from several different structures. The curves are shifted to almost the same starting point for better visualization, so the Y-Axis grid is not shown. The curve for "Chain" was obtained via simulations, and the other curves all have closed-form formulations.

3 Learning from Partial Structures

So far, we have been advocating the ESPA strategy to maximize the information we can get from a fixed budget. Since early stopping leads to partial annotations, one missing component before we can benefit from it is an approach to learning from partial structures. In this study, we assume the existence of a relatively small but complete dataset that can provide a good initialization for learning from a partial dataset, which is very similar to semi-supervised learning (SSL). SSL, in its most standard form, studies the combined usage of a labeled set $\mathcal{T} = \{(x_i, y_i)\}_i$ and an unlabeled set $\mathcal{U} = \{x_j\}_j$, where the x's are instances and y's are the corresponding labels. SSL gains information about $p(x)$ through $\mathcal{U}$, which may improve the estimation of $p(y|x)$. Specific algorithms range from self-training (Scudder, 1965; Yarowsky, 1995), co-training (Blum and Mitchell, 1998), generative models (Nigam et al., 2000), to transductive SVM (Joachims, 1999) etc., among which one of the most basic algorithms is Expectation-Maximization (EM) (Dempster et al., 1977). By treating them as hidden variables, EM "marginalizes" out the missing labels of $\mathcal{U}$ via expectation (i.e., soft EM) or maximization (i.e., hard EM). For structured ML tasks, soft and hard EMs turn into posterior regularization (PR) (Ganchev et al., 2010) and constraint-driven learning (CoDL) (Chang et al., 2007), respectively.

Unlike unlabeled data, the partially annotated structures caused by early stopping urge us to gain information not only about $p(x)$, but also from their labeled parts. There have been many existing work along this line (Tsuboi et al., 2008; Fernandes and Brefeld, 2011; Hovy and Hovy, 2012; Lou and Hamprecht, 2012), but in this paper, we decide to extend CoDL to cope with partial annotations due to two reasons. First, CoDL, which itself can be viewed as an extension of self-training to *structured* learning, is a wrapper algorithm having wide applications. Second, as its name suggests, CoDL learns from $\mathcal{U}$ by guidance of constraints, so partial annotations in $\mathcal{U}$ are technically easy to be added as extra equality constraints.

Algorithm 1 describes our *Structured Self-learning with Partial ANnotations* (SSPAN) algorithm that learns a model $\mathcal{H}$. The same as CoDL, SSPAN is a wrapper algorithm requiring two components: LEARN and INFERENCE. LEARN attempts to estimate the *local* decision function for each individual instance regardless of the *global* constraints, while INFERENCE takes those local decisions and performs a *global* inference. Lines 3-9 are the procedure of self-training, which iteratively completes the missing annotations in $\mathcal{P}$ and learns from both $\mathcal{T}$ and the completed version of $\mathcal{P}$ (i.e., $\tilde{\mathcal{P}}$).[3] Line 6 requires that the inference follows the structural constraints inherently in the task, turning the algorithm into CoDL; Line 7 enforces those partial annotations in $\mathbf{a}_i$, further turning it into SSPAN. In practice, INFERENCE can be realized by the Viterbi or beam search algorithm in sequence tagging, or more generally, by Integer Linear Programming (ILP)

---

[3] Line 9 can be interpreted in different ways, either as $\mathcal{T} \cup \tilde{\mathcal{P}}$ (adopted in this work) or as a weighted combination of LEARN($\mathcal{T}$) and LEARN($\tilde{\mathcal{P}}$) (adopted by (Chang et al., 2007)).

(Punyakanok et al., 2005); either way, the partial constraints of Line 7 can be easily incorporated.

Algorithm 1: Structured Self-learning with Partial Annotations (SSPAN)

```
   Input: $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}, \mathcal{P} = \{(\mathbf{x}_i, \mathbf{a}_i)\}_{i=N+1}^{N+M}$
1  Initialize $\mathcal{H} = \text{LEARN}(\mathcal{T})$
2  while convergence criteria not satisfied do
3      $\tilde{\mathcal{P}} = \emptyset$
4      foreach $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{P}$ do
5          $\hat{\mathbf{y}}_i$ = INFERENCE($\mathbf{x}_i; \mathcal{H}$), such that
6                  ◇ $\hat{\mathbf{y}}_i \in C(\mathcal{Y}^d)$
7                  ◇ $\hat{y}_{i,j} = a_{i,j}, \forall a_{i,j} \neq \sqcap$
8          $\tilde{\mathcal{P}} = \tilde{\mathcal{P}} \cup \{(\mathbf{x}_i, \hat{\mathbf{y}}_i)\}$
9      $\mathcal{H} = \text{LEARN}(\mathcal{T} + \tilde{\mathcal{P}})$
10 return $\mathcal{H}$
```

4 Experiment

In Sec. 2, we argued from an information theoretic view that ESPA is beneficial for structured tasks if we have a fixed annotation resource. We then proposed SSPAN in Sec. 3 to learn from the resulting partial structures. **However**, on one hand, there is still a gap between the I_k analysis and the actual system performance; on the other hand, whether the benefit can be realized in practice also depends on how effective the algorithm exploits partial annotations. Therefore, it remains to be seen how ESPA works in practice. Here we use three NLP tasks: temporal relation (TempRel) extraction, semantic role classification (SRC), and shallow parsing, analogous to the chain, assignment, and BIO structures, respectively.

For all tasks, we compare the following two schemes in Fig. 5, where we use graph structures for demonstration. Initially, we have a relatively small but complete dataset $\mathcal{T}_0$, an unannotated dataset $\mathcal{U}_0$, and some budget to annotate $\mathcal{U}_0$. The conventional scheme I, also our baseline here, is to annotate each structure completely before randomly picking up the next one. Due to the limited budget, some $\mathcal{U}_0$ remain untouched (denoted by $\mathcal{U}$). The proposed scheme II adopts ESPA so that all structures at hand are annotated but only partially. For fair comparisons, we use CoDL to incorporate $\mathcal{U}$ into scheme I as well. Finally, the systems trained on the dataset from I/II via CoDL/SSPAN are evaluated on unseen but complete testset $\mathcal{T}_{test}$. Note that because ESPA is a new annotation scheme, there exists no dataset collected this way. We use existing complete datasets and randomly throw out some annotations to mimic ESPA in the following. Due to the randomness in selecting which structures/instances to keep in scheme I/II, we repeat the whole process multiple times and report the mean F_1. The budget, defined as the total number of individual instances that can be annotated, ranges from 10% to 100% with a stepsize of 10%, where x% means x% of all instances in $\mathcal{U}_0$ can be annotated.

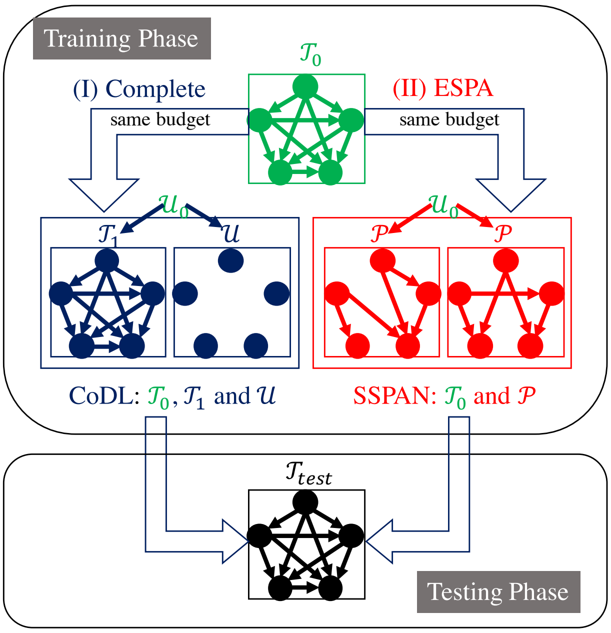


Figure 5: The two annotation schemes we compare in Sec. 4. $\mathcal{T}$, $\mathcal{P}$, and $\mathcal{U}$ denote complete, partial, and empty structures, respectively. Both schemes start with a complete and relatively small dataset and an unannotated dataset (green). (I) Conventional complete annotation scheme (blue). (II) The proposed ESPA scheme (red). Finally, they are tested on an unseen and complete dataset (black).

4.1 Temporal Relation Extraction

Temporal relations (TempRel) are a type of important relations representing the temporal ordering of events described by natural language text. That is to answer questions like which event happens earlier or later in time (see Fig. 1a). Since time is physically one-dimensional, if *A* is before *B* and *B* is also before *C*, then *A* must be before *C*. In practice, the label set for TempRels can be more complex, e.g., with labels such as SIMULTANEOUS and VAGUE, but the structure can still be represented by transitivity constraints (see Table 1 of (Ning et al., 2018a)), which can be viewed as an analogy of the chain structure in Example 1.

To avoid missing relations, annotators are required to exhaustively label every pair of events in a document (i.e., the complete annotation scheme), so it is necessary to study ESPA in this context. Here we adopt the MATRES dataset (Ning et al., 2018b) for its better inter-annotator agreement and relatively large size.

Specifically, we use 35 documents as $\mathcal{T}_0$ (the TimeBank-Dense section,[4] 147 documents as $\mathcal{U}_0$ (the TimeBank section minus those documents in $\mathcal{T}_0$), and the Platinum section (a benchmark testset of 20 documents with 1K TempRels) as $\mathcal{T}_{test}$. Note that both schemes I and II are mimicked by downsampling the original annotations in MATRES, where the budget is defined as the total number of TempRels that are kept. Following CogCompTime (Ning et al., 2018d), we choose the same features and sparse-averaged perceptron algorithm as the LEARN component and ILP as INFERENCE for SSPAN.

4.2 Semantic Role Classification (SRC)

Semantic role labeling (SRL) is to represent the semantic meanings of language and answer questions like *Who did What to Whom* and *When, Where, How* (Palmer et al., 2010). Semantic Role Classification (SRC) is a subtask of SRL, which assumes gold predicates and argument chunks and only classifies the semantic role of each argument. We use the Verb SRL dataset provided by the CoNLL-2005 shared task (Carreras and Màrquez, 2005), where the semantic roles include numbered arguments, e.g., ARG0 and ARG1, and argument modifiers, e.g., location (AM-LOC), temporal (AM-TMP), and manner (AM-MNR) (see PropBank (Kingsbury and Palmer, 2002)). The structural constraints for SRC is that each argument can be assigned to exactly one semantic role, and the same role cannot appear twice for a single verb, so SRC is an assignment problem as in Example 2.

Specifically, we use the Wall Street Journal (WSJ) part of Penn TreeBank III (Marcus et al., 1993). We randomly select 700 sentences from the Sec. 24 of WSJ, among which 100 sentences as $\mathcal{T}_0$ and 600 sentences as $\mathcal{U}_0$. Our $\mathcal{T}_{test}$ is 5700 sentences (about 40K arguments) from Secs. 00, 01, 23. The budget here is defined as the total number of the arguments. We adopt the SRL system in CogCompNLP (Khashabi et al., 2018) and uses the sparse averaged perceptron as LEARN and ILP as INFERENCE.

4.3 Shallow Parsing

Shallow parsing, also referred as *chunking*, is a fundamental NLP task to identify constituents in a sentence, such as noun phrases (NP), verb phrases (VP), and adjective phrases (ADJP), which can be viewed as extending the standard BIO structure in Example 3 with different chunk types: B-NP, I-NP, B-VP, I-VP, B-ADJP, I-ADJP, . . . , O.

We use the chunking dataset provided by the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). Specifically, we use 2K tokens' annotations as $\mathcal{T}_0$, 14K tokens as $\mathcal{U}_0$, and the benchmark testset (25K tokens) as $\mathcal{T}_{test}$. The budget here is defined as the total number of tokens' BIO labels. The algorithm we use here is the chunker provided in CogCompNLP, where the LEARN component is the sparse averaged perceptron and the INFERENCE is described in (Punyakanok and Roth, 2001).

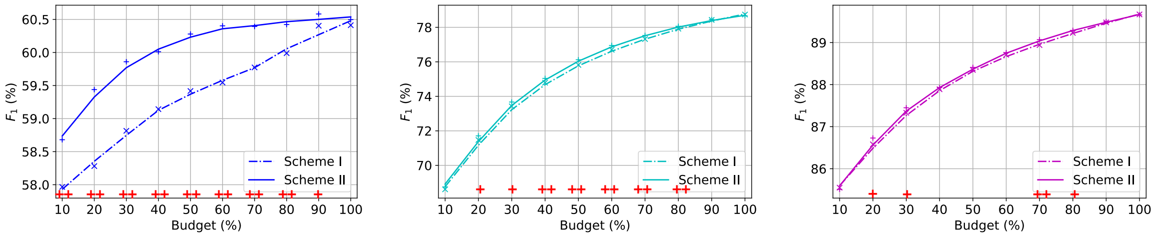
4.4 Results

We compare the F_1 performances of all three tasks in Fig. 6, averaged from 50 experiments with different randomizations. As the budget increases, the system F_1 increases for both schemes I and II in all three tasks, which confirms the capability of the proposed SSPAN framework to learn from partial structures. When the budget is 100% (i.e., the entire $\mathcal{U}_0$ is annotated), schemes I and II have negligible differences; when the budget is not large enough to cover the entire $\mathcal{U}_0$, scheme II is consistently better than I in all tasks, which follows our expectations based on the I_k analysis. The strict improvement for all budget ratios indicates that the observation is definitely not by chance.

Figure 7 further compares the improvement from I to II across tasks. When the budget goes down from 100%, the advantage of ESPA is more prominent; but when the budget is too low, the quality of $\tilde{\mathcal{P}}$ degrades and hurts the performance of SSPAN, leading to roughly hill-shaped curves in Fig. 7. We have also conjectured based on Fig. 4 that the structure strength goes up from BIO chunks, to bipartite graphs, and to chains; interestingly, the improvement brought by ESPA is consistent with this order.

---

[4] The original TimeBank-Dense section contains 36 documents, but in collecting MATRES, one of the documents was filtered out because it contained no TempRels between main-axis events.



(a) Temporal Relation Extraction (b) SRC (c) Shallow Parsing

Figure 6: Comparison of the baseline, complete annotation scheme and the proposed ESPA scheme (See I & II in Fig. 5) under three structured learning tasks (note the scale difference). Each F_1 value is the average of 50 experiments, and each curve is based on corresponding F_1 values smoothed by Savitzky-Golay filters. We can see that scheme II is consistently better than scheme I. Per the Wilcoxon rank-sum test, the significance levels at each given budget are shown on the x-axes, where $+$ and $++$ mean $p < 5\%$ and $p < 1\%$, respectively.

Admittedly, the improvement, albeit statistically significant, is small, but it does not diminish the contribution of this paper: Our goal is to remind people that the ESPA scheme (or more generally, partialness) is, at the least, comparable to (or sometimes even better than) complete annotation schemes. Also, the comparison here is in fact unfair to the partial scheme II, because we assume equal cost for both schemes, although it often costs less in a partial scheme as a large problem is decomposed into smaller parts. Therefore, the results shown here implies that the information theoretical benefit of partialness can possibly offset its disadvantages for learning.

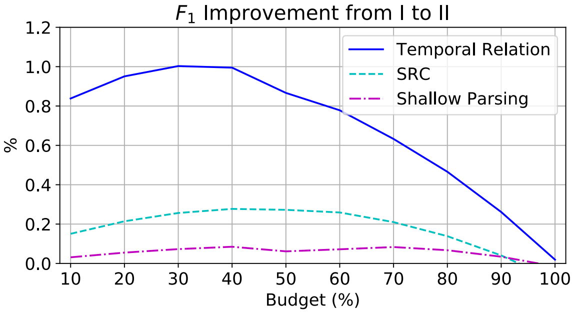


Figure 7: The improvement of F_1 brought by ESPA for each task in Fig. 6. Note that we conjectured earlier in Fig. 4 that the BIO structure is the weakest among the three, which is consistent with the fact that shallow parsing benefits the least from ESPA.

5 Dicussion and Conclusion

In this paper, we investigate a less studied, yet important question for structured learning: Given a limited annotation budget (either in time or money), which strategy is better, completely annotating each structure until the budget runs out, or annotating more structures at the cost of leaving some of them partially annotated? Neubig and Mori (2010) investigated this issue specifically in annotating word boundaries and pronunciations for Japanese. Instead of annotating full sentences, they proposed to annotate only some words in a sentence (i.e., partially) that can be chosen heuristically (e.g., skip those that we have seen or those low frequency words). Conceptually, Neubig and Mori (2010) is an active learning work, with the understanding that if the order of annotation is deliberately designed, better learning can be achieved. The current paper addresses the problem from a different angle: Even without active learning, can we still answer the question above?

The observation driving our questions is that when annotating a particular structure, the labels of the yet to be labeled variables may already be constrained by previous annotations and carry less information than those in a totally new structure. Therefore, we systematically study the ESPA scheme – stop annotating a given structure before it is completed and continue annotating another new structure.

An important notion is annotation *cost*. Throughout the paper we have an ideal assumption that the cost is linear in the total number of annotations, but in practice the case can be more complicated. First, the actual cost of each individual annotation may vary across different instances. We try to eliminate this issue by enforcing random selection of annotation instances, rather than allowing the annotators to select arbitrarily by themselves. This strategy may be useful in practice as well, to avoid people only annotating easy cases. Second, even if we only require labeling partial structures, it is likely that the annotator still needs to comprehend the entire structure, incurring additional cost (usually in terms of time). This issue, however, is not addressed in this paper.

Using this definition of cost, we provide a theoretical analysis for ESPA based on the mutual information between target structures and annotation processes. We show that for structures like chains, bipartite graphs, and BIO chunks, the information brought by an extra annotation attenuates as the annotation of the structure is more complete, suggesting to stop early and move to a new structure (although it still remains unclear when it is optimal to stop). This analysis is further supported by experiments on temporal relation extraction, semantic role classification, and shallow parsing, three tasks analogous to the three structures analyzed earlier, respectively. The ratio of the attenuation curve as in Fig. 4 is also shown to be an actionable metric to quantify the strength of a type of structure, which can be useful in various analysis, including judging whether ESPA is worthwhile for a particular task. For example, a more detailed I_k-based analysis for SRC shows that predicates with more arguments are stronger structures than those with fewer arguments; we have investigated ESPA on those with more than 6 arguments and indeed, observed much larger improvement in SRC. More details on this analysis are put in the appendix.

We think that the findings in this paper are very important. First, as far as we know, we are the first to propose the mutual information analysis that provides a unique view of structured annotation, that of the reduction in the uncertainty of a target of interest $\mathbf{Y}$ by another random variable/process. From this perspective, signals that have non-zero mutual information with $\mathbf{Y}$ can be viewed as "annotations". These can be partially labeled structures (studied here), partial labels (restricting the possible labels rather than determining a single one as in e.g., Hu et al. (2019), noisy labels (e.g., generated by crowdsourcing or heuristic rules) or, generally, other indirect supervision signals that are correlated with $\mathbf{Y}$. As we proposed, these can be studied within our mutual information framework as well. This paper thus provides a way to analyze the benefit of general *incidental supervision signals* (Roth, 2017)) and possibly even provides guidance in selecting good incidental supervision signals.

Second, the findings here open up opportunities for new annotation schemes for structured learning. In the past, partially annotated training data have been either a compromise when completeness is infeasible (e.g., when ranking entries in gigantic databases), or collected freely without human annotators (e.g., based on heuristic rules). If we intentionally ask human annotators for partial annotations, the annotation tasks can be more flexible and potentially, cost even less. This is because annotating complex structures typically require certain expertise, and smaller tasks are often easier (Fernandes and Brefeld, 2011). It is very likely that some complex annotation tasks require people to read dozens of pages of annotation guidelines, but once decomposed into smaller subtasks, even laymen can handle them. Annotation schemes driven by crowdsourced question-answering, known to provide only partial coverage are successful examples of this idea (He et al., 2015; Michael et al., 2017). Therefore, this paper is hopefully interesting to a broad audience.

Acknowledgements

This research is supported in part by a grant from the Allen Institute for Artificial Intelligence (allenai.org); the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network; Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA); and by the Army Research Laboratory (ARL) and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In *Proc. of the International*

Conference on Language Resources and Evaluation (LREC), volume 1, page 2. Citeseer.

Dana Angluin. 1988. Queries and concept learning. *Machine Learning*, 2(4):319–342.

Les E Atlas, David A Cohn, and Richard E Ladner. 1990. Training connectionist networks with queries and selective sampling. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 566–573.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of the Annual ACM Workshop on Computational Learning Theory (COLT)*.

Graham Brightwell and Peter Winkler. 1991. Counting linear extensions is #p-complete. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, pages 175–181.

Xavier Carreras and Lluis Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 152–164.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–287.

Jonghyun Choi, Jayant Krishnamurthy, Aniruddha Kembhavi, and Ali Farhadi. 2018. Structured set matching networks for one-shot part labeling. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Eraldo R Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Proc. of the Joint European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 643–653.

Dirk Hovy and Eduard Hovy. 2012. Exploiting partial annotations with em training. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 31–38.

Peiyun Hu, Zack Lipton, Anima Anandkumar, and Deva Ramanan. 2019. Active learning with partial feedback. In *Proc. of the International Conference on Learning Representations (ICLR)*.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proc. of the International Conference on Machine Learning (ICML)*.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. Cogcompnlp: Your swiss army knife for nlp. In *11th Language Resources and Evaluation Conference*.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC-2002*.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with amazon mechanical turk. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1546–1556.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proc. of International Conference on Research and Development in Information Retrieval, SIGIR*, pages 3–12.

Xinghua Lou and Fred A Hamprecht. 2012. Structured learning from partial annotations. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 371–378.

Mitchell P Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2017. Crowdsourcing question-answer meaning representations. *arXiv preprint arXiv:1711.05885*.

Graham Neubig and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*. Citeseer.

Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. Joint reasoning for temporal and causal relations. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 2278–2288.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A multi-axis annotation scheme for event temporal relations. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 1318–1328.

Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. 2018c. Exploiting partially annotated data for temporal relation extraction. In *The Joint Conference on Lexical and Computational Semantics (*Proc. of the Joint Conference on Lexical and Computational Sematics)*, pages 148–153.

Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018d. CogCompTime: A tool for understanding time in natural language. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. *Semantic role labeling*, volume 3.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 995–1001.

Vasin Punyakanok, Dan Roth, Wen tau Yih, and Dav Zimak. 2005. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Dan Roth and Kevin Small. 2006a. Active learning with perceptron for structured output. In *Proc. of the International Conference on Machine Learning (ICML)*.

Dan Roth and Kevin Small. 2006b. Margin-based active learning for structured output spaces. In *Proc. of the European Conference on Machine Learning (ECML)*.

Dan Roth and Kevin Small. 2008. Active learning for pipeline models. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. 2018. On the information bottleneck theory of deep learning. In *Proc. of the International Conference on Learning Representations (ICLR)*.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Ohad Shamir, Sivan Sabato, and Naftali Tishby. 2010. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the CoNLL-2000 and LLL-2000*, pages 127–132.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 897–904.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating time expressions, events, and temporal relations. **SEM*, 2:1–9.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervied methods. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Shujian Yu and Jose C Principe. 2018. Understanding autoencoders with information theoretic concepts. *arXiv preprint arXiv:1804.00057*.

Sequential Attention with Keyword Mask Model for Community-based Question Answering

Jianxin Yang[1,2], **Wenge Rong**[1,2], **Libin Shi**[3], **Zhang Xiong**[1,2]
[1]State Key Laboratory of Software Development Environment, Beihang University, China
[2]School of Computer Science and Engineering, Beihang University, China
[3]Microsoft, China
`{yjx17,w.rong,xiongz}@buaa.edu.cn`, `libshi@microsoft.com`

Abstract

In Community-based Question Answering system(CQA), Answer Selection(AS) is a critical task, which focuses on finding a suitable answer within a list of candidate answers. For neural network models, the key issue is how to model the representations of QA text pairs and calculate the interactions between them. We propose a Sequential Attention with Keyword Mask model(SAKM) for CQA to imitate human reading behavior. Question and answer text regard each other as context within keyword-mask attention when encoding the representations, and repeat multiple times(hops) in a sequential style. So the QA pairs capture features and information from both question text and answer text, interacting and improving vector representations iteratively through hops. The flexibility of the model allows to extract meaningful keywords from the sentences and enhance diverse mutual information. We perform on answer selection tasks and multi-level answer ranking tasks. Experiment results demonstrate the superiority of our proposed model on community-based QA datasets.

1 Introduction

Answering selection(AS) is one of the most fundamental challenges in community-based question answering(CQA) services. Given a question and a list of candidate answers, its aim is to choose the most matching one to the question. During this process of matching questions and answer candidates, how to encode the question and answer(QA) into meaningful and semantic representations impacts on the results directly.

Earlier conventional statistic methods are normally based on feature engineering and resource toolkits. Though these methods are easy in implementation, they require extra efforts and handcrafted features(Heilman and Smith, 2010; Tymoshenko and Moschitti, 2015). Recently, with the development of neural network, deep learning based models attract much attention in various tasks(Krizhevsky et al., 2012; Sutskever et al., 2014). In question answering field, the convolutional neural networks(CNNs)(Yu et al., 2014; Hu et al., 2014; Severyn and Moschitti, 2015) and recurrent neural networks(RNNs)(Wang and Nyberg, 2015; Feng et al., 2015) are widely employed to convert the question and answer text into vectors and define a feed-forward multi-layer perceptron to compute the interactions between them. These models construct sentences in an end-to-end fashion with less manual involvement. To capture fine-grained features, on the one hand, some works are concerned with matching QA pairs relationship in a more complex and diverse way, e.g., CNTN(Qiu and Huang, 2015) and MV-LSTM(Wan et al., 2016). On the other hand, latent representation models aim to jointly learn lexical and semantic information from QA sentences and influence the vector generation directly, e.g., attention mechanism(Bahdanau et al., 2015).

Attention mechanism learns attention weights of each words pairs between QA sentences. Afterwards it can calculate the weighted sum of hidden states over all time steps(dos Santos et al., 2016). This approach has shown promising results, while challenges still exist. For example, questions and answers in CQA services are generally long sentences, as such it is still difficult to compress all information into a fixed-length vector. To solve this problem, Sha et al. (2018) proposes co-attention view which brings improvement, and Zhang et al. (2018) further proposes a two-step attention to build dynamic question vectors based on various answer words. These kinds of methods usually require more parameters to learn representations. More importantly, when computing attention weights, every words

Proceedings of NAACL-HLT 2019, pages 2201–2211
Minneapolis, Minnesota, June 2 - June 7, 2019.

in QA pairs are involved. This word-to-word pattern takes meaningless noise into consideration, such as informal language usage or text irrelevant to the question. To alleviate this problem, Chen et al. (2018) proposes a context-aligned model to align phrase in QA relying on overlapped words and Stanford Core NLP tools[1]. Inspired by co-attention, we extend it to sequential style to learn better representations and try to extract useful keywords with less parameters and resource toolkits.

In this work, we propose a Sequential Attention with Keyword Mask(SAKM) model for answer selection task. We encode sentences similar to human reading behavior. When generating the question, our model refers to the answer and combines the mutual information. It is the same processing for producing answer representations. So when encoding a question, answer text is used as context and vice versa. We term this co-attention view as one “hop”. Afterwards we repeat this process several times(hops) in a sequential style. As such QA pairs review each other recurrently to remind of mutual information and refine the sentence representations to be more precise across hops. Besides, the Keyword Mask modifies the attention mechanism such that the attention is computed over keywords instead of all words in the QA pair. So only keywords in the long context are extracted at each time step.

The contributions in this paper are three folders: 1) We extend attention mechanism to sequential structure, so the question and answer review each other recurrently to improve the sentence representations. 2) Different from standard soft attention, we propose sequential attention with keyword mask(SAKM) model. Besides, our model focuses on the significant words and filters other meaningless data. 3) We analyse the proposed SAKM model not only on classical answer selection tasks and but also multi-level answer ranking tasks. Experiment results show that our model tends to encode more rich semantic representations with less parameters.

2 Related Work

In Community-based Question Answering(CQA) services, since normally there exists a large number of question and answer pairs in repository, answer selection(AS) is a critical task, which focuses on finding a suitable answer within a list of candidate answers. As for traditional methods, feature engineering is a core work, but also a time-consuming and laborious task. BM25(Robertson et al., 1994) calculates relevance with item frequency, while language models(Ponte and Croft, 2017; Zhai and Lafferty, 2004) use the maximum likelihood of a word estimated from the question. Translation-based language models(Jeon et al., 2005; Xue et al., 2008) further improve. Considering the syntactic structure, some prior works(Heilman and Smith, 2010; Tymoshenko and Moschitti, 2015) convert the sentence into a tree structure by dependency parsing. Additionally, linguistics resources such as WordNet are utilized to enhance lexical features. Classification models like chain Conditional Random Fields have been used to match the questions and answers(Kiritchenko et al., 2014).

Recently, neural network based models have shown effectiveness in various fields, such as computer vision(Krizhevsky et al., 2012) and natural language processing(Kim, 2014). Different from aforementioned approaches, deep neural architectures(Hu et al., 2014; Wang and Nyberg, 2015) map each word into an embedding space, and compress the whole sentence into a low dimension vector. Then a similarity function is defined to calculate the interactions between QA pairs. Closer vectors in the embedding space represent much more relevant text.

To model fine-grained features, Qiu and Huang (2015) combines CNN with neural tensor network(NTN) to learn complex interactions between QA pairs. But NTN increases a lot of parameters and costs more runtime and memory. MV-LSTM proposed by Wan et al. (2016) uses bi-direction LSTM to generate a positional representation at each time step. Subsequently these representations from questions and answers are fed into a tensor layer. Shen et al. (2017) learns word representations in an embedding space by a translation matrix, and calculates relevance of each word pair in QA to compute a similarity matrix. Then CNN maps this matrix to a score scalar. Recently, (Tay et al., 2018b) applies the hyperbolic distance function to model the relationship between QA.

Other latent representation models construct interactions between QA when encoding sentences. More mutual information is extracted to learn a better latent representation. Miao et al. (2016) pro-

---

[1] `https://stanfordnlp.github.io/CoreNLP/`

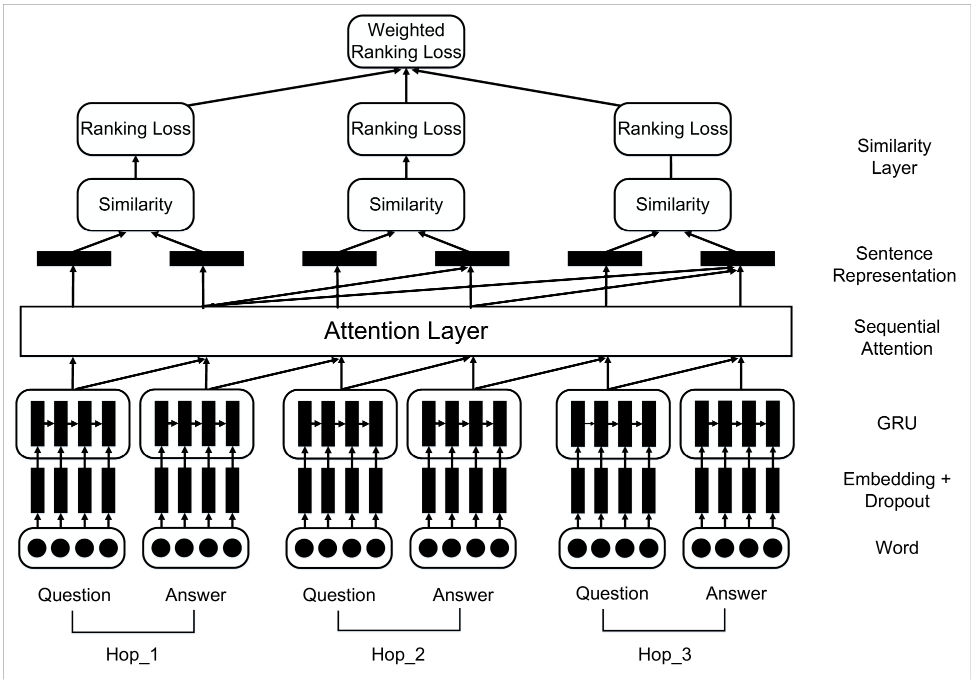


Figure 1: The architecture of 3 hops SAKM model. For simplification, we omit the lines in Sentence Representation Layer of Question part.

poses neural variational inference network. Wang et al. (2016) takes question context into consideration in RNN cell of answer network with gate mechanism. Yin et al. (2016) and dos Santos et al. (2016) propose attention-based CNN models to add attention weight matrix between a QA pair as a feature map. Additionally, Zhang et al. (2017) extends attention weight matrix to 3D tensor including more diverse information. Sha et al. (2018) proves co-attention view can significantly outperform the single attention. Further more, Zhang et al. (2018) constructs two-step attention to obtain question aware vectors based on various words in an answer sentence.

3 Sequential Attention with Keyword Mask Model

Given a question, which may contain one or more clauses, it can be denoted as $Q = (q_1, q_2, ..., q_n)$. Similarly, an answer can be denoted as $A = (a_1, a_2, ..., a_n)$. A^+ and A^- represent a positive answer and a negative answer, respectively. Fig. 1 describes the overall architecture of the proposed Sequential Attention with Keyword Mask(SAKM) model(in this figure, we use three hops as illustration). We extend our network in a sequential style. For each hop, a serial of stacked layers are constructed for the questions and the answers.

3.1 Embedding and Dropout

Firstly the questions and the answers need to be fed into the embedding layer and each word in sentences corresponds to an one-hot vector. Given a look-up table, each word is converted into an embedding space. The index of the low dimensional vector in the look-up table is the same as one-hot vector. We denote the embedding vectors of the QA pairs as $Q_{emb} = (x_1, x_2, ..., x_n) \in R^{d\times|Q|}$ and $A_{emb} = (y_1, y_2, ..., y_n) \in R^{d\times|A|}$, where d is the embedding size, $|Q|$ and $|A|$ denote the length of the question and answer respectively.

In order to mitigate the risk of overfitting, we employ dropout layer to randomly ignore different part of neurons in different hops during training. This process learns better representations of local regions and leads to better generalization during testing.

3.2 Sequential Attention

Gated Recurrent Unit(GRU) To encode a sentence into a single vector, we choose gated recurrent unit(Cho et al., 2014) and construct Q-GRU and A-GRU for the questions and answers, respectively. Given an input sentence $S = (s_1, s_2, ..., s_n) \in R^{d\times|S|}$, GRU handles each word recurrently and at time step t the operation is defined as follow:

$$\begin{aligned} r_t &= \sigma(W_r s_t + U_r h_{t-1} + b_r) \\ z_t &= \sigma(W_z s_t + U_z h_{t-1} + b_z) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \end{aligned} \quad (1)$$

where

$$\tilde{h}_t = \tanh\left(W_h s_t + U_h(r_t \odot h_{t-1}) + b_h\right) \quad (2)$$

In the above equations, W_r, W_z, $W_h \in R^{m \times d}$ and U_r, U_z, $U_h \in R^{m \times m}$ are parameters in the neurons. m is the dimension size of the hidden states. b_r, b_z and $b_h \in R^m$ are bias. σ is sigmoid function and $\odot$ means element-wise product.

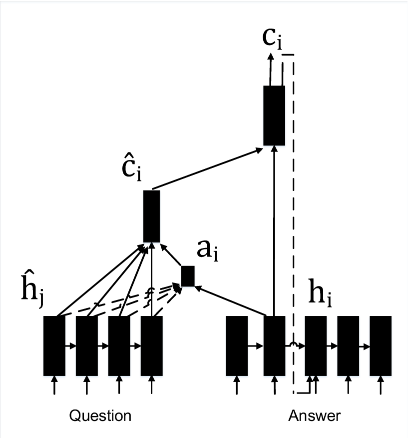


Figure 2: soft attention details.

Attention Mechanism During the GRU encoding process, an attention mechanism helps to combine context information with the current hidden states. For the standard soft attention mechanism(Bahdanau et al., 2015), as demonstrated in Fig. 2, the hidden state at time t computes attention weights with all of the context hidden states, and then obtains alignment scores after softmax operation. This mechanism takes all of the words in the context into consideration, while our model expects to extract some keywords to the current word and ignore other meaningless or noisy segments. The keyword mask relies on the attention weights and reserves top percent of words to account for alignment scores. It can be formulated as:

$$\begin{aligned} e_{ij} &= v^T \tanh(W_a[h_i; \hat{h}_j]) \\ e_{ij}^{mask} &= f_{mask\_top\_k}(e_{ij}, -inf) \\ a_{ij} &= \frac{\exp(e_{ij}^{mask})}{\sum_{k=1}^{|S|} \exp(e_{ik}^{mask})} \\ \hat{c}_i &= \sum_{j=1}^{|S|} a_{ij} \hat{h}_j \\ c_i &= \tanh(W_c[\hat{c}_i; h_i]) \end{aligned} \quad (3)$$

where $[;]$ is the operator of concatenation, and $f_{mask\_top\_k}$ denotes the function that the top percent of values are reserved while others are masked as value $-inf$. So these masked positions in a_{ij} become 0 after softmax operation, which represents no influence to the current hidden state h_i. Fig. 3 shows the details of this attention mechanism. We will discuss the keywords percentage in more detail in Section 5.2.

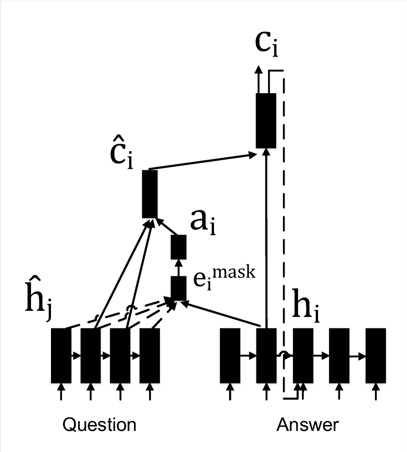


Figure 3: keyword-mask attention details.

After obtaining the representation of hidden neuron at time step t, we concat c_i with next word as input corresponding to the dotted line described in Fig. 3. When the whole sentence is processed, the final hidden output does not become representation directly because it loses much information about the beginning of the sentence. Instead, average operation over all hidden outputs is taken to produce the final representation.

Sequential Extension As shown in Fig. 1, A-GRU regards the question sentences as context to compute attention and representations. Likewise, Q-GRU reviews the answer contents to tune the representations. We extend this process in a sequential style to capture features and enhance information both from question text and answer text.

All of the parameters across hops are shared. For each hop, the vectors of QA pairs interact and improve. Compared to single direction attention or single hop attention, our model gets much more flexibility. It is capable of updating the representations towards the correct direction with the guide of a loss function and gradients across hops. We rename the Eq. 3 as $MaskAttention()$, as such the equations of the sequential extension is defined as:

$$\begin{aligned} Q^h &= MaskAttention(Q^h_{emb}, A^{h-1}) \\ A^h &= MaskAttention(A^h_{emb}, Q^h) \end{aligned} \quad (4)$$

where h is the current number of hop. The model iteratively updates the joint representations of the question and answer pair and obtains different outputs across hops.

Sentence Representation In the sentence representation layer, $Q^h_{representation}$ and $A^h_{representation}$ denote the final representation outputs of QA pairs at the hop h. Since Q^h and A^h only contain the information extracted at hop h, more meaningful content would be lost after more hops. But we expect to remember it from the beginning hops. To convey more information across hops, we do not simply take Q^h and A^h as the sentence representations. Instead, all of the previous outputs from $MaskAttention$ are involved. They can be calculated as:

$$\begin{aligned} Q^h_{representation} &= \frac{1}{h}\sum_{j=1}^{h} Q^h \\ A^h_{representation} &= \frac{1}{h}\sum_{j=1}^{h} A^h \end{aligned} \quad (5)$$

3.3 Similarity Calculation

Finally, we design a weighted loss strategy to compute the relevance and the loss value between QA pairs. For each hop, we have a pair of QA sentence representations and pass them through a similarity function described as:

$$s^h(Q,A) = \frac{{Q^h_{representation}}^T A^h_{representation}}{||Q^h_{representation}||_2 \cdot ||A^h_{representation}||_2} \quad (6)$$

where $s^h(Q,A)$ is the matching score between QA pairs at hop h. $||\cdot||_2$ means euclidean distance. As for the loss function during training, given a question, we use pair-wise margin-based ranking loss for a triple (Q, A^+, A^-). Thus the mathematical expression is:

$$\begin{aligned} L^h(Q, A^+, A^-) = \max(0, m - (s^h(Q, A^+) \\ -s^h(Q, A^-))) \end{aligned} \quad (7)$$

where m is the predefined margin.

Since we expect that vector representations generated from posterior hops are more precisely than the ones produced from prior hops, relatively small tolerance to the risk of matching incorrect QA pairs is accepted for posterior hops. Therefore, the loss values take increasing weights across hops. We denote r_h as loss weights. The objective loss function can be defined as:

$$L(Q, A^+, A^-) = \sum_{h=1}^{H} r_h L^h(Q, A^+, A^-) \quad (8)$$

4 Experimental Study

In this section, we test the proposed model on classical answer selection task and also multi-level answer ranking task to validate the model's effectiveness[2].

4.1 Answer Selection

Dataset & Implementation Details In this task, we use a community-based question answering dataset YahooCQA provided by Tay et al. (2017). It is an open-domain community forum, and the dataset contains 142,627 QA pairs. Sentences in YahooCQA are generally long and noisy. We follow the preprocessing in their work without extra process. Four negative answers are generated for a question using *Lucene*. Table 1 demonstrates the statistics of YahooCQA.

Dataset	YahooCQA	ZhihuCQA
# of Qns	50.1k / 6.2k / 6.2k	16k / 2k / 2k
# of Pairs	253k / 31.7k / 31.7k	80k / 10k / 10k

Table 1: Statistics of Datasets.

For our model, we tune the hidden size to 300, and the numbers of GRU layers for modeling questions and answers are both 1. Dropout is 0.5 and word embedding is pre-trained by skip-gram model. For the Sequential Extension layer, the number of hops is 3. For the Similarity Calculation, margin is 0.1 and weights for all hops are

---

[2] Our code is available at `https://github.com/sheep-for/question_answer_matching`

set as (0.2, 0.3, 0.5). Weights are set to constants because we promise to put more weight on later hops and keep reasonable tolerance for prior ones. Batch size is 20. All of the parameters are optimized by Back Propagation and Momentum.

Baselines We compare our model against several advanced deep neural network models. CNTN(Qiu and Huang, 2015), NTN-LSTM, HD-LSTM(Tay et al., 2017) and HyperQA(Tay et al., 2018b) are interaction focused methods, while AP-CNN, AP-BiLSTM(dos Santos et al., 2016), QRNN(Bradbury et al., 2017), CTRN(Tay et al., 2018a) and two-step attention(Zhang et al., 2018) are latent representation models. Additionally, we choose two traditional methods Random Guess and BM25(Robertson et al., 1994).

Evaluation Metrics For YahooCQA, we use Precision@1(P@1) and Mean Reciprocal Rank(MRR) to evaluate our model and the metrics are defined as:

$$\begin{aligned} P@1 &= \frac{1}{N}\sum_{i=1}^{N}\delta(r(A^{+})=1) \\ MRR &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r(q_i)} \end{aligned} \tag{9}$$

where δ is indicator function, N is the number of all queries and $r(q_i)$ is the rank of the first correct answer to question q_i.

Model	**P@1**	MRR
Random Guess	20.0%	45.7%
BM25	22.5%	49.3%
CNN	41.3%	63.2%
LSTM	46.5%	66.9%
CNTN	46.5%	63.2%
NTN-LSTM	54.5%	73.1%
AP-CNN	56.0%	72.6%
AP-BiLSTM	56.8%	73.1%
QRNN	57.3%	73.6%
HD-LSTM	55.7%	73.5%
CTRN	60.1%	75.5%
Two-step attention	62.2%	77.4%
HyperQA	68.3%	80.1%
SA	66.0%	79.2%
SAKM	**69.3%**	**81.4%**

Table 2: Experiment results on YahooCQA. SA is the sequential attention baseline without keyword-mask operation. SAKM is our sequential attention with keyword mask model.

Experiment Results The results are shown in Table 2. Firstly, it is observed that deep neural network models outperform traditional models. Most latent representation models obtain better results than interaction focused models, indicating that earlier interactions when encoding sentences produces semantic vectors. Most importantly, the proposed SAKM model achieves best results on both P@1 and MRR. Our basic SA model outperforms two-step attention model by 3.8% in terms of P@1 and 1.8% in terms of MRR, which shows that our sequential extension structure is effective. Furthermore, our SAKM model outperforms HyperQA model by 1.0% in terms of P@1 and 1.3% in terms of MRR. Since HyperQA is an interaction focused model which adopts the hyperbolic distance function to model the relevance between QA. we can combine it with our SAKM to obtain better performance in further study. The experiment results agree with our intuition that extracting meaningful keywords in attention mechanism helps to generate more precise representations.

4.2 Multi-Level Answer Ranking

Relevant relationship in answer selection datasets is binary, only including relevance and irrelevance. However, in the real CQA applications, it is difficult to verify whether the answers are completely correct or not. This scenario has caused a challenge called multi-level answer ranking(Liu et al., 2018). These answers for one question are annotated as several levels corresponding to the thumb-up numbers.

Dataset & Implementation Details To test the proposed model in multi-level answer ranking task, we choose the dataset ZhihuCQA provided by Liu et al. (2018). Zhihu[3] is a popular and professional Chinese QA community platform with more than millions of users and QA pairs. Table 1 describes the statistics of ZhihuCQA. For each question, top five answers are selected and ranked by the thumb-up numbers. We replace margin based ranking loss with RankNet(Burges et al., 2005).

In this task, we use the jieba[4] toolkits for word segmentation and tune the hidden size to 200, and the numbers of GRU layers for modeling questions and answers are both 2. Other settings are the same as YahooCQA.

---

[3] `https://www.zhihu.com/`

[4] `https://github.com/fxsjy/jieba`

Baselines We compare our model against available advanced methods in (Liu et al., 2018). ARC-II learns hierarchical pattern based on ARC-I(Hu et al., 2014). Skip-Thoughts model(Kiros et al., 2015) trains an encoder-decoder model to construct sentence vectors. Attentive LSTM, ABCNN and Compare-Aggregate(Wang and Jiang, 2017) are attention-based models and Rewrite+Rank is based on generative adversarial network.

Evaluation Metrics In this task, since the labels for an answer are not binary, we choose normalized discounted cumulative gain(NDCG) and expected reciprocal rank(ERR) for evaluation. $(o_1, o_2, ..., o_M)$ denotes the predicted orders of answers to a question. NDCG is defined as:

$$\begin{aligned} NDCG &= \frac{DCG}{iDCG} \\ DCG &= \sum_{i=1}^{M} \frac{2^{o_i} - 1}{\log(1+i)} \end{aligned} \tag{10}$$

where $iDCG$ is the ideal DCG calculated from the correct orders $(l_1, l_2, ..., l_M)$. ERR is defined as:

$$\begin{aligned} ERR &= \sum_{r=1}^{M} \frac{R_r}{r} \prod_{i=1}^{r-1} (1 - R_i) \\ R_i &= \frac{2^{o_i} - 1}{2^{o_m}} \end{aligned} \tag{11}$$

where o_m is the maximum of degree values.

Model	**NDCG**	ERR
Random Guess	41.7%	30.3%
ARC-I	64.4%	56.4%
ARC-II	68.5%	58.8%
Skip-Thoughts	69.1%	60.1%
Attentive LSTM	71.4%	61.1%
ABCNN	72.5%	62.0%
Compare-Aggregate	74.8%	63.2%
Rewrite+Rank	77.2%	65.0%
SA	80.7%	68.2%
SAKM	**81.0%**	**68.4%**

Table 3: Experiment results on ZhihuCQA.

Experiment Results Table 3 reports the results on ZhihuCQA. Similar to YahooCQA, attention-based models perform better than other baselines. Our SA model outperforms Rewrite+Rank by 3.6% in terms of NDCG and 3.2% in terms of ERR. SAKM model achieves slightly improvement compared to SA version. Results show that on multi-level answer ranking task, our review mechanism allows question and answer interaction while encoding in a more fine-grain aspect and leads to better performance.

5 Discussion and Analysis

In this section, we divide our discussion into three parts, including the trade-off between information transmission and avoidance of overfitting, the relationship between sentence length and keyword percentage, the advantages of our SAKM model.

5.1 Trade-off between Information Transmission and Avoidance of Overfitting

Our model processes QA text in a sequential style. For the first hop, the original contents are fed as inputs. Afterwards, the representations are updated based on previous outputs across hops, thus it is significant to convey rich mutual information. Meanwhile, since the sentences are long and redundant, it is necessary to avoid overfitting.

Information Transmission across Hops In order to utilize context better, We propose sequential style to refine the sentence vectors through multiple hops. When calculating the sentence representations, we take the average over outputs of all time steps instead of selecting the final hidden state, and get the final sentence representations based on all hops. Additionally, in embedding layer we pretrain the word embeddings on the corpus using word2vec(Mikolov et al., 2013). To guide the vectors update in a correct direction based on gradients, the loss function is calculated over all hops, and puts more weight on later ones.

Avoidance of Overfitting There are some tricks to reduce the risk of overfitting. Our model extracts some keywords according to the attention weights, and applies dropout to ignore different neurons in different hops. Besides, the GRU layer is shallow and the number of hops is suitable.

5.2 Relationship between Sentence Length and Keyword Percentage

In the attention mechanism, we use $f_{mask\_top\_k}$ to reserve top percent of attention weights. In this part, we propose two strategies to explore the relationship between the sentence length and keyword percentage.

Fixed-Percentage: We set the number of keywords based on the statistics of the dataset. Firstly, we count the lengths of all questions and answers respectively, and then sort them in an ascending order. We choose the value of the third quartile as the number of keywords in a question, while the value of the first quartile as the number of keywords in an answer. This strategy allows us not to calculate percentage according to various lengths in the dataset. Our experiments empirically show that it works well.

Variable-Percentage: The second strategy is to compute the number of keywords for all sentences.

Question: Since the answers are produced based on question sequences, the question generally contains more meaningful information such as inquiry type, inquiry main verb, topic and so on. We empirically calculate the number of keywords k in a question of length x as follow:

$$k = \min(10 \ln x, x) \tag{12}$$

Answer: As for answers submitted by users in community forum, there exists redundant contents, typos errors, emoticon and other informal language usage. Inspired by TF-IDF algorithm, in our experiments, we propose a heuristics rule to calculate the number of keywords k. The length of an answer is denoted as x. For the first part, we compute the Total-Length(TL) term as follow:

$$TL(x) = x\lfloor \lg x \rfloor \tag{13}$$

where $\lfloor \cdot \rfloor$ is rounding down operation. TL value increases monotonously with length of sentence. For the second part, we compute the Inverse-Noise-Frequency(INF) term as follow:

$$INF(x) = \frac{1}{\lg 2x} \tag{14}$$

This term represents the percent of the meaningful words, which is an inverse proportion to noisy words. Finally, the number of keywords k can be obtained by multiplying these two terms.

$$k = \min(TL(x) \cdot INF(x), x) \tag{15}$$

5.3 Advantages of SAKM model

Simplicity: Our SAKM model is simple but outperforms on large CQA datasets. The network is shallow and all of the parameters across hops are shared. Our model is not complicated and has less parameters than other mentioned neural network models. Table 4 demonstrates the complexity analysis of some models.

Our model is an end-to-end neural network, and trained via back-propagation automatically. The SAKM model could be an universal way to learn sentence vectors effectively and integrated in other larger neural network models. It could be a useful tool in building neural architecture based representations for text sequences.

Model	**Complexity**	**Parameters**
AP-BiLSTM	$4(md + d^2) + 4d^2$	1.08M
NTN	$2dk + 2k + d^2k$	2.1M
CTRN	$3kdm + 2dh + h$	1.05M
Two-step	$6md + 10d^2 + 12d^2k$	1.31M
SAKM (Our)	$3(md + d^2) + 4d^2$	0.9M

Table 4: Complexity analysis on YahooCQA. m is embedding size, k is the filter width, d is the output dimension and h is the size of full-connected layer.

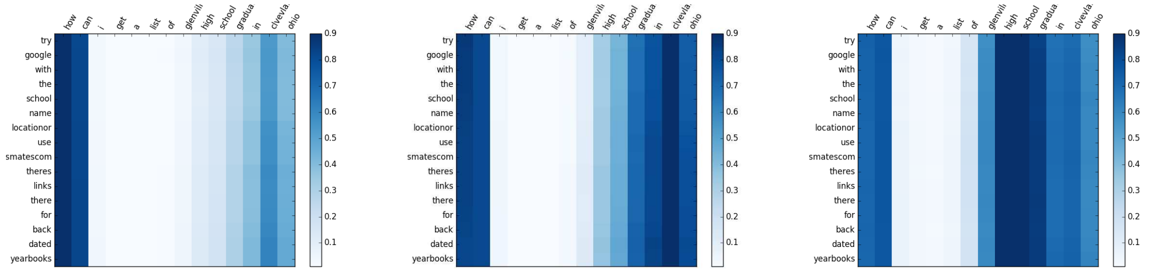
Convergency: As the sentence representations are tuned and improved across hops in one epoch, it costs less epochs for our model to converge. In our experiments, performance improve quickly in first ten epochs.

Effectiveness: The QA pairs capture features and information both from question text and answer text, iteratively updating and improving question and answer representations through hops.

At the test time, choosing the outputs of the last hop from sentence representation layer as sentence vectors can obtain better results on P@1 and MRR, demonstrating the effectiveness of improvement across hops. SAKM model outperforms SA model by more than 3% on P@1 and 2% on MRR. It proves that extracting keywords is significant and necessary. Besides, even if we choose the first hop of SA model, the gains are significant compared to two-way attention model. It means that our refinement procedure leads to better representations for all hops. Table 5 shows the details.

Representation	**P@1**	**MRR**
SA_hop_1	64.5%	78.2%
SA_hop_last	66.0%	79.2%
SA_concat_hops	65.7%	79.0%
SAKM_hop_1	68.7%	81.0%
SAKM_hop_last	**69.3%**	**81.4%**
SAKM_concat_hops	69.2%	81.3%

Table 5: Experiment results on YahooCQA using various representations. SAKM_hop_1 denotes that SAKM model tests with the sentence representations of the first hop.



(a) Visualization of SA at first hop (b) Visualization of SA at last hop (c) Visualization of SAKM at last hop

Figure 4: Attention Visualization.

Given a pair of QA as example. *Q: How can i get a list of glenville high school graduates in clvevland ohio. A: Try google with the school name locationor use classmatescom theres links there for back dated yearbooks.* Fig. 4 displays the heatmap of the attention weights. We can observe that compared to the first hop of SA model, the last hop puts more weights on phrase *glenville high school graduates in clvevland ohio*. Further more, the last hop of SAKM model focuses on keywords *how can, glenville high school graduates.* It shows that the inquiry type words and topic words achieve more attention. From this heatmap, it indicates that our SAKM model is reasonable and works well.

6 Conclusion

In this work, we propose a sequential attention with keyword mask model for CQA. Our model handles answer selection task similar to human reading behavior. The questions and answers review each other recurrently to improve the representations. This proposed attention mechanism focuses on some keywords and filters other meaningless data. We evaluate our model on two tasks, answering selection and multi-level answer ranking. The experiment results demonstrate that our model outperforms on CQA datasets and enhance mutual information between QA pairs effectively.

Acknowledgments

This work was supported in part by the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2019ZX-16).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of 2015 International Conference on Learning Representations.*

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. *Proceedings of 2017 International Conference on Learning Representations.*

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of 22nd International Conference on Machine Learning*, pages 89–96.

Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, and Liang He. 2018. CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 265–273.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Proceedings of 2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 813–820.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of 2010 Conference of the North American Chapter of the Association of Computational Linguistics*, pages 1011–1019.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of 2014 Annual Conference on Neural Information Processing Systems*, pages 2042–2050.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management*, pages 84–90.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif M Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 437–442.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of 2015 Annual Conference on Neural Information Processing Systems*, pages 3294–3302.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of 2012 Annual Conference on Neural Information Processing Systems*, pages 1097–1105.

Yang Liu, Wenge Rong, and Zhang Xiong. 2018. Improved text matching by enhancing mutual information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of 2016 International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Wentau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of 2013 Conference of the North American Chapter of the Association of Computational Linguistics*, pages 746–751.

Jay M. Ponte and W. Bruce Croft. 2017. A language modeling approach to information retrieval. *SIGIR Forum*, 51(2):202–208.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1305–1311.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline Hancockbeaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference*, pages 109–126.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382.

Lei Sha, Jinge Yao, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. A multi-view fusion neural network for answer selection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5422–5429.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. 2017. Word embedding based correlation model for question/answer matching. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3511–3517.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of 2014 Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018a. Cross temporal recurrent networks for ranking question answer pairs. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5512–5519.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 583–591.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1451–1460.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2835–2841.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 707–712.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of 2017 International Conference on Learning Representations*.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Chengxiang Zhai and John D Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Pengqing Zhang, Yuexian Hou, Zhan Su, and Yi Su. 2018. Two-step multi-factor attention neural network for answer selection. In *Proceedings of 15th Pacific Rim International Conference on Artificial Intelligence*, pages 658–670.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive interactive neural networks for answer selection in community question answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3525–3531.

Simple Attention-Based Representation Learning for Ranking Short Social Media Posts

Peng Shi,[1] **Jinfeng Rao**,[2*] and **Jimmy Lin**[1]
[1] David R. Cheriton School of Computer Science, University of Waterloo
[2] Facebook AI
`{peng.shi, jimmylin}@uwaterloo.ca`, `raojinfeng@fb.com`

Abstract

This paper explores the problem of ranking short social media posts with respect to user queries using neural networks. Instead of starting with a complex architecture, we proceed from the bottom up and examine the effectiveness of a simple, word-level Siamese architecture augmented with attention-based mechanisms for capturing semantic "soft" matches between query and post tokens. Extensive experiments on datasets from the TREC Microblog Tracks show that our simple models not only achieve better effectiveness than existing approaches that are far more complex or exploit a more diverse set of relevance signals, but are also much faster. Implementations of our **samCNN** (**S**imple **A**ttention-based **M**atching CNN) models are shared with the community to support future work.[1]

1 Introduction

Despite a large body of work on neural ranking models for "traditional" *ad hoc* retrieval over web pages and newswire documents (Huang et al., 2013; Shen et al., 2014; Guo et al., 2016; Xiong et al., 2017; Mitra et al., 2017; Pang et al., 2017; Dai et al., 2018; McDonald et al., 2018), there has been surprisingly little work (Rao et al., 2017) on applying neural networks to searching short social media posts such as tweets on Twitter. Rao et al. (2019) identified short document length, informality of language, and heterogeneous relevance signals as main challenges in relevance modeling, and proposed the first neural model specifically designed to handle these characteristics. Evaluation on a number of datasets from the TREC Microblog Tracks demonstrates state-of-the-art effectiveness as well as the necessity of different model components to capture a multitude of relevance signals.

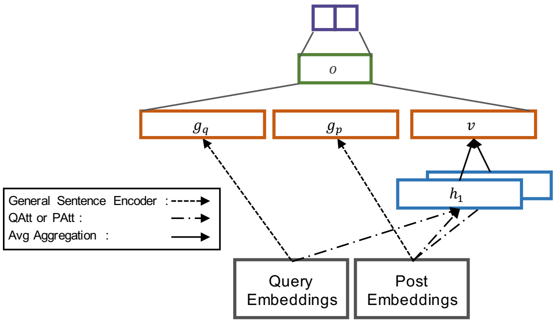


Figure 1: Our model architecture: a general sentence encoder is applied on query and post embeddings to generate $\mathbf{g}_q$ and $\mathbf{g}_p$; an attention encoder is applied on post embeddings to generate variable-length query-aware features $\mathbf{h}_i$. These features are further aggregated to yield $\mathbf{v}$, which feeds into the final prediction.

In this paper, we also examine the problem of modeling relevance for ranking short social media posts, but from a complementary perspective. As Weissenborn et al. (2017) notes, most systems are built in a *top-down* process: authors propose a complex architecture and then validate design decisions with ablation experiments. However, such experiments often lack comparisons to strong baselines, which raises the question as to whether model complexity is empirically justified. As an alternative, they advocate a *bottom-up* approach where architectural complexity is gradually increased. We adopt exactly such an approach, focused exclusively on word-level modeling. As shown in Figure 1, we examine variants of a simple, generic architecture that has emerged as "best practices" in the NLP community for tackling modeling problems on two input sequences: a Siamese CNN architecture for learning representations over both inputs (a query and a social media post in our case), followed by fully-connected layers that produce a final relevance prediction (Severyn and Moschitti, 2015; He et al., 2016; Rao

---

[*] Work done at the University of Maryland, College Park.

[1] `https://github.com/Impavidity/samCNN`

Proceedings of NAACL-HLT 2019, pages 2212–2217
Minneapolis, Minnesota, June 2 - June 7, 2019.

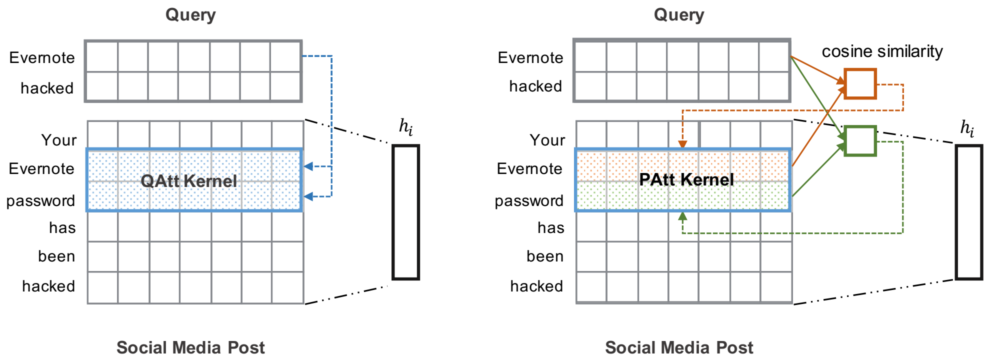


Figure 2: The Query-Aware Attention (QAtt) architecture on the left and the Position-Aware Attention (PAtt) architecture on the right. In both, we construct F convolutional kernels for *each* query token (here, one kernel for the query token ‘Evernote’ is visualized). In QAtt, the query token embedding is directly “injected” into the kernel via element-wise product (blue dotted arrows). In PAtt, cosine similarity between the query token and tokens in the post within the convolution window are used as attention weights in the kernel.

et al., 2016), which we refer to as a General Sentence Encoder in Section 2.1. Further adopting best practices, we incorporate query-aware convolutions with an average aggregation layer in the representation learning process.

Recently, a number of researchers (Conneau et al., 2017; Mohammed et al., 2018) have started to reexamine simple baselines and found them to be highly competitive with the state of the art, especially with proper tuning. For example, the InferSent approach (Conneau et al., 2017) uses a simple BiLSTM with max pooling that achieves quite impressive accuracy on several classification benchmarks. Our contribution is along similar lines, where we explore simple yet highly effective models for ranking social media posts, to gain insights into query–post relevance matching using standard neural architectures. Experiments with TREC Microblog datasets show that our best model not only achieves better effectiveness than existing approaches that leverage more signals, but also demonstrates 4× speedup in model training and inference compared to a recently-proposed neural model.

2 Model

Our model comprises a representation learning layer with convolutional encoders and another simple aggregation layer. These architectural components are described in detail below.

2.1 Representation Learning Layer

General Sentence Encoder: The general sentence encoder uses a standard convolutional layer with randomly initialized kernels to learn semantic representations for text. More formally, given query q and post p as sentence inputs, we first convert them to embedding matrices $\mathbf{Q}$ and $\mathbf{P}$ through an embedding lookup layer, where $\mathbf{Q} \in \mathbb{R}^{n \times d}$ and $\mathbf{P} \in \mathbb{R}^{m \times d}$, d is the dimension of embeddings, and n and m are the number of tokens in q and p, respectively. Then we apply a standard convolution operation with kernel window size k over the embedding matrix $\mathbf{Q}$ and $\mathbf{P}$. The convolution operation is parameterized by a weight term $\mathbf{W} \in \mathbb{R}^{F \times k \times d}$ and a bias term $b_w \in \mathbb{R}^F$, where F is the number of convolutional kernels. This generates semantic representation $\mathbf{O}_q \in \mathbb{R}^{n \times F}$ and $\mathbf{O}_p \in \mathbb{R}^{m \times F}$, on which max pooling and an MLP are applied to obtain query representation $\mathbf{g}_q \in \mathbb{R}^d$ and post representation $\mathbf{g}_p \in \mathbb{R}^d$.

The weakness of the kernels in the general sentence encoder is that they do not incorporate knowledge from the query when attempting to capture feature patterns from the post. Inspired by attention mechanisms (Bahdanau et al., 2014), we propose two novel approaches to incorporate query information when encoding the post representation, which we introduce below.

Query-Aware Attention Encoder (QAtt): In QAtt (Figure 2, left), for each query token, we construct a token-specific convolutional kernel to “inject” the query information. Unlike methods that apply attention mechanisms after the sentence representations are generated (Bahdanau et al., 2014; Seo et al., 2016), our approach aims to model the representation learning process jointly with an attention mechanism.

Formally, for each query token t_q, the QAtt kernel $\mathbf{W}_{\text{QAtt}}^{t_q}$ is composed as follows:

$$\mathbf{W}_{\text{QAtt}}^{t_q} = \mathbf{U} \otimes \mathbf{Q}_{t_q} \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{F \times k \times d}$ represents trainable parameters, $\mathbf{Q}_{t_q}$ is the embedding of token t_q with size $\mathbb{R}^d$ and $\mathbf{W}_{\text{QAtt}}^{t_q} \in \mathbb{R}^{F \times k \times d}$. The element-wise product $\otimes$ is applied between the token embedding $\mathbf{Q}_{t_q}$ and the last dimension of kernel weights $\mathbf{U}$. In other words, we create F convolutional kernels for *each* query token, where each kernel is "injected" with the embedding of that query token via element-wise product. Figure 2 (left) illustrates one kernel for the query token 'Evernote', where element-wise product is represented by blue dotted arrows. When a QAtt token-specific kernel is applied, a window slides across the post embeddings $\mathbf{P}$ and learns soft matches to each query token to generate query-aware representations.

On top of the QAtt kernels, we apply max-pooling and an MLP to produce a set of post representations $\{\mathbf{h}_i\}$, with each $\mathbf{h}_i \in \mathbb{R}^d$ standing for the representation learned from query token t_{q_i}.

Position-Aware Attention Encoder (PAtt): In the QAtt encoder, token-specific kernels learn soft matches to the query. However, they still ignore positional information when encoding the post semantics, which has been shown to be effective for sequence modeling (Gehring et al., 2017). To overcome this limitation, we propose an alternative attention encoder that captures positional information through interactions between query embeddings and post embeddings.

Given a query token t_q and the j-th position in post p, we compute the interaction scores by taking the cosine similarity between the word embeddings of token t_q and post tokens $t_{p_{j:j+k-1}}$ from position j to $j+k-1$:

$$S_j = [\cos(t_q, t_{p_j}); ...; \cos(t_q, t_{p_{j+k-1}})] \tag{2}$$

where $S_j \in \mathbb{R}^{k \times 1}$ and k is the width of the convolutional kernel we are learning. That is, for each token in the post within the window, we compute its cosine similarity with query token t_q. We then convert the similarity vector S_j into a matrix:

$$\hat{S}_j = S_j \cdot \mathbb{1}, \hat{S}_j \in \mathbb{R}^{k \times d} \tag{3}$$

where $\mathbb{1} \in \mathbb{R}^{1 \times d}$ with each element set to 1. Finally, the PAtt convolutional kernel for query token t_q at the j-th position is constructed as:

$$\mathbf{W}_{\text{PAtt}}^{t_q, j} = \mathbf{V} \otimes \hat{S}_j \tag{4}$$

where $\mathbf{V} \in \mathbb{R}^{F \times k \times d}$ represents the trainable parameters. The element-wise product $\otimes$ is applied between the attention weights $\hat{S}_j$ and the last two dimensions of kernel weights $\mathbf{V}$.

Conceptually, this operation can be thought as adding a soft attention weight (with values in the range of $[0, 1]$) to each convolutional kernel, where the weight is determined by the cosine similarity between the token from the post and a particular query token; since cosine similarity is a scalar, we fill in the value in all d dimensions of the kernel, where d is the size of the word embedding. This is illustrated in Figure 2 (right), where we show one kernel of width two for the query token 'Evernote'. The brown (green) arrows capture cosine similarity between the query token 'Evernote' and the first (second) token from the post in the window. These values then serve as weights in the kernels, shown as the hatched areas. Similar to QAtt, the PAtt encoder with max-pooling and an MLP generates a set of post representations $\{\mathbf{h}_i\}$, with each $\mathbf{h}_i$ standing for the representation learned from query token t_{q_i}.

It is worth noting that both the QAtt and PAtt encoders have no extra parameters over a general sentence encoder. However, incorporating the query-aware and position-aware information enables more effective representation learning, as our experiments show later. The QAtt and PAtt encoders can also be used as plug-in modules in any standard convolutional architecture to learn query-biased representations.

2.2 Aggregation Layer

After the representation layer, a set of vectors $\{\mathbf{g}_q, \mathbf{g}_p, \{\mathbf{h}_i\}\}$ is obtained. Because our model yields different numbers of $\mathbf{h}_i$ with queries of different lengths, further aggregation is needed to output a global feature $\mathbf{v}$. We directly average all vectors $\mathbf{v} = \frac{1}{N_q} \sum \mathbf{h_i}$ as the aggregated feature, where N_q is the length of the query.

2.3 Training

To obtain a final relevance score, the feature vectors $\mathbf{g}_q$, $\mathbf{g}_p$, and $\mathbf{v}$ are concatenated and fed into an MLP with ReLU activation for dimensionality reduction to obtain $\mathbf{o}$, followed by batch normalization and fully-connected layer and softmax to output the final prediction. The model is trained end-to-end with a Stochastic Gradient Decent optimizer using negative log-likelihood loss.

Year	2011	2012	2013	2014
# queries	49	60	60	55
# tweets	39,780	49,879	46,192	41,579
# relevant	1,940	4,298	3,405	6,812
%relevant	4.87	8.62	7.37	16.38

Table 1: Statistics of TREC MB 2011–2014 datasets.

Param	Value	Param	Value
Embedding size	300	k	0.05
Hidden size	200	Final hidden size	100
Kernel number	250	Dropout ratio	0.5
Kernel size	2	Learning rate	0.03

Table 2: Hyperparameters for our models. GloVe (Pennington et al., 2014) embeddings are used and fine-tuned during training. Unknown words are initialized from a uniform distribution $[-k, k]$.

3 Experimental Setup

Datasets and Hyperparameters. Our models are evaluated on four tweet test collections from the TREC 2011–2014 Microblog (MB) Tracks (Ounis et al., 2011; Soboroff et al., 2012; Lin and Efron, 2013; Lin et al., 2014). Each dataset contains around 50–60 queries; detailed statistics are shown in Table 1. As with Rao et al. (2019), we evaluated our models in a reranking task, where the inputs are up to the top 1000 tweets retrieved from "bag of words" ranking using query likelihood (QL). We ran four-fold cross-validation split by year (i.e., train on three years' data, test on one year's data) and followed Rao et al. (2019) for sampling validation sets. For metrics, we used average precision (AP) and precision at rank 30 (P30). We conducted Fisher's two-sided, paired randomization tests (Smucker et al., 2007) to assess statistical significance at $p < 0.05$. The best model hyperparameters are shown in Table 2.

Baselines. On top of QL, RM3 (Abdul-Jaleel et al., 2004) provides strong non-neural results using pseudo-relevance feedback. We also compared against MP-HCNN (Rao et al., 2019), the first neural model that captures specific characteristics of social media posts, which improves over many previous neural models, e.g., K-NRM (Xiong et al., 2017) and DUET (Mitra et al., 2017), by a significant margin. To the best of our knowledge, Rao et al. (2019) is the most effective neural model to date. We compared against two variants of MP-HCNN; MP-HCNN+QL includes a linear interpolation with QL scores.

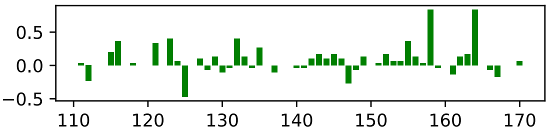


Figure 3: Per-query AP differences between PAtt and QL on TREC 2013 (queries 111–170).

4 Results and Discussion

Table 3 shows the effectiveness of all variants of our model, compared against previous results copied from Rao et al. (2019). Model 1 illustrates the effectiveness of the basic BiCNN model with a kernel window size of two; combining different window sizes (Kim, 2014) doesn't yield any improvements. It appears that this model performs worse than the QL baseline.

Comparing Model 2 to Model 1, we find that query-aware kernels contribute significant improvements, achieving effectiveness comparable to the QL baseline. With Model 3, which captures positional information with the position-aware encoder, we obtain competitive effectiveness compared to Model 8, the full MP-HCNN model that includes interpolation with QL. Note that Model 8 leverages additional signals, including URL information, character-level encodings, and external term features such as tf–idf. With Model 4, which interpolates the position-aware encoder with QL, we obtain state-of-the-art effectiveness.

Per-Query Analysis. In Figure 3, we show per-query AP differences between the PAtt model and the QL baseline on the TREC 2013 dataset. As we can see, PAtt improves on most of the queries. For the best-performing query 164 "*lindsey vonn sidelined*", we project the hidden states **o** into a low-dimensional space using t-SNE (Maaten and Hinton, 2008), shown in Figure 4. We observe that with the basic BiCNN model (left), relevant posts are scattered. With the addition of an attention mechanism (either QAtt in the middle or PAtt on the right), most of the relevant posts are clustered together and separated from the non-relevant posts. With PAtt, there appears to be tighter clustering and better separation of the relevant posts from the non-relevant posts, giving rise to a better ranking. We confirmed similar behavior in many queries, which illustrates the ability of our position-aware attention encoder to learn better query-biased representations compared to the other two models.

		2011		**2012**		**2013**		**2014**	
		P30	**AP**	**P30**	**AP**	**P30**	**AP**	**P30**	**AP**
Our Models									
1	BiCNN	0.2129	0.1634	0.2028	0.1176	0.2367	0.1284	0.3788	0.2557
2	BiCNN+QAtt	0.3966^{1}	0.3586^{1}	0.3904^{1}	0.2376^{1}	0.4861^{1}	0.2696^{1}	0.6388^{1}	0.4226^{1}
3	BiCNN+PAtt	$0.4469^{1,2}$	$0.4135^{1,2}_{5}$	0.4017^{1}_{5}	$0.2413^{1,5}$	$0.5167^{1,2}_{5}$	$0.2817^{1,2}$	$0.6642^{1,2}$	$0.4351^{1,2}_{5}$
4	BiCNN+PAtt+QL	$\mathbf{0.4735}^{1\text{-}3}_{5\text{-}7}$	$\mathbf{0.4346}^{1\text{-}3}_{5,6}$	$\mathbf{0.4164}^{1,2}_{5,6}$	$\mathbf{0.2516}^{1\text{-}3}_{5}$	$0.5256^{1\text{-}3}_{5,6}$	$\mathbf{0.2965}^{1\text{-}3}_{5}$	$\mathbf{0.6752}^{1,2}$	$\mathbf{0.4522}^{1\text{-}3}_{5,7}$
Existing Models									
5	QL	0.4000^{1}	0.3576^{1}	0.3311^{1}	0.2091^{1}	0.4450^{1}	0.2532^{1}	0.6182^{1}	0.3924^{1}
6	RM3	0.4211^{1}	0.3824^{1}	0.3452^{1}	0.2342^{1}	0.4733^{1}	0.2766^{1}	0.6339^{1}	0.4480^{1}
7	MP-HCNN(+URL)	$0.4075^{1,2}$	$0.3832^{1,2}$	$0.3689^{1,5}$	$0.2337^{1,5}$	$0.5222^{1,2}_{5}$	$0.2818^{1,2}_{5}$	0.6297^{1}	0.4304^{1}
8	MP-HCNN(+URL)+QL	$0.4293^{1,2}_{5}$	$0.4043^{1,2}_{5,6}$	$0.3791^{1,5}_{6}$	$0.2460^{1,5}$	$\mathbf{0.5294}^{1\text{-}3}_{5,6}$	$0.2896^{1,2}_{5}$	0.6394^{1}	$0.4420^{1,5}$

Table 3: Results of various models on the TREC Microblog Tracks datasets. Models 5–8 are copied from Rao et al. (2019); note that MP-HCNN exploits URL information (+URL). Models with +QL include interpolation with the QL baseline. BiCNN denotes our general sentence encoder architecture, with either query-aware attention (QAtt) or position-aware attention (PAtt). Superscripts and subscripts indicate the row indexes for which a metric difference is statistically significant at $p < 0.05$.

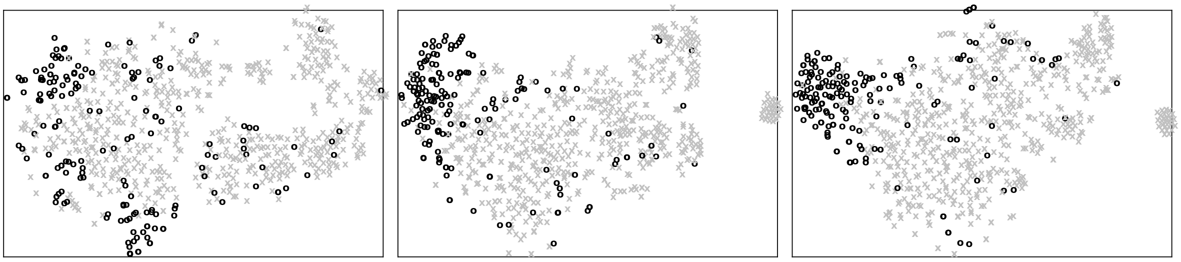


Figure 4: t-SNE visualizations of hidden states for the best-performing query 164 "*lindsey vonn sidelined*" from the BiCNN (left), QAtt (middle), and PAtt (right). Black circle (grey cross) represents relevant (non-relevant) post.

Match	Count
Oscars	28
snub	20
Affleck	25
Oscars snub	18
snub Affleck	15
Oscars Affleck	23
Oscars snub Affleck	13

Table 4: Matching patterns for the worst-performing query 127 "*Oscars snub Affleck*".

For the worst-performing query 125 "*Oscars snub Affleck*", the PAtt model lost 0.47 in AP and 0.11 in P30. To diagnose what went wrong, we sampled the top 30 posts ranked by the PAtt model and counted the number of posts that contain different combinations of the query terms in Table 4. The PAtt model indeed captures matching patterns, mostly on *Oscars* and *Affleck*. However, from the relevance judgments we see that *snub* is the dominant term in most relevant posts, while *Oscars* is often expressed implicitly. For example, QL assigns more weight to the term *snub* in the relevant post "*argo wins retributions for the snub of ben affleck*" because of the term's rarity; in contrast, the position-aware encoder places emphasis on the wrong query terms.

Model Performance. Finally, in terms of training and inference speed, we compared the PAtt model with MP-HCNN on a machine with a GeForce GTX 1080 GPU (batch size: 300). In addition to being more effective (as the above results show), PAtt is also approximately 4× faster.

5 Conclusions

In this paper, we proposed two novel attention-based convolutional encoders to incorporate query-aware and position-aware information with minimal additional model complexity. Results show that our model is simpler, faster, *and* more effective than previous neural models for searching social media posts.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *TREC*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM*, pages 126–134.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv:1705.03122*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, pages 55–64.

Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. 2016. UMD-TTIC-UW at SemEval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *SemEval-2016*, pages 1103–1108.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Jimmy Lin and Miles Efron. 2013. Overview of the TREC-2013 Microblog Track. In *TREC*.

Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *TREC*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9:2579–2605.

Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *EMNLP*, pages 1849–1860.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *WWW*, pages 1291–1299.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *NAACL*, pages 291–296.

Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. 2011. Overview of the TREC-2011 Microblog Track. In *TREC*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A new deep architecture for relevance ranking in information retrieval. In *CIKM*, pages 257–266.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*, pages 1913–1916.

Jinfeng Rao, Hua He, Haotian Zhang, Ferhan Ture, Royal Sequiera, Salman Mohammed, and Jimmy Lin. 2017. Integrating lexical and temporal signals in neural ranking models for searching social media streams. *arXiv:1707.07792*.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2019. Multi-perspective relevance matching with hierarchical ConvNets for social media search. In *AAAI*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv:1611.01603*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, pages 373–382.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*, pages 373–374.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, pages 623–632.

Ian Soboroff, Iadh Ounis, Craig Macdonald, and Jimmy Lin. 2012. Overview of the TREC-2012 Microblog Track. In *TREC*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *CoNLL*, pages 271–280.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR*, pages 55–64.

AttentiveChecker: A Bi-Directional Attention Flow Mechanism for Fact Verification

T.Y.S.S.Santosh, Vishal G, Avirup Saha, Niloy Ganguly
Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
India
`{santoshtyss, arsrivish2, saha.avirup}@gmail.com , niloy@cse.iitkgp.ac.in`

Abstract

The recently released FEVER dataset provided benchmark results on a fact-checking task in which given a factual claim, the system must extract textual evidence (sets of sentences from Wikipedia pages) that support or refute the claim. In this paper, we present a completely task-agnostic pipelined system, AttentiveChecker, consisting of three homogeneous Bi-Directional Attention Flow (BIDAF) networks, which are multi-layer hierarchical networks that represent the context at different levels of granularity. We are the first to apply to this task a bi-directional attention flow mechanism to obtain a query-aware context representation without early summarization. AttentiveChecker can be used to perform document retrieval, sentence selection, and claim verification. Experiments on the FEVER dataset indicate that AttentiveChecker is able to achieve the state-of-the-art results on the FEVER test set.

1 Introduction

The rising influence of fake news poses a clear threat to ethical journalism and the future of democracy. In order to tackle the sheer volume of fake news produced, robust automatic techniques to counter it need to be developed. To that end, in order to facilitate researchers to develop algorithms, a number of fact checking datasets have been released in the recent past (Vlachos and Riedel, 2014), (Wang, 2017), (Ferreira and Vlachos, 2016), (Pérez-Rosas et al., 2017), the 2017 Fake News Challenge (Pomerleau and Rao, 2017) dataset, the dataset released against Triple Scoring Task at the WSDM Cup 2017 (Heindorf et al., 2017) etc.

However, none of these datasets provide manual annotation for sentence or phrase-level evidence. In this work, we experiment on the Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018a), which is one of the first which provides sentence-level annotations. (An Arabic corpus (Baly et al., 2018) has been recently released.) A shared task corresponding to the dataset was floated that required verification of an input claim with potential evidence in a large database of about 5 million Wikipedia documents, and also provided a standardized benchmark setting which enabled easy and fair comparison. Table 1 lists the dataset splits and sizes.

Split	Supports	Refutes	NotEnough Info
train	80,035	29,775	35,639
dev	3,333	3,333	3,333
test	3,333	3,333	3,333
reserved	6,666	6,666	6,666

Table 1: Statistics of claims in FEVER

Several attempts have been made to tackle the task defined by FEVER, the most notable ones being (Nie et al., 2018; Yoneda et al., 2018; Hanselowski et al., 2018) which secured the 1st, 2nd and 3rd place on this shared task respectively (Thorne et al., 2018b). Diverse methods were applied, mostly using task-specific features which allowed them to beat the baseline given by (Thorne et al., 2018a). In this paper, we propose a completely task-agnostic system, AttentiveChecker to tackle the FEVER task.

AttentiveChecker is a pipelined system consisting of three identical Bi-Directional Attention Flow (BIDAF) networks, which are multi-layer hierarchical networks that represent the context at different levels of granularity. We use a bi-directional attention flow mechanism where we allow the context vector at each step to flow to the next layers in the BIDAF model. This helps to obtain a query-aware context representation without

Proceedings of NAACL-HLT 2019, pages 2218–2222
Minneapolis, Minnesota, June 2 - June 7, 2019.

early summarization. This is different from previously used attention layers employed in (Sordoni et al., 2016), (Shen et al., 2017) where the query and context are summarized into a single feature vector. AttentiveChecker achieves a FEVER Score of 66.72 on the test set, which beats the 1st ranked system (Nie et al., 2018) by more than 2 points.

2 Task Definition

The task can be described as verifying a claim using evidence from Wikipedia. The system must label the claim as SUPPORTED or REFUTED based on the evidence from Wikipedia or NotEnoughInfo if there is not sufficient evidence to either support or refute it. The system must also extract textual evidence (a set of sentences from Wikipedia pages) that support or refute the claim. A prediction is said to be correct only if both (a). the label is correct and (b). the predicted evidence set (containing at most five sentences) covers the annotated evidence set. The accuracy in percentage of such predictions is called the FEVER score. The overall task can be compartmentalized into three distinct subtasks: (i). identifying relevant documents from Wikipedia (Document Retrieval), (ii). selecting sentences forming the evidence from the documents (Sentence Selection) and (iii). classifying the claim w.r.t. collected evidence (Claim Verification).

2.1 Document Retrieval

For this sub-task, we provide the first sentence of the document and the claim as the two input sequences to the BIDAF model which outputs the probability of selecting the current document as evidence. As the number of documents is huge, we first reduce the search space by performing keyword match with titles of Wikipedia pages i.e. the document is selected if there is an exact match between the title and a span of text in the input claim. We use our BIDAF model to rank the chosen documents in order of relevance. The top-k documents based on their score are shortlisted for the next level.

2.2 Sentence Selection

For this sub-task, we provide (i). each sentence of the documents in the evidence set and (ii). the claim as the two input sequences to the BIDAF model which outputs the probability of selecting the current sentence as an evidential sentence. Since the search space is already reduced by Document Retrieval, we can directly traverse all the sentences and compare them with the claim using the BIDAF model. We rank all the sentences in every document from the evidence set and choose the top-k sentences.

2.3 Claim Verification

For this sub-task, we provide (i) the claim, and (ii) all evidential sentences together with the corresponding document names (to address coreference issues) as the two input sequences to our BIDAF model. The BIDAF model outputs the scores for three labels, namely SUPPORTED, REFUTED and NotEnoughInfo. Then the claim is labelled as the one with the highest score. Note that in order to have fair comparison across methods, the FEVER challenge limits the sentences in the evidence to a maximum of five. Also in the test set of FEVER data, the number of sentences providing evidence is at most five.

3 The BIDAF Model

In this section, we will describe the architecture of the Bi-Directional Attention Flow (BIDAF) model which constitutes the basic building block of AttentiveChecker. In each stage of the pipeline, the BIDAF model takes two input sequences and outputs labels based on the particular sub-task being performed. Let X and Y denote two input word sequences of length m and n respectively. The BIDAF model consists of four layers: (i). the embedding layer takes raw text sequences X and Y as input and encodes them into suitable vector sequences $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$, (ii). the attention layer takes $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ as input, computes the attention scores of each sequence w.r.t. the other, and outputs two attended sequences $\mathbf{C}$ and $\mathbf{D}$, (iii). the modeling layer takes $\mathbf{C}$ and $\mathbf{D}$ as input and outputs two fixed size vectors $\hat{C}$ and $\hat{D}$ which capture the semantic similarity between the two sequences, and (iv). the output layer takes $\hat{C}$ and $\hat{D}$ as input and provides the scores for the output labels. The layers are described below in detail.

Embedding Layer: In the embedding layer the input sequences are encoded at three levels of granularity viz. character, word and context. We obtain the character-level embedding of each word using Convolutional Neural Networks (CNN) as described in (Kim, 2014). For word level encoding, we use pre-trained word vectors, to obtain the word embedding of each word in the input sequences. Corresponding to each word, we output a vector concatenating the word level and character level encoding. Let $\mathbf{X}'$ denote the sequence of concatenated vectors for the input word sequence X. We use a Bi-directional Long Short-Term Memory Network (BiLSTM) on $\mathbf{X}'$, to model the temporal interactions between words within each sequence and thus obtain the contextual embedding :

$$\hat{\mathbf{A}} = \mathbf{BiLSTM}(\mathbf{X}') \in \mathbb{R}^{d_0 \times m}$$

where $\hat{\mathbf{A}}$ denotes the sequence of all output vectors of the BiLSTM. Similarly we get a sequence $\hat{\mathbf{B}} \in \mathbb{R}^{d_0 \times n}$ for the sequence Y. Note that the character-level embeddings obviate the need for task-specific embeddings or features (e.g. those used by (Nie et al., 2018) for claim verification), as character-level embeddings can encode a far more generalized set like numeric sequences, misspellings, emoticons or other languages.

Attention Layer: Intuitively, the attention layer give higher importance or weight to those parts of a sequence which overlap with parts of the other sequence. We compute attention for the two sequences $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ with respect to each other. Here we compute attention in both directions: from first sequence to second sequence and vice versa. To achieve this we make use of a similarity matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ where $\mathbf{S_{ij}}$ indicates the similarity (or attention score) between the i^{th} word in the first sequence and the j^{th} word in the second sequence and is computed by applying a linear mapping after a single layer perceptron stage on the i^{th} column vector of the first sequence and the j^{th} column vector of the second sequence (Hermann et al., 2015).

$$\mathbf{S_{ij}} = \mathbf{W_1}^{\intercal}.\tanh(\mathbf{W_2}.[\hat{\mathbf{A}}_i : \hat{\mathbf{B}}_j] + \mathbf{b}) \in \mathbb{R}$$

where $\mathbf{W_1}$, $\mathbf{W_2}$ indicate trainable weight matrices, $\mathbf{b}$ indicates trainable bias matrix, $\hat{\mathbf{A}}_i$, $\hat{\mathbf{B}}_j$ indicate i^{th} column vector of $\hat{\mathbf{A}}$ and j^{th} column vector of $\hat{\mathbf{B}}$ respectively. : indicates vector concatenation. The context vector for the i^{th} word of the sequence $\hat{\mathbf{A}}$ w.r.t. the sequence $\hat{\mathbf{B}}$ is given by

$$\tilde{\mathbf{A}}_i = \sum_j \alpha_{ij} \hat{\mathbf{B}}_j \in \mathbb{R}^{d_0}$$

where $\alpha_{ij} = \text{softmax}_j(\mathbf{S_{ij}}) = \frac{\exp(\mathbf{S_{ij}})}{\sum_{j=1}^{n} \exp(\mathbf{S_{ij}})}$.

Finally we obtain the attention vector sequence for the sequence X as $\mathbf{C} \in \mathbb{R}^{d_1 \times m}$ by adding ReLU after applying single layer perceptron to the vector obtained by concatenating the i^{th} column vector of contextual embedding ($\hat{\mathbf{A}}$) and i^{th} column vector of context vectors ($\tilde{\mathbf{A}}$):

$$\mathbf{C}_i = \max(0, \mathbf{W}.[\hat{\mathbf{A}}_i : \tilde{\mathbf{A}}_i] + \mathbf{b}) \in \mathbb{R}^{d_1}$$

where $\mathbf{C}_i$, $\mathbf{W}$, $\mathbf{b}$ indicate i^{th} column vector of $\mathbf{C}$ corresponding to the i^{th} word of X, trainable weight matrix, trainable bias matrix, respectively; : indicates concatenation of vectors. Each column vector of $\mathbf{C}$ can be considered as the Y-aware representation of each word in the sequence X. The above computation is repeated to obtain the context vector $\tilde{\mathbf{B}}$ and subsequently the attention vector sequence $\mathbf{D} \in \mathbb{R}^{d_1 \times n}$ for Y.

Modeling Layer: We apply bi-directional LSTM to the obtained sequence $\mathbf{C}$ for X to obtain a new sequence $\tilde{\mathbf{C}}$

$$\tilde{\mathbf{C}} = \mathbf{BiLSTM}(\mathbf{C}) \in \mathbb{R}^{d_2 \times m}$$

and then take the concatenation of the final forward and backward outputs of the BiLSTM to obtain a fixed size vector representation $\hat{C} = \overleftarrow{\tilde{\mathbf{C}}}_1 : \overrightarrow{\tilde{\mathbf{C}}}_m \in \mathbb{R}^{2d_2}$ which captures the semantic interaction between the two sequences because of the attention applied in the previous layer. This is different from the embedding layer, which captures the interaction among words of one sequence independent of the other sequence. Similarly we get a sequence $\tilde{\mathbf{D}} \in \mathbb{R}^{d_2 \times n}$ and a vector $\hat{D} = \overleftarrow{\tilde{\mathbf{D}}}_1 : \overrightarrow{\tilde{\mathbf{D}}}_n \in \mathbb{R}^{2d_2}$ for Y.

Output Layer: To quantify the semantic similarity between the two sequences, we apply the inverse exponential of the Manhattan distance (M) as suggested in (Mueller and Thyagarajan, 2016) to the representations obtained from the modeling layer.

$$\mathbf{O} = M(\hat{C}, \hat{D}) = \exp(-||\hat{C} - \hat{D}||_1)$$

Then $\mathbf{O}$ is fed to a single layer perceptron, to obtain the required sub-task specific output. For document retrieval and sentence selection, we have one output neuron in the single layer perceptron which indicates the probability of selecting the current document or sentence as evidence, while for claim verification, we have three output neurons indicating the scores for three labels, namely SUPPORTED, REFUTED and NotEnoughInfo.

We note that although the output layer must necessarily be sub-task specific (since the objective of each subtask is different), however the difference in architecture is only in the number of output neurons.

4 Results

In this section we first present the results for the full system and then the ablation results for each stage of the pipeline.

Full Pipelined system: We evaluated our complete system with all components on the test set by setting k=5 for both document retrieval and sentence selection. We found that the accuracy values obtained by AttentiveChecker (a). with the requirement to provide correct sentences as evidence (FEVER Score) and (b). without the requirement to provide correct sentences as evidence (Label Accuracy) for the SUPPORTED/REFUTED labels are 66.72 and 69.98 respectively. Table 4 compares performance of AttentiveChecker with two baselines (i). the FEVER baseline given by (Thorne et al., 2018a) and (ii). NSMN (Nie et al., 2018). We observe that AttentiveChecker performs better than the baselines on the overall task.

Task	Metric	FEVER Baseline	NSMN	Our System
Full System	Label Acc.	50.91	68.16	69.98
Full System	FEVER Score	31.87	64.23	**66.72**

Table 2: Performance of AttentiveChecker on the test set vis-a-vis the baseline systems for k=5.

NSMN (Nie et al., 2018) is a homogeneous BiLSTM-based pipeline but still uses task-specific features as input for claim verification. The key difference in AttentiveChecker compared to NSMN is the attention layer which we claim to be a better way of matching corresponding parts of the two sequences than the sequence alignment which is done in the analogous 'alignment layer' of NSMN. This attention layer is the major reason behind our improvement over NSMN and justifies the name of our system.

We now present the ablation results for the individual pipeline stages.

Document Retrieval & Sentence Selection: Table 4 shows the performance of our document retrieval and sentence selection systems on the dev set for different values of k (no. of documents/sentences retrieved). We report the Oracle Accuracy which is the upper bound of the FEVER score assuming perfect downstream stages.

k	Oracle Acc. (Document Retrieval)	Oracle Acc. (Sentence Selection)
3	93.30	82.37
5	94.05	88.23
7	94.15	88.24

Table 3: Performance of the retrieval systems on the dev set for top-k retrieval using k = 3, k = 5, and k =7

Claim Verification: To understand how well AttentiveChecker performs in this sub-task, we performed an oracle evaluation on the dev set by providing a gold standard evidence set and achieved an accuracy of 90.63 (this has to be done with $k = 5$ as per requirement of the shared task). The ablation results are summarized in Table 4 for k=5 and compared with the baselines. We observe that AttentiveChecker performs better in most sub-tasks (except sentence selection where it falls marginally short) compared to the baselines.

Task	Metric	FEVER Baseline	NSMN	Our System
Doc. Retrieval	Oracle Acc.	70.20	92.42	94.05
Sent. Selection	Oracle Acc.	62.81	91.19*	88.23
Claim Verif.	Oracle Acc.	88.00	N/A	90.63

Table 4: Ablation results of pipeline stages on the dev set vis-a-vis the baseline systems for k=5.
*With Annealed Sampling. Score without it is 86.65.

5 Conclusion

We developed a homogeneous BIDAF model for all three FEVER subtasks achieving the state-of-the-art on the overall task. Our system is completely task-agnostic and can therefore be transferred to other similar tasks (e.g. exaggeration detection) if need be. For the first time in this task, we have used a query-aware bi-directional attention model that avoids early summarization. Although the improvement in the FEVER Score appears modest (2 points), it is still significant considering the hardness of the problem.

References

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating stance detection and fact checking in a unified corpus. *arXiv preprint arXiv:1804.08012*.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint arXiv:1809.01479*.

Stefan Heindorf, Martin Potthast, Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2017. Wsdm cup 2017: Vandalism detection and triple scoring. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 827–828. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining fact extraction and verification with neural semantic matching networks. *arXiv preprint arXiv:1811.07039*.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Dean Pomerleau and Delip Rao. 2017. Fake news challenge.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and verification (fever) shared task. *arXiv preprint arXiv:1811.10971*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Ucl machine reading group: Four factor framework for fact finding (hexaf). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102.

Practical, Efficient, and Customizable Active Learning for Named Entity Recognition in the Digital Humanities

Alexander Erdmann,†‡ **David Joseph Wrisley,**‡ **Benjamin Allen,**† **Christopher Brown,**†
Sophie Cohen-Bodénès,* **Micha Elsner,**† **Yukun Feng,**† **Brian Joseph,**†
Béatrice Joyeux-Prunel* **and Marie-Catherine de Marneffe**†
†Ohio State University, USA ‡New York University Abu Dhabi, UAE
*Ecole Normale Supérieure, France
{ae1541,djw12}@nyu.edu,{allen.2021,brown.2583,elsner.14,
feng.749,joseph.1,demarneffe.1}@osu.edu,
sophiebodenes@gmail.com, beatrice.joyeux-prunel@ens.fr

Abstract

Scholars in inter-disciplinary fields like the Digital Humanities are increasingly interested in semantic annotation of specialized corpora. Yet, under-resourced languages, imperfect or noisily structured data, and user-specific classification tasks make it difficult to meet their needs using off-the-shelf models. Manual annotation of large corpora from scratch, meanwhile, can be prohibitively expensive. Thus, we propose an active learning solution for named entity recognition, attempting to maximize a custom model's improvement per additional unit of manual annotation. Our system robustly handles any domain or user-defined label set and requires no external resources, enabling quality named entity recognition for Humanities corpora where such resources are not available. Evaluating on typologically disparate languages and datasets, we reduce required annotation by 20-60% and greatly outperform a competitive active learning baseline.

1 Introduction

Reaping the benefits of recent advances in Named Entity Recognition (NER) is challenging when dealing with under-resourced languages, niche domains, imperfect or noisily structured data, or user-specific classification tasks. Scholars in inter-disciplinary fields like the Digital Humanities (DH) are increasingly interested in semantic annotation of specialized corpora that invoke many of these challenges. Thus, such corpora cannot easily be annotated automatically with blackbox, off-the-shelf NER models. Manual annotation of large corpora from scratch, meanwhile, can be prohibitively costly. Successful DH initiatives like the Pelagios Commons (Simon et al., 2016), which collects geospatial data from historical sources, often require extensive funding, relying on considerable manual annotation (Simon et al., 2017).

To this end, we introduce the Humanities Entity Recognizer (HER),[1] a whitebox toolkit for build-your-own NER models, freely available for public use. HER robustly handles any domain and user-defined label set, guiding users through an active learning process whereby sentences are chosen for manual annotation that are maximally informative to the model. Informativeness is determined based on novel interpretations of the *uncertainty*, *representativeness*, and *diversity* criteria proposed by Shen et al. (2004). In contrast to literature emphasizing the disproportionate or exclusive importance of uncertainty (Shen et al., 2017; Zhu et al., 2008; Olsson, 2009), we observe significant improvements by integrating all three criteria.

In addition to a robust active learning based NER toolkit, we also contribute a novel evaluation framework. This *inclusive* framework considers the accuracy with which an entire corpus is annotated, regardless of which instances are annotated manually versus automatically, such that no instance is held out when the active learning algorithm considers candidates for annotation. The standard, *exclusive* evaluation framework, by contrast, only measures the accuracy of the final trained model's predictions on a held out test set. Thus, the inclusive framework is relevant to the user who wants to annotate a finite corpus as fast and as accurately as possible by any means necessary, whereas the exclusive framework is relevant to the user who wants to build an NER tool that can generalize well to other corpora.

We conduct extensive experiments comparing several combinations of active learning algorithms and NER model architectures in both frameworks across many typologically diverse languages and domains. The systematic differences between inclusive and exclusive results demonstrate that while deep NER model architectures

---

[1] `github.com/alexerdmann/HER`.

Proceedings of NAACL-HLT 2019, pages 2223–2234
Minneapolis, Minnesota, June 2 - June 7, 2019.

(Lample et al., 2016) are highly preferable for tagging held out sentences, shallow models (Lafferty et al., 2001) perform better on sentences that could have been chosen for manual annotation but were not selected by the active learning algorithm. We argue for the importance of considering both frameworks when evaluating an active learning approach, as the intended application determines which framework is more relevant and thus, which model should be employed. Controlling for the NER model, HER's active learning sentence ranking component achieves significant improvement over a competitive baseline (Shen et al., 2017). Because HER does not reference the inference model during sentence ranking, this provides counter evidence to Lowell et al. (2018)'s hypothesis that *non-native* active learning is suboptimal.

2 Related Work

The best known NER systems among humanists are Stanford NER (Finkel et al., 2005), with pretrained models in several languages and an interface for building new models, and among researchers interested in NER for spatial research, the Edinburgh Geoparser (Grover et al., 2010), with fine grained NER for English. Erdmann et al. (2016) and Sprugnoli (2018), among others, have shown that such off-the-shelf models can be substantially improved on DH-relevant data. Work such as Smith and Crane (2001) and Simon et al. (2016) represent a large community mining such data for geospatial entities. Additional DH work on NER concerns the impact of input data structure and noisy optical character recognition (Van Hooland et al., 2013; Kettunen et al., 2017).

Low Resource NER Language agnostic NER is highly desirable, yet limited by the data available in the least resourced languages. Curran and Clark (2003) demonstrate that careful feature engineering can be typologically robust, though data hungry neural architectures have achieved state-of-the-art performance without feature engineering (Lample et al., 2016). To enable neural architectures in low resource environments, many approaches leverage external resources (Al-Rfou et al., 2015). Cotterell and Duh (2017), for instance, harvest silver annotations from structured Wikipedia data and build models for typologically diverse languages, though their approach is limited to specific domains and label sets. Lin and Lu (2018) adapt well-resourced NER systems to low resource target domains, given minimal annotation and word embeddings in domain. Several translation-based approaches leverage better resourced languages by inducing lexical information from multi-lingual resources (Bharadwaj et al., 2016; Nguyen and Chiang, 2017; Xie et al., 2018). In a slightly different vein, Shang et al. (2018) use dictionaries as distant supervision to resolve entity ambiguity. Unfortunately, external resources are not always publicly available. It is in fact impossible to replicate many of the above studies without a government contract and extensive knowledge of linguistic resources, limiting their applicability to many DH scenarios. Mayhew et al. (2017) suggest manually building bilingual dictionaries when no other translation resources are available to facilitate their method, though active learning provides a more direct means of improving NER quality.

Active Learning Active learning seeks to maximize the performance of a model while minimizing the manual annotation required to train it. Shen et al. (2004) define three broad criteria for determining which data will be most *informative* to the model if annotated: *uncertainty*, where instances which confuse the model are given priority; *diversity*, where instances that would expand the model's coverage are prioritized; and *representativeness*, prioritizing instances that best approximate the true distribution over all instances. Uncertainty-based approaches outperform other single-criterion approaches, though many works, primarily in Computer Vision, demonstrate that considering diversity reduces repetitive training examples and representativeness reduces outlier sampling (Roy and McCallum, 2001; Zhu et al., 2003; Settles and Craven, 2008; Zhu et al., 2008; Olsson, 2009; Gu et al., 2014; He et al., 2014; Yang et al., 2015; Wang et al., 2018b).

For active learning in NER, Shen et al. (2017) propose the uncertainty-based metric maximized normalized log-probability (MNLP). It prioritizes sentences based on the length normalized log probability of the model's predicted label sequence. To make neural active learning tractable, they shift workload to lighter convolutional neural networks (CNN) and update weights after each manual annotation batch instead of retraining from scratch. They demonstrate state-of-the-art performance with MNLP, though Lowell et al. (2018) show its improvement above random sampling to be less dramatic, as do our experiments. Low-

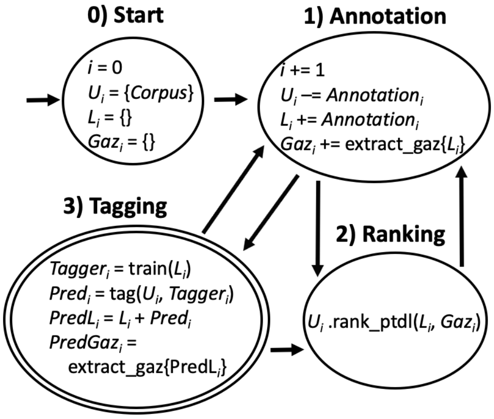


Figure 1: High level HER system architecture. Unlabeled sentences in U are manually labeled and moved to L, enabling iterative updates of gazetteers, the NER model, and the informativity ranking of sentences in U.

ell et al. (2018) compare calculating MNLP from the *native* inference model and from a *non-native* model with a separate architecture. They conclude that non-native models are ill-suited to active learning, which our findings using more robust informativeness criteria contradict.

3 The Humanities Entity Recognizer

As illustrated in Figure 1, HER consists of three components: (1) a human $User$ who provides an unlabeled corpus U at state 0 and annotates selected sentences in state 1, thus moving them from U to the labeled corpus L, (2) an active learning engine, $Ranker$, that ranks sentences from U in state 2 for $User$ to annotate based on how informative they might be to (3), the NER model, $Tagger$, to be trained on L in state 3.[2]

All states are linked by an interface that "white-boxes" the process. It advises $User$ on qualitative observations which might improve performance by manually interacting with $Ranker$, e.g., removing problematic gazetteer entries, or with $Tagger$, e.g., forcing it to sample some known minority labels. The contributions of the interface will not be reflected in our human-out-of-the-loop experiments on standard datasets, as these evaluate only the contributions of $Ranker$ and $Tagger$. Thus, reported performances should be considered a lower bound that can often be improved with minimal human intervention.

3.1 Ranking Sentences by Informativeness

At state 1 with i=0, $User$ is prompted to annotate randomly ordered sentences until 50-100 named entities are labeled. We use a 200-sentence seed for all experiments except that of Section 4.1, where an entity-sparse corpus requires a 300-sentence seed. Such a small seed, often annotated in less than 30 minutes, is sufficient to support $Ranker$'s *Pre-Tag DeLex* (PTDL) algorithm in state 2. PTDL uses only shallow, fast Conditional Random Fields (CRF) to avoid delaying manual annotation. As demonstrated on a sample corpus in Figure 2, PTDL involves four subtasks: *pre-tagging*, *delexicalized tagging*, *vocabulary weighting* and *sentence ranking*.

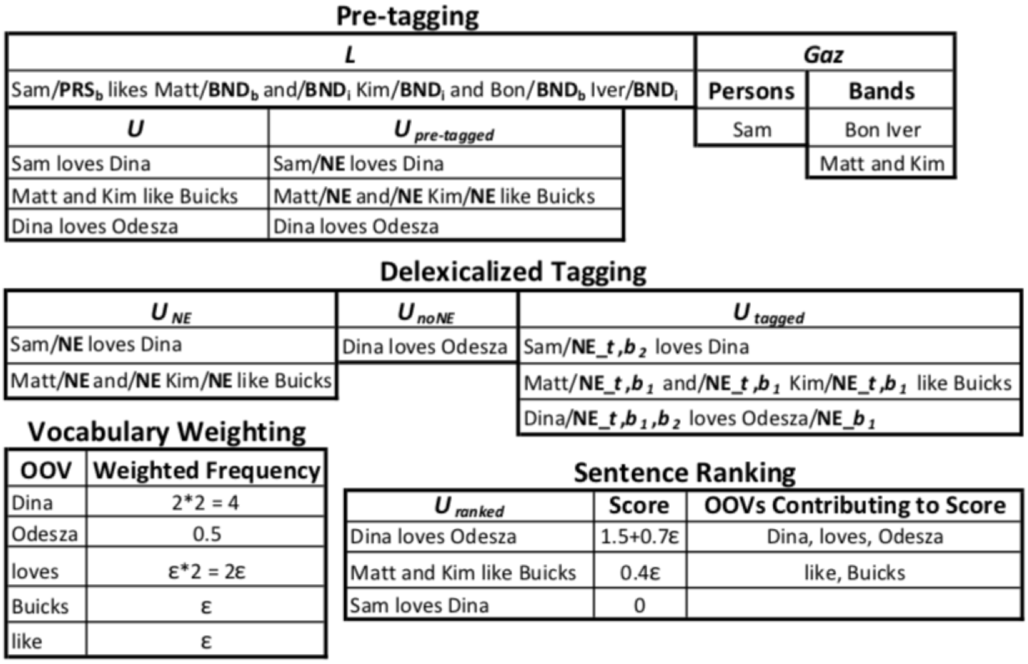
Pre-tagging We naively and greedily *pre-tag* U with binary entity–non-entity labels based on gazetteer matches. Hence, every n-gram in U matching a named entity from a gazetteer gets pre-tagged as such unless it overlaps with a longer named entity. State 2 cannot occur until the seed has been annotated, so Gaz will never be empty, as entities are automatically extracted into gazetteers after each annotation batch in state 1.

Delexicalized Tagging $U_{pre-tagged}$ is divided into U_{NE}, containing sentences with at least one pre-tagged named entity, and its complement, U_{noNE}. We train a *trusted* NER model (t) on L and two *biased* models (b_1 and b_2) on L plus random mutually exclusive halves of U_{NE}. b_1 and b_2 are biased in that they use non-gold data (U_{NE}) exhibiting an exaggerated density of named entities. Models are trained using only *delexicalized* features, which, unlike character n-grams for example, do not directly reference the focal word or its form. Many delexicalized features are context-based, like preceding and following words. Trained thus, models are less hampered by the class imbalance problem (Japkowicz, 2000), more likely to predict more named entities, and more capable of determining which Out Of Vocabulary (OOV) lexical items (in U but not L) make good named entity candidates.

Vocabulary Weighting After tagging U with delexicalized models, t, b_1, and b_2, OOVs are scored by weighted frequency. Weights are sums determined by which models tagged the OOV in an entity at least once. t contributes 1 to the sum and each biased model, $\frac{1}{2}$. OOVs not tagged by any model recieve a negligible positive weight, ϵ.

---

[2] In our experiments, we assume no previous annotation or pre-existing gazetteers at state 0, though, in practice, HER robustly leverages non-empty L_0 and/or Gaz_0 when available.



Pre-tagging

L
Sam/**PRS**$_b$ likes Matt/**BND**$_b$ and/**BND**$_i$ Kim/**BND**$_i$ and Bon/**BND**$_b$ Iver/**BND**$_i$

U	*U* $_{pre\text{-}tagged}$
Sam loves Dina	Sam/**NE** loves Dina
Matt and Kim like Buicks	Matt/**NE** and/**NE** Kim/**NE** like Buicks
Dina loves Odesza	Dina loves Odesza

Gaz	
Persons	**Bands**
Sam	Bon Iver
	Matt and Kim

Delexicalized Tagging

U $_{NE}$	*U* $_{noNE}$
Sam/**NE** loves Dina	Dina loves Odesza
Matt/**NE** and/**NE** Kim/**NE** like Buicks	

U $_{tagged}$
Sam/**NE_*t*,*b*** $_2$ loves Dina
Matt/**NE_*t*,*b*** $_1$ and/**NE_*t*,*b*** $_1$ Kim/**NE_*t*,*b*** $_1$ like Buicks
Dina/**NE_*t*,*b*** $_1$,***b*** $_2$ loves Odesza/**NE_*b*** $_1$

Vocabulary Weighting

OOV	Weighted Frequency
Dina	2*2 = 4
Odesza	0.5
loves	ε*2 = 2ε
Buicks	ε
like	ε

Sentence Ranking

U $_{ranked}$	Score	OOVs Contributing to Score
Dina loves Odesza	1.5+0.7ε	Dina, loves, Odesza
Matt and Kim like Buicks	0.4ε	like, Buicks
Sam loves Dina	0	

Figure 2: Step-by-step example outputs from ranking unlabeled sentences in a sample corpus with PTDL.

This motivates PTDL to target frequent OOVs after exhausting OOVs more likely to be named entities.

Sentence Ranking As shown in Figure 2, sentences in U are ranked by the sum of scores of unique OOVs therein, normalized by the word length of the sentence. OOVs occurring in higher ranked sentences do not count toward this sum.

While typical active learning strategies for NER rely on the inference model's output probabilities, these are noisy, especially given scarce annotation. Data-scarce models lexically memorize training instances, yielding high precision at the expense of recall. They struggle to model non-lexical features more subtly correlated with entity status but also more likely to occur on OOVs. Hence, data-scarce models know what they know but are somewhat equally perplexed by everything else (Li et al., 2008). For this reason, uncertainty-based active learners can suffer from problematically weak discriminative power in addition to redundant and outlier-prone sampling.

By forcing reliance on delexicalized features and biasing models toward recall, our three-criteria approach identifies frequent (representativeness) OOV words (diversity) that are plausible candidate members of named entities. These make for better indicators of where the model may fail (uncertainty) because named entities are minority labels in NER and minority labels are challenging.

3.2 Sentence Tagging Architectures

$User$ can stop iteratively annotating and re-ranking U at any time to train a $Tagger$ on L to perform the full NER task on U (state 3). L is combined with $Tagger$'s predictions on U ($Pred$) to form $PredL$, from which an imperfect gazetteer is extracted ($PredGaz$). $User$ must inspect these to determine if additional annotation is required. We explore three $Tagger$ architectures:

CRF For tagging with Okazaki (2007)'s feature-based CRF, $Tagger$ first trains preliminary models on L, cross-validating on folds of the random seed. Each model leverages a unique permutation drawn from a universal set of features. The best performing feature set is used to train the final model. Training and inference are fast, even with preliminary cross-validation. In the exclusive evaluation, CRF is the best tagger until about 40K tokens of training data are acquired. In the inclusive evaluation, CRF's tendency to overfit is rewarded, as it outperforms both deep models regardless of corpus size.

CNN-BiLSTM The near state-of-the-art architecture proposed by Shen et al. (2017) aims to reduce training with minimal harm to accuracy. It leverages CNNs—as opposed to slower recurrent networks—for character and word encoding, and a bidirectional long short-term memory network (BiLSTM) for tags. CNN-BiLSTM outperforms all other models in the exclusive evaluation for a stretch of the learning curve between about 40K tokens acquired and 125K. While faster than the other deep model considered here, training time is slower than the CRF and computationally costly.

BiLSTM-CRF The state-of-the-art BiLSTM-CRF architecture of (Lample et al., 2016) projects a sequence of word embeddings to a character level BiLSTM which in turn projects into a CRF at the tag level, with an additional hidden layer between the BiLSTM and CRF. In our experiments, BiLSTM-CRF surpasses CNN-BiLSTM performance once about 125K tokens are acquired.

3.3 HER in the Digital Humanities

HER was developed to benefit diverse DH projects. It is currently facilitating three such ventures.

The Herodotos Project (`u.osu.edu/herodotos`) aims at cataloguing ancient ethnogroups and their interactions (Boeten, 2015; de Naegel, 2015). HER is used to identify such groups in Classical Greek and Latin texts. Manually annotated data as well as a trained NER tagger are freely available from `github.com/alexerdmann/Herodotos-Project-Latin-NER-Tagger-Annotation`.

Artl@s `artlas.huma-num.fr` is a global database of art historical catalogs from the 19th and 20th centuries for the scholarly study of the diffusion and globalization of art. HER serves as a method for mining semi-structured texts characterized by noisy OCR and recurrent patterns of granular named entities.

Visualizing Medieval Places Wrisley (2017) concerns the study of recurrent places found within a mixed-genre corpus of digitized medieval French texts. NER has heretofore been challenged by sparsity from the unstandardized orthography. The related Open Medieval French project (`github.com/OpenMedFr`) benefits from HER's robust handling of sparsity, using the system to create open data regarding people and places referenced in medieval French texts.

4 Experiments

We now describe several experiments evaluating HER's performance on diverse corpora. When a standard test set is available, we perform inclusive evaluation on the combined train and dev sets and evaluate exclusively on test. Otherwise, we only evaluate inclusively. In both settings, we compare multiple combinations of ranking systems and taggers over a learning curve, reporting F1 exact match accuracy of identified entities. In all figures, line dashing (contiguous, dashed, or dotted) denotes inference model (CRF, BiLSTM-CRF, or CNN-BiLSTM), whereas line accents (stars, circles, triangles, or squares) denotes active learning method. Besides PTDL, we also consider a random active learning method (RAND), MNLP, and Erdmann et al. (2016)'s CAP algorithm. Like PTDL, CAP ranks sentences based on frequency weighted OOVs, but calculates weights based on capitalization patterns, prioritizing capitalized OOVs occurring in non-sentence initial position.

Quantity of training data is reported as percentage of the entire corpus for inclusive evaluations, and as tokens actively annotated (i.e., not counting the random seed sentences) for exclusive evaluations. For consistency, following seed annotation, we always fetch additional annotation batches at the following intervals, in tokens: 1K, 4K, 5K, 10K, 20K until we reach 100K total tokens, 50K until 300K total, 100K until 500K total, and 250K from there. For all experiments leveraging neural taggers, we use freely available pretrained embeddings (Grave et al., 2018), except for Latin, where we train fasttext (Bojanowski et al., 2017) embeddings on the Perseus (Smith et al., 2000) and Latin Library collections with default parameters (using pretrained embeddings yield small performance boosts that decrease with additional training data). We conclude this section with a direct comparison to the recently proposed active learning pipeline of Shen et al. (2017) and their MNLP ranking algorithm.

4.1 Consistency of Non-deterministic Results

Because the active learning pipeline involves taking a random seed and many of the experiments on larger corpora could not be averaged over several runs, we first measure performance variation as a function of ranking algorithm and quantity of annotation. Figure 3 displays our findings on a sample corpus of about 250K tokens[3] in five diverse, pre-1920 prose genres extracted from the FranText corpus (`www.frantext.fr`) and annotated for

---

[3] HER considers sentence boundaries to be tokens, as this helps users locate words, i.e., the line number will correspond to token number when rendered in CoNLL format.

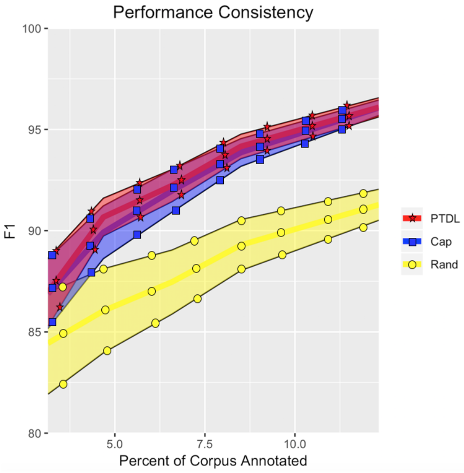


Figure 3: ±1 standard deviation bands around the mean performance of each sentence ranking algorithm using the CRF tagger over 100 inclusive evaluations on our FranText corpus.

geospatial entities. Our sample covers topics from gastronomy to travel, exhibiting inconsistent entity density characteristic of DH corpora.

Noise is much higher for the first few batches of annotation, particularly due to the low recall of data scarce models. Reluctant to generalize, they behave more like look-up tables extracted from the seed, exacerbating the effect of random seed variation. After about 20K tokens annotated or 10% of the corpus, performance becomes much more predictable. All algorithms start with about a 5 point spread for ±1 standard deviation, with means around 70 F, and all exhibit the diminishing variation trend, though RAND does less so. Unlike CAP and PTDL, subsequent annotation batches in RAND are not predictable from previous annotation batches. This results in a spread of 0.76 after annotating 12.33% of the corpus, whereas the other algorithms are close to 0.4.

While we only tested variation on one corpus, multiple runs on other corpora tended to reflect the same diminishing variation trends despite marked differences in entity granularity, density or corpus size. Switching to the exclusive evaluation only minimally increases variation. It was not feasible to rigorously test variation using neural taggers, though we note that they are somewhat more prone to seed related noise which does not diminish as rapidly as it does for CRF with more annotation.

In terms of performance, random annotation requires one to label between 23% and 31% of the corpus to achieve the performance of PTDL after labeling just 12%. For this corpus, PTDL reduces annotation time between 46% and 60%, requiring only 32K tokens from annotators instead of 60-80K. CAP's competitiveness with PTDL is not surprising given that French uses the capitalization standards it is designed to exploit. Both algorithms achieve 15% error reduction above RAND after the first post-seed annotation batch (left edge of Figure 3), increasing monotonically to 55% error reduction after the fifth batch (right edge).

4.2 Inclusive Versus Exclusive Evaluation

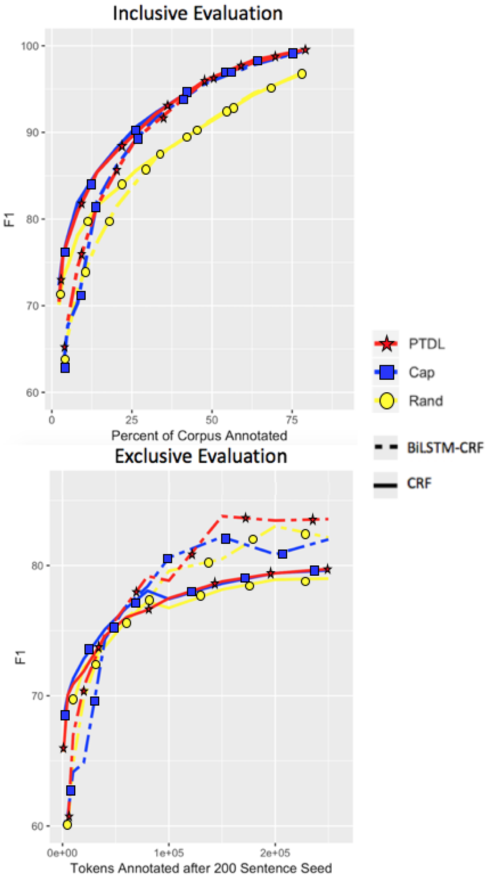


Figure 4: Comparing shallow and deep learning architectures on inclusive and exclusive evaluations with the CoNLL Spanish corpus.

Using the Spanish CoNLL corpus (Tjong Kim Sang and De Meulder, 2003) with canonical train, dev, test splits, we examine the relationship between evaluation framework, $Ranker$, and $Tagger$ in Figure 4.[4] For the inclusive framework, $Ranker$ selects sentences from train+dev for $Tagger$ to train on, and the performance is calculated over the combination of those se-

---

[4] Lample et al. (2016) achieve 85.75 F on the exclusive evaluation, slightly beating our best BiLSTM-CRF models which sacrifice some performance for speed, switching to Adam optimization limited to 5 epochs.

lected sentences (trivially all correct) and trained $Tagger$'s predictions on the rest of train+dev. By reporting results over a learning curve, this evaluation framework is meaningful to the user whose primary goal is to produce a finite annotated corpus as efficiently and accurately as possible, a frequent concern in DH. The standard exclusive framework also gives $Ranker$ access to train+dev sentences, but calculates accuracy from $Tagger$'s predictions on the held out test set. The exclusive framework is thus more meaningful for future users of $Tagger$ who need the tool to generalize to sentences outside of train+dev.

In the inclusive framework, regardless of corpus size, BiLSTM-CRFs do not surpass CRFs until the accuracy is so high that the distinction is negligible. Promoting overfitting by reducing dropout did not significantly affect this result. In the exclusive framework, BiLSTM-CRF surpasses CRF around 50K tokens annotated. This holds for all languages and corpora we investigate, suggesting quantity of data annotated is more predictive of exclusive performance trends, whereas proportion of the corpus annotated better predicts inclusive trends.

4.3 Typology, Granularity, and Corpus Size

We consider the effect of language typology, label scheme granularity, and corpus size on inclusive and exclusive evaluations of taggers and rankers.

4.3.1 Insights from German

We repeat our experiments from Section 4.2 on the German NER corpus, GermEval (Benikova et al., 2014), to determine how robust our findings are to a larger corpus with finer label granularity and different capitalization standards. Our results in Figure 5 confirm many of our previous findings, with BiLSTM-CRFs overtaking CRFs of the same ranker after 50K tokens annotated on the exclusive evaluation. Shallow CRFs again dominate inclusively, and again, exclusive performance is less predictable, though the contribution of PTDL is more obvious.

GermEval contains over 520K tokens to Spanish CoNLL's 321K, showing that deep models are not just slower to overtake shallow models in the inclusive evaluation, but they only asymptotically approach shallow performance.[5] Furthermore, the finer grained named entity distinctions in GermEval do not seem to affect our previous findings, but do cause BiLSTM-CRF to start slowly, as the model does not begin training until all possible labels manifest in the training set. While this is merely an effect of programming choices, it provides interesting insights. For instance, BiLSTM-CRF CAP models consistently start later than RAND which starts later than PTDL, meaning that PTDL is doing well on the diversity criteria, whereas CAP likely struggles because it relies on English-like capitalization standards. Since German capitalizes all nouns, CAP struggles here, having to search through many capitalized OOVs before finding named entities of each category. By not considering uncapitalized OOVs as named entity candidates, it can systematically avoid entire labels which do not take capitalization, such as dates. Thus, while PTDL performs robustly on the GermEval dataset, CAP is only weakly superior to RAND due to the weak correlation between entity status and capitalization.

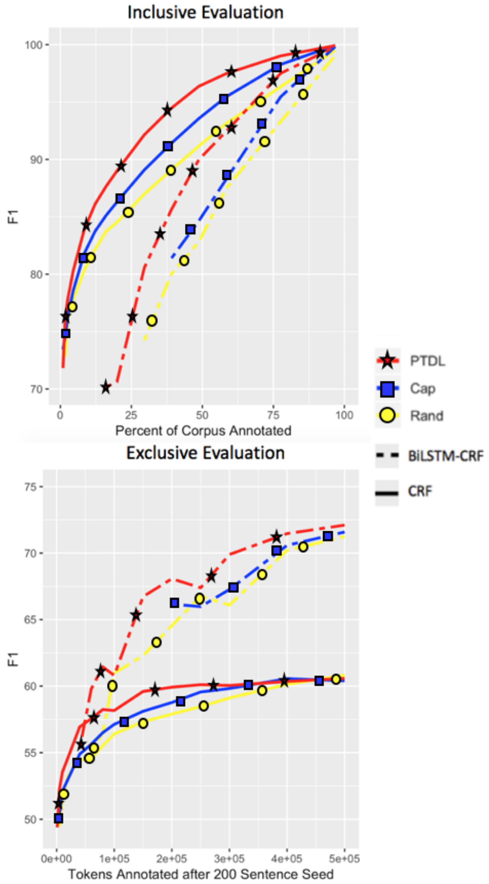


Figure 5: A comparison of shallow and deep learning architectures on inclusive and exclusive evaluations with the GermEval corpus.

---

[5] Our evaluation is equivalent to metric 3 from the shared task (Benikova et al., 2014), though our results are not comparable as we did not leverage nested labels.

4.3.2 Insights from Latin

Latin presents an opportunity to explore the impact of capitalization on ranking algorithms more thoroughly. Erdmann et al. (2016) selected their Latin corpus because English capitalization standards had been imposed during digitization, making CAP more likely to succeed. Figure 6 demonstrates that it even marginally outperforms PTDL on the corpus (left pane). However, capitalizing proper nouns is not a native attribute of Latin orthography and is not available in all digitized manuscripts, limiting the Latin texts in which CAP will succeed. The right pane in Figure 6 demonstrates this, as the same evaluation from the left pane is repeated on a lower cased version of the same corpus. The minuscule error reduction CAP achieves over RAND in this environment is due to its general preference for OOVs. Meanwhile, despite suffering from weaker named entity signals without capitalization, PTDL still manages to robustly identify what non-capitalization features are relevant, maintaining 25% error reduction over RAND. Finally, in German, where capitalization is a weak signal of entity status, PTDL is similarly better equipped to incorporate the weak signal, reducing error twice as much as CAP. Interestingly, PTDL performance in the lower cased Latin corpus almost exactly matches RAND performance on the capitalized version. This suggests the benefits of PTDL are comparable to the benefits of having English-like capitalization to mark entities.

4.3.3 Insights from Arabic

Unlike the other corpora, the news domain ANER Arabic corpus (Benajiba and Rosso, 2007) features rich templatic morphology, frequent lexical ambiguity, and an orthography lacking capitalization. Hence, not only will feature-based signals be more subtle, but the gazetteer-based pre-tagging component of PTDL will suffer from low precision, because Arabic is written in an abjad orthography where short vowels among other characters are seldom marked, making many words polysemous. Even so, PTDL significantly outperforms RAND, likely due to its ability to shift reliance to contextual features better suited for newswire, where formulaic expressions are often used to refer to certain entity types.

While PTDL compares well to RAND, it does not approach 100% accuracy after annotating 50% of the corpus as in Section 4.3.2. Besides ambiguity and lack of capitalization, this could be due to a typological bias in our "universal" feature set. Contiguous character n-grams, for example, will not capture non-concatenative subword phenomena. Going forward, we will investigate which feature sets were most useful as a function of language typology to identify gaps in our coverage.

4.4 Comparing to MNLP

Shen et al. (2017) and Lowell et al. (2018) evaluate the purely uncertainty-based MNLP active NER system on English corpora, reporting starkly different results. We address discrepancies and test the robustness of their findings by comparing MNLP to PTDL and RAND on the GermEval corpus. Results are displayed in Figure 8, with shaded regions corresponding to the range of performance over multiple runs. To compare fairly, we use the same CNN-BiLSTM tagger for all rankers and iteratively update weights instead of re-training from scratch after each annotation batch, as in Shen et al. (2017). We report results on our previously mentioned batch annotation schedule, though results were comparable using the batch schedule of Lowell et al. (2018). Shen et al. (2017) claim iterative updating does not affect accuracy significantly, though the best performing active CNN-BiLSTM in Figure 8 lags a few points behind the BiLSTM-CRF after 150K tokens annotated, with that gap reaching nearly 5 F when training on the whole corpus. Meanwhile, a CNN-BiLSTM trained from scratch on the whole corpus scores only 1 F less than the BiLSTM-CRF.

While Lowell et al. (2018) report improvement over RAND using MNLP when training on 0-10% of the corpus, we see little improvement after about 2%, and even then, PTDL greatly outperforms both. The relationship between the PTDL curves in the exclusive evaluation shows that CNN-BiLSTM is actually the optimal tagging architecture for a brief window, overtaking CRF around 30K tokens and staying in front of BiLSTM-CRF until about 125K tokens.

5 Conclusion and Future Work

We have presented the HER toolkit and its novel active learning algorithm, demonstrating robust handling of typological diversity, niche domains, and minority labels. The algorithm addresses the weak discriminative power of uncertainty-based models caused by class imbalance and precision bias. We also argued for the relevance of in-

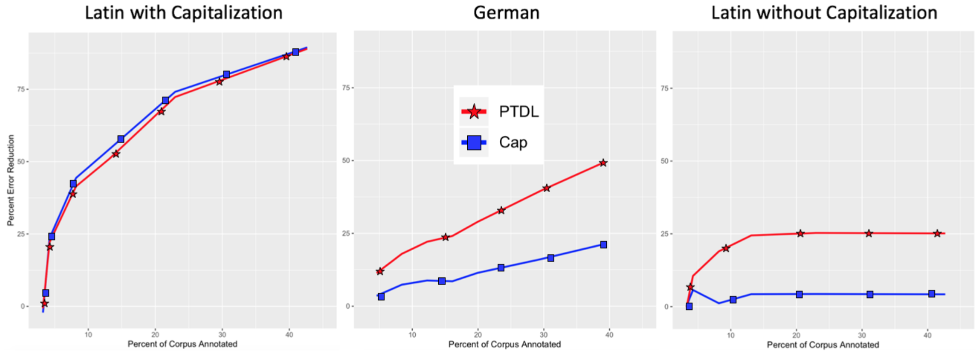


Figure 6: Percent error reduction over RAND in three corpora exhibiting typologically distinct capitalization standards. Corpora are presented in descending order of the correlation of capitalization with named entity status.

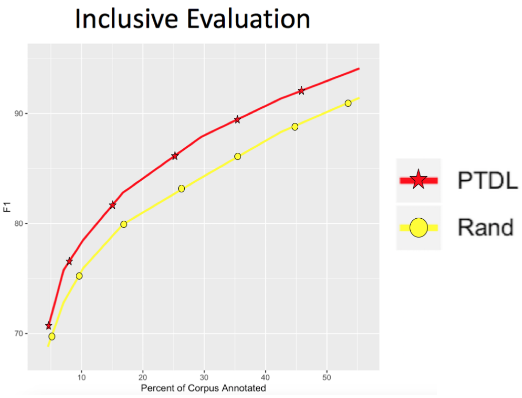


Figure 7: Effect of PTDL in the ANER Arabic corpus.

clusive evaluations, demonstrating that a shallow CRF tagger outperforms deep taggers on sentences which the active learner could have selected for training. The CRF's tendency to overfit is rewarded in this case, as the selected sentences are especially representative of the remaining sentences to be tagged. When tagging held out test sets, CRFs are only optimal until about 30K training tokens are acquired, then CNN-BiLSTMs are preferable until 125K tokens when BiLSTM-CRFs become the best high resourced tagger.

In future work, we will investigate sources of noise in performance to see if these are due to gaps in the model, idiosyncrasies of corpora, or both. Additionally, we will expand HER to model hierarchically nested entity labels. Named entities are often difficult to label deterministically, inviting a problematic level of subjectivity, which is of crucial interest in DH and should not be oversimplified. We will consider strategies such as Wang et al. (2018a)'s shift-reduced-based LSTM architecture or Sohrab and Miwa (2018)'s method of modeling the contexts of overlapping potential named entity spans with bidirectional LSTM's.

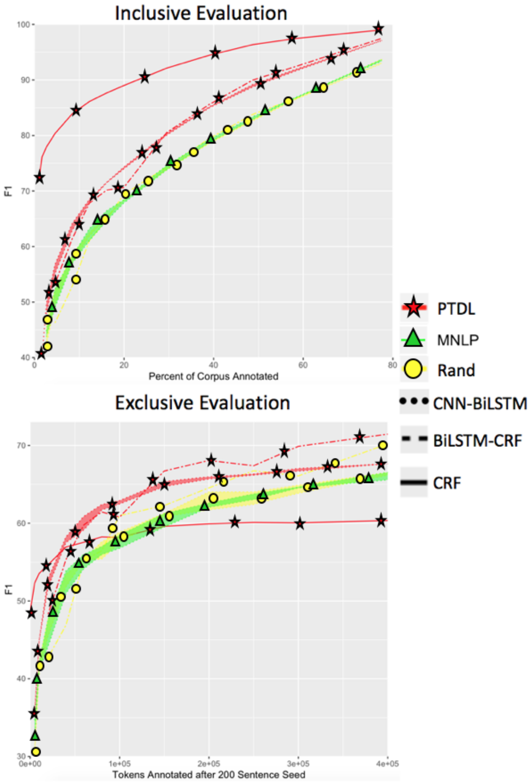


Figure 8: Inclusive and exclusive comparisons of the MNLP and PTDL rankers on GermEval.

Acknowledgments

We thank the Herodotos Project annotators for their contributions: Petra Ajaka, William Little, Andrew Kessler, Colleen Kron, and James Wolfe. Furthermore, we gratefully acknowledge support from the New York University–Paris Sciences Lettres Spatial Humanities Partnership, the Computational Approaches to Modeling Language lab at New York University Abu Dhabi, and a National Endowment for the Humanities grant, award HAA-256078-17. We also greatly appreciate the feedback of three anonymous reviewers.

References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-NER: Massive multilingual named entity recognition. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 586–594. SIAM.

Yassine Benajiba and Paolo Rosso. 2007. Conquering the NER task for the Arabic language by combining the maximum entropy with POS-tag information. In *Proceedings of Workshop on Natural Language-Independant Engineering*.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Pado. 2014. Germeval 2014 named entity recognition shared task: companion paper.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472.

Julie Boeten. 2015. The Herodotos project (OSU-UGent): Studies in ancient ethnography. Barbarians in Strabos geography (Abii–Ionians) with a case-study: the Cappadocians. Master's thesis, Gent Universiteit.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 91–96.

James R Curran and Stephen Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 164–167. Association for Computational Linguistics.

Alexander Erdmann, Christopher Brown, Brian Joseph, Mark Janse, Petra Ajaka, Micha Elsner, and Marie-Catherine de Marneffe. 2016. Challenges and solutions for Latin named entity recognition. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 85–93.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the Edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 368(1925):3875–3889.

Yingjie Gu, Zhong Jin, and Steve C Chiu. 2014. Active learning combining uncertainty and diversity for multi-class image classification. *IET Computer Vision*, 9(3):400–407.

Tianxu He, Shukui Zhang, Jie Xin, Pengpeng Zhao, Jian Wu, Xuefeng Xian, Chunhua Li, and Zhiming Cui. 2014. An active learning approach with uncertainty, representativeness, and diversity. *The Scientific World Journal*, 2014.

Nathalie Japkowicz. 2000. The class imbalance problem: Significance and strategies. In *Proceedings of the Intl Conf. on Artificial Intelligence*.

Kimmo Kettunen, Eetu Mäkelä, Teemu Ruokolainen, Juha Kuokkala, and Laura Löfberg. 2017. Old content and modern tools–Searching named entities in a Finnish OCRed historical newspaper collection 1771-1910. *DHQ: Digital Humanities Quarterly*, 11(3).

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Lihong Li, Michael L Littman, and Thomas J Walsh. 2008. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 568–575. ACM.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2012–2022.

David Lowell, Zachary C Lipton, and Byron C Wallace. 2018. How transferable are the datasets collected by active learners? *arXiv preprint arXiv:1807.04801*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2536–2545.

Anke de Naegel. 2015. The Herodotos project (OSU-UGent): Studies in ancient ethnography. Barbarians in Strabos geography (Isseans–Zygi). with a case-study: the Britons. Master's thesis, Gent Universiteit.

Toan Q Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 296–301.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields.

Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning, 2001*, pages 441–448.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 589. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256.

Rainer Simon, Elton Barker, Leif Isaksen, and Pau de Soto Cañamares. 2017. Linked data annotation without the pointy brackets: Introducing Recogito 2. *Journal of Map & Geography Libraries*, 13(1):111–132.

Rainer Simon, Leif Isaksen, ETE Barker, and Pau de Soto Cañamares. 2016. The Pleiades gazetteer and the Pelagios project. In *Placing Names: Enriching and Integrating Gazetteers*, pages 97–109. Indiana University Press.

David A Smith and Gregory Crane. 2001. Disambiguating geographic names in a historical digital library. In *International Conference on Theory and Practice of Digital Libraries*, pages 127–136. Springer.

David A Smith, Jeffrey A Rydberg-Cox, and Gregory R Crane. 2000. The Perseus project: A digital library for the humanities. *Literary and Linguistic Computing*, 15(1):15–25.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Rachele Sprugnoli. 2018. Arretium or Arezzo? a neural approach to the identification of place names in historical texts. *http://www. ceur-ws. org*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Seth Van Hooland, Max De Wilde, Ruben Verborgh, Thomas Steiner, and Rik Van de Walle. 2013. Exploring entity recognition and disambiguation for cultural heritage collections. *Digital Scholarship in the Humanities*, 30(2):262–279.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018a. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017.

Zengmao Wang, Xi Fang, Xinyao Tang, and Chen Wu. 2018b. Multi-class active learning by integrating uncertainty and diversity. *IEEE Access*, 6:22794–22803.

David Joseph Wrisley. 2017. Locating medieval French, or why we collect and visualize the geographic information of texts. *Speculum*, 92(S1):S145–S169.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1137–1144. Association for Computational Linguistics.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. 2003. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, pages 58–65. ICLM.

Doc2hash: Learning Discrete Latent Variables for Document Retrieval

Yifei Zhang
Vector Lab
JD Finance
zhangyifei11@jd.com

Hao Zhu
Vector Lab
JD Finance
zhuhao6@jd.com

Abstract

Learning to hash via generative models has become a powerful paradigm for fast similarity search in documents retrieval. To get binary representation (i.e., hash codes), the discrete distribution prior (i.e., Bernoulli Distribution) is applied to train the variational autoencoder (VAE). However, the discrete stochastic layer is usually incompatible with the backpropagation in the training stage and thus causes a gradient flow problem. In this paper, we propose a method, Doc2hash, that solves the gradient flow problem of the discrete stochastic layer by using continuous relaxation on priors, and trains the generative model in an end-to-end manner to generate hash codes. In qualitative and quantitative experiments, we show the proposed model outperforms other state of the art methods.

1 Introduction

A popular theme for deep learning is that of representation learning, whose goal is to use existing data to learn a compact and meaningful representation when building classifiers or other predictors. Learning continuous representation has been achieving a great success in various NLP tasks, including text classification, word and sentence representation (Mikolov et al., 2013; Le and Mikolov, 2014), yet learning discrete representation is potentially more suitable for tasks we are interested in such as learning to hash.

Hashing serves as a fast solution for similarity search also called approximate nearest neighbor search. In document retrieval, semantic hashing is the strategy that turn the document into binary codes (hash codes) which capture semantic information. One can use a query (a document) to retrieve similar documents by calculating the hamming distances between their hash codes. Given the fast calculation process and the efficient storage property (one modern PC can execute millions of hamming distance computations in just a few milliseconds). This semantic hashing strategy is very attractive.

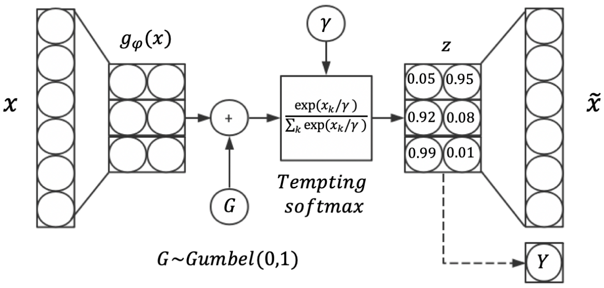


Figure 1: The end-to-end semantic hashing framework: Training the cycle $x \rightarrow z \rightarrow x$ to obtain the latent variables z. The one hot representation of discrete latent variables z has been relaxed to $\{(0.05, 0.95), (0.92, 0.08), (0.99, 0.01)\}$ by the tempting softmax. Hash codes $z_h = \{1, 0, 0\}$ is obtained via taking the $\arg\max$ of z. Y is the label/tag used in the supervised hashing

Inspired by the recent success of modeling latent variables via generative models in solving various NLP problems (van den Oord et al., 2017; Semeniuta et al., 2017; Bowman et al., 2015), Some approaches obtain binary codes of documents from a generative perspective via parameterizing models using neural networks and stochastic optimization using gradient-based techniques: (Chaidaroon and Fang, 2017) designed a two-stage training procedures to generate hash codes with variational autoencoder: (i) it first infers continuous representations of text through VAE with isotropic Gaussian distribution prior (ii) Obtain hash codes via binarizing the continuous representation of texts. Since the model parameters are not learned in an end-to-end manner, the two-stage training strategy may result in a suboptimal local optima. (Shen et al., 2018) replaced the Gaussian distribution prior with Bernoulli distribution prior so that the stochastic layer of the

Proceedings of NAACL-HLT 2019, pages 2235–2240
Minneapolis, Minnesota, June 2 - June 7, 2019.

VAE can directly produce binary codes in the latent space and train hash codes in an end-to-end manner. Unfortunately, the Bernoulli stochastic layer is non-differentiable. Although the Straight-Through (ST) estimator is adapted to propagate gradients where it skips the gradient of the stochastic layer during backpropagation (Bengio et al., 2013). The ST estimator is still a biased estimator which introduces high-variance biased gradients of objectives during the training.

In this paper, we propose a generative model with a categorical distribution prior for semantic hashing which learns binary codes of documents in an end-to-end manner. To train the generative model, instead of using the ST estimator, we use the Gumbel-Softmax trick to overcome the inability by applying the re-parameterization trick to discrete latent variables. There are two advantages: 1) a nice parameterization for a discrete (or categorical) distribution is given in terms of the Gumbel distribution (the Gumbel trick); and 2) although the corresponding function is non-continuous, it can be made continuous by applying using a continuous approximation that depends on a temperature parameter. Therefore, it produces low-variance biased gradients of the stochastic layer in the backpropagation. Our experiment shows our model achieves the state of the art performance on three standard datasets for semantic hashing.

2 Methodology

2.1 Discrete NVI Framework For Semantic Hashing

In this section, we introduce the neural variational inference (NVI) framework (Shen et al., 2018) with discrete latent variables for semantic hashing in detail. First, a generative model whose encoding distribution and decoding distribution are defined as $p(z|x), p(x|z)$. We approximate these two distributions by defining approximations $q_\phi(z|x)$, $p_\theta(x|z)$ and model them as the inference (encoding) network $g_\phi(x)$ and the generative network (decoding) $g_\theta(z)$. $x \in R^V$ is the bag of word representation of a document where V is vocabulary size. Specially, the weight schema of bag of words can be binary, TF, TFIDF. The inference network $g_\phi(x)$ produces latent variables z from x approximating the true posterior distribution $p(z|x)$ as $q_\phi(z|x)$. Then, the generative network $g_\theta(z)$ maps latent variable z back to x by approximating $p(z|x)$ as $p_\theta(x|z)$. We model the conditional probability over the word w_i as softmax:

$$p(x_i = w_i|z) = \frac{\exp(w_i g_\theta(z))}{\sum_V \exp(w_i g_\theta(z))} \quad (1)$$

where $w_i \in \{0,1\}^V$ is represented as an one-hot vector indicating the i-th word in the total vocabulary and the likelihood $p_\theta(x|z) = \prod_i p(x_i = w_i|z)$. Training the generative model that synthesizes the observed data x, we can obtain the meaningful latent variables z.

To learn discrete latent variables under the NVI framework for semantic hashing, we cast z to be the discrete random variables and assume a multivariate categorical prior on z: $p(z) \sim \mathbf{Cat}(\pi) = \prod_i^l I(z = k)\pi_{ik}$, where π_{ik} is the k-th class probability on i-th component of parameters π. In this way, the posterior distribution approximated by the encoding network is constrained in the form of $q_\phi(z|x) = \mathbf{Cat}(h)$, where $h = g_\phi(x)$ is the output of the encoding network.

2.2 Variational Low Bound of Discrete Latent Variable Training

To train the parameters θ, ϕ of the encoding network $q_\theta(z|x)$ and the decoding network $p_\phi(x|z)$, we maximize the variational low bound as same as in the VAE framework (Kingma and Welling, 2013):

$$L_{ELBO} = E_{z\sim q_\phi(z|x)}[\log \frac{q_\theta(x|z)p(z)}{q_\phi(z|x)}] \quad (2)$$

$$= E_{z\sim q_\phi(z|x)}[\log q_\theta(x|z)] - KL(q_\phi(z|x)||p(z)) \quad (3)$$

the first term is the reconstruction loss, which encourages the decoder network to learn the mapping from latent variables z to the original document x. The second term is Kullback-Leibler (KL) divergence $KL(q_\phi(z|x)||p(z))]$, which encourages the posterior distribution $q_\phi(z|x)$ to be close to the multivariate categorical prior $p(z)$. we can write the KL term in the following form:

$$KL(q_\phi(z|x)||p(z)) = \sum_i \sum_k g_\phi^{ik}(x) \log \frac{g_\phi^{ik}(x)}{\pi_{ik}} \quad (4)$$

2.3 Gradients of Discrete Stochastic Layer

To evaluate $E_{z\sim q_\phi(z|x)}$, the reparameterization trick is often employed to make the sampling process $z \sim q_\phi(z|x)$ compatible with backpropagation (Kingma and Welling, 2013). But sampling from a discrete distribution using reparameterization tricks still has problem during backpropagation. Due to the sampling from the stochastic layer usually incorporate hard threshold operators such as $\mathbf{sign(.)}$ and $\mathbf{arg\,max(.)}$ (Jang et al., 2016; Maddison et al., 2016), the stochastic layer becomes non-differentiable. In general, the solution to this issue is using the Straight-Through (ST) estimator where a biased path derivative estimator can be utilized even when z is not reparameterizable as shown in Figure 2. For example, NASH (Shen et al., 2018) passes the gradients through the Bernoulli stochastic layer using ST estimator. Specifically, they do:

$$\frac{dL}{d\phi} = \frac{dL}{dz}\frac{dz}{dg_\phi(x)}\frac{dg_\phi(x)}{d\phi} \approx \frac{dL}{dz}\frac{dg_\phi(x)}{d\phi} \quad (5)$$

where the ST estimator lets the gradients of the stochastic layer $\frac{dz}{dg_\phi(x)} \approx 1$. However, the ST estimator is backpropagating with respect to the sample-independent. It may cause discrepancies between the forward and backward pass, and thus lead to higher variance (Jang et al., 2016).

2.4 Continuous Relaxation of Discrete Stochastic Layer

As mention above, we want to learn discrete Latent variables (hash codes) by maximizing the variational low bound assuming a multivariate categorical prior. Supposed the output of the encoding network, $h = g_\phi(x)$, $h \in R^{l\times 2}$, represents the parameters of the multivariate categorical posterior $q(z|x)$. We can use reparameterization trick to draw samples of z from the categorical posterior using Gumbel distribution (Gumbel, 1954; Jang et al., 2016):

$$z_i = \arg\max_k (G_k + logh_i^k), k \in \{0, 1\} \quad (6)$$

G_1, G_2 is drawn i.i.d. from Gumbel (0, 1) = $\log(-\log(U(0, 1))$ where U is the uniform distribution. We further represent z_i as one-hot vector with 2 dimensions:

$$z_i = \text{OneHot}[\arg\max_k (G_k + logh_i^k)], k \in \{0, 1\} \quad (7)$$

discrete latent variables z can be seen as concatenation of l one-hot vectors.

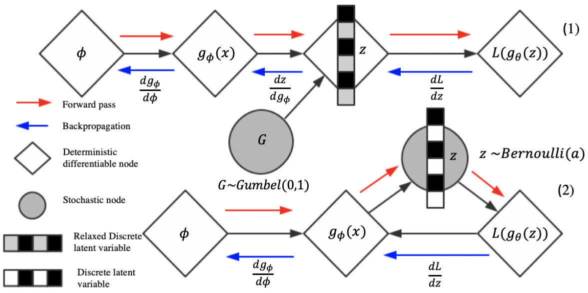


Figure 2: Gradients estimation in stochastic computation graphs (1) Gumbel-Softmax trick (2) The Straight-Through estimator, used for Bernoulli discrete variables, skips stochastic node by approximating $\frac{dz}{dg_\phi} \approx 1$

In the forward pass, we sample z from the stochastic discrete layer via Eq. 7. However, in backpropagation, we cannot pass gradient through the hard threshold operator, $\arg\max$, since it is non-differentiable. The derivative of $\arg\max$ is 0 everywhere except the boundary of state change, where it is undefined. To end this, instead of employing ST estimator to skip gradients, we use the tempting softmax as a continuous relaxation of the $\arg\max$ computation:

$$z_i = \frac{\exp((G_k + \log h_i^k)/\gamma)}{\sum_k \exp((G_k + \log h_i^k)/\gamma)}, k \in \{0, 1\} \quad (8)$$

This relaxation approximates the exactly discrete $\arg\max$ computation as temperature parameter $\gamma \rightarrow 0$ yet keeps the relative order of $(G_k + \log h_i^k)$. similar relaxation techniques have been introduced as the Gumbel-Softmax trick in (Jang et al., 2016; Maddison et al., 2016). Since Eq. 8 is differentiable, we can train the generative model in an end-to-end way via stochastic gradient descent to get the discrete latent variables (hash codes). The whole framework is summarized in the Figure 1 and the difference against the ST estimator is exemplified in the Figure 2.

2.5 Supervised Hashing

We extend our method to the supervised setting, where the mapping from latent variables z to the label y is learned, here parameterized by a two-layer multilayer perceptron (MLP) followed by a fully-connected softmax layer. To balance maximizing the variational lower bound and minimizing the discriminative loss, the following joint training objective is employed:

$$L = -L_{ELBO}(\theta, \phi, \boldsymbol{x}) + \alpha L_{dis}(\eta, \boldsymbol{z}, y) \quad (9)$$

where η refers to parameters of the MLP classifier and α controls the relative weight between the variational lower bound (L_{ELBO}) and the discriminative loss (L_{dis}). We assume a general multi-label classification setting where each document could have multiple labels/tags. $P(y_j|f(\boldsymbol{z};\eta))$ can be modeled by the logistic function defined as:

$$P(y_j|f(\boldsymbol{z},\eta)) = \frac{1}{1+\exp(-y_j^T \eta_j \boldsymbol{z})} \tag{10}$$

3 Experiment

3.1 Baseline and Setting

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
LSH	0.4388	0.4393	0.4514	0.4553	0.4773
S-RBM	0.4846	0.5108	0.5166	0.5190	0.5137
SpH	0.5807	0.6055	0.6281	0.6143	0.5891
STH	0.3723	0.3947	0.4105	0.4181	0.4123
VDSH	0.4330	0.6853	0.7108	0.4410	0.5847
NASH-DN	0.6358	0.6956	0.7327	0.7010	0.6325
Doc2hash	**0.6965**	**0.7224**	**0.7473**	**0.7532**	**0.7595**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.6608	0.6842	0.7047	0.7175	0.7243
SHTTM	0.6299	0.6571	0.6485	0.6893	0.6474
VDSH-SP	0.7498	0.7798	0.7891	0.7888	0.7970
NASH-DN-S	0.7438	0.7946	0.7987	0.801	0.8139
Doc2hash-S	**0.8140**	**0.8472**	**0.8490**	**0.8492**	**0.8439**

Table 1: Precision of the top 100 retrieved documents on TMC dataset.

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
LSH	0.2802	0.3215	0.3862	0.4667	0.5194
S-RBM	0.5113	0.5740	0.6154	0.6177	0.6452
SpH	0.6080	0.6340	0.6513	0.6290	0.6045
STH	0.6616	0.7351	0.7554	0.7350	0.6986
VDSH	0.6859	0.7165	0.7753	0.7456	0.7318
NASH-DN	0.7470	0.8013	0.8418	0.8297	0.7924
Doc2hash	**0.7543**	**0.8102**	**0.8487**	**0.8361**	**0.8344**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.7840	0.8376	0.8480	0.8537	0.8620
SHTTM	0.7992	0.8520	0.8323	0.8271	.8150
VDSH-SP	0.8890	0.9326	0.9283	0.9286	0.9395
NASH-DN-S	**0.9214**	0.9327	0.9455	0.9589	0.9502
Doc2hash-S	0.9134	**0.9338**	**0.9557**	**0.9602**	**0.9598**

Table 2: Precision of the top 100 retrieved documents on Reuters dataset.

We evaluate the proposed method named Doc2hash in both supervised and unsupervised setting for semantic hashing. For unsupervised task, these baselines are selected: Locality Sensitive Hashing (LSH) (Datar et al., 2004), Stack Restricted Boltzmann Machines (Salakhutdinov and Hinton, 2009), Spectral Hashing (Weiss et al., 2009), Self taught Hashing (STH) (Zhang et al., 2010), Variational Deep Semantic Hashing (VDSH) (Chaidaroon and Fang, 2017), and the best variant of Neural Architecture for Generative Semantic Hashing (Shen et al., 2018) (NASH-DN). For the supervised task, we also compare the proposed method with kinds of baselines: Supervised Hashing with Kernels (KSH) (Liu et al., 2012), Semantic Hashing using Tags and Topic Modeling (SHTTM) (Wang et al., 2013), Supervised VDSH (VDSH-SP) and Supervised NASH (NASH-DN-S).

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
LSH	0.4180	0.4352	0.4716	0.5214	0.5877
S-RBM	0.5106	0.5743	0.6130	0.6463	0.6531
SpH	0.5093	0.7121	0.7475	0.7559	0.7423
STH	0.3975	0.4898	0.5592	0.5945	0.5946
VDSH	0.7976	0.7944	0.8481	0.8951	0.8444
Doc2hash-S	**0.8495**	**0.8858**	**0.9001**	**0.9123**	**0.9167**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.9126	0.9146	0.9221	0.9333	0.9350
SHTTM	0.8820	0.9038	0.9258	0.9459	0.9447
VDSH-SP	0.9666	0.9757	0.9788	**0.9805**	0.9794
Doc2hash-S	**0.9720**	**0.98001**	**0.9810**	0.9802	**0.9797**

Table 3: Precision of the top 100 retrieved documents on RCV1 dataset.

We follow the same experimental protocol and setting used in (Shen et al., 2018; Chaidaroon and Fang, 2017). Three standard public datasets of documents are chosen for training and evaluation: Reuters (Lewis, 1997), TMC (Oza, 2010), and RCV1 (Lewis et al., 2004). We evaluate the proposed model by calculating the precision of top 100 retrieved documents based on hamming distance of their hash codes. The final precision score is then averaged over all test documents. To make comparable results with previous works, we experiment with latent variables size of 8, 32, 64 and 128.

3.2 Semantic Hashing Evaluation

In this section, we evaluate Doc2hash over various number of bits on the three datasets. In the unsupervised hashing, Doc2hash outperforms NASH-DN, the current state of the art model for semantic hashing, and other methods. The Table 1 shows the result of TMC dataset. Doc2hash shows its ability to assign the similar data (with the same label) to the hash codes of which hamming distances are small. The same trend and superiority of Doc2hash are also observed in both Reuters and RCV1 datasets as shown in Tables 2 and 3. Note that we do not compare with NASH-DN in RCV1 because NASH doesn't report any results on RCV1 dataset and doesn't release any code to reproduce their result. We compare the reminding methods with Doc2hash. Table 3 shows our approach improves the precision with significant margin compared with other models. We note that the retrieval results tend to drop when we set the length of hash codes to be 64 or larger, which also happens for some baselines. It is probably because of over-fitting. Compared with other methods, the proposed method is robust when the codes length increase. Our approach performs better than other methods in 64 and 128 bits setting, suggesting that Doc2hash can effectively generate hash codes to documents even with limited training data.

We also evaluate Doc2hash in the supervised hashing setting with the same datasets. We make use of the label/tag information during training. As shown in Tables 1, 2 and 3, Doc2hash yields better results than the other baseline models in different bits length.

3.3 Quantitative Analysis

In this section, we explore the reason why the proposed method outperforms the counterpart based on the ST estimator. We try to analyze the distribution of latent variables that trained with ST estimator (NASH) and Gumbel softmax relaxation (our method) respectively to quantitatively estimate the coding efficiency. To obtain efficient hash codes in the unsupervised setting, an expected pattern is that we get the same amount of 1 as well as 0. Because according to information theory, such a uniform distribution produces the maximum entropy in Bernoulli distribution. Motivated by this idea, we make a comparison between NASH and Doc2hash in the conditional probabilistic distribution of latent variables (i.e., $p(z|x)$). As we can see in Figure 4(a), the distribution of $p(z|x)$ in NASH is significantly asymmetric while the proposed method generates a well-balanced distribution (as Shown in Figure 4(b)). This comparison demonstrates that our approach is more close to the expected target in the probabilistic distribution of $p(z|x)$. Furthermore, we calculate the proportion of 1 in both methods. The proportion of NASH is 40%, while our method is 49%. Thus, it shows our approach is able to produce more efficient hash codes by using Gumbel softmax relaxation because it is very close to the up-bound of information theory for Bernoulli distribution. We demonstrate this point more clearly by showing the distribution of $g_\phi(x)$ in NASH. It is asymmetric because the peak value tilts to the left side. And quantization based on thresholding around the zero point implies that the hash codes produced by NASH contain more number of 0 than the number of 1.

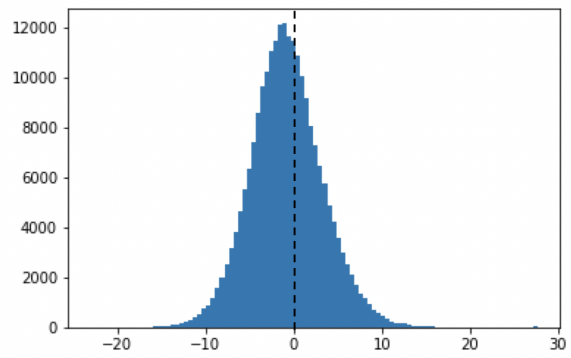
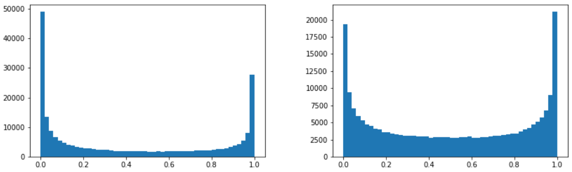


Figure 3: The distribution of $g_\phi(x)$ in NASH



(a) (b)

Figure 4: the distribution of $p(z|x)$: (a) is the distribution of NASH. (b) is the distribution of Doc2hash

4 Conclusion

In this paper, we present a generative model with discrete latent variables to learn binary representation for semantic hashing. We show that compared with the Straight-Through estimator, the Gumbel-Softmax relaxation provides a better solution to learn hash codes and thus gains the state of the art performance in three different datasets. We also explore the reason why the proposed method produces more efficient hash codes compared with the counterpart based on the ST estimator.

Acknowledgements

We sincerely thank anonymous reviewers for your constructive suggestions to improve the quality of this paper.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM.

Emil Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

David D. Lewis. 1997. Reuters-21578 text categorization collection data set.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Nikunj Oza. 2010. Siam 2007 text mining competition dataset.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *IJAR*, 50(7):969–978.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Ricardo Henao, and Lawrence Carin. 2018. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2041–2050. Association for Computational Linguistics.

Qifan Wang, Dan Zhang, and Luo Si. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 213–222. ACM.

Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760.

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM.

Evaluating Text GANs as Language Models

Guy Tevet^{*,1,2} **Gavriel Habib**^{*,1,2} **Vered Shwartz**^{3} **Jonathan Berant**^{1,4}

^{1}School of Computer Science, Tel-Aviv University
^{2}Department of Electrical Engineering, Tel-Aviv University
^{3}Computer Science Department, Bar-Ilan University
^{4}Allen Institute for Artificial Intelligence
`{guytevet@mail,gavrielhabib@mail,joberant@cs}.tau.ac.il`
`vered1986@gmail.com`

Abstract

Generative Adversarial Networks (GANs) are a promising approach for text generation that, unlike traditional language models (LM), does not suffer from the problem of "exposure bias". However, A major hurdle for understanding the potential of GANs for text generation is the lack of a clear evaluation metric. In this work, we propose to approximate the distribution of text generated by a GAN, which permits evaluating them with traditional probability-based LM metrics. We apply our approximation procedure on several GAN-based models and show that they currently perform substantially worse than state-of-the-art LMs. Our evaluation procedure promotes better understanding of the relation between GANs and LMs, and can accelerate progress in GAN-based text generation.

1 Introduction

Neural networks have revolutionized the field of text generation, in machine translation (Sutskever et al., 2014; Neubig, 2017; Luong et al., 2015; Chen et al., 2018), summarization (See et al., 2017), image captioning (You et al., 2016) and many other applications (Goldberg, 2017).

Traditionally, text generation models are trained by going over a gold sequence of symbols (characters or words) from left-to-right, and maximizing the probability of the next symbol given the history, namely, a language modeling (LM) objective. A commonly discussed drawback of such LM-based text generation is *exposure bias* (Ranzato et al., 2015): during training, the model predicts the next token conditioned on the ground truth history, while at test time prediction is based on predicted tokens, causing a train-test mismatch. Models trained in this manner often struggle to overcome previous prediction errors.

Generative Adversarial Networks (Goodfellow et al., 2014) offer a solution for *exposure bias*. Originally introduced for images, GANs leverage a discriminator, which is trained to discriminate between real images and generated images via an adversarial loss. In such a framework, the generator is not directly exposed to the ground truth data, but instead learns to imitate it using global feedback from the discriminator. This has led to several attempts to use GANs for text generation, with a generator using either a recurrent neural network (RNN) (Yu et al., 2017; Guo et al., 2017; Press et al., 2017; Rajeswar et al., 2017), or a Convolutional Neural Network (CNN) (Gulrajani et al., 2017; Rajeswar et al., 2017).

However, evaluating GANs is more difficult than evaluating LMs. While in language modeling, evaluation is based on the log-probability of a model on held-out text, this cannot be straightforwardly extended to GAN-based text generation, because the generator outputs discrete tokens, rather than a probability distribution. Currently, there is no single evaluation metric for GAN-based text generation, and existing metrics that are based on n-gram overlap are known to lack robustness and have low correlation with semantic coherence (Semeniuta et al., 2018).

In this paper, we propose a method for evaluating GANs with standard probability-based evaluation metrics. We show that the expected prediction of a GAN generator can be viewed as a LM, and suggest a simple Monte-Carlo method for approximating it. The approximated probability distribution can then be evaluated with standard LM metrics such as perplexity or Bits Per Character (BPC).

To empirically establish our claim, we implement our evaluation on several RNN-based GANs: (Press et al., 2017; Yu et al., 2017; Guo et al., 2017). We find that all models have substantially lower BPC compared to state-of-the-art LMs. By directly comparing to LMs, we put in perspective the current performance of RNN-based GANs for

---

[*] The authors contributed equally

Proceedings of NAACL-HLT 2019, pages 2241–2247
Minneapolis, Minnesota, June 2 - June 7, 2019.

text generation. Our results are also in line with recent concurrent work by Caccia et al. (2018), who reached a similar conclusion by comparing the performance of textual GANs to that of LMs using metrics suggested for GAN evaluation.

Our code is available at: `http://github.com/GuyTevet/SeqGAN-eval` and `http://github.com/GuyTevet/rnn-gan-eval`.

2 Background

Following the success of GANs in image generation, several works applied the same idea to texts using convolutional neural networks (Gulrajani et al., 2017; Rajeswar et al., 2017), and later using RNNs (Press et al., 2017; Yu et al., 2017). RNNs enable generating variable-length sequences, conditioning each token on the tokens generated in previous time steps. We leverage this characteristic in our approximation model (§4.1).

A main challenge in applying GANs for text is that generating discrete symbols is a non-differentiable operation. One solution is to perform a continuous relaxation of the GAN output, which leads to generators that emit a nearly discrete continuous distribution (Press et al., 2017). This keeps the model differentiable and enables end-to-end training through the discriminator. Alternatively, SeqGAN (Yu et al., 2017) and LeakGAN (Guo et al., 2017) used policy gradient methods to overcome the differentiablity requirement. We apply our approximation to both model types.

3 Evaluating GANs and LMs

LM Evaluation. Text generation from LMs is commonly evaluated using probabilistic metrics. Specifically, given a test sequence of symbols $(t_1, \dots, t_n)$, and a LM q, the average cross-entropy over the entire test set is computed: $ACE = -\frac{1}{n}\sum_{i=1}^{n}\log_2 q(t_i \mid t_1 ... t_{i-1})$. For word-based models, the standard metric is perplexity: $PP = 2^{ACE}$, while for character-based models it is $BPC = ACE$ directly.

Intrinsic improvement in perplexity does not guarantee an improvement in an extrinsic downstream task that uses a language model. However, perplexity often correlates with extrinsic measures (Jurafsky and Martin, 2018), and is the de-facto metric for evaluating the quality of language models today.

GAN-based Text Generation Evaluation. By definition, a text GAN outputs a discrete sequence of symbols rather than a probability distribution. As a result, LM metrics cannot be applied to evaluate the generated text. Consequently, other metrics have been proposed:

- N-gram overlap: (Yu et al., 2017; Press et al., 2017): Inspired by BLEU (Papineni et al., 2002), this measures whether n-grams generated by the model appear in a held-out corpus. A major drawback is that this metric favors conservative models that always generate very common text (e.g., *"it is"*). To mitigate this, self-BLEU has been proposed (Lu et al., 2018) as an additional metric, where overlap is measured between two independently sampled texts from the model.
- LM score: The probability of generated text according to a pre-trained LM. This has the same problem of favoring conservative models.
- Zhao et al. (2017) suggested an indirect score by training a LM on GAN-generated text, and evaluating it using perplexity. The drawback in this setting is the coupling of the performance of the GAN with that of the proxy LM.
- Heusel et al. (2017) used Frechet InferSent Distance (FID) to compute the distance between distributions of features extracted from real and generated samples. However, this approach relies on a problematic assumption that features are normally distributed.
- Rajeswar et al. (2017) used a context-free grammar (CFG) to generate a reference corpus, and evaluated the model by the likelihood the CFG assigns to generated samples. However, simple CFGs do not fully capture the complexity of natural language.
- To overcome the drawbacks of each individual method, Semeniuta et al. (2018) proposed a unified measure based on multiple evaluation metrics (N-grams, BLEU variations, FID, LM score variations and human evaluation). Specifically, they argue that the different measures capture different desired properties of LMs, e.g., quality vs. diversity.
- Following Semeniuta et al. (2018), and in parallel to this work, Caccia et al. (2018) proposed a temperature sweep method that trades-off quality for diversity using a single parameter. Similar to our findings, they concluded that GANs perform worse than LMs on this metric.

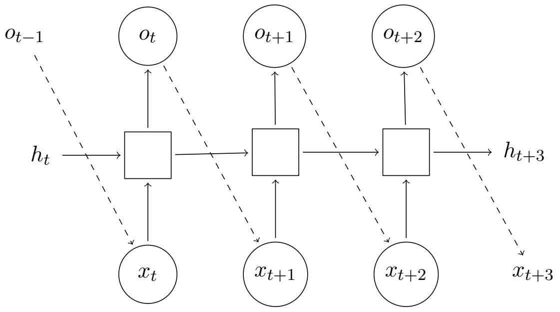


Figure 1: Generator recurrent connections. $\{h_t\}$ is the internal state sequence and $\{o_t\}$ is the generator prediction sequence (one-hot). During inference, the outputs $\{o_t\}$ are fed back as the input for the next time step (dashed lines). During LM approximation, the input $\{x_t\}$ is a sequence of one-hot vectors from the test set.

Algorithm 1 LM Evaluation of RNN-based GANs

```
Input: $G_t(\cdot)$: the generator function at time step $t$
       $(x_0, ..., x_t)$: previous gold tokens
       $x_{t+1}$: the gold next token (as ground truth)
       $f(\cdot,\cdot)$: a LM evaluation metric
       $N$: number of samples
for $n \leftarrow 1$ to $N$ do
    $g_{t,n} \longleftarrow$ sample from $G_t(x_0...x_t)$
$\tilde{G}_{t,N} = \frac{1}{N}\Sigma_{n=1}^{N} g_{t,n}$
return $f(\tilde{G}_{t,N}, x_{t+1})$
```

Overall, current evaluation methods cannot fully capture the performance of GAN-based text generation models. While reporting various scores as proposed by Semeniuta et al. (2018) is possible, it is preferable to have a single measure of progress when comparing different text generation models.

4 Proposed Method

We propose a method for approximating a distribution over tokens from a GAN, and then evaluate the model with standard LM metrics. We will describe our approach given an RNN-based LM, which is the most commonly-used architecture, but the approximation can be applied to other auto-regressive models (Vaswani et al., 2017).

4.1 Language Model Approximation

The inputs to an RNN at time step t, are the state vector h_t and the current input token x_t. The output token (one-hot) is denoted by o_t. In RNN-based GANs, the previous output token is used at inference time as the input x_t (Yu et al., 2017; Guo et al., 2017; Press et al., 2017; Rajeswar et al., 2017). In contrast, when evaluating with BPC or perplexity, the gold token x_t is given as input. Hence, LM-based evaluation neutralizes the problem of exposure bias addressed by GANs. Nevertheless, this allows us to compare the quality of text produced by GANs and LMs on an equal footing. Figure 1 illustrates the difference between inference time and during LM approximation.

We can therefore define the generator function at time step t as a function of the initial state h_0 and the past generated tokens $(x_0 \dots x_t)$, which we denote as $o_t = G_t(h_0, x_0...x_t)$ (x_0 is a start token). Given a past sequence $(x_0 \dots x_t)$, G_t is a stochastic function: the stochasticity of G_t can be gained either by using a noise vector as the initial state h_0 (Press et al., 2017), or by sampling from the GAN's internal distribution over possible output tokens (Yu et al., 2017; Guo et al., 2017). Since h_0 is constant or a noise vector that makes G_t stochastic, we can omit it to get $G_t(x_0 \dots x_t)$. In such a setup, the expected value $\mathbb{E}[G_t(x_0 \dots x_t)]$ is a distribution q over the next vocabulary token a_t:

$$q(a_t \mid a_0 \dots a_{t-1}) = \{\mathbb{E}[G_t(x_0 \dots x_t)]\}_{a_t}$$

To empirically approximate q, we can sample from it N i.i.d samples, and compute an approximation $\tilde{G}_{t,N} = \frac{1}{N}\Sigma_{n=1}^{N} g_{t,n}$, where $g_{t,n}$ is one sample from $G_t(x_0...x_t)$. Then, according to the strong law of large numbers:

$$\mathbb{E}[G_t(x_0 \dots x_t)] = \lim_{N \to \infty} \tilde{G}_{t,N} \tag{1}$$

Given this approximate LM distribution, we can evaluate a GAN using perplexity or BPC. We summarize the evaluation procedure in Algorithm 1.[1]

4.2 Approximation Bound

We provide a theoretical bound for choosing a number of samples N that results in a good approximation of $\tilde{G}_{t,N}$ to $\mathbb{E}[G_t]$.

Perplexity and BPC rely on the log-probability of the ground truth token. Since the ground truth token is unknown, we conservatively define the bad event B in which there exists $v \in V$ such that $|\{E[G_t]\}_v - \{\tilde{G}_{t,N}\}_v| > \gamma$, where V is the vocabulary. We can then bound the probability of B by some ϵ. We define the following notations:

1. The probability of a token a_t to be v is $p_v \overset{\Delta}{=} q(a_t = v | a_0 \dots a_{t-1}) = \{\mathbb{E}[G_t(x_0 \dots x_t)]\}_v$.
2. $\chi_{v,n} \overset{\Delta}{=} \{g_{t,n}\}_v$ is a random variable representing the binary value of the v'th index of

---

[1] Our evaluation algorithm is linear in the length of the test set and in the number of samples N.

$g_{t,n}$ which is a single sample of G_t. Note that the average of $\chi_{v,n}$ over N samples is $X_v \triangleq \frac{1}{N}\sum_{n=1}^{N}\chi_{v,n} = \left\{\frac{1}{N}\sum_{n=1}^{N} g_{t,n}\right\}_v = \{\tilde{G}_{t,N}\}_v$.

Using the above notation, we can re-define the probability of the bad event B with respect to the individual coordinates in the vectors:

$$\begin{aligned} Pr(B) &= Pr\left(\|\mathbb{E}[G_t] - \tilde{G}_{t,N}\|_\infty > \gamma\right) \\ &= Pr\left(\bigcup_{v\in V} |p_v - X_v| > \gamma\right) \overset{!}{<} \epsilon \end{aligned} \tag{2}$$

We note that $\chi_{v,n} \sim Bernoulli(p_v)$, and given that $\{\chi_{v,n}\}_{n=1}^{N}$ are i.i.d., we can apply the Chernoff-Hoeffding theorem (Chernoff et al., 1952; Hoeffding, 1963). According to the theorem, for every $v \in V$, $Pr(|X_v - p_v| > \gamma) < 2e^{-2N\gamma^2}$. Taking the union bound over V implies:

$$Pr(B) = Pr\left(\bigcup_{v\in V} |X_v - p_v| > \gamma\right) < 2|V|e^{-2N\gamma^2} < \epsilon \tag{3}$$

Hence, we get a lower bound on N:

$$N > \frac{1}{2\gamma^2}\ln\left(\frac{2|V|}{\epsilon}\right) \tag{4}$$

As a numerical example, choosing $\gamma = 10^{-3}$ and $\epsilon = 10^{-2}$, for a character-based LM over the text8 dataset, with $|V| = 27$, we get the bound: $N > 4.3 \cdot 10^6$. With the same γ and ϵ, a typical word-based LM with vocabulary size $|V| = 50,000$ would require $N > 8.1 \cdot 10^6$.

In practice, probability vectors of LMs tend to be sparse (Kim et al., 2016). Thus, we argue that we can use a much smaller N for a good approximation $\tilde{G}_{t,N}$. Since the sparsity of LMs is difficult to bound, as it differs between models, we suggest an empirical method for choosing N.

The approximation $\tilde{G}_{t,N}$ is a converging sequence, particularly over $\|\cdot\|_\infty$ (see Equation 1). Hence, we can empirically choose an N which satisfies $\|\tilde{G}_{t,N-\alpha} - \tilde{G}_{t,N}\|_\infty < \gamma', \quad \alpha \in \mathbb{N}$. In Section 5 we empirically measure $\|\tilde{G}_{t,N-\alpha} - \tilde{G}_{t,N}\|_\infty$ as a function of N to choose N. We choose a global N for a model, rather than for every t, by averaging over a subset of the evaluation set.

5 Evaluation

5.1 Models

We focus on character-based GANs as a test-case for our method. We evaluate two RNN-based GANs with different characteristics. As opposed to the original GAN model (Goodfellow et al., 2014), in which the generator is initialized with random noise, the GANs we evaluated both leverage gold standard text to initialize the generator, as detailed below.

Recurrent GAN (Press et al., 2017) is a continuous RNN-based generator which minimizes the improved WGAN loss (Gulrajani et al., 2017). To guide the generator, during training it is initialized with the first $i-1$ characters from the ground truth, starting the prediction in the ith character. Stochasticity is obtained by feeding the generator with a noise vector z as a hidden state. At each time step, the input to the RNN generator is the output distribution of the previous step.

SeqGAN (Yu et al., 2017) is a discrete RNN-based generator. To guide the generator, it is pretrained as a LM on ground truth text. Stochasticity is obtained by sampling tokens from an internal distribution function over the vocabulary. To overcome differentiation problem, it is trained using a policy gradient objective (Sutton et al., 2000).

We also evaluated **LeakGAN** (Guo et al., 2017), another discrete RNN-based generator, but since it is similar to SeqGAN and performed worse, we omit it for brevity.

5.2 Evaluation Settings

To compare to prior work in LM, we follow the common setup and train on the text8 dataset.[2] The dataset is derived from Wikipedia, and includes 26 English characters plus spaces. We use the standard 90/5/5 split to train/validation/test. Finally, we measure performance with BPC.

We tuned hyper-parameters on the validation set, including sequence length to generate at test time (7 for Press et al. (2017), 1000 for Yu et al. (2017)). We chose the number of samples N empirically for each model, as described in Section 4.2. We set α to 10, and the boundary to $\gamma' = 10^{-3}$ as a good trade-off between accuracy and run-time. Figure 2 plots the approximate error $\|\tilde{G}_{t,N-\alpha} - \tilde{G}_{t,N}\|_\infty$ as a function of N. For both models, $N > 1600$ satisfies this condition (red line in Figure 2). To be safe, we used $N = 2000$.

5.3 Results

Table 1 shows model performance on the test set.

---

[2] `http://mattmahoney.net/dc/textdata`

Approach	Model	BPC	Approx. BPC
Language Models	mLSTM + dynamic eval (Krause et al., 2017)	1.19	
	Large mLSTM +emb +WN +VD (Krause et al., 2016)	1.27	
	Large RHN (Zilly et al., 2016)	1.27	
	LayerNorm HM-LSTM (Chung et al., 2016)	1.29	
	BN LSTM (Cooijmans et al., 2016)	1.36	
	Unregularised mLSTM (Krause et al., 2016)	1.40	
	SeqGAN - pre-trained LM (Yu et al., 2017)	1.85	1.95
GANs (LM Approximation)	SeqGAN - full adversarial training (Yu et al., 2017)	1.99	2.08
	Recurrent GAN without pre-training (Press et al., 2017)		3.31
	Uniform Distribution	4.75	

Table 1: Test set evaluation of different character-based models on the text8 dataset. State-of-the-art results are taken from `https://github.com/sebastianruder/NLP-progress/blob/master/language_modeling.md`. The uniform distribution is equivalent to guessing the next character out of $|V| = 27$ characters.

Model	Samples
SeqGAN Pre-trained LM	1. rics things where a weeks thered databignand jacob reving the imprisoners could become poveran brown 2. nine other set of of one eight one two by belarigho and singing signal theus to accept natural corp 3. ragems the downran maintain the lagar linear stream hegels p in five six f march one nine nine nine
SeqGAN Full adversarial training	1. four zero five two memaire in afulie war formally dream the living of the centuries to quickly can f 2. part of the pract the name in one nine seven were mustring of the airports tex works to eroses exten 3. eight four th jania lpa ore nine zero zero zero sport for tail concents englished a possible for po
Recurrent GAN	1. nteractice computer may became were the generally treat he were computer may became were the general 2. lnannnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnne and and and and and and and and and and and and a 3. perors as as seases as as as as as as as as as selected see see see see see see see see see see see

Table 2: Random samples of 100 characters generated by each model.

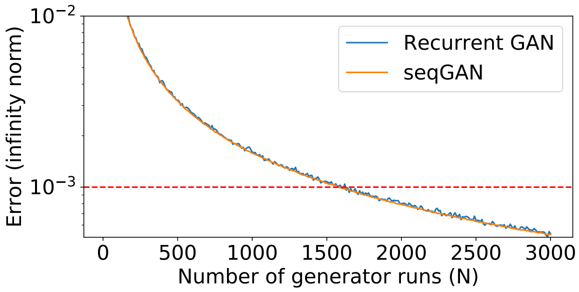


Figure 2: Approximate error $\|\tilde{G}_{t,N-\alpha} - \tilde{G}_{t,N}\|_\infty$ as a function of samples N. $\alpha = 10$, $\gamma' = 10^{-3}$.

Because SeqGAN models output a distribution over tokens at every time step, we can measure the true BPC and assess the quality of our approximation. Indeed, we observe that approximate BPC is only slightly higher than the true BPC.

GAN-based models perform worse than state-of-the-art LMs by a large margin. Moreover, in SeqGAN, the pre-trained LM performs better than the fully trained model with approximate BPC scores of 1.95 and 2.06, respectively, and the BPC deteriorates as adversarial training continues.

Finally, we note that generating sequences larger than 7 characters hurts the BPC of Press et al. (2017). It is difficult to assess the quality of generation with such short sequences.

In Table 2 we present a few randomly generated samples from each model. We indeed observe that adversarial training slightly reduces the quality of generated text for SeqGAN, and find that the quality of 100-character long sequences generated from Press et al. (2017) is low.

6 Conclusions

We propose an evaluation procedure for text GANs that is based on approximating the GAN output distribution and using standard LM metrics. We provide a bound for the number of samples required for the approximation and empirically show in practice as few as 2000 samples per time-step suffice. We evaluate character-based GAN models using our procedure, and show their performance is substantially lower than state-of-the-art LM. We hope our simple evaluation method leads to progress in GAN-based text generation by shedding light on the quality of such models.

Acknowledgements

We would like to thank Shimi Salant for his comments and suggestions. This research was partially supported by The Israel Science Foundation grant 942/16, the Blavatnik Computer Science Research Fund, and The Yandex Initiative for Machine Learning.

References

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. *arXiv preprint arXiv:1811.02549*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Herman Chernoff et al. 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. 2016. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*.

Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30.

Dan Jurafsky and James H Martin. 2018. *Speech and language processing: Third Edition*, volume 3. Pearson London.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2017. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. 2016. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*.

Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. 2017. Language generation with recurrent generative adversarial networks without pretraining. In *1st Workshop on Learning to Generate Natural Language at ICML 2017*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. 2018. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2017. Adversarially regularized autoencoders for generating discrete structures. *CoRR, abs/1706.04223*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

Latent Code and Text-based Generative Adversarial Networks for Soft-text Generation

Md. Akmal Haidar, Mehdi Rezagholizadeh, Alan Do-Omri and Ahmad Rashid
Huawei Noah's Ark Lab, Montreal Research Center, Canada
`md.akmal.haidar@huawei.com`, `mehdi.rezagholizadeh@huawei.com`
`alan.do.omri@huawei.com`, `ahmad.rashid@huawei.com`

Abstract

Text generation with generative adversarial networks (GANs) can be divided into the text-based and code-based categories according to the type of signals used for discrimination. In this work, we introduce a novel text-based approach called Soft-GAN to effectively exploit GAN setup for text generation. We demonstrate how autoencoders (AEs) can be used for providing a continuous representation of sentences, which we will refer to as soft-text. This soft representation will be used in GAN discrimination to synthesize similar soft-texts. We also propose hybrid latent code and text-based GAN (LATEXT-GAN) approaches with one or more discriminators, in which a combination of the latent code and the soft-text is used for GAN discriminations. We perform a number of subjective and objective experiments on two well-known datasets (SNLI and Image COCO) to validate our techniques. We discuss the results using several evaluation metrics and show that the proposed techniques outperform the traditional GAN-based text-generation methods.

1 Introduction

Text generation is an active research area and has many real-world applications, including, but not limited to, machine translation (Bahdanau et al., 2014), AI chat bots (Li et al., 2017), image captioning (Xu et al., 2014), question answering and information retrieval (Wang et al., 2017). Recurrent neural network language models (RNNLMs) (Mikolov et al., 2010) is the most popular approach for text generation which rely on maximum likelihood estimation (MLE) solutions such as teacher forcing (Williams and Zipser, 1989) (i.e. the model is trained to predict the next word given all the previous predicted words); however, it is well-known in the literature that MLE is a simplistic objective for this complex NLP task (Li et al., 2017). It is reported that MLE-based methods suffer from exposure bias (Huszr, 2015), which means that at training time the model is exposed to gold data only, but at test time it observes its own predictions. Hence, wrong predictions quickly accumulate and result in poor text generation quality.

However, generative adversarial networks (GANs) (Goodfellow et al., 2014) which are based on an adversarial loss function suffers less from the mentioned problems of the MLE solutions. The great success of GANs in image generation framework (Salimans et al., 2016) motivated researchers to apply its framework to NLP applications as well. GANs have been extensively used recently in various NLP applications such as machine translation (Wu et al., 2017; Yang et al., 2017a), dialogue models (Li et al., 2017), question answering (Yang et al., 2017b), and natural language generation (Gulrajani et al., 2017; Rajeswar et al., 2017; Press et al., 2017; Kim et al., 2017; Zhang et al., 2017; Cifka et al., 2018; Spinks and Moens, 2018; Haidar and Rezagholizadeh, 2019; Gagnon-Marchand et al., 2019; Rashid et al., 2019). However, applying GAN in NLP is challenging due to the discrete nature of the text. Consequently, back-propagation would not be feasible for discrete outputs and it is not straightforward to pass the gradients through the discrete output words of the generator.

Traditional methods for GAN-based text generation can be categorized according to the type of the signal used for discrimination into two categories: text-based and code-based techniques. Code-based methods such as adversarial autoencoder (AAE) (Makhzani et al., 2015) and adversarially regularized AE (ARAE) (Kim et al., 2017) derive a latent space representation of the text using an AE and attempt to learn data manifold of that latent space (Kim et al., 2017) instead of mod-

Proceedings of NAACL-HLT 2019, pages 2248–2258
Minneapolis, Minnesota, June 2 - June 7, 2019.

eling text directly. Text-based solutions, such as reinforcement learning (RL) based methods or approaches based on continuous approximation of discrete sampling, focus on generating text directly from the generator. RL-based methods treat the distribution of GAN generator as a stochastic policy and hold the discriminator responsible for providing proper reward for the generator's actions. However, the RL-based methods often need pre-training and are computationally more expensive compared to the methods of the other two categories. In the continuous approximation approach for generating text with GANs, the goal is to find a continuous approximation of the discrete sampling by using the Gumbel Softmax technique (Kusner and Hernández-Lobato, 2016) or approximating the non-differentiable argmax operator with a continuous function (Zhang et al., 2017; Gulrajani et al., 2017).

In this paper, we introduce Soft-GAN as a new solution for the main bottleneck of using GAN for text generation. Our solution is based on an AE to derive a soft representation of the real text (i.e. soft-text). This soft-text is fed to the GAN discriminator instead of the conventional one-hot representation used in (Gulrajani et al., 2017). Furthermore, we propose hybrid latent code and text-based GAN (LATEXT-GAN) approaches and show that how we can improve code-based and text-based text generation techniques by considering both signals in the GAN framework. We summarize the main contributions of this paper as:

- We introduce a new text-based solution Soft-GAN using the above soft-text discrimination. We also demonstrate the rationale behind this approach.

- We introduce LATEXT-GAN approaches for GAN-based text generation using both latent code and soft-text discrimination. To the best of our knowledge, this is the first time where a GAN-based text generation framework uses both code and text-based discrimination.

- We evaluate our methods using subjective and objective evaluation metrics. We show that our proposed approaches outperform the conventional GAN-based text generation techniques that do not need pre-training.

2 Background

Generative adversarial networks include two separate deep networks: a generator and a discriminator. The generator G_θ takes in a random variable, z following a distribution $P_z(z)$ and attempt to map it to the real data distribution $P_x(x)$. The output distribution of the generator is expected to converge to the real data distribution during the training. On the other hand, the discriminator f_w is expected to discern real samples from generated ones by outputting zeros and ones, respectively. During training, the generator and discriminator generate samples and classify them, respectively by adversarially affecting the performance of each other. In this regard, an adversarial loss function is employed for training (Goodfellow et al., 2014):

$$\begin{aligned}\min_{\theta}\max_{w}(&E_{x\sim P_x(x)}f_w(x)+\\ &E_{z\sim P_z(z)}(1-f_w(G_\theta(z))))\end{aligned} \tag{1}$$

As stated, using GANs for text generation is challenging because of the discrete nature of text. The main bottleneck is the *argmax* operator which is not differentiable and blocks the gradient flow from the discriminator to the generator.

$$\min_{\theta} E_{z\sim P_z(z)}(1-f_w(\text{argmax}(G_\theta(z)))) \tag{2}$$

3 Related Work

Text-based Solutions Generating text using pure GANs was inspired by improved Wasserstein GAN (IWGAN) work (Gulrajani et al., 2017). In IWGAN, a character level language model was developed based on adversarial training of a generator and a discriminator. Their generator is a convolution neural network (CNN) generating fixed-length texts. The discriminator is another CNN receiving 3D tensors as input sentences. The real sentences and the generated ones are represented using one-hot and softmax representations, respectively. A similar approach to IWGAN was proposed in (Rajeswar et al., 2017) with a recurrent neural network (RNN) based generator. In (Press et al., 2017), RNN is trained to generate text with GAN using curriculum learning (Bengio et al., 2009). The TextGAN (Zhang et al., 2017) method was proposed to alleviate the mode-collapsing problem by matching the high-dimensional latent feature distributions of real and synthetic sentences (Salimans et al., 2016).

Morever, several versions of the RL-based techniques using GAN have been introduced in the literature including Seq-GAN (Yu et al., 2017), RankGAN (Lin et al., 2017), MaliGAN (Che et al., 2017), LeakGAN (Guo et al., 2017), and MaskGAN (Fedus et al., 2018).

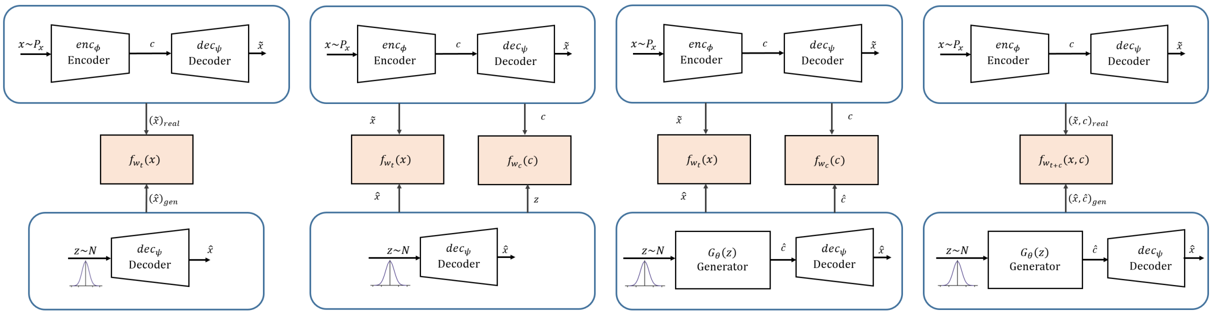
Code-based Solutions AEs have been exploited along with GANs in different architectures for computer vision application such as AAE (Makhzani et al., 2015), ALI (Dumoulin et al., 2016), and HALI (Belghazi et al., 2018). Similarly, AEs can be used with GANs for generating text. For instance, ARAE was proposed in (Kim et al., 2017) where it employs a discrete auto-encoder to learn continuous codes based on discrete inputs with a WGAN objective to learn an implicit probabilistic model over these codes. ARAE aims at exploiting GANs ability to push the generator to follow a continuous code space corresponding to the encoded real text in the AE. The generated code is then used by the decoder to generate the synthetic texts. A different version of the ARAE method was also introduced in (Spinks and Moens, 2018). In (Subramanian et al., 2018), sentence embeddings were learned by a generator to model a pre-trained sentence representation obtained by a general-purpose sentence encoder. A temperature sweeping framework was discussed in (Caccia et al., 2018) to evaluate the text generation models over whole quality-diversity spectrum and pointed out the fundamental flaws of quality-only evaluation. The Variational autoencoders (VAEs) (Kingma and Welling, 2013) were also applied for text generation in (Bowman et al., 2015; Hu et al., 2017).

4 Methodology

In this section, we introduce a new text-based solution by discriminating the reconstructed output of an AE (i.e., soft-text) with the synthesized generated text and we call it as Soft-GAN. Here, we use the decoder of the AE as the generator to generate the synthesized texts from random noise data z. The model is described in Figure 1a. We demonstrate the rationale behind this soft-GAN approach, which is to make the discrimination task of the discriminator between the real and synthetic texts more difficult and consequently providing a richer signal to the generator. We also introduce three LATEXT-GAN approaches where both code and text-based signals will be used in the GAN framework. We introduce LATEXT-GAN I approach on top of the AAE method. LATEXT-GAN II and III approaches will be proposed based on the ARAE approach. In the LATEXT-GAN I and II techniques, we use separate discriminators for the code and text discriminations. In the LATEXT-GAN III approach, the concatenation of the synthetic code and the synthetic text tries to mimic the concatenation of the latent code and the soft-text using a discriminator. The schematic diagram of the LATEXT-GAN I, II, and III approaches are described in Figures 1b, 1c, and 1d respectively. We share the parameters of the decoder of the AE to generate the synthesized text $\hat{x}$. In order to train these approaches, we train the auto-encoder and the GAN alternatingly by minimizing their loss functions using the WGAN-gradient penalty (WGAN-GP) approach (Gulrajani et al., 2017). In each iteration, the first step is the AE training in all of these techniques followed by the GAN loss functions. The autoencoder can be trained by using a cross-entropy or mean-squared loss functions. The input x to the AE is mapped to a latent code representation c which is decoded to soft-text $\tilde{x}$. In our experiments, we train the auto-encoder using mean-squared loss $\min\limits_{(\phi,\psi)} ||x - \tilde{x}||^2$. We describe the training details of the Soft-GAN, LATEXT-GAN I, II, and III methods in the following subsections where the term critic and discriminator used interchangeably. The notations that are used in this paper are described in Table 1.

x	One-hot representation of the training text
$\tilde{x}$	Soft-text: Reconstructed output of the AE
$\hat{x}$	Synthesized generated text
$\bar{x}$	$[\bar{x} \sim P_{\bar{x}}] \leftarrow \alpha\, [\tilde{x} \sim P_{\tilde{x}}] + (1-\alpha)\, [\hat{x} \sim P_{\hat{x}}]$
$z \sim N$	Random data drawn from a normal distribution
c	Latent code representation of the training text
$\hat{c}$	Synthesized generated code
$\bar{c}$	$[\bar{c} \sim P_{\bar{c}}] \leftarrow \alpha\, [c \sim P_c] + (1-\alpha)\, [\hat{c} \sim P_{\hat{c}}]$
$\bar{cz}$	$[\bar{cz} \sim P_{\bar{cz}}] \leftarrow \alpha\, [c \sim P_c] + (1-\alpha)\, [z \sim P_z]$
w_{t+c}	Parameters of the combination of text and code-based discriminator
w_t	Parameters of the text-based discriminator
w_c	Parameters of the code-based discriminator
ϕ	Parameters of the encoder
ψ	Parameters of the decoder
θ	Parameters of the generator
λ	Gradient penalty co-efficient
∇	describes gradient

Table 1: Notations that are used in this paper



(a) Soft-GAN (b) LATEXT-GAN I (c) LATEXT-GAN II (d) LATEXT-GAN III

Figure 1: Proposed Approaches

4.1 Soft-GAN

As stated, in conventional text-based discrimination approach IWGAN (Gulrajani et al., 2017), the real and the synthesized generated text are described by the one-hot and the softmax represenation respectively. A disadvantage of this technique is that the discriminator is able to tell apart the one-hot input from the softmax input very easily. One way to avoid this issue is to derive a continuous representation of words rather than their one-hot and train the discriminator to differentiate between the continuous representations. We use a conventional AE to replace the one-hot representation with softmax reconstructed output ($\tilde{x}$), which we refer to as soft-text. This soft-text representation is used as the real input to the discriminator. The synthetic generated text $\hat{x}$ is obtained by inputting the random noise data z to the shared decoder. We define the proposed method as Soft-GAN. In each iteration, the model is trained using the following steps after the AE training step:

- Train the text-based discriminator f_{w_t} for k times and the decoder once using the loss L_{critic_t} to maximize the ability of the f_{w_t} to discriminate between $\tilde{x}$ and $\hat{x}$:

$$L_{critic_t} = \min_{(w_t,\psi)} (-E_{\tilde{x}\sim P_{\tilde{x}}}[f_{w_t}(\tilde{x})] + E_{\hat{x}\sim P_{\hat{x}}}$$
$$[f_{w_t}(\hat{x})] + \lambda E_{\bar{x}\sim P_{\bar{x}}}[(||\nabla_{\bar{x}} f_{w_t}(\bar{x})||_2 - 1)^2]) \tag{3}$$

- Train the decoder based on the loss L_{gen} to fool the discriminator with improving the representation $\hat{x}$:

$$L_{gen} = \min_{\psi}(-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}}$$
$$[f_{w_t}(\tilde{x})]) \tag{4}$$

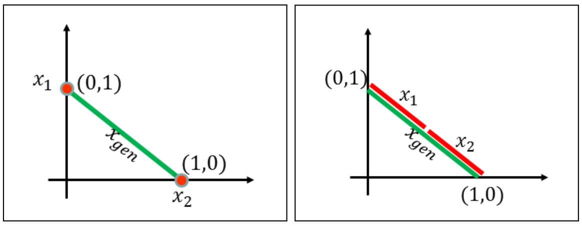


Figure 2: Locus of the input vectors to the discriminator f_{w_t} for a two-word language; Left panel: IWGAN, Right panel: Soft-GAN

4.1.1 Rationale: Why Soft-GAN should Work Better than IWGAN?

Suppose we have a language of vocabulary size of two words: x_1 and x_2. In the IWGAN approach, the one-hot representation of these two words (as two points in the Cartesian coordinates) and the span of the generated softmax outputs (as a line segment connecting them) is depicted in the left panel of Figure 2. As evident graphically, the task of the critic is to discriminate the points from the line connecting them, which is a rather simple very easy task, which makes it more prone to vanishing gradient.

On the other hand, the output locus of the soft-GAN decoder would be two red line segments as depicted in Figure 2 (Right panel) instead of two points (in the one-hot case). The two line segments lie on the output locus of the generator, which will make the generator more successful in fooling the critic.

4.2 LATEXT-GAN I

In the LATEXT-GAN I approach (Figure 1b), we deploy two critics: one for the soft-text discrimination and the other for the latent code discrimination. The text-based discriminator f_{w_t} is used to discriminate the soft-text output $\tilde{x}$ with the synthesized text $\hat{x}$ which is obtained by inputting the random noise data to the shared decoder. The code-based discriminator f_{w_c} is used to discriminate the random noise data z with the latent code c in the AAE setting which was explored for image generation. In the AAE setting (Makhzani et al., 2015), the encoder enhances its representation to a prior distribution z. It can be seen that the LATEXT-GAN I can be obtained by adding the above code-based discriminator f_{w_c} into the Soft-GAN. In each iteration, the model is trained using the following steps after the AE training step and the Equation 3 step of the Soft-GAN in section 4.1:

- Train the code-based discriminator f_{w_c} for k times using the loss L_{critic_c} to maximize the ability of the f_{w_c} to discriminate between c and z:
$$\begin{aligned} L_{critic_c} = \min_{w_c}(&E_{c\sim P_c}[f_{w_c}(c)]- \\ &E_{z\sim P_z}[f_{w_c}(z)]+ \\ &\lambda E_{\bar{c}z\sim P_{\bar{c}z}}[(||\nabla_{\bar{c}z} f_{w_c}(\bar{c}z)||_2 - 1)^2]) \end{aligned} \tag{5}$$
- Train the encoder and the decoder neural networks once with the loss L_{gen} to fool the discriminators f_{w_c} and f_{w_t} with improving the representations c and $\hat{x}$ respectively:
$$\begin{aligned} L_{\text{gen}} = \min_{(\phi,\psi)}(&-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}} \\ &[f_{w_t}(\tilde{x})] + E_{z\sim P_z}[f_{w_c}(z)] - E_{c\sim P_c}[f_{w_c}(c)]) \end{aligned} \tag{6}$$

4.3 LATEXT-GAN II

The LATEXT-GAN II approach (Figure 1c) is similar to the LATEXT-GAN I approach except the training of the code-based discriminator f_{w_c} is done as the ARAE training. The critic f_{w_c} is used to discriminate the synthetic code $\hat{c}$ with the latent code c. Here, the synthetic code is formed by using the ARAE method (Spinks and Moens, 2018). For each iteration in the model training, the AE training step and the Equation 3 step of the Soft-GAN in section 4.1 are carried out first. Then the following two steps are performed:

- Train the code-based discriminator f_{w_c} for k times and the encoder once using the loss L_{critic_c} to maximize the ability of the f_{w_c} to discriminate between c and $\hat{c}$:
$$\begin{aligned} L_{critic_c} = \min_{(w_c,\phi)}(&-E_{c\sim P_c}[f_{w_c}(c)] + E_{\hat{c}\sim P_{\hat{c}}} \\ &[f_{w_c}(\hat{c})] + \lambda E_{\bar{c}\sim P_{\bar{c}}}[(||\nabla_{\bar{c}} f_{w_c}(\bar{c})||_2 - 1)^2]) \end{aligned} \tag{7}$$
- Train the generator and the decoder neural networks once using the loss L_{gen} to fool the discriminators f_{w_t} and f_{w_c} with improving the representations $\hat{x}$ and $\hat{c}$ respectively:
$$\begin{aligned} L_{\text{gen}} = \min_{(\theta,\psi)}(&-E_{\hat{x}\sim P_{\hat{x}}}[f_{w_t}(\hat{x})] + E_{\tilde{x}\sim P_{\tilde{x}}} \\ &[f_{w_t}(\tilde{x})] - E_{\hat{c}\sim P_{\hat{c}}}[f_{w_c}(\hat{c})] + E_{c\sim P_c}[f_{w_c}(c)]) \end{aligned} \tag{8}$$

4.4 LATEXT-GAN III

In the third approach (Figure 1d), the combination of latent code c generated by ARAE (Spinks and Moens, 2018) and the soft-text output $\tilde{x}$ of an AE is used to signal the discriminator. We performed this combination by getting inspiration from an Adversarially Learned Inference (ALI) paper (Dumoulin et al., 2016) introduced for image generation. We call it as LATEXT-GAN III. Here, the discriminator $f_{w_{t+c}}$ tries to determine which combination of the samples derive from the latent code and the soft-text, ($\tilde{x}$,c), and which ($\hat{x}$, $\hat{c}$) are generated from the noise z. After the AE training step $\min\limits_{(\phi,\psi)} ||x - \tilde{x}||^2$, the LATEXT-GAN III model is trained using the next two steps in each iteration:

- Train the discriminator $f_{w_{t+c}}$ for k times, the encoder and the decoder once using the loss $L_{critic_{t+c}}$ to maximize the ability of the discriminator network to discriminate between ($\tilde{x}$,c) and ($\hat{x}$, $\hat{c}$):
$$\begin{aligned} L_{critic_{t+c}} = \min_{(w_{t+c},\phi,\psi)}(&-E_{(\tilde{x},c)\sim P_{\tilde{x}},P_c} \\ &[f_{w_{t+c}}(\tilde{x}, c)] + E_{(\hat{x},\hat{c})\sim P_{\hat{x}},P_{\hat{c}}}[f_{w_{t+c}}(\hat{x}, \hat{c})]+ \\ &\lambda E_{(\bar{x},\bar{c})\sim P_{\bar{x}},P_{\bar{c}}}[(||\nabla_{(\bar{x},\bar{c})} f_{w_{t+c}}(\bar{x}, \bar{c})||_2 - 1)^2]) \end{aligned} \tag{9}$$
- Train the generator and the decoder once based on L_{gen} to fool the discriminator $f_{w_{t+c}}$ with improving the representation ($\hat{x}$, $\hat{c}$):
$$\begin{aligned} L_{\text{gen}} = \min_{(\theta,\psi)}(&-E_{(\hat{x},\hat{c})\sim P_{\hat{x}},P_{\hat{c}}}[f_{w_{t+c}}(\hat{x}, \hat{c})] \\ &+ E_{(\tilde{x},c)\sim P_{\tilde{x}},P_c}[f_{w_{t+c}}(\tilde{x}, c)]) \end{aligned} \tag{10}$$

5 Experiments

5.1 Dataset and Experimental Procedures

We do our experiments on two different datasets: the Stanford Natural Language Inference (SNLI) corpus [1], which contains 714,667 sentences for

---

[1] https://github.com/aboev/arae-tf/tree/master/data_snli

training and 13323 sentences for testing, and the Image COCO [2] dataset's image caption annotations, where we sample [3] 10,000 sentences as training set and another 10,000 as test set (Zhu et al., 2018). We perform word-based experiments. For the SNLI dataset, we use a vocabulary size of 10000 words and use the maximum sentence length of size 15. For the COCO dataset, we use a vocabulary size of 5000 and perform experiments using the maximum sentence length of sizes 15 and 20. We train a simple AE using one layer with 512 LSTM cells (Hochreiter and Schmidhuber, 1997) for both the encoder and the decoder. For decoding, the output from the previous time step is used as the input to the next time step. We use the hidden code c from the last time step of the encoder and applied as an additional input at each time step of decoding. We normalize the code and then added an exponentially decaying noise before decoding. The greedy search approach is applied to get the best output. We train the auto-encoder using Adam (Diederik and Jimmy, 2014) optimizer with learning rate = 0.001, β_1= 0.9, and β_2= 0.999. We use CNN-based generator and discriminator with residual blocks (Gulrajani et al., 2017). The $tanh$ function is applied on the output of the ARAE generator (Kim et al., 2017). We train the generator and the discriminator using Adam optimizer with learning rate = 0.0001, β_1= 0.5, and β_2= 0.9. We do not apply any kind of attention mechanisms (Bahdanau et al., 2014; Zhang et al., 2018) and pre-training (Zhu et al., 2018) in our experiments. We use the WGAN-GP (Gulrajani et al., 2017) approach with 5 discriminator updates for every generator update and a gradient penalty co-efficient of λ=10 unlike a setup in (Zhu et al., 2018). For the AAE-based experiments, we normalize the data drawn from a prior distribution z. We train the models for 200000 iterations where in each iteration we sample a random batch and train the networks of the models.

5.2 Quantitative Evaluation

We use the frequently used BLEU metric (Papineni et al., 2002) to evaluate the word similarity between sentences and the perplexity to evaluate our techniques. We calculate BLEU-n scores for n-grams without a brevity penalty (Zhu et al., 2018). The results with the best BLEU-n scores in the synthesized generated texts are reported. To calculate the BLEU-n scores, we generate ten batches of sentences as candidate texts, i.e. 640 sentences and use the entire test set as reference texts. As the GAN-based models usually suffer from mode collapse (i.e., generating same samples over and over again), evaluating models by only BLEU metric is not appropriate. So, we also calculate recently proposed self-BLEU scores for the COCO dataset using maximum sentence length of size 20 and 10k synthetic sentences to evaluate the diversity (Zhu et al., 2018). Using one synthetic sentence as hypothesis and others as reference, the BLEU is calculated for every synthetic sentence, and define the average BLEU score as the self-BLEU (Zhu et al., 2018). A higher self-BLEU score describe less diversity. For the perplexity evaluations, we generate 100k and 10k sentences for the SNLI and the COCO datasets respectively using the models of the last iteration.

5.2.1 BLEU-n Scores evaluation

The BLEU score results for the n-grams of the synthesized texts are depicted in Table 2 and 3 with maximum sentence length of 15 for the SNLI and the COCO datasets respectively. We also report experimental results with a longer maximum sentence length of 20 using the COCO dataset to differentiate the effectiveness of code and text-based solutions (in Table 4). Furthermore, we report the BLEU and self-BLEU score results of our proposed approaches in Table 5 and 6 respectively for the COCO dataset to compare with the results of the existing approaches reported in (Zhu et al., 2018).

Model	B-2	B-3	B-4	B-5
AAE	0.797	0.614	0.449	0.294
ARAE	0.73	0.575	0.431	0.297
IWGAN	0.70	0.518	0.369	0.246
Soft-GAN	0.849	0.648	0.446	0.252
LATEXT-GAN I	**0.87**	**0.679**	**0.508**	0.336
LATEXT-GAN II	0.793	0.631	0.48	**0.338**
LATEXT-GAN III	0.782	0.617	0.466	0.33

Table 2: BLEU-n (B-n) scores results using SNLI dataset and 640 synthetic sentences

From tables 2, 3, and 4, we can see that our proposed approaches outperform the standalone code (AAE or ARAE) and text-based (IWGAN) solutions. For the maximum sentence length of size 15 experiments, the LATEXT-GAN I is better than LATEXT-GAN II and III for shorter length text (e.g., 2,3-grams). The performance of the LATEXT-GAN II and III degrades with increasing maximum sentence length to 20. This is be-

---

[2] http://cocodataset.org

[3] https://github.com/geek-ai/Texygen/tree/master/data

Model	B-2	B-3	B-4	B-5
AAE	0.733	0.477	0.284	0.156
ARAE	0.676	.457	0.287	0.172
IWGAN	0.636	0.417	0.258	0.155
Soft-GAN	**0.781**	0.492	0.296	0.155
LATEXT-GAN I	0.758	**0.496**	0.294	0.155
LATEXT-GAN II	0.707	0.489	**0.316**	**0.198**
LATEXT-GAN III	0.701	0.483	0.311	0.193

Table 3: BLEU-n (B-n) scores results for COCO dataset with maximum sentence length of size 15 and 640 synthetic sentences

Model	B-2	B-3	B-4	B-5
AAE	0.751	0.475	0.287	0.167
ARAE	0.665	0.447	0.279	0.162
IWGAN	0.669	0.454	0.294	0.178
Soft-GAN	**0.799**	**0.520**	**0.317**	**0.190**
LATEXT-GAN I	0.77	0.503	0.314	0.185
LATEXT-GAN II	0.687	0.456	0.283	0.174
LATEXT-GAN III	0.680	0.466	0.292	0.178

Table 4: BLEU-n (B-n) scores results for COCO dataset with maximum sentence length of size 20 and over 640 synthetic sentences

cause for longer sequence length experiments, the hidden code of the last time step might not be able to keep all the information from the earlier time steps. On the other hand, the LATEXT-GAN I and the Soft-GAN improve their performance with increasing maximum sentence length to 20. This might be because of the encoder enhances its representation better to the prior distribution, z from which the text is generated. Furthermore, the Soft-GAN outperforms all the proposed LATEXT GAN approaches.

We also compare our proposed approaches with TextGAN (Zhang et al., 2017), some RL-based approaches (SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017), MaliGAN (Che et al., 2017)) and MLE approach described in a benchmark platform (Zhu et al., 2018) where they apply pre-training before applying adversarial training. We evaluate the BLEU and Self-BLEU score results on 10k synthetic sentences using the maximum sentence length of size 20 for the COCO dataset with a vocabulary of size 5000 as in (Zhu et al., 2018). The BLEU and the self-BLEU score results are reported in Table 5 and 6 respectively. From Table 5, it can be noted that our proposed approaches show comparable results to the RL-based solutions for the BLEU score results. We can also see that our proposed LATEXT-GAN III approach gives lower self-BLEU scores in Table 6. From the above experimental results, we can note that LATEXT-GAN III can generate real-like and more diverse sentences compare to some approaches reported in (Zhu et al., 2018) and our other proposed approaches.

Model	B-2	B-3	B-4	B-5
TextGAN	0.593	0.463	0.277	0.207
SeqGAN	0.745	0.498	0.294	0.180
RankGAN	0.743	0.467	0.264	0.156
MaliGAN	0.673	0.432	0.257	0.159
MLE	0.731	0.497	0.305	0.189
Soft-GAN	0.787	0.496	0.286	0.150
LATEXT-GAN I	0.736	0.447	0.258	0.146
LATEXT-GAN II	0.672	0.430	0.257	0.147
LATEXT-GAN III	0.660	0.435	0.260	0.149

Table 5: *BLEU-n (B-n)* scores results for COCO dataset using sequence length 20 and 10k generated sentences

Model	B-2	B-3	B-4	B-5
TextGAN	0.942	0.931	0.804	0.746
SeqGAN	0.950	0.840	0.670	0.489
RankGAN	0.959	0.882	0.762	0.618
MaliGAN	0.918	0.781	0.606	0.437
MLE	0.916	0.769	0.583	0.408
Soft-GAN	0.988	0.950	0.847	0.612
LATEXT-GAN I	0.991	0.957	0.854	0.613
LATEXT-GAN II	0.896	0.755	0.523	0.263
LATEXT-GAN III	**0.874**	**0.706**	**0.447**	**0.205**

Table 6: Self-BLEU scores results for COCO dataset using sequence length 20 and 10k generated sentences

5.2.2 Perplexity Evaluation

The forward and reverse perplexities of the LMs trained with maximum sentence length of 15 and 20 using the SNLI and the COCO datasets respectively are described in Table 7. The forward perplexities (F-PPL) are calculated by training an RNN language model (Zaremba et al., 2015) on real training data and evaluated on the synthetic samples. This measure describe the fluency of the synthetic samples. We also calculate the reverse perplexities (R-PPL) by training an RNNLM on the synthetic samples and evaluated on the real test data. We can easily compare the performance of the LMs by using the forward perplexities while it is not possible by using the reverse perplexities as the models are trained using the synthetic samples with different vocabulary sizes. The perplexities of the LMs using real data are 16.01 and 67.05 for the SNLI and the COCO datasets respectively reported in F-PPL column. From the tables, we can note the models with lower forward perplexities (higher fluency) for the synthetic samples tend to have higher reverse perplexities except the AAE-based models (Cifka et al.,

2018) and/or the IWGAN. The forward perplexity for the IWGAN is the worst which means that the synthetic sentences of the IWGAN model are not fluent or real-like sentences. For the SNLI dataset, we can note that the LATEXT-GAN II and III approaches can generate more fluent sentences than the other approaches. For the COCO dataset, it can be seen that the forward perplexity of the LATEXT-GAN I (51.39) is far lower than the real data (67.05) which means the model suffers from mode-collapse. The Soft-GAN, the LATEXT-GAN II and III approaches suffer less from the mode-collapse.

Models	F-PPL SNLI	R-PPL SNLI	F-PPL COCO	R-PPL COCO
Real	16.01	-	67.05	-
AAE	74.56	48.04	83.34	257.21
ARAE	66.70	315.58	64.63	185.20
IWGAN	193.30	53.96	94.95	80.99
Soft-GAN	110.95	142.13	61.42	170.17
LATEXT-GAN I	86.24	56.78	**51.39**	158.21
LATEXT-GAN II	**53.22**	144.80	63.32	203.39
LATEXT-GAN III	54.80	143.42	71.60	214.76

Table 7: Forward (F) and Reverse (R) perplexity (PPL) results for the SNLI and COCO datasets using synthetic sentences of maximum length 15 and 20 respectively.

5.3 Human Evaluation

The subjective judgments of the synthetic sentences of the models trained using the COCO dataset with maximum sentence length of size 20 is reported in Table 8. We used 20 different random synthetic sentences generated by using the last iteration of each model and gave them to a group of 5 people. We asked them to rate the sentences based on a 5-point Likert scale according to their fluency. The raters are asked to score 1 which corresponds to gibberish, 3 corresponds to understandable but ungrammatical, and 5 correspond to naturally constructed and understandable sentences (Cifka et al., 2018). From Table 8, we can note that the proposed LATEXT-GAN III approach get the higher rate compare to the other approaches. From all the above different evaluations, we can note that the synthetic sentences by using the LATEXT-GAN II and III approaches are more balanced in diversity and fluency compare to the other approaches. We also depicted some examples of the synthetic sentences for the COCO and the SNLI datasets in Table 9 and 10 respectively.

Model	Fluency
Real	4.32
AAE	1.72
ARAE	2.60
IWGAN	1.58
Soft-GAN	1.66
LATEXT-GAN I	1.82
LATEXT-GAN II	2.48
LATEXT-GAN III	**2.70**

Table 8: Human Evaluation on the synthetic sentences

6 Conclusion and Future Work

In this paper, we introduced Soft-GAN as a new solution for the main bottleneck of using GAN for generating text, which is the discontinuity of text. This is based on applying soft-text to the GAN discriminator instead of the one-hot representation in the traditional approach. We also introduced three LATEXT-GAN approaches by combining the reconstructed output (soft-text) of an auto-encoder to the latent code-based GAN approaches (AAE and ARAE) for text generation. LATEXT-GAN I is formed on top of AAE method. LATEXT-GAN II and III approaches were formed based on ARAE. The LATEXT-GAN I and II approaches used separate discriminators for the synthetic text and the synthetic code discriminations. The LATEXT-GAN III used the combination of the soft-text and the latent code to compare with the concatenation of the synthetic text and the synthetic code by using a single discriminator. We evaluated the proposed approaches over the SNLI and the COCO datasets using subjective and objective evaluation metrics. The results of the experiments are consistent with different evaluation metrics. We showed the superiority of our proposed techniques, especially the LATEXT-GAN III method over other conventional GAN-based techniques which does not require pre-training. Finally, we summarize our plan for future work in the following:

1. We trained the GAN using WGAN-GP approach. Spectral normalization technique (Miyato et al., 2018) can be applied to stabilize the GAN training which could generate more diverse text in our settings.

2. The proposed approaches are the pure GAN-based techniques for text generation and they are not very powerful in generating long sentences. RL or self-attention (Zhang et al., 2018) techniques can be used as a tool to ac-

ARAE	IWGAN	Soft-GAN
a motorcycle parked outside of a counter street .	a small hang bathroom with a park and side .	a man is sitting with a counter counter .
a plane scene with focus on a toilet .	a group of kitchen sits near a sliding .	a dog with a people above a street .
a cat standing down the of a bathroom .	a picture from a giraffe in a garlic .	a blue of cat sitting in a red wall .
a bathroom with a toilet, a wall and .	a man shorts on a large table over .	a woman is looking with a red .
a white cat sits in the kitchen bowl .	a car is hidden from a small kitchen .	a dog is sitting in a bathroom .
LATEXT-GAN I	LATEXT-GAN II	LATEXT-GAN III
a bathroom bathroom with with toilet and and mirror .	a bathroom with a sink and and white plane .	there are a park with ride on around display .
a man kitchen next to a kitchen cabinets and .	a kitchen filled with wooden sink and a large window .	a kitchen with white cabinets and a black tub .
a man standing at a kitchen in a .	a person is parked on a city road .	a bathroom has a sink and a toilet .
a man in in front a a bathroom with .	a cat sitting on a bench of a city street .	a woman riding a bike with a large on .
a plane is sitting with the sky at .	a group of people sitting on a city bench .	a picture of a kitchen with a a skies .

Table 9: Example generated sentences with models trained using COCO dataset with maximum sentence length of size 20

ARAE	IWGAN	Soft-GAN
A little girl is playing outside .	A car is sitting in a house .	A children of a blue a shirt while a .
A woman is near the water .	Someone are Indian in his trinkets .	A little girl is looking while a a .
A group of people are sitting .	A woman is outside .	A couple of people are in white a a .
The lady is in the water .	the man is in the house .	A couple in a blue white is is standing .
A lady sitting on a bench .	The kids are laughing Bull .	A man in a white shirt outside a a .
LATEXT-GAN I	LATEXT-GAN II	LATEXT-GAN III
A woman is holding a .	People are walking in the park .	A group of people are sitting in a field .
A man is playing a .	The girl is playing with a a .	A boy is outside in a field .
A man is standing on an beach .	There are motorcycles outside .	A man is riding a bike in the water .
A man is walking in .	The young boy is wearing a red shirt .	A woman is riding a bike .
A little girl is playing in the .	A woman is walking on the park .	The man is standing .

Table 10: Example generated sentences with models trained using SNLI dataset with maximum sentence length of size 15

commodate this weakness.

3. We used the hidden code from the last time step of the encoder. Attention mechanism can be applied for decoding. Furthermore, the powerful transformer (Vaswani et al., 2017) can be applied for the auto-encoder which could improve the performance of the proposed approaches. Pre-train the auto-encoder before adversarial training can also improve the performance.

Acknowledgments

We would like to thank Professor Qun Liu, Chief Scientist of Speech and Language Computing, Huawei Noah's Ark Lab for his valuable comments on the paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Negar Rostamzadeh, Jovana Mitrovic, and Aaron Courville. 2018. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*, pages 41–48.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Massima Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language GANs falling short. *arXiv preprint arXiv:1811.02549*.

Tong Che, Yanran Li, Ruixiang Zhang, Devon R Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

Ondrej Cifka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. 2018. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*.

P Kingma Diederik and Ba Jimmy. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Jules Gagnon-Marchand, Hamed Sadeghi, Md. Akmal Haidar, and Mehdi Rezagholizadeh. 2019. SALSA-TEXT: Self attentive latent space based adversarial text generation. In *Canadian AI 2019*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein GANs. *arXiv preprint arXiv:1704.00028*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Md. Akmal Haidar and Mehdi Rezagholizadeh. 2019. TextKD-GAN: Text generation using knowledge distillation and generative adversarial networks. In *Canadian AI 2019*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation, 9(8):17351780*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*, pages 1587–1596.

Ferenc Huszr. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*.

Diederik P Kingma and Max. Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Matt J Kusner and José Miguel Hernández-Lobato. 2016. GANs for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. *arXiv preprint arXiv:1705.11001*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Tom Mikolov, Martin Karafit, Luk Burget, Jan ernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. 2017. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Ahmad Rashid, Alan Do-Omri, Md. Akmal Haidar, Qun Liu, and Mehdi Rezagholizadeh. 2019. Bilingual-GAN: A step towards parallel text generation. In *NeuralGen 2019*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In *NIPS*, pages 2234–2242.

Graham Spinks and Marie-Francine Moens. 2018. Generating continuous representations of medical texts. In *NAACL-HLT*, pages 66–70.

Sandeep Subramanian, Sai Rajeswar, Alessandro Sordoni, Adam Trischler, Aaron Courville, and Christopher Pal. 2018. Towards text generation with adversarially learned neural outlines. *NeurIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Zhang Dell. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1(2):270280*.

Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, and Yoshua Bengio. 2014. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017a. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017b. Semi-supervised QA with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2018. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

Yaoming Zhu, Sidi Lu, Guo Jiaxian Zheng Lei, Zhang Weinan, Wang Jun, and Yu Yong. 2018. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*.

Neural Text Generation from Rich Semantic Representations

Valerie Hajdik[1], **Jan Buys**[2], **Michael W. Goodman**[1] **and Emily M. Bender**[1]
[1]Department of Linguistics, University of Washington
[2]Paul G. Allen School of Computer Science and Engineering, University of Washington
`vhajdik@uw.edu, jbuys@cs.washington.edu,`
`{goodmami, ebender}@uw.edu`

Abstract

We propose neural models to generate high-quality text from structured representations based on Minimal Recursion Semantics (MRS). MRS is a rich semantic representation that encodes more precise semantic detail than other representations such as Abstract Meaning Representation (AMR). We show that a sequence-to-sequence model that maps a linearization of Dependency MRS, a graph-based representation of MRS, to English text can achieve a BLEU score of 66.11 when trained on gold data. The performance can be improved further using a high-precision, broad coverage grammar-based parser to generate a large silver training corpus, achieving a final BLEU score of 77.17 on the full test set, and 83.37 on the subset of test data most closely matching the silver data domain. Our results suggest that MRS-based representations are a good choice for applications that need both structured semantics and the ability to produce natural language text as output.

1 Introduction

Text generation systems often generate their output from an intermediate semantic representation (Yao et al., 2012; Takase et al., 2016). However many semantic representations are task- or domain-specific (He and Young, 2003; Wong and Mooney, 2007), while rule-based text generation systems often have incomplete coverage (Langkilde-Geary, 2002; Oepen et al., 2007).

In this work we combine the advantages of Minimal Recursion Semantics (MRS; Copestake et al., 2005) with the robustness and fluency of neural sequence-to-sequence models trained on large datasets. We hypothesize that MRS is particularly well-suited for text generation, as it is explicitly compositional, capturing the contribution to sentence meaning of all parts of the surface form (Bender et al., 2015).

In contrast, semantic representations such as Abstract Meaning Representation (AMR; Banarescu et al., 2013) seek to abstract away from the syntax of a sentence as much as possible. Therefore MRS captures meaning distinctions that AMR fails to represent (see Fig. 1).

Our approach (§2) uses neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2014) to map linearizations of directed acyclic graphs (DAGs) to text, similar to the approach proposed by Konstas et al. (2017) to generate text from AMR. We use Dependency MRS (DMRS; Copestake, 2009), a graph-based representation in which nodes are MRS predicates (annotated with additional attributes) and edges represent relations between predicates. MRS and DMRS are interconvertible and the graph-based representation enables more convenient linearization and manipulation than MRS's variable-based representation (Copestake et al., 2016).

Results (§3) show that neural DMRS to English text generation can obtain up to 83.37 BLEU and 32% exact match, substantially higher than previous work. In particular, we obtain an 11.6 BLEU improvement through semi-supervised training using the output of a grammar-based parser, compared to training on gold data only. In comparison a grammar-based generator obtained 62.05 BLEU, and an approach based on DAG Transducers (Ye et al., 2018) 68.07 BLEU. Ablation experiments show that node attributes encoding fine-grained morpho-semantic information such as number and tense contribute more than 12 BLEU points. The highest reported result for AMR generation is 33.8 BLEU (Konstas et al., 2017); on the same dataset our best model obtains 75.8 BLEU. While a more detailed meaning representation is harder to produce, our results suggest that MRS could be suitable for text generation applications where precise semantic representations are required.

Proceedings of NAACL-HLT 2019, pages 2259–2266
Minneapolis, Minnesota, June 2 - June 7, 2019.

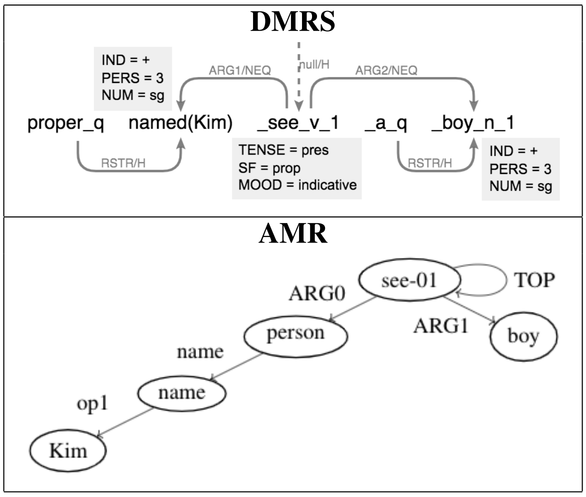


Figure 1: DMRS and AMR graphs for the sentence *Kim sees a boy*. Because DMRS includes tense and number, and has a node for the determiner, it can distinguish between, e.g. *Kim sees a boy* and *Kim saw the boys*, which AMR does not do.

PENMAN

```
(10002 / _see_v_1
  :tense PRES
  :sf PROP
  :perf -
  :mood INDICATIVE
  :ARG1-NEQ (10001 / named
    :carg "Kim"
    :pers 3
    :num SG
    :ind +)
  :ARG2-NEQ (10004 / _boy_n_1
    :pers 3
    :num SG
    :ind +
    :RSTR-H-of (10003 / _a_q)))
```

Linearization

```
( _see_v_1 mood=INDICATIVE|perf=-|sf
    ↪ =PROP|tense=PRES ARG1-NEQ (
    ↪ named0 ind=+|num=SG|pers=3 )
    ↪ ARG2-NEQ ( _boy_n_1 ind=+|num=
    ↪ SG|pers=3 RSTR-H-of ( _a_q ) )
    ↪  )
```

Figure 2: The DMRS for the sentence *Kim sees a boy.* in PENMAN format (top) and the linearization used by our model (bottom).

2 Approach

2.1 Data

Our gold training data are parallel MRS and English text corpora, derived from the 1214 release of the Redwoods Treebank (Oepen et al., 2002).[1] MRS is implemented as the semantic layer of the English Resource Grammar (ERG; Flickinger, 2000, 2011), a broad-coverage, hand-engineered computational grammar of English. The Redwoods annotation was produced in conjunction with the ERG by parsing each sentence into a forest (discarding unparsable sentences), followed by manual disambiguation (Flickinger et al., 2017).

About half of the training data comes from the Wall Street Journal (sections 00-21), while the rest spans a range of domains, including Wikipedia, e-commerce dialogues, tourism brochures, and the Brown corpus. The data is split into training, development and test sets with 72,190, 5,288, and 10,201 sentences, respectively.

2.2 Graph linearization

We use PyDelphin[2] to convert MRS annotations to DMRS. In order to apply sequence-to-sequence models to graph-to-text generation, we then linearize the DMRS into PENMAN format (which is also used to represent AMR). We follow Goodman (2018, pp. 82–86) in finding normalized spanning trees through depth-first traversal over the directed acyclic DMRS graphs.[3] The PENMAN format defines each node once, supports node attributes and edge labels, marks edges whose direction is reversed in the traversal, and represents edges which are not covered by the spanning tree.

The PENMAN format is processed further to obtain a linearization appropriate as input to sequence-to-sequence models, similar to the approach proposed by Konstas et al. (2017) for AMR linearization (see Fig. 2). Node variable identifiers are removed, node attributes are concatenated, and named entities are anonymized. Predicates that appear only once in the training data are treated as unknowns. Preprocessing and unknown word handling are described in greater detail in Appendices A and B.

2.3 Model

Our neural generator follows the standard encoder-decoder paradigm (Bahdanau et al., 2014). The encoder is a two-layer bidirectional LSTM. Predicates and their attributes are embedded separately; their embeddings are then concatenated (Sennrich and Haddow, 2016). The

---

[1] http://svn.delph-in.net/erg/tags/1214/tsdb/gold

[2] https://github.com/delph-in/pydelphin

[3] https://github.com/goodmami/mrs-to-penman

decoder uses global soft attention for alignment (Luong et al., 2015), and pointer attention to copy unknown tokens directly to the output (Gulcehre et al., 2016). The models are trained using Adam (Kingma and Ba, 2014). Dropout is applied to non-recurrent connections. Decoding uses beam search (width 5). The generator is implemented using OpenNMT-py (Klein et al., 2017). Hyperparameter details are given in Appendix C. Our code is available online.[4]

2.4 Semi-supervised training

We augment the gold training data with a *silver* dataset generated using ACE,[5] a parser for the ERG, to parse sentences to MRS. We sample one million sentences from the Gigaword corpus (Parker et al., 2011), restricted to articles published before the year 2000, to match the domain of the Wall Street Journal data. The parser failed to parse about 10.3% of the Gigaword sentences, so these were discarded. While there are robust MRS parsers (Buys and Blunsom, 2017; Chen et al., 2018), the MRSs they produce are less accurate and not guaranteed to be well-formed. Our approach thus differs from Konstas et al. (2017), who used self-training to improve AMR to text generation by iteratively training on larger amounts of data parsed by their neural parser.[6]

3 Results

We compare the performance of our neural generator when trained on either gold, silver, or gold and silver data (Table 1). Generation quality is primarily evaluated with BLEU (Papineni et al., 2002), using SacreBLEU (Post, 2018).[7] We evaluate the neural models on both the full Redwoods test set ('All') and the WSJ subset.

The results show that our neural generator obtains very strong performance. Semi-supervised training leveraging the ERG parser leads to an 11 BLEU point improvement on Redwoods, comparing to supervised training only. We found that the best semi-supervised results are obtained by upsampling the gold data so that the gold to silver ratio in training examples is 1:2. Interestingly, training on silver data performs only slightly worse than training on both gold and silver.

3.1 Baselines

Our baselines are the ERG's grammar-based generator (Carroll et al., 1999; Carroll and Oepen, 2005) and the DAG transducer generator of Ye et al. (2018). To compare our models against the grammar-based generator, implemented in ACE, we need to restrict the evaluation to examples from which ACE is able to generate ('All overlap').[8] In addition to BLEU, we also report exact match accuracy on the overlapping subset.

Results show that our neural models outperform the grammar-based generator by a large margin. ACE ranks candidate generations with a discriminative ranker based on structural features over its derivations (Velldal and Oepen, 2006). However, it does not use a language model trained on large amounts of text, which would likely improve fluency substantially.

The DAG transducer was trained to generate from Elementary Dependency Structures (EDS; Oepen and Lønning, 2006), an MRS-derived representation almost equivalent to DMRS (after edge properties are removed, which Table 3 shows has an effect of less than 1 BLEU point). It was evaluated against the same WSJ test set reference generations, but trained using both less gold data (only the WSJ subsection) and less silver data (300K vs 900K sentences). Our model trained on WSJ gold data performs only slightly worse (65.78 BLEU; see Table 2) and all our semi-supervised models obtain substantially higher results.

3.2 Out of domain evaluation

We evaluate the in- and out-of-domain performance of our approach by training models on either WSJ gold data only, or both WSJ gold data and Gigaword silver data, and evaluating on different domains. The results in Table 2 show that while the generator performs best on test data which matches the training domain (news), semi-supervised training leads to substantial out-of-domain improvements on the Wikipedia and the Brown corpus portions of the test set.

---

[4] `https://github.com/shlurbee/dmrs-text-generation-naacl2019`

[5] ACE version 0.9.25, with the 1214 ERG release, available at `http://sweaglesw.org/linguistics/ace`

[6] The ACE parser obtained 93.5 Smatch score on parsable sentences (Buys and Blunsom, 2017), while the neural AMR parser (Konstas et al., 2017) obtained 62.1 Smatch (on a different domain).

[7] `https://github.com/mjpost/sacreBLEU`

[8] Despite all test sentences being *parsable* by the ERG, there are gaps in generation coverage, primarily because ACE is unable to generate words outside the grammar's vocabulary.

Model	BLEU (All)	BLEU (WSJ)	BLEU (All overlap)	Exact Match%	Coverage%
Neural MRS (gold)	66.11	73.12	69.27	24.09	100
Neural MRS (silver)	75.43	81.76	77.13	25.82	100
Neural MRS (gold + silver)	77.17	83.37	79.15	32.07	100
ACE (ERG)	–	–	62.05	15.08	78
DAG transducer (Ye et al., 2018)	–	68.07	–	–	100

Table 1: BLEU and exact-match scores over held-out test set

	Training Data	
Test domain	WSJ	WSJ + Giga
WSJ	65.78	83.42
Brown	45.00	76.99
Wikipedia	35.90	62.26

Table 2: BLEU scores for domain match experiments

Ablation	BLEU
All attributes	72.06
No node attributes	59.37
No node attr except num, tense	67.34
No edge features	71.27

Table 3: Results of semantic feature ablation, model trained with gold data only

3.3 Attribute ablations

To understand which elements of MRS contribute most to our generator's performance, we ablate node (predicate) and edge attributes from both the training and test DMRS graphs (Table 3). In the training data, number and tense show the most variation among node attributes, and subsequently have the largest effect on the reported BLEU score. The most common value for number is SG, but 62.36% of sentences contain a node with PL. Similarly, 42.41% of sentences contain a tense value other than PRES or UNTENSED. Many other attributes are less informative: Mood has a value other than INDICATIVE in only 0.38% of sentences, and perf is + in just 9.74% of sentences. Edge features (including H, EQ and NEQ) encode constraints on scopal relationships (see Copestake 2009). Removing them, which makes the DMRS representation close to equivalent to EDS, has only a small impact on performance.

Representation	Train on Gold	Train on Gold+Silver
AMR	22.0	33.8
DMRS - no attributes	40.1	63.6
DMRS - all attributes	56.9	75.8

Table 4: BLEU scores for evaluating AMR and DMRS generators on an AMR test set

3.4 Comparison with AMR generation

We compare our approach to AMR-to-text generation by evaluating our generator on a standard AMR test set (LDC2015E86). As we do not have manually verified MRSes available on this test set, we use ACE to parse the reference sentences to silver MRSes. We then evaluate the outputs that our generator produces from those MRSes. About 20% of the examples could not be parsed by ACE, and are discarded for the MRS evaluation. We compare our generator to the neural AMR generator of Konstas et al. (2017) for models trained on gold as well as gold plus silver data.[9]

We evaluate DMRS models both with and without predicate and edge attributes, as these attributes contain information that is absent from AMR.[10] The results in Table 4 show that our MRS generator performs better than the AMR generator by a large margin, even when the additional MRS attributes are excluded. Our system results are reported on the subset for which we obtained MRS parses. AMR results are as given by Konstas et al. (2017) and cover the entire test set.

---

[9] The AMR and DMRS systems have different gold training data, but the same source of silver data.

[10] Recently, Donatelli et al. (2018) proposed adding tense and aspect to AMR, but this annotation is not yet available in a large AMR corpus.

Type	B80-89	B60-69	B40-49	All
Unproblematic	56.4	39.55	48.8	47.1
Slightly problematic	18.0	9.2	3.3	7.6
Moderately problematic	12.8	25.0	18.7	19.8
Ungrammatical	5.1	7.9	8.1	7.6
Other serious error	7.7	18.4	21.1	18.1
Number of errors	39	76	123	238
Errors per item	1.18	2.30	3.73	7.21

Table 5: Percentage of errors of each type, across 99 sampled items, grouped by BLEU score

3.5 Error analysis

We sampled 99 items for error analysis from the dev set, 33 each from among sentences with sentence-level BLEU scores of 80-89, 60-69, and 40-49.[11] We identified all differences between these strings and the reference strings and then labeled each difference with a fine-grained error type.[12] We classified the differences into 238 errors, distributed across five levels of severity (Table 5).

Almost half of the differences (47.1%) were unproblematic, including spelling variants, meaning-preserving punctuation variation and grammatical alternations (such as optional *that* or auxiliary contraction as in (1)). The slightly problematic category includes close synonyms (e.g. *sometime* v. *someday*), spelled out number names where Arabic numerals are preferred, and differences in formatting. The next more serious category (moderately problematic) includes meaning-changing differences in punctuation, tense or aspect, and minor grammatical errors such as swapping *who* and *which* in relative clauses or *a* v. *an*.

(1) a. I think I would like a Sony. [sys.]
    b. I think I'd like a Sony. [ref.]

Finally, among the most serious errors, we find cases where the generator provided ungrammatical output or grammatical output not conveying the correct semantics. The former include spurious additional tokens, ungrammatical word orders, and ungrammatical inflection. Serious errors that nonetheless resulted in grammatical strings include meaning-changing dropped or swapped tokens (as in (2)), spurious additional tokens, and word order changes that alter the semantics or pragmatics of the string.

(2) a. For such cases, machine learning techniques emulate human linguistics and learn from training examples to predict future events. [sys.]
    b. For such cases, machine learning techniques emulate human cognition and learn from training examples to predict future events. [ref.]

In summary, we find that the BLEU scores underestimate the quality of system outputs, due to unproblematic differences (N=112) and differences, like formatting markup (N=6), not reflected in the input semantic representations. Among the 108 moderate to serious differences, about a third (35) involve punctuation, suggesting that meaning signalled by punctuation could be better reflected in the semantic representations. About half (52) involve added, dropped, or swapped tokens, showing room for improvement in the generator's ability to learn appropriate connections between semantic predicates and surface forms. The remainder (21) involve inflection, grammatical alternations (such as *who*/*which*) and word order constraints, showing room for improvement in mimicking grammatical processes.

4 Conclusion

We have shown that neural sequence-to-sequence models can be used to generate high quality natural language text from Minimal Recursion Semantics representations, in contrast to both existing MRS-based generators and neural generators based on other broad-coverage semantic representations. Furthermore, we have demonstrated that a large hand-crafted grammar can be leveraged to produce large training sets, which improves performance of neural generators substantially. Therefore we argue that the ability to generate high quality text from MRS makes it a good choice of representation for text generation applications that require semantic structure. For future work, we are interested in applying graph-to-sequence neural networks (Beck et al., 2018; Song et al., 2018) to MRS-to-text generation.

Acknowledgements

Thanks to Yannis Konstas for sharing preliminary results on DMRS generation, and Swabha Swayamdipta for discussions. This research was supported in part by NSF (IIS-1524371) and Samsung AI Research.

---

[11] Items with BLEU scores lower than 40 tend to be very short and primarily involve formatting differences.

[12] This was done by a single annotator only. The labels were generated bottom up, with new labels added as needed in the course of annotation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th international conference on Computational Semantics*, pages 239–249.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226.

John Carroll, Ann Copestake, and Dan Flickinger. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86–95.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *International Conference on Natural Language Processing*, pages 165–176. Springer.

Yufei Chen, Weiwei Sun, and Xiaojun Wan. 2018. Accurate shrg-based semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 408–418, Melbourne, Australia.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyska. 2016. Resources for building applications with dependency minimal recursion semantics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, Portoro, Slovenia.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Ann A. Copestake. 2009. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 1–9.

Lucia Donatelli, Michael Regan, William Croft, and Nathan Schneider. 2018. Annotation of tense and aspect semantics for sentential amr. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions*, pages 96–108.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Nat. Lang. Eng.*, 6(1):15–28.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In E.M. Bender and J.E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, pages 31–50. CSLI Publications, Stanford.

Dan Flickinger, Stephan Oepen, and Emily M. Bender. 2017. Sustainable development and refinement of complex linguistic annotations at scale. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 353–377. Springer Netherlands, Dordrecht.

Michael Wayne Goodman. 2018. *Semantic Operations for Transfer-based Machine Translation*. Ph.D. thesis, University of Washington.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149.

Yulan He and Steve Young. 2003. A data-driven spoken language understanding system. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 583–588. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1701.02810*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion*

Volume: Proceedings of the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the International Natural Language Generation Conference*, pages 17–24.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The lingo redwoods treebank: Motivation and preliminary applications. In *The 17th International Conference on Computational Linguistics: Project Notes*.

Stephan Oepen, Erik Velldal, Jan Tore Lønning, Paul Meurer, Victoria Rosén, and Dan Flickinger. 2007. Towards hybrid quality-oriented machine translation. on linguistics and probabilities in mt. In *In Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition ldc2011t07. DVD. Philadelphia: Linguistic Data Consortium.

M. Post. 2018. A Call for Clarity in Reporting BLEU Scores. *arXiv preprint arXiv:1804.08771*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas.

Erik Velldal and Stephan Oepen. 2006. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 517–525, Sydney, Australia.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–179.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue & Discourse*, 3(2):11–42.

Yajie Ye, Weiwei Sun, and Xiaojun Wan. 2018. Language generation via dag transduction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1928–1937.

A Preprocessing Details

We preprocess the text by removing special formatting characters such as HTML tags, converting all double quotations to the same format (i.e., ” instead of `''`) and removing double brackets ‘[[’ and ‘]]’ that represent hyperlinks in Wikipedia data. These formatting characters are somewhat arbitrary and represent content (markup) that is orthogonal to the meaning of the text, so would be difficult for a model (or a human) to predict from a semantic representation for the text alone (such as MRS). After preprocessing and normalization, sentences in the gold and silver training data identical to sentences in the test set were removed.

Named entities (as annotated in MRS) are anonymized according to the conventions described in Konstas et al. (2017) prior to training. The purpose of anonymization is to reduce the sparsity of tokens such as dates and named entities that, while having a different surface form almost every time they appear in the text, should be treated similarly by the model. These tokens are replaced, both in the DMRS graph and in the raw sentence text, with numbered placeholders like *named0*, *named1*, and *month*.

Sentences are tokenized using the NLTK's implementation (Bird et al., 2009) of Moses-style tokenization (Koehn et al., 2007). During post-processing, generated sentences are de-anonymized (by replacing the placeholder with the original surface form) and de-tokenized, reversing the NLTK tokenization.

B Unknown Word Handling

We use a combination of two methods to deal with unknown words: anonymizing words to a special *UNK0* token and using the pointer attention to copy the token with highest attention value directly from the source to the target (Gulcehre et al., 2016). There are two types of unknown words in our setup: words that are not recognized by the model because they do not appear in the training data and words that may or may not appear in the training data that are not recognized by the ERG because they are not part of its lexicon. Inspection shows that words missing from the ERG lexicon are often misspellings, uncommon proper names, or domain-specific terminology.

For words that are not part of the ERG lexicon, if the word appears only once in the training data, we replace it with *UNK0*. Otherwise, if the word appears multiple times in the training set, we include its surface form (instead of a predicate) as the node token to enable the model to learn from it. In early experiments, we also tried either anonymizing all unknowns or including all unknowns (relying on the pointer copy mechanism), and found that anonymizing singletons gave best performance while also keeping the vocabulary size reasonable.

C Model Details

The encoder and decoder each have two 800-dimension Bidirectional LSTM layers. The main input embedding layer has 500 dimensions and the size of the attribute embedding layer varies between 22 and 25 depending on the number of attribute combinations in the training data. We train using negative log likelihood loss, optimizing using Adam with initial learning rate = 0.001, β_1 = 0.9 and β_2 = 0.999. For regularization, we apply dropout of 0.3 to the non-recurrent connections of the LSTM layers and after the attention layer. The number of layers, per-layer hidden state size, and dropout rates were all chosen empirically to minimize validation perplexity.

Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation[*]

Amit Moryossef[†] **Yoav Goldberg**[†‡] **Ido Dagan**[†]
amitmoryossef@gmail.com, {yogo,dagan}@cs.biu.ac.il

[†]Bar Ilan University, Ramat Gan, Israel
[‡]Allen Institute for Artificial Intelligence

Abstract

Data-to-text generation can be conceptually divided into two parts: ordering and structuring the information (planning), and generating fluent language describing the information (realization). Modern neural generation systems conflate these two steps into a single end-to-end differentiable system. We propose to split the generation process into a symbolic text-planning stage that is faithful to the input, followed by a neural generation stage that focuses only on realization. For training a plan-to-text generator, we present a method for matching reference texts to their corresponding text plans. For inference time, we describe a method for selecting high-quality text plans for new inputs. We implement and evaluate our approach on the WebNLG benchmark. Our results demonstrate that decoupling text planning from neural realization indeed improves the system's reliability and adequacy while maintaining fluent output. We observe improvements both in BLEU scores and in manual evaluations. Another benefit of our approach is the ability to output diverse realizations of the same input, paving the way to explicit control over the generated text structure.

1 Introduction

Consider the task of data-to-text generation, as exemplified in the WebNLG corpus (Colin et al., 2016). The system is given a set of RDF triplets describing facts (entities and relations between them) and has to produce a fluent text that is faithful to the facts. An example of such triplets is:

`John, birthPlace, London`
`John, employer, IBM`

With a possible output:

1. John, who was born in London, works for IBM.

Other outputs are also possible:

2. John, who works for IBM, was born in London.
3. London is the birthplace of John, who works for IBM.
4. IBM employs John, who was born in London.

These variations result from different ways of structuring the information: choosing which fact to mention first, and in which direction to express each fact. Another choice is to split the text into two different sentences, e.g.,

5. John works for IBM. John was born in London.

Overall, the choice of fact ordering, entity ordering, and sentence splits for these facts give rise to 12 different structures, each of them putting the focus on somewhat different aspect of the information. Realistic inputs include more than two facts, greatly increasing the number of possibilities.

Another axis of variation is in how to verbalize the information for a given structure. For example, (2) can also be verbalized as

2a. John works for IBM and was born in London.

and (5) as:

5a. John is employed by IBM. He was born in London.

We refer to the first set of choices (how to structure the information) as *text planning* and to the second (how to verbalize a plan) as *plan realization*.[1]

The distinction between planning and realization is at the core of classic natural language generation (NLG) works (Reiter and Dale, 2000; Gatt and Krahmer, 2017). However, a recent wave of *neural NLG systems* ignores this distinction

---

[*] This research was supported in part by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1) and by a grant from Theo Hoffenberg and Reverso.

[1] Note that the variation from 5 to 5a includes the introduction of a pronoun. This is traditionally referred to as *referring expression generation* (REG), and falls between the planning and realization stages. We do not treat REG in this work, but our approach allows natural integration REG systems' outputs.

Proceedings of NAACL-HLT 2019, pages 2267–2277
Minneapolis, Minnesota, June 2 - June 7, 2019.

and treat the problem as a single end-to-end task of learning to map facts from the input to the output text (Gardent et al., 2017; Dušek et al., 2018). These neural systems encode the input facts into an intermediary vector-based representation, which is then decoded into text. While not stated in these terms, the neural system designers hope for the network to take care of both the planning and realization aspect of text generation. A notable exception is the work of Puduppully et al. (2018), who introduce a neural content-planning module in the end-to-end architecture.

While the neural methods achieve impressive levels of output fluency, they also struggle to maintain coherency on longer texts (Wiseman et al., 2017), struggle to produce a coherent order of facts, and are often not faithful to the input facts, either omitting, repeating, hallucinating or changing facts (the NLG community refers to such errors as errors in *adequacy* or *correctness* of the generated text). When compared to template-based methods, the neural systems win in fluency but fall short regarding content selection and faithfulness to the input (Puzikov and Gurevych, 2018). Also, they do not allow control over the output's structure. We speculate that this is due to demanding too much of the network: while the neural system excels at capturing the language details required for fluent realization, they are less well equipped to deal with the higher levels text structuring in a consistent and verifiable manner.

Proposal we propose an explicit, symbolic, text planning stage, whose output is fed into a neural generation system. The text planner determines the information structure and expresses it unambiguously—in our case as a sequence of ordered trees. This stage is performed symbolically and is guaranteed to remain faithful and complete with regards to the input facts. Once the plan is determined,[2] a neural generation system is used to transform it into fluent, natural language text. By being able to follow the plan structure closely, the network is alleviated from the need to determine higher-level structural decisions and can track what was already covered more easily. This allows the network to perform the task it excels in, producing fluent, natural language outputs.

We demonstrate our approach on the WebNLG corpus and show it results in outputs which are as fluent as neural systems, but more faithful to the input facts. The method also allows explicit control of the output structure and the generation of diverse outputs (some diversity examples are available in the Appendix). We release our code and the corpus extended with matching plans in `https://github.com/AmitMY/chimera`.

2 Overview of the Approach

Task Description Our method is concerned with the task of generating texts from inputs in the form of RDF sets. Each input can be considered as a graph, where the entities are nodes, and the RDF relations are directed labeled edges. Each input is paired with one or more reference texts describing these triplets. The reference can be either a single sentence or a sequence of sentences. Formally, each input G consists of a set of triplets of the form (s_i, r_i, o_i), where $s_i, o_i \in V$ ("subject" and "object") correspond to entities from DBPedia, and $r_i \in R$ is a labeled DBPedia relation (V and R are the sets of entities and relations, respectively). For example, Figure 1a shows a triplet set G and Figure 1d shows a reference text. We consider the data set as a set of input-output pairs (G, ref), where the same G may appear in several pairs, each time with a different reference.

Method Overview We split the generation process into two parts: text planning and sentence realization. Given an input G, we first generate a text plan $plan(G)$ specifying the division of facts to sentences, the order in which the facts are expressed in each sentence, and the ordering of the sentences. This data-to-plan step is non-neural (Section 3). Then, we generate each sentence according to the plan. This plan-to-sentence step is achieved through an NMT system (Section 4).

Figure 1 demonstrates the entire process.

To facilitate our plan-based architecture, we devise a method to annotate (G, ref) pairs with the corresponding plans (Section 3.1), and use it to construct a dataset which is used to train the plan-to-text translation. The same dataset is also used to devise a plan selection method (Section 3.2).

General Applicability It is worth considering the dataset-specific vs. general applicability aspects of our method. On the low-level details, this

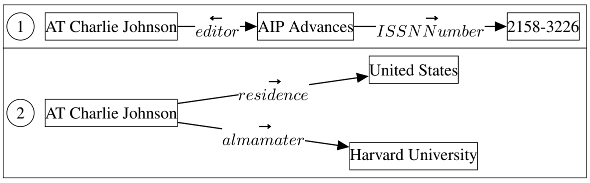
---

[2] The exact plan can be determined based on a data-driven scoring function that ranks possible suggestions, as in this work, or by other user provided heuristics or a trained ML model. The plans' symbolic nature and precise relation to the input structures allow verification of their correctness.

(a) Example input RDF

AIP_Advances | editor | A.T._Charlie_Johnson
A.T._Charlie_Johnson | almaMater | Harvard_University
AIP_Advances | ISSN_number | ”2158-3226”
A.T._Charlie_Johnson | residence | United_States

(b) Possible corresponding text plan



(c) Linearization of the text plan

A.T._Charlie_Johnson ← editor [**AIP_Advances** → issn number [**2158-3226**]] .
A.T._Charlie_Johnson → residence [**United_States**] → alma mater [**Harvard_University**]

(d) Possible output sentence

A.T. Charlie Johnson is the editor of **AIP Advances** which has the ISSN number **2158-3226**.
He lives in the **United States**, and graduated from **Harvard University**.

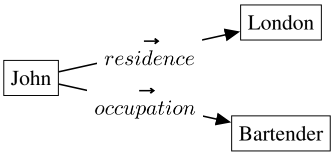
Figure 1: Summary of our proposed generation process: the planner takes the input RDF triplets in (a), and generates the explicit plan in (b). The plan is then linearized (c) and passed to a neural generation system, producing the output (d).

work is very much dataset dependent. We show how to represent plans for specific datasets, and, importantly for this work, how to automatically construct plans for this dataset given inputs and expected natural language outputs. The method of plan construction will likely not generalize “as is” to other datasets, and the plan structure itself may also be found to be lacking for more demanding generation tasks. However, on a higher level, our proposal is very general: intermediary plan structures can be helpful, and one should consider ways of obtaining them, and of using them. In the short term, this will likely take the form of ad-hoc explorations of plan structures for specific tasks, as we do here, to establish their utility. In the longer term, research may evolve to looking into how general-purpose plan are structured. Our main message is that the separation of planning from realization, even in the context of neural generation, is a useful one to be considered.

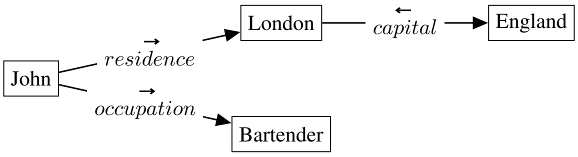
3 Text Planning

Plan structure Our text plans capture the division of facts to sentences and the ordering of the sentences. Additionally, for each sentence, the plan captures (1) the ordering of facts within the sentence; (2) The ordering of entities within a fact, which we call the *direction* of the relation. For example, the {`A, location, B`} relation can be expressed as either *A is located in B* or *B is the location of A*; (3) the structure between facts that share an entity, namely chains and sibling structures as described below.

(a) Chain: John lives in London, the capital of England.



(b) Sibling: John lives in London and works as a bartender.



(c) Combination: John lives in London, the capital of England, and works as a bartender.

Figure 2: Fact construction structure.

A text plan is modeled as a sequence of sentence plans, to be realized in order. Each sentence plan is modeled as an ordered tree, specifying the structure in which the information should be realized. Structuring each sentence as a tree enables a clear succession between different facts through shared entities. Our text-plan design assumes that each entity is mentioned only once in a sentence, which holds in the WebNLG corpus. The ordering of the entities and relations within a sentence is determined by a pre-order traversal of the tree.

Figure 1b shows an example of a text plan. Formally, given the input G, a text plan T is a sequences of sentence plans $T = s_1, ..., s_{N_T}$. A sentence plan s is a labeled, ordered tree, with arcs of the form (h, ℓ, m), where $h, m \in V$ are head and modifier nodes, each corresponding to an input entity, and $\ell = (r, d)$ is the relation between nodes, where $r \in R$ is the RDF relation, and $d \in \{\rightarrow, \leftarrow\}$ denotes the direction in which the relation is expressed: $d = \rightarrow$ if $(h, r, m) \in G$, and $d = \leftarrow$ if $(m, r, h) \in G$. A text plan T is said to match an input G iff every triplet (s, r, o) in G is expressed in T exactly once, either as an edge $(s, (r, \rightarrow), o)$ or as an edge $(o, (r_i, \leftarrow), s)$.

Chains $(h, \ell_1, m), (m, \ell_2, x)$ represent a succession of facts that share a middle entity (Figure 2a), while siblings — nodes with the same parent — $(h, \ell_1, m_1), (h, \ell_2, m_2)$ represents a succession of facts about the same entity (Figure 2b). Sibling and chain structures can be combined (Figure 2c). An example of an input we addressed in the WebNLG corpus, and matching text plan is given in Figure 1b.

Exhaustive generation For small-ish input graphs G—such as those in the WebNLG task we consider here—it is trivial to generate all possible plans by first considering all the ways of grouping the input into sets, then from each set generating all possible trees by arranging it as an undirected graph and performing several DFS traversals starting from each node, where each DFS traversal follows a different order of children.[3]

3.1 Adding Plans to Training Data

While the input RDFs and references are present in the training dataset, the plans are not. We devise a method to recover the latent plans for most of the input-reference pairs in the training set, constructing a new dataset of (G, ref, T) triplets of inputs, reference texts, and corresponding plans.

We define the reference ref, and the text-plan T to be consistent with each other iff (a) they exhibit the same splitting into sentences—the facts in every sentence in ref are grouped as a sentence plan in T, and (b) for each corresponding sentence and sentence-plan, the order of the entities is identical.

The matching of plans to references is based on the observations that (a) it is relatively easy to identify entities in the reference texts, and a pair of entities in an input is unique to a fact; (b) it is relatively easy to identify sentence splits; (c) a reference text and its matching plan must share the same entities in the same order, and with the same sentence splits.

Sentence split consistency We define a set of triplets to be *potentially consistent* with a sentence iff each triplet contains at least one entity from the sentence (either its subject or object appear in the sentence), and each entity in the sentence is covered by at least one triplet. Given a reference text, we split it into sentences using NLTK (Bird and Loper, 2004), and look for divisions of G into disjoint sets such that each set is consistent with a corresponding sentence. For each such division, we consider the exhaustive set of all induced plans.

Facts order consistency A natural criterion would be to consider a reference sentence and a sentence-plan originating from the corresponding RDF as matching iff the sets of entities in the sentence and the plan are identical, and all entities appear in the same order.[4] Based on this, we could represent each sentence and each plan as a sequence of entities, and verify the sequences match.

However, using this criterion is complicated by the fact that it is not trivial to map between the entities in the plan (that originate from the RDF triplets) and the entities in the text. In particular, due to language variability, the same plan entity may appear in several forms in the textual sentences. Some of these variations (i.e. "A.F.C Fylde" vs. "AFC Fylde") can be recognized heuristically, while others require external knowledge ("UK conservative party" vs. "the Tories"), and some are ambiguous and require full-fledged co-reference resolution ("them", "he", "the former"). Hence, we relax our matching criterion to allow for possible unrecognized entities in the text.

Concretely, we represent each sentence plan as a sequence of its entities $(pe_1, ..., pe_k)$, and each sentence as the sequence of its entities which we managed to recognize and to match with an input entity $(se_1, ..., se_m), m \leq k$.[5]

We then consider a sentence and a sentence-plan to be *consistent* if the following two condi-

---

[3] If a graph includes a cycle (0.4% of the graphs in the WebNLG corpus contain cycles) we skip it, as it is guaranteed that a different split will result in cycle-free graphs.

[4] An additional constraint is that no two triplets in the RDFs set share the same entities. This is to ensure that if two entities appeared in a structure, only one relation could have been expressed there. This almost always holds in the WebNLG corpus, failing on only 15 out of 6,940 input sets.

[5] We match plan entities to sentence entities using greedy string matching with Levenshtein distance (Levenshtein, 1966) for each token and a manually tuned threshold for a match. While this approach results in occasional false positives, most cases are detected correctly. We match dates by using the chrono-python package that parses dates from natural language texts.

tions hold: (1) The sentence entities $(se_1, ..., se_m)$ are a proper sub-sequence of the plan entities $(pe_1, ..., pe_k)$; and (2) each of the remaining entities in the plan already appeared previously in the plan. The second condition accounts for the fact that most un-identified entities are due to pronouns and similar non-lexicalized referring expressions, and that these only appear after a previous occurrence of the same entity in the text.[6]

3.2 Test-time Plan Selection

To select the plan to be realized, we propose a mechanism for ranking the possible plans. Our plan scoring method is a product-of-experts model, where each expert is a conditional probability estimate for some property of the plan. The conditional probabilities are MLE estimates based on the plans in the training set constructed in section 3.1. Estimates involving relation names are smoothed using Lidstone smoothing to account for unseen relations. We use the following experts:

Relation direction For every relation $r \in R$, we compute its probability to be expressed in the plan in its original order $(d = \rightarrow)$ or in the reverse order $(d = \leftarrow)$: $p_{dir}(d = \rightarrow | R)$. This captures the tendency of certain relations to be realized in the reversed order to how they are defined in the knowledge base. For example, in the WebNLG corpus the relation "manager" is expressed as a variation of "is managed by" instead of one of "is the manager of" in 68% of its occurrences $(p_{dir}(d = \leftarrow |\text{manager}) = 0.68)$.

Global direction We find that while the probability of each relation to be realized in a reversed order is usually below 0.5, still in most plans of longer texts there are one or two relations that appear in the reversed order. We capture this tendency using an expert that considers the conditional probability $p_{gd}(nr = n| \ |G|)$ of observing n reversed edges in an input with $|G|$ triplets.

Splitting tendencies For each input size, we keep track of the possible ways in which the set of facts can be split to subsets of particular sizes. That is, we keep track of probabilities such as $p_s(s = [3, 2, 2] \mid 7)$ of realizing an input of 7 RDF triplets as three sentences, each realizing the corresponding number of facts.

Relation transitions We consider each sentence plan as a sequence of the relation types expressed in it $r_1, \ldots, r_k$ followed by an EOS symbol, and compute the markov transition probabilities over this sequence: $p_{trans}(r_1, r_2, \ldots, r_k, EOS) = \prod_{i=1,k} p_t(r_{i+1}|r_i)$. The expert is the product of the transition probabilities of the individual sentence plans in the text plan. This captures the tendencies of relations to follow each other and in particular, the tendencies of related relations such as birth-place and birth-date to group, allowing their aggregation in the generated text (*John was born in London on Dec 12th, 1980*).

Each of the possible plans are then scored based on the product of the above quantities.[7]

The scores work well for separating good from lousy text plans, and we observe a threshold above which most generated plans result in adequate texts. We demonstrate in Section 6 that realizing highly-ranked plans manages to obtain good automatic realization scores. We note that the plan in Figure 1b is the one our ranking algorithm ranked first for the input in Figure 1a.

Possible Alternatives In addition to the single plan selection, the explicit planning stage opens up additional possibilities. Instead of choosing and realizing a single plan, we can realize **a diverse set** of high-scoring plans, or realizing a random high-scoring plan, resulting in a diverse and less templatic set of texts across runs. This relies on the combination of two factors: the ability of the scoring component to select plans that correspond to plausible human-authored texts, and the ability of the neural realizer to faithfully realize the plan into fluent text. While it is challenging to directly evaluate the plans adequacy, we later show an evaluation of the plan realization component. Figure 3 shows three random plans for the same graph and their realizations. Further examples of the diversity of generation are given in the appendix.

The explicit and symbolic planning stage also allows for **user control** over the generated text, either by supplying constraints on the possible plans (e.g., number of sentences, entities to focus on, the order of entities/relations, or others) or by supplying complete plans. We leave these options for future work.

---

[6] A sensible alternative would be to use a coreference resolution system at this stage. In our case it turned out to not help, and even performed somewhat worse.

[7] We note that for an input of n triplets, there are $\mathcal{O}(2^{2n} + n * n!)$ possible plans, making this method prohibitive for even moderately sized input graphs. However, it is sufficient for the WebNLG dataset in which $n \leq 7$. For larger graphs, better plan scoring and more efficient search algorithms should be devised. We leave this for future work.

(a) **Dessert** ← course [**Bionico** → country [**Mexico**] → ingredient [**Granola**] → region [**Jalisco**]]
*The **Dessert** **Bionico** requires **Granola** as one of its ingredients and originates from the **Jalisco** region of **Mexico** .*

(b) **Bionico** → country [**Mexico**] → region [**Jalisco**] .
Dessert ← course [**Bionico** → ingredient [**Granola**]]
***Bionico** is a food found in the **Mexico** region **Jalisco**.*
*The **Dessert** **Bionico** requires **Granola** as an ingredient.*

(c) **Bionico** → ingredient [**Granola**] → course [**Dessert**] .
Bionico → region [**Jalisco**] → country [**Mexico**]
***Bionico** contains **Granola** and is served as a **Dessert**.*
Bionico** is a food found in the region of **Jalisco**, **Mexico

Figure 3: Three random linearized plans for the same input graph, and their text realizations. All taken from the top 10% scoring plans. (a) structures the output as a single sentence, while (b) and (c) as two sentences. The second sentence in (b) puts emphasis on Bionico being a dessert, while in (c) the emphasis is on the ingredients.

4 Plan Realization

For plan realization, we use an off-the-shelf vanilla neural machine translation (NMT) system to translate plans to texts. The explicit division to sentences in the text plan allows us to realize each sentence plan individually which allows the realizer to follow the plan structure within each (rather short) sentence, reducing the amount of information that the model needs to remember. As a result, we expect a significant reduction in over- and under-generation of facts, which are common when generating longer texts. Currently, this comes at the expense of not modeling discourse structure (i.e., referring expressions). This deficiency may be handled by integrating the discourse into the text plan, or as a post-processing step.[8]. We leave this for future work.

To use text plans as inputs to the NMT, we linearize each sentence plan by performing a pre-order traversal of the tree, while indicating the tree structure with brackets (Figure 1c). The directed relations (r, d) are expressed as a sequence of two or more tokens, the first indicating the direction and the rest expressing the relation.[9] Entities that are identified in the reference text are replaced with single, entity-unique tokens. This allows the NMT system to copy such entities from the input rather than generating them. Figure 1d is an example of possible text resulting from such linearization.

Training details We use a standard NMT setup with a copy-attention mechanism (Gulcehre et al., 2016)[10] and the pre-trained GloVe.6B word embeddings[11] (Pennington et al., 2014). The pre-trained embeddings are used to initialize the relation tokens in the plans, as well as the tokens in the reference texts.

Generation details We translate each sentence plan individually. Once the text is generated, we replace the entity tokens with the full entity string as it appears in the input graph, and lexicalize all dates as *Month DAY+ordinal, YEAR* (i.e., *July 4th, 1776*) and for numbers with units (i.e., *"5"(minutes)*) we remove the parenthesis and quotation marks (*5 minutes*).

5 Experimental Setup

The WebNLG challenge (Colin et al., 2016) consists of mapping sets of RDF triplets to text including referring expression generation, aggregation, lexicalization, surface realization, and sentence segmentation. It contains sets with up to 7 triplets each along with one or more reference texts for each set. The test set is split into two parts: *seen*, containing inputs created for entities and relations belonging to DBpedia categories that were seen in the training data, and *unseen*, containing inputs extracted for entities and relations belonging to 5 unseen categories. While the unseen category is conceptually appealing, we view the seen category as the more relevant setup: generating fluent, adequate and diverse text for a mix of known relation types is enough of a challenge also without requiring the system to invent verbalizations for unknown relation types. Any realistic generation system could afford to provide at least a few verbalizations for each relation of interest. We thus focus our attention mostly on the seen case (though our system does also perform well on the unseen case).

Following Section 3.1, we manage to match a

---

[8] Minimally, each entity occurrence can keep track of the number of times it was already mentioned in the plan. Other alternatives include using a full-fledged referring expression generation system such as NeuralREG (Ferreira et al., 2018)

[9] We map DBPedia relations to sequences of tokens by splitting on underscores and CamelCase.

[10] Concretely, we use the OpenNMT toolkit (Klein et al., 2017) with the `copy_attn` flag. Exact parameter values are detailed in the appendix.

[11] `nlp.stanford.edu/data/glove.6B.zip`

consistent plan for 76% of the reference texts and use these plan-text pairs to train the plan realization NMT component. Overall, the WebNLG training set contains 18,102 RDF-text pairs while our plan-enhanced corpus contains 13,828 plan-text pairs.[12]

Compared Systems We compare to the best submissions in the WebNLG challenge (Gardent et al., 2017): Melbourne, an end-to-end system that scored best on all categories in the automatic evaluation, and UPF-FORGe (Mille et al., 2017), a classic grammar-based NLG system that scored best in the human evaluation.

Additionally, we developed an end-to-end neural baseline which outperforms the WebNLG neural systems. It uses a set encoder, an LSTM (Hochreiter and Schmidhuber, 1997) decoder with attention (Bahdanau et al., 2014), a copy-attention mechanism (Gulcehre et al., 2016) and a neural checklist model (Kiddon et al., 2016), as well as applying entity dropout. The entity-dropout and checklist component are the key differentiators from previous systems. We refer to this system as **StrongNeural**.

6 Experiments and Results

6.1 Automatic Metrics

We begin by comparing our plan-based system (**BestPlan**) to the state-of-the-art using the common automatic metrics: BLEU (Papineni et al., 2002), Meteor (Banerjee and Lavie, 2005), ROUGE$_L$ (Lin, 2004) and CIDEr (Vedantam et al., 2015), using the nlg-eval[13] tool (Sharma et al., 2017) on the entire test set and on each part separately (seen and unseen).

In the original challenge, the best performing system in automatic metric was based on end-to-end NMT (**Melbourne**). Both the **StrongNeural** and **BestPlan** systems outperform all the WebNLG participating systems on all automatic metrics (Table 1). **BestPlan** is competitive with **StrongNeural** in all metrics, with small differences either way per metric.[14]

	BLEU	**METEOR**	**ROUGE$_\mathrm{L}$**	**CIDEr**
UPF-FORGe$^{\heartsuit}$	38.5	0.390	60.9	2.500
Melbourne$^{\diamondsuit}$	45.0	0.376	63.5	2.814
RandomPlan-1$^{\spadesuit}$	43.3	0.384	57.6	2.342
RandomPlan-2$^{\spadesuit}$	43.5	0.384	57.4	2.332
RandomPlan-3$^{\spadesuit}$	43.5	0.384	57.4	2.303
StrongNeural$^{\diamondsuit}$	46.5	**0.392**	**65.4**	**2.866**
BestPlan$^{\spadesuit}$	**47.4**	0.391	63.1	2.692

Table 1: Results for all categories. Team color indicates the type of system used (NMT$^{\diamondsuit}$, Rule-Based$^{\heartsuit}$, Rule-Based + NMT$^{\spadesuit}$).

6.2 Manual Evaluation

Next, we turn to manually evaluate our system's performance regarding faithfulness to the input on the one hand and fluency on the other. We describe here the main points of the manual evaluation setup, with finer details in the appendix.

Faithfulness As explained in Section 3, the first benefit we expect of our plan-based architecture is to make the neural systems task simpler, helping it to remain faithful to the semantics expressed in the plan which in turn is guaranteed to be faithful to the original RDF input (by faithfulness, we mean expressing all facts in the graph and only facts from the graph: not dropping, repeating or hallucinating facts). We conduct a manual evaluation over the seen portion of the WebNLG human evaluated test set (139 input sets). We compare **BestPlan** and **StrongNeural**.[15] For each output text, we manually mark which relations are expressed in it, which are omitted, and which relations exist with the wrong lexicalization. We also count the number of relations the system over generated, either repeating facts or inventing new facts.[16]

Table 2 shows the results. **BestPlan** reduces all error types compared to **StrongNeural**, by 85%, 56% and 90% respectively. While on-par regarding automatic metrics, **BestPlan** substantially outperforms the new state-of-the-art end-to-end neural system in semantic faithfulness.

For example, Figure 4 compares the output of

---

[12] Note that this only affects the training stage. At test time, we do not require gold plans, and evaluate on all sentences.

[13] `https://github.com/Maluuba/nlg-eval`

[14] At least part of the stronger results for **StrongNeural** can be attributed to its ability to generate referring expressions, which we currently do not support.

[15] We do not evaluate **UPF-FORGe** as it is a verifiable grammar-based system that is fully faithful by design.

[16] This evaluation was conducted by the first author, on a set of shuffled examples from the **BestPlan** and **StrongNeural** systems, without knowing which outputs belongs to which system. We further note that evaluating for faithfulness requires careful attention to detail (making it less suitable for crowd-workers), but has a precise task definition which does not involve subjective judgment, making it possible to annotate without annotator biases influencing the results. We release our judgments for this stage together with the code.

1. William_Anders | dateOfRetirement | ”1969-09-01”
2. William_Anders | was selected by NASA | 1963
3. William_Anders | timeInSpace | ”8820.0”(minutes)
4. William_Anders | birthDate | ”1933-10-17”
5. William_Anders | occupation | Fighter_pilot
6. William_Anders | birthPlace | British_Hong_Kong
7. William_Anders | was a crew member of | Apollo_8

(a) The last RDF in the seen test-set

William Anders was born on **October 17th, 1933** in **British Hong Kong**.
He was selected by nasa in **1963** and became a crew member on the **Apollo 8** flight mission.
He retired on **September 1st, 1969**.

(b) Output from StrongNeural

William Anders was a **fighter pilot** who joined nasa in **1963** and served as a crew member of **Apollo 8**.
William Anders retired on **September 1st, 1969** and spent **8820.0 minutes** in space.
William Anders was born in **British Hong Kong** on october **October 17th, 1933**.

(c) Output from BestPlan

Figure 4: Comparing end-to-end neural generation with our plan based system.

StrongNeural (4b) and **BestPlan** (4c) on the last input in the seen test set (4b). While both systems chose three sentences split and aggregated details about birth in one sentence and details about the occupation in another, **StrongNeural** also expressed the information in chronological order. However, **StrongNeural** failed to generate facts 3 and 5. **BestPlan** made a lexicalization mistake in the third sentence by expressing “October” before the actual date, which is probably caused by faulty entity matching for one of the references, and (by design) did not generate any referring expression, which we leave for future work.

	BestPlan	StrongNeural
Expressed	417	360
Omitted	6	41
Wrong-lexicalization	17	39
Over-generation	3	29

Table 2: Semantic faithfulness of each system regarding 440 RDF triplets from 139 input sets in the seen part of the manually evaluated test set.

Fluency Next, we assess whether our systems succeed at maintaining the high-quality fluency of the neural systems. We perform pairwise evaluation via Amazon Mechanical Turk wherein each task the worker is presented with an RDF set (both in a graph form, and textually), and two texts in random order, one from **BestPlan**, the other from a competing system. We compare **BestPlan** against a strong end-to-end neural system (**StrongNeural**), a grammar-based system which is the state-of-the-art in human evaluation (**UPF-FORGe**), and the human-supplied WebNLG references (**Reference**). The workers were presented with three possible answers: **BestPlan** text is better (scored as 1), the other text is better (scored as -1), and both texts are equally fluent (scored as 0). Table 3 shows the average worker score given to each pair divided by the number of texts compared. **BestPlan** performed on-par with **StrongNeural**, and surpassed the previous state-of-the-art **UPF-FORGe**. It, however, scored worse than the reference texts, which is expected given that it does not produce referring expressions. Our approach manages to keep the same fluency level typical to end-to-end neural systems, thanks to the NMT realization component.

	StrongNeural	Reference	UPF-FORGe
BestPlan	-0.6%	-5.4%	+5.1%

Table 3: MTurk average worker score for *BestPlan* compared to each system. It is a worse than the reference texts, on-par with the neural end-to-end system, and a better than the previous state-of-the-art.

6.3 Plan Realization Consistency

We test the extent to which the realizer generates texts that are consistent with the plans. For several subsets of ranked plans (best plan, top 1%, and top 10%) for the seen and unseen test sets separately, we realize up to 100 randomly selected text-plans per input. We realize each sentence plan and evaluate using two criteria: (1) Do all entities from the plan appear in the realization; (2) Like the consis-

	Best Plan		Top 1% Plans		Top 10% Plans	
	Entities	**Order**	**Entities**	**Order**	**Entities**	**Order**
Seen	98.9%	100%	95.9%	99.9%	93.6%	100%
Unseen	66.7%	100%	45.3%	100%	41.3%	100%

Table 4: Surface realizer performance. *Entities:* Percent of sentence plans that were realized with all the requested entities. *Order*: of the sentences that were realized with all requested entities, percentage of realizations that followed the requested entity order.

tency we defined above, do all entities appear in the same order in the plan and the realization.

Table 4 indicates that for decreasingly probable plans our realizer does worse in the first criterion. However, for both parts of the test set, if the realizer managed to express all of the entities, it expressed them in the requested order, meaning the outputs are consistent with plans. This opens up a potential for user control and diverse outputs, by choosing different plans for realization.

Finally, we verify that the realization of potentially diverse plans is not only consistent with each given plan but also preserves output quality. For each input, we realize a random plan from the top 10%. We repeat this process three times with different random seeds to generate different outputs, and mark these systems as **RandomPlan-1/2/3**. Table 1 shows that these random plans maintain decent quality on the automatic metrics, with a limited performance drop, and the automatic score is stable across random seeds.[17]

7 Related Work

Text planning is a major component in classic NLG. For example, Stent et al. (2004) shows a method of producing coherent sentence plans by exhaustively generating as many as 20 sentence plan trees for each document plan, manually tagging them, and learning to rank them using the RankBoost algorithm (Schapire, 1999). Our planning approach is similar, but we only have a set of "good" reference plans without internal ranks. While the sentence planning decides on the aggregation, one crucial decision left is sentence order. We currently determine order based on a splitting heuristic which relies on the number of facts in every sentence, not on the content. Lapata (2003) devised a probabilistic model for sentence ordering which correlated well with human ordering. Our plan selection procedure is admittedly simple, and can be improved by integrating insights from previous text planning works (Barzilay and Lapata, 2006; Konstas and Lapata, 2012, 2013).

Many generation systems (Gardent et al., 2017; Dušek et al., 2018) are based on a black-box NMT component, with various pre-processing transformation of the inputs (such as delexicalization) and outputs to aid the generation process.

Generation from structured data often requires referring to a knowledge base (Mei et al., 2015; Kiddon et al., 2016; Wen et al., 2015). This led to input-coverage tracking neural components such as the checklist model (Kiddon et al., 2016) and copy-mechanism (Gulcehre et al., 2016). Such methods are effective for ensuring coverage and reducing the number of over-generated facts and are in some ways orthogonal to our approach. While our explicit planning stage reduces the amount of over-generation, our realizer may be further improved by using a checklist model.

More complex tasks, like RotoWire (Wiseman et al., 2017) require modeling also document-level planning. Puduppully et al. (2018) explored a method to explicitly model document planning using the attention mechanism.

The neural text generation community has also recently been interested in "controllable" text generation (Hu et al., 2017), where various aspects of the text (often sentiment) are manipulated (Ficler and Goldberg, 2017) or transferred (Shen et al., 2017; Zhao et al., 2017; Li et al., 2018). In contrast, like in (Wiseman et al., 2018), here we focused on controlling either the content of a generation or the way it is expressed by manipulating the sentence plan used in realizing the generation.

8 Conclusion

We proposed adding an explicit symbolic planning component to a neural data-to-text NLG system, which eases the burden on the neural component concerning text structuring and fact tracking. Consequently, while the plan-based system performs on par with a strong end-to-end neural system regarding automatic evaluation metrics and human fluency evaluation, it substantially outperforms the end-to-end system regarding faithfulness to the input. Additionally, the planning stage allows explicit user-control and generating diverse sentences, to be pursued in future work.

---

[17] While the scores for the different sets are very similar, the plans are very different from each other. See for examples the plans in Figure 3.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Regina Barzilay and Mirella Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Emilie Colin, Claire Gardent, Yassine Mrabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The webnlg challenge: Generating text from dbpedia data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 163–167.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the e2e nlg challenge. *arXiv preprint arXiv:1810.01170*.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018. Neuralreg: An end-to-end approach to referring expression generation. *arXiv preprint arXiv:1805.08093*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *CoRR*, abs/1703.09902.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 545–552. Association for Computational Linguistics.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. Forge at semeval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 920–923.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. *arXiv preprint arXiv:1809.00582*.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Robert E Schapire. 1999. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2017. Adversarially regularized autoencoders for generating discrete structures. *CoRR*, abs/1706.04223.

Evaluating Rewards for Question Generation Models

Tom Hosking
University College London
`thomas.hosking.17@ucl.ac.uk`

Sebastian Riedel
University College London
`sriedel@ucl.ac.uk`

Abstract

Recent approaches to question generation have used modifications to a Seq2Seq architecture inspired by advances in machine translation. Models are trained using teacher forcing to optimise only the one-step-ahead prediction. However, at test time, the model is asked to generate a whole sequence, causing errors to propagate through the generation process (exposure bias). A number of authors have suggested that optimising for rewards less tightly coupled to the training data might counter this mismatch. We therefore optimise directly for various objectives beyond simply replicating the ground truth questions, including a novel approach using an adversarial discriminator that seeks to generate questions that are indistinguishable from real examples. We confirm that training with policy gradient methods leads to increases in the metrics used as rewards. We perform a human evaluation, and show that although these metrics have previously been assumed to be good proxies for question quality, they are poorly aligned with human judgement and the model simply learns to exploit the weaknesses of the reward source.

1 Introduction

Posing questions about a document in natural language is a crucial aspect of the effort to automatically process natural language data, enabling machines to ask clarification questions (Saeidi et al., 2018), become more robust to queries (Yu et al., 2018), and to act as automatic tutors (Heilman and Smith, 2010).

Recent approaches to question generation have used Seq2Seq (Sutskever et al., 2014) models with attention (Bahdanau et al., 2014) and a form of copy mechanism (Vinyals et al., 2015; Gulcehre et al., 2016). Such models are trained to generate a plausible question, conditioned on an input document and answer span within that document (Zhou et al., 2018; Du et al., 2017; Du and Cardie, 2018; Yuan et al., 2017).

There are currently no dedicated question generation datasets, and authors have used the context-question-answer triples available in SQuAD. Only a single question is available for each context-answer pair, and models are trained using teacher forcing (Williams and Zipser, 1989). This lack of diverse training data combined with the one-step-ahead training procedure exacerbates the problem of exposure bias (Ranzato et al., 2015). The model does not learn how to distribute probability mass over sequences that are valid but different to the ground truth; during inference, the model must predict the whole sequence, and may not be robust to mistakes during decoding.

Recent work has investigated training the models directly on a performance based objective, either by optimising for BLEU score (Kumar et al., 2018a) or other quality metrics (Yuan et al., 2017). By decoupling the training procedure from the ground truth data, the model is able to explore the space of possible questions and learn to recover from suboptimal predictions during decoding. While the metrics used seem to be intuitively good choices, there is an assumption that they are good proxies for question quality which has not yet been confirmed.

Our contributions are as follows. We perform fine tuning using a range of rewards, including a novel adversarial objective that directly estimates the probability that a question was generated or came from the ground truth data. We show that although fine tuning leads to increases in reward scores, the resulting models perform worse when evaluated by human workers. We also demonstrate that the generated questions exploit weaknesses in the reward models.

Proceedings of NAACL-HLT 2019, pages 2278–2283
Minneapolis, Minnesota, June 2 - June 7, 2019.

Context

although united methodist practices and interpretation of beliefs have evolved over time , these practices and beliefs can be traced to the writings of the church 's founders , especially **john wesley and charles wesley** (anglicans) , but also philip william otterbein and martin boehm (united brethren) , and jacob albright (evangelical association) .

Rewards	**Output**
Ground Truth Question	who were two of the founders of the united methodist church ?
No fine tuning	which two methodist can be traced to the church 's founders ?
LM	according to the writings of the church 's founders , according to the writings of the church 's founders , [...]
QA	who in anglicans ?
LM and QA	who are the writings of the church 's founders ?
Discriminator	who founded the church 's founders ?
Adversarial discriminator	who were two western methodist practices ?
LM, QA and adversarial discriminator	who are the anglicans of the church ?

Table 1: Example generated questions for various fine-tuning objectives. The answer is highlighted in bold. The model trained on a QA reward has learned to simply point at the answer and exploit the QA model, while the model trained on a language model objective has learned to repeat common phrase templates.

2 Background

Many of the advances in natural language generation have been led by machine translation (MT) (Sutskever et al., 2014; Bahdanau et al., 2014; Gulcehre et al., 2016).

Previous work on question generation has made extensive use of MT techniques. Du et al. (2017) use a Seq2Seq based model to generate questions conditioned on context-answer pairs, and build on this work by preprocessing the context to resolve coreferences and adding a pointer network (Du and Cardie, 2018). Similarly, Zhou et al. (2018) use a part-of-speech tagger to augment the embedding vectors. Both authors perform a human evaluation of their models, and show significant improvement over their baseline. Kumar et al. (2018a) use a similar model, but apply it to the task of generating questions without conditioning on a specific answer span. Song et al. (2018) use a modified context encoder based on multi-perspective context matching (Wang et al., 2016).

Kumar et al. (2018b) propose a framework for fine tuning using policy gradients and perform a human evaluation showing promising results. However, they use as rewards various similarity metrics that are still coupled to the ground truth. Yuan et al. (2017) describe a Seq2Seq model with attention and a pointer network, with an additional encoding layer for the answer. They also describe a method for further tuning their model using policy gradients, with rewards given by an external language model and question answering (QA) system. Unfortunately they do not perform any human evaluation to determine whether this tuning led to improved question quality.

For the related task of summarisation, Paulus et al. (2017) propose a framework for fine tuning a summarisation model using reinforcement learning, with the ROUGE similarity metric used as the reward.

3 Experimental setup

The task is to generate a natural language question, conditioned on a document and the location of an answer within that document. For example, given the input document “this paper investigates rewards for question generation” and answer “question generation”, the model should produce a question such as “what is investigated in the paper?”

3.1 Model description

We use the model architecture described by Yuan et al. (2017). Briefly, this is a Seq2Seq model (Sutskever et al., 2014) with attention (Bahdanau et al., 2014) and copy mechanism (Vinyals et al., 2015; Gulcehre et al., 2016). Yuan et al. (2017) also add an additional answer encoder layer, and initialise the decoder with a hidden state constructed from the final state of the encoder. Beam search (Graves, 2012) is used to sample from the model at inference time. We train the model using maximum likelihood before fine tuning. Our implementation achieves a BLEU-4 score (Papineni et al., 2002) of 13.5 on the test set used by Du et al. (2017), before fine tuning.

Features				Metrics				
QA reward	LM reward	Discriminator reward	Adversarial discriminator	NLL	BLEU	QA	LM	Discriminator
-	✓	-	-	-0.7	-1.9	-3.7	-13.4	+1.5
✓	-	-	-	+1.7	-4.5	**+3.9**	+226	+5.4
✓	✓	-	-	-0.5	-2.6	+2.0	**-16.3**	+2.9
-	-	✓	-	**-0.8**	-1.8	-2.1	-9.4	+2.5
-	-	✓	✓	+6.4	-2.7	-2.5	-1.0	**+10.8**
✓	✓	✓	✓	+1.0	-2.4	+1.3	-6.2	+10.0

Table 2: Changes in automatic evaluation metrics after models were fine tuned on various objectives. QA refers to the F1 score obtained by a question answering system on the generated questions. LM refers to the perplexity of generated questions under a separate language model. The discriminator reward refers to the percentage of generated sequences that fooled the discriminator. Lower LM and NLL scores are better. BLEU scores decreased in all cases.

Model	Fluency	Relevance
No fine tuning	**3.34**	**3.12**
+QA, LM rewards	3.05	2.75
+QA, LM, discriminator rewards +Adversarial discriminator	2.89	2.82
Ground Truth	4.67	4.72

Table 3: Summary of human evaluation of selected models

3.2 Fine tuning

Generated questions should be formed of language that is both *fluent* and *relevant* to the context and answer. Following (Yuan et al., 2017), we perform fine tuning on a trained model, using rewards given either by the negative perplexity under a LSTM language model, or the F1 score attained by a question answering (QA) system, or a weighted combination of both. The language model is a standard recurrent neural network formed of a single LSTM layer. For the QA system, we use QANet (Yu et al., 2018) as implemented by Kim (2018).

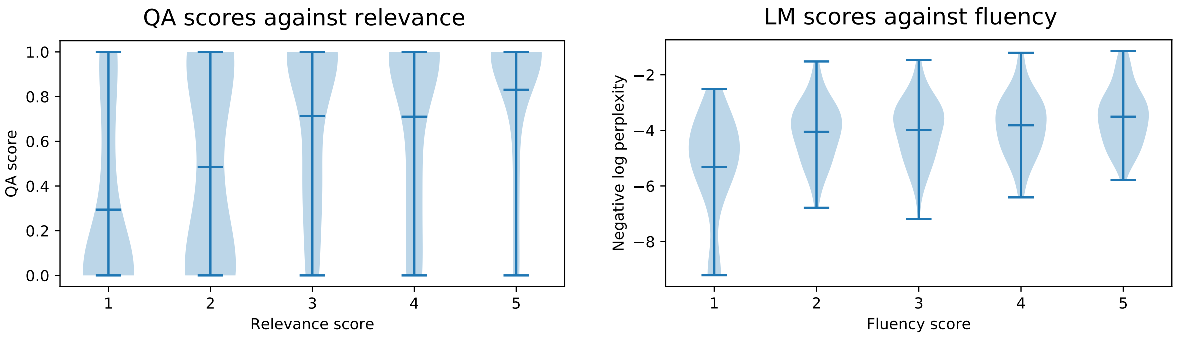
3.3 Adversarial training

Additionally, we propose a novel approach by learning the reward directly from the training data, using a *discriminator* detailed in Appendix A. We generate questions for each context-answer pair in the training set using a generator trained by maximum likelihood, and train the discriminator to predict whether an input question was generated by our model, or originated from the training data. Keeping the discriminator fixed, we then fine-tune the generator, using as reward the probability estimated by the discriminator that a generated question was in fact real. In other words, the generator is rewarded for successfully fooling the discriminator. We also experiment with interleaving updates to the discriminator within the fine tuning phase, allowing the discriminator to become adversarial and adapt alongside the generator.

The rewards described above are used to update the model parameters via the REINFORCE policy gradient algorithm (Williams, 1992). We teacher force the decoder with the generated sequence to reproduce the activations calculated during beam search, to enable backpropagation. All rewards are normalised with a simple form of PopArt (Hasselt et al., 2016), with the running mean μ_R and standard deviation σ_R updated online during training. We continue to apply a maximum likelihood training objective during this fine tuning.

3.4 Evaluation

We report the negative log-likelihood (NLL) of the test set under the different models, as well as the corpus level BLEU-4 score (Papineni et al., 2002) of the generated questions compared to the ground truth. We also report the rewards achieved on the



(a) QA scores plotted against human relevance scores for all rated questions.

(b) LM scores plotted against human fluency scores for all rated questions.

Figure 1: Comparison of human and automatic metrics.

test set, as the QA, LM and discriminator scores.

For the human evaluation, we follow the standard approach in evaluating machine translation systems (Koehn and Monz, 2006), as used for question generation by Du and Cardie (2018). We ask three workers to rate 300 generated questions between 1 (poor) and 5 (good) on two separate criteria: the fluency of the language used, and the relevance of the question to the context document and answer.

4 Results

Table 2 shows the changes in automatic metrics for models fine tuned on various combinations of rewards, compared to the model without tuning. In all cases, the BLEU score reduces, as the training objective is no longer closely coupled to the training data. In general, models achieve better scores on the metrics on which they were fine tuned. Jointly training on a QA *and* LM reward results in better LM scores than training on only a LM reward; the LM score did not increase smoothly when used as the sole objective, and we believe the additional QA reward acts as a form of regularisation. We conclude that fine tuning using policy gradients can be used to attain higher rewards, as expected.

Table 3 shows the human evaluation scores for a subset of the fine tuned models. The model fine tuned on a QA and LM objective is rated as significantly worse by human annotators, despite achieving higher scores in the automatic metrics. In other words, the training objective given by these reward sources does not correspond to true question quality, despite them being intuitively good choices.

The model fine tuned using an adversarial discriminator has also failed to achieve better human ratings, with the discriminator model unable to learn a useful reward source. Although the training process was stable and robust to different initialisations, and the outputs do not appear to be significantly worse, we conclude that the discriminator was unable to learn a sufficiently useful distinction between generated and real questions, and the additional fine tuning procedure simply added unwanted noise to the model predictions.

Table 1 shows an example where fine tuning has not only failed to improve the quality of generated questions, but has caused the model to exploit the reward source. The model fine tuned on a LM reward has degenerated into producing a loop of words that is evidently deemed probable, while the model trained on a QA reward has learned that it can simply point at the location of the answer. This observation is supported by the metrics; the model fine tuned on a QA reward has suffered a catastrophic worsening in LM score of +226.

Figure 1 shows the automatic scores against human ratings for all rated questions. The correlation coefficient between human relevance and automatic QA scores was 0.439, and between fluency and LM score was only 0.355. While the automatic scores are good indicators of whether a question will achieve the lowest human rating or not, they do not differentiate clearly between the higher ratings: training a model on these objectives will not necessarily learn to generate better questions. A good question will likely attain a high QA and LM score, but the inverse is not true; a sequence may exploit the weaknesses of the metrics and achieve a high score *despite* being unintelligible to a human. We conclude that fine tuning a question generation model on these rewards does not lead to better quality questions.

5 Conclusion

In this paper, we investigate the use of external reward sources for fine tuning question generation models to counteract the lack of task-specific training data. We show that although fine tuning can be used to attain higher rewards, this does not equate to better quality questions when rated by humans. Using QA and LM rewards as a training objective causes the generator to expose the weaknesses in these models, which in turn suggests a possible use of this approach for generating adversarial training examples for QA models. The QA and LM scores are well correlated with human ratings at the lower end of the scale, suggesting they could successfully be used as part of a reranking or filtering system. We plan to research overgenerating questions and using the reward signals to rerank the outputs, thereby including the inductive bias the rewards represent without allowing the model to exploit them.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate.

Xinya Du and Claire Cardie. 2018. Harvesting Paragraph-Level Question-Answer Pairs from Wikipedia.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. pages 1342–1352.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words.

Hado Van Hasselt, Arthur Guez, Matteo Hessel, and David Silver. 2016. Learning functions across many orders of magnitudes. *arXiv*, (Nips):1–19.

Michael Heilman and Noah A Smith. 2010. Good question! Statistical ranking for question generation. *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 609–617.

Min Sang Kim. 2018. Qanet. `https://github.com/NLPLearn/QANet`.

Philipp Koehn and Christof Monz. 2006. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 102–121.

Vishwajeet Kumar, Kireeti Boorla, Yogesh Meena, Ganesh Ramakrishnan, and Yuan Fang Li. 2018a. Automating reading comprehension by generating question and answer pairs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10939 LNAI, pages 335–348.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018b. A Framework for Automatic Question Generation from Text using Deep Reinforcement Learning. Technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wj Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *… of the 40Th Annual Meeting on …*, (July):311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A DEEP REINFORCED MODEL FOR ABSTRACTIVE SUMMARIZATION. (i):1–12.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktaschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging Context Information for Natural Question Generation. pages 569–574.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *Nips*, page 9.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pages 2692–2700.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-Perspective Context Matching for Machine Comprehension.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256.

Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine Comprehension by Text-to-Text Neural Question Generation.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2018. Neural question generation from text: A preliminary study. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10619 LNAI, pages 662–671. Springer, Cham.

A Discriminator architecture

Figure 2: Discriminator architecture diagram.

We use an architecture based on a modified QANet as shown in Figure 2, replacing the output layers of the model to produce a single probability. Since the discriminator is also able to consider a full context-question-answer triple as input (as opposed to a context-question pair for the QA task), we fuse this information in the output layers.

Specifically, we apply max pooling over time to the output of the first two encoders, and we took the mean of the outputs of the third encoder that formed part of the answer span. These three reduced encodings were concatenated, a 64 unit hidden layer with ReLU activation applied, and the output passed through a single unit sigmoid output layer to give the estimated probability that an input context-question-answer triple originated from the ground truth dataset or was generated.

Text Generation from Knowledge Graphs with Graph Transformers

Rik Koncel-Kedziorski[1], Dhanush Bekal[1], Yi Luan[1], Mirella Lapata[2], and Hannaneh Hajishirzi[1,3]

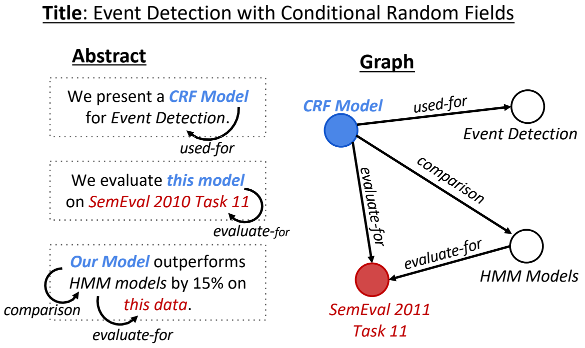
[1]University of Washington
{kedzior,dhanush,luanyi,hannaneh}@uw.edu
[2]University of Edinburgh
mlap@inf.ed.ac.uk
[3]Allen Institute for Artificial Intelligence

Abstract

Generating texts which express complex ideas spanning multiple sentences requires a structured representation of their content (document plan), but these representations are prohibitively expensive to manually produce. In this work, we address the problem of generating coherent multi-sentence texts from the output of an information extraction system, and in particular a knowledge graph. Graphical knowledge representations are ubiquitous in computing, but pose a significant challenge for text generation techniques due to their non-hierarchical nature, collapsing of long-distance dependencies, and structural variety. We introduce a novel graph transforming encoder which can leverage the relational structure of such knowledge graphs without imposing linearization or hierarchical constraints. Incorporated into an encoder-decoder setup, we provide an end-to-end trainable system for graph-to-text generation that we apply to the domain of scientific text. Automatic and human evaluations show that our technique produces more informative texts which exhibit better document structure than competitive encoder-decoder methods. [1]

1 Introduction

Increases in computing power and model capacity have made it possible to generate mostly-grammatical sentence-length strings of natural language text. However, generating several sentences related to a topic and which display overall coherence and discourse-relatedness is an open challenge. The difficulties are compounded in domains of interest such as scientific writing. Here the variety of possible topics is great (e.g. topics as diverse as driving, writing poetry, and picking stocks are all referenced in one subfield of one scientific discipline). Additionally, there are strong constraints on document structure, as scientific communication requires carefully ordered explanations of processes and phenomena.



We present a CRF Model for Event Detection.

We evaluate this model on SemEval 2010 Task 11

Our Model outperforms HMM models by 15% on this data.

Figure 1: A scientific text showing the annotations of an information extraction system and the corresponding graphical representation. Coreference annotations shown in color. Our model learns to generate texts from automatically extracted knowledge using a graph encoder decoder setup.

Many researchers have sought to address these issues by working with structured inputs. Data-to-text generation models (Konstas and Lapata, 2013; Lebret et al., 2016; Wiseman et al., 2017; Puduppully et al., 2019) condition text generation on table-structured inputs. Tabular input representations provide more guidance for producing longer texts, but are only available for limited domains as they are assembled at great expense by manual annotation processes.

The current work explores the possibility of using information extraction (IE) systems to automatically provide context for generating longer texts (Figure 1). Robust IE systems are available and have support over a large variety of textual domains, and often provide rich annotations of relationships that extend beyond the scope of

---

[1] Data and code available at https://github.com/rikdz/GraphWriter

Proceedings of NAACL-HLT 2019, pages 2284–2293
Minneapolis, Minnesota, June 2 - June 7, 2019.

a single sentence. But due to their automatic nature, they also introduce challenges for generation such as erroneous annotations, structural variety, and significant abstraction of surface textual features (such as grammatical relations or predicate-argument structure).

To effect our study, we use a collection of abstracts from a corpus of scientific articles (Ammar et al., 2018). We extract entity, coreference, and relation annotations for each abstract with a state-of-the-art information extraction system (Luan et al., 2018), and represent the annotations as a *knowledge graph* which collapses co-referential entities. An example of a text and graph are shown in Figure 1. We use these graph/text pairs to train a novel attention-based encoder-decoder model for knowledge-graph-to-text generation. Our model, GraphWriter, extends the successful Transformer for text encoding (Vaswani et al., 2017) to graph-structured inputs, building on the recent Graph Attention Network architecture (Veličković et al., 2018). The result is a powerful, general model for graph encoding which can incorporate global structural information when contextualizing vertices in their local neighborhoods.

The main contributions of this work include:

1. We propose a new graph transformer encoder that applies the successful sequence transformer to graph structured inputs.
2. We show how IE output can be formed as a connected unlabeled graph for use in attention-based encoders.
3. We provide a large dataset of knowledge-graphs paired with scientific texts for further study.

Through detailed automatic and human evaluations, we demonstrate that automatically extracted knowledge can be used for multi-sentence text generation. We further show that structuring and encoding this knowledge as a graph leads to improved generation performance compared to other encoder-decoder setups. Finally, we show that GraphWriter's transformer-style encoder is more effective than Graph Attention Networks on the knowledge-graph-to-text task.

2 Related Work

Our work falls under the larger scope of concept-to-text generation. Barzilay and Lapata (2005) introduced a collective content selection model for generating summaries of football games from tables of game statistics. Liang et al. (2009) jointly learn to segment and align text with records, reducing the supervision needed for learning. Kim and Mooney (2010) improve this technique by learning a semantic parse to logical forms. Konstas and Lapata (2013) focus on the generation objective, jointly learning planning and generating using a rhetorical (RST) grammar induction approach.

These earlier works often focused on smaller record generation datasets such as WeatherGov and RoboCup, but recently Mei et al. (2016) showed how neural models can achieve strong results on these standards, prompting researchers to investigate more challenging domains such as ours.

Lebret et al. (2016) tackles the task of generating the first sentence of a Wikipedia entry from the associated infobox. They provide a large dataset of such entries and a language model conditioned on tables. Our work focuses on a multi-sentence task where relations can extend beyond sentence boundaries.

Wiseman et al. (2017) study the difficulty of applying neural models to the data-to-text task. They introduce a large dataset where a text summary of a basketball game is paired with two tables of relevant statistics and show that neural models struggle to compete with template based methods over this data. We propose generating from graphs rather than tables, and show that graphs can be effectively encoded to capture both local and global structure in the input.

We show that modeling knowledge as a graph improves generation results, connecting our work to other graph-to-text tasks such as generating from Abstract Meaning Representation (AMR) graphs. Konstas et al. (2017) provide the first neural model for this task, and show that pretraining on a large dataset of noisy automatic parses can improve results. However, they do not directly model the graph structure, relying on linearization and sequence encoding instead. Current works improve this through more sophisticated graph encoding techniques. Marcheggiani and Perez-Beltrachini (2018) encode input graphs directly using a graph convolution encoder (Kipf and Welling, 2017). Our model extends the graph attention networks of Veličković et al. (2018), a direct descendant of the convolutional approach which offers more modeling power and has been

	Title	Abstract	KG
Vocab	29K	77K	54K
Tokens	413K	5.8M	1.2M
Entities	-	-	518K
Avg Length	9.9	141.2	-
Avg #Vertices	-	-	12.42
Avg #Edges	-	-	4.43

Table 1: Data statistics of our AGENDA dataset. Averages are computed per instance.

shown to improve performance. Song et al. (2018) uses a graph LSTM model to effect information propagation. At each timestep, a vertex is represented by a gated combination of the vertices to which it is connected and the labeled edges connecting them. Beck et al. (2018) use a similar gated graph neural network. Both of these gated models make heavy use of label information, which is much sparser in our knowledge graphs than in AMR. Generally, AMR graphs are denser, rooted, and connected, whereas the knowledge our model works with lacks these characteristics. For this reason, we focus on attention-based models such as Veličković et al. (2018), which impose fewer constraints on their input.

Finally, our work is related to Wang et al. (2018) who offer a method for generating scientific abstracts from titles. Their model uses a gated rewriter network to write and revise several draft outputs in several sequence-to-sequence steps. While we operate in the same general domain as this work, our task setup is ultimately different due to the use of extracted information as input. We argue that our setup improves the task defined in Wang et al. (2018), and our more general model can be applied across tasks and domains.

3 The AGENDA Dataset

We consider the problem of generating a text from automatically extracted information (*knowledge*). IE systems can produce high quality knowledge for a variety of domains, synthesizing information from across sentence and even document boundaries. Generating coherent text from knowledge requires a model which considers global characteristics of the knowledge as well as local characteristics of each entity. This feature of the task motivates our use of graphs for representing knowledge, where neighborhoods localize important information and paths through the graph build connections between distant nodes through intermediate ones. An example knowledge graph can be seen in Figure 1.

We formulate our problem as follows: given the title of a scientific article and a knowledge graph constructed by an automatic information extraction system, the goal is to generate an abstract that a) is appropriate for the given title and b) expresses the content of the knowledge graph in natural language text. To evaluate how well a model accomplishes this goal, we introduce the Abstract GENeration DAtaset (AGENDA), a dataset of knowledge graphs paired with scientific abstracts. Our dataset consists of 40k paper titles and abstracts from the Semantic Scholar Corpus taken from the proceedings of 12 top AI conferences (Ammar et al., 2018).

For each abstract, we create a knowledge graph in two steps. First, we apply the SciIE system of Luan et al. (2018), a state-of-the-art science-domain information extraction system. This system provides named entity recognition for scientific terms, with entity types Task, Method, Metric, Material, or Other Scientific Term. The model also produces co-reference annotations as well as seven relations that can obtain between different entities (Compare, Used-for, Feature-of, Hyponym-of, Evaluate-for, and Conjunction). For example, in Figure 1, the node labeled "SemEval 2011 Task 11" is of type 'Task', "HMM Models" is of type 'Model', and there is a 'Evaluate-For' relation showing that the models are evaluated on the task.

We form these annotations into knowledge graphs. We collapse co-referential entities into a single node associated with the longest mention (on the assumption that these will be the most informative). We then connect nodes to one another using the relation annotations, treating these as labeled edges in the graph. The result is a possibly unconnected graph representation of the SciIE annotations for a given abstract.

Statistics of the AGENDA dataset are available in Table 1. We split the AGENDA dataset into 38,720 training, 1000 validation, and 1000 test datapoints. We offer standardized data splits to facilitate comparison.

4 Model

Following most work on neural generation we adopt an encoder-decoder architecture, shown in

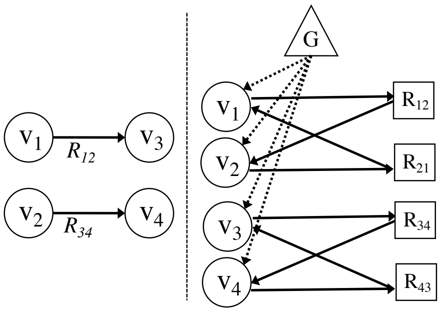


Figure 2: Converting disconnected labeled graph to connected unlabeled graph for use in attention-based encoder. v_i refer to vertices, R_{ij} to relations, and G is a global context node.

Figure 3, which we call GraphWriter. The input to GraphWriter is a title and a knowledge graph which are encoded respectively with a bidirectional recurrent neural network and a novel Graph Transformer architecture (to be discussed in Section 4.1). At each decoder time step, we attend on encodings of the knowledge graph and document title using the decoder hidden state $\mathbf{h}_t \in \mathbb{R}^d$. The resulting vectors are used to select output w_t either from the decoder's vocabulary or by copying an entity from the knowledge graph. Details of our decoding process are described in Section 4.2. The model is trained end-to-end to minimize the negative log likelihood of the mixed copy and vocabulary probability distribution and the human authored text.

4.1 Encoder

The AGENDA dataset contains a knowledge graph for each datapoint, but our model requires unlabeled, connected graphs as input. To encode knowledge graphs with this model, we restructure each graph as an unlabeled connected graph, preserving label information by the method described below and sketched in Figure 2.

Graph Preparation We convert each graph to an unlabeled connected bipartite graphs following a similar procedure to Beck et al. (2018). In this process, each labeled edge is replaced with two vertices: one representing the forward direction of the relation and one representing the reverse. These new vertices are then connected to the entity vertices so that the directionality of the former edge is maintained. This restructures the original knowledge graph as an unlabeled directed graph where all vertices correspond to entities and relations in the SciIE annotations without loss of information. To promote information flow between disconnected parts of the graph, we add a global vertex which connects all entity vertices. This global vertex will be used to initialize the decoder, analogously to the final encoder hidden state in a traditional sequence to sequence model. The final result of these restructuring operations is a connected, unlabeled graph $G = (V, E)$, where V is a list of entities, relations, and a global node and E is an adjacency matrix describing the directed edges.

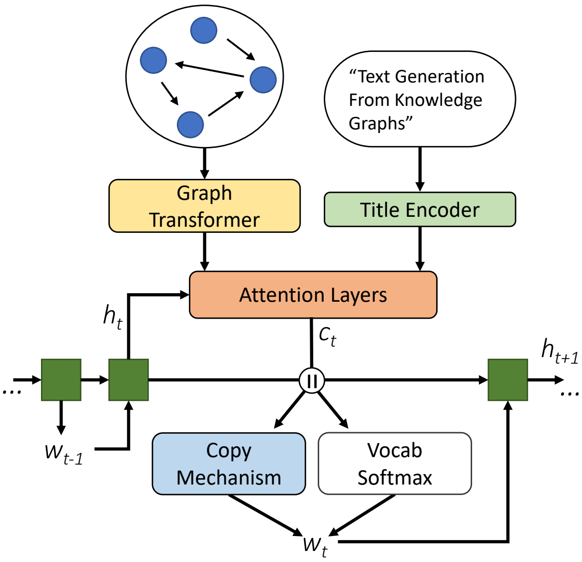


Figure 3: GraphWriter Model Overview

Graph Transformer Our model is most similar to the Graph Attention Network (GAT) of Veličković et al. (2018), which computes the hidden representations of each node in a graph by attending over its neighbors following a self-attention strategy. The use of self-attention in GAT addresses the shortcomings of prior methods based on graph convolutions (Defferrard et al., 2016; Kipf and Welling, 2017), but limits vertex updates to information from adjacent nodes. Our model allows for a more global contextualization of each vertex through the use of a transformer-style architecture. The recently proposed Transformer (Vaswani et al., 2017) addresses the inherent sequential computation shortcoming of recurrent neural networks, enabling efficient and paralleled computation by invoking a self-attention mechanism for global context modeling. These models have shown promising results in a variety of text processing tasks (Radford et al., 2018).

Our Graph Transformer encoder starts with self-

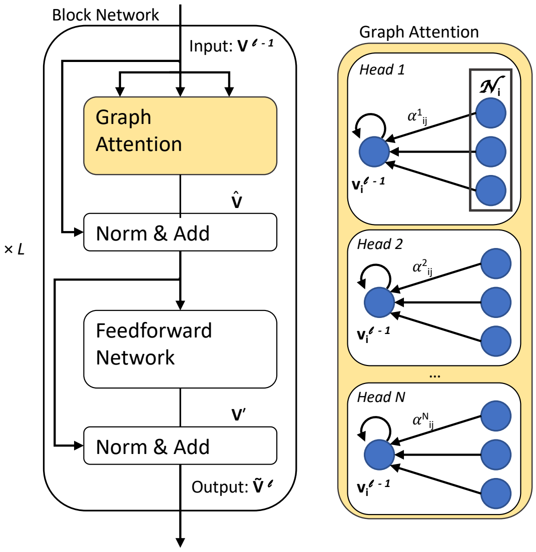


Figure 4: Graph Transformer

attention of local neighborhoods of vertices; the key difference with GAT is that our model includes additional mechanisms for capturing global context. This additional modeling power allows the Graph Transformer to better articulate how a vertex should be updated given the content of its neighbors, as well as to learn global patterns of graph structure relevant to the model's objective.

Specifically, V is embedded in a dense continuous space by the embedding process described at the end of this section, resulting in matrix $\mathbf{V}^0 = [\mathbf{v}_i], \mathbf{v}_i \in \mathbb{R}^d$ which will serve as input to the graph transformer model shown in Figure 4. Each vertex representation $\mathbf{v}_i$ is contextualized by attending over the other vertices to which v_i is connected in G. We use an N-headed self attention setup, where N independent attentions are calculated and concatenated before a residual connection is applied:

$$
\begin{aligned}
\hat{\mathbf{v}}_i &= \mathbf{v}_i + \mathop{\Big\Vert}_{n=1}^{N} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^n \mathbf{W}_V^n \mathbf{v}_j && (1) \\
\alpha_{ij}^n &= a^n(\mathbf{v}_i, \mathbf{v}_j) && (2)
\end{aligned}
$$

Here, $\Vert$ denotes the concatenation of the N attention heads, $\mathcal{N}_i$ denotes the neighborhood of v_i in G, $\mathbf{W}_V^n \in \mathbb{R}^{d \times d}$, and where a^n are attention mechanisms parameterized per head. In this work, we use attention functions of the following form:

$$
a(\mathbf{q}_i, \mathbf{k}_j) = \frac{\exp((\mathbf{W}_K \mathbf{k}_j)^\top \mathbf{W}_Q \mathbf{q}_i)}{\sum_{z \in \mathcal{N}_i} \exp((\mathbf{W}_K \mathbf{k}_z)^\top \mathbf{W}_Q \mathbf{q}_i)} \quad (3)
$$

Each a learns independent transformations $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$ of $\mathbf{q}$ and $\mathbf{k}$ respectively, and the resulting product is normalized across all connected edges. To reduce the tendency of these dot products to impede gradient flow, we scale them by $\frac{1}{\sqrt{d}}$, following Vaswani et al. (2017).

The Graph Transformer then augments these multi-headed attention layers with *block* networks. Each block applies the following transformations:

$$
\begin{aligned}
\tilde{\mathbf{v}}_i &= \text{LayerNorm}(\mathbf{v}'_i + \text{LayerNorm}(\hat{\mathbf{v}}_i)) && (4) \\
\mathbf{v}'_i &= \text{FFN}(\text{LayerNorm}(\hat{\mathbf{v}}_i)) && (5)
\end{aligned}
$$

Where $\text{FFN}(\mathbf{x})$ is a two layer feedforward network with a non-linear transformation f between layers i.e. $f(\mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$.

Stacking multiple blocks allows information to propagate through the graph. Blocks are stacked L times, with the output of layer $l-1$ taken as the input to layer l, so that $\mathbf{v}_i^l = \tilde{\mathbf{v}}_i^{l-1}$. The resulting vertex encodings $\mathbf{V}^L = [\mathbf{v}_i^L]$ represent entities, relations, and the global node contextualized by their relationships in the graph structure. We refer to the resulting encodings as *graph contextualized vertex encodings*.

Embedding Vertices, Encoding Title As stated above, the vertices of our graph correspond to entities and relations from the SciIE annotations. Because each relation is represented as both a forward- and backward-looking vertex, we learn two embeddings per relation as well as an initial embedding for the global node. Entities correspond to scientific terms which are often multi-word expressions. To produce a single d-dimensional embedding per phrase, we use the last hidden state of a bidirectional RNN run over embeddings of each word in the entity phrase, i.e. $\text{BiRNN}(\mathbf{x}_1 \ldots \mathbf{x}_m)$ for dense embeddings $\mathbf{x}$ and phrase length m. The output of our embedding step is a collection $\mathbf{V}^0$ of d-dimensional vectors representing each vertex in V.

The title input is also a short string, and so we encode it with another BiRNN to produce $\mathbf{T} = \text{BiRNN}(x'_1 \ldots x'_m)$ for title word embedding $\mathbf{x}'$.

4.2 Decoder

We decode with an attention-based decoder with a copy mechanism for copying input from the knowledge graph and title. At each decoding timestep t we use decoder hidden state $\mathbf{h}_t$ to compute context vectors $\mathbf{c}_g$ and $\mathbf{c}_s$ for the graph and title sequence respectively. $\mathbf{c}_g$ is computed using multi-headed attention contextualized by $\mathbf{h}_t$:

$$\mathbf{c}_g = \mathbf{h}_t + \Big\Vert_{n=1}^{N} \sum_{j \in V} \alpha_j^n \mathbf{W}_G^n \mathbf{v^L}_j \quad (6)$$

$$\alpha_j = a(\mathbf{h}_t, \mathbf{v^L}_j) \quad (7)$$

for a as described in Equation (1) by attending over the graph contextualized encodings $\mathbf{V}^L$. $\mathbf{c}_s$ is computed similarly, attending over the title encoding $\mathbf{T}$. We then construct the final context vector by concatenation, $\mathbf{c}_t = [\mathbf{c}_g \Vert \mathbf{c}_s]$. We use an input-feeding decoder (Luong et al., 2015) where both $\mathbf{h}_t$ and $\mathbf{c}_t$ are passed as input to the next RNN timestep.

We compute a probability p of copying from the input using $\mathbf{h}_t$ and $\mathbf{c}_t$ in a fashion similar to See et al. (2017), that is:

$$p = \sigma(\mathbf{W}_{copy}[\mathbf{h}_t \Vert \mathbf{c}_t] + b_{copy}) \quad (8)$$

The final next-token probability distribution is:

$$p * \alpha^{copy} + (1-p) * \alpha^{vocab}, \quad (9)$$

Where the probability distribution α^{copy} over entities and input tokens is computed as $\alpha_j^{copy} = a([\mathbf{h}_t \Vert \mathbf{c}_t], \mathbf{x}_j)$ for $\mathbf{x}_j \in \mathbf{V} \Vert \mathbf{T}$. The remaining $1-p$ probability is given to α^{vocab}, which is calculated by scaling $[\mathbf{h}_t \Vert \mathbf{c}_t]$ to the vocabulary size and taking a softmax.

5 Experiments

Evaluation Metrics We evaluate using a combination of human and automatic evaluations. For **human** evaluation, participants were asked to compare abstracts generated by various models and those written by the authors of the scientific articles. We used Best-Worst Scaling (BWS; (Louviere and Woodworth, 1991; Louviere et al., 2015)), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales (Kiritchenko and Mohammad, 2016). Participants were presented with two or three abstracts and asked to decide which one was better and which one was worse in order of grammar and fluency (is the abstract written in well-formed English?), coherence (does the abstract have an introduction, state the problem or task, describe a solution, and discuss evaluations or results?), and informativeness (does the abstract relate to the provided title and make use of appropriate scientific terms?). We provided examples of good and bad abstracts and explain how they succeed or fail to meet the defined criteria.

Because our dataset is scientific in nature, evaluations must be done by experts and we can only collect a limited number of these high quality datapoints.[2] The study was conducted by 15 experts (i.e. computer science students) who were familiar with the abstract writing task and the content of the abstracts they judged. To supplement this, we also provide **automatic** metrics. We use BLEU (Papineni et al., 2002), an n-gram overlap measure popular in text generation tasks, and METEOR (Denkowski and Lavie, 2014), a machine translation with paraphrase and language-specific considerations.

Comparisons We compare our GraphWriter against several strong baselines. In GAT, we replace our Graph Transformer encoder with a Graph Attention Network of (Veličković et al., 2018). This encoder consists of PReLU activations stacked between 6 self-attention layers. To determine the usefulness of including graph relations, we compare to a model which uses only entities and title (EntityWriter). Finally, we compare with the gated rewriter model of Wang et al. (2018) (Rewriter). This model uses only the document title to iteratively rewrite drafts of its output. [3]

Implementation Details Our models are trained end-to-end to minimize the negative joint log likelihood of the target text vocabulary and the copied entity indices. We use SGD optimization with momentum (Qian, 1999) and "warm restarts", a cyclical regiment that reduces the learning rate from 0.25 to 0.05 over the course of 5 epochs, then resets for the following epoch. Models are trained for 15 epochs with early stopping (Prechelt, 1998) based on the validation loss, with most models stopping between 8 and 13 epochs. We use single-layer LSTMs (Hochreiter and Schmidhuber, 1997) as recurrent networks. We use dropout (Srivastava et al., 2014) in self attention layers set to 0.3. Hidden states and embedding dimensions are fixed at 500 and attentions learn 500 dimen-

---

[2] Attempts to crowd source this evaluation failed.

[3] Due to the larger size and greater variety of our dataset and accompanying vocabularies compared to theirs, we were unable to train this model with the reported batch size of 240. We use batch size 24 instead, which is partially responsible for the lower performance.

	BLEU	METEOR
GraphWriter	**14.3** ± 1.01	**18.8** ± 0.28
GAT	12.2 ± 0.44	17.2 ± 0.63
EntityWriter	10.38	16.53
Rewriter	1.05	8.38

Table 2: Automatic Evaluations of Generation Systems.

sional projections. In Block layers, the feedforward network has an intermediate size of 2000, and we use a PReLU activation function (He et al., 2015). GraphWriter and GAT use $L = 6$ layers. The number of attention heads is set to 4. In all models, for both inputs and output, we replace words occurring fewer than 5 times with <*unk*> tokens. In each abstract, we replace all mentions in a coreference chain in the abstract with the canonical mention used in the graph. We decode with beam search (Graves, 2012; Sutskever et al., 2014) with a beam size of 4. A post-processing step deletes repeated sentences and repeated coordinated clauses.

5.1 Results

A comparison of all systems in terms of automatic metrics is shown in Table 2. Our GraphWriter model outperforms other methods. We see that models which leverage title, entities, and relations (GraphWriter and GAT) outperform models which use less information (EntityWriter and Rewriter).

We see that GraphWriter outperforms GAT across metrics, indicating that the global contextualization provided by GraphWriter improves generation. To verify the performance gap between GraphWriter and GAT, we report the average test metrics for 4 training runs of each model along with their variances. We see that the variance of the different models is non-overlapping, and in fact all training runs of GraphWriter outperformed all runs of GAT on these metrics.

Does Knowledge Help? To evaluate the value of knowledge in the generation task we compare our GraphWriter model to a model which does not generate from knowledge. We provide expert annotators with 50 randomly-selected paper titles from the test set and ask them for a single judgment according to the criteria described in Section 5. We pair each paper title with the generated abstracts produced by GraphWriter (a knowledge-informed modes), Rewriter (a knowledge-agnostic model), and the gold abstract (with canonicalized coreferential mentions).

	Best	Worst
Rewriter (No knowledge)	12%	64%
GraphWriter (Knowledge)	24%	36%
Human Authored	64%	0%

Table 3: Does knowledge improve generation? Human evaluations of best and worst abstract.

	Win	Lose	Tie
Structure	63%	17%	20%
Informativeness	43%	23%	33%
Grammar	63%	23%	13%
Overall	63%	17%	20%

Table 4: Human Judgments of GraphWriter and EntityWriter models.

Results of this comparison can be seen in Table 3. We see that GraphWriter is selected as "Best" more often than Rewriter, and is less often selected as "Worst", attesting to the value of including knowledge in the text generation process. We see that sometimes generated texts are preferred to human authored text, which is due in part to the disfluencies introduced by canonicalization of entity mentions.

To further understand the advantages of using knowledge graphs, we provide a more detailed comparison of the GraphWriter and EntityWriter models. We select 30 additional test datapoints and ask experts to provide per-criterion judgments of the outputs of the two systems. Since both models make use of extracted entities, we show this list along with the title for each datapoint, and modify the description of Informativeness to include "making use of the provided entities". Results of this evaluation are shown in Table 4. Here we see that including structured knowledge in the form of a graph improves abstract generation compared to generating from an unstructured collection of entities. The largest gains are made in terms of document structure and grammar, indicating that the structure of the input knowledge is being translated into the surface form.

Generating from Title The Rewriter model (Wang et al., 2018) considers the task of generating an abstract with only the paper's title as input. We compare against this model because it is among the first end-to-end systems to attempt to write scientific abstracts. However, the task setup used in Wang et al. (2018) differs significantly from the task introduced in this work. In order

Title	Block and Group Regularized Sparse Modeling for Dictionary Learning
Knowledge	(dictionary learning, CONJUNCTION, sparse coding) ; (optimization problems, USED-FOR, dictionary learning) ; (optimization problems, USED-FOR, sparse coding)...
GraphWriter	Sparse representations have recently been shown to be effective in many optimization problems. However, existing dictionary learning methods are limited in the number of dictionary blocks, which can be expensive to obtain. In this paper, we propose a novel approach to dictionary learning based on sparse coding ...
GAT	In this paper, we consider the problem of dictionary learning in well-known datasets. In particular, we consider the problem of dictionary learning, where the goal is to find a set of dictionary blocks that maximize the likelihood of a given set of dictionary blocks ...
EntityWriter	We propose a novel dictionary learning framework for reconstructed block/group sparse coding schemes. The dictionary learning framework is based on the descent, which is a block structure of the group structure ...
Rewriter	This paper presents a new approach to the k-means of the algorithm. The proposed approach is based on the basis of the stationarity algorithm. The algorithm is based on the fact that the number of bits is a constant of the base of the base of the input ...
Gold	This paper proposes a dictionary learning framework that combines the proposed block/group (BGSC) or reconstructed block/group (R-BGSC) sparse coding schemes with the novel Intra-block Coherence Suppression Dictionary Learning algorithm. An important and distinguishing feature of the proposed framework is that all dictionary blocks are trained simultaneously ...
Title	Image Interpolation with Directionlets
Knowledge	(directionally adaptive image interpolation USED-FOR edge information) ; (numeric and visual quality, HYPONYM-OF, directionally adaptive image interpolation) ; (directionlets, EVALUATE-FOR, multiple-direction wavelet transform) ...
GraphWriter	In this paper, we propose a novel directionally adaptive image interpolation based on the multiple-direction wavelet transform, called directionlets, which can be used as a directionlets to improve the numeric and visual quality of the directionally adaptive image interpolation ...
GAT	In this paper, we propose a novel directionally adaptive image interpolation, called directionally adaptive image interpolation, for directionally adaptive image interpolation , which is based on the multiple-direction wavelet transform ...
EntityWriter	We present a novel directionally adaptive image interpolation for numeric and visual quality. The wavelet transform is based on the wavelet transform between the low-resolution image and the interpolated image. The high-resolution image is represented by a wavelet transform ...
Rewriter	We present a new method for finding topic-specific data sets. The key technical contributions of our approach is to be a function of the terrestrial distributed memory. The key idea is to be a function of the page that seeks to be ranked the buckets of the data. The basic idea is a new tool for the embedded space ...
Gold	We present a novel directionally adaptive image interpolation based on a multiple-direction wavelet transform, called directionlets. The directionally adaptive image interpolation uses directionlets to efficiently capture directional features and to extract edge information along different directions from the low-resolution image ...

Table 5: Example outputs of various systems versus Gold.

to make a fair comparison, we construct a variant of our model which is only provided with a title as input. We develop a model that predicts entities from the title, and then uses our knowledge-aware model to generate the abstract. For this comparison we use the EntityWriter model with a collection of entities inferred from the title alone (InferEntityWriter).

To infer relevant entities, we learn to embed titles and entities extracted from the corresponding abstract in a shared dense vector space by minimizing their cosine distance. We use negative sampling to provide definition to this vector space. At test time, we use the title embedding to infer the $K = 12$ closest entities to feed into the InferEntityWriter model. Results are shown in Table 6, which shows that InferEntityWriter achieves better results than Rewriter, indicating that the intermediate entity prediction step is helpful in abstract generation.

	BLEU	METEOR
Rewriter	1.05	8.38
InferEntityWriter	3.60	12.2

Table 6: Comparison of generation without knowledge and with Inferred Knowledge (InferEntityWriter)

5.2 Analysis

Table 5 shows examples of various system outputs for a particular test instance.We see that GraphWriter makes use of more entities from the input, arranged with more articulated textual context. It demonstrates less repetition than GAT. Both GraphWriter and GAT show much better coherence than EntityWriter, which copies entities from the input into unreasonable contexts. Rewriter, while fluent and grammatical, jumps from topic to topic, failing to relate as strongly to the input as the knowledge-aware models.

To determine the shortcomings of our model, we calculate rough error statistics over the outputs of the GraphWriter on the test set. We notice that 40% of entities in the knowledge graphs do not appear in the generated text. Future work should address this coverage problem, perhaps through modifications to the inference procedure or a coverage loss (Tu et al., 2016) modified to the specifics of this task. We find that 18% of all sentences generated by our model repeat sentences or clauses and are subjected to the post-processing pruning mentioned in Section 5. While this step is a simple solution to improve generated outputs, a more advanced solution is required.

6 Conclusion

We have studied the problem of generating multi-sentence text from the output of automatic information extraction systems, and have shown that incorporating knowledge as graphs improves performance. We introduced GraphWriter, featuring a new attention model for graph encoding, and demonstrated its utility through human and automatic evaluation compared to strong baselines. Lastly, we provide a new resource for the generation community, the AGENDA dataset of abstracts and knowledge. Future work could address the problem of repetition and entity coverage in the generated texts.

Acknowledgments

This research was supported by the Office of Naval Research under the MURI grant N00014-18-1-2670, NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We gratefully acknowledge the support of the European Research Council (Lapata; award number 681760). We also thank the anonymous reviewers and the UW-NLP group for their helpful comments.

References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the Literature Graph in Semantic Scholar. In *NAACL*.

Regina Barzilay and Mirella Lapata. 2005. Collective Content Selection for Concept-to-Text Generation. In *EMNLP*, pages 331–338. Association for Computational Linguistics.

Daniel Edward Robert Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *ACL*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*.

Michael J. Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Workshop on Statistical Machine Translation*.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *arXiv preprint arXiv:1211.3711*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Joohyun Kim and Raymond J Mooney. 2010. Generative Alignment and Semantic Parsing for Learning from Ambiguous Supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Svetlana Kiritchenko and Saif Mohammad. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling. In *NAACL-HLT*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke S. Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *ACL*.

Ioannis Konstas and Mirella Lapata. 2013. Inducing Document Plans for Concept-to-Text Generation. In *EMNLP*, pages 1503–1514.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning Semantic Correspondences with Less Supervision. In *ACL/AFNLP*, pages 91–99.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-Worst Scaling: A Model for the Largest Difference Judgments. *University of Alberta: Working Paper*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *EMNLP*, pages 3219–3232.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep Graph Convolutional Encoders for Structured Data to Text Generation. *INLG*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *NAACL-HLT*, pages 720–730.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Lutz Prechelt. 1998. Early Stopping - but when? In *Neural Networks: Tricks of the Trade*, pages 55–69, London, UK, UK. Springer-Verlag.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. In *AAAI*.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Accessed at `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the Point: Summarization with Pointer-Generator Networks. *arXiv preprint arXiv:1704.04368*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. *arXiv preprint arXiv:1601.04811*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NeurIPS*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Paper Abstract Writing through Editing Mechanism. In *Proceedings of NAACL-HLT*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-document Generation. In *EMNLP*.

Open Information Extraction from Question-Answer Pairs

Nikita Bhutani [*]
University of Michigan
nbhutani@umich.edu

Yoshihiko Suhara
Megagon Labs
yoshi@megagon.ai

Wang-Chiew Tan
Megagon Labs
wangchiew@megagon.ai

Alon Halevy
Megagon Labs
alon@megagon.ai

H. V. Jagadish
University of Michigan
jag@eecs.umich.edu

Abstract

Open Information Extraction (OPENIE) extracts meaningful structured tuples from free-form text. Most previous work on OPENIE considers extracting data from one sentence at a time. We describe NEURON, a system for extracting tuples from question-answer pairs. Since real questions and answers often contain precisely the information that users care about, such information is particularly desirable to extend a knowledge base with.

NEURON addresses several challenges. First, an answer text is often hard to understand without knowing the question, and second, relevant information can span multiple sentences. To address these, NEURON formulates extraction as a multi-source sequence-to-sequence learning task, wherein it combines distributed representations of a question and an answer to generate knowledge facts. We describe experiments on two real-world datasets that demonstrate that NEURON can find a significant number of new and interesting facts to extend a knowledge base compared to state-of-the-art OPENIE methods.

1 Introduction

Open Information Extraction (OPENIE) (Banko et al., 2007) is the problem of extracting structured data from a text corpus, without knowing a priori which relations will be extracted. It is one of the primary technologies used in building knowledge bases (KBs) that, in turn, power question answering (Berant et al., 2013). The vast majority of previous work on OPENIE extracts structured information (e.g., triples) from individual sentences.

This paper addresses the problem of extracting structured data from conversational question-answer (CQA) data. Often, CQA data contains precisely the knowledge that users care about. As such, this data offers a goal-directed method for extending existing knowledge bases. Consider, for example, a KB about a hotel that is used to power its website and/or a conversational interface for hotel guests. The KB provides information about the hotel's services: complimentary breakfast, free wifi, spa. However, it may not include information about the menu/times for the breakfast, credentials for the wifi, or the cancellation policy for a spa appointment at the hotel. Given the wide range of information that may be of interest to guests, it is not clear how to extend the KB in the most effective way. However, the conversational logs, which many hotels keep, contain the actual questions from guests, and can therefore be used as a resource for extending the KB. Following examples illustrate the kind of data we aim to extract:

Example 1. *Q: Does the hotel have a gym?*
A: It is located on the third floor and is 24/7.
Tuple: ⟨*gym, is located on, third floor*⟩

Example 2. *Q: What time does the pool open?*
A: 6:00am daily.
Tuple: ⟨*pool, open, 6:00am daily*⟩

As can be seen from these examples, harvesting facts from CQA data presents significant challenges. In particular, the system must interpret information collectively between the questions and answers. In this case, it must realize that *'third floor'* refers to the location of the *'gym'* and that *6:00am* refers to the opening time of the pool. OPENIE systems that operate over individual sentences ignore the discourse and context in a QA pair. Without knowing the question, they either fail to or incorrectly interpret the answer.

This paper describes NEURON, an end-to-end system for extracting information from CQA data. We cast OPENIE from CQA as a multi-source sequence-to-sequence generation problem to explicitly model both the question and answer

---

[*] Part of the work was done while the author was at Megagon Labs.

Proceedings of NAACL-HLT 2019, pages 2294–2305
Minneapolis, Minnesota, June 2 - June 7, 2019.

in a QA pair. We propose a multi-encoder, constrained-decoder framework that uses two encoders to encode each of the question and answer to an internal representation. The two representations are then used by a decoder to generate an output sequence corresponding to an extracted tuple. For example, the output sequence of Example 2 is:

⟨*arg*$_1$⟩ pool ⟨/*arg*$_1$⟩⟨*rel*⟩ open ⟨/*rel*⟩⟨*arg*$_2$⟩ 6:00am daily ⟨/*arg*$_2$⟩

While encoder-decoder frameworks have been used extensively for machine translation and summarization, there are two key technical challenges in extending them for information extraction from cQA data. First, it is vital for the translation model to learn constraints such as, arguments and relations are sub-spans from the input sequence, output sequence must have a valid syntax (e.g., ⟨*arg*$_1$⟩ must precede ⟨*rel*⟩). These and other constraints can be integrated as *hard* constraints in the decoder. Second, the model must recognize auxiliary information that is irrelevant to the KB. For example, in the hotel application, NeurON must learn to discard greetings in the data. Since existing facts in the KB are representative of the domain of the KB, this prior knowledge can be incorporated as *soft* constraints in the decoder to rank various output sequences based on their relevance. Our contributions are summarized below:

- We develop NeurON, a system for extracting information from cQA data. NeurON is a novel multi-encoder constrained-decoder method that explicitly models both the question and the answer of a QA pair. It incorporates vocabulary and syntax as *hard* constraints and prior knowledge as *soft* constraints in the decoder.
- We conduct comprehensive experiments on two real-world cQA datasets. Our experimental results show that the use of hard and soft constraints improves the extraction accuracy and NeurON achieves the highest accuracy in extracting tuples from QA pairs compared with state-of-the-art sentence-based models, with a relative improvement as high as 13.3%. NeurON's higher accuracy and ability to discover 15-25% tuples that are not extracted by state-of-the-art models make it suitable as a tuple extraction tool for KB extension.
- We present a case study to demonstrate how a KB can be extended iteratively using tuples extracted using NeurON. In each iteration, only relevant tuples are included in the KB. In turn, the extended KB is used to improve relevance scoring for subsequent iterations.

2 Task Formulation

In this work, we choose to model an OpenIE extraction from a question-answer (QA) pair as a tuple consisting of a single relation with two arguments, where the relation and arguments are contiguous spans from the QA pair. Formally, let (q, a) be a QA pair, where question $q = (q_1, q_2, ..., q_m)$ and answer $a = (a_1, a_2, ..., a_n)$ are word sequences. The output is a triple (arg_1,rel,arg_2) extracted from (q, a). The output triple can be naturally interpreted as a sequence $y = (y_1, y_2, ..., y_o)$ where y_i is either a word or a placeholder tag (⟨*arg*$_1$⟩, ⟨*rel*⟩, ⟨*arg*$_2$⟩) that marks relevant portions of the triple. In OpenIE, the extracted tuple should be asserted by the input QA pair. Formulating this, therefore, requires the vocabulary of y to be restricted to the vocabulary of (q, a) and placeholder tags.

Following this definition, our aim is to directly model the conditional probability $p(y|q, a)$ of mapping input sequences q and a into an output sequence:

$$P(y|q, a) = \prod_{i=1}^{o} p(y_i|y_1, \ldots, y_{i-1}, q, a). \quad (1)$$

In our formulation, a triple is generated as a sequence: a *head* argument phrase arg_1, followed by a relation phrase *rel* and a *tail* argument phrase arg_2. It is possible to consider different orderings in the output sequence (such as (rel,arg_1,arg_2)). However, the goal of OpenIE is to identify the relation phrase that holds between a pair of arguments. Our representation is, thus, consistent with this definition as it models the relation phrase to depend on the head argument.

3 NeurON: A Multi-Encoder Constrained-Decoder Model

Overview of NeurON We propose to extract tuples using a variation of an encoder-decoder RNN architecture (Cho et al., 2014) operating on variable-length sequences of tokens. Fig. 1 shows the architecture of NeurON. It uses two encoders to encode question and answer sequences in a QA pair separately into fixed-length vector representations. A decoder then decodes the vector representations into a variable-length sequence corresponding to the tuple. The decoder is integrated

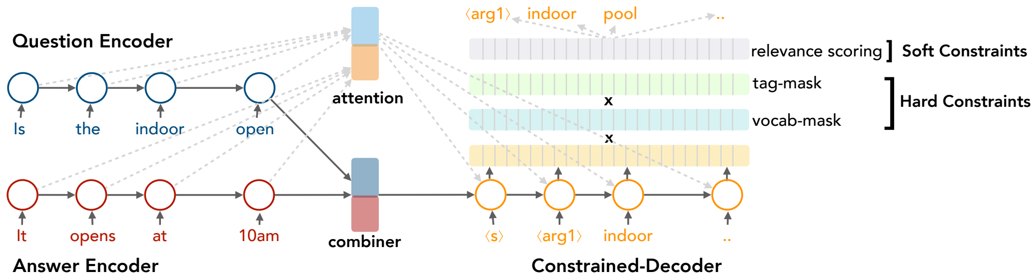


Figure 1: Multi-Encoder, Constrained-Decoder model for tuple extraction from (q, a).

with a set of *hard constraints* (e.g., output vocabulary) and *soft constraints* (e.g., relevance scoring) suited for the extraction task.

3.1 Multiple Encoders

Given an input QA pair, two RNN encoders separately encode the question and answer. The question encoder converts q into hidden representation $h^q = (h_1^q, ..., h_m^q)$ and the answer encoder converts a into $h^a = (h_1^a, ..., h_n^q)$, where $h_t^q = \text{lstm}(q_t, h_{t-1}^q)$ is a non-linear function represented by the long short-term memory (LSTM) cell. The combiner combines the encoders' states and initializes the hidden states $\mathbf{h}$ for the decoder:

$$h = \tanh(W_c[h^q \circ h^a]),$$

where $\circ$ denotes concatenation. The decoder stage uses the hidden states to generate the output y with another LSTM-based RNN. The probability of each token is defined as:

$$p(y_t) = \text{softmax}((s_t \circ c_t^q \circ c_t^a)W_y), \qquad (2)$$

where s_t denotes the decoder state, $s_0 = h$ and $s_t = \text{lstm}((y_{t-1} \circ c_t^q \circ c_t^a)W_s, s_{t-1})$. The decoder is initialized by the last hidden state from the combiner. It uses the previous output token at each step. Both W_y and W_s are learned matrices. Each decoder state is concatenated with *context vectors* derived from the hidden states of the encoders. Context vector c_t is the weighted sum of the encoder hidden states, i.e. $c_t^q = \sum_{i=1}^m \alpha_{t_i} h_i^q$, where α_{t_i} corresponds to an attention weight. The attention model (Bahdanau et al., 2015) helps the model learn to focus on specific parts of the input sequences, instead of solely relying on hidden vectors of the decoders' LSTM. This is crucial for extraction from (q, a) pairs where input sequences tend to be long.

3.2 Constrained Decoder

The decoder finds the best hypothesis (i.e., the best output sequence) for the given input representations. Typically, the output sequence is generated, one unit at a time, using *beam search*. At each time step, the decoder stores the top-k scoring partial sequences, considers all possible single token extensions of them, and keeps k most-likely sequences based on model's probabilities (Eq. 1). As soon as the $\langle/S\rangle$ symbol is appended, the sequence is removed from the beam and added to the set of complete sequences. The most-likely complete sequence is finally generated.

Hard Constraints While such encoder-decoder models typically outperform conventional approaches (Cho et al., 2014; Zoph and Knight, 2016; Xiong et al., 2017) on a wide variety of tasks including machine translation and question answering, the accuracy and training efficiency has been shown to improve when the model is integrated with the constraints of the output domain (Xiao et al., 2016; Yin and Neubig, 2017). Motivated by these, NEURON allows constraints relevant to information extraction to be incorporated in the model. Specifically, we describe how the decoder can enforce vocabulary and structural constraints on the output.

- **Vocabulary constraints.** Since the arguments and relations in the extracted tuples typically correspond to the input QA pair, the decoder must constraint the space of next valid tokens when generating the output sequence. NEURON uses a masking technique in the decoder to mask the probability of tokens (as in Eq. 2) that do not appear in the input (q, a) pair. Specifically, it computes a binary mask vector v, where $|v|$ is vocabulary size and $v_i = 1$ if and only if i-th token appears in q or a. The probability of each token is modified as:

$$p(y_t) = \text{softmax}((s_t \circ c_t^q \circ c_t^a)W_y \otimes v), \qquad (3)$$

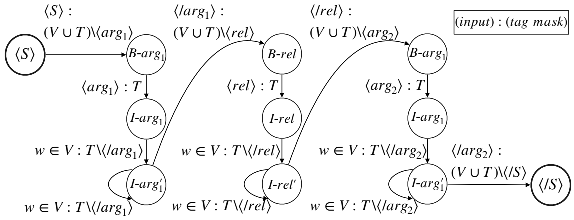


Figure 2: State diagram for tag masking rules. V is the vocabulary including placeholder tags, T is the set of placeholder tags.

where $\otimes$ indicates element-wise multiplication.

- **Structural constraints.** For the output sequence to correspond to a valid tuple with non-empty arguments, the decoding process must conform to the underlying grammar of a tuple. For instance, decoding should always begin in the $\langle S \rangle$ state, where only $\langle arg_1 \rangle$ can be generated. In subsequent time steps, all other placeholders except $\langle /arg_1 \rangle$ should be restricted to ensure a non-empty argument. Once $\langle /arg_1 \rangle$ is generated, $\langle rel \rangle$ must be generated in the next time step and so on. The various states and grammar rules can be described as a finite state transducer as shown in Figure 2.

Depending upon the state, NeurON generates a mask r based on this grammar and uses r to further modify the probabilities of the tokens as follows:

$$p(y_t) = \text{softmax}((s_t \circ c_t^q \circ c_t^a) W_y \otimes v \otimes r). \quad (4)$$

Soft Constraints OpenIE systems are typically used to extract broad-coverage facts to extend existing KBs. Facts already existing in the KB are representative of the domain of the KB. It is, therefore, useful to incorporate this prior knowledge in the extraction itself. NeurON is able to use prior knowledge (incorporated as soft constraints) in the decoder to understand the relevance of candidate extractions and adjust the ranking of various output sequences accordingly. To see why such soft constraints can be useful, consider the example:

Example 3. *Q: "Is the pool open?"*
A: "I am sorry but our pool reopens at 7:00am."
Tuple: ⟨I, am, sorry⟩; ⟨pool, reopens at, 7:00am⟩

Both the tuple facts are correct given the input QA pair but only the second tuple contains useful information. Filtering such irrelevant facts is difficult without additional evidence.

The multi-encoder and constrained-decoder in NeurON are jointly optimized to maximize the log probability of output sequence conditioned on the input sequences. At inference, the decoder estimates the likelihood of various candidate sequences and generates the sequence with the highest likelihood. As shown in Eq. 1, this likelihood is conditioned solely on the input (q, a) pair, thus increasing the possibility of obtaining facts that may be correct but irrelevant. Instead, if a relevance scoring function were integrated at extraction time, the candidate output sequences could be re-ranked so that the predicted output sequence is likely to be both correct and relevant.

Learning a relevance scoring function can be modeled as a KB completion task, where missing facts have to be inferred from existing ones. A promising approach is to learn vector representations of entities and relations in a KB by maximizing the total plausibility of existing facts in the KB (Wang et al., 2017). For a new candidate output sequence, its plausibility can be predicted using the learned embeddings for the entities and relation in the sequence.

In NeurON, we learn the entity and relation embeddings using Knowledge Embedding (KE) methods such as TransE (Bordes et al., 2013) and HolE (Nickel et al., 2016). Note that NeurON is flexible with how the relevance scoring function is learned or which KE method is chosen. In this paper, we use TransE for evaluation. TransE computes the plausibility score S of a tuple $y = \langle arg_1, rel, arg_2 \rangle$ as:

$$S(y) = ||\mathbf{v}_{arg_1} + \mathbf{v}_{rel} - \mathbf{v}_{arg_2}||,$$

where $\mathbf{v}_{arg_1}$, $\mathbf{v}_{rel}$, and $\mathbf{v}_{arg_2}$ are embedding vectors for arg_1, rel, and arg_2 respectively. Following (Jain et al., 2018), we compute the embedding vectors of out-of-vocabulary arguments (and relations) as the average of embedding vectors of known arguments (and relations). We generate the most-likely output based on its conditional probability and plausibility score:

$$\hat{y} = \underset{y}{\text{argmax}}(\log P(y|q, a) + \gamma \log S(y)). \quad (5)$$

To implement the tuple relevance scoring function, we employ the re-ranking approach, which is a common technique for sequence generation methods (Luong et al., 2015b). Our re-ranking method first obtains candidates from a beam search decoder and then re-ranks the candidates based on the objective function (Eq. 5).

4 Experiments

We evaluated the performance of NEURON on two cQA datasets. In our analysis, we find that integrating hard and soft constraints in the decoder improved the extraction performance irrespective of the number of encoders used. Also, 15-25% of the tuples extracted by NEURON were not extracted by state-of-the-art sentence-based methods.

4.1 Datasets and Data Preparation

ConciergeQA is a real-world internal corpus of 33,158 QA pairs collected via a multi-channel communication platform for guests and hotel staff. Questions (answers) are always made by guests (staff). An utterance has 36 tokens on average, and there are 25k unique tokens in the dataset. A QA utterance has 2.8 sentences on average, with the question utterance having 1.02 sentences on average and answer utterance having 1.78 sentences on average.

AmazonQA (Wan and McAuley, 2016; McAuley and Yang, 2016) is a public dataset with 314,264 QA pairs about electronic products on *amazon.com*. The dataset contains longer and more diverse utterances than the *ConciergeQA* dataset: utterances have an average of 45 tokens and the vocabulary has more than 50k unique tokens. A QA utterance has 3.5 sentences on average. The question utterances had 1.5 sentences on average and the answer having 2 sentences.

For training NEURON, we bootstrapped a large number of high-quality training examples using a state-of-the-art OPENIE system. Such bootstrapping has been shown to be effective in information extraction tasks (Mausam et al., 2012; Saha et al., 2017). The StanfordIE (Angeli et al., 2015) system is used to extract tuples from QA pairs for training examples. To further obtain high-quality tuples, we filtered out tuples that occur too infrequently (< 5) or too frequently (> 100). For each tuple in the set, we retrieved all QA pairs that contain all the content words of the tuple and included them in the training set. This helps create a training set encapsulating the multiplicity of ways in which tuples are expressed across QA pairs. We randomly sampled 100 QA pairs from our bootstrapping set and found 74 of them supported the corresponding tuples. We find this quality of bootstrapped dataset satisfactory, since the seed tuples for bootstrapping could be noisy as they were generated by another OPENIE system.

Instance type	*ConciergeQA*	*AmazonQA*
Exclusively from question	13.9%	13.8%
Exclusively from answer	25.8%	17.6%
Ambiguous	36.9%	29.8%
Jointly from Q-A	23.4%	38.8%

Table 1: Various types of training instances.

Dataset	Q-A	|V|	Train	Dev	Test (Q-A)
ConciergeQA	33k	25k	1.25M	128k	2,905
AmazonQA	314k	50k	1.43M	159k	39,663

Table 2: Training, Dev and Test splits.

Our bootstrapped dataset included training instances where a tuple matched (a) tokens in the questions exclusively, (b) tokens in the answers exclusively, (c) tokens from both questions and answers. Table 1 shows the distribution of the various types of training instances. Less than 40% (30%) of ground truth tuples for *ConciergeQA* (*AmazonQA*) exclusively appear in the questions or answers. Also, 22.1% (37.2%) of ground truth tuples for *ConciergeQA* (*AmazonQA*) are extracted from the combination of questions and answers. These numbers support our motivation of extracting tuples from QA pairs. We used standard techniques to construct training/dev/test splits so that QA pairs in the three sets are disjoint. Table 2 shows the details of the various subsets.

4.2 Baseline Approaches

We compared NEURON with two methods that can be trained for tuple extraction from QA pairs: BILSTM-CRF (Huang et al., 2015) and NEURALOPENIE (Cui et al., 2018). BILSTM-CRF is a sequence tagging model that has achieved state-of-the-art accuracy on POS, chunking, NER and OPENIE (Stanovsky et al., 2018) tasks. For OPENIE, the model predicts boundary labels (e.g., B-ARG1, I-ARG1, B-ARG2, O) for the various tokens in a QA pair. NEURALOPENIE is an encoder-decoder model that generates a tuple sequence given an input sequence. Since it uses a single encoder, we generate the input sequence by concatenating the question and answer in a QA pair. We trained all the models using the same training data.

4.3 Performance Metrics

We examine the performance of different methods using three metrics: *precision*, *recall*, and *relative coverage* (RC). Given a QA pair, each system returns a sequence. We label the sequence correct if it matches one of the ground-truth tuples for the QA pair, incorrect otherwise. We then measure precision of a method (i.e., # of correct predictions of the method / # of question-answer pairs) and recall (i.e., # of correct predictions of the method / # of correct predictions of any method) following (Stanovsky and Dagan, 2016). To compare the coverage of sequences extracted by NEURON against the baseline method, we compute relative coverage of NEURON as the fraction of all correct predictions that were generated exclusively by NEURON. Specifically,

$$RC = \frac{|\mathrm{TP}_{\text{NEURON}} \setminus \mathrm{TP}_{baseline}|}{|\mathrm{TP}_{\text{NEURON}} \bigcup \mathrm{TP}_{baseline}|},$$

where TP denotes the correct predictions.

4.4 Model Training and Optimization

We implemented NEURON using OpenNMT-tf (Klein et al., 2017) , an open-source neural machine translation system that supports multi-source encoder-decoder models. We implemented NEURALOPENIE using the same system. We used the open-source implementation of BILSTM-CRF (Reimers and Gurevych, 2017). For fair comparison, we used identical configurations for NEURON and NEURALOPENIE. Each encoder used a 3-layer bidirectional LSTM and the decoder used a 3-layer bidirectional LSTM. The models used 256-dimensional hidden states, 300-dimensional word embeddings, and a vocabulary size of 50k. The word embeddings were initialized with pre-trained GloVe embeddings (glove.6B) (Pennington et al., 2014). We used an initial learning rate of 1 and optimized the model with stochastic gradient descent. We used a decay rate of 0.7, a dropout rate of 0.3 and a batch size of 64. The models were trained for 1M steps for the *ConciergeQA* dataset and 100k steps for the *AmazonQA* dataset. We used TESLA K80 16GB GPU for training the models. We trained the KE models for relevance scoring using our bootstrapped training dataset. For integrating the relevance scoring function, we experimented with different values for γ and found it not have a major impact within a range of 0.02 to 0.2. We used a value of 0.05 in all the experiments.

4.5 Experimental Results

The BILSTM-CRF model showed extremely low (2-15%) precision values. Very few of the tagged sequences (32-39%) could be converted to a tuple. Most tagged sequences had multiple relations and arguments, indicating that it is difficult to learn how to tag a sequence corresponding to a tuple. The model only learns how to best predict tags for each token in the sequence, and does not take into account the long-range dependencies to previously predicted tags. This is still an open problem and is outside the scope of this paper.

Method	P	R	RC
NEURALOPENIE (baseline)	0.769	0.580	-
+ hard constraints	0.776	0.585	-
+ hard and soft constraints	0.796	0.600	-
NEURON (our method)	0.791	0.597	0.224
+ hard constraints	0.792	0.597	0.204
+ hard and soft constraints	**0.807**	**0.608**	**0.245**

Table 3: Precision (P), Recall (R), and Relative Coverage (RC) results on *ConciergeQA*.

Tables 3 and 4 show the performance of NEURALOPENIE and NEURON on the two CQA datasets. NEURON achieves higher precision on both the datasets. This is because NEURALOPENIE uses a single encoder to interpret the question and answer in the same vector space, which leads to lower performance. Furthermore, concatenating the question and answer makes the input sequence too long for the decoder to capture long-distance dependencies in history (Zhang et al., 2016; Toral and Sánchez-Cartagena, 2017). Despite the attention mechanism, the model ignores past alignment information. This makes it less effective than the dual-encoder model used in NEURON.

The tables also show that incorporating task-specific hard constraints helps further improve the overall precision and recall, regardless of the methods and the datasets. Re-ranking the tuples based on the soft constraints derived from the existing KB further improves the performance of both methods in *ConciergeQA* and NEURALOPENIE in *AmazonQA*. The existing KB also helps boost the likelihood of a correct candidate tuple sequence that was otherwise scored to be less likely. Lastly, we found that NEURON has significant relative coverage; it discovered significant additional, unique tuples missed by NEURALOPENIE.

Table 4 shows a slight decrease in performance for NEURON after soft constraints are added. This is likely caused by the lower quality KE model due to the larger vocabulary in *AmazonQA*. In contrast, even with the lower quality KE model, NEU-

Method	P	R	RC
NeuralOpenIE (baseline)	0.557	0.594	-
+ hard constraints	0.563	0.601	-
+ hard and soft constraints	0.571	0.610	-
NeurON (our method)	0.610	0.652	0.139
+ hard constraints	**0.631**	**0.674**	**0.164**
+ hard and soft constraints	0.624	0.666	0.149

Table 4: Precision (P), Recall (R), and Relative Coverage (RC) results on *AmazonQA* dataset.

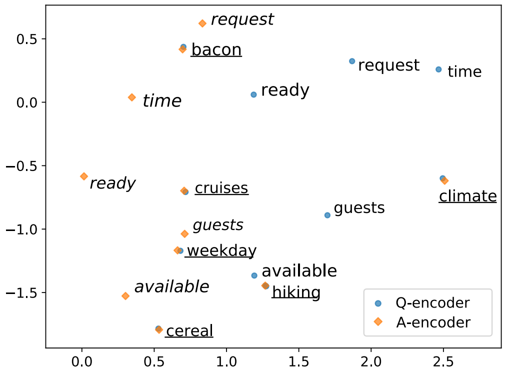


Figure 3: Example embedding vectors from question and answer encoders. Underlines denote similar embedding vectors in both the encoders.

ralOpenIE improved slightly. This is likely because the NeuralOpenIE model, at this stage, still had a larger margin for improvement. We note however that learning the best KE model is not the focus of this work.

AmazonQA is a more challenging dataset than *ConciergeQA*: longer utterances (avg. 45 tokens vs. 36 tokens) and richer vocabulary ($>$ 50k unique tokens vs. $<$ 25k unique tokens). This is reflected in lower precision and recall values of both the systems on the *AmazonQA* dataset. While the performance of end-to-end extraction systems depends on the complexity and diversity of the dataset, incorporating hard and soft constraints alleviates the problem to some extent.

End-to-end extraction systems tend to outperform rule-based systems on extraction from cQA datasets. We observed that training data for *ConciergeQA* had a large number ($>$ 750k) dependency-based pattern rules, of which $<$ 5% matched more than 5 QA pairs. The set of rules is too large, diverse and sparse to train an accurate rule-based extractor. Even though our training data was generated by bootstrapping from a rule-based extractor StanfordIE, we found only 51.5% (30.7%) of correct tuples from NeurON exactly matched the tuples from StanfordIE in *ConciergeQA* (*AmazonQA*). This indicates that NeurON combined information from question and answer, otherwise not accessible to sentence-wise extractors. As an evidence, we found 11.4% (6.1%) of tuples were extracted from answers, 16.8% (5.0%) from questions, while 79.6% (82.5%) combined information from questions and answers in *ConciergeQA* (*AmazonQA*).

Multiple Encoders: Our motivation to use different encoders for questions and answers is based on the assumption that they use different vocabulary and semantics. We found that there were 8k (72k) unique words in questions, 18k (114k) unique words in answers, and the Jaccard coefficient between two vocabulary sets was 0.25 (0.25) in *ConciergeQA* (*AmazonQA*), indicating that two sources use significantly different vocabulary. Also, the same word can have different meanings depending on a speaker, and thus such words in the two sources should be embedded differently. To visualize the embedding vectors of common words in *ConciergeQA*, we mapped them into 2D space using t-SNE (Maaten and Hinton, 2008). Fig. 3 shows that *subjective* words that represents speakers attitude (e.g., “ready”, “guests”, “time”) had significantly different embeddings in the question and answer encoders. In contrast, *objective* words such as menu, or activity (e.g., “bacon”, “cruise”, “weekday”) had similar embeddings although the two encoders do not directly share the embedding parameters. This indicates that multiple encoders not only capture the different meanings in questions and answers but also retain consistent meanings for words that keep the same meanings in the two sources.

Relevance Scoring: We compared with another NeurON model that uses HolE (Nickel et al., 2016) for relevance scoring. Both the HolE and TransE models achieved the same precision of 80.7%, with HolE achieving slightly higher recall (+1.4%). This suggests that incorporating relevance scoring in NeurON can robustly improve the extraction accuracy, regardless of the choice of the knowledge embedding method. We also estimated the upper-bound precision by evaluating if the correct tuple was included in the top-500 candidates. The upper-bound precision was 85.0% on *ConciergeQA*, indicating that there is still room for improvement on incorporating relevance scoring.

1	**Q**: Tell me what the username and password is for WiFi **A**: Absolutely! Both the username and passcode is C800. **StanfordIE**: ⟨ passcode, is, C800 ⟩ **NeurON**: ⟨ password, is, C800 ⟩
2	**Q**: Do hotel guys have ice? **A**: There is an ice machine on first floor lobby. **StanfordIE**: ⟨ hotel, do, ice ⟩ **NeurON**: ⟨ hotel, have, ice machine ⟩
3	**Q**: Is there a charge for parking rental car on the property? **A**: Self-parking will be $15 per night. **StanfordIE**: None **NeurON**: ⟨ parking, will, charge ⟩
4	**Q**: arrange late check out for tomorrow? **A**: I have notated a 12 pm check out. Normal check out time is at 11 am. **StanfordIE**: ⟨ normal check, is at, 11 am ⟩ **NeurON**: ⟨ check, is at, 11 ⟩

Table 5: Examples of successful cases (1 and 2) and failed cases (3 and 4) from test data.

4.6 Error Analysis

We examined a random sample of 100 errors shared by all the systems across the tested datasets. Arguably, encoder-decoder models suffer when extracting tuples from long utterances (avg. of 54 tokens), contributing to 43% of the errors. 34% of the incorrectly extracted tuples used words that were shared across the two sources. This indicates that the extractor makes errors when resolving ambiguity in tokens. 28% of the error cases used informal language that is generally difficult for any extractor to understand. We show some examples (1 and 2 in Table 5) where NeurON successfully combined information across two sources and examples (3 and 4 in Table 5) where it failed.

We further examined three different scenarios: a) errors are shared by both NeurON and NeuralOpenIE, b) errors are made exclusively by NeurON, c) errors are made exclusively by NeuralOpenIE. For each scenario, we examined a random sample of 100 errors. We categorize the different sources of errors and report the results in Table 6. As shown, NeurON is superior on longer utterances compared to NeuralOpenIE (54 tokens vs. 49 tokens). However, ambiguity in tokens in the two sources is a concern for NeurON because it has the flexibility to interpret the question and answer differently. Not surprisingly, informal utterances are hard to translate for both the systems.

Error Category	$\overline{N}, \overline{B}$	$N, \overline{B}$	$\overline{N}, B$
long utterances	43%	45%	40%
avg. length of utterance	54 tokens	49 tokens	54 tokens
ambiguity	34%	36%	48%
informal language	28%	36%	34%

Table 6: Different errors $\overline{N}$ and $\overline{B}$ made by NeurON (N) and NeuralOpenIE (B) respectively.

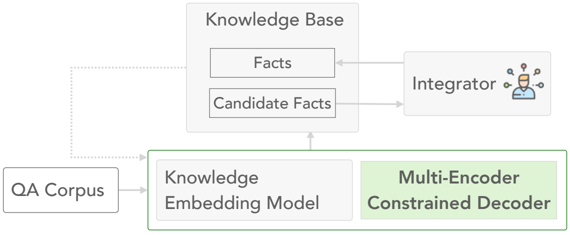


Figure 4: Human-in-the-loop system for extending a domain-specific KB.

5 Case Study - KB Extension

The extracted tuples from NeurON can be used to extend a KB for a specific domain. However, automatically fusing the tuples with existing facts in the KB can have limited accuracy. This can be due to noise in the source conversation, no prior knowledge of join rules and more. One possible solution is to design a human-in-the-loop system that iteratively extracts tuples and filters them based on human feedback (Fig. 4). In each iteration, a set of tuples is annotated by human annotators based on their relevance to the domain of the KB. The tuples marked relevant are added to the KB and the relevance scoring function is updated for extracting more relevant tuples from the corpus in the next iteration.

We conducted a crowdsourced experiment[1], simulating the first iteration of the procedure i.e., when no KE model is available. We collected annotations on top-5 tuples extracted by NeurON for 200 QA pairs in the *ConciergeQA* dataset. For reliability, we hired five workers for each extraction. The workers were asked to judge if a tuple is relevant to the hotel domain and represents concrete information to be added to a KB. We found precision@5 was 41.4%, and NeurON extracted at least one useful tuple for 83.0% of the 200 QA pairs. Overall, the system added 243 unique tuples (out of 414 tuples extracted by NeurON) to the KB. We also collected annotations for the tuples extracted by NeuralOpenIE. The precision@5 and recall@5 values were 41.3% and 79.0% respectively. Although the precision values are quite similar, NeurON can extract correct tuples from more QA pairs than NeuralOpenIE. While the precision can further be improved, the preliminary results support that NeurON is a good candidate for extraction in

---

[1] `https://www.figure-eight.com/`

a human-in-the-loop system for KB extension. We did not use any sophisticated methods for ranking tuples in our experiment. Thus, a better ranking algorithm might lead to improved precision.

6 Related Work

There is a long history of OpenIE systems for extracting tuples from plain text. They are built on hand-crafted patterns over an intermediate representation of a sentence (e.g., POS tags (Yates et al., 2007; Fader et al., 2011), dependency trees (Bhutani et al., 2016; Mausam et al., 2012)). Such rule-based systems require extensive engineering when the patterns become diverse and sparse. Recently, OpenIE systems based on end-to-end frameworks, such as sequence tagging (Stanovsky et al., 2018) or sequence-to-sequence generation (Cui et al., 2018), have been shown to alleviate such engineering efforts. However, all these systems focus on sentence-level extraction. We are the first to address the problem of extracting tuples from question-answer pairs.

Our proposed system is based on an encoder-decoder architecture, which was first introduced by Cho et al. for machine translation. Attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015b) have been shown to be effective for mitigating the problem of poor translation performance on long sequences. Their model can learn how much information to retrieve from specific parts of the input sequence at decoding time. There is abundant research on generalizing such frameworks for multiple tasks, specially by employing multiple encoders. Using multiple encoders has been shown to be useful in mutli-task learning (Luong et al., 2015a), multi-source translation (Zoph and Knight, 2016) and reading comprehension (Xiong et al., 2017). We are the first to explore a multi-source encoder-decoder architecture for extracting tuples from cQA datasets.

Traditional encoder-decoder architectures are not tailored for information extraction and knowledge harvesting. To make them suitable for information extraction, the sequence generation must be subjected to several constraints on the vocabulary, grammar etc. Recently, grammar structures have been integrated into encoder-decoder models (Iyer et al., 2017; Zhang et al., 2017). There are variations such as Pointer Networks (Vinyals et al., 2015) that yield a succession of pointers to tokens in the input sequence. All these studies share a common idea with our paper, which is to enforce constraints at sequence generation time. Since we focus on extraction from cQA datasets, our work is broadly related to the literature on relation extraction (Savenkov et al., 2015; Hixon et al., 2015; Wu et al., 2018) and ontology extraction (S and Kumar, 2018) from community generated question-answer datasets. However, we differ in our underlying assumption that the relations and entities of interest are not known in advance. Alternatively, a cQA dataset could be transformed into declarative sentences (Demszky et al., 2018) for a conventional OpenIE system. However, such a two-stage approach is susceptible to error propagation. We adopt an end-to-end solution that is applicable to generic cQA datasets.

7 Conclusions and Future Work

We have presented NeurON, a system for extracting structured data from QA pairs for the purpose of enriching knowledge bases. NeurON uses a multi-encoder, constrained-decoder framework to generate quality tuples from QA pairs.

NeurON achieves the highest precision and recall in extracting tuples from QA pairs compared with state-of-the-art sentence-based models, with a relative improvement as high as 13.3%. It can discover 15-25% more tuples which makes it suitable as a tuple extraction tool for KB extension.

There are several directions for future research. One interesting direction is to investigate whether NeurON can be extended to work on *open-domain QA corpus*, which may not be restricted to any specific domain.

Acknowledgements

We thank Tom Mitchell and the anonymous reviewers for their constructive feedback. This work was supported in part by the UM Office of Research.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proc. ACL '15/IJCNLP '15*, pages 344–354.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR '15*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007.

Open information extraction from the web. In *Proc. IJCAI '07*, pages 2670–2676.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. EMNLP '13*, pages 1533–1544.

Nikita Bhutani, HV Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proc. EMNLP '16*, pages 55–64.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. NIPS '13*, pages 2787–2795.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP '14*, pages 1724–1734.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *Proc. ACL '18*, pages 407–413.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. EMNLP '11*, pages 1535–1545.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proc. NAACL-HLT '15*, pages 851–861.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proc. ACL '17*, pages 963–973.

Prachi Jain, Shikhar Murty, Mausam, and Soumen Chakrabarti. 2018. Mitigating the effect of out-of-vocabulary entity pairs in matrix factorization for KB inference. In *Proc. IJCAI '18*, pages 4122–4129.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL '17 (System Demonstrations)*, pages 67–72.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. In *Proc. ICLR '16*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP '15*, pages 1412–1421.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proc. EMNLP '12*, pages 523–534.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proc. WWW '16*, pages 625–635.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proc. AAAI '16*, pages 1955–1961.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP '14*, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proc. EMNLP '17*, pages 338–348.

Subhashree S and P Sreenivasa Kumar. 2018. Enriching domain ontologies using question-answer datasets. In *Proc. CoDS-COMAD '18*, pages 329–332.

Swarnadeep Saha, Harinder Pal, et al. 2017. Bootstrapping for numerical open ie. In *Proc. ACL '17*, pages 317–323.

Denis Savenkov, Wei-Lwun Lu, Jeff Dalton, and Eugene Agichtein. 2015. Relation extraction from community generated question-answer pairs. In *Proc. NAACL-HLT '15*, pages 96–102.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proc. EMNLP '16*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proc. ACL '18*, pages 885–895.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proc. EACL '17*, pages 1063–1073.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. NIPS '15*, pages 2692–2700.

Mengting Wan and Julian McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *Proc. ICDM '16*, pages 489–498.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zeqiu Wu, Xiang Ren, Frank F. Xu, Ji Li, and Jiawei Han. 2018. Indirect supervision for relation extraction using question-answer pairs. In *Proc. WSDM '18*, pages 646–654.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proc. ACL '16*, pages 1341–1350.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proc. ICLR '17*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. TextRunner: Open information extraction on the web. In *Proc. NAACL-HLT '07 (Demonstrations)*, pages 25–26.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proc. ACL '17*, pages 440–450.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Cseq2seq: Cyclic sequence-to-sequence learning. *arXiv preprint arXiv:1607.08725*.

Yaoyuan Zhang, Zhenxu Ye, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. *arXiv preprint arXiv:1704.02312*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proc. NAACL-HLT '16*, pages 30–34.

A Supplementary Material

This supplementary material contains details of the analysis settings described in Section 4 and additional results not reported in the main paper.

A.1 Word Embedding Analysis

We investigated the embedding layers of the question encoder and the answer encoder of the NEURON model trained on the *ConciergeQA* dataset.

For robust analysis, we ignored non-English words and any words that contained numerical digits (e.g., #18D, $10). We used PYENCHANT[2] for filtering English words. For the remaining words, we find their embedding vectors from the two encoders, concatenate them to create a single matrix. This ensures that same embedding vectors are mapped to the same point in the visualization space. We used t-SNE[3] to map embedding vectors into 2D space for visualization.

A.2 Crowdsourced Evaluation

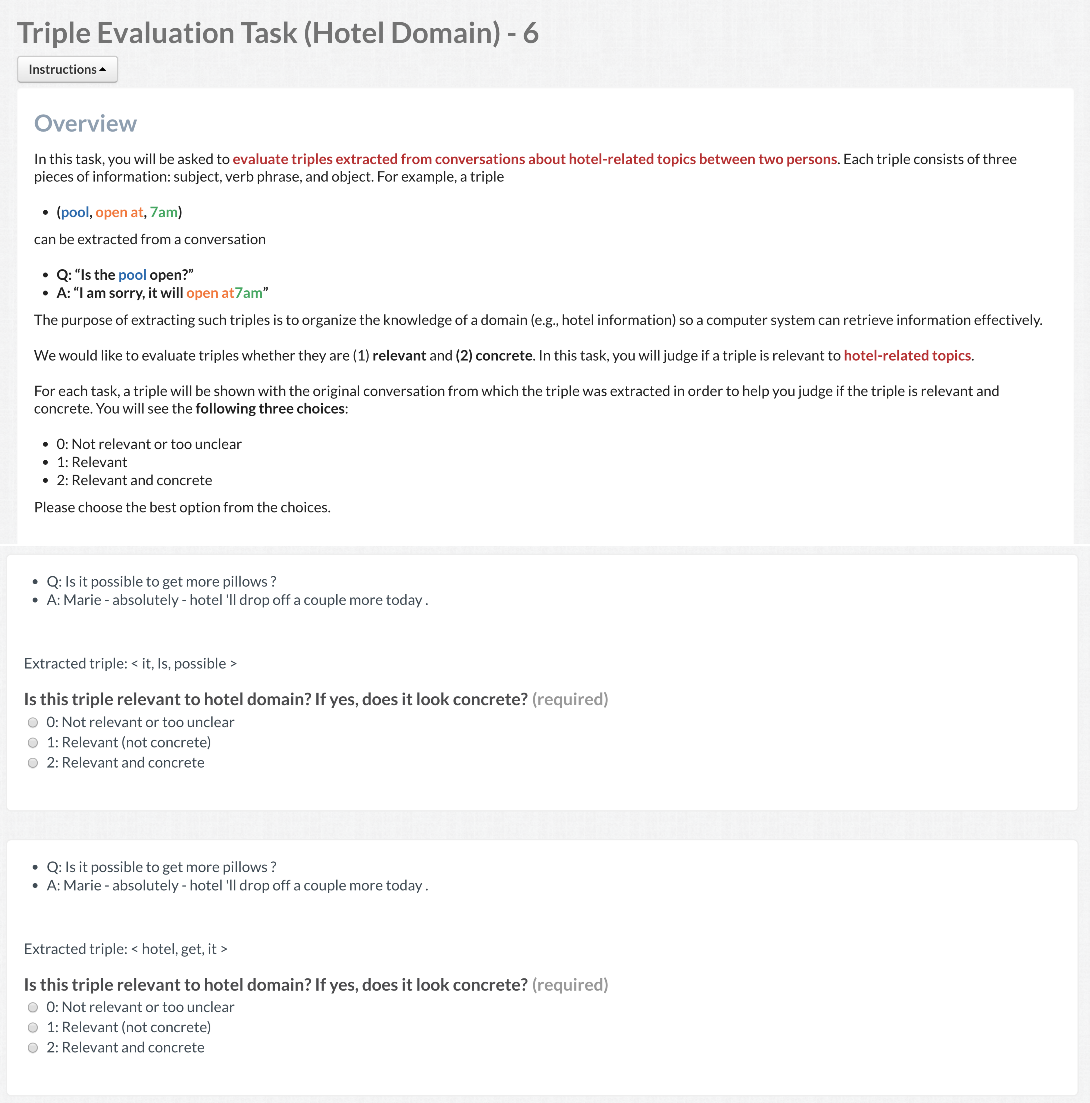
Figure 5 shows the instructions and examples of the crowdsouced task. In the crowdsourcing task, crowdsourced workers were asked to judge after reading an extracted tuple with the original QA pair. Since it is difficult to define the usefulness of the tuples without assuming a KB, we used *relevance* and *concreteness* as criteria to grade extracted tuples. Specifically, each worker was asked to choose one option from the three options: `Not relevant or unclear` (0), `Relevant` (1), `Relevant and concrete` (2).

We set $0.05 as payment for each annotation. We carefully created 51 test questions which were used to filter out untrusted judgments and workers. The platform increases the number of annotators so each tuple should always have 5 trusted annotators. The 5 annotations for each tuple were aggregated into a single label with a confidence value that takes into account the accuracy rates of the annotators based on the test questions.

---

[2] v2.0.0 https://github.com/rfk/pyenchant

[3] TSNE v0.20.0 https://scikit-learn.org/stable/ with default configuration



Triple Evaluation Task (Hotel Domain) - 6

Overview

In this task, you will be asked to evaluate triples extracted from conversations about hotel-related topics between two persons. Each triple consists of three pieces of information: subject, verb phrase, and object. For example, a triple

- (pool, open at, 7am)

can be extracted from a conversation

- **Q: "Is the pool open?"**
- **A: "I am sorry, it will open at7am"**

The purpose of extracting such triples is to organize the knowledge of a domain (e.g., hotel information) so a computer system can retrieve information effectively.

We would like to evaluate triples whether they are (1) **relevant** and **(2) concrete**. In this task, you will judge if a triple is relevant to hotel-related topics.

For each task, a triple will be shown with the original conversation from which the triple was extracted in order to help you judge if the triple is relevant and concrete. You will see the **following three choices**:

- 0: Not relevant or too unclear
- 1: Relevant
- 2: Relevant and concrete

Please choose the best option from the choices.

- Q: Is it possible to get more pillows ?
- A: Marie - absolutely - hotel 'll drop off a couple more today .

Extracted triple: < it, Is, possible >

Is this triple relevant to hotel domain? If yes, does it look concrete? (required)

○ 0: Not relevant or too unclear
○ 1: Relevant (not concrete)
○ 2: Relevant and concrete

- Q: Is it possible to get more pillows ?
- A: Marie - absolutely - hotel 'll drop off a couple more today .

Extracted triple: < hotel, get, it >

Is this triple relevant to hotel domain? If yes, does it look concrete? (required)

○ 0: Not relevant or too unclear
○ 1: Relevant (not concrete)
○ 2: Relevant and concrete

Figure 5: Screenshot of the instructions and examples of the crowdsourced task.

Question Answering by Reasoning Across Documents with Graph Convolutional Networks

Nicola De Cao
University of Edinburgh
University of Amsterdam
nicola.decao@uva.nl

Wilker Aziz
University of Amsterdam
w.aziz@uva.nl

Ivan Titov
University of Edinburgh
University of Amsterdam
ititov@inf.ed.ac.uk

Abstract

Most research in reading comprehension has focused on answering questions based on individual documents or even single paragraphs. We introduce a neural model which integrates and reasons relying on information spread within documents and across multiple documents. We frame it as an inference problem on a graph. Mentions of entities are nodes of this graph while edges encode relations between different mentions (e.g., within- and cross-document coreference). Graph convolutional networks (GCNs) are applied to these graphs and trained to perform multi-step reasoning. Our Entity-GCN method is scalable and compact, and it achieves state-of-the-art results on a multi-document question answering dataset, WIKIHOP (Welbl et al., 2018).

1 Introduction

The long-standing goal of natural language understanding is the development of systems which can acquire knowledge from text collections. Fresh interest in reading comprehension tasks was sparked by the availability of large-scale datasets, such as SQuAD (Rajpurkar et al., 2016) and CNN/Daily Mail (Hermann et al., 2015), enabling end-to-end training of neural models (Seo et al., 2016; Xiong et al., 2016; Shen et al., 2017). These systems, given a text and a question, need to answer the query relying on the given document. Recently, it has been observed that most questions in these datasets do not require reasoning across the document, but they can be answered relying on information contained in a single sentence (Weissenborn et al., 2017). The last generation of large-scale reading comprehension datasets, such as a NarrativeQA (Kocisky et al., 2018), TriviaQA (Joshi et al., 2017), and RACE (Lai et al., 2017), have been created in such a way as to address this shortcoming and to ensure that systems relying only on local information cannot achieve competitive performance.

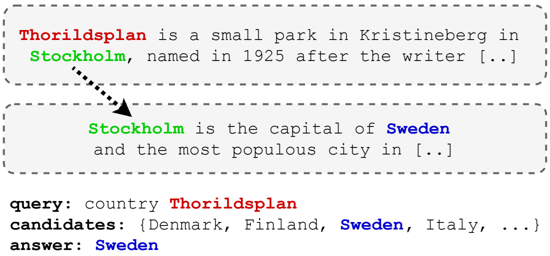


Figure 1: A sample from WIKIHOP where multi-step reasoning and information combination from different documents is necessary to infer the correct answer.

Even though these new datasets are challenging and require reasoning within documents, many question answering and search applications require aggregation of information across multiple documents. The WIKIHOP dataset (Welbl et al., 2018) was explicitly created to facilitate the development of systems dealing with these scenarios. Each example in WIKIHOP consists of a collection of documents, a query and a set of candidate answers (Figure 1). Though there is no guarantee that a question cannot be answered by relying just on a single sentence, the authors ensure that it is answerable using a chain of reasoning crossing document boundaries.

Though an important practical problem, the multi-hop setting has so far received little attention. The methods reported by Welbl et al. (2018) approach the task by merely concatenating all documents into a single long text and training a standard RNN-based reading comprehension model, namely, BiDAF (Seo et al., 2016) and FastQA (Weissenborn et al., 2017). Document concatenation in this setting is also used in Weaver (Raison et al., 2018) and MHPGM (Bauer et al., 2018). The only published paper which

Proceedings of NAACL-HLT 2019, pages 2306–2317
Minneapolis, Minnesota, June 2 - June 7, 2019.

goes beyond concatenation is due to Dhingra et al. (2018), where they augment RNNs with jump-links corresponding to co-reference edges. Though these edges provide a structural bias, the RNN states are still tasked with passing the information across the document and performing multi-hop reasoning.

Instead, we frame question answering as an inference problem on a graph representing the document collection. Nodes in this graph correspond to named entities in a document whereas edges encode relations between them (e.g., cross- and within-document coreference links or simply co-occurrence in a document). We assume that reasoning chains can be captured by propagating local contextual information along edges in this graph using a graph convolutional network (GCN) (Kipf and Welling, 2017).

The multi-document setting imposes scalability challenges. In realistic scenarios, a system needs to learn to answer a query for a given collection (e.g., Wikipedia or a domain-specific set of documents). In such scenarios one cannot afford to run expensive document encoders (e.g., RNN or transformer-like self-attention (Vaswani et al., 2017)), unless the computation can be preprocessed both at train and test time. Even if (similarly to WIKIHOP creators) one considers a coarse-to-fine approach, where a set of potentially relevant documents is provided, re-encoding them in a query-specific way remains the bottleneck. In contrast to other proposed methods (e.g., (Dhingra et al., 2018; Raison et al., 2018; Seo et al., 2016)), we avoid training expensive document encoders.

In our approach, only a small query encoder, the GCN layers and a simple feed-forward answer selection component are learned. Instead of training RNN encoders, we use contextualized embeddings (ELMo) to obtain initial (local) representations of nodes. This implies that only a lightweight computation has to be performed online, both at train and test time, whereas the rest is preprocessed. Even in the somewhat contrived WIKIHOP setting, where fairly small sets of candidates are provided, the model is at least 5 times faster to train than BiDAF.[1] Interestingly, when we substitute ELMo with simple pre-trained word embeddings, Entity-GCN still performs on par with many techniques that use expensive question-aware recurrent document encoders.

Despite not using recurrent document encoders, the full Entity-GCN model achieves over 2% improvement over the best previously-published results. As our model is efficient, we also reported results of an ensemble which brings further 3.6% of improvement and only 3% below the human performance reported by Welbl et al. (2018). Our contributions can be summarized as follows:

- we present a novel approach for multi-hop QA that relies on a (pre-trained) document encoder and information propagation across multiple documents using graph neural networks;
- we provide an efficient training technique which relies on a slower offline and a faster on-line computation that does not require expensive document processing;
- we empirically show that our algorithm is effective, presenting an improvement over previous results.

2 Method

In this section we explain our method. We first introduce the dataset we focus on, WIKIHOP by Welbl et al. (2018), as well as the task abstraction. We then present the building blocks that make up our Entity-GCN model, namely, an *entity graph* used to relate mentions to entities within and across documents, a *document encoder* used to obtain representations of mentions in context, and a *relational graph convolutional network* that propagates information through the entity graph.

2.1 Dataset and Task Abstraction

Data The WIKIHOP dataset comprises of tuples $\langle q, S_q, C_q, a^\star \rangle$ where: q is a query/question, S_q is a set of supporting documents, C_q is a set of candidate answers (all of which are entities mentioned in S_q), and $a^\star \in C_q$ is the entity that correctly answers the question. WIKIHOP is assembled assuming that there exists a corpus and a knowledge base (KB) related to each other. The KB contains triples $\langle s, r, o \rangle$ where s is a subject entity, o an object entity, and r a unidirectional relation between them. Welbl et al. (2018) used WIKIPEDIA as corpus and WIKIDATA (Vrandečić, 2012) as KB. The KB is only used for constructing WIKIHOP: Welbl

---

[1] When compared to the ‘small’ and hence fast BiDAF model reported in Welbl et al. (2018), which is 25% less accurate than our Entity-GCN. Larger RNN models are problematic also because of GPU memory constraints.

et al. (2018) retrieved the supporting documents S_q from the corpus looking at mentions of subject and object entities in the text. Note that the set S_q (not the KB) is provided to the QA system, and not all of the supporting documents are relevant for the query but some of them act as distractors. Queries, on the other hand, are not expressed in natural language, but instead consist of tuples $\langle s, r, ? \rangle$ where the object entity is unknown and it has to be inferred by reading the support documents. Therefore, answering a query corresponds to finding the entity $a^\star$ that is the object of a tuple in the KB with subject s and relation r among the provided set of candidate answers C_q.

Task The goal is to learn a model that can identify the correct answer $a^\star$ from the set of supporting documents S_q. To that end, we exploit the available supervision to train a neural network that computes scores for candidates in C_q. We estimate the parameters of the architecture by maximizing the likelihood of observations. For prediction, we then output the candidate that achieves the highest probability. In the following, we present our model discussing the design decisions that enable multi-step reasoning and an efficient computation.

2.2 Reasoning on an Entity Graph

Entity graph In an offline step, we organize the content of each training instance in a graph connecting mentions of candidate answers within and across supporting documents. For a given query $q = \langle s, r, ? \rangle$, we identify mentions in S_q of the entities in $C_q \cup \{s\}$ and create one node per mention. This process is based on the following heuristic:

1. we consider mentions spans in S_q exactly matching an element of $C_q \cup \{s\}$. Admittedly, this is a rather simple strategy which may suffer from low recall.

2. we use predictions from a coreference resolution system to add mentions of elements in $C_q \cup \{s\}$ beyond exact matching (including both noun phrases and anaphoric pronouns). In particular, we use the end-to-end coreference resolution by Lee et al. (2017).

3. we discard mentions which are ambiguously resolved to multiple coreference chains; this may sacrifice recall, but avoids propagating ambiguity.

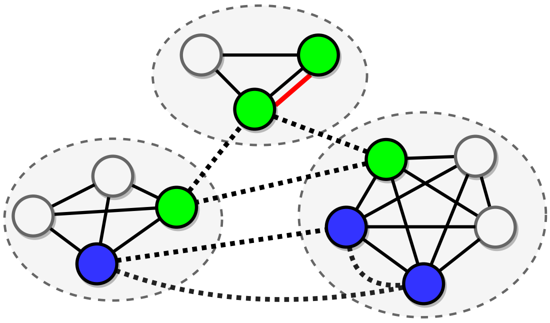


Figure 2: Supporting documents (dashed ellipses) organized as a graph where nodes are mentions of either candidate entities or query entities. Nodes with the same color indicates they refer to the same entity (exact match, coreference or both). Nodes are connected by three simple relations: one indicating co-occurrence in the same document (solid edges), another connecting mentions that exactly match (dashed edges), and a third one indicating a coreference (bold-red line).

To each node v_i, we associate a continuous annotation $\mathbf{x}_i \in \mathbb{R}^D$ which represents an entity in the context where it was mentioned (details in Section 2.3). We then proceed to connect these mentions i) if they co-occur within the same document (we will refer to this as `DOC-BASED` edges), ii) if the pair of named entity mentions is identical (`MATCH` edges—these may connect nodes across and within documents), or iii) if they are in the same coreference chain, as predicted by the external coreference system (`COREF` edges). Note that `MATCH` edges when connecting mentions in the same document are mostly included in the set of edges predicted by the coreference system. Having the two types of edges lets us distinguish between less reliable edges provided by the coreference system and more reliable (but also more sparse) edges given by the exact-match heuristic. We treat these three types of connections as three different types of relations. See Figure 2 for an illustration. In addition to that, and to prevent having disconnected graphs, we add a fourth type of relation (`COMPLEMENT` edge) between any two nodes that are not connected with any of the other relations. We can think of these edges as those in the complement set of the entity graph with respect to a fully connected graph.

Multi-step reasoning Our model then approaches multi-step reasoning by transforming node representations (Section 2.3 for details) with a differentiable message passing algorithm that propagates information through the entity graph. The algorithm is parameterized by a graph convolutional network (GCN) (Kipf and Welling, 2017), in particular, we employ relational-GCNs (Schlichtkrull et al., 2018), an extended version that accommodates edges of different types. In Section 2.4 we describe the propagation rule.

Each step of the algorithm (also referred to as a *hop*) updates all node representations in parallel. In particular, a node is updated as a function of messages from its direct neighbours, and a message is possibly specific to a certain relation. At the end of the first step, every node is aware of every other node it connects directly to. Besides, the neighbourhood of a node may include mentions of the same entity as well as others (e.g., same-document relation), and these mentions may have occurred in different documents. Taking this idea recursively, each further step of the algorithm allows a node to indirectly interact with nodes already known to their neighbours. After L layers of R-GCN, information has been propagated through paths connecting up to $L + 1$ nodes.

We start with node representations $\{\mathbf{h}_i^{(0)}\}_{i=1}^N$, and transform them by applying L layers of R-GCN obtaining $\{\mathbf{h}_i^{(L)}\}_{i=1}^N$. Together with a representation $\mathbf{q}$ of the query, we define a distribution over candidate answers and we train maximizing the likelihood of observations. The probability of selecting a candidate $c \in C_q$ as an answer is then

$$P(c|q, C_q, S_q) \propto \exp\left(\max_{i \in \mathcal{M}_c} f_o([\mathbf{q}, \mathbf{h}_i^{(L)}])\right) , \tag{1}$$

where f_o is a parameterized affine transformation, and $\mathcal{M}_c$ is the set of node indices such that $i \in \mathcal{M}_c$ only if node v_i is a mention of c. The $\max$ operator in Equation 1 is necessary to select the node with highest predicted probability since a candidate answer is realized in multiple locations via different nodes.

2.3 Node Annotations

Keeping in mind we want an efficient model, we encode words in supporting documents and in the query using only a pre-trained model for contextualized word representations rather than training our own encoder. Specifically, we use ELMo[2] (Peters et al., 2018), a pre-trained bi-directional language model that relies on character-based input representation. ELMo representations, differently from other pre-trained word-based models (e.g., *word2vec* (Mikolov et al., 2013) or GloVe (Pennington et al., 2014)), are contextualized since each token representation depends on the entire text excerpt (i.e., the whole sentence).

We choose not to fine tune nor propagate gradients through the ELMo architecture, as it would have defied the goal of not having specialized RNN encoders. In the experiments, we will also ablate the use of ELMo showing how our model behaves using non-contextualized word representations (we use GloVe).

Documents pre-processing ELMo encodings are used to produce a set of representations $\{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^D$ denotes the ith candidate mention in context. Note that these representations do not depend on the query yet and no trainable model was used to process the documents so far, that is, we use ELMo as a fixed pre-trained encoder. Therefore, we can pre-compute representation of mentions once and store them for later use.

Query-dependent mention encodings ELMo encodings are used to produce a query representation $\mathbf{q} \in \mathbb{R}^K$ as well. Here, $\mathbf{q}$ is a concatenation of the final outputs from a bidirectional RNN layer trained to re-encode ELMo representations of words in the query. The vector $\mathbf{q}$ is used to compute a query-dependent representation of mentions $\{\hat{\mathbf{x}}_i\}_{i=1}^N$ as well as to compute a probability distribution over candidates (as in Equation 1). Query-dependent mention encodings $\hat{\mathbf{x}}_i = f_x(\mathbf{q}, \mathbf{x}_i)$ are generated by a trainable function f_x which is parameterized by a feed-forward neural network.

2.4 Entity Relational Graph Convolutional Network

Our model uses a gated version of the original R-GCN propagation rule. At the first layer, all hidden node representation are initialized with the query-aware encodings $\mathbf{h}_i^{(0)} = \hat{\mathbf{x}}_i$. Then, at each layer $0 \leq \ell \leq L$, the update message $\mathbf{u}_i^{(\ell)}$ to the ith node is a sum of a transformation f_s of the current node representation $\mathbf{h}_i^{(\ell)}$ and transformations of its neighbours:

$$\mathbf{u}_i^{(\ell)} = f_s(\mathbf{h}_i^{(\ell)}) + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{ij}} f_r(\mathbf{h}_j^{(\ell)}) , \tag{2}$$

---

[2] The use of ELMo is an implementation choice, and, in principle, any other contextual pre-trained model could be used (Radford et al., 2018; Devlin et al., 2018).

where $\mathcal{N}_i$ is the set of indices of nodes neighbouring the ith node, $\mathcal{R}_{ij}$ is the set of edge annotations between i and j, and f_r is a parametrized function specific to an edge type $r \in \mathcal{R}$. Recall the available relations from Section 2.2, namely, $\mathcal{R} = \{\texttt{DOC-BASED}, \texttt{MATCH}, \texttt{COREF}, \texttt{COMPLEMENT}\}$.

A gating mechanism regulates how much of the update message propagates to the next step. This provides the model a way to prevent completely overwriting past information. Indeed, if all necessary information to answer a question is present at a layer which is not the last, then the model should learn to stop using neighbouring information for the next steps. Gate levels are computed as

$$\mathbf{a}_i^{(\ell)} = \sigma\left(f_a\left([\mathbf{u}_i^{(\ell)}, \mathbf{h}_i^{(\ell)}]\right)\right) , \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function and f_a a parametrized transformation. Ultimately, the updated representation is a gated combination of the previous representation and a non-linear transformation of the update message:

$$\mathbf{h}_i^{(\ell+1)} = \phi(\mathbf{u}_i^{(\ell)}) \odot \mathbf{a}_i^{(\ell)} + \mathbf{h}_i^{(\ell)} \odot (1 - \mathbf{a}_i^{(\ell)}) , \tag{4}$$

where $\phi(\cdot)$ is any nonlinear function (we used $\tanh$) and $\odot$ stands for element-wise multiplication. All transformations f_* are affine and they are not layer-dependent (since we would like to use as few parameters as possible to decrease model complexity promoting efficiency and scalability).

3 Experiments

In this section, we compare our method against recent work as well as preforming an ablation study using the WikiHop dataset (Welbl et al., 2018). See Appendix A in the supplementary material for a description of the hyper-parameters of our model and training details.

WikiHop We use WikiHop for training, validation/development and test. The test set is not publicly available and therefore we measure performance on the validation set in almost all experiments. WikiHop has 43,738/ 5,129/ 2,451 query-documents samples in the training, validation and test sets respectively for a total of 51,318 samples. Authors constructed the dataset as described in Section 2.1 selecting samples with a graph traversal up to a maximum chain length of 3 documents (see Table 1 for additional dataset statistics). WikiHop comes in two versions, a standard (unmasked) one and a masked one. The masked version was created by the authors to test whether methods are able to learn lexical abstraction. In this version, all candidates and all mentions of them in the support documents are replaced by random but consistent placeholder tokens. Thus, in the masked version, mentions are always referred to via unambiguous surface forms. We do not use coreference systems in the masked version as they rely crucially on lexical realization of mentions and cannot operate on masked tokens.

	Min	**Max**	**Avg.**	**Median**
# candidates	2	79	19.8	14
# documents	3	63	13.7	11
# tokens/doc.	4	2,046	100.4	91

Table 1: WikiHop dataset statistics from Welbl et al. (2018): number of candidates and documents per sample and document length.

3.1 Comparison

In this experiment, we compare our Entity-GCN against recent prior work on the same task. We present test and development results (when present) for both versions of the dataset in Table 2. From Welbl et al. (2018), we list an oracle based on human performance as well as two standard reading comprehension models, namely BiDAF (Seo et al., 2016) and FastQA (Weissenborn et al., 2017). We also compare against Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver (Raison et al., 2018). Additionally, we include results of MHQA-GRN (Song et al., 2018), from a recent arXiv preprint describing concurrent work. They jointly train graph neural networks and recurrent encoders. We report single runs of our two best single models and an ensemble one on the unmasked test set (recall that the test set is not publicly available and the task organizers only report unmasked results) as well as both versions of the validation set.

Entity-GCN (best single model without coreference edges) outperforms all previous work by over 2% points. We additionally re-ran BiDAF baseline to compare training time: when using a single Titan X GPU, BiDAF and Entity-GCN process 12.5 and 57.8 document sets per second, respectively. Note that Welbl et al. (2018) had to use BiDAF with very small state dimensionalities

Model	Unmasked Test	Unmasked Dev	Masked Test	Masked Dev
Human (Welbl et al., 2018)	74.1	–	–	–
FastQA (Welbl et al., 2018)	25.7	–	35.8	–
BiDAF (Welbl et al., 2018)	42.9	–	54.5	–
Coref-GRU (Dhingra et al., 2018)	59.3	56.0	–	–
MHPGM (Bauer et al., 2018)	–	58.2	–	–
Weaver / Jenga (Raison et al., 2018)	65.3	64.1	–	–
MHQA-GRN (Song et al., 2018)	65.4	62.8	–	–
Entity-GCN without coreference (single model)	**67.6**	**64.8**	–	70.5
Entity-GCN with coreference (single model)	**66.4**	**65.3**	–	–
Entity-GCN* (ensemble 5 models)	**71.2**	**68.5**	–	71.6

Table 2: Accuracy of different models on WIKIHOP closed test set and public validation set. Our Entity-GCN outperforms recent prior work without learning any language model to process the input but relying on a pre-trained one (ELMo – without fine-tunning it) and applying R-GCN to reason among entities in the text. * with coreference for unmasked dataset and without coreference for the masked one.

(20), and smaller batch size due to the scalability issues (both memory and computation costs). We compare applying the same reductions.[3] Eventually, we also report an ensemble of 5 independently trained models. All models are trained on the same dataset splits with different weight initializations. The ensemble prediction is obtained as $\arg\max_c \prod_{i=1}^{5} P_i(c|q, C_q, S_q)$ from each model.

3.2 Ablation Study

To help determine the sources of improvements, we perform an ablation study using the publicly available validation set (see Table 3). We perform two groups of ablation, one on the embedding layer, to study the effect of ELMo, and one on the edges, to study how different relations affect the overall model performance.

Embedding ablation We argue that ELMo is crucial, since we do not rely on any other context encoder. However, it is interesting to explore how our R-GCN performs without it. Therefore, in this experiment, we replace the deep contextualized embeddings of both the query and the nodes with GloVe (Pennington et al., 2014) vectors (insensitive to context). Since we do not have any component in our model that processes the documents, we expect a drop in performance. In other words, in this ablation our model tries to answer questions *without reading the context at all*. For example, in Figure 1, our model would be aware that "Stockholm" and "Sweden" appear in the same document but any context words, including the ones encoding relations (e.g., "is the capital of") will be hidden. Besides, in the masked case all mentions become 'unknown' tokens with GloVe and therefore the predictions are equivalent to a random guess. Once the strong pre-trained encoder is out of the way, we also ablate the use of our R-GCN component, thus completely depriving the model from inductive biases that aim at multi-hop reasoning.

The first important observation is that replacing ELMo by GloVe (GloVe with R-GCN in Table 3) still yields a competitive system that ranks far above baselines from (Welbl et al., 2018) and even above the Coref-GRU of Dhingra et al. (2018), in terms of accuracy on (unmasked) validation set. The second important observation is that if we then remove R-GCN (GloVe w/o R-GCN in Table 3), we lose 8.0 points. That is, the R-GCN component pushes the model to perform above Coref-GRU still without accessing context, but rather by updating mention representations based on their relation to other ones. These results highlight the impact of our R-GCN component.

Graph edges ablation In this experiment we investigate the effect of the different relations available in the entity graph and processed by the R-GCN module. We start off by testing our stronger encoder (i.e., ELMo) in absence of edges connecting mentions in the supporting documents (i.e., us-

---

[3] Besides, we could not run any other method we compare with combined with ELMo without reducing the dimensionality further or having to implement a distributed version.

Model	unmasked	masked
full (ensemble)	**68.5**	**71.6**
full (single)	**65.1** ± 0.11	**70.4** ± 0.12
GloVe with R-GCN	59.2	11.1
GloVe w/o R-GCN	51.2	11.6
No R-GCN	62.4	63.2
No relation types	62.7	63.9
No `DOC-BASED`	62.9	65.8
No `MATCH`	64.3	67.4
No `COREF`	64.8	–
No `COMPLEMENT`	64.1	70.3
Induced edges	61.5	56.4

Table 3: Ablation study on WikiHop validation set. The *full model* is our Entity-GCN with all of its components and other rows indicate models trained without a component of interest. We also report baselines using GloVe instead of ELMo with and without R-GCN. For the *full model* we report $\text{mean} \pm 1 \text{ std}$ over 5 runs.

ing only self-loops – No R-GCN in Table 3). The results suggest that WikipHop genuinely requires multihop inference, as our best model is 6.1% and 8.4% more accurate than this local model, in unmasked and masked settings, respectively.[4] However, it also shows that ELMo representations capture predictive context features, without being explicitly trained for the task. It confirms that our goal of getting away with training expensive document encoders is a realistic one.

We then inspect our model's effectiveness in making use of the structure encoded in the graph. We start naively by fully-connecting all nodes within and across documents without distinguishing edges by type (No relation types in Table 3). We observe only marginal improvements with respect to ELMo alone (No R-GCN in Table 3) in both the unmasked and masked setting suggesting that a GCN operating over a naive entity graph would not add much to this task and a more informative graph construction and/or a more sophisticated parameterization is indeed needed.

Next, we ablate each type of relations independently, that is, we either remove connections of mentions that co-occur in the same document (`DOC-BASED`), connections between mentions matching exactly (`MATCH`), or edges predicted by the coreference system (`COREF`). The first thing to note is that the model makes better use of `DOC-BASED` connections than `MATCH` or `COREF` connections. This is mostly because i) the majority of the connections are indeed between mentions in the same document, and ii) without connecting mentions within the same document we remove important information since the model is unaware they appear closely in the document. Secondly, we notice that coreference links and complement edges seem to play a more marginal role. Though it may be surprising for coreference edges, recall that the `MATCH` heuristic already captures the easiest coreference cases, and for the rest the out-of-domain coreference system may not be reliable. Still, modelling all these different relations together gives our Entity-GCN a clear advantage. This is our best system evaluating on the development. Since Entity-GCN seems to gain little advantage using the coreference system, we report test results both with and without using it. Surprisingly, with coreference, we observe performance degradation on the test set. It is likely that the test documents are harder for the coreference system.[5]

We do perform one last ablation, namely, we replace our heuristic for assigning edges and their labels by a model component that predicts them. The last row of Table 3 (Induced edges) shows model performance when edges are not predetermined but predicted. For this experiment, we use a bilinear function $f_e(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) = \sigma\left(\hat{\mathbf{x}}_i^\top \mathbf{W}_e \hat{\mathbf{x}}_j\right)$ that predicts the importance of a single edge connecting two nodes i, j using the query-dependent representation of mentions (see Section 2.3). The performance drops below 'No R-GCN' suggesting that it cannot learn these dependencies on its own.

Most results are stronger for the masked settings even though we do not apply the coreference resolution system in this setting due to masking. It is not surprising as coreferred mentions are labeled with the same identifier in the masked version, even if their original surface forms did not match (Welbl et al. (2018) used Wikipedia links for masking). Indeed, in the masked version, an entity is always referred to via the same unique surface form (e.g., `MASK1`) within and across documents. In the unmasked setting, on the other hand, mentions to an entity may differ (e.g., "US" vs "United States") and they might not be retrieved by the coreference system we are employing, mak-

---

[4] Recall that all models in the ensemble use the same local representations, ELMo.

[5] Since the test set is hidden from us, we cannot analyze this difference further.

	Relation	Accuracy	P@2	P@5	Avg. $\vert C_q \vert$	Supports
	overall (ensemble)	68.5	81.0	94.1	20.4 ± 16.6	5129
	overall (single model)	65.3	79.7	92.9	20.4 ± 16.6	5129
3 best	member_of_political_party	85.5	95.7	98.6	5.4 ± 2.4	70
	record_label	83.0	93.6	99.3	12.4 ± 6.1	283
	publisher	81.5	96.3	100.0	9.6 ± 5.1	54
3 worst	place_of_birth	51.0	67.2	86.8	27.2 ± 14.5	309
	place_of_death	50.0	67.3	89.1	25.1 ± 14.3	159
	inception	29.9	53.2	83.1	21.9 ± 11.0	77

Table 4: Accuracy and precision at K (P@K in the table) analysis overall and per query type. Avg. $|C_q|$ indicates the average number of candidates with one standard deviation.

ing the task harder for all models. Therefore, as we rely mostly on exact matching when constructing our graph for the masked case, we are more effective in recovering coreference links on the masked rather than unmasked version.[6]

4 Error Analysis

In this section we provide an error analysis for our best single model predictions. First of all, we look at which type of questions our model performs well or poorly. There are more than 150 query types in the validation set but we filtered the three with the best and with the worst accuracy that have at least 50 supporting documents and at least 5 candidates. We show results in Table 4. We observe that questions regarding places (birth and death) are considered harder for Entity-GCN. We then inspect samples where our model fails while assigning highest likelihood and noticed two principal sources of failure i) a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA, and ii) a different degree of granularity (e.g., born in “London” vs “UK” could be considered both correct by a human but not when measuring accuracy). See Table 6 in the supplement material for some reported samples.

Secondly, we study how the model performance degrades when the input graph is large. In particular, we observe a negative Pearson’s correlation (-0.687) between accuracy and the number of candidate answers. However, the performance does not decrease steeply. The distribution of the number of candidates in the dataset peaks at 5 and has an average of approximately 20. Therefore, the model does not see many samples where there are a large number of candidate entities during training. Differently, we notice that as the number of nodes in the graph increases, the model performance drops but more gently (negative but closer to zero Pearson’s correlation). This is important as document sets can be large in practical applications. See Figure 3 in the supplemental material for plots.

5 Related Work

In previous work, BiDAF (Seo et al., 2016), FastQA (Weissenborn et al., 2017), Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver / Jenga (Raison et al., 2018) have been applied to multi-document question answering. The first two mainly focus on single document QA and Welbl et al. (2018) adapted both of them to work with WIKIHOP. They process each instance of the dataset by concatenating all $d \in S_q$ in a random order adding document separator tokens. They trained using the first answer mention in the concatenated document and evaluating exact match at test time. Coref-GRU, similarly to us, encodes relations between entity mentions in the document. Instead of using graph neural network layers, as we do, they augment RNNs with jump links corresponding to pairs of corefereed mentions. MHPGM uses a multi-attention mechanism in combination with external commonsense relations to perform multiple hops of reasoning. Weaver is a deep co-encoding model that uses several alternating bi-LSTMs to process the concatenated documents and the query.

Graph neural networks have been shown successful on a number of NLP tasks (Marcheggiani and Titov, 2017; Bastings et al., 2017; Zhang et al.,

---

[6] Though other systems do not explicitly link matching mentions, they similarly benefit from masking (e.g., masks essentially single out spans that contain candidate answers).

2018a), including those involving document level modeling (Peng et al., 2017). They have also been applied in the context of asking questions about knowledge contained in a knowledge base (Zhang et al., 2018b). In Schlichtkrull et al. (2018), GCNs are used to capture reasoning chains in a knowledge base. Our work and unpublished concurrent work by Song et al. (2018) are the first to study graph neural networks in the context of multi-document QA. Besides differences in the architecture, Song et al. (2018) propose to train a combination of a graph recurrent network and an RNN encoder. We do not train any RNN document encoders in this work.

6 Conclusion

We designed a graph neural network that operates over a compact graph representation of a set of documents where nodes are mentions to entities and edges signal relations such as within and cross-document coreference. The model learns to answer questions by gathering evidence from different documents via a differentiable message passing algorithm that updates node representations based on their neighbourhood. Our model outperforms published results where ablations show substantial evidence in favour of multi-step reasoning. Moreover, we make the model fast by using pre-trained (contextual) embeddings.

Acknowledgments

We would like to thank Johannes Welbl for helping to test our system on WikiHop. This project is supported by SAP Innovation Center Network, ERC Starting Grant BroadSem (678254) and the Dutch Organization for Scientific Research (NWO) VIDI 639.022.518. Wilker Aziz is supported by the Dutch Organisation for Scientific Research (NWO) VICI Grant nr. 277-89-002.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967. Association for Computational Linguistics.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 42–48, New Orleans, Louisiana. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1601–1611.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Kinga, D., and J. Ba Adam. "A method for stochastic optimization." International Conference on Learning Representations (ICLR).*, 5.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*.

Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI.*

Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. 2018. Weaver: Deep coencoding of questions and documents for machine reading. *In Proceedings of the International Conference on Machine Learning (ICML).*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *International Conference on Learning Representations (ICLR).*

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring Graph-structured Passage Representation for Multihop Reading Comprehension with Graph Neural Networks. *arXiv preprint arXiv:1809.02040.*

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, pages 1063–1064. ACM.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280. Association for Computational Linguistics.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604.*

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018a. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215. Association for Computational Linguistics.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018b. Variational reasoning for question answering with knowledge graph. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18).*

A Implementation and Experiments Details

A.1 Architecture

See table 5 for an outline of Entity-GCN architectural detail. Here the computational steps

1. ELMo embeddings are a concatenation of three 1024-dimensional vectors resulting in 3072-dimensional input vectors $\{\mathbf{x}_i\}_{i=1}^N$.

2. For the query representation $\mathbf{q}$, we apply 2 bi-LSTM layers of 256 and 128 hidden units to its ELMo vectors. The concatenation of the forward and backward states results in a 256-dimensional question representation.

3. ELMo embeddings of candidates are projected to 256-dimensional vectors, concatenated to the $\mathbf{q}$, and further transformed with a two layers MLP of 1024 and 512 hidden units in 512-dimensional query aware entity representations $\{\hat{\mathbf{x}}_i\}_{i=1}^N \in \mathbb{R}^{512}$.

4. All transformations f_* in R-GCN-layers are affine and they do maintain the input and output dimensionality of node representations the same (512-dimensional).

5. Eventually, a 2-layers MLP with [256, 128] hidden units takes the concatenation between $\{\mathbf{h}_i^{(L)}\}_{i=1}^N$ and $\mathbf{q}$ to predict the probability that a candidate node v_i may be the answer to the query q (see Equation 1).

During preliminary trials, we experimented with different numbers of R-GCN-layers (in the range 1-7). We observed that with WIKIHOP, for $L \geq 3$ models reach essentially the same performance, but more layers increase the time required to train them. Besides, we observed that the gating mechanism learns to keep more and more information from the past at each layer making unnecessary to have more layers than required.

A.2 Training Details

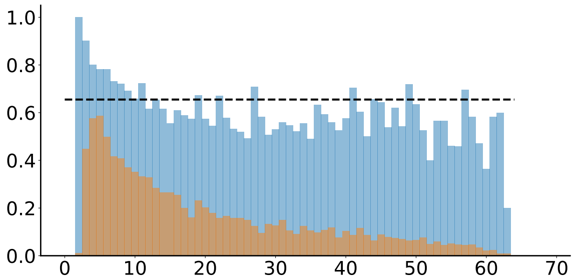
We train our models with a batch size of 32 for at most 20 epochs using the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 10^{-4}. To help against overfitting, we employ dropout (drop rate $\in 0, 0.1, 0.15, 0.2, 0.25$) (Srivastava et al., 2014) and early-stopping on validation accuracy. We report the best results of each experiment based on accuracy on validation set.

B Error Analysis

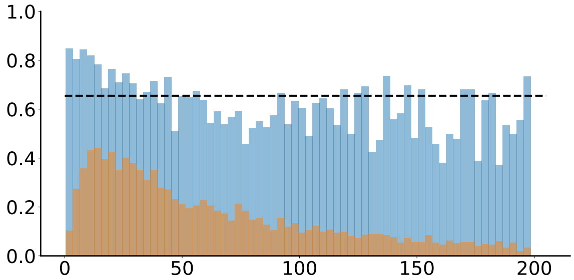
In Table 6, we report three samples from WIKIHOP development set where out Entity-GCN fails. In particular, we show two instances where our model presents high confidence on the answer, and one where is not. We commented these samples explaining why our model might fail in these cases.

C Ablation Study

In Figure 3, we show how the model performance goes when the input graph is large. In particular, how Entity-GCN performs as the number of candidate answers or the number of nodes increases.



(a) Candidates set size (x-axis) and accuracy (y-axis). Pearson's correlation of -0.687 ($p < 10^{-7}$).



(b) Nodes set size (x-axis) and accuracy (y-axis). Pearson's correlation of -0.385 ($p < 10^{-7}$).

Figure 3: Accuracy (blue) of our best single model with respect to the candidate set size (on the *top*) and nodes set size (on the *bottom*) on the validation set. Rescaled data distributions (orange) per number of candidate (*top*) and nodes (*bottom*). Dashed lines indicate average accuracy.

Input - q, $\{v_i\}_{i=1}^{N}$	
query ELMo 3072-dim	candidates ELMo 3072-dim
2 layers bi-LSTM [256, 128]-dim	1 layer FF 256-dim
concatenation 512-dim	
2 layer FF [1024, 512]-dim: : $\{\hat{\mathbf{x}}_i\}_{i=1}^{N}$	
3 layers R-GCN 512-dim each (shared parameters)	
concatenation with $\mathbf{q}$ 768-dim	
3 layers FF [256,128,1]-dim	
Output - probabilities over C_q	

Table 5: Model architecture.

ID	WH_dev_2257	**Gold answer**	2003 ($p = 14.1$)
Query	inception (of) Derrty Entertainment	**Predicted answer**	2000 ($p = 15.8$)
Support 1	**Derrty Entertainment** is a record label founded by [...]. The first album released under **Derrty Entertainment** was Nelly 's **Country Grammar**.		
Support 2	**Country Grammar** is the debut single by American rapper Nelly. The song was produced by Jason Epperson. It was released in **2000**, [...]		

(a) In this example, the model predicts the answer correctly. However, there is a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA. In WIKIHOP, answers are generated with WIKIDATA.

ID	WH_dev_2401	**Gold answer**	Adolph Zukor ($p = 7.1\text{e}{-4}\%$)
Query	producer (of) Forbidden Paradise	**Predicted answer**	Jesse L. Lask ($p = 99.9\%$)
Support 1	**Forbidden Paradise** is a [...] drama film produced by **Famous Players-Lasky** [...]		
Support 2	**Famous Players-Lasky** Corporation was [...] from the merger of **Adolph Zukor**'s Famous Players Film Company [..] and the **Jesse L. Lasky** Feature Play Company.		

(b) In this sample, there is ambiguity between two entities since both are correct answers reading the passages but only one is marked as correct. The model fails assigning very high probability to only on one of them.

ID	WH_dev_3030	**Gold answer**	Scania ($p = 0.029\%$)
Query	place_of_birth (of) Erik Penser	**Predicted answer**	Eslöv ($p = 97.3\%$)
Support 1	Nils Wilhelm **Erik Penser** (born August 22, 1942, in **Eslöv**, **Skåne**) is a Swedish [...]		
Support 2	**Skåne** County, sometimes referred to as " **Scania** County " in English, is the [...]		

(c) In this sample, there is ambiguity between two entities since the city Eslöv is located in the Scania County (English name of Skåne County). The model assigning high probability to the city and it cannot select the county.

Table 6: Samples from WIKIHOP set where Entity-GCN fails. p indicates the predicted likelihood.

A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC

Mark Yatskar
Allen Institute for Artificial Intelligence
`marky@allenai.org`

Abstract

We compare three new datasets for question answering: SQuAD 2.0, QuAC, and CoQA, along several of their new features: (1) unanswerable questions, (2) multi-turn interactions, and (3) abstractive answers. We show that the datasets provide complementary coverage of the first two aspects, but weak coverage of the third. Because of the datasets' structural similarity, a single extractive model can be easily adapted to any of the datasets and we show improved baseline results on both SQuAD 2.0 and CoQA. Despite the similarity, models trained on one dataset are ineffective on another dataset, but we find moderate performance improvement through pretraining. To encourage cross-evaluation, we release code for conversion between datasets at https://github.com/my89/co-squac.

1 Introduction

Question answering on textual data has served as a challenge problem for the NLP community (Voorhees, 2001; Richardson et al., 2013). With the development of large scale benchmarks and sufficiently simple evaluations (Trischler et al., 2016; Nguyen et al., 2016; Hermann et al., 2015) progress has been rapid. In recent evaluation on SQuAD (Rajpurkar et al., 2016), performance exceeded that of annotators (Wang et al., 2018; Hu et al., 2017; Wang et al., 2017).

In response to this development, there have been a flurry of new datasets. In this work, we analyze three such new proposed datasets, SQuAD 2.0 (Rajpurkar et al., 2018), QuAC (Choi et al., 2018), and CoQA (Reddy et al., 2018).[1] In each of these datasets, crowd workers are asked to (1) produce questions about a paragraph of text (context) and (2) produce a reply by either indicating there is no answer, or providing an extractive answer from the context by highlighting one contiguous span. QuAC and CoQA contain two other features: questions are asked in the form of a dialog, where co-reference to previous interactions is possible and directly answering yes/no is possible. CoQA also allows workers to edit the spans to provide abstractive answers.[2]

We compare these three datasets along several of their new features: (1) unanswerable questions, (2) multi-turn interactions, and (3) abstractive answers. Unanswerable question coverage is complementary among datasets; SQuAD 2.0 focuses more on questions of extreme confusion, such as false premise questions, while QuAC primarily focuses on missing information. QuAC and CoQA dialogs simulate different types of user behavior: QuAC dialogs often switch topics while CoQA dialogs include more queries for details. Unfortunately, no dataset provides significant coverage of abstractive answers beyond yes/no answers, and we show that a method can achieve an extractive answer upper bound of 100 and 97.8 F1 on QuAC and CoQA , respectively.

Motivated by the above analysis, we apply the baseline presented in QuAC (Choi et al., 2018), BiDAF++, a model based on BiDAF (Seo et al., 2016), augmented with self attention (Clark and Gardner, 2018) and ELMo contextualized embeddings (Peters et al., 2018) to all datasets. Experiments show that this extractive baseline outperforms existing extractive and abstractive baselines on CoQA by 14.2 and 2.7 F1 respectively. Finally, we show models can transfer between datasets with pretraining yielding moderate gains.[3]

---

[1] A review of other new datasets is in the related work.

[2] Also, SQuAD 2.0 and QuAC cover only Wikipedia text, CoQA covers six other domains and QuAC is the only one of these datasets that doesn't allow the questioner to see the context before formulating a question.

[3] To facilitate easy future cross-evaluation, we release tools for conversion between these dataset.

Proceedings of NAACL-HLT 2019, pages 2318–2323
Minneapolis, Minnesota, June 2 - June 7, 2019.

Dataset	Entity Salad	False Premise	Topic Error	Missing Information	Content Negation	Answerable Questions	Total Questions
CoQA	0.0	0.0	0.0	60.0	0.0	40.0	5 (0.5%)
SQuAD 2.0	21.3	21.3	13.5	16.1	16.1	10.9	230 (50.1%)
QuAC	5.5	0.0	16.4	71.2	0.0	6.8	73 (20.2%)

Table 1: Comparison of unanswerable questions on 50 random contexts from the development set of each dataset. SQuAD 2.0 contains a diverse set of circumstances that make questions unanswerable, QuAC focuses on information that could plausibly be in context material and CoQA does not significantly cover unanswerable questions.

Dataset	Topic Shift	Drill Down	Return to Topic	Clarification Question	Definition Question	Sentence Coverage	Total Questions
CoQA	21.6	72.0	2.9	0.0	0.7	63.3	722
QuAC	35.4	55.3	5.6	0.7	3.0	28.4	302

Table 2: Comparison of dialog features in 50 random contexts from the development set of each dataset. CoQA contains questions that drill into details about topics and cover 60% of sentences in the context while in QuAC dialog switch topic more often and cover less than 30% of sentences. Neither dataset has a significant number of returns to previous topics, clarifications, or definitional interactions.

2 Dataset Analysis

In this section we analyze unanswerable questions, dialog features, abstractive answers in SQuAD 2.0, QuAC, and CoQA. All analysis was performed by the authors, on a random sample of 50 contexts (300-700 questions) from the development set of each dataset.

2.1 Unanswerable Questions

In Table 1 we compare types of unanswerable questions across dataset. We identify five types of questions found between the datasets:

1. Entity Salad A nonsensical reference to entities found in the context or made-up entities (e.g. *"What infinite hierarchy implies that the graph isomorphism problem s NQ-complete?"*). Such questions are unanswerable for any context.

2. False Premise A fact that contradicts the context is asserted in the question (e.g. *"When is the correlation positive?"* but in the context says *"the correlation is strictly negative"*).

3. Topic Error A questions that references an entity in the context but the context does not focus on that entity (e.g *"How many earthquakes occur in California?"* when the article focus is actually about *"Southern California"*). Such questions potentially have answers, but it would be unlikely for the answer to be found in the context.

4. Missing Information A question who's answer could be plausibly in the context but is not (e.g. *"What is the record high in January?"* and the article is about temperature extremes). Such questions have an answer but it is not mentioned.

5. Content Negation A question which asks for the opposite information of something mentioned in the context (e.g. *"Who* **didnt** *cause the dissolution of the Holy Roman Empire?"*). Such questions either have answers that are the set of all entities other than the one mentioned or answers that could be found in some other context.

Results SQuAD 2.0 contains the highest diversity of unanswerable questions of all datasets analyzed. Some SQuAD 2.0 questions are unlikely to be asked without significant foreknowledge of the context material and do not occur in QuAC. [4] Both SQuAD 2.0 and QuAC cover a significant number of unanswerable questions that could be plausibly in the article. The difference in settings and distributions of unanswerable questions in SQuAD 2.0 and QuAC appear to be complementary: SQuAD 2.0 focuses more on questions simulating questioner confusion, while QuAC primarily focuses on missing information. [5]

2.2 Dialog Features

In Table 2 we analyze five dialog behaviors:

1. Topic Shift A question about something previously discussed (e.g. "Q: How does he try to take over? ... Q: Where do they live?").

2. Drill Down A request for more information about a topic being discussed (e.g. "A: The Sherpas call Mount Everest Chomolungma. Q: Is Mt. Everest a holy site for them?")

3. Topic Return Asking about a topic again after it had previously been shifted away from.

---

[4] Such questions resemble text from entailment datasets such as SNLI (Bowman et al., 2015) and seem more likely to arise if questioners are receiving very complex information and become confused.

[5] CoQA does not contain a significant number of unanswerable questions, and many of the ones that do exist are erroneously marked.

Dataset	Yes/No	Coref	Counting	Picking	Fluency	Max F1
CoQA	21.4	3.2	1.3	0.6	4.2	97.8
QuAC	21.1	0.0	0.0	0.0	0.0	100.0

Table 3: Comparison of abstractive features in 50 random contexts in the develoment set of each dataset. Both QuAC and CoQA contain yes/no questions while CoQA also contains answers that improve fluency through abstractive behavior. The extractive upper bound from CoQA is high because most absractivive answers involve adding a pronoun (Coref) or inserting prepositions and changing word forms (Fluency) to existing extractive answers, resulting in extremely high overlap with possible extractive answers.

4. Clarification Reformulating a question that had previously been asked.

5. Definition Asking what is meant by a term (e.g. "What are polygenes?")

Results QuAC and CoQA contain many similar features but at very different rates, offering complementary coverage of types of user behavior. CoQA dialogs drill down for details significantly more frequently and cover more than 60% of sentences in the context material (Sentence Coverage). QuAC dialogs shift to new topics frequently and cover less than 30% of sentences in the context. Both datasets contain only a small numbers of definition questions and returns to previous topics and few requests for clarification.

2.3 Abstractive Answers

Table 3 compares abstractive behavior in CoQA and QuAC. We observed five phenomena:

1. Yes/No Questions annotated with yes/no. In QuAC such questions and their corresponding yes or no are marked in addition to an extractive answer. In CoQA, the single token "yes" or "no" is simply asserted as the abstractive answer, with an extractive answer provided in the rationale (e.g. "Q: Is atmosphere one of them? A: yes").

2. Coref Coreference is added to previously mentioned entities in either context or question (e.g. "Q: How was France's economy in the late 2000s? A: **it** entered the recession").

3. Count Counting how many entities of some type were mentioned (e.g. "Q: how many specific genetic traits are named? A: five")

4. Picking A question that requires the answer to pick from a set defined in the question (e.g. "Q: Is this a boy or a girl? A: boy)

5. Fluency Adding a preposition, changing the form of a word, or merging two non-contiguous spans (e.g. "Q: how did he get away? A: **by** foot)

Results Both QuAC and CoQA have a similar rate of yes/no questions. QuAC contains no other abstractive phenomena while CoQA contains a small number of predominately insertions, often at the beginning of an extractive span, for coreference and or other fluency improvements. Because abstractive behavior in CoQA includes mostly small modifications to spans in the context, the maximum achievable performance by a model that predicts spans from the context is 97.8 F1. [6]

	Overall F1
DrQA (Extractive)	54.7
DrQA + PGNet (Abstractive)	66.2
BiDAF++ w/ 0-ctx	63.4
BiDAF++ w/ 3-ctx	**69.2**

Table 4: Development set performance by training BiDAF++ (Choi et al., 2018) models (extractive) on CoQA data with handling yes/no and no-answer questions as in QuAC. Despite being extractive, these models significantly outperform reported baselines, DrQA and DrQA + PGNet (Reddy et al., 2018).

	in-F1	out-F1	F1
DrQA	54.5	47.9	52.6
DrQA + PGNet	67.0	60.4	65.1
BiDAF++ w/ 3-ctx	**69.4**	**63.8**	**67.8**

Table 5: Test set results on CoQA. We report in domain F1 (in-F1), out of domain F1 on two held out domains, Reddit and Science (out-F1) and the overall F1 (F1).

3 New Extractive Baseline for CoQA

Our analysis strongly implies that beyond yes/no questions, abstractive behavior is not a significant component in either QuAC or CoQA. As such, QuAC models can be trivially adapted to CoQA.

We train a set of BiDAF++ baselines from the original QuAC dataset release (Choi et al., 2018) by optimizing the model to predict the span with maximum F1 overlap with respect to annotated abstractive answers.[7] If the abstractive answer is ex-

---

[6] To compute the upper bound, if abstractive answer is exactly "yes", "no", or "unknown", we consider the upper bound to be 100. Otherwise, we use the CoQA evaluation script to find a span in the context that has maximum F1 with respect to the abstractive answer.

[7] We use the implementation on http://allennlp.org, and do not modify any hyper-parameters except the the maximum dialog length and that models were allowed to train up to 65 epochs.

	F1	HEQQ	HEQD
BiDAF++ w/ 2-ctx	60.6	55.7	4.0
Train SQuAD 2.0	34.3	18.0	0.3
Train CoQA	31.2	19.2	0.0
Ft from SQuAD 2.0	62.6	58.3	**5.9**
Ft from CoQA	**63.3**	**59.2**	5.1

Table 6: Cross dataset transfer to QuAC development set. Models do not transfer directly (rows 3 and 4), but after fine tuning improve performance (rows 5 and 6).

actly "yes" or "no", we train the model to output the whole rationale span, and classify the question as yes/no with the appropriate answer. At evaluation time, if the model predicts a question is a yes/no question, instead of returning the extracted span, we simply return "yes" or "no".

Results Table 4 and Table 5 summarize our results for training BiDAF++ with varying contexts on CoQA. Beyond the difference of underlying base question-answer models (DrQA (Chen et al., 2017) vs. BiDAF (Seo et al., 2016) with self attention (Clark and Gardner, 2018)), BiDAF++ has two core differences with respect to DRQA+PGNet: (1) instead of appending previous questions and answers to input question tokens, BiDAF++ marks answers of previous questions directly on the context, and (2) BiDAF++ uses contextualized word embeddings through ELMo (Peters et al., 2018). These differences, in combination with appropriate handling of yes/no and unanswerable questions significantly improves on the existing extractive baseline **(+14.2 F1)** and even on the existing abstractive baseline **(+2.7 F1)**.

4 Cross-Dataset Experiments

In this section we consider whether models can benefit from transfer between SQuAD 2.0, QuAC, and CoQA, and show that the datasets, while ineffective for direct transfer, can be used as pretraining. In all experiments, we use BiDAF++, either with two context or no context, depending on if we are training for dialog settings or not, with default configurations. Models are trained by initializing from other models trained on different datasets and we do not decrease initial learning rates from just training directly on the target dataset. When SQuAD 2.0 is used to initialize models that use context, we randomly order questions in SQuAD 2.0 and train as if questions were asked in the form of a dialog. [8]

	In Domain F1
DrQA + PGNet	66.2
BiDAF++ w/ 2-ctx	67.6
SQuAD 2.0	41.4
QuAC	29.1
Ft from SQuAD 2.0	**69.2**
Ft from QuAC	68.0

Table 7: Cross dataset transfer to CoQA development set. Models do not transfer directly (rows 3 and 4), but after fine tuning improve performance (rows 5 and 6). For an explanation of why BiDAF++ outperforms DrQA + PGNet, see Section 3.

	F1	EM
Baseline	67.6	65.1
BiDAF++	70.5	67.4
CoQA	38.1	32.4
QuAC	25.4	16.8
Ft from CoQA	**72.5**	**69.4**
Ft from QuAC	69.5	66.8

Table 8: Cross dataset transfer to SQuAD 2.0 development set. BiDAF++ (Choi et al., 2018) outperforms the baseline, a different implementation of the same model (Rajpurkar et al., 2018) likely because of better hyper parameter tuning.

Results Tables 6-8 summarize our results. Across all of the datasets, BiDAF++ outperforms other baselines, and there exists at least one other dataset that significantly improves performance on a target dataset on average **+2.1 F1**. Experiments do not support that direct transfer is possible.

5 Related Work

Other proposals exist other than the three we analyzed that expand on features in SQuAD (Rajpurkar et al., 2016). For example, maintaining question independence of context to reduce the role of string matching and having long context length (Joshi et al., 2017; Kociský et al., 2017), higher level reasoning (Khashabi et al., 2018; Clark et al., 2018; Yang et al., 2018), multi-turn information seeking interactions, in either table settings (Iyyer et al., 2017; Talmor and Berant, 2018; Saha et al., 2018), regulation settings (Saeidi et al., 2018), or Quiz Bowl settings (Elgohary et al., 2018). Other work considers multi-modal contexts where interactions are a single turn (Tapaswi et al., 2016; Antol et al., 2015; Lei et al., 2018) or multi-turn (Das et al., 2017; Pasunuru and Bansal, 2018). These efforts contain alternative challenges than ones we analyze in this paper.

---

[8] Likely a better strategy exists but we would like to demonstrate transfer in the simplest way. We only report development numbers as these experiments are meant to be exploratory.

Acknowledgement

We thank Eunsol Choi, Hsin-Yuan Huang, Mohit Iyyer, He He, Yejin Choi, Percy Liang, and Luke Zettlemoyer for their helpful discussions in formulating this work. Also, Siva Reddy and Danqi Chen for help evaluating on CoQA and all reviewers for their comments.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Empirical Methods in Natural Language Processing*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. *Proceedings of the Association for Computational Linguistics*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Computer Vision and Pattern Recognition*.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. A dataset and baselines for sequential open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2017. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*.

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Association for Computational Linguistics*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, abs/1712.07040.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. In *EMNLP*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv*, abs/1611.09268.

Ramakanth Pasunuru and Mohit Bansal. 2018. Game-based video-context dialogue. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Conference of the North American Chapter of the Association for Computational Linguistics*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the Association for Computational Linguistics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *ArXiv*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Signh, Tim Rocktschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of Empirical Methods in Natural Language Processing*.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Association for the Advancement of Artificial Intelligence*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *Proceedings of the International Conference on Learning Representations*.

A. Talmor and J. Berant. 2018. The web as knowledge-base for answering complex questions. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Ellen M Voorhees. 2001. The trec question answering track. *Natural Language Engineering*, 7(4):361–378.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis

Hu Xu[1], **Bing Liu**[1], **Lei Shu**[1] and **Philip S. Yu**[1,2]
[1]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
[2]Institute for Data Science, Tsinghua University, Beijing, China
{hxu48, liub, lshu3, psyu}@uic.edu

Abstract

Question-answering plays an important role in e-commerce as it allows potential customers to actively seek crucial information about products or services to help their purchase decision making. Inspired by the recent success of machine reading comprehension (MRC) on formal documents, this paper explores the potential of turning customer reviews into a large source of knowledge that can be exploited to answer user questions. We call this problem *Review Reading Comprehension* (RRC). To the best of our knowledge, no existing work has been done on RRC. In this work, we first build an RRC dataset called ReviewRC based on a popular benchmark for aspect-based sentiment analysis. Since ReviewRC has limited training examples for RRC (and also for aspect-based sentiment analysis), we then explore a novel post-training approach on the popular language model BERT to enhance the performance of fine-tuning of BERT for RRC. To show the generality of the approach, the proposed post-training is also applied to some other review-based tasks such as aspect extraction and aspect sentiment classification in aspect-based sentiment analysis. Experimental results demonstrate that the proposed post-training is highly effective[1].

1 Introduction

For online commerce, question-answering (QA) serves either as a standalone application of customer service or as a crucial component of a dialogue system that answers user questions. Many intelligent personal assistants (such as Amazon Alexa and Google Assistant) support online shopping by allowing the user to speak directly to the assistants. One major hindrance for this mode of shopping is that such systems have limited capability to answer user questions about products (or services), which are vital for customer decision making. As such, an intelligent agent that can automatically answer customers' questions is very important for the success of online businesses.

Given the ever-changing environment of products and services, it is very hard, if not impossible, to pre-compile an up-to-date and reliable knowledge base to cover a wide assortment of questions that customers may ask, such as in factoid-based KB-QA (Xu et al., 2016; Fader et al., 2014; Kwok et al., 2001; Yin et al., 2015). As a compromise, many online businesses leverage community question-answering (CQA) (McAuley and Yang, 2016) to crowdsource answers from existing customers. However, the problem with this approach is that many questions are not answered, and if they are answered, the answers are delayed, which is not suitable for interactive QA. In this paper, we explore the potential of using product reviews as a large source of user experiences that can be exploited to obtain answers to user questions. Although there are existing studies that have used information retrieval (IR) techniques (McAuley and Yang, 2016; Yu and Lam, 2018) to find a whole review as the response to a user question, giving the whole review to the user is undesirable as it is quite time-consuming for the user to read it.

Inspired by the success of Machine Reading Comphrenesions (MRC) (Rajpurkar et al., 2016, 2018), we propose a novel task called Review Reading Comprehension (RRC) as following.

Problem Definition: Given a question $q = (q_1, \dots, q_m)$ from a customer (or user) about a product and a review $d = (d_1, \dots, d_n)$ for that product containing the information to answer q, find a sequence of tokens (a text span) $a = (d_s, \dots, d_e)$ in d that answers q correctly, where $1 \leq s \leq n$, $1 \leq e \leq n$, and $s \leq e$.

A sample *laptop* review is shown in Table 1. We can see that customers may not only ask factoid

---

[1] The datasets and code are available at `https://www.cs.uic.edu/~hxu/`.

Proceedings of NAACL-HLT 2019, pages 2324–2335
Minneapolis, Minnesota, June 2 - June 7, 2019.

Questions
Q1: Does it have an internal hard drive ?
Q2: How large is the internal hard drive ?
Q3: is the capacity of the internal hard drive OK ?
Review
Excellent value and a must buy for someone looking for a Macbook . You ca n't get any better than this price and it **come with**$_{A1}$ an internal disk drive . All the newer MacBooks do not . Plus you get **500GB**$_{A2}$ which is also a **great**$_{A3}$ feature . Also , the resale value on this will keep . I highly recommend you get one before they are gone .

Table 1: An example of review reading comprehension: we show 3 questions and their corresponding answer spans from a review.

questions such as the specs about some aspects of the laptop as in the first and second questions but also subjective or opinion questions about some aspects (capacity of the hard drive), as in the third question. RRC poses some *domain challenges* compared to the traditional MRC on Wikipedia, such as the need for rich product knowledge, informal text, and fine-grained opinions (there is almost no subjective content in Wikipedia articles). Research also shows that yes/no questions are very frequent for products with complicated specifications (McAuley and Yang, 2016; Xu et al., 2018b).

To the best of our knowledge, no existing work has been done in RRC. This work first builds an RRC dataset called ReviewRC, using reviews from SemEval 2016 Task 5[2], which is a popular dataset for aspect-based sentiment analysis (ABSA) (Hu and Liu, 2004) in the domains of *laptop* and *restaurant*. We detail ReviewRC in Sec. 5. Given the wide spectrum of domains (types of products or services) in online businesses and the prohibitive cost of annotation, ReviewRC can only be considered to have a limited number of annotated examples for supervised training, which still leaves the domain challenges partially unresolved.

This work adopts BERT (Devlin et al., 2018) as the base model as it achieves the state-of-the-art performance on MRC (Rajpurkar et al., 2016, 2018). Although BERT aims to learn contextualized representations across a wide range of NLP tasks (to be task-agnostic), leveraging BERT alone still leaves the domain challenges unresolved (as BERT is trained on Wikipedia articles and has almost no understanding of opinion text), and it also introduces another challenge of task-awareness (the RRC task), called the *task challenge*. This challenge arises when the task-agnostic BERT meets the limited number of fine-tuning examples in ReviewRC (see Sec. 5) for RRC, which is insufficient to fine-tune BERT to ensure full task-awareness of the system[3]. To address all the above challenges, we propose a novel joint post-training technique that takes BERT's pre-trained weights as the initialization[4] for basic language understanding and adapt BERT with both domain knowledge and task (MRC) knowledge before fine-tuning using the domain end task annotated data for the domain RRC. This technique leverages knowledge from two sources: unsupervised domain reviews and supervised (yet out-of-domain) MRC data [5], where the former enhances domain-awareness and the latter strengthens MRC task-awareness. As a general-purpose approach, we show that the proposed method can also benefit ABSA tasks such as aspect extraction (AE) and aspect sentiment classification (ASC).

The main contributions of this paper are as follows. (1) It proposes the new problem of *review reading comprehension* (RRC). (2) To solve this new problem, an annotated dataset for RRC is created. (3) It proposes a general-purpose post-training approach to improve RRC, AE, and ASC. Experimental results demonstrate that the proposed approach is effective.

2 Related Works

Many datasets have been created for MRC from formally written and objective texts, e.g., Wikipedia (WikiReading (Hewlett et al., 2016), SQuAD (Rajpurkar et al., 2016, 2018), WikiHop (Welbl et al., 2018), DRCD (Shao et al., 2018), QuAC (Choi et al., 2018), HotpotQA (Yang et al., 2018)) news and other articles (CNN/Daily Mail (Hermann et al., 2015), NewsQA (Trischler et al., 2016), RACE (Lai et al., 2017)), fictional stories (MCTest (Richardson et al., 2013),

---

[2] `http://alt.qcri.org/semeval2016/task5/`. We choose these review datasets to align RRC with existing research on sentiment analysis.

[3] The end tasks from the original BERT paper typically use tens of thousands of examples to ensure that the system is task-aware.

[4] Due to limited computation resources, it is impractical for us to pre-train BERT directly on reviews from scratch (Devlin et al., 2018).

[5] To simplify the writing, we refer MRC as a general-purpose RC task on formal text (non-review) and RRC as an end-task specifically focused on reviews.

CBT (Hill et al., 2015), NarrativeQA (Kočiskỳ et al., 2018)), and general Web documents (MS MARCO (Nguyen et al., 2016), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017)). Also, CoQA (Reddy et al., 2018) is built from multiple sources, such as Wikipedia, Reddit, News, Mid/High School Exams, Literature, etc. To the best of our knowledge, MRC has not been used on reviews, which are primarily subjective. As such, we created a review-based MRC dataset called ReviewRC. Answers from ReviewRC are extractive (similar to SQuAD (Rajpurkar et al., 2016, 2018)) rather than abstractive (or generative) (such as in MS MARCO (Nguyen et al., 2016) and CoQA (Reddy et al., 2018)). This is crucial because online businesses are typically cost-sensitive and extractive answers written by humans can avoid generating incorrect answers beyond the contents in reviews by an AI agent.

Community QA (CQA) is widely adopted by online businesses (McAuley and Yang, 2016) to help users. However, since it solely relies on humans to give answers, it often takes a long time to get a question answered or even not answered at all as we discussed in the introduction. Although there exist researches that align reviews to questions as an information retrieval task (McAuley and Yang, 2016; Yu and Lam, 2018), giving a whole review to the user to read is time-consuming and not suitable for customer service settings that require interactive responses.

Knowledge bases (KBs) (such as Freebase (Dong et al., 2015; Xu et al., 2016; Yao and Van Durme, 2014) or DBpedia (Lopez et al., 2010; Unger et al., 2012)) have been used for question answering (Yu and Lam, 2018). However, the ever-changing nature of online businesses, where new products and services appear constantly, makes it prohibitive to build a high-quality KB to cover all new products and services.

Reviews also serve as a rich resource for sentiment analysis (Pang et al., 2002; Hu and Liu, 2004; Liu, 2012, 2015). Although document-level (review) sentiment classification may be considered as a solved problem (given ratings are largely available), aspect-based sentiment analysis (ABSA) is still an open challenge, where alleviating the cost of human annotation is also a major issue. ABSA aims to turn unstructured reviews into structured fine-grained aspects (such as the “battery” of a laptop) and their associated opinions (e.g., “good battery” is *positive* about the aspect battery). Two important tasks in ABSA are aspect extraction (AE) and aspect sentiment classification (ASC) (Hu and Liu, 2004), where the former aims to extract aspects (e.g., “battery”) and the latter targets to identify the polarity for a given aspect (e.g., *positive* for *battery*). Recently, supervised deep learning models dominate both tasks (Wang et al., 2016, 2017; Xu et al., 2018a; Tang et al., 2016; He et al., 2018) and many of these models use handcrafted features, lexicons, and complicated neural network architectures to remedy the insufficient training examples from both tasks. Although these approaches may achieve better performances by manually injecting human knowledge into the model, human baby-sat models may not be intelligent enough[6] and automated representation learning from review corpora is always preferred (Xu et al., 2018a; He et al., 2018). We push forward this trend with the recent advance in pre-trained language models from deep learning (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018; Radford et al., 2018a,b). Although it is practical to train domain word embeddings from scratch on large-scale review corpora (Xu et al., 2018a), it is impractical to train language models from scratch with limited computational resources. As such, we show that it is practical to adapt language models pre-trained from formal texts to domain reviews.

3 BERT and Review-based Tasks

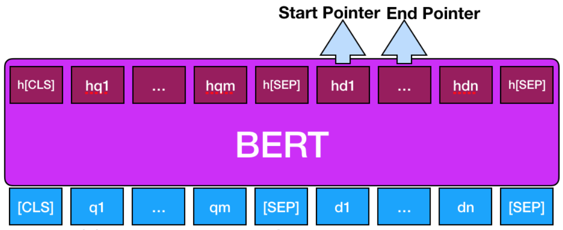
In this section, we briefly review BERT and derive its fine-tuning formulation on three (3) review-based end tasks.

3.1 BERT

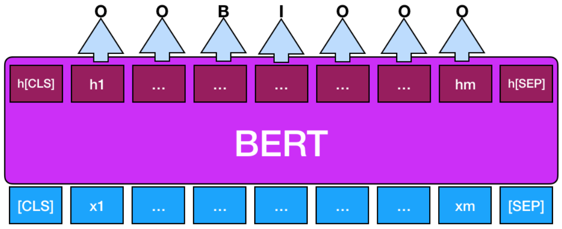
BERT is one of the key innovations in the recent progress of contextualized representation learning (Peters et al., 2018; Howard and Ruder, 2018; Radford et al., 2018a; Devlin et al., 2018). The idea behind the progress is that even though the word embedding (Mikolov et al., 2013; Pennington et al., 2014) layer (in a typical neural network for NLP) is trained from large-scale corpora, training a wide variety of neural architectures that encode contextual representations only from the limited supervised data on end tasks is insufficient. Unlike ELMo (Peters et al., 2018) and ULMFiT

---

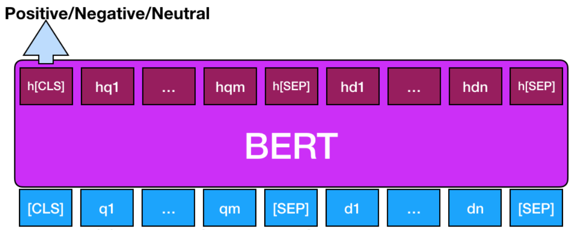
[6] `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`



(1) Review Reading Comprehension



(2) Aspect Extraction



(3) Aspect Sentiment Classification

Figure 1: Overview of BERT settings for review reading comprehension (RRC), aspect extraction (AE) and aspect sentiment classification (ASC).

(Howard and Ruder, 2018) that are intended to provide additional features for a particular architecture that bears human's understanding of the end task, BERT adopts a fine-tuning approach that requires almost no specific architecture for each end task. This is desired as an intelligent agent should minimize the use of prior human knowledge in the model design. Instead, it should learn such knowledge from data. BERT has two parameter intensive settings:

BERT$_{\text{BASE}}$: 12 layers, 768 hidden dimensions and 12 attention heads (in transformer) with the total number of parameters, 110M;

BERT$_{\text{LARGE}}$: 24 layers, 1024 hidden dimensions and 16 attention heads (in transformer) with the total number of parameters, 340M.

We only extend BERT with one extra task-specific layer and fine-tune BERT on each end task. We focus on three (3) review-based tasks: review reading comprehension (RRC), aspect extraction (AE) and aspect sentiment classification (ASC). The inputs/outputs settings are depicted in Figure 1 and detailed in the following subsections.

3.2 Review Reading Comprehension (RRC)

Following the success of SQuAD (Rajpurkar et al., 2016) and BERT's SQuAD implementation, we design review reading comprehension as follows. Given a question $q = (q_1, \dots, q_m)$ asking for an answer from a review $d = (d_1, \dots, d_n)$, we formulate the input as a sequence $x = (\texttt{[CLS]}, q_1, \dots, q_m, \texttt{[SEP]}, d_1, \dots, d_n, \texttt{[SEP]})$, where `[CLS]` is a dummy token not used for RRC and `[SEP]` is intended to separate q and d. Let $\text{BERT}(\cdot)$ be the pre-trained (or post-trained as in the next section) BERT model. We first obtain the hidden representation as $h = \text{BERT}(x) \in \mathbb{R}^{r_h * |x|}$, where $|x|$ is the length of the input sequence and r_h is the size of the hidden dimension. Then the hidden representation is passed to two separate dense layers followed by softmax functions: $l_1 = \text{softmax}(W_1 \cdot h + b_1)$ and $l_2 = \text{softmax}(W_2 \cdot h + b_2)$, where W_1, $W_2 \in \mathbb{R}^{r_h}$ and $b_1, b_2 \in \mathbb{R}$. The softmax is applied along the dimension of the sequence. The output is a span across the positions in d (after the `[SEP]` token of the input), indicated by two pointers (indexes) s and e computed from l_1 and l_2: $s = \arg\max_{\text{Idx}_{\texttt{[SEP]}} < s < |x|}(l_1)$ and $e = \arg\max_{s \le e < |x|}(l_2)$, where $\text{Idx}_{\texttt{[SEP]}}$ is the position of token `[SEP]` (so the pointers will never point to tokens from the question). As such, the final answer will always be a valid text span from the review as $a = (d_s, \dots, d_e)$.

Training the RRC model involves minimizing the loss that is designed as the averaged cross entropy on the two pointers:

$$\mathcal{L}_{\text{RRC}} = -\frac{\sum \log l_1 \mathbb{I}(s) + \sum \log l_2 \mathbb{I}(e)}{2},$$

where $\mathbb{I}(s)$ and $\mathbb{I}(e)$ are one-hot vectors representing the ground truths of pointers.

RRC may suffer from the prohibitive cost of annotating large-scale training data covering a wide range of domains. And BERT severely lacks two kinds of prior knowledge: (1) large-scale domain knowledge (e.g., about a specific product category), and (2) task-awareness knowledge (MRC/RRC in this case). We detail the technique of jointly incorporating these two types of knowledge in Sec. 4.

3.3 Aspect Extraction

As a core task in ABSA, aspect extraction (AE) aims to find aspects that reviewers have expressed opinions on (Hu and Liu, 2004). In supervised settings, it is typically modeled as a sequence labeling task, where each token from a sentence is labeled as one of $\{\underline{B}egin, \underline{I}nside, \underline{O}utside\}$. A continuous chunk of tokens that are labeled as one *B* and followed by zero or more *I*s forms an aspect. The input sentence with m words is constructed as $x = (\texttt{[CLS]}, x_1, \ldots, x_m, \texttt{[SEP]})$. After $h = \text{BERT}(x)$, we apply a dense layer and a softmax for each position of the sequence: $l_3 = \text{softmax}(W_3 \cdot h + b_3)$, where $W_3 \in \mathbb{R}^{3*r_h}$ and $b_3 \in \mathbb{R}^3$ (3 is the total number of labels (*BIO*)). Softmax is applied along the dimension of labels for each position and $l_3 \in [0, 1]^{3*|x|}$. The labels are predicted as taking argmax function at each position of l_3 and the loss function is the averaged cross entropy across all positions of a sequence.

AE is a task that requires intensive domain knowledge (e.g., knowing that "screen" is a part of a laptop). Previous study (Xu et al., 2018a) has shown that incorporating domain word embeddings greatly improve the performance. Adapting BERT's general language models to domain reviews is crucial for AE, as shown in Sec. 5.

3.4 Aspect Sentiment Classification

As a subsequent task of AE, aspect sentiment classification (ASC) aims to classify the sentiment polarity (positive, negative, or neutral) expressed on an aspect extracted from a review sentence. There are two inputs to ASC: an aspect and a review sentence mentioning that aspect. Consequently, ASC is close to RRC as the question is just about an aspect and the review is just a review sentence but ASC only needs to output a class of polarity instead of a textual span.

Let $x = (\texttt{[CLS]}, q_1, \ldots, q_m, \texttt{[SEP]}, d_1, \ldots, d_n, \texttt{[SEP]})$, where $q_1, \ldots, q_m$ now is an aspect (with m tokens) and $d_1, \ldots, d_n$ is a review sentence containing that aspect. After $h = \text{BERT}(x)$, we leverage the representations of `[CLS]` $h_{\text{[CLS]}}$, which is the aspect-aware representation of the whole input. The distribution of polarity is predicted as $l_4 = \text{softmax}(W_4 \cdot h_{\text{[CLS]}} + b_4)$, where $W_4 \in \mathbb{R}^{3*r_h}$ and $b_4 \in \mathbb{R}^3$ (3 is the number of polarities). Softmax is applied along the dimension of labels on `[CLS]`: $l_4 \in [0, 1]^3$. Training loss is the cross entropy on the polarities.

As a summary of these tasks, insufficient supervised training data significantly limits the performance gain across these 3 review-based tasks. Although BERT's pre-trained weights strongly boost the performance of many other NLP tasks on formal texts, we observe in Sec. 5 that BERT's weights only result in limited gain or worse performance compared with existing baselines. In the next section, we introduce the post-training step to boost the performance of all these 3 tasks.

4 Post-training

As discussed in the introduction, fine-tuning BERT directly on the end task that has limited tuning data faces both domain challenges and task-awareness challenge. To enhance the performance of RRC (and also AE and ASC), we may need to reduce the bias introduced by non-review knowledge (e.g., from Wikipedia corpora) and fuse domain knowledge (DK) (from unsupervised domain data) and task knowledge (from supervised MRC task but out-of-domain data). Given MRC is a general task with answers of questions covering almost all document contents, a large-scale MRC supervised corpus may also benefit AE and ASC. Eventually, we aim to have a general-purpose post-training strategy that can exploit the above two kinds of knowledge for end tasks.

To post-train on domain knowledge, we leverage the two novel pre-training objectives from BERT: masked language model (MLM) and next sentence[7] prediction (NSP). The former predicts randomly masked words and the latter detects whether two sides of the input are from the same document or not. A training example is formulated as $(\texttt{[CLS]}, x_{1:j}, \texttt{[SEP]}, x_{j+1:n}, \texttt{[SEP]})$, where $x_{1:n}$ is a document (with randomly masked words) split into two sides $x_{1:j}$ and $x_{j+1:n}$ and `[SEP]` separates those two.

MLM is crucial for injecting review domain knowledge and for alleviating the bias of the knowledge from Wikipedia. For example, in the Wikipedia domain, BERT may learn to guess the `[MASK]` in "The `[MASK]` is bright" as "sun". But in a laptop domain, it could be "screen". Further, if the `[MASK]`ed word is an opinion word in "The touch screen is `[MASK]`", this objective challenges BERT to learn the representations for fine-grained opinion words like "great" or "terrible" for `[MASK]`. The objective of NSP further encourages BERT to learn contextual representation beyond word-level. In the context of reviews,

---

[7] The BERT paper refers a sentence as a piece of text with one to many natural language sentences.

NSP formulates a task of "artificial review prediction", where a negative example is an original review but a positive example is a synthesized fake review by combining two different reviews. This task exploits the rich relationships between two sides in the input, such as whether two sides of texts have the same rating or not (when two reviews with different ratings are combined as a positive example), or whether two sides are targeting the same product or not (when two reviews from different products are merged as a positive example). In summary, these two objectives encourage to learn a myriad of fine-grained features for potential end tasks.

We let the loss function of MLM be $\mathcal{L}_{\text{MLM}}$ and the loss function of next text piece prediction be $\mathcal{L}_{\text{NSP}}$, the total loss of the domain knowledge post-training is $\mathcal{L}_{\text{DK}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}}$.

To post-train BERT on task-aware knowledge, we use SQuAD (1.1), which is a popular large-scale MRC dataset. Although BERT gains great success on SQuAD, this success is based on the huge amount of training examples of SQuAD (100,000+). This amount is large enough to ameliorate the flaws of BERT that has almost no questions on the left side and no textual span predictions based on both the question and the document on the right side. However, a small amount of fine-tuning examples is not sufficient to turn BERT to be more task-aware, as shown in Sec. 5. We let the loss on SQuAD be $\mathcal{L}_{\text{MRC}}$, which is in a similar setting as the loss $\mathcal{L}_{\text{RRC}}$ for RRC. As a result, the joint loss of post-training is defined as $\mathcal{L} = \mathcal{L}_{\text{DK}} + \mathcal{L}_{\text{MRC}}$.

One major issue of post-training on such a loss is the prohibitive cost of GPU memory usage. Instead of updating parameters over a batch, we divide a batch into multiple sub-batches and accumulate gradients on those sub-batches before parameter updates. This allows for a smaller sub-batch to be consumed in each iteration.

Algorithm 1: Post-training Algorithm

```
Input: $\mathcal{D}_{\text{DK}}$: one batch of DK data;
       $\mathcal{D}_{\text{MRC}}$ one batch of MRC data;
       $u$: number of sub-batches.
1 $\nabla_\Theta \mathcal{L} \leftarrow 0$
2 $\{\mathcal{D}_{\text{DK},1}, \ldots, \mathcal{D}_{\text{DK},u}\} \leftarrow \text{Split}(\mathcal{D}_{\text{DK}}, u)$
3 $\{\mathcal{D}_{\text{MRC},1}, \ldots, \mathcal{D}_{\text{MRC},u}\} \leftarrow \text{Split}(\mathcal{D}_{\text{MRC}}, u)$
4 for $i \in \{1, \ldots, u\}$ do
5     $\mathcal{L}_{\text{partial}} \leftarrow \frac{\mathcal{L}_{\text{DK}}(\mathcal{D}_{\text{DK},i}) + \mathcal{L}_{\text{MRC}}(\mathcal{D}_{\text{MRC},i})}{u}$
6     $\nabla_\Theta \mathcal{L} \leftarrow \nabla_\Theta \mathcal{L} + \text{BackProp}(\mathcal{L}_{\text{partial}})$
7 end
8 $\Theta \leftarrow \text{ParameterUpdates}(\nabla_\Theta \mathcal{L})$
```

Algorithm 1 describes one training step and takes one batch of data on domain knowledge (DK) $\mathcal{D}_{\text{DK}}$ and one batch of MRC training data $\mathcal{D}_{\text{MRC}}$ to update the parameters Θ of BERT. In line 1, it first initializes the gradients ∇_Θ of all parameters as 0 to prepare gradient computation. Then in lines 2 and 3, each batch of training data is split into u sub-batches. Lines 4-7 spread the calculation of gradients to u iterations, where the data from each iteration of sub-batches are supposed to be able to fit into GPU memory. In line 5, it computes the partial joint loss $\mathcal{L}_{\text{partial}}$ of two sub-batches $\mathcal{D}_{\text{DK},i}$ and $\mathcal{D}_{\text{MRC},i}$ from the i-th iteration through forward pass. Note that the summation of two sub-batches' losses is divided by u, which compensate the scale change introduced by gradient accumulation in line 6. Line 6 accumulates the gradients produced by backpropagation from the partial joint loss. To this end, accumulating the gradients u times is equivalent to computing the gradients on the whole batch once. But the sub-batches and their intermediate hidden representations during the i-th forward pass can be discarded to save memory space. Only the gradients ∇_Θ are kept throughout all iterations and used to update parameters (based on the chosen optimizer) in line 8. We detail the hyper-parameter settings of this algorithm in Sec. 5.3.

5 Experiments

We aim to answer the following research questions (RQs) in the experiment:

RQ1: what is the performance gain of post-training for each review-based task, with respect to the state-of-the-art performance?

RQ2: what is the performance of BERT's pre-trained weights on three review-based tasks without any domain and task adaptation?

RQ3: upon ablation studies of separate domain knowledge post-training and task-awareness post-training, what is their respective contribution to the whole post-training performance gain?

5.1 End Task Datasets

As there are no existing datasets for RRC and to be consistent with existing research on sentiment analysis, we adopt the *laptop* and *restaurant* reviews of SemEval 2016 Task 5 as the source to create datasets for RRC. We do not use SemEval 2014 Task 4 or SemEval 2015 Task 12 because these datasets do not come with the review(document)-level XML tags to recover whole reviews from review sentences. We keep the split of training and testing of the SemEval 2016 Task 5 datasets and annotate multiple QAs for each review following the way of constructing QAs for the SQuAD 1.1 datasets (Rajpurkar et al., 2016).

To make sure our questions are close to real-world questions, 2 annotators are first exposed to 400 QAs from CQA (under the laptop category in Amazon.com or popular restaurants in Yelp.com) to get familiar with real questions. Then they are asked to read reviews and independently label textual spans and ask corresponding questions when they feel the textual spans contain valuable information that customers may care about. The textual spans are labeled to be as concise as possible but still human-readable. Note that the annotations for sentiment analysis tasks are not exposed to annotators to avoid biased annotation on RRC. Since it is unlikely that the two annotators can label the same QAs (the same questions with the same answer spans), they further mutually check each other's annotations and disagreements are discussed until agreements are reached. Annotators are encouraged to label as many questions as possible from testing reviews to get more test examples. A training review is encouraged to have 2 questions (training examples) on average to have good coverage of reviews.

The annotated data is in the format of SQuAD 1.1 (Rajpurkar et al., 2016) to ensure compatibility with existing implementations of MRC models. The statistics of the RRC dataset (ReviewRC) are shown in Table 2. Since SemEval datasets do not come with a validation set, we further split 20% of reviews from the training set for validation.

Statistics of datasets for AE and ASC are given in Table 3. For AE, we choose SemEval 2014 Task 4 for laptop and SemEval-2016 Task 5 for restaurant to be consistent with (Xu et al., 2018a) and other previous works. For ASC, we use SemEval 2014 Task 4 for both laptop and restaurant as existing research frequently uses this version. We use 150 examples from the training set of all these datasets for validation.

5.2 Post-training datasets

For domain knowledge post-training, we use Amazon laptop reviews (He and McAuley, 2016) and Yelp Dataset Challenge reviews[8]. For laptop, we filtered out reviewed products that have appeared in the validation/test reviews to avoid training bias for test data (Yelp reviews do not have this issue as the source reviews of SemEval are not from Yelp). Since the number of reviews is small, we choose a duplicate factor of 5 (each review generates about 5 training examples) during BERT data pre-processing. This gives us 1,151,863 post-training examples for laptop domain knowledge.

For the restaurant domain, we use Yelp reviews from restaurant categories that the SemEval reviews also belong to (Xu et al., 2018a). We choose 700K reviews to ensure it is large enough to generate training examples (with a duplicate factor of 1) to cover all post-training steps that we can afford (discussed in Section 5.3)[9]. This gives us 2,677,025 post-training examples for restaurant domain knowledge learning.

For MRC task-awareness post-training, we leverage SQuAD 1.1 (Rajpurkar et al., 2016) that come with 87,599 training examples from 442 Wikipedia articles.

5.3 Hyper-parameters

We adopt **BERT$_{\text{BASE}}$** (uncased) as the basis for all experiments[10]. Since post-training may take a large footprint on GPU memory (as BERT pre-training), we leverage FP16 computation[11] to reduce the size of both the model and hidden representations of data. We set a static loss scale of 2 in FP16, which can avoid any over/under-flow of floating point computation. The maximum length of post-training is set to 320 with a batch size of 16 for each type of knowledge. The number of sub-batch u is set to 2, which is good enough to store each sub-batch iteration into a GPU memory of 11G. We use Adam optimizer and set the learning rate to be 3e-5. We train 70,000 steps for the laptop domain and 140,000 steps for the restaurant

---

[8] `https://www.yelp.com/dataset/challenge`

[9] We expect that using more reviews can have even better results but we limit the amount of reviews based on our computational power.

[10] We expect **BERT$_{\text{LARGE}}$** to have better performance but leave that to future work due to limited computational power.

[11] `https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html`

Dataset	Num. of Questions	Num. of Reviews
Laptop Training	1015	443
Laptop Testing	351	79
Restaurant Training	799	347
Restaurant Testing	431	90

Table 2: Statistics of the ReviewRC Dataset. Reviews with no questions are ignored.

	AE	ASC
Laptop	SemEval14 Task4	SemEval14 Task4
Training	3045 S./2358 A.	987 P./866 N./460 Ne.
Testing	800 S./654 A.	341 P./128 N./169 Ne.
Restaurant	SemEval16 Task5	SemEval14 Task4
Training	2000 S./1743 A.	2164 P./805 N./633 Ne.
Testing	676 S./622 A.	728 P./196 N./196 Ne.

Table 3: Summary of datasets on aspect extraction and aspect sentiment classification. S: number of sentences; A: number of aspects; P., N., and Ne.: number of positive, negative and neutral polarities.

domain, which roughly have one pass over the pre-processed data on the respective domain.

5.4 Compared Methods

As BERT outperforms existing open source MRC baselines by a large margin, we do not intend to exhaust existing implementations but focus on variants of BERT introduced in this paper.

DrQA is a baseline from the document reader[12] of DrQA (Chen et al., 2017). We adopt this baseline because of its simple implementation for reproducibility. We run the document reader with random initialization and train it directly on ReviewRC. We use all default hyper-parameter settings for this baseline except the number of epochs, which is set as 60 for better convergence.

DrQA+MRC is derived from the above baseline with official pre-trained weights on SQuAD. We fine-tune document reader with ReviewRC. We expand the vocabulary of the embedding layer from the pre-trained model on ReviewRC since reviews may have words that are rare in Wikipedia and keep other hyper-parameters as their defaults.

For AE and ASC, we summarize the scores of the state-of-the-arts on SemEval (based the best of our knowledge) for brevity.

DE-CNN (Xu et al., 2018a) reaches the state-of-the-arts for AE by leveraging domain embeddings.

MGAN (Li et al., 2018) reaches the state-of-the-art ASC on SemEval 2014 task 4.

Lastly, to answer RQ1, RQ2, and RQ3, we have the following BERT variants.

BERT leverages the vanilla BERT pre-trained weights and fine-tunes on all 3 end tasks. We use this baseline to answer RQ2 and show that BERT's pre-trained weights alone have limited performance gains on review-based tasks.

BERT-DK post-trains BERT's weights only on domain knowledge (reviews) and fine-tunes on the 3 end tasks. We use BERT-DK and the following BERT-MRC to answer RQ3.

BERT-MRC post-trains BERT's weights on SQuAD 1.1 and then fine-tunes on the 3 end tasks.

BERT-PT (proposed method) post-trains BERT's weights using the joint post-training algorithm in Section 4 and then fine-tunes on the 3 end tasks.

5.5 Evaluation Metrics and Model Selection

To be consistent with existing research on MRC, we use the same evaluation script from SQuAD 1.1 (Rajpurkar et al., 2016) for RRC, which reports Exact Match (EM) and F1 scores. EM requires the answers to have exact string match with human annotated answer spans. F1 score is the averaged F1 scores of individual answers, which is typically higher than EM and is the major metric. Each individual F1 score is the harmonic mean of individual precision and recall computed based on the number of overlapped words between the predicted answer and human annotated answers.

For AE, we use the standard evaluation scripts come with the SemEval datasets and report the F1 score. For ASC, we compute both accuracy and Macro-F1 over 3 classes of polarities, where Macro-F1 is the major metric as the imbalanced classes introduce biases on accuracy. To be consistent with existing research (Tang et al., 2016), examples belonging to the *conflict* polarity are dropped due to a very small number of examples.

We set the maximum number of epochs to 4 for BERT variants, though most runs converge just within 2 epochs. Results are reported as averages of **9** runs (9 different random seeds for random batch generation).[13]

5.6 Result Analysis

The results of RRC, AE and ASC are shown in Tables 4, 5 and 6, respectively. To answer RQ1, we observed that the proposed joint post-training (BERT-PT) has the best performance over all tasks in all domains, which show the benefits of having two types of knowledge.

---

[12] https://github.com/facebookresearch/DrQA

[13] We notice that adopting 5 runs used by existing researches still has a high variance for a fair comparison.

Domain	Laptop		Rest.	
Methods	**EM**	**F1**	**EM**	**F1**
DrQA(Chen et al., 2017)	38.26	50.99	49.52	63.73
DrQA+MRC(Chen et al., 2017)	40.43	58.16	52.39	67.77
BERT	39.54	54.72	44.39	58.76
BERT-DK	42.67	57.56	48.93	62.81
BERT-MRC	47.01	63.87	54.78	68.84
BERT-PT	48.05	**64.51**	59.22	**73.08**

Table 4: RRC in EM (Exact Match) and F1.

Domain	Laptop	Rest.
Methods	**F1**	**F1**
DE-CNN(Xu et al., 2018a)	81.59	74.37
BERT	79.28	74.1
BERT-DK	83.55	77.02
BERT-MRC	81.06	74.21
BERT-PT	**84.26**	**77.97**

Table 5: AE in F1.

To answer RQ2, to our surprise we found that the vanilla pre-trained weights of BERT do not work well for review-based tasks, although it achieves state-of-the-art results on many other NLP tasks (Devlin et al., 2018). This justifies the need to adapt BERT to review-based tasks.

To answer RQ3, we noticed that the roles of domain knowledge and task knowledge vary for different tasks and domains. For RRC, we found that the performance gain of BERT-PT mostly comes from task-awareness (MRC) post-training (as indicated by BERT-MRC). The domain knowledge helps more for restaurant than for laptop. We suspect the reason is that certain types of knowledge (such as specifications) of laptop are already present in Wikipedia, whereas Wikipedia has little knowledge about restaurant. We further investigated the examples improved by BERT-MRC and found that the boundaries of spans (especially short spans) were greatly improved.

For AE, we found that great performance boost comes mostly from domain knowledge post-training, which indicates that contextualized representations of domain knowledge are very important for AE. BERT-MRC has almost no improvement on restaurant, which indicates Wikipedia may have no knowledge about aspects of restaurant. We suspect that the improvements on laptop come from the fact that many answer spans in SQuAD are noun terms, which bear a closer relationship with laptop aspects.

For ASC, we observed that large-scale annotated MRC data is very useful. We suspect the reason is that ASC can be interpreted as a special MRC problem, where all questions are about the polarity of a given aspect. MRC training data may help BERT to understand the input format of ASC given their closer input formulation. Again, domain knowledge post-training also helps ASC.

Domain	Laptop		Rest.	
Methods	**Acc.**	**MF1**	**Acc.**	**MF1**
MGAN (Li et al., 2018)	76.21	71.42	81.49	71.48
BERT	75.29	71.91	81.54	71.94
BERT-DK	77.01	73.72	83.96	75.45
BERT-MRC	77.19	74.1	83.17	74.97
BERT-PT	78.07	**75.08**	84.95	**76.96**

Table 6: ASC in Accuracy and Macro-F1(MF1).

We further investigated the errors from BERT-PT over the 3 tasks. The errors on RRC mainly come from boundaries of spans that are not concise enough and incorrect location of spans that may have certain nearby words related to the question. We believe precisely understanding user's experience is challenging from only domain post-training given limited help from the RRC data and no help from the Wikipedia data. For AE, errors mostly come from annotation inconsistency and boundaries of aspects (e.g., apple OS is predicted as OS). Restaurant suffers from rare aspects like the names of dishes. ASC tends to have more errors as the decision boundary between the negative and neutral examples is unclear (e.g., even annotators may not sure whether the reviewer shows no opinion or slight negative opinion when mentioning an aspect). Also, BERT-PT has the problem of dealing with one sentence with two opposite opinions ("The screen is good but not for windows."). We believe that such training examples are rare.

6 Conclusions

We proposed a new task called *review reading comprehension* (RRC) and investigated the possibility of turning reviews as a valuable resource for answering user questions. We adopted BERT as our base model and proposed a joint post-training approach to enhancing both the domain and task knowledge. We further explored the use of this approach in two other review-based tasks: aspect extraction and aspect sentiment classification. Experimental results show that the post-training approach before fine-tuning is effective.

Acknowledgments

Bing Liu's work was partially supported by the National Science Foundation (NSF IIS 1838770) and by a research gift from Huawei.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multicolumn convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Exploiting document knowledge for aspect-level sentiment classification. *arXiv preprint arXiv:1806.04346*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *World Wide Web*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. *arXiv preprint arXiv:1608.03542*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, Xin Li, and Qiang Yang. 2018. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. *arXiv preprint arXiv:1811.10999*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Vanessa Lopez, Andriy Nikolov, Marta Sabou, Victoria Uren, Enrico Motta, and Mathieu d'Aquin. 2010. Scaling up question-answering to linked data. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 193–210. Springer.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language understanding paper.pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018b. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2018. Drcd: a chinese machine reading comprehension dataset. *arXiv preprint arXiv:1806.00920*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. *arXiv preprint arXiv:1603.06679*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018a. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Hu Xu, Sihong Xie, Lei Shu, and Philip S. Yu. 2018b. Dual attention network for product compatibility and function satisfiability analysis. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *arXiv preprint arXiv:1603.00957*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 956–966.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015. Neural generative question answering. *arXiv preprint arXiv:1512.01337*.

Qian Yu and Wai Lam. 2018. Aware answer prediction for product-related questions incorporating aspects. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 691–699. ACM.

Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text

Ahmad Sakor[1*], **Isaiah Onando Mulang'**[2], **Kuldeep Singh**[2],
Saeedeh Shekarpour[3], **Maria-Esther Vidal**[4], **Jens Lehmann**[2], and **Sören Auer**[4]

[1]L3S Research Center, Hannover, Germany
[2]Fraunhofer IAIS, Sankt Augustin, Germany
[3]University of Dayton, Dayton, USA
[4]TIB, Hannover, Germany
`{sakor}@l3s.de`,`{maria.vidal,auer}@tib.eu`
`{isaiah.mulang.onando,kuldeep.singh}@iais.fraunhofer.de`
`{sshekarpour1}@udayton.edu`
`{jens.lehmann}@iais.fraunhofer.de`

Abstract

Short texts challenge NLP tasks such as named entity recognition, disambiguation, linking and relation inference because they do not provide sufficient context or are partially malformed (e.g. wrt. capitalization, long tail entities, implicit relations). In this work, we present the Falcon approach which effectively maps entities and relations within a short text to its mentions of a background knowledge graph. Falcon overcomes the challenges of short text using a light-weight linguistic approach relying on a background knowledge graph. Falcon performs joint entity and relation linking of a short text by leveraging several fundamental principles of English morphology (e.g. compounding, headword identification) and utilizes an extended knowledge graph created by merging entities and relations from various knowledge sources. It uses the context of entities for finding relations and does not require training data. Our empirical study using several standard benchmarks and datasets show that Falcon significantly outperforms state-of-the-art entity and relation linking for short text query inventories.

1 Introduction

Entity Linking (EL) task annotates surface forms in the text with the corresponding reference mentions in knowledge bases such as Wikipedia. It involves the two sub-tasks, i.e. Named Entity Recognition and Disambiguation (NER and NED) tasks. The state of the art contains considerable research body for EL from text to its Wikipedia mention (Cucerzan, 2007; Ferragina and Scaiella, 2010; Hoffart et al., 2011; Balog, 2018; Shen et al., 2015; Ferragina and Scaiella, 2010; Hoffart et al., 2014). With the emergence of Knowledge Graphs (KGs) which represent data in a higher structured and semantic format such as DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and Wikidata (Vrandecic, 2012) that utilize Wikipedia as familiar knowledge source, retrieval-based applications such as question answering (QA) systems or keyword-based semantic search systems are empowered to provide more cognitive capabilities. Entity linking is a crucial component for a variety of applications built on knowledge graphs. For instance, an ideal NED tool on DBpedia recognizes the entities embedded in the question 'Who wrote the book The Pillars of The Earth?' and links them to the corresponding DBpedia entity (e.g. 'Pillars of The Earth' to `dbr:The_Pillars_of_the_Earth`)[1]. Another important NLP task is relation linking; it is about linking surface forms in text representing a relation to equivalent relations (predicates) of a KG. In our example question, an ideal relation linking (RL) tool links '`wrote`' to `dbo:author`[2]. There are existing approaches which address EL and RL tasks either jointly or independently (Miwa and Sasaki, 2014; Kirschnick et al., 2016; Wang et al., 2018; Dubey et al., 2018; Singh et al., 2017). However, they mostly fail in case of short text (e.g. question or key words based query) because the short text does not provide sufficient context which is essential for the disambiguation process. More importantly, a short text is often malformed meaning the text is incomplete,

---

[*] First three authors have equal contribution.

[1] dbr is the prefix for `http://dbpedia.org/resource/`

[2] dbo is the prefix for `http://dbpedia.org/ontology/`

Proceedings of NAACL-HLT 2019, pages 2336–2346
Minneapolis, Minnesota, June 2 - June 7, 2019.

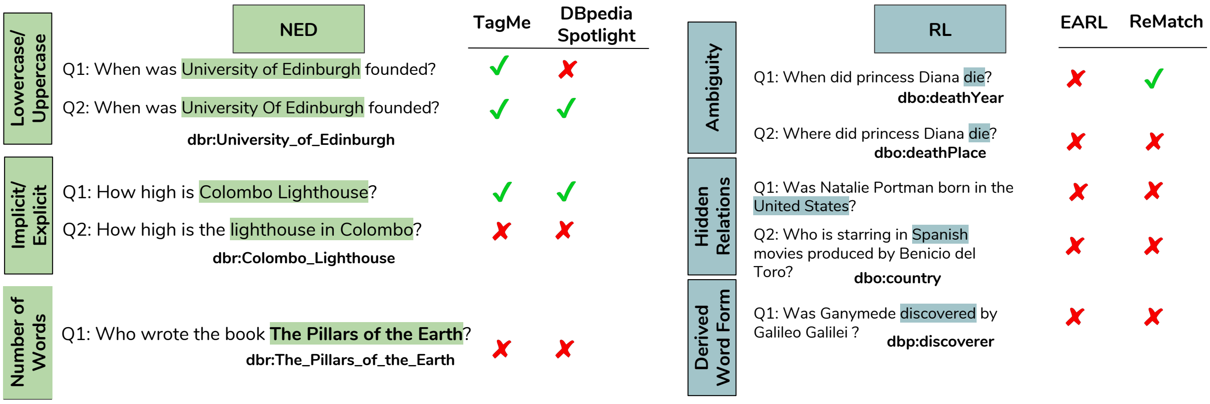


Figure 1: Performance of two EL and RL approaches on Specific Questions. TagMe and DBpedia Spotlight are the top-2 NED systems over the LC-QuAD QA dataset. However, considering short text questions, their behavior varies concerning question features, e.g., lowercase vs. uppercase, having implicit vs. explicit mappings, etc. Similar behavior has been observed for the top relation linking tools.

inexpressive, or implicit which is the case, particularly for relations in short sentences.

In this paper, we contribute to proposing a novel approach for jointly linking entities and relations within a short text into the entities and relations of DBpedia KG. This approach is robust to the challenges of short text, and moreover, it is efficient.

Research Objectives. Existing approaches and systems for NER, NED, EL, and RL resort to machine learning and deep learning approaches that require a large training data (Cao et al., 2018; Mudgal et al., 2018). These approaches achieve high performance on data similar to seen data. For instance, Singh et al. (2018c) evaluated 20 NED tools for question answering over the DBpedia KG including TagMe (Ferragina and Scaiella, 2012), DBpedia Spotlight (Mendes et al., 2011), Babelfy (Moro et al., 2014), and several APIs released by industry including Ambiverse (Ambiverse, 2018), TextRazor (TextRazor, 2018), and Dandelion (Dati, 2018). Among all, TagMe reports the highest F-score (0.67) over the complex question answering dataset LC-QuAD (TagMe is one of the top performing tools with an F-score of 0.91 on the generic WikiDisamb30 dataset (Ferragina and Scaiella, 2012)). Please be noted that TagMe was explicitly released for short text. However, when the input text is from a domain different from the training domain, its performance significantly falls down. Regarding the performance of various RL approaches such as ReMatch (Mulang' et al., 2017), SIBKB (Singh et al., 2017) is still low concerning accuracy and run-time even if they are purposefully developed for a particular domain or task. This deficiency is due to disregarding the context of the entities (Singh et al., 2018c,b). Therefore, when aiming for annotating entities and relations of short text, it is important to develop an approach which a) is agnostic of the requirement of large training data and b) jointly links entities and relations to its KG equivalence.

Approach. We target the problem of joint entity and relation linking within short text using the DBpedia KG as background knowledge. We propose a novel approach that resorts to several fundamental principles of English morphology such as compounding (Bauer and Laurie, 1983), right-hand rule for headword identification (Williams, 1981) and utilizes an extended knowledge graph created by merging entities and relations from various knowledge sources. The approach focuses on capturing semantics underlying the input text by using the context of entities for finding relations and does not require any training data. Albeit simple, to the best of our knowledge, the combination of strategies and optimization of our approach is unique. Our evaluations show that it leads to substantial gains in recall, precision, and F-score on various benchmarks and domains.

Resource. Falcon is available as an open Web API[3], and its source code is released to ensure reproducibility. Another open source contribution is an extended knowledge graph which we built by merging information from several sources, e.g. DBpedia, Wikidata, Oxford dictionary, and Wordnet. These contributions are in our public Github[4].

The paper is structured as follows: the next sec-

---

[3] `https://labs.tib.eu/falcon/`

[4] `https://github.com/AhmadSakor/falcon`

tion motivates our work by illustrating several limitations of state of the art over short text. Section 3 detailed our approach and we present evaluation results in Section 4. We describe related literature in Section 5 and Section 6 concludes our findings.

2 Motivating Example

We motivate our work by analyzing the performance of state-of-the-art EL and RL tools regarding query inventories on the DBpedia KG. In the following, we categorize the observed limitations. **Effect of Capitalization on EL tools** TagMe and DBpedia Spotlight are the best two performing EL systems for question answering over DBpedia (Singh et al., 2018c). Considering the question 'When was University of Edinburgh founded', where the entity `University of Edinburgh` has one word (i.e. 'of') starting with lowercase letters. TagMe can identify this entity and link to its corresponding DBpedia entity `dbr:University_of_Edinburgh` but DBpedia Spotlight fails. However, when all words in the entity label are in uppercase, both tools recognize and link entities correctly (cf. Figure 1).

Effect of Implicit/Explicit Entities on EL tools The vocabulary mismatch problem (Shekarpour et al., 2017) is common for text paraphrasing and significantly affects the performance of EL approaches. In Figure 1, both EL tools can correctly link the entity in the question 'How high is Colombo Lighthouse?' but fail when the question is rephrased to 'How high is the lighthouse in Colombo?' due to the vocabulary mismatch problem. In the first representation of the question, the entity label `Colombo Lighthouse` exactly matches to the DBpedia entity `dbr:Colombo_Lighthouse` which is not the case in the rephrased question (`dbr:Colombo_Lighthouse` is expected entity for `lighthouse in Colombo`).

Effect of the Number of Words in an Entity Label on EL tools Long tail entities were studied as a separate phenomenon such as in news (Esquivel et al., 2017). For question answering, an increasing number of words jeopardizes entity linking performance. In our motivating example, both EL tools can not link the entity present from the question 'Who wrote the book The Pillars of the Earth?' where the entity label ('The Pillars of the Earth') has five words (a question from LC-QuAD dataset (Trivedi et al., 2017)).

Effect of Ambiguity of Question on RL tools EARL (Dubey et al., 2018) and Rematch (Mulang' et al., 2017) are the two top performing relation linking tools for question answering over two different datasets QALD-5 (Unger et al., 2015) and LC-QuAD respectively. In Figure 1, for the question 'When did princess Diana die', Rematch correctly recognizes the relation `die` and links it to `dbo:deathYear`. However, when the question slightly changed to "Where did princess Diana die?" in which the expected relation is `dbo:deathPlace`, both tools fail to understand the ambiguity of the question intent and cannot provide the correct DBpedia IRIs.

Effect of Hidden Relation in a Question on RL tools Questions are typically relatively short and sometimes there is no natural language label for the relation. For example, to correctly answer the LC-QuAD question 'Was Natalie Portman born in the United States?' contains two relations: 1) the relational label `born` needs to be linked to `dbo:birthPlace` and 2) `dbo:country` is the hidden relation for which no relation surface form is present. A similar case can be observed in another question from the same dataset 'Who is starring in Spanish movies produced by Benicio del Toro?' where one of the expected relations is `dbo:country` for which no relation label is present. For both questions, EARL and ReMatch cannot identify hidden relations.

Effect of Derived Word Form of Relation Label on RL tools Consider the question 'Was Ganymede discovered by Galileo Galilei?' in which the relation label `discovered` is expected to link to the DBpedia ontology `dbo:discoverer`. The word `discoverer` is the derived word form of relation label `discovered`, and due to this, both tools fail to provide correct relation linking.

3 The Falcon Approach

The Falcon approach maps the surface forms within the short text into the textual representation of entities in KG. This mapping follows a particular strategy which is formalized in the following. Formally, a given short text is a set of tokens $\mathcal{T} = \{t_1, ..., t_n\}$. The set of entities in KG is the union of all KG resources $\mathcal{E} = C \cup P \cup I$ (where C, P, I are respectively a set of classes, properties, and instances), and L is the set of literals associated with entities. The task of entity linking is

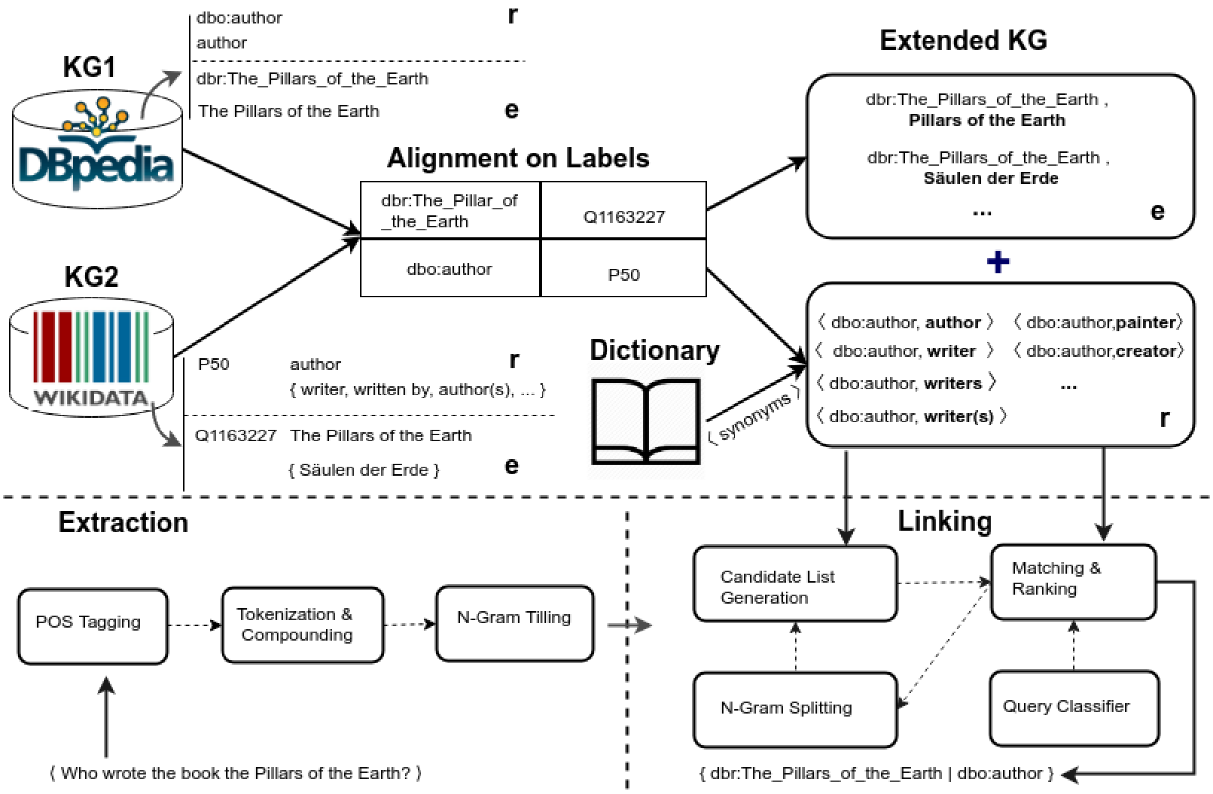


Figure 2: Overview of Falcon Approach. Falcon consists of two building blocks: 1) An extended knowledge graph which is built by merging information from various knowledge sources such as DBpedia, Wikidata, Oxford Dictionary, and WordNet. 2) Falcon architecture that has several modules focusing on surface form extraction and linking them to KG.

about mapping a subset of the input tokens denoted by $\mathcal{S} \in \mathcal{P}(\mathcal{T})$ (where $\mathcal{P}(\mathcal{T})$ is the power set of $\mathcal{T}$) to a set of entities denoted by $\mathcal{S}' \in \mathcal{P}(\mathcal{E})$ (where $\mathcal{P}(\mathcal{E})$ is the power set of $\mathcal{E}$), this mapping formally is represented as $\rho : \mathcal{S} \to \mathcal{S}'$. The Falcon approach deals with two optimization tasks as while it tries to maximize the number of tokens included in the set $\mathcal{S}$ (equation 1), it reduces the number of mapped entities in the set $\mathcal{S}'$ (eq. 2).

$$\gamma = \arg \max_{t_i \in \mathcal{A} | \mathcal{S} \in \mathcal{S}} \{\#t_i\} \tag{1}$$

$$\omega = \arg \min_{e_i \in \mathcal{A} | \mathcal{A} \in \mathcal{S}'} \{\#e_i\} \tag{2}$$

Extended Knowledge Graph The DBpedia KG contains over 5.6 million entities and 111 million facts (consisting of subject-predicate-object triples) which require overall 14.2GB storage (Auer et al., 2007). A major portion of this large information is not useful for EL/RL. Therefore we sliced DBpedia and extracted all the entity and relation labels to create a local KG. For example, the entity `Barack Obama`[5] in DBpedia has the natural language label 'Barack Obama' but DBpedia does not contain another representation of this label. However, the Wikidata KG is much richer and contains several aliases (or `known_as` labels) of Barack Obama such as Hussein Obama II, Barack Obama II , Obama, Barak Obama, President Obama, BHO and others[6]. We extended our local KG with this information from Wikidata. Similarly, for relation labels, the local KG is enriched with traditional linguistic resources such as Oxford dictionary (OED, 1989), and semantic dictionaries like WordNet (Miller, 1995a) to provide synonyms, derived word forms, etc. Use of background knowledge is common in question answering over DBpedia such as AskNow (Dubey et al., 2016) uses Wordnet to support relation linking. However, we also propose extending entity labels using Wikidata which is not yet used in literature. These two separate extended KGs with a total size of 1.4GB are used as an underlying

---

[5] `http://dbpedia.org/page/Barack_Obama`

[6] `https://www.wikidata.org/wiki/Q76`

source of knowledge and act as the core of our approach (cf. Figure 2).

POS Tagging In the first module illustrated in Figure 2, short input text annotated with POS tag information using spaCy (Honnibal and Johnson, 2015). This step is used primarily to identify verb and noun phrases in the sentence.

Tokenization and Compounding The next module creates tokens from the input sentence removing the stop words. In the first step, we break the sentence into potential tokens by removing all the stop words, and we use the stopword list provided by Fox (1990). For creating tokens, we also reuse basic compounding principle of English morphology. Compound words are lexeme that contains two or more stems (Bauer and Laurie, 1983). The words which do not have any stop words between them considered as one compound word during token formation. For example, in question "Who is the wife of Barack Obama?", Barack Obama is noun phrases which do not have any stop word between, they considered as a single compound word. Compounding allows us to reduce the total number of tokens.

N-gram Tiling Typically, approaches described in (Shekarpour et al., 2017, 2013) dealing with short text start with the shortest token (or N-gram) to search associated candidates in the knowledge graph. This approach is not effective when an entity has many words in its label as it creates several additional tokens. For example in question "Who wrote the book The Pillars of the Earth?", It may generate several little tokens such as book, Pillars, Earth and it will result in several potential candidates in KG. In contrast, Bill et al. (2002) applied an N-gram tiling algorithm in a question answering system to find the long answer in case of overlapping small answers. For example, answers "PQR" and "QRS" merged into single long answer "PQRS." This algorithm proceeds greedily until high scoring longest tilled N-gram found. We applied a similar approach to find the longest possible token for extracting the potential entity label. In the exemplary question " Who wrote the book The Pillars of the Earth?", The previous module generates tokens "wrote, book, Pillars, Earth." In N-gram tiling algorithm, we do not consider identified verbs of the sentence because in most cases a verb cannot be an entity label. Hence three tokens "book, Pillars, Earth" are merged as a single token. Also, verb token acts as a division point of the sentence in case of two entities, and we do not merge tokens from either side of the verb. In this process, the N-gram tiling algorithm starts with the first token from either side of the verb (which is a case of two entities in a sentence) and ends at the last non-stop word. The tiling algorithm also considers the stop words and provide the longest tilled N-gram. After N-gram tiling, we have two tokens: "wrote" and "book The Pillars of the Earth."

Candidate List Generation From the tokens, we create two list 1) `potential relation candidates` which contain verbs ("wrote") 2) `potential entity candidates` ("book The Pillars of the Earth"). We first search tokens of potential relation candidates in an extended KG of relations and get all the possible `DBpedia relation candidates.` Similar process has been repeated separately for `potential entity candidates` and all the `DBpedia entity candidates` are generated. For search, we use elastic search (elasticsearch, 2015) over indexed extended KG. The reason behind the use of elastic search is its effectiveness over indexed KGs as reported by Dubey et al. (2018). In few cases, it is also possible that there is no verb in a sentence (e.g. Who is the prime minister of USA?). Then, we keep the list `potential relation candidates` empty, and search all the tokens of `potential entity candidates` into extended KG of DBpedia relations because number of relations in DBpedia are very less and when tokens in `potential entity candidates` find any match, they are pushed to `potential relation candidates.`

Candidate Ranking To rank best DBpedia candidates, we utilize the fundamental principle of knowledge graph creation. In any knowledge graph, a sentence is represented as triple with <subject, predicate, object>. Therefore, we rank the candidates by creating a triple consisting of the relation and entity candidates from `DBpedia entity candidates` and `DBpedia relation candidates,` then check if these triples exist in the DBpedia KG. We do it by passing the triple to DBpedia SPARQL endpoint. This can be done by executing a simple Ask query against a KG endpoint which would return a boolean value indicative of the existence of triple or otherwise of this triple. For each existing triple, we increase the weight of the entities and relations involved in the triple.

While ranking, we also consider question headwords (who, what, when, etc.) for question classification (Huang et al., 2009). Each relation in DBpedia has its domain and range associated with an entity such as person, place, date, etc. The headwords are used to determine the correct range and domain of the DBpedia relation. For example in the question "Who is starring in Spanish movies produced by Benicio del Toro?" there is a hidden relation `dbo:country` for which no surface form is present. While checking the domain of each token in relation and entity candidate lists, we can extract that word "Spanish" has the domain country; therefore, it is also an expected relation.

N-Gram Splitting In the previous module, if we do not get any triple in DBpedia for candidates present in `potential entity candidates` and `potential relation candidates`, we split the tokens (N-grams). To split the tokens, we again use the fundamentals of English morphology. The compound words in English have their headword always towards right side (Williams, 1981). Therefore, we start splitting tokens from "N-Gram tiling" module from the right side and pass these tokens to candidate generation module. This greedy algorithm stops when it finds triple(s) of DBpedia candidate list.

4 Experimental Study

Experiment Setup. We used a local laptop machine, with eight cores and 16GB RAM running Ubuntu 18.04 for implementation. Falcon is deployed as public API on a server with 723GB RAM, 96 cores (Intel(R) Xeon(R) Platinum 8160 CPU with 2.10GHz) running Ubuntu 18.04. This API is used for calculating all the results. The EL systems have been evaluated on different settings in literature, therefore to provide a fair evaluation we utilize Gerbil (Usbeck et al., 2015), which is a benchmarking framework for EL systems and integrated Falcon API into the Gerbil architecture. We report macro precision (P), macro recall (R), and macro F-score[7] in the tables. Falcon average run time is 1.9 seconds per question. Gerbil does not benchmark RL systems; therefore, RL systems are benchmarked using Frankenstein platform (Singh et al., 2018a). Our code, Extended KG, and data is in Github.[8]

Datasets. We employ two distinct datasets: 1) the LC-QuAD (Trivedi et al., 2017) dataset comprises 5,000 complex questions for DBpedia (80 percent questions are with more than one entity and relation) where average question length is 12.29 words. 2) QALD-7 (Usbeck et al., 2017) is the most popular benchmarking dataset for QA over DBpedia comprising 215 questions. In QALD, the average question length is 7.41 words and over 50% of the questions include a single entity and relation. For our linguistic based approach, we randomly selected 100 questions each from SimpleQuestions dataset (Bordes et al., 2015) and complex questions[9] for the formation of rules.

System	**Dataset**	**P**	**R**	**F**
KEA (Waitelonis and Sack, 2016)	QALD-7	0.06	0.06	0.06
EARL (Dubey et al., 2018)	QALD-7	0.58	0.60	0.58
FOX (Speck and Ngomo, 2014)	QALD-7	0.59	0.57	0.57
Babelfy (Moro et al., 2014)	QALD-7	0.40	0.55	0.44
AIDA (Hoffart et al., 2011)	QALD-7	0.61	0.58	0.59
DBpedia Spotlight (Mendes et al., 2011)	QALD-7	0.68	0.72	0.69
TagMe (Ferragina and Scaiella, 2012)	QALD-7	0.64	0.76	0.67
Falcon	QALD-7	**0.78**	**0.79**	**0.78**
KEA (Waitelonis and Sack, 2016)	LC-QuAD	0.001	0.001	0.001
EARL (Dubey et al., 2018)	LC-QuAD	0.53	0.55	0.53
FOX (Speck and Ngomo, 2014)	LC-QuAD	0.53	0.51	0.51
Babelfy (Moro et al., 2014)	LC-QuAD	0.43	0.50	0.44
AIDA (Hoffart et al., 2011)	LC-QuAD	0.50	0.45	0.47
DBpedia Spotlight (Mendes et al., 2011)	LC-QuAD	0.60	0.65	0.61
TagMe (Ferragina and Scaiella, 2012)	LC-QuAD	0.65	0.77	0.68
Falcon	LC-QuAD	**0.81**	**0.86**	**0.83**
(Singh et al., 2018c)	LC-QuAD3253	0.69	0.66	0.67
Falcon	LC-QuAD3253	**0.73**	**0.74**	**0.73**

Table 1: Performance of the Falcon Framework compared to various entity linking tools.

Baselines. The state-of-the-art outperforming tools are TagMe and DBpedia Spotlight reported in (2018c). These two systems in addition to the systems already integrated in Gerbil i.e., KEA (Waitelonis and Sack, 2016), FOX (Speck and Ngomo, 2014), Babelfy (Moro et al., 2014), AIDA (Hoffart et al., 2011) are included in our benchmark. We also report the performance of EARL (Dubey et al., 2018) for entity linking as it jointly performs EL and RL. For relation linking, the recently released EARL system is our baseline. We evaluate NED and RL systems on the LC-QuAD3253 subset of the LC-QuAD dataset (containing 3,253 LC-QuAD questions) to compare the

---

[7] `https://github.com/dice-group/gerbil/wiki/Precision,-Recall-and-F1-measure`

[8] `https://github.com/AhmadSakor/falcon`

[9] `http://qa.mpi-inf.mpg.de/comqa/`

performance with the 20 NED and five RL systems evaluated by Singh et al. (2018c). Many of these 20 tools are APIs from industry (Ambiverse (Ambiverse, 2018), TextRazor (TextRazor, 2018), and Dandelion (Dati, 2018)) which use state of the art machine learning approaches.

QA Component	Dataset	P	R	F
SIBKB (Singh et al., 2017)	QALD-7	0.29	0.31	0.30
ReMatch (Mulang' et al., 2017)	QALD-7	0.31	0.34	0.33
EARL (Dubey et al., 2018)	QALD-7	0.27	0.28	0.27
Falcon	QALD-7	**0.58**	**0.61**	**0.59**
SIBKB (Singh et al., 2017)	LC-QuAD	0.13	0.15	0.14
ReMatch (Mulang' et al., 2017)	LC-QuAD	0.15	0.17	0.16
EARL (Dubey et al., 2018)	LC-QuAD	0.17	0.21	0.18
Falcon	LC-QuAD	**0.42**	**0.44**	**0.43**
(Singh et al., 2018c)	LC-QuAD3253	0.25	0.22	0.23
Falcon	LC-QuAD3253	**0.56**	**0.57**	**0.56**

Table 2: Performance of the Falcon Framework compared to various Relation Linking tools.

Performance Evaluation Table 1 summarizes Falcon's performance compared to state-of-the-art systems integrated in Gerbil. For the QALD and LC-QuAD datasets, Falcon significantly outperforms the baseline. Similar observations are made for relation linking, where the performance of Falcon is approximately twice as high as the next best competitor on all datasets (cf. Table 2).

Success cases of Falcon: Falcon overcomes several major issues of short text such as capitalization of surface forms, derived word forms of relation labels and successfully handles long tail entities. For entity linking, we achieve slightly better performance on LC-QuAD than QALD. This is due to the fact that LC-QuAD questions mostly contain more than one entity and relation and thus provide more context to understand the short text. Also, major failure cases of state-of-the-art EL systems over these datasets are due to the short length and limitation to exploit the context. For example the question 'Give me the count of all people who ascended a peak in California.' (`dbr:California` is correct entity), TagMe provides two entities: `dbr:California` (for surface form California) and `dbr:Give_In_to_Me` (for "Give me"). Fundamental principles such as compounding and N-gram tiling have positive impact on the Falcon performance and we can correctly annotate several long tail entities and entities containing compound words. For example, Falcon correctly annotates question from LC-QuAD: 'Name the military unit whose garrison is Arlington County, Virginia and command structure is United States Department of Defense' where expected entities are `dbr:Arlington_County,_Virginia` and `dbr:United_States_Department_of_Defense`. Also, extended local KG has provided several interpretation of entities and their derived forms. The extended KG act as source of background knowledge during the linking process and provide extra information about entities. Generally, other entity linking tools directly map surface forms to the underlying KG using several novel techniques. However, this concept of enriching a local extended KG is not exploited in the literature and it has positively impacted the performance of the Falcon.

For relation linking, taking the context of the entities into account improved the overall performance of the Falcon. In our example question 'Who wrote the book The Pillars of the Earth?', EARL, SIBKB and Rematch aim for directly mapping `wrote` to DBpedia which results in several wrong relations such as `dbo:writer`, `dbo:creator` but when Falcon considers entity references of the question to verify which triples exist with the given entity `dbr:The_Pillars_of_the_Earth`, Falcon determines the correct relation `dbo:author`. It is important to note that existing relation linking tools completely ignore the context of the entities. Secondly, Falcon uses a fundamental principle of creating an RDF knowledge graph. While ranking the candidates in the Candidate List Ranking step, Falcon verifies the presence of the correct triple containing entity and associated relation in the KG. It has been done by cross-checking all the combinations of `potential entity candidates` and `potential relation candidates` as triple using an ASK query. Three concepts (utilization of entity context, ranking the candidates based on the presence of triple in the KG, and use of extended KG) have collectively resulted into a significant jump over other relation linking tools as observed in the Table 2.

Failure cases of Falcon: There are few EL cases where Falcon fails. For example, in question 'How many writers worked on the album Main Course?', the expected entity is `dbr:Main_Course`. However, Falcon returns `dbr:Critters_2:_The_Main_Course`. This is caused by compounding and the resulting token for this question was 'album Main Course'. For the same question Falcon correctly links the relations. We further analyzed failure cases of Falcon for RL. We found that more than half of the questions which were unanswered have implicit relations. For example, for the question 'In what city is the Heineken brewery?' with the two relations `dbo:locationCity` and `dbo:manufacturer`, Falcon returns `dbo:city` as relation. There are few types of questions ('Count all the scientologists.') for which Falcon fails both for EL and RL tasks. This question is relatively short and requires reasoning to provide correct entities and relations (`dbr:Scientology` and `dbo:religion`).

5 Related Work

A wide range of tools and research work exist in the area of NER and NED (please see (Balog, 2018; Shen et al., 2015) for a detailed survey). Mostly, research in this domain targets news corpus, documents and Wikipedia abstract having long sentences. Such systems have been trained and benchmarked for NER/NED performance over several related datasets such as ACE2004, IITB, AIDA/CoNLL, Wiki-Disamb30, Spotlight Corpus, etc (Usbeck et al., 2015). It is important to note that most of these approaches use state of the art machine learning techniques and require a large amount of training data. However, when these tools applied to short text in a new domain such as question answering (QA) or key word based search, the performance is limited. (Singh et al., 2018c; Derczynski et al., 2015). Considering short text, the tool TagMe (Ferragina and Scaiella, 2010) is one of the popular works in this area, and uses a dictionary of entity surface forms extracted from Wikipedia to detect entity mentions in the parsed input text. These mentions passed through a voting scheme that computes the score for each mention-entity pair as the sum of votes given by candidate entities of all other mentions in the text (Ferragina and Scaiella, 2010), finally a pruning step filters out less relevant annotations. However, TagMe considered sentence length 30 for referring it as short text; in contrast for Falcon we target relatively more shorter text such as questions where average length is much less than 30 words (e.g., average question length in LC-QuAD dataset is 12.29 (Trivedi et al., 2017)). Following the popularity of KGs, scholars have shifted focus to use KGs such as DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and Wikidata (Vrandecic, 2012) for the NED task. DBpedia Spotlight (Mendes et al., 2011) is one such tool that performs NED on DBpedia. After an initial step of entity spotting, DBpedia Spotlight uses contextual information to resolve the surface forms of an entity to corresponding DBpedia resources. DBpedia Spotlight has also been reused in question answering systems (Dubey et al., 2016). Relation extraction from a sentence have been long-standing research field (Zelenko et al., 2003; Bunescu and Mooney, 2005; Banko and Etzioni, 2008; Zhu et al., 2009; Fundel et al., 2007). However, linking relation label to its KG mention as independent approach is a relatively new field of research. Mulang' et al. (Mulang' et al., 2017) had the first attempt in this direction and developed Rematch. ReMatch characterizes both the properties in a KG and the relations in a question as comparable triples, then leverages both synonyms and semantic similarity measures based on graph distances from the lexical knowledge base - Wordnet (Miller, 1995b). SIBKB (Singh et al., 2017) approach for relation linking uses PATTY to derive word embeddings for a bipartite semantically indexed knowledge base which assist in RL, likewise also in full QA systems such as AskNow (Dubey et al., 2016) where PATTY is deployed as an underlying source of relation patterns. Since NER/D and RE/L are parallel tasks and the occurrence of a named entity is often accompanied by relations, recent research has attempted to perform NED and RL as a joined process. EARL (Dubey et al., 2018) is a tool for joined NED and RL that relies on Generalized Travelling Salesman Problem to find the right path between entities in the question. Several techniques exist in the literature for the collective entity and relation extraction in a text (Miwa and Sasaki, 2014; Kirschnick et al., 2016; Wang et al., 2018) but we are not aware of any other approach besides EARL that perform joint entity and relation linking to a KG.

6 Conclusion

In this article we presented Falcon, an approach for linking Named Entities (EL) and Relations (RL) in short text to corresponding Knowledge Graph entities. The Falcon approach adopts two novel concepts. First we demonstrated how a fused KG comprising several complimentary semantic and linguistic resources can be employed as background knowledge. Secondly, we devised a linguistic understanding based method for processing the text, that leverages the extended background KG for EL/RL. Our comprehensive empirical evaluations provide evidence that the approach outperforms the state-of-the-art on several benchmarks. Although, we evaluate our approach on DBpedia, there is no specific assumption in our work on the structure or schema of the underlying knowledge graph, and our method should be equally applicable and can be extended to any other knowledge graph. Additionally, Falcon is offered as an online tool as well as an API.

Our approach provides considerable benefits over machine learning based approaches for short text. While Falcon achieves better results, it does not require training data and is easily adaptable to new domains. This work has highlighted the importance of background knowledge available in fused KGs as well as the linguistic understanding of the text. The linguistic methods (e.g. compounding) employed in Falcon can made more robust by using dependency parsing information. In future, we plan to explore the option of augmenting Falcon with deep learning methods for further improvement in performance specially in entity and relation extraction module.

Acknowledgments

This work has received funding from the EU H2020 Project No. 727658 (IASIS) and partially funded from Fraunhofer IAIS KDDS project No. 500747.

References

Ambiverse. 2018. *Ambiverse GmbH.*

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735. Springer.

Krisztian Balog. 2018. *Entity Linking*, pages 147–188. Springer International Publishing, Cham.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio. Association for Computational Linguistics.

Laurie Bauer and Bauer Laurie. 1983. *English word-formation.* Cambridge university press.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *CoRR*, abs/1506.02075.

Eric Brill, Susan T. Dumais, and Michele Banko. 2002. An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002.*

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 675–686.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 708–716.

Spazio Dati. 2018. *Dandelion Ltd.*

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Inf. Process. Manage.*, 51(2):32–49.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. EARL: joint entity and relation linking for question answering over knowledge graphs. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, pages 108–126.

Mohnish Dubey, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, and Jens Lehmann. 2016. AskNow: A Framework for Natural Language Query Formalization in SPARQL. In *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, pages 300–316. Springer.

elasticsearch. 2015. elasticsearch/elasticsearch.

José Esquivel, Dyaa Albakour, Miguel Martinez-Alvarez, David Corney, and Samir Moussa. 2017. On the long-tail entities in news. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*, pages 691–697.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1625–1628. ACM.

Paolo Ferragina and Ugo Scaiella. 2012. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software*, 29(1):70–75.

Christopher J. Fox. 1990. A stop list for general text. *SIGIR Forum*, 24(1-2):19–35.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. 2014. Discovering emerging entities with ambiguous names. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 385–396. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792. ACL.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1373–1378.

Zhiheng Huang, Marcus Thint, and Asli Çelikyilmaz. 2009. Investigation of question classifier in question answering. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 543–550.

Johannes Kirschnick, Holmer Hemsen, and Volker Markl. 2016. JEDI: joint entity and relation detection using type inference. In *Proceedings of ACL-2016 System Demonstrations, Berlin, Germany, August 7-12, 2016*, pages 61–66.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, pages 1–8. ACM.

G. Miller. 1995a. WORDNET: A lexical database for English. *Communications of the ACM*, 38(11).

George A. Miller. 1995b. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1858–1869.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL*, 2:231–244.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, An-Hai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34.

Isaiah Onando Mulang', Kuldeep Singh, and Fabrizio Orlandi. 2017. Matching natural language relations to knowledge graph properties for question answering. In *Proceedings of the 13th International Conference on Semantic Systems, SEMANTICS 2017, Amsterdam, The Netherlands, September 11-14, 2017*, pages 89–96.

OED. 1989. *Oxford English Dictionary*, second edition. Oxford University Press.

Saeedeh Shekarpour, Edgard Marx, Sören Auer, and Amit P. Sheth. 2017. RQUERY: Rewriting Natural Language Queries on Knowledge Graphs to Alleviate the Vocabulary Mismatch Problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3936–3943.

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2013. Question answering on interlinked data. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 1145–1156.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460.

Kuldeep Singh, Andreas Both, Arun Sethupat Radhakrishna, and Saeedeh Shekarpour. 2018a. Frankenstein: A Platform Enabling Reuse of Question Answering Components. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 624–638. Springer.

Kuldeep Singh, Ioanna Lytra, Arun Sethupat Radhakrishna, Saeedeh Shekarpour, Maria-Esther Vidal, and Jens Lehmann. 2018b. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph. *CoRR*, abs/1809.10044.

Kuldeep Singh, Isaiah Onando Mulang', Ioanna Lytra, Mohamad Yaser Jaradeh, Ahmad Sakor, Maria-Esther Vidal, Christoph Lange, and Sören Auer. 2017. Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 31:1–31:8.

Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria-Esther Vidal, Jens Lehmann, and Sören Auer. 2018c. Why Reinvent the Wheel: Let's Build Question Answering Systems Together. In *Proceedings of the 2018 World Wide Web Conference, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1247–1256. ACM.

René Speck and Axel-Cyrille Ngonga Ngomo. 2014. Ensemble learning for named entity recognition. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pages 519–534.

TextRazor. 2018. *TextRazor Ltd.*

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, pages 210–218. Springer.

Christina Unger, Corina Forascu, Vanessa López, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. 2015. Question Answering over Linked Data (QALD-5). In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.* CEUR-WS.org.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (QALD-7). In *Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, pages 59–69.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. 2015. GERBIL: general entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1133–1143.

Denny Vrandecic. 2012. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1063–1064. ACM.

Jörg Waitelonis and Harald Sack. 2016. Named entity linking in #tweets with KEA. In *Proceedings of the 6th Workshop on 'Making Sense of Microposts' co-located with the 25th International World Wide Web Conference (WWW 2016), Montréal, Canada, April 11, 2016.*, pages 61–63.

Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4461–4467.

Edwin Williams. 1981. On the notions" lexically related" and" head of a word". *Linguistic inquiry*, 12(2):245–274.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: A statistical approach to extracting entity relationships. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 101–110. ACM.

Be Consistent! Improving Procedural Text Comprehension using Label Consistency

Xinya Du[1*] **Bhavana Dalvi Mishra**[2] **Niket Tandon**[2] **Antoine Bosselut**[2]
Wen-tau Yih[2] **Peter Clark**[2] **Claire Cardie**[1]
[1]Department of Computer Science, Cornell University, Ithaca, NY
`{xdu, cardie}@cs.cornell.edu`
[2]Allen Institute for Artificial Intelligence, Seattle, WA
`{bhavanad, nikett, antoineb, scottyih, peterc}@allenai.org`

Abstract

Our goal is procedural text comprehension, namely tracking how the properties of entities (e.g., their location) change with time given a procedural text (e.g., a paragraph about photosynthesis, a recipe). This task is challenging as the world is changing throughout the text, and despite recent advances, current systems still struggle with this task. Our approach is to leverage the fact that, for many procedural texts, multiple independent descriptions are readily available, and that predictions from them should be consistent (label consistency). We present a new learning framework that leverages label consistency during training, allowing consistency bias to be built into the model. Evaluation on a standard benchmark dataset for procedural text, ProPara (Dalvi et al., 2018), shows that our approach significantly improves prediction performance (F1) over prior state-of-the-art systems.

1 Introduction

We address the task of procedural text comprehension, namely tracking how the properties of entities (e.g., their location) change with time throughout the procedure (e.g., photosynthesis, a cooking recipe). This ability is an important part of text understanding, allowing the reader to infer unstated facts such as how ingredients change during a recipe, what the inputs and outputs of a scientific process are, or who met whom in a news article about a political meeting. Although several procedural text comprehension systems have emerged recently (e.g., EntNet (Henaff et al., 2017), NPN (Bosselut et al., 2018), and ProStruct (Tandon et al., 2018)), they still make numerous prediction errors. A major challenge is that fully annotated training data for this task is expensive to collect, because many state changes by multiple entities may occur in a single text, requiring complex annotation.

(1) ...**oxygen** is given off...
(2) ...the plant produces **oxygen**...
(3) ...is used to create sugar and **oxygen**...

Figure 1: Fragments from three independent texts about photosynthesis. Although (1) is ambiguous as to whether oxygen is being created or merely moved, evidence from (2) and (3) suggests it is being created, helping to correctly interpret (1). More generally, encouraging consistency between predictions from different paragraphs about the same process/procedure can improve performance.

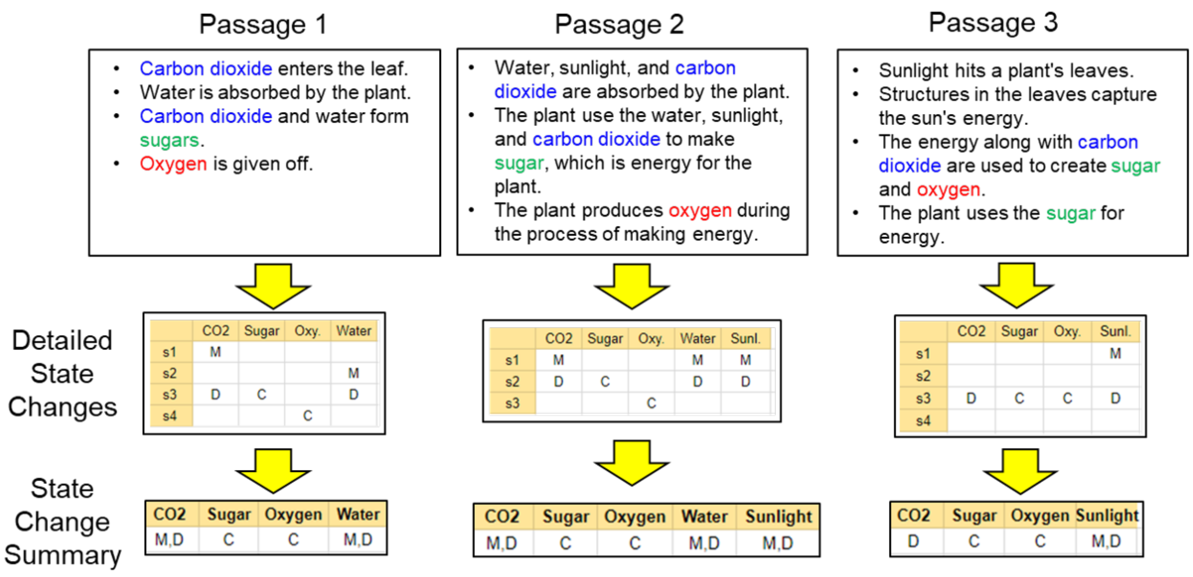
To address this challenge, and thus improve performance, our goals are two-fold: first, to better leverage the training data for procedural text comprehension that *is* available, and second, to utilize additional unlabeled data for the task (semi-supervised learning). Our approach in each case is to exploit *label consistency*, the property that two distinct texts covering the same procedure should be generally consistent in terms of the state changes that they describe, which constitute the labels to be predicted for the text. For example, in different texts describing photosynthesis, we expect them to be generally consistent about what happens to oxygen (e.g., that it is created), even if the wordings differ (Figure 1).

Using multiple, distinct passages to understand a process or procedure is challenging. Although the texts describe the same process, they might express the underlying facts at different levels of granularity, using different wordings, and including or omitting different details. As a result, the details may differ between paragraphs, making them hard to align and to check for consistency. Nonetheless, even if the details differ, we conjecture that the top-level *summaries* of each paragraph, which

---

[*] Work done while at the Allen Institute for Artificial Intelligence.

Proceedings of NAACL-HLT 2019, pages 2347–2356
Minneapolis, Minnesota, June 2 - June 7, 2019.



Passage 1

- Carbon dioxide enters the leaf.
- Water is absorbed by the plant.
- Carbon dioxide and water form sugars.
- Oxygen is given off.

Passage 2

- Water, sunlight, and carbon dioxide are absorbed by the plant.
- The plant use the water, sunlight, and carbon dioxide to make sugar, which is energy for the plant.
- The plant produces oxygen during the process of making energy.

Passage 3

- Sunlight hits a plant's leaves.
- Structures in the leaves capture the sun's energy.
- The energy along with carbon dioxide are used to create sugar and oxygen.
- The plant uses the sugar for energy.

	CO2	Sugar	Oxy.	Water
s1	M			
s2				M
s3	D	C		D
s4			C	

	CO2	Sugar	Oxy.	Water	Sunl.
s1	M			M	M
s2	D	C		D	D
s3			C		

	CO2	Sugar	Oxy.	Sunl.
s1				M
s2				
s3	D	C	C	D
s4				

CO2	Sugar	Oxygen	Water
M,D	C	C	M,D

CO2	Sugar	Oxygen	Water	Sunlight
M,D	C	C	M,D	M,D

CO2	Sugar	Oxygen	Sunlight
D	C	C	M,D

Figure 2: Three (simplified) passages from ProPara describing photosynthesis, the (gold) state changes each entity undergoes at each step $s_1, s_2, \ldots, s_T$, and the summary of state changes that each entity undergoes (an aggregation of the step-by-step changes), where `M = MOVED, D = DESTROYED, C = CREATED`. Although the language and detailed changes for each passage differ considerably, the **overall summaries are largely consistent** (e.g., sugar is `CREATED` in all three). We exploit this consistency when training a model to make these predictions, by biasing the model to prefer predictions whose summary is consistent with the (predicted) summaries of other passages about the same topic. Note that in the summary, we do not care about the order in which state changes happen, so summary `M, D` for participant CO_2 in passage 1 denotes a set of state changes rather than a sequence of state changes.

describe the types of state change that each entity undergoes, will be mostly consistent. For example, although independent texts describing photosynthesis vary tremendously, we expect them to be consistent about what generally happens to sugar, e.g., that it is created (Figure 2).

In this paper, we introduce a new training framework, called LaCE (Label Consistency Explorer), that leverages label consistency among paragraph summaries. In particular, it encourages label consistency during end-to-end training of a neural model, allowing consistency bias to improve the model itself, rather than be enforced in a post-processing step, e.g., posterior regularization (Ganchev et al., 2010). We evaluate on a standard benchmark for procedural text comprehension, called ProPara (Dalvi et al., 2018). We show that this approach achieves a new state-of-the-art performance in the fully supervised setting (when all paragraphs are annotated), and also demonstrate that it improves performance in the semi-supervised setting (using additional, unlabeled paragraphs) with limited training data. In the latter case, summary predictions from labeled data act as noisy gold labels for the unlabeled data, allowing additional learning to occur. Our contributions are thus:

1. A new learning framework, LaCE, applied to procedural text comprehension that improves the label consistency among different paragraphs on the same topic.
2. Experimental results demonstrating that LaCE achieves state-of-the-art performance on a standard benchmark dataset, ProPara, for procedural text.

2 Related Work

Our work is related to several important branches of work in both NLP and ML, as we now summarize.

Leveraging Label Consistency Leveraging information about label consistency (i.e., similar instances should have consistent labels at a certain granularity) is an effective idea. It has been studied in computer vision (Haeusser et al., 2017; Chen et al., 2018) and IR (Clarke et al., 2001; Dumais et al., 2002). Learning by association (Haeusser et al., 2017) establishes implicit cross-modal links between similar descriptions and leverage more unlabeled data during training. Schütze et al. (2018);

Hangya et al. (2018) adapt the similar idea to exploit unlabeled data for the cross-lingual classification. We extend this line of research in two ways: by developing a framework allowing it to be applied to the task of structure prediction; and by incorporating label consistency into the model itself via end-to-end training, rather than enforcing consistency as a post-processing step.

Semi-supervised Learning Approaches Besides utilizing the label consistency knowledge, our learning framework is also able to use unlabeled paragraphs, which fits in the literature of semi-supervised learning approaches (for NLP). Zhou et al. (2003) propose an iterative label propagation algorithm similar to spectral clustering. Zhu et al. (2003) propose a semi-supervised learning framework via harmonic energy minimization for data graph. Talukdar et al. (2008) propose a graph-based semi-supervised label propagation algorithm for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text sources. Our framework extends these ideas by introducing the notion of groups (examples that are expected to be similar) and summaries (what similarities are expected), applied in an end-to-end-framework.

Procedural Text Understanding and Reading Comprehension There has been a growing interest in procedural text understanding/QA recently. The ProcessBank dataset (Berant et al., 2014) asks questions about event ordering and event arguments for biology processes. bAbI (Weston et al., 2015) includes questions about movement of entities, however it's synthetically generated and with a small lexicon. Kiddon et al. (2015)'s RECIPES dataset introduces the task of predicting the locations of cooking ingredients, and Kiddon et al. (2016) for recipe generation. In this paper, we continue this line of exploration using ProPara, and illustrate how the previous two lines of work (label consistency and semi-supervised learning) can be integrated.

3 Problem Definition

3.1 Input and Output

A general condition for applying our method is having multiple examples where, for *some* properties, we expect to see similar values. For example, for procedural text, we expect paragraphs about the same process to be similar in terms of which entities move, are created, and destroyed; for different news stories about a political meeting, we expect top-level features (e.g., where the meeting took place, who attended) to be similar; for different recipes for the same item, we expect loosely similar ingredients and steps; and for different images of the same person, we expect some high-level characteristics (e.g., height, face shape) to be similar. Note that this condition does not apply to every learning situation; it only applies when training examples can be grouped, where all group members are expected to share some characteristics that we can identify (besides the label used to form the groups in the first place).

More formally, for training, the input is a set of labeled examples (x_{gi}, y_{gi}) (where y_{gi} are the labels for x_{gi}), partitioned into G groups, where the g subscript denotes which group each example belongs to. Groups are defined such that examples of the same group g are expected to have similar labels for a subset of labels y_{gi}. We call this subset the **summary labels**. We assume that both the groupings and the identity of the summary labels are provided. The output of training is a model M for labeling new examples. For testing, the input is the model M and a set of unlabeled (and ungrouped) examples x_t, and the output are their predicted labels $\hat{y}_t$. Note that this formulation is agnostic to the learning algorithm used. Later, we will consider both the fully supervised setting (all training examples are labeled) and semi-supervised setting (only a subset are labeled).

3.2 Instantiation

We instantiate this framework for procedural text comprehension, using the ProPara task (Dalvi et al., 2018). In this task, x_{gi} are paragraphs of text describing a process (e.g., photosynthesis), the labels y_{gi} describe the state changes that each entity in the paragraph undergoes at each step (sentence) (e.g., that oxygen is created in step 2), and the groups are paragraphs about the same topic (ProPara tags each paragraph with a topic, e.g., there are three paragraphs in ProPara describing photosynthesis). More precisely, each x_{gi} consists of:

- the name (topic) of a process, e.g., photosynthesis
- a sequence (paragraph) of sentences $S = [s_1, ..., s_T]$ that describes that process
- the set of entities E mentioned in that text, e.g., oxygen, sugar

and the targets (labels) to predict are:

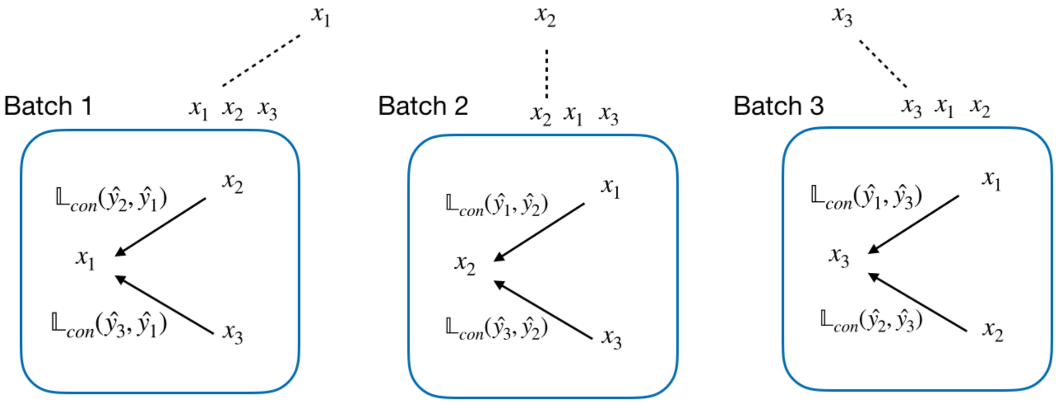


Figure 3: Example of batches constructed from a group (here, the group contains three labeled examples x_1, x_2, x_3). From three examples, three batches are constructed. Taking the predicted labels for the first element in the batch as reference we compute the consistency loss for the remaining elements.

- the state changes that each entity in E undergoes at each step (sentence) of the process, where a state change is one of {`Moved`,`Created`,`Destroyed`,`None`}. These state changes can be conveniently expressed using a $|S| \times |E|$ matrix (Figure 2). State changes also include arguments, e.g., the source and destination of a move. We omit these in this paper to simplify the description.

Finally, we define the summary labels as the set of state changes that each entity undergoes at *some* point in the process, without concern for when. For example, in Passage 1 in Figure 2, CO_2 is `Moved` (M) and `Destroyed` (D), while sugar is `Created` (C). These summary labels can be computed from the state-change matrix by aggregating the state changes for each entity over all steps. Our assumption here is that these summaries will generally be the same (i.e., consistent) for different paragraphs about the same topic. LaCE then exploits this assumption by encouraging this inter-paragraph consistency during training, as we now describe.

4 Label Consistency Explorer: LaCE

4.1 The LaCE Learning Framework

While a traditional supervised learning model operates on individual examples, LaCE operates on *batches* of grouped examples X_g. Given a group g containing N labeled examples $\{x_1, ..., x_N\}$ (we drop the g subscript for clarity), LaCE creates N batches, each containing all the examples but with a different x_i labeled as "primary", along with the gold labels y_i for (only) the primary example. (We informally refer to the primary example as the "first example" in each batch). Then for each batch, LaCE jointly optimizes the usual supervised loss $\mathbb{L}_{sup}(\hat{y}_i, y_i)$ for the primary example, along with a consistency loss between (summary) predictions for all other members of the group and the primary example, $\mathbb{L}_{con}(\hat{y}_j, \hat{y}_i)$ for all $j \neq i$. This is illustrated in Figures 4 and 3. This is repeated for all batches.

For example, for the three paragraphs about photosynthesis (Figure 2), batch 1 compares the first paragraph's predictions with its gold labels, and also compares the summary predictions of paragraphs 2 and 3 with those of the first paragraph (Figure 3). This is then repeated using paragraph 2, then paragraph 3 as primary.

The result is that LaCE jointly optimizes the supervised loss $\mathbb{L}_{sup}$ and consistency loss $\mathbb{L}_{con}$ to train a model that is both accurate for the given task as well as consistent in its predictions across examples that belong to the same group.

This process is approximately equivalent to jointly optimizing the usual supervised loss $\mathbb{L}_{sup}(\hat{y}_i, y_i)$ for all examples in the group, and the pairwise consistency loss $\mathbb{L}_{con}(\hat{y}_j, \hat{y}_i)$ for all pairs (x_j, x_i), $j \neq i$ in the group. However, there is an important difference, namely the relative contributions of $\mathbb{L}_{sup}$ and $\mathbb{L}_{con}$ is varied among batches, depending on how accurate the predictions for the primary example are (i.e., how small $\mathbb{L}_{sup}$ is), as we describe later in Section 4.3. This has the effect of paying more attention to consistency loss when predictions on the primary are more accurate.

We also extend LaCE to the semi-supervised setting as follows. For the semi-supervised setting, where only m of n ($m < n$) examples are labeled, we only form m batches, where each batch has

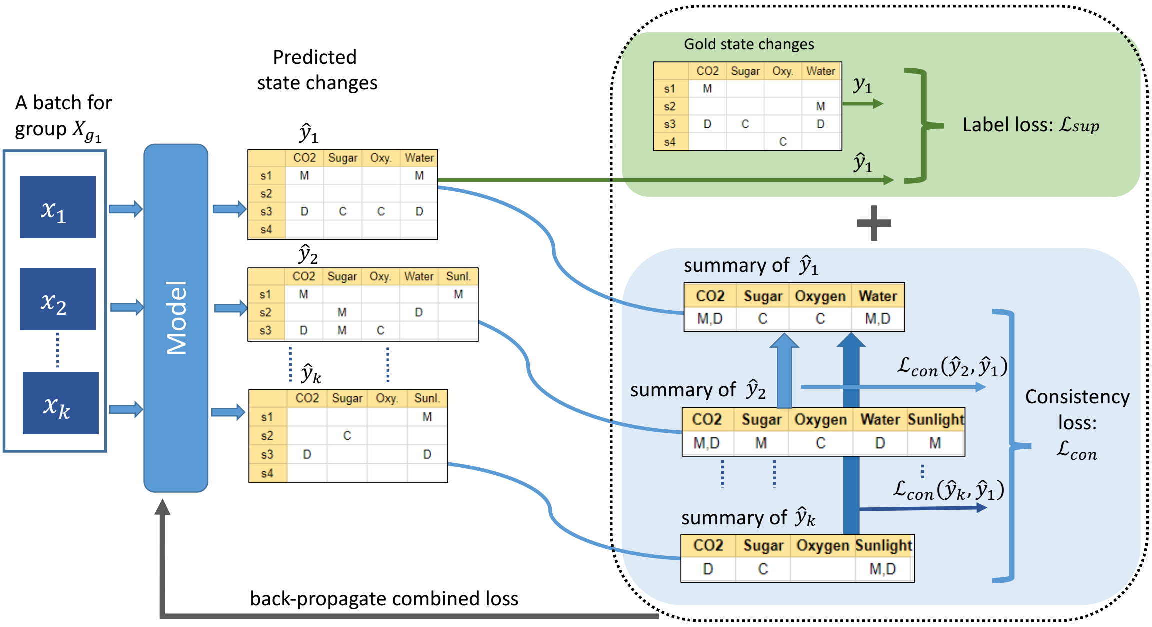


Figure 4: Overview of the LaCE training framework, illustrated for the procedural comprehension task ProPara. During training, LaCE processes *batches* of examples $\{x_1,...,x_k\}$ for each group X_g, where predictions for one example (here $\hat{y}_1$) are compared against its gold (producing loss $\mathbb{L}_{sup}$), and its summary against summaries of all other examples to encourage consistency of predictions (producing $\mathbb{L}_{con}$), repeating for each example in the batch.

a different labeled example as primary. We later report experiments results for both the fully and semi-supervised settings.

4.2 Base Model for Procedural Text

We now describe how LaCE is applied to our goal of comprehending procedural text. Note that LaCE is agnostic to the learner used within the framework. For this application, we use a simplified version of ProStruct (Tandon et al., 2018), a publicly available system designed for the ProPara task. Our implementation simplifies ProStruct by reusing its encoder, but then predicting (a distribution over) each state change label independently during decoding for every cell in the $|S| \times |E|$ grid (Figure 2). We briefly summarize this here.

4.2.1 Encoder

ProStruct uses an encoder-decoder architecture that takes procedural text as input and predicts the state changes of entities E in the text as output. During encoding, each step s_t is encoded using $|E|$ embeddings, one for each entity $e_j \in E$. Each embedding represents the action that s_t describes, applied to e_k. The model thus allows the same action to have different effects on different entities (e.g., a transformation destroys one entity, and creates another).

For each $(s_t, e_j) \in S \times E$ pair, the step is fed into a BiLSTM (Hochreiter and Schmidhuber, 1997), using pretrained GloVe (Pennington et al., 2014) vectors v_w for each word w_i concatenated with two indicator variables, one indicating whether w_i is a word referring to e_j, and one indicating whether w_i is a verb. A bilinear attention layer then computes attention over the contextualized vectors h_i output by the BiLSTM: $a_i = h_i * B * h_{ev} + b$, where B and b are learned parameters, and h_{ev} is the concatenation of h_e (the averaged contextualized embedding for the entity words w_e) and h_v (the averaged contextualized embedding for the verb words w_v).

Finally, the output vector c_{tj} is the attention-weighted sum of the h_i: $c_{tj} = \sum_{i=1}^{I} a_i * h_i$. Here, c_{tj} can be thought of as representing the action s_t applied to entity e_j. This is repeated for all steps and entities.

4.2.2 Decoder

To decode the action vectors c_{tj} into their resulting state changes they imply, each is passed through a feedforward layer to generate $logit(\pi_{tj})$, a set of logistic activations over the K possible state changes π_{tj} for entity e_j in step s_t. For ProPara, there are $K = 4$ possible state changes: `Move`, `Create`, `Destroy`, and `None`. These logits form a distribution over possible state changes to predict, for each entity and step in the text. We then compute loss, described next, using these distributions directly rather than discretizing them into exact predictions at this stage, so as not to lose information.

4.3 Applying LaCE

4.3.1 Batching

We start by creating training batches for each $X_{g_i} \in X_g$. From a group X_{g_i} comprising of n examples, we create n training batches. A batch consists of all n examples $(x_1, x_2, ..., x_n)$, but the loss computation is different in each batch. Figure 3 illustrates this.

4.3.2 Loss Computation

The loss computation in a batch is based on the usual supervised loss and additionally the consistency loss, as follows:

$$\mathbb{L}_{batch} = \lambda \underbrace{\mathbb{L}_{sup}(\hat{y}_1, y_1)}_{supervised\ loss} + (1-\lambda) \underbrace{\sum_{i=2}^{n} \mathbb{L}_{con}(\hat{y}_i, \hat{y}_1)}_{consistency\ loss} \tag{1}$$

Here, $\mathbb{L}_{sup}(\hat{y}_1, y_1)$ is the negative log likelihood loss[*] against the gold labels y_1, and λ is a hyperparameter tuned on the dev set.

To compute the consistency loss $\mathbb{L}_{con}(\hat{y}_i, \hat{y}_1)$, we compare the summaries computed from $\hat{y}_i$ and $\hat{y}_1$. In our particular application, a summary lists all the state changes each entity undergoes, formed by aggregating its step-by-step state changes. For example, for paragraph x_1 in Figure 4, as `CO2` first moves (`M`), then later is destroyed (`D`), we summarize its state changes as $s(\mathtt{CO_2}, \hat{y}_1) = \{\mathtt{M,D}\}$. In practice, as our decoder outputs distributions over the four possible values $\{\mathtt{M,C,D,N}\}$ rather than a single value, we summarize by adding and normalizing these distributions, producing a summary distribution $s(e, \hat{y}_j)$ over the four values rather than a discrete set of values.

To compute the consistency loss $\mathbb{L}_{con}(\hat{y}_i, \hat{y}_1)$ itself, we compare summaries for each entity e that occurs in both paragraph x_1 and paragraph x_i (referred to as Ent(x_1) and Ent(x_i) respectively), and compute the average mean squared error (MSE) between their summary distributions. We also tried other alternatives (e.g., Kullback-Leibler divergence) for calculating the distance between summary distributions, but mean squared error performs best. Equation 2 shows the details for computing the consistency loss.

$$\mathbb{L}_{con}(\hat{y}_i, \hat{y}_1) = \frac{\sum_{e \in Ent(x_i) \cap Ent(x_1)} \mathrm{MSE}(s(e, \hat{y}_i), s(e, \hat{y}_1))}{|Ent(x_i) \cap Ent(x_1)|} \tag{2}$$

Note that each paragraph contains varying number of entities and sentences. It is possible that some paragraphs do not mention exactly the same entities as the labeled paragraph (first element in the batch). In such cases, we penalize the model only for predictions for co-occurring entities. Unmatched entities are not penalized.

4.3.3 Adaptive Loss

The supervised loss $\mathbb{L}_{sup}(\hat{y}_1, y_1)$ is large in the early epochs when the model is not sufficiently trained. At this point, it is beneficial for the model to pay no attention to the consistency loss $\mathbb{L}_{con}(\hat{y}_j, \hat{y}_1)$ as the predicted action distributions are inaccurate. To implement this, if $\mathbb{L}_{sup}$ is above a defined threshold then the consistency loss term in Equation 1 is ignored (i.e. $\lambda = 1$). Otherwise, Equation 1 is used as is. This can loosely be seen as a form of simulated annealing (Kirkpatrick et al., 1988), using just two temperatures. Note that the time (epoch number) when the temperature (lambda) changes will vary across batches depending on the supervised loss within that batch of data, hence we call it an "adaptive" loss.

5 Experimental Results

We now present results on ProPara, the procedural text comprehension dataset introduced in (Dalvi et al., 2018). There are 187 topics in this dataset and a total of 488 labeled paragraphs (around 3 labeled paragraphs per topic). The task is to track how entities change state through the paragraph (as described in Section 3.2) and answer 4 classes of questions about those changes (7043/913/1095 questions in each of the train/dev/test partitions respectively). We compare LaCE with the baselines and prior state-of-the-art model ProStruct (Tandon et al., 2018) in two settings: (1) Fully supervised learning (using all the training data). (2) Semi-supervised learning (using some or all of the training data, plus additional unlabeled data).

5.1 Fully Supervised Learning

We evaluated LaCE by comparing its performance against published, state-of-the-art results on ProPara, using the full training set to train LaCE.

---

[*] Loss function $\mathbb{L}_{sup}$ is exactly same as the loss function used in the base model so that we can measure the effect of adding consistency loss.

Models	P	R	F1
EntNet (Henaff et al., 2017)	54.7	30.7	39.4
QRN (Seo et al., 2017)	60.9	31.1	41.1
ProLocal (Dalvi et al., 2018)	**81.7**	36.8	50.7
ProGlobal (Dalvi et al., 2018)	61.7	44.8	51.9
ProStruct (Tandon et al., 2018)	74.3	43.0	54.5
LaCE (our model)	75.3	**45.4**	**56.6**

Table 1: Comparing the performance of LaCE with prior methods on the test partition of ProPara.

Models	P	R	F1
LaCE	**75.3**	**45.4**	**56.6**
- consistency loss $\mathbb{L}_{con}$	69.6	43.1	53.2

Table 2: LaCE ablation results

The results are shown in Table 1. In Table 1, all the baseline numbers are the results reported in (Tandon et al., 2018). Note that all these baselines are trying to reduce the gap between predicted labels and gold labels on the training dataset. LaCE, however, also optimizes for consistency across labels for *groups* of paragraphs belonging to the same topic. As LaCE uses parts of ProStruct as its learning algorithm, the gains over ProStruct appear to be coming directly from its novel learning framework described in Section 4.1. To confirm this, we also performed an ablation study, removing the consistency loss term and just using the base model in LaCE. The results are shown in Table 2, and show that the F1 score drops from 56.6 to 53.2, illustrating that the consistency loss is responsible for the improvement. In addition, Table 2 indicates that consistency loss helps improve both precision and recall.

Also note that LaCE simplifies parts of ProStruct. For example, unlike ProStruct, LaCE does not use a pre-computed knowledge base during decoding. Thus LaCE is more efficient to train than ProStruct (>15x faster at training time).

5.2 Semi-Supervised Learning

Unlike the other systems in Table 1, LaCE is able to use unlabeled data during training. As described in Section 4.1, given a group containing both labeled and unlabeled paragraphs, we create as many batches as the number of labeled paragraphs in the group. Hence, paragraphs x_i with gold labels y_i can contribute to both supervised loss $\mathbb{L}_{sup}$ and consistency loss $\mathbb{L}_{con}$. Additionally, we can use unlabeled paragraphs x_j (i.e., without gold labels y_j), while computing consistency loss $\mathbb{L}_{con}$. This way LaCE can make use of unlabeled data during training.

Models	Proportion of labeled paragraphs used per training topic		
	33%	66%	100%
ProStruct	45.4	50.6	54.5
LaCE	47.3	51.2	56.6
LaCE + unlabeled data	49.9	52.9	56.7

Table 3: Comparing LaCE vs. ProStruct with varying amount of labeled paragraphs available per training topic. We compare their performance in terms of F1 on ProPara test partition.

To evaluate this, we collected 877 additional unlabeled paragraphs for ProPara topics[†]. As the original ProPara dataset makes some simplifying assumptions, in particular that events are mentioned in chronological order, we used Mechanical Turk to collect additional paragraphs that conformed to those assumptions (rather than collecting paragraphs from Wikipedia, say). Approximately 3 extra paragraphs were collected for each topic in ProPara. Note that collecting unlabeled paragraphs is substantially less expensive than labeling paragraphs.

	Train	Dev	Test	Unlabeled
# paragraphs	391	54	43	877

Table 4: ProPara Paragraphs Statistics

We then trained the ProStruct and LaCE models varying two different parameters: (1) the percentage of the labeled (ProPara) training data used to train the system (2) for LaCE only, whether the additional unlabeled data was also used. This allows us to see performance under different conditions of sparsity of labeled data, and (for LaCE) also assess how much unlabeled data can help under those conditions. During training, the unused labeled data was ignored (not used as unlabeled data). We keep the dev and test partitions the same as original dataset, picking a model based on dev performance and report results on test partition. The results are shown in Table 3. In the first two rows, ProStruct and LaCE are both trained with x% of labeled data, while the last row reports per-

---

[†] The unlabeled paragraphs are available at `http://data.allenai.org/propara/`.

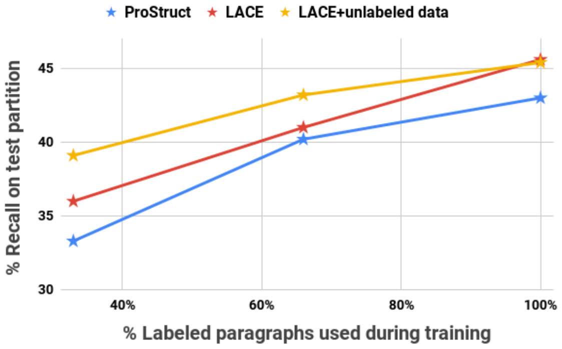


Figure 5: Comparing LaCE vs. ProStruct based on Recall on the test partition, by varying amount of labeled paragraphs available per training topic

formance of LaCE when it also has access to new unlabeled paragraphs.

Table 3 demonstrates that LaCE results in even larger improvements over ProStruct when the amount of labeled data is limited. In addition, unlabeled data adds an additional boost to this performance, in particular when labeled data is sparse. Further examination suggests that the gains in F1 are resulting mainly from improved recall, as shown in Figure 5. We believe that having access to unlabeled paragraphs and optimizing consistency across paragraphs for training topics, helps LaCE generalize better to unseen topics.

5.3 Implementation Details for LaCE

We implement our proposed model LaCE in PyTorch (Paszke et al., 2017) using the AllenNLP (Gardner et al., 2018) toolkit. We added a new data iterator that creates multiple batches per topic (Figure 3) which enables easy computation of consistency loss. We use 100D Glove embeddings (Pennington et al., 2014), trained on Wikipedia 2014 and Gigaword 5 corpora (6B tokens, 400K vocab, uncased). Starting from glove embeddings appended by entity and verb indicators, we use bidirectional LSTM layer to create contextual representation for every word in a sentence. We use 100D hidden representations for the bidirectional LSTM (Hochreiter and Schmidhuber, 1997) shared between all inputs (each direction uses 50D hidden vectors). We use attention layer on top of BiLSTM, using a bilinear similarity function similar to (Chen et al., 2016) to compute attention weights over the contextual embedding for each word in the sentence.

To compute the likelihood of all state changes individually, we use a single layer feedforward network with input dimension of 100 and output 4. In these experiments, we check if the supervised loss $\mathbb{L}_{sup}$ is less than a threshold (0.2 in our case) then we use equation 1 and $lambda = 0.05$. All hyper-parameters are tuned on the dev data.

	Consistency Score (%)	
	Train	Test
ProStruct	46.70	37.21
LaCE	54.39	38.36

Table 5: Consistency score comparison

During training we use multiple paragraphs for a topic to optimize for both supervised and consistency loss. At test time, LaCE's predictions are based on only one given paragraph. All the performance gains are due to the base model being more robust due to proposed training procedure. The code for LaCE model is published at `https://github.com/allenai/propara`.

5.4 Analysis and Discussion

We first discuss the predicted label consistency across paragraphs for LaCE vs. ProStruct. We then identify some of the limitations of LaCE.

Label Consistency

LaCE attempts to encourage consistency between paragraphs about the same topic during training, and yield similar benefit at test time. To examine whether this happens in practice, we compute and report the consistency score between paragraphs about the same topic (Table 5). Specifically, for an entity that appears in two paragraphs about the same topic, we compare whether the summaries of state change predictions for each match. The results are shown in Table 5.

The table shows that LaCE achieves greater prediction consistency during training, and that this benefit plays out to some extent at test time even though label consistency is not enforced at test time (we do not assume that examples are grouped at test time, hence consistency between groups cannot be enforced as the grouping is unknown). As an illustration, for the topic *describe the life cycle of a tree* which is unseen at training time, for the three paragraphs on the topic, ProStruct predicts that tree is created; not-changed; and created respectively, while LaCE correctly predicts that tree is created;

created; and created respectively. This illustrates a case where LaCE has learned to make predictions that are more consistent and correct.

Error Analysis for LaCE

To understand LaCE's behavior further, we examined cases where LaCE's and ProStruct's predictions differ, and examined their agreement with gold labels. In this analysis we found three major sources of errors for LaCE:

- **The label consistency assumption does not always hold:** In Section 3.1, we explain that LaCE relies on summary labels being consistent across examples in the same group. We found that for some of the topics in our training dataset this assumption is sometimes violated. E.g., for the topic *How does the body control its blood sugar level?*, there are two different paragraphs; one of them describes the entity `sugar` as being `Created` and then `Destroyed` to create `bloodsugar`, while the other paragraph describes the same event in a different way by saying that the entity `sugar` is *`Created`* and then *`Moved`* to the blood. LaCE can thus goes wrong when trying to enforce consistency in such cases.

- **Lexical variance between entities across paragraphs:** Different paragraphs about the same topic may describe the procedure using different wordings, resulting in errors. For example, in paragraphs about the topic *what happens during photosynthesis?*, the same entity (`carbon dioxide`) is referred to by two different strings, $\mathtt{CO}_2$ in one paragraph and `carbon dioxide` in another. Currently, LaCE does not take into account entity synonyms, so it is unable to encourage consistency here. An interesting line of future work would be to use the embedding space similarity between entity names, to help address this problem.

- **LaCE can make incorrect predictions to improve consistency:** For the topic *Describe how to make a cake* at training time, when presented with two paragraphs, LaCE tries to be consistent and incorrectly predicts that `cake` is *`Destroyed`* in both paragraphs. ProStruct does not attempt to improve prediction consistency, here resulting in less consistent but in this case more accurate predictions for this topic.

5.5 Directions For Enhancing LaCE

- Improve LaCE for ProPara: LaCE's performance on ProPara can be improved further by a) soft matching of entities across paragraphs instead of current exact string match b) exploring more systematic ways (e.g., simulated annealing) to define adaptive loss c) using additional sources of unlabeled data (e.g., web, textbooks) weighed by their reliability.

- Apply LaCE on other tasks: Architecturally, LaCE is a way to train any existing structured prediction model for a given task to produce consistent labels across similar datapoints. Hence it can be easily applied to other tasks where parallel data is available (grouping function) and there is a way to efficiently compare predictions (summary labels) across parallel datapoints, e.g. event extraction from parallel news articles (Chinchor, 2002).

  Further, summary labels need not be action categories (e.g., `Created`, `Destroyed`). Consistency can also be computed for QA task where multiple parallel text is available for reading comprehension. We plan to explore this direction in the future.

6 Conclusion

Our goal is procedural text comprehension, a task that current systems still struggle with. Our approach has been to exploit the fact that, for many procedures, multiple independent descriptions exist, and that we expect some consistency between those descriptions. To do this, we have presented a task- and model-general learning framework, LaCE, that can leverage this expectation, allowing consistency bias to be built into the learned model. Applying this framework to procedural text, the resulting system obtains new state-of-the-art results on the ProPara dataset, an existing benchmark for procedural text comprehension. It also demonstrates the ability to benefit from unlabeled paragraphs (semi-supervised learning), something that prior systems for this task were unable to do. We have also identified several avenues for further improvement (Section 5.4), and are optimistic that further gains can be achieved.

Acknowledgements

Computations on beaker.org were supported in part by credits from Google Cloud.

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP'14*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR*, abs/1606.02858.

Kevin Chen, Christopher B. Choy, Manolis Savva, Angel X. Chang, Thomas A. Funkhouser, and Silvio Savarese. 2018. Text2shape: Generating shapes from natural language by learning joint embeddings. *CoRR*, abs/1803.08495.

Nancy A. Chinchor. 2002. Message understanding conference (muc) tests of discourse processing.

Charles LA Clarke, Gordon V Cormack, and Thomas R Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. *NAACL-HLT'18*, arXiv preprint arXiv:1805.06975.

Susan T. Dumais, Michele Banko, Eric Brill, Jimmy J. Lin, and Andrew Y. Ng. 2002. Web question answering: is more always better? In *SIGIR*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Philip Haeusser, Alexander Mordvintsev, and Daniel Cremers. 2017. Learning by association-a versatile semi-supervised training method for neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 6.

Viktor Hangya, Fabienne Braune, Alexander Fraser, and Hinrich Schütze. 2018. Two methods for domain adaptation of bilingual tasks: Delightfully simple and broadly applicable. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proc. EMNLP'15*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proc. EMNLP'16*.

Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi. 1988. Optimization by simulated annealing.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hinrich Schütze, Fabienne Braune, Alexander M. Fraser, and Viktor Hangya. 2018. Two methods for domain adaptation of bilingual tasks: Delightfully simple and broadly applicable. In *ACL*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *ICLR*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP*.

Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wen-tau Yihs, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. *EMNLP'18*, arXiv preprint arXiv:1808.10012.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *NIPS*.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML)*, pages 912–919.

MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms

Aida Amini[1], Saadia Gabriel[1], Shanchuan Lin[1], Rik Koncel-Kedziorski[1],
Yejin Choi[1,2], and Hannaneh Hajishirzi[1,2]

[1]University of Washington
[2]Allen Institute for AI
{*amini91, skgabrie, linsh, kedzior, yejin, hannaneh*}*@cs.washington.edu*

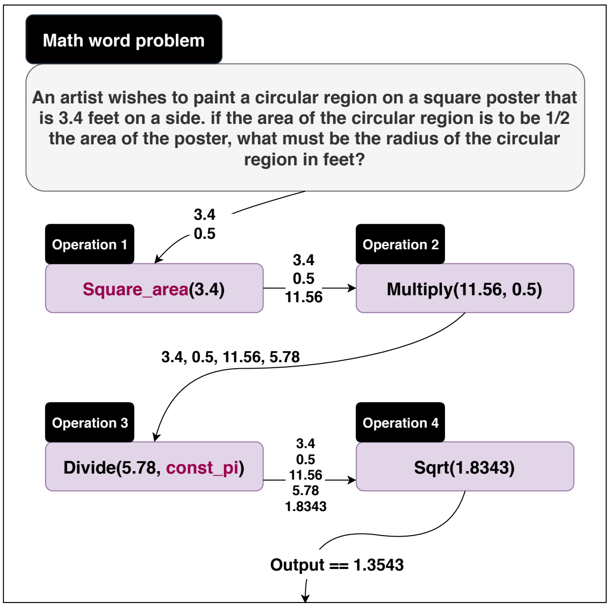
Abstract

We introduce a large-scale dataset of math word problems and an interpretable neural math problem solver that learns to map problems to operation programs. Due to annotation challenges, current datasets in this domain have been either relatively small in scale or did not offer precise operational annotations over diverse problem types. We introduce a new representation language to model precise operation programs corresponding to each math problem that aim to improve both the performance and the interpretability of the learned models. Using this representation language, our new dataset, MathQA, significantly enhances the AQuA dataset with fully-specified operational programs. We additionally introduce a neural sequence-to-program model enhanced with automatic problem categorization. Our experiments show improvements over competitive baselines in our MathQA as well as the AQuA datasets. The results are still significantly lower than human performance indicating that the dataset poses new challenges for future research. Our dataset is available at: `https://math-qa.github.io/math-QA/`.

1 Introduction

Answering math word problems poses unique challenges for logical reasoning over implicit or explicit quantities expressed in text. Math word-problem solving requires extraction of salient information from natural language narratives. Automatic solvers must transform the textual narratives into executable meaning representations, a process that requires both high precision and, in the case of story problems, significant world knowledge.

As shown by the geometry question in Figure 1, math word problems are generally narratives describing the progress of actions and relations over some entities and quantities. The operation program underlying the problem in Figure 1 highlights the complexity of the problem-solving task. Here, we need the ability to deduce implied constants (pi) and knowledge of domain-specific formulas (area of the square).



An artist wishes to paint a circular region on a square poster that is 3.4 feet on a side. if the area of the circular region is to be 1/2 the area of the poster, what must be the radius of the circular region in feet?

Figure 1: Example of a math word problem aligned with representation language by crowd-sourced annotation

In this paper, we introduce a new operation-based representation language for solving math word problems. We use this representation language to construct MathQA[1], a new large-scale, diverse dataset of 37k English multiple-choice math word problems covering multiple math domain categories by modeling operation programs corresponding to word problems in the AQuA dataset (Ling et al., 2017). We introduce a neural model for mapping problems to operation programs with domain categorization.

---

[1] The dataset is available at: `https://math-qa.github.io/math-QA/`

Proceedings of NAACL-HLT 2019, pages 2357–2367
Minneapolis, Minnesota, June 2 - June 7, 2019.

Most current datasets in this domain are small in scale (Kushman et al., 2014) or do not offer precise operational annotations over diverse problem types (Ling et al., 2017). This is mainly due to the fact that annotating math word problems precisely across diverse problem categories is challenging even for humans, requiring background math knowledge for annotators. Our representation language facilitates the annotation task for crowd-sourcing and increases the interpretability of the proposed model.

Our sequence-to-program model with categorization trained on our MathQA dataset outperforms previous state-of-the-art on the AQuA test set in spite of the smaller training size. These results indicate the superiority of our representation language and the quality of the formal annotations in our dataset. Our model achieves competitive results on MathQA, but is still lower than human performance indicating that the dataset poses new challenges for future research. Our contributions are as follows:

- We introduce a large-scale dataset of math word problems that are densely annotated with operation programs
- We introduce a new representation language to model operation programs corresponding to each math problem that aim to improve both the performance and the interpretability of the learned models.
- We introduce a neural architecture leveraging a sequence-to-program model with automatic problem categorization, achieving competitive results on our dataset as well as the AQuA dataset

2 Background and Related Work

Large-Scale Datasets Several large-scale math word problem datasets have been released in recent years. These include Dolphin18K (Huang et al., 2016), Math23K (Wang et al., 2017) and AQuA. We choose the 2017 AQUA-RAT dataset to demonstrate use of our representation language on an existing large-scale math word problem solving dataset. The AQuA provides over 100K GRE- and GMAT-level math word problems. The problems are multiple choice and come from a wide range of domains.

The scale and diversity of this dataset makes it particularly suited for use in training deep-learning models to solve word problems. However there is a significant amount of unwanted noise in the dataset, including problems with incorrect solutions, problems that are unsolvable without brute-force enumeration of solutions, and rationales that contain few or none of the steps required to solve the corresponding problem. The motivation for our dataset comes from the fact we want to maintain the challenging nature of the problems included in the AQuA dataset, while removing noise that hinders the ability of neuralized models to learn the types of signal neccessary for problem-solving by logical reasoning.

Additional Datasets Several smaller datasets have been compiled in recent years. Most of these works have focused on algebra word problems, including MaWPS (Koncel-Kedziorski et al., 2016), Alg514 (Kushman et al., 2014), and DRAW-1K (Upadhyay and Chang, 2017). Many of these datasets have sought to align underlying equations or systems of equations with word problem text. While recent works like (Liang et al., 2018; Locascio et al., 2016) have explored representing math word problems with logical formalisms and regular expressions, our work is the first to provide well-defined formalisms for representing intermediate problem-solving steps that are shown to be generalizable beyond algebra problems.

Solving with Handcrafted Features Due to sparsity of suitable data, early work on math word problem solving used pattern-matching to map word problems to mathematical expressions (Bobrow, 1964; Charniak, 1968, 1969), as well as non-neural statistical modeling and semantic parsing approaches (Liguda and Pfeiffer, 2012).

Some effort has been made on parsing the problems to extract salient entities (Hosseini et al., 2017). This approach views entities as containers, which can be composed into an equation tree representation. The equation tree representation is changed over time by operations implied by the problem text.

Many early works focused on solving addition and subtraction problems (Briars and Larkin, 1984; Dellarosa, 1986; Bakman, 2007). As word problems become more diverse and complex, we require models capable of solving simultaneous equation systems. This has led to an increasing focus on finding semantic alignment of math word problems and mentions of numbers (Roy and Roth, 2018). The main idea behind those work is to find all possible patterns of equations and rank them based on the problem.

Neural Word Problem Solvers Following the increasing availability of large-scale datasets like AQuA, several recent works have explored deep neural approaches to math word problem solving (Wang et al., 2017). Our representation language is motivated by exploration of using intermediate formalisms in the training of deep neural problem-solving networks, as is done in the work of (Huang et al., 2018b) to solve problems with sequence to sequence models. While this work focused on single-variable arithmetic problems, our work introduces a formal language of operations for covering more complex multivariate problems and systems of equations.

Interpretability of Solvers While the statistical models with handcrafted features introduced by prior work are arguably "interpretable" due to the relative sparsity of features as well as the clear alignments between inputs and outputs, new neuralized approaches present new challenges to model interpretability of math word problem solvers (Huang et al., 2018a). While this area is relatively unexplored, a prior approach to increasing robustness and interpretability of math word problem-solving models looks at using an adversarial dataset to determine if models are learning logical reasoning or exploiting dataset biases through pattern-matching (Liang et al., 2018).

3 Representing Math Word Problems

A math word problem consists of a narrative that grounds mathematical formalisms in real-world concepts. Solving these problems is a challenge for both humans and automatic methods like neural network-based solvers, since it requires logical reasoning about implied actions and relations between entities. For example, in Figure 2, operations like addition and division are not explicitly mentioned in the word problem text, but they are implied by the question. As we examine the context of a math word problem, we have to select arguments for operations based on which values are unimportant for solving the problem and which are salient. In Figure 2, the numeric value "100" appears in the context but does not appear in the underlying equation.

By selecting implied operations and arguments, we can generate a program of intermediate steps for solving a math word problem. Each step involves a mathematical operation and its related arguments. In Figure 2, there are three addition operations and one division. As illustrated in the figure, operations can be dependant to the previous ones by the values they use as arguments. Every math word problem can be solved by sequentially executing these programs of dependent operations and arguments.

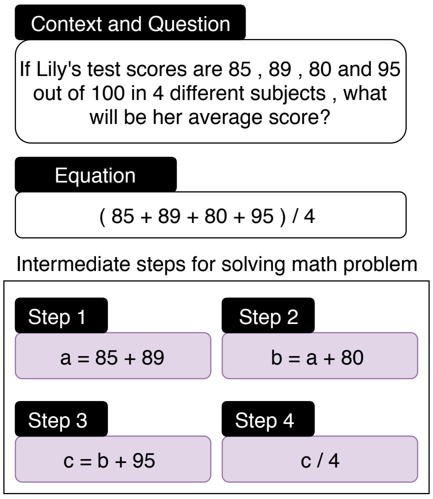


Figure 2: Example of a math word problem with its underlying equation and intermediate steps for problem-solving

We define formalisms for expressing these sequential operation programs with a domain-aware representation language. An operation program in our representation language is a sequence with n operations. The general form is shown below. Each operation o_i takes in a list of arguments $\mathbf{a}$ of length i:

$$o_1(\mathbf{a}_1)o_2(\mathbf{a}_2)...o_n(\mathbf{a}_n) \tag{1}$$

Given this general definition, the problem in Figure 2 has the following representation[2]:

$$\begin{aligned} \text{add}_1(85, 89)\text{add}_2(174, 80) \\ \text{add}_3(254, 95)\text{divide}_4(349, 4) \end{aligned} \tag{2}$$

Our representation language consists of 58 operations and is designed considering the following objectives.

- Correctness → Operation programs should result in the correct solution when all operations are executed.
- Domain-awareness → Operation problems should make use of both math knowledge and

---

[2] Here the arguments 174, 254 and 349 are the outputs of operations 1, 2 and 3 respectively.

domain knowledge associated with subfields like geometry and probability to determine which operations and arguments to use.

- Human interpretability → Each operation and argument used to obtain the correct solution should relate to part of the input word problem context or a previous step in the operation program.

Learning logical forms has led to success in other areas of semantic parsing (Cheng et al., 2017; Zelle and Mooney, 1996; Zettlemoyer and Collins, 2007, 2005) and is a natural representation for math word problem-solving steps. By augmenting our dataset with these formalisms, we are able to cover most types of math word problems[3]. In contrast to other representations like simultaneous equations, our formalisms ensure that every problem-solving step is aligned to a previous one. There are three advantages to this approach. First, we use this representation language to provide human annotators with clear steps for how a particular problem should be solved with math and domain knowledge. Second, our formalisms provide neural models with a continuous path to execute operations for problems with systems of equations, instead of forcing models to align equations before problem solving. This reduces the possibility of intermediate errors being propagated and leading to a incorrect solution. Finally, by having neural models generate a solution path in our representation language before computing the final solution, we are able to reconstruct the logical hops inferred by the model output, increasing model interpretability.

4 Dataset

Our dataset (called MathQA) consists of 37,200 math word problems, corresponding lists of multiple-choice options and aligned operation programs. We use problems in the AQuA dataset and carefully annotate those problems with formal operation programs.

Math problems are first categorized into math domains using term frequencies (more details in Section 5.2). These domains are used to prune the search space of possible operations to align with the word problem text. Figure 3 shows the category-based hierarchies for operation formalisms.

Category	#Prob.	Avg #words	#Vocab	Avg #ops
Geometry	3,316	34.3	1,839	4.8
Physics	9,830	37.3	3,340	5.0
Probability	663	38.9	937	5.0
Gain-Loss	4,377	34.3	1,533	5.7
General	17,796	38.6	6,912	5.1
Other	1,277	31.3	1,425	4.7
All	37,259	37.9	6,664	5.3

Table 1: Statistics for our dataset; the total number of operations in the dataset is 58.

We use crowdsourcing to carefully align problems with operation programs (Section 4.1). Table 1 shows overall statistics of the dataset.[4]

4.1 Annotation using Crowd Workers

Annotating GRE level math problems can be a challenging and time consuming task for humans. We design a dynamic annotation platform to annotate math word problems with formal operation programs. Our annotation platform has the following properties: (a) it provides basic math knowledge to annotators, (b) it is dynamic by iteratively calculating intermediate results after an operation submission, and (c) it employs quality control strategies.

Dynamic Annotation Platform The annotators are provided with a problem description, a list of operations related to the problem category, and a list of valid arguments. They iteratively select operations and arguments until the problem is solved.

- **Operation Selection** The annotators are instructed to sequentially select an operation from the list of operations in the problem category. Annotators are provided with math knowledge by hovering over every operation and getting the related hint that consists of arguments, formula and a short explanation of the operation.

- **Argument Selection** After selecting the operation the list of valid arguments are presented

---

[3] We omit high-order polynomials and problems where the solutions are entirely nonnumeric.

[4] We also experimented with an automatic dynamic programming approach to annotation that generates operation programs for problems using numbers in the AQuA rationales. Due to the noise in the rationales, only 61% of those problems pass our human validation. This is mainly due to the fact that the rationales are not complete programs and fail to explicitly describe all important numbers and operations required to solve the problem. To maintain interpretability of operation paths, we did not include automatic annotations from our dataset and focus on operation programs derived by crowdsourcing.

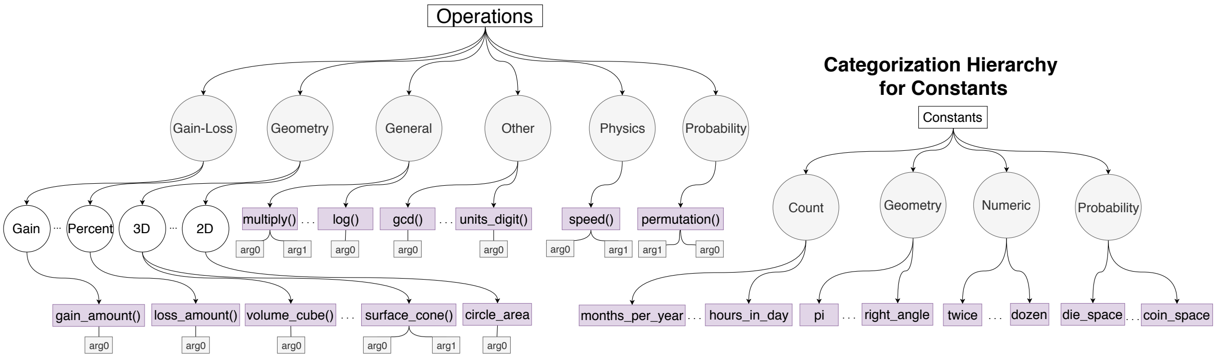


Figure 3: Category-based Hierarchies for Operation Formalisms

  to the annotators to choose from. Valid arguments consist of numbers in the problem, constants in the problem category, and the previous calculations. The annotators are restricted to select only from these valid arguments to prevent having noisy and dangling numbers. After submission of an operation and the corresponding arguments, the result of the operation is automatically calculated and will be added as a new valid argument to the argument list.

- **Program Submission** To prevent annotators from submitting arbitrary programs, we enforce restrictions to the final submission. Our platform only accepts programs which include some numbers from the problem, and whose final calculation is very close to the correct numeric solution.

High Quality Crowd Workers We dynamically evaluate and employ high-quality annotators through a collection of quality-control questions. We take advantage of the annotation platform in *Figure Eight*.[5] The annotators are randomly evaluated through a pre-defined set of test questions, and they have to maintain an accuracy threshold to be able to continue their annotations. If an annotator's accuracy drops below a threshold, their previous annotations are labeled as untrusted and will be added to the pool of annotations again.

Alignment Validation To further evaluate the quality of the annotated programs, we leverage a validation strategy to check whether the problems and annotated programs are aligned or not. According to this strategy, at least 2 out of 3 validators should rank the operation program as valid for it to be selected. The validation accuracy is 94.64% across categories.

5 Models

We develop encoder-decoder neural models to map word problems to a set of feasible operation programs. We match the result of the executed operation program against the list of multiple-choice options given for a particular problem. The matching solution is the final model output.

We frame the problem of aligning an operation program with a math word problem as a neural machine translation (NMT) task, where the word problem $\mathbf{x}$ and gold operation program $\mathbf{y}$ form a parallel text pair. The vocabulary of $\mathbf{y}$ includes all possible operations and arguments in our representation language.

5.1 Sequence-to-Program

For our initial sequence-to-program model, we follow the attention-based NMT paradigm of (Bahdanau et al., 2015; Cho et al., 2014). We encode the source word problem text $\mathbf{x} = (x_1, x_2, ..., x_M)$ using a bidirectional RNN encoder θ^{enc}. The decoder θ^{dec} predicts a distribution over the vocabulary and input tokens to generate each operation or argument in the target operation program. For our sequence-to-program model vocabulary, we use informed generation, in which the program tokens are generated separately from the vocabulary of operations O or arguments A.

The encoded text is represented by a sequence of d-dimensional hidden states $\mathbf{h}^{enc} = (h_1^{enc}, h_2^{enc}, .., h_M^{enc})$, where M is the length of the input text. A context vector a_i is computed by taking the weighted sum of the attention model weights $\alpha_{t,i}$ for each timestep $t \in (1, 2, ..., T)$ and each encoder hidden state h_i^{enc}:

$$a_i = \sum_{i=1}^{M} \alpha_{t,i} h_i^{enc}.$$

We compute the d-dimensional decoder hidden

---

[5] https://www.figure-eight.com

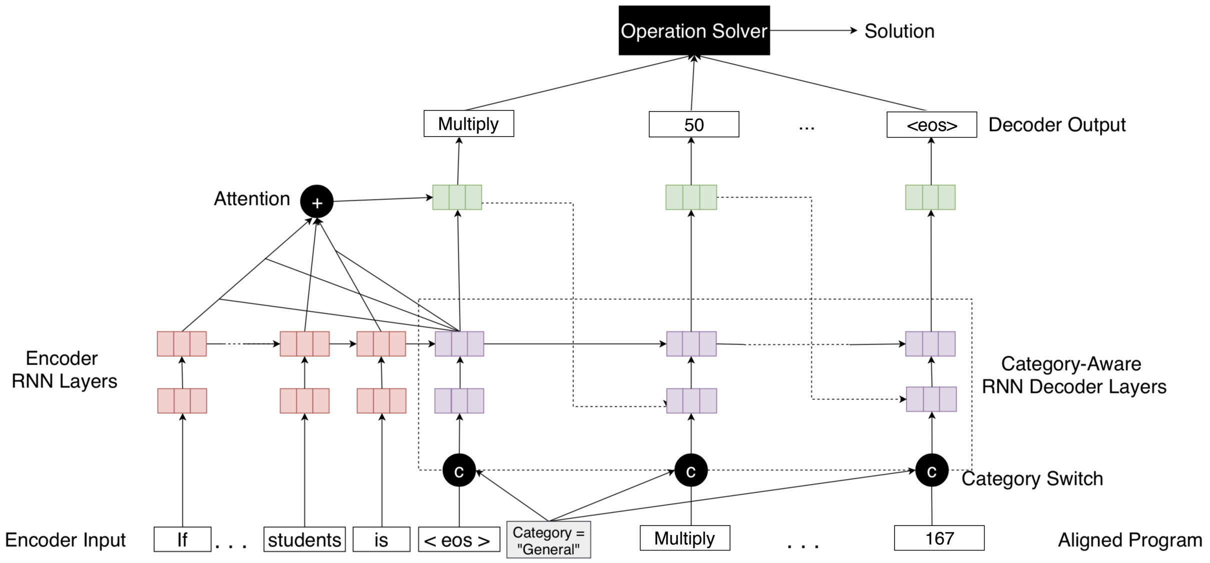


Figure 4: Architecture of sequence-to-program model with categorization. Shown with example problem "If the average marks of three batches of 62 , 60 and 45 students respectively is 50 , 55 , 60 , then the average marks of all the students is."

state h_i^{dec} using a LSTM recurrent layer:

$$h_i^{dec} = LSTM(h_{i-1}^{dec}, y_{i-1}, a_i) \tag{3}$$

At each timestep, we make a prediction for an operator op_i or argument arg_{ik}, where k corresponds to the index of the argument in operator i's argument list. This prediction is conditioned on the previous tokens $(y_1, ..., y_{i-1})$ and the input $\mathbf{x}$ to decode an entire operation program $\mathbf{y} = (y_1, y_2, ..., y_N)$ of length N:

$$P(\mathbf{y}|x) = \prod_{i=1}^{N} P(y_i|y_{<i}, \mathbf{x}) \tag{4}$$

$$P(y_i|y_{<i}, \mathbf{x}) = g(f(h_i^{dec}, y_i, a_i)) \tag{5}$$

Here f is a 1-layer feed-forward neural network and g is the softmax function. During training time, we minimize the negative log-likelihood (NLL) using the following objective:

$$\mathcal{L}(\theta^{enc}, \theta^{dec}) = -logP(\mathbf{y}|\mathbf{x}; \theta^{enc}, \theta^{dec}) \tag{6}$$

At test time, we only observe the input text when predicting operation programs:

$$\hat{\mathbf{y}} = \texttt{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{7}$$

5.2 Categorized Sequence-to-Program Model

We extend our base sequence-to-program model to integrate knowledge of math word problem domain categories. We modify the RNN decoder layers that compute the decoder hidden state to be category-aware. Here, the category label c is deterministically computed by the category extractor (explained below). It functions as a hard decision switch that determines which set of parameters to use for the hidden state computation:

$$h_i^{dec} = LSTM_c(h_{i-1}^{dec}, y_{i-1}, a_i) \tag{8}$$

The updated objective function from equation (7) is shown below:

$$\mathcal{L}(\theta^{enc}, \theta_c^{dec}) = -logP(\mathbf{y}|\mathbf{x}; \theta^{enc}, \theta_c^{dec}) \tag{9}$$

The full model architecture is shown in Figure 4.

Domain-Specific Category Extraction We first construct a lexicon of n-grams relating to a specific domain. The lexicon is a list consisting of domain-specific categories and associated n-grams. For each domain category c in the lexicon, we select associated n-grams $\mathbf{n}_c$ that occur frequently in word problems belonging to domain category c, but rarely appear in other domain categories. We compute n-gram frequency f_{pc} as the number of n-grams associated with a category c appearing in the text of a word problem p. We obtain a list of potential categories for p by choosing all categories for which $f_{pc} > 0$, and then assign a category label to p based on which category has the highest n-gram frequency.

5.3 Solving Operation Programs

Once a complete operation program has been decoded, each operator in the program is executed sequentially along with its predicted set of arguments to obtain a possible solution. For each word problem p and options o, we generate a beam of the top n decoded operation programs. We execute each decoded program g to find the solution from the list of options $\mathbf{o}$ of the problem. We first choose options that are within a threshold of the executed value of g. We select g as the predicted solution by checking the number of selected options and the minimum distance between the executed value of g and a possible option for p. For the problems in AQuA that do not belong in any category of MathQA, we randomly choose an option.

6 Experimental Setup

6.1 Datasets

Our dataset consists of $37k$ problems which are randomly split in $(80/12/8)\%$ training/dev/test problems. Our dataset significantly enhances the AQuA dataset by fully annotating a portion of *solvable* problems in the AQuA dataset into formal operation programs.

We carefully study the AQuA dataset. Many of the problems are near-duplicates with slight changes to the math word problem stories or numerical values since they are expanded from a set of 30,000 seed problems through crowdsourcing (Ling et al., 2017). These changes are not always reflected in the rationales, leading to incorrect solutions. There are also some problems that are not solvable given current math word problem solving frameworks because they require a level of reasoning not yet modeled by neural networks. Sequence problems, for example, require understanding of patterns that are difficult to intuit without domain knowledge like sequence formulas, and can only be solved automatically through brute-force or guessing. Table 2 shows a full breakdown of the AQuA dataset by solvability.[6]

Subset	Train	Valid
Unsolvable - No Words	37	0
Unsolvable - Sequence	1,991	4
Unsolvable - Requires Options	6,643	8
Unsolvable - Non-numeric	10,227	14
Duplicates	17,294	0
Solvable	65,991	229
Total	97,467	254

Table 2: Full original AQuA solvability statistics.

6.2 Annotation Details

We follow the annotation strategy described in Section 4 to formally annotate problems with operation programs. [7]

Annotator Agreements and Evaluations Our expert evaluation of the annotation procedure for a collection of 500 problems shows that 92% of the annotations are valid. Additionally, it has 87% agreement between the expert validation and the crowd sourcing validation task.

Annotation Expansion The AQuA dataset consists of a group of problems which share similar characteristics. These problems can be solved with similar operation programs. We find closely similar problems, replace numeric values with generic numbers, and expand annotations to cover more problems from the AQuA dataset. For similarity, we use Levenshtein distance with a threshold of 4 words in edit distance.

6.3 Model and Training Details

We use the official python implementation of OpenNMT (Klein et al.). We choose a LSTM-based encoder-decoder architecture. We use Adam optimizer (Kingma and Ba, 2015), and the learning rate for training is 0.001. The hidden size for the encoder and decoder is set to $d = 100$. Both the encoder and decoder have 2 layers. The word embedding vectors are randomly initialized. At inference time, we implemented a beam search with beam size of 200 for AQuA and 100 for MathQA.

The program vocabulary consists of the operations O in our representation language and valid arguments A. For valid arguments, we do not use their actual values since the space is very large. Instead, we keep a list of numbers according to their source. Constants are predefined numbers that are available to all problems. Problem numbers are added to the list according to their order in the problem text. Calculated numbers in the interme-

---

[6] There is overlap between unsolvable subsets. For example, a sequence problem may also be a duplicate of another problem in the AQuA dataset.

[7] We tried two other strategies of showing extra information (rationales or end solutions) to annotators to facilitate solving problems. However, our manual validation showed that annotators mostly used those extra information to artificially build an operation program without reading the problem.

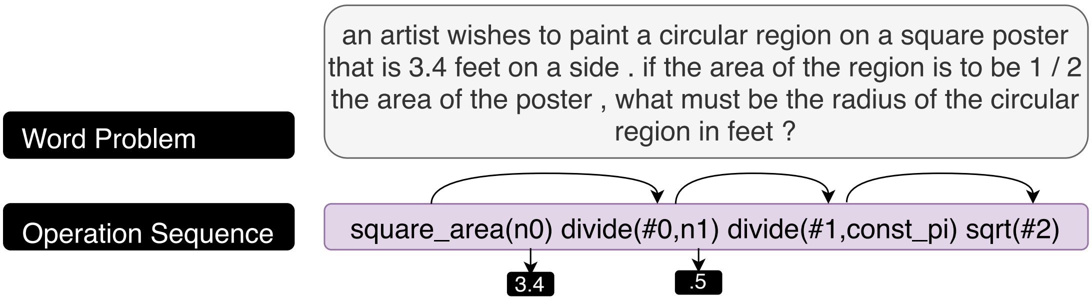


Figure 5: Example of an operation program generated by our Seq2prog model with categorization

Model	MathQA	AQuA
Random	20.0	20.0
AQuA Model	-	36.4
Seq2prog	51.9	33.0
Seq2prog + cat	**54.2**	**37.9**

Table 3: Experimental results for accuracy on our MathQA and AQuA test sets

diate steps are added to the list according to the operation order.

7 Experimental Results

7.1 Results

Table 3 compares the performance of our sequence-to-program models trained on MathQA with baselines on MathQA and AQuA test sets. The base model is referred to as "Seq2prog," while our model with categorization is "Seq2prog + cat." For accuracy, the performance was measured in terms of how well the model would perform on an actual math test.

We observe improvement for our "Seq2prog + cat" model despite the fact that our training data is proportionally smaller than the AQuA dataset, and our model is much simpler than the state-of-the-art model on this dataset. This indicates the effectiveness of our formal representation language to incorporate domain knowledge as well as the quality of the annotations in our dataset.

7.2 Analysis

Qualitative Analysis Table 5 and Figure 5 show some examples of problems solved by our method. We analyzed 50 problems that are solved wrongly by our system on the MathQA dataset. Table 4 summarizes four major categories of errors.

The most common type of errors are problems that need complicated or long chain of mathematical reasoning. For example, the first problem in Table 4 requires reasoning that goes beyond one sentence. Other errors are due to limitations in our representation language. For example, the second problem in Table 4 requires the *factorization* operation which is not defined in our representation language. Future work can investigate more domains of mathematics such as logic, number factors, etc. Some errors are due to the slightly noisy nature of our categorization strategy. For example, the third problem in Table 4 is mistakenly categorized as belonging to *physics* domain due to the presence of words *m, cm, liter* in the problem text, while the correct category for the problem is *geometry*. The final category of errors are due to problems that do not have enough textual context or erroneous problems (e.g., fourth problem in Table 4).

Impact of Categorization Table 3 indicates that our category-aware model outperforms the base model on both AQuA and MathQA datasets. The gain is relatively small because the current model only uses categorization decisions as hard constraints at decoding time. Moreover, the problem categorization might be noisy due to our use of only one mathematical interpretation for each domain-specific n-gram. For example, the presence of the words "square" or "cube" in the text of a math word problem indicate that the word problem is related to the geometry domain, but these unigrams can also refer to an exponential operation (n^2 or n^3).

To measure the effectiveness of our categorization strategy, we used human annotation over 100 problems. The agreement between human annotators is 84% and their agreement with our model is 74.5%. As a future extension of this work, we would like to also consider the context in which domain-specific n-grams appear.

Discussions As we mentioned in section 3, the continuous nature of our formalism allows us to solve problems requiring systems of equations. However, there are other types of word prob-

Error type	**Problem**
Hard problems (45%)	Jane and Ashley take 8 days and 40 days respectively to complete a project when they work on it alone. They thought if they worked on the project together, they would take fewer days to complete it. During the period that they were working together, Jane took an eight day leave from work. This led to Jane' s working for four extra days on her own to complete the project. How long did it take to finish the project?
Limitation in representation language (25%)	How many different positive integers are factors of 25?
Categorization errors (12.5%)	A cistern of capacity 8000 litres measures externally 3.3 m by 2.6 m by 1.3 m and its walls are 5 cm thick. The thickness of the bottom is:
Incorrect or insufficient problem text) (17.5%)	45 x ? = 25 % of 900

Table 4: Examples of mistakes made by our system. The reason of the errors are underlined.

Problem : A rectangular field is to be fenced on three sides leaving a side of 20 feet uncovered. if the area of the field is 10 sq. feet, how many feet of fencing will be required? **Operations :** `divide(10,20), multiply(#0, const_2), add(20, 1)`
Problem : How long does a train 110m long running at the speed of 72 km/hr takes to cross a bridge 132m length? **Operations :** `add(110, 132), multiply(72, const_0.2778), divide(#0, #1), floor(#2)`

Table 5: Problems solved correctly by Seq2prog+cat model.

lems that are currently unsolvable or have multiple interpretations leading to multiple correct solutions. While problems that can only be solved by brute-force instead of logical reasoning and non-narrative problems that do not fit the definition of a math word problem (in Table 2 these appear as "no word") are removed from consideration, there are other problems that are beyond the scope of current models but could pose an interesting challenge for future work. One example is the domain of sequence problems. Unlike past word problem-solving models, our models incorporate domain-specific math knowledge, which is potentially extensible to common sequence and series formulas.

8 Conclusion

In this work, we introduced a representation language and annotation system for large-scale math word problem-solving datasets that addresses unwanted noise in these datasets and lack of formal operation-based representations. We demonstrated the effectiveness of our representation language by transforming solvable AQuA word problems into operation formalisms. Experimental results show that both our base and category-aware sequence-to-program models outperform baselines and previous results on the AQuA dataset when trained on data aligned with our representation language. Our representation language provides an extra layer of supervision that can be used to reduce the influence of statistical bias in datasets like AQuA. Additionally, generated operation programs like the examples in figure 5 demonstrate the effectiveness of these operation formalisms for representing math word problems in a human interpretable form.

The gap between the performance of our models and human performance indicates that our MathQA still maintains the challenging nature of AQuA problems. In future work, we plan to extend our representation language and models to cover currently unsolvable problems, including sequence and high-order polynomial problems.

Acknowledgements This research was supported by ONR (N00014-18-1-2826), NSF (IIS 1616112), Allen Distinguished Investigator Award, and gifts from Google, Allen Institute for AI, Amazon, and Bloomberg. We thank Marti A. Hearst, Katie Stasaski, and the anonymous reviewers for their helpful comments.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Machine translation by jointly learning to align and translate. In *ICLR*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. In *arXiv preprint math/0701393*.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Diane J Briars and Jill H Larkin. 1984. An integrated model of skill in solving elementary word problems. In *Cognition and instruction 1(3)*, pages 245–296.

Eugene Charniak. 1968. Calculus word problems. Ph.D. thesis, Massachusetts Institute of Technology.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st international joint conference on Artificial intelligence*, pages 303–316. Morgan Kaufmann Publishers Inc.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoderdecoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Denise Dellarosa. 1986. A computer simulation of childrens arithmetic word-problem solving. In *Behavior Research Methods, Instruments, Computers 18(2)*, pages 147–154.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2017. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018a. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018b. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *the 3rd International Conference for Learning Representations (ICLR)*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. Opennmt: Open-source toolkit for neural machine translation. *ArXiv e-prints*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *NAACL*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2018. A meaning-based statistical english math word problem solver. In *Proceedings of NAACL-HLT 2018*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *International Conference on Application of Natural Language to Information Systems*, pages 247–252. Springer.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of ACL*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 247–252. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. In *Proceedings of NAACL-HLT 2018*.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055. AAAI Press/MIT Press.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI 05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language*

Processing and Computational Natural Language Learning (EMNLPCoNLL), pages 678–687. Association for Computational Linguistics.

DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs

Dheeru Dua♣, **Yizhong Wang**♢[*], **Pradeep Dasigi**♡,
Gabriel Stanovsky♡+, **Sameer Singh**♣, and **Matt Gardner**♠
♣University of California, Irvine, USA
♢Peking University, Beijing, China
♡Allen Institute for Artificial Intelligence, Seattle, Washington, USA
♠Allen Institute for Artificial Intelligence, Irvine, California, USA
+University of Washington, Seattle, Washington, USA
`ddua@uci.edu`

Abstract

Reading comprehension has recently seen rapid progress, with systems matching humans on the most popular datasets for the task. However, a large body of work has highlighted the brittleness of these systems, showing that there is much work left to be done. We introduce a new English reading comprehension benchmark, DROP, which requires **D**iscrete **R**easoning **O**ver the content of **P**aragraphs. In this crowdsourced, adversarially-created, 96k-question benchmark, a system must resolve references in a question, perhaps to multiple input positions, and perform discrete operations over them (such as addition, counting, or sorting). These operations require a much more comprehensive understanding of the content of paragraphs than what was necessary for prior datasets. We apply state-of-the-art methods from both the reading comprehension and semantic parsing literatures on this dataset and show that the best systems only achieve 32.7% F_1 on our generalized accuracy metric, while expert human performance is 96.4%. We additionally present a new model that combines reading comprehension methods with simple numerical reasoning to achieve 47.0% F_1.

1 Introduction

The task of *reading comprehension*, where systems must understand a single passage of text well enough to answer arbitrary questions about it, has seen significant progress in the last few years, so much that the most popular datasets available for this task have been solved (Chen et al., 2016; Devlin et al., 2019). We introduce a substantially more challenging English reading comprehension dataset aimed at pushing the field towards more comprehensive analysis of paragraphs of text. In this new benchmark, which we call DROP, a system is given a paragraph and a question and must perform some kind of **D**iscrete **R**easoning **O**ver the text in the **P**aragraph to obtain the correct answer.

These questions that require discrete reasoning (such as addition, sorting, or counting; see Table 1) are inspired by the complex, compositional questions commonly found in the semantic parsing literature. We focus on this type of questions because they force a structured analysis of the content of the paragraph that is detailed enough to permit reasoning. Our goal is to further *paragraph understanding*; complex questions allow us to test a system's understanding of the paragraph's semantics.

DROP is also designed to further research on methods that combine distributed representations with symbolic, discrete reasoning. In order to do well on this dataset, a system must be able to find multiple occurrences of an event described in a question (presumably using some kind of soft matching), extract arguments from the events, then perform a numerical operation such as a sort, to answer a question like *"Who threw the longest touchdown pass?"*.

We constructed this dataset through crowdsourcing, first collecting passages from Wikipedia that are easy to ask hard questions about, then encouraging crowd workers to produce challenging questions. This encouragement was partially through instructions given to workers, and partially through the use of an *adversarial baseline*: we ran a baseline reading comprehension method (BiDAF) (Seo et al., 2017) in the background as crowd workers were writing questions, requiring them to give questions that the baseline system could not correctly answer. This resulted in a dataset of 96,567 questions from a variety of categories in Wikipedia, with a particular emphasis on sports game summaries and history passages. The answers to the questions are required to be spans in the passage or

---

[*] Work done as an intern at the Allen Institute for Artificial Intelligence in Irvine, California.

Proceedings of NAACL-HLT 2019, pages 2368–2378
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

question, numbers, or dates, which allows for easy and accurate evaluation metrics.

We present an analysis of the resulting dataset to show what phenomena are present. We find that many questions combine complex question semantics with SQuAD-style argument finding; e.g., in the first question in Table 1, BiDAF correctly finds the amount the painting sold for, but does not understand the question semantics and cannot perform the numerical reasoning required to answer the question. Other questions, such as the fifth question in Table 1, require finding all events in the passage that match a description in the question, then aggregating them somehow (in this instance, by counting them and then performing an argmax). Very often entity coreference is required. Table 1 gives a number of different phenomena, with their proportions in the dataset.

We used three types of systems to judge baseline performance on DROP: (1) heuristic baselines, to check for biases in the data; (2) SQuAD-style reading comprehension methods; and (3) semantic parsers operating on a pipelined analysis of the passage. The reading comprehension methods perform the best, with our best baseline achieving 32.7% F_1 on our generalized accuracy metric, while expert human performance is 96.4%. Finally, we contribute a new model for this task that combines limited numerical reasoning with standard reading comprehension methods, allowing the model to answer questions involving counting, addition and subtraction. This model reaches 47% F_1, a 14.3% absolute increase over the best baseline system.

The dataset, code for the baseline systems, and a leaderboard with a hidden test set can be found at `https://allennlp.org/drop`.

2 Related Work

Question answering datasets With systems reaching human performance on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), many follow-on tasks are currently being proposed. All of these datasets throw in additional complexities to the reading comprehension challenge, around tracking conversational state (Reddy et al., 2019; Choi et al., 2018), requiring passage retrieval (Joshi et al., 2017; Yang et al., 2018; Talmor and Berant, 2018), mismatched passages and questions (Saha et al., 2018; Kociský et al., 2018; Rajpurkar et al., 2018), integrating knowledge from external sources (Mihaylov et al., 2018; Zhang et al., 2019), tracking entity state changes (Mishra et al., 2018; Ostermann et al., 2018) or a particular kind of "multi-step" reasoning over multiple documents (Welbl et al., 2018; Khashabi et al., 2018). Similar facets are explored in medical domain datasets (Pampari et al., 2018; Šuster and Daelemans, 2018) which contain automatically generated queries on medical records based on predefined templates. We applaud these efforts, which offer good avenues to study these additional phenomena. However, we are concerned with *paragraph understanding*, which on its own is far from solved, so DROP has none of these additional complexities. It consists of single passages of text paired with independent questions, with only linguistic facility required to answer the questions.[1] One could argue that we are adding numerical reasoning as an "additional complexity", and this is true; however, it is only simple reasoning that is relatively well-understood in the semantic parsing literature, and we use it as a necessary means to force more comprehensive passage understanding.

Many existing algebra word problem datasets also contain similar phenomena to what is in DROP (Koncel-Kedziorski et al., 2015; Kushman et al., 2014; Hosseini et al., 2014; Clark et al., 2016; Ling et al., 2017). Our dataset is different in that it uses much longer contexts, is more open domain, and requires deeper paragraph understanding.

Semantic parsing The semantic parsing literature has a long history of trying to understand complex, compositional question semantics in terms of some grounded knowledge base or other environment (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Berant et al., 2013a, *inter alia*). It is this literature that we modeled our questions on, particularly looking at the questions in the WikiTableQuestions dataset (Pasupat and Liang, 2015). If we had a structured, tabular representation of the content of our paragraphs, DROP would be largely the same as WikiTableQuestions, with similar (possibly even simpler) question semantics. Our novelty is that we are the first to combine these complex questions with paragraph understanding, with the aim of encouraging systems that can produce comprehensive structural analyses of paragraphs, either explicitly or implicitly.

Adversarial dataset construction We continue

---

[1] Some questions in our dataset require limited sports domain knowledge to answer; we expect that there are enough such questions that systems can reasonably learn this knowledge from the data.

Reasoning	**Passage** (some parts shortened)	**Question**	**Answer**	**BiDAF**
Subtraction (28.8%)	That year, his **Untitled (1981)**, a painting of a haloed, black-headed man with a bright red skeletal body, depicted amid the artists signature scrawls, was **sold by Robert Lehrman for $16.3 million, well above its $12 million high estimate**.	How many more dollars was the Untitled (1981) painting sold for than the 12 million dollar estimation?	4300000	$16.3 million
Comparison (18.2%)	In **1517, the seventeen-year-old King sailed to Castile**. There, his Flemish court . . . . **In May 1518, Charles traveled to Barcelona in Aragon**.	Where did Charles travel to first, Castile or Barcelona?	Castile	Aragon
Selection (19.4%)	In 1970, to commemorate the 100th anniversary of the founding of Baldwin City, **Baker University professor and playwright Don Mueller and Phyllis E. Braun, Business Manager, produced a musical play entitled The Ballad Of Black Jack** to tell the story of the events that led up to the battle.	Who was the University professor that helped produce The Ballad Of Black Jack, Ivan Boyd or Don Mueller?	Don Mueller	Baker
Addition (11.7%)	Before the UNPROFOR fully deployed, the HV clashed with an armed force of the RSK in the village of Nos Kalik, located in a pink zone near Šibenik, and captured the village at 4:45 p.m. on **2 March 1992**. The JNA formed a battlegroup to counterattack the **next day**.	What date did the JNA form a battlegroup to counterattack after the village of Nos Kalik was captured?	3 March 1992	2 March 1992
Count (16.5%) and Sort (11.7%)	Denver would retake the lead with kicker **Matt Prater nailing a 43-yard field goal**, yet Carolina answered as kicker **John Kasay ties the game with a 39-yard field goal**. . . . Carolina closed out the half with **Kasay nailing a 44-yard field goal**. . . . In the fourth quarter, Carolina sealed the win with **Kasay's 42-yard field goal**.	Which kicker kicked the most field goals?	John Kasay	Matt Prater
Coreference Resolution (3.7%)	**James Douglas** was the second son of Sir George Douglas of Pittendreich, and Elizabeth Douglas, daughter David Douglas of Pittendreich. Before **1543 he married Elizabeth**, daughter of James Douglas, 3rd Earl of Morton. **In 1553 James Douglas succeeded to the title and estates of his father-in-law**.	How many years after he married Elizabeth did James Douglas succeed to the title and estates of his father-in-law?	10	1553
Other Arithmetic (3.2%)	Although the movement initially gathered some **60,000 adherents**, the subsequent establishment of the Bulgarian Exarchate **reduced their number by some 75%**.	How many adherents were left after the establishment of the Bulgarian Exarchate?	15000	60,000
Set of spans (6.0%)	According to some sources 363 civilians were killed in **Kavadarci**, 230 in **Negotino** and 40 in **Vatasha**.	What were the 3 villages that people were killed in?	Kavadarci, Negotino, Vatasha	Negotino and 40 in Vatasha
Other (6.8%)	This **Annual Financial Report** is our principal financial statement of accountability. The **AFR gives a comprehensive view** of the Department's financial activities ...	What does AFR stand for?	Annual Financial Report	one of the Big Four audit firms

Table 1: Example questions and answers from the DROP dataset, showing the relevant parts of the associated passage and the reasoning required to answer the question.

a recent trend in creating datasets with adversarial baselines in the loop (Paperno et al., 2016; Minervini and Riedel, 2018; Zellers et al., 2018; Zhang et al., 2019; Zellers et al., 2019). In our case, instead of using an adversarial baseline to filter automatically generated examples, we use it in a crowdsourcing task, to teach crowd workers to avoid easy questions, raising the difficulty level of the questions they provide.

Neural symbolic reasoning DROP is designed to encourage research on methods that combine neural methods with discrete, symbolic reasoning. We present one such model in Section 6. Other related work along these lines has been done by Reed and de Freitas (2016), Neelakantan et al. (2016), and Liang et al. (2017).

3 DROP Data Collection

In this section, we describe our annotation protocol, which consists of three phases. First, we automatically extract passages from Wikipedia which are expected to be amenable to complex questions. Second, we crowdsource question-answer pairs on these passages, eliciting questions which require discrete reasoning. Finally, we validate the development and test portions of DROP to ensure their quality and report inter-annotator agreement.

Passage extraction We searched Wikipedia for passages that had a narrative sequence of events, particularly with a high proportion of numbers, as our initial pilots indicated that these passages were the easiest to ask complex questions about. We found that National Football League (NFL) game summaries and history articles were particularly promising, and we additionally sampled from any Wikipedia passage that contained at least twenty numbers.[2] This process yielded a collection of about 7,000 passages.

Question collection We used Amazon Mechanical Turk[3] to crowdsource the collection of question-answer pairs, where each question could be answered in the context of a single Wikipedia passage. In order to allow some flexibility during the annotation process, in each human intelligence task (HIT) workers were presented with a random sample of 5 of our Wikipedia passages, and were asked to produce a total of at least 12 question-answer pairs on any of these.

We presented workers with example questions from five main categories, inspired by questions from the semantic parsing literature (addition/subtraction, minimum/maximum, counting, selection and comparison; see examples in Table 1), to elicit questions that require complex linguistic understanding and discrete reasoning. In addition, to further increase the difficulty of the questions in DROP, we employed a novel adverserial annotation setting, where workers were only allowed to submit questions which a real-time QA model BiDAF *could not* solve.[4]

Next, each worker answered their own question with one of three answer types: spans of text from either question or passage, a date (which was common in history and open-domain text) and numbers, allowed only for questions which explicitly stated a specific unit of measurement (e.g., "How many *yards* did Brady run?"), in an attempt to simplify the evaluation process.

Initially, we opened our HITs to all United States workers and gradually reduced our worker pool to workers who understood the task and annotated it well. Each HIT paid 5 USD and could be completed within 30 minutes, compensating a trained worker with an average pay of 10 USD/ hour.

Statistic	**Train**	**Dev**	**Test**
Number of passages	5565	582	588
Avg. passage len [words]	213.45	191.62	195.12
Number of questions	77,409	9,536	9,622
Avg. question len [words]	10.79	11.17	11.23
Avg. questions / passage	13.91	16.38	16.36
Question vocabulary size	29,929	8,023	8,007

Table 2: Dataset statistics across the different splits.

Overall, we collected a total of 96,567 question-answer pairs with a total Mechanical Turk budget of 60k USD (including validation). The dataset was randomly partitioned by passage into training (80%), development (10%) and test (10%) sets, so all questions about a particular passage belong to only one of the splits.

Validation In order to test inter-annotator agreement and to improve the quality of evaluation against DROP, we collected at least two additional answers for each question in the development and test sets.

In a separate HIT, workers were given context passages and a previously crowdsourced question, and were asked to either answer the question or mark it as invalid (this occurred for 0.7% of the data, which we subsequently filtered out). We found that the resulting inter-annotator agreement was good and on par with other QA tasks; overall Cohen's κ was 0.74, with 0.81 for numbers, 0.62 for spans, and 0.65 for dates.

4 DROP Data Analysis

In the following, we quantitatively analyze properties of passages, questions, and answers in DROP. Different statistics of the dataset are depicted in Table 2. Notably, questions have a diverse vocabulary of around 30k different words in our training set.

Question analysis To assess the question type distribution, we sampled 350 questions from the training and development sets and manually annotated the categories of discrete operations required to answer the question. Table 1 shows the distribution of these categories in the dataset. In addition, to get a better sense of the lexical diversity of questions in the dataset, we find the most frequent

---

[2] We used an October 2018 Wikipedia dump, as well as scraping of online Wikipedia.

[3] `www.mturk.com`

[4] While BiDAF is no longer state-of-the-art, performance is reasonable and the AllenNLP implementation (Gardner et al., 2017) made it the easiest to deploy as a server.

Answer Type	Percent	Example
NUMBER	66.1	12
PERSON	12.2	Jerry Porter
OTHER	9.4	males
OTHER ENTITIES	7.3	Seahawks
VERB PHRASE	3.5	Tom arrived at Acre
DATE	1.5	3 March 1992

Table 3: Distribution of answer types in training set, according to an automatic named entity recognition.

trigram patterns in the questions per answer type. We find that the dataset offers a huge variety of linguistic constructs, with the most frequent pattern ("Which team scored") appearing in only 4% of the span type questions. For number type questions, the 5 most frequent question patterns all start with "How many", indicating the need to perform counting and other arithmetic operations. A distribution of the trigrams containing the start of the questions are shown in Figure 1.

Answer analysis To discern the level of *passage understanding* needed to answer the questions in DROP, we annotate the set of spans in the passage that are necessary for answering the 350 questions mentioned above. We find that on an average 2.18 spans need to be considered to answer a question and the average distance between these spans is 26 words, with 20% of samples needing at least 3 spans (see appendix for examples). Finally, we assess the answer distribution in Table 3, by running the part-of-speech tagger and named entity recognizer from spaCy[5] to automatically partition all the answers into various categories. We find that a majority of the answers are numerical values and proper nouns.

5 Baseline Systems

In this section we describe the initial baselines that we evaluated on the DROP dataset. We used three types of baselines: state-of-the-art semantic parsers (§5.1), state-of-the-art reading comprehension models (§5.2), and heuristics looking for annotation artifacts (§5.3). We use two evaluation metrics to compare model performance: *Exact-Match*, and a numeracy-focused (macro-averaged) F_1 score, which measures overlap between a bag-of-words representation of the gold and predicted answers. We employ the same implementation of Exact-Match accuracy as used by SQuAD, which removes articles and does other simple normalization, and our F_1 score is based on that used by SQuAD. Since DROP is numeracy-focused, we define F_1 to be 0 when there is a number mismatch between the gold and predicted answers, regardless of other word overlap. When an answer has multiple spans, we first perform a one-to-one alignment greedily based on bag-of-word overlap on the set of spans and then compute average F_1 over each span. When there are multiple annotated answers, both metrics take a max over all gold answers.

5.1 Semantic Parsing

Semantic parsing has been used to translate natural language utterances into formal executable languages (e.g., SQL) that can perform discrete operations against a structured knowledge representation, such as knowledge graphs or tabular databases (Zettlemoyer and Collins, 2005; Berant et al., 2013b; Yin and Neubig, 2017; Chen and Mooney, 2011, *inter alia*). Since many of DROP's questions require similar discrete reasoning, it is appealing to port some of the successful work in semantic parsing to the DROP dataset. Specifically, we use the grammar-constrained semantic parsing model built by Krishnamurthy et al. (2017) (KDG) for the WIKITABLEQUESTIONS tabular dataset (Pasupat and Liang, 2015).

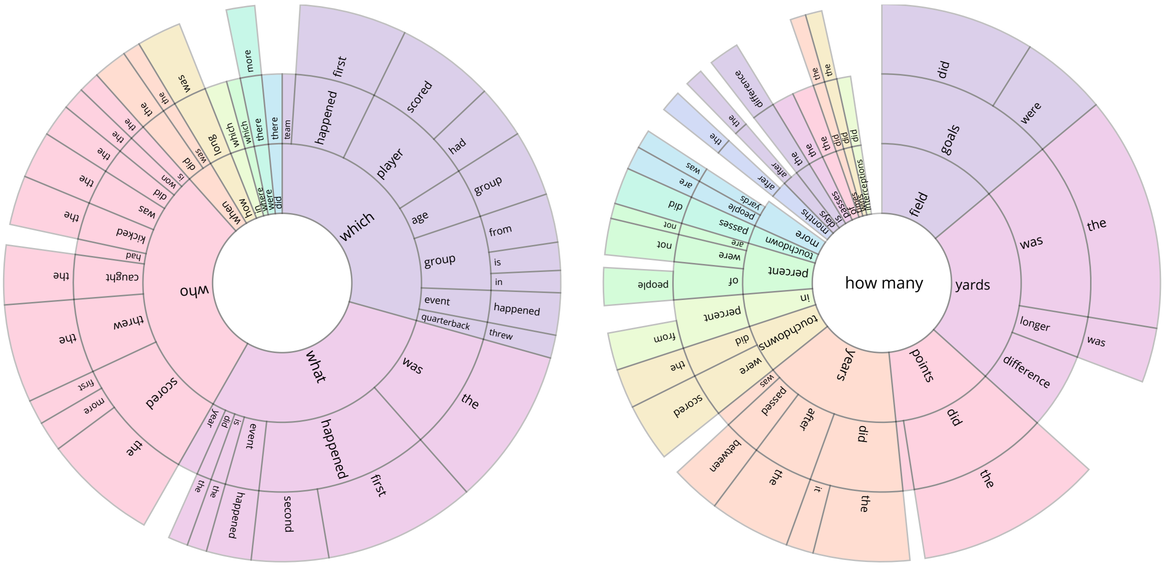
Sentence representation schemes We experimented with three paradigms to represent paragraphs as structured contexts: (1) Stanford dependencies (de Marneffe and Manning, 2008, Syn Dep); which capture word-level syntactic relations, (2) Open Information Extraction (Banko et al., 2007, Open IE), a shallow semantic representation which directly links predicates and arguments; and (3) Semantic Role Labeling (Carreras and Màrquez, 2005, SRL), which disambiguates senses for polysemous predicates and assigns predicate-specific argument roles.[6] To adhere to KDG's structured representation format, we convert each of these representations into a table, where rows are predicate-argument structures and columns correspond to different argument roles.

Logical form language Our logical form language identifies five basic elements in the table representation: *predicate-argument structures* (i.e., table rows), *relations* (column-headers), *strings*, *num-*

---

[5] https://spacy.io/

[6] We used the AllenNLP implementations of state-of-the-art models for all of these representations (Gardner et al., 2017; Dozat et al., 2017; He et al., 2017; Stanovsky et al., 2018).



(a) For span type answers

(b) For number type answers

Figure 1: Distribution of the most popular question prefixes for two different subsets of the training data.

bers, and *dates*. In addition, it defines functions that operate on these elements, such as counters and filters.[7] Following Krishnamurthy et al. (2017), we use the argument and return types of these functions to automatically induce a grammar to constrain the parser. We also add context-specific rules to produce strings occurring in both question and paragraph, and those paragraph strings that are neighbors of question tokens in the GloVe embedding space (Pennington et al., 2014), up to a cosine distance of d.[8] The complete set of functions used in our language and their induced grammar can be found in the code release.

Training and inference During training, the KDG parser maximizes the marginal likelihood of a set of (possibly spurious) question logical forms that evaluate to the correct answer. We obtain this set by performing an exhaustive search over the grammar up to a preset tree depth. At test time, we use beam search to produce the most likely logical form, which is then executed to predict an answer.

5.2 SQuAD-style Reading Comprehension

We test four different SQuAD-style reading comprehension models on DROP: (1) **BiDAF** (Seo et al., 2017), which is the adversarial baseline we used in data construction (66.8% EM on SQuAD 1.1); (2) **QANet** (Yu et al., 2018), currently the best-performing published model on SQuAD 1.1 without data augmentation or pretraining (72.7% EM); (3) **QANet + ELMo**, which enhances the QANet model by concatenating pretrained ELMo representations (Peters et al., 2018) to the original embeddings (78.7% EM); (4) **BERT** (Devlin et al., 2019), which recently achieved improvements on many NLP tasks with a novel pretraining technique (84.7% EM).[9]

These models require a few minor adaptations when training on DROP. While SQuAD provides answer indices in the passage, our dataset only provides the answer strings. To address this, we use the marginal likelihood objective function proposed by Clark and Gardner (2018), which sums over the probabilities of all the matching spans.[10] We also omitted the training questions which cannot be answered by a span in the passage (45%), and therefore cannot be represented by these systems.

For the BiDAF baseline, we use the implementation in AllenNLP but change it to use the marginal objective. For the QANet model, our settings differ from the original paper only in the batch size (16 v.s. 32) and number of blocks in the modeling layer

---

[7] For example `filter_number_greater` takes a set of predicate-argument structures, the name of a relation, and a number, and returns all those structures where the numbers in the argument specified by the relation are greater than the given number.

[8] $d = 0.3$ was manually tuned on the development set.

[9] The first three scores are based on our own implementation, while the score for BERT is based on an open-source implementation from Hugging Face: https://github.com/huggingface/pytorch-pretrained-bert

[10] For the black-box BERT model, we convert DROP to SQuAD format by using the first match as the gold span.

(6 v.s. 7) due to the GPU memory limit. We adopt the ELMo representations trained on 5.5B corpus for the QANet+ELMo baseline and the large uncased BERT model for the BERT baseline. The hyper-parameters for our NAQANet model (§6) are the same as for the QANet baseline.

5.3 Heuristic Baselines

A recent line of work (Gururangan et al., 2018; Kaushik and Lipton, 2018) has identified that popular crowdsourced NLP datasets (such as SQuAD (Rajpurkar et al., 2016) or SNLI (Bowman et al., 2015)) are prone to have artifacts and annotation biases which can be exploited by supervised algorithms that learn to pick up these artifacts as signal instead of more meaningful semantic features. We estimate artifacts by training the QANet model described in Section 5.2 on a version of DROP where either the question or the paragraph input representation vectors are zeroed out (**question-only** and **paragraph-only**, respectively). Consequently, the resulting models can then only predict answer spans from either the question or the paragraph.

In addition, we devise a baseline that estimates the answer variance in DROP. We start by counting the unigram and bigram answer frequency for each wh question-word in the train set (as the first word in the question). The **majority baseline** then predicts an answer as the set of 3 most common answer spans for the input question word (e.g., for "when", these were "quarter", "end" and "October").

6 NAQANet

DROP is designed to encourage models that combine neural reading comprehension with symbolic reasoning. None of the baselines we described in Section 5 can do this. As a preliminary attempt toward this goal, we propose a numerically-aware QANet model, NAQANet, which allows the state-of-the-art reading comprehension system to produce three new answer types: (1) spans from the question; (2) counts; (3) addition or subtraction over numbers. To predict numbers, the model first predicts whether the answer is a count or an arithmetic expression. It then predicts the specific numbers involved in the expression. This can be viewed as the neural model producing a *partially executed* logical form, leaving the final arithmetic to a symbolic system. While this model can currently only handle a very limited set of operations, we believe this is a promising approach to combining neural methods and symbolic reasoning. The model is trained by marginalizing over all execution paths that lead to the correct answer.

6.1 Model Description

Our NAQANet model follows the typical architecture of previous reading comprehension models, which is composed of embedding, encoding, passage-question attention, and output layers. We use the original QANet architecture for everything up to the output layer. This gives us a question representation $\mathbf{Q} \in \mathbb{R}^{m\times d}$, and a projected question-aware passage representation $\bar{\mathbf{P}} \in \mathbb{R}^{n\times d}$. We have four different output layers, for the four different kinds of answers the model can produce:

Passage span As in the original QANet model, to predict an answer in the passage we apply three repetitions of the QANet encoder to the passage representation $\bar{\mathbf{P}}$ and get their outputs as $\mathbf{M}_0$, $\mathbf{M}_1$, $\mathbf{M}_2$ respectively. Then the probabilities of the starting and ending positions from the passage can be computed as:

$$\mathbf{p}^{\text{p\_start}} = \text{softmax}(\text{FFN}([\mathbf{M}_0; \mathbf{M}_1]), \quad (1)$$

$$\mathbf{p}^{\text{p\_end}} = \text{softmax}(\text{FFN}([\mathbf{M}_0; \mathbf{M}_2]) \quad (2)$$

where FFN is a two-layer feed-forward network with the RELU activation.

Question span Some questions in DROP have their answer in the *question* instead of the passage. To predict an answer from the question, the model first computes a vector $\mathbf{h}^P$ that represents the information it finds in the passage:

$$\boldsymbol{\alpha}^P = \text{softmax}(\mathbf{W}^P \bar{\mathbf{P}}), \quad (3)$$

$$\mathbf{h}^P = \boldsymbol{\alpha}^P \bar{\mathbf{P}} \quad (4)$$

Then it computes the probabilities of the starting and ending positions from the question as:

$$\mathbf{p}^{\text{q\_start}} = \text{softmax}(\text{FFN}([\mathbf{Q}; \mathbf{e}^{|Q|} \otimes \mathbf{h}^P]), \quad (5)$$

$$\mathbf{p}^{\text{q\_end}} = \text{softmax}(\text{FFN}([\mathbf{Q}; \mathbf{e}^{|Q|} \otimes \mathbf{h}^P]) \quad (6)$$

where the outer product with the identity $(\mathbf{e}^{|Q|} \otimes \cdot)$ simply repeats $\mathbf{h}^P$ for each question word.

Count We model the capability of counting as a multi-class classification problem. Specifically, we consider ten numbers (0–9) in this preliminary model and the probabilities of choosing these numbers is computed based on the passage vector $\mathbf{h}^P$:

$$\mathbf{p}^{\text{count}} = \text{softmax}(\text{FFN}(\mathbf{h}^P)) \quad (7)$$

Arithmetic expression Many questions in DROP require the model to locate multiple numbers in the passage and add or subtract them to get the final answer. To model this process, we first extract all the numbers from the passage and then learn to assign a plus, minus or zero for each number. In this way, we get an arithmetic expression composed of signed numbers, which can be evaluated to give the final answer.

To do this, we first apply another QANet encoder to $\mathbf{M}_2$ and get a new passage representation $\mathbf{M}_3$. Then we select an index over the concatenation of $\mathbf{M}_0$ and $\mathbf{M}_3$, to get a representation for each number in this passage. The i^{th} number can be represented as $\mathbf{h}_i^N$ and the probabilities of this number being assigned a plus, minus or zero are:

$$\mathbf{p}_i^{\text{sign}} = \text{softmax}(\text{FFN}(\mathbf{h}_i^N)) \tag{8}$$

Answer type prediction We use a categorical variable to decide between the above four answer types, with probabilities computed as:

$$\mathbf{p}^{\text{type}} = \text{softmax}(\text{FFN}([\mathbf{h}^P, \mathbf{h}^Q])) \tag{9}$$

where $\mathbf{h}^Q$ is computed over $\mathbf{Q}$, in a similar way as we did for $\mathbf{h}^P$. At test time, we first determine this answer type greedily and then get the best answer from the selected type.

6.2 Weakly-Supervised Training

For supervision, DROP contains only the answer string, not which of the above answer types is used to arrive at the answer. To train our model, we adopt the weakly supervised training method widely used in the semantic parsing literature (Berant et al., 2013a). We find all executions that evaluate to the correct answer, including matching passage spans and question spans, correct count numbers, as well as sign assignments for numbers. Our training objective is then to maximize the marginal likelihood of these executions.[11]

7 Results and Discussion

The performance of all tested models on the DROP dataset is presented in Table 4. Most notably, all models perform significantly worse than on other prominent reading comprehension datasets, while human performance remains at similar high levels.[12] For example, BERT, the current state-of-the-art on SQuAD, *drops* by more than 50 absolute F1 points. This is a positive indication that DROP is indeed a challenging reading comprehension dataset, which opens the door for tackling new and complex reasoning problems on a large scale.

Method	**Dev**		**Test**	
	EM	F_1	EM	F_1
Heuristic Baselines				
Majority	0.09	1.38	0.07	1.44
Q-only	4.28	8.07	4.18	8.59
P-only	0.13	2.27	0.14	2.26
Semantic Parsing				
Syn Dep	9.38	11.64	8.51	10.84
OpenIE	8.80	11.31	8.53	10.77
SRL	9.28	11.72	8.98	11.45
SQuAD-style RC				
BiDAF	26.06	28.85	24.75	27.49
QANet	27.50	30.44	25.50	28.36
QANet+ELMo	27.71	30.33	27.08	29.67
BERT	30.10	33.36	29.45	32.70
NAQANet				
+ Q Span	25.94	29.17	24.98	28.18
+ Count	30.09	33.92	30.04	32.75
+ Add/Sub	43.07	45.71	40.40	42.96
Complete Model	**46.20**	**49.24**	**44.07**	**47.01**
Human	-	-	94.09	96.42

Table 4: Performance of the different models on our development and test set, in terms of Exact Match (EM), and numerically-focused F_1 (§5). Both metrics are calculated as the maximum against a set of gold answers.

The best performance is obtained by our NAQANet model. Table 6 shows that our gains are obtained on the challenging and frequent number answer type, which requires various complex types of reasoning. Future work may also try combining our model with BERT. Furthermore, we find that all heuristic baselines do poorly on our data, hopefully attesting to relatively small biases in DROP.

Difficulties of building semantic parsers We see that all the semantic parsing baselines perform quite poorly on DROP. This is mainly because of our pipeline of extracting tabular information from paragraphs, followed by the denotation-driven logical form search, can yield logical forms only for a subset of the training data. For SRL and syntactic dependency sentence representation schemes,

---

[11] Due to the exponential search space and the possible noise, we only search the addition/subtraction of two numbers. Given this limited search space, the search and marginalization are exact.

[12] Human performance was estimated by the authors collectively answering 560 questions from the test set, which were then evaluated using the same metric as learned systems. This is in contrast to holding out one gold annotation and evaluating it against the other annotations, as done in prior work, which underestimates human performance relative to systems.

Phenomenon	Passage Highlights	Question	Answer	Our model
Subtraction + Coreference	… Twenty-five of his 150 men were sick, and his advance stalled …	How many of Bartolom de Amsqueta's 150 men were not sick?	125	145
Count + Filter	… Macedonians were the largest ethnic group in Skopje, with 338,358 inhabitants … Then came … Serbs (14,298 inhabitants), Turks (8,595), Bosniaks (7,585) and Vlachs (2,557) …	How many ethnicities had less than 10000 people?	3	2
Domain knowledge	… Smith was sidelined by a torn pectoral muscle suffered during practice …	How many quarters did Smith play?	0	2
Addition	… culminating in the Battle of Vienna of 1683, which marked the start of the 15-year-long Great Turkish War …	What year did the Great Turkish War end?	1698	1668

Table 5: Representative examples from our model's error analysis. We list the identified semantic phenomenon, the relevant passage highlights, a gold question-answer pair, and the erroneous prediction by our model.

the search was able to yield logical forms for 34% of the training data, whereas with OpenIE, it was only 25%. On closer examination of a sample of 60 questions and the information extracted by the SRL scheme (the best performing of the three), we found that only 25% of the resulting tables contained information needed to the answer the questions. These observations show that high quality information extraction is a strong prerequisite for building semantic parsers for DROP. Additionally, the fact that this is a weakly supervised semantic parsing problem also makes training hard. The biggest challenge in this setup is the spuriousness of logical forms used for training, where the logical form evaluates to the correct denotation but does not actually reflect the semantics of the question. This makes it hard for the model trained on these spurious logical forms to generalize to unseen data. From the set of logical forms for a sample of 60 questions analyzed, we found that only 8 questions (13%) contained non-spurious logical forms.

Error Analysis Finally, in order to better understand the outstanding challenges in DROP, we conducted an error analysis on a random sample of 100 erroneous NAQANet predictions. The most common errors were on questions which required complex type of reasoning, such as arithmetic operations (evident in 51% of the errors), counting (30%), domain knowledge and common sense (23%), co-reference (6%), or a combination of different types of reasoning (40%). See Table 5 for examples of some of the common phenomena.

Type	(%)	Exact Match		F1	
		QN+	BERT	QN+	BERT
Date	1.57	28.7	38.7	35.5	42.8
Numbers	61.94	44.0	14.5	44.2	14.8
Single Span	31.71	58.2	64.6	64.6	70.1
$>$ 1 Spans	4.77	0	0	17.13	25.0

Table 6: Dev set performance breakdown by different answer types; our model (NAQANet, marked as *QN+*) vs. BERT, the best-performing baseline.

8 Conclusion

We have presented DROP, a dataset of complex reading comprehension questions that require **D**iscrete **R**easoning **O**ver **P**aragraphs. This dataset is substantially more challenging than existing datasets, with the best baseline achieving only 32.7% F1, while humans achieve 96%. We hope this dataset will spur research into more comprehensive analysis of paragraphs, and into methods that combine distributed representations with symbolic reasoning. We have additionally presented initial work in this direction, with a model that augments QANet with limited numerical reasoning capability, achieving 47% F1 on DROP.

Acknowledgments

We would like to thank Noah Smith, Yoav Goldberg, and Jonathan Berant for insightful discussions that informed the direction of this work. The computations on `beaker.org` were supported in part by credits from Google Cloud.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew G Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013a. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013b. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CONLL*, pages 152–164.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke S. Zettlemoyer. 2018. Quac: Question answering in context. In *EMNLP*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, abs/1810.04805.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *CoNLL Shared Task*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. of NAACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *ACL*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Mandar S. Joshi, Eunsol Choi, Daniel S. Weld, and Luke S. Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Divyansh Kaushik and Zachary Chase Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *EMNLP*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL-HLT*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *TACL*, 6:317–328.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *CFCFPE@COLING*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Pasquale Minervini and Sebastian Riedel. 2018. Adversarially regularising neural nli models to integrate logical background knowledge. In *CoNLL*.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. *ICLR*.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: a novel dataset for assessing machine comprehension using script knowledge. *LREC Proceedings, 2018*.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *ACL*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge. *TACL*.

Scott E. Reed and Nando de Freitas. 2016. Neural programmer-interpreters. *ICLR*.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *ACL*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Gabriel Stanovsky, Julian Michael, Luke S. Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL-HLT*.

Simon Šuster and Walter Daelemans. 2018. Clicr: a dataset of clinical case reports for machine reading comprehension.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL'17*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *ICLR*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. *CVPR*, abs/1811.10830.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2019. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension.

An Encoding Strategy Based Word-Character LSTM for Chinese NER

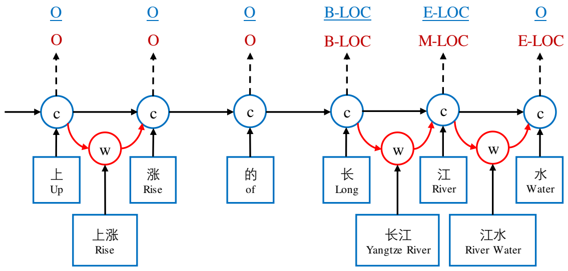
Wei Liu[1,2], **Tongge Xu**[3,1], **Qinghua Xu**[1,2], **Jiayu Song**[2], **Yueran Zu**[2]
[1]Hefei Innovation Research Institute, Beihang University
[2]School of Computer Science and Engineering, Beihang University
[3]School of Cyber Science and Technology, Beihang University
`{liuwei1206, xutg, xuqh_buaa, heavensyc, yueranzu}@buaa.edu.cn`

Abstract

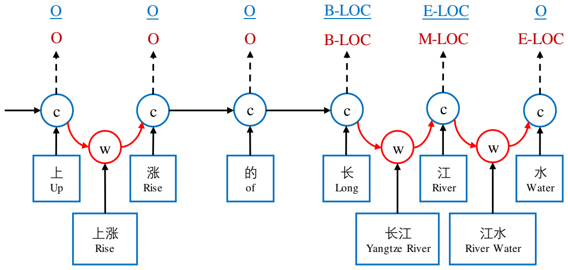
A recently proposed lattice model has demonstrated that words in character sequence can provide rich word boundary information for character-based Chinese NER model. In this model, word information is integrated into a shortcut path between the start and the end characters of the word. However, the existence of shortcut path may cause the model to degenerate into a partial word-based model, which will suffer from word segmentation errors. Furthermore, the lattice model can not be trained in batches due to its DAG structure. In this paper, we propose a novel word-character LSTM(WC-LSTM) model to add word information into the start or the end character of the word, alleviating the influence of word segmentation errors while obtaining the word boundary information. Four different strategies are explored in our model to encode word information into a fixed-sized representation for efficient batch training. Experiments on benchmark datasets show that our proposed model outperforms other state-of-the-arts models.

1 Introduction

Name Entity Recognition(NER) is a basic task of many NLP systems including Information Retrieval (Virga and Khudanpur, 2003), Relationship Extraction (Miwa and Bansal, 2016), Question Answering (Mollá et al., 2006). The main task of NER is to identify named entities such as person, location, organization, etc. in given text. Various methods have been proposed to tackle this problem, including Hidden Markov Models(HMMs) (Saito and Nagata, 2003), Maximum Entropy Models(ME) (Chieu and Ng, 2003), Support Vector Machines(SVM) (Ekbal and Bandyopadhyay, 2010) and Conditional Random Fields(CRF) (Feng et al., 2006). With the development of deep learning, neural networks (Huang et al., 2015; Lample et al., 2016; Habibi et al., 2017) have been introduced to NER task. To avoid the segmentation errors, most of neural Chinese NER models are character-based.



(a) Original lattice model



(b) Degraded lattice model in extreme cases

Figure 1: An example of the lattice model degenerates into a partial word-based model. Due to the shortcut path ”江(River)” → ”江水(River Water)” → ”水(Water)”, the model incorrectly predicts that ”江(River)” and ”水(Water)” belong to the same entity. Red labels(without underline) denote predicted labels, and blue labels(with underline) denote gold labels.

Although character-based method has achieved good performance, it does not exploit word information in character sequence. Entity boundaries usually coincide with some word boundaries, which suggests that words in character sequence can provide rich boundary information for character-based model. To integrate words information into character-based model, Zhang and Yang (2018) propose a lattice-structured LSTM

Proceedings of NAACL-HLT 2019, pages 2379–2389
Minneapolis, Minnesota, June 2 - June 7, 2019.

model to encode a sequence of input characters as well as all potential words that match a lexicon. Their model is an extension of character-based LSTM-CRF model and uses extra "shortcut paths" to link the memory cell between the start and the end characters of a word for utilizing word information. And the gated recurrent unit is used to control the contribution of shortcut paths and path between adjacent characters. However, as the study of (Yang et al., 2018) shown, the gate mechanism fails to choose the right path sometimes. As shown in Figure 1, wrong choices may cause lattice model to degenerate into a partial word-based model, which suffers from word segmentation errors. In addition, due to the variable length of words, the length of the whole path is not fixed. Besides, each character is bounded with a variable-sized candidate word sets, which means the amount of incoming and outcoming paths is not fixed either. In this case, lattice LSTM model is deprived of the power of batch training, and hence it is highly inefficient.

To address the above problems, we propose a novel word-character LSTM(WC-LSTM) to integrate word information into character-based model. To prevent our model from degenerating into a partial word-based model, we assign word information to a single character and ensure that there are no shortcut paths between characters. Specifically, word information is assigned to its end character and start character in forward WC-LSTM and backward WC-LSTM respectively. We introduce four strategies to extract fixed-sized useful information from different words, which ensures that our proposed model can perform batch training without losing word information.

We demonstrate the effectiveness of our architecture on four widely used datasets. Experimental results show that our proposed model outperforms other state-of-the-art models on the four datasets.

Our contributions of this paper can be concluded as follows:

- We propose a novel word-character LSTM(WC-LSTM) to incorporate word information into character-based model.

- We explore four different strategies to encode word information into a fixed-sized vector, which enables our proposed model to be trained in batches and adapted to various application scenarios.

- Our proposed model outperforms other models and achieves new state-of-the-art over four Chinese NER datasets. We release the source code for further research[1].

2 Related Work

Neural Networks have been shown to achieve impressive results on Name Entity Recognition task (Gregoric et al., 2018; Lin and Lu, 2018). Based on the level of granularity, most of the models can be divided into three categories: word-based models, character-based models, and hybrid models.

Word-Based Models. Collobert and Weston (2008) propose one of the first word-based models for NER, with feature constructed from orthographic features, dictionaries and lexicons (Yadav and Bethard, 2018). Collobert et al. (2011) replace the hand-crafted features with word embeddings. Huang et al. (2015) propose a BiLSTM-CRF model for NER and achieves good performance. Ma and Hovy (2016) and Chiu and Nichols (2016) use CNN to capture spelling characteristics and Lample et al. (2016) use LSTM instead. When applied to Chinese NER, the above models all suffer from segmentation errors, since Chinese word segmentation is compulsory for those models.

Character-Based Models. Peng and Dredze (2015) propose to add segmentation features for better recognition of entity boundary. Dong et al. (2016) integrate radical-level features into character-based model. To eliminate the ambiguity of character, Sun and He (2017) take the position of character into account. Although the above models have achieved good results, they all ignore word information in character sequence.

Hybrid Models. Some efforts have been made to integrate word boundary information into character-based models. Motivated by the success of multi-task learning for Natural Language Processing (Liu et al., 2016, 2017; Zhang et al., 2018), Peng and Dredze (2016) first proposed to jointly train Chinese NER with Chinese word segmentation(CWS) task. Cao et al. (2018) apply adversarial transfer learning framework to integrate the task-shared word boundary information into Chinese NER task. Another way to obtain word boundary information is proposed by (Zhang and Yang, 2018), using a lattice LSTM to integrate word information into character-based model, which is similar to what is proposed in

---

[1] https://github.com/liuwei1206/CCW-NER

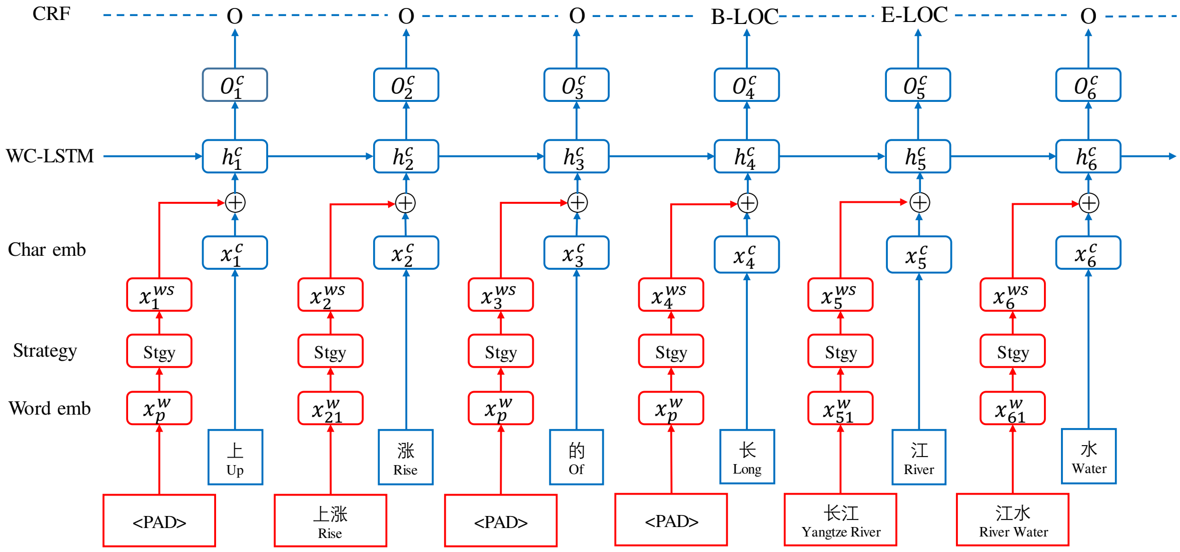


Figure 2: The architecture of our unidirectional model. The blue part can be seen as a standard character-based model but with a word-character LSTM(WC-LSTM), and the red part indicates the process of encoding word information into a fixed-size representation. Word information is integrated into the end character of the word. Where "<PAD>" denotes padding value; "Stgy" denotes a certain encoding strategy and ⊕ denotes concatenation operation.

this paper. The main differences are as follows. Firstly, they exploit word information by a DAG-structured LSTM, while we use a chain-structured LSTM. Secondly, instead of integrating to the hidden state of LSTM, our model add word information into the input vector. Finally, our model can be trained in batches and is more efficient.

3 Method

The architecture of our proposed model is shown in Figure 2. Same as the widely used neural Chinese NER model, we use LSTM-CRF as our main network structure. The differences between our model and a standard LSTM-CRF model are mainly on the embedding layer and LSTM and can be summarized as follows. First, we represent a Chinese sentence as a sequence of character-words pairs to integrate word information into each character. Second, to enable our model to train in batches and to meet different application requirements, we introduce four encoding strategies to extract fixed-sized but different information from words. Finally, a chain-structured word-character LSTM is used to extract features from both character and word for better predicting.

Next, we will explain the main ideas for each component, including word-character embedding layer, word encoding strategy, and word-character LSTM.

Formally, we denote a Chinese sentence as $\mathbf{s} = \{c_1, c_2, ..., c_n\}$, where c_i denotes the i_{th} character. We use $c_{b,e}$ to denote a character subsequence in $\mathbf{s}$, which begins with b_{th} character and ends with e_{th} character. Take the sentence in Figure 2 for example, $c_{1,2}$ is "上涨(Rise)". We use $\overrightarrow{ws_i}$ to denote words assigned to i_{th} character in forward WC-LSTM, which are a set of character subsequences $c_{b,i}$, where $b < i$ and $c_{b,i}$ matches a word in lexicon $\mathbf{D}$. The lexicon $\mathbf{D}$ is the same as the one used in (Zhang and Yang, 2018), which is built by using automatically segmented large raw text. Similarly, we use $\overleftarrow{ws_i}$ to denote the words for i_{th} character in backward WC-LSTM, which are a set of character subsequences $c_{i,e}$, where $e > i$ and $c_{i,e}$ matches a word in lexicon $\mathbf{D}$. Finally, the sentence $\mathbf{s}$ is represented as $\overrightarrow{\mathbf{rs}} = \{(c_1, \overrightarrow{ws_1}), (c_2, \overrightarrow{ws_2}), ..., (c_n, \overrightarrow{ws_n})\}$ in our model, and its reverse representation is $\overleftarrow{\mathbf{rs}} = \{(c_n, \overleftarrow{ws_n}), (c_{n-1}, \overleftarrow{ws_{n-1}}), ..., (c_1, \overleftarrow{ws_1})\}$.

3.1 Word-Character Embedding Layer

In our model, Each position i in $\overrightarrow{\mathbf{rs}}$ consists of two parts: i_{th} character c_i and the assigned words $\overrightarrow{ws_i}$. The origin number of words in $\overrightarrow{ws_i}$ is s_i^t, and words are sorted by their length. We ensure each $\overrightarrow{ws_i}$ has the same number s_i^p [2] in the whole batch by padding. We embed each character c_i in dis-

---

[2] The number depends on the maximum s_i^t in the whole batch, and it can not be less than 1.

tributional space as $\mathbf{x}_i^c$:

$$\mathbf{x}_i^c = \mathbf{e}^c(c_i) \tag{1}$$

where $\mathbf{e}^c$ denotes a pre-trained character embedding lookup table. Similarly, for each $\overrightarrow{ws_i} = \{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^p}}\}$, the l_{th} word $\overrightarrow{w_{il}}$ in $\overrightarrow{ws_i}$ is represented using

$$\mathbf{x}_{il}^{\overrightarrow{w}} = \mathbf{e}^w(\overrightarrow{w_{il}}) \tag{2}$$

where $\mathbf{e}^w$ denotes a pre-trained word embedding lookup table. As a result, the distributional representation of words $\overrightarrow{ws_i}$ is $\{\mathbf{x}_{i1}^{\overrightarrow{w}}, ..., \mathbf{x}_{is_i^p}^{\overrightarrow{w}}\}$.

3.2 Words Encoding Strategy

Although the number of assigned words s_i^p for each character c_i is same in one batch, the number varies from batch to batch. As a result, the size of input to the model is not fixed, which is not conducive to batch training. To acquire fixed-sized input, we introduce four different encoding strategies in this section. And we use $\mathbf{x}_i^{\overrightarrow{ws}}$ to denote the final representation of word information for position i in following sections.

Shortest Word First: For each word set $\overrightarrow{ws_i} = \{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^p}}\}$, we simply select word whose length is the shortest, i.e. $\overrightarrow{w_{i1}}$. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \mathbf{x}_{i1}^{\overrightarrow{w}} \tag{3}$$

Longest Word First: Contrary to the shortest word first, we select word whose length is the longest, i.e. $\overrightarrow{w_{is_i^t}}$. Note that s_i^t may be 0, in this case, we set it to 1. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \mathbf{x}_{is_i^t}^{\overrightarrow{w}} \tag{4}$$

Average: While the first two strategies can only use the information of partial words, we introduce an average strategy to utilize all word information. As its name indicates, the average strategy computes the centroid of the embeddings of all elements except paddings in word set , i.e. $\{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^t}}\}$. If $s_i^t = 0$, we simply average all the padding value in the word set. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \begin{cases} \frac{1}{s_i^t}\sum_{l=1}^{s_i^t} \mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t > 0 \\ \frac{1}{s_i^p}\sum_{l=1}^{s_i^p} \mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t = 0 \end{cases} \tag{5}$$

Self-Attention: Inspired by self-attention mechanism applied to sentence embedding (Lin et al., 2017), we exploit self-attention to better capture useful information from assigned words. For simplicity, we denote all the $\mathbf{x}_{il}^{\overrightarrow{w}}$ as W_i, which has the size s_i^p-by-d^w, where d^w denotes the dimensionality of word embedding $\mathbf{e}^w$.

$$W_i = (\mathbf{x}_{i1}^{\overrightarrow{w}}, ..., \mathbf{x}_{is_i^p}^{\overrightarrow{w}}) \tag{6}$$

We use self-attention mechanism to obtain a linear combination of s_i^p word embeddings in W_i. The attention mechanism takes W_i as input, and generates a weight vector $\mathbf{a}_i$.

$$\mathbf{a}_i = softmax(\mathbf{w}_2 tanh(W_1 W_i^T)) \tag{7}$$

W_1 is a weight matrix with the size of d_a-by-d^w and $\mathbf{w}_2$ is a d_a dimensional vector, where d_a is a hyperparameter. Both of them are trainable parameters.

If $s_i^t > 0$, we use the mask to exclude the padding values; otherwise we reserve them. Finally, we use $\mathbf{a}_i$ to get the weighted sum of all words.

$$\mathbf{x}_i^{\overrightarrow{ws}} = \begin{cases} \sum_{l=1}^{s_i^t} a_{il}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t > 0 \\ \sum_{l=1}^{s_i^p} a_{il}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t = 0 \end{cases} \tag{8}$$

where a_{il} denotes the l_{th} value in $\mathbf{a}_i$.

3.3 Word-Character LSTM(WC-LSTM)

Inspired by the way character bigram is integrated into sequence labeling model (Chen et al., 2015; Yang et al., 2017), we concatenate each $\mathbf{x}_i^c$ with $\mathbf{x}_i^{\overrightarrow{ws}}$ to utilize word information. And this is quite different from the way used in (Zhang and Yang, 2018), since they use extra shortcut paths to integrate word information into the hidden layer of LSTM. By concatenating, there is no shortcut path in our model and information can only flow between adjacent characters, which ensures that our model will not degenerate into a partial word-based model. Then the WC-LSTM functions are:

$$\begin{bmatrix} \tilde{\mathbf{c}}_i \\ \mathbf{o}_i \\ \mathbf{i}_i \\ \mathbf{f}_i \end{bmatrix} = \begin{bmatrix} tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_p \begin{bmatrix} \mathbf{x}_i \\ \mathbf{h}_{i-1} \end{bmatrix} + \mathbf{b}_p \right) \tag{9}$$

$$\begin{aligned} \mathbf{x}_i &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overrightarrow{ws}} \\ \mathbf{c}_i &= \tilde{\mathbf{c}_i} \odot \mathbf{i}_i + \mathbf{c}_{i-1} \odot \mathbf{f}_i \\ \mathbf{h}_i &= \mathbf{o}_i \odot tanh(\mathbf{c}_i) \end{aligned} \tag{10}$$

where $\mathbf{o}_i$, $\mathbf{i}_i$ and $\mathbf{f}_i$ denote output gate, input gate and forget gate respectively. $\mathbf{W}_p$ and $\mathbf{b}_p$ are parameters of affine transformation; σ denotes the logistic sigmoid function; $\oplus$ denotes concatenation operation and $\odot$ denotes elementwise multiplication.

The bidirectional WC-LSTM is applied in our model to leverage both information from the past and the future. To get the future information, we use a second WC-LSTM that reads the reverse representation of $\overrightarrow{\mathbf{rs}}$, i.e., $\overleftarrow{\mathbf{rs}} = \{(c_n, \overleftarrow{ws_n}), (c_{n-1}, \overleftarrow{ws_{n-1}}), ..., (c_1, \overleftarrow{ws_1})\}$. And the following operations to get each backward WC-LSTM hidden vector $\overleftarrow{\mathbf{h}_i}$ is the same as the one in the forward WC-LSTM. Finally, the update of each bidirectional WC-LSTM unit can be written as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{x}_i} &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overrightarrow{ws}} \\
\overleftarrow{\mathbf{x}_i} &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overleftarrow{ws}} \\
\overrightarrow{\mathbf{h}_i} &= \overrightarrow{\mathrm{WC-LSTM}}(\overrightarrow{\mathbf{h}_{i-1}}, \overrightarrow{\mathbf{x}_i}) \\
\overleftarrow{\mathbf{h}_i} &= \overleftarrow{\mathrm{WC-LSTM}}(\overleftarrow{\mathbf{h}_{i+1}}, \overleftarrow{\mathbf{x}_i}) \\
\mathbf{h}_i &= \overrightarrow{\mathbf{h}_i} \oplus \overleftarrow{\mathbf{h}_i}
\end{aligned} \tag{11}$$

where $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$ are hidden states at position i of forward and backward WC-LSTM respectively, and $\oplus$ denotes concatenation operation.

3.4 Decoding and Training

Considering the dependencies between successive labels, we use a CRF layer to make sequence tagging. We define matrix $\mathbf{O}$ to be scores calculated based on the output $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$:

$$\mathbf{O} = \mathbf{W}_o\mathbf{H} + \mathbf{b}_o \tag{12}$$

For a label sequence $\mathbf{y} = \{y_1, y_2, ..., y_n\}$, we define its probability to be:

$$p(\mathbf{y}|\mathbf{s}) = \frac{exp\left(\sum_i \left(\mathbf{O}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}\right)\right)}{\sum_{\tilde{\mathbf{y}}} exp\left(\sum_i \left(\mathbf{O}_{i,\tilde{y}_i} + \mathbf{T}_{\tilde{y}_{i-1},\tilde{y}_i}\right)\right)} \tag{13}$$

Where $\mathbf{W_o}$ and $\mathbf{b_o}$ are paramters to calculate $\mathbf{O}$; $\mathbf{T}$ is a transition score matrix and $\tilde{\mathbf{y}}$ denotes all possible tag sequences.

While decoding, we use the Viterbi algorithm to find the label sequences that obtained the highest score:

$$\mathbf{y}^* = arg \max_{\mathbf{y} \in \tilde{\mathbf{y}}} \sum_i \left(\mathbf{O}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}\right) \tag{14}$$

Given N manually labeled data $\{(\mathbf{s}_j, \mathbf{y}_j)\}|_{j=1}^N$, we minimize the sentence-level negative log-likelihood loss to train the model:

$$L = -\sum_j log(p(\mathbf{y}_j|\mathbf{s}_j)) \tag{15}$$

Dataset	Train sent	Dev sent	Test sent
OntoNotes	15724	4301	4346
MSRA	46364	-	4365
Weibo NER	1350	270	270
Chinese resume	3821	463	477

Table 1: Statistics of the datasets

4 Experiments

4.1 Experimental Settings

Dataset. We evaluate our model on four datasets, including OntoNotes4 (Weischedel et al., 2011), MSRA (Levow, 2006), Weibo NER (Peng and Dredze, 2015) and a Chinese resume dataset (Zhang and Yang, 2018). Both OntoNotes4 and MSRA datasets are news in simplified Chinese. Weibo NER dataset is social media data, which is drawn from the Sina Weibo. Chinese resume dataset consists of resumes of senior executives, which is annotated by (Zhang and Yang, 2018). For OntoNotes, we use the same training, development and test splits as (Che et al., 2013). For other datasets which have already been split, and we don't change them. We summarize the datasets in Table 1.

Implementation Details. We utilize the character and word embeddings used in (Zhang and Yang, 2018), both of which are pre-trained on Chinese Giga-Word using word2vec model. Following (Zhang and Yang, 2018), we use the word embedding dictionary as Lexicon $\mathbf{D}$ in our model. For characters and words that do not appear in the pretrained embeddings, we initialize them with a uniform distribution[3]. When training the model, character embeddings and word embeddings are updated along with other parameters.

For hyper-parameter configurations, we mostly refer to the settings in (Zhang and Yang, 2018). We set both character embedding size and word embedding size to 50. The dimensionality of each unidirectional multi-input LSTM hidden states is 100 for Weibo NER and Chinese Resume, and 200 for OntoNote 4 and MSRA. For self-attention strategy, we set the d_a to 50. To avoid overfitting, we apply dropout to both embeddings and LSTM with a rate of 0.5. We use SGD to optimize all the trainable parameters. Learning rate is set to 0.015 initially and decays during training at a rate

---

[3] The range is $\left[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}\right]$, where dim demotes the size of embedding.

Input	Models	P	R	F1
Gold seg	Wang et al. (2013)	76.43	72.32	74.32
	Che et al. (2013)	**77.71**	72.51	**75.02**
	Yang et al. (2016)	65.59	71.84	68.57
No seg	Lattice (Zhang and Yang, 2018)	76.35	71.56	73.88
	Character baseline	70.08	60.53	64.95
	WC-LSTM + shortest	**76.39**	72.39	**74.34**
	WC-LSTM + longest	75.62	72.76	**74.16**
	WC-LSTM + average	76.04	72.03	**73.98**
	WC-LSTM + self-attention	76.09	**72.85**	**74.43**

Table 2: Results on OntoNotes

of 0.05.

For evaluation, we use the Precision(P), Recall(R) and F1 score as metrics in our experiments.

4.2 Experimental Results

OntoNotes. Table 2 shows the experimental results on OntoNote 4 dataset. The ”Input” column shows the representation of input sentence, where ”Gold seg” means a sequence of words with gold-standard segmentation, and ”No seg” means a sequence of character without any segmentation.

The first block in Table 2 are the results of word-based models (Wang et al., 2013; Che et al., 2013; Yang et al., 2016). By using gold-standard segmentation and external labeled data, all of them achieve good performance. But the only resource used in our model are pretrained character and word embeddings.

The first two rows in the second block show the performance of the lattice model and character-based model. The character baseline denotes the original character-based BiLSTM-CRF model. Zhang and Yang (2018) propose a lattice LSTM to exploit word information in character sequence, giving the F1 score of 73.88%. Compared with the character baseline, lattice model gains 8.92% improvement in F1 score, which shows the importance of word information in character sequence.

In the last four rows, we list the results of our proposed model. The results show that all of our models outperform other character-based models, and the one with self-attention strategy achieves the best result. Without gold-standard segmentation and external labeled data, our model gives competitive results to the word-based models on this dataset. Compared with the character baseline, our model with self-attention obtains 9.48% improvement in F1 score, which proves the effectiveness of our way to integrating word information. Compared with lattice model, all of our models achieve better results, which shows that our approach to integrating word information is more reasonable than lattice model.

Models	P	R	F1
Zhang et al. (2006)	92.20	90.18	91.18
Zhou et al. (2013)	91.86	88.75	90.28
Dong et al. (2016)	91.28	90.62	90.95
Cao et al. (2018)	91.73	89.58	90.64
Lattice (Zhang and Yang, 2018)	93.57	92.79	93.18
Character baseline	89.61	86.98	88.37
WC-LSTM + shortest	93.97	92.59	**93.28**
WC-LSTM + longest	94.33	**93.11**	**93.71**
WC-LSTM + average	**94.58**	92.91	**93.74**
WC-LSTM + self-attention	94.36	92.38	**93.36**

Table 3: Results on MSRA

Models	NE	NM	Overall
Peng and Dredze (2015)	51.96	61.05	56.05
Peng and Dredze (2016)	**55.28**	62.97	58.99
Sun and He (2017)	54.50	62.17	58.23
He and Sun (2017)	50.60	59.32	54.82
Cao et al. (2018)	54.34	57.35	58.70
Lattice (Zhang and Yang, 2018)	53.04	62.25	58.79
Character baseline	47.98	57.94	52.88
WC-LSTM + shortest	52.99	65.75	**59.20**
WC-LSTM + longest	52.55	**67.41**	**59.84**
WC-LSTM + average	**53.19**	64.17	58.67
WC-LSTM + self-attention	49.86	65.31	57.51

Table 4: Results on Weibo NER

MSRA. Table 3 shows the results on MSRA dataset. Zhang et al. (2006) and Zhou et al. (2013) use the statistical model with rich hand-crafted features. Dong et al. (2016) exploit radical features in Chinese character. Cao et al. (2018) joint train Chinese NER task with Chinese word segmentation, in which adversarial learning and self-attention mechanism are applied for better performance. We can observe that our proposed models outperformance the above models and the one with average strategy achieves new state-of-the-art performance.

Weibo. Table 4 shows the results[4] on Weibo dataset. The ”NE”, ”NM” and ”Overall” columns denote F1-score for named entities, nominal entities(excluding named entities) and both respectively. We can see that WC-LSTM model with longest word first strategy achieves new state-of-the-art performance. Multi-task learning (Peng and Dredze, 2015, 2016; Cao et al., 2018) and semi-supervised learning (Sun and He, 2017; He and Sun, 2017) are the most common methods

---

[4] The results of (Peng and Dredze, 2015, 2016) are taken from (Peng and Dredze, 2017)

Models	**P**	**R**	**F1**
Lattice (Zhang and Yang, 2018)	94.81	94.11	94.46
Character baseline	93.26	93.44	93.35
WC-LSTM + shortest	94.97	94.91	**94.94**
WC-LSTM + longest	**95.27**	**95.15**	**95.21**
WC-LSTM + average	95.09	94.97	**95.03**
WC-LSTM + self-attention	95.14	94.79	**94.96**

Table 5: Results on Chinese Resume

	Time(s)/epoch
Character baseline (*batch_size*=1)	880
Character baseline (*batch_size*=8)	253
Lattice	2245
WC-LSTM (*batch_size*=1)	980
WC-LSTM (*batch_size*=8)	350

Table 6: Time per epoch of models

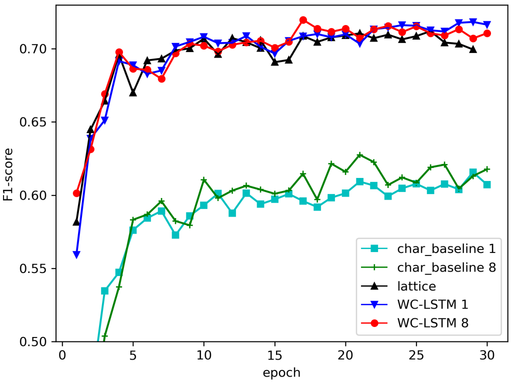


Figure 3: Convergence curve of models. Our model can converge within the same epochs as lattice model does. ”1” and ”8” denotes batch_size. Lattice model can only be trained with batch_size=1 due to its DAG structure.

for Weibo NER task due to the small amount of training data. All of the above models require additional cross-domain or semi-supervised data. Compared with those models, our model does not need additional labeled data; we only exploit pre-trained character and word embeddings.

Resume. Table 5 shows the results on Chinese Resume dataset. Consistent with the previous results, our models outperform lattice model (Zhang and Yang, 2018). The above experimental results strongly verify that our method to utilize word information is more effective than the lattice model.

Our proposed model has achieved state-of-the-art results on **various domains** such as news, social media, and Chinese resume.

4.3 Efficiency

To further explore the efficiency of our model, we conduct some comparative experiments on training time and convergence speed. The lattice model proposed in (Zhang and Yang, 2018) is our principal comparison object, since it also utilizes the word information in character sequence. Our model is an extension of the character-based model, so we also report the results on character-based model as character baseline. We only conduct our experiments on OnteNotes dataset due to space limitation. And we choose the model with the self-attention strategy for the comparative experiments, as it outperforms other strategies on OntoNotes dataset.

The training time of each epoch for all models is shown in Table 6. The lattice model needs the most training time for each epoch, since it can only be trained with batch_size=1 due to its complex DAG structure. Compared with it, our model with batch_size=1 only need half of the training time. Which shows that our model is more efficient. With batch_size=8, our model is nearly 6 times faster than the lattice model, which further demonstrates the efficiency of our model. Compared with the character baseline, our model only adds a small amount of training time but greatly improves the performance. All the experiments are conducted on a single GPU with NVIDIA Tesla K40m.

Figure 3 shows the learning curve of the models in Table 6. As we can see from the figure, whether with batch_size=1 or 8, our model can converge within the same epochs as lattice model does. But compared with the lattice model, our model with batch_size=8 only takes about 1/7 of their training time per epoch. Besides, we can observe from Figure 3, both our model and lattice model significantly outperform the character baseline, which shows the importance of the word information again.

4.4 Detailed Analysis

Case Study. Word information is very useful for Chinese NER task, since it can provide rich word boundary information. To verify that our model can better utilize the boundary information, we analyze an example from OntoNotes dataset. As shown in Table 7, the character-based model cannot detect the existence of the entity ”东北亚(Northeast Asia)” without word information. The lattice model incorrectly recognizes ”东北亚

Sentence (truncated)	新的 New		东北亚 Northeast Asian			大陆桥 Continental Bridge		
Latent words	东北, 东北亚, 北亚, 亚大, 亚大陆, 大陆, 大陆桥, 陆桥 Northest, Northeast Asia, North Asia, Second largest, Subcontinent, Continent, Continental bridge, Land bridge							
Gold labels	新 O	的 O	东 B-LOC	北 M-LOC	亚 E-LOC	大 O	陆 O	桥 O
Character	新 O	的 O	东 O	北 O	亚 O	大 O	陆 O	桥 O
Lattice	新 O	的 O	东 B-LOC	北 M-LOC	亚 M-LOC	大 M-LOC	陆 M-LOC	桥 E-LOC
Shortest	新 O	的 O	东 B-LOC	北 M-LOC	亚 E-LOC	大 O	陆 O	桥 O
Longest	新 O	的 O	东 B-LOC	北 M-LOC	亚 E-LOC	大 O	陆 O	桥 O
Average	新 O	的 O	东 B-LOC	北 M-LOC	亚 E-LOC	大 O	陆 O	桥 O
Self-attention	新 O	的 O	东 B-LOC	北 M-LOC	亚 E-LOC	大 O	陆 O	桥 O

Table 7: An example of that our models can mitigate the influence of wrong boundary information while utilizing word information. ”Latent words” denotes all words in character sequences; ”Character” denotes the character-based model; ”Lattice” denotes lattice model and the last four rows are our models with different encoding strategies.

大陆桥(Northeast Asian Continental Bridge)” as an entity, which is caused by the wrong selection of paths. Different from the lattice model, our models are not disturbed by the wrong boundary information and make the correct predictions.

Strategies Analysis. In this part, we analyze the difference between strategies. The application scenarios of shortest word first and longest word first can be explained by Nested Name Entity Recognition (Ju et al., 2018; Sohrab and Miwa, 2018). Short word first is good at identifying inner nested entities due to the short word information, while longest word first tends to identify flat entities with the help of long word information. Taking ”长江三角洲(Yangtze River Delta)” as an example, shortest word first recognizes ”长江(Yangtze)” and ”三角洲(Delta)” as entities, but longest word first tend to think that they are part of the entity ”长江三角洲(Yangtze River Delta)”. Both results are reasonable, but the right result depends on specific needs.

The average and self-attention strategies are the combination of all words information and can use more information. Intuitively, they should outperform the shortest word first and the longest word first. But results on Weibo NER(Table 4) and Resume(Table 5) show the opposite effect. We conjecture that this is caused by the small amount of training data since more word information but small dataset will lead to overfitting. The average strategy is a special case of the self-attention strategy where all weights are the same, so we would like to see the latter outperforms the former when training data is sufficient. Surprisingly, the average strategy achieves higher F1 score than the self-attention strategy in MSRA dataset(Table 3). We carefully analyze the experimental results and find that there are a large number of informal texts in the MSRA test set. Specifically, the MSRA test set contains some very long sentences, in which there are a series of Chinese person name without delimiter. As shown in Table 8, when applied to such informal text, the self-attention strategy fails to determine the entity boundary sometimes while the average strategy correctly recognizes the entities. And we conjecture that, with more trainable parameters, the self-attention strategy can better fit the formal text in the training set but cannot adapt well to the informal data in the test set, so it performs worse than the average strategy.

Sentence (truncated)	房维中经叔平 Fang Weizhong and Jing Shuping					
Latent words	房维, 中经, 经叔平, 经叔, 叔平 Fang Wei, Zhongjing, Jing Shuping, Jingshu, Shuping					
Gold labels	房 B-PER	维 M-PER	中 E-PER	经 B-PER	叔 M-PER	平 E-PER
Average	房 B-PER	维 M-PER	中 E-PER	经 B-PER	叔 M-PER	平 E-PER
Self-attention	房 B-PER	维 M-PER	中 M-PER	经 E-PER	叔 B-PER	平 E-PER

Table 8: An example of our model applied to informal text using the average strategy and self-attention strategy. ”Latent words” denotes all words in character sequences.

Finally, the application scenarios of different strategies can be summarized as followings. If the training data is sufficient, we recommend using self-attention for formal texts and average strategy for informal texts. If there is only a very small amount of annotated data, we recommend using the shortest words first for inner nested entities and longest word first strategy for flat entities.

Lexicon and Embeddings. To further analyze the contribution from word lexicon and pretrained word embeddings, we conduct some comparative experiments by using the same word lexicon with and without pretrained embeddings. We choose the strategy that achieving the best performance for each dataset. We estimate the contribution of the lexicon by replacing pretrained word embeddings with randomly initialized embeddings[5]. As shown in Table 9, both lexicon

---

[5] Same initialization strategy as in ”Implement Details”.

	OntoNotes	**Resume**	**MSRA**	**Weibo**
Character baseline	64.95	93.35	88.37	52.88
WC-LSTM + init	67.81(+2.86)	94.51(+1.16)	90.68(+2.31)	54.94(+2.06)
WC-LSTM + pretrain	74.43(+9.48)	95.21(+1.86)	93.74(+5.37)	59.84(+6.96)

Table 9: Comparison F1 scores between our proposed model with and without pretrained word embeddings. Where ”init” and ”pretrain” denote without and with pretrained embeddings respectively. ”+” denotes the boost value to baseline.

and pretrained word embeddings are useful to our model. However, different from the result in lattice model(Yang et al., 2018), pretrained word embeddings contribute more than lexicon to our model. Taking the result on Ontonote for example, the contribution of pretrained embeddings can be estimated as $(9.48\% - 2.86\%) = 6.62\%$, which is higher than the contribution of lexicon 2.86%. The results show that our model relies more on pretrained embeddings instead of the lexicon, which explains the excellent performance of our model in different domains.

5 Conclusion and Future Work

In this paper, we propose a novel method to utilize word information in character sequence for Chinese NER. Four encoding strategies are introduced to extract fixed-sized but different information for batch training. By using WC-LSTM to extract features from the character vector and word vector, our model can effectively exploit word boundary information and mitigate the influence of word segmentation errors. Experiments on datasets in different domains show that our model is more efficient and faster than the lattice model and also outperforms other state-of-the-art models.

In the future, we plan to further improve and perfect the proposed method, such as exploring some strategies to handle OOV words. Also, the proposed methods can be further extended to other Chinese NLP tasks, such as CWS, Text Classification, and Sentiment Analysis.

Acknowledgments

The research work is supported by the No. BHKX-17-07 project of Hefei Innovation Research Institute, Beihang University. We would like to thank Jie Yang for his open-source toolkit ”NCRF++” (Yang and Zhang, 2018). Moreover, we sincerely thank all anonymous reviewers for their valuable comments.

References

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 182–192.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 52–62.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1197–1206.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 160–163.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for chinese named entity recognition. In *In Natural Language Understanding and Intelligent Applications*, pages 239–250.

Asif Ekbal and Sivaji Bandyopadhyay. 2010. Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical and Electronics Engineering*, 4(2):155–170.

Yuanyong Feng, Le Sun, and Yuanhua Lv. 2006. Chinese word segmentation and named entity recognition based on conditional random fields models. In

Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pages 181–184.

Andrej Zukov Gregoric, Yoram Bachrach, and Sam Coope. 2018. Named entity recognition with parallel recurrent neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 69–74.

Maryam Habibi, Leon Weber, Mariana L. Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i38.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3216–3222.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Computing Research Repository*, arXiv:1508.01991.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1446–1459.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–27.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22-23, 2006*, pages 108–117.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2012–2022.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations 2017*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 118–127.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116.

Diego Mollá, Menno van Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, pages 51–58.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Nanyun Peng and Mark Dredze. 2017. Supplementary results for named entity recognition on chinese social media with an updated dataset. Technical report.

Kuniko Saito and Masaaki Nagata. 2003. Multi-language named-entity recognition system based on hmm. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2843–2849.

Xu Sun and Hangfeng He. 2017. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, pages 713–718.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Ann Taylor Nianwen Xue, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. Ontonotes release 4.0 ldc2011t03. dvd. In *Linguistic Data Consortium*.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158. Association for Computational Linguistics.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part I*, pages 140–154.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 839–849.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice LSTM for chinese word segmentation. *Computing Research Repository*, abs/1810.12594.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018. Neural networks incorporating dictionaries for chinese word segmentation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5682–5689. Springer Berlin Heidelberg.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for SIGHAN bakeoff3. In *Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22-23, 2006*, pages 158–161.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 1554–1564. Springer Berlin Heidelberg.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.

Highly Effective Arabic Diacritization using Sequence to Sequence Modeling

Hamdy Mubarak **Ahmed Abdelali** **Hassan Sajjad** **Younes Samih** **Kareem Darwish**
Qatar Computing Research Institute, HBKU Research Complex, Doha 5825, Qatar
{hmubarak,aabdelali,hsajjad,ysamih,kdarwish}@qf.org.qa

Abstract

Arabic text is typically written without short vowels (or diacritics). However, their presence is required for properly verbalizing Arabic and is hence essential for applications such as text to speech. There are two types of diacritics, namely core-word diacritics and case-endings. Most previous works on automatic Arabic diacritic recovery rely on a large number of manually engineered features, particularly for case-endings. In this work, we present a unified character level sequence-to-sequence deep learning model that recovers both types of diacritics without the use of explicit feature engineering. Specifically, we employ a standard neural machine translation setup on overlapping windows of words (broken down into characters), and then we use voting to select the most likely diacritized form of a word. The proposed model outperforms all previous state-of-the-art systems. Our best settings achieve a word error rate (WER) of 4.49% compared to the state-of-the-art of 12.25% on a standard dataset.

1 Introduction

Arabic uses two types of vowels, namely long vowels, which are explicitly placed in the text, and short vowels, which are diacritic marks that are typically omitted during writing. In order to read Arabic words properly, readers need to reintroduce the missing diacritics. Therefore, accurate diacritic recovery is essential for some applications such as text-to-speech. There are, in turn, two types of Arabic diacritics, namely core-word diacritics (CW), which specify lexical selection, and case endings (CE), which typically indicate syntactic role. For example, the word “AlElm”[1] can accept many possible core-word diacritics depending on the intended meaning such as: “AaloEalam” (the flag) and “AaloEilom” (the knowledge/science). In our training corpus, 17.1% of the word-cores have more than one valid diacritized form. In the sentence “Zahara AaloEalam**u**” (the flag appeared), “AaloEalam**u**” is the subject and takes the case ending “**u**”, and in the sentence “ra>ayotu AaloEalam**a**” (I saw the flag), “AaloEalam**a**” is the object and takes the case ending “**a**”. Aside from function words, past tense and accusative verb forms, and foreign names, most words can accept different case-endings depending on context.

In this paper, we introduce a unified model for both diacritic types while improving upon the state-of-the-art. Specifically, we approached the task as a sequence-to-sequence (seq2seq) problem (Cho et al., 2014); taking advantage of the recent advancements in Neural Machine Translation (NMT) (Britz et al., 2017; Kuchaiev et al., 2018) among other applications where seq2seq models made a breakthrough (Yu et al., 2016; Witten et al., 2016; Abadi et al., 2016). Using the analogy of translation which employs a sequential encoder and a sequential decoder, the input undiacritized text will be encoded and then decoded into diacritized form.

As we show later, directly applying a seq2seq model at sentence level using word or character representations produces nearly unusable results that are much worse than the state-of-the-art due to word insertions, omissions, and substitutions. Such problems are exaggerated when using word-based models due to Out-Of-Vocabulary words (OOVs). Conversely, character-based models suffer from not learning long-term dependencies. To avoid these problems, we train a seq2seq model on a sliding window of words that are represented using characters, and we employ voting to pick the best most likely diacritized form from different windows. In doing so, we provide suffi-

---

[1] In this paper, we use Buckwalter transliteration.

Proceedings of NAACL-HLT 2019, pages 2390–2395
Minneapolis, Minnesota, June 2 - June 7, 2019.

cient context to properly guess proper diacritized forms, while stinting the aforementioned undesirable word operations. Further, the use of voting has the effect of picking the most frequent diacritized form obtained from applying the model on different contexts. The resultant system makes full use of NMT machinery to achieve error rates that are 63.3% lower than the best state-of-the-art system[2].

The contributions in this paper are:

- Adaptation of neural machine translation for Arabic diacritic recovery with voting.
- Unified model to handle both core-diacritics and case-endings.
- Substantial improvement over state-of-the-art with 4.49% word error rate compared to 12.25%.

2 Background

One of the first approaches to Arabic diacritization used a Hidden Markov Model (Gal, 2002; Elshafei et al., 2006) that was applied to the Qur'an achieving double digit word error rates (WER). Vergyri and Kirchhoff (2004) used acoustic features in conjunction with morphological and contextual constrains to train a diacritizer and reported a 9% and 28% diacritics error rate (DER) without and with CEs. Since words are composed of multiple letters with corresponding diacritics, DER values are typically lower than WER values. Nelken and Shieber (2005) used a cascade of word-level, character-level, and morphological models finite state transducers to attain a Word Error Rate (WER) of 7.33% without CE and and 23.61% WER with CE. Zitouni et al. (2006) trained a maximum entropy model for sequence classification using the LDCs Arabic Treebank (ATB) and attained a WER of 18% (with CE) on 600 articles from An-Nahar Newspaper.

Habash and Rambow (2005); Rashwan et al. (2011); Bebah et al. (2014); Pasha et al. (2014); Metwally et al. (2016); Darwish et al. (2017) combined morphological features along with POS tagging information and n-gram language models. MADA-D system (Habash and Rambow, 2007) achieved a 5.5% and 14.9% WER respectively without and with CE. MADAMIRA (Pasha et al., 2014) ranks a list of candidate analyses from the Buckwalter analyzer (Buckwalter, 2004) using an SVM classifier and achieves 19.0% and 6.7% WER with and without CE respectively. Similarly, Microsoft Arabic Toolkit Services (ATKS) diacritizer (Said et al., 2013) uses a rule-based morphological analyzer that produces possible analyses and an HMM in conjunction with rules to guess the most likely analysis. They report WER of 11.4% and 4.4% with and without CE. Darwish et al. (2017) used a Viterbi decoder to guess core word diacritics and SVM-rank to guess case-endings and their system achieves a WER of 3.29% and 12.77% for words without and with CE. They trained their system using a large corpus of roughly 4.5 million words.

More recent work employed different neural architectures to model the diacritization problem. Abandah et al. (2015) used a biLSTM-based recurrent neural network trained on the same dataset as (Zitouni et al., 2006), and they report a WER of 9.1% including CE on ATB. Similar architectures were explored but with lower results (Rashwan et al., 2015; Belinkov and Glass, 2015). Azmi and Almajed (2015); Osama Hamed (2017) survey recent work on Arabic diacritization. They concluded that: reported results are often incomparable due to the usage of different test sets; a large unigram LM for CW diacritic recovery is competitive with many of the systems in the literature. In this paper, we compare our results to those of Said et al. (2013); Pasha et al. (2014); Rashwan et al. (2015); Belinkov and Glass (2015); Darwish et al. (2017) on standard test set.

3 Methodology

Representation Unit. The diacritization of Arabic is a word-internal property dependent on both character and word-level contexts. Therefore, we consider characters as units of representation. We represent source sentences as a sequence of characters by adding a space after every character and a word boundary "_" between words. The target side, which is fully diacritized, is split into a sequence of subword units each consisting of a letter and its diacritic(s). For example, source word "AlElm" would be represented as "A/l/E/l/m" and its diacritized target "AaloEalamu" as "Aa/lo/Ea/la/mu".

The character-level representation has several benefits, such as reducing the vocabulary size and avoiding OOV words. The splitting of diacritized

---

[2] Patent pending.

words into subword units simplifies the problem as there will be identical number of source and target tokens in a parallel sentence. Later, we support our design decisions with results in the experiments section. Subwords (BPE (Sennrich et al., 2016)) have been used as a defacto standard in building NMT systems. They are a natural choice to handle unknown words. However, BPE does not fit in our scenario as it may create source and target segments of different lengths. In the Arabic diacritization problem, both source and target words and characters are strictly tied to each other and loosening it would result in sub-optimal performance and may generate unexpected errors.

Context Window. The diacritization of Arabic words is highly sensitive to context. Character representations significantly increase the size of the source and target sequences. This leads to a well known limitation of character-based LSTM-based models, namely poor handling of long range dependencies (Sennrich, 2017). An easy fix is to split sentences greater than a certain length into multiple lines. However, boundary words may loose context in the newly created sequences. To handle this, we propose to keep a fixed size context window c for every word. Given a sentence, we use a sliding context window to split it into segments of overlapping windows of size c as in Table 1. This fixes the problems of both long range dependencies and context of neighboring words. We are further aided by the fact that local context can conclusively determine the correct diacritization in the vast majority of cases.

Voting. As shown in Table 1, the sliding window approach replicates a word present in a sentence c times. At test time, a word may get different diacritized forms from different contexts. We use voting to choose the most frequent diacritized form from the c predictions. In case of a tie, we favor the context where the word appears in the middle.

Sentence	w_1	w_2	w_3	w_4	w_5
c=3	w_1	w_2	w_3		
		w_2	w_3	w_4	
			w_3	w_4	w_5

Table 1: Example sentence: w_1 w_2 w_3 w_4 w_5 with context window c of size 3.

	Train		Test		
	Total	Uniq	Total	Uniq	OOV
Diacritized	4.5M	333k	18.3k	7.9k	5.0%
Undiacritized		209k		6.8k	3.3%

Table 2: Number of tokens in training and test data.

Seq2Seq Model. We use a seq2seq model consisting of three main components: i) Encoder, ii) Decoder, and iii) Attention. Given a source sentence $s = w_1, ..., w_N$ and a target sentence $t = v_1, ..., v_N$, the encoder models the source sentence and computes a set of hidden states $h = h_1, ..., h_N$. The attention mechanism (Bahdanau et al., 2014) computes a weighted average of these hidden states from the previous decoder state, known as the context vector c_i, while decoder models the target sentence. The seq2seq model is trained jointly on a large parallel corpus by maximizing the log-likelihood of the data:

$$log\ p(t|s) = \sum_{i} \sum_{j=1}^{|t^i|} log\ p(v_j^i | v_1^i, ..., v_{j-1}^i, s^i) \tag{1}$$

where s^i and t^i are the i^{th} source and target sentences. In addition, we experiment with a Transformer model (Vaswani et al., 2017), which is an attention based architecture without LSTM, and compare its results to the aforementioned seq2seq model with attention.

4 Experimental Setup and Results

Data. We used a modern diacritized corpus of 4.5 million tokens that covers a wide range of topics such as politics, religion, sport, health, and economics. For testing, we used the freely available WikiNews corpus (18,300 words) (Darwish et al., 2017) as a test set, which covers a variety of genres. Table 2 reports the size of the training and test sets including the unique diacritized and undiacritized tokens and the percentage of OOVs in the test set that don't appear in the training set. We randomly used 10% of the train data for validation and the rest for training. We used a sequence length of 100, 500 and 7 tokens for word-, character-, and window-based systems respectively. The vocabulary is restricted to 100k words types and 1,000 character units.

System Settings. The settings for LSTM-based Seq2Seq model were: word embeddings and LSTM states = 512; 2 layer unidirectional LSTM;

Exp	Description	Core WER%	CE WER%	WER%	DER%	OOV%
01	Baseline Word	44.29	54.95	54.31	41.62	13.03
02	Baseline Char	41.29	41.95	48.31	36.62	0.00
03	Word 7g	14.83	19.01	20.69	18.92	11.04
04	Char 7g	2.78	6.11	8.32	2.19	0.00
05	Word 7g+overlap	14.50	16.57	18.05	18.14	10.97
06	Char 7g+overlap	2.04	3.23	4.94	1.34	0.00
07	Char 3g+overlap+voting	2.31	5.97	7.79	2.01	0.00
08	Char 5g+overlap+voting	2.37	3.57	5.49	1.49	0.00
*09	Char 7g+overlap+voting	1.99	3.07	4.77	1.30	0.00
10	Char 11g+overlap+voting	3.03	3.93	6.40	1.78	0.00
†11	Char 7g+overlap+voting (Transformer)	2.05	3.04	4.77	1.29	0.00
12	Combination *09 +† 11	**1.89**	**2.89**	**4.49**	**1.21**	0.00

Table 3: Diacritization results: *g represents ngram size e.g. 7g means 7-gram context. Experiment 09 and 11 are comparing NMT models – LSTM-based architecture with attention mechanism and Transformer model

and dropout rate = 0.3. The setting for the Transformer were: 6 encoder and 6 decoder layers each of size 512; number of attention heads = 8; feed forward dimension = 2048; and dropout = 0.1. We used the OpenNMT (Klein et al., 2017) implementation with tensorflow for all experiments.

System Runs. We conducted a variety of experiments as follows, namely: **Word-level experiments** where the input is a sequence of words and the output is a sequence of diacritized words:
– Baseline Word: uses the full sentences and shows the deficiency of using NMT directly.
– Word 7g: uses non-overlapping windows of 7 words to compare to our best character-level model, which also uses a window of length 7.
– Word 7g+overlap: uses a sliding window of 7 words.
Character-level experiments where the input is represented as a sequence of character and the output as a sequence of diacritized characters:
– Baseline Char: uses the full sentence.
– Char 7g: uses non-overlapping sequences of 7 words.
– Char 7g+overlap: uses a sliding window of 7 words without voting.
– Char ng-overlap+voting: uses a sliding window of n words with voting, where we varied n to equal 3, 5, 7, and 11. When $n = 7$, we experimented with a seq2seq model with attention, a Transformer model, and a combination of both.

Results. Table 3 summarizes the results of our experiments. As the results clearly show, using an NMT model at word or character level produced unusable results. Both Baselines suffer also from excessive repetition of characters that are often meaningless hallucination (e.g. "AalofaA}iti AaloHaAdiy waAlt∼awaAliy Aaloayoiy AloanohaAti AaloanohaAti"). When we limited the context to 7 words, the results improved dramatically, nonetheless, the output still suffered from a high ratio of OOVs. Using characters instead alleviated the OOV problem. The results improved dramatically with contexts of length 7 yielding the best results. Using voting lowered WER rate further, leading to a 4.77% WER. Using a Transformer model led to nearly identical WER to using our NMT model with attention. However their results are somewhat complimentary. Thus, voting on the predictions across both systems improved the results further with a 4.49% WER. Table 4 compares our best system with other systems. The WER of our best system is 63.3% lower than the state-of-the-art.

Setup	WER%
Our System	**04.49**
Microsoft ATKS (Said et al., 2013)	12.25
Farasa (Darwish et al., 2017)	12.76
RDI (Rashwan et al., 2015)	15.95
MADAMIRA (Pasha et al., 2014)	19.02
MIT (Belinkov and Glass, 2015)	30.50

Table 4: Comparison to other systems for full diacritization (Darwish et al., 2017).

Error Analysis. we randomly selected 100 errors word-core and 100 case-ending errors we ascertain the most common error types. For case-ending, the top 4 error types were: long-distance dependency (e.g. coordination or verb subj/obj), which is an artifact of using limited context – 24%

of errors; confusion between different syntactic functions (e.g. N N vs. N ADJ or V Subj vs. V Obj) – 22%; wrong selection of morphological analysis (e.g. present tense vs. past tense) – 20%; and named entities (NEs) – 16%. For long distance dependencies, increasing context size may help in some case, but may introduce additional errors (see Table 3). Perhaps combining multiple context sizes may help. As for word-core, the top 4 errors were: incorrect selection for ambiguous words, where most of these errors were related to active vs. passive voice – 60%; NEs – 32%; borrowed words – 4%; and words with multiple valid diacritized words – 4%.

5 Conclusion

In this paper, we adapted a seq2seq model to build a unified model for Arabic diacritic recovery. We trained the model on a fixed length sliding window of n words that are represented using their characters. We further employed voting to pick the most common diacritized form of a word in different contexts. The adaptation yielded a word error rate of 4.49%, which is 63.3% lower than the best state-of-the-art system. One possible future direction is to use a system combination with varying context sizes with a weighted voting scheme. Further, the explicit inclusion of a large gazetteer of diacritized NEs in the training set would help diacritize them properly. We also want to examine the effect of training data size to determine if more data would yield better results.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Aqil M Azmi and Reham S Almajed. 2015. A survey of automatic arabic diacritization techniques. *Natural Language Engineering*, 21(03):477–495.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of arabic texts. *arXiv preprint arXiv:1410.2646*.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285, Lisbon, Portugal.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

Tim Buckwalter. 2004. Buckwalter arabic morphological analyzer version 2.0. *LDC catalog number LDC2004L02, ISBN 1-58563-324-0*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Kareem Darwish, Hamdy Mubarak, and Ahmed Abdelali. 2017. Arabic diacritization: Stats, rules, and hacks. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 9–17.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. 2006. Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in arabic and hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to Semitic languages*, pages 1–7. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. 2018. Openseq2seq: extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. *arXiv preprint arXiv:1805.10387*.

Aya S Metwally, Mohsen A Rashwan, and Amir F Atiya. 2016. A multi-layered approach for arabic text diacritization. In *Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on*, pages 389–393. IEEE.

Rani Nelken and Stuart M Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics.

Torsten Zesch Osama Hamed. 2017. A Survey and Comparative Study of Arabic Diacritization Tools. *JLCL*, 32(1):27–47.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC-2014*, Reykjavik, Iceland.

Mohsen Rashwan, Ahmad Al Sallab, M. Raafat, and Ahmed Rafea. 2015. Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. In *IEEE Transactions on Audio, Speech, and Language Processing*, pages 505–516.

Mohsen AA Rashwan, Mohamed ASAA Al-Badrashiny, Mohamed Attia, Sherif M Abdou, and Ahmed Rafea. 2011. A stochastic arabic diacritizer based on a hybrid of factorized and unfactorized textual features. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):166–175.

Ahmed Said, Mohamed El-Sharqwi, Achraf Chalabi, and Eslam Kamal. 2013. A hybrid approach for arabic diacritization. In *Natural Language Processing and Information Systems*, pages 53–64, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of arabic for acoustic modeling in speech recognition. In *Proceedings of the workshop on computational approaches to Arabic script-based languages, COLING'04*, pages 66–73, Geneva, Switzerland. Association for Computational Linguistics.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593.

Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics.

SC-LSTM: Learning Task-Specific Representations in Multi-Task Learning for Sequence Labeling

Peng Lu
RALI-DIRO
Université de Montréal
Montréal, Canada
peng.lu@umontreal.ca

Ting Bai[*]
School of Information
Renmin University of China
Beijing, China
baiting@ruc.edu.cn

Philippe Langlais
RALI-DIRO
Université de Montréal
Montréal, Canada
felipe@iro.umontreal.ca

Abstract

Multi-task learning (MTL) has been studied recently for sequence labeling. Typically, auxiliary tasks are selected specifically in order to improve the performance of a target task. Jointly learning multiple tasks in a way that benefit all of them simultaneously can increase the utility of MTL. In order to do so, we propose a new LSTM cell which contains both shared parameters that can learn from all tasks, and task-specific parameters that can learn task specific information. We name it a Shared-Cell Long-Short Term Memory (SC-LSTM). Experimental results on three sequence labeling benchmarks (named-entity recognition, text chunking, and part-of-speech tagging) demonstrate the effectiveness of our SC-LSTM cell.

1 Introduction

As one of the fundamental tasks in NLP, sequence labeling has been studied for years. Before the blooming of neural network methods, handcrafted features were widely used in traditional approaches like CRFs, HMMs, and maximum entropy classifiers (Lafferty et al., 2001; McCallum et al., 2000; McCallum and Li, 2003; Florian et al., 2003). However, applying them to different tasks or domains is hard. Recently, instead of using handcrafted features, end-to-end neural network based systems have been developed for sequence labeling tasks, such as LSTM-CNN (Chiu and Nichols, 2015), LSTM-CRF (Huang et al., 2015; Lample et al., 2016), and LSTM-CNN-CRF (Ma and Hovy, 2016). These models utilize LSTM to encode the global information of a sentence into a word-level representation of its tokens, which avoids manual feature engineering. Moreover, by incorporating a character-level representation of tokens, these models further improve.

In many such studies, though, neural network models are trained toward a single task in a supervised way by making use of relatively small annotated training material. Jointly learning multiple tasks can reduce the risk of over-fitting to one task, and many attempts have been made at doing so for sequence labeling tasks (Caruana, 1997; Collobert and Weston, 2008; Collobert et al., 2011). Results so far are not conclusive.

Some works have reported negative results overall. For instance in their pioneering work, Collobert et al. (2011) observed that training their model on NER, POS tagging and chunking altogether led to slight decrease in performance compared to a similar model trained on each task separately. Søgaard and Goldberg (2016) study chunking and CCG super tagging, coupled with an additional POS tagging task. They do report gains on both target tasks over single task models, but results varied depending where the additional task was taken care of in their architecture. The authors actually reported a failure to leverage other labelling tasks, and concluded that combined tasks should be sufficiently similar to the target one, for significant gains to be observed. Similarly, Alonso and Plank (2017) achieved significant improvements for only 1 out of 5 tasks considered. Also of interest is the work of (Changpinyo et al., 2018) where the authors investigate the classical shared encoder-based MTL framework (Collobert et al., 2011; Collobert and Weston, 2008) on 11 sequence labeling datasets including POS, NER, and chunking. They report that chunking is beneficial to NER, while POS tagging can be harmful.

We present in Section 2 the two major approaches proposed for multi-task learning and discuss their limitations. We describe our approach in Section 3, and present our experimental settings and results in Section 4 and 5 respectively. We further analyze our approach in Section 6, discuss

---

[*] Co-first author.

Proceedings of NAACL-HLT 2019, pages 2396–2406
Minneapolis, Minnesota, June 2 - June 7, 2019.

related works in Section 7 and conclude in Section 8.

2 Multi-task Learning

2.1 Problem Statement

We are given a set of K tasks (in our case named-entity recognition, text chunking and part of speech tagging) that we want to train jointly in an end-to-end fashion. Each task k has an associated training set $S_k = \{(x_i^k, y_i^k)_{i \in [1,n_k]}\}$ of n_k examples, where x_i^k and y_i^k are sequences of size m_i of tokens and tags respectively. We wish to learn a single function $\mathcal{F}$ which maps any token input sequence x_i to its task-specific labels, where the mapping defines a probabilistic distribution for each involved task: $p(y_i^1, \ldots, y_i^K) = \mathcal{F}(x_i)$.

2.2 Existing Shared Encoder Methods

There are two kinds of neural-based MTL methods. The first one — LSTM-s hereafter — uses an identical representation for all tasks, as proposed in (Collobert et al., 2011). This is illustrated in Figure 1 (left), where 3 layers of LSTMs are being stacked.[1] While different tasks directly interact with all parameters of the model, this increases the risk of optimization conflicts when gold-standard labels from different tasks have no significant correlation.

The second class of multi-task architectures is depicted in the middle part of Figure 1, and is named LSTM-d hereafter. In this configuration, each LSTM layer feeds a task-specific classifier and serves as input to the next stacked LSTM layer (Søgaard and Goldberg, 2016). The underlying assumption is that tasks may be ordered in such a way that easier tasks are learned first, the target tasks being the latest one considered, thus benefiting the hidden state of the lower layers. One drawback, however, is that one must decide which task to consider first, a decision which may impact the overall performance. Furthermore, using the hidden state of lower layers increases the limitation of learning representation for that task.

We believe that one reason for the lack of consistent benefits of MTL in the labelling literature is that the proposed models share all or part of parameters for extracting hidden states, which leads to optimization conflicts when different tasks require different features. We believe it would be helpful if we make the model have ability to learn a task-specific representation (Ammar et al., 2016; Östling and Tiedemann, 2016; Kiperwasser and Ballesteros, 2018) at the same time. This observation led us to design a new LSTM cell which allows at almost no additional computation cost to efficiently train a single RNN-based model, where task-specific labelers clearly outperform their singly-tasked counterparts. Actually, by training our model on NER, chunking and POS tagging, we report state-of-the-art (or highly competitive) results on each task, without using external knowledge (such as gazetteers that has been shown to be important for NER), or hand-picking tasks to combine.

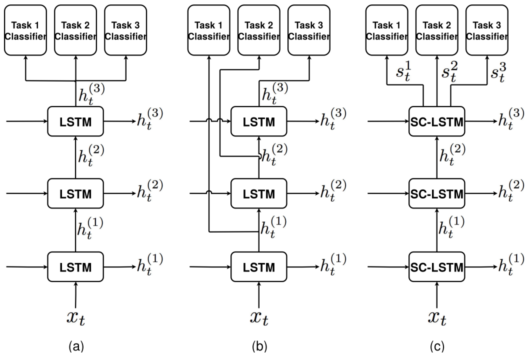


Figure 1: Overview of three shared encoder MTL tagging system: (a) LSTM-s (b) LSTM-d (c) our SC-LSTM based system.

3 Proposed Method

Our solution is depicted in the right part of Figure 1 and detailed in the next section. It is actually very similar to the LSTM-s one, except that each LSTM layer passes on task-specific hidden representations that learn the peculiarities of individual tasks. In the last layer, each classifier is fed with a concatenation of a global representation (as in LSTM-s) and the task-specific one. By doing so, we keep the advantages of both aforementioned approaches, where one task can have its task-specific representation as in LSTM-d, while not enforcing any task order, further giving the freedom to the model to learn specificities of each task. For this architecture to work, we need to modify the classical LSTM cell, which is described in the next section.

---

[1] This typically delivers better performance than having just one. In practice also, LSTM layers are replaced by bi-LSTM ones.

3.1 LSTM Cell

An LSTM cell (Hochreiter and Schmidhuber, 1997) is made up of four functional gates which control the input and output of the memory state $\mathbf{c_t}$: a forget gate $\mathbf{f_t}$ controls what information to remove from the memory state of the last time step, an input gate $\mathbf{i_t}$ controls the information to add to the current memory, and an output gate $\mathbf{o_t}$ controls what information to release from the current memory state. This mechanism is formalized in Equation 1 where $\mathbf{x}_t$, $\mathbf{h}_t$ is the input vector and hidden vector at time step t, σ is the sigmoid function, $\hat{\mathbf{c}}$ is the new candidate state, $\mathbf{c}_t$ is the memory state, which encodes information of the current input and history information, and $*$ indicates the element-wise product. $\mathbf{W}_f, \mathbf{U}_f, \mathbf{W}_i, \mathbf{U}_i, \mathbf{W}_o, \mathbf{U}_o, \mathbf{W}_c, \mathbf{U}_c$ are weight matrices that are being learned.

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t), \\
\hat{\mathbf{c}} &= \tanh(\mathbf{W}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{x}_t), \\
\mathbf{c}_t &= \mathbf{i}_t * \hat{\mathbf{c}} + \mathbf{f}_t * \mathbf{c}_{t-1}, \\
\mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{c}_t),
\end{aligned}
\tag{1}
$$

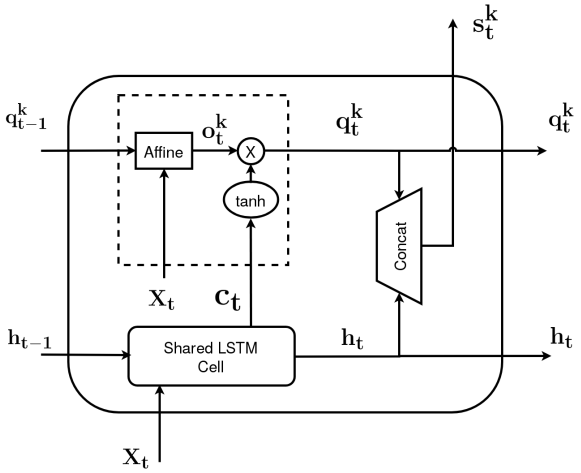


Figure 2: Structure of an SC-LSTM cell. The dashed delimited box depicts the task-specific cell. For clarity reasons, we only show one such cell, while in practice there is one task-specific cell for each task.

3.2 SC-LSTM Cell

The overall structure of our cell is depicted in Figure 2. On top of a standard LSTM cell, we add one cell per task with its own parameters. The standard LSTM cell is thus shared among the K task-specific cells, therefore the name we choose for this new cell, which stands for Shared-Cell LSTM. Task-specific cells are each parametrized by an output gate $\mathbf{o}_t^k$ which learns to select the useful information from the shared memory cell $\mathbf{c}_t$ and outputs $\mathbf{q}_t^k$. This is formally described in Equation 2, where $\mathbf{W}_k$ and $\mathbf{U}_k$ are two extra weight matrices that parametrize the kth task, and $\mathbf{q}_t^k$ has to be understood as a task-specific hidden representation since parameters of kth task-specific cell are only updated by supervision from task k.

$$
\begin{aligned}
\mathbf{o}_t^k &= \sigma(\mathbf{W}_k \mathbf{q}_{t-1}^k + \mathbf{U}_k \mathbf{x}_t^k) \\
\mathbf{q}_t^k &= \mathbf{o}_t^k * \tanh(\mathbf{c}_t)
\end{aligned}
\tag{2}
$$

In order to make use of both shared and task-specific information (Kim et al., 2016; Peng et al., 2017; Hershcovich et al., 2018), for the kth task, we concatenate the output of the shared cell $\mathbf{h}_t$ and of the task-specific one $\mathbf{q}_t^k$ to generate the final latent representation, as noted in Equation 3, where $\oplus$ is the concatenation operation. In practice, we stack SC-LSTM layers. The top-most layer uses $\mathbf{s}_t^k$ as a representation of the current input, while cells in lower layers pass the current shared hidden state $\mathbf{h}_t$ to the upper SC-LSTM cell. The use of s in the topmost layer only is arbitrary and should be investigated.

$$
\mathbf{s}_t^k = \mathbf{q}_t^k \oplus \mathbf{h}_t \tag{3}
$$

3.3 Training procedure

The training material available may be gathered from different datasets S_k which means that input sequences differ from one task to another. Therefore in practice, we build K dataloaders, and the training is achieved in a stochastic manner by looping over the tasks at each epoch, as detailed in Algorithm 1.

The loss function ϵ we minimize is a linear combination of task-specific loss functions, where the weighting coefficients (λ^k in Equation 4) are hyper-parameters.

$$
\epsilon = \sum_{k=1}^{K} \lambda^k L(\mathbf{y}^k, \hat{\mathbf{y}}^k) \tag{4}
$$

Algorithm 1 Stochastic training procedure

```
1: procedure TRAINING
2:     for each epoch do
3:         Randomly choose a task $k$.
4:         Randomly choose a training example not yet considered in the dataset $S_k$
5:         Update the parameters for this task by taking a gradient step based on this example.
6:         Go to step 3 until using all examples.
```

We seek to minimize cross-entropy of the predicted and true distributions, therefore task-specific loss functions are defined according to Equation 5.

$$L(\mathbf{y}^k, \hat{\mathbf{y}}^k) = -\sum_{i=1}^{n_k} \sum_{j=1}^{m_i} \mathbf{y}_{i,j}^k \log \hat{\mathbf{y}}_{i,j}^k \qquad (5)$$

where $\hat{\mathbf{y}}_{i,j}^k$ is the prediction of the kth softmax classifier parametrized by a projection matrix $\mathbf{W}^k$ and a bias vector $\mathbf{b}^k$:

$$\hat{\mathbf{y}}_{i,j}^k = softmax(\mathbf{W}^k \mathbf{s}_{\mathbf{i,j}}^{\mathbf{k}} + \mathbf{b}^k) \qquad (6)$$

where $\mathbf{s}_{\mathbf{i,j}}^{\mathbf{k}}$ stands for jth token of the ith sequence for task k. In an SC-LSTM cell with k tasks, we will add k matrices and k bias vectors, compared with a vanilla LSTM cell, which increases the capacity of the resulting model. This extra calculation is conducted in parallel with the original LSTM computations.

4 Experimental Setting

4.1 Benchmarks

We test several baseline systems and our SC-LSTM model on three well-established sequence labeling benchmarks: CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) for named-entity recognition, CoNLL2000 (Tjong Kim Sang and Buchholz, 2000) for chunking, and the more recent Universal Dependency dataset (Nivre et al., 2016) for part-of-speech tagging,[2] and on which recent MTL investigations have been conducted (Alonso and Plank, 2017; Changpinyo et al., 2018).

We conformed to the pre-defined splits into train/dev/test except for chunking that does not contain a validation set. For this dataset, following recent works (Peters et al., 2017; Liu et al., 2017), sections 15-18 of the Wall Street Journal are used for training, and we randomly sampled 1000 sentences in the training set as the development set. Section 20 is used for tests. Table 1 presents the main characteristics of the training, development and test sets we used.

	Train	Dev	Test
NER			**#tags** = 4
#sentences	14,987	3,644	3,486
#tokens	205k	52k	47k
#entities	23,523	5,943	5,654
Chunking			**#tags** = 10
#sentencies	8,936	-	2,012
#tokens	212k	-	47k
#chunks	107k	-	24k
POS			**#tags** = 17
#sentences	12,543	2,002	2,077
#tokens	204k	25k	25

Table 1: Main characteristics of the datasets used.

4.2 Architectural Choices

We used bidirectional LSTM or SC-LSTM as our encoders for the vector representation of words. Bidirectional LSTM can capture global information of the whole sentence, thanks to the encoding of a sequence in a recurrent way. The vector representation of words consists of three parts: word embedding, character-level representation and contextual representation:

- previous works have proven that pre-trained word embeddings like Word2vec (Mikolov et al., 2013), SENNA (Collobert et al., 2011), or Glove (Pennington et al., 2014) have a positive impact on sequence labeling tasks. We used Glove embeddings[3] of dimension 100, that are fine-tuned during training;

- character-level information has been proven useful for three sequence labeling tasks (Santos and Zadrozny, 2014; Ma and Hovy, 2016; Lample et al., 2016) and Some works further show its effectiveness (Reimers and Gurevych, 2017; Yang et al., 2018). In order to encode character sequences, we used a CNN. The character embedding look-up table is initialized by randomly sampling from

---

[2] We used UD English POS v1.3.

[3] The embedding file `glove6B` are available `https://nlp.stanford.edu/projects/glove/`

the uniform distribution in the range [-0.1, 0.1];

- the third part of the input vector is contextual embedding. Most of the recent works found that contextualized features such as ELMo or BERT (Peters et al., 2018; Devlin et al., 2018) can greatly boost performance. We incorporate ELMo into our input vector by using the ELMo implementation of Gardner et al. (2018).

Conditional random field (CRF) classifiers can consider the dependency of output tags and has been proven useful for tasks like NER or chunking. We consider CRF layers in our models.

To compare the effectiveness of three MTL models, we first test our SC-LSTM and other vanilla LSTM based models without CNN based character-level information extractor and contextual embeddings. In this case, the input will be a concatenation of word embedding and capitalization features[4]. To compare with the state-of-art models, we further implemented three more variants: SC-LSTM-CNN-CRF which makes use of CNN-based character level features with a CRF layer on top, very similar to (Ma and Hovy, 2016), and SC-LSTM-LM-CNN, a variant which considers contextualized word embeddings as in (Peters et al., 2018).

4.3 Implementation

For SC-LSTM-LM-CNN, we used mainly the configuration advocated in (Peters et al., 2018) except for the hidden size of SC-LSTM, which we report in Table 2. We further weighted the NER task in the objective function of Equation 4 to 3 (λ_{NER}), the weights of the other tasks where set to 1.[5] We trained this model SC-LSTM-LM-CNN using the Adam optimizer (Kingma and Ba, 2014) with default setting. Such a model spends typically less than 30 epochs to converge. We choose the mini-batch size of 10 and used the gradient clipping threshold 5. All models are implemented using the Pytorch library (Paszke et al., 2017).

Several tagging schemas have been proposed for conducting chunking and NER tasks. We used the most common BIO one in this work.

Layers	Parameters	
Char-level Embedding	dimension	16
Character-CNN	kernel size	3
	padding	1
	stride	1
	channel	128
Word Embedding	dimension	100
SC-LSTM	hidden size	256∗3
	layer	3
ELMo	dimension	1024
Dropout	rate	0.5

Table 2: Hyper-parameters used for training the SC-LSTM-LM-CNN model.

5 Experiments

In this section, we start by comparing MTL approaches based on LSTM and SC-LSTM cells. We then report the performance of variants of our approach we implemented and compare it to state-of-the-art models.

5.1 biLSTM versus biSC-LSTM

We compare bidirectional LSTM (STL), SC-LSTM, and baseline MTL models. The results are shown in Table 3. Training Bi-LSTM models on each task separately (STL) was first conducted as a point of comparison (line 1).

For LSTM-s and LSTM-d, we regard one task as the main task and the others as auxiliary tasks; a setting consistent with Søgaard and Goldberg (2016). Because LSTM-s and LSTM-d always fail to achieve stable and competitive results on the three tasks we considered at the same time, we report the best performance we could obtain for each task (line 2 and 3) specifically. On the contrary, our SC-LSTM model is trained once jointly on all tasks, and only one model is being tested in the end, which is much easier and more realistic of a real deployment (line 4).

The results show that our SC-LSTM model improves the performance of the three tasks simultaneously compared with LSTM (STL), and outperforms the other two MTL methods. By joint learning three tasks, both LSTM-s and LSTM-d can boost the chunking task significantly, but both fail to improve NER and POS tasks. This is consistent with observations made in (Collobert et al., 2011; Søgaard and Goldberg, 2016). We also observe that our SC-LSTM model also benefits the chunking task the most. We will analyze this fur-

---

[4] Eight features encode capitalization patterns, such as AllUpper, InitialUpper, etc.

[5] We found the loss of the NER task to be around 1/3 of the loss of the chunking and POS tasks.

ther in Section 6.

	POS	chunking	NER
LSTM (STL)	95.46	94.44	89.39
LSTM-s	95.45	95.12	89.35
LSTM-d	95.44	95.24	89.37
SC-LSTM	**95.51**	**96.04**	**89.96**

Table 3: Results of models being trained in STL or MTL mode. For all MTL models, we report the best performance via a small grid search over combinations of the hidden size [100, 200, 300, 400] and the number of layers [1, 2, 3]. The best performance of each MTL model was obtained with hidden size 300 and 3 layers.

5.2 SC-LSTM versus state-of-the-art

To further demonstrate the effectiveness of our SC-LSTM model, we compared different variants with state-of-the-art approaches, that we classify into three broad categories:

- **Single sequence labeling** where models are trained without the supervision of other tasks. Specifically, we compare our results to the LSTM-CRF model of Lample et al. (2016) and the LSTM-CNN-CRF of Ma and Hovy (2016), since those are state-of-the-art singly tasked sequence labelers.
- **Multi-tasked sequence labelers** where models leverage the supervision of other tasks. We compare our model with the representative approaches of Luo et al. (2015); Søgaard and Goldberg (2016); Collobert and Weston (2008); Collobert et al. (2011).
- **Models with language model.** Recently, several studies in using contextualized word embeddings achieved great success in a number of tasks. Some recent studies (Peters et al., 2017; Rei, 2017; Peters et al., 2018; Devlin et al., 2018) are particularly considered.

It is worth noting that we did not engineer tasks specific features or integrate external ressources such as gazetteers in our variants.

5.2.1 NER

Results for the CoNLL 2003 dataset are reported in Table 4. We observe that our SC-LSTM-LM-CNN model outperforms all approaches but Devlin et al. (2018) and Akbik et al. (2018). The latter work is using the development set as training material, which avoids a direct comparison. The former model (BERT) is achieving great success by leveraging a huge amount of unannotated data as well as a lot of computation resources we could not afford in this study. We are however pleased that our model is leveraging contextual embeddings with 0.38 absolute F1 improvement over the results of Peters et al. (2018). We leave as future work to investigate whether our MTL model can leverage BERT embeddings.

NER	F1-score
Collobert et al. (2011)	89.59
Chiu and Nichols (2015) ♣	91.62
Huang et al. (2015)	88.83
Luo et al. (2015)	91.20
Ma and Hovy (2016)	91.21
Lample et al. (2016)	90.94
Shen et al. (2017)	90.89
Yang et al. (2017)	91.20
Rei (2017)	86.26
Liu et al. (2017)	91.71
Peters et al. (2017)	91.93
Zhang et al. (2018)	91.2
Liu et al. (2018)	91.95
Peters et al. (2018)	92.22
Clark et al. (2018)	92.60
Akbik et al. (2018) ♣	**93.09**
Devlin et al. (2018)	92.80
SC-LSTM	89.96
SC-LSTM-CNN-CRF	91.37
SC-LSTM-LM-CNN	92.60

Table 4: F1-score on the CoNLL03 NER dataset. Models with ♣ use both train and dev splits for training.

5.2.2 Chunking

We compared a number of models on the CoNLL2000 chunking dataset. A few of them (Ma and Hovy, 2016; Lample et al., 2016; Peters et al., 2018) where not tested on this benchmark, and we reimplemented them. We also trained the companion toolkits of those models, but (as detailed in the next section) got slightly lower results for some reasons. Table 5 reports the performance of the many approaches we tested.

We observe that our SC-LSTM-LM-CNN architecture achieves a new state-of-the-art F1 score, with over 1 absolute point over the competitive approach of Peters et al. (2017), and an improvement of 0.4% over the current state-of-the-art method of Clark et al. (2018).

Chunking	F1-score
Collobert et al. (2011)	94.32
Huang et al. (2015)	94.13
Ma and Hovy (2016)	94.81†
Lample et al. (2016)	94.68†
Hashimoto et al. (2016)	95.77
Søgaard and Goldberg (2016)	95.56
Shen et al. (2017)	93.88
Yang et al. (2017)	94.66
Liu et al. (2017)	95.96
Peters et al. (2017)	96.37
Liu et al. (2018)	96.13
Peters et al. (2018)	96.92†
Clark et al. (2018)	97.00
Akbik et al. (2018)	96.72
SC-LSTM	96.04
SC-LSTM-CNN-CRF	96.41
SC-LSTM-LM-CNN	**97.40**

Table 5: F1-score on the CoNLL00 chunking dataset. Configurations with a † sign are approaches we reimplemented.

5.2.3 POS

We conducted experiments on the Universal Dependency POS English dataset and present the results in Table 6. The only study we found that reports results on the UD v1.3 benchmark we used here is (Bjerva et al., 2016), and we report the results they published. For Liu et al. (2017), Peters et al. (2018) we used the available companion toolkits[6] that we trained ourself with the default settings. We re-implemented the other approaches.

Again, we observe that SC-LSTM-LM-CNN outperforms all other approaches we tested. The absolute improvement in F1 score over the current state-of-the-art of Peters et al. (2018) is 0.21%.

In order to further validate our implementations, we also ran the toolkits of Ma and Hovy (2016) and Lample et al. (2016) and obtained slightly lower results[7].

POS	Accuracy
Collobert et al. (2011)	95.41†
Huang et al. (2015)	95.63†
Ma and Hovy (2016)	95.80†
Lample et al. (2016)	95.78†
Bjerva et al. (2016)	95.67
Liu et al. (2017)	95.95†
Peters et al. (2018)	96.62 †
SC-LSTM	95.51
SC-LSTM-CNN-CRF	95.83
SC-LSTM-LM-CNN	**96.83**

Table 6: Accuracy on the Universal Dependency English POS dataset. Configurations with a † sign are approaches we reproduced.

6 Analysis

We conducted a number of investigations in order to understand better why our multi-task learning model is effective.

6.1 Convergence Analysis

We report in Figure 3 the convergence of different MTL models on the development set. To obtain those curves, we collected the F1-score on the NER and chunking tasks as well as the accuracy of the POS task, and averaged them after each epoch.

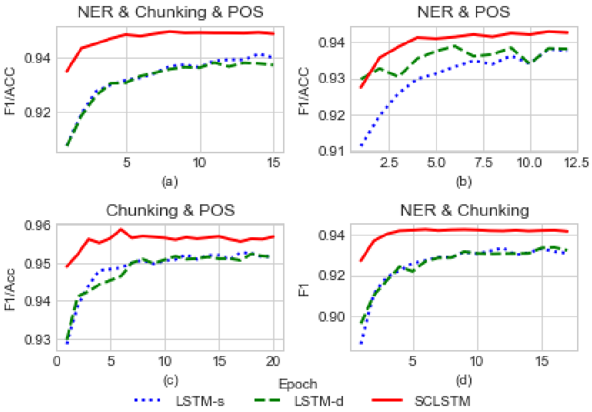


Figure 3: The change of F1-score/Accuracy on the development datasets at different epochs for SC-LSTM (red solid line), LSTM-s (blue dotted line) and LSTM-d (green dashed line) baselines. (a) reports results with NER, chunking and POS; (b) results with NER and POS; (c) results with chunking and POS; and (d) results with NER and chunking. All models were trained using the Adam optimizer with the same setting, and the mini-batch size was set to 10.

We clearly see that the SC-LSTM model converges faster than other ones. It achieves higher

---

[6] `https://github.com/LiyuanLucasLiu/LM-LSTM-CRF` and `https://github.com/allenai/allennlp`

[7] We obtained 95.7% for the toolkit we took from `https://github.com/XuezheMax/NeuroNLP2` and 95.6% for the one at `https://github.com/glample/tagger`.

performance after the first epoch, and after about ten epochs, it shows a smooth performance curve, while LSTM-s and LSTM-d models still fluctuate. This indicates that our model can learn the hidden representation of multiple tasks in a faster and smoother way than the other two methods.

Besides, we observe in Figure 3c and 3d that combinations of tasks involving chunking typically show a smooth training curve, on the contrary to Figure 3b where NER and POS tasks are combined. The fact that the training regimen fluctuates in the latter case for both LSTM-s and LSTM-d suggests that conflicts with those two tasks happen during optimisation, which we do not observe for our model. Also, Figure 3a illustrates that combining the three tasks altogether leads to comparably better performance of our model over LSTM-s and LSTM-d.

6.2 Effect of Different Task Combinations

We analyzed which task is benefited or harmed by others under the three MTL settings we considered and present results in Figure 4.

We find that it leads to better performance for all MTL models by jointly learning chunking with NER or POS(see in Figure 4b). This is in particular the case of our SC-LSTM model which records the largest gain, especially when all tasks are being trained on.

Figure 4c shows results obtained on POS. Only our SC-LSTM model achieves a meaningful improvement. We however observe that the NER task tends to hurt the performance of POS, since in most cases, the performance of POS+NER is lower than the one obtained with POS+chunking.

For NER, Figure 4 shows that POS hurts LSTM-s and LSTM-d models, while the chunking task is beneficial for all MTL methods.

Clearly, the combination of different tasks has a different effect on the final performance of each task. The chunking task seems compatible with NER and POS tasks, and it boosts the other two tasks in all three MTL settings, which is consistent with the results of Changpinyo et al. (2018). Directly jointly training on POS and NER datasets tends to reduce the performance in LSTM-s and LSTM-d, which is also consistent with the conclusion in (Changpinyo et al., 2018). In conclusion, all of the results show that our SC-LSTM model is effective at capturing the mutual benefits of all combined tasks. Since it performs consistently better in various settings, we believe our model to be more robust.

7 Related Work

There are many works that use extra knowledge to improve the performance of sequence labeling tasks.

Many works have focussed on jointly learning two tasks, often with one being considered as the main task, the other being the auxiliary one (Søgaard and Goldberg, 2016; Bjerva et al., 2016; Alonso and Plank, 2017). For instance, chunking, combinatory categorical grammar supertagging, NER, super senses (SemCor), or multiword expression + supersense will be taken as the main task, while POS is the auxiliary task in (Søgaard and Goldberg, 2016). Exceptions to this line of work include (Collobert et al., 2011) that evaluates four tasks: POS, chunking, NER and semantic role labeling; (Kiperwasser and Ballesteros, 2018) that considers a machine translation task with POS and dependency parsing. And Niehues and Cho (2017) considers machine translation with POS and NER tasks; Zhang and Weiss (2016) show that jointly learning a POS tagger and a dependency parser is effective. Miwa and Bansal (2016) jointly trained models for entity detection and relation extraction in the field of relation extraction.

Other works are also trying to leverage language models to empower the performance of sequence labeling tasks. Notably, Liu et al. (2017) propose a model which uses a neural language model to learn character-level knowledge, and conducts sequence labeling to guide the language model towards specific tasks. Others (Peters et al., 2017, 2018; Devlin et al., 2018) use neural language models pre-trained on a large unlabeled corpus to learn context-sensitive representations of words, and leverage this representation into the sequence labeling model.

More related to the present work are studies that analyze the effectiveness of different combinations of sequence labeling tasks in a multi-task learning. In particular, Changpinyo et al. (2018) conduct an investigation on 11 sequence labeling tasks, while Alonso and Plank (2017) evaluate 5 tasks but report signifiant gains for only one task.

8 Conclusion

In this paper, we propose a simple yet powerful LSTM cell that leverage both shared and task-

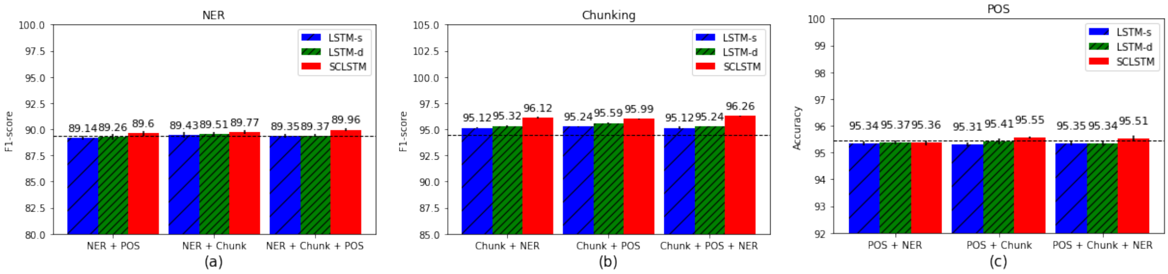


Figure 4: Results of different task groups on each test set: (a) NER, (b) chunking, and (c) POS. Horizontal lines show results for single-task models(NER: 89.39, Chunking: 94.44 and POS: 95.46).

specific parameters in a multi-task setting designed for sequence labelling. We conduct extensive experiments to compare both single-task learning and multi-task learning models. We analyzed the influence of grouping different tasks under various multi-task settings. Experiments demonstrate the effectiveness of our model for sequence labeling tasks. We report new state-of-the-art results on both POS and chunking tasks, and close to state-of-the-art performance on NER, without exploiting external ressources, neither diving into dedicated feature engineering.

Despite those positive outcomes, several issues with multi-task learning for sequence labeling remain open. In particular, we only considered 3 tasks here, and therefore plan to test our approach on more tasks, perhaps understanding better why some tasks are less useful to others. Also, there are several ways we could have used our SC-LSTM cell into our models which we would like to investigate further. In particular, in this work, we only used the specific-task hidden states in the last layer of the model, which can obviously be revisited.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Hector Martinez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *EACL 2017-15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–10.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. *arXiv preprint arXiv:1609.07053*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. *arXiv preprint arXiv:1808.04151*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *ACL workshop for NLP Open Source Software*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 373–385.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 387–396.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *arXiv preprint arXiv:1804.08915*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liyuan Liu, Xiang Ren, Jingbo Shang, Jian Peng, and Jiawei Han. 2018. Efficient contextualized representation: Language model pruning for sequence labeling. *arXiv preprint arXiv:1804.07827*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*, 2000, pages 591–598.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. *arXiv preprint arXiv:1708.00993*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with markov chain monte carlo. *The Prague Bulletin of Mathematical Linguistics*, 106(1):125–146.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2037–2048.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1756–1765.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2121–2130.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. *arXiv preprint arXiv:1806.04470*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Yuan Zhang, Hongshen Chen, Yihong Zhao, Qun Liu, and Dawei Yin. 2018. Learning tag dependencies for sequence tagging. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4581–4587. International Joint Conferences on Artificial Intelligence Organization.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1557–1566.

Learning to Denoise Distantly-Labeled Data for Entity Typing

Yasumasa Onoe and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
{yasumasa, gdurrett}@cs.utexas.edu

Abstract

Distantly-labeled data can be used to scale up training of statistical models, but it is typically noisy and that noise can vary with the distant labeling technique. In this work, we propose a two-stage procedure for handling this type of data: denoise it with a learned model, then train our final model on clean and denoised distant data with standard supervised training. Our denoising approach consists of two parts. First, a filtering function discards examples from the distantly labeled data that are wholly unusable. Second, a relabeling function repairs noisy labels for the retained examples. Each of these components is a model trained on synthetically-noised examples generated from a small manually-labeled set. We investigate this approach on the ultra-fine entity typing task of Choi et al. (2018). Our baseline model is an extension of their model with pre-trained ELMo representations, which already achieves state-of-the-art performance. Adding distant data that has been denoised with our learned models gives further performance gains over this base model, outperforming models trained on raw distant data or heuristically-denoised distant data.

1 Introduction

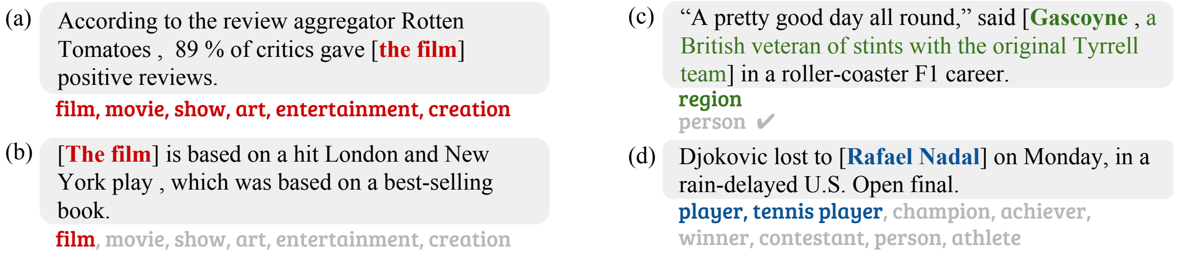
With the rise of data-hungry neural network models, system designers have turned increasingly to unlabeled and weakly-labeled data in order to scale up model training. For information extraction tasks such as relation extraction and entity typing, distant supervision (Mintz et al., 2009) is a powerful approach for adding more data, using a knowledge base (Del Corro et al., 2015; Rabinovich and Klein, 2017) or heuristics (Ratner et al., 2016; Hancock et al., 2018) to automatically label instances. One can treat this data just like any other supervised data, but it is noisy; more effective approaches employ specialized probabilistic models (Riedel et al., 2010; Ratner et al., 2018a), capturing its interaction with other supervision (Wang and Poon, 2018) or breaking down aspects of a task on which it is reliable (Ratner et al., 2018b). However, these approaches often require sophisticated probabilistic inference for training of the final model. Ideally, we want a technique that handles distant data just like supervised data, so we can treat our final model and its training procedure as black boxes.

This paper tackles the problem of exploiting weakly-labeled data in a structured setting with a two-stage denoising approach. We can view a distant instance's label as a noisy version of a true underlying label. We therefore learn a model to turn a noisy label into a more accurate label, then apply it to each distant example and add the resulting denoised examples to the supervised training set. Critically, the denoising model can condition on both the example and its noisy label, allowing it to fully leverage the noisy labels, the structure of the label space, and easily learnable correspondences between the instance and the label.

Concretely, we implement our approach for the task of fine-grained entity typing, where a single entity may be assigned many labels. We learn two denoising functions: a *relabeling* function takes an entity mention with a noisy set of types and returns a cleaner set of types, closer to what manually labeled data has. A *filtering* function discards examples which are deemed too noisy to be useful. These functions are learned by taking manually-labeled training data, synthetically adding noise to it, and learning to denoise, similar to a conditional variant of a denoising autoencoder (Vincent et al., 2008). Our denoising models embed both entities and labels to make their predictions, mirroring the structure of the final entity typing model itself.

We evaluate our model following Choi et al. (2018). We chiefly focus on their ultra-fine en-

Proceedings of NAACL-HLT 2019, pages 2407–2417
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) According to the review aggregator Rotten Tomatoes , 89 % of critics gave [**the film**] positive reviews.

film, movie, show, art, entertainment, creation

(b) [**The film**] is based on a hit London and New York play , which was based on a best-selling book.

film, movie, show, art, entertainment, creation

(c) "A pretty good day all round," said [**Gascoyne** , a British veteran of stints with the original Tyrrell team] in a roller-coaster F1 career.

region

person ✔

(d) Djokovic lost to [**Rafael Nadal**] on Monday, in a rain-delayed U.S. Open final.

player, tennis player, champion, achiever, winner, contestant, person, athlete

Figure 1: Examples selected from the Ultra-Fine Entity Typing dataset of Choi et al. (2018). (a) A manually-annotated example. (b) The head word heuristic functioning correctly but missing types in (a). (c) Entity linking providing the wrong types. (d) Entity linking providing correct but incomplete types.

tity typing scenario and use the same two distant supervision sources as them, based on entity linking and head words. On top of an adapted model from Choi et al. (2018) incorporating ELMo (Peters et al., 2018), naïvely adding distant data actually hurts performance. However, when our learned denoising model is applied to the data, performance improves, and it improves more than heuristic denoising approaches tailored to this dataset. Our strongest denoising model gives a gain of 3 F_1 absolute over the ELMo baseline, and a 4.4 F_1 improvement over naive incorporation of distant data. This establishes a new state-of-the-art on the test set, outperforming concurrently published work (Xiong et al., 2019) and matching the performance of a BERT model (Devlin et al., 2018) on this task. Finally, we show that denoising helps even when the label set is projected onto the OntoNotes label set (Hovy et al., 2006; Gillick et al., 2014), outperforming the method of Choi et al. (2018) in that setting as well.

2 Setup

We consider the task of predicting a structured target y associated with an input x. Suppose we have high-quality labeled data of n (input, target) pairs $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$, and noisily labeled data of n' (input, target) pairs $\mathcal{D}' = \{(x^{(1)}, y^{(1)}_{\text{noisy}}), \dots, (x^{(n')}, y^{(n')}_{\text{noisy}})\}$. For our tasks, $\mathcal{D}$ is collected through manual annotation and $\mathcal{D}'$ is collected by distant supervision. We use two models to denoise data from $\mathcal{D}'$: a filtering function f disposes of unusable data (e.g., mislabeled examples) and a relabeling function g transforms the noisy target labels y_{noisy} to look more like true labels. This transformation improves the noisy data so that we can use it to $\mathcal{D}$ without introducing damaging amounts of noise. In the second stage, a classification model is trained on the augmented data ($\mathcal{D}$ combined with denoised $\mathcal{D}'$) and predicts y given x in the inference phase.

2.1 Case Study: Ultra-Fine Entity Typing

The primary task we address here is the fine-grained entity typing task of Choi et al. (2018). Instances in the corpus are assigned types from a vocabulary of more than 10,000 types, which are divided into three classes: 9 *general* types, 121 *fine-grained* types, and $10,201$ *ultra-fine* types. This dataset consists of 6K manually annotated examples and approximately 25M distantly-labeled examples. 5M examples are collected using entity linking (EL) to link mentions to Wikipedia and gather types from information on the linked pages. 20M examples (HEAD) are generated by extracting nominal head words from raw text and treating these as singular type labels.

Figure 1 shows examples from these datasets which illustrate the challenges in automatic annotation using distant supervision. The manually-annotated example in (a) shows how numerous the gold-standard labeled types are. By contrast, the HEAD example (b) shows that simply treating the head word as the type label, while correct in this case, misses many valid types, including more general types. The EL example (c) is incorrectly annotated as `region`, whereas the correct coarse type is actually `person`. This error is characteristic of entity linking-based distant supervision since identifying the correct link is a challenging problem in and of itself (Milne and Witten, 2008): in this case, *Gascoyne* is also the name of a region in Western Australia. The EL example in (d) has reasonable types; however, human annotators could choose more types (grayed out) to describe the mention more precisely. The average number of types annotated by humans is 5.4 per example while the two distant supervision techniques combined yields 1.5 types per example on average.

In summary, distant supervision can (1) produce

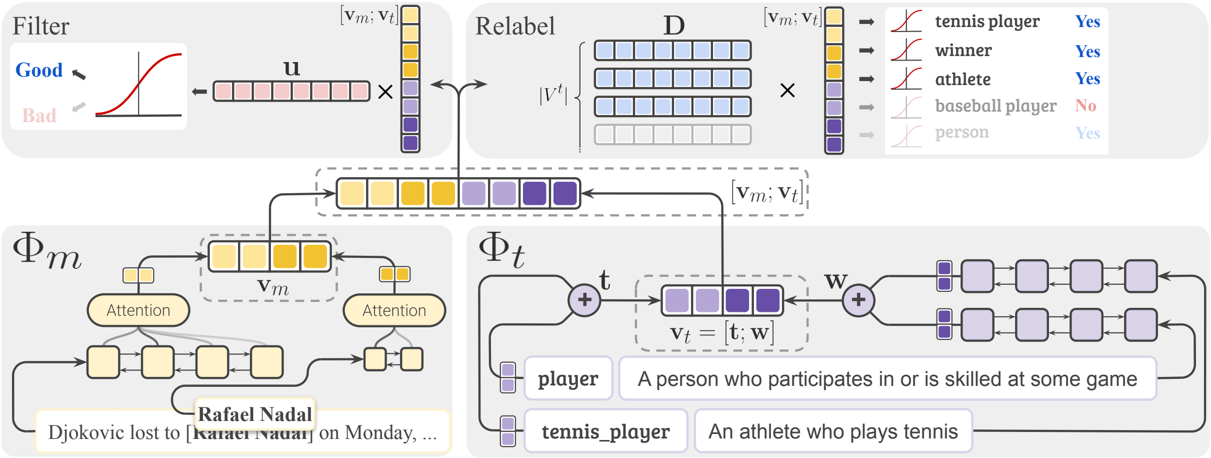


Figure 2: Denoising models. The *Filter* model predicts whether the example should be kept at all; if it is kept, the *Relabel* model attempts to automatically expand the label set. Φ_m is a mention encoder, which can be a state-of-the-art entity typing model. Φ_t encodes noisy types from distant supervision.

completely incorrect types, and (2) systematically miss certain types.

3 Denoising Model

To handle the noisy data, we propose to learn a denoising model as shown in Figure 2. This denoising model consists of filtering and relabeling functions to discard and relabel examples, respectively; these rely on a shared mention encoder and type encoder, which we describe in the following sections. The filtering function is a binary classifier that takes these encoded representations and predicts whether the example is good or bad. The relabeling function predicts a new set of labels for the given example.

We learn these functions in a supervised fashion. Training data for each is created through synthetic noising processes applied to the manually-labeled data, as described in Sections 3.3 and 3.4.

For the entity typing task, each example (x, y) takes the form $((s, m), t)$, where s is the sentence, m is the mention span, and t is the set of types (either clean or noisy).

3.1 Mention Encoder

This encoder is a function $\Phi_m(s, m)$ which maps a sentence s and mention m to a real-valued vector $\mathbf{v}_m$. This allows the filtering and relabeling function to recognize inconsistencies between the given example and the provided types. Note that these inputs s and m are the same as the inputs for the supervised version of this task; we can therefore share an encoder architecture between our denoising model and our final typing model. We use an encoder following Choi et al. (2018) with a few key differences, which are described in Section 4.

3.2 Type Encoder

The second component of our model is a module which produces a vector $\mathbf{v}_t = \Phi_t(t)$. This is an encoder of an unordered bag of types. Our basic type encoder uses trainable vectors as embeddings for each type and combines these with summing. That is, the noisy types $t_1, \ldots, t_m$ are embedded into type vectors $\{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$. The final embedding of the type set $\mathbf{t} = \sum_j \mathbf{t}_j$.

Type Definition Encoder Using trainable type embeddings exposes the denoising model to potential data sparsity issues, as some types appear only a few or zero times in the training data. Therefore, we also assign each type a vector based on its definition in WordNet (Miller, 1995). Even low-frequent types are therefore assigned a plausible embedding.[1]

Let w_i^j denote the ith word of the jth type's most common WordNet definition. Each w_i^j is embedded using GloVe (Pennington et al., 2014). The resulting word embedding vectors $\mathbf{w}_i^j$ are fed into a bi-LSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005), and a concatenation of the last hidden states in both directions is used as the definition representation $\mathbf{w}^j$. The final representation of the definitions is the sum over these vectors for each type: $\mathbf{w}$ =

---

[1] We found this technique to be more effective than using pretrained vectors from GloVe or ELMo. It gave small improvements on an intrinsic evaluation over not incorporating it; results are omitted due to space constraints.

$\sum_k \mathbf{w}^k$.

Our final $\mathbf{v}_t = [\mathbf{t}; \mathbf{w}]$, the concatenation of the type and definition embedding vectors.

3.3 Filtering Function

The filtering function f is a binary classifier designed to detect examples that are completely mislabeled. Formally, f is a function mapping a labeled example (s, m, t) to a binary indicator z of whether this example should be discarded or not.

In the forward computation, the feature vectors $\mathbf{v}_m$ and $\mathbf{v}_t$ are computed using the mention and type encoders. The model prediction is defined as $P(\text{error}) = \sigma\left(\mathbf{u}^\top \text{Highway}\left([\mathbf{v}_m; \mathbf{v}_t]\right)\right)$, where σ is a sigmoid function, $\mathbf{u}$ is a parameter vector, and $\text{Highway}(\cdot)$ is a 1-layer highway network (Srivastava et al., 2015). We can apply f to each distant pair in our distant dataset $\mathcal{D}'$ and discard any example predicted to be erroneous ($P(\text{error}) > 0.5$).

Training data We do not know a priori which examples in the distant data should be discarded, and labeling these is expensive. We therefore construct synthetic training data $\mathcal{D}_{\text{error}}$ for f based on the manually labeled data $\mathcal{D}$. For 30% of the examples in $\mathcal{D}$, we replace the gold types for that example with *non-overlapping* types taken from another example. The intuition for this procedure follows Figure 1: we want to learn to detect examples in the distant data like *Gascoyne* where heuristics like entity resolution have misfired and given a totally wrong label set.

Formally, for each selected example $((s, m), t)$, we repeatedly draw another example $((s', m'), t')$ from $\mathcal{D}$ until we find t'_{error} that does not have any common types with t. We then create a positive training example $((s, m, t'_{\text{error}}), z = 1)$. We create a negative training example $((s, m, t), z = 0)$ using the remaining 70% of examples. f is trained on $\mathcal{D}_{\text{error}}$ using binary cross-entropy loss.

3.4 Relabeling Function

The relabeling function g is designed to repair examples that make it through the filter but which still have errors in their type sets, such as missing types as shown in Figure 1b and 1d. g is a function from a labeled example (s, m, t) to an improved type set $\tilde{t}$ for the example.

Our model computes feature vectors $\mathbf{v}_m$ and $\mathbf{v}_t$ by the same procedure as the filtering function f. The decoder is a linear layer with parameters $\mathbf{D} \in \mathbb{R}^{|V^t| \times (d_m + d_t)}$. We compute $\mathbf{e} = \sigma\left(\mathbf{D}\left[\mathbf{v}_m; \mathbf{v}_t\right]\right)$, where σ is an element-wise sigmoid operation designed to give binary probabilities for each type.

Once g is trained, we make a prediction $\tilde{t}$ for each $(s, m, t) \in \mathcal{D}'$ and replace t by $\tilde{t}$ to create the denoised data $\mathcal{D}'_{\text{denoise}} = \{(s, m, \tilde{t}), \dots\}$. For the final prediction, we choose all types t_ℓ where $e_\ell > 0.5$, requiring at least two types to be present or else we discard the example.

Training data We train the relabeling function g on another synthetically-noised dataset $\mathcal{D}_{\text{drop}}$ generated from the manually-labeled data $\mathcal{D}$. To mimic the type distribution of the distantly-labeled examples, we take each example (s, m, t) and randomly drop each type with a fixed rate 0.7 independent of other types to produce a new type set t'. We perform this process for all examples in $\mathcal{D}$ and create a noised training set $\mathcal{D}_{\text{drop}}$, where a single training example is $((s, m, t'), t)$. g is trained on $\mathcal{D}'_{\text{drop}}$ with a binary classification loss function over types used in Choi et al. (2018), described in the next section.

One can think of g as a type of denoising autoencoder (Vincent et al., 2008) whose reconstructed types $\tilde{t}$ are conditioned on $\mathbf{v}$ as well as t.

4 Typing Model

In this section, we define the sentence and mention encoder Φ_m, which is use both in the denoising model as well as in the final prediction task. We extend previous attention-based models for this task (Shimaoka et al., 2017; Choi et al., 2018). At a high level, we have an instance encoder Φ_m that returns a vector $\mathbf{v}_m \in \mathbb{R}^{d_\Phi}$, then multiply the output of this encoding by a matrix and apply a sigmoid to get a binary prediction for each type as a probability of that type applying.

Figure 3 outlines the overall architecture of our typing model. The encoder Φ_m consists of four vectors: a sentence representation $\mathbf{s}$, a word-level mention representation $\mathbf{m}^{\text{word}}$, a character-level mention representation $\mathbf{m}^{\text{char}}$, and a head-word mention vector $\mathbf{m}^{\text{head}}$. The first three of these were employed by Choi et al. (2018). We have modified the mention encoder with an additional bi-LSTM to better encode long mentions, and additionally used the headword embedding directly in order to focus on the most critical word. These pieces use pretrained contextualized word embeddings (ELMo) (Peters et al., 2018) as input.

Pretrained Embeddings Tokens in the sentence s are converted into contextualized word vectors

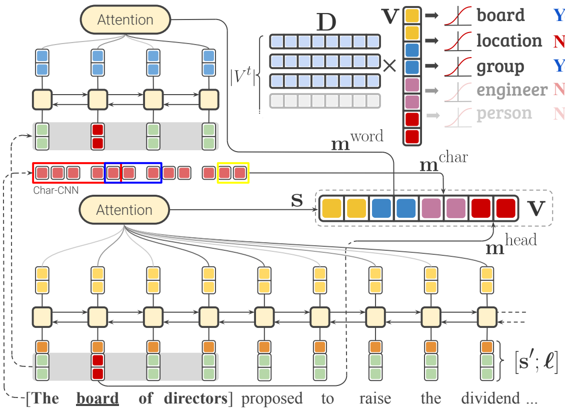


Figure 3: Sentence and mention encoder used to predict types. We compute attention over LSTM encodings of the sentence and mention, as well as using character-level and head-word representations to capture additional mention properties. These combine to form an encoding which is used to predict types.

using ELMo; let $\mathbf{s}'_i \in \mathbb{R}^{d_{ELMo}}$ denote the embedding of the ith word. As suggested in Peters et al. (2018), we learn task specific parameters $\gamma^{\text{task}} \in \mathbb{R}$ and $\mathbf{s}^{\text{task}} \in \mathbb{R}^3$ governing these embeddings. We do not fine-tune the parameters of the ELMo LSTMs themselves.

Sentence Encoder Following Choi et al. (2018), we concatenate the mth word vector $\mathbf{s}_m$ in the sentence with a corresponding location embedding $\boldsymbol{\ell}_m \in \mathbb{R}^{d_{\text{loc}}}$. Each word is assigned one of four location tokens, based on whether (1) the word is in the left context, (2) the word is the first word of the mention span, (3) the word is in the mention span (but not first), and (4) the word is in the right context. The input vectors $[\mathbf{s}'; \boldsymbol{\ell}]$ are fed into a bi-LSTM encoder, with hidden dimension is d_{hid}, followed by a span attention layer (Lee et al., 2017; Choi et al., 2018): $\mathbf{s} = \text{Attention}(\text{bi-LSTM}([\mathbf{s}'; \mathbf{l}]))$, where $\mathbf{s}$ is the final representation of the sentence s.

Mention Encoder To obtain a mention representation, we use both word and character information. For the word-level representation, the mention's contextualized word vectors $\mathbf{m}'$ are fed into a bi-LSTM with hidden dimension is d_{hid}. The concatenated hidden states of both directions are summed by a span attention layer to form the word-level mention representation: $\mathbf{m}^{\text{word}} = \text{Attention}(\text{bi-LSTM}(\mathbf{m}'))$.

Second, a character-level representation is computed for the mention. Each character is embedded and then a 1-D convolution (Collobert et al., 2011) is applied over the characters of the mention. This gives a character vector $\mathbf{m}^{\text{char}}$.

Finally, we take the contextualized word vector of the headword $\mathbf{m}^{\text{head}}$ as a third component of our representation. This can be seen as a residual connection (He et al., 2016) specific to the mention head word. We find the headwords in the mention spans by parsing those spans in isolation using the spaCy dependency parser (Honnibal and Johnson, 2015). Empirically, we found this to be useful on long spans, when the span attention would often focus on incorrect tokens.

The final representation of the input x is a concatenation of the sentence, the word- & character-level mention, and the mention headword representations, $\mathbf{v} = \left[\mathbf{s}; \mathbf{m}^{\text{word}}; \mathbf{m}^{\text{char}}; \mathbf{m}^{\text{head}}\right] \in \mathbb{R}^{d_\Phi}$.

Decoder We treat each label prediction as an independent binary classification problem. Thus, we compute a score for each type in the type vocabulary V^t. Similar to the decoder of the relabeling function g, we compute $\mathbf{e} = \sigma(\mathbf{E}\mathbf{v})$, where $\mathbf{E} \in \mathbb{R}^{|V^t| \times d_\Phi}$ and $\mathbf{e} \in \mathbb{R}^{|V^t|}$. For the final prediction, we choose all types t_ℓ where $e_\ell > 0.5$. If none of e_ℓ is greater than 0.5, we choose $t_\ell = \arg\max \mathbf{e}$ (the single most probable type).

Loss Function We use the same loss function as Choi et al. (2018) for training. This loss partitions the labels in general, fine, and ultra-fine classes, and only treats an instance as an example for types of the class in question if it contains a label for that class. More precisely:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\text{general}} \mathbb{1}_{\text{general}}(t) + \mathcal{L}_{\text{fine}} \mathbb{1}_{\text{fine}}(t) \\ &+ \mathcal{L}_{\text{ultra-fine}} \mathbb{1}_{\text{ultra-fine}}(t),\end{aligned} \tag{1}$$

where $\mathcal{L}_{\dots}$ is a loss function for a specific type class: general, fine-grained, or ultra-fine, and $\mathbb{1}_{\dots}(t)$ is an indicator function that is active when one of the types t is in the type class. Each $\mathcal{L}_{\dots}$ is a sum of binary cross-entropy losses over all types in that category. That is, the typing problem is viewed as independent classification for each type.

Note that this loss function already partially repairs the noise in distant examples from missing labels: for example, it means that examples from HEAD do not count as negative examples for general types when these are not present. However, we show in the next section that this is not sufficient for denoising.

Implementation Details The settings of hyperparameters in our model largely follows Choi et al. (2018) and recommendations for using the pre-trained ELMo-Small model.[2] The word embedding size d_{ELMo} is 1024. The type embedding size and the type definition embedding size are set to 1024. For most of other model hyperparameters, we use the same settings as Choi et al. (2018): $d_{\text{loc}} = 50$, $d_{\text{hid}} = 100$, $d_{\text{char}} = 100$. The number of filters in the 1-d convolutional layer is 50. Dropout is applied with $p = 0.2$ for the pretrained embeddings, and $p = 0.5$ for the mention representations. We limit sentences to 50 words and mention spans to 20 words for computational reasons. The character CNN input is limited to 25 characters; most mentions are short, so this still captures subword information in most cases. The batch size is set to 100. For all experiments, we use the Adam optimizer (Kingma and Ba, 2014). The initial learning rate is set to 2e-03. We implement all models[3] using PyTorch. To use ELMo, we consult the AllenNLP source code.

5 Experiments

Ultra-Fine Entity Typing We evaluate our approach on the ultra-fine entity typing dataset from Choi et al. (2018). The 6K manually-annotated English examples are equally split into the training, development, and test examples by the authors of the dataset. We generate synthetically-noised data, $\mathcal{D}_{\text{error}}$ and $\mathcal{D}_{\text{drop}}$, using the 2K training set to train the filtering and relabeling functions, f and h. We randomly select 1M EL and 1M HEAD examples and use them as the noisy data $\mathcal{D}'$. Our augmented training data is a combination of the manually-annotated data $\mathcal{D}$ and $\mathcal{D}'_{\text{denoised}}$.

OntoNotes In addition, we investigate if denoising leads to better performance on another dataset. We use the English OntoNotes dataset (Gillick et al., 2014), which is a widely used benchmark for fine-grained entity typing systems. The original training, development, and test splits contain 250K, 2K, and 9K examples respectively. Choi et al. (2018) created an augmented training set that has 3.4M examples. We also construct our own augmented training sets with/without denoising using our noisy data $\mathcal{D}'$, using the same label mapping from ultra-fine types to OntoNotes types described in Choi et al. (2018).

5.1 Ultra-Fine Typing Results

We first compare the performance of our approach to several benchmark systems, then break down the improvements in more detail. We use the model architecture described in Section 4 and train it on the different amounts of data: manually labeled only, naive augmentation (adding in the raw distant data), and denoised augmentation. We compare our model to Choi et al. (2018) as well as to BERT (Devlin et al., 2018), which we fine-tuned for this task. We adapt our task to BERT by forming an input sequence "`[CLS] sentence [SEP] mention [SEP]`" and assign the segment embedding A to the sentence and B to the mention span.[4] Then, we take the output vector at the position of the `[CLS]` token (i.e., the first token) as the feature vector **v**, analogous to the usage for sentence pair classification tasks. The BERT model is fine-tuned on the 2K manually annotated examples. We use the pretrained BERT-Base, uncased model[5] with a step size of 2e-05 and batch size 32.

Results Table 1 compares the performance of these systems on the development set. Our model with no augmentation already matches the system of Choi et al. (2018) with augmentation, and incorporating ELMo gives further gains on both precision and recall. On top of this model, adding the distantly-annotated data lowers the performance; the loss function-based approach of (Choi et al., 2018) does not sufficiently mitigate the noise in this data. However, denoising makes the distantly-annotated data useful, improving recall by a substantial margin especially in the general class. A possible reason for this is that the relabeling function tends to add more general types given finer types. BERT performs similarly to ELMo with denoised distant data. As can be seen in the performance breakdown, BERT gains from improvements in recall in the fine class.

Table 2 shows the performance of all settings on the test set, with the same trend as the performance on the development set. Our approach outperforms the concurrently-published Xiong et al. (2019); however, that work does not use ELMo. Their improved model could be used for both de-

---

[2] https://allennlp.org/elmo

[3] The code for experiments is available at https://github.com/yasumasaonoe/DenoiseET

[4] We investigated several approaches, including taking the head word piece from the last layer and using that for classification (more closely analogous to what Devlin et al. (2018) did for NER), but found this one to work best.

[5] https://github.com/google-research/bert

Model	Total			General			Fine			Ultra-Fine		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Ours + GloVe w/o augmentation	46.4	23.3	31.0	57.7	65.5	61.4	41.3	31.3	35.6	42.4	9.2	15.1
Ours + ELMo w/o augmentation	**55.6**	28.1	37.3	**69.3**	77.3	73.0	**47.9**	35.4	40.7	**48.9**	12.6	20.0
Ours + ELMo w augmentation	55.2	26.4	35.7	69.4	72.0	70.7	46.6	38.5	42.2	48.7	10.3	17.1
Ours + ELMo w augmentation + filter & relabel	50.7	**33.1**	**40.1**	66.9	**80.7**	73.2	41.7	46.2	43.8	45.6	**17.4**	**25.2**
BERT-Base, Uncased	51.6	32.8	**40.1**	67.4	80.6	**73.4**	41.6	**54.7**	**47.3**	46.3	15.6	23.4
Choi et al. (2018) w augmentation	48.1	23.2	31.3	60.3	61.6	61.0	40.4	38.4	39.4	42.8	8.8	14.6

Table 1: Macro-averaged P/R/F1 on the dev set for the entity typing task of Choi et al. (2018) comparing various systems. ELMo gives a substantial improvement over baselines. Over an ELMo-equipped model, data augmentation using the method of Choi et al. (2018) gives no benefit. However, our denoising technique allow us to effectively incorporate distant data, matching the results of a BERT model on this task (Devlin et al., 2018).

noising as well as prediction in our setting, and we believe this would stack with our approach.

Usage of Pretrained Representations Our model with ELMo trained on denoised data matches the performance of the BERT model. We experimented with incorporating distant data (raw and denoised) in BERT, but the fragility of BERT made it hard to incorporate: training for longer generally caused performance to go down after a while, so the model cannot exploit large external data as effectively. Devlin et al. (2018) prescribe training with a small batch size and very specific step sizes, and we found the model very sensitive to these hyperparameters, with only 2e-05 giving strong results. The ELMo paradigm of incorporating these as features is much more flexible and modular in this setting. Finally, we note that our approach could use BERT for denoising as well, but this did not work better than our current approach. Adapting BERT to leverage distant data effectively is left for future work.

5.1.1 Comparing Denoising Models

We now explicitly compare our denoising approach to several baselines. For each denoising method, we create the denoised EL, HEAD, and EL & HEAD dataset and investigate performance on these datasets. Any denoised dataset is combined with the 2K manually-annotated examples and used to train the final model.

Heuristic Baselines These heuristics target the same factors as our filtering and relabeling functions in a non-learned way.

Synonyms and Hypernyms For each type observed in the distant data, we add its synonyms and hypernyms using WordNet (Miller, 1995). This is motivated by the data construction process in Choi et al. (2018).

Model	P	R	F1
Ours + GloVe w/o augmentation	47.6	23.3	31.3
Ours + ELMo w/o augmentation	**55.8**	27.7	37.0
Ours + ELMo w augmentation	55.5	26.3	35.7
Ours + ELMo w augmentation + filter & relabel	51.5	**33.0**	**40.2**
BERT-Base, Uncased	51.6	**33.0**	**40.2**
Choi et al. (2018) w augmentation	47.1	24.2	32.0
LabelGCN (Xiong et al., 2019)	50.3	29.2	36.9

Table 2: Macro-averaged P/R/F1 on the test set for the entity typing task of Choi et al. (2018). Our denoising approach gives substantial gains over naive augmentation and matches the performance of a BERT model.

Common Type Pairs We use type pair statistics in the manually labeled training data. For each base type that we observe in a distant example, we add any type which is seen more than 90% of the time the base type occurs. For instance, the type `art` is given at least 90% of the times the `film` type is present, so we automatically add `art` whenever `film` is observed.

Overlap We train a model on the manually-labeled data only, then run it on the distantly-labeled data. If there is an intersection between the noisy types t and the predicted type $\hat{t}$, we combine them and use as the expanded type $\tilde{t}$. Inspired by tri-training (Zhou and Li, 2005), this approach adds "obvious" types but avoids doing so in cases where the model has likely made an error.

Results Table 3 compares the results on the development set. We report the performance on each of the EL & HEAD, EL, and HEAD dataset. On top of the baseline Original, adding synonyms and hypernyms by consulting external knowledge does not improve the performance. Expanding labels with the Pair technique results in small gains over Original. Overlap is the most ef-

Type	Denoising Method	EL & HEAD			EL			HEAD		
		P	R	F1	P	R	F1	P	R	F1
Heuristic Baselines	Raw Data	**55.2**	26.4	35.7	52.3	26.1	34.8	**52.8**	28.4	36.9
	Synonyms & Hypernyms	43.0	30.0	35.3	47.5	26.3	33.9	44.8	31.7	37.1
	Pair	50.2	29.0	36.8	49.6	27.0	35.0	50.6	31.2	38.6
	Overlap	50.0	32.3	39.2	49.5	**30.8**	38.0	50.6	31.4	38.7
Proposed Approach	Filter	53.1	28.2	36.8	51.9	26.5	35.1	51.2	31.2	38.7
	Relabel	52.1	32.2	39.8	50.2	31.4	38.6	50.2	31.8	38.9
	Filter & Relabel	50.7	**33.1**	**40.1**	**52.7**	30.5	**38.7**	50.7	**32.1**	**39.3**
	Choi et al. (2018)	48.1	23.2	31.3	50.3	19.6	28.2	48.4	22.3	30.6

Table 3: Macro-averaged P/R/F1 on the dev set for the entity typing task of Choi et al. (2018) with various types of augmentation added. The customized loss from Choi et al. (2018) actually causes a decrease in performance from adding any of the datasets. Heuristics can improve incorporation of this data: a relabeling heuristic (Pair) helps on HEAD and a filtering heuristic (Overlap) is helpful in both settings. However, our trainable filtering and relabeling models outperform both of these techniques.

fective heuristic technique. This simple filtering and expansion heuristic improves recall on EL. Filter, our model-based example selector, gives similar improvements to Pair and Overlap on the HEAD setting, where filtering noisy data appears to be somewhat important.[6] Relabel and Overlap both improve performance on both EL and HEAD while other methods do poorly on EL. Combining the two model-based denoising techniques, Filter & Relabel outperforms all the baselines.

5.2 OntoNotes Results

We compare our different augmentation schemes for deriving data for the OntoNotes standard as well. Table 4 lists the results on the OntoNotes test set following the adaptation setting of Choi et al. (2018). Even on this dataset, denoising significantly improves over naive incorporation of distant data, showing that the denoising approach is not just learning quirks of the ultra-fine dataset. Our augmented set is constructed from 2M seed examples while Choi et al. (2018) have a more complex procedure for deriving augmented data from 25M examples. Ours (total size of 2.1M) is on par with their larger data (total size of 3.4M), despite having 40% fewer examples. In this setting, BERT still performs well but not as well as our model with augmented training data.

One source of our improvements from data augmentation comes from additional data that is able to be used because *some* OntoNotes type can be derived. This is due to denoising doing a better job of providing correct general types. In the EL setting, this yields 730k usable examples out of 1M (vs 540K for no denoising), and in HEAD, 640K out of 1M (vs. 73K).

Model	Acc.	Ma-F1	Mi-F1
Ours + ELMo w/o augmentation	42.7	72.7	66.7
Ours + ELMo w augmentation	59.3	76.5	70.7
Ours + ELMo w augmentation + filter & relabel	63.9	**84.5**	78.9
Ours + ELMo w augmentation by Choi et al. (2018)	**64.9**	**84.5**	**79.2**
BERT-Base, Uncased	51.8	76.6	69.1
Shimaoka et al. (2017)	51.7	70.9	64.9
AFET (Ren et al., 2016a)	55.1	71.1	64.7
PLE (Ren et al., 2016b)	57.2	71.5	66.1
Choi et al. (2018)	59.5	76.8	71.8
LabelGCN (Xiong et al., 2019)	59.6	77.8	72.2

Table 4: Test results on OntoNotes. Denoising helps substantially even in this reduced setting. Using fewer distant examples, we nearly match the performance using the data from Choi et al. (2018) (see text).

5.3 Analysis of Denoised Labels

To understand what our denoising approach does to the distant data, we analyze the behavior of our filtering and relabeling functions. Table 5 reports the average numbers of types added/deleted by the relabeling function and the ratio of examples discarded by the filtering function.

Overall, the relabeling function tends to add more and delete fewer number of types. The HEAD examples have more general types added than the EL examples since the noisy HEAD labels are typically finer. Fine-grained types are added to both EL and HEAD examples less frequently. Ultra-fine examples are frequently added to both datasets, with more added to EL; the noisy EL labels are mostly extracted from Wikipedia defini-

---

[6] One possible reason for this is identifying stray word senses; *film* can refer to the physical photosensitive object, among other things.

	General		Fine		Ultra-Fine		
Data	Add	Del	Add	Del	Add	Del	Filter (%)
EL	0.87	0.01	0.36	0.17	2.03	0.12	9.4
HEAD	1.18	0.00	0.51	0.01	1.15	0.16	10.0

Table 5: The average number of types added or deleted by the relabeling function per example. The right-most column shows that the rate of examples discarded by the filtering function.

tions, so those labels often do not include ultra-fine types. The filtering function discards similar numbers of examples for the EL and HEAD data: 9.4% and 10% respectively.

Figure 4 shows examples of the original noisy labels and the denoised labels produced by the relabeling function. In example (a), taken from the EL data, the original labels, {`location, city`}, are correct, but human annotators might choose more types for the mention span, **Minneapolis**. The relabeling function retains the original types about the geography and adds ultra-fine types about administrative units such as {`township, municipality`}. In example (b), from the HEAD data, the original label, {`dollar`}, is not so expressive by itself since it is a name of a currency. The labeling function adds coarse types, {`object, currency`}, as well as specific types such as {`medium of exchange, monetary unit`}. In another EL example (c), the relabeling function tries to add coarse and fine types but struggles to assign multiple diverse *ultra-fine* types to the mention span **Michelangelo**, possibly because some of these types rarely cooccur (`painter` and `poet`).

6 Related Work

Past work on denoising data for entity typing has used multi-instance multi-label learning (Yaghoobzadeh and Schütze, 2015, 2017; Murty et al., 2018). One view of these approaches is that they delete noisily-introduced labels, but they cannot add them, or filter bad examples. Other work focuses on learning type embeddings (Yogatama et al., 2015; Ren et al., 2016a,b); our approach goes beyond this in treating the label set in a structured way. The label set of Choi et al. (2018) is distinct in not being explicitly hierarchical, making past hierarchical approaches difficult to apply.

Denoising techniques for distant supervision have been applied extensively to relation extraction. Here, multi-instance learning and probabilistic graphical modeling approaches have been used (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Takamatsu et al., 2012) as well as deep models (Lin et al., 2016; Feng et al., 2017; Luo et al., 2017; Lei et al., 2018; Han et al., 2018), though these often focus on incorporating signals from other sources as opposed to manually labeled data.

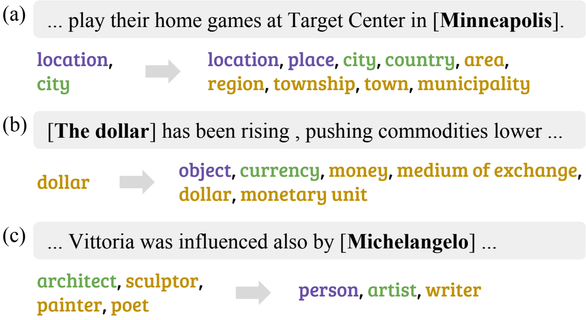


Figure 4: Examples of the noisy labels (left) and the denoised labels (right) for mentions (bold). The colors correspond to type classes: *general* (purple), *fine-grained* (green), and *ultra-fine* (yellow).

7 Conclusion

In this work, we investigated the problem of denoising distant data for entity typing tasks. We trained a filtering function that discards examples from the distantly labeled data that are wholly unusable and a relabeling function that repairs noisy labels for the retained examples. When distant data is processed with our best denoising model, our final trained model achieves state-of-the-art performance on an ultra-fine entity typing task.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, NSF Grant SHF-1762299, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. Thanks as well to the anonymous reviewers for their thoughtful comments, members of the UT TAUR lab and Pengxiang Cheng for helpful discussion, and Eunsol Choi for providing the full datasets and useful resources.

References

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. FINET: Context-Aware Fine-Grained Named Entity Typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective Deep Memory Networks for Distant Supervised Relation Extraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-Dependent Fine-Grained Entity Type Tagging. *CoRR*, abs/1412.1820.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Denoising Distant Supervision for Relation Extraction via Instance-Level Adversarial Training. *CoRR*, abs/1805.10959.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training Classifiers with Natural Language Explanations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Kai Lei, Daoyuan Chen, Yaliang Li, Nan Du, Min Yang, Wei Fan, and Ying Shen. 2018. Cooperative Denoising for Distantly Supervised Relation Extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with Noise: Enhance Distantly Supervised Relation Extraction with Dynamic Transition Matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Maxim Rabinovich and Dan Klein. 2017. Fine-Grained Entity Typing with High-Multiplicity Assignments. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2018a. Snorkel: Rapid Training Data Creation with Weak Supervision. In *Proceedings of VLDB*.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2018b. Training Complex Models with Multi-Task Weak Supervision. *CoRR*, abs/1810.02840.

Alexander Ratner, Chris De Sa, Sen Wu, Daniel Selsam, , and Christopher Ré. 2016. Data Programming: Creating Large Training Sets, Quickly. In *Proceedings of NeurIPS*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016b. Label Noise Reduction in Entity Typing by Heterogeneous Partial-Label Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Machine Learning and Knowledge Discovery in Databases*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural Architectures for Fine-grained Entity Type Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway Networks. *CoRR*, abs/1505.00387.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing Wrong Labels in Distant Supervision for Relation Extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*.

Hai Wang and Hoifung Poon. 2018. Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level Fine-grained Entity Typing Using Contextual Information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding Methods for Fine Grained Entity Type Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Trans. Knowl. Data Eng.*, 17(11):1529–1541.

A Simple and Robust Approach to Detecting Subject–Verb Agreement Errors

Simon Flachs[1,2], **Ophélie Lacroix**[1],
Marek Rei[3], **Helen Yannakoudakis**[3], **Anders Søgaard**[2]
[1] Siteimprove, Denmark
[2] CoAStaL DIKU, Department of Computer Science, University of Copenhagen, Denmark
[3] The ALTA Institute, Dept. of CS & Technology, University of Cambridge, United Kingdom
{sfl, ola}@siteimprove.com,
{marek.rei, helen.yannakoudakis}@cl.cam.ac.uk,
soegaard@di.ku.dk

Abstract

While rule-based detection of subject–verb agreement (SVA) errors is sensitive to syntactic parsing errors and irregularities and exceptions to the main rules, neural sequential labelers have a tendency to overfit their training data. We observe that rule-based *error generation* is less sensitive to syntactic parsing errors and irregularities than error detection and explore a simple, yet efficient approach to getting the best of both worlds: We train neural sequential labelers on the combination of large volumes of silver standard data, obtained through rule-based error generation, and gold standard data. We show that our simple protocol leads to more robust detection of SVA errors on both in-domain and out-of-domain data, as well as in the context of other errors and long-distance dependencies; and across four standard benchmarks, the induced model on average achieves a new state of the art.

1 Introduction

Grammatical Error Detection. Grammatical Error Detection (GED, Leacock et al., 2010) is the task of detecting grammatical errors in text. It is used in various real-world applications, such as writing assistance tools, self-assessment frameworks and language tutoring systems, facilitating incremental and/or exploratory editing of one's writing. Accurate error detection systems also have potential applications for language generation and machine translation systems, guiding automatically generated output towards grammatically correct sequences.

The problem of detecting subject–verb agreement (SVA) errors is an important subtask of GED. In this work, we focus on detecting subject–verb agreement errors in the English as a Second Language (ESL) domain. Most SVA errors occur at the third-person present tense when determining whether the subject describes a singular or a plural concept. The following examples demonstrate subject–verb agreement errors (bold):

(1) a. *They all **knows** where the conference is.
    b. *The Hotel **are** very close to Town Hall.

The task can be formulated as a sequence labeling problem, with the goal of labeling subject–verb pairs as being in agreement or not.

Approaches. Sequence labeling problems in NLP, including GED and the subtask of identifying SVA errors, have, in recent years, been handled with Recurrent Neural Networks (RNNs) trained on large amounts of data (Rei and Yannakoudakis, 2016, 2017). However, most publicly available datasets for GED are relatively small, making it difficult to learn a general grammar representation and potentially leading to over-fitting. Previous work has also shown that neural language models with a similar architecture have difficulty learning subject–verb agreement patterns in the presence of agreement attractors (Linzen et al., 2016).

Rule-based approaches (Andersen et al., 2013) are still considered a strong alternative to end-to-end neural networks, with many industry solutions still relying on rules defined over syntactic trees. The rule-based approach has the advantage of not requiring manual annotation, while also allowing easy access to adding and removing individual rules. On the other hand, language is continuously evolving, and there are exceptions to most grammar rules we know. Additionally, rule-based matching typically relies on syntactic pre-processing, which is error-prone, leading to compounding errors that hurt the downstream GED performance.

Our contributions. In this work, we compare the performance of rule-based approaches and

Proceedings of NAACL-HLT 2019, pages 2418–2427
Minneapolis, Minnesota, June 2 - June 7, 2019.

end-to-end neural models for the detection of SVA errors. We show that rule-based systems are vulnerable to errors in the underlying syntactic parsers, while also failing to capture irregularities and exceptions. In contrast, end-to-end neural architectures are limited by the available labeled examples and sensitive to the variance in these datasets. We then make the following observation: while rule-based error *detection* is severely affected by errors and irregularities in syntactic parsing, rule-based error *generation* is more robust. SVA errors can be generated without identifying subject dependency relations in advance, and changing the number of a verb almost always leads to an error. This generated data can be used as a silver standard for optimizing neural sequence labeling models. We demonstrate that a system trained on a combination of available labeled data and large volumes of silver standard data outperforms both neural and rule-based baselines by a margin on three out of four standard benchmarks, and on average achieves a new state-of-the-art on detecting SVA errors.

2 Related work

Neural approaches. Recent neural approaches to GED include Rei and Yannakoudakis (2016) who argue that bidirectional (bi-) LSTMs, in particular, are superior to other RNNs when evaluated on standard ESL benchmarks for GED and give state-of-the-art results. Rei and Yannakoudakis (2017) show even better performance using a multi-task learning architecture for training bi-LSTMs that additionally predicts linguistic properties of words, such as their part of speech (PoS).

Recent studies (Linzen et al., 2016; Gulordava et al., 2018; Kuncoro et al., 2018) have specifically analyzed the performance of LSTMs in learning syntax-sensitive dependencies such as SVA.

Rule-based approaches. Cai et al. (2009) use a combination of dependency parsing and sentence simplification, as well as special handling of *wh*-elements, to detect SVA errors. Once the subject–verb relation is identified, after parsing the simplified input sentence, a PoS tagger is used to check agreement. This is similar in spirit to the rule-based baseline system used in our experiments below. Wang et al. (2015) use a similar approach, distinguishing between four different sentence types and using slightly different rules for each type. Their rules are, again, defined over the outputs of a dependency parser and a PoS tagger. Sun et al. (2007) use labeled data to derive rules based on dependency tree patterns.

Automatic error generation. Because of the scarcity of annotated datasets in GED, research has been carried out on creating artificial errors, where errors are injected into otherwise correct text using deterministic rules or probabilistic approaches using linguistic information (Felice and Yuan, 2014; Kasewa et al., 2018). Studies focusing on detecting specific error types such as determiners and prepositions (Rozovskaya and Roth, 2011) or noun number (Brockett et al., 2006) are mainly developed within the framework of automatic error generation. Recent work, expanding the detection (Rei et al., 2017) and the correction (Xie et al., 2018) tasks to all types of errors, improves the performance of neural models by training on additional artificial error data generated via machine translation methods.

Miscellaneous. Recent work has also led to good performance in correcting grammatical errors (Yannakoudakis et al., 2017; Bryant and Briscoe, 2018; Chollampatt and Ng, 2018). However, in this paper, we are interested in the task of grammatical error *detection* and we therefore compare our work to current state-of-the-art approaches to detecting errors and do not report the performance of correction systems.

3 Subject–verb agreement detection

Following recent work on GED (Rei and Yannakoudakis, 2016), we define SVA error detection as a sequence labeling task, where each token is simply labeled as correct or incorrect. For a given SVA error, only the verb is labeled as incorrect. Error types other than SVA are ignored, i.e., we do not correct the errors in the text and we do not attempt to predict them as incorrect.

In this paper, we only study SVA in English. We note that even for English, there is some controversy about what constitutes an SVA error. Manaster-Ramer (1987), cites this example, which has been used by some as an argument for English exhibiting cross-serial dependencies:

(2) The man and the women dance and sing, respectively.

We also note that subject–verb agreement can be more or less pervasive across languages, depending on how rich the morphology is, whether the given language exhibits *pro-drop*, and how far apart subjects and verbs are likely to occur.

4 Systems

4.1 Rule-based system

Typically, building a GED rule-based system is time-consuming and requires specific knowledge to deal with the multiple exceptions and irregularities of languages. Difficult cases (such as long distance subject–verb relations) are often ignored in order to ensure high precision, at the expense of the recall of the system. However, our rule-based system is not limited to the detection of simple cases of SVA errors. It relies on PoS tags and dependency relations to identify all types of SVA errors. Specifically, our rule-based system operates as follows: (i) it identifies the candidate verbs based on PoS tags;[1] (ii) for a given verb, it uses the dependency relations to find its subject;[2] (iii) the PoS tag of the verb and its subject are used to check whether they agree in number and person. We use predicted Penn Treebank PoS tags and dependency relations provided by the Stanford Log-linear PoS Tagger (Toutanova et al., 2003) and the Stanford Neural Network Dependency Parser (Chen and Manning, 2014) respectively.

4.2 Neural system

We use the state-of-the-art neural sequence labeling architecture for error detection (Rei and Yannakoudakis, 2016). The model receives a sequence of tokens $(w_1, ..., w_T)$ as input and outputs a sequence of labels $(l_1, ..., l_T)$, i.e., one for each token, indicating whether a token is grammatically correct (in agreement) or not, in the given context. All tokens are first mapped to distributed word representations, pre-trained using word2vec (Mikolov et al., 2013) on the Google News corpus. Following Lample et al. (2016), character-based representations are also built for every word using a bi-LSTM (Hochreiter and Schmidhuber, 1997) and then concatenated onto the word embedding.

The combined embeddings are then given as input to a word-level bi-LSTM, creating representations that are conditioned on the context from both sides of the target word. These representations are then passed through an additional feed-forward layer, in order to combine the extracted features and map them to a more suitable space. A softmax output layer returns the probability distribution over the two possible labels (*correct* or *incorrect*) for each word. We also include the language modeling objective proposed by Rei (2017), which encourages the model to learn better representations via multi-tasking and predicting surrounding words in the sentence. Dropout (Srivastava et al., 2014) with probability 0.5 is applied to word representations and to the output from the word-level bi-LSTM. The model is optimised using categorical cross-entropy with AdaDelta (Zeiler, 2012).

5 Data

5.1 Data preprocessing

As the public datasets either have their own taxonomy or they are not annotated with error types at all, we apply the error type extraction tool of Bryant, Felice, and Briscoe (2017) to automatically get error types mapped to the same taxonomy for all datasets. The tool automatically annotates parallel original and corrected sentences with error type information. When evaluated by human raters, the predicted error types were rated as "good" or "acceptable" in at least 95% of the cases. We use their publicly available tool[3] to automatically get error types for all public datasets mapped to the same taxonomy of 25 error types in total. We then set SVA errors as our target class.

5.2 Test data

We compare the rule-based and neural approaches for the task of SVA error detection on four benchmarks in the ESL domain.

- **FCE**. The Cambridge Learner Corpus of First Certificate in English (FCE) exam scripts consists of texts produced by ESL learners taking the FCE exam, which assesses English at the upper-intermediate proficiency level (Yannakoudakis et al., 2011). We use the publicly available test set.

- **AESW**. The dataset from the Automated Evaluation of Scientific Writing Shared Task

---

[1] Present tense verbs + "was" and "were".

[2] The subject can be direct – attached with a `nsubj` relation – or indirect, such as when the syntactic subject is a relative pronoun, e.g., *who*, or an expletive, e.g., *there*.

[3] https://github.com/chrisjbryant/errant

2016 (AESW) is a collection of text extracts from published journal articles (mostly in physics and mathematics) along with their (sentence-aligned) corrected counterparts (Daudaravicius et al., 2016). We test on the combined trained, development and test set.[4]

- **JFLEG**. The JHU Fluency-Extended GUG corpus (JFLEG) represents a cross-section of ungrammatical data, consisting of sentences written by ESL learners with different proficiency levels and L1s (Napoles et al., 2017). We evaluate our models on the public test set.
- **CoNLL14**. The test dataset from the CoNLL 2014 shared task consists of (mostly argumentative) essays written by advanced undergraduate students from the National University of Singapore, and are annotated for grammatical errors by two native speakers of English (Ng et al., 2014).

5.3 Training data

ESL writings. We use the following ESL datasets as training data:

- **Lang8** is a parallel corpus of sentences with errors and their corrected versions created by scraping the Lang-8 website[5], which is an open platform where language learners can write texts and native speakers of that language can provide feedback via error correction (Mizumoto et al., 2011). It contains $1,047,393$ sentences.
- **NUCLE** comprises around $1,400$ essays written by students from the National University of Singapore. It is annotated for error tags and corrections by professional English instructors (Dahlmeier et al., 2013). It contains $57,151$ sentences.
- **FCE train set**. We use the publicly available FCE training set, containing $25,748$ sentences. A subset of $5,000$ sentences was separated and used for development experiments.

Artificial errors. We generate artificial subject–verb agreement errors from large amounts of data. Specifically, we use the British National Corpus (BNC, BNC-Consortium et al., 2007), a collection of British English sentences that includes samples from different media such as newspapers, journals, letters or essays. Subject–verb agreement in English merely consists of inflecting 3rd person singular verbs in the present tense (and *be* in the past), which makes any text in English fairly easy to corrupt with SVA errors. We assume that the BNC data is written in correct British English. Using predicted PoS tags provided by the Stanford Log-linear PoS Tagger, we identify verbs in present tense, as well as *was* and *were* for the past tense, and flip them to their respective opposite version using the list of inflected English words (annotated with morphological features) from the Unimorph project (Kirov et al., 2016). The final artificial training set includes the sentences with injected errors ($265,742$ sentences), their original counterpart, and sentences where SVA errors could not be injected due to not containing candidate verbs that could be flipped ($241,295$ sentences).

6 Experiments

The models. We compare our neural model trained on both artificially generated errors and ESL data (LSTM$_{ESL+art}$) to three baselines: a neural model trained only on ESL data (LSTM$_{ESL}$) (i.e., reflecting the performance of current state-of-the-art approaches for GED), a language model based method (BERT-LM) and our rule-based system.

In order to measure the real performance of a language model (LM) on the detection of SVA errors, we choose to use the BERT system (Devlin et al., 2018) to assign probabilities to different versions of the test sentences. Specifically, we use the pre-trained uncased BERT-Base model. We duplicate the sentences each time a corruptible verb occurs (flipping its number). The LM assigns a probability to both possible versions of the verbs. We select the version which has the highest probability, if this probability is at least 0.1[6] higher than the probability of the verb in the original sentence.

---

[4] Sentences containing special placeholders for mathematical equations, dates, etc. are filtered out.

[5] http://lang-8.com/

[6] We tune the threshold on the test dataset from the CoNLL 2013 shared task on Grammatical Error Correction of ESL learner essays.

	FCE			AESW			CoNLL14			JFLEG			
System	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	$F_{0.5}$ avg.
Rules	43.75	40.23	43.00	14.82	**49.75**	17.24	27.93	31.96	28.65	37.50	**48.21**	39.24	32.03
BERT-LM	66.67	**52.87**	63.36	18.36	39.61	20.57	50.00	35.24	46.13	60.00	32.14	51.14	45.30
LSTM$_{ESL}$	71.88	26.44	53.49	**27.75**	10.33	20.75	**54.84**	17.53	38.46	**73.91**	30.91	**57.82**	42.63
LSTM$_{ESL+art}$	**72.41**	48.84	**66.04**	19.05	40.66	**21.31**	49.32	**37.11**	**46.27**	64.71	39.29	57.29	**47.73**

Table 1: Performance of our systems (rule-based and LSTMs) and baselines. BERT-LM is the language model baseline.

Hyper-parameters. We tune the model hyper-parameters on the FCE development set, according to the $F_{0.5}$ score. Training is stopped when $F_{0.5}$ on the FCE development set does not improve over 7 epochs. Word representations have size 300, while character representations have size 100. The word-level LSTM hidden layers have size 300 for each direction, and the character-level LSTM hidden layers have size 100 for each direction.

Evaluation. Existing approaches are typically optimised for high precision at the cost of recall, as a system's utility depends strongly on the ratio of true to false positives, which has been found to be more important in terms of learning effect. A high number of false positives would mean that the system often flags correct language as incorrect, and may therefore end up doing more harm than good (Nagata and Nakatani, 2010). Because of this, $F_{0.5}$ is preferred to F_1 in the GED domain as it puts more weight on precision than recall. For each experiment, we report the token-level precision (P), the recall (R), and the $F_{0.5}$ scores.

7 Results

The main results are summarized in Table 1. Looking at the performance of the LSTM$_{ESL+art}$ system, we see that on 3 out of 4 benchmarks, our neural model trained on artificially generated errors outperforms the LSTM$_{ESL}$ system with respect to $F_{0.5}$. On average, over the four benchmarks, its $F_{0.5}$ score is 2.43 points higher than the best performing baseline. Both neural models obtain higher $F_{0.5}$ scores than the rule-based baseline, on average and across the board, i.e., $+10.6$ for LSTM$_{ESL}$ and $+15.7$ for LSTM$_{ESL+Art}$. The BERT-LM outperforms the LSTM$_{ESL}$ (mostly due to its higher recall, i.e., $+18.66$) but still does not reach the $F_{0.5}$ score of the LSTM$_{ESL+Art}$ system which gets higher precision and recall overall ($+2.62$ and $+1.51$ respectively).

Furthermore, we observe a trend that the two LSTM systems trade off precision and recall, with the LSTM$_{ESL}$ system yielding the highest precision across most datasets, but also yielding significantly lower recall than LSTM$_{ESL+Art}$. It is also evident that the performance varies over domains: all models struggle with AESW. This is likely due to the complexity of the scientific writing genre where, for example, sentences contain parentheses interposed between a verb and its subject. We also note errors are far less frequent in this genre, leading to moderate recall and very low precision. For the rest of the datasets, system performance is generally better.

8 Analysis

We analyze the effect of adding artificial errors to the training data. In particular, we focus on the robustness of our models by looking at how sensitive they are to grammatical errors in the surrounding context; and by looking at how good the models are at predicting agreement relative to the distance between the subject and verb. This set of experiments is similar in spirit to Linzen et al. (2016). We also analyze our rule-based baseline: so far, we know our rule-based baseline was sensitive to parser errors and irregularities. We inspect the quality of the underlying parser by evaluating it on data that resembles the data used in our experiments, to see whether errors seem to result more from parser errors or irregularities. Finally, we also look at the sensitivity of our systems to other linguistic phenomena such as relative clauses or conjunctions.

8.1 Sensitivity to other errors in the surrounding context

In ESL writings, multiple errors can occur in the same sentence. This means more variable contexts, which can lead to degradation in the performance of both syntactic parsers / rule-based systems and GED models.

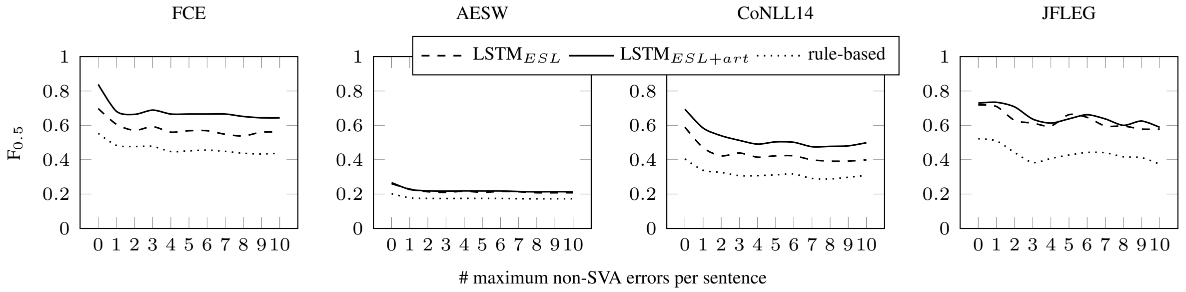


Figure 1: Performance ($F_{0.5}$ scores) of the systems with respect to the noise in test data (i.e., the number of additional non-SVA errors in sentences).

Testing on noisy contexts We first evaluate how our systems are impacted by additional non-SVA errors in the surrounding context of SVA errors in our test data. For each of the test datasets, we create multiple versions, allowing for n non-SVA errors per sentence (we correct the extra non-SVA errors). This way we can create datasets with different levels of complexity with respect to the grammatical errors within them.

In Figure 1, the $F_{0.5}$ scores of the models are shown for different numbers of grammatical errors per sentence. It is evident that all of the models are negatively affected by the presence of other errors in the same sentence. Using more data for training – i.e., our artificial training data which does not include context errors – generally boosts performance on data with and without grammatical errors in the context. In other words, training with additional artificially generated errors seems, overall, to be making our model more robust. We also note that our rule-based baseline is affected by errors to roughly the same extent as our baseline neural model is. One might have thought the rule-based baseline would suffer more, because of it being sensitive to errors in the underlying syntactic parser. We return to this issue below.

Training on non-noisy contexts In order to assess the benefit of training on non-erroneous contexts, we create a new dataset from our ESL training data (see §5.3). Based on the annotations in the data, we apply the corrections of error types other than SVA, thereby only leaving SVA errors in the data. We experiment with how adding this 'clean' dataset to the training set of our existing systems affects performance. The resulting $F_{0.5}$ scores are listed in Table 2. Using 'clean' sentences in addition to our original ESL data for training always positively affects performance. In this regard, as experimented in (Rei and Yannakoudakis, 2016), training on more data in the same domain is a valid solution for improving the performance of LSTM models. However, when also adding artificially generated data to the training set, we reach higher scores only on 2 out of the 4 benchmarks. It greatly improves the average recall (+11.03), without hurting the precision on FCE and CoNLL14 but affects negatively the precision on AESW and JFLEG.

	FCE	**AESW**	**CoNLL14**	**JFLEG**
System	$F_{0.5}$	$F_{0.5}$	$F_{0.5}$	$F_{0.5}$
LSTM$_{ESL}$	53.49	20.75	38.46	57.82
LSTM$_{ESL+art}$	66.04	21.31	46.27	57.29
LSTM$_{ESL+cor}$	65.08	**27.16**	46.26	**59.52**
LSTM$_{ESL+art+cor}$	**67.16**	21.12	**52.28**	54.64

Table 2: Performance ($F_{0.5}$ scores) of the LSTM models when trained using an additional set of 'clean' sentences (*cor*) where non-SVA errors have been corrected.

8.2 Sensitivity to long-distance dependencies

Next, we want to study how well our models perform when the subjects and verbs are far apart, i.e., when the agreement relation is defined over a long-distance dependency. In order to see how our systems are affected by the distance between the subject and verb, we split the test sets based on different subject–verb distances.

Note, however, that our benchmarks are not annotated with PoS tags and dependency relations. If we binned our test data based on predicted dependencies, the inductive bias of our syntactic parser and the errors it made would bias our evaluation. Instead, we perform our analyses on section 22 and 23 of the Penn Treebank (PTB) dataset (Marcus et al., 1993). The PTB however is not annotated with grammatical errors. We therefore corrupt the sentences by injecting SVA errors, in the same way we corrupted the BNC (§5.3) to create additional training data.

For each sentence in the PTB, we identify a subject–verb pair, and group the sentences by the subject–verb distance. We then run our models on two versions of each sentence: an unaltered version and a corrupted one, where we have generated an SVA error by corrupting the verb, using the method described earlier (§5.3). This way we can compute the performance of our models as $F_{0.5}$ scores over this dataset. The results are displayed in Figure 2. We can see that the LSTM trained with artificial data performs significantly better on long-distance subject–verb pairs than the LSTM trained only on ESL data. This suggests that training on artificially generated errors also makes our models more robust to this potential source of error.

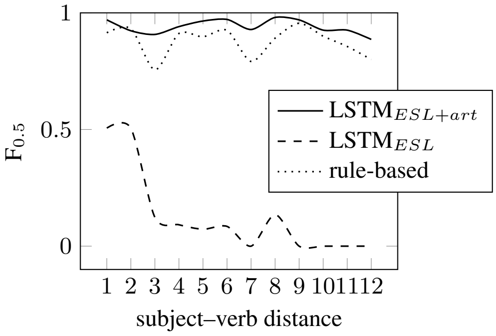


Figure 2: $F_{0.5}$ scores of the systems on the PTB as a function of subject–verb distance.

Note that, in general, there is a substantial gap between the performance of the two LSTM models. This is because one is trained on artificial data – similar to the data we use in our analysis. However, the conclusions are based on the relative differences in performance over long-distance dependencies, and these differences should still be comparable across the two models.

8.3 Sources of error for our rule-based baseline

There are two obvious potential sources of error for our rule-based baseline: sensitivity to errors in the underlying syntactic parsers, and sensitivity to the irregularities of language, e.g., when collective nouns or named entities are subjects, subject–verb agreement cannot always be determined by the PoS tags. We show that the main source of error seems to be irregularities by showing that the underlying syntactic parsers perform relatively well, even in the ESL domain.

Table 3 lists the parsing and tagging performance of our underlying syntactic parsers across three domains: learner data (ESL) and web data (EWT) from the Universal Dependencies (UD) project (Nivre et al., 2017), as well as the newswire data it was trained on (PTB). We only evaluate subject–verb relations, since these are the only ones of interest in this paper. We see that while there is a noticeable out-of-domain drop going from newswire to learner language or web data, the parser is still able to detect subject–verb relations with high precision and recall. This suggests that the vulnerability of our rule-based baseline is primarily a result of linguistic irregularities and exceptions to the implemented rules.

	UD-ESL	UD-EWT	PTB 23
Subject–verb precision	88.47	88.86	91.31
Subject–verb recall	89.37	85.11	89.84
PoS tags accuracy	96.36	93.20	97.79

Table 3: The Stanford PoS Tagger and Dependency Parser's performance on different treebanks. Subject–verb precision/recall relates to subject–verb relations. PoS tag accuracy is only for PoS tags of the subjects and verbs.

8.4 Sensitivity to other linguistic phenomena

Finally, manually reviewing the errors made by the rule-based system, we identified frequent linguistic sources of errors, including relative clauses, conjunctions, ambiguous PoS tags, and collective nouns. We therefore analyze how the LSTMs and the rule-based system are globally sensitive to these potential sources of error. Since our benchmarks are not annotated with PoS and dependency relations, we again use the corrupted PTB sentences (see §8.2).

Many of the examples in which our rule-based baseline fails include *relative clauses* (when the verb is the root of a relative clause) and *conjunctions* (when the subject is a conjunction). A second major cause of failure is ambiguous verbs, i.e., verb forms that can also be nouns (*ambiguous PoS*, e.g., "need", "stop", "point", etc.), and subjects which are singular nouns describing groups of people or things (*collective nouns*, e.g., "team", "family", "staff", etc.). The following examples illustrate these cases (underlined):

(3) a. The church and the cathedral **are** very interesting [. . .] (*conjunction*)

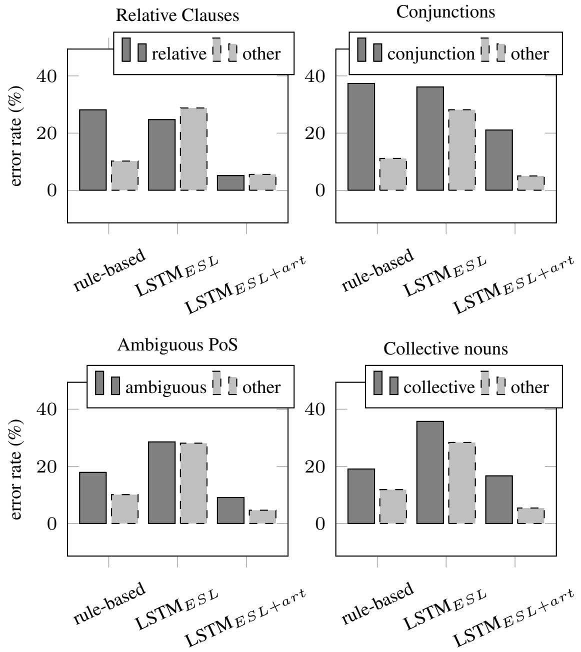


Figure 3: SVA error rates on the PTB data for complex syntactic structures and ambiguous cases.

b. If there is someone who **doesn't** agree with me, he or she [. . .] (*relative clause*)

c. It is said that the majority of the citizens **has** got a car [. . .] (*collective noun*)

d. [. . .] and police officer **walk** around the building as well. (*ambiguous PoS*)

We evaluate our models on the PTB data and report the error rate (the lower the better) on present tense verbs (Figure 3). Overall, results show that all models are negatively affected when they encounter complex syntactic structures and ambiguous cases. Figure 3 also confirms that the rule-based baseline is the most sensitive one to complex structures. Especially in comparison with the LSTM$_{ESL+art}$ model, the rule-based system achieves good scores on verbs which are not part of complex structures, but performs significantly worse on difficult cases. The LSTM$_{ESL}$ model is the worst across almost all cases, while the LSTM$_{ESL+art}$ shows significant improvements over the baselines, in particular for the difficult cases.

9 Conclusion

In this paper, we argue for artificial error generation as an effective approach to learning more robust neural models for subject–verb agreement detection. We demonstrate that error generation is much less sensitive to parsing errors and irregularities than rule-based systems for detecting subject–verb agreement. On the other hand, artificial error generation enables us to utilise much more training data, and therefore can develop more robust neural models for SVA error detection that do not overfit the available, manually annotated training data. Our simple approach to detecting subject–verb agreements achieves a new state of the art on three out of four available benchmarks, and, on average, is better than previous approaches on the task. We show that, in particular, models trained on large volumes of artificially generated errors become more robust to other errors in the surrounding context of SVA, long-distance dependencies, and other challenging linguistic phenomena.

Acknowledgements

This project was supported by Siteimprove and the Innovation Fund of Denmark through an industrial PhD grant. Marek Rei and Helen Yannakoudakis were supported by Cambridge Assessment, University of Cambridge.

References

Øistein E. Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. In *Proceedings of BEA 2013*.

BNC-Consortium et al. 2007. The British National Corpus, Version 3. *Distributed by Bodleian Libraries, University of Oxford*.

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL Errors Using Phrasal SMT Techniques. In *Proceedings of COLING-ACL 2006*.

Christopher Bryant and Ted Briscoe. 2018. Language Model Based Grammatical Error Correction without Annotated Training Data. In *Proceedings of BEA 2018*.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In *Proceedings of ACL 2017*.

Dongfeng Cai, Yonghua Hu, Xuelei Miao, and Yan Song. 2009. Dependency Grammar Based English Subject-Verb Agreement Evaluation. In *Proceedings of PACLIC 2009*.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of EMNLP 2014*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction. In *Proceedings of AAAI 2018*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of BEA 2013*.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A Report on the Automatic Evaluation of Scientific Writing Shared Task. In *Proceedings of BEA 2016*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. pages 116–126.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless Green Recurrent Networks Dream Hierarchically. In *Proceedings of NAACL 2018*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of EMNLP 2018*.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large Scale Parsing and Normalization of Wiktionary Morphological Paradigms. In *Proceedings of LREC 2016*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better. In *Proceedings of ACL 2018*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL 2016*.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated Grammatical Error Correction for Language Learners. *Synthesis lectures on human language technologies*, 3(1):1–134.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 4.

Alexis Manaster-Ramer. 1987. Subject-verb Agreement in Respective Coordinations and Context-freeness. *Computational Linguistics*, 13:64–65.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NIPS 2013*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of IJCNLP 2011*.

Ryo Nagata and Kazuhide Nakatani. 2010. Evaluating performance of grammatical error detection to maximize learning effect. In *Proceedings of COLING 2010*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of EACL 2017*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of CoNLL 2014: Shared Task*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, et al. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. *Proceedings of ACL 2017*.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial Error Generation with Machine Translation and Syntactic Patterns. In *Proceedings of BEA 2017*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of ACL 2016*.

Marek Rei and Helen Yannakoudakis. 2017. Auxiliary Objectives for Neural Error Detection Models. In *Proceedings of BEA 2017*.

Alla Rozovskaya and Dan Roth. 2011. Algorithm Selection and Model Adaptation for ESL Correction Tasks. In *Proceedings of ACL 2011*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Guihua Sun, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining Sequential Patterns and Tree Patterns to Detect Erroneous Sentences. In *Proceedings of AAAI 2007*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of NAACL 2003*.

Yuzhu Wang, Hai Zhao, and Dan Shi. 2015. A Light Rule-based Approach to English Subject-Verb Agreement Errors on the Third Person Singular Forms. In *Proceedings of PACLIC 2015*.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of NAACL 2018*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of ACL 2011*.

Helen Yannakoudakis, Marek Rei, Øistein E Andersen, and Zheng Yuan. 2017. Neural Sequence-Labelling Models for Grammatical Error Correction. In *Proceedings of EMNLP 2017*.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

A Grounded Unsupervised Universal Part-of-Speech Tagger for Low-Resource Languages

Ronald Cardenas♣ **Ying Lin**♠ **Heng Ji**♠ **Jonathan May**♡
♣ Institute of Formal and Applied Linguistics, Charles University in Prague
♠ Computer Science Department, Rensselaer Polytechnic Institute
♡ Information Sciences Institute, University of Southern California
`ronald.cardenas@matfyz.cz` `liny9@rpi.edu`
`jih@rpi.edu` `jonmay@isi.edu`

Abstract

Unsupervised part of speech (POS) tagging is often framed as a clustering problem, but practical taggers need to *ground* their clusters as well. Grounding generally requires reference labeled data, a luxury a low-resource language might not have. In this work, we describe an approach for low-resource unsupervised POS tagging that yields fully grounded output and requires no labeled training data. We find the classic method of Brown et al. (1992) clusters well in our use case and employ a decipherment-based approach to grounding. This approach presumes a sequence of cluster IDs is a 'ciphertext' and seeks a POS tag-to-cluster ID mapping that will reveal the POS sequence. We show intrinsically that, despite the difficulty of the task, we obtain reasonable performance across a variety of languages. We also show extrinsically that incorporating our POS tagger into a name tagger leads to state-of-the-art tagging performance in Sinhalese and Kinyarwanda, two languages with nearly no labeled POS data available. We further demonstrate our tagger's utility by incorporating it into a true 'zero-resource' variant of the MALOPA (Ammar et al., 2016) dependency parser model that removes the current reliance on multilingual resources and gold POS tags for new languages. Experiments show that including our tagger makes up much of the accuracy lost when gold POS tags are unavailable.

1 Introduction

While cellular, satellite, and hardware advances have ensured that sophisticated NLP technology can reach all corners of the earth, the language barrier upon reaching remote locales still remains. As an example, when international aid organizations respond to new disasters, they are often unable to deploy technology to understand local reports detailing specific events (Munro and Manning, 2012; Lewis et al., 2011). An inability to communicate with partner governments or civilian populations in a timely manner leads to preventable casualties.

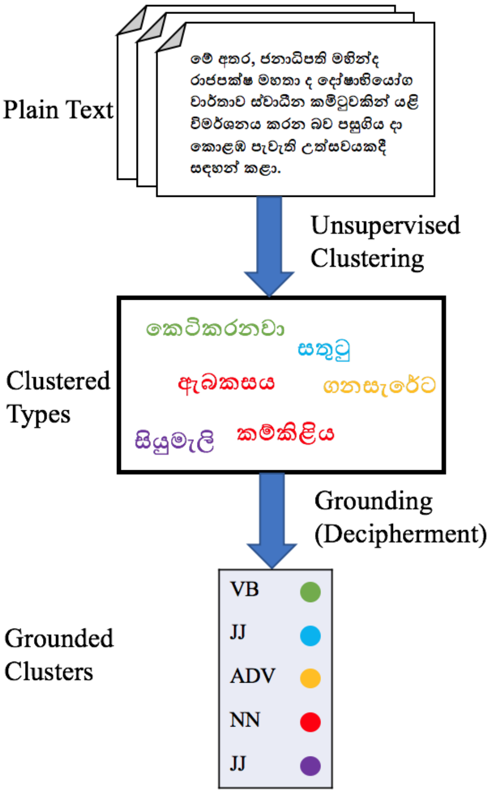


Figure 1: Overview of our approach to grounded POS tagging. We use an unsupervised clustering method (Section 3.2) then reduce and ground the clusters using a decipherment approach informed by POS tag sequence data from many languages (Section 3.3).

The lack of adequate labeled training data has been the major obstacle to expanding NLP's outreach more multilingually. Developments in unsupervised techniques that require only monolingual corpora (Lample et al., 2018a; Artetxe et al., 2018) and the ability to leverage labeled resources in other languages have been proposed to address this issue (Das and Petrov, 2011; Duong et al., 2014; Ammar et al., 2016). Unfortunately, these

Proceedings of NAACL-HLT 2019, pages 2428–2439
Minneapolis, Minnesota, June 2 - June 7, 2019.

methods either do not work in practice on true low-resource cases or unrealistically assume the availability of some amount of supervision.

Consider syntactic parsing as a prime example. Past editions of the CoNLL Shared Task on Multilingual Parsing (Zeman et al., 2017, 2018) featured a category of target languages for which either little or no training data was provided. However, even in the 'no-resource' scenario that most closely matches our use case, gold part-of-speech (POS) tags for test data were provided for the participants to use. Prior to these shared tasks, Ammar et al. (2016) proposed a variant of their main model, MALOPA, that was meant to produce reasonable parses for languages under "zero-resource" conditions. In order to function, however, the model requires users to provide gold POS tags and word mappings from these languages into a common semantic space, using approaches that require parallel data (Guo et al., 2015).

Indeed, the compulsion to use POS tag-labeled data in zero-resource circumstances extends to the vast, varied lines of research in unsupervised POS tagging itself! Every approach explored so far ultimately requires POS-annotated resources for the language being studied in order to produce a final, grounded output. Even the most conservative strategies (Goldwater and Griffiths, 2007; Berg-Kirkpatrick et al., 2010; Stratos et al., 2016) that do not require any supervised signal during training still ultimately produce only ungrounded clusters, and require a reference annotated corpus to map the inferred clusters or states to actual POS tags.

Making matters worse, evaluation is generally offered in terms of the 'many-to-one' or 'one-to-one' analyses Johnson (2007). These metrics use a reference corpus to determine the *optimal* mapping of clusters to tags. While this evaluation approach is intuitively sensible for measuring cluster purity, to actually *use* such an output, an entire annotated training corpus is required.[1] It is not enough to simply rely on ungrounded clusters in real-world systems; grounded labels offer a sort of universal API between other resources such as rule-based modules that operate on certain word types or between resources built from other annotated high-resource language data.

Since POS tag and parallel data resources for new languages are often unavailable or unreliable, we make the following contributions to ensure the surprise of a new language does not immobilize us:

- We introduce a decipherment-based approach to POS grounding, which yields fully grounded output and does not require any annotated data or parallel corpora in the language to be analyzed. The approach uses pre-existing human-labeled POS tag sequences from high-resource *parent languages* (PL) but no labeled data or sequences for the target, or *child language* (CL). An overview of the approach is shown in Figure 1.
- We demonstrate our approach by evaluating over a variety of languages spanning 4 families and 8 genera (Germanic, Romance, Slavic, Japanese, Semitic, Iranian, Indic, and Bantoid), and show across-the-board reasonable intrinsic performance, given the difficulty of the task and the stringency (straightforward accuracy) in comparison to other unsupervised evaluation strategies.
- We test the utility of our grounded tags in a name tagging task, obtaining state-of-the-art performance for Sinhalese and Kiryarwanda, two languages with nearly no labeled POS or named entity resources.
- We further pare down the annotated resources required in an existing 'zero-resource' dependency parser model and show that our unsupervised and grounded tags are helpful at closing the gap between a nihilistic tag-free setting and an unrealistic gold tag setting.
- We release our code so that others may create zero-resource syntactic analysis and information extraction systems at the onset of the next new emergency.[2]

2 POS Grounding as Decipherment

We consider the task of POS induction as a two-step pipeline: from word sequence w to POS tag sequence p via cluster sequence c. Formally, our conditional probability model is

---

[1] Additionally, Headden III et al. (2008) demonstrated that these metrics are not indicative of downstream performance.

[2] `https://github.com/isi-nlp/universal-cipher-pos-tagging.git`

$$\begin{aligned}&\underset{p}{\operatorname{argmax}} P_\theta(p|w)\\&\qquad= \underset{p}{\operatorname{argmax}} \sum_{c\in C^{|w|}} P_\theta(p,c|w)\\&\qquad= \underset{p}{\operatorname{argmax}} \sum_{c\in C^{|w|}} P_\theta(p|c,w)P_\theta(c|w)\end{aligned}$$

where C is the cluster vocabulary and θ parameterizes our probability model. If we assume a deterministic pipelined clustering of words and a tag labeling model that does not depend on words, then for chosen $\hat{c}$, this becomes

$$\begin{aligned}&\underset{p}{\operatorname{argmax}} \sum_{c\in C^{|w|}} P_\theta(p|c,w)P_\theta(c|w)\\&\qquad= \underset{p}{\operatorname{argmax}}\, P_\theta(p|\hat{c})\\&\qquad= \underset{p}{\operatorname{argmax}}\, P_\theta(\hat{c}|p)P_\theta(p) \qquad (1)\end{aligned}$$

We call this model the *cipher grounder*. As presented it requires an estimate for $P_\theta(p)$ for the CL, which requires POS training data. Under the zero-resource scenario, we instead approximate $P_\theta(p)$ by the tag distribution of a PL. Then, the *cipher table* $P_\theta(\hat{c}|p)$ can be trained using a noisy-channel, expectation-maximization (EM)-based approach as in Ravi and Knight (2011).

3 POS Tagger construction

We approach the search for optimal components in the two-step pipeline outlined in Section 2 in a cascaded manner. First, an optimal word clustering is determined by means of the many-to-one evaluation method. This method is explained well by Johnson (2007):

> “ ...deterministically map each hidden state to the POS tag it co-occurs most frequently with, and return the proportion of the resulting POS tags that are the same as the POS tags of the gold-standard corpus.”

While unrealistic for POS tagger performance purposes, many-to-one is a good choice for determining cluster ‘purity’ and provides a reasonable grounding upper bound. As the calculation of many-to-one does require labeled data, we constrain the use of these labels for development and will evaluate extrinsically using languages for which we do not have any training data; see Section 5.2.

Secondly, we search for the best approach to ground the chosen clusters, given several possible PL options.

After the optimal components and parameters are determined, we validate POS tag quality intrinsically via tag accuracy on reference data where it exists, and then extrinsically on two downstream tasks. We investigate a simulated no-resource scenarios in the task of dependency parsing, and a real low-resource scenario in name tagging.

3.1 Datasets

For intrinsic evaluation and optimization of the tagging pipeline, including all preliminary experiments, we use annotated corpora from Universal Dependencies (UD) v2.2[3] for the following languages: English (en), German (de), French (fr), Italian (it), Spanish (es), Japanese (ja), Czech (cs), Russian (ru), Arabic (ar), and Farsi (fa). For Swahili (sw), we use the Helsinki Corpus of Swahili 2.0.[4] Overall in these experiments we cover 11 languages and 4 language families.

In our dependency parsing experiments, we use the Universal Treebank v2.0 (McDonald et al., 2013) for en, de, fr, es, it, Portuguese (pt), and Swedish (sv). This set of treebanks is chosen instead of UD in order to obtain results comparable to those of previous work on simulated zero-resource parsing scenarios (Ammar et al., 2016; Zhang and Barzilay, 2015; Rasooli and Collins, 2015).

In our name tagging experiments, we use monolingual texts for Sinhalese (si) and Kinyarwanda (rw) provided by DARPA’s Low Resource Languages for Emergent Incidents (LORELEI) Program during the 2018 Low Resource Human Languages Technologies (LoReHLT) evaluation.

3.2 Unsupervised Clustering

In this step we compare two approaches to unsupervised ungrounded labeling. The first strategy is to cluster by word types and thus label each token with its cluster ID independently of its context.[5] We consider Brown’s hierarchical clustering algo-

---

[3] `http://universaldependencies.org/`

[4] `http://urn.fi/urn:nbn:fi:lb-2016011301`

[5] We refer to ungrounded POS tag labels as ‘clusters’ even though not all methods induce a clustering.

rithm, (Brown et al., 1992)[6] BROWN; Brown's exchange algorithm,[7] (Martin et al., 1998) MARLIN; and k-means clustering of monolingual word embeddings of dimension size 100, trained using *fastText* (Joulin et al., 2016), E-KMEANS. The second labeling strategy is context-sensitive; it uses the Bayesian HMM tagger proposed by Stratos et al. (2016), which we call A-HMM. As noted previously, we evaluate unsupervised labeling extrinsically, via the many-to-one approach, and use the best performing labeling in the complete two-step grounded tagging pipeline.

In preliminary experiments, we vary the number of clusters and hidden states ($|C|$) between 17 and 500. We initially sought to create one cluster per UD POS tag and then choose the proper 1:1 assignment of cluster to tag, following the approach of Stratos et al. (2016). However, cluster purity is low when only 17 clusters are allowed (i.e. each cluster has words with a variety of POS tags). Naturally, as the number of clusters is raised, the purity of each cluster improves. We ultimately fix the cluster limit at 500, which gives a good trade-off between overall cluster quality for all the ungrounded tagging methods, and size small enough to allow EM-based decipherment to be tractable.

Given this setting, we evaluate our four labeling strategies using the many-to-one approach, as presented in Table 1. Due to the larger number of clusters, the results presented here are higher than and not comparable to the original literature describing the methods.[8] We can, nevertheless, make relative judgements. In all cases, clustering by type with Brown-based algorithms works better than using a sophisticated tagger such as A-HMM. Since BROWN and MARLIN obtain similar results, with no consistently dominant model, in all subsequent experiments we use the BROWN labeler with 500 clusters.

3.3 Grounding via Decipherment

We now seek an appropriate method for grounding the clusters generated in Section 3.2. We experiment with en, fr, fa, and sw as CLs. For each CL t, we instantiate our model following Equation 1, using the Carmel toolkit (Graehl, 1997) and forming the cipher table as a one-state transducer. We train these models using EM for 500 iterations or until convergence, and we select the model with the lowest perplexity from among 70 random restarts.

Yet unspecified is the nature of the POS language model $P_\theta(p)$. We begin by training bigram models of POS tag sequences with additive smoothing using the SRILM toolkit (Stolcke, 2002) for each PL $s \in \mathcal{S}$ = {en, de, fr, it, es, ja, ar, cs, ru, sw}. But which PL's POS tag data to use for each CL? We explore two initial criteria for choosing a single suitable PL s: confidence of the model during decoding (perplexity, PPL), and typological similarity. For the first criterion, the PL whose cipher grounder s-t yields the better performance is chosen. For the second criterion, the most similar language to CL t is chosen according to the cosine similarity between typological features vectors. We employ 102 features obtained from WALS[9] related to word order and morphosyntactic alignment, further reduced to 50 dimensions using PCA. However, none these criteria correlates significantly to tagging accuracy, as we elaborate in Section 5.1. We instead try a combined approach.

The likelihood of cluster ID replacement, $P_\theta(\hat{c}_i|p_j), \forall \hat{c}_i \in C, \forall p_j$ in the tagset, is replaced by

$$P_{avg}(\hat{c}_i|p_j) \sim \frac{\sum_{s \in \mathcal{S}, s \neq t} P_\theta(\hat{c}_i|p_j^s)}{|S| - 1}$$

where $P_\theta(\hat{c}_i|p_j^s)$ is the likelihood of POS tag p_j being represented by cluster $\hat{c}_i$ after training with the language s tag distribution. Note that the CL is excluded from $\mathcal{S}$ for the combination. The combined cipher grounder is then defined by

$$\underset{p}{\operatorname{argmax}}\, P_{all}(p) P_{avg}(\hat{c}|p) \qquad (2)$$

where $P_{all}(p)$ is a language model trained over the concatenation of POS sequences of all parent languages in $\mathcal{S}$. We call this approach CIPHER-AVG.

4 Downstream Tasks

4.1 Name Tagging

We experiment with the LSTM-CNN model proposed by Chiu and Nichols (2016), one of the

---

[6] `https://github.com/percyliang/brown-cluster`

[7] Optimized and implemented by Müller and Schuetze (2015). Available at `http://cistern.cis.lmu.de/marlin/`

[8] As noted by Clark (2003) and Johnson (2007), in the limit, keeping each type (or, in the case of A-HMM, TOKEN in its own cluster will result in the maximum possible many-to-one (polysemic types prevent perfect accuracy when type clustering).

[9] `https://wals.info/`

Seq. Tagger	en	de	fr	ru	fa	sw
BROWN	81.37	**81.28**	84.81	**79.78**	**86.94**	87.35
MARLIN	**81.53**	81.25	**85.4**	79.14	86.64	**88.81**
A-HMM	77.12	74.85	81.48	73.88	80.25	76.69
E-KMEANS	63.01	65.14	68.68	70.80	76.94	65.08

Table 1: Comparison of labeling strategies using many-to-one mapping for target languages with available test data, using 500 clusters or number of states. Accuracy is shown in percentage points.

state-of-the-art name tagging models, as our baseline model. To incorporate POS features, we extend the token representation (word and character embeddings) with a one-hot vector representation of the POS tag. Figure 2 presents an outline of the architecture.

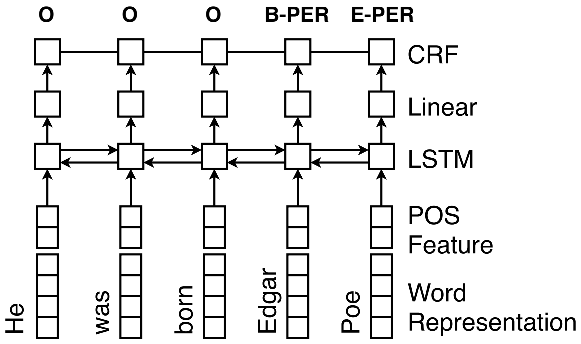


Figure 2: Name tagging model evaluated in Section 5.2. This is an extension of the model of Chiu and Nichols (2016) with POS tag features added.

4.2 Multilingual Dependency Parsing

We base our experiments on the no-treebank setup of MALOPA (Ammar et al., 2016), but change the underlying transition-based parser to the graph-based parser proposed by Dozat and Manning (2017) for implementation convenience. Following this setup, for each CL except en, we train the parser on the concatenation of treebanks of the other 6 languages as PLs.

The original MALOPA work enriches the input representation by concatenating pretrained multilingual word embeddings (Guo et al., 2016), multilingual Brown cluster IDs, and POS tag information. However, these representations are obtained using parallel corpora and gold POS tags are required for parsing at test time.

In contrast, we are interested in the realistic scenario in which no resource is available in the child language but raw text. It is important to note, however, that our objective is not to beat the state-of-the-art on this benchmark but to investigate parsing performance fluctuation when cross-lingual components (gold POS annotations and supervised multilingual embeddings) are replaced by those obtained in an unsupervised manner.

We investigate the following variations to each component of the input representation.

- **Multilingual word and cluster embeddings.** The original work of Ammar et al. (2016) uses 'robustly projected' pre-trained embeddings (Guo et al., 2015) for word embeddings and embeddings learned from English Brown cluster IDs projected through word alignments (Guo et al., 2016) for cluster embeddings; both of these rely on parallel data and we refer to them collectively as GUO. We replace these with monolingual *fastText* embeddings (Bojanowski et al., 2017) projected to a common space using MUSE, the unsupervised method of Lample et al. (2018b). For cluster embeddings we start with *fastText* monolingual embeddings trained over Brown cluster ID sequences instead of word tokens ($|C| = 256$, the same as in Guo et al. (2016)). Then, unsupervised multilingual embeddings are derived, again using MUSE.[10] Note that this approach, which we refer to collectively as MUSE, requires no parallel data. We compare both MUSE and GUO approaches in Section 5.2 and Table 5.
- **POS tag scheme.** The original work uses gold POS tag data at both train and test time. While realistic to have gold POS info from PLs for training, it is unrealistic to have this data available for new CLs at test time. We thus compare the original GOLD scenario with the realistic CIPHER scenario, where the training data is still gold, but the test POS tags use the method presented in this work. Another realistic scenario dispenses

---

[10] Both cluster and word MUSE embeddings are projected to the corresponding English space.

with POS disambiguation except for the trivial distinction of punctuation; for compatibility purposes this is done in both train and test data and is labeled NONE.

We investigate all combinations of {GUO, MUSE}-{GOLD, CIPHER, NONE}.

5 Results and Discussion

5.1 Labeling and Cipher Grounding

The results in Table 1 are somewhat at odds with those presented in Stratos et al. (2016), but these are done at different operating points; we use different data, the UD-17 tag set instead of the Universal Treebank 12 tag set, and, perhaps most importantly, generate more clusters. We further note that to some degree, choosing Brown clusters based on the results in Table 1 compromises claims of our approach being fully ‘unsupervised’ for those six languages, however our subsequent experiments on additional languages are truly unsupervised.

Table 2 presents the intrinsic performance of the cipher grounder over all PL-CL pairs considered. The difference between the best and the worst performing PL for each CL ranges from 24.62 percentage points for Swahili to 48.34 points for French, and an average difference of 34.5 points among all languages. The case when PL=CL is also presented in Table 2 as a reference and provides a reliable upper-bound under zero-resource conditions. It is worth noting the difference in accuracy when comparing the best performing PL for each CL with its corresponding PL=CL upper-bound. Among all CLs, the best cipher grounder for French (es-fr) gets the closest to its upper-bound with just 4.81 percentage points of difference, followed by the English grounder (fr-en) with 13.53 points of difference. On the other hand, the best Swahili grounder (ar-sw) is the most distant from its upper-bound with 30.45 points of difference.

Given such wide performance gaps in the CL set, the choice of a suitable PL becomes crucial for performance; therein the cipher model confidence and typological similarity are explored as possible choice criteria. With regards to model confidence, the Pearson correlation between accuracy scores and PPL, expected to be negative, ranges from -0.71 for English to 0.40 for Farsi. Since the PPL values for different PLs are not comparable, we first z-normalize PPL per CL and then concatenate the results for all CLs. The Pearson correlation of the resulting PPL-accuracy values is -0.13. This last result indicates that the most confident model might not be the most accurate, hence this criterion is not suitable for choosing a suitable PL.

With regards to typological similarity, we find that the Pearson correlation between accuracy scores and cosine similarity of typological feature vectors, expected to be positive, ranges from 0.44 for English to -0.14 for Farsi. The total correlation is found to be 0.18. Again, we find that the most typologically similar s might not be the most accurate, hence this criterion is not suitable either.

Hence, it becomes obvious that choosing a single PL is an inefficient strategy that does not leverage the contribution that other PLs could bring. In this situation, the combination of cipher grounders for several PLs represents a sound strategy when no prior linguistic information of a certain CL is available. As shown in Table 2, this model, CIPHER-AVG, obtains accuracy scores of 56.4, 58.6, 37.4, and 37.8 % for en, fr, fa, and sw, respectively. When compared to the best performing PL for each CL (see bold cells in Table 2), it can be noticed that the performance gap ranges from just 1.2 percentage points for Swahili to 13.3 points for French, with an average of 6.1 points among all target languages.

Let us now compare the performance of CIPHER-AVG with that of a vanilla supervised neural model.[11] Table 3 shows precision, recall, and F1 scores for the NOUN tag. Even though CIPHER-AVG achieved mixed results (mid to low accuracy), the model robustly achieves mid-range performance according to F1-score for all CLs. The results are even more optimistic in terms of recall for English and French, and in terms of precision for Farsi and Swahili. This gives us hope that CIPHER-AVG can provide a useful, if noisy, signal to downstream tasks that depend on non-trivial performance over specific POS tags, such as name tagging, as exposed in the next section.

---

[11] We use UDPipe v1.2.0 (Straka and Straková, 2017) to train the models.

CL	Parent Language (PL) en	de	fr	it	es	ja	cs	ru	ar	sw	CIPHER-AVG	PL=CL
en	-	57.1	**60.4**	59.9	59.4	25.1	52.8	49.0	30.7	28.4	56.4	73.9
fr	58.1	56.0	-	68.6	**71.9**	23.6	48.3	47.8	35.0	26.7	58.6	76.7
fa	13.8	32.3	29.7	22.7	33.3	19.7	33.3	**43.5**	37.0	38.2	37.4	73.3
sw	24.9	14.3	37.3	21.2	35.9	21.3	25.8	27.9	**38.96**	-	37.8	69.4

Table 2: Performance of cipher grounder using BROWN ($|C| = 500$) as labeler. The best PL for each CL besides itself, is shown in bold. The artificial case where we have CL POS data (PL=CL) is shown for comparison, as is the ultimately used CIPHER-AVG method.

	CIPHER-AVG			Supervised		
CL	P	R	F1	P	R	F1
en	47.70	64.4	54.81	94.04	90.44	92.20
fr	56.26	78.82	65.65	96.15	93.72	94.92
fa	64.94	51.23	57.27	96.48	97.77	97.12
sw	53.46	51.82	52.63	98.88	97.50	98.18

Table 3: Comparison of performance over the NOUN tag, as measured by precision (P), recall (R), and F1 scores, between our combined cipher grounder (CIPHER-AVG) and a supervised tagger.

5.2 Extrinsic evaluation

In the name tagging task, our LSTM-CNN baseline obtains 78.76% and 70.76% F1 score for Kinyarwanda and Sinhalese, respectively. When enriching the input representation with CIPHER-AVG tags, the performance goes up to 80.16% and 71.71% respectively. These results suggest that the signal provided by the combined cipher grounder is significant enough for relevant tags such as common, proper nouns and noun modifiers. As an example, consider the sentence `Kwizera Peace Ndaruhutse , wari wambaye nomero 11.` The baseline model fails to recognize `Kwizera Peace Ndaruhutse` as a person name. In contrast, with the PROPN tag assigned by CIPHER-AVG to `Kwizera`, `Peace`, and `Ndaruhutse`, our model is able to identify this name.

Likewise, the utility of CIPHER-AVG tags for dependency parsing under zero-resource scenarios is summarized in Table 4 and Table 5. It is important to point out that, even though the MALOPA setup follows the no-treebank setup of Ammar et al. (2016), parsing scores in the first row of Table 4 differ from those reported by them (Table 8 in Ammar et al. (2016)). Such difference is to be expected since the underlying parser used in our experiments is a graph-based neural parser (Dozat and Manning, 2017) instead of a transition-based one (Dyer et al., 2015).[12] As mentioned earlier, our objective is to analyze the effect of our tagger's signal on parsing performance under no-resource scenarios, instead of pushing the state-of-the-art for the task.

We first analyze the effect of POS tag information at test time for the MALOPA setup in Table 4. First we remove all POS signal except trivial punctuation information (NONE row), and, predictably, the scores drop significantly across all target languages. Then, we use our cipher tags (CIPHER row) and see improvements for all languages in LAS and for all but one language in UAS (de). This demonstrates the value of our cipher approach.

We then take the next logical step and remove the parallel data-grounded embeddings, replacing them with fully unsupervised MUSE embeddings. Table 5 summarizes these results. Let us compare MUSE-NONE setup (no POS signal at train or test time) with MUSE-GOLD (gold POS signal at train and test time). It can be observed that POS signal improves performance greatly for all languages when using MUSE embeddings. However, consider GUO-GOLD and MUSE-NONE. Here we note a mixed result: whilst de, sv, and it do benefit from POS information, the other languages do not, obtaining great improvements from MUSE embed-

---

[12] Due to time constraints, we could not experiment with longer training regimes possibly needed given the high block dropout rates in Dozat and Manning (2017).

dings instead. Finally, consider MUSE-CIPHER (gold POS tags during training, cipher tags during testing). When compared to MUSE-NONE setup, it can be observed that, unfortunately, the heuristic POS tagger is too noisy and gets in MUSE's way.

6 Related Work

Our proposed tagging pipeline can be interpreted as first reducing the vocabulary size to a fixed number of clusters, and then finding a cluster–POS tag mapping table that best explains the data without any path constraint (a cluster ID could be mapped to any POS tag). In this sense, our approach applies EM to simplify the task (e.g. when using Brown clustering (Brown et al., 1992)), followed by another EM run to optimize cipher table parameters.

Under this lens, the methods closest to our approach are those which attempt to reduce or constrain the parameter search space prior to running EM. For instance, Ravi and Knight (2009) explicitly search for the smallest model that explains the data using Integer Programming, and then use EM to set parameter values. In a different approach, Goldberg et al. (2008) obtain competitive performance with a classic HMM model by initializing the emission probability distribution with a mixture of language-specific, linguistically constrained distributions. However, both of these approaches are framed around the task of unsupervised POS *disambiguation* with a full dictionary (Merialdo, 1994). Previous work relaxes the full dictionary constraint by leveraging monolingual lexicons (Haghighi and Klein, 2006; Smith and Eisner, 2005; Merialdo, 1994; Ravi and Knight, 2009), multilingual tagged dictionaries (Li et al., 2012; Fang and Cohn, 2017), and parallel corpora (Duong et al., 2014; Täckström et al., 2013; Das and Petrov, 2011).

In addition, previous work includes sequence models that do not rely on any resource besides raw text during training, namely unsupervised POS *induction* models. These models are based, with few exceptions, on extensions to the standard HMM; most, in the form of appropriate priors over the HMM multinomial parameters (Goldwater and Griffiths, 2007; Johnson, 2007; Ganchev et al., 2009); others, by using logistic distributions instead of multinomial ones (Berg-Kirkpatrick et al., 2010; Stratos et al., 2016). However, these models still need to ground or map hidden states to actual POS tags to evaluate, and they inevitably resort to many-to-one or one-to-one accuracy scoring. Some previous work has been cautious in pointing out this ill-defined setting (Ravi and Knight, 2009; Christodoulopoulos et al., 2010), and we argue its inappropriateness for scenarios in which the test set is extremely small or even when no annotated reference corpus exists.

Therefore, the problem of grounding the sequence of states or cluster IDs to POS tags without using any linguistic resource remains unsolved. We formulate this task as a decipherment problem. Decipherment aims to find a substitution table between alphabets or tokens of an encrypted code and a known language without the need of parallel corpora. The task has been successfully applied in alphabet mapping for lost languages (Snyder et al., 2010), and machine translation at the character (Pourdamghani and Knight, 2017) and token level (Ravi and Knight, 2011; Dou et al., 2015). For the task of POS tag grounding, the sequence of states or cluster IDs is modeled as an encrypted code to be deciphered back to a POS sequence. Furthermore, we tackle the problem from a 'universal' perspective by allowing the cipher learn from POS sequences from a varied pool of languages.

Other recent work has declared a 'radically universal' mantra to language inclusivity. Hermjakob et al. (2018) presents a Romanizer that covers all writing systems known to Unicode. Pan et al. (2017) extends name tagging and linking capability to hundreds of languages by leveraging Wikipedia. Kirov et al. (2016) has semi-automatically built inflectional paradigms for hundreds of languages.

7 Conclusion

We present a POS tag grounding strategy based on decipherment that does not require human-labeled data to map states or clusters to actual POS tags and thus can be used in real-world situations requiring grounded POS tags. The decipherment model considers state or word cluster IDs of a CL as a cipher text to be deciphered back to a POS sequence.

The model operates on top of Brown cluster IDs and requires a POS language model trained on annotated corpora of one or more PLs. Experimental results over a large and linguistically varied set of PLs show that the choice of which PL to decipher

Test Tags	de		fr		es		it		pt		sv	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
GOLD	65.57	52.37	71.27	59.80	73.26	63.13	71.46	59.66	63.28	54.93	77.50	64.90
NONE	40.90	18.61	51.14	30.91	43.82	17.67	48.22	33.29	37.89	16.72	38.15	17.96
CIPHER (this work)	38.31	**24.72**	**54.46**	**41.04**	**55.56**	**41.16**	**54.05**	**39.78**	**46.97**	**36.07**	**55.06**	**36.51**

Table 4: Impact of grounded unsupervised POS tagging on MALOPA's 'zero-resource' condition. Bold entries indicate an improvement over the baseline condition of having no POS tag information (beyond punctuation)

Embeddings	Test Tags	de		fr		es		it		pt		sv	
		UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
GUO	GOLD	65.57	52.37	71.27	59.80	73.26	63.13	71.46	59.66	63.28	54.93	**77.50**	**64.90**
MUSE	GOLD	**66.19**	**56.28**	**80.86**	**72.65**	**81.06**	**73.62**	**82.08**	**72.40**	**81.17**	**76.17**	72.46	61.71
MUSE	NONE	57.26	45.10	*73.84*	*63.09*	*77.01*	*67.06*	71.36	*60.48*	*75.31*	*68.36*	60.82	45.25
MUSE	CIPHER	48.56	37.13	69.94	59.22	73.86	61.68	69.30	56.85	73.41	65.23	57.39	41.49

Table 5: Changing to unsupervised MUSE embeddings boosts MALOPA's zero-resource performance significantly (**bold** entries), in many cases doing so even without any POS tag information (*italic* entries), however noisy decipherment-based POS tags are no longer helpful.

POS tags from is crucial for performance. We explore model confidence, as measured by perplexity and typological similarities, as intuitive criteria for PL choice. However, both criteria prove to be not correlated with tagging accuracy scores. Thus, we propose a cipher model combination strategy in order to leverage the word-order patterns in several PLs, at the cost of an accuracy drop ranging from just 1.15 percentage points to 13.33 points.

The resulting combined grounder is completely language agnostic, making it attractive for the analysis of languages new to the academic community. Furthermore, analysis over the tasks of name tagging and dependency parsing demonstrate that the tags induced by the combined grounder provide a non-trivial signal for improvement of the downstream task. We obtain state-of-the-art results for name tagging in Kinyarwanda and Sinhalese, languages for which POS annotated corpora is nearly non-existent.

Acknowledgments

Thanks to Xusen Yin, Nima Pourdamghani, Thamme Gowda, and Nanyun Peng for fruitful discussions. This work was sponsored by DARPA LORELEI (HR0011-15-C-0115).

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4(1):357–370.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA. Association for Computational Linguistics.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 59–66. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying bayesian inference and vector space models for improved decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 836–845, Beijing, China. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. 2014. What can we get from 1000 tokens? a case study of multilingual POS tagging for resource-poor languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 886–897, Doha, Qatar. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 587–593, Vancouver, Canada. Association for Computational Linguistics.

Kuzman Ganchev, Ben Taskar, Fernando Pereira, and Joao Graca. 2009. Posterior vs parameter sparsity in latent variable models. In *Advances in Neural Information Processing Systems*, pages 664–672.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL-08: HLT*, pages 746–754, Columbus, Ohio. Association for Computational Linguistics.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic. Association for Computational Linguistics.

Jonathan Graehl. 1997. Carmel finite-state toolkit. *ISI/USC*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *AAAI Conference on Artificial Intelligence*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA. Association for Computational Linguistics.

William P. Headden III, David McClosky, and Eugene Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 329–336, Manchester, UK. Coling 2008 Organizing Committee.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018b. Word translation without parallel data. In *International Conference on Learning Representations*.

Will Lewis, Robert Munro, and Stephan Vogel. 2011. Crisis MT: Developing a cookbook for MT in crisis situations. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1389–1398, Jeju Island, Korea. Association for Computational Linguistics.

Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Thomas Müller and Hinrich Schuetze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, Colorado. Association for Computational Linguistics.

Robert Munro and Christopher Manning. 2012. Short message communications: users, topics, and in-language processing. In *Proceedings of the Second Annual Symposium on Computing for Development*, Atlanta.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Nima Pourdamghani and Kevin Knight. 2017. Deciphering related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2513–2518, Copenhagen, Denmark. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338, Lisbon, Portugal. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Portland, Oregon, USA. Association for Computational Linguistics.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan. Association for Computational Linguistics.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057, Uppsala, Sweden. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden Markov models. *Transactions of the Association for Computational Linguistics*, 4(1):245–257.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de-Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867, Lisbon, Portugal. Association for Computational Linguistics.

On Difficulties of Cross-Lingual Transfer with Order Differences: A Case Study on Dependency Parsing

Wasi Uddin Ahmad[1*], **Zhisong Zhang**[2*], **Xuezhe Ma**[2]
`wasiahmad@cs.ucla.edu`, `{zhisongz,xuezhem}@cs.cmu.edu`
Eduard Hovy[2], **Kai-Wei Chang**[1], **Nanyun Peng**[3†]
`ehovy@cs.cmu.edu`, `kwchang@cs.ucla.edu`, `npeng@isi.edu`

[1]University of California, Los Angeles, [2]Carnegie Mellon University
[3]University of Southern California

Abstract

Different languages might have different word orders. In this paper, we investigate cross-lingual transfer and posit that an order-agnostic model will perform better when transferring to distant foreign languages. To test our hypothesis, we train dependency parsers on an English corpus and evaluate their transfer performance on 30 other languages. Specifically, we compare encoders and decoders based on Recurrent Neural Networks (RNNs) and modified self-attentive architectures. The former relies on sequential information while the latter is more flexible at modeling word order. Rigorous experiments and detailed analysis shows that RNN-based architectures transfer well to languages that are close to English, while self-attentive models have better overall cross-lingual transferability and perform especially well on distant languages.

1 Introduction

Cross-lingual transfer, which transfers models across languages, has tremendous practical value. It reduces the requirement of annotated data for a target language and is especially useful when the target language is lack of resources. Recently, this technique has been applied to many NLP tasks such as text categorization (Zhou et al., 2016a), tagging (Kim et al., 2017), dependency parsing (Guo et al., 2015, 2016) and machine translation (Zoph et al., 2016). Despite the preliminary success, transferring across languages is challenging as it requires understanding and handling differences between languages at levels of morphology, syntax, and semantics. It is especially difficult to learn invariant features that can robustly transfer to distant languages.

Prior work on cross-lingual transfer mainly focused on sharing word-level information by leveraging multi-lingual word embeddings (Xiao and Guo, 2014; Guo et al., 2016; Sil et al., 2018). However, words are not independent in sentences; their combinations form larger linguistic units, known as *context*. Encoding context information is vital for many NLP tasks, and a variety of approaches (e.g., convolutional neural networks and recurrent neural networks) have been proposed to encode context as a high-level feature for downstream tasks. In this paper, we study how to transfer generic contextual information across languages.

For cross-language transfer, one of the key challenges is the variation in word order among different languages. For example, the Verb-Object pattern in English can hardly be found in Japanese. This challenge should be taken into consideration in model design. RNN is a prevalent family of models for many NLP tasks and has demonstrated compelling performances (Mikolov et al., 2010; Sutskever et al., 2014; Peters et al., 2018). However, its sequential nature makes it heavily reliant on word order information, which exposes to the risk of encoding language-specific order information that cannot generalize across languages. We characterize this as the "*order-sensitive*" property. Another family of models known as "Transformer" uses self-attention mechanisms to capture context and was shown to be effective in various NLP tasks (Vaswani et al., 2017; Liu et al., 2018; Kitaev and Klein, 2018). With modification in position representations, the self-attention mechanism can be more robust than RNNs to the change of word order. We refer to this as the "*order-free*" property.

In this work, we posit that *order-free* models have better transferability than *order-sensitive* models because they less suffer from overfitting

---

[*] Equal contribution. Listed by alphabetical order.

[†] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 2440–2452
Minneapolis, Minnesota, June 2 - June 7, 2019.

Language Families	Languages
Afro-Asiatic	Arabic (ar), Hebrew (he)
Austronesian	Indonesian (id)
IE.Baltic	Latvian (lv)
IE.Germanic	Danish (da), Dutch (nl), English (en), German (de), Norwegian (no), Swedish (sv)
IE.Indic	Hindi (hi)
IE.Latin	Latin (la)
IE.Romance	Catalan (ca), French (fr), Italian (it), Portuguese (pt), Romanian (ro), Spanish (es)
IE.Slavic	Bulgarian (bg), Croatian (hr), Czech (cs), Polish (pl), Russian (ru), Slovak (sk), Slovenian (sl), Ukrainian (uk)
Japanese	Japanese (ja)
Korean	Korean (ko)
Sino-Tibetan	Chinese (zh)
Uralic	Estonian (et), Finnish (fi)

Table 1: The selected languages grouped by language families. "IE" is the abbreviation of Indo-European.

language-specific word order features. To test our hypothesis, we first quantify language distance in terms of word order typology, and then systematically study the transferability of order-sensitive and order-free neural architectures on cross-lingual dependency parsing.

We use dependency parsing as a test bed primarily because of the availability of unified annotations across a broad spectrum of languages (Nivre et al., 2018). Besides, word order typology is found to influence dependency parsing (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Ammar et al., 2016; Aufrant et al., 2016). Moreover, parsing is a low-level NLP task (Hashimoto et al., 2017) that can benefit many downstream applications (McClosky et al., 2011; Gamallo et al., 2012; Jie et al., 2017).

We conduct evaluations on 31 languages across a broad spectrum of language families, as shown in Table 1. Our empirical results show that *order-free* encoding and decoding models generally perform better than the *order-sensitive* ones for cross-lingual transfer, especially when the source and target languages are distant.

2 Quantifying Language Distance

We first verify that we can measure "language distance" base on word order since it is a significant distinctive feature to differentiate languages (Dryer, 2007). The World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013) provides a great reference for word order typology

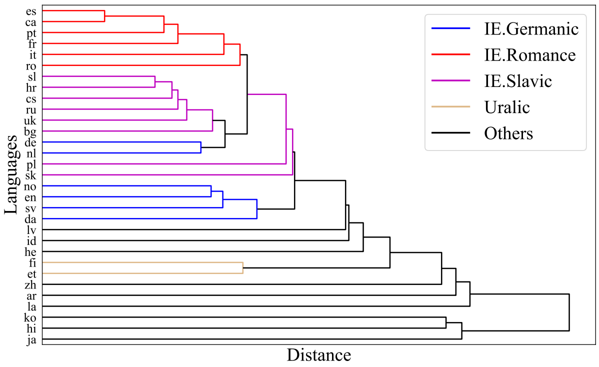


Figure 1: Hierarchical clustering (with the Nearest Point Algorithm) dendrogram of the languages by their word-ordering vectors.

and can be used to construct feature vectors for languages (Littell et al., 2017). But since we already have the universal dependency annotations, we take an empirical way and directly extract word order features using directed dependency relations (Liu, 2010).

We conduct our study using the Universal Dependencies (UD) Treebanks (v2.2) (Nivre et al., 2018). We select 31 languages for evaluation and analysis, with the selection criterion being that the total token number in the treebanks of that language is over 100K. We group these languages by their language families in Table 1. Detailed statistical information of the selected languages and treebanks can be found in Appendix A[1].

We look at finer-grained dependency types than the 37 universal dependency labels[2] in UD v2 by augmenting the dependency labels with the universal part-of-speech (POS) tags of the head and modifier[3] nodes. Specifically, we use triples "(ModifierPOS, HeadPOS, DependencyLabel)" as the augmented dependency types. With this, we can investigate language differences in a fine-grained way by defining directions on these triples (i.e. modifier before head or modifier after head).

We conduct feature selection by filtering out rare types as they can be unstable. We defer the results in 52 selected types and more details to Appendix C. For each dependency type, we collect the statistics of directionality (Liu, 2010; Wang and Eisner, 2017). Since there can be only two directions for an edge, for each dependency type,

---

[1] Please refer to the supplementary materials for all the appendices of this paper.

[2] http://universaldependencies.org/u/dep/index.html

[3] In this paper, we use the term of "modifier", which can also be described as "dependent" or "child" node.

we use the relative frequency of the left-direction (modifier before head) as the directional feature. By concatenating the directional features of all selected triples, we obtain a word-ordering feature vector for each language. We calculate the *word-ordering distance* using these vectors. In this work, we simply use Manhattan distance, which works well as shown in our analysis (Section 4.3).

We perform hierarchical clustering based on the word-ordering vectors for the selected languages, following Östling (2015). As shown in Figure 1, the grouping of the ground truth language families is almost recovered. The two outliers, German (de) and Dutch (nl), are indeed different from English. For instance, German and Dutch adopt a larger portion of Object-Verb order in embedded clauses. The above analysis shows that word order is an important feature to characterize differences between languages. Therefore, it should be taken into consideration in the model design.

3 Models

Our primary goal is to conduct cross-lingual transfer of syntactic dependencies without providing any annotation in the target languages. The overall architecture of models that are studied in this research is described as follows. The first layer is an input embedding layer, for which we simply concatenate word and POS embeddings. The POS embeddings are trained from scratch, while the word embeddings are fixed and initialized with the multilingual embeddings by Smith et al. (2017). These inputs are fed to the encoder to get contextual representations, which is further used by the decoder for predicting parse trees.

For the cross-lingual transfer, we hypothesize that the models capturing less language-specific information of the source language will have better transferability. We focus on the word order information, and explore different encoders and decoders that are considered as *order-sensitive* and *order-free*, respectively.

3.1 Contextual Encoders

Considering the sequential nature of languages, RNN is a natural choice for the encoder. However, modeling sentences word by word in the sequence inevitably encodes word order information, which may be specific to the source language. To alleviate this problem, we adopt the self-attention based encoder (Vaswani et al., 2017) for cross-lingual parsing. It can be less sensitive to word order but not necessarily less potent at capturing contextual information, which makes it suitable for our study.

RNNs Encoder Following prior work (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017), we employ k-layer bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) on top of the input vectors to obtain contextual representations. Since it explicitly depends on word order, we will refer it as an *order-sensitive* encoder.

Self-Attention Encoder The original self-attention encoder (Transformer) takes absolute positional embeddings as inputs, which capture much order information. To mitigate this, we utilize relative position representations (Shaw et al., 2018), with further simple modification to make it order-agnostic: the original relative position representations discriminate left and right contexts by adding signs to distances, while we discard the directional information.

We directly base our descriptions on those in (Shaw et al., 2018). For the relative positional self-attention encoder, each layer calculates multiple attention heads. In each head, the input sequence of vectors $\mathbf{x} = (x_1, \ldots, x_n)$ are transformed into the output sequence of vectors $\mathbf{z} = (z_1, \ldots, z_n)$, based on the self-attention mechanism:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

Here, a_{ij}^V and a_{ij}^K are relative positional representations for the two position i and j. Similarly, we clip the distance with a maximum threshold of k (which is empirically set to 10), but we do not discriminate positive and negative values. Instead, since we do not want the model to be aware of directional information, we use the absolute values of the position differences:

$$a_{ij}^K = w_{clip(|j-i|,k)}^K \quad a_{ij}^V = w_{clip(|j-i|,k)}^V$$
$$clip(x, k) = \min(|x|, k)$$

Therefore, the learnable relative postion representations have $k+1$ types rather than $2k+1$: we have $w^K = (w_0^K, \ldots, w_k^K)$, and $w^V = (w_0^V, \ldots, w_k^V)$.

With this, the model knows only what words are surrounding but cannot tell the directions. Since self-attention encoder is less sensitive to word order, we refer to it as an *order-free* encoder.

3.2 Structured Decoders

With the contextual representations from the encoder, the decoder predicts the output tree structures. We also investigate two types of decoders with different sensitivity to ordering information.

Stack-Pointer Decoder Recently, Ma et al. (2018) proposed a top-down transition-based decoder and obtained state-of-the-art results. Thus, we select it as our transition-based decoder. To be noted, in this Stack-Pointer decoder, RNN is utilized to record the decoding trajectory and also can be sensitive to word order. Therefore, we will refer to it as an *order-sensitive* decoder.

Graph-based Decoder Graph-based decoders assume simple factorization and can search globally for the best structure. Recently, with a deep biaffine attentional scorer, Dozat and Manning (2017) obtained state-of-the-art results with simple first-order factorization (Eisner, 1996; McDonald et al., 2005). This method resembles the self-attention encoder and can be regarded as a self-attention output layer. Since it does not depend on ordering information, we refer to it as an *order-free* decoder.

4 Experiments and Analysis

In this section, we compare four architectures for cross-lingual transfer dependency parsing with a different combination of order-free and order-sensitive encoder and decoder. We conduct several detailed analyses showing the pros and cons of both types of models.

4.1 Setup

Settings In our main experiments[4] (those except Section 4.3.5), we take English as the source language and 30 other languages as target languages. We only use the source language for both training and hyper-parameter tuning. During testing, we directly apply the trained model to target languages with the inputs from target languages passed through pretrained multilingual embeddings that are projected into a common space as the source language. The projection is done by the offline transformation method (Smith et al., 2017) with pre-trained $300d$ monolingual embeddings from FastText (Bojanowski et al., 2017). We freeze word embeddings since fine-tuning on them may disturb the multi-lingual alignments. We also adopt gold UPOS tags for the inputs.

For other hyper-parameters, we adopted similar ones as in the Biaffine Graph Parser (Dozat and Manning, 2017) and the Stack-Pointer Parser (Ma et al., 2018). Detailed hyper-parameter settings can be found in Appendix B. Throughout our experiments, we adopted the language-independent UD labels and a sentence length threshold of 140. The evaluation metrics are Unlabeled attachment score (UAS) and labeled attachment score (LAS) with punctuations excluded[5]. We trained our cross-lingual models five times with different initializations and reported average scores.

Systems As described before, we have an *order-free* (Self-Attention) and an *order-sensitive* (BiLSTM-RNN) encoder, as well as an *order-free* (Biaffine Attention Graph-based) and an *order-sensitive* (Stack-Pointer) decoder. The combination gives us four different models, named in the format of "Encoder" plus "Decoder". For clarity, we also mark each model with their encoder-decoder order sensitivity characteristics. For example, "SelfAtt-Graph (OF-OF)" refers to the model with self-attention order-free encoder and graph-based order-free decoder. We benchmark our models with a baseline shift-reduce transition-based parser, which gave previous state-of-the-art results for single-source zero-resource cross-lingual parsing (Guo et al., 2015). Since they used older datasets, we re-trained the model on our datasets with their implementation[6]. We also list the supervised learning results using the "RNN-Graph" model on each language as a reference of the upper-line for cross-lingual parsing.

4.2 Results

The results on the test sets are shown in Table 2. The languages are ordered by their order typology distance to English. In preliminary experiments, we found our lexicalized models performed poorly

---

[4] Our implementation is publicly available at: https://github.com/uclanlp/CrossLingualDepParser

[5] In our evaluations, we exclude tokens whose POS tags are "PUNCT" or "SYM". This setting is different from the one adopted in the CoNLL shared task (Zeman et al., 2018). However, the patterns are similar as shown in Appendix D where we report the punctuation-included test evaluations.

[6] https://github.com/jiangfeng1124/acl15-clnndep. We also evaluated our models on the older dataset and compared with their results, as shown in Appendix F.

Lang	Dist. to English	SelfAtt-Graph (OF-OF)	RNN-Graph (OS-OF)	SelfAtt-Stack (OF-OS)	RNN-Stack (OS-OS)	Baseline (Guo et al., 2015)	Supervised (RNN-Graph)
en	0.00	90.35/88.40	90.44/88.31	90.18/88.06	**91.82†/89.89†**	87.25/85.04	90.44/88.31
no	0.06	80.80/72.81	80.67/72.83	80.25/72.07	**81.75†/73.30†**	74.76/65.16	94.52/92.88
sv	0.07	80.98/73.17	81.23/73.49	80.56/72.77	**82.57†/74.25†**	71.84/63.52	89.79/86.60
fr	0.09	77.87/72.78	**78.35†/73.46†**	76.79/71.77	75.46/70.49	73.02/64.67	91.90/89.14
pt	0.09	**76.61†**/67.75	76.46/**67.98**	75.39/66.67	74.64/66.11	70.36/60.11	93.14/90.82
da	0.10	76.64/67.87	77.36/68.81	76.39/67.48	**78.22†/68.83**	71.34/61.45	87.16/84.23
es	0.12	74.49/66.44	**74.92†/66.91†**	73.15/65.14	73.11/64.81	68.75/59.59	93.17/90.80
it	0.12	80.80/75.82	**81.10/76.23†**	79.13/74.16	80.35/75.32	75.06/67.37	94.21/92.38
hr	0.13	**61.91†/52.86†**	60.09/50.67	60.58/51.07	60.80/51.12	52.92/42.19	89.66/83.81
ca	0.13	73.83/65.13	**74.24†/65.57†**	72.39/63.72	72.03/63.02	68.23/58.15	93.98/91.64
pl	0.13	**74.56†/62.23†**	71.89/58.59	73.46/60.49	72.09/59.75	66.74/53.40	94.96/90.68
uk	0.13	**60.05/52.28†**	58.49/51.14	57.43/49.66	59.67/51.85	54.10/45.26	85.98/82.21
sl	0.13	**68.21†/56.54†**	66.27/54.57	66.55/54.58	67.76/55.68	60.86/48.06	86.79/82.76
nl	0.14	68.55/60.26	67.88/60.11	67.88/59.46	**69.55†/61.55†**	63.31/53.79	90.59/87.52
bg	0.14	**79.40†/68.21†**	78.05/66.68	78.16/66.95	78.83/67.57	73.08/61.23	93.74/89.61
ru	0.14	60.63/51.63	59.99/50.81	59.36/50.25	**60.87/51.96**	55.03/45.09	94.11/92.56
de	0.14	**71.34†/61.62†**	69.49/59.31	69.94/60.09	69.58/59.64	65.14/54.13	88.58/83.68
he	0.14	**55.29/48.00†**	54.55/46.93	53.23/45.69	54.89/40.95	46.03/26.57	89.34/84.49
cs	0.14	**63.10†/53.80†**	61.88/52.80	61.26/51.86	62.26/52.32	56.15/44.77	94.03/91.87
ro	0.15	**65.05†/54.10†**	63.23/52.11	62.54/51.46	60.98/49.79	56.01/44.04	90.07/84.50
sk	0.17	**66.65/58.15†**	65.41/56.98	65.34/56.68	66.56/57.48	57.75/47.73	90.19/86.38
id	0.17	**49.20†/43.52†**	47.05/42.09	47.32/41.70	46.77/41.28	40.84/33.67	87.19/82.60
lv	0.18	70.78/49.30	**71.43†/49.59**	69.04/47.80	70.56/48.53	62.33/41.42	83.67/78.13
fi	0.20	66.27/48.69	**66.36/48.74**	64.82/47.50	66.25/48.28	58.51/38.65	88.04/85.04
et	0.20	**65.72†/44.87†**	65.25/44.40	64.12/43.26	64.30/43.50	56.13/34.86	86.76/83.28
zh*	0.23	**42.48†/25.10†**	41.53/24.32	40.56/23.32	40.92/23.45	40.03/20.97	73.62/67.67
ar	0.26	**38.12†/28.04†**	32.97/25.48	32.56/23.70	32.85/24.99	32.69/22.68	86.17/81.83
la	0.28	**47.96†/35.21†**	45.96/33.91	45.49/33.19	43.85/31.25	39.08/26.17	81.05/76.33
ko	0.33	**34.48†/16.40†**	33.66/15.40	32.75/15.04	33.11/14.25	31.39/12.70	85.05/80.76
hi	0.40	**35.50†/26.52†**	29.32/21.41	31.38/23.09	25.91/18.07	25.74/16.77	95.63/92.93
ja*	0.49	**28.18†/20.91†**	18.41/11.99	20.72/13.19	15.16/9.32	15.39/08.41	89.06/78.74
Average	0.17	**64.06†/53.82†**	62.71/52.63	62.22/52.00	62.37/51.89	57.09/45.41	89.44/85.62

Table 2: Results (UAS%/LAS%, excluding punctuation) on the test sets. Languages are sorted by the word-ordering distance to English, as shown in the second column. '*' refers to results of delexicalized models, '†' means that the best transfer model is statistically significantly better (by paired bootstrap test, $p < 0.05$) than all other transfer models. Models are marked with their encoder and decoder order sensitivity, OF denotes order-free and OS denotes order-sensitive.

on Chinese (zh) and Japanese (ja). We found the main reason was that their embeddings were not well aligned to English. Therefore, we use delexicalized models, where only POS tags are used as inputs. The delexicalized results[7] for Chinese and Japanese are listed in the rows marked with "*".

Overall, the "SelfAtt-Graph" model performs the best in over half of the languages and beats the runner-up "RNN-Graph" by around 1.3 in UAS and 1.2 in LAS on average. When compared with "RNN-Stack" and "SelfAtt-Stack", the average difference is larger than 1.5 points. This shows that models capture less word order information generally perform better at cross-lingual parsing. Compared with the baseline, our superior results show the importance of the contextual encoder. Compared with the supervised models, the cross-lingual results are still lower by a large gap, indicating space for improvements.

After taking a closer look, we find an interesting pattern in the results: while the model performances on the source language (English) are similar, RNN-based models perform better on languages that are closer to English (upper rows in the table), whereas for languages that are "distant" from English, the "SelfAtt-Graph" performs much better. Such patterns correspond well with our hypothesis, that is, the design of models considering word order information is crucial in cross-lingual transfer. We conduct more thorough analysis in the next subsection.

---

[7] We found delexicalized models to be better only at zh and ja, for about 5 and 10 points respectively. For other languages, they performed worse for about 2 to 5 points. We also tried models without POS, and found them worse for about 10 points on average. We leave further investigation of input representations to future work.

Model	UAS%	LAS%
SelfAtt-Relative (Ours)	64.57	54.14
SelfAtt-Relative+Dir	63.93	53.62
RNN	63.25	52.94
SelfAtt-Absolute	61.76	51.71
SelfAtt-NoPosi	28.18	21.45

Table 3: Comparisons of different encoders (averaged results over all languages on the original training sets).

4.3 Analysis

We further analyze how different modeling choices influence cross-lingual transfer. Since we have not touched the training sets for languages other than English, in this subsection, we evaluate and analyze the performance of target languages using training splits in UD. Performance of English is evaluated on the test set. We verify that the trends observed in test set are similar to those on the training sets. As mentioned in the previous section, the bilingual embeddings for Chinese and Japanese do not align well with English. Therefore, we report the results with delexicalizing. In the following, we discuss our observations, and detailed results are listed in Appendix E.

4.3.1 Encoder Architecture

We assume models that are less sensitive to word order perform better when transfer to distant languages. To empirically verify this point, we conduct controlled comparisons on various encoders with the same graph-based decoder. Table 3 shows the average performances in all languages.

To compare models with various degrees of sensitivity to word order, we include several variations of self-attention models. The "SelfAtt-NoPosi" is the self-attention model without any positional information. Although it is most insensitive to word order, it performs poorly possibly because of the lack of access to the locality of contexts. The self-attention model with absolute positional embeddings ("SelfAtt-Absolute") also does not perform well. In the case of parsing, relative positional representations may be more useful as indicated by the improvements brought by the directional relative position representations ("SelfAtt-Relative+Dir") (Shaw et al., 2018). Interestingly, the RNN encoder ranks between "SelfAtt-Relative+Dir" and "SelfAtt-Absolute"; all these three encoders explicitly capture word order information in some way. Finally, by discarding the information of directions, our relative position representation ("SelfAtt-Relative") performs the best (significantly better at $p < 0.05$).

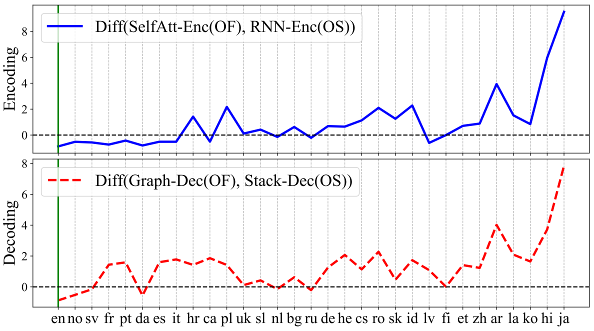


Figure 2: Evaluation score differences between Order-Free (OF) and Order Sensitive (OS) modules. We show results of both encoder (blue solid curve) and decoder (dashed red curve). Languages are sorted by their word-ordering distances to English from left to right. The position of English is marked with a green bar.

One crucial observation we have is that the patterns of breakdown performances for "SelfAtt-Relative+Dir" are similar to those of RNN: on closer languages, the direction-aware model performs better, while on distant languages the non-directional one generally obtains better results. Since the only difference between our proposed "SelfAtt-Relative" model and the "SelfAtt-Relative+Dir" model is the directional encoding, we believe the better performances should credit to its effectiveness in capturing useful context information without depending too much on the language-specific order information.

These results suggest that a model's sensitivity to word order indeed affects its cross-lingual transfer performances. In later sections, we stick to our "SelfAtt-Relative" variation of the self-attentive encoder and focus on the comparisons among the four main models.

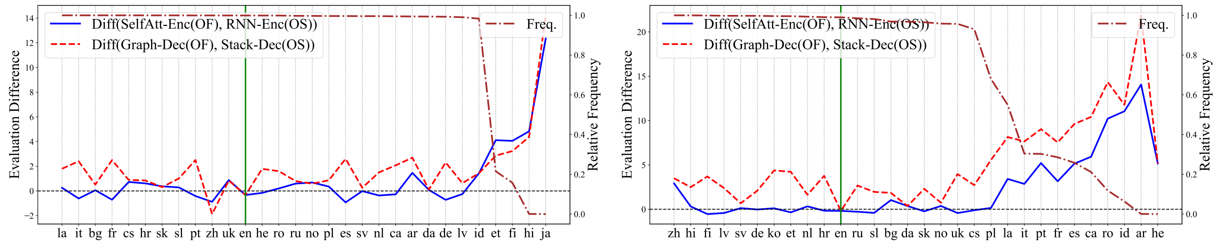
4.3.2 Performance v.s. Language Distance

We posit that order-free models can do better than order-sensitive ones on cross-lingual transfer parsing when the target languages have different word orders to the source language. Now we can analyze this with the word-ordering distance.

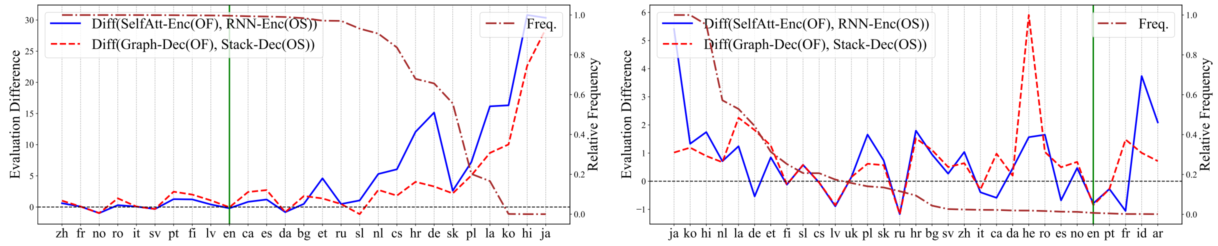
For each target language, we collect two types of distances when comparing it to English: one is the *word-ordering distance* as described in Section 2, the other is the *performance distance*, which is the gap of evaluation scores[8] between the target language and English. The performance distance can represent the general transferability from En-

---

[8] In the rest of this paper, we simply average UAS and LAS for evaluation scores unless otherwise noted.



(a) Adposition & Noun (ADP, NOUN, case)
(b) Adjective & Noun (ADJ, NOUN, amod)



(c) Auxiliary & Verb (AUX, VERB, aux)
(d) Object & Verb (NOUN, VERB, obj)

Figure 3: Analysis on specific dependency types. To save space, we merge the curves of encoders and decoders into one figure. The blue and red curves and left y-axis represent the differences in evaluation scores, the brown curve and right y-axis represents the relative frequency of left-direction (modifier before head) on this type. The languages (x-axis) are sorted by this relative frequency from high to low.

glish to this language. We calculate the correlation of these two distances on all the concerned languages, and the results turn to be quite high: the Pearson and Spearman correlations are *around 0.90 and 0.87* respectively, using the evaluations of any of our four cross-lingual transfer models. This suggests that word order can be an important factor of cross-lingual transferability.

Furthermore, we individually analyze the encoders and decoders of the dependency parsers. Since we have two architectures for each of the modules, when examining one, we take the highest scores obtained by any of the other modules. For example, when comparing RNN and Self-Attention encoders, we take the best evaluation scores of "RNN-Graph" and "RNN-Stack" for RNN and the best of "SelfAtt-Graph" and "SelfAtt-Stack" for Self-Attention. Figure 2 shows the score differences of encoding and decoding architectures against the languages' distances to English. For both the encoding and decoding module, we observe a similar overall pattern: the order-free models, in general, perform better than order-sensitive ones in the languages that are distant from the source language English. On the other hand, for some languages that are closer to English, order-sensitive models perform better, possibly benefiting from being able to capture similar word ordering information. The performance gap between order-free and order-sensitive models are positively correlated with language distance.

4.3.3 Performance Breakdown by Types

Moreover, we compare the results on specific dependency types using concrete examples. For each type, we sort the languages by their relative frequencies of left-direction (modifier before head) and plot the performance differences for encoders and decoders. We highlight the source language English in green. Figure 3 shows four typical example types: Adposition and Noun, Adjective and Noun, Auxiliary and Verb, and Object and Verb. In Figure 3a, we examine the "case" dependency type between adpositions and nouns. The pattern is similar to the overall pattern. For languages that mainly use prepositions as in English, different models perform similarly, while for languages that use postpositions, order-free models get better results. The patterns of adjective modifier (Figure 3b) and auxiliary (Figure 3c) are also similar.

On dependencies between verbs and object nouns, although in general order-free models perform better, the pattern diverges from what we expect. There can be several possible explanations for this. Firstly, the tokens which are noun objects of verbs only take about 3.1% on average over all tokens. Considering just this specific dependency type, the correlation between frequency distances and performance differences is 0.64, which is far

d	English	Average
<-2	14.36	12.93
-2	15.45	11.83
-1	31.55	30.42
1	7.51	14.22
2	9.84	10.49
>2	21.29	20.11

Table 4: Relative frequencies (%) of dependency distances. English differs from the Average at d=1.

less than 0.9 when considering all types. Therefore, although Verb-Object ordering is a typical example, we cannot take it as the whole story of word order. Secondly, Verb-Object dependencies can often be difficult to decide. They sometimes are long-ranged and have complex interactions with other words. Therefore, merely reducing modeling order information can have complicated effects. Moreover, although our relative-position self-attention encoder does not explicitly encode word positions, it may still capture some positional information with relative distances. For example, the words in the middle of a sentence will have different distance patterns from those at the beginning or the end. With this knowledge, the model can still prefer the pattern where a verb is in the middle as in English's Subject-Verb-Object ordering and may find sentences in Subject-Object-Verb languages strange. It will be interesting to explore more ways to weaken or remove this bias.

4.3.4 Analysis on Dependency Distances

We now look into dependency lengths and directions. Here, we combine dependency length and direction into dependency distance d, by using negative signs for dependencies with left-direction (modifier before head) and positive for right-direction (head before modifier). We find a seemingly strange pattern at dependency distances $|d|$=1: for all transfer models, evaluation scores on d=-1 can reach about 80, but on d=1, the scores are only around 40. This may be explained by the relative frequencies of dependency distances as shown in Table 4, where there is a discrepancy between English and the average of other languages at d=1. About 80% of the dependencies with $|d|$=1 in English is the left direction (modifier before head), while overall other languages have more right directions at $|d|$=1. This suggests an interesting future direction of training on more source languages with different dependency distance distributions.

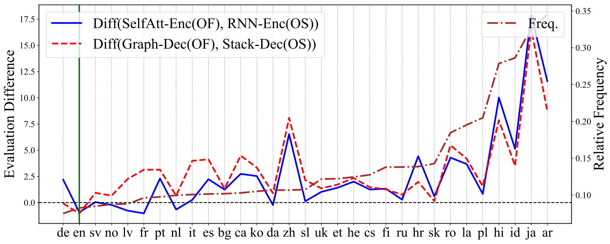


Figure 4: Evaluation differences of models on d=1 dependencies. Annotations are the same as in Figure 3, languages are sorted by percentages (represented by the brown curve and right y-axis) of d=1 dependencies.

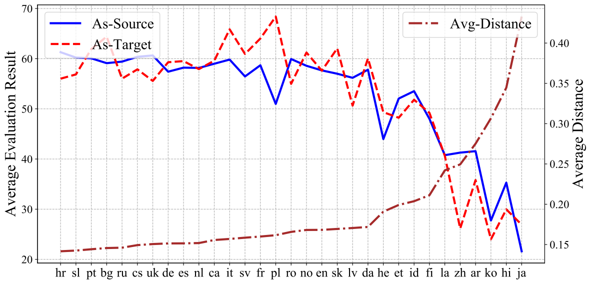


Figure 5: Transfer performance of all source-target language pairs. The blue and red curves show the averages over columns and over rows of the source-target pair performance matrix (see text for details). The brown curve and the right y-axis legend represent the average language distance between one language and all others.

We further compare the four models on the d=1 dependencies and as shown in Figure 4, the familiar pattern appears again. The order-free models perform better at the languages which have more d=1 dependencies. Such finding indicates that our model design of reducing the ability to capture word order information can help on short-ranged dependencies of different directions to the source language. However, the improvements are still limited. One of the most challenging parts of unsupervised cross-lingual parsing is modeling cross-lingually shareable and language-unspecific information. In other words, we want flexible yet powerful models. Our exploration of the order-free self-attentive models is the first step.

4.3.5 Transfer between All Language Pairs

Finally, we investigate the transfer performance of all source-target language pairs.[9] We first generate a performance matrix A, where each entry (i, j) records the transfer performance from a source language i to a target language j. We then report the following two aggregate perfor-

---

[9] Because the size of training corpus for each language is different in UD, to compare among languages, we train a parser on the first 4,000 sentences for each language and evaluate its transfer performance on all other languages.

mance measures on A in Figure 5: 1) *As-source* reports the average over columns of A for each row of the source language and 2) *As-target* reports the average over rows of A for each column of the target language. As a reference, we also plot the average word-order distance between one language to other languages. Results show that both *As-source* (blue line) and *As-target* (red line) highly are anti-correlated (Pearson correlation coefficients are -0.90 and -0.87, respectively) with average language distance (brown line).

5 Related Work

Cross-language transfer learning employing deep neural networks has widely been studied in the areas of natural language processing (Ma and Xia, 2014; Guo et al., 2015; Kim et al., 2017; Kann et al., 2017; Cotterell and Duh, 2017), speech recognition (Xu et al., 2014; Huang et al., 2013), and information retrieval (Vulić and Moens, 2015; Sasaki et al., 2018; Litschko et al., 2018). Learning the language structure (e.g., morphology, syntax) and transferring knowledge from the source language to the target language is the main underneath challenge, and has been thoroughly investigated for a wide variety of NLP applications, including sequence tagging (Yang et al., 2016; Buys and Botha, 2016), name entity recognition (Xie et al., 2018), dependency parsing (Tiedemann, 2015; Agić et al., 2014), entity coreference resolution and linking (Kundu et al., 2018; Sil et al., 2018), sentiment classification (Zhou et al., 2015, 2016b), and question answering (Joty et al., 2017).

Existing work on unsupervised cross-lingual dependency parsing, in general, trains a dependency parser on the source language and then directly run on the target languages. Training of the monolingual parsers are often delexicalized, i.e., removing all lexical features from the source treebank (Zeman and Resnik, 2008; McDonald et al., 2013), and the underlying feature model is selected from a shared part-of-speech (POS) representation utilizing the Universal POS Tagset (Petrov et al., 2012). Another pool of prior work improves the delexicalized approaches by adapting the model to fit the target languages better. Cross-lingual approaches that facilitate the usage of lexical features includes choosing the source language data points suitable for the target language (Søgaard, 2011; Täckström et al., 2013), transferring from multiple sources (McDonald et al., 2011; Guo et al., 2016; Täckström et al., 2013), using cross-lingual word clusters (Täckström et al., 2012) and lexicon mapping (Xiao and Guo, 2014; Guo et al., 2015). In this paper, we consider single-source transfer–train a parser on a single source language, and evaluate it on the target languages to test the transferability of neural architectures.

Multilingual transfer (Ammar et al., 2016; Naseem et al., 2012; Zhang and Barzilay, 2015) is another broad category of techniques applied to parsing where knowledge from many languages having a common linguistic typology is utilized. Recent works (Aufrant et al., 2016; Wang and Eisner, 2018a,b) demonstrated the significance of explicitly extracting and modeling linguistic properties of the target languages to improve cross-lingual dependency parsing. Our work is different in that we focus on the neural architectures and explore their influences on cross-lingual transfer.

6 Conclusion

In this work, we conduct a comprehensive study on how the design of neural architectures affects cross-lingual transfer learning. We examine two notable families of neural architectures (sequential RNN v.s. self-attention) using dependency parsing as the evaluation task. We show that *order-free* models perform better than *order-sensitive* ones when there is a significant difference in the word order typology between the target and source language. In the future, we plan to explore multi-source transfer and incorporating prior linguistic knowledge into the models for better cross-lingual transfer.

Acknowledgments

We thank anonymous reviewers for their helpful feedback. We thank Robert Östling for reaching out when he saw the earlier arxiv version of the paper and providing insightful comments about word order and related citations. We are grateful for the Stanford NLP group's comments and feedback when we present the preliminary results in their seminar. We thank Graham Neubig and the MT/Multilingual Reading Group at CMU-LTI for helpful discussions. We also thank USC Plus Lab and UCLA-NLP group for discussion and comments. This work was supported in part by National Science Foundation Grant IIS-1760523.

References

Željko Agić, Jörg Tiedemann, Kaja Dobrovoljc, Simon Krek, Danijela Merkler, and Sara Može. 2014. Cross-lingual dependency parsing of related languages with rich morphosyntactic tagsets. In *EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics* 4:431–444.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource dependency parsing: Boosting delexicalized cross-lingual transfer with linguistic knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 119–130.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1954–1964.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 91–96.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Internation Conference on Learning Representations* .

Matthew S Dryer. 2007. Word order. *Language typology and syntactic description* 1:61–131.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 340–345.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP*. Association for Computational Linguistics, pages 10–18.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1234–1244.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16, pages 2734–2740.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1923–1933.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 7304–7308.

Zhanming Jie, Aldrian Obaja Muis, and Wei Lu. 2017. Efficient dependency-guided named entity recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 3457–3465.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. pages 226–237.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* page 19932003.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2832–2838.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2676–2686.

Gourab Kundu, Avi Sil, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual coreference resolution and its application to entity linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 395–400.

Robert Litschko, Goran Glavaš, Simone Paolo Ponzetto, and Ivan Vulić. 2018. Unsupervised cross-lingual information retrieval using monolingual data only. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18, pages 1253–1256.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 8–14.

Haitao Liu. 2010. Dependency direction as a means of word-order typology: A method based on dependency treebanks. *Lingua* 120(6):1567–1578.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *Internation Conference on Learning Representations* .

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL 2014*. Baltimore, Maryland, pages 1337–1348.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 1626–1635.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*. Ann Arbor, Michigan, USA, pages 91–98.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 92–97.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 62–72.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 629–637.

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 205–211.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 2227–2237.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC-2012*. Istanbul, Turkey, pages 2089–2096.

Shota Sasaki, Shuo Sun, Shigehiko Schamoni, Kevin Duh, and Kentaro Inui. 2018. Cross-lingual learning-to-rank with shared representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. volume 2, pages 458–463.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, pages 464–468.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. pages 5464–5472.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *Internation Conference on Learning Representations* .

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 682–686.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1061–1071.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1061–1071.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*. Association for Computational Linguistics, pages 477–487.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*. pages 340–349.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, pages 363–372.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with supervised learning. *Transactions of the Association for Computational Linguistics* 5:147–161.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics (TACL)* .

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 1325–1337.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. pages 119–129.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 369–379.

Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. 2014. Cross-language transfer learning for deep neural network based speech enhancement. In *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*. IEEE, pages 336–340.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270* .

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, pages 1–21.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1857–1867.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 430–440.

Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W. Tsang, and Shen-Shyang Ho. 2016a. Transfer learning for cross-language text categorization through active correspondences construction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16, pages 2400–2406.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1403–1412.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1568–1575.

A Multi-Task Approach for Disentangling Syntax and Semantics in Sentence Representations

Mingda Chen **Qingming Tang** **Sam Wiseman** **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
`{mchen,qmtang,swiseman,kgimpel}@ttic.edu`

Abstract

We propose a generative model for a sentence that uses two latent variables, with one intended to represent the syntax of the sentence and the other to represent its semantics. We show we can achieve better disentanglement between semantic and syntactic representations by training with multiple losses, including losses that exploit aligned paraphrastic sentences and word-order information. We also investigate the effect of moving from bag-of-words to recurrent neural network modules. We evaluate our models as well as several popular pretrained embeddings on standard semantic similarity tasks and novel syntactic similarity tasks. Empirically, we find that the model with the best performing syntactic and semantic representations also gives rise to the most disentangled representations.[1]

1 Introduction

As generative latent variable models, especially of the continuous variety (Kingma and Welling, 2014; Goodfellow et al., 2014), have become increasingly important in natural language processing (Bowman et al., 2016; Gulrajani et al., 2017), there has been increased interest in learning models where the latent representations are disentangled (Hu et al., 2017). Much of the recent NLP work on learning disentangled representations of text has focused on disentangling the representation of attributes such as sentiment from the representation of content, typically in an effort to better control text generation (Shen et al., 2017; Zhao et al., 2017; Fu et al., 2018).

In this work, we instead focus on learning sentence representations that disentangle the syntax and the semantics of a sentence. We are moreover interested in disentangling these representations not for the purpose of controlling generation, but for the purpose of calculating semantic or syntactic similarity between sentences (but not both). To this end, we propose a generative model of a sentence which makes use of both semantic and syntactic latent variables, and we evaluate the induced representations on both standard semantic similarity tasks and on several novel syntactic similarity tasks.

We use a deep generative model consisting of von Mises Fisher (vMF) and Gaussian priors on the semantic and syntactic latent variables (respectively) and a deep bag-of-words decoder that conditions on these latent variables. Following much recent work, we learn this model by optimizing the ELBO with a VAE-like (Kingma and Welling, 2014; Rezende et al., 2014) approach.

Our learned semantic representations are evaluated on the SemEval semantic textual similarity (STS) tasks (Agirre et al., 2012; Cer et al., 2017). Because there has been less work on evaluating syntactic representations of sentences, we propose several new syntactic evaluation tasks, which involve predicting the syntactic analysis of an unseen sentence to be the syntactic analysis of its nearest neighbor (as determined by the latent syntactic representation) in a large set of annotated sentences.

In order to improve the quality and disentanglement of the learned representations, we incorporate simple additional losses in our training, which are designed to force the latent representations to capture different information. In particular, our semantic multi-task losses make use of aligned paraphrase data, whereas our syntactic multi-task loss makes use of word-order information. Additionally, we explore different encoder and decoder architectures for learning better syntactic representations. Experimentally, we find that by training in this way we are able to force the learned represen-

---

[1] Code and data are available at `github.com/mingdachen/disentangle-semantics-syntax`

Proceedings of NAACL-HLT 2019, pages 2453–2464
Minneapolis, Minnesota, June 2 - June 7, 2019.

tations to capture different information (as measured by the performance gap between the latent representations on each task). Moreover, we find that we achieve the best performance on all tasks when the learned representations are most disentangled.

2 Related Work

There is a growing amount of work on learning interpretable or disentangled latent representations both in machine learning (Tenenbaum and Freeman, 2000; Reed et al., 2014; Makhzani et al., 2015; Mathieu et al., 2016; Higgins et al., 2016; Chen et al., 2016; Hsu et al., 2017) and in various NLP applications, including sentence sentiment and style transfer (Hu et al., 2017; Shen et al., 2017; Fu et al., 2018; Zhao et al., 2018, *inter alia*), morphological reinflection (Zhou and Neubig, 2017), semantic parsing (Yin et al., 2018), text generation (Wiseman et al., 2018), and sequence labeling (Chen et al., 2018). Another related thread of work is text-based variational autoencoders (Miao et al., 2016; Bowman et al., 2016; Serban et al., 2017; Xu and Durrett, 2018).

In terms of syntax and semantics in particular, there is a rich history of work in analyzing their interplay in sentences (Jurafsky, 1988; van Valin, Jr., 2005). We do not intend to claim that the two can be entirely disentangled in distinct representations. Rather, our goal is to propose modica of knowledge via particular multi-task losses and measure the extent to which this knowledge leads learned representations to favor syntactic or semantic information from a sentence.

There has been prior work with similar goals for representations of words (Mitchell and Steedman, 2015) and bilexical dependencies (Mitchell, 2016), finding that decomposing syntactic and semantic information can lead to improved performance on semantic tasks. We find similar trends in our results, but at the level of sentence representations. A similar idea has been explored for text generation (Iyyer et al., 2018), where adversarial examples are generated by controlling syntax.

Some of our losses use sentential paraphrases, relating them to work in paraphrase modeling (Wieting et al., 2016; Wieting and Gimpel, 2018). Deudon (2018) recently proposed a variational framework for modeling paraphrastic sentences, but our focus here is on learning disentangled representations.

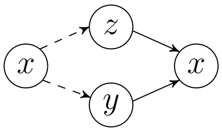


Figure 1: Graphical model of VGVAE. Dashed lines indicate inference model. Solid lines indicate generative model.

As part of our evaluation, we develop novel syntactic similarity tasks for sentence representations learned without any syntactic supervision. These evaluations relate to the broad range of work in unsupervised parsing (Klein and Manning, 2004) and part-of-speech tagging (Christodoulopoulos et al., 2010). However, our evaluations differ from previous evaluations in that we employ k-nearest-neighbor syntactic analyzers using our syntactic representations to choose nearest neighbors.

There is a great deal of work on applying multitask learning to various NLP tasks (Plank et al., 2016; Rei, 2017; Augenstein and Søgaard, 2017; Bollmann et al., 2018, *inter alia*) and, recently, as a way of improving the quality or disentanglement of learned representations (Zhao et al., 2017; Goyal et al., 2017; Du et al., 2018; John et al., 2018).

3 Proposed Approach

Our goal is to extract the disentangled semantic and syntactic information from sentence representations. To achieve this, we introduce the vMF-Gaussian Variational Autoencoder (VGVAE). As shown in Figure 1, VGVAE assumes a sentence is generated by conditioning on two independent variables: semantic variable y and syntactic variable z. In particular, our model gives rise to the following joint likelihood

$$\begin{aligned} p_\theta(x, y, z) &= p_\theta(y) p_\theta(z) p_\theta(x|y, z) \\ &= p_\theta(y) p_\theta(z) \prod_{t=1}^{T} p(x_t \,|\, y, z), \end{aligned}$$

where x_t is the tth word of x, T is the sentence length, and $p(x_t|y, z)$ is given by a softmax over a vocabulary of size V. Further details on the parameterization are given below.

To perform inference, we assume a factored posterior $q_\phi(y, z|x) = q_\phi(y|x) q_\phi(z|x)$, as has been used in prior work (Zhou and Neubig, 2017; Chen et al., 2018). Learning of VGVAE maximizes a lower bound on marginal log-likelihood:

$$
\begin{aligned}
&\log p_\theta(x) \geq \mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} [\log p_\theta(x|z,y) \\
&- \log \frac{q_\phi(z|x)}{p_\theta(z)} - \log \frac{q_\phi(y|x)}{p_\theta(y)}] \\
&= \mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} [\log p_\theta(x|z,y)] - KL(q_\phi(z|x)\|p_\theta(z)) \\
&- KL(q_\phi(y|x)\|p_\theta(y)) \stackrel{\text{def}}{=\!=} \text{ELBO}
\end{aligned} \tag{1}
$$

3.1 Parameterizations

VGVAE uses two distribution families in defining the posterior over latent variables, namely, the von Mises-Fisher (vMF) distribution and the Gaussian distribution.

vMF Distribution. vMF can be regarded as a Gaussian distribution on a hypersphere with two parameters: μ and κ. $\mu \in \mathbb{R}^m$ is a normalized vector (i.e. $\|\mu\|_2 = 1$) defining the mean direction. $\kappa \in \mathbb{R}_{\geq 0}$ is often referred to as a concentration parameter analogous to the variance in a Gaussian distribution. vMF has been used for modeling similarity between two sentences (Guu et al., 2018), which is particularly suited to our purpose here, since we will evaluate our semantic representations in the context of modeling paraphrases (See Sections 4.1 and 4.2 for more details). Therefore, we assume $q_\phi(y|x)$ follows vMF$(\mu_\alpha(x), \kappa_\alpha(x))$ and the prior $p_\theta(y)$ follows the uniform distribution vMF$(\cdot, 0)$.

With this choice of prior and posterior distribution, the $KL(q_\phi(y|x)\|p_\theta(y))$ appearing in the ELBO can be computed in closed-form:

$$
\begin{aligned}
&\kappa_\alpha \frac{\mathcal{I}_{m/2}(\kappa_\alpha)}{\mathcal{I}_{m/2-1}(\kappa_\alpha)} + (m/2-1)\log\kappa_\alpha - \\
&(m/2)\log(2\pi) - \log\mathcal{I}_{m/2-1}(\kappa_\alpha) + \\
&\frac{m}{2}\log\pi + \log 2 - \log\Gamma(\frac{m}{2}),
\end{aligned} \tag{2}
$$

where $\mathcal{I}_v$ is the modified Bessel function of the first kind at order v and $\Gamma(\cdot)$ is the Gamma function. We follow Davidson et al. (2018) and use an acceptance-rejection scheme to sample from vMF.

Gaussian Distribution.[2] We assume $q_\phi(z|x)$ follows a Gaussian distribution $\mathcal{N}(\mu_\beta(x), \text{diag}(\sigma_\beta(x)))$ and that the prior $p_\theta(z)$ is $\mathcal{N}(0, I_d)$, where I_d is an $d \times d$ identity matrix. Since we only consider a diagonal covariance matrix, the KL divergence term $KL(q_\phi(z|x)\|p_\theta(z))$ can also be computed efficiently:

$$
\frac{1}{2}(-\sum_i \log\sigma_{\beta i} + \sum_i \sigma_{\beta i} + \sum_i \mu_{\beta i}^2 - d) \tag{3}
$$

Inference and Generative Models. The inference models $q_\phi(y|x)$ and $q_\phi(z|x)$ are two independent word averaging encoders with additional linear feedforward neural networks for producing $\mu(x)$ and $\sigma(x)$ (or $\kappa(x)$). The generative model $p_\theta(x|y,z)$ is a feedforward neural network g_θ with the output being a bag of words. In particular, the expected output log-probability (the first term in Eq. 1) is computed as follows:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} [\log p_\theta(x|y,z)] = \\
&\mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} \left[\sum_{t=1}^{T} \log \frac{\exp g_\theta([y;z])_{x_t}}{\sum_{j=1}^{V} \exp g_\theta([y;z])_j}\right]
\end{aligned}
$$

Where V is the vocabulary size, $[;]$ indicates concatenation, T is the sentence length and x_t is the index of the t'th word's word type.

Recurrent Neural Networks. To facilitate better learning of syntax, we also consider replacing both the generative and inference models with RNN-based sequence models, rather than bag-of-words models. In this setting, the generative model $p_\theta(x|y,z)$ is a unidirectional long-short term memory network (LSTM; Hochreiter and Schmidhuber, 1997) and a linear feedforward neural network for predicting the word tokens (shown in Figure 2). The expected output log-probability is computed as follows:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} [\log p_\theta(x|y,z)] = \\
&\mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x)\\ z\sim q_\phi(z|x)}} \left[\sum_{t=1}^{T} \log p_\theta(x_t|y,z,x_{1:t-1})\right]
\end{aligned}
$$

Where V is the vocabulary size, T is the sentence length and x_t is the index of the t'th word's word type.

The inference model $q_\phi(y|x)$ is still a word averaging encoder, but $q_\phi(z|x)$ is parameterized by

---

[2] In preliminary experiments, we observed that using two distribution families can lead to better performance. This is presumably because the Gaussian distribution complements the norm information lost in the vMF distribution.

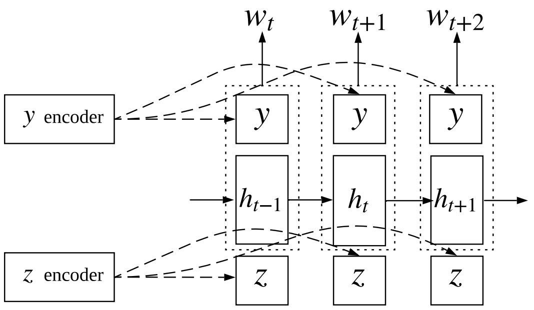


Figure 2: Diagram showing LSTM decoder that uses the semantic variable y and the syntactic variable z.

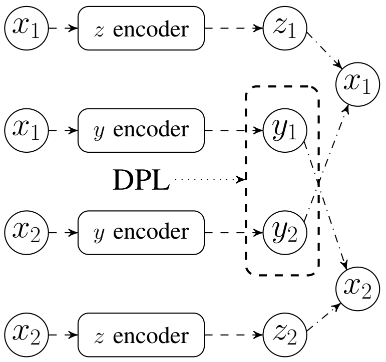


Figure 3: Diagram showing the training process when using the discriminative paraphrase loss (DPL; dotted lines) and paraphrase reconstruction loss (PRL; dash-dotted lines). The pair (x_1, x_2) is a sentential paraphrase pair, the y's are the semantic variables corresponding to each x, and the z's are syntactic variables.

a bidirectional LSTM, where we concatenate the forward and backward hidden states and then take the average. The output of the LSTM is then used as input to a feedforward network with one hidden layer for producing $\mu(x)$ and $\sigma(x)$ (or $\kappa(x)$).

In the following sections, we will introduce several losses that will be added into the training of our base model, which empirically shows the ability of further disentangling the functionality between the semantic variable y and the syntactic variable z.

4 Multi-Task Training

We attempt to improve the quality and disentanglement of our semantic and syntactic representations by introducing additional losses, which encourage y to capture semantic information and z to capture syntactic information. We elaborate on these losses below.

4.1 Paraphrase Reconstruction Loss

Our first loss is a paraphrase reconstruction loss (PRL). The key assumption underlying the PRL is that for a paraphrase pair x_1, x_2, the semantic information is equivalent between the two sentences and only the syntactic information varies. To impose such constraints, PRL is defined as

$$\begin{aligned}&\underset{\substack{y_2 \sim q_\phi(y|x_2)\\ z_1 \sim q_\phi(z|x_1)}}{\mathbb{E}}[-\log p_\theta(x_1|y_2, z_1)]+\\ &\underset{\substack{y_1 \sim q_\phi(y|x_1)\\ z_2 \sim q_\phi(z|x_2)}}{\mathbb{E}}[-\log p_\theta(x_2|y_1, z_2)]\end{aligned} \tag{4}$$

That is, we swap the semantic variables, keep the syntactic variables, and attempt to reconstruct the sentences (shown in Figure 3). While instead of using a multi-task objective we could directly model paraphrases x_1 and x_2 as being generated by the same y (which naturally suggests a product-of-experts style posterior, as in Wu and Goodman (2018)), we found that for the purposes of our downstream tasks training with the multi-task loss gave superior results.

4.2 Discriminative Paraphrase Loss

Our second loss is a discriminative paraphrase loss (DPL). The DPL explicitly encourages the similarity of paraphrases x_1, x_2 to be scored higher than the dissimilar sentences n_1, n_2 (i.e., negative samples; see Sec. 5 for more details) by a given margin δ. As shown in Figure 3, the similarity function in this loss only uses the semantic variables in the sentences. The loss is defined as

$$\begin{aligned}&\max(0, \delta - d(x_1, x_2) + d(x_1, n_1))+\\ &\max(0, \delta - d(x_1, x_2) + d(x_2, n_2))\end{aligned} \tag{5}$$

The similarity function we choose is the cosine similarity between the mean directions of the semantic variables from the two sentences:

$$d(x_1, x_2) = \text{cosine}(\mu_\alpha(x_1), \mu_\alpha(x_2)) \tag{6}$$

4.3 Word Position Loss

It has been observed in previous work that word order typically contributes little to the modelling of semantic similarity (Wieting et al., 2016). We interpret this as evidence that word position information is more relevant to syntax than semantics, at least as evaluated by STS tasks. To guide the syntactic variable to represent word order, we introduce a word position loss (WPL). Although our word averaging encoders only have access to the bag of words of the input, using this loss can be viewed as a denoising autoencoder where we have maximal input noise (i.e., an orderless representation of the input) and the encoders need to learn to reconstruct the ordering. For both word averaging encoders and LSTM encoders, WPL is parameterized by a three-layer feedforward neural network $f(\cdot)$ with input from the concatenation of the samples of the syntactic variable z and the embedding vector e_i at input position i; we then attempt to predict a one-hot vector representing the position i. More specifically, we define

$$\mathrm{WPL} \overset{\mathrm{def}}{=\!=} \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \left[-\sum_i \log \mathrm{softmax}(f([e_i; z]))_i \right]$$

where $\mathrm{softmax}(\cdot)_i$ indicates the probability at position i.

5 Training

KL Weight. Following previous work on VAEs (Higgins et al., 2016; Alemi et al., 2016), we attach a weight to the KL divergence and tune it based on development set performance.

Negative Samples. When applying DPL, we select negative samples based on maximizing cosine similarity to sentences from a subset of the data. In particular, we accumulate k mini-batches during training, yielding a "mega-batch" $\mathcal{S}$ (Wieting and Gimpel, 2018). Then the negative samples are selected based on the following criterion:

$$n_1 = \mathop{\mathrm{argmax}}_{n\in\mathcal{S}\wedge n\neq x_2} \mathrm{cosine}(\mu_\alpha(x_1), \mu_\alpha(n))$$

where x_1, x_2 forms the paraphrase pair and the mega-batch size is fixed to $k = 20$ for all of our experiments. Since all of our models are trained from scratch, we observed some instabilities with DPL during the initial stages of training. We suspect that this is because the negative samples at these initial stages are of low quality. To overcome this issue, DPL is included starting at the second epoch of training so that the models can have a warm start.

6 Experiments

6.1 Setup

We subsampled half a million paraphrase pairs from ParaNMT-50M (Wieting and Gimpel, 2018) as our training set. We use SemEval semantic textual similarity (STS) task 2017 (Cer et al., 2017) as a development set. For semantic similarity evaluation, we use the STS tasks from 2012 to 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016) and the STS benchmark test set (Cer et al., 2017). For evaluating syntactic similarity, we propose several evaluations. One uses the gold parse trees from the Penn Treebank (Marcus et al., 1993), and the others are based on automatically tagging and parsing five million paraphrases from ParaNMT-50M; we describe these tasks in detail below.

For hyperparameters, the dimensions of the latent variables are 50. The dimensions of word embeddings are 50. We use cosine similarity as similarity metric for all of our experiments. We tune the weights for PRL and reconstruction loss from 0.1 to 1 in increments of 0.1 based on the development set performance. We use one sample from each latent variable during training. When evaluating VGVAE based models on STS tasks, we use the mean direction of the semantic variable y, while for syntactic similarity tasks, we use the mean vector of the syntactic variable z.

6.2 Baselines

Our baselines are a simple word averaging (WORDAVG) model and bidirectional LSTM averaging (BLSTMAVG) model, both of which have been shown to be very competitive for modeling semantic similarity when trained on paraphrases (Wieting and Gimpel, 2018). Specifically, WORDAVG takes the average over the word embeddings in the input sequence to obtain the sentence representation. BLSTMAVG uses the averaged hidden states of a bidirectional LSTM as the sentence representation, where forward and backward hidden states are concatenated. These models use 50 dimensional word embeddings and 50 dimensional LSTM hidden vectors per direction. These baselines are trained with DPL only. Additionally, we scramble the input sentence for BLSTMAVG since it has been reported beneficial for its performance in semantic similarity tasks (Wieting and Gimpel, 2017).

We also benchmark several pretrained embeddings on both semantic similarity and syntactic similarity datasets, including GloVe (Pennington et al., 2014),[3] SkipThought (Kiros et al., 2015),[4]

---

[3] We use 300 dimensional Common Crawl embeddings available at `nlp.stanford.edu/projects/glove`

[4] `github.com/ryankiros/skip-thoughts`

	semantic var.		syntactic var.	
	bm	avg	bm	avg
GloVe	39.0	48.7	-	-
SkipThought	42.1	42.0	-	-
InferSent	67.8	61.0	-	-
ELMo	57.7	60.3	-	-
BERT	4.5	15.0	-	-
WORDAVG	71.9	64.8	-	-
BLSTMAVG	71.4	64.4	-	-
VGVAE	45.5	42.7	40.8	43.2
VGVAE + WPL	51.5	49.3	28.1	31.0
VGVAE + DPL	68.4	58.2	37.8	40.5
VGVAE + PRL	67.9	57.8	29.6	32.7
VGVAE + PRL + WPL	69.8	61.3	23.2	27.9
VGVAE + PRL + DPL	71.2	64.2	31.7	33.9
VGVAE + DPL + WPL	71.0	63.5	24.1	29.0
ALL	72.3	65.1	20.1	24.2
ALL + LSTM enc.	72.5	65.1	16.3	24.5
ALL + LSTM enc. & dec.	**72.9**	**65.5**	**11.3**	**19.3**

Table 1: Pearson correlation (%) for STS test sets. bm: STS benchmark test set. avg: the average of Pearson correlation for each domain in the STS test sets from 2012 to 2016. Results are in bold if they are highest in the "semantic variable" columns or lowest in the "syntactic variable" columns. "ALL" indicates all of the multi-task losses are used.

InferSent (Conneau et al., 2017),[5] ELMo (Peters et al., 2018),[6] and BERT (Devlin et al., 2018).[7] For GloVe, we average word embeddings to form sentence embeddings. For ELMo, we average the hidden states from three layers and then average the hidden states across time steps. For BERT, we use the averaged hidden states from the last attention block.

7 Results and Analysis

7.1 Semantic Similarity

As shown in Table 1, the semantic and syntactic variables of our base VGVAE model show similar performance on the STS test sets. As we begin adding multi-task losses, however, the performance of these two variables gradually diverges, indicating that different information is being captured in the two variables. More interestingly, note that when *any* of the three losses is added to the base VGVAE model (even the WPL loss which makes no use of paraphrases), the performance of the semantic variable increases and the performance of the syntactic variable decreases; this suggests that each loss is useful in encouraging the latent variables to learn complementary information.

Indeed, the trend of additional losses both increasing semantic performance and decreasing syntactic performance holds even as we use more than two losses, except for the single case of VGVAE + PRL + DPL, where the syntactic performance increases slightly. Finally, we see that when the bag-of-words VGVAE model is used with all of the multi-task losses ("ALL"), we observe a large gap between the performance of the semantic and syntactic latent variables, as well as strong performance on the STS tasks that outperforms all baselines.

Using LSTM modules further strengthens the disentanglement between the two variables and leads to even better semantic performance. While using an LSTM encoder and a bag-of-words decoder is difficult to justify from a generative modeling perspective, we include results with this configuration to separate out the contributions of the LSTM encoder and decoder.

7.2 Syntactic Similarity

So far, we have only confirmed empirically that the syntactic variable has learned to *not* capture semantic information. To investigate what the syntactic variable has captured, we propose several syntactic similarity tasks.

In particular, we consider using the syntactic latent variable in calculating nearest neighbors for a 1-nearest-neighbor syntactic parser or part-of-speech tagger. We use our latent variables to define the similarity function in these settings and evaluate the quality of the output parses and tag sequences using several metrics.

Our first evaluation involves constituency parsing, and we use the standard training and test splits from the Penn Treebank. We predict a parse tree for each sentence in the test set by finding its nearest neighbor in the training set based on the cosine similarity of the mean vectors for the syntactic variables. The parse tree of the nearest neighbor will then be treated as our prediction for the test sentence. Since the train and test sentences may differ in length, standard parse evaluation metrics are not applicable, so we use tree edit distance (Zhang and Shasha, 1989)[8] to compute the distance between two parse tree without consider-

---

[5] We use model V1 available at `github.com/facebookresearch/InferSent`

[6] We use the original model available at `allennlp.org/elmo`

[7] We use bert-large-uncased available at `github.com/huggingface/pytorch-pretrained-BERT`

[8] `github.com/timtadh/zhang-shasha`

	Constituent Parsing (TED, $\downarrow$)		Constituent Parsing (F_1, $\uparrow$)		POS Tagging (%Acc., $\uparrow$)	
GloVe	120.8		27.3		23.9	
SkipThought	99.5		30.9		29.6	
InferSent	138.9		28.0		25.1	
ELMo	103.8		30.4		27.8	
BERT	101.7		28.6		25.4	
Random baseline	121.4		19.2		12.9	
Upper bound performance	51.6		71.1		62.3	
WordAvg	107.0		25.5		21.4	
BLSTMAvg	106.8		25.7		21.6	
	semantic var.	syntactic var.	semantic var.	syntactic var.	semantic var.	syntactic var.
VGVAE	109.3	111.4	25.2	25.0	21.1	21.0
VGVAE + WPL	112.3	105.9	**24.1**	28.2	**20.3**	24.2
VGVAE + DPL	108.1	110.6	25.1	26.1	21.3	21.8
VGVAE + PRL	111.9	110.9	24.7	26.9	21.0	22.2
VGVAE + DPL + WPL	111.2	105.0	25.1	28.8	21.5	24.6
VGVAE + PRL + DPL	108.0	110.4	25.0	26.2	21.1	22.1
VGVAE + PRL + WPL	109.4	105.1	24.4	28.1	20.6	23.6
ALL	110.0	104.7	25.4	29.3	21.4	25.5
ALL + LSTM enc.	112.0	101.0	25.7	37.3	22.1	34.0
ALL + LSTM enc. & dec.	**114.6**	**100.5**	25.3	**38.8**	21.4	**35.7**

Table 2: Syntactic similarity evaluations, showing tree edit distance (TED) and labeled F_1 score for constituent parsing, and accuracy (%) for part-of-speech tagging. Numbers are bolded if they are worst in the "semantic variable" column or best in the "syntactic variable" column. "ALL" indicates all the multi-task losses are used.

ing word tokens.

To better understand the difficulty of this task, we introduce two baselines. The first randomly selects a training sentence. We calculate its performance by running it ten times and then reporting the average. We also report the upper bound performance given the training set. Since computing tree edit distance is time consuming, we subsample 100 test instances and compute the minimum tree edit distance for each sampled instance. Thus, this number can be seen as the approximated upper bound performance for this task given the training set.

To use a more standard metric for these syntactic similarity tasks, we must be able to retrieve training examples with the same number of words as the sentence we are trying to parse. We accordingly parse and tag the five million paraphrase subset of the ParaNMT training data using Stanford CoreNLP (Manning et al., 2014). To form a test set, we group sentences in terms of sentence length and subsample 300 sentences for each sentence length. After removing the paraphrases of the sentences in the test set, we use the rest of the training set as candidate sentences for nearest neighbor search, and we restrict nearest neighbors to have the same sentence length as the sentence we are attempting to parse or tag, which allows us to use standard metrics like labeled F_1 score and tagging accuracy for evaluation.

7.2.1 Results

As shown in Table 2, the syntactic variables and semantic variables demonstrate similar trends across these three syntactic tasks. Interestingly, both DPL and PRL help to improve the performance of the syntactic variables, even though these two losses are only imposed on the semantic variables. We saw an analogous pattern in Table 1, which again suggests that by pushing the semantic variables to learn information shared by paraphrastic sentences, we also encourage the syntactic variables to capture complementary syntactic information. We also find that adding WPL brings the largest improvement to the syntactic variable, and keeps the syntactic information carried by the semantic variables at a relatively low level. Finally, when adding all three losses, the syntactic variable shows the strongest performance across the three tasks.

In addition, we observe that the use of the LSTM encoder improves syntactic performance by a large margin and the LSTM decoder improves further, which suggests that the use of the LSTM decoder contributes to the amount of syntactic information represented in the syntactic variable.

Among pretrained representations, SkipThought shows the strongest performance overall and ELMo has the second best performance in the last two columns. While InferSent performs worst in the first column, it gives reasonable performance for the other two. BERT performs

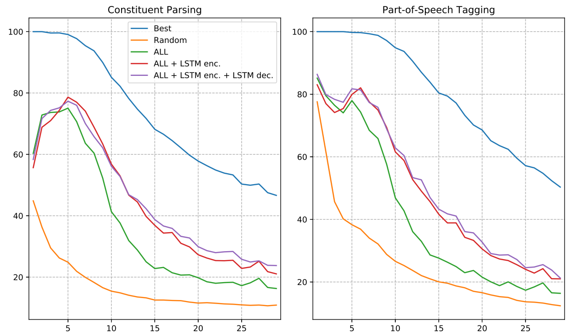


Figure 4: Constituency parsing F_1 scores and part-of-speech tagging accuracies by sentence length, for 1-nearest neighbor parsers based on semantic and syntactic variables, as well as a random baseline and an oracle nearest neighbor parser ("Best").

relatively well in the first column but worse in the other two.

To investigate the performance gap between the bag-of-words VGVAE and VGVAE with LSTM modules, in Figure 4 we plot the performance of our models and baselines as the length of the target sentence increases. We see that performance in all settings degrades as the sentences get longer. This may be due to the fact that the data is much sparser as sentence length increases (leaving fewer candidate nearest neighbors for prediction). We also see that above 4 words or so the performance gap between the bag-of-words VGVAE and VGVAE with LSTM modules becomes more and more obvious. This may be because the bag-of-words encoder has a harder time capturing syntactic information as sentence length increases. In addition, there is a slight improvement from using an LSTM decoder when the sentence length increases beyond 12 or so, which suggests that a bag-of-words decoder may struggle to capture certain parts of the syntactic information in the sentence, even when using an LSTM encoder.

7.3 Qualitative Analysis

To qualitatively evaluate our latent variables, we find (via cosine similarity) nearest neighbor sentences to test set examples in terms of both the semantic and syntactic representations. We also find nearest neighbors of words (which we view as single-word sentences). We discuss the results of this analysis below.

7.3.1 Lexical Analysis

Table 3 shows word nearest neighbors for both syntactic and semantic representations. We see that the most similar words found by the syntactic variable share the same part-of-speech tags with the query words. For example, "starting" is close to "getting" and "taking," even though these words are not semantically similar. Words retrieved according to the semantic variable, however, are more similar semantically, e.g., "begin" and "starts". As another example, "times" is similar to words that are either related to descriptions of frequency (e.g., "twice" and "often") or related to numbers (e.g., "thousand", "seven").

7.3.2 Sentential Analysis

As shown in Table 4, sentences that are similar in terms of their semantic variables tend to have similar semantics. However, sentences that are similar in terms of their syntactic variables are mostly semantically unrelated but have similar surface forms. For example, "you 're gon na save her life ." has the same meaning as "you will save her ." while having a similar syntactic structure to "you 're gon na give a speech ." (despite having very different meanings). As another example, although the semantic variable does not find a good match for "i have much more colours at home .", which can be attributed to the limited size of candidate sentences, the nearest syntactic neighbor ("you have a beautiful view from here .") has a very similar syntactic structure to the query sentence.

8 Discussion

In this paper we explored simple methods to disentangle syntax and semantics in latent representations of sentences. One goal was to measure the impact of simple decisions on the disentanglement of both the semantic and syntactic variables, even when restricting ourselves to simplified bag-of-words encoders. Due to the constrained nature of these bag-of-words models, we found that it was important to use different word embedding spaces for the semantic and syntactic encoders. In preliminary experiments, we experimented with the use of the same word embedding space but distinct feed-forward layers in the two latent variable encoders. However, this setting proved extremely difficult to achieve a disentanglement between syntax and semantics. Hence an important component of disentanglement with these bag-of-words encoders is the use of different word embedding spaces.

starting	*syntactic:* getting heading sitting chasing taking require trying sharing bothering pushing paying *semantic:* begin start stopping forward rising wake initial starts goes started again getting beginning
area	*syntactic:* engines certificate guests bottle responsibility lesson pieces suit bags vessel applications *semantic:* sector location zone fields rooms field places yard warehouse seats coordinates territory
considered	*syntactic:* stable limited odd scary classified concerned awful purple impressive embarrassing jealous *semantic:* thought assumed regard reasons wished understood purposes seemed expect guessed meant
jokes	*syntactic:* gentlemen photos finding baby missile dna parent shop murder science recognition sheriff *semantic:* funny humor prize stars cookie paradise dessert worthy smile happiness thrilled ideal kidding
times	*syntactic:* princess officer wounds plan gang ships feelings user liar elements coincidence degrees pattern *semantic:* twice later thousand pages seven every once often decade forgotten series four eight day time

Table 3: Examples of the most similar words to particular query words using syntactic variable (first row) or semantic variable (second row).

Query Sentence	Semantically Similar	Syntactically Similar
i have much more colours at home .	even if there was food , would n't it be at least 300 years old ?	you have a beautiful view from here .
victor had never known darkness like it .	he had never experienced such darkness as this .	you seem like a really nice kid .
this is , uh , too serious .	but this is too serious .	it is , however , illegal discrimination .
you 're gon na save her life .	you will save her .	you 're gon na give a speech .
we 've got to get a move on .	come on , we got ta move .	you 'll have to get in there .
and that was usually the highlight of my day .	i really enjoyed it when i did it .	and yet that was not the strangest aspect of the painting .
we do need to collect our taxes somehow .	we have to earn the money we need .	now i have to do my job .
this is just such a surprise .	oh . this is a surprise .	this is just a little gain .
okay . aw , that 's so romantic .	it 's so romantic !	oh . well , that 's not good .
we 're gon na have to do something about this .	we 'll have to do something about that .	we 're gon na have to do something about yours .

Table 4: Examples of most similar sentences to particular query sentences in terms of the semantic variable or the syntactic variable.

We also conducted experiments using LSTM encoders and decoders as recurrent neural networks are a natural way to capture syntactic information in a sentence. We found this approach to give us additional benefits for both disentangling semantics and syntax and achieving better results overall. Nonetheless, we find it encouraging that even when using bag-of-words encoders, our multi-task losses are able to achieve a separation as measured by our semantic and syntactic similarity tasks.

9 Conclusion

We proposed a generative model and several losses for disentangling syntax and semantics in sentence representations. We also proposed syntactic similarity tasks for measuring the amount of disentanglement between semantic and syntactic representations. We characterized the effects of the losses as well as the use of LSTM modules on both semantic tasks and syntactic tasks. Our models achieve the best performance across both sets of similarity tasks when the latent representations are most disentangled.

Acknowledgments

We would like to thank the anonymous reviewers, NVIDIA for donating GPUs used in this research, and Google for a faculty research award to K. Gimpel that partially supported this research.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346. Association for Computational Linguistics.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 215–226. Association for Computational Linguistics.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2180–2188, USA. Curran Associates Inc.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. 2018. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*.

Michel Deudon. 2018. Learning semantic similarity in a continuous space. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 993–1004. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. 2018. Variational autoregressive decoder for neural response generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3154–3163.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of NIPS*.

Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pages 6713–6723.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, and Aaron Courville Vincent Dumoulin. 2017. Improved Training of Wasserstein GANs. In *Proceedings of NIPS*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wei-Ning Hsu, Yu Zhang, and James Glass. 2017. Unsupervised learning of disentangled and interpretable representations from sequential data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1878–1889. Curran Associates, Inc.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for text style transfer. *arXiv preprint arXiv:1808.04339*.

Daniel Jurafsky. 1988. Issues in relating syntax and semantics. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial Autoencoders. *arXiv:1511.05644*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. 2016. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA. PMLR.

Jeff Mitchell. 2016. Decomposing bilexical dependencies into semantic and syntactic vectors. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 127–136. Association for Computational Linguistics.

Jeff Mitchell and Mark Steedman. 2015. Orthogonality of syntax and semantics within distributional spaces. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1301–1310. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. 2014. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pages 1431–1439.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of ICML*.

Iulian Vlad Serban, Alexander G. Ororbia, Joelle Pineau, and Aaron Courville. 2017. Piecewise latent variables for neural variational text processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 422–432. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Joshua B Tenenbaum and William T Freeman. 2000. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283.

Robert D. van Valin, Jr. 2005. *Exploring the Syntax-Semantics Interface*. Cambridge University Press.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of International Conference on Learning Representations*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088, Vancouver, Canada. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187. Association for Computational Linguistics.

Mike Wu and Noah Goodman. 2018. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513. Association for Computational Linguistics.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–765. Association for Computational Linguistics.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2018. Adversarially Regularized Autoencoders. In *Proceedings of ICML*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320. Association for Computational Linguistics.

Self-Discriminative Learning for Unsupervised Document Embedding

Hong-You Chen^{*1}, **Chin-Hua Hu**^{*1}, **Leila Wehbe**^{2}, **Shou-De Lin**^{1}
^{1}Department of Computer Science and Information Engineering, National Taiwan University
^{2}Machine Learning Department, Carnegie Mellon University
`{b03902128, r07922028}@ntu.edu.tw,`
`lwehbe@cmu.edu, sdlin@csie.ntu.edu.tw`

Abstract

Unsupervised document representation learning is an important task providing pre-trained features for NLP applications. Unlike most previous work which learn the embedding based on self-prediction of the surface of text, we explicitly exploit the inter-document information and directly model the relations of documents in embedding space with a discriminative network and a novel objective. Extensive experiments on both small and large public datasets show the competitiveness of the proposed method. In evaluations on standard document classification, our model has errors that are relatively 5 to 13% lower than state-of-the-art unsupervised embedding models. The reduction in error is even more pronounced in scarce label setting.

1 Introduction

Rapid advance in deep methods for natural language processing has contributed to a growing need for vector representation of documents as input features. Applications for such vector representations include machine translation (Sutskever et al., 2014), text classification (Dai and Le, 2015), image captioning (Mao et al., 2015), multi-lingual document matching (Pham et al., 2015), question answering (Rajpurkar et al., 2016), and more. This work studies *unsupervised* training for encoders that can efficiently encode long paragraph of text into compact vectors to be used as pre-trained features. Existing solutions are mostly based on the assumption that a good document embedding can be learned through modeling the **intra**-document information by predicting the occurrence of terms inside the document itself. We argue that such an assumption might not be sufficient to obtain meaningful a document embedding as they do not consider **inter**-document relationships.

Traditional document representation models such as Bag-of-words (BoW) and TF-IDF show competitive performance in some tasks (Wang and Manning, 2012). However, these models treat words as flat tokens which may neglect other useful information such as word order and semantic distance. This in turn can limit the models effectiveness on more complex tasks that require deeper level of understanding. Further, BoW models suffer from high dimensionality and sparsity. This is likely to prevent them from being used as input features for downstream NLP tasks.

Continuous vector representations for documents are being developed. A successful thread of work is based on the distributional hypothesis, and use contextual information for context-word predictions. Similar to Word2Vec (Mikolov et al., 2013), PV (Le and Mikolov, 2014) is optimized by predicting the next words given their contexts in a document, but it is conditioned on a unique document vector. Word2Vec-based methods for computing document embeddings achieve state-of-the-art performance on document embedding. Such methods rely on one strong underlying assumption: it is necessary to train the document embedding to optimize the prediction of the target words in the document. In other words, the objective requires the model to learn to predict the target words in surface text. We argue that there are several concerns with such a self-prediction assumption.

The strategy of predicting target words therefore only exploits in-document information, and do not explicitly model the inter-document distances. We believe an ideal embedding space should also infer the relations among training documents. For example, if all documents in the corpus are about *machine learning*, then the con-

---

[*] Equally contribution.

Proceedings of NAACL-HLT 2019, pages 2465–2474
Minneapolis, Minnesota, June 2 - June 7, 2019.

cept of *machine learning* becomes less critical in the embedding. However, if the corpus contains documents from different areas of computer science, then the concept of *machine learning* should be encoded in any document relevant to it. We therefore claim that *the embedding of a document should not only depend on the document itself but also the other documents in the corpus*, even though previous work seldom makes this consideration.

In addition, accurate predictions at the lexicon or word level do not necessarily reflect that the "true semantics" have been learned. For example, in IMDB dataset review No.10007:

$$\text{"}...the\,father\,did\,such\,a\,good\,job.\text{"}$$

Obviously, *good* can be replaced with synonyms like *nice* without significantly altering the meaning of the sentence. However, since the synonyms are treated as independent tokens in PV and Doc2VecC, the lexicon *good* must be predicted exactly. Moreover, to accurately predict the final word *job*, the embedding probably only needs to know that *did a good job* is a very common phrase, without having to understand the true meaning of *job*. This example shows that in order to accurately predict a local lexicon, the embedding might opt to encode the syntactic relationship instead of true semantics. Enforcing document embeddings to make predictions at the word level could be too strong of an objective. More specifically, we argue that the true semantics should not only depend on a small context, but also the relations with other training documents at document level.

To address the above concerns we propose a novel model for learning document embedding unsupervisedly. In contrast with previous work (PV and Doc2Vec), we model documents according to two aspects.

First, we abandon the concept of context word prediction when training an embedding model. Instead we propose a self-supervision learning framework to model inter-document information. Conceptually, we use the embedding to determine whether a sentence belongs to a document. Our encoder is equipped with a discriminator to classify whether a sentence embedding is derived from a document given that document's embedding. This explicitly enforces documents to be spread reasonably in the embedding space without any labels so that they can be discriminated. To the best of our knowledge, this is the first deep embedding work to explicitly model the inter-document relationship.

Second, in our approach the predictions are inferred at the sentence level. This avoids the effect of only predicting the surface meaning in word level (e.g. good vs. nice). Unlike previous work, our model is explicitly optimized to represent documents as combinations of sequence embedding beyond words seen in training.

Below we summarize the key contributions:

- We present a deep and general framework and a novel objective for learning document representation unsupervisedly. Our models are end-to-end, easy to implement, and flexible to extend.
- We perform experiments through sentiment analysis and topic classification to show that our model, referred to as *self-discriminative document embedding (SDDE)*, is competitive to the state-of-the-art solutions based on traditional context-prediction objectives.
- Our extensive experiments quantitatively and qualitatively show that SDDE learns more effective features that capture more document-level information. To the best of our knowledge, SDDE is the first deep network to model inter-instance information at document level.
- We further propose to evaluate unsupervised document embedding models in weakly-supervised classification. That is, lots of unlabeled documents with only few labels attached to some of them, which is a realistic scenario that unsupervised embedding could be particularly useful.

2 Related Work

Here we give an overview of other related methods on learning unsupervised text representations. Besides BoW and TF-IDF, Latent Dirichlet Allocation models (Deerwester et al., 1990; Blei et al., 2003) leverage the orthogonality of high-dimensional BoW features by clustering a probabilistic BoW for latent topics.

Several models extend from Word2Vec (Mikolov et al., 2013), using context-word predictions for training document embedding end-to-end. PV (Le and Mikolov, 2014) keeps a document embedding matrix in memory and is jointly trained. The required training parameters are linear to the number of documents and thus prohibit PV from being trained on a large corpus. Moreover, expensive inference for new documents is required during testing. To address the above concerns, Doc2VecC (Chen, 2017) combines PV and a denoising autoencoder (DEA) (Chen et al., 2012) with BoW vectors as global document information instead. The final document embedding are then produced by simply averaging the jointly trained word embedding.

Another thread of work uses two-stage pipelines to construct sentence/document embedding from pre-trained word embedding. Arora et al. (2017) propose post-processing weighting strategies on top of word embedding to build sentence representations. WME (Wu et al., 2018) propose a random feature kernel method based on distance between pairs of words, which also shows inter-document information helps. However, the cost scales with the size of training samples such that it is hard to be applied on large-scale dataset.

There have been more embedding work on sentences compared to documents. These approaches mostly learn the sentence embedding by modeling the sentence-level (Kiros et al., 2015; Tang et al., 2017b,a; Logeswaran and Lee, 2018) or word-level (Pagliardini et al., 2018; Kenter et al., 2016; Arora et al., 2018) distribution hypothesis (Harris, 1954; Polajnar et al., 2015) in a large ordered corpus. We note that the main difference between learning embedding for sentences and documents is that documents are not ordered in a corpus. Some other work model sentences with RNN autoencoders (Hill et al., 2016a; Gan et al., 2017). Documents often refer to long-length text containing multiple sentences, which might be hard to model with RNNs (Pascanu et al., 2013; Jing et al., 2017) and time-consuming on large corpus.

3 Model and Design Rationale

To facilitate downstream tasks, a document embedding is required to compress useful features into a compact representation. It is not an easy task to learn discriminable features unsupervisedly since validation information is not accessible for training. We first introduce some notations:

- $\mathcal{V}$: the training corpus vocabulary of size $|\mathcal{V}|$;
- $\mathcal{X} = \{X_1, \cdots, X_n\}$: a training corpus of document size $n = |\mathcal{X}|$, in which each document X_i is a set of sentences $\mathcal{S}_i$;
- $\mathcal{S}_i = \{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^{|x_i|}\}$: a document divided into a set of sentences, of set size $|\mathcal{S}_i|$, in which each sentence $\mathbf{s}_i^j \in \mathcal{R}^{|\mathcal{V}| \times T_j}$ contains a sequence of variable length T_j of word one-hot vectors $\mathbf{w}_j^1, \cdots, \mathbf{w}_j^{T_j}$, each in $\mathcal{R}^{|\mathcal{V}| \times 1}$. $\mathcal{S}$ is the set of total sentences $\bigcup\limits_{i=1}^{n} \mathcal{S}_i$ in the training corpus, of size $m = |\mathcal{S}|$;
- h_w: the size of the word embedding and $\mathbf{U} \in \mathcal{R}^{h_w \times |\mathcal{V}|}$: the word embedding projection matrix. We use $\mathbf{u}_w$ to denote the column in $\mathbf{U}$ for word $\mathbf{w}$
- h_s: the size of the sentence embedding and $\mathbf{e_s} \in \mathcal{R}^{h_s}$: the embedding of sentence $\mathbf{s}$.
- $\mathbf{d}_i \in \mathcal{R}^{h_s}$: document X_i's embedding.

Our goal is to learn a function $F : \mathcal{X} \rightarrow \mathcal{R}^{h_s \times n}$ that maps document X_i to $\mathbf{d}_i$ unsupervisedly.

Next, we formulate how SDDE represents a document, then introduce our self-discriminative learning procedure we use to train it.

3.1 Document Representation

We consider a document as **mean of sentences**, i.e., breaking a document into several subsequences. We demonstrate several benefits of the design in SDDE. First, decomposing long documents into shorter but reasonable semantic unit (e.g., sentences) makes encoding easier and faster since they can be processed in parallel. Similar concepts of modeling documents hierarchically have shown benefits in some supervised tasks such as text classification (Yang et al., 2016). It also makes the model insensitive to document length, which is important because length varies greatly in real documents (see Table 1).

In training, we further propose to represent a document *during training* using the average of only a subset of its sentences. This special **sentence-level dropout** is beneficial for training by creating up to $\binom{|X_i|}{q}$ combinations for each document, where q is the number of sentences to keep. This enforces the local sentence representations to capture global information of a document by representing it with a subset of sentences in it. The word embedding is used as globally shared building blocks for sentence embedding.

For a document $X_i = \{\mathbf{s}_i^j\}$, or $\mathcal{S}_i$, the embedding is derived from averaging the respective representations of subsequences. Noted as:

$$\mathbf{d}_i = \frac{1}{q} \sum_{\substack{j=1,\\ \mathbf{s}\sim P_{\mathcal{S}_i}(\mathbf{s})}}^{q} \mathbf{e}_{\mathbf{s}}^{j}, \tag{1}$$

where

$$\mathbf{e}_{\mathbf{s}} = E(\mathbf{s}), \tag{2}$$

where a sentence encoder E is introduced to produce sentence embedding for $\mathbf{s}_i^j$. In practice, sentences can be obtained by simply segmenting documents with punctuation. In testing, the document embedding is obtained by averaging all the sentences in which:

$$\mathbf{d} = \frac{1}{|\mathcal{S}_i|} \sum_{\forall \mathbf{s} \in \mathcal{S}_i} \mathbf{e}_{\mathbf{s}}. \tag{3}$$

We note that averaging subsequences differs from averaging of words in two aspects. First, each sentence is encoded individually before being averaged, allowing incorporation of word order into design rationale at least in a reasonable range. Second, subsequences may have different lengths that reveal syntactic information. To illustrate, BoW/mean-of-word models suffer from ambiguously modeling two different documents which are similar in word distributions but differ in some aspects of interest. Mean-of-sentence model avoids such concern by modeling documents at the sentence level. It could be expected that it is much less likely to find two documents with similar sentence distribution than similar word distribution. Mean-of-sentences formulation can be smoothly reduced to mean-of-word models (by treating each word as a sentence) or pure sequence models (by treating each document as a very long sentence).

3.2 Self-Discriminative Learning

Unlike PV or Doc2VecC which emphasize modeling distributional information within individual documents, we model relations across documents. The basic idea is that we hope to learn an embedding for each sentence in the document as well as a discriminator that determines whether a sentence belongs to a document. Self-discriminative learning uses a discriminator network D to determine whether a sentence belongs to a document. The aim is to learn a suitable embedding and a good discriminator to determine if a sentence belongs to a document. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Self-Discriminative Learning for Unsupervised Document Embedding

```
Input: Documents $\mathcal{X} = \{X_i\}_1^n$, $p$, $k$, $h_w$, $h_s$.
Output: Function $F : \mathcal{X} \to \mathcal{R}^{h_s}$ that maps text of a document to an embedding $\mathbf{d} \in \mathcal{R}^{h_s}$.
Compute $\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{S}_i$ from set $\mathcal{S}_i$ of each $X_i$.
Create and initialize $D$, $\mathbf{U}$.
Create and initialize $E$ as in Eq. 5 or Eq. 6.
while not converge do:
  for $i = 1, \dots, n$ do
    Sample $\mathbf{s}$ from $\mathcal{S}_i$.
    Get $\mathbf{e}_{\mathbf{s}}^p \leftarrow E(\mathbf{s})$ as a positive sample.
    Sample $\mathbf{s}_i^1, \dots, \mathbf{s}_i^q$ from $\mathcal{S}_i \setminus \{\mathbf{s}\}$.
    for $j = 1, \dots, q$ do
      Get $\mathbf{e}_{\mathbf{s}}^j \leftarrow E(\mathbf{s}_i^j)$
    end for
    Get $\mathbf{d}_i$ with Eq. 1 given $\{\mathbf{e}_{\mathbf{s}}^j\}_{j=1}^q$.
    for $\ell = 1, \dots, k$ do
      Sample $\mathbf{s}'_\ell$ from $\mathcal{S} \setminus \mathcal{S}_i$.
      Get $\mathbf{e}_{\mathbf{s}'}^\ell \leftarrow E(\mathbf{s}'_\ell)$ as a negative sample.
    end for
    Compute Eq. 4 given $\mathbf{d}_i$, $\mathbf{e}_{\mathbf{s}}^p$, $\{\mathbf{e}_{\mathbf{s}'}^\ell\}_{\ell=1}^k$.
    Backprop and update for $E$, $D$, $\mathbf{U}$.
  end for
end while
Return $F(E, \mathbf{U})$,
```

We propose an objective that explicitly optimizes SDDE towards representing a document with mean of (encoded) sentences. To optimize the discriminator D, we formulate it as a binary classifier that takes pairs of document embedding $\mathbf{d}$ of a document X_i and a sentence embedding $\mathbf{e}_{\mathbf{s}}$, $(\mathbf{d}, \mathbf{e}_{\mathbf{s}})$, as inputs. The discriminator is asked to discriminate using $\mathbf{d}$ whether the sentence $\mathbf{s}$ belongs to the document X_i or the other documents $X' \in \mathcal{X} \setminus \{X_i\}$. The loss then becomes:

$$log(1 - D(\mathbf{d}, \mathbf{e}_{\mathbf{s}}^p)) + \sum_{\ell=1}^{k} \mathop{\mathbb{E}}_{\substack{\mathbf{s}'\sim P_{\mathcal{S}},\\ \mathbf{s}'\notin \mathcal{S}_i}} \left[log(D(\mathbf{d}, \mathbf{e}_{\mathbf{s}'}^{\ell})) \right], \tag{4}$$

for each document with one positive sample $\mathbf{e}_{\mathbf{s}}^p$ and k negative samples of sentences $\mathbf{e}_{\mathbf{s}'}$, where $\mathbf{s}'$ are not in the sentence set $\mathcal{S}_i$ of X_i, as $\mathbf{s}' \notin \mathcal{S}_i$. Note that $\mathbf{e}_{\mathbf{s}}^p$ is not used for $\mathbf{d}$ otherwise it would be trivial to be solved by the discriminator.

The spirits of self-discriminative learning can be understood as unsupervisedly mimicking supervised inference without knowledge of any label information by treating sentences from other documents as fake/negative samples. One main concern that it is possible to find similar sentences in two different documents. Our discriminator particularly addresses this issue by optimizing for the most discriminative sentences rather than similar ones that might not be critical to shape the embedding. To minimize the loss, the encoder would tend to preserve the most essential feature to facilitate the discriminator to push away any two documents, which should encourage the embedding points spread even more widely across the space. This in turn should result in more ease in downstream tasks: for example in learning a decision hyperplane in a classification task.

3.3 Sentence Encoder

Next, we narrow down to sentence encoder E. Given a sequence of word one-hot vectors as a sentence $\mathbf{s} = [\mathbf{w}^1, \ldots, \mathbf{w}^T]$, we project them into an embedding layer $\mathbf{U}$ to retrieve their corresponding word embedding. Note that the word embedding are trained jointly.

Our first method uses:

$$E(\mathbf{s}) = \phi(ReLU(G(\mathbf{U}\mathbf{s}^t))), \tag{5}$$

where G is a single-layer RNN encoder using GRU cells to process the word embedding sequences and ϕ is a linear transform for dimension h_s of sentence embedding.

Our second method, we use a schema of mean-of-word for advantage of fast generation, we average the word embedding $\mathbf{w}$ within a sentence $\mathbf{s}$ along time axis as AVG encoder:

$$E(\mathbf{s}) = \phi(ReLU(\frac{1}{|\mathbf{s}|}\sum_{i=1}^{|\mathbf{s}|}\mathbf{U}\mathbf{w}_i^t)), \tag{6}$$

Let us stress that the role of encoder E is to extract local feature from every sentence, and the overall objective encourages SDDE to represent documents as mean of sentence embedding.

3.4 Discriminator

An undesired pitfall comes from a learned weak encoder with a powerful discriminator causing the embedding produced by the encoder useless for downstream tasks. To avoid such a pitfall, we adopt lightweight network structures for discriminators. For the IMDB datasets, we find inner product $(\mathbf{d}\mathbf{V})^t E(\mathbf{s})$ with a learnable matrix $\mathbf{V}$ sufficient. For the other datasets in Table 1, two fully-connected layers with ReLU activations in latent are used.

Dataset	#Class	#Train / #Test	Doc Length	Sent Length
IMDB	2	75k / 25k	124.6±8,856.7	11.6±105.5
Yelp P.	2	560k / 38k	70.0±4,117.8	8.2±48.7
AG's News	4	120k / 7.6k	27.2±66.1	11.8±66.7
DBPedia	14	560k / 70k	32.8±231.3	9.7±68.6

Table 1: Statistics of the datasets. Length of document and sentence in words (mean±variance), which could be high for real-world scenarios such as online reviews.

Dataset	k	p	h_w	h_s
IMDB	1	3	100	100
Yelp P.	1	3	300	500
AG's News	3	1	100	100
DBPedia	4	2	300	500

Table 2: Hyperparameters used in experiments. The document embedding trained with the same hyperparameters are used for all the evaluations without task-specific tuning.

3.5 Sampling Sentences

We relate our method to the Negative Sampling (Mikolov et al., 2013) technique which is a simplified objective of softmax approximation (Mikolov et al., 2013; Mnih and Teh, 2012; Zoph et al., 2016). Negative sampling has been used as an efficient and effective technique in learning word embedding. We reformulate it to train document embedding by sampling in sentence level, which is easy to implement and efficient to train just like Word2Vec (Mikolov et al., 2013).

In practice, when training with mini-batches the documents for negative samples are from the same mini-batch, which requires small extra computation efforts.

SDDE requires a similar number of parameters as Doc2VecC does, but much less than PV. In addition, the sentence encoder is flexible and can incorporate other techniques of text processing such as attention methods.

4 Experiments

4.1 Setup

Public datasets on sentiment analysis and topic classification across diverse domains including online reviews, news, and Wiki pages are used including IMDB dataset (Maas et al., 2011) and the others from Zhang et al. (2015). Table 1 provides a summary. Only the training splits are used in training embedding with subsampled training split for cross-validations.

We preprocess the datasets by normalizing the text to lower class and replacing words appearing less than 10 times with a *UNK* token. Out-of-vocabulary words in testing set are also replaced. All our baseline models use the same input data. To define the sentences for experiment, we utilize the sentence tokenizer from NLTK. For the documents containing only one sentence we simply divide it into multiple subsequences for sampling. We use RMSProp method for optimization. Dropout 50% of input to the discriminator. Weights are random-uniformly initialized between [-1, 1].

The other hyperparameters are summarized in Table 2. All the models use the same embedding size for fair comparison. The trained document embedding are used for all the evaluations without specific tuning.

4.2 Evaluation

Generally, it is not easy to evaluate an unsupervised embedding model. In Section 4.3 and 4.4, we evaluate the performance on standard document classification following the common practice used by previous work (Chen, 2017; Wu et al., 2018; Arora et al., 2017): a classification task with a linear SVM (Fan et al., 2008) trained on the labels in each dataset. Next, we study unsupervised document embedding on two novel aspects. In Section 4.5, we study a weakly-supervised classification setting that fits the realistic scenario of using unsupervised embedding with only a few labels. In Section 4.6, we provide a metric to evaluate the effectiveness of modeling inter-document information.

4.3 Sentiment Analysis with IMDB Dataset

We first compare our models with the others state-of-the-art competitors. RNN-LM (Mikolov et al., 2010) and Skip-thought (Kiros et al., 2015) are RNN-based. SIF (Arora et al., 2017), W2V-AVG (Mikolov et al., 2013), and WME (Wu et al., 2018) are two-stage approach that post-processing on word embedding. We collect the results reported on the widely-used benchmark sentiment classification dataset IMDB. For PV, we use Gensim implementation (Řehůřek and Sojka, 2010); versions of both Distributed Memory (DM) and Distributed Bag of Words (DBoW) are reported. For different encoders in SDDE, AVG is for averaging word embedding and RNN is for the RNN encoder.

Model	Error%
Skip-thought* (Kiros et al., 2015)	17.4
SIF (GloVe) (Arora et al., 2017)	15.0
RNN-LM* (Mikolov et al., 2010)	13.6
W2V-AVG* (Mikolov et al., 2013)	12.7
DEA* (Chen et al., 2012)	12.5
PV-DM	20.0
PV-DBoW	12.0
WME (Wu et al., 2018)	11.5
Doc2VecC* (Chen, 2017)	11.7
SDDE-AVG	**10.6**
SDDE-RNN	**10.2**

Table 3: Sentiment Classification on IMDB Benchmark. *Results are collected from Chen (2017).

Pred	SDDE-RNN		Doc2VecC	
	True	False	True	False
#Sent	14.8	14.7	14.5	14.8

Table 4: Mean of #sentence per document in IMDB dataset, in groups of classification correctness.

Self-Discriminative Learning is Effective From the experiment result in Table 3, we can see that our self-discriminative learning is effective and superior on the document embedding models for both AVG and RNN versions. SDDE-RNN achieves best accuracy on IMDB dataset 1.5% margin against Doc2VecC.

Study the Property of SDDE Unlike previous work modeling documents on the word or short context level, SDDE operates on the sentence level. We study the false and true predictions output by SVM upon SDDE in comparison with Doc2VecC. Table 5 show some examples that have the largest difference. We observed SDDE can better capture contradicting or contrasting opinions. We observe some wrong predictions (Row 3) are due to the ambivalent reviews. SDDE is insensitive to the number of sentences; we found the effect of the number of sentences per document was trivial as shown in Table 4.

4.4 Large-Scale Document Classification on More Dataset

Next, we borrow some public large-scale dataset in Table 1 to further validate the effectiveness of SDDE compared to the other models. For Doc2vecC and SIF, we use the code from the au-

---

https://www.nltk.org/

Label: 1 SDDE: 0.89 Doc2VecC: 0.27	i don t even like watching those late night talk shows , but i found this one really interesting . i imagine it s probably close to the truth — it feels like an honest account , if that means anything . kinda feel for the people somewhat when you watch it . a nice movie for a saturday night .
Label: 0 SDDE: 0.12 Doc2VecC: 0.75	i m a boorman fan but this is arguably his least successful film . comedy has never been his strong suit , and here his attempts at screwball farce are clumsily done . still , it s almost worth seeing for boorman s eye for talent : this is one of uma thurman s first starring roles , and as always she is ravishing to watch . on a sad side note boorman wrote the script with his daughter , <UNK> who died a couple years ago .
Label: 0 SDDE: 0.77 Doc2VecC: 0.10	michael dudikoff stars as joe armstrong a martial artist who fights ninjas who are stealing weapons from the u s army , in this entertaining yet admittedly brainless martial arts actioner , which is hampered by too many long pauses without action , but helped by some high energy action <UNK> as well as steve james performance .

Table 5: IMDB Examples with scores 0 to 1 for negative to positive assigned by SVM.

Model	Yelp P.	AG	DBP.
PV-DM	17.6	39.6	21.6
PV-DBoW	12.1	16.8	10.4
Word2Vec AVG	8.0	14.2	2.7
SIF	8.4	13.8	2.9
Doc2VecC	7.4	10.4	2.0
SDDE-RNN	8.7	12.7	8.6
SDDE-AVG	**6.7**	**9.8**	**1.8**

Table 6: Testing error rate (%) with standard classification on public datasets. Bold text indicates passing hypothesis test with p-value < 0.05 with different random initialization.

Model	IMDB	Yelp P.	AG	DBP.
Doc2VecC	12.5	11.5	13.9	3.6
PV-DBoW	13.7	18.6	13.9	13.1
SDDE	**11.7**	**10.0**	**10.0**	**2.9**

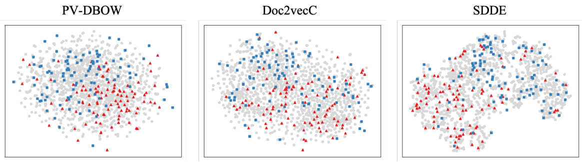
Table 7: Testing error (%) in weakly-supervised setting. Only 1k labeled data per class were used. 4.5).

thors. We use SIF to generate document embedding with Word2Vec trained on each dataset as its inputs.

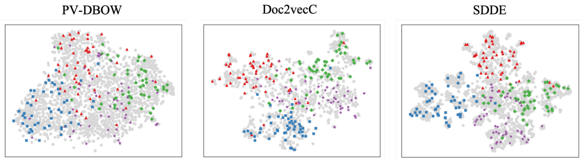
Results are shown in Table 6. SDDE-AVG performs slightly better across different dataset. We hypothesis SDDE gets larger improvement on IMDB dataset since SDDE can handle longer documents better by exploiting sentence embeddings. On the other hand, the RNN version of SDDE performs significantly worse than the word-averaging version. We may remind the reader that state-of-the-art unsupervised document embedding models are not RNN-based. The effects of word order are still unclear. Wieting and Gimpel (2017) provides a study of sentence embedding. We hypothesize that it may be difficult for an RNN encoder to learn to incorporate multi-domain information in datasets with many classes (e.g., DBpedia) unsupervisedly. This would be our future work.

4.5 Classification with Few Labeled Data

Next, we consider a more real-world weakly-supervised learning scenario: classification on the datasets we have used in previous experiments, but this time only when very few labels are available.



(a) IMDB



(b) AG's news

Figure 1: t-SNE embedding Visualizations of weakly-supervised learning experiments in Section 4.5. Colored points are labeled training data in the experiments and gray points are the unlabeled testing documents.

We hypothesize that SDDE is particularly useful for classification with few labels since the self-discriminative learning has exploited the possible features to map the text onto the embedding space properly during the representation learning phase. The embedding is expected to be more discriminable to facilitate finding the classification decision hyperplanes with fewer labeled data.

We randomly sample equal number of instances from each class to train a SVM and verify with the whole testing set. PV-DBoW, Doc2VecC, and SDDE-AVG are examined in this experiment. We use the same pre-trained document embedding as in the previous experiments. We repeat the whole procedure 30 times and report the means and tune the penalty parameter C to find the best value for each model. Results in Figure 2 and Table 7 show SDDEs outperform PV and Doc2VecC. We visualize the training points with t-SNE. As shown in Figure 1, SDDE seems to be able to spread the embedded data more widely, which eventually leads to better usage of scarce data for classification.

`https://github.com/mchen24/iclr2017`
`https://github.com/PrincetonML/SIF`

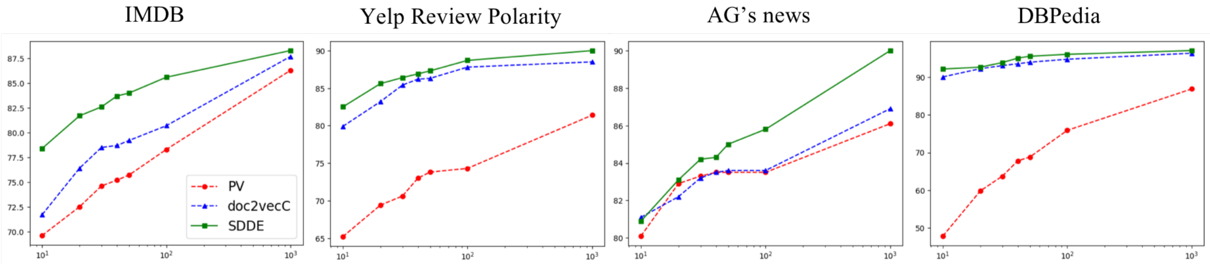


Figure 2: Weakly-supervised learning on datasets in Table 1. Training with [10, 20, 30, 40, 50, 100, 1000] instances per class (X-axis) and computing the accuracy on the whole testing set (Y-axis).

Model	IMDB			Yelp P.			AG			DBP.		
Metric	A	E	**A-E**	A	E	**A-E**	A	E	**A-E**	A	E	**A-E**
Doc2VecC	.78	.76	.02	.52	.49	.03	.57	.46	.11	.60	.40	.20
Word2Vec AVG	.86	.85	.01	.54	.51	.03	.57	.44	.13	.75	.61	.14
PV-DBoW	.28	.27	.01	.11	.10	.01	.71	.60	.09	.45	.36	.09
SDDE	.36	.27	**.09**	.14	.08	**.06**	.20	.01	**.19**	.58	-.03	**.61**

Table 8: Distances of Intra & Inter-class cosine similarity. A for IntraCos and E for InterCos, note that they cannot be compared across different models. Instead, distance A-E defined in Equation 7 is reported to study a method's effectiveness of modeling inter-document features. The higher the number the better.

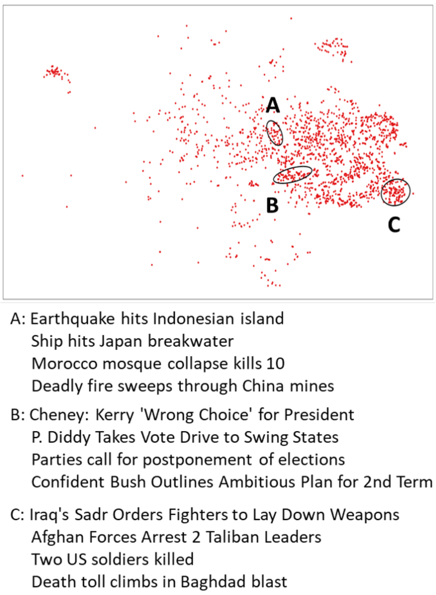
4.6 Distance of Intra&Inter-Class Pairwise Cosine Similarity

Definition We examine our assumption of the ability of SDDE to model inter-document feature. Similar to (Hill et al., 2016b), we consider pairwise cosine similarity between documents with topic labels, this allows us to quantitatively evaluate unsupervised document embedding at inter-document level. Our assumption is that: if pairwise similarity between documents is calculated based on different kinds of embedding, the better embedding results should comply with the properties of both high similarities between those documents within the same underlying class, denoted as $IntraCos(d, d')$ and low similarities between document pairs from different classes, or $InterCos(d, \tilde{d})$. The mean distance:

$$mean(IntraCos) - mean(InterCos), \quad (7)$$

is considered as our metric to avoid simply maximizing IntraCos or minimizing InterCos.

SDDE Provides High Separation Table 8 shows the evaluation. The distances (Eq. 7) for the baseline models are small, which support our assumption that these methods are not able to model inter-document features properly. On the other hand, distances for SDDE are significantly larger. With the classification experiments, we believe SDDE better preserves meaningful inter-document features. Figure 3 shows some meaningful clusters in SDDE in the World class as cohesive sub-classes.



A: Earthquake hits Indonesian island
Ship hits Japan breakwater
Morocco mosque collapse kills 10
Deadly fire sweeps through China mines

B: Cheney: Kerry 'Wrong Choice' for President
P. Diddy Takes Vote Drive to Swing States
Parties call for postponement of elections
Confident Bush Outlines Ambitious Plan for 2nd Term

C: Iraq's Sadr Orders Fighters to Lay Down Weapons
Afghan Forces Arrest 2 Taliban Leaders
Two US soldiers killed
Death toll climbs in Baghdad blast

Figure 3: SDDE-AVG t-SNE visualization of class "World" in AG News dataset testing set. Titles of documents are shown. Meaningful clusters such as (A) disaster, (B) election, and (C) war.

5 Conclusion

Compared to mainstream unsupervised document embedding models (trained to perform predictions on the lexicon level) SDDE embeddings capture information at the inter-document level, as they are trained to maximize the distance between a sentence and a corresponding document. We hope the underlying idea of SDDE offers the document-embedding community a new investigation direction. Self-discriminative learning shows potential for real-world scarcely-labeled scenarios, and our future work will focus on joint training of representations for semi-supervised learning.

Acknowledgements

This material is based upon work supported by Microsoft Research Asia (MSRA) grant, and by Taiwan Ministry of Science and Technology (MOST) under grant number 108-2634-F-002 -019.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Sanjeev Arora, Yingyu Liang, Tengyu Ma, Mikhail Khodak, Nikunj Saunshi, and Brandon Stewart. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 12–22.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *ICLR*.

Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*, pages 3079–3087.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *JMLR*, pages 1871–1874.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *EMNLP*.

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016a. Learning distributed representations of sentences from unlabelled data. In *NAACL HLT*, pages 1367–1377.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016b. Learning to understand phrases by embedding the dictionary. *TACL*, 4:17–30.

Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljacic, and Yoshua Bengio. 2017. Gated orthogonal recurrent units: On learning to forget. *CoRR*, abs/1706.02761.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: optimizing word embeddings for sentence representations. In *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In *ICLR*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* *NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 528–540.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Hieu Pham, Minh-Thang Luong, and Christopher D. Manning. 2015. Learning distributed representations for multilingual text sequences. In *NAACL-HLT*, pages 88–94.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2015. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2017a. Rethinking skip-thought: A neighborhood based approach. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 211–218.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2017b. Trimming and improving skip-thought vectors. *CoRR*, abs/1706.03148.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 90–94.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2078–2088.

Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018. Word mover's embedding: From word2vec to document embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4524–4534.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL HLT*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *NAACL HLT*, pages 1217–1222.

Adaptive Convolution for Text Classification

Byung-Ju Choi **Jun-Hyung Park** **SangKeun Lee**
Department of Computer Science and Engineering
Korea University, Korea
`{bj1123, irish07, yalphy}@korea.ac.kr`

Abstract

In this paper, we present an adaptive convolution for text classification to give stronger flexibility to convolutional neural networks (CNNs). Unlike traditional convolutions that use the same set of filters regardless of different inputs, the adaptive convolution employs adaptively generated convolutional filters that are conditioned on inputs. We achieve this by attaching filter-generating networks, which are carefully designed to generate input-specific filters, to convolution blocks in existing CNNs. We show the efficacy of our approach in existing CNNs based on our performance evaluation. Our evaluation indicates that adaptive convolutions improve all the baselines, without any exception, as much as up to 2.6 percentage point in seven benchmark text classification datasets.

1 Introduction

Text classification assigns topics to texts by understanding the semantics of the texts. It is one of the fundamental tasks in natural language processing (NLP) which has a broad range of applications, including web search (Broder et al., 2007), contextual advertising (Lee et al., 2013), and user profiling (Kazai et al., 2016). Traditional approaches to text classification use sparse representations of text, such as bag-of-words (Lodhi et al., 2002). To date, neural network-based text embedding techniques, particularly convolutional neural networks (CNNs) (Kim, 2014; Zhang et al., 2015; Wang et al., 2018b) have shown remarkable results in text classification.

One of the driving forces of CNNs is a convolution operation. It screens local information which appear in inputs (either input texts or outputs from the previous convolution block) by convolving a set of filters with inputs. In the context of text classification, this operation is analogous to questions and answers. Convolutional filters are like questions asking for the intensity of particular patterns in receptive fields. Outputs of convolution operations are the answers from the inputs to the questions. CNNs derive the right class with stacked convolution blocks[1]. On this point, CNNs can be likened to players of the *twenty questions* who guess the answers by iteratively asking questions and receiving information.

However, differences exist between humans and traditional CNNs in the manner in which they play this game. Humans adaptively ask questions by fully utilizing information they have obtained. If players have narrowed the answer down to the name of a person, they would not want to ask questions such as, “Does that have four legs?”. Rather, they would prefer questions related to the target’s profession or origin that are practical for inferring the answer. In contrast, typical CNNs use the same set of filters in any circumstances (Kim, 2014; Johnson and Zhang, 2017; Wang et al., 2018b). This may hamper CNNs from leveraging the information they have as intermediate hidden representations of inputs processed in consecutive convolution operations, and focusing their capacity on disentangling uncertainty.

Motivated by this, we propose an adaptive convolution to give stronger flexibility to networks and allow networks to simulate human capabilities of utilizing the information they have. The adaptive convolution performs convolution operations with filters (questions) dynamically generated conditioned on inputs (outputs from the previous convolution block). We achieve this by attaching filter-generating networks, carefully

---

[1] Pooling can be interleaved with convolution. However, this does not affect the nature of a convolution block.

Proceedings of NAACL-HLT 2019, pages 2475–2485
Minneapolis, Minnesota, June 2 - June 7, 2019.

designed modular networks for generating filters, to convolutional blocks in CNNs. Each attached filter-generating network produces filters from the input and pass the filters to its convolution block. Generated filters are reflections of the information contained in the inputs and allow the networks to focus on extracting informative features. We further propose a hashing technique to substantially compress the size of the filter-generating networks, and prevent a considerable increase in the number of parameters when applying the adaptive convolution.

Our adaptive convolution can easily be applied to existing CNNs, because of the modularity of the filter-generating networks. We demonstrate that significant gains can be realized by applying adaptive convolutions to baseline CNNs (Kim, 2014; Johnson and Zhang, 2017; Wang et al., 2018b), based on a performance evaluation. Our adaptive convolutions improve performance of all the baseline CNNs as much as up to 2.6 percentage point, without any exception, in seven text classification benchmark datasets.

To summarize, our technical contributions are three fold:

- We propose an adaptive convolution which can give stronger flexibility to existing CNNs.
- We design a hashing technique to apply the adaptive convolution without a considerable increase in the number of required parameters.
- We show the effectiveness of our approach based on an evaluation on seven text classification benchmark datasets.

The remainder of this paper is organized as follows. Section 2 discusses related works and Section 3 describes the proposed methodology. We present our evaluation in Section 4 and conclude the paper in Section 5.

2 Related Works

2.1 CNN-based Text Classification

Ever since single layer CNNs were successfully applied to text classification with pre-trained word embeddings (Kim, 2014), many researches have sought effective utilization of CNNs in text classification. Kalchbrenner et al. (2014) introduced a dynamic k-max pooling to handle variable length sequences. Zhang et al. (2015) classified texts wholly based on the characters in the texts. Lai et al. (2015) and Xiao et al. (2016) incorporated recurrent neural networks (RNNs) into CNNs. Conneau et al. (2017) and Johnson et al. (2017) investigated deepening CNNs. Wang et al. (2018b) used dense connections to reuse features from upstream layers at downstream layers. Interests of these researches were concentrated to network architectures, pooling operations or input structures, accepting the nature of the convolution operation. Our work is different from them in that we focus on the convolution operation.

2.2 Parameter Generation

Generating parameters in neural networks have been examined in various researches. Noh et al. (2016) used embedded questions to adaptively create parameters for a fully connected layer in visual question answering. Bertinetto et al. (2016) predicted parameters of a predictor network from an exemplar in a one-shot learning framework. Van den Oord et al. (2016) generated feature-wise biases from descriptive labels or tags that were directly added to layer outputs for conditional image generation. Liu et al. (2017) introduced a specifically designed meta network to produce weights for compositional matrices in tree structured neural networks.

Several researches have adopted parameter generation for conditional normalization (CN). In these studies, parameters in the normalization layer were substituted with learned functions of conditioning information, the outputs from which were then used as normalizing parameters. Different types of CN include conditional instance normalization (Dumoulin et al., 2017) for style transfer, dynamic layer normalization (Kim et al., 2017) for speech recognition and conditional batch normalization (De Vries et al., 2017) for visual question answering. Perez et al. (Perez et al., 2018) relaxed CN by modulating inputs with affine transformation without normalization.

Studies that are most similar to our own involve generating convolutional filters in the field of computer vision. Klein et al. (2015) proposed a dynamic convolution layer for weather prediction. They generated filters for subsequent frames with the previous image. De Brabandere et al. (2016) expanded the dynamic convolution layer by allowing position-specific filters. Niklaus et al (2017) estimated convolutional filters with images in receptive patches to interpolate their corresponding output pixels. Kang et al. (2017) produced convolutional filters with side information such as camera perspective for crowd counting. These approaches can be generalized by the hypernetworks (Ha et al., 2017) in which all weights for the main networks are generated with original inputs. Although this idea can be directly applied for text processing (Shen et al., 2018a), our approach is different from theirs in that we generate filters with outputs from previous convolution blocks, to fully utilize intermediate information obtained from networks as they process inputs.

3 Proposed Methodology

This section introduces our adaptive convolution. Instead of convolving the same set of filters with inputs, an adaptive convolution uses dynamically generated filters conditioned on the outputs from the previous convolution block. In Section 3.1, we explain how the filters are generated in the filter-generating network. In Section 3.2, we show how the adaptive convolution operates with the generated filters, and how it can be applied to existing CNNs.

3.1 Filter-Generating Network

Figure 1 schematically shows the overall architecture of our filter-generating network. The filter-generating network takes an input to a convolution block $I = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$. m is the length which can be the number of words in each text or reduced number of it as a result of a pooling operation. Entry $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional vector in the i^{th} position in the input. d is equal to the number of filters of the previous convolution block, except for the first convolution block which uses word embedding dimension. It outputs a set of k convolutional filters $\mathcal{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_k]$. Each filter $\boldsymbol{f}_i \in \mathbb{R}^{h \cdot d}$ consists of a filter size h by d weights.

The filter-generating network generates filters in two phases: context vector generation and filter generation. During context vector generation, the variable size input I is encapsulated into a fixed size g-dimensional context vector $\boldsymbol{c} \in \mathbb{R}^g$. During filter generation, the filter-generating network adaptively produces filters from the context vector.

Context Vector Generation We generate a context vector by leveraging the sequential information inherent in inputs. An input to each convolution block is an intermediate hidden representation of a text. Clearly, text has a sequential nature. This property is preserved when text is processed in typical CNNs because they do not shuffle position information between convolution blocks. Operations in CNNs produce an entry for each position from the corresponding position of the filters. As words within a text are dependent on each other, entries within an input are related.

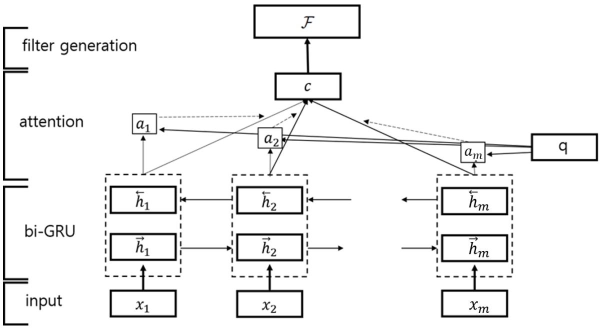


Figure 1: Graphical illustration of filter-generating network. It generates context vector by self-attending hidden states of bidirectional GRU. Convolutional filters are generated from context vector.

To gain some dependencies between entries in a sequence, we use the Gated Recurrent Unit (GRU) (Cho et al., 2014). We found from our preliminary experiments that GRU shows comparable performance to the Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) with fewer parameters and lower computational cost. We obtain a hidden state $\boldsymbol{h}_t \in \mathbb{R}^g$ for entry $\mathbf{x}_t$ by concatenating two hidden states of bidirectional GRU at time step t :

$$\begin{aligned} \overrightarrow{\boldsymbol{h}_t} &= \overrightarrow{GRU}(\mathbf{x}_t, \overrightarrow{\boldsymbol{h}_{t-1}}), \\ \overleftarrow{\boldsymbol{h}_t} &= \overleftarrow{GRU}(\mathbf{x}_t, \overleftarrow{\boldsymbol{h}_{t+1}}), \\ \boldsymbol{h}_t &= [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}] \end{aligned} \tag{1}$$

We then have m number of hidden states $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_m]$. We encapsulate H into a fixed size context vector $\boldsymbol{c}$ by the weighted sum of hidden states :

$$\boldsymbol{c} = \sum_{j=1}^{m} a_j \boldsymbol{h}_j \tag{2}$$

where a_j is a scalar weight of each hidden state h_j

calculated as follows:

$$a_j = \frac{\exp(\boldsymbol{q}^\top \boldsymbol{h}_j)}{\sum_{k=1}^{m} \exp(\boldsymbol{q}^\top \boldsymbol{h}_k)} \tag{3}$$

where $\boldsymbol{q}$ is a trainable query vector. This is a special case of self-attention (Lin et al., 2017) with a hop size of one and without a hidden state projection. The context vector $\boldsymbol{c}$ is an effective summarization of the input, and convolutional filters can be readily generated from the fixed size context vector.

Filter Generation : Once we obtain the context vector $\boldsymbol{c}$ by attending hidden states of the bidirectional GRU, we generate convolutional filters $\mathcal{F}$ by a function of $\boldsymbol{c}$.

$$\mathcal{F} = f(\boldsymbol{c}) \tag{4}$$

Although any kinds of function can be applied, we are interested in adding filter-generating networks to existing CNNs. In order for existing CNNs to be trained in an end-to-end fashion even after filter-generating networks are added to them, a differentiable architecture is preferred so that gradients can be backpropagated. We use a fully-connected layer for its simplicity. With this layer, convolutional filters can be generated in two different approaches: *full generation* and *hashed generation*. For clarity, we explain how each filter $\boldsymbol{f}_i$ is generated. All filters in $\mathcal{F}$ are produced in the same manner.

Full Generation: The most straightforward way to generate a convolutional filter is to use an output of a fully-connected layer directly as a convolutional filter. The layer takes the context vector $\boldsymbol{c}$ and yields filter $\boldsymbol{f}_i$ as follows:

$$\boldsymbol{f}_i = \boldsymbol{W}_i \boldsymbol{c} \tag{5}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{(h \cdot d) \times g}$ is the weight matrix for generating i^{th} filters.

Hashed Generation: Full generation requires numerous parameters because the size of the matrix $\boldsymbol{W}_i$ increases quadratically between the size of the context vector and the convolutional filter. This may cause a memory issue in very deep CNNs. To address this issue, we employ a hashing trick. The hashing trick allows the filter to be generated with only a fraction of the required number of parameters for full generation.

Our hashing trick is motivated by the recently proposed hash embeddings (Svenstrup et al., 2017), which constructs word embeddings with the weighted sum of n *component vectors* from a shared pool. Based on this idea, we generate each $\boldsymbol{f}_i$ by a linear combination of *component filters* from a shared pool. The shared pool $\boldsymbol{E} \in \mathbb{R}^{b \times (h \cdot d)}$ contains b component filters. We select n component filters for $\boldsymbol{f}_i$ from the shared pool using predefined hash functions. The filter $\boldsymbol{f}_i$ is generated by the linear combination as follows:

$$\boldsymbol{f}_i = \sum_{j=1}^{n} p_{i,j} \mathcal{H}_j(\hat{\boldsymbol{f}}_i) \tag{6}$$

where $p_{i,j}$ is the *importance parameter* which determines the weight for the linear combination, and $\mathcal{H}_j$ is a function that outputs a component filter from an ID of the filter, which is denoted as $\hat{\boldsymbol{f}}_i$. $\mathcal{H}_j$ is implemented by $\mathcal{H}_j = \boldsymbol{E}_{D_j(\hat{\boldsymbol{f}}_i)}$, where D_j is a hash function. More specifically, D_j takes $\hat{\boldsymbol{f}}_i$, and outputs a bucket index in $\{1, \cdots, b\}$. The component filter is extracted by taking a row of the bucket index in $\boldsymbol{E}$. To obtain input-specific filters, we control p_i^k as follows:

$$p_{i,j} = \boldsymbol{w_{i,j}}^\top \boldsymbol{c} \tag{7}$$

where $\boldsymbol{w}_{i,j} \in \mathbb{R}^g$ is a vector for generating $p_{i,j}$ from the context vector.

Because we require n numbers of $p_{i,j}$, $n \cdot g$ parameters are needed to generate each filter. The number of the importance parameters n can be chosen to be quite small (we typically use five), and it can provide a huge reduction in the number of parameters compared to the full generation, which uses $h \cdot d \cdot g$ parameters. The additional parameters for the hashed generation come from the shared pool E. Yet, their portion is relatively small, because E is shared across the filters and the size of the shared pool b can be moderate (we typically use 20).

3.2 Adaptive Convolution

We achieve the adaptive convolution by adding a filter-generating network to each convolution block. The filter-generating network yields filters from its input (output from the previous convolution block). The adaptive convolution involves the input-specifically generated filters, which are applied to inputs to produce new outputs. More formally, a feature $o_{i,j}$ corresponding to the filter $\boldsymbol{f}_i$ and j^{th} position

Dataset	AG	DBPedia	Yelp.p	Yelp.f	Yahoo	Amazon.p	Amazon.f
# of training data	120k	560k	560k	650k	1400k	3600k	3000k
# of test data	7.6k	70k	38k	50k	60k	400k	650k
# of classes	4	14	2	5	10	2	5
# of average words	44	54	155	157	108	90	92
# of vocabulary	27k	129k	63k	68k	161k	223k	202k

Table 1: Statistics of the datasets

Algorithm 1 Generalized forward propagation of CNNs applied with the adative convolution.

```
Inputs : A text as a sequence of words
Outputs : Label predictions
$\boldsymbol{I} \leftarrow$ embeds a sequence of words
for each convolution block in CNNs do
    $\boldsymbol{c} \leftarrow$ context generation($\boldsymbol{I}$)
    $\mathcal{F} \leftarrow$ filter generation($\boldsymbol{c}$)
    $\boldsymbol{O} \leftarrow$ convolution($\boldsymbol{I}$,$\mathcal{F}$)
        // apply convolution to $\boldsymbol{I}$ with $\mathcal{F}$
    $\boldsymbol{O} \leftarrow$ pooling($\boldsymbol{O}$)
    $\boldsymbol{I} \leftarrow \boldsymbol{O}$
end for
label predictions $\leftarrow$ softmax(I)
return label predictions
```

of window in inputs is computed as follows:

$$o_{i,j} = \phi(\boldsymbol{f}_i^\top \mathbf{x}_{j:j+h-1}) \tag{8}$$

where $\mathbf{x}_{j:j+h-1}$ denotes the concatenation of inputs $[\mathbf{x}_j, \mathbf{x}_{j+1}, \cdots, \mathbf{x}_{j+h-1}]$, and ϕ is an activation function such as ReLU (Nair and Hinton, 2010). A position feature $\boldsymbol{o}_j$ is produced by concatenating features from all filters $\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_k$, which are applied to the j^{th} position of a window. The output of an adaptive convolution is a sequence of position features $\boldsymbol{O} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_{m-h+1}]$. This output becomes the input to the next convolution block, after operations predefined in the network structure (such as a pooling) are applied. This procedure is repeated for all convolution blocks in the network. Algorithm 1 details our approach. Models adopted with our adaptive convolution can be trained in a typical backpropagation, as all components in adaptive convolution are differentiable.

4 Experiments

4.1 Experimental Settings

Datasets and Data Preprocessing We employ seven datasets covering seven different classification tasks compiled by Zhang et al. (2015). Statistics are summarized in Table 1. 'AG','DBPedia' and 'Yahoo' are news, ontology, and topic classification datasets, respectively. The others are sentiment classification datasets, where '.p'(polarity) in the dataset name indicates that labels are binary and '.f' (full) means that the labels refer to the number of stars.

We tokenize each text using Stanford's CoreNLP (Manning et al., 2014) after converting all uppercase letters to lowercase letters. In building a vocabulary, we retain words that appear more than five times in a corpus. We replace remaining words with the special 'UNK' tokens.

Baselines We select three baseline CNNs to which we apply our adaptive convolution. First one is *CNN* (Kim, 2014), the basic form of CNNs consists of a single convolution block. The others are recently proposed *DPCNN* (Johnson and Zhang, 2017) and *DenseCNN* (Wang et al., 2018b) which employ multiple convolution blocks. We reproduce these three models and apply adaptive convolutions to assess the efficacy of our methodology. All of them are word-level CNNs. We do not apply adaptive convolutions to character-level CNNs (Zhang et al., 2015; Conneau et al., 2017) because of their relatively poor performance compared to word-level CNNs (Johnson and Zhang, 2016). We also compare the performance of our methodology with *ACNN* (Shen et al., 2018a). Similar to our approach, *ACNN* employs dynamically generated filters for convolutions. Different from our approach, however, it generates filters with original inputs from a single subnetwork. Note that *ACNN* is a specifically designed network architecture, so its filter generation approach can not readily be applied to other existing CNNs. Models other than CNNs, such as RNNs (Yang et al., 2016; Lin et al., 2017; Wang et al., 2017) and word embedding-based models (Joulin et al., 2017; Shen et al., 2018b; Wang et al., 2018a) are also included as baseline models. We do not include

Models	AG	DBPedia	Yelp.p	Yelp.f	Yahoo	Amazon.p	Amazon.f
CharCNN (Zhang et al., 2015) *	91.45	98.45	95.12	62.05	71.20	95.07	59.57
VDCNN (Conneau et al., 2017) *	91.3	98.7	95.7	64.7	73.4	95.7	63.0
FastText (Joulin et al., 2017) *	92.5	98.6	95.7	63.9	72.3	94.6	60.2
WSEM (Shen et al., 2018b) *	92.66	98.57	95.81	63.79	73.53		
LEAM (Wang et al., 2018a) *	92.45	99.02	95.31	64.09	**77.42**		
HAN (Yang et al., 2016)	93.28	98.99	97.08	67.92	75.88	95.94	63.54
					(75.8)		(63.6)
KnnLSTM (Wang et al., 2017) *	94.2	99.1	94.5	61.9	74.4	95.3	60.3
Self-Attention (Lin et al., 2017) †	91.5	98.3	94.9	63.4			59.8
ACNN (Shen et al., 2018a)	93.82	99.01	96.51	65.98	74.95	95.54	62.03
		(98.93)	(96.21)				
CNN (Kim, 2014)	93.15	98.92	96.33	65.52	74.2	95.24	61.09
	(93.05)	(98.88)	(96.54)	(65.79)	(73.94)	(95.73)	(62.49)
(Ours) *AC CNN* - full generation	94.07	99.17	**97.18**	**68.12**	76.15	96.23	63.60
(Ours) *AC CNN* - hashed generation	93.81	99.13	97.16	68.07	76.01	96.25	63.74
DPCNN (Johnson and Zhang, 2017)	92.87	98.98	96.77	67.01	75.33	96.07	63.30
(Ours) *AC DPCNN* - full generation	94.03	99.13	97.05	67.91	76.36	**96.31**	**63.94**
(Ours) *AC DPCNN* - hashed generation	93.70	99.10	97.12	67.98	76.07	96.20	63.72
DenseCNN (Wang et al., 2018b)	93.30	99.00	96.48	66.02	74.91	95.95	62.71
	(93.6)	**(99.2)**	(96.5)	(66.0)			(63.0)
(Ours) *AC DenseCNN* - full generation	**94.35**	99.12	97.01	67.63	76.43	96.30	63.91
(Ours) *AC DenseCNN* - hashed generation	93.63	99.09	96.96	67.74	75.95	96.14	63.59

Table 2: Test accuracies [%] on the seven text classification datasets. Results marked with * are reported in each reference, while results marked with † are re-printed following (Wang et al., 2018b). If not specified, results are from our implementations. Values in the parentheses are from their reference, except for *CNN* whose performances is reported in (Johnson and Zhang, 2016).

	CNN	*DPCNN* (depth=11)	*DenseCNN* (depth=6)
Baseline	0.4M	3.1M	2.2M
Hashed generation	2.1M	7.1M	11.6M
Full generation	217.5M	220M	263.2M

Table 3: Number of parameters in each model. Parameter counts do not include word embeddings.

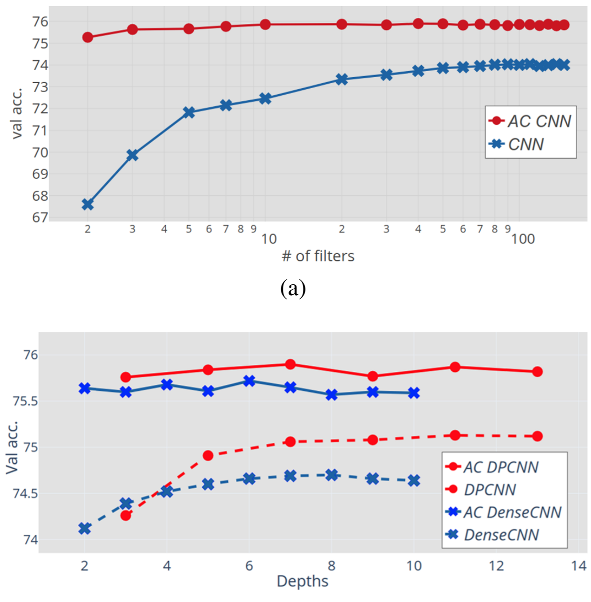
the models where transfer learning is applied, such as ULMFiT (Howard and Ruder, 2018) to compare the capacity of models by themselves, not the effectiveness of transfer.

Training Details We implement all of the models with PyTorch (Paszke et al., 2017) framework. For all the models and datasets, we use 300 dimensional GloVe (Pennington et al., 2014) vectors trained on 840 billion words for word embedding initialization and initialize out-of-vocabulary words with Gaussian distribution with the standard deviation of 0.6. We do not use the *text region embedding* (Johnson and Zhang, 2017), for fair comparisons with other comparative models. We optimize parameters using Adam (Kingma and Ba, 2015) with initial learning rate of 0.01 and batch size of 128. Gradients are clipped at 5.0 by norm. We use ReLU activation after convolution operations.

Model-specific configurations are as follows:

- *CNN*: We use the total of 300 filters, with 100 filters each having window size of 3,4 and 5.
- *DPCNN*: We use 100 filters with a size of 3, for each convolution operation. We set the depth to 11 for all the datasets except for the 'AG' dataset in which depth is set to 9.
- *DenseCNN*: We use 75 filters with a size of 3 for each convolution block. Input texts are padded or truncated to a fixed length. We set the fixed length to 300, except for the 'AG' in which 100 is used. For 'AG' dataset, we use six convolution blocks and seven for all the other datasets.

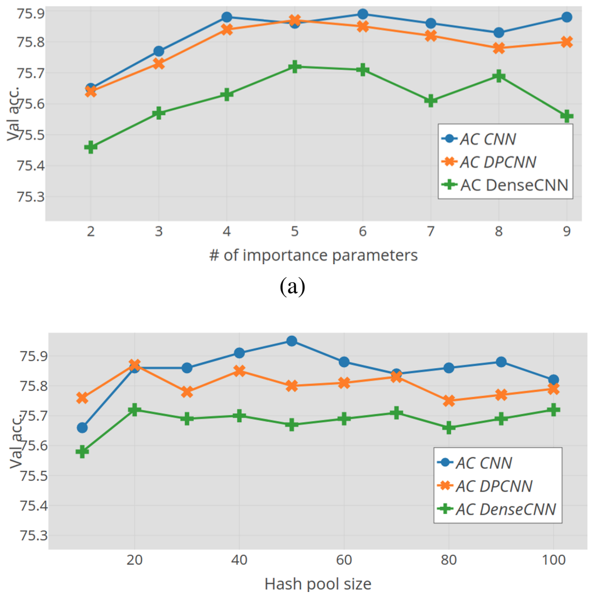
These configurations are set on the validation set held out by 10% from the training data. If not specified, the same configurations are used in all the datasets. Once we fit model settings, we apply our adaptive convolution to those settings. We use 600 for the context vector size (i.e. 300 for GRU hidden states). In the hashed generation, we use 20 for the hash (shared) pool size and five for the number of importance parameters.



(a)

(b)

Figure 2: Validation accuracies on Yahoo dataset with different model settings. (a) shows the results with different filter size, where x-axis is in logarithmic scale. (b) illustrates varying performance on different depths.



(a)

(b)

Figure 3: Validation accuracies on Yahoo dataset with different hashed generation settings. (a) shows the results with different number of importance parameters, and (b) depicts performance with different hash (shared) pool size.

4.2 Main Results

Table 2 shows the evaluation results on the datasets. In the table, *CharCNN* and *VDCNN* are character level CNNs. *FastText*, *SWEM* and *LEAM* are word embedding-based models. *HAN*, *KnnLSTM* and *Self-Attention* are RNN variants. '*AC*' in the model names indicate that adaptive convolution is applied in the model. Both 'full generation' and 'hashed generation' are filter-generating methods.

As shown in the table, adaptive convolutions improve all baseline CNNs in all datasets, with no exception. The performance improvements are relatively small for datasets in which known performances are already nearly 100%. In DBPedia dataset, performance improves by as much as 0.15 percentage point (%p) over the baselines. However, for datasets with a potential for considerable performance improvement, such as Yelp.f and Yahoo datasets, adaptive convolutions produce significant results. The performance improvements are up to 2.6%p for Yelp.f dataset.

Without adaptive convolutions, RNNs show better performance than CNNs on most datasets. However, when adaptive convolutions are applied, our baselines perform better than RNN variants on every dataset. Also our approaches perform better than word embedding-based models except for LEAM model on Yahoo dataset. Furthermore, our adaptive convolutions beat *ACNN* on all the datasets. This suggests that generating filters block by block with outputs from the previous convolution block is much more efficient than generating filters with original inputs in a single subnetwork.

Adaptive convolutions are found to be effective both in the full and hashed generation. The performance differences between the hashed generation and the full generation are within 0.3%p except for a few cases (AG and Yahoo datasets in *DenseCNN*). However, the hashed generation is much more efficient than the full generation in terms of the parameter size. Overall, the total number of parameters for models with the hashed generation is less than 3% that of the full generation (Table 3).

4.3 Analysis

Analysis on model settings To further demonstrate the effectiveness of our adaptive convolutions, we compare the performance with varying filter sizes and depths. We select *CNN* and *AC CNN* with filter sizes ranging from 2 to 150 to check the performance with different filter sizes. To analyze the effect of depths, we choose

DPCNN and *DenseCNN*, and their counterparts to which adaptive convolutions are applied. We evaluate the performance with depths ranging from 2 to 13.

The results are illustrated in Figure 2. Note that validation accuracies are lower than test accuracies because only 90% of the training set is used to save remaining 10% for the validation set. As can be seen in the figure, models adopted with our adaptive convolutions show stable performance. In case of filter sizes, *CNN* drastically drops performance when the filter size gets smaller. Its performance with the filter size of two is 8.4%p lower than that of the filter size of 100. However, the performance of *AC CNN* with the filter size of two declines by 0.6%p from the model with the filter size of 100, which is only 10 percent of the performance loss of *CNN*.

This tendency is also found in the analysis on depths. Performance of *DPCNN* and *DenseCNN* with depths of two or three are 0.6%p lower than that of the baselines with the best performing depths. Contrarily, models with adaptive convolutions perform only 0.12%p lower with depths of two or three than models with the best performing depths.

These results demonstrate that adaptive convolutions effectively generate filters for capturing important information that need to be disambiguated given current inputs. Only few filters and shallow depths are enough for the adaptive convolution to extract such information. This suggests that the required effort to tune hyperparameters can be mitigated by applying adaptive convolutions.

Additional noteworthy fact is that increasing filter sizes and depths beyond a certain level does not lead to performance improvements in all the baseline CNNs. In case of *CNN*, no change in performance is observed when the filter size exceeds 100. In case of *DenseCNN*, increasing depths more than seven rather results in a performance drop and increasing depths of *DPCNN* exceeding eleven has no effect on the performance. This implies that performance gain is caused by the effectiveness of the proposed adaptive convolution, instead of the increased number of parameters (Table. 3).

Analysis on hashed generation We investigate the effect of the hashed generation settings on the performance with different importance parameters and hash (shared) pool sizes. The number of importance parameters is ranging from 2 to 9 and the hash (shared) pool size is in the range from 10 to 100. The results are shown in Figure 3. As illustrated, increasing the sizes of hash pool and importance parameters beyond certain threshold does not guarantee performance gain. The number of importance parameter is optimal at five. Higher value of it has no effect on performance and in some cases, negatively affect performance. In case of the hash pool size, 20 is enough for containing candidate filters.

	AC CNN	*AC DPCNN*	*AC DenseCNN*
with GRU	75.86	75.87	75.72
w/o GRU	74.76	75.37	75.36

Table 4: Validation accuracies on Yahoo dataset for the hashed generation-based models with different context generation settings.

This observation supports the previous finding (Denil et al., 2013) that many redundant parameters exist in deep neural networks. Our results reveal that the networks can be parameterized with a set of candidate weights, and their size can be sufficiently small to significantly reduce the number of required parameters in the network with little performance loss.

Effect of GRU We perform an ablation test to validate the usage of GRUs in generating the context vector. Table 4 shows the results of the ablation test. In all models, utilizing GRUs in generating the context vector significantly improves performance as much as up to 1.1%p. This clearly indicates the existence of dependencies between entries in each layer. These can be effectively captured and incorporated into the context vector with GRUs and attention-based context vector generation scheme.

Filter visualization To better understand generated filters by adaptive convolutions with different inputs, we visualize filters with *t*-SNE (Maaten and Hinton, 2008). We compare filters trained with the baseline *CNN* as well as filters generated by *AC CNN* from different input texts. The corresponding results are shown in Figure 4. As clearly seen, filters from *CNN* are dispersed in the projected space. By contrast, filters generated by *AC CNN* with a positively and a negatively labeled sample are concentrated on the upper right

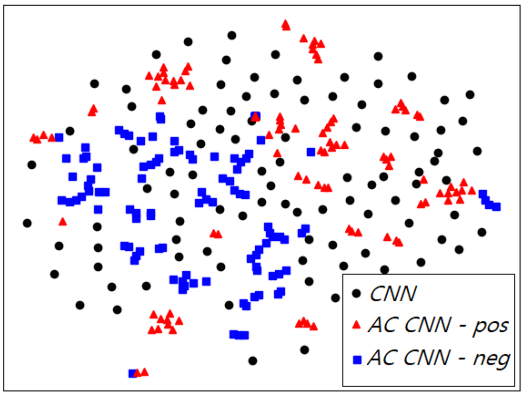


Figure 4: Filters visualized with *t*-SNE. Filters generated by *AC CNN* and trained in *CNN* on Yelp.p dataset are visualized. Negatively and positively labeled samples are used to generate filters from *AC CNN*.

and the lower left part of the space, respectively. This demonstrates that the generated filters in adaptive convolutions are focused to disambiguate uncertainty in given information. Filters trained in *CNN* are not specified to given inputs, but are generally tuned to solve given tasks.

5 Conclusion

In this paper, we have introduced the adaptive convolution to endow flexibility to convolution operations. Further, we have proposed the hashing technique which can drastically reduce the number of parameters for adaptive convolutions. We have validated our approach based on the performance evaluation with seven datasets, and investigated the effectiveness of adaptive convolutions through analysis. We believe that our methodology is applicable to other NLP tasks with text pairs, such as textual entailment, question answering. We plan to apply the proposed approach to those tasks in the future.

Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MIST) (No.2018R1A2A1A05078380). This research was also in part supported by the Information Technology Research Center (ITRC) support program supervised by the Institute for Information & communications Technology Promotion (IITP) (IITP-2019-2016-0-00464).

References

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. 2016. Learning feed-forward one-shot learners. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 523–531.

Andrei Z Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. Robust classification of rare queries using web knowledge. In *Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 231–238.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc V Gool. 2016. Dynamic filter networks. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 667–675.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. 2017. Modulating early visual processing by language. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 6594–6604.

Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. 2013. Predicting parameters in deep learning. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 2148–2156.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A learned representation for artistic style. In *International Conference on Learning Representations (ICLR)*.

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *International Conference on Learning Representations (ICLR)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 328–339.

Rie Johnson and Tong Zhang. 2016. Convolutional neural networks for text categorization: Shallow word-level vs. deep character-level. *arXiv preprint arXiv:1609.00718*.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proc. of European Chapter of the Association for Computational Linguistics (EACL)*, volume 2, pages 427–431.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, 1:655–665.

Di Kang, Debarun Dhar, and Antoni Chan. 2017. Incorporating side information by adaptive convolution. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 3867–3877.

Gabriella Kazai, Iskander Yusof, and Daoud Clarke. 2016. Personalised news and blog recommendations based on user location, facebook and twitter user profiling. In *Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1129–1132. ACM.

Taesup Kim, Inchul Song, and Yoshua Bengio. 2017. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In *Conference of the International Speech Communication Association (INTERSPEECH)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Benjamin Klein, Lior Wolf, and Yehuda Afek. 2015. A dynamic convolutional layer for short range weather prediction. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4848.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, volume 333, pages 2267–2273.

Jung-Hyun Lee, Jongwoo Ha, Jin-Yong Jung, and SangKeun Lee. 2013. Semantic contextual advertising based on the open directory project. *ACM Transactions on the Web (TWEB)*, 7(4):24.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations (ICLR)*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. In *Proc. of International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 4054–4060. AAAI Press.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 55–60.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. of International Conference on Machine Learning (ICML)*, pages 807–814.

Simon Niklaus, Long Mai, and Feng Liu. 2017. Video frame interpolation via adaptive convolution. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 670–679.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. 2016. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 30–38.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 4790–4798.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems (NIPS), Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, pages 3935–3942.

Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. 2018a. Learning context-sensitive convolutional filters for text processing. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1839–1848.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018b. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 440–450.

Dan Tito Svenstrup, Jonas Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 4928–4936.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018a. Joint embedding of words and labels for text classification. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2321–2331.

Shiyao Wang, Minlie Huang, and Zhidong Deng. 2018b. Densely connected cnn with multi-scale feature attention for text classification. In *Proc. of International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 4468–4474.

Zhiguo Wang, Wael Hamza, and Linfeng Song. 2017. k-nearest neighbor augmented neural networks for text classification. *arXiv preprint arXiv:1708.07863*.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proc. of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pages 649–657.

Zero-Shot Cross-Lingual Opinion Target Extraction

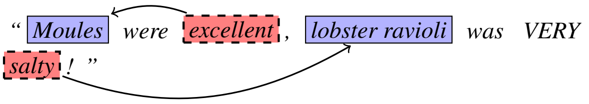
Soufian Jebbara and Philipp Cimiano
Semalytix GmbH, Bielefeld, Germany
Semantic Computing Group, CITEC - Bielefeld University, Bielefeld, Germany
`soufian.jebbara@semalytix.com`
`cimiano@cit-ec.uni-bielefeld.de`

Abstract

Aspect-based sentiment analysis involves the recognition of so called opinion target expressions (OTEs). To automatically extract OTEs, supervised learning algorithms are usually employed which are trained on manually annotated corpora. The creation of these corpora is labor-intensive and sufficiently large datasets are therefore usually only available for a very narrow selection of languages and domains. In this work, we address the lack of available annotated data for specific languages by proposing a zero-shot cross-lingual approach for the extraction of opinion target expressions. We leverage multilingual word embeddings that share a common vector space across various languages and incorporate these into a convolutional neural network architecture for OTE extraction. Our experiments with 5 languages give promising results: We can successfully train a model on annotated data of a source language and perform accurate prediction on a target language without ever using any annotated samples in that target language. Depending on the source and target language pairs, we reach performances in a zero-shot regime of up to 77% of a model trained on target language data. Furthermore, we can increase this performance up to 87% of a baseline model trained on target language data by performing cross-lingual learning from multiple source languages.

1 Introduction

In recent years, there has been an increasing interest in developing sentiment analysis models that predict sentiment at a more fine-grained level than at the level of a complete document. A paradigm coined as Aspect-based Sentiment Analysis (ABSA) addresses this need by defining the sentiment expressed in a text relative to an opinion target (also called *aspect*). Consider the following example from a restaurant review:



In this example, there are two sentiment statements, one positive and one negative. The positive one is indicated by the word “*excellent*” and is expressed towards the opinion target “*Moules*”. The second, negative sentiment, is indicated by the word “*salty*” and is expressed towards the “*lobster ravioli*”.

A key task within this fine-grained sentiment analysis consists of identifying so called *opinion target expressions* (OTE). To automatically extract OTEs, supervised learning algorithms are usually employed which are trained on manually annotated corpora. In this paper, we are concerned with how to transfer classifiers trained on one domain to another domain. In particular, we focus on the transfer of models across languages to alleviate the need for multilingual training data. We propose a model that is capable of accurate zero-shot cross-lingual OTE extraction, thus reducing the reliance on annotated data for every language. Similar to Upadhyay et al. (2018), our model leverages multilingual word embeddings (Smith et al., 2017; Lample et al., 2018) that share a common vector space across various languages. The shared space allows us to transfer a model trained on source language data to predict OTEs in a target language for which no (i.e. zero-shot setting) or only small amounts of data are available, thus allowing to apply our model to under-resourced languages.

Our main contributions can be summarized as follows:

- We present the first approach for zero-shot cross-lingual opinion target extraction and achieve up to 87% of the performance of a monolingual baseline.
- We investigate the benefit of using multi-

Proceedings of NAACL-HLT 2019, pages 2486–2495
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

ple source languages for cross-lingual learning and show that we can improve by 6 to 8 points in F_1-Score compared to a model trained on a single source language.

- We investigate the benefit of augmenting the zero-shot approach with additional data points from the target language. We observe that we can save hundreds of annotated data points by employing a cross-lingual approach.
- We compare two methods for obtaining cross-lingual word embeddings on the task.

2 Approach

A common approach for extracting opinion target expressions is to phrase the task as a sequence tagging problem using the well-known IOB scheme (Tjong Kim Sang and Veenstra, 1999) to represent OTEs as a sequence of tags. According to this scheme, each word in our text is marked with one of three tags, namely **I**, **O** or **B** that indicate if the word is at the **B**eginning[1], **I**nside or **O**utside of a target expression. An example of such an encoding can be seen below:

The	*wine*	*list*	*is*	*also*	*really*	*nice*	*.*
O	**I**	**I**	O	O	O	O	O

By rephrasing the task in this way, we can address it using established sequence tagging models. In this work, we use a multi-layer convolutional neural network (CNN) as our sequence tagging model. The model receives a sequence of words as input features and predicts an output sequence of IOB tags. In order to keep our model simple and our results clear, we restrict our input representation to a sequence of word embeddings. While additional features such as Part-of-Speech (POS) tags are known to perform well in the domain of OTE extraction (Toh and Su, 2016; Kumar et al., 2016; Jebbara and Cimiano, 2016), they would require a separately trained model for POS-tag prediction which can not be assumed to be available for every language. We refrain from using more complex architectures such as memory networks as our goal is mainly to investigate the possibility of performing zero-shot cross-lingual transfer learning for OTE prediction. Being the first approach proposing this, we leave the question of how to increase performance of the approach by using more complex architectures to future work.

In the following, we describe our monolingual CNN model for OTE extraction which we use as our baseline model. Afterwards, we show how we adapt this model for a cross-lingual and even zero-shot regime.

2.1 Monolingual Model

Our monolingual baseline model consists of a word embedding layer, a stack of convolution layers, a standard feed-forward layer followed by a final output layer. Formally, the word sequence $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$ is passed to the word embedding layer that maps each word $\mathbf{w}_i$ to its embedding vector $\mathbf{x}_i$ using an embedding matrix $\mathbf{W}$. The sequence of word embedding vectors $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is processed by a stack of L convolutional layers[2], each with a kernel width of l^{conv}, d^{conv} filter maps and RELU activation function f (Nair and Hinton, 2010). The final output of these convolution layers is a sequence of abstract representations $\mathbf{h}^L = (\mathbf{h}_1^L, \ldots, \mathbf{h}_n^L)$ that incorporate the immediate context of each word by means of the learned convolution operations. The hidden states $\mathbf{h}_i^L$ of the last convolution layer are processed by a regular feed-forward layer to further increase the model's capacity and the resulting sequence is passed to the output layer.

In a last step, each hidden state is projected to a probability distribution over all possible output tags $q_i = (q_i^B, q_i^I, q_i^O)$ using a standard feed-forward layer with weights $\mathbf{W}^{tag}$, bias $\mathbf{b}^{tag}$ and a softmax activation function.

Since the prediction of each tag can be interpreted as a classification, the network is trained to minimize the categorical cross-entropy between expected tag distribution p_i and predicted tag distribution q_i of each word i:

$$H(p_i, q_i) = -\sum_{t \in \mathcal{T}} p_i^t \log(q_i^t),$$

where $\mathcal{T} = \{I, O, B\}$ is the set of IOB tags, $p_i^t \in \{0, 1\}$ is the expected probability of tag t and $q_i^t \in [0, 1]$ the predicted probability. Figure 1 depicts the sequence labeling architecture.

---

[1] Note that the **B** token is only used to indicate the boundary of two consecutive phrases.

[2] The input sequences are padded with zeros to allow the application of the convolution operations to the edge words.

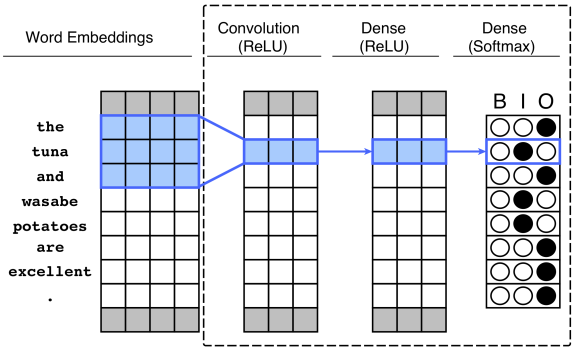


Figure 1: Model for sequence tagging using convolution operations. For simplicity, we only show a single convolution operation. The gray boxes depict padding vectors. The layers inside the dashed box are shared across multiple languages.

2.2 Cross-Lingual Model

Our cross-lingual model works purely with cross-lingual embeddings that have been trained on monolingual datasets and in a second step have been aligned across languages. In fact, the embeddings are pre-computed in an offline fashion and are not adapted while training the convolutional network on data from a specific language. As the inputs to the convolutional network are only the cross-lingual embeddings, the network can be applied to any language for which the embeddings have been aligned.

Since the word embeddings for source and target language share a common vector space, the shared parts of the target language model are able to process data samples from the completely unseen target language and perform accurate prediction i.e. enabling zero-shot cross-lingual extraction of opinion target expressions.

We rely on two approaches to compute embeddings that are aligned across languages. Both methods rely on fastText (Bojanowski et al., 2017) to compute monolingual embeddings trained on Wikipedia articles. The first method is the one proposed by Smith et al. (2017), which computes a singular value decomposition (SVD) on a dictionary of translated word pairs to obtain an optimal, orthogonal projection matrix from one space into the other. We refer to this method as `SVD-aligned`. We use these embeddings[3] in our experiments in Sections 3.3, 3.4 and 3.6.

The second method proposed by Lample et al. (2018) performs the alignment of embeddings across languages in an unsupervised fashion, without requiring translation pairs.

The approach uses adversarial training to initialize the cross-lingual mapping and a synthetically generated bilingual dictionary to fine-tune it with the Procrustes algorithm (Schönemann, 1966). We refer to the multilingual embeddings[4] from Lample et al. (2018) as `ADV-aligned`. These are used in Section 3.5.

3 Evaluation

In this section, we investigate the proposed zero-shot cross-lingual approach and evaluate it on the widely used dataset of Task 5 of the SemEval 2016 workshop. With our evaluation, we answer the following research questions:

RQ1: To what degree is the model capable of performing OTE extraction for unseen languages?

RQ2: Is there a benefit in training on more than one source language?

RQ3: What improvements can be expected when a small amount of samples for the target language are available?

RQ4: How big is the impact of the used alignment method on the OTE extraction performance?

Before we answer these questions, we give a brief overview over the used datasets and resources.

3.1 Datasets

As part of Task 5 of the SemEval 2016 workshop (Pontiki et al., 2016), a collection of datasets for aspect-based sentiment analysis on various languages and domains was published. Due to its relatively large number of samples and high coverage of languages and domains, the datasets are commonly used to evaluate ABSA approaches. To answer our research questions, we make use of a selection of the available datasets. We evaluate our cross-lingual approach on the available datasets for the restaurant domain for the 5 languages Dutch (`nl`), English (`en`), Russian (`ru`), Spanish (`es`) and Turkish (`tr`)[5]. Table 1 gives a brief overview of the used datasets.

---

[3] Obtained from: `https://github.com/Babylonpartners/fastText_multilingual`

[4] Obtained from: `https://github.com/facebookresearch/MUSE`

[5] We tried to include the dataset of French reviews in our evaluation but the provided download script no longer works.

Dataset	#Sent.	#Tokens	#Targets
`en` (train)	2000	29278	1880
`en` (test)	676	10080	650
`es` (train)	2070	36164	1937
`es` (test)	881	13290	731
`nl` (train)	1722	24981	1283
`nl` (test)	575	7690	394
`ru` (train)	3655	53734	3159
`ru` (test)	1209	17856	972
`tr` (train)	1232	12702	1385
`tr` (test)	144	1360	159

Table 1: Statistics of the SemEval 2016 ABSA dataset for the restaurant domain.

3.2 Experimental Settings

In all our experiments, we report F_1-scores for the extracted opinion target expressions computed on exact matches of the character spans as in the original SemEval task (Pontiki et al., 2016).

As described in Section 2.2, our model relies on pretrained multilingual embeddings. For both `SVD-aligned` and `ADV-aligned`, we use the embeddings as provided by the original authors. However, we restrict our vocabulary to the most frequent 50,000 words per language[6] to reduce memory consumption.

For all experiments, we fix our model architecture to 5 convolution layers with each having a kernel size of 3, a dimensionality of 300 units and a ReLU activation function (Nair and Hinton, 2010). The penultimate feed-forward layer has 300 dimensions and a ReLU activation, as well. We apply dropout (Srivastava et al., 2014) on the word embedding layer with a rate of 0.3 and between all other layers with 0.5. The word embeddings and the penultimate layer are L1-regularized (Ng, 2004).

The network's parameters are optimized using the stochastic optimization technique *Adam* (Kingma and Ba, 2015). We optimize the number of training epochs for each model using early stopping (Caruana et al., 2000) but do not tune other hyperparameters of our models. We always pick 20% of our available training data for the validation process. For the zero-shot scenario, this entails that we optimize the number of epochs on the source language and not on the target language to simulate true zero-shot learning.

3.3 Zero-Shot Transfer Learning

In this section, we present our evaluation for zero-shot learning. We first examine a setting with a single source language. Then, we evaluate the effect of cross-lingual learning from multiple source languages.

Single Source Language This part of our evaluation addresses our first research question:

RQ1: To what degree is the model capable of performing OTE extraction for unseen languages?

To answer this question, we perform a set of experiments in the zero-shot setting. We train a model on the training portion of a source language and evaluate the model performance on all possible target languages. Figure 2 shows the obtained scores. The reported results are averaged over 10 runs with different random seeds. The main diagonal represents results of models both trained and tested on target language data. We considered these our monolingual baselines.

In general, the proposed approach achieves relatively high scores for some language pairs, although with large performance differences depending on the exact source and target language pairs. Looking at the absolute scores, the best performing cross-lingual language pair is `en`→`es` with an F_1-score of 0.5. This is followed by `en`→`nl` at 0.46. The lowest is `es`→`tr` with an F_1-score of 0.14. When considering the results relative to their respective monolingual baselines, the highest relative performance is achieved by `en`→`nl` at 77% of a `nl`→`nl` model, followed by `en`→`es` and `ru`→`nl`, which both reach an F-Measure of about 74%. The weakest performing language pair is still `es`→`tr` at 29% relative performance. In general, the Turkish language seems to benefit the least from the cross-lingual transfer learning, while Russian is on average the best source language in terms of relative performance achievement for the target languages.

Overall, the presented results show that it is in fact possible for most considered languages to train a model for OTE extraction without ever using any annotated data in that target language.

Multiple Source Languages In the next experiment, we want to address our second research question:

---

[6] As appearing in the respective embedding files.

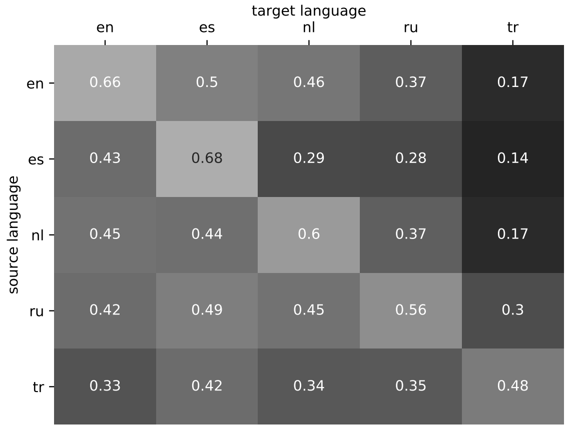


Figure 2: Zero-shot F_1-scores for cross-lingual learning from a single source to a target language.

target	en	es	nl	ru	tr
best→target	0.45	0.50	0.46	0.37	0.30
all others→target	0.52	0.58	0.53	0.43	0.27
target→target	0.66	0.68	0.60	0.56	0.48

Table 2: Zero-shot results for cross-lingual learning from multiple source languages to a target language. The row `best→target` represents the best performing cross-lingual model from Figure 2 for each target language. `all others→target` are the results for training on all languages except for the target language. `target→target` shows the monolingual scores that act as a baseline.

RQ2: Is there a benefit in training on more than one source language?

As we explained in Section 2.2, our approach allows us to train and test on any number of source and target languages, provided that we have aligned word embeddings for each considered language.

In order to answer our second research question, we train a model on the available training data for all but one language and perform prediction on the test data for the left-out language. The results for these experiments are summarized in Table 2. We can see that all languages with the exception of Turkish seem to profit from a cross-lingual transfer setting with multiple source languages. The absolute improvements are in the range of 6 to 8 points in F_1-Score while the performance on Turkish samples drops by 3 points.

We can summarize that we can obtain substantial improvements for most languages when training on a combination of multiple source languages. In fact, for `en`, `es`, `nl` and `ru`, the results of our cross-lingual models trained on all other languages reach between 78% to 87% relative performance of a model trained with target language data.

3.4 Cross-Lingual Transfer Learning with Additional Target Language Data

While our goal is to reduce the effort of annotating huge amounts of data in a target language to which the model is to be transferred, it might still be reasonable to provide a few annotated samples for a target language. Our next research question addresses this issue:

RQ3: What improvements can be expected when a small amount of samples for the target language are available?

We answer this question by training our models jointly on a source language dataset as well as a small amount of target language samples and compare this to a baseline model that only uses target language samples. By gradually increasing the available target samples, we can directly observe their benefit on the test performance. Figure 3 shows a visualization for the source language `en` and the target languages `es`, `nl`, `ru`, and `tr`.

We can immediately see that a monolingual model requires at least 100 target samples to produce meaningful results as opposed to a cross-lingual model that performs well with source language samples alone. Training on increasing amounts of target samples improves the model performances monotonically for each target language and the model leveraging the bilingual data consistently outperforms the monolingual baseline model. The benefits of the source language data are especially pronounced when very few target samples are available, i.e. less than 200. As an example, a model trained on bilingual data using all available English samples and 200 Dutch samples is competitive to a monolingual model trained on 1000 Dutch samples (0.55 vs. 0.56).

As one would expect, the results in Table 2 and Figure 3 suggest that training the model on more data samples leads to a better performance. Since our model can leverage the data from all languages simultaneously, we can exhaust our resources and train an instance of our model that has access to all training data samples from all languages, including the target training data. This is reflected by the dashed line in Figure 3. We see, however, that the model cannot leverage the other source languages

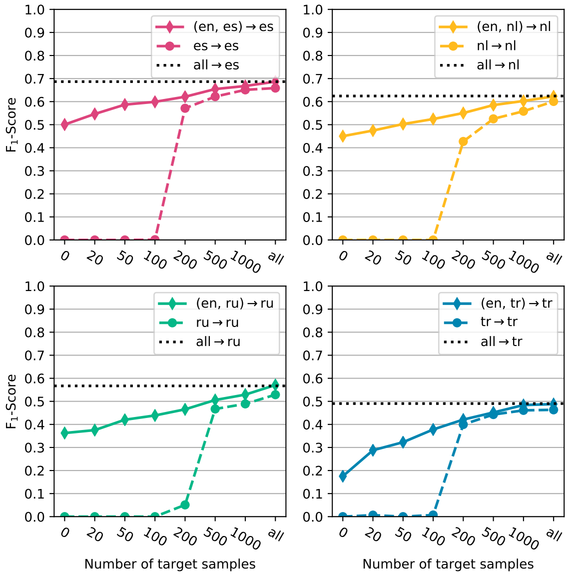
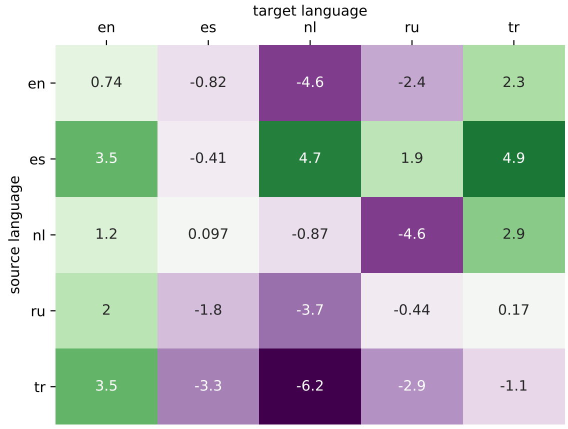


Figure 3: Cross-lingual results for increasing numbers of training samples from the target language.



	en	es	nl	ru	tr
en	0.74	-0.82	-4.6	-2.4	2.3
es	3.5	-0.41	4.7	1.9	4.9
nl	1.2	0.097	-0.87	-4.6	2.9
ru	2	-1.8	-3.7	-0.44	0.17
tr	3.5	-3.3	-6.2	-2.9	-1.1

Figure 4: Zero-shot results comparing the multilingual embeddings `ADV-aligned` to `SVD-aligned`. A positive value means higher absolute F_1 score for `ADV-aligned` and vice versa. For readability, score differences are scaled by a factor of 100.

beyond what it achieves with the combination of the full target and English language data alone.

3.5 Comparison of Alignment Methods

The previous experiments show that we can achieve good performance in a cross-lingual setting for OTE extraction using the multilingual word embeddings proposed by Smith et al. (2017). Now we address our final research question:

RQ4: How big is the impact of the used alignment method on the OTE extraction performance?

With our final research question, we compare our previous results to an alternative method of aligning word embeddings in multiple languages. We repeat our experiments in Section 3.3 using the embeddings of Lample et al. (2018) which we refer to as `ADV-aligned`.

To enable a direct comparison to the zero-shot results in Section 3.3, we report absolute differences in F_1-Score to the scores obtained with `SVD-aligned` for all source and target language combinations.

As can be seen in Figure 4, the two methods do perform well overall, albeit different for specific language pairs. In a monolingual setting (i.e. main diagonal), `ADV-aligned` performs slightly worse than `SVD-aligned` with the exception of `en`→`en`. Using `ADV-aligned`, Spanish appears to be a more effective source language than using `SVD-aligned` as the average performance is about 2.9 points higher. It can also be observed that the cross-lingual transfer learning works better for English as a target language using `ADV-aligned` since the average performance is about 2.2 points higher than for `SVD-aligned`. The opposite is true for Dutch as a target language, which shows a reduction in performance by 2.1 points on average. Overall, for 13 of the 25 language pairs, the embeddings based on `SVD-aligned` perform better than embeddings aligned with `ADV-aligned`.

3.6 Comparison to State-of-the-Art

In this last part of our evaluation, we want to put our work into perspective of prior systems for opinion target extraction on the SemEval 2016 restaurant datasets. We report results for our multilingual model that is trained on the combined training data of all languages and evaluated on the corresponding test datasets. We compare our model to the respective state-of-the-art for each language in Table 3.

We can see that the competition is strongest for English where we fall behind recent monolingual systems. This corresponds to rank 7 of 19 of the original SemEval competition. Regarding the other languages, we see that we are close to the best Spanish and Dutch systems and even clearly outperform systems for Russian and Turkish by at least 7 points in F_1-score. With that, we present the first approach on this task to achieve such competitive performances for a variety of languages

System	`en`	`es`	`nl`	`ru`	`tr`
Toh and Su (2016)	0.723	–	–	–	–
Àlvarez-López et al. (2016)	0.666	0.685	–	–	–
Kumar et al. (2016)	0.685	**0.697**	**0.644**	–	–
Pontiki et al. (2016)*	0.441	0.520	0.506	0.493	0.419
Li and Lam (2017)	**0.734**	–	–	–	–
`all→target` (Ours)	0.660	0.687	0.624	**0.567**	**0.490**

Table 3: Overview of the current state-of-the-art for opinion target extraction for 5 languages. Our model is trained on the combined training data of all languages and evaluated on the respective test datasets. The row marked with * is the baseline provided by the workshop organizers. To our knowledge, no better model is published for Russian and Turkish.

with a single, multilingual model.

3.7 Discussion and Future Work

The presented experiments shed light on the performance of our proposed approach under various circumstances. In the following, we want to discuss its limitations and consider explanations for performance differences of different language pairs.

Model Limitations The core of our proposed sequence labeling approach consists of aligned word embeddings and shared CNN layers. Due to the limited context of a CNN layer, the model can only base its decisions for each word on the local information around that word. In many cases, this information is sufficient since most opinion target expressions are adjective-noun phrases[7] which are well enough identified by the local context for most considered languages.

As future work, it is worth to investigate in how far our findings translate to more complex model architectures that have been proposed for OTE extraction, such as memory networks or attention-based models.

Language Characteristics Due to the inherent variability of natural languages and of the used datasets, it is difficult to identify the exact reasons for the observed performance differences between language pairs. However, we suspect that language features such as word order, inflection, or agglutination affect the *compatibility* of languages. As an example, Turkish is considered a highly agglutinative language, that is, complex words are composed by attaching several suffixes to a word stem. This sets it apart from the other 4 languages. This language feature might present a difficulty in our approach since the appending of suffixes is not optimally reflected in the tokenization process and the used word embeddings. An approach that performs alignment of languages on subword units might alleviate this problem and lead to performance gains for language pairs with similar inflection rules.

Syntactic regularities such as word order might also play a role in our transfer learning approach. It is reasonable to assume that the CNN layers of our approach pick up patterns in the word order of a source language that are indicative of an opinion target expression, e.g. *"the [NOUN] is good"*. When applying such a model to a target language with drastically different word order regularities, these patterns might not appear as such in the target language.

For the considered languages, we see following characteristics: Where English and Spanish are generally considered to follow a Subject-Verb-Object (SVO) order, Dutch largely exhibits a combination of SOV and SVO cases. Turkish and Russian are overall flexible in their word order and allow a variety of syntactic structures. In the case of Turkish, its morphological and syntactic features seem to explain some of the relatively low results. However, with the small sample of languages and the many potential influencing factors at play, we are aware that it is not possible to draw any strong conclusions. Further research has to be conducted in this direction to answer open questions.

4 Related Work

Our work brings together the domains of opinion target extraction on the one side and cross lingual learning on the other side. In this section, we give a brief overview of both domains and point out parallels to previous work.

Opinion Target Extraction San Vicente et al. (2015) present a system that addresses opinion target extraction as a sequence labeling problem based on a perceptron algorithm with token, word shape and clustering-based features.

Toh and Wang (2014) propose a Conditional Random Field (CRF) as a sequence labeling model that includes a variety of features such as Part-of-Speech (POS) tags and dependency tree features, word clusters and features derived from the WordNet taxonomy. The model is later improved us-

---

[7] 90% of OTEs in the English dataset consist of zero or more adjectives followed by at least one noun.

ing neural network output probabilities (Toh and Su, 2016) and achieved the best results on the SemEval 2016 dataset for English restaurant reviews.

Jakob and Gurevych (2010) follow a very similar approach that addresses opinion target extraction as a sequence labeling problem using CRFs. Their approach includes features derived from words, Part-of-Speech tags and dependency paths, and performs well in a single and cross-domain setting.

Kumar et al. (2016) present a CRF-based model that makes use of a variety of morphological and linguistic features and is one of the few systems that submitted results for more than one language for the SemEval 2016 ABSA challenge. The strong reliance on high-level NLP features, such as dependency trees, named-entity information and WordNet features restricts its wide applicability to resource-poor languages.

Among neural network models Poria et al. (2016) and Jebbara and Cimiano (2016) use deep convolutional neural network (CNN) with Part-of-Speech (POS) tag features. Poria et al. (2016) also extend their base model using linguistic rules.

Wang et al. (2017) use coupled multi-layer attentions to extract opinion expressions and opinion targets jointly. This approach, however, relies on additional annotations for opinion expressions alongside annotations for the opinion targets.

Li and Lam (2017) propose two LSTMs with memory interaction to detect aspect and opinion terms. In order to generate opinion expression annotations for the SemEval dataset, a sentiment lexicon is used in combination with high precision dependency rules.

For a more comprehensive overview of ABSA and OTE extraction approaches we refer to Pontiki et al. (2016).

Cross-Lingual and Zero-Shot Learning for Sequence Labelling With the CLOpinionMiner, Zhou et al. (2015) present a method for cross-lingual opinion target extraction that relies on machine translation. The approach derives an annotated dataset for a target language by translating the annotated source language data. Part-of-Speech tags and dependency path-features are projected into the translated data using the word alignment information of the translation algorithm. The approach is evaluated for English to Chinese reviews. A drawback of the presented method is that it requires access to a strong machine translation algorithm for source to target language that also provides word alignment information. Additionally, it builds upon NLP resources that are not available for many potential target languages.

Addressing the task of zero-shot spoken language understanding (SLU), Upadhyay et al. (2018) follow a similar approach as our work. They use the aligned embeddings from Smith et al. (2017) in combination with a bidirectional RNN and target zero-shot SLU for Hindi and Turkish.

Overall, our work differs from the related work by presenting a simple model for the zero-shot extraction of opinion target expressions. By using no annotated target data or elaborate NLP resources, such as Part-of-Speech taggers or dependency parsers, our approach is easily applicable to many resource-poor languages.

5 Conclusion

In this work, we presented a method for cross-lingual and zero-shot extraction of opinion target expressions which we evaluated on 5 languages. Our approach uses multilingual word embeddings that are aligned into a single vector space to allow for cross-lingual transfer of models.

Using English as a source language in a zero-shot setting, our approach was able to reach an F_1-score of 0.50 for Spanish and 0.46 for Dutch. This corresponds to relative performances of 74% and 77% compared to a baseline system trained on target language data. By using multiple source languages, we increased the zero-shot performance to F_1-scores of 0.58 and 0.53, respectively, which correspond to 85% and 87% in relative terms. We investigated the benefit of augmenting the zero-shot approach with additional data points from the target language. Here, we observed that we can save several hundreds of annotated data points by employing a cross-lingual approach. Among the 5 considered languages, Turkish seemed to benefit the least from cross-lingual learning in all experiments. The reason for this might be that Turkish is the only agglutinative language in the dataset. Further, we compared two approaches for aligning multilingual word embeddings in a single vector space and found their results to vary for individual language pairs but to be comparable overall. Lastly, we compared our multilingual model with the state-of-the-art for all languages and saw that we achieve competitive performances for some languages and even present the best system for Russian and Turkish.

Acknowledgments

This work was supported in part by the H2020 project Prêt-à-LLOD under Grant Agreement number 825182.

References

Tamara Àlvarez-López, Jonathan Juncal-Martínez, Milagros Fernández Gavilanes, Enrique Costa-Montenegro, and Francisco Javier González-Castaño. 2016. GTI at SemEval-2016 Task 5: SVM and CRF for Aspect Detection and Unsupervised Aspect-Based Sentiment Analysis. In *SemEval@NAACL-HLT*, pages 306–311. The Association for Computer Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2000. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In *NIPS*, pages 402–408. MIT Press.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045.

Soufian Jebbara and Philipp Cimiano. 2016. Aspect-Based Relational Sentiment Analysis Using a Stacked Neural Network Architecture. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, pages 1123—-1131.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis. In *SemEval@NAACL-HLT*, pages 1129–1135. The Association for Computer Linguistics.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herv Jgou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Xin Li and Wai Lam. 2017. Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2886–2892. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Andrew Y. Ng. 2004. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA. ACM.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia V. Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez Zafra, and Gülsen Eryigit. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 19–30.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect Extraction for Opinion Miningwith a Deep Convolutional Neural Network. *Knowledge-Based Systems*, 108:42–49.

Iñaki San Vicente, Xabier Saralegi, and Rodrigo Agerri. 2015. Elixa: A modular and flexible ABSA platform. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 748–752, Denver, Colorado. Association for Computational Linguistics.

Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of European Chapter of the ACL (EACL)*, pages 173–179. Bergen, Norway.

Zhiqiang Toh and Jian Su. 2016. NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016*, volume 2015, pages 282–288.

Zhiqiang Toh and Wenting Wang. 2014. DLIREC: Aspect Term Extraction and Term Polarity Classification System. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 235–240.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2018. (Almost) Zero-Shot Cross-Lingual Spoken Language Understanding. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled Multi-Layer Attentions for Co-Extraction of Aspect and Opinion Terms. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3316–3322.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2015. CLOpinionMiner: Opinion Target Extraction in a Cross-Language Scenario. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23:1–1.

Adversarial Category Alignment Network for Cross-domain Sentiment Classification

Xiaoye Qu[1*] **Zhikang Zou**[1*] **Yu Cheng**[2] **Yang Yang**[3] **Pan Zhou**[1†]
[1]Huazhong University of Science and Technology
[2]Microsoft AI & Research
[3]University of Electronic Science and Technology of China
{xiaoye, panzhou}@hust.edu.cn | yu.cheng@microsoft.com
{zhikangzou001, dlyyang}@gmail.com

Abstract

Cross-domain sentiment classification aims to predict sentiment polarity on a target domain utilizing a classifier learned from a source domain. Most existing adversarial learning methods focus on aligning the global marginal distribution by fooling a domain discriminator, without taking category-specific decision boundaries into consideration, which can lead to the mismatch of category-level features. In this work, we propose an adversarial category alignment network (ACAN), which attempts to enhance category consistency between the source domain and the target domain. Specifically, we increase the discrepancy of two polarity classifiers to provide diverse views, locating ambiguous features near the decision boundaries. Then the generator learns to create better features away from the category boundaries by minimizing this discrepancy. Experimental results on benchmark datasets show that the proposed method can achieve state-of-the-art performance and produce more discriminative features.

1 Introduction

Sentiment classification aims to automatically identify the sentiment polarity (i.e., positive or negative) of the textual data. It has attracted a surge of attention due to its widespread applications, ranging from movie reviews to product recommendations. Recently, deep learning-based methods have been proposed to learn good representations and achieved remarkable success. However, the performances of these works are highly dependent on manually annotated training data while annotation process is time-consuming and expensive. Thus, cross-domain sentiment classification, which aims to transfer knowledge learned on labeled data from related domains (called source domain) to a new domain (called target domain), becomes a promising direction.

One key challenge of cross-domain sentiment classification is that the expression of emotional tendency usually varies across domains. For instance, considering reviews about two sorts of products: **Kitchen** and **Electronics**. One set of reviews would contain opinion words such as “delicious” or “tasty”, and the other “rubbery” or “blurry”, to name but a few. Due to the small intersection of two domain words, it remains a significant challenge to bridge the two domains divergence effectively.

Researchers have developed many algorithms for cross-domain sentiment classification in the past. Traditional pivot-based works (Blitzer et al., 2007; Yu and Jiang, 2016) attempt to infer the correlation between pivot words, i.e., the domain-shared sentiment words, and non-pivot words, i.e., the domain-specific sentiment words by utilizing multiple pivot prediction tasks. However, these methods share a major limitation that manual selection of pivots is required before adaptation. Recently, several approaches (Sun et al., 2016; Zellinger et al., 2017) focus on learning domain invariant features whose distribution is similar in source and target domain. They attempt to minimize the discrepancy between domain-specific latent feature representations. Following this idea, most existing adversarial learning methods (Ganin et al., 2016; Li et al., 2017) reduce feature difference by fooling a domain discriminator. Despite the promising results, these adversarial methods suffer from inherent algorithmic weakness. Even if the generator perfectly fools the discriminator, it merely aligns the marginal distribution of the two domains and ignores the category-specific decision boundaries. As shown in Figure 1 (left), the generator may generate ambiguous or even mismatched features near the decision boundary, thus

---

[*] Equal contribution
[†] Corresponding author

Proceedings of NAACL-HLT 2019, pages 2496–2508
Minneapolis, Minnesota, June 2 - June 7, 2019.

hindering the performance of adaptation.

To address the aforementioned limitations, we propose an adversarial category alignment network (ACAN) which enforces the category-level alignment under a prior condition of global marginal alignment. Based on the cluster assumption in (Chapelle et al., 2009), the optimal predictor is constant on high density regions. Thus, we can utilize two classifiers to provide diverse views to detect points near the decision boundaries and train the generator to create more discriminative features into high-density region. Specifically, we first maximize the discrepancy of the outputs of two classifiers to locate the inconsistent polarity prediction points. Then the generator is trained to avoid these points in the feature space by minimizing the discrepancy. In such an adversarial manner, the ambiguous points are kept away from the decision boundaries and correctly distinguished, as shown in Figure 1 (right).

We evaluate our method on the Amazon reviews benchmark dataset which contains data collected from four domains. ACAN is able to achieve the state-of-the-art results. We also provide analyses to demonstrate that our approach can generate more discriminative features than the approaches only aligning global marginal distribution (Zhuang et al., 2015).

2 Related Work

Sentiment Classification: Deep learning based models have achieved great success on sentiment classification (Zhang et al., 2011). These models usually contain one embedding layer which maps each word to a dense vector, and different network architectures then process combined word vectors to generate a representation for classification. According to diverse network architectures, four categories are divided including Convolutional Neural Networks (CNNs) (Kalchbrenner et al., 2014; Kim, 2014), Recurrent Neural Networks (RNNs) (Yang et al., 2016; Zhou et al., 2016b), Recursive Neural Networks (RecNNs) (Socher et al., 2013) and other neural networks (Iyyer et al., 2015).

Domain Adaption: The fundamental challenge to solve the domain adaptation lies here is that data from the source domain and target domain have different distributions. To alleviate this difference, there are many pivot-based methods (Blitzer et al., 2007; He et al., 2011; Gouws et al., 2012; Yu and Jiang, 2016; Ziser and Reichart, 2018) which try to align domain-specific opinion (non-pivot) words through domain-shared opinion (pivot) words as the expression of emotional tendency usually varies across domains, which is a major reason of the domain difference. However, selecting pivot words for these methods first is very tedious, and the pivot words they find may not be accurate. Apart from pivot-based methods, denoising auto-encoders (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2014) have been extensively explored to learn transferable features during domain adaption by reconstructing noise input. Despite their promising results, they are based on discrete representation. Recently, some adversarial learning methods (Ganin et al., 2016; Li et al., 2017, 2018) propose to reduce this difference by minimizing the distance between feature distributions. But these methods solely focus on aligning the global marginal distribution by fooling a domain discriminator, which can lead to the mismatch of category-level features. To solve this issue, we propose to further align the category-level distribution by taking the decision boundary into consideration. Some recent works with class-level alignment have been explored in computer vision applications (Saito et al., 2017, 2018).

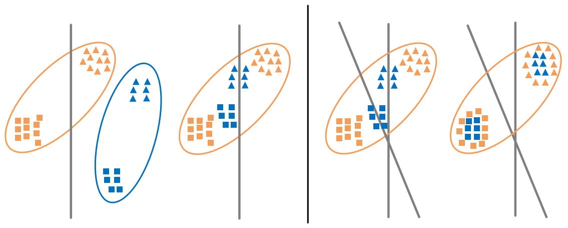


Figure 1: **Left:** marginal distribution alignment by minimizing the distance between two domains can generate ambiguous feature near the decision boundary. **Right:** two different classifiers locate ambiguous features by considering decision boundary to make category-level alignment.

Semi-supervised learning: Considering the target samples as unlabeled data, our work is somehow related to semi-supervised learning (SSL). SSL has several critical assumptions, such as cluster assumption that the optimal predictor is constant or smooth on connected high density regions (Chapelle et al., 2009), and manifolds assumption that support set data lies on low-dimensional manifolds (Chapelle et al., 2009; Luo et al., 2017). Our work takes these assumptions to develop the approach.

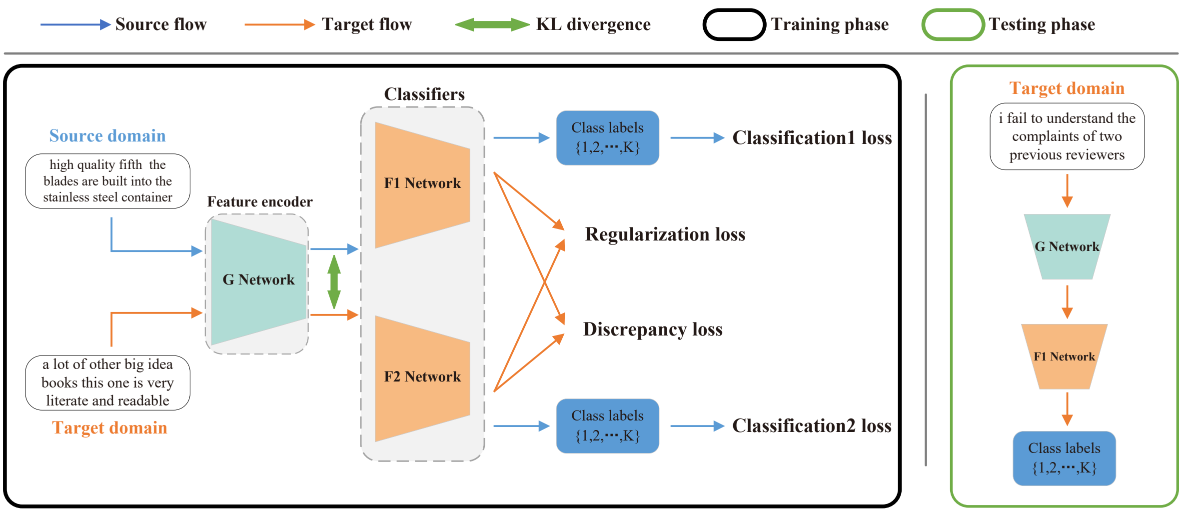


Figure 2: The overview of the proposed adversarial category alignment network in training and test phase.

3 Method

3.1 Problem Definition and Overall Framework

We are given two domains D_s and D_t, denoting the source domain and the target domain respectively. $D_s = \left\{x_i^{(s)}, y_i^{(s)}\right\}_{i=1}^{n_s}$ are n_s labeled source domain examples, where $x_i^{(s)}$ means a sentence and $y_i^{(s)}$ is the corresponding polarity label. $D_t = \left\{x_i^{(t)}\right\}_{i=1}^{n_t}$ are n_t unlabeled target domain examples. In our proposed method, we denote G as a feature encoder that extracts features from the input sentence. Then two classifiers F_1 and F_2 map these features to soft probabilistic outputs $p_1(y|x)$ and $p_2(y|x)$ respectively.

The goal is to train a model to classify the target examples correctly with the aid of source labeled data and target unlabeled data. To achieve this, we first train G, F_1 and F_2 to obtain global marginal alignment. This step reduces the distance between two domains but generates ambiguous target features near the decision boundary. Thus, F_1 and F_2 are adjusted to detect them by maximizing prediction discrepancy. After that, G is trained to generate better features avoiding appearing near the decision boundary. The method also regularizes G by taking the target data samples into consideration. In this way, we can achieve the category alignment. The proposed Adversarial Category Alignment Network (ACAN) is illustrated in Figure 2. The detailed training progress is described in Appendix D.

3.2 Marginal Distribution Alignment

To solve the domain adaption problem, we first consider minimize the classification error on the source labeled data for two classifiers:

$$\begin{aligned} L_{\text{cls}} = &-\frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{j=1}^{K} y_i^{(s)}(j)\log \widetilde{y}_{1i}^{(s)}(j) \\ &-\frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{j=1}^{K} y_i^{(s)}(j)\log \widetilde{y}_{2i}^{(s)}(j) \\ \widetilde{y}_{1i} =& F_1(G(x_i^{(s)})) \quad \widetilde{y}_{2i} = F_2(G(x_i^{(s)})) \end{aligned} \tag{1}$$

where K denotes the number of different polarities. In addition, similar to (Zhuang et al., 2015), our method tries to explicitly minimize the distance between the embedding features from both the source and the target domains. We adopt the Kullback−Leibler (KL) to estimate the distribution divergence:

$$\begin{aligned} L_{\text{kl}} &= \sum_{i=1}^{n} g_s(i)\log\frac{g_s(i)}{g_t(i)} + \sum_{i=1}^{n} g_t(i)\log\frac{g_t(i)}{g_s(i)} \\ g'_s &= \frac{1}{n_s}\sum\nolimits_{i=1}^{n_s} G(x_i^{(s)}), \quad g_s = \frac{g'_s}{||g'_s||_1} \\ g'_t &= \frac{1}{n_t}\sum\nolimits_{i=1}^{n_t} G(x_i^{(t)}), \quad g_t = \frac{g'_t}{||g'_t||_1} \end{aligned} \tag{2}$$

where $g_s, g_t \in \mathcal{R}^D$, $||\cdot||_1$ denotes L1 normalization. In this way, the latent network representations of two domains are encouraged to be similar. In other words, the marginal distribution is forced to be aligned.

3.3 Category-level Alignment

Diverse Views: Considering the marginal distribution alignment, there could be some ambiguous features near the decision boundary, which are easy to be incorrectly categorized into a specific class. If we alter the boundary of classifier F_1 and F_2, the samples closer to the decision boundary would have larger change. To explore these samples, we use F_1 and F_2 to provide diverse guidance. We define a discrepancy between probabilistic outputs of the two classifiers $p_1(y|x)$ and $p_2(y|x)$. The formula is:

$$L_{\text{dis}} = E_{x \sim D_t}[d(p_1(y|x), p_2(y|x))] \tag{3}$$

where $d(p_1(y|x), p_2(y|x))$ defines the average absolute difference for K classes, which is:

$$d(p_1(y|x), p_2(y|x)) = \frac{1}{K} \sum_{i=1}^{K} |p_{1_i}(y|x) - p_{2_i}(y|x)| \tag{4}$$

Specifically, we first fix the generator G and train the classifiers F_1,F_2 to detect points near the decision boundary by maximizing their discrepancy. The objective is as follows:

$$\max_{F1,F2} E_{x \sim D_t}[\frac{1}{K} \sum_{i=1}^{K} |p_{1_i}(y|x) - p_{2_i}(y|x)|] \tag{5}$$

Then, this discrepancy is minimized by optimizing G in order to keep these points away from the decision boundary and categorized into correct classes. The objective is as follows:

$$\min_{G} E_{x \sim D_t}[\frac{1}{K} \sum_{i=1}^{K} |p_{1_i}(y|x) - p_{2_i}(y|x)|] \tag{6}$$

This adversarial step is repeated in the whole training process so that we can continuously locate non-discriminative points and classify them correctly, forcing the model to achieve category-level alignment on two domains.

3.4 Training Steps

The whole training procedure can be divided into three steps. In the first step, we consider both minimizing the classification error and marginal distribution discrepancy to achieve global marginal alignment. The loss function of this step can be written as:

$$L_1 = L_{\text{cls}} + \lambda_1 L_{\text{kl}} \tag{7}$$

In the second step, we consider increasing the difference of two classifiers F_1 and F_2 for the fixed G, thus the ambiguous features can be located by the diverse views. The loss function is defined as below:

$$L_2 = L_{\text{cls}} - \lambda_2 L_{\text{dis}} \tag{8}$$

L_{cls} is used here to ensure the stability of the training process. λ_2 is a hyper-parameter controlling the range of classifiers. In the third step, the difference of two classifiers should be reduced for the fixed F_1 and F_2:

$$L_3 = L_{\text{cls}} + \lambda_3 L_{\text{dis}} \tag{9}$$

L_{cls} and λ_3 used here are similar to the second step. We repeat this step n times to balance the generator and two classifiers. After each step, the corresponding part of the network parameters will be updated. Algorithm 1 describes the overall training procedure.

Algorithm 1 Training procedure of ACAN

```
Require: $D_s, D_t, G, F_1, F_2$
Require: $\lambda_1$, $\lambda_2$, $\lambda_3$, iteration number $n$
  for $i \in [1, max-epochs]$ do
    for minibatch $B^{(s)}$, $B^{(t)} \in D^{(s)}$, $D^{(t)}$ do
      compute $L_{\text{cls}}$ on $\left[x_i \in B^{(s)}, y_i \in B^{(s)}\right]$
      compute $L_{\text{kl}}$ on $\left[x_i \in B^{(s)}, x_j \in B^{(t)}\right]$
      $L_1 = L_{\text{cls}} + \lambda_1 L_{\text{kl}}$
      update $G$, $F_1$, $F_2$ by minimizing $L_1$
      compute $L_{\text{cls}}$ on $\left[x_{i \in B^{(s)}}, y_{i \in B^{(s)}}\right]$
      compute $L_{\text{dis}}$ on $\left[x_{i \in B^{(t)}}, x_{i \in B^{(t)}}\right]$
      $L_2 = L_{\text{cls}} - \lambda_2 L_{\text{dis}}$
      fix $G$, update $F_1$, $F_2$ by minimizing $L_2$.
      for $j \in [1, n]$ do
        compute $L_{cls}$ on $\left[x_i \in B^{(s)}, y_i \in B^{(s)}\right]$
        compute $L_{\text{dis}}$ on $\left[x_i \in B^{(t)}, x_i \in B^{(t)}\right]$
        $L_3 = L_{\text{cls}} + \lambda_3 L_{\text{dis}}$
        fix $F_1$, $F_2$, update $G$ by minimizing $L_3$.
      end for
    end for
  end for
```

3.5 Generator Regularizer

To further enhance the feature generator, we introduce to regularize G with the information of unlabeled target data. Generally, the mapping of $G(\cdot)$ can been seen a low-dimensional feature of the input. According to the manifolds assumption (Chapelle et al., 2009), this feature space is expected to be low-dimensional manifold and linearly separable. Inspired by (Luo et al., 2017), we consider the connections between data points to regularize $G(\cdot)$ in the feature space. Specifically, the regularizer is formulated as follows:

$$R(G) = \sum_{x \in D_t} l_G(x_i, x_j) \tag{10}$$

here l_G is to approximate the semantic similarity of two feature embeddings. Possible options include triplet loss (Wang et al., 2016), Laplacian eigenmaps (Belkin and Niyogi, 2003) etc. After exploring many tricks, we find below is optimal which is also used by (Luo et al., 2017):

$$l_G = \begin{cases} d_{i,j}^2 & s_{ij} = 1 \\ \max(0, m - d_{i,j})^2 & s_{ij} = 0 \end{cases} \tag{11}$$

where $d_{i,j}$ is L2 distance between data points, m is a predefined distance, and s_{ij} indicates whether x_i and x_j belong to the same class or not. Eq. 10 serves as a regularization that encourages the output of $R(G)$ to be distinguishable among classes. It is applied on target data and integrated in the framework in the third training step, weighted by λ_4. During the training, the underlying label of x_i is estimated by taking the maximum posterior probability of the two classifiers.

3.6 Theoretical Analysis

In this subsection, we provide a theoretical analysis of our method, which is inspired by the theory of domain adaptation in (Ben-David et al., 2010).

For each domain, there is a labeling function on inputs X, defined as $f : X \to [0, 1]$. Thus, the source domain is denoted as $\langle D_s, f_s \rangle$ and the target domain as $\langle D_t, f_t \rangle$. We define a hypothesis function h: $X \to [0, 1]$ and a disagreement function:

$$\epsilon(h_1, h_2) = E[|h_1(x) - h_2(x)|] \tag{12}$$

Then the expected error on the source samples $\epsilon_s(h, f)$ of h is defined as:

$$\epsilon_s(h) = \epsilon_s(h, f_s) = E_{x \sim D_s}[|h(x) - f_s(x)|] \tag{13}$$

Also for the target domain, we have

$$\epsilon_t(h) = \epsilon_s(h, f_t) = E_{x \sim D_t}[|h(x) - f_t(x)|] \tag{14}$$

As is introduced in (Ben-David et al., 2010), the probabilistic bound of the error of hypothesis h on the target domain $\epsilon_t(h)$ is defined as:

$$\forall h \in \mathcal{H}, \epsilon_t(h) \leq \epsilon_s(h) + \tfrac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) + \lambda \tag{15}$$

where the expected error $\epsilon_t(h)$ is bounded by three terms: (1) the expected error on the source examples $\epsilon_s(h)$; (2) the divergence between the distributions D_s and D_t; (3) the combined error of the ideal joint hypothesis λ.

First, the training algorithm is easy to minimize $\epsilon_s(h)$ with source label information. Second, λ is expected to be negligibly small and can be usually disregarded. Therefore, the second term $d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t)$ is important quantitatively in computing the target error.

Regarding $d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t)$, we have

$$\begin{aligned} d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) &= 2 \sup_{h, h' \in \mathcal{H}} |\epsilon_s(h, h') - \epsilon_t(h, h')| \\ &= 2 \sup_{h, h' \in \mathcal{H}} |E_{x \sim D_s}[|h(x) - h'(x)|] - E_{x \sim D_t}[|h(x) - h'(x)|]| \end{aligned} \tag{16}$$

where h and h' are two sets of hypotheses in $\mathcal{H}$. As we have sufficient labeled source examples to train, h and h' can have consistent and correct predictions on the source domain data. Thus, $d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t)$ is approximately calculated as $E_{x \sim D_t}[|h(x) - h'(x)|]$. In our model, the hypothesis h can be decomposed into the feature extractor G and the classifier F using the notation $\circ$. Thus $d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t)$ can be formulated as:

$$\sup_{F1, F2} E_{x \sim D_t}[|F_1 \circ G(x) - F_2 \circ G(x)|] \tag{17}$$

For fixed G, sup can be replaced by max. Therefore, F_1 and F_2 are trained to maximize the discrepancy of their outputs and we expect G to minimize this discrepancy. So we obtain

$$\min_{G} \max_{F1, F2} E_{x \sim D_t}[|F_1 \circ G(x) - F_2 \circ G(x)|] \tag{18}$$

The maximization of F_1 and F_2 is to provide diverse views, to find ambiguous points near the decision boundary, and the minimization of G is to keep these points away from the decision boundary. To optimize Eq. 18, we assist the model to capture the whole feature space on the target domain better and achieve lower errors.

4 Experiments

4.1 Data and Experimental Setting

We evaluate the proposed ACAN on the **Amazon reviews** benchmark datasets collected by Blitzer (2007). It contains reviews from four different domains: Books (B), DVDs (D), Electronics (E), Kitchen appliances (K). There are 1000

Source → Target	Previous Work Models					ACAN Models			
	SVM	AuxNN	DANN	PBLM	DAS	Baseline	ACAN-KL	ACAN-KM	ACAN
D → B	75.20	80.80	81.70	82.50	82.05	81.30	**83.00**	82.85	82.35
E → B	68.85	78.00	78.55	71.40	80.00	79.50	**80.30**	79.80	79.75
K → B	70.00	77.85	79.25	74.20	80.05	79.05	79.10	79.60	**80.80**
B → D	77.15	81.75	82.30	**84.20**	82.75	82.50	83.35	83.25	83.45
E → D	69.50	80.65	79.70	75.00	80.15	79.25	81.00	80.80	**81.75**
K → D	71.40	78.90	80.45	79.80	81.40	79.10	80.15	**82.25**	82.10
B → E	72.15	76.40	77.60	77.60	81.15	77.80	78.80	80.85	**81.20**
D → E	71.65	77.55	79.70	79.60	81.55	78.00	81.30	82.75	**82.80**
K → E	79.75	84.05	86.65	**87.10**	85.80	84.35	84.70	86.20	86.60
B → K	73.50	78.10	76.10	82.50	82.25	78.00	77.30	81.00	**83.05**
D → K	72.00	80.05	77.35	**83.20**	81.50	74.65	73.05	77.65	78.60
E → K	82.80	84.15	83.95	**87.80**	84.85	81.05	83.70	83.70	83.35
Average	73.66	79.85	80.29	80.40	81.96	79.55	80.48	81.78	**82.15**

Table 1: Accuracy of adaptation on Amazon benchmark. All results are the averaged performance of each neural model by a 5-fold cross-validation protocol.

positive and 1000 negative reviews for each domain, as well as a few thousand unlabeled examples, of which the positive and negative reviews are balanced. Following the convention of previous works (Zhou et al., 2016a; Ziser and Reichart, 2018; He et al., 2018), we construct 12 cross-domain sentiment classification tasks. In our transferring task, we employ a 5-fold cross-validation protocol, that is, in each fold, 1600 balanced samples are randomly selected from the labeled data for training and the rest 400 for validation. The results we report are the averaged performance of each model across these five folds.

4.2 Training Details and Hyper-parameters

In our implementation, the feature encoder G consists of three parts including a 300-dimensional word embedding layer using GloVe (Pennington et al., 2014), a one-layer CNN with ReLU activation function adopted in (Yu and Jiang, 2016; He et al., 2018) and a max-over-time pooling through which final sentence representation is obtained. Specifically, the convolution filter and the window size of this one-layer CNN are 300 and 3 separately. Similarly, the classifier F_1 and F_2 can be decomposed into one dropout layer and one fully connected output layer. For the fully connected layer, we constrain the l2-norm of the weight vector, setting its max norm to 3. For the implementation of generator regularizer, we apply doubly stochastic sampling approximation due to the computational complexity.

The margin m is set to 1 in this procedure. During training period, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and n are set to 5.0, 0.1, 0.1, 1.5, 2. Similar to (He et al., 2018), we parametrize λ_4 as a dynamic weight $\exp[-5(1-\frac{t}{max-epochs})^2]\lambda_4$. This is to minimize the effort of the regularizer as the predictor is not good at the beginning of training. We train 30 epochs for all our experiments with batch-size 50 and dropout rate 0.5. RMSProp (Tieleman and Hinton, 2012) optimizer with learning rate set to 0.0001 is used for all experiments.

4.3 Methods for Comparison

We consider the following approaches for comparisons (The URLs of previous methods code and data we use are in Appendix A):

SVM (Fan et al., 2008): This is a non-domain-adaptation method, which trains a linear SVM on the raw bag-of-words representation of the labeled source domain.

AuxNN (Yu and Jiang, 2016): This method uses two auxiliary tasks to learn sentence embeddings that works well across two domains. For fair comparison, we replace the neural model in this work with our CNN encoder.

DANN (Ganin et al., 2016): This method exploits a domain classifier to minimize the discrepancy between two domains via adversarial training manner. we replace its encoder with our CNN-based encoder.

PBLM (Ziser and Reichart, 2018): This is a representation learning model that exploits the structure of the input text. Specifically, we choose CNN as the task classifier.

DAS (He et al., 2018): This method employs two regularizations: entropy minimization and self-ensemble bootstrapping to refine the classifier while minimizing the domain divergence.

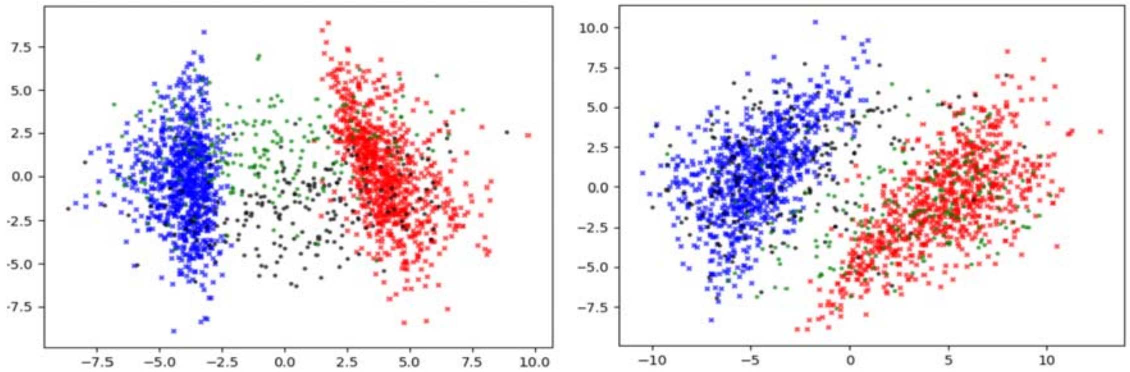


Figure 3: Visualization by applying principal component analysis to the representation of source training data and target testing data produced by ACAN-KL (left) and ACAN (right) for K→E task. The red, blue, green, and black points denote the source positive, source negative, target positive, and target negative examples correspondingly.

Baseline: Our baseline model is a non-adaptive CNN similar to (Kim, 2014), trained without using any target domain information, which is a variant of our model by setting $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ to zeros.
ACAN-KL: ACAN-KL is a variant of our model which minimizes the distance between the features of two domains by minimizing the KL divergence. (set $\lambda_2 = \lambda_3 = \lambda_4 = 0$)
ACAN-KM: ACAN-KM introduces the adversarial category mapping based on ACAN-KL without the regularizer. (set $\lambda_4 = 0$).
ACAN: It is our full model.

4.4 Results

Table 1 shows the classification accuracy of different methods on the Amazon reviews, and we can see that the proposed **ACAN** outperforms all other methods generally. It is obvious to see that **SVM** performs not well in domain transferring task, beaten by **Baseline**. We can notice that exploring the structure of the input text (**AuxNN** and **PBLM**) brings some improvements over **Baseline**. However, these two pivot-based methods present relatively lower ability than **DAS**, which jointly minimizes global feature divergence and refines classifier. Compared to **DAS**, our proposed **ACAN** can improve 0.19% on the average accuracy. This can be explained by that we deal with the relationship between target features distribution and classifier more precisely. Finally, we conduct experiments on the variants of the **ACAN**. It is clear that the performances of **Baseline**, **ACAN-KL**, **ACAN-KM** and **ACAN** present a growing trend in most cases. Compared with **ACAN-KL**, **ACAN** achieves large gain from 80.48% to 82.15%, showing the effectiveness of category-level alignment.

4.5 Case Study

To better understand the results of different models, we conduct experiments on task B → E. For each sentiment polarity, we first extract the most related CNN filters according to the learned weights of the output layer in classifier F_1. Since all listed models use a window size of 3, the outputs of CNN with the highest activation values correspond to the most useful trigrams.

As shown in Table 2, we identify the top trigrams from 10 most related CNN filters on the target domain. It is obvious that **Baseline** and **ACAN-KL** are more likely to capture the domain-independent words, such as “pointless”, “disappointing” and “great”. Thus, the performance of these two models drops much when applied to the target domain. Besides, **DAS** can capture more words of the target domain, but it is limited to nouns with less representativeness, such as ”receiver”, ”product” and etc. Compared to them, **ACAN** is able to extract the domain-specific words like “flawlessly” and “rechargeable”. These results are consistent with the accuracy of each model’s predictions. We also conduct experiments on the tasks **B** → **K** and **K** → **D**. Due to the space limitations, the results are presented in Appendix B.

4.6 Visualization of features

For more intuitive understanding of the differences between the global marginal alignment and category alignment, we further perform a visualization of the feature representations of the **ACAN-KL** and **ACAN** model for the training data in the source domain and the testing data in the target domain for the K→E task. As can be seen in Figure 3, global marginal alignment causes ambiguous

Method	Negative Sentiment	Positive Sentiment
Baseline	**audio-was-distorted**, is-absolutely-pointless, *-very-disappointing, waste-of-money, was-point-most, an-unsupported-config, an-extremely-disappointed, **author-album-etc**, **cure-overnight-headphones**, **aa-rechargable-batteries**	**wep-encryption-detailed**, **totally-wireless-headset**, best-!-i, love-it-!, again-period-!, beautifully-great-price, **awesome-accurate-sound**, beautifully-designed-futuristic, wonderful-product-*, glad-i-purchased
ACAN-KL	totally-useless-method, **audio-was-distorted**, *-very-weak, *-very-disappointing, **extra-ridiculous-buttons**, hopeless-mess-no, now-as-useless, waste-of-cash, is-absolutely-pointless, **manual-is-useless**	gift-i-love, **uniden-cordless-telephone**, a-journey-to, **totally-wireless-headset**, your-own-frequencies, a-gift-excellent, **exceptional-being-rechargeable**, gorgeous-picture-excellent, **with-wireless-security**, beautifully-designed-futuristic
DAS	**receiver-was-faulty**, **defective-product-i**, is-useless-i, do-not-waste, did-not-work, **very-poor-quality**, **the-crappy-keyboard**, just-too-weak, is-absolutely-pointless, **very-stupid-design**	is-an-excellent, **excellent-monitor-with**, is-very-nice, **truly-excellent-headphones**, **an-incredible-sound** **advanced-technology-incredible**, this-is-an, !-highly-recommended show-very-easy, **picture-is-fabulous**
ACAN	**very-poorly-designed**, garbage-im-sorry, **handed-was-defective**, **receiver-was-faulty**, *-very-disappointing, **audio-was-distorted** **dirty-and-scratched**, **extra-ridiculous-buttons**, **cartridges-are-incompatible**, awful-absolutely-horrible	**performs-flawlessly-hours**, a-gift-excellent, beautifully-great-price, **encryption-detailed-monitoring**, **fit-excellent-sound**, !-very-happy, **exceptional-being-rechargeable**, beautifully-designed-futuristic, **smooth-accurate-tracking**, **digital-camera-during**

Table 2: Comparison of the top trigrams chosen from 10 most related CNN filters learned on the task **B** → **E**. The entire table contains the results achieved by the variants of our method. * denotes a padding. The domain-specific words are in **bold**.

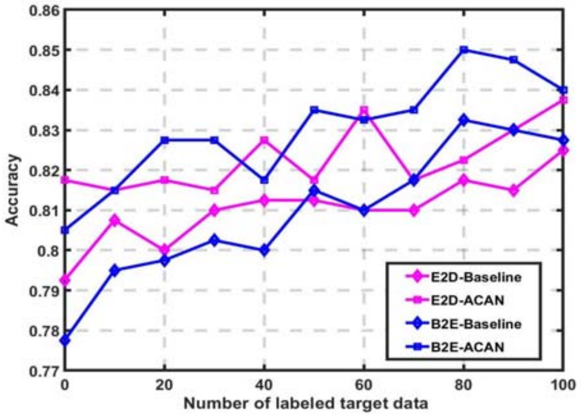


Figure 4: The influence of the number of labeled target data on the task **E** → **D** and **B** → **E**.

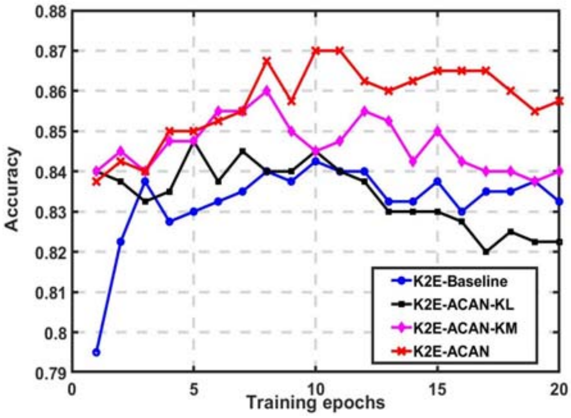


Figure 5: The training process of four ACAN model variants on the task **K** → **E**.

features locating between two clusters while category alignment effectively projects these points into clusters, thus leading a more robust classification result. We also conduct experiments on the tasks **B** → **E** and **B** → **K**. Due to the space limitations, the results are presented in Appendix C.

4.7 Model Analysis

In this part, we provide analysis to our proposed **ACAN** variants. In Figure 4, we show the comparison between **Baseline** and **ACAN** under a setting that some labeled target data are randomly selected and mixed with training data. Here, we present results on two transferring tasks while a similar tendency can be observed in other pairs. With an increase in the number of randomly selected labeled target data, the difference between the two models gradually decreases and **ACAN** also progressively obtains better results. These trends indicate that our **ACAN** is more effective with no or little-labeled target data and can further benefit from more labeled target data. In Figure 5, we can easily observe that **ACAN** continuously shows better results during the whole training process among four settings. After some epochs, **ACAN-KL** starts presenting lower testing accuracy than **Baseline**. One possible reason is that those categories which are initially well aligned between the source and target may be incorrectly mapped because of ignoring category-level feature distribution. This observation can prove our motivation in some degree.

5 Conclusion

In this paper, we propose a novel approach, which utilizes diverse view classifiers to achieve category-level alignment for sentiment analysis. Unlike previous works, we take the decision boundary into consideration, thus classifying the target samples correctly into the corresponding category. Experiments show the proposed ACAN significantly outperforms state-of-the-art methods on the Amazon benchmark. In future we would like to adapt our method to other domain adaptation tasks and consider more effective alternatives for the generator regularizer.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61602197, and in part by the Fundamental Research Funds for the Central Universities, HUST: 2016YXMS085.

References

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1):151–175.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Stephan Gouws, GJ Van Rooyen, MIH Medialab, and Yoshua Bengio. 2012. Learning structural correspondences across different linguistic domains with synchronous neural language models. In *Proc. of the xLite Workshop on Cross-Lingual Technologies, NIPS*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Adaptive semi-supervised learning for cross-domain sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3467–3476. Association for Computational Linguistics.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 123–131. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, Lousiana, USA, February 2–7, 2018*.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. 2017. Smooth neighbors on teacher graphs for semi-supervised learning. *arXiv preprint arXiv:1711.00258*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. 2017. Adversarial dropout regularization. *arXiv preprint arXiv:1711.01575*.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum classifier discrepancy for unsupervised domain adaptation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3723–3732.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *AAAI*.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Jing Wang, Yu Cheng, and Rogério Schmidt Feris. 2016. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. In *CVPR*, pages 2295–2304. IEEE Computer Society.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 538–544. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 236–246. Association for Computational Linguistics.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschlger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (CMD) for domain-invariant representation learning. *arxiv preprint arXiv:1702.08811*.

Kunpeng Zhang, Yu Cheng, Yusheng Xie, Daniel Honbo, Ankit Agrawal, Diana Palsetia, Kathy Lee, Wei-keng Liao, and Alok N. Choudhary. 2011. SES: sentiment elicitation system for social media data. In *ICDM*, pages 129–136. IEEE Computer Society.

Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. 2016a. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 322–332. Association for Computational Linguistics.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016b. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495. The COLING 2016 Organizing Committee.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1241–1251. Association for Computational Linguistics.

A URLs of Data and Code

Here, we provide a list of URLs about the dataset and the code of the previous methods we compare.

- The Amazon product review dataset gathered by Blitzer et al (2007): http://jmcauley.ucsd.edu/data/amazon/
- Code for **AuxNN** (Yu and Jiang, 2016): https://github.com/jefferyYu/Learning-Sentence-Embeddings-for-cross-domain-sentiment-classification
- Code for **DANN** (Ganin et al., 2016): https://github.com/pumpikano/tf-dann
- Code for **PBLM** (Ziser and Reichart, 2018): https://github.com/yftah89/PBLM-Domain-Adaptation
- Code for **DAS** (He et al., 2018): https://github.com/ruidan/DAS

B Trigram Full Results

In the paper, Table 2 shows the top trigrams chosen from 10 most related CNN filters learned on the task **B** → **E** by the DAS the and variants of the ACAN. For a more comprehensive presentation, we also conduct experiments on the task **B** → **K** and **K** → **D**, and the results are listed in Table 3 and Table 4 respectively. It is obvious that the proposed **ACAN** is better to capture domain-specific words, compared to its variants and DAS.

C Visualization full results

In this paper, Figure 3 visualizes the feature representations of the **ACAN-KL** and **ACAN** model for the training data in the source domain and the testing data in the target domain for the K→E task. For a more comprehensive presentation, we also conduct experiments on the task **B** → **E** and **B** → **K**, and the results are listed in Figure 7 and Figure 8 respectively. It is obvious that global marginal alignment causes ambiguous features locating between two clusters while category alignment effectively projects these points into clusters.

D Detailed Illustration of Training Phase

The overview of the propose ACAN is shown in Figure 2. For a better understanding, we present the changes of decision boundaries and data distribution during the network training process, shown in Figure 6. First, we train F_1 and F_2 to locate the points near the decision boundary by maximizing their discrepancy. Then, we train G to minimize the discrepancy to achieve category-level alignment. At the same time, the generator G is regularized with data from target domain.

Method	Negative Sentiment	Positive Sentiment
Baseline	**rice-also-disappointing**, be-such-shoddy, basically-worthless-*, **does-n't-toast**, waste-your-time, is-totally-useless, waste-of-time, was-sorely-disappointed, **safe-stainless-versus**, were-very-dull	**sophisticated-gorgeous-retro**, **lodge-properly-packaged**, **delonghi-cooked-pretty**, **this-stunning-slice**, beautifully-i-highly, perfection-!-i, an-excellent-performer, beautiful-shape-!, **your-cooking-equipment**, *-highly-recommend
ACAN-KL	totally-useless-and, **rice-also-disappointing**, **be-such-shoddy**, **flatware-is-unusable**, misleading-advertising-i, waste-of-time, poorly-made-expensive, were-very-dull, **makes-weak-coffee**, was-sorely-disappointed	dishwasher-nonstick-!, beautifully-get-a, **this-stunning-slice**, !-happy-holidays, look-wonderful-and, beautifully-i-highly, month-i-!, excellent-addition-to, *-highly-recommend, **grilled-meats-and**
DAS	was-very-disappointing, **shoddy-junk-garbage**, totally-useless-and, **thermometer-very-disappointing**, **disappointing-coffee-maker**, **by-flimsy-brittle**, do-not-waste, waste-of-time, very-disappointing-and, **be-lukewarm-disgusting**	**makes-wonderful-tasting**, **this-beautiful-pan**, is-an-excellent, **sophisticated-gorgeous-retro**, is-highly-recommend, awesome-!-!, **makes-great-coffee**, **and-versatile-pan**, also-highly-recommend, it-is-great
ACAN	**kettle-was-leaking**, totally-useless-and, **rice-also-disappointing**, **flatware-is-unusable**, was-sorely-disappointed, waste-of-money **flat-crooked-ugly**, **is-no-metal**, now-basically-worthless, **makes-weak-coffee**	**sophisticated-gorgeous-retro**,**great-hot-drinks**, it-a-learning, !-happy-holidays, **great-grilled-sandwiches**, !-highly-recommend, **it-toasts-beautifully**, look-wonderful-and, excellent-addition-to, **nonstick-!-you**

Table 3: Comparison of the top trigrams chosen from 10 most related CNN filters learned on the task **B** → **K**. The entire table contains the results achieved by the variants of our method. * denotes a padding. The domain-specific words are in **bold**.

Method	Negative Sentiment	Positive Sentiment
Baseline	beyond-is-badly, **such-gross-audio**, returning-for-the, poorly-executed-poorly, **director-john-ford**, is-so-disappointing, does-not-work, is-a-disappointment, **lighting-poor-directing**, **pathetic-remake-from**	**hip-hop-dvd**, **combines-multiple-genres**, the-most-amazing, very-good-price, **stylish-photography**, **best-performances-since**, amazing-!-the, lasting-and-unique, **accomplished-and-dedicated**, loves-being-able
ACAN-KL	return-an-even, **such-gross-audio**, **star-hollywood-material**, poorly-executed-poorly, does-not-work, a-complete-failure, is-a-disappointment, **pathetic-remake-from**, waste-of-money, **failed-miserably-alyson**	**beautifully-classic-comedy**, adult-who-enjoys, i-bought-loves, the-most-acclaimed, **combines-multiple-genres**, very-good-price, **'s-memorable-entrance**, great-and-splendid, **acclaimed-romantic-comedies**, **stylish-photography-and**
DAS	**release-the-movie**, a-total-waste, **dodging-bullets-and**, **incompetent-direction-by**, is-absolutely-horrible, very-disappointing-once, **is-pretty-pathetic**, was-a-waste **awful-the-ending**, **pathetic-remake-from**	**entertaining-and-inspirational**, truly-enjoy-it, **an-amazing-artist**, **superb-production-and**, very-good-price, **fantastic-film-!**, **best-performance-since**, perfect-love-on, **amazing-film-from**, **and-fascinating-documentaries**
ACAN	**unfunny-overrated-movie**, **of-gross-caricature**, was-a-waste, **pathetic-remake-from**, poorly-executed-poorly, was-awful-from, **directing-poor-writing**, is-a-disappointment, **disgusting-badly-written**, **tasteless-unoriginal-drivel**	**combines-multiple-genres**, very-good-price, are-great-featuring, **accomplished-and-dedicated**, **great-performance-tongue**, **'s-stylish-photography**, is-my-favourite, **family-classics-action** the-most-amazing, **fantastic-action-picture**

Table 4: Comparison of the top trigrams chosen from 10 most related CNN filters learned on the task **K** → **D**. The entire table contains the results achieved by the variants of our method. * denotes a padding. The domain-specific words are in **bold**.

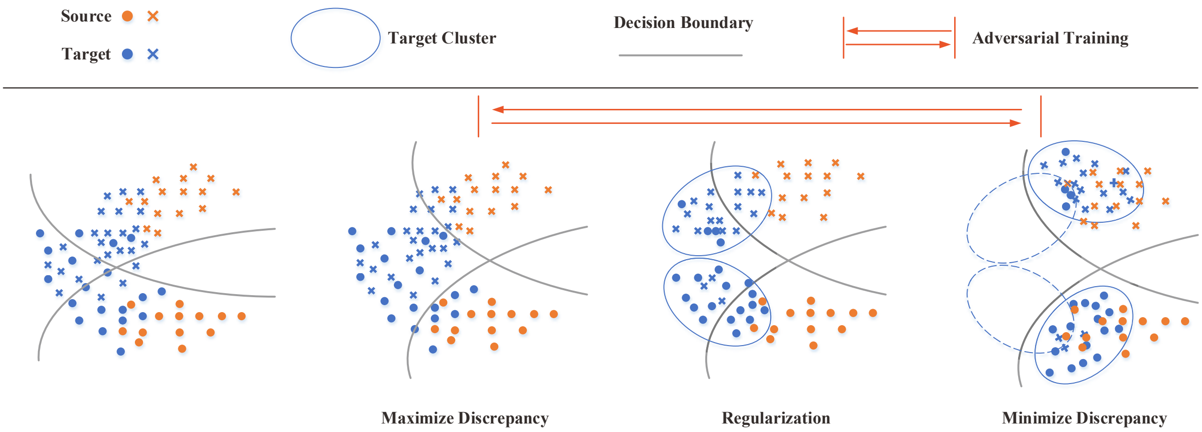


Figure 6: The detail of changes in decision boundaries and data distribution during the network training process.

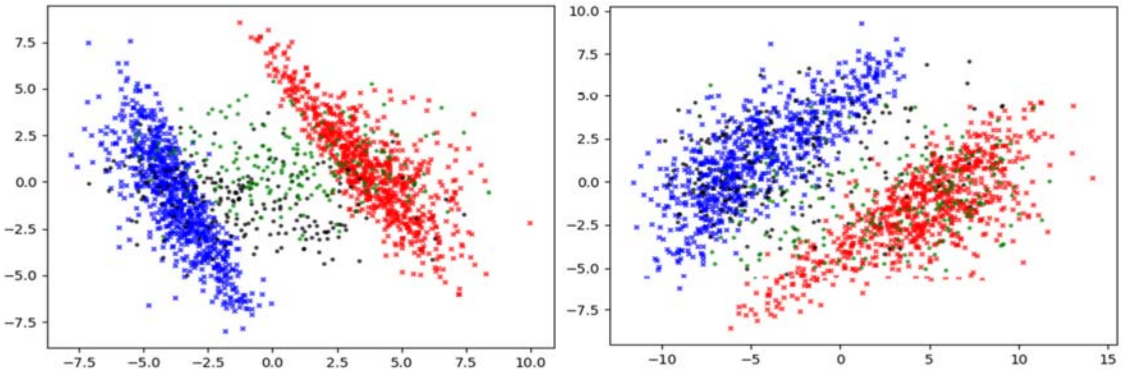


Figure 7: Visualization by applying principal component analysis to the representation of source training data and target testing data produced by ACAN-KL (left) and ACAN (right) for B→E task. The red, blue, green, and black points denote the source positive, source negative, target positive, and target negative examples correspondingly.

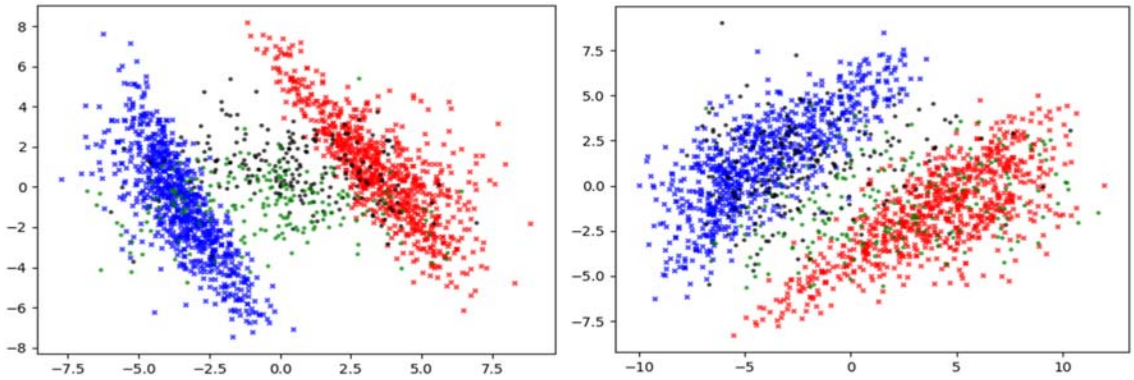


Figure 8: Visualization by applying principal component analysis to the representation of source training data and target testing data produced by ACAN-KL (left) and ACAN (right) for B→K task. The red, blue, green, and black points denote the source positive, source negative, target positive, and target negative examples correspondingly.

Target-oriented Opinion Words Extraction with Target-fused Neural Sequence Labeling

Zhifang Fan, Zhen Wu, Xin-Yu Dai,[*] Shujian Huang, Jiajun Chen
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, 210023, China
{fanzf,wuz}@nlp.nju.edu.cn;{daixinyu,huangsj,chenjj}@nju.edu.cn

Abstract

Opinion target extraction and opinion words extraction are two fundamental subtasks in Aspect Based Sentiment Analysis (ABSA). Recently, many methods have made progress on these two tasks. However, few works aim at extracting opinion targets and opinion words as pairs. In this paper, we propose a novel sequence labeling subtask for ABSA named TOWE (Target-oriented Opinion Words Extraction), which aims at extracting the corresponding opinion words for a given opinion target. A target-fused sequence labeling neural network model is designed to perform this task. The opinion target information is well encoded into context by an Inward-Outward LSTM. Then left and right contexts of the opinion target and the global context are combined to find the corresponding opinion words. We build four datasets for TOWE based on several popular ABSA benchmarks from laptop and restaurant reviews. The experimental results show that our proposed model outperforms the other compared methods significantly. We believe that our work may not only be helpful for downstream sentiment analysis task, but can also be used for pair-wise opinion summarization.

1 Introduction

Sentiment analysis, also known as opinion mining (Pang and Lee, 2007; Liu, 2012), has drawn increasing attention of researchers and industries in recent years. It can provide valuable information from user-generated reviews. However, sentiment analysis at sentence level or document level sometimes cannot provide more detailed information, thus a finer-grained task, Aspect-Based Sentiment Analysis (ABSA) (Pontiki et al., 2014), is proposed to identify the opinions of a specific target or aspect in reviews. ABSA consists of multiple subtasks including aspect category detection, opinion target extraction, aspect level sentiment classification etc. Opinion target extraction (OTE) and opinion words extraction (OWE) are two such fundamental subtasks. Opinion targets, sometimes called aspect terms, are the words or phrases in the sentence representing features or entities towards which users show attitude. Opinion words (or opinion terms) refer to those terms used to express attitude explicitly. For example, in the sentence "*The menu is limited but almost all of the dishes are excellent.*", the words "*menu*" and "*dishes*" are two opinion targets, and the words "*limited*" and "*excellent*" are opinion words. More examples can be found in the upper part of Figure 1.

Review:
"My friends and I were on vacation in NY and was referred to Chance by a friend. I found the food to be outstanding, particularly the salmon dish I had. I also ordered the Change Mojito, which was out of this world. My friends settled for rice dishes, but we came back the following day to try the dim sum, which was not outstanding, but good. We ate out in the back patio, which is worth it as it's cool and the music is hear well there. Overall, excellent restaurant!"

The list of extracted targets and opinion words as pairs:

- food : [outstanding]
- salmon dish : [outstanding]
- Change Mojito : [out of this world]
- dim sum : [not outstanding, good]
- back patio : [worth, cool]
- music : [hear well]
- restaurant : [excellent]

Figure 1: The upper part is a restaurant review and the lower part shows the pairs of extracted opinion targets (in red) and opinion words (in blue).

Recently, a great number of works based on neural networks have been done on these two subtasks (Liu et al., 2015; Poria et al., 2016; Xu et al., 2018). Furthermore, some works also integrate the two subtasks into a multi-task learning architecture to extract them jointly, which achieves great progress on both subtasks (Wang et al., 2016, 2017;

---

[*] Corresponding author.

Proceedings of NAACL-HLT 2019, pages 2509–2518
Minneapolis, Minnesota, June 2 - June 7, 2019.

Li and Lam, 2017). However, the extracted opinion targets and opinion words are not in pairs and the correspondence is not extracted. For instance, in the example sentence, ⟨*menu:limited*⟩ and ⟨*dishes:excellent*⟩ are two opinion pairs. Obviously, extracting them as pairs is significant for ABSA. Additionally, in Figure 1, the list of pairs extracted from the example review can be considered to be an extractive pair-wise opinion summarization.

Considering the significance of the pairs in reviews and promising results of targets extraction in previous works, in this paper, we propose a new subtask for ABSA named **TOWE** (Target-oriented Opinion Words Extraction). Given a review and a target in the review, the objective of TOWE is to extract the corresponding opinion words describing or evaluating the target from the review. Then, TOWE can form pairs of the given target and its corresponding opinion words.

Motivated by the success of neural networks in natural language processing, we design a powerful sequence labeling neural network model to perform TOWE. The task TOWE aims to extract the target-oriented opinion terms. In the same review, for different targets, the model needs to output different results. Therefore, a core challenge is the learning of target-specific context representations. We design a neural encoder to incorporate target information and generate the target-fused context. To be specific, we propose an Inward-Outward LSTM to pass target information to the left context and the right context of the target respectively. Then we combine the left, right and global context to encode the sentence and make sequence labeling. It is essential and reasonable to formulate TOWE as a sequence labeling task because some opinion terms include several words and one opinion target may correspond to multiple opinion terms. We try two different decoding strategies in the experiment.

Our main contributions are summarized as follows:

- We propose a sequence labeling subtask for ABSA: TOWE (Target-oriented Opinion Words Extraction), which can offer assistance and interpretability for downstream tasks in ABSA.

- We design a novel sequence labeling neural network model to perform TOWE. It can generate target-specific context representations for different targets in the same review.

- We build four datasets from different domains serving as a benchmark for future works. We conduct extensive experiments on these datasets, and the results show that our model could significantly exceed a variety of baselines.

We release the datasets and our source code at `https://github.com/NJUNLP/TOWE`

2 Related works

A lot of works have been carried out for Opinion Targets Extraction. Traditional methods can be categorized into unsupervised/semi-supervised methods (Hu and Liu, 2004; Zhuang et al., 2006; Qiu et al., 2011) and supervised methods (Jakob and Gurevych, 2010; Shu et al., 2017). Recently, deep learning methods have also made progress in this task. Liu et al. (2015) apply a recurrent neural network with pre-trained word emebddings to solve this task. Yin et al. (2016) exploit a CRF with dependency-paths enhanced word embeddings for aspect term extraction. Poria et al. (2016) use a deep convolutional neural network (CNN) and Xu et al. (2018) propose a CNN model with double embeddings.

Some works extract the targets and opinion words jointly as a co-extraction strategy. Qiu et al. (2011) propose double propagation to expand opinion targets and opinion words lists in a bootstrapping way. Liu et al.(2013) extract the targets and opinion words jointly with modeling the relation from a statistical word alignment model. This co-extraction strategy can also be adopted in neural networks with multi-task learning (Wang et al., 2016, 2017; Li and Lam, 2017). However, in all these works, the extracted targets and opinion words are separated.

In the literature, only a few works discussed opinion pairs. Hu and Liu (2004) use the distance information and recognize the nearest adjective of target as the opinion words. Zhuang et al. (2006) utilize lexicons and human-built word lists to extract the targets and opinion words in the corpus, and then identify valid feature-opinion pairs with syntactic rule templates based on dependency parsing trees. However, these two methods heavily depend on the external resources such as parsers or lexicons and the performance of these approaches relies on the quality of parsing result. By contrast, our model is a purely data-driven supervised learning method and does not need any external linguistic knowledge, lexicons or handcrafted templates. Moreover, in these two methods, the process of detecting opinion words and the process of discovering correspondence is separated into two tasks, which suffers from error propagation. Our model for TOWE aims at detecting the corresponding opinion words in one step with sequence labeling.

3 Our Methods

3.1 Task Formulation

Given a sentence $s = \{w_1, w_2, \dots, w_i, \dots, w_n\}$ consisting of n words, and a opinion target t in the sentence, the task is to make sequence labelling on the sentence to extract the target-oriented opinion words. We use the BIO tagging scheme (Ramshaw and Marcus, 1995) on this task. For each word w_i in the sentence s, it should be tagged as $y_i \in \{B, I, O\}$ (B: Beginning, I: Inside, O: Others).

For example, for different opinion targets, the sentence "*Waiters are very friendly and the pasta is out of this world .*" is tagged in w_i/y_i style as follows:

1. *Waiters/O are/O very/O* [*friendly/B*] *and/O the/O pasta/O is/O out/O of/O this/O world/O ./O* (Given opinion target: ***waiter***, extract ***"friendly"*** as corresponding opinion word).

2. *Waiters/O are/O very/O friendly/O and/O the/O pasta/O is/O* [*out/B of/I this/I world/I*] *./O* (Given Opinion target: ***pasta***, extract ***"out of this world"*** as corresponding opinion words).

3.2 Framework

Figure 2 shows the framework of our methods, which follows an encoder-decoder architecture. We propose a target-fused encoder to incorporate the target information into context and learn target-specific context representations, then pass them to the decoder for sequence labeling. In the target-fused encoder, we first use an Inward-Outward LSTM to model the left context and right context of the target, then combine them with the global context. In the decoder, we can adopt two different decoding strategies. We present the details of each component in the following sections.

3.3 Target-Fused Encoder

We first generate the input vectors for each word by using a embedding lookup table $\mathbb{L} \in \mathbb{R}^{d \times |V|}$, where d is the embedding dimension and $|V|$ is the vocabulary size. The embedding lookup table will map $s = \{w_1, w_2, \dots, w_t, \dots, w_n\}$ to a sequence of vectors $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \dots, \mathbf{e}_n\}$ as words representations where $\mathbf{e}_i \in \mathbb{R}^d$.

Typically, neural sequence labeling models use recurrent neural networks, such as LSTM (Hochreiter and Schmidhuber, 1997) or BiLSTM, to model the sentence. However, merely using BiLSTM to model the whole sentence is totally target-independent. For the different target terms in the same sentence, BiLSTM outputs equal representation and cannot generate target-specific results.

As mentioned before, the core challenge of TOWE is the learning of target-specific context representations. It is evident that different targets have different positions in the sentence and thus different contexts. So, we first split the sentence into three segments: left context $\{w_1, w_2, \cdots, w_l\}$, target term $\{w_{l+1}, \cdots, w_{r-1}\}$ and right context $\{w_r, \cdots, w_n\}$ and left and right contexts are target-specific.We use a left LSTM to model the left context plus target and a right LSTM to model the target plus right context respectively. In this way the target-specific contexts could generate target-specific context representations. However, the direction of the two LSTMs is a crucial problem.

3.3.1 Inward-LSTM

We can use a simple strategy called Inward-LSTM, which follows the design of TD-LSTM (Tang et al., 2016). As Figure 2 shows, Inward-LSTM runs the two LSTMs from the two ends of the sentence to the middle target respectively. It runs the left LSTM from the first word to opinion target as a forward-LSTM and a right LSTM from the last word to the opinion target as a backward-LSTM, so we call it as Inward. This is a process of passing the context to target. We obtain left context representations $\mathbf{H}^{\mathrm{L}}$ and right context representations $\mathbf{H}^{\mathrm{R}}$ as follows:

$$\mathbf{h}_i^{\mathrm{L}} = \overrightarrow{\mathrm{LSTM}}(\mathbf{h}_{i-1}^{\mathrm{L}}, \mathbf{e}_i), \forall i \in [1, \cdots, r-1]\,, \quad (1)$$

$$\mathbf{h}_i^{\mathrm{R}} = \overleftarrow{\mathrm{LSTM}}(\mathbf{h}_{i+1}^{\mathrm{R}}, \mathbf{e}_i), \forall i \in [l+1, \cdots, n]\,. \quad (2)$$

It is obvious that the words of opinion target $\{w_{l+1}, \cdots, w_{r-1}\}$ are represented twice in the left LSTM and right LSTM. We simply average the two representations for the same word to get the representation of target words:

$$\mathbf{h}_i^{\mathrm{LR}} = \frac{(\mathbf{h}_i^{\mathrm{L}} + \mathbf{h}_i^{\mathrm{R}})}{2}, \forall i \in [l+1, \cdots, r-1]\,. \quad (3)$$

Then the context representation is: $\mathbf{H}^{\mathrm{I}} = \{\mathbf{h}_1^{\mathrm{L}}, \cdots, \mathbf{h}_l^{\mathrm{L}}, \mathbf{h}_{l+1}^{\mathrm{LR}}, \cdots, \mathbf{h}_{r-1}^{\mathrm{LR}}, \mathbf{h}_r^{\mathrm{R}}, \cdots, \mathbf{h}_n^{\mathrm{R}}\}$.

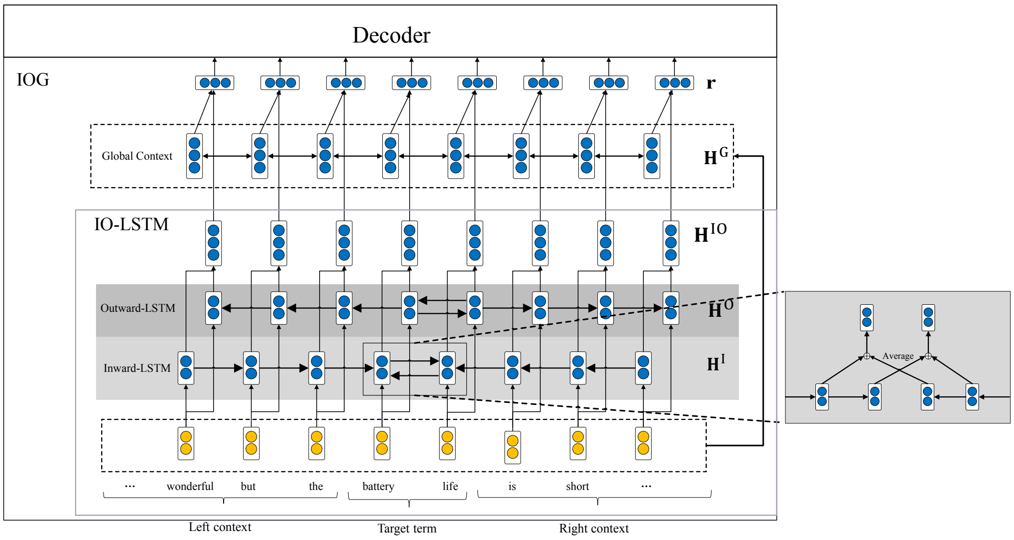


Figure 2: The framework of our method. The subfigure on the right side is an illustration for averaging the hidden states of the Opinion Target.

3.3.2 Outward-LSTM

Although passing contexts to the target in Inward-LSTM is a good strategy for encoding whole sentence representation, only using this strategy is not enough for TOWE because the target information is not passed to the left and right context. For example, in the sentence *"I found the food to be outstanding."*, the opinion target is *"food"*, the Inward-LSTM will first model *"outstanding"* and then model *"food"*. The representation of *"outstanding"* does not contain the information of *"food"*.

To solve this problem, we design a novel strategy specifically for TOWE, i.e., Outward-LSTM. The idea of the Outward-LSTM is to pass the target to the context, which we believe is a better choice. As Figure 2 shows, the Outward-LSTM starts two LSTMs from the target in the middle and run towards the both ends of the sentence, which means the left LSTM is a backward LSTM and the right LSTM is a forward LSTM. We average the duplicate target hidden states and get the target-fused context representations $\mathbf{H}^{\mathrm{O}} = \{\mathbf{h}_1^{\mathrm{L}}, \cdots, \mathbf{h}_l^{\mathrm{L}}, \mathbf{h}_{l+1}^{\mathrm{LR}}, \cdots, \mathbf{h}_{r-1}^{\mathrm{LR}}, \mathbf{h}_r^{\mathrm{R}}, \cdots, \mathbf{h}_n^{\mathrm{R}}\}$:

$$\mathbf{h}_i^{\mathrm{L}} = \overleftarrow{\mathrm{LSTM}}(\mathbf{h}_{i+1}^{\mathrm{L}}, \mathbf{e}_i), \forall i \in [1, \cdots, r-1], \tag{4}$$

$$\mathbf{h}_i^{\mathrm{R}} = \overrightarrow{\mathrm{LSTM}}(\mathbf{h}_{i-1}^{\mathrm{R}}, \mathbf{e}_i), \forall i \in [l+1, \cdots, n], \tag{5}$$

$$\mathbf{h}_i^{\mathrm{LR}} = \frac{(\mathbf{h}_i^{\mathrm{L}} + \mathbf{h}_i^{\mathrm{R}})}{2}, \forall i \in [l+1, \cdots, r-1]. \tag{6}$$

This concise and reasonable strategy can solve the problems remaining in the Inward-LSTM. As we start the LSTM from the target, the target's information is fused into each word in the sentence. Also, the Outward-LSTM ensures that for different targets each word has different representations. Take the sentence *"Its camera is wonderful but the battery life is short !"* as an example. For target *"camera"* or *"battery life"*, the target-fused representations for *"short"* are different and can generate target-specific results.

3.3.3 IO-LSTM

We can combine the both strategy and adopt an Inward-Outward LSTM (IO-LSTM). IO-LSTM concatenates the outputs of Outward-LSTM and Inward-LSTM. The output of Outward-LSTM is crucial for incorporating target information into context, while the Inwards-LSTM is included so they can complement each other and act as a Target-specific Bidirectional LSTM. The target-fused context representations are denoted as $\mathbf{H}^{\mathrm{IO}}$:

$$\mathbf{h}_i^{\mathrm{IO}} = \left[\mathbf{h}_i^{\mathrm{I}}; \mathbf{h}_i^{\mathrm{O}}\right]. \tag{7}$$

3.3.4 IOG: IO-LSTM + Global context

To extract the target-oriented opinion words, only considering the context of each side in isolation is not enough. The left context and right context in the IO-LSTM are separated, and the left LSTM and right LSTM only share the opinion target. It is important to understand the global meaning of the whole sentence while detecting the opinion words on the left and right context. So we introduce the global context to further improve the IO-LSTM. We use a BiLSTM to model the whole sentence embeddings $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \ldots, \mathbf{e}_n\}$ and obtain global contextualized representation $\mathbf{H}^{\mathrm{G}}$ as follows:

$$\mathbf{h}_i^{\mathrm{G}} = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right], \tag{8}$$

$$\overrightarrow{\mathbf{h}}_i = \mathrm{LSTM}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{e}_i), \tag{9}$$

$$\overleftarrow{\mathbf{h}}_i = \mathrm{LSTM}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{e}_i). \tag{10}$$

Then we combine left-right contexts from IO-LSTM and global context, as shown in Figure 2. This enables us to obtain the final target-specific contextualized representation $\mathbf{r}$ for each word:

$$\mathbf{r}_i = \left[\mathbf{h}_i^{\mathrm{IO}}; \mathbf{h}_i^{\mathrm{G}}\right]. \tag{11}$$

The final representation $\mathbf{r}$ is fused with both target information and global context information, which can be passed to the decoder for sequence labeling.

3.4 Decoder and Training

Given a sequential representation $\mathbf{r}$, we can use $\mathbf{r}$ to compute $p(\mathbf{y}|\mathbf{r})$ where $\mathbf{y} = \{y_1, \cdots, y_n\}$ are BIO-label sequence for the sentence and $y_i \in \{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$. Two different decoding policies can be adopted.

3.4.1 Greedy decoding

The first is greedy decoding, formulated as a three-class classification problem at each position independently. We use softmax to compute the probability:

$$p(y_i|\mathbf{r}_i) = \mathrm{softmax}\left(\mathbf{W}_s\mathbf{r}_i + \mathbf{b}_s\right). \tag{12}$$

Greedy decoding just simply selects the tag with highest point-wise probability. It does not consider the dependencies between tags but runs faster. We use the negative log likelihood (NLL) as the loss for one sentence:

$$L(s) = -\sum_{i=1}^{n}\sum_{k=1}^{3}\mathbb{I}(y_i = k)\log p(y_i = k|w_i). \tag{13}$$

3.4.2 CRF

The second decoding method is to use Conditional Random Field (CRF) (Lafferty et al., 2001). CRF considers the correlations between tags in neighborhoods and score the whole sequence of tags. Specifically, we use a linear-chain CRF and score the tag sequence as conditional probability:

$$p(\mathbf{y}|\mathbf{r}) = \frac{\exp(s(\mathbf{r}, \mathbf{y}))}{\sum_{\mathbf{y}' \in Y}\exp(s(\mathbf{r}, \mathbf{y}'))}, \tag{14}$$

where Y is the set of all possible tag sequences and $s(\mathbf{r}, \mathbf{y})) = \sum_i^n \left(\mathbf{A}_{y_{i-1}, y_i} + \mathbf{P}_{i, y_i}\right)$ is the score function. $\mathbf{A}_{y_{i-1}, y_i}$ measures the transition score from y_{i-1} to y_i and $\mathbf{P}_i = \mathbf{W}_s\mathbf{r}_i + \mathbf{b}_s$. So we use negative log likehood as the loss of the sentence:

$$L(s) = -\log p(\mathbf{y}|\mathbf{r}). \tag{15}$$

When given a new sentence for decoding, we will output the tag sequence that maximizes the conditional probability with Viterbi algorithm.

Finally, we minimize the loss for training:

$$J(\theta) = \sum_{s}^{|D|} L(s). \tag{16}$$

4 Experiments

4.1 Datasets

We build the datasets based on the SemEval challenge 2014 Task4, SemEval Challenge 2015 task 12 and SemEval Challenge 2016 task 5 (Pontiki et al., 2014, 2015, 2016). The SemEval challenge provides several datasets from restaurant and laptop domain. These datasets are very popular benchmarks for many ABSA subtasks, including Aspect category detection, Opinion Target Extraction, Opinion Words Extraction and Target-Dependent Sentiment Analysis (TDSA).

In the original datasets of SemEval challenge, the opinion targets (aspect terms) are annotated, but the opinion words and the correspondence with targets are not provided. So we annotate the corresponding opinion words for the annotated targets. Every sentence is annotated by two people, and the conflicts will be checked. Each instance of the datasets consists of a sentence, the position of the target and the positions of the corresponding opinion words. Note that we only keep the sentences that contain pairs of target and opinion words. The sentences without targets or with implicit opinion expressions are not included.

Finally, we generate four datasets: **14res** and **14lap** from SemEval 2014, **15res** from SemEval 2015 and **16res** from SemEval 2016. **14res**, **15res**, and **16res** contain reviews from restaurant domain. The sentences in **14lap** come from laptop domain. The statistics of the four datasets is shown in Table 1.

4.2 Settings

In our experiments, we initialize word embedding vectors with 300-dimension GloVe vectors which

Dataset		#sentences	#targets
14res	Training	1627	2643
	Testing	500	865
14lap	Training	1158	1634
	Testing	343	482
15res	Training	754	1076
	Testing	325	436
16res	Training	1079	1512
	Testing	329	457

Table 1: Statistics of datasets. The number of targets is identical to the number of pairs and instances

are pre-trained on unlabeled data of 840 billion tokens (Pennington et al., 2014). The word embeddings are fixed and not fine-tuned during the training stage. The dimension of hidden states in all the LSTM cell is set as 200. Adam (Kingma and Ba, 2015) is chosen as the optimization method with the default setting in the original paper. We randomly split 20% of the train set as dev set for tunning the hyperparameters and early stopping. Then we test the models on testing sets and the average result of five runs is reported.

4.3 Evaluation Metrics

Precision, recall and F1 score are used as the metrics to measure the performance of models. An extracted opinion words span is regarded as a correct prediction when the starting and ending offset of the predicted span are both identical to those of a golden opinion words span. We compute Precision, Recall and F1 with the span as the unit.

4.4 Compared Methods

Since we are the first to study this sequence labeling task, there is no available sequence labeling model in the literature to be compared. Although there are a number of complicated models in TDSA, the task is different. Those TDSA models focus on sentence-level representations for sentiment classification, while for TOWE the representation on token-level representations is more crucial. Simply transferring the TDSA models for TOWE is not suitable.

Except for two rule-based methods, we can only design and implement the baselines for TOWE ourselves. Our final model is the IOG encoder with a greedy decoding strategy. We compare it with the following baselines:

- **Distance-rule**: Hu and Liu(2004) use the distance and POS tags to determine the opinion words. Following this idea, we first use the nltk toolkit to make part-of-speech tagging on each word and select the nearest adjective from the target as the corresponding opinion word.
- **Dependecy-rule**: We adopt the strategies proposed in (Zhuang et al., 2006) which uses dependency-tree based templates to identify opinion pairs. The POS tag of opinion targets and opinion words and the dependency path between them in the training set are recorded as rule templates.[1] The high-frequency dependency templates are used for detecting the related opinion words in the testing set.
- **LSTM/BiLSTM**: This method is an LSTM/BiLSTM network built on top of word embeddings proposed by (Liu et al., 2015). We pass the whole sentence into the LSTM/BiLSTM and each hidden state is fed to a softmax layer for three-class classification, which works as sentence-level opinion words extraction.
- **Pipeline**: This method combines BiLSTM and Distance-rule method in a pipelined way. We first train a sentence-level opinion words extraction model with BiLSTM and extract all the opinion words in the test sentences; then we select the closest extracted opinion words of the target as the result.
- **Target-Concatenated BiLSTM (TC-BiLSTM)**: This method incorporates the target information into sentence by concatenation. A target vector is obtained by the average pooling of target word embeddings. The word representation at each position is the concatenation of word embedding and target vector, which is then fed into a BiLSTM for sequence labeling.

4.5 Results and Discussion

The main results can be found in Table 2. Note that all the neural models in Table 2 adopt greedy decoding. The performance of Distance-rule method is not satisfactory and the worst among all the methods; its recall score is especially low. IOG obtains an F1 score with a greater-than 30% improvement over the Distance-rule method. Dependency-rule method obtains a general improvement than Distance-rule, but it was still lower than the below sequence-labeling based methods. This reveals the

---

[1] We use the parsers in spaCy: https://spacy.io

Models	14res			14lap			15res			16res		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Distance-rule	58.39	43.59	49.92	50.13	33.86	40.42	54.12	39.96	45.97	61.90	44.57	51.83
Dependency-rule	64.57	52.72	58.04	45.09	31.57	37.14	65.49	48.88	55.98	76.03	56.19	64.62
Pipeline	77.72	62.33	69.18	72.58	56.97	63.83	74.75	60.65	66.97	81.46	67.81	74.01
LSTM	52.64	65.47	58.34	55.71	57.53	56.52	57.27	60.69	58.93	62.46	68.72	65.33
BiLSTM	58.34	61.73	59.95	64.52	61.45	62.71	60.46	63.65	62.00	68.68	70.51	69.57
TC-BiLSTM	67.65	67.67	67.61	62.45	60.14	61.21	66.06	60.16	62.94	73.46	72.88	73.10
IOG	**82.85**	**77.38**	**80.02**	**73.24**	**69.63**	**71.35**	**76.06**	**70.71**	**73.25**	**85.25**	**78.51**	**81.69**

Table 2: Main Results in terms of Precsion, Recall and F1-score. Best results are in bold. IOG outperms all the baselines significantly ($p < 0.01$).

Models	14res			14lap			15res			16res		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Inward-LSTM	53.64	70.06	60.67	55.49	54.96	55.18	53.84	68.60	60.32	66.46	73.83	69.92
Outward-LSTM	81.08	73.65	77.17	70.34	64.90	67.48	74.18	65.60	69.62	81.12	77.68	79.35
IO-LSTM	82.09	75.44	78.62	72.90	66.49	69.51	74.71	67.22	70.74	83.31	78.67	80.91
IOG	82.85	77.38	80.02	73.24	69.63	71.35	76.06	70.71	73.25	85.25	78.51	81.69
IOG + CRF	82.97	77.73	80.24	74.19	68.96	71.39	75.50	71.68	73.51	84.41	79.43	81.84

Table 3: Comparisions for different model design in terms of Precsion, Recall and F1-score.

lack of robustness in rule-based approaches. The error propagation from syntactic parsers is also a reason for poor performance.

The Pipeline model performs much better than rule-based methods, obtaining an especially high precision, showing that machine-learning methods can obtain better opinion words extraction. However, pipeline model is still not ideal, and the F1-score is approximately 10% lower than our proposed model in several datasets. This reflects that the distance information is not sufficient for detecting target-oriented opinion words while IOG could better handle long distance dependency problem. Also, this strategy cannot solve the cases where one target corresponds to more than one opinion term. It also suffers from error propagation.

LSTM and BiLSTM are both target-independent leading to low precision, and their performance is even worse than pipeline method. IOG outperforms BiLSTM by about 15% averagely, which indicates the target information should be included.

TC-BiLSTM includes the target information by concatenation and obtains better general performance than LSTM and BiLSTM. However, TC-BiLSTM is still over 10% lower than IOG and is slightly inferior to Pipeline, showing that the concatenation is not a good way to incorporate the target information for TOWE. We believe that the problem is that the concatenated target may interfere with the other targets in the same sentence.

IOG achieves the best performance on all datasets from different domains compared to both the rule-based methods and neural models. We can conclude that IOG can learn target-specific representations more effectively and can better capture the correspondence between targets and opinion words.

4.6 Model Analysis

To compare the different design of our model and provide more compared models, we also report the results of the variants of our models in Table 3.

- **Inward-LSTM**: $\mathbf{H}^{\mathrm{I}}$ computed from (1), (2), (3) are fed to the greedy decoder for sequence labeling.
- **Outward-LSTM**: $\mathbf{H}^{\mathrm{O}}$ computed from (4), (5), (6) are fed for greedy decoding
- **IO-LSTM**: Combining Inwards-LSTM and Outwards-LSTM, $\mathbf{H}^{\mathrm{IO}}$ is obtained by concatenation of $\mathbf{H}^{\mathrm{I}}$ and $\mathbf{H}^{\mathrm{O}}$ in (7), which is then used for greedy decoding.
- **IOG+CRF**: Passing the representations $\mathbf{r}$ in IOG to a CRF decoder.

The performance of Inward-LSTM is inferior, similar to the target-independent BiLSTM. This demonstrates that only passing the context to target is similar to not considering the target information owing to the problems we discussed before.

The F1-score of Outward-LSTM exceeds that of the Inward-LSTM by more than 10%. This shows

Sentence	Distance-rule	Dependency rule	Pipeline	BiLSTM	TC-BiLSTM	IOG
The bread is top notch as well .	*top*✗	NULL✗	*top notch*✓	*top notch*✓	NULL✗	*top notch*✓
BEST spicy tuna roll, great asian salad .	*asian*✗	*“great”, ”asian”*✗	*great*✓	*asian*✗	*“BEST”, ”great”*✗	*great*✓
I love the drinks , esp lychee martini , and the food is also VERY good .	*lyche*✗	*love*✓	*love*✓	*“love”,“good”*✗	*“love”,“good”*✗	*love*✓
Food was decent , but not great .	*decent* ✗	*“decent”, “great”* ✗	*decent* ✗	*“decent”, “not great”* ✓	*decent* ✗	*“decent”, “not great”* ✓
The food was excellent - authentic Italian cuisine made absolutely fresh .	*excellent*✓	*NULL*✗	*excellent*✓	*excellent*✓	*“excellent”, “fresh”*✗	*excellent*✓
The food was excellent - authentic Italian cuisine made absolutely fresh .	*Italian*✗	*“authentic”, “Italian ”*✗	*authentic*✗	*excellent*✗	*“authentic”, “excellent ”, “fresh”*✗	*“authentic”, “fresh”*✓

Table 4: Examples for the extracted result, the target terms are in red and the golden corresponding opinion words are in blue.

that passing target into context is a better choice and learning the target-specific word representations is crucial. In fact, Outward-LSTM has already outperformed all the previous baselines, which indicates that this is a really good design for TOWE. IO-LSTM which combine the Inward and Outward is slightly better than Outward-LSTM, showing that Inward-LSTM can still provide supplementary information for Outward-LSTM. Through combining global context with IO-LSTM as IOG model, we roughly obtain a further 1% improvement.

We also test our model with a linear Conditional-Random-Field as the decoder. CRF considers the label dependencies. It can be observed that IOG with CRF obtains a slight improvement.

4.7 Case Study

To demonstrate the effectiveness of our model, we pick some examples in the test dataset in **14res** and show the extracted results of different models.

In the first sentence, since the Distance-rule cannot extract phrases, the extraction it makes is incorrect. In addition, merely selecting the nearest adjective using the Distance-rule approach does not enable coverage in all cases, as shown in the second and third sentence (e.g., the *“asian”* and *“lyche”*). Dependency-rule in some cases fails to extract any word owing to the error of parser and no template to match. Pipeline method has the problem that it cannot handle the cases that one target corresponds to multiple opinion terms (e.g., *“not great”* is not extracted in the fourth sentence). The drawback of BiLSTM is that it does not include target information, so it extracts both *“love”* and *“good”* in the third sentence while only *“love”* is the corresponding opinion word for *“drinks”*. Although TC-BiLSTM is a target-specific model, it tends to extract irrelevant opinion words because of the interference from concatenation. In the last two rows of Table 4, we show the same sentence with two different targets and only IOG does not make mistakes for both targets. IOG outputs the correct results for all the sentences in the table.

5 Conclusion and Future Works

In this paper, we propose a novel subtask for aspect-based sentiment analysis: Target-oriented Opinion Words Extraction (TOWE) which aims at extracting the corresponding opinion words for a given opinion target. We design a novel neural model IOG to solve this task. IOG can effectively encode target information into left and right context respectively. Then we combine the left and right context of the opinion target and global context for extracting the corresponding opinion word in the decoder. We contribute four datasets based on several benchmarks. The experimental results demonstrate that our model achieves the best performance across all the datasets from different domains.

In future works, TOWE could be utilized to further improve the performance on downstream sentiment analysis tasks with building a more interpretable model, such as enhanced-feature or multi-task learning. In addition, an end-to-end opinion extractive summary method without given golden targets is also a future work.

Acknowledgements

The authors would like to thank Fang Qian and Fei Zhao for their contribution to building the datasets, and Robert Ridley for his comments on this paper. We also express the gratitude to the anonymous reviewers for their valuable feedback. This work is supported by the National Natural Science Foundation of China (No. 61672277, U1836221), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

References

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–

1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2886–2892.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. 2013. Opinion target extraction using partially-supervised word alignment model. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 2134–2140.

Pengfei Liu, Shafiq R. Joty, and Helen M. Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia V. Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez Zafra, and Gülsen Eryigit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 19–30.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 486–495.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 27–35.

Soujanya Poria, Erik Cambria, and Alexander F. Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning CRF for supervised aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 148–154.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective lstms for target-dependent sentiment classification. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 616–626.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3316–3322.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 592–598.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2979–2985.

Li Zhuang, Feng Jing, and Xiaoyan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 43–50.

Abstractive Summarization of *Reddit* Posts with Multi-level Memory Networks

Byeongchang Kim **Hyunwoo Kim** **Gunhee Kim**
Department of Computer Science and Engineering & Center for Superintelligence
Seoul National University, Seoul, Korea
`{byeongchang.kim,hyunwoo.kim}@vision.snu.ac.kr` `gunhee@snu.ac.kr`

Abstract

We address the problem of abstractive summarization in two directions: proposing a novel dataset and a new model. First, we collect *Reddit TIFU* dataset, consisting of 120K posts from the online discussion forum Reddit. We use such informal crowd-generated posts as text source, in contrast with existing datasets that mostly use formal documents as source such as news articles. Thus, our dataset could less suffer from some biases that key sentences usually locate at the beginning of the text and favorable summary candidates are already inside the text in similar forms. Second, we propose a novel abstractive summarization model named *multi-level memory networks* (MMN), equipped with multi-level memory to store the information of text from different levels of abstraction. With quantitative evaluation and user studies via Amazon Mechanical Turk, we show the *Reddit TIFU* dataset is highly abstractive and the MMN outperforms the state-of-the-art summarization models. The code and dataset are available at `http://vision.snu.ac.kr/projects/reddit-tifu`.

1 Introduction

Abstractive summarization methods have been under intensive study, yet they often suffer from inferior performance compared to extractive methods (Allahyari et al., 2017; Nallapati et al., 2017; See et al., 2017). Admittedly, by task definition, abstractive summarization is more challenging than extractive summarization. However, we argue that such inferior performance is partly due to some biases of existing summarization datasets. The source text of most datasets (Over et al., 2007; Hermann et al., 2015; Cohan et al., 2018; Grusky et al., 2018; Narayan et al., 2018a) originates from formal documents such as news articles, which have some structural patterns of which extractive methods better take advantage.

In formal documents, there could be a strong tendency that key sentences locate at the beginning of the text and favorable summary candidates are already inside the text in similar forms. Hence, summarization methods could generate good summaries by simply memorizing keywords or phrases from particular locations of the text. Moreover, if abstractive methods are trained on these datasets, they may not show much abstraction (See et al., 2017), because they are implicitly forced to learn structural patterns (Kedzie et al., 2018). Grusky et al. (2018) and Narayan et al. (2018a) recently report similar extractive bias in existing datasets. They alleviate this bias by collecting articles from diverse news publications or regarding intro sentences as gold summary.

Different from previous approaches, we propose to alleviate such bias issue by changing the source of summarization dataset. We exploit user-generated posts from the online discussion forum `Reddit`, especially `TIFU` subreddit, which are more casual and conversational than news articles. We observe that the source text in `Reddit` does not follow strict formatting and disallows models to simply rely on locational biases for summarization. Moreover, the passages rarely contain sentences that are nearly identical to the gold summary. Our new large-scale dataset for abstractive summarization named as *Reddit TIFU* contains 122,933 pairs of an online post as source text and its corresponding long or short summary sentence. These posts are written by many different users, but each pair of post and summary is created by the same user.

Another key contribution of this work is to propose a novel memory network model named *multi-level memory networks* (MMN). Our model is equipped with multi-level memory networks, storing the information of source text from different

Proceedings of NAACL-HLT 2019, pages 2519–2531
Minneapolis, Minnesota, June 2 - June 7, 2019.

levels of abstraction (*i.e.* word-level, sentence-level, paragraph-level and document-level). This design is motivated by that abstractive summarization is highly challenging and requires not only to understand the whole document, but also to find salient words, phrases and sentences. Our model can sequentially read such multiple levels of information to generate a good summary sentence.

Most abstractive summarization methods (See et al., 2017; Li et al., 2017; Zhou et al., 2017; Liu et al., 2018; Cohan et al., 2018; Paulus et al., 2018) employ sequence-to-sequence (seq2seq) models (Sutskever et al., 2014) where an RNN encoder embeds an input document and another RNN decodes a summary sentence. Our MMN has two major advantages over seq2seq-based models. First, RNNs accumulate information in a few fixed-length memories at every step regardless of the length of an input sequence, and thus may fail to utilize far-distant information due to vanishing gradient. It is more critical in summarization tasks, since input text is usually very long (>300 words). On the other hand, our convolutional memory explicitly captures long-term information. Second, RNNs cannot build representations of different ranges, since hidden states are sequentially connected over the whole sequence. This still holds even with hierarchical RNNs that can learn multiple levels of representation. In contrast, our model exploits a set of convolution operations with different receptive fields; hence, it can build representations of not only multiple levels but also multiple ranges (*e.g.* sentences, paragraphs, and the whole document).

Our experimental results show that the proposed MMN model improves abstractive summarization performance on both our new Reddit TIFU and existing Newsroom-Abs (Grusky et al., 2018) and XSum (Narayan et al., 2018a) datasets. It outperforms several state-of-the-art abstractive models with seq2seq architecture such as (See et al., 2017; Zhou et al., 2017; Li et al., 2017). We evaluate with quantitative language metrics (*e.g.* perplexity and ROUGE (Lin, 2004)) and user studies via Amazon Mechanical Turk (AMT).

The contributions of this work are as follows.

1. We newly collect a large-scale abstractive summarization dataset named *Reddit TIFU*. As far as we know, our work is the first to use non-formal text for abstractive summarization.
2. We propose a novel model named *multi-level memory networks* (MMN). To the best of our knowledge, our model is the first attempt to leverage memory networks for the abstractive summarization. We discuss the unique updates of the MMN over existing memory networks in Section 2.
3. With quantitative evaluation and user studies via AMT, we show that our model outperforms state-of-the-art abstractive summarization methods on both Reddit TIFU, Newsroom abstractive subset and XSum dataset.

2 Related Work

Our work can be uniquely positioned in the context of the following three topics.

Neural Abstractive Summarization. Many deep neural network models have been proposed for abstractive summarization. One of the most dominant architectures is to employ RNN-based seq2seq models with attention mechanism such as (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Cohan et al., 2018; Hsu et al., 2018; Gehrmann et al., 2018).In addition, recent advances in deep network research have been promptly adopted for improving abstractive summarization. Some notable examples include the use of variational autoencoders (VAEs) (Miao and Blunsom, 2016; Li et al., 2017), graph-based attention (Tan et al., 2017), pointer-generator models (See et al., 2017), self-attention networks (Liu et al., 2018), reinforcement learning (Paulus et al., 2018; Pasunuru and Bansal, 2018), contextual agent attention (Celikyilmaz et al., 2018) and integration with extractive models (Hsu et al., 2018; Gehrmann et al., 2018).

Compared to existing neural methods of abstractive summarization, our approach is novel to replace an RNN-based encoder with explicit multi-level convolutional memory. While RNN-based encoders always consider the whole sequence to represent each hidden state, our multi-level memory network exploits convolutions to control the extent of representation in multiple levels of sentences, paragraphs, and the whole text.

Summarization Datasets. Most existing summarization datasets use formal documents as source text. News articles are exploited the most, including in DUC (Over et al., 2007), Gigaword (Napoles et al., 2012), CNN/DailyMail (Nallapati et al., 2016; Hermann et al., 2015), News-

[Short Summary] (16 words)

TIFU by forgetting my chemistry textbook and all of my notes in a city five hours away

[Long Summary] (29 words)

TL;DR I forgot my chemistry textbook and binder full of notes in Windsor, which is five hour drive away and I am now screwed for the rest of the semester.

[Source Text] (282 words)

(…) So the past three days I was at a sporting event in Windsor. I live pretty far from Windsor, around a 5 hour drive. (…)
A five hour drive later, I finally got back home. I was ready to start catching up on some homework when I realized I left my binder (which has all of my assignments, homework etc.) in it, and my chemistry textbook back in Windsor.
I also have a math and chem test next week which I am now so completely screwed for. (…)

Figure 1: An example post of the `TIFU` subreddit.

room (Grusky et al., 2018) and XSum (Narayan et al., 2018a) datasets. Cohan et al. (2018) introduce datasets of academic papers from arXiv and PubMed. Hu et al. (2015) propose the LCSTS dataset as a collection of Chinese microblog's short text each paired with a summary. However, it selects only formal text posted by verified organizations such as news agencies or government institutions. Compared to previous summarization datasets, our dataset is novel in that it consists of posts from the online forum Reddit.

Rotten Tomatoes and Idebate dataset (Wang and Ling, 2016) use online text as source, but they are relatively small in scale: 3.7K posts of RottenTomatoes compared to 80K posts of TIFU-short as shown in Table 1. Moreover, Rotten Tomatoes use multiple movie reviews written by different users as single source text, and one-sentence consensus made by another professional editor as summary. Thus, each pair of this dataset could be less coherent than that of our TIFU, which is written by the same user. The Idebate dataset is collected from short arguments of debates on controversial topics, and thus the text is rather formal. On the other hand, our dataset contains the posts of interesting stories happened in daily life, and thus the text is more unstructured and informal.

Neural Memory Networks. Many memory network models have been proposed to improve memorization capability of neural networks (Kaiser et al., 2017; Na et al., 2017; Yoo et al., 2019). Weston et al. (2014) propose one of early memory networks for language question answering (QA); since then, many memory networks have been proposed for QA tasks (Sukhbaatar et al., 2015; Kumar et al., 2016; Miller et al., 2016). Park et al. (2017) propose a convolutional read memory network for personalized image captioning. One of the closest works to ours may be Singh et al. (2017), which use a memory network for text summarization. However, they only deal with extractive summarization by storing embeddings of individual sentences into memory.

Dataset	# posts	# words/post	# words/summ
`RottenTomatoes`	3,731	2124.7 (1747)	22.2 (22)
`Idebate`	2,259	178.3 (160)	11.4 (10)
`TIFU-short`	79,949	342.4 (269)	9.33 (8)
`TIFU-long`	42,984	432.6 (351)	23.0 (21)

Table 1: Statistics of the Reddit TIFU dataset compared to existing opinion summarization corpora, *RottenTomatoes* and *Idebate* (Wang and Ling, 2016). We show average and median (in parentheses) values.

Compared to previous memory networks, our MMN has four novel features: (i) building a multi-level memory network that better abstracts multi-level representation of a long document, (ii) employing a dilated convolutional memory write mechanism to correlate adjacent memory cells, (iii) proposing normalized gated tanh units to avoid covariate shift within the network, and (iv) generating an output sequence without RNNs.

3 Reddit TIFU Dataset

We introduce the Reddit TIFU dataset whose key statistics are outlined in Table 1. We collect data from Reddit, which is a discussion forum platform with a large number of subreddits on diverse topics and interests. Specifically, we crawl all the posts from 2013-Jan to 2018-Mar in the `TIFU` subreddit, where every post should strictly follow the posting rules, otherwise they are removed. Thanks to the following rules[1], the posts in this subreddit can be an excellent corpus for abstractive summarization: *Rule 3: Posts and titles without context will be removed. Your title must make an attempt to encapsulate the nature of your f***up. Rule 11: All posts must end with a TL;DR summary that is descriptive of your f***up and its consequences.* Thus, we regard the body text as source, the title as short summary, and the TL;DR summary as long summary. As a result, we make two sets of datasets: *TIFU-short* and *TIFU-long*. Figure 1 shows an example post of the `TIFU` subreddit.

---

[1] `https://reddit.com/r/tifu/wiki/rules`.

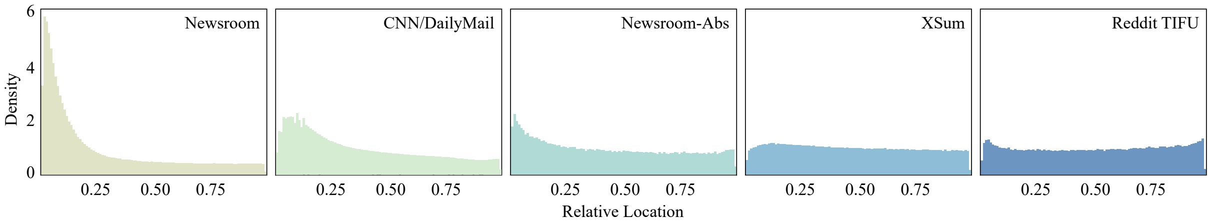


Figure 2: Relative locations of bigrams of gold summary in the source text across different datasets.

	`PG`			`Lead`			`Ext-Oracle`			`PG/Lead`	`PG/Oracle`
Dataset	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	Ratio (R-L)	Ratio (R-L)
`CNN/DM` (Nallapati et al., 2016)	36.4	15.7	33.4	39.6	17.7	36.2	54.7	30.4	50.8	0.92x	0.66x
`NY Times` (Sandhaus, 2008)	44.3	27.4	40.4	31.9	15.9	23.8	52.1	31.6	46.7	1.70x	0.87x
`Newsroom` (Grusky et al., 2018)	26.0	13.3	22.4	30.5	21.3	28.4	41.4	24.2	39.4	0.79x	0.57x
`Newsroom-Abs` (Grusky et al., 2018)	**14.7**	**2.2**	**10.3**	13.7	2.4	11.2	29.7	10.5	27.2	0.92x	0.38x
`XSum` (Narayan et al., 2018a)	29.7	9.2	23.2	16.3	1.6	12.0	29.8	8.8	22.7	1.93x	1.02x
`TIFU-short`	18.3	6.5	17.9	**3.4**	**0.0**	**3.3**	**8.0**	**0.0**	**7.7**	**5.42x**	**2.32x**
`TIFU-long`	19.0	3.7	15.1	**2.8**	**0.0**	**2.7**	**6.8**	**0.0**	**6.6**	**5.59x**	**2.29x**

Table 2: Comparison of F1 ROUGE scores between different datasets (row) and methods (column). `PG` is a state-of-the-art abstractive summarization method, and `Lead` and `Ext-Oracle` are extractive ones. `PG/Lead` and `PG/Oracle` are the ROUGE-L ratios of `PG` with `Lead` and `Ext-Oracle`, respectively. We report the numbers for each dataset (row) from the corresponding cited papers.

3.1 Preprocessing

We build a vocabulary dictionary $\mathcal{V}$ by choosing the most frequent V(=15K) words in the dataset. We exclude any urls, unicodes and special characters. We lowercase words, and normalize digits to 0. Subreddit names and user ids are replaced with @subreddit and @userid token, respectively. We use `markdown`[2] package to strip markdown format, and `spacy`[3] to tokenize words. Common prefixes of summary sentences (*e.g.* tifu by, tifu-, tl;dr, etc) are trimmed. We do not take OOV words into consideration, since our vocabulary with size 15K covers about 98% of word frequencies in our dataset. We set the maximum length of a document as 500. We exclude the gold summaries whose lengths are more than 20 and 50 for *TIFU-short* and *TIFU-long*, respectively. They amount to about 0.6K posts in both datasets (*i.e.* less than 1% and 3%). We use these maximum lengths, based on previous datasets (*e.g.* 8, 31, 56 words on average per summary in Gigaword, DUC, and CNN/DailyMail datasets, respectively). We randomly split the dataset into 95% for training, 5% for test.

3.2 Abstractive Properties of Reddit TIFU

We discuss some abstractive characteristics found in Reddit TIFU dataset, compared to existing summarization datasets based on news articles.

Weak Lead Bias. Formal documents including news articles tend to be structured to emphasize key information at the beginning of the text. On the other hand, key information in informal online text data are more spread across the text. Figure 2 plots the density histogram of the relative locations of bigrams of gold summary in the source text. In the CNN/DailyMail and Newsroom, the bigrams are highly concentrated on the front parts of documents. Contrarily, our Reddit TIFU dataset shows rather uniform distribution across the text.

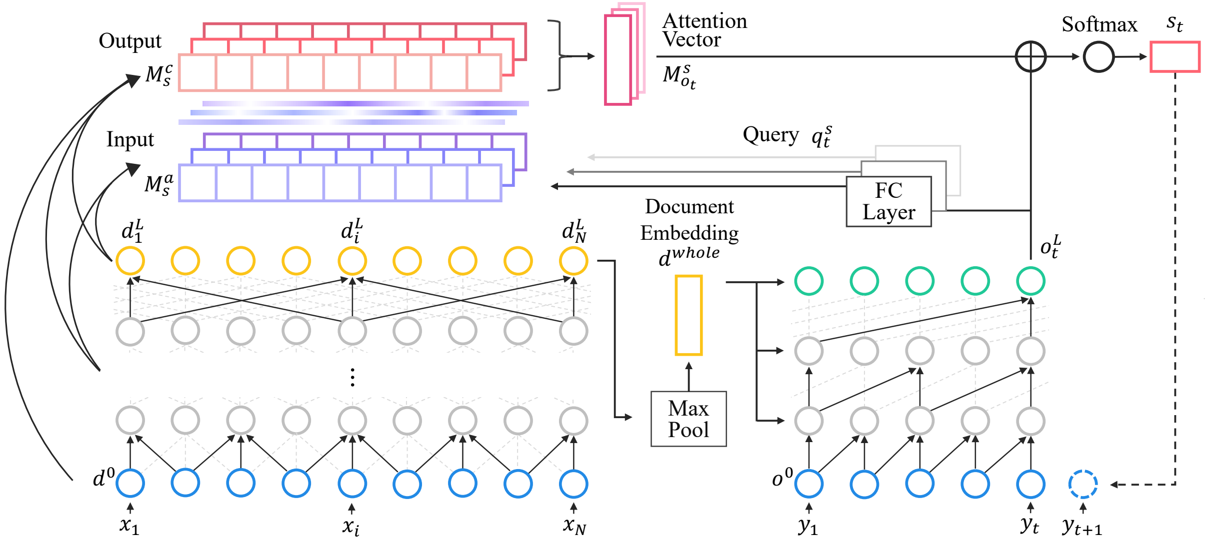
This characteristic can be also seen from the ROUGE score comparison in Table 2. The `Lead` baseline simply creates a summary by selecting the first few sentences or words in the document. Thus, a high score of the Lead baseline implicates a strong lead bias. The `Lead` scores are the lowest in our TIFU dataset, in which it is more difficult for models to simply take advantage of locational bias for the summary.

Strong Abstractness. Besides the locational bias, news articles tend to contain wrap-up sentences that cover the whole article, and they often have resemblance to its gold summary. Its existence can be measured by the score of the `Ext-Oracle` baseline, which creates a summary by selecting the sentences with the highest average score of F1 ROUGE-1/2/L. Thus, it can be viewed as an upper bound for extractive models (Narayan et al., 2018a,b; Nallapati et al., 2017).

---

[2] `https://python-markdown.github.io/`.

[3] `https://spacy.io`.



(a) Multi-level Memory (b) Sequence Decoder

Figure 3: Illustration of the proposed *multi-level memory network* (MMN) model.

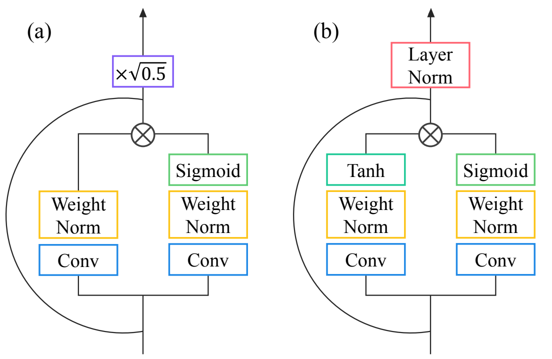


Figure 4: Comparison between (a) the gated linear unit (Gehring et al., 2017) and (b) the proposed normalized gated tanh unit.

In Table 2, the ROUGE scores of the `Ext-Oracle` are the lowest in our TIFU dataset. It means that the sentences that are similar to gold summary scarcely exist inside the source text in our dataset. This property forces the model to be trained to focus on comprehending the entire text instead of simply finding wrap-up sentences.

Finally, `PG/Lead` and `PG/Oracle` in Table 2 are the ROUGE-L ratios of `PG` with `Lead` and `Ext-Oracle`, respectively. These metrics can quantify the dataset according to the degree of difficulty for extractive methods and the suitability for abstractive methods, respectively. High scores of the TIFU dataset in both metrics show that it is potentially an excellent benchmark for evaluation of abstractive summarization systems.

4 Multi-level Memory Networks (MMN)

Figure 3 shows the proposed *multi-level memory network* (MMN) model. The MMN memorizes the source text with a proper representation in the memory and generates a summary sentence one word at a time by extracting relevant information from memory cells in response to previously generated words. The input of the model is a source text $\{x_i\} = x_1, ..., x_N$, and the output is a sequence of summary words $\{y_t\} = y_1, ..., y_T$, each of which is a symbol from the dictionary $\mathcal{V}$.

4.1 Text Embedding

Online posts include lots of morphologically similar words, which should be closely embedded. Thus, we use the `fastText` (Bojanowski et al., 2016) trained on the Common Crawl corpus, to initialize the word embedding matrix $\mathbf{W}_{emb}$. We use the same embedding matrix $\mathbf{W}_{emb}$ for both source text and output sentences. That is, we represent a source text $\{x_i\}_{i=1}^N$ in a distributional space as $\{\mathbf{d}_i^0\}_{i=1}^N$ by $\mathbf{d}_i^0 = \mathbf{W}_{emb}\mathbf{x}_i$ where $\mathbf{x}_i$ is a one-hot vector for i-th word in the source text. Likewise, output words $\{y_t\}_{t=1}^T$ is embedded as $\{\mathbf{o}_t^0\}_{t=1}^T$, and $\mathbf{d}_i^0$ and $\mathbf{o}_t^0 \in \mathbb{R}^{300}$.

4.2 Construction of Multi-level Memory

As shown in Figure 3(a), the multi-level memory network takes the source text embedding $\{\mathbf{d}_i^0\}_{i=1}^N$ as an input, and generates S number of memory tensors $\{\mathbf{M}_s^{a/c}\}_{s=1}^S$ as output, where superscript a and c denote input and output memory representation, respectively. The multi-level memory network is motivated by that when human understand a document, she does not remember it as a single whole document but ties together several levels of abstraction (*e.g.* word-level, sentence-level, paragraph-level and document-level). That is, we generate S sets of memory tensors, each of which associates each cell with different number of neighboring word embeddings based on the level of abstraction. To build memory slots of such multi-level memory, we exploit a multi-layer CNN as the write network, where each layer is chosen based on the size of its receptive field.

However, one issue of convolution is that large receptive fields require many layers or large filter sizes. For example, stacking 6 layers with a filter size of 3 results in a receptive field size of 13, *i.e.* each output depends on 13 input words. In order to grow the receptive field without increasing the computational cost, we exploit the *dilated* convolution (Yu and Koltun, 2016; Oord et al., 2016a) for the write network.

Memory Writing with Dilated Convolution. In dilated convolution, the filter is applied over an area larger than its length by skipping input values with a certain gap. Formally, for a 1-D n-length input $\mathbf{x} \in \mathbb{R}^{n\times 300}$ and a filter $\mathbf{w} : \{1, ..., k\} \to \mathbb{R}^{300}$, the dilated convolution operation $\mathcal{F}$ on s elements of a sequence is defined as

$$\mathcal{F}(\mathbf{x}, s) = \sum_{i=1}^{k} \mathbf{w}(i) * \mathbf{x}_{s+d\cdot(i-\lfloor k/2 \rfloor)} + \mathbf{b}, \quad (1)$$

where d is the dilation rate, k is the filter size, $s - d \cdot (i - \lfloor k/2 \rfloor)$ accounts for the direction of dilation and $\mathbf{w} \in \mathbb{R}^{k\times 300\times 300}$ and $\mathbf{b} \in \mathbb{R}^{300}$ are the parameters of the filter. With $d = 1$, the dilated convolution reduces to a regular convolution. Using a larger dilation enables a single output at the top level to represent a wider range of input, thus effectively expanding the receptive field.

To the embedding of a source text $\{\mathbf{d}_i^0\}_{i=1}^N$, we recursively apply a series of dilated convolutions $F(\mathbf{d}^0) \in \mathbb{R}^{N\times 300}$. We denote the output of the l-th convolution layer as $\{\mathbf{d}_i^l\}_{i=1}^N$.

Normalized Gated Tanh Units. Each convolution is followed by our new activation of *normalized gated tanh unit* (NGTU), which is illustrated in Figure 4(b):

$$\text{GTU}(\mathbf{d}^l) = \tanh(\mathcal{F}_f^l(\mathbf{d}^l)) \circ \sigma(\mathcal{F}_g^l(\mathbf{d}^l)), \quad (2)$$

$$\mathbf{d}^{l+1} = \text{LayerNorm}(\mathbf{d}^l + \text{GTU}(\mathbf{d}^l)), \quad (3)$$

where σ is a sigmoid, $\circ$ is the element-wise multiplication and F_f^l and F_g^l denote the filter and gate for l-th layer dilated convolution, respectively.

The NGTU is an extension of the existing gated tanh units (GTU) (Oord et al., 2016a,b) by applying weight normalization (Salimans and Kingma, 2016) and layer normalization (Ba et al., 2016). This mixed normalization improves earlier work of Gehring et al. (2017), where only weight normalization is applied to the GLU. As in Figure 4(a), it tries to preserve the variance of activations throughout the whole network by scaling the output of residual blocks by $\sqrt{0.5}$. However, we observe that this heuristic does not always preserve the variance and does not empirically work well in our dataset. Contrarily, the proposed NGTU not only guarantees preservation of activation variances but also significantly improves the performance.

Multi-level Memory. Instead of using only the last layer output of CNNs, we exploit the outputs of multiple layers of CNNs to construct S sets of memories. For example, memory constructed from the 4-th layer, whose receptive field is 31, may have sentence-level embeddings, while memory from the 8-th layer, whose receptive field is 511, may have document-level embeddings. We obtain each s-th level memory $\mathbf{M}_s^{a/c}$ by resembling key-value memory networks (Miller et al., 2016):

$$\mathbf{M}_s^a = \mathbf{d}^{\mathbf{m}(s)}, \ \mathbf{M}_s^c = \mathbf{d}^{\mathbf{m}(s)} + \mathbf{d}^0. \quad (4)$$

Recall that $\mathbf{M}_s^a$ and $\mathbf{M}_s^c \in \mathbb{R}^{N\times 300}$ are input and output memory matrix, respectively. $\mathbf{m}(s)$ indicates an index of convolutional layer used for the s-th level memory. For example, if we set $S = 3$ and $\mathbf{m} = \{3, 6, 9\}$, we make three-level memories, each of which uses the output of the 3-rd, 6-th, and 9-th convolution layer, respectively. To output memory representation $\mathbf{M}_s^c$, we add the document embedding $\mathbf{d}^0$ as a skip connection.

4.3 State-Based Sequence Generation

We discuss how to predict the next word y_{t+1} at time step t based on the memory state and previously generated words $y_{1:t}$. Figure 3(b) visualizes the overall procedure of decoding.

We first apply max-pooling to the output of the last layer of the encoder network to build a whole document embedding $\mathbf{d}^{whole} \in \mathbb{R}^{300}$:

$$\mathbf{d}^{whole} = \text{maxpool}([\mathbf{d}_1^L; ...; \mathbf{d}_N^L]). \quad (5)$$

The decoder is designed based on WaveNet (Oord et al., 2016a) that uses a series of causal dilated convolutions, denoted by $\hat{\mathcal{F}}(\mathbf{o}^l_{1:t}) \in \mathbb{R}^{t\times 300}$. We globally condition $\mathbf{d}^{whole}$ to obtain embeddings of previously generated words $\mathbf{o}^l_{1:t}$ as:

$$\mathbf{h}^l_{f/g} = \hat{\mathcal{F}}^l_{f/g}(\mathbf{o}^l_{1:t} + \mathbf{W}^l_{f/g}\mathbf{d}^{whole}), \quad (6)$$

$$\mathbf{h}^l_a = \tanh(\mathbf{h}^l_f) \circ \sigma(\mathbf{h}^l_g), \quad (7)$$

$$\mathbf{o}^{l+1}_{1:t} = \text{LayerNorm}(\mathbf{o}^l_{1:t} + \mathbf{h}^l_a), \quad (8)$$

where $\mathbf{h}^l_{f/g}$ are the filter and gate hidden state respectively, and learnable parameters are $\mathbf{W}^l_f$ and $\mathbf{W}^l_g \in \mathbb{R}^{300\times 300}$. We initialize $\mathbf{o}^0_t = \mathbf{W}_{emb}\mathbf{y}_t$. We set the level of the decoder network to $L = 3$ for TIFU-short and $L = 5$ for TIFU-long.

Next, we generate S number of query vectors $\{\mathbf{q}^s_t\}^S_{s=1}$ at time t to our memory network as

$$\mathbf{q}^s_t = \tanh(\mathbf{W}^s_q\mathbf{o}^L_t + \mathbf{b}^s_q), \quad (9)$$

where $\mathbf{W}^s_q \in \mathbb{R}^{300\times 300}$ and $\mathbf{b}^s_q \in \mathbb{R}^{300}$.

Each of these query vectors $\{\mathbf{q}^s_t\}^S_{s=1}$ is fed into the attention function of each level of memory. As in (Vaswani et al., 2017), the attention function is

$$\mathbf{M}^s_{o_t} = \text{softmax}(\frac{\mathbf{q}^s_t(\mathbf{M}^a_s)^T}{\sqrt{d^{emb}}})\mathbf{M}^c_s, \quad (10)$$

where we set $d^{emb} = 300$ for the embedding dimension and $\mathbf{M}^s_{o_t} \in \mathbb{R}^{300}$.

Next, we obtain the output word probability:

$$\mathbf{s}_t = \text{softmax}(\mathbf{W}_o[\mathbf{M}^1_{o_t}; ...; \mathbf{M}^S_{o_t}; \mathbf{o}^L_t]), \quad (11)$$

where $\mathbf{W}_o \in \mathbb{R}^{(300\times(S+1))\times V}$. Finally, we select the word with the highest probability $y_{t+1} = \text{argmax}_{\mathbf{s}\in\mathcal{V}}(\mathbf{s}_t)$. Unless y_{t+1} is an EOS token, we repeat generating the next word by feeding y_{t+1} into the output convolution layer of Eq.(8).

4.4 Training

We use the softmax cross-entropy loss from estimated y_t to its target $y_{GT,t}$. However, it forces the model to predict extremes (zero or one) to distinguish among the ground truth and alternatives. The label smoothing alleviates this issue by acting as a regularizer that makes the model less confident in its prediction. We smooth the target distribution with a uniform prior distribution u (Pereyra et al., 2017; Edunov et al., 2017; Vaswani et al., 2017). Thus, the loss over the training set $\mathcal{D}$ is

$$\mathcal{L} = -\sum \log p_\theta(\mathbf{y}|\mathbf{x}) - D_{KL}(u||p_\theta(\mathbf{y}|\mathbf{x})).$$

We implement label smoothing by modifying the ground truth distribution for word $y_{GT,t}$ to be $p(y_{GT,t}) = 1 - \epsilon$ and $p(y') = \epsilon/\mathcal{V}$ for $y' \neq y_{GT,t}$ where ϵ is a smoothing parameter set to 0.1. Further details can be found in the Appendix.

5 Experiments

5.1 Experimental Setting

Evaluation Metrics. We evaluate the summarization performance with two language metrics: perplexity and standard F1 ROUGE scores (Lin, 2004). We remind that lower perplexity and higher ROUGE scores indicate better performance.

Datasets. In addition to Reddit TIFU, we also evaluate on two existing datasets: abstractive subset of `Newsroom` (Grusky et al., 2018) and `XSum` (Narayan et al., 2018a). These are suitable benchmarks for evaluation of our model in two aspects. First, they are specialized for abstractive summarization, which meets well the goal of this work. Second, they have larger vocabulary size (40K, 50K) than Reddit TIFU (15K), and thus we can evaluate the learning capability of our model.

Baselines. We compare with three abstractive summarization methods, one basic seq2seq model, two heuristic extractive methods and variants of our model. We choose `PG` (See et al., 2017), `SEASS` (Zhou et al., 2017), `DRGD` (Li et al., 2017) as the state-of-the-art methods of abstractive summarization. We test the attention based seq2seq model denoted as `s2s-att` (Chopra et al., 2016). As heuristic extractive methods, the `Lead-1` uses the first sentence in the text as summary, and the `Ext-Oracle` takes the sentence with the highest average score of F1 ROUGE-1/2/L with the gold summary in the text. Thus, `Ext-Oracle` can be viewed as an upper-bound for extractive methods.

We also test variants of our method `MMN-*`. To validate the contribution of each component, we exclude one of key components from our model as follows: (i) `-NoDilated` with conventional convolutions instead, (ii) `-NoMulti` with no multi-level memory (iii) `-NoNGTU` with existing gated linear units (Gehring et al., 2017). That is, `-NoDilated` quantifies the improvement by the dilated convolution, `-NoMulti` assesses the effect of multi-level memory, and `-NoNGTU` validates the normalized gated tanh unit.

Please refer to the Appendix for implementation details of our method.

TIFU-short				
Methods	PPL	R-1	R-2	R-L
`Lead-1`	n/a	3.4	0.0	3.3
`Ext-Oracle`	n/a	8.0	0.0	7.7
`s2s-att` (Chopra et al., 2016)	46.2	18.3	6.4	17.8
`PG` (See et al., 2017)	40.9	18.3	6.5	17.9
`SEASS` (Zhou et al., 2017)	62.6	18.5	6.4	18.0
`DRGD` (Li et al., 2017)	69.2	14.6	3.3	14.2
`MMN`	32.1	**20.2**	**7.4**	**19.8**
`MMN-NoDilated`	**31.8**	19.5	6.8	19.1
`MMN-NoMulti`	34.4	19.0	6.1	18.5
`MMN-NoNGTU`	40.8	18.6	5.6	18.1
TIFU-long				
`Lead-1`	n/a	2.8	0.0	2.7
`Ext-Oracle`	n/a	6.8	0.0	6.6
`s2s-att` (Chopra et al., 2016)	180.6	17.3	3.1	14.0
`PG` (See et al., 2017)	175.3	16.4	3.0	13.5
`SEASS` (Zhou et al., 2017)	387.0	17.5	2.9	13.9
`DRGD` (Li et al., 2017)	176.6	16.8	2.0	13.6
`MMN`	**114.1**	**19.0**	**3.7**	**15.1**
`MMN-NoDilated`	124.2	17.6	3.4	14.1
`MMN-NoMulti`	124.5	14.0	1.5	11.8
`MMN-NoNGTU`	235.4	14.0	2.6	12.1

Table 3: Summarization results measured by perplexity and ROUGE-1/2/L on the TIFU-short/long dataset.

	Newsroom-Abs			**XSum**		
Methods	R-1	R-2	R-L	R-1	R-2	R-L
`s2s-att`	6.2	1.1	5.7	28.4	8.8	22.5
`PG`	14.7	2.2	11.4	29.7	9.2	23.2
`ConvS2S`	-	-	-	31.3	11.1	25.2
`T-ConvS2S`	-	-	-	31.9	11.5	25.8
`MMN` (Ours)	**17.5**	**4.7**	**14.2**	**32.0**	**12.1**	**26.0**

Table 4: Summarization results in terms of ROUGE-1/2/L on Newsroom-Abs (Grusky et al., 2018) and XSum (Narayan et al., 2018a). Except `MMN`, all scores are referred to the original papers. `T-ConvS2S` is the topic-aware convolutional seq2seq model.

	TIFU-short			**TIFU-long**		
vs. Baselines	Win	Lose	Tie	Win	Lose	Tie
`s2s-att`	**43.0**	28.3	28.7	**32.0**	24.0	44.0
`PG`	**38.7**	28.0	33.3	**42.3**	33.3	24.3
`SEASS`	**35.7**	28.0	36.3	**47.0**	37.3	15.7
`DRGD`	**46.7**	17.3	15.0	**61.0**	23.0	16.0
Gold	27.0	**58.0**	15.0	22.3	**73.7**	4.0

Table 5: AMT results on the TIFU-short/long between our `MMN` and four baselines and gold summary. We show percentages of responses that turkers vote for our approach over baselines.

5.2 Quantitative Results

Table 3 compares the summarization performance of different methods on the TIFU-short/long dataset. Our model outperforms the state-of-the-art abstractive methods in both ROUGE and perplexity scores. `PG` utilizes a pointer network to copy words from the source text, but it may not be a good strategy in our dataset, which is more abstractive as discussed in Table 2. `SEASS` shows strong performance in DUC and Gigaword dataset, in which the source text is a single long sentence and the gold summary is its shorter version. Yet, it may not be sufficient to summarize much longer articles of our dataset, even with its second-level representation. `DRGD` is based on the variational autoencoder with latent variables to capture the structural patterns of gold summaries. This idea can be useful for the similarly structured formal documents but may not go well with diverse online text in the TIFU dataset.

These state-of-the-art abstractive methods are not as good as our model, but still perform better than extractive methods. Although the `Ext-Oracle` heuristic is an upper-bound for extractive methods, it is not successful in our highly abstractive dataset; it is not effective to simply retrieve existing sentences from the source text. Moreover, the performance gaps between abstractive and extractive methods are much larger in our dataset than in other datasets (See et al., 2017; Paulus et al., 2018; Cohan et al., 2018), which means too that our dataset is highly abstractive.

Table 4 compares the performance of our MMN on Newsroom-Abs and XSum dataset. We report the numbers from the original papers. Our model outperforms not only the RNN-based abstractive methods but also the convolutional-based methods in all ROUGE scores. Especially, even trained on single end-to-end training procedure, our model outperforms `T-ConvS2S`, which necessitates two training stages of LDA and `ConvS2S`. These results assure that even on formal documents with large vocabulary sizes, our multi-level memory is effective for abstractive datasets.

5.3 Qualitative Results

We perform two types of qualitative evaluation to complement the limitation of automatic language metrics as summarization evaluation.

User Preferences. We perform Amazon Mechanical Turk (AMT) tests to observe general users' preferences between the summarization of different algorithms. We randomly sample 100 test examples. At test, we show a source text and two summaries generated by our method and one baseline in a random order. We ask turkers to choose the more relevant one for the source text. We obtain answers from three different turkers for each test example. We compare with four abstractive baselines (`s2s-att`, `PG`, `SEASS` and `DRGD`)

[Source Text]

(…) I decided to go over to my friends house to a small party at 1 in the morning. I knew my parents would say no so I snuck out of the house. (…) I had been talking to my mom about how sad even hearing the theme song made me. Also she had seen me watching a bunch of sad anime theme songs and tearing up a little so she must have thought I was depressed. When I got home today my mom was practically in tears. (…)

[Short Summary]

(GT) sneaking out of my friends house last night
(Ours) sneaking out of my friends house
(PG) not watching my friends
(SEASS) accidentally spoiling my mom song
(s2s-att) sneaking out of town
(DRGD) watching a movie

[Source Text]

(…) Saturday was on my way to a party and this dog was walking in the road. (…) Since it was a holiday I couldn't get her scanned for a chip but she was obviously neglected. Missing fur from flea infestation, (…) Yesterday I was able to go get her scanned for a chip. No chip. So I get ready to take her home and deflea her. (…) Anyway a third party today starts accusing me of stealing (…) and talking about pressing charges. (…)

[Long Summary]

(GT) Saved a dog. Had to give dog back to possible abusers. Being accused of stealing the fucking dog. No good deed goes unpunished.
(Ours) tried to help a dog got a bit and got accused of stealing
(PG) _EOS
(SEASS) called a dog a _UNK might get charged with _UNK
(s2s-att) got accused of being a dog by stealing a _UNK bit the dog and accused of stealing dog to the police
(DRGD) i was a _UNK dog and I wasn't playing attention and got arrested for being a _UNK _UNK

Figure 5: Examples of abstractive summary generated by our model and baselines. In each set, we too show the source text and gold summary.

and the gold summary (Gold).

Table 5 summarizes the results of AMT tests, which validate that human annotators significantly prefer our results to those of baselines. As expected, the gold summary is voted the most.

Summary Examples. Figure 5 shows selected examples of abstractive summarization. Baselines often generate the summary by mostly focusing on some keywords in the text, while our model produces the summary considering both keywords and the whole context thanks to multi-level memory. We present more examples in the Appendix.

6 Conclusions

We introduced a new dataset *Reddit TIFU* for abstractive summarization on informal online text. We also proposed a novel summarization model named *multi-level memory networks* (MMN). Experiments showed that the Reddit TIFU dataset is uniquely abstractive and the MMN model is highly effective. There are several promising future directions. First, ROUGE metrics are limited to correctly capture paraphrased summaries, for which a new automatic metric of abstractive summarization may be required. Second, we can explore the data in other online forums such as Quora, Stackoverflow and other subreddits.

Acknowledgments

We thank Chris Dongjoo Kim, Yunseok Jang and the anonymous reviewers for their helpful comments. This work was supported by Kakao and Kakao Brain corporations and IITP grant funded by the Korea government (MSIT) (No. 2017-0-01772, Development of QA systems for Video Story Understanding to pass the Video Turing Test). Gunhee Kim is the corresponding author.

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. In *arXiv:1707.02268*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. In *Stat*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. In *TACL*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep Communicating Agents for Abstractive Summarization. In *NAACL-HLT*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive Sentence Summarization With Attentive Recurrent Neural Networks. In *NAACL-HLT*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. In *NAACL-HLT*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical Structured Prediction Losses for Sequence to Sequence Learning. In *NAACL-HLT*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *AISTATS*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies. In *NAACL-HLT*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *NIPS*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss. In *ACL*.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *EMNLP*.

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to Remember Rare Events. In *ICLR*.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content Selection in Deep Learning Models of Summarization. In *EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me Anything: Dynamic Memory Networks for Natural Language Processing. In *ICML*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep Recurrent Generative Decoder for Abstractive Text Summarization. In *EMNLP*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *TSBO*.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. In *ICLR*.

Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *EMNLP*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. 2017. A Read-Write Memory Network for Movie Story Understanding. In *ICCV*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization Using Sequence-to-sequence RNNs and Beyond. In *CoNLL*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *NAACL-HLT AKBC-WEKEX*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *EMNLP*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. Ranking Sentences for Extractive Summarization with Reinforcement Learning. In *NAACL-HLT*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016a. WaveNet: A Generative Model for Raw Audio. In *SSW*.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016b. Conditional Image Generation With Pixelcnn Decoders. In *NIPS*.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. In *IPM*.

Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. 2017. Attend to You: Personalized Image Captioning with Context Sequence Memory Networks. In *CVPR*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-Reward Reinforced Summarization with Saliency and Entailment. In *NAACL-HLT*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *ICLR*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *ICLR*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*.

Tim Salimans and Diederik P Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *NIPS*.

Evan Sandhaus. 2008. New York Times Annotated Corpus. In *LDC*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the Point: Summarization with Pointer-Generator Networks. In *ACL*.

Abhishek Kumar Singh, Manish Gupta, and Vasudeva Varma. 2017. Hybrid MemNet for Extractive Summarization. In *CIKM*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end Memory Networks. In *NIPS*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive Document Summarization with a Graph-based Attentional Neural Model. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Lu Wang and Wang Ling. 2016. Neural Network-Based Abstract Generation for Opinions and Arguments. In *NAACL-HLT*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. In *ICLR*.

Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. 2019. Coloring with Limited Data: Few-shot Colorization via Memory-Augmented Networks. In *CVPR*.

Fisher Yu and Vladlen Koltun. 2016. Multi-scale Context Aggregation by Dilated Convolutions. In *ICLR*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective Encoding for Abstractive Sentence Summarization. In *ACL*.

Description	**Common Configurations**
Initial learning rate	0.001
Embedding dimension (d^{emb})	300
Kernel size (k)	3
Dilation rate (d)	2^l

Description	**TIFU-short**	**TIFU-long**
Grad clip	0.3	0.3
# of encoder layers	9	8
# of decoder layers	3	5
Layers used for memory ($\mathbf{m}$)	{3, 6, 9}	{4, 8}
Smoothing parameter (ϵ)	0.1	0.05
Description	**Newsroom-Abs**	**XSum**
Grad clip	0.3	0.8
# of encoder layers	10	9
# of decoder layers	6	6
Layers used for memory ($\mathbf{m}$)	{3, 6, 10}	{4, 7, 9}
Smoothing parameter (ϵ)	0.05	0.05

Table 6: Model hyperparameters in experiments on TIFU-short/long, Newsroom abstractive subset and XSum.

	Novel N-gram Ratio			
Dataset	1-gram	2-gram	3-gram	4-gram
CNN/DailyMail	10.3	49.9	70.5	80.3
NY Times	11.0	45.5	67.2	77.9
Newsroom	15.6	45.4	57.2	62.2
Newsroom-Ext	1.5	5.9	8.9	11.1
Newsroom-Mix	11.6	47.0	66.5	76.8
Newsroom-Abs	33.9	83.9	97.1	99.5
XSum	35.8	83.5	95.5	98.5
TIFU-short	29.7	71.5	88.1	93.8
TIFU-long	27.4	76.7	92.5	97.0

Table 7: Comparison of novel N-gram ratios between Reddit TIFU and other summarization datasets.

A Implementation Details

All the parameters are initialized with the Xavier method (Glorot and Bengio, 2010). We apply the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e - 8$. We apply weight normalization (Salimans and Kingma, 2016) to all layers. We set learning rate to 0.001 and clip gradient at 0.3. At every 4 epochs, we divide learning rate by 10 until it reaches 0.0001. We train our models up to 12 epochs for TIFU-short and 60 epochs for TIFU-long.

Table 6 summarizes the setting of hyperparameters for our model in all experiments on TIFU-short/long dataset, Newsroom abstractive subset and XSum.

B Novel N-gram Ratios

Table 7 compares the ratios of novel N-grams in the reference summary between datasets. Following (See et al., 2017; Narayan et al., 2018a), we compute this ratio as follows; we first count the number of N-grams in the reference summary that do not appear in the source text and divide it with the total number of N-grams. The higher the ratio is, the less the identical N-grams are in the source text. The CNN/DailyMail, New York Times, Newsroom datasets all, for example, exhibit low novel 1-gram ratios as 10.3%, 11.0%, 15.6 % respectively. This means that about 90% of the words in reference summary already exist inside the source text. It is due to that the summaries from formal documents (*e.g.* news and academic papers) tend to have same expressions with the source documents. Therefore, these datasets may be more suitable for extractive summarization than abstractive one; on the other hand, our dataset is more abstractive.

We also compare the novel N-gram ratio for XSum and three subsets of Newsroom;

(i) Newsroom-Ext, a subset favorable for extractive methods, (ii) Newsroom-Mix, a subset favorable for mixed methods, and (iii) Newsroom-Abs, a subset favorable for abstrac-

[Source Text]
(…) This weekend I went out and bought myself a motorcycle! I've been planning on buying one this summer and I finally went out and did it. (…) I geared up and went on my way for my first ride. (…) I went all the way back to my house and alas, no phone. So I spent the next 4 hours walking the route to my girlfriends house looking along the side of the road for my phone. These are all back roads to my girlfriends house so I had to really get in the undergrowth to look for my phone. It got dark and I headed home feeling dejected. The next morning my friend came over with my phone! The only damage was a few scuffs to my otterbox. I couldn't believe he found it, and as it turns out it fell off my bike and tumbled onto the side of his driveway. I chalked this up as a win and considered myself lucky. Until yesterday. I woke up with poison ivy all over my body. And when I say all over my body, I mean ALL OVER my body. I have some on my arms, legs, face, and most importantly all over my dick. And the worst spot of them all is on my dick. I have never been so uncomfortable in my life. I must have had some of the oil on my hands and scratched an itch down there. Needless to say I haven't been on my new motorcycle since I've had this and I've been doing as little moving as possible at my job and at home. F***.

[Short Summary]
(Gold) taking a ride on my new motorcycle
(Ours) buying my new motorcycle
(s2s-att) trying to be a good samaritan
(PG) not wearing my cargo helmet
(SEASS) going for a ride
(DRGD) getting my phone stuck

[Source Text]
(…) Later that night, the rest of the family proceeded to play cards and become quite intoxicated. Me, being the little shit that I was, and probably still am, took this opportunity to raid the liquor cooler, and made off with a bottle of wine. I, along with a few other of my underage cousins, ran off to consume our loot. Now, in most situations, this really wouldn't be a that big deal, however, getting drunk at 18 wasn't where my f*** up occurred. (…) John tells her it'll be okay, that he will smooth talk it over with them, and that she should bring them a bottle of their favorite wine. Well John and Jane are having dinner outside to meet the parents, and when Jane goes to retrieve her gift of wine to John's parents, she discovers the wine is gone. Jane then begins to panic, and starts tearing up the surrounding area looking for it. John's parents have no idea what she's freaking out about, or why John would bring crazy to the family reunion. (…) Jane now slips into complete hysteria, and runs inside to lock herself in the bathroom. (…) The day after, John convinces his parents to try again, and all goes very well, especially since last night's thief is still recovering from his first wine hangover. (…)

[Short Summary]
(Gold) stealing a bottle of wine
(Ours) getting drunk and stealing a wine bottle
(s2s-att) getting drunk and making a family cry
(PG) ruining my future dinner vacation
(SEASS) accidentally stealing alcohol from my cousin’s parent′s party
(DRGD) smoking a cigarette

[Source Text]
(…) We use an internal messaging application software at work which has been great for communicating with other teammates. A lot of us have started using it to complain about things we are not happy about at work. (…) This leads me to today where just as i am about to go home my manager calls me in to a private meeting looking really upset. Then they mentioned the program name and that they had received an email, and suddenly I realized I had fucked up one of the quirks of this program is that when someone is offline it emails them the message. A recently ex co-worker is still active on the chat for quick questions for the next 2 weeks. They came online so we started having a conversation, then another co-worker walked up to me for a chat who has been having a rough week and complained about our boss. When they finished their rant, I then messaged my ex co-worker that my boss wasn't popular with the staff at the moment as that was the second minor complaint I had heard that week. They had gone offline, so an email was sent to their old work email. Past employees emails get sent to the boss, in case important emails are sent to them. So after the meeting I still have my job. I had an awkward conversation with my boss (…)

[Long Summary]
(Gold) message program at work emailed a private message between a past co worker and myself to my boss saying how people where not happy with them
(Ours) sent a message to my boss and now i ’m in a meeting with my boss
(s2s-att) i lied about my boss to get my job and now i ’m in a job with a new job
(PG) i got a program that sacked from work and i got sent to a metting by my boss

[Source Text]
(…) My girlfriend and I have just started to "get a bit more uncomfortable with each other". (…) My curfew for the night had been midnight, however we both got a bit carried away in the beautiful setting (…) I had over extended my curfew by around an hour. This prompted my mother to call me angrily and groggily claiming that she was mad at me. I defended myself by saying that we both fell asleep listening to the docile tones of Steve Harvey on "Family Feud". (…) I sent my girlfriend a text that said something along the lines of "X acts of affection. Comments questions, or complaints?" However upon hitting the send button I noticed that it was heading in the direction of my birth giver. I panicked and luckily managed to put it into airplane mode! (…) I quickly googled "How to prevent texts from sending" and it said to simply delete the text while it was on airplane mode. I did so, and proceeded to turn airplane mode off, however the "always correct Internet" was wrong and I hear my mom receive the text in the next room over. I quickly went into her room to try and crack her phone's code in order to diffuse the bomb (…) I then asked my mother for her phone so that I could "call mine". She reluctantly agreed and I hurriedly rushed to my room. (…)

[Long Summary]
(Gold) did some funky stuff with my date
(Ours) accidentally sent an inappropriate text to a girl and then accidentally sent it to my mother′s phone and now i’m trouble
(PG) tifu by answering an airplane call and accidentally adding my mothers phone and her phone to find out she was sleeping on me
(SEASS) I accidentally sent a text to my mom that I was sending her a text from the _UNK bomb

Figure 6: Examples of abstractive summary generated by our model and baselines. In each set, we too show the source text and reference summary.

tive methods. We summarize two interesting observations as follows. First, as expected, the more favorable for abstractive methods is, the higher novel n-gram ratio is. Second, novel n-gram ratios of `Newsroom-Abs` and `XSum` are higher than those of our dataset, even though their data sources are news publications. Thus, we argue that novel n-gram ratios are pretty good but not a sufficient measure to find extractive bias in the summarization dataset.

C More Examples

Figure 6 illustrates selected examples of summary generation. In each set, we show a source text, a reference summary and generated summaries by our method and baselines. In the examples, while baselines generate summary by mostly focusing on some keywords, our model produces summary considering both keywords and the whole context thanks to the multi-level memory.

Automatic learner summary assessment for reading comprehension

Menglin Xia
Amazon Research Cambridge*
ximengli@amazon.co.uk

Ekaterina Kochmar
ALTA Institute
Computer Laboratory
University of Cambridge
ek358@cl.cam.ac.uk

Ted Briscoe
ALTA Institute
Computer Laboratory
University of Cambridge
ejb@cl.cam.ac.uk

Abstract

Automating the assessment of learner summaries provides a useful tool for assessing learner reading comprehension. We present a summarization task for evaluating non-native reading comprehension and propose three novel approaches to automatically assess the learner summaries. We evaluate our models on two datasets we created and show that our models outperform traditional approaches that rely on exact word match on this task. Our best model produces quality assessments close to professional examiners.

1 Introduction

Summarization is a well-established method of measuring reading proficiency in traditional English as a second or other language (ESOL) assessment. It is considered an effective approach to test both cognitive and contextual dimensions of reading (Weir et al., 2013). However, due to the high time and cost demands of manual summary assessment, modern English exams usually replace the summarization task with multiple choice or short answer questions that are easier to score (Alderson, 2005). Automating the assessment of learner summarization skills provides an efficient evaluation method for the quality of the learner summary and can lead to effective educational applications to enhance reading comprehension tasks.

In this paper, we present a summarization task for evaluating non-native reading comprehension and propose three novel machine learning approaches to assessing learner summaries. First, we extract features to measure the content similarity between the reading passage and the summary. Secondly, we calculate a similarity matrix based on sentence-to-sentence similarity between the reading passage and the summary, and apply a Convolutional Neural Network (CNN) model to assess the summary quality using the similarity matrix. Thirdly, we build an end-to-end summarization assessment model using the Long Short Term Memory (LSTM) model. Finally, we combine the three approaches in a single system using a simple parallel ensemble modeling technique. We compiled two datasets to evaluate our models, and we release this data with the paper. We show that our models outperform traditional approaches that rely on exact word match on the task and that our best model produces quality assessments close to professional examiners.

2 Related Work

Most of the previous studies on summary assessment are aimed at evaluating automated summarization systems (Lin, 2004; Lin and Hovy, 2003; Nenkova et al., 2007). In contrast to this line of work, our goal is to assess human-written summaries rather than machine-generated ones.

Within the educational domain, several applications have been developed to help students with their writing summarization skills. Summary Street (Wade-Stein and Kintsch, 2004) is an educational software designed for children to develop summarization skills. It asks students to write a summary to a reading passage, and scores the summary by using Latent Semantic Analysis (LSA) to construct semantic representations of the text. This system uses the cosine similarity score based on LSA as the sole indicator of content similarity. OpenEssayist (Whitelock et al., 2013) is an interactive system that provides students with the automated feedback about their summative essays. The system extracts the key words, phrases and sentences from the essays and helps the students to investigate their distribution in text and the po-

---

[*] The work by the first author was done at the University of Cambridge prior to joining Amazon Research.

Proceedings of NAACL-HLT 2019, pages 2532–2542
Minneapolis, Minnesota, June 2 - June 7, 2019.

tential implications for the clarity of the narrative.

The work that is most similar to ours is the automatic scoring of a summarization task by Madnani et al. (2013), who designed a task to measure the reading comprehension skills of American students. In their experiments, students were asked to write a four-sentence summary for each of the two three-paragraph reading passages, with the first sentence summarizing the whole passage and the following three sentences summarizing each paragraph. To build an automated system to score the summaries, they randomly select a student summary with the highest score as the reference, and use 8 feature types to train a logistic regression classifier to predict the summary score. They train a separate classifier for each of the two passages, and obtain accuracy scores of 65% and 52% respectively, outperforming the most-frequent-score baselines of 51% and 32%. Most of the features used in Madnani et al. (2013) are based on verbatim overlap. Although such metrics prove to be effective in various tasks, they cannot capture the content similarity when paraphrasing or a higher level of abstraction are used in the summary.

Few studies have addressed summarization assessment at a higher level. More recently, Ruseti et al. (2018) have used Recurrent Neural Networks (RNNs) to automatically score summaries. In their model, a concatenated representation of the summary and the text built from two separate RNNs as well as a complexity score of the text are fed to a fully connected layer to predict a real number between $[0, 1]$. This number is then mapped to 3 discrete classes representing the quality of the summary using linear regression. Their best model achieves 55.24% in accuracy on a dataset of 636 summaries collected using Mechanical Turk.

In this paper, we address several limitations of previous work. We build a system that uses verbatim features as well as features capturing higher level of abstraction. First, we aim to build a generic system that can evaluate the quality of a summary without having to train a separate model for each text. Second, whereas Madnani et al. (2013) use a student summary with the highest score as the reference to evaluate the candidate summary, our goal is to build a fully-automated system that does not require selecting a pre-defined reference. Third, we aim to explore features and structures capable of better modeling semantic similarity beyond verbatim overlap.

3 Data

This section outlines the summarization task used in our experiments. First, learners, regardless of their proficiency level, were asked to read three reading passages extracted from the Cambridge English Exams dataset of Xia et al. (2016) at the lower (B1), upper intermediate (B2), and advanced (C1) levels of the Common European Framework of Reference for Languages (CEFR). Then they were asked to write a summary of 50, 100, and 120 words for each of the three tasks.[1]

3.1 Pilot study with simulated learner data

Before launching the experiments with the actual language learners and in order to develop the automated summary evaluation system, we first ran a pilot study and collected "simulated learner" summaries from 50 members of our university. Since most participants of this study would speak English at an advanced (C1-C2) level, we asked them to write a "good summary" and a "bad summary" for each reading passage, mimicking a learner. The participants were asked to produce grammatically correct sentences and to write a "bad" summary in a way that a learner who does not fully understand the original passage might produce. In total, 300 summaries were collected (with 150 good summaries and 150 bad ones). The simulated learner data was then used to train binary classification systems to assess whether a summary captures the passage content properly or not.

3.2 Real learner data

Learner levels	Count
B1	40
B2	40
C1-C2	57
Total	137

Table 1: The distribution of the learner proficiency levels in the real learner data

Next, we collected summaries from second language learners at B1, B2 and C1-C2 levels of proficiency.[2] In total, 411 summaries from 137 learners were collected. The distribution of the learner proficiency levels is shown in Table 1.

---

[1] The word limits on the summarization tasks are set to keep a relatively constant compression ratio between the summary and the length of the original passage.

[2] The proficiency levels of learners were self-identified.

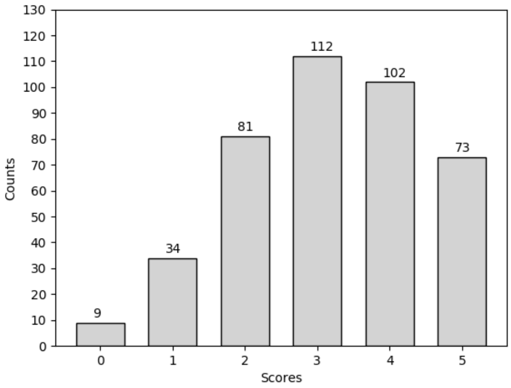


Figure 1: A histogram illustrating the score distribution in the real learner data

The summaries were then scored by three professional ESOL annotators using a 6-point scale defined as follows:

Band 5: *The summary demonstrates excellent understanding of the original passage: Content covers all the main points of the passage. All content included is accurate, with no irrelevant details or repetitions. Target reader is fully informed.*

Band 4: *Performance shares features of Bands 3 and 5.*

Band 3: *The summary demonstrates acceptable understanding of the passage: Most of the main points are included. Most of the content is relevant and paraphrased, with some irrelevant details, repetitions or inaccuracy of content. Target reader is on the whole informed.*

Band 2: *Performance shares features of Bands 1 and 3.*

Band 1: *The summary demonstrates very little understanding of the passage: Most of the content is of limited relevance, with repetitions or verbatim borrowing from the original text. In some paraphrased parts of the text, inaccuracy of content or omissions of main points are evident. Target reader is minimally informed.*

Band 0: *No understanding of the passage is demonstrated. The content is totally irrelevant to the original passage. Target reader is not informed.*

Figure 1 shows the distribution of the scores for the learner summaries. The pairwise correlation between annotators ranges between 0.690 and 0.794. To derive the final score for each summary, we take the average of the scores by the three annotators. This results in a set of real-valued average scores on the scale of $[0, 5]$ and allows us to treat this task as a regression problem and make use of the continuity of the assessment scale. The goal of the experiments on this data is then to train a regression model to predict a score that correlates well with the annotators' judgments.

4 Methods

In this section, we introduce three different approaches to the automated evaluation of the learner summaries.

4.1 Measures for summary assessment

First of all, we extract a number of features to describe the similarity of the summary and the reading text and apply a machine learning model to predict the summary quality.

The summarization task for reading comprehension examines the content relevance and the ability to convey the main ideas of the text in the summary. To automatically assess the learner summary, we compare the candidate summary against a reference to assess the quality of its content.

We experiment with two types of references to evaluate the candidate summary: firstly, we compare the candidate summary against the *original passage* directly, and secondly, we extract key sentences from the original text with an automated extractive summarizer and compare the candidate summary to the set of key sentences. Ideally, an extractive summarizer extracts a subset of sentences from the passage that are highly representative of the original text. Although the extracted key sentences are not necessarily coherent among themselves, they provide a representation of the main ideas of the text. Comparing the candidate summary against the key sentences allows us to examine the content relevance and the coverage of the main ideas in the candidate summary. We compare two popular summarizers in selecting the key sentences for reference: *TextRank* (Mihalcea and Tarau, 2004) and *MEAD* (Radev et al., 2004). We also compare the extractive summarizers against the baseline of using a *random selection* of sentences as the reference.

After obtaining the reference, we derive four types of linguistic features to evaluate the quality of the learner summary: (1) verbatim features, (2) semantic similarity features, (3) features based on distributed vector representations of the summary, and (4) features that describe discourse and other textual characteristics of the summary.

4.1.1 Verbatim features

Verbatim similarity is the most straightforward measure that indicates content similarity. Verbatim features measure the lexical overlap of the text units between the candidate summary and the reference. We use the following metrics to measure verbatim similarity: **ROUGE** (Lin, 2004), **BLEU** (Papineni et al., 2002), and **METEOR** (Denkowski and Lavie, 2011). The three metrics are commonly used to assess automated summarization systems. ROUGE and BLEU are based on exact word match of N-grams, and METEOR extends the exact word match with stem, synonym, and paraphrase matches extracted from the WordNet (Miller, 1995) and a background dictionary, which allows for more flexible expressions.

4.1.2 Semantic similarity features

Although verbatim overlap metrics prove to be effective in various tasks, they fail to capture the content similarity when paraphrasing and higher levels of abstraction are used in the summary. To compensate for this, word embeddings and sentence embeddings are used to model semantic similarity at the word and the sentence level. We measure the semantic similarity between words and sentences in the texts and combine the scores into a measure of document-level semantic similarity.

1. **Word similarity**: **Word2vec** (Mikolov et al., 2013) is a model for learning distributed vector representations of words from a large corpus of text. We use embeddings pre-trained on Wikipedia to compute word-to-word cosine similarity between the candidate summary and the reference. We experiment with three scoring functions to construct the text-level semantic similarity measures from the word-to-word scores:

   (1) *average word similarity* on every word pair in the candidate summary and the reference;

   (2) *a greedy method* (Mihalcea et al., 2006) that finds the best-matching word with maximum similarity scores and computes the average over the greedily selected pairs;

   (3) *optimal matching* (Rus and Lintean, 2012) that finds the optimal alignment of word pairs and then takes the average over the alignment.

2. **Sentence similarity**: **Skip-thought** (Kiros et al., 2015) is a model for learning distributed representations of sentences. It uses an RNN-encoder to compose the sentence vector, and a decoder conditioned on the resulting vector that tries to predict the previous and the next sentences in the context. We use the model pre-trained on the BookCorpus (Zhu et al., 2015) to generate our sentence vectors. Additionally, we experiment with composing the sentence vectors using word embedding summation and taking the average (**average word embeddings**). We use the same functions for word-level similarity to compute the text semantic similarity from the sentence vectors.

4.1.3 Distributed vector representations of the summary

In addition to the word and sentence similarities, we investigate methods to model the content similarity between the candidate summary and the reference directly at the document level.

Specifically, we use the following five approaches to construct vector representations of learner summaries:

TF-IDF is a common method to construct document representations in information retrieval. TF-IDF weighted document vectors are frequently used for measuring query-document similarity.

Doc2Vec (Le and Mikolov, 2014) is a neural network model for learning distributed representation of documents. We use the "distributed memory of paragraph vectors (PV-DM)" variant of the model to construct our vector representation of the summary. The PV-DM model maps the document to a vector space and uses a combination of the document vector and the vectors of surrounding words to predict a target word.

Latent Semantic Analysis (LSA) (Landauer, 2006) applies singular value decomposition (SVD) on the term-document matrix to obtain vector space representation of documents.

Latent Dirichelet Allocation (LDA) (Blei et al., 2003) represents the documents as mixtures of topics. It can be used to measure the content similarity and topical relevance of documents.

We also make use of the **average word embeddings** to encode the summary.

We use the Simple English Wikipedia corpus[3] as our background resource to learn the document representations. The Simple English Wikipedia data is used to train the models because its documents are rendered simple for English learners. Therefore, the lexical usage and syntactic structure in Simple English Wikipedia are more similar to the summaries written by learners. We take

---

[3] `https://simple.wikipedia.org`

the cosine similarity between the candidate and the reference vectors to evaluate their similarity.

4.1.4 Discourse and other textual features

Apart from the content-based measures of the summary, the textual quality of the summary is also important for its overall quality estimation. For instance, good summaries tend to be more coherent and logically consistent. We extract **lexical chain**-based discourse measures to assess the coherence of the text. Lexical chains model the semantic relations among entities throughout the text. We implement the lexical chaining algorithm developed by Galley and McKeown (2003) and extract 7 lexical chain-based features.[4]

We also measure the following superficial textual features:

Length: Number of words in the summary.

Compression ratio: The ratio of the number of words in the summary to the number of words in the reading passage.

Type-token ratio: The ratio of the number of unique words to the total number of words in the summary.

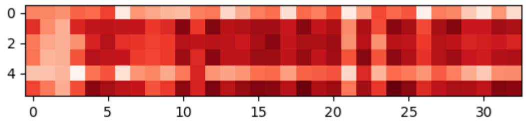
Text readability: The reading difficulty (the CEFR level) of the passage to be summarized.

After the features are extracted, we train a Support Vector Machine (SVM) (Cortes and Vapnik, 1995) model for the classification task (Section 3.1) and a Kernel Ridge Regression (KRR) (Saunders et al., 1998) model for the regression task (Section 3.2).

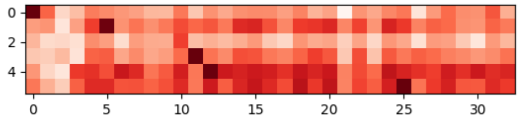
4.2 Assessing summary using similarity matrix

Secondly, we construct a sentence similarity matrix between the candidate summary and the original reading passage and apply a Convolutional Neural Network (CNN) model on the similarity matrix to predict the quality of the summary.

Lemaire et al. (2005) proposed a computational cognitive model for assessing extractive summarization. In their experiments, they presented 278 American school students with two reading passages and asked them to underline three to five sentences that they considered the most important in the texts. The underlined sentences were compared against the set of all the sentences from the original passage. They observed that the important sentences selected by the students are highly connected to the rest of the sentences in the text, where the connection is defined by the semantic similarity of the sentences.



(a) The similarity matrix of a good Summary A



(b) The similarity matrix of a bad Summary B

Figure 2: Similarity matrices of two summaries for the same reading passage from the simulated learner data. Summary A is a good summary and Summary B is a bad summary. The rows of the matrix represent sentences in the summary and the columns of the matrix represent sentences in the reading passage.

Based on their observations, we hypothesize that sentences in a good summary should have a well-distributed connection with as many sentences as possible in the original text, because a good summary is supposed to cover all the important information in the text. In contrast, sentences in a bad summary may fail to form a well-distributed connection with sentences in the original text. For example, if a bad summary only captures a few of the main points in the original text, then the sentences in such a summary would be connected only to the sentences where these points are mentioned in the original text, lacking the connections to the rest of the text. If a bad summary is generally irrelevant to the original text, sentences in such a summary would be minimally connected to most of the sentences in the original text. Beside these extreme cases on summary quality scale, summaries of intermediate quality may display patterns of connection to the original passage that share the characteristics of the good summary and the bad summary to various degrees.

Following this idea, we construct a sentence similarity matrix between the candidate summary and the original text. Each element of the matrix is a cosine similarity score between the vector representations of a sentence from the summary and a

---

[4] Features include: number of lexical chains per document, number of lexical chains normalized by text length, average/maximum lexical chain length, average/maximum lexical chain span, and the number of long chains.

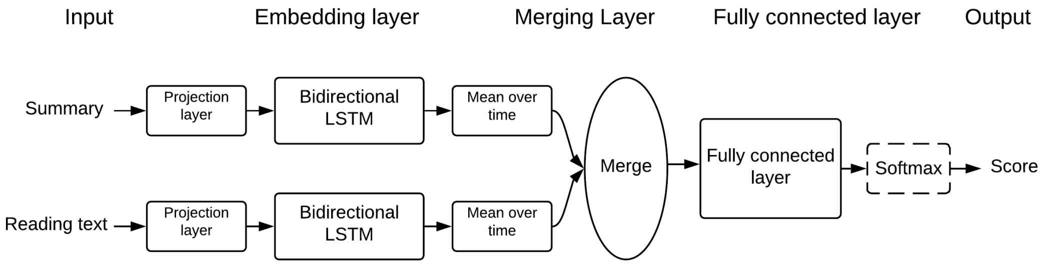


Figure 3: The merged LSTM model

sentence from the original text. We use the two sentence similarity models described in Section 4.1.2, skip-thought and average word embeddings, to build the sentence vectors.

According to our hypothesis, the quality of the summary corresponds to different patterns in the similarity matrix. The similarity matrix can be viewed as a heat map "image" from which we can learn patterns to detect the quality of the summary. Figure 2 demonstrates the similarity matrices of two summaries for the same reading passage from the simulated learner data. The shade of the coloured map indicates the degree of similarity between two sentences: the darker the shade is, the more similar the sentences are. In this example, Summary A is an example of a good summary, and Summary B is an example of a bad summary. We can see that sentences in Summary A are similar to a number of sentences in the original text, resulting in a well-distributed heat map. By contrast, sentences in Summary B are similar to five particular sentences in the text and are less similar to other sentences, which is reflected by the isolated dark patches in the image. On the whole, Summary A has higher similarity scores than Summary B, which makes its heat map darker. These two examples illustrate how different patterns may be observed in the heat map of the summaries of different quality.

To learn these patterns automatically, we apply a CNN model on the similarity matrix to predict the quality of the summary. However, it should be noted that CNNs usually work best when a large amount of training data is available, whereas the summary data we have collected represents a relatively small dataset. We compare the results of the CNN model against the feature extraction approach to investigate how well the model can learn from the limited amount of data.

4.3 Assessing summary using LSTM-based models

Thirdly, we experiment with several LSTM-based neural network models for assessing the summary quality. The LSTM-based models are used to learn representations of the summary and estimate its quality automatically, without having to manually extract features from it.

Recurrent neural networks with LSTM units (Hochreiter and Schmidhuber, 1997) have shown impressive results on various NLP tasks (Wang and Jiang, 2016; Rocktäschel et al., 2015). In essence, they are capable of embedding long text sequences into a vector representation which can later be decoded for use in various applications.

4.3.1 Merged LSTM model

Inspired by the recent advances with LSTMs in NLP tasks, we propose a merged LSTM model (see Figure 3) for assessing learner summaries. The merged LSTM model encodes the summary and the reading text separately with two bidirectional LSTMs, and merges the embedded summary and embedded reading text representations into a joint representation to predict the summary score. We explore four functions to merge the encoded vectors, including *concatenation*, *addition*, *dot product* and *linear combination*.

4.3.2 Attention-based LSTM model

As the merged LSTM model encodes the summary and reading text separately, it needs to propagate dependencies over long sequences to compare the summary and the text. The joint representation obtained in the merged LSTM model cannot fully capture the connection between the summary and the text. In this section, we propose an attention-based LSTM model which makes use of an attention mechanism to better model the relation between the summary and the reading text.

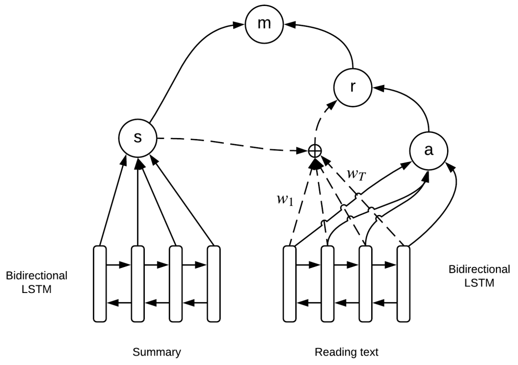


Figure 4: Attention mechanism architecture in the attention-based LSTM model for summary assessment

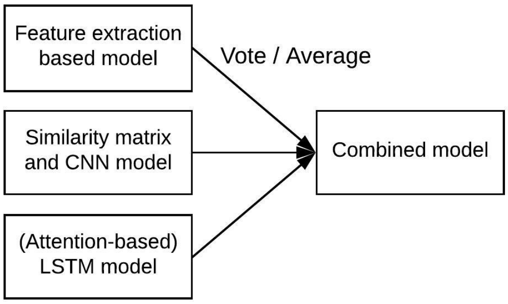


Figure 5: Combining three approaches using ensemble modelling

The attention mechanism was first introduced by Bahdanau et al. (2014) for machine translation. In general, the attention model learns a soft alignment between the input and the output in the encoder-decoder framework. The attention mechanism allows the model to learn what to attend to in the input states and mitigates the long-dependency bottleneck of the LSTM.

In the attention-based model for summary assessment, the original text and the summary are still encoded using two separate LSTMs. However, the text representation is formed by a weighted sum of the hidden states of the text encoder, where the weights can be interpreted as the degree to which the summary attends to a particular token in the text. The summary representation and the text representation are combined with a nonlinear function into a joint representation and then fed into the fully connected layer to predict the summary quality. Figure 4 is an illustration of the attention mechanism between the embedded summary and the embedded input text.

Mathematically, suppose s is the encoded summary vector and $a(t)$ is the hidden state of the LSTM for the text at each token t. Then the final representation r of the encoded text is a weighted sum of $a(t)$:

$$r = a \cdot w = \sum_{t=1}^{T} a(t)w(t)$$

The weight for each token $w(t)$ is computed by:

$$w(t) = \frac{exp(\alpha(t))}{\sum_{t=1}^{T} exp(\alpha(t))}$$

where

$$\alpha(t) = W_{a\alpha} \cdot a(t) + W_{s\alpha} \cdot s$$

is an alignment model.

The joint representation m of the summary and the text is a combination of the summary vector s and the weighted input text vector r.

$$m = tanh(W_{sm} * s + W_{rm} * r + b)$$

where W_{sm}, W_{rm} and b are the parameters of a linear combination function.

4.4 Ensemble modelling

Ensemble modelling combines several machine learning techniques into one model in order to improve the stability and accuracy of the prediction. We explore combining the three different models (see Figure 5) into a single model by taking the majority vote from the binary classification models and taking the average value of the predicted scores from the regression models. We compare the performance of the combined models against the individual models to investigate if and to what extent ensemble modelling is useful for assessing the summaries.

5 Experiments and Results

5.1 Experimental set-up

We evaluate our models on the real learner data and on the simulated learner data, for consistency, using 5-fold cross validation. In each fold, 60% of the data is used as the training set, 20% as the development set, and 20% as the test set.[5]

We compare our models against five baselines: *most frequent baseline*, *random baseline*, *ROUGE*

---

[5] We choose the best model based on the development set, retrain the selected model on the combination of the training and the development data, and evaluate the model on the test set.

Models	Variants			Accuracy
Baseline	Baseline type		most-frequent	50.0%
			random	50.0%
			ROUGE	59.3%
			BLEU	51.7%
			ROUGE + BLEU	59.6%
SVM	reference type		random	58.8%
			TextRank	63.8%
			MEAD	62.9%
			original text	**65.6%**
CNN	similarity matrix type		avg word embeddings	**65.8%**
			skip-thought vectors	63.4%
LSTM	Merged LSTM	merging function	concatenation	68.0%
			addition	68.1%
			multiplication	69.1%
			linear combination	70.4%
	Attention LSTM			**71.1%**
Combined model	SVM+CNN+LSTM			**75.3%***

Table 2: Model performance on the simulated learner data. We use the bold font to indicate the best model for each method. The asterisk sign indicates the best performance across all models.

baseline,[6] *BLEU baseline*, and *ROUGE + BLEU baseline*.

We use accuracy to evaluate the models on the simulated learner data, and on the real learner data, we report scores of two evaluation metrics: Pearson correlation coefficient (PCC) and Root Mean Square Error (RMSE), which are commonly used for evaluating regression models.

5.2 Results

Table 2 shows the results of the baseline and the four types of models on the simulated learner data, and Table 3 reports the results of the models on the real learner data.

On the simulated learner data, the best variants from all three methods outperform the baselines. The improvement is statistically significant ($p<0.05$) using t-test for all three methods. We combine the best variants from the three approaches into a single system by taking the majority vote from the models. The resulting system achieves the best accuracy of 75.3% in predicting the binary type of the summary on the simulated learner data.

On the real learner data, we found that the feature extraction-based model outperforms the CNN model and LSTM-based models, which also significantly outperform the baselines. The results suggest that the neural network-based models are not as effective as the traditional feature extraction-based method for the regression task, at least when the training data is limited in size.

However, although the CNN and LSTM models are not the best-performing models individually, a combination of the three methods (KRR, CNN and LSTM) still improves the performance. We believe that this is because the three independent models capture different aspects of the summary quality that are complementary to each other. In addition, the combined model is more robust to outliers. For example, when two models agree on an instance while the third model does not, the combined model will select the majority vote or the average score of the model estimations, hence achieving a better performance in estimating the summary quality. Overall, the best model performance is close to human performance.

We also observe that when assessing the summaries with extracted features, using the original document as the reference works better than using other types of reference. This might be because the extractive summarizers only select sentences that are highly related to others, where the relation

---

[6] A baseline trained on ROUGE features only.

Models	Variants			PCC	RMSE
Baseline	Baseline type		most-frequent	-	1.30
			random	0.011	1.79
			ROUGE	0.499	1.12
			BLEU	0.208	2.88
			ROUGE + BLEU	**0.499**	**1.11**
KRR	reference type		random	0.517	1.11
			TextRank	0.576	1.06
			MEAD	0.557	1.08
			original text	**0.636**	**0.99**
CNN	similarity matrix type		avg word embeddings	**0.504**	**1.12**
			skip-thought vectors	0.458	1.14
LSTM	Merged LSTM	merging function	concatenation	0.487	1.13
			addition	0.466	1.13
			multiplication	0.490	1.12
			linear combination	0.484	1.13
	Attention LSTM			**0.494**	**1.12**
Combined model	KRR+CNN+LSTM			**0.665***	**0.97***

Table 3: Results of the regression model performance on the learner data. We use the bold font to indicate the best model for each method. The asterisk sign indicates the best performance across all models.

is typically judged by the word overlap, therefore missing the bits of text where topical words occur less often.

6 Conclusion

In this paper, we introduce a summarization task for testing reading comprehension of learners and present several automated systems to assess the quality of the learner summary. We collected summaries from members of our university and from the real learners to evaluate our systems. We propose and compare three approaches to assess the summaries, including the feature extraction-based model, the CNN-based model using similarity matrix, and the LSTM-based model. The best system, built using a combination of three models, yields an accuracy of 75.3% on the simulated learner data, and $PCC = 0.665$, $RMSE = 0.97$ on the real learner data. Although not directly comparable to other studies, we note that these results are higher than those reported in previous work.

Our systems are generalizable and address the limitations of the previous research in this area as: (1) they are capable of evaluating the quality of a summary without the need of being trained on each input text separately, (2) they do not require a pre-defined reference, and (3) they are capable of capturing content similarity beyond verbatim overlap, taking into account paraphrasing and higher levels of abstraction.

We believe that although the application presented in this paper focuses on assessing learner summaries, these techniques may also be useful for benchmarking automated summarization systems. Evaluation of these techniques for benchmarking automated summarization systems is one direction for our future research.

We make the summary data available at `https://www.cl.cam.ac.uk/~ek358/learner-summaries.html`.

Acknowledgments

This paper reports on research supported by Cambridge Assessment, University of Cambridge. We also thank Cambridge Assessment for their assistance in the collection of the real learner data. We are grateful to the anonymous reviewers for their valuable feedback.

References

J. Charles Alderson. 2005. *Assessing reading*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91. Association for Computational Linguistics.

Michel Galley and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *International Joint Conference on Artificial Intelligence*, volume 3, pages 1486–1488.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Thomas K. Landauer. 2006. *Latent semantic analysis*. Wiley Online Library.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, volume 14, pages 1188–1196.

Benoit Lemaire, Sonia Mandin, Philippe Dessus, and Guy Denhiere. 2005. Computational cognitive models of summarization assessment skills. In *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*, pages 1266–1271.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, volume 8, pages 74–81. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 71–78. Association for Computational Linguistics.

Nitin Madnani, Jill Burstein, John Sabatini, and Tenaha OReilly. 2013. Automated scoring of a summary writing task designed to measure reading comprehension. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–168.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1, pages 775–780.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*, 4(2):4.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, and Danyu Liu. 2004. MEAD - A platform for multi-document multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kovcisky, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. Association for Computational Linguistics.

Stefan Ruseti, Mihai Dascalu, Amy M. Johnson, Danielle S. McNamara, Renu Balyan, Kathryn S. McCarthy, and Stefan Trausan-Matu. 2018. Scoring Summaries Using Recurrent Neural Networks. In *Intelligent Tutoring Systems*, pages 191–201, Cham. Springer International Publishing.

Craig Saunders, Alexander Gammerman, and Volodya Vovk. 1998. Ridge Regression Learning Algorithm in Dual Variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 515–521.

David Wade-Stein and Eileen Kintsch. 2004. Summary Street: Interactive computer support for writing. *Cognition and Instruction*, 22(3):333–362.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Cyril J. Weir, Ivana Vidakovic, and Evelina Galaczi. 2013. *Measured constructs*. Cambrdige University Press.

Denise Whitelock, Debora Field, Stephen Pulman, John T. E. Richardson, and Nicolas Van Labeke. 2013. OpenEssayist: an automated feedback system that supports university students as they write summative essays. In *Proceedings of the 1st International Conference on Open Learning: Role, Challenges and Aspirations*.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Building Educational Applications Using NLP*, pages 12–22. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27.

Data-efficient Neural Text Compression with Interactive Learning

Avinesh P.V.S and **Christian M. Meyer**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
`www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de`

Abstract

Neural sequence-to-sequence models have been successfully applied to text compression. However, these models were trained on huge automatically induced parallel corpora, which are only available for a few domains and tasks. In this paper, we propose a novel interactive setup to neural text compression that enables transferring a model to new domains and compression tasks with minimal human supervision. This is achieved by employing active learning, which intelligently samples from a large pool of unlabeled data. Using this setup, we can successfully adapt a model trained on small data of 40k samples for a headline generation task to a general text compression dataset at an acceptable compression quality with just 500 sampled instances annotated by a human.

1 Introduction

Text compression is the task of condensing one or multiple sentences into a shorter text of a given length preserving the most important information. In natural language generation applications, such as summarization, text compression is a major step to condense the extracted important content of the source documents. But text compression can also be applied in a wide range of related applications, including the generation of headlines (Filippova et al., 2015), captions (Wubben et al., 2016), subtitles (Vandegehinste and Pan, 2004; Luotolahti and Ginter, 2015), and the compression of text for small screens (Corston-Oliver, 2001).

Neural *sequence-to-sequence* (Seq2Seq) models have shown remarkable success in many areas of natural language processing and specifically in natural language generation tasks, including text compression (Rush et al., 2015; Filippova et al., 2015; Yu et al., 2018; Kamigaito et al., 2018). Despite their success, Seq2Seq models have a major drawback, as they require huge parallel corpora with pairs of source and compressed text to be able to learn the parameters for the model. So far, the size of the training data has been proportional to the increase in the model's performance (Koehn et al., 2003; Suresh, 2010), which is a major hurdle if only limited annotation capacities are available to manually produce a corpus. That is why existing research employs large-scale automatically extracted compression pairs, such as the first sentence and the presumably shorter headline of a news article. However, such easy-to-extract source data is only available for a few tasks, domains, and genres and the corresponding models do not generalize well from the task of headline generation to other text compression tasks.

In this paper, we propose an *interactive setup* to neural text compression, which learns to compress based on user feedback acquired during training time. For the first time, we apply *active learning* (AL) methods to neural text compression, which greatly reduces the amount of the required training data and thus yields a much more data-efficient training and annotation workflow. In our experiments, we find that this approach enables the successful transfer of a model trained on headline generation data to a general text compression task with a minimum of parallel training instances.

The objective of AL is to efficiently select unlabeled instances that a user should annotate to advance the training. A key component of AL is the choice of the *sampling strategy*, which curates the samples in order to maximize the model's performance with a minimum amount of user interaction. Many AL sampling strategies have proven effective for human-supervised natural language processing tasks other than compression (Hahn et al., 2012; Peris and Casacuberta, 2018; Liu et al., 2018).

In our work, we exploit the application of uncertainty-based sampling using attention disper-

Proceedings of NAACL-HLT 2019, pages 2543–2554
Minneapolis, Minnesota, June 2 - June 7, 2019.

sion and structural similarity for choosing samples to be annotated for our interactive Seq2Seq text compression model. We employ the AL strategies for (a) learning a model with a minimum data, and (b) adapting a pretrained model with few user inputs to a new domain.

In the remaining paper, we first discuss related work and introduce the state-of-the-art Seq2Seq architecture for the neural text compression task. Then, we propose our novel interactive compression approach and demonstrate how batch mode AL can be integrated with neural Seq2Seq models for text compression. In section 4, we introduce our experimental setup, and in section 5, we evaluate our AL strategies and show that our approach successfully enables (a) learning the Seq2Seq model with a minimum of data, (b) transfer of a pretrained headline generation model to a new compression task and dataset with minimal user interaction. To encourage further research and enable reproducing our results, we publish our code as open-source software.[1]

2 Related Work

In this section, we discuss related work to our research concerning: (1) neural text compression models, (2) existing text compression corpora and (3) active learning for neural models.

Neural text compression. Neural text compression models can be broadly classified into two categories: (a) deletion-based extractive models and (b) abstractive models. The goal of the deletion-based models is to delete unimportant words from a source text to generate a shorter version of the text. In contrast, abstractive models generate a shorter text by inserting, reordering, reformulating, or deleting words of the source text.

Previously, deletion-based extractive methods explored various modeling approaches, including the noisy-channel model (Knight and Marcu, 2002; Turner and Charniak, 2005), integer linear programming (Clarke and Lapata, 2007), variational autoencoders (Miao and Blunsom, 2016), and Seq2Seq models (Filippova et al., 2015). Similarly, recent abstractive models have seen tree-to-tree transduction models (Cohn and Lapata, 2013) and variations of Seq2Seq models, such as attention (Rush et al., 2015), attentive long short-term memory (LSTM) models (Wubben et al., 2016) and operation networks where the Seq2Seq model decoder is replaced with a deletion decoder and a copy-generate decoder (Yu et al., 2018).

Filippova et al. (2015) show that Seq2Seq models without any linguistic features have the ability to delete unimportant information. Kamigaito et al. (2018) incorporate higher-order dependency features into a Seq2Seq model and report promising results. Rush et al. (2015) propose an attention-based Seq2Seq model for generating headlines. Chopra et al. (2016) further improve this task with recurrent neural networks. Although Seq2Seq models show state-of-the-art results on different compression datasets, there is yet no work which investigates whether large training corpora are needed to train neural compression models and if there are efficient ways to train and adapt them to other datasets with few annotations.

Text compression corpora. Early publicly available text compression datasets are manually curated but small (Cohn and Lapata, 2008; Clarke and Lapata, 2006, 2008). These datasets are typically used by unsupervised approaches as they are 200 times smaller in size compared to the annotated data used for training state-of-the-art supervised approaches. Filippova and Altun (2013) introduce an extractive compression dataset of 250k headline and first sentence compression pairs based on Google News, which they use for training a supervised compression method. Similarly, Rush et al. (2015) create another large abstractive dataset of 4 million headline and first sentence compression pairs from news articles extracted from the Annotated Gigaword corpus (Napoles et al., 2012). Although these datasets are large, they predominantly address headline generation for news.

Creating such large corpora manually for a new task or domain is hard. Toutanova et al. (2016) pioneered the manual creation of a multi-reference compression dataset MSR-OANC with 6k sentence–short paragraph pairs from business letters, newswire, journals, and technical documents sampled from the Open American National Corpus[2]. They provide five crowd-sourced rewrites for a fixed compression ratio and also acquire quality judgments. This dataset covers multiple genres compared to the large automatically collected compression datasets, and Toutanova et al. (2016) show that neural Seq2Seq

---

[1] `https://github.com/UKPLab/NAACL2019-interactiveCompression`

[2] `https://www.anc.org/data/oanc`

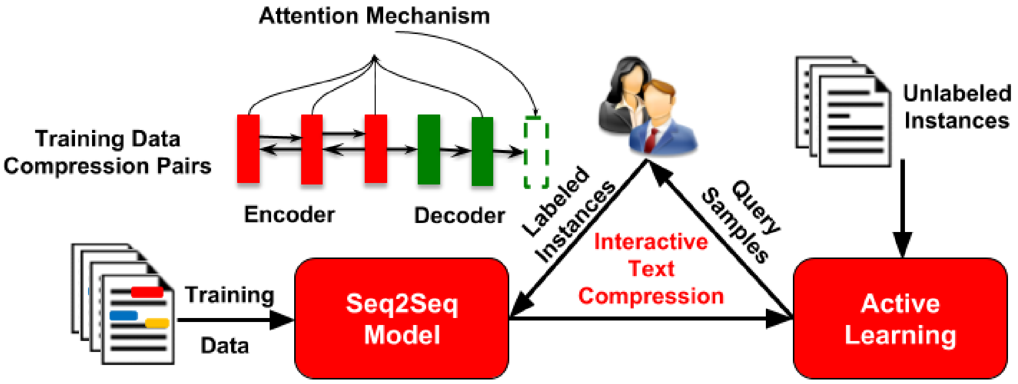


Figure 1: Pipeline of our interactive text compression model. The pipeline is divided into three main components: (1) Neural Seq2Seq text compression model, (2) active learning, and (3) interactive text compression

models trained on headline generation datasets fail to achieve state-of-the-art results as compared to an ILP-based unsupervised method. In our work, we go beyond that and investigate strategies to easily adapt pretrained models to such small datasets employing minimal user input.

Active learning for neural models. AL has been successfully applied to various natural language processing tasks, including corpus annotation (Hahn et al., 2012; Yan et al., 2011), domain adaptation (Chan and Ng, 2007), personalized summarization (P. V. S. and Meyer, 2017), machine translation (Haffari and Sarkar, 2009), language generation (Mairesse et al., 2010), and many more. Only recently, it has been applied to neural models: Wang et al. (2017a) propose an AL approach for a black box semantic role labelling (SRL) model where the AL framework is an add-on to the neural SRL models. Peris and Casacuberta (2018) use AL in neural machine translation. They propose quality estimation sampling, coverage sampling, and attention distraction sampling strategies to query data for interactive machine translation. Liu et al. (2018) additionally propose an AL simulation trained on a high-resource language pair to transfer their model to low-resource language pairs. In another line of research, Sener and Savarese (2018) discuss a core-set AL approach as a batch sampling method for neural image classification based on convolutional neural networks. Although AL techniques have been widely used in natural language processing, to our knowledge, there is yet no work on the use of AL for neural text compression. We fill this gap by putting the human in the loop to learn effectively from a minimal amount of interactive feedback and for the first time, we explore this data-efficient AL-based approach to adapt a model to a new compression dataset.

3 Approach

To address this research problem, we first describe the neural Seq2Seq text compression models we use. Then, we introduce our active learning strategies to select the training samples interactively for in-domain training as well as for domain adaptation, and we describe a novel interactive neural text compression setup. Figure 1 illustrates the main components of our system.

3.1 Neural Seq2Seq Text Compression

In this work, we employ state-of-the-art Seq2Seq models with attention (Seq2Seq-gen) (Rush et al., 2015) and pointer-generated networks with coverage (Pointer-gen) (See et al., 2017) as our base models, which we use for our AL-based interactive text compression setup.

Both Seq2Seq models are built upon the encoder-decoder framework by Sutskever et al. (2014). The encoder encodes the input sequence $\mathrm{x} = (x_1, x_2.., x_n)$ represented by an embedding matrix into a continuous space using a bidirectional LSTM network and outputs a sequence of hidden states. The decoder is a conditional bidirectional LSTM network with attention distribution (Luong et al., 2015)

$$a_i^j = \frac{\exp(e_i^j)}{\sum_{k=1}^{n} \exp(e_k^j)} \quad (1)$$

where e_i^j is computed at each generation step j with the encoder states h_i^{enc} and the decoder states h_j^{dec}:

$$e_i^j = q \cdot \tanh(W_h^{\text{enc}} h_i^{\text{enc}} + W_h^{\text{dec}} h_j^{\text{dec}} + b_{\text{att}}) \quad (2)$$

where q , W_h^{enc}, W_h^{dec} and b_{att} are learnable parameters. The attention distribution a_i^j is used to compute the weighted sum of the encoder hidden states, also known as the context vector

$$c_j^* = \sum_i^n a_i^j h_i^{\text{enc}} \tag{3}$$

To obtain the vocabulary distribution P_j^{vocab} at generation step j, we concatenate the fixed context vector with the decoder state h_j^{dec} and pass it through two linear layers:

$$P_j^{\text{vocab}} = \text{softmax}(W_v(W_v'[h_j^{\text{dec}}; c_j^*] + b_v') + b_v) \tag{4}$$

where W_v, W_v', b_v and b_v' are learnable parameters. P_j^{vocab} is a probability distribution over all words in the vocabulary V. Based on the vocabulary distribution, the model generates the target sequence $\text{y} = y_1, y_2, \ldots, y_m$, $m \leq n$ with

$$y_j = \text{argmax}_w P_j^{\text{vocab}}(w),\ w \in V \tag{5}$$

for each generation step j.

Finally during training, we define the loss function for generation step j as the negative log likelihood of the target word y_j and the overall loss function for the target word sequence as $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{m} \sum_{j=0}^{m} - \log P_j^{\text{vocab}}(y_j) \tag{6}$$

Another state-of-the-art approach we use for our experiments is the pointer-generator networks (Pointer-gen) proposed by See et al. (2017). This model uses a pointer-generator network that determines a probability function to generate the words from the vocabulary V or copy the words from the source text by sampling from the attention distribution a_i^j as shown in Eq. 8. The model achieves this by calculating an additional generation probability p_{gen} for generation step j, which is calculated from the context vector c_j^*, the decoder state h_j^{dec}, and the current input to the decoder x_j':

$$p_{\text{gen}} = \sigma(W_c^T c_j^* + W_{h^{\text{dec}}}^T h_j^{\text{dec}} + W_{x'}^T x_j' + b_{\text{gen}}) \tag{7}$$

$$P_j(w) = p_{\text{gen}} P_j^{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i=0}^{n} a_i^j \tag{8}$$

where vectors W_c, $W_{h^{\text{dec}}}$, $W_{x'}$, b_{gen} are learnable parameters, n is the number of words in the source text and σ is the sigmoid function.

The model also uses an extra feature of coverage to keep track of words generated by the model to discourage repetition. In the coverage model, a coverage vector is calculated which is the sum of the attention distribution across all the previous decoding steps and it is passed on as an extra input to the attention mechanism:

$$c_i^j = \sum_{k=0}^{j-1} a_i^k \tag{9}$$

$$e_i^j = q \cdot \tanh(W_h^{\text{enc}} h_i^{\text{enc}} + W_h^{\text{dec}} h_j^{\text{dec}} + W_c c_i^j + b_{\text{att}}) \tag{10}$$

where W_c is an additional learnable parameter.

3.2 Active Learning

Toutanova et al. (2016) show that Seq2Seq models, which perform well on large news headline generation datasets, fail to achieve good performance on their MSR-OANC multi-genre compression dataset. A major issue with training Seq2Seq models is the lack of domain-specific data and the expensive process to create parallel compression pairs. It is therefore indispensable to minimize the cost of data annotation. Thus, AL comes into play whose key element is to find a strategy for selecting samples the user should annotate which yield a more efficient training process. For text compression, we suggest AL strategies to maximize the model's coverage and the diversity of the samples. To this end, we build upon work in uncertainty sampling by (Peris and Casacuberta, 2018; Wang et al., 2017b) and propose a new strategy to predict the sample diversity at a structural level.

Coverage constraint sampling (Coverage-AL). An important factor on which text compression models are evaluated is the coverage (Marsi et al., 2010). Coverage can be defined as the text compression models being able to learn the deletion or generation rules from the training samples and apply them on an input source text. Wu et al. (2016) first proposed the idea of using attention weights to calculate coverage penalty for active learning based machine translation systems. The attention weights were further extended by Peris and Casacuberta (2018) to estimate an attention dispersion based uncertainty score for a sentence. The idea of attention dispersion is that if the neural Seq2Seq compression model is uncertain then the attention weights will be dispersed across the source text while generating the target words. The samples with higher dispersion will have their attention weights uniformly distributed across the source sentences. Thus, the goal is to find the samples with high uncertainty based on attention dispersion. As we want to define to the extent to which the attention distribution differs from a normal distribution we propose to use a skewness score. The skewness score calculates the attention dispersion while decoding a target word y_j.

$$\text{skewness}(y_j) = \frac{\frac{1}{n}\sum_{i=1}^{n}(a_i^j - \frac{1}{n})^3}{(\frac{1}{n}\sum_{i=1}^{n}(a_i^j - \frac{1}{n})^2)^{3/2}} \quad (11)$$

a_i^j is the attention weight assigned by the attention layer to the i-th source word when decoding the j-th target word and $\frac{1}{n}$ is the mean of the attention weights of the target word y_j.

The skewness for a normal distribution is zero, and since we are interested in the skewness of samples with heavy tails, we take the negative of the skewness averaged across all target words to obtain the uncertainty coverage score C_{score}.

$$C_{\text{score}}(\mathrm{x}, \mathrm{y}) = \frac{\sum_{j=1}^{m} -\text{skewness}(y_j)}{m} \quad (12)$$

where m is the number of target words.

Diversity constraint sampling (Diversity-AL). Diversity sampling methods have been used in information retrieval (Xu et al., 2007) and image classification (Wang et al., 2017b). The core idea is that samples that are highly similar to each other typically yield little new information and thus low performance. Similarly, to increase the diversity of the samples in neural text compression, we propose a novel scoring metric to measure the diversity of multiple source texts at a structural level. Our intuition is that integrating part-of-speech, dependency and named entity information is useful for text compression, e.g., to learn which named entities are important and how to compress a wide range of phrase types and syntactically complex sentences. Thus, we consider part of speech tags, dependency trees, and named entity embeddings and calculate the structural similarity of the source text with regard to the target text. We use a multi-task convolutional neural network similar to Søgaard and Goldberg (2016) trained on OntoNotes and Common Crawl to learn the structural embeddings consisting of tag, dependency and named entity embeddings. The diversity score D_{score} is calculated using the cosine distance between the average of the structural embeddings of the words in the source sentence and the average of the structural embeddings of the words in the target compression as in Eq. 13:

$$D_{\text{score}}(\mathrm{x}, \mathrm{y}) = \frac{\mathrm{E}_{\text{struc}}(\mathrm{x}) \cdot \mathrm{E}_{\text{struc}}(\mathrm{y})}{||\mathrm{E}_{\text{struc}}(\mathrm{x})|| \cdot ||\mathrm{E}_{\text{struc}}(\mathrm{y})||} \quad (13)$$

where $\mathrm{E}_{\text{struc}}(\cdot)$ is the average structural embedding of a text.

These AL sampling strategies are applied interactively while training to make better use of the data by selecting the most uncertain instances. Additionally, both strategies can be applied for domain adaptation by actively querying user annotations for a domain-specific dataset in an interactive text compression setup, which we describe next.

3.3 Interactive Text Compression

In this subsection, we introduce our interactive text compression setup. Our goal is to select the batch of samples for training efficiently with minimal samples and to become able to transfer the models to new datasets for different domains and genres with few labeled data.

We consider an initial collection of parallel instances $D = \{(\mathrm{x}_i, \mathrm{y}_i) \mid 1 \leq i \leq N\}$ consisting of pairs of input text x_i and their corresponding compression y_i. Additionally, we consider unlabeled instances $D' = \{\mathrm{x}_i \mid i > N\}$, for which we only know the uncompressed source texts. Our goal is to sample sets of unlabeled instances $S_t \subset D'$ which should be annotated by a user in each time step t. The interactive compression model can only see the labeled pairs from the initial dataset D in the beginning, but then incrementally learns from the user annotations.

Algorithm 1 provides an overview of our interactive compression setup. The inputs are the labeled compression pairs D and the unlabeled source texts D'. D is used to initially train the neural text compression model M. In line 5, we start the interactive feedback loop iterating over $t = 0, \ldots, T$. We first sample a set of unlabeled source texts S_t (line 6) by using our AL strategies introduced in section 3.2 and then loop over each of the unlabeled samples to be annotated or supervised by the human in line 10. As the user feedback in the current time step of sample S_t, we obtain the compressions Y_t of the sampled source texts S_t from the user and use them for online training of the model M. After T iterations or if there are no samples left for querying (i.e., $S_t = \emptyset$), we stop the iteration and return the updated Seq2Seq model M.

Algorithm 1 Interactive Text Compression

```
 1: procedure INTERACTIVECOMPRESSION()
 2:     input: Text Compression Pairs $D$,
 3:            Unlabeled Text $D'$
 4:     $M \leftarrow$ learnSeq2Seq($D$)
 5:     for $t = 0, ..., T$ do
 6:         $S_t \leftarrow$ getSample($D'$)
 7:         if $S_t = \emptyset$ then
 8:             return $M$
 9:         else
10:             $Y_t \leftarrow$ queryUser($S_t$)
11:             $M \leftarrow$ update($M, S_t, Y_t$)
12:             $D' \leftarrow D' - S_t$
13:         end if
14:     end for
15: end procedure
```

4 Experimental Setup

4.1 Data

For our experiments, we use the large Google News text compression corpus[3] by Filippova and Altun (2013), which contains 250k automatically extracted the deletion-based compressions from aligned headlines and first sentences of news articles. Recent studies on text compression have extensively used this dataset (e.g., Zhao et al., 2018; Kamigaito et al., 2018). We carry out in-domain active learning experiments on the Google News compression corpus.

To evaluate our interactive setup, we adapt the trained models to the MSR-OANC text compression corpus by Toutanova et al. (2016), which contains 6k crowdsourced multi-genre compressions from the Open American National Corpus. This corpus is well-suited to evaluate our interactive setup, since it is sourced from mixture of newswire, letters, journals, and non-fiction genres, in contrast to the Google News corpus covering only newswire.

Dataset	# Train	# Dev	# Test
Google News	195,000	5,000	10,000
MSR-OANC	5,000	448	785

Table 1: Statistics of the compression datasets

For evaluating the compressions against the reference compressions, we use a Python wrapper[4] of the ROUGE metric (Lin, 2004) with the parameters suggested by Owczarzak et al. (2012) yielding high correlation with human judgments (i.e., with stemming and without stopword removal).[5]

4.2 Preprocessing and Parameters

To preprocess the datasets, we perform tokenization. We obtain the structural embeddings for a sentence using spaCy[6] embeddings learned using a multi-task convolutional neural network.

To evaluate and assess the effectiveness of our active learning-based sampling approaches, we set up our interactive text compression approach for the two state-of-the-art Seq2Seq models consisting of a generative model (Seq2Seq-gen) and a generate-and-copy model (Pointer-gen) as described in Section 3.1. For the neural Seq2Seq text compression experiments, we set the beam size and batch size to 10 and 30 respectively. We use the Adam optimizer (Kingma and Ba, 2015) for the gradient-based optimization. Finally, the parameters for the neural network parameters like weights and biases are randomly initialized.

In order to asses the effectiveness of AL for neural text compression we extend the OpenNMT[7] implementations with our interactive framework following Algorithm 1. The sampling strategy selects instances to be annotated interactively by the user in batches. Next, the neural text compression model is incrementally updated with the selected samples.

Due to the presence of a human in the loop, it typically demands real user feedback, but the cost of collecting sufficient data for various settings of our models is prohibitive. Thus in our experiments, the users were simulated by using the com-

---

[3] `https://github.com/google-research-datasets/sentence-compression`

[4] `https://github.com/pltrdy/files2rouge`

[5] -n 2 -c 95 -r 1000 -a -m

[6] `https://spacy.io/`

[7] `https://github.com/OpenNMT/OpenNMT-py`

Methods	UB			Random			Coverage-AL			Diversity-AL		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
Seq2Seq-gen	59.94	52.08	59.78	61.60	50.03	61.37	**62.89**	**51.38**	**62.56**	62.54	50.19	62.13
Pointer-gen	79.26	71.77	79.08	71.61	61.15	71.28	**78.11**	**70.50**	**77.89**	77.45	70.30	77.38

Table 2: ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) achieved by the state-of-the-art models using our sampling strategies evaluated on the Google compression test set. Bold marks best AL strategy.

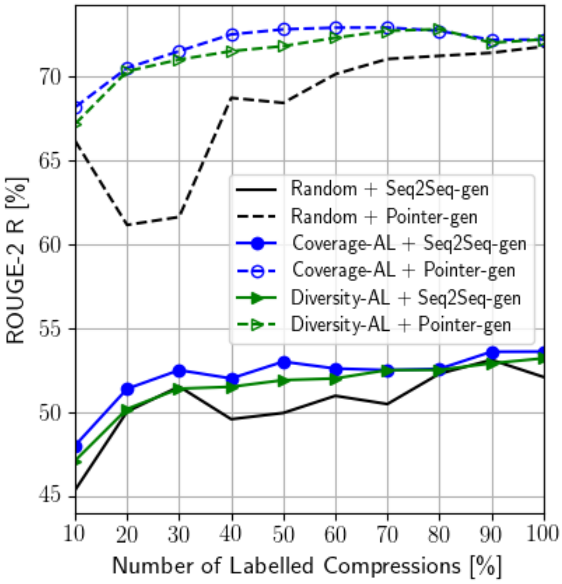


Figure 2: Analysis of the active learning approaches combined with state-of-the-art Seq2Seq compression models on Google compression dataset while varying the training sizes.

pression pairs from our corpus as the sentences annotated by the user.

5 Results and Analysis

Our experiments address two main research questions for in-domain training and domain adaptation of neural text compression:

- Which active learning strategies are useful in text compression to select training samples such that higher performance can be achieved with a minimum of labeled instances?
- Which instances are to be annotated interactively by the user such that the model adapts quickly to a new dataset?

In-domain Active Learning. For in-domain active learning experiments, we choose the Google News text compression training corpus and sample for corpus sizes between 10% and 100% in ten percent point steps. As a baseline, we use a random sampling strategy to test the state-of-the-art Seq2Seq neural text compression models. Figure 2 suggests that our coverage-based sampling (Coverage-AL) and diversity-based sampling (Diversity-AL) strategies outperform the random sampling strategy throughout all training sizes. A key observation is that our sampling strategies are behind the upper bound by just 0.5% ROUGE-2 when only 20% of the training data is used. Table 2 illustrates the results of our sampling strategies when 20% of the data is used for training. All the results are in comparison to the upper bound (UB) receiving 100% of the training data.

Coverage-AL performs better than the Diversity-AL for both the Seq2Seq-gen and Pointer-gen models. However, they are still not effective in the Seq2Seq-gen model where random sampling performs on par with the active learning sampling approaches. We believe this is due to the Seq2Seq-gen model's inability to copy from the source text in the sampled set as a consequence of active learning in the batch setting. Whereas for Pointer-gen model, we observed that both Coverage-AL and Diversity-AL strategies of adding new samples for training had a greater impact when the model has not adapted. We attribute the effectiveness of the Coverage-AL strategy over Diversity-AL to the exploitation of the model uncertainty, as the Diversity-AL only uses the similarity based on the samples, but misses to integrate the model uncertainty.

Table 3 presents an example sentence compression pair from the Google News dataset and the generated compressions of both neural Seq2Seq models when using one of the three sampling strategies. The example shows that detailed descriptions like the names of the ships “JING GANGSHA” and “HENG SHUI” are dropped by all models. In particular, the Seq2Seq-gen model has the problem of generating words not present in the original text (e.g.,

Source text:	Two Chinese war ships , “ JING GANGSHA ” and “ HENG SHUI ” arrived at the port of Trincomalee on 13 th January 2014 on a good will visit .
Reference:	Two Chinese war ships , arrived at the port of Trincomalee will visit .
Seq2Seq-gen	
+ Random:	Two Chinese war ships , arrived at the port of toddlers on 13 th January 2014 .
+ Coverage-AL:	Two Chinese war ships , arrived at the port of Trincomalee on a good will visit .
+ Diversity-AL:	Two Chinese war ships arrived at the port of Scottsbluff on 13 th .
Pointer-gen	
+ Random:	Two Chinese war ships , arrived at the port of Trincomalee on 13 th January 2014 .
+ Coverage-AL:	Two Chinese war ships arrived at the port of Trincomalee will visit .
+ Diversity-AL:	Two Chinese war ships , arrived at the port of Trincomalee .

Table 3: In-domain active learning example sentence and compressions for Google News compression dataset when using 20% of labelled compressions with Random, Coverage-AL, Diversity-AL sampling strategies

Methods	MSR-OANC ID			Random			Coverage-AL			Diversity-AL		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
Seq2Seq-gen	30.05	10.42	26.87	33.51	13.60	30.26	**35.10**	**15.00**	**32.78**	34.85	14.92	32.41
Pointer-gen	35.24	16.57	32.56	38.19	21.87	37.94	**39.59**	**24.87**	**37.02**	39.42	24.70	36.86

Table 4: ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) achieved by the state-of-the-art models using our sampling strategies when interactively retrained using 10% of the MSR-OANC training set. The results are in comparison to the models trained on in-domain training set (MSR-OANC ID). Bold marks best AL strategy.

“toddlers”, “Scottsbluff”). In contrast, the Pointer-gen model’s ability to copy from the original text restrains the model from generating irrelevant words. Although Diverysity-AL based models recognized the phrasal constructs crucial for the sentence meaning, Coverage-AL generated the closest compression to the reference.

Active learning for domain adaptation. To test our interactive Seq2Seq model using active learning strategies for the domain adaptation scenario, we train the model on the Google News compression corpus and test it on the multi-genre MSR-OANC compression dataset. Additionally, for domain adaptation, the neural Seq2Seq model is updated incrementally using our interactive compression Algorithm 1. The sampling strategies select the instances to be interactively annotated by the user. As the cost of interactive experimentation with real users, we use simulated feedback from the labeled sentence compressions from the MSR-OANC training data. The two sampling strategies used for in-domain active learning are used for interactive compression with the state-of-the-art Seq2Seq models. Table 4 illustrates the results of the interactive text compression model when applied to the MSR-OANC text compression dataset. One interesting observation is the fact that our sampling strategies at 10% of the training data ($\approx$ 500 samples) perform better than models trained on in-domain training data (MSR-OANC ID) with 5k training instances by +8.3% and +8.2% ROUGE-2.

Figure 3 shows the results for the various sample sizes of the 5k training instances. The results show a similar trend as the active learning for the interactive data-selection scenario. The Coverage-AL and Diversity-AL strategies do not show significant differences from each other. However, the two active learning strategies achieve on average +2.5% ROUGE-2 better results than the random sampling. The results demonstrate that the use of relevant training samples is useful for transferring the models to new domains and genres.

Table 5 shows an example from the MSR-OANC compression dataset. The example illustrates similar compression properties as seen in the in-domain settings. In particular, the two models learned to drop appositions, optional modifiers, detailed clauses, etc. Additionally, we also observed that the difficult cases where those where there is little to be removed, but due to higher compression ratios during the training, the models removed more than required. This confirms the

Source text:	Given the urgency of the situation in Alaska , Defenders needs your immediate assistance to help save Alaska ’s wolves from same - day airborne land - and - shoot slaughter .
Reference:	Given the urgency of the situation in Alaska , Defenders needs your immediate assistance saving Alaska ’s wolves from slaughter .
Seq2Seq-gen	
+ Random:	Immediate assistance to save Alaska’s tundra .
+ Coverage-AL:	Sometimes needs your assistance to help save Alaska ’s wolves .
+ Diversity-AL:	The situation in Alaska, help save Alaska ’s tundra .
Pointer-gen	
+ Random:	Immediate assistance to help save Alaska s wolves .
+ Coverage-AL:	The urgency of the situation in Alaska , Defenders needs your immediate assistance .
+ Diversity-AL:	Defenders needs your assistance to help save Alaska ’s wolves .

Table 5: Domain adaptation example from the MSR-OANC dataset when trained on a 20% of labelled compressions with Random, Coverage-AL, and Diversity-AL sampling strategies

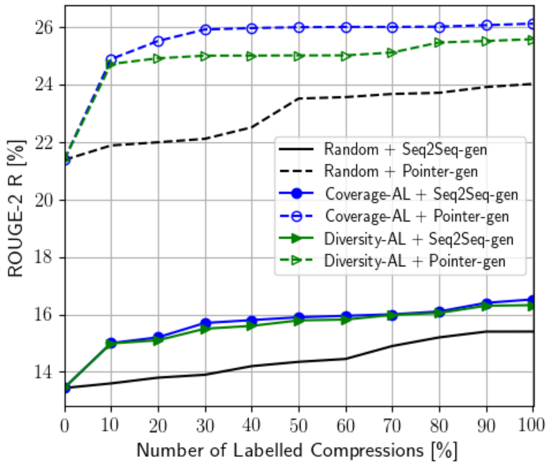


Figure 3: Analysis of the active learning for domain adaptation on the MSR-OANC dataset while varying the training data.

cause for lower ROUGE scores compared to the Google News corpus.

6 Conclusion

We propose a novel neural text compression approach using a neural Seq2Seq method with an interactive setup that aims at (a) learning an in-domain model with a minimum of data and (b) adapting a pretrained model with few user inputs to a new domain or genre. In this paper, we investigate two uncertainty-based active learning strategies with (a) a coverage constraint using attention dispersion and (b) a diversity constraint using structural similarity to make better use of the user in the loop for preparing training data pairs. The active learning based data selection methodology samples the data such that the most uncertain samples are available for training first. Experimental results show that the selected samples achieve comparable performance to the state-of-the-art systems, but trained on 80% less in-domain training data. Active learning with an interactive text compression model helps in transferring models trained on a large parallel corpus for a headline generation task to a general compression dataset with just 500 sampled instances. Additionally, the same in-domain active learning based data selection shows a notable performance improvement in an online interactive domain adaptation setup. Our experiments demonstrate that instead of more training data, relevant training data is essential for training Seq2Seq models in both in-domain training as well as domain adaptation.

In future work, we plan to explore several lines of work. First, we intend to investigate further applications of our interactive setup, e.g., in movie subtitle compression or television closed captions where there is no sufficient training data to build neural models. On a more general level, the interactive setup and the active learning strategies presented can also be used for other natural language processing tasks, such as question answering, to transfer a model to a new domain or genre.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We also acknowledge the useful suggestions of the anonymous reviewers.

References

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), June 23–30, 2007, Prague, Czech Republic*, pages 49–56.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT), June 12–17, 2016, San Diego CA, USA*, pages 93–98.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL), July 17–21, 2006, Sydney, Australia*, pages 377–384.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL), June 28–30, 2007, Prague, Czech Republic*, pages 1–11.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 137–144.

Trevor Cohn and Mirella Lapata. 2013. An abstractive approach to sentence compression. *ACM Transactions on Intelligent Systems and Technology*, 4(3):41:1–41:35.

Simon Corston-Oliver. 2001. Text compaction for display on very small screens. In *Proceedings of the NAACL Workshop on Automatic Summarization, June 3, 2001, Pitsburgh, PA, USA*, pages 89–98.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP) September 17–21, 2015, Lisbon, Portugal*, pages 360–368.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP) October 18–21, 2013, Seattle, WA, USA*, pages 1481–1491.

Gholamreza Haffari and Anoop Sarkar. 2009. Active learning for multilingual statistical machine translation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL/IJCNLP), August 2–7, 2009, Singapore*, pages 181–189.

Udo Hahn, Elena Beisswanger, Ekaterina Buyko, and Erik Faessler. 2012. Active learning-based corpus annotation - the pathojen experience. In *American Medical Informatics Association Annual Symposium (AMIA), November 3–7, 2012, Chicago, IL, USA*.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2018. Higher-order syntactic attention network for longer sentence compression. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL/HLT), June 1–6, 2018, New Orleans, LA, USA*, pages 1716–1726.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR), May 7–9, 2015, San Diego, CA, USA*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), May 27–June 1, 2003, Edmonton, Canada*, pages 48–54.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL Workshop "Text Summarization Branches Out", July 25-26, 2004, Barcelona, Spain*, pages 74–81.

Ming Liu, Wray L. Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL), October 31–November 1, 2018, Brussels, Belgium*, pages 334–344.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural*

Language Processing (EMNLP), September 17–21, 2015, Lisbon, Portugal, pages 1412–1421.

Juhani Luotolahti and Filip Ginter. 2015. Sentence compression for automatic subtitling. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA), May 11–13, 2015, Vilnius, Lithuania*, pages 135–143.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve J. Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), July 11–16, 2010, Uppsala, Sweden*, pages 1552–1561.

Erwin Marsi, Emiel Krahmer, Iris Hendrickx, and Walter Daelemans. 2010. On the limits of sentence compression by deletion. In *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*, volume 5790 of *Lecture Notes in Artificial Intelligence*, pages 45–66.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), November 1–4, 2016, Austin, TX, USA*, pages 319–328.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), June 7–8, 2012, Montréal, Canada*, pages 95–100.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of the NAACL-HLT Workshop on Evaluation Metrics and System Comparison for Automatic Summarization, June 3-8, 2012, Montréal, Canada*, pages 1–9.

Avinesh P. V. S. and Christian M. Meyer. 2017. Joint optimization of user-desired content in multi-document summaries by learning from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), July 30–August 4, Vancouver, Canada*, pages 1353–1363.

Álvaro Peris and Francisco Casacuberta. 2018. Active learning for interactive neural machine translation of data streams. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL), October 31–November 1, 2018, Brussels, Belgium*, pages 151–160.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), September 17–21, 2015, Lisbon, Portugal*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), July 30–August 4, Vancouver, Canada*, pages 1073–1083.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of the 6th International Conference on Learning Representations (ICLR), May 6–9, New Orleans, LA, USA*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), August 7–12, 2016, Berlin, Germany*.

Bipin Suresh. 2010. Inclusion of large input corpora in statistical machine translation. Technical report, Stanford University.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montréal, Canada*, pages 3104–3112.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), November 1-4, 2016, Austin, TX, USA*, pages 340–350.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), June 25-30, 2005, Ann Arbor, MI, USA*, pages 290–297.

Vincent Vandeghinste and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop "Text Summarization Branches Out", July 25-26, 2004, Barcelona, Spain*, pages 89–95.

Chenguang Wang, Laura Chiticariu, and Yunyao Li. 2017a. Active learning for black-box semantic role labeling with neural factors. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), August 19–25, 2017, Melbourne, Australia*, pages 2908–2914.

Gaoang Wang, Jenq-Neng Hwang, Craig S. Rose, and Farron Wallace. 2017b. Uncertainty sampling based active learning with diversity constraint by sparse selection. In *19th IEEE International Workshop* *on Multimedia Signal Processing (MMSP), October 16–18, 2017, Luton, UK*, pages 1–6.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv*, 1609.08144.

Sander Wubben, Emiel Krahmer, Antal van den Bosch, and Suzan Verberne. 2016. Abstractive compression of captions with attentive recurrent neural networks. In *Proceedings of the Ninth International Natural Language Generation Conference (INLG), September 5–8, 2016, Edinburgh, UK*, pages 41–50.

Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating diversity and density in active learning for relevance feedback. In *Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2–5, 2007, Proceedings*, pages 246–257.

Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G. Dy. 2011. Active learning from crowds. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML), June 28–July 2, 2011, Bellevue, WA, USA*, pages 1161–1168.

Naitong Yu, Jie Zhang, Minlie Huang, and Xiaoyan Zhu. 2018. An operation network for abstractive sentence compression. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING), August 20-26, 2018, Santa Fe, NM, USA*, pages 1065–1076.

Yang Zhao, Zhiyuan Luo, and Akiko Aizawa. 2018. A language model based evaluator for sentence compression. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), July 15–20, 2018, Melbourne, Australia*, pages 170–175.

Text Generation with Exemplar-based Adaptive Decoding

Hao Peng♠∗ **Ankur P. Parikh**♢ **Manaal Faruqui**♢ **Bhuwan Dhingra**♣∗ **Dipanjan Das**♢
♠ Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
♢ Google AI Language, New York, NY
♣ School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
`hapeng@cs.washington.edu, {aparikh,mfaruqui,dipanjand}@google.com,`
`bdhingra@andrew.cmu.edu`

Abstract

We propose a novel conditioned text generation model. It draws inspiration from traditional template-based text generation techniques, where the source provides the content (i.e., *what to say*), and the template influences *how to say it*. Building on the successful encoder-decoder paradigm, it first encodes the content representation from the given input text; to produce the output, it retrieves exemplar text from the training data as “soft templates,” which are then used to construct an exemplar-specific decoder. We evaluate the proposed model on abstractive text summarization and data-to-text generation. Empirical results show that this model achieves strong performance and outperforms comparable baselines.

1 Introduction

Conditioned text generation is the essence of many natural language processing (NLP) tasks, e.g., text summarization (Mani, 1999), machine translation (Koehn, 2009), and data-to-text generation (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997). In its common neural sequence-to-sequence formulation (Sutskever et al., 2014; Cho et al., 2014), an encoder-decoder architecture is used. The decoder generates the text autoregressively, token-by-token, conditioning on the feature representations encoded from the source, typically with attention (Bahdanau et al., 2015) and copy mechanisms (Gu et al., 2016; See et al., 2017). This paradigm is capable of generating fluent abstractive text, but in an uncontrolled and sometimes unreliable way, often producing degenerate outputs and favoring generic utterances (Vinyals and Le, 2015; Li et al., 2016).

The encoder-decoder approach differs considerably from earlier template-based methods (Becker, 2002; Foster and White, 2004; Reiter et al., 2005; Gatt and Reiter, 2009, *inter alia*), where the source content is filled into the slots of a handcrafted template. These solutions offer higher generation precision compared to neural approaches (Wiseman et al., 2017), but tend to lack the naturalness of neural systems, and are less scalable to open domain settings, where the number of required templates can be prohibitively large.

To sidestep the scalability problems with handcrafted templates, it has been proposed to use similar training samples as **exemplars**, to guide the decoding process (Gu et al., 2018; Guu et al., 2018; Weston et al., 2018; Pandey et al., 2018; Cao et al., 2018a, *inter alia*).[1] In general, existing methods accomplish this by (a) using traditional information retrieval (IR) techniques for exemplar extraction (e.g., TF-IDF), and then (b) concatenating the exemplar to the source as additional inputs, allowing the decoder to attend over and copy from both.

We propose a different strategy for using exemplars. For motivation, Figure 1 shows a source-target pair together with its exemplar from the Gigaword dataset (Graff et al., 2003). The target is a summary of the source sentence, and the exemplar is retrieved from the training set (§3.2).[2] There is word overlap between the exemplar and the desired output, which would be easily captured by an attention/copy mechanism (e.g. *Norway* and *aid*). Despite this, ideally, the model should also exploit the structural and stylistic aspects to produce an output with a similar sentence structure, even if the words are different.

Indeed, in traditional templates, the source

∗ Work done during internship at Google.

[1] The term **exemplar** indicates a training instance used to help generation. We aim to distinguish from “templates,” since here no explicit slot-filling procedure is involved.

[2] We use the *training target* as the exemplar, whose source is most similar to the current input. §3.2 describes the details.

Proceedings of NAACL-HLT 2019, pages 2555–2565
Minneapolis, Minnesota, June 2 - June 7, 2019.

Source: Norway said Friday it would give Zimbabwe 40 million kroner (7.02 million dollars, 4.86 million euros) in aid to help the country deal with a lack of food and clean drinking water and a cholera outbreak.
Exemplar: Norway boosts earthquake aid to Pakistan.
Target: Norway grants aid of 4.86 million euros to Zimbabwe.

Figure 1: A source-target pair from Gigaword training set, along with its exemplar.

is supposed to determine "what to say," while the templates aim to address "how to say it," reminiscent of the classical content selection and surface realization pipeline (Reiter and Dale, 1997). For instance, an ideal template for this example might look as follows:

`_____ grants aid of _____ to _____`

In the neural formulation, the "how to say it" aspect is primarily controlled by the decoder.

Inspired by the above intuition, we propose **exemplar-based adaptive decoding**, where a customized decoder is constructed for each exemplar. This is achieved by letting the exemplars to directly influence decoder parameters through a reparameterization step (§3.1). The adaptive decoder can be used as a drop-in replacement in the encoder-decoder architecture. It offers the potential to better incorporate the exemplars' structural and stylistic aspects into decoding, without excessive increase in the amount of parameters or computational overhead.

We empirically evaluate our approach on abstractive text summarization and data-to-text generation (§4), on which most of the recent efforts on exemplar-guided text generation have been studied. On three benchmark datasets, our approach outperforms comparable baselines, and achieves performance competitive with the state of the art. The proposed method can be applicable in many other conditioned text generation tasks. Our implementation is available at `https://homes.cs.washington.edu/~hapeng`.

2 Background

This section lays out the necessary background and notations for further technical discussion. We begin with conditioned text generation and the encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014). In the interest of the notation clarity, §3 will use an Elman network (Elman, 1990) as a running example for the decoder, which is briefly reviewed in §3. The proposed technique generalizes to other neural network architectures (§3.3).

Conditioned text generation and the encoder-decoder architecture. Our discussion centers around conditioned text generation, i.e., the model aims to output the target $\mathbf{y} = y_1 y_2 \dots y_T$ given the source input $\mathbf{x} = x_1 x_2 \dots x_S$, both of which are sequences of tokens. Each token x_i, y_i takes one value from a vocabulary $\mathcal{V}$. $\mathbf{x}$ and $\mathbf{y}$ could vary depending on the tasks, e.g., they will respectively be articles and summaries for text summarization; and for data-to-text generation, $\mathbf{x}$ would be structured data, which can sometimes be linearized (Lebret et al., 2016; Wiseman et al., 2018, *inter alia*), and $\mathbf{y}$ is the output text. We aim to learn a (parameterized) conditional distribution of the target text $\mathbf{y}$ given the source $\mathbf{x}$,

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{T} p(y_t \mid \mathbf{y}_{<t}, \mathbf{x}), \tag{1}$$

where $\mathbf{y}_{<t} = y_1 \dots y_{t-1}$ is the prefix of $\mathbf{y}$ up to the $(t-1)^{\text{th}}$ token (inclusive).

The probability of each target token is usually estimated with a $\operatorname{softmax}$ function:

$$p(y_t \mid \mathbf{y}_{<t}, \mathbf{x}) = \frac{\exp \mathbf{h}_{t-1}^{\top} \mathbf{w}_{y_t}}{\sum_y \exp \mathbf{h}_{t-1}^{\top} \mathbf{w}_y}. \tag{2}$$

$\mathbf{w}_y$ denotes a learned vector for token $y \in \mathcal{V}$. $\mathbf{h}_{t-1}$ depends on $\mathbf{y}_{<t}$ and $\mathbf{x}$, and is computed by a function which we will describe soon.

A typical implementation choice for computing $\mathbf{h}_t$ is the encoder-decoder architecture (Sutskever et al., 2014). More specifically, an **encoder** $\mathbf{g}_{\boldsymbol{\theta}}$ first gathers the feature representations from the source $\mathbf{x}$; then a **decoder** $\mathbf{f}_{\boldsymbol{\phi}}$ is used to compute the $\mathbf{h}_t$ feature vectors:

$$\mathbf{h}_t = \mathbf{f}_{\boldsymbol{\phi}}\Big(y_t, \mathbf{h}_{t-1}, \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x})\Big). \tag{3}$$

$\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are, respectively, the collections of parameters for the encoder and the decoder, both of which can be implemented as recurrent neural networks (RNNs) such as LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014), or the transformer (Vaswani et al., 2017). In Sutskever et al. (2014), the dependence of $\mathbf{f}_{\boldsymbol{\phi}}$ on $\mathbf{g}_{\boldsymbol{\theta}}$ is made by using the last hidden state of the encoder as the initial state of the decoder. Such dependence can be further supplemented with attention (Bahdanau et al., 2015) and copy mechanisms (Gu et al., 2016; See et al., 2017), as we will do in this work.

§3 introduces how we use exemplars to inform decoding, by dynamically constructing the decoder's parameters $\boldsymbol{\phi}$. For the notation clarity, we will use the Elman network as a running example, reviewed below.

Elman networks. Given input sequence $\mathbf{x}$, an Elman network (Elman, 1990) computes the hidden state at time step t from the previous one and the current input token by

$$\mathbf{h}_t = \tanh\big(\mathbf{P}\mathbf{h}_{t-1} + \mathbf{Q}\mathbf{v}_t\big), \tag{4}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are learned $d \times d$ parameter matrices (with d being the hidden dimension), and $\mathbf{v}_t$ is the embedding vector for token x_t. We omit the bias term for clarity.

3 Method

This section introduces the proposed method in detail. Our aim is to use exemplars to inform the decoding procedure (i.e., *how to say it*). To accomplish this, we reparameterize the decoder's parameters with weighted linear sums, where the coefficients are determined by an exemplar. The decoder is **adaptive**, in the sense that its parameters vary according to the exemplars. The adaptive decoder can be used as a drop-in replacement in the encoder-decoder architecture. Before going into details, let us first overview the high-level generation procedure of our model. Given source text $\mathbf{x}$, the model generates an output as follows:

1. Run a standard encoder to gather the content representations $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x})$ from the source.
2. Retrieve its exemplar $\mathbf{z}_{\mathbf{x}}$, and compute exemplar-specific coefficients (§3.2).
3. Construct the adaptive decoder parameters $\boldsymbol{\phi}$ (§3.1), using the coefficients computed at step 2. Then the output is generated by applying the adaptive decoder followed by a $\operatorname{softmax}$, just as in any other encoder-decoder architecture.

Aiming for a smoother transition, we will first describe step 3 in §3.1, and then go back to discuss step 2 in §3.2. For clarity, we shall assume that the decoder is implemented as an Elman network (Elman, 1990; Equation 4). The proposed technique generalizes to other neural network architectures, as we will discuss later in §3.3.

3.1 Reparameterizing the RNN Decoder

At its core, the exemplar-specific adaptive decoder involves a reparameterization step, which we now describe. We focus on the parameters of the Elman network decoder, i.e., $\mathbf{P}$ and $\mathbf{Q}$ in Equation 4.

Parameter construction with linear sums. We aim to reparameterize the pair of matrices $(\mathbf{P}, \mathbf{Q})$, in a way that they are influenced by the exemplars.

Let us first consider an extreme case, where one assigns a different pair of parameter matrices to each exemplar, *without* any sharing. This leads to an unreasonably large amount of parameters, which are difficult to estimate reliably.[3]

We instead construct $\mathbf{P}$ and $\mathbf{Q}$ from a set of predefined parameters matrices. Take $\mathbf{P}$ for example, it is computed as the weighted sum of $\mathbf{P}_i$ matrices:

$$\mathbf{P} = \sum_{i=1}^{r} \lambda_i \mathbf{P}_i, \tag{5}$$

where $\mathbf{P}_i \in \mathbb{R}^{d\times d}$, with d being the size of the hidden states. r is a hyperparameter, determining the number of $\mathbf{P}_i$ matrices to use.[4] The summation is weighted by the coefficients λ_i, which are computed from the exemplar $\mathbf{z}_{\mathbf{x}}$. For clarity, the dependence of both $\mathbf{P}$ and λ_i on $\mathbf{z}_{\mathbf{x}}$ is suppressed when the context is clear.

Equation 5 constructs the decoder's parameter matrix $\mathbf{P}$ using a linear combination of $\{\mathbf{P}_i\}_{i=1}^{r}$. The exemplar informs this procedure through the coefficients λ_i's, the detailed computation of which is deferred to §3.2. The other matrix $\mathbf{Q}$ can be similarly constructed by $\mathbf{Q} = \sum_i \lambda_i \mathbf{Q}_i$.

Rank-1 constraints. In the above formulation, the number of parameters is still r times more than a standard Elman network, which can lead to overfitting with a limited amount of training data. Besides, it would be more interesting to compare the adaptive decoder to a standard RNN under a comparable parameter budget. Therefore we want to further limit the amount of parameters. This can be achieved by forcing the ranks of $\mathbf{P}_i$ and $\mathbf{Q}_i$ to be 1, since it then takes $2d$ parameters to form each of them, instead of d^2. More formally, we upper-

---

[3] The amount of parameters grows linearly with the number of possible exemplars, which, as we will soon discuss in §3.2, can be as large as the training set.

[4] Instead of choosing r empirically, we set it equal to d in the experiments. Please see the end of §3.1 for a related discussion.

bound their ranks by construction:

$$\mathbf{P}_i = \mathbf{u}_i^{(p)} \otimes \mathbf{v}_i^{(p)}. \tag{6}$$

$\mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^\top$ denotes the outer product of two vectors; $\mathbf{u}_i^{(p)}$ and $\mathbf{v}_i^{(p)}$ are learned d-dimensional vectors. Each $\mathbf{Q}_i$ can be similarly constructed by a separate set of vectors $\mathbf{Q}_i = \mathbf{u}_i^{(q)} \otimes \mathbf{v}_i^{(q)}$.

Let $\mathbf{U}_p, \mathbf{V}_p \in \mathbb{R}^{d \times r}$ denote the stack of $\mathbf{u}_i^{(p)}$, $\mathbf{v}_i^{(p)}$ vectors, i.e.,

$$\mathbf{U}_p = \left[\mathbf{u}_1^{(p)}, \dots, \mathbf{u}_r^{(p)}\right], \tag{7a}$$

$$\mathbf{V}_p = \left[\mathbf{v}_1^{(p)}, \dots, \mathbf{v}_r^{(p)}\right]. \tag{7b}$$

Equations 5 and 6 can be compactly written as

$$\mathbf{P} = \mathbf{U}_p \, \mathbf{\Lambda} \, \mathbf{V}_p^\top. \tag{8}$$

where $\mathbf{\Lambda}$ is the diagonal matrix built from the r-dimensional coefficient vector $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_r]^\top$:

$$\mathbf{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda}) = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r \end{bmatrix}. \tag{9}$$

The construction of $\mathbf{Q}$ is similar, but with a different set of parameters matrices $\mathbf{U}_q$ and $\mathbf{V}_q$:[5]

$$\mathbf{Q} = \mathbf{U}_q \, \mathbf{\Lambda} \, \mathbf{V}_q^\top. \tag{10}$$

Note that, despite their similarities to SVD at a first glance, Equations 8 and 10 are not performing matrix factorization. Rather, we are learning $\{\mathbf{U}_p, \mathbf{V}_p, \mathbf{U}_q, \mathbf{V}_q\}$ directly; $\mathbf{P}$, $\mathbf{Q}$, $\{\mathbf{P}_i\}$, and $\{\mathbf{Q}_i\}$ are never explicitly instantiated (Peng et al., 2017, 2018c).

To summarize, we reparameterize $\mathbf{P}$ and $\mathbf{Q}$ as interpolations of rank-1 matrices. By the fact that $\operatorname{rank}(\mathbf{A} + \mathbf{B}) \leq \operatorname{rank}(\mathbf{A}) + \operatorname{rank}(\mathbf{B})$, the ranks of $\mathbf{P}$ and $\mathbf{Q}$ are upper-bounded by r. As pointed out by Krueger and Memisevic (2017), the parameter matrices of a trained RNN tend to have full rank. Therefore, in the experiments, we set r equal to the hidden size d, aiming to allow the adaptive decoder to use full-rank matrices in the recurrent computation. Yet, if one holds *a priori* beliefs that the matrices should have lower ranks, using $r < d$ could be desirable. When $r = d$, an adaptive RNN constructed by the above approach has $4d^2$ parameters, which is comparable to the $2d^2$ parameters in a standard Elman network.[6]

3.2 Incorporating Exemplars

We now discuss the computation of coefficients $\boldsymbol{\lambda}$, through which the exemplars inform the decoder construction (Equations 8 and 10). Before detailing the neural network architecture, we begin by describing the exemplar retrieval procedure.

Retrieving exemplars $\mathbf{z}_\mathbf{x}$. Intuitively, similar source texts should hold similar targets. Therefore, given source input $\mathbf{x}$, we use the *training target* as its exemplar $\mathbf{z}_\mathbf{x}$, whose source is most similar to $\mathbf{x}$.[7] To compute the similarities between source texts, we use bag-of-words (BOW) features and cosine similarity. We extract the top-1 exemplar for each instance. This step is part of the preprocessing, and we do *not* change the exemplars as the training proceeds.

There are, of course, many other strategies to get the exemplars, e.g., using handcrafted or heuristically created hard templates (Reiter et al., 2005; Becker, 2002; Foster and White, 2004, *inter alia*), randomly sampling *multiple* training instances (Guu et al., 2018), or learning a neural reranker (Cao et al., 2018a). Using more sophistically extracted exemplars is definitelly interesting to explore, which we defer to future work.

Computing coefficients. Next we describe the computation of $\boldsymbol{\lambda}$, the r-dimensional coefficient vector, which is used to construct the adaptive decoder (Equations 8 and 10).

Intuitively, the rank-1 matrices ($\mathbf{P}_i$'s and $\mathbf{Q}_i$'s in Equation 6 and thereafter) can be seen as capturing different aspects of the generated text. And $\boldsymbol{\lambda}$ determines how much each of them contributes to the adaptive decoder construction. A natural choice to calculate $\boldsymbol{\lambda}$ is to use the similarities between the exemplar and each of the aspects.

To accomplish this, we run a RNN encoder over $\mathbf{z}_\mathbf{x}$, and use the last hidden state as its vector representation $\mathbf{a}$.[8] We further associate each $(\mathbf{P}_i, \mathbf{Q}_i)$ pair with a learned vector $\mathbf{c}_i$; and then λ_i is computed as the similarity between $\mathbf{a}$ and $\mathbf{c}_i$, using an inner product $\lambda_i = \mathbf{a}^\top \mathbf{c}_i$. More compactly,

$$\boldsymbol{\lambda} = \mathbf{C}\mathbf{a}, \tag{11}$$

with $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_r]^\top$.

---

[5] The bias term in the Elman network $\mathbf{b}$ can be constructed as $\mathbf{b} = \mathbf{B}\boldsymbol{\lambda}$, with $\mathbf{B}$ being a learned $d \times r$ matrix.

[6] This does not include the bias term, which contributes additional d^2 parameters to the former, and d to the latter.

[7] The source of an exemplar is only used in the retrieval and never fed into the encoder-decoder model. For a training instance, we additionally disallow using its own target as the exemplar.

[8] For clarity, the dependence of $\mathbf{a}$ on the exemplar $\mathbf{z}_\mathbf{x}$ is suppressed, just as $\boldsymbol{\lambda}$.

Algorithm 1 Adaptive decoder construction.

```
procedure $(\mathbf{x})$
    Retrieve the exemplar $\mathbf{z}_\mathbf{x}$                ▷ §3.2
    Compute $\mathbf{z}_\mathbf{x}$'s representation $\mathbf{a}$      ▷ §3.2
    Compute coefficients $\boldsymbol{\lambda}$                 ▷ Eq.11
    Construct the decoder $\mathbf{f}_\phi$             ▷ Eqs.8, 10
end procedure
```

Closing this section, Algorithm 1 summarizes the procedure to construct an adaptive decoder.

3.3 Discussion.

Although we've based our discussion on Elman networks so far, it is straightforward to apply this method to its gated variants (Hochreiter and Schmidhuber, 1997; Cho et al., 2014, *inter alia*), and other quasi-/non-recurrent neural architectures (Bradbury et al., 2017; Vaswani et al., 2017; Peng et al., 2018a, *inter alia*). Throughout the experiments, we will be using an adaptive LSTM decoder (§4). As a drop-in replacement in the encoder-decoder architecture, it introduces a reasonable amount of additional parameters and computational overhead, especially when one uses a small encoder for the exemplar (i.e., the sizes of the $\mathbf{c}_i$ vectors in Equation 11 are small). It can benefit from the highly-optimized GPU implementations, e.g., CuDNN, since it uses the same recurrent computation as a standard nonadaptive RNN.

In addition to the neural networks, the adaptive decoder requires access to the full training set due to the retrieval step. In this sense it is *semi-parametric*.[9] The idea to dynamically construct the parameters is inspired by Hypernetworks (Ha et al., 2017) and earlier works therein. It proves successful in tasks such as classification (Jia et al., 2016; Liu et al., 2017) and machine translation (Platanios et al., 2018). Many recent template-based generation models include the exemplars as *content* in addition to the source, and allow the decoder to attend over and copy from both (Gu et al., 2018; Guu et al., 2018; Weston et al., 2018; Pandey et al., 2018; Cao et al., 2018a, *inter alia*). We compare to this approach in the experiments, and show that our model offers favorable performance, and that they can potentially be combined to achieve further improvements.

		NYT	Giga	Wikibio
# inst.	Train	92K	3.8M	583K
	Dev.	9K	190K	73K
	Test	9,706	1,951	73K
Avg. len.	Src.	939.0	31.4	N/A
	Tgt.	48.6	8.2	26.0

Table 1: Number of instances and average text lengths for the datasets used in the experiments. The lengths are averaged over training instances.

4 Experiments

This section empirically evaluates the proposed model on two sets of text generation tasks: abstractive summarization (§4.2) and data-to-text generation (§4.3). Before heading into the experimental details, we first describe the architectures of the compared models in §4.1.

4.1 Compared Models

In addition to previous works, we compare to the following baselines, aiming to control for confounding factors due to detailed implementation choices.

- SEQ2SEQ. The encoder-decoder architecture enhanced with attention and copy mechanisms. The encoder is implemented with a bi-directional LSTM (BiLSTM; Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997; Graves, 2012), and the decoder a uni-directional one. We tie the input embeddings of both the encoder and the decoder, as well as the $\mathrm{softmax}$ weights (Press and Wolf, 2017). We use beam search during evaluation, with length penalty (Wu et al., 2016).
- ATTEXP. It is based on SEQ2SEQ. It encodes, attends over, and copies from the exemplars, in addition to the source inputs.

Our model using the adaptive decoder (ADADEC) closely builds upon SEQ2SEQ. It uses a dynamically constructed LSTM decoder, and *does not* use attention or copy mechanisms over the encoded exemplars. The extracted exemplars are the same as those used by ATTEXP. To ensure fair comparisons, we use comparable training procedures and regularization techniques for the above models. The readers are referred to the appendix for further details such as hyperparameters.

---

[9] Nothing prohibits adaptively constructing other components of the model, e.g., the encoder $\mathbf{g}_\theta$. Yet, our motivation is to use exemplars to inform *how to say it*, which is primarily determined by the decoder (in contrast, the encoder relates more to selecting the content).

4.2 Text Summarization

Datasets. We empirically evaluate our model on two benchmark text summarization datasets:

- Annotated Gigaword corpus (Gigaword; Graff et al., 2003; Napoles et al., 2012). Gigaword contains news articles sourced from various news services over the last two decades. To produce the dataset, we follow the split and preprocessing by Rush et al. (2015), and pair the first sentences and the headlines in the news articles. It results in a 3.8M/190K/1,951 train/dev./test split. The average lengths of the source and target texts are 31.4 and 8.2, respectively.
- New York Times Annotated Corpus (NYT; Sandaus, 2008). It contains news articles published between 1996 and 2007 by New York Times. We use the split and preprocessing by Durrett et al. (2016).[10] Following their effort, we evaluate on a smaller portion of the test set, where the gold summaries are longer than 50 tokens. We further randomly sample 9,000 instances from the training data for validation, resulting in a 91,834/9,000/3,452 train/dev./test split. Compared to Gigaword, the inputs and targets in NYT are much longer (averaging 939.0 and 48.6, respectively).

Table 1 summarizes some statistics of the datasets. We note that some recent works use a different split of the NYT corpus (Paulus et al., 2018; Gehrmann et al., 2018), and thus are not comparable to the models in Table 3. We decide to use the one by Durrett et al. (2016) because their preprocessing script is publicly available.

For both datasets, we apply byte-paired encoding (BPE; Sennrich et al., 2016), which proves to improve the generation of proper nouns (Fan et al., 2018).

Empirical results. Table 2 compares the models on Gigaword test set in ROUGE F_1 (Lin, 2004).[11]

Model	RG-1	RG-2	RG-L
Open-NMT	35.0	16.6	32.4
†Cao et al., 2018a (BASIC)	36.0	17.1	33.2
†Cao et al., 2018a (FULL)	37.0	**19.0**	34.5
⋆Cao et al. (2018b)	**37.3**	17.6	34.2
This work (SEQ2SEQ)	35.8	17.5	33.5
†This work (ATTEXP)	36.0	17.7	33.1
†This work (ADADEC)	**37.3**	18.5	**34.7**

Table 2: Text summarization performance in ROUGE F_1 scores (dubbed as RG-X) on Gigaword test set (§4.2). † denotes the models using retrieved exemplars, while ⋆ uses handcrafted features. Bold font indicates best performance. Open-NMT numbers are taken from Cao et al. (2018a).

By using adaptive decoders, our model (ADADEC) improves over SEQ2SEQ by more than 1.1 ROUGE scores. Cao et al. (2018b) and the FULL model by Cao et al. (2018a) hold the best published results. The former uses extensive handcrafted features and relies on external information extraction and syntactic parsing systems; while the latter uses additional encoding, attention and copy mechanisms over the exemplars extracted using a novel neural reranker. ADADEC achieves better or comparable performance to the state-of-the-art models, *without* using any handcrafted features or reranking techniques. The BASIC model by Cao et al. (2018a) ablates the reranking component from their FULL model, and uses the top exemplar retrieved by the IR system. Therefore it is a more comparable baseline to ours. ADADEC outperforms it by more than 1.3 ROUGE scores. Surprisingly, we do not observe interesting improvements by ATTEXP over the sequence-to-sequence baseline. We believe that our model can benefit from better extracted exemplars by, e.g., applying a reranking system. Such exploration is deferred to future work.

The NYT experimental results are summarized in Table 3. We follow previous works and report limited-length ROUGE *recall* values.[12] Durrett et al. (2016) is an extractive model, and Paulus et al. (2018) an abstractive approach based on reinforcement learning. Our ADADEC model outperforms both. We observe similar trends when comparing ADADEC to the SEQ2SEQ and ATTEXP baselines, with the exception that ATTEXP *does* improve over SEQ2SEQ.

4.3 Data-to-text Generation

Data-to-text generation aims to generate textual descriptions of structured data, which can be

---

[10] `https://github.com/gregdurrett/berkeley-doc-summarizer`.

[11] Version 1.5.5 of the official script.

[12] Following Durrett et al. (2016) and Paulus et al. (2018), we truncate the predictions to the lengths of the gold summaries, and evaluate ROUGE recall, instead of F_1 on full-length predictions.

Model	**ROUGE-1**	**ROUGE-2**
Durrett et al. (2016)	42.2	24.9
Paulus et al. (2018)	42.9	26.0
This work (SEQ2SEQ)	41.9	25.1
†This work (ATTEXP)	42.5	25.7
†This work (ADADEC)	**43.2**	**26.4**

Table 3: NYT text summarization test performance in ROUGE *recall* values. This is a smaller portion of the original test data, after filtering out instances with summaries shorter than 50 tokens (§4.2; Durrett et al., 2016). † denotes the models using retrieved exemplars, and bold font indicates best performance.

seen as a table consisting of a collection of records (Liang et al., 2009). For a given entity, each record is an *(attribute, value)* tuple. Figure 2 shows an example for entity *Jacques-Louis David*. The table specifies the entity's properties with tuples *(born, 30 August 1748)*, *(nationality, French)*, and so forth. The table is paired with a description, which the model is supposed to generate using the table as input. We refer the readers to Lebret et al. (2016) for more details about the task.

Dataset and implementation details. We use the Wikibio dataset (Lebret et al., 2016). It is automatically constructed by pairing the tables and the opening sentences of biography articles from English Wikipedia. We follow the split and preprocessing provided along with the dataset, with around 583K/73K/73K train/dev./test instances. Following Lebret et al. (2016), we linearize the tables, such that we can conveniently train the sequence-to-sequence style models described in §4.1. Table 1 summarizes some statistics of the dataset.

In contrast to the text summarization experiment (§4.2), we do *not* apply BPE here. Further, the word embeddings are initialized with `GloVe` (Pennington et al., 2014; fixed during training), and *not* tied with the softmax weights. In addition to the models introduced in §4.1, we additionally compare to ADADEC+ATTEXP, aiming to study whether the adaptive decoder can further benefit from attention and copy mechanisms over the exemplars.

Jacques-Louis David

Born	30 August 1748 Paris, Kingdom of France
Died	29 December 1825 (aged 77) Brussels, United Netherlands
Nationality	French
Alma mater	Collège des Quatre-Nations, University of Paris
Known for	Painting, Drawing
Notable work	*Oath of the Horatii* (1784), *The Death of Marat* (1793)
Movement	Neoclassicism
Awards	Prix de Rome Commander of the Legion of Honour

Jacques-Louis David (30 August 1748 – 29 December 1825) was a French painter in the Neoclassical style.

Figure 2: A training instance from the Wikibio dataset. It consists of a collections of records for Jacques-Louis David (top), and a piece of textual description (bottom).

Empirical results. Following Liu et al. (2018) we report ROUGE-4 and BLEU scores (Papineni et al., 2002).[13] Table 4 summarizes the data-to-text generation results on the Wikibio test set. Overall, we observe similar trends to those in the summarization experiment (§4.2): by attending over and copying from the exemplars, ATTEXP improves upon the SEQ2SEQ baseline by around 0.6 absolute scores. Also utilizing exemplar information, our ADADEC model outperforms SEQ2SEQ by a larger margin: 1.3 for ROUGE-4 and 1.1 for BLEU. We further study whether we can get further improvements by combining both. ADADEC+ATTEXP achieves around 0.5 absolute improvements over ADADEC, less than those by ATTEXP over SEQ2SEQ. This provides evidence that, to some extend, the ways ATTEXP and ADADEC incorporate exemplar information might be complementary. Wiseman et al. (2018) is a template-motivated model based on a semi-Markov model. Liu et al. (2018) hold the current state-of-the-art results. They encode the table structures by using (a) position and filed embeddings, and (b) structure-aware attention and gating techniques. These techniques are beyond the scope of this work, which focuses mainly on the decoding end.

---

[13] We use the script by Lin (2004) to calculate the ROUGE score, and the mteval script for BLEU: `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/mteval-v13a.pl`.

Model	**RG-4**	**BLEU**
Wiseman et al. (2018)	38.6	34.8
Liu et al. (2018)	41.7	44.7
This work (Seq2Seq)	39.3	42.5
†This work (AttExp)	40.0	43.1
†This work (AdaDec)	40.6	43.6
†This work (AdaDec+AttExp)	41.1	44.1

Table 4: Data-to-text generation performance in Rouge-4 and Bleu on the Wikibio test set (§4.3). † indicates the models using retrieved exemplars.

Source: A Portuguese train derailed in the northern region of Oporto on Wednesday, killing three people...

Exemplar 1: Two die in a Britain train collision.
Output 1: Three killed in Portuguese train derailment.

Exemplar 2: Two **people were killed** in Britain train collision.
Output 2: Three *people* killed in Portuguese train derailment.

Exemplar 3: A train collision in **Canada killed two people**.
Output 3: Portuguese train derails in *northern Mexico killing three*.

Source: Sales of new homes in the U.S. increased by 11.8 percent in May, the biggest gain in 26 years...

Exemplar 1: U.S. sales of new homes up strongly in March.
Output 1: US new home sales rise 11.8 percent in May.

Exemplar 2: The **sales of new homes in the U.S. grow** strongly.
Output 2: *Sales of new homes in US* rise in May.

Exemplar 3: **U.S. economic statistics: new home sales grow** by 2.5 percent.
Output 3: *US new home sales grow 26* percent in May.

Figure 3: Two randomly sampled Gigaword development instances used for qualitative evaluation (§5). `Exemplar 1`'s are retrieved by the system (§3.2), while the remaining ones are produced by the authors. Notable exemplars changes are highlighted in **bold purple**, and output changes in *italic yellow*.

5 Analysis

We now qualitatively evaluate our model, by studying how its outputs are affected by using different exemplars. Figure 3 shows two randomly sampled Gigaword development instances. It compares the outputs by AdaDec (i.e., without attention/copy over exemplars; §4.1) when receiving different exemplars, controlling for the same source inputs. In each example, `Exemplar 1` is retrieved by the system (i.e., a training target; §3.2); while the remaining ones are produced by the authors, by modifying the first one in styles and sometimes introducing distractions in the content.

In the top example, the model includes *people* into the subject (*Three* vs. *Three people*) under the influence by `Exemplar 2`; `Exemplar 3` changes the tense and adds some distraction by changing the place from *Britain* to *Canada*. The model follows the tense switch, but gets confused by the distraction, and decides to let a train in southern Europe collide into North America, which it should not. Looking at the bottom example, the model in general follows the exemplar in using noun adjuncts or prepositional phrases (e.g., *new home sales* vs. *sales of new homes*), except the first one. Perhaps confused by the distraction in `Exemplar 3`, the model makes a judgment on the specific amount of growth, but gets it wrong.

6 Related Work

Exemplar-based generation. Partly inspired by traditional template-based generation (Kukich, 1983; Reiter and Dale, 1997, *inter alia*), many recent efforts have been devoted to augmenting text generation models with retrieved exemplars (Hodosh et al., 2013; Mason and Charniak, 2014; Song et al., 2016; Lin et al., 2017, *inter alia*). Without committing to an explicit slot-filling process, a typical method is to include exemplars as additional inputs to the sequence-to-sequence models (Gu et al., 2018; Pandey et al., 2018; Guu et al., 2018, *inter alia*). Wiseman et al. (2018) took a different approach and used a semi-Markov model to learn templates.

Dynamic parameter construction. The idea of using a smaller network to generate weights for a larger one dues back to Stanley et al. (2009) and Koutnik et al. (2010), mainly under the evolution computing context. It is later revisited with representation learning (Moczulski et al., 2015; Fernando et al., 2016; Al-Shedivat et al., 2017, *inter alia*), and successfully applied to classification (Jia et al., 2016; Liu et al., 2017) and machine translation (Platanios et al., 2018). It also relates to the meta-learning set-up (Thrun and Pratt, 1998).

7 Conclusion

We presented a text generation model using exemplar-informed adaptive decoding. It reparameterizes the decoder using the information gathered from retrieved exemplars. We experimented with text summarization and data-to-text generation, and showed that the proposed model achieves strong performance and outperforms comparable baselines on both. The proposed model can be applicable in other conditioned text generation tasks. We release our implementation at `https://homes.cs.washington.edu/~hapeng`.

Acknowledgments

We thank Antonios Anastasopoulos, Ming-Wei Chang, Michael Collins, Jacob Devlin, Yichen Gong, Luheng He, Kenton Lee, Dianqi Li, Zhouhan Lin, Slav Petrov, Oscar Täckström, Kristina Toutanova, and other members of the Google AI language team for the helpful discussion, and the anonymous reviewers for their valuable feedback.

References

Maruan Al-Shedivat, Avinava Dubey, and Eric P. Xing. 2017. Contextual explanation networks. *arXiv:1705.10301*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Tilman Becker. 2002. Practical, template–based natural language generation with tag. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Network. In *Proc. of ICLR*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018a. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proc. of ACL*.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018b. Faithful to the original: Fact aware neural abstractive summarization. In *Proc. of AAAI*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proc. of ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*.

Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. 2016. Convolution by evolution: Differentiable pattern producing networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*.

Mary Ellen Foster and Michael White. 2004. Techniques for text planning with xslt. In *Proceeedings of the Workshop on NLP and XML: RDF/RDFS and OWL in Language Technology*.

Albert Gatt and Ehud Reiter. 2009. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proc. of EMNLP*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English Gigaword Second Edition.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. of ACL*.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided nonparametric neural machine translation. In *Proc. of AAAI*.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL*, 6:437–450.

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *Proc. of ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *Proc. of CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 47(1):853–899.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. 2016. Dynamic filter networks. In *Proc. of NeuralIPS*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Proc. of NeuralIPS*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Jan Koutnik, Faustino Gomez, and Jürgen Schmidhuber. 2010. Evolving neural networks in compressed weight space. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*.

David Krueger and Roland Memisevic. 2017. Regularizing rnns by stabilizing activations. In *Proc. of ICLR*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proc. of ACL*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proc. of EMNLP*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-ting Sun. 2017. Adversarial ranking for language generation. In *Proc. of NeuralIPS*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Dynamic compositional neural networks over tree structure. In *Proc. of IJCAI*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proc. of AAAI*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Inderjeet Mani. 1999. *Advances in automatic text summarization*. MIT press.

Rebecca Mason and Eugene Charniak. 2014. Domain-specific image captioning. In *Proc. of CoNLL*.

Kathleen McKeown. 1992. *Text generation*. Cambridge University Press.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. 2015. ACDC: A structured efficient linear layer. *arXiv:1511.05946*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *Proc. of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proc. of ICLR*.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018a. Rational recurrences. In *In Proc. of EMNLP*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018b. Backpropagating through structured argmax using a spigot. In *Proc. of ACL*.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018c. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proc. of EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proc. of EMNLP*.

Evan Sandaus. 2008. *The New York Times Annotated Corpus*. LDC corpora. Linguistic Data Consortium.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Transactions on Signal Proccesing*, 45(11):2673–2681.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *arXiv:1610.07149*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NeuralIPS*.

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to Learn*. Kluwer Academic Publishers.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeuralIPS*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proc. of ICML*.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the International Workshop on Search-Oriented Conversational AI*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proc. of EMNLP*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. In *Proc. of EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*.

Guiding Extractive Summarization with Question-Answering Rewards

Kristjan Arumae and **Fei Liu**
Computer Science Department
University of Central Florida, Orlando, FL 32816, USA
k.arumae@knights.ucf.edu feiliu@cs.ucf.edu

Abstract

Highlighting while reading is a natural behavior for people to track salient content of a document. It would be desirable to teach an extractive summarizer to do the same. However, a major obstacle to the development of a supervised summarizer is the lack of ground-truth. Manual annotation of extraction units is cost-prohibitive, whereas acquiring labels by automatically aligning human abstracts and source documents can yield inferior results. In this paper we describe a novel framework to guide a supervised, extractive summarization system with question-answering rewards. We argue that quality summaries should serve as a document surrogate to answer important questions, and such question-answer pairs can be conveniently obtained from human abstracts. The system learns to promote summaries that are informative, fluent, and perform competitively on question-answering. Our results compare favorably with those reported by strong summarization baselines as evaluated by automatic metrics and human assessors.

1 Introduction

Our increasingly digitized lifestyle calls for summarization techniques to produce short and accurate summaries that can be accessed at any time. These summaries should factually adhere to the content of the source text and present the reader with the key points therein. Although neural abstractive summarization has shown promising results (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017), these methods can have potential drawbacks. It was revealed that abstracts generated by neural systems sometimes alter or falsify objective details, and introduce new meanings not present in the original text (Cao et al., 2018). Reading these abstracts can lead to misinterpretation of the source materials, which is clearly undesirable. In this work, we focus on extractive summarization, where the summaries are guaranteed to remain faithful to the original content. Our system seeks to identify *salient and consecutive sequences of words* from the source document, and highlight them in the text to assist users in browsing and comprehending lengthy documents. An example is illustrated in Table 1.

Former TSA agent sentenced to six months in jail for restroom recording

(CNN) **A judge this week sentenced a former TSA agent to six months in jail for secretly videotaping a female co-worker** while she was in the bathroom, prosecutors said.

During the investigation, detectives with **the Metro Nashville Police Department in Tennessee** also found that the agent, 33-year-old Daniel Boykin, entered **the woman's** home multiple times, where he took videos, photos and other data.

Police found **more than 90 videos and 1,500 photos of the victim** on Boykin's phone and computer.

The victim filed a complaint after seeing images of herself on his phone last year. [...]

Comprehension Questions (Human Abstract):

Former ______ Daniel Boykin, 33, videotaped his female co-worker in the restroom, authorities say.

Authorities say they found 90 videos and 1,500 photos of the victim on ______ and computer.

Table 1: An example extractive summary bolded in the article (top). Highlighted sections indicate salient segments useful for answering fill-in-the-blank questions generated from human abstracts (bottom).

A primary challenge faced by extractive summarizers is the lack of annotated data. The cost of hiring humans to label a necessary amount of source articles with summary words, good for training a modern classifier, can be prohibitive. Previous work has exploited using human abstracts to derive labels for extraction units (Woodsend and Lapata, 2010). E.g., a source word is tagged 1 if it appears in the abstract, 0 otherwise. Although pairs of source articles and human abstracts are abundant, labels derived in this way are not necessarily best since summary saliency can not be easily captured with a rule based categorization. Considering that human abstracts involve generalization, paraphrasing, and can con-

Proceedings of NAACL-HLT 2019, pages 2566–2577
Minneapolis, Minnesota, June 2 - June 7, 2019.

tain words not present in the source text, leveraging them to derive labels for extraction units can be suboptimal. In this work, we investigate a new strategy that seeks to better utilize human abstracts to guide the extraction of summary text units.

We hypothesize that quality extractive summaries should contain informative content so that they can be used as document surrogates to answer important questions, thereby satisfying users' information needs. The question-answer pairs can be conveniently developed from human abstracts. Our proposed approach identifies answer tokens from each sentence of the human abstract, then replaces each answer token with a blank to create a Cloze-style question-answer pair. To answer *all* questions ($\approx$human abstract), the system summary must contain content that is semantically close to and collectively resembles the human abstract.

In this paper, we construct an extractive summary by selecting consecutive word sequences from the source document. To accomplish this we utilize a novel reinforcement learning framework to explore the space of possible extractive summaries and assess each summary using a novel reward function judging the summary's adequacy, fluency, length, and its competency to answer important questions. The system learns to sample extractive summaries yielding the highest expected rewards, with no pre-derived extraction labels needed. This work extends the methodology of Arumae and Liu (2018) with new representations of extraction units and thorough experimental evaluation. The contributions of this research can be summarized as follows:

- we describe a novel framework generating extractive summaries by selecting consecutive sequences of words from source documents. This new system explores various encoding mechanisms, as well as new sampling techniques to capture phrase level data. Such a framework has not been thoroughly investigated in the past;

- We conduct a methodical empirical evaluation from the point of view of information saliency. Rather than solely relying on automatic summarization evaluation methods, we also show the advantages of our system by assessing the summary quality with reading comprehension tasks. Our summaries compare favorably with the automatic metrics against state of the art, and show promising results against baselines when evaluated by humans for question answering.

2 Related Work

Extractive summarization has seen growing popularity in the past decades (Nenkova and McKeown, 2011). The methods focus on selecting representative sentences from the document(s) and optionally deleting unimportant sentence constituents to form a summary (Knight and Marcu, 2002; Radev et al., 2004; Zajic et al., 2007; Martins and Smith, 2009; Gillick and Favre, 2009; Lin and Bilmes, 2010; Wang et al., 2013; Li et al., 2013, 2014; Hong et al., 2014; Yogatama et al., 2015). A majority of the methods are unsupervised. They estimate sentence importance based on the sentence's length and position in the document, whether the sentence contains topical content and its relationship with other sentences. The summarization objective is to select a handful of sentences to maximize the coverage of important content while minimizing summary redundancy. Although unsupervised methods are promising, they cannot benefit from the large-scale training data harvested from the Web (Sandhaus, 2008; Hermann et al., 2015; Grusky et al., 2018).

Neural extractive summarization has focused primarily on extracting sentences (Nallapati et al., 2017; Cao et al., 2017; Isonuma et al., 2017; Tarnpradab et al., 2017; Zhou et al., 2018; Kedzie et al., 2018). These studies exploit parallel training data consisting of source articles and story highlights (i.e., human abstracts) to create ground-truth labels for sentences. A neural extractive summarizer learns to predict a binary label for each source sentence indicating if it is to be included in the summary. These studies build distributed sentence representations using neural networks (Cheng and Lapata, 2016; Yasunaga et al., 2017) and use reinforcement learning to optimize the evaluation metric (Narayan et al., 2018b) and improve summary coherence (Wu and Hu, 2018). However, sentence extraction can be coarse and in many cases, only a part of the sentence is worthy to be added to the summary. In this study, we perform finer-grained extractive summarization by allowing the system to select consecutive sequences of words rather than sentences to form a summary.

Interestingly, studies reveal that summaries generated by recent neural abstractive systems are, in fact, quite "extractive." Abstractive systems often adopt the encoder-decoder architecture with an attention mechanism (Rush et al., 2015; Nallapati et al., 2016; Paulus et al., 2017; Guo et al., 2018; Gehrmann et al., 2018; Lebanoff et al., 2018; Celikyilmaz et al., 2018). The encoder condenses a source sequence to a fixed-length vector and the decoder takes the vector as input and generates a summary by predicting one word at a time. See, Liu, and Manning (2017) suggest that about 35% of the summary sentences occur in the source documents, and 90% of summary n-grams appear in the source. Moreover, the summaries may contain inaccurate factual details and introduce new meanings not present in the original text (Cao et al., 2018; Song et al., 2018). It thus raises concerns as to whether such systems can be used in real-world scenarios to summarize materials such as legal documents. In this work, we choose to focus on extractive summarization where selected word sequences can be highlighted on the source text to avoid change of meaning.

Our proposed method is inspired by the work of Lei et al. (2016) who seek to identify rationales from textual input to support sentiment classification and question retrieval. Distinct from this previous work, we focus on generating generic document summaries. We present a novel supervised framework encouraging the selection of consecutive sequences of words to form an extractive summary. Further, we leverage reinforcement learning to explore the space of possible extractive summaries and promote those that are fluent, adequate, and competent in question answering. We seek to test the hypothesis that successful summaries can serve as document surrogates to answer important questions, and moreover, ground-truth question-answer pairs can be derived from human abstracts. In the following section we describe our proposed approach in details.

3 Our Approach

Let $\mathcal{S}$ be an extractive summary consisting of text segments selected from a source document $\mathbf{x}$. The summary can be mapped to a sequence of binary labels $\mathbf{y}$ assigned to document words. In this section we first present a supervised framework for identifying *consecutive sequences of words* that are summary-worthy, then proceed by describing our question-answering rewards and a deep reinforcement learning framework to guide the selection of summaries so that they can be used as document surrogates to answer important questions.[1]

3.1 Representing an Extraction Unit

How best to decompose a source document into a set of text units useful for extractive summarization remains an open problem. A natural choice is to use words as extraction units. However, this choice ignores the cohesiveness of text. A text chunk (e.g., a prepositional phrase) can be either selected to the summary in its entirety or not at all. In this paper we experiment with both schemes, using either *words* or *chunks* as extraction units. When a text chunk is selected in the summary, all its consisting words are selected. We obtain text chunks by breaking down the sentence constituent parse tree in a top-down manner until each tree fragment governs at most 5 words. A chunk thus can contain from 1 to 5 words. Additionally, word level modeling can be considered a special case of chunks where the length of each phrase is 1. It is important to note that using sentences as extraction units is out of the scope of this paper, because our work focuses on finer-grained extraction units such as words and phrases and this is notably a more challenging task.

The most successful neural models for encoding a piece of text to a fixed-length vector include the recurrent (Hochreiter and Schmidhuber, 1997) and convolutional neural networks (CNN; Kim et al., 2014), among others. A recent study by Khandelwal et al. (2018) reported that the recurrent networks are capable of memorizing a recent context of about 20 tokens and the model is highly sensitive to word order, whereas this is less the case for CNN whose max-pooling operation makes it agnostic to word order. We implement both networks and are curious to compare their effectiveness at encoding extraction units for summarization.

$$\{\mathbf{h}_t^e\} = f_1^{\text{Bi-LSTM}}(\mathbf{x}) \quad (1)$$

$$\text{or} \quad \{\mathbf{h}_t^e\} = f_2^{\text{CNN}}(\mathbf{x}) \quad (2)$$

Our model first encodes the source document using a bidirectional LSTM with the forward and backward passes (Eq. (1)). The representation of the t-th source word $\mathbf{h}_t^e = [\overleftarrow{\mathbf{h}}_t^e || \overrightarrow{\mathbf{h}}_t^e]$ is the concatenation of the hidden states in both directions. A chunk is similarly denoted by $\mathbf{h}_t^e = [\overleftarrow{\mathbf{h}}_t^e || \overrightarrow{\mathbf{h}}_{t+n}^e]$ where t and $t+n$ are the indices of its beginning and ending words. In both cases, a fixed-length vector ($\mathbf{h}_t^e \in \mathbb{R}^m$) is created for the word/chunk. Further, our CNN encoder (Eq. (2)) uses a sliding window of {1,3,5,7} words, corresponding to the kernel sizes, to scan through the source document. We apply a number of filters to each window size

---

[1] We have made our code and models available at `https://github.com/ucfnlp/summ_qa_rewards`

to extract local features. The t-th source word is represented by the concatenation of feature maps (an m-dimensional vector). To obtain the chunk vector we perform max-pooling over the representations of its consisting words (from t to $t+n$). In the following we use $\mathbf{h}^e_t$ to denote the vector representation of the t-th extraction unit, may it be a word or a chunk, generated using either encoder.

3.2 Constructing an Extractive Summary

It is desirable to first develop a supervised framework for identifying summary-worthy text segments from a source article. These segments collectively form an extractive summary to be highlighted on the source text. The task can be formulated as a sequence labeling problem: a source text unit (a word or chunk) is labelled 1 if it is to be included in the summary and 0 otherwise. It is not unusual to develop an auto-regressive model to perform sequence labeling, where the label of the t-th extraction unit (y_t) depends on all previous labels ($\mathbf{y}_{<t}$). Given this hypothesis, we build a framework to extract summary units where the importance of the t-th source unit is characterized by its informativeness (encoded in $\mathbf{h}^e_t$), its position in the document, and relationship with the partial summary. The details are presented below.

We use a positional embedding ($\mathbf{g}_t$) to signify the position of the t-th text unit in the source document. The position corresponds to the index of the source sentence containing the t-th unit, and further, all text units belonging to the same sentence share the same positional embedding. We apply sinusoidal initialization to the embeddings, following Vaswani et al. (2017). Importantly, positional embeddings allow us to inject macro-positional knowledge about words/chunks into a neural summarization framework to offset the natural bias that humans tend to have on putting important content at the beginning of an article.

Next, we build a representation for the partial summary to aid the system in selecting future text units. The representation $\mathbf{s}_t$ is expected to encode the extraction decisions up to time t-1 and it can be realized using a unidirectional LSTM network (Eq. (3)). The t-th input to the network is represented as $y_{t-1} \otimes \mathbf{h}^e_{t-1}$ where y_{t-1} is a binary label serving as a gating mechanism to control if the semantic content of the previous text unit ($\mathbf{h}^e_{t-1}$) is to be included in the summary ("$\otimes$" corresponds to elementwise product). During training, we apply teacher forcing and y_{t-1} is the ground-truth extraction label for the $(t-1)$-th unit; at test time, y_{t-1} is generated on-the-fly by obtaining the label yielding the highest probability according to Eq. (5). In the previous work of Cheng and Lapata (2016) and Nallapati et al. (2017), similar auto-regressive models are developed to identify summary sentences. Different from the previous work, this study focuses on extracting consecutive sequences of words and chunks from the source document, and the partial summary representation is particularly useful for predicting if the next unit is to be included in the summary to improve summary fluency.

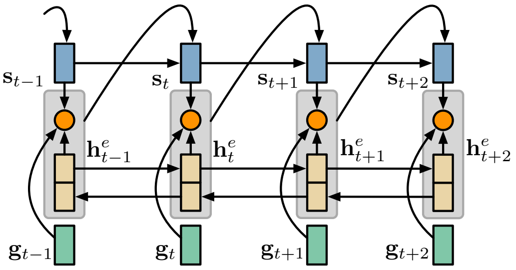


Figure 1: A unidirectional LSTM (blue, Eq. (3)) encodes the partial summary, while the multilayer perceptron network (orange, Eq. (4-5)) utilizes the text unit representation ($\mathbf{h}^e_t$), its positional embedding ($\mathbf{g}_t$), and the partial summary representation ($\mathbf{s}_t$) to determine if the t-th text unit is to be included in the summary. Best viewed in color.

$$\mathbf{s}_t = f_3^{\text{Uni-LSTM}}(\mathbf{s}_{t-1}, y_{t-1} \otimes \mathbf{h}^e_{t-1}) \quad (3)$$

Given the partial summary representation ($\mathbf{s}_t$), and representation of the text unit ($\mathbf{h}^e_t$) and its positional encoding ($\mathbf{g}_t$), we employ a multilayer perceptron to predict how likely the unit is to be included in the summary. This process is described by Eqs. (4-5) and further illustrated in Figure 1.

$$\mathbf{a}_t = f^{\text{ReLU}}(\mathbf{W}^a[\mathbf{h}^e_t; \mathbf{g}_t; \mathbf{s}_t] + \mathbf{b}^a) \quad (4)$$
$$p(y_t|\mathbf{y}_{<t}, \mathbf{x}) = \sigma(\mathbf{w}^y \mathbf{a}_t + b^y) \quad (5)$$

Our model parameters include $\{\mathbf{W}^a, \mathbf{b}^a, \mathbf{w}^y, b^y\}$ along with those required by $f_1^{\text{Bi-LSTM}}$, f_2^{CNN} and $f_3^{\text{Uni-LSTM}}$. It is possible to train this model in a fully supervised fashion by minimizing the negative log-likelihood of the training data. We generate ground-truth labels for source text units as follows. A source *word* receives a label of 1 if both itself and its adjacent word appear in the human abstract (excluding cases where both words are stopwords). This heuristic aims to label consecutive source words (2 or more) as summary-worthy, as opposed to picking single words which can be less informative. A source *text chunk* receives a label of 1 if one of its component words is labelled 1 in the above process.

Because human abstracts are often short and contain novel words not present in source documents, they can be suboptimal for generating ground-truth labels for extraction units. Only a small portion of the source words (about 8% in our dataset) are labelled as positive, whereas the vast majority are negative. Such labels can be ineffective in providing supervision. In the following section, we investigate a new learning paradigm, which encourages extractive summaries to contain informative content useful for answering important questions, while question-answer pairs can be automatically derived from human abstracts.

3.3 Using Summaries to Answer Questions

Our hypothesis is that high-quality summaries should contain informative content making them appropriate to serve as document surrogates to satisfy users' information needs. We train the extractive summarizer to identify source text units necessary for answering questions, and the question-answer (QA) pairs can be conveniently developed from human abstracts.

To obtain QA pairs, we set an answer token to be either a salient word or a named entity to limit the space of potential answers. For any sentence in the human abstract, we identify an answer token from it, then replace the answer token with a blank to create a Cloze-style question-answer pair (see Table 1). When a sentence contains multiple answer tokens, a set of QA pairs can be obtained from it. It is important to note that at least one QA pair should be extracted from each sentence of the abstract. Because a system summary is trained to contain content useful for answering *all* questions ($\approx$human abstract), any missing QA pair is likely to cause the summary to be insufficient.

We collect answer tokens using the following methods: (a) we extract a set of entities with tag {PER, LOC, ORG, MISC} from each sentence using the Stanford CoreNLP toolkit (Manning et al., 2014); (b) we also identify the ROOT word of each sentence's dependency parse tree along with the sentence's subject/object word, whose type is {NSUBJ, CSUBJ, OBJ, IOBJ} (if exists), then add them to the collection of answer tokens. Further, we prune the answer space by excluding those which appear fewer than 5 times overall. Having several methods for question construction allows us to explore the answer space properly. In the results section we perform experiments on root, subject/object, and named entities to see which model provides the best extraction guide.

Given an extractive summary $\mathcal{S}$ containing a set of source text units, and a collection of question-answer pairs $\mathcal{P} = \{(Q_k, e_k^*)\}_{k=1}^K$ related to the source document, we want to develop a mechanism leveraging the extractive summary to answer these questions. We first encode each question Q_k to a vector representation ($\mathbf{q}_k$). This is achieved by concatenating the last hidden states of the forward/backward passes of a bidirectional LSTM (Eq. (6)). Next, we exploit the attention mechanism to locate summary parts that are relevant to answering the k-th question. Given the attention mechanism, an extractive summary $\mathcal{S}$ can be used to answer multiple questions related to the document. We define $\alpha_{t,k}$ to be the semantic relatedness between the t-th source text unit and the k-th question. Following Chen et al. (2016a), we introduce a bilinear term to characterize their relationship ($\alpha_{t,k} \propto \mathbf{h}_t^e \mathbf{W}^\alpha \mathbf{q}_k$; see Eq. (7)). In this process, we consider only those source text units selected in summary $\mathcal{S}$. Using $\alpha_{t,k}$ as weights, we then compute a context vector $\mathbf{c}_k$ condensing summary content related to the k-th question (Eq. (8)).

$$\mathbf{q}_k = f_4^{\text{Bi-LSTM}}(Q_k) \tag{6}$$

$$\alpha_{t,k} = \frac{\exp(\mathbf{h}_t^e \mathbf{W}^\alpha \mathbf{q}_k)}{\sum_t \exp(\mathbf{h}_t^e \mathbf{W}^\alpha \mathbf{q}_k)} \tag{7}$$

$$\mathbf{c}_k = \textstyle\sum_t \alpha_{t,k} \mathbf{h}_t^e \tag{8}$$

$$\mathbf{u}_k = [\mathbf{c}_k; \mathbf{q}_k; |\mathbf{c}_k - \mathbf{q}_k|; \mathbf{c}_k \otimes \mathbf{q}_k] \tag{9}$$

To predict the most probable answer, we construct a fully-connected network as the output layer. The input to the network includes a concatenation of the context vector ($\mathbf{c}_k$), question vector ($\mathbf{q}_k$), absolute difference ($|\mathbf{c}_k - \mathbf{q}_k|$) and element-wise product ($\mathbf{c}_k \otimes \mathbf{q}_k$) of the two vectors (Eq. (9)). A softmax function is used to estimate a probability distribution over the space of candidate answers:

$$P(e_k|\mathcal{S}, Q_k) = \text{softmax}(\mathbf{W}^e f^{\text{ReLU}}(\mathbf{W}^u \mathbf{u}_k + \mathbf{b}^u)).$$

Such a fully-connected output layer has achieved success on natural language inference (Mou et al., 2016; Chen et al., 2018); here we test its efficacy on answer selection. The model parameters include $\{\mathbf{W}^\alpha, \mathbf{W}^e, \mathbf{W}^u, \mathbf{b}^u\}$ and those of $f_4^{\text{Bi-LSTM}}$.

3.4 A Reinforcement Learning Framework

In this section we introduce a reinforcement learning framework to explore the space of possible extractive summaries and present a novel reward function to promote summaries that are adequate, fluent, restricted in length, and competent in question answering. Our reward function consists of four components, whose interpolation weights γ, α, and β are tuned on the dev set.

$$\mathcal{R}(\mathbf{y}) = \mathcal{R}_c(\mathbf{y}) + \gamma\mathcal{R}_a(\mathbf{y}) + \alpha\mathcal{R}_f(\mathbf{y}) + \beta\mathcal{R}_l(\mathbf{y})$$

We define *QA competency* (Eq. (10)) as the average log-likelihood of correctly answering questions using the system summary ($\mathbf{y}$). A high-quality system summary is expected to resemble reference summary by using similar wording. The *adequacy* metric (Eq. (11)) measures the percentage of overlapping unigrams between the system ($\mathbf{y}$) and reference summary ($\mathbf{y}^*$). The *fluency* criterion (Eq. (12)) encourages consecutive sequences of source words to be selected by preventing many 0/1 switches in the label sequence (i.e., $|y_t - y_{t-1}|$). Finally, we limit the summary size by setting the ratio of selected words to be close to a threshold δ (Eq. (13)).

$$\text{QA} \quad \mathcal{R}_c(\mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K} \log P(e_k^* | \mathbf{y}, Q_k) \quad (10)$$

$$\text{Adequ.} \quad \mathcal{R}_a(\mathbf{y}) = \frac{1}{|\mathbf{y}^*|}\mathcal{U}(\mathbf{y}, \mathbf{y}^*) \quad (11)$$

$$\text{Fluency} \quad \mathcal{R}_f(\mathbf{y}) = -\sum_{t=2}^{|\mathbf{y}|} |y_t - y_{t-1}| \quad (12)$$

$$\text{Length} \quad \mathcal{R}_l(\mathbf{y}) = -\left|\frac{1}{|\mathbf{y}|}\sum_t y_t - \delta\right| \quad (13)$$

The reward function $\mathcal{R}(\mathbf{y})$ successfully combines intrinsic measures of summary fluency and adequacy (Goldstein et al., 2005) with extrinsic measure of summary responsiveness to given questions (Dang, 2006; Murray et al., 2008). A reinforcement learning agent finds a policy $P(\mathbf{y}|\mathbf{x})$ to maximize the expected reward $\mathbb{E}_{P(\mathbf{y}|\mathbf{x})}[\mathcal{R}(\mathbf{y})]$. Training the system with policy gradient (Eq. (14)) involves repeatedly sampling an extractive summary $\hat{\mathbf{y}}$ from the source document $\mathbf{x}$. At time t, the agent takes an action by sampling a decision based on $p(y_t|\hat{\mathbf{y}}_{<t}, \mathbf{x})$ (Eq. (5)) indicating whether the t-th source text unit is to be included in the summary. Once the full summary sequence $\hat{\mathbf{y}}$ is generated, it is compared to the ground-truth sequence to compute the reward $\mathcal{R}(\hat{\mathbf{y}})$. In this way, reinforcement learning explores the space of extractive summaries and promotes those yielding high rewards. At inference time, rather than sampling actions from $p(y_t|\mathbf{y}_{<t}, \mathbf{x})$, we choose y_t that yields the highest probability to generate the system summary $\mathbf{y}$. This process is deterministic and no QA is required.

$$\begin{aligned} &\nabla_\theta \mathbb{E}_{P(\mathbf{y}|\mathbf{x})}[\mathcal{R}(\mathbf{y})] \\ &= \mathbb{E}_{P(\mathbf{y}|\mathbf{x})}[\mathcal{R}(\mathbf{y})\nabla_\theta \log P(\mathbf{y}|\mathbf{x})] \\ &\approx \tfrac{1}{N}\textstyle\sum_{n=1}^{N} \mathcal{R}(\hat{\mathbf{y}}^{(n)})\nabla_\theta \log P(\hat{\mathbf{y}}^{(n)}|\mathbf{x}) \end{aligned} \quad (14)$$

4 Experiments

We proceed by discussing the dataset and settings, comparison systems, and experimental results obtained through both automatic metrics and human evaluation in a reading comprehension setting.

4.1 Dataset and Settings

Our goal is to build an extractive summarizer identifying important textual segments from source articles. To investigate the effectiveness of the proposed approach, we conduct experiments on the CNN/Daily Mail dataset using a version provided by See et al. (2017). The reference summaries of this dataset were created by human editors exhibiting a moderate degree of extractiveness. E.g., 83% of summary unigrams and 45% of bigrams appear in source articles (Narayan et al., 2018a). On average, a CNN article contains 761 words / 34 sentences and a DM article contains 653 words / 29 sentences. We report results respectively for the CNN and DM portion of the dataset.

Our hyperparameter settings are as follows. We set the hidden state dimension of the LSTM to be 256 in either direction. A bidirectional LSTM $f_1^{\text{Bi-LSTM}}(\cdot)$ produces a 512-dimensional vector for each content word. Similarly, $f_4^{\text{Bi-LSTM}}(\cdot)$ generates a question vector $\mathbf{q}_k$ of the same size. Our CNN encoder $f_2^{\text{CNN}}(\cdot)$ uses multiple window sizes of $\{1, 3, 5, 7\}$ and 128 filters per window size. $\mathbf{h}_t^e$ is thus a 512-dimensional vector using either CNN or LSTM encoder. We set the hidden state dimension of $\mathbf{s}_t$ to be 128. We also use 100-dimensional word embeddings (Pennington et al., 2014) and sinusoidal positional encodings (Vaswani et al., 2017) of 30 dimensions.

The maximum article length is set to 400 words. Compared to the study of Arumae and Liu (2018), we expand the search space dramatically from 100 to 400 words, which poses a challenge to the RL-based summarizers. We associate each article with at most 10 QA pairs (K=10) and use them to guide the extraction of summary segments. We apply mini-batch training with Adam optimizer (Kingma and Ba, 2014), where a mini-batch contains 128

System	#Ans.	CNN R-1	R-2	R-L
Lead-3	–	28.8	11.5	19.3
PointerGen+Cov.	–	29.9	10.9	21.1
Graph Attn.	–	30.3	9.8	20.0
LexRank	–	26.1	9.6	17.7
SumBasic	–	22.9	5.5	14.8
KLSum	–	20.7	5.9	13.7
Distraction-M3	–	27.1	8.2	18.7
QASumm+NoQ	0	16.38	7.25	11.30
QASumm+SUBJ/OBJ	9,893	26.16	8.97	18.24
QASumm+ROOT	3,678	26.67	9.19	18.76
QASumm+NER	6,167	**27.38**	**9.38**	**19.02**

Table 2: Summarization results on CNN test set. Summaries are evaluated at their full-length by ROUGE F_1-scores.

System	#Ans.	Daily Mail R-1	R-2	R-L
Lead-3	–	22.5	9.3	20.0
PointerGen+Cov.	–	31.2	17.0	28.9
Graph Attn.	–	27.4	11.3	15.1
NN-WE	–	15.7	6.4	9.8
NN-SE	–	22.7	8.5	12.5
SummaRuNNer	–	26.2	10.8	14.4
QASumm+NoQ	0	22.26	9.16	19.78
QASumm+SUBJ/OBJ	19,151	23.38	9.54	20.14
QASumm+ROOT	5,498	**26.87**	**11.97**	**23.07**
QASumm+NER	15,342	25.74	11.89	22.38

Table 3: Summarization results on DM test set. To ensure a fair comparison, we follow the convention to report ROUGE recall scores evaluated at 75 bytes.

articles and their QA pairs. The summary ratio δ is set to 0.15, yielding extractive summaries of about 60 words. Following Arumae and Liu (2018), we set hyperparameters $\beta = 2\alpha$; α and γ are tuned on the dev set using grid search.

4.2 Experimental Results

Comparison systems We compare our method with a number of extractive and abstractive systems that have reported results on the CNN/DM datasets. We consider non-neural approaches that extract sentences from the source article to form a summary. These include *LexRank* (Radev et al., 2004), *SumBasic* (Vanderwende et al., 2007), and *KLSum* (Haghighi and Vanderwende, 2009). Such methods treat sentences as bags of words, and then select sentences containing topically important words. We further include the Lead-3 baseline that extracts the first 3 sentences from any given article. The method has been shown to be a strong baseline for summarizing news articles.

Neural extractive approaches focus on learning vector representations for sentences and words, then performing extraction based on the learned representations. Cheng et al. (2016) describe a neural network method composed of a hierarchical document encoder and an attention-based extractor. The system has two variants: *NN-WE* extracts words from the source article and *NN-SE* extracts sentences. *SummaRuNNer* (Nallapati et al., 2017) presents an autoregressive sequence labeling method based on recurrent neural networks. It selects summary sentences based on their content, salience, position, and novelty representations.

Abstractive summarization methods are not directly comparable to our approach, but we choose to include three systems that report results respectively for CNN and DM datasets. *Distraction-M3* (Chen et al., 2016b) trains the summarization system to distract its attention to traverse different regions of the source article. *Graph attention* (Tan et al., 2017) introduces a graph-based attention mechanism to enhance the encoder-decoder framework. *PointerGen+Cov.* (See et al., 2017) allows the system to not only copy words from the source text but also generate summary words by selecting them from a vocabulary. Abstractive methods can thus introduce new words to the summary that are not present in the source article. However, system summaries may change the meaning of the original texts due to this flexibility.

Results We present summarization results of various systems in Tables 2 and 3, evaluated on the standard CNN/DM test sets by R-1, R-2, and R-L metrics (Lin, 2004), which respectively measure the overlap of unigrams, bigrams, and longest common subsequences between system and reference summaries. We investigate four variants of our method: *QASumm+NoQ* does not utilize any question-answer pairs during training. It extracts summary text chunks by learning from ground-truth labels (§3.2) and the chunks are encoded by $f_1^{\text{Bi-LSTM}}$. Other variants initialize their models using pretrained parameters from QASumm+NoQ, then integrate the reinforcement learning objective (§3.4) to exploit the space of possible extractive summaries and reward those that are useful for answering questions. We consider three types of QA pairs: the answer token is the root of a sentence dependency parse tree (+ROOT), a subject or object (+SUBJ/OBJ), or an entity found in the sentence (+NER). In all cases, the question is generated by replacing the answer token with a blank symbol.

As illustrated in Tables 2 and 3, our QASumm methods with reinforcement learning (+ROOT,

	NoText			QASumm+NoQ			GoldSumm			FullText		
	Train	Dev	Gap	Train	Dev	Gap	Train	Dev	Gap	Train	Dev	Gap
SUBJ/OBJ	49.7	24.4	25.3	55.9	31.2	24.7	69.3	48.6	20.7	67.6	43.3	24.3
ROOT	68.1	34.9	33.2	71.6	36.3	35.3	76.9	44.9	32.0	76.0	35.7	40.3
NER	61.0	15.8	45.2	66.0	32.7	33.3	85.2	54.0	31.2	82.4	46.3	36.1

Table 4: Question-answering accuracies using different types of QA pairs (ROOT, SUBJ/OBJ, NER) and different source input (NoText, QASumm+NoQ, GoldSumm, and FullText) as the basis for predicting answers.

+SUBJ/OBJ, +NER) perform competitively with strong baselines. They outperform the counterpart *QASumm+NoQ* that makes no use of the QA pairs by a substantial margin. They outperform or perform at a comparable level to state-of-the-art published systems on the CNN/DM datasets but are generally inferior to PointerGen. We observe that exacting summary chunks is highly desirable in real-world applications as it provides a mechanism to generate concise summaries. Nonetheless, accurately identifying summary chunks is challenging because the search space is vast and spuriousness arises in chunking sentences. Cheng and Lapata (2016) report a substantial performance drop when adapting their system to extract words. Our QASumm methods focusing on chunk extraction perform on par with competitive systems that extract whole sentences. We additionally present human evaluation results of summary usefulness for a reading comprehension task in §4.3.

In Tables 2 and 3, we further show the number of unique answers per QA type. We find that the ROOT-type QA pairs have the least number of unique answers. They are often main verbs of sentences. In contrast, the SUBJ/OBJ-type has the most number of answers. They are subjects and objects of sentences and correspond to an open class of content words. The NER-type has a moderate number of answers compared to others. Note that all answer tokens have been filtered by frequency; those appearing less than 5 times in the dataset are removed to avoid overfitting.

Among variants of the QASumm method, we find that *QASumm+ROOT* achieves the highest scores on DM dataset. *QASumm+NER* performs consistently well on both CNN and DM datasets, suggesting QA pairs of this type are effective in guiding the system to extract summary chunks. We conjecture that maintaining a moderate number of answers is important to maximize performance. To answer questions with missing entities, the summary is encouraged to contain similar content as the question body. Because questions are derived from the human abstract, this in turn requires the system summary to carry similar semantic content as the human abstract.

Question-answering accuracy We next dive into the QA component of our system to investigate question-answering performance when different types of summaries and QA pairs are supplied to the system (§3.3). Given a question, the system predicts an answer using an extractive summary as the source input. Intuitively, an informative summary can lead to high QA accuracy, as the summary content serves well as the basis for predicting answers. With the same summary as input, certain types of questions can be more difficult to answer than others, and the system must rely heavily on the summary to gauge correct answers.

We compare various types of summaries. These include (a) *QASumm+NoQ* which extracts summary chunks without requiring QA pairs; and (b) *GoldSumm*, which are gold-standard extractive summaries generated by collecting source words appearing in human summaries. We further consider *NoText* and *FullText*, corresponding to using no source text or the full source article as input. They represent the two extremes. In all cases the QA component (§3.3) is trained on the training set and we report QA accuracies on the dev set.

In Table 4, we observe that question-answering with *GoldSumm* performs the best for all QA types. It outperforms the scenarios using *FullText* as the source input. This indicates that distilled information contained in a high-quality summary can be useful for answering questions, as searching for answers in a succinct summary can be more efficient than that in a full article. Moreover, we observe that the performance of *QASumm+NoQ* is in between *NoText* and *GoldSumm* for all answer types. The results suggest that extractive summaries with even modest ROUGE scores can prove useful for question-answering. Regarding different types of QA pairs, we find that the ROOT-type can achieve high QA accuracy when using *NoText* input. It suggests that ROOT answers can to some extent be predicted based on the question context. The NER-type QA

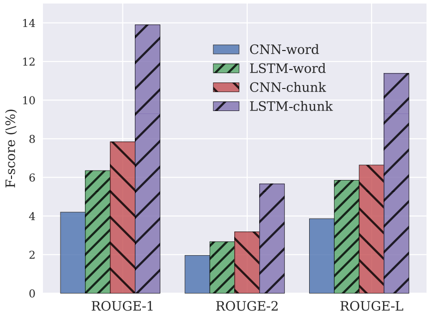


Figure 2: Summarization results using f_1^{LSTM} or f_2^{CNN} encoder with word/chunk as the extraction unit.

pairs work the best for both *GoldSumm* and *FullText*, likely because the source texts contain necessary entities required to correctly answer those questions. We also find the SUBJ/OBJ-type QA pairs have the smallest gap between train/dev accuracies, despite that they have a large answer space. Based on the analysis we would suggest future work to consider using NER-based QA pairs as they encourage the summaries to contain salient source content and be informative.

Extraction units We finally compare the performance of using either words or chunks as extraction units (§3.1). The chunks are obtained by breaking down sentence constituent parse trees in a top-down manner until all tree fragments contain 5 words or less. We observe that 70% of the chunks are 1-grams, and 2/3/4/5-grams are 9%, 7%, 6%, and 8% respectively. We compare the bidirectional LSTM (f_1^{LSTM}) and CNN (f_2^{CNN}) encoders for their effectiveness on generating representations for extraction units. Figure 2 presents the results of the QASumm+NoQ system under various settings. We find that extracting chunks performs superior, and combining chunks with LSTM representations yield the highest scores.

4.3 Human Evaluation

Testing the usefulness of an extractive system driven by reading comprehension is not inherently measured by automatic metrics (i.e. ROUGE). We conducted a human evaluation to assess whether the highlighted summaries contribute to document understanding. Similar to our training paradigm we presented each participant with the document and three fill-in-the-blank questions created from the human abstracts. It was guaranteed that each question was from a unique human abstract to avoid seeing the answer adjacent to the same template. The missing section was randomly generated to be either the root word, the subject or object of the sentence, or a named entity. We compare our reinforced extracted summary (presented as a bold overlay to the document), against our supervised method (section 3.2), abstractive summaries generated by See et al. (2017), and the human abstracts in full. Additionally we asked the participants to rate the quality of the summary presented (1-5, with 5 being most informative). We utilized Amazon Mechanical Turk, and conducted an experiment where we sampled 80 documents from the CNN test set. The articles were evenly split across the four competing systems, and each HIT was completed by 5 turkers. Upon completion the data was analyzed manually for accuracy since turkers entered each answer as free text, and to remove any meaningless datapoints.

Summary	Time	Accuracy	Inform.
Human	69.5s	87.3	4.23
QASumm+NoQ	87.9s	53.7	1.76
Pointer+Cov.	100.9s	52.3	2.14
QASumm+NER	95.0s	**62.3**	**2.14**

Table 5: Amazon mechanical turk experiments. Human abstracts were the goldstandard summaries, Pointer+Cov. were summaries generated by See et al. (2017). Our systems tested were the supervised extractor, and our full model (NER).

Table 5 shows the average time (in seconds) to complete a single question, the overall accuracy of the participants, and the informativeness of a given summary type. Excluding the use of human abstracts, all systems resulted in similar performance times. However we observe a large margin in QA accuracy in our full system compared to the abstractive and our supervised approach. Although participants rated the informativeness of the summaries to be the same our systems yielded a higher performance. This strongly indicates that having a system which makes using of document comprehension has a tangible effect when applied towards a real-world task.

5 Conclusion

We exploited an extractive summarization framework using deep reinforcement learning to identify consecutive word sequences from a document to form an extractive summary. Our reward function promotes adequate and fluent summaries that can serve as document surrogates to answer important questions, directly addressing users' information needs. Experimental results on benchmark datasets demonstrated the efficacy of our proposed method over state-of-the-art baselines, assessed by both automatic metrics and human evaluators.

References

Kristjan Arumae and Fei Liu. 2018. Reinforced extractive summarization with question-focused rewards. In *Proceedings of the ACL Student Research Workshop*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016a. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016b. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL*.

Hoa Trang Dang. 2006. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the NAACL Workshop on Integer Linear Programming for Natural Langauge Processing*.

Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss. 2005. Intrinsic and extrinsic evaluation measures for machine translation and/or summarization. In *Proceedings of the ACL-05 Workshop*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. NEWSROOM: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft, layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Neural Information Processing Systems (NIPS)*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chris Kedzie, Kathleen McKeown, and Hal Daume III. 2018. Content selection in deep learning models of summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gunhee Kim, Leonid Sigal, and Eric P. Xing. 2014. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Proceedings of CVPR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-document summarization by sentence compression based on expanded constituent parse tree. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of ACL Workshop on Text Summarization Branches Out*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *Proceedings of the Association for Computational Linguistics (ACL) System Demonstrations*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gabriel Murray, Thomas Kleinbauer, Peter Poller, Steve Renals, Jonathan Kilgour, and Tilman Becker. 2008. Extrinsic summarization evaluation: A decision audit task. *Machine Learning for Multimodal Interaction*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of SIGNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of EMNLP*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of* *the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sansiri Tarnpradab, Fei Liu, and Kien A. Hua. 2017. Toward extractive summarization of online forum discussions via hierarchical attention networks. In *Proceedings of the 30th Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of EMNLP*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Beyond Task Success:
A Closer Look at Jointly Learning to See, Ask, and GuessWhat

Ravi Shekhar[†], **Aashish Venkatesh**[*], **Tim Baumgärtner**[*], **Elia Bruni**[*],
Barbara Plank[♥], **Raffaella Bernardi**[†] and **Raquel Fernández**[*]
[†]University of Trento, [*]University of Amsterdam, [♥]IT University of Copenhagen
`ravi.shekhar@unitn.it aashishv@outlook.com`
`baumgaertner.t@gmail.com elia.bruni@gmail.com bapl@itu.dk`
`raffaella.bernardi@unitn.it raquel.fernandez@uva.nl`

Abstract

We propose a grounded dialogue state encoder which addresses a foundational issue on how to integrate visual grounding with dialogue system components. As a test-bed, we focus on the *GuessWhat?!* game, a two-player game where the goal is to identify an object in a complex visual scene by asking a sequence of yes/no questions. Our visually-grounded encoder leverages synergies between guessing and asking questions, as it is trained jointly using multi-task learning. We further enrich our model via a cooperative learning regime. We show that the introduction of both the joint architecture and cooperative learning lead to accuracy improvements over the baseline system. We compare our approach to an alternative system which extends the baseline with reinforcement learning. Our in-depth analysis shows that the linguistic skills of the two models differ dramatically, despite approaching comparable performance levels. This points at the importance of analyzing the linguistic output of competing systems beyond numeric comparison solely based on task success.[1]

1 Introduction

Over the last few decades, substantial progress has been made in developing dialogue systems that address the abilities that need to be put to work during conversations: Understanding and generating natural language, planning actions, and tracking the information exchanged by the dialogue participants. The latter is particularly critical since, for communication to be effective, participants need to represent the state of the dialogue and the common ground established through the conversation (Stalnaker, 1978; Lewis, 1979; Clark, 1996).

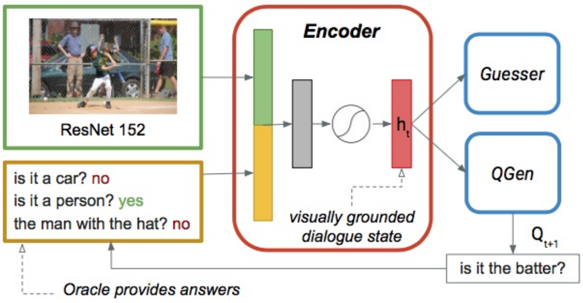


Figure 1: Our questioner model with a single visually grounded dialogue state encoder.

In addition to the challenges above, dialogue is often situated in a perceptual environment. In this study, we develop a dialogue agent that builds a representation of the context and the dialogue state by integrating information from both the *visual* and *linguistic* modalities. We take the *GuessWhat?!* game (de Vries et al., 2017) as our test-bed, a two-player game where a Questioner faces the task of identifying a target object in a visual scene by asking a series of yes/no questions to an Oracle. We model the agent in the Questioner's role.

To model the Questioner, previous work relies on two independent models to learn to ask questions and to guess the target object, each equipped with its own encoder (de Vries et al., 2017; Strub et al., 2017; Zhu et al., 2017; Lee et al., 2017; Shekhar et al., 2018; Zhang et al., 2018). We propose an end-to-end architecture with a single *visually-grounded dialogue state encoder* (cf. Figure 1). Our system is trained jointly in a supervised learning setup, extended with a cooperative learning (CL) regime: By letting the model play the game with self-generated dialogues, the components of the Questioner agent learn to better perform the overall Questioner's task in a cooperative manner. Das et al. (2017b) have explored the use of CL to train two visual dialogue agents that receive joint rewards when they play a game successfully. To our knowledge, ours is the first ap-

---

[1] Equal contribution by R. Shekhar and A. Venkatesh.

Proceedings of NAACL-HLT 2019, pages 2578–2587
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

proach where cooperative learning is applied to the internal components of a grounded conversational agent.

Our cooperative learning regime can be seen as an interesting alternative to reinforcement learning (RL)—which was first applied to *GuessWhat?!* by Strub et al. (2017)—because it is entirely differentiable and computationally less expensive to train than RL. Little is known on how this learning approach compares to RL not only regarding task success, but also in terms of the quality of the linguistic output, a gap we seek to fill in this paper. In particular, our contributions are:[2]

- The introduction of a single visually-grounded dialogue state encoder jointly trained with the guesser and question generator modules to address a foundational question of how to integrate visual grounding with dialogue system components; this yields up to 9% improvement on task success.
- The effectiveness of cooperative learning, which yields an additional increase of 8.7% accuracy, while being easier to train than RL.
- A first in-depth study to compare cooperative learning to a state-of-the-art RL system. Our study shows that the linguistic skills of the models differ dramatically, despite approaching comparable task success levels. This underlines the importance of linguistic analysis to complement solely numeric evaluation.

2 Related Work

Task-oriented dialogue systems The conventional architecture of task-oriented dialogue systems includes a pipeline of components, and the task of tracking the dialogue state is typically modelled as a partially-observable Markov decision process (Williams et al., 2013; Young et al., 2013; Kim et al., 2014) that operates on a symbolic dialogue state consisting of predefined variables. The use of symbolic representations to characterise the state of the dialogue has some advantages (e.g., ease of interfacing with knowledge bases), but it has also some key disadvantages: the variables to be tracked have to be defined in advance and the system needs to be trained on data annotated with explicit state configurations.

Given these limitations, there has been a shift towards neural end-to-end systems that learn their own representations. Early works focus on non-goal-oriented chatbots (Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a,b). Bordes et al. (2017) propose a memory network to adapt an end-to-end system to task-oriented dialogue. Recent works combine conventional symbolic with neural approaches (Williams et al., 2017; Zhao and Eskenazi, 2016; Rastogi et al., 2018), but all focus on language-only dialogue. We propose a visually grounded task-oriented end-to-end dialogue system which, while maintaining the crucial aspect of the interaction of the various modules at play in a conversational agent, grounds them through vision.

Visual dialogue agents In recent years, researchers in computer vision have proposed tasks that combine visual processing with dialogue interaction. Pertinent datasets created by Das et al. (2017a) and de Vries et al. (2017) include *VisDial* and *GuessWhat?!*, respectively, where two participants ask and answer questions about an image. While impressive progress has been made in combining vision and language, current models make simplifications regarding the integration of these two modalities and their exploitation for task-related actions. For example, the models proposed for *VisDial* by Das et al. (2017a) concern an image guessing game where one agent does not see the target image (thus, no multimodal understanding) and is required to 'imagine' it by asking questions. The other agent does see the image, but only responds to questions without the need to perform additional actions.

In *GuessWhat?!*, the Questioner agent sees an image and asks questions to identify a target object in it. The Questioner's role hence involves a complex interaction of vision, language, and guessing actions. Most research to date has investigated approaches consisting of different models trained independently (de Vries et al., 2017; Strub et al., 2017; Zhu et al., 2017; Lee et al., 2017; Shekhar et al., 2018; Zhang et al., 2018). We propose the first multimodal dialogue agent for the *GuessWhat?!* task where all components of the Questioner agent are integrated into a joint architecture that has at its core a *visually-grounded dialogue state encoder* (cf. Figure 1).

Reinforcement learning for visual dialogue agents was introduced by Das et al. (2017b) for

---

[2] Code and supplementary material are available at `https://vista-unitn-uva.github.io`.

VisDial and by Strub et al. (2017) for *GuessWhat?!*. Our joint architecture allows us to explore a simpler solution based on cooperative learning between the agent's internal modules (see Section 5 for details).

3 Task and Data

The *GuessWhat?!* game (de Vries et al., 2017) is a simplified instance of a referential communication task where two players collaborate to identify a referent—a setting used extensively in human-human collaborative dialogue (Clark and Wilkes-Gibbs, 1986; Yule, 1997; Zarrieß et al., 2016).

The *GuessWhat?!* dataset[3] was collected via Amazon Mechanical Turk by de Vries et al. (2017). The task involves two human participants who see a real-world image, taken from the MS-COCO dataset (Lin et al., 2014). One of the participants (the Oracle) is assigned a target object in the image and the other participant (the Questioner) has to guess it by asking Yes/No questions to the Oracle. There are no time constraints to play the game. Once the Questioner is ready to make a guess, the list of candidate objects is provided and the game is considered successful if the Questioner picks the target object. The dataset consists of around 155k English dialogues about approximately 66k different images. Dialogues contain on average 5.2 questions-answer pairs.

4 Models

We focus on developing an agent who plays the role of the Questioner in *GuessWhat?!*.

4.1 Baseline Model

As a baseline model (BL), we consider our own implementation of the best performing system put forward by de Vries et al. (2017). It consists of two independent models: a Question Generator (QGen) and a Guesser. For the sake of simplicity, QGen asks a fixed number of questions before the Guesser predicts the target object.

QGen is implemented as an Recurrent Neural Network (RNN) with a transition function handled with Long-Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), on which a probabilistic sequence model is built with a Softmax classifier. At each time step in the dialogue, the model receives as input the raw image and the dialogue history and generates the next question one word at a time. The image is encoded by extracting its VGG-16 features (Simonyan and Zisserman, 2014). In our new joint architecture (described below in Section 4.2), we use ResNet152 (He et al., 2016) features instead of VGG, because they tend to yield better performance in image classification and are more efficient to compute. For the baseline model it turns out that the original VGG-16 features lead to better performance (41.8% accuracy for VGG-16 vs. 37.3% with ResNet152 features). While we use ResNet152 features in our models, we keep the original VGG-16 feature configuration as de Vries et al. (2017), which constitutes a stronger baseline.

The Guesser model exploits the annotations in the MS-COCO dataset (Lin et al., 2014) to represent candidate objects by their object category and their spatial coordinates. This yields better performance than using raw image features in this case, as reported by de Vries et al. (2017). The objects' categories and coordinates are passed through a Multi-Layer Perceptron (MLP) to get an embedding for each object. The Guesser also takes as input the dialogue history processed by its own dedicated LSTM. A dot product between the hidden state of the LSTM and each of the object embeddings returns a score for each candidate object.

The model playing the role of the Oracle is informed about the target object o_{target}. Like the Guesser, the Oracle does not have access to the raw image features. It receives as input embeddings of the target object's category, its spatial coordinates, and the current question asked by the Questioner, encoded by a dedicated LSTM. These three embeddings are concatenated and fed to an MLP that gives an answer (Yes or No).

4.2 Visually-Grounded Dialogue State Encoder

In line with the baseline model, our Questioner agent includes two sub-modules, a QGen and a Guesser. As in the baseline, the Guesser guesses after a fixed number of questions, which is a parameter tuned on the validation set. Our agent architecture differs from the baseline model by de Vries et al.: Rather than operating independently, the language generation and guessing modules are connected through a common *grounded dialogue state encoder* (GDSE) which combines linguistic and visual information as a prior for the two modules. Given this representation, we will

---

[3] Dataset: `https://guesswhat.ai/download`.

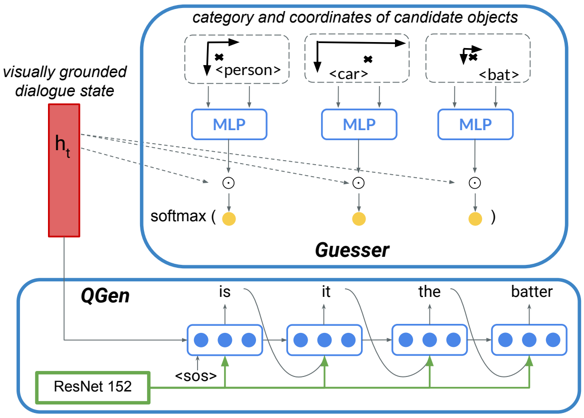


Figure 2: Question Generation and Guesser modules.

refer to our Questioner agent as GDSE.

As illustrated in Figure 1, the encoder receives as input representations of the visual and linguistic context. The visual representation consists of the second to last layer of ResNet152 trained on ImageNet. The linguistic representation is obtained by an LSTM (LSTM$_e$) which processes each new question-answer pair in the dialogue. At each question-answer QA_t, the last hidden state of LSTM$_e$ is concatenated with the image features I, passed through a linear layer and a *tanh* activation to result in the final layer h_t:

$$h_t = \tanh\left(W \cdot [\text{LSTM}_e(qa_{1:t-1});\ I]\right) \qquad (1)$$

where $[\cdot;\cdot]$ represents concatenation, $I \in \mathbb{R}^{2048\times 1}$, LSTM$_e$ $\in \mathbb{R}^{1024\times 1}$ and $W \in \mathbb{R}^{512\times 3072}$ (identical to prior work except for tuning the ResNet-specific parameters). We refer to this final layer as the *dialogue state*, which is given as input to both QGen and Guesser.

As illustrated in Figure 2, our QGen and Guesser modules are like the corresponding modules by de Vries et al. (2017), except for the crucial fact that they receive as input the same grounded dialogue state representation. QGen employs an LSTM (LSTM$_q$) to generate the token sequence for each question conditioned on h_t, which is used to initialise the hidden state of LSTM$_q$. As input at every time step, QGen receives a dense embedding of the previously generated token w_{i-1} and the image features I:

$$p(w_i) = p(w_i | w_1, ..., w_{i-1}, h_t, I) \qquad (2)$$

We optimise QGen by minimising the Negative Log Likelihood (NLL) of the human dialogues and use the Adam optimiser (Kingma and Ba, 2015):

$$\mathcal{L}_Q = \sum_i - \log p(w_i) \qquad (3)$$

Thus, in our architecture the LSTM$_q$ of QGen in combination with the LSTM$_e$ of the Encoder form a sequence-to-sequence model (Sutskever et al., 2014), conditioned on the visual and linguistic context — in contrast to the baseline model, where question generation is performed by a single LSTM on its own.

The Guesser consists of an MLP which is evaluated for each candidate object in the image. It takes the dense embedding of the category and the spatial information of the object to establish a representation $r_j \in \mathbb{R}^{512\times 1}$ for each object. A score is calculated for each object by performing the dot product between the dialogue state h_t and the object representation. Finally, a softmax over the scores results in a probability distribution over the candidate objects:

$$p(o_j) = \frac{e^{h_t^T \cdot r_j}}{\sum_j e^{h_t^T \cdot r_j}} \qquad (4)$$

We pick the object with the highest probability and the game is successful if $o_{guess} = o_{target}$, where $o_{guess} = \arg\max_j p(o_j)$. As with QGen, we optimise the Guesser by minimising the NLL and again make use of Adam:

$$\mathcal{L}_G = -\log p(o_{target}) \qquad (5)$$

The resulting architecture is fully differentiable. In addition, the GDSE agent faces a multi-task optimisation problem: While the QGen optimises $\mathcal{L}_Q$ and the Guesser optimises $\mathcal{L}_G$, the parameters of the Encoder (W, LSTM$_e$) are optimised via both $\mathcal{L}_Q$ and $\mathcal{L}_G$. Hence, both tasks faced by the Questioner agent contribute to the optimisation of the dialogue state h_t, and thus to a more effective encoding of the input context.

5 Learning Approach

We first introduce the supervised learning approach used to train both BL and GDSE, then our cooperative learning regime, and finally the reinforcement learning approach we compare to.

5.1 Supervised Learning

In the baseline model, the QGen and the Guesser modules are trained autonomously with supervised learning (SL): QGen is trained to replicate human questions and, independently, the Guesser is trained to predict the target object. Our new architecture with a common dialogue state encoder allows us to formulate these two tasks as a multi-task problem, with two different losses (Eq. 3 and 5 in Section 4.2). These two tasks are not equally difficult: While the Guesser has to learn the probability distribution of the set of possible objects in the image, QGen needs to fit the distribution of natural language words. Thus, QGen has a harder task to optimize and requires more parameters and training iterations. We address this issue by making the learning schedule task-dependent. We call this setup *modulo-n* training, where *n* indicates after how many epochs of QGen training the Guesser is updated together with QGen.

Using the validation set, we experimented with *n* from 5 to 15 and found that updating the Guesser every 7 epochs worked best. With this optimal configuration, we then train GDSE for 100 epochs (batch size of 1024, Adam, learning rate of 0.0001) and select the Questioner module best performing on the validation set (henceforth, GDSE-SL or simply SL).

5.2 Cooperative Learning

Once the model has been trained with SL, new training data can be generated by letting the agent play new games. Given an image from the training set used in the SL phase, we generate a new training instance by randomly sampling a target object from all objects in the image. We then let our Questioner agent and the Oracle play the game with that object as target, and further train the common encoder using the generated dialogues by backpropagating the error with gradient descent through the Guesser. After training the Guesser and the encoder with generated dialogues, QGen needs to 'readapt' to the newly arranged encoder parameters. To achieve this, we re-train QGen on the human data with SL, but using the new encoder states. Also here, the error is backpropagated with gradient descent through the common encoder.

Regarding *modulo-n*, in this case QGen is updated at every n^{th} epoch, while the Guesser is updated at all other epochs; we experimented with *n* from 3-7 and set it to the optimal value of 5. The GDSE previously trained with SL is further trained with this cooperative learning regime for 100 epochs (batch size of 256, Adam, learning rate of 0.0001), and we select the Questioner module performing best on the validation set (henceforth, GDSE-CL or simply CL).

5.3 Reinforcement Learning

Strub et al. (2017) proposed the first extension of BL (de Vries et al., 2017) with deep reinforcement learning (RL). They present an architecture for end-to-end training using an RL policy. First, the Oracle, Guesser, and QGen models are trained independently using supervised learning. Then, QGen is further trained using a policy gradient.

We use the publicly available code and pre-trained model based on Sampling (Strub et al., 2017), which resulted in the closest performance to what was reported by the authors.[4] This is the RL model we use throughout the rest of the paper.

5.4 Experimental Details

We use the same train (70%), validation (15%), and test (15%) splits as de Vries et al. (2017). The test set contains new images not seen during training. We use two experimental setups for the number of questions to be asked by the question generator, motivated by prior work: 5 questions (5Q) following de Vries et al. (2017), and 8 questions (8Q) as in Strub et al. (2017). As noted in Section 3, on average, there are 5.2 questions per dialogue in the *GuessWhat?!* data set.

For evaluation, we report task success in terms of accuracy (Strub et al., 2017). To neutralize the effect of random sampling in training CL, we trained the model 3 times. RL is tested 3 times with sampling. We report means and standard deviation (for some tables these are provided in the supplementary material; see footnote 2).

6 Results

Table 1 reports the results for all models. There are several take-aways.

Grounded joint architecture First of all, our visually-grounded dialogue state encoder is effective. GDSE-SL outperforms the baseline by de Vries et al. (2017) significantly in both setups (absolute accuracy improvements of 6.6%

---

[4] Their result of 53.3% accuracy published in Strub et al. (2017) is obsolete, as stated on their GitHub page (`https://github.com/GuessWhatGame/guesswhat`) where they report 56.5% for sampling and 58.4% for greedy search. By running their code, we could only replicate their results with sampling, obtaining 56%, while greedy and beam search resulted in similar or worse performance. Our analysis showed that greedy and beam search have the additional disadvantage of learning a smaller vocabulary.

Model	5Q	8Q
Baseline	41.2	40.7
GDSE-SL	47.8	49.7
GDSE-CL	53.7 (±.83)	58.4 (±.12)
RL	56.2 (±.24)	56.3 (±.05)

Table 1: Test set accuracy for each model (for setups with 5 and 8 questions). GDSE-SL is our grounded supervised learning system, GDSE-CL the cooperative learning setup, and RL the results we obtain with the reinforcement learning system by Strub et al. (2017).

and 9%). To evaluate the impact of the multi-task learning aspect, we did an ablation study and used the encoder-decoder architecture to train the QGen and Guesser modules independently. With such a decoupled training we obtain lower results: 44% and 43.7% accuracy for 5Q and 8Q, respectively. Hence, the multi-task component brings an increase of up to 6% over the baseline.[5]

Cooperative learning and RL The introduction of the cooperative learning approach results in a clear improvement over GDSE-SL: +8.7% (8Q: from 49.7 to 58.4) and +5.9% (with 5Q). Despite its simplicity, our GDSE-CL model achieves a task success rate which is comparable to RL: In the 8Q setup, GDSE-CL reaches an average accuracy of 58.4 versus 56.3 for RL, giving CL a slight edge in this setup (+2.1%), while in the 5Q setup RL is slightly better (+2.5%). Overall, the accuracy of the CL and RL models is close. The interesting question is how the linguistic skills and strategy of these two models differ, to which we turn in the next section.

We compared to Strub et al. (2017), but RL has also been put forward by Zhang et al. (2018), who report 60.7% accuracy (5Q). This result is close to our highest GDSE-CL result (60.8 ±0.51, when optimized for 10Q).[6] Their RL system integrates several partial reward functions to increase coherence, which is an interesting aspect. Yet their code is not publicly available. We leave the comparison to Zhang et al. (2018) and adding RL to GDSE to future work.

7 Analysis

In this section, we present a range of analyses that aim to shed light on the performance of the models. They are carried out on the test set data using the 8Q setting, which yields better results than the 5Q setting for the GDSE models and RL. Given that there is only a small difference in accuracy for the baseline with 5Q and 8Q, for comparability we analyse dialogues with 8Q also for BL.

7.1 Quantitative Analysis of Linguistic Output

We analyse the language produced by the Questioner agent with respect to three factors: (1) lexical diversity, measured as type/token ratio over all games, (2) question diversity, measured as the percentage of unique questions over all games, and (3) the number of games with questions repeated verbatim. We compute these factors on the test set for the models and for the human data (H).

As shown in Table 2, the linguistic output of SL & CL is closer to the language used by humans: Our agent is able to produce a much richer and less repetitive output than both BL and RL. In particular, it learns to use a more diverse vocabulary, generates more unique questions, and repeats questions within the same dialogue at a much lower rate than the baseline and RL: 93.5% of the games played by BL contain at least one verbatim question repetition, for RL this happens in 96.47% of the cases, whereas for SL and CL this is for only 55.8% and 52.19% of the games, respectively.

	Lexical diversity	Question diversity	% Games with repeated Q's
BL	0.030	1.60	93.50
SL	0.101	13.61	55.80
CL	0.115 (±.02)	14.15 (±3.0)	52.19 (±4.7)
RL	0.073 (±.00)	1.04 (±.03)	96.47 (±.04)
H	0.731	47.89	—

Table 2: Statistics of the linguistic output of all models with the 8Q setting and of humans (H) in all test games.

7.2 Dialogue Strategy

To further understand the variety of questions asked by the agents, we classify questions into different types. We distinguish between questions that aim at getting the category of the target object (ENTITY questions, e.g., *'is it a vehicle?'*) and questions about properties of the queried objects (ATTRIBUTE questions, e.g., *'is it square?'* or *'are*

---

[5] While de Vries et al. (2017) originally report an accuracy of 46.8%, this result was later revised to 40.8%, as clarified on their GitHub page. Our own implementation of the baseline system achieves an accuracy of 41.2%.

[6] Since our aim is to compare to the best setup for BL (5Q) and RL (8Q), we do not report our results with 10Q in Table 1.

Humans	*[success]*	**GDSE-CL**	*[success]*	**RL**	*[failure]*
1. does it have cereal on it?	no	1. is it a donut?	yes	1. is it food?	yes
2. does it have pink frosting?	no	2. is it on the left?	no	2. is it a donut?	yes
3. does it have cookies?	no	3. is it on the right?	yes	3. is it in left?	no
4. is it a donut?	yes	4. is it the whole donut?	yes	4. is it in top?	yes
5. does it have maple frosting?	no	5. does it have pink icing?	no	5. is it in top?	yes
6. is there candy?	yes	6. is it touching the donut with the sprinkles?	yes	6. is it in top?	yes
7. is it the doughnut with M&Ms?	no	7. does it have chocolate icing?	no	7. is it in top?	yes
8. is it dark brown?	no	8. is it the third donut?	yes	8. is it top?	yes

Figure 3: Game example where GDSE-CL succeeds and RL fails at guessing the target object (framed).

they standing?'). Within ATTRIBUTE questions, we make a distinction between color, shape, size, texture, location, and action questions. Within ENTITY questions, we distinguish questions whose focus is an object category or a super-category (see the supplementary material for example questions). The classification is done by manually extracting keywords for each question type from the human dialogues, and then applying an automatic heuristic that assigns a class to a question given the presence of the relevant keywords.[7] This procedure allows us to classify 91.41% of the questions asked by humans. The coverage is higher for the questions asked by the models: 98.88% (BL), 94.72% (SL), 94.11% (CL) and 99.51 % (RL).[8]

The statistics are shown in Table 3. We use Kullback-Leibler (KL) divergence to measure how the output of each model differs from the human distribution of fine-grained question classes. The baseline's output has the highest degree of divergence: For instance, the BL model does never ask any SHAPE or TEXTURE questions, and hardly any SIZE questions. The output of the RL model also differs substantially from the human dialogues: It asks a very large number of LOCATION questions (74.8% vs. 40% for humans). Our model, in contrast, generates question types that resemble the human distribution more closely.

Question type	BL	SL	CL	RL	H
ENTITY	**49.00**	**48.07**	**46.51**	**23.99**	**38.11**
SUPER-CAT	19.6	12.38	12.58	14.00	14.51
OBJECT	29.4	35.70	33.92	9.99	23.61
ATTRIBUTE	**49.88**	**46.64**	**47.60**	**75.52**	**53.29**
COLOR	2.75	13.00	12.51	0.12	15.50
SHAPE	0.00	0.01	0.02	0.003	0.30
SIZE	0.02	0.33	0.39	0.024	1.38
TEXTURE	0.00	0.13	0.15	0.013	0.89
LOCATION	47.25	37.09	38.54	74.80	40.00
ACTION	1.34	7.97	7.60	0.66	7.59
Not classified	**1.12**	**5.28**	**5.90**	**0.49**	**8.60**
KL (wrt human)	0.953	0.042	0.038	0.396	0.0

Table 3: Percentage of questions per question type in all the test set games played by humans (H) and the models with the 8Q setting, and KL divergence from human distribution of fine-grained question types.

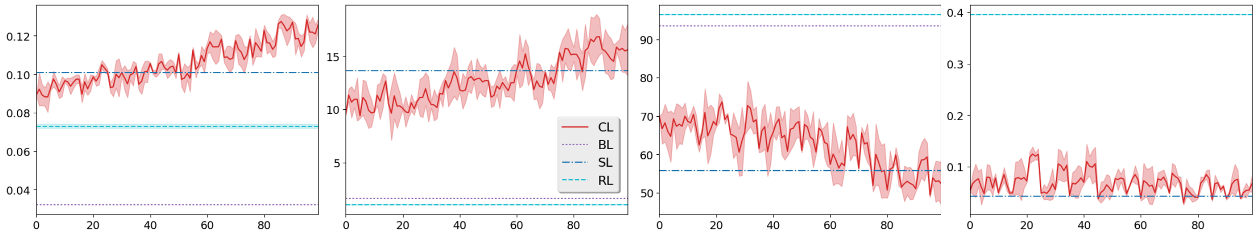
We also analyse the structure of the dialogues in terms of the sequences of question types asked. As expected, both humans and models almost always start with an ENTITY question (around 97% for BL, SL and CL, 98.7% for RL, and 78.48% for humans), in particular a SUPER-CATEGORY (around 70% for BL, SL and CL, 84% for RL, and 52.32% for humans). In some cases, humans start by asking questions directly about an attribute that may easily distinguish an object from others, while this is very uncommon for models. Figure 3 shows an example: The human dialogue begins with an ATTRIBUTE question (*'does it have cereal on it?'*), which in this case is not very effective and leads to a change in strategy at turn 4. The CL model starts by asking an OBJECT question (*'is it a donut?'*) while the RL model begins with a more generic SUPER-CATEGORY question (*'is it food?'*).

We check how the answer to a given question type affects the type of the follow-up question. In principle, we expect to find that question types that are answered positively will be followed by more specific questions. This is indeed what we observe in the human dialogues, as shown in Table 4. For example, when a SUPER-CATEGORY question is answered positively, humans follow up with an OBJECT or ATTRIBUTE question 89.56% of the time. This trend is mirrored by all models. Overall, the models also learn the strategy to move from an OBJECT to an ATTRIBUTE question when an OBJECT question receives a Yes answer. The BL, SL, and CL models do this to a lesser extent than humans, while the RL model systematically

---

[7] A question may be tagged with several attribute classes if keywords of different types are present. E.g., *"Is it the white one on the left?"* is classified as both COLOR and LOCATION.

[8] In the supplementary material we provide details on the question classification procedure: the lists of keywords by class, the procedure used to obtain these lists, as well as the pseudo-code of the heuristics used to classify the questions.



(a) Lexical diversity (b) Question diversity (c) % Games w/ repeated Q's (d) KL-distance from human

Figure 4: Evolution of linguistic factors over 100 training epochs for our GDSE-CL model. Note: lexical and question diversity of the human data fall outside the range in (a) / (b). The same is the case with KL for BL in (d).

Question type shift	BL	SL	CL	RL	H
SUPER-CAT → OBJ/ATT	89.05	92.61	89.75	95.63	89.56
OBJECT → ATTRIBUTE	67.87	60.92	65.06	99.46	88.70

Table 4: Proportion of question type shift vs. no type shift in consecutive questions $Q_t \rightarrow Q_{t+1}$ where Q_t has received a Yes answer.

transitions to attributes (in 99.46% of cases), using mostly LOCATION questions as pointed out above. For example (Figure 3), after receiving an affirmative answer to the OBJECT question 'is it a donut?' both CL and RL shift to a LOCATION question. Once location is established, CL moves on to other attributes while RL keeps asking the same LOCATION question, which leads to failure. Further illustrative examples are given in the supplementary material.

7.3 Analysis of the CL Learning Process

In order to better understand the effect of the cooperative learning regime, we trace the evolution of linguistic factors identified above over the CL epochs. As illustrated in Figure 4 (a) and (b), through the epochs the CL model learns to use a richer vocabulary and more diverse questions, moving away from the levels achieved by BL and RL, overpassing SL and moving toward humans.

The CL model progressively produces fewer repeated questions within a dialogue, improving over SL in the last few epochs, cf. Figure 4 (c). Finally, (d) illustrates the effect of modulo-n training: As the model is trained on generated dialogues, its linguistic output drifts away from the human distribution of question types; every 5^{th} epoch QGen is trained via supervision, which brings the model's behaviour closer back to human linguistic style and helps decrease the drift.

8 Conclusion

We present a new visually-grounded joint Questioner agent for goal-oriented dialogue. First, we show that our architecture archives 6–9% accuracy improvements over the *GuessWhat?!* baseline system (de Vries et al., 2017). This way, we address a foundational limitation of previous approaches that model guessing and questioning separately.

Second, our joint architecture allows us to propose a two-phase cooperative learning approach (CL), which further improves accuracy. It results in our overall best model and reaches state-of-the-art results (cf. Section 6). We compare CL to the system proposed by Strub et al. (2017) which extends the baseline with reinforcement learning (RL). We find that the two approaches (CL and RL) achieve overall relatively similar task success rates. However, evaluating on task success is only one side of the coin. Finally and most importantly, we propose to pursue an in-depth analysis of the quality of the dialogues by visual conversational agents, which is an aspect often neglected in the literature. We analyze the linguistic output of the two models across three factors (lexical diversity, question diversity, and repetitions) and find them to differ substantially. The CL model uses a richer vocabulary and inventory of questions, and produces fewer repeated questions than RL. In contrast, RL highly relies on asking location questions, which might be explained by a higher reliance on spatial and object-type information explicitly given to the Guesser and Oracle models. Limiting rewards to task success or other rewards not connected to the language proficiency does not stimulate the model to learn rich linguistic skills, since a reduced vocabulary and simple linguistic structures may be an efficient strategy to succeed at the game.

Overall, the presence of repeated questions remains an important weakness of all models, resulting in unnatural dialogues. This shows that there is still a considerable gap to human-like conversational agents. Looking beyond task success can provide a good basis for extensions of current architectures, e.g., Shekhar et al. (2018) add a decision-making component that decides when to stop asking questions which results in less repetitive and more human-like dialogues. Our joint architecture could easily be extended with such a component.

Acknowledgements

The work carried out by the Amsterdam team was partially funded by the Netherlands Organisation for Scientific Research (NWO) under VIDI grant nr. 276-89-008, *Asymmetry in Conversation*. We thank the University of Trento for generously funding a research visit by Barbara Plank to CIMeC that led to part of the work presented in this paper. In addition, we kindly acknowledge the support of NVIDIA Corporation with the donation to the University of Trento of the GPUs used in our research.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal oriented dialog. In *Proceedings of ICLR*.

Herbert H Clark. 1996. *Using Language*. Cambridge University Press.

Herbert H Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision (ICCV)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dongho Kim, Catherine Breslin, Pirros Tsiakoulis, M Gašić, Matthew Henderson, and Steve Young. 2014. Inverse reinforcement learning for micro-turn management. In *Fifteenth Annual Conference of the International Speech Communication Association*.

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Sang-Woo Lee, Yujung Heo, and Byoung-Tak Zhang. 2017. Answerer in questioner's mind for goal-oriented visual dialogue. In *NIPS Workshop on Visually-Grounded Interaction and Language (ViGIL)*.

David Lewis. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8(1):339–359.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-2016*, pages 110–119.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of EMNLP*.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, Dollar, P., and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of ECCV (European Conference on Computer Vision)*.

Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of SIGdial*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Ravi Shekhar, Tim Baumgärtner, Aashish Venkatesh, Elia Bruni, Raffaella Bernardi, and Raquel Fernández. 2018. Ask no more: Deciding when to guess in referential visual dialogue. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1218–1233.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. Technical Report arXiv:1409.1556, Visual Geometry Group, University of Oxford.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015.

A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*, pages 196–205.

Robert Stalnaker. 1978. Assertion. In P. Cole, editor, *Pragmatics*, volume 9 of *Syntax and Semantics*. New York Academic Press.

Florian Strub, Harm de Vries, Jeremie Mary, Bilal Piot, Aaron Courville, and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! Visual object discovery through multi-modal dialogue. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jason Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of ACL-2017*. Association for Computational Linguistics.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5).

George Yule. 1997. *Referential communication tasks*. Routledge.

Sina Zarrieß, Julian Hough, Casey Kennington, Ramesh Manuvinakurike, David DeVault, Raquel Fernández, and David Schlangen. 2016. Pentoref: A corpus of spoken references in task-oriented dialogues. In *10th edition of the Language Resources and Evaluation Conference*.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton van den Hengel. 2018. Goal-oriented visual question generation via intermediate rewards. In *Proceedings of the European Conference of Computer Vision (ECCV)*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state traching and management using deep reinforcement learning. In *Proceedings of SIGDIAL-2016*.

Yan Zhu, Shaoting Zhang, and Dimitris Metaxas. 2017. Interactive reinforcement learning for object grounding via self-talking. In *NIPS Workshop on Visually-Grounded Interaction and Language (ViGIL)*.

The World in My Mind: Visual Dialog with Adversarial Multi-modal Feature Encoding

Yiqun Yao[123], **Jiaming Xu**[12*] **and Bo Xu**[1234]
[1]Institute of Automation, Chinese Academy of Sciences (CASIA). Beijing, China
[2]Research Center for Brain-inspired Intelligence, CASIA
[3]University of Chinese Academy of Sciences
[4]Center for Excellence in Brain Science and Intelligence Technology, CAS. China
`{yaoyiqun2014,jiaming.xu,xubo}@ia.ac.cn`

Abstract

Visual Dialog is a multi-modal task that requires a model to participate in a multi-turn human dialog grounded on an image, and generate correct, human-like responses. In this paper, we propose a novel Adversarial Multi-modal Feature Encoding (AMFE) framework for effective and robust auxiliary training of visual dialog systems. AMFE can force the language-encoding part of a model to generate hidden states in a distribution closely related to the distribution of real-world images, resulting in language features containing general knowledge from both modalities by nature, which can help generate both more correct and more general responses with reasonably low time cost. Experimental results show that AMFE can steadily bring performance gains to different models on different scales of data. Our method outperforms both the supervised learning baselines and other fine-tuning methods, achieving state-of-the-art results on most metrics of VisDial v0.5/v0.9 generative tasks.

1 Introduction

In recent years, there has been a rising attention in Artificial Intelligence on how to train a model to understand visual inputs from the physical world, and communicate them with human language. Typical problems include Visual Question Answering (VQA) (Antol et al., 2015) and Image Captioning (Xu et al., 2015). These tasks require a model to read an image and generate a proper response, such as answering a question grounded on the image, or generating a sentence to describe the image. As a more difficult extension, Visual Dialog (De Vries et al., 2017; Das et al., 2017a; Mostafazadeh et al., 2017) is a cluster of tasks featuring two agents conducting a multi-turn dialog grounded on an image. A model is usually trained to predict every single response of one agent in the two, based on the image and dialog history. There are also some different task settings such as directly training two agents to complete a goal-driven cooperative task such as Guessing Game (Das et al., 2017b).

Tasks involving both the physical world (visual images) and abstract world (languages) share a core issue: how to establish connections between these two worlds, and is there a framework to leverage these connections for learning? Temporarily, the majority of answers are learning end-to-end models with multi-modal feature fusion (Kim et al., 2016; Fukui et al., 2016; Yu et al., 2018). These methods usually merge the visual and language features into rich representations containing information from both sides. Some cross-modal attention methods (Lu et al., 2016; Nam et al., 2017) formulate the visual-language connections explicitly by parameterizing the attention weights to learn whether there is high correlation within certain pairs of language and visual feature vectors. However, in all these works, the merged representations or attention weights are only learned from pairwise (one image, one sentence) co-occurrence, and serve for the optimization of a loss function only related to the final ground-truth response. In fact, the features from both sides are not truly connected in an aspect of general distributions, but only merged into a new vector for each training/testing sample. We suppose that this is not good enough for a model to distill knowledge from both of the two worlds because the language/visual vectors do not contain knowledge from the other modality in the bottom level before they are merged.

In this paper, we discuss another possibility. We want to establish an unsupervised framework of multi-modal encoding, which directly generates an “image feature distribution” from a language

---

[*] Corresponding Author

Proceedings of NAACL-HLT 2019, pages 2588–2598
Minneapolis, Minnesota, June 2 - June 7, 2019.

distribution, or vice versa. For example, when a neural network based model receives a natural language sentence x as input, it encodes x into a sequence of high-dimensional continuous vectors. All these language vectors can be projected into another latent space to have a new distribution p_l. We train the language encoder to let the new distribution p_l be the same as, or very close to, the distribution p_v of all *image features* observed and encoded in the task data. Since we can partly recover a real-world image distribution from the language vectors achieved in this way, these language vectors intrinsically contain both language semantics and real-world image properties. This is a higher-level connection between the two worlds.

In order to train a model to generate samples subject to a certain distribution p_v from an original distribution p_l, Generative Adversarial Networks (GANs) have been proved very effective (Goodfellow et al., 2014; Arjovsky et al., 2017; Miyato et al., 2018). Lample et al. (2018) used adversarial training on the vectors produced by sentence encoders for different languages in unsupervised machine translation. However, different languages in their task are in single modality and share encoder structures, making the same method not directly usable and extendable for multi-modal tasks with largely different prior distributions and complex encoder structures with attention. In our work, we propose Adversarial Multi-modal Feature Encoding (AMFE), a novel GAN-based training schedule with an attention-based sample-selecting method, which can successfully force the multi-modal vectors to have closely related distributions, benefitting the performances of various visual dialog systems.

We test our method on the VisDial (Das et al., 2017a) benchmark (one example is shown in Table 1). A normal sample of VisDial contains an image and 10 turns of question-answering dialog from two people grounded on the image. A series of models have been proposed to solve the task, including memory and attention based models (Das et al., 2017a), reinforcement learning (Das et al., 2017b), knowledge transfer techniques (Lu et al., 2017) and GAN (Wu et al., 2017). Wu et al. (2017) designed a complex attention model and applied GAN in a traditional way to force the generated tokens to mimic real-world language (language vs. language), making their model only trainable through sequence sampling and reinforcement learning. Our work, on the other hand, applies a directly differentiable GAN on continuous vectors as a multi-modal feature encoding method (language vs. image).

	**Caption**: A dog with goggles is in a motorcycle side car A(1): can you tell what kind of dog this is B(1): he looks like beautiful pit bull mix A(2): can you tell if motorcycle is moving or still B(2): it's parked A(3): is dog's tongue lolling out B(3): not really

Table 1: An example from VisDial dataset.

Our contributions include:

- We propose AMFE: a novel Adversarial Multi-modal Feature Encoding framework to benefit visual dialog models. The core idea is to force features from different modalities to have closely related distributions.
- We develop efficient AMFE implementations, including a novel attention-based sample selecting method, for various commonly-used visual dialog models.
- Experimental results show that AMFE brings robust performance gains to different visual dialog models. We achieve state-of-the-arts on most metrics of VisDial v0.5/v0.9 generative tasks.

2 Related Work

2.1 Visual Dialog

Visual Dialog is a cluster of tasks sharing two properties: multi-turn and cross-modality. VisDial (Das et al., 2017a) is a widely-used benchmark with question-answering style dialogs grounded on real-world images. As a special case of dialog generation tasks, VisDial share some of the research concerns with single-modal natural language dialog generation (Dhingra et al., 2016; Serban et al., 2017; Sordoni et al., 2015; Serban et al., 2016; Liu et al., 2016). Natural language dialogs are usually discrete, state-dependent and style-free, thus some reinforcement learning (RL) methods have been proposed (Li et al., 2016). Das et al. (2017b) built an cooperative image guessing task on VisDial: they train both the questioner and the answerer, making them complete a same goal to help the questioner produce a guessing or "imagination" of the unseen image described by the answerer. The distance between the guessing and the target image is used as reward for reinforcement learning. In some extreme settings, such a task definition can even lead to emergence of a new language between robots (Kottur et al., 2017). After per-training, using their reinforcement learning method as an auxiliary loss can also bring performance gain in standard VisDial metrics such as mean rank.

However, generating a reward based on just one target image for a training sample may lead to a kind of overfitting. Language is highly abstract: one dialog can correctly describe a lot of different scenes in real world, so why should we force a dialog to fit one single example among them? Therefore, generating a reward from adversarial training is a more efficient way because it goes beyond individual samples into distributions. There are two previous works (Wu et al., 2017; Lu et al., 2017) that use GAN-like methods to boost the performances of pre-trained VisDial models. (Wu et al., 2017) proposes to use adversarial reinforcement learning. A discriminator is trained to distinguish the tokens of real/generated answers, and the answerer (generator) is trained via RL using a reward related to the score given by discriminator. This method is very effective, but using both RL Monte Carlo and GAN brings high computational cost. Also, a lot of tricks are involved for a good training. Our method, on the other hand, does not need Monte-Carlo sampling to compute immediate reward while generating each of the N words in a sentence ($O(N)$ time cost). (Lu et al., 2017) uses a knowledge-transferring method between generative and discriminative task settings. However, this requires the models on both settings to be pre-trained well enough. Our work is also an adversarial learning based method, but it is more robust, time-efficient and effective.

2.2 GAN for Generative Tasks

GAN (Goodfellow et al., 2014; Arjovsky et al., 2017; Miyato et al., 2018) has raised much attention because of its ability to directly generate samples subject to a target distribution. Many training techniques have been proposed to solve the unstable training problems of GAN (Gulrajani et al., 2017; Kurach et al., 2018). Wasserstein GAN (WGAN) (Arjovsky et al., 2017) is a successful method using critic learning loss and weight clipping operations. We borrow some ideas from WGAN in the adversarial training of our model.

GAN well suits the image generation tasks because image signals are continuous and thus differentiable, enabling the gradient directly flowing back from the discriminator to generator. In language generation tasks, however, how to deal with the discrete sequence of symbols generated by the generator has long been a problem. A widely-used solution is applying RL with rewards generated by the discriminator (Wang et al., 2018; Li et al., 2017). As mentioned above, this is time-costing because RL needs to explore a large action space by sampling multiple action sequence. Besides, how the immediate reward is computed after generating each word is also a difficult problem.

Another solution is to avoid the discrete problem by applying adversarial training on the hidden states of the generator. This requires that there is a known distribution p for the hidden states we want the model to generate. A successful case is reported by (Lample et al., 2018): using adversarial training to restrict the hidden states of source language and target language (both from vanilla LSTMs) into a same latent space can boost the performance of unsupervised machine translation. Our AMFE framework is also an adversarial training on the language hidden states, but we are the first to use this kind of methods to establish connections between different modalities. Our training procedure is also largely different from (Lample et al., 2018) with our modified WGAN-like algorithm and a novel attention-based sample selection method: they are critical for training convergency on multi-modal tasks, with complex attention-based model structures.

3 Model

We first define the task and our framework formally, and then describe how it is implemented and trained on different visual dialog models.

3.1 VisDial Task Definition

In the VisDial task, each sample contains an image I, a caption sentence C and a dialog D with $T = 10$ turns in total. In each turn t, there is a question q_t about the image, and a ground truth answer a_t. The model needs to read the dialog history $H = \{C, (q_1, a_1), ..., (q_{t-1}, a_{t-1})\}$ and image I, to generate an answer as a response to q_t. We rewrite $H_t = (q_t, a_t)$ and $H_0 = C$. Formally, the dialog agent (named A-Bot) outputs an answer

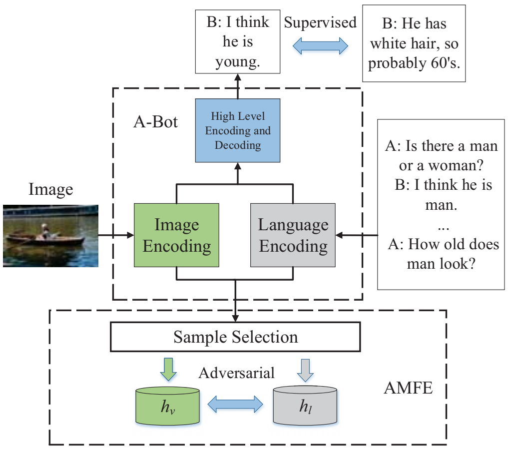


Figure 1: AMFE framework with a generative encoder-decoder model and multi-modal adversarial training.

prediction $\hat{a}_t$:

$$\hat{a}_t = ABot(q_t, I, H_{0\sim t-1}). \tag{1}$$

3.2 AMFE Framework

The goal of our Adversarial Multi-modal Feature Encoding (AMFE) is to restrict the distribution of feature representations from one modality m_1 to be closely related to that from another modality m_2. We take $m_1 = l(anguange)$ and $m_2 = v(isual)$. Specifically, A-Bot encodes language inputs into vectors h_l, and visual inputs into h_v, respectively. We want h_l and h_v to have indistinguishable distributions:

$$h_l, h_v \sim p(h). \tag{2}$$

To achieve this goal, we use a discriminative model (named D-Bot) to classify whether a vector encoded by A-Bot comes from modality l or v. D-Bot is trained with real h_l and h_v samples, while A-Bot is trained to generate language vectors h_l that can confuse D-Bot to classify them as label v. Figure 1 shows our framework.

3.3 A-Bot

We implement our AMFE method on two commonly-used visual dialog models, using them as A-Bot. The two A-Bot models are named Hierarchical Recurrent Encoder (HRE) and History-Conditioned Image Attentive Encoder (HCIAE), respectively. A-Bot learns to predict the right answer in each turn. In this process, it also encodes language and visual inputs into h_l and h_v samples, which we use for AMFE training.

3.3.1 Hierarchical Recurrent Encoder (HRE)

HRE is a hierarchical LSTM (Hochreiter and Schmidhuber, 1997) model used in (Das et al., 2017a,b). In HRE, a pre-trained Convolutional Neural Network (CNN) encodes the image into a single feature vector, which is further mapped into a visual representation I by a trainable Multi-Layer Perceptron (MLP). In each turn t, the question is encoded by a word-level LSTM into a question vector q_t, and the dialog history in the previous step H_{t-1} is encoded by another LSTM into vector f_{t-1}. There is a state-tracker $LSTM^{st}$ on the top level: $LSTM^{st}$ is forwarded one step each turn, integrating all the encoded vectors mentioned above. It reads the encoded history f_{t-1}, image vector I, the question q_t and the previous hidden state s_{t-1} from itself, and produces the new hidden state representation s_t:

$$s_t = LSTM^{st}([q_t, I, f_{t-1}], s_{t-1}), \tag{3}$$

where $[\cdot]$ stands for concatenation.

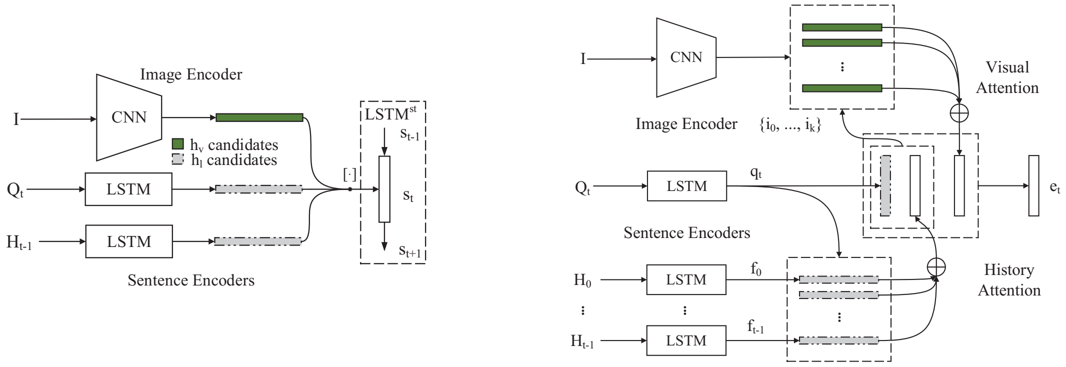
The answer decoder in HRE is an LSTM that takes s_t as initial state, and predicts one word at a time by a softmax probability over the vocabulary, to generate the whole answer sentence. Figure 2(a) shows the encoder structure of HRE. We use image vectors I as h_v samples (dark green) in AMFE, and both the q and f vectors as h_l samples (pink).

3.3.2 History-Conditioned Image Attentive Encoder (HCIAE)

HCIAE model (Lu et al., 2017) contains an textual attention on all history vectors based on the question, and a visual attention based on both the history and the question. In detail, it uses a pre-trained CNN to encode the image into a set of visual feature vectors V. Each vector in V is further passed through a trainable MLP, resulting in a visual feature set $\{i_0, ...i_{K-1}\}$. In each turn t, the question is encoded by an LSTM into vector q_t; the dialog history $\{H_0, H_1, ..., H_{t-1}\}$ is encoded by another word-level LSTM into vectors $\{f_0, ..., f_{t-1}\}$. The attention weight between q_t and each history vector f_j is computed as:

$$\begin{aligned} z_t^j &= \omega_a^T tanh(W_f f_j + W_q q_t), \\ \alpha_t^j &= softmax(z_t^j), \end{aligned} \tag{4}$$

where $\omega_a^T \in \mathbb{R}^{d\times 1}$, $W_f \in \mathbb{R}^{d\times d}$, $W_q \in \mathbb{R}^{d\times d}$ are trainable parameters; d is the length of both



(a) HRE encoder model.

(b) HCIAE encoder model.

Figure 2: A-Bot models we use for AMFE training. The intermediate vectors used as h_v and h_l candidates in AMFE are colored dark green and pink, respectively.

question and history features. A memory vector $\hat{m}_t$ is computed by:

$$\hat{m}_t = \sum_{j=0}^{t-1} \alpha_t^j f_j. \tag{5}$$

The memory vector is further used as a key to compute a similar visual attention over $\{i_0, ...i_{k-1}\}$ to achieve a final image vector $\hat{v}_t$. The final output of the encoder is computed by:

$$e_t = tanh(W_e[q_t, \hat{m}_t, \hat{v}_t]), \tag{6}$$

where $W_e \in \mathbb{R}^{d \times 3d}$ is trainable parameters; $[\cdot]$ stands for concatenation.

The answer decoder is an LSTM like that of HRE, taking e_t as input. Figure 2(b) shows the structure of HCIAE encoder. HCIAE produces more visual vectors for each image than HRE. We take all the q and f vectors as h_l candidates of AMFE, and the spatial visual features $\{i_0, ...i_{k-1}\}$ as candidates for h_v.

3.4 D-Bot

Despite the multiple choices of A-Bot, our D-Bot is always an MLP with two hidden layers of size 512 and ReLU activation. It is used to compute a loss function that forces all the h_l samples to be subject to the same distribution $p(h)$ as the visual vectors h_v. D-Bot takes a vector h in size d as its input, and predicts the probability of h coming from real image distribution and the visual encoder:

$$\hat{p}_v(modality = v|h) = DBot(h). \tag{7}$$

D-Bot is the discriminator from a GAN viewpoint. A-Bot must learn to confuse D-Bot in order to generate language features indistinguishable from image features.

3.5 Training

3.5.1 Loss Functions

To train our model, we use standard supervised training with cross-entropy loss function for pretraining, and add in adversarial training to produce an auxiliary loss to improve feature encoding.

The supervised learning loss is:

$$L_{su} = \frac{1}{N} \sum_{n=1}^{N} -\log(p(w_n^t | w_{<n}^t)), \tag{8}$$

where N is the full length of the decoded sentence.

For adversarial learning, A-Bot is trained to minimize the probability that D-Bot predicts the generated features to be fake samples. Following WGAN (Arjovsky et al., 2017), we do not use logarithm but directly optimize the likelihood itself:

$$L_{adv} = -E_{h_l}[DBot(h_l)]. \tag{9}$$

We sum L_{adv} as an auxiliary loss with a tunable weight λ, making A-Bot minimize:

$$L_G = L_{su} + \lambda L_{adv}. \tag{10}$$

On the other hand, D-Bot maximizes the following objective to distinguish real-world image vectors h_v from the language vectors h_l:

$$L_D = E_{h_v}[DBot(h_v)] - E_{h_l}[DBot(h_l)]. \tag{11}$$

We switch between A-Bot and D-Bot updates for each batch of dialog samples.

3.5.2 Attention-based Sample Selection

We have specified where the h_l and h_v samples come from while using different A-Bots in Section 3.3. Typically, for each batch of samples with batch-size M, in each turn t, there are M question vectors and $M \times t$ history vectors as h_l candidates. For HRE encoder, there are M image vectors as h_v candidates, while the number is $M \times K$ for the HCIAE encoder; K is the number of "pixels" in the final CNN feature-map. Thus, it is impossible to use all the generated samples in AMFE. For a successful training, the selected samples must be efficient, informative and balanced.

While using HCIAE, in order to compute L_{adv}, we use M question vectors and $M * w$ history vectors as h_l samples. The history vectors are selected using textual attention weights α_t^j produced by the temporary model: for each dialog, we pick the top w history vectors with the highest attention weights. We call this Attention-based Sample Selection (AbS). While computing L_D to train D-Bot, we use the same technique on the image, using the top-attended $M * w$ image vectors, together with another M image vectors randomly sampled from the dataset as positive samples h_v. The M question vectors and $M * w$ history vectors are used as a pool of negative samples h_l. In our experiments, $w = 1, 2$ works well.

While using HRE, since the model always "attends" on f_{t-1} by default (Eq. 3), we directly select q_t and f_{t-1} as h_l samples. We use the M image vectors I in this batch, together with another M image vectors randomly sampled from the dataset as the pool of h_v. The full training procedure is specified in Algorithm 1.

Algorithm 1 AMFE Training Procedure.

```
Require: $\alpha$ the learning rate; $c$ the clipping parameter; $M$ the batch size; $w_0$ the initial D-Bot parameters; $\theta_0$ the initial A-Bot parameters; dialog samples.
  Pre-train A-Bot with $L_{su}$ (Eq. 8).
  while $\theta$ has not converged do
      Sample $M$ turns of $(q, H, I, a)$ dialog samples.
      Forward A-Bot and select $h_l$ by attention weights.
      Compute $L_{adv} = -\sum_{k=1}^{M} DBot(h_l^k))$.
      Compute $L_{su}$ with ground-truth answers using (Eq. 8).

      Update $\theta$ to minimize $L_G = L_{su} + \lambda L_{adv}$.
      Switch to D-Bot training.
      Select image vectors $h_v$ by attention weights.
      Re-generate $h_l$ samples using the updated A-Bot.
      Compute $L_D = \sum_{k=1}^{M} DBot(h_v^k) - \sum_{k=1}^{M} DBot(h_l^k)$.
      Update $w$ to maximize $L_D$.
      Clip D-Bot weight $w$ into range $(-c, c)$.
  end while
```

4 Experiments

4.1 The VisDial Dataset

VisDial is a visual dialog dataset based on MS COCO (Lin et al., 2014) images. There are 10 turns of human-posed question-answering dialogs on each image, with the questioner kept not seeing the image during the data collection process. For generative models, a model must give the probability of generating each candidate answer without seeing other candidates, and the rank of the ground-truth answer in the 100 candidates is used to compute different evaluation metrics; for discriminative models, the model can read and encode all the candidate answers and directly assign scores on them. According to the nature of GANs and similarities to real-world application scenarios, we use the generative setting for our model: it is equipped with a sequential decoder instead of a scoring module.

For fair and sufficient comparison, we evaluate our model on both VisDial v0.5 and VisDial v0.9. VisDial v0.5 has 68k COCO images, for a total of 680k QA-pairs. Following (Das et al., 2017a) and (Das et al., 2017b), we use 50,729 images for training, 7,663 for validation and 9,628 for testing. Visdial v0.9 has 123,287 images. There are different splitting of train/valid/test in previous work. We follow (Lu et al., 2017) to use 82k for training, 1k for validation and 40k for testing. [1]

We compare our results to several existing models on the VisDial dataset, including:

- *Answer Prior* (Das et al., 2017a): directly encoding answer candidates with an LSTM and scoring by a linear model that captures the frequency of answers in the training set.
- *NN-QI* (Das et al., 2017a): a k-Nearest Neighborhood method considering only the question and the image. Unlike generative methods, both Answer Prior and NN-QI need to know the answer candidates.
- *LF-QIH-G* (Das et al., 2017a): a Late Fusion encoder that encodes the question, image and history separately. The encoded features are concatenated and linearly transformed to a

---

[1] VisDial has released v1.0 recently, and claims that models trained on v0.9 should also use the new v1.0 test set. Due to lack of baselines in the *generative* task, we follow the original widely-used settings of v0.5 and v0.9.

joint representation. The answer is produced by a generative decoder.

- *HRE* (Das et al., 2017b): the HRE model introduced in Section 3.3.
- *HREA-QIH-G* (Das et al., 2017a): a modified HRE A-Bot with attention to dialog history.
- *MN-QIH-G* (Das et al., 2017a): a Memory Network encoder that stores each piece of dialog history embeddings in an explicit memory. These embeddings can be attended and fused while generating the answer.
- *HCIAE* (Lu et al., 2017): the HCIAE model introduced in Section 3.3.
- *CoAtt* (Wu et al., 2017): this is a previous state-of-the-art model with a more complex co-attention encoder; the decoder is enhanced by adversarial reinforcement learning for better answer generation.

4.2 AMFE for HRE

We first test the efficiency of AMFE on the simpler A-Bot model: HRE. We use VisDial v0.5 as our benchmark for fair comparison with other HRE-based models and auxiliary training methods.

4.2.1 Implementation Details

For Visdial v0.5 dataset, we follow the preprocessing procedure and hyper-parameters described in (Das et al., 2017b). We pass each image through a pre-trained VGG-16 (Simonyan and Zisserman, 2015) CNN, and pick the single f7 vector as input image feature. We limit the maximum lengths of captions, questions and answers to be 40, 20 and 20, respectively; we remove words appearing less than 5 times in the training set, and replace them by a UNK token. We use vector size 300 for word embedding and 512 for all language and visual feature vectors. All LSTMs have two layers.

We pre-train A-Bot with L_{su} for 20 epochs before L_{adv} is added in. The batch-size is set to be 32. After each update of A-Bot, we perform 5 D-Bot updates. We use the 32 encoded image vectors in the batch, together with 32 image vectors randomly sampled from the dataset, to form 64 positive samples; for negative samples, we use the 32 question vectors and 32 history vectors ($t-1$) from the updated A-Bot. We use Adam (Kingma and Ba, 2014) for A-Bot and RMSprop (Tieleman and Hinton, 2012) algorithm for D-Bot to perform gradient descending. The learning rate is set to 1e-3 for pre-training, further decayed to 5e-5; after adversarial training starts, the learning rate is fixed to 5e-5 for both A- and D-Bot. In the weight clipping step of WGAN (Arjovsky et al., 2017), we use a clipping parameter $c = 0.01$.

Model	MRR	R@1	R@5	R@10	Mean
Answer Prior	0.311	19.85	39.14	44.28	31.56
NN-QI	0.385	29.71	46.57	49.86	30.90
LF-QIH-G	0.430	33.27	51.96	58.09	23.04
HREA-QIH-G	0.442	34.47	53.43	59.73	21.83
MN-QIH-G	0.443	**34.62**	53.74	60.18	21.69
HRE-MLE	0.436	33.02	53.41	60.09	21.83
Frozen-Q-Multi	0.437	33.22	53.67	60.48	21.13
HRE-AMFE	**0.445**	**34.62**	**53.95**	**60.76**	**20.98**

Table 2: VisDial v0.5 evaluation results. The five metrics are mean reciprocal rank, recall of the ground-truth answer in the top-1/5/10 ranked candidates (higher is better), and the mean rank of the ground-truth answer (lower is better).

4.2.2 VisDial v0.5 Evaluation Results

On VisDial v0.5, two previous top models are a Memory Network based model (MN-QIH-G) by (Das et al., 2017a) and a multi-loss training on HRE encoder (Frozen-Q-Multi) based on goal-driven reinforcement learning (Das et al., 2017b). We start from the same HRE hyper-parameters and checkpoint as (Das et al., 2017b), but continue with our AMFE instead of reinforcement learning.

Table 2 shows the results on all the five evaluation metrics on VisDial v0.5. Results in the first 4 rows are copied from (Das et al., 2017a). AMFE achieves better performances than the supervised training of A-Bot model (HRE-MLE), especially significant on R@5, R@10 and mean rank, indicating that the adversarial feature encoding results in “generally better” dialogs. It also outperforms the another HRE-like model with history attentions (HREA-QIH-G). While used for multi-loss training, AMFE is significantly better than Frozen-Q-Multi, setting a new state-of-the-art on all metrics. We point out that in Frozen-Q-Multi (Das et al., 2017b), the goal-driven reinforcement leaning reward is computed pair-wise (considering how much can the questioner rebuild the image from the answerer’s words), but the reward computed with a single image is not good enough to evaluate the dialog actions. This is because language is much more abstract than image, and failure to recover an image does not necessarily mean that the dialog is actually bad. Our method could avoid this issue because adversarial training is based on general distributions.

4.3 AMFE for HCIAE

In this section, we test the efficiency of AMFE for the HCIAE model with attention. We use VisDial v0.9 as our benchmark for fair comparison with (Lu et al., 2017).

4.3.1 Implementation Details

For Visdial v0.9 dataset, we follow the preprocessing procedure and HCIAE structure described in (Lu et al., 2017). We pass each image through a pre-trained VGG-19 CNN, resulting in a 512 × 7 × 7 feature-map as visual input. To speed up convergence, we add a Batch Normalization (Ioffe and Szegedy, 2015) after the MLP that further encodes these visual vectors. We limit the maximum lengths of captions, questions and answers to be 24, 16 and 8, respectively. All LSTMs have only one layer.

HCIAE can be trained with either supervised loss (HCIAE-G-MLE) or with multi-loss involving knowledge-transfer (HCIAE-G-DIS). We test AMFE in both settings. For HCIAE-G-MLE, we pre-train HCIAE model with supervised loss for 20 epochs using learning rate 4e-4, and switch to AMFE training with learning rate 5e-5. For HCIAE-G-DIS, we start from the generative model trained with AMFE, together with a pre-trained HCIAE discriminative model. We follow the original knowledge-transfer training schedule, and add our L_{adv} to the original mixed loss function with weight 1. We use batch-size 32 for AMFE training, although the original paper used 128. Other settings are kept the same. For more details please see (Lu et al., 2017).

4.3.2 VisDial v0.9 Evaluation Results

Table 3 shows the results on v0.9. All the HCIAE results are picked from (Lu et al., 2017), and all CoAtt results are picked from (Wu et al., 2017); CoAtt-GAN-TF stands for training a CoAtt model with adversarial reinforcement learning and supervised teacher-forcing; HCIAE-AMFE stands for using AMFE on an HCIAE-G-MLE pre-trained model; HCIAE-GD-AMFE means using AMFE as an additional loss to join the HCIAE-G-DIS multi-loss training.

Model	MRR	R@1	R@5	R@10	Mean
Answer Prior	0.374	23.55	48.52	53.23	26.50
NN-QI	0.427	33.13	50.83	58.69	19.62
LF-QIH-G	0.520	41.83	61.78	67.59	17.07
HREA-QIH-G	0.524	42.28	62.33	68.17	16.79
MN-QIH-G	0.526	42.29	62.85	68.88	17.06
CoAtt-G-MLE	0.541	44.32	63.82	69.75	16.47
CoAtt-GAN-TF	**0.558**	**46.10**	65.69	71.74	14.43
HCIAE-G-MLE	0.539	44.06	63.55	69.24	16.01
HCIAE-G-DIS	0.546	44.35	65.28	71.55	14.23
HCIAE-AMFE	0.547	44.40	65.35	71.69	14.42
HCIAE-GD-AMFE	0.554	45.42	**66.09**	**72.30**	**14.11**

Table 3: VisDial v0.9 evaluation results. The five metrics are mean reciprocal rank, recall of the ground-truth answer in the top-1/5/10 ranked candidates, (higher is better) and the mean rank of the ground-truth answer (lower is better).

On VisDial v0.9, we observe that using AMFE on HCIAE can also boost the performances. Comparing HCIAE-G-MLE and HCIAE-AMFE, we can observe the same advantage over supervised training as on HRE, indicating that our method works for different dataset scales and A-Bot structures; comparing HCIAE-AMFE and HCIAE-G-DIS, AMFE is a competitive method for auxiliary training. Combining AMFE and HCIAE-G-DIS achieves better results than previous state-of-the-art (Wu et al., 2017) on R@5, R@10 and mean rank, and comparable on MRR and R@1. Besides, AMFE trains reasonably faster because we avoid the $O(N)$ time cost for Monte-Carlo sampling while computing temporary rewards (Wu et al., 2017).

We explain the efficiency of AMFE in two aspects. Firstly, AMFE is an adversarial training procedure forcing the language to be encoded into a distribution closely connected to the images. With attention-based sample selection, the most informative samples from both modalities are able to transfer knowledge. Secondly, like Batch Normalization, AMFE contributes to bring better numerical properties to the intermediate tensors in a network, especially on their means and variances, which could potentially benefit model performance.

4.4 Ablation Study

Both the weight of adversarial loss and the attention-based sample selection are critical to good performance. Table 4 shows ablation studies on these factors on HCIAE and VisDial v0.9.

Model	MRR	R@1	R@5	R@10	Mean
HCIAE-G-MLE	0.539	44.06	63.55	69.24	16.01
AMFE-standard	0.547	44.40	65.35	71.69	14.42
w/o AbS	0.537	44.03	63.52	69.33	15.45
$\lambda = 0.1$	0.538	44.10	63.59	69.18	16.07
$\lambda = 10$	0.534	44.01	62.83	68.64	16.56

Table 4: VisDial v0.9 ablation results training with HCIAE-G-MLE and AMFE fine-tuning. AMFE-standard uses $\lambda = 1$ with AbS selection.

#1 Caption: There are several airplanes parked on the tarmac.			
Question	Ground Truth	HRE-MLE	HRE-AMFE
1. are these commercial airplanes	yes they are	yes	yes
2. do they have company names on them	1 says UNK and other looks like it says UNK	yes , but i ca n't read them	yes , but i ca n't read them
3. are all of planes white	they are white 1 has blue and yellow on it and other 2 have red and blue	no they are white	no they are all white
4. are there people visible	yes there are 4 people visible	yes	yes
5. can you tell what sex they are	looks like they are all men	it is male	i think they are both male
6. are any of them wearing sunglasses	i think maybe 1 is it 's hard to tell they are kind of far away	i ca n't tell i can only see their backs	i ca n't tell , they are far away
7. are they dressed for warm or cold weather	looks like warm weather they are wearing their safety gear	they are wearing jackets and shorts	looks warm out
8. are there any trees visible	in far background there are lot of trees	no trees at all	yes there are trees in background
9. can you see any planes taking off or landing	no they all appear to be parked	no i do n't	no i do n't
10. are there any pilots visible inside	no i do not see any	no i do not see any	no there are not
#2 Caption: Some trucks move down a city road , while children play nearby.			
Question	Ground Truth	HRE-MLE	HRE-AMFE
1. how many children are there	i see 2	i can see 3 clearly 3 others are too far away	i see 1
2. girls or boys	bot boys	girls	i think they are boys
3. is there fire	no emergency	not that i can see	not that i can see
4. is anyone on fire truck that you can see	i can barely she driver	no it 's just him	no
5. what are kids playing	they look more like they are walking then playing	i think they are hockey	they are n't shown
6. does it look sunny out	not at all	yes	yes
7. is it black and white picture	no it is not	no	no
8. is it snowing	no it is not	no it is not	no it is not
9. is it raining	no , but it may soon	no it is not	no it is not
10. do you see dark clouds	yes i do	no	no

Table 5: Two examples in VisDial v0.5 dataset for case study.

4.5 Human Evaluation

The above results show that AMFE is especially strong at more "general" metrics such as R@5 and mean rank. To confirm that adversarial training on hidden states can help much to generate responses that are more natural, we randomly select 100 dialog samples from both VisDial v0.5 and v0.9 dataset, and ask two human subjects to vote for the responses generated by two groups of models: HRE-MLE vs. HRE-AMFE on v0.5, and HCIAE-G-MLE vs. HCIAE-AMFE on v0.9, both with beam-size 5. Model names are hidden while voting. We ask the human subjects to consider two metrics separately: (1) the fluency of the generated answer sentences and (2) the correctness of the answers compared to the ground-truths. As shown in Table 6, AMFE wins all the votes with different metric and different models, indicating that AMFE is robust in generating more natural responses.

Models	MLE Wins	AMFE Wins	Tie
HRE-fluency	30	52	18
HCIAE-fluency	34	43	23
HRE-correctness	33	42	25
HCIAE-correctness	29	38	33

Table 6: Human voting result on 100 samples from VisDial v0.5 and v0.9.

5 Case Studies

We randomly sample some dialogs from VisDial v0.5 validation set and illustrate the ground-truth answers and the generated answers with/without AMFE. Two results are shown in Table 5. In the first example, the model trained with AMFE generates a right vs. wrong answer in the 8-th turn, and a grammatically better response in the 5-th turn, compared to supervised pre-training. In the second example, the model trained with AMFE has a generally right understanding of the questions and the image, while the HRE-MLE model is generating response as if it does not see the image. This indicates that encoding language features in the image space leads to better understanding on both modalities.

6 Conclusion

We propose AMFE: an unsupervised multi-modal feature encoding framework and its implementations on different commonly-used visual dialog models. Our core idea is to force features from different modalities to have closely related distributions. Experiments show that AMFE can bring performance gains to both simple and complex models on different scales of VisDial dataset. Future work will possibly be visualizing the visual and language features encoded by AMFE to find more straightforward interpretations, as well as trying our method on more complex structures, discriminative models, and on discriminative tasks such as VQA and visual reasoning.

Acknowledgements

We thank the reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (61602479), the Beijing Brain Science Project (Z181100001518006) and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB32070000).

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *CVPR*, volume 1, page 3.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2016. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge'naturally'in multi-agent dialog. *arXiv preprint arXiv:1706.08502*.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. 2018. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *ICLR*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems*, pages 314–324.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. *arXiv preprint arXiv:1701.08251*.

Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual attention networks for multimodal reasoning and matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 299–307.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. *arXiv preprint arXiv:1804.09160*.

Qi Wu, Peng Wang, Chunhua Shen, Ian Reid, and Anton van den Hengel. 2017. Are you talking to me? reasoned visual dialog generation through adversarial learning. *arXiv preprint arXiv:1711.07613*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. 2018. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems*, (99):1–13.

Strong and Simple Baselines for Multimodal Utterance Embeddings

Paul Pu Liang*, **Yao Chong Lim***,
Yao-Hung Hubert Tsai, Ruslan Salakhutdinov, Louis-Philippe Morency
School of Computer Science, Carnegie Mellon University
`{pliang,yaochonl,yaohungt,rsalakhu,morency}@cs.cmu.edu`

Abstract

Human language is a rich multimodal signal consisting of spoken words, facial expressions, body gestures, and vocal intonations. Learning representations for these spoken utterances is a complex research problem due to the presence of multiple heterogeneous sources of information. Recent advances in multimodal learning have followed the general trend of building more complex models that utilize various attention, memory and recurrent components. In this paper, we propose two simple but strong baselines to learn embeddings of multimodal utterances. The first baseline assumes a conditional factorization of the utterance into unimodal factors. Each unimodal factor is modeled using the simple form of a likelihood function obtained via a linear transformation of the embedding. We show that the optimal embedding can be derived in closed form by taking a weighted average of the unimodal features. In order to capture richer representations, our second baseline extends the first by factorizing into unimodal, bimodal, and trimodal factors, while retaining simplicity and efficiency during learning and inference. From a set of experiments across two tasks, we show strong performance on both supervised and semi-supervised multimodal prediction, as well as significant (10 times) speedups over neural models during inference. Overall, we believe that our strong baseline models offer new benchmarking options for future research in multimodal learning.

1 Introduction

Human language is a rich multimodal signal consisting of spoken words, facial expressions, body gestures, and vocal intonations (Streeck and Knapp, 1992). At the heart of many multimodal modeling tasks lies the challenge of learning rich representations of spoken utterances from multiple modalities (Papo et al., 2014). However, learning representations for these spoken utterances is a complex research problem due to the presence of multiple heterogeneous sources of information (Baltrušaitis et al., 2017). This challenging yet crucial research area has real-world applications in robotics (Montalvo et al., 2017; Noda et al., 2014), dialogue systems (Johnston et al., 2002; Rudnicky, 2005), intelligent tutoring systems (Mao and Li, 2012; Banda and Robinson, 2011; Pham and Wang, 2018), and healthcare diagnosis (Wentzel and van der Geest, 2016; Lisetti et al., 2003; Sonntag, 2017). Recent progress on multimodal representation learning has investigated various neural models that utilize one or more of attention, memory and recurrent components (Yang et al., 2017; Liang et al., 2018). There has also been a general trend of building more complicated models for improved performance.

In this paper, we propose two simple but strong baselines to learn embeddings of multimodal utterances. The first baseline assumes a factorization of the utterance into unimodal factors conditioned on the joint embedding. Each unimodal factor is modeled using the simple form of a likelihood function obtained via a linear transformation of the utterance embedding. We derive a coordinate-ascent style algorithm (Wright, 2015) to learn the optimal multimodal embeddings under our model. We show that, under some assumptions, maximum likelihood estimation for the utterance embedding can be derived in closed form and is equivalent to computing a weighted average of the language, visual and acoustic features. Only a few linear transformation parameters need to be learned. In order to capture bimodal and trimodal representations, our second baseline extends the first one by assuming a factorization into unimodal, bimodal, and trimodal factors (Zadeh et al., 2017). To summarize, our simple baselines 1) consist primarily of linear functions, 2) have few parameters, and 3) can be approximately solved in a closed form solution. As a result, they demonstrate simplicity and

---

* authors contributed equally

Proceedings of NAACL-HLT 2019, pages 2599–2609
Minneapolis, Minnesota, June 2 - June 7, 2019.

efficiency during learning and inference.

We perform a set of experiments across two tasks and datasets spanning multimodal personality traits recognition (Park et al., 2014) and multimodal sentiment analysis (Zadeh et al., 2016). Our proposed baseline models 1) achieve competitive performance on supervised multimodal learning, 2) improve upon classical deep autoencoders for semi-supervised multimodal learning, and 3) are up to 10 times faster during inference. Overall, we believe that our baseline models offer new benchmarks for future multimodal research.

2 Related Work

We provide a review of *sentence embeddings*, *multimodal utterance embeddings*, and *strong baselines*.

2.1 Language-Based Sentence Embeddings

Sentence embeddings are crucial for down-stream tasks such as document classification, opinion analysis, and machine translation. With the advent of deep neural networks, multiple network designs such as Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986), Long-Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997), Temporal Convolutional Networks (Bai et al., 2018), and the Transformer (Vaswani et al., 2017) have been proposed and achieve superior performance. However, more training data is required for larger models (Peters et al., 2018). In light of this challenge, researchers have started to leverage unsupervised training objectives to learn sentence embedding which showed state-of-the-art performance across multiple tasks (Devlin et al., 2018). In our paper, we go beyond unimodal language-based sentence embeddings and consider multimodal spoken utterances where additional information from the nonverbal behaviors is crucial to infer speaker intent.

2.2 Multimodal Utterance Embeddings

Learning multimodal utterance embeddings brings a new level of complexity as it requires modeling both intra-modal and inter-modal interactions (Liang et al., 2018). Previous approaches have explored variants of graphical models and neural networks for multimodal data. RNNs (Elman, 1990; Jain and Medsker, 1999), LSTMs (Hochreiter and Schmidhuber, 1997), and convolutional neural networks (Krizhevsky et al., 2012) have been extended for multimodal settings (Rajagopalan et al., 2016; Lee et al., 2018). Experiments on more advanced networks suggested that encouraging correlation between modalities (Yang et al., 2017), enforcing disentanglement on multimodal representations (Tsai et al., 2018), and using attention to weight modalities (Gulrajani et al., 2017) led to better performing multimodal representations. In our paper, we present a new perspective on learning multimodal utterance embeddings by assuming a conditional factorization over the language, visual and acoustic features. Our simple but strong baseline models offer an alternative approach that is extremely fast and competitive on both supervised and semi-supervised prediction tasks.

2.3 Strong Baseline Models

A recent trend in NLP research has been geared towards building simple but strong baselines (Arora et al., 2017; Shen et al., 2018; Wieting and Kiela, 2019; Denkowski and Neubig, 2017). The effectiveness of these baselines indicate that complicated network components are not always required. For example, Arora et al. (2017) constructed sentence embeddings from weighted combinations of word embeddings which requires no trainable parameters yet generalizes well to down-stream tasks. Shen et al. (2018) proposed parameter-free pooling operations on word embeddings for document classification, text sequence matching, and text tagging. Wieting and Kiela (2019) discovered that random sentence encoders achieve competitive performance as compared to larger models that involve expensive training and tuning. Denkowski and Neubig (2017) emphasized the importance of choosing a basic neural machine translation model and carefully reporting the relative gains achieved by the proposed techniques. Authors in other domains have also highlighted the importance of developing strong baselines (Lakshminarayanan et al., 2017; Sharif Razavian et al., 2014). To the best of our knowledge, our paper is the first to propose and evaluate strong, non-neural baselines for multimodal utterance embeddings.

3 Baselines for Multimodal Learning

3.1 Notation

Suppose we are given video data where each utterance segment is denoted as $\mathbf{s}$. Each segment contains individual words w in a sequence $\mathbf{w}$, visual features v in a sequence $\mathbf{v}$, and acoustic features a in a sequence $\mathbf{a}$. We aim to learn a representation

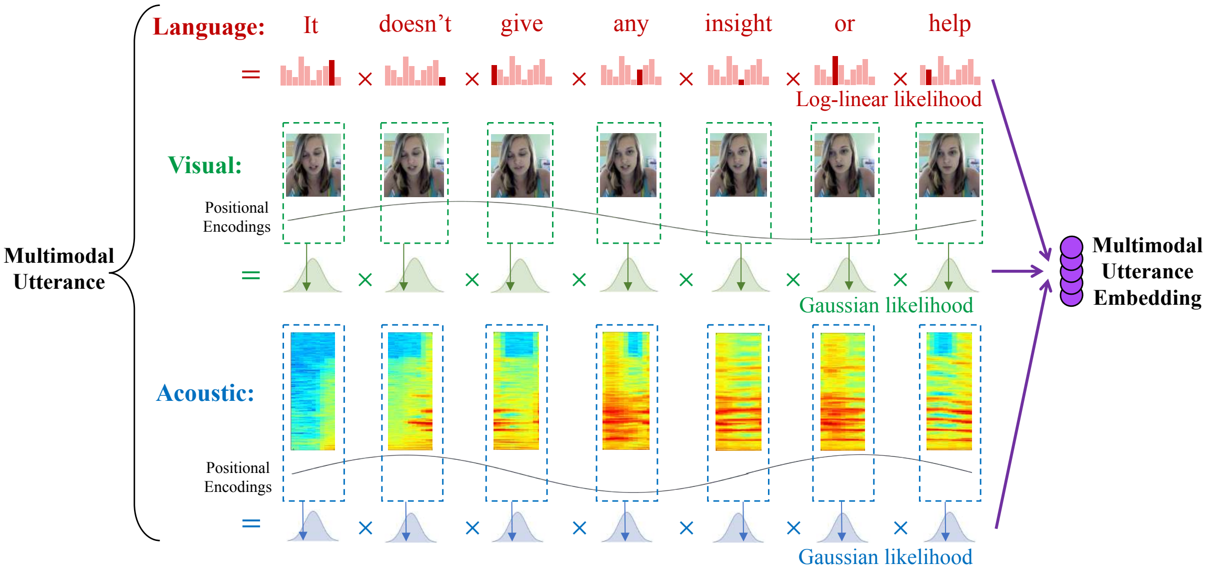


Figure 1: Our baseline model assumes a factorization of the multimodal utterance into unimodal factors conditioned on the joint utterance embedding. Each unimodal factor is modeled using the simple form of a likelihood function obtained via a linear transformation of the utterance embedding. We show that, under some assumptions, maximum likelihood estimation for the utterance embedding can be derived in closed form and is equivalent to taking a weighted average of the language, visual and acoustic features.

$m_{\mathbf{s}}$ for each segment that summarizes information present in the multimodal utterance.

3.2 Background

Our model is related to the work done by Arora et al. (2016) and Arora et al. (2017). In the following, we first provide a brief review of their method. Given a sentence, Arora et al. (2016) aims to learn a sentence embedding c_s. They do so by assuming that the probability of observing a word w_t at time t is given by a log-linear word production model (Mnih and Hinton, 2007) with respect to c_s:

$$\mathbb{P}[w_t|c_s] = \frac{\exp\left(\langle v_{w_t}, c_s\rangle\right)}{Z_{c_s}}, \tag{1}$$

where c_s is the sentence embedding (context), v_{w_t} represents the word vector associated with word w_t and $Z_{c_s} = \sum_{w \in V} \exp\left(\langle v_w, c_s\rangle\right)$ is a normalizing constant over all words in the vocabulary. Given this posterior probability, the desired sentence embedding c_s can be obtained by maximizing Equation (1) with respect to c_s. Under some assumptions on c_s, this maximization yields a closed-form solution which provides an efficient learning algorithm for sentence embeddings.

Arora et al. (2017) further extends this model by introducing a "smoothing term" α to account for the production of frequent stop words or out of context words independent of the discourse vector. Given estimated unigram probabilities $p(w)$, the probability of a word at time t is given by

$$\mathbb{P}[w_t|c_s] = \alpha p(w_t) + (1-\alpha)\frac{\exp\left(\langle v_{w_t}, c_s\rangle\right)}{Z_{c_s}}. \tag{2}$$

Under this model with the additional hyperparameter α, we can still obtain a closed-form solution for the optimal c_s.

3.3 Baseline 1: Factorized Unimodal Model

In this subsection, we outline our method for learning representations of multimodal utterances. An overview of our proposed baseline model is shown in Figure 1. Our method begins by assuming a factorization of the multimodal utterance into unimodal factors conditioned on the joint utterance embedding. Next, each unimodal factor is modeled using the simple form of a likelihood function obtained via a linear transformation of the utterance embedding. Finally, we incorporate positional encodings to represent temporal information in the features. We first present the details of our proposed baseline before deriving a coordinate ascent style optimization algorithm to learn utterance embeddings in our model.

Unimodal Factorization: We use $m_{\mathbf{s}}$ to represent the multimodal utterance embedding. To begin, we simplify the composition of $m_{\mathbf{s}}$ by assuming that the segment $\mathbf{s}$ can be conditionally factorized into words ($\mathbf{w}$), visual features ($\mathbf{v}$), and acoustic features ($\mathbf{a}$). Each factor is also associated with a temperature hyperparameter ($\alpha_{\mathbf{w}}$, $\alpha_{\mathbf{v}}$, $\alpha_{\mathbf{a}}$) that represents the contribution of each factor towards the multimodal utterance. The likelihood of a segment s given the embedding $m_{\mathbf{s}}$ is therefore

$$\begin{aligned}\mathbb{P}[\mathbf{s}|m_{\mathbf{s}}] &= \mathbb{P}[\mathbf{w}|m_{\mathbf{s}}]^{\alpha_{\mathbf{w}}} \left(\mathbb{P}[\mathbf{v}|m_{\mathbf{s}}]\right)^{\alpha_{\mathbf{v}}} \mathbb{P}[\mathbf{a}|m_{\mathbf{s}}]^{\alpha_{\mathbf{a}}} \\ &= \prod_{w\in\mathbf{w}} \mathbb{P}[w|m_{\mathbf{s}}]^{\alpha_{\mathbf{w}}} \prod_{v\in\mathbf{v}} \mathbb{P}[v|m_{\mathbf{s}}]^{\alpha_{\mathbf{v}}} \prod_{a\in\mathbf{a}} \mathbb{P}[a|m_{\mathbf{s}}]^{\alpha_{\mathbf{a}}}.\end{aligned} \tag{3}$$

Choice of Likelihood Functions: As suggested by Arora et al. (2017), given $m_{\mathbf{s}}$, we model the probability of a word w using Equation (2). In order to analytically solve for $m_{\mathbf{s}}$, a lemma is introduced by Arora et al. (2016, 2017) which states that the partition function $Z_{m_{\mathbf{s}}}$ is concentrated around some constant Z (for all $m_{\mathbf{s}}$). This lemma is also known as the "self-normalizing" phenomenon of log-linear models (Andreas and Klein, 2015; Andreas et al., 2015). We use the same assumption and treat $Z_{m_{\mathbf{s}_t}} \approx Z$ for all $m_{\mathbf{s}}$.

Unlike discrete text tokens, the visual features are continuous. We assume that the visual features are generated from an isotropic Gaussian distribution. In section 5.1, we visually analyze the distribution of the features for real world datasets and show that these likelihood modeling assumptions are indeed justified. The Gaussian distribution is parametrized by simple linear transformations $W_v^{\mu}, W_v^{\sigma} \in \mathbb{R}^{|v|\times|m_{\mathbf{s}}|}$ and $b_v^{\mu}, b_v^{\sigma} \in \mathbb{R}^{|v|}$:

$$v|m_{\mathbf{s}} \sim \mathcal{N}(\mu_v, \sigma_v^2), \tag{4}$$

$$\mu_v = W_v^{\mu} m_{\mathbf{s}} + b_v^{\mu}, \tag{5}$$

$$\sigma_v = \operatorname{diag}\left(\exp\left(W_v^{\sigma} m_{\mathbf{s}} + b_v^{\sigma}\right)\right). \tag{6}$$

Similarly, we also assume that the continuous acoustic features are generated from a different isotropic Gaussian distribution parametrized as:

$$a|m_{\mathbf{s}} \sim \mathcal{N}(\mu_a, \sigma_a^2), \tag{7}$$

$$\mu_a = W_a^{\mu} m_{\mathbf{s}} + b_a^{\mu}, \tag{8}$$

$$\sigma_a = \operatorname{diag}\left(\exp\left(W_a^{\sigma} m_{\mathbf{s}} + b_a^{\sigma}\right)\right). \tag{9}$$

Positional Encodings: Finally, we incorporate positional encodings (Vaswani et al., 2017) into the features to represent temporal information. We use d-dimensional positional encodings with entries:

$$PE_{pos,2i} = \sin\left(pos/10000^{2i/d}\right), \tag{10}$$

$$PE_{pos,2i+1} = \cos\left(pos/10000^{2i/d}\right). \tag{11}$$

where pos is the position (time step) and $i \in [1, d]$ indexes the dimension of the positional encodings. We call this resulting model Multimodal Baseline 1 (**MMB1**).

Algorithm 1 Baseline 1

```
1: procedure BASELINE 1
2:     Initialize $m_{\mathbf{s}}, W, b$.
3:     for each iteration do
4:         Fix $W^{(k)}, b^{(k)}$, compute $m_{\mathbf{s}}^{(k)}$ by (13).
5:         Fix $m_{\mathbf{s}}^{(k)}$, compute $\nabla_W \mathcal{L}$ by (21-22).
6:         Fix $m_{\mathbf{s}}^{(k)}$, compute $\nabla_b \mathcal{L}$ by (23-24).
7:         Update $W^{(k+1)}$ from $W^{(k)}$ and $\nabla_W \mathcal{L}$.
8:         Update $b^{(k+1)}$ from $b^{(k)}$ and $\nabla_b \mathcal{L}$.
```

3.4 Optimization for Baseline 1

We define our objective function by the log-likelihood of the observed multimodal utterance s. The maximum likelihood estimator of the utterance embedding $m_{\mathbf{s}}$ and the linear transformation parameters W and b can then be obtained by maximizing this objective

$$\mathcal{L}(m_{\mathbf{s}}, W, b; \mathbf{s}) = \log \mathbb{P}[\mathbf{s}|m_{\mathbf{s}}; W, b], \tag{12}$$

where we use W and b to denote all linear transformation parameters.

Coordinate Ascent Style Algorithm: Since the objective (12) is not jointly convex in $m_{\mathbf{s}}$, W and b, we optimize by alternating between: 1) solving for $m_{\mathbf{s}}$ given the parameters W and b at the current iterate, and 2) given $m_{\mathbf{s}}$, updating W and b using a gradient-based algorithm. This resembles the coordinate ascent optimization algorithm which maximizes the objective according to each coordinate separately (Tseng, 2001; Wright, 2015). Algorithm 1 presents our method for learning utterance embeddings. In the following sections, we describe how to solve for $m_{\mathbf{s}}$ and update W and b.

Solving for $m_{\mathbf{s}}$: We first derive an algorithm to solve for the optimal $m_{\mathbf{s}}$ given the log likelihood objective in (12), and parameters W and b.

Theorem 1. *[Solving for $m_{\mathbf{s}}$] Assume the optimal $m_{\mathbf{s}}$ lies on the unit sphere (i.e. $\|m_{\mathbf{s}}\|_2^2 = 1$), then closed form of $m_{\mathbf{s}}$ in line 4 in Algorithm 1 is*

$$\begin{aligned} m_{\mathbf{s}}^* = &\sum_{w\in s} \psi_w w \\ &+ \sum_{v\in s}\left(W_v^{\mu\top}\tilde{v}^{(1)}\psi_v^{(1)} + W_v^{\sigma\top}\tilde{v}^{(2)}\psi_v^{(2)}\right) \\ &+ \sum_{a\in s}\left(W_a^{\mu\top}\tilde{a}^{(1)}\psi_a^{(1)} + W_a^{\sigma\top}\tilde{a}^{(2)}\psi_a^{(2)}\right). \end{aligned} \tag{13}$$

where the shifted visual and acoustic features are:

$$\tilde{v}^{(1)} = v - b_v^{\mu},\ \tilde{v}^{(2)} = (v - b_v^{\mu}) \otimes (v - b_v^{\mu}), \tag{14}$$

$$\tilde{a}^{(1)} = a - b_a^{\mu},\ \tilde{a}^{(2)} = (a - b_a^{\mu}) \otimes (a - b_a^{\mu}), \tag{15}$$

where $\otimes$ denotes Hadamard (element-wise) product and the weights ψ's are given as follows:

$$\psi_w = \frac{\alpha_{\mathbf{w}}(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)}, \tag{16}$$

$$\psi_v^{(1)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{v}}}{\exp\left(2b_v^{\sigma}\right)}\right), \tag{17}$$

$$\psi_v^{(2)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{v}}}{\exp\left(2b_v^{\sigma}\right)} - \alpha_{\mathbf{v}}\right), \tag{18}$$

$$\psi_a^{(1)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{a}}}{\exp\left(2b_a^{\sigma}\right)}\right), \tag{19}$$

$$\psi_a^{(2)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{a}}}{\exp\left(2b_a^{\sigma}\right)} - \alpha_{\mathbf{a}}\right). \tag{20}$$

Proof. The proof is adapted from Arora et al. (2017) and involves computing the gradients $\nabla_{m_{\mathbf{s}}} \log \mathbb{P}[\cdot|m_{\mathbf{s}}]^{\alpha_{\cdot}}$. We express $\log \mathbb{P}[\cdot|m_{\mathbf{s}}]$ via a Taylor expansion approximation and we observe that $\log \mathbb{P}[\cdot|m_{\mathbf{s}}] \approx c + \langle m_{\mathbf{s}}, g\rangle$ for a constant c and a vector g. Then, we can obtain $m_{\mathbf{s}}^*$ by computing $\arg\max_{m_{\mathbf{s}}^*} \mathcal{L}(m_{\mathbf{s}}, W, b; \mathbf{s})$ which yields a closed-form solution. Please refer to the supplementary material for proof details. ☐

Observe that the optimal embedding $m_{\mathbf{s}}^*$ is a weighted average of the word features w and the (shifted and transformed) visual and acoustic features, $\tilde{v}$ and $\tilde{a}$. Our choice of a Gaussian likelihood for the visual and acoustic features introduces a squared term $(v - b_v^{\mu}) \otimes (v - b_v^{\mu})$ to account for the ℓ_2 distance present in the pdf. The transformation matrix $W^{\top}$ transforms the visual and acoustic features into the multimodal embedding space. Regarding the weights ψ, note that: 1) the weights are proportional to the global temperatures α assigned to that modality, 2) the weights ψ_w are inversely proportional to $p(w)$ (rare words carry more weight), and 3) the weights ψ_v and ψ_a scale each feature dimension inversely by their magnitude.

Updating W and b: To find the optimal linear transformation parameters W and b to maximize the objective in (12), we perform gradient-based optimization on W and b (in Algorithm 1 line 5-8).

Proposition 1. *[Updating W and b] The gradients $\nabla_W \mathcal{L}(m_{\mathbf{s}}, W, b)$ and $\nabla_b \mathcal{L}(m_{\mathbf{s}}, W, b)$, in each dimension, are:*

$$\nabla_{W_{v\,ij}^{\mu}} \mathcal{L}(m_{\mathbf{s}}, W, b) = \alpha_{\mathbf{v}} \operatorname{tr}\left[\left(\sigma_v^{-2}(v - \mu_v)\right)^{\top} m_{\mathbf{s}j}\right], \tag{21}$$

$$\nabla_{W_{v\,ij}^{\sigma}} \mathcal{L}(m_{\mathbf{s}}, W, b) = -\frac{\alpha_{\mathbf{v}}}{2} \operatorname{tr}\left[\left(\sigma_v^{-2} - \sigma_v^{-2}(v - \mu_v)(v - \mu_v)^{\top}\sigma_v^{-2}\right)^{\top} \sigma_{vii} m_{\mathbf{s}j}\right], \tag{22}$$

$$\nabla_{b_{v\,i}^{\mu}} \mathcal{L}(m_{\mathbf{s}}, W, b) = \alpha_{\mathbf{v}} \operatorname{tr}\left[\left(\sigma_v^{-2}(v - \mu_v)\right)^{\top}\right], \tag{23}$$

$$\nabla_{b_{v\,i}^{\sigma}} \mathcal{L}(m_{\mathbf{s}}, W, b) = -\frac{\alpha_{\mathbf{v}}}{2} \operatorname{tr}\left[\left(\sigma_v^{-2} - \sigma_v^{-2}(v - \mu_v)(v - \mu_v)^{\top}\sigma_v^{-2}\right)^{\top} \sigma_{vii}\right]. \tag{24}$$

Proof. The proof involves differentiating the log likelihood of a multivariate Gaussian with respect to μ and σ before applying the chain rule to $\mu = W^{\mu} m_{\mathbf{s}} + b^{\mu}$ and $\sigma = \operatorname{diag}\left(\exp\left(W^{\sigma} m_{\mathbf{s}} + b^{\sigma}\right)\right)$. ☐

3.5 Baseline 2: Incorporating Bimodal and Trimodal Interactions

So far, we have assumed the utterance segment s can be independently factorized into unimodal features. In this subsection, we extend the setting to take account for bimodal and trimodal interactions. We adopt the idea of early-fusion (Srivastava and Salakhutdinov, 2012), which means the bimodal and trimodal interactions are captured by the concatenated features from different modalities. Specifically, we define our factorized model as:

$$\begin{aligned}
&\mathbb{P}[\mathbf{s}|m_{\mathbf{s}}] = \mathbb{P}[\mathbf{w}|m_{\mathbf{s}}]^{\alpha_{\mathbf{w}}} \mathbb{P}[\mathbf{v}|m_{\mathbf{s}}]^{\alpha_{\mathbf{v}}} \mathbb{P}[\mathbf{a}|m_{\mathbf{s}}]^{\alpha_{\mathbf{a}}} \\
&\mathbb{P}[(\mathbf{w} \oplus \mathbf{v})\,|m_{\mathbf{s}}]^{\alpha_{\mathbf{wv}}} \mathbb{P}[(\mathbf{w} \oplus \mathbf{a})\,|m_{\mathbf{s}}]^{\alpha_{\mathbf{wa}}} \\
&\mathbb{P}[(\mathbf{v} \oplus \mathbf{a})\,|m_{\mathbf{s}}]^{\alpha_{\mathbf{va}}} \mathbb{P}[(\mathbf{w} \oplus \mathbf{v} \oplus \mathbf{a})\,|m_{\mathbf{s}}]^{\alpha_{\mathbf{wva}}},
\end{aligned} \tag{25}$$

where $\oplus$ denotes vector concatenation for bimodal and trimodal features. Each of the individual probabilities factorize in the same way as Equation (3) (i.e. $\mathbb{P}[\mathbf{a}|m_{\mathbf{s}}]^{\alpha_{\mathbf{a}}} = \prod_{a \in \mathbf{a}} \mathbb{P}[a|m_{\mathbf{s}}]^{\alpha_{\mathbf{a}}}$). Similar to baseline 1, we assume a log-linear likelihood (2) for $\mathbb{P}[w|m_{\mathbf{s}}]$ and a Gaussian likelihood (4) for all remaining terms. We call this Multimodal Baseline 2 (**MMB2**).

3.6 Optimization for Baseline 2

The optimization algorithm derived in section 3.4 can be easily extended to learn $m_{\mathbf{s}}$, W and b in Baseline 2. We again alternate between the 2 steps of 1) solving for $m_{\mathbf{s}}$ given the parameters W and b at the current iterate, and 2) given $m_{\mathbf{s}}$, updating W and b using a gradient-based algorithm.

Solving for $m_{\mathbf{s}}$: We state a result that derives the closed-form of $m_{\mathbf{s}}$ given W and b:

Corollary 1. *[Solving for $m_{\mathbf{s}}$] Assume that the optimal $m_{\mathbf{s}}$ lies on the unit sphere (i.e. $\|m_{\mathbf{s}}\|_2^2 = 1$). The closed-form (in Algorithm 1 line 4) for $m_{\mathbf{s}}$ is:*

$$
\begin{aligned}
m_{\mathbf{s}}^* &= \sum_{w\in\mathbf{w}} \psi_w w \\
&+ \sum_{v\in\mathbf{v}} \left(W_v^{\mu\top} \tilde{v}^{(1)} \psi_v^{(1)} + W_v^{\sigma\top} \tilde{v}^{(2)} \psi_v^{(2)} \right) \\
&+ \sum_{a\in\mathbf{a}} \left(W_a^{\mu\top} \tilde{a}^{(1)} \psi_a^{(1)} + W_a^{\sigma\top} \tilde{a}^{(2)} \psi_a^{(2)} \right) \\
&+ \sum_{\substack{\mathbf{f}\in\{\mathbf{w}\oplus\mathbf{v},\mathbf{w}\oplus\mathbf{a},\\ \mathbf{v}\oplus\mathbf{a},\mathbf{w}\oplus\mathbf{v}\oplus\mathbf{a}\}}} \sum_{f\in\mathbf{f}} \left(W_f^{\mu\top} \tilde{f}^{(1)} \psi_f^{(1)} + W_f^{\sigma\top} \tilde{f}^{(2)} \psi_f^{(2)} \right)
\end{aligned}
\tag{26}
$$

where the shifted (and squared) visual features are:

$$\tilde{v}^{(1)} = v - b_v^{\mu},\ \tilde{v}^{(2)} = (v - b_v^{\mu}) \otimes (v - b_v^{\mu}), \tag{27}$$

(and analogously for $\tilde{f}^{(1)}, \tilde{f}^{(2)}, f \in \{a, w \oplus v, w \oplus a, v \oplus a, w \oplus v \oplus a\}$). The weights ψ's are:

$$\psi_w = \frac{\alpha_{\mathbf{w}}(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)}, \tag{28}$$

$$\psi_v^{(1)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{v}}}{\exp\left(2 b_v^{\sigma}\right)} \right), \tag{29}$$

$$\psi_v^{(2)} = \operatorname{diag}\left(\frac{\alpha_{\mathbf{v}}}{\exp\left(2 b_v^{\sigma}\right)} - \alpha_{\mathbf{v}} \right). \tag{30}$$

(and analogously for $\psi_f^{(1)}, \psi_f^{(2)}, f \in \{a, w \oplus v, w \oplus a, v \oplus a, w \oplus v \oplus a\}$).

Proof. The proof is a symmetric extension of Theorem 1 to take into account the Gaussian likelihoods for bimodal and trimodal features. □

Updating W and b: The gradient equations for updating W and b are identical to those derived in Proposition 1, Equations (21-24).

3.7 Multimodal Prediction

Given the optimal embeddings $m_{\mathbf{s}}$, we can now train a classifier from $m_{\mathbf{s}}$ to labels y for multimodal prediction. $m_{\mathbf{s}}$ can also be fine-tuned on labeled data (i.e. taking gradient descent steps to update $m_{\mathbf{s}}$ with respect to the task-specific loss functions) to learn task-specific multimodal utterance representations. In our experiments, we use a fully connected neural network for our classifier.

4 Experimental Setup

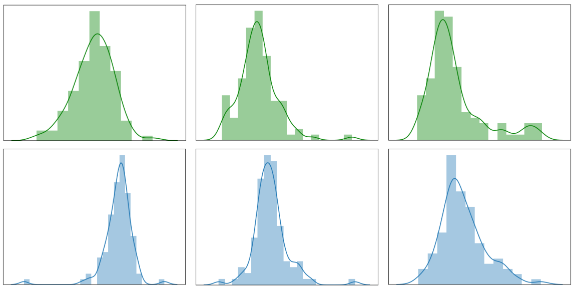


Figure 2: Histogram visualizations of the visual (top) and acoustic (bottom) features in some CMU-MOSI multimodal utterances. Many of the features converge to a Gaussian distribution across the time steps in the utterance, justifying our parametrization for the visual and acoustic likelihood functions.

To evaluate the generalization of our models, we perform experiments on multimodal speaker traits recognition and multimodal sentiment analysis. The code for our experiments is released at `https://github.com/yaochie/multimodal-baselines`, and all datasets for our experiments can be downloaded at `https://github.com/A2Zadeh/CMU-MultimodalSDK`.

4.1 Datasets

All datasets consist of monologue videos where the speaker's intentions are conveyed through the language, visual and acoustic modalities. The multimodal features are described in the next subsection.

Multimodal Speaker Traits Recognition involves recognizing speaker traits based on multimodal utterances. **POM** (Park et al., 2014) contains 903 videos each annotated for speaker traits: confident (con), voice pleasant (voi), dominance (dom), vivid (viv), reserved (res), trusting (tru), relaxed (rel), outgoing (out), thorough (tho), nervous (ner), and humorous (hum). The abbreviations (inside parentheses) are used in the tables.

Multimodal Sentiment Analysis involves analyzing speaker sentiment based on video content. Multimodal sentiment analysis extends conventional language-based sentiment analysis to a multimodal setup where both verbal and non-verbal signals contribute to the expression of sentiment. We use **CMU-MOSI** (Zadeh et al., 2016) which consists of 2199 opinion segments from online videos each annotated with sentiment from strongly negative (-3) to strongly positive ($+3$).

4.2 Multimodal Features and Alignment

GloVe word embeddings (Pennington et al., 2014), Facet (iMotions, 2017) and COVAREP (Degottex et al., 2014) are extracted for the language, visual

Dataset	**POM Personality Trait Recognition, measured in** MAE										
Task	Con	Voi	Dom	Viv	Res	Tru	Rel	Out	Tho	Ner	Hum
Majority	1.483	1.089	1.167	1.158	1.166	0.743	0.753	0.872	0.939	1.181	1.774
SVM	1.071	0.938	0.865	1.043	0.877	0.536	0.594	0.702	0.728	0.714	0.801
DF	1.033	0.899	0.870	0.997	0.884	0.534	0.591	0.698	0.732	0.695	0.768
EF-LSTM$^{(\star)}$	1.035	0.911	0.880	0.981	0.872	0.556	0.594	0.700	0.712	0.706	0.762
MV-LSTM	1.029	0.971	0.944	0.976	0.877	0.523	0.625	0.703	0.792	**0.687**	0.770
BC-LSTM	1.016	0.914	**0.859**	**0.905**	0.888	0.564	0.630	0.708	**0.680**	0.705	0.767
TFN	1.049	0.927	0.864	1.000	0.900	0.572	0.621	0.706	0.743	0.727	0.770
MFN	**0.952**	**0.882**	**0.835**	**0.908**	**0.821**	**0.521**	**0.566**	**0.679**	**0.665**	**0.654**	**0.727**
MMB2	**1.015**	**0.878**	0.885	0.967	**0.857**	**0.522**	**0.578**	**0.685**	0.705	0.692	**0.726**

Dataset	**POM Personality Trait Recognition, measured in** r										
Task	Con	Voi	Dom	Viv	Res	Tru	Rel	Out	Tho	Ner	Hum
Majority	-0.041	-0.104	-0.031	-0.044	0.006	-0.077	-0.024	-0.085	-0.130	0.097	-0.069
SVM	0.063	-0.004	0.141	0.076	0.134	0.168	0.104	0.066	0.134	0.068	0.147
DF	0.240	0.017	0.139	0.173	0.118	0.143	0.019	0.093	0.041	0.136	0.259
EF-LSTM$^{(\star)}$	0.221	0.042	0.151	0.239	0.268	0.069	0.092	0.215	0.252	0.159	0.272
MV-LSTM	0.358	0.131	0.146	0.347	0.323	**0.237**	0.119	0.238	0.284	0.258	0.317
BC-LSTM	**0.359**	0.081	0.234	0.417	0.450	0.109	0.075	0.078	0.363	0.184	0.319
TFN	0.089	0.030	0.020	0.204	-0.051	-0.064	0.114	0.060	0.048	-0.002	0.213
MFN	**0.395**	**0.193**	**0.313**	**0.431**	**0.333**	**0.296**	**0.255**	**0.259**	**0.381**	**0.318**	**0.386**
MMB2	0.350	**0.220**	**0.333**	**0.434**	**0.332**	0.176	**0.224**	**0.318**	**0.394**	**0.296**	**0.366**

Dataset	**CMU-MOSI**	
Task	**Sentiment**	
Metric	A(2)	F1
Majority	50.2	50.1
RF	56.4	56.3
THMM	50.7	45.4
EF-HCRF$^{(\star)}$	65.3	65.4
MV-HCRF$^{(\star)}$	65.6	65.7
SVM-MD	71.6	72.3
C-MKL	72.3	72.0
DF	72.3	72.1
SAL-CNN	73.0	72.6
EF-LSTM$^{(\star)}$	74.3	74.3
MV-LSTM	73.9	74.0
BC-LSTM	73.9	73.9
TFN	74.6	74.5
MFN	**77.4**	**77.3**
MMB1	73.6	73.4
MMB2	**75.2**	**75.1**

Table 1: Results for multimodal personality trait recognition on POM (left) and multimodal sentiment analysis on CMU-MOSI (right). EF-LSTM$^{(\star)}$and HCRF$^{(\star)}$ denote the best result obtained from the LSTM and HCRF variants respectively. The top two results are highlighted in bold. Our proposed baseline model (MMB2), despite its simplicity, often ranks in the top two models and outperforms many large neural models such as C-MKL, DF, SAL-CNN, EF-LSTM, MV-LSTM, BC-LSTM, TFN, and MFN.

and acoustic modalities respectively[1]. Forced alignment is performed using P2FA (Yuan and Liberman, 2008) to obtain the exact utterance times of each word. The video and audio features are aligned by computing the expectation of their features over each word interval (Liang et al., 2018).

4.3 Evaluation Metrics

For classification, we report multiclass classification accuracy A(c) where c denotes the number of classes and F1 score. For regression, we report Mean Absolute Error (MAE) and Pearson's correlation (r). For MAE lower values indicate better performance. For all remaining metrics, higher values indicate better performance.

5 Results and Discussion

5.1 Gaussian Likelihood Assumption

Before proceeding to the experimental results, we perform some sanity checks on our modeling assumptions. We plotted histograms of the visual and acoustic features in CMU-MOSI utterances to visually determine if they resemble a Gaussian distribution. From the plots in Figure 2, we observe that many of the features indeed converge approximately to a Gaussian distribution across the time steps in the utterance, justifying the parametrization for the visual and acoustic likelihood functions in our model.

5.2 Supervised Learning

Our first set of experiments evaluates the performance of our baselines on two multimodal prediction tasks: multimodal sentiment analysis on CMU-MOSI and multimodal speaker traits recognition on POM. On CMU-MOSI (right side of Table 1), our model **MMB2** performs competitively against many neural models including early fusion deep neural networks (Nojavanasghari et al., 2016), several variants of LSTMs (stacked, bidirectional etc.) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997), Multi-view LSTMs (Rajagopalan et al., 2016), and tensor product recurrent models (TFN) (Zadeh et al., 2017). For multimodal personality traits recognition on POM (left side of Table 1), our baseline is able to additionally outperform more complicated memory-based recurrent models such as MFN (Zadeh et al., 2018) on several metrics. We view this as an impressive achievement considering the simplicity of our model and the significantly fewer parameters that our model contains. As we will later show, our model's strong performance comes with the additional benefit of being significantly faster than the existing models.

---

[1] Details on feature extraction are in supplementary.

% Labels	Dataset	**CMU-MOSI**	
	Task	**Sentiment**	
	Metric	A(2)	F1
40%	AE	55.4	54.7
	seq2seq	56.4	49.3
	MMB2	**72.9**	**72.8**
60%	AE	55.2	54.2
	seq2seq	56.3	51.5
	MMB2	**73.6**	**73.5**
80%	AE	55.2	54.8
	seq2seq	55.7	54.7
	MMB2	**74.1**	**74.1**
100%	AE	55.2	53.2
	seq2seq	57.0	54.1
	MMB2	**75.1**	**75.1**

Table 2: Semi-supervised sentiment prediction results on CMU-MOSI. Our model outperforms deep autoencoders (AE) and their recurrent variant (seq2seq), remaining strong despite limited labeled data.

Method	**Average Time (s)**	**Inferences Per Second (IPS)**
DF	0.305	1850
EF-LSTM	0.022	31200
MV-LSTM	0.490	1400
BC-LSTM	0.210	3270
TFN	2.058	333
MFN	0.144	4760
MMB1	**0.00163**	**421000**
MMB2	**0.00219**	**313000**

Table 3: Average time taken for inference on the CMU-MOSI test set and Inferences Per Second (IPS) on a single Nvidia GeForce GTX 1080 Ti GPU, averaged over 5 trials. Our proposed baselines are more than 10 times faster than the closest neural model (EF-LSTM).

Dataset			**CMU-MOSI**	
Task			**Sentiment**	
Model	PE	FT	A(2)	F1
MMB2, language only	✓	✓	72.3	73.7
MMB2	✗	✗	74.1	73.9
MMB2	✗	✓	74.6	74.6
MMB2	✓	✗	74.6	74.6
MMB2	✓	✓	**75.2**	**75.1**

Table 4: Ablation studies on CMU-MOSI test set. Incorporating nonverbal (visual and acoustic) features, positional encodings (PE), and task-specific fine tuning (FT) are important for good prediction performance.

5.3 Semi-supervised Learning

Our next set of experiments evaluates the performance of our proposed baseline models when there is limited labeled data. Intuitively, we expect our model to have a lower sample complexity since training our model involves learning fewer parameters. As a result, we hypothesize that our model will generalize better when there is limited amounts of labeled data as compared to larger neural models with a greater number of parameters.

We test this hypothesis by evaluating the performance of our model on the CMU-MOSI dataset with only 40%, 60%, 80%, and 100% of the training labels. The remainder of the train set now consists of unlabeled data which is also used during training but in a semi-supervised fashion. We use the entire train set (both labeled and unlabeled data) for unsupervised learning of our multimodal embeddings before the embeddings are fine-tuned to predict the label using limited labeled data. A comparison is performed with two models that also learn embeddings from unlabeled multimodal utterances: 1) deep averaging autoencoder (**AE**) (Iyyer et al., 2015; Hinton and Salakhutdinov, 2006) which averages the temporal dimension before using a fully connected autoencoder to learn a latent embedding, and 2) sequence to sequence autoencoder (**seq2seq**) (Sutskever et al., 2014) which captures temporal information using a recurrent neural network encoder and decoder. For each of these models, an autoencoding model is used to learn embeddings on the entire training set (both labeled and unlabeled data) before the embeddings are fine-tuned to predict the label using limited labeled data. The validation and test sets remains unchanged for fair comparison.

Under this semi-supervised setting, we show prediction results on the CMU-MOSI test set in Table 2. Empirically, we find that our model is able to outperform deep autoencoders and their recurrent variant. Our model remains strong and only suffers a drop in performance of about 3% (75.1% → 72.9% binary accuracy) despite having access to only 40% of the labeled training data.

5.4 Inference Timing Comparisons

To demonstrate another strength of our model, we compare the inference times of our model with existing baselines in Table 3. Our model achieves an inference per second (IPS) of more than 10 times the closest neural model (EF-LSTM). We attribute this speedup to our (approximate) closed form solution for m_s as derived in Theorem 1 and Corollary 1, the small size of our model, as well as the fewer number of parameters (linear transformation parameters and classifier parameters) involved.

5.5 Ablation Study

To further motivate our design decisions, we test some ablations of our model: 1) we remove the modeling capabilities of the visual and acoustic modalities, instead modeling only the language modality, 2) we remove the positional encodings, and 3) we remove the fine tuning step. We provide these results in Table 4 and observe that each component is indeed important for our model. Although the text only model performs decently, incorporating visual and acoustic features under our modeling assumption improves performance. Our results also demonstrate the effectiveness of positional encodings and fine tuning without having to incorporate any additional learnable parameters.

6 Conclusion

This paper proposed two simple but strong baselines to learn embeddings of multimodal utterances. The first baseline assumes a factorization of the utterance into unimodal factors conditioned on the joint embedding while the second baseline extends the first by assuming a factorization into unimodal, bimodal, and trimodal factors. Both proposed models retain simplicity and efficiency during both learning and inference. From experiments across multimodal tasks and datasets, we show that our proposed baseline models: 1) display competitive performance on supervised multimodal prediction, 2) outperform classical deep autoencoders for semi-supervised multimodal prediction and 3) attain significant (10 times) speedup during inference. Overall, we believe that our strong baseline models provide new benchmarks for future research in multimodal learning.

Acknowledgements

PPL and LM were partially supported by Samsung and NSF (Award 1750439). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Samsung and NSF, and no official endorsement should be inferred. YHT and RS were supported in part by the NSF IIS1763562, Office of Naval Research N000141812861, and Google focused award. We would also like to acknowledge NVIDIA's GPU support and the anonymous reviewers for their constructive comments on this paper.

References

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249. Association for Computational Linguistics.

Jacob Andreas, Maxim Rabinovich, Michael I. Jordan, and Dan Klein. 2015. On the accuracy of self-normalized log-linear models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1783–1791, Cambridge, MA, USA. MIT Press.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*, ICLR.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406*.

Ntombikayise Banda and Peter Robinson. 2011. Multimodal affect recognition in intelligent tutoring systems. In *Affective Computing and Intelligent Interaction*, pages 200–207, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarepa collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 960–964. IEEE.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

iMotions. 2017. Facial expression analysis.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics.

L. C. Jain and L. R. Medsker. 1999. *Recurrent Neural Networks: Design and Applications*, 1st edition. CRC Press, Inc., Boca Raton, FL, USA.

Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. 2002. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 376–383, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

Chan Woo Lee, Kyu Ye Song, Jihoon Jeong, and Woo Yong Choi. 2018. Convolutional attention networks for multimodal emotion recognition from speech and text data. *arXiv preprint arXiv:1805.06606*.

Paul Pu Liang, Ziyin Liu, Amir Zadeh, and Louis-Philippe Morency. 2018. Multimodal language analysis with recurrent multistage fusion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP.

C. Lisetti, F. Nasoz, C. LeRouge, O. Ozyer, and K. Alvarez. 2003. Developing multimodal intelligent affective interfaces for tele-home health care. *Int. J. Hum.-Comput. Stud.*, 59(1-2):245–255.

Xia Mao and Zheng Li. 2012. Multimodal intelligent tutoring systems. In Elvis Pontes, Anderson Silva, Adilson Guelfi, and Sergio Takeo Kofuji, editors, *E-Learning*, chapter 6. IntechOpen, Rijeka.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA. ACM.

Melissa Montalvo, Eduardo Calle-Ortiz, and Juan Chica. 2017. A multimodal robot based model for the preservation of intangible cultural heritage. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 213–214, New York, NY, USA. ACM.

Kuniaki Noda, Hiroaki Arie, Yuki Suga, and Tetsuya Ogata. 2014. Multimodal integration learning of robot behavior using deep neural networks. *Robotics and Autonomous Systems*, 62(6):721 – 736.

Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. 2016. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 284–288. ACM.

D. Papo, G. Vigliocco, P. Perniss, R.L. Thompson, D. Vinson, and Royal Society (Great Britain). 2014. *Language as a Multimodal Phenomenon: Implications for Language Learning, Processing and Evolution ; Papers of a Theme Issue*. Philosophical Transactions of the Royal Society: Biological Sciences. Royal Society Publishing.

Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. 2014. Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, ICMI '14, pages 50–57, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Phuong Pham and Jingtao Wang. 2018. Predicting learners' emotions in mobile mooc learning via a multimodal intelligent tutor. In *Intelligent Tutoring Systems*, pages 150–159, Cham. Springer International Publishing.

Shyam Sundar Rajagopalan, Louis-Philippe Morency, Tadas Baltrušaitis, and Goecke Roland. 2016. Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*.

Alexander I. Rudnicky. 2005. *Multimodal Dialogue Systems*, pages 3–11. Springer Netherlands, Dordrecht.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450. Association for Computational Linguistics.

Daniel Sonntag. 2017. Interakt - A multimodal multisensory interactive cognitive assessment tool. *CoRR*, abs/1709.01796.

Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230.

Jurgen Streeck and Mark L. Knapp. 1992. The interaction of visual and verbal features in human communication. *Advances in Non-Verbal Communication: Sociocultural, clinical, esthetic and literary perspectives*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yao-Hung Hubert Tsai, Paul Pu Liang, Amir Zadeh, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2018. Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*.

P. Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jobke Wentzel and Thea van der Geest. 2016. Focus on accessibility: Multimodal healthcare technology for all. In *Proceedings of the 2016 ACM Workshop on Multimedia for Personal Health and Health Care*, MMHealth '16, pages 45–48, New York, NY, USA. ACM.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *International Conference on Learning Representations*.

Stephen J. Wright. 2015. Coordinate descent algorithms. *Math. Program.*, 151(1):3–34.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles. 2017. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiahong Yuan and Mark Liberman. 2008. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, 123(5):3878.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Empirical Methods in Natural Language Processing, EMNLP*.

Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018. Memory fusion network for multi-view sequential learning. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems*, 31(6):82–88.

Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout

Hao Tan **Licheng Yu** **Mohit Bansal**
UNC Chapel Hill
`{haotan, licheng, mbansal}@cs.unc.edu`

Abstract

A grand goal in AI is to build a robot that can accurately navigate based on natural language instructions, which requires the agent to perceive the scene, understand and ground language, and act in the real-world environment. One key challenge here is to learn to navigate in new environments that are unseen during training. Most of the existing approaches perform dramatically worse in unseen environments as compared to seen ones. In this paper, we present a generalizable navigational agent. Our agent is trained in two stages. The first stage is training via mixed imitation and reinforcement learning, combining the benefits from both off-policy and on-policy optimization. The second stage is fine-tuning via newly-introduced 'unseen' triplets (environment, path, instruction). To generate these unseen triplets, we propose a simple but effective 'environmental dropout' method to mimic unseen environments, which overcomes the problem of limited seen environment variability. Next, we apply semi-supervised learning (via back-translation) on these dropped-out environments to generate new paths and instructions. Empirically, we show that our agent is substantially better at generalizability when fine-tuned with these triplets, outperforming the state-of-art approaches by a large margin on the private unseen test set of the Room-to-Room task, and achieving the top rank on the leaderboard.[1]

1 Introduction

One of the important goals in AI is to develop a robot/agent that can understand instructions from humans and perform actions in complex environments. In order to do so, such a robot is required to perceive the surrounding scene, understand our spoken language, and act in a real-world house. Recent years have witnessed various types of embodied action based NLP tasks being proposed (Correa et al., 2010; Walters et al., 2007; Hayashi et al., 2007; Zhu et al., 2017b; Das et al., 2018; Anderson et al., 2018b).

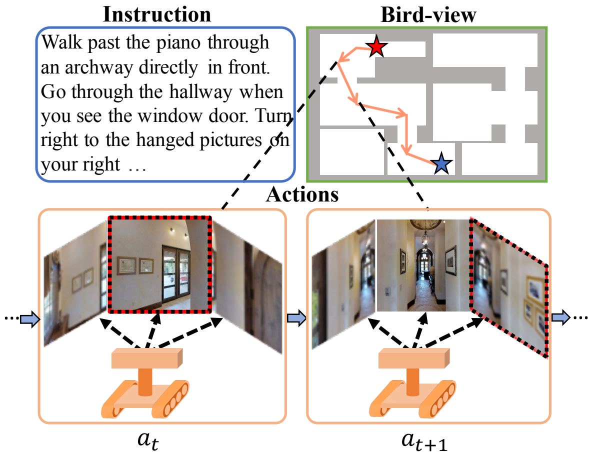


Figure 1: Room-to-Room Task. The agent is given an instruction, then starts its navigation from some staring viewpoint inside the given environment. At time t, the agent selects one view (highlighted as red dotted bounding boxes) from a set of its surrounding panoramic views to step into, as an action a_t.

In this paper, we address the task of instruction-guided navigation, where the agent seeks a route from a start viewpoint to an end viewpoint based on a given natural language instruction in a given environment, as shown in Fig. 1. The navigation simulator we use is the recent Room-to-Room (R2R) simulator (Anderson et al., 2018b), which uses real images from the Matterport3D (Chang et al., 2017) indoor home environments and collects complex navigable human-spoken instructions inside the environments, hence connecting problems in vision, language, and robotics. The instruction in Fig. 1 is *"Walk past the piano through an archway directly in front. Go through the hallway when you see the window door. Turn*

---

[1] Our code, data, and models publicly available at: `https://github.com/airsplay/R2R-EnvDrop`

Proceedings of NAACL-HLT 2019, pages 2610–2621
Minneapolis, Minnesota, June 2 - June 7, 2019.

right to the hanged pictures...". At each position (viewpoint), the agent perceives panoramic views (a set of surrounding images) and selects one of them to step into. In this challenging task, the agent is required to understand each piece of the instruction and localize key views (*"piano"*, *"hallway"*, *"door"*, etc.) for making actions at each time step. Another crucial challenge is to generalize the agent's navigation understanding capability to unseen test room environments, considering that the R2R task has substantially different unseen (test) rooms as compared to seen (trained) ones. Such generalization ability is important for developing a practical navigational robot that can operate in the wild.

Recent works (Fried et al., 2018; Wang et al., 2019, 2018a; Ma et al., 2019) have shown promising progress on this R2R task, based on speaker-follower, reinforcement learning, imitation learning, cross-modal, and look-ahead models. However, the primary issue in this task is that most models perform substantially worse in unseen environments than in seen ones, due to the lack of generalizability. Hence, in our paper, we focus on improving the agent's generalizability in unseen environments. For this, we propose a two-stage training approach. The first stage is training the agent via mixed imitation learning (IL) and reinforcement learning (RL) which combines off-policy and on-policy optimization; this significantly outperforms using IL or RL alone.

The second, more important stage is semi-supervised learning with generalization-focused 'environmental dropout'. Here, the model is fine-tuned using additional training data generated via back-translation. This is usually done based on a neural speaker model (Fried et al., 2018) that synthesizes new instructions for additional routes in the *existing* environments. However, we found that the bottleneck for this semi-supervised learning method is the limited variability of given (seen) environments. Therefore, to overcome this, we propose to generate novel and diverse environments via a simple but effective 'environmental dropout' method based on view- and viewpoint-consistent masking of the visual features. Next, the new navigational routes are collected from these new environments, and lastly the new instructions are generated by a neural speaker on these routes, and these triplets are employed to fine-tune the model training.

Overall, our fine-tuned model based on back-translation with environmental dropout substantially outperforms the previous state-of-the-art models, and achieves the most recent rank-1 on the Vision and Language Navigation (VLN) R2R challenge leaderboard's private test data, outperforming all other entries in success rate under all evaluation setups (single run, beam search, and pre-exploration).[2] We also present detailed ablation and analysis studies to explain the effectiveness of our generalization method.

2 Related Work

Embodied Vision-and-Language Recent years are witnessing a resurgence of active vision. For example, Levine et al. (2016) used an end-to-end learned model to predict robotic actions from raw pixel data, Gupta et al. (2017) learned to navigate via mapping and planning, Sadeghi and Levine (2017) trained an agent to fly in simulation and show its performance in the real world, and Gandhi et al. (2017) trained a self-supervised agent to fly from examples of drones crashing. Meanwhile, in the intersection of active perception and language understanding, several tasks have been proposed, including instruction-based navigation (Chaplot et al., 2018; Anderson et al., 2018b), target-driven navigation (Zhu et al., 2017b; Gupta et al., 2017), embodied question answering (Das et al., 2018), interactive question answering (Gordon et al., 2018), and task planning (Zhu et al., 2017a). While these tasks are driven by different goals, they all require agents that can perceive their surroundings, understand the goal (either presented visually or in language instructions), and act in a virtual environment.

Instruction-based Navigation For instruction-based navigation task, an agent is required to navigate from start viewpoint to end viewpoint according to some given instruction in an environment. This task has been studied by many works (Tellex et al., 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Andreas and Klein, 2015; Mei et al., 2016; Misra et al., 2017) in recent years. Among them, (Anderson et al., 2018b) differs from the others as it introduced a photo-realistic dataset – Room-to-Room (R2R), where all images are real ones taken by Matterport3D (Chang et al., 2017) and the instructions are also natural. In R2R

---

[2] `https://evalai.cloudcv.org/web/challenges/challenge-page/97/overview`

environments, the agent's ability to perceive real-world images and understanding natural language becomes even more crucial. To solve this challenging task, a lot of works (Fried et al., 2018; Wang et al., 2018a, 2019; Ma et al., 2019) have been proposed and shown some potential. The most relevant work to us is Fried et al. (2018), which proposed to use a speaker to synthesize new instructions and implement pragmatic reasoning. However, we observe there is some performance gap between seen and unseen environments. In this paper, we focus on improving the agent's generalizability in unseen environment.

Back-translation Back translation (Sennrich et al., 2016), a popular semi-supervised learning method, has been well studied in neural machine translation (Hoang et al., 2018; Wang et al., 2018b; Edunov et al., 2018; Prabhumoye et al., 2018). Given paired data of source and target sentences, the model first learns two translators – a forward translator from source to target and a backward translator from target to source. Next, it generates more source sentences using the back translator on an external target-language corpus. The generated pairs are then incorporated into the training data for fine-tuning the forward translator, which proves to improve the translation performance. Recently, this approach (also known as data augmentation) was applied to the task of instruction-based navigation (Fried et al., 2018), where the source and target sentences are replaced with instructions and routes.

3 Method

3.1 Problem Setup

Navigation in the Room-to-Room task (Anderson et al., 2018b) requires an agent to find a route $\mathbf{R}$ (a sequence of viewpoints) from the start viewpoint $\mathbf{S}$ to the target viewpoint $\mathbf{T}$ according to the given instruction $\mathbf{I}$. The agent is put in a photo-realistic environment $\mathbf{E}$. At each time step t, the agent's observation consists of a panoramic view and navigable viewpoints. The panoramic view o_t is discretized into 36 single views $\{o_{t,i}\}_{i=1}^{36}$. Each single view $o_{t,i}$ is an RGB image $v_{t,i}$ accompanied with its orientation $(\theta_{t,i}, \phi_{t,i})$, where $\theta_{t,i}$ and $\phi_{t,i}$ are the angles of heading and elevation, respectively. The navigable viewpoints $\{l_{t,k}\}_{k=1}^{N_t}$ are the N_t reachable and visible locations from the current viewpoint. Each navigable viewpoint $l_{t,k}$ is represented by the orientation $(\hat{\theta}_{t,k}, \hat{\phi}_{t,k})$ from current viewpoint to the next viewpoints. The agent needs to select the moving action a_t from the list of navigable viewpoints $\{l_{t,k}\}$ according to the given instruction $\mathbf{I}$, history/current panoramic views $\{o_\tau\}_{\tau=1}^{t}$, and history actions $\{a_\tau\}_{\tau=1}^{t-1}$. Following Fried et al. (2018), we concatenate the ResNet (He et al., 2016) feature of the RGB image and the orientation as the view feature $f_{t,i}$:

$$\begin{aligned} f_{t,i} = [\, & \mathrm{ResNet}(v_{t,i}); \\ & (\cos\theta_{t,i}, \sin\theta_{t,i}, \cos\phi_{t,i}, \sin\phi_{t,i})\,] \end{aligned} \tag{1}$$

The navigable viewpoint feature $g_{t,k}$ is extracted in the same way.

3.2 Base Agent Model

For our base instruction-to-navigation translation agent, we implement an encoder-decoder model similar to Fried et al. (2018). The encoder is a bidirectional LSTM-RNN with an embedding layer:

$$\hat{w}_j = \text{embedding}(w_j) \tag{2}$$

$$u_1, u_2, \cdots, u_{\mathrm{L}} = \text{Bi-LSTM}(\hat{w}_1, \cdots, \hat{w}_{\mathrm{L}}) \tag{3}$$

where u_j is the j-th word in the instruction with a length of L. The decoder of the agent is an attentive LSTM-RNN. At each decoding step t, the agent first attends to the view features $\{f_{t,i}\}$ computing the attentive visual feature $\tilde{f}_t$:

$$\alpha_{t,i} = \mathrm{softmax}_i(f_{t,i}^{\intercal} W_{\mathrm{F}}\, \tilde{h}_{t-1}) \tag{4}$$

$$\tilde{f}_t = \sum\nolimits_i \alpha_{t,i} f_{t,i} \tag{5}$$

The input of the decoder is the concatenation of the attentive visual feature $\tilde{f}_t$ and the embedding of the previous action $\tilde{a}_{t-1}$. The hidden output h_t of the LSTM is combined with the attentive instruction feature $\tilde{u}_t$ to form the instruction-aware hidden output $\tilde{h}_t$. The probability of moving to the k-th navigable viewpoint $p_t(a_{t,k})$ is calculated as softmax of the alignment between the navigable viewpoint feature $g_{t,k}$ and the instruction-aware hidden output $\tilde{h}_t$.

$$h_t = \mathrm{LSTM}\left([\tilde{f}_t; \tilde{a}_{t-1}], \tilde{h}_{t-1}\right) \tag{6}$$

$$\beta_{t,j} = \mathrm{softmax}_j\left(u_j^{\intercal}\, W_{\mathrm{U}}\, h_t\right) \tag{7}$$

$$\tilde{u}_t = \sum\nolimits_j \beta_{t,j}\, u_j \tag{8}$$

$$\tilde{h}_t = \tanh\left(W\, [\tilde{u}_t;\, h_t]\right) \tag{9}$$

$$p_t(a_{t,k}) = \mathrm{softmax}_k\left(g_{t,k}^{\intercal}\, W_{\mathrm{G}}\, \tilde{h}_t\right) \tag{10}$$

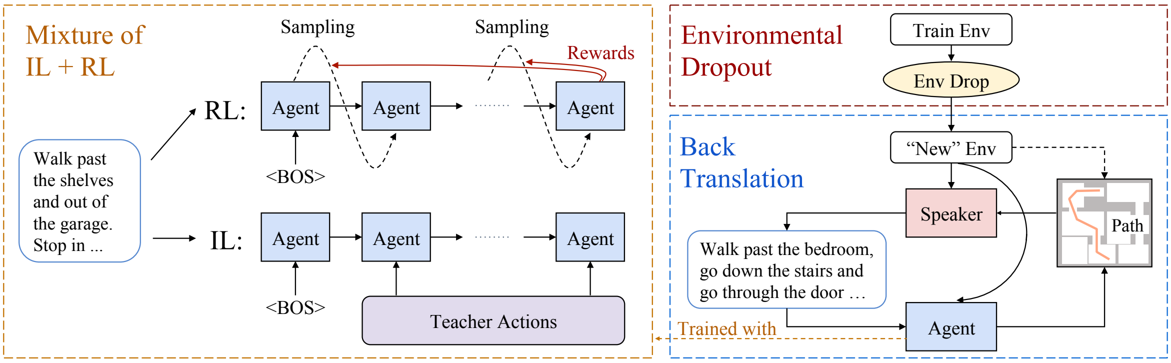


Figure 2: Left: IL+RL supervised learning (stage 1). Right: Semi-supervised learning with back translation and environmental dropout (stage 2).

Different from Fried et al. (2018), we take the instruction-aware hidden vector $\tilde{h}_{t-1}$ as the hidden input of the decoder instead of h_{t-1}. Thus, the information about which parts of the instruction have been attended to is accessible to the agent.

3.3 Supervised Learning: Mixture of Imitation+Reinforcement Learning

We discuss our IL+RL supervised learning method in this section.[3]

Imitation Learning (IL) In IL, an agent learns to imitate the behavior of a teacher. The teacher demonstrates a teacher action a_t^* at each time step t. In the task of navigation, a teacher action a_t^* selects the next navigable viewpoint which is on the shortest route from the current viewpoint to the target $\mathbf{T}$. The off-policy[4] agent learns from this weak supervision by minimizing the negative log probability of the teacher's action a_t^*. The loss of IL is as follows:

$$\mathcal{L}^{\text{IL}} = \sum_t \mathcal{L}_t^{\text{IL}} = \sum_t \text{-}\log p_t(a_t^*) \tag{11}$$

For exploration, we follow the IL method of Behavioral Cloning (Bojarski et al., 2016), where the agent moves to the viewpoint following the teacher's action a_t^* at time step t.

Reinforcement Learning (RL) Although the route induced by the teacher's actions in IL is the shortest, this selected route is not guaranteed to satisfy the instruction. Thus, the agent using IL is biased towards the teacher's actions instead of finding the correct route indicated by the instruction. To overcome these misleading actions, the on-policy reinforcement learning method Advantage Actor-Critic (Mnih et al., 2016) is applied, where the agent takes a sampled action from the distribution $\{p_t(a_{t,k})\}$ and learns from rewards. If the agent stops within $3m$ around the target viewpoint $\mathbf{T}$, a positive reward $+3$ is assigned at the final step. Otherwise, a negative reward -3 is assigned. We also apply reward shaping (Wu et al., 2018): the direct reward at each non-stop step t is the change of the distance to the target viewpoint.

IL+RL Mixture To take the advantage of both off-policy and on-policy learners, we use a method to mix IL and RL. The IL and RL agents share weights, take actions separately, and navigate two independent routes (see Fig. 2). The mixed loss is the weighted sum of $\mathcal{L}^{\text{IL}}$ and $\mathcal{L}^{\text{RL}}$:

$$\mathcal{L}^{\text{MIX}} = \mathcal{L}^{\text{RL}} + \lambda_{\text{IL}}\mathcal{L}^{\text{IL}} \tag{12}$$

IL can be viewed as a language model on action sequences, which regularizes the RL training.[5]

3.4 Semi-Supervised Learning: Back Translation with Environmental Dropout

3.4.1 Back Translation

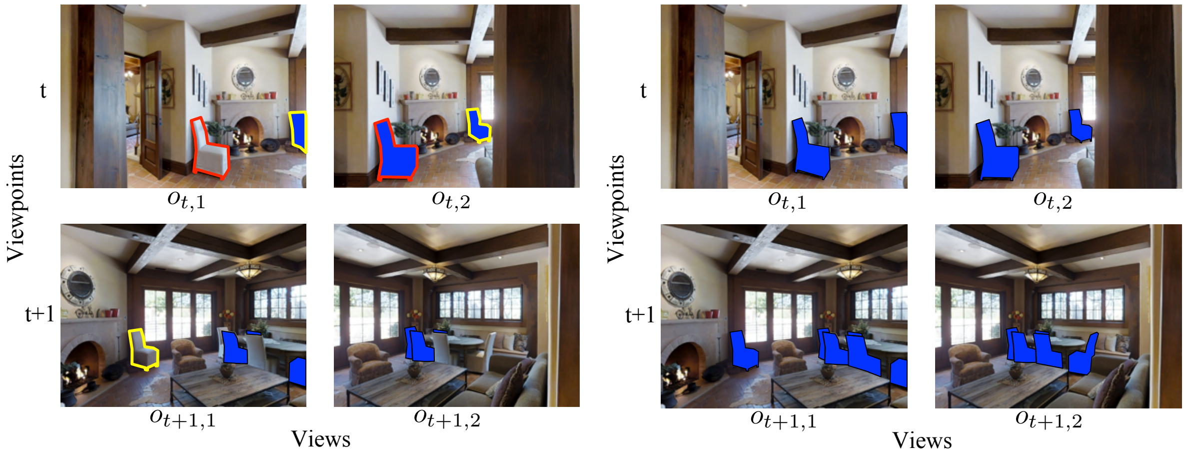
Suppose the primary task is to learn the mapping of $\mathbf{X} \rightarrow \mathbf{Y}$ with paired data $\{(\mathbf{X}, \mathbf{Y})\}$ and unpaired data $\{\mathbf{Y}'\}$. In this case, the back translation method first trains a forward model $P_{\mathbf{X}\rightarrow\mathbf{Y}}$ and a backward model $P_{\mathbf{Y}\rightarrow\mathbf{X}}$, using paired data $\{(\mathbf{X}, \mathbf{Y})\}$. Next, it generates additional datum $\mathbf{X}'$

---

[3] As opposed to semi-supervised methods in Sec. 3.4, in this section we view both imitation learning and reinforcement learning as supervised learning.

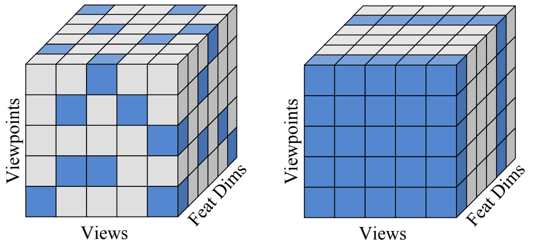
[4] According to Poole and Mackworth (2010), an off-policy learner learns the agent policy independently of the agent's navigational actions. An on-policy learner learns the policy from the agent's behavior including the exploration steps.

[5] This approach is similar to the method ML+RL in Paulus et al. (2018) for summarization. Recently, Wang et al. (2018a) combines purely supervised learning and RL training however, they use a different algorithm named MIXER (Ranzato et al., 2015), which computes cross entropy (XE) losses for the first k actions and RL losses for the remaining.



(a) Feature dropout (b) Environmental dropout

Figure 3: Comparison of the two dropout methods (based on an illustration on an RGB image).



(a) Feature dropout (b) Environmental dropout

Figure 4: Comparison of the two dropout methods (based on image features).

from the unpaired $\mathbf{Y}'$ using the backward model $P_{\mathbf{Y}\to\mathbf{X}}$. Finally, $(\mathbf{X}', \mathbf{Y}')$ are paired to further fine-tune the forward model $P_{\mathbf{X}\to\mathbf{Y}}$ as additional training data (also known as 'data augmentation').

Back translation was introduced to the task of navigation in Fried et al. (2018). The forward model is a navigational agent $P_{\mathbf{E},\mathbf{I}\to\mathbf{R}}$ (Sec. 3.2), which navigates inside an environment $\mathbf{E}$, trying to find the correct route $\mathbf{R}$ according to the given instruction $\mathbf{I}$. The backward model is a *speaker* $P_{\mathbf{E},\mathbf{R}\to\mathbf{I}}$, which generates an instruction $\mathbf{I}$ from a given route $\mathbf{R}$ inside an environment $\mathbf{E}$. Our speaker model (details in Sec. 3.4.3) is an enhanced version of Fried et al. (2018), where we use a stacked bidirectional LSTM-RNN encoder with attention flow.

For back translation, the Room-to-Room dataset labels around 7% routes $\{\mathbf{R}\}$ in the training environments[6], so the rest of the routes $\{\mathbf{R}'\}$ are unlabeled. Hence, we generate additional instructions $\mathbf{I}'$ using $P_{\mathbf{E},\mathbf{R}\to\mathbf{I}}(\mathbf{E}, \mathbf{R}')$, so to obtain the new triplets $(\mathbf{E}, \mathbf{R}', \mathbf{I}')$. The agent is then fine-tuned with this new data using the IL+RL method described in Sec. 3.3. However, note that the environment $\mathbf{E}$ in the new triplet $(\mathbf{E}, \mathbf{R}', \mathbf{I}')$ for semi-supervised learning is still selected from the *seen* training environments. We demonstrate that the limited amount of environments $\{\mathbf{E}\}$ is actually the bottleneck of the agent performance in Sec. 7.1 and Sec. 7.2. Thus, we introduce our environmental dropout method to mimic the "new" environment $\mathbf{E}'$, as described next in Sec. 3.4.2.

3.4.2 Environmental Dropout

Failure of Feature Dropout Different from dropout on neurons to regularize neural networks, we drop raw feature dimensions (see Fig. 4a) to mimic the removal of random objects from an RGB image (see Fig. 3a). This traditional feature dropout (with dropout rate p) is implemented as an element-wise multiplication of the feature f and the dropout mask ξ^f. Each element ξ^f_e in the dropout mask ξ^f is a sample of a random variable which obeys an independent and identical Bernoulli distribution multiplied by $1/(1-p)$. And for different features, the distributions of dropout masks are independent as well.

$$\mathrm{dropout}_p(f) = f \odot \xi^f \tag{13}$$

$$\xi^f_e \sim \frac{1}{1-p}\mathrm{Ber}(1-p) \tag{14}$$

Because of this independence among dropout masks, the traditional feature dropout fails in augmenting the existing environments because the 'removal' is inconsistent in different views at the same viewpoint, and in different viewpoints.

---

[6] The number of all possible routes (shortest paths) in the 60 existing training environments is 190K. Of these, the Room-to-Room dataset labeled around 14K routes with one navigable instruction for each, so the amount of labeled routes is around 7% of 190K.

To illustrate this idea, we take the four RGB views in Fig. 3a as an example, where the chairs are randomly dropped from the views. The removal of the left chair (marked with a red polygon) from view $o_{t,2}$ is inconsistent because it also appears in view $o_{t,1}$. Thus, the speaker could still refer to it and the agent is aware of the existence of the chair. Moreover, another chair (marked with a yellow polygon) is completely removed from viewpoint observation o_t, but the views in next viewpoint o_{t+1} provides conflicting information which would confuse the speaker and the agent. Therefore, in order to make generated environments consistent, we propose our environmental dropout method, described next.

Environmental Dropout We create a new environment $\mathbf{E}'$ by applying environmental dropout on an existing environment $\mathbf{E}$.

$$\mathbf{E}' = \mathrm{envdrop}_p(\mathbf{E}) \tag{15}$$

The view feature $f'_{t,i}$ observed from the new environment $\mathbf{E}'$ is calculated as an element-wise multiplication of the original feature $f_{t,i}$ and the environmental dropout mask $\xi^{\mathbf{E}}$ (see Fig. 4b):

$$f'_{t,i} = f_{t,i} \odot \xi^{\mathbf{E}} \tag{16}$$

$$\xi^{\mathbf{E}}_e \sim \frac{1}{1-p}\mathrm{Ber}(1-p) \tag{17}$$

To maintain the spatial structure of viewpoints, only the image feature $\mathrm{ResNet}(v_{t,i})$ is dropped while the orientation feature $(\cos(\theta_{t,i}), \sin(\theta_{t,i}), \cos(\phi_{t,i}), \sin(\phi_{t,i}))$ is fixed. As illustrated in Fig. 3b, the idea behind environmental dropout is to mimic new environments by removing one specific class of object (e.g., the chair). We demonstrate our idea by running environmental dropout on the ground-truth semantic views in Sec. 7.3, which is proved to be far more effective than traditional feature dropout. In practice, we perform the environmental dropout on image's visual feature where certain structures/parts are dropped instead of object instances, but the effect is similar.

We apply the environmental dropout to the back translation model as mentioned in Sec. 3.4.1. Note the environmental dropout method still preserves the connectivity of the viewpoints, thus we use the same way (Fried et al., 2018) to collect extra unlabeled routes $\{\mathbf{R}'\}$. We take *speaker* to generate an additional instruction $\mathbf{I}'$=$P_{\mathbf{E},\mathbf{R}\rightarrow\mathbf{I}}\,(\mathbf{E}', \mathbf{R}')$ in the new environment $\mathbf{E}'$. At last, we use IL+RL (in Sec. 3.3) to fine-tune the model with this new triplet $(\mathbf{E}', \mathbf{R}', \mathbf{I}')$.

3.4.3 Improvements on Speaker

Our speaker model is an enhanced version of the encoder-decoder model of Fried et al. (2018), with improvements on the visual encoder: we stack two bi-directional LSTM encoders: a route encoder and a context encoder. The route encoder takes features of ground truth actions $\{a^*_t\}_{t=1}^{T}$ from the route as inputs. Each hidden state r_t then attends to surrounding views $\{f_{t,i}\}_{i=1}^{36}$ at each viewpoint. The context encoder then reads the attended features and outputs final visual encoder representations:

$$r_1, ..., r_{\mathrm{T}} = \text{Bi-LSTM}^{\mathrm{RTE}}(g_{1,a^*_1}, ..., g_{\mathrm{T},a^*_{\mathrm{T}}}) \tag{18}$$

$$\gamma_{t,i} = \mathrm{softmax}_i(f_{t,i}^{\intercal} W_{\mathrm{R}}\, r_t) \tag{19}$$

$$\hat{f}_t = \sum\nolimits_i \gamma_{t,i} f_{t,i} \tag{20}$$

$$c_1, ..., c_{\mathrm{T}} = \text{Bi-LSTM}^{\mathrm{CTX}}(\hat{f}_1, ..., \hat{f}_{\mathrm{T}}) \tag{21}$$

The decoder is a regular attentive LSTM-RNN, which is discussed in Sec. 3.2. Empirically, our enhanced speaker model improves the BLEU-4 score by around 3 points.

4 Experimental Setup

Dataset and Simulator We evaluate our agent on the Matterport3D simulator (Anderson et al., 2018b). Navigation instructions in the dataset are collected via Amazon Mechanical Turk by showing them the routes in the Matterport3D environment (Chang et al., 2017). The dataset is split into training set (61 environments, 14,025 instructions), seen validation set (61 environments, 1,020 instructions), unseen validation set (11 environments, 2,349 instructions), and unseen test set (18 environments, 4,173 instructions). The unseen sets only involve the environments outside the training set.

Evaluation Metrics For evaluating our model, Success Rate (SR) is the primary metric. The execution route by the agent is considered a success when the navigation error is less than 3 meters. Besides success rate, we use three other metrics[7] : Navigation Length (NL), Navigation Error (NE), and Success rate weighted by Path Length (SPL) (Anderson et al., 2018a). Navigation Error (NE) is

---

[7] The Oracle Success Rate (OSR) is not included because it's highly correlated with the Navigation Length.

Models	Test Unseen (Leader-Board)								
	Single Run			Beam Search			Pre-Explore		
	NL	SR(%)	*SPL*	NL	*SR(%)*	SPL	NL	SR(%)	*SPL*
Random (Anderson et al., 2018b)	9.89	13.2	0.12	-	-	-	-	-	-
Seq-to-Seq (Anderson et al., 2018b)	8.13	20.4	0.18	-	-	-	-	-	-
Look Before You Leap (Wang et al., 2018a)	9.15	25.3	0.23	-	-	-	-	-	-
Speaker-Follower (Fried et al., 2018)	14.8	35.0	0.28	1257	53.5	0.01	-	-	-
Self-Monitoring (Ma et al., 2019)	18.0	48.0	0.35	373	61.0	**0.02**	-	-	-
Reinforced Cross-Modal (Wang et al., 2019)	12.0	43.1	0.38	358	63.0	**0.02**	9.48	60.5	0.59
Ours	11.7	**51.5**	**0.47**	687	**68.9**	0.01	9.79	**63.9**	**0.61**

Table 1: Leaderboard results under different experimental setups. NL, SR, and SPL are Navigation Length, Success Rate and Success rate weighted by Path Length. The primary metric for each setup is in italics. The best results are in bold font and the second best results are underlined.

the distance between target viewpoint $\mathbf{T}$ and agent stopping position.

Implementation Details Similar to the traditional dropout method, the environmental dropout mask is computed and applied at each training iteration. Thus, the amount of unlabeled semi-supervised data used is not higher in our dropout method. We also find that sharing the environmental dropout mask in different environments inside a batch will stabilize the training. To avoid over-fitting, the model is early-stopped according to the success rate on the unseen validation set. More training details in appendices.

5 Results

In this section, we compare our agent model with the models in previous works on the Vision and Language Navigation (VLN) leaderboard. The models on the leaderboard are evaluated on a private unseen test set which contains 18 new environments. We created three columns in Table 1 for different experimental setups: single run, beam search, and unseen environments pre-exploration. For the result, our model outperforms all other models in all experimental setups.

Single Run Among all three experimental setups, single run is the most general and highly correlated to the agent performance. Thus, it is considered as the primary experimental setup. In this setup, the agent navigates the environment once and is not allowed[8] to: (1) run multiple trials, (2) explore nor map the test environments before starting. Our result is 3.5% and 9% higher than the second-best in Success Rate and SPL, resp.

Beam Search In the beam search experimental setup, an agent navigates the environment, collects multiple routes, re-ranks them, and selects the route with the highest score as the prediction. Besides showing an upper bound, beam search is usable when the environment is explored and saved in the agent's memory but the agent does not have enough computational capacity to fine-tune its navigational model. We use the same beam-search algorithm, state factored Dijkstra algorithm, to navigate the unseen test environment. Success Rate of our model is 5.9% higher than the second best. SPL metric generally fails in evaluating beam-search models because of the long Navigation Length (range of SPL is 0.01-0.02).

Pre-Exploration The agent pre-explores the test environment before navigating and updates its agent model with the extra information. When executing the instruction in the environment, the experimental setup is still "single run". The "pre-exploration" agent mimics the domestic robots (e.g., robot vacuum) which only needs to navigate the seen environment most of the time. For submitting to the leaderboard, we simply train our agent via back translation with environmental dropout on test unseen environments (see Sec.7.2). Our result is 3.4% higher than Wang et al. (2019) in Success Rate and 2.0% higher in SPL. [9]

6 Ablation Studies

Supervised Learning We first show the effectiveness of our IL+RL method by comparing it with the baselines (Table 2). We implement Behavioural Cloning[10] and Advantage Actor-Critic

---

[8] According to the Vision and Language Navigation (VLN) challenge submission guidelines

[9] To fairly compare with Wang et al. (2019), we exclude the exploration route in calculating Navigation Length.

[10] The Behavioral Cloning (IL) baseline is the same as the panoramic view baseline in Fried et al. (2018) except for two differences: (1) The agent takes the teacher action instead of the sampled action from the distribution (see "imitation learning" of Sec. 3.3), (2) The hidden input of the LSTM is the instruction-aware hidden from the previous step (see Sec. 3.2). We improve our baseline result with these modifi-

Models	Val Seen				Val Unseen			
	NL(m)	NE(m)	SR(%)	SPL	NL(m)	NE(m)	SR(%)	SPL
SUPERVISED LEARNING								
Behavioral Cloning (IL)	10.3	5.39	48.4	0.46	9.15	6.25	43.6	0.40
Advantage Actor-Critic (RL)	73.8	7.11	22.0	0.03	73.8	7.32	24.0	0.03
IL + RL	10.1	4.71	55.3	0.53	9.37	5.49	46.5	0.43
SEMI-SUPERVISED LEARNING								
Back Translation	10.3	4.19	58.1	0.55	10.5	5.43	48.2	0.44
+ Feat Drop	10.3	4.13	58.4	0.56	9.62	5.43	48.4	0.45
+ Env Drop (No Tying)	10.3	4.32	57.3	0.55	9.51	5.27	49.0	0.46
+ Env Drop (Tying)	11.0	3.99	62.1	0.59	10.7	5.22	52.2	0.48
FULL MODEL								
Single Run	11.0	3.99	62.1	0.59	10.7	5.22	52.2	0.48
Beam Search	703	2.52	75.7	0.01	663	3.08	69.0	0.01
Pre-Explore	9.92	4.84	54.7	0.52	9.57	3.78	64.5	0.61

Table 2: For the ablation study, we show the results of our different methods on validation sets. Our full model (single run) gets 8.6% improvement in validation unseen success rate above our baseline. And both the supervised learning (IL+RL) and semi-supervised learning methods (back translation + env drop) have substantial contributions to our final result.

as our imitation learning (IL) and reinforcement learning (RL) baselines, respectively. The mixture of IL+RL (see Sec. 3.3) outperforms the IL-only model and RL-only model by 2.9% and 22.5%, which means that our IL+RL could overcome the misleading teacher actions in IL and significantly stabilize the training of RL.

Semi-Supervised Learning We then fine-tune our best supervised model (i.e., IL+RL) with back translation. Besides providing a warm-up, IL+RL is also used to learn the new generated data triplets in back translation. As shown in Table 2, back translation with environmental dropout improves the best supervised model by 5.7%, where the improvement is 3 times more than the back translation without new environments. We then show the results of the alternatives to environmental dropout. The performance with feature dropout is almost the same to the original back translation, which is 3.8% lower than the environmental dropout. We also prove that the improvement from the environmental dropout method does not only come from generating *diverse* instructions introduced by dropout in the speaker, but also comes from using the same dropout mask in the follower agent too. To show this, we use two independent (different) environmental dropout masks for the speaker and the follower (i.e., no tying of the dropout mask), and the result drops a lot as compared to when we tie the speaker and follower dropout masks.

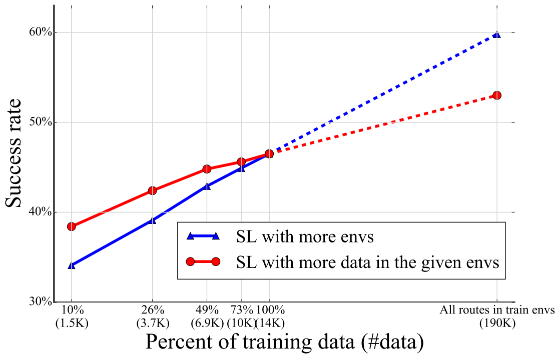


Figure 5: Success rates of agents trained with different amounts of data. X-axis in log-scale. The blue line represents the growth of results by gradually adding new environments to the supervised training method. The red line is trained with the same amounts of data as the blue line, but the data is randomly selected from all 60 training environments. The dashed lines are predicted.

Full Model Finally, we show the performance of our best agent under different experimental setups. The "single run" result is copied from the best semi-supervised model for comparison. The state-factored Dijkstra algorithm (Fried et al., 2018) is used for the beam search result. The method for pre-exploration is described in Sec. 7.2, where the agent applies back translation with environmental dropout on the validation unseen environment.

7 Analysis

In this section, we present analysis experiments that first exposed the limited environments bottleneck to us, and hence inspired us to develop our environmental dropout method to break this bottleneck.

---

cations.

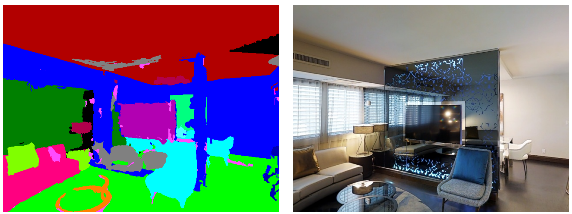

(a) Semantic View (b) RGB view

Figure 6: Comparison of semantic and raw RGB views.

7.1 More Environments vs. More Data

In order to show that more environments are crucial for better performance of agents, in Fig. 5, we present the result of Supervised Learning (SL) with different amounts of data selected by two different data-selection methods. The first method gradually uses more #environments (see the blue line "SL with more envs") while the second method selects data from the whole training data with all 60 training environments (see the red line "SL with more data"). Note that the amounts of data in the two setups are the same for each plot point As shown in Fig. 5, the "more envs" selection method shows *higher growth rate* in success rate than the "more data" method. We also predict the success rates (in dashed line) with the prediction method in Sun et al. (2017). The predicted result is much higher when training with more environments. The predicted result (the right end of the red line) also shows that the upper bound of Success Rate is around 52% if all the 190K routes in the training environments is labeled by human (instead of being generated by *speaker* via back translation), which indicates the need for "new" environments.

7.2 Back Translation on Unseen Environments

In this subsection, we show that back translation could significantly improve the performance when it uses new data triplets from testing environments — the unseen validation environments where the agent is evaluated in. Back translation (w.o. Env Drop) on these unseen environments achieves a success rate of 61.9%, while the back translation on the training environments only achieves 46.5%. The large margin between the two results indicates the need of "new" environments in back translation. Moreover, our environmental dropout on testing environments could further improve the result to 64.5%, which means that the amount of environments in back translation is far from enough.

7.3 Semantic Views

To demonstrate our intuition of the success of environmental dropout (in Sec. 3.4.2), we replace the image feature $\mathrm{ResNet}(v_{t,i})$ with the semantic view feature. The semantic views (as shown in Fig. 6) are rendered from the Matterport3D dataset (Chang et al., 2017), where different colors indicate different types of objects. Thus, dropout on the semantic view feature would remove the object from the view. With the help of this additional information (i.e., the semantic view), the success rate of IL+RL is 49.5% on the unseen validation set. Back translation (without dropout) slightly improves the result to 50.5%. The result with feature dropout is 50.2% while the environmental dropout could boost the result to 52.0%, which supports our claim in Sec. 3.4.2.

8 Conclusion

We presented a navigational agent which better generalizes to unseen environments. The agent is supervised with a mixture of imitation learning and reinforcement learning. Next, it is fine-tuned with semi-supervised learning, with speaker-generated instructions. Here, we showed that the limited variety of environments is the bottleneck of back translation and we overcome it via 'environmental dropout' to generate new unseen environments. We evaluate our model on the Room-to-Room dataset and achieve rank-1 in the Vision and Language Navigation (VLN) challenge leaderboard under all experimental setups.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by ARO-YIP Award #W911NF-18-1-0336, ONR Grant #N00014-18-1-2871, and faculty awards from Google, Facebook, Adobe, Baidu, and Salesforce. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agency.

References

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh

Mottaghi, Manolis Savva, et al. 2018a. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2018. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2.

Andrew Correa, Matthew R Walter, Luke Fletcher, Jim Glass, Seth Teller, and Randall Davis. 2010. Multimodal interaction with an autonomous forklift. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 243–250. IEEE Press.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *CVPR)*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3318–3329.

Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. 2017. Learning to fly by crashing. *IROS*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *CVPR*.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. 2017. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Kotaro Hayashi, Daisuke Sakamoto, Takayuki Kanda, Masahiro Shiomi, Satoshi Koizumi, Hiroshi Ishiguro, Tsukasa Ogasawara, and Norihiro Hagita. 2007. Humanoid robots as a passive-social medium-a field experiment at a train station. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 137–144. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *JMLR*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019. Self-monitoring navigation agent via auxiliary progress estimation.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, volume 1, page 2.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

David L Poole and Alan K Mackworth. 2010. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proc. ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. In *ICLR*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Fereshteh Sadeghi and Sergey Levine. 2017. CAD2RL: Real single-image flight without a single real image. *RSS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 86–96.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 843–852. IEEE.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2.

Michael L Walters, Kerstin Dautenhahn, Sarah N Woods, and Kheng Lee Koay. 2007. Robotic etiquette: results from user studies involving a fetch and carry task. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 317–324. ACM.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018a. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, pages 38–55.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018b. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. 2018. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*.

Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017a. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 483–492.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017b. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE.

A Appendices

A.1 Implementation Details

We use ResNet-152 (He et al., 2016) pretrained on the ImageNet (Russakovsky et al., 2015) to extract the 2048-dimensional image feature. The agent model is first trained with supervised learning via the mixture of imitation and reinforcement learning. The model is then fine-tuned by back translation with environmental dropout. To stabilize the optimization of back translation, we calculate supervised loss for half of the batch and semi-supervised loss for the other half. We find that sharing the environmental dropout mask in different environments inside the same batch will stabilize the training.

The word embedding is trained from scratch with size 256 and the dimension of the action embedding is 64. The size of the LSTM units is set to 512 (256 for the bidirectional LSTM). In RL training, the discounted factor γ is 0.9. We use the reward shaping (Wu et al., 2018): the direct reward r_t at time step t is the change of the distance $d_{t-1} - d_t$, supposing d_t is the distance to the target position at time step t. The maximum decoding action length is set to 35. For optimizing the loss, we use RMSprop (Hinton et al., 2012) (as suggested in Mnih et al. (2016)) with a fixed learning rate $1e-4$ and the batch size is 64. We applying dropout rate 0.4 to the environmental dropout and 0.5 to the dropout layers which regularize the network. The global gradient norm is clipped by 40. We tuned the hyper-parameters based on the Success Rate of the unseen validation set.

When working with the semantic view, the *key* labels (e.g., *wall*, *floor*, *ceil*) are not dropped, because they are the basic structure of the environment. Empirically, no improvement will be achieved when the *key* labels are dropped as well.

Towards Content Transfer through Grounded Text Generation

Shrimai Prabhumoye
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15219
`sprabhum@andrew.cmu.edu`

Chris Quirk, Michel Galley
Microsoft Research
One Microsoft Way
Redmond, WA 98052
`{chrisq,mgalley}@microsoft.com`

Abstract

Recent work in neural generation has attracted significant interest in controlling the *form* of text, such as style, persona, and politeness. However, there has been less work on controlling neural text generation for content. This paper introduces the notion of Content Transfer for long-form text generation, where the task is to generate a next sentence in a document that both fits its context and is grounded in a content-rich external textual source such as a news story. Our experiments on Wikipedia data show significant improvements against competitive baselines. As another contribution of this paper, we release a benchmark dataset of 640k Wikipedia referenced sentences paired with the source articles to encourage exploration of this new task.

1 Introduction

Recent work in neural natural language generation (NLG) has witnessed a growing interest in controlling text for various form-related and linguistic properties, such as style (Ficler and Goldberg, 2017), affect (Ghosh et al., 2017), politeness (Sennrich et al., 2016), persona (Li et al., 2016b) voice (Yamagishi et al., 2016), grammatical correctness (Ji et al., 2017), and length (Kikuchi et al., 2016). This trend offers the promise of empowering existing authoring tools such as Grammarly, Google Smart Compose, and Microsoft Word with the ability to control a much greater variety of textual properties, which are currently mostly limited to grammar, spelling, word choice, and wordiness. What has been relatively less explored in neural NLG research is the ability to control the generation of a current sentence not only in its *form*, but also its *content*.[1] Consider for example Fig. 1, which illustrates a situation where an author edits a document (here a Wikipedia article), and the goal is to generate or suggest a next sentence (shown in orange) to the author. This type of unconstrained, long-form text generation task (Mostafazadeh et al., 2016; Fan et al., 2018) is of course extremely difficult. Free-form generation can easily go astray due to two opposing factors. On one hand, ensuring that the generated output is of relatively good quality often comes at the cost of making it bland and devoid of factual content (Li et al., 2016a). On the other hand, existing techniques can help steer neural models away

Monkey selfie copyright dispute

From Wikipedia, the free encyclopedia

On 4 July 2011 several publications picked up the story and quoted Slater as describing the photographs as self-portraits. Slater said reports that a monkey ran off with his camera and "began taking self-portraits" were incorrect and that the portrait was shot when his camera had been on a tripod, with the primates playing around with a remote cable release as he fended off other monkeys.[14]

One of the monkey selfies at issue in the dispute

Ape-rture priority photographer plays down monkey reports

amateurphotographer.co.uk
July 5, 2011

A photographer who says he witnessed monkeys taking pictures of themselves, tells Amateur Photographer (AP) that much of the media coverage has been exaggerated.

Speaking to AP, David explained that his camera had been mounted on a tripod when the primates began playing around with a remote 'cable release' as he was trying to fend off other monkeys.

Figure 1: Example of content transfer: Given existing curated text (yellow) and a document with additional relevant information (green), the task is to update the curated text (orange) to reflect the most salient updates.

---

[1] Historically, NLG has focused on generation from structured content such as a database or semantic representation, but this paper is interested in generation from free-form text.

Proceedings of NAACL-HLT 2019, pages 2622–2632
Minneapolis, Minnesota, June 2 - June 7, 2019.

from blandness in order to produce more contentful outputs (using temperature sampling (Fan et al., 2018), GAN (Goodfellow et al., 2014), etc.), but often at the cost of "hallucinating" (Wiseman et al., 2017) words or concepts that are totally irrelevant. Neither situation provides a compelling experience to the user.

What is clearly missing from the aforementioned authoring scenario is the notion of *grounding*: there is often a profusion of online resources that bear at least some relevance to any given document currently being written. Much of the general-purpose world knowledge is available in the form of encyclopedias (e.g., Wikipedia), books (e.g., Project Gutenberg, Google Books), and news articles. While the generation of good quality texts without any conditioning on "external" sources (Fan et al., 2018) might be an interesting research endeavor on its own, we argue that grounding can make the generation task much easier, e.g., as shown in Fig. 1 where a passage of a news article (green) can be reformulated considering the current context of the document (yellow) in order to produce a natural next sentence (orange). In light of this desideratum, this paper addresses the problem of grounded text generation, where the goal is to infuse the content or knowledge from an external source (e.g., a news article as in Fig. 1) in order to generate a follow-up sentence of an existing document. We see this as a form of *Content Transfer*, as other characteristics of the external source—such as style and linguistic form—are not controlled.

In addition to formulating this new task, our work makes the following contributions: We provide a large dataset of 640k instances that contain parallel data of a source document (news articles), a context, and sentence to be produced. The latter two are extracted from Wikipedia, which is an attractive dataset for grounded generation as many of the statements in Wikipedia cite external sources (i.e., grounded in an external article). Finally, we also provide simple yet efficient models that condition both on the external article and the context of the current document. We compare our models against extractive and abstractive baselines, including summarization methods that simply try to condense the external article without considering the context of the document. Our experiments show that our models which incorporate the context gain 7.0 ROUGE-L F1 points —in other words, treating our task as a summarization problem is not enough. Our human evaluations also show that models that are aware of the context generate relevant and fluent sentences that are coherent to the context.

2 Task

This research is concerned with the general problem of grounded authorship assistance, i.e., the task of suggesting text to insert or append in an existing document draft, in such a way that all the added *content* reflects information from external sources, such as news articles and books. This type of grounded generation task could take many forms, so we decided to formalize the task as follows, while still keeping the task both challenging and practically interesting. Given an external **document** (green in Fig. 1), and some existing **curated text** (yellow), the task is to generate a single **update sentence** (orange). This update sentence should be both relevant to the context and reflective of the information contained in the document.

This task bears some similarity with automatic summarization (Nenkova and McKeown, 2011), as a naïve approach to the above problem is to append a one-sentence summary of the document to the curated text. While indeed related, the two tasks differ in two key points. First, the one-sentence summary must be contextually appropriate given the previous context of the curated text. Second, summarization is mostly concerned with finding *salient* information, but—in the case of our task—information relevant to the context might actually only be auxiliary within the external document. Section 6 (Related Work) further contrasts our task with summarization.

Formally we define our task as follows: given an existing curated text $\boldsymbol{s}$ and a document $\boldsymbol{d}$ describing novel information relevant to that text, the system must produce a revised text $\boldsymbol{s}'$ that incorporates the most salient information from $\boldsymbol{d}$. We restrict our focus to the cases where the revised text $\boldsymbol{s}'$ can be obtained by appending the new information from $\boldsymbol{d}$ to the original curated text $\boldsymbol{s}$.[2] In particular, we assume we can transform the old curated text $\boldsymbol{s}$ into the new text $\boldsymbol{s}'$ by appending one additional update sentence $\boldsymbol{x}$ to $\boldsymbol{s}$.

---

[2] In general, updated information from $\boldsymbol{d}$ might demand substantial changes to $\boldsymbol{s}$: perhaps core assumptions of $\boldsymbol{s}$ were contradicted, necessitating many removed and rewritten sentences. We postpone this complex setting to future work.

3 Models

This paper operates in a conventional supervised learning setting. For training data, we rely on a large dataset of existing curated text $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$, corresponding documents with novel information $\boldsymbol{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n\}$, and the update sentences $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Our task is to generate the update sentence $\boldsymbol{x}_i$ that could be appended to the curated text $\boldsymbol{s}_i$ in order to incorporate the additional information from document $\boldsymbol{d}_i$. The goal would be to identify new information (in particular, $\boldsymbol{d}_i \setminus \boldsymbol{s}_i$) that is most salient to the topic or focus of the text, then generate a single sentence that represents this information.

3.1 Generative models

A natural though difficult means of generating this additional update sentence $\boldsymbol{x}$ is to use a generative model conditioned on the information in the curated text $\boldsymbol{s}$ and the new document $\boldsymbol{d}$. Recent methods inspired by successful neural machine translation systems have produced impressive results in abstractive summarization (Nallapati et al., 2016). Hence, our first step is to use the sequence-to-sequence encoder-decoder model (Bahdanau et al., 2015) with attention (Luong et al., 2015) for our task. This kind of model assumes that the output sentence can be generated word-by-word. Each output word $\boldsymbol{x}_i^t$ generated is conditioned on all prior words $\boldsymbol{x}_i^{<t}$ and an encoded representation of the context $\boldsymbol{z}$:

$$\prod_t p(\hat{\boldsymbol{x}}_i^t | \hat{\boldsymbol{x}}_i^{<t}, \boldsymbol{z}) \tag{1}$$

Context Agnostic Generative (CAG) Model: One simple baseline is to train a sequence-to-sequence model for the document $\boldsymbol{d}$ alone that does not directly incorporate information from the curated text $\boldsymbol{s}$. Here, the algorithm is trained to generate the most likely update sentence $\hat{\boldsymbol{x}} = \arg\max p(\boldsymbol{x}|\boldsymbol{d})$. In this setting, we consider the reference document $\boldsymbol{d}_i$ as the source and the update sentence to be generated $\boldsymbol{x}_i$ as the target.

$$\boldsymbol{z} = \text{Encoder}(\boldsymbol{d}_i, \boldsymbol{\theta}) \tag{2}$$

The encoder and decoder do not directly see the information from the curated text $\boldsymbol{s}$, but the update $\boldsymbol{x}$ inherently carries some information about it. The parameters of the model are learned from updates that were authored given the knowledge of the curated text. Hence, the model may capture some generalizations about the kinds of information and locations in $\boldsymbol{d}$ that are most likely to contribute novel information to $\boldsymbol{s}$.

Context Only Generative (COG) Model: This algorithm is trained to generate the most likely update sentence $\hat{\boldsymbol{x}} = \arg\max p(\boldsymbol{x}|\boldsymbol{s})$. This model is similar to CAG except that we consider the curated $\boldsymbol{s}_i$ as the source. In this setting, there is no grounding of the content to be generated.

Context Informed Generative (CIG) Model: An obvious next step is to incorporate information from the curated text $\boldsymbol{s}$ as well. We can concatenate the document and the curated text, and produce an encoded representation of this sequence.

$$\boldsymbol{z} = \text{Encoder}([\boldsymbol{d}_i; \boldsymbol{s}_i], \boldsymbol{\theta}) \tag{3}$$

This approach incorporates information from both sources, though it does not differentiate them clearly. Thus, the model may struggle to identify which pieces of information are novel with respect to the curated text.

To clearly identify the information that is already present in the curated text $\boldsymbol{s}$, a model could encode $\boldsymbol{s}$ and $\boldsymbol{d}$ separately, then incorporate both signals into the generative procedure.

Context Receptive Generative (CRG) Model: Our next step was to condition our generative process more concretely on the curated text $\boldsymbol{s}$. We condition the generative process on the representation of $\boldsymbol{s}$ at each time step. Formally:

$$\begin{aligned} \boldsymbol{z_d} &= \text{Encoder}_{\boldsymbol{d}}(\boldsymbol{d}_i, \boldsymbol{\theta_d}) && (4) \\ \boldsymbol{z_s} &= \text{Encoder}_{\boldsymbol{s}}(\boldsymbol{s}_i, \boldsymbol{\theta_s}) && (5) \\ \hat{\boldsymbol{x}}_{\boldsymbol{i}} &\sim \prod_t p(\hat{x}_i^t | [\hat{\boldsymbol{x}}_i^{<t}; \boldsymbol{z_s}], \boldsymbol{z_d}) && (6) \end{aligned}$$

where, $\boldsymbol{\theta_d}$ and $\boldsymbol{\theta_s}$ are the parameters of the encoder for the document $\boldsymbol{d}$ and encoder for the curated text $\boldsymbol{s}$ respectively, $\boldsymbol{z_d}$ and $\boldsymbol{z_s}$ are the encoded representations of the document $\boldsymbol{d_i}$ and curated text $\boldsymbol{s_i}$ respectively. At each time step of generation, the output is conditioned on the tokens generated up to the time step t concatenated with $\boldsymbol{z_s}$. Hence, the generative process is receptive of the context at each time step.

3.2 Extractive models

Generative models that construct new sentences conditioned on the relevant context are compelling but have a number of modeling challenges. Such a model must both select the most relevant content *and* generate a fluent linguistic realization of this information.

We also consider extractive models: approaches that select the most relevant sentence from the document $\boldsymbol{d}$ to append to the curated text $\boldsymbol{s}$. These approaches can focus solely on the content selection problem and ignore the difficulties of generation. This simplification does come at a cost: the most effective sentence to add might require only a subset of information from some sentence in the document, or incorporate information from more than one sentence.

Sum-Basic (SB): One common baseline is Sum-Basic, an extractive summarization technique that relies on word frequency statistics to select salient sentences (Nenkova and Vanderwende, 2005). As an initial step, unigram probabilities are computed from the set of input documents using relative frequency estimation. Then, sentences are selected one-by-one in greedy rounds until the summary budget is saturated. At each round, this model selects the most likely sentence according to the current unigram distribution. The selected sentence is added to the summary and removed from the pool of available sentences. The unigram probabilities of all words in the selected sentence are heuristically discounted (replaced by square root). Select-then-discount operations continue until the summary is written. Discounting is crucial to prevent repetition: once a word (or ideally a concept) has been selected for the summary, it is much less likely to be picked in a subsequent round.

We use Sum-Basic as a Context Agnostic extractive model: we provide the document $\boldsymbol{d}$ as an input to the model and run Sum-Basic for exactly one round. The selected sentence is considered to be the update sentence $\boldsymbol{x}$.

Context Informed Sum-Basic (CISB): We developed a simple modification of the Sum-basic technique to incorporate information from the curated text $\boldsymbol{s}$ as context. Initial unigram probabilities are computed using word counts from *both* the curated text *and* the document. Next, for each sentence in the curated text, we apply just the discount procedure, updating the probability distribution as if those sentences were selected. Finally, we select the single sentence from the document that is most likely according to the resulting discounted unigram probabilities. This simple modification of Sum-Basic helps select a sentence that is novel with respect to the curated text by lowering the probability of all words already present.

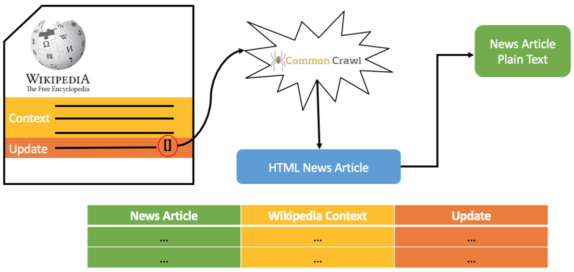


Figure 2: Dataset creation process

Extractive CAG, CIG, CRG Models: Any generative model of $\boldsymbol{x}$ can also be used as an extractive model: we simply estimate the likelihood of each sentence in the document according to the model, and select the most likely one. Generative models may fail because either they are unable to select the most relevant information, or because the resulting sentence is ill-formed. Extractive ranking circumvents all errors due to generation and can help isolate model issues.

Hybrid CAG, CIG, CRG Models: Since the document $\boldsymbol{d}$ can be quite large, a generative model may struggle to pick the most salient information based on the context. To simplify the generative modeling task, we can pre-filter the document toward only the most salient parts. We use the Context Informed Sum-Basic technique to first select the top five sentences from the document. We supply only these five sentences in place of the source document $\boldsymbol{d}$, then apply the CAG, CIG, and CRG techniques described above.

4 Dataset

Our ideal dataset would capture the edits made to some curated reference text in light of a stream of new articles describing changes. For instance, one might maintain reference software documentation about a system, making additions or changes in light of incoming emails describing updates or additions. This type of data is unfortunately difficult to obtain due to privacy considerations.

However, Wikipedia can provide a naturally-occurring body of text with references to primary sources. A substantial fraction of Wikipedia sentences include citations to supporting documentation, a ripe source of data for content transfer. That

Corpus	Input	Output	#Examples	Rouge-1 R
Gigaword (Graff and Cieri, 2003)	10^1	10^1	10^6	78.7
CNN/DailyMail (Nallapati et al., 2016)	10^2–10^3	10^1	10^5	76.1
WikiSum (Liu et al., 2018)	10^2–10^6	10^1–10^3	10^6	59.2
Content Transfer (this paper)	10^1–10^3	10^1–10^2	10^5	66.9

Table 1: Key characteristics of the dataset: approximate size of input and output instances, approximate dataset size, and recall of reference output against the source material, as a measure of dataset difficulty.

said, some of the citations are quite difficult to follow or trust: broken URLs might lead to lost information; citations to books are difficult to consume given the large scope of information; etc. Therefore, we only consider cases where the reference links to some well-known news sources.

Based on citation frequency, we selected a list of 86 domains,[3] primarily news outlets. During the data creation process we only considered citations belonging to one of these eighty six domains. We make this simplifying assumption for several reasons. First, our English Wikipedia dump contained approximately 23.7 million citation URLS belonging to 1.6 million domains; fine-grained filtering would be a daunting task. Our hand-vetted list of domains is a high-precision (albeit low-recall) means of selecting clean data. Second, we wanted to ground the generated text on credible, consistent, and well-written sources of information. Furthermore, well-known domains are readily available on Common Crawl,[4] leading to an easily-reproducible dataset.

Fig. 2 illustrates the procedure used to create a dataset for the task described in Section 2 from Wikipedia. For each Wikipedia article, we extracted the plain text without markdown. When encountering a citation belonging to a selected domain, we considered the sentence just before the citation to be generated based on the content of the citation. This sentence became our reference update sentence: the additional update sentence $\boldsymbol{x}$ added to the curated text $\boldsymbol{s}$ to produce the new text $\boldsymbol{s}'$. The k sentences prior to the target sentence in the Wikipedia article were considered to be the curated text $\boldsymbol{s}$. In our case, we used a window of $k = 3$ sentences to select our context. The cited article acted as the document $\boldsymbol{d}$, from which the appropriate update $\boldsymbol{x}$ can be generated.

The HTML source of the citation was downloaded from Common Crawl for reproducibility and consistency. The HTML derived from Common Crawl is then processed to get the plain text of the news article. The resulting dataset $\boldsymbol{C}$ consists of aligned tuples $\boldsymbol{C} = \left(\boldsymbol{d_i}, \boldsymbol{s_i}, \boldsymbol{x_i}\right)_{i\in[1,n]}$, where n is the total number of samples in the dataset.

Alternatively, one might rely on Wikipedia edit history to create a dataset. In this setting, edits which include a new citation would act as the update $\boldsymbol{x}$. Although this has the upside of identifying potentially complex, multi-sentence updates, preliminary analysis suggested that these edits are noisy. Editors may first generate the content in one edit, then add the citation in a subsequent edit, they may only rephrase a part of the text while adding the citation, or they may check in a range of changes across the document in a single edit. Our simpler sentence-based approach leads to an interesting dataset with fewer complications.

Dataset Statistics and Analysis Table 1 describes some key statistics of our dataset and how it compares with other datasets used for similar tasks. The ROUGE-1 recall scores of reference output $\boldsymbol{x}$ against document $\boldsymbol{d}$ suggest this task will be difficult for conventional extractive summarization techniques.[5] We hypothesize that during content transfer, the language in document $\boldsymbol{d}$ often undergoes substantial transformations to fit the curated text $\boldsymbol{s}$. The average unigram overlap (after stopword removal) between the document $\boldsymbol{d}$ and the reference update sentence $\boldsymbol{x}$ is 55.79%; overlap of the curated text $\boldsymbol{s}$ and the reference update sentence $\boldsymbol{x}$ is 30.12%. This suggests the reference update sentence $\boldsymbol{x}$ can be derived from the document $\boldsymbol{d}$, though not extracted directly. Furthermore, the content of $\boldsymbol{x}$ is very different from the content of $\boldsymbol{s}$ but appears topically related.

Our dataset consists of approximately 290k unique Wikipedia articles. Some heavily-cited

---

[3] This list is provided in the data release of this paper.

[4] `http://commoncrawl.org/`

[5] ROUGE-1 recall was computed on a sample of 50k instances from the entire dataset.

articles include ‘Timeline of investigations into Trump and Russia (2017)’, ‘List of England Test cricketers’, and ‘2013 in science’. We randomly split the dataset into 580k training instances, 6049 validation instances, and 50k test instances, ensuring that any Wikipedia article appearing in the train set must not appear in validation or test.

5 Experimental results

We evaluate our models using both automated metrics and, for a subset of promising systems, human assessment. One key evaluation is the similarity between the model generated update sentence and reference update sentence. We also ask human judges to assess grammaticality and coherence.

Hyper-parameter settings: For all our experiments with generative models, we have used bidirectional encoder, 2 layers in encoder and decoder, RNN size of 128, word vector size of 100. We have used sentencepiece toolkit[6] to use byte-pair-encoding (BPE) with a vocabulary size of 32k. We used stochastic gradient descent optimizer and the stopping criterion was perplexity on the validation set. We filtered our dataset to contain instances which have length of the document between 50 and 2000 tokens, length of the curated text between 20 and 500 tokens and the length of the update sentence between 5 and 200 tokens.

5.1 Automated Evaluation

Our primary automated evaluation metric for system-generated update sentences is ROUGE-L F1 against reference update sentence,[7] though we also include BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2011) as additional indicators. ROUGE is a standard family of metrics for summarization tasks; ROUGE-L measures the longest common subsequence between the system and the reference, capturing both lexical selection and word order.

Table 2 illustrates that this task is quite difficult for extractive techniques. Furthermore, the results emphasize the importance of having curated text as context when generating the update. In all experimental conditions, models aware of context perform much better than models agnostic of it. In contrast to Liu et al. (2018), generative approaches outperformed hybrid, likely because we only had a single input document. Extractive CAG, CIG, and CRG all outperformed both Sum-Basic and the context informed variant. Extractive CAG was on-par with generative CAG, suggesting the generated sentences were of reasonable quality. However, generative CIG and CRG were substantially better: rewriting to match context was beneficial.

Model	ROUGE-L	BLEU	METEOR
SB	5.6 (5.6–5.7)	0.6	2.0
CISB	7.0 (7.0–7.1)	1.0	2.8
CAG	9.1 (9.0–9.2)	1.2	4.6
COG	13.5 (13.4–13.6)	1.7	3.5
CIG	**16.0** (15.9-16.1)	**3.5**	**5.3**
CRG	14.7 (14.6–14.8)	2.6	4.5
Hybrid CAG	8.0 (7.9–8.0)	1.0	3.8
Hybrid CIG	15.0 (14.9–15.1)	2.7	4.7
Hybrid CRG	13.5 (13.4–13.6)	2.3	4.1
Extractive CAG	9.3 (9.2–9.3)	1.1	3.2
Extractive CIG	9.3 (9.2–9.3)	1.1	3.2
Extractive CRG	9.2 (9.1–9.3)	1.1	3.2
Oracle	*28.8* *(28.7–29.0)*	*11.0*	*10.9*

Table 2: Automated metrics; 95% confidence interval in parentheses.

The *Oracle* system of Table 2 aims to establish an upper limit attainable by extractive methods, using the following oracle experiment: For each test instance $(\boldsymbol{d_i}, \boldsymbol{s_i}, \boldsymbol{x_i})$, we enumerate each extracted sentence $\boldsymbol{e}$ of document $\boldsymbol{d_i}$ and select the one with highest ROUGE-L score as *Oracle*’s update sentence $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ (i.e., $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = \arg\max_{\boldsymbol{e} \in \boldsymbol{d_i}} \text{ROUGE-L}(\boldsymbol{x_i}, \boldsymbol{e})$). Note this yields a very optimistic upper bound, as the same ground truth $\boldsymbol{x_i}$ is used both to select an extractive sentence from a large pool of candidates and for final automatic metric scoring.[8] Nevertheless, these oracle results let us draw two conclusions: (1) They give us better perspective to assess the non-oracle systems, and we believe that their seemingly low

---

[6] https://github.com/google/sentencepiece

[7] We use the pyrouge toolkit along with ROUGE-1.5.5: `https://github.com/bheinzerling/pyrouge`

[8] Previous work has shown that this type of oracle can yield upper bounds that are unrealistically high, and they tend to be above human performance (Och et al., 2004, Table 1). One remedy suggested by Och et al. is a round-robin oracle ensuring that the reference (ground truth) used by the argmax is distinct from that of the final automatic evaluation, but that scheme is only possible with a multi-reference test set.

automatic evaluation scores are quite reasonable relative to the optimistic upper bound (e.g., CIGs ROUGE-Ls score is 55% of the oracle). (2) The oracle results suggest that humans are substantially changing the surface realization as they summarize for Wikipedia, as otherwise the oracle results would be much closer to maximum metric scores (i.e., 100%). This shows that extractive methods are not enough for this task, justifying our use of generation techniques.

5.2 Human Evaluations

For careful evaluation of the performance of the most promising configurations (CAG and CIG models) we also asked human judges for quality assessments. We solicited several types of evaluation, including two relative comparisons between pairs of system outputs and an absolute quality evaluation of individual system outputs.

Close to reference (Relative): The first relative comparison measured how accurately the generated update reflected information in the reference update. Here, the annotators saw only the reference update sentence and the outputs of two systems labeled A and B in a randomized order. We asked the annotators "Which system output is closest in meaning to the reference update?" The annotators could pick system A, system B, or indicate that neither was preferred. This is a simple evaluation task though potentially biased toward the sole reference update.

Coherent to context (Relative): The second relative comparison measured whether the generated output contained salient information from the document written in a manner appropriate to the curated text. The annotators saw the document $\boldsymbol{d}$, the curated text $\boldsymbol{s}$, and the outputs of the two systems A and B, again in a random order. They were asked, "Which system output is more accurate relative to the background information given in the snippet of the article?" Each judge had to consider whether the information fits with the curated text and also whether system-generated content could be supported by the document.

Four human judges each annotated 30 unique output pairs for these two relative comparison settings, a total of 240 relative judgments. Table 3 shows the results: the context-aware CIG system was substantially better in both settings.

	prefer		
Evaluation task	CAG	neither	CIG
Close to reference	15.8%	53.3%	30.8%
Coherent to context	7.5%	53.3%	39.2%

Table 3: Human preferences of CAG vs. CIG.

Model	Grammar	Non-redund.	Ref. clarity	Focus	Structure
CAG	2.6	1.8	2.7	2.6	2.4
CIG	**4.3**	**3.9**	**3.6**	**3.5**	**3.2**

Table 4: Human absolute quality assessments.

DUC Guidelines (Absolute): In addition, we performed an absolute quality evaluation following the guidelines from DUC 2007.[9] Each judge was presented with a single system output, then they were asked to evaluate five aspects of system output: grammaticality, non-redundancy, referential clarity, focus, and structure/coherence. For each aspect, the judge provided an assessment on a five-point scale: (1) Very Poor, (2) Poor, (3) Barely Acceptable, (4) Good, (5) Very Good. We gathered 120 additional judgments in this setting (4 judges, 30 outputs). Again, context-aware CIG substantially outperforms CAG across the board, as seen in Table 4.

Observations: Systems unaware of the curated text $\boldsymbol{s}$ tend to generate long updates with repeated frequent words or phrases. Consider the ratio of unique tokens over the total number of tokens in the generated output, which we denote by $\boldsymbol{R}$. A small $\boldsymbol{R}$ indicates many repeated tokens. We find that 88% of the time this ratio $\boldsymbol{R}$ falls below 0.5 for the CAG model, i.e. for 88% instances, more than 50% of the words in the generated output are repeats. This number is relatively small – 14% for CIG and 20% for CRG – in context aware models. In the reference updates only 0.21% instances repeat more than 50% of words.

Figs. 3 and 4 show good and bad examples generated by the CIG model along with the document, curated text and the reference update. Table 5 has a set of updates generated by the CIG model as

---

[9] `http://duc.nist.gov/duc2007/quality-questions.txt`

Document (News Article)

sequels are fairly new to bollywood, but director sanjay gadhvi realised there was cash to be made from resurrecting his hit action thriller dhoom, by casting sexy young stars like hrithik rosha, aishwarya rai and abhishek bachchan in an even bigger game of cops and robbes...that the twist in dhoom 2’s tail is not explained is yet another shortcoming. it’s only roshan’s charismatic performance as the criminal mastermind, and the sizzling chemistry he shares with rai’s sassy cohort, that rescues this adventure from becoming an elongated tourism commercial.

Curated Text (Wikipedia Context)

it makes no lasting contributions to world cinema, but if two-and-a-half hours of disposable entertainment are all you’re after, you could do far worse. “l.a. weekly’s david chute stated the film was, ”a movie meal as satisfying as this one can make you feel that nothing else matters.” jaspreet pandohar of the bbc gave it a two-star rating, writing “by roping in acclaimed action director alan amin to take care of the thrills and spills, you’d expect gadhvi to have spent time crafting out a sophisticated storyline instead of simply sending his cast on a cat-and-mouse chase around the globe.

Reference Update

it’s only roshan’s charismatic performance as the criminal mastermind, and the sizzling chemistry he shares with rai’s sassy cohort, that rescues this adventure from becoming an elongated tourism commercial.”

Generated Update

it’s only roshan’s finest performance as the criminal terrorist, and the sizzling chemistry he shares with rai’s sassy anatomy, that attues this adventure from becoming an elongated tourism commercial.”

Figure 3: Example of good quality generation, where the system-generated update is close to the reference.

Document (News Article)

anne kirkbride, who portrayed bespectacled, gravelly-voiced deirdre barlow in coronation street for more that four decades, has died. the 60-year-old, whose first appearance in the soap opera was in 1972, died in a manchester hospital after a short illness.... kirkbride had left the soap opera after she was diagnosed with non-hodgkin’s lymphoma in 1993 but returned some months later after treatment and spoke candidly about how she had struggled with depression following the diagnosis...

Curated Text (Wikipedia Context)

in 1993, kirkbride was diagnosis with non-hodgkin’s lymphoma. she spoke to the british press about her bout of depression following the diagnosis. she was cured within a year of being diagnosed.

Reference Update

anne kirkbride died of breast cancer in a manchester hospital on 19 january 2015, aged 60.

Generated Update

she was diagnosed with non-hodgkin’s lymphoma.

Figure 4: Example of lower-quality output: the generated update unnecessarily restates information yet misses the most salient detail from the document.

well as the reference update. As we can see in examples 3 and 4, the CIG model misplaces the date but correctly generates the remaining content. In examples 1 and 2, the CIG model appears to successfully select the correct pronouns for co-reference resolution, though it gets confused as to when to use the pronoun or the named entity. Examples 5 and 6 represent failure cases due to missing words.

6 Related Work

The proposed content transfer task is clearly related to a long series of papers in summarization, including recent work with neural techniques (Rush et al., 2015; Nallapati et al., 2016). In particular, one recent paper casts the the task of generating an entire Wikipedia article as a multi-document summarization problem (Liu et al., 2018). Their best-performing configuration was a two-stage extractive-abstractive framework; a multi-stage approach helped circumvent the difficulties of purely abstractive methods given quite large input token sequences.

Looking beyond the clear task similarity of authoring Wikipedia style content, there are several crucial differences in our approach. First, the goal of that paper is to author the whole page, starting from nothing more than a set of primary sources, such as news articles. In practice, however, Wikipedia articles often contain information outside these primary sources, including common sense knowledge, framing statements to set the article in context, and inferences made from those primary sources. Our task restricts the focus to content where a human editor explicitly decided to cite some external source. Hence, it is much more likely that the resulting summary can be derived from the external source content. Furthermore, we focus on the act of adding information to existing articles, rather than writing a complete article without any context. These two scenarios are clearly useful yet complementary: sometimes people want to produce a new reference text where nothing existed before; in other cases the goal is to maintain and update an existing reference.

Another closely related task is update summarization (Dang and Owczarzak, 2008), where systems attempt to provide a brief summary of the novel information in a new article assuming the user has read a known set of prior documents. Our focus on curating an authoritative resource

	Reference Update	Generated Update
1.	rob brydon, the comedian was born in baglan.	he was born in baglan.
2.	in may 2014 he was diagnosed with prostate cancer.	st. clair was diagnosed with prostate cancer.
3.	on april 3, 2014, manning signed a one-year deal with the cincinnati bengals.	on march 9, 2014, manning signed a one-year contract with the cincinnati bengals.
4.	on oct 10, 2013, barrett signed with the memphis grizzlies.	on feb 9, 2013, barrett signed with the memphis grizzlies.
5.	some people think elvis is still alive, but most of us think he's dead and gone."	some people think elvis, but most of us think he's dead and gone."
6.	it's always the goal of the foreign-language film award executive committee to be as inclusive as possible."	it's always the goal of the foreign- entry film award executive to be as possible."

Table 5: Example generations from the CIG system, paired with the human generated updates.

is a substantial difference. Also our datasets are substantially larger, enabling generative models to be used in this space, where prior update summarization techniques have been primarily extractive (Fisher and Roark, 2008; Li et al., 2015).

For any generation task, it is important to address both the content ('what' is being said) as well its style ('how' it is being said). Recently, a great deal of research has focused on the 'how' (Li et al., 2018; Shen et al., 2017), including efforts to collect a parallel dataset that differs in politeness (Rao and Tetreault, 2018), to control author characteristics in the generated sentences (Prabhumoye et al., 2018), to control the perceived personality traits of dialog responses (Zhang et al., 2018). We believe this research thread is complementary to our efforts on generating the 'what'.

Another form of content transfer bridges across modalities: text generation given schematized or semi-structured information. Recent research has addressed neural natural language generation techniques given a range of structured sources: selecting relevant database records and generating natural language descriptions of them (Mei et al., 2016), selecting and describing slot-value pairs for task-specific dialog response generation (Wen et al., 2015), and even generating Wikipedia biography abstracts given Infobox information (Lebret et al., 2016). Our task, while grounded in external content, is different in that it leverages *linguistic* grounding as well as prior text context when generating text. This challenging setting enables a huge range of grounded generation tasks: there are vast amounts of unstructured textual data.

7 Conclusions

This article highlights the importance of the task of *content transfer*: generation guided by an existing curated text to set context and tone, and grounded in a new source providing useful information. We demonstrate how multiple models can address this challenging problem on a novel dataset derived from Wikipedia and Common Crawl. This dataset is released to the community along with scripts and models.[10] We find this setting particularly promising given the opportunity for human interaction: in contrast to approaches that do not rely on human-generated context, we establish a collaboration between user and computer. Each newly suggested sentence can be rejected, accepted, or edited before inclusion, and the edits can provide more training data.

We believe there are many natural extensions to this work. The models described here are mostly extensions of existing approaches; approaches targeting novelty detection, focus, and document structure could lead to substantial improvements. We could apply models in series to incorporate changes for a set of documents. Future work could also explore changes that modify existing content rather than simply appending.

Acknowledgments

We are grateful to the anonymous reviewers, as well as Alan W. Black, Chris Brockett, Bill Dolan, Sujay Jauhar, Michael Gamon, Jianfeng Gao, Dheeraj Rajagopal, and Xuchao Zhang for their helpful comments and suggestions on this work. We also thank Emily Ahn, Khyati Chandu, Ankush Das, Priyank Lathwal, and Dheeraj Rajagopal for their help with the human evaluation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

---

[10] `https://www.microsoft.com/en-us/research/project/content-transfer/`

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *In TAC 2008 Workshop - Notebook papers and results*, pages 10–23.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*, pages 85–91. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proc. of EMNLP*, page 94.

Seeger Fisher and Brian Roark. 2008. Query-focused supervised sentence ranking for update summaries. In *TAC*.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A neural language model for customizable affective text generation. In *ACL*, volume 1, pages 634–642.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

David Graff and Christopher Cieri. 2003. English Gigaword LDC2003T05. In *Philadelphia: Linguistic Data Consortium.*

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213. Association for Computational Linguistics.

Chen Li, Yang Liu, and Lin Zhao. 2015. Improving update summarization via supervised ILP and sentence reranking. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1317–1322.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1865–1874.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Hongyuan Mei, TTI UChicago, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of NAACL-HLT*, pages 720–730.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural*

Language Learning, pages 280–290. Association for Computational Linguistics.

Ani Nenkova and Kathleen R. McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. Technical report, Microsoft Research.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. of HLT-NAACL*, pages 161–168.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 129–140.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Hayahide Yamagishi, Shin Kanouchi, Takayuki Sato, and Mamoru Komachi. 2016. Controlling the voice of a sentence in Japanese-to-English neural machine translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 203–210, Osaka, Japan. The COLING 2016 Organizing Committee.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Improving Machine Reading Comprehension with General Reading Strategies

Kai Sun[1*] **Dian Yu**[2] **Dong Yu**[2] **Claire Cardie**[1]
[1]Cornell University, Ithaca, NY, USA
[2]Tencent AI Lab, Bellevue, WA, USA
`ks985@cornell.edu`, `{yudian, dyu}@tencent.com`
`cardie@cs.cornell.edu`

Abstract

Reading strategies have been shown to improve comprehension levels, especially for readers lacking adequate prior knowledge. Just as the process of knowledge accumulation is time-consuming for human readers, it is resource-demanding to impart rich general domain knowledge into a deep language model via pre-training. Inspired by reading strategies identified in cognitive science, and given limited computational resources – just a pre-trained model and a fixed number of training instances – we propose three general strategies aimed to improve non-extractive machine reading comprehension (MRC): (i) BACK AND FORTH READING that considers both the original and reverse order of an input sequence, (ii) HIGHLIGHTING, which adds a trainable embedding to the text embedding of tokens that are relevant to the question and candidate answers, and (iii) SELF-ASSESSMENT that generates practice questions and candidate answers directly from the text in an unsupervised manner.

By fine-tuning a pre-trained language model (Radford et al., 2018) with our proposed strategies on the largest general domain multiple-choice MRC dataset RACE, we obtain a 5.8% absolute increase in accuracy over the previous best result achieved by the same pre-trained model fine-tuned on RACE without the use of strategies. We further fine-tune the resulting model on a target MRC task, leading to an absolute improvement of 6.2% in average accuracy over previous state-of-the-art approaches on six representative non-extractive MRC datasets from different domains (i.e., ARC, OpenBookQA, MCTest, SemEval-2018 Task 11, ROCStories, and MultiRC). These results demonstrate the effectiveness of our proposed strategies and the versatility and general applicability of our fine-tuned models that incorporate these strategies. Core code is available at `https://github.com/nlpdata/strategy/`.

1 Introduction

Recent years have seen a growing interest in machine reading comprehension (MRC) (Rajpurkar et al., 2016; Choi et al., 2018; Kočiskỳ et al., 2018; Reddy et al., 2018). In this paper, we mainly focus on *non-extractive MRC* (Khashabi et al., 2018; Ostermann et al., 2018; Clark et al., 2018), in which a significant percentage of candidate answers are not restricted to text spans from the reference document or corpus. In comparison to *extractive MRC* tasks (Section 2.1), non-extractive MRC (Section 2.2) requires diverse reading skills and, as a result, the performance of machine readers on these tasks more accurately indicates the comprehension ability of machine readers in realistic settings such as exams (Lai et al., 2017).

Recently, significant progress has been achieved on many natural language processing tasks including MRC by fine-tuning a pre-trained general-purpose language model (Radford et al., 2018; Devlin et al., 2018). However, similar to the process of knowledge accumulation for human readers, it is time-consuming and resource-demanding to impart massive amounts of general domain knowledge from external corpora into a deep language model via pre-training. For example, it takes a month to pre-train a 12-layer transformer on eight P100 GPUs over the BooksCorpus (Zhu et al., 2015; Radford et al., 2018); Devlin et al. (2018) pre-train a 24-layer transformer using 64 TPUs for four days on the BooksCorpus plus English Wikipedia, a feat not easily reproducible considering the tremendous computational resources (≈ one year to train on eight P100 GPUs).

From a practical viewpoint, given a limited number of training instances and a pre-trained

---

[*] This work was done when K. S. was an intern at the Tencent AI Lab, Bellevue, WA.

Proceedings of NAACL-HLT 2019, pages 2633–2643
Minneapolis, Minnesota, June 2 - June 7, 2019.

model, can we improve machine reading comprehension during fine-tuning instead of imparting more prior knowledge into a model via expensive pre-training? Inspired by reading strategies identified in cognitive science research that have been shown effective in improving comprehension levels of human readers, especially those who lack adequate prior knowledge of the topic of the text (Mokhtari and Sheorey, 2002; Mokhtari and Reichard, 2002; McNamara et al., 2004), we propose three corresponding domain-independent strategies to improve MRC based on an existing pre-trained transformer (Section 3.1):

- BACK AND FORTH READING (*"I go back and forth in the text to find relationships among ideas in it."*):
  consider both the original and reverse order of an input sequence (Section 3.2)
- HIGHLIGHTING (*"I highlight information in the text to help me remember it."*):
  add a trainable embedding to the text embedding of those tokens deemed relevant to the question and candidate answers (Section 3.3)
- SELF-ASSESSMENT (*"I ask myself questions I would like to have answered in the text, and then I check to see if my guesses about the text are right or wrong."*):
  generate practice questions and their associated span-based candidate answers from the existing reference documents (Section 3.4)

By fine-tuning a pre-trained transformer (Radford et al., 2018) according to our proposed strategies on the largest general domain multiple-choice MRC dataset RACE (Lai et al., 2017) collected from language exams, we obtain a 5.8% absolute improvement in accuracy over the previous best result achieved by the same pre-trained transformer fine-tuned on RACE without the use of strategies (Section 4.2). We further fine-tune the resulting model on a target MRC task. Experiments show that our method achieves new state-of-the-art results on six representative non-extractive MRC datasets that require a range of reading skills such as commonsense and multi-sentence reasoning (i.e., ARC (Clark et al., 2016, 2018), OpenBookQA (Mihaylov et al., 2018), MCTest (Richardson et al., 2013), SemEval-2018 Task 11 (Yang et al., 2017), ROCStories (Mostafazadeh et al., 2016), and MultiRC (Khashabi et al., 2018)) (Section 4.4). These results indicate the effectiveness of our proposed strategies and the versatility and generality of our fine-tuned models that incorporate the strategies.

2 Task Introduction

We roughly categorize machine reading comprehension tasks into two groups: extractive (Section 2.1) and non-extractive (Section 2.2) based on the expected answer types.

2.1 Extractive MRC

Recently large-scale extractive MRC datasets have been constructed (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016; Chen and Choi, 2016; Mostafazadeh et al., 2016; Bajgar et al., 2016; Nguyen et al., 2016; Joshi et al., 2017; Ma et al., 2018), such as SQuAD (Rajpurkar et al., 2016) and NewsQA (Trischler et al., 2017). Given a reference document and a question, the expected answer is a short span from the document. In contrast, answers in datasets such as SearchQA (Dunn et al., 2017) and NarrativeQA (Kočiský et al., 2018) are free-form human generated texts based on given documents (Nguyen et al., 2016; Reddy et al., 2018; Choi et al., 2018). However, since annotators tend to directly copy spans as answers, the majority of answers are still extractive (Reddy et al., 2018; Kočiský et al., 2018).

2.2 Non-Extractive MRC

In this section, we primarily discuss multiple-choice MRC datasets, in which answer options are not restricted to extractive text spans. Given a question and a reference document/corpus, multiple answer options are provided, and at least one of them is correct. It involves extensive human efforts to build such a dataset (e.g., MCTest (Richardson et al., 2013), SemEval-2018 Task 11 (Ostermann et al., 2018), MultiRC (Khashabi et al., 2018), and OpenBookQA (Mihaylov et al., 2018)) by crowdsourcing. Besides crowdsourcing, datasets such as RACE (Lai et al., 2017) and ARC (Clark et al., 2018) are collected from language or science exams designed by educational experts (Penas et al., 2014; Shibuki et al., 2014; Tseng et al., 2016) to evaluate the comprehension level of human participants. Compared to questions in extractive MRC tasks, besides surface matching, there are various types of complicated questions such as math word problems, summarization, logical reasoning, and sentiment analysis, requiring advanced read-

	RACE	ARC	OpenBookQA	MCTest	SemEval-2018 Task 11	ROCStories	MultiRC
construction method	exams	exams	crowd.	crowd.	crowd.	crowd.	crowd.
sources of documents	general	science	science	stories	narrative text	stories	mixed-domain
average # of answer options	4.0	4.0	4.0	4.0	2.0	2.0	5.4
# of documents	27,933	14M†	1,326†	660	2,119	3,742	871
# of questions	**97,687**	7,787	5,957	2,640	13,939	–	9,872
non-extractive answer* (%)	87.0	43.3	83.8	45.3	89.9	100.0	82.1

Table 1: Statistics of multiple-choice machine reading comprehension datasets. Some values come from Reddy et al. (2018), Kočiský et al. (2018), and Lai et al. (2017) (crowd.: crowdsourcing; †: regarding each sentence/claim as a document (Clark et al., 2018); *: correct answer options that are not text snippets from reference documents).

ing skills and prior world knowledge. Besides, in most cases, we can adopt an objective evaluation criteria such as accuracy to evaluate system performance (Clark et al., 2016; Lai et al., 2017). As these kind of datasets are relatively difficult to construct or collect, most existing datasets are small in size, which hinders the development of state-of-the-art deep neural models.

In response, in this paper we investigate how to make use of limited resources to improve MRC, using seven representative multiple-choice MRC datasets as case studies. As shown in Table 1, the majority of the correct answer options in most of the datasets (except for ARC and MCTest) are non-extractive. Except for MultiRC, there is exactly one correct answer option for each question. For ARC and OpenBookQA, a reference corpus is provided instead of a single reference document associated with each question.

Here we give a formal **task definition**. Given a reference document d, a question q, and associated answer options $\{o_1, o_2, \ldots, o_m\}$, the goal is to select the correct answer option(s). We can easily adapt our method to an MRC task that only provides a reference corpus (Section 4.4).

3 Approach

We first introduce a neural reader based on a pre-trained transformer (Section 3.1) and then elaborate on the strategies that are applied during the fine-tuning stage — back and forth reading (Section 3.2), highlighting (Section 3.3), and self-assessment (Section 3.4).

3.1 Framework Overview

Our neural reader follows the framework of discriminatively fine-tuning a generative pre-trained transformer (GPT) (Radford et al., 2018). It adapts a pre-trained multi-layer transformer (Vaswani et al., 2017; Liu et al., 2018) language model to a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens $x^1, \ldots, x^n$, along with a label y, by maximizing:

$$\sum_{x,y} \log P(y \,|\, x^1, \ldots, x^n) + \lambda \cdot L(\mathcal{C}) \qquad (1)$$

where L is the likelihood from the language model, λ is the weight of language model, and $P(y \,|\, x^1, \ldots, x^n)$ is obtained by a linear classification layer over the final transformer block's activation of the language model. For multiple-choice MRC tasks, $x^1, \ldots, x^n$ come from the concatenation of a start token, a reference document, a question, a delimiter token, an answer option, and an end token; y indicates the correctness of an answer option. We refer readers to Radford et al. (2018) for more details.

Apart from placing a delimiter to separate the answer option from the document and question, the original framework pays little attention to task-specific structures in MRC tasks. Inspired by reading strategies, with limited resources and a pre-trained transformer, we propose three strategies to improve machine reading comprehension. We show the whole framework in Figure 1.

3.2 Back and Forth Reading (BF)

For simplicity, we represent the original input sequence of GPT during fine-tuning (Radford et al., 2018) as $[dq\,\$\,o]$, where [, \$, and] represent the start token, delimiter token, and end token, respectively. Inspired by back and forth reading, we consider both the original order and the reverse order $[o\,\$\,qd]$. The token order within d, q, and o is still preserved. We fine-tune two GPTs that use $[dq\,\$\,o]$ and $[o\,\$\,qd]$ as the input sequence respectively, and then we ensemble the two models. We also consider other similar pairs of input sequences such as $[qd\,\$\,o]$ and $[o\,\$\,dq]$ in the experiments (Section 4.3).

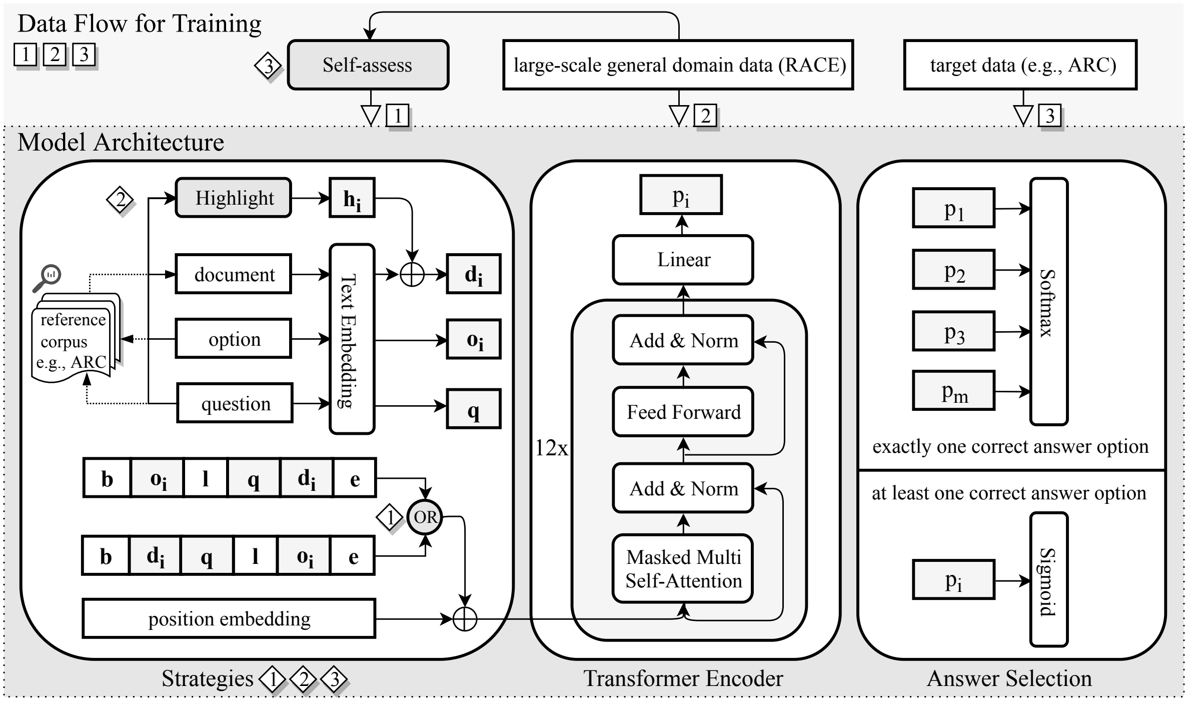


Figure 1: Framework Overview. Strategy 1, 2, and 3 refer to back and forth reading (BF) (Section 3.2), highlighting (HL) (Section 3.3), and self-assessment (SA) (Section 3.4), respectively.

3.3 Highlighting (HL)

In the original implementation (Radford et al., 2018), during the fine-tuning stage of GPT, the text embedding of a document is independent of its associated questions and answer options. Inspired by highlights used in human reading, we aim to make the document encoding aware of the associated question-answer option pair (q, o_i). We focus on the content words in questions and answer options since they appear to provide more useful information (Mirza and Bernardi, 2013), and we identify them via their part of speech (POS) tags, one of: noun, verb, adjective, adverb, numeral, or foreign word.

Formally, we let T be the set of POS tags of the content words. We let $\boldsymbol{d}$ denote the sequence of the text embedding of document d. We use d^j to represent the j^{th} token in d and $\boldsymbol{d^j}$ to denote the text embedding of d^j. Given d and a (q, o_i) pair, we define a *highlight embedding* $\boldsymbol{h_i^j}$ for the j^{th} token in d as:

$$\boldsymbol{h_i^j} = \begin{cases} \boldsymbol{\ell^+} & \text{if the POS tag of } d^j \text{ belongs to } T, \\ & \text{and } d^j \text{ appears in either } q \text{ or } o_i \\ \boldsymbol{\ell^-} & \text{otherwise} \end{cases} \qquad (2)$$

where $\boldsymbol{\ell^+}$ and $\boldsymbol{\ell^-}$ are two trainable vectors of the same dimension as $\boldsymbol{d^j}$.

Following the above definition, the sequence of the highlight embedding $\boldsymbol{h_i} = \boldsymbol{h_i^1}, \boldsymbol{h_i^2}, \ldots, \boldsymbol{h_i^n}$ is of the same length as $\boldsymbol{d}$. We replace $\boldsymbol{d}$ with $\boldsymbol{d_i} = \boldsymbol{d} + \boldsymbol{h_i}$ when we encode a document. More specifically, we use the concatenation of $\boldsymbol{b}$, $\boldsymbol{d_i}$, $\boldsymbol{q}$, $\boldsymbol{l}$, $\boldsymbol{o_i}$, and $\boldsymbol{e}$ as the new input of GPT during fine-tuning (Section 3.1), where $\boldsymbol{b}$, $\boldsymbol{l}$, and $\boldsymbol{e}$ denote the embedding of the start token, delimiter token, and end token, respectively, and $\boldsymbol{q}$ and $\boldsymbol{o_i}$ represent the sequence of the text embedding of q and o_i, respectively.

3.4 Self-Assessment (SA)

While in previous work (Radford et al., 2018), the original GPT is directly fine-tuned on an MRC end task, we instead develop a fine-tuning approach inspired by the self-assessment reading strategy. In particular, we propose a simple method to generate questions and their associated multiple span-based answer options, which cover the content of multiple sentences from a reference document. By first fine-tuning a pre-trained model on these *practice* instances, we aim to render the resulting fine-tuned model more aware of the input structure and to integrate information across multiple sentences as may be required to answer a given question.

Concretely, we randomly generate no more than n_q questions and associated answer options based on each document from the end task (i.e., RACE in this paper). We describe the steps as follows.

- **Input:** a reference document from the end task.
- **Output:** a question and four answer options associated with the reference document.

1. Randomly pick no more than n_s sentences from the document and concatenate these sentences together.
2. Randomly pick no more than n_c non-overlapping spans from the concatenated sentences. Each span randomly contains no more than n_t tokens within a single sentence. We concatenate the selected spans to form the correct answer option. We remove the selected spans from the concatenated sentences and use the remaining text as the question.
3. Generate three distractors (i.e., wrong answer options) by randomly replacing spans in the correct answer option with randomly picked spans from the document.

where n_q, n_s, n_c, and n_t are used to control the number and difficulty level of the questions.

4 Experiment

4.1 Experiment Settings

For most of the hyperparameters, we follow the work of Radford et al. (2018). We use the same preprocessing procedure and the released pretrained transformer. We generate 119k instances based on the reference documents from the training and development set of RACE (Lai et al., 2017), with $n_q = 10$, $n_s = 3$, $n_c = 4$, and $n_t = 4$ (Section 3.4). We first fine-tune the original pretrained model on these automatically generated instances with 1 training epoch (data flow 1 boxed in Figure 1). We then fine-tune the model on a large-scale general domain MRC dataset RACE with 5 training epochs (data flow 2 boxed in Figure 1). Finally, we fine-tune the resulting model on one of the aforementioned six out-of-domain MRC datasets (at max 10 epochs). See data flow 3 boxed in Figure 1. When we fine-tune the model on different datasets, we set the batch size to 8, language model weight λ to 2. We ensemble models by averaging logits after the linear layer. For strategy highlighting (Section 3.3), the content-word POS tagset $T = \{$NN, NNP, NNPS, NNS, VB, VBD, VBG, VBN, VBP, VBZ, JJ, JJR, JJS, RB, RBR, RBS, CD, FW$\}$, and we randomly initialize ℓ^+ and ℓ^-.

Approach		**#**	**RACE-M**	**RACE-H**	**RACE**
MMN (Tang et al., 2019)		9	64.7	55.5	58.2
GPT (Radford et al., 2018)		1	62.9	57.4	59.0
Human performance (Lai et al., 2017)		1	85.1	69.4	73.3
GPT*		1	60.9	57.8	58.7
		2	62.6	58.4	59.6
		9	63.5	59.3	60.6
GPT* + Strategies	SA	1	63.2	59.2	60.4
	HL	1	67.4	61.5	63.2
	BF	2	67.3	60.7	62.6
	SA + HL	1	**69.2**	**61.5**	**63.8**
	SA + HL + BF	2	**70.9**	**63.2**	**65.4**
	SA + HL + BF	9	**72.0**	**64.5**	**66.7**

Table 2: Accuracy (%) on the test set of RACE (#: number of (ensemble) models; SA: Self-Assessment; HL: Highlighting; BF: Back and Forth Reading; *: our implementation).

4.2 Evaluation on RACE

In Table 2, we first report the accuracy of the state-of-the-art models (MMN and original fine-tuned GPT) and Amazon Turkers (Human performance). We then report the performance of our implemented fine-tuned GPT baselines and our models (GPT+Strategies). Results are shown on the RACE dataset (Lai et al., 2017) and its two subtasks: RACE-M collected from middle school exams and RACE-H collected from high school exams.

Our single and ensemble models outperform previous state-of-the-art (i.e., GPT and GPT (9×)) by a large margin (63.8% vs. 59.0%; 66.7% vs. 60.6%). The two single-model strategies – self-assessment and highlighting – improve over the single-model fine-tuned GPT baseline (58.7%) by 1.7% and 4.5%, respectively. Using the back and forth reading strategy, which involves two models, gives a 3.0% improvement in accuracy compared to the ensemble of two original fine-tuned GPTs (59.6%). Strategy combination further boosts the performance. By combining self-assessment and highlighting, our single model achieves a 5.1% improvement in accuracy over the fine-tuned GPT baseline (63.8% vs. 58.7%). We apply all the strategies by ensembling two such single models that read an input sequence in either the original or the reverse order, leading to a 5.8% improvement in accuracy over the ensemble of two original fine-tuned GPTs (65.4% vs. 59.6%).

To further analyze performance, we roughly divide the question types into five categories: detail (*facts and details*), inference (*reasoning ability*), main (*main idea or purpose of a document*), attitude (*author's attitude toward a topic or tone/source of a document*), and vocabulary (*vocabulary questions*) (Qian and Schedl, 2004; Lai et al., 2017) and annotate all the instances of the RACE development set. As shown in Figure 2, compared to the fine-tuned GPT baseline, our single-model strategies (SA and HL) consistently improve the results across all categories. Compared to other strategies, highlighting is likely to lead to bigger gains for most question types.

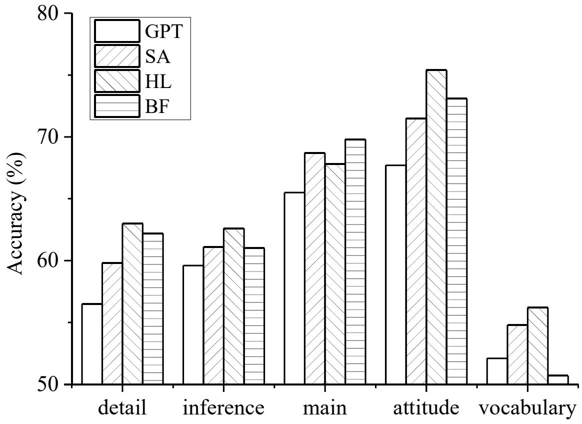


Figure 2: Performance on different question types.

Compared to human performance, there is still a considerable room for improvements, especially on RACE-M. We take a close look at the instances from the RACE-M development set that all our implementations fail to answer correctly. We notice that 82.0% of them require one or multiple types of world knowledge (e.g., negation resolution, commonsense, paraphrase, and mathematical/logic knowledge (Sugawara et al., 2017b,a, 2018)), especially when correct answer options are not explicitly mentioned in the reference document. For example, we need the knowledge — *the type of thing that is written by a writer can probably be a book* — to answer the question *"follow your heart is a _"* from the context *"Follow your heart by Andrew Matthews, an Australian writer, tells us that making our dreams real is life's biggest challenge"*. Besides, 19.7% of these failed instances require coreference resolution. It might be promising to leverage coreference resolvers to connect nonadjacent relevant sentences.

4.3 Further Discussions on Strategies

Besides the strategies introduced in Section 3, we also explore other reading strategies such as SUMMARIZATION (*"I take an overall view of the text to see what it is about before carefully reading it."*) by appending an extractive summary (Boudin et al., 2015) before each reference document, which is shown less effective for machine reading comprehension in our experiments compared to the strategies we focus on. In this section, we further discuss the three strategies.

Back and Forth Reading We notice that the input order difference between two ensemble models is likely to yield performance gains. Besides ensembling two models that use input sequence $[dq\,\$\,o]$ and $[o\,\$\,qd]$ respectively, we also investigate other reverse or almost reverse pairs. For example, we can achieve better results by ensembling $[qd\,\$\,o]$ and $[o\,\$\,dq]$ (61.0%) or $[qd\,\$\,o]$ and $[o\,\$\,qd]$ (61.7%), compared to the ensemble of two original fine-tuned GPTs (both of them use $[d\,\$\,qo]$) on the RACE dataset (59.6% in Table 2).

Highlighting We try two variants to define highlight embeddings (Equation 2 in Section 3.3) by considering the content of questions only or answer options only. Experiments show that using partial information yields a decrease in accuracy (60.6% and 61.0%, respectively) compared to 63.2% (Table 2), achieved by considering the content words in a question and its answer options. We attempt to also highlight the coreferential mentions of the content words, which does not lead to further gains, though.

Self-Assessment We explore alternative approaches to generate questions. For example, we use the Wikipedia articles from SQuAD (Rajpurkar et al., 2016) instead of the general domain documents from the end task RACE. We generate the same number of questions as the number of questions we generate using RACE following the same steps mentioned in Section 3.4. Experiments show that this method also improves the accuracy of the fine-tuned GPT baseline (59.7% vs. 58.7%). As self-assessment can be somehow regarded as a data augmentation method, we investigate other unsupervised question generation methods such as sentence shuffling and paraphrasing via back-translation (Ding and Zhou, 2018; Yu et al., 2018). Our experiments demonstrate that neither of them results in performance improvements on the RACE dataset.

Task	Metric	Previous STOA		GPT	GPT (2×)	GPT+Strategies	GPT+Strategies (2×)
ARC-Easy	Acc.	Clark et al. (2018)	62.6	57.0	57.1	66.6	68.9
ARC-Challenge	Acc.	Ni et al. (2018)	36.6	38.2	38.4	40.7	42.3
OpenBookQA	Acc.	Mihaylov et al. (2018)	50.2	52.0	52.8	55.2	55.8
MCTest-MC160	Acc.	Chung et al. (2018)	76.4	65.4	65.8	80.0	81.7
MCTest-MC500	Acc.	Chung et al. (2018)	72.3	61.5	61.0	78.7	82.0
SemEval	Acc.	Chen et al. (2018)	84.1	88.0	88.6	88.8	89.5
ROCStories	Acc.	Radford et al. (2018)	86.5	87.1	87.5	88.0	88.3
MultiRC	F1_m	Khashabi et al. (2018)	66.5	69.3	70.3	71.5	73.1
	F1_a	Khashabi et al. (2018)	63.2	67.2	67.7	69.2	70.5
	Acc.†	Khashabi et al. (2018)	11.8	15.2	16.5	22.6	21.8
Average	Acc.		60.1	58.1	58.5	**65.1**	**66.3**

Table 3: Performance (%) on the test sets of ARC, OpenBookQA, MCTest, SemEval-2018 Task 11, and ROCStories and the development set of MultiRC (Acc.: Accuracy; F1_m: macro-average F1; F1_a: micro-average F1; †: using the joint exact match accuracy (i.e., EM_0 reported by the official evaluation (Khashabi et al., 2018))). RACE is used as the source task for all our implementations.

Approach	ARC	OpenBookQA	MCTest	SemEval	ROCStories	MultiRC	Average
	Easy | Challenge	-	MC160 | MC500	-	-	-	-
	Acc.	Acc.	Acc.	Acc.	Acc.	F1_m | F1_a | Acc.†	Acc.
GPT	54.0 | 30.3	50.0	58.8 | 52.0	87.3	86.7	69.3 | 66.2 | 11.9	53.9
GPT (2×)	53.9 | 30.7	50.0	60.0 | 54.0	88.0	87.0	69.3 | 66.5 | 13.1	54.6
GPT+Strategies	61.9 | 35.0	54.2	67.5 | 64.7	87.6	87.4	68.8 | 67.4 | 16.2	**59.3**
GPT+Strategies (2×)	63.1 | 35.4	55.0	70.8 | 64.8	88.1	88.1	69.7 | 67.9 | 16.9	**60.3**

Table 4: Performance (%) on the test sets of ARC, OpenBookQA, MCTest, SemEval-2018 Task 11, and ROCStories and the development set of MultiRC using the target data only (i.e., without the data flow 1 and 2 boxed in Figure 1) (Acc.: Accuracy; F1_m: macro-average F1; F1_a: micro-average F1; †: using the joint exact match accuracy (i.e., EM_0 reported by the official evaluation (Khashabi et al., 2018))).

4.4 Adaptation to Other Non-Extractive Machine Reading Comprehension Tasks

We follow the philosophy of transferring the knowledge from a high-performing model pre-trained on a large-scale supervised data of a source task to a target task, in which only a small amount of training data is available (Chung et al., 2018). RACE has been used to pre-train a model for other MRC tasks as it contains the largest number of general domain non-extractive questions (Table 1) (Ostermann et al., 2018; Wang et al., 2018a). In our experiment, we also treat RACE as the source task and regard six representative non-extractive multiple-choice MRC datasets from multiple domains as the target tasks.

We require some task-specific modifications considering the different structures of these datasets. In ARC and OpenBookQA, there is no reference document associated with each question. Instead, a reference corpus is provided, which consists of unordered science-related sentences relevant to questions. We therefore first use Lucene (McCandless et al., 2010) to retrieve the top 50 sentences by using the non-stop words in a question and one of its answer options as a query. The retrieved sentences are used to form the reference document for each answer option. In MultiRC, a question could have more than one correct answer option. Therefore, we use a sigmoid function instead of softmax at the final layer (Figure 1) and regard the task as a binary (i.e., correct or incorrect) classification problem over each (document, question, answer option) instance. When we adapt our method to the non-conventional MRC dataset ROCStories, which aims at choosing the correct ending to a four-sentence incomplete story from two answer options (Mostafazadeh et al., 2016), we leave the question context empty as no explicit questions are provided. Since the test set of MultiRC is not publicly available, we report the performance of the model that achieves the highest micro-average F1 (F1_a) on the development set. For other tasks, we select the model that achieves the highest accuracy on the development set and report the accuracy on the test set.

We first fine-tune GPT using our proposed three strategies on RACE and further fine-tune the resulting model on one of the six target tasks (see Table 3). During the latter fine-tuning stage, besides the *highlighting* embeddings inherited from the previous fine-tuning stage, we also apply the strategy *back and forth reading*, and we do not consider *self-assessment* since the model has already benefited from the high-quality RACE instances during the first fine-tuning stage. We compare with the baselines that are first fine-tuned on RACE and then fine-tuned on a target task without the use of strategies, which already outperform previous state-of-the-art (SOTA) on four out of six datasets (OpenBookQA, SemEval-2018 Task 11, ROCStories, and MultiRC). By using the strategies, we obtain a 7.8% absolute improvement in average accuracy over the ensemble baseline (58.5%) and a 6.2% absolute improvement over previous SOTA (60.1%).

To further investigate the contribution of the strategies, we directly fine-tune GPT on a target task without using the labeled data in RACE (i.e., we only keep data flow 3 in Figure 1). Compared to the baseline that is fine-tuned without using strategies (54.6%), we obtain a 10.4% relative improvement in average accuracy (60.3%) and especially large improvements on datasets ARC, OpenBookQA, and MCTest (Table 4).

5 Related Work

5.1 Methods for Multiple-Choice Machine Reading Comprehension

We primarily discuss methods applied to large-scale datasets such as RACE (Lai et al., 2017). Researchers develop a variety of methods with attention mechanisms (Chen et al., 2016; Dhingra et al., 2017; Xu et al., 2018; Tay et al., 2018; Tang et al., 2019) for improvement such as adding an elimination module (Parikh et al., 2018) or applying hierarchical attention strategies (Zhu et al., 2018; Wang et al., 2018b). These methods seldom take the rich external knowledge (other than pre-trained word embeddings) into considerations. Instead, we investigate different strategies based on an existing pre-trained transformer (Radford et al., 2018) (Section 3.1), which leverages rich linguistic knowledge from external corpora and achieves state-of-the-art performance on a wide range of natural language processing tasks including machine reading comprehension.

5.2 Transfer Learning for Machine Reading Comprehension and Question Answering

Transfer learning techniques have been successfully applied to machine reading comprehension (Golub et al., 2017; Chung et al., 2018) and question answering (Min et al., 2017; Wiese et al., 2017). Compared to previous work, we simply fine-tune our model on the source data and then further fine-tune the entire model on the target data. The investigation of methods such as adding additional parameters or an L2 loss and fine-tuning only part of the parameters is beyond the scope of this work.

5.3 Data Augmentation for Machine Reading Comprehension Without Using External Datasets

Previous methods augment the training data for extractive machine reading comprehension and question answering by randomly reordering words or shuffling sentences (Ding and Zhou, 2018; Li and Zhou, 2018) or generating questions through paraphrasing (Yang et al., 2017; Yuan et al., 2017), which require a large amount of training data or limited by the number of training instances (Yu et al., 2018). In comparison, our problem (i.e., question and answer options) generation method does not rely on any existing questions in the training set, and the generated questions can involve the content of multiple sentences in a reference document.

6 Conclusions

Inspired by previous research on reading strategies for improved comprehension levels of human readers, we propose three strategies (i.e., back and forth reading, highlighting, and self-assessment), aiming at improving machine reading comprehension using limited resources: a pre-trained language model and a limited number of training instances. By applying the proposed three strategies, we obtain a 5.8% absolute improvement in accuracy over the state-of-the-art performance on the RACE dataset. By fine-tuning the resulting model on a new target task, we achieve new state-of-the-art results on six representative non-extractive MRC datasets from multiple domains that require a diverse range of reading skills. These results consistently indicate the effectiveness of our proposed strategies and the general applicability of our fine-tuned model that incorporates these strategies.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive suggestions. We thank Hai Wang, Chengzhu Yu, and Chao Weng for their useful discussions. We especially thank Chao for helping us speed up the release of the preprint[1] with technical supports. We thank Rishi Bommasani for proofreading the paper and Saku Sugawara for sharing annotations with us.

References

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2016. Embracing data abundance: Booktest dataset for reading comprehension. *CoRR*, cs.CL/1610.00956v1.

Florian Boudin, Hugo Mougard, and Benoit Favre. 2015. Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions. In *Proceedings of the EMNLP*, pages 17–21, Lisbon, Portugal.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the ACL*, pages 2358–2367, Berlin, Germany.

Yu-Hsin Chen and Jinho D Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in TV shows. In *Proceedings of the SIGDial*, pages 90–100, Los Angeles, CA.

Zhipeng Chen, Yiming Cui, Wentao Ma, Shijin Wang, Ting Liu, and Guoping Hu. 2018. HFL-RC system at SemEval-2018 Task 11: Hybrid multi-aspects model for commonsense reading comprehension. *CoRR*, cs.CL/1803.05655v1.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the EMNLP*, pages 2174–2184, Brussels, Belgium.

Yu-An Chung, Hung-yi Lee, and James Glass. 2018. Supervised and unsupervised transfer learning for question answering. In *Proceedings of the NAACL-HLT*, pages 1585–1594, New Orlean, LA.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *CoRR*, cs.CL/1803.05457v1.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the AAAI*, pages 2580–2586, Phoenix, AZ.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the NAACL-HLT*, Minneapolis, MN.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the ACL*, pages 1832–1846, Vancouver, Canada.

Peng Ding and Xiaobing Zhou. 2018. Ynu deep at semeval-2018 task 12: A bilstm model with neural attention for argument reasoning comprehension. In *Proceedings of The SemEval*, pages 1120–1123, New Orleans, LA.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new Q&A dataset augmented with context from a search engine. *CoRR*, cs.CL/1704.05179v3.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. *CoRR*, cs.CL/1706.09789v3.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the NIPS*, pages 1693–1701, Montreal, Canada.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the ICLR*, Caribe Hilton, Puerto Rico.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, cs.CL/1705.03551v2.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the NAACL-HLT*, pages 252–262, New Orleans, LA.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the EMNLP*, pages 785–794, Copenhagen, Denmark.

---

[1] `https://arxiv.org/abs/1810.13441v1`.

Yongbin Li and Xiaobing Zhou. 2018. Lyb3b at semeval-2018 task 11: Machine comprehension task using deep learning models. In *Proceedings of the SemEval*, pages 1073–1077, New Orleans, LA.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proceedings of the ICLR*, Vancouver, Canada.

Kaixin Ma, Tomasz Jurczyk, and Jinho D Choi. 2018. Challenging reading comprehension on daily conversation: Passage completion on multiparty dialog. In *Proceedings of the NAACL-HLT*, pages 2039–2048, New Orleans, LA.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT.

Danielle S McNamara, Irwin B Levinstein, and Chutima Boonthum. 2004. iSTART: Interactive strategy training for active reading and thinking. *Behavior Research Methods, Instruments, & Computers*, 36(2):222–233.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the EMNLP*, pages 2381–2391, Brussels, Belgium.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the ACL*, pages 510–517, Vancouver, Canada.

Paramita Mirza and Raffaella Bernardi. 2013. Ccg categories for distributional semantic models. In *Proceedings of the RANLP*, pages 467–474, Hissar, Bulgaria.

Kouider Mokhtari and Carla A Reichard. 2002. Assessing students' metacognitive awareness of reading strategies. *Journal of educational psychology*, 94(2):249.

Kouider Mokhtari and Ravi Sheorey. 2002. Measuring esl students' awareness of reading strategies. *Journal of developmental education*, 25(3):2–11.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *Proceedings of the NAACL-HLT*, pages 839–849, San Diego, CA.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, cs.CL/1611.09268v2.

Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2018. Learning to attend on essential terms: An enhanced retriever-reader model for open-domain question answering. In *Proceedings of the NAACL-HLT*, Minneapolis, MN.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did What: A large-scale person-centered cloze dataset. In *Proceedings of the EMNLP*, pages 2230–2235, Austin, TX.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. SemEval-2018 Task 11: Machine comprehension using commonsense knowledge. In *Proceedings of the SemEval*, pages 747–757, New Orleans, LA.

Soham Parikh, Ananya Sai, Preksha Nema, and Mitesh M Khapra. 2018. Eliminet: A model for eliminating options for reading comprehension with multiple choice questions. In *Proceedings of the IJCAI-ECAI*, pages 4272–4278, Stockholm, Sweden.

Anselmo Penas, Yusuke Miyao, Alvaro Rodrigo, Eduard H Hovy, and Noriko Kando. 2014. Overview of CLEF QA Entrance Exams Task 2014. In *Proceedings of the CLEF*, pages 1194–1200, Sheffield, UK.

David D Qian and Mary Schedl. 2004. Evaluation of an in-depth vocabulary knowledge measure for assessing reading performance. *Language Testing*, 21(1):28–52.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *Preprint*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the EMNLP*, pages 2383–2392, Austin, TX.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. CoQA: A conversational question answering challenge. *Transactions of the Association of Computational Linguistics*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the EMNLP*, pages 193–203, Seattle, WA.

Hideyuki Shibuki, Kotaro Sakamoto, Yoshinobu Kano, Teruko Mitamura, Madoka Ishioroshi, Kelly Y Itakura, Di Wang, Tatsunori Mori, and Noriko Kando. 2014. Overview of the NTCIR-11 QA-Lab Task. In *NTCIR*.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of the EMNLP*, pages 4208–4219, Brussels, Belgium.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017a. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the ACL*, pages 806–817, Vancouver, Canada.

Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. 2017b. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *Proceedings of the AAAI*, pages 3089–3096, San Francisco, CA.

Min Tang, Jiaran Cai, and Hankz Hankui Zhuo. 2019. Multi-matching network for multiple choice reading comprehension. In *Proceddings of the AAAI*, Honolulu, HI.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *CoRR*, cs.CL/1803.09074v1.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the RepL4NLP*, pages 191–200, Vancouver, Canada.

Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Towards machine comprehension of spoken content: Initial toefl listening comprehension test by machine. In *Proceedings of the Interspeech*, San Francisco, CA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the NIPS*, pages 5998–6008, Long Beach, CA.

Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Jingming Liu. 2018a. Yuanfudao at SemEval-2018 Task 11: Three-way attention and relational knowledge for commonsense machine comprehension. In *Proceedings of the SemEval*, pages 758–762, New Orleans, LA.

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018b. A co-matching model for multi-choice reading comprehension. In *Proceedings of the ACL*, pages 1–6, Melbourne, Australia.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *Proceedings of the CoNLL*, pages 281–289, Vancouver, Canada.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2018. Dynamic fusion networks for machine reading comprehension. *CoRR*, cs.CL/1711.04964v2.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the ACL*, pages 1040–1050, Vancouver, Canada.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of the ICLR*, Vancouver, Canada.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the RepL4NLP*, pages 15–25, Vancouver, Canada.

Haichao Zhu, Furu Wei, Bing Qin, and Ting Liu. 2018. Hierarchical attention flow for multiple-choice reading comprehension. In *Proceedings of the AAAI*, pages 6077–6084, New Orleans, LA.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE ICCV*, pages 19–27, Santiago, Chile.

Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension

Yichong Xu[1]*, **Xiaodong Liu**[2], **Yelong Shen**[2], **Jingjing Liu**[2] **and Jianfeng Gao**[2]
[1] Carnegie Mellon University
[2] Microsoft Research
`yichongx@cs.cmu.edu`
`{xiaodl, yeshen, jingjl,jfgao}@microsoft.com`

Abstract

We propose a multi-task learning framework to learn a joint Machine Reading Comprehension (MRC) model that can be applied to a wide range of MRC tasks in different domains. Inspired by recent ideas of data selection in machine translation, we develop a novel sample re-weighting scheme to assign sample-specific weights to the loss. Empirical study shows that our approach can be applied to many existing MRC models. Combined with contextual representations from pre-trained language models (such as ELMo), we achieve new state-of-the-art results on a set of MRC benchmark datasets. We release our code at `https://github.com/xycforgithub/MultiTask-MRC`.

1 Introduction

Machine Reading Comprehension (MRC) has gained growing interest in the research community (Rajpurkar et al., 2016; Yu et al., 2018). In an MRC task, the machine reads a text passage and a question, and generates (or selects) an answer based on the passage. This requires the machine to possess strong comprehension, inference and reasoning capabilities. Over the past few years, there has been much progress in building end-to-end neural network models (Seo et al., 2016) for MRC. However, most public MRC datasets (e.g., SQuAD, MS MARCO, TriviaQA) are typically small (less than 100K) compared to the model size (such as SAN (Liu et al., 2018c,b) with around 10M parameters). To prevent over-fitting, recently there have been some studies on using pre-trained word embeddings (Pennington et al., 2014) and contextual embeddings in the MRC model training, as well as back-translation approaches (Yu et al., 2018) for data augmentation.

Multi-task learning (Caruana, 1997) is a widely studied area in machine learning, aiming at better model generalization by combining training datasets from multiple tasks. In this work, we explore a multi-task learning (MTL) framework to enable the training of one universal model across different MRC tasks for better generalization. Intuitively, this multi-task MRC model can be viewed as an implicit data augmentation technique, which can improve generalization on the target task by leveraging training data from auxiliary tasks.

We observe that merely adding more tasks cannot provide much improvement on the target task. Thus, we propose two MTL training algorithms to improve the performance. The first method simply adopts a sampling scheme, which randomly selects training data from the auxiliary tasks controlled by a ratio hyperparameter; The second algorithm incorporates recent ideas of data selection in machine translation (van der Wees et al., 2017). It learns the sample weights from the auxiliary tasks automatically through language models.

Prior to this work, many studies have used upstream datasets to augment the performance of MRC models, including word embedding (Pennington et al., 2014), language models (ELMo) (Peters et al., 2018) and machine translation (Yu et al., 2018). These methods aim to obtain a robust semantic encoding of both passages and questions. Our MTL method is orthogonal to these methods: rather than enriching semantic embedding with external knowledge, we leverage existing MRC datasets across different domains, which help make the whole comprehension process more robust and universal. Our experiments show that MTL can bring further performance boost when combined with contextual representations from pre-trained language models, e.g., ELMo (Peters et al., 2018).

---

* Most of this work was performed when the author was interning at Microsoft.

Proceedings of NAACL-HLT 2019, pages 2644–2655
Minneapolis, Minnesota, June 2 - June 7, 2019.

To the best of our knowledge, this is the first work that systematically explores multi-task learning for MRC. In previous methods that use language models and word embedding, the external embedding/language models are pre-trained separately and remain fixed during the training of the MRC model. Our model, on the other hand, can be trained with more flexibility on various MRC tasks. MTL is also faster and easier to train than embedding/LM methods: our approach requires no pre-trained models, whereas back translation and ELMo both rely on large models that would need days to train on multiple GPUs (Jozefowicz et al., 2016; Peters et al., 2018).

We validate our MTL framework with two state-of-the-art models on four datasets from different domains. Experiments show that our methods lead to a significant performance gain over single-task baselines on SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017) and Who-Did-What (Onishi et al., 2016), while achieving state-of-the-art performance on the latter two. For example, on NewsQA (Trischler et al., 2017), our model surpassed human performance by **13.4** (46.5 vs 59.9) and **3.2** (72.6 vs 69.4) absolute points in terms of exact match and F1.

The contribution of this work is three-fold. First, we apply multi-task learning to the MRC task, which brings significant improvements over single-task baselines. Second, the performance gain from MTL can be easily combined with existing methods to obtain further performance gain. Third, the proposed sampling and re-weighting scheme can further improve the multi-task learning performance.

2 Related Work

Studies in machine reading comprehension mostly focus on architecture design of neural networks, such as bidirectional attention (Seo et al., 2016), dynamic reasoning (Xu et al., 2017), and parallelization (Yu et al., 2018). Some recent work has explored transfer learning that leverages outdomain data to learn MRC models when no training data is available for the target domain (Golub et al., 2017). In this work, we explore multi-task learning to make use of the data from other domains, while we still have access to target domain training data.

Multi-task learning (Caruana, 1997) has been widely used in machine learning to improve generalization using data from multiple tasks. For natural language processing, MTL has been successfully applied to low-level parsing tasks (Collobert et al., 2011), sequence-to-sequence learning (Luong et al., 2015), and web search (Liu et al., 2015). More recently, (McCann et al., 2018) proposes to cast all tasks from parsing to translation as a QA problem and use a single network to solve all of them. However, their results show that multi-task learning hurts the performance of most tasks when tackling them together. Differently, we focus on applying MTL to the MRC task and show significant improvement over single-task baselines.

Our sample re-weighting scheme bears some resemblance to previous MTL techniques that assign weights to tasks (Kendall et al., 2018). However, our method gives a more granular score for each sample and provides better performance for multi-task learning MRC.

3 Model Architecture

We call our model Multi-Task-SAN (MT-SAN), which is a variation of SAN (Liu et al., 2018c) model with two main differences: i) we add a highway network layer after the embedding layer, the encoding layer and the attention layer; ii) we use exponential moving average (Seo et al., 2016) during evaluation. The SAN architecture and our modifications are briefly described below and in Section 5.2, and detailed description can be found in (Liu et al., 2018c).

3.1 Input Format

For most tasks we consider, our MRC model takes a triplet (Q, P, A) as input, where $Q = (q_1, ..., q_m), P = (p_1, ..., p_n)$ are the word index representations of a question and a passage, respectively , and $A = (a_{\text{begin}}, a_{\text{end}})$ is the index of the answer span. The goal is to predict A given (Q, P).

3.2 Lexicon Encoding Layer

We map the word indices of P and Q into their 300-dim Glove vectors (Pennington et al., 2014). We also use the following additional information for embedding words: i) 16-dim part-of-speech (POS) tagging embedding; ii) 8-dim named-entity-recognition (NER) embedding; iii) 3-dim exact match embedding: $f_{\text{exact\_match}}(p_i) = \mathbb{I}(p_i \in Q)$, where matching is determined based on the original word, lower case, and lemma form, respectively; iv) Question enhanced passage word embeddings: $f_{\text{align}}(p_i) = \sum_j \gamma_{i,j} h(\text{GloVe}(q_j))$, where

$$\gamma_{i,j} = \frac{\exp(h(\text{GloVe}(p_j)),h(\text{GloVe}(q_i)))}{\sum_{j'}\exp(h(\text{GloVe}(p_{j'})),h(\text{GloVe}(q_i)))} \quad (1)$$

is the similarity between word p_j and q_i, and $g(\cdot)$ is a 300-dim single layer neural net with Rectified Linear Unit (ReLU) $g(x) = \text{ReLU}(W_1 x)$; v) Passage-enhanced question word embeddings: the same as iv) but computed in the reverse direction. To reduce the dimension of the input to the next layer, the 624-dim input vectors of passages and questions are passed through a ReLu layer to reduce their dimensions to 125.

After the ReLU network, we pass the 125-dim vectors through a highway network (Srivastava et al., 2015), to adapt to the multi-task setting: $g_i = \text{sigmoid}(W_2 p_i^t), p_i^t = \text{ReLU}(W_3 p_i^t) \odot g_i + g_i \odot p_i^t$, where p_i^t is the vector after ReLU transformation. Intuitively, the highway network here provides a neuron-wise weighting, which can potentially handle the large variation in data introduced by multiple datasets.

3.3 Contextual Encoding Layer

Both the passage and question encodings go through a 2-layer Bidirectional Long-Short Term Memory (BiLSTM, Hochreiter and Schmidhuber, 1997) network in this layer. We append a 600-dim CoVe vector (McCann et al., 2017) to the output of the lexicon encoding layer as input to the contextual encoders. For the experiments with ELMo, we also append a 1024-dim ELMo vector. Similar to the lexicon encoding layer, the outputs of both layers are passed through a highway network for multi-tasking. Then we concatenate the output of the two layers to obtain $H^q \in \mathbb{R}^{2d\times m}$ for the question and $H^p = \mathbb{R}^{2d\times n}$ the passage, where d is the dimension of the BiLSTM.

3.4 Memory/Cross Attention Layer

We fuse H^p and H^q through cross attention and generate a working memory in this layer. We adopt the attention function from (Vaswani et al., 2017) and compute the attention matrix as $C = \text{dropout}\left(f_{\text{attention}}(\hat{H}^q, \hat{H}^p)\right) \in \mathbb{R}^{m\times n}$. We then use C to compute a question-aware passage representation as $U^p = \text{concat}(H^p, H^q C)$. Since a passage usually includes several hundred tokens, we use the method of (Lin et al., 2017) to apply self attention to the representations of passage to rearrange its information: $\hat{U}^p = U^p \text{drop}_{\text{diag}}(f_{\text{attention}}(U^p, U^p))$, where $\text{drop}_{\text{diag}}$ means that we only drop diagonal elements on the similarity matrix (i.e., attention with itself). Then, we concatenate U^p and $\hat{U}^p$ and pass them through a BiLSTM: $M = \text{BiLSTM}([U^p]; \hat{U}^p])$. Finally, output of the BiLSTM (after concatenating two directions) goes through a highway layer to produce the memory.

3.5 Answer Module

The base answer module is the same as SAN, which computes a distribution over spans in the passage. Firstly, we compute an initial state s_0 by self attention on H^q: $s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H_j^q)}{\sum_{j'}\exp w_4 H_{j'}^q} \cdot H_j^q\right)$. The final answer is computed through T time steps. At step $t \in \{1, ..., T-1\}$, we compute the new state using a Gated Recurrent Unit (GRU, Cho et al., 2014) $s_t = \text{GRU}(s_{t-1}, x_t)$, where x_t is computed by attention between M and s_{t-1}: $x_t = \sum_j \beta_j M_j, \beta_j = \text{softmax}(s_{t-1} W_5 M)$. Then each step produces a prediction of the start and end of answer spans through a bilinear function: $P_t^{\text{begin}} = \text{softmax}(s_t W_6 M)$, $P_t^{\text{end}} = \text{softmax}(s_t W_7 M)$. The final prediction is the average of each time step: $P^{\text{begin}} = \frac{1}{T}\sum_t P_t^{\text{begin}}, P^{\text{end}} = \frac{1}{T}\sum_t P_t^{\text{end}}$. We randomly apply dropout on the step level in each time step during training, as done in (Liu et al., 2018c). During training, the objective is the log-likelihood of the ground truth: $l(Q, P, A) = \log P^{\text{begin}}(a_{\text{begin}}) + \log P^{\text{end}}(a_{\text{end}})$.

4 Multi-task Learning Algorithms

We describe our MTL training algorithms in this section. We start with a very simple and straightforward algorithm that samples one task and one mini-batch from that task at each iteration. To improve the performance of MTL on a target dataset, we propose two methods to re-weight samples according to their importance. The first proposed method directly lowers the probability of sampling from a particular auxiliary task; however, this probability has to be chosen using grid search. We then propose another method that avoids such search by using a language model.

Suppose we have K different tasks, the simplest version of our MTL training procedure is shown in Algorithm 1. In each epoch, we take all the mini-batches from all datasets and shuffle them for

Algorithm 1 Multi-task Learning of MRC

```
Input: k different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$, max_epoch
Initialize the model $\mathcal{M}$
for epoch= $1, 2, ...,$ max_epoch do
    Divide each dataset $\mathcal{D}_k$ into $N_k$ mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}, 1 \leq k \leq K$
    Put all mini-batches together and randomly shuffle the order of them, to obtain a sequence $B = (b_1, ..., b_L)$, where $L = \sum_k N_k$
    for each mini-batch $b \in B$ do
        Perform gradient update on $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A) \in b} l(Q, P, A)$
    end for
    Evaluate development set performance
end for
Output: Model with best evaluation performance
```

model training, and the same set of parameters is used for all tasks. Perhaps surprisingly, as we will show in the experiment results, this simple baseline method can already lead to a considerable improvement over the single-task baselines.

4.1 Mixture Ratio

One observation is that the performance of our model using Algorithm 1 starts to deteriorate as we add more and more data from other tasks into our training pool. We hypothesize that the external data will inevitably bias the model towards auxiliary tasks instead of the target task.

To avoid such adverse effect, we introduce a mixture ratio parameter during training. The training algorithm with the mixture ratio is presented in Algorithm 2, with $\mathcal{D}_1$ being the target dataset. In each epoch, we use all mini-batches from $\mathcal{D}_1$, while only a ratio α of mini-batches from external datasets are used to train the model. In our experiment, we use hyperparameter search to find the best α for each dataset combination. This method resembles previous methods in multi-task learning to weight losses differently (e.g., Kendall et al., 2018), and is very easy to implement. In our experiments, we use Algorithm 2 to train our network when we only use 2 datasets for MTL.

4.2 Sample Re-Weighting

The mixture ratio (Algorithm 2) dramatically improves the performance of our system. However, it requires to find an ideal ratio by hyperparameter search which is time-consuming. Furthermore, the ratio gives the same weight to every auxiliary data, but the relevance of every data point to the target task can vary greatly.

Algorithm 2 Multi-task Learning of MRC with mixture ratio, targeting $\mathcal{D}_1$

```
Input: K different datasets $\mathcal{D}_1, ..., \mathcal{D}_K$, max_epoch, mixture ratio $\alpha$
Initialize the model $\mathcal{M}$
for epoch= $1, 2, ...,$ max_epoch do
    Divide each dataset $\mathcal{D}_k$ into $N_k$ mini-batches $\mathcal{D}_k = \{b_1^k, ..., b_{N_k}^k\}, 1 \leq k \leq K$
    $S \leftarrow \{b_1^1, ..., b_{N_1}^1\}$
    Randomly pick $\lfloor \alpha N_1 \rfloor$ mini-batches from $\bigcup_{k=2}^K \mathcal{D}_k$ and add to $S$
    Assign mini-batches in $S$ in a random order to obtain a sequence $B = (b_1, ..., b_L)$, where $L = N_1 + \lfloor \alpha N_1 \rfloor$
    for each mini-batch $b \in B$ do
        Perform gradient update on $\mathcal{M}$ with loss $l(b) = \sum_{(Q,P,A) \in b} l(Q, P, A)$
    end for
    Evaluate development set performance
end for
Output: Model with best evaluation performance
```

We develop a novel re-weighting method to resolve these problems, using ideas inspired by data selection in machine translation (Axelrod et al., 2011; van der Wees et al., 2017). We use (Q^k, P^k, A^k) to represent a data point from the k-th task for $1 \leq k \leq K$, with $k = 1$ being the target task. Since the passage styles are hard to evaluate, we only evaluate data points based on Q^k and A^k. Note that only data from auxiliary task ($2 \leq k \leq K$) is re-weighted; target task data always have weight 1.

Our scores consist of two parts, one for questions and one for answers. For questions, we create language models (detailed in Section 5.2) using questions from each task, which we represent as LM_k for the k-th task. For each question Q^k from auxiliary tasks, we compute a cross-entropy score:

$$H_{C,Q}(Q^k) = -\frac{1}{m} \sum_{w \in Q^k} \log(LM_C(w)), \quad (2)$$

where $C \in \{1, k\}$ is the target or auxiliary task, m is the length of question Q^k, and w iterates over all words in Q^k.

It is hard to build language models for answers since they are typically very short (e.g., answers

Dataset	SQuAD(v1)	NewsQA	MS MARCO(v1)	WDW
# Training Questions	87,599	92,549	78,905	127,786
Text Domain	Wikipedia	CNN News	Web Search	Gigaword Corpus
Avg. Document Tokens	130	638	71	365
Answer type	Text span	Text span	Natural sentence	Cloze
Avg. Answer Tokens	3.5	4.5	16.4	N/A

Table 1: Statistics of the datasets. Some numbers come from (Sugawara et al., 2017).

on SQuAD includes only one or two words in most cases). We instead just use the length of answers as a signal for scores. Let l_a^k be the length of A^k, the cross-entropy answer score is defined as:

$$H_{C,A}(A^k) = -\log \text{freq}_C(l_a^k), \tag{3}$$

where freq_C is the frequency of answer lengths in task $C \in \{1, k\}$.

The cross entropy scores are then normalized over all samples in task C to create a comparable metric across all auxiliary tasks:

$$H'_{C,Q}(Q^k) = \frac{H_{C,Q}(Q^k) - \min(H_{C,Q})}{\max(H_{C,Q}) - \min(H_{C,Q})} \tag{4}$$

$$H'_{C,A}(A^k) = \frac{H_{C,A}(A^k) - \min(H_{C,A})}{\max(H_{C,A}) - \min(H_{C,A})} \tag{5}$$

for $C \in \{1, 2, ..., K\}$. For $C \in \{2, ..., K\}$, the maximum and minimum are taken over all samples in task k. For $C = 1$ (target task), they are taken over all available samples.

Intuitively, $H'_{C,Q}$ and $H'_{C,A}$ represents the similarity of text Q, A to task C; a low $H'_{C,Q}$ (resp. $H'_{C,A}$) means that Q^k (resp. A^k) is easy to predict and similar to C, and vice versa. We would like samples that are most similar from data in the target domain (low H'_1), and most different (informative) from data in the auxiliary task (high H'_k). We thus compute the following cross-entropy difference for each external data:

$$\begin{aligned}\text{CED}(Q^k, A^k) =&(H'_{1,Q}(Q^k) - H'_{k,Q}(Q^k)) + \\ &(H'_{1,A}(A^k) - H'_{k,A}(A^k))\end{aligned} \tag{6}$$

for $k \in \{2, ..., K\}$. Note that a low CED score indicates high importance. Finally, we transform the scores to weights by taking negative, and normalize between $[0, 1]$:

$$\text{CED}'(Q^k, A^k) = 1 - \frac{\text{CED}(Q^k, A^k) - \min(\text{CED})}{\max(\text{CED}) - \min(\text{CED})}. \tag{7}$$

Here the maximum and minimum are taken over all available samples and task. Our training algorithm is the same as Algorithm 1, but for mini-batch b we instead use the loss

$$l(b) = \sum_{(P,Q,A) \in b} \text{CED}'(Q, A) l(P, Q, A) \tag{8}$$

in step 6. We define $\text{CED}'(Q^1, A^1) \equiv 1$ for all target samples (P^1, Q^1, A^1).

5 Experiments

Our experiments are designed to answer the following questions on multi-task learning for MRC:
1. Can we improve the performance of existing MRC systems using multi-task learning?
2. How does multi-task learning affect the performance if we combine it with other external data?
3. How does the learning algorithm change the performance of multi-task MRC?
4. How does our method compare with existing MTL methods?

We first present our experiment details and results for MT-SAN. Then, we provide a comprehensive study on the effectiveness of various MTL algorithms in Section 5.4. At last, we provide some additional results on combining MTL with DrQA (Chen et al., 2017) to show the flexibility of our approach [1].

5.1 Datasets

We conducted experiments on SQuAD (Rajpurkar et al., 2016), NewsQA(Trischler et al., 2017), MS MARCO (v1, Nguyen et al.,2016) and WDW (Onishi et al., 2016). Dataset statistics is shown in Table 1. Although similar in size, these datasets are quite different in domains, lengths of text, and types of task. In the following experiments, we will validate whether including external datasets as additional input information (e.g., pre-trained language model on these datasets) helps boost the performance of MRC systems.

---

[1] We include the results in the appendix due to space limitations.

Model	Dev Set Performance
Single Model without Language Models	EM,F1
BiDAF (Seo et al., 2016)	67.7, 77.3
SAN (Liu et al., 2018c)	76.24, 84.06
MT-SAN on SQuAD (single task, ours)	76.84, 84.54
MT-SAN on SQuAD+NewsQA(ours)	78.60, 85.87
MT-SAN on SQuAD+MARCO(ours)	77.79, 85.23
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**78.72, 86.10**
Single Model with ELMo	
SLQA+ (Wang et al., 2018a)	80.0, 87.0
MT-SAN on SQuAD (single task, ours)	80.04, 86.54
MT-SAN on SQuAD+NewsQA(ours)	81.36, 87.71
MT-SAN on SQuAD+MARCO(ours)	80.37, 87.17
MT-SAN on SQuAD+NewsQA+MARCO(ours)	**81.58, 88.19**
BERT (Devlin et al., 2018)	**84.2, 91.1**
Human Performance (test set)	82.30, 91.22

Table 2: Performance of our method to train SAN in multi-task setting, competing published results, leaderboard results and human performance, on SQuAD dataset (single model). Note that BERT uses a much larger language model, and is not directly comparable with our results. We expect our test performance is roughly similar or a bit higher than our dev performance, as is the case with other competing models.

5.2 Experiment Details

We mostly focus on span-based datasets for MT-SAN, namely SQuAD, NewsQA, and MS MARCO. We convert MS MARCO into an answer-span dataset to be consistent with SQuAD and NewsQA, following (Liu et al., 2018c). For each question, we search for the best span using ROUGE-L score in all passage texts and use the span to train our model. We exclude questions with maximal ROUGE-L score less than 0.5 during training. For evaluation, we use our model to find a span in all passages. The prediction score is multiplied with the ranking score, trained following Liu et al. (2018a)'s method to determine the final answer.

We train our networks using algorithms in Section 4, using SQuAD as the target task. For experiments with two datasets, we use Algorithm 2; for experiments with three datasets we find the re-weighting mechanism in Section 4.2 to have a better performance (a detailed comparison will be presented in Section 5.4).

For generating sample weights, we build a LSTM language model on questions following the implementation of Merity et al. (2017) with the same hyperparameters. We only keep the 10,000 most frequent words, and replace the other words with a special out-of-vocabulary token.

Parameters of MT-SAN are mostly the same as in the original paper (Liu et al., 2018c). We utilize spaCy[2] to tokenize the text and generate part-of-speech and named entity labels. We use a 2-layer BiLSTM with 125 hidden units as the BiLSTM throughout the model. During training, we drop the activation of each neuron with 0.3 probability. For optimization, we use Adamax (Kingma and Ba, 2014) with a batch size of 32 and a learning rate of 0.002. For prediction, we compute an exponential moving average (EMA, Seo et al. 2016) of model parameters with a decay rate of 0.995 and use it to compute the model performance. For experiments with ELMo, we use the model implemented by AllenNLP [3]. We truncate passage to contain at most 1000 tokens during training and eliminate those data with answers located after the 1000th token. The training converges in around 50 epochs for models without ELMo (similar to the single-task SAN); For models with ELMo, the convergence is much faster (around 30 epochs).

5.3 Performance of MT-SAN

In the following sub-sections, we report our results on SQuAD and MARCO development sets,

---

[2] https://spacy.io

[3] https://allennlp.org/

as well as on the development and test sets of NewsQA [4]. All results are single-model performance unless otherwise noted.

The multi-task learning results of SAN on SQuAD are summarized in Table 2. By using MTL on SQuAD and NewsQA, we can improve the exact-match (EM) and F1 score by (2%, 1.5%), respectively, both with and without ELMo. The similar gain indicates that our method is orthogonal to ELMo. Note that our single-model performance is slightly higher than the original SAN, by incorporating EMA and highway networks. By incorporating with multi-task learning, it further improves the performance. The performance gain by adding MARCO is relatively smaller, with 1% in EM and 0.5% in F1. We conjecture that MARCO is less helpful due to its differences in both the question and answer style. For example, questions in MS MARCO are real web search queries, which are short and may have typos or abbreviations; while questions in SQuAD and NewsQA are more formal and well written.

Using 3 datasets altogether provides another marginal improvement. Our model obtains the best results among existing methods that do not use a large language model (e.g., ELMo). Our ELMo version also outperforms any other models which are under the same setting. We note that BERT (Devlin et al., 2018) uses a much larger model than ours(around 20x), and we leave the performance of combining BERT with MTL as interesting future work.

The results of multi-task learning on NewsQA are in Table 3. The performance gain with multi-task learning is even larger on NewsQA, with over 2% in both EM and F1. Experiments with and without ELMo give similar results. What is worth noting is that our approach not only achieves new state-of-art results with a large margin but also surpasses human performance on NewsQA.

Finally we report MT-SAN performance on MS MARCO in Table 4. Multi-tasking on SQuAD and NewsQA provides a similar performance boost in terms of BLEU-1 and ROUGE-L score as in the case of NewsQA and SQuAD. Our method does not achieve very high performance compared to previous work, probably because we do not apply common techniques like yes/no classification or cross-passage ranking (Wang et al., 2018b).

Model	Dev Set	Test Set
Model W/o ELMo	EM,F1	EM, F1
Match-LSTM[1]	34.4, 49.6	34.9, 50.0
FastQA[2]	43.7, 56.1	42.8, 56.1
AMANDA[3]	48.4, 63.3	48.4, 63.7
MT-SAN (Single task)	55.8, 67.9	55.6, 68.0
MT-SAN (S+N)	**57.8, 69.9**	**58.3, 70.7**
Model With ELMo		
MT-SAN (Single task)	57.7, 70.4	57.0, 70.4
MT-SAN (S+N)	**60.1, 72.5**	**59.9, 72.6**
Human Performance	-,-	46.5, 69.4

Table 3: Performance of our method to train SAN in multi-task setting, with published results and human performance on NewsQA dataset. All SAN results are from our models. "S+N" means jointly training on SQuAD and NewsQA References: [1]: implemented by Trischler et al. (2017). [2]:Weissenborn et al.(2017). [3]: Kundu and Ng(2018).

Model	Scores
Single Model W/o ELMo	
FastQAExt[1] (test set)	33.99, 32.09
Reasonet++[2]	38.62, 38.01
V-Net[3]	-, 45.65
SAN[4]	43.85, 46.14
MT-SAN	34.13, 42.65
MT-SAN: SQuAD+MARCO	34.29, 43.47
MT-SAN: 3 datasets	**36.99, 43.64**
Single Model With ELMo	
MT-SAN	34.57, 42.88
MT-SAN: SQuAD+MARCO	37.02, 43.89
MT-SAN: 3 datasets	**37.12, 44.12**
Human Performance (test set)	48.02, 49.72

Table 4: Performance of our method to train SAN in multi-task setting, competing published results and human performance, on MS MARCO dataset. The scores stand for (BLEU-1, ROUGE-L) respectively. All SAN results are our results. "3 dataset" means we train using SQuAD+NewsQA+MARCO. References: [1]: (Weissenborn et al., 2017). [2]: implemented by (Shen et al., 2017). [3]:(Wang et al., 2018b). [4]: (Liu et al., 2018c)

We also test the robustness of our algorithm by performing another set of experiments on SQuAD and WDW. WDW is much more different than the other three datasets (SQuAD, NewsQA, MS MARCO): WDW guarantees that the answer is always a person, whereas the percentage of

---

[4] The official submission for SQuAD v1.1 and MARCO v1.1 are closed, so we report results on the development set. According to their leaderboards, performances on development and test sets are usually similar.

Model	SQuAD	WDW
MT-SAN (Single Task)	76.8, 84.5	77.5
MT-SAN (S+W)	**77.6, 85.1**	**78.5**
SOTA(Yang et al., 2016).	86.2, 92.2	71.7
Human Performance	82.3, 91.2	84

Table 5: Performance of MT-SAN on SQuAD Dev and WDW test set. Accuracy is used to evaluate WDW. "S+W" means jointly training on SQuAD and WDW.

Model	EM, F1	+/-
QANet	73.6, 82.7	0.0, 0.0
QANet + BT	75.1, 83.8	+1.5,+1.1
SAN	76.8, 84.5	0.0, 0.0
MT-SAN	**78.7, 86.0**	**+1.9,+1.5**
SAN + ELMo	80.0, 86.5	+3.2,+2.0
MT-SAN + ELMo	**81.6, 88.2**	**+4.8, +3.7**

Table 6: Comparison of methods to use external data. BT stands for back translation (Yu et al., 2018).

such questions in SQuAD is 12.9%. Moreover, WDW is a cloze dataset, whereas in SQuAD and NewsQA answers are spans in the passage. We use a task-specific answer layer in this experiment and use Algorithm 2; the WDW answer module is the same as in AS Reader (Kadlec et al., 2016), which we describe in the appendix for completeness. Despite these large difference between datasets, our results (Table 5) show that MTL can still provide a moderate performance boost when jointly training on SQuAD (around 0.7%) and WDW (around 1%).

Comparison of methods using external data. As a method of data augmentation, we compare our approach to previous methods for MRC in Table 6. Our model achieves better performance than back translation. We also observe that language models such as ELMo obtain a higher performance gain than multi-task learning, however, combining it with multi-task learning leads to the most significant performance gain. This validates our assumption that multi-task learning is more robust and is different from previous methods such as language modeling.

5.4 Comparison of Different MTL Algorithms

In this section, we provide ablation studies as well as comparisons with other existing algorithms on the MTL strategy. We focus on MT-SAN without ELMo for efficient training.

Model	Performance
SQuAD + MARCO	EM,F1
Simple Combine (Alg. 1)	77.1, 84.6
Loss Uncertainty	77.3, 84.7
Mixture Ratio	77.8, 85.2
Sample Re-weighting	**77.9,85.3**
SQuAD + NewsQA + MARCO	
Simple Combine (Alg. 1)	77.6, 85.2
Loss Uncertainty	78.2, 85.6
Mixture Ratio	78.4, 85.7
Sample Re-weighting	**78.8**, **86.0**

Table 7: Comparison of different MTL strategies on MT-SAN. Performance is on SQuAD. Loss Uncertainty is from Kendall et al. (2018).

Table 7 compares different multi-task learning strategies for MRC. Both the mixture ratio (Sec 4.1) and sample re-weighting (Sec 4.2) improves over the naive baseline of simply combining all the data (Algorithm 1). On SQuAD+MARCO, they provide around 0.6% performance boost in terms of both EM and F1, and around 1% on all 3 datasets. We note that this accounts for around a half of our overall improvement. Although sample re-weighting performs similar as mixture ratio, it significantly reduces the amount of training time as it eliminates the need for a grid searching the best ratio. Kendal et al., (2018) use task uncertainty to weight tasks differently for MTL; our experiments show that this has some positive effect, but does not perform as well as our proposed two techniques. We note that Kendal et al. (as well as other previous MTL methods) optimizes the network to perform well for all the tasks, whereas our method focuses on the target domain which we are interested in, e.g., SQuAD.

Sensitivity of mixture ratio. We also investigate the effect of mixture ratio on the model performance. We plot the EM/F1 score on SQuAD dev set vs. mixture ratio in Figure 1 for MT-SAN when trained on all three datasets. The curve peaks at $\alpha = 0.4$; however if we use $\alpha = 0.2$ or $\alpha = 0.5$, the performance drops by around 0.5%, well behind the performance of sample re-weighting. This shows that the performance of MT-SAN is sensitive to changes in α, making the hyperparameter search even more difficult. Such sensitivity suggests a preference for using our sample re-weighting technique. On the other hand, the ratio

	Samples/Groups	CED$'$	H_Q	H_A
Examples	(NewsQA) Q: Where is the drought hitting? A: Argentina	0.824	0.732	0.951
	(MARCO) Q: thoracic cavity definition A: is the chamber of the human body ... and fascia.	0.265	0.332	0.240
Averages	Samples in NewsQA	0.710	0.593	0.895
	Samples in MARCO	0.587	0.550	0.669
	MARCO Questions that start with “When” or “Who”	0.662	0.605	0.761
	All samples	0.654	0.573	0.791

Table 8: Scores for examples from NewsQA and MS MARCO and average scores for specific groups of samples. CED$'$ is as in (7), while H_Q and H_A are normalized version of question and sample scores. “Sum” are the actual scores we use, and “LM”, “Answer” are scores from language models and answer lengths.

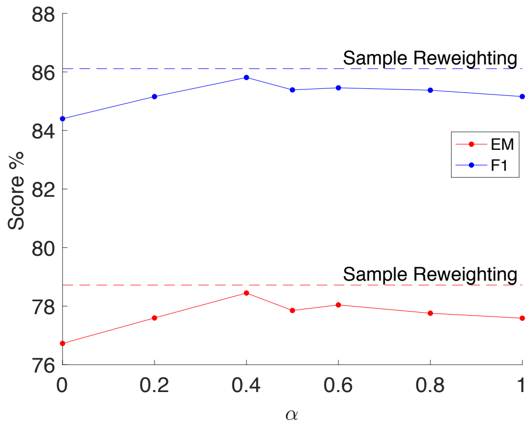


Figure 1: Effect of the mixture ratio on the performance of MT-SAN. Note that $\alpha = 0$ is equivalent to single task learning, and $\alpha = 1$ is equivalent to simple combining.

based approach is pretty straightforward to implement.

Analysis of sample weights. Dataset comparisons in Table 1 and performance in Table 2 suggests that NewsQA share more similarity with SQuAD than MARCO. Therefore, a MTL system should weight NewsQA samples more than MARCO samples for higher performance. We try to verify this in Table 8 by showing examples and statistics of the sample weights. We present the CED$'$ scores, as well as normalized version of question and answer scores (resp. $(H'_{1,Q} - H'_{k,Q})$ and $(H'_{1,A} - H'_{k,A})$ in (6), and then negated and normalized over all samples in NewsQA and MARCO in the same way as in (7)). A high H_Q score indicates high importance of the question, and H_A of the answer; CED$'$ is a summary of the two. We first show one example from NewsQA and one from MARCO. The NewsQA question is a natural question (similar to SQuAD) with a short answer, leading to high scores both in questions and answers. The MARCO question is a phrase, with a very long answer, leading to lower scores. From overall statistics, we also find samples in NewsQA have a higher score than those in MARCO. However, if we look at MARCO questions that start with “when” or “who” (i.e., probability natural questions with short answers), the scores go up dramatically.

6 Conclusion

We proposed a multi-task learning framework to train MRC systems using datasets from different domains and developed two approaches to re-weight the samples for multi-task learning on MRC tasks. Empirical results demonstrated our approaches outperform existing MTL methods and the single-task baselines as well. Interesting future directions include combining with larger language models such as BERT, and MTL with broader tasks such as language inference (Liu et al., 2019) and machine translation.

Acknowledgements

Yichong Xu has been partially supported by DARPA (FA8750-17-2-0130).

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*, pages 355–362. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 835–844.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. 2018. Read+ verify: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1808.05759*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 908–918.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018a. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. 2018b. Stochastic answer networks for squad 2.0. *arXiv preprint arXiv:1809.09194*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018c. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what:

A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yelong Shen, Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2017. An empirical analysis of multiple-turn reasoning strategies in reading comprehension tasks. *arXiv preprint arXiv:1711.03230*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. 2017. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 806–817.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wei Wang, Ming Yan, and Chen Wu. 2018a. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1918–1927.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Dynamic fusion networks for machine reading comprehension. *arXiv preprint arXiv:1711.04964*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

A Answer Module for WDW

We describe the answer module for WDW here for completeness. For WDW we need to choose an answer from a list of candidates; the candidates are people names that have appeared in the passage. We use the same way to summary information in questions as in span-based models: $s_0 \leftarrow \text{Highway}\left(\sum_j \frac{\exp(w_4 H^q_j)}{\sum_{j'} \exp w_4 H^q_{j'}} \cdot H^q_j\right)$. We then compute an attention score via simple dot product: $s = \text{softmax}(s_0^T M)$. The probability of a candidate being the true answer is the aggregation of attention scores for all appearances of the candidate:

$$\Pr(c|Q,P) \propto \sum_{1 \leq i \leq n} s_i \mathbb{I}(p_i \in C)$$

Setup	SQuAD (v1)	SQuAD (v2)	NewsQA	WDW
Single Dataset	69.5,78.8 (paper) 68.6, 77.8 (ours)	61.9, 65.2	51.9, 64.6	**75.8**
MT-DrQA on Sv1+NA	70.2, 79.3	-,-	52.8, 65.8	-
MT-DrQA on Sv1+W	69.2, 78.4	-,-	-,-	75.7
MT-DrQA on Sv1+N+W	**70.2, 79.3**	-,-	**53.1, 65.7**	75.4
MT-DrQA on Sv2+N	-,-	**63.6, 66.7**	52.7, 65.7	-
MT-DrQA on Sv2+W	-,-	63.5, 66.3	-,-	75.4
MT-DrQA on Sv2+N+W	-,-	63.1, 66.3	52.5, 65.6	75.3
SOTA (Single Model)	80.0, 87.0	72.3, 74.8	48.4, 63.7 (test)	71.7 (test)
MT-DrQA Best Performance	**70.2, 79.3**	**63.6, 66.7**	**53.0, 66.2(test)**	**75.4 (test)**
Human Performance (test set)	82.3, 91.2	86.8, 89.5	46.5, 69.4	84

Table 9: Single model performance of our method to train DrQA on multi-task setting, as well as state-of-the-art (SOTA) results and human performance. SQuAD and NewsQA performance are measured by (EM, F1), and WDW by accuracy percentage. All results are on development set unless otherwise noted. Published SOTA results come from (Wang et al., 2018a; Hu et al., 2018; Kundu and Ng, 2018; Yang et al., 2016) respectively.

for each candidate C. Recall that n is the length of passage P, and p_i is the i-th word; therefore $\mathbb{I}(p_i \in C)$ is the indicator function of p_i appears in candidate C. The candidate with the largest probability is chosen as the predicted answer.

B Experiment Results on DrQA

To demonstrate the flexibility of our approach, we also adapt DrQA (Chen et al., 2017) into our MTL framework. We only test DrQA using the basic Algorithm 2, since our goal is mainly to test the MTL framework.

B.1 Model Architecture

Similar to MT-SAN, we add a highway network after the lexicon encoding layer and the contextual encoding layer and use a different answer module for each dataset. We apply MT-DrQA to a broader range of datasets. For span-detection datasets such as SQuAD, we use the same answer module as DrQA. For cloze-style datasets like Who-Did-What, we use the attention-sum reader (Kadlec et al., 2016) as the answer module. For classification tasks required by SQuAD v2.0 (Rajpurkar et al., 2018), we apply a softmax to the last state in the memory layer and use it as the prediction.

B.2 Performance of MT-DrQA

We apply MT-DrQA to SQuAD (v1.1 and v2.0), NewsQA, and WDW. We follow the setup of (Chen et al., 2017) for model architecture and hyperparameter setup. We use Algorithm 1 to train all MT-DrQA models. Different than (Rajpurkar et al., 2018), we do not optimize the evaluation score by changing the threshold to predict unanswerable question for SQuAD v2.0; we just use the argmax prediction. As a result, we expect the gap between dev and test performance to be lower for our model. The results of MT-DrQA are presented in Table 9. The results of combining SQuAD and NewsQA obtain similar performance boost as our SAN experiment, with a performance boost between 1-2% in both EM and F1 for the two datasets. The results of MTL including WDW is different: although adding WDW to SQuAD still brings a marginal performance boost to SQuAD, the performance on WDW drops after we add SQuAD and NewsQA into the training process. We conjecture that this negative transfer phenomenon is probably because of the drastic difference between WDW and SQuAD/NewsQA, both in their domain, answer type, and task type; and DrQA might not be capable of caputuring all these features using just one network. We leave the problem of further preventing such negative transfer to future work.

Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems

Ting-Rui Chiang **Yun-Nung Chen**
National Taiwan University, Taipei, Taiwan
`r07922052@csie.ntu.edu.tw` `y.v.chen@ieee.org`

Abstract

Solving math word problems is a challenging task that requires accurate natural language understanding to bridge natural language texts and math expressions. Motivated by the intuition about how human generates the equations given the problem texts, this paper presents a neural approach to automatically solve math word problems by operating symbols according to their semantic meanings in texts. This paper views the process of generating equations as a bridge between the semantic world and the symbolic world, where the proposed neural math solver is based on an encoder-decoder framework. In the proposed model, the encoder is designed to understand the semantics of problems, and the decoder focuses on tracking semantic meanings of the generated symbols and then deciding which symbol to generate next. The preliminary experiments are conducted in a benchmark dataset Math23K, and our model significantly outperforms both the state-of-the-art single model and the best non-retrieval-based model over about 10% accuracy, demonstrating the effectiveness of bridging the symbolic and semantic worlds from math word problems.[1]

1 Introduction

Automatically solving math word problems has been an interesting research topic and also been viewed as a way of evaluating machines' ability (Mandal and Naskar, 2019). For human, writing down an equation that solves a math word problem requires the ability of reading comprehension, reasoning, and sometimes real world understanding. Specifically, to solve a math word problem, we first need to know the goal of the given problem, then understand the semantic meaning of each numerical number in the problem, perform reasoning based on the comprehension in the previous step, and finally decide what to write in the equation.

Most prior work about solving math word problems relied on hand-crafted features, which required more human knowledge. Because those features are often in the lexical level, it is not clear whether machines really understand the math problems. Also, most prior work evaluated their approaches on relatively small datasets, and the capability of generalization is concerned.

This paper considers the reasoning procedure when writing down the associated equation given a problem. Figure 1 illustrates the problem solving process. The illustration shows that human actually assigns the semantic meaning to each number when manipulating symbols, including operands (numbers) and operators ($+ - \times \div$). Also, we believe that the semantic meaning of operands can help us decide which operator to use. For example, the summation of "*price of one pen*" and "*number of pens Tom bought*" is meaningless; therefore the addition would not be chosen.

Following the observation above, this paper proposes a novel encoder decoder model, where the encoder extracts semantic meanings of numbers in the problem, and the decoder is equipped with a stack that facilitates tracking the semantic meanings of operands. The contributions of this paper are 4-fold:

- This paper is the first work that models semantic meanings of operands and operators for math word problems.
- This paper proposes an end-to-end neural math solver with a novel decoding process that utilizes the stack to generate associated equations.

---

[1] The source code is available at `https://github.com/MiuLab/E2EMathSolver`.

Proceedings of NAACL-HLT 2019, pages 2656–2668
Minneapolis, Minnesota, June 2 - June 7, 2019.

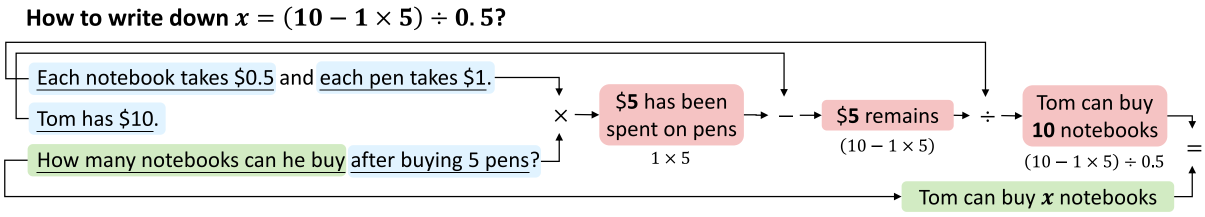


Figure 1: The solving process of the math word problem "*Each notebok takes $0.5 and each pen takes $1. Tom has $10. How many notebook can he buy after buying 5 pens?*" and the associated equation is $x = (10 - 1 \times 5) \div 0.5$. The associated equation is $x = (10 - 1 \times 5) \div 0.5$.

- This paper achieves the state-of-the-art performance on the large benchmark dataset Math23K.
- This paper is capable of providing interpretation and reasoning for the math word problem solving procedure.

2 Related Work

There is a lot of prior work that utilized hand-crafted features, such as POS tags, paths in the dependency trees, keywords, etc., to allow the model to focus on the quantities in the problems (Kushman et al., 2014; Hosseini et al., 2014; Roy et al., 2015; Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Roy et al., 2016; Upadhyay et al., 2016; Upadhyay and Chang, 2017; Roy and Roth, 2018; Wang et al., 2018). Recently, Mehta et al.; Wang et al.; Ling et al. attempted at learning models without predefined features. Following the recent trend, the proposed end-to-end model in this paper does not need any hand-crafted features.

Kushman et al. first extracted templates about math expressions from the training answers, and then trained models to select templates and map quantities in the problem to the slots in the template. Such two-stage approach has been tried and achieved good results (Upadhyay and Chang, 2017). The prior work highly relied on human knowledge, where they parsed problems into equations by choosing the expression tree with the highest score calculated by an operator classifier, working on a hand-crafted "trigger list" containing quantities and noun phrases in the problem, or utilizing features extracted from text spans (Roy et al., 2015, 2016; Koncel-Kedziorski et al., 2015). Shi et al. defined a Dolphin language to connect math word problems and logical forms, and generated rules to parse math word problems. Upadhyay et al. parsed math word problems without explicit equation annotations. Roy and Roth classified math word problems into 4 types and used rules to decide the operators accordingly. Wang et al. trained the parser using reinforcement learning with hand-crafted features. Hosseini et al. modeled the problem text as transition of world states, and the equation is generated as the world states changing. Our work uses a similar intuition, but hand-crafted features are not required and our model can be trained in an end-to-end manner. Some end-to-end approaches have been proposed, such as generating equations directly via a seq2seq model (Wang et al., 2017). Ling et al. tried to generate solutions along with its rationals with a seq2seq-like model for better interpretability.

This paper belongs to the end-to-end category, but different from the previous work; we are the first approach that generates equations with stack actions, which facilitate us to simulate the way how human solves problems. Furthermore, the proposed approach is the first model that is more interpretable and provides reasoning steps without the need of rational annotations.

3 End-to-End Neural Math Solver

Our approach composes of two parts, an encoder and a decoder, where the process of solving math word problems is viewed as transforming multiple text spans from the problems into the target information the problems ask for. In the example shown in Figure 1, all numbers in the problem are attached with the associated semantics. Motivated by the observation, we design an encoder to extract the semantic representation of each number in the problem text. Considering that human usually manipulates those numbers and operators (such as addition, subtraction, etc.) based on their semantics for problem solving, a decoder is designed to construct the equation, where the semantics is aligned with the representations extracted by the encoder. The idea of the proposed model

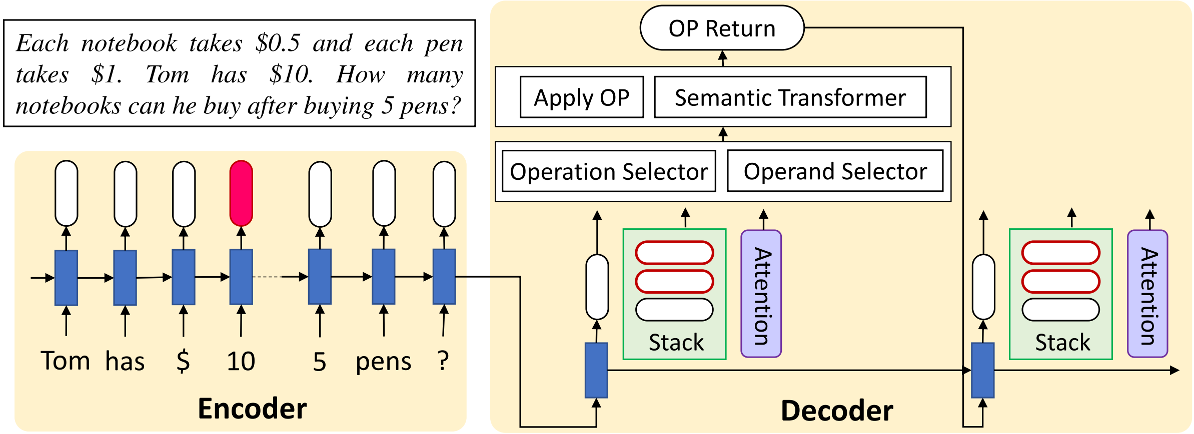


Figure 2: The encoder-decoder model architecture of the proposed neural solver machine.

is to imitate the human reasoning process for solving math word problems. The model architecture is illustrated in Figure 2.

3.1 Encoder

The encoder aims to extract the semantic representation of each constant needed for solving problems. However, the needed constants may come from either the given problem texts or domain knowledge, so we detail these two procedures as follows.

3.1.1 Constant Representation Extraction

For each math word problem, we are given a passage consisting of words $\{w_t^P\}_{t=1}^m$, whose word embeddings are $\{e_t^P\}_{t=1}^m$. The problem text includes some numbers, which we refer as constants. The positions of constants in the problem text are denoted as $\{p_i\}_{i=1}^n$. In order to capture the semantic representation of each constant by considering its contexts, a bidirectional long short-term memory (BLSTM) is adopted as the encoder (Hochreiter and Schmidhuber, 1997):

$$h_t^E, c_t^E = \text{BLSTM}(h_{t-1}^E, c_{t-1}^E, e_t^P), \quad (1)$$

and then for the i-th constant in the problem, its semantic representation e_i^c is modeled by the corresponding BLSTM output vector:

$$e_i^c = h_{p_i}^E. \quad (2)$$

3.1.2 External Constant Leveraging

External constants, including 1 and π, are leveraged, because they are required to solve a math word problem, but not mentioned in the problem text. Due to their absence from the problem text, we cannot extract their semantic meanings by BLSTM in (2). Instead, we model their semantic representation e^π, e^1 as parts of the model parameters. They are randomly initialized and are learned during model training.

3.2 Decoder

The decoder aims at constructing the equation that can solve the given problem. We generate the equation by applying stack actions on a stack to mimic the way how human understands an equation. Human knows the semantic meaning of each term in the equation, even compositing of operands and operators like the term ”$(10-1\times5)$” in Figure 1. Then what operator to apply on a pair operands can be chosen based on their semantic meanings accordingly. Hence we design our model to generate the equation in a postfix manner: a operator is chosen base on the semantic representations of two operands the operator is going to apply to. Note that the operands a operator can apply to can be any results generated previously. That is the reason why we use “stack” as our data structure in order to keep track of the operands a operator is going to apply to. The stack contains both symbolic and semantic representations of operands, denoted as

$$S = [(v_{l_t}^S, e_{l_t}^S), (v_{l_t-1}^S, e_{l_t-1}^S), \cdots, (v_1^S, e_1^S)], \quad (3)$$

where v^S of each pair is the symbolic part, such as $x+1$, while e^S is the semantic representation, which is a vector. The components in the decoder are shown in the right part of Figure 2, each of which is detailed below.

3.3 Decoding State Features

At each decoding step, decisions are made based on features of the current state. At each step, fea-

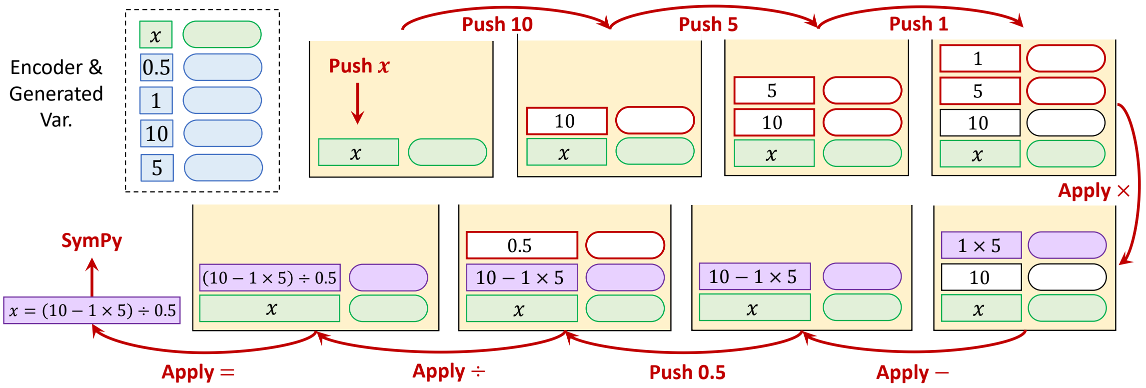


Figure 3: Illustration of the inference process. The purple round blocks denote the transformed semantics, while the green ones are generated by the variable generator.

tures r^{sa} and r^{opd} are extracted to select a stack action (section 3.3.2) and an operand to push (section 3.3.3). Specifically, the features are the gated concatenation of following vectors:

- h_t^D is the output of an LSTM, which encodes the history of applied actions:

  $$h_t^D, c_t^D = \mathrm{LSTM}(h_{t-1}^D, c_{t-1}^D, \mathrm{res}_{t-1}), \quad (4)$$

  where res_{t-1} is the result from the previous stack action similar to the seq2seq model (Sutskever et al., 2014). For example, if the previous stack action o_{t-1} is "push", then res_{t-1} is the semantic representation pushed into the stack. If the previous stack action o_{t-1} is to apply an operator $\diamond$, then res_{t-1} is the semantic representation generated by $f_\diamond$.

- s_t is the stack status. It is crucial because some operators are only applicable to certain combinations of operand semantics, which is similar to the type system in programming languages. For example, operating multiplication is applicable to the combination of "*quantity of an item*" and "*price of an item*", while operating addition is not. Considering that all math operators supported here $(+, -, \times, \div)$ are binary operators, the semantic representations of the stack's top 2 elements at the time $t-1$ are considered:

  $$s_t = [e_{l_t}^S; e_{l_t}^S]. \quad (5)$$

- q_t incorporates problem information in the decision. It is believed that the attention mechanism (Luong et al., 2015) can effectively capture dependency for longer distance. Thus, the attention mechanism over the encoding problem $h_1^E, h_2^E, \cdots$ is adopted:

  $$q_t = \mathrm{Attention}(h_t^D, \{h_i^E\}_{i=1}^m), \quad (6)$$

  where the attention function in this paper is defined as a function with learnable parameters w, W, b:

  $$\mathrm{Attention}(u, \{v_i\}_{i=1}^m) = \sum_{i=1}^m \alpha_i h_i, \quad (7)$$

  $$\alpha_i = \frac{\exp(s_i)}{\sum_{l=1}^m \exp(s_i)}, \quad (8)$$

  $$s_i = w^\intercal \tanh(W^T[u; v_i] + b). \quad (9)$$

In order to model the dynamic features for different decoding steps, features in r_t^{sa} is gated as follows:

$$r_t^{sa} = [g_{t,1}^{sa} \cdot h_t^D; g_{t,2}^{sa} \cdot s_t; g_{t,3}^{sa} \cdot q_t], \quad (10)$$

$$g_t^{sa} = \sigma(W^{sa} \cdot [h_t^D; s_t; q_t]), \quad (11)$$

where σ is a sigmoid function and W^{sa} is a learned gating parameter. r_t^{opd} is defined similarly, but with a different learned gating parameter W^{opd}.

3.3.1 Stack Action Selector

The stack action selector is to select an stack action at each decoding step (section 3.3.2) until the unknowns are solved. The probability of choosing action a at the decoding step t is calculated with a network NN constituted of one hidden layer and ReLU as the activation function:

$$\begin{aligned} P(Y_t \mid \{y_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m) \quad (12) \\ = \mathrm{StackActionSelector}(r_t^{sa}) \\ = \mathrm{softmax}(\mathrm{NN}(r_t^{sa})), \end{aligned}$$

where r_t^{sa} is decoding state features as defined in section 3.3.

3.3.2 Stack Actions

The available stack actions are listed below:

- **Variable generation**: The semantic representation of an unknown variable x is generated dynamically as the first action in the decoding process. Note that this procedure provides the flexibility of solving problems with more than one unknown variables. The decoder module can decide how many unknown variables are required to solve the problem, and the semantic representation of the unknown variable is generated with an attention mechanism:

$$e^x = \mathrm{Attention}(h_t^D, \{h_i^E\}_{i=1}^m). \tag{13}$$

- **Push**: This stack action pushes the operand chosen by the operand selector (section 3.3.3). Both the symbolic representation v_* and semantic representation e_* of the chosen operand would be pushed to the stack S in (3). Then the stack state becomes

$$S = [(v_*^S, e_*^S), (v_{l_t}^S, e_{l_t}^S), \cdots, (v_1^S, e_1^S)]. \tag{14}$$

- **Operator $\diamond$ application** ($\diamond \in \{+, -, \times, \div\}$): One stack action pops two elements from the top of the stack, which contains two pairs, (v_i, e_i) and (v_j, e_j), and then the associated symbolic operator, $v_k = v_i \diamond v_j$, is recorded. Also, a semantic transformation function $f_\diamond$ for that operator is invoked, which generates the semantic representation of v_k by transforming semantic representations of v_i and v_j to $e_k = f_\diamond(e_i, e_j)$. Therefore, after an operator is applied to the stack specified in (3), the stack state becomes

$$\begin{aligned} S =[&(v_{l_t}^S \diamond v_{l_t-1}^S, f_\diamond(e_{l_t}^S, e_{l_t-1}^S)), \\ &(v_{l_t-2}^S, e_{l_t-2}^S), \cdots, (v_1^S, e_1^S)]. \end{aligned} \tag{15}$$

- **Equal application**: When the equal application is chosen, it implies that an equation is completed. This stack action pops 2 tuples from the stack, $(v_i, e_i), (v_j, e_j)$, and then $v_i = v_j$ is recorded. If one of them is an unknown variable, the problem is solved. Therefore, after an OP is applied to the stack specified in (3), the stack state becomes

$$S = [(v_{l_t-2}^S, e_{l_t-2}^S), \cdots, (v_1^S, e_1^S)]. \tag{16}$$

3.3.3 Operand Selector

When the stack action selector has decided to push an operand, the operand selector aims at choosing which operand to push. The operand candidates e include constants provided in the problem text whose semantic representations are $e_1^c, e_2^c, \cdots, e_n^c$, unknown variable whose semantic representation is e^x, and two external constants 1 and π whose semantic representations are e^1, e^π:

$$e = [e_1^c, e_2^c, \cdots, e_n^c, e^1, e^\pi, e^x]. \tag{17}$$

An operand has both symbolic and semantic representations, but the selection focuses on its semantic meaning; this procedure is the same as what human does when solving math word problems.

Inspired by addressing mechanisms of neural Turing machine (NTM) (Graves et al., 2014), the probability of choosing the i-th operand candidate is the attention weights of r_t over the semantic representations of the operand candidates as in (8):

$$\begin{aligned} &P(Z_t \mid \{y_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m) \qquad (18) \\ &= \mathrm{OperandSelector}(r_t^{opd}) \\ &= \mathrm{AttentionWeight}(r_t^{opd}, \{e_i\}_{i=1}^m \cup \{e^1, e^\pi, e^x\}), \end{aligned}$$

and r_t^{opd} is defined in section 3.3.

3.3.4 Semantic Transformer

A semantic transformer is proposed to generate the semantic representation of a new symbol resulted from applying an operator, which provides the capability of interpretation and reasoning for the target task. The semantic transformer for an operator $\diamond \in \{+, -, \times, \div\}$ transforms semantic representations of two operands e_1, e_2 into

$$f_\diamond(e_1, e_2) = \tanh(U_\diamond \mathrm{ReLU}(W_\diamond[e_1; e_2] + b_\diamond) + c_\diamond), \tag{19}$$

where $W_\diamond, U_\diamond, b_\diamond, c_\diamond$ are model parameters. Semantic transformers for different operators have different parameters in order to model different transformations.

3.4 Training

Both stack action selection and operand selection can be trained in a fully supervised way by giving problems and associated ground truth equations. Because our model generates the equation with stack actions, the equation is first transformed into its postfix representation. Let the postfix representation of the target equation be $y_1, \cdots y_t, \cdots, y_T$, where y_t can be either an operator $(+, -, \times, \div, =)$ or a target operand. Then for each time step t, the loss can be computed as

$$L(y_t) = \begin{cases} L_1(\text{push\_op}) + L_2(y_t) & y_t \text{ is an operand} \\ L_1(y_t) & \text{otherwise} \end{cases},$$

where L_1 is the stack action selection loss and L_2 is the operand selection loss defined as

$$L_1(y_t) = -\log P(Y_t = y_t \mid \{o_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^{m}),$$
$$L_2(y_t) = -\log P(Z_t = y_t \mid r_t).$$

The objective of our training process is to minimize the total loss for the whole equation, $\sum_{t=1}^{T} L(y_t)$.

3.5 Inference

When performing inference, at each time step t, the stack action with the highest probability $P(Y_t|\{\tilde{y}_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^{m})$ is chosen. If the chosen stack action is "*push*", the operand with the highest probability $P(Z_t|\{\tilde{Y}_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^{m})$ is chosen. When the stack has less than 2 elements, the probability of applying operator $+, -, \times, \div, =$ would be masked out to prevent illegal stack actions, so all generated equations must be legal math expressions. The decoder decodes until the unknown variable can be solved. After the equations are generated, a Python package SymPy (Meurer et al., 2017) is used to solve the unknown variable. The inference procedure example is illustrated in Figure 3. The detailed algorithm can be found in Algorithm 1.

Algorithm 1 Training and Inference

```
function SOLVEPROBLEM(problem_text)
    $v \leftarrow$ ExtractConstants(problem_text)
                        ▷ $v$ is a list of constants in the problem.
    $h^E, h_0^D, c_0^D, E \leftarrow$ Encoder(problem_text)
    $S \leftarrow$ Stack()
    ret, loss, t, equations $\leftarrow$ padding, 0, 1, {}
    while not solvable(equations) do
        $h_t^D \leftarrow$ LSTM($h_{t-1}^D, c_{t-1}$, ret)
        $s_t \leftarrow S$.get_top2()
        $h^E \leftarrow$ Attention($h_{t-1}^D, h^E$)
        $r_t \leftarrow [h_t^D, s_t, h^E]$
        $p_{sa} \leftarrow$ StackActionSelector($r_t$)
        $p_{opd} \leftarrow$ OperandSelector($r_t$)
        if training then
            ▷ Target equation $y$ is available when training.
            $Y_t \leftarrow y_t$
            if $y_t$ is operand then
                loss $\leftarrow$ loss + $L_1$(push) + $L_2(y_t)$
            else
                loss $\leftarrow$ loss + $L_1(y_t)$
            end if
        else
            $Y_t \leftarrow$ StackActionSelector($r_t^{sa}$)
            if $Y_t$ = push then
                $Z_t \leftarrow$ OperandSelector($r_t^{opd}$)
            end if
        end if
        if $Y_t$ = gen_var then
            $e^x \leftarrow$ Attention($h_t^D, h^E$)
            ret $\leftarrow e^x$
        else if $Y_t$ = push then
            $S$.push($v_{Z_t}, e_{Z_t}$)
            ret $\leftarrow e_{Z_t}$
        else if $Y_t \in \{+, -, \times, \div\}$ then
            $(v_a, e_a), (v_b, e_b) = S$.pop(), $S$.pop()
            $S$.push($v_a Y_t v_b, f_{Y_t}(e_a, e_b)$)
            ret $\leftarrow f_{Y_t}(e_a, e_b)$
        else if $Y_t$ = equal then
            $(v_a, e_a), (v_b, e_b) = S$.pop(), $S$.pop()
            equations = equations $\cup$ "$v_a = v_b$"
            ret $\leftarrow S$.top()
        end if
    end while
    return solve(equations)
end function
```

4 Experiments

To evaluate the performance of the proposed model, we conduct the experiments on the benchmark dataset and analyze the learned semantics.

4.1 Settings

The experiments are benchmarked on the dataset Math23k (Wang et al., 2017), which contains 23,162 math problems with annotated equations. Each problem can be solved by a single-unknown-variable equation and only uses operators $+, -, \times, \div$. Also, except π and 1, quantities in the equation can be found in the problem text. There are also other large scale datasets like Dolphin18K (Shi et al., 2015) and AQuA (Ling et al., 2017), containing 18,460 and 100,000 math word problems respectively. The reasons about not evaluating on these two datasets are 1) Dolphin18k contains some unlabeled math word problems and some incorrect labels, and 2) AQuA contains rational for solving the problems, but the equations in the rational are not formal (e.g. mixed with texts, using x to represent $\times$, etc.) and inconsistent. Therefore, the following experiments are performed and analyzed using Math23K, the only large scaled, good-quality dataset.

4.2 Results

The results are shown in Table 1. The retrieval-based methods compare problems in test data with problems in training data, and choose the most

Model		Accuracy
Retrieval	Jaccard	47.2%
	Cosine	23.8%
Classification	BLSTM	57.9%
	Self-Attention	56.8%
Generation	Seq2Seq w/ SNI	58.1%
	Proposed Word-Based	**65.3%**
	Proposed Char-Based	**65.8%**
Hybrid	Retrieval + Seq2Seq	64.7%

Table 1: 5-fold cross validation results on Math23K.

Model	Accuracy
Char-Based	65.8%
Word-Based	65.3%
Word-Based - Gate	64.1%
Word-Based - Gate - Attention	62.5%
Word-Based - Gate - Attention - Stack	60.1%
Word-Based - Semantic Transformer	64.1%
Word-Based - Semantic Representation	61.7%

Table 2: 5-fold cross validation results of ablation tests.

similar one's template to solve the problem (Kushman et al., 2014; Upadhyay and Chang, 2017). The classification-based models choose equation templates by a classifier trained on the training data. Their performance are reported in Robaidek et al.. The seq2seq and hybrid models are from Wang et al., where the former directly maps natural language into symbols in equations, and the latter one ensembles prediction from a seq2seq model and a retrieval-based model. The ensemble is the previous state-of-the-art results of Math23K.

Our proposed end-to-end model belongs to the generation category, and the single model performance achieved by our proposed model is new state-of-the-art ($> 65\%$) and even better than the hybrid model result (64.7%). In addition, we are the first to report character-based performance on this dataset, and the character-based results are slightly better than the word-based ones. Among the single model performance, our models obtain about more than 7% accuracy improvement compared to the previous best one (Wang et al., 2017). The performance of our character-based model also shows that our model is capable of learning the relatively accurate semantic representations without word boundaries and achieves better performance.

4.3 Ablation Test

To better understand the performance contributed by each proposed component, we perform a series of ablation tests by removing components one by one and then checking the performance by 5-fold cross validation. Table 2 shows the ablation results.

Char-Based v.s. Word-Based As reported above, using word-based model instead of character-based model only causes 0.5% performance drop. To fairly compare with prior word-based models, the following ablation tests are performed on the word-based approach.

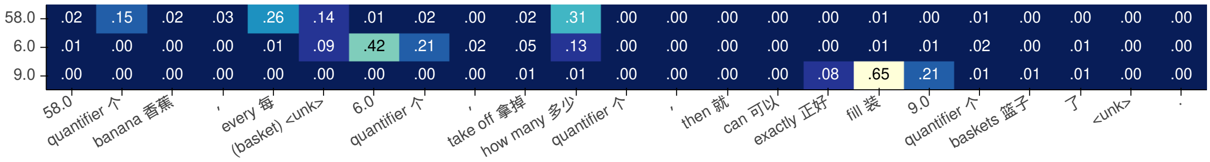
Word-Based - Gate It uses r_t instead of r_t^{sa} and r_t^{opr} as the input of both $\mathrm{StackActionSelector}$ and $\mathrm{OperandSelector}$.

Word-Based - Gate - Attention Considering that the prior generation-based model (seq2seq) did not use any attention mechanism, we compare the models with and without the attention mechanism. Removing attention means excluding q_{t-1} in (11), so the input of both operator and operand selector becomes $r_t = [h_t^D; s_t]$. The result implies that our model is not better than previous models solely because of the attention.

Word-Based - Gate - Attention - Stack To check the effectiveness of the stack status (s_t in (11)), the experiments of removing the stack status from the input of both operator and operand selectors ($r_t = h_t^D$) are conducted. The results well justify our idea of choosing operators based on semantic meanings of operands.

Word-Based - Semantic Transformer To validate the effectiveness of the idea that views an operator as a semantic transformer, we modify the semantic transformer function of the operator $\diamond$ into $f_\diamond(e_1, e_2) = e_\diamond$, where $e_\diamond$ is a learnable parameter and is different for different operators. Therefore, $e_\diamond$ acts like the embedding of the operator $\diamond$, and the decoding process is more similar to a general seq2seq model. The results show that the semantic transformer in the original model encodes not only the last operator applied on the operands but other information that helps the selectors.

Word-Based - Semantic Representation To explicitly evaluate the effectiveness of operands' semantic representations, we rewrite semantic representation of the i-th operand in the problem texts



	58.0	quantifier 个	banana 香蕉	,	every 每	(basket) <unk>	6.0	quantifier 个	,	take off 拿掉	how many 多少	quantifier 个	,	then 就	can 可以	exactly 正好	fill 装	9.0	quantifier 个	baskets 篮子	了	<unk>	.
58.0	.02	.15	.02	.03	.26	.14	.01	.02	.00	.02	.31	.00	.00	.00	.00	.00	.01	.00	.01	.00	.00	.00	.00
6.0	.01	.00	.00	.00	.01	.09	.42	.21	.02	.05	.13	.00	.00	.00	.00	.00	.01	.01	.02	.00	.01	.00	.00
9.0	.00	.00	.00	.00	.00	.00	.00	.00	.00	.01	.01	.00	.00	.00	.00	.08	.65	.21	.01	.01	.01	.00	.00

Figure 4: The self-attention map visualization of operands' semantic expressions for the problem "*There are 58 bananas. Each basket can contain 6 bananas. How many bananas are needed to be token off such that exactly 9 baskets are filled?*".

from (2) to $e^c_i = b^c_i$, where b^c_i is a parameter. Thus for every problem, the representation of the i-th operand is identical, even though their meanings in different problems may be different. This modification assumes that no semantic information is captured by b^c_i, which can merely represent a symbolic placeholder in an equation. Because the semantic transformer is to transform the semantic representations, applying this component is meaningless. Here the semantic transformer is also replaced with $f_\diamond(e_1, e_2) = e_\diamond$ as the setting of the previous ablation test. The results show that the model without using semantic representations of operands causes a significant accuracy drop of 3.5%. The main contribution of this paper about modeling semantic meanings of symbols is validated and well demonstrated here.

5 Qualitative Analysis

To further analyze whether the proposed model can provide interpretation and reasoning, we visualize the learned semantic representations of constants to check where the important cues are,

5.1 Constant Embedding Analysis

To better understand the information encoded in the semantic representations of constants in the problem, a self-attention is performed when their semantic representations are extracted by the encoder. Namely, we rewrite (2) as

$$e^c_i = \text{Attention}(h^E_{p_i}, \{h^E_t\}^m_{t=1}. \quad (20)$$

Then we check the trained self-attention map (α in the attention function) on the validation dataset.

For some problems, the self-attention that generates semantic representations of constants in the problem concentrates on the number's quantifier or unit, and sometimes it also focuses on informative verbs, such as "*gain*", "*get*", "*fill*", etc., in the sentence. For example, Figure 4 shows the attention weights for an example math word problem, where lighter colors indicate higher weights. The numbers "*58*" and "*6*" focus more on the quantifier-related words (e.g. "*every*" and "*how many*"), while "*9*" pays higher attention to the verb "*fill*". The results are consistent with those handcraft features for solving math word problems proposed by the prior research (Hosseini et al., 2014; Roy and Roth, 2015; Roy et al., 2015). Hence, we demonstrate that the automatically learned semantic representations indeed capture critical information that facilitates solving math word problems without providing human-crafted knowledge.

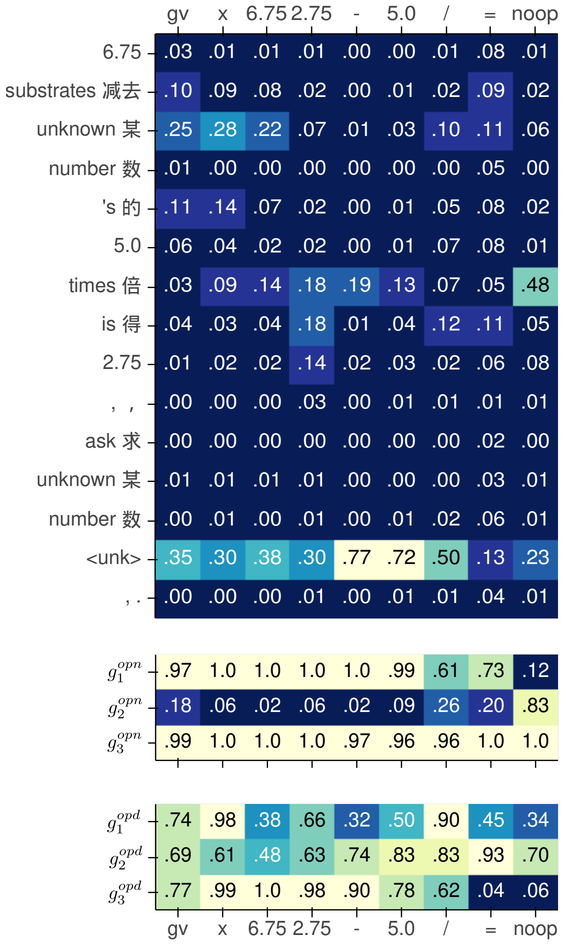
5.2 Decoding Process Visualization

We visualize the attention map (q_t in (6)) to see how the attention helps the decoding process. An example is shown in the top of Figure 5, where most attention focuses on the end of the sentence. Unlike the machine translation task, the attention shows the word-level alignment between source and target languages, solving math word problems requires high-level understanding due to the task complexity.

To further analyze the effectiveness of the proposed gating mechanisms for stack action and operand selection, the activation of gates g^{sa}, g^{opd} at each step of the decoding process is shown in the bottom of Figure 5. It shows that most of time, the gate activation is high, demonstrating that the proposed gating mechanisms play an important role during decoding. We also observe a common phenomenon that the activation g^{sa}_2, which controls how much attention the stack action selector puts on the stack state when deciding an stack action, is usually low until the last "operator application" stack action. For example, in the example of Figure 5, g^{sa}_2 is less than 0.20 till the last argument selection stack action, and activates when deciding the *division operator application* ($\div$) and the *equal application* ($=$). It may result from the higher-level semantics of the operand $(6.75-2.75)$ on the stack when selecting the stack action *division operator application* ($\div$). In terms

Problem & Results
红花有60朵，黄花比红花多1/6朵，黄花有多少朵． (There are 60 red flowers. Yellow flowers are more than red ones by 1/6. How many yellow flowers are there?) *Generated Equation*: $60 + \frac{1}{6}$ *Correct Answer*: 70
火车48小时行驶5920千米，汽车25小时行驶2250千米，汽车平均每小时比火车每小时慢多少千米？ (The train travels 5920 kilometers in hours, and the car travels 2250 kilometers in 25 hours. How many kilometers per hour is the car slower than the train?) *Generated Equation*: $2250 \div 25 - 5920 \div 48$ *Correct Answer*: $33\frac{1}{3}$
小红前面5人，后面7人，一共有多少人？ (There are 5 people in front of Little Red and 7 people behind. How many persons are there in total?) *Generated Equation*: $5 + 7$ *Correct Answer*: 13

Table 3: Randomly sampled incorrect predictions.



	gv	x	6.75	2.75	-	5.0	/	=	noop
6.75	.03	.01	.01	.01	.00	.00	.01	.08	.01
substrates 减去	.10	.09	.08	.02	.00	.01	.02	.09	.02
unknown 某	.25	.28	.22	.07	.01	.03	.10	.11	.06
number 数	.01	.00	.00	.00	.00	.00	.00	.05	.00
's 的	.11	.14	.07	.02	.00	.01	.05	.08	.02
5.0	.06	.04	.02	.02	.00	.01	.07	.08	.01
times 倍	.03	.09	.14	.18	.19	.13	.07	.05	.48
is 得	.04	.03	.04	.18	.01	.04	.12	.11	.05
2.75	.01	.02	.02	.14	.02	.03	.02	.06	.08
，,	.00	.00	.00	.03	.00	.01	.01	.01	.01
ask 求	.00	.00	.00	.00	.00	.00	.00	.02	.00
unknown 某	.01	.01	.01	.01	.00	.00	.00	.03	.01
number 数	.00	.01	.00	.01	.00	.01	.02	.06	.01
<unk>	.35	.30	.38	.30	.77	.72	.50	.13	.23
, .	.00	.00	.00	.01	.00	.01	.01	.04	.01
g_1^{opn}	.97	1.0	1.0	1.0	1.0	.99	.61	.73	.12
g_2^{opn}	.18	.06	.02	.06	.02	.09	.26	.20	.83
g_3^{opn}	.99	1.0	1.0	1.0	.97	.96	.96	1.0	1.0
g_1^{opd}	.74	.98	.38	.66	.32	.50	.90	.45	.34
g_2^{opd}	.69	.61	.48	.63	.74	.83	.83	.93	.70
g_3^{opd}	.77	.99	1.0	.98	.90	.78	.62	.04	.06

Figure 5: Word attention and gate activation (g^{sa} and g^{opd}) visualization when generating stack actions for the problem "*6.75 deducting 5 times of an unknown number is 2.75. What is the unknown number?*", where the associated equation is $x = (6.75 - 2.75) \div 5$. Note that g^{opd} is meaningful only when the t-th stack action is push_op.

of the activation of g^{opd}, we find that three features are important in most cases, demonstrating the effectiveness of the proposed mechanisms.

5.3 Error Analysis

We randomly sample some results predicted incorrectly by our model shown in Table 3. In the first example, the error is due to the language ambiguity, and such ambiguity cannot be resolved without considering the exact value of the number. From the second example, although our model identifies the problem as a comparison problem successfully, it handles the order of the operands incorrectly. For the third problem, it cannot be solved by using only the surface meaning but requires some common sense. Therefore, above phenomena show the difficulty of solving math word problems and the large room for improvement.

6 Conclusion

We propose an end-to-end neural math solver using an encoder-decoder framework that incorporates semantic representations of numbers in order to generate mathematical symbols for solving math word problems. The experiments show that the proposed model achieves the state-of-the-art performance on the benchmark dataset, and empirically demonstrate the effectiveness of each component in the model. In sum, the proposed neural math solver is designed based on how human performs reasoning when writing equations, providing better interpretation without the need of labeled rationals.

References

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 523–533.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 271–281.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 158–167.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Sourav Mandal and Sudip Kumar Naskar. 2019. Solving arithmetic mathematical word problems: A review and recent advancements. In *Information Technology and Applied Mathematics*, pages 95–114. Springer.

Purvanshi Mehta, Pruthwik Mishra, Vinayak Athavale, Manish Shrivastava, and Dipti Misra Sharma. 2017. Deep neural network based system for solving arithmetic word problems. In *Proceedings of the IJCNLP 2017*, pages 65–68.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. 2018. Data-driven methods for solving algebra word problems. *CoRR*, abs/1804.10718.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *TACL*, 6:159–172.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing : Mapping sentences to grounded equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1088–1097.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *TACL*, 3:1–13.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1132–1142.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 494–504.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. MathDQN: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

A Algorithm Detail

The training and inference procedures are shown in Algortihm 1.

B Hyperparameter Setup

The model is trained with the optimizer `adam` (Kingma and Ba, 2014), and the learning rate is set to 0.001. Pretrained embeddings using FastText (Joulin et al., 2016) are adopted. The hidden state size of LSTM used in the encoder and decoder is 256. The dimension of hidden layers in attention, semantic transformer and operand/stack action selector is 256. The dropout rate is set as 0.1 before inputting the decoder LSTM, before the stack action selector and after the hidden layer of the stack action selector and attention. The reported accuracy is the result of 5-fold cross-validation, same as Wang et al. for fair comparison.

C Error Analysis between Seq2Seq

We implement the seq2seq model as proposed by Wang et al. and compare the performance difference between our proposed model and the baseline seq2seq model. Table 4 shows the generated results seq2seq predicts correctly but our model predicts incorrectly. Table 5 show the results our model can predict correctly but seq2seq cannot.

Problem & Results
小红前面 5 人，后面 7 人，一共有多少人？ (There are 5 people in front of Little Red and 7 people behind. How many persons are there in total?) *Proposed Model*: $5 + 7$ *Seq2Seq Model*: $5 + 7 + 1$
两个数相差28，如果被减数减少3，减数增加5，那么它们的差=？ (The difference between two numbers is 28. If the minuend is reduced by 3, and the subtrahend is increased by 5, then their difference=?) *Proposed Model*: $(28 - 3) \div 5$ *Seq2Seq Model*: $28 - (3 + 5)$
机床厂第一车间有55人，第二车间有45人，每人每天平均生产261个零件，这两个车间每天共生产多少个零件？ (There are 55 people in the first workshop of the machine tool factory and 45 people in the second workshop. Each person produces 261 small components per day in average. How many components do the two workshops produce every day in total?) *Proposed Model*: $(55 + 45) \div 261$ *Seq2Seq Model*: $(55 + 45) \times 261$
箭鱼游动时的速度是28米/秒，8秒可以游多少米？ (The swordfish swims at speed 28 meters/sec. How many meters can it swim in 8 seconds?) *Proposed Model*: $28 \div 8$ *Seq2Seq Model*: 28×8
水果店有梨子387千克，卖出205千克后，又运来一批，现在水果店共有梨子945千克．水果店又运来梨子多少千克？ (The fruit shop has 387 kilograms of pears . After selling 205 kilograms, some pears arrive. Now the fruit shop has 945 kilograms of pears in total. How many kilograms of pears does the fruit shop get?) *Proposed Model*: $945 \times (387 - 205)$ *Seq2Seq Model*: $945 - (387 - 205)$
王老师买排球用了40元，买篮球用的钱数是排球的3倍．王老师买球一共用了多少元？ (Teacher Wang spent 40 dollars buying volleyballs and 3 times of money for basketballs. How many dollars did Teacher Wang spend for the balls?) *Proposed Model*: $40 \div 3 + 40$ *Seq2Seq Model*: $40 + 40 \times 3$
筑路队修筑一条长1200米的公路，甲队单独修40天可以完成任务，乙队单独修30天可以完成任务．甲队每天修的比乙队少多少米？ (The road construction team built a road with a length of 1200 meters. Team A can complete the task in 40 days alone, and team B can complete the task in 30 days alone. How many meters does team A construct more than team B every day?) *Proposed Model*: $1200 \div 40 - 1200 \div 30$ *Seq2Seq Model*: $1200 \div 30 - 1200 \div 40$
一共1800本，我们六年级分得2/9，分给五年级的本数相当于六年级的4/5，五年级分得多少本？ (There are 1800 books in total. We sixth grade get 2/9. The number of books given to the fifth grade is equal to 4/5 of the number to the sixth grade. How many books does the fifth grade get?) *Proposed Model*: $1800 \times \frac{2}{9} \div \frac{4}{5}$ *Seq2Seq Model*: $1800 \times \frac{2}{9} \times \frac{4}{5}$
有一批布料，如果只做上衣可以做10件，如果只做裤子可以做15条，那么这批布料可以做几套这样的衣服？ (There is a batch of fabrics. If all is used for making shirts, 10 pieces can be made, and 15 pieces if used to make pants only. Then how many suits of such clothes can be made with this batch of fabric?) *Proposed Model*: $10 \times 1 \div 15$ *Seq2Seq Model*: $1 \div (1 \div 10 + 1 \div 15)$
贝贝的钱买一本5.9元笔记本差0.6元，他买一本4.8元的，剩下的钱正好买一只圆珠笔，这只圆珠笔多少钱？ (Beibei needs 0.6 dollars more to buy a notebook of 5.9 dollars. If he buys one of 4.8 dollars, the remaining money allows her to buy exactly one ball pen. How much is the ball pen?) *Proposed Model*: $5.9 + 0.6 - 4.8$ *Seq2Seq Model*: $5.9 - 0.6 - 4.8$

Table 4: Examples that Seq2Seq predicts correctly while our proposed model predicts incorrectly.

Problem & Results
医院里经常要给病人输入葡萄糖水，这种葡萄糖水是把葡萄糖和水按1：19配制的，根据这些信息，你能知道什么？ (In hospital, it is often necessary to give glucose injection to patient. This glucose water is prepared by mixing glucose and water at 1:19. Based on this information, what do you know?) *Proposed Model*: $1 \div (1 + 19.0)$ *Seq2Seq Model*: $1 \times (1 + 19.0)$
一根长2.45米的木桩打入河底，现在测得木桩水上部分长0.75米，水中长1.05米，求这根桩打在泥中的长度=多少米？ (A wooden pile of 2.45 meters long is hammered into the bottom of a river. Now the part above water is measured as 0.75 meters long, and the part in the water is measured as 1.05 meters long. How long is the part of the pile in the mud?) *Proposed Model*: $2.45 - 0.75 - 1.05$ *Seq2Seq Model*: $2.45 + 0.75 + 1.05$
李强6月份的生活费为255元，比计划节省了15%，节省了多少元． (Li Qiang's living expenses in June were 255 dollars, 15% savings over the plan. How much did he save?) *Proposed Model*: $(255.0 \div (1 - 0.15)) \times 0.15$ *Seq2Seq Model*: $0.15 = 6.0/(1 - 255.0) - 6.0$
小芳在计算一个数除以10时，将除号看成了乘号，结果得3.2，正确的结果应该=． (When Xiaofang calculates a number divided by 10 , the division sign is mistakenly treated as a multiplication sign, and the result is 3.2. The correct result should be = .) *Proposed Model*: $3 \div 10 \div 10$ *Seq2Seq Model*: $3.2 \div (1 + 10)$
24 + 91 的 2/13，所得的和再除 19/20，商 = ？ (2/13 of 91 + 24, and the sum is divided by 19/20, quotient = ?) *Proposed Model*: $\frac{19}{20} \div (24 + 91 \times \frac{2}{13})$ *Seq2Seq Model*: $\frac{19}{20} \div (24 \times 91 - \frac{2}{13})$
1/3 + 0.25 = ？ (1/3 + 0.25 = ?) *Proposed Model*: $\frac{1}{3} + 0.25$ *Seq2Seq Model*: $\frac{1}{3} \times 0.25$
商店运来鸡蛋和鸭蛋各7箱．鸡蛋每箱重26千克，鸭蛋每箱重31千克，商店一共运来的鸡蛋和鸭蛋共多少千克？ (The store shipped 7 boxes of eggs and duck eggs respectively. Eggs weigh 26 kilograms per box, duck eggs weigh 31 kilograms per box. How many kilograms of eggs and duck eggs are shipped from the store in total?) *Proposed Model*: $26 \times 7 + 31 \times 7$ *Seq2Seq Model*: $26 \times 7 + 31$
3.8 - 2.54 + 1.46 = ？ (3.8 - 2.54 + 1.46 =) *Proposed Model*: $3.8 - 2.54 + 1.46$ *Seq2Seq Model*: $3.8 + 2.54 + 1.46$
有一池水，第一天放出200吨，第二天比第一天多放20%，第3天放了整池水的36%，正好全部放完．这池水共 有多少吨？ (There was a pool of water, which released 200 tons of water in the first day, 20% more in the second day than the first day, and 36% of the whole pool on the third day. Then the water is gone. How many tons of water did this pool have?) *Proposed Model*: $(200.0 + 200.0 \times (1 + 0.2)) \div (1 - 0.36)$ *Seq2Seq Model*: $(200.0 + 0.2) \times 3.0 + 0.2 \times (1 - 0.36)$
16 的 5/12 比一个数的 7 倍多 2 ， 这个数 = ？ (5/12 of 16 is more than 7 times of a number by 2. What is the number=?) *Proposed Model*: $(16 \times \frac{5}{12} - 2) \div 7$ *Seq2Seq Model*: $(16 \times \frac{5}{12} + 7) \div 2$

Table 5: Examples that Seq2Seq predicts incorrectly while our proposed model predicts correctly.

Iterative Search for Weakly Supervised Semantic Parsing

Pradeep Dasigi[♡], **Matt Gardner**[♥], **Shikhar Murty**[◇],
Luke Zettlemoyer[♣], and **Eduard Hovy**[♠]
[♡]Allen Institute for Artificial Intelligence, Seattle, Washington
[♥]Allen Institute for Artificial Intelligence, Irvine, California
[◇]Mila, Université de Montréal [♣]University of Washington [♠]Carnegie Mellon University
`pradeepd@allenai.org`

Abstract

Training semantic parsers from question-answer pairs typically involves searching over an exponentially large space of logical forms, and an unguided search can easily be misled by spurious logical forms that coincidentally evaluate to the correct answer. We propose a novel iterative training algorithm that alternates between searching for consistent logical forms and maximizing the marginal likelihood of the retrieved ones. This training scheme lets us iteratively train models that provide guidance to subsequent ones to search for logical forms of increasing complexity, thus dealing with the problem of spuriousness. We evaluate these techniques on two hard datasets: WIKITABLEQUESTIONS (WTQ) and Cornell Natural Language Visual Reasoning (NLVR), and show that our training algorithm outperforms the previous best systems, on WTQ in a comparable setting, and on NLVR with significantly less supervision.

1 Introduction

Semantic parsing is the task of translating natural language utterances into machine-executable meaning representations, often called *programs* or *logical forms*. These logical forms can be executed against some representation of the context in which the utterance occurs, to produce a *denotation*. This setup allows for complex reasoning over contextual knowledge, and it has been successfully used in several natural language understanding problems such as question answering (Berant et al., 2013), program synthesis (Yin and Neubig, 2017) and building natural language interfaces (Suhr et al., 2018).

Recent work has focused on training semantic parses via *weak supervision* from denotations alone (Liang et al., 2011; Berant et al., 2013). This is because obtaining logical form annotations is generally expensive (although recent work has addressed this issue to some extent (Yih et al., 2016)), and not assuming full supervision lets us be agnostic about the logical form language. The second reason is more important in open-domain semantic parsing tasks where it may not be possible to arrive at a complete set of operators required by the task. However, training semantic parsers with weak supervision requires not only searching over an exponentially large space of logical forms (Berant et al., 2013; Artzi and Zettlemoyer, 2013; Pasupat and Liang, 2015; Guu et al., 2017, *inter alia*) but also dealing with *spurious* logical forms that evaluate to the correct denotation while not being semantically equivalent to the utterance. For example, if the denotations are binary, 50% of all syntactically valid logical forms evaluate to the correct answer, regardless of their semantics. This problem renders the training signal extremely noisy, making it hard for the model to learn anything without some additional guidance during search.

We introduce two innovations to improve learning from denotations. Firstly, we propose an iterative search procedure for gradually increasing the complexity of candidate logical forms for each training instance, leading to better training data and better parsing accuracy. This procedure is implemented via training our model with two interleaving objectives, one that involves searching for logical forms of limited complexity during training (online search), and another that maximizes the marginal likelihood of retrieved logical forms. Second, we include a notion of coverage over the question in the search step to guide the training algorithm towards logical forms that not only evaluate to the correct denotation, but also have some connection to the words in the utterance.

We demonstrate the effectiveness of these two techniques on two difficult reasoning tasks: WIK-

Proceedings of NAACL-HLT 2019, pages 2669–2680
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

iTableQuestions(WTQ) (Pasupat and Liang, 2015), an open domain task with significant lexical variation, and Cornell Natural Language Visual Reasoning (NLVR) (Suhr et al., 2017), a closed domain task with binary denotations, and thus far less supervision. We show that: 1) interleaving online search and MML over retrieved logical forms (§4) is a more effective training algorithm than each of those objectives alone; 2) coverage guidance during search (§3) is helpful for dealing with weak supervision, more so in the case of NLVR where the supervision is weaker; 3) a combination of the two techniques yields 44.3% test accuracy on WTQ, outperforming the previous best single model in a comparable setting, and 82.9% test accuracy on NLVR, outperforming the best prior model, which also relies on greater supervision.

2 Background

2.1 Weakly supervised semantic parsing

We formally define semantic parsing in a weakly supervised setup as follows. Given a dataset where the i^{th} instance is the triple $\{x_i, w_i, d_i\}$, representing a sentence x_i, the world w_i associated with the sentence, and the corresponding denotation d_i, our goal is to find y_i, the translation of x_i in an appropriate logical form language (see §5.3), such that $[\![y_i]\!]^{w_i} = d_i$; i.e., the execution of y_i in world w_i produces the correct denotation d_i. A semantic parser defines a distribution over logical forms given an input utterance: $p(Y|x_i;\theta)$.

2.2 Training algorithms

In this section we describe prior techniques for training semantic parsers with weak supervision: maximizing marginal likelihood, and reward-based methods.

2.2.1 Maximum marginal likelihood

Most work on training semantic parsers from denotations maximizes the likelihood of the denotation given the utterance. The semantic parsing model itself defines a distribution over *logical forms*, however, not *denotations*, so this maximization must be recast as a *marginalization* over logical forms that evaluate to the correct denotation:

$$\max_{\theta} \prod_{x_i, d_i \in \mathcal{D}} \sum_{y_i \in Y | [\![y_i]\!]^{w_i} = d_i} p(y_i|x_i;\theta) \qquad (1)$$

This objective function is called *maximum marginal likelihood* (MML). The inner summation is in general intractable to perform during training, so it is only approximated.

Most prior work (Berant et al., 2013; Goldman et al., 2018, *inter alia*) approximate the intractable marginalization by summing over logical forms obtained via beam search during training. This typically results in frequent search failures early during training when model parameters are close to random, and in general may only yield spurious logical forms in the absence of any guidance. Since modern semantic parsers typically operate without a lexicon, new techniques are essential to provide guidance to the search procedure (Goldman et al., 2018).

One way of providing this guidance during search is to perform some kind of heuristic search up front to find a set of logical forms that evaluate to the correct denotation, and use those logical forms to approximate the inner summation (Liang et al., 2011; Krishnamurthy et al., 2017). The particulars of the heuristic search can have a large impact on performance; a smaller candidate set has lower noise, while a larger set makes it more likely that the correct logical form is in it, and one needs to strike the right balance. In this paper, we refer to the MML that does search during training as *dynamic MML*, and the one that does an offline search as *static MML*.

The main benefit of dynamic MML is that it adapts its training signal over time. As the model learns, it can increasingly focus its probability mass on a small set of very likely logical forms. The main benefit of static MML is that there is no need to search during training, so there is a consistent training signal even at the start of training, and it is typically more computationally efficient than dynamic MML.

2.2.2 Reward-based methods

When training weakly supervised semantic parsers, it is often desirable to inject some prior knowledge into the training procedure by defining arbitrary reward or cost functions. There exists prior work that use such methods, both in a reinforcement learning setting (Liang et al., 2017, 2018), and otherwise (Iyyer et al., 2017; Guu et al., 2017). In our work, we define a customized cost function that includes a coverage term, and use a Minimum Bayes Risk (MBR) (Goodman, 1996; Goel and Byrne, 2000; Smith and Eisner,

2006) training scheme, which we describe in §3.

3 Coverage-guided search

Weakly-supervised training of semantic parsers relies heavily on lexical cues to guide the initial stages of learning to good logical forms. Traditionally, these lexical cues were provided in the parser's lexicon. Neural semantic parsers remove the lexicon, however, and so need another mechanism for obtaining these lexical cues. In this section we introduce the use of coverage to inject lexicon-like information into neural semantic parsers.

Coverage is a measure of relevance of the candidate logical form y_i to the input x_i, in terms of how well the productions in y_i map to parts of x_i. We use a small manually specified lexicon as a mapping from source language to the target language productions, and define coverage of y_i as the number of productions triggered by the input utterance, according to the lexicon, that are included in y_i.

We use this measure of coverage to augment our loss function, and train using an MBR based algorithm as follows. We use beam search to train a model to minimize the expected value of a cost function $\mathcal{C}$:

$$\min_{\theta} \sum_{i=1}^{N} \mathbb{E}_{\tilde{p}(y_i|x_i;\theta)} \mathcal{C}(x_i, y_i, w_i, d_i) \tag{2}$$

where $\tilde{p}$ is a *re-normalization*[1] of the probabilities assigned to all logical forms on the beam.

We define the cost function $\mathcal{C}$ as:

$$\mathcal{C}(x_i, y_i, w_i, d_i) = \lambda \mathcal{S}(y_i, x_i) + (1-\lambda) \mathcal{T}(y_i, w_i, d_i) \tag{3}$$

where the function $\mathcal{S}$ measures the number of items that y_i is missing from the actions (or grammar production rules) triggered by the input utterance x_i given the lexicon; and the function $\mathcal{T}$ measures the consistency of the evaluation of y_i in w_i, meaning that it is 0 if $[\![y_i]\!]^{w_i} = d_i$, or a value e otherwise. We set e as the maximum possible value of the coverage cost for the corresponding instance, to make the two costs comparable in magnitude. λ is a hyperparameter that gives the relative weight of the coverage cost.

4 Iterative search

In this section we describe the iterative technique for refining the set of candidate logical forms associated with each training instance.

As discussed in §2.2, most prior work on weakly-supervised training of semantic parsers uses dynamic MML. This is particularly problematic in domains like NLVR, where the supervision signal is binary—it is very hard for dynamic MML to bootstrap its way to finding good logical forms. To solve this problem, we interleave static MML, which has a consistent supervision signal from the start of training, with the coverage-augmented MBR algorithm described in §3.

In order to use static MML, we need an initial set of candidate logical forms. We obtain this candidate set using a bounded-length exhaustive search, filtered using heuristics based on the same lexical mapping used for coverage in §3. A bounded-length search will not find logical forms for the entire training data, so we can only use a subset of the data for initial training. We train a model to convergence using static MML on these logical forms, then use that model to initialize coverage-augmented MBR training. This gives the model a good starting place for the dynamic learning algorithm, and the search at training time can look for logical forms that are longer than could be found with the bounded-length exhaustive search. We train MBR to convergence, then use beam search on the MBR model to find a new set of candidate logical forms for static MML on the training data. This set of logical forms can have a greater length than those in the initial set, because this search uses model scores to not exhaustively explore all possible paths, and thus will likely cover more of the training data. In this way, we can iteratively improve the candidate logical forms used for static training, which in turn improves the starting place for the online search algorithm.

Algorithm 1 concretely describes this process. `Decode` in the algorithm refers to running a beam search decoder that returns a set of consistent logical forms (i.e. $\mathcal{T} = 0$) for each of the input utterances. We start off with a seed dataset $\mathcal{D}^0$ for which consistent logical forms are available.

5 Datasets

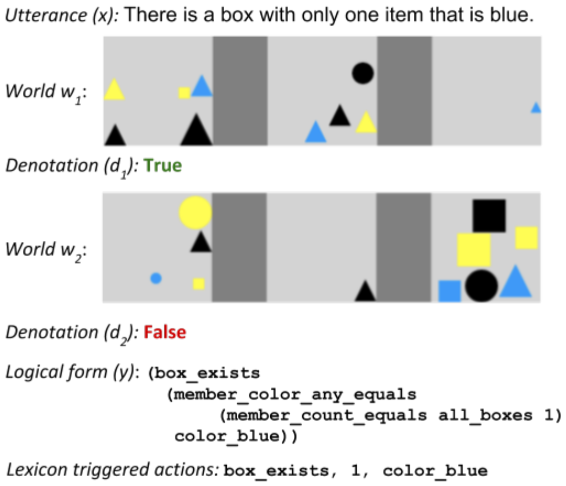
We will now describe the two datasets we use in this work to evaluate our methods – Cornell NLVR

---

[1] Note that without this re-normalization, and with a -1/0 cost function based on denotation accuracy, MBR will maximize the likelihood of correct logical forms on the beam, which is equivalent to dynamic MML.

```
Input  : Dataset $\mathcal{D} = \{X, W, D\}$; and
         seed set $\mathcal{D}^0 = \{X^0, Y^0\}$ such that
         $X^0 \subset X$ and $\mathcal{C}(x_i^0, y_i^0, W_i, D_i) = 0$
Output: Model parameters $\theta^{\text{MBR}}$
Initialize dataset $\mathcal{D}^{\text{MML}} = \mathcal{D}^0$;
while $Acc(\mathcal{D}_{dev})$ is increasing do
    $\theta^{\text{MML}} = \texttt{MML}(\mathcal{D}^{\text{MML}})$;
    Initialize $\theta^{\text{MBR}} = \theta^{\text{MML}}$;
    Update $\theta^{\text{MBR}} = \texttt{MBR}(\mathcal{D}; \theta^{\text{MBR}})$;
    Update $\mathcal{D}^{\text{MML}} = \texttt{Decode}(\mathcal{D}; \theta^{\text{MBR}})$;
end
```

Algorithm 1: Iterative coverage-guided search



Utterance (x): There is a box with only one item that is blue.

World w_1:

Denotation (d_1): True

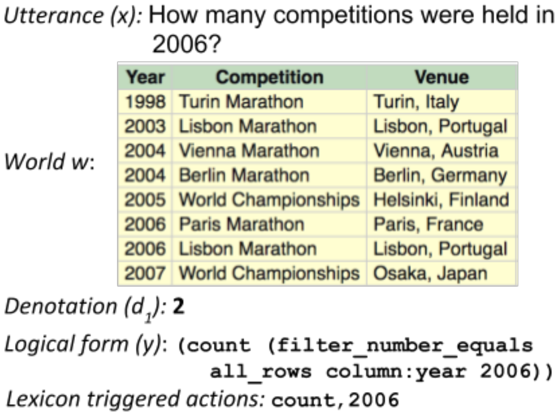
World w_2:

Denotation (d_2): False

Logical form (y):
```
(box_exists
    (member_color_any_equals
        (member_count_equals all_boxes 1)
     color_blue))
```

Lexicon triggered actions: `box_exists, 1, color_blue`

Figure 1: Example from NLVR dataset showing an utterance associated with two worlds and corresponding binary denotations. Also shown are the logical form and the actions triggered by the lexicon from the utterance.



Utterance (x): How many competitions were held in 2006?

World w:

Year	Competition	Venue
1998	Turin Marathon	Turin, Italy
2003	Lisbon Marathon	Lisbon, Portugal
2004	Vienna Marathon	Vienna, Austria
2004	Berlin Marathon	Berlin, Germany
2005	World Championships	Helsinki, Finland
2006	Paris Marathon	Paris, France
2006	Lisbon Marathon	Lisbon, Portugal
2007	World Championships	Osaka, Japan

Denotation (d_1): **2**

Logical form (y):
```
(count (filter_number_equals
        all_rows column:year 2006))
```

Lexicon triggered actions: `count, 2006`

Figure 2: Example from WIKITABLEQUESTIONS dataset showing an utterance, a world, associated denotation, corresponding logical form, and actions triggered by the lexicon.

and WIKITABLEQUESTIONS.

5.1 Cornell NLVR

Cornell NLVR is a language-grounding dataset containing natural language sentences provided along with synthetically generated visual contexts, and a label for each sentence-image pair indicating whether the sentence is true or false in the given context. Figure 1 shows two example sentence-image pairs from the dataset (with the same sentence). The dataset also comes with structured representations of images, indicating the color, shape, size, and x- and y-coordinates of each of the objects in the image. While we show images in Figure 1 for ease of exposition, we use the structured representations in this work.

Following the notation introduced in §2.1, x_i in this example is *There is a box with only one item that is blue*. The structured representations associated with the two images shown are two of the worlds (w_i^1 and w_i^2), in which x_i could be evaluated. The corresponding labels are the denotations d_i^1 and d_i^2 that a translation y_i of the sentence x_i is expected to produce, when executed in the two worlds respectively. That the same sentence occurs with multiple worlds is an important property of this dataset, and we make use of it by defining the function $\mathcal{T}$ to be 0 only if $\forall_{w_i^j, d_i^j} [\![y_i]\!]^{w_i^j} = d_i^j$.

5.2 WIKITABLEQUESTIONS

WIKITABLEQUESTIONS is a question-answering dataset where the task requires answering complex questions in the context of Wikipedia tables. An example can be seen in Figure 2. Unlike NLVR, the answers are not binary. They can instead be cells in the table or the result of numerical or set-theoretic operations performed on them.

5.3 Logical form languages

For NLVR, we define a typed variable-free functional query language, inspired by the GeoQuery language (Zelle and Mooney, 1996). Our language contains six basic types: `box` (referring to one of the three gray areas in Figure 1), `object` (referring to the circles, triangles and squares in Figure 1), `shape`, `color`, `number` and `boolean`. The constants in our language are color and shape names, the set of all boxes in an image, and the set of all objects in an image. The functions in our language include those for filtering objects and boxes, and making assertions, a higher order function for handling negations, and a function for querying objects in boxes. This type specification of constants and functions gives us a grammar with 115 productions, of which 101 are terminal productions (see Appendix A.1 for the complete set of rules in our grammar). Figure 1 shows an example of a complete logical form in our language.

For WTQ, we use the functional query language used by (Liang et al., 2018) as the logical form language. Figure 2 shows an example logical form.

5.4 Lexicons for coverage

The lexicon we use for the coverage measure described in §3 contains under 40 rules for each logical form language. They mainly map words and phrases to constants and unary functions in the target language. The complete lexicons are shown in the Appendix. Figures 1 and 2 also show the actions triggered by the corresponding lexicons for the utterances shown. We find that small but precise lexicons are sufficient to guide the search process away from spurious logical forms. Moreover, as shown empirically in §6.4, the model for NLVR does not learn much without this simple but crucial guidance.

6 Experiments

We evaluate both our contributions on NLVR and WIKITABLEQUESTIONS.

6.1 Model

In this work, we use a grammar-constrained encoder-decoder neural semantic parser for our experiments. Of the many variants of this basic architecture (see §7), all of which are essentially seq2seq models with constrained outputs and/or re-parameterizations, we choose to use the parser of Krishnamurthy et al. (2017), as it is particularly well-suited to the WIKITABLEQUESTIONS dataset, which we evaluate on.

The encoder in the model is a bi-directional recurrent neural network with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells, and the decoder is a grammar-constrained decoder also with LSTM cells. Instead of directly outputting tokens in the logical form, the decoder outputs *production rules* from a CFG-like grammar. These production rules sequentially build up an abstract syntax tree, which determines the logical form. The model also has an entity linking component for producing table entities in the logical forms; this component is only applicable to WIKITABLEQUESTIONS, and we remove it when running experiments on NLVR. The particulars of the model are not the focus of this work, so we refer the reader to the original paper for more details.

In addition, we slightly modify the constrained decoding architecture from (Krishnamurthy et al., 2017) to bias the predicted actions towards those that would decrease the value of $\mathcal{S}(y_i, x_i)$. This is done using a coverage vector, $v_i^{\mathcal{S}}$ for each training instance that keeps track of the production rules triggered by x_i, and gets updated whenever one of those desired productions is produced by the decoder. That is, $v_i^{\mathcal{S}}$ is a vector of 1s and 0s, with 1s indicating the triggered productions that are yet to be produced by the decoder. This is similar to the idea of checklists used by Kiddon et al. (2016). The decoder in the original architecture scores output actions at each time step by computing a dot product of the predicted action representation with the embeddings of each of the actions. We add a weighted sum of all the actions that are yet to produced:

$$s_i^a = e^a.(p_i + \gamma * v_i^{\mathcal{S}}.E) \tag{4}$$

where s_i^a is the score of action a at time step i, e^a is the embedding of that action, p_i is the predicted action representation, E is the set of embeddings of all the actions, and γ is a learned parameter for regularizing the bias towards yet-to-be produced triggered actions.

6.2 Experimental setup

NLVR We use the standard train-dev-test split for NLVR, containing 12409, 988 and 989 sentence-image pairs respectively. NLVR contains most of the sentences occurring in multiple worlds (with an average of 3.9 worlds per sentence). We set the word embedding and action embedding sizes to 50, and the hidden layer size of both the encoder and the decoder to 30. We initialized all the parameters, including the word and action embeddings using Glorot uniform initialization (Glorot and Bengio, 2010). We found that using pretrained word representations did not help. We added a dropout (Srivastava et al., 2014) of 0.2 on the outputs of the encoder and the decoder and before predicting the next action, set the beam size to 10 both during training and at test time, and trained the model using ADAM (Kingma and Ba, 2014) with a learning rate of 0.001. All the hyper-parameters are tuned on the validation set.

WIKITABLEQUESTIONS This dataset comes with five different cross-validation folds of training data, each containing a different 80/20 split for training and development. We first show results aggregated from all five folds in §6.3, and then show results from controlled experiments on fold 1. We replicated the model presented in Krishnamurthy et al. (2017), and only changed the training algorithm and the language used. We used a beam size of 20 for MBR during training and decoding, and 10 for MML during decoding, and trained the model using Stochastic Gradient Descent (Kiefer et al., 1952) with a learning rate of 0.1, all of which are tuned on the validation sets.

Specifics of iterative search For our iterative search algorithm, we obtain an initial set of candidate logical forms in both domains by exhaustively searching to a depth of 10[2]. During search we retrieve the logical forms that lead to the correct denotations in all the corresponding worlds, and sort them based on their coverage cost using the coverage lexicon described in §5.4, and choose the top-k[3]. At each iteration of the search step in our iterative training algorithm, we increase the maximum depth of our search with a step-size of 2, finding more complex logical forms and covering a larger proportion of the training data. While exhaustive search is prohibitively expensive beyond a fixed number of steps, our training process that uses beam search based approximation can go deeper.

Implementation We implemented our model and training algorithms within the AllenNLP (Gardner et al., 2018) toolkit. The code and models are publicly available at `https://github.com/allenai/iterative-search-semparse`.

6.3 Main results

WIKITABLEQUESTIONS Table 1 compares the performance of a single model trained using Iterative Search, with that of previously published single models. We excluded ensemble models since there are differences in the way ensembles are built for this task in previous work, either in terms of size or how the individual models were chosen. We show both best and average (over 5 folds) single model performance from Liang et al. (2018) (Memory Augmented Policy Optimization). The best model was chosen based on performance on the development set. Our single model performances are computed in the same way. Note that Liang et al. (2018) also use a lexicon similar to ours to prune the seed set of logical forms used to initialize their memory buffer.

Approach	Dev	Test
Pasupat and Liang (2015)	37.0	37.1
Neelakantan et al. (2017)	34.1	34.2
Haug et al. (2018)	-	34.8
Zhang et al. (2017)	40.4	43.7
Liang et al. (2018) (MAPO) (avg.)	42.3	43.1
Liang et al. (2018) (MAPO) (best)	42.7	43.8
Iterative Search (avg.)	42.1	43.9
Iterative Search (best)	**43.1**	**44.3**

Table 1: Comparison of single model performances of Iterative Search with previously reported single model performances

Algorithm	Dev acc.	Test acc.
MAPO	42.1	42.7
Static MML	40.0	42.2
Iterative MML	42.5	43.1
Iterative Search	**43.0**	**43.8**

Table 2: Comparison of iterative search with static MML, iterative MML, and the previous best result from (Liang et al., 2018), all trained on the official split 1 of WIKITABLEQUESTIONS and tested on the official test set.

In Table 2, we compare the performance of our iterative search algorithm with three baselines: 1) Static MML, as described in §2.2.1 trained on the candidate set of logical forms obtained through the heuristic search technique described in §6.2; 2) Iterative MML, also an iterative technique but unlike iterative search, we skip MBR and iteratively train static MML models while increasing the number of decoding steps; and 3) MAPO (Liang et al., 2018), the current best published system on WTQ. All four algorithms are trained and evaluated on the first fold, use the same language, and the bottom three use the same model and the same set of logical forms used to train static MML.

---

[2] It was prohibitively expensive to search beyond depth of 10.

[3] k is a hyperparameter that is chosen on the dev set at each iteration in iterative search, and is typically 10 or 20

NLVR In Table 3, we show a comparison of the performance of our *iterative coverage-guided search* algorithm with the previously published approaches for NLVR. The first two rows correspond to models that are not semantic parsers. This shows that semantic parsing is a promising direction for this task. The closest work to ours is the weakly supervised parser built by (Goldman et al., 2018). They build a lexicon similar to ours for mapping surface forms in input sentences to abstract clusters. But in addition to defining a lexicon, they also manually annotate complete sentences in this abstract space, and use those annotations to perform data augmentation for training a supervised parser, which is then used to initialize a weakly supervised parser. They also explicitly use the abstractions to augment the beam during decoding using caching, and a separately-trained discriminative re-ranker to re-order the logical forms on the beam. As a discriminative re-ranker is orthogonal to our contributions, we show their results with and without it, with "Abs. Sup." being more comparable to our work. Our model, which uses no data augmentation, no caching during decoding, and no discriminative re-ranker, outperforms their variant without reranking on the public test set, and outperforms their best model on the hidden test set, achieving a new state-of-the-art result on this dataset.

6.4 Effect of coverage-guided search

To evaluate the contribution of coverage-guided search, we compare the the performance of the NLVR parser in two different settings: with and without coverage guidance in the cost function. We also compare the performance of the parser in the two settings, when initialized with parameters from an MML model trained to maximize the likelihood of the set of logical forms obtained from exhaustive search. Table 4 shows the results of this comparison. We measure accuracy and consistency of all four models on the publicly available test set, using the official evaluation script. Consistency here refers to the percentage of logical forms that produce the correct denotation in all the corresponding worlds, and is hence a stricter metric than accuracy. The cost weight (λ in Equation 3) was tuned based on validation set performance for the runs with coverage, and we found that $\lambda = 0.4$ worked best.

It can be seen that both with and without initialization, coverage guidance helps by a big margin, with the gap being even more prominent in the case where there is no initialization. When there is neither coverage guidance nor a good initialization, the model does not learn much from unguided search and get a test accuracy not much higher than the majority baseline of 56.2%.

We found that coverage guidance was not as useful for WTQ. The average value of the best performing λ was around 0.2, and higher values neither helped nor hurt performance.

6.5 Effect of iterative search

To evaluate the effect of *iterative search*, we present the accuracy numbers from the search (S) and maximization (M) steps from different iterations in Tables 5 and 6, showing results on NLVR and WTQ, respectively. Additionally, we also show number of decoding steps used at each iterations, and the percentage of sentences in the training data for which we were able to obtain consistent logical forms from the S step, the set that was used in the M step of the same iteration. It can be seen in both tables that a better MML model gives a better initialization for MBR, and a better MBR model results in a larger set of utterances for which we can retrieve consistent logical forms, thus improving the subsequent MML model. The improvement for NLVR is more pronounced (a gain of 21% absolute) than for WTQ (a gain of 3% absolute), likely because the initial exhaustive search provides a much higher percentage of spurious logical forms for NLVR, and thus the starting place is relatively worse.

Complexity of Logical Forms We analyzed the logical forms produced by our iterative search algorithm at different iterations to see how they differ. As expected, for NLVR, allowing greater depths lets the parser explore more complex logical forms. Table 7 shows examples from the validation set that indicate this trend.

7 Related Work

Most of the early methods used for training semantic parsers required the training data to come with annotated logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). The primary limitation of such methods is that manually producing these logical forms is expensive, making it hard to scale these methods across domains.

	Dev.		Test-P		Test-H	
Approach	Acc.	Cons.	Acc.	Cons.	Acc.	Cons.
MaxEnt (Suhr et al., 2017)	68.0	-	67.7	-	67.8	-
BiATT-Pointer (Tan and Bansal, 2018)	74.6	-	73.9	-	71.8	-
Abs. Sup. (Goldman et al., 2018)	84.3	66.3	81.7	60.1	-	-
Abs. Sup. + ReRank (Goldman et al., 2018)	85.7	67.4	84.0	65.0	82.5	63.9
Iterative Search	85.4	64.8	82.4	61.3	**82.9**	**64.3**

Table 3: Comparison of our approach with previously published approaches. We show accuracy and consistency on the development set, and public (Test-P) and hidden (Test-H) test sets.

	No coverage		+ coverage	
	Acc.	Cons.	Acc.	Cons.
No init.	56.4	12.0	73.9	43.6
MML init.	77.7	51.1	80.7	56.4

Table 4: Effect of coverage guidance on NLVR parsers trained with and without initialization from an MML model. Metrics shown are accuracy and consistency on the public test set.

Iter.	Length	% cov.	Step	Dev. Acc
0	10	51	M	64.0
1	12	65	S	81.6
			M	76.5
2	14	65	S	82.7
			M	81.8
3	16	73	S	85.4
			M	83.1
4	18	75	S	84.7
			M	81.2

Table 5: Effect of iterative search (S) and maximization (M) on NLVR. % cov. is the percentage of training data for which the S step retrieves consistent logical forms.

Iter.	Length	% cov.	Step	Dev. Acc
0	10	83.3	M	40.0
1	12	70.2	S	42.5
			M	42.5
2	14	71.3	S	43.1
			M	42.7
3	16	71.0	S	42.8
			M	42.5
4	18	71.0	S	43.0
			M	42.7

Table 6: Iterative search on WIKITABLEQUESTIONS. M and S refer to Maximization and Search steps.

More recent research has focused on training semantic parsers with *weak supervision* (Liang et al., 2011; Berant et al., 2013), or trying to automatically infer logical forms from denotations (Pasupat and Liang, 2016). However, matching the performance of a fully supervised semantic parser with only weak supervision remains a significant challenge (Yih et al., 2016).

The main contributions of this work deal with training semantic parsers with weak supervision, and we gave a detailed discussion of related training methods in §2.2.

We evaluate our contributions on the NLVR and WIKITABLEQUESTIONS datasets. Other work that evaluates on on these datasets include Goldman et al. (2018), Tan and Bansal (2018), Neelakantan et al. (2017), Krishnamurthy et al. (2017), Haug et al. (2018), and (Liang et al., 2018). These prior works generally present modeling contributions that are orthogonal (and in some cases complementary) to the contributions of this paper. There has also been a lot of recent work on neural semantic parsing, most of which is also orthogonal to (and could probably benefit from) our contributions (Dong and Lapata, 2016; Jia and Liang, 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017; Rabinovich et al., 2017). Recent attempts at dealing with the problem of spuriousness include Misra et al. (2018) and Guu et al. (2017).

Coverage has recently been used in machine translation (Tu et al., 2016) and summarization (See et al., 2017). There have also been many methods that use coverage-like mechanisms to give lexical cues to semantic parsers. Goldman et al. (2018)'s abstract examples is the most recent and related work, but the idea is also related to lexicons in pre-neural semantic parsers (Kwiatkowski et al., 2011).

0	*There is a tower with four blocks* `(box_exists (member_count_equals all_boxes 4))`
1	*Atleast one black triangle is not touching the edge* `(object_exists (black (triangle ((negate_filter touch_wall) all_objects))))`
2	*There is a yellow block as the top of a tower with exactly three blocks.* `(object_exists (yellow (top (object_in_box (member_count_equals all_boxes 3)))))`
3	*The tower with three blocks has a yellow block over a black block* `(object_count_greater_equals (yellow (above (black (object_in_box (member_count_equals all_boxes 3))))) 1)`

Table 7: Complexity of logical forms produced at different iterations, from iteration 0 to iteration 3; each logical form could not be produced at the previous iterations

8 Conclusion

We have presented a new technique for training semantic parsers with weak supervision. Our key insights are that lexical cues are crucial for guiding search during the early stages of training, and that the particulars of the approximate marginalization in maximum marginal likelihood have a large impact on performance. To address the first issue, we used a simple coverage mechanism for including lexicon-like information in neural semantic parsers that do not have lexicons. For the second issue, we developed an iterative procedure that alternates between statically-computed and dynamically-computed training signals. Together these two contributions greatly improve semantic parsing performance, leading to new state-of-the-art results on NLVR and WIKITABLEQUESTIONS. As these contributions are to the learning algorithm, they are broadly applicable to many models trained with weak supervision. One potential future work direction is investigating whether they extend to other structured prediction problems beyond semantic parsing.

Acknowledgments

We would like to thank Jonathan Berant and Noah Smith for comments on earlier drafts and Chen Liang for helping us with implementation details of MAPO. Computations on beaker.org were supported in part by credits from Google Cloud.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL'16*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Vaibhava Goel and William J Byrne. 2000. Minimum bayes-risk automatic speech recognition. *Computer Speech & Language*, 14(2):115–135.

Omer Goldman, Veronica Latcinnik, Udi Naveh, Amir Globerson, and Jonathan Berant. 2018. Weakly-supervised semantic parsing with abstract examples. In *ACL*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 177–183. Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. In *ECIR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Association for Computational Linguistics*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *ACL'16*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Jack Kiefer, Jacob Wolfowitz, et al. 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1512–1523. Association for Computational Linguistics.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–33.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis with generalization. *arXiv preprint arXiv:1807.02322*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *EMNLP*.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *ICLR*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1470–1480.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–32.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *ACL*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *ACL*.

David A Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 787–794. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2238–2249.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 217–223.

Hao Tan and Mohit Bansal. 2018. Object ordering with bidirectional matchings for visual reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ACL*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *ACL'17*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 2*, pages 1050–1055. AAAI Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI'05*.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

A Logical form language and lexicon for NLVR

Basic Types: `bool (t),box (b),object (o),shape (s),color (c),number (n)`

In the grammar and lexicon that follow we use the following placeholders,

quantifier ∈ {any, all, none}

comparator ∈ {equals, not_equals, lesser, lesser_equals, greater, greater_equals}

color ∈ {yellow, blue, black}

shape ∈ {square, triangle, circle}

size ∈ {big, medium, small}

location ∈ {above, below, top, left, right, bottom, corner, wall}

number ∈ {1...9}

A.1 Grammar

Constants

```
b -> all_boxes
c -> color_black,
     color_blue, color_yellow
n -> 1, 2, ..., 9
o -> all_objects
s -> shape_circle,
     shape_square, shape_triangle
```

Object filtering functions

```
<o,o> -> [location], [color],
 [shape], [size], same_color,
 same_shape, touch_object,
 touch_bottom, touch_top,
 touch_left, touch_right,
 touch_corner, touch_wall,
```

Box filtering functions

```
<b,<s,b>> ->
  member_shape_[quantifier]_equals
<b,<c,b>> ->
  member_color_[quantifier]_equals
<b,<n,b>> ->
  member_count_[comparator]
  member_color_count_[comparator],
  member_shape_count_[comparator]
<b,b> -> member_color_different,
 member_color_same,
 member_shape_different,
 member_shape_same
```

Assertion functions

```
<b,t> -> box_exists
<o,t> -> object_exists
<b,<n,t>> -> box_count_[comparator]
<o,<c,t>> ->
  object_color_[quantifier]_equals
<o,<s,t>> ->
  object_shape_[quantifier]_equals
<o,<n,t>> ->
  object_color_count_[comparator],
  object_shape_count_[comparator],
  object_count_[comparator]
```

Other functions

```
<b,o> -> object_in_box
<<o,o>,<o,o>> -> negate_filter
```

A.2 Lexicon for NLVR

there is a box → box_exists
there is a [other] → object_exists
box . . . [color] → color_[color]
box . . . [shape] → shape_[shape]
not → negate_filter
contains → object_in_box
touch . . . [location] → touch_[location]
[location] → [location]
[shape] → [shape]
[color] → [color]
[size] → [size]
[number] → [number]

B Logical form language and lexicon for WIKITABLEQUESTIONS

We use the language from Liang et al. (2018). For coverage, in addition to triggering productions for numbers, and column names and cell strings in the table, we use the following lexicon for coverage.

B.1 Lexicon for WIKITABLEQUESTIONS

at least → $\texttt{filter}_{\geq}$
[greater|larger|more] than → $\texttt{filter}_{\geq}$
at most → $\texttt{filter}_{\leq}$
no [greater|larger|more] than → $\texttt{filter}_{\leq}$
[next|below|after] → `next`
[previous|above|before] → `previous`
[first|top] → `top`
[last|bottom] → `bottom`
same → `same_as`
total → `sum`
difference → `diff`
average → `average`
[least|smallest|lowest|smallest] → `argmin`
[most|longest|highest|largest] → `argmax`
[what|when] . . . [last|least] → `min`
[what|when] . . . [first|most] → `max`
how many → `count`

Alignment over Heterogeneous Embeddings for Question Answering

Vikas Yadav, Steven Bethard, Mihai Surdeanu
University of Arizona, Tucson, AZ, USA
`{vikasy,bethard,msurdeanu}@email.arizona.edu`

Abstract

We propose a simple, fast, and mostly-unsupervised approach for non-factoid question answering (QA) called Alignment over Heterogeneous Embeddings (AHE). AHE simply aligns each word in the question and candidate answer with the most similar word in the retrieved supporting paragraph, and weighs each alignment score with the inverse document frequency of the corresponding question/answer term. AHE's similarity function operates over embeddings that model the underlying text at different levels of abstraction: character (FLAIR), word (BERT and GloVe), and sentence (InferSent), where the latter is the only supervised component. Despite its simplicity and lack of supervision, AHE obtains a new state-of-the-art performance on the "Easy" partition of the AI2 Reasoning Challenge (ARC) dataset (64.6% accuracy), top-two performance on the "Challenge" partition of ARC (34.1%), and top-three performance on the WikiQA dataset (74.08% MRR), outperforming many other complex, supervised approaches. Our error analysis indicates that alignments over character, word, and sentence embeddings capture substantially different semantic information. We exploit this with a simple meta-classifier that learns how much to trust the predictions over each representation, which further improves the performance of unsupervised AHE[1].

1 Introduction

Question - Which sequence of energy transformations occurs after a battery-operated flashlight is turned on?

1. *electrical → light → chemical*
2. *electrical →chemical → light*
3. *chemical → light → electrical*
4. **chemical → electrical → light**

Supporting paragraph(s): "a chemical cell converts chemical energy into electrical energy; a flashlight chemical energy to light energy"

Figure 1: A multiple-choice question from the ARC dataset with the correct answer in bold font. This question is answered correctly by our alignment method that relies on contextualized word embeddings that capture the correct sequence, and cannot be answered correctly when relying on uncontextualized embeddings.

The "deep learning tsunami"(Manning, 2015) has had a major impact on important natural language processing (NLP) applications such as question answering (QA). Many neural approaches for QA have been proposed in the past few years, with impressive results on several QA tasks (Seo et al., 2016; Wang and Jiang, 2016; Wang et al., 2017b; Tymoshenko et al., 2017; Xiong et al., 2016a; Wang et al., 2018; Radford et al., 2018; Li et al., 2018, inter alia). However, an undesired effect of this focus on neural approaches was that other methods have fallen out of focus, including strong unsupervised benchmarks that are necessary to highlight the true gains of supervised approaches. For instance, alignment approaches have received considerably less interest recently, despite their initial successes (Echihabi and Marcu, 2003; Surdeanu et al., 2011, inter alia). While a few recent efforts have adapted these alignment methods to operate over word representations (Kenter and De Rijke, 2015; Kim et al., 2017; Yadav et al., 2018), they generally underperfom supervised neural methods due to their underlying bag-of-word (BoW) assumptions and reliance on uncontextualized word representations such as GloVe (Pennington et al., 2014).

In this work we argue that alignment approaches are more meaningful today after the advent of *contextualized* word representations, which mitigate the above BoW limitations. For example, Figure 1 shows an example of a question from AI2's Reasoning Challenge (ARC) dataset (Clark et al., 2018),

---

[1] Code: `https://github.com/vikas95/AHE`

Proceedings of NAACL-HLT 2019, pages 2681–2691
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

which is not answered correctly by a state-of-the-art BoW alignment method (Yadav et al., 2018), but is correctly answered by our alignment approach when operating over Bidirectional Encoder Representations from Transformers (BERT) embeddings (Devlin et al., 2018).

We propose a simple, fast, and mostly-unsupervised approach for non-factoid QA called Alignment over Heterogeneous Embeddings (AHE). AHE uses an off-the-shelf information retrieval (IR) component to retrieve likely supporting paragraphs from a knowledge base (KB) given a question and candidate answer. Then AHE aligns each word in the question and candidate answer with the most similar word in the retrieved supporting paragraph, and weighs each alignment score with the inverse document frequency (IDF) of the corresponding question/answer term. AHE's overall alignment score is the sum of the IDF weighted scores of each of the question/answer term.

Importantly, AHE's alignment function operates over contextualized embeddings that model the underlying text at different levels of abstraction: character (FLAIR) (Akbik et al., 2018), word (BERT) (Devlin et al., 2018), and sentence (InferSent) (Conneau et al., 2017), where the latter is the only supervised component in the proposed approach. The different representations are combined through an ensemble approach that by default is unsupervised (using a variant of the NoisyOr formula), but can be replaced with a supervised meta-classifier.

The contributions of our work are the following:

1. To our knowledge, this is the first unsupervised alignment approach for QA that: (a) operates over contextualized embeddings, and (b) captures text at multiple levels of abstraction, including character, word, and sentence.
2. We obtain (near) state-of-the-art results (top three or higher) on three QA datasets: WikiQA (Yang et al., 2015) (74.08 mean reciprocal rank), ARC the Challenge partition (34.1% precision at 1 (P@1)) and ARC Easy (64.6 P@1). Our approach outperforms information retrieval methods, other unsupervised alignment approaches, and many supervised, neural approaches, despite the fact that it is mostly unsupervised and much simpler. Importantly, unlike many neural approaches, our results are robust across several datasets. Minimally, these results indicate that the work proposed here should be considered as a new, strong baseline for the task.
3. Our analysis indicates that alignments over character, word, and sentence embeddings capture substantially different semantic information. We highlight this complementarity with an oracle system that chooses the correct answer when it is proposed by any of the AHE's representations, which achieves 68% P@1 on ARC Challenge, 86% on ARC Easy, and 93.7% mean average precision (MAP) on WikiQA. We exploit this complementarity with a simple meta-classifier that learns when and how much to trust the predictions over each representation, which further improves the performance of unsupervised AHE.

2 Related Work

We highlight major trends in the field, and how our work compares with them. We focus mostly on non-factoid QA, which is usually implemented in two forms: multiple-choice QA such as AI2's Reasoning Challenge (ARC), where the answer must be selected from multiple candidates and (optionally) supported by explanatory texts extracted from external knowledge bases (Clark et al., 2018); or answer sentence selection, where candidate answer sentences are provided and the task is to select the sentences containing the correct answers (Yang et al., 2015). Alignment models have also been proposed for other types of QA, such as reading comprehension (RC) QA (Chakravarti et al., 2017). We believe AHE can be similarly extended to RC, but, in this work, we have limited our experiments to answer selection and multiple-choice QA tasks.

Most QA approaches today use neural, supervised methods. Most use stacked architectures usually coupled with attention mechanisms (He and Lin, 2016; Yin et al., 2015; Seo et al., 2016; Xiong et al., 2016b; Kumar et al., 2016; Tan et al., 2015; Wang et al., 2017a; Chen et al., 2016; Cheng et al., 2016; Golub and He, 2016). Some of these works also rely on structured knowledge bases (Zhong et al., 2018a; Ni et al., 2018) such as ConceptNet (Speer et al., 2017). Some approaches use query expansion methods in addition to the above methods (Musa et al., 2018; Nogueira and Cho, 2017; Ni et al., 2018). For example, Musa et al. (2018) used a sequence to sequence model (Sutskever et al., 2014) to generate an enhanced query for ARC which retrieves better supporting passages.

However, in general, all these approaches rely on annotated training data, and, some, on structured KBs, which are expensive to create (Jauhar et al., 2016). Further, as we demonstrate in Section 5, these methods tend to be tailored to a specific dataset and do not port well to other domains or even within different splits of the same dataset. In contrast, our method is mostly unsupervised and does not require training. Even then, our approach performs well on three distinct QA datasets, with top three performance in all.

Our work is inspired by previous efforts on using alignment methods for NLP (Echihabi and Marcu, 2003). Unsupervised alignment models have been proposed for several NLP tasks such as short text similarity (Kenter and De Rijke, 2015), answer phrase/sentence selection in reading comprehension (RC) (Chakravarti et al., 2017), document retrieval (Kim et al., 2017), etc. Other works have utilized word alignments as features in supervised models (Surdeanu et al., 2011; Wang and Ittycheriah, 2015). For example, Wang and Ittycheriah (2015) utilized the alignment of words between two questions as a feature in a feedforward neural network that matches similar FAQ questions. Recently, Yadav et al. (2018) showed that alignment methods remain competitive for non-factoid QA.

However, the majority of alignment models that rely on representation learning utilize uncontextualized word embeddings such as GloVe, coupled with other BoW models such as IBM Model 1 (Brown et al., 1993) for alignment (Kenter and De Rijke, 2015; Kim et al., 2017; Yadav et al., 2018). To our knowledge, we are the first to adapt these ideas to contextualized embeddings, which mitigates the BoW limitations of previous efforts (as shown in Figure 1). While contextualized representations have been shown to be extremely useful for multiple NLP tasks (Devlin et al., 2018; Peters et al., 2018; Howard and Ruder, 2018), our work is the first to apply them to an unsupervised alignment approach. Further, we show that different contextualized representations of text (character, word, sentence) capture complementary information, and combining them improves performance further.

3 Approach

The core component of our approach computes the score of a candidate answer by aligning two texts. For multiple-choice questions, the first text consists of the question concatenated with the candidate answer, and the second is a supporting paragraph such as the one shown in Figure 1, which consists of one or more sentences retrieved from a larger textual KB using an off-the-shelf IR system (Section 3.1). For answer selection tasks, the first text is the question and the second is the sentence that contains the candidate answer. Answer candidates are then sorted in descending order of their alignment scores. In both cases, the alignment approach operates over multiple contextualized embeddings that model the two texts at different levels of abstraction: character, word, and sentence. The overall architecture is illustrated in Figure 2. We detail the alignment method in §3.2, the multiple representations of text considered in §3.3, and the ensemble strategies over these representations in §3.4.

3.1 Retrieving Supporting Paragraphs

For multiple-choice question datasets such as ARC, we retrieve supporting information from external KBs using Lucene, an off-the-shelf IR system[2]. We use as query the question concatenated with the corresponding answer candidate, and BM25 (Robertson et al., 2009) as the ranking function[3]. For each query, we keep the top C Lucene documents, where each document consists of a sentence retrieved from the ARC corpus. Similar to our previous work (Yadav et al., 2018), we boost candidate answer terms by a factor of 3 while keeping question terms as it is in the BM25 ranking function. All texts were preprocessed by discarding the case of the tokens, removing the stop words from Lucene's list, and lemmatizing the remaining tokens using NLTK (Bird, 2006). For all experiments reported on the ARC dataset we used $C = 20$.

Here we also calculate the IDF of each query term q_i (required later during alignment):

$$idf(q_i) = \log \frac{N - docfreq(q_i) + 0.5}{docfreq(q_i) + 0.5} \quad (1)$$

where N is the number of documents (e.g., 14.3M for the ARC KB) and $docfreq(q_i)$ is the number of documents that contain q_i.

3.2 Alignment Algorithm

For representations that produce word embeddings (e.g., FLAIR, BERT, GloVe), we use the alignment algorithm in Figure 3. Our method computes the alignment score of each query token with every token in the given KB paragraph, using the cosine

---

[2] https://lucene.apache.org

[3] https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html

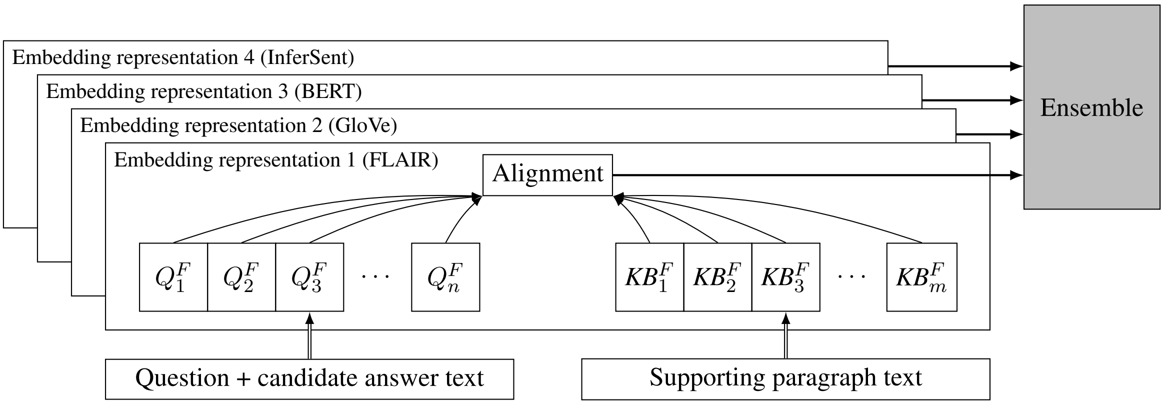


Figure 2: AHE architecture illustrated for the multiple-choice question setting. The left text consists with of the question concatenated with the answer candidate; the right text is a supporting paragraph retrieved from an external KB. The same alignment score is computed over multiple representations of text, and then aggregated through an ensemble model.

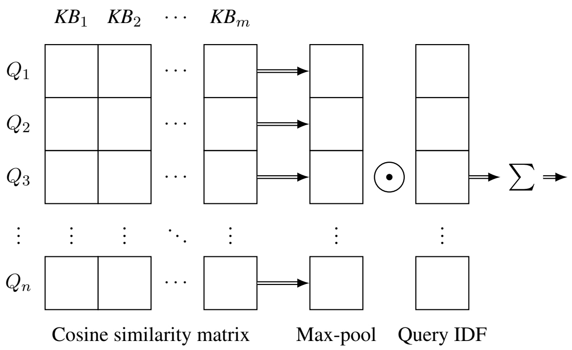


Figure 3: Alignment component of AHE, where a cosine similarity matrix is constructed by comparing token embeddings of input query tokens (Q_i) and supporting KB paragraph tokens (KB_j), and the maximal alignment cosine score for each input query token is weighted by its IDF.

similarity of the two embedding vectors. Then, a max-pooling layer over this cosine similarity matrix is used to retrieve the most similar token in the supporting passage for each query token. Lastly, this max-pooled vector of similarity scores is multiplied with the vector containing the IDF values of the query tokens and the resultant vector is summed to produce the overall alignment score s for the given query Q_a (formed from question Q and candidate answer a) and the supporting paragraph P_j:

$$s(Q_a, P_j) = \sum_{i=1}^{|Q_a|} idf(q_i) \cdot align(q_i, P_j) \qquad (2)$$

$$align(q_i, P_j) = \max_{k=1}^{|P_j|} cosSim(q_i, p_k) \qquad (3)$$

$$cosSim(q_i, p_k) = \frac{\vec{q_i} \cdot \vec{p_k}}{||\vec{q_i}|| \cdot ||\vec{p_k}||} \qquad (4)$$

where $\vec{q_i}$ and $\vec{p_k}$ are the embedding vectors of the terms q_i and p_k.

In addition to alignments over word-level embeddings, we include InferSent (Conneau et al., 2017), which generates sentence-level embeddings (see §3.3 for details). For InferSent, the alignment score between a query Q_a and a supporting paragraph P_j is computed as the dot product of the two corresponding sentence vectors, $\vec{Q_a}$ and $\vec{P_j}$, normalized using softmax over all candidate answers:

$$s(Q_a, P_j) = softmax(\vec{Q_a} \cdot \vec{P_j}) \qquad (5)$$

For ARC, the above alignment scores are computed for each supporting paragraph in the set of C paragraphs retrieved in §3.1. For WikiQA, this score is computed just for the sentence containing the candidate answer.

To aggregate the retrieved ARC paragraph scores (for ARC) into an overall score for the corresponding candidate answer, we consider:

Max: selects the maximum alignment score between all available paragraphs as the final score for candidate answer a:

$$S(cand_a) = \max_{j=1}^{C}(s(Q_a, P_j)) \qquad (6)$$

Weighted average: averages all available paragraph scores, using as weights the inverse IR ranks of the corresponding paragraphs:

$$S(cand_a) = \sum_{j=1}^{C} \frac{1}{j}(s(Q_a, P_j)) \qquad (7)$$

During tuning, we observed that the max strategy is better for ARC Challenge, while the weighted average is better for ARC Easy. We conjecture that this happens because Challenge questions require information that is sparser in the collection, and, thus, including more than the top paragraph tends to introduce noise.

3.3 Text Representations

AHE computes alignments over four different embedding representations that model the text at different levels of abstraction: character, word, and sentence (as detailed below). Although all these embeddings can be tuned for specific domains to improve performance, here we highlight the potential of publicly-available, pre-trained embeddings. Hence, we did not train embeddings on any domain specific corpus, and directly used off-the-shelf embeddings in all but one situation. The details of all four component embeddings of AHE are discussed below.

Character-based embeddings: We used the FLAIR contextual character language model of Akbik et al. (2018). They used long short-term memory (LSTM) networks that operate at character level over the entire text to generate character embeddings (in both forward and backward directions). Similar to them, to generate the embedding for token i, we concatenate the embedding from the forward LSTM for the character following the token, with the embedding from the backward LSTM for the character preceding the token:

$$\mathbf{w}_i^{FLAIR} := \begin{bmatrix} \mathbf{h}^f_{t_i+1-1} \\ \mathbf{h}^b_{t_i-1} \end{bmatrix} \quad (8)$$

where t_i is the character offset of the i^{th} token in the input text, and $\mathbf{h}$ is the corresponding LSTM's hidden state. We used the "mix-forward" and "mix-backward" pretrained models provided by the authors to produce two character embeddings, each of size 2048, resulting in word embeddings of size 4096.

Word-based embeddings: We incorporated two different word-based embeddings:

BERT – we used the Bidirectional Encoder Representations from Transformers (BERT) embedding model of Devlin et al. (2018). We concatenated the last four layers (as suggested by the authors[4]) of the BERT Large language model, where each layer has size 1024, summing up to size 4096 embeddings for each token:

$$\mathbf{w}_i^{BERT} := [Layer_{-1}, ..., Layer_{-4}] \quad (9)$$

GloVe – we also include GloVe embeddings (Pennington et al., 2014), under the hypothesis that these uncontextualized word embeddings will provide complementary information to the contextualized BERT embeddings. We used GloVe embeddings of size 300, trained over 840B tokens from Wikipedia, resulting in 2.2M words vocabulary.

Sentence-based embeddings: Lastly, we used InferSent, the sentence-based embeddings of Conneau et al. (2017). InferSent was originally trained on several natural language inference (NLI) datasets to generate the sentence representations that maximize the probability of correct inference. This model achieved poor performance on our QA tasks (see rows 8a in Table 1 and row 7a in Table 2).

Therefore, rather than using this NLI model, we trained InferSent on our data by maximizing the inference probability from the input query[5] to the supporting paragraph. We used the same number of supporting passages ($C = 20$) and the same scoring functions as explained in Section 3.2. We trained InferSent using batches of size 32, the Adam optimizer, learning rate = 0.001, and 50 epochs. We used max pooling over the token's LSTM hidden states to generate an overall sentence embedding. We tuned the sentence representation size on the development sets,[6] which resulted in 128 for WikiQA and 384 for ARC.

3.4 Aggregating Multiple Representations

We aggregate the scores of candidate answers over the four different embedding representations using an unsupervised variant of the NoisyOr formula:

$$NoisyOr_M(i) = 1 - (\prod_{m=0}^{M} (1 - \alpha^m * S_i^m)) \quad (10)$$

which computes the overall score for answer candidate i. M is the total number of representations (e.g., 4 in our case), and S_i^m is the score of answer candidate i under representation m. Lastly, α^m is a hyperparameter used to dampen peaky distributions of answer probabilities. We included this hyperparameter because we observed that InferSent produces a probability distribution over candidate answers where one answer tends to take most of the probability mass, and these scores dominate in the NoisyOr. Thus, the α^m weights are set to 1 for

---

[4] https://github.com/google-research/bert

[5] In ARC, the input query concatenates the question with a candidate answer.

[6] This was a light process that inspected only five possible values: 64, 128, 256, 384, and 512.

#	Supervised?	Type of KB	Model	Easy P@1	Challenge P@1
			Baselines		
1	No	text	Random baseline	25.02	25.02
2	No	text	AI2 IR Solver (Clark et al., 2018)	59.99	23.98
3	No	text	AI2 IR Solver (our implementation)	60.31	23.74
4	No	text	Sanity Check (Yadav et al., 2018)	58.36	26.56
5	No	text	AHE over GloVe	60.71	28.75
6	No	text	AHE over FLAIR	62.29	31.05
7	No	text	AHE over BERT	62.73	32.76
8a	No	text	AHE over InferSent (trained on NLI)	32.13	25.36
8b	Yes	text	AHE over *InferSent* (trained on ARC)	54.01	31.66
			Previous work		
9	Yes	text, structured	Tuple-Inf (Clark et al., 2018)	60.71	23.83
10	Yes	text	Decomp-att (Clark et al., 2018)	52.95	24.40
11	Yes	text, structured	DGEM (Clark et al., 2018)	58.97	27.11
12	Yes	text	BiDAF (for ARC) (Clark et al., 2018)	51.05	26.54
13	Yes	text, structured	KG2 (Zhang et al., 2018)	-	31.70
14	Yes	–	Bi-LSTM max-out (Mihaylov et al., 2018)	34.26	33.87
15	Yes	text, structured	NCRF++/match-LSTM (Musa et al., 2018)	52.22	33.20
16	Yes	text, structured	TriAN+f(dir)(cs)+f(ind)(cs) (Zhong et al., 2018b)	-	33.39
17	Yes	text, structured	ET-RR (Ni et al., 2018)	-	**36.56**
			Unsupervised AHE		
18	No	text	AHE (FLAIR+BERT)	63.45	33.87
19	No	text	AHE (FLAIR+BERT+GloVe)	64.60	31.06
20	Minimal	text	AHE (FLAIR+BERT+*InferSent*)	62.21	34.05
21	Minimal	text	AHE (FLAIR+BERT+GloVe+*InferSent*)	63.22	33.28
			Supervised AHE		
22	Yes	text	*AHE* (FLAIR+BERT+GloVe+*InferSent*)	65.19	33.70
23	Yes	text	*AHE* (FLAIR+BERT+GloVe+*InferSent*) with grade	**65.66**	34.47
			Oracle		
24	–	text	Oracle ensemble (FLAIR+BERT+GloVe+*InferSent*)	85.11	68.09

Table 1: Performance on the ARC dataset, measured using precision at 1 (P@1), on both the Easy and Challenge partitions. Italic font indicates which AHE components are supervised, e.g., *InferSent* is the InferSent model trained on ARC data; *AHE* is the AHE variant that uses the supervised meta-classifier ensemble. Line 8a shows performance of alignment over the original InferSent embeddings (trained on NLI datasets); line 8b shows performance when using *InferSent* embeddings trained on ARC training data. The "minimal" supervision configurations (lines 20 and 21) include the supervised *InferSent*, but use the unsupervised NoisyOr strategy for aggregation.

all representations with the exception of InferSent, for which we tuned its value to 0.2.

Of course, other types of aggregation are possible. To explore this space, we also implemented a supervised meta-classifier, which aims to learn the aggregation function directly from data. We implemented this multi-classifier as a feed forward network with two fully connected dense layers of hidden size 16 and K respectively, where K is the maximum number of candidate answers for the given dataset. The activation function of the first dense layer was tanh; we used a softmax in the second output layer. The input to this network was a vector of size $M \times K$. For example, for ARC this vector has a size $4 \times 5 = 20$. For WikiQA this vector has size $4 \times 22 = 88$. Each element in the input vector is the score of one candidate answer under a given representation. Additionally, for ARC we used an extra position in the input vector to indicate the grade of the corresponding exam question (provided in the dataset) with the intuition that the meta-classifier will learn to trust different representations for different grade levels.

4 Empirical Results

We evaluate AHE on two QA tasks:

AI2's Reasoning Challenge (ARC): this is a multiple-choice question dataset, containing science exam questions (Clark et al., 2018). The dataset is split in two partitions: Easy and Challenge, where the latter partition contains the more difficult questions that require reasoning. Each partition is split into train/development/test as follows: Easy contains 2251/570/2376 questions, and Challenge 1119/299/1172. Most of the questions have 4 answer choices, with only $< 1\%$ of all the questions having either 3 or 5 answer choices. ARC also includes a textual KB of 14.3M passages suitable for solving ARC questions. Note that we use solely this KB for retrieving supporting paragraphs,

#	Supervised?	Model	MAP	MRR
		Baselines		
1	No	Wgt Word Cnt (Yang et al., 2015)	50.99	51.32
2	Yes	LCLR (Yang et al., 2015)	59.93	60.86
3	No	Sanity Check (Yadav et al., 2018)	64.02	-
4	No	AHE over GloVe	63.40	65.39
5	No	AHE over FLAIR	64.91	66.51
6	No	AHE over BERT	65.13	66.40
7	Yes	AHE over *InferSent*	66.93	68.70
		Previous work		
8	Yes	CNN+Cnt (Yang et al., 2015)	65.20	66.52
9	Yes	RNN-1way (Jurczyk et al., 2016)	66.64	68.70
10	Yes	RNN-Attention_pool (Jurczyk et al., 2016)	67.47	68.92
11	Yes	CNN (avg + emb) (Jurczyk et al., 2016)	68.78	70.82
12	Yes	AP-CNN (dos Santos et al., 2016)	68.86	69.57
13	Yes	LSTM-att (Miao et al., 2016)	68.86	70.69
14	Yes	ABCNN (Yin et al., 2015)	69.21	71.08
15	Yes	Key-value memory network (Miller et al., 2016)	70.69	72.65
16	Yes	CubeCNN (He and Lin, 2016)	70.90	72.34
17	Yes	BiMPM (Wang et al., 2017b)	71.80	73.10
18	Yes	(Tymoshenko et al., 2017)	72.19	74.08
19	Yes	Compare-Aggregate (Wang and Jiang, 2016)	74.33	75.45
20	Yes	(Li et al., 2018)	**75.41**	**76.59**
		Unsupervised AHE		
21	No	AHE (FLAIR+BERT)	66.98	68.10
22	No	AHE (FLAIR+BERT+Glove)	67.31	68.53
23	Minimal	AHE (FLAIR+BERT+*InferSent*)	71.52	73.85
24	Minimal	AHE (FLAIR+BERT+Glove+*InferSent*)	71.77	74.08
		Supervised AHE		
25	Yes	*AHE* (FLAIR+BERT+Glove+*InferSent*)	72.13	74.64
		Oracle		
26	–	Oracle ensemble (FLAIR+BERT+Glove+*InferSent*)	93.71	95.49

Table 2: Performance on the WikiQA dataset, measured using mean average precision (MAP) and mean reciprocal rank (MRR). Italic font indicates which AHE components are supervised, e.g., *InferSent* is the InferSent model trained on WikiQA data; *AHE* is the AHE variant that uses the supervised meta-classifier ensemble. The "minimal" supervision configurations (lines 23 and 24) include the supervised *InferSent*, but use the unsupervised NoisyOr strategy for aggregation.

unlike many other approaches that use additional structured KBs such as ConceptNet (Zhong et al., 2018b) (see column 3 in Table 1).

WikiQA: is an open-domain answer selection dataset (Yang et al., 2015). It was constructed from Bing queries and candidate answer sentences from Wikipedia articles. It contains 1040/140/293 questions in train/development/test, and each question has an average of 9.6 candidate answer sentences.

Results and discussion: Tables 1 and 2 summarize the performance of multiple AHE variants, compared against several baselines and previous works, on two datasets. We draw several observations from these:

(1) The mostly unsupervised AHE, i.e., with the only supervised component being the InferSent embeddings, has solid and stable performance across the three datasets: best on ARC Easy, second best on ARC Challenge (see lines 18 – 21 in Table 1), and top three on WikiQA for MRR (see lines 21 – 24 in Table 2). We find these results encouraging: AHE outperforms many complex supervised neural approaches, including methods having multiple RNNs and stacked attention layers (Wang et al., 2017b; He and Lin, 2016; Miller et al., 2016; Yin et al., 2015; Miao et al., 2016; Musa et al., 2018; Mihaylov et al., 2018), despite the fact that it relies mostly on simple, unsupervised components.

(2) AHE ports well between different partitions (Easy and Challenge) of same dataset (ARC), unlike many of the previous approaches. For example, neural architectures that perform well on ARC Challenge perform worse than a simple IR baseline on ARC Easy (see, e.g., rows 14 and 15 in Table 1) or vice versa (see lines 9 – 12). This lack of portability occurs despite these models being trained/tested within the same partition in Table 1. To emphasize this issue, we explore more aggressive domain transfer settings in Section 5.2.

(3) Ablation analysis – The alignment performance from individual components of AHE are shown in the baseline blocks of Tables 1 and 2, while the combinations of AHE's components are shown in the corresponding unsupervised and supervised blocks, i.e.: rows 5–8 in table 1 and rows 4–7 in table 2 show performance from individual embeddings of AHE, while rows 18–23 and rows 21–25, respectively, show performance from combinations of AHE components. This comparison indicates that the combination of two or more embedding types are always better than individual embeddings. Further, we see that word embeddings such as GloVe are useful for ARC Easy but not for the Challenge partition of ARC (row 19). In contrast, sentence-level embeddings (InferSent) show the opposite behavior (row 20), suggesting that the more complex the task, the more high-level representations are required.

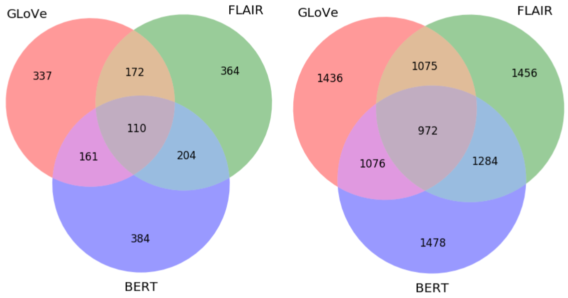
(4) The oracle system (line 24 in table 1 and line 26 in table 2) indicates that the different representations of text are to a large extent complementary: when selecting the correct answer when at least one of the representations proposes it, the oracle system achieves 85.1 P@1 on ARC Easy, 68.1 P@1 on ARC Challenge, and 93.71 MAP on WikiQA. The supervised AHE, which uses a feed-forward neural network to learn when to trust each representation demonstrates that (some of) this complementarity can be learned: the supervised AHE consistently outperforms its unsupervised counterpart, albeit by small amounts. Further, line 23 in Table 1 indicates that additional information about the questions (i.e., grade information) is beneficial, as it provides the meta-classifier more grounding on when to trust which representation. We analyze this complementarity further in Section 5.1.

5 Analysis

To explore the potential of AHE and further understand its individual components, we conducted the following analyses:

5.1 Complementarity of Representations

We calculated the overlap of questions answered correctly by each component of AHE to investigate the complementarity of the different representations. The results are visualized in Figure 4. For simplicity, the figure shows the number of questions answered correctly by the first three (unsupervised) components of AHE, but we found similar trends for the InferSent as well. As shown in the figure, the overlap between any two components is within the range [42 - 53]% in the Challenge partition (GloVe and BERT overlap = (161/384 = 42%), FLAIR and BERT overlap = (204/384 = 53%)) and [73 - 86]% in the Easy partition. Our current meta-classifier only begins to mine this complementarity, but it is limited because it has no information about the question and candidate answers (other than their scores). We conjecture that considerable performance improvements are possible when such a meta-classifier includes additional information such as question type, question encoding, etc. Our initial results that include grade information (line 23 in Table 1) support this hypothesis. We leave a further exploration of this direction as future work.



ARC Challenge ARC Easy

Figure 4: Overlap of correct questions answered by AHE models when they operate over different embeddings. This was a post-hoc analysis on the test partitions; we observed similar trends on training and development. This figure is best viewed in color.

5.2 Domain Transfer

As shown in Table 1 and discussed in the previous section, many supervised neural methods do not perform robustly across different partitions (Easy and Challenge) of the same ARC dataset, even though they were trained within each partition. This raises the question of how stable is their performance when trained/tested in different domains, which is closer to a real-world deployment scenario? To answer this question, we trained and tested two state-of-the-art neural models, BiLSTM Max-out (Mihaylov et al., 2018; Conneau et al., 2017) and BiMPM (Wang et al., 2017b), across three domains: ARC Easy, ARC Challenge, and WikiQA. We selected these two approaches because of they are end-to-end neural methods, and they achieve good performance on all datasets. Further, BiMPM is reminiscent of a supervised alignment method, since it computes the overall similarity of question and answers by aligning the

Train \Test	ARC Easy (P@1)	ARC Challenge (P@1)	WikiQA (MAP, MRR)
ARC Easy	34.26, 38.84	23.12, 24.10	(38.71, 40.51), (52.13, 53.87)
ARC Challenge	27.02, 36.17	33.87, 26.39	(39.05, 40.68), (40.09,41.48)
WikiQA	25.84, 38.40	24.32, 25.36	(67.40, 69.08), (69.20, 71.19)
Unsupervised AHE	64.60	33.87	(67.31, 68.53)

Table 3: Performance of two neural QA methods, BiLSTM Max-out and BiMPM, when trained/tested across datasets. The first value in each cell corresponds to BiLSTM Max-out, and the second to BiMPM. The last row contains the best unsupervised performance of AHE, which was *not* trained on any of these three datasets.

tokens' LSTM hidden states.

The results are summarized in Table 3. The table highlights that the performance of these systems varies considerably based on the training domain, even underperforming a random baseline in some configurations. In contrast, the unsupervised AHE does not require training, and obtains state-of-the-art, stable performance across the three datasets. This analysis suggests that future QA evaluations should consider domain transfer as another evaluation measure, to quantify the performance of QA systems under realistic scenarios.

5.3 Brief Qualitative Analysis

We manually analyzed the questions answered incorrectly by AHE and observed that many of the candidate answers were partially answering the questions. As shown in Figure 5, candidate answers 2 and 5 are partially answering the question, while candidate answers 1 and 3 provide topically relevant information. To select the correct answer in such complex questions, especially for short questions, a successful method would have to incorporate inference, e.g., recognizing process questions such as the one in the figure and coupling with it with a dedicated problem solving method (Clark et al., 2013). We leave the integration of inference methods with AHE as future work.

6 Conclusion

We proposed a simple, mostly-unsupervised alignment model for non-factoid QA, which operates over multiple contextualized embedding representations that model the text at different levels of abstraction. Despite its simplicity, our approach obtains good performance (top three or higher) that is stable across three QA datasets. Our analysis indicates that the different levels of abstraction (character, word, sentence) capture distinct semantics. We showed that this can be modeled with a meta-classifier that learns when and how much to trust the predictions over each representation, and that this has a beneficial impact on performance.

Question - how a water pump works?

1. *A large, electrically driven pump (electropump) for waterworks near the Hengsteysee , Germany.*
2. *A pump is a device that moves fluids (liquids or gases), or sometimes slurries , by mechanical action.*
3. *Pumps can be classified into three major groups according to the method they use to move the fluid: direct lift, displacement, and gravity pumps.*
4. **Pumps operate by some mechanism (typically reciprocating or rotary), and consume energy to perform mechanical work by moving the fluid.**
5. *Pumps operate via many energy sources, including manual operation, electricity, engines , or wind power .*

Figure 5: A question with correct answer in bold font from the WikiQA dataset, which was incorrectly answered by AHE, BiLSTM Max-out and BiMPM.

All in all, our work indicates that the first, and possibly best, investment in the design of a QA system should be on contextualized embeddings rather than custom, complex neural architectures.

When such embeddings are available, state-of-the-art performance that is competitive with modern neural approaches for QA can be obtained with simple alignment-based aggregation strategies. Minimally, our work should be regarded as a new, strong baseline for non-factoid question answering or answer sentence selection.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the World Modelers program, grant number W911NF1810014. Mihai Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Steven Bird. 2006. Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, COLING-ACL 2006, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Rishav Chakravarti, Jiri Navratil, and Cicero Nogueira dos Santos. 2017. Improved answer selection with pre-trained word embeddings. *arXiv preprint arXiv:1708.04326*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 37–42. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 16–23. Association for Computational Linguistics.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 474–483.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 820–827. IEEE.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM.

Sun Kim, Nicolas Fiorini, W John Wilbur, and Zhiyong Lu. 2017. Bridging the gap: Incorporating a semantic similarity measure for effectively mapping pubmed queries to documents. *Journal of biomedical informatics*, 75:122–127.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Weikang Li, Wei Li, and Yunfang Wu. 2018. A unified model for document-based question answering based on human-like reading strategy. In *AAAI*.

Christopher D Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Ryan Musa, Xiaoyan Wang, Achille Fokoue, Nicholas Mattei, Maria Chang, Pavan Kapanipathi, Bassem Makni, Kartik Talamadupula, and Michael Witbrock. 2018. Answering science exam questions using query rewriting with background knowledge. *arXiv preprint arXiv:1809.05726*.

Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2018. Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering. *CoRR*, abs/1808.09492.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *Computing Research Repository, abs/1602.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2).

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017. Ranking kernels for structures and embeddings: A hybrid preference and classification model. In *EMNLP*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017a. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016a. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016b. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Vikas Yadav, Rebecca Sharp, and Mihai Surdeanu. 2018. Sanity check: A strong alignment and information retrieval baseline for question answering.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Yuyu Zhang, Hanjun Dai, Kamil Toraman, and Le Song. 2018. Kg$\hat{2}$: Learning to reason science exam questions with contextual knowledge graph embeddings. *CoRR*, abs/1805.12393.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018a. Improving question answering by commonsense-based pre-training. *arXiv preprint arXiv:1809.03568*.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018b. Improving question answering by commonsense-based pre-training. *CoRR*, abs/1809.03568.

Bridging the Gap: Attending to Discontinuity in Identification of Multiword Expressions

Omid Rohanian[†*], **Shiva Taslimipoor**[†*], **Samaneh Kouchaki**[‡], **Le An Ha**[†], **Ruslan Mitkov**[†]
[†]Research Group in Computational Linguistics, University of Wolverhampton, UK
[‡]Institute of Biomedical Engineering, University of Oxford, UK
`{omid.rohanian, shiva.taslimi, l.a.ha, r.mitkov}@wlv.ac.uk`
`samaneh.kouchaki@eng.ox.ac.uk`

Abstract

We introduce a new method to tag Multiword Expressions (MWEs) using a linguistically interpretable language-independent deep learning architecture. We specifically target discontinuity, an under-explored aspect that poses a significant challenge to computational treatment of MWEs. Two neural architectures are explored: Graph Convolutional Network (GCN) and multi-head self-attention. GCN leverages dependency parse information, and self-attention attends to long-range relations. We finally propose a combined model that integrates complementary information from both, through a gating mechanism. The experiments on a standard multilingual dataset for verbal MWEs show that our model outperforms the baselines not only in the case of discontinuous MWEs but also in overall F-score.[1]

1 Introduction

Multiword expressions (MWEs) are linguistic units composed of more than one word whose meanings cannot be fully determined by the semantics of their components (Sag et al., 2002; Baldwin and Kim, 2010). As they are fraught with syntactic and semantic idiosyncrasies, their automatic identification remains a major challenge (Constant et al., 2017). Occurrences of discontinuous MWEs are particularly elusive as they involve relationships between non-adjacent tokens (e.g. ***put*** *one of the blue masks* ***on***).

While some previous studies disregard discontinuous MWEs (Legrand and Collobert, 2016), others stress the importance of factoring them in (Schneider et al., 2014). Using a CRF-based and a transition-based approach respectively, Moreau et al. (2018) and Al Saied et al. (2017) try to capture discontinuous occurrences with help from dependency parse information. Previously explored neural MWE identification models (Gharbieh et al., 2017) suffer from limitations in dealing with discontinuity, which can be attributed to their inherently sequential nature. More sophisticated architectures are yet to be investigated (Constant et al., 2017).

Graph convolutional neural networks (GCNs) (Kipf and Welling, 2017) and attention-based neural sequence labeling (Tan et al., 2018) are methodologies suited for modeling non-adjacent relations and are hence adapted to MWE identification in this study. Conventional GCN (Kipf and Welling, 2017) uses a global graph structure for the entire input. We modify it such that GCN filters convolve nodes of dependency parse tree on a per-sentence basis. Self-attention, on the other hand, learns representations by relating different parts of the same sequence. Each position in a sequence is linked to any other position with $O(1)$ operations, minimising maximum path (compared to RNN's $O(n)$) which facilitates gradient flow and makes it theoretically well-suited for learning long-range dependencies (Vaswani et al., 2017).

The difference in the two approaches motivates our attempt to incorporate them into a hybrid model with an eye to exploiting their individual strengths. Other studies that used related syntax-aware methods in sequence labeling include Marcheggiani and Titov (2017) and Strubell et al. (2018) where GCN and self-attention were separately applied to semantic role labelling.

Our contribution in this study, is to show for the first time, how GCNs can be successfully applied to MWE identification, especially to tackle discontinuous ones. Furthermore, we propose a novel architecture that integrates GCN with self-attention outperforming state-of-the-art. The resulting models not only prove superior to existing

---

*The first two authors contributed equally.

[1] The code is available on `https://github.com/omidrohanian/gappy-mwes`.

Proceedings of NAACL-HLT 2019, pages 2692–2698
Minneapolis, Minnesota, June 2 - June 7, 2019.

methods in terms of overall performance but also are more robust in handling cases with gaps.

2 Methodology

To specifically target discontinuity, we explore two mechanisms both preceding a Bi-LSTM: 1) a GCN layer to act as a syntactic ngram detector, 2) an attention mechanism to learn long-range dependencies.

2.1 Graph Convolution as Feature Extraction

Standard convolutional filters act as sequential ngram detectors (Kim, 2014). Such filters might prove inadequate in modeling complex language units like discontinuous MWEs. One way to overcome this problem is to consider non-sequential relations by attending to syntactic information in parse trees through the application of GCNs.

GCN is defined as a directed multi-node graph $G(V, E)$ where $v_i \in V$ and $(v_i, r, v_j) \in E$ are entities (words) and edges (relations) respectively. By defining a vector x_v as the feature representation for the word v, the convolution equation in GCN can be defined as a non-linear activation function f and a filter W with a bias term b as:

$$c = f(\sum_{i \in r(v)} W x_i + b) \qquad (1)$$

where $r(v)$ shows all words in relation with the given word v in a sentence, and c represents the output of the convolution.

Following Kipf and Welling (2017) and Schlichtkrull et al. (2017), we represent graph relations using adjacency matrices as mask filters for inputs. We derive associated words from the dependency parse tree of the target sentence. Since we are dealing with a sequence labelling task, there is an adjacency matrix representing relations among words (as nodes of the dependency graph) for each sentence. We define the sentence-level convolution operation with filter W_s and bias b_s as follows:

$$C_s = f(W_s X^T A + b_s) \qquad (2)$$

where X, A, and C are representation of words, adjacency matrix, and the convolution output, all at the level of sentence. The above formalism considers only one relation type, while depending on the application, multiple relations can be defined.

Kipf and Welling (2017) construct separate adjacency matrices corresponding to each relation type and direction. Given the variety of dependency relations in a parse tree (e.g. obj, nsubj, advcl, conj, etc), and per-sentence adjacency matrices, we would end up with an over-parametrised model in a sequence labeling task. In this work, we simply treat all relations equally, but consider only three types of relations: 1) the head to the dependents, 2) the dependents to the head, and 3) each word to itself (self-loops). The final output is obtained by aggregating the outputs from the three relations.

2.2 Self-Attention

Attention (Bahdanau et al., 2014) helps a model address the most relevant parts of a sequence through weighting. As attention is designed to capture dependencies in a sequence regardless of distance, it is complementary to RNN or CNN models where longer distances pose a challenge. In this work we employ multi-head self-attention with a weighting function based on scaled dot product which makes it fast and computationally efficient.

Based on the formulation of Transformer by Vaswani et al. (2017), in the encoding module an input vector x is mapped to three equally sized matrices K, Q, and V (representing key, query and value) and the output weight matrix is then computed as follows:

$$\text{Att}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \qquad (3)$$

The timing signal required for the self-attention to work is already contained in the preceding CNN layers alleviating the need for position encoding.

2.3 Model Architecture

The overall scheme of the proposed model, composed of two parallel branches, is depicted in Figure 1. We employ multi-channel CNNs as the step preceding self-attention. One channel is comprised of two stacked 1D CNNs and the other is a single 1D CNN. After concatenation and batch normalisation, a multi-head self attention mechanism is applied (Section 2.2).

Parallel to the self-attention branch, GCN learns a separate representation (Section 2.1). Since the GCN layer retains important structural information and is sensitive to positional data from the syntax tree, we consider it as a position-based approach. On the other hand, the self-attention layer

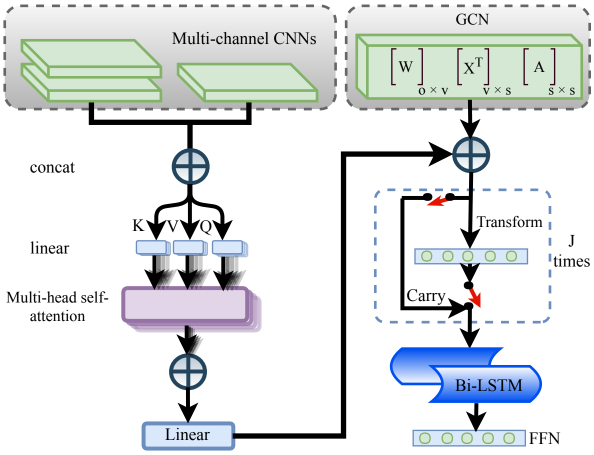


Figure 1: A hybrid sequence labeling approach integrating GCN (o: output dimension; v: word vectors dimension; s: sentence length) and Self-Attention.

is intended to capture long-range dependencies in a sentence. It relates elements of the same input through a similarity measure irrespective of their distance. We therefore regard it as a content-based approach. As these layers represent different methodologies, we seek to introduce a model that combines their complementary traits in our particular task.

Gating Mechanism. Due to the considerable overlap between the GCN and self-attention layers, a naive concatenation introduces redundancy which significantly lowers the learning power of the model. To effectively integrate the information, we design a simple gating mechanism using feed-forward highway layers (Srivastava et al., 2015) which learn to regulate information flow in consecutive training epochs. Each highway layer consists of a Carry (Cr) and a Transform (Tr) gate which decide how much information should pass or be modified. For simplicity Cr is defined as $1 - Tr$. We apply a block of J stacked highway layers (the section inside the blue dotted square in Figure 1). Each layer regulates its input x using the two gates and a feedforward layer H as follows:

$$y = Tr \odot H + (1 - Tr) \odot x \qquad (4)$$

where $\odot$ denotes the Hadamard product and Tr is defined as $\sigma(W_{Tr}x + b_{Tr})$. We set b_{Tr} to a negative number to reinforce carry behavior which helps the model learn temporal dependencies early in the training.

Our architecture bears some resemblance to Marcheggiani and Titov (2017) and Zhang et al. (2018) in its complementary view of GCN and BiLSTM. However there are some important differences. In these works, BiLSTM is applied prior to GCN in order to encode contextualised information and to enhance the teleportation capability of GCN. Marcheggiani and Titov (2017) stack a few BiLSTM layers with the idea that the resulting representation would enable GCN to consider nodes that are multiple hops away in the input graph. Zhang et al. (2018) use a similar encoder, however the model employs single BiLSTM and GCN layers, and the graph of relations is undirected.

In our work, we use pre-trained contextualised embeddings that already contain all the informative content about word order and disambiguation. We put BiLSTM on top of GCN, in line with how CNNs are traditionally applied as feature generating front-ends to RNNs. Furthermore, Marcheggiani and Titov (2017) use an edge-wise gating mechanism in order to down-weight uninformative syntactic dependencies. This method can mitigate noise when parsing information is deemed noisy, however in Zhang et al. (2018) it caused performance to drop. Given our low-resource setting, in this work we preferred not to potentially down-weight contribution of individual edges, therefore treating them equally. We rely on gating as the last step when we combine GCN and self-attention.

3 Experiments

Data. We experiment with datasets from the shared task on automatic identification of verbal Multiword Expressions (Ramisch et al., 2018). The datasets are tagged for different kinds of verbal MWEs including idioms, verb particle constructions, and light verb constructions among others. We focus on annotated corpora of four languages: French (FR), German (DE), English (EN), and Persian (FA) due to their variety in size and proportion of discontinuous MWEs. Tags in the datasets are converted to a variation of IOB which includes the tags `B` (beginning of MWEs), `I` (other components of MWEs), and `O` (tokens outside MWEs), with the addition of `G` for arbitrary tokens in between the MWE components e.g. $make_{\texttt{[B]}}$ $important_{\texttt{[G]}}$ $decisions_{\texttt{[I]}}$.

ELMo. In our experiments, we make use of ELMo embeddings (Peters et al., 2018) which are contextualised and token-based as opposed to

		All		Discontinuous			
		Token-based	MWE-based		MWE-based		
L	model	F	F	%	P	R	F
EN	baseline	41.37	35.38	32	24.44	10.48	14.67
	GCN-based	39.78	39.11		39.53	16.19	22.97
	Att-based	33.33	31.79		46.88	14.29	21.90
	H-combined	41.63	**40.76**		63.33	18.10	**28.15**
DE	baseline	62.27	57.17	43	69.50	45.37	54.90
	GCN-based	65.48	**61.17**		65.19	47.69	55.08
	Att-based	61.20	58.19		67.86	43.98	53.37
	H-combined	63.80	60.71		68.59	49.54	**57.53**
FR	baseline	76.62	72.16	43	75.27	52.04	61.54
	GCN-based	79.59	75.15		79.58	56.51	66.09
	Att-based	78.21	74.23		71.49	60.59	65.59
	H-combined	80.25	**76.56**		77.94	59.11	**67.23**
FA	baseline	88.45	86.50	14	67.76	55.88	61.29
	GCN-based	87.78	86.42		78.72	54.41	64.35
	Att-based	87.55	84.20		62.32	63.24	62.77
	H-combined	88.76	**87.15**		75.44	63.24	**68.80**

Table 1: Model performance (P, R and F) for development sets for all MWE and only discontinuous ones (%: proportion of discontinuous MWES)

type-based word representations like `word2vec` or `GLoVe` where each word type is assigned a single vector. Token-based embeddings better reflect the syntax and semantics of each word in its context compared to traditional type-based ones. We use the implementation by Che et al. (2018) to train ELMo embeddings on our data.

Validation. In the validation phase, we start with a strong baseline which is a CNN + Bi-LSTM model based on the top performing system in the VMWE shared task (Taslimipoor and Rohanian, 2018). Our implemented baseline differs in that we employ ELMo rather than `word2vec` resulting in a significant improvement. We perform hyper-parameter optimisation and make comparisons among our systems, including GCN + Bi-LSTM (GCN-based), CNN + attention + Bi-LSTM (Att-based), and their combination using a highway layer (H-combined) in Table 1.

4 Evaluation and Results

Systems are evaluated using two types of precision, recall and F-score measures: strict MWE-based scores (every component of an MWE should be correctly tagged to be considered as true positive), and token-based scores (a partial match between a predicted and a gold MWE would be considered as true positive). We report results for all MWEs as well as discontinuous ones specifically.

According to Table 1, GCN-based outperforms Att-based and they both outperform the strong baseline in terms of MWE-based F-score in three out of four languages. Combining GCN with attention using highway networks results in further improvements for EN, FR and FA. The H-combined model consistently exceeds the baseline for all languages. As can be seen in Table 1, GCN and H-combined models each show significant improvement with regard to discontinuous MWEs, regardless of the proportion of such expressions.

In Table 2 we show the superior performance (in terms of MWE-based F-score) of our top systems on the test data compared to the baseline and state-of-the-art systems, namely, ATILF-LLF (Al Saied et al., 2017) and SHOMA (Taslimipoor and Rohanian, 2018). GCN works the best for discontinuous MWEs in EN and FA, while H-combined outperforms based on results for all MWEs except for FA. The findings are further discussed in Section 5.

5 Discussion and Analysis

The overall results confirm our assumption that a hybrid architecture can mitigate errors of individual models and bolster their strengths. To demonstrate the effectiveness of the models in detecting discontinuous MWEs, in Figure 2 we plot their performance for FR and EN given a range of different gap sizes. As an ablation study, we show the results for the baseline, GCN-based, Att-based only, as well as H-combined models. GCN and Att-based models each individually outperform the baseline, and the combined model clearly improves the results further.

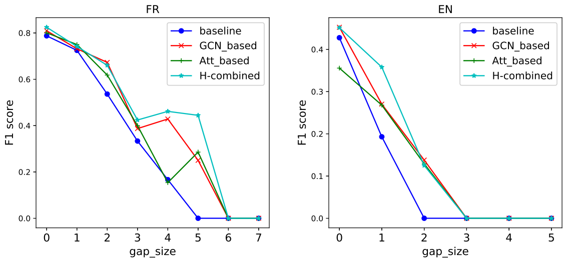


Figure 2: Model performance given different gap sizes

The example in Figure 3 taken from the English dataset demonstrates the way GCN considers relations between non-adjacent tokens in the sentence. Our baseline is prone to disregarding these links. Similar cases captured by both GCN and H-combined (but not the baseline) are ***take*** *a final* ***look***, ***picked*** *one* ***up***, and ***cut*** *yourself* ***off***.

	All | Discontinuous			
	EN	DE	FR	FA
baseline	33.01 | 16.53	54.12 | 53.94	67.66 | 58.70	**81.62** | 61.73
GCN-based	36.27 | **24.15**	56.96 | 54.87	70.79 | 59.95	81.00 | **62.35**
H-combined	**41.91** | 22.73	**59.29** | **55.00**	**70.97** | **63.90**	80.04 | 61.90
ATILF-LLF	31.58 | 09.91	54.43 | 40.34	58.60 | 51.96	77.48 | 53.85
SHOMA	26.42 | 01.90	48.71 | 40.12	62.00 | 51.43	78.35 | 56.10

Table 2: Comparing the performance of the systems on test data in terms of MWE-based F-score

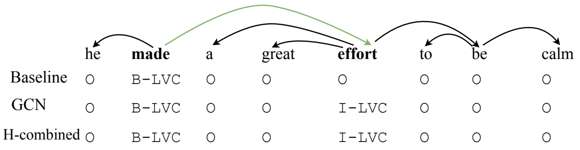


Figure 3: Sample sentence with a discontinuous occurrence of an English MWE, *make an effort*.

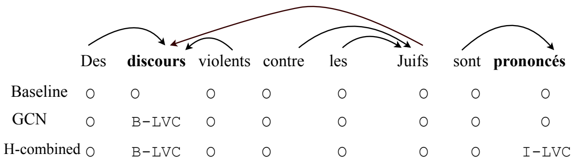


Figure 4: Example sentence with a discontinuous occurrence of a French MWE, *prononcer un discours* 'to make a speech'.

In more complicated constructs where syntactic dependencies might not directly link all constituents, GCN alone is not always conducive to optimal performance. In Figure 4, the French sentence is in the passive form and MWE parts are separated by 5 tokens. This is an MWE skipped by GCN but entirely identified by the H-combined model.

It is important to note that model performance is sensitive to factors such as percentage of seen expressions and variability of MWEs (Pasquer et al., 2018). In FA for instance, 67% of the MWEs in the test set are seen at training time, making them easy to be captured by the baseline (Taslimipoor et al., 2018). Furthermore, only 21% of MWEs in FA and 15% in EN are discontinuous as opposed to 44% in FR and 38% in DE. In this case, a sequential model can already learn the patterns with high accuracy and the potential of a GCN and self-attention is not fully exploited.

Also in DE, a sizable portion of MWEs are verbal idioms (VIDs) which are known for their lexico-syntactic fixedness and prevalence of tokens that lack a standalone meaning and occur only in a limited number of contexts (also known as cranberry words). Furthermore, MWEs in the Persian dataset are all Light Verb Constructions (LVCs), which can be modelled using lexical semantic templates (Megerdoomian, 2004). For such MWEs, our models compete with strong sequential baselines.

6 Conclusion and Future Work

In this paper, we introduced the application of GCN and attention mechanism to identification of verbal MWEs and finally proposed and tested a hybrid approach integrating both models. Our particular point of interest is discontinuity in MWEs which is an under-explored area. All the individual and combined models outperform state-of-the-art in all considered criteria. In future, we will further develop our system using structured attention (Kim et al., 2017) and try to improve the accuracy of parsers in multi-tasking scenarios.

References

Hazem Al Saied, Matthieu Constant, and Marie Candito. 2017. The ATILF-LLF system for Parseme shared task: a transition-based verbal multiword expression tagger. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 127–132, Valencia, Spain. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In *Handbook of Natural Language Processing, second edition.*, pages 267–292. CRC Press.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages

55–64, Brussels, Belgium. Association for Computational Linguistics.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke Van Der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Multiword expression processing: A survey. *Computational Linguistics*, 43(4):837–892.

Waseem Gharbieh, Virendra Bhavsar, and Paul Cook. 2017. Deep learning models for multiword expression identification. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 54–64, Vancouver, Canada. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Joël Legrand and Ronan Collobert. 2016. Phrase representations for multiword expressions. In *Proceedings of the 12th Workshop on Multiword Expressions (MWE 2016)*, Berlin, Germany.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Karine Megerdoomian. 2004. A semantic template for light verb constructions. In *Proceedings of the First Workshop on Persian Language and Computers. Tehran University, Iran*, pages 25–26.

Erwan Moreau, Ashjan Alsulaimani, Alfredo Maldonado, and Carl Vogel. 2018. CRF-Seq and CRF-DepTree at PARSEME Shared Task 2018: Detecting verbal MWEs using sequential and dependency-based approaches. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 241–247. Association for Computational Linguistics.

Caroline Pasquer, Agata Savary, Jean-Yves Antoine, and Carlos Ramisch. 2018. Towards a variability measure for multiword expressions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 426–432.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (Long Papers)*, volume 1, pages 2227–2237.

Carlos Ramisch, Silvio Cordeiro, Agata Savary, Veronika Vincze, Verginica Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, et al. 2018. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 222–240.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '02, pages 1–15, London, UK. Springer-Verlag.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *TACL*, 2:193–206.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *AAAI Conference on Artificial Intelligence*.

Shiva Taslimipoor and Omid Rohanian. 2018. SHOMA at Parseme shared task on automatic identification of VMWEs: Neural multiword expression tagging with high generalisation. *arXiv preprint arXiv:1809.03056*.

Shiva Taslimipoor, Omid Rohanian, Ruslan Mitkov, and Afsaneh Fazly. 2018. Identification of multiword expressions: A fresh look at modelling and evaluation. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword expressions at length and in depth.*

Extended papers from the MWE 2017 workshop, pages 299–317. Language Science Press, Berlin, Germany.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Incorporating Word Attention into Character-Based Word Segmentation

Shohei Higashiyama[1], **Masao Utiyama**[1], **Eiichiro Sumita**[1],
Masao Ideuchi[2], **Yoshiaki Oida**[2], **Yohei Sakamoto**[2], **Isaac Okada**[2]
[1]National Institute of Information and Communications Technology, Kyoto, Japan
[2]FUJITSU LIMITED, Tokyo, Japan
`{shohei.higashiyama, mutiyama, eiichiro.sumita}@nict.go.jp,`
`{ideuchi.masao,oida.yoshiaki,yohei.sakamoto,isaac-okada}`
`@jp.fujitsu.com`

Abstract

Neural network models have been actively applied to word segmentation, especially Chinese, because of the ability to minimize the effort in feature engineering. Typical segmentation models are categorized as character-based, for conducting exact inference, or word-based, for utilizing word-level information. We propose a character-based model utilizing word information to leverage the advantages of both types of models. Our model learns the importance of multiple candidate words for a character on the basis of an attention mechanism, and makes use of it for segmentation decisions. The experimental results show that our model achieves better performance than the state-of-the-art models on both Japanese and Chinese benchmark datasets.[1]

1 Introduction

Word segmentation is the first step of natural language processing (NLP) for most East Asian languages, such as Japanese and Chinese. In recent years, neural network models have been widely applied to word segmentation, especially Chinese, because of their ability to minimize the effort in feature engineering. These models are categorized as character-based or word-based. Word-based models (Zhang et al., 2016; Cai and Zhao, 2016; Cai et al., 2017; Yang et al., 2017) directly segment a character sequence into words and can easily achieve the benefits of word-level information. However, these models cannot usually conduct exact inference because of strategies, such as beam-search decoding and constraints of maximum word length, which are necessary as the number of candidate segmentations increases exponentially with the sentence length. On the other hand, character-based models (Zheng et al., 2013; Mansur et al., 2013; Pei et al., 2014; Chen et al., 2015a) treat word segmentation as sequence labeling. These models typically predict optimal label sequences while considering adjacent labels.

Limited efforts have been devoted to leveraging the advantages of both types of models, such as utilizing word information and conducting exact inference, which are complementary characteristics. In particular, the candidate word information for a character is beneficial to disambiguate word boundaries because a character in the sentence has multiple candidate words that contain the character. For example, there are three or four candidate words for characters x_3, x_4 and x_5 in a sentence $x_{1:5}$ in Figure 1. A feasible solution to develop a model with both characteristics is to incorporate word information into a character-based framework. An example of such work is that of Wang and Xu (2017). They concatenated embeddings of a character and candidate words and used it in their convolutional neural network (CNN)-based model. They treated candidate words equivalently, although the plausibility of a candidate word differs in the context of a target character.

In this paper, we propose a character-based word segmentation model that utilizes word information. Our model is based on a BiLSTM-CRF architecture that has been successfully applied to sequence labeling tasks (Huang et al., 2015; Chen et al., 2015b). Differing from the work of Wang and Xu (2017), our model learns and distinguishes the importance of candidate words for a character in a context, by applying an attention mechanism (Bahdanau et al., 2015).

Our contributions are as follows:

- We introduce word information and an attention mechanism into a character-based word segmentation framework, to distinguish and leverage the importance of candidate words

---

[1] We have released our code at `https://github.com/shigashiyama/seikanlp`.

Proceedings of NAACL-HLT 2019, pages 2699–2709
Minneapolis, Minnesota, June 2 - June 7, 2019.

Sentence:

$x_{1:5}$ = 彼は日本人 ⟨*kare wa nihonjin*⟩ (He is a Japanese.)

Candidate words $\{w_j\}$ for characters $\{x_i\}$ in the sentence:

i	$x_i \diagdown w_j$ j	1 彼 ⟨*kare*⟩ (he)	2 は ⟨*wa*⟩ (NOM)	3 日 ⟨*hi*⟩ (day)	4 本 ⟨*hon*⟩ (book)	5 人 ⟨*hito*⟩ (person)	6 日本 ⟨*nihon*⟩ (Japan)	7 本人 ⟨*honnin*⟩ (the person)	8 日本人 ⟨*nihonjin*⟩ (Japanese)	No. of candidate words
1	彼	1	0	0	0	0	0	0	0	1
2	は	0	1	0	0	0	0	0	0	1
3	日	0	0	1	0	0	1	0	1	3
4	本	0	0	0	1	0	1	1	1	4
5	人	0	0	0	0	1	0	1	1	3

Figure 1: An example of candidate words $w_{1:8}$ retrieved from a vocabulary for a sentence $x_{1:5}$. Strings in angle brackets "⟨⟩" and in parentheses "()" respectively indicate (typical) readings and English translations of words. The value in each (i, j) represents whether the i-th character is contained by the j-th word (i.e., δ_{ij} in Eq. (4)).

in different contexts.

- We empirically reveal that accurate attention to proper candidate words leads to correct segmentations.
- Our model outperforms the state-of-the-art word segmentation models on both Japanese and Chinese datasets.

2 Task Definition

Word segmentation can be regarded as a character-level sequence labeling task. Given a sentence $x = x_{1:n} := (x_1, \dots, x_n)$ of length n, each character x_i will be assigned a segmentation label y_i of tag set $\mathcal{T}$, and a label sequence $y = y_{1:n}$ will be predicted. We employ tag set $\mathcal{T} = \{\text{B}, \text{I}, \text{E}, \text{S}\}$, where B, I and E, respectively, represent the beginning, inside and end of a multi-character word, and S represents a single character word (Xue, 2003).

3 Baseline Model

We use a BiLSTM-CRF architecture for our baseline model. The model consists of a character embedding layer, recurrent layers based on long short-term memory (LSTM) and a conditional random fields (CRF) layer as in Figure 2.

Character Embedding Layer Let $\mathcal{V}_c$ be a character vocabulary. Each character in a given sentence is transformed into a character embedding $\mathbf{e}^c$ of d_c-dimensional vector by a lookup operation that retrieves the corresponding column of the embedding matrix $E_c \in \mathbb{R}^{d_c \times |\mathcal{V}_c|}$.

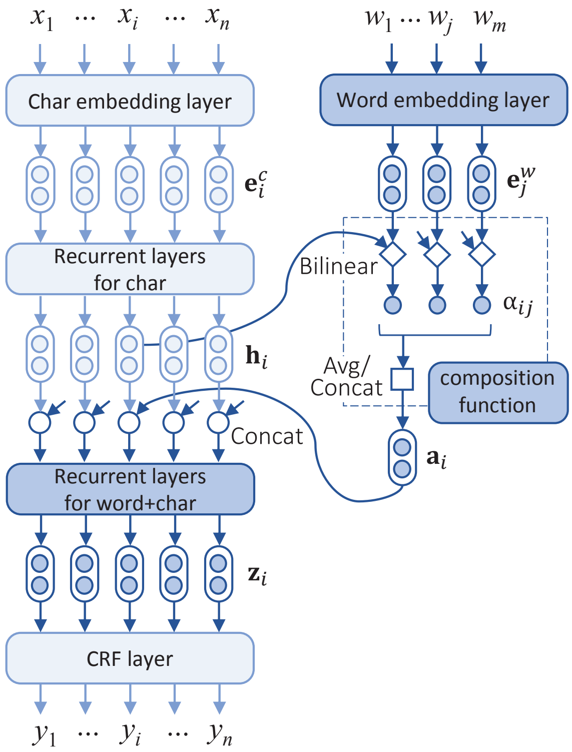


Figure 2: Architecture of our proposed model, which comprises the common components to the baseline model (light blue) and additional ones (dark blue).

Recurrent Layers for Character Representation A sequence of character embeddings $\mathbf{e}^c_{1:n}$ is fed into a recurrent neural network (RNN) to derive contextualized representations $\mathbf{h}_{1:n}$, which we call *character context vectors*. We adopt a stacked (multi-layer) and bidirectional variant of an LSTM (Hochreiter and Schmidhuber, 1997) network, which addresses the issue of learning long-term dependencies and the gradient vanishing problem.

Hidden vectors $\mathbf{h}^{(l)}_{1:n}$ of l-th bidirectional LSTM (BiLSTM) layer are calculated by forward LSTM

(LSTM_f) and backward LSTM (LSTM_b):

$$\begin{aligned}\mathbf{h}_i^{(l)} &= \mathrm{BiLSTM}(\mathbf{h}_{1:n}^{(l-1)}, i) \qquad (1)\\ &:= \mathrm{LSTM}_f(\mathbf{h}_{1:n}^{(l-1)}, i)\\ &\quad \oplus \mathrm{LSTM}_b(\mathbf{h}_{n:1}^{(l-1)}, n-i+1)\ ,\end{aligned}$$

where $\mathbf{h}_i^{(0)} = \mathbf{e}_i^c$ and $\oplus$ denotes a concatenation operation, and $\mathbf{h}_{n:1}^{(l-1)}$ denotes the reversed sequence of the original vectors $\mathbf{h}_{1:n}^{(l-1)}$.

More concretely, each forward or backward LSTM calculates hidden vectors $\mathbf{h}_{1:n}$ from an input sequence $\mathbf{v}_{1:n}$ of d_v-dimensional vectors as follows:

$$\begin{aligned}\mathbf{h}_i &= \mathrm{LSTM}(\mathbf{v}_{1:n}, i) := \mathbf{o}_i \odot \tanh(\mathbf{c}_i)\ ,\\ \mathbf{c}_i &= \mathbf{i}_i \odot \mathbf{t}_i + \mathbf{f}_i \odot \mathbf{c}_{i-1},\\ \mathbf{g}_i &= \sigma(W_g \mathbf{v}_i + U_g \mathbf{h}_{i-1} + \mathbf{b}_g)\ ,\\ \mathbf{t}_i &= \tanh(W_t \mathbf{v}_i + U_t \mathbf{h}_{i-1} + \mathbf{b}_t)\ ,\end{aligned}$$

where σ is sigmoid function, $\odot$ denotes element-wise multiplication, $\mathbf{g}$ indicates an input gate $\mathbf{i}$, a forget gate $\mathbf{f}$ or an output gate $\mathbf{o}$, $W_g, U_g \in \mathbb{R}^{d_r \times d_v}$ and $\mathbf{b}_g \in \mathbb{R}^{d_r}$ are trainable parameters for each gate $g \in \{i, f, o\}$, and d_r is a hyperparameter.

CRF Layer A character context vector $\mathbf{h}_i$ is mapped into a $|\mathcal{T}|$-dimensional vector representing scores of segmentation labels:

$$\mathbf{s}_i = W_s \mathbf{h}_i + \mathbf{b}_s\ ,$$

where $W_s \in \mathbb{R}^{|\mathcal{T}| \times 2d_r}$ and $b \in \mathbb{R}^{|\mathcal{T}|}$ are trainable parameters. Following previous sequence labeling work (Collobert et al., 2011), we introduce a CRF (Lafferty et al., 2001) layer, which has a transition matrix $A \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}$ to give transition scores of adjacent labels. Thus, the score of a label sequence $y = y_{1:n}$ for a sentence $x = x_{1:n}$ is calculated as follows:

$$\mathrm{score}(x, y; \theta) = \sum_{i=1}^{n} (A_{y_{i-1}, y_i} + \mathbf{s}_i[y_i])\ ,$$

where θ denotes all the parameters and $\mathbf{s}[y]$ indicates the dimension of a vector $\mathbf{s}$ corresponding to a label y. We can find the best label sequence $y^\star$ by maximizing the sentence score:

$$y^\star = \mathrm{argmax}_{y \in \mathcal{T}^n}\ \mathrm{score}(x, y; \theta)\ . \qquad (2)$$

Training Objective During training, parameters θ of the network are learned by minimizing the negative log likelihood over all the sentences in training data $\mathcal{D}$ w.r.t θ:

$$\begin{aligned}\min_\theta \quad &- \sum_{(x,y)\in\mathcal{D}} \log p(y|x,\theta)\ ,\\ p(y|x,\theta) &= \frac{\mathrm{score}(x, y; \theta)}{\sum_{y'} \mathrm{score}(x, y'; \theta)}\ . \qquad (3)\end{aligned}$$

Note that the Viterbi algorithm can be used for efficient calculation of the probability of a label sequence in Eq. (3) similarly to decoding in Eq. (2).

4 Proposed Model

To disambiguate word boundaries more effectively, we integrate word information into the character-based framework. More concretely, we transform embeddings of multiple candidate words for each character into a fixed size word vector, which we call a *word summary vector*, by a *word feature composition function*. We show the architecture of our model in Figure 2. In addition to the layers of the baseline model, the model comprises a word embedding layer, a word feature composition function, and additional recurrent layers.

Word Embedding Layer Given a character sequence $x = x_{1:n}$, we search for all words corresponding to subsequences of the input sequence from a word vocabulary $\mathcal{V}_w$ within a maximum word length. Then, we obtain a unique list[2] $\mathcal{W}_x$ of candidate words of size m. For example, for a given sentence $x_{1:5}$ in Figure 1, candidate words $\{w_1, \cdots, w_8\}$ will be found. Each word $w \in \mathcal{W}_x \subseteq \mathcal{V}_w$ is transformed into a d_w-dimensional vector $\mathbf{e}^w$ by the embedding matrix $E_w \in \mathbb{R}^{d_w \times |V_w|}$.

We can construct a word vocabulary to search candidate words by using external dictionaries or auto-segmented texts processed by any segmenter. Regarding the construction method used in our experiments, refer to §5.1.

Composition Functions of Word Features For a character x_i, the embeddings of all the candidate words that contain it are aggregated into a word summary vector $\mathbf{a}_i$ by a composition function. We introduce two attention-based composition functions, weighted average (WAVG) and

---

[2] List indicates set where each element has a unique number from 1 to the size of the set.

weighted concatenation (WCON), which enable a model to pay more or less attention according to the importance of candidate words.

Both functions calculate the importance score u_{ij} from a character x_i to a word w_j in $\mathcal{W}_x$ by a bilinear transformation, which indicates the interaction between the character context vector $\mathbf{h}_i$ and the word embedding $\mathbf{e}_j^w$. Then the weight $\alpha_{ij} \in [0, 1]$ is obtained by a softmax operation to normalize the scores:

$$\begin{aligned} u_{ij} &= \mathbf{h}_i^T W_a \mathbf{e}_j^w \ , \\ \alpha_{ij} &= \frac{\delta_{ij} \exp(u_{ij})}{\sum_{k=1}^{m} \delta_{ik} \exp(u_{ik})} \ , \end{aligned} \tag{4}$$

where $W_a \in \mathbb{R}^{2d_r \times d_w}$ is a trainable parameter. For simplification of equations, we introduce an indicator variable $\delta_{ij} \in \{0, 1\}$ that indicates whether the character x_i is included in the word w_j as Figure 1 illustrates.

Next, WAVG and WCON calculate a word summary vector $\mathbf{a}_i$ as the weighted average and the weighted concatenation of word embeddings, respectively:

$$\mathbf{a}_i = \mathrm{WAVG}(x_i, \{w_j\}_{j=1}^m) = \sum_{j=1}^{m} \alpha_{ij} \mathbf{e}_j^w \ , \tag{5}$$

$$\mathbf{a}_i = \mathrm{WCON}(x_i, \{w_j\}_{j=1}^m) = \bigoplus_{l=1}^{L} \alpha_{i,i_l} \mathbf{e}_{i_l}^w \ , \tag{6}$$

where $\{w_j\} = \mathcal{W}_x$ and $\bigoplus(\cdot)$ indicates concatenation of given arguments. Let K be the maximum word length, $L = \sum_{k=1}^{K} k$, and i_l for the character x_i denotes the corresponding index in $\mathcal{W}_x$ of l-th words w'_l in the list $\{w'_1, \dots, w'_L\} = \bigcup_{k=1}^{K} \bigcup_{p=-k+1}^{0} \{x_{i+p:i+p+k-1}\}$. If $w'_l \notin \mathcal{V}_w$, we use a zero vector as the l-th argument in Eq. (6). For example, if $K = 3$, WCON concatenates embeddings of words corresponding to x_i (length 1), $x_{i-1:i}$, $x_{i:i+1}$ (length 2), $x_{i-2:i}$, $x_{i-1:i+1}$ and $x_{i:i+2}$ (length 3) in this order, into a single vector for the character x_i. WAVG and WCON finally output a summary vector of size d_w and Ld_w, respectively. Note that we use zero vector as a summary vector if no candidate words are found for a character.

We also use two more variants of composition functions without the attention mechanism, the average function (AVG) and the concatenation function (CON). AVG is a special case of WAVG, where $\alpha_{ij} = \delta_{ij} / \sum_k \delta_{ik}$ for all (i, j) in Eq. (5). CON is the equivalent function to the word features used in Wang and Xu (2017) and a special case of WCON, where $\alpha_{i,i_l} = 1$ for all (i, i_l) in Eq. (6).

Recurrent Layers for Word-Integrated Character Representation The summary vector $\mathbf{a}_i$ and the context vector $\mathbf{h}_i$ for a character are together fed into additional recurrent layers, which are BiLSTM layers, to further contextualize character representations using word information of surrounding characters. Given the input $\mathbf{h}_i \oplus \mathbf{a}_i$, BiLSTMs calculate hidden vectors, and the hidden vectors $\mathbf{z}_{1:n}$ of the last BiLSTM layer are fed into the CRF layer.

5 Experiments

5.1 Settings

Datasets We evaluated our model on three datasets, CTB6 and MSR for Chinese word segmentation and BCCWJ (short unit word annotation) for Japanese word segmentation. CTB6 is Chinese Penn Treebank 6.0 (Xue et al., 2005). MSR is provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005). BCCWJ is Balanced Corpus of Contemporary Written Japanese 1.1 (Maekawa et al., 2014). We followed the same training/development/test split as in previous work (Yang and Xue, 2012; Chen et al., 2015b) for CTB6, official training/test split for MSR, and the same training/test split as in the Project Next NLP[3] for BCCWJ. We randomly selected 90% of the sentences in the training data as a training set and used the other 10% as a development set, respectively for MSR and BCCWJ.

Word Vocabulary Construction Apart from the given training and development sets for each dataset, we assumed no annotated information, including external dictionaries and third-party segmenters, was available in our experiments. Therefore, we used the training set and large unlabeled texts to obtain a word vocabulary to be used in our proposed model.

First, we trained a baseline model from each training set and applied it to large unlabeled texts. Then we collected auto-segmented words appearing in the texts[4] and gold words in the training set, and regarded the union of both kinds of words as a

---

[3] `http://www.ar.media.kyoto-u.ac.jp/mori/research/topics/PST/NextNLP.html`

[4] We discarded words occurring less than five times in auto-segmented texts, since their pre-trained embeddings were not learned by Word2Vec with the default minimum frequency of five as described later.

Parameter	Value
Character embedding size	300
Number of BiLSTM-C layers	2
Number of BiLSTM-C hidden units	600
Mini-batch size	100
Initial learning rate	1.0
Learning rate decay ratio	0.9
Gradient clipping threshold	5.0
Recurrent layer dropout rate	0.4
Word embedding size	300
Number of BiLSTM-WC layers	1
Number of BiLSTM-WC hidden units	600
Word vector dropout rate	0.2
Maximum word length	4

Table 1: Hyperparameters common between the baseline and the proposed model (top) and specific to the proposed model (bottom). BiLSTM-C and BiLSTM-WC, respectively, indicate recurrent layers for character and word-integrated character representation.

Method	BCCWJ	CTB6	MSR
BASE	98.63	95.43	96.70
AVG	98.82†	95.94†	97.63†
WAVG	98.85†	96.00†	**97.81**†‡
CON	98.84†	96.04†	97.77†
WCON	**98.93**†‡	**96.38**†‡	97.79†

Table 2: Results of the test sets. The table shows the mean of F1 scores of three runs for the baseline (BASE) and the proposed model variants. The symbols † and ‡ indicate statistical significance at 0.001 level over the baseline and over the variant without attention, respectively.

Method	BCCWJ	CTB6	MSR
Our WCON model$^{\circ}$	**98.9**	**96.4**	97.8
(Kitagawa and Komachi, 2018)	98.4	–	–
(Zhao and Kit, 2008)*	–	–	97.6
(Zhou et al., 2017)$^{\circ}$	–	96.2	97.8
(Wang and Xu, 2017)$^{\circ}$	–	–	**98.0**
(Liu et al., 2016)$^{\circ}$	–	95.5	97.6
(Zhang et al., 2016)$^{\circ}$	–	96.0	97.7
(Neubig et al., 2011)*	98.2*	–	–
(Sun et al., 2017)$^{\circ\bullet}$	–	96.3	97.9

Table 3: Comparison with state-of-the-art character-based (top) and word-based (middle) and other types of models (bottom) on the test sets. Models marked with a symbol indicate ones based on linear statistical algorithms ($\star$), ones with additional unlabeled texts ($\circ$) and ones replacing specific characters as preprocessing ($\bullet$). The result with $*$ is from our run on their released implementation.

word vocabulary. We used the non-core section of BCCWJ (BCCWJ-NC)[5] for the Japanese dataset and Chinese Gigaword Fifth Edition[6] for the Chinese datasets as unlabeled texts.

Pre-training of Embedding Parameters Following previous work (Collobert et al., 2011), we pre-trained word embeddings from large texts and used them to initialize the word embedding matrix in our proposed segmenter. To pre-train word embeddings, we used the gensim (Řehůřek and Sojka, 2010) implementation of Word2Vec (Mikolov et al., 2013) and applied it to the same texts as ones used to construct the word vocabularies, i.e., the auto-segmented BCCWJ-NC or Chinese Gigaword texts processed by the baseline segmenters. We used the toolkit with a skip-gram model, embedding size 300, the number of iterations one, and other default parameters. For words occurring only in a training set, we randomly initialized their embeddings. We fine-tuned all word embeddings during training of the proposed segmenter.

In contrast, we randomly initialized all character embeddings, since pre-trained character embeddings did not improve performance in our preliminary experiments.

Hyperparameter Setting Table 1 gives the hyperparameters for the proposed model. The same dropout strategy as in Zaremba et al. (2015) was applied to non-recurrent connections of recurrent layers. We used word vector dropout, which randomly replaces a word embedding $\mathbf{e}^{w}$ to a zero vector when calculating a word summary vector in Eq. (5) or (6). A mini-batch stochastic gradient descent was used to optimize parameters and decayed the learning rate with a fixed decay ratio every epoch after the first five epochs. We trained models for up to 20 epochs and selected the best model on the development set.

5.2 Results on the Test Sets

We evaluated our baseline and proposed model variants on the test sets of the three benchmark datasets. Table 2 shows the mean of F1 scores of three runs for each dataset and each model. Among the proposed model variants, WCON achieved the best performance in almost all cases. We observed the following three findings from the results.

First, all the word-integrated model variants consistently outperformed the pure character-based baseline. We conducted McNemar's tests in a similar manner to Kudo et al. (2004), and the improvement of each variant over the base-

---

[5] We restored provided auto-segmented texts to the original raw sentences and used them as unlabeled texts.

[6] `https://catalog.ldc.upenn.edu/ldc2011t13`

		Attention				Segmentation			
	Method	NoC	Lower	Upper	Acc	F1	Acc	Acc-CA	Acc-IA
BCCWJ	WAVG	2.09	30.09	94.53	79.69	99.14	99.22	99.74	**97.18**
	WCON				**90.97**	**99.21**	**99.29**	**99.88**	93.98
CTB6	WAVG	2.39	18.01	94.54	82.97	95.98	96.64	98.72	**86.50**
	WCON				**86.65**	**96.35**	**96.99**	**99.24**	82.37
MSR	WAVG	2.24	21.61	91.76	83.42	**98.53**	**98.75**	**99.65**	**94.26**
	WCON				**85.02**	98.51	98.72	99.63	93.53

Table 4: Attention accuracy, segmentation accuracy and F1-score on the development sets. The table also shows segmentation accuracy based on the cases where attention is correct (Acc-CA) and incorrect (Acc-IA), the average number of candidate words for a character (NoC) and lower and upper bounds of attention accuracy.

line was significant at 0.001 level. Second, the attention-based variants further boosted performance in comparison with their counterparts without attention. The improvements of WCON over CON on BCCWJ and CTB, and that of WAVG over AVG were statistically significant according to the McNemar's tests. We discuss the reason for the slight and insignificant performance difference between CON and WCON on MSR in §5.3. Third, the concatenation-based variants performed better than the average-based counterparts in almost all cases. This is probably because CON and WCON keep word length and character position information. For example, (d_w+1)-th to $2d_w$-th dimensions of a summary vector always represent a word whose length is two and which ends with a target character (namely $x_{i-1:i}$ for x_i), while AVG and WAVG lose this kind of information.

Table 3 shows the performance of state-of-the-art models without additional annotated data. We listed only the WCON results in Table 3 since it performed the best among all variants on the development sets. In comparison with the best previous models, we obtained better performance on BCCWJ and CTB6, and achieved approximately 31% and 3% error reductions, respectively. On MSR, we obtained a comparable performance with the character-based model with word features in Wang and Xu (2017), which used different unlabeled texts from ours to pre-train word embeddings. To our knowledge, our model is the first neural network-based model that has achieved state-of-the-art results on both Japanese and Chinese word segmentation.

5.3 Analysis of Word Attention

To analyze how the attention mechanism affected segmentation performance, we show in Table 4 attention accuracy of the proposed model with the attention-based functions of the development sets. Attention accuracy regards a predicted result as correct if a character x_i most strongly attends to the word corresponding to the gold segmentation. The table also shows the segmentation performance, where accuracy indicates character-level accuracy of segmentation label prediction.

Note that the attention accuracy of a model falls between lower and upper bounds shown in the table. The upper bound indicates the ratio of characters whose candidate words contain the gold word (then attention can be correctly paid) and the lower bound indicates the ratio of characters whose candidate words consist only of the gold word (then attention is always correctly paid). For example, assuming that the gold segmentation of the sentence in Figure 1 is "$x_1|x_2|x_3x_4x_5$ $(= w_1|w_2|w_8)$", candidate words for all characters contain their gold words, and those for characters x_1 and x_2 consist only of respective gold words w_1 and w_2. Thus, the upper and lower bounds for the sentence are $5/5$ and $2/5$, respectively.

Both WAVG and WCON achieved approximately $\geq$80% attention accuracy over more than two candidate words on average. The "Acc-CA" ("Acc-IA") column denotes the segmentation accuracy in cases where the attention was correctly (incorrectly) paid. We obtained particularly high segmentation accuracy (close to or higher than 99%). However, incorrect attention led to a large drop in segmentation accuracy. Moreover, we can see a clear tendency for WCON resulting in poorer segmentation accuracy in cases with incorrect attention, compared with WAVG. This suggests that attention by WCON is more sensitive to segmentation decisions; information on attended words is more directly propagated to succeeding layers. As for the slight performance difference between CON and WCON on MSR in Table 2, a possible explanation is that existence of fewer gold words (observed from the upper bound of accuracy) leads to inaccurate attention and segmentation.

	(a)	(b)	(c)	(d)	(e)	(f)
BASE	アドレスバー	ひい | て | は	オフライン	代金 | 引換 | 金 | 額	お | 茶ノ水	昼夜
CON	アドレス | バー	ひいては	オフライン	代金 | 引換 | 金 | 額	お | 茶 | ノ | 水	昼夜
WCON	アドレス | バー	ひいては	オフ | ライン	代金 | 引換 | 金額	お | 茶 | ノ | 水	昼 | 夜
Gold	アドレス | バー	ひいては	オフ | ライン	代金 | 引換 | 金額	お茶ノ水	昼夜
	⟨*adoresu*|*bā*⟩ (address|bar)	⟨*hītewa*⟩ (besides)	⟨*ofu*|*rain*⟩ (off-|line)	⟨*daikin*|*hikikae*|*kingaku*⟩ (the amount of| payment|on delivery)	⟨*ochanomizu*⟩ (Ochanomizu)	⟨*chūya*⟩ (day and night)

Figure 3: Examples of segmentation results

w_j ＼ x_i	···	バ ⟨*ba*⟩	···	バー ⟨*bā*⟩ (bar)	ドレス ⟨*doresu*⟩ (dress)	アドレス ⟨*adoresu*⟩ (address)
ア		–		–	–	**1.00**
ド		–		–	0.21	**0.78**
レ		–		–	0.07	**0.93**
ス		–		–	0.02	**0.98**
バ		0.40		**0.52**	–	–
ー		–		**1.00**	–	–

(a)

w_j ＼ x_i	ひ ⟨*hi*⟩	い ⟨*i*⟩	て ⟨*te*⟩	は ⟨*wa*⟩	···	ひいては ⟨*hītewa*⟩ (besides)
ひ	0.00	–	–	–		**1.00**
い	–	0.00	–	–		**1.00**
て	–	–	0.00	–		**1.00**
は	–	–	–	0.00		**1.00**

(b)

w_j ＼ x_i	オ ⟨*o*⟩	フ ⟨*fu*⟩	ラ ⟨*ra*⟩	···	オフ ⟨*ofu*⟩ (off-)	ライン ⟨*rain*⟩ (line)
オ	0.00	–	–		**1.00**	–
フ	–	0.00	–		**1.00**	–
ラ	–	–	0.00		–	**1.00**
イ	–	–	–		–	**1.00**
ン	–	–	–		–	**1.00**

(c)

w_j ＼ x_i	···	代金 ⟨*daikin*⟩ (payment)	引換 ⟨*hikikae*⟩ (on delivery)	換金 ⟨*kankin*⟩ (cashing)	金額 ⟨*kingaku*⟩ (amount of money)
代		**1.00**	–	–	–
金		**1.00**	–	–	–
引		–	**1.00**	–	–
換		–	0	**1.00**	–
金		–	–	**0.97**	0.03
額		–	–	–	**1.00**

(d)

w_j ＼ x_i	お ⟨*o*⟩	茶 ⟨*cha*⟩ (tea)	ノ ⟨*no*⟩	水 ⟨*mizu*⟩ (water)
お	**1.00**	–	–	–
茶	–	**1.00**	–	–
ノ	–	–	**1.00**	–
水	–	–	–	**1.00**

(e)

w_j ＼ x_i	昼 ⟨*hiru*⟩ (daytime)	夜 ⟨*yoru*⟩ (night)	昼夜 ⟨*chūya*⟩ (day and night)
昼	**0.96**	–	0.04
夜	–	**0.99**	0.01

(f)

Figure 4: Weight α_{ij}, which indicates the importance from character x_i to word w_i, learned by WCON for sentences (a)-(f) in Figure 3. Weights to gold words are highlighted with blue.

5.4 Segmentation Examples

To examine segmentation results of actual sentences by different methods, we picked up sentence segments (a)-(f) from the BCCWJ development set. We show in Figure 3 the results obtained by BASE, WCON, and CON, which are selected as the character-based baseline, the best of our model variants, and its counterpart without attention, respectively. In addition, we also show values of weight α_{ij} learned by WCON in Figure 4.

In examples (a) and (b), BASE resulted in a wrong segmentation. However, both word-integrated methods correctly segmented words with the benefit of word information corresponding to gold segmentations (*adoresu*, *bar* and *hītewa*). This suggests that word information enables a model to utilize information on distant characters with target characters directly. From WCON results, we confirmed that all characters strongly attended to correct words, as in Figure 4. This suggests that accurate attention contributed to predicting correct segmentations.

In examples (c) and (d), only WCON predicted correct segmentations. The existence of correct words in the vocabulary and correct attention probably resulted in the correct segmentation for (c). As for (d), although parts of characters attended to a wrong word (*kankin*), correct attention regarding surrounding characters (*hikikae* and *kingaku*) seems to lead to the correct segmentation.

In examples (e) and (f), WCON predicted the wrong results. The wrong results of (e) by CON and WCON are probably due to the non-existence of the gold word *ochanomizu*, which is a location name, in the vocabulary. As for (f), WCON paid incorrect attention and predicted the wrong segmentation, even though the correct word *chūya* exists in the vocabulary. The model learned the incorrect weights likely due to the infrequent occurrence of the correct words; the single words *hiru* and *yoru* occur in the training set tens or hundreds of times while the compound word *chūya* occurs only twice. We may reduce these errors due to no or infrequent occurrences of gold words by increasing word vocabulary size, e.g., using larger texts to pre-train word embeddings.

6 Related Work

Word Segmentation For both Chinese and Japanese, word segmentation has been traditionally addressed by applying linear statistical algorithms, such as maximum entropy (Xue, 2003), CRF (Peng et al., 2004; Kudo et al., 2004; Zhao and Kit, 2008), and logistic regression (Neubig et al., 2011).

Various neural network architectures have been explored for Chinese word segmentation to reduce the burden of manual feature engineering. Specifically, character-based neural models have been developed to model the task as a sequence labeling problem, starting with earlier work by (Zheng et al., 2013) and (Mansur et al., 2013), which applied feed-forward neural networks. Pei et al. (2014) used a neural tensor network to capture interactions between tags and characters. More sophisticated architectures have also been used as standard components of word segmentation models to derive effective features automatically. Chen et al. (2015a) proposed gated recursive neural networks to model complicated combinations of characters. Chen et al. (2015b) used LSTM to capture long distance dependencies. Xu and Sun (2016) combined LSTM and GRNN to capture long term information better by utilizing chain and tree structures. CNNs have been used to extract complex features such as character n-grams (Chen et al., 2017) and graphical features of Chinese characters (Shao et al., 2017).

On the other hand, word-based neural models have also been proposed. Typical word-based models (Zhang et al., 2016; Cai and Zhao, 2016; Cai et al., 2017; Yang et al., 2017) sequentially determine whether or not to segment each character on the basis of word-level features and segmentation history, while keeping multiple segmentation candidates by beam search decoding. Liu et al. (2016) combined neural architectures for segment (i.e., word) representations into a semi-CRF framework, which searches for an optimal segmentation sequence consisting of variable length segments. Sun et al. (2017) proposed a gap-based model to predict whether or not to segment two consecutive characters, using a deep CNN consisting of more than ten layers.

Recent works utilized word information on a character-based framework. Zhou et al. (2017) pre-trained character embeddings using word boundary information from auto-segmented texts. Wang and Xu (2017) explicitly introduced word information into their CNN-based model. They concatenated embeddings of a character and multiple words corresponding to n-grams (n ranging from 1 to 4) that include the target character.

For Japanese, less work employed neural models for word segmentation than for Chinese. Morita et al. (2015) integrated an RNN language model into a statistical Japanese morphological analysis framework, which simultaneously segments a sentence into words and predicts word features, such as POS and lemma. Kitagawa and Komachi (2018) applied a pure neural model based on LSTM and achieved a better performance than a popular statistical Japanese segmenter (Neubig et al., 2011).

Around the same time as our work, two other character-based models for word segmentation have been proposed. Ma et al. (2018) showed a standard BiLSTM model can achieve state-of-the-art results when combined with deep learning best practices, including dropout to recurrent connections (Gal and Ghahramani, 2016) and pre-trained embeddings of character bigrams. These techniques can also be applied to and can further boost performance of our model. Yang et al. (2018) proposed a lattice LSTM-based model with subsequence (word or subword) information. Their model also considers the importance of multiple words by integrating character and word information into an LSTM cell vector using a gate-mechanism. However, their model might not fully exploit word information, since word information is given to only the first and last characters of the word.

LSTM-CRF LSTM-CRF is a popular neural architecture, which has been applied to various tagging tasks, including word segmentation (Chen et al., 2015b), POS tagging and NER (Huang et al., 2015; Ma and Hovy, 2016; Rei et al., 2016). Ma and Hovy (2016) and Rei et al. (2016) introduced the internal character information of words on word-level labeling tasks in contrast to our work introducing candidate word information of characters in the character-level labeling task.

Attention Mechanism An attention mechanism (Bahdanau et al., 2015) was first introduced in machine translation to focus on appropriate parts of a source sentence during decoding. This mechanism has been widely applied to various NLP tasks, including question answering (Sukhbaatar et al., 2015), constituency parsing (Vinyals et al., 2015), relation extraction (Lin et al., 2016) and natural language inference (Parikh et al., 2016). Rei et al. (2016) introduced a gate-like attention mechanism on their word-based sequence labeling model to determine the importance between the word itself and the internal characters for each word.

7 Conclusion and Future Work

In this paper, we proposed a word segmentation model that integrates word-level information into a character-based framework, aiming to take the advantages of both character- and word-based models. The experimental results show that our model with an attention-based composition function outperforms the state-of-the-art models on both Japanese and Chinese benchmark datasets.

Our analysis suggests that a word vocabulary with larger coverage can reduce errors deriving from unknown words. In future work, we will explore (1) the relationship between vocabulary coverage and segmentation performance, and (2) the effect of using pre-trained word vectors learned from different domain texts in domain adaptation scenarios.

Acknowledgments

We would like to thank Atsushi Fujita, Rui Wang, and the anonymous reviewers for their helpful feedback on this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3960–3966.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for Chinese word segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1744–1753, Beijing, China. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for Chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206, Lisbon, Portugal. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Thomas Emerson. 2005. The 2nd international Chinese word segmentation bakeoff. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *Computing Research Repository*, arXiv:1508.01991.

Yoshiaki Kitagawa and Mamoru Komachi. 2018. Long short-term memory for Japanese word segmentation. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2880–2886.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. 2014. Balanced corpus of contemporary written Japanese. *Language Resources and Evaluation*, 48(2):345–371.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and Chinese word segmentation. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan. Asian Federation of Natural Language Processing.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for Chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, Baltimore, Maryland. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568, Geneva, Switzerland. COLING.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Marek Rei, Gamal K.O. Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan. The COLING 2016 Organizing Committee.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Zhiqing Sun, Gehui Shen, and Zhihong Deng. 2017. A gap-based framework for Chinese word segmentation via very deep convolutional networks. *Computing Research Repository*, arXiv:1712.09509.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781.

Chunqi Wang and Bo Xu. 2017. Convolutional neural network with word embeddings for Chinese word segmentation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 163–172, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for Chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–572, Berlin, Germany. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pages 29–48.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice LSTM for Chinese word segmentation. *Computing Research Repository*, arXiv:1810.12594.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–794, Jeju Island, Korea. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2017. Word-context character embeddings for Chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.

VCWE: Visual Character-Enhanced Word Embeddings

Chi Sun, Xipeng Qiu*, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{sunc17,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Chinese is a logographic writing system, and the shape of Chinese characters contain rich syntactic and semantic information. In this paper, we propose a model to learn Chinese word embeddings via three-level composition: (1) a convolutional neural network to extract the intra-character compositionality from the visual shape of a character; (2) a recurrent neural network with self-attention to compose character representation into word embeddings; (3) the Skip-Gram framework to capture non-compositionality directly from the contextual information. Evaluations demonstrate the superior performance of our model on four tasks: word similarity, sentiment analysis, named entity recognition and part-of-speech tagging.[1]

1 Introduction

Distributed representations of words, namely word embeddings, encode both semantic and syntactic information into a dense vector. Currently, word embeddings have been playing a pivotal role in many natural language processing (NLP) tasks. Most of these NLP tasks also benefit from the pre-trained word embeddings, such as *word2vec* (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014), which are based on the distributional hypothesis (Harris, 1954): words that occur in the same contexts tend to have similar meanings. Earlier word embeddings often take a word as a basic unit, and they ignore compositionality of its sub-word information such as morphemes and character n-grams, and cannot competently handle the rare words. To improve the performance of word embeddings, sub-word information has been employed (Luong et al., 2013; Qiu et al., 2014; Cao and Rei, 2016; Sun et al., 2016a; Wieting et al., 2016; Bojanowski et al., 2016).

Compositionality is more critical for Chinese, since Chinese is a logographic writing system. In Chinese, each word typically consists of fewer characters and each character also contains richer semantic information. For example, Chinese character "休" (rest) is composed of the characters for "人" (person) and "木" (tree), with the intended idea of someone leaning against a tree, i.e., resting.

Based on the linguistic features of Chinese, recent methods have used the character information to improve Chinese word embeddings. These methods can be categorized into two kinds:

1) One kind of methods learn word embeddings with its constituent character (Chen et al., 2015), radical[2] (Shi et al., 2015; Yin et al., 2016; Yu et al., 2017) or strokes[3] (Cao et al., 2018). However, these methods usually use simple operations, such as averaging and n-gram, to model the inherent compositionality within a word, which is not enough to handle the complicated linguistic compositionality.

2) The other kind of methods learns word embeddings with the visual information of the character. Liu et al. (2017) learn character embedding based on its visual characteristics in the text classification task. Su and Lee (2017) also introduce a pixel-based model that learns character features from font images. However, their model is not shown to be better than *word2vec* model because it has little flexibility and fixed character features.

Besides, most of these methods pay less attention to the non-compositionality. For example, the

---

*Corresponding author.

[1] The source codes are available at https://github.com/HSLCY/VCWE

[2] the graphical component of Chinese, referring to https://en.wikipedia.org/wiki/Radical_(Chinese_characters)

[3] the basic pattern of Chinese characters, referring to https://en.wikipedia.org/wiki/Stroke_(CJKV_character)

Proceedings of NAACL-HLT 2019, pages 2710–2719
Minneapolis, Minnesota, June 2 - June 7, 2019.

semantic of Chinese word “沙发” (sofa) cannot be composed by its contained characters “沙” (sand) and “发” (hair).

In this paper, we fully consider the compositionality and non-compositionality of Chinese words and propose a visual character-enhanced word embedding model (VCWE) to learn Chinese word embeddings. VCWE learns Chinese word embeddings via three-level composition:

- The first level is to learn the intra-character composition, which gains the representation of each character from its visual appearance via a convolutional neural network;
- The second level is to learn the inter-character composition, where a bidirectional long short-term neural network (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) with self-attention to compose character representation into word embeddings;
- The third level is to learn the non-compositionality, we can learn the contextual information because the overall framework of our model is based on the skip-gram.

Evaluations demonstrate the superior performance of our model on four tasks such as word similarity, sentiment analysis, named entity recognition and part-of-speech tagging.

2 Related Work

In the past decade, there has been much research on word embeddings. Bengio et al. (2003) use a feedforward neural network language model to predict the next word given its history. Later methods (Mikolov et al., 2010) replace feedforward neural network with the recurrent neural network for further exploration. The most popular word embedding system is *word2vec*, which uses continuous-bag-of-words and Skip-gram models, in conjunction with negative sampling for efficient conditional probability estimation (Mikolov et al., 2013a).

A different way to learn word embeddings is through factorization of word co-occurrence matrices such as GloVe embeddings (Pennington et al., 2014), which have been shown to be intrinsically linked to Skip-gram and negative sampling (Levy and Goldberg, 2014).

The models mentioned above are popular and useful, but they regard individual words as atomic tokens, and the potentially useful internal structured information of words is ignored. To improve the performance of word embedding, sub-word information has been employed (Luong et al., 2013; Qiu et al., 2014; Cao and Rei, 2016; Sun et al., 2016a; Wieting et al., 2016; Bojanowski et al., 2016). These methods focus on alphabetic writing systems, but they are not directly applicable to logographic writing systems.

For the alphabetic writing systems, research on Chinese word embedding has gradually emerged. These methods focus on the discovery of making full use of sub-word information. Chen et al. (2015) design a CWE model for jointly learning Chinese characters and word embeddings. Based on the CWE model, Yin et al. (2016) present a multi-granularity embedding (MGE) model, additionally using the embeddings associated with radicals detected in the target word. Xu et al. (2016) propose a similarity-based character-enhanced word embedding (SCWE) model, exploiting the similarity between a word and its component characters with the semantic knowledge obtained from other languages. Shi et al. (2015) utilize radical information to improve Chinese word embeddings. Yu et al. (2017) introduce a joint learning word embedding (JWE) model and Cao et al. (2018) represent Chinese words as sequences of strokes and learn word embeddings with stroke n-grams information.

From another perspective, Liu et al. (2017) provide a new way to automatically extract character-level features, creating an image for the character and running it through a convolutional neural network to produce a visual character embedding. Su and Lee (2017) also introduce a pixel-based model that learns character features from its image.

Chinese word embeddings have recently begun to be explored, and have so far shown great promise. In this paper, we propose a visual character-enhanced word embedding (VCWE) model that can learn Chinese word embeddings from corpus and images of characters. The model combines the semantic information of the context with the image features of the character, with superior performance in several benchmarks.

3 Proposed Model

In this section, we introduce the visual character-enhanced word embedding (VCWE) model for Chinese word representation.

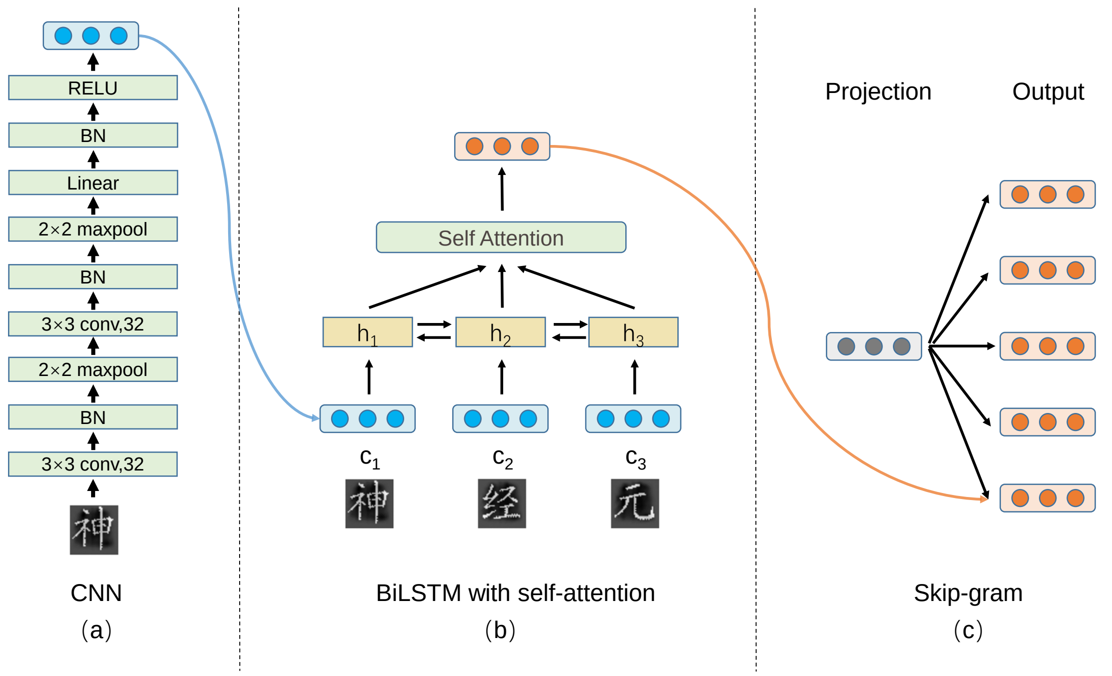


Figure 1: The overall architecture of our approach.

Given a Chinese word w consisting of n characters $c_1, \cdots, c_n$, its semantic may come from either its contained characters or its contexts. Therefore, we use the two-level hierarchical composition to compose the word embedding, which further learned according to its context.

The overall architecture of our approach is on Figure 1. We first use a convolutional neural network (CNN) to model the intra-character compositionality of character from its visual shape information. We use the output of CNN as the embeddings of the character. Then the character embeddings are used as the input of the bidirectional LSTM network to model the inter-character compositionality. After a self-attention layer, we can get the representation of the word. Finally, based on the Skip-Gram framework, we learn the word embeddings with the visual character-enhanced embedding of the context.

3.1 Intra-Character Composition

Since the shape of a Chinese character provides rich syntactic and semantic information, the representation of a character can be composed by its intrinsic visual components. Following the success of the convolutional neural network (CNN) (LeCun et al., 1995) in computer vision, we use CNN to directly model the natural composition of a character from its image.

We first convert each character into an image of size 40×40, a deep CNN is used to fuse its visual information fully. The specific structure of the CNN is shown in Figure 1(a), which consists of two convolution layers and one linear layer. Each convolution layer is followed by a max pooling layer and a batch normalization layer. The lower layers aim to capture the stroke-level information, and the higher layers aim to capture the radical-level and component-level information.

The output of CNN can be regarded as the representation of the character. The character representation by its visual information can fully capture its intrinsic syntactic and semantic information with the intra-character compositionality.

The parameters of CNN are learned through backpropagation in end-to-end fashion.

3.2 Inter-Character Composition

After obtaining the representation of characters, we combine them into word embedding. The word embedding need to capture the character-level compositionality fully. Here, we use the bidirectional LSTM (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) with self-attention to fuse the inter-character information of a word.

The structure of our Bi-LSTM with self-attention is shown in Figure 1(b).

Given a word w consisting of n characters $c_1, \cdots, c_n$, we use $\mathbf{e}_1, \cdots, \mathbf{e}_n$ denote is the character representations, which are the output of the CNN rather than randomly initialized.

The word w is firstly encoded using a Bi-LSTM:

$$\mathbf{h}_i^F = \text{LSTM}(\mathbf{h}_{i-1}^F, \mathbf{e}_i), \tag{1}$$

$$\mathbf{h}_i^B = \text{LSTM}(\mathbf{h}_{i+1}^B, \mathbf{e}_i), \tag{2}$$

$$\mathbf{h}_i = [\mathbf{h}_i^F; \mathbf{h}_i^B], \tag{3}$$

$$H = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n], \tag{4}$$

where $\mathbf{h}_i$ is the hidden state of the i-th character in w.

Then we use self-attention to obtain the inter-character compositionality. Following the self-attention proposed by (Lin et al., 2017), we compute an attention vector α:

$$\alpha = \text{softmax}(\mathbf{v} \tanh(U \mathbf{h}_i^{\mathrm{T}})), \tag{5}$$

where $\mathbf{v}$ and U are learnable weight parameters.

Finally, the representation of word w is:

$$\mathbf{m} = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i. \tag{6}$$

Since the Bi-LSTM's hidden state of each character is different according to its contexts, we believe the hidden state can capture both the compositional and non-compositional relations of the characters within a word.

After obtaining the word representation, Skip-Gram (Mikolov et al., 2013a) is used to learn the word embedding with its context information. Skip-Gram is a useful framework for learning word vectors, which aims to predict context words given a target word in a sentence.

Given a pair of words (w, c), we denote $p(c|w)$ as the probability that the word c is observed in the context of the target word w.

With the negative-sampling approach, skip-gram formulates the probability $p(c|w)$ as follows:

Given a pair of words (w, c), the probability that the word c is observed in the context of the target word w is given by

$$p(D = 1|w, c) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{c}), \tag{7}$$

where $\mathbf{w}$ and $\mathbf{c}$ are embedding vectors of w and c respectively, σ is the sigmoid function.

The probability of not observing word c in the context of w is given by:

$$p(D = 0|w, c) = 1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{c}). \tag{8}$$

4 Training

4.1 Objective Function

Given the target word w, its context word c and k negative words $\tilde{c}_1, ..., \tilde{c}_k$. The word w is a word selected from a sentence in the corpus, and the context c is a nearby word within a window size l. The negative sample $\tilde{c}_i$ is a word that is randomly sampled at a certain frequency in the vocabulary.

The loss function of VCWE model is as follows:

$$L = L_1 + L_2, \tag{9}$$

$$L_1 = \log \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{c}) + \sum_{i=1}^{k} \log \sigma(-\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{c}}_i), \tag{10}$$

$$L_2 = \log \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{m}_c) + \sum_{i=1}^{k} \log \sigma(-\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{m}}_i), \tag{11}$$

where $\mathbf{w}$ is the lookup embedding of target word; $\mathbf{c}$ and $\tilde{\mathbf{c}}_i$ are the lookup embeddings of the context and negative words respectively; $\mathbf{m}_c$ and $\tilde{\mathbf{m}}_i$ are visual enhanced word embeddings of the context and negative words respectively.

Here, we use the visually enhanced word embedding as the representation of context word instead of the target word. The final embedding of the target word is indirectly affected by the visual information. Thus, the final word embedding can have an advantage of fully utilizing intra-character compositionality from CNN, inter-character compositionality from LSTM, and context information from Skip-gram.

4.2 Word Sampling

We use a word sampling scheme similar to the implementation in word2vec (Mikolov et al., 2013a,b) to balance the importance of frequent words and rare words. Frequent words such as "的"(of), "是"(is), "这"(this) are not as meaningful as relatively less frequent words such as "猫"(cat), "喜欢"(like), "水果"(fruit). To improve the performance of word embeddings, we use subsampling(Mikolov et al., 2013b) to discard the word w with the probability of $P(w) = 1 - \sqrt{\frac{t}{f(w)}}$ when generating the batch, where $f(w)$ is the frequency of word w and t is a chosen threshold, typically around 10^{-5}.

To generate negative context words, we sample each word w according to distribution $P(w) \propto U(w)^{\frac{3}{4}}$, where $U(w)$ is the unigram distribution, which is the frequency of single words appearing in the corpus. This method also plays a role in reducing the frequency of occurrence of high-frequency words.

5 Experiments

5.1 Preparation for training data

We download Chinese Wikipedia dump[4] on May 20, 2018, which consists of 278K Chinese Wikipedia articles. We use the WikiExtractor toolkit[5] to convert data from XML into text format. We find that the corpus consists of both simplified and traditional Chinese characters. Hence we utilize the opencc toolkit[6] to normalize all characters as simplified Chinese. We remove non-Chinese characters such as punctuation marks by retaining the characters whose Unicode falls into the range between 0x4E00 and 0x9FA5. We use THULAC [7](Sun et al., 2016b) for word segmentation.

We discard words that appeared less than 100 times and obtain a vocabulary of size 66,856. We count the frequency of occurrence of each word to prepare for the subsampling work.

In all 66,856 words, we extract 5030 unique characters. We use a Chinese character image generation software to generate the images of these Chinese characters. We subtract a mean image from each input image to center it before feeding it into the CNN. The pre-processed Chinese character images are shown in Figure 2.

Figure 2: The pre-processed Chinese character images.

5.2 Hyperparameters

Models used for evaluation have dimension $D = 100$ and use context window $l = 5$ unless stated otherwise. We use the threshold $t = 10^{-5}$ for subsampling, which is the recommended value for word2vec Skip-gram (Mikolov et al., 2013a) on large datasets. The number of negative samples per word is 5.

We use mini-batch asynchronous gradient descent with Adam (Kingma and Ba, 2014). The initial learning rate is 0.001.

5.3 Baselines

We compare our model to the following open source state-of-art models:

- word2vec[8] (Mikolov et al., 2013a) is arguably the most popular word embedding, which uses continuous-bag-of-words (CBOW) and Skip-gram models. We train word2vec with both Skip-gram and CBOW models. We did not train Glove(Pennington et al., 2014) because it did not perform well in many previous Chinese word embedding papers.
- CWE[9] (Chen et al., 2015) is character-enhanced word embeddings which introduce internal character information into word embedding methods to alleviate excessive reliance on the external information.
- GWE[10](Su and Lee, 2017) is a pixel-based Chinese word embedding model, which exploits character features from font images by convolutional autoencoders.
- JWE[11](Yu et al., 2017) is a model to jointly learn the embeddings of Chinese words, characters, and sub character components.

For a fair comparison between different algorithms, we use the same corpus and the same hyperparameters mentioned in previous subsections.

5.4 Word Similarity Task

We evaluate our embeddings on the Chinese word similarity datasets wordsim-240 and wordsim-296 provided by (Chen et al., 2015). Besides, we translate two English word similarity datasets MC-30

---

[4] https://dumps.wikimedia.org/zhwiki/20180520/

[5] https://github.com/attardi/wikiextractor/blob/master/WikiExtractor.py

[6] https://github.com/BYVoid/OpenCC

[7] https://github.com/thunlp/THULAC-Python

[8] https://code.google.com/archive/p/word2vec/

[9] https://github.com/Leonard-Xu/CWE

[10] https://github.com/ray1007/gwe

[11] https://github.com/hkust-knowcomp/jwe

Model	WS-240	WS-296	MC-30	RG-65	avg	Δ
Skip-gram	50.23	56.94	69.66	59.86	59.17	-
CBOW	51.49	61.01	68.97	63.85	61.33	+2.16
CWE	52.63	58.98	68.82	59.60	60.01	+0.84
GWE	52.74	58.22	68.23	60.74	59.98	+0.81
JWE	51.92	59.84	70.27	62.83	61.22	+2.05
VCWE	57.81	**61.29**	**72.77**	**70.62**	**65.62**	**+6.45**
-CNN	55.82	59.60	66.87	68.53	62.71	+3.54
-LSTM	**58.13**	60.85	68.03	69.78	64.20	+5.03

Table 1: Spearman correlation for word similarity datasets, "-CNN" represents replacing the CNN and image information with randomly initialized character embedding, "-LSTM" represents replacing Bi-LSTM network and self-attention with the averaging operation. For each dataset, we boldface the score with the best performance across all models.

Model	NOTEBOOK	CAR	CAMERA	PHONE	ALL	avg	Δ
Skip-gram	69.84	77.12	80.80	81.25	86.65	79.13	-
CBOW	74.60	75.42	82.59	82.81	84.07	79.90	+0.77
CWE	73.02	80.51	81.25	81.25	82.09	79.62	+0.49
GWE	74.60	78.81	79.46	83.98	83.92	80.15	+1.02
JWE	77.78	78.81	81.70	81.64	85.13	81.01	+1.88
VCWE	80.95	**85.59**	**83.93**	**84.38**	**88.92**	**84.75**	**+5.62**
-CNN	**84.13**	81.36	81.70	83.69	84.22	83.02	+3.89
-LSTM	79.37	80.51	80.36	**84.38**	85.58	82.04	+2.91

Table 2: Accuracy for Sentiment analysis task. The configurations are the same of the ones used in Table 1.

(Miller and Charles, 1991) and RG-65 (Rubenstein and Goodenough, 1965) to Chinese[12]. Each dataset contains a list of word pairs with a human score of how related or similar the two words are.

We calculate the Spearman correlation (Spearman, 1904) between the labels and our scores generated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels. The evaluation results of our model and baseline methods on word similarity datasets are shown in Table 1.

From the results, we can see that VCWE outperforms other baseline models. The effect of CBOW is much better than Skip-gram. The impact of GWE and CWE are relatively close. The JWE model works better than other benchmark models. In the VCWE model, when we remove the CNN and the image information, the result falls by 2.91. When we replace Bi-LSTM network and self-attention with the averaging operation, the result drops by 1.42.

In the last subsection, we will qualitatively analyze the results of word similarity for different models.

5.5 Sentiment Analysis Task

To evaluate the quality of our vectors regarding semantics, we use datasets[13] collected by (Peng et al., 2018), which contain Chinese reviews in four domains: notebook, car, camera, and phone. They manually labeled the sentiment polarity towards each aspect target as either positive or negative. It is a binary classification task. Similar to how we process the training data, we remove non-Chinese characters and use THULAC for performing Chinese word segmentation. We build classifiers with the bidirectional LSTM (Hochreiter and Schmidhuber, 1997) network with self-attention (Lin et al., 2017). We use the standard training/dev/test split and report accuracy using different embeddings generated by different meth-

---

[12] https://github.com/FudanNLP/VCWE

[13] http://sentic.net/chinese-review-datasets.zip

ods in Table 2.

As shown in Table 2, Skip-gram performs well on the combination of the four groups, but it does not perform well in the works of a particular group. JWE outstrips other baseline methods by around 1.1 points. The VCWE model has achieved outstanding results in the car, camera and phone category, with an accuracy rate of at least 3 points higher than other models, indicating that this method of training word embeddings with visual character-level features can achieve better results on downstream tasks.

5.6 Named Entity Recognition Task

We evaluate our model on the named entity recognition task. We use an open source Chinese NER model to test our word embeddings on MSRA dataset[14]. MSRA is a dataset for simplified Chinese NER. It comes from SIGHAN 2006 shared task for Chinese NER (Levow, 2006). We pretrain word embeddings from different models and feed them into the input layer as features.

The key to the task is to extract named entities and their associated types. Better word embeddings could get a higher F1 score of NER. The results in Table 3 show that our model also outperforms baseline models in this task. The performance of CWE and GWE models are similar, both slightly lower than Skip-gram and CBOW models. The F1 score of the JWE model exceeds that of other baseline models and is similar to our model. When removing the CNN and image information, our LSTM with the self-attention model can also achieve the best results on this task, indicating that the learned inter-character composition is practical.

5.7 Part-of-speech Tagging Task

The evaluation is performed on the PKU's People's Daily [15] (PPD) (Yu et al., 2001) with the standard training/dev/test split. The model is trained with the bidirectional LSTM model using the same hyper-parameters. Results on the POS accuracy on the test set are reported in Table 3.

The gap between the usage of different embeddings is not significant, and our model has achieved the best results with a slight advantage.

Model	NER			POS Tag
	Prec.	Recall	F1	Acc
Skip-gram	85.30	84.18	84.74	95.87
CBOW	85.64	82.98	84.29	95.79
CWE	83.89	82.57	83.23	95.45
GWE	84.06	82.52	83.28	95.45
JWE	85.74	84.87	85.30	95.91
VCWE	86.93	84.64	**85.77**	**96.00**
-CNN	86.73	84.83	**85.77**	95.92
-LSTM	85.98	84.53	85.25	95.96

Table 3: Chinese NER and POS tagging results for different pretrained embeddings. The configurations are the same of the ones used in Table 1.

5.8 Qualitative analysis

To better understand the quality of the learning word embedding for each model, we conduct a qualitative analysis by doing some case studies in Table 4 to illustrate the most similar words for certain target words under different methods. Explicitly, we present the top 10 words that are most similar to our target word. The similar words are retrieved based on the cosine similarity calculated using the learned embeddings.

The first example word we consider is "唐诗(Tang poetry)". It refers to poetry written in or around the time of or in the characteristic style of China's Tang Dynasty.

All the top-ranked words identified by GWE contain the character "唐(Tang)" and "诗(poetry)", but in addition to the Tang Dynasty, "唐(Tang)" also has other meanings such as surnames. GWE yields several words such as "唐璜(Don Juan)", "唐寅(Tang Yin)", "唐僧(Monk Tang)" and "唐括(Tang Ku)", which do not appear to be semantically close to the target word. In Skip-gram and JWE, certain words such as "佚(anonymity)" and "古今(ancient and modern)" do not appear to be semantically very closely related to the target word. In our VCWE model, all the top-ranked words are semantically related to the target word, including the genre of poetry, poets of the Tang Dynasty, and so on.

We choose the "沙发(sofa)" as the second target word. Like the first two words, GWE only pays attention to the character "沙(sand)". Skip-gram and JWE have some irrelevant words such as "电话亭(telephone box)" and "广告牌(billboard)". VCWE pays more attention to

---

[14] https://github.com/bamtercelboo/pytorch_NER_PosTag_BiLSTM_CRF

[15] http://klcl.pku.edu.cn/zygx/zyxz/index.htm

Targets	Skip-gram	GWE	JWE	VCWE
唐诗(Tang poetry)	散曲(Qu-Poetry)	宋诗(Song poetry)	诗话(notes on poetry)	诗话(notes on poetry)
	琴谱(music score)	赋诗(indite)	古今(ancient and modern)	宋诗(Song poetry)
	佚(anonymity)	诗韵(rhyme)	佚(anonymity)	绝句(jueju)
	宋词(Song Ci Poetry)	汉诗(Chinese poetry)	乐府(Yuefu)	宋词(Song Ci Poetry)
	白居易(Bai Juyi)	唐璜(Don Juan)	琴谱(music score)	吟咏(chant)
	绝句(jueju)	吟诗(recite poems)	辑录(compile)	乐府(Yuefu)
	著录(record)	唐寅(Tang Yin)	刻本(carving copy)	七言(seven-character)
	楚辞(Chu Songs)	唐僧(Monk Tang)	传世(be handed down)	李商隐(Li Shangyin)
	乐府(Yuefu)	唐括(Tang Ku)	古诗(ancient poetry)	古诗(ancient poetry)
	辑录(compile)	诗(poetry)	散曲(Qu-Poetry)	诗文(poetic prose)
沙发(sofa)	办公桌(bureau)	沙漏(hourglass)	桌子(desk)	衣柜(wardrobe)
	卧室(bedroom)	沙尘(sand)	衣柜(wardrobe)	卧室(bedroom)
	椅子(chair)	沙袋(sandbag)	毛巾(washcloth)	浴缸(bathtub)
	楼上(upstairs)	沙盒(sandbox)	书桌(secretaire)	客厅(living room)
	客厅(living room)	沙哑(raucity)	棉被(quilt)	窗帘(curtain)
	浴缸(bathtub)	沙嘴(sandspit)	长椅(bench)	椅子(chair)
	楼下(downstairs)	沙嗲(satay)	窗帘(curtain)	壁炉(fireplace)
	雨衣(raincoat)	沙包(sandbag)	浴缸(bathtub)	房门(door)
	血迹(bloodstain)	沙织(Saori)	房门(door)	长椅(bench)
	电话亭(telephone box)	沙蚕(nereid)	广告牌(billboard)	桌子(desk)

Table 4: Case study for qualitative analysis. Given the target word, we list the top 10 similar words from each algorithm so as to observe the differences.

the non-compositionality, and the results are better than other models.

Limited to the width of the table, we do not show the results of CWE model. The results of the GWE model are not much different from the CWE model, indicating that the image features obtained by pre-training of GWE may not play a decisive role. However, our model does not pre-train image information, but jointly trains and dynamically updates image feature information and it works better. JWE model is similar to Skip-gram model in that they pay more attention to contextual information, but sometimes the model gets some irrelevant words.

6 Discussion

Unlike phonograms, logograms have word and phrase meanings singularly. The images of Chinese characters contain rich semantic information. Since logographic languages are more closely associated with images than alphabet languages, it makes sense to mine the characteristics of these images.

Liu et al. (2017) provide a new way to automatically extract character-level features, creating an image for the character and running it through a convolutional neural network to produce a visual character embedding. However, this method does not utilize the rich semantic information of contextual words. Our model extracts both image features and contextual semantic information.

Su and Lee (2017) introduce a pixel-based model that learns character features from font images. However, they use convolutional auto-encoder(convAE) to extract image features in advance, and then add these features to the CWE (Chen et al., 2015) model. In the end, the effect of the model is not much different from CWE. Our model is an end-to-end model. We update the image's feature parameters in real time during training, and our model achieves better results than the GWE model.

Our research focuses on simplified Chinese word embeddings, and the idea can also be applied to other languages that share a similar writing system, such as traditional Chinese, Japanese, and so on.

7 Conclusion and Future Work

In this paper, we proposed a pixel-based model to learn Chinese word embeddings with character embeddings that are compositional in the components of the characters. We utilized the visual features of Chinese characters to enhance the word embedding. We showed that our model outperforms the baseline model in the word similarity, sentiment analysis, named entity recognition and part-of-speech tagging tasks.

In summary, we optimized our pixel-based word embedding method to make the model end-to-end and make full use of the contextual information. In the future, we hope to apply our model to other downstream tasks and other logographic writing systems.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), National Key Research and Development Program of China (No. 2017YFB1002104), and National Natural Science Foundation of China (No. 61672162 and 61751201).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *arXiv preprint arXiv:1606.02601*.

Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. 2018. cw2vec: Learning chinese word embeddings with stroke n-gram information.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *IJCAI*, pages 1236–1242.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. *arXiv preprint arXiv:1704.04859*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Haiyun Peng, Yukun Ma, Yang Li, and Erik Cambria. 2018. Learning multi-grained aspect target sequence for chinese sentiment analysis. *Knowledge-Based Systems*, 148:167–176.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 594–598.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Tzu-Ray Su and Hung-Yi Lee. 2017. Learning chinese word representations from glyphs of characters. *arXiv preprint arXiv:1708.04755*.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016a. Inside out: Two jointly predictive models for word representations and phrase representations. In *AAAI*, pages 2821–2827.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016b. Thulac: An efficient lexical analyzer for chinese. Technical report, Technical Report.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve chinese word embeddings by exploiting internal structure. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1041–1050.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 981–986.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 286–291.

Shiwen Yu, Jianming Lu, Xuefeng Zhu, Huiming Duan, Shiyong Kang, Honglin Sun, Hui Wang, Qiang Zhao, and Weidong Zhan. 2001. Processing norms of modern chinese corpus. Technical report, Technical report.

Subword Encoding in Lattice LSTM for Chinese Word Segmentation

Jie Yang[♠♣]**, Yue Zhang**[♦]**, Shuailong Liang**[▲]
[♠]Brigham and Women's Hospital. Boston, USA
[♣]Harvard Medical School, Harvard University. Boston, USA
[♦]School of Engineering, Westlake University. Hangzhou, China
[▲]Singapore University of Technology and Design. Singapore
`jieynlp@gmail.com`
`yue.zhang@wias.org.cn`
`shuailong_liang@mymail.sutd.edu.sg`

Abstract

We investigate subword information for Chinese word segmentation, by integrating sub word embeddings trained using byte-pair encoding into a Lattice LSTM (LaLSTM) network over a character sequence. Experiments on standard benchmark show that subword information brings significant gains over strong character-based segmentation models. To our knowledge, this is the first research on the effectiveness of subwords on neural word segmentation.

1 Introduction

Chinese word segmentation (CWS) is a traditional NLP task (Sproat et al., 1996), the features for which have been a central research topic. Statistical methods consider characters (Xue et al., 2003), subwords (Zhang et al., 2006), and words (Zhang and Clark, 2007) as input features. Among these, both characters (Chen et al., 2015a) and words (Zhang et al., 2016; Cai and Zhao, 2016; Yang et al., 2017) have also shown useful in recent neural models. However, how to utilize the subword features in neural networks has not been investigated yet.

In this paper, we fill this gap by proposing a subword-based neural word segmentor, by integrating two strands of works: the byte pair encoding (BPE) algorithm (Gage, 1994) and the lattice LSTM structure (Zhang and Yang, 2018). The BPE algorithm constructs a subword list from raw data and lattice LSTM introduces subwords into character LSTM representation. In particular, our baseline is a BiLSTM-CRF segmentor (Chen et al., 2015b) and we replace LSTM with lattice LSTM using subwords to encode character composition information. Our code[1] is based on NCRF++ (Yang and Zhang, 2018).

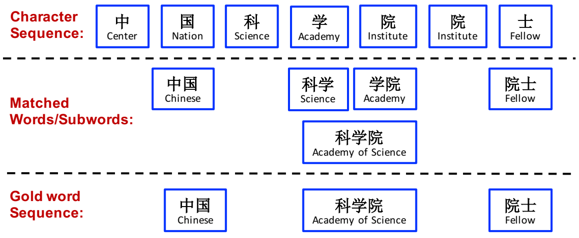


Figure 1: Segmentation with ambiguous words.

Compared with character-based neural segmentors, our model can utilize abundant character combination (subword) information, which is effective to disambiguate characters. For example, in Figure 1, the subword “学院(Academy)” ensures that the character “学” means “Academy(noun)” rather than “study(verb)”. Compared with the word-based neural models (Zhang et al., 2016; Cai and Zhao, 2016), ambiguous subwords in a context can provide additional information for disambiguation. For instance, the subword “科学院(Academy of Sciences)” and “学院(Academy)” can be useful in determining the correct segmentation, which is “科学院/(Academy of Sciences/)”.

To our knowledge, we are the first to use subwords in a neural network segmentor. We investigate the contributions of subword lexicons and their pretrained embeddings through controlled experiments. Results on four benchmarks show that the proposed model can give comparable results with state-of-the-art models.

2 Related Work

State-of-the-art statistical segmentors use either sequence labeling methods e.g. CRF (Lafferty et al., 2001) with character features (Peng et al., 2004; Zhao et al., 2006) or the transition-based models with word features (Zhang and Clark, 2007; Sun, 2010). Neural segmentors (Chen et al.,

---

[1] Our code is released at `https://github.com/jiesutd/SubwordEncoding-CWS`.

Proceedings of NAACL-HLT 2019, pages 2720–2725
Minneapolis, Minnesota, June 2 - June 7, 2019.

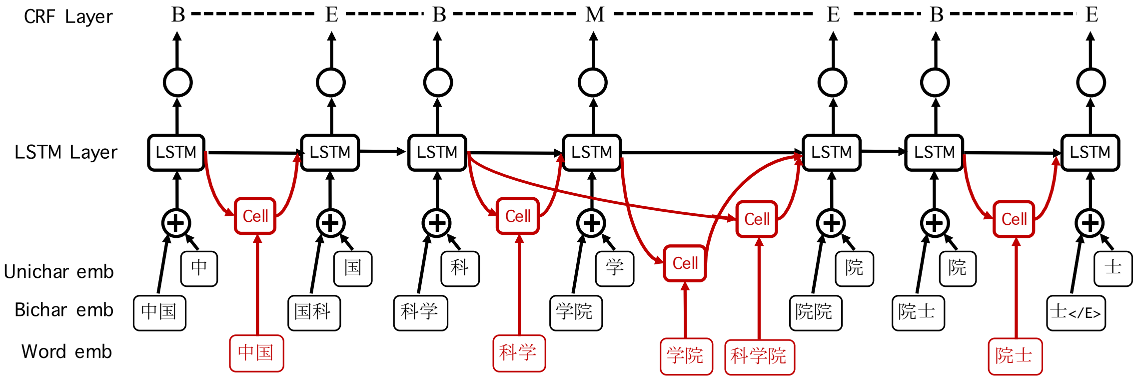


Figure 2: Models. Only forward LSTM is illustrated here.

2015a; Cai and Zhao, 2016) generally take the same framework except using neural networks as automatic feature extractor.

Lattice LSTM was proposed by Zhang and Yang (2018) for Chinese named entity recognition (NER). It integrates the character sequence features and all lexicon word embeddings that match a character subsequence in the input into a sequence labeling model. Zhu et al. (2016) proposed a DAG-structured LSTM structure which is similar to the lattice LSTM model but binarizing the paths in the merging process. Chen et al. (2017) also built a DAG-LSTM structure for word segmentation but without memory cells. Our model consistently gives better performance.

BPE is a data compression algorithm (Gage, 1994) which has been used in neural machine translation (NMT) by capturing the most frequent subwords instead of words (Sennrich et al., 2016). Here we use it for collecting subwords in Chinese, similar to the use in Chinese NMT.

3 Models

We take the state-of-the-art LSTM-CRF framework as our baseline. For an input sentence with m characters $s = c_1, c_2, \ldots, c_m$, where c_i denotes the ith character, the segmentor is to assign each character c_i with a label l_i. Figure 2 shows the segmentor framework on input character sequence "中国科学院院士 (Fellow of the Chinese Academy of Sciences)", where the black part represents the baseline LSTM-CRF model and the red part shows the lattice structure.

3.1 Baseline Model

As shown in Figure 2, for each input character c_i, the corresponding character unigram embeddings and character bigram embeddings are represented as $\mathbf{e}_{c_i}$ and $\mathbf{e}_{c_i c_{i+1}}$, respectively. The character representation is calculated as following:

$$\mathbf{x}_i = \mathbf{e}_{c_i} \oplus \mathbf{e}_{c_i c_{i+1}}, \qquad (1)$$

where $\oplus$ represents *concatenate* operation.

Unlike Zhang et al. (2016) which uses a window to strengthen the local features, or Zhou et al. (2017) which adds a non-linear layer before the LSTM layer, we feed $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ into a bidirectional LSTM:

$$\begin{aligned} \overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_m &= \overrightarrow{LSTM}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \\ \overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_m &= \overleftarrow{LSTM}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m), \end{aligned} \qquad (2)$$

where $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ represent the forward and backward LSTM, respectively. The detailed equations are listed in *Appendix*. The hidden vector of character c_i is

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i \qquad (3)$$

3.2 Lattice LSTM

The lattice LSTM adds "shortcut paths" (red part in Figure 2) to LSTM. The input of the lattice LSTM model is character sequence and all subsequences which are matched words in a lexicon $\mathbb{D}$, collected from BPE. Following Zhang and Yang (2018), we use $w_{b,e}$ to represent the subsequence that has a start character index b and a end character index e, and the embeddings of the subsequence is represented as $\mathbf{e}_{w_{b,e}}$.

During the forward lattice LSTM calculation, the "cell" in Figure 2 of a subsequence $w_{b,e}$ takes the hidden vector of the start character $\mathbf{h}_b$ and the subsequence embeddings $\mathbf{e}_{w_{b,e}}$ as input, an extra

Parameter	Value	Parameter	Value
char emb size	50	bigram emb size	50
word emb size	50	subword emb size	50
char dropout	0.5	lattice dropout	0.5
LSTM layer	1	LSTM hidden	200
learning rate *lr*	0.01	*lr* decay	0.05

Table 1: Hyper-parameter values.

LSTM cell is applied to calculate the memory vector of the sequence $\mathbf{c}_{w_{b,e}}$:

$$\mathbf{c}_{w_{b,e}} = LSTMCell(\mathbf{h}_b, \mathbf{e}_{w_{b,e}}), \tag{4}$$

where the $LSTMCell$ is a simplified LSTM unit which calculate the memory only. The output memory vector $\mathbf{c}_{w_{b,e}}$ links to the end character c_e to calculate its hidden vector $\overrightarrow{\mathbf{h}}_e$. For character with multiple memory cell inputs[2], we assign a gate for each subsequence input to control its contribution. The detailed equations are listed in *Appendix*. The final output $\overrightarrow{\mathbf{h}}_i$ includes both the character sequence history information and all the matched subsequence information.

3.3 Decoding and Training

We use a standard CRF layer for inference (details in *Appendix*). Viterbi (1967) is used to find the highest scored label sequence over the input. During training, we choose sentence-level log-likelihood as the loss function.

$$Loss = \sum_{i=1}^{N} log(P(y_i|s_i)), \tag{5}$$

where y_i is the gold labels of sentence s_i.

4 Experiments

4.1 Experimental Settings

Data. We evaluate our model on four standard Chinese word segmentation datasets: CTB6, PKU, MSR, and Weibo. PKU and MSR are taken from the SIGHAN 2005 bake-off (Emerson, 2005) and Weibo dataset is the NLPCC 2016 shared task (Qiu et al., 2016), standard split are used. We take CTB6 as the main dataset and split the train/dev/test following Zhang et al. (2016). The statistics of the datasets are listed in *Appendix*.

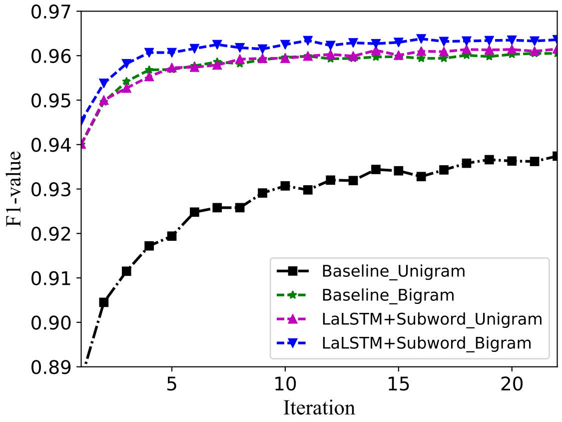


Figure 3: F1-value against training iterations.

Hyperparameters. We keep the hyperparameters the same among all datasets. Standard gradient descent (SGD) with a learning rate decay is used as the optimizer. The embedding sizes of character unigram/bigram and subword are all 50. Dropout (Srivastava et al., 2014) is used on both the character input and the subword input to prevent overfitting. Details are listed in Table 1.

Embeddings. We take the same character unigram and bigram embeddings as Zhang et al. (2016), who pretrain embeddings using word2vec (Mikolov et al., 2013) on Chinese Gigaword[3]. The vocabulary of subword is constructed with 200000 merge operations and the subword embeddings are also trained using word2vec (Heinzerling and Strube, 2018). Trie (Fredkin, 1960) is used to accelerate lattice building. All the embeddings are fine-tuned during training.

4.2 Development Experiments

We perform experiments on the CTB6 development dataset to investigate the contribution of character bigram information and the subword information. Figure 3 shows the iteration curve of F-scores against different numbers of training iterations with different character representations. “_Bigram” represents the model using both character unigram and bigram information (embedding concatenation). Character bigram information can improve the baseline significantly. When the “LaLSTM+Subword” structure is added, the model performance is further improved. This shows that subword information has a great ability to disambiguate the characters.

---

[2] e.g. The first “院” in Figure 2 takes two subsequence memory vectors of both “学院” and “科学院” as input.

[3] `https://catalog.ldc.upenn.edu/LDC2011T13`.

Models	CTB6	SIGHAN		Weibo
		MSR	PKU	
Zheng et al. (2013)	–	93.3	92.4	–
Pei et al. (2014)	–	97.2	95.2	–
Ma and Hinrichs (2015)	–	96.6	95.1	–
Liu et al. (2016)	95.5	97.6	95.7	–
Zhang et al. (2016)	96.0	**97.7**	95.7	–
Xu and Sun (2016)	95.8	96.3	**96.1**	–
Cai et al. (2017)	–	97.1	95.8	–
Chen et al. (2017)	95.6	96.1	–	–
Yang et al. (2017)†	95.4	96.8	95.0	**94.5**
Ma et al. (2018)	**96.7**	97.4	**96.1**	–
Baseline	95.8	97.4	95.3	95.0
LaLSTM+Subword	**96.1**	**97.8**	**95.8**	**95.3**

Table 2: Main results (F1).

Model	P	R	F1	ER%	R_{IV}	R_{OOV}
Baseline	95.93	95.62	95.78	0	96.70	77.36
Random Emb	96.13	95.82	95.97	-4.5	96.85	78.37
Pretrain Emb	96.23	95.90	96.07	-6.9	96.86	79.79

Table 3: Lexicon and embeddings on CTB6.

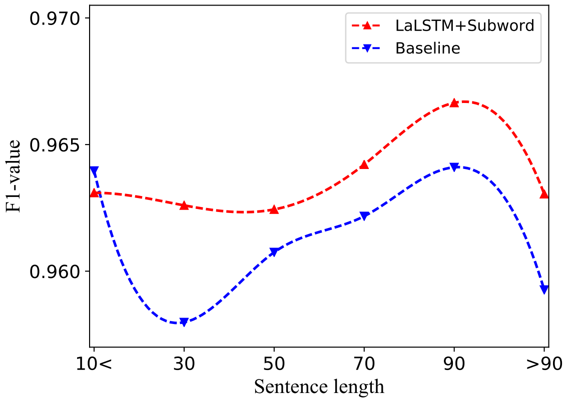


Figure 4: F1-value against the sentence length.

Zhang and Yang (2018) observed that character bigram information has a negative effect in lattice LSTM on Chinese NER task, while we find a different result on Chinese word segmentation where character bigram information gives significant improvements in the lattice LSTM. This is likely because character bigrams are informative but ambiguous. They can provide more useful character disambiguation evidence in segmentation than in NER where lattice LSTM works well in disambiguating characters.

4.3 Results

Table 2 shows the main results and the recent state-of-the-art neural CWS models. Zhang et al. (2016) integrated both discrete features and neural features in a transition-based framework. Xu and Sun (2016) proposed the dependency-based gated recursive neural network to utilize long distance dependencies. Yang et al. (2017)† utilized pretrained character representations from multitasks. We examine their non-pretrained model performance for fair comparison. Ma et al. (2018) built a bidirectional LSTM model with carefully hyperparameter selection. These methods are orthogonal to and can be integrated into our lattice structure.

As shown in Table 2, the subword lattice LSTM gives significant improvements on all evaluated datasets. In the PKU dataset, our model is slightly behind Xu and Sun (2016) which preprocesses the dataset by replacing all the Chinese idioms, leading the comparison not entirely fair. Our model gives the best performance on MSR and Weibo datasets, which demonstrates that subword encoding can help the lattice LSTM model gives comparable performance to the state-of-the-art word segmentation models.

4.4 Analysis

Lexicon and Embeddings. To distinguish the contribution of subword lexicon and their pretrained embeddings, we conduct a set of experiments by using the same subword lexicon with randomly initialized embeddings[4] on CTB6 data. As shown in Table 3, the contribution of the error reduction by the lexicon is 4.5%. While 6.9% error reduction comes from both lexicon and pretrained embeddings. We can estimate that the contribution of pretraining is $(6.9\% - 4.5\%) = 2.4\%$. This roughly shows that both lexicon and pretraining are useful to lattice LSTM, and the former contributes more than the latter.

OOV Analysis. Table 3 also shows the recall of in-vocabulary ($\mathbf{R}_{IV}$) and out-of-vocabulary ($\mathbf{R}_{OOV}$) words, respectively. As shown in the table, the $\mathbf{R}_{OOV}$ can be largely improved with the lattice structure (2.43% absolute improvement).

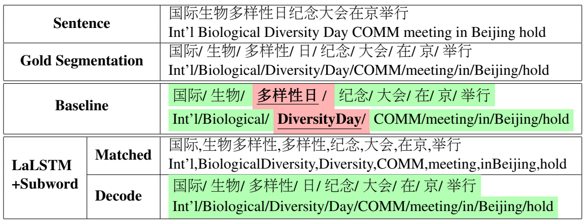
Sentence Length. We compare the baseline model with our proposed model on the sentence length distribution in Figure 4. The performance of the baseline has a valley in around 30-character length and decreases when the sentence length over 90. This phenomenon has also been observed in transition-based neural segmentor Yang et al. (2017). While ”LaLSTM+Subword” gives a more stable performance along sentence length.

---

[4] Within $[-\sqrt{\frac{3}{dim}}, \sqrt{\frac{3}{dim}}]$, dim is the embedding size.

Data	**Split**	**#Word**	**#Match**	**Ratio** (%)	**ER** (%)
CTB6	Train	641k	536k	83.57	–
	Test	81.6k	68.6k	84.13	7.14
MSR	Train	2.12m	1.93m	91.12	–
	Test	107k	98.2k	91.91	15.4
PKU	Train	1.01m	918k	90.87	–
	Test	104k	95.4k	91.42	10.6
Weibo	Train	421k	337k	80.10	–
	Test	188k	147k	78.39	6.0

Table 4: Subword coverage.



Sentence		国际生物多样性日纪念大会在京举行 Int'l Biological Diversity Day COMM meeting in Beijing hold
Gold Segmentation		国际/ 生物/ 多样性/ 日/ 纪念/ 大会/ 在/ 京/ 举行 Int'l/Biological/Diversity/Day/COMM/meeting/in/Beijing/hold
Baseline		🟩 国际/ 生物/ 🟥 多样性日 / 🟩 纪念/ 大会/ 在/ 京/ 举行 🟩 Int'l/Biological/ 🟥 **DiversityDay/** 🟩 COMM/meeting/in/Beijing/hold
LaLSTM +Subword	**Matched**	国际,生物多样性,多样性,纪念,大会,在京,举行 Int'l,BiologicalDiversity,Diversity,COMM,meeting,inBeijing,hold
	Decode	🟩 国际/ 生物/ 多样性/ 日/ 纪念/ 大会/ 在/ 京/ 举行 🟩 Int'l/Biological/Diversity/Day/COMM/meeting/in/Beijing/hold

Figure 5: Example.

Subword Coverage in lexicon. Table 4[5] shows the subword coverage rate in four datasets. Subword level coverage is consistently higher than the entity level coverage in Zhang and Yang (2018). We can see that higher subword coverage (PKU/MSR, $> 90\%$) gives better error reduction rate. Weibo dataset gets the minimum improvement due to the low subword coverage.

Case Study. Figure 5 shows an example of CTB6 test dataset. In this example, there are two matched subwords "生物多样性(BiologicalDiversity)" and "多样性(Diversity)" which can guide the segmentor to get the right split of "多样性日(DiversityDay)", which is segmented incorrectly by the baseline.

5 Conclusion

We examined the effectiveness of subwords for neural CWS. Subwords are deduced using BPE, and then integrated into a character-based neural segmentor through lattice LSTM. Results on four benchmarks show that subword brings significant improvements over a character baseline, and our proposed model gives comparable performances to the best systems on all datasets. Our experiments also showed that the matched subwords contribute more than embedding pertaining, which indicates that the lattice LSTM structure with domain lexicons can be useful for cross-domain segmentation training.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. Yue Zhang is the corresponding author.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 409–420.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. In *Proceedings of ACL*, pages 608–615.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of EMNLP*, pages 1385–1394.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Dag-based long short-term memory for neural word segmentation. *arXiv preprint arXiv:1707.00248*.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3(9):490–499.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

---

[5] **#Word** is the word number in the corresponding dataset, **#Match** is the matched words number between the dataset and subword lexicon, **#Ratio** $= \frac{\textbf{\#Match}}{\textbf{\#Word}}$ represents the subword coverage rate. **#ER** is the error reduction compared with baseline model.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art chinese word segmentation with bi-lstms. In *Proceedings of EMNLP*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of ACL-IJCNLP*, pages 1733–1743, Beijing, China.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*, pages 293–303. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the International Conference on Computational Linguistics*, page 562.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the nlpcc-iccpol 2016 shared task: Chinese word segmentation for micro-blog texts. In *International Conference on Computer Processing of Oriental Languages*, pages 901–906. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725, Berlin, Germany.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics*, 22(3):377–404.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Proceedings of the International Conference on Computational Linguistics*, pages 1211–1219.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 567. Association for Computational Linguistics.

Nianwen Xue et al. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 839–849, Vancouver, Canada.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. In *Proceedings of HLT-NAACL*, pages 193–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 45, page 840.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proceedings of PACLIC*, volume 20, pages 87–94. Citeseer.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of EMNLP*, pages 647–657.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of EMNLP*, pages 760–766, Copenhagen, Denmark.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2016. Dag-structured long short-term memory for semantic compositionality. In *Proceedings of NAACL-HLT*, pages 917–926.

Improving Cross-Domain Chinese Word Segmentation with Word Embeddings

Yuxiao Ye[1]*, **Yue Zhang**[2]*†, **Weikang Li**[3], **Likun Qiu**[1]†, **Jian Sun**[1]
[1]Alibaba Group, [2]Peking University, [3]Westlake University
{yuxiao.yyx, likun.qlk, jian.sun}@alibaba-inc.com
yue.zhang@wias.org.cn
wavejkd@pku.edu.cn

Abstract

Cross-domain Chinese Word Segmentation (CWS) remains a challenge despite recent progress in neural-based CWS. The limited amount of annotated data in the target domain has been the key obstacle to a satisfactory performance. In this paper, we propose a semi-supervised word-based approach to improving cross-domain CWS given a baseline segmenter. Particularly, our model only deploys word embeddings trained on raw text in the target domain, discarding complex hand-crafted features and domain-specific dictionaries. Innovative subsampling and negative sampling methods are proposed to derive word embeddings optimized for CWS. We conduct experiments on five datasets in special domains, covering domains in novels, medicine, and patent. Results show that our model can obviously improve cross-domain CWS, especially in the segmentation of domain-specific noun entities. The word F-measure increases by over 3.0% on four datasets, outperforming state-of-the-art semi-supervised and unsupervised cross-domain CWS approaches with a large margin. We make our code and data available on Github.

1 Introduction

Chinese Word Segmentation (CWS) is the first step for many Chinese Natural Language Processing (NLP) tasks (Cai and Zhao, 2016; Zhao et al., 2017). Approaches to CWS could be categorized into two categories: character-based and word-based. The former treats CWS as a sequence labeling problem, labeling each character in a sequence with *B/I/E/S* (*Beginning, Internal, End, Single*) labels (Tseng et al., 2005). Traditional character-based approaches often use Conditional Random Fields (CRF) models to label sequences, with complex hand-crafted discrete features (Peng et al., 2004; Tseng et al., 2005). Unlike character based CWS, word-based CWS operates on a word-level, directly exploiting word-level features. Typical CRF models are replaced with semi-CRF models, in which labels are assigned to subsequences instead of characters (Sarawagi and Cohen, 2005; Liu et al., 2014). Transition-based approaches have also been used to exploit larger feature contexts (Zhang and Clark, 2007). More recent approaches exploit neural networks including Recurrent Neural Networks (RNN) to replace hand-crafted discrete features with real-valued features (Cai and Zhao, 2016; Chen et al., 2015, 2017).

Existing studies have achieved satisfactory results for in-domain CWS, with F-scores over 96.0% in the newspaper domain (Chen et al., 2017). Nevertheless, cross-domain CWS remains a big challenge (Liu et al., 2014; Liu and Zhang, 2012). The main reason is the lack of annotated data in the target domain, which makes supervised approaches less useful. To tackle this problem, some unsupervised and semi-supervised approaches have been proposed. One way is to exploit complex features including character types, lexical features and accessor varieties (Wu et al., 2014), which requires much efforts on feature engineering. Another way is to deploy machine learning algorithms including self-training and model ensemble (Gao and Stephan, 2010; Liu and Zhang, 2012; Qiu and Zhang, 2015), which is time-consuming and inefficient.

In this paper, we investigate a different approach to deploying unsupervised data for cross-domain CWS, in order to completely break free from the reliance on manual annotation, complex feature engineering, and even parametric training to some extent. We propose a Word-Embedding-

---

[*] Yuxiao Ye and Yue Zhang contributed equally to this work.

[†] Corresponding authors.

Proceedings of NAACL-HLT 2019, pages 2726–2735
Minneapolis, Minnesota, June 2 - June 7, 2019.

Based CWS (WEB-CWS) model, which aims to improve the performance of an existing baseline segmenter in cross-domain CWS. WEB-CWS is a conceptually simple word-based model, using word embeddings, which are expected to carry semantic and syntax information (Mitchell and Lapata, 2010), as the only input of a non-parametric word segmentor. The basic intuition is that embeddings of words within a same context window should be close to each other (Goldberg and Levy, 2014; Mikolov et al., 2013). If a sequence is incorrectly segmented, those incorrectly segmented words are likely to be semantically and syntactically inconsistent with their surrounding words. Consequently, the embedding of an incorrectly segmented word should be far away from embeddings of its surrounding words.

Based on the hypothesis above, we propose WEB-CWS. Word embeddings are first derived with a CWS-oriented word embedding model with innovative subsampling and negative sampling methods. A word-embedding-based decoder is then used for segmentation, with cosine similarities among word embeddings as the metric for probability calculation. WEB-CWS is a *semi-supervised* model, because it only uses word embeddings trained on raw text in the target domain, which is first automatically segmented by the baseline segmenter. The model is also *cross-domain* in the sense that it can improve the performance of the baseline segmenter, when the source text for training the baseline segmenter and the target text to be segmented are in different domains.

The main contributions of this paper include:

- To our knowledge, we are the first to directly use word embeddings for CWS, without any neural structures, which makes our model conceptually simpler and run faster.
- We have proposed novel sampling methods to make the embeddings optimized for CWS, which has never been used for embedding training.
- Our model can be used on top of any existing CWS models to improve their performances, without the need to re-train those models with annotated domain specific data.
- On four datasets in different special domains, our model improves the word F-measure by more than 3.0%, compared with the state-of-the-art baseline segmenter. We release our code and data on Github[1].

2 Related Work

Our work is related with existing research on word-based CWS, cross-domain CWS, and embedding-based CWS.

2.1 Word-Based CWS

Instead of labeling a sequence character-wise, word-based CWS tries to pick the most probable segmentation of a sequence. Zhang and Clark (2007) design a statistical method for word-based CWS, extracting word-level features directly from segmented text. The perceptron algorithm (Collins, 2002) is used for training and beam-search is used for decoding. Cai and Zhao (2016) use Gated Combination Neural Networks and LSTM to present both character sequences and partially segmented word sequences, combining word scores and link scores for segmentation. Our work is in line with their work in directly using word information for CWS. In contrast, our method is conceptually simpler by directly using word embeddings. In addition, our work aims at domain-adaptation, rather than training from scratch.

2.2 Cross-Domain CWS

Supervised, semi-supervised and unsupervised approaches have been proposed for domain adaptation for CWS. Chen et al. (2017) use an Adversarial Network to learn shared knowledge for different segmentation criteria and domains. This approach requires annotated data in the target domain.

However, one challenge for cross-domain CWS is the lack of such annotated data. Liu and Zhang (2012) propose an unsupervised model, in which they use features derived from character clustering, together with a self-training algorithm to jointly model CWS and POS-tagging. This approach is highly time-consuming (Qiu and Zhang, 2015). Another challenge is the segmentation of domain-specific noun entities. In a task of segmenting Chinese novels, Qiu and Zhang (2015) design a double-propagation algorithm with complex feature templates to iteratively extract noun entities and their context, to improve segmentation

---

[1] https://github.com/vatile/CWS-NAACL2019

performance. This approach still relies heavily on feature templates. Similarly, our model does not require any annotated target data. In contrast to their work, our model is efficient and feature-free.

2.3 CWS Using Embeddings

There are CWS models deploying embeddings. Ma and Hinrichs (2015) and Deng and Sun (2018) propose embedding matching CWS models, in which embeddings of characters in a sequence are compared with high dimensional representations of CWS-specific actions (e.g., separation and combination) or CWS-specific labels (e.g., *B/M/E/S*). Then each character is labeled according to the similarity between its embedding and the high dimensional representation.

Particularly, Zhou et al. (2017) propose to use character embeddings trained on a word-based context to improve the performance of existing neural CWS models, which is similar to our approach in terms of making use of CWS-oriented word embeddings derived with automatically segmented raw corpus. However, in their work, when doing cross-domain CWS, word embeddings are fed into the baseline neural model trained on a large annotated general corpus, with annotated special domain data as the development set. In our model, on the contrary, word embeddings are used directly for CWS with a non-parametric decoder, which does not require to re-construct the baseline model, and annotation is not required at all for the special domain data.

3 Word-Embedding-Based CWS

The overall architecture of our method is shown in Figure 1. Given a baseline segmenter and a target domain raw corpus T, we obtain an automatically segmented corpus $T^{'}$ by applying the baseline segmenter to the target corpus. We then execute our CWS-oriented word embedding model (Section 3.1) on $T^{'}$ to derive a set of word embeddings E. In addition, all tokens from $T^{'}$ are collected as a target domain dictionary D. Finally, E and D are used to re-segment T with our word-embedding-based segmenter (Section 3.2).

3.1 CWS-Oriented Word Embedding Model

We use a CWS-oriented model modified from the Skip-gram model (Mikolov et al., 2013) to derive word embeddings. A typical Skip-gram model using negative sampling tries to maximize the following objective (Mikolov et al., 2013):

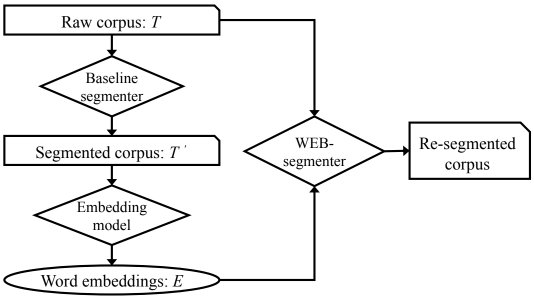


Figure 1: The pipeline of WEB-CWS.

$$\sum_{(w,c)\in P} \log \sigma(v_w \cdot v_c^{\top}) + \sum_{(w,c^{'})\in N} \log \sigma(-v_w \cdot v_{c^{'}}^{\top}) \tag{1}$$

with P being the set of positive samples (w, c) consisting of a target word w and a context word c, N being the set of negative samples $(w, c^{'})$ consisting of a target word w and a word $c^{'}$ drawn randomly from a noise distribution $P_n(w)$, v_w being the word embedding of w, and σ being the sigmoid activation function.

Subsampling is applied when choosing the target word w to reduce training time and to improve the quality of embeddings of rare words (Mikolov et al., 2013). For a natural language, the frequency distribution of all words is expected to obey Zipf's law: a word's frequency is inversely proportional to its rank in the frequency table (Newman, 2005). This highly biased distribution makes the training of the Skip-gram model inefficient, in that very frequent words can make a large portion of training samples, but their embeddings may not change much after being seen for a certain time (Mikolov et al., 2013). Therefore, a subsampling method is used by Mikolov et al. (2013), with the probability for a word w being sampled as:

$$p_{sub}(w) = min(1, \sqrt{\frac{\epsilon}{f(w)}}) \tag{2}$$

where ϵ is an arbitrarily chosen threshold, and $f(w)$ is the frequency of w.

Since word embeddings in our model are used for segmentation, we cannot directly use the training objective in Mikolov et al. (2013), which is designed for language modeling. To make the training objective more consistent with the goal of CWS, we modify the negative sampling Skip-gram model in various ways, including adding

CWS-oriented negative samples, changing the method for subsampling multi-character words, normalizing the dot product of embeddings, and smoothing the weights of positive and negative samples in training.

3.1.1 Context Negative Sampling

When training a typical Skip-gram model, a target word and a word within its context window are taken together as a positive sample (Mikolov et al., 2013). From the perspective of CWS, it can be perceived as teaching the model how to correctly segment a sequence. From this perspective, we develop a method to generate negative samples from a word's context (i.e., words within the context window), in order to tell the model what the incorrect segmentations of a sequence are.

Given a target word w and its context C, and SL/SR as the sequence of characters on the left/right of w within C, the proposed context negative sampling method generates negative samples in the following way: for any substring $s^{'}$ of SL and SR, if $s^{'}$ is in the dictionary D but not in C, (w, $s^{'}$) will be generated as a negative sample.

3.1.2 In-Word Negative Sampling

Another way to generate negative samples concerning CWS is to split multi-character words and combine its substrings as negative samples. For instance, given a multi-character target word $w = c_1c_2c_3$, supposing that all its substrings are in D, the proposed in-word negative sampling method will then generate the following negative samples: $(c_1, c_2), (c_1, c_3), (c_2, c_3), (c_1c_2, c_3)$ and (c_1, c_2c_3). By doing so, our model is expected to learn not to split those multi-character words when segmenting.

3.1.3 Subsampling Multi-Character Words

In the Chinese language, there are some frequent multi-character words consisting of substrings which are also very frequent words themselves. For example, the Chinese word 'dànshì (but)' can be decomposed into two words 'dàn (but)' and 'shì (be)'. Due to the nature of the Skip-gram model, embeddings of frequent words are relatively close to each other since they co-occur with other words more frequently. This nature makes our model inclined to split such multi-character words when segmenting. Although the in-word negative sampling method proposed above is expected to prevent our model from incorrectly splitting multi-character words, we still want our model to pay more attention to the segmentation of such words. As a result, for subsampling, we will not discard a multi-character word w if:

$$p_{sub}(w) < \frac{\mu}{N} \sum_{w_{sub} \in S \cap D} p_{sub}(w_{sub}) \tag{3}$$

where S is the set of substrings of w, and N is the size of all $w_{sub} \in S \cap D$, which is smoothed by a threshold μ (which is empirically set to 0.5 in our model). By doing so, we can keep those multi-character words whose substrings are more frequent words themselves. Our model is thus expected to learn better how to segment such words through samples generated with in-word negative sampling.

3.1.4 Dot Product Normalization

In the original Skip-gram model, the dot product of embeddings of two words is directly used as the input for the sigmoid layer (Mikolov et al., 2013). To make word embeddings derived from the CWS-oriented word embedding model more consistent with the metric used for segmentation as described in Section 3.2.2, we modify the training objective described in Equation (1) as follows:

$$\sum_{(w,c) \in P} \log \sigma(||v_w \cdot v_c^{\top}||) + \sum_{(w,c^{'}) \in N} \log \sigma(-||v_w \cdot v_{c^{'}}^{\top}||) \tag{4}$$

3.1.5 Smoothing Class Weights

For any target word, when training the CWS-oriented word embedding model, only one positive sample but many negative samples are generated. To balance the influence of positive and negative samples, a different weight is assigned to each class as follows:

$$class\_weight = \begin{cases} 1.0 & , positive \\ (\frac{N_{pos}}{N_{neg}} + \eta)/(1 + \eta) & , negative \end{cases} \tag{5}$$

where N_{pos} and N_{pos} are the amount of positive and negative samples respectively, and η is a smooth factor. The smooth factor can prevent the weight of negative samples being too low when negative samples are much more than positive samples.

3.2 Word-Embedding-Based Segmentater

In WEB-CWS, we formalize the process of segmenting a sequence as a problem of hypotheses-based Viterbi decoding (Forney, 1973): given a sequence, generating segmentation hypotheses character-wise from the first to the last character, and then searching for the optimal path according to the predefined metric of probability.

3.2.1 Hypothesis Generation

Given a sentence consisting of n characters $S =< c_0 > c_1c_2...c_n < c_{n+1} >$ (c_0 and c_{n+1} are markers of the beginning/end of a sentence), we generate segmentation hypotheses character-wise from c_0 to c_{n+1}. At each time step t, a hypothesis h_t is defined as:

$$h_t = \begin{cases} SEG_t : [w_{0\{c_0\}} w_{1\{c_1...c_g\}} ... w_{m\{c_j...c_k\}}] \\ BUF_t : [c_{k+1}...c_t] \\ M_t = m + 1 \end{cases} \tag{6}$$

which includes a partial segmentation container SEG_t, a buffer container BUF_t, and the number of segmented words M_t. In h_t, characters $c_0c_1...c_k$ are segmented into words $w_0w_1...w_m$ stored in SEG_t; characters $c_{k+1}..c_t$ remain unsegmented and are stored in BUF_t. For the initial hypothesis h_0 and the final hypothesis h_{n+1}, the buffer container will be empty.

Given a character c_{t+1} and a hypothesis h_t, h_{t+1} can be generated in two ways, by either appending c_{t+1} to BUF_t ($t \neq n$), or first moving the sequence in BUF_t into SEG_t as a new word, and then appending c_{t+1} to BUF_t ($t \neq n$). In the former case:

$$h_{t+1} = \begin{cases} SEG_t : [w_{0\{c_0\}} ... w_{m\{c_j...c_k\}}] \\ BUF_t : [c_{k+1}...c_t c_{t+1}] \\ M_t = m + 1 \end{cases} \tag{7}$$

In the latter case:

$$h_{t+1} = \begin{cases} SEG_t = w_{0\{c_0\}} ... w_{m\{c_j...c_k\}} w_{m+1\{c_{k+1}...c_t\}} \\ BUF_t = c_{t+1} \\ M_t = m + 2 \end{cases} \tag{8}$$

Particularly, when generating the hypothesis for c_{n+1} (the end of sentence marker), the sequence in BUF_n has to be moved into SEG_n, and c_{n+1} also needs to be moved into SEG_n as a new word.

If a word w is not in the dictionary D, we cannot get its embedding. As a result, any hypothesis containing w in the segmentation container will be discarded. Moreover, to reduce search space, once the size of the sequence in a buffer container reaches a threshold m (i.e., the maximum word length), this sequence will be moved into the segmentation container when generating hypotheses at the next time step.

3.2.2 Probability Calculation

The log probability of a hypothesis h_t is defined as:

$$\log p(h_t) = \begin{cases} 0, & t = 0 \\ \frac{1}{M_t - 1} \log \prod_{i=1}^{M_t - 1} p(w_i | SEG_t, f), & t \neq 0 \end{cases} \tag{9}$$

where f is the window size (e.g., if $f = 2$, $p(w_i|SEG_t, f)$ will be decided by w_{i-1} and w_{i-2}).

In WEB-CWS, we use cosine similarity between embeddings of two words as the metric of probability. Given a hypothesis h_t, $p(w_i|SEG_t, f)$ is calculated as follows:

$$p(w_i|SEG_t, f) = \begin{cases} e^0, & i = 0 \\ e^{\frac{1}{min(f,i)} \sum\limits_{j=1}^{min(f,i)} \cos(v_{w_i}, v_{w_{i-j}})}, & i \neq 0 \end{cases} \tag{10}$$

where $\cos(v_{w_i}, v_{w_{i-j}})$ refers to the cosine similarity between embeddings of w_i and w_{i-j}.

Given a hypothesis h_t, the log probability of h_{t+1} can be dynamically computed as:

$$\begin{aligned} \log p(h_{t+1}) = & \frac{1}{M_{t+1} - 1}((M_t - 1) \log p(h_t) \\ & + \sum_{w_i \in \{SEG_{t+1} - SEG_t\}} \log p(w_i | SEG_{t+1}, f)) \end{aligned} \tag{11}$$

3.2.3 Dynamic Beam-Size and Maximum Word Length

Theoretically, a sequence of length n can have at most 2^{n-1} possible segmentations. By discarding hypotheses containing out-of-vocabulary (OOV) words and setting the maximum word length, the search space can be significantly reduced. The very limited search space makes dynamically deciding the beam-size and the maximum word length possible.

Given an initial beam-size k, at each time step t, the segmenter will only keep at most top k hypothesis sorted by log probabilities in a descending order. Some hypotheses will also be discarded due to OOV words and the maximum word length limit. As a result, it is sometimes possible for a sequence to have no hypothesis at all after some time steps. Once it happens, the segmenter will increase the beam-size by 10 and the maximum word length by 1, and then re-generate hypothesis from the beginning, till at least one hypothesis is generated at the final time step.

Dynamic beam-size and maximum word length ensure that for each sequence, at least one segmentation (the one given by the baseline segmenter) will be generated as the final segmentation result. This mechanism can guarantee the efficiency and reliability of the segmenter at the same time.

3.2.4 Similarity Score Look-up

To improve the decoding speed, we pre-calculate the cosine similarity of all word pairs co-occurring at least once in the automatically segmented corpus, and store them in a file (which only consists of millions of word pairs for a Chinese novel). In doing so, when decoding, for most word pairs, we only need to look up to this file for similarity scores, which can significantly improve the decoding speed. According to later experiments, this look-up strategy can cover about 92% of the similarity calculation needed for decoding.

4 Experiments

We conduct experiments on various datasets in different domains to thoroughly evaluate the performance of our model.

4.1 Experimental Setup

4.1.1 Datasets

We evaluate our model in terms of cross-domain CWS on five datasets, including three Chinese novel datasets (Qiu and Zhang, 2015): DL (*DouLuoDaLu*), FR (*FanRenXiuXianZhuan*) and ZX (*ZhuXian*), and two CWS datasets in special domains (Qiu et al., 2015): DM (dermatology) and PT (patent). We use the standard split for all datasets as they are published. Raw test data is also included for deriving word embeddings.

Statistics of these datasets are shown in Table 1. Since there are no gold segmentation of full novels for three Chinese novel datasets, their statistics are based on the segmentation given by the baseline segmenter.

Dataset	Sentence (K)		Token (K)		Character (K)	
	Full	Eval	Full	Eval	Full	Eval
DL	40	1	1,982	32	2,867	47
FR	148	1	5,004	17	7,126	25
ZX	59	1	2,131	21	3,006	31
DM	32	1	709	17	1,150	30
PT	17	1	556	34	903	57

Table 1: Statistics of full and evaluation datasets.

4.1.2 Pre-Processing

In some studies, pre-processing is applied in order to improve the performance of CWS models, including substituting consecutive digits and English letters, Chinese idioms and long words with unique symbols (Cai and Zhao, 2016; Cai et al., 2017; Chen et al., 2015). However, we do not deploy such techniques for fair comparison, focusing only on the possible improvements brought by word embeddings. The only pre-processing adopted in our model is to first split a sentence with a set of pre-defined delimiters: characters that are not Chinese characters, English letters or digits. Those fragments of a sentence are then fed into the segmenter, and a complete segmented sentence is returned by reassembling the segmented fragments and delimiters in the original order.

4.1.3 Hyperparameters

Hyperparameters used in our WEB-CWS model are explained and their values are displayed in Table 2. All hyperparameters are tuned on a small excerpt of ZX, which consists of 300 sentences (Qiu and Zhang, 2015). It is worth noting that, according to Mikolov et al. (2013), for each positive sample, the optimal number of negative samples drawn from a noise distribution is usually between 5 to 20. However, in our model, we find that, for each target word, drawing one negative sample from a noise distribution is good enough, which may be caused by the large amount of negative samples generated by context and in-word negative sampling. Also, Mikolov et al. (2013) report that the unigram distribution raised to the 3/4ths power is better than the uniform distribution for negative sampling. But in WEB-CWS, using the uniform distribution leads to better segmentation results.

H-param	Explanation	Value
ϵ	threshold for overall subsampling	10^{-5}
μ	threshold for multi-character word subsampling	0.5
P_n	noise distribution for general negative sampling	uniform
n	number of general negative samples per word	1
d	dimension of word embeddings	100
η	smoothing factor for class weights	0.2
f	window size	4
m	initial maximum word length	5
k	initial beam-size	10
e	number of epochs in Skip-gram training	1

Table 2: Hyperparameters used in WEB-CWS.

4.1.4 Evaluation

For consistency, all segmentation results are automatically calculated with the script provided in the SIGHAN Bakeoff (Emerson, 2005) and are reported as word F-measures.

4.2 Baseline Segmenter

Two state-of-the-art CWS models trained on a People's Daily corpus in 2000 January are tested. One is a joint word segmentation and POS-tagging model (Zhang and Clark, 2010), and the other is a word-based neural CWS model (Cai et al., 2017). When training both models, default settings are used, except that the maximum word length in Cai et al.'s model is set to 5, which is in line with the setting of WEB-CWS.

On the evaluation set of PKU (Emerson, 2005), both models yield comparable results, but on the evaluation set of DL, Zhang and Clark's model (F-measure = 0.905) performs better than Cai et al.'s model (F-measure = 0.849). It is very possible that Zhang and Clark's model can handle cross-domain CWS more effectively. As a result, we choose Zhang and Clark's model as the baseline segmenter for following experiments.

5 Results

Results in Table 3 show that our WEB-CWS model can obviously improve CWS on four datasets in special domains, including DL, FR, ZX and DM, with an increase of over 3.0% in F-measure. Those four datasets are all in domains (novel and dermatology) which are very different from that of the baseline segmenter (newspaper). This result suggests that WEB-CWS can effectively improve cross-domain CWS.

However, on another dataset in the special domain PT, the improvement is not significant. There are two possible reasons for this result. First, the size of PT is the smallest among all datasets, which may make the quality of word embeddings unsatisfactory. Second, the PT dataset contains a huge amount of decimal points (e.g., '3.14'), percentage signs (e.g., '28%'), hyphens (e.g., 'pMIV-Pnlp') and very long English strings (e.g., 'agctgagtcg'), which are all cases that cannot be handled by WEB-CWS without corresponding preprocessing techniques.

5.1 Comparison with State-of-the-Art Models

We also compare WEB-CWS with two state-of-the-art semi-supervised and unsupervised cross-domain CWS models by Qiu and Zhang (2015) and Liu and Zhang (2012), both of which use the same baseline model proposed by Zhang and Clark (2010) as used in our model. We adopt the method of combining character clustering and self-training in Liu and Zhang (2012) with datasets in our experiments. Results of the model in Qiu and Zhang (2015) are directly copied from the corresponding paper.

Results in Table 3 show that WEB-CWS outperforms these two state-of-the-art models with a large margin in terms of F-measure. Particularly, on the DM dataset, Liu and Zhang's model only achieves a relatively low F-score improvement rate (1.8%), which is likely to be caused by the large difference between the source and target domains. This result suggests that WEB-CWS is more robust to domain dissimilarity compared with self-training.

5.2 Run Time for Decoding

We test the run time for our decoder on a 3.5 GHz Intel Core i7 CPU. On all five test sets, the average decoding speed is 20.3 tokens per millisecond, when the initial beam-size is set to 10. In the work of Zhou et al. (2017), the decoding speed of their model is 14.7 tokens per millisecond for greedy segmentation. However, these results cannot be compared directly since they are produced on different machines. Our similarity look-up strategy is proved to be efficient in improving the decoding speed.

6 Analysis and Discussion

In order to assess the effect of negative sampling and subsampling methods, we conduct a series of

Dataset	Baseline (%)	WEB-CWS (%)	IR_WEB (%)	IR_Qiu&Zhang (%)	IR_Liu&Zhang (%)
DL	90.5	**93.5**	**+3.3**	+1.6	+2.1
FR	85.9	**89.6**	**+4.3**	+2.9	+2.5
ZX	86.8	**89.6**	**+3.2**	+2.2	+2.4
DM	77.9	**82.2**	**+5.5**	-	+1.8
PT	84.6	**85.1**	**+0.6**	-	+0.2

Table 3: F-measures of the baseline segmenter and WEB-CWS on datasets in special domains, and F-measure improvement rates (IR) of WEB-CWS, Qiu and Zhang (2015) and Liu and Zhang (2012).

Model	DL (%)	FR (%)	DM (%)
Baseline	90.5	85.9	77.9
WEB-basic	76.2	72.2	66.6
WEB + c_n	92.7	89.3	75.8
WEB + w_n	93.0	89.5	76.2
WEB + c_w_n	93.3	89.2	79.7
WEB + c_w_n + m_s	**93.5**	**89.6**	**82.2**

Table 4: F-measures of WEB-CWS with different subsampling and negative sampling methods.

Word	Meaning	Pre_base (%)	Pre_WEB (%)
点头	nod	0.0	**84.6**
兽武魂	**beast warrior spirit**	18.2	**100.0**
糖皮质激素	**glucocorticoid**	3.8	**76.9**
蛮胡子	**person name**	28.2	**97.4**
虚天鼎	**person name**	25.7	**94.3**
一对一	one on one	33.3	**100.0**
见于	to be seen in	34.5	**100.0**
鸡冠蛇	**cockscomb snake**	8.7	**73.9**
皮损	**skin lesions**	28.6	**92.9**
宁荣荣	**person name**	34.9	**98.8**

Table 5: Ten most improved words in terms of segmentation precision. Pre_base: segmentation precision by the baseline segmenter. Pre_WEB: segmentation precision by WEB-CWS.

ablation experiments. A detailed analysis is presented to understand in what way WEB-CWS can improve cross-domain CWS. All experiments and analyses in this section are carried out on three datasets with most significant improvements in F-measure: DL, FR and DM.

6.1 Ablation Experiments

In ablation experiments, we study the influence of two CWS-oriented negative sampling and multi-character words subsampling. Results in Table 4 show that WEB-CWS using word embeddings derived with the basic Skip-gram model ('*basic*') performs obviously worse than the baseline segmenter. When CWS-oriented negative sampling is applied alone, either context ('*c_n*') or in-word ('*w_n*') negative sampling, the performance of WEB-CWS is obviously better than or similar to that of the baseline segmenter. When both CWS-oriented negative sampling methods are applied together ('*c_w_n*'), WEB-CWS is ensured to obviously outperform the baseline segmenter. Also, when multi-character subsampling ('*m_s*') is applied, the performance of WEB-CWS can further improve a little.

6.2 Improvements in Noun Entity Segmentation

To see which words are incorrectly segmented by the baseline segmenter but correctly by WEB-CWS, all words occurring at least ten times in the three datasets are sorted in a descending order, by improvements in terms of segmentation precision. Table 5 displays the ten most improved words. As shown in Table 5, among the ten most improved words, seven words are domain-specific noun entities, including person names, disease names and chemical compound names. For some noun entities (e.g., *glucocorticoid*), even if the baseline segmenter can rarely segment them correctly, WEB-CWS can still find the correct segmentation in most cases. This result suggests that WEB-CWS is especially effective in segmenting domain-specific noun entities.

7 Conclusion

We have proposed WEB-CWS, a semi-supervised model that can be used to effectively improve cross-domain CWS. Our model only requires a baseline segmenter and a raw corpus in the target domain, deploying only word embeddings for CWS. WEB-CWS obviously improves the performance of the state-of-the-art baseline segmenter on four datasets in special domains, especially in segmenting domain-specific noun entities.

Acknowledgments

This paper was partially supported by the National Natural Science Foundation of China (No.

61572245). Thanks to You Wang et al. and the anonymous reviewers for their constructive and insightful comments on this paper.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. *arXiv preprint arXiv:1606.04300*.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. *arXiv preprint arXiv:1704.07047*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Xiaolong Deng and Yingfei Sun. 2018. An improved embedding matching model for chinese word segmentation. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Qin Gao and Vogel Stephan. 2010. A multi-layer chinese word segmentation system optimized for out-of-domain tasks. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

Yoav Goldberg and Omer Levy. 2014. Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. *Proceedings of COLING 2012: Posters*, pages 745–754.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1733–1743.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Mark EJ Newman. 2005. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, page 562. Association for Computational Linguistics.

Likun Qiu, Linlin Shi, and Houfeng Wang. 2015. Construction of chinese dependency tree-bank and analysis of factors in statistical parsing in various domains. *Journal of Chinese Information Processing*, 29(5):69–76.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*, pages 2440–2446.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Guohua Wu, Dezhu He, Keli Zhong, Xue Zhou, and Caixia Yuan. 2014. Leveraging rich linguistic features for cross-domain chinese segmentation. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 101–107.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 840–847.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852. Association for Computational Linguistics.

Hai Zhao, Deng Cai, Yang Xin, Yuzhu Wang, and Zhongye Jia. 2017. A hybrid model for chinese spelling check. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):21.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766.

Neural Semi-Markov Conditional Random Fields for Robust Character-Based Part-of-Speech Tagging

Apostolos Kemos[2*] **Heike Adel**[1,3*] **Hinrich Schütze**[1]
[1] Center for Information and Language Processing (CIS), LMU Munich, Germany
[2] Department of Computer Engineering and Informatics, University of Patras, Greece
[3] Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany
kemos@ceid.upatras.gr heike.adel@de.bosch.com
inquiries@cislmu.org

Abstract

Character-level models of tokens have been shown to be effective at dealing with within-token noise and out-of-vocabulary words. However, they often still rely on correct token boundaries. In this paper, we propose to eliminate the need for tokenizers with an end-to-end character-level semi-Markov conditional random field. It uses neural networks for its character and segment representations. We demonstrate its effectiveness in multilingual settings and when token boundaries are noisy: It matches state-of-the-art part-of-speech taggers for various languages and significantly outperforms them on a noisy English version of a benchmark dataset. Our code and the noisy dataset are publicly available at `http://cistern.cis.lmu.de/semiCRF`.

1 Introduction

Recently, character-based neural networks (NNs) gained popularity for different tasks, ranging from text classification (Zhang et al., 2015) and language modeling (Kim et al., 2016) to machine translation (Luong and Manning, 2016). Character-level models are attractive since they can effectively model morphological variants of words and build representations even for unknown words, suffering less from out-of-vocabulary problems (Pinter et al., 2017).

However, most character-level models still rely on tokenization and use characters only for creating more robust token representations (Santos and Zadrozny, 2014; Lample et al., 2016; Ma and Hovy, 2016; Plank et al., 2016). This leads to high performance on well-formatted text or text with misspellings (Yu et al., 2017; Sakaguchi et al., 2017) but ties the performance to the quality of the tokenizer. While humans are very robust to noise caused by insertion of spaces (e.g., “car nival”) or deletion of spaces (“deeplearning”), this can cause severe underperformance of machine learning models. Similar challenges arise for languages with difficult tokenization, such as Chinese or Vietnamese. For text with difficult or noisy tokenization, more robust models are needed.

In order to address this challenge, we propose a model that does not require any tokenization. It is based on semi-Markov conditional random fields (semi-CRFs) (Sarawagi and Cohen, 2005) which jointly learn to segment (tokenize) and label the input (e.g., characters). To represent the character segments, we compare different NN approaches.

In our experiments, we address part-of-speech (POS) tagging. However, our model is generally applicable to other sequence-tagging tasks as well since it does not require any task-specific hand-crafted features. Our model achieves state-of-the-art results on the Universal Dependencies dataset (Nivre et al., 2015). To demonstrate its effectiveness, we evaluate it not only on English but also on languages with inherently difficult tokenization, namely Chinese, Japanese and Vietnamese. We further analyze the robustness of our model against difficult tokenization by randomly corrupting the tokenization of the English dataset. Our model significantly outperforms state-of-the-art token-based models in this analysis.

Our contributions are: 1) We present a truly end-to-end character-level sequence tagger that does not rely on any tokenization and achieves state-of-the-art results across languages. 2) We show its robustness against noise caused by corrupted tokenization, further establishing the importance of character-level models as a promising research direction. 3) For future research, our code and the noisy version of the dataset are publicly available at `http://cistern.cis.lmu.de/semiCRF`.

---

[*] Work was done at Center for Information and Language Processing, LMU Munich.

Proceedings of NAACL-HLT 2019, pages 2736–2743
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Model

This section describes our model which is also depicted in Figure 1.

2.1 Character-based Input Representation

The input to our model is the raw character sequence. We convert each character to a one-hot representation. Out-of-vocabulary characters are represented with a zero vector. Our vocabulary does not include the space character since there is no part-of-speech label for it. Instead, our model represents space as two "space features" (lowest level in Figure 1): two binary dimensions indicate whether the previous or next character is a space. Then, a linear transformation is applied to the extended one-hot encoding to produce a character embedding. The character embeddings are fed into a bidirectonal LSTM (biLSTM) (Hochreiter and Schmidhuber, 1997) that computes context-aware representations. These representations form the input to the segment-level feature extractor.

2.2 Semi-Markov CRF

Our model partitions a sequence of characters $x = \{x_1, \dots, x_T\}$ of length T, into (token-like) segments $s = \{s_1, \dots, s_{|s|}\}$ with $s_j = \langle a_j, d_j, y_j \rangle$ where a_j is the starting position of the j^{th} segment, d_j is its length and y_j is its label. Thus, it assigns the same label y_j to the whole segment s_j. The sum of the lengths of the segments equals the number of non-space characters: $\sum_{j=1}^{|s|} d_j = T$.[1]

The semi-CRF defines the conditional distribution of the input segmentations as:

$$p(s|x) = \frac{1}{Z(x)} \exp\left(\sum_{j=1}^{|s|} F(s_j, x) + A(y_{j-1}, y_j)\right)$$

$$Z(x) = \sum_{s' \in S} \exp\left(\sum_{j=1}^{|s'|} F(s'_j, x) + A(y'_{j-1}, y'_j)\right)$$

where $F(s_j, x)$ is the score for segment s_j (including its label y_j), and $A(y_{t-1}, y_t)$ is the transition score of the labels of two adjacent segments. Thus, $p(s|x)$ jointly models the segmentation and label assignment. For the normalization term $Z(x)$, we sum over the set of all possible segmentations S.

The score $F(s_j, x)$ is computed as:

$$F(s_j, x) = \mathbf{w}_{y_j}^{\top} f(s_j, x) + b_{y_j}$$

where $W = (\mathbf{w}_1, \dots, \mathbf{w}_{|Y|})^{\top} \in \mathbb{R}^{|Y| \times D}$ and $b = (b_1, \dots, b_{|Y|})^{\top} \in \mathbb{R}^{|Y|}$ are trained parameters, $f(s_j, x) \in \mathbb{R}^D$ is the feature representation of the labeled segment s_j, $|Y|$ is the number of output classes and D is the length of the segment representation.

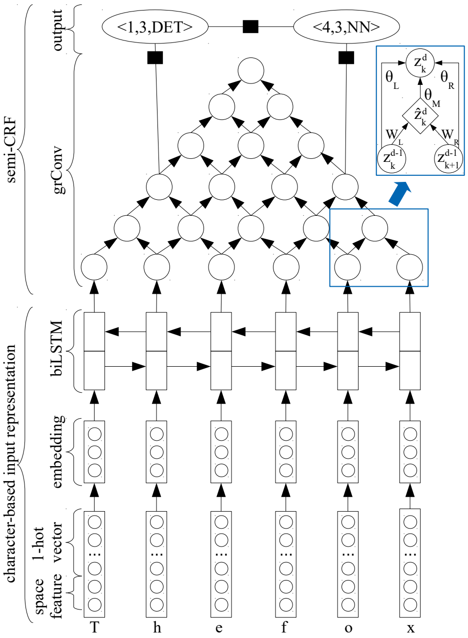


Figure 1: Overview of our model. Illustration of gating for grConv taken from (Zhuo et al., 2016).

For training and decoding, we use the semi-Markov analogies of the forward and Viterbi algorithm, respectively (Sarawagi and Cohen, 2005). In order to avoid numerical instability, all computations are performed in log-space.

2.2.1 Segment-level Features

Sarawagi and Cohen (2005) and Yang and Cardie (2012) compute segment-level features by handcrafted rules. Recent work learns the features automatically with NNs (Kong et al., 2015; Zhuo et al., 2016). This avoids the manual design of new features for new languages/tasks. We adopt ***Gated Recursive Convolutional Neural Networks* (grConv)** (Cho et al., 2014; Zhuo et al., 2016) since they allow to hierarchically combine features for segments. We argue that this is especially useful for compositionality in language. An example is the word "airport" which can be composed of the segments "air" and "port".

GrConv constructs features by recursively combining adjacent segment representations in a pyramid shape way (see Figure 1). The d^{th} level of the pyramid consists of all representations for segments of length d. The first level holds the char-

---

[1] For efficiency, we define a maximum segment length L: $d_j < L, 1 \leq j \leq |s|$. L is a hyperparameter. We choose it based on the observed segment lengths in the training set.

acter representations from our biLSTM. The representation $z_k^{(d)} \in \mathbb{R}^D$, stored in the k^{th} node of layer d, is computed as follows:

$$z_k^{(d)} = \theta_L \circ z_k^{(d-1)} + \theta_R \circ z_{k+1}^{(d-1)} + \theta_M \circ \hat{z_k}^{(d)}$$

$$\text{with } \hat{z_k}^{(d)} = g(W_L z_k^{(d-1)} + W_R z_{k+1}^{(d-1)} + b_w)$$

where $W_L, W_R \in \mathbb{R}^{D \times D}$ and $b_w \in \mathbb{R}^D$ are globally shared parameters, θ_L, θ_M and θ_R are gates, g is a non-linearity and $\circ$ denotes element-wise multiplication. The gates are illustrated in the blue box of Figure 1 and described in (Zhuo et al., 2016).

3 Experiments and Analysis

Our implementation is in PyTorch (Paszke et al., 2017). Hyperparameters are tuned on the development set. We use mini-batch gradient descent with a batch size of 20 and Adam (Kingma and Ba, 2014) as the optimizer. The learning rate is 1e-3, the coefficients for computing running averages of the gradient and its square are 0.9 and 0.999, respectively. A term of 1e-8 is added to the denominator for numerical stability. We use character embeddings of size 60 and three stacked biLSTM layers with 100 hidden units for each direction. For the semi-CRF, we set the maximum segment length to $L = 23$ as tokens of bigger length are rarely seen in the training sets. To avoid overfitting, we apply dropout with a probability of 0.25 on each layer including the input. For input dropout, we randomly replace a character embedding with a zero vector, similar to Gillick et al. (2016). This avoids overfitting to local character patterns. Moreover, we employ early stopping on the development set with a minimum of 20 training epochs. We run our experiments on a gpu which speeds up the training compared to multiple cpu cores considerably. We assume that it especially benefits from parallelizing the computation of each level of the grConv pyramid.

3.1 Multilingual Experiments on Clean Data

Data and Evaluation. To compare our model to state-of-the-art character-based POS taggers, we evaluate its accuracy on the English part of the Universal Dependencies (UD) v1.2 dataset (Nivre et al., 2015). For multilingual experiments, we use the English (EN), Chinese (ZH), Japanese (JA) and Vietnamese (VI) part of UD v2.0[2] (Nivre and Željko Agic, 2017), using the splits, training and evaluation rules from the CoNNL 2017 shared task (Zeman et al., 2017). In particular, we calculate joint tokenization and UPOS (universal POS) F_1 scores.

Model	$\vec{w}$	$\vec{c}$
MarMot	**94.36**	-
bilstm-aux	92.10	91.62
CNN Tagger	92.64	93.76
Our	-	**94.27**
Our without space feature	-	93.35
Our with SRNN	-	93.86

Table 1: POS tag accuracy on UD v1.2 (EN). '-' denotes that the model does not use this input.

Baselines for UD v1.2. We compare our model to two character-based models that are state of the art on UD v1.2: **bilstm-aux** (Plank et al., 2016) and **CNN Tagger** (Yu et al., 2017). We also compare to a state-of-the-art word-based CRF model **MarMot**[3] (Müller and Schütze, 2015).

Results on English (UD v1.2). Table 1 provides our results on UD v1.2, categorizing the models into token-level ($\vec{w}$) and character-only models ($\vec{c}$). While most pure character-level models cannot ensure consistent labels for each character of a token, our semi-CRF outputs correct segments in most cases (tokenization F_1 is 98.69%, see Table 4), and ensures a single label for all characters of a segment. Our model achieves the best results among all character-level models and comparable results to the word-level model MarMot.

In addition, we assess the impact of two components of our model: the space feature (see Section 2.1) and grConv (see Section 2.2.1). Table 1 shows that the performance of our model decreases when ablating the space feature, confirming that information about spaces plays a valuable role for English. To evaluate the effectiveness of grConv for segment representations, we replace it with a ***Segmental Recurrent Neural Network (SRNN)*** (Kong et al., 2015).[4] SRNN uses dynamic programming and biLSTMs to create segment representations. Its performance is slightly worse compared to grConv (last row of Table 1). We attribute

---

[2] UD v1.2 does not provide data for JA, VI, ZH.

[3] `http://cistern.cis.lmu.de/marmot/`

[4] In an initial experiment, we also replaced it with a simpler method that creates a segment representation by subtracting the character biLSTM hidden state of the segment start from the hidden state of the segment end. This is one of the segment-level features employed, for instance, by Ye and Ling (2018). However, this approach did not lead to promising results in our case. We assume that more sophisticated methods like grConv or SRNN are needed in this setup.

	UDPipe 1.2		Stanford		FBAML		TRL		IMS		Our	
	Tokens	POS	Tokens	POS	Tokens	POS	Tokens	POS	Tokens	POS	Tokens	POS
EN	**99.03**	93.50	98.67	**95.11**	98.98	94.09	94.31	82.41	98.67	93.29	98.79	93.45
JA	90.97	88.19	89.68	88.14	93.32	91.04	**98.59**	**98.45**	91.68	89.07	93.86	91.34
VI	84.26	75.29	82.47	75.28	83.80	75.84	85.41	74.53	86.67	**77.88**	**88.06**	77.67
ZH	89.55	83.47	88.91	85.26	**94.57**	**88.36**	83.64	71.31	92.81	86.33	93.82	88.15
Avg	90.95	85.11	89.93	85.95	92.67	87.33	90.49	81.68	92.46	86.64	**93.66**	**87.65**

Table 2: Tokenization and joint token-POS F_1 on UD v2.0. Best scores are in bold, second-best are underlined.

this to the different way of feature creation: While grConv hierarchically combines context-enhanced n-grams, SRNN constructs segments in a sequential order. The latter may be less suited for compositional segments like "airport".

Baselines for UD v2.0. We compare to the top performing models for EN, JA, VI, ZH from the CoNLL 2017 shared task: UDPipe 1.2 (Straka and Straková, 2017), Stanford (Dozat et al., 2017), FBAML (Qian and Liu, 2017), TRL (Kanayama et al., 2017), and IMS (Björkelund et al., 2017).

Multilingual Results (UD v2.0). Table 2 provides our results. While for each language another shared task system performs best, our system performs consistently well across languages (best or second-best except for EN), leading to the best average scores for both tokenization and POS tagging. Moreover, it matches the state of the art for Chinese (ZH) and Vietnamese (VI), two languages with very different characteristics in tokenization.

3.2 Analysis on Noisy Data

To further investigate the robustness of our model, we conduct experiments with different levels of corrupted tokenization in English. We argue that this could also give us insights into why it performs well on languages with difficult tokenization, e.g., on Chinese which omits spaces between tokens, or on Vietnamese which has spaces inside tokens, after each syllable. Note that we do not apply input dropout for these experiments, since the corrupt tokenization already acts as a regularizer.

Data. We are not aware of a POS tagging dataset with corrupted tokenization. Thus, we create one based on UD v1.2 (EN). For each token, we either delete the space after it with probability $P = p_d$ or insert a space between two characters with $P = p_i$: *"The fox chased the rabbit"* → *"The f ox cha sed therabbit"*. We vary p_d and p_i to construct three datasets with different noise levels (LOW, MID, HIGH, see Table 3). We note that there are more sophisticated ways of creating "errors" in text. An example is Kasewa et al. (2018) who generate grammatical errors. We leave the investigation of other methods for generating tokenization errors to future work.

level	p_d	# deletions	p_i	# insertions
LOW	0.1	15198	0.05	26497
MID	0.3	39361	0.11	40474
HIGH	0.6	65387	0.33	68209

Table 3: Noisy dataset statistics (three different noise levels).

Labeling. As mentioned before, we either delete the space after a token with probability p_d or insert a space between two of its characters with probability p_i. We assign the label from the original token to every sub-token created by space insertion. For space deletions, we randomly choose one of the two original labels for training and evaluate against the union of them. Figure 2 shows an example.

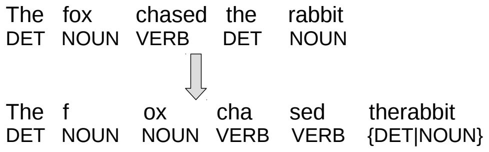


Figure 2: Example of label assignment.

Baseline. We compare our joint model to a traditional pipeline of tokenizer (UDpipe 1.0)[5] and token-level POS tagger (MarMot).[6] We re-train MarMot on the corrupted datasets.

Evaluation. We evaluate the models on the noisy datasets using two different metrics: (i) tokenization and joint token-POS F_1 as in Table 2, and (ii) a relaxed variant of POS tag accuracies. With the latter, we can assess the performance of MarMot without penalizing it for potential errors of UDpipe. For calculating the relaxed accuracy, we count the POS tag of a gold token as correct if MarMot predicts the tag for any subpart of it.

---

[5] http://lindat.mff.cuni.cz/services/udpipe/

[6] In contrast to Table 1 where we use gold tokens for MarMot.

	UDpipe+MarMot			Our		
	F_1		acc	F_1		acc
Noise	Tokens	POS	POS	Tokens	POS	POS
CLEAN	98.48	92.75	93.48	**98.69**	**93.48**	**94.27**
LOW	70.90	65.56	83.73	**96.08**	**90.51**	**92.80**
MID	20.62	19.07	58.53	**95.28**	**89.80**	**92.54**
HIGH	20.47	18.05	56.96	**95.45**	**89.82**	**92.14**

Table 4: Tokenization F_1, joint token-POS F_1 and (relaxed) POS tag accuracies on noisy version of UD v1.2.

We provide more details on the relaxed evaluation (description, examples and implementation) in our code repository. Note that we apply the relaxed evaluation only to UDpipe+MarMot but not to our model. The output of our model is directly evaluated against the gold labels of the clean corpus.

Results. The performance of our model decreases only slightly when increasing the noise level while the performance of UDpipe+MarMot drops significantly (Table 4). This confirms that our model is robust against noise from tokenization. Note that most other character-based models would suffer from the same performance drop as MarMot since they rely on tokenized inputs.

Discussion. The results in Table 4 show that our model can reliably recover token boundaries, even in noisy scenarios. This also explains its strong performance across languages: It can handle different languages, independent of whether the language merges tokens without whitespaces (e.g., Chinese) or separates tokens with whitespaces into syllables (e.g., Vietnamese).

4 Related Work

Character-based POS Tagging. Most work uses characters only to build more robust token representations but still relies on external tokenizers (Santos and Zadrozny, 2014; Lample et al., 2016; Plank et al., 2016; Dozat et al., 2017; Liu et al., 2017). In contrast, our model jointly learns segmentation and POS tagging. Gillick et al. (2016) do not rely on tokenization either but in contrast to their greedy decoder, our model optimizes the whole output sequence and is able to revise local decisions (Lafferty et al., 2001). For processing characters, LSTMs (Lample et al., 2016; Plank et al., 2016; Dozat et al., 2017) or CNNs (Ma and Hovy, 2016; Yu et al., 2017) are used. Our model combines biLSTMs and grConv to model both the context of characters (LSTM) and the compositionality of language (grConv).

Joint Segmentation and POS Tagging. The top performing models of EN, JA, VI and ZH use a pipeline of tokenizer and word-based POS tagger but do not treat both tasks jointly (Björkelund et al., 2017; Dozat et al., 2017; Kanayama et al., 2017; Qian and Liu, 2017). Especially for Chinese, there is a lot of work on joint word segmentation and POS tagging, e.g., (Zhang and Clark, 2008; Sun, 2011; Hatori et al., 2012; Zheng et al., 2013; Kong et al., 2015; Cai and Zhao, 2016; Chen et al., 2017; Shao et al., 2017), of which some use CRFs to predict one POS tag per character. However, this is hard to transfer to languages like English and Vietnamese where single characters are less informative and tokens are much longer, resulting in a larger combinatory label space. Thus, we choose a semi-Markov formalization to directly model segments.

Semi-Markov CRFs for Sequence Tagging. Zhuo et al. (2016) and Ye and Ling (2018) apply semi-CRFs to word-level inputs for named entity recognition. In contrast, we model character-based POS tagging. Thus, the expected length of our character segments is considerably larger than the expected length of word-based segments for NER. Kong et al. (2015) build SRNNs that we use as a baseline. In contrast to their 0-order model, we train a 1-order semi-CRF to model dependencies between segment labels.

5 Conclusion

We presented an end-to-end model for character-based part-of-speech tagging that uses semi-Markov conditional random fields to jointly segment and label a sequence of characters. Input representations and segment representations are trained parameters learned in end-to-end training by the neural network part of the model. The model achieves state-of-the-art results on two benchmark datasets across several typologically diverse languages. By corrupting the tokenization of the dataset, we show the robustness of our model, explaining its good performance on languages with difficult tokenization.

Acknowledgments

This work was funded by the European Research Council (ERC #740516). We would like to thank the anonymous reviewers for their helpful comments.

References

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3960–3966, Melbourne, Australia. AAAI Press.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California. Association for Computational Linguistics.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hiroshi Kanayama, Masayasu Muraoka, and Katsumasa Yoshikawa. 2017. A semi-universal pipelined approach to the conll 2017 ud shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 265–273, Vancouver, Canada. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI Conference on Artificial Intelligence*, pages 2741–2749. AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. *arXiv preprint arXiv:1511.06018*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 526–536, Denver, Colorado. Association for Computational Linguistics.

Joakim Nivre and Lars Ahrenberg Željko Agic. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. lindat/clarin digital library at the institute of formal and applied linguistics, charles university.

Joakim Nivre et al. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (
'UFAL), Faculty of Mathematics and Physics, Charles University.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *The future of gradient-based machine learning software and techniques, NIPS 2017*.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. *arXiv preprint arXiv:1707.06961*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Xian Qian and Yang Liu. 2017. A non-DNN feature engineering approach to dependency parsing – FBAML at CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 143–151, Vancouver, Canada. Association for Computational Linguistics.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3281–3287. AAAI Press.

Cícero N. dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Conference on Machine Learning*, pages 1818–1826.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1385–1394, Portland, Oregon, USA. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345, Jeju Island, Korea. Association for Computational Linguistics.

Zhi-Xiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov crf for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Xiang Yu, Agnieszka Falenska, and Ngoc Thang Vu. 2017. A general-purpose tagger with convolutional neural networks. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 124–129, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang, and Zaiqing Nie. 2016. Segment-level sequence modeling using gated recursive semi-markov conditional random fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1413–1423, Berlin, Germany. Association for Computational Linguistics.

Shrinking Japanese Morphological Analyzers With Neural Networks and Semi-supervised Learning

Arseny Tolmachev **Daisuke Kawahara** **Sadao Kurohashi**

Kyoto University, Graduate School of Informatics
Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
`arseny@kotonoha.ws` `dk@i.kyoto-u.ac.jp` `kuro@i.kyoto-u.ac.jp`

Abstract

For languages without natural word boundaries, like Japanese and Chinese, word segmentation is a prerequisite for downstream analysis. For Japanese, segmentation is often done jointly with part of speech tagging, and this process is usually referred to as morphological analysis. Morphological analyzers are trained on data hand-annotated with segmentation boundaries and part of speech tags. A segmentation dictionary or character n-gram information is also provided as additional inputs to the model. Incorporating this extra information makes models large. Modern neural morphological analyzers can consume gigabytes of memory. We propose a compact alternative to these cumbersome approaches which do not rely on any externally provided n-gram or word representations. The model uses only unigram character embeddings, encodes them using either stacked bi-LSTM or a self-attention network, and independently infers both segmentation and part of speech information. The model is trained in an end-to-end and semi-supervised fashion, on labels produced by a state-of-the-art analyzer. We demonstrate that the proposed technique rivals performance of a previous dictionary-based state-of-the-art approach and can even surpass it when training with the combination of human-annotated and automatically-annotated data. Our model itself is significantly smaller than the dictionary-based one: it uses less than 15 megabytes of space.

1 Introduction

Languages with a continuous script, like Japanese and Chinese, do not have natural word boundaries in most cases. Natural language processing for such languages requires to perform some variation of word segmentation.

Although some NLP applications, like neural machine translation, started to use unsupervised segmentation methods (Kudo and Richardson, 2018), resulting segmentation often has decisions which are not natural to humans. Supervised segmentation based on a human-defined standard is essential for applications which are designed for interaction on a word-level granularity, for example, full-text search. Segmentation is commonly done jointly with part of speech (**POS**) tagging and usually referred to as Morphological Analysis.

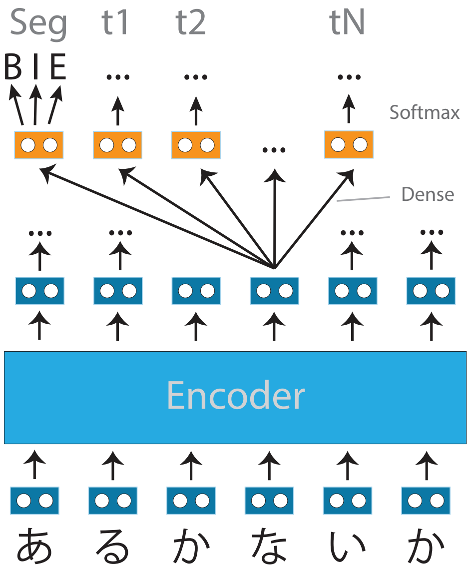


Figure 1: Proposed model. We encode character unigram embeddings into shared representations for each character. The shared representation is projected into a tag-specific representations from which we independently infer segmentation and per-character tags.

Modern Japanese Morphological Analyzers (**MA**) are very accurate, having a >99 segmentation tokenwise F1 score on news domain and a >98.5 F1 on web domain (Tolmachev et al., 2018). They often use segmentation dictionaries which define possible words. Also, their models are generally large and unwieldy, spanning hundreds of megabytes in case of traditional symbolic feature-based approaches. Neural models with word or n-

Proceedings of NAACL-HLT 2019, pages 2744–2755
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

gram embeddings are even larger, easily reaching gigabytes. This makes it difficult to deploy MA in space-constrained environments such as mobile applications and browsers.

It has been shown that simple or straightforward models can match or outperform complex models when using a large number of training data. For example, a straightforward backoff technique rivals a complicated smoothing technique for language models (Brants et al., 2007). Pretraining a bidirectional language model on a large dataset helps to solve a variety of NLP tasks (Devlin et al., 2018). Our approach is inspired by this line of work.

Contributions We propose a very straightforward fully-neural morphological analyzer which uses *only character unigrams* as its input[1]. Such an analyzer, when trained only on human-annotated *gold* data has low accuracy. However, when trained on a large amount of automatically tagged *silver* data, the analyzer rivals and even outperforms, albeit slightly, the bootstrapping analyzer. We conclude that there is no need for rich input representation. Neural networks learn the information to combine characters into words by themselves when given enough data.

Ignoring explicit dictionary information and rich input representations makes it possible to make analyzers that are highly accurate and very compact at the same time. We also perform ablation experiments which show that the encoder component of such an analyzer is more important than character embeddings.

2 Morphological Analysis Overview

Segmentation is a cornerstone requirement for processing languages with a continuous script, and thus it has been studied for a long time. Most current approaches use either rich feature representation, e.g. character n-grams or their embeddings, or a segmentation dictionary. There exist two main lines of approaches: pointwise and search-based. Pointwise approaches make a segmentation decision for each character, usually based on the information from its surroundings. Search-based approaches look for a maximum scored interpretation in some structure over the input sentence.

Most Japanese analyzers use segmentation dictionaries which define corpus segmentation standards. They usually have rich POS information attached and are human-curated. One focus of segmentation dictionaries is to be consistent: it should be possible to segment a sentence using the dictionary entries only in a single correct way. Such dictionaries are often maintained together with annotated corpora. On the other hand, Chinese-focused systems do not put much focus on dictionaries. Still, almost all aproaches use rich feature templates or additional resources such as pretrained character n-gram or word embeddings, which increase the model size.

Pointwise approaches make a segmentation decision independently for each position. They can be seen as a sequence tagging task. Such approaches are more popular for Chinese.

KyTea (Neubig et al., 2011) is an example of this approach in Japanese. It makes a binary decision for each character: whether to insert a boundary before it or not. It can be seen as sequence tagging with {B, I} tagset. POS tagging is done after inferring segmentation. The decisions are made by feature-based approaches, using characters, character n-grams, character type information, and dictionary information as features. KyTea can use word features obtained from a dictionary. It checks whether the character sequence before and after the current character forms a word from the dictionary. It also checks whether the current word is inside a word.

Neural networks were shown to be useful for Japanese in this paradigm as well (Kitagawa and Komachi, 2017). They use character embeddings, character type embeddings, character n-gram embeddings, and tricks to incorporate dictionary information into the model.

Many studies on Chinese adopt the pointwise approach. Often, the segmentation task is reformulated as sequence tagging (Xue, 2003) with {B, I, E, S} tagset. Peng et al. (2004) showed that CRFs help further in this task. This tactic was followed by many subsequent feature-based approaches (Tseng et al., 2005; Zhao et al., 2006; Zhang et al., 2013), using character n-gram, character type and word features.

Neural networks were applied to this paradigm as well. Zheng et al. (2013) used a feed-forward network on character and categorical features that were shown to be useful for computing a segmentation score from a fixed window. Qi et al. (2014) used a similar architecture. They predicted not only segmentation but POS tags and performed

---

[1] The source code is avaliable at `https://github.com/eiennohito/rakkyo`

named entity recognition as well. The character representation was pretrained on a language modeling task. Shao et al. (2017) used a bidirectional recurrent network with GRU cells followed by a CRF layer for joint segmentation and POS tagging. They used pretrained character n-gram embeddings together with sub-character level information extracted by CNNs as features. Using a dictionary with NN is also popular (Zhang et al., 2018b; Liu et al., 2018).

Search-based approaches induce a structure over a sentence and perform a search over it. A most frequently used structure is a lattice which contains all possible segmentation tokens. The search then finds the highest scoring path through the lattice. Another branch of search-based approaches splits decisions into transitions (starting a new token and appending a character to the token) and searches for the highest scoring chain of transitions. This also can be seen as dynamically constructing a lattice while performing the search in it at the same time.

Lattice-based approaches are popular for the Japanese language. Most of the time, the lattice is based on words which are present in a segmentation dictionary and a rule-based component for handling out-of-dictionary words. Usually, there are no machine-learning components in lattice creation, but the scoring can be machine-learning based. We believe that the availability of high quality consistent morphological analysis dictionaries is the reason for that. Still, the work of Kaji and Kitsuregawa (2013) is a counterexample of a lattice-based approach for Japanese which uses a machine-learning component for creating the lattice.

Traditional lattice-based approaches for Japanese use mostly POS tags or other hidden information accessible from the dictionary to score paths through the lattice. JUMAN (Kurohashi, 1994) is one of the first analyzers, which uses a hidden Markov model with manually-tuned weights for scoring. Lattice path scores are computed using connection weights for each pair of part of speech tags.

Probably the most known and used morphological analyzer for Japanese is MeCab (Kudo et al., 2004), where CRFs were used for learning the scoring. MeCab is very fast: it can analyze almost 50k sentences per second. It also achieves acceptable accuracy, and so the tool is very popular. The speed is realized by precomputing feature weights, but it takes a lot of space when the total number of features gets large. For example, the UniDic model for modern Japanese v2.3.0[2] takes 5.5GB because it uses many feature templates.

There were studies which tried to integrate NN into lattice-based approaches as well. Juman++ (Morita et al., 2015) uses dictionary-based lattice construction with the combination of two models for path scoring: the feature-based linear model using soft-confidence weighted learning (Wang et al., 2016) and a recurrent neural network (Mikolov, 2012). It significantly reduced the number of both segmentation and POS tagging errors. However, it was very slow, being able to analyze only about 15 sentences per second, hence the original version was impractical. The following improvement (Tolmachev et al., 2018) greatly increased analysis speed by doing aggressive beam trimming and performing heavyweight NN evaluation only after lightweight scoring by the linear model.

Direct lattice-based approaches are not very popular for Chinese, but some are lattice-based in spirit. A line of work by Zhang and Clark (2008, 2010) builds the lattice dynamically from partial words, searching paths with a perceptron-based scorer and customized beam search. The dictionary is built dynamically from the training data as frequent word-tag pairs which help the system to prune unlikely POS tags for word candidates.

One more variation on lattice-based approaches for Chinese is the work by Cai and Zhao (2016). In this work, a segmentation dictionary is used to construct a subnetwork, which combines character representations into word representations used for computing sentence-wise segmentation scores. This can be seen as explicitly learning dictionary information by a model. Resulting segmentation is still created from the start to the end by growing words one by one while performing beam search. The follow up (Cai et al., 2017) simplifies that model and shows that greedy search can be enough for estimating segmentation when using neural networks. Still, this line of work does not consider POS tagging.

Transition-based approaches treat input data (most frequently – characters) as input queue and store a current, possibly incomplete, token in a buffer. Models usually infer whether they should create a new token from a character in the input

---

[2] https://unidic.ninjal.ac.jp/

queue or append an input character to the already existing token. Neural models are often used in this paradigm (Ma and Hinrichs, 2015; Zhang et al., 2016; Yang et al., 2017; Ma et al., 2018; Zhang et al., 2018a). Almost all of them use both word and charcter n-gram embeddings. This paradigm was extended to do parsing jointly with MA (Hatori et al., 2012; Kurita et al., 2017).

Semi-supervised approaches to segmentation and POS tagging fall into several categories. The first one uses raw or automatically-annotated data to precompute feature representations and then uses these feature representations for supervised learning. For example, Sun and Xu (2011) and Wang et al. (2011) use data from automatically segmented texts as features. They precomute the features beforehand and train an analyzer afterwards. In addition to that, Zhang et al. (2013) use a variation of smoothing for handling automatic annotation errors. A lot of neural-based methods pretrain word and character n-gram embeddings. Yang et al. (2017) pretrain a part of the model on different data sources, including automatically segmented text, but the model itself is trained only on the gold data.

Another approach is to use heterogeneous data (annotated in incompatible annotation standards). In addition to corpus statistics from a raw corpus, Zhao and Kit (2008) exploit heterogeneous annotations. Li et al. (2015) use corpora with different annotation standards. They combine tags into “bundles” (e.g. [NN, n]) and infer them at the same time while paying attention to ambiguity. Chen et al. (2016) train a classifier that can annotate several standards jointly.

Finally, it is possible to use raw or automatically-annotated data directly. A study (Suzuki and Isozaki, 2008) is an example of a feature-based algorithm which uses raw data. Tri-training (Zhou and Li, 2005) is a generic way to use raw data. They propose to train on automatically analyzed examples where two of three diverse analyzers agree. Søgaard (2010) show that tri-training helps English POS-tagging with SVM and MaxEnt-based approaches. Zhou et al. (2017) use self-training and tri-training for Chinese word segmentation. They, however, also pretrain other features like word-context character embeddings, chracter unigrams and bigrams.

3 Proposed Approach

In order for MA to be practical, it should be not only accurate, but also fast and have relatively compact models. The speed of search-based approaches is dependent on how computationally heavy a weighting function is. Heavyweight models, like neural networks, require a large number of computations, and we think that it will be very difficult to create a practical search-based fully NN morphological analyzer with analysis speed comparable to traditional analyzers.

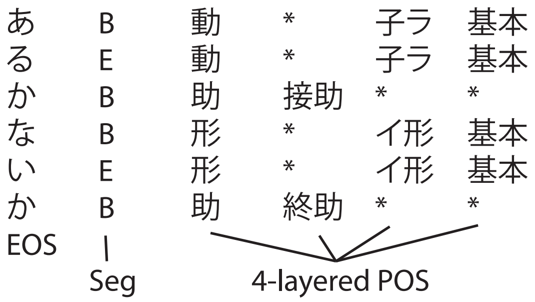
We do not want to use any explicit information about how to combine characters to form a word, like dictionaries, which takes space and is not trivial to incorporate into a character-based model. We also want our model to be fast, at least comparable with the speed of traditional analyzers. To this end, we follow a pointwise approach and force the neural network to learn the dictionary information from a corpus.

We use a straightforward architecture shown in Figure 1. We embed each character, and then apply an encoder, which produces an encoded representation for each character. Encoded character representations are independently transformed into tag representations. For each tag, the encoded representation is projected with a fully-connected layer with SeLU non-linearity (Klambauer et al., 2017). Finally, we multiply the tag representation by tag-specific embeddings and apply softmax non-linearity to get normalized tag probabilities.

Encoder Architectures We use two architectures for the encoder: a stacked bidirectional recurrent architecture with LSTM cells (Hochreiter and Schmidhuber (1997), **bi-LSTM**) and a Transofrmer-inspired mutihead self-attention network (Vaswani et al. (2017), **SAN**). We concatenate both directions of bi-LSTM outputs before passing them to the next layer without residual connections. We also apply layer normalization (Ba et al., 2016) to the concatenated outputs. We do not use dropout in encoders when using silver data for training.

Data Encoding Our model infers a tag for every input character. While this decision is natural for segmentation, POS tags are not usually tagged in this way.

For segmentation, we adopt {B, I, E} scheme. For POS tagging we broadcast tags to every character which is contained in a token. We use cor-



あ	B	動	*	子ラ	基本
る	E	動	*	子ラ	基本
か	B	助	接助	*	*
な	B	形	*	イ形	基本
い	E	形	*	イ形	基本
か	B	助	終助	*	*
EOS					

Figure 2: An example of full sentence annotation

あ	B	動	*	?	?
る	?	動	*	?	?
か	?	?	?	?	?
な	B	?	?	イ形	基本
い	E	?	?	イ形	基本
か	B	助	終助	*	*
EOS					

Figure 3: An example of partial sentence annotation

pora with the JUMAN-based segmentation standard (**Jumandic**), which has 4-layered POS tags: rough POS, fine POS, conjugation type and conjugation form. We treat each tag layer independently in our model, as shown in Figure 2.

We also consider a **partial annotation scheme**, where some tags are unknown. An example of partial sentence annotation is shown in Figure 3. Unknown tags are displayed by "?" symbols. We create partially annotated silver data by marking as unknown all tags which are ambiguous in a top-k analysis result. When computing the training loss, we treat unknown tags as padding: corresponding values are masked out of loss computation.

Loss Following Vaswani et al. (2017), we smooth softmax labels. They use the technique described by Szegedy et al. (2016), which uniformly distributes some small factor ϵ like 0.1 to incorrect labels. However, we do not induce a uniform smoothing. Instead, we want to prevent the model from being overconfident in its decisions without inducing uniformity. We slightly modify the cross-entropy loss as follows.

Remember that softmax probabilities are computed from unnormalized log-probabilities l_i as $q_i = e^{l_i}/Z$, where $Z = \sum_j e^{l_j}$. The cross-entropy loss will be $L = -\sum_i p_i \log q_i$, where p_i are gold probabilities. In our case the vector p is one-hot, meaning that $p_c = 1$ and other values are zero. This gives a sparse cross-entropy $L = -\log q_c = \log Z - l_c$, which is often implemented in deep learning frameworks. It has a minimum when $\log Z$ is equal to l_c, but it makes the model overconfident. Instead, we want to stop when $q_c = 1 - \epsilon$, or in other words $e^{l_c}/Z = 1 - \epsilon$. This gives us our modified loss:

$$L = \max(\log Z - l_c + \log(1 - \epsilon), 0).$$

It can be efficiently implemented using the sparse cross-entropy operation. In our experiments we use $\epsilon = 0.2$.

	Train		Test	
Corpus	Sents	Tokens	Sents	Tokens
KU	37k	930k	1783	46k
Leads	14k	217k	2195	36k

Table 1: Benchmark corpora sizes

Our final loss is a weighted sum of individual tag softmax losses. We use a weight coefficient of 10 for segmentation and 2 for the first POS tag layer.

4 Experiments

We conduct experiments on Japanese morphological analysis. For training we use two data sources. The first is usual human-annotated *gold* training data. The second is *silver* data from the results of automatic analysis. We use Juman++ V2 – the current state-of-the-art analyzer for the JUMAN segmentation standard as the bootstrap analyzer.

We use two gold corpora. The first is the Kyoto University Text Corpus (Kurohashi and Nagao (2003), referred to as **KU**), containing newspaper data. The second is the Kyoto University Web Document Leads Corpus (Hangyo et al. (2012), referred to as **Leads**) which consists of web documents. Corpus statistics are shown in Table 1. We denote models which use gold training data by **G**.

We take raw data to generate our silver annotated data from a crawled web corpus of 9.8B *unique* sentences. We sample 3B sentences randomly from it and analyze them using the Juman++ baseline model. From it we sample 500M sentences, which become our training silver data, prioritizing sentences which contain at least one not very frequent word. We prepare both top-scored (denoted as **T**) and non-ambigous in beam (denoted as **B**) variants of the silver data. Our silver data is in-domain for Leads and out-of-domain for KU.

Baselines We use four baselines: **JUMAN**, **MeCab**, **KyTea** and **Juman++** (V2). For MeCab,

Parameter	bi-LSTM	SAN
Char embedding size	128	128
Tag embedding size	32	32
# Layers	4	6
Hidden Size	128×2	32
# Heads	-	4
Projection Inner Dim	-	512
# Emedding Parameters	2.38M	2.38M
# Total Parameters	3.88M	3.59M

Table 2: Hyperparameters for neural models

KyTea and Juman++ we train a model using the same dictionary and merged training sections of KU and Leads, which is evaluated on each corpus independently.

Neural Models The hyper-parameters of the bi-LSTM-based model are displayed in Table 2. We use all unique characters present in our huge web corpus (18,581) as input. We select sizes of both neural models restricting the total number of parameters to be less than 4M. For optimization we use the Adam optimizer (Kingma and Ba, 2016) with hyperparameters and learning rate scheduling described by Vaswani et al. (2017). We train all models on Nvidia GPUs. On a single GeForce 1080Ti the bi-LSTM model can consume about 4,500 sentences per second and the SAN-based model about 6,500 sentences per second for training. We denote bi-LSTM-based models by **L** and SAN-based models by **S** in experimental results.

Treatment of Gold Data Existing methods are already highly accurate on this task, and it is difficult to perform hyperparameter and architecture selection reliably with a small development set. Because of that, we split our data in an unusual way. Generally, we use the silver data (B or T) as a train set, the human-annotated original training data (G) as a dev set and the original test set as a test set. Our hyperparameter selection decisions were based entirely on this setting. We do not perform additional hyperparameter search for a combination of silver and gold data for training.

The exception is cases when we use *only gold data* for training. For that, we cheat and optimize our hyperparameters, including dropout, which we use only for this setting, on test scores. Nontheless, the best scores on this setting are significantly lower than the worst baseline.

	KU, News			Leads, Web		
Analyzer	Seg	+P1	+P2	Seg	+P1	+P2
Baselines						
JUMAN	98.41	97.18	95.45	98.09	96.96	95.71
MeCab	99.10	98.56	97.59	98.25	97.60	96.22
KyTea	99.13	98.25	97.01	97.98	96.85	95.11
Juman++	**99.52**	**99.10**	97.86	98.61	98.07	96.70
bi-LSTM						
L:G	97.46	96.56	94.78	96.33	95.43	93.46
L:B	99.22	98.82	97.50	98.57	98.01	96.61
L:T	99.33	98.90	97.59	98.68	98.16	96.71
L:BG	99.43	99.05	**98.06**	98.59	98.04	96.76
L:TG	99.43	99.05	98.01	**98.71**	**98.19**	96.80
Self-attention						
S:G	98.28	97.67	95.66	97.23	96.36	93.91
S:B	99.19	98.75	97.34	98.56	97.99	96.59
S:T	99.23	98.78	97.36	98.66	98.15	96.75
S:BG	99.30	98.90	97.83	98.60	98.03	96.70
S:TG	99.37	98.97	97.93	**98.70**	98.15	**96.83**
Pre-training scenario						
S:B→G(a)	99.24	98.85	97.75	98.58	98.03	96.64
S:B→G(b)	99.15	98.65	97.55	98.39	97.78	96.36
S:B→G(c)	99.27	98.82	97.75	98.50	97.91	96.43
S:B→G(d)	99.26	98.82	97.76	98.52	97.94	96.48

Table 3: Test F1 score comparison on benchmark corpora. Legend: bi-[L]STM, [S]AN, [G]old data, [T]op-only and [B]eam-non-ambigous silver data.

Experimental Results Results of our experiments are shown in Table 3. For each analyzer, we show six values. **Seg** is a tokenwise F1 measure on segmentation. **+P1** requires the 1st layer of POS tags (coarse-grained POS tags) also to match gold data. For the sake of simplicity, we use only POS tags co-located with "B" Seg tags for the evaluation. **+P2** is analogous for the 2nd layer of POS tags. For all results in this table, we train NN-based models for a single epoch, which means *the training procedure sees each silver sentence only once*. We use one gold example for ten silver examples for mixed-data settings, looping over the gold data until the silver data is extinguished.

Training neural models only on gold data quickly results in overfitting which can be seen in L:G and S:G results. These scores are significantly lower than that of our worst baseline: JUMAN.

Models trained on only non-ambiguous silver data (*:B) are comparable to the best baseline on Leads (in-domain), although they cannot reach the accuracy of Juman++ on KU. Using top-only silver data (*:T) further improves accuracy. Both of our models in this setting slightly outperform previous Leads SOTA and have more or less the same accuracy. On KU, the LSTM-based model seems to be slightly better than the SAN-based one. In the context of semi-supervised learning, tri-training emphasizes using data when there exists a disagreement between the analyzers. Instead, we throw

	Size, MB		
Analyzer	Dictionary	Model	Total
JUMAN	288	1	289
MeCab	312	8	320
KyTea:G	-	569	569
KyTea:TG	-	3218	3218
Juman++	157	288	434
bi-LSTM	1	14	15
SAN	1	13	14

Table 4: MA model sizes for Jumandic

away difficult cases for beam-based data, denoising it in a sense, but NN seem to handle that kind of noise relatively well.

Adding the gold data to the silver data (*:BG, *:TG) allows both models to improve their accuracy further. Results on Leads are comparable for both L:TG and S:TG and higher than the previous SOTA, giving segmentation error reduction of 8% in comparison to Juman++. On KU, the LSTM-based models seem to perform better without a significant difference on the TG and BG settings, while still underperforming the Juman++ baseline except +P2 case, where both models are stronger than Juman++.

Pre-training Scenario We also check the fine-tuning approach when we first learn the representations on a large corpus and then refine the model on a gold corpus. S:B→G(a-d) are four such runs of a SAN-based model with different hyperparameters. All four runs are initialized with the same S:B model and trained on the gold data only. We found it difficult to find good hyperparameters for fine-tuning. The models were prone to overfit very fast. Mixing gold and silver data resulted in stable training without hyperparameter search.

Model Sizes We compare the model sizes of analyzers in Table 4. In case of dictionary-based analyzers the dictionary takes most of the space. We count sizes of compiled models for all analyzers. KyTea, as another example of pointwise MA, uses string-based features and treats its features uniformly, hence dictionary size is not applicable to it. A **KyTea:TG** variant that uses additional 2M silver sentences takes almost 6x the space of the original model, reaching 3GB. When using neural networks, on the other hand, it is possible to control model sizes more easily. Moreover, our proposed models take significantly less space while having comparable accuracy.

Analyzer	KU	Leads
KyTea-D:G	98.45	97.04
KyTea-D:T	98.51	98.10
KyTea-D:TG	99.18	98.31
KyTea:G	99.13	97.98
KyTea:TG	99.33	98.42

Table 5: KyTea test Seg F1 comparison. -D models do not use the dictionary. T models use silver data (2M sentences, created like in the main experiment)

Dictionary-based analyzers store other information, like readings and lemma forms, in addition to token surface forms and POS, but removing that information would not make model sizes comparable with NN-based ones. For NN-based analyzers, we count a dictionary as 1 MB because they need a character-to-id mapping to work. However, the list of characters contains non-frequently used characters, some of which could be treated as UNKs without any accuracy loss. We also treat weights as 4-byte floating points, and so it would be possible to further decrease the NN model size, for example by using less precise storage formats.

5 Discussion

Dictionaries Dictionary information is usually added to character-based models either using a binary feature vector (e.g. a dictionary contains a trigram to the left of the decision point) or word embeddings. We believe that a dictionary can be replaced with a large training corpus which includes most of the entries from that dictionary. A neural model with only the unigram character input can solve word segmentation and POS tagging only if it builds some knowledge about the dictionary internally. Our main experimental results (Table 4) show that it seems to be the case and there is no need to model the dictionary explicitly.

Table 5 shows an effect of using dictionaries and silver data on KyTea, an instance of symbolic feature-based analyzer. Models tagged with **T** use additional 2M silver training data analyzed by Juman++. KyTea has better accuracy in settings when it uses the dictionary. The dictionary even helps in the setting with additional silver data. Unfortunately, the model size increases as well, limiting the amount of silver data we can use, and the accuracy cannot rival neural approaches.

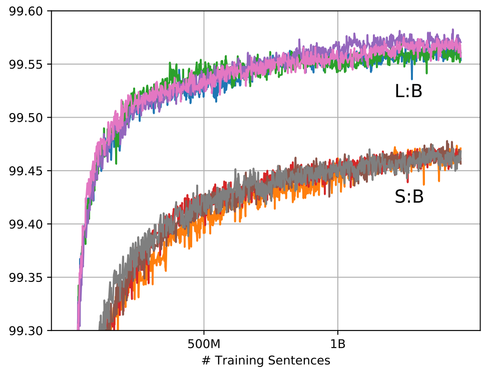


Figure 4: Dev Seg F1 curves for L:B and S:B

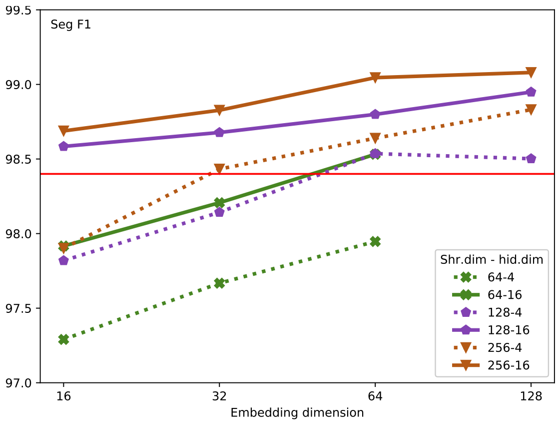


Figure 5: Effect of embedding size on Seg F1

How much data do we need? For our main experiments, we train all models for a single epoch on our silver dataset. Figure 4 shows KU train (our dev set) Seg F1 curves for L:B and S:B for three epochs. We ran each experiment four times with different random seeds. The learning curves become less sloppy when reaching 500M sentences but do not become flat there. The training does not seem to completely converge even after 3 epochs. We still use one full epoch (500M) for our main experiments. The curves are pretty noisy, but it seems that the model is robust with respect to initialization.

SAN Ablation Experiments The proposed MA achieves high accuracy while having very compact models. The inputs do not contain any information on how to combine characters into words and we assume that the model learns it from the data. To get the model size even smaller, we check which model parts contribute more to the resulting analysis accuracy, meaning that they contain the dictionary knowledge.

We perform ablation experiments on the SAN model by varying its hyperparameters and checking how it affects the accuracy of the resulting analyzer. The LSTM model could not converge in this setting. We used 2.5M of silver training data for these experiments.

Figure 5 shows the segmentation F1 score when varying input embedding, shared representation and SAN hidden dimension sizes. JUMAN score, as a lowest acceptable baseline, is shown in red. The embedding size seems to have a lower impact on accuracy than the shared representation and the SAN hidden dimension size. Namely, the (128-16) model with the embedding size of 16 has higher accuracy than the (128-4) model with the embedding size of 128. Accordingly, we believe that the encoder contributes much stronger to learning the dictionary than character embeddings.

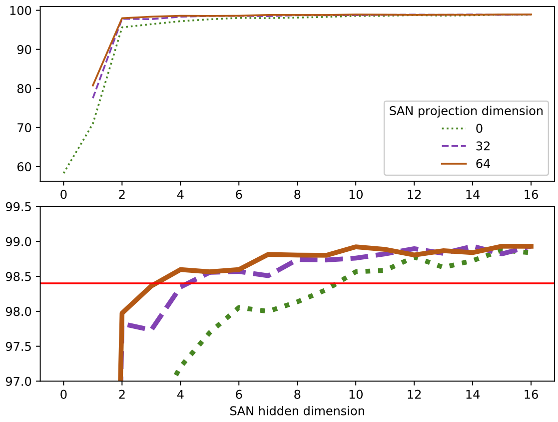


Figure 6: Effect of SAN hidden dimension on Seg F1

One more interesting observation is that the models are still better than JUMAN, while having much less parameters than our base model. We explore more extreme settings of the SAN hidden state, shown in Figure 6. We fix embedding and shared representation dimensions to 128 and vary the SAN hidden and projection dimensions. The lower subgraph is a scale-up version of top graph. The point at SAN hidden size equal to 0 means that we directly use unigram embeddings to predict segmentation without any encoder.

The SAN projection size is consistent with accuracy, especially on smaller SAN hidden sizes. An interesting observation here is that the SAN model seems to work even with hidden dimension of 2. When the hidden dimension size reaches 4, the extremely small model accuracy is higher than the JUMAN baseline. This shows that it is possible to create an extremely small MA with acceptable accuracy.

<table>
<tr><th>System Segmentation</th><th>Correct Segmentation</th><th>Transliteration</th><th>Meaning</th></tr>
<tr><td>こう|ゆう|曲って</td><td>こうゆう|曲|って</td><td>ko:yu: kyoku tte</td><td>this song is</td></tr>
<tr><td>なんて|すん|ごい</td><td>なんて|すんごい</td><td>nante sungoi</td><td>how awesome</td></tr>
<tr><td>んな|わけな|いって</td><td>んな|わけ|ない|って</td><td>nna wake nai tte</td><td>no way!</td></tr>
<tr><td>あっ|ちゃんと|遊び|たい</td><td>あっ|ちゃん|と|遊び|たい</td><td colspan="2"><i>see main text</i></td></tr>
</table>

Table 6: n-grams with inconsistent POS tags which are also Juman++ errors

Label Uncertainty and Error Analysis Because our neural models infer all tags independently, they can be inconsistent, for example, a word can have different POS tags on different characters. We looked into frequent 3-grams where the central word has inconsistent tags (POS tags are not the same for all characters, or they do not form a correct 4-layered tag). Most of these trigrams occur in ambiguous situations.

We have picked several examples which are actually errors in Juman++ segmentation as well. They are shown in Table 6. In Japanese, words often have several orthographic forms. The most common variant is usage of hiragana (phonetic script) instead of kanji (ideographic characters). Verbs can have different possible endings, e.g. 曲がる and 曲る (magaru – to turn or bend) are two orthographic variants of a single verb. There are also colloquial variants; namely the verb 言う is usually read as いう (iu – to say), but can also be written as ゆう because the pronunciation is close. These phenomena are relatively common in web and user-generated texts, but corpus and segmentation dictionary coverage of them is not very good.

The first two examples contain alternative colloquial spellings of words こういう (ko:iu – such) and すごい (sugoi – awesome). In the first example the system incorrectly recognizes 曲|って (kyoku tte) as 曲って (magatte) – a conjugation of 曲る. The fourth example (a chanto asobitai/ac-chan to asobitai - ah! [I] want to play properly/[I] want to play with ac-chan <person name>) is actually ambiguous and can have two meanings. The second one is more probable though. The fact that frequent words with uncertain POS tags are Juman++ errors as well implies that insufficient gold data causes the uncertainty.

We also compare differences between Juman++ and our models to get an insight on general problems with proposed methods. Neural models make many errors in hiragana words. For example, both neural models make errors in the sentence 弱者|が|とうた|さ|れて (jyakusya ga to:ta sarete - weaklings lose to natural selection). LSTM makes a segmentation mistake (と|うたさ) and SAN does a POS tagging mistake, while Juman++ produces the correct answer. It knows that とうた is a special type of noun that is often followed by されて from POS tags. Hiragana-based spellings of most content words are somewhat rare in Japanese, and NN models do not have enough training data for these spellings. It could be possible to improve the situation by using data augmentation techniques. Another frequent problem is segmentation and tagging of proper nouns. We believe that this problem could be solved by data augmentation, but we leave this as future work.

6 Conclusion and Future Work

We presented a novel way to train small neural models for Japanese Morphological analysis by directly feeding the network a large number of silver training data. Our method achieves new SOTA on web domain when combining the silver data with gold one. This is an empirical evidence that there is no need for feature engineering for neural morphological analysis at all. A neural network can learn implicit dictionary information itself and it does not need to be large. We also show that training by mixing the data together works better than fine-tuning and is more stable.

Our work can be extended in the future in different ways. We will consider how to make the model to recognize new words, which is an important feature for a practical analyzer. Using tri-training also seems to be a natural extension for this work. It is easy to provide diverse models, required for tri-training, by using different types of encoder and varying network parameters. Furthermore, our tagging approach should be universal and work with other tasks like named entity recognition. A method to incorporate tags with a large number of possible values (like readings and lemmas) without introducing embeddings for them, hence keeping the models small, could also be a useful extension.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450 [cs]*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and Accurate Neural Word Segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural Network for Heterogeneous Annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 731–741, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a Diverse Document Leads Corpus Annotated with Semantic Relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali, Indonesia. Faculty of Computer Science, Universitas Indonesia.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental Joint Approach to Word Segmentation, POS Tagging, and Dependency Parsing in Chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Nobuhiro Kaji and Masaru Kitsuregawa. 2013. Efficient Word Lattice Generation for Joint Word Segmentation and POS Tagging in Japanese. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 153–161, Nagoya, Japan. Asian Federation of Natural Language Processing.

Diederik P. Kingma and Jimmy Ba. 2016. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*.

Yoshiaki Kitagawa and Mamoru Komachi. 2017. Long Short-Term Memory for Japanese Word Segmentation. *arXiv:1709.08011 [cs]*. ArXiv: 1709.08011.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural Joint Model for Transition-based Chinese Syntactic Analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1214, Vancouver, Canada. Association for Computational Linguistics.

Sadao Kurohashi. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of The International Workshop on Sharable Natural Language, 1994*.

Sadao Kurohashi and Makoto Nagao. 2003. Building A Japanese Parsed Corpus. In *Treebanks: Building and Using Parsed Corpora*, Text, Speech and Language Technology, pages 249–260. Springer Netherlands, Dordrecht.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled Sequence Labeling on Heterogeneous Annotations: POS Tagging as a Case Study. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1783–1792, Beijing, China. Association for Computational Linguistics.

Junxin Liu, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2018. Neural Chinese Word Segmentation with Dictionary Knowledge. In *Natural Language Processing and Chinese Computing*, Lecture Notes in Computer Science, pages 80–91. Springer International Publishing.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate Linear-Time Chinese Word Segmentation via Embedding Matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1733–1743, Beijing, China. Association for Computational Linguistics.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph. D. Thesis, Brno University of Technology, Brno.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese Segmentation and New Word Detection Using Conditional Random Fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yanjun Qi, Sujatha G. Das, Ronan Collobert, and Jason Weston. 2014. Deep Learning for Character-Based Information Extraction. In *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 668–674. Springer International Publishing.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based Joint Segmentation and POS Tagging for Chinese using Bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese Word Segmentation Using Unlabeled Data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jun Suzuki and Hideki Isozaki. 2008. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Anders Søgaard. 2010. Simple Semi-Supervised Training of Part-Of-Speech Taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208, Uppsala, Sweden. Association for Computational Linguistics.

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A Morphological Analysis Toolkit for Scriptio Continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. 2016. Soft Confidence-Weighted Learning. *ACM Trans. Intell. Syst. Technol.*, 8(1):15:1–15:32.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese Word Segmentation and POS Tagging with Semi-supervised Methods Using Large Auto-Analyzed Data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Nianwen Xue. 2003. Chinese Word Segmentation as Character Tagging. In *International Journal of Computational Linguistics & Chinese Language Process-*

ing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing, pages 29–48.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural Word Segmentation with Rich Pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA. Association for Computational Linguistics.

Meishan Zhang, Nan Yu, and Guohong Fu. 2018a. A Simple and Effective Neural Model for Joint Word Segmentation and POS Tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1528–1538.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-Based Neural Word Segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018b. Neural Networks Incorporating Dictionaries for Chinese Word Segmentation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5682–5689.

Yue Zhang and Stephen Clark. 2008. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2010. A Fast Decoder for Joint Word Segmentation and POS-Tagging Using a Single Discriminative Model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective Tag Set Selection in Chinese Word Segmentation via Conditional Random Field Modeling. In *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Exploiting unlabeled text with different unsupervised segmentation criteria for chinese word segmentation. *Research in Computing Science*, 33:93–104.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep Learning for Chinese Word Segmentation and POS Tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-Context Character Embeddings for Chinese Word Segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Neural Constituency Parsing of Speech Transcripts

Paria Jamshid Lou
Macquarie University
Sydney, Australia

Yufei Wang
Macquarie University
Sydney, Australia

{paria.jamshid-lou,yufei.wang}@hdr.mq.edu.au

Mark Johnson
Macquarie University
Sydney, Australia
mark.johnson@mq.edu.au

Abstract

This paper studies the performance of a neural self-attentive parser on transcribed speech. Speech presents parsing challenges that do not appear in written text, such as the lack of punctuation and the presence of speech disfluencies (including filled pauses, repetitions, corrections, etc.). Disfluencies are especially problematic for conventional syntactic parsers, which typically fail to find any EDITED disfluency nodes at all. This motivated the development of special disfluency detection systems, and special mechanisms added to parsers specifically to handle disfluencies. However, we show here that neural parsers can find EDITED disfluency nodes, and the best neural parsers find them with an accuracy surpassing that of specialized disfluency detection systems, thus making these specialized mechanisms unnecessary. This paper also investigates a modified loss function that puts more weight on EDITED nodes. It also describes tree-transformations that simplify the disfluency detection task by providing alternative encodings of disfluencies and syntactic information.

1 Introduction

While a great deal of effort has been expended on parsing written text, parsing speech (either transcribed or ASR output) has received less attention. Parsing speech is important because speech is the easiest and most natural means of communication, it is increasingly used as an input modality in human-computer interactions. Speech presents parsing challenges that do not appear in written text, such as the lack of punctuation and sentence boundaries, speech recognition errors and the presence of speech disfluencies (including filled pauses, repetitions, corrections, etc.) (Kahn et al., 2005). Of the major challenges associated with transcribed speech, we focus here on speech disfluencies, which are frequent in spontaneous speech.

Disfluencies include filled pauses (“um”, “uh”), parenthetical asides (”you know”, ”I mean”), interjections (“well”, ”like”) and partial words (“wou-”, “oper-”). One type of disfluency which is especially problematic for conventional syntactic parsers are speech repairs. Following the analysis of Shriberg (1994), a speech repair consists of three main parts; the *reparandum*, the *interregnum* and the *repair*. As illustrated in the following example, the reparandum *we don’t* is the part of the utterance that is replaced or repaired, the interregnum *uh I mean* (which consists of a filled pause *uh* and a discourse marker *I mean*) is an optional part of the disfluency, and the repair *a lot of states don’t* replaces the reparandum. The fluent version is obtained by removing the reparandum and the interregnum.

$$\overbrace{\textit{We don't}}^{\text{reparandum}}\ \overbrace{\textit{uh I mean}}^{\text{interregnum}}\ \overbrace{\textit{a lot of states don't}}^{\text{repair}}\ \textit{have capital punishment.} \qquad (1)$$

In the Switchboard treebank corpus (Mitchell et al., 1999) the reparanda, filled pauses and discourse markers are dominated by EDITED, INTJ and PRN nodes, respectively (see Figure 1). Of these disfluency nodes, EDITED nodes pose a major problem for conventional syntactic parsers, as the parsers typically fail to find any EDITED nodes at all. Conventional parsers mainly capture tree-structured dependencies between words, while the relation between reparandum and repair is quite different: the repair is often a “rough copy” of the reparandum, using the same or very similar words in roughly the same order (Charniak and Johnson, 2001; Johnson and Charniak, 2004). The “rough copy” dependencies are strong evidence of a disfluency, but conventional syntac-

Proceedings of NAACL-HLT 2019, pages 2756–2765
Minneapolis, Minnesota, June 2 - June 7, 2019.

tic parsers cannot capture them. Moreover, the reparandum and the repair do not form conventional syntactic phrases, as illustrated in Figure 1, which is an additional difficulty when integrating disfluency detection with syntactic parsing. This motivated the development of special disfluency detection systems which find and remove disfluent words from the input prior to parsing (Charniak and Johnson, 2001; Kahn et al., 2005; Lease and Johnson, 2006), and special mechanisms added to parsers specifically to handle disfluencies (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Yoshikawa et al., 2016; Tran et al., 2018).

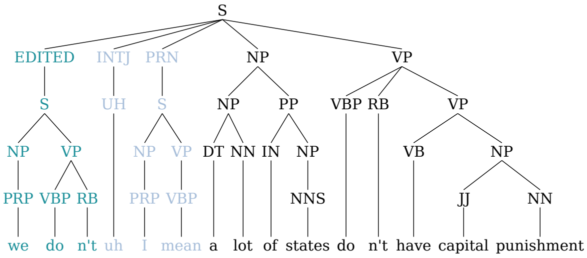


Figure 1: An example parse tree from the Switchboard corpus – *We don't uh I mean a lot of states don't have capital punishment,* where reparandum *We don't*, filled pause *uh* and discourse marker *I mean* are dominated by EDITED, INTJ and PRN nodes.

In this paper, we investigate the performance of a neural self-attentive constituency parser on speech transcripts. We show that an "off-the-shelf" self-attentive parser, unlike conventional parsers, can detect disfluent words with a performance which is competitive to or better than specialized disfluency detection systems. In summary, the main contributions of this paper are:

- We show that the self-attentive constituency parser sets a new state-of-the-art for syntactic parsing of transcribed speech,
- A neural constituency parser can detect EDITED words with an accuracy surpassing that of specialized disfluency detection models,
- We demonstrate that syntactic information helps the neural syntactic parsing detect disfluent words more accurately,
- Replacing the constituent-based representation of disfluencies with a word-based representation of disfluencies improves the detection of disfluent words,
- Modifying the training loss function to put more weight on EDITED nodes during training also improves disfluency detection.

2 Related Work

Speech recognition errors, unknown sentence boundaries and disfluencies are three major problems addressed by previous work on parsing speech. In this work, we focus on the problem of disfluency detection in parsing human-transcribed speech, where we assume that sentence boundaries are given and there are no word recognition errors. This section reviews approaches that add special mechanisms to parsers to handle disfluencies as well as specialized disfluency detection models.

2.1 Joint Parsing and Disfluency Detection

Many speech parsers adopt a transition-based dependency approach to (i) find the relationship between head words and words modifying the heads, and (ii) detect and remove disfluent words and their dependencies from the sentence. Transition-based parsers can be augmented with new parse actions to specifically handle disfluent words (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Yoshikawa et al., 2016; Wu et al., 2015). A classifier is trained to choose between the standard and the augmented parse actions at each time step. Using pattern-match features in the classifier significantly improves disfluency detection (Honnibal and Johnson, 2014). This reflects the fact that parsing based models use pattern-matching to capture the "rough copy" dependencies that are characteristic of speech disfluencies.

Speech parsing models usually use lexical features. One recent approach (Tran et al., 2018) integrates lexical and prosodic cues in an encoder-decoder constituency parser. Prosodic cues result in very small performance gain in both parsing and disfluency detection. Augmenting the parser with a location-aware attention mechanism is specially useful for detecting disfluencies (Tran et al., 2018).

In general, parsing models are poor at detecting disfluencies, mainly due to "rough copy" dependencies in disfluent sentences, which are difficult for conventional parsers to detect.

2.2 Specialized Disfluency Detection Models

Disfluency detection models often use a sequence tagging technique to assign a single label to each word of a sequence. Previous work shows that LSTMs and CNNs operating on words alone are poor at disfluency detection (Zayats et al., 2016; Wang et al., 2016; Jamshid Lou et al., 2018). The performance of state-of-the-art disfluency detection models depends heavily on hand-crafted pattern match features, which are specifically designed to find "rough copies". One recent paper (Jamshid Lou et al., 2018) augments a CNN model with a new kind of layer called an *auto-correlational layer* to capture "rough copy" dependencies. The model compares the input vectors of words within a window to find identical or similar words. The addition of the auto-correlational layer to a "vanilla" CNN significantly improves the performance over the baseline CNN model. The results are competitive to models using complex hand-crafted features or external information sources, indicating that the auto-correlation model learns "rough copies".

One recent paper (Wang et al., 2018) introduces a semi-supervised approach to disfluency detection. Their self-attentive model is the current state-of-the-art result in disfluency detection. The common factor in Wang et al. (2018) and the approach presented here is the self-attentive transformer architecture, which suggests that this architecture is capable of detecting disfluencies with very high accuracy. The work we present goes beyond the work of Wang et al. (2018) in also studying the impact of jointly predicting syntactic structure and disfluencies (so it can be understood as a kind of multi-task learning). We also investigate the impact of different ways of representing disfluency information in the context of a syntactic parsing task.

3 Neural Constituency Parser

We use the self-attentive constituency parser introduced by Kitaev and Klein (2018) and train it on the Switchboard corpus of transcribed speech (we describe the training and evaluation conditions in more detail in Section 4). The self-attentive parser achieves state-of-the-art performance on WSJ data, which is why we selected it as the best "off-the-shelf" parsing model. The constituency parser uses a self-attentive transformer (Vaswani et al., 2017) as an encoder and a chart-based parser (Stern et al., 2017) as a decoder, as reviewed in the following sections.

3.1 Self-Attentive Encoder

The encoder of a transformer is a stack of n identical layers, each consists of two stacked sublayers: a multi-head attention mechanism, and a point-wise fully connected network. The inputs to the encoder first flow through a self-attention sublayer, which helps the encoder attends to several words in the sentence as it encodes a specific word. Because the model lacks recurrent layers, this sublayer is the only mechanism which propagates information between positions in the sentence. The self-attention maps the input to three vectors called query, key and value and defines an attention function as mapping a query and a set of key-value pairs to an output vector. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Each self-attention sublayer has several attention heads, where each head has its own query, key and value weight matrices. The multi-head attention allows the model to jointly attend to information from several different positions. The outputs of the self-attention layer are fed to a feed-forward neural network, which is applied to each position independently. For further detail, see Vaswani et al. (2017).

We believe that the self-attention mechanism is especially useful for detecting disfluencies in a sentence. In pilot experiments we found that similar LSTM-based parsers, such as the AllenNLP parser (Gardner et al., 2018), were much worse at disfluency detection than the self-attentive parser. As shown in Figure 1, the "rough copy" similarity between the repair and the reparandum is a strong indicator of disfluency. "Rough copies" involve same or very similar words in roughly same word order; for example, in the Switchboard training data, over 60% of the words in the reparandum are exact copies of the words in the repair. Using the multi-head self-attention mechanism the model can presumably learn to focus on "rough copies" when detecting a reparandum.

3.2 Tree Score and Chart Parse Decoder

A chart-based parser scores a tree as a sum of potentials on its labeled constituent spans as follows:

$$s(T) = \sum_{(i,j,l)\in T} s(i,j,l) \qquad (2)$$

where $s(i, j, l)$ is a score of a constituent located between string positions i and j with label l. At test time, a modified CYK algorithm is used to find the highest scoring parse tree for a given sentence.

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, s(T) \tag{3}$$

Given the gold tagged tree $T^\star$, we train the model by minimizing a hinge loss:

$$max\left(0, \max_{T \neq T^\star}[s(T) + \triangle(T, T^\star)] - s(T^\star)\right) \tag{4}$$

where $\triangle$ is the Hamming loss on labeled spans. For further detail, see Kitaev and Klein (2018) and Stern et al. (2017).

3.3 External Embedding and Edited Loss

Peters et al. (2018) have recently introduced a new approach for word representation called Embeddings from Language Models (ELMo) which has achieved state-of-the-art results in various NLP tasks. These embeddings are produced by a LSTM language model (LM) which inputs words and characters and generates a vector representation for each word of the sentence. The ELMo output is a concatenation of both the forward and backward LM hidden states. We found that using external ELMo embedding as the only lexical representation used by the model leads to the highest EDITED word f-score. Following Kitaev and Klein (2018), we use a trainable weight matrix to project the ELMo pretrained weights of 1024 dimension to a 512-dimensional content representation. We tried different combinations of input including predicted POS tags, character LSTM and word embeddings with ELMo, but the result was either worse or not significantly better than when using ELMo alone.

The sole change we made to the self-attentive parser was to modify the loss function, so it puts more weight onto EDITED nodes. We show below that this improves the model's ability to recover EDITED nodes. We modify the tree scoring in 2 as follows:

$$s(T) = \sum_{(i,j,l) \in T} w_l\, s(i, j, l) \tag{5}$$

where w_l depends on the label l. We only used two different values of w_l here, one for EDITED nodes and one for all other node labels. We treat these as hyperparameters, and tune them to maximize EDITED nodes f-score (this is $\mathrm{F(S_E)}$) in Section 4.1 below).

4 Experiments

We evaluate the self-attentive parser on the Penn Treebank-3 Switchboard corpus (Mitchell et al., 1999). Following Charniak and Johnson (2001), we split the Switchboard corpus into training, dev and test sets as follows: training data consists of the sw[23]∗.mrg files, dev data consists of the sw4[5-9]∗.mrg files and test data consists of the sw4[0-1]∗.mrg files. Except as explicitly noted below, we remove all partial words (words tagged XX and words ending in "-") and punctuation from data, as they are not available in realistic ASR applications (Johnson and Charniak, 2004).

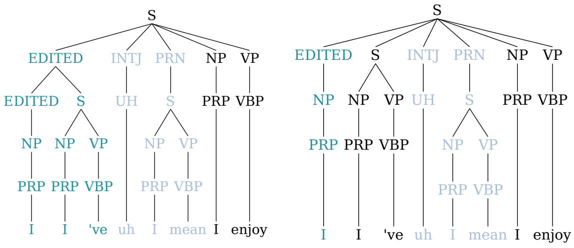
4.1 Evaluation Metrics

We evaluate the self-attentive parser in terms of parsing accuracy and disfluency detection performance. We report *precision* (P), *recall* (R) and *f-score* (F) for both *constituent spans* (S) and *word positions* (W), treating each word position as labeled by all the constituents that contain that word. We also consider subsets of constituent spans and word positions; specifically: (i) $\mathrm{S_E}$, the set of constituent spans labeled EDITED, (ii) $\mathrm{W_E}$, the set of word positions dominated by one or more EDITED nodes, and (iii) $\mathrm{W_{EIP}}$, the set of word positions dominated by one or more EDITED, INTJ or PRN nodes.

We demonstrate the evaluation metrics with an example here. Consider the gold and predicted parse trees illustrated in Figure 2. The constituency trees are viewed as a set of labeled spans over the words of the sentence, where constituent spans are pairs of string positions. As explained earlier, we ignore punctuation and partial words when calculating evaluation scores. To calculate fscore for a span, i.e., F(S), the gold, predicted and correct labeled spans are counted. In this case, the number of predicted, gold and correctly predicted spans is 13, 14 and 12.

Since a parse tree with EDITED nodes identifies certain words as EDITED, we can evaluate how accurately a parser classifies words as EDITED (i.e. $\mathrm{F(W_E)}$). Continuing with the example in Figure 2, the number of predicted, gold and correctly predicted EDITED words is 1, 3 and 1.

Similarly, we can also measure how well the parser can identify all disfluency words, i.e., the words dominated by EDITED, INTJ or PRN nodes. Continuing with the example in Figure 2, the number of predicted, gold and correctly pre-



(a) Gold tree

(b) Predicted tree

$_{0}\ I\ _{1}\ I\ _{2}\ \text{'ve}\ _{3}\ \text{uh}\ _{4}\ I\ _{5}\ \text{mean}\ _{6}\ I\ _{7}\ \text{enjoy}\ _{8}$

Figure 2: An example gold and predicted parse tree.

dicted EDITED, INTJ and PRN words is 4, 6, 4.

4.2 Model Training

We use randomized search (Bergstra and Bengio, 2012) to tune the optimization and architecture parameters of the model on the dev set. We optimize the model for its performance on parsing EDITED nodes F(S_E). The hyperparameters include dimensionality of the model, learning rate, edited loss weight, dropout, number of layers and heads as shown in Table 1. All other hyperparameters not mentioned here are the same as in Kitaev and Klein (2018).

Configuration	**Parser**
hidden label dim	340
model dim	2048
non-EDITED label weight	0.7
EDITED label weight	2
learning rate	0.0006
learning rate warmup steps	110
step decay factor	0.52
num heads	7
num layers	4
attention dropout	0.27
relu dropout	0.09
residual dropout	0.26
elmo dropout	0.57
tag embedding dropout	0.35
word embedding dropout	0.2

Table 1: Hyperparameter setting for the self-attentive constituency parser.

4.3 Edited Loss

Our best dev model (see Table 1) uses an edited loss that puts more weight on EDITED nodes and less weight on non-EDITED nodes. To explore the effect of edited loss, we retrained the best model with an equally weighted loss. The results in Table 2 indicate that differential weighting improves parsing EDITED nodes as well as EDITED word detection. It also rebalances the precision vs. recall trade-off and slightly increases overall parsing accuracy F(S).

	equal weight	**different weight**
P(S_E)	83.0	83.3
R(S_E)	91.6	91.4
F(S_E)	87.1	87.2
F(S)	92.8	93
F(W_E)	86.9	87.5

Table 2: Parsing precision P(S_E), recall R(S_E) and f-score F(S_E) of EDITED nodes, parsing f-score F(S) and EDITED word f-score F(W_E) on the Switchboard dev set for the equally and differentially weighted loss.

4.4 Modifying the Training Data

We investigate the effect of modifying the training data on the performance of the parser.

4.4.1 Simplified Tree Structures

We use different tree-transformations to explore the effect of different amounts of and encodings of disfluencies and syntactic information on the performance of the model.

- **Baseline:** Parse trees as they appear in the Switchboard corpus. A sample is shown in Figure 3.

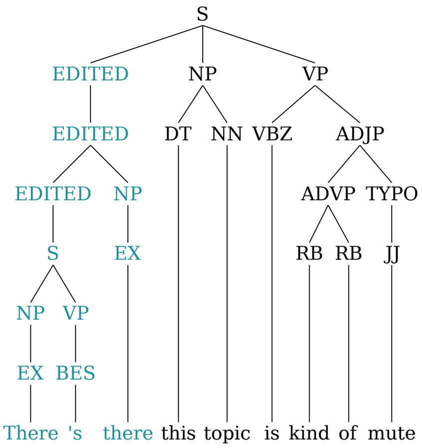


Figure 3: Baseline parse trees as they appear in the Switchboard corpus

- **Transformation PosDisfl:** Pushing disfluency nodes (i.e. EDITED, INTJ and PRN) down to POS tags, as shown in Figure 4.

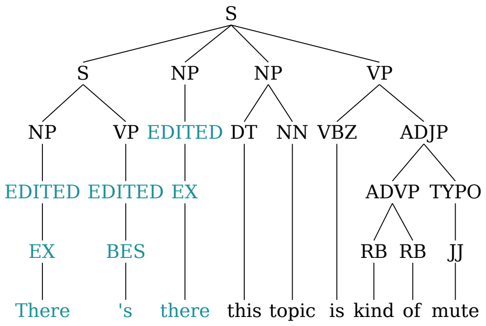


Figure 4: Transformation PosDisfl, where disfluency nodes are pushed down to POS tags.

- **Transformation NoSyntax:** Deleting all non-disfluency nodes, as shown in Figure 5.

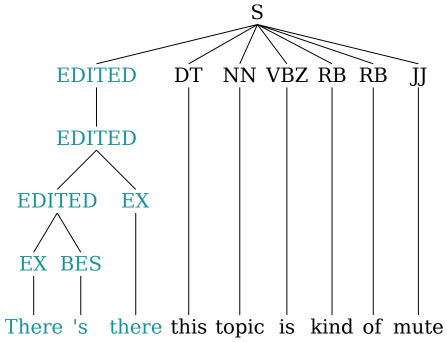


Figure 5: Transformation NoSyntax, where all non-disfluency nodes are deleted.

- **Transformation PosDisfl+NoSyntax:** Pushing disfluency nodes down to POS tags and deleting all non-disfluency nodes, as shown in Figure 6.

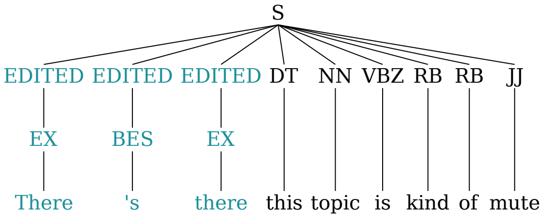


Figure 6: Transformation PosDisfl+NoSyntax, where disfluency nodes are pushed down to POS tags and all non-disfluency nodes are deleted.

- **Transformation TopDisfl:** Deleting all disfluency nodes but the top ones, as shown in Figure 7.

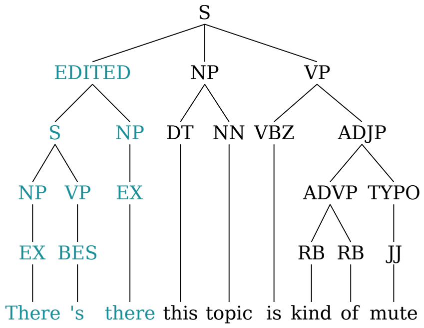


Figure 7: Transformation TopDisfl, where all disfluency nodes but the top ones are deleted.

- **Transformation TopDisfl+NoSyntax:** Deleting all nodes but the top-most disfluency nodes, as shown in Figure 8.

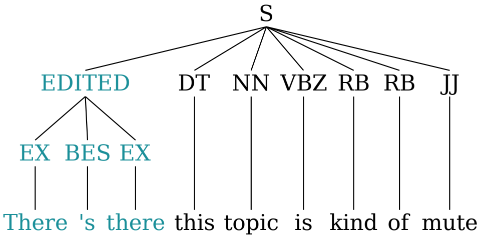


Figure 8: Transformation TopDisfl+NoSyntax, where all nodes but the top-most disfluency nodes are deleted.

We report the performance of the self-attentive parser in terms of EDITED word f-score and disfluency word f-score in Table 3. Since the transformations change the tree shapes, it is not meaningful to compare their parsing f-scores. As illustrated in Table 3, pushing disfluency nodes down to POS tags (i.e. *Transformation PosDisfl*) increases precision about 2%, resulting in 1% improvement in word f-score $F(W_E)$. It also improves $F(W_{EIP})$ by 0.4%. In general, the model can take advantage of the simplified encoding of disfluency nodes (see *Transformations PosDisfl* and *TopDisfl*). Moreover, deleting all but the top-most disfluency nodes as in *Transformation TopDisfl+NoSyntax* significantly drops precision (about 20%), resulting in more than 13% decrease in EDITED word f-score. It also hurts detecting all types of disfluency (more than 7% decrease in $F(W_{EIP})$). In general, removing syntactic structure dramatically degrades the performance of the model in terms of $F(W_E)$ and $F(W_{EIP})$, as shown in *Transformations NoSyntax*, *PosDisfl+NoSyntax* and *TopDisfl+NoSyntax*. This indicates that syntactic information is important for detecting disfluencies.

Setting	**$P(W_E)$**	**$R(W_E)$**	**$F(W_E)$**	**$F(W_{EIP})$**
Baseline	81.6	94.2	87.5	94.0
PosDisfl	83.7	94.2	**88.7**	**94.4**
NoSyntax	73.0	95.0	82.5	92.3
PosDisfl+NoSyntax	73.4	93.4	82.2	91.6
TopDisfl	81.9	94.3	87.7	93.8
TopDisfl+NoSyntax	61.3	93.1	74.0	86.7

Table 3: EDITED word precision $P(W_E)$, recall $R(W_E)$ and f-score $F(W_E)$ as well as EDITED, INTJ and PRN word f-score $F(W_{EIP})$ on the Switchboard dev set for different encodings of disfluency nodes in data. The best f-scores are shown in bold.

4.4.2 Punctuation and Partial words

As mentioned before, speech recognition models generally do not produce punctuation and partial words in their outputs. Thus, prior work has removed them from the data to make the evaluation more realistic. However, it is interesting to see what information partial words and punctuation convey about syntactic structure in general and disfluencies in particular, so we did an experiment to investigate the effect of including these in the training and test data. We use the best hyperparameter configuration on the Switchboard dev set and retrain the model on two versions of the data: (i) with partial words and (ii) with punctuation and partial words. As shown in Table 4, keeping punctuation and partial words in the training data increases EDITED word f-score by about 4%, indicating that punctuation and partial words greatly help disfluency detection. Punctuation leads to more gain in disfluency detection than partial words. Punctuation also improves the word f-score for all types of disfluencies by more than 1%.

Setting	**$F(W_E)$**	**$F(W_{EIP})$**
without punctuation & partial words	88.7	94.4
with partial words	89.7	94.4
with punctuations & partial words	92.2	95.5

Table 4: EDITED word $F(W_E)$ and EDITED, INTJ and PRN word f-score $F(W_{EIP})$ on the Switchboard dev set for three versions of the training data.

5 Results

We selected our best model based on the dev set results (including differentially weighted loss) and compared the results achieved for the *Tree Transformation PosDisfl* and *No Tree Transformation* on the test set with previous work. Although most previous work has used the Switchboard corpus, it is sometimes difficult to compare systems directly due to different scoring metrics and differences in experimental setup, such as the use of partial words, punctuation, prosodic cues and so on. Since some studies report their results using partial words and/or punctuation, we divide prior work according to the setting they used and report the results of the self-attentive parser on the test data for each setting.

Table 5 shows the test performance of the self-attentive constituency parser against previous parsing models of speech transcripts. The self-attentive parser outperforms all previous models in parsing accuracy. It has also better performance than Kahn et al. (2005) and Tran et al. (2018), who used acoustic/prosodic cues from speech waveform as well as the words in the transcript.

Parsing Model	**F(S)**
without partial words	
self-attentive parser (PDT)	92.4
self-attentive parser (NT)	**92.7**
partial words:	
Hale et al. (2006)	71.1
Kahn et al. (2005)	86.6
Tran et al. (2018)	88.5
self-attentive parser (NT)	92.3
self-attentive parser (PDT)	**92.6**

Table 5: Parse f-score F(S) for all constituent spans on the Switchboard test set with and without partial words. NT = No Transformation and PDT = PosDisfl Transformation.

We also compare the performance of the self-attentive parser with state-of-the-art disfluency detection methods in terms of EDITED word f-score. As shown in Table 6, the self-attentive parser (with *PosDisfl Transformation*) achieves a new state-of-the-art for detecting EDITED words. Its performance is competitive with specialized disfluency detection models that directly optimize for disfluency detection. Using partial words increases edited word f-score for *No Transformation* mode by 0.1% and for *PosDisfl Transformation* mode by 0.6%, which is not surprising as the presence of partial words is strongly correlated with the presence of a disfluency.

It is interesting to compare the self-attentive parser with the ACNN model presented in Jamshid Lou et al. (2018). They introduce a new ACNN layer which is able to learn the "rough copy" dependencies between words, for which previous models heavily relied on hand-crafted pattern-matching features. "Rough copies" are a strong indicator of disfluencies that can help the model detect reparanda (i.e. EDITED nodes). That the self-attentive parser is better than the ACNN model (Jamshid Lou et al., 2018) in detecting disfluencies may indicate that the self-attention mechanism can learn "rough copy" dependencies.

Model	**F(W_E)**
without partial words:	
Honnibal and Johnson (2014)	84.1
Jamshid Lou et al. (2018) •	84.5
Wu et al. (2015)	85.1
Ferguson et al. (2015) •	85.4
Wang et al. (2016) •	86.7
Jamshid Lou and Johnson (2017) •	86.8
self-attentive parser (NT)	86.9
Wang et al. (2017) •	87.5
self-attentive parser (PDT)	**88.1**
partial words:	
Hale et al. (2006)	41.7
Tran et al. (2018)	77.5
Kahn et al. (2005)	78.2
Rasooli and Tetreault (2013)	81.4
Zayats et al. (2016) •	85.9
self-attentive parser (NT)	87.0
self-attentive parser (PDT)	**88.7**

Table 6: Edited word f-score F(W_E) on the Switchboard test set with and without partial words. • Specialized disfluency detection models. NT = No Transformation and PDT = PosDisfl Transformation.

We also compare the performance of the self-attentive parser with Wang et al.'s (2018) self-attentive disfluency detection model in terms of disfluency (i.e. EDITED, INTJ and PRN) word f-score. As shown in Table 7, the self-attentive parser outperforms this state-of-the-art specialized self-attentive disfluency detection model.

Self-attentive Model	**F(W_{EIP})**
punctuation & partial words	
Wang et al. (2018)	91.1
self-attentive parser (NT)	93.7
self-attentive parser (PDT)	**94.0**

Table 7: EDITED, INTJ and PRN word f-score F(W_{EIP}) on the Switchboard test set with punctuation and partial words. NT = No Transformation and PDT = PosDisfl Transformation.

6 Conclusion and Future Work

This paper shows that using an "off-the-shelf" constituency parser achieves a new state-of-the-art in parsing transcribed speech. The self-attentive parser is effective in detecting disfluent words as it outperforms specialized disfluency detection models, suggesting that it is feasible to use standard neural architectures to perform disfluency detection as part of some other task, rather than requiring a separate disfluency detection pre-processing step. We also show that removing syntactic information hurts word f-score. That is, performing syntactic parsing and disfluency detection as a multi-task training objective yields higher disfluency detection accuracy than performing disfluency detection in isolation. Modifying encoding by indicating disfluencies at the word level leads to further improvements in disfluency detection.

In future work we hope to integrate syntactic parsing more closely with automatic speech recognition. A first step is to develop parsing models that parse ASR output, rather than speech transcripts. It may also be possible to more directly integrate an attention-based syntactic parser with a speech recogniser, perhaps trained in an end-to-end fashion.

Acknowledgments

This research was supported by a Google award through the Natural Language Understanding Focused Program, CRP 8201800363 from Data61/CSIRO, and under the Australian Research Councils Discovery Projects funding scheme (project number DP160102156). We also thank the anonymous reviewers for their valuable comments that helped to improve the paper.

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies (NAACL)*, pages 118–126, Stroudsburg, USA.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-Markov model and prosodic features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pages 257–262, Denver, USA.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

John Godfrey and Edward Holliman. 1993. Switchboard-1 release 2 LDC97S62. Published by: Linguistic Data Consortium, Philadelphia, USA.

John Hale, Izhak Shafran, Lisa Yung, Bonnie J. Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang (2) Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. Pcfgs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 161–168, Sydney, Australia. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics (TACL)*, 2(1):131–142.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency detection using auto-correlational neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4610–4619, Brussels, Belgium.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency detection using a noisy channel model and a deep neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 547–553, Vancouver, Canada.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04)*, pages 33–39, Barcelona, Spain.

Jeremy Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT'05)*, pages 233–240, Tallinn, Estonia.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Matthew Lease and Mark Johnson. 2006. Early deletion of fillers in processing conversational speech. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 73–76, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marcus Mitchell, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 LDC99T42. Published by: Linguistic Data Consortium, Philadelphia, USA.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 124–129, Seattle, USA.

Elizabeth Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, University of California, Berkeley, USA.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827. Association for Computational Linguistics.

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2018. Parsing speech: A neural approach to integrating lexical and acoustic-prosodic information. In *Proceedings of the Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 69–81, New Orleans, USA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, USA.

Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. 2018. Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 3529–3538, Santa Fe, USA.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING): Technical Papers*, pages 278–287, Osaka, Japan.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2785–2794, Copenhagen, Denmark.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 495–503. Association for Computational Linguistics.

Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1036–1041.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional LSTM. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2523–2527, San Francisco, USA.

Acoustic-to-Word Models with Conversational Context Information

Suyoun Kim[1] **and Florian Metze**[2]
[1]Electrical & Computer Engineering
[2]Language Technologies Institute, School of Computer Science
Carnegie Mellon University
`{suyoung1, fmetze}@andrew.cmu.edu`

Abstract

Conversational context information, higher-level knowledge that spans across sentences, can help to recognize a long conversation. However, existing speech recognition models are typically built at a sentence level, and thus it may not capture important conversational context information. The recent progress in end-to-end speech recognition enables integrating context with other available information (e.g., acoustic, linguistic resources) and directly recognizing words from speech. In this work, we present a direct acoustic-to-word, end-to-end speech recognition model capable of utilizing the conversational context to better process long conversations. We evaluate our proposed approach on the Switchboard conversational speech corpus and show that our system outperforms a standard end-to-end speech recognition system.

1 Introduction

Many real-world speech recognition applications, including teleconferencing, and AI assistants, require recognizing and understand long conversations. In a long conversation, there exists the tendency of semantically related words or phrases reoccur across sentences, or there exists topical coherence. Thus, such conversational context information, higher-level knowledge that spans across sentences, provides important information that can improve speech recognition. However, the long conversations typically split into short sentence-level audios to make building speech recognition models computationally feasible in current state-of-the-art recognition systems (Xiong et al., 2017; Saon et al., 2017).

Over the years, there have been many studies have attempted to inject a longer context information into language models. Based on a recurrent neural network (RNNs) language models (Mikolov et al., 2010), (Mikolov and Zweig, 2012; Wang and Cho, 2015; Ji et al., 2015; Liu and Lane, 2017; Xiong et al., 2018), proposed using a context vector that would encode the longer context information as an additional network input. However, all of these models have been developed on text data, and therefore, it must still be integrated with a conventional acoustic model which is built separately without a longer context information, for speech recognition on long conversations.

Recently, new approaches to speech recognition models integrate all available information (e.g. acoustic, linguistic resources) in a so-called end-to-end manner proposed in (Graves et al., 2006; Graves and Jaitly, 2014; Hannun et al., 2014; Miao et al., 2015; Bahdanau et al., 2014; Chorowski et al., 2014, 2015; Chan et al., 2015; Kim et al., 2017). In these approaches, a single neural network is trained to recognize graphemes or even words from speech directly. Especially, the model using semantically meaningful units, such as words or sub-word (Sennrich et al., 2015), rather than graphemes have been showing promising results (Audhkhasi et al., 2017b; Li et al., 2018; Soltau et al., 2016; Zenkel et al., 2017; Palaskar and Metze, 2018; Sanabria and Metze, 2018; Rao et al., 2017; Zeyer et al., 2018).

In this work, motivated by such property of the end-to-end speech recognition approaches, we propose to integrate conversational context information within a direct acoustic-to-word, end-to-end speech recognition to better process long conversations. Thus far, the research in speech recognition systems has focused on recognizing sentences and to the best of our knowledge, there have been no studies of word-based models incorporating conversational context information. There has been recent work attempted to use the conversational context information from the preceding graphemes (Kim and Metze, 2018), however, it is

Proceedings of NAACL-HLT 2019, pages 2766–2771
Minneapolis, Minnesota, June 2 - June 7, 2019.

limited to encode semantically meaningful context representation. Another recent work attempted to use a context information (Pundak et al., 2018), however, their method requires a list of phrases at inference (i.e. personalized contact list). We evaluate our proposed approach on the Switchboard conversational speech corpus (Godfrey and Holliman, 1993; Godfrey et al., 1992), and show that our model outperforms the sentence-level end-to-end speech recognition model.

2 Models

2.1 Acoustic-to-Words Models

We perform end-to-end speech recognition using a joint CTC/Attention-based approach (Kim et al., 2017; Watanabe et al., 2017). The neural network is trained by both CTC (Graves et al., 2006) and Attention-based sequence-to-sequence (seq2seq) objectives (Bahdanau et al., 2014) to combine the strength of the two. With CTC, it preserves left-right order between input and output and with attention-based seq2seq, it learns the language model jointly without relying on the conditional independence assumption.

As an output, we use word-level symbols which generated from the bite-pair encoding (BPE) algorithm (Sennrich et al., 2015). This method creates the target units based on the frequency of occurrence in training sets. Similar to (Zeyer et al., 2018; Palaskar and Metze, 2018; Sanabria and Metze, 2018), we use BPE-10k which contains roughly 10k units (9,838), including 7,119 words and 2719 sub-words.

2.2 Conversational Context Representation

In order to use conversational context information within the end-to-end speech recognition framework, we extend the decoder sub-network to predict the output additionally conditioning on conversational context. To do so, we encode the preceding sentence into a single vector, a conversational context vector, then inject to decoder network as an additional input at every output step.

Let we have K sentences in a conversation. For k-th sentence, s^k, we have T^k-length input acoustic feature ($\boldsymbol{x^k}$) and U^k-length output words. Our proposed decoder generates the probability distribution over words (y_u^k), conditioned on 1) high-level representation ($\boldsymbol{h^k}$) of input ($\boldsymbol{x^k}$) generated from encoder, and 2) all the words seen previously ($y_{1:u-1}^k$), and 3) previous decoder state (d_{u-1}^k) 4) additionally conditioning on conversational context vector (c^{k-1}), which represents the information of the preceding sentence ($k-1$):

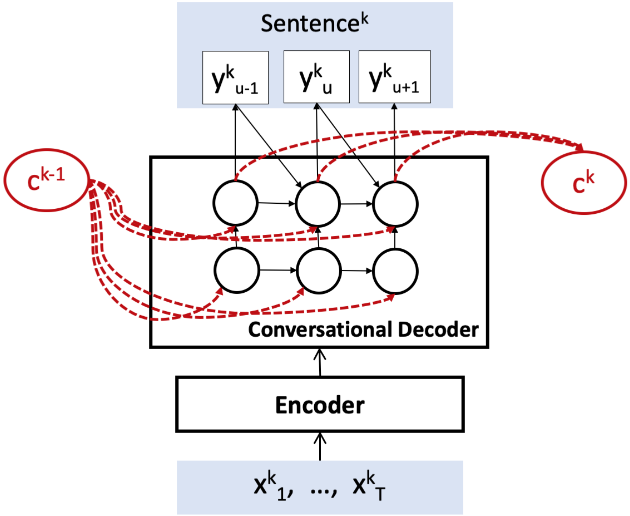


Figure 1: The architecture of our end-to-end speech recognition model with conversational context information. The c^{k-1} is the conversational context vector generated from the preceding $k-1$ sentence red curved dashed line represents the context information flow within the same conversation.

$$\boldsymbol{h^k} = Encoder(\boldsymbol{x^k}) \tag{1}$$

$$y_u^k \sim Decoder(\boldsymbol{h^k}, y_{1:u-1}^k, d_{u-1}^k, c^{k-1}) \tag{2}$$

We represent the context vector, c^{k-1}, from the preceding sentence in two different ways: (a) mean of word embedding, and (b) attentional word embedding. We first generate one-hot word vectors, and then we simply take the mean over word vectors to obtain a single vector in method (a), or we use attention mechanism over word vectors to obtain the weight over the words and then perform the weighted-sum. The parameter of the attention mechanism is optimized towards minimizing the conversation ID classification error similar to (Kim and Metze, 2018). The context vector is merged with a decoder state at every output step as follows:

$$\hat{d}_{u-1}^k = \tanh(W d_{u-1}^k + V c^{k-1} + b) \tag{3}$$

$$y_u^k \sim \text{softmax}(\text{LSTM}(\hat{d}_{u-1}^k, \boldsymbol{h_u^k}, y_{1:u-1}^k)) \tag{4}$$

where W, V, b are trainable parameters.

In order to learn and use the conversational-context during training and decoding, we serialize the sentences based on their onset times and their conversations rather than the random shuffling of data. We shuffle data at the conversation level and create mini-batches that contain only one sentence of each conversation.

3 Experiments

3.1 Datasets

We investigated the performance of the proposed model on the Switchboard LDC corpus (97S62) which has a 300 hours training set. We split the Switchboard data into two groups, then used 285 hours of data (192 thousand sentences) for model training and 5 hours of data (4 thousand sentences) for hyper-parameter tuning. The evaluation was carried out on the HUB5 Eval 2000 LDC corpora (LDC2002S09, LDC2002T43), which have 3.8 hours of data (4.4 thousand sentences), and we show separate results for the Callhome English (CH) and Switchboard (SWB) evaluation sets. We denote train_nodup, train_dev, SWB, and CH as our training, development, and two evaluation datasets for CH and SWB, respectively. There are 2,402 conversations in training sets and 20 conversations in CH, and 20 conversations in SWB.

We sampled all audio data at 16kHz, and extracted 80-dimensional log-mel filterbank coefficients with 3-dimensional pitch features, from 25 ms frames with a 10ms frame shift. We used 83-dimensional feature vectors to input to the network in total. We used 9,840 distinct labels: 9,838 word-level BPE units, start-of-speech/end-of-speech, and blank tokens. Note that no pronunciation lexicon was used in any of the experiments.

3.2 Training and decoding

We used joint CTC/Attention end-to-end speech recognition architecture (Kim et al., 2017; Watanabe et al., 2017) with ESPnet toolkit (Watanabe et al., 2018). We used a CNN-BLSTM encoder as suggested in (Zhang et al., 2017; Hori et al., 2017). We followed the same six-layer CNN architecture as the prior study, except we used one input channel instead of three since we did not use delta or delta delta features. Input speech features were downsampled to (1/4 x 1/4) along with the time-frequency axis. Then, the 6-layer BLSTM with 320 cells was followed by CNN. We used a location-based attention mechanism (Chorowski et al., 2015), where 10 centered convolution filters of width 100 were used to extract the convolutional features.

The decoder network of both our proposed models and the baseline models was a 2-layer LSTM with 300 cells. Our proposed models additionally require linear projection layer in order to encode the conversational context vector and merge with decoder states.

We also built an external RNN-based language model (RNNLM) on the same BPE-10k sets on the same Switchboard transcriptions. The RNNLM network architecture was a two-layer LSTM with 650 cells. This network was used only for decoding.

The AdaDelta algorithm (Zeiler, 2012) with gradient clipping (Pascanu et al., 2013) was used for optimization. We used $\lambda = 0.5$ for joint CTC/Attention training. We bootstrap the training our proposed conversational end-to-end models from the baseline end-to-end models. When we decode with RNNLM, we used joint decoder which combines the output label scores from the AttentionDecoder, CTC, and RNNLM by using shallow fusion (Hori et al., 2017):

$$\begin{aligned} \boldsymbol{y}* = \text{argmax}\{ & \log p_{att}(\boldsymbol{y}|\boldsymbol{x}) \\ & + \alpha \log p_{att}(\boldsymbol{y}|\boldsymbol{x}) \\ & + \beta \log p_{rnnlm}(\boldsymbol{y})\} \end{aligned} \tag{5}$$

The scaling factor of CTC, and RNNLM scores were $\alpha = 0.3$, and $\beta = 0.3$, respectively. We used a beam search algorithm similar to (Sutskever et al., 2014) with the beam size 10 to reduce the computation cost. We adjusted the score by adding a length penalty, since the model has a small bias for shorter utterances. The final score $s(\boldsymbol{y}|\boldsymbol{x})$ is normalized with a length penalty 0.5.

The models were implemented by using the PyTorch deep learning library (Paszke et al., 2017), and ESPnet toolkit (Kim et al., 2017; Watanabe et al., 2017, 2018).

4 Results

We evaluated both the end-to-end speech recognition model which was built on sentence-level data and our proposed end-to-end speech recognition model which leveraged conversational context information.

Table 1 shows the WER of our baseline, proposed models, and several other published results those were only trained on 300 hours Switchboard training data. As shown in Table 1, we obtained a performance gain over our baseline by using the

Table 1: Comparison of word error rates (WER) on Switchboard 300h with standard end-to-end speech recognition models and our proposed end-to-end speech recognition models with conversational context.

Model	Output Units	LM	SWB (WER %)	CH (WER %)
Prior Models				
LF-MMI (Povey et al., 2016)	context-dependend phones	O	9.6	19.3
CTC (Zweig et al., 2017)	Char	O	19.8	32.1
CTC (Audhkhasi et al., 2017a)	Word (Phone init.)	O	14.6	23.6
CTC (Sanabria and Metze, 2018)	Char, BPE-{300, 1k, 10k}	O	12.5	23.7
Seq2Seq (Palaskar and Metze, 2018)	BPE-10k	O	21.3	35.7
Seq2Seq (Zeyer et al., 2018)	BPE-1k	O	11.8	25.7
Our Sentence-level Baseline				
Our baseline	BPE-10k	x	17.6	30.6
Our baseline	BPE-10k	O (only swb)	17.0	29.7
Our Proposed Conversational Model				
w/ Context *(a) mean*	BPE-10k	O (only swb)	16.3	29.0
w/ Context *(b) att*	BPE-10k	O (only swb)	16.4	29.2
w/ Context *(b) att* + pre-training	BPE-10k	O (only swb)	16.0	28.9

Table 2: Perplexities on a held-out set of our proposed conversational context LM and baselines.

Models	Fisher text	PPL
Baseline LM	x	74.15
Baseline LM	o	72.81
Proposed Conversational LM	x	67.03
Proposed Conversational LM	o	64.30

conversational context information. Our proposed model *(a) mean* shows 4.1% and 2.4% relative improvement over our baseline on SWB and CH evaluation set, respectively. Our proposed model *(b) att* shows 3.5% and 1.7% relative improvement over our baseline on SWB and CH evaluation set, respectively. We also found that we can obtain further accuracy improvement by pre-training the decoder part only with transcription. With this pre-training technique, the *(b) att* shows 5.9% and 2.7% relative improvement. Unlike the previous work (Renduchintala et al., 2018), we did not use any additional encoder for the text data.

We also build the language model with or without the conversational context information. Table 2 shows the perplexity on a held-out set of our baseline LM and our conversational LM. We observed that incorporating the conversational context improves performance showing that 9.6% and 11.7% relative improvement on *SWBD only* and *SWBD + Fisher*. Note that the Fisher (LDC2004T19) parts (Cieri et al., 2004) of transcriptions is only used in these experiments.

We performed analyses in order to verify the conversational vector helps to improve recognition accuracy. We generate the context vector from an oracle preceding sentence and a random sentence, in addition to our predicted sentence. As described in Figure 2, the model using the oracle context performed best and the model using the random context was even worse than the baseline. Our model outperformed over the baseline and the model using the random context, we can conclude that the benefit from our proposed method is coming from the conversational context information.

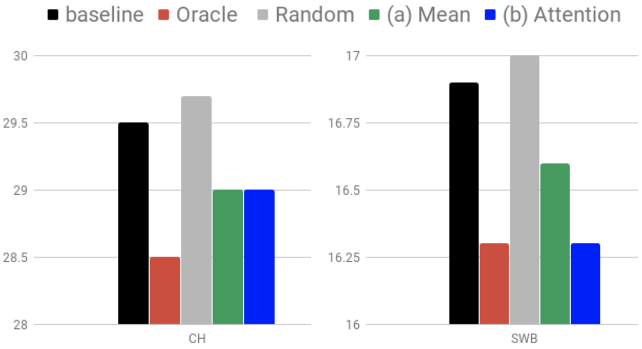


Figure 2: The architecture of our end-to-end speech recognition model with conversational context information. The c^{k-1} is the conversational context vector generated from the preceding $k-1$ sentence red curved line represents the context information flow within the same conversation.

5 Conclusion

We proposed an acoustic-to-word model capable of utilizing the conversational context to better process long conversations. A key aspect of our model is that the whole system can be trained with conversational context information in an end-to-end framework. Our model was shown to outperform previous end-to-end speech recognition models trained on isolated utterances by incorporating preceding conversational context representations.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. This work also used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC). This research was supported by a fellowship from the Center for Machine Learning and Health (CMLH) at Carnegie Mellon University.

References

Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny. 2017a. Building competitive direct acoustics-to-word models for english conversational speech recognition. *arXiv preprint arXiv:1712.03133*.

Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo. 2017b. Direct acoustics-to-word models for english conversational speech recognition. *arXiv preprint arXiv:1703.07754*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*, volume 4, pages 69–71.

John Godfrey and Edward Holliman. 1993. Switchboard-1 release 2 ldc97s62. *Linguistic Data Consortium, Philadelphia*, LDC97S62.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. 2017. Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. *arXiv preprint arXiv:1706.02737*.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962*.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4835–4839. IEEE.

Suyoun Kim and Florian Metze. 2018. Dialog-context aware end-to-end speech recognition. *arXiv preprint arXiv:1808.02171*.

Jinyu Li, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong. 2018. Advancing acoustic-to-word ctc model. *arXiv preprint arXiv:1803.05566*.

Bing Liu and Ian Lane. 2017. Dialog context language modeling with recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5715–5719. IEEE.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. *SLT*, 12:234–239.

Shruti Palaskar and Florian Metze. 2018. Acoustic-to-word recognition with sequence-to-sequence models. *arXiv preprint arXiv:1807.09597*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755.

Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao. 2018. Deep context: end-to-end contextual speech recognition. *arXiv preprint arXiv:1808.02480*.

Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. 2017. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 193–199. IEEE.

Adithya Renduchintala, Shuoyang Ding, Matthew Wiesner, and Shinji Watanabe. 2018. Multi-modal data augmentation for end-to-end asr. *arXiv preprint arXiv:1803.10299*.

Ramon Sanabria and Florian Metze. 2018. Hierarchical multi task learning with ctc. *arXiv preprint arXiv:1807.07104*.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. 2017. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Hagen Soltau, Hank Liao, and Hasim Sak. 2016. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al. 2018. Espnet: End-to-end speech processing toolkit. *arXiv preprint arXiv:1804.00015*.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2017. The microsoft 2016 conversational speech recognition system. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5255–5259. IEEE.

Wayne Xiong, Lingfeng Wu, Jun Zhang, and Andreas Stolcke. 2018. Session-level language modeling for conversational speech. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2764–2768.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Thomas Zenkel, Ramon Sanabria, Florian Metze, and Alex Waibel. 2017. Subword and crossword units for ctc acoustic models. *arXiv preprint arXiv:1712.06855*.

Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2018. Improved training of end-to-end attention models for speech recognition. *arXiv preprint arXiv:1805.03294*.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional networks for end-to-end speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4845–4849. IEEE.

Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke. 2017. Advances in all-neural speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4805–4809. IEEE.

A Dynamic Speaker Model for Conversational Interactions

Hao Cheng **Hao Fang** **Mari Ostendorf**
University of Washington
{chenghao,hfang,ostendorf}@uw.edu

Abstract

Individual differences in speakers are reflected in their language use as well as in their interests and opinions. Characterizing these differences can be useful in human-computer interaction, as well as analysis of human-human conversations. In this work, we introduce a neural model for learning a dynamically updated speaker embedding in a conversational context. Initial model training is unsupervised, using context-sensitive language generation as an objective, with the context being the conversation history. Further fine-tuning can leverage task-dependent supervised training. The learned neural representation of speakers is shown to be useful for content ranking in a socialbot and dialog act prediction in human-human conversations.[1]

1 Introduction

Representing language in context is key to improving natural language processing (NLP). There are a variety of useful contexts, including word history, related documents, author/speaker information, social context, knowledge graphs, visual or situational grounding, etc. This paper addresses the problem of modeling the speaker. Accounting for author/speaker variations has been shown to be useful in many NLP tasks, including language understanding (Hovy and Søgaard, 2015; Volkova et al., 2013), language generation (Mirkin et al., 2015; Li et al., 2016), human-computer dialog policy (Bowden et al., 2018), query completion (Jaech and Ostendorf, 2018; Shokouhi, 2013), comment recommendation (Agarwal et al., 2011) and more. In this work, we specifically focus on dialogs, including both human-computer (socialbot) and human-human conversations.

While many studies rely only on discrete metadata and/or demographic information, such information is not always available. Thus, it is of interest to learn about the speaker from the language directly, as it relates to the person's interests and speaking style. Motivated by the success of unsupervised contextualized representation learning for words and documents (Mikolov et al., 2013; Kiros et al., 2015; McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019), our approach is to use unsupervised learning with a neural model of a speaker's dialog history. The model uses latent speaker mode vectors for representing a speaker turn as in (Cheng et al., 2017), which provides a framework for analysis of what the model learns about speaking style. Further, the model is structured to allow a dynamic update of the speaker vector at each turn in a dialog, in order to capture changes over time and improve the speaker representation with added data.

The speaker embeddings can be used as context in conversational language understanding tasks, e.g., as an additional input in dialog policy prediction in human-computer dialogs or in understanding dialog acts in human-human dialogs. In the supervised training of such tasks, the speaker model can be fine-tuned.

This work makes two primary contributions. First, we propose a neural model for learning dynamically updated speaker embeddings in conversational interactions. The model training is unsupervised, relying on only the speaker's conversation history rather than meta information (e.g., age, gender) or audio signals which may not be available in a privacy-sensitive situation. The model also has a learnable component for analyzing the latent modes of the speaker, which can be helpful for aligning the learned characteristics of a speaker with the human-interpretable factors. Second, we use the learned dynamic speaker embed-

---

[1] The implementation of code is available at `https://github.com/hao-cheng/dynamic_speaker_model.git`

Proceedings of NAACL-HLT 2019, pages 2772–2785
Minneapolis, Minnesota, June 2 - June 7, 2019.

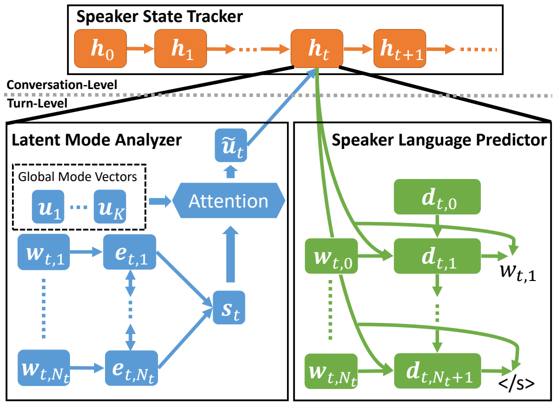


Figure 1: The dynamic speaker model. The speaker state tracker operates at the conversation level. The latent model analyzer and speaker language predictor operate at the turn level. The figure only shows processes in those two components for the turn t.

dings in two representative tasks in dialogs: predicting user topic decisions in socialbot dialogs, and classifying dialog acts in human-human dialogs. Empirical results show that using the dynamic speaker embeddings significantly outperforms the baselines in both tasks. In the public dialog act classification task, the proposed model achieves the state-of-the-art results.

2 Dynamic Speaker Model

In this section, we start with an overview of the proposed model for learning speaker embeddings that are dynamically refined over the course of a conversation. Details about individual components are described in subsequent subsections.

The model is based on two motivations. First, a speaker's utterances reflect intents, speaking style, etc. Thus, we may build speaker embeddings by analyzing latent modes that characterize utterances in terms of such characteristics, apart from topic-related interests a user might have. Second, the information about a speaker is accumulated as the conversation evolves, which allows us to gradually refine and update the speaker embeddings. The speaker embeddings can be directly used as features or fine-tuned based on the downstream tasks. We design the dynamic speaker model to focus on learning cues from the speaker's utterances, and leave the modeling of different speaker-addressee interactions for supervised downstream tasks.

The model consists of three components as illustrated in Fig. 1. First, a **latent mode analyzer** reads in an utterance and analyzes its latent modes. It processes the speaker's turns independently of each other and builds a local speaker mode vector for each turn. To accumulate speaker information as the conversation evolves, we build a **speaker state tracker** that maintains speaker states at individual turns. At each turn, it takes two input vectors to update the speaker state: 1) the local speaker mode vector for the current turn from the latent mode analyzer, and 2) the speaker state at the previous turn from the tracker itself. Finally, we employ a **speaker language predictor** to drive the learning of the latent model analyzer and the speaker state tracker. It reconstructs the utterance using the corresponding speaker state. Intuitively, the speaker language predictor models overall linguistic regularities itself and uses the speaker state to supply information related to speaker characteristics. For sequence modeling in all three components, we use the long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997). In our experiments, the three components are trained jointly.

2.1 Latent Mode Analyzer

At each turn t, the latent mode analyzer constructs a local speaker mode vector $\tilde{\mathbf{u}}_t \in \mathbb{R}^c$ that captures salient characteristics of the speaker's current utterance for use in the dynamic speaker model. First, the utterance word sequence $w_{t,1}, \cdots, w_{t,N_t}$ is mapped to an embedding sequence, where $w_{t,n}$ is represented with $\mathbf{w}_{t,n} \in \mathbb{R}^d$ according a lookup with dictionary $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}|\times d}$ associated with vocabulary $\mathcal{V}$. Then, the latent mode analyzer goes through two stages to construct $\tilde{\mathbf{u}}_t$.

In the first stage, a bi-directional LSTM (Bi-LSTM), which consists of a forward LSTM and a backward LSTM, is used to encode the word embedding sequence into a fixed-size utterance summary vector $\mathbf{s}_t \in \mathbb{R}^{2m}$, where m is the dimension of the hidden layer in the forward and backward LSTMs. Formally, the forward LSTM computes its hidden states as $\mathbf{e}^F_{t,n} = g^F(\mathbf{w}_{t,n}, \mathbf{e}^F_{t,n-1}) \in \mathbb{R}^m$ for $n = 1, \ldots, N_t$, where $g^F(\cdot,\cdot)$ denotes the forward LSTM function. The backward LSTM computes its hidden states $\mathbf{e}^B_{t,n} \in \mathbb{R}^m$ similarly. The initial hidden states $\mathbf{e}^F_{t,0}$ and $\mathbf{e}^B_{t,N_t+1}$ are set to zeros. The summary vector $\mathbf{s}_t$ is the concatenation of the two final hidden states, $\mathbf{s}_t = [\mathbf{e}^F_{t,N_t}, \mathbf{e}^B_{t,1}]$.

In the second stage, the utterance summary vector $\mathbf{s}_t$ is compared with K global mode vectors

$\mathbf{u}_1, \dots, \mathbf{u}_K \in \mathbb{R}^c$ which are learned as part of the model. The association score $a_{t,k}$ between $\mathbf{s}_t$ and $\mathbf{u}_k$ is computed using the dot-product attention mechanism (Vaswani et al., 2017) as follows,

$$a_{t,k} = \frac{\exp(\langle \mathbf{P}\mathbf{s}_t, \mathbf{Q}\mathbf{u}_k \rangle)}{\sum_{k'=1}^{K} \exp(\langle \mathbf{P}\mathbf{s}_t, \mathbf{Q}\mathbf{u}_{k'} \rangle)}, \qquad (1)$$

where $\mathbf{P} \in \mathbb{R}^{c \times 2m}$ and $\mathbf{Q} \in \mathbb{R}^{c \times c}$ are learnable weights, and $\langle \cdot, \cdot \rangle$ indicates the dot-product of two vectors. The local speaker mode vector is then constructed as $\tilde{\mathbf{u}}_t = \sum_{k=1}^{K} a_{t,k} \mathbf{u}_k$.

2.2 Speaker State Tracker

The speaker state tracker provides a dynamic summary of speaker language features observed in the conversation history, using an LSTM to encode the sequence of local speaker mode vectors $\tilde{\mathbf{u}}_{t,1}, \cdots, \tilde{\mathbf{u}}_{t,N_t}$. At turn t, this LSTM updates its hidden state $\mathbf{h}_t \in \mathbb{R}^m$ using the local speaker mode vector $\tilde{\mathbf{u}}_t$ and its previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^m$, i.e., $\mathbf{h}_t = g^S(\tilde{\mathbf{u}}_t, \mathbf{h}_{t-1})$, where $g^S(\cdot, \cdot)$ is the speaker LSTM function. The hidden state $\mathbf{h}_t$ provides the speaker state vector at turn t.

2.3 Speaker Language Predictor

The speaker language predictor is a conditional LSTM language model (LM) that reconstructs the word sequence in the current turn. Language modeling is a way to provide a training signal for unsupervised learning that models the conditional probability $\Pr(w_{t,n} | w_{t,<n})$, where $w_{t,<n}$ denotes all preceding words of $w_{t,n}$ in the turn t.

The speaker language predictor uses the same dictionary $\mathbf{W}$ for word embeddings as the latent mode analyzer to represent words at time t. The initial hidden state $\mathbf{d}_{t,0} \in \mathbb{R}^m$ of the LSTM is set to $\tanh(\mathbf{L}\mathbf{h}_t)$, where $\mathbf{L} \in \mathbb{R}^{m \times m}$ is a learnable matrix and $\tanh(\cdot)$ is the hyperbolic tangent function. Subsequent LSTM hidden states are computed as

$$\mathbf{d}_{t,n} = g^{LM}(r^I(\mathbf{w}_{t,n-1}, \mathbf{h}_t), \mathbf{d}_{t,n-1}),$$

for $n = 1, \dots, N_t + 1$, where $r^I(\mathbf{w}_{t,n-1}, \mathbf{h}_t) = \mathbf{R}_w^I \mathbf{w}_{t,n-1} + \mathbf{R}_h^I \mathbf{h}_t$ is a linear transformation with learned parameters $\mathbf{R}_w^I \in \mathbb{R}^{m \times d}$ and $\mathbf{R}_h^I \in \mathbb{R}^{m \times m}$, $g^{LM}(\cdot, \cdot)$ is a forward LSTM function, and $\mathbf{w}_{t,0}$ is the word embedding for the start-of-sentence token. By injecting the speaker state vector at every time step n in the turn t, the model is more likely to favor directly using the speaker state vector (vs. the word history) for predicting the speaker language. The conditional probability is then computed as

$$\Pr(w_{t,n} | w_{t,<n}) = \text{softmax}(\mathbf{V} r^O(\mathbf{h}_t, \mathbf{d}_{t,n})), \quad (2)$$

where $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times m}$ is the weight matrix, and $r^O(\mathbf{h}_t, \mathbf{d}_{t,n}) = \mathbf{R}_h^O \mathbf{h}_t + \mathbf{R}_d^O \mathbf{d}_{t,n}$ is another linear function with learnable parameters $\mathbf{R}_h^O, \mathbf{R}_d^O \in \mathbb{R}^{m \times m}$. The last word w_{t,N_t+1} is always the end-of-sentence token.

2.4 Model Training and Tuning

The training objective for a given conversation is the log-likelihood $\sum_t \sum_n \log \Pr(w_{t,n} | w_{t,<n})$, where the conditional probability is defined in (2). The Adam optimizer (Kingma and Ba, 2015) is used with a configuration of $\beta_1 = 0.9$ and $\beta_2 = 0.97$. The initial learning rate is set to 0.002. We halve the learning rate at each epoch once the development log-likelihood decreases, and terminate the training when it decreases for the second time. This validation protocol is used throughout the paper for training the proposed model.

In our experiments, the embedding dictionary $\mathbf{W}$ is initialized using pre-trained 300-dimensional word embeddings (Bojanowski et al., 2017) for words within the vocabulary of this resource. The remaining part of $\mathbf{W}$ and other model parameters are randomly initialized based on $\mathcal{N}(0, 0.01)$. The mode vector dimension c is set to 64. We tune the number of global mode vectors K from $\{16, 32\}$ and the hidden layer size m from $\{128, 160\}$. The final model is selected based on the log-likelihood on the development set.

3 User Topic Decision Prediction

We first study a prediction task that estimates whether the user engaged in a socialbot conversation would accept a suggested topic. Specifically, we use a corpus of human-socialbot conversations collected during the 2017 Alexa Prize competition (Ram et al., 2017) from the Sounding Board system (Fang et al., 2018; Fang, 2019). Due to privacy concerns, the socialbot does not have access to any identity information about users. Also, since each device may be used by multiple users, the device address is not a reliable indicator of the user ID. Therefore, the ability to profile the user through one conversational interaction is desirable for guiding the socialbot's dialog policy.

	train	dev	test
# conversations	19,076	6,321	6,465
# topic decisions	85,340	28,060	29,561

Table 1: Data statistics of the topic decision dataset.

3.1 Data

Each conversation begins with a greeting and ends when the user makes a stop command. The socialbot engages the user in the conversation using a wide range of content indexed by topics, where a topic corresponds to a noun or noun phrase that refers to a named entity (e.g., Google) or a concept (e.g., artificial intelligence). These topics are extracted using both constituency parsing results of the textual content and content meta-information. During the conversation, the socialbot sometimes negotiates the topic with the user using an explicit confirmation turn and records the user's binary decision (accept or reject) on the topic.

In socialbot conversations, a system turn is always followed by a user turn and vice versa. We tag system turns making explicit confirmation about a topic and attach the corresponding binary user decisions with them. To curate the dataset for the topic decision prediction task, we use a total of 31,862 conversations with more than 5 user turns. On average there are around 22 user turns per conversation. Not every system turn makes a topic suggestion, and the average number of topic decisions per conversation is 4.5. We randomly split the conversations into training, development, and test sets by $3/1/1$. The data statistics are shown in Table 1. In our experiments, we directly use the speech recognition output of user utterances. The vocabulary $\mathcal{V}$ consists of roughly 11K words that appear at least 5 times in the training set.

3.2 Topic Decision Classifier

We use a feed-forward neural network (FFNN) to make binary predictions (accept vs. reject) for individual topic suggestions. For each topic suggestion, the FFNN takes two inputs: 1) an embedding $\mathbf{x}_{t'}$ for the suggested topic at system turn t', and 2) a user embedding vector $\mathbf{z}_t$ at user turn t. Note the model does not have information about user turns after the system turn t' when making the prediction, i.e., the user turn t appears before the system turn t'.

The topic embedding $\mathbf{x}_{t'}$'s are looked up from the embedding dictionary learned by the FFNN. They are initialized by averaging the embeddings of their component words using the public pretrained 300-dimensional word embeddings (Bojanowski et al., 2017).

For the user embedding vector, we explore two settings that use different numbers of user turns as context. In both settings, topic decisions occurring in the first 5 user turns are not used for evaluations.

Static User Embeddings: Motivated by the findings that most user characteristics can be inferred from initial interactions (Ravichander and Black, 2018), we derive a static user embedding vector for a conversation using the first 5 user turns and apply it for predicting topic decisions afterwards.

Dynamic User Embeddings: Alternatively, we build a user embedding vector for user turn t using all previous user turns. Here, a topic decision for system turn t' is aligned with its preceding user turn t.

In our experiments, we compare different unsupervised models with our proposed dynamic speaker model. For both settings, all unsupervised models are pre-trained on *all* user turns in training conversations. They are fixed when training the FFNN classifier. The FFNN classifier is trained with the logistic loss using the Adam optimizer (Kingma and Ba, 2015). The training protocol is similar to that described in §2.4. We tune the hidden layer size from $\{64, 128\}$ and the number of hidden layers from $\{0, 1\}$. The model is selected based on the loss on the development set.

In addition, we use a user-agnostic **TopicPrior** baseline. It builds a probability lookup for each topic using its acceptance rate on the training set. We tune a universal probability threshold for all topics based on the development set accuracy.

In all experiments, three evaluation metrics are used: accuracy, area under the receiver operating characteristic curve (AUC), and normalized cross-entropy (N-CE). N-CE is computed as the relative cross-entropy reduction of the model over the TopicPrior baseline.

3.3 Experiments: Static User Embeddings

As described in §3.2, we use the first 5 user turns to derive the user embedding vector for a conversation. We compare our dynamic speaker model with three other unsupervised models.

DynamicSpeakerModel: For the proposed dynamic speaker model, we concatenate the speaker state vector $\mathbf{h}_t$ and the local speaker mode vector

Model	Acc	AUC	N-CE
TopicPrior	68.8	72.5	0
UtteranceLDA	68.8	73.1	12.6
UtteranceAE	68.8	73.4	12.8
TopicDecisionEncoder	68.9	73.8	13.4
DynamicSpeakerModel	**69.5**	**74.2**	**13.7**

Table 2: Test set results (in %) for topic decision predictions using *static* user embeddings.

$\tilde{\mathbf{u}}_t$ for each of the first 5 user turns. Then, we apply the max-pooling operation over the 5 concatenated vectors to summarize all the information. The resulting vector $\tilde{\mathbf{h}}$ is used as the user embedding vector.

UtteranceLDA: The latent Dirichlet allocation (LDA) model (Blei et al., 2003) is trained with 16 latent groups by treating all user utterances in a conversation as a document.[2] The trained LDA model builds a 16-dimensional probability vector as the user embedding vector by loading the first 5 user turns as a single document.

UtteranceAE: The utterance auto-encoder model is built upon the sequence auto-encoder (Dai and Le, 2015). We replace the original encoder by a BiLSTM that encodes the utterance at user turn t into a summary vector $\mathbf{s}_t$ in the same way as the first stage of the latent mode analyzer described in §2.1. The auto-encoder is trained on all user utterances in the training data, using the same training protocol described in §2.4. We set the hidden layer size to 128. The user embedding vector is constructed by applying the max-pooling operation over the summary vectors $\mathbf{s}_1, \dots, \mathbf{s}_5$ for the first 5 user turns.

TopicDecisionEncoder: This model encodes the topic decisions occurred in the first 5 user turns. The user embedding vector is the concatenation of two vectors. One is max-pooled from the topic embeddings for accepted topics, and the other for rejected topics, both include a dummy topic vector as default. The topic embeddings are composed by averaging the public pre-trained 300-dimensional embeddings (Bojanowski et al., 2017) for words in the topic.

Experiment results are summarized in Table 2. The TopicPrior is a very strong predictor, with an accuracy on par with other user embeddings. This indicates that the popularity-based approach is a good start for content ranking in socialbots when there is little user information. Nevertheless, we can still observe some improvement over the TopicPrior in terms of AUC and N-CE, which suggests using information from initial interactions reduces the uncertainty of predictions. The proposed dynamic speaker model performs the best among the compared models, reducing the cross-entropy by 13.7% over the TopicPrior baseline.

Model	Acc	AUC	N-CE
TopicDecisionLSTM	69.3	74.8	14.6
UtteranceAE + LSTM	69.9	75.4	15.3
DynamicSpeakerModel	**72.4**	**79.0**	**20.0***

Table 3: Test set results (in %) for topic decision predictions using *dynamic* user embeddings. *: The improvement of DynamicSpeakerModel over both TopicDecisionLSTM and UtteranceAE + LSTM is statistically significant based on both t-test and McNemar's test ($p < .001$).

3.4 Experiments: Dynamic User Embeddings

Here, we use all information accumulated before the system turn of suggesting the topic to build the corresponding user embedding vector. Since the UtteranceLDA is not as effective based on static embedding experiments, we only consider extending UtteranceAE and TopicDecisionEncoder models for comparison here.

DynamicSpeakerModel: The speaker state tracker in our model accumulates the user information as the conversation evolves. Thus, we directly concatenate the speaker state vector $\mathbf{h}_t$ and the local speaker mode vector $\tilde{\mathbf{u}}_t$ as the user embedding vector at user turn t. Other than using more turns, this is the same DynamicSpeakerModel configuration as in §3.3.

UtteranceAE+LSTM: This model uses an LSTM to encode the summary vector sequence derived from the same utterance auto-encoder used in §3.3. The LSTM hidden states are treated as user embedding vectors at individual user turns.

TopicDecisionLSTM: Similarly, an LSTM is used to encode the topic decision sequence. At each time step, the LSTM reads the concatenation of the topic embedding and the one-hot vector encoding the topic decision. We use the same topic embeddings as the TopicDecisionEncoder in §3.3. Since not every user turn is associated with a topic

---

[2] To allow the LDA model to take into account bi-grams, we replace the uni-gram token w_i with its bi-gram (w_i, w_{i+1}) concatenated as a single token if the bi-gram is among the top 500 frequent bi-grams.

decision, the time steps of this LSTM are aligned to a sequence of non-consecutive user turns. The LSTM hidden states are treated as user embedding vectors at corresponding user turns.

For UtteranceAE+LSTM and TopicDecisionLSTM, the hidden layer size of the LSTM is set to 128. While the utterance auto-encoder and topic embeddings are pre-trained, the LSTM components are jointly learned with the FFNN for composing dynamic user embeddings.

Experiment results are shown in Table 3. The DynamicSpeakerModel performs the best. Comparing to results in Table 2, all three unsupervised models outperform their static counterparts, which suggests the advantage of using dynamic context for predicting user topic decisions as conversation evolves.

Statistical significance tests of the difference in performance of two systems were conducted under both the t-test using the predicted probabilities and McNemar's test using the binary predictions. Under both tests, the predictions from the TopicDecisionLSTM and the DynamicSpeakerModel are highly signification ($p < .001$). Predictions from UtteranceAE + LSTM and DynamicSpeakerModel are also significantly different based on both tests ($p < .001$).

3.5 Qualitative Analysis

First, we manually inspect the predictions from the TopicDecisionLSTM and DynamicSpeakerModel used in §3.4 and the static baseline TopicPrior in §3.3. Compared with TopicPrior, we find that TopicDecisionLSTM is able to utilize the semantic relatedness between neighboring topics and corresponding user decisions. For example, "Elon Musk" (the CEO) is likely to be rejected if "Tesla" (the company) has been rejected earlier, though both are popular topics with high acceptance rates. In addition, it seems that the DynamicSpeakerModel is able to make use of user reactions. In the anecdotal example illustrated in Table 4, the user accepts the topic "Arnold Schwarzenegger" which is correctly predicted by both TopicDecisionLSTM and DynamicSpeakerModel, but only the DynamicSpeakerModel correctly predicts the rejection of "politics" later.

We then analyze what language features are learned by latent modes in our dynamic speaker model. For each mode, we extract top utterances sorted by their association scores as computed in (1). Examples from the most representative modes are provided in Appendix A. In brief, we find two separate modes related to positive and negative reactions; other modes correspond to classes of dialog acts, such as yes/no answers, topic requests and conversation-closing. Within topic request modes, some involve short topic phrases (e.g., "*holidays*") while others use complete requests (e.g. "*can we talk about cats*"). Along this line, some modes are associated with relatively terse users and others with talkative users. These findings indicate that our model cpatures various user characteristics that might be useful for predicting their interaction preferences.

Bot: Do you like the actor Arnold Schwarzenegger?
User: yeah before he got into politics
Bot: Super, would you like to know a fun fact about Arnold Schwarzenegger?
    - **TopicDecisionLSTM**: accept
    - **DynamicSpeakerModel**: accept
User: why not sure
...
Bot: I'm running out of things to say about him. Do you wanna hear some news about politics?
    - **TopicDecisionLSTM**: accept
    - **DynamicSpeakerModel**: reject
User: no

Table 4: A dialog snippet showing topic decision predictions from TopicDecisionLSTM and DynamicSpeakerModel. Topics are shown with underscores.

4 Dialog Act Classification

Dialog act analysis is widely used for conversations, which identifies the illocutionary force of a speaker's utterance following the speech act theory (Austin, 1975; Searle, 1969). In this section, we apply the proposed dynamic speaker model to the dialog act classification task.

4.1 Data

We use the Switchboard Dialog Act Corpus (SwDA), which has dialog act annotations on two-party human-human speech conversations (Jurafsky et al., 1997; Stolcke et al., 2000). In total, there are 1155 open-domain conversations with manual transcripts. Following recent work, we use 1115 conversations for training, 19 for testing, and the rest 21 for development.[3] The original fine-grained dialog act labels are mapped to 42

---

[3] The training and test split files are downloaded from `https://web.stanford.edu/~jurafsky/ws97/`.

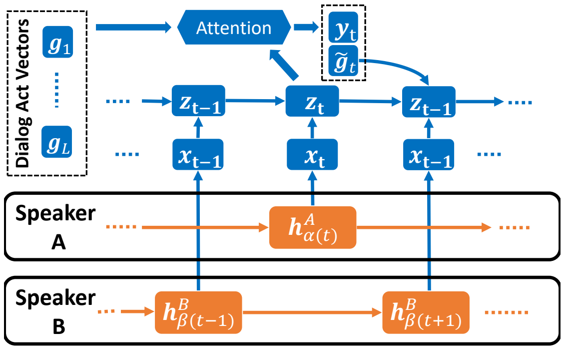


Figure 2: The attention-based LSTM tagging model for dialog act classification. The figure only shows the attention operation for turn t. The lower two boxes represent two speaker state trackers.

classes.[4] For this set of experiments, we use the golden segmentation and manual transcripts provided in the dataset.

Motivated by the recent success of unsupervised models (Peters et al., 2018; Devlin et al., 2019), we also study whether the dynamic speaker model can benefit from training on external unlabelled data. Thus, we use speech transcripts from 5850 conversations from the Fisher English Training Speech Part 1 Transcripts (Cieri et al., 2004), which (like Switchboard) consists of two-party human-to-human telephone conversations but without annotations for dialog acts.

4.2 Dialog Act Tagging Model

We use an attention-based LSTM tagging model for the dialog act classification. As shown in Fig. 2, the tagging LSTM is stacked on two speaker state trackers. Note the two trackers share the same parameters as well as the underlying latent mode analyzer and speaker language predictor. They generate speaker embeddings by tracking corresponding speakers separately.

Let $\alpha(t)$ and $\beta(t)$ denote the mappings from the global turn index t to the speaker-specific turn indices for speaker A and speaker B, respectively. The mapping returns a null value if the turn t is not associated with the corresponding speaker. The speaker state vectors are used as the input to the tagging LSTM for corresponding turns, i.e., $\mathbf{x}_t = I(\mathbf{h}^A_{\alpha(t)}, \mathbf{h}^B_{\beta(t)})$ where $I(\cdot,\cdot)$ is a switcher that chooses $\mathbf{h}^A_{\alpha(t)}$ or $\mathbf{h}^B_{\beta(t)}$ depending on whether $\alpha(t)$ and $\beta(t)$ return a non-null value.

The tagging LSTM also maintains a dictionary of L dialog act vectors $\mathbf{g}_1, \ldots, \mathbf{g}_L$. The dialog act probabilities $\mathbf{y}_t \in \mathbb{R}^L$ at turn t are computed using the dot-product attention mechanism, i.e., $\mathbf{y}_t = f(\mathbf{z}_t, [\mathbf{g}_1, \ldots, \mathbf{g}_L])$, where $f(\cdot,\cdot)$ is defined as in (1), and $\mathbf{z}_t$ is the hidden state vector of the LSTM.

The tagging LSTM computes hidden states as

$$\mathbf{z}_{t+1} = g^{DA}\left(r^{DA}(\tilde{\mathbf{g}}_t, \mathbf{x}_{t+1}), \mathbf{z}_t\right)$$

where $\tilde{\mathbf{g}}_t = \sum_{l=1}^{L} y_{t,l}\mathbf{g}_l$, $g^{DA}(\cdot,\cdot)$ is the LSTM function, and $r^{DA}(\cdot,\cdot)$ is a linear function with learnable parameters. In this way, both the history dialog act predictions and the utterance information are encoded in the hidden states.

The training objective of the tagging LSTM is the sum of the cross-entropy between the dialog act label and the probabilities $\mathbf{y}_t$ at each turn. The training configuration is the same as the topic decision classifier described in §3.2. We tune the size of hidden states $\mathbf{z}_t$ and dialog act embeddings $\mathbf{g}_l$ from $\{64, 128\}$ with arbitrary combinations, and vary the number of LSTM hidden layers from $\{1, 2\}$. The best model is selected according to the development set accuracy.

4.3 Experiment Results

In our experiments, we compare three settings for using the dynamic speaker model. In the **pre-train** setting, the dynamic speaker model is trained on the SwDA data without the dialog act labels. We then freeze the model when training the tagging LSTM. In contrast, in the **pretrain + fine-tune** setting, the dynamic speaker model is fine-tuned together with the tagging LSTM. Finally, in the **pre-train w/Fisher + fine-tune** setting, the dynamic speaker model is pre-trained on the combination of SwDA and Fisher datasets, and then fine-tuned together with the tagging LSTM on the SwDA dataset. For all three settings, we use the same vocabulary $\mathcal{V}$ of size 21K which combines all tokens from the SwDA training set and those appearing at least 5 time in the Fisher corpus.

We compare our results to best published results. In (Kalchbrenner and Blunsom, 2013), a convolutional neural network (CNN) is used to encode utterances. A recurrent neural network (RNN) is then applied on top of the CNN to encode both utterances and speaker label information for predicting the dialog acts. Ji et al. (2016) propose a discourse-aware RNN LM by treating

---

[4] Dialog act labels are mapped using scripts from `http://compprag.christopherpotts.net/swda.html`. Utterances labelled as "segment" are merged with corresponding previous utterance by the same speaker.

Model	Acc (%)
(Kalchbrenner and Blunsom, 2013)	73.9
(Tran et al., 2017a)	74.2
(Tran et al., 2017b)	74.5
(Tran et al., 2017c)	75.6
(Ji et al., 2016)	77.0
pre-train	75.6
pre-train + fine-tune	**77.2**
pre-train w/ Fisher + fine-tune	**78.6***

Table 5: Test set accuracy for SwDA dialog act classification. *: The improvement of pre-train w/ Fisher + fine-tune is statistically significant over pre-train + fine-tune based on McNemar's test ($p < .001$).

the dialog act as a conditional variable to the LM. Tran et al. (2017a,b,c) focus on building hierarchies of RNNs to model the dialog context using previous utterances or dialog act predictions. Results from (Lee and Dernoncourt, 2016) and (Liu et al., 2017) are not directly comparable due to different experiment settings.

Experiment results are summarized in Table 5. Our pre-train setting performs on par with previous state-of-the-art supervised models except (Ji et al., 2016). Fine-tuning significantly improves the performance and allows the model to achieve a similar accuracy as (Ji et al., 2016). The best result is achieved by pre-training the dynamic speaker model with both SwDA and Fisher datasets. The improvement of pre-train w/ Fisher + fine-tune is statistically significant over pre-train + fine-tune based on McNemar's test ($p < .001$). This illustrates the advantage of the unsupervised learning approach for the proposed model as it can exploit a large amount of unlabelled data.

4.4 Qualitative Analysis

We analyze the latent modes learned on SwDA using the same approach as in §3.5. Again, specific examples are included in Appendix A. Overall, there are several modes corresponding to coarse-grained dialog acts, such as statements, questions, agreement, backchannel and conversation-closing. Many modes characterize statements, probably due to their high relative frequency in the corpus. Among the statement modes, there are two distinct groups, one containing multiple filled pauses, such as *uh, you know, well*, and the other one with *because*-clauses. The fact that coarse-grained dialog act information is partly encoded in the modes may be helping with recognizing the dialog act.

In addition, we use the speaker gender information available in the SwDA data to determine whether the latent modes in the dynamic speaker model pick up gender-related language variation. Specific examples and statistics are included in Appendix B. The Cohen-d score (Cohen, 1988) is used to measure the strength of the difference between association score distributions of male vs. female utterances for individual modes. Based on the Cohen-d score, we identified two modes that have a strong association with male speakers, and two with female speakers. All have significantly different ($p < 0.001$) distributions of association scores for female vs. male speakers using Mann-Whitney U test. In the top associated utterances for the male modes, we find utterances with several filled pauses, which has been found to be indicative of male speakers in previous work on Switchboard (Boulis and Ostendorf, 2005). The female modes are mostly agreement, acknowledgement and backchannel, which aligns with a popular sociolinguistic theory that females are more responsive (Coates, 1998). Based on this, we conclude that some speaker gender language variations are indeed captured by the learned modes.

5 Related Work

As reviewed by Zukerman and Litman (2001), user modeling for conversational systems has a long history. The research can be tracked back to the GRUNDY system (Rich, 1979) which categorizes users in terms of hand-crafted sets of user properties for book recommendation. Other systems have focused on different aspects of users, e.g., the expertise level of the user in a specific domain (Chin, 1986; Sleeman, 1985; Paris, 1987; Hovy, 1987), the user's intent and plan (Allen and Perrault, 1980; Carberry, 1983; Litman, 1986; Moore and Paris, 1992), and the user's personality (Mairesse and Walker, 2006; DeVault et al., 2014; Fung et al., 2016; Fang et al., 2017). User modeling has also been employed for personalized topic suggestion in recent Alexa Prize socialbots, using a pre-defined mapping between personality types and topics (Fang et al., 2017), or a conditional random field sequence model with hand-crafted user and context features (Ahmadvand et al., 2018). Modeling speakers with continuous embeddings for neural conversation models is studied in (Li et al., 2016), where the model directly learns a dictionary of speaker embeddings. Our unsupervised dynamic speaker model differs from previous work in that we build speaker embeddings as a weighted combination of latent modes with weights computed based on the utterance. Thus, the model can construct embeddings for any new users and dynamically update the embeddings as the conversation evolves.

Speaker language variances have been analyzed by previous work and incorporated in NLP models. Preoţiuc-Pietro et al. (2016) and Johannsen et al. (2015) find that speaker-level language variance affects lexical choices and even syntactic structure based on psycholinguistic hypotheses. Speaker demographics are used to improve both low-level tasks such as part-of-speech tagging (Hovy and Søgaard, 2015) and high-level applications such as sentiment analysis (Volkova et al., 2013) and machine translation (Mirkin et al., 2015). Lynn et al. (2017) introduce a continuous adaptation method to include user age, gender, personality traits and language features for personalizing several supervised NLP models. Different from previous work, we study the use of speaker embeddings learned from utterances in an unsupervised fashion and analyze the possible interpretability of the latent modes.

6 Conclusion

In this paper, we address the problem of modeling speakers from their language using an unsupervised approach. A dynamic speaker model is proposed to learn speaker embeddings that are updated as the conversation evolves. The model achieves promising results on two representative tasks in dialogs: user topic decision prediction in human-socialbot conversations and dialog act classification in human-human conversations. In particular, we demonstrate that the model can benefit from unlabelled data in the dialog act classification task, where we achieve the state-of-the-art results. Finally, we carry out analysis on the learned latent modes on both tasks, and find cues that suggest the model captures speaker characteristics such as intent, speaking style, and gender. For future work, it could be interesting to explore guiding some latent modes with a few examples to pick up specific user features such as personality traits.

Acknowledgements

We thank the anonymous reviewers for their helpful feedback. We also thank Trang Tran for her feedback on the paper. This research is supported by Amazon Alexa Fellowship and Tencent AI Lab Rhino-Bird Gift Fund. The conclusions and findings are those of the authors and do not necessarily reflect the views of sponsors.

References

Deepak Agarwal, Bee-Chung Chen, and Bo Pang. 2011. Personalized recommendation of user comments via factor models. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Ali Ahmadvand, Ingyu Choi, Harshita Sahijwani, Justus Schmidt, Mingyang Sun, Sergey Volokhin, Zihao Wang, and Eugene Agichtein. 2018. Emory IrisBot: An open-domain conversational bot for personalized information access. In *Proc. Alexa Prize 2018*.

James F. Allen and C. Raymond Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178.

John L. Austin. 1975. *How To Do Things with Words*, 2nd edition. Harvard University Press, Cambridge, MA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Machine Learning Research*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Constantinos Boulis and Mari Ostendorf. 2005. A quantitative analysis of lexical differences between genders in telephone conversations. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 435–442.

Kevin K. Bowden, Jiaqi Wu, Wen Cui, Juraj Juraska, Vrindavan Harrison, Brian Schwarzmann, Nick Santer, and Marilyn Walker. 2018. SlugBot: Developing a computational model and framework of a novel dialogue genre. In *Proc. Alexa Prize 2018*.

Sandra Carberry. 1983. Tracking user goals in an information-seeking environment. In *Proc. AAAI Conf. Artificial Intelligence*, pages 59–63.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2017. A factored neural network model for characterizing online discussions in vector space. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2296–2306.

David N. Chin. 1986. User modeling in UC, the UNIX consultant. In *Proc. Computer Human Interactions (CHI)*, pages 24–28.

Christopher Cieri, David Graff, Owen Kimball, Dave Miller, and Kevin Walker. 2004. Fisher english training speech part 1 transcripts LDC2004T19. Web Download.

Jennifer Coates, editor. 1998. *Language and gender: a reader*. Wiley-Blackwell.

Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 3079–3087.

David DeVault, Ron Artstein, Grace Benn, Teresa Dey, Ed Fast, Alesia Gainer, Kallirroi Georgila, Jon Gratch, Arno Hartholt, Margaux Lhommet, et al. 2014. SimSensei kiosk: A virtual human interviewer for healthcare decision support. In *Proc. Int. Conf. Autonomous Agents and Multi-agent Systems*, pages 1061–1068.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*.

Hao Fang. 2019. *Building a User-Centric and Content-Driven Socialbot*. Ph.D. thesis, University of Washington.

Hao Fang, Hao Cheng, Elizabeth Clark, Ariel Holtzman, Maarten Sap, Mari Ostendorf, Yejin Choi, and Noah Smith. 2017. Sounding Board – University of Washington's Alexa Prize submission. In *Proc. Alexa Prize*.

Hao Fang, Hao Cheng, Maarten Sap, Elizabeth Clark, Ariel Holtzman, Yejin Choi, Noah Smith, and Mari Ostendorf. 2018. Sounding Board – a user-centric and content-driven social chatbot. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL) (System Demonstrations)*.

Pascale Fung, Anik Dey, Farhad Bin Siddique, Ruixi Lin, Yang Yang, Yan Wan, and Ricky Ho Yin Chan. 2016. Zara the supergirl: An empathetic personality recognition system. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL) (System Demonstrations)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 483–488.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11:689–710.

Aaron Jaech and Mari Ostendorf. 2018. Personalized language model for query auto-completion. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 700–705.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 332–342.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proc. Conf. Computational Natural Language Learning (CoNLL)*, pages 103–112.

Dan Jurafsky, Elizabeth Shriberg, , and Debra Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado, Boulder.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 3294–3302.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 515–520.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016. A persona-based neural conversation model. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 994–1003.

Diane J. Litman. 1986. Linguistic coherence: a plan-based alternative. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 215–223.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using context information for dialog act classification in dnn framework. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2170–2178.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2017. Human centered nlp with user-factor adaptation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1146–1155.

François Mairesse and Marilyn Walker. 2006. Automatic recognition of personality in conversation. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 85–88.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 3111–3119.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1102–1108.

Johanna D. Moore and Cecile Paris. 1992. Exploiting user feedback to compensate for the unreliability of user models. *User Modeling and User-Adapted Interaction*, 2:287–330.

Cécile L. Paris. 1987. *The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise*. Ph.D. thesis, Columbia University.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*, pages 2227–2237.

Daniel Preoţiuc-Pietro, Wei Xu, and Lyle Ungar. 2016. Discovering user attribute stylistic differences via paraphrasing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3030–3037.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2017. Conversational AI: The science behind the alexa prize. In *Proc. Alexa Prize 2017*.

Abhilasha Ravichander and Alan Black. 2018. An empirical study of self-disclosure in spoken dialogue systems. In *Proc. SIGdial Meeting Discourse and Dialogue*, pages 253–263.

Elaine Rich. 1979. User modeling via stereotypes. *Cognitive Science*, 3:329–354.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Milad Shokouhi. 2013. Learning to personalize query auto-completion. In *SIGIR*, pages 103–112. ACM.

Derek Sleeman. 1985. UMFE: A user modelling front-end subsystem. *Int. J. Man-Machine Studies*, 23:71–88.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Quan Hung Tran, Gholamreza Haffari, and Ingrid Zukerman. 2017a. A generative attentional neural network model for dialogue act classification. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 524–529.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2017b. A hierarchical neural model for learning sequences of dialogue acts. In *Proc. European Chapter Assoc. for Computational Linguistics (EACL)*, pages 428–437.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2017c. Preserving distributional information in dialogue act classification. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2151–2156.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 5998–6008.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1815–1827.

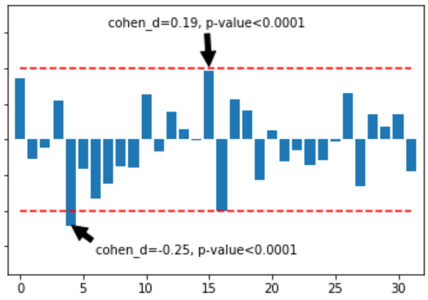
Ingrid Zukerman and Diane Litman. 2001. Natural language processing and user modeling. *User Modeling and User-Adapted Interaction*, 11:129–158.

A Examples for Mode Analysis

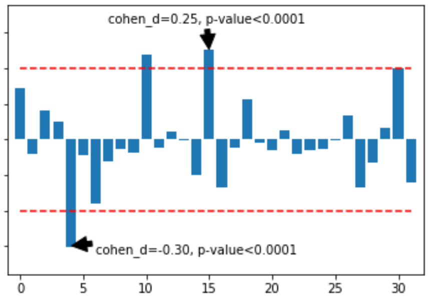
For each mode, we list top associated user utterances in Table 6 and Table 7 for the user topic decision corpus and SwDA corpus, respectively.

For modes learned in the user topic decision corpus, mode 4 seems to include positive reactions, while mode 2 involves slightly negative reactions. Modes 0 and 6 are mostly yes/no answers. Utterances associated with mode 3 are mostly conversation ending. Modes 9, 14, and 10 are mostly set topic commands, differing in style. Mode 10 is associated with complete requests (e.g., "*let's/can we talk about cats*)," while mode 9 and Mode 14 involve short topic phrases (e.g., "*holidays*"). Modes 8 and 11 capture talkative users, whereas modes 1 and 7 capture relatively terse users.

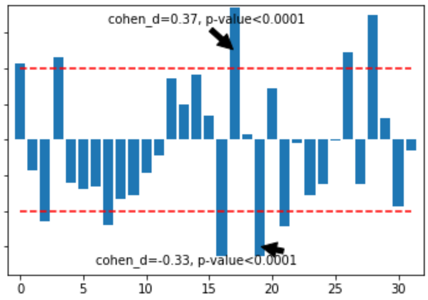
For latent modes learned in the SwDA corpus, there are several modes corresponding to coarse-grained dialog acts, such as statements (modes 2, 4, 6, 16, 19), questions (modes 8, 9), agreement (modes 12, 20), backchannel (modes 0, 28), and conversation-closing (mode 13). Among the statement modes, there are two distinct groups, one (modes 4, 6, 16, 19) containing multiple filled pauses, such as *uh, you know, well*, and the other one (mode 2) with *because*-clauses. The fact that coarse-grained dialog act information is partly encoded in the modes may be helping with recognizing the dialog act.



(a) all conversations



(b) same-gender conversations



(c) cross-gender conversations

Figure 3: Cohen-d scores for gender group tests. The x-axis is the mode index. The y-axis is the Cohen-d score, with a larger magnitude suggesting a large effect size, and a positive value for a more female-like mode. The red dash lines indicate the ±0.20 threshold.

B Speaker Gender Analysis

We use the speaker gender information from the SwDA data and analyze whether latent modes unsupervisedly learned in the dynamic speaker model could pick up some gender language variations. First, we gather the latent mode association scores for each of the 32 modes for all utterances as computed in (1). Then we carry out the group mean tests for individual modes to test the associate score distributions of male vs female utterances. The Cohen-d score is used to measure the strength of the difference (Cohen, 1988). We also compute the p-value using the Mann-Whitney U test. Previous work has observed larger gender language differences when the two speakers have the same gender (Boulis and Ostendorf, 2005). Thus, we carry out the group mean tests on the following three sets: 1) all conversations, 2) conversations involving only males or females, and 3) conversations involving both genders. The Cohen-d scores for overall, same-gender and cross-gender conversations are shown in Fig. 3. For each set, we identify the most female-like mode (with the most positive Cohen-d score) and the most male-like mode (with the most negative Cohen-d score). For female-like modes, modes 15 and 17 are identified in this way, whereas modes 4 and 19 are identified for male-like modes. By examining representative patterns in modes 15 and 17, they are mostly backchannel, acknowledgement, or agreement. For modes 4 and 19, filled pauses are prevalent.

Mode-0	• no no no no no no go back to my alexa . . . • no no no no let's stop talking now goodbye . . . • no let's chat let's chat about donald trump . . .
Mode-1	• gotcha • hiya • possibly
Mode-2	• serious • are you serious • that is a paradox
Mode-3	• alexa resume pandora • alexa connect bluetooth • no bye bye alexa
Mode-4	• that is fascinating • whoa • that that's cool
Mode-5	• i did not that's not surprising • i did not i did not knew that • unfortunately
Mode-6	• somewhat • yes yes yes yes yes • yes i did it was on the news
Mode-7	• mhm • ok • fascinating
Mode-8	• yes it was very much was i saw it i i was there i choose to the dark side did you choose that via uh right . . . • the online selanne jungle the mighty jungle the line the jungle in the jungle the mighty jungle the mighty jungle . . . • no if your life was narrated by someone and the choice was either • i was curious if you 'd rather have your life narrated by regis philbin or by morgan freeman • did you know the answer rogers because like a better go bike and probably i just do n't know it was just a long time ago • i thought bill murray was very very funny
Mode-9	• meow • award shows • celebrity
Mode-10	• no let's talk about butterflies • no let's talk about snakes • can we talk about kardashians
Mode-11	• is king kong real or is he bake but is he awesome or . . . • that is so true the concept of pencils are really stupid and should i even exist imagine if we have pencil do we wanna be able to write on paper so that makes you stupid • is this randomly talking to this is the dawning alligators okay so did we get bored i don't know you somehow or . . .
Mode-12	• do you know alexa how do you how do you know all the stuff you're an a. i. • what what alexa what how do you talk about • alexa do you know alexa do you know a joke today • alexa do you tell me what you know about the new vision nuclear plant
Mode-13	• ten million • thirty percent • what's p. r.
Mode-14	• dog • dogs • tv
Mode-15	• now • not now

Table 6: User utterances in socialbot conversations that have top association scores for individual latent modes.

Statements	Mode-2	• cause i know there 's one not too far from from me here in dallas • because they really had no idea NONVERBAL what was involved once i got home • because like i said i worked with a lot of those • because he left home at five thirty in the morning • and then she would like to turn in half of the parents that drop their kids off because of the condition the kids are in you know
	Mode-4	• uh some more in interest type topics in in other countries • uh the uh the credit union has got a deal now where you decide what you want • well it would be lower middle class housing here • uh the only other thing i have noticed though is that uh it seems that there 's been a lot of or more empha emphasis at least in what we 've been dealing with
	Mode-6	• and i know that uh you know it can be freezing cold in the wintertime and hot and uh sticky in the summertime • it 's uh it 's uh it 's uh plywood uh face i guess • but i NONVERBAL i i i think you know the biggest causes even then a lot of times are uh uh like when i was up in boston just all the cars you know just all over the place • and so i i it 's i think i to me i think uh something that 's going to help our medical uh arena is for um • you know it 's like it 's like a luxury car except that it 's the dodge aries NONVERBAL you know
	Mode-16	• but uh this last ski trip they took uh she had in contracted chicken pox first • but uh we lived in malaysia for t i in nineteen uh eighty one two three and four • well my uh my sister lives in houston • i i was only twenty five years old or something • it 's uh uh c n n has been a welcome addition to NONVERBAL the t v scene here in the last uh number of years
	Mode-19	• uh i traded off an eighty two oldsmobile for the eighty nine mazda • because i mean after i figured out i was getting eighty cents an hour i said bag it • uh we have a a mazda nine twenty nine and a ford crown victoria and a little two seater c r x. • and uh you know i i was amazed cause i 'd pick up a local paper and i 'd read about all of these you know really interesting things going on • well a friend of mine at work here said that he tried it with his dog
Backchannel	Mode-0	• yes • yes NONVERBAL
	Mode-15	• see • probably • like
	Mode-17	• uh • um
	Mode-18	• oh oh yeah • oh well • oh okay
	Mode-28	• uh huh NONVERBAL • uh huh NONVERBAL NONVERBAL • uh huh ery faint
Agreement	Mode-12	• exactly
	Mode-20	• yep ause • definitely • absolutely • i agree
Quesetion	Mode-8	• are you and your roommate a similar size • did you do the diagnosis or was it just an assumption that that 's probably the part that failed • or do you have powered you know a • NONVERBAL what kind of a car do you have now • did they know that all along
	Mode-9	• so what do you think about uh what do you think about what you see on t v about them like in the news or on the ads • what do you think about what do you think about the the lower grades you know k through seven • so uh what do you think about our involvement in the middle east • you are talking about p o w s or missing in actions
Conversation-closing	Mode-13	• bye • bye bye • appreciation talking to you

Table 7: Utterances for each mode in SwDA dataset.

Fluent Translations from Disfluent Speech in End-to-End Speech Translation

Elizabeth Salesky[1], **Matthias Sperber**[2], **and Alex Waibel**[1,2]
[1]Carnegie Mellon University, Pittsburgh PA, U.S.A.
[2]Karlsruhe Institute of Technology, Karlsruhe, Germany
esalesky@cs.cmu.edu

Abstract

Spoken language translation applications for speech suffer due to conversational speech phenomena, particularly the presence of disfluencies. With the rise of end-to-end speech translation models, processing steps such as disfluency removal that were previously an intermediate step between speech recognition and machine translation need to be incorporated into model architectures. We use a sequence-to-sequence model to translate from noisy, disfluent speech to fluent text with disfluencies removed using the recently collected 'copy-edited' references for the Fisher Spanish-English dataset. We are able to directly generate fluent translations and introduce considerations about how to evaluate success on this task. This work provides a baseline for a new task, the translation of conversational speech with joint removal of disfluencies.

1 Introduction & Related Work

Spoken language translation (SLT) applications suffer due to conversational speech phenomena, particularly the presence of disfluencies. In conversational speech, speakers often use disfluencies such as filler words, repetitions, false starts, and corrections which do not naturally occur in text and may not be desired in translation outputs. Disfluency recognition and removal has previously been performed as an intermediate step between speech recognition (ASR) and machine translation (MT), to make disfluent ASR output better-matched to typically clean machine translation training data (Cho et al., 2013, 2014; Wang et al., 2010; Honal and Schultz, 2005; Zayats et al., 2016). With the rise of end-to-end sequence-to-sequence speech translation systems (Weiss et al., 2017; Bansal et al., 2018), disfluency removal can no longer be handled as an intermediate step between ASR and MT but needs to be incorporated into the model or handled as a post-processing step.

Generating fluent translations from disfluent speech may be desired for simultaneous SLT applications where removing disfluencies will improve the application's clarity and usability. To train end-to-end speech translation requires parallel speech and text translations. This introduces data considerations not previously relevant with chained ASR+MT models, as different datasets could be used to train ASR and MT components. Where aligned speech and translations exist, data is typically clean speech→clean text, as in news or TED talks, or disfluent speech→disfluent translations, as in Fisher or meeting data, where disfluencies were faithfully included in the references for completeness. While some corpora with labeled disfluencies exist (Cho et al., 2014; Burger et al., 2002), only subsets have been translated and/or released. Salesky et al. (2018) introduced a set of fluent references[1] for Fisher Spanish-English, enabling a new task: end-to-end training and evaluation against fluent references.

Previous work on disfluency removal has treated it as a sequence labeling task using word or span-level labels. However, in some cases, simply removing disfluencies from an utterance can create ill-formed output. Further, corpora can have different translation and annotation schemes: for example for Fisher Spanish-English, translated using Mechanical Turk, Salesky et al. (2018) found 268 unique filler words due to spelling and casing. Disfluencies can also be context-specific, such as false starts or corrections where a phrase may be 'disfluent' due to its surroundings. To remove disfluencies as a post-processing step would require a separate model trained with appropriate data and disfluency labels, and may lead to ill-formed output. By translating directly to fluent target data instead, we aim to handle these concerns implicitly. We present the first results translating directly from disfluent source speech to fluent target text.

---

[1] Data available at: https://github.com/isl-mt/fluent-fisher

Proceedings of NAACL-HLT 2019, pages 2786–2792
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Data

For our experiments, we use Fisher Spanish speech (Graff et al.) and with two sets of English translations (Salesky et al., 2018; Post et al., 2013). The speech dataset comprises telephone conversations between mostly native Spanish speakers recorded in realistic noise conditions. The original English translations were collected through crowdsourcing, as described in Post et al. (2013). Four references were collected for each of the development and test sets, and one for training. The training data consists of 819 conversations yielding ∼160 hours of speech and 150k utterances; the development and test sets are ∼4k utterances each. We use only the first of the two development sets (dev, not dev2).

This data is conversational and disfluent. The original translations faithfully maintain and translate phenomena in the Spanish transcripts such as filler words and hesitations, discourse markers (*you know*, *well*, *mm*), repetitions, corrections and false starts, among others. Salesky et al. (2018) introduced a new set of fluent reference translations collected on Mechanical Turk. They collected two references for each of the development and test sets, and one for the training set. Rather than labeling the disfluencies in the original target data, Turkers were asked to rewrite the utterance in a 'copy-edited' manner without disfluent phenomena. In some cases, simply removing disfluencies would created ill-formed structure in the resulting utterance. This scheme instead creates a sentence-level edit allowing for reordering and insertions as necessary to create fluent content, akin instead to monolingual translation or paraphrasing. Examples of source transcripts and original translations with the fluent counterparts are shown below in Table 1.

SRC	eh, eh, eh, um, yo pienso que es así
ORG	uh, uh, uh, um, i think it's like that
FLT	i think it's like that
SRC	también tengo um eh estoy tomando una clase ..
ORG	i also have um eh i'm taking a marketing class ..
FLT	i'm also taking a marketing class
SRC	porque qué va, mja ya te acuerda que ..
ORG	because what is, mhm do you recall now that ..
FLT	do you recall now that ..
SRC	y entonces am es entonces la universidad donde yo estoy es university of pennsylvania
ORG	and so am and so the university where i am it's the university of pennsylvania
FLT	i am at the university of pennsylvania

Table 1: Disfluency examples in Spanish source (SRC), original (ORG) and fluent (FLT) English translations

3 Speech-to-Text Model

Initial work on the Fisher-Spanish dataset used HMM-GMM ASR models linked with phrase-based MT using lattices (Post et al., 2013; Kumar et al., 2014). More recently, it was shown in Weiss et al. (2017) and Bansal et al. (2018) that end-to-end SLT models perform competitively on this task. As in Bansal et al. (2018), we use a sequence-to-sequence architecture inspired by Weiss et al. but modified to train within available resources; specifically, all models may be trained in less than 5 days on one GPU. We build an encoder-decoder model with attention in `xnmt` (Neubig et al., 2018) with 512 hidden units throughout. We use a 3-layer BiLSTM encoder. We do not use the additional convolutional layers from Weiss et al. and Bansal et al. to reduce temporal resolution, but rather use network-in-network (NiN) projections from previous work in sequence-to-sequence ASR (Zhang et al., 2017; Sperber et al., 2018) to get the same total $4\times$ downsampling in time. This gives the benefit of added depth with fewer parameters. We closely follow the LSTM/NiN encoder used in Sperber et al. (2018) for ASR and use the same training procedure, detailed in Appendix A.

We extract 40-dimensional mel filterbank features with per-speaker mean and variance normalization with Kaldi (Povey et al., 2011). We did not see significant difference between 40, 40+deltas and 80-dimensional features in initial experiments, similar to Bansal et al. (2018), who chose 80-dim. Weiss et al. (2017) used 240-dim features comprising 80-dim filterbanks stacked with deltas and delta-deltas. We exclude utterances longer than 1500 frames to manage memory requirements.

Like Weiss et al. (2017), we translate to target characters as opposed to words (Bansal et al., 2018). We also use an MLP-based attention with 1 hidden layer with 128 units and 64-dimensional target embeddings, though we use only 1 decoder hidden layer as opposed to 3 or 4 in these works. We use input feeding (Luong et al., 2015). All models use the same preprocessing as previous work on this dataset: lowercasing and removing punctuation aside from apostrophes.

4 Experiments

4.1 Experimental Setup

We focus on the problem of translating directly from noisy speech to clean references without a separate disfluency removal step. We first demonstrate the efficacy of our models on the original disfluent Fisher Spanish-English task, comparing to the previously reported numbers on the SLT task (Weiss et al., 2017; Bansal et al., 2018). We then compare these results with models trained using the collected 'clean' target data with disfluencies removed. Finally, we look at the mismatched case where we train on disfluent data and evaluate on a cleaned test set; this is a more realistic scenario, as clean training data is difficult to collect, and we cannot expect to have it for each language and use case we encounter.

We evaluate using both BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) to compare different aspects of model behavior on our two tasks.[2] BLEU assesses how well predicted translations match a set of reference translations using modified n-gram precision, weighted by a brevity penalty in place of recall to penalize short hypothesis translations without full coverage. The brevity penalty is computed as $e^{(1-r/c)}$, where r is the length of the reference and c the candidate translation. For our task of implicitly removing disfluencies during translation, our generated translations should contain much of the same content but with certain tokens removed, creating shorter translations. When scoring *fluent output* against the *original disfluent references*, then, differences in BLEU score will come from two sources: shorter n-gram matches, and the brevity penalty. METEOR, on the other hand, can be considered a more 'semantic' evaluation metric. It uses a harmonic mean of precision and recall, with greater weight given to recall. Further, while BLEU uses exact n-gram matches, METEOR also takes into account stem, synonym, and paraphrase matches. For our fluent task, we aim to maintain semantic meaning while removing disfluent tokens. Accordingly, when scored against the fluent target references, we hope to see similar METEOR scores between the disfluent models and fluent models. Both metrics are used for a holistic view of the problem: METEOR will indicate if meaning is maintained, but not assess disfluency removal, while BLEU changes will indicate whether disfluencies have been removed.

We provide both multi-reference and single-reference BLEU and METEOR scores: the original Fisher target data has four reference translations for the dev and test sets, which boosts scores considerably as hypothesis n-grams can match in any of the references. The fluent target data has two references, so the single reference scores better enable comparison between the two tasks.

4.2 Results & Discussion

Table 2 shows our results on the original disfluent data with comparisons to Weiss et al. (2017) and Bansal et al. (2018). All results are single task end-to-end speech translation models. Weiss et al.'s deeper model reaches a BLEU score of 47.3 on `test` after 2.5 weeks of training. Our model is more similar in depth to Bansal et al. (2018), having both made modifications to train on one GPU in < 5 days (see Section 3). While Bansal et al. use words on the target side to improve convergence time at a slight performance cost, we are able to use characters like Weiss et al. by having a still shallower architecture (2 fewer layers on both the encoder and decoder), giving us approximately the same training time per epoch they observe with words ($\sim$2 hours). We converge to a test BLEU of 33.7, 3-4 BLEU improved over Bansal et al. on dev and test. This demonstrates our model has reasonable performance on the original data, providing a strong baseline before turning to our targeted task of directly generating fluent translations.

	Weiss et al.		Bansal et al.		Ours	
Metric	**dev**	**test**	**dev**	**test**	**dev**	**test**
BLEU 4Ref	46.5	47.3	29.5	29.4	32.4	33.7
BLEU 1Ref	–	–	–	–	19.0	19.6
METEOR 4Ref	36.5	–	28.2	–	30.0	30.9
METEOR 1Ref	–	–	–	–	25.1	26.1

Table 2: Single task end-to-end speech translation using **original disfluent references** to train and evaluate. Comparing multi-reference scores using all four references (4Ref) vs average single reference score (1Ref).

Table 3 compares performance of speech translation models trained with the fluent target translations to models trained with the original disfluent translations, as scored on the fluent references. Comparing the disfluent and fluent models, we see that METEOR scores are almost the same while BLEU scores are lower with the disfluent model. This is as we would hope: with our fluent model, we want to generate translations that are semantically the same but with disfluencies removed. Therefore similar METEOR scores with similar

---

[2] BLEU scores are 4-gram word-level BLEU computed using `multi-bleu.pl` from the Moses toolkit (Koehn et al., 2007). METEOR is computed using the script from http://www.cs.cmu.edu/~alavie/METEOR/

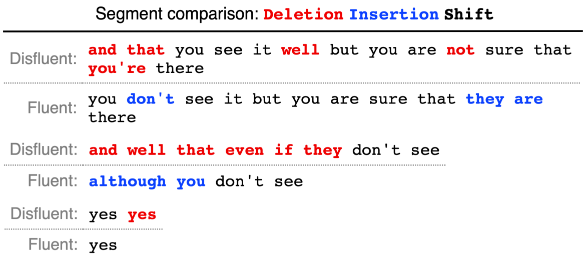
recall (52) on the fluent references are encouraging. For BLEU, however, the disfluencies generated by the disfluent model break up n-grams in the fluent references, thereby lowering scores.

Model	Metric	dev 1Ref	dev 2Ref	test 1Ref	test 2Ref
Disfluent	BLEU	13.0	16.2	13.5	17.0
Fluent	BLEU	14.6	18.1	14.6	18.1
Disfluent	METEOR	22.2	23.9	23.1	24.8
Fluent	METEOR	22.3	24.0	23.1	24.9

Table 3: End-to-end model performance evaluated with **new fluent references**. Comparing average single reference scores (1Ref) vs multi-reference scores using both generated references (2Ref).

Comparing single-reference scores with Table 2, we see that they are distinctly lower. This is to be expected with the shorter fluent references; a difference of a single token carries greater weight. Translating directly to the fluent references is a more challenging task. As shown in Table 1, the original English translations and Spanish speech are very one-to-one while the edited translations introduce deletions and reorderings. In learning to generate fluent translations, the model needs to learn to handle these more inconsistent behaviors.

Figure 1 shows a visual comparison between outputs generated by the two models. Using the fluent target data to train constrains the model output vocabulary, so filler words such as *'um', 'ah', 'mhm'* are not generated. We also see significant reductions in repetitions of both words and phrases from the model trained with fluent reference translations. Further, we also see instances where the fluent model generates a shorter paraphrase of a disfluent phrase, as in the 2nd example.



Segment comparison: **Deletion** **Insertion** **Shift**

Disfluent: **and** **that** you see it **well** but you are **not** sure that **you're** there

Fluent: you **don't** see it but you are sure that **they** **are** there

Disfluent: **and** **well** **that** **even** **if** **they** don't see

Fluent: **although** **you** don't see

Disfluent: yes **yes**

Fluent: yes

Figure 1: Comparison of example outputs generated by disfluent and fluent models, created with CharCut (Lardilleux and Lepage, 2017).

Disfluency removal for speech translation has traditionally been done as an intermediate step between ASR and MT to better-match additional clean corpora used for MT training; we do not compare to a pipeline approach here. However, to contextualize these results, we compare disfluency removal as a post-processing step after end-to-end speech translation with the original disfluent parallel data. Simply filtering filler words and repetitions from the disfluent model (Filter) outputs as a post-processing step, the dev scores improve slightly, but test stays the same or decreases. In some cases, treating disfluency removal as a filtering task can reduce the fluency of an utterance:

Model	dev 1Ref	dev 2Ref	test 1Ref	test 2Ref
Postproc. Filter	13.6	16.5	13.5	16.8
Postproc. MonoMT	14.4	17.8	14.4	18.0

Table 4: End-to-end disfluent model with different post-processing steps. Performance evaluated with **new fluent references**.

Disfluent	*mm well and from and the email is a scandal the spam.*
Fluent	*the email is a scandal it's spam.*

A filtering or tagging system may not capture all false starts or corrections, leading to lower fluency, and requires labeled spans. Treating the post-processing step as a monolingual translation task (MonoMT) rather than a filtering task allows for reordering and insertions, which we saw boost fluency. We trained a 4-layer BiLSTM encoder-decoder model to translate between the disfluent and fluent English references and applied this to the output of the end-to-end disfluent model. BLEU scores approach the results with the end-to-end fluent target model (Table 3), but we note, this requires the same resources as the direct task.

Showing the importance of fluent references for evaluation, Table 5 shows the performance of fluent models as evaluated on the original disfluent references. Disfluent target scores are the same as in Table 2, and have been copied for easy compar-

Model	Metric	dev 1Ref	dev 4Ref	test 1Ref	test 4Ref
Fluent	BLEU	16.6	29.8	17.0	30.4
Disfluent	BLEU	19.0	32.4	19.6	33.7
Fluent	METEOR	21.8	25.9	22.7	27.0
Disfluent	METEOR	25.1	30.0	26.1	30.9

Table 5: Performance evaluated with **original disfluent references**. Comparing average single reference scores (1Ref) vs multi-reference scores using all references (4Ref).

ison. As we would expect, here there is a greater difference in scores. The fluent references have fewer long n-gram matches with disfluencies removed, lowering BLEU. The fluent model's METEOR scores suffer more than BLEU due to the recall calculation; recall on the disfluent references is lower because the fluent model does not produce many of the disfluencies (indeed filler words are not in the vocabulary when trained with the fluent references). Recall is reduced by $\sim 14\%$ with the fluent model, reflecting the approximate distribution of disfluencies in the original data.

The differences in scores with these two metrics do not show the full picture. Outputs generated by the fluent model are on average 13% shorter and contain 1.5 fewer tokens per utterance than the disfluent model, which is significant with average utterance lengths of 10-11 tokens. When scoring the fluent output against the original disfluent references, the shorter length significantly contributes to the lower scores, with the BLEU brevity penalty calculated as 0.86 as opposed to 0.96-1.0 for all other conditions. Removing the length penalty from the BLEU score calculation, single-reference scores are boosted to 19.3 and 19.8 from 16.6 and 17.0 for dev and test, respectively (Table 5). This is a somewhat fairer comparison of the disfluent and fluent models, as we do not want the fluent output to match the disfluent sequence length, and the disfluent models are not penalized due to length. These BLEU scores are now very similar to those of the disfluent model on the disfluent references, though the outputs are very different (Figure 1). The changes here and the difference in trends between the two metrics with respect to the two types of references show that evaluating this task cannot be simply accomplished with one existing metric: depending on the combination of metric and references, it's possible to mask the difference between disfluent and fluent systems, unless you have word-level disfluency annotations, which are more difficult to obtain.

5 Conclusion

Machine translation applications for speech can suffer due to conversational speech phenomena, particularly the presence of disfluencies. Previous work to remove disfluencies in speech translation did so as a separate step between speech recognition and machine translation, which is not possible using end-to-end models. Using clean references for disfluent data collected by Salesky et al. (2018), we extend their text baseline to speech input and provide first results for direct generation of fluent text from noisy disfluent speech.

While fluent training data enables research on this task with end-to-end models, it is unlikely to have this resource for every corpus and domain and it is expensive to collect. In future work, we hope to reduce the dependence on fluent target data during training through decoder pretraining on external non-conversational corpora or multitask learning. Further, standard metrics alone do not tell the full story for this task; additional work on evaluation metrics may better demonstrate the differences between such systems.

References

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. *Proc. of Interspeech.* ArXiv:1803.09164.

Susanne Burger, Victoria MacLaren, and Hua Yu. 2002. The isl meeting corpus: The impact of meeting type on speech style.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Eunah Cho, Sarah Fünfer, Sebastian Stüker, and Alex Waibel. 2014. A corpus of spontaneous speech in lectures: The kit lecture corpus for spoken language processing and translation.

Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2013. Crf-based disfluency detection using semantic features for german to english spoken language translation.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks.

David Graff, Shudong Huang, Ingrid Cartagena, Kevin Walker, and Christopher Cieri. Fisher spanish speech (LDC2010S01). Https://catalog.ldc.upenn.edu/ ldc2010s01.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*

Matthias Honal and Tanja Schultz. 2005. Automatic disfluency removal on recognized spontaneous speech-rapid adaptation to speaker-dependent disfluencies.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proc. of ICLR*. ArXiv:1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation.

Gaurav Kumar, Matt Post, Daniel Povey, and Sanjeev Khudanpur. 2014. Some insights from translating conversational telephone speech.

Adrien Lardilleux and Yves Lepage. 2017. Charcut: Human-targeted character-based mt evaluation with loose differences. *Proc. of IWSLT*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, et al. 2018. Xnmt: The extensible neural machine translation toolkit. *arXiv:1803.00188*.

Toan Q Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. *Proc. of NAACL HLT*. ArXiv:1710.01329.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved speech-to-text translation with the fisher and callhome spanish–english speech translation corpus.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit.

Elizabeth Salesky, Susanne Burger, Jan Niehues, and Alex Waibel. 2018. Towards fluent translations from disfluent speech. *Proc. of SLT*.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-attentional acoustic models. *Proc. of EMNLP*. ArXiv:1803.09519.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision.

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. Automatic disfluency removal for improving spoken language translation.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. *arXiv:1703.08581*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *Proc. of Interspeech*. ArXiv:1604.03209.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional networks for end-to-end speech recognition.

A Appendix. LSTM/NiN Encoder and Training Procedure Details

A.1 Encoder Downsampling Procedure

Weiss et al. (2017) and Bansal et al. (2018) use two strided convolutional layers atop three bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) layers to downsample input sequences in **time** by a total factor of 4. Weiss et al. (2017) additionally downsample **feature** dimensionality by a factor of 3 using a ConvLSTM layer between their convolutional and LSTM layers. This is in contrast to the pyramidal encoder (Chan et al., 2016) from sequence-to-sequence speech recognition, where pairs of consecutive layer outputs are concatenated before being fed to the next layer to halve the number of states between layers.

To downsample in time we instead use the LSTM/NiN model used in Sperber et al. (2018) and Zhang et al. (2017), which stacks blocks consisting of an LSTM, a network-in-network (NiN) projection, layer batch normalization and then a ReLU non-linearity. NiN denotes a simple linear projection applied at every timestep, performing downsampling by a factor of 2 by concatenating pairs of adjacent projection inputs. The LSTM/NiN blocks are extended by a final LSTM layer for a total of three BiLSTM layers with the same total downsampling of 4 as Weiss et al. (2017) and Bansal et al. (2018). These blocks give us the benefit of added depth with fewer parameters.

A.2 Training Procedure

We follow the training procedure from Sperber et al. (2018). The model uses variational recurrent dropout with probability 0.2 and target character dropout with probability 0.1 (Gal and Ghahramani, 2016). We apply label smoothing (Szegedy et al., 2016) and fix the target embedding norm to 1 (Nguyen and Chiang, 2018). For inference, we use a beam size of 15 and length normalization with exponent 1.5. We set the batch size dynamically depending on the input sequence length such that the average batch size was 36. We use Adam (Kingma and Ba, 2015) with initial learning rate of 0.0003, and decay by 0.5 when validation BLEU did not improve first over 10 epochs and after 5 epochs after the first decay. We do not use L2 weight decay or Gaussian noise, and use a single model replica. All models use the same preprocessing as previous work on this dataset: lowercasing and removing punctuation aside from apostrophes.

Relation Classification Using Segment-Level Attention-based CNN and Dependency-based RNN

Van-Hien Tran[1], **Van-Thuy Phi**[1], **Hiroyuki Shindo**[1,2], and **Yuji Matsumoto**[1,2]

[1] Nara Institute of Science and Technology
[2]RIKEN Center for Advanced Intelligence Project (AIP)
[1]`{tran.van_hien.ts1,phi.thuy.ph8,shindo,matsu}@is.naist.jp`

Abstract

Recently, relation classification has gained much success by exploiting deep neural networks. In this paper, we propose a new model effectively combining Segment-level Attention-based Convolutional Neural Networks (SACNNs) and Dependency-based Recurrent Neural Networks (DepRNNs). While SACNNs allow the model to selectively focus on the important information segment from the raw sequence, DepRNNs help to handle the long-distance relations from the shortest dependency path of the related entities. Experiments on the SemEval-2010 Task 8 dataset show that our model is comparable to the state-of-the-art without using any external lexical features.

1 Introduction

Relation classification (RC) is a fundamental task in Natural Language Processing (NLP) that aims to identify semantic relations between pairs of marked entities in given sentences (instances). It has attracted much research effort as it plays a vital role in many NLP applications such as Information Extraction and Question Answering (Nguyen and Grishman, 2015). Traditional approaches (Kambhatla, 2004; Zhang et al., 2006) usually rely heavily on hand-crafted features and lexical resources, or elaborately designed kernels, which are time-consuming and challenging to adapt to novel domains. Recently, neural network (NN) models have dominated the work on RC since they can effectively learn meaningful hidden features without human intervention.

However, most previous NN models only exploit one of the following structures to represent relation instances: raw word sequences (Zhou et al., 2016; Wang et al., 2016) and dependency trees (Wen, 2017; Le et al., 2018). While raw sequences can provide all the information of relation instances, they also add noise to the models from redundant information. While dependency tree structures help the models focus on the concise information captured by the shortest dependency path (SDP) between two entities, they lose some supplementary context in the raw sequence. It is clear that the raw sequence and SDP highly complement each other. We, therefore, combine them to be more effective in determining the relation without losing any information.

While CNNs are able to learn short patterns (local features) (LeCun et al., 1995), RNNs have been effective in learning word sequence information (long-distance features) (Chung et al., 2014). In this paper, we present a new model combining both CNNs and RNNs, exploiting the information from both the raw sequence and the SDP.

Our contributions are summarized as follows:

(a) We combine Entity Tag Feature (ETF) (Qin et al., 2016) and Tree-based Position Feature (TPF) (Yang et al., 2016) to improve the semantic information between the marked entities in the raw input sentences.

(b) We propose Segment-Level Attention-based Convolutional Neural Networks (SACNNs) which automatically pay special attention to the important text segments from the raw sentence for RC.

(c) We build Dependency-based Recurrent Neural Networks (DepRNNs) on the SDP to gain long-distance features. Then, we combine the SACNN and the DepRNN to preserve the full relational information. Our proposed model achieves new state-of-the-art results on SemEval-2010 Task 8, compared with other complex models.

2 Related Work

RC plays a significant role in many NLP applications. Recent work usually present the task from a supervised perspective.

Proceedings of NAACL-HLT 2019, pages 2793–2798
Minneapolis, Minnesota, June 2 - June 7, 2019.

Traditional supervised approaches can be divided into feature-based methods and kernel methods. Feature-based methods focus on extracting and combining relevant features. Rink and Harabagiu (2010) leveraged useful features to achieve the best performance on SemEval-2010 Task 8. Meanwhile, kernel methods measure the structural similarity between two data samples, based on carefully designed kernels. Wang (2008) combined convolutional kernel and syntactic features to gain benefits for relation extraction.

Nowadays, deep neural networks are widely utilized in RC. Zeng et al. (2014) exploited a CNN to extract lexical and sentence features. Qin et al. (2016) used ETF to specify target entities in input sentences and fed them to a CNN. Vu et al. (2016) combined CNN and RNN to improve performance. Some recent work leveraged SDP for RC. Yang et al. (2016) proposed a position encoding CNN based on dependency parse trees, while Wen (2017) presented a model that learns representations from SDP, using both CNN and RNN.

3 Our Method

Given a sentence S with an annotated pair of entities ($e1$, $e2$), we aim to identify the semantic relation between them. Since the set of target relations is pre-defined, RC can be treated as a multi-class classification problem. In this section, we describe our model in detail for resolving this problem.

3.1 Input Representation

In Figure 1, Entity Tag Feature (ETF) is firstly used to annotate two entities in each raw sentence. Then, each word is represented by the concatenation of two parts: Word Embedding (WE) and Tree-based Position Features (TPFs). The representation sequence is then fed to the SACNN.

Entity Information. As the pairs of entities ($e1$, $e2$) are previously known, it is important to provide their information to the NNs. Following the work of Qin et al. (2016), we also use ETF which involves adding four tokens: $\langle e1S\rangle$, $\langle e1E\rangle$, $\langle e2S\rangle$ and $\langle e2E\rangle$ to each input sentence.

Word Embedding. Distributed representations of words in a vector space have helped learning algorithms to achieve better performance in NLP tasks (Mikolov et al., 2013). Following most previous work, we also use pre-trained word embeddings to initialize input word tokens in our model.

Tree-based Position Features. Yang et al. (2016) proposed TPFs for encoding relative distances of the current word to marked entities in dependency trees. The relative distance refers to the length of the SDP between the current word and the target entity. Then, each integer number is represented by a randomly initialized vector. Since TPFs help the neural network focus on crucial words and phrases in a sentence (Yang et al., 2016), we therefore utilize TPFs in our model. In Figure 1, TPF$_1$ and TPF$_2$ are relative distance features of each word to $e1$ and $e2$, respectively. For the four tokens: $\langle e1S\rangle$, $\langle e1E\rangle$, $\langle e2S\rangle$ and $\langle e2E\rangle$, which do not belong to the dependency tree, we simply pad zero vectors for their TPFs.

SDP. For the input of the DepRNN, we merely use the SDP between two marked entities from the original sentence as in Figure 1. Each normal word in the SDP is represented by a vector from pre-trained word embeddings. Meanwhile, following Le et al. (2018), we also consider dependency relations between words in the SDP and represent each dependency relation d_i as a vector D_i that is the concatenation of two vectors as follows:

$$D_i = Dtyp_i \oplus Ddir_i,$$

where *Dtyp* is the undirected dependency vector (*i.e.,* nmod), and *Ddir* is the orientation of the dependency vector (*i.e.,* left-to-right or vice versa). Both *Dtyp* and *Ddir* are initialized randomly.

3.2 Framework

The architecture of our model is illustrated in Figure 1. The example sentence with two entities $e1$ (*play*) and $e2$ (*religion*) is labeled by the directional relation "*Message-Topic(e1;e2)*". While the raw sequence is passed to the SACNN, the SDP between $e1$ and $e2$ is used in the DepRNN.

Segment Attention-based CNN. In the SACNN, each raw sentence is divided into three segments according to two entities: the left segment, the middle segment, and the right segment. The repetitions of $e1$ and $e2$ in these segments help the semantic meaning of each segment to be more clear. Intuitively, the middle segment is often more important to reflect the semantic relation. Qin et al. (2016) only used the middle segment with a CNN for RC, while Vu et al. (2016) proposed an extended middle context to pay special attention to the middle part.

Although the middle segment is more significant than two remaining segments in many cases,

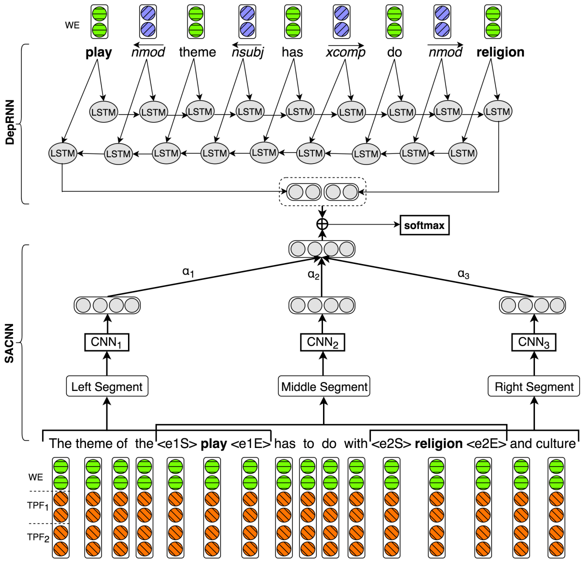


Figure 1: Our model for relation classification.

it is not always true for all. For example, in the sentence "*All other* $\langle e1S \rangle$ ***blood*** $\langle e1E \rangle$ $\langle e2S \rangle$ ***products*** $\langle e2E \rangle$ *are derived from whole blood.*" with the relation label "*Entity-Origin(e2;e1)*", the right segment is more important to reflect the relation type. Besides, the left and right segments might also provide the necessary information to RC. We therefore proceed three segments independently through three separate CNNs, which allow the model to automatically identify segments containing important information. Each CNN includes one convolutional layer and one max-pooling layer.

Let M be a matrix consisting of output vectors of three CNNs: $M = [m_1, m_2, m_3]$, where m_i is the output of CNN_i. The final representation r_1 of the raw sentence generated by SACNN is formed by a weighted sum of output vectors in M:

$$z_i = tanh(m_i),$$

$$\alpha_i = \frac{exp(w^T z_i + b)}{\sum_{i=1}^{3} exp(w^T z_i + b)},$$

$$r_1 = \sum_{i=1}^{3} \alpha_i m_i,$$

where w is a weight vector, w^T is its transformation, and b is a bias parameter.

Dependency-based RNN. While SACNN can learn local features, it cannot handle long-distance dependency between two entities. This disadvantage causes difficulty in correctly assigning subject and object roles of two entities when capturing the directional relation. Meanwhile, RNN could tackle the problem of long-distance pattern learning (Zhang and Wang, 2015). Besides, the SDP naturally offers the relative positions of subjects and objects through the path directions (Xu et al., 2015). We, therefore, exploit SDP based on RNN to gain the information in the directional relation.

An shown in Figure 1, we use Bidirectional Long Short-Term Memory (BLSTM) on the SDP between two entities. Due to its ability to capture long term memory, the BLSTM accumulates increasingly richer information as it goes through the SDP from both two forward and backward directions (Palangi et al., 2016). When it reaches the last two words, the last two hidden states are expected to provide the full semantic meaning of the whole SDP. Additionally, since the length of the SDP is often not so long, we concatenate two output vectors of the last two hidden states as the final representation r_2 of the SDP by DepRNN.

Combination of SACNN and DepRNN. Finally, we combine both SACNN and DepRNN models to exploit fully their own distinct advantages. While SACNN can focus on important segments and gain local features, DepRNN helps to handle long-distance dependency between two entities based on the SDP as well as provide subject and object roles of two entities for the directional relation. Therefore, the final representation r of the relation instance is concatenated by two output vectors (r_1 , r_2) of SACNN and DepRNN, which is then fed to a softmax classifier.

4 Experiments

4.1 Datasets and Settings

We evaluate our model on the SemEval2010 Task 8 which contains $8,000$ training sentences and $2,717$ test sentences, with 19 relations (9 directed relations and an undirected *Other* class). Therefore, the relation classification task is treated as a multi-class classification problem. Following previous work, the official macro-averaged F1-score, which excludes the *Other* relation, is used for evaluation.

We randomly held out 10% of the training set for validation. The Stanford Parser is also used to convert sentences to dependency trees.

For word embeddings, we use the 300-dimensional embeddings of Komninos and Manandhar (2016). In this work, we do not focus on comparing the effectiveness of the different pre-

SACNN	F1
WE, ETF	83.9
WE, TPF	84.5
WE, ETF, TPF	**85.1**

Table 1: Comparison of different features in SACNN. WE, ETF, and TPF stand for Word Embedding, Entity Tag Feature, and Tree-based Position Feature.

Model	Input	F1
CNN	Original Sentence	83.5
CNN	Middle Segment	84.1
SACNN	Three Segments	**85.1**

Table 2: Effectiveness of the segment-level attention.

Model	F1
DepRNN	83.8
SACNN	85.1
Combined	**85.8**

Table 3: Evaluation of our combined model.

Model	Features	F1
SVM (Rink and Harabagiu, 2010)	Rich features	82.2
BLSTM+Attention (Zhou et al., 2016)	WE, ETF	84.0
PECNN (Yang et al., 2016)	WE, DT, TPF, *POS, NER, WordNet*	84.6
CNN+BLSTM (Wang et al., 2017)	WE, DT, PF, *POS,* *GR, NER, WordNet*	84.7
SR-BRCNN (Wen, 2017)	WE, DT, *POS, NER, WordNet*	85.1
CNN+BLSTM (Zhang et al., 2018)	WE, PF	83.7
CNN+BLSTM+Attention Our model	WE, DT, TPF, ETF	**85.8**

Table 4: Comparison of different classification models. DT, PF stand for Dependency Tree, Position Feature. The *italic* features are external lexical features used.

trained embedding sets. The above pre-trained embedding set is selected since it embeds dependency context to provide valuable syntactic information. Four tokens: $\langle e1S \rangle$, $\langle e1E \rangle$, $\langle e2S \rangle$, $\langle e2E \rangle$ and out-of-vocabulary words are initialized by sampling from a uniform distribution (Kim, 2014). TPF is 15-dimensional and initialized randomly. Thus, the representation of each word has a dimensionality of 330 in the raw sentence.

Hyper-parameters in our model are as follows: 100 filters for each window size [3, 4, 5] and ReLU as the activation function for each CNN in SACNN. In DepRNN, the dimension of each token is 300, the $tanh$ activation function is applied to the last two hidden states, the dimension of each hidden state vector is 150. Other parameters include: L2 regularization with a weight of 10^{-4}, a mini-batch size of 64, a dropout rate at the final layer p = 0.5 before a softmax classifier.

4.2 Results and Analysis

Impact of SACNN and DepRNN. We consider the performance of each model by feeding separately their output vector to a softmax classifier.

In Table 1, we see the effect of different features to SACNN's performance. Combining ETF and TPF significantly enhances the $F1$ score by 0.6%. It proves that ETF and TPF complement each other to more fully provide information about the marked entities and important words to SACNN.

We also examine the segment-level attention mechanism of SACNN. In Table 2, with the same input features (WE, ETF, TPF), the segment-level attention mechanism makes a great contribution by increasing the $F1$ score by 1%.

To check the effect of combining SACNN and DepRNN, in Table 3, we compare the performance of each model to our combined model. First, the SACNN's performance is superior to the DepRNN. One possible reason is that while SACNN selectively focuses on the important segments as well as gains local features from the raw sentences, DepRNN based on the SDP, which is short in the SemEval2010 Task 8, can only provide effectively the entities role. Then, by combining SACNN and DepRNN, our model can exploit the fully necessary information and achieve the best performance.

Comparisons with the State of the Art. We compare our model to some recent work on RC in Table 4. Most previous work exploited some external lexical features (WordNet, NER) and combine NNs to improve the performance (Yang et al., 2016; Wang et al., 2017). Wang et al. (2017) and Wen (2017) proposed complex structures for integrating the CNN and the LSTM, and achieved an $F1$ of 84.7% and 85.1% respectively. Zhang et al. (2018) combined CNN and BLSTM, and reached an $F1$ of 83.7% using only WE, PF features.

Without using any external lexical resources, our model achieves an $F1$ of 85.8%, showing that combining SACNN and DepRNN is very effective, since SACNN helps to selectively focus on the important segments and gains local features, DepRNN provides the role information of subject and object of two entities in addressing the relation directionality. Comparing with some recent work, our model obtains a notable performance.

5 Conclusion

This work presents a new model that combines the SACNN and the DepRNN for RC. Combining ETF and TPF provides entity and semantic information of the input sentences to the model effectively. We also propose the SACNN which automatically focus on the essential segments and gains local features. Besides, the DepRNN helps to exploit long-distance dependency between two entities and their roles. Finally, combining the SACNN and the DepRNN brings the best performance since they highly complement each other. Our model achieved a notable performance on the SemEval2010 Task 8 without using any external lexical resources.

Acknowledgments

This work was partly supported by JST CREST Grant Number JPMJCR1513, Japan. We are grateful to the members of the Computational Linguistics Laboratory, NAIST and the anonymous reviewers for their insightful comments.

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Hoang-Quynh Le, Duy-Cat Can, Sinh T. Vu, Thanh Hai Dang, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Large-scale exploration of neural relation classification architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2266–2277, Brussels, Belgium. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Pengda Qin, Weiran Xu, and Jun Guo. 2016. An empirical convolutional neural network approach for semantic relation classification. *Neurocomputing*, 190:1–9.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. *arXiv preprint arXiv:1605.07333*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Pengfei Wang, Zhipeng Xie, and Junfeng Hu. 2017. Relation classification via cnn, segmented max-pooling, and sdp-blstm. In *International Conference on Neural Information Processing*, pages 144–154. Springer.

Ji Wen. 2017. Structure regularized bidirectional recurrent convolutional neural network for relation classification. *arXiv preprint arXiv:1711.02509*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 65–74.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

Xiaobin Zhang, Fucai Chen, and Ruiyang Huang. 2018. A combination of rnn and cnn for attention-based relation classification. *Procedia computer science*, 131:911–917.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Document-Level Event Factuality Identification via Adversarial Neural Network

Zhong Qian[1], **Peifeng Li**[1 2], **Qiaoming Zhu**[1 2] **and Guodong Zhou**[1 2]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]AI Research Institute, Soochow University, Suzhou, China
`qianzhongqz@163.com`, `{pfli, qmzhu, gdzhou}@suda.edu.cn`

Abstract

Document-level event factuality identification is an important subtask in event factuality and is crucial for discourse understanding in Natural Language Processing (NLP). Previous studies mainly suffer from the scarcity of suitable corpus and effective methods. To solve these two issues, we first construct a corpus annotated with both document- and sentence-level event factuality information on both English and Chinese texts. Then we present an LSTM neural network based on adversarial training with both intra- and inter-sequence attentions to identify document-level event factuality. Experimental results show that our neural network model can outperform various baselines on the constructed corpus.

1 Introduction

Document-level event factuality identification is the task of deciding the commitment of relevant sources towards the factual nature of an event, and to determine whether an event is a fact, a possibility, or an impossible situation from the view of document. Identifying document-level factuality of events requires comprehensive understanding of documents. As illustrated in Figure 1 where events are in ***bold***, the event "*reach*" (including its other forms) have various factuality values in different sentences. For example, in paragraph 2, "*reach*" is impossible/CT- according to the negative word "*denied*", while in paragraph 3, "*reach*" is possible/PS+ due to the speculative word "*may*". The main contents of this document is "*Mexico denied that they will reach an agreement with the U.S. on the new trade deal*", and the document-level factuality of the event "*reach*" is CT-.

> *According to Politico.com, it is said the United States will* ***reach*****(CT+)** *an agreement with Mexico on the new trade deal that will replace North American Free Trade Agreement (NAFTA) before December, 2017.*
>
> *However, Mexican Economy Minister Ildefonso Guajardo denied that they plan to* ***reach*****(CT-)** *any agreement with the U.S. on the trade deal talks.*
>
> *"We are not going to sacrifice the quality of an agreement because of pressure of time. We will keep engaged." he said. Just two days ago, Guajardo said the two sides may* ***reach*****(PS+)** *an agreement within hours.*
>
> *The government has not been informed that any agreement will be* ***reached*****(CT-)** *yet, said another two Mexican officials.*
>
> *During the past few weeks, the U.S. has been negotiating with Mexico on the new trade deal and has achieved much progress. Thus, some media speculate that they will possibly* ***reach*****(PS+)** *an agreement. But now it seems that the negotiations will continue before they can get a good deal.*
>
> (Time: November, 2017)
> (Document-level factuality of the event "*reach*" is **CT-**.)

Figure 1: An example document with both sentence- and document-level event factuality.

Document-level event factuality identification is fundamental for document-level NLP applications, such as machine reading comprehension, which aims to have machines read a text passage and then answer questions about the text. According to the document in Figure 1, the answer of the following question should be "*No*", which is consistent with the document-level factuality of the event "*reach*" (CT-):

Q*: Does the U.S. reach an agreement with Mexico on the new trade deal before December 2017?*

A*: No.*

Previous studies mostly reported on sentence-level event factuality identification tasks. On one hand, due to the scarcity of document-level event factuality corpus, these studies only considered the corpora annotated with sentence-level event factuality information, such as ACE 2005[1], LU (Diab et al., 2009), FactBank (Saurí and Pustejovsky, 2009), and UDS-IH2 (Rudinger et al., 2018).

On the other hand, previous studies only con-

---

[1] https://catalog.ldc.upenn.edu/LDC2006T06

Proceedings of NAACL-HLT 2019, pages 2799–2809
Minneapolis, Minnesota, June 2 - June 7, 2019.

sidered information within sentences, using rules (Saurí, 2008; Saurí and Pustejovsky, 2012), machine learning models (de Marneffe et al., 2012; Werner et al., 2015; Baly et al., 2018), and combinations of them (Qian et al., 2015; Stanovsky et al., 2017) for modeling. Neural network models have also recently been used for the sentence-level event factuality identification (He et al., 2017; Rudinger et al., 2018; Qian et al., 2018). According to Figure 1, document-level event factuality can not be deduced from each sentence-level factuality separately, but depends on the comprehensive semantic information of sentences. However, no suitable model for document-level task has been proposed yet.

To solve the issues above, this paper focuses on **document-level** event factuality identification. Our contributions can be summarized as follows.

1) We construct a document-level event factuality corpus, i.e. DLEF, on both English and Chinese texts. To our best knowledge, this is the first document-level event factuality corpus. The statistics on the corpora and the experimental results show that our corpus can sufficiently reflect linguistic characteristics of news texts, and provide adequate support on resource for research.

2) We propose an LSTM neural network with both intra- and inter-sequence attentions to identify document-level event factuality, and consider dependency paths from speculative and negative cues to the event and sentences containing the event as features. Due to the diversity of various contents of the texts in DLEF corpus, we employ Adversarial Training to improve the robustness of our model. Experimental results show that our model is superior to various baselines. The corpus and code of this paper will be released at `https://github.com/qz011/dlef`.

2 Corpus Annotation

This section introduces our **D**ocument-**L**evel **E**vent **F**actuality (DLEF) corpus, including the source, detailed guidelines for both document- and sentence-level event factuality, and the main statistics of the corpus.

2.1 Source

News texts contain sufficient speculative and negative information that is significant for event factuality identification, and usually focus on one event with a specific topic. Moreover, FactBank (Saurí and Pustejovsky, 2009), the sentence-level event factuality corpus, is also based on news texts.

	+	-	u
CT	CT+	CT-	CTu
PS	PS+	PS-	(NA)
U	(NA)	(NA)	Uu

Table 1: Event factuality values.

Therefore, we choose news texts in both English and Chinese to construct our corpus. The English corpus consists of 1727 documents from January 2017 to January 2018, among which 1506 documents are from China Daily[2], and 221 documents are from Sina Bilingual News[3]. The Chinese corpus consists of 4649 documents from Sina News[4]. These news documents cover various topics, e.g., politics, economy, culture, military, and society, which can reflect the heterogeneity of language in news texts.

2.2 Factuality Values

Saurí (2008) employed modality and polarity to describe event factuality values. Modality conveys the certainty degree of events, such as *certain* (CT), *probable* (PR), and *possible* (PS), while polarity expresses whether the event happened, including *positive*(+) and *negative*(-).

We use the factuality values in Table 1 according to Saurí (2008). Both PR and PS are speculative values and share similar certainty degrees in our corpus, and are merged into PS . U/u means *underspecified*. PSu and U+/- are not applicable (NA) and are not considered. Although CTu is applicable, neither document-level nor sentence-level event can be annotated as CTu in our corpus.

2.3 Annotation Guidelines

We adopt the definition of events proposed by TimeML (Pustejovsky et al., 2003) and consider the events that can be critical for computing the factuality. To ensure that the task is meaningful, we focus on the events that have various types of sentence-level factuality values. If there is more than one suitable event in a document, we annotate them separately.

First, the annotation of document-level event factuality is based on the definition, i.e., determining the factuality of an event from the view of

---

[2] http://www.chinadaily.com.cn

[3] http://roll.edu.sina.com.cn/english/syxw/index.shtml

[4] http://news.sina.com.cn

the document requires to understand the semantic of the document, including various sentence-level event factuality.

Second, sentence-level event factuality is essential for document-level task, which makes sense when document- and sentence-level factuality of events have different values. Therefore, we annotate the sentence-level event factuality as follows:

CT- events are negated by negative cues. For example, the events "*enter*" and "*merger*" are governed by negative cues "*impossible*" and "*denied*" in sentence S1 and S2, respectively.

(S1) *He said that the loss made it impossible for them to **enter** the semifinals.*

(S2) *Sinopec responded to National Business Daily, and denied the rumors of a **merger** with PetroChina.*

PS+ events (e.g. "*improve*" and "*fallen*") are governed by speculative cues (e.g., "*impossible*" and "*denied*"), just as illustrated in sentence S4 and S5.

(S4) *We think that further investigation may help to **improve** the treatment of people with similar infections.*

(S5) *The missing parts may have **fallen** during the flight of the plane.*

PS- events are governed by both speculative and negative cues. Different from CT-, PS- means incompletely negation. For example, the PS- event "*noticed*" is governed by the speculative cue "*probably*" and the negative cue "*not*" in sentence S6, and "*fall*" is modified by the cues "*may*" and "*not*" in sentence S7.

(S6) *The bus driver had probably not **noticed** the truck early enough.*

(S7) *Oil prices may not **fall** sharply due to the strong global demand.*

Uu events can appear in questions (e.g., "*considering*" in sentence S8) and in the intensional contexts with underspecified semantics (e.g., "*raises*" in sentence S9):

(S8) *Is France **considering** to leave EU?*

(S9) *The US dollar's declination can not be reversed even if the Federal Reserve **raises** rates three times.*

CT+ events are factual and do not meet the above conditions.

2.4 Statistics

The task is trivial if most documents have only one type of sentence-level factuality value, and in this case, document-level factuality probably shares the same value. To understand the usefulness of document-level event factuality identification and DLEF corpus, we launched the statistics of documents with n different types of sentence-level event factuality values shown in Table 2. From the table we can find that for English corpus there are 41.94% CT- and 66.06% PS+ documents with different sentence-level event factuality values, but these CT+ documents only cover 11.13%. While for Chinese corpus, these CT- and PS+ documents cover 63.41% and 62.15%, but these CT+ documents only make up 14.23%.

Corpus	**Docs**	***n*=1**	***n*=2**	***n*⩾3**
English	CT-	162	97	20
	PS+	93	157	24
	PS-	2	6	4
	Uu	5	6	1
	CT+	1022	119	9
	Total	1284	385	58
Chinese	CT-	491	612	239
	PS+	321	425	102
	PS-	9	11	16
	Uu	8	5	7
	CT+	2061	290	52
	Total	2890	1343	416

Table 2: Statistics of the documents in DLEF corpus with *n* types of sentence-level event factuality values.

Table 2 indicates that sentence-level factuality usually agrees with document-level factuality in CT+ documents, making them straightforward to be identified. However, in those non-CT+ documents with non-factual document-level values, sentence-level factuality is likely to have different values from documents, making them more difficult to be identified. In general, English and Chinese corpus have 25.64% and 37.84% documents with different sentence-level event factuality values, indicating this corpus is suitable for the document-level event factuality identification.

Table 3 shows the statistics of the DLEF corpus. CT+ document-level events are in the majority, because information reported by news texts is usually real.

Kappa (Cohen, 1960) is employed to measure the inter-annotator agreement of annotating document- and sentence-level event factuality between the two independent annotators who annotate the entire corpus, just as shown in Table 4. These two annotators are postgraduate stu-

Corpus	Statistics		
English	Documents	CT-	279/16.16%
		PS+	274/15.87%
		PS-	12/0.69%
		Uu	12/0.69%
		CT+	1150/66.59%
		Total	1727
	Sentence-Level Events	CT-	662/11.52%
		PS+	574/9.99%
		PS-	37/6.44%
		Uu	71/1.24%
		CT+	4401/76.61%
		Total	5745
	Avg. Len. of Sentences		14.73
	Avg. Len. of Documents		467.25
Chinese	Documents	CT-	1342/28.87%
		PS+	848/18.24%
		PS-	36/0.77%
		Uu	20/0.43%
		CT+	2403/51.69%
		Total	4649
	Sentence-Level Events	CT-	3923/20.69%
		PS+	2879/15.18%
		PS-	123/0.65%
		Uu	555/2.93%
		CT+	11482/60.55%
		Total	18962
	Avg. Len. of Sentences		29.00
	Avg. Len. of Documents		716.38

Table 3: Statistics of DLEF corpus. The units of length of English and Chinese texts are tokens and Chinese characters, respectively.

Corpus	Value	Sent-Level	Doc-Level
English	All	0.81	0.91
	CT-	0.82	0.89
	PS+	0.77	0.87
	CT+	0.84	0.93
Chinese	All	0.82	0.81
	CT-	0.83	0.82
	PS+	0.79	0.78
	CT+	0.83	0.84

Table 4: Inter-annotator agreement of event factuality.

dents who major in NLP. In addition, the Kappa of events on English and Chinese corpus are 0.83 and 0.85, respectively. All the Kappa values are larger than 0.75, proving the effectiveness and meaningfulness of our DLEF corpus.

3 Adversarial Neural Network for Document-Level Event Factuality Identification

This section describes the LSTM neural network for document-level event factuality identification in detail. As shown in Figure 2, to extract feature representations of events from the view of documents, we consider both intra- and inter-sequence attention for dependency paths and sentences. In addition, due to the diversity of contents of documents in DLEF corpus, we consider adversarial training to ensure the robustness of our model.

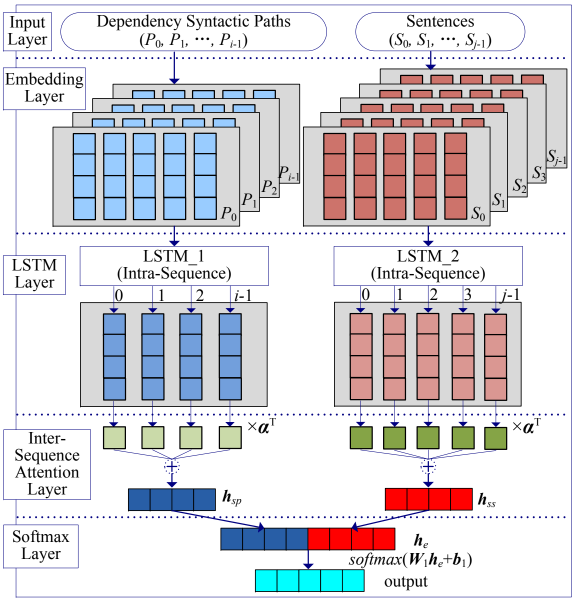


Figure 2: Neural network architecture for document-level event factuality identification.

3.1 Input Features

For our task, we use the specified events that have been annotated, and utilize the Chinese cues in CNeUn corpus (Zou et al., 2015) and the English cues in BioScope corpus (Vincze et al., 2008) that also considers multi-word cues, e.g., *rule out*. We do not use any annotated sentence-level event factuality. For one event, we consider all the sentences containing it, and mainly employ the following two features in our model:

1) **Syntactic Features**: Previous studies (Saurí and Pustejovsky, 2012; de Marneffe et al., 2012) have proved the effectiveness of dependency trees on event factuality identification tasks. Hence, we employ the dependency paths from speculative or negative cues to the event as syntactic features.

2) **Semantic Features**: We use the sentences containing the event as semantic features.

In addition, we also consider the above features in contexts of each sentence containing the event as the input, and set the windows size as 3, i.e., one sentence before and after the current one. If adjacent sentences contain speculative or negative cues, the dependency path is the concatenation of the path from the cue to the root and the path from the root to the event (Quirk and Poon, 2017).

3.2 LSTM with Two Attention Layers

A dependency path or sentence can be represented as $\boldsymbol{X}_0$ according to the embedding table. We employ LSTM with hidden units n_h to model the sequences from both directions to produce the forward hidden sequence $\overrightarrow{\boldsymbol{H}}$, the backward hidden sequence $\overrightarrow{\boldsymbol{H}}$, and the output sequence $\boldsymbol{H} = \overrightarrow{\boldsymbol{H}} + \overleftarrow{\boldsymbol{H}}$. We adopt the attention mechanism to capture the most important information from $\boldsymbol{H}$, and obtain the output $\boldsymbol{h}$:

$$\boldsymbol{H}_m = \tanh(\boldsymbol{H}) \tag{1}$$

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{v}^T \boldsymbol{H}) \tag{2}$$

$$\boldsymbol{h} = \tanh(\boldsymbol{H}\boldsymbol{\alpha}^T) \tag{3}$$

where $\boldsymbol{v} \in \mathbb{R}^{n_h}$ is the parameter. One event can have k sequences $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{k-1}$, whose representation is $\boldsymbol{H}_s = \boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{k-1}$ according to the above equations, where $\boldsymbol{H}_s \in \mathbb{R}^{k \times n_h}$. To extract the feature representation $\boldsymbol{h}_s \in \mathbb{R}^{n_h}$ from the k sequences, we utilize an inter-sequence attention mechanism that is computed as:

$$\boldsymbol{H}_{ms} = \tanh(\boldsymbol{H}_s) \tag{4}$$

$$\boldsymbol{\alpha}_s = \mathrm{softmax}(\boldsymbol{v}_s^T \boldsymbol{H}_{ms}) \tag{5}$$

$$\boldsymbol{h}_s = \tanh(\boldsymbol{H}_s \boldsymbol{\alpha}_s^T) \tag{6}$$

where $\boldsymbol{v}_s \in \mathbb{R}^{n_h}$ is the parameter. Suppose that an event has i dependency paths $\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_{i-1}$, and appears in j sentences $\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{j-1}$. Considering that dependency paths and sentences contain syntactic and semantic information, respectively, we employ two LSTM neural networks defined above to learn vector representations $\boldsymbol{h}_{sp}$ and $\boldsymbol{h}_{ss}$ of dependency paths and sentences, and concatenate them into the feature representation of the event $\boldsymbol{h}_e$:

$$\boldsymbol{h}_e = \boldsymbol{h}_{sp} \oplus \boldsymbol{h}_{ss} \tag{7}$$

where $\oplus$ is the concatenation operator. Finally, $\boldsymbol{h}_e$ is fed into the softmax layer to compute the probability of the factuality values of the event:

$$\boldsymbol{o} = \mathrm{softmax}(\boldsymbol{W}_1 \boldsymbol{h}_e + \boldsymbol{b}_1) \tag{8}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{c \times \dim(\boldsymbol{h}_e)}$ and $\boldsymbol{b}_1 \in \mathbb{R}^c$ are parameters, and $c = 5$ is the number of categories of factuality values (CT+, CT-, PS+, PS-, Uu). The objective function of the proposed neural network is designed as:

$$L_D(\theta) = -\frac{1}{m} \sum_{i=0}^{m-1} \log p(y_j^{(i)} | x^{(i)}, \theta) \tag{9}$$

where $y^{(i)}$ is the golden label of the instance $x^{(i)}$ and $p(y_j^{(i)} | x^{(i)})$ is the probability, m is the number of instances, and θ is the parameter set to learn. This model with **TWO** attention layers is denoted as **Att_2** in the next section.

3.3 Adversarial Training

As described in Section 2, documents in DLEF corpus cover various topics. To improve the robustness of our model, we consider Adversarial Training. Similar to previous work (Miyato et al., 2016; Wu et al., 2017), we add a small adversarial perturbation $\boldsymbol{e}_{adv}$ to word embeddings, and employ the following objective function:

$$L_{adv}(\boldsymbol{X}|\theta) = L(\boldsymbol{X} + \boldsymbol{e}_{adv}|\theta) \tag{10}$$

$$\boldsymbol{e}_{adv} = \underset{\|\boldsymbol{e}\| \leqslant \epsilon}{\arg\max}\, L(\boldsymbol{X} + \boldsymbol{e}|\hat{\theta}) \tag{11}$$

where $\hat{\theta}$ is a fixed copy value of the current θ and $\boldsymbol{X}$ is the input. Due to the intractable nature in the computation of Eq. (11), Goodfellow et al. (2014) proposed Eq. (12) to linear $L(\boldsymbol{X}|\theta)$ near $\boldsymbol{X}$ to approximate Eq. (11):

$$\boldsymbol{e}_{adv} = \epsilon \boldsymbol{g} / \|\boldsymbol{g}\| \tag{12}$$

$$\boldsymbol{g} = \nabla_{\boldsymbol{T}} L(\boldsymbol{X}|\hat{\theta}) \tag{13}$$

where $\boldsymbol{T}$ is the embedding table.

4 Experiments

We introduce the experimental settings and the baselines, finally presenting the experimental results and analysis in detail.

4.1 Experimental Settings

The PS- and Uu documents only cover 1.39% and 1.20% in our English and Chinese corpus, respectively. Therefore, we mainly focus on the performance of CT+, CT-, and PS+.

For fair comparison, we perform 10-fold cross-validation on English and Chinese corpora, respectively. In addition to Precision, Recall, and F1-measure for each category of factuality value, we consider macro- and micro-averaging to obtain the overall performance of all the categories of factuality values. The hidden units of LSTM are set as $n_h = 50$. We initialize word embeddings via Word2Vec (Mikolov et al., 2013), setting the dimensions as $d_0 = 100$, and fine-tuning them during training. SGD with momentum is applied to optimize our models.

Att_2 and **Att_2+AT** are the models proposed in Section 3 that consider the contexts, i.e., one sentence before and after the current sentence containing the event as the input. Compared to Att_2, Att_2+AT considers Adversarial Training (AT, the same below). We also consider the following baselines for the comparison with our models:

MaxEntVote is a maximum entropy model that only considers the view of *AUTHOR* (de Marneffe et al., 2012). We use maximum entropy model to identify sentence-level event factuality, and consider voting mechanism, i.e., choose the value committed by the most sentences as the document-level factuality value. We also consider other machine learning models, e.g. Lee et al. (2015), but obtain lower micro-/macro-averaged F1 on English (59.38/33.36) and Chinese corpus (53.91/43.20).

SentVote identifies sentence-level event factuality, and does not consider inter-sequence attention in the model proposed in Section 3. Similar to MaxEntVote model, voting mechanism is used to identify document-level event factuality in this SentVote model.

MP_2 considers Max-Pooling instead of attention compared with Att_2.

Att_1 considers only intra-sequence attention, but not the inter-sequence attention. For an event, we concatenate its i dependency paths and j sentences into one path and one sentence as the input, respectively.

4.2 Results and Analysis

4.2.1 Architecture of Neural Networks

Table 5 presents the performances of our models and baselines. MaxEntVote gives relatively lower results than other models, especially on CT- and PS+. SentVote models are better than MaxEntVote, but still obtain lower results than Att_2, which can prove that inter-sequence attention is more useful than voting. Max-pooling only selects the most active information for each dimension of features, While attention takes into account all the features and assigns weights for them according their degrees of importance. Hence, Att_2 gets better results than MP_L2. Att_1 only considers the intra-sequence attention and obtains lower results than Att_2, which proves the effectiveness of inter-sequence attention. Att_2 and Att_2+AT achieve better results than other baselines. Compared to Att_2, Att_2+AT considers the adversarial perturbation and training that can alleviate overfitting. Therefore, Att_2+AT is superior to Att_2, which can prove the effectiveness of adversarial training.

On both English and Chinese corpora, the performance of CT+ is better than those of PS+ and CT-. On one hand, it is easier to identify CT+ documents due to their majority. On the other hand, most news texts hardly contain bogus and false contents. Therefore, in most CT+ documents, sentence-level factuality values are consistent with the document-level value, just as in S10. However, in PS+ and CT- documents with non-CT+ document-level values, sentence-level factuality values have different viewpoints with the corresponding document, varying among CT-, PS+, and CT+, making the task more difficult, e.g., S11.

(S10) *India successfully* ***tested***(CT+) *a supersonic missile, capable of destroying an incoming ballistic missile at low altitude. ...... The* ***test***(CT+) *was carried out from a test range in Odisha , official sources said.*

(S11) *Argentine navy said it had not* ***contacted***(CT-) *the SAN Juan submarine. ... ... Some media previously said the navy may have received signals from the submarine and* ***contacted***(PS+) *it.*

4.2.2 Input of Neural Networks

For Att_2+AT, we also investigate the effects of contexts of the sentences containing events as the input on the performance. The results is given in Table 6, which shows that contexts can improve the performance more significantly on the Chinese corpus than the English corpus. We find that in the Chinese corpus these sentences are commonly in the same paragraph and have a strong semantic coherence. Therefore, information in adjacent sentences can contribute to the identification of the document-level factuality of the events in the current sentences.

Sentences S12 and S13 are adjacent sentences in one paragraph. The document-level factuality value of the event "*provided*" in S12 is CT-. However, the sentence-level value of "*provided*" is PS+. If we consider S13, the negative cue "*denied*" can lead to the correct document-level factuality value of "*provided*". While in the English corpus, similar sentences are much fewer, because paragraphs in most English news texts only contain one or two sentences, and sentences in different paragraphs share less semantic correlation

Corpus	Systems	CT-	PS+	CT+	Micro-A	Macro-A
English	MaxEntVote	58.17	35.89	75.14	68.42	56.40
	SentVote	70.22	57.85	83.98	78.06	70.68
	MP_2	70.57	56.39	83.72	77.65	70.23
	Att_1	65.25	53.65	79.18	73.23	66.03
	Att_2	73.88	59.29	88.59	81.84	73.92
	Att_2+AT	**76.87**	**62.14**	**89.84**	**83.56**	**76.28**
Chinese	MaxEntVote	62.44	58.29	72.22	67.72	64.32
	SentVote	72.66	58.39	80.68	74.70	70.58
	MP_2	74.34	65.17	78.91	75.22	72.81
	Att_1	68.82	49.78	81.89	71.12	67.28
	Att_2	81.41	73.35	86.58	82.79	80.45
	Att_2+AT	**83.35**	**74.06**	**87.52**	**84.03**	**81.64**

Table 5: F1-measures of baselines and our model.

Corpus	Systems	CT-	PS+	CT+	Micro-A	Macro-A
English	Att_2+AT	76.87	62.14	89.84	83.56	76.28
	w/o CTX	+0.95	+0.47	-0.80	-0.31	+0.21
	w/o Dpath (Only Sent)	-16.49	-8.28	-4.88	-7.08	-9.88
	w/o Sent (Only Dpath)	-20.79	-12.63	-19.25	-19.20	-17.56
Chinese	Att_2+AT	83.35	74.06	87.52	84.03	81.64
	w/o CTX	-3.05	-2.62	-1.03	-1.84	-2.23
	w/o Dpath (Only Sent)	-10.31	-7.23	-7.80	-8.49	-8.44
	w/o Sent (Only Dpath)	-17.53	-11.02	-14.79	-15.19	-14.44

Table 6: F1-measures of Att_2+AT with different input features.

than those in the same paragraph. Hence, performance improvement is less when considering adjacent sentences in the English corpus.

(S12) 外界质疑在竞标过程中，墨西哥政府为相关企业**提供**了“有利位置”。(*It is doubted that the Mexican government* ***provided*** *“vantage points” for the enterprises involved during the bidding process.*)

(S13) 墨西哥外交部在7日对此予以回应，否认了这种说法。(*The Mexican Foreign Ministry responded and denied the rumor on 7th.*)

If we consider more adjacent sentences, e.g., two sentences before and after the current sentence, however, the results will be a bit lower. The micro-/macro-averaged F1 on English and Chinese corpus are 81.20/75.65 and 82.57/80.91, respectively. We think the reason is that some sentences are far away from the current sentence and have little effect on the current event, and considering more contexts may also lead to overfitting.

Moreover, we explore the effects of considering only dependency path (Dpath) and only sentence (Sent) in Table 6. Att_2+AT achieves the best results when considering both paths and sentences as input, proving that both of them are effective features for our model. Att_2+AT obtains higher performance with only sentences than only paths as input, meaning that Att_2+AT is mainly beneficial from sentences that can offer semantic information. Error analysis shows that documents with incorrect identified values contains sentences with more speculative or negative cues:

(S14) *When asked if it might be* ***arson****, authorities said that no fire raiser has been found now, but the possibility of artificial arson should not been ruled out.*

S14 contains speculative cues “*if*”, “*might*”, “*possibility*” and negative cues “*no*”, “*not*”, “*ruled out*”. It is difficult to identify whether the events are governed by the cues when only considering the dependency paths and ignoring the semantic information offered by sentences. S14 can demonstrate the importance of semantic features.

4.2.3 Documents with Different Sentence-Level Event Factuality Values

As mentioned in Section 2.4, the document-level task becomes trivial if most documents have only one category of sentence-level factuality value that

Corpus	*n*	CT-	PS+	CT+	Micro-A	Macro-A
English	n=1	**85.63**	64.46	**94.91**	**91.36**	**81.67**
	$n\geqslant 2$	58.33	**65.79**	56.01	60.91	60.04
Chinese	n=1	**84.45**	**76.73**	**93.44**	**89.85**	**84.87**
	$n\geqslant 2$	82.68	69.91	58.51	73.22	70.37

Table 7: F1-measures of Att_2+AT on the documents with *n* types of sentence-level factuality values.

Corpus	Level	CT-	PS+	CT+	Micro-A	Macro-A
English	Sentence	72.05	59.68	91.14	85.96	74.29
	Document (with Joint Opt)	75.46	**62.80**	88.65	82.89	75.64
	Document (w/o Joint Opt)	**76.87**	62.14	**89.84**	**83.56**	**76.28**
Chinese	Sentence	74.20	68.88	87.73	81.98	76.94
	Document (with Joint Opt)	83.30	73.74	87.40	83.83	81.48
	Document (w/o Joint Opt)	**83.35**	**74.06**	**87.52**	**84.03**	**81.64**

Table 8: F1-measures of Att_2+AT with joint optimization.

is the same as document-level value. Table 7 shows the performance of Att_2+AT on the documents with n different types of sentence-level factuality values. The micro- and macro-averaged F1 of $n\geqslant 2$ are lower than those of n=1, indicating that the factuality of documents that have different types of sentence-level factuality are more difficult to identify due to the interference from sentence-level values.

We notice that in the Chinese corpus, the performance of CT- is much higher than that of PS+ and CT+ when $n\geqslant 2$. According to the analysis on the Chinese corpus, we find that most CT- documents are usually used to deny the rumors, i.e., those sentence-level events whose factuality values are not CT-. Therefore, the sentence-level CT- events are often in the topic sentences of the documents and dominate among sentences, which can contribute to the better results of document-level CT- events in Chinese corpus.

4.2.4 Joint Optimization Model

Because document-level event factuality is related with sentence-level factuality information, we also consider the joint optimization model for them. For sentence-level task, we use the LSTM neural network in Section 3 and only consider the current sentence, i.e., do not consider information in adjacent sentences and the inter-sequence attention layer. The objective of document- and sentence-level task are denoted as $L_D(\theta)$ and $L_S(\theta)$, and the objective of our joint optimization model is:

$$L_J(\theta) = \varepsilon L_D(\theta) + (1-\varepsilon) L_S(\theta) \qquad (14)$$

where ε=0.6 is the trade-off. The performance of both sentence-level and document-level event factuality identification is shown in Table 8. The micro-/macro-averaged F1 of joint optimization model on English and Chinese corpus are 82.89/75.64 and 83.83/81.48, respectively. Although document-level event factuality is based on the factuality information in sentences, sentence-level factuality value of an event only depends on the current sentence, and is likely to have a different value compared to the current document-level factuality. Therefore, the joint model can not improve the performance of document-level task.

5 Related Work

Researchers have studied document-level tasks in many NLP applications, e.g., sentiment analysis (Xu et al., 2016; Dou, 2017), named entity recognition (Luo et al., 2018), and machine translation (Born et al., 2017). But related studies on event factuality are limited to the sentence-level task. Diab et al. (2009) and Prabhakaran et al. (2010) presented studies of belief annotation and tagging, and classified predicate events into Committed Belief (CB), Non-CB or Not Applicable using a supervised framework. For factuality assessment, Lee et al. (2015) employed dependency features, while Stanovsky et al. (2017) considered deep linguistic information, such as modality classes, syntactic re-ordering with PropS tree annotation structure (Lotan et al., 2013). Baly et al. (2018) considered a set of features and predicted the factuality of reporting and bias of news media.

Saurí (2008) and Saurí and Pustejovsky (2012) proposed a rule-based model to identify event factuality on FactBank. de Marneffe et al. (2012) used a machine learning model and Qian et al. (2015) utilized a two-step framework combining machine learning and rule-based approaches on FactBank. In addition to FactBank, Prabhakaran et al. (2015) proposed a ongoing framework for a larger corpus based on LU, and Cao et al. (2013) constructed a Chinese corpus annotated with event factuality based on ACE 2005. However, no previous work annotated a document-level corpus. We construct DLEF corpus with document-level event factuality for the first time.

Some studies focused on document-level event identification task. Choubey et al. (2018) designed a rule-based classifier to identify central events according to event coreference relations. Liu et al. (2018) utilized a kernel-based neural model that captured semantic relations between discourse units for event salience identification. However, they did not consider the document-level event factuality. To our best knowledge, this paper is the first work on document-level event factuality identification task.

Previous studies (He et al., 2017; Rudinger et al., 2018; Qian et al., 2018) have tried neural network models on sentence-level factuality identification. Recent research has shown that neural networks with multi-level attention can extract meaningful information from heterogeneous input and improve the performance of NLP tasks, e.g., discourse relation (Liu and Li, 2016), relation classification (Wang et al., 2016), and question answering (Yu et al., 2017). Moreover, to improve the robustness of neural networks, related studies considered adversarial perturbation and training on text classification (Miyato et al., 2016) and relation extraction (Wu et al., 2017). This paper is in line in proposing an adversarial neural network with both intra- and inter-sequence attention.

6 Conclusion

We investigated document-level event factuality identification task by constructing a corpus annotated with document- and sentence-level event factuality based on both English and Chinese texts. To identify document-level event factuality, we proposed an LSTM neural network with both intra- and inter-sequence attention, and consider adversarial training to improve the robustness. Experimental results showed that document-level event identification on our DLEF corpus is useful, and our adversarial training model outperforms several baselines. To our knowledge, this is the first paper for the document-level event factuality identification.

In the future work, we will consider to detect events and their sentence-level and document-level factuality with a joint framework, and we will also continue to expand the scale of our DLEF corpus.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their comments on this paper. This work was partially supported by national Natural Science Foundation of China (NSFC) via Grant Nos. 61836007, 6177235, 61773276, 61673290.

References

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James R. Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3528–3539.

Leo Born, Mohsen Mesgar, and Michael Strube. 2017. Using a graph-based coherence model in document-level machine translation. In *Proceedings of DiscoMT@EMNLP 2017*, pages 26–35.

Yuan Cao, Qiaoming Zhu, and Peifeng Li. 2013. The construction of chinese event factuality corpus. *Journal of Chinese Information Processing*, 27(6):38–45.

Prafulla Kumar Choubey, Kaushik Raju, and Ruihong Huang. 2018. Identifying the most dominant event in a news article by mining event coreference relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 340–345.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational & Psychological Measurement*, 20(1):37–46.

Mona T. Diab, Lori S. Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed belief annotation and tagging. In *Proceedings of the Third Linguistic Annotation Workshop, LAW 2009*, pages 68–73.

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of EMNLP 2017*, pages 521–526.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

Tianxiong He, Peifeng Li, and Qiaoming Zhu. 2017. Identifying chinese event factuality with convolutional neural networks. In *Proceedings of CLSW 2017*, pages 284–292.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of EMNLP 2015*, pages 1643–1648.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of EMNLP 2016*, pages 1224–1233.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard H. Hovy. 2018. Automatic event salience identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1226–1236.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. Truthteller: Annotating predicate truth. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 752–757.

Ling Luo, Zhihao Yang, Pei Yang, Yin Zhang, Lei Wang, Hongfei Lin, and Jian Wang. 2018. An attention-based bilstm-crf approach to document-level chemical named entity recognition. *Bioinformatics*, 34(8):1381–1388.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*, pages 3111–3119.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2016. Virtual adversarial training for semi-supervised text classification. *CoRR*, abs/1605.07725.

Vinodkumar Prabhakaran, Tomas By, Julia Hirschberg, Owen Rambow, Samira Shaikh, Tomek Strzalkowski, Jennifer Tracey, Michael Arrigo, Rupayan Basu, Micah Clark, Adam Dalton, Mona Diab, Louise Guthrie, Anna Prokofieva, Stephanie Strassel, Gregory Werner, Yorick Wilks, and Janyce Wiebe. 2015. A new dataset and evaluation for belief/factuality. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 82–91, Denver, Colorado. Association for Computational Linguistics.

Vinodkumar Prabhakaran, Owen Rambow, and Mona T. Diab. 2010. Automatic committed belief tagging. In *COLING 2010*, pages 1014–1022.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium*, pages 28–34, Stanford University, Stanford, CA, USA.

Zhong Qian, Peifeng Li, Yue Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Event factuality identification via generative adversarial networks with auxiliary classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4293–4300.

Zhong Qian, Peifeng Li, and Qiaoming Zhu. 2015. A two-step approach for event factuality identification. In *IALP 2015*, pages 103–106.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of EACL 2017*, pages 1171–1182.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 731–744. Association for Computational Linguistics.

Roser Saurí. 2008. *A Factuality Profiler for Eventualities in Text*. Ph.D. thesis, Brandeis University, Waltham, MA, USA.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299.

Gabriel Stanovsky, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych. 2017. Integrating deep linguistic features in factuality prediction over unified datasets. In *Proceedings of ACL 2017*, pages 352–357.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(S-11).

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of ACL 2016*, pages 1298–1307.

Gregory Werner, Vinodkumar Prabhakaran, Mona Diab, and Owen Rambow. 2015. Committed belief tagging on the factbank and lu corpora: A comparative study. In *Proceedings of the Second Workshop on Extra-Propositional Aspects of Meaning in Computational Semantics (ExProM 2015)*, pages 32–40, Denver, Colorado. Association for Computational Linguistics.

Yi Wu, David Bamman, and Stuart J. Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP 2017*, pages 1778–1783.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of EMNLP 2016*, pages 1660–1669.

Dongfei Yu, Jianlong Fu, Tao Mei, and Yong Rui. 2017. Multi-level attention networks for visual question answering. In *Proceedings of CVPR 2017*, pages 4187–4195.

Bowei Zou, Qiaoming Zhu, and Guodong Zhou. 2015. Negation and speculation identification in chinese language. In *Proceedings of ACL-IJCNLP 2015*, pages 656–665.

Distant Supervision Relation Extraction with Intra-Bag and Inter-Bag Attentions

Zhi-Xiu Ye
University of Science and
Technology of China
`zxye@mail.ustc.edu.cn`

Zhen-Hua Ling
University of Science and
Technology of China
`zhling@ustc.edu.cn`

Abstract

This paper presents a neural relation extraction method to deal with the noisy training data generated by distant supervision. Previous studies mainly focus on sentence-level de-noising by designing neural networks with intra-bag attentions. In this paper, both intra-bag and inter-bag attentions are considered in order to deal with the noise at sentence-level and bag-level respectively. First, relation-aware bag representations are calculated by weighting sentence embeddings using intra-bag attentions. Here, each possible relation is utilized as the query for attention calculation instead of only using the target relation in conventional methods. Furthermore, the representation of a group of bags in the training set which share the same relation label is calculated by weighting bag representations using a similarity-based inter-bag attention module. Finally, a bag group is utilized as a training sample when building our relation extractor. Experimental results on the New York Times dataset demonstrate the effectiveness of our proposed intra-bag and inter-bag attention modules. Our method also achieves better relation extraction accuracy than state-of-the-art methods on this dataset[1].

1 Introduction

Relation Extraction is a fundamental task in natural language processing (NLP), which aims to extract semantic relations between entities. For example, sentence "[*Barack Obama*]$_{e1}$ was born in [*Hawaii*]$_{e2}$" expresses the relation *BornIn* between entity pair **Barack Obama** and **Hawaii**.

Conventional relation extraction methods, such as (Zelenko et al., 2002; Culotta and Sorensen, 2004; Mooney and Bunescu, 2006), adopted supervised training and suffered from the lack of large-scale manually labeled data. To address this issue, the distant supervision method (Mintz et al., 2009) was proposed, which generated the data for training relation extraction models automatically. The distant supervision assumption says that if two entities participate in a relation, **all** sentences that mention these two entities express that relation. It is inevitable that there exists noise in the data labeled by distant supervision. For example, the precision of aligning the relations in Freebase to the New York Times corpus was only about 70% (Riedel et al., 2010).

bag	sentence	correct?
B1	S1. **Barack Obamna** was born in **the United States**.	**Yes**
	S2. **Barack Obamna** was the 44th president of **the United States**	**No**
B2	S3. **Kyle Busch** , a **Las Vegas** resident who ran second to Johnson last year, finished third, followed by Kasey Kahne, Jeff Gordon and mark martin .	**No**
	S4. Hendrick drivers finished in three of the top four spots at **Las Vegas** , including **Kyle Busch** in second and ...	**No**

Table 1: Examples of sentences with relation *place_of_birth* annotated by distant supervision, where "Yes" and "No" indicate whether or not each sentence actually expresses this relation.

Thus, the relation extraction method proposed in (Riedel et al., 2010) argued that the distant supervision assumption was too strong and relaxed it to *expressed-at-least-once* assumption. This assumption says that if two entities participate in a relation, **at least one sentence** that mentions these two entities might express that relation. An example is shown by sentences S1 and S2 in Table 1. This relation extraction method first divided the training data given by distant supervision into bags where each bag was a set of sentences containing the same entity pair. Then, bag representations were derived by weighting sentences within

---

[1] The code is available at `https://github.com/ZhixiuYe/Intra-Bag-and-Inter-Bag-Attentions`.

Proceedings of NAACL-HLT 2019, pages 2810–2819
Minneapolis, Minnesota, June 2 - June 7, 2019.

each bag. It was expected that the weights of the sentences with incorrect labels were reduced and the bag representations were calculated mainly using the sentences with correct labels. Finally, bags were utilized as the samples for training relation extraction models instead of sentences.

In recent years, many relation extraction methods using neural networks with attention mechanism (Lin et al., 2016; Ji et al., 2017; Jat et al., 2018) have been proposed to alleviate the influence of noisy training data under the *expressed-at-least-once* assumption. However, these methods still have two deficiencies. First, only the target relation of each bag is used to calculate the attention weights for deriving bag representations from sentence embeddings at training stage. Here we argue that the bag representations should be calculated in a relation-aware way. For example, the bag B1 in Table 1 contains two sentences S1 and S2. When this bag is classified to relation *BornIn*, the sentence S1 should have higher weight than S2, but when classified to relation *PresidentOf*, the weight of S2 should be higher. Second, the *expressed-at-least-once* assumption ignores the **noisy bag problem** which means that all sentences in one bag are incorrectly labeled. An example is shown by bag B2 in Table 1.

In order to deal with these two deficiencies of previous methods, this paper proposes a neural network with multi-level attentions for distant supervision relation extraction. At the instance/sentence-level, i.e., intra-bag level, all possible relations are employed as queries to calculate the relation-aware bag representations instead of only using the target relation of each bag. To address the noisy bag problem, a bag group is adopted as a training sample instead of a single bag. Here, a bag group is composed of bags in the training set which share the same relation label. The representation of a bag group is calculated by weighting bag representations using a similarity-based inter-bag attention module.

The contributions of this paper are threefold. First, an improved intra-bag attention mechanism is proposed to derive relation-aware bag representations for relation extraction. Second, an inter-bag attention module is introduced to deal with the noisy bag problem which is ignored by the expressed-at-least-once assumption. Third, our methods achieve better extraction accuracy than state-of-the-art models on the widely used New York Times (NYT) dataset (Riedel et al., 2010).

2 Related Work

Some previous work (Zelenko et al., 2002; Mooney and Bunescu, 2006) treated relation extraction as a supervised learning task and designed hand-crafted features to train kernel-based models. Due to the lack of large-scale manually labeled data for supervised training, the distant supervision approach (Mintz et al., 2009) was proposed, which aligned raw texts toward knowledge bases automatically to generate relation labels for entity pairs. However, this approach suffered from the issue of noisy labels. Therefore, some subsequent studies (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) considered distant supervision relation extraction as a multi-instance learning problem, which extracted relation from a bag of sentences instead of a single sentence.

With the development of deep learning techniques (LeCun et al., 2015), many neural-network-based models have been developed for distant supervision relation extraction. Zeng et al. (2015) proposed piecewise convolutional neural networks (PCNNs) to model sentence representations and chose the most reliable sentence as the bag representation. Lin et al. (2016) employed PCNNs as sentence encoders and proposed an intra-bag attention mechanism to compute the bag representation via a weighted sum of all sentence representations in the bag. Ji et al. (2017) adopted a similar attention strategy and combined entity descriptions to calculate the weights. Liu et al. (2017) proposed a soft-label method to reduce the influence of noisy instances. All these methods represented a bag with a weighted sum of sentence embeddings, and calculated the probability of the bag being classified into each relation using the same bag representation at training stage. In our proposed method, intra-bag attentions are computed in a relation-aware way, which means that different bag representations are utilized to calculate the probabilities for different relation types. Besides, these existing methods focused on intra-bag attentions and ignored the noisy bag problem.

Some data filtering strategies for robust distant supervision relation extraction have also been proposed. Feng et al. (2018) and Qin et al. (2018b) both employed reinforcement learning to train instance selector and to filter out the samples with wrong labels. Their rewards were calcu-

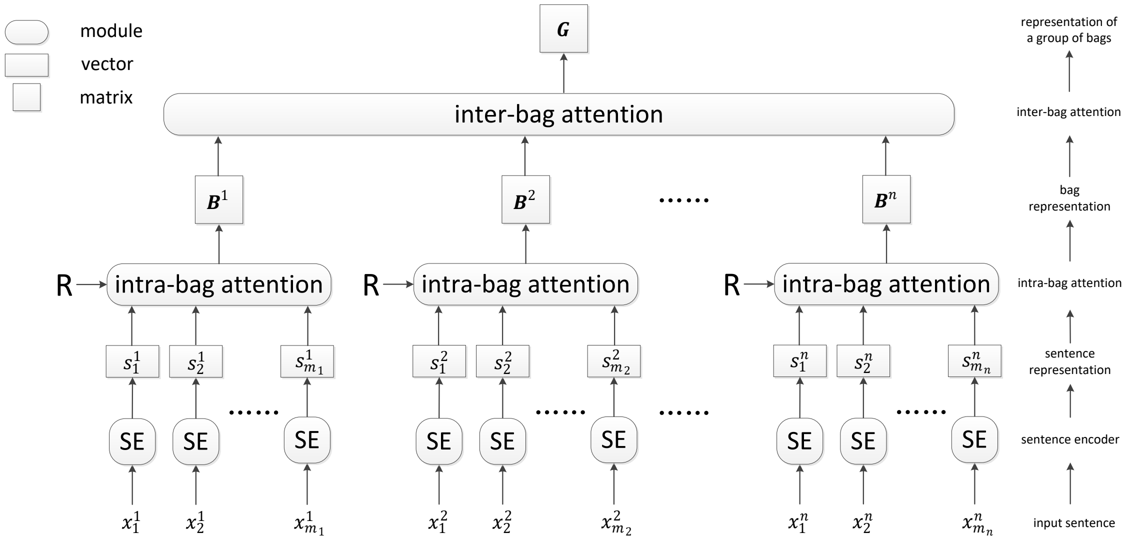


Figure 1: The framework of our proposed neural network with intra-bag and inter-bag attentions for relation extraction.

lated from the prediction probabilities and the performance change of the relation classifier respectively. Qin et al. (2018a) designed an adversarial learning process to build a sentence-level generator via policy-gradient-based reinforcement learning. These methods were proposed to filter out the noisy data at sentence-level and also failed to deal with the noisy bag problem explicitly.

3 Methodology

In this section, we introduce a neural network with intra-bag and inter-bag attentions for distant supervision relation extraction. Let $g = \{b^1, b^2, ..., b^n\}$ denote a group of bags which have the same relation label given by distant supervision, and n is the number of bags within this group. Let $b^i = \{x_1^i, x_2^i, ..., x_{m_i}^i\}$ denote all sentences in bag b^i, and m_i is the number of sentences in bag b^i. Let $x_j^i = \{w_{j1}^i, w_{j2}^i, ..., w_{jl_{ij}}^i\}$ denote the j-th sentence in the i-th bag and l_{ij} is its length (i.e., number of words). The framework of our model is shown in Fig. 1, which has three main modules.

- **Sentence Encoder** Given a sentence x_j^i and the positions of two entities within this sentence, CNNs or PCNNs (Zeng et al., 2015) are adopted to derive the sentence representation $\mathbf{s}_j^i$.
- **Intra-Bag Attention** Given the sentence representations of all sentences within a bag b^i and a relation embedding matrix $\mathbf{R}$, attention weight vectors $\boldsymbol{\alpha}_k^i$ and bag representations $\mathbf{b}_k^i$ are calculated for all relations, where k is the relation index.
- **Inter-Bag Attention** Given the representations of all bags with the group g, a weight matrix $\boldsymbol{\beta}$ is further calculated via similarity-based attention mechanism to obtain the representation of the bag group.

More details of these three modules will be introduced in the following subsections.

3.1 Sentence Encoder

3.1.1 Word Representation

Each word w_{jk}^i within the sentence x_j^i is first mapped into a d_w-dimensional word embedding. To describe the position information of two entities, the position features (PFs) proposed in (Zeng et al., 2014) are also adopted in our work. For each word, the PFs describe the relative distances between current word and the two entities and are further mapped into two vectors $\mathbf{p}_{jk}^i$ and $\mathbf{q}_{jk}^i$ of d_p dimensions. Finally, these three vectors are concatenated to get the word representation $\mathbf{w}_{jk}^i = [\mathbf{e}_{jk}^i; \mathbf{p}_{jk}^i; \mathbf{q}_{jk}^i]$ of $d_w + 2d_p$ dimensions.

3.1.2 Piecewise CNN

For sentence x_j^i, the matrix of word representations $\mathbf{W}_j^i \in \mathbb{R}^{l_{ij} \times (d_w+2d_p)}$ is first input into a CNN with d_c filters. Then, piecewise max pooling (Zeng et al., 2015) is employed to extract features from the three segments of CNN outputs, and the segment boundaries are determined by the positions of the two entities. Finally, the sentence representation $\mathbf{s}^i_j \in \mathbb{R}^{3d_c}$ can be obtained.

3.2 Intra-Bag Attention

Let $\mathbf{S}^i \in \mathbb{R}^{m_i \times 3d_c}$ represent the representations of all sentences within bag b^i, and $\mathbf{R} \in \mathbb{R}^{h \times 3d_c}$ denote a relation embedding matrix where h is the number of relations.

Different from conventional methods (Lin et al., 2016; Ji et al., 2017) where a unified bag representation was derived for relation classification, our method calculates bag representations $\mathbf{b}^i_k$ for bag b^i on the condition of all possible relations as

$$\mathbf{b}^i_k = \sum_{j=1}^{m_i} \alpha^i_{kj} \mathbf{s}^i_j, \tag{1}$$

where $k \in \{1, 2, ..., h\}$ is the relation index and α^i_{kj} is the attention weight between the k-th relation and the j-th sentence in bag b^i. α^i_{kj} can be further defined as

$$\alpha^i_{kj} = \frac{exp(e^i_{kj})}{\sum_{j'=1}^{m_i} exp(e^i_{kj'})}, \tag{2}$$

where e^i_{kj} is the matching degree between the k-th relation query and the j-th sentence in bag b^i. In our implementation, a simple dot product between vectors is adopted to calculate the matching degree as

$$e^i_{kj} = \mathbf{r}_k \mathbf{s}^{i\top}_j, \tag{3}$$

where $\mathbf{r}_k$ is the k-th row of the relation embedding matrix $\mathbf{R}$[2].

Finally, the representations of bag b^i compose the matrix $\mathbf{B}^i \in \mathbb{R}^{h \times 3d_c}$ in Fig. 1, where each row corresponds to a possible relation type of this bag.

3.3 Inter-Bag Attention

In order to deal with the noisy bag problem, a similarity-based inter-bag attention module is designed to reduce the weights of noisy bags dynamically. Intuitively, if two bags b^{i_1} and b^{i_2} are both labeled as relation k, their representations $\mathbf{b}^{i_1}_k$ and $\mathbf{b}^{i_2}_k$ should be close to each other. Given a group of bags with the same relation label, we assign higher weights to those bags which are close to other bags in this group. As a result, the representation of bag group g can be formulated as

$$\mathbf{g}_k = \sum_{i=1}^{n} \beta_{ik} \mathbf{b}^i_k, \tag{4}$$

where $\mathbf{g}_k$ is the k-th row of the matrix $\mathbf{G} \in \mathbb{R}^{h \times 3d_c}$ in Fig. 1, k is the relation index and β_{ik} composes the attention weight matrix $\boldsymbol{\beta} \in \mathbb{R}^{n \times h}$. Each β_{ik} is defined as

$$\beta_{ik} = \frac{exp(\gamma_{ik})}{\sum_{i=1}^{n} exp(\gamma_{ik})}, \tag{5}$$

where γ_{ik} describes the confidence of labeling bag b^i with the k-th relation.

Inspired by the self-attention algorithm (Vaswani et al., 2017) which calculates the attention weights for a group of vectors using the vectors themselves, we calculate the weights of bags according to their own representations. Mathematically, γ_{ik} is defined as

$$\gamma_{ik} = \sum_{i'=1,...,n, i' \neq i} \text{similarity}(\mathbf{b}^i_k, \mathbf{b}^{i'}_k), \tag{6}$$

where the function *similarity* is a simple dot product in our implementation as

$$\text{similarity}(\mathbf{b}^i_k, \mathbf{b}^{i'}_k) = \mathbf{b}^i_k \mathbf{b}^{i'\top}_k. \tag{7}$$

And also, in order to prevent the influence of vector length, all bag representations $\mathbf{b}^i_k$ are normalized to unit length as $\overline{\mathbf{b}^i_k} = \mathbf{b}^i_k / ||\mathbf{b}^i_k||_2$ before calculating Eq.(4)-(7).

Then, the score o_k of classifying bag group g into relation k is calculated via $\mathbf{g}_k$ and relation embedding $\mathbf{r}_k$ as

$$o_k = \mathbf{r}_k \mathbf{g}^\top_k + d_k, \tag{8}$$

where d_k is a bias term. Finally, a softmax function is employed to obtain the probability that the bag group g is classified into the k-th relation as

$$p(k|g) = \frac{exp(o_k)}{\sum_{k'=1}^{h} exp(o_{k'})}. \tag{9}$$

It should be noticed that the same relation embedding matrix $\mathbf{R}$ is used for calculating Eq.(3) and Eq.(8). Similar to Lin et al. (2016), the dropout strategy (Srivastava et al., 2014) is applied to bag representation $\mathbf{B}^i$ to prevent overfitting.

---

[2] We also tried $\mathbf{r}_k \mathbf{A} \mathbf{s}^{i\top}_j$, where $\mathbf{A}$ was a diagonal matrix, in experiments and achieved similar performance.

3.4 Implementation Details

3.4.1 Data Packing

First of all, all sentences in the training set that contain the same two entities are accumulated into one bag. Then, we tie up every n bags that share the same relation label into a group. It should be noticed that a bag group is one training sample in our method. Therefore, the model can also be trained in mini-batch mode by packing multiple bag groups into one batch.

3.4.2 Objective Function and Optimization

In our implementation, the objective function is defined as

$$\mathrm{J}(\theta) = -\sum_{(g,k)\in T} \mathrm{logp}(k|g;\theta) \qquad (10)$$

where T is the set of all training samples and θ is the set of model parameters, including word embedding matrix, position feature embedding matrix, CNN weight matrix and relation embedding matrix. The model parameters are estimated by minimizing the objective function J(θ) through mini-batch stochastic gradient descent (SGD).

3.4.3 Training and Test

As introduced above, at the training phase of our proposed method, n bags which have the same relation label are accumulated into one bag group and the weighted sum of bag representations is calculated to obtain the representation $\mathbf{G}$ of the bag group. Due to the fact that the label of each bag is unknown at test stage, each single bag is treated as a bag group (i.e., n=1) when processing the test set.

And also, similar to (Qin et al., 2018b), we only apply inter-bag attentions to positive samples, i.e., the bags whose relation label is not NA ($NoRelation$). The reason is that the representations of the bags that express no relations are always diverse and it's difficult to calculate suitable weights for them.

3.4.4 Pre-training Strategy

In our implementation, a pre-training strategy is adopted. We first train the model with only intra-bag attentions until convergence. Then, the inter-bag attention module is added and the model parameters are further updated until convergence again. Preliminary experimental results showed that this strategy can lead to better model performance than considering inter-bag attentions from the very beginning.

Component	Parameter	Value
word embedding	dimension	50
position feature	max relative distance	±30
	dimension	5
CNN	window size	3
	filter number	230
dropout	dropout rate	0.5
optimization	strategy	SGD
	learning rate	0.1
	batch size N_p	50
	batch size N_t	10
	group size n	5
	gradient clip	5.0

Table 2: Hyper-parameters of the models built in our experiments.

4 Experiment

4.1 Dataset and Evaluation Metrics

The New York Times (NYT) dataset was adopted in our experiments. This dataset was first released by (Riedel et al., 2010) and has been widely used by previous research on distant supervision relation extraction (Liu et al., 2017; Jat et al., 2018; Qin et al., 2018a,b). This dataset was generated by aligning Freebase with the New York Times (NYT) corpus automatically. There were 52 actual relations and a special relation NA which indicated there was no relation between two entities.

Following previous studies (Mintz et al., 2009; Liu et al., 2017), we evaluated our models on the held-out test set of the NYT dataset. Precision-recall (PR) curves, area under curve (AUC) values and Precision@N (P@N) values (Lin et al., 2016) were adopted as evaluation metrics in our experiments. All of the numerical results given by our experiments were the mean values of 10 repetitive trainings, and the PR curves were randomly selected from the repetitions because there was no significant visual difference among them.

4.2 Training Details and Hyperparameters

All of the hyperparameters used in our experiments are listed in Table 2. Most of them followed the hyperparameter settings in (Lin et al., 2016). The 50-dimensional word embeddings released by (Lin et al., 2016)[3] were also adopted for initialization. The vocabulary contained the words which appeared more than 100 times in the NYT corpus.

---

[3] `https://github.com/thunlp/NRE`.

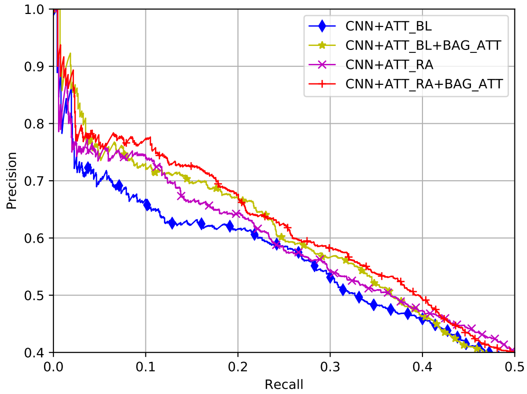


Figure 2: PR curves of different models using CNN sentence encoders.

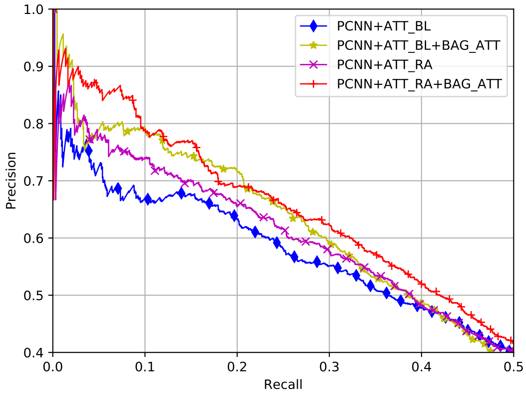


Figure 3: PR curves of different models using PCNN sentence encoders.

No.	Model	AUC
1	CNN+ATT_BL	0.376 ± 0.003
2	CNN+ATT_BL+BAG_ATT	0.388 ± 0.002
3	CNN+ATT_RA	0.398 ± 0.004
4	CNN+ATT_RA+BAG_ATT	0.407 ± 0.004
5	PCNN+ATT_BL	0.388 ± 0.004
6	PCNN+ATT_BL+BAG_ATT	0.403 ± 0.002
7	PCNN+ATT_RA	0.403 ± 0.003
8	PCNN+ATT_RA+BAG_ATT	$\mathbf{0.422 \pm 0.004}$

Table 3: AUC values of different models.

Two different batch sizes N_p and N_t were used for pre-training and training respectively. In our experiments, a grid search is employed using training set to determine the optimal values of n, N_p and N_t among $n \in \{3, 4, ..., 10\}$, $N_p \in \{10, 20, 50, 100, 200\}$ and $N_t \in \{5, 10, 20, 50\}$. Note that increasing the bag group size n may boost the effect of inter-bag attentions but lead to less training samples. The effects of inter-bag attentions would be lost when n=1. For optimization, we employed mini-batch SGD with the initial learning rate of 0.1. The learning rate was decayed to one tenth every 100,000 steps. The pre-trained model with only intra-bag attentions converged within 300,000 steps in our experiments. Thus, the initial learning rate for training the model with inter-bag attentions was set as 0.001.

4.3 Overall performance

Eight models were implemented for comparison. The names of these models are listed in Table 3, where *CNN* and *PCNN* denote using CNNs or piecewise CNNs in sentence encoders respectively, *ATT_BL* means the baseline intra-bag attention method proposed by (Lin et al., 2016), *ATT_RA* means our proposed relation-aware intra-bag attention method, and *BAG_ATT* means our proposed inter-bag attention method. At the training stage of the *ATT_BL* method, the relation query vector for attention weight calculation was fixed as the embedding vector associated with the distant supervision label for each bag. At the test stage, all relation query vectors were applied to calculate the posterior probabilities of relations respectively and the relation with the highest probability was chosen as the classification result (Lin et al., 2016). The means and standard deviations of the AUC values given by the whole PR curves of these models are shown in Table 3 for a quantitative comparison. Following (Lin et al., 2016), we also plotted the PR curves of these models in Fig. 2 and 3 with recall smaller than 0.5 for a visualized comparison.

From Table 3, Fig. 2 and Fig. 3, we have the following observations. (1) Similar to the results of previous work (Zeng et al., 2015), PCNNs worked better than CNNs as sentence encoders. (2) When using either CNN or PCNN sentence encoders, *ATT_RA* outperformed *ATT_BL*. It can be attributed to that the *ATT_BL* method only considered the target relation when deriving bag representations at training time, while the *ATT_RA* method calculated intra-bag attention weights using all relation embeddings as queries, which improved the flexibility of bag representations. (3) For both sentence encoders and both intra-bag attention methods, the models with *BAG_ATT* always achieved better performances than the ones without *BAG_ATT*. This result verified the effectiveness of our proposed inter-bag attention

# of Test Sentences	**one**				**two**				**all**			
P@N(%)	100	200	300	mean	100	200	300	mean	100	200	300	mean
(Lin et al., 2016)	73.3	69.2	60.8	67.8	77.2	71.6	66.1	71.6	76.2	73.1	47.4	72.2
(Liu et al., 2017)	84.0	75.5	68.3	75.9	86.0	77.0	73.3	78.8	87.0	84.5	77.0	82.8
CNN+ATT_BL	74.2	68.9	65.3	69.5	77.8	71.5	68.1	72.5	79.2	74.9	70.3	74.8
CNN+ATT_RA	76.8	72.7	67.9	72.5	79.6	73.9	70.7	74.7	81.4	76.3	72.5	76.8
CNN+ATT_BL+BAG_ATT	78.6	74.2	69.7	74.2	82.4	76.2	72.1	76.9	83.0	78.0	74.0	78.3
CNN+ATT_RA+BAG_ATT	79.8	75.3	71.0	75.4	83.2	76.5	72.1	77.3	87.2	78.7	74.9	80.3
PCNN+ATT_BL	78.6	73.5	68.1	73.4	77.8	75.1	70.3	74.4	80.8	77.5	72.3	76.9
PCNN+ATT_RA	79.4	73.9	69.6	74.3	82.2	77.6	72.4	77.4	84.2	79.9	73.0	79.0
PCNN+ATT_BL+BAG_ATT	85.2	**78.2**	71.3	78.2	84.8	**80.0**	74.3	79.7	88.8	83.7	77.4	83.9
PCNN+ATT_RA+BAG_ATT	**86.8**	77.6	**73.9**	**79.4**	**91.2**	79.2	**75.4**	**81.9**	**91.8**	**84.0**	**78.7**	**84.8**

Table 4: P@N values of the entity pairs with different number of test sentences.

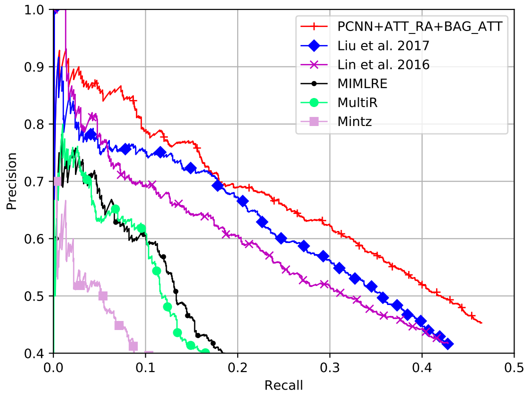


Figure 4: PR curves of several models in previous work and our best model.

method for distant supervision relation extraction. (4) The best AUC performance was achieved by combining PCNN sentence encoders with the intra-bag and inter-bag attentions proposed in this paper.

4.4 Comparison with previous work

4.4.1 PR curves

The PR curves of several models in previous work and our best model *PCNN+ATT_RA+BAG_ATT* are compared in Fig. 4, where Mintz (Mintz et al., 2009), MultiR (Hoffmann et al., 2011) and MIMLRE (Surdeanu et al., 2012) are conventional feature-based methods, and (Lin et al., 2016) and (Liu et al., 2017) are PCNN-based ones[4]. For a fair comparison with (Lin et al., 2016) and (Liu et al., 2017), we also plotted the curves with only the top 2000 points. We can see that our model achieved better PR performance than all the other models.

model	**CNN**	**PCNN**
ATT_BL†	0.219	0.253
ATT_BL+RL	0.229	0.261
ATT_BL+DSGAN	0.226	0.264
ATT_BL‡	0.242	0.271
ATT_RA	0.254	0.297
ATT_BL+BAG_ATT	0.253	0.285
ATT_RA+BAG_ATT	**0.262**	**0.311**

Table 5: AUC values of previous work and our models, where *ATT_BL+DSGAN* and *ATT_BL+RL* are two models proposed in (Qin et al., 2018a) and (Qin et al., 2018b) respectively, † indicates the baseline result reported in (Qin et al., 2018a,b) and ‡ indicates the baseline result given by our implementation.

4.4.2 AUC values

ATT_BL+DSGAN (Qin et al., 2018a) and *ATT_BL+RL* (Qin et al., 2018b) are two recent studies on distant supervision relation extraction with reinforcement learning for data filtering, which reported the AUC values of PR curves composed by the top 2000 points. Table 5 compares the AUC values reported in these two papers and the results of our proposed models. We can see that introducing the proposed *ATT_RA* and *BAG_ATT* methods to baseline models achieved larger improvement than using the methods proposed in (Qin et al., 2018a,b).

4.5 Effects of Intra-Bag Attentions

Following (Lin et al., 2016), we evaluated our models on the entity pairs with more than one training sentence. One, two and all sentences for each test entity pair were randomly selected to construct three new test sets. The P@100, P@200, P@300 values and their means given by our proposed models on these three test sets are reported in Table 4 together with the best results of (Lin et al., 2016) and (Liu et al., 2017). Here, P@N

---

[4] All of these curve data are from `https://github.com/tyliupku/soft-label-RE`.

bag	sentence	correct?	intra-bag weights	inter-bag weights
B1	[**Panama City Beach**]$_{e2}$, too , has a glut of condos , but the area was one of only two in [**Florida**]$_{e1}$ where sales rose in march , compared with a year earlier.	**Yes**	0.71	0.48
	Like much of [**Florida**]$_{e1}$, [**Panama City Beach**]$_{e2}$ has been hurt by the downturn in the real estate market.	**Yes**	0.29	
B2	Among the major rivers that overflowed were the Housatonic , Still , Saugatuck , Norwalk , Quinnipiac , Farmington , [**Naugatuck**]$_{e1}$, Mill , Rooster and [**Connecticut**]$_{e2}$.	**No**	1.00	0.13
B3	..., the army chose a prominent location in [**Virginia**]$_{e1}$, at the foot of the Arlington memorial bridge , directly across the [**Potomac River**]$_{e2}$ from the Lincoln memorial .	**No**	0.13	0.39
	... , none of those stars carried the giants the way barber did at Fedex field , across the [**Potomac River**]$_{e1}$ from [**Virginia**]$_{e2}$, where he grew up as a redskins fan .	**Yes**	0.87	

Table 6: A test set example of relation */location/location/contains* from the NYT corpus.

sentence number	mean±std
1	0.163 ± 0.029
2	0.187 ± 0.033
3	0.210 ± 0.034
4	0.212 ± 0.037
$\geq$ **5**	0.256 ± 0.043

Table 7: The distributions of inter-bag attention weights for the bags with different number of sentences.

means the precision of the relation classification results with the top N highest probabilities in the test set.

We can see our proposed methods achieved higher P@N values than previous work. Furthermore, no matter whether *PCNN* or *BAG_ATT* were adopted, the *ATT_RA* method outperformed the *ATT_BL* method on the test set with only one sentence for each entity pair. Note that the decoding procedures of *ATT_BL* and *ATT_RA* were equivalent when there was only one sentence in a bag. Therefore, the improvements from *ATT_BL* to *ATT_RA* can be attributed to that *ATT_RA* calculated intra-bag attention weights in a relation-aware way at the training stage.

4.6 Distributions of Inter-Bag Attention Weights

We divided the training set into 5 parts according to the number of sentences in each bag. For each bag, the inter-bag attention weights given by the *PCNN+ATT_RA+BAG_ATT* model were recorded. Then, the mean and standard deviation of inter-bag attention weights for each part of the training set were calculated and are shown in Table 7. From this table, we can see that the bag with smaller number of training sentences were usually assigned with lower inter-bag attention weights. This result was consistent with the finding in (Qin et al., 2018b) that the entity pairs with fewer training sentences were more likely to have incorrect relation labels.

4.7 Case Study

A test set example of relation */location/location/contains* is shown in Table 6. The bag group contained 3 bags, which consisted of 2, 1, and 2 sentences respectively. We calculated the intra-bag and inter-bag attentions for this bag group using our *PCNN+ATT_RA+BAG_ATT* model and the weights of the target relation are also shown in Table 6.

In this example, the second bag was a noisy bag because the only sentence in this bag didn't express the relation */location/location/contains* between the two entities **Naugatuck** and **Connecticut**. In conventional methods, these three bags were treated equally for model training. After introducing inter-bag attention mechanism, the weight of this noisy bag was reduced significantly as shown in the last column of Table 6.

5 Conclusion

In this paper, we have proposed a neural network with intra-bag and inter-bag attentions to cope with the noisy sentence and noisy bag problems in distant supervision relation extraction. First, relation-aware bag representations are calculated by a weighted sum of sentence embeddings where the noisy sentences are expected to have smaller weights. Further, an inter-bag attention module is designed to deal with the noisy bag problem by calculating the bag-level attention weights dynamically during model training. Experimental results on New York Times dataset show that our models achieved significant and consistent improvements compared with the models using only conventional intra-bag attentions. To deal with the multi-label problem of relation extraction and to integrate external knowledge into our model will be the tasks of our future work.

Acknowledgments

We thank the anonymous reviewers for their valuable comments.

References

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550. Association for Computational Linguistics.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention. *arXiv preprint arXiv:1804.06987*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Pengda Qin, Weiran XU, and William Yang Wang. 2018a. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505. Association for Computational Linguistics.

Pengda Qin, Weiran XU, and William Yang Wang. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics.

Ranking-Based Autoencoder for Extreme Multi-label Classification

Bingyu Wang
Khoury College of CS
Northeastern University
`rainicy@ccs.neu.edu`

Li Chen
JD.com Inc.
`li.chen@jd.com`

Wei Sun
Department of ECE
North Carolina State University
`wsun12@ncsu.edu`

Kechen Qin
Khoury College of CS
Northeastern University
`qin.ke@husky.neu.edu`

Kefeng Li
JD.com Inc.
`likefeng@jd.com`

Hui Zhou
JD.com Inc.
`hui.zhou@jd.com`

Abstract

Extreme Multi-label classification (XML) is an important yet challenging machine learning task, that assigns to each instance its most relevant candidate labels from an extremely large label collection, where the numbers of labels, features and instances could be thousands or millions. XML is more and more on demand in the Internet industries, accompanied with the increasing business scale / scope and data accumulation. The extremely large label collections yield challenges such as computational complexity, inter-label dependency and noisy labeling. Many methods have been proposed to tackle these challenges, based on different mathematical formulations. In this paper, we propose a deep learning XML method, with a word-vector-based self-attention, followed by a ranking-based AutoEncoder architecture. The proposed method has three major advantages: 1) the autoencoder simultaneously considers the inter-label dependencies and the feature-label dependencies, by projecting labels and features onto a common embedding space; 2) the ranking loss not only improves the training efficiency and accuracy but also can be extended to handle noisy labeled data; 3) the efficient attention mechanism improves feature representation by highlighting feature importance. Experimental results on benchmark datasets show the proposed method is competitive to state-of-the-art methods.

1 Introduction and Related Work

In multi-label classification (Tsoumakas and Katakis, 2007; Zhang and Zhou, 2014), one assigns multiple labels to each instance. Multi-label classification has many real-word applications: for example, a movie may be associated with multiple genres, a web page may contain several topics, and an image can be tagged with a few objects. In these classification tasks, labels often exhibit complex dependencies: for example, *Documentary* and *Sci-Fi* are usually mutually exclusive movie genres, while *Horror* and *Thriller* are typically highly correlated. Predicting labels independently fails to capture these dependencies and suffers suboptimal performance (Tsoumakas and Katakis, 2007; Ghamrawi and McCallum, 2005; Li et al., 2016). Several methods that capture label dependencies have been proposed, including Conditional Random Fields (CRF) (Lafferty et al., 2001; Ghamrawi and McCallum, 2005), Classifier Chains (CC) (Read et al., 2011; Dembczynski et al., 2010), Conditional Bernoulli Mixtures (CBM) (Li et al., 2016), and Canonical Correlated AutoEncoder (C2AE) (Yeh et al., 2017). However, these methods typically only work well on small-to-medium scale datasets.

Extreme multi-label classification (XML) is a multi-label classification task in which the number of instances, features and labels are very large, often on the order of thousands to millions (Zubiaga, 2012; Bhatia et al., 2015). It has numerous real-world applications such as merchandise tagging and text categorization. Although the label vocabulary is large, typically each instance only matches a few labels. The scale of the classification task, the inter-dependent labels, and label sparsity all pose significant challenges for accurate and efficient classification.

Many methods have been proposed for extreme multi-label classification. We group them into different categories and describe representative methods in each category.

Independent Classification: A popular method is to divide the multi-label classification problem into multiple binary classification prob-

Proceedings of NAACL-HLT 2019, pages 2820–2830
Minneapolis, Minnesota, June 2 - June 7, 2019.

lems (Tsoumakas and Katakis, 2007; Hariharan et al., 2012; Babbar and Schölkopf, 2017; Yen et al., 2016, 2017). A typical implementation is to treat labels independently and train one-vs-all classifiers for each of the labels. These independent classifiers can be trained in parallel and thus are computationally efficient in practice. Ignoring the inter-label dependency also enables efficient optimization algorithm, which further reduces computational cost. However, ignoring label dependency inherently limits prediction accuracy. A competitive method in this category is called PD-Sparse (Yen et al., 2016), with a variant of the Block-Coordinate Frank-Wolfe training algorithm that exploits data sparsity and achieves complexity sub-linear in the number of primal and dual variables. PD-Sparse (Yen et al., 2016) shows better performance with less training and prediction time than 1-vs-all Logistic Regression or SVM on extreme multi-label datasets.

Tree Based Classifiers: Following the success of tree-based algorithms in binary classification problems, people also proposed tree-based algorithms for multi-label classification (Agrawal et al., 2013; Weston et al., 2013; Prabhu and Varma, 2014), which achieve promising prediction accuracy. Similar to decision trees, these methods make classification decisions in each branch split. Different from decision trees, each split evaluates all features, instead of one, to make a decision. Also, each decision is for a subset of labels rather than one label. Finally, via ensembling and parallel implementation, trees can boost their prediction accuracy with practically affordable computational cost. Among these tree based classifiers, FastXML (Prabhu and Varma, 2014) further optimizes an nDCG-based ranking loss function and achieves significantly higher accuracy than other peer methods.

Embedding: A major difficulty of extreme multi-label classification is the large number of labels. When labels are inter-dependent, one can attempt to find a lower dimensional latent label space from which one can fully reconstruct the original label space. Over the past decade, many methods were proposed to find this latent label space. In early work, methods were proposed to linearly project the original label space into a lower-dimension space and reconstruct predictions from that space (Tai and Lin, 2012; Balasubramanian and Lebanon, 2012). However, there are two assumptions: (1) the label dependency is linear and (2) the label matrix is low-rank, which do not always hold, as reflected by the low prediction accuracy of these methods. To overcome the limitation of the linear assumption, different methods were proposed using non-linear embeddings, including kernels, sub-sampling (Yu et al., 2014), feature-aware (Lin et al., 2014; Yeh et al., 2017) and pairwise distance preservation (Bhatia et al., 2015). Among these methods, SLEEC (Bhatia et al., 2015) stands out for less training time and higher accuracies. SLEEC introduces a method for learning a small ensemble of local pairwise distance preserving embeddings which allows it to avoid the low-rank and linear-dependency assumption.

Deep Learning: Deep learning has not been well studied for XML, although it has achieved great successes in binary and multi-class classification problems (Lin et al., 2017; Kim, 2014).

FastText (Grave et al., 2017) reconstructs a document representation by averaging the embedding of the words in the document, followed by a softmax transformation. It is a simple but very effective and accurate multi-class text classifier, as demonstrated in both sentiment analysis and multi-class classification (Grave et al., 2017). However, FastText may not be directly applicable for more complicated problems, like XML.

BoW-CNN (Johnson and Zhang, 2014) learns powerful embedding of small text regions by applying CNN to high-dimensional text data. The embedding of all regions are sent to one or multiple convolutional layers, a pooling layer and the output layer at the end.

XML-CNN (Liu et al., 2017) achieves computational efficiency by training a deep neural network with a hidden bottleneck layer much smaller than the output layer. However, this method has a few drawbacks. First, it is trained using the binary cross entropy loss. This loss tends to be sensitive to label noise, which is frequently observed in extreme multi-label data. Since the label vocabulary is large, it is quite common for human annotator to miss relevant tags. When the classifier's prediction (which might be correct) disagrees with the annotation, the cross entropy loss can potentially assign an unbounded penalty to the classifier during training procedure. The second issue is that because labels are trained independently as separate binary classification tasks, their prediction probabilities/scores may not be directly comparable. This is problematic because in many applications the requirement is to rank all labels according to their relevance, as opposed to making an independent binary decision on each label. The third defect is that XML-CNN requires raw documents as input since it adopts the CNN structure on top of sentences (Kim, 2014); this is problematic when datasets are given in other formats such as bag-of-words for text.

C2AE (Yeh et al., 2017) uses a ranking loss as the training objective. But the ranking loss employed there needs to compare all (positive label, negative label) pairs, and therefore does not scale well to extreme data. Furthermore, C2AE only takes the bag-of-words representation (one-hot encoding) as the input, which makes it harder to learn powerful representations from extreme multi-label dataset.

Our Contribution In this paper, we propose a new deep learning method to address extreme multi-label classification. Our contributions are as follows:

- Motivated by the recent success of attention techniques, we propose an efficient attention mechanism that can learn rich representations from any type of input features, including but not limited to bag-of-words, raw documents and images.
- Inspired by C2AE, our proposed model projects both features and labels onto common latent spaces wherein correlations between features and labels are exploited. By decoding this latent space into the original label space in the prediction stage, the dependencies between labels are implicitly captured.
- We propose a margin-based ranking loss that is simultaneously more effective for extreme settings and more tolerant towards noisy labeling.

2 The Proposed Method (Rank-AE)

In this section, we introduce the data format in XML, and the proposed networks, including the marginal-based ranking loss and attentions.

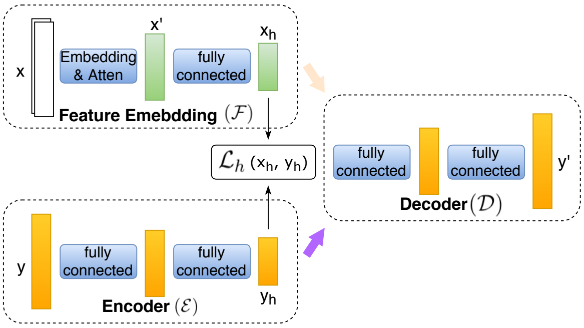


Figure 1: Ranking-based AutoEncoder for XML.

2.1 Data Format

In XML, we are given a set of label candidates $\mathcal{Y} = \{1, 2, \ldots, L\}$. The dataset D consists of features and labels: $D = \{(x_i, y_i)\}_{i=1}^N$, wherein N is number of data, and each instance $x \in \mathbb{R}^V$ (V is the feature dimension) matches a label subset $y \subseteq \mathcal{Y}$, which can be written as a binary vector $y = \{0, 1\}^L$, with each bit y_l representing the presence or absence of label l. Given such dataset, our goal is to build a classifier c: $\mathbb{R}^V \rightarrow \{0, 1\}^L$, mapping an instance to a subset of labels with arbitrary size.

2.2 Auto-Encoder Network

Inspired by the C2AE (Yeh et al., 2017), we propose a Ranking-based Auto-Encoder (**Rank-AE**), as depicted in Figure 1. Similar to C2AE, Rank-AE includes three mapping functions to be trained: a mapping from input features x to feature embeddings x_h, denoted as $\mathcal{F}(x)$, where h is the embedding size; an encoder from output labels y to label embeddings y_h as $\mathcal{E}(y)$; a decoder from label embeddings y_h to output labels y', written as $\mathcal{D}(y_h)$. The proposed model is built on two assumptions: first, each instance can be represented from two different aspects, features x and labels y, so there exists a common latent space between x and y; second, labels can be reproduced by an autoencoder. Based on these two assumptions, we design the object function as below:

$$\mathcal{L}(D) = \min_{\mathcal{F},\mathcal{E},\mathcal{D}} \mathcal{L}_h(x_h, y_h) + \lambda \mathcal{L}_{ae}(y, y') \quad (1)$$

wherein loss $\mathcal{L}_h(x_h, y_h)$ aims to find the common latent space for input x and output y and $\mathcal{L}_{ae}(y, y')$ enforces the output to be reproducible. λ is a hyper-parameter to balance these two losses. During the training, the model learns a joint network including $\mathcal{F}, \mathcal{E}$ and $\mathcal{D}$ to minimize the empirical loss Eq (1).

During inference, a given input $\hat{x}$ will be first transformed into a vector in latent space $\hat{x}_h = \mathcal{F}(\hat{x})$, which will then be fed into the label decoder to compute the predictions $\hat{y} = \mathcal{D}(\hat{x}_h)$. It is worth mentioning that although the label encoder $\mathcal{E}$ is ignored during the prediction, it is able to exploit cross-label dependency during the label embedding stage (Yeh et al., 2017). Recent work (Kurata et al., 2016; Baker and Korhonen, 2017) also show that using co-occurring labels information to initialize the neural network can further improve accuracy in multi-label classification.

2.3 $\mathcal{L}_h$ and $\mathcal{L}_{ae}$ Loss Functions

Learning Common Embedding ($\mathcal{L}_h$). Minimizing the common hidden space loss $\mathcal{L}_h$ has been proposed based on different considerations (Zhang and Schneider, 2011; Yeh et al., 2017; Shen et al., 2018), ranging from canonical correlation analysis to alignment of two spaces with a perspective of cross-view. Since the hidden space is usually small and requires less computational cost, we simply employ the mean squared loss for $\mathcal{L}_h$.

Reconstructing Output ($\mathcal{L}_{ae}$). Unlike $\mathcal{L}_h$ with small space, $\mathcal{L}_{ae}$ loss usually involves a large number of labels. Moreover, $\mathcal{L}_{ae}$ also directly affects the classification performance significantly since different loss functions lead to their own properties (Hajiabadi et al., 2017). Accordingly, solving such problems with large scale and desirable properties presents open challenges in three aspects: 1) how to improve time efficiency, 2) how to produce comparable labels scores and 3) how to deal with noise labels. Unfortunately, most of the related deep learning methods only target one or two aspects. C2AE attempts to minimize the number of misclassified pairs between relevant and irrelevant labels, as a result its computational complexity is quadratic with number of labels in the worst case; also it fails to scale well on large number of input features or labels due to its inefficient implementation[1]. XML-CNN (Liu et al., 2017) achieves computational efficiency by training a deep neural network with hidden layers much smaller than the output layer with binary cross-entropy loss (BCE), which has linear complexity in number of labels. Despite this, BCE loss could neither capture label dependencies nor produce directly comparable label scores, since each label is treated independently. Moreover, BCE loss tends to be sensitive to label noise, which is frequently observed in XML data (Reed et al., 2014; Ghosh et al., 2017).

To void the aforementioned issues, we propose a marginal-based ranking loss in AutoEncoder:

$$\mathcal{L}_{ae}(y, y') = \mathcal{L}_P(y, y') + \mathcal{L}_N(y, y') \tag{2}$$

$$\mathcal{L}_P(y, y') = \sum_{n \in N(y)} \max_{p \in P(y)} (m + y'_n - y'_p)_+ \tag{3}$$

$$\mathcal{L}_N(y, y') = \sum_{p \in P(y)} \max_{n \in N(y)} (m + y'_n - y'_p)_+ \tag{4}$$

wherein $N(y)$ is the set of negative label indexes, $P(y)$ is the complement of $N(y)$, and margin $m \in [0, 1]$ is a hyper-parameter for controlling the minimal distance between positive and negative labels. The loss consists of two parts: 1) $\mathcal{L}_P$ targets to raise the minimal score from positive labels over all negative labels at least by m; 2) $\mathcal{L}_N$ aims to penalize the most violated negative label under all positive labels by m. The proposed loss has the following attractive properties: 1) having linear complexity in number of labels $\mathcal{O}(L)$; 2) capturing the relative rankings between positive and negative labels; 3) tolerating the noisy labels with a tunable hyper-parameter m. To explain the last property, assume y'_n and y'_p are the predicted probabilities bounded in $[0, 1]$, then on one extreme case with noise-free labels and $m = 1$, all positive labels are raised to probability 1, while negatives are penalized to 0; on the other extreme case with all random labels, e.g. from i.i.d. Bernoulli distribution, setting $m = 0$ indicates that the annotated labels are completely random noises.

2.4 Dual Attention

Extracting rich feature representations in XML is helpful for predicting more complicated labels structures, but on the other hand, requires an efficient and feasible method. A recent work (CBAM) (Woo et al., 2018) proposes a block attention module, with a Channel-Attention and a Spatial Attention for images tasks only, wherein Channel-Attention emphasizes information from channels, e.g. RBG, and Spatial-Attention pays attention to partial areas in an image. By sequentially applying channel and spatial attentions, CBAM is shown to be effective in images classification and object detection. We take advantage of the attentions in CBAM and apply it on textual data.

---

[1] `https://github.com/dhruvramani/C2AE-Multilabel-Classification`

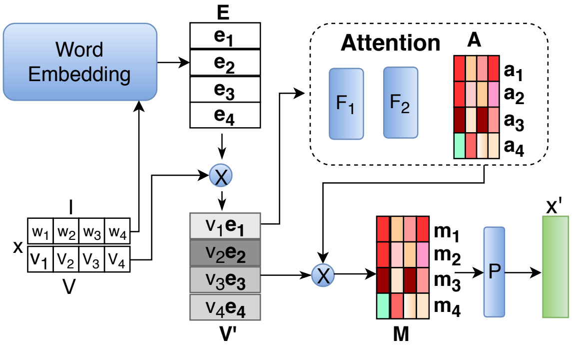


Figure 2: Dual-Attention in Feature Embedding.

In our proposed attention module, it also consists of spatial-wise and channel-wise attentions. First, we force spatial-wise attention to attend on a list of important words in a way that simply multiply word embeddings by term-frequency or tf-idf (whichever is provided in the feature matrix). It is worth noting that spatial-wise module does not involve any parameters, but it efficiently captures the importance of words with numerical statistics, like tf-idf. We demonstrate spatial-wise attention on the left side of Figure 2, where the input $x = (I, V)$ contains bag-of-words vector $I = (w_1, w_2, \ldots, w_n)^T \in \mathbb{R}^n$ and tf-idf vector $V = (v_1, v_2, \ldots, v_n)^T \in \mathbb{R}^n$. Bag-of-words I are fed into an embedding layer $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)^T \in \mathbb{R}^{n\times C}$ to get the word embeddings, where $\mathbf{e}_j \in \mathbb{R}^C$ is word embedding vector of w_j. Then we multiply word embeddings by V to obtain weighted word embeddings: $\mathbf{V}' = (v_1\mathbf{e}_1, v_2\mathbf{e}_2, \ldots, v_n\mathbf{e}_n)^T \in \mathbb{R}^{n\times C}$.

The channel attention is designed to emphasize the significant aspects by assigning different weights on bits in a word embedding. For example, in the word embedding of "apple", some of the bits may reflect fruit, while others may indicate the company name. To achieve this, we adopt the excitation network from the SENet (Hu et al., 2017) with a slight increase in model complexity. The excitation network includes two fully connected layers with a non-linearity activation function in between, see the top-right part in Figure 2:

$$\mathbf{A}^T = \sigma\left(F_2\delta(F_1\mathbf{V}'^T)\right) = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n) \quad (5)$$

wherein $\mathbf{A} \in \mathbb{R}^{n\times C}$, δ and σ refer to the two activation functions, *ReLU* and *Sigmoid*, $F_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $F_2 \in \mathbb{R}^{C\times\frac{C}{r}}$ are the two fully connected layers, with word embedding size C and reduction ratio r. After obtaining the attention matrix $\mathbf{A}$, we can apply those attentions to the weighted word embeddings to get a re-scaled word embedding matrix $\mathbf{M}$: $\mathbf{M} = \mathbf{V}' \circ \mathbf{A} = (\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n)^T \in \mathbb{R}^{n\times C}$, which is computed via the element-wise product. The obtained attention matrix $\mathbf{A}$ introduces dynamics conditioned on the input weighted word embeddings and further boosts feature discriminability. The last step is to feed the re-scaled embedding matrix into an average pooling to obtain the feature embedding $x' \in \mathbb{R}^C$. With the proposed spatial-wise and channel-wise attentions, Rank-AE can learn rich feature representations in an efficient way.

Dataset	train	test	label	feature	cardinality	inst/L
Delicious*	12,920	3,185	983	500	19.03	312
Mediamill*	30,993	12,914	101	120	4.38	1,902
RCV*	623,847	155,962	2,456	47,236	4.79	1,219
IMDb	27,417	6,740	28	115,554	2.4	2,937
EURLex	15,539	3,909	3,993	5,000	5.31	26
Wiki10	14,146	6,616	30,938	101,938	18.64	9

Table 1: Dataset Characteristics: train, test, label and feature are the numbers of training, testing, labels and features respectively; cardinality is the average number of label per instance; inst/L is the average number of instances per label. * indicates only feature matrix is provided and no raw document.

3 Experiments & Analysis

3.1 Dataset & Experiment Setup

Dataset. Our experiments are conducted on six extreme multi-label datasets and their characteristics are shown in Table 1, among which IMDb is crawled from online movie database[2] and the rest five datasets are downloaded from the extreme classification repository[3]. For datasets from the repository, we adopt the provided train/test split, and for IMDb we randomly choose 20% of the data as test set and the rest of 80% as training set. For all datasets, we reserve another 20% of training data as validation for tuning hyper-parameters. After tuning, all models are trained on the entire training set.

Among these datasets, three of them are only provided with BoW feature matrix: Delicious, Mediamill (dense feature matrix extracted from image data) and RCV, which are only feasible for the non-deep learning methods (SLEEC, FastXML, PDSparse) and Rank-AE. We provide

---

[2] `https://www.imdb.com/`

[3] `http://manikvarma.org/downloads/XC/XMLRepository.html`

both feature matrix and raw documents for IMDb, EURLex and Wiki10, which are feasible for both deep learning and non-deep learning methods. For those data with both formats, we remove the words from the raw documents that do not have corresponding BoW features so that the vocabulary size is the same for both deep and non-deep learning methods.

Evaluation Metrics. To evaluate the performances of each model, we adopt the metrics that have been widely used in XML: Precision at top k ($P@k$), and the Normalized Discounted Cummulated Gains at top k ($n@k$) (Bhatia et al., 2015; Prabhu and Varma, 2014; Yen et al., 2016; Liu et al., 2017). $P@k$ is a measure based on the fraction of correct predictions in the top k predicted scoring labels and $n@k$ is a normalized metric for Discounted Cumulative Gain:

$$P@k = \frac{1}{k} \sum_{l \in rank_k(\hat{y})} y_l \tag{6}$$

$$DCG@k = \sum_{l \in rank_k(\hat{y})} \frac{y_l}{log(l+1)} \tag{7}$$

$$nDCG@k = \frac{DCG@K}{\sum_{l=1}^{\min(k,|y|)} \frac{1}{\log(l+1)}} \tag{8}$$

wherein the $rank_k$ returns k largest indices of the prediction $\hat{y}$ in a descending order, and $|y|$ is the number of positive labels in ground truth. In the results, we report the average $P@k$ and $n@k$ on testing set with $k = 1, 3, 5$ respectively.

Hyper-parameters. In Rank-AE, we use the fixed neural network architecture, with two fully connected layers in both **Encoder** and **Decoder**, and one fully connected layer following **Embedding & Atten** network in **Feature Embedding**. We also fix most of the hyper-parameters, including hidden dimension h (100 for small number of labels data and 200 for large ones), word embedding size $C = 100$, and reduction ratio $r = 4$. The remaining hyper-parameters, such as balance λ between $\mathcal{L}_h$ and $\mathcal{L}_{ae}$, margin m in $\mathcal{L}_{ae}$, and others (decay, learning rate) in the optimization algorithms, are tuned on validation set. In addition, if the vocabulary for BoW is available, e.g. IMDb and Wiki10, the **Word Embedding** component is initialized by Glove[4], a pre-trained word embeddings of 100 dimensions; if it is not, e.g. Mediamill, Delicious and RCV, a random initialization is employed.

For the existing methods with the same train/test split, we take the scores from the original papers for SLEEC, FastXML and PD-Sparse directly. For the new datasets and splits, the hyper-parameters are tuned on the validation set for all methods, as suggested in their papers.

3.2 Comparisons with Related Methods

We evaluate the proposed Rank-AE with other six state-of-the-art methods, SLEEC, FastXML, PD-Sparse, FastText, Bow-CNN and XML-CNN, which are the leading methods among their categories. Among them, FastText, Bow-CNN and XML-CNN only take raw documents, which are not available for Delicious, Mediamill and RCV datasets. For Rank-AE, we adopt the raw text as the input for IMDb, and feature matrix for the rest.

The performances evaluated on $P@k$ and $n@k$ with $k = 1, 3, 5$ are summarized in Table 2 (a) and (b) separately. As reported, Rank-AE reaches the best performances on two datasets (IMDb and EURLex) out of 6 datasets, while SLEEC achieves the best performances on Mediamill and Wiki10, and FastXML performs the best on Delicious and RCV. In general, SLEEC and FastXML are very competitive to each other in non-deep learning methods, but PD-Sparse performs worse. Rank-AE always performs better than PD-Sparse with at least 1% increase, up to almost 20% improvement on Delicious data. When compared with FastXML, Rank-AE outperforms on 4 datasets with 1% to 10% growth, but underperforms on Delicious and RCV with 1% decrease. SLEEC, as the best non-deep learning method in our experiments, performs almost identical to Rank-AE, but on IMDb data, it performs $7\% \sim 15\%$ less than non-deep methods, and even worse than Rank-AE.

Comparing Rank-AE with deep learning methods, we narrow down to three datasets with available raw documents: IMDb, EURLex and Wiki10. As shown in Table 2, FastText and Bow-CNN, not planned for XML but for multi-class, perform much worse than XML-CNN and Rank-AE as expected. On the other hand, XML-CNN achieves close performance to Rank-AE: with similar performance on IMDb dataset, but lower scores on EURLex and Wiki10 with 2% drop in $P@k$ and $n@k$. In spite of this, Rank-AE, trained on feature matrix for EURLex and Wiki10, surprisingly performs better than XML-CNN on raw data.

In the comparisons, there is no such method that

---

[4] `https://nlp.stanford.edu/projects/glove/`

Dataset	Metrics	SLEEC	FastXML	PD-Sparse	FastText	Bow-CNN	XML-CNN	Rank-AE
Delicious	P@1	67.59	**69.61**	51.82	-	-	-	69.26
	P@3	61.38	**64.12**	44.18	-	-	-	62.72
	P@5	56.56	**59.27**	38.95	-	-	-	57.63
Mediamill	P@1	**87.82**	84.22	81.86	-	-	-	86.53
	P@3	**73.45**	67.33	62.52	-	-	-	70.17
	P@5	**59.17**	53.04	45.11	-	-	-	55.44
RCV	P@1	90.25	**91.23**	90.08	-	-	-	90.9
	P@3	72.42	**73.51**	72.03	-	-	-	72.82
	P@5	51.88	**53.31**	51.09	-	-	-	52.05
IMDb	P@1	51.37	66.45	66.84	69.55	66.59	75.55	**75.91**
	P@3	34.46	48.32	46.29	48.76	48.42	52.59	**52.66**
	P@5	27.34	36.28	35.04	36.53	36.56	**38.90**	38.48
EURLex	P@1	79.26	71.36	76.43	71.51	64.99	76.38	**79.52**
	P@3	64.30	59.90	60.37	60.37	51.68	62.81	**65.14**
	P@5	52.33	50.39	49.72	50.41	42.32	51.41	**53.18**
Wiki10	P@1	**85.88**	83.03	81.03	68.86	81.16	84.11	83.6
	P@3	**72.98**	67.47	57.36	54.65	50.67	70.24	72.07
	P@5	**62.70**	57.76	44.10	47.61	36.03	59.87	62.07
Rank Score	avg	2.83	3.33	4.56	4.56	5.78	2.56	**1.78**

Dataset	Metrics	SLEEC	FastXML	PD-Sparse	FastText	Bow-CNN	XML-CNN	Rank-AE
Delicious	n@1	67.59	**69.61**	51.82	-	-	-	69.26
	n@3	62.87	**65.47**	46.00	-	-	-	64.16
	n@5	59.28	**61.90**	42.02	-	-	-	60.39
Mediamill	n@1	**87.82**	84.22	81.86	-	-	-	86.53
	n@3	**81.50**	75.41	70.21	-	-	-	78.36
	n@5	**79.22**	72.37	63.71	-	-	-	75.28
RCV	n@1	90.25	**91.23**	90.08	-	-	-	90.9
	n@3	88.86	**89.63**	88.50	-	-	-	89.29
	n@5	89.49	**90.33**	88.79	-	-	-	89.75
IMDb	n@1	51.37	66.45	66.84	69.55	66.59	75.55	**75.91**
	n@3	49.75	67.14	64.84	68.47	67.26	**74.02**	73.5
	n@5	54.43	71.72	69.69	72.99	72.07	**78.48**	77.37
EURLex	n@1	79.26	71.36	76.43	71.51	64.99	76.38	**79.52**
	n@3	68.13	62.87	64.31	63.32	55.03	66.28	**68.76**
	n@5	61.60	58.06	58.78	58.56	49.92	60.32	**62.33**
Wiki10	n@1	**85.88**	83.03	81.03	68.86	81.16	84.11	83.6
	n@3	**76.02**	75.35	62.62	56.72	56.14	73.52	74.78
	n@5	**68.13**	63.36	52.03	51.19	45.29	65.50	67.18
Rank Score	**avg**	2.83	3.22	4.33	4.67	5.78	2.56	**1.89**

Table 2: Comparisons with other methods ($P@k$ and $n@k$ are reported in the top and bottom tables respectively). '-' indicates unavailable due to raw documents are not available for these deep learning methods, and number in bold is the best result in the line.

could perform the best on all datasets. We discover that each dataset has its own intrinsic properties, such as diversity of labels, number of features, average number of relevant labels per instance and average number of training instances per label, see Table 1. All those properties will affect training procedure, for example, how much flexibility a model should be in order to explain labels well by the given training data. Because those factors are always changing from data to data, they also influence the performances on different models. In order to have a reasonable comparisons, we report the average ranking score for each method. To compute the average ranks, we first rank the methods based on their performance in each row in Table 2, then average them through all rows, and report the final ranking scores in the last row of each table. The average ranking scores show that Rank-AE is the best model with ranking scores 1.78 in $P@k$ and 1.89 in $n@k$.

3.3 Comparisons with Noise Labels

As mentioned previously, noisy labels in XML are a quite common issue in the real-world applications (Yeh et al., 2017; Ghosh et al., 2017), but our proposed marginal ranking loss naturally mitigates this problem. Since IMDb is a real-world dataset with relatively clean labels, we conduct the noise experiments on it. In the experiments, we control the noise labels in two different ways: 1) missing labels: changing each positive label from $y_l = 1$ to $y_l = 0$ with certain rate, 2) both

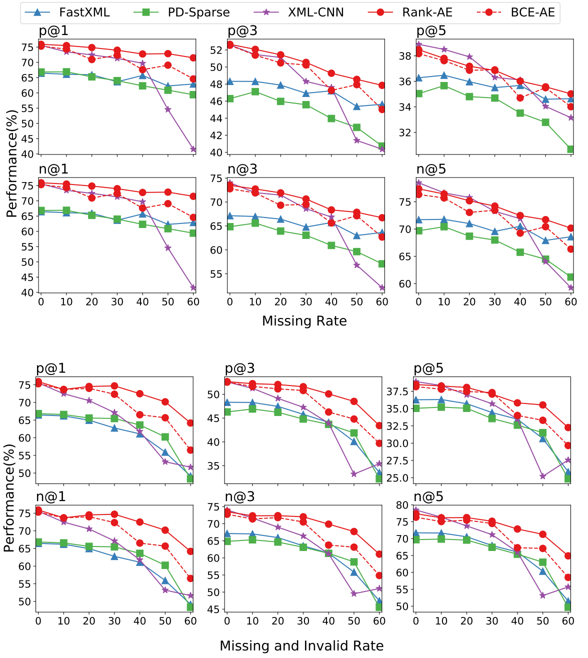


Figure 3: Comparisons on noisy labelling IMDb data.

missing and invalid labels: flipping either from positive to negative or from negative to positive with a noise rate. The noise rates are varied from 0% to 60% on 80% of the training set, and the rest of 20% is noise-free validation set for model selection. We select five algorithms: FastXML, PD-Sparse, XML-CNN, Rank-AE and BCE-AE, wherein BCE-AE is our proposed method but using binary cross-entropy loss in $\mathcal{L}_{ae}(y, y')$. Comparing BCE-AE with Rank-AE can be used to verify whether the robustness to label noise is due to the use of marginal ranking loss.

The performances are reported on the same clean test set, shown in Figure 3. Rank-AE consistently outperforms other four approaches and has the best robustness tolerating noise labels. Besides, FastXML and PD-Sparse are more tolerant to missing noises than XML-CNN, which may due to XML-CNN has greater capacity and thus more prone to over-fitting the noise. Furthermore, when comparing Rank-AE with BCE-AE, both of which share the same structure but have different loss functions, the proposed marginal-based ranking loss seems to be robuster than binary cross-entropy loss.

3.4 More Analysis in Rank-AE

Ablation Study. The effectiveness and robustness of Rank-AE have been demonstrated in the previous section. However, it is not clear to us yet that if the effectiveness benefits from the proposed components, such as attention mechanism and marginal ranking loss. To further understand the impacts from these two factors, we conduct a controlling experiment with three different settings: 1) removing the Attention component $\mathcal{A}$ in Figure 2 from Rank-AE, in which case $\mathbf{V}'$ is directly passed to the average pooling to obtain x', called **No attn**; or 2) examining the performances by replacing the marginal ranking loss ($\mathcal{L}_{ae}$) with a binary cross entropy loss, named **No loss**; or 3) keeping the original **Rank-AE** without any change. In Figure 4, $P@k$ is reported on the six datasets for the ablation experiment, because $n@k$ is similar to $P@k$, thus eliminated here. The comparisons results show that Rank-AE without any change works better than the other two on all datasets consistently, especially on Wiki10. First, channel-attention extracts richer information from the word embeddings by introducing the channel weights. Thus, it is more suitable when classification tasks become more complicated and a word more likely represents multiple aspects. Second, Rank-AE gains some advantage of tolerating noise labels with marginal ranking loss comparing to BCE loss. We could even further infer that IMDb and RCV may have relatively less noise labels since the performance does not benefit much from the marginal ranking loss.

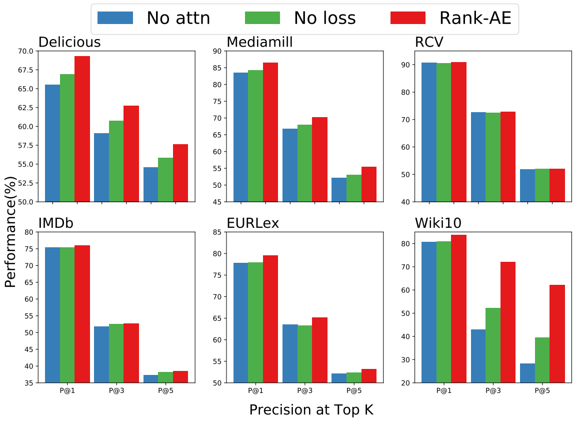
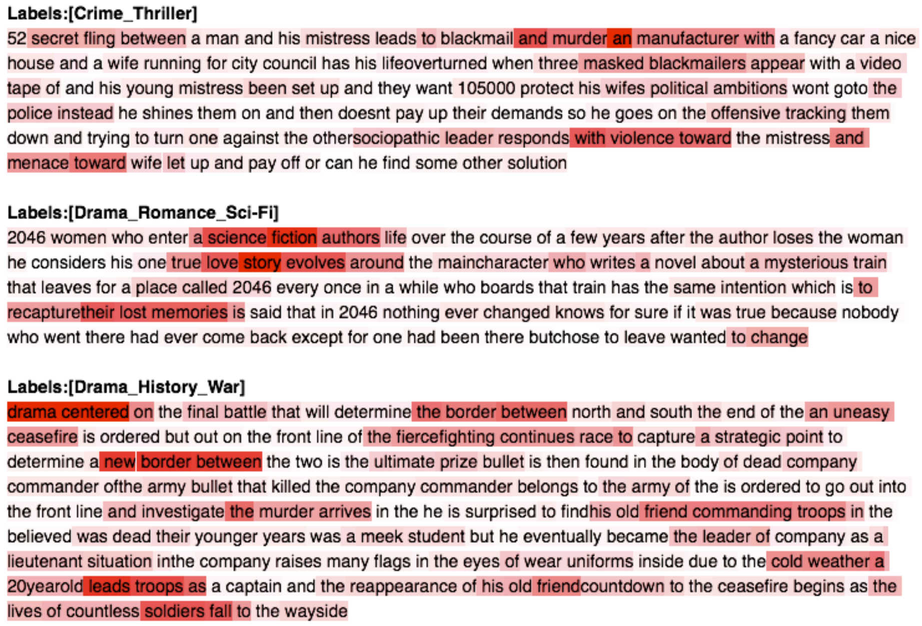


Figure 4: Precision at top k comparisons: No attn (/) is no attention Rank-AE; No loss (+) is using binary cross entropy lose instead of marginal ranking loss; Rank-AE (x) is our proposed model.

Channel-Attention Visualization. Our channel-attention is implemented by an excitation network, which is adopted from SENet (Hu et al., 2017) and only applied to images before. To demonstrate its effectiveness and feasibility on textual data, we



Labels:[Crime_Thriller]

52 secret fling between a man and his mistress leads to blackmail and murder an manufacturer with a fancy car a nice house and a wife running for city council has his lifeoverturned when three masked blackmailers appear with a video tape of and his young mistress been set up and they want 105000 protect his wifes political ambitions wont goto the police instead he shines them on and then doesnt pay up their demands so he goes on the offensive tracking them down and trying to turn one against the othersociopathic leader responds with violence toward the mistress and menace toward wife let up and pay off or can he find some other solution

Labels:[Drama_Romance_Sci-Fi]

2046 women who enter a science fiction authors life over the course of a few years after the author loses the woman he considers his one true love story evolves around the maincharacter who writes a novel about a mysterious train that leaves for a place called 2046 every once in a while who boards that train has the same intention which is to recapturetheir lost memories is said that in 2046 nothing ever changed knows for sure if it was true because nobody who went there had ever come back except for one had been there butchose to leave wanted to change

Labels:[Drama_History_War]

drama centered on the final battle that will determine the border between north and south the end of the an uneasy ceasefire is ordered but out on the front line of the fiercefighting continues race to capture a strategic point to determine a new border between the two is the ultimate prize bullet is then found in the body of dead company commander ofthe army bullet that killed the company commander belongs to the army of the is ordered to go out into the front line and investigate the murder arrives in the he is surprised to findhis old friend commanding troops in the believed was dead their younger years was a meek student but he eventually became the leader of company as a lieutenant situation inthe company raises many flags in the eyes of wear uniforms inside due to the cold weather a 20yearold leads troops as a captain and the reappearance of his old friendcountdown to the ceasefire begins as the lives of countless soldiers fall to the wayside

Figure 5: Visualization for Attention in Rank-AE. The labels are movie genres split by underline, and the input is a movie story line.

employ the visualization tool (Lin et al., 2017) to highlight important words based on the attention output. Specifically, we run our method on IMDb dataset, wherein each instance is a movie story associated with relevant genres as labels. Instead of extracting $\mathbf{V}'$ matrix using the proposed spatial-wise attention, we obtain a fixed size embeddings from a bidirectional LSTM on variable length of sentence, fed to our channel-attention network. Through the channel-attention network, we can observe the attention matrix $\mathbf{A}$ for each input document. By summing up the attention weights of each word embedding vector, we can visualize the overall attention for that word with the visualization tool[5]. We randomly select three movies from IMDb testing set (See Figure 5). By looking at the highlighted regions, we can see that the proposed channel-attention is able to focus more on the words that are highly related to the topics.

4 Conclusion

In this paper, we propose a marginal ranking loss, which not only predicts comparable labels scores between labels, more suitable for ranking metrics, but also consistently performs better on noisy labeling data, with both missing and invalid labels. In addition, the dual-attention component allows Rank-AE to learn more powerful feature representations efficiently. By integrating those components, Rank-AE usually achieves the best or the second best on six benchmark XML datasets comparing with other outstanding methods in state-of-the-art.

Acknowledgements

Most of this work was done when Bingyu Wang and Wei Sun were interning at JD.Com Inc. We thank Javed Aslam, Virgil Pavlu and Cheng Li from Khoury College of Computer Sciences at Northeastern University for comments that greatly improved the manuscript. We would also like to show our gratitude to our anonymous NAACL-HLT reviewers for the helpful suggestions to make the paper better.

References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd*

---

[5] The visualization tool is provided by `https://github.com/kaushalshetty/Structured-Self-Attention`

international conference on World Wide Web, pages 13–24. ACM.

Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM.

Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. *BioNLP 2017*, pages 307–315.

Krishnakumar Balasubramanian and Guy Lebanon. 2012. The landmark selection method for multiple output prediction. *arXiv preprint arXiv:1206.6479*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hullermeier. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, volume 10, pages 279–286.

Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 3–7.

Hamideh Hajiabadi, Diego Molla-Aliod, and Reza Monsefi. 2017. On extending neural networks with loss ensembles for text classification. *arXiv preprint arXiv:1711.05170*.

Bharath Hariharan, SVN Vishwanathan, and Manik Varma. 2012. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine learning*, 88(1-2):127–155.

Jie Hu, Li Shen, and Gang Sun. 2017. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*.

Cheng Li, Bingyu Wang, Virgil Pavlu, and Javed Aslam. 2016. Conditional bernoulli mixtures for multi-label classification. In *International Conference on Machine Learning*, pages 2482–2491.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. 2014. Multi-label classification via feature-aware implicit label space encoding. In *International conference on machine learning*, pages 325–333.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Xiaobo Shen, Weiwei Liu, Ivor Wai-Hung Tsang, Quansen Sun, and Yew-Soon Ong. 2018. Compact multi-label learning. In *AAAI*.

Farbound Tai and Hsuan-Tien Lin. 2012. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label partitioning for sublinear ranking. In *International Conference on Machine Learning*, pages 181–189.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. Cbam: Convolutional block attention module. In *Proc. of European Conf. on Computer Vision (ECCV)*.

Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning deep latent space for multi-label classification. In *AAAI*, pages 2838–2844.

Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM.

Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pdsparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077.

Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. 2014. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Yi Zhang and Jeff Schneider. 2011. Multi-label output codes using canonical correlation analysis. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 873–882.

Arkaitz Zubiaga. 2012. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*.

Posterior-regularized REINFORCE for Instance Selection in Distant Supervision

Qi Zhang[1], **Siliang Tang**[1]*, **Xiang Ren**[3], **Fei Wu**[1], **Shiliang Pu**[2] **& Yueting Zhuang**[1]
[1]Zhejiang University, [2]Hikvision, [3]University of Southern California,
{zhangqihit,siliang,wufei,yzhuang}@zju.edu.cn,
pushiliang@hikvision.com,
xiangren@usc.edu

Abstract

This paper provides a new way to improve the efficiency of the REINFORCE training process. We apply it to the task of instance selection in distant supervision. Modeling the instance selection in one bag as a sequential decision process, a reinforcement learning agent is trained to determine whether an instance is valuable or not and construct a new bag with less noisy instances. However unbiased methods, such as REINFORCE, could usually take much time to train. This paper adopts posterior regularization (PR) to integrate some domain-specific rules in instance selection using REINFORCE. As the experiment results show, this method remarkably improves the performance of the relation classifier trained on cleaned distant supervision dataset as well as the efficiency of the REINFORCE training.

1 Introduction

Relation extraction is a fundamental work in natural language processing. Detecting and classifying the relation between entity pairs from the unstructured document, it can support many other tasks such as question answering.

While relation extraction requires lots of labeled data and make methods labor intensive, (Mintz et al., 2009) proposes distant supervision (DS), a widely used automatic annotating way. In distant supervision, knowledge base (KB) , such as Freebase, is aligned with nature documents. In this way, the sentences which contain an entity pair in KB all express the exact relation that the entity pair has in KB. We usually call the set of instances that contain the same entity pair a bag. In this way, the training instances can be divided into N bags $\mathbf{B} = \{B^1, B^2, ..., B^N\}$. Each bag B^k are corresponding to an unique entity pair $E^k = (e^k_1, e^k_2)$ and contains a sequence of instances $\{x^k_1, x^k_2, ..., x^k_{|B^k|}\}$. However, distant supervision may suffer a wrong label problem. In other words, the instances in one bag may not actually have the relation.

To resolve the wrong label problem, just like Fig.2 shows, (Feng et al., 2018) model the instance selection task in one bag B^k as a sequential decision process and train an agent $\pi(a|s, \theta_\pi)$ denoting the probability $P_\pi(A_t = a, |S_t = s, \theta_t = \theta_\pi)$ that action a is taken at time t given that the agent is in state s with parameter vector θ_π by REINFORCE algorithm (Sutton and Barto, 1998). The action a can only be 0 or 1 indicating whether an instance x^k_i is truly expressing the relation and whether it should be selected and added to the new bag $\overline{B^k}$. The state s is determined by the entity pair corresponding to the bag, the candidate instance to be selected and the instances that have already been selected. Accomplishing this task, the agent gets a new bag $\overline{B^k}$ at the terminal of the trajectory with less wrong labeled instances. With the newly constructed dataset $\overline{\mathbf{B}} = \{\overline{B^1}, \overline{B^2}, ..., \overline{B^N}\}$ with less wrong labeling instances, we can train bag level relation predicting models with better performance. Meanwhile, the relation predicting model gives reward to the instance selection agent. Therefore, the agent and the relation classifier can be trained jointly.

However, REINFORCE is a Monte Carlo algorithm and need stochastic gradient method to optimize. It is unbiased and has good convergence properties but also may be of high variance and slow to train (Sutton and Barto, 1998).

Therefore, we train a REINFORCE based agent by integrating some other domain-specific rules to accelerate the training process and guide the agent to explore more effectively and learn a better policy. Here we use a rule pattern as the Fig.1 shows (**?**). The instances that return true (match

---

*Corresponding author

Proceedings of NAACL-HLT 2019, pages 2831–2835
Minneapolis, Minnesota, June 2 - June 7, 2019.

the pattern and label in any one of the rules) are denoted as x_{MI} and we adopt posterior regularization method (Ganchev, 2010) to regularize the posterior distribution of $\pi(a|s, \theta_\pi)$ on x_{MI}. In this way, we can construct a rule-based agent π_r. π_r tends to regard the instances in x_{MI} valuable and select them without wasting time in trial-and-error exploring. The number of such rules is 134 altogether and can match nearly four percents of instances in the training data.

Our contributions include:

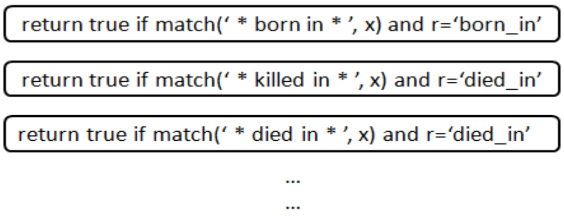
- We propose PR REINFORCE by integrating domain-specific rules to improve the performance of the original REINFORCE.
- We apply the PR REINFORCE to the instance selection task for DS dataset to alleviate the wrong label problem in DS.

2 Related Work

Among the previous studies in relation extraction, most of them are supervised methods that need a large amount of annotated data (Bach and Badaskar, 2007). Distant supervision is proposed to alleviate this problem by aligning plain text with Freebase. However, distant supervision inevitably suffers from the wrong label problem.

Some previous research has been done in handling noisy data in distant supervision. An expressed-at-least-once assumption is employed in (Mintz et al., 2009): if two entities participated in a relation, at least one instance in the bag might express that relation. Many follow-up studies adopt this assumption and choose a most credible instance to represent the bag. (Lin et al., 2016; Ji et al., 2017) employs the attention mechanism to put different attention weight on each sentence in one bag and assume each sentence is related to the relation but have a different correlation.

Another key issue for relation extraction is how to model the instance and extract features, (Zeng et al., 2014, 2015; Zhou et al., 2016) adopts deep neural network including CNN and RNN, these methods perform better than conventional feature-based methods.



```
return true if match(' * born in * ', x) and r='born_in'
return true if match(' * killed in * ', x) and r='died_in'
return true if match(' * died in * ', x) and r='died_in'
...
...
```

Figure 1: Rule Pattern Examples

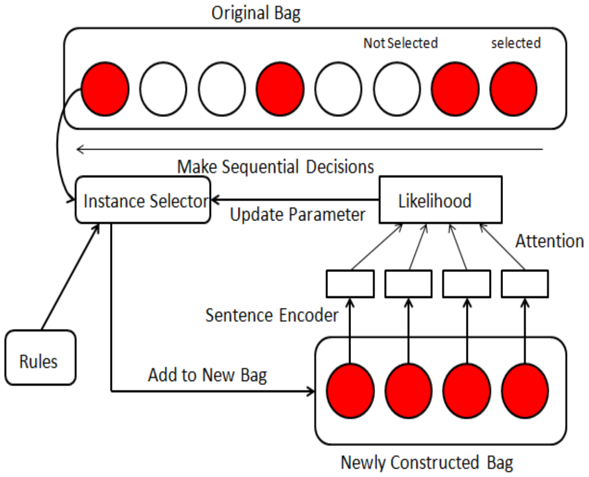


Figure 2: Overall Framework

Reinforcement learning has been widely used in data selection and natural language processing. (Feng et al., 2018) adopts REINFORCE in instance selection for distant supervision which is the basis of our work.

Posterior regularization (Ganchev, 2010) is a framework to handle the problem that a variety of tasks and domains require the creation of large problem-specific annotated data. This framework incorporates external problem-specific information and put a constraint on the posterior of the model. In this paper, we propose a rule-based REINFORCE based on this framework.

3 Methodology

In this section, we focus on the model details. Besides the interacting process of the relation classifier and the instance selector, we will introduce how to model the state, action, reward of the agent and how we add rules for the agent in training process.

3.1 Basic Relation Classifier

We need a pretrained basic relation classifier to define the reward and state. In this paper, we adopt the BGRU with attention bag level relation classifier f_b (Zhou et al., 2016). With $\mathbf{o}$ denoting the output of f_b corresponding to the scores associated to each relation, the conditional probability can be written as follows:

$$P_{f_b}(r|B^k, \theta_b) = \frac{exp(o_r)}{\sum_{k=1}^{n_r} exp(o_k)} \quad (1)$$

where r is relation type, n_r is the number of relation types, θ_b is the parameter vector of the basic relation classifier f_b and B^k denotes the input bag of the classifier.

In the basic classifier, the sentence representation is calculated by the sentence encoder network BGRU, the BGRU takes the instance x_i^k as input and output the sentence representation BGRU(x_i^k). And then the sentence level(ATT) attention will take $\{BGRU(x_1^k), BGRU(x_2^k), ..., BGRU(x_{|B^k|}^k)\}$ as input and output **o** which is the final output of f_b corresponding to the scores associated to each relation.

3.2 Original REINFORCE

Original REINFORCE agent training process is quite similar to (Feng et al., 2018). The instance selection process for one bag is completed in one trajectory. Agent $\pi(a|s, \theta_\pi)$ is trained as an instance selector.

The key of the model is how to represent the state in every step and the reward at the terminal of the trajectory. We use the pretrained f_b to address this key problem. The reward defined by the basic relation classifier is as follows:

$$R = \log P_{f_b}(r^k|\overline{B^k}, \theta_b) \quad (2)$$

In which r^k denotes the corresponding relation of B^k.

The state s mainly contained three parts: the representation of the candidate instance, the representation of the relation and the representation of the instances that have been selected.

The representation of the candidate instance are also defined by the basic relation classifier f_b. At time step t, we use BGRU(x_t^k) to represent the candidate instance x_t^k and the same for the selected instances. As for the embedding of relation, we use the entity embedding method introduced in TransE model (Bordes et al., 2013) which is trained on the Freebase triples that have been mentioned in the training and testing dataset, and the relation embedding re_k will be computed by the difference of the entity embedding element-wise.

The policy π with parameter $\theta_\pi = \{W, b\}$ is defined as follows:

$$P_\pi(A_t|S_t, \theta_\pi) = softmax(WS_t + b) \quad (3)$$

With the model above, the parameter vector can be updated according to REINFORCE algorithm (Sutton and Barto, 1998).

3.3 Posterior Regularized REINFORCE

REINFORCE uses the complete return, which includes all future rewards up until the end of the trajectory. In this sense, all updates are made after the trajectory is completed (Sutton and Barto, 1998). These stochastic properties could make the training slow. Fortunately, we have some domain-specific rules that could help to train the agent and adopt posterior regularization framework to integrate these rules. The goal of this framework is to restrict the posterior of π. It can guide the agent towards desired behavior instead of wasting too much time in meaninglessly exploring.

Since we assume that the domain-specific rules have high credibility, we designed a rule-based policy agent π_r to emphasize their influences on π. The posterior constrains for π is that the policy posterior for x_{MI} is expected to be 1 which indicates that agent should select the x_{MI}. This expectation can be written as follows:

$$E_{P_\pi}[\mathbf{l}(A_t = 1)] = 1 \quad (4)$$

where **l** here is the indicator function. In order to transfer the rules into a new policy π_r, the KL divergence between the posterior of π and π_r should be minimized, this can be formally defined as

$$minKL(P_\pi(A_t|S_t, \theta_\pi)||P_{\pi_r}(A_t|S_t, \theta_\pi)) \quad (5)$$

Optimizing the constrained convex problem defined by Eq.(4) and Eq.(5), we get a new policy π_r:

$$P_{\pi_r}(A_t|S_t, \theta_\pi) = \frac{P_\pi(A_t|S_t, \theta_\pi)exp(\mathbf{l}(A_t = 1) - 1)}{Z} \quad (6)$$

where Z is a normalization term.

$$Z = \sum_{A_t=0}^{1} P_{\pi_r}(A_t|X, \theta_\pi)exp(\mathbf{l}(A_t = 1) - 1)$$

Algorithm 1 formally define the overall framework of the rule-based data selection process.

4 Experiment

Our experiment is designed to demonstrate that our proposed methodologies can train an instance selector more efficiently.

```
Data: Original DS Dataset:
      $\mathbf{B} = \{B^1, B^2, ..., B^N\}$, Max
      Episode:M, Basic Relation
      Classifier:$f_b$, Step Size: $\alpha$
Result: An Instance Selector
initialization policy weight $\theta'_\pi = \theta_\pi$;
initialization classifier weight $\theta'_b = \theta_b$;
for episode m=1 to M do
    for $B^k$ in B do
        $B^k = \{x_1^k, x_2^k, ..., x_{|B^k|}^k\}, \overline{B^k} = \{\}$;
        for step i in $|B^k|$ do
            construct $s_i$ by $\overline{B^k}, x_i^k, re_k$;
            if $x_i^k \in x_{MI}$ then
                construct $\pi_r$;
                sample action $A_i$ follow
                  $\pi_r(a|s_i, \theta'_\pi)$;
            else
                sample action $A_i$ follow
                  $\pi(a|s_i, \theta'_\pi)$;
            end
            if $A_i$=1 then
                Add $x_i^k$ in $\overline{B^k}$;
            end
        end
        Get terminal reward:
          $R = \log P_{f_b}(r^k|\overline{B^k}, \theta'_b)$;
        Get step delayed reward: $R_i$=R;
        Update agent:
          $\theta_\pi \leftarrow \theta_\pi + \alpha \sum_{i=1}^{|B^k|} R_i \nabla_{\theta_\pi} \log \pi$
    end
    $\theta'_\pi = \tau\theta_\pi + (1-\tau)\theta'_\pi$;
    Update the classifier $f_b$;
end
```

Algorithm 1: PR REINFORCE

We tuned our model using three-fold cross validation on the training set. For the parameters of the instance selector, we set the dimension of entity embedding as 50, the learning rate as 0.01. The delay coefficient τ is 0.005. For the parameters of the relation classifier, we follow the settings that are described in (Zhou et al., 2016).

The comparison is done in rule-based reinforcement learning method, original reinforcement learning and method with no reinforcement learning which is the basic relation classifier trained on original DS dataset. We use the last as the baseline.

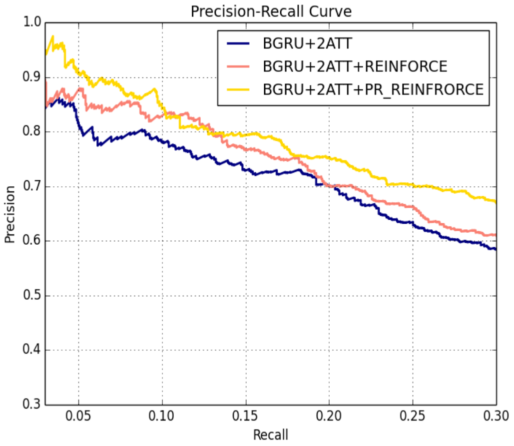


Figure 3: Precision/Recall Curves

4.1 Dataset

A widely used DS dataset, which is developed by (Riedel et al., 2010), is used as the original dataset to be selected. The dataset is generated by aligning Freebase with New York Times corpus.

4.2 Metric and Performance Comparison

We compare the data selection model performance by the final performance of the basic model trained on newly constructed dataset selected by different models. We use the precision/recall curves as the main metric. Fig.3 presents this comparison. PR REINFORCE constructs cleaned DS dataset with less noisy data compared with the original REINFORCE so that the BGRU+2ATT classifier can reach better performance.

5 Conclusions

In this paper, we develop a posterior regularized REINFORCE methodology to alleviate the wrong label problem in distant supervision. Our model makes full use of the hand-crafted domain-specific rules in the trial and error search during the training process of REINFORCE method for DS dataset selection. The experiment results show that PR REINFORCE outperforms the original REINFORCE. Moreover, PR REINFORCE greatly improves the efficiency of the REINFORCE training.

Acknowledgments

This work has been supported in part by NSFC (No.61751209, U1611461), 973 program (No. 2015CB352302), Hikvision-Zhejiang University

Joint Research Center, Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Engineering Research Center of Digital Library, Ministry of Education. Xiang Ren's research has been supported in part by National Science Foundation SMA 18-29268.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*, 2.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data.

Kuzman Ganchev. 2010. *Posterior regularization for learning with side information and weak supervision*. Ph.D. thesis, University of Pennsylvania.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Scalable Collapsed Inference for High-Dimensional Topic Models

Rashidul Islam and **James Foulds**
Department of Information Systems
University of Maryland, Baltimore County
{islam.rashidul,jfoulds}@umbc.edu

Abstract

The bigger the corpus, the more topics it can potentially support. To truly make full use of massive text corpora, a topic model inference algorithm must therefore scale efficiently in 1) documents and 2) topics, while 3) achieving accurate inference. Previous methods have achieved two out of three of these criteria simultaneously, but never all three at once. In this paper, we develop an online inference algorithm for topic models which leverages *stochasticity* to scale well in the number of documents, *sparsity* to scale well in the number of topics, and which operates in the *collapsed representation* of the topic model for improved accuracy and run-time performance. We use a Monte Carlo inner loop in the online setting to approximate the collapsed variational Bayes updates in a sparse and efficient way, which we accomplish via the Metropolis-Hastings Walker method. We showcase our algorithm on LDA and the recently proposed mixed membership skip-gram topic model. Our method requires only amortized $O(k_d)$ computation per word token instead of $O(K)$ operations, where the number of topics occurring for a particular document $k_d \ll$ the total number of topics in the corpus K, to converge to a high-quality solution.

1 Introduction

Topic models are powerful tools for analyzing today's massive, constantly expanding digital text information by representing high-dimensional data in a low-dimensional subspace. We can recover the main themes of a corpus by using topic models such as latent Dirichlet allocation (LDA) to organize, understand, search, and explore the documents (Blei et al., 2003).

Traditional LDA inference techniques such as variational Bayes and collapsed Gibbs sampling do not readily scale to corpora containing millions of documents. To scale up inference, the main approaches are distributed algorithms (Newman et al., 2008) and stochastic algorithms (Hoffman et al., 2010, 2013). Stochastic algorithms, such as stochastic variational inference (SVI), operate in an online fashion, and hence do not need to see all of the documents before updating the topics, so they can be applied to corpora of any size, without expensive distributed hardware (Hoffman et al., 2010). The "collapsed" representation of topic models is also frequently important, as it leads to faster convergence, efficient updates, and lower variance in estimation (Griffiths and Steyvers, 2004). The stochastic collapsed variational Bayesian inference (SCVB0) algorithm, proposed by (Foulds et al., 2013), combines the benefits of stochastic and collapsed inference.

Larger corpora typically support more topics, which brings the additional efficiency challenge of training a larger model (Mimno et al., 2012). This challenge has been addressed by exploiting sparsity to perform updates in time sublinear in the number of topics. A sparse variant of the SVI algorithm for LDA, SSVI, proposed by (Mimno et al., 2012), is scalable to large numbers of topics, but does not fully exploit the collapsed representation of LDA, which is important for faster convergence and improved inference accuracy, due to a better variational bound (Teh et al., 2007). The Metropolis Hastings Walker (MHW) method (Li et al., 2014) scales well in the number of topics, and uses a collapsed inference algorithm, but it operates in the batch setting, so it is not scalable to large corpora. LightLDA (Yuan et al., 2015)

Proceedings of NAACL-HLT 2019, pages 2836–2845
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

is a distributed approach to the MHW method which adopts a data-and-model-parallel strategy to maximize memory and CPU efficiency. However, it is not an online approach, and furthermore requires multiple expensive computer clusters to converge faster. Tensor methods are another approach to speeding up topic models (Anandkumar et al., 2014; Arora et al., 2012), which theoretically guarantee the recovery of the true parameters by overcoming the problem of local optima. These techniques use the method of moments instead of maximum likelihood estimation or Bayesian inference, which leads to lower data efficiency, and sometimes unreliable performance.

In this work, we propose a highly efficient and scalable inference algorithm for topic models. We develop an online algorithm which leverages *stochasticity* to scale well in the number of documents, *sparsity* to scale well in the number of topics, and which operates in the *collapsed* representation of topic models. We thereby combine the individual benefits of SVI, SSVI, SCVB0, and MHW into a single algorithm. Our approach is to develop a sparse version of SCVB0. Inspired by SSVI, we use a Monte Carlo inner loop to approximate the SCVB0 variational distribution updates in a sparse and efficient way, which we accomplish via MHW method. To show the generality of our algorithm, we explore the benefits of our inference method for LDA and another recently proposed topic model, MMSGTM, with experiments on both small and large-scale datasets.

2 Background

To build the foundation for our proposed method, in this section we provide the necessary background on LDA and MMSGTM topic models and their associated inference algorithms. This is followed by a description of the MHW sampler for reducing topic model sampling complexity.

2.1 Latent Dirichlet Allocation and SCVB0

Probabilistic topic models such as LDA (Blei et al., 2003) use latent variables to encode co-occurrence patterns between words in text corpora and other bag-of-words represented data. In LDA, we assume that the D documents in a corpus are each from mixture distributions of K individual topics ϕ_k, $k \in \{1, ..., K\}$, each of which are discrete distributions over words. For a document j of length N_j, the local (document-level) variables θ_j are a distribution over topics drawn from a Dirichlet prior with parameters α_k and for each token, global variables (corpus-level) ϕ_k are drawn from a Dirichlet prior with parameters β_w. Due to conjugacy, we can marginalize out topics $\boldsymbol{\Theta}$ and distributions over topics $\boldsymbol{\Phi}$, and perform inference only on the topic assignments $\boldsymbol{Z}$ in the collapsed representation of LDA (Griffiths and Steyvers, 2004). For scalable and accurate inference, Foulds et al. (2013) proposed a stochastic collapsed variational inference algorithm, SCVB0. The SCVB0 approach computes a variational discrete distribution γ_{ij} over the K topic assignment probabilities for each word i in each document j, but does not maintain the γ variables that increase the memory requirement of original batch CVB0 algorithm (Asuncion et al., 2009). SCVB0 iteratively updates each γ_{ij} using

$$\gamma_{ijk} :\propto \frac{N^{\Phi}_{w_{ij}k} + \beta_{w_{ij}}}{N^{Z}_{k} + \sum_w \beta_w}(N^{\Theta}_{jk} + \alpha_k) \qquad (1)$$

for each topic k, with j^{th} document's i^{th} word w_{ij}. The $\boldsymbol{N^Z}$, $\boldsymbol{N^\Theta}$, and $\boldsymbol{N^\Phi}$ are referred to as CVB0 statistics, where $\boldsymbol{N^Z}$ is the vector of expected number of words assigned to each topic, each entry j, k of matrix $\boldsymbol{N^\Theta}$, and each entry w, k of matrix $\boldsymbol{N^\Phi}$ are the expected number of times document j, and word w are assigned to topic k, respectively, across the corpus. To do stochastic updates of these variables, one sequence of step-sizes ρ^Φ for $\boldsymbol{N^\Phi}$ and $\boldsymbol{N^Z}$ and another sequence ρ^Θ for $\boldsymbol{N^\Theta}$ are maintained. The update of $\boldsymbol{N^\Theta_j}$ for every token i of document j with an online average of the current value and its expected value is

$$\boldsymbol{N^\Theta_j} := (1 - \rho^\Theta_t)\boldsymbol{N^\Theta_j} + \rho^\Theta_t C_j \gamma_{ij} \qquad (2)$$

where C_j is the document length. In practice, it is too expensive to update $\boldsymbol{N^\Phi}$ after every token. This leads to the use of minibatch updates with the average of the M per-token estimates of the form $\boldsymbol{Y_{ij}}$, which is a $W \times K$ matrix with the w_{ij}th row being γ_{ij} and with zeros in the other entries:

$$\boldsymbol{N^\Phi} := (1 - \rho^\Phi_t)\boldsymbol{N^\Phi} + \rho^\Phi_t \hat{\boldsymbol{N}}^{\boldsymbol{\Phi}} \qquad (3)$$

$$\boldsymbol{N^Z} := (1 - \rho^\Phi_t)\boldsymbol{N^Z} + \rho^\Phi_t \hat{\boldsymbol{N}}^{\boldsymbol{Z}}\ , \qquad (4)$$

where $\hat{\boldsymbol{N}}^{\boldsymbol{\Phi}} = \frac{C}{|M|}\sum_{ij\in M}\boldsymbol{Y_{ij}}$, $\hat{\boldsymbol{N}}^{\boldsymbol{Z}} = \frac{C}{|M|}\sum_{ij\in M}\gamma_{ij}$, and C is the number of words in the corpus. The SCVB0 algorithm outperforms stochastic VB (Hoffman et al., 2010) on large corpora by converging faster and often to a better solution (Foulds et al., 2013). However, the SCVB0 algorithm does not leverage sparsity, and hence requires $O(K)$ operations per word token.

2.2 MMSG Topic Model

To show the generality of our approach to topic models other than LDA, we will also apply our method to a recent model called the Mixed Membership Skip-gram Topic Model (MMSGTM) (Foulds, 2018), which combines ideas from topic models and word embeddings (cf. also (Das et al., 2015; Liu et al., 2015)). MMSGTM's generative model for words and their surrounding context is:

- For each word w_i in the corpus
  - Sample a topic $z_i \sim$ Discrete(θ_{w_i})
  - For each word $w_c \in context(i)$
    - Sample a context word $w_c \sim$ Discrete(ϕ_{z_i}) .

The inferred model can then be used to train embeddings for topics and words, although we do not consider this here. The MMSGTM admits a collapsed Gibbs sampler (CGS) which efficiently resolves the cluster assignments. With Dirichlet priors on the parameters, the CGS update is

$$p(z_i = k|.) :\propto (N^{(\Phi)-i}_{w_i k} + \alpha_k) \times \prod_{c=1}^{|context(i)|} \frac{N^{(\Phi)-i}_{w_c k} + \beta_{w_c} + N^{(i,c)}_{w_c}}{N^{(Z)-i}_{k} + \sum_w \beta_w + c - 1}\,, \quad (5)$$

where α and β are parameter vectors for Dirichlet priors over the topic and word distributions, $\boldsymbol{N^{\Phi}_{w_i}}$ and $\boldsymbol{N^{\Phi}_{w_c}}$ are input and output word-topic counts (excluding the current word), N^Z is the total topic counts in output word-topic counts, and $N^{(i,c)}_{w_c}$ is the number of occurrences of word w_c before the c^{th} word in the i^{th} context. MMSGTM exploits the MHW algorithm, which scales sub-linearly in K, but not in the number of training documents.

2.3 Metropolis-Hastings-Walker Sampler

The MHW method (Li et al., 2014), which is a key component of our approach, uses a data structure called an alias table which allows sampling from a discrete distribution in amortized $O(1)$ time. Assuming initial probabilities $p_0, p_1, ..., p_{l-1}$ of a distribution over l outcomes and average of probabilities $a = \frac{1}{l}$, the alias table A can be formed as follows (Marsaglia et al., 2004):

- Initialize: for i from 0 to $l - 1$
  - $A_{alias}[i] = i$ and $A_{prob} = (i + 1)a$
- Do the following steps $n - 1$ times
  - Find smallest p_i and largest p_j
  - Set $A_{alias}[i] = j$ and $A_{prob} = i \times a + p_i$
  - $p_j := p_j - (a - p_i)$ and $p_i := a$.

Then, to sample from p using the alias table:

- Roll l-sided fair die to choose element i of A
- If $Rand(1) < A_{prob}[i]$ return i, else return $A_{alias}[i]$.

Li et al. (2014) cache alias table samples, avoiding the need to store the table. Once the supply of samples is exhausted they compute a new alias table. They draw samples from the Gibbs sampling update, analogous to γ_{ij} in Equation 1, in amortized $O(k_d)$ time by decomposing the update into

$$p(z_{ij} = k|.) \propto N^{\Theta}_{jk}\frac{N^{\Phi}_{w_{ij}k} + \beta_{w_{ij}}}{N^{Z}_{k} + \sum_w \beta_w} + \alpha_k \frac{N^{\Phi}_{w_{ij}k} + \beta_{w_{ij}}}{N^{Z}_{k} + \sum_w \beta_w} \quad (6)$$

where the first term, sparse in k_d, admits sampling in $O(k_d)$ time, and the second term is dense but slow changing. A Metropolis-Hastings (M-H) update is used to correct for approximating the CGS update with a proposal distribution $q(k)$ based on the stale alias samples. Foulds et al. (2018) propose to apply simulated annealing to optimize instead of sample, and which improves mixing for the MMSGTM. This is achieved by raising the model part of the M-H acceptance ratio for a new sample $z_i^{(new)} \sim q(k)$ to the power of $\frac{1}{T_j}$ at iteration j:

$$p(accept\ z_i^{(new)}|.) = \quad (7)$$
$$min(1, (\frac{p(z_i^{(new)})}{p(z_i^{(old)})})^{\frac{1}{T_j}} \frac{q(z_i^{(old)})}{q(z_i^{(new)})})\,.$$

3 Sparse Stochastic CVB0

In this section, we introduce our approach, a sparse version of SCVB0, which combines the individual benefits of the SVI, SSVI, SCVB0 and MHW algorithms, to scale well not only in the

SparseSCVB0			Original SCVB0			SVI		
units	neurons	support	data	matrix	eeg	word	cells	circuit
unit	model	kernel	clustering	pca	time	words	cell	current
net	synaptic	margin	cluster	linear	data	character	cortex	circuits
hidden	input	function	clusters	principal	brain	recognition	response	figure
output	neuron	vector	algorithm	eigenvectors	activity	characters	firing	input
neural	response	svm	model	eigenvalues	signal	trained	cortical	analog
networks	cell	machines	problem	eigenvalue	analysis	input	inhibitory	filter

Table 1: Randomly selected example topics, while models trained on the NIPS corpus for $K = 500$.

Algorithm 1 SparseSCVB0 for TM Inference

```
Randomly initialize $\boldsymbol{N^G}, \boldsymbol{N^L}; \boldsymbol{N^Z} := \sum_w \boldsymbol{N^G_w}$;
$doSparse = true$ or $false$
for each minibatch $M$ do
    $\hat{\boldsymbol{N}}^{\boldsymbol{G}} := 0; \hat{\boldsymbol{N}}^{\boldsymbol{Z}} := 0$
    for each document $j$ in $M$ do
        for each token $i$ in $j$ do
            $\gamma_{ij}^{(pseudo)} := 0$
            for each sample $s$ in $S$ do
                draw $z_i^{(new)} \sim q(k)$
                    //via efficient sampling or
                    cached alias samples
                accept or reject $z_i^{(new)}$ via Eq. 7
                if (accept), $z_i := z_i^{(new)}$
                $\gamma_{ij}^{(pseudo)}[z_i] := \gamma_{ij}^{(pseudo)}[z_i] + \frac{1}{S}$
            end for
            $\boldsymbol{N_j^L} := (1 - \rho_t^L)\boldsymbol{N_j^L} + \rho_t^L C_j \gamma_{ij}^{(pseudo)}$
            if (not burn-in pass),
                    //update estimates for i or for
                    each context word of i:
                $\hat{\boldsymbol{N}}^{\boldsymbol{G}}_{\boldsymbol{w_{ij}}} := \hat{\boldsymbol{N}}^{\boldsymbol{G}}_{\boldsymbol{w_{ij}}} + \frac{C}{|M|}\gamma_{ij}^{(pseudo)}$
                $\hat{\boldsymbol{N}}^{\boldsymbol{Z}} := \hat{\boldsymbol{N}}^{\boldsymbol{Z}} + \frac{C}{|M|}\gamma_{ij}^{(pseudo)}$
            elseif($doSparse$),$\boldsymbol{N_j^L}[k] < \tau\rho_{t_{C_j}}^L C_j := 0$
        end for
    end for
    update $\boldsymbol{N^G} := (1 - \rho_t^G)\boldsymbol{N^G} + \rho_t^G \hat{\boldsymbol{N}}^{\boldsymbol{G}}$
    update $\boldsymbol{N^Z} := (1 - \rho_t^G)\boldsymbol{N^Z} + \rho_t^G \hat{\boldsymbol{N}}^{\boldsymbol{Z}}$
end for
```

number of documents but also in the number of topics, while gaining the benefits of collapsed inference. We refer to our method as the sparse stochastic collapsed variational Bayesian inference (SparseSCVB0) algorithm.

In SparseSCVB0, we obtain sparsity by substituting sparse Monte Carlo approximations $\gamma_{ij}^{(pseudo)}$ for the original SCVB0 variational distributions γ_{ij}. The justification for this procedure, also used by (Mimno et al., 2012), is that the expected value of an average over one-hot samples from a distribution is equal to that distribution:

$$E_{s_i \sim p(\mathbf{s})}\Big[\frac{\sum_{i=1}^S \delta_k(s_i)}{S}\Big] = \frac{1}{S}\sum_{i=1}^S E_{s_i \sim p(\mathbf{s})}\Big[\delta_k(s_i)\Big]$$

$$= \frac{1}{S}\sum_{i=1}^S \sum_{k'} \delta_k(s_i = k')p(s_i = k') = p(s = k)\,.$$

Thus, the overall procedure is still a valid stochastic optimization algorithm. We approximate the inner loop sampler in time sublinear in K, constructing $\gamma^{(pseudo)}$ by generating S samples from γ_{ij} using the MHW method. To describe the general form of our algorithm, we introduce SparseSCVB0 statistics: local (e.g. document-level) expected counts $\boldsymbol{N^L}$, global (corpus-level) expected counts $\boldsymbol{N^G}$, and total expected topic counts $\boldsymbol{N^Z}$. We approximate local sufficient statistics $\boldsymbol{N^L}$ for each token i in document j via:

$$N_{jk}^L \approx C_j E_{\gamma_{ij}^{(pseudo)}}\Big[\frac{\sum_{s \in S}\delta_{z_{ij}^s = k}}{S}\Big]\,.$$

Since $\gamma^{(pseudo)}$ is sparse, we can efficiently update these statistics using only its non-zero entries. This approach allows us to learn high-dimensional topic models efficiently on very large corpora. Before updating global parameters in a similar fashion, it may also be beneficial to perform a small number of burn-in passes to learn the local parameters $\boldsymbol{N^L}$ (Foulds et al., 2013). For large-scale datasets (e.g. Wikipedia), SparseSCVB0 operates in a "mini-epoch" approach where we process a large subset of the corpus (e.g. $5,000$ documents) several (e.g. 3) times, before discarding and processing the next subset, and so on. This allows a "warm start" of $\boldsymbol{N^L}$ in repeating iterations, with a small memory overhead. Pseudo-code of SparseSCVB0 for a mini-epoch is provided in Algorithm 1, which we discuss more in the next two sections, including model-specific aspects.

SparseSCVB0			Original SCVB0			SVI		
infection	human	liver	decreased	dna	medication	mutations	hormone	intensive
infected	humans	hcv	decrease	replication	patients	mutation	invasive	icu
infections	chimpanzee	hbv	increased	chromatin	medications	mutated	androgen	delirium
host	macaque	hepatic	increase	origins	prescribed	crosses	hormones	occurrences
infectious	monkey	hepatitis	decreases	ssdna	patient	missense	testosterone	sedation
antiviral	primates	hepatocytes	decreasing	primase	prescription	mut	chr	haloperidol
inoculation	chimpanzees	cirrhosis	dramatically	xenopus	pharmacy	hpf	hormonal	psychotropic

Table 2: Randomly selected example topics, while models trained on the PubMed corpus with $K = 1,000$.

SparseSCVB0			Original SCVB0			SVI		
band	town	blood	army	data	gold	president	japanese	engine
music	city	treatment	war	system	silver	political	china	test
released	road	disease	division	software	diamond	court	chinese	center
show	south	patients	battle	computer	golden	senate	japan	small
song	river	medical	forces	systems	bronze	constitution	people	gas
live	village	health	corps	version	diamonds	politics	countries	engines
records	school	effects	infantry	bit	pit	elected	asia	base

Table 3: Randomly selected example topics, while models trained on the Wikipedia corpus for $K = 1,000$.

4 SparseSCVB0 for LDA

To deploy SparseSCVB0 for LDA, we use the M-H proposal distribution from (Li et al., 2014) which involves drawing samples exactly from document-specific sparse terms or approximately using cached samples from the alias table:

$$q(k) := \frac{P_L}{P_L + Q_G} p_L(k) + \frac{Q_G}{P_L + Q_G} q_G(k) \quad (8)$$

where $p_L(k)$ and $q_G(k)$ represent the sparse and dense part, respectively from Equation 6 and $P_L = \sum_k p_L(k)$, $Q_G = \sum_k q_G(k)$. When $\frac{P_L}{P_L+Q_G} > RandUnif(1)$, we sample from the sparse part in $O(k_d)$ time depending on only the non-zero entries of $\boldsymbol{N_j^L}$ (analogous to $\boldsymbol{N_j^\Theta}$), as $\boldsymbol{N_j^L}$ is sparse in the LDA setting. Otherwise, we sample from the dense part in amortized $O(1)$ time using the alias method. Unfortunately, due to the stochastic update, any entry of $\boldsymbol{N_j^L}$ never becomes exactly zero, however it may maintain a very small value. To address this, we apply a sparsification heuristic, where we threshold $\boldsymbol{N_j^L}$ after burn-in passes for each document iteration. We parameterize the threshold as $\tau \rho^L_{t_{C_j}} C_j$; where C_j is the length of the j document, $\rho^L_{t_{C_j}}$ is the step size for the last token of this document, and constant $0 < \tau \leq 1$ controls the sparsity. In our preliminary experiments, we found that, somewhat counter-intuitively, the Monte Carlo and sparsification approximations actually improve convergence in early iterations. We believe that this is because they help SCVB0 escape the initial high-entropy regime, during which convergence of variational algorithms is poor (Salakhutdinov et al., 2003). This property makes the benefit of annealing insignificant, so we do not use simulated annealing for LDA inference, fixing $T_j = 1$.

An additional optimization of SparseSCVB0 for LDA inference can be performed by "clumping" (Teh et al., 2007; Foulds et al., 2013), where one update of the local parameters is performed for each distinct word type in each document. This is performed by fixing the variational distribution, and scaling the update by number of copies of the distinct word type in the document. If we observe the distinct word type w_{aj}, which occurs m_{aj} times in document j, the update is

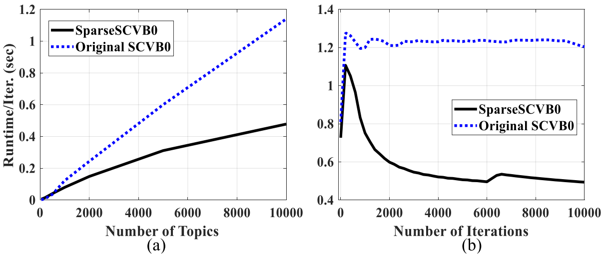
$$\begin{aligned}\boldsymbol{N_j^L} &:= (1 - \rho_t^L)^{m_{aj}} \boldsymbol{N_j^L} \\ &\quad + (1 - (1 - \rho_t^L)^{m_{aj}}) C_j \gamma_{aj}^{(pseudo)} \,.\end{aligned} \quad (9)$$

5 SparseSCVB0 for MMSGTM

The main contribution of our approach for the MMSGTM algorithm is to scale this algorithm in number of documents with online inference, as MMSGTM already scales sublinearly in K using MHW. Foulds et al. (2018) use an MHW proposal which approximates the CGS update, interpreted as a product of experts (Hinton, 2002) in which each word in the context is an "expert" which weighs in multiplicatively on the update, with a mixture of experts. In the proposal, they draw a

		Input word = learning Top words in top 2 topics for the input word
NIPS	SparseSCVB0	algorithms algorithm reinforcement problem problems | learning gradient descent rate weight machine
	Original SCVB0	learning networks network algorithm neural | propagation function reinforcement gradient algorithms
Wikipedia	SparseSCVB0	university school college education students research | public center science schools article information
	Original SCVB0	students school education university college schools | center year program degree systems information

Table 4: Top words in the top 2 topics for an input word using original SCVB0 and SparseSCVB0 for MMSGTM.



(a) (b)

Figure 1: Comparison of runtime per iteration for LDA in terms of: (a) number of topics K and (b) number of iterations when $K = 10,000$. Original SCVB0 is linear in K, while SparseSCVB0 is sublinear in K.

context word w_c uniformly from the context of the current word, $w_c \sim Uniform(|context(w_i)|)$, and then sample a word based on the chosen context word's contribution to the update:

$$q(k) :\propto \frac{N^G_{w_c k} + \beta_{w_c}}{N^Z_k + \sum_w \beta_w} \qquad (10)$$

where $\boldsymbol{N^G_{w_c}}$ is analogous to the output context word-topic counts $\boldsymbol{N^\Phi_{w_c}}$ of original MMSGTM model. The proposal samples via the alias method in amortized $O(1)$ time, instead of $O(k_d)$ time, since it does not involve the sparse term. We use this proposal to approximate the CVB0 update for the model, which is a deterministic version of Equation 5, neglecting to exclude the current assignment of z_i. We update $\boldsymbol{N^G_{w_i}}$ (analogous to $\boldsymbol{N^\Phi_{w_c}}$) for each current word w_i locally, but update $\boldsymbol{N^G_{w_c}}$ and $\boldsymbol{N^z}$ via minibatch counts $\boldsymbol{\hat{N}^G_{w_c}}$ and $\boldsymbol{\hat{N}^Z}$, respectively, for each context word w_c of current word w_i. Unlike for LDA, C_j in the local updates represents the total number of input word j in the corpus. As we draw multiple output words from each topic assignment, the effective temperature of the MMSGTM model is much lower than for standard LDA which may cause problems with mixing and leads it to get stuck in the initial regime. Following Foulds et al. (2018), we perform simulated annealing which varies the M-H acceptance ratio to improve mixing. We parameterize the temperature schedule as $T_j = T_0 + \lambda\kappa^{\frac{\mu j}{D}}$, where T_0 is the target final temperature, $\kappa \leq 1$, constant μ controls the amount of temperature reduction after each document iteration j and λ controls the initial temperature.

6 Experiments

In this section we study the performance of our SparseSCVB0 [1] algorithm, on small as well as large corpora to validate the proposed method for topic models such as LDA and MMSGTM, and to compare with other state-of-the-art algorithms.

6.1 Experimental Environment and Datasets

We compared SparseSCVB0 to SCVB0 and SVI. For a fair comparison, we implemented all of them in the fast high-level language Julia V0.6.2 (Bezanson et al., 2017). We conducted all experiments on a computer with 64GB memory and an Intel Xeon E5-2623 V4 processor with 2.60 GHz clock rate, 8×256KB L2 Cache and 10MB L3 Cache. As we only use one single thread for sampling across all experiments, only one CPU core is active throughout the experiment with only 256KB available L2 Cache.[2]

We used *NIPS*, *Reuters-150*, *PubMed Central*, and *Wikipedia* as representative very small, small, medium, and large-scale datasets, respectively. The *NIPS* corpus has 1740 scientific articles from years 1987-1999 with 2.3M tokens, due to Sam Roweis. The newswire corpus *Reuters-150* contains $15,500$ articles with dictionary size of $8,350$ words. *PubMed Central* has $320M$ tokens across $165,000$ scientific articles and a vocabulary size of around $38,500$ words. The *Wikipedia* corpus contained 4.6 million articles from the online

---

[1] Code implementing SparseSCVB0 can be found at `https://github.com/dr97531/SparseSCVB0`.

[2] Since SSVI relies on multiple complex implementation details, we were unable to develop a fair implementation, nor were we able to obtain soure code for a previous implementation. We expect that its accuracy would be similar to SVI, with a speed-up at or below that bestowed by a MHW-based inner loop. (Mimno et al., 2012) apply it with only 200 topics for most of their experiments, and at most 1000 topics.

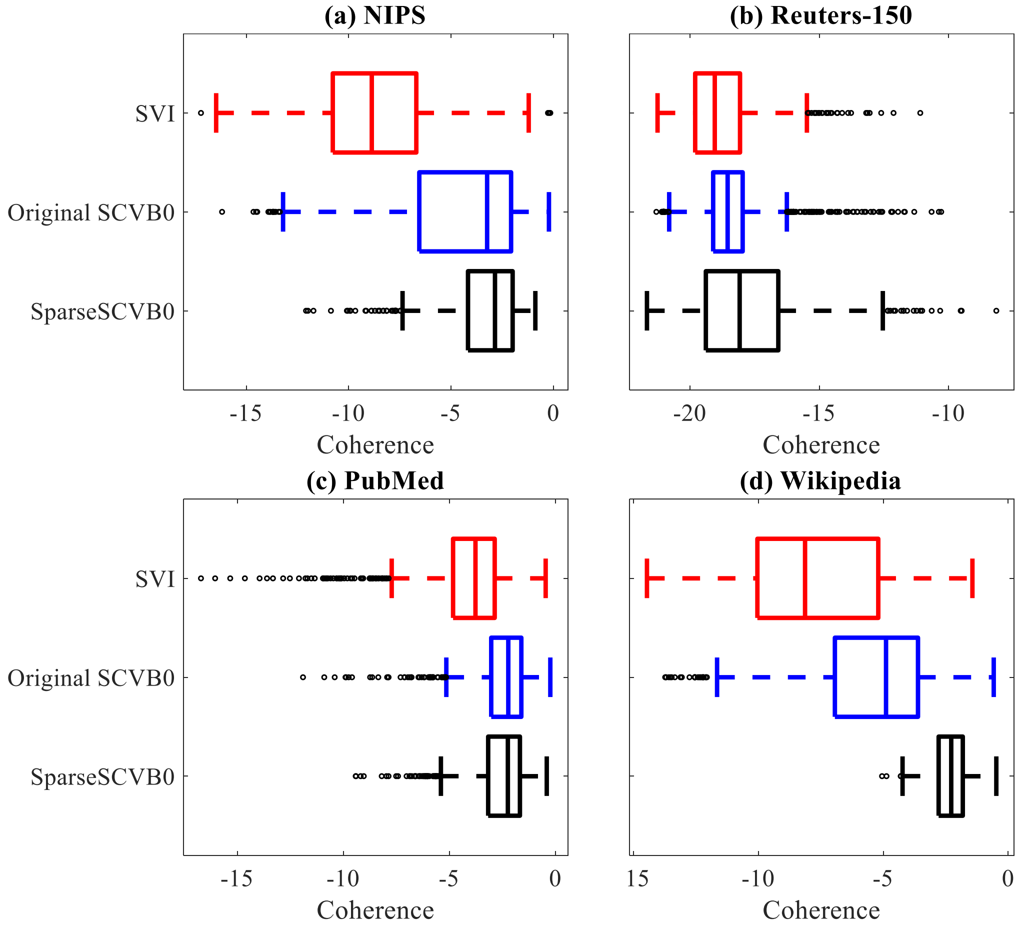


Figure 2: Per-topic coherence for LDA when $K = 1,000$ on (a) NIPS, (b) PubMed, and (c) Wikipedia. SparseSCVB0 completely outperforms other models for large-scale corpus.

network data communication information communicate connection
prison prisoners prisoner imprisoned jail escaped detained guards
dog dogs shepherd hounds bred coat scent instinct eating companion
song sung sing singing sings sang songs recorded melody tune
votes vote cast elections voted candidate parties majority election
wind winds blowing speed blows direction high low blown chill
hour hours noon time daylight minutes midnight morning seconds

Table 5: Randomly selected topics from a $10,000$-topic model trained using SparseSCVB0 on Wikipedia

encyclopedia. We used the dictionary of $7,700$ words which was extracted by Hoffman et al. (Hoffman et al., 2013). There were 811M tokens in the corpus.

6.2 Performance for LDA

We implement SparseSCVB0, original SCVB0 and SVI algorithms using the clumping optimization (Teh et al., 2007) technique. In all LDA experiments, each algorithm was trained using mini-batches of size 20 for the NIPS corpus and 100 for other corpora. For PubMed and Wikipedia, we chose mini-epoch subsets of size $5,000$ documents and processed for 5 passes. We used a step-size schedule of $\frac{scale}{(\eta+t)^{\kappa}}$ as in original SCVB0 for global parameters, where t is the document iteration with $scale = 100.0$, $\eta = 1000.0$ and $\kappa = 0.9$. For document-level parameters, we used the $scale = 1.0$, $\eta = 10.0$ and $\kappa = 0.9$, with t referring here to the word iteration of the current document. In case of PubMed corpus, we found out that original SCVB0 and SVI tend to stuck in the initial regime for document-level step-size parameter $\eta = 1.0$ which we later fixed by setting $\eta = 10.0$, while SparseSCVB0 didn't suffer from this problem due to the extra randomness from the sparse sampled updates. Finally, we choose hyper-parameters $\alpha = 0.1$ and $\beta = 0.01$ and burn-in pass of 5 for each document in all LDA experiments. For SparseSCVB0, we used sample size $S = 5$ to approximate $\gamma^{(pseudo)}$ and $\tau = 1/K$ for the sparsification heuristic on local parameters.

To study the acceleration benefits of our approach, we evaluated the runtime performance per

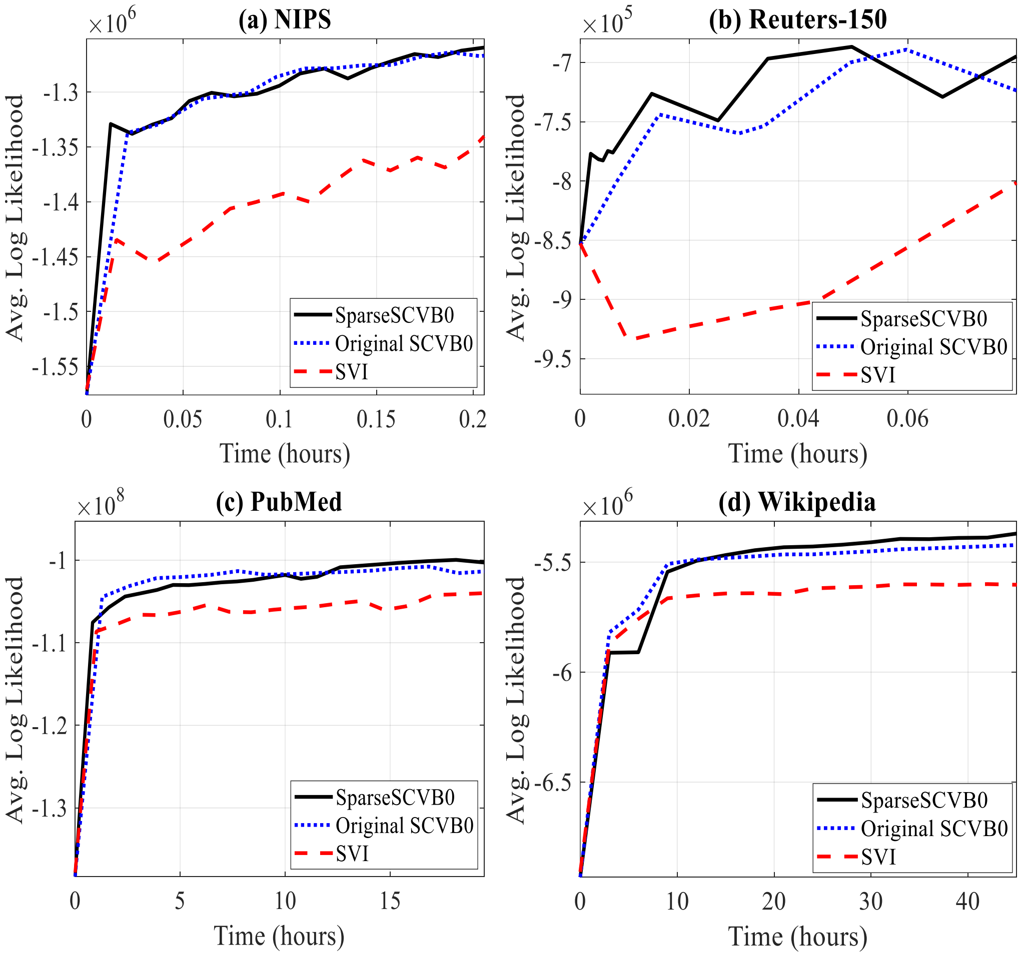


Figure 3: Comparison of average log-likelihood *vs*. Time for LDA on (a) NIPS, (b) PubMed, and (c) Wikipedia.

iteration on the number topics and the number of iterations. In Figure 1(a), SparseSCVB0 is compared to original SCVB0 in terms of the average runtime per document iteration as a function of the number topics. We see that original SCVB0 requires average linear runtime due to $O(K)$ operations to compute collapsed variational distribution, while the average runtime for SparseSCVB0 grows sublinearly in K, due to $O(k_d)$ operations instead of $O(K)$ operations. SparseSCVB0 starts with approximately $O(K)$ operations in its initial stage of iterations, but it starts getting a benefit from sparsification heuristic after burning in, as shown in Figure 1(b) for $K = 10,000$.

To evaluate the performance in terms of learned topic quality, we start by comparing all of the algorithms in qualitative experiments (see Table 1, Table 2, and Table 3) where we show randomly selected example topics, while all the models were trained on the NIPS, PubMed, and Wikipedia corpus for $K = 500$, $K = 1,000$, and $1,000$, respectively. To get a quantitative insight we evaluated the topics using the per topic coherence metric, which measures the semantic quality of a topic based on the W most probable words for the topics (Mimno et al., 2011), thereby approximating the user viewing experience. In Figure 2, we see that SparseSCVB0 generates better quality topics with higher coherence scores than the other two models for $K = 1000$ with $W = 10$ after running all the models on each corpus for the same amount of time. The coherence performance of SparseSCVB0 increases substantially in the case of the large-scale corpus (Figure 2(c)), since it gets the opportunity to use its runtime advantage and process more documents than the other algorithms.

To investigate model convergence, we measured the held-out log-probability versus wall-clock time for all the algorithms. For each experiment we held-out a set of documents (150 documents for NIPS, 3500 documents for Reuters, and 1000 documents for all other corpora) as test data

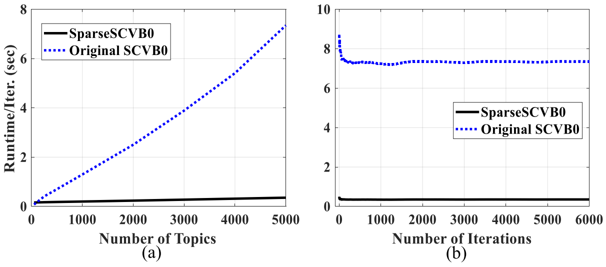


Figure 4: Comparison of runtime per iteration for MMSGTM in terms of: (a) number of topics K and (b) number of iterations when $K = 5,000$. SparseSCVB0 runs in amortized $O(1)$ time, while original SCVB0 is linear in K.

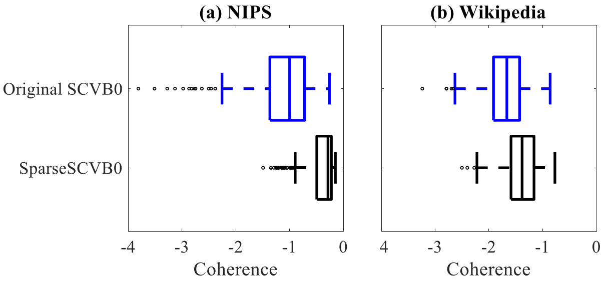


Figure 5: Per-topic coherence for MMSGTM, $K = 500$, on (a) NIPS and (b) Wikipedia. SparseSCVB0 has higher coherence scores for both the small and large corpora.

and trained the model on the rest of the corpus. Then, we split each test document in half, estimated local parameters on first half and finally computed the log-likelihood of the remaining half of the document. Figure 3 shows the comparison of average log-likelihood versus wall-clock time for all four corpora. In terms of log-likelihood, SparseSCVB0 provides an approximately similar result to original SCVB0 for the small corpus, but it converged to a better solution than others in the case of large corpora like Wikipedia (see Figure 3(c)), likely due to its processing a larger number of documents.

SparseSCVB0 enables the large-scale computation needed to learn high-dimensional topic models that could not feasibly be trained using previous methods due to their runtime complexity in the number of documents and/or topics. We show randomly selected topics from the LDA model with $K = 10,000$ in Table 5. This big topic model was trained for 36 hours using SparseSCVB0 on Wikipedia. We performed a dense initialization, running original SCVB0 for the first 5 hours, which was found to help avoid local optima.

6.3 Performance for MMSGTM

We also conducted experiments to evaluate the performance of SparseSCVB0 for MMSGTM and compare with original SCVB0. In all MMSGTM experiments, we kept the same step size schedule for global parameters as $scale = 1.0$, $\eta = 5.0$ and $\kappa = 0.9$, but for local parameter updates we maintain a separate step-size schedule of $\frac{scale}{(\eta+t)^\kappa}$ for each input word, with t referring to the number of times we processed this input word, while η and κ values remained the same. For simulated annealing of SparseSCVB0, we used $T_0 = 0.00001$, $\kappa = 0.9$, $\mu = 5$ and $\lambda = |context|$ with a $context$ size of 5. We kept the same number of document burn-in passes as we did for the LDA experiments.

In Figure 4, we show the runtime improvement of SparseSCVB0 over original SCVB0 for MMSGTM in a similar experiment to the one for LDA. For MMSGTM, SparseSCVB0 substantially outperforms original SCVB0 by processing each document in amortized $O(1)$ time. We provide qualitative results in the case of MMSGTM model by showing several top words in the top 2 topics for an input word using original SCVB0 and SparseSCVB0 in Table 4 for $K = 500$. As for LDA, SparseSCVB0 allows us to generate topics with higher coherence scores compared to the original SCVB0 after running for the same amount of time (Figure 5) on both small and large corpora.

7 Conclusions

This paper introduced SparseSCVB0, a sparse version of the SCVB0 inference algorithm which performs fast, scalable high-dimensional topic model inference. SparseSCVB0 leverages stochasticity to scale well in both the corpus size and in the number of topics. It operates in the collapsed representation of topic models which leads to fast convergence while providing an improved variational bound. We show that SparseSCVB0 reduces the operational complexity for the variational Bayes update of online topic models from $O(K)$ to $O(k_d)$ time for LDA and amortized $O(1)$ time for MMSGTM. We evaluated and compared the performance of our approach with state-of-the-art models such as original SCVB0 and SVI to demonstrate that SparseSCVB0 converges much more efficiently, while maintaining high quality topics with a better per-topic coherence score.

References

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. 2014. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832.

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning topic models–going beyond SVD. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 795–804.

J. R. Foulds. 2018. Mixed membership word embeddings for computational social science. *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS).*

James Foulds, Levi Boyles, Christopher DuBois, Padhraic Smyth, and Max Welling. 2013. Stochastic collapsed variational Bayesian inference for latent Dirichlet allocation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 446–454. ACM.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 856–864.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. 2014. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 891–900. ACM.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

George Marsaglia, Wai Wan Tsang, Jingbo Wang, et al. 2004. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11.

David Mimno, Matt Hoffman, and David Blei. 2012. Sparse stochastic inference for latent Dirichlet allocation. *Proceedings of the 29th International Conference on Machine Learning.*

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

David Newman, Padhraic Smyth, Max Welling, and Arthur U Asuncion. 2008. Distributed inference for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1081–1088.

Ruslan Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. 2003. Optimization with EM and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 672–679.

Yee W Teh, David Newman, and Max Welling. 2007. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1353–1360.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. LightLDA: Big topic models on modest computer clusters. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1351–1361. International World Wide Web Conferences Steering Committee.

An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction

Wang Chen[1], **Hou Pong Chan**[1], **Piji Li**[2], **Lidong Bing**[3], **Irwin King**[1]
[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]Tencent AI Lab
[3]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[1]{wchen, hpchan, king}@cse.cuhk.edu.hk
[2]pijili@tencent.com, [3]l.bing@alibaba-inc.com

Abstract

In this paper, we present a novel integrated approach for keyphrase generation (KG). Unlike previous works which are purely extractive or generative, we first propose a new multi-task learning framework that jointly learns an extractive model and a generative model. Besides extracting keyphrases, the output of the extractive model is also employed to rectify the copy probability distribution of the generative model, such that the generative model can better identify important contents from the given document. Moreover, we retrieve similar documents with the given document from training data and use their associated keyphrases as external knowledge for the generative model to produce more accurate keyphrases. For further exploiting the power of extraction and retrieval, we propose a neural-based merging module to combine and re-rank the predicted keyphrases from the enhanced generative model, the extractive model, and the retrieved keyphrases. Experiments on the five KG benchmarks demonstrate that our integrated approach outperforms the state-of-the-art methods.

1 Introduction

Keyphrases are short text pieces that can quickly express the key ideas of a given document. The keyphrase generation task aims at automatically generating a set of keyphrases given a document. As shown in the upper part of Figure 1, the input is a document and the output is a set of keyphrases. Due to the concise and precise expression, keyphrases are beneficial to extensive downstream applications such as text summarization (Zhang et al., 2004; Wang and Cardie, 2013), sentiment analysis (Wilson et al., 2005; Berend, 2011), and document clustering (Hulth and Megyesi, 2006; Hammouda et al., 2005).

Document:
Futility-Based Offspring Sizing. **Parameter control** in **evolutionary algorithms** (EAs) has been shown to be beneficial; however, the control of offspring size has so far received very little attention. This paper introduces Futility-Based **Offspring Sizing** (FuBOS), a method for controlling offspring size on a per generation basis without even requiring the user to set an initial offspring size value. . .
Keyphrases:
{**evolutionary algorithm**; parameterless evolutionary algorithm; **parameter control**; **offspring sizing**; optimization}

Retrieved Document:
An Exploration into Dynamic **Population Sizing**. Traditional **evolutionary algorithms** are powerful problem solvers that have several fixed parameters which require prior specification. . . While many methods of **parameter control** have been published that focus on removing the population size parameter, μ, all hampered by a variety of problems. This paper investigates the benefits of making μ a dynamic parameter and introduces two novel methods for population control. . .
Retrieved Keyphrases:
{**evolutionary algorithm**; parameterless evolutionary algorithm; **parameter control**; **population sizing**; optimization}

Figure 1: An example of keyphrase generation and retrieval. The present keyphrases are bold.

Existing methods on keyphrase generation can be divided into two categories: *extractive* and *generative*. Extractive methods (Medelyan et al., 2009; Mihalcea and Tarau, 2004; Zhang et al., 2016; Luan et al., 2017) identify present keyphrases that appear in the source text like "parameter control" in Figure 1. Although extractive methods are simple to implement, they cannot predict absent keyphrases which are not in the document like "optimization" in Figure 1. Generative methods (Meng et al., 2017; Chen et al., 2018a; Ye and Wang, 2018; Yuan et al., 2018) adopt the well-known encoder-decoder generative model (Luong et al., 2015; Bahdanau et al., 2014) with copy mechanism (Gu et al., 2016; See et al., 2017) to produce keyphrases. In a generative model, the decoder generates keyphrases word by word through either selecting from a predefined vocabulary according to a language model or copying from the source text according to the copy probability distribution computed by a copy mechanism. Thus, these generative methods are capable of generating both present and absent keyphrases.

From a high-level perspective, extractive methods directly locate essential phrases in the docu-

Proceedings of NAACL-HLT 2019, pages 2846–2856
Minneapolis, Minnesota, June 2 - June 7, 2019.

ment while generative models try to understand the document first and then produce keyphrases. To the best of our knowledge, these two kinds of methods have been developing independently without any combinations among them.

However, when human annotators are asked to assign keyphrases to a document, they usually first obtain a global sense about which parts of the document are important and then write down the keyphrases word by word based on a more detailed understanding. To achieve such a goal, we propose a multi-task learning framework to take advantage of both extractive and generative models. For keyphrase extraction, we adopt a neural sequence labeling model to output the likelihood of each word in the source text to be a keyphrase word (or the importance score of each word). These importance scores are then employed to rectify the copy probability distribution of the generative model. Since the extractive model is explicitly trained to identify keyphrases from the source text, its importance scores can help the copy mechanism to identify important source text words more accurately. Different from the copy probability distribution which is dynamic at each generation step, these importance scores are static. Therefore, they can provide a global sense about which parts of the document are important. In addition, these scores are also utilized to extract present keyphrases which will be exploited by the merging module.

Moreover, human annotators can also incorporate relevant external knowledge like the keyphrases of similar documents that they read before to assign more appropriate keyphrases. Correspondingly, to incorporate external knowledge, we propose a *retriever* to retrieve similar documents of the given document from training data. For instance, as shown in Figure 1, we retrieve a document from the **KP20k** training dataset that has the highest similarity with the upper document. The retrieved document is assigned with almost the same keyphrases as the upper document. Therefore, keyphrases from similar documents (i.e., retrieved keyphrases) can give useful knowledge to guide the generation of keyphrases for the given document. More concretely, we encode the retrieved keyphrases as vector representations and use them as an external memory for the decoder of the generative model in our multi-task learning framework. Besides providing external knowledge, the retrieved keyphrases themselves are regarded as a kind of keyphrase prediction and can be utilized by the merging module.

Finally, to imitate the integrated keyphrase assignment process of humans more comprehensively, we further exploit the extractive model and the retrieved keyphrases by proposing a merging module. This merging module collects and re-ranks the predictions from our aforementioned components. First, keyphrase candidates are collected from three different sources: (1) keyphrases generated by the enhanced generative model; (2) keyphrases extracted by the extractive model; and (3) the retrieved keyphrases. Then, we design a neural-based merging algorithm to merge and re-rank all the keyphrase candidates, and consequently return the top-ranked candidates as our final keyphrases.

We extensively evaluate the performance of our proposed approach on five popular benchmarks. Experimental results demonstrate the effectiveness of the extractive model and the retrieved keyphrases in our multi-task learning framework. Furthermore, after introducing the merging module, our integrated approach consistently outperforms all the baselines and becomes the new state-of-the-art approach for keyphrase generation.

In summary, our main contributions include: (1) a new multi-task learning framework that leverages an extractive model and external knowledge to improve keyphrase generation; (2) a novel neural-based merging module that combines the predicted keyphrases from extractive, generative, and retrieval methods to further improve the performance; and (3) the new state-of-the-art performance on five real-world benchmarks.

2 Related Work

2.1 Automatic Keyphrase Extraction

Keyphrase extraction focuses on predicting the keyphrases that are present in the source text. Existing methods can mainly be categorized into two-step extraction approaches and sequence labeling models. Two-step extraction approaches first identify a set of candidate phrases from the document using different heuristics, such as the phrases that match specific part-of-speech (POS) tags (Liu et al., 2011; Wang et al., 2016; Le et al., 2016). Then, they learn a score for each candidate and select the top-ranked candidates as predicted keyphrases. The scores can be learned by either supervised methods with hand-crafted textual features (Medelyan et al., 2009; Witten et al., 1999; Nguyen and Kan, 2007; Frank et al., 1999; Hulth, 2003) or unsupervised graph ranking methods (Mihalcea and Tarau, 2004; Grineva et al., 2009; Wan and Xiao, 2008). Sequence labeling models are built on a recurrent neural network to sequentially go through a source text and learn the likelihood of each word in the source text to be a keyphrase word (Zhang et al., 2016; Luan et al., 2017; Gollapalli et al., 2017). In contrast to these extractive methods, our approach can generate both absent and present keyphrases.

2.2 Automatic Keyphrase Generation

Keyphrase generation aims at predicting both present and absent keyphrases for a source text. Meng et al. (2017) proposed CopyRNN, which is built on the attentional encoder-decoder model (Bahdanau et al., 2014) with copy mechanism (Gu et al., 2016) to generate keyphrases. CorrRNN (Chen et al., 2018a), an extension of CopyRNN, was proposed to model the correlations among keyphrases. This model utilizes hidden states and attention vectors of previously generated keyphrases to avoid generating repetitive keyphrases. The title information of the source text was explicitly exploited by Ye and Wang (2018) and Chen et al. (2018b) to further improve the performance. Ye and Wang (2018) first considered a semi-supervised setting for keyphrase generation. In contrast, inspired by Hsu et al. (2018) and Cao et al. (2018), we enhance existing generative methods by adopting an extractive model to assist the copy mechanism and exploiting external knowledge from retrieved keyphrases to help the generation. Furthermore, we also design a merging module to combine the predictions from different components.

3 Our Methodology

As shown in Figure 2, our integrated framework consists of a retriever, two encoders, an extractor, a decoder, and a merging module. Given a document $\mathbf{x}$, the retriever returns the keyphrases $\mathbf{r}$ retrieved from the training corpus. In addition to acting as keyphrase candidates, these retrieved keyphrases are also exploited to provide external guidance for the decoder. Then keyphrase extraction and generation are jointly conducted by the extractor and the decoder through sharing an encoder. Besides extracting keyphrase candidates, the importance scores of the source text words, β, predicted by the extractor are also employed to rescale the original copy probability distribution of the decoder. Thus, they can help the copy mechanism to detect important words more accurately. Finally, the merging module merges the candidates from three different sources (i.e., the retrieved, extracted, and generated candidates) and output the final predictions.

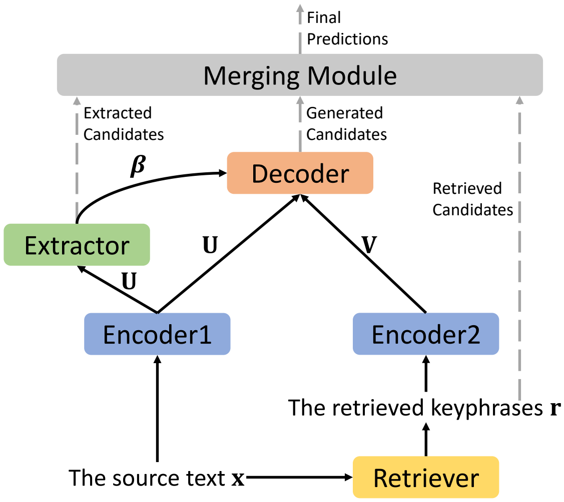


Figure 2: Our integrated framework. The “Encoder1” and the “Extractor” compose our extractive model. Our generative model mainly includes the “Encoder1”, the “Encoder2”, and the “Decoder”.

3.1 Retriever

Given a document $\mathbf{x}$, the retriever module retrieves top K (document, keyphrases) pairs from the training corpus. The retrieval is based on the Jaccard similarities of the non-stop-word sets between $\mathbf{x}$ and the corpus documents. After that, the keyphrases of the top K pairs are returned and used in the later modules in two ways. First, these retrieved keyphrases are regarded as the keyphrase candidates of $\mathbf{x}$ and directly fed into the final merging module. In addition, these keyphrases are concatenated together as a guidance input $\mathbf{r}$ for the decoder to provide useful external knowledge for the generation process. A separator token is inserted among keyphrases when concatenating them together.

3.2 Joint Extraction and Generation

We propose a multi-task learning framework which simultaneously learns to extract keyphrases from the source text and generate keyphrases word by word with external knowledge. Before describing in detail, we first define the tasks of the extraction and the generation.

3.2.1 Problem Definition

The inputs of the multi-task learning framework are the source text $\mathbf{x}$ and the concatenated retrieved keyphrases $\mathbf{r}$. Both $\mathbf{x}$ and $\mathbf{r}$ are a sequence of tokens (i.e., $\mathbf{x} = [x_1, ..., x_{L_\mathbf{x}}]$, $\mathbf{r} = [r_1, ..., r_{L_\mathbf{r}}]$), where $L_\mathbf{x}$ and $L_\mathbf{r}$ are the length of $\mathbf{x}$ and $\mathbf{r}$ respectively. The output of the extractor is a sequence of importance scores $\boldsymbol{\beta} = [\beta_1, ..., \beta_{L_\mathbf{x}}]$, where β_i is the probability of the i-th source word of being a keyphrase word. The output of the generator is a set of keyphrases $\mathcal{Y} = \{\mathbf{y}^i\}_{i=1,..,N}$, where N is the keyphrase number of $\mathbf{x}$ and $\mathbf{y}^i = [y_1^i, ..., y_{L_{\mathbf{y}^i}}^i]$ is a token sequence with length $L_{\mathbf{y}^i}$.

To fit the encoder-decoder framework, N tuples $\{(\mathbf{x}, \mathbf{r}, \boldsymbol{\beta}^*, (\mathbf{y}^i)^*)\}_{i=1,...,N}$ are split during training, where $\boldsymbol{\beta}^*$ and $(\mathbf{y}^i)^*$ are the gold binary importance scores and one of the gold keyphrases of $\mathbf{x}$ correspondingly. For simplicity, we adopt $(\mathbf{x}, \mathbf{r}, \boldsymbol{\beta}^*, \mathbf{y}^*)$ to represent such a tuple.

3.2.2 Encoders

Two encoders are employed in our multi-task learning framework. One is for the source text encoding (i.e., "Encoder1" in Figure 2) and the other is for retrieved keyphrases encoding (i.e., "Encoder2" in Figure 2). Both of them employ a bidirectional GRU (Cho et al., 2014) layer to obtain a context-aware representation of each word:

$$\mathbf{u}_i = \text{BiGRU}_1(\mathbf{x}_i, \overrightarrow{\mathbf{u}}_{i-1}, \overleftarrow{\mathbf{u}}_{i+1}), \quad (1)$$

$$\mathbf{v}_j = \text{BiGRU}_2(\mathbf{r}_j, \overrightarrow{\mathbf{v}}_{j-1}, \overleftarrow{\mathbf{v}}_{j+1}), \quad (2)$$

where $i = 1, 2, ..., L_\mathbf{x}$ and $j = 1, 2, ..., L_\mathbf{r}$. $\mathbf{x}_i$ and $\mathbf{r}_j$ are the d_e-dimensional embedding vectors of the i-th source text word x_i and j-th retrieved keyphrases word r_j respectively. $\mathbf{u}_i = [\overrightarrow{\mathbf{u}}_i; \overleftarrow{\mathbf{u}}_i] \in \mathbb{R}^d$ and $\mathbf{v}_j = [\overrightarrow{\mathbf{v}}_j; \overleftarrow{\mathbf{v}}_j] \in \mathbb{R}^d$ are regarded as the corresponding context-aware representations, where d is the hidden size of the biderectional GRU layer. Finally, we obtain the internal memory bank $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}]$ for later extraction and generation, and the external memory bank $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}]$ for later generation.

3.2.3 Extractor

Based on the internal memory bank, we use the following sequence identifier as our extractor to identify whether the word is a keyphrase word in the source text. We denote the importance score $P(\beta_j = 1 | \mathbf{u}_j, \mathbf{s}_j, \mathbf{d})$ as β_j for simplicity:

$$\begin{aligned}\beta_j = \text{sigmoid}(&\mathbf{W}_c \mathbf{u}_j + \mathbf{u}_j^T \mathbf{W}_s \mathbf{d} \\ &- \mathbf{u}_j^T \mathbf{W}_n \text{tanh}(\mathbf{s}_j) + b),\end{aligned} \quad (3)$$

where $\mathbf{d} = \text{tanh}(\mathbf{W}_d[\overrightarrow{\mathbf{u}}_{L_\mathbf{x}}; \overleftarrow{\mathbf{u}}_1] + \mathbf{b})$ is the global document representation and $\mathbf{s}_j = \sum_{i=1}^{j-1} \mathbf{u}_i \beta_i$ is current summary representation. $\mathbf{W}_c, \mathbf{W}_s$, and $\mathbf{W}_n$ are the content, salience and novelty weights respectively. Although this extractor is inspired by Nallapati et al. (2017), our extractor identifies important words instead of sentences within the source text.

3.2.4 Decoder

In addition to the internal memory bank $[\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}]$, our decoder employs the external memory bank $[\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}]$ to provide external guidance for the generation process. We exploit a decoder equipped with attention and copy mechanisms (Luong et al., 2015; See et al., 2017) to generate keyphrases. This decoder mainly consists of a forward GRU layer:

$$\mathbf{h}_t = \overrightarrow{\text{GRU}}([\mathbf{e}_{t-1}; \tilde{\mathbf{h}}_{t-1}], \mathbf{h}_{t-1}), \quad (4)$$

$$\mathbf{c}_t^{in} = \text{attn}(\mathbf{h}_t, [\mathbf{u}_1, ..., \mathbf{u}_{L_\mathbf{x}}], \mathbf{W}_{in}), \quad (5)$$

$$\mathbf{c}_t^{ex} = \text{attn}(\mathbf{h}_t, [\mathbf{v}_1, ..., \mathbf{v}_{L_\mathbf{r}}], \mathbf{W}_{ex}), \quad (6)$$

$$\tilde{\mathbf{h}}_t = \text{tanh}(\mathbf{W}_1[\mathbf{c}_t^{in}; \mathbf{c}_t^{ex}]; \mathbf{h}_t), \quad (7)$$

where $\mathbf{e}_{t-1}$ is the embedding vector of the $(t-1)$-th predicted word. The "attn" operation in Eq. (5) is defined as $\mathbf{c}_t^{in} = \sum_{i=1}^{L_\mathbf{x}} \alpha_{t,i}^{in} \mathbf{u}_i$, where $\alpha_{t,i}^{in} = \exp(s_{t,i}) / \sum_{j=1}^{L_\mathbf{x}} \exp(s_{t,j})$ and $s_{t,i} = (\mathbf{h}_t)^T \mathbf{W}_{in} \mathbf{u}_i$. Similarly, we can obtain the external aggregated vector $\mathbf{c}_t^{ex}$.

Then, the final predicted probability distribution at the current time step is:

$$P(y_t) = (1 - g_t) P_v(y_t) + g_t P_c(y_t), \quad (8)$$

where $g_t = \sigma(\mathbf{w}_g^T \tilde{\mathbf{h}}_t + b_g) \in \mathbb{R}$ is the soft switch between generating from the predefined vocabulary V and copying from X that are all words appearing in the source text. $P_v(y_t) = \text{softmax}(\mathbf{W}_2 \tilde{\mathbf{h}}_t + \mathbf{b}_v) \in \mathbb{R}^{|V|}$ is the generating probability distribution over V and $P_c(y_t) = \sum_{i:x_i = y_t} \alpha_{t,i}^c \in \mathbb{R}^{|X|}$ is the copying probability distribution over X. Previous work either directly uses the internal attention scores as the copy probabilities (i.e., $\alpha_{t,i}^c = \alpha_{t,i}^{in}$) or employs extra neural network layers to calculate new copy scores. But we employ the rescaled internal attention scores α_t^{in} by the importance scores $[\beta_1, ..., \beta_{L_{\mathbf{x}}}]$ from the extractor as the final copy probabilities:

$$\alpha_{t,i}^c = \frac{\alpha_{t,i}^{in} * \beta_i}{\sum_{j=1}^{L_{\mathbf{x}}} \alpha_{t,j}^{in} * \beta_j}. \tag{9}$$

The purpose of this rescaling is to provide extra guidance that which words within the source text are important and thus should obtain more attention when copying.

3.2.5 Joint Training

Finally, the summation of the following extraction loss and generation loss is used to train the whole joint framework in an end-to-end way.

Extraction Loss. We choose the source text words appearing in the assigned keyphrases as the gold important words and use the weighted cross-entropy loss for the extraction training i.e., $\mathcal{L}_e = -\frac{1}{L_{\mathbf{x}}} \sum_{j=1}^{L_{\mathbf{x}}} w\beta_j^* \log\beta_j + (1-\beta_j^*)\log(1-\beta_j)$, where $\beta_j^* \in \{0, 1\}$ is the ground-truth label for the j-th word and w is the loss weight for the positive training samples.

Generation Loss. The negative log likelihood loss is utilized for the generation training i.e., $\mathcal{L}_g = -\sum_{t=1}^{L_{\mathbf{y}^*}} \log P(y_t^*|\mathbf{y}_{t-1}, \mathbf{x}, \mathbf{r})$, where $\mathbf{y}_t = [y_1, ..., y_{t-1}]$ is the previously predicted word sequence, $L_{\mathbf{y}^*}$ is the length of target keyphrase $\mathbf{y}^*$, and y_t^* is the t-th target word in $\mathbf{y}^*$.

3.3 Merging Module

In this module, the retrieved, extracted and generated keyphrases are collected and then merged to produce the final keyphrase predictions.

3.3.1 Keyphrase Candidate Collection

Retrieved Candidate Collection. The retrieved keyphrases from the retriever are regarded as the retrieved candidates. Each retrieved candidate (rk) obtains a retrieval score (rs) that is the Jaccard similarity between the corresponding document and $\mathbf{x}$. The duplicates with lower retrieval scores are removed. Finally, we get $N_{\mathbf{rk}}$ retrieved keyphrase candidates $\mathbf{rk} = [rk_1, \ldots, rk_{N_{\mathbf{rk}}}]$ and their retrieval scores $\mathbf{rs} = [rs_1, \ldots, rs_{N_{\mathbf{rk}}}]$.

Extracted Candidate Collection. The extracted keyphrase candidates are from the extractor. We select the word x_j as a keyword if its importance score β_j is larger or equal than a threshold ϵ (i.e., $\beta_j \geq \epsilon$). The adjacent keywords compound a keyphrase candidate. If no other adjacent keywords, the keyword itself becomes a single-word keyphrase candidate. Each extracted keyphrase candidate (ek) is accompanied by an extraction score (es) that is the mean of the importance scores of the words within this candidate. Similarly, duplicates with lower extraction scores are removed. Consequently, we obtain $N_{\mathbf{ek}}$ extracted keyphrase candidates $\mathbf{ek} = [ek_1, \ldots, ek_{N_{\mathbf{ek}}}]$ and the corresponding extraction scores $\mathbf{es} = [es_1, \ldots, es_{N_{\mathbf{ek}}}]$.

Generated Candidate Collection. The generated keyphrase candidates directly come from the beam search process of the decoder. Each generated phrase is a keyphrase candidate. The beam search score of the generated candidate (gk) represents its generation score (gs). Duplicates with lower generation scores are removed. Then, we get $N_{\mathbf{gk}}$ generated candidates $\mathbf{gk} = [gk_1, \ldots, gk_{N_{\mathbf{gk}}}]$ and their generation scores $\mathbf{gs} = [gs_1, \ldots, gs_{N_{\mathbf{gk}}}]$.

3.3.2 Merging

In addition to the original importance scores (i.e., $\mathbf{rs}, \mathbf{es}, \mathbf{gs}$), we also employ an auxiliary *scorer* to assign an auxiliary importance score to each keyphrase candidate. Given a document-candidate pair ($\mathbf{x}$, candidate), the scorer should output the probability that the candidate is one of the keyphrases of $\mathbf{x}$. That means the scorer should determine the relationship between the given document $\mathbf{x}$ and the candidate, which is similar to a natural language inference (NLI) problem. Therefore, we adapt the most popular NLI model (Parikh et al., 2016) as our scorer. Different from typical natural language inference which is a multi-class classification problem, we use a binary classification setting to train the scorer. Besides, we learn the word embeddings and use two bi-directional GRU to obtain the input representations. The positive samples are the ground-truth keyphrases. The negative samples come from either the phrases in the document or the retrieved candidates. Notably, the ground-truth keyphrases are filtered when selecting negative samples. Consequently, a cross-entropy loss is utilized to train the scorer. Finally, the trained scorer is used to help the merging process as shown in Algorithm 1. The $\frac{u_{\mathbf{gs}}}{u_{\mathbf{rs}}}$ and $\frac{u_{\mathbf{gs}}}{u_{\mathbf{es}}}$ factors are used to enforce the average of $\mathbf{rs}$ and $\mathbf{es}$ to be the same with the average of $\mathbf{gs}$ and thus these three scores become comparable.

Algorithm 1 Merging Algorithm

```
Require: The retrieved, extracted and generated candidates $\mathbf{rk}, \mathbf{ek}, \mathbf{gk}$. The retrieval, extraction and generation scores $\mathbf{rs}, \mathbf{es}, \mathbf{gs}$; The average of each kind of score: $u_{\mathbf{rs}}, u_{\mathbf{es}}, u_{\mathbf{gs}}$; The trained $scorer$; The document $\mathbf{x}$.
Adjust $\mathbf{gs}$: $gs_i = gs_i \times scorer(\mathbf{x}, gk_i)$ where $i = 1, \ldots, N_{\mathbf{gk}}$.
Adjust $\mathbf{rs}$: $rs_i = rs_i \times \frac{u_{\mathbf{gs}}}{u_{\mathbf{rs}}} \times scorer(\mathbf{x}, rk_i)$ where $i = 1, \ldots, N_{\mathbf{rk}}$.
Adjust $\mathbf{es}$: $es_i = es_i \times \frac{u_{\mathbf{gs}}}{u_{\mathbf{es}}} \times scorer(\mathbf{x}, ek_i)$ where $i = 1, \ldots, N_{\mathbf{ek}}$.
Merge $\mathbf{rk}, \mathbf{ek}, \mathbf{gk}$: the final importance score of a candidate is the summation of its adjusted retrieval, extraction and generation scores. If not in $\mathbf{rk}$, $\mathbf{ek}$ or $\mathbf{gk}$, the corresponding scores are set to 0.
Sort all the candidates based on the final importance scores and then output the final predictions.
```

4 Experiment Settings

4.1 Datasets

Similar to Meng et al. (2017), we use **KP20k** dataset (Meng et al., 2017) to train our models. The released dataset contains 530,809 articles for training, 20,000 for validation, and the other 20,000 for testing. However, there exist duplicates in the **KP20k** training dataset with itself, the **KP20k** validation dataset, the **KP20k** testing dataset, and other four popular testing datasets (i.e., **Inspec** (Hulth, 2003), **Krapivin** (Krapivin et al., 2009), **NUS** (Nguyen and Kan, 2007), and **SemEval** (Kim et al., 2010)). After removing these duplicates, we maintain 509,818 articles in the training dataset. As for testing, following Meng et al. (2017), we employ five popular testing datasets from scientific publications as our testbeds for the baselines and our methods, which include **Inspec**, **Krapivin**, **NUS**, **SemEval**, and **KP20k**.

4.2 Baseline Models and Evaluation Metrics

For a comprehensive evaluation, we compare our methods with the traditional extractive baselines and the state-of-the-art generative methods. The extractive baselines include two unsupervised methods (i.e., TF-IDF and TextRank (Mihalcea and Tarau, 2004)) and one supervised method Maui (Medelyan et al., 2009). The generative baselines consist of CopyRNN (Meng et al., 2017) and CorrRNN (Chen et al., 2018a). We also conduct several ablation studies as follows:

- **KG-KE.** The joint extraction and generation model without using the retrieved keyphrases and merging process.
- **KG-KR.** The encoder-decoder generative model with retrieved keyphrases as external knowledge, but without combining with the extractive model and using the merging process.
- **KG-KE-KR.** The joint extraction and generation model with the retrieved keyphrases without using the merging process.

All the above ablation models directly use the generated candidates as the final predictions. We denote our final integrated method which combines all the proposed modules as **KG-KE-KR-M**.

Similar to CopyRNN and CorrRNN, we adopt macro-averaged *recall* (R) and *F-measure* (F_1) as our evaluation metrics. In addition, we also apply Porter Stemmer before determining whether two keyphrases are identical. Duplications are removed after stemming.

4.3 Implementation Details

We apply similar preprocessing procedures with Meng et al. (2017) including lowercasing, tokenizing and replacing digits with $\langle digit \rangle$ symbol. The title and the abstract of each article are concatenated as the source text input. We use the **KP20k** training dataset as the retrieval corpus. The implementations of our models are based on the OpenNMT system (Klein et al., 2017). The encoders, the decoder, and the scorer have the same vocabulary V with 50,000 tokens. The multi-task learning model and the scorer are trained separately.

The embedding dimension d_e and the hidden size d are set to 100 and 300 respectively. The initial state of the decoder GRU cell (i.e., $\mathbf{h}_0$) is set to $[\overrightarrow{\mathbf{u}}_{L_{\mathbf{x}}}; \overleftarrow{\mathbf{u}}_1]$. The other GRU cells are set to zero. The retrieval number K is set to 3 after evaluating the retrieved keyphrases on the evaluation dataset. When concatenating the retrieved keyphrases together as an external knowledge input, we use ';' as the separator among them. During training, all the trainable parameters including the embeddings are randomly initialized with uniform distribution in [-0.1, 0.1]. We engage Adam (Kingma and Ba, 2014) as the optimizer with positive extraction loss weight w=9.0, batch size=64, dropout rate=0.1, max gradient norm=1.0, initial learning rate=0.001. The training is early stopped when the validation perplexity stops dropping for several continuous evaluations. While testing, the beam search depth, and beam size are set as 6 and 200

Model	**Inspec**		**Krapivin**		**NUS**		**SemEval**		**KP20k**	
	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10
TF-IDF	0.188	0.269	0.092	0.120	0.103	0.142	0.076	0.135	0.087	0.113
TextRank	0.194	0.244	0.142	0.128	0.147	0.153	0.107	0.130	0.151	0.132
Maui	0.037	0.032	0.196	0.181	0.205	0.234	0.032	0.036	0.223	0.204
CorrRNN*	0.229_7	0.248_9	0.255_2	0.238_4	0.273_5	0.265_4	0.197_3	0.221_5	0.291_2	0.264_2
CopyRNN*	0.251_7	0.279_3	$\underline{0.268_4}$	0.243_1	0.275_2	0.268_2	0.190_6	0.214_5	0.306_1	0.273_0
KG-KE	$\underline{0.254_4}$	$\underline{0.281_2}$	0.265_3	0.240_1	0.278_4	0.273_1	$\mathbf{0.207_4}$	$\mathbf{0.227_7}$	0.307_0	0.274_0
KG-KR	0.244_2	0.275_1	0.266_5	$\underline{0.247_1}$	0.278_2	0.276_2	0.189_7	0.215_7	0.311_1	0.278_0
KG-KE-KR	0.245_1	0.278_4	0.267_3	0.246_2	$\underline{0.285_9}$	$\underline{0.279_4}$	0.194_4	0.220_2	$\underline{0.314_0}$	$\underline{0.280_0}$
KG-KE-KR-M	$\mathbf{0.257_2}$	$\mathbf{0.284_3}$	$\mathbf{0.272_3}$	$\mathbf{0.250_2}$	$\mathbf{0.289_4}$	$\mathbf{0.286_4}$	$\underline{0.202_6}$	$\underline{0.223_3}$	$\mathbf{0.317_0}$	$\mathbf{0.282_0}$

Table 1: Total keyphrase prediction results on all testing datasets. The best results are bold and the second best results are underlined. The subscripts are corresponding standard deviations for neural-based models (e.g. 0.257_2 means 0.257±0.002). The ‘*’ indicates our implementations based on Luong et al. (2015) attention and See et al. (2017) copying. The implementations of our proposed models are based on “CopyRNN*”.

Model	**Inspec**	**Krapivin**	**NUS**	**SemEval**	**KP20k**
TF-IDF	0.141	0.069	0.069	0.043	0.064
TextRank	0.158	0.110	0.094	0.062	0.110
Maui	0.024	0.162	0.161	0.012	0.196
CorrRNN*	0.172_6	0.217_6	0.212_4	0.125_3	0.268_3
CopyRNN*	0.195_7	$\underline{0.229_3}$	0.216_8	0.120_8	0.285_1
KG-KE	$\underline{0.197_3}$	0.225_3	0.219_3	$\mathbf{0.135_5}$	0.287_0
KG-KR	0.190_1	0.228_5	0.222_7	0.120_6	0.293_0
KG-KE-KR	0.191_3	$\underline{0.229_3}$	$\underline{0.224_5}$	0.127_5	$\underline{0.295_1}$
KG-KE-KR-M	$\mathbf{0.201_2}$	$\mathbf{0.234_2}$	$\mathbf{0.234_6}$	$\underline{0.131_4}$	$\mathbf{0.299_0}$

Table 2: MAP@10 scores of total keyphrase predictions. The best results are bold and the second best results are underlined. The meanings of the subscripts and the ‘*’ are the same as in Table 1.

correspondingly. The extraction threshold ϵ is set to 0.7 after evaluating the extracted keyphrases on the evaluation dataset. Notably, the stemmer is not applied to the gold keyphrases of **SemEval** testing dataset since they have already been stemmed. We do not remove any single-word predictions for **KP20k** but only keep one single-word prediction for other testing datasets. The averaged results of three different random seed are reported[1].

5 Results and Analysis

5.1 Total Keyphrase Prediction

Unlike the previous works which only separately analyze the present and absent keyphrase prediction ability, we also compare the whole keyphrase prediction ability regardless of the presence or absence of keyphrases, which is more reasonable in real applications. We show the F_1 scores at the top 5 and 10 predictions on Table 1.

This table displays our KG-KE-KR-M method consistently outperforms the state-of-the-art models CopyRNN and CorrRNN demonstrating the effectiveness of our method. Moreover, we also observe that the KG-KE model exceeds CopyRNN and CorrRNN on most datasets, which indicates the strength of our combination with the extractive model. Besides, we also see the KG-KR model perform comparably or better than the baselines, suggesting the effective guidance ability of the retrieved keyphrases. In addition, after combining these two ideas, the KG-KE-KR model surpasses both or one of KG-KE and KG-KR on all datasets, which shows the effectiveness of the combination with extraction model and the retrieved keyphrases again. Finally, the performance gap between KG-KE-KR and KG-KE-KR-M implies the power of our merging module. For mean average precision (MAP) metric which considers prediction orders, we obtain similar conclusions as shown in Table 2.

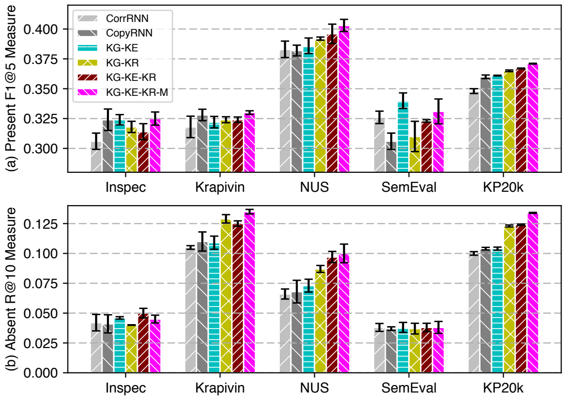


Figure 3: The present and absent keyphrase prediction performance of all neural-based methods.

5.2 Present and Absent Keyphrase Prediction

In this section, we analyze the performance of present and absent keyphrase prediction. Only

---

[1] Our code is available at https://github.com/Chen-Wang-CUHK/KG-KE-KR-M

Candidate Sources	Total F_1@10	Present F_1@5	Absent R@10
$\mathbf{gk}, \mathbf{ek}, \mathbf{rk}$	**0.250±0.002**	**0.330±0.002**	**0.172±0.002**
$\mathbf{gk}, \mathbf{ek}$	0.249±0.003	0.328±0.003	0.154±0.002
$\mathbf{gk}, \mathbf{rk}$	0.249±0.002	0.329±0.002	0.172±0.002
$\mathbf{gk}$	0.248±0.003	0.327±0.003	0.154±0.002
$\mathbf{gk}$, no merging	0.246±0.002	0.324±0.002	0.158±0.002
$\mathbf{ek}$, no merging	0.152±0.005	0.226±0.010	N/A
$\mathbf{rk}$, no merging	0.093±0.000	0.121±0.000	0.107±0.000

Table 3: Ablation study of the candidate sources of Algorithm 1 on Krapivin dataset. "no merging" means we do not use the merging algorithm.

Scoring Method	Total F_1@10	Present F_1@5	Absent R@10
Combined	**0.250±0.002**	**0.330±0.002**	**0.172±0.002**
Only $\mathbf{gs}, \mathbf{es}, \mathbf{rs}$	0.248±0.003	0.325±0.003	0.166±0.003
Only *scorer*	0.210±0.005	0.291±0.006	0.106±0.005

Table 4: Ablation study of the scoring method of Algorithm 1 on Krapivin dataset. "Only $\mathbf{gs}, \mathbf{es}, \mathbf{rs}$" means we do not use the *scorer*. "Only *scorer*" represents we directly use the scores predicted by the *scorer* as the final importance scores.

the present (absent) predictions and gold present (absent) keyphrases are preserved for the corresponding evaluation. We use F_1@5 metric for present predictions and R@10 for absent predictions. Since the neural-based baselines are the state-of-the-art models, we focus on the comparison with them in this section. The results are depicted on Figure 3.

The main observations are similar to the conclusions of total keyphrase prediction. Besides, we also note that after incorporating retrieved keyphrases, KG-KR model achieves substantial improvement gains over baselines on absent keyphrase prediction on **Krapivin**, **NUS**, and **KP20k**. These results demonstrate that the retrieved keyphrases indeed help the model to understand the main topics of the given document since generating absent keyphrase is an abstractive process and requires more powerful text understanding abilities. We notice that the KG-KE-KR-M method does not outperform the KG-KE-KR model on absent keyphrase prediction on **Inspec** dataset. One potential reason is that the merging module only merges two sources for absent keyphrases (i.e., the generated and retrieved keyphrases) instead of three sources like the present keyphrases do. Hence, the improvement for the absent keyphrases from the merging module is less stable than that for the present keyphrases. Moreover, we find that after combining with the extraction model, the KG-KE model achieves a huge improvement gain over CopyRNN on present keyphrase prediction on **SemEval** dataset, which manifests such a combination can improve the keyphrase extraction ability of the generative model.

5.3 Ablation Study on Merging Module

We also conduct in-depth ablation studies on our merging module. The objectives of these ablation studies are to (1) evaluate the effects of different candidate sources (i.e., what kinds of candidates are merged), and (2) analyze the effects of different final importance score calculating methods.

Concerning candidate sources, we show the ablation study results on Table 3. When comparing "$\mathbf{gk}$" with "$\mathbf{gk}$, no merging", we can see that the merging algorithm improves the performance of total and present keyphrase predictions, but it degrades the performance of absent keyphrase prediction. These results indicate the trained *scorer* performs better on scoring present keyphrases than scoring absent keyphrases. One possible reason is that scoring absent keyphrases requires a stronger text understanding ability than scoring present keyphrases. However, as shown in the row of "$\mathbf{gk}, \mathbf{rk}$" on Table3, this problem can be solved by incorporating the retrieved keyphrases which provide external information to this module. Besides absent keyphrase prediction, it is observed that the retrieved keyphrases can also benefit the present keyphrase prediction. For the extracted keyphrases, as shown in the "$\mathbf{gk}, \mathbf{ek}$" row, they only improve the present keyphrase prediction ability and do not affect absent keyphrases as we anticipated.

Regarding the scoring method, we further explore the effects of not using or only using the *scorer* in Algorithm 1. We show the results on Table 4. From this table, we note that after removing the *scorer* (i.e., "Only $\mathbf{gs}, \mathbf{es}, \mathbf{rs}$"), both present and absent keyphrase prediction performance become worse, which demonstrates the effectiveness of the combination with the *scorer*. Moreover, if we totally ignore the previously obtained retrieval, extraction and generation scores, and only use the *scorer* to predict the final keyphrase importance score (i.e., "Only *scorer*"), we find the performance decreases dramatically, which indicates the indispensability of the previously obtained retrieval, extraction, and generation scores.

Approximating minimum power covers of ***intersecting families*** and directed edge connectivity problems . Given a (directed) graph with costs on the edges , the power of a node is the maximum cost of an edge leaving it , and the power of the graph is the sum of the powers of its nodes . . . We consider problems that seek to find a min power spanning subgraph G of g that satisfies a prescribed ***edge connectivity*** property . . . We give ***approximation algorithms*** with ratio o (k ln vertical bar v vertical bar) . Our algorithms are based on a more general o (ln vertical bar v vertical bar) approximation algorithm for the problem of finding a min power directed edge cover of an intersecting set family . . .

(a) Present Keyphrases {approximation algorithms; edge connectivity; intersecting families}
CopyRNN: 1. ***approximation algorithms***, 2. ***edge connectivity***, 3. algorithms, 4. set cover, 5. connectivity, . . .
Retrieved: 1. power, 2. graphs, 3. approximation, 4. ***edge connectivity***, 5. ***approximation algorithms***, . . .
KG-KE-KR: 1. ***approximation algorithms***, 2. ***edge connectivity***, 3. power, 4. set cover, 5. minimum power, . . . 7. ***intersecting families***, . . .
KG-KE-KR-M: 1. ***approximation algorithms***, 2. ***edge connectivity***, 3. minimum power, . . . 6. ***intersecting families***, . . .

(b) Absent Keyphrases {wireless networks; power minimization; directed graphs}
CopyRNN: 1. graph algorithms, 2. combinatorial problems, 3. computational complexity, 4. ***directed graphs***, 5. randomized algorithms, . . .
Retrieved: 1. wireless, 2. degree, 3. k connectivity, 4. tree augmentation, . . . 7. power assignment, 8. ***wireless networks***
KG-KE-KR: 1. graph algorithms, 2. ***directed graphs***, 3. graph theory, 4. randomized algorithms, 5. spanning tree, 6. ***wireless networks***, . . .
KG-KE-KR-M: 1. graph algorithms, 2. ***directed graphs***, 3. power assignment, 4. graph theory, 5. ***wireless networks***, . . .

Figure 4: A keyphrase prediction example of CopyRNN, KG-KE-KR, and KG-KE-KR-M. "Retrieved" is the retrieved keyphrases. The extracted keyphrases by the extractor of KG-KE-KR are underlined in the source text. Top 10 present and absent predictions are compared and some incorrect predictions are omitted for simplicity. The correct predictions are bold and italic.

5.4 Case Study

To illustrate the advantages of our proposed methods, we show an example of the present and absent keyphrase predictions in Figure 4. For fairness, we only compare with CopyRNN since our models are based on its implementation. From the results of the present keyphrase prediction, we find the extractor of the KG-KE-KR model successfully extracts all the present keyphrases from the source text, which shows the power of the extractor. With the help of the copy probability rescaling from the extractor, the KG-KE-KR model correctly predicts the keyphrase "intersecting families" which is not successfully predicted by CopyRNN and retrieved by the retriever. Moreover, by merging the extracted keyphrases into the final predictions, the KG-KE-KR-M model assigns a higher rank to this keyphrase (i.e., from 7 to 6). As for absent keyphrase prediction, we note that KG-KE-KR successfully predicts the keyphrase "wireless networks" while CopyRNN fails. Since the retriever successfully retrieves this absent keyphrase, it shows that the retrieved keyphrases can provide effective external guidance for the generation process. Furthermore, the KG-KE-KR-M method assigns a higher rank to this keyphrase after merging the retrieved keyphrases into the final predictions (i.e., from 6 to 5). The overall results demonstrate the effectiveness of our proposed methods.

6 Conclusion and Future Work

In this paper, we propose a novel integrated approach for keyphrase generation. First, an end-to-end multi-task learning framework is introduced, which not only combines the keyphrase extraction and generation but also leverages the retrieved keyphrases from similar documents to guide the generation process. Furthermore, we introduce a neural-based merging algorithm to merge the candidates from three different components. Comprehensive empirical studies demonstrate the effectiveness of our approach. One interesting future work is to incorporate the similar documents themselves into keyphrase generation.

Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 of the General Research Fund) and Meitu (No. 7010445). We would like to thank Jiani Zhang for her comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1162–1170.

Ziqiang Cao, Wenjie Li, Furu Wei, and Sujian Li. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

152–161. Association for Computational Linguistics.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018a. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4057–4066.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2018b. Title-guided encoding for keyphrase generation. *CoRR*, abs/1808.08575.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 668–673.

Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3180–3187.

Maria P. Grineva, Maxim N. Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 661–670.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition, 4th International Conference, MLDM 2005, Leipzig, Germany, July 9-11, 2005, Proceedings*, pages 265–274.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, July 11-12, 2003*.

Anette Hulth and Beáta Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 21–26.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, System Demonstrations*, pages 67–72.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. Technical report, University of Trento.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In *AI 2016: Advances in Artificial Intelligence - 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings*, pages 665–671.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011*, pages 135–144.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2641–2651.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3075–3081.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007, Proceedings*, pages 317–326.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 855–860.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1395–1405.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. PTR: phrase-based topical ranking for automatic keyphrase extraction in scientific publications. In *Neural Information Processing - 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16-21, 2016, Proceedings, Part IV*, pages 120–128.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 347–354.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries, August 11-14, 1999, Berkeley, CA, USA*, pages 254–255.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4142–4153.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. 2018. Generating diverse numbers of diverse keyphrases. *CoRR*, abs/1810.05241.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 836–845.

Yongzheng Zhang, A. Nur Zincir-Heywood, and Evangelos E. Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems*, 2(1):39–53.

Predicting Malware Attributes from Cybersecurity Texts

Arpita Roy[1], **Youngja Park**[2], **Shimei Pan**[1]
[1]University of Maryland, Baltimore County
Baltimore, Maryland, USA
{arpita2, shimei}@umbc.edu
[2]IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
young_park@us.ibm.com

Abstract

Text analytics is a useful tool for studying malware behavior and tracking emerging threats. The task of automated malware attribute identification based on cybersecurity texts is very challenging due to a large number of malware attribute labels and a small number of training instances. In this paper, we propose a novel feature learning method to leverage diverse knowledge sources such as small amount of human annotations, unlabeled text and specifications about malware attribute labels. Our evaluation has demonstrated the effectiveness of our method over the state-of-the-art malware attribute prediction systems.

1 Introduction

Securing computer systems has become a necessity for both organizations and individuals, as many cyber attacks result in devastating consequences. An outbreak of the WannaCry ransomware in 2017 affected more than 200,000 computers across 150 countries, with total damages ranging from hundreds of millions to billions of dollars (Berr, 2017). Detection of malware often relies on an understanding of the characteristics of malware behavior. To establish a standard for unambiguously characterizing malware, MAEC (Malware Attribute Enumeration and Characterization), a community-based project organized by MITRE, has specified a set of standard malware attributes (Kirillov et al., 2011). Based on MAEC, the actions of a malware can be categorized by four attributes: *ActionName*, *Capability*, *StrategicObjectives* and *TacticalObjectives*. *ActionName* specifies the actions taken by a malware. For example, "delete file" is a malware action that deletes existing files from affected systems. *Capability* defines the general capabilities of a malware. For example, "anti-removal" is a malware capability that prevents itself from being removed from a system. *StrategicObjectives* and *TacticalObjectives* are subcategories of *Capability* to capture more details. For example, a malware can have a *StrategicObjective* of "staging data for exfiltration" and a *TacticalObjective* of "moving data to a staging server". In total, MAEC specified 211 *ActionNames*, 20 *Capabilities*, 65 *StrategicObjectives*, and 148 *TacticalObjectives*.

The goal of this research is to automatically assign malware attribute labels based on cybersecurity texts. The task is challenging. The system needs to assign a large number of labels (444 in total). However, it is difficult to obtain sufficient training examples for each label since malware attribute labeling requires extensive cybersecurity knowledge and only domain experts can do this reliably. Given a large number of possible labels and a small number of training examples, typical supervised text classification techniques do not work well. In this work, we focus on incorporating additional knowledge sources to improve feature learning. The main contributions of this work include

1. Develop a novel ***malware attribute prediction system*** with the state of the art performance to automatically characterize malware behavior based on cybersecurity text.

2. Propose a novel ***Word Annotation Embedding (WAE) algorithm*** to encode diverse information from heterogeneous knowledge sources such as human annotations, raw texts and MAEC specifications.

Proceedings of NAACL-HLT 2019, pages 2857–2861
Minneapolis, Minnesota, June 2 - June 7, 2019.

3. Since WAE generates embeddings for both words and malware attribute labels, we construct ***high-quality predicting features*** based on both types of embeddings.

2 Related Work

Cybersecurity researchers have recently recognized the benefits of leveraging information extracted from security documents. Most prior work utilizing NLP for cybersecurity has focused on privacy policy analysis and Android security (Peng et al., 2012; Pandita et al., 2013; Qu et al., 2014; Slavin et al., 2016; Zhu and Dumitras, 2016). These systems aim to map written policies and the actual permission requests from Android applications and assess the risk level of these applications.

Lim et al. (2017) represents the first major effort to apply NLP techniques for general text-based malware behavior analysis. It processes reports by cybersecurity companies (e.g., FireEye, IBM X-Force, Symantec and Trend Micro) on malware or campaigns associated with Advanced Persistent Threat (APT) groups (Blanda and Westcott, 2018) and assign attributes to identified malware actions. They use word unigrams as predicting features. SVM and Naive Bayes are used to build classifiers for attribute label prediction.

To extend this effort, SemEval organized a shared task (called SecureNLP) on semantic analysis for cybersecurity texts. It adopted the same dataset and task definitions as (Lim et al., 2017). There are four subtasks in SemEval SecureNLP: (1) identifying sentences containing malware actions from APT reports; (2) identifying "Malware Action", "Subject of Action", "Object of Action" and "Modifier of Action"in the identified sentences; (3) identifying four relations,"Subject-Action", "Action-Object", "Modifier-Action" and "Action-Modifier" in identified sentences; (4) assigning attribute labels to each identified action based on the MAEC specification. Figure 1 shows an annotated example for these tasks. This paper describes our approach to solve subtask 4. The input to our system includes all the sentences identified in subtask 1 with additional labels for the entities identified in subtask 2. Each training and testing instance used in SemEval SecureNLP only contains a single malware action.

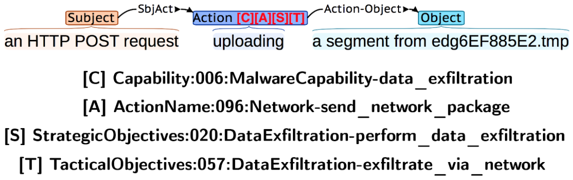


Figure 1: Annotated sentence fragment for SemEval shared task.

3 System Overview

Figure 2 shows the high-level system architecture. First, we augment all the raw APT reports with annotations to encode the knowledge from both the training data and MAEC. This allows us to design a unified representation for both types of knowledge. The annotated texts are then used by WAE to simultaneously learn embeddings for both words and malware attribute labels. The learned word and attribute label embeddings are then used to construct high qualify prediction features. Finally, we employ supervised machine learning to predict malware attribute labels. We build four classifiers, one for each malware attribute. Each classifier performs n+1-way classification, where n is the number of possible labels for each attribute and 1 is for 'no value' when the value of an attribute is not conveyed in the text.

4 Annotation Generation

To annotate text with additional information, the first step is to map each attribute label to a set of keywords based on both MAEC and the human annotations in the training data.

Identify Keywords from MAEC: Figure 3 shows a snippet of the MAEC specification. Each malware attribute label in MAEC includes a description and a few keywords. The malware action 004 in Figure 3 has a name 'emulate driver', a description 'specify the defined action of emulating an existing driver on a system' and two keywords: 'driver' and 'emulate'. Since the keywords carry the most essential information about a malware at-

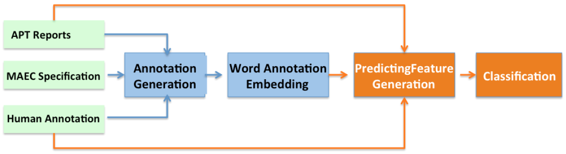
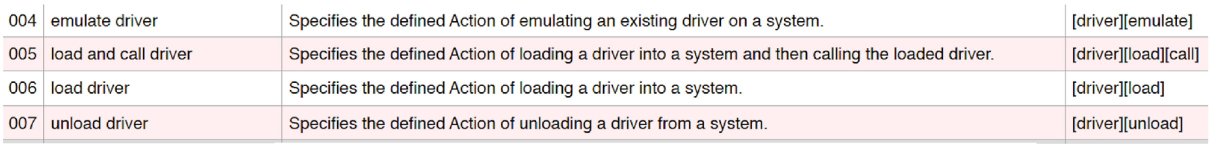


Figure 2: System Architecture



004	emulate driver	Specifies the defined Action of emulating an existing driver on a system.	[driver][emulate]
005	load and call driver	Specifies the defined Action of loading a driver into a system and then calling the loaded driver.	[driver][load][call]
006	load driver	Specifies the defined Action of loading a driver into a system.	[driver][load]
007	unload driver	Specifies the defined Action of unloading a driver from a system.	[driver][unload]

Figure 3: A Snippet of the MAEC Specification

tribute label, we link each label with these keywords (e.g., ActionName004: ‘driver’, ‘emulate’).
Identify Keywords from Training Data: Since a malware attribute label in the training data is at the sentence level, to extract keywords for each attribute label, we extract all the sentences associated with the same label and consider them one document. To select the most relevant keywords, we only keep those conveying “Malware Action”, “Subject of Action”, or “Object of Action” (which were identified in subtask 2).
Keywords Ranking: For the same attribute label, we merge the keywords from MAEC and those from the training data to form a single document. We then use TF-IDF scores to select the most informative keywords to differentiate these labels. In our experiments, we use the top 25 keywords based on their TF-IDF scores.
Text Annotation Generation Finally, for all the APT documents, we annotate the text with malware attribute labels. Specifically, for any word in the APT documents, if it is a keyword associated with K different labels, we annotate the word with K attribute labels.

5 Word Annotation Embedding (WAE)

Similar to word embedding (Mikolov et al., 2013), we want to learn features that capture the semantic relations between words. In addition, to facilitate attribute classification, we want to capture the semantic relations between words and their labels. Specifically, we want the words and their attribute labels to be close to each other in the embedding space and the embeddings of different labels to be far away from each other.

To achieve the goals, we developed a novel Word Annotation Embedding method. As shown in Figure 4, the target word is used to predict not only its context words but also its labels. To further strengthen their relations, the labels of the target word are also used to predict the target word. Specifically, given a sequence of T words (W_1,..,W_t,...,W_T) and their annotations (($A_{1,1}$,...,A_{1,M_1}),...,($A_{T,1}$,...,A_{T,M_T})), the objective of the WAE model is to maximize the average log probability shown in Equation 1, where C is the size of the context window, W_t is the target word, W_{t+j} is a context word, M_t is number of annotations W_t has and $A_{t,k}$ is the k-th annotation of W_t.

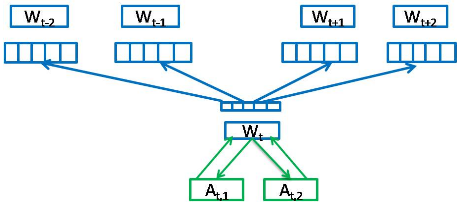


Figure 4: Architecture of WAE Model

$$\frac{1}{T}\sum_{t=1}^{T}\Big(\sum_{-C\leq j\leq C, j\neq 0}\log P(W_{t+j}|W_t) + \sum_{0\leq k\leq M_t}(\log P(A_{t,k}|W_t)+\log P(W_t|A_{t,k}))\Big) \qquad (1)$$

Label-aware Negative Sampling: Negative sampling (Mikolov et al., 2013) was introduced as an approximation method in word2vec to improve the efficiency of model training. Previously, negative samples were selected either randomly or based on popularity. The method, however, is insensitive to class labels. Here, we propose a new annotation-aware negative sampling method to (1) keep different annotations apart in the embedding space and (2) to keep words associated with different labels apart, in addition to bringing words and their associated labels closer to each other via positive samples. To achieve this, WAE randomly selects (1) a word as a negative sample if it does not share the same annotations as the target word; (2) an attribute label as a negative sample if it is not the same as the labels of the target word.

6 Feature Generation and Classification

We construct six sets of features to train the classifiers.
(S1) WAE_W*+Sim:* Assume the average word embeddings for a given data instance generated by WAE is WE_{wae}, and the malware attribute label

Models	**ActionName**	**Capability**	**StratObj**	**TactObj**
(B1) (Lim et al., 2017)	0.33	0.41	0.27	0.22
(B2) Word2vec	0.40	0.57	0.41	0.36
(B3) Word2vec+Cap	NA	NA	0.41	0.39
(S1) WAE_W+Sim	0.44	0.61	0.43	0.41
(S2) $w.WAE_W$+Sim	**0.46**	0.62	0.44	0.43
(S3) WAE_W+WAE_L+Sim	0.45	**0.63**	0.46	0.43
(S4) $w.WAE_W$+WAE_L+Sim	0.45	**0.63**	0.45	0.43
(s5) WAE_W+WAE_L+Sim+Cap	NA	NA	**0.47**	**0.45**
(S6) $w.WAE_W$+WAE_L+Sim+Cap	NA	NA	**0.47**	**0.45**
$\Delta 1$: over (Lim et al., 2017)	↑**39%** ($p < 0.05$)	↑**54%** ($p < 0.05$)	↑**74%**($p < 0.05$)	↑**105%** ($p < 0.05$)
$\Delta 2$: over word2vec	↑**15%** ($p < 0.05$)	↑**11%**($p < 0.05$)	↑**15%**($p < 0.05$)	↑**25%**($p < 0.05$)
$\Delta 3$: over word2vec+Cap	NA	NA	↑**15%** ($p < 0.05$)	↑**15%**($p < 0.05$)

Table 1: Evaluation Results. B1-B3 are the three baselines. S1-S6 are our models with six different sets of predicting features. Δ1-Δ3 are the improvements of our best models over the three baselines.

embedding learned by WAE is $LabelE_i$ for each label i. For each $LabelE_i$, we compute SIM_i, which is the cosine similarity between WE_{wae} and $LabelE_i$. WAE_W+*Sim* is the concatenation of WE_{wae} and all the SIM_i. For example, to predict *ActionName*, the model will include 100 word embedding features learned by WAE plus 211 similarity features, one for each *ActionNames*.

(S2) $w.WAE_W$*+Sim:* This feature set is similar to WAE_W+*Sim* except when computing WE_{wae}, we assign twice as much weight for a word with a label as one without a label. The intuition is words with labels are important keywords based on either MAEC or the training data.

(S3) WAE_W+WAE_L*+Sim:* It is similar to WAE_W+*Sim* except we also include the average embeddings of attribute labels associated with the instance.

(S4) $w.WAE_W$+WAE_L*+Sim:* This is the weighted version of (S3).

(S5) WAE_W+WAE_L*+Sim+Cap:* Since the label of *StrategicObjective* and *TacticalObjective* depends on the label of *Capability*, we added the *capability* label in the feature set. We use a 1-hot vector with 20 elements to encode a *Capability* label. We use the ground truth and the predicted label of *Capability* during training and testing respectively.

(S6) $w.WAE_W$+WAE_L*+Sim+Cap:* This is the weighted version of (S5)

7 Experiments

7.1 Dataset

The dataset used in the experiments was provided as a part of the SemEval shared task. It contains 456 APT reports (Blanda and Westcott, 2018), 39 of them were annotated by humans. Among them, 2975 sentences contain malware actions, which are the data instances used in this study. The annotated data are very sparse. Out of the 444 attribute labels, 190 labels do not appear in the labeled data. For the remaining 254 attribute labels, 92 labels occur less than five times, and 50 labels occur only once. In our experiments, we used the raw text in all the APT reports to train WAE. There are 16423 unique tokens and a total of 2544645 tokens in the dataset. We trained both word2vec and WAE with context window size 5 and 100 dimension vectors. The 39 annotated documents were divided into a training set (23 documents), a validation set (8 documents) and a test set (8 documents). Only the training dataset was used to generate annotations for WAE.

7.2 Evaluation Results

For classification, we tried both SVM and neural network-based models such as multilayer perceptron. After experimenting with different model parameters, we found that the best SVM model with a linear kernel performed slightly better than the best neural network models. We speculate that this might be because SVM is less likely to overfit when the training data are sparse. Table 1 shows the average F-scores over 5 runs by the SVM models on the test data. We compare our models with three baseline systems: (B1) (Lim et al., 2017), (B2) word2vec and (B3) word2vec + cap. Among them, (B1) represents the best published results on the same dataset. (B2) and (B3) are all the comparable models with embeddings learned by word2vec.

As shown in Table 1, all our models outperformed all the baseline systems. The improvement over the word2vec model is 15%, 11%, 15% and 25% respectively, and, the improvement over (Lim et al., 2017), a previous state of the art, is 39%, 54%, 74% and 105% respectively. The improvement over $Word2vec + Cap$ is 15% and 15% respectively for *StrategicObjective* and *TechnicalObjective*. We also conducted t-tests to verify the significance of the improvements. The t-test results confirmed that our models significantly outperformed the baseline models with $p<0.05$. Moreover, the value of ”Capability” seems to help the prediction of *StrategicObjective* and *TechnicalObjective*.

8 Conclusion

In this paper, we present a novel method to predict malware attribute labels from cybersecurity text. Given a large number of attribute labels and limited training data, we propose a new feature learning method to incorporate knowledge from diverse knowledge sources such as raw text, MAEC specifications and human annotations. We tested our system using the SemEval shared task data and our evaluation demonstrates that the features learned by our models are much more effective than an existing state of the art as well as embedding features learned by word2vec. Our investigation has highlighted the importance of incorporating diverse knowledge sources in complex classification tasks when human annotations are sparse.

References

Jonathan Berr. 2017. ”wannacry” ransomware attack losses could reach $4 billion. *CBSNews*.

Kiran Blanda and David Westcott. 2018. Aptnotes. `https://github.com/aptnotes/`.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2011. Malware attribute enumeration and characterization.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. Malwaretextdb: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1557–1567.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. 2013. Whyper: Towards automating risk assessment of mobile applications. In *USENIX Security Symposium*, volume 2013.

Hao Peng, Chris Gates, Bhaskar Sarma, Ninghui Li, Yuan Qi, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. 2012. Using probabilistic generative models for ranking risks of android apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS ’12.

Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. 2014. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1354–1365. ACM.

Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. 2016. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering*, pages 25–36. ACM.

Ziyun Zhu and Tudor Dumitras. 2016. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 767–778. ACM.

Improving Distantly-supervised Entity Typing with Compact Latent Space Clustering

Bo Chen[1], **Xiaotao Gu**[2], **Yufeng Hu**[1], **Siliang Tang**[1*], **Guoping Hu**[3],
Yueting Zhuang[1] **& Xiang Ren**[4]
[1]Zhejiang University, [2]University of Illinois at Urbana Champaign
[3]iFLYTEK Research, [4]University of Southern California,
{chenbo123, xiaofeem, siliang, yzhuang}@zju.edu.cn,
xiaotao2@illinois.edu, gphu@iflytek.com, xiangren@usc.edu

Abstract

Recently, distant supervision has gained great success on Fine-grained Entity Typing (FET). Despite its efficiency in reducing manual labeling efforts, it also brings the challenge of dealing with false entity type labels, as distant supervision assigns labels in a context-agnostic manner. Existing works alleviated this issue with partial-label loss, but usually suffer from confirmation bias, which means the classifier fit a pseudo data distribution given by itself. In this work, we propose to regularize distantly supervised models with Compact Latent Space Clustering (CLSC) to bypass this problem and effectively utilize noisy data yet. Our proposed method first dynamically constructs a similarity graph of different entity mentions; infer the labels of noisy instances via label propagation. Based on the inferred labels, mention embeddings are updated accordingly to encourage entity mentions with close semantics to form a compact cluster in the embedding space, thus leading to better classification performance. Extensive experiments on standard benchmarks show that our CLSC model consistently outperforms state-of-the-art distantly supervised entity typing systems by a significant margin.

1 Introduction

Recent years have seen a surge of interests in fine-grained entity typing **(FET)** as it serves as an important cornerstone of several nature language processing tasks including relation extraction (Mintz et al., 2009), entity linking (Raiman and Raiman, 2018), and knowledge base completion (Dong et al., 2014). To reduce manual efforts in labelling training data, distant supervision (Mintz et al., 2009) has been widely adopted by recent FET systems. With the help of an external knowledge base (KB), an entity mention is first linked to an existing entity in KB, and then labeled with all possible types of the KB entity as supervision. However, despite its efficiency, distant supervision also brings the challenge of **out-of-context noise**, as it assigns labels in a context agnostic manner. Early works usually ignore such noise in supervision (Ling and Weld, 2012; Shimaoka et al., 2016), which dampens the performance of distantly supervised models.

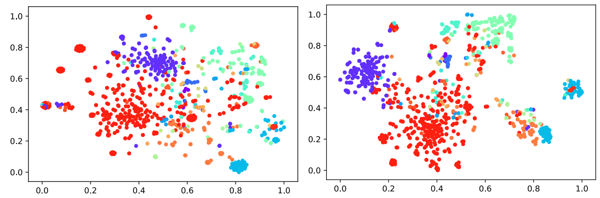


Figure 1: T-SNE visualization of the mention embeddings generated by NFETC (left) and CLSC (right) on the BBN dataset. Our model (CLSC) clearly groups mentions of the same type into a compact cluster.

Towards overcoming out-of-context noise, two lines of work have been proposed to distantly supervised FET. The first kind of work try to filter out noisy labels using heuristic rules (Gillick et al., 2014). However, such heuristic pruning significantly reduces the amount of training data, and thus cannot make full use of distantly annotated data. In contrast, the other thread of works try to incorporate such imperfect annotation by partial-label loss (**PLL**). The basic assumption is that, *for a noisy mention, the maximum score associated with its candidate types should be greater than the scores associated with any other non-candidate types* (Ren et al., 2016a; Abhishek et al., 2017; Xu and Barbosa, 2018). Despite their success, **PLL**-based models still suffer from ***Confirmation Bias*** by taking its own prediction as optimization objective in the next step. Specifically, given an entity mention, if the typing system selected a wrong

---

[*] Corresponding Author.

Proceedings of NAACL-HLT 2019, pages 2862–2872
Minneapolis, Minnesota, June 2 - June 7, 2019.

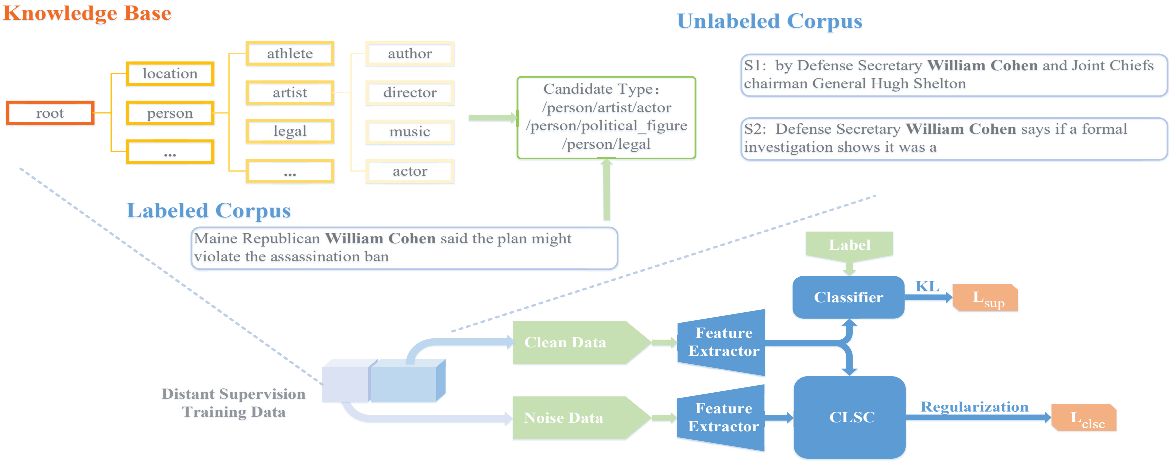


Figure 2: The overall framework of CLSC. We calculate classification loss only on clean data, while regularize the feature extractor with CLSC using both clean and noisy data.

type with the maximum score among all candidates, it will try to further maximize the score of the wrong type in following optimization epoches (in order to minimize **PLL**), thus amplifying the confirmation bias. Such bias starts from the early stage of training, when the typing model is still very suboptimal, and can accumulate in training process. Related discussion can be also found in the setting of semi-supervised learning (Lee et al., 2006; Laine and Aila, 2017; Tarvainen and Valpola, 2017).

In this paper, we propose a new method for distantly supervised fine-grained entity typing. Enlightened by (Kamnitsas et al., 2018), we propose to effectively utilize imperfect annotation as model regularization via **C**ompact **L**atent **S**pace **C**lustering (**CLSC**). More specifically, our model encourages the feature extractor to group mentions of the same type as a compact cluster (dense region) in the representation space, which leads to better classification performance. For training data with noisy labels, instead of generating pseudo supervision by the typing model itself, we dynamically construct a similarity-weighted graph between clean and noisy mentions, and apply label propagation on the graph to help the formation of compact clusters. Figure 1 demonstrates the effectiveness of our method in clustering mentions of different types into dense regions. In contrast to **PLL**-based models, we do not force the model to fit pseudo supervision generated by itself, but only use noisy data as part of regularization for our feature extractor layer, thus avoiding bias accumulation.

Extensive experiments on standard benchmarks show that our method consistently outperforms state-of-the-art models. Further study reveals that, the advantage of our model over the competitors gets even more significant as the portion of noisy data rises.

2 Problem Definition

Fine-grained entity typing takes a corpus and an external knowledge base (KB) with a type hierarchy $\mathcal{Y}$ as input. Given an entity mention (i.e., a sequence of token spans representing an entity) in the corpus, our task is to uncover its corresponding type-path in $\mathcal{Y}$ based on the context.

By applying distant supervision, each mention is first linked to an existing entity in KB, and then labeled with all its possible types. Formally, a labeled corpus can be represented as triples $\mathcal{D} = \{(m_i, c_i, \mathcal{Y}_i)\}_{i=1}^{n}$, where m_i is the i-th mention, c_i is the context of m_i, $\mathcal{Y}_i$ is the set of candidate types of m_i. Note that types in $\mathcal{Y}_i$ can form one or more type paths. In addition, we denote all terminal (leaf) types of each type path in $\mathcal{Y}_i$ as the target type set $\mathcal{Y}_i^t$ (*e.g.*, for $\mathcal{Y}_i = \{artist, teacher, person\}$, $\mathcal{Y}_i^t = \{artist, teacher\}$). This setting is also adopted by (Xu and Barbosa, 2018).

As each entity in KB can have several type paths, *out-of-context* noise may exist when $\mathcal{Y}_i$ contains type paths that are irrelevant to m_i in context c_i. In this work, we argue triples where $\mathcal{Y}_i$ contains only one type path (i.e., $|\mathcal{Y}_i^t| = 1$) as **clean data**. Other triples are treated as **noisy data**, where $\mathcal{Y}_i$ contains both the true type path and irrel-

Figure 3: The architecture of feature extractor $z((m_i, c_i); \theta_z)$

evant type paths. Noisy data usually takes a considerable portion of the entire dataset. The major challenge for distantly supervised typing systems is to incorporate both clean and noisy data to train high-quality type classifiers.

3 The Proposed Approach

Overview. The basic assumptions of our idea are: (1) all mentions belong to the same type should be close to each other in the representation space because they should have similar context, (2) similar contexts lead to the same type. For clean data, we compact the representation space of the same type to comply (1). For noisy data, given assumption (2), we infer the their type distributions via label propagation and candidate types constrain.

Figure 2 shows the overall framework of the proposed method. Clean data is used to train classifier and feature extractor end-to-endly, while noisy data is only used in CLSC regularization. Formally, given a batch of samples $\{(m_i, c_i, \mathcal{Y}_i^t)\}_{i=1}^B$, we first convert each sample (m_i, c_i) into a real-valued vector z_i via a feature extractor $z((m_i, c_i); \theta_z)$ parameterized by θ_z. Then a type classifier $g(z_i; \theta_g)$ parameterized by θ_g gives the posterior $P(y|z_i; \theta_g)$. By incorporating CLSC regularization in the objective function, we encourage the feature extractor z to group mentions of the same type into a compact cluster, which facilitates classification as is shown in Figure 1. Noisy data enhances the formation of compact clusters with the help of label propagation.

3.1 Feature Extractor

Figure 3 illustrates our feature extractor. For fair comparison, we adopt the same feature extraction pipeline as used in (Xu and Barbosa, 2018). The feature extractor is composed of an embedding layer and two encoders which encode mentions and contexts respectively.

Embedding Layer: The output of this layer is a concatenation of word embedding and word position embedding. We use the popular 300-dimensional word embedding supplied by (Pennington et al., 2014) to capture the semantic information and random initialized position embedding (Zeng et al., 2014) to acquire information about the relation between words and the mentions.

Formally, Given a word embedding matrix W_{word} of shape $d_w \times |V|$, where V is the vocabulary and d_w is the size of word embedding, each column of W_{word} represents a specific word w in V. We map each word w_j in (m_i, c_i) to a word embedding $\mathbf{w_j^d} \in R^{d_w}$. Analogously, we get the word position embedding $\mathbf{w}_j^p \in R^{d_p}$ of each word according to the relative distance between the word and the mention, we only use a fixed length context here. The final embedding of the j-th word is $\mathbf{w_j^E} = [\mathbf{w_j^d}, \mathbf{w_j^P}]$.

Mention Encoder: To capture lexical level information of mentions, an averaging mention encoder and a LSTM mention encoder (Hochreiter and Schmidhuber, 1997) is applied to encode mentions. Given $m_i = (w_s, w_{s+1}, \cdots, w_e)$, the averaging mention representation $r_{a_i} \in R^{d_w}$ is :

$$r_{a_i} = \frac{1}{e-s+1} \sum_{j=s}^{e} \mathbf{w_j^d} \tag{1}$$

By applying a LSTM over an extended mention $(w_{s-1}, w_s, w_{s+1}, \cdots, w_e, w_{e+1})$, we get a sequence $(h_{s-1}, h_s, h_{s+1}, \cdots, h_e, h_{e+1})$. We use h_{e+1} as LSTM mention representation $r_{l_i} \in R^{d_l}$. The final mention representation is $r_{m_i} = [r_{a_i}, r_{l_i}] \in R^{d_w+d_l}$.

Context Encoder: A bidirectional LSTM with d_l hidden units is employed to encode embedding sequence $(\mathbf{w^E_{s-W}}, \mathbf{w^E_{s-W+1}}, \cdots, \mathbf{w^E_{e+W}})$:

$$\begin{aligned}\overrightarrow{h_j} &= LSTM(\overrightarrow{h_{j-1}}, \mathbf{w^E_{j-1}}) \\ \overleftarrow{h_j} &= LSTM(\overleftarrow{h_{j-1}}, \mathbf{w^E_{j-1}}) \\ h_j &= [\overrightarrow{h_j} \oplus \overleftarrow{h_j}]\end{aligned} \tag{2}$$

where $\oplus$ denotes element-wise plus. Then, the word-level attention mechanism computes a score $\beta_{i,j}$ over different word j in the context c_i to get the final context representation r_{c_i}:

$$\begin{aligned}\alpha_j &= w^T tanh(h_j) \\ \beta_{i,j} &= \frac{exp(\alpha_j)}{\sum_k exp(\alpha_k)} \\ r_{c_i} &= \sum_j \beta_{i,j} h_{i,j}\end{aligned} \tag{3}$$

We use $r_i = [r_{m_i}, r_{c_i}] \in R^{d_z} = R^{d_w+d_l+d_l}$ as the feature representation of (m_i, c_i) and use a Neural Networks q over r_i to get the feature vector z_i. q has n layers with h_n hidden units and use ReLu activation.

3.2 Compact Latent Space Clustering for Distant Supervision

The overview of CLSC regularization is exhibited in Figure 4, which includes three steps: dynamic graph construction (Figure 4c), label propagation (Figure 4d, e) and Markov chains (Figure 4g). The idea of compact clustering for semi-supervised learning is first proposed by (Kamnitsas et al., 2018). The basic idea is to encourage mentions of the same type to be clustered into a dense region in the embedding space. We introduce more details of CLSC for distantly supervised FET in following sections.

Dynamic Graph Construction: We start by creating a fully connected graph G over the batch of samples $\mathbf{Z} = \{z_i\}_{i=1}^{B}$, as shown in Figure 4c[1]. Each node of G is a feature representation z_i, while the distance between nodes is defined by a scaled dot-product distance function (Vaswani et al., 2017):

$$\begin{aligned}A_{ij} &= exp(\frac{z_i^T z_j}{\sqrt{d_z}}), \forall z_i, z_j \in \mathbf{Z} \\ A &= exp(\frac{Z^T Z}{\sqrt{d_z}})\end{aligned} \tag{4}$$

Each entry A_{ij} measures the similarity between z_i and z_j, $A \in R^{B \times B}$ can be viewed as the weighted adjacency matrix of G.

Label Propagation: The end goal of CLSC is to cluster mentions of the same type to a dense region. For mentions which have more than one labeled types, we apply label propagation (**LP**) on G to estimate their type distribution. Formally, we denote $\mathbf{\Phi} \in R^{B \times K}$ as the label propagation posterior of a training batch.

The original label propagation proposed by (Zhu and Ghahramani, 2002) uses a transition matrix H to model the probability of a node i propagating its type posterior $\phi_i = P(y_i|x_i) \in R^K$ to the other nodes. Each entry of the transition matrix $H \in R^{B \times B}$ is defined as:

$$H_{ij} = A_{ij} / \sum_b A_{ib} \tag{5}$$

The original label propagation algorithm is defined as:

1. Propagate the label by transition matrix H, $\mathbf{\Phi}^{(t+1)} = H\mathbf{\Phi}^{(t)}$
2. Clamp the labeled data to their true labels. Repeat from step 1 until $\mathbf{\Phi}$ converges

In this work $\Phi^{(0)}$ is randomly initialized[2]. Unlike unlabeled data in semi-supervised learning, distantly labeled mentions in FET have a limited set of candidate types. Based on this observation, We assume that (m_i, c_i) can only transmit and receive probability of types in $\mathcal{Y}_i^t$ no matter it is noisy data or clean data. Formally, define a $B \times K$ indicator matrix $M \in R^{B \times K}$, where $M_{ij} = 1$ if type j in $\mathcal{Y}_i^t$ otherwise 0, where B is the batch size and K

---

[1] $\mathbf{Z} = \{z_i\}_{i=1}^{B}$ is a small subsample of the entire data, we didn't observe significant performance gain when the batch size increases.

[2] We also explored other initialization (e.g. uniform initialization), but found no essential performance difference between different initialization setups.

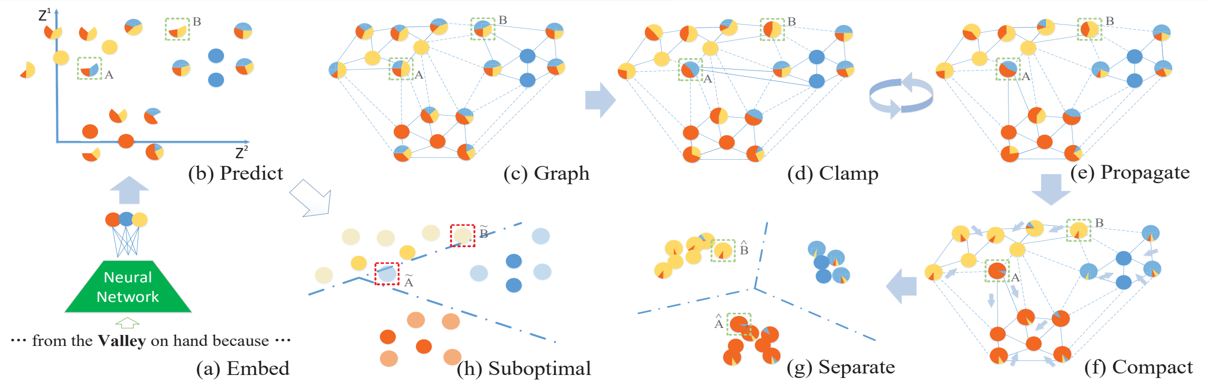


Figure 4: A demonstration of the CLSC process. (a) represents the feature extraction step; (b)→(h) shows the traditional type classification process (each color represents one candidate type), where suboptimal classifiers make predictions for each mention and misclassifies A into the Blue type; (c)→(d)→(e)→(f)→(g) demonstrates the process of CLSC as described in Section 3. Through label propagation and compact clustering, our model is able to group mentions of the same type into a dense region and leaves clear separation boundaries in sparse regions.

is the number of types. Our clamping step relies on M as is shown in Figure 4d:

$$\Phi_{ij}^{(t+1)} \leftarrow \Phi_{ij}^{(t+1)} M_{ij} / \sum_{k} \Phi_{ik}^{(t+1)} M_{ik} \tag{6}$$

For convenience, we iterate through these two steps S_{lp} times, S_{lp} is a hyperparameter.

Compact Clustering: The **LP** posterior $\mathbf{\Phi} = \mathbf{\Phi}^{(S_{lp}+1)}$ is used to judge the label agreement between samples. In the desired optimal state, transition probabilities between samples should be uniform inside the same class, while be zero between different classes. Based on this assumption, the desirable transition matrix $T \in R^{B\times B}$ is defined as:

$$T_{ij} = \sum_{k=1}^{K} \Phi_{ik} \frac{\Phi_{jk}}{m_k}, m_k = \sum_{b=1}^{B} \Phi_{bk} \tag{7}$$

m_k is a normalization term for class k. Transition matrix H derived from $z((m_i, c_i); \theta_z)$ should be in keeping with T. Thus we minimize the cross entropy between T and H:

$$\mathcal{L}_{1-step} = -\frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} T_{ij} log(H_{ij}) \tag{8}$$

For instance, if T_{ij} is close to 1, H_{ij} needs to be bigger, which results in the growth of A_{ij} and finally optimize θ_z (Eq.4). The loss $\mathcal{L}_{1-step}$ has largely described the regularization we use in $z((m_i, c_i); \theta_z)$ for compression clustering.

In order to keep the structure of existing clusters, (Kamnitsas et al., 2018) proposed an extension of $\mathcal{L}_{1-step}$ to the case of **Markov chains** with multiple transitions between samples, which should remain within a single class. The extension maximizes probability of paths that only traverse among samples belong to one class. Define $E \in R_{B\times B}$ as:

$$E = \mathbf{\Phi}^T \mathbf{\Phi} \tag{9}$$

E_{ij} measures the label similarities between z_i and z_j, which is used to mask the transition between different clusters. The extension is given by:

$$\begin{aligned} H^{(1)} &= H \\ H^{(s)} &= (H \odot E)^{(s-1)} H \\ &= (H \odot E) H^{(s-1)}, \end{aligned} \tag{10}$$

where $\odot$ is Hadamard Product, and $H_{ij}^{(s)}$ is the probability of a Markov process to transit from node i to node j after $s-1$ steps within the same class. The extended loss function models paths of different length s between samples on the graph:

$$\mathcal{L}_{clsc} = -\frac{1}{S_m} \frac{1}{B^2} \sum_{s=1}^{S_m} \sum_{i=1}^{B} \sum_{j=1}^{B} T_{ij} log(H_{ij}^{(s)}). \tag{11}$$

For $S_m = 1$, $\mathcal{L}_{clsc} = \mathcal{L}_{1-step}$. By minimizing the cross entropy between T and $H^{(s)}$ (Eq.11), $\mathcal{L}_{clsc}$ compact paths of different length between samples within the same class. Here, S_m is a hyper-parameter to control the maximum length of Markov chain. $\mathcal{L}_{clsc}$ is added to the final objective function as regularization to encourage compact cluttering.

3.3 Overall Objective

Given the representation of a mention, the type posterior is given by a standard softmax classifier parameterized by θ_g:

$$P(\hat{y}_i|z_i;\theta_g) = softmax(W_c z_i + b_c), \tag{12}$$

where $W_c \in R^{K \times d_z}$ is a parameter matrix, $b \in R^K$ is the bias vector, where K is the number of types. The predicted type is then given by $\hat{t}_i = argmax_{y_i} P(\hat{y}_i|z_i;\theta_g)$.

Our loss function consists of two parts. $\mathcal{L}_{sup}$ is supervision loss defined by KL divergence:

$$L_{sup} = -\frac{1}{B_c}\sum_{i=1}^{B_c}\sum_{k=1}^{K} y_{ik} log(P(y_i|z_i;\theta_g))_k \tag{13}$$

Here B_c is the number of clean data in a training batch, K is the number of target types. The regularization term is given by $\mathcal{L}_{clsc}$. Hence, the overall loss function is:

$$\mathcal{L}_{final} = \mathcal{L}_{sup} + \lambda_{clsc} \times \mathcal{L}_{clsc} \tag{14}$$

λ_{clsc} is a hyper parameter to control the influence of CLSC.

4 Experiments

4.1 Dataset

We evaluate our method on two standard benchmarks: OntoNotes and BBN:

- **OntoNotes:** The OntoNotes dataset is composed of sentences from the Newswire part of OntoNotes corpus (Weischedel et al., 2013). (Gillick et al., 2014) annotated the training part with the aid of DBpedia spotlight (Daiber et al., 2013), while the test data is manually annotated.
- **BBN:** The BBN dataset is composed of sentences from Wall Street Journal articles and is manually annotated by (Weischedel and Brunstein, 2005). (Ren et al., 2016a) regenerated the training corpus via distant supervision.

In this work we use the preprocessed datasets provided by (Abhishek et al., 2017; Xu and Barbosa, 2018). Table 2 shows detailed statistics of the datasets.

4.2 Compared Methods

We compare the proposed method with several state-of-the-art FET systems[3]:

- **Attentive** (Shimaoka et al., 2016) uses an attention based feature extractor and doesn't distinguish clean from noisy data;
- **AFET** (Ren et al., 2016a) trains label embedding with partial label loss;
- **AAA** (Abhishek et al., 2017) learns joint representation of mentions and type labels;
- **PLE+HYENA/FIGER** (Ren et al., 2016b) proposes heterogeneous partial-label embedding for label noise reduction to boost typing systems. We compare two PLE models with HYENA (Yogatama et al., 2015) and FIGER (Ling and Weld, 2012) as the base typing system respectively;
- **NFETC** (Xu and Barbosa, 2018) trains neural fine-grained typing system with hierarchy-aware loss. We compare the performance of the NFETC model with two different loss functions: partial-label loss and **PLL**+hierarchical loss. We denote the two variants as $\mathbf{NFETC}$ and $\mathbf{NFETC}_{hier}$ respectively;
- **NFETC-CLSC** is the proposed model in this work. We use the NFETC model as our base model, based on which we apply Compact Latent Space Clustering Regularization as described in Section 3.2; Similarly, we report results produced by using both KL-divergense-based loss (**NFETC-CLSC**) and **KL**+hierarchical loss ($\mathbf{NFETC\text{-}CLSC}_{hier}$).

4.3 Evaluation Settings

For evaluation metrics, we adopt strict accuracy, loose macro, and loose micro F-scores widely used in the FET task (Ling and Weld, 2012). To fine tuning the hyper-parameters, we randomly sampled 10% of the test set as a development set for both datasets. With the fine-tuned hyper-parameter as mentioned in 4.4, we run the model five times and report the average strict accuracy, macro F1 and micro F1 on the test set.

---

[3] The baselines result are reported on (Abhishek et al., 2017; Xu and Barbosa, 2018) in addition to performance of NFETC on BBN, we search the hyper parameters for it. (Xu and Barbosa, 2018) didn't report the results on BBN

Method		**OntoNotes**			**BBN**		
		Strict Acc.	**Macro F1**	**Micro F1**	**Strict Acc.**	**Macro F1**	**Micro F1**
AFET (Ren et al., 2016a)		55.3	71.2	64.6	68.3	74.4	74.7
AAA (Abhishek et al., 2017)		52.2	68.5	63.3	65.5	73.6	75.2
Attentive (Shimaoka et al., 2016)		51.7	71.0	64.91	48.4	73.2	72.4
PLE+HYENA (Ren et al., 2016b)		54.6	69.2	62.5	69.2	73.1	73.2
PLE+FIGER (Ren et al., 2016b)		57.2	71.5	66.1	68.5	77.7	75.0
NFETC	$clean$	54.4±0.3	71.5±0.4	64.9±0.3	71.2±0.2	77.1±0.3	76.9±0.3
	$+noisy$	54.8±0.4	71.8±0.4	65.0±0.4	73.8±0.6	78.4±0.6	78.9±0.6
NFETC$_{hier}$	$clean$	59.6±0.2	76.1±0.2	69.7±0.2	70.3±0.3	76.8±0.3	76.6±0.2
	$+noisy$	60.2±0.2	76.4±0.1	70.2±0.2	**73.9±1.2**	78.8±1.2	**79.4±1.1**
NFETC-CLSC	$clean$	59.1±0.4	75.3±0.3	69.1±0.3	73.0±0.3	79.0±0.3	78.8±0.3
	$+noisy$	59.6±0.2	75.5±0.4	69.3±0.4	**74.7±0.3**	**80.7±0.2**	**80.5±0.2**
NFETC-CLSC$_{hier}$	$clean$	61.5±0.3	77.4±0.3	71.4±0.4	70.5±0.2	78.2±0.2	78.0±0.2
	$+noisy$	**62.8±0.3**	**77.8±0.4**	**72.0±0.4**	71.9±0.3	79.8±0.4	79.5±0.3

Table 1: Performance comparision of FET systems on the two datasets.

	OntoNotes	**BBN**
#types	89	47
Max hierarchy depth	3	2
#mentions-train	253241	86078
#mentions-test	8963	12845
%clean mentions-train	73.13	75.92
%clean mentions-test	94.00	100
Average $\lvert\mathcal{Y}_i^t\rvert$	1.40	1.26

Table 2: Detailed statistics of the two datasets.

4.4 Hyper Parameters

We search the hyper parameter of Ontonotes and BBN respectively via Hyperopt proposed by (Bergstra et al., 2013). Hyper parameters are shown in **Appendix A**. We optimize the model via Adam Optimizer. The full hyper parameters includes the learning rate lr, the dimension d_p of word position embedding, the dimension d_l of the mention encoder's output (equal to the dimension of the context encoder's ourput), the input dropout keep probability p_i and output dropout keep probability p_o for LSTM layers (in context encoder and LSTM mention encoder), the L2 regularization parameter λ, the factor of hierarchical loss normalization α ($\alpha > 0$ means use the normalization), BN (whether using Batch normalization), the max step S_{lp} of the label propagation, the max length S_m of Markov chain, the influence parameter λ_{clsc} of CLSC, the batch size B, the number n of hidden layers in q and the number h_n of hidden units of the hidden layers. We implement all models using Tensorflow[4].

4.5 Performance comparison and analysis

Table 1 shows performance comparison between the proposed CLSC model and state-of-the-art FET systems. On both benchmarks, the CLSC model achieves the best performance in all three metrics. When focusing on the comparison between **NFETC** and CLSC, we have following observation:

- Compact Latent Space Clustering shows its effectiveness on both clean data and noisy data. By applying CLSC regularization on the basic **NFETC** model, we observe consistent and significant performance boost;
- Hierarchical-aware loss shows significant advantage on the OntoNotes dataset, while showing insignificant performance boost on the BBN dataset. This is due to different distribution of labels on the test set. The proportion of terminal types of the test set is 69% for the BBN dataset, while is only 33% on the OntoNotes dataset. Thus, applying hierarchical-aware loss on the BBN dataset brings little improvement;
- Both algorithms are able to utilize noisy data to improve performance, so we would like to further study their performance in different noisy scenarios in following discussions.

---

[4] The code for experiments is available at https://github.com/herbertchen1/NFETC-CLSC

4.6 How robust are the methods to the proportion of noisy data?

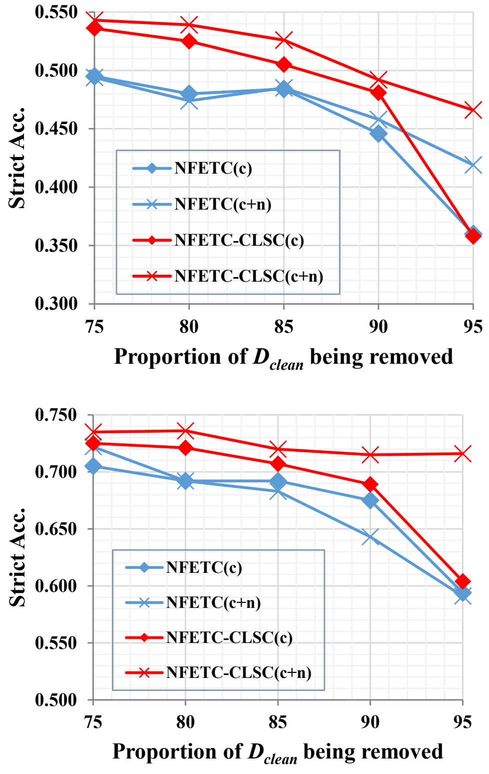


Figure 5: Performance comparison between NFETC-CLSC and NFETC by removing 75%-95% clean data.

By principle, with sufficient amount of clean training data, most typing systems can achieve satisfying performance. To further study the robustness of the methods to label noise, we compare their performance with the presence of $25\%, 20\%, 15\%, 10\%$ and 5% clean training data and all noisy training data. Figure 5 shows the performance curves as the proportion of clean data drops. As it reveals, the CLSC model consistently wins in the comparison. The advantage is especially clear on the BBN dataset, which offers less amount of training data. Note that, with only 27.9% of training data (when only leaving 5% clean data) on the BBN dataset, the CLSC model yield a comparable result with the **NFETC** model trained on full data. This comparison clearly shows the superiority of our approach in the effectiveness of utilizing noisy data.

4.7 Ablation: Do Markov Chains improve typing performance?

Table 3 shows the performance of CLSC with one-step transition ($\mathcal{L}_{1-step}$) and with Markov Chains ($\mathcal{L}_{clsc}$) as described in Section 3.2. Results show that the use of Markov Chains does bring improvement to the overall performance, which is consistent with the model intuition.

5 Related Work

Named entity Recognition (NER) has been excavated for a long time (Collins and Singer, 1999; Manning et al., 2014), which classifies coarse-grained types (e.g. person, location). Recently, (Nagesh and Surdeanu, 2018a,b) applied ladder network (Rasmus et al., 2015) to coarse-grained entity classification in a semi-supervised learning fashion. (Ling and Weld, 2012) proposed Fine-Grained Entity Recognition (FET). They used distant supervision to get training corpus for FET. Embedding techniques was applied to learn feature representations since (Yogatama et al., 2015; Dong et al., 2015). (Shimaoka et al., 2016) introduced attention mechanism for FET to capture informative words. (Xin et al., 2018a) used the TransE entity embeddings (Bordes et al., 2013) as the query vector of attention.
Early works ignore the out-of-context noise, (Gillick et al., 2014) proposed context dependent FET and use three heuristics to clean the noisy labels with the side effect of losing training data. To utilize noisy data, (Ren et al., 2016a) distinguished the loss function of noisy data from clean data via partial label loss (**PLL**). (Abhishek et al., 2017; Xu and Barbosa, 2018) proposed variants of **PLL**, which still suffer from confirmation bias. (Xu and Barbosa, 2018) proposed hierarchical loss to handle over-specific noise. On top of **AFET**, (Ren et al., 2016b) proposed a method **PLE** to reduce the label noise, which lead to a great success in FET. Because label noise reduction is separated from the learning of FET, there might be error propagation problem. Recently, (Xin et al., 2018b) proposed utilizing a pretrained language model measures the compatibility between context and type names, and use it to repel the interference of noisy labels. However, the compatibility got by language model may not be right and type information is defined by corpus and annotation guidelines rather than type names as is mentioned in (Azad et al., 2018). In addition, there are some work about entity-level typing which aim to figure out the types of entities in KB (Yaghoobzadeh and Schütze, 2015; Jin et al., 2018).

	Strict Acc.
CLSC(c)($\mathcal{L}_{1-step}$)	72.0±0.1
CLSC(c)($\mathcal{L}_{clsc}$)	73.0±0.3
CLSC(c+n)($\mathcal{L}_{1-step}$)	73.0±0.1
CLSC(c+n)($\mathcal{L}_{clsc}$)	74.7±0.3

Table 3: The comparison of $\mathcal{L}_{1-step}$ and $\mathcal{L}_{clsc}$ on BBN.

6 Conclusion

In this paper, we propose a new method for distantly supervised fine-grained entity typing, which leverages imperfect annotations as model regularization via Compact Latent Space Clustering (CLSC). Experiments on two standard benchmarks demonstrate that our method consistently outperforms state-of-the-art models. Further study reveals our method is more robust than the former state-of-the-art approach as the portion of noisy data rises. The proposed method is general for other tasks with imperfect annotation. As a part of future investigation, we plan to apply the approach to other distantly supervised tasks, such as relation extraction.

7 Acknowledgments

This work has been supported in part by NSFC (No.61751209, U1611461), Zhejiang University-iFLYTEK Joint Research Center, Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Engineering Research Center of Digital Library, Ministry of Education. Xiang Ren's research has been supported in part by National Science Foundation SMA 18-29268.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 797–807.

Amar Prakash Azad, Balaji Ganesan, Ashish Anand, Amit Awekar, et al. 2018. A unified labeling approach by pooling diverse datasets for entity typing. *arXiv preprint arXiv:1810.08782*.

James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20. Citeseer.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *IJCAI*, pages 1243–1249.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. 2018. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 282–292.

Konstantinos Kamnitsas, Daniel Castro, Loic Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya Nori. 2018. Semi-supervised learning via compact latent space clustering. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2459–2468, Stockholmsmssan, Stockholm Sweden. PMLR.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *Proc. International Conference on Learning Representations (ICLR)*.

Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. 2006. Fine-grained named entity recognition using conditional random fields for question answering. In *Asia Information Retrieval Symposium*, pages 581–587. Springer.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Ajay Nagesh and Mihai Surdeanu. 2018a. An exploration of three lightly-supervised representation learning approaches for named entity classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2312–2324.

Ajay Nagesh and Mihai Surdeanu. 2018b. Keep your bearings: Lightly-supervised information extraction with ladder networks that avoids semantic drift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 352–358.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: Multilingual entity linking by neural type system evolution. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5406–5413.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1825–1834. ACM.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018a. Improving neural fine-grained entity typing with knowledge attention.

Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2018b. Put it back: Entity typing with language model enhancement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 993–998.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 16–25.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 291–296.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation.

A Hyper parameters

	Ont.(C)	**BBN(C)**	**BBN(N)**
lr	0.0006	0.0007	0.0007
d_p	70	40	20
d_l	1000	1000	240
p_i	0.7	0.3	0.5
p_o	0.6	1.0	0.4
λ	0.0000	0.0000	0.0002
α	0.25/0.0	0.4/0.0	0.4/0.0
BN	FALSE	FALSE	TRUE
S_{lp}	200	200	-
S_m	8	12	-
λ_{clsc}	2.0	1.5	-
B	512	512	512
n	2	1	-
h_n	700	560	-

Table 4: Hyper parameters of our experiments: (C) denotes CLSC, (N) denotes the hyper parameter is used for NFETC.

Modelling Instance-Level Annotator Reliability for Natural Language Labelling Tasks

Maolin Li[1,2], **Arvid Fahlström Myrman**[2], **Tingting Mu**[2] and **Sophia Ananiadou**[1,2]
[1]National Centre for Text Mining, University of Manchester, United Kingdom
[2]School of Computer Science, University of Manchester, United Kingdom
`{maolin.li, arvidfm, tingting.mu, sophia.ananiadou}@manchester.ac.uk`

Abstract

When constructing models that learn from noisy labels produced by multiple annotators, it is important to accurately estimate the reliability of annotators. Annotators may provide labels of inconsistent quality due to their varying expertise and reliability in a domain. Previous studies have mostly focused on estimating each annotator's overall reliability on the entire annotation task. However, in practice, the reliability of an annotator may depend on each specific instance. Only a limited number of studies have investigated modelling per-instance reliability and these only considered binary labels. In this paper, we propose an unsupervised model which can handle both binary and multi-class labels. It can automatically estimate the per-instance reliability of each annotator and the correct label for each instance. We specify our model as a probabilistic model which incorporates neural networks to model the dependency between latent variables and instances. For evaluation, the proposed method is applied to both synthetic and real data, including two labelling tasks: text classification and textual entailment. Experimental results demonstrate our novel method can not only accurately estimate the reliability of annotators across different instances, but also achieve superior performance in predicting the correct labels and detecting the least reliable annotators compared to state-of-the-art baselines.[1]

1 Introduction

In many natural language processing (NLP) applications, the performance of supervised machine learning models depends on the quality of the corpus used to train the model. Traditionally, labels are collected from multiple annotators/experts who are assumed to provide reliable labels. However, in reality, these experts may have varying levels of expertise depending on the domains, and thus may disagree on labelling in certain cases (Aroyo and Welty, 2013). A rapid and cost-effective alternative is to obtain labels through crowdsourcing (Snow et al., 2008; Poesio et al., 2013, 2017). In crowdsourcing, each instance is presented to multiple expert or non-expert annotators for labelling. However, labels collected in this manner could be noisy, since some annotators could produce a significant number of incorrect labels. This may be due to differing levels of expertise, lack of financial incentive and interest (Poesio et al., 2017), as well as the tedious and repetitive nature of the annotation task (Raykar et al., 2010; Bonald and Combes, 2017).

Thus, in order to ensure the accuracy of the labelling and the quality of the corpus, it is crucial to estimate the reliability of the annotators automatically without human intervention.

Previous studies have mostly focused on evaluating the annotators' overall reliability (Gurevych and Kim, 2013; Sheshadri and Lease, 2013; Poesio et al., 2017). Measuring the reliability on a per-instance basis is however useful as we may expect certain annotators to have more expertise in one domain than another, and as a consequence certain annotation decisions will be more difficult than others. This resolves a potential issue of models that only assign an overall reliability to each annotator, where such a model would determine an annotator with expertise in a single domain to be unreliable for the model, even though the annotations are reliable within the annotator's domain of expertise.

Estimating per-instance reliability is also helpful for unreliable annotator detection and task allocation in crowdsourcing, where the cost of labelling data is reduced using proactive learn-

---

[1] Code is available at `https://github.com/createmomo/instance-level-reliability`

Proceedings of NAACL-HLT 2019, pages 2873–2883
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing strategies for pairing instances with the most cost-effective annotators (Donmez and Carbonell, 2008; Li et al., 2017). Although reliability estimation has been studied for a long time, only a limited number of studies have examined how to model the reliability of each annotator on a per-instance basis. Additionally, these in turn have only considered binary labels (Yan et al., 2010, 2014; Wang and Bi, 2017), and cannot be extended to multi-class classification in a straightforward manner.

In order to handle both binary and multi-class labels, our approach extends one of the most popular probabilistic models for label aggregation, proposed by Hovy et al. (2013). One challenge of extending the model is the definition of the label and reliability probability distributions on a per-instance basis. Our approach introduces a classifier which predicts the correct label of an instance, and a reliability estimator, providing the probability that an annotator will label a given instance correctly. The approach allows us to simultaneously estimate the per-instance reliability of the annotators and the correct labels, allowing the two processes to inform each other. Another challenge is to select appropriate training methods to learn a model with high and stable performance. We investigate training our model using the EM algorithm and cross entropy. For evaluation, we apply our method to six datasets including both synthetic and real-world datasets (see Section 4.1). In addition, we also investigate the effect on the performance when using different text representation methods and text classification models (see Section 4.2).

Our contributions are as follows: firstly, we propose a novel probabilistic model for the simultaneous estimation of per-instance annotator reliability and the correct labels for natural language labelling tasks. Secondly, our work is the first to propose a model for modelling per-instance reliability for both binary and multi-class classification tasks. Thirdly, we show experimentally how our method can be applied to different domains and tasks by evaluating it on both synthetic and real-world datasets. We demonstrate that our method is able to capture the reliability of each annotator on a per-instance basis, and that this in turn helps improve the performance when predicting the underlying label for each instance and detecting the least reliable annotators.

2 Related Work

2.1 Modelling Annotator Reliability

Probabilistic graphical models have been widely used for inferring the overall reliability of annotators in the absence of ground truth labels. Approaches include modelling a single overall reliability score for each annotator (Whitehill et al., 2009; Welinder et al., 2010; Karger et al., 2011; Liu et al., 2012; Demartini et al., 2012; Hovy et al., 2013; Rodrigues et al., 2014; Li et al., 2014a,b), estimating the reliability of each annotator on a per-category basis (Dawid and Skene, 1979; Zhou et al., 2012; Kim and Ghahramani, 2012; Zhang et al., 2014), and estimating the sensitivity and specificity for each annotator in binary classification tasks (Raykar et al., 2010).

Fewer attempts have been made to model the per-instance reliability of annotators, focusing mainly on medical image classification. One approach is that by Yan et al. (2010; 2014) who use logistic regression to predict the per-instance reliability of annotators. Wang and Bi (2017) used a modified support vector machine (SVM; Cortes and Vapnik 1995) loss, modelling the per-instance reliability as the distance from the given instance to a separation boundary.

2.2 True Label Prediction in Crowdsourcing

True label prediction in crowdsourcing is the aggregation of labels produced by different annotators to infer the correct label of each instance. Majority voting assigns to each instance the most commonly occurring label among the annotators, which can result in a high agreement between the predicted label and the ground truth for some NLP tasks (Snow et al., 2008). Dawid and Skene (1979), Whitehill et al. (2009), Raykar et al. (2010), Welinder et al. (2010), Liu et al. (2012), Zhou et al. (2012), Kim and Ghahramani (2012), Hovy et al. (2013), Yan et al. (2010; 2014), Li et al. (2014b) and Zhang et al. (2014) investigated binary or multi-class label prediction using probabilistic graphical models. Karger et al. (2011), Wang and Bi (2017), and Bonald and Combes (2017) formalised the label prediction as an optimisation problem. Rodrigues et al. (2014) and Nguyen et al. (2017) investigated how to aggregate sequence labels using probabilistic graphical models.

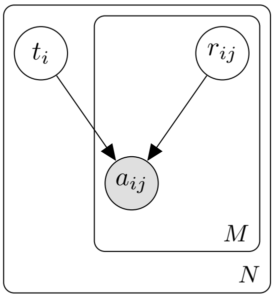


Figure 1: Graphical model.

```
for $i = 1$ to $N$
    $t_i \sim \text{Categorical}(f_t(\mathbf{x}_i))$
    for $j = 1$ to $M$
        $r_{ij} \sim \text{Bernoulli}(f_r(\mathbf{x}_i, j))$
        $a_{ij} \sim \begin{cases} \text{Uniform}(T) & r_{ij} = 0 \\ \delta_{t_i} & r_{ij} = 1 \end{cases}$
```

Figure 2: Generative process for our method. f_t is the classifier, returning a probability distribution over predicted labels, and f_r is the reliability estimator, returning the probability that the annotator is accurate for the instance. $\text{Uniform}(T)$ is a uniform distribution over the categories in T. δ_{t_i} is the deterministic distribution that only takes on the value t_i.

3 Methodology

3.1 Model

In the description of our model we let N be the number of training instances, M the number of annotators, $\mathbf{x}_i$ the ith training instance, t_i its true underlying label, T the set of values t_i can take on, r_{ij} whether annotator j is reliable for the ith instance, and a_{ij} the label that annotator j gave the ith instance. Below we describe the components of the model in more detail.

Probabilistic Model: Our model is inspired by the method proposed by Hovy et al. (2013), and it shares the same graphical representation (see Figure 1). The distributions of the model, however, are defined differently, as can be seen in Figure 2, due to the inclusion of a classifier and a reliability estimator.

We assume that the underlying label t_i depends only on the corresponding instance, while the reliability r_{ij} depends on the instance and the identity of the annotator. If $r_{ij} = 0$, then the annotator j is unreliable for instance $\mathbf{x}_i$, and a label is chosen randomly from among the available categories. Otherwise, the annotation a_{ij} is set to be the correct label.

Classifier: The classifier $f_t(\mathbf{x}_i)$ provides the predicted probabilities of an instance belonging to each category, $p(t_i \mid \mathbf{x}_i)$. t_i is the underlying label for instance $\mathbf{x}_i$, the ith instance, and takes a value in the set of categories T. Note that there is no restriction on what classifier is used, other than that it can be trained using expectation maximisation. The inclusion of a classifier directly in the model means that it can be trained while taking into account the uncertainty of the data and predictions, as opposed to first making a hard assignment of a label for each instance and training the classifier post-hoc.

Reliability Estimator: The reliability estimator $f_r(\mathbf{x}_i, j)$ predicts the probability of annotator j producing the correct label for instance $\mathbf{x}_i$, $p(r_{ij} \mid \mathbf{x}_i)$. r_{ij} is a binary variable, with 1 and 0 representing annotator j being reliable and unreliable for instance $\mathbf{x}_i$, respectively. The reliability estimator is modelled as a feed-forward neural network, where j is encoded as a one-hot vector. The exact representation of $\mathbf{x}_i$ depends on the model used for the classifier. If the classifier is a neural network, the output of the last hidden layer is used; otherwise, the original feature vector is used.

3.2 Learning

Pre-training

As the number of parameters in our model is much larger than that of previous studies (Yan et al., 2010, 2014; Wang and Bi, 2017) due to the introduction of both a classifier and a reliability estimator, the model is much harder to train from scratch. Therefore, before we start training the model, we first pre-train the classifier using labels predicted by a simpler method as targets, using e.g. majority voting or the method proposed by Dawid and Skene (1979). Although these labels may be noisy, we have observed empirically that a better initialisation strategy does result in better performance (see Section 5). For the reliability estimator, for each instance $\mathbf{x}_i$ we compare each annotation a_{ij} to the labels predicted in the previous step. If a_{ij} is the same as the predicted label, we take the corresponding r_{ij} to be 1, and 0 otherwise. We then pre-train the reliability estimator f_r to predict these values for $\mathbf{r}$.

EM Training

We first consider training our model using expectation maximisation (EM; Dempster et al. 1977).

This involves maximising the expectation of the complete log likelihood of the model with respect to the posterior of the latent variables in the model. For the posterior of the model, we fix the parameters of the model and denote them $\boldsymbol{\theta}^{(k)}$ at iteration k of the algorithm. We only maximise the expectation with respect to the parameters $\boldsymbol{\theta}$ of the complete log likelihood.

The expectation is calculated as:

$$
\begin{aligned}
Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) &= \mathbb{E}[\log p(\mathbf{a}, \mathbf{t}, \mathbf{r} \mid \mathbf{x}, \boldsymbol{\theta})] \\
&= \sum_{i=1}^{N} \mathbb{E}\left[\log p(t_i \mid \mathbf{x}_i, \boldsymbol{\theta})\right] \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{E}\left[\log p(r_{ij} \mid \mathbf{x}_i, \boldsymbol{\theta})\right] \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{E}\left[\log p(a_{ij} \mid t_i, r_{ij}, \mathbf{x}_i, \boldsymbol{\theta})\right],
\end{aligned} \tag{1}
$$

where each expectation is calculated with respect to the posterior $p(\mathbf{t}, \mathbf{r} \mid \mathbf{a}, \mathbf{x}, \boldsymbol{\theta}^{(k)})$.

E Step: For the E step we compute the posterior with fixed parameters $\boldsymbol{\theta}^{(k)}$, $\pi_{ij}(t, r) = p(t_i = t, r_{ij} = r \mid \mathbf{a}_i, \mathbf{x}_i)$, as:

$$
\begin{aligned}
\pi_{ij}(t, r) &= p(t_i = t, r_{ij} = r \mid \mathbf{a}_i, \mathbf{x}_i) \\
&\propto p(t_i = t \mid \mathbf{x}_i) p(r_{ij} = r \mid \mathbf{x}_i) \\
&\cdot p(a_{ij} \mid t_i = t, r_{ij} = r, \mathbf{x}_i) \cdot \prod_{j' \neq j} \gamma_{ij'}^{(k)}(t)
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
\gamma_{ij}^{k}(t) = \sum_{r' \in \{0,1\}} &\big(p(r_{ij} = r' \mid \mathbf{x}_i) \\
&\cdot p(a_{ij} \mid t_i = t, r_{ij} = r', \mathbf{x}_i)\big),
\end{aligned} \tag{3}
$$

where we drop the dependency on $\boldsymbol{\theta}^{(k)}$ for brevity. Note that $\pi_{ij}(t, r) = 0$ when $r = 1$ and $a_{ij} \neq t$.

We can then compute the marginalised posteriors, needed for Equation (1), as follows:

$$
p(t_i = t \mid \mathbf{a}_i, \mathbf{x}_i) = \sum_{r \in \{0,1\}} \pi_{i1}(t, r) \tag{4}
$$

$$
p(r_{ij} = r \mid \mathbf{a}_i, \mathbf{x}_i) = \sum_{t \in T} \pi_{ij}(t, r), \tag{5}
$$

where the posterior $p(t_i, r_{i1} \mid \mathbf{a}_i, \mathbf{x}_i)$ of the model is chosen arbitrarily to marginalise over to get the posterior for t_i.

M Step: Using the posterior calculated in the E step we can compute the expectation of the complete log likelihood, $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$, and calculate its gradient with respect to the parameters $\boldsymbol{\theta}$. We then use gradient ascent to update the classifier and reliability estimator jointly.

Algorithm 1 Training procedure

```
Input:
    $\mathbf{a}$, the annotations
    $\mathbf{x}$, the instances
    $L$, the number of inner iterations
    $m$, the training mode: 0 for expectation maximisation, 1 for cross entropy (training alternatingly), 2 for cross entropy (training jointly)

Pre-train $\boldsymbol{\theta} = \{\boldsymbol{\theta}_t, \boldsymbol{\theta}_r\}$ (Section 3.2)
$k \leftarrow 0$
while stopping criteria not met do
    $k \leftarrow k + 1$
    Calculate $\pi_{ij}(t, r)$ (Equation (2))                ▷ E step
    if $m = 0$ then                                          ▷ M step
        for $l = 1$ to $L$
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$
        end for
    else if $m = 1$ then
        for $l = 1$ to $L$
            $\boldsymbol{\theta}_r \leftarrow \boldsymbol{\theta}_r - \alpha \nabla_{\boldsymbol{\theta}_r} L(\boldsymbol{\theta}_r \mid \boldsymbol{\theta}^{(k)})$
        end for
        for $l = 1$ to $L$
            $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}^{(k)})$
        end for
    else
        for $l = 1$ to $L$
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$
        end for
    end if
end while
```

Cross Entropy Training

As an alternative training procedure, we also consider training the model using cross entropy. As with expectation maximisation, we first calculate the posterior $\pi_{ij}(t, r)$ using the fixed parameters $\boldsymbol{\theta}^{(k)}$. The networks f_t and f_r are then trained to minimise the cross entropy between the priors $p(t_i \mid \mathbf{x}_i)$ and $p(r_{ij} \mid \mathbf{x}_i)$, and the corresponding posteriors $p(t_i \mid \mathbf{a}_i, \mathbf{x}_i)$ and $p(r_{ij} \mid \mathbf{a}_i, \mathbf{x}_i)$.

The networks can be trained in an alternating fashion, with f_r being trained while f_t is kept fixed, and the other way around. Denoting the parameters of f_t as $\boldsymbol{\theta}_t$ and f_r as $\boldsymbol{\theta}_r$, the loss functions for the respective networks then become

$$
\begin{aligned}
L(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}^{(k)}) &= -\frac{1}{N} \sum_{i,t,r} \pi_{i1}(t, r) \log p(t_i \mid \mathbf{x}_i) \\
L(\boldsymbol{\theta}_r \mid \boldsymbol{\theta}^{(k)}) &= -\frac{1}{NM} \sum_{i,j,t,r} \pi_{ij}(t, r) \log p(r_{ij} \mid \mathbf{x}_i)
\end{aligned} \tag{6}
$$

Alternatively, they can be trained jointly by minimising the total cross entropy.

$$
L(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)}) = L(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}^{(k)}) + L(\boldsymbol{\theta}_r \mid \boldsymbol{\theta}^{(k)}) \tag{7}
$$

The training algorithm is summarised in Algorithm 1. The algorithm is run until either a maximum number of iterations is reached, or the objective function stops improving.

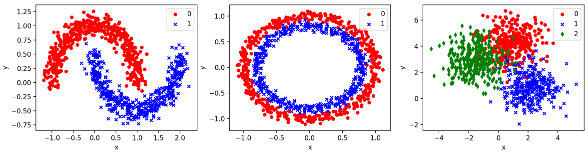


Figure 3: Three 2-dimensional datasets.

4 Evaluation Settings

4.1 Data

Simulated Annotators

2-Dimensional Datasets: In order to see whether our method can work well on simple cases, we create three 2-dimensional synthetic datasets, which we refer to as moon, circle and 3-class as shown in Figure 3.

Text Classification: For text classification we use the datasets Question Classification (Li and Roth, 2002), which contains short questions along with the type of answer expected, and Sentence Classification (Chambers, 2013), which consists of sentences selected from medical publications. Examples of instance/class pairs for the text classification datasets include ”Where is the Orinoco?” (class: ”location”) for the Question Classification dataset, and ”New types of potent force clamps are discovered.” (class: ”author’s own work”) for the Sentence Classification dataset.

For these datasets that do not include crowd annotations, we synthesise annotations by simulating different annotators as follows: 1) **Narrow Expert**: has expertise in a single domain (i.e. class). For the instances of this class, the annotator will always provide the correct label. For other classes, a correct label will be provided with a probability of 0.65; otherwise, a random label will be selected with uniform probability; 2) **Broad Expert**: has expertise in every domain and only makes mistakes with a probability of 0.05; 3) **Random Annotator**: selects labels at random; 4) **Adversarial Annotator**: deliberately provides incorrect labels with a probability of 0.8.

For each of the datasets, we generated annotations using one narrow expert per class, one broad expert, one random annotator and one adversarial annotator, for a total of $|T| + 3$ annotators, where $|T|$ is the number of classes in the dataset.

In order to evaluate the generality of our model, we also apply it to another task in which we have 5 annotators with different overall reliabilities for the text classification tasks. They produce incorrect labels with probabilities 0.1, 0.3, 0.5, 0.7, 0.9 respectively.

Dataset	Class	# Instances
moon	0/1	500/500
circle	0/1	500/500
3-class	0/1/2	334/333/333
Question Classification	DESCRIPTION (DESC)	1162
	ENTITY (ENTY)	1250
	ABBREV. (ABBR)	86
	HUMAN (HUM)	1223
	NUMERIC (NUM)	896
	LOCATION (LOC)	835
Sentence Classification	AIMX	94
	OWNX	427
	CONT	104
	BASE	33
	MISC	852
RTE	0/1	400/400

Table 1: Classes and per-class instance counts.

Datasets	Classifier	# Units
2-Dimensional Datasets (3 datasets)	2D→FNN	5
Text Classification (2 datasets)	BoW→FNN	100
	Avg. →FNN	50
	Embed.→LSTM→FNN	100
RTE	Cat. Avg. →FNN	100
	Embed.→LSTM→FNN (Bowman et al., 2015)	200

Table 2: Classifiers used in the experiments.

Real-World Crowdsourcing Annotators

Recognising Textual Entailment: Finally, we evaluate our model on a real-world dataset for the recognising textual entailment (RTE) task (Snow et al., 2008). Given a text pair, the annotator decides whether the hypothesis sentence can be inferred from the text fragment. The dataset includes both ground truth and crowdsourced labels from 164 annotators.

Table 1 shows the number of instances of each class[2] in the above-mentioned datasets.

4.2 Experimental and Model Settings

Our model was implemented using the Chainer deep learning framework[3] (Tokui et al., 2015).

Classifier: As shown in Table 2, in each experiment the output of the classifier is generated by a feed-forward neural network (FNN). Each FNN consists of an input layer, two hidden layers and a softmax output layer. The number of hidden units in each layer is listed in the third column of the table. The ReLU activation function (Nair and

---

[2] The classes in the Sentence Classification dataset are defined as follows: AIMX—the goal of the paper; OWNX—the author’s own work; CONT—the comparison including contrast and critique of past work; BASE—the past research that provides the basis for the work; MISC—any other sentences.

[3] `https://chainer.org/`

Hinton, 2010) was applied after each hidden layer. The output size of all the Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) layers in our experiments is 100.

For the 2-dimensional classification task, each instance is simply represented using its position in 2-dimensional space. For the text classification tasks, we investigated 3 methods of representing the sentences: bag-of-words (BoW) weighted by Term Frequency–Inverse Document Frequency (TFIDF), an average word embedding (Avg.) and the output at the last step of an LSTM layer (Embed.→LSTM). For the embedding we use word2vec embeddings pre-trained on Google News (Mikolov et al., 2013) for the question classification and RTE tasks, and a pre-trained embedding (Pyysalo et al., 2013) trained on a combination of English Wikipedia, PubMed and PMC texts for the sentence classification task.

For the RTE task, we implemented two classifiers. For the first one, each instance (i.e. a sentence pair) was represented as a concatenation of the average word embedding for each sentence (Cat. Avg.). We also implemented Bowman et al. (2015), which runs each sentence through an LSTM, concatenates the outputs, and then feeds the concatenated output to an FNN with tanh activations.

Reliability Estimator: We model the reliability estimator as an FNN. Its structure is the same as the classifier, albeit with different sizes of the two hidden layers. For the experiments listed in Table 2, the number of units of each hidden layer in the FNN are 5, 100, 25, 25, 50, and 100 respectively. The input to the estimator is the concatenation of the instance $\mathbf{x}_i$ (i.e. its original feature vector or the output of the last hidden layer of the classifier) and a one-hot vector representing the annotator identity.

Learning Settings: For every experiment we use the Adam (Kingma and Ba, 2015) optimiser with a weight decay rate 0.001, a gradient clipping of 5.0, $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We pre-train the classifier and reliability estimator for 200 epochs, using both majority voting and the model proposed by Dawid and Skene (1979). The maximum number of outer iterations is set to 500 and 20 for EM training and cross entropy training respectively. The number of inner iterations is 50 in both cases.

True Label Prediction and Reliability Estimation: After training, for each instance $\mathbf{x}_i$ we take its underlying label to be the most probable label according to the posterior of t_i (see Equation (4)). We compared our predicted labels to the following state-of-the-art baselines: Majority Voting (MV), DS (Dawid and Skene, 1979), GLAD (Whitehill et al., 2009), LFC (Raykar et al., 2010), CUBAM (Welinder et al., 2010), Yan et al. (2010), KOS (Karger et al., 2011), VI (Liu et al., 2012), BCC (Kim and Ghahramani, 2012), MINIMAX (Zhou et al., 2012), MACE (Hovy et al., 2013), CATD (Li et al., 2014a), PM (Li et al., 2014b) and EM-MV and Opt (Zhang et al., 2014).

Note that CUBAM, Yan et al. (2010), KOS and VI are only suitable for aggregating binary labels, and Yan et al. (2010) is the state-of-the-art method that models per-instance reliability. We take the reliability of annotator j on instance $\mathbf{x}_i$ to be the posterior probability that r_{ij} is 1 (see Equation (5)).

5 Results and Analysis

We measure the inter-annotator agreement (IAA) of each dataset. Fleiss's kappa (Fleiss et al., 2013), denoted by κ, is measured for the 2-dimensional and text classification datasets, and Krippendorff's alpha (Krippendorff, 1970) is calculated for the RTE dataset[4]. We find that the IAA values indicate slight agreement among annotators for all datasets.

Our experiments using different settings are shown as follows: our model is denoted by *O*, with *M* and *D* denoting the model pre-trained using MV and DS respectively. *E* denotes training using expectation maximisation, while *C* denotes cross entropy training. *AL* and *JT* denote cross entropy training done alternatingly and jointly, respectively.

In the rest of this section, Tables 3 to 7 and Tables 8 to 10 present the results on the synthetic datasets and RTE dataset respectively. For the synthetic datasets, in Tables 3 to 6, we first consider a scenario where we have multiple narrow experts (N), one broad expert (B), one random annotator (R) and one adversarial annotator (A). In Table 7, we further consider a scenario with 5 annotators,

---

[4] Although there are 164 annotators in total in this dataset, each instance was labelled by only 10 of these annotators. Therefore we use Krippendorff's alpha which is applicable to incomplete data to measure the inter-annotator agreement.

	2-Dimensional Datasets		
	moon	**circle**	**3-class**
	$\kappa = 0.029$	$\kappa = 0.029$	$\kappa = 0.153$
MV	84.6	84.6	89.0
DS (Dawid and Skene, 1979)	97.8	97.9	99.4
GLAD (Whitehill et al., 2009)	97.9	97.9	93.7
LFC (Raykar et al., 2010)	97.9	97.9	99.4
CUBAM (Welinder et al., 2010)	97.9	97.9	-
Yan et al. (Yan et al., 2010)	84.1	84.1	-
KOS (Karger et al., 2011)	96.7	96.7	-
VI (Liu et al., 2012)	97.9	97.9	-
BCC (Kim and Ghahramani, 2012)	97.9	97.9	99.4
MINIMAX (Zhou et al., 2012)	97.9	97.9	99.2
MACE (Hovy et al., 2013)	97.8	97.9	97.0
CATD (Li et al., 2014a)	94.5	94.5	94.3
PM (Li et al., 2014b)	94.5	94.5	94.3
EM-MV (Zhang et al., 2014)	97.9	97.9	96.3
EM-Opt (Zhang et al., 2014)	72.7	72.7	93.2
O-ME	97.0	97.7	94.8
O-MC	97.3	98.9	99.3
O-DE	91.2	97.2	96.5
O-DC-AL	99.7	**99.7**	**99.5**
O-DC-JT	**99.8**	99.6	**99.5**

Table 3: F1 scores of predicted labels on the 2-dimensional datasets when using the output of the last hidden layer of the classifier to represent an instance for the reliability estimator.

each of differing reliability, as explained in Section 4.1.

Table 3 shows that our method performs well on the 2-dimensional datasets, obtaining higher label prediction F1 scores than the baselines. We omit the analysis of the true label prediction and reliability estimation results on these datasets as all models performed similarly, choosing instead to focus the discussion on the results for the NLP tasks.

5.1 Classifier and Reliability Estimator

In order to explore the separate performance contribution of classifier and reliability estimator, we compare the performance of our model to a classifier pre-trained using DS labels, as well as a variant of our model without the reliability estimator, i.e. setting all the annotators have the same reliability on all the instances.

As shown in Tables 4, 7 and 8, the pre-trained classifier performed worse than some aggregation methods. This indicates that the noise in the labels predicted by DS has an adverse effect on the training of the classifier. The much lower performance of the model with the reliability estimator removes hints at the importance of modelling per-annotator reliability to ensure accurate predictions.

	Question Classification		Sentence Classification	
	$\kappa = 0.094$		$\kappa = 0.0634$	
	BoW	**Avg.**	**BoW**	**Avg.**
MV	78.8	78.8	71.3	71.3
DS (Dawid and Skene, 1979)	98.3	98.3	97.3	97.3
GLAD (Whitehill et al., 2009)	87.1	87.1	79.9	79.9
LFC (Raykar et al., 2010)	98.2	98.2	97.0	97.0
BCC (Kim and Ghahramani, 2012)	98.3	98.3	98.1	98.1
MINIMAX (Zhou et al., 2012)	28.2	28.2	50.9	50.9
MACE (Hovy et al., 2013)	91.6	91.6	63.6	63.6
CATD (Li et al., 2014a)	91.1	91.1	92.2	92.2
PM (Li et al., 2014b)	91.1	91.1	92.2	92.2
EM-MV (Zhang et al., 2014)	88.3	88.3	66.9	66.9
EM-Opt (Zhang et al., 2014)	13.5	13.5	26.5	26.5
Classifier (pre-trained by DS labels)	92.3	86.5	89.6	85.4
O-DE (without reliability estimator)	92.1	91.2	77.6	87.1
O-DC-AL (without reliability estimator)	90.5	93.2	84.0	88.2
O-DC-JT (without reliability estimator)	88.7	91.1	78.5	87.4
O-DE (using feature vector)	95.1	97.0	88.9	97.8
O-DC-AL (using feature vector)	97.5	98.9	97.2	97.5
O-DC-JT (using feature vector)	97.5	98.9	97.2	97.5
O-ME (full model)	92.4	95.0	84.4	89.2
O-MC (full model)	97.2	98.3	90.6	94.6
O-DE (full model)	98.0	98.3	**97.7**	**98.9**
O-DC-AL (full model)	98.3	98.6	97.6	97.9
O-DC-JT (full model)	**98.7**	**99.0**	97.3	97.8

Table 4: F1 scores of predicted labels on the text classification datasets.

5.2 Instance Representation for Reliability Estimator

For the representation of the instance $\mathbf{x}_i$ as it is fed to the reliability estimator, we compared the performance of using the original feature vector of $\mathbf{x}_i$ to using the last hidden layer output of the classifier (which we refer to as the "full model").

We found that using the hidden layer representation can not only improve the label prediction performance (see Tables 4, 7 and 8), but also sped up the training compared to using the feature vector directly. The hidden layer representation allows us to reduce the number of parameters in the model, by sharing parameters with the classifier.

5.3 Full Model on Synthetic Datasets

Based on the results of the full model in Table 4, we can conclude that per-instance reliability modelling is beneficial to the label prediction task, and using the average pre-trained embedding can result in slightly better performance. It is worth noting that the method used to pre-train the model had a noticeable effect on its performance, with better F1 scores being obtained when using DS pre-training. In the following experiments we only consider models pre-trained using the DS algorithm.

In order to investigate whether our method can successfully capture per-instance annotator reliability, for each annotator, we counted the number of correctly labelled instances and calculated the

	Question Classification						
	DESC	**ENTY**	**ABBR**	**HUM**	**NUM**	**LOC**	**Accuracy**
1 (N)	99	1	0	0	0	0	100
2 (N)	0	100	0	0	0	0	100
3 (N)	31	13	40	5	2	9	100
4 (N)	0	0	0	100	0	0	100
5 (N)	0	0	0	0	100	0	100
6 (N)	0	0	0	0	0	100	100
7 (B)	20	0	0	77	0	3	100
8 (R)	30	32	8	8	15	7	100
9 (A)	45	19	9	14	5	8	100

Table 5: Number of correctly labelled examples for each annotator (N: narrow expert, B: broad expert, R: random annotator and A: adversarial annotator) among the 100 instances with highest per-instance reliability on the question classification dataset.

	Question Classification						
	DESC	**ENTY**	**ABBR**	**HUM**	**NUM**	**LOC**	**Overall**
1 (N)	90.6	24.4	35.5	21.4	26.6	22.2	36.8
2 (N)	28.7	92.3	26.6	25.2	24.3	26.7	37.3
3 (N)	24.7	24.9	75.8	23.5	24.7	23.6	32.9
4 (N)	27.5	26.3	32.3	93.4	24.2	23.7	37.9
5 (N)	25.6	21.9	25.2	25.7	93.8	26.0	36.4
6 (N)	26.4	25.0	24.1	22.1	27.5	94.4	36.6
7 (B)	94.3	93.5	95.1	94.2	93.4	93.9	94.2
8 (R)	5.60	6.30	8.40	4.80	6.20	5.90	6.20
9 (A)	9.10	8.90	12.0	8.20	7.90	8.30	9.10

Table 6: Average reliability of each annotator among the 100 instances with the highest per-instance reliability on the question classification dataset.

	Question Classification $\kappa = 0.126$		**Sentence Classification** $\kappa = 0.0776$	
	FNN	**LSTM+FNN**	**FNN**	**LSTM+FNN**
MV	71.8	71.8	65.8	65.8
DS (Dawid and Skene, 1979)	90.1	90.1	83.9	83.9
GLAD (Whitehill et al., 2009)	80.9	80.9	71.8	71.8
LFC (Raykar et al., 2010)	88.3	88.3	80.3	80.3
BCC (Kim and Ghahramani, 2012)	90.4	90.4	85.6	85.6
MINIMAX (Zhou et al., 2012)	30.4	30.4	44.0	44.0
MACE (Hovy et al., 2013)	84.6	84.6	62.4	62.4
CATD (Li et al., 2014a)	85.2	85.2	80.5	80.5
PM (Li et al., 2014b)	85.2	85.2	80.5	80.5
EM-MV (Zhang et al., 2014)	75.4	75.4	56.6	56.6
EM-Opt (Zhang et al., 2014)	19.4	19.4	14.4	14.4
Classifier (pre-trained by DS labels)	89.7	76.6	79.6	78.7
O-DE (without reliability estimator)	91.9	91.0	80.1	82.5
O-DC-AL (without reliability estimator)	91.8	91.2	80.0	83.5
O-DC-JT (without reliability estimator)	91.5	91.6	79.3	83.5
O-DE (using feature vector)	94.3	-	81.7	-
O-DC-AL (using feature vector)	94.1	-	86.2	-
O-DC-JT (using feature vector)	94.1	-	86.2	-
O-DE (full model)	94.5	94.1	86.1	86.1
O-DC-AL (full model)	94.7	**96.0**	**90.3**	88.5
O-DC-JT (full model)	**95.1**	95.1	88.5	**89.0**

Table 7: F1 scores on text classification tasks when only the reliability differs between the annotators.

Method	**F-measure**
MV	91.9
DS (Dawid and Skene, 1979)	92.6
GLAD (Whitehill et al., 2009)	92.4
LFC (Raykar et al., 2010)	92.5
CUBAM (Welinder et al., 2010)	92.6
Yan et al. (Yan et al., 2010)	90.4
KOS (Karger et al., 2011)	63.2
VI (Liu et al., 2012)	92.5
BCC (Kim and Ghahramani, 2012)	92.3
MINIMAX (Zhou et al., 2012)	92.4
MACE (Hovy et al., 2013)	92.4
CATD (Li et al., 2014a)	92.3
PM (Li et al., 2014b)	92.0
EM-MV (Zhang et al., 2014)	92.5
EM-Opt (Zhang et al., 2014)	92.4
Classifier (pre-trained by DS labels)	89.9
O-DC-JT (FNN) (without reliability estimator)	90.2
O-DC-JT (FNN) (using feature vector)	92.6
O-DE (FNN) (full model)	92.9
O-DC-AL (FNN) (full model)	92.8
O-DC-JT (FNN) (full model)	**93.0**
O-DE (LSTM+FNN) (full model)	92.7
O-DC-AL (LSTM+FNN) (full model)	92.8
O-DC-JT (LSTM+FNN) (full model)	92.8

Table 8: Performance of predicted labels on the RTE dataset (Krippendorff's alpha = 0.0995).

average reliability for each class among the top 100 instances with the highest per-instance reliability as shown in Table 5 and 6[5]. The cells with grey background colour indicate which domain, or class, the annotator has expertise in. It can be seen that all annotators obtain high accuracy on these instances. In general our method also captured the varying expertise of each narrow annotator, estimating their reliability on instances belonging to the corresponding classes as particularly high.

For these experiments in Table 7, we also investigated the performance when using two different classification models. As seen in this table, both of them outperformed all baselines significantly.

5.4 Full Model on RTE Dataset

Table 8 presents the label prediction performance on the RTE dataset. As not every annotator has provided labels for every instance in this dataset, for both the EM and cross entropy training we simply omitted missing instance/annotator pairs when calculating the loss functions. As seen in the table, most of the baselines obtained high performance as the textual entailment recognition task is easy for non-expert annotators. However, our full model still achieved better prediction performance than all of the baseline methods.

We also investigated the effectiveness of our model for removing noisy labels. We compare our model to the five best-performing baselines (DS, LFC, CUBAM, VI and EM-MV in Table 8). Each of these models are trained on the RTE dataset, after which the least reliable annotation for each instance is removed. We use the per-instance reliability for our model, the global reliability score of each annotator for LFC, CUBAM and VI, and the per-category annotator reliability for DS and EM-MV as the measure of the reliability of each annotation. For each of these models, we then retrain the models in Table 8 using the denoised dataset; the difference in performance can be seen in Table 9. We can see that using per-instance reliability results in the largest improvement, while only considering the annotators' overall reliability may

---

[5] We omit the results for the sentence classification task for lack of space, as we consider the results on the question classification dataset to be representative.

Method	LFC	CUBAM	VI	DS	EM-MV	Ours
MV	-0.2	-0.9	+0.6	-0.2	-0.2	**+0.8**
DS (Dawid and Skene, 1979)	0	-0.2	+0.1	0	0	**+0.6**
GLAD (Whitehill et al., 2009)	**+0.2**	0	0	**+0.2**	-0.3	**+0.2**
LFC (Raykar et al., 2010)	+0.1	-0.1	+0.3	+0.1	+0.1	**+0.6**
CUBAM (Welinder et al., 2010)	+0.3	0	+0.4	+0.3	-0.2	**+0.9**
Yan et al. (Yan et al., 2010)	+1.5	+0.4	+2.3	+1.5	+0.8	**+2.5**
KOS (Karger et al., 2011)	+0.7	+3.9	+11.3	+0.7	+13.4	**+17.3**
VI (Liu et al., 2012)	+0.1	+0.1	+0.1	+0.1	+0.2	**+0.6**
BCC (Kim and Ghahramani, 2012)	+0.3	+0.2	+0.3	+0.3	+0.4	**+0.8**
MINIMAX (Zhou et al., 2012)	+0.3	-0.4	**+0.7**	+0.3	+0.2	**+0.7**
MACE (Hovy et al., 2013)	+0.2	-0.2	0	+0.2	0	**+0.4**
CATD (Li et al., 2014a)	+0.3	-0.8	-0.2	+0.3	+0.2	**+0.3**
PM (Li et al., 2014b)	**+0.9**	0	+0.3	**+0.9**	+0.1	**+0.9**
EM-MV (Zhang et al., 2014)	+0.7	+0.6	+0.7	+0.7	+0.7	**+1.1**
EM-Opt (Zhang et al., 2014)	+0.2	+0.5	+0.2	+0.2	+0.6	**+0.7**
O-DC-JT (FNN) (full model)	+0.1	0	+0.3	+0.1	+0.1	**+0.5**

Table 9: F1 score improvements after removing the label produced by the least reliable annotator by using the estimated overall reliability (LFC, CUBAM, VI, DS, EM-MV) and per-instance reliability (Ours).

		True Entailment		**False Entailment**		
	Annotator	**#Cor. Labels**	**Avg. Reliability**	**#Cor. Labels**	**Avg. Reliability**	**Acc.**
Top 15 Instances	1	15	95.5	-	-	100
	2	15	92.6	-	-	100
	3	10	88.4	3	86.9	86.7
	4	11	94.2	2	92.6	86.7
	5	8	71.9	2	23.7	66.7
Bottom 15 Instances	1	4	0.1	2	0.1	40
	2	1	1.0	8	51.4	60
	3	0	9.7	1	13.8	6.6
	4	2	28.6	5	63.1	46.7
	5	3	22.3	2	23.7	33.3

Table 10: Number of correct labels and average reliability for each annotator among the instances with highest and lowest per-instance reliability on the RTE dataset.

cause a reduction in performance.

In order to analyse the per-instance reliability of the human annotators, for each annotator we rank the instances according to the annotator's per-instance reliability. We look at the top 15 and bottom 15 instances, then count how many of them were correctly labelled (Cor. Labels) as well as the average reliability on these instances (Avg. Reliability). Table 10 shows the results of five annotators[6]. It can be seen that each annotator has considerably different reliabilities across instances.

5.5 Training Stability

Pre-training: As discussed in Section 3.2, the predicted labels produced by a simpler method are used for pre-training. Although these labels are not perfect, we assume that our method can still learn some useful information from them for a better starting point than random parameter initialisation.

EM and Cross Entropy Training: From Tables 3, 4, 7 and 8, it can be seen that, in most cases, using cross entropy achieved much better and more stable performance than the models learned using EM training. We also noticed that the objective function would improve when using cross entropy training, and tended to converge faster in our experiments—generally within just a few epochs. Therefore, we recommend to use this training method in practice.

Early Stopping: When using both EM and cross entropy training, we found that even if the objective function improved between iterations, the label prediction performance would eventually start to decrease. It is worth to investigate the reason for this phenomenon. To counteract this issue we used early stopping, where training is halted when the objective function does not improve more than 0.001 between iterations. Another option is to reduce the maximum number of outer iterations, e.g. to 20.

6 Conclusion and Future Work

We propose a novel probabilistic model which learns from noisy labels produced by multiple annotators for NLP crowdsourcing tasks by incorporating a classifier and a reliability estimator. Our work constitutes the first effort to model the per-instance reliability of annotators for both binary and multi-class NLP labelling tasks. We investigate two methods of training our model using the EM algorithm and cross entropy. Experimental results on 6 datasets including synthetic and real datasets demonstrate that our method can not only capture the per-instance reliability of each annotator, but also obtain better label prediction and the least reliable annotator detection performance compared to state-of-the-art baselines.

For future work, we plan to apply our model to other NLP tasks such as relation extraction and named entity recognition. We also plan to investigate the use of variational inference (Jordan et al., 1999) as a means of training our model. Using variational inference might improve the stability and performance of our model.

Acknowledgement

We would like to thank the anonymous reviewers and Paul Thompson for their valuable comments. Discussions with Austin J. Brockmeier have been insightful. The work is funded by *School of Computer Science Kilburn Overseas Fees Bursary* from University of Manchester.

---

[6] For lack of space, we only present the results for 5 of the 164 annotators.

References

Lora Aroyo and Chris Welty. 2013. Measuring crowd truth for medical relation extraction. In *AAAI 2013 Fall Symposium on Semantics for Big Data*, pages 1–9, Arlington, Virginia.

Thomas Bonald and Richard Combes. 2017. A minimax optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4352–4360. Curran Associates, Inc.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

America Chambers. 2013. *Statistical Models for Text Classification and Clustering: Applications and Analysis*. Ph.D. thesis, University of California, Irvine.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web (WWW)*, WWW '12, pages 469–478, New York, NY, USA. Association for Computing Machinery.

Arthur P. Dempster, Nan Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39:1–38.

Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, CIKM '08, pages 619–628, New York, NY, USA. Association for Computing Machinery.

Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. 2013. *Statistical methods for rates and proportions*, 3 edition. John Wiley Sons.

Iryna Gurevych and Jungi Kim, editors. 2013. *The People's Web Meets NLP: Collaboratively Constructed Language Resources*. Springer-Verlag Berlin Heidelberg.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

David R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 1953–1961. Curran Associates, Inc.

Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 619–627, La Palma, Canary Islands. PMLR.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–13, San Diego, CA, USA.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Maolin Li, Nhung Nguyen, and Sophia Ananiadou. 2017. Proactive learning for named entity recognition. In *BioNLP 2017*, pages 117–125, Vancouver, Canada. Association for Computational Linguistics.

Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014a. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436.

Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014b. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1187–1198, New York, NY, USA. Association for Computing Machinery.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, volume 1, pages 1–7, Taipei, Taiwan. Association for Computational Linguistics.

Qiang Liu, Jian Peng, and Alexander T Ihler. 2012. Variational inference for crowdsourcing. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 692–700. Curran Associates, Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems 26 (NIPS)*, pages 3111–3119. Curran Associates, Inc.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

An Thanh Nguyen, Byron Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, pages 299–309, Vancouver, Canada. Association for Computational Linguistics.

Massimo Poesio, Jon Chamberlain, and Udo Kruschwitz. 2017. Crowdsourcing. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 277–295. Springer Netherlands, Dordrecht.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(1):3:1–3:44.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of the Fifth International Symposium on Languages in Biology and Medicine (LBM)*, pages 39–44, Tokyo, Japan.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *The Journal of Machine Learning Research*, 11(Apr):1297–1322.

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2014. Sequence labeling with multiple annotators. *Machine Learning*, 95(2):165–181.

Aashish Sheshadri and Matthew Lease. 2013. SQUARE: a benchmark for research on computing crowd consensus. In *Proceedings of the 1st AAAI Conference on Human Computation (HCOMP)*, pages 156–164, Palm Springs, CA, USA. The AAAI Press.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1–6.

Xin Wang and Jinbo Bi. 2017. Bi-convex optimization to learn classifiers from multiple biomedical annotations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(3):564–575.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23 (NIPS)*, pages 2424–2432. Curran Associates, Inc.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 2035–2043. Curran Associates, Inc.

Yan Yan, Romer Rosales, Glenn Fung, Mark Schmidt, Gerardo Hermosillo, Luca Bogoni, Linda Moy, and Jennifer Dy. 2010. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 932–939, Chia Laguna Resort, Sardinia, Italy. PMLR.

Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. 2014. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327.

Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. 2014. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 1260–1268. Curran Associates, Inc.

Denny Zhou, Sumit Basu, Yi Mao, and John C. Platt. 2012. Learning from the wisdom of crowds by minimax entropy. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 2195–2203. Curran Associates, Inc.

Review-Driven Multi-Label Music Style Classification by Exploiting Style Correlations

Guangxiang Zhao[1]*, **Jingjing Xu**[2]*, **Qi Zeng**[2], **Xuancheng Ren**[2], **Xu Sun**[1,2]
[1]Center for Data Science, Beijing Institute of Big Data Research, Peking University
[2]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
`{zhaoguangxiang,jingjingxu,pkuzengqi,renxc,xusun}@pku.edu.cn`

Abstract

This paper explores a new natural language processing task, review-driven multi-label music style classification. This task requires systems to identify multiple styles of music based on its reviews on websites. The biggest challenge lies in the complicated relations of music styles. To tackle this problem, we propose a novel deep learning approach to automatically learn and exploit style correlations. Experiment results show that our approach achieves large improvements over baselines on the proposed dataset. Furthermore, the visualized analysis shows that our approach performs well in capturing style correlations.[1]

1 Introduction

As music style (e.g., Jazz, Pop, and Rock) is one of the most frequently used labels for music, music style classification is an important task for applications of music recommendation, music information retrieval, etc. Several methods have been proposed for automatic music style classification (Qin and Ma, 2005; Zhou et al., 2006; Wang et al., 2009; Choi et al., 2017). Most of them mainly focus on using audio information to identify styles. Motivated by the fact that a pieces of music could has different styles, several studies (Wang et al., 2009; Oramas et al., 2017) also aim at multi-label music style classification.

Although these methods make promising progress, they are limited in two aspects. First, not all audio data is available in real-world applications because of copyright restrictions, which limits the generalization ability. Second, some of them are based on a strong assumption that a piece of music should be assigned with only one style. Different from these studies, we focus on using easily obtained reviews in conjunction with multi-label music style classification. The motivation comes from the fact that lots of user reviews contain rich style-related information, which can be used for music style classification.

The major challenge of this task lies in the complicated correlations of music styles. For example, Soul Music[2] contains elements of R&B and Jazz. These three labels can be used alone or in combination. Traditional multi-label classification methods may mistake the true label [Soul Music, R&B, Jazz] for the false label [R&B, Jazz]. If well learned, style relations are useful knowledge for improving the performance, e.g., increasing the probability of Soul Music if we find that it is heavily linked with two high probability labels: R&B and Jazz. Therefore, to better exploit style correlations, we propose a novel deep learning approach with two parts: a label-graph based neural network, and a soft training mechanism with correlation based continuous label representation.

Our contributions are listed as follows:

- To the best of our knowledge, this work is the first to explore review-driven multi-label music style classification.
- To learn the relations among music styles, we propose a label-graph based neural network and a soft training mechanism with correlation-based label representation.

2 Related work

This paper is related with music style classification and multi-label classification. In this section, we give a detailed introduction about the related studies.

---

[*] Equal Contribution

[1] The code and the dataset are available at `https://github.com/lancopku/RMSC`

[2] Soul Music is a popular music genre that originated in the United States in the late 1950s and early 1960s. It contains elements of African-American Gospel Music, R&B and Jazz.

Proceedings of NAACL-HLT 2019, pages 2884–2891
Minneapolis, Minnesota, June 2 - June 7, 2019.

Music Title	Mozart: The Great Piano Concertos, Vol.1
Styles	Classical Music, Piano Music
Reviews	*(1) I've been listening to* ***classical*** *music all the time.* *(2) Mozart is always good. There is a reason he is ranked in the top 3 of lists of greatest* ***classical*** *composers.* *(3) The sound of* ***piano*** *brings me peace and relaxation.* *(4) This volume of Mozart concertos is superb.*

Table 1: An illustration of review-driven multi-label music style classification. For easy interpretation, we select a simple and clear example where styles can be easily inferred from reviews. In practice, the correlation between styles and associated reviews is relatively complicated.

2.1 Music Style Classification

Previous work mainly focuses on using audio information to identify music styles. Traditional machine learning algorithms are adopted in these studies, such as Support Vector Machine (SVM) (Xu et al., 2003), Hidden Markov Model (HMM) (Chai and Vercoe, 2001; Pikrakis et al., 2006), and Decision Tree (DT) (Zhou et al., 2006). In addition to audio information, Fell and Sporleder (2014) also propose to classify music by statistical analysis of lyrics. Motivated by the fact that a piece of music could has different styles, several studies (Wang et al., 2009; Oramas et al., 2017) also aim at multi-label music style classification. Different from these studies, we focus on using easily obtained reviews in conjunction with multi-label music style classification.

2.2 Multi-Label Classification

Multi-label classification has been widely applied to diverse problems, including image classification (Qi et al., 2007; Wang et al., 2008), audio classification (Boutell et al., 2004; Sanden and Zhang, 2011), web mining (Kazawa et al., 2004), information retrieval (Zhu et al., 2005; Gopal and Yang, 2010), etc. Compared with the existing multi-label learning methods (Wei et al., 2018; Li et al., 2018b,a; Yang et al., 2018; Lin et al., 2018), our method has the following novelties: a label graph that explicitly models the relations of styles; a soft training mechanism that introduces correlation-based continuous label representation.

3 Review-Driven Multi-Label Music Style Classification

3.1 Task Definition

Given several reviews from a piece of music, this task requires models to predict a set of music styles. Assume that $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_i, \dots, \boldsymbol{x}_K\}$ denotes the input K reviews, and $\boldsymbol{x}_i = x_{i,1}, \dots, x_{i,J}$ represents the i^{th} review with J words. The term $Y = \{y_1, y_2, \dots, y_M\}$ denotes the gold set with M labels, and M varies in different samples. The target of review-driven multi-label music style classification is to learn the mapping from input reviews to style labels.

3.2 Dataset

The dataset is collected from a popular Chinese music review website,[3] where registered users are allowed to comment on all released music albums. Each sample includes a music title, a set of human annotated styles, and associated reviews. An example is shown in Table 1.

In order to build a high-quality dataset, we refer to the literature about music styles. We merge similar music styles and delete music styles that violate the music classification list. 22 styles are defined in our dataset.[4] For user reviews, we first delete reviews with too little information by rule-based methods and then select top 40 voted reviews. Music samples with too few reviews are also deleted. The constructed datataset contains over 7.1k samples, 288K reviews, and 3.6M words.

4 Proposed Approach

The proposed approach contains two parts: a label-graph based neural network and a soft training mechanism with continuous label representation. An illustration of the proposed method is shown in Figure 1.

4.1 Label-Graph Based Neural Network

The first layer is a hierarchical attention layer (Yang et al., 2016) that lets the model to pay more or less attention to individual words

---

[3] `https://music.douban.com`

[4] Alternative Music, Britpop, Classical Music, Country Music, Dark Wave, Electronic Music, Folk Music, Heavy Metal Music, Hip-Hop, Independent Music, Jazz, J-Pop, New-Age Music, OST, Piano Music, Pop, Post-Punk, Post-Rock, Punk, R&B, Rock, and Soul Music.

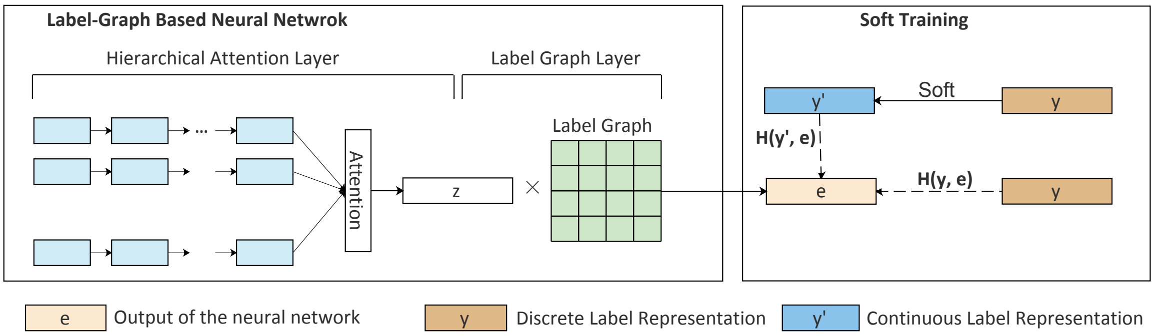


Figure 1: An illustration of the proposed approach. Left: The label-graph based neural network. Right: The soft training method. The label graph defines the relations of labels. $\boldsymbol{e}$ is the label probability distribution. Soft training means that we combine the continuous label representation $\boldsymbol{y}'$ and the discrete label representation $\boldsymbol{y}$ together to train the model. The hierarchical attention layer is responsible for extracting style-related information. The label graph layer and soft training are used for exploiting label correlations.

and reviews when constructing "raw" label probability distribution $\boldsymbol{z}$.

Label Graph. To explicitly take advantage of the label correlations when classifying music styles, we add a label graph layer to the network. This layer takes $\boldsymbol{z}$ as input and generates a "soft" label probability distribution $\boldsymbol{e}$.

Formally, we denote $\mathcal{G} \in R_{m \times m}$ as the label graph, where m is the number of labels, $\mathcal{G}$ is initialized by an identity matrix. An element $\mathcal{G}[l_i, l_j]$ is a real-value score indicating how likely label l_i and label l_j are related. The graph $\mathcal{G}$ is a part of parameters and can be learned by back-propagation.

Then, given the "raw" label probability distribution $\boldsymbol{z}$ and the label graph $\mathcal{G}$, the output of this layer is:

$$\boldsymbol{e} = \mathrm{sigmoid}(\boldsymbol{z} \cdot \mathcal{G}). \tag{1}$$

The probability of l_j is not only determined by the current classification result, but also determined by other labels probabilities and their correlations to l_j. For example, the probability of a label heavily linked with many high-probability labels will be increased.

4.2 Soft Training

Given a predicted label probability distribution $\boldsymbol{e}$ and a target discrete label representation $\boldsymbol{y}$, the typical loss function is computed as

$$L(\theta) = \mathcal{H}(\boldsymbol{y}, \boldsymbol{e}) = -\sum_{i=1}^{m} y_i \log e_i, \tag{2}$$

where θ denotes all parameters, and m is the number of the labels. The function $\mathcal{H}$ denotes the cross entropy between two distributions.

However, the widely used discrete label representation does not apply to the task of music style classification, because the music styles are not mutually exclusive and highly related to each other. The discrete distribution without label relations makes the model over-distinguish the related labels. Therefore, it is hard for the model to learn the label correlations.

Instead, we propose a soft training method by combining a discrete label representation $\boldsymbol{y}$ with a correlated-based continuous label representation $\boldsymbol{y}'$. The probability gap between two similar labels in $\boldsymbol{y}'$ should not be large.

A straight-forward approach to produce the continuous label representation is to use the label graph matrix $\mathcal{G}$ to transform the discrete representation $\boldsymbol{y}$ into a continuous form:

$$\boldsymbol{y}' = \boldsymbol{y} \cdot \mathcal{G}. \tag{3}$$

Based on the discrete label representation $\boldsymbol{y}$ and continuous label representation $\boldsymbol{y}'$, we define the loss function as

$$Loss(\theta) = \mathcal{H}(\boldsymbol{e}, \boldsymbol{y}) + \mathcal{H}(\boldsymbol{e}, \boldsymbol{y}'), \tag{4}$$

where the loss $\mathcal{H}(\boldsymbol{e}, \boldsymbol{y})$ aims to correctly classify labels, and the loss $\mathcal{H}(\boldsymbol{e}, \boldsymbol{y}')$ aims to avoid the over-distinguishing problem and to better learn label correlations.

5 Experiment

In this section, we evaluate our approach on the proposed dataset. We first introduce evaluation metrics, then show experiment results and give a detailed analysis. The training details can be found at Appendices.

Models	**OE(-)**	**HL(-)**	**Macro F1(+)**	**Micro F1(+)**
ML-KNN	77.3	0.094	23.6	38.1
Binary Relevance	74.4	0.083	24.7	41.8
Classifier Chains	67.5	0.107	29.9	44.3
Label Powerset	56.2	0.096	37.7	50.3
MLP	71.5	0.081	29.8	45.8
CNN	37.9	0.099	32.5	49.3
LSTM	30.5	0.089	33.0	53.9
HAN	25.9	0.079	52.1	61.0
+Label Graph	23.4	0.077	54.2	62.8
+ Soft Training	**22.6**	**0.074**	**54.4**	**64.5**

Table 2: Comparisons between our approach and the baselines on the test set. OE and HL denote one-error and hamming loss respectively. HAN denotes the hierarchical attention network. "(+)" represents that higher scores are better and "(-)" represents that lower scores are better. It can be seen that the proposed approach significantly outperforms the baselines.

5.1 Evaluation Metrics

Multi-label classification requires different evaluation metrics compared with traditional single-label classification. In this paper, we use the following widely-used evaluation metrics for multi-label classification.

- F1-score: We calculate macro F1 and micro F1, respectively. Macro F1 computes the metric independently for each label and then takes the average as the final score. Micro F1 aggregates the contributions of all labels to compute the average score.
- One-Error: One-error evaluates the fraction of examples whose top-ranked label is not in the gold label set.
- Hamming Loss: Hamming loss counts the fraction of the wrong labels to the total number of labels.

5.2 Baselines

We implement several widely-used multi-label classification methods as baselines, such as ML-KNN (Zhang and Zhou, 2007), Binary Relevance (Tsoumakas et al., 2010), Classifier Chains (Read et al., 2011), Label Powerset (Tsoumakas and Vlahavas, 2007). The details of baselines can be found at Appendices.

5.3 Results

The results on the test set are summarized in Table 2. The proposed approach significantly outperforms the baselines, with micro F1 of 64.5, macro F1 of 54.4, and one-error of 22.6, improving the metrics by 10.6, 21.4, and 7.9 respectively. The improvements are attributed to two parts, a hierarchical attention network and a label correlation mechanism. Only using the hierarchical attention network outperforms the baselines, which shows the effectiveness of hierarchically paying attention to different words and sentences. The greater F1-score is achieved by adding the proposed label graph, which demonstrates that the proposed label graph helps a lot by taking advantage of label correlations.

It can be clearly seen that with the help of soft training, the proposed method achieves the best performance. Especially, the micro F-score is improved from 62.8 to 64.5, and the one-error is reduced from 23.4 to 22.6. With the new loss function, the model not only knows how to distinguish the right labels from the wrong ones, but also can learn the label correlations that are useful knowledge, especially when the input data contains too much style unrelated words for the model to extract all necessary information.

5.4 Visualization Analysis

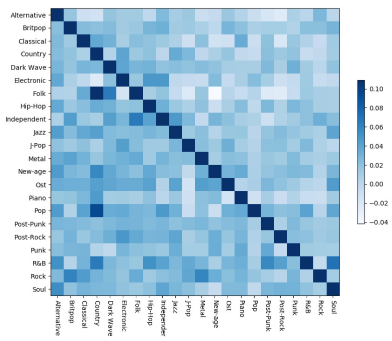


Figure 2: A heatmap of the automatically learned label graph. Deeper color represents closer relation. We can see that some obvious relations are well captured by the model, e.g., "Heavy Metal Music (Metal)" and "Rock", "Country Music (Country)" and "Folk".

Figure 2 shows a whole heatmap of the automatically learned label graph, for convenience of display, we have subtracted the label graph by an identity matrix. We can see from the picture that some obvious music style relations are well captured. For "Country Music", the most related label is "Folk Music". In reality, these two music styles are highly similar and the boundary between them is not well-defined. For three kinds of rock music, "Heavy Metal Music", "Britpop Music", and "Alternative Music", the label graph correctly captures that the most related label for them is “Rock”. For a more complicated relation where “Soul Music” is highly linked with two different labels, “R&B” and “Jazz”, the label graph also correctly capture such relation. These examples demonstrate that the proposed approach performs well in capturing relations among music styles.

5.5 Selected Exmaples

Ground Truth	Without LCM	With LCM
Britpop[5], Rock	Britpop	Britpop, Rock
Hip-Hop[6], Pop, R&B[7]	Electronic Music, Pop	Pop, R&B
Pop, R&B	Pop, Rock, Britpop	Pop, R&B
Country Music, Folk, Pop	Country Music, Pop	Country Music, Pop, Folk
Classical Music, New-Age Music[8], Piano Music	Piano Music, Classical Music	Piano Music, New-Age Music, Classical Music

Table 3: Examples generated by the methods with and without the label correlation mechanism (LCM). The labels correctly predicted by two methods are shown in blue. The labels correctly predicted by the method with the label correlation mechanism are shown in orange. We can see that the method with the label correlation mechanism classifies music styles more precisely.

For clearer understanding, we compare several examples generated with and without the label correlation mechanism in Table 3. By comparing gold labels and predicted labels generated by different methods, we find that the proposed label correlation mechanism identifies the related styles more precisely. This is mainly attributed to the learned label correlations. For example, the correct prediction in the first example shows that the label correlation mechanism captures the close relation between “Britpop” and “Rock”, which helps the model to generate an appropriate prediction.

5.6 Error Analysis

Although the proposed method has achieved significant improvements, we also notice that there are some failure cases. In this section, we give the detailed error analysis.

First, the proposed method performs worse on the styles with low frequency in the training set. Table 4 compares the performance on the top 5 music styles of highest and lowest frequencies. As we can see, the top 5 fewest music styles get much worse results than top 5 most music styles. This is because the label distribution is highly imbalanced where unpopular music styles have too little training data.

Second, we find that some music items are wrongly classified into the styles that are similar with the gold styles. For example, a sample with a gold set [Country Music] is wrongly classified into [Folk] by the model. The reason is that some music styles share many common elements and only subtly differ from each other. It poses a great challenge for the model to distinguish them. For future work, we would like to research how to effectively address this problem.

Most Styles	% of Samples	Micro F1
Rock	30.4	75.8
Independent Music	30.0	64.8
Pop	26.2	67.1
Folk Music	21.9	73.7
Electronic Music	13.9	61.8
Fewest Styles	**% of Samples**	**Micro F1**
Jazz	4.3	37.5
Heavy Metal Music	3.9	55.6
Hip-Hop	3.1	7.5
Post-punk	2.5	17.1
Dark Wave	1.3	17.4

Table 4: The performance of the proposed method on the most and fewest styles.

6 Conclusions

In this paper, we focus on classifying multi-label music styles with user reviews. To meet the challenge of label correlations, we propose a label-graph neural network and a soft training mechanism. Experiment results have showed the effectiveness of the proposed approach. The visualization of label graph also shows that our method performs well in capturing label correlations.

Acknowledgments

We thank all reviewers for providing the thoughtful and constructive suggestions. This work was supported in part by National Natural Science Foundation of China (No. 61673028). Email correspondence to Xu Sun.

---

[5] Britpop is a style of British Rock.

[6] Hip-Hop is a mainstream Pop style.

[7] Rhythm and Blues, often abbreviated as R&B, is a genre of popular music.

[8] New-Age Music is a genre of music intended to create artistic inspiration, relaxation, and optimism. It is used by listeners for yoga, massage, and meditation.

References

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Wei Chai and Barry Vercoe. 2001. Folk music classification using hidden markov models. In *Proceedings of International Conference on Artificial Intelligence*, volume 6.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2392–2396.

Michael Fell and Caroline Sporleder. 2014. Lyrics-based analysis and classification of music. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 620–631, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Siddharth Gopal and Yiming Yang. 2010. Multilabel classification with meta-level features. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 315–322.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. 2004. Maximal margin labeling for multi-topic text categorization. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 649–656.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wei Li, Xuancheng Ren, Damai Dai, Yunfang Wu, Houfeng Wang, and Xu Sun. 2018a. Sememe prediction: Learning semantic knowledge from unstructured textual wiki descriptions. *CoRR*, abs/1808.05437.

Wei Li, Zheng Yang, and Xu Sun. 2018b. Exploration on generating traditional chinese medicine prescription from symptoms with an end-to-end method. *CoRR*, abs/1801.09030.

Junyang Lin, Qi Su, Pengcheng Yang, Shuming Ma, and Xu Sun. 2018. Semantic-unit-based dilated convolution for multi-label text classification. *arXiv preprint arXiv:1808.08561*.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. 2017. Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*.

Aggelos Pikrakis, Sergios Theodoridis, and Dimitris Kamarotos. 2006. Classification of musical patterns using variable duration hidden markov models. *IEEE Trans. Audio, Speech & Language Processing*, 14(5):1795–1807.

Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. 2007. Correlative multi-label video annotation. In *Proceedings of the 15th International Conference on Multimedia 2007, Augsburg, Germany, September 24-29, 2007*, pages 17–26.

Dan Qin and GZ Ma. 2005. Music style identification system based on mining technology. *Computer Engineering and Design*, 26:3094–3096.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

Chris Sanden and John Z. Zhang. 2011. Enhancing multi-label music genre classification through ensemble techniques. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 705–714.

Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017. Label embedding network: Learning label representation for soft training of deep networks. *CoRR*, abs/1710.10393.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 667–685.

Grigorios Tsoumakas and Ioannis P. Vlahavas. 2007. Random k -labelsets: An ensemble method for multilabel classification. In *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*, pages 406–417.

Fei Wang, Xin Wang, Bo Shao, Tao Li, and Mitsunori Ogihara. 2009. Tag integrated multi-label music style classification with hypergraph. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, pages 363–368.

Mei Wang, Xiangdong Zhou, and Tat-Seng Chua. 2008. Automatic image annotation via local multi-label classification. In *Proceedings of the 7th ACM International Conference on Image and Video Retrieval, CIVR 2008, Niagara Falls, Canada, July 7-9, 2008*, pages 17–26.

Bingzhen Wei, Xuancheng Ren, Xu Sun, Yi Zhang, Xiaoyan Cai, and Qi Su. 2018. Regularizing output distribution of abstractive chinese social media text summarization for improved semantic consistency. *CoRR*, abs/1805.04033.

Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao, and Qi Tian. 2003. Musical genre classification using support vector machines. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–429.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Yatong Zhou, Taiyi Zhang, and Jiancheng Sun. 2006. Music style classification with a novel bayesian model. In *Advanced Data Mining and Applications, Second International Conference, ADMA 2006, Xi'an, China, August 14-16, 2006, Proceedings*, pages 150–156.

Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong. 2005. Multi-labelled classification using maximum entropy method. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 274–281.

A Appendices

A.1 Training Details

In the proposed method and baselines, we use skip gram (Mikolov et al., 2013) to get pre-trained word embeddings of reviews. The Jieba toolkit is used to split sentences into words. To help with the training of the label graph, we use soft target mechanism on the continuous label representations (Hinton et al., 2015; Sun et al., 2017; Liu et al., 2017) and add the negative of the l2 loss of the difference between the label graph and the identity matrix to the loss function. For evaluation, we introduce a hyper-parameter p. If the probability of a style is greater than p, we consider it as one of the final music styles. We tune hyper-parameters based on the performance on the validation set. We set p to 0.2, hidden size to 128, embedding size to 128, vocabulary size to 135K, learning rate to 0.001, and batch size to 128. The optimizer is Adam (Kingma and Ba, 2014) and the maximum training epoch is set to 100. We choose parameters with the best Micro F1 scores on the validation set and then use the selected parameters on the test set.

A.2 Baselines

We implement the following widely-used multi-label classification methods for comparison. Their inputs are the music representations which are produced by averaging review representations. The review representation is obtained by averaging word embeddings.

- ML-KNN (Zhang and Zhou, 2007): It is a multi-label learning approach derived from the traditional K-Nearest Neighbor (KNN) algorithm.
- Binary Relevance (Tsoumakas et al., 2010): It decomposes a multi-label learning task into a number of independent binary learning tasks. It learns several single binary models without considering the dependencies among labels.
- Classifier Chains (Read et al., 2011): It takes label dependencies into account and keeps the computational efficiency of the binary relevance method.
- Label Powerset (Tsoumakas and Vlahavas, 2007): All classes assigned to an example are combined into a new and unique class in this method.
- MLP: It feeds the music representations into a multi-layer perceptron, and generates the probability of music styles through a sigmoid layer.

Different from the above baselines, the following two methods only take word embeddings as inputs. Similar to MLP, they produce label probability distribution by a sigmoid function.

- CNN: It includes two layers of CNN which has multiple convolution kernels.
- LSTM: It includes two layers of LSTM, which processes words and sentences separately to get the music representations.

Fact Discovery from Knowledge Base via Facet Decomposition

Zihao Fu[1], **Yankai Lin**[2], **Zhiyuan Liu**[2]*, **Wai Lam**[1]
[1] Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
[2] Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing, China

Abstract

During the past few decades, knowledge bases (KBs) have experienced rapid growth. Nevertheless, most KBs still suffer from serious incompletion. Researchers proposed many tasks such as knowledge base completion and relation prediction to help build the representation of KBs. However, there are some issues unsettled towards enriching the KBs. Knowledge base completion and relation prediction assume that we know two elements of the fact triples and we are going to predict the missing one. This assumption is too restricted in practice and prevents it from discovering new facts directly. To address this issue, we propose a new task, namely, fact discovery from knowledge base. This task only requires that we know the head entity and the goal is to discover facts associated with the head entity. To tackle this new problem, we propose a novel framework that decomposes the discovery problem into several facet discovery components. We also propose a novel autoencoder based facet component to estimate some facets of the fact. Besides, we propose a feedback learning component to share the information between each facet. We evaluate our framework using a benchmark dataset and the experimental results show that our framework achieves promising results. We also conduct extensive analysis of our framework in discovering different kinds of facts. The source code of this paper can be obtained from `https://github.com/thunlp/FFD`.

1 Introduction

Recent years have witnessed the emergence and growth of many large-scale knowledge bases (KBs) such as Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015), YAGO (Suchanek et al., 2007) and Wikidata (Vrandečić and Krötzsch, 2014) to store facts of the real world. Most KBs typically organize the complex structured information about facts in the form of triples (*head entity*, `relation`, *tail entity*), e.g., (*Bill Gates*, `CEOof`, *Microsoft Inc.*). These KBs have been widely used in many AI and NLP tasks such as text analysis (Berant et al., 2013), question answering (Bordes et al., 2014a), and information retrieval (Hoffmann et al., 2011).

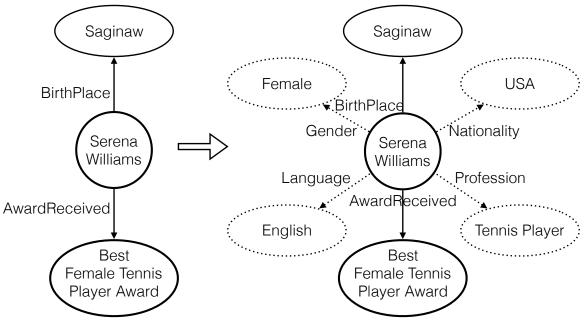


Figure 1: In the FDKB task, only the head entity is given. The relation and tail entity should be discovered simultaneously given the head entity.

The construction of these KBs is always an ongoing process due to the endless growth of real-world facts. Hence, many tasks such as knowledge base completion (KBC) and relation prediction (RP) are proposed to enrich KBs.

The KBC task usually assumes that one entity and the relation r are given, and another entity is missing and required to be predicted. In general, we wish to predict the missing entity in $(h, r, ?)$ or $(?, r, t)$, where h and t denote a head and tail entity respectively. Similarly, the RP task predicts the missing relation given the head and tail entities and their evidence sentences, i.e. filling $(h, ?, t)$. Nevertheless, the assumption of knowing two parts of the triple is too strong and is usually restricted in practice.

In many cases, we only know the entity of in-

---

* Corresponding author: Zhiyuan Liu (liuzy@tsinghua.edu.cn).

Proceedings of NAACL-HLT 2019, pages 2892–2901
Minneapolis, Minnesota, June 2 - June 7, 2019.

terest, and are required to predict both its attributive relations and the corresponding entities. As shown in Figure 1, the task is to predict the fact triples when given only the head entity, i.e. filling $(h, ?, ?)$. Since any entity can serve as the head entity for identifying its possible fact triples, this task should be more practical for real-world settings. This task is non-trivial since less information is provided for prediction. We name the task as Fact Discovery from Knowledge Base (FDKB).

Some existing methods such as knowledge base representation (KBR) can be applied to tackle the FDKB task with simple modifications. KBR models typically embed the semantics of both entities and relations into low-dimensional semantic space, i.e., embeddings. For example, TransE (Bordes et al., 2013) learns low-dimensional and real-valued embeddings for both entities and relations by regarding the relation of each triple fact as a translation from its head entity to the tail entity. TransE can thus compute the valid score for each triple by measuring how well the relation can play a translation between the head and tail entities. Many methods have been proposed to extend TransE to deal with various characteristics of KBs (Ji et al., 2015, 2016; He et al., 2015; Lin et al., 2015a).

To solve the FDKB task using KBR, one feasible way is to exhaustively calculate the scores of all (r, t) combinations for the given head entity h. Afterwards, the highly-scored facts are returned as results. However, this idea has some drawbacks: (1) It takes all relations to calculate ranking scores for each head entity, ignoring the nature of the head entity. The combination of all possible relations and tail entities will lead to huge amount of computations. (2) A large set of candidate triples immerses the correct triples into a lot of noisy triples. Although the probability of invalid facts getting a high score is small, with the large size of the candidate set, the total number of invalid facts with high score is non-negligible.

To address the above issues, we propose a new framework named as fact facet decomposition (FFD). The framework follows human being's common practice to identify unknown facts: One typically firstly investigates which relation that a head may have, and then predicts the tail entity based on the predicted relation. This procedure actually utilizes information from several perspectives. Similarly, FFD decomposes fact discovery into several facets, i.e., head-relation facet, tail-relation facet, and tail inference facet, and model each facet respectively. The candidate fact is considered to be correct when all of the facets are trustworthy. We propose a novel auto-encoder based entity-relation component to discover the relatedness between entities and relations. Besides, we also propose a feedback learning component to share the information between each facet.

We have conducted extensive experiments using a benchmark dataset to show that our framework achieves promising results. We also conduct an extensive analysis of the framework in discovering different kinds of facts. The contributions of this paper can be summarized as follows: (1) We introduce a new task of fact discovery from knowledge base, which is more practical. (2) We propose a new framework based on the facet decomposition which achieves promising results.

2 Related Work

In recent years, many tasks (Wang et al., 2017) have been proposed to help represent and enrich KBs. Tasks such as knowledge base completion (KBC) (Bordes et al., 2013; Wang et al., 2014; Ji et al., 2015, 2016; Wang et al., 2017) and relation prediction (RP) (Mintz et al., 2009; Lin et al., 2015a; Xie et al., 2016) are widely studied and many models are proposed to improve the performance on these tasks. However, the intention of these tasks is to test the performance of models in representing KBs and thus they cannot be used directly to discover new facts of KBs. Moreover, our FDKB task is not a simple combination of the KBC and RP task since both of these two tasks require to know two of the triples while we assume we only know the head entity.

A common approach to solving these tasks is to build a knowledge base representation (KBR) model with different kinds of representations. Typically, one element of the triples is unknown. Then, all entities are iterated on the unknown element and the scores of all combinations of the triples are calculated and then sorted.

Many works focusing on KBR attempt to encode both entities and relations into a low-dimensional semantic space. KBR models can be divided into two major categories, namely translation-based models and semantic matching models (Wang et al., 2017).

Translation-based models such as TransE (Bordes et al., 2013) achieves promising performance in KBC with good computational efficiency. TransE regards the relation in a triple as a translation between the embedding of head and tail entities. It means that TransE enforces that the head entity vector plus the relation vector approximates the tail entity vector to obtain entity and relation embeddings. However, TransE suffers from problems when dealing with 1-to-N, N-to-1 and N-to-N relations. To address this issue, TransH (Wang et al., 2014) enables an entity to have distinct embeddings when involving in different relations. TransR (Lin et al., 2015b) models entities in entity space and uses transform matrices to map entities into different relation spaces when involving different relations. Then it performs translations in relation spaces. In addition, many other KBR models have also been proposed to deal with various characteristics of KBs, such as TransD (Ji et al., 2015), KG2E (He et al., 2015), PTransE (Lin et al., 2015a), TranSparse (Ji et al., 2016).

Semantic matching models such as RESCAL (Nickel et al., 2011), DistMult(Yang et al., 2014), Complex (Trouillon et al., 2016), HolE (Nickel et al., 2016) and ANALOGY (Liu et al., 2017) model the score of triples by the semantic similarity. RESCAL simply models the score as a bilinear projection of head and tail entities. The bilinear projection is defined with a matrix for each relation. However, the huge amount of parameters makes the model prone to overfitting. To alleviate the issue of huge parameter space, DistMult is proposed to restrict the relation matrix to be diagonal. However, DistMult cannot handle the asymmetric relations. To tackle this problem, Complex is proposed assuming that the embeddings of entities and relations lie in the space of complex numbers. This model can handle the asymmetric relations. Later, Analogy is proposed by imposing restrictions on the matrix rather than building the matrix with vector. It achieves the state-of-the-art performance. Besides, (Bordes et al., 2011; Socher et al., 2013; Chen et al., 2013; Bordes et al., 2014b; Dong et al., 2014; Liu et al., 2016) conduct the semantic matching with neural networks. An energy function is used to jointly embed relations and entities.

3 Problem Formulation

We denote $\mathcal{E}$ as the set of all entities in KBs, $\mathcal{R}$ is the set containing all relations. $|\mathcal{E}|$ and $|\mathcal{R}|$ stand for the size of each set respectively. A fact is a triple (h, r, t) in which $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. $\mathcal{T}$ is the set of all true facts.

When a head entity set $\mathcal{H}$ is given, a new fact set is to be discovered based on these head entities. The discovered fact set is denoted as $\mathcal{T}_d = \{(h, r, t) | h \in \mathcal{H}\}$. Our goal is to find a fact set $\mathcal{T}_d$ that maximize the number of correct discovered facts:

$$\begin{aligned} &\max_{\mathcal{T}_d} |\mathcal{T}_d \cap \mathcal{T}| \\ &\text{s.t. } |\mathcal{T}_d| = K, \end{aligned} \tag{1}$$

in which K is a user-specified size.

4 Methodology

4.1 Fact Facet Decomposition Framework

Problem (1) is intractable since the set $\mathcal{T}$ is unknown. We tackle this problem by estimating a fact confidence score function $c(h, r, t)$ for each fact in $\mathcal{T}_d$ and maximize the total score. The problem is then formulated as:

$$\begin{aligned} &\max_{\mathcal{T}_d} \sum_{(h,r,t) \in \mathcal{T}_d} c(h, r, t) \\ &\text{s.t. } |\mathcal{T}_d| = K. \end{aligned} \tag{2}$$

To integrate the information from various facets of the fact, our framework, known as Fact Facet Decomposition (FFD) framework, decomposes the fact discovery problem into several facet-oriented detection tasks. A fact is likely to be correct if all facets provide supportive evidence. The facets are as follows:

1. Head-relation facet: A fact is likely true, if the head entity has a high probability of containing the relation. This is denoted as $f_h(r)$;
2. Tail-relation facet: A fact is likely true, if the tail entity has a high probability of containing the relation. This is denoted as $f_t(r)$;
3. Tail inference facet: A fact is likely true, if the score of the tail entity is high with respect to the given head and relation. This is denoted as $f_{h,r}(t)$.

Therefore, the facet confidence score can be expressed as:

$$c(h, r, t) = \lambda_1 f_h(r) + \lambda_2 f_t(r) + \lambda_3 f_{h,r}(t), \tag{3}$$

where $\lambda_1, \lambda_2, \lambda_3$ are weight parameters. The head-relation facet and the tail-relation facet can be both

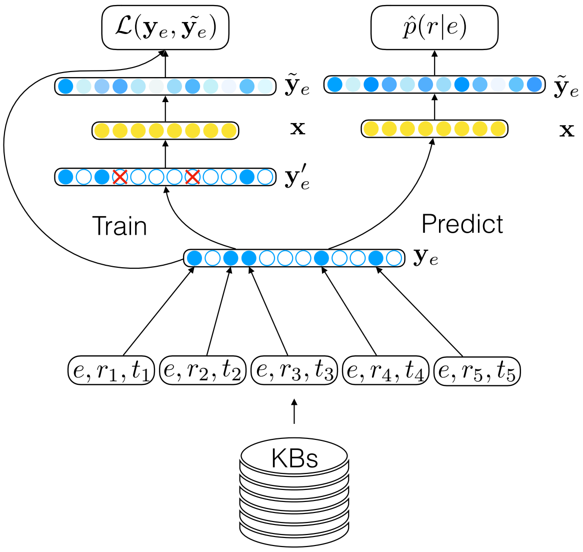


Figure 2: The structure of the entity-relation component.

modeled with an entity-relation facet component. The tail inference facet can be modeled by a KBR component.

4.1.1 Entity-relation Facet Component

The entity-relation component estimates the probability of a relation given an entity. The structure is shown in Figure 2. It is modeled as the log of the estimated conditional probability:

$$f_e(r) = \log \hat{p}(r|e), \tag{4}$$

where $e = h$ or t. $\hat{p}(r|e)$ aims at measuring the probability of a relation that this entity may have. In order to estimate this probability, the existing relations of a head or tail entity is used to infer other related relations. For example, if a head entity has an existing fact in which the relation is "BirthPlace", we may infer that this head entity may be a person and some relations such as "Gender", "Language" may have a high probability of association with this head entity. Therefore, the problem is transformed into a problem that estimates the relatedness between relations. To infer the probability of each relation based on existing relations, we employ a denoising auto-encoder (Vincent et al., 2008) which can recover almost the same representation for partially destroyed inputs. Firstly, facts related to an entity is extracted from the KBs. Then, this entity is encoded by the existing relations. Let $\mathbf{y}_e \in \mathbb{R}^{|\mathcal{R}|}$ be the 0-1 representation of relations that e has. $\mathbf{y}_{ei}$ indicates whether the entity e has the relation i or not. During the training phase, non-zero elements in $\mathbf{y}_e$ is randomly set to zero and the auto-encoder is trained to recover the corrupted elements. The corrupted vector is denoted as $\mathbf{y}'_e$.

Formally, our structure encoder first maps the corrupted one-hot vector $\mathbf{y}'_e$ to a hidden representation $\mathbf{x} \in \mathbb{R}^{d_1}$ of the entity through a fully connected layer:

$$\mathbf{x} = \tanh(\mathbf{W}_f \mathbf{y}'_e + \mathbf{b}_f), \tag{5}$$

where $\mathbf{W}_f \in \mathbb{R}^{d_1 \times |\mathcal{R}|}$ is the translation matrix and $\mathbf{b}_f \in \mathbb{R}^{d_1}$ is the bias vector. $\mathbf{x}$ is the vector representation of the entities in a hidden semantic space. In this space, similar entities are close to each other while entities of different types are far from each other. If some relations are missing, the fully connected layer will also map the entity into a nearby position.

Afterwards, $\mathbf{x}$ is used to recover the probability distribution for all relations through a fully connected layer and a sigmoid layer:

$$\tilde{\mathbf{y}}_e = \mathrm{sigmoid}(\mathbf{W}_g \mathbf{x} + \mathbf{b}_g), \tag{6}$$

where $\mathbf{W}_g \in \mathbb{R}^{|\mathcal{R}| \times d_1}$ and $\mathbf{b}_g \in \mathbb{R}^{|\mathcal{R}|}$ is the weight matrix and bias vector of the reverse mapping respectively. $\tilde{\mathbf{y}}_e$ is the recovered probability distribution of each relation (therefore, the sum of each element in $\tilde{\mathbf{y}}_e$ does not necessarily equal to 1). This layer will map the entity representation in the semantic space into a probability vector over all relations. Since similar entities are located in the adjacent area, they are likely to have a similar relation probability. Therefore, the probability of missing relations will also be high though the relations are unknown.

We use the original one-hot representation of the relations and the recovered relation probability to calculate a loss function:

$$\begin{aligned}\mathcal{L}(\mathbf{y}_e, \tilde{\mathbf{y}}_e) = -\sum_{e=1}^{|\mathcal{E}|}\sum_{i=1}^{|\mathcal{R}|}\{\mathbf{y}_{ei}\log(\tilde{\mathbf{y}}_{ei}) + \\ (1-\mathbf{y}_{ei})\log(1-\tilde{\mathbf{y}}_{ei})\}.\end{aligned} \tag{7}$$

The loss function forces the output $\tilde{\mathbf{y}}_{ei}$ to be consistent with $\mathbf{y}_{ei}$ which makes it capable to discover all related relations from known relations. It can be optimized with an Adam (Kingma and Ba, 2015) based optimizer.

When predicting new facts, the one-hot representation $\mathbf{y}_e$ is sent into the auto-encoder directly instead of using the corrupted representation. The result $\tilde{\mathbf{y}}_e$ is the estimated probability of each relation, i.e.

$$\hat{p}(r = i|e) = \tilde{\mathbf{y}}_{ei}. \tag{8}$$

This probability will be high if relation i is closely related to the existing relations of the entity e.

4.1.2 Tail Inference Facet Component

We use a KBR component to model the tail inference facet $f_{h,r}(t)$. Three KBR models are investigated namely DistMult, Complex, and Analogy.

The DistMult model defines the score function as $f_r(h,t) = \mathbf{h}^T \text{diag}(\mathbf{r})\mathbf{t}$, in which $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are vector representation of the head, relation and tail respectively. The learning objective is to maximize the margin between true facts and false facts. It can decrease the score of the wrong facts and increase the the score of the true facts at the same time.

The Complex model employs complex number as the KBR embedding. Therefore, the score function is defined as $f_r(h,t) = \text{Re}(\mathbf{h}^T \text{diag}(\mathbf{r})\bar{\mathbf{t}})$, in which $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are complex vectors and $\bar{\mathbf{t}}$ stands for the conjugate of $\mathbf{t}$.

The Analogy model does not restrict the relation matrix to be diagonal. Therefore, the score function is $f_r(h,t) = \mathbf{h}^T \mathbf{M_r} \mathbf{t}$, in which $\mathbf{M_r}$ is the matrix corresponding to the relation r. Since many relations satisfy normality and commutativity requirements, the constraints can thus be set as $W_r W_r^T = W_r^T W_r, \forall r \in \mathcal{R}$ and $W_r W_{r'} = W_{r'} W_r, \forall r, r' \in \mathcal{R}$. Solving such a problem is equivalent to optimizing the same objective function with the matrix constrained to almost-diagonal matrices(Liu et al., 2017).

After the score function is calculated, the tail inference facet $f_{h,r}(t)$ is modeled by a softmax function:

$$f_{h,r}(t) = \log \hat{p}(t|h,r) = \log \frac{e^{f_r(h,t)}}{\sum_{t' \in \mathcal{E}} e^{f_r(h,t')}}. \tag{9}$$

It should be noted that the normalizing step is only conducted on the tail entities since the head and relation are the input of the model. We only use these three models due to the limited space. Other models can be embedded into our framework easily in the same way.

4.2 Fact Discovery Algorithm

As mentioned above, we need to calculate $f_h(r)$, $f_t(r)$ and $f_{h,r}(t)$. $f_h(r)$ and $f_t(r)$ are computed by the entity-relation component while $f_{h,r}(t)$ is computed by the tail inference component. Recall that a fact is likely to be true when all the facets exhibit strong support. In other words, we can prune away the fact if one of the facets is low and stop calculating other facets. Based on this strategy, we design two additional constraints on Problem (2). Therefore, this method can be viewed as a shrink of the constraint space of the optimization problem. The new problem can be expressed as:

$$\begin{aligned} \max_{\mathcal{T}_d} \quad & \sum_{(h,r,t) \in \mathcal{T}_d} \{\lambda_1 f_h(r) + \lambda_2 f_t(r) + \lambda_3 f_{h,r}(t)\} \\ \text{s.t.} \quad & h \in \mathcal{H}; |\mathcal{T}_d| = K \\ & f_h(r) > \tau_h; \ \sum_r \mathbb{1}(f_h(r) > \tau_h) = n_h \\ & f_t(r) > \tau_t; \ \sum_r \mathbb{1}(f_t(r) > \tau_t) = n_t, \end{aligned} \tag{10}$$

where $\mathbb{1}_A(x)$ is an indicator function. $\mathbb{1}_A(x) = 1$ if $x \in A$ and $\mathbb{1}_A(x) = 0$ otherwise. n_h and n_t are the user-specified parameters indicating top-n_h or top-n_t relations are considered. λ_1, λ_2 and λ_3 are fixed hyperparameters.

Problem (10) is actually a mixed integer linear programming problem. We start to solve this problem from the constraints. Since $f_t(r)$ is independent of the given $\mathcal{H}$, it can be preprocessed and can be reused for other queries. When a head entity h is given, we firstly calculate $f_h(r)$ and get top-n_h relations ranked by $f_h(r)$. Then, for each relation, $f_t(r)$ is used to get the top-n_t entities. Afterwards, the tail inference facet $f_{h,r}(t)$ will be calculated for all remaining relations and entities and top-n_f triples will be cached. Finally, top-$\bar{K}$ facts ranked by the facet confidence score $c(h,r,t)$ is returned as the new facts discovered for the entity h, where $\bar{K} = K/|\mathcal{H}|$ stands for the average fact number for each head entity.

4.3 Feedback Learning

The three facets depict the characteristics of the KBs from different perspectives. For example, the head-relation facet indicates which relation the head entity may have. The tail-relation facet can be interpreted in a similar manner. We propose a feedback learning (FL) component for the facets to share the information in different perspective with each other. FL feeds the predicted facts back to the training set to enhance the training procedure and iterates predicting and training several times. In the iteration, the information from different perspectives is shared with each facet via the newly added facts.

Specifically, after predicting the top-n_h facts for each head entity, we select top-n_{fb} ($n_{fb} < n_h$) most probable facts according to the score of each triple and then feed them into the existing knowledge base for re-training the FFD model. We repeat the above updating operation several rounds.

5 Experiment

5.1 Dataset

We evaluate our framework by re-splitting a widely used dataset FB15k (Bordes et al., 2013), which is sampled from Freebase. It contains $1,345$ relations and $14,951$ entities. In FB15k, some of the testing set's head entities do not appear in the training set as head. To evaluate our framework, we construct the new dataset. We re-split FB15k into training ($\mathcal{T}_{train}$), validation ($\mathcal{T}_{valid}$) and testing ($\mathcal{T}_{test}$) set, and make sure that there is no overlap between the three sets. For all head entities in $\mathcal{H}$, a relation ratio $R\%$ is used to assign the facts into training and testing set. $R\%$ relations of a head entity are in the training set while the other $1 - R\%$ are in the testing set. In order to evaluate the task, we require that the head entities in $\mathcal{H}$ is the same as the testing head entity and is a subset of the training head set, i.e. $\mathcal{H} = \{h|(h,r,t) \in \mathcal{T}_{test}, \exists r,t \in \mathcal{E}\} \subset \{h|(h,r,t) \in \mathcal{T}_{train}, \exists r,t \in \mathcal{E}\}$. We set $R = 50$. After the splitting, the training, testing and validation set size is $509,339$, $41,861$ and $41,013$ respectively.

5.2 Comparison Models

To demonstrate the effectiveness of our framework, we provide several strong comparison models that can be used in solving this task.

5.2.1 Matrix Factorization Models (SVD and NMF)

MF models firstly count the frequency of all relation-tail pairs. Some low-frequency relation-tail pairs are ignored to save computational time. Afterwards, we build a (head, relation-tail) co-occurrence matrix $M^C \in \mathbb{R}^{|\mathcal{E}| \times p}$, in which p is the size of the relation-tail pair set. Each element M^C_{ij} in the matrix represents whether the head entity i has the relation-tail pair j or not. Then, the matrix will be decomposed by the product of two matrices, i.e.

$$M^C \approx WH, \tag{11}$$

in which $W \in \mathbb{R}^{|\mathcal{E}| \times k}, H \in \mathbb{R}^{k \times p}$. k is the hidden category number of the head and relation-tail pairs. The decomposition can be achieved in several ways with different assumptions. Two kinds of matrix decomposition models are used namely SVD (Halko et al., 2011) and NMF (Lee and Seung, 1999).

In the prediction stage, a new matrix is constructed by $M'^C = WH$. For each row in M'^C, we record top-$\bar{K}$ relation-tail pairs and their scores. The MF models always suffer from the sparsity problem since a lot of relation-tail pairs are ignored.

5.2.2 KBR+ Models (DistMult+, Complex+ and Analogy+)

The most straightforward method of estimating the fact confidence score $c(h,r,t)$ is to use KBR model directly to evaluate each triples' score. We exhaustively score all possible combinations of relations and tails and use the highly-scored facts to make up the set $\mathcal{T}_d$. We select some state-of-the-art models including DistMult (Yang et al., 2014), Complex (Trouillon et al., 2016) and Analogy (Liu et al., 2017). We denote them as DistMult+, Complex+ and Analogy+.

After a KBR model learns a score function $f_r(h,t)$, the probability of each (r,t) pair with respect to a given head entity can be estimated by a softmax function:

$$\hat{p}(r,t|h) = \frac{e^{f_r(h,t)}}{\sum_{r' \in \mathcal{R}} \sum_{t' \in \mathcal{E}} e^{f_{r'}(h,t')}}. \tag{12}$$

Afterwards, the score of each fact is sorted and top-$\bar{K}$ relation-tail pairs for a head entity are regarded as the predicted results.

5.3 Experimental Setup

There are 2,000 head entities in the testing set. Therefore, we predict the corresponding relation and tail entity with respect to these 2,000 head entities. In MF models, only relation-tail pairs that occur more than 3 times in the training set are considered (24,615 pairs in total). For each head entity, we set $\bar{K} = 50$. In KBR+, we also set $\bar{K} = 50$. For our framework, we set $n_h = n_t = 30$, $n_f = 10$, $\bar{K} = 50$, $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 0.5$. The auto-encoder iterates for 1,000 epochs and the learning rate for Adam is 0.005. For the feedback learning component, we set $n_{fb} = 20,000$. With this setting, each model returns 100,000 facts.

We use four evaluation metrics, including precision, recall MAP, and F1 in relation prediction. Precision is defined as the ratio of the true positive candidates' count over the number of all the retrieved candidates' count. Recall is defined as the ratio of the true positive candidates' count over all the positive facts' count in the testing set. MAP (Manning et al., 2008) is a common evaluation method in information retrieval tasks. F1 is defined as the harmonic mean of the precision and recall.

5.4 Experimental Results

The experimental result is shown in Table 1. From the experiment result, we observe that:

Method	MAP	precision	recall	F1
SVD	0.0873	0.0897	0.2143	0.1265
NMF	0.0827	0.0857	0.2048	0.1209
DistMult+	0.1086	0.1068	0.2552	0.1506
Complex+	0.2384	0.1608	0.3842	0.2267
Analogy+	0.2367	0.1606	0.3837	0.2265
FFD (DistMult)	0.2486	0.1939	0.4633	0.2734
FFD (Complex)	0.2723	0.1991	0.4758	0.2808
FFD (Analogy)	**0.2769**	**0.2001**	**0.4779**	**0.2821**
FFD (Analogy) w/o FL	0.2308	0.1978	0.4725	0.2788

Table 1: Results of our framework and comparison models.

1. FFD based model outperforms other models in all metrics. It illustrates the advantage of our decomposition design. Moreover, in FFD, using Analogy to predict $c(h, r, t)$ outperforms Complex. One reason is that the discovery algorithm harness the relatively large parameter space of Analogy and avoids some occasionally emerging wrong facts;
2. The relation of the head entity can be correctly predicted. This is because, in training, we remove some relations and the auto-encoder is trained to learn to recover the missing relations based on the remaining relations.
3. The MF based models (i.e. SVD and NMF) perform not as good as KBR+ models and FFD. The reason is partially due to the sparsity problem in MF models. A lot of relation-tail pairs have not been used as the feature and thus cannot be predicted;

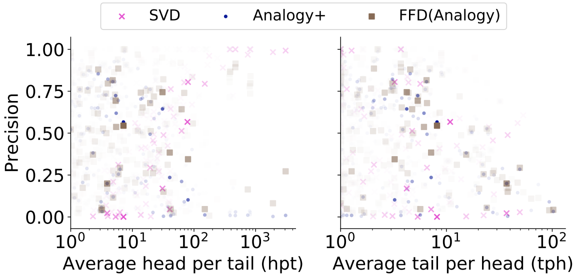


Figure 3: Precision on each relation. Deep color stand for the number of fact is large.

4. Different from the traditional KBC task, Complex performs slightly better than Analogy. One reason is that Analogy's constraint is looser than Complex. Therefore, it may easily predict wrong facts due to error propagation;
5. The ablation experiment shows that the feedback learning can improve the performance effectively.

To illustrate the capability of handling different kinds of relations, we plot the accuracy with respect to different kinds of relations. We use heads per tail (hpt) and tails per head (tph) index to represent the difficulty of each relation. If the relation's index is high, it means that each head of the relation may have more tails or vice versa. These relations are more difficult to predict. This is the similar problem of 1-N, N-1 and N-N relation in KBC task. The plot is shown in Figure 3. From the figure, we can observe that:

1. FFD can be adapted to all kinds of relations with different hpt and tph;
2. MF, KBR+, FFD models can handle relations with relatively high hpt but fail with high tph. This is because our goal is to predict relation and tail based on the head. Therefore, the choice may be harder to make with high tph;
3. As the hpt grows, the precision of SVD model also grows. The reason is that as hpt grows, the sparsity problem is alleviated. Therefore, the performance of SVD grows.

5.5 Sparsity Investigation

The MF model suffers from the sparsity problem since a lot of relation-pair does not appear in the training set. We examine the training set and observe that 97.46% relation-tail pairs does not appear and 0.34% relation-tail pairs appear for only one time. These pairs can hardly provide any information for the MF models either.

Relation Ratio	train	test	valid
10%	451,214	41,861	41,013
20%	462,395		
30%	475,841		
40%	491,498		
50%	509,339		

Table 2: Statistics of dataset with different relation ratio.

To test whether our framework is capable of dealing with the data sparsity problem. We remove training facts which contains head entities in $\mathcal{H}$ according to a specific ratio. We decrease the relation ratio $R\%$ from 50% to 10% to explore the effectiveness of our framework in discovering new facts. The dataset statistics is shown in Table 2. We apply FFD (Analogy) on each dataset. As shown in Table 3, precision, F1 and recall decrease since the data becomes more and more sparse. MAP increase slightly since it is averaged on all extracted facts. When the extracted facts number decreases, some facts rank at the tail with low scores are excluded.

Relation Ratio	MAP	precision	recall	F1
50%	0.2101	0.1851	0.4421	0.2609
40%	0.2099	0.1768	0.4224	0.2493
30%	0.2167	0.1686	0.4029	0.2378
20%	0.2236	0.1623	0.3878	0.2289
10%	0.2497	0.1534	0.3664	0.2162

Table 3: Result of dataset in different relation ratio.

Relation	Tail	In RT pair	SVD	Analog+	FFD (Analogy)
Located In	USA	√	√		√
Located In	California	√		√	√
Located In	Stanford				√
Educational Institution	Stanford Law School			√	√
Graduates Degree	Law Degree	√	√	√	√
Graduates Degree	Juris Doctor	√	√	√	√
Mail Address State	California	√		√	√
Mail Address City	Stanford				√
Parent Institution	Stanford University			√	√
Tuition Measurement	US Dollar	√	√		√
Webpage Category	WebPage	√	√		√

Table 4: Predicted facts by SVD, FFD+ and FFD (Analogy). "√" stands for whether the prediction is correct.

5.6 Case Study

We provide a case study to demonstrate the characteristics of different models and show that our FFD can utilize more information. We choose the head entity "Stanford Law School" (Freebase ID: /m/021s9n). The predicted facts of SVD, Analogy+ and FFD (Analogy) are shown in Table 4. From the table, we can observe that:

1. FFD (Analogy) can predict facts such as ("Located In", "Stanford") and ("Mail Address City", "Stanford") while other methods fail to. It implies that this model can predict some relation with multiple possible tails;

2. Analogy+ outperforms SVD in general while fails to exceed FFD (Analogy). The reason is that it fails to predict some general facts like ("Located In", "USA") or ("Tuition Measurement", "US Dollar"). This may due to the high scores given to some wrong facts;

3. The SVD model can only predict those facts whose relation and tail belong to the selected relation-tail pairs while Analogy+ and FFD (Analogy) can predict more facts;

4. SVD model prefers to predict some basic facts such as "Located In" and "Tuition Measurement". This is because those relations appear a lot of times in the training set and have limited possible tail entities. Therefore, it is easy for SVD model to make such prediction.

6 Conclusions and Future Work

In this paper, we introduce a new task of fact discovery from knowledge base, which is quite important for enriching KBs. It is challenging due to the limited information available about the given entities for prediction. We propose an effective framework for this task. Experimental results on real-world datasets show that our model can effectively predict new relational facts. We also demonstrate that the feedback learning approach is useful for alleviating the issue of data sparsity for the head entities with few facts.

Facts discovery from knowledge base is essential for enriching KBs in the real world. Despite the fact that our work shows some promising results, there still remains some challenges: (1) There exists much more internal information such as relational paths and external information such as text, figures and videos on the web, which can be used to further improve the performance. (2) The feedback learning approach in this paper is to simply utilize those confident predicted relational facts to enhance the model. Reinforcement learning may help us dynamically select those informative and confident relational facts.

Acknowledgments

The work described in this paper is partially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055093). Liu and Lin are supported by the National Key Research and Development Program of China (No. 2018YFB1004503) and the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007).

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014b. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, pages 301–306.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. In *Proceedings of ICLR*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of SIGKDD*, pages 601–610.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of CIKM*, pages 623–632.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, pages 687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of AAAI*, pages 985–991.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of EMNLP*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *ICML*, pages 2168–2178.

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*, pages 697–706. ACM.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *proceedings of ICML*, pages 2071–2080.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of ICML*, pages 1096–1103.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE TKDE*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, pages 1112–1119.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*, pages 2659–2665.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

A Richer-but-Smarter Shortest Dependency Path with Attentive Augmentation for Relation Extraction

Duy-Cat Can[1], **Hoang-Quynh Le**[1]*, **Quang-Thuy Ha**[1] **and Nigel Collier**[2]

[1]Faculty of Information Technology, VNU University of Engineering and Technology,
Hanoi, Vietnam
[2]Department of Theoretical and Applied Linguistics, University of Cambridge, UK
`{catcd,lhquynh,thuyhq}@vnu.edu.vn`, `nhc30@cam.ac.uk`

Abstract

To extract the relationship between two entities in a sentence, two common approaches are (1) using their shortest dependency path (SDP) and (2) using an attention model to capture a context-based representation of the sentence. Each approach suffers from its own disadvantage of either missing or redundant information. In this work, we propose a novel model that combines the advantages of these two approaches. This is based on the basic information in the SDP enhanced with information selected by several attention mechanisms with kernel filters, namely RbSP (Richer-but-Smarter SDP). To exploit the representation behind the RbSP structure effectively, we develop a combined deep neural model with a LSTM network on word sequences and a CNN on RbSP. Experimental results on the SemEval-2010 dataset demonstrate improved performance over competitive baselines. The data and source code are available at `https://github.com/catcd/RbSP`.

1 Introduction

One of the most fundamental tasks in natural language processing, as well as in information extraction, is Relation Extraction (RE), i.e., determining the semantic relation between pairs of named entities or nominals in a sentence or a paragraph. Take the following sentences from the SemEval-2010 task 8 dataset (Hendrickx et al., 2009) as examples:

(i) We put the soured $[cream]_{e1}$ *in the butter* $[churn]_{e2}$ *and started stirring it.*

(ii) The agitating $[students]_{e1}$ *also put up a* $[barricade]_{e2}$ *on the Dhaka-Mymensingh highway.*

Here the nominals 'cream' and 'churn' in sentence (i) are of relation `Entity-Destination(e1,e2)` while nominals 'students' and 'barricade' in sentence (ii) are of relations `Product-Producer(e2,e1)`.

The research history of RE has witnessed the development as well as the competition of a variety of RE methodologies. All of them are proven to be effective and have different strengths by leveraging different types of linguistic knowledge, however, also suffer from their own limitations. Some early studies stated that the shortest dependency path (SDP) in dependency tree is usually concise and contains essential information for RE (Bunescu and Mooney, 2005; Fundel et al., 2006). By 2016, this approach became dominant with many studies demonstrating that using SDP brings better experimental results than previous approaches that used the whole sentence (Xu et al., 2015a,b; Mehryary et al., 2016; Cai et al., 2016; Le et al., 2018). However, using the SDP may lead to the omission of useful information (i.e., negation, adverbs, prepositions, etc.). Recognizing this disadvantage, some studies have sought to improve SDP approaches, such as adding the information from the sub-tree attached to each node in the SDP (Liu et al., 2015) or applying a graph convolution over pruned dependency trees (Zhang et al., 2018b).

Another approach to extract the relation between two entities is using whole sentence in which both are mentioned. This approach seems to be slightly weaker than using the SDP since not all words in a sentence contribute equally to classify relations and this leads to unexpected noises (Nguyen and Grishman, 2015). However, the emergence and development of attention mechanism (Bahdanau et al., 2015) has re-vitalized this approach. For RE, the attention mechanism is capable of picking out the relevant words concerning target entities/relations, and then we can find critical words which determine primary useful se-

---

[*] Corresponding author

Proceedings of NAACL-HLT 2019, pages 2902–2912
Minneapolis, Minnesota, June 2 - June 7, 2019.

mantic information (Zhou et al., 2016; Verga et al., 2018). We therefore need to determine the object of attention, i.e., nominals themselves, their entity types or relation label. However, conventional attention mechanism on sequence of words cannot make use of structural information on dependency tree. Moreover, it is hard for machines to learn the attention weights from a long sequence of input text.

In this work we propose an enhanced representation for relations that combines the advantages of the above approaches. Basically, we focus on condensed semantic and syntactic information on the SDP. Compensating for the limitations of the SDP may still lead to missing information so we enhance this with syntactic information from the full dependency parse tree. Our idea is based on fundamental notion that the syntactic structure of a sentence consists of binary asymmetrical relations between words (Nivre, 2005). Since these dependency relations hold between a head word (parent, predicate) and a dependent word (children, argument), we try to use all child nodes of a word in the dependency tree to augment its information. Depending on a specific set of relations, it will turn out that not all children are useful to enhance the parent node; we select relevant children by applying several attention mechanisms with kernel filters. This new representation of relation is named Richer-but-Smarter SDP (RbSP).

Recently, deep neural networks (DNNs) have been effectively used to learn robust syntactic and semantic representations behind complex structures. Thus, we propose a novel DNN framework which combines Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Convolutional Neural Networks (CNN) (LeCun et al., 1989) with a multi-attention layer.

Our work has three main contributions:

- We proposed a novel representation of relation based on attentive augmented SDP that overcomes the disadvantages of traditional SDP.
- We improved the attention mechanism with kernel filters to capture the features from context vectors.
- We proposed an advanced DNN architecture that utilizes the proposed Richer-but-Smarter Shortest Dependency Path (RbSP) and other types of linguistic and architectural features.

2 Related Work

RE has been widely studied in NLP community for many years. Unsupervised (Hasegawa et al., 2004; Yan et al., 2009; Quan et al., 2014), semi-supervised (Chen et al., 2006; Carlson et al., 2010; Ammar et al., 2017) and distant supervision (Verga et al., 2018; Ji et al., 2017) methods have been proven effective for the task of detecting relations from unstructured text. However, in this paper, we mainly focus on supervised approaches, which usually have higher accuracy. In earlier RE studies, researchers focused on extracting various kinds of linguistic features, including both syntactic features and semantic cues (Chan and Roth, 2010; Nguyen and Grishman, 2014). However, all the feature-based methods depend strongly on the quality of designed features from an explicit linguistic pre-processing step.

Based on the idea that SDPs contain the essential information for RE, many studies exploit it with several refinements. Typical refinements include negative sampling (Xu et al., 2015a) and BRCNN (Cai et al., 2016) which model the directed shortest path. Liu et al. (2015) suggested incorporating additional network architectures to further improve the performance of SDP-based methods, which uses a recursive neural network to model the sub-tree. Some works utilized information over the whole dependency tree, such as Li et al. (2017) used dynamic extended tree conditioned LSTM for RE and Panyam et al. (2018) exploited whole dependency graph for relation extraction in biomedical text.

Recently, with the introduction and development of attention mechanism, many works tend to use whole sentence or paragraph and focus on the most relevant information using attention technique. Some studies apply a single attention layer, that focus on the word itself (Shen and Huang, 2016; Zhang et al., 2018a); word position (Zhang et al., 2017) and global relation embedding (Su et al., 2018). Other works apply several attention layers, such as word, relation and pooling attention (Wang et al., 2016), multi-head attention (Verga et al., 2018) and word- and entity-based attention (Jat et al., 2017). Luo et al. (2018) used a bidirectional Long Short-Term Memory architecture with an attention layer and a tensor layer for organizing the context information and detecting the connections between two nominals.

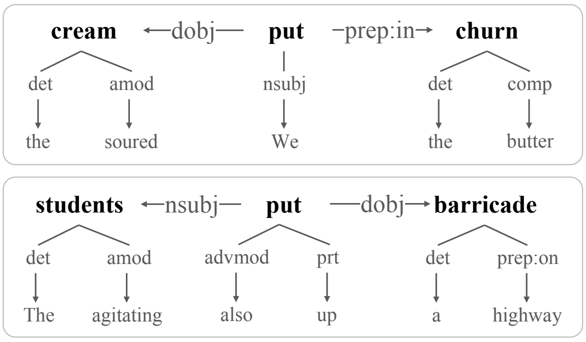


Figure 1: Examples of SDPs and attached child nodes.

3 Richer-but-Smarter SDP

As previously mentioned, we utilize the condensed information in the SDP to learn the relation between two nominals. The simple structure of the SDP is one of its weaknesses since there exists some useful information in dependency tree that does not appear in the SDP. This information can be leveraged to represent the relation more precisely. Two examples in Figure 1 belong to different relation types, but the paths between two nominals in these examples contain only one token (“**put**”). However, the meaning of token “**put**” in two SDPs are completely different. In this situation, it is difficult for the machine to distinguish the two shortest dependency paths from these instances.

We notice that the child nodes attached to the shortest dependency paths and their dependency relation from their parent can provide supplemental information for relation classification. In the previous examples, the sub-structure “−prt→ **up**” provides semantic information about token ‘**put**” in the specific sentence to make it discriminated from the stand-alone one. Based on similar observations, we propose the idea of combining subtree information with original SDP to form a more precise structure for classifying relations. In this RbSP structure each token t is represented by itself and its attached children on the dependency tree.

4 Proposed Model

The overall architecture of our proposed model is shown in Figure 2. Given a sentence and its dependency tree, we build our model on the SDP between two nominals and its directed children on the tree. Here, we mainly focus on the SDP representation, which is composed of dependency embeddings, token embeddings, and token's augmented information. After SDP representation phase, each token and dependency relation is transformed into a vector. This sequence of vectors is then fed to a convolutional neural network to capture the convolved features that can be used to determine which relation two nominals are of.

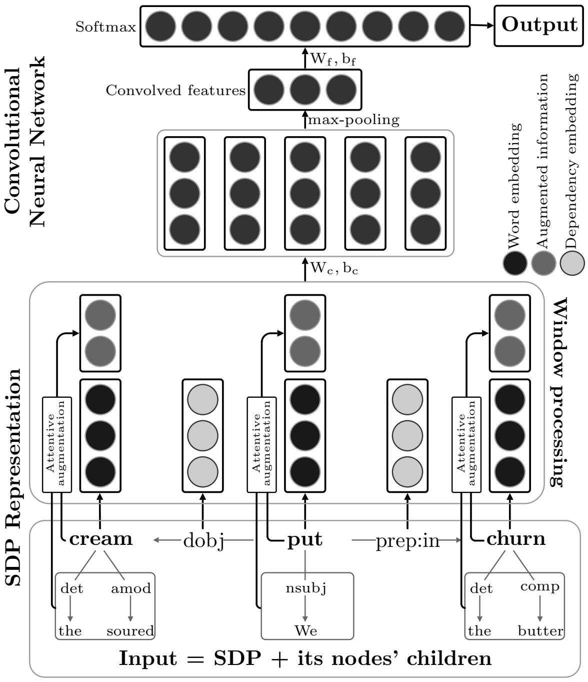


Figure 2: The architecture of RbSP model for relation classification. A CNN model is applied to the output of the SDP representation. Our proposed model takes the Augmented SDP between two nominals that includes dependencies, tokens and their children as input.

4.1 SDP Representation

The goal of this phase is to represent each component on the shortest path (dependency relation and token) by a corresponding vector. We concatenate the dependency type and dependency direction to form the embedding for a dependency relation, a non-linear transformer is followed to produce the final D-dimensional representation $\mathbf{d}_i \in \mathbb{R}^D$ of i-th dependency relation as follow:

$$\mathbf{d}_i = \tanh\left(\left[\mathbf{d}_i^{typ} \oplus \mathbf{d}_i^{dir}\right] \mathbf{W}_d + \mathbf{b}_d\right) \qquad (1)$$

where $\mathbf{d}^{typ} \in \mathbb{R}^{dim_{typ}}$ and $\mathbf{d}^{dir} \in \mathbb{R}^{dim_{dir}}$ are dependency type and direction respectively; $\mathbf{W}_d \in \mathbb{R}^{(dim_{typ}+dim_{dir})\times D}$ and $\mathbf{b}_d \in \mathbb{R}^D$ are trainable parameters of the network.

For token representation, as mentioned above, we assume that each token should be interpreted by itself and its children. Then, the word information $\mathbf{t}_i$ of each token on the SDP is concatenated with its attentive augmented information $\mathbf{a}_i$ based on the attached children (which is calculated by Multi-layer attention with Kernel filters, see Section 4.2). In this work, we utilize four types of embeddings to represent the word information of each token, including:

- ***Pre-trained fastText embeddings*** (Bojanowski et al., 2017): which learned the word representation based on its external context.
- ***Character-based embeddings***: we use an internal LSTM to learn the information about word morphology (like the prefix or suffix).
- ***POS tag embeddings***: we embed the token's grammatical tag using a randomly initialized look-up table and update this parameter on model learning phase.
- ***WordNet embeddings***: which is in form of a sparse vector that figure out which basic WordNet synsets the token belongs to.

To take advantage of the original sentence sequence information, we use a recurrent neural network with LSTM units to pick up the information along the sentence $\mathbf{S} = \{\mathbf{t}_i\}_{i=1}^{n}$ as follow:

$$\mathbf{H} = \overleftrightarrow{\text{biLSTM}}(\mathbf{S}) = \{\mathbf{h}_i\}_{i=1}^{n} \tag{2}$$

Each token t_i is then augmented by the corresponding hidden state $\mathbf{h}_i$ from $\mathbf{H}$. Finally, this concatenation is transformed into an X-dimensional vector to form the representation $\mathbf{x}_i \in \mathbb{R}^X$ of the token. I.e.,

$$\mathbf{x}_i = \tanh\left(\left[\mathbf{t}_i \oplus \mathbf{a}_i \oplus \mathbf{h}_i\right] \mathbf{W}_x + \mathbf{b}_x\right) \tag{3}$$

where $\mathbf{W}_x$ and $\mathbf{b}_x$ are trainable parameters of the network.

4.2 Multi-layer attention with Kernel filters

To capture the appropriate augmented information from the child nodes of each token, we propose a novel multi-layer attention with kernel filters architecture. As illustrated in Figure 3, we employ two sequential attention layers on the children of a token to produce children context vectors. Afterward, to utilize all informative child nodes and preserve the integrity of the word information, we capture the token's augmented information using kernel filters instead of using the average of context vectors weighted by multi-layer attention.

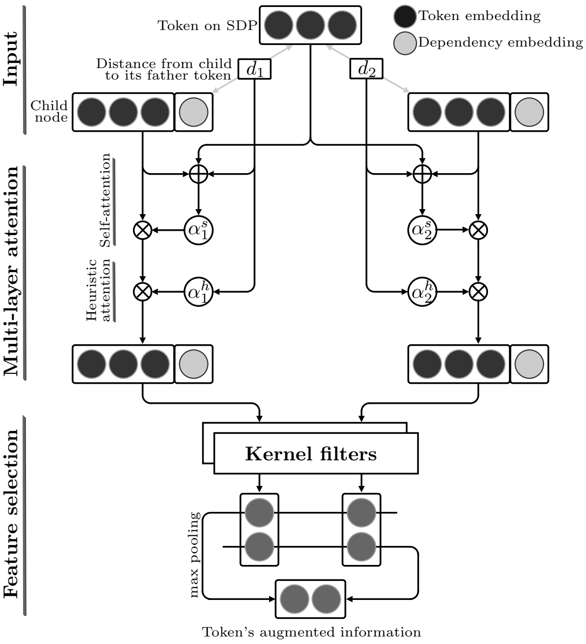


Figure 3: The multi-layer attention architecture to extract the augmented information from the children of a token on SDP. $\bigoplus$ denotes the concatenation of components. $\bigotimes$ denotes the scalar multiplication. Kernel filters are applied on context vectors to capture the features by convolution operation.

Given a token t and their child nodes, we first represent every token by a real-valued vector to provide lexical semantic features. Token t is transformed into a token embedding vector $\mathbf{t} \in \mathbb{R}^{dim}$ which is the concatenation of its word embedding and part-of-speech (POS) tag embedding. To utilize all the information in the sub-structure of token's children, we form a child node not only by its token embedding as in parent node but also by the dependency relation from its direct ancestor on the sentence's parse tree. Suppose t has a set C of M children, i.e., $C = \{c_1, c_2, ..., c_M\}$. Our model represents each child in C with a real-valued vector $\mathbf{c}_i \in \mathbb{R}^{dim+dim_{dep}}$. To additionally capture information about the child node to the target token, we incorporate the position embeddings d_i to reflect the relative distances between the i-th child's token to the target token on the original sentence.

We then apply a simple self-attentive network to child nodes $\{\mathbf{c_i}\}_{i=1}^{M}$ where the attention weights are calculated based on the concatenation of themselves with parent information and distance from parent, as follow:

$$\begin{aligned}\bar{\mathbf{C}} &= \left\{\mathbf{c}_i \oplus \mathbf{t} \oplus d_i\mathbf{w}_d\right\}_{i=1}^{M} = \left\{\bar{\mathbf{c}}_i\right\}_{i=1}^{M} \\ \mathbf{e} &= \left\{\bar{\mathbf{c}}_i\mathbf{W}_e + b_e\right\}_{i=1}^{M} = \left\{e_i\right\}_{i=1}^{M} \\ \alpha_i^s &= \frac{\exp(e_i)}{\sum_{k=1}^{M}\exp(e_k)}\end{aligned} \tag{4}$$

where $\oplus$ denotes the concatenation operation; $\mathbf{w}_d \in \mathbb{R}^{dim_d}$ is the base distance embedding; $\mathbf{W}_e \in \mathbb{R}^{(2dim+dim_{dep}+dim_d)\times 1}$ and $b_e \in \mathbb{R}$ are weight and bias term. The self-attentive context vector $\mathbf{a}^s$ of the target token is the weighted sum of the self-attentive children context vectors based on the weights as follows:

$$\begin{aligned}\mathbf{c}_i^s &= \alpha_i^s\mathbf{c}_i \\ \mathbf{a}^s &= \sum_i \mathbf{c}_i^s\end{aligned} \tag{5}$$

We observe that the importance of a child node to the parent node depends on the distance between them on the original sentence. Therefore, we apply a heuristic attentive layer on the self-attentive children context vectors based on the distances $d_1, d_2, ..., d_M$ to keep track of how close each child is to the target token. We heuristically choose the activation function for the distances $d_1, d_2, ..., d_M$ as $f(d) = \beta d^2$ with $\beta = -0.03$, and a $\mathrm{softmax}$ layer is followed to calculate the heuristic attention weight. I.e.,

$$\begin{aligned}\alpha_i^h &= \frac{\exp(\beta d_i^2)}{\sum_{k=1}^{N}\exp(\beta d_k^2)} \\ \mathbf{c}_i^h &= \alpha_i^h\mathbf{c}_i \\ \mathbf{a}^h &= \sum_i \mathbf{c}_i^h\end{aligned} \tag{6}$$

The multi-attentive context vector $\mathbf{a}^h$ is a synthetic representation of all child nodes with the target token node taken into account. Since the child nodes are usually distinct from each other, an average vector is not suitable to represent the children information. We propose to use the kernel filters to capture the relevant and important information from the output of the multi-attention layer. K kernel filters are applied to each child's attentive vector to produce K features from each child. I.e.,

$$\mathbf{F} = \left\{\mathrm{ReLU}\left(\mathbf{c}_i^h\mathbf{W}_f + \mathbf{b}_f\right)\right\}_{i=1}^{M} \tag{7}$$

where $\mathbf{W}_f \in \mathbb{R}^{(2dim+dim_{dep}+dim_d)\times K}$ is the weight of K kernel filters; and $\mathbf{b}_f \in \mathbb{R}^K$ is bias term. Finally, to produce the final augmented information $\mathbf{a}$, we apply a max-pooling (Boureau et al., 2010) layer to the feature matrix $\mathbf{F}$ and select the most important features as follow:

$$\mathbf{a} = \left\{\max\left(\mathbf{F}_k^{\intercal}\right)\right\}_{k=1}^{K} \tag{8}$$

4.3 CNN on RbSP

After SDP representation layer, the input SDP is transformed into:

$$\mathbf{SDP} = \left[\mathbf{x}_1, \overleftarrow{\mathbf{d}_1}, \mathbf{x}_2, ..., \mathbf{x}_{N-1}, \overrightarrow{\mathbf{d}_{N-1}}, \mathbf{x}_N\right] \tag{9}$$

where the over arrow on $\mathbf{d}_i$ denotes the direction of the dependency relation. We build the CNN model on this $\mathbf{SDP}$; our model is similar to the model of Xu et al. (2015a). In general, let us define the vector $\mathbf{x}_{i:i+j}$ as the concatenation of j tokens and $j-1$ dependency relation between them. I.e.,

$$\mathbf{x}_{i:i+j} = \mathbf{x}_i \oplus \mathbf{d}_i \oplus \mathbf{x}_{i+1} \oplus ... \oplus \mathbf{d}_{i+j-2} \oplus \mathbf{x}_{i+j-1} \tag{10}$$

The convolution operation with region size r applies k filters to all possible window of r successive tokens to produce convolved feature map. We then gather the most important features by applying a max pooling (Boureau et al., 2010) layer over the entire feature map. I.e., the convolutional layer computes the i-th element of the convolved feature vector $\mathbf{f}$ as follows:

$$\mathbf{f}_i = \max_{0\le j\le N-r+1}\left[\mathbf{x}_{j:j+r}\mathbf{W}_c + \mathbf{b}_c\right]_i \tag{11}$$

where $\mathbf{W}_c \in \mathbb{R}^{(rX+(r-1)D)\times k}$ and $\mathbf{b}_c \in \mathbb{R}^k$ are the weight matrix and bias vector of the convolutional layer. The output $\mathbf{f}$ of the convolutional layer is then fed to a softmax classifier to predict a $(K+1)$-class distribution over labels $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \mathrm{softmax}\left(\mathbf{f}\mathbf{W}_y + \mathbf{b}_y\right) \tag{12}$$

where $\mathbf{W}_y$ and $\mathbf{b}_y$ are parameter of the network to be learned.

4.4 Model Training

The proposed model can be stated as a parameter tuple $\theta = (\mathbf{W}, \mathbf{b})$. To compute the model parameters θ, we define the training objective for a data sample as:

$$L(\theta) = -\sum_{i=0}^{K} \mathbf{y}_i \log \hat{\mathbf{y}}_i + \lambda \left\|\theta\right\|^2 \qquad (13)$$

where $\mathbf{y} \in \{0,1\}^{(K+1)}$ indicating the one-hot vector represented ground truth; and λ is a regularization coefficient. By minimizing $L(\theta)$ using mini-batch gradient descent (GD) with Adam optimizer (Kingma and Ba, 2014), θ is updated through neural network structures.

4.5 Additional techniques

For this paper, we directly utilize the pre-trained fastText word embeddings model (Bojanowski et al., 2017) which is trained on Wikipedia data. The look-up tables for dependency embeddings, word characters, POS tags are randomly constructed using the Glorot initializer (Glorot and Bengio, 2010) and are treated as the parameters to be learned during the training phase.

Since the CNN model takes the fixed size matrix as input, we pad the inputs in each batch of data dynamically to the longest input length of the batch. We further use the batch normalization (Ioffe and Szegedy, 2015) which is able to enable higher learning rates and reduces over-fitting.

During the training phase, we make use of several techniques, including: *clipping the gradients* if their norm exceeds a given threshold (Goldberg, 2017); applying *dropout* (Srivastava et al., 2014) with the probability of 0.5 on embeddings layer, CNN hidden states, and penultimate layer; and using *early stopping* (Caruana et al., 2001) by validation loss.

Further, to reduce the impact of random effects on our model, we employ the ensemble mechanism (Krogh and Sollich, 1997). For this study, we run the model for 20 times and uses the strict majority vote to obtain the final results.

5 Experimental Evaluation

5.1 Dataset

Our model was evaluated on SemEval-2010 Task 8 dataset (Hendrickx et al., 2009), which contains $10,717$ annotated relation classification examples and is separated into two subsets: $8,000$ instances for training and $2,717$ for testing. We randomly split 10 percents of the training data for validation. There are 9 directed relations and one undirected `Other` class.

We conduct the training-testing process 20 times and calculate the averaged results. For evaluation, the predicted labels were compared to the golden annotated data using standard precision (P), recall (R), and F1 score metrics.

5.2 Performance of the RbSP Model

Table 1 summarizes the performance of our model and comparative models. For a fair comparison with other researches, we implemented a baseline model, in which we remove all the proposed augmented information (multi-layer attention with kernel filters and LSTM on original sentence). This baseline model is similar to the model of Xu et al. (2015a) with some technical improvements and additional information sources. It yields higher F1 than competitors which are based on SDP without any data augmentation methods. This result is also comparative when is placed next to the result of basic Attention-CNN model.

The results also demonstrate the effectiveness of our proposed methods that brings an improvement of 1.5% in F1, compared to the baseline result. Our RbSP model yields an F1-score of 86.3%, outperforms other comparative models, except Multi-Att-CNN model of Wang et al. (2016) with multi-level attention CNN. However, we have tried to re-implement the Multi-Att-CNN, but we failed to reproduce the positive result in the original paper. The performance of our re-implementation is about 84.9% of F1. This result has a high consensus with Luo et al. (2018) since they also tried to re-build this model, and their re-implemented result is not much different from us, as 85.5%.

It is worth to note that when comparing with another augmented method of Liu et al. (2015), our multi-layer attention with kernel filters architecture brings more significant improvement. Relatively, in comparison of efficiency of augmented methods on the baseline model, the full-tree augmentation only brings 1% improvement of F1 while our attentive augmentation boosts up to 1.5%. Unlike the method of using the whole subtree to supplement information for the target node, our method only uses the most relevant nodes that are direct children to represent augmented infor-

Model	Source of information	F1
depLCNN (Xu et al., 2015a)	Word embeddings, SDP, CNN	81.9
	+ WordNet, word around nominals	83.7
	+ Negative sampling	85.6
BRCNN (Cai et al., 2016)	Word embeddings, SDP, LSTM, CNN	85.4
	+ POS, NER, WordNet embeddings, inverse SDP	86.3
DepNN (Liu et al., 2015)	200-d Gigaword embeddings, SDP, CNN	81.8
	+ Augmented sub-tree, Recursive Neural Network	82.8
	+ NER	83.6
Attention-CNN (Shen and Huang, 2016)	Sentence convolution, Attention-based context	84.3
	+ WordNet, Words around nominals	85.9
AT-BLSTM (Luo et al., 2018)	Word embeddings, Sentence attention features, Tensor feature	86.3
Multi-Att-CNN (Wang et al., 2016)	Multi-Level Attention CNNs, Attention pooling	88.0[†] 85.5[‡]
Baseline	Word embeddings, POS tag, WordNet	84.8
RbSP (our model)	Baseline + Augmented Information	86.3
	+ ensemble	**86.7**

Table 1: The comparison of our RbSP model with other comparative models on SemEval-2010 task 8 dataset. The reported results are macro-averaged F1 scores of (9+1)-way evaluation with directionality taken into account. Since the comparative models did not report the precision (P) and recall (R), we also report the F1 score only. [†]: We failed to reproduce good result with the Multi-Att-CNN model, the performance of our implementation is just about 84.9. [‡]: Another re-implemented result of Multi-Att-CNN model reported by Luo et al. (2018).

mation. In addition, our method further focuses on the most important children through two attention layers.

We also observe that during many training-testing processes, the results may vary. The standard deviation of 20 runs is about 0.27. We perform the ensemble strategy by majority voting on the results of 20 runs, and it drives our model to achieve a better result of 86.7%. This result is outperformed other comparative models.

5.3 Contribution of different components

Figure 4 shows the changes in F1 when removing each proposed component from the RbSP model. The F1 reductions illustrate the contributions of all proposals to the final result. However, the impact levels vary with different components. Between two proposed component, the multi-layer attention with kernel filters (augmented information) plays a vital role when contributing 1.22% to the final performance while the contribution of the LSTM on the original sentence is 0.33%.

An interesting observation comes from the interior of the multi-layer attention with kernel filters. The impact of removing the whole augmented information is much higher than the total impact of removing multi-layer attention or kernel filters (1.22 vs. 0.42+0.18 = 0.6). These results demonstrate that the combination of constituent parts is thoroughly utilized by our sequential augmented architecture.

Another experiment is on investigating the meaning of each attention component. The result lightly reduces when we remove the self-attention or heuristic attention component. The results also prove that our proposed heuristic attention method is simple but effective. Its improvement is equivalent to the self-attention which is a complex attention mechanism. Among the input of multi-layer attention, the word embedding has a great influence on the model performance. However, children POS tag and relation to parent are also essential components to have the good results.

5.4 Results Analysis

We studied model outputs to analyze system errors in the cases of using the baseline model and using the proposed model with RbSP representation.

In Figure 5, we considered four types of errors: If the model makes a wrong decision and labels an `Other` relation (negative) as an actual relation (positive), it indicates 1 `FP` (False Positive) error.

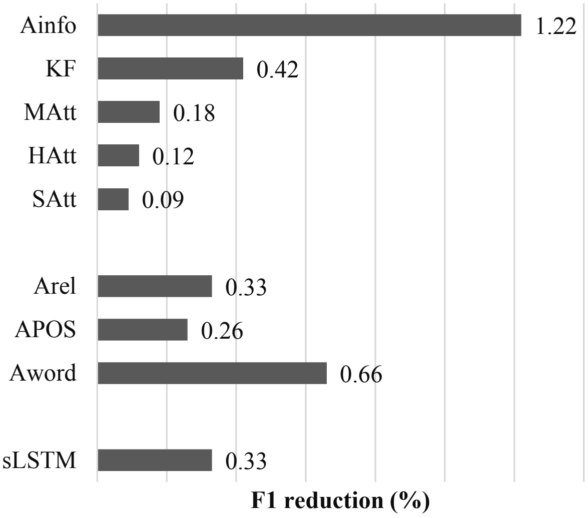


Figure 4: Comparing the contribution of proposed components by removing these components from the model: self-attention (SAtt), heuristic attention (HAtt), multi-layer attention (MAtt), kernel filers (KF), augmented information (Ainfo), augmentation using word embedding (Aword), augmentation using POS tag (APOS), augmentation using dependency relation (Arel), and LSTM on original sentence (sLSTM). F1 reduction is calculated by the average result of 20 runs.

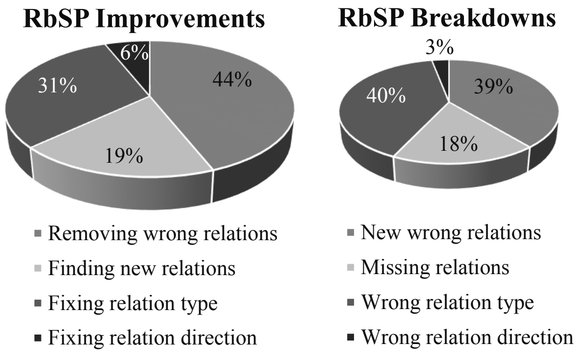


Figure 5: Comparing the effects of using RbSP in two aspects, (i) RbSP improved performance and (ii) RbSP yielded some additional wrong results. Four types of errors are analyzed, note that actual relations are considered as positive relations while `Other` is considered as negative: Labelling an `Other` relation as an actual relation; labelling an actual relation as `Other`; Confusion between types of relations; Direction errors.

Vice versa, if it labels an actual relation as `Other`, it brings 1 `FN` (False Negative). In the case that model confused between two types of relations, the model will be penalized twice, with 1 `FP` and 1 `FN`. Direction error, i.e., the model predicts the relation correctly but its direction wrongly, also brings 1 `FP` and 1 `FN`. The proportions of the left and the right of Figure 5 are quite consistent. In which, RbSP seems to have the most impact on determining whether an instance is positive or negative. RbSP also changes the decision of the relation type in quite many cases. It also influences the decision-making about relation's directionality, but not much.

Totally, the use of RbSP helps to correct more than 150 errors of the baseline model. However, it also yields some new errors (about 70 errors). Therefore, the difference of $F1$ between the baseline model and our RbSP model is only 1.5%, as stated in table 1.

Table 2 gives some realistic examples of different results when using the RbSP and not. We observed that the baseline model seems to be stuck in over-fitting problem, for examples, it classified all SDP with *prep:with* as `Instrument-Agency` and all SDP with *prep:in* as `Member-Collection` (examples $1-2$). RbSP is really useful for solving these cases partly since it uses attentive augmentation information to distinguish the same SDP or the same preposition with different meanings. RbSP is also proven to be stronger in examples $3-4$ to find new results and examples $5-7$ to fix wrong results. In our statistic, the use of RbSP bring the big advantage for the relations `Component-Whole`, `Message-Topic`, `Entity-Destination`, `Product-Producer` and `Instrument-Agency`. The results are almost constant for `Member-Collection` relations. Vice versa, we regret to state that using RbSb brings some worse results (examples $8-11$), especially for `Cause-Effect` and `Content-Container` relations.

Many errors seem attributable to the parser or our model's limitations that still cannot be overcome by using the RbSP (Examples $12-13$). We listed here some highlight problems to prioritize future researches (a) information on the SDP and its child nodes is still insufficient or redundant to make the correct prediction, (b) the direction of relations is still challenging since some errors appeared because we predict the relation correctly but its direction wrongly (c) the over-fitting problem (leading to wrong prediction - `FP`) and (d) lacking in generality (cannot predict new relation - `FN`).

#	SID[†]	SDP	Label[*] Golden	RbSP	Baseline
1	8652	**Heating** *prep:with* **wood**	**Other**	**Other**	IA-21
2	10402	**officer** *prep:of* **college**	**Other**	**Other**	MC-12
3	9728	**news** *acl* **crashed** *nsubj* **plane**	**MT-12**	**MT-12**	Other
4	8421	**lane** *prep:on* **road**	**CW-12**	**CW-12**	Other
5	9092	**hurts** *prep:from* **memories**	**EO-12**	**EO-12**	CE-21
6	8081	**bar** *prep:of* **seats**	**CW-12**	**CW-12**	MC-21
7	10457	**show** *nsubj* **offers** *dobj* **discussion**	**MT-12**	**MT-12**	MT-21
8	10567	**stand** *prep:against* **violence**	**Other**	MT-12	**Other**
9	10296	**fear** *prep:from* **robbers**	**CE-21**	Other	**CE-21**
10	9496	**casket** *nsubjpass* **placed** *prep:inside* **casket**	**CC-12**	ED-12	**CC-12**
11	9734	**documents** *acl* **discussed** *prep:at* **meeting**	**MT-21**	MT-12	**MT-21**
12	9692	**rhyme** *prep:by* **thing**	**PP-12**	Other	Other
13	10562	**profits** *prep:from* **inflation**	**Other**	CE-21	CE-21

Table 2: The examples of error from RbSP and Baseline models. The predicted labels are from the best runs. [†]SIDs are sentence IDs in the testing dataset. [*]Abbreviation of relations: CC (Content-Container), CE (Cause-Effect), CW (Component-Whole), ED (Entity-Destination), EO (Entity-Origin), IA (Instrument-Agency), MC (Member-Collection), MT (Message-Topic), PP (Product-Producer). [*]Abbreviation of relation directions: 12 (e1,e2), 21 (e2,e1).

6 Conclusions

In this paper, we have presented RbSP, a novel representation of relation between two nominals in a sentence that overcomes the disadvantages of traditional SDP. Our RbSP is created by using multi-layer attention to choose relevant information to augment a token in SDP from its child nodes. We also improved the attention mechanisms with kernel filters to capture the features on the context vector. We evaluated our model on SemEval-2010 task 8 dataset, then compared the results with very recent state-of-the-art models. Experiments were also constructed to verify the rationality and effectiveness of each of the model's components and information sources. The results demonstrated the advantage and robustness of our model, includes the LSTM on the original sentence, combination of self-attention and heuristic mechanisms and several augmentation inputs as well. The analysis of the results still points our some weaknesses of the model. We aim to address them and further extensions of our model in future works. We released our source code and data on the public repository to support the re-producibility of our work and facilitate other related studies.

Acknowledgments

This research was supported by an EPSRC Experienced Researcher Fellowship (N. Collier: EP/M005089/1). We also thank the anonymous reviewers for their comments and suggestions.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 592–596.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations.*

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 756–765.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160. Association for Computational Linguistics.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 129–136. Association for Computational Linguistics.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2006. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 448–456. JMLR. org.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In *6th Workshop on Automated Knowledge Base Construction (AKBC) at NIPS 2017*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Anders Krogh and Peter Sollich. 1997. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811.

Hoang-Quynh Le, Duy-Cat Can, Sinh T Vu, Thanh Hai Dang, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Large-scale exploration of neural relation classification architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2266–2277.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lishuang Li, Jieqiong Zheng, and Jia Wan. 2017. Dynamic extended tree conditioned lstm-based biomedical event extraction. *International Journal of Data Mining and Bioinformatics*, 17(3):266–278.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and WANG Houfeng. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 285–290.

Xiong Luo, Wenwen Zhou, Weiping Wang, Yueqin Zhu, and Jing Deng. 2018. Attention-based relation extraction with bidirectional gated recurrent unit and highway network in the analysis of geological data. *IEEE Access*, 6:5705–5715.

Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep learning with minimal training data: Turkunlp entry in the bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 73–81.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 68–74.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. *NAACL HLT 2015*, pages 39–48.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Nagesh C. Panyam, Karin Verspoor, Trevor Cohn, and Kotagiri Ramamohanarao. 2018. Exploiting graph kernels for high performance biomedical relation extraction. *Journal of biomedical semantics*, 9(1):7.

Changqin Quan, Meng Wang, and Fuji Ren. 2014. An unsupervised text mining method for relation extraction from biomedical literature. *PloS one*, 9(7):e102039.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yu Su, Honglei Liu, Semih Yavuz, Izzeddin Gur, Huan Sun, and Xifeng Yan. 2018. Global relation embedding for relation extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 820–830.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1298–1307.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1785–1794.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1021–1029. Association for Computational Linguistics.

Xiaobin Zhang, Fucai Chen, and Ruiyang Huang. 2018a. A combination of rnn and cnn for attention-based relation classification. *Procedia computer science*, 131:911–917.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases

Yu Chen
Rensselaer Polytechnic Institute
cheny39@rpi.edu

Lingfei Wu
IBM Research
wuli@us.ibm.com

Mohammed J. Zaki
Rensselaer Polytechnic Institute
zaki@cs.rpi.edu

Abstract

When answering natural language questions over knowledge bases (KBs), different question components and KB aspects play different roles. However, most existing embedding-based methods for knowledge base question answering (KBQA) ignore the subtle inter-relationships between the question and the KB (e.g., entity types, relation paths and context). In this work, we propose to directly model the two-way flow of interactions between the questions and the KB via a novel Bidirectional Attentive Memory Network, called BAMnet. Requiring no external resources and only very few hand-crafted features, on the WebQuestions benchmark, our method significantly outperforms existing information-retrieval based methods, and remains competitive with (hand-crafted) semantic parsing based methods. Also, since we use attention mechanisms, our method offers better interpretability compared to other baselines.

1 Introduction

With the rapid growth in large-scale knowledge bases (KBs) such as DBPedia (Auer et al., 2007) and FreeBase (Google, 2018), knowledge base question answering (KBQA) has drawn increasing attention over the past few years. *Given questions in natural language (NL), the goal of KBQA is to automatically find answers from the underlying KB*, which provides a more natural and intuitive way to access the vast underlying knowledge resources.

One of the most prominent challenges of KBQA is the lexical gap. For instance, the same question can be expressed in various ways in NL while a KB usually has a canonical lexicon. It is therefore nontrivial to map an NL question to a structured KB. The approaches proposed to tackle the KBQA task can be roughly categorized into two groups: semantic parsing (SP) and information retrieval (IR) approaches. SP-based approaches address the problem by constructing a semantic parser that converts NL questions into intermediate logic forms, which can be executed against a KB. Traditional semantic parsers (Wong and Mooney, 2007) require annotated logical forms as supervision, and are limited to narrow domains with a small number of logical predicates. Recent efforts overcome these limitations via the construction of hand-crafted rules or features (Abujabal et al., 2017; Hu et al., 2018) schema matching (Cai and Yates, 2013), and using weak supervision from external resources (Krishnamurthy and Mitchell, 2012).

Unlike SP-based approaches that usually assume a pre-defined set of lexical triggers or rules, which limit their domains and scalability, IR-based approaches directly retrieve answers from the KB in light of the information conveyed in the questions. These IR-based approaches usually do not require hand-made rules and can therefore scale better to large and complex KBs. Recently, deep neural networks have been shown to produce strong results on many NLP tasks. In the field of KBQA, under the umbrella of IR-based approaches, many embedding-based methods (Bordes et al., 2014b; Hao et al., 2017) have been proposed and have shown promising results. These methods adopt various ways to encode questions and KB subgraphs into a common embedding space and directly match them in that space, and can be typically trained in an end-to-end manner.

Compared to existing embedding-based methods that encode questions and KB subgraphs independently, we introduce a novel **B**idirectional **A**ttentive **M**emory **net**work, called **BAMnet** that captures the mutual interactions between questions and the underlying KB, which is stored in a content-addressable memory. We assume that the world knowledge (i.e., the KB) is helpful for better understanding the questions. Similarly, the questions

Proceedings of NAACL-HLT 2019, pages 2913–2923
Minneapolis, Minnesota, June 2 - June 7, 2019.

themselves can help us focus on important KB aspects. To this end, we design a *two-layered bidirectional attention network*. The *primary attention network* is intended to focus on important parts of a question in light of the KB and important KB aspects in light of the question. Built on top of that, the *secondary attention network* is intended to enhance the question and KB representations by further exploiting the two-way attention. Through this idea of *hierarchical two-way attention*, we are able to distill the information that is the most relevant to answering the questions on both sides of the question and KB.

We highlight the contributions of this paper as follows: 1) we propose a novel bidirectional attentive memory network for the task of KBQA which is intended to directly model the two-way interactions between questions and the KB; 2) by design, our method offers good interpretability thanks to the attention mechanisms; 3) on the WebQuestions benchmark, our method significantly outperforms previous information-retrieval based methods while remaining competitive with (hand-crafted) semantic parsing based methods.

2 Related work

Two broad classes of SP-based and IR-based approaches have been proposed for KBQA. The former attempts to convert NL questions to logic forms. Recent work focused on approaches based on weak supervision from either external resources (Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Yao and Van Durme, 2014; Hu et al., 2018; Yih et al., 2015; Yavuz et al., 2016), schema matching (Cai and Yates, 2013), or using hand-crafted rules and features (Unger et al., 2012; Berant et al., 2013; Berant and Liang, 2015; Reddy et al., 2016; Bao et al., 2016; Abujabal et al., 2017; Hu et al., 2018; Bast and Haussmann, 2015; Yih et al., 2015). A thread of research has been explored to generate semantic query graphs from NL questions such as using coarse alignment between phrases and predicates (Berant et al., 2013), searching partial logical forms via an agenda-based strategy (Berant and Liang, 2015), pushing down the disambiguation step into the query evaluation stage (Hu et al., 2018), or exploiting rich syntactic information in NL questions (Xu et al., 2018a,b). Notably, another thread of SP-based approaches try to exploit IR-based techniques (Yao and Van Durme, 2014; Bast and Haussmann, 2015; Yang et al., 2014; Yih et al., 2015; Bao et al., 2016; Yavuz et al., 2016; Liang et al., 2016) by computing the similarity of two sequences as features, leveraging a neural network-based answer type prediction model, or training end-to-end neural symbolic machine via REINFORCE (Williams, 1992). However, most SP-based approaches more or less rely on handcrafted rules or features, which limits their scalability and transferability.

The other line of work (the IR-based) has focused on mapping answers and questions into the same embedding space, where one could query any KB independent of its schema without requiring any grammar or lexicon. Bordes et al. (2014b) were the first to apply an embedding-based approach for KBQA. Later, Bordes et al. (2014a) proposed the idea of subgraph embedding, which encodes more information (e.g., answer path and context) about the candidate answer. In follow-up work (Bordes et al., 2015; Jain, 2016), memory networks (Weston et al., 2014) were used to store candidates, and could be accessed iteratively to mimic multi-hop reasoning. Unlike the above methods that mainly use a bag-of-words (BOW) representation to encode questions and KB resources, (Dong et al., 2015; Hao et al., 2017) apply more advanced network modules (e.g., CNNs and LSTMs) to encode questions. Hybrid methods have also been proposed (Feng et al., 2016; Xu et al., 2016; Das et al., 2017), which achieve improved results by leveraging additional knowledge sources such as free text. While most embedding-based approaches encode questions and answers independently, (Hao et al., 2017) proposed a cross-attention mechanism to encode questions according to various candidate answer aspects. Differently, in this work, our method goes one step further by modeling the bidirectional interactions between questions and a KB.

The idea of bidirectional attention proposed in this work is similar to those applied in machine reading comprehension (Wang and Jiang, 2016; Seo et al., 2016; Xiong et al., 2016). However, these previous works focus on capturing the interactions between two bodies of text, in this work, we focus on modeling the interactions between one body of text and a KB.

3 A modular bidirectional attentive memory network for KBQA

Given an NL question, the goal is to fetch answers from the underlying KB. Our proposed BAMnet

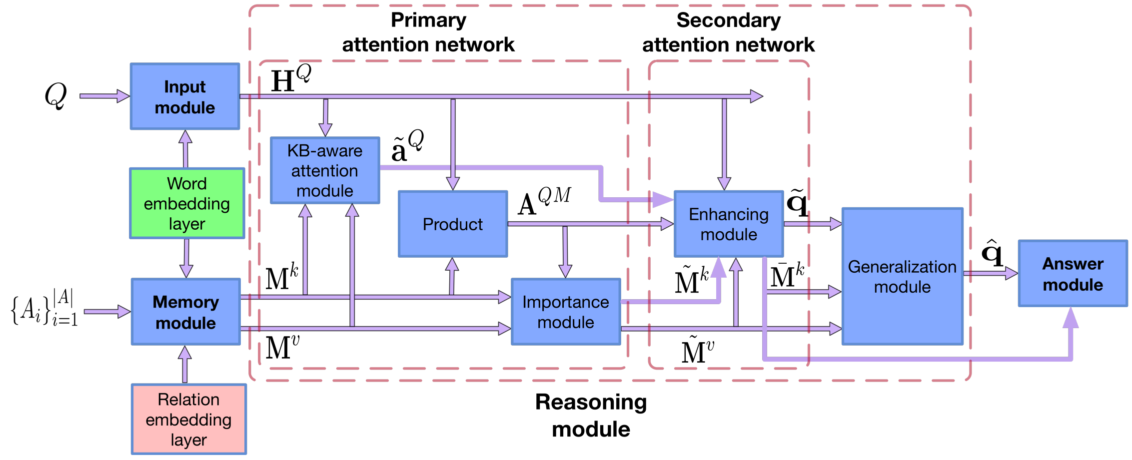


Figure 1: Overall architecture of the BAMnet model.

model consists of four components which are the *input module*, *memory module*, *reasoning module* and *answer module*, as shown in Fig. 1.

3.1 Input module

An input NL question $Q = \{q_i\}_{i=1}^{|Q|}$ is represented as a sequence of word embeddings (q_i) by applying a word embedding layer. We then use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the question as $\mathbf{H}^Q$ (in $\mathbb{R}^{d\times|Q|}$) which is the sequence of hidden states (i.e., the concatenation of forward and backward hidden states) generated by the BiLSTM.

3.2 Memory module

Candidate generation Even though all the entities from the KB could in principle be candidate answers, this is computationally expensive and unnecessary in practice. We only consider those entities which are "close" to the main topic entity of a question. An answer is the text description (e.g., a name) of an entity node. For example, *Ohio* is the topic entity of the question "Who was the secretary of state of Ohio in 2011?" (see Fig. 2). After getting the topic entity, we collect all the entities connected to it within h hops as candidate answers, which we denote as $\{A_i\}_{i=1}^{|A|}$.

KB representation For each candidate answer from the KB, we encode three types of information: answer type, path and context.

Answer type Entity type information is an important clue in ranking answers. For example, if a question uses the interrogative word *where*, then candidate answers with types relevant to the concept of location are more likely to be correct. We use a BiLSTM to encode its text description to get a d-dimensional vector $\mathbf{H}_i^{t_1}$ (i.e., the concatenation of last forward and backward hidden states).

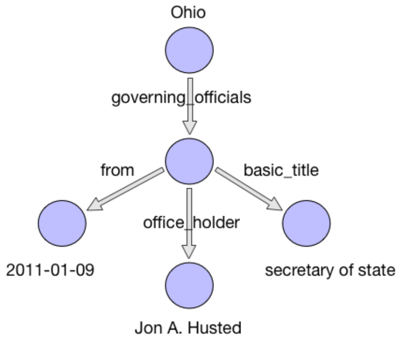


Figure 2: A working example from Freebase. Relations in Freebase have hierarchies where high-level ones provide too broad or even noisy information about the relation. Thus, we choose to use the lowest level one.

Answer path We define an answer path as a sequence of relations from a candidate answer to a topic entity. For example, for the *Ohio* question (see Fig. 2), the answer path of *Jon A. Husted* can be either represented as a sequence of relation ids [*office_holder, governing_officials*] or the text description [*office, holder, governing, officials*]. We thus encode an answer path as $\mathbf{H}_i^{p_1}$ via a BiLSTM, and as $\mathbf{H}_i^{p_2}$ by computing the average of its relation embeddings via a relation embedding layer.

Answer context The answer context is defined as the surrounding entities (e.g., sibling nodes) of a candidate which can help answer questions with constraints. For example, in Fig. 2, the answer context of *Jon A. Husted* includes the government position title *secretary of state* and starting date *2011-01-09*. However, for simple questions without constraints, the answer context is unnecessary and can potentially incorporate noise. We tackle this issue with two strategies: 1) we use a novel *importance module* (explained later) to focus on important answer aspects, and 2) we only consider those context nodes that have overlap with the question. Specifically, for each context node (i.e., a sequence of words) of a candidate, we first compute the longest common subsequence between it and the question, we then encode it via a BiLSTM only if we get a non-stopwords substring. Finally, the answer context of a candidate answer will be encoded as the average of all context node representations, which we denote as $\mathbf{H}_i^c$.

Key-value memory module In our model, we use a key-value memory network (Miller et al., 2016) to store candidate answers. Unlike a basic memory network (Weston et al., 2014), its addressing stage is based on the key memory while the reading stage uses the value memory, which gives greater flexibility to encode prior knowledge via functionality separation. Thus, after encoding the answer type, path and context, we apply linear projections on them as follows:

$$\begin{aligned} \mathbf{M}_i^{k_t} &= f_t^k(\mathbf{H}_i^{t_1}) & \mathbf{M}_i^{v_t} &= f_t^v(\mathbf{H}_i^{t_1}) \\ \mathbf{M}_i^{k_p} &= f_p^k([\mathbf{H}_i^{p_1};\mathbf{H}_i^{p_2}]) & \mathbf{M}_i^{v_p} &= f_p^v([\mathbf{H}_i^{p_1};\mathbf{H}_i^{p_2}]) \\ \mathbf{M}_i^{k_c} &= f_c^k(\mathbf{H}_i^{c}) & \mathbf{M}_i^{v_c} &= f_c^v(\mathbf{H}_i^{c}) \end{aligned} \tag{1}$$

where $\mathbf{M}_i^{k_t}$ and $\mathbf{M}_i^{v_t}$ are d-dimensional key and value representations of answer type A_i^t, respectively. Similarly, we have key and value representations for answer path and answer context. We denote $\mathbf{M}$ as a key-value memory whose row $\mathbf{M}_i = \{\mathbf{M}_i^k, \mathbf{M}_i^v\}$ (both in $\mathbb{R}^{d\times 3}$), where $\mathbf{M}_i^k = [\mathbf{M}_i^{k_t};\mathbf{M}_i^{k_p};\mathbf{M}_i^{k_c}]$ comprises the keys, and $\mathbf{M}_i^v = [\mathbf{M}_i^{v_t};\mathbf{M}_i^{v_p};\mathbf{M}_i^{v_c}]$ comprises the values. Here $[,]$ and $[;]$ denote row-wise and column-wise concatenations, respectively.

3.3 Reasoning module

The reasoning module consists of a generalization module, and our novel *two-layered bidirectional attention network* which aims at capturing the two-way interactions between questions and the KB. The *primary attention network* contains the KB-aware attention module which focuses on the important parts of a question in light of the KB, and the importance module which focuses on the important KB aspects in light of the question. The *secondary attention network* (*enhancing module* in Fig. 1) is intended to enhance the question and KB vectors by further exploiting the two-way attention.

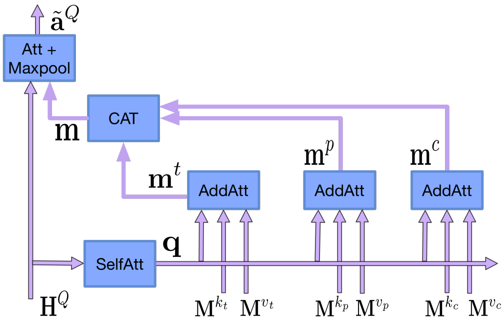


Figure 3: KB-aware attention module. CAT: concatenation, SelfAtt: self-attention, AddAtt: additive attention.

KB-aware attention module Not all words in a question are created equal. We use a KB-aware attention mechanism to focus on important components of a question, as shown in Fig. 3. Specifically, we first apply self-attention (SelfAtt) over all question word vectors $\mathbf{H}^Q$ to get a d-dimensional question vector $\mathbf{q}$ as follows

$$\begin{aligned} \mathbf{q} &= \text{BiLSTM}([\mathbf{H}^Q \mathbf{A}^{QQ^T}, \mathbf{H}^Q]) \\ \mathbf{A}^{QQ} &= \text{softmax}((\mathbf{H}^Q)^T \mathbf{H}^Q) \end{aligned} \tag{2}$$

where softmax is applied over the last dimension of an input tensor by default. Using question summary $\mathbf{q}$, we apply another attention (AddAtt) over the memory to obtain answer type $\mathbf{m}_t$, path $\mathbf{m}_p$ and context summary $\mathbf{m}_c$:

$$\begin{aligned} \mathbf{m}_x &= \sum_{i=1}^{|A|} a_i^x \cdot \mathbf{M}_i^{v_x} \\ \mathbf{a}^x &= \text{Att}_{\text{add}}(\mathbf{q}, \mathbf{M}^{k_x}) \end{aligned} \tag{3}$$

where $x \in \{t, p, c\}$, and $\text{Att}_{\text{add}}(\mathbf{x}, \mathbf{y}) = \text{softmax}(\tanh([\mathbf{x}^T, \mathbf{y}]\mathbf{W_1})\mathbf{W_2})$, with $\mathbf{W_1} \in \mathbb{R}^{2d\times d}$ and $\mathbf{W_2} \in \mathbb{R}^{d\times 1}$ being trainable weights.

So far, we have obtained the KB summary $\mathbf{m} = [\mathbf{m}_t;\mathbf{m}_p;\mathbf{m}_c]$ in light of the question. We proceed to compute the question-to-KB attention between question word q_i and KB aspects as formulated by $\mathbf{A}^{Qm} = \mathbf{H}^{Q^T}\mathbf{m}$. By applying max pooling over the last dimension (i.e., the KB aspect dimension) of $\mathbf{A}^{Qm}$, that is, $\mathbf{a}_i^Q = \max_j \mathbf{A}_{ij}^{Qm}$, we select the strongest connection between q_i and the KB. The idea behind it is that each word in a question serves a specific purpose (i.e., indicating answer type, path or context), and max pooling can help find out that purpose. We then apply a softmax over the resulting vector to obtain $\tilde{\mathbf{a}}^Q$ which is a *KB-aware question attention vector* since it indicates the importance of q_i in light of the KB.

Importance module The importance module focuses on important KB aspects as measured by their relevance to the questions. We start by computing a $|Q| \times |A| \times 3$ attention tensor $\mathbf{A}^{QM}$ which indicates the strength of connection between each pair of $\{q_i, A_j^x\}^{x=\{t,p,c\}}$. Then, we take the max of the question word dimension of $\mathbf{A}^{QM}$ and normalize it to get an attention matrix $\tilde{\mathbf{A}}^M$, which indicates the importance of each answer aspect for each candidate answer. After that, we proceed to compute question-aware memory representations $\tilde{\mathbf{M}}^k$. Thus, we have:

$$\begin{aligned}
\tilde{\mathbf{M}}^v &= \{\tilde{\mathbf{M}}_i^v\}_{i=1}^{|A|} \in \mathbb{R}^{|A|\times d} \quad & \tilde{\mathbf{M}}_i^v &= \sum_{j=1}^{3} \mathbf{M}_{ij}^v \\
\tilde{\mathbf{M}}^k &= \{\tilde{\mathbf{M}}_i^k\}_{i=1}^{|A|} \in \mathbb{R}^{|A|\times d} \quad & \tilde{\mathbf{M}}_i^k &= \tilde{\mathbf{A}}_i^M \mathbf{M}_i^k \\
\tilde{\mathbf{A}}^M &= \text{softmax}(\mathbf{A}^{M^T})^T \quad & \mathbf{A}^M &= \max_i \{\mathbf{A}_i^{QM}\}_{i=1}^{|Q|} \\
\mathbf{A}^{QM} &= \left(\mathbf{M}^k \mathbf{H}^Q\right)^T
\end{aligned} \tag{4}$$

Enhancing module We further enhance the question and KB representations by exploiting two-way attention. We compute the KB-enhanced question representation $\tilde{\mathbf{q}}$ which incorporates the relevant KB information by applying max pooling over the last dimension (i.e., the answer aspect dimension) of $\mathbf{A}^{QM}$, that is, $\mathbf{A}_M^Q = \max_k\{\mathbf{A}_{.,.,k}^{QM}\}_{k=1}^3$, and then normalizing it to get a question-to-KB attention matrix $\tilde{\mathbf{A}}_M^Q$ from which we compute the question-aware KB summary and incorporate it into the question representation $\tilde{\mathbf{H}}^Q = \mathbf{H}^Q + \tilde{\mathbf{a}}^Q \odot (\tilde{\mathbf{A}}_M^Q \tilde{\mathbf{M}}^v)^T$. Finally, we obtain a d-dimensional KB-enhanced question representation $\tilde{\mathbf{q}} = \tilde{\mathbf{H}}^Q \tilde{\mathbf{a}}^Q$.

Similarly, we compute a question-enhanced KB representation $\overline{\mathbf{M}}^k$ which incorporates the relevant question information:

$$\begin{aligned}
\overline{\mathbf{M}}^k &= \tilde{\mathbf{M}}^k + \tilde{\mathbf{a}}^M \odot (\tilde{\mathbf{A}}_Q^M (\tilde{\mathbf{H}}^Q)^T) \\
\tilde{\mathbf{a}}^M &= (\tilde{\mathbf{A}}_M^Q)^T \tilde{\mathbf{a}}^Q \in \mathbb{R}^{|A|\times 1} \\
\tilde{\mathbf{A}}_Q^M &= \text{softmax}(\mathbf{A}_M^{Q\,T}) \in \mathbb{R}^{|A|\times|Q|}
\end{aligned} \tag{5}$$

Generalization module We add a one-hop attention process before answering. We use the question representation $\tilde{\mathbf{q}}$ to query over the key memory $\overline{\mathbf{M}}^k$ via an attention mechanism, and fetch the most relevant information from the value memory, which is then used to update the question vector using a GRU (Cho et al., 2014). Finally, we apply a residual layer (He et al., 2016) (i.e., $y = f(x) + x$) and batch normalization (BN) (Ioffe and Szegedy, 2015), which help the model performance in practice. Thus, we have

$$\begin{aligned}
&\hat{\mathbf{q}} = \text{BN}(\tilde{\mathbf{q}} + \mathbf{q}') \quad \mathbf{q}' = \text{GRU}(\tilde{\mathbf{q}}, \tilde{\mathbf{m}}) \\
&\tilde{\mathbf{m}} = \sum_{i=1}^{|A|} a_i \cdot \tilde{\mathbf{M}}_i^v \quad \mathbf{a} = \text{Att}_{\text{add}}^{\text{GRU}}(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k)
\end{aligned} \tag{6}$$

3.4 Answer module

Given the representation of question Q which is $\hat{\mathbf{q}}$ and the representation of candidate answers $\{A_i\}_{i=1}^{|A|}$ which is $\{\overline{\mathbf{M}}_i^k\}_{i=1}^{|A|}$, we compute the matching score $S(\hat{\mathbf{q}}, \overline{\mathbf{M}}_i^k)$ between every pair (Q, A_i) as $S(\mathbf{q}, \mathbf{a}) = \mathbf{q}^T \cdot \mathbf{a}$. The candidate answers are then ranked by their scores.

3.5 Training and testing

Training Intermediate modules such as the *enhancing module* generate "premature" representations of questions (e.g., $\tilde{\mathbf{q}}$) and candidate answers (e.g., $\overline{\mathbf{M}}^k$). Even though these intermediate representations are not optimal for answer prediction, we can still use them along with the final representations to jointly train the model, which we find helps the training probably by providing more supervision since we are directly forcing intermediate representations to be helpful for prediction. Moreover, we directly match interrogative words to KB answer types. A question Q is represented by a 16-dimensional interrogative word (we use "which", "what", "who", "whose", "whom", "where", "when", "how", "why" and "whether") embedding $\mathbf{q}^w$ and a candidate answer A_i is represented by entity type embedding $\mathbf{H}_i^{t_2}$ with the same size. We then compute the matching score $S(\mathbf{q}^w, \mathbf{H}_i^{t_2})$ between them. Although we only have weak labels (e.g., incorrect answers do not necessarily imply incorrect types) for the type matching task, and there are no shared representations between two tasks, we find in practice this strategy helps the training process as shown in Section 4.4.
Loss Function: In the training phase, we force positive candidates to have higher scores than negative candidates by using a triplet-based loss function:

$$\begin{aligned}
o = g(\mathbf{H}^Q \tilde{\mathbf{a}}^Q, \sum_{j=1}^{3} \mathbf{M}_{.,j}^k) + g(\tilde{\mathbf{q}}, \overline{\mathbf{M}}^k) \\
+ g(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k) + g(\mathbf{q}^w, \mathbf{H}^{t_2})
\end{aligned} \tag{7}$$

where $g(\mathbf{q}, \mathbf{M}) = \sum_{\substack{a^+ \in A^+ \\ a^- \in A^-}} \ell(S(\mathbf{q}, \mathbf{M}_{a^+}), S(\mathbf{q}, \mathbf{M}_{a^-}))$, and $\ell(y, \hat{y}) = \max(0, 1 + \hat{y} - y)$ is a hinge loss function, and A^+ and A^- denote the positive (i.e., correct) and negative (i.e., incorrect) answer sets, respectively. Note that at training time, the candidate answers are extracted from the KB subgraph of the gold-standard topic entity, with the memory size set to N_{max}. We adopt the following sampling strategy which works well in practice: if N_{max} is larger than the number of positive answers $|A^+|$, we keep all the positive answers and randomly select negative answers to fill up the memory; otherwise, we randomly select $\min(N_{max}/2, |A^-|)$ negative answers and fill up the remaining memory with random positive answers.

Testing At testing time, we need to first find the topic entity. We do this by using the top result returned by a separately trained topic entity predictor (we also compare with the result returned by the Freebase Search API). Then, the *answer module* returns the candidate answer with the highest scores as predicted answers. Since there can be multiple answers to a given question, the candidates whose scores are close to the highest score within a certain margin, θ, are regarded as good answers as well. Therefore, we formulate the inference process as follows:

$$\hat{A} = \{\hat{a} \quad | \ \hat{a} \in A \ \& \ \max_{a' \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a'})\} - S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{\hat{a}}) < \theta\} \tag{8}$$

where $\max_{a' \in A}\{S(\hat{\mathbf{q}}, \overline{\mathbf{M}}^k_{a'})\}$ is the score of the best matched answer and $\hat{A}$ is the predicted answer set. Note that θ is a hyper-parameter which controls the degree of tolerance. Decreasing the value of θ makes the model become stricter when predicting answers.

3.6 Topic entity prediction

Given a question Q, the goal of a topic entity predictor is to find the best topic entity $\hat{c}$ from the candidate set $\{C_i\}_{i=1}^{|C|}$ returned by external topic entity linking tools (we use the Freebase Search API and S-MART (Yang and Chang, 2016) in our experiments). We use a convolutional network (CNN) to encode Q into a d-dimensional vector $\mathbf{e}$. For candidate topic entity C_i, we encode three types of KB aspects, namely, the entity name, entity type and surrounding relations where both entity name and type are represented as a sequence of words while surrounding relations are represented as a bag of sequences of words. Specifically, we use three CNNs to encode them into three d-dimensional vectors, namely, $\mathbf{C}_i^n$, $\mathbf{C}_i^t$ and $\mathbf{C}_i^{r_1}$. Note that for surrounding relations, we first encode each of the relations and then compute their average. Additionally, we compute an average of the relation embeddings via a relation embedding layer which we denote as $\mathbf{C}_i^{r_2}$. We then apply linear projections on the above vectors as follows:

$$\begin{aligned} \mathbf{P}_i^k &= f^k([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \\ \mathbf{P}_i^v &= f^v([\mathbf{C}_i^n; \mathbf{C}_i^t; \mathbf{C}_i^{r_1}; \mathbf{C}_i^{r_2}]) \end{aligned} \tag{9}$$

where $\mathbf{P}_i^k$ and $\mathbf{P}_i^v$ are d-dimensional key and value representations of candidate C_i, respectively. Furthermore, we compute the updated question vector $\hat{\mathbf{e}}$ using the generalization module mentioned earlier. Next, we use a dot product to compute the similarity score between Q and C_i. A triplet-based loss function is used as formulated by $o = g(\mathbf{e}, \mathbf{P}_i^k) + g(\hat{\mathbf{e}}, \mathbf{P}_i^k)$ where $g(.)$ is the aforementioned hinge loss function. When training the predictor, along with the candidates returned from external entity linking tools, we do negative sampling (using string matching) to get more supervision. In the testing phase, the candidate with the highest score is returned as the best topic entity and no negative sampling is applied.

4 Experiments

This section provides an extensive evaluation of our proposed BAMnet model against state-of-the-art KBQA methods. The implementation of BAMnet is available at `https://github.com/hugochan/BAMnet`.

4.1 Data and metrics

We use the Freebase KB and the WebQuestions dataset, described below:

Freebase This is a large-scale KB (Google, 2018) that consists of general facts organized as subject-property-object triples. It has 41M non-numeric entities, 19K properties, and 596M assertions.

WebQuestions This dataset (Berant et al., 2013) (`nlp.stanford.edu/software/sempre`) contains 3,778 training examples and 2,032 test examples. We further split the training instances into a training set and development set via a 80%/20% split. Approximately 85% of questions can be directly answered via a single FreeBase predicate. Also, each question can have multiple answers. In our experiments, we use a development version of the dataset (Baudis and Pichl, 2016), which additionally provides (potentially noisy) entity mentions for each question.

Following (Berant et al., 2013), macro F1 scores (i.e., the average of F1 scores over all questions) are reported on the WebQuestions test set.

4.2 Model settings

When constructing the vocabularies of words, entity types or relation types, we only consider those questions and their corresponding KB subgraphs appearing in the training and validation sets. The vocabulary size of words is $V = 100,797$. There are 1,712 entity types and 4,996 relation types in the KB subgraphs. Notably, in FreeBase, one entity might have multiple entity types. We only use the first one available, which is typically the most concrete one. For those non-entity nodes which are boolean values or numbers, we use "bool" or "num" as their types, respectively.

We also adopt a **query delexicalization** strategy where for each question, the topic entity mention as well as constraint entity mentions (i.e., those belonging to "date", "ordinal" or "number") are replaced with their types. When encoding KB context, if the overlap belongs to the above types, we also do this delexicalization, which will guarantee it matches up with the delexicalized question well in the embedding space.

Given a topic entity, we extract its 2-hop subgraph (i.e., $h = 2$) to collect candidate answers, which is sufficient for *WebQuestions*. At training time, the memory size is limited to $N_{max} = 96$ candidate answers (for the sake of efficiency). If there are more potential candidates, we do random sampling as mentioned earlier. We initialize word embeddings with pre-trained GloVe vectors (Pennington et al., 2014) with word embedding size $d_v = 300$. The relation embedding size d_p, entity type embedding size d_t and hidden size d are set as 128, 16 and 128, respectively. The dropout rates on the word embedding layer, question encoder side and the answer encoder side are 0.3, 0.3 and 0.2, respectively. The batch size is set as 32, and answer module threshold $\theta = 0.7$. As for the topic entity prediction, we use the same hyperparameters. For each question, there are 15 candidates after negative sampling in the training time. When encoding a question, we use a CNN with filter sizes 2 and 3. A linear projection is applied to merge features extracted with different filters. When encoding a candidate aspect, we use a CNN with filter size 3. Linear activation and max-pooling are used together with CNNs. In the training process, we use the Adam optimizer (Kingma and Ba, 2014) to train the model. The initial learning rate is set as 0.001 which is reduced by a factor of 10 if no improvement is observed on the validation set in 3 consecutive epochs. The training procedure stops if no improvement is observed on the validation set in 10 consecutive epochs. The hyper-parameters are tuned on the development set.

4.3 Performance comparison

As shown in Table 1, our method can achieve an F1 score of 0.557 when the gold topic entity is known, which gives an upper bound of our model performance. When the gold topic entity is unknown, we report the results using: 1) the Freebase Search API, which achieves a recall@1 score of 0.857 on the test set for topic entity linking, and 2) the topic entity predictor, which achieves a recall@1 score of 0.898 for entity retrieval.

As for the performance of BAMnet on WebQuestions, it achieves an F1 score of 0.518 using the topic entity predictor, which is significantly better than the F1 score of 0.497 using the Freebase Search API. We can observe that BAMnet significantly outperforms previous state-of-the-art IR-based methods, which conclusively demonstrates the effectiveness of modeling bidirectional interactions between questions and the KB.

It is important to note that unlike the state-of-the-art SP-based methods, BAMnet relies on no external resources and very few hand-crafted features, but still remains competitive with those approaches. Based on careful hand-drafted rules, some SP-based methods (Bao et al., 2016; Yih et al., 2015) can better model questions with constraints and aggregations. For example, (Yih et al., 2015) applies many manually designed rules and features to improve performance on questions with constraints and aggregations, and (Bao et al., 2016) directly models temporal (e.g., "after 2000"), ordinal (e.g., "first") and aggregation constraints (e.g., "how many") by adding detected constraint nodes to query graphs. In contrast, our method is end-to-end, with very few hand-crafted rules.

Additionally, (Yavuz et al., 2016; Bao et al., 2016) train their models on external Q&A datasets to get extra supervision. For a fairer comparison, we only show their results without training on external Q&A datasets. Similarly, for hyhrid systems (Feng et al., 2016; Xu et al., 2016), we only report results without using Wikipedia free text. It is in-

Methods (ref)	Macro F_1
SP-based	
(Berant et al., 2013)	0.357
(Yao and Van Durme, 2014)	0.443
(Wang et al., 2014)	0.453
(Bast and Haussmann, 2015)	0.494
(Berant and Liang, 2015)	0.497
(Yih et al., 2015)	**0.525**
(Reddy et al., 2016)	0.503
(Yavuz et al., 2016)	0.516
(Bao et al., 2016)	0.524
(Feng et al., 2016)	0.471
(Reddy et al., 2017)	0.495
(Abujabal et al., 2017)	0.510
(Hu et al., 2018)	0.496
IR-based	
(Bordes et al., 2014a)	0.392
(Yang et al., 2014)	0.413
(Dong et al., 2015)	0.408
(Bordes et al., 2015)	0.422
(Xu et al., 2016)	**0.471**
(Hao et al., 2017)	0.429
Our Method: BAMnet	
w/ gold topic entity	0.557
w/ Freebase Search API	0.497
w/ topic entity predictor	**0.518**

Table 1: Results on the WebQuestions test set. Bold: best in-category performance.

teresting to note that both (Yih et al., 2015) and (Bao et al., 2016) also use the ClueWeb dataset for learning more accurate semantics. The F1 score of (Yih et al., 2015) drops from 0.525 to 0.509 if ClueWeb information is removed. To summarize, BAMnet achieves state-of-the-art performance of 0.518 without recourse to any external resources and relies only on very few hand-crafted features. If we assume gold-topic entities are given then BAMnet achieves an F1 of 0.557.

4.4 Ablation study

Methods	Macro F_1
all	**0.557**
w/o two-layered bidirectional attn	0.534
w/o kb-aware attn (+self-attn)	0.544
w/o importance module	0.540
w/o enhancing module	0.550
w/o generalization module	0.542
w/o joint type matching	0.545
w/o topic entity delexicalization	0.529
w/o constraint delexicalization	0.554

Table 2: Ablation results on the WebQuestions test set. Gold topic entity is assumed to be known.

We now discuss the performance impact of the different modules and strategies in BAMnet. Note that gold topic entity is assumed to be known when we do this ablation study, because the error introduced by topic entity prediction might reduce the real performance impact of a module or strategy. As shown in Table 2, significant performance drops were observed after turning off some key attention modules, which confirms that the real power of our method comes from the idea of hierarchical two-way attention. As we can see, when turning off the *two-layered bidirectional attention network*, the model performance drops from 0.557 to 0.534. Among all submodules in the attention network, the *importance module* is the most significant since the F1 score drops to 0.540 without it, thereby confirming the effectiveness of modeling the query-to-KB attention flow. On the flip side, the importance of modeling the KB-to-query attention flow is confirmed by the fact that replacing the *KB-aware attention module* with self-attention significantly degrades the performance. Besides, the secondary attention layer, the *enhancing module*, also contributes to the overall model performance. Finally, we find that the topic entity delexicalization strategy has a big influence on the model performance while the constraint delexicalization strategy only marginally boosts the performance.

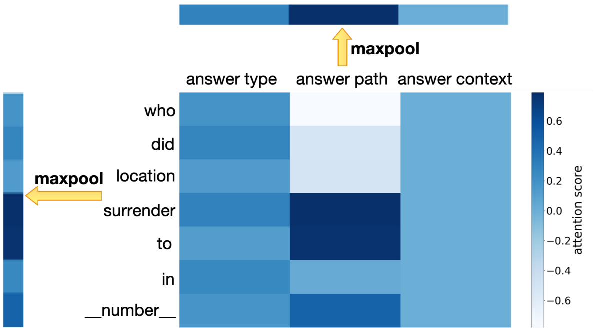


Figure 4: Attention heatmap generated by the reasoning module. Best viewed in color.

4.5 Interpretability analysis

Here, we show that our method does capture the mutual interactions between question words

KB Aspects	Questions	BAMnet w/o BiAttn.	BAMnet	Gold Answers
Answer Type	What degrees did Obama get in college?	Harvard Law School, Columbia University, Occidental College	Bachelor of Arts, Juris Doctor, Political Science	Juris Doctor, Bachelor of Arts
	What music period did Beethoven live in?	Austrian Empire, Germany, Bonn	Classical music, Opera	Opera, Classical music
Answer Path	Where did Queensland get its name from?	Australia	Queen Victoria	Queen Victoria
	Where does Delaware river start?	Delaware Bay	West Branch Delaware River, Mount Jefferson	West Branch Delaware River, Mount Jefferson
Answer Context	What are the major cities in Ukraine?	Kiev, Olyka, ... Vynohradiv, Husiatyn	Kiev	Kiev
	Who is running for vice president with Barack Obama 2012?	David Petraeus	Joe Biden	Joe Biden

Table 3: Predicted answers of BAMnet w/ and w/o bidirectional attention on the WebQuestions test set.

and KB aspects, by visualizing the attention matrix $\mathbf{A}^{QM}$ produced by the *reasoning module*. Fig. 4 shows the attention heatmap generated for a test question "who did location surrender to in __number__" (where "location" and "__number__" are entity types which replace the topic entity mention "France" and the constraint entity mention "ww2", respectively in the original question). As we can see, the attention network successfully detects the interactions between "who" and answer type, "surrender to" and answer path, and focuses more on those words when encoding the question.

To further examine the importance of the two-way flow of interactions, in Table 3, we show the predicted answers of BAMnet with and without the *two-layered bidirectional attention network* on samples questions from the WebQuestions test set. We divide the questions into three categories based on which kind of KB aspect is the most crucial for answering them. As we can see, compared to the simplified version which is not equipped with bidirectional attention, our model is more capable of answering all the three types of questions.

4.6 Error analysis

To better examine the limitations of our approach, we randomly sampled 100 questions on which our method performed poorly (i.e., with per-question F1 score less than 0.6), and categorized the errors. We found that around 33% of errors are due to label issues of gold answers and are not real mistakes. This includes incomplete and erroneous labels, and also alternative correct answers. Constraints are another source of errors (11%), with temporal constraints accounting for most. Some questions have implicit temporal (e.g., tense) constraints which our method does not model. A third source of error is what we term type errors (13%), for which our method generates more answers than needed because of poorly utilizing answer type information. Lexical gap is another source of errors (5%). Finally, other sources of errors (38%) include topic entity prediction error, question ambiguity, incomplete answers and other miscellaneous errors.

5 Conclusions and future work

We introduced a novel and effective bidirectional attentive memory network for the purpose of KBQA. To our best knowledge, we are the first to model the mutual interactions between questions and a KB, which allows us to distill the information that is the most relevant to answering the questions on both sides of the question and KB. Experimental results show that our method significantly outperforms previous IR-based methods while remaining competitive with hand-crafted SP-based methods. Both ablation study and interpretability analysis verify the effectiveness of the idea of modeling mutual interactions. In addition, our error analysis shows that our method actually performs better than what the evaluation metrics indicate.

In the future, we would like to explore effective ways of modeling more complex types of constraints (e.g., ordinal, comparison and aggregation).

Acknowledgments

This work is supported by IBM Research AI through the IBM AI Horizons Network. We thank the anonymous reviewers for their constructive suggestions.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*, pages 1191–1200.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *COLING*, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *CIKM*, pages 1431–1440. ACM.

Petr Baudis and Jan Pichl. 2016. Dataset factoid webquestions. https://github.com/brmson/dataset-factoid-webquestions.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *TACL*, 3:545–558.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *EMNLP*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *ECML-PKDD*, pages 165–180. Springer.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *ACL-IJCNLP*, volume 1, pages 260–269.

Yansong Feng, Songfang Huang, Dongyan Zhao, et al. 2016. Hybrid question answering over knowledge base and free text. In *COLING*, pages 2397–2407.

Google. 2018. Freebase data dumps. https://developers.google.com/freebase.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *ACL*, volume 1, pages 221–231.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *TKDE*, 30(5):824–837.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *NAACL-HLT*, pages 109–115.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *EMNLP-CoNLL*, pages 754–765.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *TACL*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *EMNLP*, pages 89–101.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*, pages 639–648. ACM.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. 2014. An overview of microsoft deep qa system on stanford webquestions benchmark. Technical Report MSR-TR-2014-121, Microsoft Research.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, pages 960–967.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *arXiv preprint arXiv:1603.00957*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. *arXiv preprint arXiv:1808.07624*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Yi Yang and Ming-Wei Chang. 2016. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. *arXiv preprint arXiv:1609.08075*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*, volume 1, pages 956–966.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *EMNLP*, pages 149–159.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, pages 1321–1331.

BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions

Christopher Clark[*][1], **Kenton Lee**[†], **Ming-Wei Chang**[†], **Tom Kwiatkowski**[†]

Michael Collins [†][2], **Kristina Toutanova**[†]

[*]Paul G. Allen School of CSE, University of Washington
`csquared@cs.uw.edu`

[†]Google AI Language
`{kentonl, mingweichang, tomkwiat, mjcollins, kristout}@google.com`

Abstract

In this paper we study yes/no questions that are *naturally occurring* — meaning that they are generated in unprompted and unconstrained settings. We build a reading comprehension dataset, BoolQ, of such questions, and show that they are unexpectedly challenging. They often query for complex, non-factoid information, and require difficult entailment-like inference to solve. We also explore the effectiveness of a range of transfer learning baselines. We find that transferring from entailment data is more effective than transferring from paraphrase or extractive QA data, and that it, surprisingly, continues to be very beneficial even when starting from massive pre-trained language models such as BERT. Our best method trains BERT on MultiNLI and then re-trains it on our train set. It achieves 80.4% accuracy compared to 90% accuracy of human annotators (and 62% majority-baseline), leaving a significant gap for future work.

Q: Has the UK been hit by a hurricane?
P: The Great Storm of 1987 was a violent extratropical cyclone which caused casualties in England, France and the Channel Islands . . .
A: Yes. [An example event is given.]

Q: Does France have a Prime Minister and a President?
P: . . . The extent to which those decisions lie with the Prime Minister or President depends upon . . .
A: Yes. [Both are mentioned, so it can be inferred both exist.]

Q: Have the San Jose Sharks won a Stanley Cup?
P: . . . The Sharks have advanced to the Stanley Cup finals once, losing to the Pittsburgh Penguins in 2016 . . .
A: No. [They were in the finals once, and lost.]

Figure 1: Example yes/no questions from the BoolQ dataset. Each example consists of a question (**Q**), an excerpt from a passage (**P**), and an answer (**A**) with an explanation added for clarity.

1 Introduction

Understanding what facts can be inferred to be true or false from text is an essential part of natural language understanding. In many cases, these inferences can go well beyond what is immediately stated in the text. For example, a simple sentence like “Hanna Huyskova won the gold medal for Belarus in freestyle skiing.” implies that (1) Belarus is a country, (2) Hanna Huyskova is an athlete, (3) Belarus won at least one Olympic event, (4) the USA did *not* win the freestyle skiing event, and so on.

To test a model’s ability to make these kinds of inferences, previous work in natural language inference (NLI) proposed the task of labeling candidate statements as being entailed or contradicted by a given passage. However, in practice, generating candidate statements that test for complex inferential abilities is challenging. For instance, evidence suggests (Gururangan et al., 2018; Jia and Liang, 2017; McCoy et al., 2019) that simply asking human annotators to write candidate statements will result in examples that typically only require surface-level reasoning.

In this paper we propose an alternative: we test models on their ability to answer naturally occurring yes/no questions. That is, questions that were authored by people who were not prompted to write particular kinds of questions, including even being required to write yes/no questions, and who did not know the answer to the question they were asking. Figure 1 contains some examples from our dataset. We find such questions often query for

---

[1] Work completed while interning at Google.

[2] Also affiliated with Columbia University, work done at Google.

Proceedings of NAACL-HLT 2019, pages 2924–2936
Minneapolis, Minnesota, June 2 - June 7, 2019.

non-factoid information, and that human annotators need to apply a wide range of inferential abilities when answering them. As a result, they can be used to construct highly inferential reading comprehension datasets that have the added benefit of being directly related to the practical end-task of answering user yes/no questions.

Yes/No questions do appear as a subset of some existing datasets (Reddy et al., 2018; Choi et al., 2018; Yang et al., 2018). However, these datasets are primarily intended to test other aspects of question answering (QA), such as conversational QA or multi-step reasoning, and do not contain naturally occurring questions.

We follow the data collection method used by Natural Questions (NQ) (Kwiatkowski et al., 2019) to gather 16,000 naturally occurring yes/no questions into a dataset we call BoolQ (for Boolean Questions). Each question is paired with a paragraph from Wikipedia that an independent annotator has marked as containing the answer. The task is then to take a question and passage as input, and to return "yes" or "no" as output. Figure 1 contains some examples, and Appendix A.1 contains additional randomly selected examples.

Following recent work (Wang et al., 2018), we focus on using transfer learning to establish baselines for our dataset. Yes/No QA is closely related to many other NLP tasks, including other forms of question answering, entailment, and paraphrasing. Therefore, it is not clear what the best data sources to transfer from are, or if it will be sufficient to just transfer from powerful pre-trained language models such as BERT (Devlin et al., 2018) or ELMo (Peters et al., 2018). We experiment with state-of-the-art unsupervised approaches, using existing entailment datasets, three methods of leveraging extractive QA data, and using a few other supervised datasets.

We found that transferring from MultiNLI, and the unsupervised pre-training in BERT, gave us the best results. Notably, we found these approaches are surprisingly complementary and can be combined to achieve a large gain in performance. Overall, our best model reaches 80.43% accuracy, compared to 62.31% for the majority baseline and 90% human accuracy. In light of the fact BERT on its own has achieved human-like performance on several NLP tasks, this demonstrates the high degree of difficulty of our dataset. We present our data and code at `https://goo.gl/boolq`.

2 Related Work

Yes/No questions make up a subset of the reading comprehension datasets CoQA (Reddy et al., 2018), QuAC (Choi et al., 2018), and HotPotQA (Yang et al., 2018), and are present in the ShARC (Saeidi et al., 2018) dataset. These datasets were built to challenge models to understand conversational QA (for CoQA, ShARC and QuAC) or multi-step reasoning (for HotPotQA), which complicates our goal of using yes/no questions to test inferential abilities. Of the four, QuAC is the only one where the question authors were not allowed to view the text being used to answer their questions, making it the best candidate to contain naturally occurring questions. However, QuAC still heavily prompts users, including limiting their questions to be about pre-selected Wikipedia articles, and is highly class imbalanced with 80% "yes" answers.

The MS Marco dataset (Nguyen et al., 2016), which contains questions with free-form text answers, also includes some yes/no questions. We experiment with heuristically identifying them in Section 4, but this process can be noisy and the quality of the resulting annotations is unknown. We also found the resulting dataset is class imbalanced, with 80% "yes" answers.

Yes/No QA has been used in other contexts, such as the templated bAbI stories (Weston et al.) or some Visual QA datasets (Antol et al., 2015; Wu et al., 2017). We focus on answering yes/no questions using natural language text.

Question answering for reading comprehension in general has seen a great deal of recent work (Rajpurkar et al., 2016; Joshi et al., 2017), and there have been many recent attempts to construct QA datasets that require advanced reasoning abilities (Yang et al., 2018; Welbl et al., 2018; Mihaylov et al., 2018; Zellers et al., 2018; Zhang et al., 2018). However, these attempts typically involve engineering data to be more difficult by, for example, explicitly prompting users to write multi-step questions (Yang et al., 2018; Mihaylov et al., 2018), or filtering out easy questions (Zellers et al., 2018). This risks resulting in models that do not have obvious end-use applications since they are optimized to perform in an artificial setting. In this paper, we show that yes/no questions have the benefit of being very challenging even when they are gathered from natural sources.

Natural language inference is also a well studied area of research, particularly on the MultiNLI (Williams et al., 2018) and SNLI (Bowman et al., 2015) datasets. Other sources of entailment data include the PASCAL RTE challenges (Bentivogli et al., 2009, 2011) or SciTail (Khot et al., 2018). We note that, although SciTail, RTE-6 and RTE-7 did not use crowd workers to generate candidate statements, they still use sources (multiple choices questions or document summaries) that were written by humans with knowledge of the premise text. Using naturally occurring yes/no questions ensures even greater independence between the questions and premise text, and ties our dataset to a clear end-task. BoolQ also requires detecting entailment in paragraphs instead of sentence pairs.

Transfer learning for entailment has been studied in GLUE (Wang et al., 2018) and SentEval (Conneau and Kiela, 2018). Unsupervised pre-training in general has recently shown excellent results on many datasets, including entailment data (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018).

Converting short-answer or multiple choice questions into entailment examples, as we do when experimenting with transfer learning, has been proposed in several prior works (Demszky et al., 2018; Poliak et al., 2018; Khot et al., 2018). In this paper we found some evidence suggesting that these approaches are less effective than using crowd-sourced entailment examples when it comes to transferring to natural yes/no questions.

Contemporaneously with our work, Phang et al. (2018) showed that pre-training on supervised tasks could be beneficial even when using pre-trained language models, especially for a textual entailment task. Our work confirms these results for yes/no question answering.

This work builds upon the Natural Questions (NQ) (Kwiatkowski et al., 2019), which contains some natural yes/no questions. However, there are too few (about 1% of the corpus) to make yes/no QA a very important aspect of that task. In this paper, we gather a large number of additional yes/no questions in order to construct a dedicated yes/no QA dataset.

3 The BoolQ Dataset

An example in our dataset consists of a question, a paragraph from a Wikipedia article, the title of the article, and an answer, which is either "yes" or "no". We include the article title since it can potentially help resolve ambiguities (e.g., coreferent phrases) in the passage, although none of the models presented in this paper make use of them.

3.1 Data Collection

We gather data using the pipeline from NQ (Kwiatkowski et al., 2019), but with an additional filtering step to focus on yes/no questions. We summarize the complete pipeline here, but refer to their paper for a more detailed description.

Questions are gathered from anonymized, aggregated queries to the Google search engine. Queries that are likely to be yes/no questions are heuristically identified: we found selecting queries where the first word is in a manually constructed set of indicator words[3] and are of sufficient length, to be effective.

Questions are only kept if a Wikipedia page is returned as one of the first five results, in which case the question and Wikipedia page are given to a human annotator for further processing.

Annotators label question/article pairs in a three-step process. First, they decide if the question is *good*, meaning it is comprehensible, unambiguous, and requesting factual information. This judgment is made before the annotator sees the Wikipedia page. Next, for good questions, annotators find a passage within the document that contains enough information to answer the question. Annotators can mark questions as "not answerable" if the Wikipedia article does not contain the requested information. Finally, annotators mark whether the question's answer is "yes" or "no". Annotating data in this manner is quite expensive since annotators need to search entire Wikipedia documents for relevant evidence and read the text carefully.

Note that, unlike in NQ, we only use questions that were marked as having a yes/no answer, and pair each question with the selected passage instead of the entire document. This helps reduce ambiguity (ex., avoiding cases where the document supplies conflicting answers in different paragraphs), and keeps the input small enough so that existing entailment models can easily be applied to our dataset.

We combine 13k questions gathered from this

---

[3] The full set is: {"did", "do", "does", "is", "are", "was", "were", "have", "has", "can", "could", "will", "would"}.

Question Topic

Category	Example	Percent	Yes%
Entertainment Media	Is You and I by Lady Gaga a cover?	22.0	65.9
Nature/Science	Are there blue whales in the Atlantic Ocean?	22.0	56.8
Sports	Has the US men's team ever won the World Cup?	11.0	54.5
Law/Government	Is there a seat belt law in New Hampshire?	10.0	70.0
History	Were submarines used in the American Civil War?	5.0	70.0
Fictional Events	Is the Incredible Hulk part of the avengers?	4.0	87.5
Other	Is GDP per capita same as per capita income?	26.0	65.4

Question Type

Category	Example	Percent	Yes%
Definitional	Is thread seal tape the same as Teflon tape?	14.5	55.2
Existence	Is there any dollar bill higher than a 100?	14.5	69.0
Event Occurrence	Did the great fire of London destroy St. Paul's Cathedral?	11.5	73.9
Other General Fact	Is there such thing as a dominant eye?	29.5	62.7
Other Entity Fact	Is the Arch in St. Louis a national park?	30.0	63.3

Table 1: Question categorization of BoolQ. Question topics are shown in the top half and question types are shown in the bottom half.

pipeline with an additional 3k questions with yes/no answers from the NQ training set to reach a total of 16k questions. We split these questions into a 3.2k dev set, 3.2k test set, and 9.4k train set, ensuring questions from NQ are always in the train set. "Yes" answers are slightly more common (62.31% in the train set). The queries are typically short (average length 8.9 tokens) with longer passages (average length 108 tokens).

3.2 Analysis

In the following section we analyze our dataset to better understand the nature of the questions, the annotation quality, and the kinds of reasoning abilities required to answer them.

3.3 Annotation Quality

First, in order to assess annotation quality, three of the authors labelled 110 randomly chosen examples. If there was a disagreement, the authors conferred and selected a single answer by mutual agreement. We call the resulting labels "gold-standard" labels. On the 110 selected examples, the answer annotations reached 90% accuracy compared to the gold-standard labels. Of the cases where the answer annotation differed from the gold-standard, six were ambiguous or debatable cases, and five were errors where the annotator misunderstood the passage. Since the agreement was sufficiently high, we elected to use singly-annotated examples in the training/dev/test sets in order to be able to gather a larger dataset.

3.4 Question Types

Part of the value of this dataset is that it contains questions that people genuinely want to answer. To explore this further, we manually define a set of topics that questions can be about. An author categorized 200 questions into these topics. The results can be found in the upper half of Table 1.

Questions were often about entertainment media (including T.V., movies, and music), along with other popular topics like sports. However, there are still a good portion of questions asking for more general factual knowledge, including ones about historical events or the natural world.

We also broke the questions into categories based on what kind of information they were requesting, shown in the lower half of Table 1. Roughly one-sixth of the questions are about whether anything with a particular property exists (Existence), another sixth are about whether a particular event occurred (Event Occurrence), and another sixth ask whether an object is known by a particular name, or belongs to a particular category (Definitional). The questions that do not fall into these three categories were split between requesting facts about a specific entity, or requesting more general factual information.

We do find a correlation between the nature of the question and the likelihood of a "yes" answer. However, this correlation is too weak to help outperform the majority baseline because, even if the topic or type is known, it is never best to guess the minority class. We also found that question-only models perform very poorly on this task (see Section 5.3), which helps confirm that the questions

Reasoning Types		Yes/No Question Answering Examples
Paraphrasing (38.7%) The passage explicitly asserts or refutes what is stated in the question.	**Q**:	Is Tim Brown in the Hall of Fame?
	P:	Brown has also played for the Tampa Bay Buccaneers. In 2015, he was inducted into the Pro Football Hall of Fame.
	A:	Yes. ["inducted into" directly implies he is in Hall of Fame.]
By Example (11.8%) The passage provides an example or counter-example to what is asserted by the question.	**Q**:	Are there any nuclear power plants in Michigan?
	P:	. . . three nuclear power plants supply Michigan with about 30% of its electricity.
	A:	Yes. [Since there must be at least three.]
Factual Reasoning (8.5%) Answering the question requires using world-knowledge to connect what is stated in the passage to the question.	**Q**:	Was designated survivor filmed in the White House?
	P:	The series is. . . filmed in Toronto, Ontario.
	A:	No. [The White House is not located in Toronto.]
Implicit (8.5%) The passage mentions or describes entities in the question in way that would not make sense if the answer was not yes/no.	**Q**:	Is static pressure the same as atmospheric pressure?
	P:	The aircraft designer's objective is to ensure the pressure in the aircraft's static pressure system is as close as possible to the atmospheric pressure. . .
	A:	No. [It would not make sense to bring them "as close as possible" if those terms referred to the same thing.]
Missing Mention (6.6%) We can conclude the answer is yes or no because, if this was not the case, it would have been mentioned in the passage.	**Q**:	Did Bonnie Blair's daughter make the Olympic team?
	P:	Blair and Cruikshank have two children: a son, Grant, and daughter, Blair.... Blair Cruikshank competed at the 2018 United States Olympic speed skating trials at the 500 meter distance.
	A:	No. [The passage describes Blair Cruikshank's daughter's skating accomplishments, so it would have mentioned it if she had qualified.]
Other Inference (25.9%) The passage states a fact that can be used to infer whether the answer is true or false, and does not fall into any of the other categories.	**Q**:	Is the sea snake the most venomous snake?
	P:	. . . the venom of the inland taipan, drop by drop, is the most toxic among all snakes
	A:	No. [If inland taipan is the most venomous snake, the sea snake must not be.]

Table 2: Kinds of reasoning needed in the BoolQ dataset.

do not contain sufficient information to predict the answer on their own.

3.5 Types of Inference

Finally, we categorize the kinds of inference required to answer the questions in BoolQ[4]. The definitions and results are shown in Table 2.

Less than 40% of the examples can be solved by detecting paraphrases. Instead, many questions require making additional inferences (categories "Factual Reasoning", "By Example", and "Other Inference") to connect what is stated in the passage to the question. There is also a significant class of questions (categories "Implicit" and "Missing Mention") that require a subtler kind of inference based on how the passage is written.

3.6 Discussion

Why do natural yes/no questions require inference so often? We hypothesize that there are several factors. First, we notice factoid questions that ask about simple properties of entities, such as "Was Obama born in 1962?", are rare. We suspect this is because people will almost always prefer to phrase such questions as short-answer questions (e.g., "When was Obama born?"). Thus, there is a natural filtering effect where people tend to use yes/no questions exactly when they want more complex kinds of information.

Second, both the passages and questions rarely include negation. As a result, detecting a "no" answer typically requires understanding that a positive assertion in the text excludes, or makes unlikely, a positive assertion in the question. This requires reasoning that goes beyond paraphrasing (see the "Other-Inference" or "Implicit" examples).

We also think it was important that annotators only had to answer questions, rather than generate them. For example, imagine trying to construct questions that fall into the categories of "Missing Mention" or "Implicit". While possible, it would require a great deal of thought and creativity. On the other hand, detecting when a yes/no question can be answered using these strategies seems much easier and more intuitive. Thus, having annotators answer pre-existing questions opens the door to building datasets that contain more inference and have higher quality labels.

---

[4] Note the dataset has been updated since we carried out this analysis, so it might be slighly out-of-date.

4 Training Yes/No QA Models

Models on this dataset need to predict an output class given two pieces of input text, which is a well studied paradigm (Wang et al., 2018). We find training models on our train set alone to be relatively ineffective. Our best model reaches 69.6% accuracy, only 8% better than the majority baseline. Therefore, we follow the recent trend in NLP of using transfer learning. In particular, we experiment with *pre-training* models on related tasks that have larger datasets, and then *fine-tuning* them on our training data. We list the sources we consider for pre-training below.

Entailment: We consider two entailment datasets, *MultiNLI* (Williams et al., 2018) and *SNLI* (Bowman et al., 2015). We choose these datasets since they are widely-used and large enough to use for pre-training. We also experiment with ablating classes from MultiNLI. During fine-tuning we use the probability the model assigns to the "entailment" class as the probability of predicting a "yes" answer.

Multiple-Choice QA: We use a multiple choice reading comprehension dataset, *RACE* (Lai et al., 2017), which contains stories or short essays paired with questions built to test the reader's comprehension of the text. Following what was done in SciTail (Khot et al., 2018), we convert questions and answer-options to statements by either substituting the answer-option for the blanks in fill-in-the-blank questions, or appending a separator token and the answer-option to the question. During training, we have models independently assign a score to each statement, and then apply the softmax operator between all statements per each question to get statement probabilities. We use the negative log probability of the correct statement as a loss function. To fine-tune on BoolQ, we apply the sigmoid operator to the score of the question given its passage to get the probability of a "yes" answer.

Extractive QA: We consider several methods of leveraging extractive QA datasets, where the model must answer questions by selecting text from a relevant passage. Preliminary experiments found that simply transferring the lower-level weights of extractive QA models was ineffective, so we instead consider three methods of constructing entailment-like data from extractive QA data.

First, we use the *QNLI* task from GLUE (Wang et al., 2018), where the model must determine if a sentence from SQuAD 1.1 (Rajpurkar et al., 2016) contains the answer to an input question or not. Following previous work (Hu et al., 2018), we also try building entailment-like training data from *SQuAD 2.0* (Rajpurkar et al., 2018). We concatenate questions with either the correct answer, or with the incorrect "distractor" answer candidate provided by the dataset, and train the model to classify which is which given the question's supporting text.

Finally, we also experiment with leveraging the long-answer portion of NQ, where models must select a paragraph containing the answer to a question from a document. Following our method for Multiple-Choice QA, we train a model to assign a score to (question, paragraph) pairs, apply the softmax operator on paragraphs from the same document to get a probability distribution over the paragraphs, and train the model on the negative log probability of selecting an answer-containing paragraph. We only train on questions that were marked as having an answer, and select an answer-containing paragraph and up to 15 randomly chosen non-answer-containing paragraphs for each question. On BoolQ, we compute the probability of a "yes" answer by applying the sigmoid operator to the score the model gives to the input question and passage.

Paraphrasing: We use the Quora Question Paraphrasing (*QQP*) dataset, which consists of pairs of questions labelled as being paraphrases or not.[5] Paraphrasing is related to entailment since we expect, at least in some cases, passages will contain a paraphrase of the question.

Heuristic Yes/No: We attempt to heuristically construct a corpus of yes/no questions from the MS Marco corpus (Nguyen et al., 2016). MS Marco has free-form answers paired with snippets of related web documents. We search for answers starting with "yes" or "no", and then pair the corresponding questions with snippets marked as being related to the question. We call this task *Y/N MS Marco*; in total we gather 38k examples,

---

[5] data.quora.com/First-Quora-Dataset-Release-Question-Pairs

80% of which are "yes" answers.

Unsupervised: It is well known that unsupervised pre-training using language-modeling objectives (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2018), can improve performance on many tasks. We experiment with these methods by using the pre-trained models from *ELMo*, *BERT*, and OpenAI's Generative Pre-trained Transformer (*OpenAI GPT*) (see Section 5.2).

5 Results

5.1 Shallow Models

First, we experiment with using a linear classifier on our task. In general, we found features such as word overlap or TF-IDF statistics were not sufficient to achieve better than the majority-class baseline accuracy (62.17% on the dev set). We did find there was a correlation between the number of times question words occurred in the passage and the answer being "yes", but the correlation was not strong enough to build an effective classifier. "Yes" is the most common answer even among questions with zero shared words between the question and passage (with a 51% majority), and more common in other cases.

5.2 Neural Models

For our experiments that do not use unsupervised pre-training (except the use of pre-trained word vectors), we use a standard recurrent model with attention. Our experiments using unsupervised pre-training use the models provided by the authors. In more detail:

Our *Recurrent* model follows a standard recurrent plus attention architecture for text-pair classification (Wang et al., 2018). It embeds the premise/hypothesis text using fasttext word vectors (Mikolov et al., 2018) and learned character vectors, applies a shared bidirectional LSTM to both parts, applies co-attention (Parikh et al., 2016) to share information between the two parts, applies another bi-LSTM to both parts, pools the result, and uses the pooled representation to predict the final class. See Appendix A.2 for details.

Our *Recurrent +ELMo* model uses the language model from Peters et al. (2018) to provide contextualized embeddings to the baseline model outlined above, as recommended by the authors.

Our *OpenAI GPT* model fine-tunes the 12 layer 768 dimensional uni-directional transformer from Radford et al. (2018), which has been pre-trained as a language model on the Books corpus (Zhu et al., 2015).

Our $BERT_L$ model fine-tunes the 24 layer 1024 dimensional transformer from Devlin et al. (2018), which has been trained on next-sentence-selection and masked language modelling on the Book Corpus and Wikipedia.

We fine-tune the $\mathrm{BERT_L}$ and the OpenAI GPT models using the optimizers recommended by the authors, but found it important to tune the optimization parameters to achieve the best results. We use a batch size of 24, learning rate of 1e-5, and 5 training epochs for BERT and a learning rate of 6.25e-5, batch size of 6, language model loss of 0.5, and 3 training epochs for OpenAI GPT.

5.3 Question/Passage Only Results

Following the recommendation of Gururangan et al. (2018), we first experiment with models that are only allowed to observe the question or the passage. The pre-trained $\mathrm{BERT_L}$ model reached 64.48% dev set accuracy using just the question and 66.74% using just the passage. Given that the majority baseline is 62.17%, this suggests there is little signal in the question by itself, but that some language patterns in the passage correlate with the answer. Possibly, passages that present more straightforward factual information (like Wikipedia introduction paragraphs) correlate with "yes" answers.

5.4 Transfer Learning Results

The results of our transfer learning methods are shown in Table 3. All results are averaged over five runs. For models pre-trained on supervised datasets, both the pre-training and the fine-tuning stages were repeated. For unsupervised pre-training, we use the pre-trained models provided by the authors, but continue to average over five runs of fine-tuning.

QA Results: We were unable to transfer from RACE or SQuAD 2.0. For RACE, the problem might be domain mismatch. In RACE the passages are stories, and the questions often query for passage-specific information such as the author's intent or the state of a particular entity from the passage, instead of general knowledge.

We would expect SQuAD 2.0 to be a better match for BoolQ since it is also Wikipedia-based, but its possible detecting the adversarially-

Transfer Task	Model	Transfer Data	#Examples	Source Acc.	BoolQ Acc.
N/A	Majority	-	-	-	62.17
N/A	Recurrent	-	-	-	69.60
Extractive QA	Recurrent	QNLI	108k	79.66	71.36
		SQuAD 2.0	130k	69.45	69.83
		NQ Long Answer	93k	71.78	72.78
Paraphrasing	Recurrent	QQP	364k	89.58	71.30
Heuristic Y/N	Recurrent	Y/N MS Marco	39k	87.26	71.40
Entailment	Recurrent	MultiNLI	392k	78.23	75.57
		- w/o Entail	262k	84.26	72.95
		- w/o Contradict	262k	81.16	72.85
		- w/o Neutral	262k	89.72	74.83
		SNLI	351k	88.17	73.16
MC QA	Recurrent	RACE	549k	42.30	68.40
Unsupervised	Recurrent +ELMo	Billion Word	1000M	-	71.41
	OpenAI GPT	Books	800M	-	72.87
	BERT_L	Books/Wikipedia	3,300M	-	76.90

Table 3: Transfer learning results on the BoolQ dev set after fine-tuning on the BoolQ training set. Results are averaged over five runs. In all cases directly using the pre-trained model without fine-tuning did not achieve results better than the majority baseline, so we do not include them here.

constructed distractors used for negative examples does not relate well to yes/no QA.

We got better results using QNLI, and even better results using NQ. This shows the task of selecting text relevant to a question is partially transferable to yes/no QA, although we are only able to gain a few points over the baseline.

Entailment Results: The MultiNLI dataset out-performed all other supervised methods by a large margin. Remarkably, this approach is only a few points behind BERT despite using orders of magnitude less training data and a much more light-weight model, showing high-quality pre-training data can help compensate for these deficiencies.

Our ablation results show that removing the neutral class from MultiNLI hurt transfer slightly, and removing either of the other classes was very harmful, suggesting the neutral examples had limited value. SNLI transferred better than other datasets, but worse than MultiNLI. We suspect this is due to limitations of the photo-caption domain it was constructed from.

Other Supervised Results: We obtained a small amount of transfer using QQP and Y/N MS Marco. Although Y/N MS Marco is a yes/no QA dataset, its small size and class imbalance likely contributed to its limited effectiveness. The web snippets it uses as passages also present a large domain shift from the Wikipedia passages in BoolQ.

Model	Dev Acc.	Test Acc.
Majority Class	62.17	62.31
Recurrent	70.28	67.52
+MultiNLI	76.15	74.24
Pre-trained BERT_L	78.09	76.70
+MultiNLI	82.20	80.43

Table 4: Test set results on BoolQ, "+MultiNLI" indicates models that were additionally pre-trained on MultiNLI before being fine-tuned on the train set.

Unsupervised Results: Following results on other datasets (Wang et al., 2018), we found BERT_L to be the most effective unsupervised method, surpassing all other methods of pre-training.

5.5 Multi-Step Transfer Results

Our best single-step transfer learning results were from using the pre-trained BERT_L model and

MultiNLI. We also experiment with combining these approaches using a two-step pre-training regime. In particular, we fine-tune the pre-trained BERT_L on MultiNLI, and then fine-tune the resulting model again on the BoolQ train set. We found decreasing the number of training epochs to 3 resulted in a slight improvement when using the model pre-trained on MultiNLI.

We show the test set results for this model, and some other pre-training variations, in Table 4. For these results we train five versions of each model using different training seeds, and show the model that had the best dev-set performance.

Given how extensively the BERT_L model has been pre-trained, and how successful it has been across many NLP tasks, the additional gain of 3.5 points due to using MultiNLI is remarkable. This suggests MultiNLI contains signal orthogonal to what is found in BERT's unsupervised objectives.

5.6 Sample Efficiency

In Figure 2, we graph model accuracy as more of the training data is used for fine-tuning, both with and without initially pre-training on MultiNLI. Pre-training on MultiNLI gives at least a 5-6 point gain, and nearly a 10 point gain for BERT_L when only using 1000 examples. For small numbers of examples, the recurrent model with MultiNLI pre-training actually out-performs BERT_L.

5.7 Discussion

A surprising result from our work is that the datasets that more closely resemble the format of BoolQ, meaning they contain questions and multi-sentence passages, such as SQuAD 2.0, RACE, or Y/N MS Marco, were not very useful for transfer. The entailment datasets were stronger despite consisting of sentence pairs. This suggests that adapting from sentence-pair input to question/passage input was not a large obstacle to achieving transfer. Preliminary work found attempting to convert the yes/no questions in BoolQ into declarative statements did not improve transfer from MultiNLI, which supports this hypothesis.

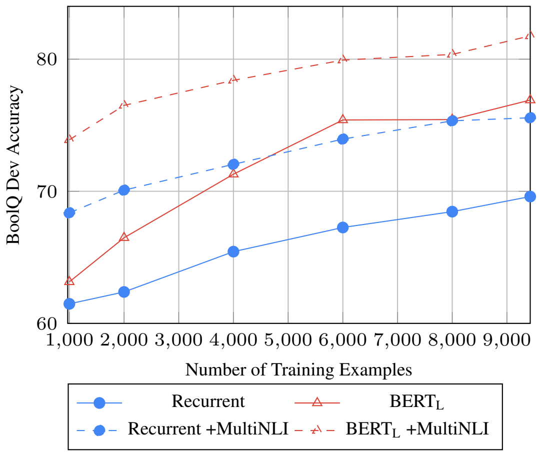


Figure 2: Accuracy for various models on the BoolQ dev set as the number of training examples varies.

The success of MultiNLI might also be surprising given recent concerns about the generalization abilities of models trained on it (Glockner et al., 2018), particularly related to "annotation artifacts" caused by using crowd workers to write the hypothesis statements (Gururangan et al., 2018). We have shown that, despite these weaknesses, it can still be an important starting point for models being used on natural data.

We hypothesize that a key advantage of MultiNLI is that it contains examples of contradictions. The other sources of transfer we consider, including the next-sentence-selection objective in BERT, are closer to providing examples of entailed text vs. neutral/unrelated text. Indeed, we found that our two step transfer procedure only reaches 78.43% dev set accuracy if we remove the contradiction class from MultiNLI, regressing its performance close to the level of BERT_L when just using unsupervised pre-training.

Note that it is possible to pre-train a model on several of the suggested datasets, either in succession or in a multi-task setup. We leave these experiments to future work. Our results also suggest pre-training on MultiNLI would be helpful for other corpora that contain yes/no questions.

6 Conclusion

We have introduced BoolQ, a new reading comprehension dataset of naturally occurring yes/no questions. We have shown these questions are challenging and require a wide range of inference abilities to solve. We have also studied how transfer learning performs on this task, and found crowd-sourced entailment datasets can be leveraged to boost performance even on top of language model pre-training. Future work could include building a document-level version of this task, which would increase its difficulty and its correspondence to an end-user application.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the IEEE international conference on computer vision*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2011. The Seventh PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. *EMNLP*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *ACL*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuaC: Question Answering in Context. *EMNLP*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An Evaluation Toolkit for Universal Sentence Representations. In *LREC*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming Question Answering Datasets Into Natural Language Inference Datasets. arXiv:1809.02922. Version 2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. Version 1.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. *ACL*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation Artifacts in Natural Language Inference Data. *NAACL*.

Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. 2018. Read+ Verify: Machine Reading Comprehension with Unanswerable Questions. *CoRR*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. *EMNLP*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *ACL*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A Textual Entailment Dataset from Science Question Answering. In *Proceedings of AAAI*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ICLR*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *TACL*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-Scale Reading Comprehension Dataset from Examinations. *EMNLP*.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. *Computing Research Repository*, arXiv:1902.01007. Version 1.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *EMNLP*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. arXiv:1611.09268. Version 3.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks. *Computing Research Repository*, arXiv:1811.01088. Version 2.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation. In *EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-training.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ Questions for Machine Comprehension of Text. *EMNLP*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. CoQA: A Conversational Question Answering Challenge. In *TACL*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of Natural Language Rules in Conversational Machine Reading. In *EMNLP*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *ICLR*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. In *ACL*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *ICLR*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *NACL*.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual Question Answering: A Survey of Methods and Datasets. In *Computer Vision and Image Understanding*. Elsevier.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. *Computing Research Repository*, arXiv:1810.12885. Version 1.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendices

A.1 Randomly Selected Examples

We include a number of randomly selected examples from the BoolQ train set in Figure 3. For each example we show the question in bold, followed by the answer in parentheses, and then the passage below.

A.2 Recurrent Model

Our recurrent model is a standard model from the text pair classification literature, similar to the one used in the GLUE baseline (Wang et al., 2018) and the model from Chen et al. (2017). Our model has the following stages:

Embed: Embed the words using a character CNN following what was done by Seo et al. (2017), and the fasttext crawl word embeddings (Mikolov et al., 2018). Then run a BiLSTM over the results to get context-aware word hypothesis embeddings $\langle u_1, u_2, u_3, ...\rangle$ and premise embeddings $\langle v_1, v_2, v_3, ...\rangle$.

Co-Attention: Compute a co-attention matrix, A, between the hypothesis and premise where $A_{ij} = w_1 \cdot u_i + w_2 \cdot v_j + w_3 \cdot (u_i \circ v_j)$, $\circ$ is elementwise multiplication, and w_1, w_2, and w_3 are weights to be learned.

Attend: For each row in A, apply the softmax operator and use the results to compute a weighed sum of the hypothesis embeddings, resulting in attended vectors $\langle \tilde{u}_1, \tilde{u}_2, ...\rangle$. We use the transpose of A to compute vectors $\langle \tilde{v}_1, \tilde{v}_2, ...\rangle$ from the premise embeddings in a similar manner.

Pool: Run another BiLSTM over $\langle [v_1; \tilde{v}_1; \tilde{v}_1 \circ v_1], [v_2; \tilde{v}_2; \tilde{v}_2 \circ v_2], ...\rangle$ to get embeddings $\langle h_1, h_2, ...\rangle$. Then pool these embeddings by computing attention scores $a_i = w \cdot h_i$, $p = softmax(a)$, and then the sum v^* = $\sum_i p_i h_i$. Likewise we compute p^* from the premise.

Classify: Finally we feed $[v^*; p^*]$ into a fully connected layer, and then through a softmax layer to predict the output class.

We apply dropout at a rate of 0.2 between all layers, and train the model using the Adam optimizer (Kingma and Ba, 2014). The learning rate is decayed by 0.999 every 100 steps. We use 200 dimensional LSTMs and a 100 dimensional fully connected layer.

Is there a catalytic converter on a diesel? (Y)

A catalytic converter is an exhaust emission control device that converts toxic gases and pollutants in exhaust gas from an internal combustion engine into less-toxic pollutants by catalyzing a redox reaction (an oxidation and a reduction reaction). Catalytic converters are usually used with internal combustion engines fueled by either gasoline or diesel–including lean-burn engines as well as kerosene heaters and stoves.

Is there a season 2 of Pride and Prejudice? (N)

Pride and Prejudice is a six-episode 1995 British television drama, adapted by Andrew Davies from Jane Austen's 1813 novel of the same name. Jennifer Ehle and Colin Firth starred as Elizabeth Bennet and Mr. Darcy. Produced by Sue Birtwistle and directed by Simon Langton, the serial was a BBC production with additional funding from the American A&E Network. BBC1 originally broadcast the 55-minute episodes from 24 September to 29 October 1995. The A&E Network aired the series in double episodes on three consecutive nights beginning 14 January 1996. There are six episodes in the series.

Is Saving Private Ryan based on a book? (N)

In 1994, Robert Rodat wrote the script for the film. Rodat's script was submitted to producer Mark Gordon, who liked it and in turn passed it along to Spielberg to direct. The film is loosely based on the World War II life stories of the Niland brothers. A shooting date was set for June 27, 1997.

Is The Talk the same as The View? (N)

In November 2008, the show's post-election day telecast garnered the biggest audience in the show's history at 6.2 million in total viewers, becoming the week's most-watched program in daytime television. It was surpassed on July 29, 2010, during which former President Barack Obama first appeared as a guest on The View, which garnered a total of 6.6 million viewers. In 2013, the show was reported to be averaging 3.1 million daily viewers, which outpaced rival talk show The Talk.

Does the concept of a contact force apply to both a macroscopic scale and an atomic scale? (N)

In the Standard Model of modern physics, the four fundamental forces of nature are known to be non-contact forces. The strong and weak interaction primarily deal with forces within atoms, while gravitational effects are only obvious on an ultra-macroscopic scale. Molecular and quantum physics show that the electromagnetic force is the fundamental interaction responsible for contact forces. The interaction between macroscopic objects can be roughly described as resulting from the electromagnetic interactions between protons and electrons of the atomic constituents of these objects. Everyday objects do not actually touch; rather, contact forces are the result of the interactions of the electrons at or near the surfaces of the objects.

Legal to break out of prison in Germany? (Y)

In Mexico, Belgium, Germany and Austria, the philosophy of the law holds that it is human nature to want to escape. In those countries, escapees who do not break any other laws are not charged for anything and no extra time is added to their sentence. However, in Mexico, officers are allowed to shoot prisoners attempting to escape, and an escape is illegal if violence is used against prison personnel or property, or if prison inmates or officials aid the escape.

Is the movie sand pebbles based on a true story? (N)

The Sand Pebbles is a 1966 American war film directed by Robert Wise in Panavision. It tells the story of an independent, rebellious U.S. Navy machinist's mate, first class aboard the fictional gunboat USS San Pablo in 1920s China.

Is Burberrys of London the same as Burberry? (Y)

Burberry was founded in 1856 when 21-year-old Thomas Burberry, a former draper's apprentice, opened his own store in Basingstoke, Hampshire, England. By 1870, the business had established itself by focusing on the development of outdoors attire. In 1879, Burberry introduced in his brand the gabardine, a hardwearing, water-resistant yet breathable fabric, in which the yarn is waterproofed before weaving. "Burberry" was the original name until it became "Burberrys", due to many customers from around the world began calling it "Burberrys of London". In 1999, the name was reverted to the original, "Burberry". However, the name "Burberrys of London" is still visible on many older Burberry products. In 1891, Burberry opened a shop in the Haymarket, London. Before being termed as trench, it was known as the Tielocken worn by the British officers and featured a belt with no buttons, was double breasted, and protected the body from neck to knees.

Is the Saturn Vue the same as the Chevy Equinox? (N)

Riding on the GM Theta platform, the unibody is mechanically similar to the Saturn Vue and the Suzuki XL7. However, the Equinox and the Torrent are larger than the Vue, riding on a 112.5 in (2,858mm) wheelbase, 5.9 in (150mm) longer than the Vue. Front-wheel drive is standard, with optional all-wheel drive. They are not designed for serious off-roading like the truck-based Chevrolet Tahoe and Chevrolet TrailBlazer.

Is Destin FL on the Gulf of Mexico? (Y)

The city is located on a peninsula separating the Gulf of Mexico from Choctawhatchee Bay. The peninsula was originally an island; hurricanes and sea level changes gradually connected the island to the mainland.

Figure 3: Randomly sampled examples from the BoolQ train set.

Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering

Kun Xu[1], **Yuxuan Lai**[2], **Yansong Feng**[2,3], **Zhiguo Wang**[4]
[1]Tencent AI Lab
[2]ICST, Peking University
[3]The MOE Key Laboratory of Computational Linguistics, Peking University
[4]Amazon
`syxu828@gmail.com`
`{erutan,fengyansong}@pku.edu.cn`
`zgw.tomorrow@gmail.com`

Abstract

Traditional Key-value Memory Neural Networks (KV-MemNNs) are proved to be effective to support shallow reasoning over a collection of documents in domain specific Question Answering or Reading Comprehension tasks. However, extending KV-MemNNs to Knowledge Based Question Answering (KB-QA) is not trivia, which should properly decompose a complex question into a sequence of queries against the memory, and update the query representations to support multi-hop reasoning over the memory. In this paper, we propose a novel mechanism to enable conventional KV-MemNNs models to perform interpretable reasoning for complex questions. To achieve this, we design a new *query updating* strategy to mask previously-addressed memory information from the query representations, and introduce a novel *STOP* strategy to avoid invalid or repeated memory reading without strong annotation signals. This also enables KV-MemNNs to produce structured queries and work in a semantic parsing fashion. Experimental results on benchmark datasets show that our solution, trained with question-answer pairs only, can provide conventional KV-MemNNs models with better reasoning abilities on complex questions, and achieve state-of-art performances.

1 Introduction

Memory Neural Networks (MemNNs) [Weston *et al.*, 2014; Sukhbaatar *et al.*, 2015b] are a family of neural network models that aim to learn how to reason with a long-term memory component and various inference components. The memory component serves as a knowledge base to recall facts from the past. MemNNs have been successfully applied in many natural language processing applications such as question answering and reading comprehension (RC). Recently, Miller *et al.* [2016] proposed a variant of MemNNs, namely Key-Value Memory Neural Networks (KV-MemNNs), which generalizes the original MemNNs by storing facts in a key-value structured memory. Figure 1 illustrates the basic architecture of KV-MemNNs, which consists of five components. The question is first fed to the **Embedding** component and the **Hashing** component. The former converts the incoming question to an internal feature representation. The Hashing component uses the question to pre-select a list of facts to compose the key-value memory. The **Key Addressing** component takes the input question representation and the current memory to compute the relevance probability between the question and each key in the memory. The **Value Reading** component reads the values of all addressed memories by taking their weighted sum using the relevance probabilities. The obtained value representation is then added to the query representation to change the query focus for the next round of memory reading. After multiple hops of reasoning over the memories, the final value representation is treated as the answer representation to perform the final prediction over all candidate answers in the **Rank-**

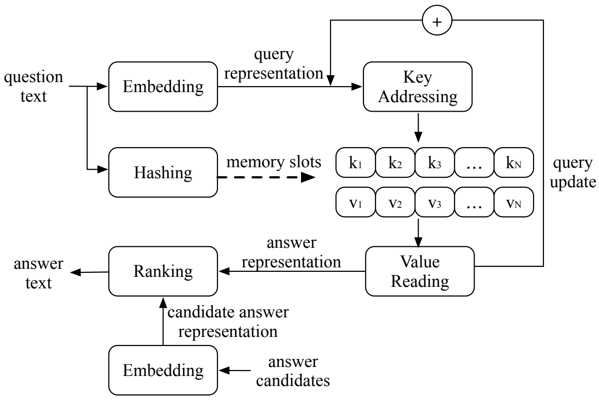


Figure 1: The Key-Value Memory Neural Network Architecture.

Proceedings of NAACL-HLT 2019, pages 2937–2947
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing component.

The KV-MemNNs have been shown to support shallow reasoning in domain-specific knowledge based question answering (KB-QA) tasks such as MovieQA [Tapaswi *et al.*, 2016]. However, when applied to a more challenging scenario, e.g., open domain KB-QA, the KV-MemNNs models do not perform as well as expected, possibly due to two reasons. First of all, the focus of conventional KV-MemNNs is about understanding the facts in the memory rather than properly understanding the questions, where the latter requires incrementally decomposing a complex natural language question into a set of focused queries with the help of the facts in the memory. However, in open domain KB-QA, questions are usually more complicated, e.g., multi-relation questions such as *who does maggie grace play in taken*, where more than one entity and relation are mentioned. Secondly, as shown in Figure 1, KV-MemNNs usually work in an information retrieval (IR) fashion, which first retrieve a set of candidate answers from KB, then rank them by computing the similarity between the value representation and candidates, and finally select the top one or a fixed number of top candidates as the answer. We can imagine that it is not trivial for such IR-styled KV-MemNNs to properly resolve complex constraints from natural language questions, or to handle questions with multiple answers.

We believe that an ideal framework for open domain KB-QA should first understand the natural language questions, explicitly represent the meaning, and make the answer retrieval process more interpretable. To build such an interpretable KV-MemNN KB-QA model, we need to deal with the following challenges: (1) KV-MemNNs often read the memory repeatedly since they do not know when to stop; (2) during multiple memory readings, conventional KV-MemNNs often fail to precisely update the queries for multi-relation questions; (3) strong annotations are usually required to train an interpretable QA model, e.g., the supervision for the memory selection at each hop. To address the challenges, we propose a novel solution to make conventional KV-MemNNs feasible to open domain KB-QA. In particular, we introduce a flexible KV-MemNN solution that can work in both the IR and semantic parsing style with large-scale memory. To this end, we first present a novel query updating method that is able to decompose complex questions and precisely address a relevant key at each hop. Secondly, we introduce a new **STOP** strategy during memory readings, which imports a special key **STOP** into the memory and guides our model to avoid repeated or invalid memory readings. In addition, our proposed model can learn to reason over memory slots with weak supervision, e.g., question-answer pairs only, opposing the strong supervision that most current neural semantic parsers demand, which incurs high labor costs. Experimental results on two benchmark datasets show that our proposed model can not only enhance the reasoning capability of KV-MemNNs, but also be flexible enough to work as a semantic parser, with state-of-the-art performances.

2 Related Work

There are usually two main challenges in the open domain KB-QA task: (1) it often requires the ability to properly analyze and represent the natural language questions against knowledge bases, especially for those involving multiple entities and relations, which we also call as reasoning over the KBs; (2) training such interpretable question understanding models requires considerable strong annotations, which is expensive to obtain in practice. Existing works address these using either the information retrieval (IR) based solutions or the semantic paring (SP) based approaches. The IR-based models [Yao and Van Durme, 2014; Yao, 2015; Bast and Haussmann, 2015; Bordes *et al.*, 2015; Dong *et al.*, 2015; Jain, 2016; Lai *et al.*, 2019] tackle the KB-QA task by developing various ranking models towards the candidate answers, which implicitly meet the reasoning requirements during the candidate-searching step or in designing the ranking functions. In contrast, the SP-based approaches [Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Berant and Liang, 2014; Reddy *et al.*, 2014; Yih *et al.*, 2015; Xu *et al.*, 2016] explicitly represent the meaning of questions as logical forms or structured queries that naturally support reasoning over structured KBs.

More recently, memory based reasoning solutions [Weston *et al.*, 2014; Miller *et al.*, 2016] are proposed to tackle the task. [Weston *et al.*, 2014] proposed the Memory Neural Networks (MemNNs), which enable the neural network models read/write on an external memory component, and are further extended into an End-to-End fashion [Sukhbaatar *et al.*, 2015a]. [Miller *et al.*, 2016] further proposed the Key-Value Memory Network, which generalizes the MemNN by storing facts in a key-value structured memory. Both of the two models could perform shallow reasoning over the memory, since they can find answers by consecutively making predictions over multiple memory slots. Compared to the flat memory slots in MemNNs, the Key-Value design can precisely accommodate more complex structured resources, e.g., discriminating subjects and objects in structured KBs, thus makes KV-MemNNs more flexible and better fit to different applications. These neural networks models are often trained in an end-to-end fashion, making the models relatively less interpretable.

Conceptually similar to our STOP strategy, Shen *et al.* [2017] propose the termination gate mechanism based on a random variable generated from the internal state for reading comprehension. In contrast, our model attempts to learn a general STOP key embedding based on the incrementally updated query representations, which can be learned from question-answer pairs only and lead to more explicit reasoning interpretations over structured KBs. This make our model potentially suit more real scenarios.

Our work is also related to [Jain, 2016; Bao *et al.*, 2016], which are designed to support reasoning for multi-relation questions by exploring the relation path and certain KB schema, e.g., CVT nodes, in the Freebase. The former also considers previously-addressed keys during query updating, but ignores the value representations. Thus, it still requires predefined rules and threshold to artificially add intermediate value representations to update the query. The latter also relies on a set of predefined rules to perform reasoning over Freebase. In contrast, our model incorporates both the key and value representations into the query representations, and update in a more uniform way, thus is more general and supports more reasoning scenarios in KB-QA.

3 Our Model

For a given question x, a knowledge base $\mathcal{KB}$ and the question's answer y, we aim to learn a model such that

$$\mathcal{F}(x, \mathcal{KB}) = \hat{y} \rightarrow y$$

where $\hat{y}$ is the predicted answer. In standard KV-MemNNs, the function $\mathcal{F}$ can be composed of five components, i.e., key hashing, key addressing, value reading, query updating and answer prediction. Next, we will introduce how we design a novel mechanism upon those components to equip KV-MemNNs with more powerful reasoning ability. The architecture of our model is shown in Figure 2.

3.1 Key Hashing

The knowledge facts in $\mathcal{KB}$ are usually organized in a triple <subject, *relation*, object>, such as <Maggie Grace, *fb:actor_character*, Kim>. In KV-MemNNs, these facts are stored in a key-value structured memory, where the key k is composed of the left-hand side entity (subject) and the relation, e.g., $\underline{\text{Maggie Grace } \textit{fb:actor\_character}}$, and the value v is the right-hand side entity (object), e.g., $\underline{\text{Kim}}$. Traditional KV-MemNNs first preselect a list of candidate KB facts $(k_1,v_1),...,(k_n,v_n)$ to compose the key-value memory. Particularly, one can first detect entity mentions in the question, and include all KB facts that contains with one of those entities as subject into the memory. In our experiments, we directly use the entity linking results of Xu *et al.* [2016] and filter out their relations that have more than 100 objects. In practice, we find this strategy could effectively avoid the memory exploding for popular entities. In order to help the model avoid repeated or invalid memory reading, we introduce a special key, **STOP**, into the memory for all questions. The corresponding value of the STOP key is a special symbol represented by an all-zero vector. The STOP key is designed to tell our model that we has already accumulated sufficient facts at hand to answer the question, so there is no need to find other knowledge facts from the memory in later hops.

3.2 Key Addressing & Value Reading

Key addressing is basically a matching process, aiming to find the most suitable key for a given query. It can be formulated as a function that computes the relevance probability p_i between the question x and each key k_i:

$$p_i = \text{Softmax}(A\Phi(x) \cdot A\Phi(k_i))$$

where Φ is a feature map of dimension D, A is a $d \times D$ matrix. The values of memories are then read by taking their weighted sum using the relevance probabilities, and the value representation o is returned to locate the answers or update the

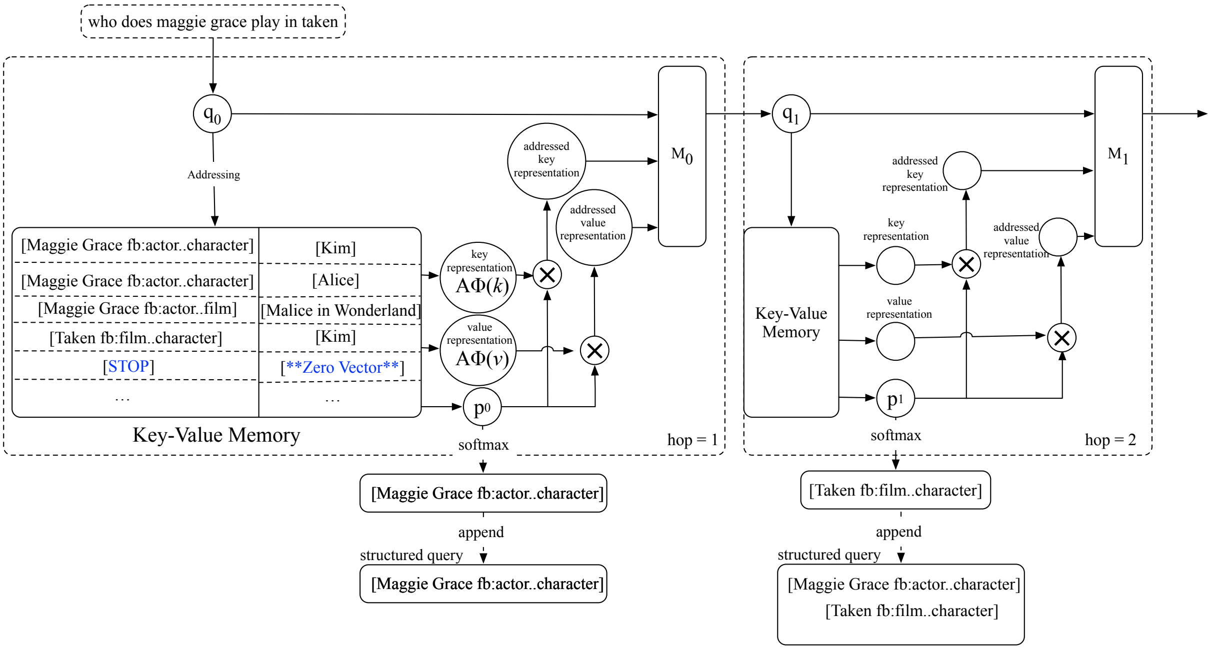


Figure 2: A running example of our key-value memory network model to answer the question *who does maggie grace play in taken*.

query for further memory addressing:

$$o = \sum_i p_i A\Phi(v_i)$$

There are many methods to represent questions and memory slots, including keys and values. Here we simply use the Bag-of-Words model to produce the representations, where we sum the embedding of each word in the question or memory slot together to obtain their vector representations.

3.3 Query Updating

After reading the addressed memory, the initial query representation $q = A\Phi(x)$ should be updated so that the new evidence o collected in the current hop can be properly considered to retrieve more pertinent information in later steps. Traditional KV-MemNNs simply add the initial query q and the returned value o, then perform a linear transformation to obtain the new query representation. This updating strategy is effective in the RC task, since the questions in RC are relatively simple, and their main emphasis is to select proper memory values to change the focus of the query, until reaching the answer. However, the questions in open domain KB-QA tasks are more complicated, usually involved with multiple relations or constraints.

Take the question "*who does maggie grace play in taken*" as an example, the expected answer should follow two constraints: (1) *Maggie_Grace* plays this answer; and (2) this answer is from the movie *Taken*. To answer this question, the model needs to perform two hops of inference consecutively, i.e., matching two keys `Maggie_Grace fb:actor..character` and `Taken fb:film..character` in the memory. Conventional query updating methods, e.g., adding q and o, may not be helpful in guiding the model to predict the other key in later hops, and possibly even hurt the performance, since it may introduce unrelated information into the new query. Intuitively, in KB-QA, masking previously-addressed keys from the query could benefit latter inference, since the model will be able to focus on the next hop, e.g., the movie *Taken*. Therefore, we take into account the query and addressed memories at the t-th hop when updating the query q^{t+1} for the next hop:

$$q^{t+1} = \mathbf{M}_t \cdot (q^t \oplus \underbrace{\sum_i p_i^t A\Phi(k_i)}_{\textit{addressed key}} \oplus \underbrace{\sum_i p_i^t A\Phi(v_i)}_{\textit{addressed value}})$$

where $\oplus$ denotes the concatenation of vectors. The query updating step is parameterized with a different matrix M_t on the t-th hop, which is designed to learn a proper way to combine these three representations.

3.4 Answer Prediction

Conventional KV-MemNNs use the value o at the final hop of inference to retrieve the answers, by simply computing the similarity between o and all candidate answers. This may be of risk for our task. First of all, many questions in the open domain KB-QA have multiple answers, but, KV-MemNNs are supposed to select the candidate with the highest similarity score as the only answer. Secondly, the value representation at the final step may not fully capture the answer information throughout the whole inference process. For example, for multi-constraint questions, the model may address different constraints at different hops, which requires the model to take the value representations in every hop into consideration in order to produce the final answer representation from a global view.

We therefore propose to accumulate the value representations of all hops to make the resulting answer representation lean more on satisfying all constraints. We compute the answer representation m at each hop by adding the value representations of both the current hop and the previous one: $m^{t+1} = o^{t+1} + o^t$, $m^0 = o^0$.

With the final m at hand, we could follow traditional IR-based methods to use the final **A**nswer **R**epresentation to find the best match over all possible candidate values in the memory, namely the **AR** approach. Instead, we can also collect all the best matched keys at every hop to construct a **S**tructured **Q**uery and execute it over the KB to obtain all qualified answers, namely the **SQ** approach. Specifically, the structured query can be constructed as: we select the keys that have the highest relevance probabilities in every hop, resulting a sequence of keys sk_0,.... Starting from sk_0, we append the key sk_i into the final structured query until we see the **STOP** key for the first time at the k-th hop, i.e., $SQ = \{sk_0, ..., sk_{k-1}\}$.

Apparently, the SQ approach can easily output all qualified answers through excuting the queries over the KB, while the AR approach still has difficulties in selecting multiple answers from a ranked list. However, keep in mind that, we do not have gold-standard structured queries for training. As a result, we have to adopt different strategies to find answers in the training and test phases. During **training**, after a fixed number H hops, we follow the AR approach to use the final m^H to compute a prediction over possible candidates, and we train the model by minimizing the cross-entropy between the prediction and gold-standard answers. During **testing**, we follow the SQ approach to collect the final answers by constructing and executing the structured queries over the KB to obtain all answers. As illustrated in Figure 2, for the example question *who does maggie grace play in taken*, our model selects three keys, i.e., [`<maggie grace, fb:actor..character>`, `<taken, film..character>`, and `<STOP>`]. We combine the previous two triples to construct the structured query, which is then executed over the KB to get the answer. We should point out that our model could still use the AR approach to predict the answers just like in the training phase. And as far as we know, our model is the first one that is suitable to tackle the KB-QA task in both the information retrieval and semantic parsing fashions.

3.5 Objective Function and Learning

Given an input question x, the network with parameters θ uses the answer representation m_x^h to perform the prediction over candidate answers at hop h, resulting a prediction vector a_x^h, where the i-th component is respect to the probability of candidate answer i. We denote t_x as the target distribution vector. We compute the standard cross-entropy loss between a_x^h and t_x, and further define the objective function over all training data:

$$L(\theta) = \sum_x \sum_{h=1}^{H} t_x \log a_x^h + \lambda ||\theta||^2$$

where λ is a vector of regularization parameters. Intuitively, this loss function makes our model generate shorter paths to reach answers from the question. On the other hand, it encourages the query updating method to mask the information already addressed in previous hops for the next query representation. This design, together with the query updating method, are the keys to learn the STOP strategy.

4 Experiments

We evaluate our model on two benchmark datasets to investigate whether our enhanced KV-MemNNs model can better perform reasoning over the memory in the open domain KB-QA task, and whether it can make the QA procedure more interpretable.

Semantic Paring Method	Answer F_1
Berant *et al.* [2013]	35.7%
[Berant and Liang, 2014]	39.9%
Yih *et al.* [2015]	52.5%
Bast and Haussmann [2015]	49.4%
Xu *et al.* [2016] (KB Only)	47.1%
Bao *et al.* [2016]	54.4%
CQU + AR (a)	42.0%
KVQU + AR (b)	43.2%
CQU + SQ (c)	39.6%
KVQU + SQ (d)	41.8%
STOP + CQU + AR (e)	43.3%
STOP + CQU + SQ (f)	45.8%
STOP + KVQU + AR (g)	48.6%
STOP + KVQU + SQ (h)	**54.6%**

Table 1: The performance of different models on the test set of WebQuestions.

4.1 Settings

We use the WebQuestions dataset [Berant *et al.*, 2013] as our main dataset, which contains 5,810 question-answer pairs. This dataset is built on Freebase [Bollacker *et al.*, 2008] and all answers are Freebase entities. We use the same training, development and test split as [Berant *et al.*, 2013], containing 3000, 778 and 2032 questions, respectively.

Our model is trained using the Adam optimizer [Kingma and Ba, 2014] with mini-batch size 60. The learning rate is set to 0.001. The complexity of model was penalized by adding L2 regularization to the cross entropy loss function. Gradients are clipped when their norm is bigger than 20. The hop size is set to 3. We initialize word embeddings using pre-trained word representations from Turian *et al.* [2010] and the dimension of word embedding is set to 50.

We use the average question-wise F_1 as our evaluation metric. We compare our model with representative IR based KB-QA models, and several state-of-the-art semantic parsing models. We also include different variants of our model for comparisons to shed light on the advantages of our proposed strategies, especially our three important building blocks - the STOP strategy, how to update a query, and how to obtain the answers.

CQU+AR: uses the **C**onventional **Q**uery **U**pdating method (CQU) which performs a linear transformation over the sum of the query and value representations. This method adopts the AR approach to predict answers where the one with highest probability is selected as the answer.

KVQU+AR: applies the approach introduced in this paper that additionally considers both the **K**ey and **V**alue representations in the **Q**uery **U**pdating (KVQU). This method updates the query representation after reading the memory values at each hop of inference. Note that this model can be seen as a variant of Jain [2016], which is essentially an IR-based approach.

CQU+SQ: uses the CQU method to update the query representations, and applies the SQ approach to obtain the answers.

KVQU+SQ: uses the KVQU method to update the query representations, and adopts the SQ approach to obtain the answers.

STOP+CQU+AR: introduces the STOP key into the memory, but still uses the conventional query updating method and answer representations to find the answers.

STOP+CQU+SQ: introduces the STOP key and uses the conventional query updating method, but uses the SQ approach to obtain the answers.

STOP+KVQU+AR: introduces the STOP key to the memory, uses the KVQU approach to update the query representations, and adopts the AR approach to retrieve the answers.

STOP+KVQU+SQ: is our **main model** that introduces the STOP key, applies the KVQU query updating method, and retrieves answers using the post-constructed structured queries.

4.2 Results and Discussion

Table 1 summarizes the performance of various methods on the test set of WebQuestions. We can see that our main model (**STOP+KVQU+SQ**) performs the best among all its variations, and significantly outperforms the state-of-the-art methods on WebQuestions (with one-tailed t-test significance of $p < 0.05$). We can also see that even the conventional KV-MemNN model (i.e., model (a)) could still outperform traditional semantic parsing models [Berant *et al.*, 2013; Berant and Liang, 2014], showing the effectiveness of memory networks in organizing and utilizing structured data. By replacing the CQU method with our proposed KVQU method, the KV-MemNN model (i.e., model (b)) could further gain an improvement of 1.2%, indicating a proper query representation updating method is critical to KV-MemNNs. After introducing the STOP strategy, the KV-MemNN model is capable to perform proper multi-hop reasoning over the memory, thus outperform most existing methods by a large margin except Bao *et al.* [2016].

Previously, Bao *et al.* [2016] achieves the state-

Question	Addressed Key	
who did armie hammer play in the social network	Armie Hammer fb:award_nominations..award_nominee Armie Hammer fb:award_nominations..award_nominee Armie Hammer fb:award_nominations..award_nominee	before
	Armie Hammer fb:actor..character Social Network fb:film..character **STOP**	after
who is the governor of India 2009	Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..office_holder	before
	Indiana fb:governing_officials..office_holder Indiana fb:governing_officials..from **STOP**	after
what team did david beckham play for in 2011	David Beckham fb:loans..borrowing_team David Beckham fb:player..team David Beckham fb:player..team	before
	David Beckham fb:player..team **STOP** **STOP**	after

Table 2: Running examples of addressed keys and corresponding relevance probabilities before and after introducing the STOP strategy. The question with **blue** color is correctly answered by introducing the STOP strategy while the **red** ones are still not resolved (see discussion in Session 4.8).

of-the-art performance on the WebQuestions by explicitly addressing the multi-constraint questions. Specifically, Bao *et al.* [2016] designs a multi-constraint graph for such questions based on the Freebase schema, and introduces a set of predefined rules to solve complex representations such as max, min, top-X and so on. On the other hand, our model only requires the KB facts to be stored in a Key-Value memory, independent of certain KB schemas. Although those complex expressions in WebQuestions could be easily covered by hand-crafted rules as many did, our model does not work upon such rules, and we think it is crucial to properly enhance KV-MemNNs with such more advanced reasoning abilities, which we leave for future work.

4.3 Impact of the STOP Key

As shown Table 1, we can see that when introducing the STOP strategy, almost all models improve by around 4%, not only in the SQ setting (e.g., from (d) to (f)), but also in the AR setting (e.g., from (a) to (e)). This is not surprising, since the STOP key is introduced to help our model learn to determine when it should stop reading the memory to avoid repeated or invalid addressing. This apparently will lead to more accurate key addressing at each hop, and produce more accurate structured queries. And, better key addressing can also help to generate better value representations at each hop, and finally better answer representation at the last hop. Keep in mind that we obtain such improvement in the case that we do not have gold-standard annotations for the STOP key, but with question-answer pairs only.

To better investigate how the STOP key works, we randomly select 200 questions from the test set and manually analyze the structured queries generated by our model. We find that, for 184 questions (92%) our model predicts the STOP key after one hop of inference and continuously predicts STOP keys in later hops. For the remaining questions, our model predicts two distinct keys before predicting the STOP key. Among these 184 questions, 178 questions (96.7%) can be resolved using exactly a one-triple query. For the remaining 6 questions, it requires two distinct triples to find the answers. This indicates that our model can successfully utilize the STOP strategy to avoid repeated or invalid reading over the memory, at least, for simple questions.

For the multi-relation questions which require at least two hops of inference to find the answers, we evaluate our model on 326 multi-constraint questions selected in Bao *et al.* [2016]. We find that for 283 questions (86.8%), our model performs two hops of inference before predicting the STOP key while for the remaining 13.2%, our model only performs one hop of inference. This demonstrates that even without strong annotations in terms of structured queries, our model still manages to recognize the multi-relation structure and properly stops the invalid reading process. Table 2 illustrates several examples before and after using the STOP strategy.

4.4 Query Updating

In KV-MemNNs, it is important to properly update the query representation after each hop, since the updated query will be used to address more focused information in the next hop, which is especially crucial to support multi-hop reasoning over the memory. We experimented with two query updating approaches, the traditional adding-based method, CQU, and our proposed KVQU, which additionally considers the addressed key representations in previous hops to update the query representation. From Table 1, we can see that no matter which techniques are used to retrieve the final answers, our model can always benefit from replacing the CQU with our KVQU. The main reason may be that KVQU learns to mask already addressed information and retain those untouched, which leads to more accurate key addressing and more expressive answer representations. But, we also observe that by switching from AR to SQ, STOP+KVQU+SQ achieves more improvement than STOP+CQU+SQ. One possible reason is that the KVQU method, together with the STOP strategy, can help address the most relevant keys (i.e., top#1 key) more accurately than CQU does at each hop, which results in more accurate structured queries.

We randomly select 50 multi-constraint questions from the test set of WebQuestions and analyze the addressed keys at each hop. We find that for 48 questions, the model with CQU (model (f)) repeatedly selects the same keys that have the highest relevance probabilities at the first two hops. However, when replacing CQU with KVQU, the model addresses different keys at the first two hops (examples shown in Table 2). One possible reason is that, in contrast to the CQU that only uses the value representation of current hop to update the query, KVQU additionally considers the addressed keys in the current hop, and aims to mask the already addressed information, so that in later hops, the model will focus on the remaining untouched part of the question.

4.5 Candidate Ranking v.s. Structured Query

We investigate two different answer retrieval methods, the IR styled method AR and the semantic parsing styled method SQ, in the experiments. By comparing CQU+AR (a) and CQU+SQ (c) in Table 1, we can see that replacing the answer retrieval method does not boost the performance, but actually hurts. This is not surprising, because with the vanilla CQU, the model does not know how to stop reading the memory, thus may select repeated/invalid keys after the first hop, which will produce incorrect structured queries. Similarly, the KVQU+SQ also suffers from the incorrect structured queries.

Hop Size	STOP + KVQU + SQ	STOP + KVQU + AR
1	38.7%	35.6%
2	45.9%	41.2%
3	**53.2%**	50.4%
4	53.1%	46.7%
5	53.0%	44.2%

Table 3: Performance (answer F_1) of STOP+KVQU+SQ and STOP+KVQU+AR with different hop sizes on the development set of WebQuestions.

After introduce the STOP key into the memory, we find that no matter which query updating methods are used, the models using structured queries (STOP+*+SQ) significantly outperform their corresponding versions using the ranking based method (STOP+*+AR). Despite of the STOP strategy, the improvement may come from two aspects. First, the structured queries in SQ are built from the best key at every hop, thus have captured more global information throughout the reasoning procedure, while the AR method only uses the o in the last hop to retrieve answers. Secondly, the models with SQ can easily output multiple qualified answers from the KB, while the methods with AR can only output the top one from a ranked list in the memory. Moreover, from a practical perspective, the SQ methods provide more interpretability than the ranking based methods.

4.6 Impact of the Hop Size

To investigate the impact of the hop size to the model performance, we evaluate the model STOP+KVQU+SQ and STOP+KVQU+AR with various hop size settings on the development set of WebQuestions. We evaluate both of the two models since we wonder which answer retrieval method is more sensitive to the hop size. As shown in Table 3, the model with AR answer retrieval method achieves the best performance with hop size of 3. Then with the hop size increasing, the performance drops. In contrast, the model with the SQ method also achieves the best performance when the hop size is 3. When the hop size increases, the performance does not drop significantly, but almost remains unchanged. We think the reason may be that the model with the AR method keeps predicting the keys as the hop size increasing, which inevitably introduces noise to the answer representations. On the other hand, the model with the SQ method is less sensitive to the hop size, since once it sees the STOP key, the latter predictions do not affect the resulted structured query.

4.7 Results on QALD-4

To further examine our model under different conditions, besides WebQuestions, we also evaluate our main model on the QALD-4 dataset [Unger *et al.*], which is built upon another KB, DBpedia [Auer *et al.*, 2007]. Like Freebase, DBpedia stores real world facts in the triple format, making it easier for our model to adapt to this new KB. The QALD-4 dataset consists of three QA datasets, namely, Multilingual QA, Biomedical QA and Hybrid QA. Here we evaluate our model on the multilingual QA dataset which contains 250 English question-SPARQL pairs, where 200 questions (80%) are used for training and 50 questions for testing. These questions are more complicated than those of WebQuestions, e.g., all QALD-4 questions require at least two hops of inference over the KB to answer. Table 4 lists the results of our model and other participated systems [Unger *et al.*]. We can see that our model can achieve the best performance on this dataset, without importing extra rules.

Method	Recall	Precision	F-measure
Xser	0.71	0.72	0.72
gAnswer	0.37	0.37	0.37
CASIA	0.40	0.32	0.36
our model	**0.78**	**0.82**	**0.81**

Table 4: Results of the models on the test set of QALD.

4.8 Error Analysis

We analyze the errors of our main model on the development set of WebQuestions. Around 80% errors are caused by incorrect key addressing. The key addressing errors are mainly due to (1) the entities in the addressed keys are incorrect. Entity linking itself is a challenging task and we do not employ any existing entity linking tools to incorporate the linking confidence score, which may further improve our model; (2) the relations in the addressed keys are incorrect, which is mainly caused by insufficient context in the question. For example, in *what is duncan bannatyne*, the information from the question is quite limited for our model to predict the correct key *fb:profession*. Although the STOP strategy proves to be effective on most questions, there are still some questions where the STOP strategy does not work, e.g., *what team did david beckham play for in 2011* shown in Table 2. We failed to perform reasoning over the time constraint *2011* due to limited context left for *2011* after we addressed *play for* in earlier hops. Also due to this same reason, our model can not correctly answer the second question in Table 2, *who is the governor of India 2009*. Keep in mind that, we actually do not have annotated addressed keys at each hop to explicitly teach our model. The remaining cases also include failures in deep analysis according to specific KBs. For example, in *who is the mother of prince michael jackson*, our model only matches the triple *michael jackson fb:parents*, but fail to spot *female* in the next hop.

5 Conclusions

In this paper, we propose to apply the KV-MemNNs as a semantic parsing module to approach the open-domain KB-QA task. We introduce a novel STOP strategy to derive structured queries with flexible number of query triples during multi-hop memory reading, and present a new query updating method, which considers the already-addressed keys in previous hops as well as the value representations. Experimental results show that the STOP strategy not only enables multi-hop reasoning over the memory, but also acts as the key to construct the structured queries, which help our model achieve the state-of-the-art performances on two benchmark datasets.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments and valuable suggestions. We also thank Yao Fu for his constructive comments on the early version of the draft. This work is supported by National High Technology R&D Program of China (Grant No.2018YFC0831905), Natural Science Foundation of China (Grant No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

Jun-Wei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-based question answering with knowledge graph. In *Proceedings of the International Conference on Computational Linguistics (COLING 2016)*, pages 2503–2514, 2016.

Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *CIKM*, 2015.

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 2014.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

Sarthak Jain. Question answering over knowledge base using factual memory networks. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 109–115, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.

Yuxuan Lai, Yansong Feng, Xiaohan Yu, Zheng Wang, Kun Xu, and Dongyan Zhao. Lattice cnns for matching based chinese question answering. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, 2019.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, pages 377–392, 2014.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM, 2017.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 384–394, 2010.

Christina Unger, Corina Forascu, Vanessa López, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. Question answering over linked data (QALD-4). In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, 2014.

Xuchen Yao. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2015)*, 2015.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

Repurposing Entailment for Multi-Hop Question Answering Tasks

Harsh Trivedi[♣], Heeyoung Kwon[♣], Tushar Khot[♠], Ashish Sabharwal[♠], Niranjan Balasubramanian[♣]

[♣] Stony Brook University, Stony Brook, U.S.A.
`{hjtrivedi,heekwon,niranjan}@cs.stonybrook.edu`
[♠] Allen Institute for Artificial Intelligence, Seattle, U.S.A.
`{tushark,ashishs}@allenai.org`

Abstract

Question Answering (QA) naturally reduces to an entailment problem, namely, verifying whether some text entails the answer to a question. However, for multi-hop QA tasks, which require reasoning with *multiple* sentences, it remains unclear how best to utilize entailment models pre-trained on large scale datasets such as SNLI, which are based on sentence pairs.

We introduce Multee, a general architecture that can effectively use entailment models for multi-hop QA tasks. Multee uses (i) a local module that helps locate important sentences, thereby avoiding distracting information, and (ii) a global module that aggregates information by effectively incorporating importance weights. Importantly, we show that both modules can use entailment functions pre-trained on a large scale NLI datasets. We evaluate performance on MultiRC and OpenBookQA, two multihop QA datasets. When using an entailment function pre-trained on NLI datasets, Multee outperforms QA models trained only on the target QA datasets and the OpenAI transformer models.

1 Introduction

How can we effectively use textual entailment models for question answering? Previous attempts at this have resulted in limited success (Harabagiu and Hickl, 2006; Sacaleanu et al., 2008; Clark et al., 2012). With recent large scale entailment datasets (Bowman et al., 2015; Williams et al., 2018; Khot et al., 2018) pushing entailment models to high accuracies (Chen et al., 2017; Parikh et al., 2016; Wang et al., 2017), we re-visit this challenge and propose a novel method for re-purposing neural entailment models for QA.

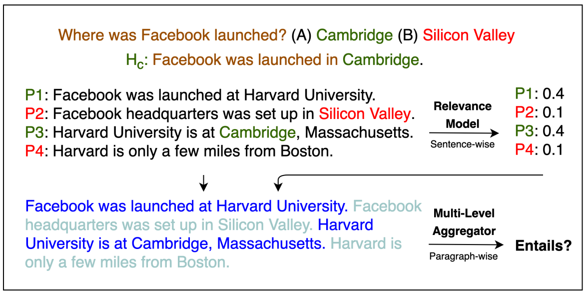


Figure 1: An example illustrating the challenges in using sentence-level entailment model for multi-sentence reasoning needed for QA, and the high-level approach used in Multee.

A key difficulty in using entailment models for QA turns out to be the mismatch between the inputs to the two tasks: large-scale entailment datasets are typically framed at a *sentence level*, whereas question answering requires verifying whether *multiple sentences*, taken together as a premise, entail a hypothesis.

There are two straightforward ways to address this mismatch: (1) aggregate independent entailment decisions over each premise sentence, or (2) make a single entailment decision after concatenating all premise sentences. Neither approach is fully satisfactory. To understand why, consider the set of premises in Figure 1, which entail the hypothesis H_c. Specifically, the combined information in $P1$ and $P3$ entails H_c, which corresponds to the correct answer *Cambridge*. On one hand, aggregating independent decisions will fail because no individual premise entails H_C. On the other hand, simply concatenating premises to form a single paragraph will fail because distracting information in $P2$ and $P4$ can muddle useful information in $P1$ and $P3$. An effective approach, therefore, must recognize relevant sentences (i.e., avoid distracting ones) and compose their sentence-level information.

Our solution to this challenge is based on the observation that a sentence-level entailment function can be re-purposed for both recognizing relevant sentences, and for computing sentence-level representations. Both tasks require comparing in-

Proceedings of NAACL-HLT 2019, pages 2948–2958
Minneapolis, Minnesota, June 2 - June 7, 2019.

formation in a pair of texts, but the objectives of the comparison are different. This means we can take an entailment function that is trained for basic entailment (i.e., comparing information in texts), and adapt it to work for both recognizing relevance and computing representations. Thus, this architecture allows us to incorporate advances in entailment architectures and to leverage pre-trained models obtained using large scale entailment datasets.

To this end, we propose a general architecture that uses a (pre-trained) entailment function f_e for multi-sentence QA. Given a hypothesis statement H_{qa} representing a candidate answer, and the set of premise sentences $\{P_i\}$, our proposed architecture uses the same function f_e for two components: (a) a sentence relevance module that scores each P_i based on its potential relevance to H_{qa}, with the goal of weeding out distractors; and (b) a relevance-weighted aggregator that combines entailment information from multiple P_i.

Thus, we build effective entailment aware representations of larger contexts (i.e., multiple sentences) from those of small contexts (i.e., individual sentences). The main strength of our approach is that, unlike standard attention mechanisms, the aggregator module uses the attention scores from the relevance module at *multiple* levels of abstractions (e.g., multiple layers of a neural network) within f_e, using *join* operations that compose representations at each level. We refer to this **multi-level** aggregation of **te**xtual **e**ntailment representations as Multee (pronounced multi).

Our implementation of Multee uses ESIM (Chen et al., 2017), a recent sentence-level entailment model, pre-trained on SNLI and MultiNLI datasets. We demonstrate its effectiveness on two challenging multi-sentence reasoning datasets: MultiRC (Khashabi et al., 2018) and OpenBookQA (Mihaylov et al., 2018). Multee using ELMo contextual embeddings (Peters et al., 2018) matches state-of-the-art results achieved with large transfomer-based models (Radford et al., 2018) that were trained on a sequence of large scale tasks (Sun et al., 2019). Ablation studies demonstrate that both relevance scoring and multi-level aggregation are valuable, and that pre-training on large entailment corpora is particularly helpful for OpenBookQA.

This work makes three main contributions: **(i)** A novel approach to use pre-trained entailment models for question answering. **(ii)** A model that incorporates local (sentence level) entailment decisions with global (document level) entailment decisions to effectively aggregate information for multi-hop QA task. **(iii)** An empirical evaluation that shows entailment based QA can achieve state-of-the-art performance on two challenging multi-hop QA datasets, OpenBookQA and MultiRC.

2 Question Answering using Entailment

Non-extractive question answering can be seen as a textual entailment problem, where we verify whether a hypothesis constructed out of a question and a candidate answer is entailed by the knowledge—a collection of sentences[1] in the source text. The probability of an answer A, given a question Q, can be modeled as the probability of a set of premises $\{P_i\}$ entailing a hypothesis statement H_{qa} constructed from Q and A:

$$\Pr[A \mid Q, \{P_i\}] \triangleq \Pr[\{P_i\} \vDash H_{qa}] \qquad (1)$$

Here we use $\vDash$ to denote textual entailment. Given QA training data, we can then learn a model that approximates the entailment probability $\Pr[\{P_i\} \vDash H_{qa}]$.

Can one build an effective QA model g_e using an existing entailment model f_e that has been pre-trained on a large-scale entailment dataset? Figure 2 illustrates two straightforward ways of doing so, using f_e as a black-box function:

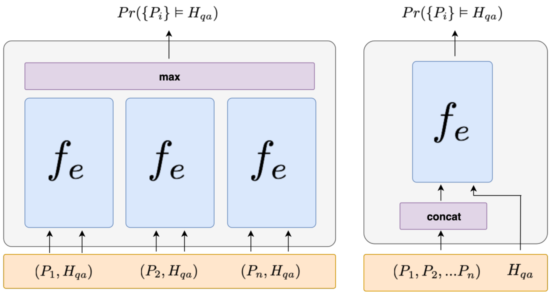


Figure 2: Black Box Applications of Textual Entailment Model for QA: Max and Concat models

(i) Aggregate Local Decisions (Max): Use f_e to check how much each sentence P_i entails H_{qa} on its own, and aggregate these local entailment decisions, for instance, using a max operation.

$$g_e(\{P_i\}, H_{qa}) = \max_i f_e(P_i, H_{qa}) \qquad (2)$$

---

[1] This collection can be a *sequence* in the case of passage comprehension or a *list* of sentences, potentially from varied sources, in the case of QA over multiple documents.

(ii) Concatenate Premises (Concat): Combine the premise sentences in a sequence to form a single large passage P, and use f_e to check whether this passage as a whole entails the hypothesis H_{qa}, making a single entailment decision:

$$g_e(\{P_i\}, H_{qa}) = f_e(P, H_{qa}) \qquad (3)$$

Our experiments reveal, however, that neither approach is an effective means of using pre-trained entailment models for QA (see Table 1). For the example in Figure 1, *Max* model would not be able to consider information from P1 and P3 together. Instead, it will pickup *Silicon Valley* as the answer since P2 is close to H_s, *"Facebook was launched in Silicon Valley"*. Similarly, *Concat* would also be muddled by distracting information in P2, which will weaken its confidence in answer *Cambridge*. Therefore, without careful guidance, simple aggregation can easily add distracting information into the premise representation, causing entailment to fail. This motivates the need for new, effective mechanisms for global reasoning over a collection of premises.

3 Our Approach: Multee

We propose a new entailment based QA model, Multee, with two components: (i) **a sentence relevance model**, which learns to focus on the relevant sentences, and (ii) **a multi-layer aggregator**, which uses an entailment model to obtain multiple layers of question-relevant representations for the premises and then composes them using the sentence-level scores from the relevance model. Finding relevant sentences is a form of local entailment between each premise and the answer hypothesis, whereas aggregating question-relevant representations is a form of global entailment between all premises and the answer hypothesis. This means, we can effectively re-purpose the same pre-trained entailment function f_e for both components. Figure 3 shows an architecture that uses multiple copies of f_e to achieve this.

3.1 Sentence Relevance Model

The goal of this module is to identify sentences in the paragraph that are important for the given hypothesis. As shown in Figure 1, this helps the global module aggregate relevant content while reducing the chances of picking up distracting information. A sentence is considered important if it contains information that is relevant to answering the question. In other words, the importance of a sentence can be modeled as its entailment probability, i.e., how well the sentence by itself supports the answer hypothesis. We can use a pre-trained entailment model to obtain this. The importance α_i of a sentence P_i can be modeled as:

$$\alpha_i = f_e(P_i, H_{qa}) \qquad (4)$$

This can be further improved by modeling the sentence with its surrounding context. This is especially useful for passage-level QA, where the neighboring sentences provide useful context. Given a premise sentence P_i, the entailment function f_e computes a single hypothesis-aware representation x_i containing information in the premise that is relevant to entailing the answer hypothesis H_{qa}. This is essentially the output of last layer of neural function f_e before projecting it to logits. We denote this part of f_e that outputs last *vector* representation as f_{e_v} and full f_e that gives entailment *probability* as f_{e_p}.

We use these hypothesis-aware x_i vectors for each sentence as inputs to a BiLSTM producing a contextual representation c_i for each premise sentence P_i, which is then fed to a feedforward layer that predicts the sentence-level importance as:

$$\alpha_i = \mathrm{softmax}(W^T c_i + b) \qquad (5)$$

The components for generating x_i are part of the original entailment function f_e and can be pre-trained on the entailment dataset. The BiLSTM to compute c_i and the parameters W and b for computing α_i are not part of the original entailment function and thus can only be trained on the target QA task. We perform this additional *contextualization* only when sentences form contiguous text. Additionally, for datasets such as MultiRC, where the relevant sentences have been marked, we introduce a loss term based on the true relevance label and predicted weights, α_i.

3.2 Multi-level Aggregation

The goal of this module is to aggregate representations from important sentences in order to make a global entailment decision. There are two key questions to answer: (1) how to combine the sentence-level information into a paragraph-level representation and (2) how to use the sentence relevance weights $\{\alpha_i\}$.

Most entailment models include many layers that transform the input premise and the hypothesis. A typical neural entailment stack includes en-

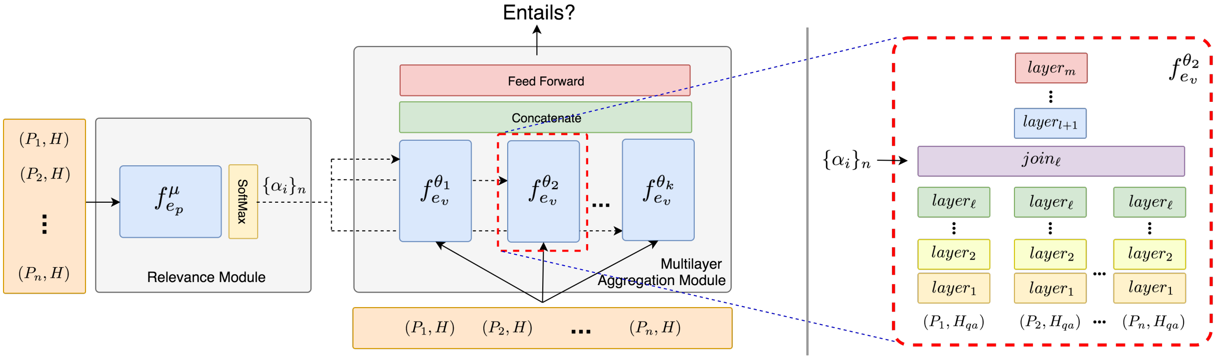


Figure 3: Multee overview: Multee includes two main components, a relevance module, and a multi-layer aggregator module. Both modules use pre-trained entailment functions (f_{e_p} and f_{e_v}). f_{e_p} is the full entailment model that gives entailment probability, and f_{e_v} is part of it excluding last projection to logits and softmax. The multi-level aggregator uses multiple copies of entailment function f_{e_v}, one for each sub-aggregator performing a *join* at a different layer. Right part of figure zooms in on one such sub-aggregator joining at layer ℓ.

coding layers that independently generate contextual representations of the premise and the hypothesis, followed by some cross-attention layer that yields relationships between the premise and hypothesis words, and additional layers that use this cross-attention to generate premise attended representations of the hypothesis and vice versa. The final layers are classification layers which determine entailment based on the representations from the previous layer. Each layer thus generates intermediate representation that captures different type of entailment related information. This presents us with a choice of multiple points for aggregation.

Figure 3 illustrates our approach for aggregating sentence-level representations into a single paragraph level representation. For each premise P_i in the passage, we first process the pair (P_i, H_{qa}) through the entailment stack (f_{e_v}) resulting in a set of intermediate representations $\{\tilde{X_i}^{\ell}\}$ for each layer ℓ. We can choose a particular layer ℓ to be the aggregation layer. We then compute a weighted combination of the sentence-level outputs at this layer $\{\tilde{X_i}^{\ell}\}$ to produce a passage-level representation $\tilde{Y}^{\ell}$. The weights for the sentences are obtained from the *Sentence Relevance* model. We refer to this as a *join* operation as shown in the Figure 3. Layers of the entailment function f_{e_v} that are below the *join* operate at a sentence-level, while layers above the join now operate over paragraph-wise representations. The final layer (i.e. the top most layer) of f_{e_v} thus gives us a vector representation of the entire passage. This type of join can be applied at multiple layers resulting in paragraph vectors that correspond to multiple levels of aggregation. We concatenate these paragraph vectors and pass them through a feed-forward network projecting them down to logits, that can be used to compute the final passage wide entailment probabilities.

3.2.1 Join Operations

Given a set of sentence-wise outputs from the lower layer $\{\tilde{X_i}\}$ and the corresponding sentence-relevance weights $\{\alpha_i\}$, the join operation combines them into a single passage-level representation $\tilde{Y}$, which can be directly consumed by the layer above it in the stack. The specifics of the join operation depends on the shape of the outputs from the lower layer, and the shape of the inputs expected by the layer after the join. Here we show four possible join operations, one for each layer. The ones defined for *Score Layer* and *Embedding Layer* can be reduced to black-box baselines, while we use the other two in Multee.

Score Layer: The score layer outputs the entailment probabilities $\{s_i\}$ for each premise to hypothesis independently, which need to be joined to one entailment score. One way to do this is to simply take a weighted maximum of the individual entailment probabilities. So we have $\tilde{X_i} = s_i \quad \forall i$ and $\tilde{Y} = \max_i \big(\alpha_i s_i\big)$. This reduces to black-box *Max* model (Equation 2) when using $\{\alpha_i\} = \mathbf{1}$.

Embedding Layer: The embedding layer outputs a sequence of embedded vectors of $[\bar{P_i}]$[2] one sequence for each premise P_i and another sequence of embedded vectors $[\bar{H}_{qa}]$ for the answer hypothesis H_{qa}. A join operation in this case scales each embedded vector in a premise by its relevance weight and concatenates them together to

---

[2] We use [.] to denote a sequence and $\bar{.}$ to denote a vector

form $[\bar{P}]$. $\bar{H}_{qa}$ is passed through unchanged.

$$\tilde{X}_i = ([\bar{P}_i], [\bar{H}_{qa}]) \quad \forall i$$
$$[\bar{P}] = [\alpha_1[\bar{P}_1]; \alpha_2[\bar{P}_2]; \ldots; \alpha_n[\bar{P}_n]]$$
$$\tilde{Y} = \left([\bar{P}], [\bar{H}_{qa}]\right)$$

For non-contextual word embeddings, this reduces to *Concat Premises* (Eq. 3) when $\{\alpha_i\} = \mathbf{1}$.

Final Layer (FL): The final layer in the entailment stack usually outputs a single vector $\bar{h}$ which is then used in a linear layer and softmax to produce label probabilities. The join operation here is a weighted sum of the premise-level vectors. So we have $\tilde{X}_i = \bar{h}_i \quad \forall i$ and $\tilde{Y} = \sum_i \alpha_i \bar{h}_i$.

This is similar to a standard attention mechanism, where attended representation is computed by summing the scaled representations. However, such scaled addition is not possible when the outputs from lower layers are not of the same shapes, as in the following case.

Cross Attention Layer (CA): Cross-attention is a standard component of many entailment and reading comprehension models. This layer produces three outputs: (i) For each premise P_i, we get a hypothesis to premise cross attention matrix M^{hp_i} with shape (h $\times$ p_i), where h is the number of hypothesis tokens, and p_i is the number of tokens in premise P_i; (ii) for each premise P_i, we get a sequence of vectors $[\bar{P}_i]$ that corresponds to the token sequence of the premise P_i; and (iii) for the hypothesis, we get a single sequence of vectors $[\bar{H}_{qa}]$ that corresponds to its token sequence. M^{hp_i} attention matrix was generated by cross attention from $[\bar{H}_{qa}]$ to $[\bar{P}_i]$.

The join operation in this layer produces a cross attention matrix that spans the entire passage, i.e., has shape $(h \times p)$, where p is the total number of tokens across all premises. The operation first scales the cross-attention matrices by the sentence-relevance weights $\{\alpha_i\}$ in order to "tone down" the influence of distracting/irrelevant sentences, and then re-normalizes the final matrix:

$$\tilde{X}_i = (M^{hp_i}, [\bar{P}_i], [\bar{H}_{qa}]) \quad \forall i$$
$$M^{hp} = \left[\alpha_i M^{hp_1}; \quad \ldots; \quad \alpha_i M^{hp_n}\right]$$
$$M^{hp}_{ij} = \frac{M^{hp}_{ij}}{\sum_k M^{hp}_{ik}}$$
$$[\bar{P}] = \left[[\bar{P}_1]; \quad [\bar{P}_2]; \ldots; [\bar{P}_n]\right]$$
$$\tilde{Y} = (M^{hp}, \bar{P}, \bar{H}_{qa})$$

where M^{hp}_{ij} is i^{th} row and j^{th} column of M^{hp}.

Multee's multi-layer aggregator module uses join operations at two levels: **Cross Attention Layer (CA)** and **Final Layer (FL)**. The two corresponding aggregators share parameters up till the lower of the two join layers (CA in this case), where they both operate at the sentence level. Above this layer, one aggregator switches to operating at the paragraph level, where it has its own, unshared parameters. In general, if Multee were to aggregate at layers $\ell_{i1}, \ell_{i2}, \ldots, \ell_{ik}$, then the aggregators with joins at layers ℓ and ℓ' respectively could share parameters at layers $1, \ldots, \min\{\ell, \ell'\}$.

3.3 Implementation Details

Multee uses the ESIM stack as the entailment function pre-trained on SNLI and MultiNLI for both the relevance module and for the multi-layer aggregator module. It uses aggregation at two-levels, one at the cross-attention level (CA) and one at the final layer (FL). All uses of the entailment function in Multee are initialized with the same pre-trained entailment model weights. The embedding layer and the BiLSTM layer process paragraph-level contexts but processing at higher layers are done either at premise level or paragraph-level depending on where the join operation is performed.

4 Experiments

Datasets: We evaluate Multee on two datasets, OpenBookQA (Mihaylov et al., 2018) and MultiRC (Khashabi et al., 2018), both of which are specifically designed to test reasoning over multiple sentences. MultiRC is paragraph-based multiple-choice QA dataset derived from varying topics where the questions are answerable based on information from the paragraph. In MultiRC, each question can have more than one correct answer choice, and so it can be viewed as a binary classification task (one prediction per answer choice), with 4,848 / 4,583 examples in Dev/Test sets. OpenBookQA, on the other hand, has multiple-choice science questions with exactly one correct answer choice and no associated paragraph. As a result, this dataset requires the relevant facts to be retrieved from auxiliary resources including the open book of facts released with the paper and other sources such as WordNet (Miller, 1995) and ConceptNet (Speer and Havasi, 2012). It contains 500 questions in the Dev and Test sets.

		OpenBookQA		MultiRC	
		Accuracy		F1a | F1m | EM	
		Dev	Test	Dev	Test
Entailment Based Models with ELMo	Multee	**56.2**	54.8	69.6 | 73.0 | **22.8**	**70.4** | **73.8** | **24.5**
	Concatenate Premises	47.2	47.6	68.3 | 71.3 | 17.9	68.5 | 72.5 | 18.0
	Max of Local Decisions	47.8	45.2	66.5 | 69.3 | 16.3	66.70 | 70.4 | 19.4
Entailment Based Models with GloVe	Multee	54.6	**55.8**	68.3 | 71.7 | 16.4	69.9 | 73.6 | 19.0
	Concatenate Premises	47.4	42.6	66.9 | 70.7 | 14.8	68.7 | 72.4 | 16.3
	Max of Local Decisions	44.4	47.6	66.8 | 70.3 | 16.9	67.7 | 71.1 | 18.2
Previously Published Results	LR (Khashabi et al., 2018)	—	—	63.7 | 66.5 | 11.8	63.5 | 66.9 | 12.8
	IR (Khashabi et al., 2018)	—	—	60.0 | 64.3 |	54.8 | 53.9 |
	QM + ELMo (Mihaylov et al., 2018)	54.6	50.2	—	—
	ESIM + ELMo (Mihaylov et al., 2018)	53.9	48.9	—	—
	KER (Mihaylov et al., 2018)	55.6	51.4	—	—
Large Transformer Models	OFT (Sun et al., 2019)	—	52.0	67.2 | 69.3 | 15.2	—
	OFT (ensemble) (Sun et al., 2019)	—	52.8*	67.7 | 70.3 | 16.5*	—
	RS (Sun et al., 2019)	—	55.2*	69.2 | 71.5 | 22.6*	—
	RS (ensemble) (Sun et al., 2019)	—	**55.8***	**70.5** | **73.1** | 21.8*	—

Table 1: Comparison of Multee with other systems. Starred (*) results are based on contemporaneous system. Results marked (—) are not available. RS is Reading Strategies, KER is Knowledge Enhanced Reader, OFT is OpenAI FineTuned Transformer.

Preprocessing: For each question and answer choice, we create an answer hypothesis statement using a modified version of the script used in SciTail (Khot et al., 2018) construction. We wrote a handful of rules to better convert the question and answer to a hypothesis. We also mark the span of answer in the hypothesis with special begin and end tokens, `@@@answer` and `answer@@@` respectively[3]. For MultiRC, we also apply an off-the-shelf coreference resolution model[4] and replace the mentions when they resolve to pronouns occurring in a different sentence[5]. For OpenBookQA, we use the exact same retrieval as released by the authors of OpenBookQA[6] and use the OpenBook and WordNet as the knowledge source with top 5 sentences retrieved per query.

Training Multee: For OpenBookQA we use cross entropy loss for labels corresponding to 4 answer choices. For MultiRC, we use binary cross entropy loss for each answer-choice separately since in MultiRC each question can have more than one correct answer choice. The entailment components are pre-trained on sentence-level entailment tasks and then fine-tuned as part of end-to-end QA training. The MultiRC dataset includes sentence-level relevance labels. We supervise the Sentence Relevance module with a binary cross entropy loss for predicting these relevance labels when available. We used PyTorch (Paszke et al., 2017) and AllenNLP to implement our models and ran them on Beaker[7]. For pre-training we use the same hyper-parameters of ESIM(Chen et al., 2017) as available in implementation of AllenNLP (Gardner et al., 2017) and fine-tune the model parameters. We do not perform any hyper-parameter tuning for any of our models. We fine-tune all layers in ESIM except for the embedding layer.

Models Compared: We experiment with Glove (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings for Multee and compare with following three types of systems:

(A) Baselines using entailment as a black-box We use the pre-trained entailment model as a black-box in two ways: concatenate premises (*Concat*) and aggregate sentence level decisions with a max operation (*Max*). Both models were also pre-trained on SNLI and MultiNLI datasets and fine-tuned on the target QA datasets with same

---

[3] Answer span marking gave substantial gains for all entailment based models including the baselines.

[4] https://github.com/huggingface/neuralcoref

[5] It is hard to learn co-reference, as these target datasets are too small to learn this in an end-to-end fashion.

[6] https://github.com/allenai/OpenBookQA

[7] `https://beaker.org/`

pre-processing.

(B) Previously published results: For MultiRC, there are two published baselines: IR (Information Retrieval) and LR (Logistic Regression). These simple models turn out to be strong baselines on this relatively smaller sized dataset. For OpenBookQA, we report published baselines from (Mihaylov et al., 2018): Question Match with ELMo (QM + ELMo), Question to Answer ESIM with ELMo (ESIM + ELMo) and their best result with the Knowledge Enhanced Reader (KER).

(C) Large Transformer based models: We compare with OpenAI-Transformer (OFT), pre-trained on large-scale language modeling task and fine-tuned on respective datasets. A contemporaneous work,[8] which published these transformer results, also fine-tuned this transformer further on a large scale reading comprehension dataset, RACE (Lai et al., 2017), before fine-tuning on the target QA datasets with their method, *Reading Strategies*.

4.1 Results

Table 1 summarizes the performance of all models. Multee outperforms the black-box entailment baselines (Concat and Max) that were pre-trained on the same data, previously published baselines, OpenAI transformer models. We note that the 95% confidence intervals around baseline accuracy for OpenBookQA and MultiRC are 4.3% and 1.3%, respectively.

On OpenBookQA test set, Multee with GloVe outperforms ensemble version of OpenAI transformer by 3.0 points in accuracy. It also outperforms single model version of Reading Strategies system and is comparable to their ensemble version. On MultiRC dev set, Multee with ELMo outperforms ensemble version of OpenAI transformer by 1.9 points in F1a, 2.7 in F1m and 6.3 in EM. It also outperforms single model version of Reading Strategies system and is comparable to their ensemble version. Recall that the Reading Strategies results are reported with an additional fine-tuning on another larger QA dataset, RACE (Lai et al., 2017) aside from the target QA datasets we use here.

While ELMo contextual embeddings helped in MultiRC, it did not help OpenBookQA. We believe this is in part due to the mismatch between our ELMo training setup where all sentences are treated as a single sequence, which, while true in MultiRC, is not the case in OpenBookQA.

	OpenBookQA	MultiRC
	Accuracy	F1a | F1m
✗α_i	50.6	67.3 | 70.3
✓α_i	**55.8**	67.4 | 71.0
✓α_i + supervise	—	**68.3** | **71.7**

Table 2: Relevance Model Ablation of Multee. ✗α_i: without relevance weights, ✓α_i: with relevance weights respectively, ✓α_i + supervise: with supervised relevance weights. Test results on OpenBookQA and Dev results on MultiRC.

Aggregator	OpenBookQA	MultiRC
	Accuracy	F1a | F1m
Cross Attention (CA)	45.8	67.2 | 71.1
Final Layer (FL)	51.0	68.3 | 71.5
CA +FL	**55.8**	**68.3** | **71.7**

Table 3: Aggregator Level Ablation of Multee. On MultiRC, Multee uses relevance supervision but not on OpenBookQA because of unavailibility. Test results on OpenBookQA and Dev results on MultiRC.

In general, gains from Multee are more prominent in OpenBookQA than in MultiRC. We hypothesize that a key contributor to this difference is distraction being a lesser challenge in MultiRC, where premise sentences come from a single paragraph whose other sentences are often irrelevant and rarely distract towards incorrect answers. OpenBookQA has a noisier set of sentences, since an equal number of sentences is retrieved for the correct and each incorrect answer choice.

4.2 Ablations

Relevance Model Ablation. Table 2 shows the utility of the relevance module. We use the same setting as the full model (aggregation at Cross Attention (CA) and the Final Layer (FL)). As shown in the table, using the relevance module weights (✓α_i) leads to improved accuracy on both datasets (substantially so in OpenBookQA) as compared to ignoring the module, i.e., setting all weights to 1 (✗α_i). In MultiRC, we show that the additional supervision for the relevance module leads to even further improvements in score.

Multi-Level Aggregator Ablation. Multee performs aggregation at two levels: Cross Attention Layer (CA) and Final Layer (FL). We denote this by CA+FL. To show that multi-level aggregation is better than individual aggregations, we train

---

[8] Published on arXiv on Oct 31, 2018 (Sun et al., 2019).

	OpenBookQA Accuracy	MultiRC F1a | F1m
Snli + MultiNli	**55.8**	**69.9** | **73.6**
Snli	50.4	69.3 | 73.3
Scratch	42.2	68.3 | 72.6

Table 4: Effect (on test data) of pre-training the entailment model used in Multee.

models with aggregation at only FL and at only CA. Table 3 shows that multi-layer aggregation is better than CA or FL alone on both the datasets.

4.3 Effect of Pre-training

One of the benefits of using entailment based components in a QA model is that we can pre-train them on large scale entailment datasets and fine-tune them as part of the QA model. Table 4 shows that such pre-training is valuable. The model trained from scratch is substantially worse in the case of OpenBookQA, highlighting the benefits of our entailment-based QA model.

Multee benefits come from two sources: (i) Re-purposing of entailment function for multi-sentence question answering, and (ii) transferring from a large-scale entailment task. In the case of OpenBookQA, both are helpful. For MultiRC, only the first is a significant contributor. Table 5 shows that re-purposing was a bigger factor for MultiRC, since Max and Concat models do not work well when trained from scratch.

		OpenBookQa Accuracy	MultiRc F1a | F1m
Max	Snli + MultiNli	**47.6**	**66.8** | **70.3**
	Scratch	32.4	42.8 | 44.0
Concat	Snli + MultiNli	**42.6**	**66.9** | **70.7**
	Scratch	35.8	51.3 | 50.4

Table 5: Pre-training ablations of black-box entailment baselines for OpenBookQA (test) and MultiRC (dev).

5 Analysis

Relevance Loss. The sentence-level relevance model provides a way to dig deeper into the overall QA model's behavior. When sentence-level supervision is available, as in the case of MultiRC, we can analyze the impact of different auxiliary losses for the relevance module. Table 6 shows the QA performance with different relevance losses, and Figure 5 shows a visualization of attention scores for a question in MultiRC. Overall, we find that two types of behaviors emerge from different loss functions. For instance, trying to minimize the sum of attention probability mass on irrelevant sentences i.e. $\sum_i \alpha_i(1 - y_i)$, called *IR Sum Loss*, causes the attention scores to become "peaky" i.e, high for one or two sentences, and close to zero for others. This leads to higher precision but at significantly lower recall for the QA system, as it now uses information from fewer but highly relevant sentences. Binary cross entropy loss (BCE) allows the model to attend to more relevant sentences thereby increasing recall without too much drop in precision.

	F1a precision	F1a recall
IR Sum Loss	**59.5**	68.5
BCE Loss	58.0	**83.2**

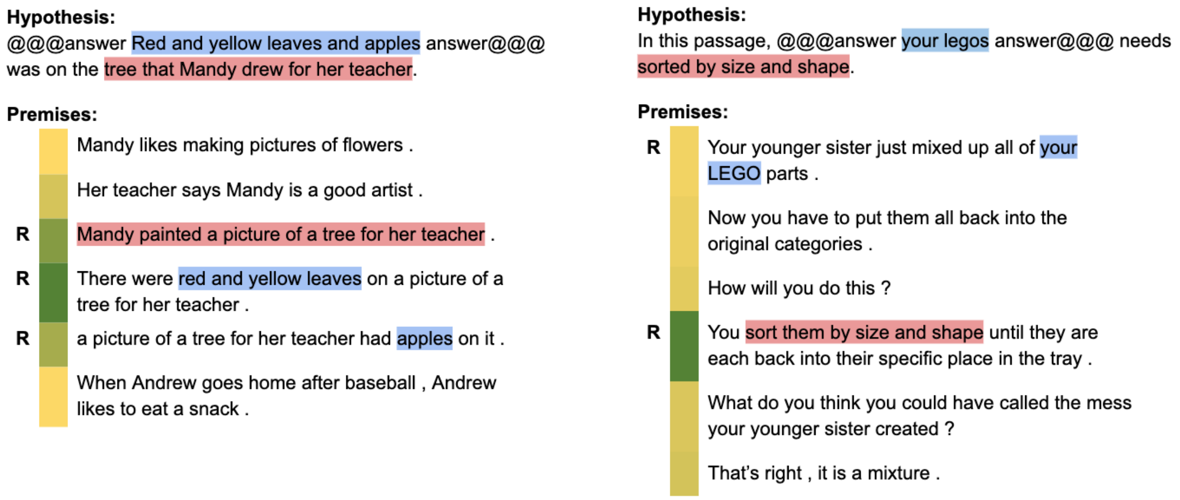
Table 6: F1a precision and recall on MultiRC Dev with 2 kinds of relevance losses. IR Sum is the sum of attention probability mass on irrelevant sentences. BCE is Binary Cross Entropy loss.

Failure Cases. As Figure 5 shows, our model with BCE loss tends to distribute the attention, especially to sentences close to the relevant ones. We hypothesize that the model is learning to use the contextualized BiLSTM representations to incorporate information from neighboring sentences, which is useful for this task and for passage understanding in general. For example, more than 60% of Dev questions in MultiRC have at least one adjacent relevant sentence pair. Figure 4a illustrates this behavior.

On the other hand, if the relevant sentences are far apart, the model finds it difficult to handle such long-range cross sentence dependencies in its contextualized representations. As a result, it ends up focusing attention on the most relevant sentence, missing out on other relevant sentences (Figure 4b). When these unattended but relevant sentences contain the answer, the model fails.

6 Related Work

Entailment systems have been applied to question-answering before but have only had limited success (Harabagiu and Hickl, 2006; Sacaleanu et al., 2008; Clark et al., 2012) in part because of the small size of the early entailment datasets (Dagan et al., 2006, 2013). Recent large scale entailment datasets such as SNLI (Bowman et al.,



Hypothesis:
@@@answer Red and yellow leaves and apples answer@@@ was on the tree that Mandy drew for her teacher.

Premises:
Mandy likes making pictures of flowers .
Her teacher says Mandy is a good artist .
R Mandy painted a picture of a tree for her teacher .
R There were red and yellow leaves on a picture of a tree for her teacher .
R a picture of a tree for her teacher had apples on it .
When Andrew goes home after baseball , Andrew likes to eat a snack .

Hypothesis:
In this passage, @@@answer your legos answer@@@ needs sorted by size and shape.

Premises:
R Your younger sister just mixed up all of your LEGO parts .
Now you have to put them all back into the original categories .
How will you do this ?
R You sort them by size and shape until they are each back into their specific place in the tray .
What do you think you could have called the mess your younger sister created ?
That's right , it is a mixture .

(a) Positive Example (b) Negative Example

Figure 4: Success and failure examples of Multee from MultiRC. **R**: annotated relevant sentences. Green/yellow: high/low predicted relevance.

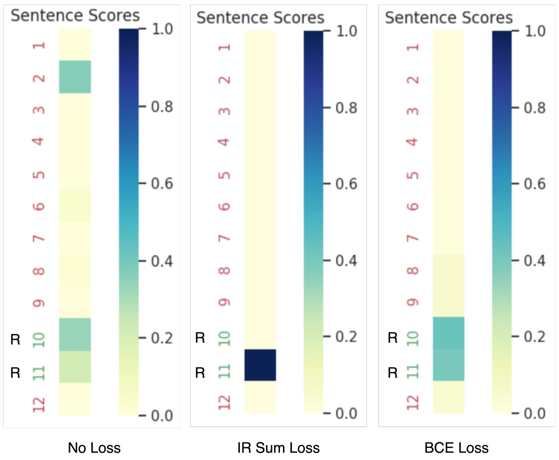


Figure 5: Sentence level attentions for various sentence relevance losses. **R**: annotated relevant sentences.

2015) and MultiNLI (Williams et al., 2018) have led to many new powerful neural entailment models that are not only more effective, but also produce better representations of sentences (Conneau et al., 2017). Models such as Decomposable Attention (Parikh et al., 2016) and ESIM (Chen et al., 2017), on the other hand, find alignments between the hypothesis and premise words through cross-attention. However, these improvements in entailment models have not yet translated to improvements in end tasks such as question answering.

SciTail (Khot et al., 2018) was created from a science QA task to push for models with a direct impact on QA. Entailment models trained on this dataset show minor improvements on the Aristo Reasoning Challenge (Clark et al., 2018; Musa et al., 2018). However, these QA systems make independent predictions and can not combine information from multiple supporting sentences.

Combining information from multiple sentences is a key problem in language understanding. Recent Reading comprehension datasets (Welbl et al., 2018; Khashabi et al., 2018; Yang et al., 2018; Mihaylov et al., 2018) explicitly evaluate a system's ability to perform such reasoning through questions that need information from multiple sentences in a passage. Most approaches on these tasks perform simple attention-based aggregation (Mihaylov et al., 2018; Song et al., 2018; Cao et al., 2018) and do not exploit the entailment models trained on large scale datasets.

7 Conclusions

Using entailment for question answering has seen limited success. Neural entailment models are designed and trained on tasks defined over sentence pairs, whereas QA often requires reasoning over longer texts spanning multiple sentences. We propose Multee, a novel QA model that addresses this mismatch. It uses an existing entailment model to both focus on relevant sentences and aggregate information from these sentences. Results on two challenging QA datasets, as well as our ablation study, indicate that entailment based QA can achieve state-of-the-art performance and is a promising direction for further research.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant IIS-1815358. The computations on beaker.org were supported in part by credits from Google Cloud.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *CoRR* abs/1808.09920.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457* .

Peter Clark, Philip Harrison, and Xuchen Yao. 2012. An entailment-based approach to the qa4mre challenge. In *CLEF*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 670–680. https://www.aclweb.org/anthology/D17-1070.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies* 6(4):1–220.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 905–912.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 785–794. https://doi.org/10.18653/v1/D17-1082.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2381–2391. http://aclweb.org/anthology/D18-1260.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Ryan Musa, Xiaoyan Wang, Achille Fokoue, Nicholas Mattei, Maria Chang, Pavan Kapanipathi, Bassem Makni, Kartik Talamadupula, and Michael Witbrock. 2018. Answering science exam questions using query rewriting with background knowledge. *CoRR* abs/1809.05726.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2249–2255. https://doi.org/10.18653/v1/D16-1244.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, Technical report, OpenAI.

Bogdan Sacaleanu, Constantin Orasan, Christian Spurk, Shiyan Ou, Oscar Ferrandez, Milen Kouylekov, and Matteo Negri. 2008. Entailment-based question answering for structured data. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. Association for Computational Linguistics, pages 173–176.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR* abs/1809.02040.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics* 6:287–302. http://aclweb.org/anthology/Q18-1021.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 1112–1122. https://doi.org/10.18653/v1/N18-1101.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

GenderQuant: Quantifying Mention-Level Genderedness

Ananya
University of California
Irvine, CA
aananya@uci.edu

Nitya Parthasarthi
Massachusetts Institute of
Technology
Cambridge, MA

Sameer Singh
University of California
Irvine, CA
sameer@uci.edu

Abstract

Language is gendered if the context surrounding a mention is suggestive of a particular binary gender for that mention. Detecting the different ways in which language is gendered is an important task since gendered language can bias NLP models (such as for coreference resolution). This task is challenging since genderedness is often expressed in subtle ways. Existing approaches need considerable annotation efforts for each language, domain, and author, and often require handcrafted lexicons and features. Additionally, these approaches do not provide a quantifiable measure of how gendered the text is, nor are they applicable at the fine-grained mention level.

In this paper, we use existing NLP pipelines to automatically annotate gender of mentions in the text. On corpora labeled using this method, we train a supervised classifier to predict the gender of any mention from its context and evaluate it on unseen text. The model confidence for a mention's gender can be used as a proxy to indicate the level of genderedness of the context. We test this gendered language detector on movie summaries, movie reviews, news articles, and fiction novels, achieving an AUC-ROC of up to 0.71, and observe that the model predictions agree with human judgments collected for this task. We also provide examples of detected gendered sentences from aforementioned domains.

1 Introduction

Language can be extraordinarily gendered (Moulton et al., 1978). *Genderedness* in language is when we use words or phrases that are stereotypical or indicative of a particular gender (we only consider male vs female in this work) (Prior, 2017). It is important to detect this bias in language since not only is this bias propagated to the readers (Menegatti and Rubini, 2017), but also machine learning algorithms trained on gendered corpora tend to become biased (Zhao et al., 2018a; Rudinger et al., 2018), often aggravating the disparity (Zhao et al., 2017).

Bias in language and machine learning systems can lead to unfair treatment, e.g., early work by Moulton et al. (1978) shows that males have an advantage in contexts where they are referred to by a putative neutral term. Recent work on coreference resolution systems (Zhao et al., 2018a) shows that bias in machine learning systems originates from training on existing corpora, resulting in male-stereotyped professions like surgeon and president incorrectly resolved to males instead of females. Such biases in machine learning systems can lead to unintentional biases in downstream tasks producing effects like preferential treatment to male candidates over female candidates when selecting resumes (Dastin, 2018). Detecting these biases is the first step in finding a solution.

Most of the current works for related problems tend to be domain-specific (Fu et al., 2016), rely on techniques such as simple counting of gender occurrences (Ali et al., 2010), or use manually constructed lexicons and features for analysis (Trix and Psenka, 2003), and thus do not generalize well and require expensive manual supervision. Existing approaches also tend to either focus on the whole corpus/article being gendered (Schmader et al., 2007; Trix and Psenka, 2003) or a specific word being gendered (Caliskan et al., 2017; Bolukbasi et al., 2016; Zhao et al., 2018b), thus failing to capture the subtle occurrences of genderedness at mention-level or giving a quantifiable measure of how gendered the text is.

In this paper, we develop a method that eliminates the manual annotation requirement, and can generalize to words, phrases, sentences, articles, as well as whole corpora. We present a framework for automated data labeling by combining existing NLP pipelines to identify sentence boundaries and mentions (using NER tagger) and using a gen-

Proceedings of NAACL-HLT 2019, pages 2959–2969
Minneapolis, Minnesota, June 2 - June 7, 2019.

Female	Their client is 🟥 who has got gorgeous hair.
Male	🟥 intends to marry lovely Gauri.
Female	Raja intends to marry lovely 🟥 .

Figure 1: **Examples of gendered sentences.** Our model predicts gender for mentions located at positions indicated by colored boxes.

der classifier for names to get the gender of the mentions. We build a classifier using this annotation to predict gender of a mention *only from its context* and quantitatively analyze the *genderedness* of various contexts using this model. Figure 1 shows example inputs and outputs for our model. Input to the model is the context sentence around a target mention (indicated by colored boxes), and the model prediction is the gender of this mention. For the first sentence, the model uses context information coming from 'gorgeous hair' to predict the gender of mention to be female, indicating a more gendered sentence. Similarly for the third sentence, model uses contextual information from the adjective 'lovely' and its proximity to the target mention to predict female. For the second sentence, the target mention is subject for the verb in phrase 'intends to marry' and the object to be married is 'lovely Gauri'. Our model uses this information to predict gender of target mention as male.

Since our data labeling pipeline is automated, we can easily annotate millions of documents and train complex classifiers that can accurately model the context. These classifiers can be used to predict the gender of a mention from given context and quantify genderedness. We present instantiations of this framework on four domains: news articles, novels, movie summaries, and movie reviews. Since we are the first to study the task of mention-level gender detection, we evaluate the difficulty of the task and introduce the first benchmark using a user study. We find that the task is challenging, and our model predictions corroborate with human predictions. We present qualitative results of our model showing genderedness at different granularities – word, phrase, sentence, and corpus.

2 Related Work

Gender Bias in Datasets A number of approaches have considered gendered language use. Blatt (2017) shows that *shivered, wept, screamed* are disproportionately used to describe women while *muttered, grinned* are used to describe men. Studies on gender bias in student evaluations for instructors (Eidinger, 2017; MacNell et al., 2015; Boring et al., 2016; Centra and Gaubatz, 2000), and recommendation letters (Trix and Psenka, 2003; Schmader et al., 2007) also show similar disparities in terms of harshness of evaluations, length of letters, descriptive words, and use of standout adjectives. Bias in language has also been studied for textbooks (Otlowski, 2003; Gharbavi and Mousavi, 2012; Macaulay and Brice, 1997), Wikipedia edits (Recasens et al., 2013), political text (Yano et al., 2010), media content (Ali et al., 2010; Len-Ríos et al., 2005; Smith, 1997), sports journalism (Eastman and Billings, 2000; Tyler Eastman, 2001; Kinnick, 1998; Fu et al., 2016) and in movie character portrayals (Ramakrishna et al., 2017; Sap et al., 2017). These approaches are domain-specific and rely on techniques like counting gender occurrences, manually annotating words or mentions, constructing list of keywords and lexicons, carrying out surveys, etc. Our approach works across domains, and does not require manual annotations.

There has been significant amount of work in detecting author's gender (Koppel et al., 2002; Herring and Paolillo, 2006; Sarawgi et al., 2011; Mukherjee and Liu, 2010; Burger et al., 2011) for text, speaker gender for dialogues (Schofield and Mehr, 2016) in films, and to detect and reduce biases in these (Tatman, 2017; Thelwall, 2018; Koolen and van Cranenburgh, 2017). While we do not focus on predicting the gender of the author, our framework can be used as a tool to compare the use of gendered language across various authors, or across various works by the same author.

Gender Bias in NLP Pipelines There has also been recent interest in examining the role of gender bias in existing NLP pipelines. Caliskan et al. (2017) and Bolukbasi et al. (2016) show that word embeddings exhibit gender stereotypes. Garg et al. (2018) build on this idea, using word embeddings to characterize the evolution of gender stereotypes during the 20th and 21st centuries. Subsequent works attempt to mitigate this bias in embeddings (Zhao et al., 2018b). Zhao et al. (2019) extend the idea to contextualized word embeddings (Peters et al., 2018), and quantify and propose ways to mitigate gender bias in them , while Gonen and Goldberg (2019) show that current approaches for debiasing embeddings are superficial.

Researchers have studied gender bias outside word embeddings as well. Zhao et al. (2017) show that datasets for multi-label object classification and visual semantic role labeling are gender-biased and that models trained on these datasets amplify this bias, while Rudinger et al. (2017) find racial, religious and gender stereotypes in the SNLI corpus and Park et al. (2018) analyze gender bias in abusive language datasets. Zhao et al. (2018a) and Rudinger et al. (2018) detect bias in existing coreference resolution systems, and Webster et al. (2018) build a gender-balanced labeled corpus of ambiguous pronoun-name pairs to understand this bias. All of these either focus on whether the corpus as a whole is gendered or if a single word is gendered (in case of word embeddings). Instead, we train a classifier to detect and quantify gendered language at mention-level. Our framework can also be used to quantify genderedness at different levels – mention, sentence, document, or corpus.

3 Gendered Language Detection

In this section, we elaborate on gendered language, and describe our proposed architecture and training method to detect and quantify it.

Gendered Language Gendered language is the use of words and phrases that *discriminate*[1] the gender of a subject. In other words, the gender of mentioned person should be easy to predict from context if the text is gendered. Examples of gendered language can be found in the use of stereotypes like linking women to homemakers and men to programmers (Bolukbasi et al., 2016) or when pronouns, adverbs, adjectives, nouns are used carelessly, e.g., when a masculine pronoun mention 'he' is used to refer to both sexes (Cottier, 2018; Stout and Dasgupta, 2011) or when pronoun mentions are used exclusively to define professions by gender (using 'she' when talking about a nurse). Detecting gendered language is incredibly challenging since the ways in which gender is expressed can vary considerably across authors, domains, and time periods, making any approach that requires annotations to be corpus-specific.

Proposed Architecture We are interested in determining the extent to which language in context of the mention reveals the gender. Humans learn to detect gendered language based on a lifetime of reading and observing society, and learning language specific to each gender. We use this intuition to propose an automated framework for detecting mention-level genderedness for any corpus (an overview is shown in Figure 2).

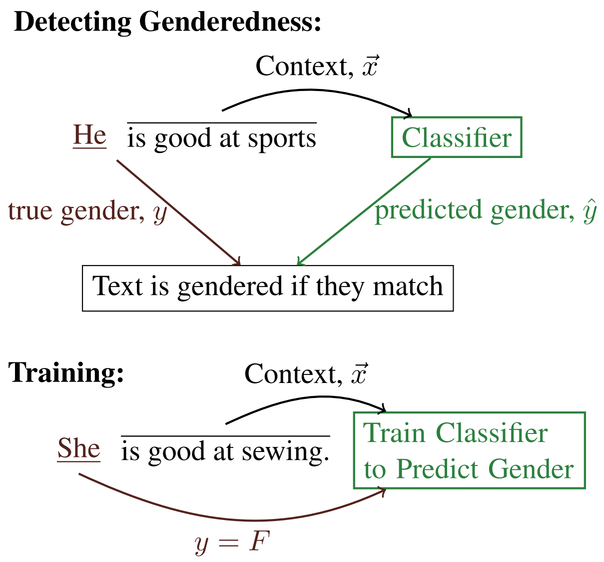


Figure 2: **Model overview.** Using an automatically labeled corpus, we train a classifier to predict the gender for each mention given its context (without the mention). For each target mention and its context sentence, we check whether the predicted gender matches the true prediction; the level of agreement indicates the *genderedness* of the text.

The input to the gender detector is the context (sentence) without the target mention and the output is the detector prediction for gender of that mention. For a mention i, let C_b^i be the context before the mention, C_a^i be the context after the mention and f_θ be the gender detector. Then,

$$p_i = f_\theta(C_a^i, C_b^i) \tag{1}$$

is the detector's probability (confidence) that the mention i is female, i.e. p_i close to 0 indicates high confidence for predicting male while close to 1 indicates high confidence for females.

We use the detector's probability of the *true* gender of the mention (g_i) as an estimate of how gendered the text is: a high probability indicates that gender is heavily reflected in the context. We define this as the gendered score, given by G_m^i here:

$$G_m^i = \begin{cases} p_i & \text{True gender } g_i \text{ is female} \\ 1 - p_i & \text{True gender } g_i \text{ is male} \end{cases} \tag{2}$$

We define gendered score for a document as the average of gender score for all of its mentions. i.e.

$$G_{\text{doc}} = \frac{1}{N} \sum_{i=1}^{N} G_m^i \tag{3}$$

where N is the number of person mentions.

---

[1] in the machine learning sense of the word

Dataset	Male Mentions	Female Mentions
Reviews	298,580	104,632
Summaries	405,368	186,626
News	19,012,473	3,902,510
Novels	18,433,400	6,982,348

Table 1: **Dataset details.** Number of male and female mentions in the different datasets are shown.

As detector, rather than relying on frequency-based linear models or Naive Bayes model, we can use simple as well as more complex classifiers that can accurately model semantics and syntax of the context. For example, we use bag-of-words models (with logistic regression classifier) and recurrent neural network models as described in Section 4.2.

4 Datasets and Classifiers

In this section, we give details about the datasets, our pipeline for automated data labeling, filtering and processing applied to contexts in order to remove obvious gender information, and the classifiers used to classify gender of a given mention.

4.1 Datasets and data processing

To illustrate the utility of our proposed approach, we analyze text from four different domains:

- New York Times articles from the Annotated Gigaword corpus (Napoles et al., 2012)
- Novels from Gutenberg corpus [2]
- IMDB Movie Reviews (Maas et al., 2011)
- Movie Summaries (Bamman et al., 2013)

These domains cover a variety of writing styles. While the novels represent fictional writing, news articles are non-fictional. Movie summaries dataset describes the plot of the movies, i.e., how gender is represented in the plots, and the movie reviews dataset provides the ways in which people express their views on the plots, i.e., how gender is represented in user perception of the movie.

We train classifiers for each domain to predict the gender of mentions from the context they appear in, and use the resulting classifiers to detect gendered language. Similar idea is explored in Choi et al. (2018) to detect the type of mention from the context. For news, data from the first 6 months for every year is used for training, next three for validation, and last three for testing. For novels and movie summaries, we divide the data randomly into $50:20:30$ split for train, validation and test data. For movie reviews, we use the predefined split for train and test data, further dividing the training data into training and validation data.

1 Miss Mary Briganza will go to Korea with her parents.
2 Miss <*female*> will go to Korea with her parents.
3 <*Title*> <*female*> will go to Korea with <*their*> parents.
4 <*Title*> ____ will go to Korea with <*Their*> parents.

Figure 3: **Annotation pipeline.** We show the annotation and data processing pipeline step-by-step. **1** is the original sentence. **2** is the sentence after detecting positions of all mentions in the sentence, and replacing them with a placeholder for gender of that mention. **3** is the sentence after removing obvious gender information such as replacing gendered-pronouns (*he*, *she*) with a gender-neutral pronoun (*them*), and titles *Mr., Mrs., Miss* with a placeholder title word. **4** is the final input to our classifier where ____ is the position of mention for which classifier needs to predict gender (male or female).

Labeling Gender for Mentions We illustrate our processing pipeline via the example sentence in Figure 3. For mention-level gender prediction, we need a dataset with identified person mentions and their genders. Since we do not have labeled data, we need to identify mentions in contexts, and assign gender labels to them. Along with pronouns 'he' and 'she', we use spacy[3] to tag all corpora with NER tags to identify the set of person mentions. We use the SSN baby names dataset[4] from 1880 to 2016 to assign gender to each name. If a name is associated with more than one sex, we exclude it if it is ambiguous (being less than 4 times more frequent for one sex), but otherwise assign it to the more frequent sex. If a name is absent from our list of names, we replace the mention with a placeholder <*Person*>. Table 1 shows the count of male and female mention-context pairs generated using this pipeline. Processed sentence after this step is sentence 2 in Figure 3.

Filtering and Input Context Processing To remove obvious, uninteresting gender information, we discard sentences that contain any word from a gender-specific lexicon as used by Bolukbasi et al. (2016) such as gender-specific occupation words and gender-specific familial relation words, e.g., 'man', 'woman', 'prince', and 'hostess'. Complete list is given in Appendix A. For contexts that contain gender-indicative pronouns ('him', 'her', 'his',

---

[2] Project Gutenberg, from www.gutenberg.org

[3] https://spacy.io

[4] https://www.ssa.gov/oact/babynames/

'hers', 'himself', 'herself'), we replace them with a gender-neutral pronoun (*'them', 'their'*). All other mentions in the context (including 'he' and 'she') are replaced with a gender neutral word, and titles (*'Mr', 'Mrs', 'Miss'*) are replaced with a gender-neutral title word. Sentence 3 in Figure 3 is the result after this stage.

4.2 Classifier Details

The input to classifier is sentence 4 in Figure 3 and the target is 1 (for female). We extract such mention-context pairs from large text corpora to train classifiers that can predict the gender of individual mentions from their context using minimal manual supervision (as illustrated in Figure 2).

Bag-of-words and ngrams We construct bag-of-word classifiers by selecting the $50,000$ most frequent words from the training subset, and bag-of-ngrams models by selecting the $100,000$ most frequent n-grams (up to 3-grams), for each dataset. We explore a number of classifiers like logistic regression, support vector machines, random forest classifier, and choose logistic regression classifier since it consistently performs better than others.

LSTMs and CNNs We use both uni- and bidirectional LSTM recurrent neural networks for the context. In the 2-way LSTM model, we use two separate LSTMs: one for context before the mention, and the other for context after the mention. The direction of LSTM for latter part is reversed so that the model gives more importance to words closer to the target mention. This is followed by a sigmoid layer after the concatenation of the final hidden states. The input layers are initialized using the Glove vectors (Pennington et al., 2014), and are updated during training. We train the classifier with log-loss and Adam (Kingma and Ba, 2014) optimization algorithm, including dropout (Srivastava et al., 2014) and early stopping for regularization. Hyper-parameters are tuned for different domains separately. We experiment with ELMo embeddings (Peters et al., 2018), convolutional neural network (CNN)-based architectures, vanilla recurrent neural network (RNN), and gated recurrent unit (GRU) (Cho et al., 2014) models as well.

Performance Since our datasets are imbalanced, we use AUC-ROC as a performance metric. Table 2 shows AUC-ROCs for various models for all the datasets. Conventional bag-of-word/ngrams classifiers exhibit AUC-ROCs comparable to more complex LSTM and CNN classifiers. We use 2-way LSTM as the classifier for our final analysis.

	Reviews	**Summaries**	**News**	**Novels**
Bag-of-ngrams	0.64	0.62	**0.70**	**0.71**
Bag-of-word	0.63	0.62	**0.70**	**0.71**
Single LSTM	**0.67**	0.62	0.63	0.63
2-way LSTM	**0.67**	**0.66**	0.68	0.67
2-way LSTM + ELMo	**0.67**	0.65	**0.70**	0.69
2-way RNN	0.65	0.63	0.65	0.63
2-way GRU	**0.67**	**0.66**	0.69	0.66
CNN	0.66	0.64	0.68	0.64

Table 2: **AUC-ROCs for different models (evaluated on test data).**

5 Human Annotation Evaluation

To assess the difficulty of this task and to compare performance of our gendered language detector against a human baseline, we use Amazon Mechanical Turk to get human annotations for 500 random sentences from the test sets of each domain.

Task Description Turkers are shown sentences with missing mention, e.g., *'Sandwich maker ____ said mojo and fresh roasted pork are key to a great Cuban sandwich'*, and are asked to guess the gender of the missing mention. We sample the sentences such that the true labels (male/female) are balanced. For our study, we use two tasks that slightly differ from one another in the decisions turkers need to make. In one task, turkers are given only two options, *male* and *female*, forcing them to make a choice. In the second task, turkers are given five options on the Likert scale: *extremely likely male, likely male, neutral, likely female, extremely likely female* allowing for a finer scale of decision. We include examples in the instructions, and a few extremely easy examples as probes to verify quality (Munro et al., 2010). Each worker is shown 35 sentences from a single domain. On average, we collect 7 human annotations per sentence.

Do humans predict gender well? Sentences that do not have a clear majority are removed from our analysis. As a measure of inter-rater reliability, we compute pairwise and majority agreement, in Table 3. Percentage improvement over chance agreement is higher for 5-scale rating compared to 2-scale rating indicating that users tend to agree more when they are able to tag the borderline (possibly confusing) mentions as gender-neutral (chance agreement is 0.5 for 2-scale, and 0.2 for

Dataset	Pairwise		Majority	
	2-Scale	5-Scale	2-Scale	5-Scale
Reviews	0.62	0.32	0.74	0.52
News	0.65	0.38	0.77	0.55
Novels	0.60	0.33	0.73	0.52
Summaries	0.61	0.33	0.73	0.53
Combined	0.62	0.35	0.74	0.53

Table 3: **Pairwise and majority inter-annotator agreement for instances with clear majority.** 2-Scale indicates when users are asked to indicate male or female, while 5-Scale indicates gender on a scale of 5.

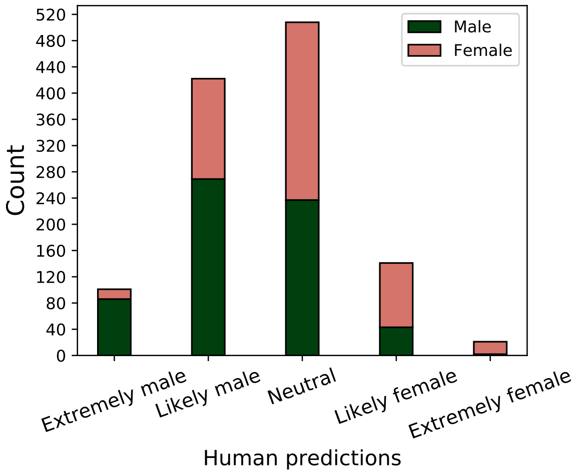


Figure 4: **Comparing Human Predictions to Truth.** x-axis represents the human prediction for mentions, while the green and pink bars represent counts of true male and female mentions respectively.

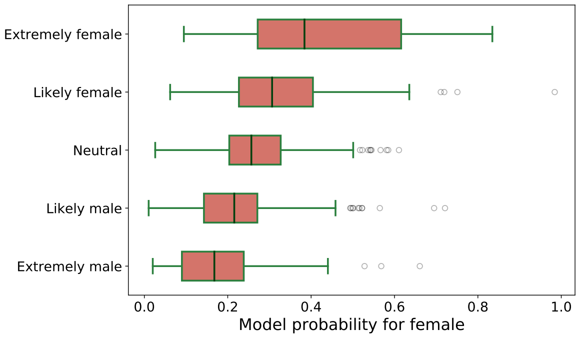


Figure 5: **Human and Model Predictions.** x-axis is the classifier probability where higher probability indicates female prediction. The points outside the range indicate outliers. Right shift of green line (representing median values) as we move from extremely male predictions to extremely female predictions corroborates the agreement between humans and classifiers.

5-scale task). To analyze the kind of mistakes humans make, we show the true distribution of male and female mentions compared against human predictions in Figure 4. 42% of examples are predicted *'Neutral'* by humans showing that the task is pretty difficult for humans as they often find mention gender ambiguous. Further, *'Likely male'* and *'Likely female'* categories have around 30% wrong predictions as well. These high error rates explain low F1 of 0.52 for human annotations. *'Extremely male'* and *'Extremely female'* have the least error rates showing that humans are more precise when they are more confident about the predictions.

Does our model match humans? In order to compare human annotations against model predictions more concretely, we choose to use Kendall's τ_c statistic (Berry et al., 2009), because it allows us to compare two variables when their underlying scales have different numbers of values. Like correlation coeeficients, τ_c ranges from -1 (fully negative association) to +1 (fully positive association). τ_c between humans and our LSTM model predictions vary from 0.23 to 0.36 showing positive correlation. We also look at classifier probability distribution for human decisions shown in box and whisker plot in Figure 5, where x-axis is the classifier probability of the mention being female. The median value of classifier prediction for each category (shown in green line) shifts towards female prediction as we move from 'Extremely male' to 'Extremely female' category, corroborating the agreement between humans and model.

6 Analysing Gendered Language

We first show aggregate word level and phrase level analysis, then show more complex and subtle sources of gendered language on sentence level.

Word-level Analysis Table 4 shows the top nouns and verbs extracted using bag-of-word classifier. We also train separate classifiers only on nouns, on adjectives, and on verbs in order to find out which are most informative for gender. Classifiers trained only on nouns performs best, indicating that nouns have most information. Top male-indicative nouns stem from typically male-dominated sports, while top female-indicative nouns are related to fashion and home industry.

Phrase-level Analysis Table 5 shows some of the top phrases for predicting males and females for different domains extracted using bag-of-ngrams models. We see phrases like *'clasped their hands'*, *'fashion show'*, *'jealous rage'*, *'was asked to'* for females, while *'clockwork orange'*, *'action hero'*,

Female-specific Nouns	**Male-specific Nouns**
Summaries: cherie, elisabet, crawlers, plastics, governess, cheerleader, prostitution, overdosing, bimbo, spinner	**Summaries:** quacker, platoon, tweety, shemp, cellmate, ham, nibbles, falstaff, pup, towel, mousehole, bullies
Novels: godmother, melvina, skirt, girlhood, lucile, womanly, eyebright, womanhood, shawl, dressmaker, demurely	**Novels:** disciples, yussuf, rifle, jr, pepe, cigar, colleague, followers, erasmus, judas, opponents
Reviews: comedienne, floriane, slut, adela, tch, topless, actress, tits, feminist, modeling, redhead, helen, vamp, bettie	**Reviews:** seagal, inventor, panther, opponent, sellers, ratso, comedian, lawman, yossi, creators, brutus, ted
News: gymnasts, dietitian, lpga, hingis, feminist, dowd, sorenstam, wie, receptionist, omnimedia, quilting, homemaker	**News:** spurs, astros, nicks, jets, sprewell, nets, vikings, clippers, lakers, holyfield, sonics, councilman, nba, bucs, pitches
Female-specific Verbs	**Male-specific Verbs**
Summaries: giggles, conceive, type, spurned, distorted, strokes, railing, rehearse, gag, disowned, plaguing, forgo	**Summaries:** commanding, barks, crack, credited, embezzled, executes, opposing, foils, relying, assassinate, engineered
Novels: sobbed, sew, blushed, wailed, pouted, scream, moaned, giggled, weeping, blushing, sob, shrieked, faltered	**Novels:** preached, elected, growled, states, yelled, roared, nominated, voted, grinned, slew, preach, fire, attack
Reviews: swims, bare, willed, raped, married, pouting, pleading, glows, kisses, liberated, seduces, fled, numbed	**Reviews:** direct, assuming, elected, defeat, casted, laid, mumbles, rule, directing, flicks, drinking, produce
News: fax, widowed, choreograph, raped, graduates, decorating, sobbing, majoring, giggling, married, cries, decorate	**News:** coach, pitching, batted, disarm, sacked, benched, fumbled, lightning, averaged, traded, sprained, vetoed

Table 4: Most important nouns and verbs for predicting male/female.

	News	**Reviews**	**Summaries**	**Novels**
Female gendered phrases	was pregnant	femme fetale	fashion show	suffrage association
	administrative assistant	nude scenes	jealous rage	clasped *their* hands
	their baby	strong willed	same way that	glass eye
	staff to vice	pop star	was asked to	corresponding secretary
	social worker	is hot	unborn child	dressing room
Male gendered phrases	defense secretary	action hero	construction worker	lieut col
	their locker	*their* screenplay	of the next	*their* pipe
	super tuesday	clockwork orange	school bully	*their* rifle
	offensive coordinator	court martial	Vietnam war	old fellow
	majority leader	nothing like the	make living	jimmy skunk

Table 5: **Most important phrases for predicting male/female.** *'their'* represents gender-neutral pronoun.

'construction worker' occur for males. Similarly, the term *'secretary'* occurs frequently with females, however phrases like *'defense secretary', 'treasury secretary'* are positive features for male mentions indicating male-domination of certain fields.

Sentence-level Analysis Our approach is the first to find gendered sentences from a huge corpora. We present examples of detected gendered language in Table 6; the first two columns show contexts for which a high level of gendering is estimated (most confident estimates), while the classifier has very low confidence for the examples in third column, indicating gender-neutral use of language. We see several interesting examples, e.g., male-gendered contexts from summaries show that society attributes roles like billionaire computer moguls and FBI-agents to males. The first female gendered example from novels depicts the way in which females are described and portrayed in fiction, which is in stark contrast to male descriptions.

Corpus-level Analysis Our approach allows us to automatically analyze large corpus of text, enabling high-level analysis of what documents are most, or least, gendered. Table 7 shows such an analysis, where the languages for movie reviews and summaries, genres for novels, and news desks for news are organized by their estimated genderedness. We see that children's history books are more gendered than their literature books. Books related to opera and one-act plays are among the least gendered ones, while those related to war, history, and philosophy use gendered language the most. Movie reviews for movies in Vietnamese, Turkish and Polish are among the most gendered, while Greek and Japanese are the least gendered. We see a similar pattern in movie summaries - summaries for movies filmed in Polish and Turkish are more gendered than for movies in Korean or Romanian. *Sports* is the most gendered category for news articles, while *Cultural*, *Leisure*, *Society*, and *Home* are among the least gendered ones. The table also contains some unexpected predictions, such as the low gendering of *Girls* and *Young women* novels.

	Female gendered	Male gendered	Gender neutral
News	– J.J.'s research brings *them* to find *Person*, a farmer who is slowly devouring a 747 to win the heart of lovely 🟥 , editor of the local newspaper. – It's a film about *Person* directed by *Person*, and I play the soprano 🟥 .	– USC is expected to announce that former New England patriots coach 🟥 will be its next football coach. – It has fired up a torrid debate over President-elect 🟥 's $ 1.3 trillion tax cut proposal.	– *Person* had props to bolster *their* story, 🟥 added. – In North Dakota, *Person* and 🟥 - who are married to each other - were running against each other for foster county prosecutor.
Novels	– *Person* looked untidier than ever; 🟥 wore a slatternly wrapper, and *their* hair was thrust unbrushed into its net. –"What is it?" asked *Person*, as 🟥 folded and smoothed *their* best gown.	– If the collector will remember that, though 🟥 is the present owner of *their* prints... – *Person* is not an orator; *person* is not a writer; 🟥 is not a thinker.	– 🟥 got up and went so – so unexpectedly. – The hand of the child admitted *them* to the chamber of death; the door closed, and 🟥 stood motionless.
Reviews	– Lake's secretary, 🟥 , is *Person*'s sweetheart . – Aeon played by the lovely 🟥 in this adaptation, is dexterous as a line-dancer and deadly as a viper-snake.	– Herein, only old-time Broadway producer 🟥 and *their* fey secretary *Person* maintain interest. – 🟥 's rolling in the sheets with *their* beautiful secretary Meredith.boris.	– 🟥 's hysterical and so are *their* backup singers. – 🟥 is the only one worth seeing in this film , but *person* doesn't get to do much.
Summaries	– 🟥 is raped by the estate owner, who then writes off *person*'s debt. – *Person*, an architect, is married to *their* sweetheart 🟥 with two children.	– *Person* befriends a billionaire computer mogul 🟥 and a cafe waitress.. – FBI-agent 🟥 becomes an unwitting pawn of the white hand drug cartel.	– *Person* goes over to check on *them*, insisting *person* reopen the blinds, but 🟥 denies doing it. – In order to stifle *their* theatrical aspirations, 🟥 arranges a screen test.

Table 6: **Examples of gendered and non-gendered mention-context pairs from different domains.** *them/their* represents a gender-pronoun replaced with a gender neutral pronoun. *Person* represents mention. Note that in the first example for news (in female-gendered column), *J.J.* has not been identified as a mention. This is a weakness in the preprocessing.

Reviews Language	Summaries Language	Novels Genre	News News Desk
Vietnamese	Serbian	US Civil War	Sports
Turkish	Arabic	US Politics	Foreign
Polish	Thai	Mathematics	Financial
Cantonese	Czech	Military Science	Week in Review
Arabic	Polish	Evolution	National
Mandarin	Turkish	Dictionaries	Business
Korean	Khmer	Girls	Living
Latin	Korean	Young women	Travel
Portuguese	Sinhala	Sisters	Society
French	Hungarian	Family Life	Home
Greek	Punjabi	Marriage	Style
Japanese	Romanian	Italy	Performing Arts

Table 7: **High-level Analysis of Corpora.** Most and least gendered languages for movie reviews and summaries; genre for novels; news desk for news articles.

7 Discussion and Future Work

Sex vs Gender Since the current English language use is mostly limited to binary gender identities (both due to grammar and usage), we treat gender as a binary concept in this work. Inclusion of genderqueer and non-binary identities will require data annotated by humans with sufficient domain knowledge, which was out of scope for this work. We assume that mentions for which our labeling has associated the wrong 'gender' because of difference in sex/gender identities are sufficiently low in proportion that model is still able to learn relevant signals when trained on large corpora.

Facts vs Stereotypes In this work, we do not delineate between factual information (women get *pregnant*) and the intentional use of stereotypes (women are *sweethearts*). In some domains, such as news, ignoring this difference can be misleading, and exploring approaches that are able to better separate these different biases is important.

Extension to New Domains There remain a number of exciting avenues for future work. Although we analyze a variety of domains that differ from each other, our analysis focused on independently investigating each; it may be much more fruitful to compare and contrast the gendered language across multiple domains. When extending this work to other domains like Twitter, blogs, etc., the performance of the system can be affected by various factors like accuracy of NER system for the domain (e.g., it would be lower for tweets) and names to gender mapping (which can vary for different geographies and cultures).

8 Conclusions

We present a concrete implementation and evaluation of our gendered language detector. The main advantages of our pipeline and method are: (1) *Flexibility*, in application to different domains with minimal manual intervention, (2) *Mention-level* analysis, instead of article-level analysis in previous works, enabling more granular analysis, and (3) *Quantitative measure* of the extent of genderedness of context given a mention, allowing large-scale and detailed analyses and comparisons.

Our pipeline automatically extracts person mentions from a corpus, and by using an accurate gender predictor, trains a classifier to learn the ways in which language is gendered *for that corpus*. This automation provides multiple benefits; not only are there no humans in the loop to inject their biases about what is, and is not, gendered language, but further, collection of a large annotated corpus allows us to train sophisticated neural models that are able to capture semantic and syntactic constructions in the language. Evaluation suggests that our model is fairly accurate on this challenging task, and further, allows us to carry out analysis on multiple domains at varying levels of granularity, demonstrating potential applications of this work. The code to support such endeavours, and to reproduce the results, is available at https://ucinlp.github.io/GenderQuant.

Acknowledgements

We would like to thank Dheeru Dua, Matt Gardner, Robert L. Logan IV, and the reviewers for their feedback and suggestions. This work is supported in part by Allen Institute for Artificial Intelligence (AI2) and in part by NSF award #IIS-1756023.

References

Omar Ali, Ilias Flaounas, Tijl De Bie, Nick Mosdell, Justin Lewis, and Nello Cristianini. 2010. Automating news content analysis: An application to gender bias and readability. In *Workshop on Applications of Pattern Analysis*.

David Bamman, Brendan O'Connor, and Noah A Smith. 2013. Learning latent personas of film characters. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kenneth J Berry, Janis E Johnston, Sammy Zahran, and Paul W Mielke. 2009. Stuarts tau measure of effect size for ordinal variables: Some methodological considerations. *Behavior Research Methods*, 41(4).

Ben Blatt. 2017. *Nabokov's favorite word is mauve*. Simon & Schuster.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Anne Boring, Kellie Ottoboni, and Philip B Stark. 2016. Student evaluations of teaching (mostly) do not measure teaching effectiveness. *ScienceOpen Research*.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334).

John A Centra and Noreen B Gaubatz. 2000. Is there gender bias in student evaluations of teaching? *The Journal of Higher Education*, 71(1).

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Annual Meeting of the Association for Computational Linguistic (ACL)*.

Cody Cottier. 2018. From mouth to mind: How language governs our perceptions of gender.

Jeffrey Dastin. 2018. Amazon scraps secret ai recruiting tool that showed bias against women.

Susan Tyler Eastman and Andrew C Billings. 2000. Sportscasting and sports reporting: The power of gender bias. *Journal of Sport and Social Issues*, 24(2).

Andrea Eidinger. 2017. She's hot: Female sessional instructors, gender bias, and student evaluations.

Liye Fu, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Tie-breaker: Using language models to quantify gender bias in sports journalism. In *IJCAI Workshop on Natural Language Processing meets Journalism*.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2018. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16).

Abdullah Gharbavi and Seyyed Ahmad Mousavi. 2012. The application of functional linguistics in exposing gender bias in iranian high school english textbooks. *English Language and Literature Studies*, 2(1).

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Susan C Herring and John C Paolillo. 2006. Gender and genre variation in weblogs. *Journal of Sociolinguistics*, 10(4).

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Katherine N Kinnick. 1998. Gender bias in newspaper profiles of 1996 olympic athletes: A content analysis of five major dailies. *Women's Studies in Communication*, 21(2).

Corina Koolen and Andreas van Cranenburgh. 2017. These are not the stereotypes you are looking for: Bias and fairness in authorial gender attribution. In *ACL Workshop on Ethics in Natural Language Processing*.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4).

María E Len-Ríos, Shelly Rodgers, Esther Thorson, and Doyle Yoon. 2005. Representation of women in news and photos: Comparing content to perceptions. *Journal of Communication*, 55(1).

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

Monica Macaulay and Colleen Brice. 1997. Don't touch my projectile: Gender bias and stereotyping in syntactic examples. *Language*.

Lillian MacNell, Adam Driscoll, and Andrea N Hunt. 2015. What's in a name: Exposing gender bias in student ratings of teaching. *Innovative Higher Education*, 40(4).

Michela Menegatti and Monica Rubini. 2017. Gender bias and sexism in language. *Oxford Research Encyclopedia of Communication*.

Janice Moulton, George M Robinson, and Cherin Elias. 1978. Sex bias in language use: Neutral pronouns that aren't. *American Psychologist*, 33(11).

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Robert Munro, Steven Bethard, Victor Kuperman, Vicky Tzuyin Lai, Robin Melnick, Christopher Potts, Tyler Schnoebelen, and Harry Tily. 2010. Crowdsourcing and language studies: The new generation of linguistic data. In *NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX.

Marcus Otlowski. 2003. Ethnic diversity and gender bias in efl textbooks. *Asian EFL Journal*, 5(2).

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jemma Prior. 2017. Teachers, what is gendered language?

Anil Ramakrishna, Victor R Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. 2017. Linguistic analysis of differences in portrayal of movie characters. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *ACL Workshop on Ethics in Natural Language Processing*.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Maarten Sap, Marcella Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: tracing stylometric evidence beyond topic and genre. In *Computational Natural Language Learning (CoNLL)*.

Toni Schmader, Jessica Whitehead, and Vicki H Wysocki. 2007. A linguistic comparison of letters of recommendation for male and female chemistry and biochemistry job applicants. *Sex Roles*, 57(7-8).

Alexandra Schofield and Leo Mehr. 2016. Gender-distinguishing features in film dialogue. In *Workshop on Computational Linguistics for Literature*.

Kevin B Smith. 1997. When all's fair: Signs of parity in media coverage of female candidates. *Political Communication*, 14(1).

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1).

Jane G. Stout and Nilanjana Dasgupta. 2011. When he doesn't mean you: Gender-exclusive language as ostracism.

Rachael Tatman. 2017. Gender and dialect bias in youtube's automatic captions. In *ACL Workshop on Ethics in Natural Language Processing*.

Mike Thelwall. 2018. Gender bias in machine learning for sentiment analysis. *Online Information Review*, 42(3).

Frances Trix and Carolyn Psenka. 2003. Exploring the color of glass: Letters of recommendation for female and male medical faculty. *Discourse & Society*, 14(2).

Susan Tyler Eastman, Andrew C. Billings. 2001. Biased voices of sports: Racial and gender stereotyping in college basketball announcing. *Howard Journal of Communication*, 12(4).

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguous pronouns. In *Transactions of the Association for Computational Linguistics (TACL)*.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a thousand points of) light on biased language. In *NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias smplification using corpus-level constraints. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. Gender bias in coreference resolution: Evaluation and debiasing methods. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. Learning gender-neutral word embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A Set of Gender-Specific Words

List from Bolukbasi et al. (2016) without pronouns: *man, women, men, woman, spokesman, wife, son, mother, father, chairman, daughter, husband, guy, girls, girl, boy, boys, brother, spokeswoman, female, sister, male, herself, brothers, dad, actress, mom, sons, girlfriend, daughters, lady, boyfriend, sisters, mothers, king, businessman, grandmother, grandfather, deer, ladies, uncle, males, congressman, grandson, bull, queen, businessmen, wives, widow, nephew, bride, females, aunt, prostate cancer, lesbian, chairwoman, fathers, moms, maiden, granddaughter, younger brother, lads, lion, gentleman, fraternity, bachelor, niece, bulls, husbands, prince, colt, salesman, dude, beard, filly, princess, lesbians, councilman, actresses, gentlemen, stepfather, monks, ex girlfriend, lad, sperm, testosterone, nephews, maid, daddy, mare, fiance, fiancee, kings, dads, waitress, maternal, heroine, nieces, girlfriends, sir, stud, mistress, lions, estranged wife, womb, grandma, maternity, estrogen, ex boyfriend, widows, gelding, diva, teenage girls, nuns, czar, ovarian cancer, countrymen, teenage girl, penis, bloke, nun, brides, housewife, spokesmen, suitors, menopause, monastery, motherhood, brethren, stepmother, prostate, hostess, twin brother, schoolboy, brotherhood, fillies, stepson, congresswoman, uncles, witch, monk, viagra, paternity, suitor, sorority, macho, businesswoman, eldest son, gal, statesman, schoolgirl, fathered, goddess, hubby, stepdaughter, blokes, dudes, strongman, uterus, grandsons, studs, mama, godfather, hens, hen, mommy, estranged husband, elder brother, boyhood, baritone, grandmothers, grandpa, boyfriends, feminism, countryman, stallion, heiress, queens, witches, aunts, semen, fella, granddaughters, chap, widower, salesmen, convent, vagina, beau, beards, handyman, twin sister, maids, gals, housewives, horsemen, obstetrics, fatherhood, councilwoman, princes, matriarch, colts, ma, fraternities, pa, fellas, councilmen, dowry, barbershop, fraternal, ballerina.*

Analyzing Polarization in Social Media: Method and Application to Tweets on 21 Mass Shootings

Dorottya Demszky[1] **Nikhil Garg**[1] **Rob Voigt**[1] **James Zou**[1]
Matthew Gentzkow[1] **Jesse Shapiro**[2] **Dan Jurafsky**[1]
[1]Stanford University [2]Brown University
`{ddemszky, nkgarg, robvoigt, jamesz, gentzkow, jurafsky}@stanford.edu`
`jesse_shapiro_1@brown.edu`

Abstract

We provide an NLP framework to uncover four linguistic dimensions of political polarization in social media: topic choice, framing, affect and illocutionary force. We quantify these aspects with existing lexical methods, and propose clustering of tweet embeddings as a means to identify salient topics for analysis across events; human evaluations show that our approach generates more cohesive topics than traditional LDA-based models. We apply our methods to study 4.4M tweets on 21 mass shootings. We provide evidence that the discussion of these events is highly polarized politically and that this polarization is primarily driven by partisan differences in framing rather than topic choice. We identify framing devices, such as grounding and the contrasting use of the terms "terrorist" and "crazy", that contribute to polarization. Results pertaining to topic choice, affect and illocutionary force suggest that Republicans focus more on the shooter and event-specific facts (news) while Democrats focus more on the victims and call for policy changes. Our work contributes to a deeper understanding of the way group divisions manifest in language and to computational methods for studying them.[1]

1 Introduction

Elites, political parties, and the media in the US are increasingly polarized (Layman et al., 2010; Prior, 2013; Gentzkow et al., forthcoming), and the propagation of partisan frames can influence public opinion (Chong and Druckman, 2007) and party identification (Fiorina and Abrams, 2008).

Americans increasingly get their news from internet-based sources (Mitchell et al., 2016), and political information-sharing is highly ideologically segregated on platforms like Twitter (Conover et al., 2011; Halberstam and Knight, 2016) and Facebook (Bakshy et al., 2015). Prior NLP work has shown, e.g., that polarized messages are more likely to be shared (Zafar et al., 2016) and that certain topics are more polarizing (Balasubramanyan et al., 2012); however, we lack a more broad understanding of the many ways that polarization can be instantiated linguistically.

This work builds a more comprehensive framework for studying linguistic aspects of polarization in social media, by looking at topic choice, framing, affect, and illocutionary force.

1.1 Mass Shootings

We explore these aspects of polarization by studying a sample of more than 4.4M tweets about 21 mass shooting events, analyzing polarization within and across events.

Framing and polarization in the context of mass shootings is well-studied, though much of the literature studies the role of media (Chyi and McCombs, 2004; Schildkraut and Elsass, 2016) and politicians (Johnson et al., 2017). Several works find that frames have changed over time and between such events (Muschert and Carr, 2006; Schildkraut and Muschert, 2014), and that frames influence opinions on gun policies (HaiderMarkel and Joslyn, 2001). Prior NLP work in this area has considered how to extract factual information on gun violence from news (Pavlick et al., 2016) as well as quantify stance and public opinion on Twitter (Benton et al., 2016) and across the web (Ayers et al., 2016); here we advance NLP approaches to the public discourse surrounding gun violence by introducing methods to analyze other linguistic manifestations of polarization.

1.2 The Role of the Shooter's Race

We are particularly interested in the role of the shooter's race in shaping polarized responses to these events. Implicit or explicit racial biases can be central in people's understanding of social

---

[1] All data and code is available at: `https://github.com/ddemszky/framing-twitter`

Proceedings of NAACL-HLT 2019, pages 2970–3005
Minneapolis, Minnesota, June 2 - June 7, 2019.

problems (Drakulich, 2015); in the mass shooting context, race is a factor in an event's newsworthiness (Schildkraut et al., 2018) and is often mentioned prominently in media coverage, particularly when the shooter is non-white (Mingus and Zopf, 2010; Park et al., 2012). Duxbury et al. (2018) find that media representations of white shooters disproportionately divert blame by framing them as mentally ill while representations of non-white shooters are more frequently criminalized, highlighting histories of violent behavior.

The important question remains as to how polarized ideologies surrounding race take shape on forums such as Twitter. Therefore, in all of the analyses throughout this paper we consider the race of the shooter as a potential factor. We note that in the 21 shooting events we study, shootings in schools and places of worship are overwhelmingly carried out by white perpetrators, so we cannot fully disentangle the effect of race from other factors.

2 Data: Tweets on Mass Shootings

Data collection. We compiled a list of mass shootings between 2015 and 2018 from the Gun Violence Archive.[2] For each, we identified a list of keywords representative of their location (see Appendix A). Given Twitter search API's limitations on past tweets, we retrieved data from a Stanford lab's archived intake of the Twitter firehose.[3]

For each event, we built a list of relevant tweets for the two weeks following the event. A tweet is relevant if it contained at least one of the event's location-based representative keywords and at least one lemma from the following list: "shoot", "gun", "kill", "attack", "massacre", "victim". We filtered out retweets and tweets from users who have since been deactivated. We kept those 21 events with more than 10,000 tweets remaining. For more details see Appendix A.

Partisan assignment. We estimate the party affiliation of users in the dataset from the political accounts they follow using a method similar to that of Volkova et al. (2014), which takes advantage of homophily in the following behavior of users on twitter (Halberstam and Knight, 2016). We compile a list of Twitter handles of US Congress members in 2017, the 2016 presidential and vice presidential candidates, and other party-affiliated pages.[4] We label a user as a Democrat if they followed more Democratic than Republican politicians in November 2017, and as a Republican if the reverse is true. For each event, 51–72% of users can be assigned partisanship in this way; to validate our method we compare state averages of these inferred partisan labels to state two-party vote shares, finding a high correlation (Figure 1).[5]

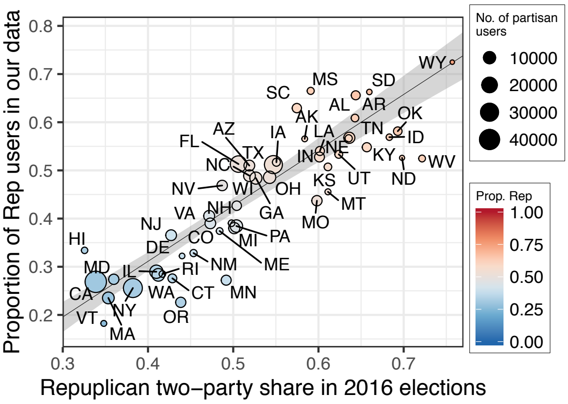


Figure 1: To validate our partisanship assignment, we compare the proportion of users we classify as Republican separately for each US state against the Republican share of the two-party vote. A weighted linear regression with the number of partisan users per state as weights obtains an adjusted R^2 of $.82$, indicating that the distribution of Republican users in our data and that of Republican voters across states is similar.

3 Quantifying Overall Polarization

We begin by quantifying polarization (equivalently, partisanship) between the language of users labeled Democrats and Republicans after mass shooting events. We establish that there is substantial polarization, and that the polarization increases over time within most events.

3.1 Methods

Pre-processing. We first build a vocabulary for each event as follows. Each vocabulary contains unigrams and bigrams that occur in a given event's tweets at least 50 times, counted after stemming via NLTK's SnowballStemmer and stopword removal.[6] We refer to these unigrams and bigrams collectively as *tokens*.

---

[2] `https://www.gunviolencearchive.org`

[3] With the exception of the two most recent shootings in Pittsburgh and Thousand Oaks, for which we collected tweets real time via the Twitter API.

[4] See Appendix B.1 for the complete list.

[5] We performed the sanity check for all partisan users with a valid US state as part of their geo-location (∼350k users).

[6] Stopword list is provided in Appendix A.1

Measure of partisanship. We apply the leave-out estimator of phrase partisanship from Gentzkow et al. (forthcoming). Partisanship is defined as the expected posterior probability that an observer with a neutral prior would assign to a tweeter's true party after observing a single token drawn at random from the tweets produced by the tweeter. If there is no difference in token usage between the two parties, then this probability is .5, i.e. we cannot guess the user's party any better after observing a token.

The leave-out estimator consistently estimates partisanship under the assumption that a user's tokens are drawn from a multinomial logit model. The estimator is robust to corpus size. The leave-out estimate of partisanship π^{LO} between Democrats $i \in D$ and Republicans $i \in R$ is

$$\pi^{LO} = \frac{1}{2}\left(\frac{1}{|D|}\sum_{i\in D}\hat{\mathbf{q}}_i \cdot \hat{\boldsymbol{\rho}}_{-i} + \frac{1}{|R|}\sum_{i\in R}\hat{\mathbf{q}}_i \cdot (1 - \hat{\boldsymbol{\rho}}_{-i})\right)$$

where $\hat{\mathbf{q}}_i = \mathbf{c}_i / m_i$ is the vector of empirical token frequencies for tweeter i, with $\mathbf{c}_i$ being the vector of token counts for tweeter i and m_i the sum of token counts for tweeter i; and $\hat{\boldsymbol{\rho}}_{-i} = (\hat{\mathbf{q}}^{D\setminus i} \oslash (\hat{\mathbf{q}}^{D\setminus i} + \hat{\mathbf{q}}^{R\setminus i}))$ is a vector of empirical posterior probabilities, excluding speaker i and any token that is not used by at least two speakers. Here we let $\oslash$ denote element-wise division and $\hat{\mathbf{q}}^{G} = \sum_{i\in G}\mathbf{c}_i / \sum_{i\in G} m_i$ denote the empirical token frequency of tweeters in group G. The estimator thus captures two intuitive components of partisanship: between-group difference (posterior probability for each feature), and within-group similarity (dot-product between the feature vector of each speaker and that of their group).

User-level measures. As the above leave-out estimator represents the average of user-level polarization values, we take the user-level dot product $(\hat{\mathbf{q}}_i \cdot \hat{\boldsymbol{\rho}}_{-i})$ as an estimate of the polarization of user i's language. We consider the correlation of this value and the number of politicians a user follows in total and from their preferred party.

3.2 Results and Discussion

Overall polarization. As Figure 2 shows, the discussion of each event[7] is highly polarized: values range from .517 to .547. For comparison, this measure, for most events, is similar to or higher than the polarization in the US congress ($\sim$ .53 in recent years) (Gentzkow et al., forthcoming). While we observe a slight increase in polarization over the past three years, this increase is not statistically significant ($p \approx .26$).

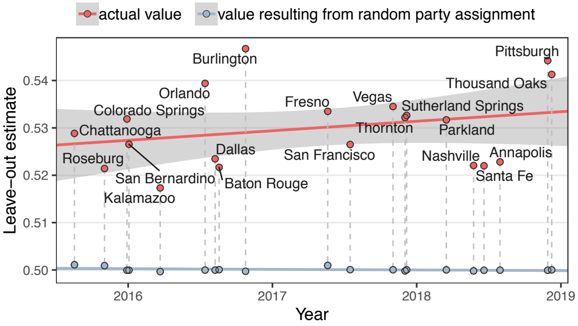


Figure 2: Tweets on mass shootings are highly polarized, as measured by the leave-out estimate of phrase partisanship (Gentzkow et al., forthcoming). The shaded region represents the 95% confidence interval of the linear regression fit to the actual values. To quantify noise, we also calculate the leave-out estimate after randomly assigning users to parties, matching the ratio of parties in the true data. The "values resulting from random assignment" are all close to .5, suggesting that the actual values are not a result of noise.

Post-event polarization. To see how polarization changes at the event level, we computed the leave-out estimate for each of the first 10 days following the events (see Figure 3). An event-day level regression of partisanship on days since the event suggests a slight increase in post-event polarization across events (slope = .002, $p < 0.05$). Fitting separate regressions, we find that the five events with the steepest increase in polarization are Burlington (slope = .03, $p < 0.05$), Orlando (slope = .006, $p < 0.001$), Las Vegas (slope = .003, $p < 0.001$), Chattanooga (slope = .003, $p < 0.05$) and Roseburg (slope = .003, $p < 0.05$) — note that except for Las Vegas, these shootings are all committed by a person of color.

Are the changes in the leave-out score due to *different* users tweeting at different times or due to the *same* users becoming more or less political? We found that while on average only $\sim 10\%$ of users tweeted on multiple days ($SD = 5\%$) across the events, these users contribute $\sim 28\%$ of the tweets ($SD = 15\%$). After removing these users from the leave-out estimation, we found that the temporal patterns remain with the same statistical significance, providing one piece of evidence

---

[7] For all the experiments using the leave-out estimator, we exclude Fort Lauderdale for which we only have tweets for the first day after the event; polarization is most dominant a few days after each event, making it incomparable. For reference, the leave-out estimate for Fort Lauderdale is .51.

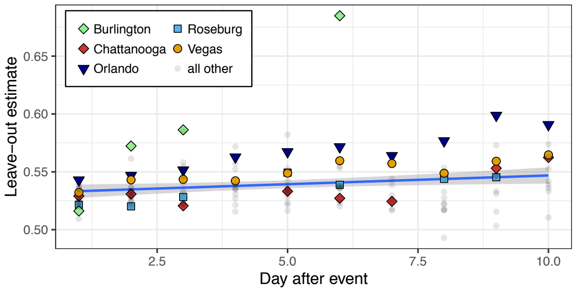


Figure 3: Leave-out estimate post-event.

that changes in polarization are not due to changes within users who tweet on multiple days.

User-level polarization. We estimated a linear regression of the leave-out score on the total number of followed politicians and the number from the user's preferred party, with controls for event indicators. The estimates imply that, fixing the total number of followed politicians, one more followed politician from one's preferred party is associated with an *increase* of .009 SD in the leave-out. Fixing the number of followed politicians from the user's preferred party, one more followed politician is associated with a *decrease* of .02 SD in the leave-out.

4 Topics and Framing

Topic choice can be a tool for agenda-setting by establishing what an author or institution deems worthy of discussion (McCombs, 2002), and works in NLP have used topic modeling as an approach to measure this effect (Tsur et al., 2015; Field et al., 2018). The strategy of highlighting particular aspects within topics as a means of framing (Entman, 2007) has also been quantified in the NLP literature (Boydstun et al., 2013; Card et al., 2015; Naderi and Hirst, 2017).

Previous work largely focuses on the relation between topic and framing in the news media; we study social media, proposing methods to identify general, non-event-specific topics and to quantify between- and within-topic polarization.

4.1 Methods

Topic assignment. Our goal is to induce topics that are salient in our narrow domain and comparable across events. This presents a challenge for traditional topic modeling approaches, since the discourse surrounding these events is inherently tied to concrete aspects of the events that tend to covary with topic usage, like location, setting, and demographics of the shooter and victims.

We build on the ability of vector space models to represent higher-level semantics to develop our own embedding-based topic assignment approach, comparing it with two traditional LDA-based methods: MALLET[8] and the Biterm Topic Model (BTM) (Yan et al., 2013); BTM was developed specifically for tweets. For all of these methods, we first randomly sample 10k tweets from each event forming our subset S of all tweets T; then, we create a vocabulary V of word stems that occur at least ten times in at least three events within S ($\sim$2000 word stems) and remove all stems from T are not part of V. Sampling is crucial for encouraging event-independent topics given the large disparity among event-level tweet counts (the largest event, Orlando, has 225× more tweets than the smallest event, Burlington).

For the embedding-based approach, we:

1. Train GloVe embeddings (Pennington et al., 2014) on V based on 11-50k random samples of tweets from each event.[9]
2. Create sentence embeddings $\mathbf{e}_t, \forall t \in T$ using Arora et al. (2017)'s method, by computing the weighted average $\mathbf{v}_t$ of the embeddings of stems within t and removing $\mathbf{v}_t$'s projection onto the first principal component of the matrix the rows of which are $\mathbf{v}_t, \forall t \in S$. Stem weights are set to be inversely proportional to their frequencies in S.
3. Jointly cluster the embeddings $\mathbf{e}_t, \forall t \in S$ via k-means using cosine distance and assign all tweet embeddings $\mathbf{e}_t, \forall t \in T$ to the centroids to which they are closest.

We also trained MALLET and BTM on S and used the resulting models to infer topics for all tweets in T, assigning each tweet to its highest probability topic. Henceforth, we use d to mean cosine distance for k-means and probabilities for MALLET and BTM.

A manual inspection found that about 25% of the tweets are either difficult to assign to any topic or they represent multiple topics equally. To filter out such tweets, for each tweet we looked at the ratio of d to its closest and second closest topic and removed tweets that have ratios higher than the 75th percentile (calculated at the model-level).[10]

---

[8] `http://mallet.cs.umass.edu/topics.php`

[9] This sample is different from S as it includes more tweets to increase data size, which is important for training the embeddings, where the slightly disproportional representation of events is less problematic.

[10] This procedure filters out 11-26% tweets (M=22%,

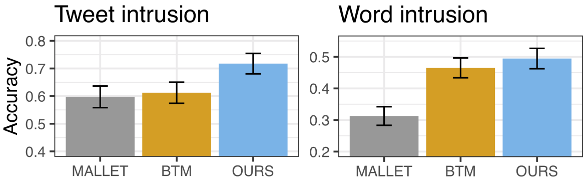


Figure 4: Topic model evaluations, collapsed across $k = 6 - 10$. Error bars denote standard errors.

Topic	10 Nearest Stems
news (19%)	break, custodi, #breakingnew, #updat, confirm, fatal, multipl, updat, unconfirm, sever
investigation (9%)	suspect, arrest, alleg, apprehend, custodi, charg, accus, prosecutor, #break, ap
shooter's identity & ideology (11%)	extremist, radic, racist, ideolog, label, rhetor, wing, blm, islamist, christian
victims & location (4%)	bar, thousand, california, calif, among, los, southern, veteran, angel, via
laws & policy (14%)	sensibl, regul, requir, access, abid, #gunreformnow, legisl, argument, allow, #guncontolnow
solidarity (13%)	affect, senseless, ach, heart, heartbroken, sadden, faculti, pray, #prayer, deepest
remembrance (6%)	honor, memori, tuesday, candlelight, flown, vigil, gather, observ, honour, capitol
other (23%)	dude, yeah, eat, huh, gonna, ain, shit, ass, damn, guess

Table 1: Our eight topics (with their average proportions across events) and nearest-neighbor stem embeddings to the cluster centroids. Topic names were manually assigned based on inspecting the tweets.

To compare the models, we ran two MTurk experiments: a word intrusion task (Chang et al., 2009) and our own, analogically defined tweet intrusion task, with the number of topics k ranging between 6-10. Turkers were presented with either a set of 6 words (for word intrusion) or a set of 4 tweets (for tweet intrusion), all except one of which was close (in terms of d) to a randomly chosen topic and one that was far from that topic but close to another topic. Then, Turkers were asked to pick the odd one out among the set of words / tweets. More details in Appendix D.

We find that our model outperforms the LDA-based methods with respect to both tasks, particularly tweet intrusion — see Figure 4. This suggests that our model both provides more cohesive topics at the word level and more cohesive groupings by topic assignment. The choice of k does not yield a significant difference among model-level accuracies. However, since $k = 8$ slightly outperforms other k-s in tweet intrusion, we use it for further analysis. See Table 1 for nearest neighbor stems to each topic and Appendix C.2 for example tweets.

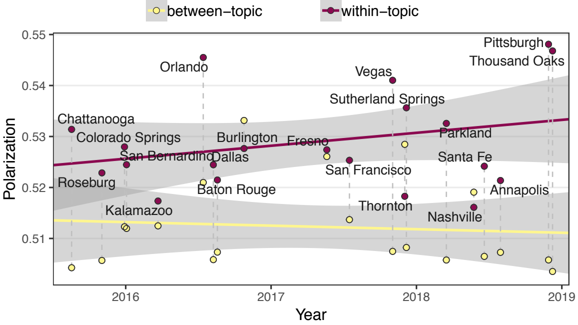


Figure 5: Measurements of between-topic and within-topic polarization of the 21 events in our dataset show that within-topic polarization is increasing over time while between-topic polarization remains stable.

Measuring within-topic and between-topic partisanship. Recall that the leave-out estimator from Section 3.1 provides a measure of partisanship. The information in a tweet, and thus partisanship, can be decomposed into *which* topic is discussed, and *how* it's discussed.

To measure *within-topic* partisanship for a particular event, i.e. how a user discusses a given topic, we re-apply the leave-out estimator. For each topic, we calculate the partisanship using only tweets categorized to that topic. Then, overall within-topic partisanship for the event is the weighted mean of these values, with weights given by the proportion of tweets categorized to each topic within each event.

Between-topic partisanship is defined as the expected posterior that an observer with a neutral prior would assign to a user's true party after learning only the topic — but not the words — of a user's tweet. We estimate this value by replacing each tweet with its assigned topic and applying the leave-out estimator to this data.

4.2 Results

Figure 5 shows that for most events within-topic is higher than between-topic partisanship, suggesting that while topic choice does play a role in phrase partisanship (its values are meaningfully higher than .5), within-topic phrase usage is significantly more polarized. Linear estimates of the relationship between within and between topic partisanship and time show that while within-topic polarization has increased over time, between-topic polarization has remained stable. This finding supports the idea that topic choice and topic-level framing are distinct phenomena.

Partisanship also differs by topic, and within days after a given event. Figure 6 shows polarization within topics for 9 days after Las Ve-

---

SD=4%) across events, for our model, for eight topics.

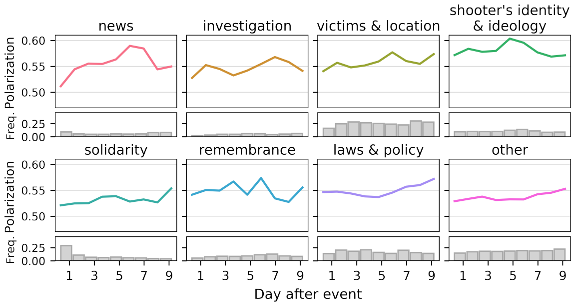


Figure 6: Las Vegas within-topic polarization in the days after the event. The bar charts show the proportion of each topic in the data at a given time.

gas. We find that *solidarity* has the lowest and *shooter's identity & ideology* the highest polarization throughout; polarization in most topics increases over time and *news* has the steepest increase. Similar patterns are present after Orlando (Figure 17 in Appendix I). Measuring polarization of topics for other events over time is noisy, given the sparsity of the data, but overall within-topic polarization is consistent: the most polarized topics on average across events are *shooter's identity & ideology* (.55) and *laws & policy* (.54), where people are apparently polarized about both why an event happened and what to do about it. Fact- and sympathy-based topics display less polarization: *news* (.51), *victims & location* (.52), *solidarity* (.52) and *remembrance* (.52).

As shown in Figure 7, *investigation*, *news*, and *shooter's identity & ideology* are more likely to be discussed by Republicans and *laws & policy* and *solidarity* more likely to be discussed by Democrats across events.[11] Topics preferred by Republicans seem to relate more to the shooter than to the victims, while topics preferred by Democrats seem to relate more closely to the victims. The shooter's race appears to play a role in topic preference: if the shooter is white, Democrats become more likely to focus on *shooter's identity & ideology* and *laws & policy* and Republicans on *news* and *investigation* than if the shooter is a person of color.

5 Specific Framing Devices

In the previous section, we show that topic-level framing and topic choice are different dimensions of polarization. We now look at the specific terms

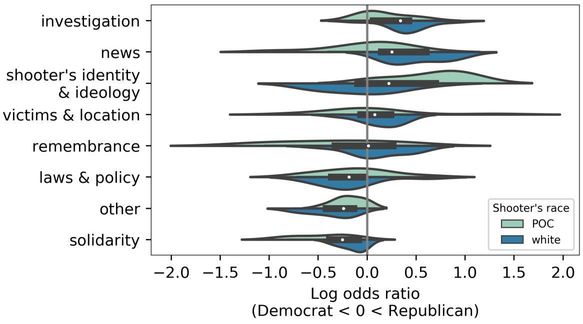


Figure 7: The plot shows the kernel density of the partisan log odds ratios of each topic (one observation per event). The white points show the median and the black rectangles the interquartile range across events.

and types of grounding that are used as partisan framing devices, contributing to polarization.

5.1 Methods

Partisan tokens. We estimate the partisanship of tokens via their event-level log odds ratio of Democrats relative to Republicans (based on the vocabularies we create in Section 3.1). We compare these estimates across events.[12]

Grounding. We study whether there is polarization in which prior tragic events are referenced in the context of a particular mass shooting. We compile a list of keywords representing major events of mass violence in the US in the past two decades and kept those that were mentioned at least 100 times by Democrat or Republican users. For all tweets for each event in our dataset, we counted the mentions of past *context events*. For example, in the following tweet posted after Las Vegas: "Dozens of preventable deaths should not be the cost of living in America. Stand up to the #NRA. #LasVegasShooting #SandyHook #Charleston", Sandy Hook and Charleston are the context events. Finally, we calculated the partisan log odds ratio of each context event.

5.2 Results

We focus on the partisanship of the term "terrorist" and "crazy", which exhibit differential pat-

---

[11] p-values are calculated using a one sample t-test, comparing with zero: *shooter's identity & ideology* ($p < 0.05$), *investigation* ($p < 0.001$), *laws & policy* ($p < 0.1$), *news* ($p < 0.05$), *solidarity* ($p < 0.001$).

[12] To compare the partisanship of tokens at the event- and topic-level, we also z-score the log odds ratios (Monroe et al., 2008) within events and topics; the most partisan tokens are reported in Appendix F. The reason why we do not z-score for the between-event comparison is because data verbosity disproportionately affects the range of the values' magnitudes. Note that the signs of values — which we focus on for the cross-event comparison — are not affected by z-scoring.

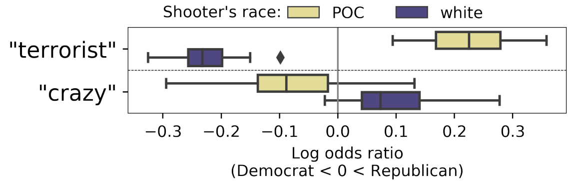


Figure 8: The log odds ratios of "terrorist" and "crazy" across events, grouped by the shooter's race. The boxes show the interquartile range and the diamond an outlier.

terns across events based on the shooter's race.[13] "Terrorist" is *always* more likely to be used by Democrats than Republicans in events where the shooter is white, and the opposite is true when the shooter is a person of color (Figure 8); "crazy" is more likely used by Republicans if the shooter is white than if they are a person of color and the opposite is true (although the pattern is weaker) when a shooter is white.

These findings support related work (Perea, 1997; Delgado and Stefancic, 2017) discussing binary conceptualization of race in the US, and its influence in determining whether a shooter's mental health or aspects of their identity are discussed. However, the fact that the influence of race flips completely for Democrats and Republicans is a striking result that calls for further exploration.

The partisanship of contextual grounding also corroborates our finding that the shooter's race influences how people conceptualize a certain event. Our results in Figure 9 suggest a few key takeaways: the two most frequently employed context events are both highly partisan (Sandy Hook for Democrats and 9/11 for Republicans); shootings at schools and places of worship are more likely to be brought up by Democrats; Democrats are more likely to reference events with white shooters, while Republicans are more likely to reference those with shooters who are people of color.

6 Affect

Affect is intimately tied to ideological reasoning (Redlawsk, 2002; Taber et al., 2009), and so emotional expression represents another semantic layer relevant to polarization (Iyengar et al., 2012; Suhay, 2015). Others have shown that emotion words can help detect political ideology on Twitter (Preoţiuc-Pietro et al., 2017) and that emotive political tweets are more likely to be shared (Stieglitz and Dang-Xuan, 2012). Here, we employ a lexicon-based approach to measure valence (positive and negative) and five basic emotion categories (disgust, fear, trust, anger, and sadness).

6.1 Methods

Since word-affect associations are highly domain dependent, we tailored an existing affect lexicon, the NRC Emotion Lexicon (Mohammad and Turney, 2013), to our domain via label propagation (Hamilton et al., 2016).

Specifically, we stem all the words in the lexicon and select 8-10 representative stems per emotion category that have an association with that emotion in the context of mass shootings. For each emotion category, we compute pairwise cosine distances between the GloVe embedding of each in-vocabulary stem and the representative stems for that emotion, and include the 30 stems with the lowest mean cosine distances. The resulting lexicons can be found in Appendix E.

We use these lexicons to measure the partisanship of each affect category. For each event and each party we aggregate stem frequencies per emotion category. We then calculate the partisan log odds ratio of each category for each event.

6.2 Results

The log odds ratio of each affect category is shown in Figure 10. These findings suggest that positive sentiment, sadness and trust are more likely to be expressed by Democrats across events, while fear and disgust are more likely to be expressed by Republicans, particularly when the shooter is a person of color. Anger, trust and negative sentiment is similarly likely to be expressed by both parties.[14]

Our results about fear and disgust accord with existing literature on emotion and political ideology: conservatives score higher than liberals on subjective measures of fear (e.g. Jost et al., 2017; Federico et al., 2009; Hibbing et al., 2014) and disgust sensitivity is also associated with political conservativism (e.g. Inbar et al., 2009, 2012).

7 Modality and Illocutionary Force

Modality is a lexical category concerned with necessity and possibility (Kratzer, 2002; Fintel,

---

[13] Note that these words in fact have the *largest* difference (negative and positive, respectively) if we calculate the differences between the mean z-scores — grouped by the shooter's race — for all tokens in our joint vocabulary.

[14] p-values are calculated using a one sample t-test, comparing to zero: anger ($p \approx 0.43$), disgust ($p \approx 0.06$), fear ($p < 0.001$), negative ($p \approx .2$), positive ($p < 0.001$), sadness ($p < 0.02$), trust ($p \approx 0.07$).

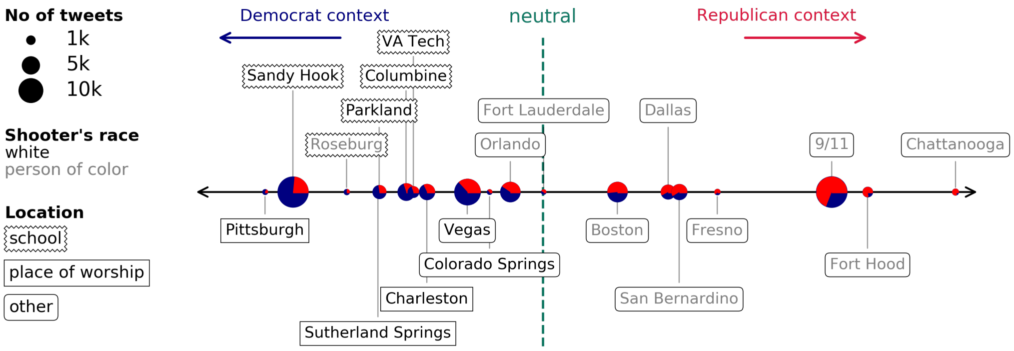


Figure 9: The partisanship of events of mass violence when used as a context for a given mass shooting. The position of the events on the line represents their partisan log odds ratio (Democrat $<$ 0 (neutral) $<$ Republican). The pie charts indicate the proportion of Democrat and Republican users' tweets that involve this "context" event.

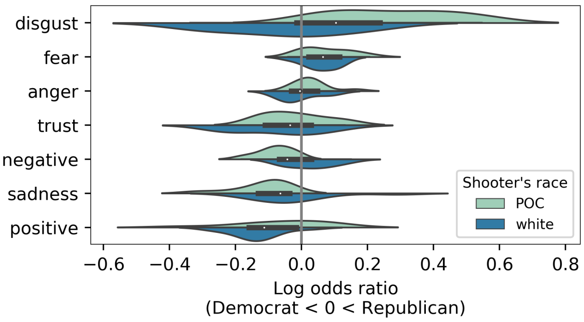


Figure 10: The log odds of each emotion category in our lexicon (one observation represents one event).

2006). In the aftermath of a tragic event, people seek solutions, a process that often involves reflecting on what *should have* happened or *should* happen now or in the future (e.g. to prevent such events). We hypothesize that the use of modals in our data gives insight into the kind of (illocutionary) acts (Austin, 1962) the users are performing via their tweets, such as calling for action, assigning blame, expressing emotions, and stating facts.

7.1 Methods

We work with all forms of the four most frequent necessity modals in our data — *should, must, have to* and *need to*. For each, we quantify its partisanship via its partisan log odds ratio. We also annotate a random sample of 200 tweets containing modals to see whether they are indeed used in contexts that imply calls for change / action (e.g. 'We must have gun control!') and / or to express the user's mental state about the event, such as despair or disbelief (e.g. 'Why do people have to die?').

7.2 Results

Table 2 shows a random sample of tweets containing some form of either *should, must, have to*, or *need to*. More collocations, as well as their partisanship, can be found in Appendix H. These examples, as well as our annotation, support the hypothesis that these modals are primarily used to call for action. Of the 200 modal uses, ∼78% express calls for change/action, ∼40% express the user's mental state.[15] We also compute the representation p^m_x of each modal m in each topic $x \in X$ via $(f^m_x/\sum_{x' \in X} f^m_{x'})/(f_x/\sum_{x' \in X} f_{x'})$, where f_x is the number of tweets from topic x, and f^m_x the number of those also containing m. We find that that modals are over-represented in the *laws & policy* topic (see Figure 11). This evidence suggests that calls for policy change — especially gun control, based on annotated samples — are a dominant subset of calls for action.

This roller coaster debate **MUST** STOP! Sensible gun ownership is one thing but assault weapons massacre innocent lives. The savagery of gore at #Parkland was beyond belief & **must** be the last.
In times of tragedy **shouldn't** we all come together?! Prayers for those harmed in the #PlannedParenthood shooting.
Communities **need to** step up and address white on white crime like the Las Vegas massacre. White men are out of control.
he BLM protest shooting, planned parenthood, now cali... domestic terrorism will crumble this country, SANE PPL **HAVE TO** FIGHT BACK
Shooting cops is horrible, cannot be condoned. But **must be** understood these incidents are outgrowth of decades of police abuses. #BatonRouge
1. Islamic terrorists are at war with us 2. Gun free zones = kill zones 3. Americans **should be** allowed to defend themselves #Chattanooga
Las Vegas shooting Walmart shooting and now 25 people killed in Texas over 90 people killed Mexico **should** build that wall to keep the US out
CNN reporting 20 dead, 42 injured in Orlando night club shooting. Just awful. The US **must** act to control guns or this carnage will continue.

Table 2: Random sample of tweets with modals. Only one of the eight (Ex. 6) tweets expresses ideas traditionally associated with conservative ideology.

The log odds ratio of modals shows that *all* of them are more likely to be used by Democrats across events: *have to* (mean:$-.39$, $p < 0.001$),

---

[15] Other uses are primarily epistemic ones (e.g. 'The suspect must be mentally ill').

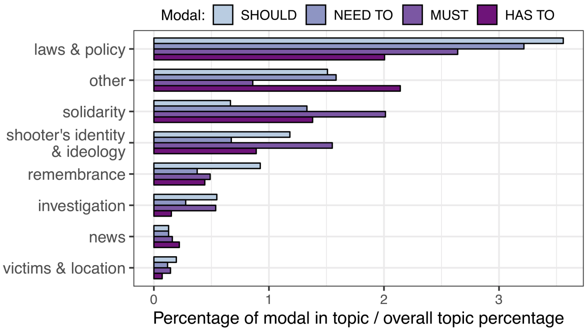


Figure 11: The representation of modals in each topic. Values represent averages across events.

must (mean:$-.3$, $p < 0.001$), *should* (mean:$-.18$, $p < 0.01$), *need to* (mean:$-.18$, $p < 0.01$) — where Democrat and Republican log odds are negative and positive, respectively.[16] A two-tailed t-test shows that only *should* exhibits statistically significant difference based on the shooter's race ($p < 0.03$), as it is even more likely to be used by Democrats when the shooter is white.

To understand whether assigning blame in this domain is a partisan propensity, we also study uses of *should have*.[17] The log odds of *should have* (mean:$-.22$, $p < 0.05$) show that it is similarly likely to be used by Democrats as *should* ($p \approx 0.8$ from two-tailed t-test). Interestingly the log odds ratio of *should have*, unlike that of *should*, does not differ significantly based on the shooter's race ($p \approx 0.8$ from two-tailed t-test). Moreover, we did not find a significant difference in the partisanship of *should have* nor any other modal based on the administration (Obama or Trump) a shooting took place under, suggesting that Democrats are more likely call for change and assign blame even if their preferred party is in power.

8 Conclusion

We show that inspecting polarization on social media from various angles can shed light on salient phenomena pertinent to group divisions. Applying the leave-out estimator of phrase partisanship to data on mass shootings, we find that reactions to these events are highly polarized politically.

To disentangle topic choice and topic-level framing — two phenomena that contribute to polarization — we introduce a tweet-clustering approach. By sampling, requiring words in the vocabulary to appear in multiple events and relying on the abstraction of a vector space model, we generate cohesive topic representations that are robust to disparities among event-level vocabularies and tweet counts. Human evaluation shows that our method outperforms LDA-based approaches.

Our induced topics suggest that Republicans preferentially discuss topics about the shooter's identity and ideology, investigation and news, while Democrats preferentially discuss solidarity and policy-related topics. We also find that the setting and the shooter's race interact with polarization. For example, Democrats are more likely to contextualize any mass shooting among school shootings and call white shooters "terrorists" than are Republicans, who in turn are more likely to liken any shooting to other violent events perpetrated by people of color — whom they are more likely to call "terrorist" than are Democrats. Moreover, Democrats are more likely to frame the shooter as mentally ill when they are a person of color and Republicans when they are white.

We also demonstrate that looking at affect and illocutionary force can help us understand users' polarized responses to these tragic events: Republicans are more likely to express fear and disgust than are Democrats, while Democrats are more likely to express sadness and positive sentiment, to make calls for action and assign blame.

Polarization is a multi-faceted phenomenon: in this paper we present a set of measures to study these different facets through the lens of language. We show that these measures provide convergent evidence, creating a clearer picture of the complex ideological division permeating public life.

Acknowledgements. We thank Jure Leskovec and Adrijan Bradaschia for data, and Cleo Condoravdi, Chris Potts, Linda Ouyang, David Ritzwoller and Frank Yang for helpful feedback. We are grateful for the support of the Stanford Cyber Initiative, the Melvin and Joan Lane Stanford Graduate Fellowship (to D.D.), NSF GRF DGE-114747 (to N.G.), the Michelle and Kevin Douglas Stanford Interdisciplinary Graduate Fellowship (to R.V.), NSF grant CRII 1657155 (to J.Z.), the Stanford Institute for Economic Policy Research and the Knight Foundation (to M.G.) and the Brown University Population Studies and Training Center (to J.S.).

---

[16] p-values are from one sample t-test, comparing to 0.

[17] On average across events, forms of *should have* constitute 16% of the uses of *should* (SD=5%).

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *International Conference on Learning Representations (ICLR) 2017*.

J. L. Austin. 1962. *How to Do Things with Words*. Harvard University Press.

John W Ayers, Benjamin M Althouse, Eric C Leas, Ted Alcorn, and Mark Dredze. 2016. Can Big Media Data Revolutionarize Gun Violence Prevention? *Bloomberg Data for Good Exchange*.

Eytan Bakshy, Solomon Messing, and Lada A Adamic. 2015. Exposure to Ideologically Diverse News and Opinion on Facebook. *Science*, 348(6239):1130–1132.

Ramnath Balasubramanyan, William W Cohen, Douglas Pierce, and David P Redlawsk. 2012. Modeling Polarizing Topics: When Do Different Political Communities Respond Differently to the Same News? In *Sixth International AAAI Conference on Weblogs and Social Media*.

Adrian Benton, Braden Hancock, Glen Coppersmith, John W Ayers, and Mark Dredze. 2016. After Sandy Hook Elementary: A year in the gun control debate on Twitter. *Bloomberg Data for Good Exchange*.

Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2013. Identifying Media Frames and Frame Dynamics Within and Across Policy Issues. In *New Directions in Analyzing Text as Data Workshop, London*.

Dallas Card, Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The Media Frames Corpus: Annotations of Frames Across Issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 438–444.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Neural Information Processing Systems*.

Dennis Chong and James N Druckman. 2007. Framing Public Opinion in Competitive Democracies. *American Political Science Review*, 101(4):637–655.

Hsiang Iris Chyi and Maxwell McCombs. 2004. Media Salience and the Process of Framing: Coverage of the Columbine School Shootings. *Journalism & Mass Communication Quarterly*, 81(1):22–35.

Michael Conover, Jacob Ratkiewicz, Matthew R Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. 2011. Political Polarization on Twitter. *Fifth International AAAI Conference on Weblogs and Social Media*, 133:89–96.

Richard Delgado and Jean Stefancic. 2017. *Critical Race Theory: An Introduction*. NYU Press.

Kevin M Drakulich. 2015. Explicit and Hidden Racial Bias in the Framing of Social Problems. *Social Problems*, 62(3):391–418.

Scott W Duxbury, Laura C Frizzell, and Sadé L Lindsay. 2018. Mental Illness, the Media, and the Moral Politics of Mass Violence: The Role of Race in Mass Shootings Coverage. *Journal of Research in Crime and Delinquency*, 55(6):766–797.

Robert M. Entman. 2007. Framing Bias: Media in the Distribution of Power. *Journal of Communication*, 57(1):163–173.

Christopher M Federico, Corrie V Hunt, and Damla Ergun. 2009. Political Expertise, Social Worldviews, and Ideology: Translating "Competitive Jungles and Dangerous Worlds into Ideological Reality. *Social Justice Research*, 22(2-3):259–279.

Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky, and Yulia Tsvetkov. 2018. Framing and agenda-setting in russian news: a computational analysis of intricate political strategies. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3570–3580.

Kai Von Fintel. 2006. Modality and Language. In D. Borchert, editor, *Encyclopedia of Philosophy*, pages 20–27. Macmillan Reference.

Morris P Fiorina and Samuel J Abrams. 2008. Political polarization in the American public. *Annual Review of Political Science*, 11:563–588.

Matthew Gentzkow, Jesse M Shapiro, and Matt Taddy. forthcoming. Measuring polarization in high-dimensional data: Method and application to congressional speech. Technical report.

Donald P. HaiderMarkel and Mark R. Joslyn. 2001. Gun Policy, Opinion, Tragedy, and Blame Attribution: The Conditional Influence of Issue Frames. *Journal of Politics*, 63(2):520–543.

Yosh Halberstam and Brian Knight. 2016. Homophily, Group Size, and the Diffusion of Political Information in Social Networks: Evidence from Twitter. *Journal of Public Economics*, 143:73–88.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.

John R Hibbing, Kevin B Smith, and John R Alford. 2014. Differences in negativity bias underlie variations in political ideology. *Behavioral and Brain Sciences*, 37(3):297–307.

Jacob B Hirsh and Jordan B Peterson. 2009. Personality and language use in self-narratives. *Journal of Research in Personality*, 43(3):524–527.

Yoel Inbar, David Pizarro, Ravi Iyer, and Jonathan Haidt. 2012. Disgust Sensitivity, Political Conservatism, and Voting. *Social Psychological and Personality Science*, 3(5):537–544.

Yoel Inbar, David A Pizarro, and Paul Bloom. 2009. Conservatives are More Easily Disgusted than Liberals. *Cognition & Emotion*, 23(4):714–725.

Shanto Iyengar, Gaurav Sood, and Yphtach Lelkes. 2012. Affect, Not Ideology: A Social Identity Perspective on Polarization. *Public Opinion Quarterly*, 76(3):405–431.

Kristen Johnson, Di Jin, and Dan Goldwasser. 2017. Modeling of Political Discourse Framing on Twitter. In *Eleventh International AAAI Conference on Web and Social Media*.

John T Jost, Chadly Stern, Nicholas O Rule, and Joanna Sterling. 2017. The Politics of Fear: Is There an Ideological Asymmetry in Existential Motivation? *Social Cognition*, 35(4):324–353.

Angelika Kratzer. 2002. The Notional Category of Modality. *Formal Semantics: The Essential Readings*, pages 289–323.

Geoffrey C Layman, Thomas M Carsey, John C Green, Richard Herrera, and Rosalyn Cooperman. 2010. Party polarization, party commitment, and conflict extension among American party activists. *American Political Science Review*, 104(2):324–346.

Maxwell McCombs. 2002. The Agenda-Setting Role of the Mass Media in the Shaping of Public Opinion. In *Proceedings of the 2002 Conference of Mass Media Economics, London School of Economics*.

William Mingus and Bradley Zopf. 2010. White Means Never Having to Say You're Sorry The Racial Project in Explaining Mass Shootings. *Social Thought & Research*, 31:57–77.

Amy Mitchell, Jeffrey Gottfried, Michael Barthel, and Elisa Shearer. 2016. The Modern News Consumer. *Pew Research Center*, 7.

Saif M Mohammad and Peter D Turney. 2013. NRC Emotion Lexicon. *National Research Council, Canada*.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Glenn W. Muschert and Dawn Carr. 2006. Media Salience and Frame Changing across Events: Coverage of Nine School Shootings, 19972001. *Journalism & Mass Communication Quarterly*, 83(4):747–766.

Nona Naderi and Graeme Hirst. 2017. Classifying Frames at the Sentence Level in News Articles. *Recent Advances in Natural Language Processing*, pages 536–542.

Sung-Yeon Park, Kyle J Holody, and Xiaoqun Zhang. 2012. Race in Media Coverage of School Shootings A Parallel Application of Framing Theory and Attribute Agenda Setting. *Journalism & Mass Communication Quarterly*, 89(3):475–494.

Ellie Pavlick, Heng Ji, Xiaoman Pan, and Chris Callison-Burch. 2016. The Gun Violence Database: A new task and data set for NLP. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1018–1024.

James W Pennebaker. 2011. The secret life of pronouns. *New Scientist*, 211(2828):42–45.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic Inquiry and Word Count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Juan F Perea. 1997. The Black/White Binary Paradigm of Race: The Normal Science of American Racial Thought. *California Law Review*, 85.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond Binary Labels: Political Ideology Prediction of Twitter Users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

Markus Prior. 2013. Media and Political Polarization. *Annual Review of Political Science*, 16:101–127.

Lin Qiu, Han Lin, Jonathan Ramsay, and Fang Yang. 2012. You are what you tweet: Personality expression and perception on Twitter. *Journal of Research in Personality*, 46(6):710–718.

David P Redlawsk. 2002. Hot cognition or cool consideration? testing the effects of motivated reasoning on political decision making. *The Journal of Politics*, 64(4):1021–1044.

Jaclyn Schildkraut and H. Jaymi Elsass. 2016. *Mass Shootings: Media, Myths, and Realities: Media, Myths, and Realities*. ABC-CLIO.

Jaclyn Schildkraut, H. Jaymi Elsass, and Kimberly Meredith. 2018. Mass shootings and the media: why all events are not created equal. *Journal of Crime and Justice*, 41(3):223–243.

Jaclyn Schildkraut and Glenn W. Muschert. 2014. Media Salience and the Framing of Mass Murder in Schools: A Comparison of the Columbine and Sandy Hook Massacres. *Homicide Studies*, 18(1):23–43.

Stefan Stieglitz and Linh Dang-Xuan. 2012. Political communication and influence through microblogging: An empirical analysis of sentiment in Twitter messages and retweet behavior. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 3500–3509. IEEE.

Elizabeth Suhay. 2015. Explaining Group Influence: The Role of Identity and Emotion in Political Conformity and Polarization. *Political Behavior*, 37(1):221–251.

Charles S Taber, Damon Cann, and Simona Kucsova. 2009. The Motivated Processing of Political Arguments. *Political Behavior*, 31(2):137–155.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A Frame of Mind: Using Statistical Models for Detection of Framing and Agenda Setting Campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1629–1638.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring User Political Preferences from Streaming Communications. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 186–196.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A Biterm Topic Model for Short Texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456. ACM.

Tal Yarkoni. 2010. Personality in 100,000 Words: A large-scale analysis of personality and word use among bloggers. *Journal of Research in Personality*, 44(3):363–373.

Muhammad Bilal Zafar, Krishna P Gummadi, and Cristian Danescu-Niculescu-Mizil. 2016. Message Impartiality in Social Media Discussions. In *Tenth International AAAI Conference on Web and Social Media*, pages 466–475.

A Data

Table 3 contains properties of the data. Figure 12 contains the distribution of partisan tweets for each event.

Event-specific keywords. We use the following location-specific keywords (case insensitive) to find tweets on the events:

- Chattanooga: chattanooga, military recruitment center
- Roseburg: umpqua, roseburg
- Colorado Springs: colorado springs, coloradosprings, planned parenthood, plannedparenthood
- San Bernardino: san bernardino, sanbernardino
- Kalamazoo: kalamazoo
- Orlando: orlando, pulse nightclub
- Dallas: dallas
- Baton Rouge: baton rouge, batonrouge
- Burlington: burlington, cascade mall
- Fort Lauderdale: lauderdale
- Fresno: fresno
- San Francisco: ups, san francisco
- Las Vegas: vegas, route91, harvest festival, harvestfestival, mandalay bay
- Thornton: thornton, walmart, denver
- Sutherland Springs: sutherland springs, sutherlandsprings
- Parkland: parkland, marjory stoneman
- Nashville: nashville, waffle house
- Santa Fe: santa fe, santafe
- Annapolis: annapolis, capital gazette
- Pittsburgh: pittsburgh, treeoflife, tree of life
- Thousand Oaks: thousand oaks, thousandoaks

Event city / town	**State**	**Specific location**	**Date**	**No. victims**	**Race / ethnicity of shooter**	**No. tweets**	**No. partisan tweets**	**No. Dem tweets**	**No. Rep tweets**
Chattanooga	TN	Military Recruitment Center	7/16/15	7	Middle Eastern	29573	20709	5925	14784
Roseburg	OR	Umpqua Community College	10/1/15	18	Mixed	18076	11505	6419	5086
Colorado Springs	CO	Planned Parenthood clinic	11/27/15	12	White	55843	39719	26614	13105
San Bernardino	CA	Inland Regional Center	12/2/15	35	Middle Eastern	70491	45819	20798	25021
Kalamazoo	MI	multiple	2/20/16	8	White	10986	6807	4350	2457
Orlando	FL	Pulse nightclub	6/12/16	102	Middle Eastern	1831082	872022	450784	421238
Dallas	TX	Black Lives Matter protest	7/7/16	16	Black	260377	144205	64628	79577
Baton Rouge	LA	streets	7/17/16	6	Black	46126	29015	12019	16996
Burlington	WA	Cascade Mall	9/23/16	5	Middle Eastern	8171	4993	1838	3155
Fort Lauderdale	FL	Fort Lauderdale airport	1/6/17	11	Hispanic	12525	7194	3073	4121
Fresno	CA	downtown	4/18/17	3	Black	8868	6128	1377	4751
San Francisco	CA	UPS store	6/14/17	5	Asian	10487	6627	4346	2281
Vegas	NV	Route 91 Harvest Festival	10/1/17	604	White	1286399	726739	315343	411396
Thornton	CO	Walmart	11/1/17	3	White	14341	9170	5527	3643
Sutherland Springs	TX	Texas First Baptist Church	11/5/17	46	White	154076	106220	52513	53707
Parkland	FL	Marjory Stoneman Douglas High School	2/14/18	31	White	272499	186570	113856	72714
Nashville	TN	Waffle House	4/22/18	8	White	38680	24326	14606	9720
Santa Fe	CA	Santa Fe High School	5/18/18	23	White	73621	42968	26784	16184
Annapolis	MD	Capital Gazette	6/28/18	7	White	27715	18468	11863	6605
Pittsburgh	PA	Tree of Life Synagogue	10/27/18	18	White	59925	36920	22735	14185
Thousand Oaks	CA	Borderline Bar and Grill	11/7/18	23	White	117815	62812	40328	22484

Table 3: Data properties.

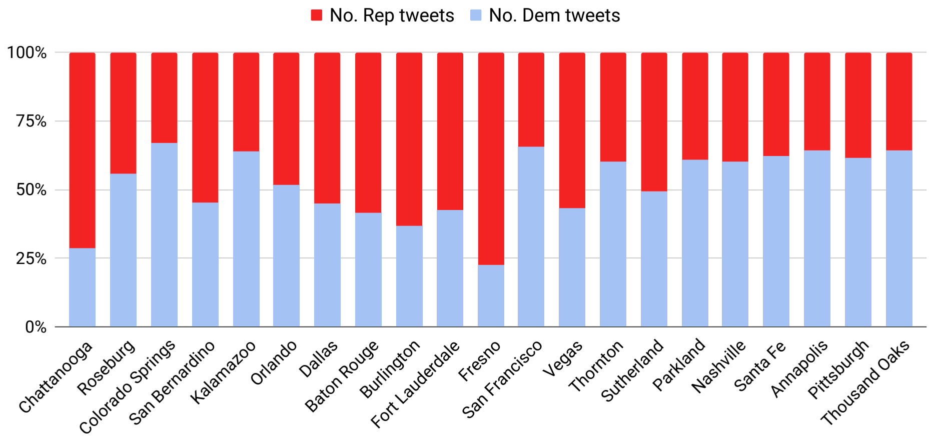


Figure 12: Distribution of partisan tweets for each event.

A.1 Stopwords

no, noone, nobody, nowhere, nothing, nor, not, none, non, a, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, aren, around, as, aside, ask, asking, associated, at, available, away, awfully, b, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, came, can, cannot, cant, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, couldnt, couldn, couldve, course, currently, d, definitely, described, despite, did, didnt, didn, different, do, dont, don, does, doesn, doesnt, doing, done, down, downwards, during, e, each, edu, eg, eight, either, else, elsewhere, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except, f, far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, further, furthermore, g, get, gets, getting, given, gives, go, goes, going, gone, got, gotten, greetings, h, had, hadn, happens, hardly, has, hasnt, hasn, have, havent, haven, having, he, hes, hell, hello, help, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, hi, him, himself, his, hither, hopefully, how, howbeit, however, i, im, ive, ie, if, i, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, isn, it, its, itself, j, just, k, keep, keeps, kept, know, knows, known, l, last, lately, later, latter, latterly, least, less, lest, let, like, liked, likely, little, ll, look, looking, looks, ltd, m, mainly, many, may, maybe, me, mean, meanwhile, merely, might, mightve, more, moreover, most, mostly, much, must, mustn, mustnt, mustve, my, myself, n, name, namely, nd, near, nearly, necessary, need, neednt, needs, neither, never, nevertheless, new, next, nine, nor, normally, novel, now, o, obviously, of, off, often, oh, ok, okay, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own, p, particular, particularly, per, perhaps, placed, please, plus, possible, presumably, probably, provides, q, que, quite, qv, r, rather, rd, re, really, reasonably, regarding, regardless, regards, relatively, respectively, right, s, said, same, saw, say, saying, says, second, secondly, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sensible, sent, serious, seriously, seven, several, shall, she, shell, shes, should, shouldnt, shouldn, shouldve, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specified, specify, specifying, still, sub, such, sup, sure, t, take, taken, tell, tends, th, than, thank, thanks, thanx, that, thats, the, their, theirs, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, theres, thereupon, these, they, theyre, theyve, think, third, this, thorough, thoroughly, those, though, three, through, throughout, thru, thus, to, together, too, took, toward, towards, tried, tries, truly, try, trying, twice, two, u, un, under, unfortunately, unless, unlikely, until, unto, up, upon, us, use, used, useful, uses, using, usually, uucp, v, value, various, ve, very, via, viz, vs, w, want, wants, was, way, we, welcome, well, went, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, willing, wish, with, within, without, wonder, would, wouldn, wouldnt, wouldve, x, y, yes, yet, you, youre, youve, your, yours, yourself, yourselves, z, zero

B Partisanship Assignment

B.1 Political Twitter Handles

Democrat. AGBecerra, AlanGrayson, AngusKing2018, AnthonyBrownMD4, BarbaraBoxer, BenCardinforMD, BennetForCO, BennieGThompson, BernieSanders, BettyMcCollum04, BillPascrell, Bob_Casey, BobbyScott, Booker4Senate, BradSherman, Call_Me_Dutch, ChrisCoons, ChrisMurphyCT, ChrisVanHollen, Clyburn, CongressmanRaja, CongressmanRuiz, CoryBooker, DWStweets, DianneFeinstein, DickBlumenthal, DickDurbin, DonaldNorcross, DorisMatsui, EPWDems, EdMarkey, EleanorNorton, EnergyDems, FrankPallone, GKButterfield, GerryConnolly, HELPCmteDems, HeidiHeitkamp, Heinrich4NM, HillaryClinton, HouseDemocrats, JECDems, JacksonLeeTX18, JeanneShaheen, JeffMerkley, JimLangevin, JoaquinCastrotx, JoeManchinWV, JohnCarneyDE, JuliaBrownley, JuliaBrownley26, KamalaHarris, LacyClayMO1, LloydDoggettTX, LorettaSanchez, MariaCantwell, MarkWarner, MartinHeinrich, McCaskillOffice, Menendez4NJ, MurrayCampaign, NancyPelosi, NelsonForSenate, NitaLowey, NormaJTorres, NydiaVelazquez, PattyMurray, PeterWelch, Peters4Michigan, RepAdamSchiff, RepAdamSmith, RepAlGreen, RepAlLawsonJr, RepAndreCarson, RepAnnaEshoo, RepAnnieKuster, RepBRochester, RepBarbaraLee, RepBarragan, RepBeatty, RepBera, RepBetoORourke, RepBillFoster, RepBobbyRush, RepBonamici, RepBonnie, RepBradAshford, RepBrady, RepBrendanBoyle, RepBrianHiggins, RepCarbajal, RepCardenas, RepCartwright, RepCharlieCrist, RepCheri, RepCicilline, RepCohen, RepCuellar, RepCummings, RepDanKildee, RepDannyDavis, RepDarrenSoto, RepDavidEPrice, RepDeSaulnier, RepDebDingell, RepDelBene, RepDennyHeck, RepDerekKilmer, RepDianaDeGette, RepDonBeyer, RepDonaldPayne, RepDwightEvans, RepEBJ, RepEliotEngel, RepEspaillat, RepEsty, RepFilemonVela, RepGaramendi, RepGeneGreen, RepGraceMeng, RepGregoryMeeks, RepGutierrez, RepGwenMoore, RepHanabusa, RepHankJohnson, RepHastingsFL, RepHuffman, RepJackyRosen, RepJaredPolis, RepJayapal, RepJeffries, RepJerryNadler, RepJimCosta, RepJimMcDermott, RepJimmyPanetta, RepJoeCourtney, RepJoeKennedy, RepJohnConyers, RepJohnDelaney, RepJohnLarson, RepJohnYarmuth, RepJoseSerrano, RepJoshG, RepJuanVargas, RepJudyChu, RepKClark, RepKarenBass, RepKathleenRice, RepKihuen, RepLawrence, RepLindaSanchez, RepLipinski, RepLoisCapps, RepLoisFrankel, RepLouCorrea, RepLowenthal, RepLujanGrisham, RepMaloney, RepMarciaFudge, RepMarcyKaptur, RepMarkTakai, RepMarkTakano, RepMaxineWaters, RepMcEachin, RepMcGovern, RepMcNerney, RepMikeHonda, RepMikeQuigley, RepOHalleran, RepPaulTonko, RepPerlmutter, RepPeteAguilar, RepPeterDeFazio, RepRaskin, RepRaulGrijalva, RepRichardNeal, RepRichmond, RepRickLarsen, RepRoKhanna, RepRobinKelly, RepRonKind, RepRoybalAllard, RepRubenGallego, RepSarbanes, RepSchakowsky, RepSchneider, RepSchrader, RepScottPeters, RepSeanMaloney, RepSheaPorter, RepSinema, RepSires, RepSpeier, RepStephMurphy, RepStephenLynch, RepSteveIsrael, RepSusanDavis, RepSwalwell, RepTedDeutch, RepTedLieu, RepTerriSewell, RepThompson, RepTimRyan, RepTimWalz, RepTomSuozzi, RepValDemings, RepVeasey, RepVisclosky, RepWilson, RepYvetteClarke, RepZoeLofgren, RonWyden, SanfordBishop, SchatzforHawaii, SenAngusKing, SenBennetCO, SenBillNelson, SenBlumenthal, SenBobCasey, SenBooker, SenBrianSchatz, SenCoonsOffice, SenCortezMasto, SenDonnelly, SenDuckworth, SenFeinstein, SenFranken, SenGaryPeters, SenGillibrand, SenJackReed, SenJeffMerkley, SenKaineOffice, SenKamalaHarris, SenMarkey, SenSanders, SenSchumer, SenSherrodBrown, SenStabenow, SenWarren, SenWhitehouse, Sen_JoeManchin, SenateApprops, SenateDems, SenatorBaldwin, SenatorBarb, SenatorBoxer, SenatorCantwell, SenatorCardin, SenatorCarper, SenatorDurbin, SenatorHassan, SenatorHeitkamp, SenatorLeahy, SenatorMenendez, SenatorReid, SenatorShaheen, SenatorTester, SenatorTomUdall, SenatorWarner, SherrodBrown, StaceyPlaskett, SupJaniceHahn, TheDemocrats, TomCarperforDE, TomUdallPress, TulsiPress, USRepKCastor, USRepKeating, USRepMikeDoyle, USRepRHinojosa, USRepRickNolan, WhipHoyer, WydenForOregon, WydenPress, alfranken, amyklobuchar, brianschatz, cbrangel, chakafattah, chelliepingree, chuckschumer, clairecmc, collinpeterson, coons4delaware, daveloebsack, dscc, elizabethforma, gracenapolitano, jahimes, janschakowsky, jontester, keithellison, louiseslaughter, mazieforhawaii, maziehirono, mikecapuano, nikiinthehouse, pedropierluisi, repbenraylujan, repblumenauer, repcleaver, repdavidscott, repdinatitus, repdonnaedwards, repjimcooper, repjoecrowley, repjohnlewis, repmarkpocan, repsandylevin, rosadelauro, sethmoulton, stabenow, tammybaldwin, timkaine, tomudall

Republicans. AustinScottGA08, BankingGOP, CarlyFiorina, ChrisChristie, ChuckGrassley, ConawayTX11, CongCulberson, CongMikeSimpson, CongressmanDan, CongressmanGT, CongressmanHice, DanaRohrabacher, DarrellIssa, DrNealDunnFL2, DrPhilRoe, EPWGOP, EdWorkforce, EnergyGOP, FinancialCmte, GOP, GOPHELP, GOPLeader, GOPSenFinance, GOPoversight, GOPpolicy, GovMikeHuckabee, GrahamBlog, GrassleyPress, GreggHarper, HASCRepublicans, HerreraBeutler, HouseAdmnGOP, HouseAgNews, HouseApropsGOP, HouseCommerce, HouseGOP, HouseHomeland, HouseJudiciary, HouseScience, HouseSmallBiz, HouseVetAffairs, HurdOnTheHill, InhofePress, JebBush, JeffFlake, JeffFortenberry, JerryMoran, JimInhofe, JimPressOffice, Jim_Jordan, JohnBoozman, JohnCornyn, JohnKasich, JudgeCarter, JudgeTedPoe, KeithRothfus, KenCalvert, LamarSmithTX21, MacTXPress, MarioDB, MarkAmodeiNV2, MarshaBlackburn, McConnellPress, MikeCrapo, MikeKellyPA, NatResources, PatTiberi, PatrickMcHenry, PeteSessions, PeterRoskam, PortmanPress, RandPaul, Raul_Labrador, RealBenCarson, RepAbraham, RepAdrianSmith, RepAndyBarr, RepAndyHarrisMD, RepAnnWagner, RepArrington, RepBillFlores, RepBillJohnson, RepBillShuster, RepBlainePress, RepBobGibbs, RepBradWenstrup, RepBrianBabin, RepBrianMast, RepBuddyCarter, RepByrne, RepCharlieDent, RepChrisCollins, RepChrisSmith, RepChrisStewart, RepChuck, RepComstock, RepCurbelo, RepDLamborn, RepDaveJoyce, RepDavid, RepDavidValadao, RepDavidYoung, RepDeSantis, RepDennisRoss, RepDianeBlack, RepDougCollins, RepDrewFerguson, RepEdRoyce, RepErikPaulsen, RepFrankLucas, RepFredUpton, RepFrenchHill, RepGallagher, RepGarretGraves, RepGoodlatte, RepGosar, RepGusBilirakis, RepGuthrie, RepHalRogers, RepHartzler, RepHensarling, RepHolding, RepHuizenga, RepHultgren, RepJBridenstine, RepJackBergman, RepJasonLewis, RepJasonSmith, RepJeffDenham, RepJeffDuncan, RepJimBanks, RepJimRenacci, RepJoeBarton, RepJoeWilson, RepJohnFaso, RepJohnKatko, RepKayGranger, RepKenBuck, RepKenMarchant, RepKevinBrady, RepKevinCramer, RepKevinYoder, RepKinzinger, RepKristiNoem, RepLaHood, RepLaMalfa, RepLanceNJ7, RepLarryBucshon, RepLeeZeldin, RepLoBiondo, RepLouBarletta, RepLoudermilk, RepLukeMesser, RepLynnJenkins, RepMGriffith, RepMarkMeadows, RepMarkWalker, RepMarthaRoby, RepMcCaul, RepMcClintock, RepMcKinley, RepMeehan, RepMiaLove, RepMikeBishop, RepMikeCoffman, RepMikeRogersAL, RepMikeTurner, RepMimiWalters, RepMoBrooks, RepMoolenaar, RepMullin, RepNewhouse, RepPaulCook, RepPaulMitchell, RepPeteKing, RepPeteOlson, RepPoliquin, RepRWilliams, RepRalphNorman, RepRichHudson, RepRickAllen, RepRobBishop, RepRodBlum, RepRussell, RepRyanCostello, RepRyanZinke, RepSanfordSC, RepScottPerry, RepSeanDuffy, RepShimkus, RepSmucker, RepStefanik, RepStevePearce, RepSteveStivers, RepTedYoho, RepThomasMassie, RepTipton, RepTomEmmer, RepTomGarrett, RepTomGraves, RepTomMacArthur, RepTomMarino, RepTomPrice, RepTomReed, RepTomRice, RepTrentFranks, RepTrey, RepWalberg, RepWalorski, RepWalterJones, RepWebster, RepWesterman, Rep_Hunter, RepublicanStudy, RobWittman, Robert_Aderholt, RodneyDavis, RosLehtinen, RoyBlunt, RoyBluntPress, RulesReps, SASCMajority, SamsPressShop, SenAlexander, SenBobCorker, SenCapito, SenCoryGardner, SenDanSullivan, SenDeanHeller, SenJohnBarrasso, SenJohnHoeven, SenJohnKennedy, SenJohnMcCain, SenJohnThune, SenJoniErnst, SenMikeLee, SenPatRoberts, SenRonJohnson, SenRubioPress, SenSasse, SenTedCruz, SenThadCochran, SenThomTillis, SenToddYoung, SenTomCotton, SenToomey, SenateCommerce, SenateGOP, SenateMajLdr, SenateRPC, SenatorBurr, SenatorCollins, SenatorEnzi, SenatorFischer, SenatorIsakson, SenatorLankford, SenatorRisch, SenatorRounds, SenatorTimScott, SenatorWicker, SpeakerRyan, SteveDaines, SteveKingIA, SteveScalise, SusanWBrooks, TGowdySC, TXRandy14, ToddRokita, TomColeOK04, TomRooney, Transport, USRepGaryPalmer, USRepLong, USRepRodney, VernBuchanan, WarrenDavidson, WaysandMeansGOP, amashoffice, boblatta, cathymcmorris, congbillposey, davereichert, farenthold, housebudgetGOP, lisamurkowski, marcorubio, michaelcburgess, mike_pence, realDonaldTrump, rep_stevewomack, repdavetrott, repdonyoung, repgregwalden, replouiegohmert, reppittenger, senategopfloor, sendavidperdue, senorrinhatch, senrobportman, tedcruz, virginiafoxx

B.2 Partisan Assignment Sanity Check

In our sanity check, we exclude DC, as 1) it is not an official US state and 2) there the population of users (including politicians and media) is expected to differ from the voting population. Figure 13 gives DC values.

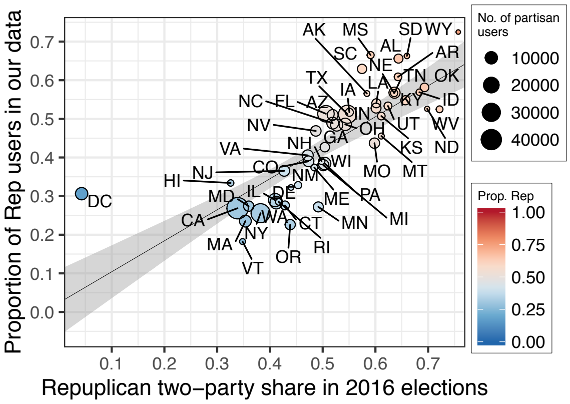


Figure 13: This plot was generated the same way as Figure 1, except that it also includes DC, which shows that it is an outlier in our data due to the fact that there are many Republican politicians and media outlets who are affiliated on Twitter with DC, while DC's voting population tends to be Democratic.

B.3 Russian account presence

We find no substantial presence of Russian accounts in the tweet set that we used for analysis, after all pre-processing. We use the list of Russian accounts identified by Twitter and banned in November 2017.[18] This list is likely to be an underestimate of the true presence of foreign influence, but it nevertheless provides some indication of such activity. Table 4 contains a breakdown by event; we exclude the events that occurred after the accounts were banned. Orlando had one account that tweeted 115 times, and Vegas had 4 that tweeted a total of 70 times.

Event	Number of Russian accounts	Number of tweets
Baton Rouge	8	12
Burlington	3	9
Chattanooga	0	0
Colorado Springs	0	0
Dallas	4	39
Fort Lauderdale	0	0
Fresno	2	2
Kalamazoo	0	0
Orlando	1	115
Roseburg	0	0
San Bernardino	0	0
San Francisco	0	0
Sutherland Springs	0	0
Thornton	0	0
Vegas	4	70

Table 4: Tweets by Russian accounts

---

[18] Available at `https://www.recode.net/2017/11/2/16598312/russia-twitter-trump-twitter-deactivated-handle-list`

Topic	10 Highest Probability Stems
1	gun, shoot, law, control, peopl, shooter, church, hous, stop, school
2	school, shoot, high, gun, student, parkland, kid, shooter, texa, kill
3	shoot, victim, famili, prayer, thought, pray, today, kill, gun, heart
4	shoot, polic, dead, shooter, report, suspect, peopl, shot, kill, airport
5	shoot, shooter, attack, terrorist, gun, terror, san, trump, plan, call
6	shoot, vega, mass, las, thousand, victim, kill, california, bar, church
7	victim, shoot, trump, hous, flag, capit, honor, half, presid, staff
8	kill, peopl, white, hous, shoot, shooter, shot, black, polic, man

Table 5: The highest probability stems per topic (for $k = 8$) for MALLET.

Topic	10 Highest Probability Stems
1	victim, famili, prayer, today, thought, life, communiti, violenc, pray, heart
2	polic, report, suspect, shooter, dead, offic, shot, capit, break, multipl
3	shoot, mass, vega, news, gunman, thousand, las, dead, california, die
4	attack, san, trump, orlando, terrorist, plan, bernardino, call, terror, obama
5	shoot, church, shot, live, airport, time, day, texa, fire, talk
6	kill, peopl, hous, white, stop, man, guy, black, murder, colorado
7	shooter, dalla, cop, killer, media, blame, make, show, cnn, post
8	gun, school, high, control, law, parkland, student, nra, kid, arm

Table 6: The highest probability stems per topic (for $k = 8$) for BTM.

C Topic Model Outputs

C.1 Topic Words

Table 5 and Table 6 show the highest probability words per topic (for $k = 8$) for MALLET and BTM, respectively.

C.2 Topic Tweets

We present a sample of tweets belonging to one of the 8 topics assigned by our model.

News.

- HAPPENING NOW: Multiple people wounded in shooting at Colorado Walmart
- 3 people are reported dead in #FtLauderdaleshooting
- UPDATE: Baton Rouge 2 Officers confirmed dead 7 officers have been shot in total. Details are slowly coming in.
- San Francisco police responding to reports of a shooting at a UPS facility, according to multiple reports.
- BREAKING: Police confirm several fatalities in #Annapolis newsroom shooting and multiple people seriously injured. The suspect is in custody. @SkyNewsAust

Investigation.

- Alleged synagogue shooter charged in deaths of 11 people
- Michigan prosecutor: Suspect in Kalamazoo rampage that killed 6 admitted to shootings.
- #COSshooting #PlannedParenthood It's over...suspect walked out and they apprehended him-no altercation in arrest. Suspect turned himself in
- Capital Gazette shooter has been identified as Jarrod W. Ramos, 38, who had previously filed a defamation lawsuit against the paper and a columnist in 2012.
- Waffle House gunman's father facing charges for GIVING him gun used to kill four

Shooter's identity & ideology.

- HATE CRIME : Fresno Islamic Killer Referred to White People as "Devils"
- To say that extremist Muslims represent all Muslims is like saying the gunmen in Colorado springs represents all Christians
- So who has #BatonRouge blamed for this shooting Imperialist Obama BLM Movement The Black Man Or the isolated mentally ill white lone wolf
- Again, the Lafayette white killer is a "lone wolf" but the Chattanooga Arab killer is an entire religion. Aggressively insane troll logic.
- Who is surprised that the San Bernardino shooters were radicalized Muslims?

Victims & location.

- 12 Killed in California Shooting; Gunman Targeted Bar in Thousand Oaks
- Synagogue shooting in Pittsburgh: what we know
- Navy Veteran Who Survived Afghanistan Dies in Las Vegas Shooting
- Aaron Feis, who died in the mass shooting at Marjory Stoneman Douglas High, was praised by college recruiters and former players for helping dozens of high school athletes land a chance to play in college.
- Texas church, site of deadly massacre, to be demolished (Via WLTX 19)

Laws & policy.

- This Parkland Tragedy was completely a security meltdown in or security officials. 100% preventable. Gun laws had nothing to do with this massacre. But gun control could have diminish the carnage. Two different things...
- NRA allowed acts like #Chattanooga to become commonplace. Their lobbying permits people on the terror watch list to buy guns. Remember that.
- I will not just #prayforsutherlandsprings, today I vote for @PhilMurphyNJ and stronger gun control laws in NJ #ElectionDay #GunControlNow
- Again the mental health flags were waving about shooter in Santa Fe no one acted. By now, if ones in charge cared and wanted to protect students, Every School would have had a security assessment and have hardened access points. Can never stop these but can make it harder.
- This is a mental health issue, security issue AND a gun issue. Our government has taken action on NONE of these to protect our students. So clearly we are not being heard, and our kids are being executed. What do we do now? #EnoughIsEnough #DoSomething #SantaFe #WeMustBeHeard

Solidarity.

- Our thoughts and prayers go out the victims and their families involved in the #Chattanooga tragedy 2nite.
- Praying for the loved ones of the victims of the shooting yesterday at the synagogue in Pittsburgh.
- Our prayers go out to the victims, families, friends & everyone affected in #SanBernardino #BeSafe #BeAware
- My heart goes out to the friends and family of the victims in that parkland shooting :-(
- Our hearts goes to #UmpquaCommunityCollege in the #oregonshooting #ChristCenteredEnt is praying 4 u #AllLivesMatter

Remembrance.

- @HarfordCC will honor victims @UmpquaCC TODAY by observing a National Moment of Silence 1:45PM in Quad. Students should attend #IamUCC
- Photo: US flag flying at half-staff at White House for victims of Roseburg, Ore., school shooting
- More than 100 Romans gather in solidarity against hate, honor victims of Orlando shooting
- Trump Denies Request To Lower Flags To Honor Capital Gazette Shooting Victims #DouchebagDon #CrookedTrump @FLOTUS @realDonaldTrump
- Live from #ChattanoogaShootings memorial next on #11Alive

Other.

- I'm sure coming from you, this has nothing to do with the Fresno shooting?
- I realize the dude from the Waffle House shooting did a miraculous thing disarming the shooter, but if I see the gaping wound on his forearm one more time I'm gonna lose my mind.
- The only thing that stops a bad guy with a gun, is a good guy with a gun!!! #Kalamazoo
- The little clip I saw of what DT just said about #PittsburghShooting #PittsburghSynagogueShooting was ok (I guess)
- But when is someone gonna reassure me a planned parenthood attack won't happen

D Topic Model Evaluation

In the next two subsections we describe the tasks which we crowdsourced to compare the three topic models: MALLET, BTM and our embedding-based model.

D.1 Word intrusion

Our word intrusion task is the same as is described in Chang et al. (2009). Our topic-word distance metric for MALLET and BTM is probability (we use the exact topic-word matrix that these models output) and for our model it is cosine distance. We created 2850 experimental items (i.e. sets of words) with the following procedure:

1. Sample a model M, a k (between 6 – 10) and a topic x ranging between $1 - k$.
2. For a given choice of M, k and x,
   (a) sample 5 words among the closest 10 words to x.
   (b) sample one word that is among the 5% of the furthest words from x but also among the 5% of the closest words to another topic.
3. Shuffle the words.

Turkers were asked to pick the odd word out among a set of words. Each MTurk HIT consisted of 6 sets of words created via this procedure.

D.2 Tweet intrusion

Since we only assign one topic to each tweet, we would like to evaluate how coherent the tweets are that are assigned to the same topic. Therefore, we define the tweet intrusion task analogously to word intrusion. Our distance metric in this case is the ratio of proximities (probability for MALLET and BTM and cosine for our model) between the closest topic and second closest topic — this value quantifies the proximity to the closest topic as well as how uniquely close that topic is (in contrast to the second topic). We created 1800 experimental items via the following procedure:

1. Sample a model M, a k (between 6 – 10) and a topic x ranging between $1 - k$.
2. For a given choice of M, k and x,
   (a) sample 3 tweets among the closest 1% of tweets to x.
   (b) sample one tweet that is among the 1% of the furthest tweets from x but also among the 1% of the closest tweets to another topic.
3. Shuffle the tweets.

Turkers were asked to pick the odd tweet out from these tweets. Each MTurk HIT consisted of three sets of tweets.

E Emotion Lexicon

The following words were the final stems in our emotional lexicon.

positive love, friend, pray, thought, affect, bless, god, pleas, communiti, hope, stand, thank, help, condol, will, comfort, time, strong, work, support, effect, strength, feel, peac, word, rest, give, great, action, good

negative hate, violenc, hatr, of, evil, tragedi, will, word, attack, sad, feel, anger, murder, shoot, massacr, want, need, pain, kill, griev, crime, ignor, victim, lost, grief, senseless, tragic, fear, loss, sick

sadness senseless, loss, tragedi, lost, devast, sad, love, griev, horrif, terribl, pain, violenc, condol, broken, hurt, feel, victim, mourn, horrifi, will, grief, ach, suffer, sick, kill, aw, sicken, evil, massacr, mad

disgust disgust, sick, shame, ignor, wrong, blame, hell, ridicul, idiot, murder, evil, coward, sicken, feel, disgrac, slaughter, action, bad, insan, attack, pathet, outrag, polit, terrorist, mad, damn, lose, shit, lie, asshol

anger gun, will, murder, kill, violenc, wrong, shoot, bad, death, attack, feel, shot, action, arm, idiot, crazi, crimin, terrorist, mad, hell, crime, blame, fight, ridicul, insan, shit, die, threat, terror, hate

fear danger, threat, fear, arm, gun, still, shooter, attack, feel, fight, hide, murder, shot, shoot, bad, kill, chang, serious, violenc, forc, risk, defend, warn, govern, concern, fail, polic, wrong, case, terrorist

trust school, like, good, real, secur, show, nation, don, protect, call, teacher, help, law, great, save, true, wonder, respons, sad, answer, person, feel, safe, thought, continu, love, guard, church, fact, support

The following words were used as seeds to generate this lexicon, as described in the main text.

positive love, donat, heart, thought, strength, bless, solidar

negative hatr, hate, griev, grief, wrong

sadness mourn, sadden, griev, grief, sad, suffer, affect, broken, senseless, loss, heartbroken

disgust disgust, disgrac, shame, gut, slaughter, sicken, sick, ill, lunat, coward

anger deserv, lynch, gang, threat, mad, sicken, harm, enforc, firearm, ridicul, assault

fear risk, hide, danger, warn, fear

trust secur, coach, safe, hero, nation

F Most Partisan Phrases

F.1 Most Partisan Phrases Overall

We list the 20 most Democrat and Republican unigrams and bigrams that occur at least 100 times in tweets about a particular event. The number in brackets indicates the z-score of the log odds of these words (Monroe et al., 2008) — values with an absolute value greater than 2 are significantly partisan.

Chattanooga. *Most Republican phrases:* obama (7.41), gun free (5.89), zone (5.70), free (5.69), free zone (5.53), flag (5.33), #tcot (5.33), #chattanoogaattack (5.13), #wakeupamerica (4.94), islam (4.69), #obama (4.01), #islam (3.83), attack (3.67), #gunfreezon (3.66), lower (3.66), liber (3.54), arm (3.44), workplac (3.32), white hous (3.23), workplac violenc (3.21)
Most Democrat phrases: blame georg (-8.62), bush invad (-8.62), invad iraq (-8.60), invad (-8.60), base lie (-8.55), georg bush (-8.53), war base (-8.51), lie happen (-8.51), georg (-8.35), iraq war (-8.31), iraq (-8.24), bush (-7.77), lie (-7.45), #charleston (-6.99), mass (-6.82), #lafayett (-6.48), happen (-6.19), #charlestonshoot (-5.96), blame (-5.40), #gunsens (-5.09)

Roseburg. *Most Republican phrases:* obama (8.02), #2a (6.28), #obama (6.01), gun free (5.37), #tcot (5.32), free (5.22), christian (5.09), zone (5.08), chicago (5.06), shooter (4.78), free zone (4.75), #gunfreezon (4.04), agenda (3.95), religion (3.85), train (3.79), liber (3.73), #christian (3.71), secur (3.45), guard (3.40), skarlato (3.39)
Most Democrat phrases: #gunsens (-4.69), heart (-4.59), gun nut (-3.92), fuck (-3.83), mass (-3.82), gun violenc (-3.81), violenc (-3.80), nra (-3.54), thought (-3.47), nut (-3.28), school (-3.22), #gunviol (-3.10), countri (-3.08), chang (-3.04), congress (-3.02), love (-2.99), vigil (-2.86), mass shoot (-2.75), protest (-2.74), america (-2.63)

Colorado Springs. *Most Republican phrases:* babi (13.62), liber (11.39), kill babi (8.77), polic (8.46), kill (8.30), left (7.42), offic (7.01), bank (6.97), babi kill (6.84), lib (6.42), obama (6.34), #activeshoot (6.23), #tcot (6.13), activ (6.12), parenthood kill (6.01), report (5.99), injur (5.90), #break (5.81), activ shooter (5.75), plan (5.72)
Most Democrat phrases: terrorist (-13.88), terror (-9.71), attack (-9.52), terrorist attack (-9.09), white (-8.33), #plannedparenthoodshoot (-7.63), #plannedparenthood (-7.29), gop (-7.04), domest (-6.67), candid (-6.40), #gopdeb (-6.32), #standwithpp (-6.27), attack #plannedparenthood (-6.23), women (-5.84), radic (-5.74), #gop (-5.39), defund (-5.05), christian (-5.02), call (-4.89), aliv (-4.73)

San Bernardino. *Most Republican phrases:* obama (12.59), attack (12.08), #tcot (9.33), terrorist (9.26), islam (9.20), terrorist attack (8.65), muslim (7.48), liber (6.94), #2a (6.82), climat (6.62), climat chang (6.34), blame (5.91), #obama (5.79), islam terrorist (5.65), #pjnet (5.61), #wakeupamerica (5.35), workplac violenc (5.19), foxnew (5.05), call (5.02), vet (4.89)
Most Democrat phrases: mass (-13.70), mass shoot (-10.50), #gunsens (-10.35), shoot (-7.97), disabl (-7.17), gop (-5.89), development (-5.82), fuck (-5.73), development disabl (-5.67), #gopdeb (-5.64), heart (-5.37), center (-5.34), thought (-5.29), day (-4.96), action (-4.93), domest (-4.78), #gunviol (-4.73), congress (-4.50), sick (-4.40), normal (-4.32)

Kalamazoo. *Most Republican phrases:* michigan (4.02), #break (3.39), ap (3.37), barrel (3.03), counti (2.98), cracker (2.98), cracker barrel (2.93), suspect (2.80), polic (2.71), area (2.56), dead (2.53), random shoot (2.49), mich (2.43), year girl (2.33), 14 year (2.24), victim (2.22), shoot (2.22), charg (2.20), 7 (2.20), counti michigan (2.18)
Most Democrat phrases: gun (-4.37), white (-4.11), mass (-4.00), terrorist (-3.63), america (-3.39), #gunviol (-3.08), white man (-2.72), mental (-2.72), gun violenc (-2.60), coverag (-2.58), #kalamazooshoot (-2.52), ill (-2.42), mental ill (-2.38), unarm (-2.37), guy (-2.27), countri (-2.27), white male (-2.24), black (-2.20), violenc (-2.02), talk (-1.88)

Orlando. *Most Republican phrases:* islam (59.38), attack (53.07), terrorist (48.13), obama (47.01), #tcot (39.27), fbi (38.32), terror (37.21), blame (36.48), terror attack (35.32), terrorist attack (33.95), mateen (32.23), hillari (32.04), jihad (31.77), shooter (31.72), isi (31.63), radic (31.07), democrat (30.47), killer (29.49), liber (29.01), islam terror (28.69)
Most Democrat phrases: victim (-54.39), love (-40.34), heart (-34.47), hate (-34.40), violenc (-31.13), communiti (-29.11), #loveislov (-28.11), gun violenc (-27.06), lgbt (-26.84), lgbtq (-26.44), donat (-24.12), #prayfororlando (-24.04), mass (-

23.94), #weareorlando (-23.04), #lgbt (-22.97), #endgunviol (-22.94), #gunviol (-22.68), thought (-22.49), peopl (-21.87), fuck (-21.81)

Dallas. *Most Republican phrases:* obama (23.09), #bluelivesmatt (17.78), offic (12.76), #obama (11.52), white (10.33), polic offic (10.13), hillari (10.10), kill white (9.77), racist (9.09), foxnew (9.00), shot (8.98), offic shot (8.95), #tcot (8.81), hate crime (8.67), democrat (8.51), blame gun (8.49), crime (8.47), white cop (8.46), cop (8.30), 5 (8.03)
Most Democrat phrases: #altonsterl (-21.29), #philandocastil (-21.13), #altonsterl #philandocastil (-16.32), guy gun (-15.96), good guy (-15.26), violenc (-14.83), gun (-14.54), open carri (-13.00), guy (-12.63), carri (-12.47), open (-10.82), #gunviol (-10.54), good (-10.53), stop (-10.23), #philandocastil #altonsterl (-10.07), nra (-10.02), gun violenc (-9.80), #nra (-9.23), #disarmh (-8.64), answer (-8.54)

Baton Rouge. *Most Republican phrases:* #bluelivesmatt (8.69), obama (8.13), islam (5.87), nation islam (5.62), #obama (5.17), shot (5.06), cop killer (5.06), killer (5.05), terrorist (4.99), nation (4.74), offic shot (4.63), hillari (4.34), #backtheblu (4.25), #tcot (4.05), offic (3.99), thug (3.96), islam member (3.79), 3 (3.53), #trumppence16 (3.45), democrat (3.41)
Most Democrat phrases: gun (-8.88), violenc (-8.34), gun violenc (-6.65), #nra (-6.19), guy gun (-5.77), #altonsterl (-5.57), open carri (-5.48), assault (-5.48), weapon (-5.30), good guy (-5.14), blame (-4.88), citizen (-4.87), assault weapon (-4.84), carri (-4.74), race relat (-4.74), relat (-4.71), open (-4.61), guy (-4.58), civilian (-4.47), #enough (-4.46)

Burlington. *Most Republican phrases:* cetin (5.12), arcan cetin (5.06), arcan (5.01), muslim (4.72), turkish (4.57), vote (4.37), turkey (4.32), hispan (4.14), immigr (4.04), citizen (3.79), hillari (3.64), elect (3.45), immigr turkey (3.19), turkish muslim (3.14), id (3.11), ed arcan (3.05), id ed (3.03), shooter id (3.00), citizen vote (2.97), cetin immigr (2.96)
Most Democrat phrases: victim (-4.05), gun (-3.98), famili (-2.97), thought (-2.65), peopl (-2.48), dead (-2.36), heart (-2.32), seattl (-2.30), violenc (-2.23), larg (-2.11), mile (-2.10), shooter larg (-2.08), latest (-2.05), day (-2.04), tonight (-2.03), safe (-2.01), watch (-2.00), communiti (-1.98), shoot (-1.96), #break (-1.94)

Fort Lauderdale. *Most Republican phrases:* garag (4.54), shot fire (3.45), terrorist (3.14), fox (2.93), free zone (2.76), attack (2.71), gun free (2.70), fire (2.62), zone (2.61), free (2.43), muslim (2.36), shot (2.31), obama (2.09), park (2.08), terrorist attack (2.08), park garag (2.06), shooter (2.01), fox news (2.01), terror (1.93), report shot (1.81)
Most Democrat phrases: stop (-2.65), violenc (-2.52), custodi 9 (-2.25), gun (-2.14), mass (-2.12), fll (-2.10), 9 injur (-2.05), week (-2.00), multipl peopl (-1.91), 2017 (-1.86), airport suspect (-1.85), stay safe (-1.83), love (-1.78), heart (-1.70), safe (-1.70), report fire (-1.66), smh (-1.65), local (-1.64), msnbc (-1.64), thought (-1.61)

Fresno. *Most Republican phrases:* akbar (4.67), allahu (3.84), allahu akbar (3.76), yell (3.32), yell allahu (2.98), ap (2.61), suspect yell (2.38), allah (2.33), shout (2.25), terror attack (2.06), shout allahu (2.06), allah akbar (2.03), terrorist (1.98), terrorist attack (1.97), muslim (1.96), kill suspect (1.94), msm (1.83), islam (1.81), god (1.80), akbar hate (1.79)
Most Democrat phrases: chief (-3.97), victim (-3.89), polic (-3.74), famili (-3.65), fatal (-3.62), offic (-2.98), downtown (-2.92), dyer (-2.75), men (-2.71), polic chief (-2.70), gunman (-2.69), tuesday (-2.47), gun (-2.29), shoot (-2.22), suspect custodi (-2.17), kill california (-2.10), mass (-2.09), sad (-2.08), white men (-1.95), kill hate (-1.95)

San Francisco. *Most Republican phrases:* polic (4.85), multipl (4.41), report (4.32), pistol (4.18), assault pistol (4.12), 4 injur (3.41), shooter (3.33), assault (3.24), free (3.19), warehous (3.05), injur (3.01), facil 4 (2.98), multipl victim (2.90), report facil (2.85), stolen (2.82), shot facil (2.76), hospit (2.68), law (2.60), gun law (2.59), compani (2.51)
Most Democrat phrases: today (-4.82), mass (-4.18), sf (-4.07), die (-3.47), mention (-3.29), coverag (-2.97), yesterday (-2.95), america (-2.75), #upsshoot (-2.49), #sf (-2.40), gun violenc (-2.40), mass shoot (-2.40), barclay (-2.31), virginia (-2.30), #up (-2.20), violenc (-2.17), morn (-2.17), shoot (-2.17), gop (-2.07), peopl kill (-2.04)

Vegas. *Most Republican phrases:* shooter (41.28), fbi (36.07), video (31.96), isi (31.08), democrat (26.78), paddock (26.34), liber (26.07),

multipl (25.37), antifa (23.76), multipl shooter (23.61), #maga (22.44), cbs (21.55), truth (21.49), mandalay (21.06), left (20.68), islam (20.57), guard (19.73), dem (19.51), hillari (19.02), proof (18.78)
Most Democrat phrases: #guncontrolnow (-48.73), gun (-42.42), nra (-34.84), terrorist (-33.42), gun violenc (-31.92), #guncontrol (-30.54), violenc (-29.87), domest (-29.17), mass (-28.41), white (-27.19), terror (-24.88), domest terror (-24.68), congress (-24.64), mass shoot (-23.77), gop (-22.84), thought (-22.82), #nra (-22.19), talk (-22.06), fuck (-21.43), talk gun (-21.10)

Thornton. *Most Republican phrases:* multipl (6.04), suspect (5.24), parti (5.06), multipl parti (5.03), break (4.90), news (4.80), injur (4.36), dead (4.23), report (3.98), updat (3.87), polic (3.84), suspect custodi (3.75), activ (3.71), chicago (3.57), detail (3.55), 2 (3.52), report multipl (3.28), #break (3.27), video (3.09), activ shooter (3.08)
Most Democrat phrases: white (-7.61), guy (-5.37), gun (-5.24), white guy (-3.92), bibl (-3.75), terrorist (-3.75), week (-3.61), white man (-3.48), good guy (-3.46), stack bibl (-3.33), live stack (-3.29), stack (-3.28), furnitur (-3.27), terror (-3.16), guy gun (-3.06), penalti (-3.04), bibl furnitur (-3.02), death penalti (-3.01), talk (-3.01), vega (-2.97)

Sutherland Springs. *Most Republican phrases:* shooter (19.89), church shooter (17.26), liber (16.58), antifa (16.38), democrat (15.16), atheist (14.70), attack (14.58), christian (13.31), texa church (13.17), zone (13.15), gun free (13.03), texa (12.66), free zone (12.58), leftist (12.45), illeg (12.40), hero (11.78), free (11.09), citizen (10.60), carri (10.43), #antifa (10.27)
Most Democrat phrases: #guncontrolnow (-17.29), prayer (-16.62), school (-15.11), thought prayer (-13.98), thought (-12.83), girl (-12.76), gun violenc (-12.56), talk (-11.53), mental (-11.29), white (-11.17), church pray (-10.81), concert (-10.79), prosecut (-10.76), #guncontrol (-10.63), mass shoot (-10.52), gop (-10.36), violenc (-10.21), children (-10.03), congress (-9.95), jone (-9.88)

Parkland. *Most Republican phrases:* fbi (30.42), sheriff (25.67), liber (21.67), school (21.47), cruz (19.47), shooter (18.68), #2a (18.39), broward (17.87), fail (17.17), israel (17.12), polic (16.73), deputi (16.03), failur (15.41), democrat (15.41), counti (15.05), #qanon (14.65), enforc (14.40), gun free (14.35), truck (14.30), free zone (14.23)
Most Democrat phrases: #gunreformnow (-26.02), #guncontrolnow (-25.00), #neveragain (-22.39), nra (-18.15), gop (-16.64), gun violenc (-16.45), #parklandstrong (-15.80), #nrabloodmoney (-15.00), vote (-14.48), trump (-13.27), violenc (-13.23), congress (-12.82), ar 15 (-12.74), ar (-12.46), #banassaultweapon (-12.28), #marchforourl (-11.85), survivor (-11.57), support (-11.51), assault (-11.33), fuck (-11.22)

Nashville. *Most Republican phrases:* gun free (11.69), zone (10.94), free zone (10.69), free (10.32), photo shoot (8.02), #wbb #wilsonbrothersband (7.96), #wilsonbrothersband (7.96), #wbb (7.96), band photo (7.96), brother band (7.96), wilson brother (7.96), wilson (7.96), fbi (7.94), band (7.72), gun (7.22), photo (6.95), law (6.65), brother (6.32), liber (6.21), hous gun (6.20)
Most Democrat phrases: black (-11.21), white (-9.41), trump (-8.06), terrorist (-7.67), tweet (-6.16), hero (-6.13), american (-6.05), shaw (-5.77), black man (-5.73), jame shaw (-5.58), jr (-5.44), jame (-5.42), shaw jr (-5.41), mention (-5.30), domest (-5.05), bond (-5.02), unarm (-4.74), domest terrorist (-4.69), black peopl (-4.63), man (-4.63)

Santa Fe. *Most Republican phrases:* shotgun (10.99), revolv (7.57), illeg (7.10), shooter (6.20), liber (6.20), ban (6.14), metal detector (6.12), detector (6.00), secur (5.93), metal (5.87), truck (5.79), rack (5.53), high school (5.45), bomb (5.43), high (5.33), leftist (5.30), gun rack (5.27), law stop (5.15), updat (5.15), rifl (5.12)
Most Democrat phrases: #guncontrolnow (-12.17), #gunreformnow (-10.36), #neveragain (-9.79), #enoughisenough (-8.90), nra (-8.57), children (-7.85), vote (-7.69), gun violenc (-7.59), america (-6.75), thought prayer (-6.72), violenc (-6.66), congress (-6.63), thought (-6.50), #enough (-6.21), fuck (-6.11), white (-6.01), #parkland (-5.74), gun (-5.64), republican (-5.44), gop (-5.40)

Annapolis. *Most Republican phrases:* blame (8.23), blame trump (7.82), liber (5.86), maryland (5.76), reuter (5.70), reuter editor (5.62), apolog (5.34), #break (5.23), editor apolog (5.22), disciplin (4.90), apolog disciplin (4.84), hat (4.84), dis-

ciplin blame (4.74), maga hat (4.60), claim shooter (4.48), fals (4.39), polic (4.38), #fakenew (4.36), democrat (4.32), wore maga (4.32)
Most Democrat phrases: journalist (-7.03), enemi (-6.10), press (-5.62), lower flag (-5.52), request lower (-5.36), request (-5.02), gazett victim (-4.55), declin request (-4.46), memori capit (-4.42), lower (-4.40), enemi peopl (-4.39), press enemi (-4.37), white (-4.35), flag memori (-4.34), declin (-4.30), flag (-4.13), memori (-4.12), trump declin (-4.09), mass (-4.08), #guncontrolnow (-4.05)

Pittsburgh. *Most Republican phrases:* moment silenc (16.24), silenc (15.99), interrupt (14.74), interrupt moment (14.68), scream (13.09), moment (12.37), march (12.13), blackburn (11.85), leftist (11.73), silenc life (11.59), blame trump (11.28), silenc synagogu (11.20), protest (10.78), protest interrupt (10.56), rabbi blame (9.64), leftist interrupt (9.55), scream leftist (9.55), rage scream (9.52), horribl rage (9.43), scream insult (9.40)
Most Democrat phrases: violenc (-6.73), heart (-6.29), supremacist (-6.05), white supremacist (-6.05), muslim (-5.96), white (-5.94), mr (-5.88), trump vile (-5.86), result (-5.84), of (-5.80), result of (-5.80), inevit (-5.80), inevit result (-5.78), group (-5.77), synagogu inevit (-5.77), vile nation (-5.73), muslim group (-5.73), of trump (-5.71), roger (-5.69), massacr heart (-5.63)

Thousand Oaks. *Most Republican phrases:* california (16.51), zone (12.80), free (12.40), gun free (12.22), free zone (11.80), bar (9.98), california bar (9.61), strictest (9.01), strictest gun (8.52), men (7.63), #foxnew (7.33), killer ian (7.12), california strictest (7.11), fear resid (7.11), report killer (7.08), prayer massacr (7.04), communist (7.04), long mock (7.00), mock hope (6.80), ian long (6.77)
Most Democrat phrases: inact (-17.50), pattern (-17.46), pattern inact (-17.45), shoot pattern (-17.43), shoot (-16.92), januari (-13.09), inact januari (-12.92), #guncontrolnow (-8.09), mass shoot (-7.89), day (-7.34), fuck (-6.94), nra (-6.94), violenc (-6.71), mother (-6.64), mass (-6.59), thought (-6.44), high (-6.37), januar (-6.22), inact januar (-6.15), #potus (-6.05)

F.2 Most Partisan Phrases Per Topic

Topic colors
shooter’s identity & ideology
news
victims & location
laws & policy
investigation
solidarity
remembrance
other

Republican	Democrat
#obama (0.45), toler (0.45), presid (0.45), #wakeupamerica (0.45), celebr (0.45)	blame georg (-0.94), invad iraq (-0.94), bush invad (-0.94), iraq war (-0.94), base lie (-0.94)
zone (0.35), fox news (0.34), 5th victim (0.33), #tcot (0.33), class (0.32)	mass (-0.68), #prayforchattanooga (-0.37), horrif (-0.37), #breakingnew (-0.35), 4 dead (-0.32)
victim (0.25), dead (0.18), 5 (0.15), 4 marin (0.09), gunman (0.09)	skip (-0.29), skip well (-0.25), well (-0.23), carson (-0.23), holmquist (-0.23)
#chattanoogaattack (0.33), #wakeupamerica (0.33), clinton (0.33), safe (0.33), muhammad (0.33)	nra (-0.99), nut (-0.97), #charleston (-0.86), #lafayett (-0.82), gun violenc (-0.71)
flag (0.41), terrorist (0.32), fire (0.31), search (0.30), navi (0.30)	#fbi (-0.47), special (-0.43), treat (-0.38), natur (-0.35), agent (-0.31)
#rednationris (0.44), #islam #rednationris (0.44), #chattanoogashoot #chattanoogastrong (0.44), marin #wakeupamerica (0.44), #chattanoogastrong #isi (0.44)	#guncontrol (-0.95), #charleston (-0.95), #sandrabland (-0.82), gun violenc (-0.81), #charlestonshoot (-0.72)
decid (0.28), refus (0.28), command (0.28), troop (0.28), kill tennesse (0.28)	local (-0.63), mourn (-0.56), victim shoot (-0.52), tune (-0.46), communiti (-0.41)
#teardownthismosqu (0.42), #islamocar (0.42), #stoprush (0.42), #obama (0.42), #wakeupamerica (0.42)	middl (-0.75), #nra (-0.75), mass (-0.73), #aurora (-0.73), #charleston (-0.71)

Table 7: Most partisan phrases per topic for **Chattanooga**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
skarlato (0.46), #2a (0.46), gun control (0.39), ask (0.37), agenda (0.34)	#gunsens (-0.65), news (-0.55), pro gun (-0.42), pro (-0.41), mass (-0.39)
ore ap (0.27), secur (0.27), author (0.26), #umpquacommunitycolleg (0.23), thought (0.23)	#liveonk2 (-0.53), #gunsens (-0.52), merci (-0.50), center (-0.46), 9 (-0.45)
colleg (0.12), communiti colleg (0.05), communiti (0.05), chris (0.00)	chris (0.00), communiti (0.05), communiti colleg (0.05), colleg (0.12)
liber (0.67), guard (0.65), chicago (0.63), ban (0.58), secur (0.58)	gun nut (-0.65), fuck (-0.58), legisl (-0.57), #sandyhook (-0.55), congress (-0.54)
father (0.51), christian (0.50), releas (0.42), identifi (0.30), 2 (0.30)	sheriff (-0.41), gun (-0.25), year (-0.21), 20 year (-0.17), victim (-0.13)
famili involv (0.42), 9 (0.41), pray famili (0.40), school communiti (0.38), 7 (0.38)	#gunviol (-0.52), fuck (-0.50), #gunsens (-0.40), chang (-0.38), gun violenc (-0.37)
#2a (0.80), #tcot (0.76), special (0.70), rt (0.66), media (0.54)	3 (-0.45), tonight (-0.39), 11 (-0.37), dougla (-0.36), dougla counti (-0.35)
#obama (0.69), chicago (0.68), #2a (0.57), blame (0.56), stay (0.52)	countri (-0.61), nut (-0.53), #roseburgoregon (-0.49), world (-0.43), thought (-0.42)

Table 8: Most partisan phrases per topic for **Roseburg**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
regist vote (1.01), young (1.01), shop (1.01), mcdonald (1.01), #liber (1.01)	parenthood #gopdeb (-0.37), #carlyfiorina (-0.37), health clinic (-0.37), #tedcruz (-0.37), morn joe (-0.37)
connect (0.56), #bluelivesmatt (0.56), #tcot (0.54), #new (0.52), shooter surrend (0.47)	#standwithpp (-0.63), #gunsens (-0.62), #terror (-0.60), #plannedparenthoodshoot (-0.58), month (-0.57)
kill (0.17), plan parenthood (0.13), parenthood (0.13), plan (0.11), #plannedparenthood (-0.03)	war (-0.18), veteran (-0.17), thousand (-0.10), iraq (-0.08), #plannedparenthood (-0.03)
kill babi (0.92), unborn (0.91), parenthood kill (0.88), sell bodi (0.87), gun free (0.83)	#terror (-0.39), focus (-0.39), terrorist attack (-0.39), forc (-0.39), attack women (-0.39)
parenthood employe (0.81), employe (0.72), femal (0.49), gunman attack (0.42), record (0.38)	#plannedparenthood gunman (-0.58), #laquanmcdonald (-0.58), terrorist (-0.47), domest (-0.46), custodi aliv (-0.45)
babi (0.94), #tcot (0.71), pro life (0.63), pro (0.55), #prolif (0.54)	#istandwithpp (-0.39), #standwithpp (-0.39), terror (-0.37), terrorist attack (-0.37), parenthood attack (-0.35)
clinic (0.57), support (0.47), swasey (0.44), hous (0.43), friday (0.40)	carson (-0.42), terrorist (-0.42), cover (-0.41), terror (-0.41), terrorist attack (-0.41)
ppact (0.87), profit (0.84), lib (0.81), chicago (0.76), bodi part (0.74)	4 cop (-0.42), parenthood #gopdeb (-0.42), discuss (-0.42), #terroristattack (-0.42), captur aliv (-0.42)

Table 9: Most partisan phrases per topic for **Colorado Springs**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
moron call (0.44), shooter bonni (0.44), warm (0.44), global warm (0.44), #2a (0.44)	shooter post (-0.85), gun nut (-0.81), ted cruz (-0.76), domest terrorist (-0.74), ted (-0.67)
muslim (0.66), #pjnet (0.66), #sanbernardinoattack (0.66), #tcot #pjnet (0.66), #tcot (0.61)	shoot today (-0.63), #gunsens (-0.62), 355th (-0.56), houston (-0.55), savannah (-0.52)
husband wife (0.40), massacr (0.35), husband (0.30), wife (0.22), calif (0.20)	live (-0.42), mass shoot (-0.27), shoot (-0.24), mass (-0.15), attack (-0.15)
law work (0.67), disarm (0.67), lib (0.67), administr (0.60), pari (0.60)	normal (-0.72), #ppshoot (-0.72), gun nut (-0.64), gop (-0.63), topic (-0.62)
chief burguan (0.59), foxnew (0.51), #sanbernardinoattack (0.50), #tcot (0.48), devout (0.48)	presser (-0.51), expect (-0.46), purchas legal (-0.44), suspect home (-0.44), detain (-0.41)
#tcot (0.75), sanbernardinopd (0.67), terrorist attack (0.55), job (0.55), polic offic (0.53)	#gunsens (-0.63), savannah (-0.63), #gunviol (-0.54), development disabl (-0.54), send love (-0.52)
#tcot (0.70), #pjnet (0.70), #tcot #pjnet (0.70), #ccot (0.70), wh (0.47)	donat (-0.68), rais (-0.53), 10 (-0.44), public (-0.42), governor (-0.40)
blame gun (0.70), #wakeupamerica (0.70), #liber (0.70), pc (0.70), fast (0.63)	#endgunviol (-0.69), movi (-0.69), 352 (-0.69), #sandyhook (-0.59), shoot day (-0.59)

Table 10: Most partisan phrases per topic for **San Bernardino**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
polit (0.66), motiv (0.65), violenc (0.46), obama (0.26), peopl (0.23)	coverag (-0.44), news (-0.44), open (-0.44), cover (-0.44), white male (-0.37)
counti michigan (0.41), cracker (0.27), random shoot (0.27), cracker barrel (0.27), barrel (0.27)	coverag (-0.59), public (-0.52), #michigan (-0.50), connect (-0.50), pray (-0.43)
safeti (0.45), pass (0.34), suspect (0.26), prevent (0.24), #uber (0.22)	talk (-0.36), #guncontrol (-0.34), prayer (-0.32), #kalamazooshoot (-0.29), gun violenc (-0.27)
saturday (0.45), ap (0.43), mich (0.38), 5 (0.36), 14 year (0.32)	black (-0.54), white man (-0.53), white (-0.41), white male (-0.36), pick (-0.35)
crazi (0.53), peopl kill (0.32), prayer victim (0.29), #prayforkalamazoo (0.27), 6 (0.27)	#gunviol (-0.53), place (-0.53), america (-0.45), work (-0.43), gun violenc (-0.36)
#kalamazoostrong (0.29), victim (0.21), #kalamazooshoot (0.18), michigan (0.12), kill (0.11)	rememb (-0.35), mass (-0.35), gun (-0.12), saturday (-0.05), 6 (0.00)
chicago (0.74), cracker barrel (0.62), barrel (0.62), cracker (0.62), 1 (0.44)	male (-0.39), white male (-0.39), gun violenc (-0.39), mass shooter (-0.39), matter (-0.31)

Table 11: Most partisan phrases per topic for **Kalamazoo**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
gun boston (0.47), state dept (0.47), boston pressur (0.47), attack solut (0.47), gohmert gun (0.47)	#stoprush #uniteblu (-0.92), #connecttheleft (-0.92), #p2 #connecttheleft (-0.92), muslim allianc (-0.92), lgbtq hate (-0.92)
#homelandsecur (0.61), #pjnet (0.61), investig islam (0.61), #trump2016 (0.61), park spot (0.61)	#floridashoot #puls (-0.78), derail (-0.78), maryland (-0.78), #enough (-0.78), bbcworld (-0.78)
regist democrat (0.69), shooter regist (0.69), zone state (0.69), state law (0.69), blog gay (0.69)	#gay #new (-0.70), connecticut nativ (-0.70), #new #glbt (-0.70), #stoprush #uniteblu (-0.70), #lgbtnew (-0.70)
help shooter (0.80), sourc suggest (0.80), dc sourc (0.80), terrorist work (0.80), leftist (0.80)	#lalovesorlando (-0.59), implor (-0.59), implor fight (-0.59), tragedi implor (-0.59), 91 (-0.59)
break insid (0.49), #8230 (0.49), hous #8230 (0.49), #8230 #8217 (0.49), shooter make (0.49)	bump poll (-0.89), give bump (-0.89), insid disappoint (-0.89), poll report (-0.89), disappoint give (-0.89)
reaction tragic (0.92), wrong reaction (0.92), search massacr (0.92), kohn (0.92), sick muslim (0.92)	lgbtqia (-0.47), lgbtqia communiti (-0.47), inform horrif (-0.47), readingatthedisco (-0.47), happen omar (-0.47)
netanyahu releas (0.80), lgbt hop (0.80), obama contact (0.80), hop make (0.80), trump #nrabangunfreezon (0.80)	#stoprush #uniteblu (-0.58), demand end (-0.58), #connecttheleft (-0.58), #p2 #connecttheleft (-0.58), #gay #new (-0.58)
#americafirst (0.80), killari (0.80), told word (0.80), carri terror (0.80), #buildthewal (0.80)	learn hate (-0.59), hint wasn (-0.59), hate hint (-0.59), wasn osama (-0.59), shot girl (-0.59)

Table 12: Most partisan phrases per topic for **Orlando**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
ag lynch (0.45), doj (0.45), lectur (0.45), #maga (0.45), #dallaspoliceshoot #bluelivesmatt (0.45)	#disarmh (-0.90), communiti color (-0.90), #disarmh address (-0.90), address #gunviol (-0.90), 2 #disarmh (-0.90)
#pjnet (0.57), mt prayer (0.57), #bluelivesmatt #pjnet (0.57), panther (0.57), #tcot (0.57)	lawrenc (-0.73), npr (-0.69), bbc (-0.63), #enough (-0.60), read (-0.58)
3 (0.36), attack (0.33), #dallasshoot (0.31), iraq (0.28), live (0.23)	2 (-0.38), soldier (-0.33), shot (-0.29), show (-0.23), surviv (-0.18)
shot arm (0.81), 12 shot (0.81), #fbi (0.81), #dallaspoliceshoot #bluelivesmatt (0.81), black caucus (0.81)	scrutini (-0.58), #disarmh (-0.58), denomin (-0.58), beget (-0.58), gun answer (-0.58)
comey (0.67), recommend (0.67), #tcot (0.67), #obama (0.67), sex (0.67)	plaster (-0.62), #altonsterl #philandocastil (-0.58), dallaspd man (-0.58), innoc man (-0.54), #philandocastil (-0.53)
#pjnet (0.64), mt prayer (0.64), #bluelivesmatt #pjnet (0.64), sam (0.64), #maga (0.64)	poc (-0.74), #blacklivesmatt #altonsterl (-0.74), #endgunviol (-0.69), #gunsens (-0.66), #gunviol (-0.64)
lectur (0.55), #maga (0.55), agenda (0.55), incit (0.55), #bluelivesmatt #dallaspoliceshoot (0.55)	#gunviol (-0.73), program (-0.72), #philandocastil #dallaspoliceshoot (-0.67), brother (-0.57), red (-0.57)
#maga (0.65), alabama (0.65), #wakeupamerica (0.65), mr presid (0.65), miss alabama (0.65)	#disarmh (-0.74), carri law (-0.74), access (-0.74), famili friend (-0.74), #stopgunviol (-0.74)

Table 13: Most partisan phrases per topic for **Dallas**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
#orlando (0.46), #maga (0.46), strike (0.46), 3 dead (0.46), assassin (0.46)	peopl blame (-0.92), crazi (-0.82), polic race (-0.81), blame #blacklivesmatt (-0.80), report polic (-0.79)
#pjnet (0.52), #tcot (0.52), realdonaldtrump (0.52), servic (0.52), #backtheblu (0.52)	polic arriv (-0.79), #altonsterl (-0.49), msnbc (-0.46), arriv (-0.46), sterl (-0.42)
killer (0.30), video (0.26), identifi (0.24), gunman (0.16), marin serv (0.12)	veteran (-0.25), jackson (-0.25), mo (-0.25), heartbreak (-0.25), citi mo (-0.25)
#bluelivesmatt (0.72), policemen (0.70), earth (0.64), liber (0.61), order (0.60)	guy gun (-0.56), #enough (-0.56), thought prayer (-0.56), #gun (-0.48), good guy (-0.48)
shooter sovereign (0.57), #breakingnew (0.57), fbi (0.38), citi missouri (0.33), la (0.33)	msnbc (-0.59), identifi shooter (-0.46), matter (-0.43), pd (-0.43), live (-0.42)
#backtheblu (0.61), leadership (0.61), #thinbluelin (0.53), killer (0.51), 2 offic (0.50)	gun violenc (-0.78), #gunviol (-0.78), #disarmh (-0.78), #altonsterl (-0.66), news polic (-0.66)
offic shot (0.40), realdonaldtrump (0.37), #bluelivesmatt (0.35), 7 (0.33), fallen (0.31)	#altonsterl (-0.62), funer (-0.42), montrel (-0.35), black (-0.30), moment (-0.29)
#trumppence16 (0.61), hillari (0.61), lectur (0.61), hillaryclinton (0.50), assassin (0.50)	#enough (-0.71), #nra (-0.67), #philandocastil (-0.66), unjustifi (-0.64), open carri (-0.58)

Table 14: Most partisan phrases per topic for **Baton Rouge**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
turkey (0.19), hillari support (0.19), turkish muslim (0.19), shooter arcan (0.19), immigr turkey (0.19)	gun (-0.47), shoot (-0.45), trump (-0.39), peopl (-0.38), news (-0.29)
zone (0.60), free zone (0.60), gun free (0.60), free (0.60), cetin (0.52)	famili (-0.55), watch (-0.53), latest (-0.46), 4 women (-0.32), male (-0.29)
victim (-0.28)	victim (-0.28)
shooter citizen (0.26), illeg (0.26), id (0.26), citizen vote (0.24), 3 elect (0.23)	support (-0.39), peopl (-0.36), gun (-0.35), control (-0.28), law (-0.26)
turkish muslim (0.41), hillari (0.41), voter (0.41), clinton (0.41), ed arcan (0.38)	motiv (-0.47), komo (-0.44), gun (-0.44), activ shooter (-0.40), 3 (-0.40)
pray (0.34), peopl (0.27), sad (0.24), prayer victim (0.21), washington (0.16)	time (-0.45), gun (-0.44), love (-0.30), affect (-0.30), mass (-0.29)
victim (-0.25)	victim (-0.25)
attack (0.49), polic (0.40), muslim (0.33), hispan (0.27), live (0.24)	seattl (-0.67), fuck (-0.35), news (-0.31), hope (-0.27), gun (-0.26)

Table 15: Most partisan phrases per topic for **Burlington**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
attack (0.24), terrorist (0.20), terrorist attack (0.16), ft airport (0.15), ft (0.10)	shooter (-0.22), white (-0.08), muslim (-0.01), terror (-0.01), obama (0.05)
airport victim (0.59), obama (0.47), garag (0.47), realdonaldtrump (0.45), phone (0.42)	custodi 9 (-0.40), stori (-0.37), msnbc (-0.36), report fire (-0.34), sad (-0.31)
airport gun (0.47), free zone (0.42), gun free (0.42), free (0.38), zone (0.33)	mass (-0.50), #ftlauderdal (-0.19), peopl (-0.07), gun airport (-0.07), control (-0.06)
airport (0.23), shooter (-0.04)	shooter (-0.04), airport (0.23)
affect airport (0.41), shooter (0.34), make (0.33), involv ft (0.28), prayer affect (0.27)	gun (-0.42), violenc (-0.41), week (-0.39), stay (-0.37), travel (-0.37)
airport (-0.01)	airport (-0.01)
shot fire (0.42), fire (0.38), attack (0.33), terror (0.30), man (0.30)	#airport (-0.37), gunman (-0.32), week (-0.32), stop (-0.27), 2017 (-0.25)

Table 16: Most partisan phrases per topic for **Fort Lauderdale**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
islamist (0.18), fake (0.18), fake news (0.18), white male (0.18), facebook (0.18)	mass (-0.64), nation (-0.44), matter (-0.43), tweet (-0.38), support (-0.31)
2 (0.37), open (0.34), open fire (0.33), cnn (0.33), muslim (0.33)	4 (-0.52), offic (-0.44), dyer (-0.33), fatal (-0.26), spree (-0.24)
2 (0.46), 1 (0.46), california (0.37), kill (0.16)	kill (0.16), california (0.37), 1 (0.46), 2 (0.46)
california (0.32), stop (0.23), control (0.18), gun control (0.17), law (0.09)	mental (-0.47), shot (-0.38), shooter (-0.26), kill (-0.07), peopl (-0.02)
shooter shout (0.27), translat (0.27), #maga (0.27), fake (0.27), fake news (0.27)	offic (-0.68), fatal (-0.59), gun (-0.58), connect (-0.54), kori muhammad (-0.40)
attack (0.25), love (0.23), 3 (0.21), kill (0.21), prayer (0.10)	sad (-0.40), hate (-0.34), today (-0.23), peopl (-0.16), shoot (-0.11)
victim (0.17), kill (-0.03)	kill (-0.03), victim (0.17)
yesterday (0.31), terror (0.24), black man (0.23), kill 4 (0.21), target (0.21)	trump (-0.39), spree (-0.35), happen (-0.32), tweet (-0.30), make (-0.29)

Table 17: Most partisan phrases per topic for **Fresno**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
cover (0.53), workplac (0.39), trump (0.28), shooter (0.19), sf (0.16)	shoot (-0.40), blame (-0.25), terror (-0.25), today (-0.21), gun (-0.21)
4 injur (0.59), issu (0.58), compani (0.55), facil 4 (0.46), fox news (0.44)	barclay (-0.59), cover (-0.59), america (-0.59), 3 kill (-0.52), talk (-0.48)
4 (0.21), gunfir (0.19), kill 4 (0.19), dead (0.19), worker (0.17)	victim (-0.19), facil includ (-0.15), kill facil (-0.13), time (-0.10), facil (-0.08)
assault pistol (0.87), pistol (0.81), stolen (0.73), free (0.72), california (0.71)	congress (-0.47), secur (-0.43), die (-0.39), american (-0.39), gop (-0.38)
motiv (0.53), identifi (0.30), assault pistol (0.30), polic (0.28), warehous (0.28)	sfpd (-0.36), suspect (-0.32), #up (-0.32), lam (-0.31), jimmi (-0.31)
polic (0.54), find (0.41), colleagu (0.27), world (0.26), 2 (0.22)	countri (-0.33), gun violenc (-0.23), sf (-0.22), mass (-0.22), peopl (-0.21)
sf (-0.13), today (-0.14), victim (-0.15)	victim (-0.15), today (-0.14), sf (-0.13)
person (0.49), law (0.49), killer (0.39), grown (0.39), down (0.39)	america (-0.29), shot kill (-0.28), morn (-0.27), place (-0.27), murder (-0.26)

Table 18: Most partisan phrases per topic for **San Francisco**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
clinton blame (0.52), cover call (0.52), today cover (0.52), elitist (0.52), #ccot (0.52)	#whiteterror (-0.86), time talk (-0.86), patch (-0.86), #whiteterrorist (-0.86), patch pm (-0.86)
shooter long (0.52), long detail (0.52), video point (0.52), wsbtv report (0.52), seanhann (0.52)	global stay (-0.86), jule (-0.86), dhs jule (-0.86), jule rt (-0.86), #gunsens (-0.86)
#dtmag (0.52), #uscg #usarmi (0.52), #uscg (0.52), #usarmi #usairforc (0.52), #usairforc (0.52)	#lalat (-0.87), (-0.87), rt time (-0.87), #lasvegasmassacr stephen (-0.87), #lalat #breakingnew (-0.87)
gowdi (0.78), massacr arm (0.78), bash gun (0.78), gowdi difficult (0.78), paddock patsi (0.78)	#tytliv (-0.61), call rep (-0.61), money #guncontrolnow (-0.61), tweet thought (-0.61), obama era (-0.61)
fbi sourc (0.44), make statement (0.44), made lot (0.44), gun make (0.44), life made (0.44)	npr (-0.95), 95 (-0.95), click full (-0.95), white terrorist (-0.87), full stori (-0.86)
rt realdonaldtrump (0.71), #mandalaybay #nra (0.71), #bluelivesmatt (0.71), shooter reno (0.71), leftist (0.71)	trophi (-0.68), violenc text (-0.68), #notonemor (-0.68), mourn tomorrow (-0.68), tomorrow fight (-0.68)
injur stand (0.60), impeach postpon (0.60), green trump (0.60), #dtmag (0.60), stand countri (0.60)	receiv year (-0.79), click full (-0.79), raw stori (-0.79), #theresist (-0.79), #bakersfield (-0.79)
kain (0.67), tim kain (0.67), kain attack (0.67), room wish (0.67), kaya (0.67)	#tytliv (-0.72), #notmypresid (-0.72), senjohnthun (-0.72), prayer stop (-0.72), prayer nice (-0.72)

Table 19: Most partisan phrases per topic for **Vegas**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
news (0.60), motiv (0.42), blame (0.34), talk (0.32), tweet (0.31)	live stack (-0.29), furnitur (-0.29), bibl furnitur (-0.29), stack (-0.27), stack bibl (-0.27)
detail (0.34), involv (0.34), #coloradoshoot (0.34), park lot (0.34), #colorado #walmart (0.33)	talk (-0.67), #guncontrolnow (-0.58), control (-0.55), ago (-0.48), white (-0.46)
vega (-0.05)	vega (-0.05)
free (0.49), crime (0.48), #walmart (0.44), black (0.44), skill (0.33)	talk gun (-0.48), violenc (-0.48), place (-0.38), las (-0.37), las vega (-0.36)
updat (0.52), danger (0.44), kill peopl (0.43), #break (0.41), drive (0.40)	week (-0.50), delay (-0.50), white guy (-0.46), time (-0.45), guy (-0.44)
pray (0.39), time (0.36), night (0.30), peopl (0.21), love (0.18)	vega (-0.41), tonight (-0.32), dead (-0.22), hear (-0.18), heart (-0.17)
shot (0.21), kill (-0.25)	kill (-0.25), shot (0.21)
skill (0.78), chicago (0.75), lol (0.64), cover (0.61), friend (0.50)	open fire (-0.35), reason (-0.28), #guncontrol (-0.27), hispan (-0.25), man kill (-0.24)

Table 20: Most partisan phrases per topic for **Thornton**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
lunat (0.48), handler blame (0.48), countri music (0.48), sudanes (0.48), chelseahandl (0.48)	#thoughtsandpray (-0.91), 4 black (-0.91), black girl (-0.83), studi (-0.82), bibl studi (-0.82)
#antifa (0.62), #theresist (0.62), #nov4thitbegin (0.62), #uniteblu (0.62), #atheist (0.62)	#enough (-0.64), #gunsens (-0.63), mall (-0.63), #gun (-0.61), #nra (-0.60)
#usmarin #tricar (0.60), #tricar (0.60), #usnavi #usmarin (0.60), #usmarin (0.60), #usairforc #usnavi (0.60)	#guncontrolnow (-0.79), york time (-0.55), nightclub (-0.52), npr (-0.46), massacr texa (-0.41)
stop legal (0.74), #antifa (0.74), joe biden (0.74), shirt (0.74), democrat kill (0.74)	msnbc (-0.64), spineless (-0.64), earli talk (-0.64), prayer good (-0.64), prayer thought (-0.64)
fan (0.58), #antifa (0.58), #usairforc (0.58), gun stop (0.58), texa hero (0.58)	black girl (-0.81), klan (-0.81), bomber (-0.81), success (-0.81), birmingham church (-0.81)
frank (0.69), fbc (0.66), agenda (0.61), polit agenda (0.59), involv church (0.59)	relax (-0.69), gop (-0.69), governor church (-0.69), liter pray (-0.69), epidem (-0.69)
ted (0.54), lieu (0.54), ted lieu (0.54), song (0.54), app (0.54)	gun violenc (-0.68), thought (-0.62), #guncontrolnow (-0.61), great (-0.58), medic (-0.58)
tn church (0.75), biden man (0.75), church tn (0.75), hero stephen (0.75), septemb (0.75)	kill girl (-0.64), 4 black (-0.64), #impeachtrump (-0.64), #nraterror (-0.64), elementari school (-0.64)

Table 21: Most partisan phrases per topic for **Sutherland Springs**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
bush (0.76), student push (0.76), dirti (0.76), page (0.76), boston bomber (0.76)	attack survivor (-0.62), #trumprussia (-0.62), #russianbot (-0.62), #impeachtrump (-0.62), #votethemout (-0.62)
#emet news (0.65), #emet (0.65), news press (0.65), broadcast #emet (0.65), #new broadcast (0.65)	#enough (-0.74), result death (-0.74), fuck (-0.66), veloc (-0.66), high veloc (-0.63)
redondo (0.73), california high (0.73), expos (0.73), absolut (0.73), app (0.67)	make (-0.66), facebook post (-0.66), anonym (-0.66), washington post (-0.57), midterm (-0.56)
media politic (0.87), #greatawaken (0.87), #antifa (0.87), back window (0.87), #noguncontrol (0.87)	#throwthemout (-0.51), vote candid (-0.51), dollar nra (-0.51), #nrabloodmoney #nraisaterroristorgan (-0.51), concert goer (-0.51)
#emet news (0.54), #emet (0.54), news press (0.54), broadcast #emet (0.54), #new broadcast (0.54)	#maga hat (-0.84), wear maga (-0.76), grade (-0.73), cap (-0.73), hat (-0.72)
#qanon (0.89), join pray (0.79), #pray (0.70), pastor (0.68), heaven (0.68)	blah (-0.50), #gunreformnow #neveragain (-0.50), admiss (-0.50), blah blah (-0.50), #nraboycott (-0.50)
#emet news (0.82), #emet (0.82), news press (0.82), #new broadcast (0.82), broadcast #emet (0.82)	support #marchforourl (-0.56), #marchforourl #boycottnra (-0.56), refus accept (-0.56), #banassaultweapon (-0.56), #gunsensenow (-0.56)
#liberalismisamentaldisord (0.89), #firesheriffisrael (0.89), school 2015 (0.89), #gunfreezoneskil (0.89), #thegreatawaken (0.89)	#stopthenra (-0.49), prayer work (-0.49), #nomorethoughtsandpray (-0.49), #standwithparkland (-0.49), black brown (-0.49)

Table 22: Most partisan phrases per topic for **Parkland**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
left (0.76), liber (0.58), democrat (0.55), anti (0.44), blame (0.42)	school (-0.34), wait (-0.34), church (-0.34), hous attack (-0.34), 4 peopl (-0.34)
polic suspect (0.49), hold (0.48), east (0.45), fox news (0.43), pd (0.42)	abc news (-0.46), rt (-0.43), death (-0.43), tweet (-0.35), citi (-0.35)
gun (0.32), waffl hous (0.06), hous (0.06), waffl (0.05), worker (-0.07)	diploma (-0.43), mother (-0.33), cbs (-0.30), cbs news (-0.30), kill (-0.28)
democrat (0.56), agenda (0.56), gun seiz (0.54), gun confisc (0.52), free zone (0.50)	assault style (-0.73), 4 dead (-0.63), terrorist (-0.61), thought prayer (-0.60), gop (-0.60)
come (0.75), awar (0.63), antioch waffl (0.62), 6 (0.54), suspect 2 (0.49)	plastic (-0.64), mass murder (-0.57), domest (-0.56), #guncontrolnow (-0.54), murder 4 (-0.54)
student (0.41), colleg (0.37), opri mill (0.37), opri (0.37), mill (0.37)	condol victim (-0.51), massacr (-0.41), hous massacr (-0.41), rest (-0.38), hous tn (-0.37)
antioch waffl (0.69), visit (0.62), antioch (0.59), held (0.58), song (0.43)	hous white (-0.42), white shooter (-0.42), volum (-0.42), cnn (-0.41), hero white (-0.40)
robberi (0.77), goal (0.73), liber (0.72), pred (0.69), #2a (0.69)	sight (-0.40), theater (-0.40), justic (-0.40), movi theater (-0.40), effort (-0.40)

Table 23: Most partisan phrases per topic for **Nashville**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
leftist (0.91), narrat (0.81), #2a (0.78), left (0.70), control (0.67)	creat (-0.48), nut (-0.48), male (-0.42), immigr (-0.39), white male (-0.39)
school victim (0.54), femal (0.51), road (0.49), patient (0.49), fox news (0.42)	#gunreformnow (-0.65), #gunsens (-0.65), 22 school (-0.65), #resist (-0.65), #gunreform (-0.65)
video (0.40), famili (0.33), star (0.21), mass (0.21), student (0.18)	parkland survivor (-0.55), dead (-0.21), david (-0.21), hogg (-0.21), news (-0.20)
leftist (0.83), rack (0.83), gun rack (0.82), pickup (0.80), truck (0.79)	terrorist organ (-0.51), address gun (-0.51), prayer stop (-0.51), #ifidieinaschoolshoot (-0.51), 16th (-0.51)
#khou11 (0.74), nikola cruz (0.53), nikola (0.53), coach (0.52), watch (0.51)	minut (-0.65), shot kill (-0.55), nra (-0.54), loesch (-0.53), dana (-0.52)
secur (0.77), polic offic (0.67), brave (0.63), pray high (0.61), personnel (0.59)	#votethemout (-0.51), #nraisaterroristorgan (-0.51), action gun (-0.51), custodi (-0.51), fun (-0.51)
#txlege (0.62), kill 10 (0.57), pres (0.50), church (0.46), roundtabl (0.43)	#guncontrolnow (-0.70), show high (-0.70), wave (-0.70), school scene (-0.70), #enoughisenough (-0.70)
gun rack (0.89), dixon (0.86), rack (0.83), 38 revolv (0.83), truck (0.82)	entitl (-0.42), incel (-0.42), ritalin (-0.42), supremacist (-0.42), white supremacist (-0.42)

Table 24: Most partisan phrases per topic for **Santa Fe**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
reuter editor (0.85), apolog disciplin (0.78), disciplin blame (0.78), disciplin (0.78), gun control (0.76)	#guncontrolnow (-0.54), enemi american (-0.49), multipl (-0.45), call enemi (-0.45), domest (-0.44)
pray (0.57), alert (0.54), prayer victim (0.53), depart wisconsin (0.51), wisconsin (0.51)	milo (-0.60), gun journalist (-0.60), yiannopoulo (-0.60), milo yiannopoulo (-0.60), start gun (-0.60)
victim (0.51), editor (0.39), hiaasen (0.33), rob hiaasen (0.32), 5 (0.28)	kill (-0.15), newspap (-0.07), mass (-0.02), newsroom (-0.01), capit (0.11)
person (0.51), shotgun (0.49), crime (0.48), arm (0.47), mental health (0.47)	journal (-0.52), continu (-0.43), free press (-0.42), freedom (-0.40), #gunreformnow (-0.40)
hat (0.59), break (0.46), year jarrod (0.45), gazett kill (0.43), counti (0.39)	aliv (-0.55), accus shooter (-0.50), exclus (-0.47), dark link (-0.46), exclus accus (-0.46)
scene (0.54), blame (0.51), conscienc (0.47), address (0.47), rt (0.47)	#guncontrolnow (-0.44), congress (-0.44), #neveragain (-0.44), remind (-0.44), report capit (-0.44)
sander (0.59), game (0.59), speechless (0.56), opinion (0.53), build (0.49)	press enemi (-0.39), mind (-0.39), milo (-0.39), gazett mass (-0.39), yiannopoulo (-0.39)
blame trump (0.74), motiv shooter (0.63), bring (0.58), #fakenew (0.57), narrat (0.56)	start gun (-0.45), victim capit (-0.45), shot dead (-0.45), journalist gun (-0.38), nra (-0.38)

Table 25: Most partisan phrases per topic for **Annapolis**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
trump cnn (0.88), sarsour (0.88), cnn press (0.88), rabbi blame (0.84), camerota (0.81)	#magabomb (-0.51), black church (-0.51), spread (-0.51), long (-0.51), uncl (-0.47)
kill 6 (0.61), dozen (0.60), treat (0.59), synagogu pray (0.57), read (0.57)	arm (-0.51), white (-0.47), (-0.47), morn report (-0.45), respond saturday (-0.45)
polit insid (0.91), polit (0.91), defend (0.86), insid (0.79), survivor life (0.69)	site (-0.48), come (-0.48), stay (-0.47), tell (-0.47), offici (-0.47)
democrat (0.94), final (0.90), danger (0.77), free (0.76), understand (0.73)	rip (-0.35), blame victim (-0.35), avoid (-0.35), vote (-0.31), weak (-0.31)
pull (0.87), synagogu age (0.51), thursday (0.49), warn (0.49), expect (0.43)	includ 11 (-0.52), trail anti (-0.43), white man (-0.43), long (-0.42), review (-0.42)
target kill (0.88), horrif target (0.74), netanyahu (0.65), kill jewish (0.61), murder attack (0.57)	holocaust (-0.51), angri (-0.51), gun violenc (-0.48), rabbi synagogu (-0.48), synagogu call (-0.48)
michigan host (0.69), univers michigan (0.69), jihad day (0.69), palestinian jihad (0.69), advoc palestinian (0.69)	pictur (-0.70), add (-0.70), victim hate (-0.70), hebrew (-0.70), meltdown (-0.67)
wrong (0.45), wait (0.45), mind (0.41), kill 11 (0.41), lot (0.40)	insid (-0.40), templ (-0.40), liter (-0.33), black (-0.32), synagogu victim (-0.31)

Table 26: Most partisan phrases per topic for **Pittsburgh**. Brackets show the z-scores of the log odds of each phrase.

Republican	Democrat
leftist (0.84), anti gun (0.84), anti (0.77), left (0.74), liber (0.66)	violenc women (-0.55), republican (-0.49), women (-0.48), histori (-0.47), link (-0.47)
forc (0.65), insid california (0.57), want (0.57), california wildfir (0.53), camp (0.51)	mass 12 (-0.61), 12 victim (-0.61), gun violenc (-0.47), hour (-0.42), violenc (-0.42)
knelt block (0.84), wit men (0.84), knelt (0.84), real men (0.84), men knelt (0.84)	lane (-0.55), stay lane (-0.55), doctor stay (-0.55), nra told (-0.55), offer thought (-0.55)
divers (0.86), steal (0.86), 98 mass (0.86), communist divers (0.86), die communist (0.86)	#gunviol (-0.52), 307th (-0.52), #enoughisenough (-0.52), 307th mass (-0.52), marshablackburn (-0.52)
ball (0.77), earlier (0.77), commit gunman (0.77), gunman ian (0.77), clear (0.70)	women (-0.50), week (-0.49), men (-0.49), white men (-0.49), terrorist (-0.48)
men (0.88), involv california (0.86), strictest gun (0.75), gt gt (0.72), strictest (0.72)	#gunviol (-0.41), gun amp (-0.41), access gun (-0.41), mass place (-0.41), gunman kill (-0.41)
proclam (0.80), proclam honor (0.80), presidenti proclam (0.80), presidenti (0.80), tragedi california (0.63)	warm (-0.56), share (-0.56), #gunviol (-0.56), buck wear (-0.56), plan (-0.49)
california strictest (0.84), liber (0.84), strictest (0.81), left (0.81), california gun (0.81)	blackburn (-0.45), marsha (-0.45), mass die (-0.45), live gun (-0.45), gop (-0.45)

Table 27: Most partisan phrases per topic for **Thousand Oaks**. Brackets show the z-scores of the log odds of each phrase.

G Personal Pronouns

Pronoun usage has been found to express different degrees of self-reflection, awareness, inclusivity, perspective-taking, among several other psychological phenomena, both on Twitter (Qiu et al., 2012) and in various other domains (Hirsh and Peterson, 2009; Yarkoni, 2010; Pennebaker, 2011). We rely on these findings to treat pronouns as proxies to measure polarization in terms of users' personalization of the experience (1st person sg.), inclusion (1st person pl.), deflection (3rd person).

G.1 Methods

To quantify the partisanship of personal pronouns, we take the five personal pronoun categories in LIWC (Pennebaker et al., 2001) (I, You, We, SheHe, They) and then calculate their partisan log-odds ratio.

G.2 Results

The log odds of pronouns suggest that *first and second person pronouns* are significantly more likely to be used by Democrats across the events, while *They* is somewhat more likely to be used by Democrats and *SheHe* is used similarly by the two parties: *I* (mean:$-.26$, $p < .001$), *We* (mean:$-.26$, $p < .001$), *You* (mean:$-.13$, $p < 0.05$), *They* (mean:$-.06$, $p < .1$), *SheHe* (mean:$.05$, $p \approx 0.36$) (see Figure 15).

The use of two pronouns is significantly different based on the shooter's race: *SheHe* and *You* are both more likely to be used by Democrats when the shooter is white and by Republicans if the shooter is a person of color ($p < 0.01$ for *SheHe* and $p < 0.05$ for *You* from two-tailed t test). The pattern pertaining to *SheHe* might be partly explained by differential mentions of the president (see Appendix F with the most partisan words), since it so happens that most of the events where the shooter was a person of color occurred under Obama's presidency while the ones where the shooter was white predominantly occurred under Trump's presidency.

To better understand this result, we are again interested to see if there is a link between pronoun usage and topic preferences — we use the same procedure to measure the representation of pronouns in topics as in the case of modals. Our findings (see Figure 15) show that first *I* predominantly occurs in *solidarity* and *other*, which, coupled with previous findings about these topics being preferred by Democrats and about affect, suggest that Democrats in our dataset are more likely to personalize their tweets and write about their own mental state and feelings towards the event and the victims. Similarly *We* is overrepresented in *laws & policy*, also a topic that is preferred by Democrats, which, building on our results about modals (Section 7.2), provide evidence that Democrats are more likely to call for collective action.

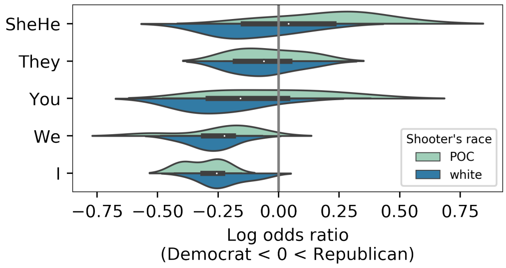


Figure 14: The partisan log odds ratio of pronouns.

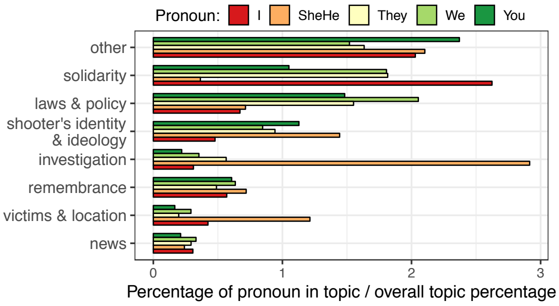


Figure 15: Distribution of pronouns across topics (mean across 21 events).

SheHe, on the other hand, are most frequent in *investigation*, *shooter's identity & ideology* and *victims & location* — topics that are more likely to be discussed by Republicans. This result, supported by our finding about affect, suggests that Republicans in this domain are more likely to author tweets that focus on a third person (e.g. the shooter).

H Modal Collocations

We study the subjects and complements of modals and their partisanship to get a better view of *how* these modals are used. We calculate the z-scored partisan log odds ratio of the collocations (subject + modal + complement constructions) at the event- and modal-level. We keep all collocations that are more likely to be used by Democrats or Republicans by at least .5 SD of the event- and modal-level log odds ratios (in other words, whose z-score has an absolute value that is at least .5). In the list below, we show those collocations that are partisan in the same direction (Democrat or Republican) in at least three events.

Note that before calculating the log odds ratio, we replace contracted forms with their non-contracted form (e.g. "shouldn't" with "should not", "should've" with "should have"). The number following the collocations is the number of events for which a particular term is partisan towards a given party. Often when a collocation seems ungrammatical, it is because it is comes from a question (e.g. "long must we" → "how long must we").

Note that the patterns in the collocations accord with the findings discussed in the paper. Democrats are more likely to use modals to call for collective and proactive action (e.g. "we must do", "we have to do", "something needs to be done", "we need to act") and also to express emotion (e.g. "[why do] people have to die", "[why do] people need to die") than Republicans. Republicans are more likely to use modals epistemically (e.g. "it must have been") and in other, idiomatic, senses that do not imply necessity (e.g. "it has to do [with]", "I have to say", "I must say"). Republicans are less likely to use modals in contexts with first person plural subjects ("we") than Democrats, but are more likely to use "we" in contexts where the modal's prejacent implies "stopping" something rather than "doing" something (e.g. "we must ban", "we must protect", "we should ban", "we need to protect").

MUST

Democrat

we must do (14), this must stop (8), we must end (7), something must be done (7), we must act (7), long must we (7), congress must act (7), we must stand (6), we must stop (6), he must be white (6), we must address (6), violence must end (6), violence must stop (6), lives must be lost (6), killing must stop (6), we must make (6), times must this (5), it must stop (5), we must all (5), people must die (5), many must die (5), we must keep (5), and must do (5), we must pass (5), we must continue (5), times must we (5), this must end (5), we must take (5), we must honor (5), shooter must be white (5), shooter must have been (4), insanity must stop (4), we must treat (4), we must remember (4), shootings must stop (4), is a must (4), shootings must end (4), change must happen (4), congress must take (4), this must change (4), we must get (4), action must be taken (4), control must happen (4), we must push (4), we must find (4), we must have gun (4), we must change (4), we must work (4), we must combat (4), we must deal (4), lives must be taken (4), we must try (4), congress must pass (4), madness must end (4), we must hold (4), #gunviolence must end (4), we must enact (4), rifles must be banned (4), they must know (3), i must say (3), we must ask (3), we must always (3), #guncontrol must happen (3), we must reclaim (3), madness must stop (3), it must take (3), there must have been (3), action must follow (3), we must condemn (3), i must ask (3), we must #stopgunviolence (3), innocents must die (3), nra must be so (3), why must we (3), we must come (3), we must fight (3), things must change (3), tragedies must end (3), violence must be stopped (3), we must confront (3), violence must be addressed (3), we must stay (3), you must be so (3), laws must change (3), we must (3), shootings must we (3), we must join (3), you must really (3), weapons must be banned (3), we must have change (3), love must prevail (3), shooting must have been (3), more must we (3), we must not become (3), we must create (3), we must allow (3), children must die (3), you must be proud (3), that must mean (3), we must fix (3), shooting must stop (3), we must look (3), must feel (3), we must as (3), more must die (3), there must be something (3), you must do (3), and must be stopped (3), we must talk (3), we must not forget (3), we must vote (3)

Republican

it must have been (6), they must be stopped (5), must watch (5), i must say (5), we must ban (5), they must know (4), you must know (4), we must protect (4), we must remain (4), i must have missed (4), he must resign (4), why must you (4), you must protect (4), we must be vigilant (4), attacks must stop (4), you must be happy (4), we must reform (4), it must be stopped (4), this must be one (4), rhetoric must stop (4), we must understand (4), we must stand (4), we must return (4)

SHOULD

Democrat

this should not be normal (8), we should all (7), who should not have guns (6), who should not have them (6), this should not be happening (6), this should not happen (6), who should not have had (6), this should not have happened (5), we should just (5), you should be ashamed (5), congress should be forced (5), that should not be happening (5), people should be able (5), we should never (5), who should not have a gun (5), civilians should not have access (5), we should know (5), you should see (5), never should have had (5), everyone should be able (4), we should ban (4), we should start (4), you should (4), we should probably (4), should we (4), you should not be able (4), we should bring (4), one should die (4), there should be some (4), we should not do (4), we should wait (4), this should never (4), we should not need (4), it should not be this (4), one should be able (4), we should not be afraid (4), everyone should have a gun (4), civilians should have access (4), they should have had (4), we should now (4), people should not have access (4), one should be afraid (4), he should not have had (4), this should not be our (4), we should try (4), one should live (4), we should not allow (4), hook should have been (4), people should have access (4), everyone should have guns (4), everyone should own (4), it should not take (4), this should have never (4), guns should be allowed (4), there should be a ban (3), media should be ashamed (3), we should change (3), people should be more (3), you should ask (3), we should politicize (3), you should never (3), i should have known (3), shootings should not be the norm (3), we should do (3), we should be upset (3), we should be afraid (3), there should have been (3), there should never (3), it should not be that (3), we should be focused (3), one should have access (3), we should not care (3), you should also (3), you should tweet (3), something should be done (3), should not they (3), it should have been (3), we should not be surprised (3), they should know (3), there should be more (3), why should it (3), you should call (3), congress should do (3), he should go (3), why should anyone (3), people should not be shot (3), guns should be outlawed (3), one should feel (3), weapons should be outlawed (3), rifles should be banned (3), guns should be illegal (3), people should not be killed (3), never should have happened (3), one should have to fear (3), people should feel (3), we should leave (3), what should be a safe (3), guns should be legal (3), people should be allowed (3), anyone should be able (3), media should stop (3), we should remember (3), but should not we (3), that should be illegal (3), they should feel (3), you should all (3), one should lose (3), you should too (3), civilian should have access (3), we should focus (3), we should (3), person should not be able (3), you should donate (3), should i (3), one should be getting (3), weapons should not be sold (3), everyone should have the right (3), guns should never (3), civilians should not have assault (3), people should not have to worry (3), you should not be allowed (3), shooting should not happen (3), we should stand (3), you should be embarrassed (3), we should build (3), you should leave (3), you should be more (3), we should not be able (3), what should you (3), you should not own (3), civilians should be able (3), guns should not be allowed (3), one should have to worry (3), we should have gun (3), he should never (3), everyone should read (3), we should not rush (3), it should not be allowed (3), we should say (3), all should (3), this should not be a political (3), he should be able (3), we should be having (3), people should not be able (3), i should not feel (3), they should put (3), man should not have been (3), civilian should ever (3), it should not be lost (3), civilians should not own (3), weapons should be legal (3), i should never (3), people should have died (3), fbi should investigate (3), we should attack (3), when should we (3), parent should ever (3), we should name (3), shooting should of (3), u should get (3), civilians should own (3), we should arm (3), it should be for (3), teachers should have guns (3), nra should have to pay (3), you should not have the right (3), shooting should have never (3), they should have been (3), we should really (3), it should not matter (3), who should get (3), that should not have guns (3), can should do (3), we should discuss (3), it should (3), one should be shot (3), should people (3), we should treat (3), we should not talk (3), they should (3), rifles should not be available (3), someone should tell (3), you should be shot (3), that should have been (3), we should not have laws (3), people should not be allowed (3), it should not even (3), or should we (3), civilian should own (3), they should call (3), you should be tweeting (3), one should have to go (3), that should be safe (3), that should never (3), we should definitely (3), child should ever (3), somebody should tell (3), nothing should be done (3), we should not let (3)

Republican

we should ban (8), we should make (7), and should be prosecuted (6), they should just (6), police should have guns (5), i should have been (5), you should know (5), they should of (5), you should go (5), he should have been (5), we should keep (5), he should be arrested (5), they should not be allowed (5), government should have guns (5), we should see (5), should we (5), they should give (4), they should be allowed (4), everyone should know (4), we should at (4), they should make (4), shooting should not we (4), obama should resign (4), someone should inform (4), we should outlaw (4), you should do (4), you should ask (4), he should be fired (4), us should take (4), you should be fired (4), what should be done (4), fbi should have been (4), they should (4), why should he (4), and should have been (4), we should be asking (4), we should have guns (4), parents should be held (4), he should have stopped (4), we should add (4), he should take (4), and should be fired (4), they should use (4), democrats should not be allowed (4), they should hang (4), you should be mad (4), we should work (4), what should they (4), they should take (4), we should be able (4), guns should be taken (4), you should stop (4), you should see (4), there should never (4), shooter should get (4), who should we (4), shooter should not have had (4)

NEED TO

Democrat

we need to take (8), more need to die (8), we need to act (8), we need to talk (8), we need to do (8), we need to stand (7), we need to stop (7), people need to die (7), shootings need to happen (6), laws need to change (6), we need to change (6), we need to fix (6), we need to vote (6), we need to make (6), we need to call (5), they need to carry (5), we need to remember (5), don't need to see (5), all need to take (5), we need to hear (4), you need to rethink (4), we need to be better (4), we need to start (4), guns need to go (4), we need to figure (4), people need to stop (4), you need to change (4), we need to keep (4), we need to solve (4), laws need to be changed (4), we need to work (4), we need to end (4), we need to be able (4), we need to address (4), we need to help (4), we need to demand (4), weapons need to be banned (4), we need to have more (4), i need to know (4), lives need to be lost (3), acts need to stop (3), you need to know (3), we need to focus (3), don't need to keep (3), many need to die (3), we need to have before (3), we need to look (3), we need to say (3), we need to see (3), we need to pass (3), massacres need to happen (3), you need to recognize (3), also need to make (3), this need to happen (3), lives need to be taken (3), we need to #endgunviolence (3), all need to stand (3), we need to acknowledge (3), we need to mourn (3), shootings need to stop (3), don't need to know (3), violence need to end (3), we need to have better (3), we need to live (3), laws need to happen (3), violence need to stop (3), they need to know (3), people need to be shot (3), we need to wait (3), you need to talk (3), we need to listen (3), we need to honor (3), people need to stand (3), i need to stop (3), we need to be talking (3), we need to also (3), we need to speak (3), who need to get (3), we need to go (3), people need to hear (3), we need to wake (3), guns need to be restricted (3), really need to get (3), we need to have a real (3), felt the need to take (3), we need to understand (3), all need to see (3), we need to be outraged (3), people need to start (3), i need to add (3), they need to pay (3), don't need to do (3), we need to ban (3), leaders need to do (3), you need to tell (3), we need to limit (3), we need to prevent (3), we need to really (3)

Republican

you need to know (8), killings need to stop (6), we need to protect (6), we need to get (6), we need to focus (6), we need to know (6), we need to be more (5), really need to look (5), people need to realize (5), people need to stop (5), people need to know (5), we need to pray (4), we need to wait (4), we need to allow (4), they need to kill (4), americans need to be armed (4), really need to stop (4), we need to return (4), we need to kill (4), you need to watch (4), seriously need to change (4), heads need to roll (4), you need to investigate (4), democrats need to stop (4), people need to get (4), you need to address (4), you need to go (4), we need to enforce (4), we need to find (4), guns need to be banned (4), we need to look (4), you need to look (4), you need to ask (4), we need to punish (4)

NEEDS TO

Democrat

something needs to be done (8), this needs to stop (8), violence needs to stop (7), it needs to end (6), something needs to change (5), there needs to be stricter (5), this needs to end (5), violence needs to end (5), gop needs to stop (5), america needs to do (4), change needs to happen (4), that needs to be addressed (4), shit needs to stop (4), congress needs to stop (4), it needs to stop (4), america needs to end (3), else needs to happen (3), seriously needs to be done (3), control needs to happen (3), one needs to own (3), what needs to happen (3), reform needs to happen (3), control needs to be a thing (3), there needs to be more (3), there needs to be gun (3), hatred needs to stop (3), it needs to be said (3), world needs to change (3), work needs to be done (3), law needs to change (3), seriously needs to stop (3), control needs to be addressed (3), cnn needs to stop (3), congress needs to do (3), someone needs to tell (3), story needs to be told (3), he needs to take (3), it needs to happen (3), he needs to get (3), congress needs to act (3), this needs to be stopped (3), trump needs to stop (3)

Republican

she needs to go (4), someone needs to ask (4)

HAVE TO

Democrat

people have to die (10), we have to do (7), more have to die (6), many have to die (6), don’t have to live (5), should not have to fear (4), we have to say (4), don’t have to do (4), this have to happen (4), you have to say (4), things have to change (4), should have to fear (4), we have to change (4), we have to talk (4), we have to act (4), we have to keep (4), i have to worry (4), we have to read (3), we have to go (3), we have to have before (3), we have to endure (3), we have to lose (3), lives have to be lost (3), we have to accept (3), we have to hear (3), and not have to worry (3), you have to offer (3), we have to end (3), i have to see (3), shootings have to stop (3), we have to look (3), should have to go (3), we have to pay (3), children have to die (3)

Republican

i have to wonder (7), that have to do (6), shooting have to do (6), you have to say (6), we have to ask (5), i have to say (5), will have to live (5), shootings have to stop (5), i have to go (5), you have to kill (5), i have to agree (5), you have to change (5), you have to be to shoot (5), does have to do (5), nra have to do (5), trump have to do (5), these have to happen (4), we have to listen (4), you have to go (4), you have to put (4), will have to wait (4), going to have to work (4), you have to talk (4), we have to pray (4), we have to see (4), you have to know (4), you have to stop (4), this have to do (4), people have to stop (4), you have to give (4), going to have to get (4), who have to deal (4), you have to ask (4), we have to remember (4), you have to be really (4)

HAS TO

Democrat

this has to stop (11), something has to change (6), it has to stop (5), violence has to end (5), shooting has to do (4), this has to end (4), something has to be done (4), more has to happen (4), killing has to stop (4), it has to end (4), what the has to say (4), shit has to stop (3), what has to happen (3), this has to change (3), madness has to end (3), simply has to stop (3), humanity has to offer (3), what has to be done (3), stuff has to stop (3), else has to happen (3), there has to be footage (3), killings has to stop (3)

Republican

it has to do (8), that has to do (6), obama has to say (5), he has to say (5), this has to end (4), there has to be some (4), what has to say (4), one has to ask (4), shit has to end (4)

I Results: Additional Plots

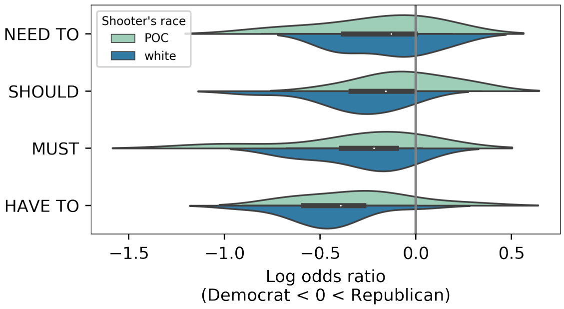


Figure 16: The log odds ratio of necessity modals.

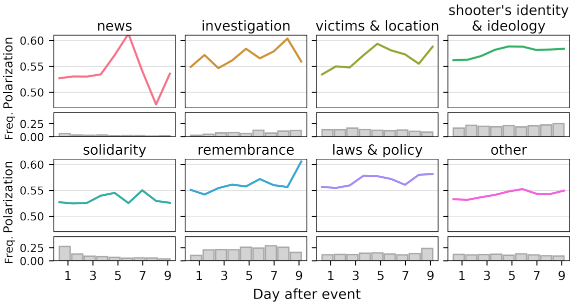


Figure 17: Topic polarization of Orlando over time, as measured by the leave-out estimate of phrase partisanship. The bar charts show the proportion of each topic in the data at a given time.

Learning to Decipher Hate Symbols

Jing Qian, Mai ElSherief, Elizabeth Belding, William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`{jing_qian,mayelsherief,ebelding,william}@cs.ucsb.edu`

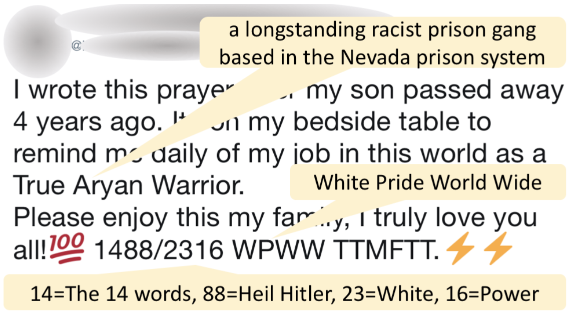
Abstract

Existing computational models to understand hate speech typically frame the problem as a simple classification task, bypassing the understanding of hate symbols (e.g., *14 words*, *kigy*) and their secret connotations. In this paper, we propose a novel task of deciphering hate symbols. To do this, we leverage the Urban Dictionary and collected a new, symbol-rich Twitter corpus of hate speech. We investigate neural network latent context models for deciphering hate symbols. More specifically, we study Sequence-to-Sequence models and show how they are able to crack the ciphers based on context. Furthermore, we propose a novel Variational Decipher and show how it can generalize better to unseen hate symbols in a more challenging testing setting.

1 Introduction

The statistics are sobering. The Federal Bureau of Investigation of United States[1] reported over 6,000 criminal incidents motivated by bias against race, ethnicity, ancestry, religion, sexual orientation, disability, gender, and gender identity during 2016. The most recent 2016 report shows an alarming 4.6% increase, compared with 2015 data[2]. In addition to these reported cases, thousands of Internet users, including celebrities, are forced out of social media due to abuse, hate speech, cyberbullying, and online threats. While such social media data is abundantly available, the broad question we are asking is—What can machine learning and natural language processing do to help and prevent online hate speech?

The vast quantity of hate speech on social media can be analyzed to study online abuse. In recent years, there has been a growing trend of developing computational models of hate speech. However, the majority of the prior studies focus solely on modeling hate speech as a binary or multiclass classification task (Djuric et al., 2015; Waseem and Hovy, 2016; Burnap and Williams, 2016; Wulczyn et al., 2017; Pavlopoulos et al., 2017).



a longstanding racist prison gang based in the Nevada prison system

I wrote this praye my son passed away 4 years ago. I on my bedside table to remind m daily of my job in this world as a True Aryan Warrior.

White Pride World Wide

Please enjoy this my fam y, I truly love you all!💯 1488/2316 WPWW TTMFTT.⚡⚡

14=The 14 words, 88=Heil Hitler, 23=White, 16=Power

Figure 1: An example tweet with hate symbols.

While developing new features for hate speech detection certainly has merits, we believe that understanding hate speech requires us to design computational models that can decipher hate symbols that are commonly used by hate groups. Figure 1 shows an example usage of hate symbols in an otherwise seemingly harmless tweet that promotes hate. For example, *Aryan Warrior* is a longstanding racist prison gang based in the Nevada prison system. *WPWW* is the acronym for *White Pride World Wide*. The hate symbols *1488* and *2316* are more implicit. *14* symbolizes the 14 words: *"WE MUST SECURE THE EXISTENCE OF OUR PEOPLE AND A FUTURE FOR WHITE CHILDREN"*, spoken by members of the Order neo-Nazi movement. *H* is the 8th letter of the alphabet, so *88=HH=Heil Hitler*. Similarly, *W* is the 23rd and *P* is the 16th letter of the alphabet, so *2316=WP=White Power*.

---

[1] https://www.fbi.gov/news/stories/2016-hate-crime-statistics

[2] https://www.fbi.gov/news/stories/2015-hate-crime-statistics-released

Proceedings of NAACL-HLT 2019, pages 3006–3015
Minneapolis, Minnesota, June 2 - June 7, 2019.

In this work, we propose the first models for deciphering hate symbols. We investigate two families of neural network approaches: the Sequence-to-Sequence models (Sutskever et al., 2014; Cho et al., 2014) and a novel Variational Decipher based on the Conditional Variational Autoencoders (Kingma and Welling, 2014; Sohn et al., 2015; Larsen et al., 2016). We show how these neural network models are able to guess the meaning of hate symbols based on context embeddings and even generalize to unseen hate symbols during testing. Our contributions are three-fold:

- We propose a novel task of learning to decipher hate symbols, which moves beyond the standard formulation of hate speech classification settings.
- We introduce a new, symbol-rich tweet dataset for developing computational models of hate speech analysis, leveraging the Urban Dictionary.
- We investigate a sequence-to-sequence neural network model and show how it is able to encode context and crack the hate symbols. We also introduce a novel Variational Decipher, which generalizes better in a more challenging setting.

In the next section, we outline related work in text normalization, machine translation, conditional variational autoencoders, and hate speech analysis. In Section 3, we introduce our new dataset for deciphering hate speech. Next, in Section 4, we describe the design of two neural network models for the decipherment problem. Quantitative and qualitative experimental results are presented in Section 5. Finally, we conclude in Section 6.

2 Related Work

2.1 Text Normalization in Social Media

The proposed task is related to text normalization focusing on the problems presented by user-generated content in online sources, such as misspelling, rapidly changing out-of-vocabulary slang, short-forms and acronyms, punctuation errors or omissions, etc. These problems usually appear as out-of-vocabulary words. Extensive research has focused on this task (Beaufort et al., 2010; Liu et al., 2011; Gouws et al., 2011; Han and Baldwin, 2011; Han et al., 2012; Liu et al., 2012; Chrupała, 2014). However, our task is different from the general text normalization in social media in that instead of the out-of-vocabulary words, we focus on the symbols conveying hateful meaning. These hate symbols can go beyond lexical variants of the vocabulary and thus are more challenging to understand.

2.2 Machine Translation

An extensive body of work has been dedicated to machine translation. Knight et al. (2006) study a number of natural language decipherment problems using unsupervised learning. Ravi and Knight (2011) further frame the task of machine translation as decipherment and tackle it without parallel training data. Machine translation using deep learning (Neural Machine Translation) has been proposed in recent years. Sutskever et al. (2014) and Cho et al. (2014) use Sequence to Sequence (Seq2Seq) learning with Recurrent Neural Networks (RNN). Bahdanau et al. (2015) further improve translation performance using the attention mechanism. Google's Neural Machine Translation System (GNMT) employs a deep attentional LSTM network with residual connections (Wu et al., 2016). Recently, machine translation techniques have been also applied to explain non-standard English expressions (Ni and Wang, 2017). However, our deciphering task is not the same as machine translation in that hate symbols are short and cannot be modeled as language.

Our task is more closely related to (Hill et al., 2016) and (Noraset et al., 2017). Hill et al. (2016) propose using neural language embedding models to map the dictionary definitions to the word representations, which is the inverse of our task. Noraset et al. (2017) propose the definition modeling task. However, in their task, for each word to be defined, its pre-trained word embedding is required as an input, which is actually the prior knowledge of the words. However, such kind of prior knowledge is not available in our decipherment task. Therefore, our task is more challenging and is not simply a definition modeling task.

2.3 Conditional Variational Autoencoder

Unlike the original Seq2Seq model that directly encodes the input into a latent space, the Variational Autoencoder (VAE) (Kingma and Welling, 2014) approximates the underlying probability distribution of data. VAE has shown promise in multiple generation tasks, such as handwritten digits (Kingma and Welling, 2014; Salimans et al., 2015), faces (Kingma and Welling, 2014; Rezende et al., 2014), and machine translation (Zhang et al., 2016). Conditional Variational Autoencoder (Larsen et al., 2016; Sohn et al., 2015) extends the original VAE framework by incorporating conditions during generation. In addition to image generation, CVAE has been successfully applied to some NLP tasks. For example, Zhao et al. (2017) apply CVAE to dialog generation, while Guu et al. (2018) use CVAE for sentence generation.

2.4 Hate Speech Analysis

Closely related to our work are Pavlopoulos et al. (2017); Gao et al. (2017). Pavlopoulos et al. (2017) build an RNN supplemented by an attention mechanism that outperforms the previous state of the art system in user comment moderation (Wulczyn et al., 2017). Gao et al. (2017) propose a weakly-supervised approach that jointly trains a slur learner and a hate speech classifier. While their work contributes to the automation of harmful content detection and the highlighting of suspicious words, our work builds upon these contributions by providing a learning mechanism that deciphers suspicious hate symbols used by communities of hate to bypass automated content moderation systems.

3 Dataset

In this section, we describe the dataset we collected for hate symbol decipherment.

3.1 Hate Symbols

We first collect hate symbols and the corresponding definitions from the Urban Dictionary. Each term with one of the following hashtags: *#hate, #racism, #racist, #sexism, #sexist, #nazi* is selected as a candidate and added to the set S_0. We collected a total of 1,590 terms. Next, we expand this set by different surface forms using the Urban Dictionary API. For each term s_i in set S_0, we obtain a set of terms R_i that have the same meaning as s_i but with different surface forms. For example, for the term *brown shirt*, there are four terms with different surface forms: *brown shirt, brown shirts, Brownshirts, brownshirt*. Each term in R_i has its own definition in Urban Dictionary, but since these terms have exactly the same meaning, we select a definition d_i with maximum upvote/downvote ratio for all the terms in R_i. For example, for each term in the set R_i={*brown shirt, brown shirts, Brownshirts, brownshirt*}, the corresponding definition is *"Soldiers in Hitler's storm trooper army, SA during the Nazi regime..."* After expanding, we obtain 2,105 distinct hate symbol terms and their corresponding definitions. On average, each symbol consists of 9.9 characters, 1.5 words. Each definition consists of 96.8 characters, 17.0 words.

3.2 Tweet Collection

For each of the hate symbols, we collect all tweets from 2011-01-01 to 2017-12-31 that contain exactly the same surface form of hate symbol in the text. Since we only focus on hate speech, we train an SVM (Cortes and Vapnik, 1995) classifier to filter the collected tweets. The SVM model is trained on the dataset published by Waseem and Hovy (2016). Their original dataset contains three labels: Sexism, Racism, and None. Since the SVM model is used to filter the non-hate speech, we merge the instances labeled as sexism and racism, then train the SVM model to do binary classification. After filtering out all the tweets classified as non-hate, our final dataset consists of 18,667 (tweet, hate symbol, definition) tuples.

4 Our Approach

We formulate hate symbol deciphering as the following equation:

$$Obj = \sum_{(u,s,d^*)\in X} \log p(d^*|(u,s)) \qquad (1)$$

X is the dataset, (u, s, d^*) is the (tweet, symbol, definition) tuple in the dataset. The inputs are the tweet and the hate symbol in this tweet. The output is the definition of the symbol. Our objective is to maximize the probability of the definition given the (tweet, symbol) pair. This objective function is very similar to that of machine translation. So we first try to tackle it based on the Sequence-to-Sequence model, which is commonly used in machine translation.

4.1 Sequence-to-Sequence Model

We implement an RNN Encoder-Decoder model with attention mechanism based on Bahdanau et al. (2015). We use GRU (Cho et al., 2014) for decoding. However, instead of also using GRU for encoding, we found that LSTM (Hochreiter

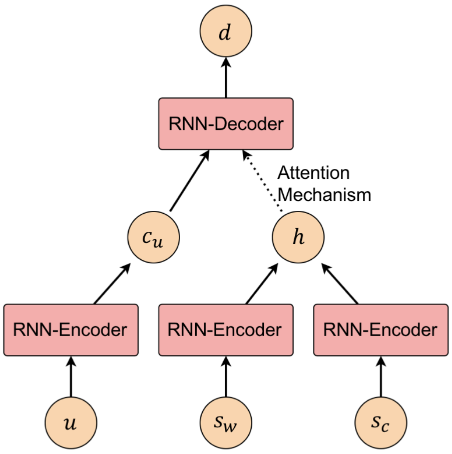


Figure 2: Our Seq2Seq model. u, s_w are the word embeddings of the tweet text and hate symbol. s_c is the character embedding of the symbol. c_u is the encoded tweet and h is the concatenated hidden states. d is the generated text. Detailed explanation is in section 4.1.

and Schmidhuber, 1997) performs better on our task. Therefore, our Seq2Seq model uses LSTM encoders and GRU decoders. An overview of our Seq2Seq model is shown in Figure 2. The computation process is shown as the following equations:

$$c_u, h_u = f_u(u) \tag{2}$$

$$c_{sw}, h_{sw} = f_{sw}(s_w) \tag{3}$$

$$c_{sc}, h_{sc} = f_{sc}(s_c) \tag{4}$$

u is the word embedding of the tweet text, s_w is the word embedding of the hate symbol, s_c is the character embedding of the symbol. f_u, f_{sw}, and f_{sc} are LSTM functions. c_u, c_{sw}, c_{sc} are the outputs of the LSTMs at the last time step and h_u, h_{sw}, h_{sc} are the hidden states of the LSTMs at all time steps. We use two RNN encoders to encode the symbol, one encodes at the word level and the other one encodes at the character level. The character-level encoded hate symbol is used to provide the feature of the surface form of the hate symbol while the word-level encoded hate symbol is used to provide the semantic information of the hate symbol. The hidden states of the two RNN encoders for hate symbols are concatenated:

$$h = h_{sw} \oplus h_{sc} \tag{5}$$

c_u is the vector of encoded tweet text. The tweet text is the context of the hate symbol, which provides additional information during decoding. Therefore, the encoded tweet text it is also fed into the RNN decoder. The detailed attention mechanism and decoding process at time step t are as follows:

$$w_t = \sigma(l_w(d_{t-1} \oplus e_{t-1})) \tag{6}$$

$$a_t = \sum_{i=1}^{T} w_{ti} h_i \tag{7}$$

$$b_t = \sigma(l_c(d_{t-1} \oplus a_t)) \tag{8}$$

$$o_t, e_t = k(c_u \oplus b_t, e_{t-1}) \tag{9}$$

$$p(d_t|u, s) = \sigma(l_o(o_t)) \tag{10}$$

w_t is the attention weights at time step t and w_{ti} is the i_{th} weight of w_t. d_{t-1} is the generated word at last time step and e_{t-1} is the hidden state of the decoder at last time step. h_i is the i_{th} time step segment of h. l_w, l_c, and l_o are linear functions. σ is a nonlinear activation function. k is the GRU function. o_t is the output and e_t is the hidden state of the GRU. $p(d_t|u, s)$ is the probability distribution of the vocabulary at time step t. The attention weights w_t are computed based on the decoder's hidden state and the generated word at time step $t-1$. Then the computed weights are applied to the concatenated hidden states h of encoders. The result a_t is the context vector for the decoder at time step t. The context vector and the last generated word are combined by a linear function l_c followed by a nonlinear activation function. The result b_t is concatenated with the encoded tweet context c_u, and then fed into GRU together with the decoder's last hidden state e_{t-1}. Finally, the probability of each vocabulary word is computed from o_t.

4.2 Variational Decipher

The Variational Decipher is based on the CVAE model, which is another model that can be used to parametrize the conditional probability $p(d^*|(u, s))$ in the objective function (Equation 1). Unlike the Seq2Seq model, which directly parametrizes $p(d^*|(u, s))$, our variational decipher formulates the task as follows:

$$\begin{aligned} Obj &= \sum_{(u,s,d^*)\in X} \log p(d^*|(u, s)) \\ &= \sum_{(u,s,d^*)\in X} \log \int_z p(d^*|z) p(z|(u, s)) dz \end{aligned} \tag{11}$$

where z is the latent variable. $p(d^*|(u, s)$ is written as the marginalization of the product of two terms over the latent space. Since the integration

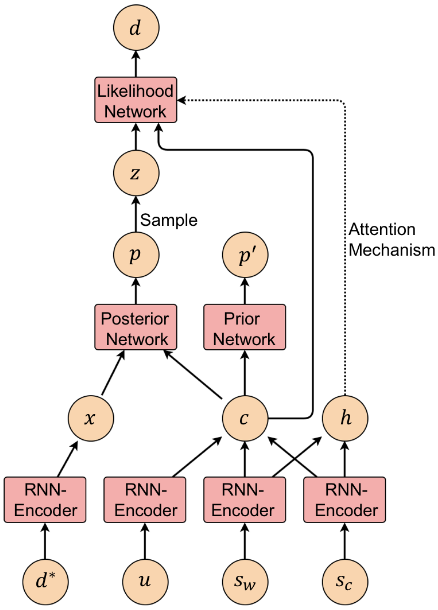


Figure 3: The Variational Decipher. Note that this structure is used during training. During testing, the structure is slightly different. d^* is the word embeddings of the definition. x is the encoded definition. c is the concatenation of the encoded tweet and hate symbol. p and p' are output distributions. z is the latent variable. The definitions of other variables are the same as those in Figure 2. Detailed explanation is in section 4.2.

over z is intractable, we instead try to maximize the evidence lower bound (ELBO). Our variational lower bound objective is in the following form:

$$\begin{aligned} Obj = & E[\log p_\varphi(d^*|z,u,s)] - \\ & D_{KL}[p_\alpha(z|d^*,u,s)||p_\beta(z|u,s)] \end{aligned} \tag{12}$$

where $p_\varphi(d^*|z,u,s)$ is the likelihood, $p_\alpha(z|d^*,u,s)$ is the posterior, $p_\beta(z|u,s)$ is the prior, and D_{KL} is the Kullback-Leibler (KL) divergence. We use three neural networks to model these three probability distributions. An overview of our variational decipher is shown in Figure 3. We first use four recurrent neural networks to encode the (tweet, symbol, definition) pair in the dataset. Similar to what we do in the Seq2Seq model, there are two encoders for the hate symbol. One is at the word level and the other is at the character level. The encoding of symbols and tweets are exactly the same as in our Seq2Seq model (see Equations 2-4). The difference is that we also need to encode definitions for the Variational Decipher.

$$x, h_d = f_d(d^*) \tag{13}$$

Here, f_d is the LSTM function. x is the output of the LSTM at the last time step and h_d is the hidden state of the LSTM at all time steps. The condition vector c is the concatenation of the encoded symbol words, symbol characters, and the tweet text:

$$c = c_u \oplus c_{sw} \oplus c_{sc} \tag{14}$$

We use multi-layer perceptron (MLP) to model the posterior and the prior in the objective function. The posterior network and the prior network have the same structure and both output a probability distribution of latent variable z. The only difference is that the input of the posterior network is the concatenation of the encoded definition x and the condition vector c while the input of the prior network is only the condition vector c. Therefore, the output of the posterior network $p = p_\alpha(z|d^*,u,s)$ and the output of the prior network $p' = p_\beta(z|u,s)$. By assuming the latent variable z has a multivariate Gaussian distribution, the actual outputs of the posterior and prior networks are the mean and variance: (μ, Σ) for the posterior network and (μ', Σ') for the prior network.

$$\mu, \Sigma = g(x \oplus c) \tag{15}$$

$$\mu', \Sigma' = g'(c) \tag{16}$$

g is the MLP function of the posterior network and g' is that of the prior network. During training, the latent variable z is randomly sampled from the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ and fed into the likelihood network. During testing, the posterior network is replaced by the prior network, so z is sampled from $\mathcal{N}(\mu', \Sigma')$. The likelihood network is modeled by an RNN decoder with attention mechanism, very similar to the decoder of our Seq2Seq model. The only difference lies in the input for the GRU. The decoder in our Variational Decipher model is to model the likelihood $p_\varphi(d^*|z,u,s)$, which is conditioned on the latent variable, tweet context, and the symbol. Therefore, for the Variational Decipher, the condition vector c and the sampled latent variable z are fed into the decoder.

$$o_t, e_t = k(z \oplus c \oplus b_t, e_{t-1}) \tag{17}$$

e_{t-1} is the hidden state of the RNN decoder at the last time step. k is the GRU function. o_t is its output and e_t is its hidden state. Detailed decoding process and explanations are in section 4.1.

According to the objective function in Equation 12, the loss function of the Variational Decipher is as follows:

$$\begin{aligned}\mathcal{L} =& L_{REC} + L_{KL} \\ =& E_{z\sim p_\alpha(z|d^*,u,s)}[-\log p_\varphi(d^*|z,u,s)]+ \\ & D_{KL}[p_\alpha(z|d^*,u,s)||p_\beta(z|u,s)]\end{aligned} \quad (18)$$

It consists of two parts. The first part L_{REC} is called reconstruction loss. Optimizing L_{REC} can push the sentences generated by the posterior network and the likelihood network closer to the given definitions. The second part L_{KL} is the KL divergence loss. Optimizing this loss can push the output Gaussian Distributions of the prior network closer to that of the posterior network. This means we teach the prior network to learn the same knowledge learned by the posterior network, such that during testing time, when the referential definition d^* is no longer available for generating the latent variable z, the prior network can still output a reasonable probability distribution over the latent variable z. The complete training and testing process for the Variational Decipher is shown in Algorithm 1. M is the predefined maximum length of the generated text. BCE refers to the Binary Cross Entropy loss.

5 Experiments

5.1 Experimental Settings

We use the dataset collected as described in section 3 for training and testing. We randomly selected 2,440 tuples for testing and use the remaining 16,227 tuples for training. Note that there are no overlapping hate symbols between the training dataset U and the testing dataset D.

We split the 2,440 tuples of the testing dataset D into two separate parts, D_s and D_d. D_s consists of 1,681 examples and D_d consists of 759 examples. In the first testing dataset D_s, although each hate symbol does not appear in the training dataset, the corresponding definition appears in the training dataset. In the second testing dataset D_d, neither the hate symbols nor the corresponding definitions appear in the training dataset. We do this split because deciphering hate symbols in these two cases has different levels of difficulty.

Algorithm 1 Train & Test Variational Decipher

```
function TRAIN($U$)
    randomly initialize network parameters $\varphi$, $\alpha$, $\beta$;
    for $epoch = 1, E$ do
        for $(tweet, symbol, definition)$ in $U$ do
            get embeddings $u$, $s_w$, $s_c$, $d^*$;
            compute $x$, $c$ and $h$ with RNN encoders;
            compute $\mu$, $\Sigma$ with the posterior network;
            compute $\mu'$, $\Sigma'$ with the prior network;
            compute KL-divergence loss $L_{KL}$;
            sample $z = reparameterize(\mu, \Sigma)$;
            initialize the decoder state $e_0 = c$;
            $L_{REC} = 0$;
            for $t = 1, M$ do
                compute attention weights $w_t$;
                compute $o_t$, $e_t$ and $p(d_t|z, u, s)$;
                $d_t = indmax(p(d_t|z, u, s))$;
                $L_{REC} += BCE(d_t, d^*_t)$;
                if $d_t$==EOS then
                    break;
                end if
            end for
            update $\varphi$, $\alpha$, $\beta$ on $\mathcal{L} = L_{REC} + L_{KL}$;
        end for
    end for
end function

function TEST($V$)
    for $(tweet, symbol, definition)$ in $V$ do
        get embeddings $u$, $s_w$, $s_c$;
        compute $c$ and $h$ with RNN encoders;
        compute $\mu'$, $\Sigma'$ with the prior network;
        sample $z = reparameterize(\mu', \Sigma')$;
        initialize the decoder state $e_0 = c$;
        for $t = 1, M$ do
            compute attention weights $w$;
            compute $o_t$, $e_t$ and $p(d_t|z, u, s)$;
            $d_t = indmax(p(d_t|z, u, s))$;
            if $d_t$==EOS then
                break;
            end if
        end for
    end for
end function
```

This split criterion means that for each hate symbol in D_s, there exists some symbol in the training dataset that has the same meaning but in different surface forms. For example, the hate symbol *wigwog* and *Wig Wog* have the same definition but one is in the training dataset, the other is in the first testing dataset. We assume that such types of hate symbols share similar surface forms or similar tweet contexts. Therefore, the first testing dataset D_s is to evaluate how well the model captures the semantic similarities among the tweet contexts in different examples or the similarities among different surface forms of a hate symbol.

Deciphering the hate symbols in the second testing dataset D_d is more challenging. Both the unseen hate symbols and definitions require the model to have the ability to accurately capture the semantic information in the tweet context and then make a reasonable prediction. The second testing dataset D_d is used to evaluate how well the model generalizes to completely new hate symbols.

Dataset	Method	BLEU	ROUGE-L	METEOR
D_s	Seq2Seq	**37.80**	**41.05**	**36.67**
	VD	34.77	32.96	31.03
D_d	Seq2Seq	25.44	12.96	**5.54**
	VD	**28.38**	**14.01**	5.41
D	Seq2Seq	**33.96**	**32.32**	**26.98**
	VD	32.75	27.00	23.16

Table 1: The BLEU, ROUGE-L and METEOR scores on testing datasets. VD refers to the Variational Decipher. D is the entire testing dataset. D_s is the first part of D and D_d is the second part. The better results are in bold.

For the Seq2Seq model, we use negative log-likelihood loss for training. Both models are optimized using Adam optimizer (Kingma and Ba, 2015). The hyper-parameters of two models are exactly the same. We set the maximum generation length $M = 50$. The hidden size of the encoders is 64. The size of the word embedding is 200 and that of character embedding is 100. The word embeddings and character embeddings are randomly initialized. Each model is trained for 50 epochs. We report the deciphering results of two models on three testing datasets D, D_s and D_d.

5.2 Experimental Results

Quantitative Results: We use equally weighted BLEU score for up to 4-grams (Papineni et al., 2002), ROUGE-L (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) to evaluate the decipherment results. The results are shown in Table 1. Figure 4 shows the BLEU score achieved by the two models on three testing datasets D, D_s and D_d during the training process. Both our Seq2Seq model and Variational Decipher achieve reasonable BLEU scores on the testing datasets. The Seq2Seq model outperforms the Variational Decipher on D_s while Variational Decipher outperforms Seq2Seq on D_d. Note that D_s is more than twice the size of D_d. Therefore, Seq2Seq outperforms Variational Decipher on the entire testing dataset D. The different performance of the two models on D_s and D_d is more obvious in Figure 4. The gap between the performance of the Seq2Seq model on D_s and D_d is much larger than that between the performance of the Variational Decipher on these two datasets.

Human Evaluation: We employed crowd-sourced workers to evaluate the deciphering results of two models. We randomly sampled 100 items of deciphering results from D_s and another 100 items from D_d. Each item composes a choice question and each choice question is assigned to five workers on Amazon Mechanical Turk. In each choice question, the workers are given the hate symbol, the referential definition, the original tweet and two machine-generated plain texts from the Seq2Seq model and Variational Decipher. Workers are asked to select the more reasonable of the two results. In each choice question, the order of the results from the two models is permuted. Ties are permitted for answers. We batch five items in one assignment and insert an artificial item with two identical outputs as a sanity check. The workers who fail to choose "tie" for that item are rejected from our test. The human evaluation results are shown in Table 2, which coincide with the results in Table 1 and Figure 4.

Discussion: When deciphering the hate symbols that have the same definitions as in the training dataset, the model can rely more on the surface forms of hate symbols than the tweet context to make a prediction because usually the hate symbols that share the same definitions also have similar surface forms. However, when it comes to the hate symbols with unseen definitions, simply relying on the surface forms cannot lead to a reasonable deciphering result. Instead, the model should learn the relationships between the con-

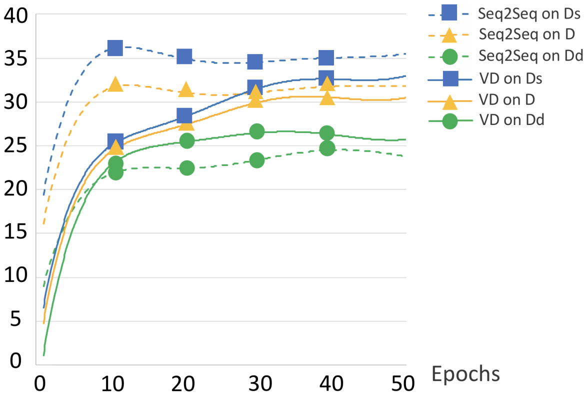
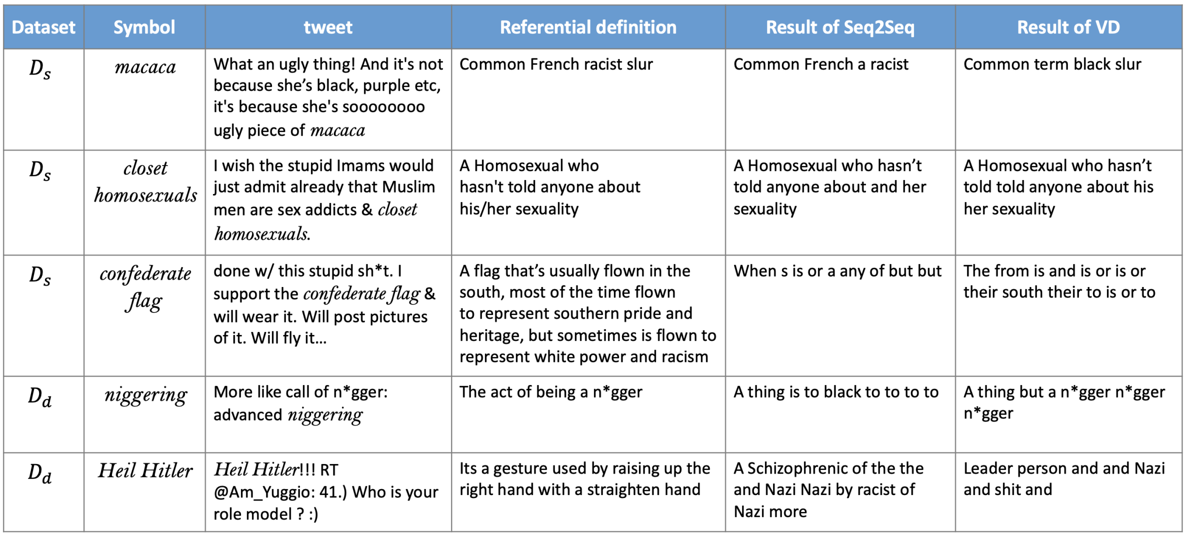


Figure 4: BLEU scores of two models on the testing dataset D, D_s and D_d. The three dotted curves represent the performance of the Seq2Seq model while the three solid curves represent the performance of the Variational Decipher.

Dataset	Seq2Seq Lose	Seq2Seq Win	Tie
D_s	31.0%	32.0%	37.0%
D_d	30.5%	22.0%	47.5%

Table 2: The results of human evaluation on two separate testing datasets D_s and D_d.



Dataset	Symbol	tweet	Referential definition	Result of Seq2Seq	Result of VD
D_s	*macaca*	What an ugly thing! And it's not because she's black, purple etc, it's because she's soooooooo ugly piece of *macaca*	Common French racist slur	Common French a racist	Common term black slur
D_s	*closet homosexuals*	I wish the stupid Imams would just admit already that Muslim men are sex addicts & *closet homosexuals.*	A Homosexual who hasn't told anyone about his/her sexuality	A Homosexual who hasn't told anyone about and her sexuality	A Homosexual who hasn't told told anyone about his her sexuality
D_s	*confederate flag*	done w/ this stupid sh*t. I support the *confederate flag* & will wear it. Will post pictures of it. Will fly it...	A flag that's usually flown in the south, most of the time flown to represent southern pride and heritage, but sometimes is flown to represent white power and racism	When s is or a any of but but	The from is and is or is or their south their to is or to
D_d	*niggering*	More like call of n*gger: advanced *niggering*	The act of being a n*gger	A thing is to black to to to to	A thing but a n*gger n*gger n*gger
D_d	*Heil Hitler*	*Heil Hitler*!!! RT @Am_Yuggio: 41.) Who is your role model ? :)	Its a gesture used by raising up the right hand with a straighten hand	A Schizophrenic of the the and Nazi Nazi by racist of Nazi more	Leader person and and Nazi and shit and

Figure 5: Some example errors in the generated results of our Seq2Seq model and Variational Decipher.

text information and the definition of the symbol. Therefore, the different performances of two models on the two testing datasets D_s and D_d indicate that the Seq2Seq model is better at capturing the similarities among different surface forms of a hate symbol, while the Variational Decipher is better at capturing the semantic relationship between the tweet context and the hate symbol. The Sequence-to-Sequence model tries to capture such kinds of relationships by compressing all the context information into a fixed length vector, so its deciphering strategy is actually behavior cloning. On the other hand, the Variational Decipher captures such relationships by explicitly modeling the posterior and likelihood distributions. The modeled distributions provide higher-level semantic information compared to the compressed context, which allows the Variational Decipher to generalize better to the symbols with unseen definitions. This explains why the gap between the performance of the Seq2Seq model on two datasets is larger.

5.3 Error Analysis

Figure 5 shows some example errors of the deciphering results of our Seq2Seq model and Variational Decipher. One problem with the deciphering results is that the generated sentences have poor grammatical structure, as shown in Figure 5. This is mainly because the size of our dataset is small, and the models need a much larger corpus to learn the grammar. We anticipate that the generation performance will be improved with a larger dataset.

For the hate symbols in D_s, the deciphering results are of high quality when the length of referential definitions are relatively short. An example is *macaca*, a French slur shows in Figure 5. The deciphering result of the Seq2Seq model is close to the referential definition. As to the Variational Decipher, although the result is not literally the same as the definition, the meaning is close. *closet homosexuals* in Figure 5 is another example. However, when the length of the referential definition increases, the performance of both models tends to be unsatisfactory, as the third example *confederate flag* shows in Figure 5. Although there exists the symbol *Confederate Flag* with the same definition in the training set, both models fail on this example. One possible reason is that the complexity of generating the referential definition grows substantially with the increasing length, so when the tweet context and the symbol itself cannot provide enough information, the generation model cannot learn the relationship between the symbol and its definition.

Deciphering hate symbols in D_d is much more challenging. Even for humans, deciphering completely new hate symbols is not a simple task. The two examples in Figure 5 show that the models have some ability to capture the semantic similarities. For the symbol *niggering*, the Variational Decipher generates the word *nigger* and Seq2Seq model generates *black*. For *Heil Hitler*, the Variational Decipher generates *leader person* and *Nazi*, while Seq2Seq also generates *Nazi*. Although these generated words are not in the definition, they still make some sense.

6 Conclusion

We propose a new task of learning to decipher hate symbols and create a symbol-rich tweet dataset. We split the testing dataset into two parts to analyze the characteristics of the Seq2Seq model and the Variational Decipher. The different performance of these two models indicates that the models can be applied to different scenarios of hate symbol deciphering. The Seq2Seq model outperforms the Variational Decipher for deciphering the hate symbols with similar definitions to that in the training dataset. This means the Seq2Seq model can better explain the hate symbols when Twitter users intentionally misspell or abbreviate common slur terms. On the other hand, the Variational Decipher tends to be better at deciphering hate symbols with unseen definitions, so it can be applied to explain newly created hate symbols on Twitter. Although both models show promising deciphering results, there still exists much room for improvement.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779. Association for Computational Linguistics.

Pete Burnap and Matthew L Williams. 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 680–686.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th ACM International Conference on World Wide Web*, pages 29–30.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 774–782.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association of Computational Linguistics*, 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 499–506.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 1558–1566.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044. Association for Computational Linguistics.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 71–76. Association for Computational Linguistics.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 413–417.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages 1278–1286.

Tim Salimans, Diederik Kingma, and Max Welling. 2015. Markov chain monte carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1218–1226.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 3483–3491.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016*, pages 88–93.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks

Ningyu Zhang[1,2,3] **Shumin Deng**[1,3] **Zhanlin Sun**[1,3] **Guanying Wang**[1,3]
Xi Chen[4] **Wei Zhang**[2,3] **Huajun Chen**[1,3*]
1. College of Computer Science and Technology, Zhejiang University
2. Alibaba Group
3. AZFT[†]Joint Lab for Knowledge Engine
4. Tencent
{3150105645,231sm,guanying_wgy,huajunsir}@zju.edu.cn
jasonxchen@tencent.com,{ningyu.zny,lantu.zw}@alibaba-inc.com

Abstract

We propose a distance supervised relation extraction approach for long-tailed, imbalanced data which is prevalent in real-world settings. Here, the challenge is to learn accurate "few-shot" models for classes existing at the tail of the class distribution, for which little data is available. Inspired by the rich semantic correlations between classes at the long tail and those at the head, we take advantage of the knowledge from data-rich classes at the head of the distribution to boost the performance of the data-poor classes at the tail. First, we propose to leverage implicit relational knowledge among class labels from knowledge graph embeddings and learn explicit relational knowledge using graph convolution networks. Second, we integrate that relational knowledge into relation extraction model by coarse-to-fine knowledge-aware attention mechanism. We demonstrate our results for a large-scale benchmark dataset which show that our approach significantly outperforms other baselines, especially for long-tail relations.

1 Introduction

Relation extraction (RE) is an important task in information extraction, aiming to extract the relation between two given entities based on their related context. Due to the capability of extracting textual information and benefiting many NLP applications (e.g., information retrieval, dialog generation, and question answering), RE appeals to many researchers. Conventional supervised models have been widely explored in this task (Zelenko et al., 2003; Zeng et al., 2014); however, their performance heavily depends on the scale and quality of training data.

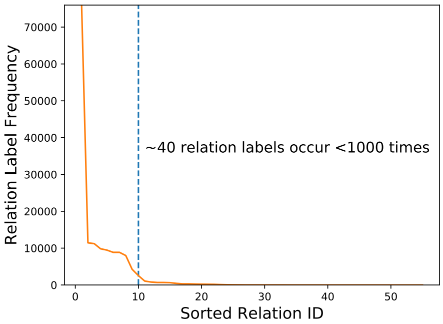


Figure 1: Label frequency distribution of classes without NA in NYT dataset.

To construct large-scale data, (Mintz et al., 2009) proposed a novel distant supervision (DS) mechanism to automatically label training instances by aligning existing knowledge graphs (KGs) with text. DS enables RE models to work on large-scale training corpora and has thus become a primary approach for RE recently (Wu et al., 2017; Feng et al., 2018). Although these DS models achieve promising results on common relations, their performance still degrades dramatically when there are only a few training instances for some relations. Empirically, DS can automatically annotate adequate amounts of training data; however, this data usually only covers a limited part of the relations. Many relations are long-tail and still suffer from data deficiency. Current DS models ignore the problem of long-tail relations, which makes it challenging to extract comprehensive information from plain text.

Long-tail relations are important and cannot be ignored. Nearly 70% of the relations are long-tail in the widely used New York Times (NYT) dataset[1] (Riedel et al., 2010; Lei et al., 2018) as shown in Figure 1. Therefore, it is crucial for mod-

---

[*] Corresponding author.

[†] Alibaba-Zhejiang University Frontier Technology Research Center

[1] http://iesl.cs.umass.edu/riedel/ecml/

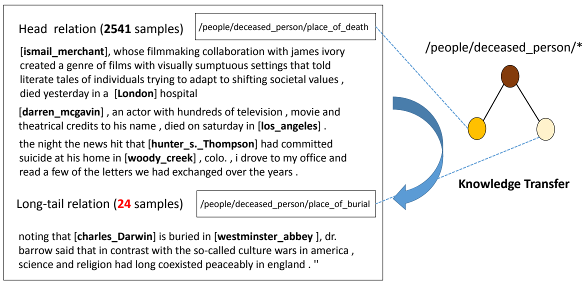
Proceedings of NAACL-HLT 2019, pages 3016–3025
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

els to be able to extract relations with limited numbers of training instances.

Dealing with long tails is very difficult as few training examples are available. Therefore, it is natural to transfer knowledge from data-rich and semantically similar head classes to data-poor tail classes (Wang et al., 2017). For example, the long-tail relation /people/deceased_person/place_of_burial and head relation /people/deceased_person/place_of_death are in the same branch /people/deceased_person/* as shown in Figure 2. They are semantically similar, and it is beneficial to leverage head relational knowledge and transfer it to the long-tail relation, thus enhancing general performance. In other words, long-tail relations of one entity tuple can have class ties with head relations, which can be leveraged to enhance RE for narrowing potential search spaces and reducing uncertainties between relations when predicting unknown relations (Ye et al., 2017). If one pair of entities contains /people/deceased_person/place_of_death, there is a high probability that it will contain /people/deceased_person/place_of_burial. If we can incorporate the relational knowledge between two relations, extracting head relations will provide evidence for the prediction of long-tail relations.

However, there exist two problems: (1) *Learning relational knowledge:* Semantically similar classes may contain more relational information that will boost transfer, whereas irrelevant classes (e.g., /location/location/contains and /people/family/country) usually contain less relational information that may result in negative transfer. (2) *Leveraging relational knowledge:* Integrating relational knowledge to existing RE models is challenging.

To address the problem of learning relational knowledge, as shown in (Lin et al., 2016; Ye et al., 2017), we use class embeddings to represent relation classes and utilize KG embeddings and graph convolution networks (GCNs) to extract implicit and explicit relational knowledge. Specifically, previous studies (Yang et al., 2015) have shown that the embeddings of semantically similar relations are located near each other in the latent space. For instance, the relation /people/person/place_lived and /people/person/nationality are more relevant, whereas the relation /people/person/profession has less correlation with the former two relations. Thus, it is natural to leverage this knowledge from KGs. However, because there are many one-to-multiple relations in KGs, the relevant information for each class may be scattered. In other words, there may not be enough relational signal between classes. Therefore, we utilize GCNs to learn explicit relational knowledge.



[**ismail_merchant**], whose filmmaking collaboration with james ivory created a genre of films with visually sumptuous settings that told literate tales of individuals trying to adapt to shifting societal values , died yesterday in a [**London**] hospital

[**darren_mcgavin**] , an actor with hundreds of television , movie and theatrical credits to his name , died on saturday in [**los_angeles**] .

the night the news hit that [**hunter_s._Thompson**] had committed suicide at his home in [**woody_creek**] , colo. , i drove to my office and read a few of the letters we had exchanged over the years .

noting that [**charles_Darwin**] is buried in [**westminster_abbey**], dr. barrow said that in contrast with the so-called culture wars in america , science and religion had long coexisted peaceably in england . ''

Figure 2: Head and long-tail relations.

To address the problem of leveraging relational knowledge, we first use convolution neural networks (Zeng et al., 2014, 2015) to encode sentences; then introduce coarse-to-fine knowledge-aware attention mechanism for combining relational knowledge with encoded sentences into bag representation vectors. The relational knowledge not only provides more information for relation prediction but also provides a better reference message for the attention module to raise the performance of long-tail classes.

Our experimental results on the NYT dataset show that: (1) our model is effective compared to baselines especially for long-tail relations; (2) leveraging relational knowledge enhances RE and our model is efficient in learning relational knowledge via GCNs.

2 Related Work

Relation Extraction. Supervised RE models (Zelenko et al., 2003; GuoDong et al., 2005; Mooney and Bunescu, 2006) require adequate amounts of annotated data for training which is time-consuming. Hence, (Mintz et al., 2009) proposd DS to automatically label data. DS inevitably accompanies with the wrong labeling problem. To alleviate the noise issue, (Riedel et al., 2010; Hoffmann et al., 2011) proposed multi-instance learning (MIL) mechanisms. Recently, neural models have been widely used for RE; those models can accurately capture textual relations without explicit linguistic analysis (Zeng et al., 2015;

Lin et al., 2016; Zhang et al., 2018a). To further improve the performance, some studies incorporate external information (Zeng et al., 2017; Ji et al., 2017; Han et al., 2018a) and advanced training strategies (Ye et al., 2017; Liu et al., 2017; Huang and Wang, 2017; Feng et al., 2018; Zeng et al., 2018; Wu et al., 2017; Qin et al., 2018). These works mainly adopt DS to make large-scale datasets and reduce the noise caused by DS, regardless of the effect of long-tail relations.

There are only a few studies on long-tail for RE (Gui et al., 2016; Lei et al., 2018; Han et al., 2018b). Of these, (Gui et al., 2016) proposed an explanation-based approach, whereas (Lei et al., 2018) utilized external knowledge (logic rules). These studies treat each relation in isolation, regardless of the rich semantic correlations between the relations. (Han et al., 2018b) proposed a hierarchical attention scheme for RE, especially for long-tail relations. Different from those approaches, we leverage implicit and explicit relational knowledge from KGs and GCNs rather than data-driven learned parameter spaces where similar relations may have distinct parameters, hindering the generalization of long-tail classes.

Knowledge Graph Embedding. Recently, several KG embedding models have been proposed. These methods learn low-dimensional vector representations for entities and relations (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015). TransE (Bordes et al., 2013) is one of the most widely used models, which views relations as translations from a head entity to a tail entity on the same low-dimensional hyperplane. Inspired by the rich knowledge in KGs, recent works (Han et al., 2018a; Wang et al., 2018; Lei et al., 2018) extend DS models under the guidance of KGs. However, these works neglect rich correlations between relations. Relation structure (relational knowledge) has been studied and is quite effective for KG completion (Zhang et al., 2018b). To the best of our knowledge, this is the first effort to consider the relational knowledge of classes (relations) using KGs for RE.

Graph Convolutional Networks. GCNs generalize CNNs beyond two-dimensional and one-dimensional spaces. (Defferrard et al., 2016) developed spectral methods to perform efficient graph convolutions. (Kipf and Welling, 2016) assumed the graph structure is known over input instances and apply GCNs for semi-supervised learning. GCNs were applied to relational data (e.g., link prediction) by (Schlichtkrull et al., 2018). GCNs have also had success in other NLP tasks such as semantic role labeling (Marcheggiani and Titov, 2017), dependency parsing (Strubell and McCallum, 2017), and machine translation (Bastings et al., 2017).

Two GCNs studies share similarities with our work. (1) (Chen et al., 2017) used GCNs on structured label spaces. However, their experiments do not handle long-tail labels and do not incorporate attention but use an average of word vectors to represent each document. (2) (Rios and Kavuluru, 2018) proposed a few-shot and zero-shot text classification method by exploiting structured label spaces with GCNs. However, they used GCNs in the label graph whereas we utilize GCNs in the hierarchy graph of labels.

3 Methodology

In this section, we introduce the overall framework of our approach for RE, starting with the notations.

3.1 Notations

We denote a KG as $\mathcal{G} = \mathcal{E}, \mathcal{R}, \mathcal{F}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F}$ indicate the sets of entities, relations and facts respectively. $(h, r, t) \in \mathcal{F}$ indicates that there is a relation $r \in \mathcal{R}$ between $h \in \mathcal{E}$ and $t \in \mathcal{E}$. We follow the MIL setting and split all instances into multiple entity-pair bags $\{\mathcal{S}_{h_1,t_1}, \mathcal{S}_{h_2,t_2}, ...\}$. Each bag $\mathcal{S}_{h_i,t_i}$ contains multiple instances $\{s_1, s_2, ...\}$ mentioning both entities h_i and t_i. Each instance s in these bags is denoted as a word sequence $s = \{w_1, w_2, ...\}$.

3.2 Framework

Our model consists of three parts as shown in Figure 3:

Instance Encoder. Given an instance and its mentioned entity pair, we employ neural networks to encode the instance semantics into a vector. In this study, we implement the instance encoder with convolutional neural networks (CNNs) given both model performance and time efficiency.

Relational Knowledge Learning. Given pretrained KG embeddings (e.g., TransE (Bordes et al., 2013)) as implicit relational knowledge, we employ GCNs to learn explicit relational knowledge. By assimilating generic message-passing inference algorithms with neural-network counterpart, we can learn better embeddings for Knowl-

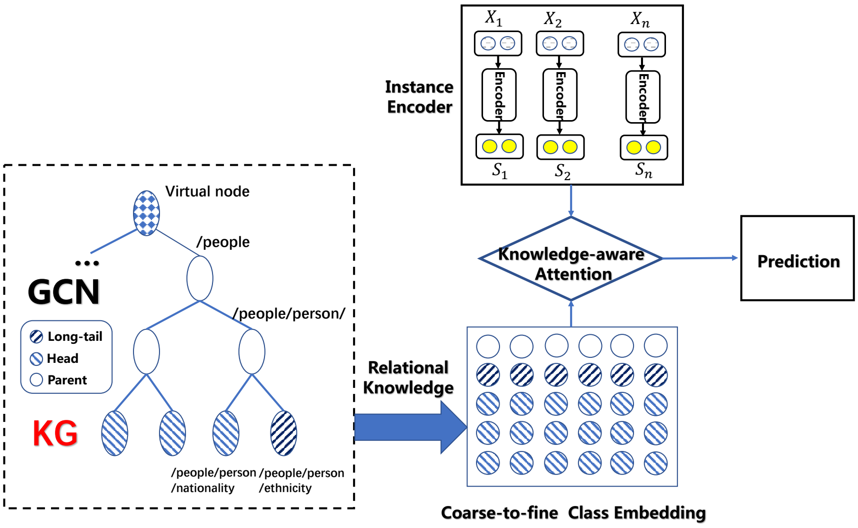


Figure 3: Architecture of our proposed model.

edge Relation. We concatenate the outputs of GCNs and the pretrained KG embeddings to form the final class embeddings.

Knowledge-aware Attention. Under the guidance of final class embeddings, knowledge-aware attention is aimed to select the most informative instance exactly matching relevant relation.

3.3 Instance Encoder

Given an instance $s = \{w_1, ..., w_n\}$ mentioning two entities, we encode the raw instance into a continuous low-dimensional vector x, which consists of an embedding layer and an encoding layer.

Embedding Layer. The embedding layer is used to map discrete words in the instance into continuous input embeddings. Given an instance s, we map each word w_i in the instance to a real-valued pretrained Skip-Gram (Mikolov et al., 2013) embedding $w_i \in \mathbb{R}^{d_w}$. We adopt position embeddings following (Zeng et al., 2014). For each word w_i, we embed its relative distances to the two entities into two d_p-dimensional vectors. We then concatenate the word embeddings and position embeddings to achieve the final input embeddings for each word and gather all the input embeddings in the instance. We thus obtain an embedding sequence ready for the encoding layer.

Encoding Layer. The encoding layer aims to compose the input embeddings of a given instance into its corresponding instance embedding. In this study, we choose two convolutional neural architectures, CNN (Zeng et al., 2014) and PCNN (Zeng et al., 2015) to encode input embeddings into instance embeddings. Other neural architectures such as recurrent neural networks (Zhang and Wang, 2015) can also be used as sentence encoders. Because previous works show that both convolutional and recurrent architectures can achieve comparable state-of-the-art performance, we select convolutional architectures in this study. Note that, our model is independent of the encoder choices, and can, therefore, be easily adapted to fit other encoder architectures.

3.4 Relational Knowledge Learning through KG Embeddings and GCNs.

Given pretrained KG embeddings and predefined class (relation) hierarchies[2], we first leverage the implicit relational knowledge from KGs and initialize the hierarchy label graph; then we apply two layer GCNs to learn explicit fine-grained relational knowledge from the label space.

Hierarchy Label Graph Construction. Given a relation set $\mathcal{R}$ of a KG $\mathcal{G}$ (e.g., Freebase), which consists of base-level relations (e.g., /people/person/ethnicity), we can generate the corresponding higher-level relation set $\mathcal{R}^H$. The relations in a high-level set (e.g., people) are more general and common; they usually contain several sub-relations in the base-level set. The relation hierarchies are tree-structured, and the generation process can be done recursively. We use a virtual father node to construct the highest level associa-

---

[2] For datasets without predefined relation hierarchies, hierarchy clustering (Johnson, 1967) or K-means can construct relation hierarchies (Zhang et al., 2018b); details can be found in supplementary materials.

tions between relations as shown in Figure 3. In practice, we start from $\mathcal{R}^0 = R$ which is the set of all relations we focus on for RE, and the generation process is performed $L-1$ times to get the hierarchical relation sets $\{\mathcal{R}^0, \mathcal{R}^1, ..., \mathcal{R}^L\}$, where $\mathcal{R}^L$ is the virtual father node. Each node has a specific type $t \in \{0, 1, ..., L\}$ to identify its layer hierarchies. For example, as shown in Figure 3, node /people/person/ethnicity has a specific type 0 to indicate it is in the bottom layer of the graph. The vectors of each node in the bottom layer are initialized through pretrained TransE (Bordes et al., 2013) KG embeddings. Other KG embeddings such as TransR (Lin et al., 2015) can also be adopted. Their parent nodes are initialized by averaging all children vectors. For example, the node vector of /people/person/ is initialized by averaging all the nodes under the branch of /people/person/* (all child nodes).

GCN Output Layer. Due to one-to-multiple relations and incompleteness in KGs, the implicit relevant information obtained by KG embeddings for each label is not enough. Therefore, we apply GCNs to learn explicit relational knowledge among labels. We take advantage of the structured knowledge over our label space using a two-layer GCNs. Starting with the pretrained relation embedding $v_i^{implicit} \in \mathbb{R}^d$ from KGs, we combine the label vectors of the children and parents for the i-th label to form,

$$v_i^1 = f(W^1 v_i + \sum_{j \in \mathcal{N}_p} \frac{W_p^1 v_j}{|\mathcal{N}_p|} + \sum_{j \in \mathcal{N}_c} \frac{W_c^1 v_j}{|\mathcal{N}_c|} + b_g^1) \tag{1}$$

where $W^1 \in \mathbb{R}^{q*d}$, $W_p^1 \in \mathbb{R}^{q*d}$, $W_c^1 \in \mathbb{R}^{q*d}$, $b_g^1 \in \mathbb{R}^q$, f is the rectified linear unit (Nair and Hinton, 2010) function, and $\mathcal{N}_c$ ($\mathcal{N}_p$) is the index set of the i-th labels children (parents). We use different parameters to distinguish each edge type where parent edges represent all edges from high level labels and child edges represent all edges from low level labels. The second layer follows the same formulation as the first layer and outputs $v_i^{explicit}$. Finally, we concatenate both pretrained $v_i^{implicit}$ with GCNs node vector $v_i^{explicit}$ to form hierarchy class embeddings,

$$q_r = v_i^{implicit} || v_i^{explicit} \tag{2}$$

where $q_r \in \mathbb{R}^{d+q}$.

3.5 Knowledge-aware Attention

Traditionally, the output layer of PCNN/CNN would learn label specific parameters optimized by a cross-entropy loss. However, the label specific parameters spaces are unique to each relation, matrices associated with the long-tails can only be exposed to very few facts during training, resulting in poor generalization. Instead, our method attempts to match sentence vectors to their corresponding class embeddings rather than learning label specific attention parameters. In essence, this becomes a retrieval problem. Relevant information from class embeddings contains useful relational knowledge for long-tails among labels.

Practically, given the entity pair (h, t) and its bag of instances $S_{h,t} = \{s_1, s_2, ..., s_m\}$, we achieve the instance embeddings $\{s_1, s_2, ..., s_m\}$ using the sentence encoder. We group the class embeddings according to their type (i.e., according to their layers in the hierarchy label graph), e.g., $q_{r^i}, i \in \{0, 1, ..., L\}$. We adopt $q_{r^i}, i \neq L$ (layer L is the virtual father node) as layer-wise attention query vector. Then, we apply coarse-to-fine knowledge-aware attention to them to obtain the textual relation representation $r_{h,t}$. For a relation r, we construct its hierarchical chain of parent relations $(r^0, ..., r^{L-1})$ using the hierarchy label graph, where r^{i-1} is the sub-relation of r^i. We propose the following formulas to compute the attention weight (similarity or relatedness) between each instances feature vector s_k and q_{r^i},

$$\begin{aligned} e_k &= W_s(tanh[s_k; q_{r^i}]) + b_s \\ \alpha_k^i &= \frac{exp(e_k)}{\sum_{j=1}^{m} exp(e_j)} \end{aligned} \tag{3}$$

where $[x_1; x_2]$ denotes the vertical concatenation of x_1 and x_2, W_s is the weight matrix, and b_s is the bias. We compute attention operations on each layer of hierarchy label graph to obtain corresponding textual relation representations,

$$r_{h,t}^i = ATT(q_{r^i}, \{s_1, s_2, ..., s_m\}) \tag{4}$$

Then we need to combine the relation representations on different layers. Direct concatenation of all the representations is a straightforward choice. However, different layers have different contributions for different tuples. For example, the relation /location/br_state/ has only one sub-relation /location/br_state/capital, which indicates that it is more important. In other words, if the sentence has high attention weights on relation /location/br_state/, it has a very high probability to have relation /location/br_state/capital. Hence, we use an attention mechanism to emphasize the layers,

$$\begin{aligned} g_i &= W_g tanh(r_{h,t}) \\ \beta_i &= \frac{exp(g_i)}{\sum_{j=0}^{L-1} exp(g_j)} \end{aligned} \tag{5}$$

where W_g is a weight matrix, $r_{h,t}$ is referred to as a query-based function that scores how well the input textual relation representations and the predict relation r match. The textual relation representations in each layer are computed as,

$$r^i_{h,t} = \beta_i r^i_{h,t} \tag{6}$$

We simply concatenate the textual relation representations on different layers as the final representation,

$$r_{h,r} = Concat(r^0_{h,t}, .., r^{L-1}_{h,t}) \tag{7}$$

The representation $r_{h,t}$ will be finally fed to compute the conditional probability $\mathcal{P}(r|h, t, \mathcal{S}_{h,t})$,

$$\mathcal{P}(r|h, t, \mathcal{S}_{h,t}) = \frac{exp(o_r)}{\sum_{\tilde{r} \in R} exp(o_{\tilde{r}})} \tag{8}$$

where o is the score of all relations defined as,

$$o = M r_{h,t} \tag{9}$$

where M is the representation matrix to calculate the relation scores. Note that, attention weight q_{r^i} is obtained from the outputs of GCNs and pretrained KG embeddings, which can provide more informative parameters than data-driven learned parameters, especially for long-tails.

4 Experiments

4.1 Datasets and Evaluation

We evaluate our models on the NYT dataset developed by (Riedel et al., 2010), which has been widely used in recent studies (Lin et al., 2016; Liu et al., 2017; Wu et al., 2017; Feng et al., 2018). The dataset has 53 relations including the NA relation, which indicates that the relations of instances are not available. The training set has 522611 sentences, 281270 entity pairs, and 18252 relational facts. In the test set, there are 172448 sentences, 96678 entity pairs, and 1950 relational facts. In both training and test set, we truncate sentences with more than 120 words into 120 words.

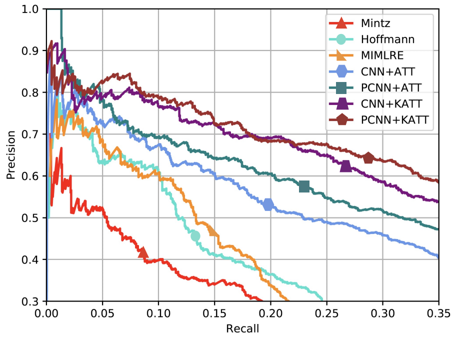


Figure 4: Precision-recall curves for the proposed model and various baseline models.

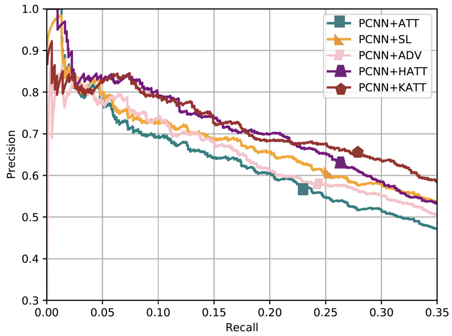


Figure 5: Precision-recall curves for the proposed model and various attention-based neural models.

We evaluate all models in the held-out evaluation. It evaluates models by comparing the relational facts discovered from the test articles with those in Freebase and provides an approximate measure of precision without human evaluation. For evaluation, we draw precision-recall curves for all models. To further verify the effect of our model for long-tails, we follow previous studies (Han et al., 2018b) to report the Precision@N results. The dataset and baseline code can be found on Github [3].

4.2 Parameter Settings[4]

To fairly compare the results of our models with those baselines, we also set most of the experimental parameters by following (Lin et al., 2016). We apply dropout on the output layers of our models to prevent overfitting. We also pretrain the sentence encoder of PCNN before training our model.

---

[3] https://github.com/thunlp/OpenNRE

[4] Details of hyper-parameters settings and evaluation of different instances can be found in supplementary materials

Training Instances		<100			<200		
Hits@K (Macro)		10	15	20	10	15	20
CNN	+ATT	<5.0	<5.0	18.5	<5.0	16.2	33.3
	+HATT	5.6	31.5	57.4	22.7	43.9	65.1
	+KATT	**9.1**	**41.3**	**58.5**	**23.3**	**44.1**	**65.4**
PCNN	+ATT	<5.0	7.4	40.7	17.2	24.2	51.5
	+HATT	29.6	51.9	61.1	41.4	60.6	68.2
	+KATT	**35.3**	**62.4**	**65.1**	**43.2**	**61.3**	**69.2**

Table 1: Accuracy (%) of Hits@K on relations with training instances fewer than 100/200.

4.3 Overall Evaluation Results

To evaluate the performance of our proposed model, we compare the precision-recall curves of our model with various previous RE models. The evaluation results are shown in Figure 4 and Figure 5. We report the results of neural architectures including CNN and PCNN with various attention based methods: **+KATT** denotes our approach, **+HATT** is the hierarchical attention method (Han et al., 2018b), **+ATT** is the plain selective attention method over instances (Lin et al., 2016), **+ATT+ADV** is the denoising attention method by adding a small adversarial perturbation to instance embeddings (Wu et al., 2017), and **+ATT+SL** is the attention-based model using soft-labeling method to mitigate the side effect of the wrong labeling problem at entity-pair level (Liu et al., 2017). We also compare our method with feature-based models, including **Mintz** (Mintz et al., 2009), **MultiR** (Hoffmann et al., 2011) and **MIML** (Surdeanu et al., 2012).

As shown in both figures, our approach achieves the best results among all attention-based models. Even when compared with PCNN+HATT, PCNN+ATT+ADV, and PCNN+ATT+SL, which adopt sophisticated denoising schemes and extra information, our model is still more advantageous. This indicates that our method can take advantage of the rich correlations between relations through KGs and GCNs, which improve the performance. We believe the performance of our model can be further improved by adopting additional mechanisms like adversarial training, and reinforcement learning, which will be part of our future work.

4.4 Evaluation Results for Long-tail Relations

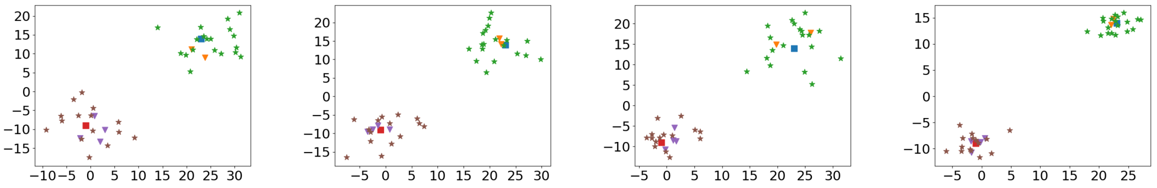
To further demonstrate the improvements in performance for long-tail relations, following the study by (Han et al., 2018b) we extract a subset of the test dataset in which all the relations have fewer than 100/200 training instances. We employ the Hits@K metric for evaluation. For each entity pair, the evaluation requires its corresponding golden relation in the first K candidate relations recommended by the models. Because it is difficult for the existing models to extract long-tail relations, we select K from $\{10,15,20\}$. We report the macro average Hits@K accuracies for these subsets because the micro-average score generally overlooks the influences of long-tails. From the results shown in Table 1, we observe that for both CNN and PCNN models, our model outperforms the plain attention model and the HATT model. Although our KATT method has achieved better results for long-tail relations as compared to both plain ATT method and HATT method, the results of all these methods are still far from satisfactory. This indicates that distantly supervised RE models still suffer from the long-tail relation problem, which may require additional schemes and extra information to solve this problem in the future.

Training Instances	<100			<200		
Hits@K (Macro)	10	15	20	10	15	20
+KATT	**35.3**	**62.4**	**65.1**	**43.2**	**61.3**	**69.2**
w/o hier	34.2	62.1	65.1	42.5	60.2	68.1
w/o GCNs	30.5	61.9	63.1	39.5	58.4	66.1
Word2vec	30.2	62.0	62.5	39.6	57.5	65.8
w/o KG	30.0	61.0	61.3	39.5	56.5	62.5

Table 2: Results of ablation study with PCNN.

4.5 Ablation Study

To analyze the contributions and effects of different technologies in our approach, we perform ablation tests. **+KATT** is our method; **w/o hier** is the method without coarse-to-fine attention (only utilizes bottom node embeddings of the hierarchy label graph), which implies no knowledge transfer from its higher level classes; **w/o GCN** is the method without GCNs, which implies no explicit relational knowledge; **Word2vec** is the method in which the node is initialized with pretrained Skip-Gram (Mikolov et al., 2013) embeddings; and **w/o KG** is the method in which the node is initialized with random embeddings, which implies no prior relational knowledge from KGs. From the evaluation results in Table 2, we observe that the performance slightly degraded without coarse-to-fine attention, which proves that knowledge transfer from the higher node is useful. We also noticed that the performance slightly degraded without KG or using word embeddings, and the perfor-



(a) HATT (b) +KG (c) +GCNs (d) KATT

Figure 6: T-SNE visualizations of class embeddings. Cluster in the upper right is the relation /location/*/* and cluster in the bottom left is the relation /people/*/*). The square, triangle, and star refer to the high (/location) middle (/location/location/) and base (/location/location/contains) level relations, respectively.

mance significantly degraded when we removed GCNs. This is reasonable because GCNs can learn more explicit correlations between relation labels, which boost the performance for long-tail relations.

/people/deceased_person/place_of_burial	**HATT**	**KATT**
richard_wagner had his **bayreuth**, with its festspielhaus specially designed to accommodate his music dramas.	0.21	0.07
wotan and alberich are **richard_wagner**; and the rheingold and valhalla are wagner's real-life grail, the opera house in **bayreuth**.	0.15	0.13
/location/br_state/capital	**HATT**	**KATT**
there's violence everywhere, said ms. mesquita, who, like her friend, lives in **belo_horizonte**, the capital of **minas_gerais** state	0.47	0.51
all the research indicates that we are certain to find more gas in th amazon, eduardo braga, the governor of **amazonas**, said in an interview in **manaus**	0.46	0.45

Table 3: Example sentences for case study.

4.6 Case Study

We give some examples to show how our method affects the selection of sentences. In Table 3, we display the sentence's attention score in the lowest level[5]. Both the relation /people/deceased_person/place_of_burial (24 instances) and /location/br_state/capital (4 instances) are long-tail relations. On one hand, relation /people/deceased_person/place_of_burial has semantically similar data-rich relation such as /people/deceased_person/place_of_death. We observe that HATT erroneously assigns high attention to the incorrect sentence whereas KATT successfully assigns the right attention weights, which demonstrates the efficacy of knowledge transfer from semantically similar relations (Both HATT and KATT methods can take advantage of knowledge transfer of high-level relations.). On the other hand, the relation /location/br_state/capital does not have semantically similar relations. However, we notice that KATT still successfully assigns the right attention weights, which demonstrates the efficacy of knowledge transfer from high-level relations using coarse-to-fine knowledge-aware attention.

4.7 Visualizations of Class Embeddings

We visualize the class embeddings via t-SNE (Maaten and Hinton, 2008) to further show how GCNs and KG embeddings can help RE for long-tail relations. We observe that (1) Figure 6(a) and 6(d) show that semantically similar class embeddings are closer with GCNs and pretrained KG embeddings, which help select long-tail instances. (2) Figure 6(b) and 6(c) show that KG embeddings and GCNs have different contributions for different relations to learn relational knowledge between classes. For example, /location/location/contain has a sparse hierarchy structure, which leads to inefficient learning for GCNs; therefore, the relative distance changes only slightly, which reveals the necessity of implicit relational knowledge from KGs. (3) Figure 6(d) shows that there are still some semantically similar class embeddings located far away, which may degrade the performance for long-tails. This may be caused by either sparsity in the hierarchy label graph or equal treatment for nodes with the same parent in GCNs, which is not a reasonable hypothesis. We will address this by integrating more information such as relation descriptions or combing logic reasoning as a part of future work.

---

[5] Both HATT and KATT methods can successfully select the correct sentence at the higher-level; details can be found in supplementary materials.

5 Conclusion and Future Work

In this paper, we take advantage of the knowledge from data-rich classes at the head of distribution to boost the performance of the data-poor classes at the tail. As compared to previous methods, our approach provides fine-grained relational knowledge among classes using KG and GCNs, which is quite effective and encoder-agnostic.

In the future, we plan to explore the following directions: (1) We may combine our method with recent denoising methods to further improve performance. (2) We may combine rule mining and reasoning technologies to learn better class embeddings to boost performance. (3) It will be promising to apply our method to zero-shot RE and further adapt to other NLP scenarios.

Acknowledgments

We want to express gratitude to the anonymous reviewers for their hard work and kind comments, which will further improve our work in the future. This work is funded by NSFC91846204/61473260, national key research program YS2018YFB140004, Alibaba CangJingGe (Knowledge Engine) Research Plan and Natural Science Foundation of Zhejiang Province of China (LQ19F030001).

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Meihao Chen, Zhuoru Lin, and Kyunghyun Cho. 2017. Graph convolutional networks for classification with a structured label space. *arXiv preprint arXiv:1710.04908*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Yaocheng Gui, Qian Liu, Man Zhu, and Zhiqiang Gao. 2016. Exploring long tail data in distantly supervised relation extraction. In *Natural Language Understanding and Intelligent Applications*, pages 514–522. Springer.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018a. Neural knowledge acquisition via mutual attention between knowledge graph and text.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018b. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550. Association for Computational Linguistics.

Yi Yao Huang and William Yang Wang. 2017. Deep residual learning for weakly-supervised relation extraction. *arXiv preprint arXiv:1707.08866*.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of AAAI*, pages 3060–3066.

Stephen C Johnson. 1967. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kai Lei, Daoyuan Chen, Yaliang Li, Nan Du, Min Yang, Wei Fan, and Ying Shen. 2018. Cooperative denoising for distantly supervised relation extraction. In *Proceedings of Coling*, pages 426–436.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, volume 1, pages 2124–2133.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of ACL*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of EMNLP*, pages 3132–3142.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Emma Strubell and Andrew McCallum. 2017. Dependency parsing with dilated iterated graph cnns. *arXiv preprint arXiv:1705.00403*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465. Association for Computational Linguistics.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of EMNLP*, pages 2246–2255.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Proceedings of NIPS*, pages 7029–7039.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*, pages 1778–1783.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *Proceedings of ICLR*.

Hai Ye, Wenhan Chao, Zhunchen Luo, and Zhoujun Li. 2017. Jointly extracting relations with class ties via effective deep ranking. In *Proceedings of ACL*, volume 1, pages 1810–1820.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceddings of EMNLP*.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Processings of AAAI*, volume 2, page 3.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. 2018a. Attention-based capsule networks with dynamic routing for relation extraction. In *Proceedings of EMNLP*.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. 2018b. Knowledge graph embedding with hierarchical relation structure. In *Proceedings of EMNLP*, pages 3198–3207.

GAN Driven Semi-distant Supervision for Relation Extraction

Pengshuai Li[1], **Xinsong Zhang**[1], **Weijia Jia**[2,1*] and **Hai Zhao**[1*]
[1]Dept. of CSE, Shanghai Jiao Tong University, Shanghai, China
[2]State Key Lab of IoT for Smart City, CIS, University of Macau, Macao, SAR China
`{pengshuai.li,xszhang0320}@sjtu.edu.cn`
`{jia-wj,zhaohai}@cs.sjtu.edu.cn`

Abstract

Distant supervision has been widely used in relation extraction tasks without hand-labeled datasets recently. However, the automatically constructed datasets comprise numbers of wrongly labeled negative instances due to the incompleteness of knowledge bases, which is neglected by current distant supervised methods resulting in seriously misleading in both training and testing processes. To address this issue, we propose a novel semi-distant supervision approach for relation extraction by constructing a small accurate dataset and properly leveraging numerous instances without relation labels. In our approach, we construct accurate instances by both knowledge base and entity descriptions determined to avoid wrong negative labeling and further utilize unlabeled instances sufficiently using generative adversarial network (GAN) framework. Experimental results on real-world datasets show that our approach can achieve significant improvements in distant supervised relation extraction over strong baselines.

1 Introduction

Relation extraction aims to identify relations for a pair of entities in a sentence to construct relation triples like [*Steve Jobs, Founder, Apple*]. It has been well studied by supervised approaches with hand-labeled data. However, supervised methods are limited to costly hand-labeled training sets and hard to be extended to large-scale relations. To break the bottleneck of hand-labeled training set, distant supervision (Mintz et al., 2009) automatically construct datasets with knowledge bases. It assumes that if two entities have a known relation in a knowledge base, all sentences that mention these two entities will probably express the same relation and can be called positive instances. At the same time, it treats sentences as negative instances whose entity pairs do not have a known relation in knowledge bases. Due to the strong assumption, instances are likely to be mislabeled. To alleviate the wrong labeling problem, distant supervised methods have been implemented with multi-instance learning and neural networks (Riedel et al., 2010; Hoffmann et al., 2011; Zeng et al., 2015; Lin et al., 2016, 2017). However, previous works focus on positive instances and few methods have addressed the issue of false negative instances.

The false negative instances, which contain true relations, are misclassified sentences in the negative set due to the incomplete nature of knowledge bases. For example, over 70% of people included in Freebase have no known place of birth (Dong et al., 2014). As shown in Figure 1, *S1* presents the relation *place of birth*, while it is labeled as a negative instance. The other three sentences are mislabeled in the same way. The missing relation triples in knowledge bases yield numbers of false negative instances in the automatically labeled dataset. These instances will not only mislead the training method to an unreliable convergence but also make the measurement criteria inaccurate in the testing process. Table 1 compares the precision of automatic and manual evaluation methods for top N predictions by the previous relation extractor (Lin et al., 2016) on the NYT dataset. From the table, we can see that manual evaluation is more precise than automatic evaluation by over 19.8%.

---

[*] Corresponding authors: Weijia Jia, Hai Zhao, {jia-wj, zhaohai}@cs.sjtu.edu.cn. This work is supported by National China 973 Project No. 2015CB352401; Chinese National Research Fund (NSFC) Key Project No. 61532013 and No. 61872239. 0007/2018/A1, DCT-MoST Joint-project No. 025/2015/AMJ,FDCT,SAR Macau, China, and University of Macau Grant Nos: MYRG2018-00237-RTO, CPG2019-00004-FST and SRG2018-00111-FST, National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of NAACL-HLT 2019, pages 3026–3035
Minneapolis, Minnesota, June 2 - June 7, 2019.

ID	Instances	Dataset Label	Predicted Label
S1	[James Hillier] was born in [Brantford], Ontario.	NA	PB
S2	Dr. Fortner will be interred in [Bedford], [Indiana] with his parents.	NA	LC
S3	"This is an expression of what has been going on", archbishop [Phillip Aspinall] of [Australia] said at a news briefing here.	NA	PN
S4	What Dr. Sims did is called user-driven innovation by [Eric Von Hippel], a professor at the [Massachusetts Institute of Technology]'s Sloan School of management.	NA	PC

PB: /person/place of birth
LC: /location/contains
PN: /person/nationality
PC: /person/company
NA: non-relation

Figure 1: Illustration of the false negative instances in relation extraction by distant supervision. Instances are selected from a widely used dataset NYT (Riedel et al., 2010).

The huge bias mainly comes from false negative instances in the testing set, which severely limits the upper bound of accuracy for relation extraction. Therefore, handling false negative instances is a pivotal issue to improve the performance of distant supervised relation extraction.

Evaluations	P@100	P@200	P@300
Automatic	76.2	73.1	67.4
Manual	96.0(+19.8)	95.5(+22.4)	91.0(+23.6)

Table 1: The Precision at top N predictions (%) of the model Lin et al. (2016) upon automatic and manual evaluations on the NYT Dataset

To alleviate the effect of false negative instances, there are two possible ways. One is improving the accuracy of the automatically labeled dataset, and the other is properly leveraging unlabeled instances which cannot be labeled as positive or negative. The former way is to construct an accurate dataset by filtering credible negative instances but limited by high annotation cost and the resulting dataset size. The latter way is to train relation extraction models with abundant unlabeled instances but restricted by the prerequisite of an accurate dataset used as ground truth. Therefore, we propose a novel semi-distant supervised approach by integrating both ways to decrease the influence of false negative instances for better relation extraction.

In our approach, we additionally use entity descriptions together with a knowledge base to construct an accurate dataset. Supervised by the dataset as ground truth, to effectively exploit numbers of unlabeled instances, we train our relation extractor using a generative adversarial network (GAN) framework. In detail, We propose a three-player min-max game to generate proper relation representations for unlabeled instances in an adversarial way which minimizes the difference between labeled and unlabeled data and maximizes the probability of distinguishing from each other at the same time. The experiments demonstrate that our approach is effective and outperforms the state-of-the-art work. In summary, we make the following major contributions:

- We propose a novel semi-distant supervision method for relation extraction to alleviate the influence of false negative instances.
- To the best of our knowledge, we are the first to generate valid relation representations for sentences by an adversarial algorithm. Numbers of unlabeled instances are used to improve the performance of relation extraction. Moreover, our generative adversarial training strategy is proved effective on an additional sentiment classification with sixteen real-world datasets.
- We construct a new accurate dataset for relation extraction extended from the NYT dataset. Our approach increases the area of the Precision-Recall curve from 0.39 to 0.56 over the baselines.

2 Related Work

To extend relation extraction to large-scale datasets, distant supervision (Mintz et al., 2009) automatically labeled training sets with knowledge bases such as Freebase. Although this method is working well for large-scale relation extraction, it is trapped in the wrong labeling problem for positive instances. To deal with this problem, multi-instance learning was combined with distant supervision (Riedel et al., 2010; Hoffmann et al., 2011). Inspired by the pioneering work, a series of later studies were conducted to further improve distant supervised relation extraction with methods such as multi-instance multi-label learning (Surdeanu et al., 2012), graph model for label generation (Takamatsu et al., 2012), partial supervision (Angeli et al., 2014), matrix completion with low-rank criterion (Fan et al., 2014) and modeling the neighbor consistency with Markov logic (Han and Sun, 2016).

However, the performance of the methods mentioned above strongly depends on the quality of human-designed features. With the development of neural models, relation features with semantic meaning can be accurately, simply and automatically extracted. Zeng et al. (2015) proposed the first neural relation extraction with distant supervision. Mnih et al. (2014), Lin et al. (2016), Zhang et al. (2018), Han et al. (2018) and Du et al. (2018) showed that attention model could improve the accuracy of neural relation extraction. Another similar work (Ji et al., 2017) assigned better attention weights with extra data like entity descriptions. DSGAN (Qin et al., 2018a), a GAN-based method, was also used to recognize true positive instances from noisy datasets. To further alleviate the effect of wrong labeling problem, soft-label training algorithm (Liu et al., 2017b), reinforcement learning methods (Feng et al., 2018; Qin et al., 2018b) and additional side information (Vashishth et al., 2018; Wang et al., 2018) have been used. Most recently, a few methods focused on the pre-training embeddings for word tokens and relations including adversarial training (Wu et al., 2017), transfer learning (Liu et al., 2018) and relation decoder (Su et al., 2018).

All the above methods mainly pay attention to positive instances. Whereas, few studies work on the quality of negative instances, which is exactly the focus of this paper. We effectively construct a reliable dataset with both entity descriptions and a knowledge base, and thus propose a novel semi-distant supervised method to extract relations precisely.

3 Method

In the distant supervised relation extraction paradigm, all sentences labeled by a relation triple constitute a bag, and each sentence is called an instance. The relation triple is described as [*head, relation, tail*], where $head$ and $tail$ are both entities. We extract relation features from labeled training bags and then predict relations for unseen bags in the test set. This section presents our method about constructing an accurate dataset, the sentence encoder for relation representation and the semi-supervised way for relation extraction.

3.1 Dataset Construction

To reduce false negative instances, we construct a new reliable dataset extended from a widely used dataset NYT (Riedel et al., 2010) with entity descriptions. Entity descriptions are crawled from Wikipedia with entity name matching[1]. We assume that if an entity is relevant to another entity, its name is possibly mentioned in the description of the other entity. For example, the entity *Apple Inc.* is mentioned in the description of *Steve Jobs*. To verify the assumption, we count the number of all the accurate positive instances whose entity descriptions mention the other entity name in the NYT corpora. There are 163,108 positive sentences in total, in which 161,392 ones contain entity pairs that related to each other in their descriptions at least once. In other words, over 98.9% instances in positive set fitting our assumption indicates that most entity pairs in positive instances contain each other in their descriptions. Therefore, a former negative instance has a big chance to be credible negative if any of its entities is not mentioned in the description of the other one. Excluding instances that contain entity pairs related to each other in their descriptions, we can obtain more confident negative instances. Finally, we filter credible positive and negative instances from the dataset, and the other instances are unlabeled ones that cannot be labeled as positive or negative.

3.2 Sentence Encoder

3.2.1 Input Embedding

We pre-train input embeddings of word tokens including word and position embeddings. Word embeddings are distributed representations that map each word to a vector $word \in \mathcal{R}^w$, where the parameter w indicates the dimension of the vector. The vectors are trained in advance by $word2vec$ in the setting of Skip-gram (Mikolov et al., 2013). In the task of relation extraction, the relative positions of input tokens are important information.

---

[1] One entity name may refer to multiple entities which have their own pages. In our work, all the matched pages are collected together to obtain its description.

Position embeddings are defined as the combination of the relative distances from the current word to *head* and *tail*. For instance, the relative distances from [*co-founder*] to [*Steve Jobs*] and [*Apple*] are respectively 3 and -6 in the sentence *Steve Jobs was the co-founder and the CEO of the Apple*. We encode distances to vectors $position \in \mathcal{R}^p$, where p is the dimension. The position embeddings are initialized randomly and updated in the training process. Finally, word embeddings and position embeddings are concatenated together to feed the neural model. We denote all the words in an instance as an initial vector sequence $b^* = \{x_1, \cdots, x_i, \cdots, x_q\}$, where $x_i \in \mathcal{R}^{w+p}$ and q is the number of words in the instance b^*.

3.2.2 Convolutional Encoder

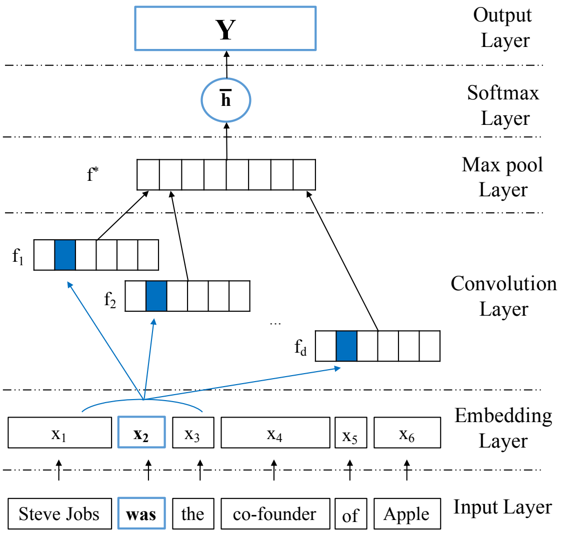


Figure 2: The architecture of our sentence encoder illustrating the procedure for handling one instance and predicting the relation between [Apple] and [Steve Jobs]. All forms of f are hidden states, $\bar{h}$ is relation representation of the sentence and Y represents relation labels.

Convolutional Neural Network (CNN) is a widely used structure for sentence encoder as shown in Figure 2. With the input embeddings, the convolutional layer extracts local features with a sliding window of length k over the input tokens. In the figure, we extract local features from 3 ($k = 3$) adjacent word tokens with dot production between convolutional kernels and input embeddings. The convolutional kernels are weight vectors represented by $W \in \mathcal{R}^{d \times k(w+p)}$ and the number of kernels is d. In summary, the convolutional operation follows the equation,

$$f_{ij} = W_i \cdot [x_{j-1}; x_j; x_{j+1}], \tag{1}$$

where $[x; y]$ denotes the vertical concatenation of x and y. f_{ij} presents j-th value of the i-th filter, where i and j are in range $[1, d]$ and $[1, q]$ respectively. Out-of-range input values such as x_0 and x_{q+1} are taken to be zero. A max-pooling operation selects the most important features of each f_i with $f_i^* = max(f_{ij})$, where $f^* \in \mathcal{R}^d$. Furthermore, PCNN (Zeng et al., 2015) improves the max-pooling operation with a piecewise method whose outputs of convolutional filters are divided into three segments by *head* and *tail* entities. Therefore, the max pooling procedure is performed in three segments separately.

Then, we summarize f^* to $\bar{h}$ by a non-linear function such as the hyperbolic tangent. The final feature vector $\bar{h}$ is fed into output layer after the softmax method $\hat{p} = softmax(W_r \bar{h} + b_r)$, where $W_r \in \mathcal{R}^{z \times d}$ and $b_r \in \mathcal{R}^z$ are variables, $\hat{p} \in \mathcal{R}^z$ is the estimated probability for each class and z is the number of relations. A cost function for one instance is the negative log-likelihood of the relations,

$$J_{truth}(\hat{p}, y, \theta) = -\frac{1}{z} \sum_{j=1}^{z} y_j \log \hat{p}_j, \tag{2}$$

where $y \in \mathcal{R}^z$ is the one-hot represented ground truth and θ presents all the parameters.

3.3 Semi-distant Supervision

The architecture of our semi-distant supervision is shown in Figure 3. To sufficiently utilize the reconstructing dataset including accurately labeled instances and unlabeled ones, we propose a generative adversarial training strategy, which transforms unlabeled instances (x_{ul}) to labeled data (x_l) space by generating valid relation representations (x_{gen}) and making the distribution of labeled instances $p(x_l)$ equal to that of generative data $p(x_{gen})$ in relation space[2]. Inspired by Goodfellow et al. (2014), we further devise a three-player min-max game to generate valid data distribution $p(x_{gen})$ with sentence encoder, generative and discriminative modules. The generative module minimizes the difference of $p(x_l)$ and $p(x_{gen})$, and the discriminative module maximizes the probability of distinguishing from each other at the same

---

[2] $p(x_l)$ and $p(x_{gen})$ represents the data distribution of labeled and generative instances.

Algorithm 1 GAN driven Semi-Distant Supervision algorithm

```
Require: discriminator $D$, generator $G$, sentence encoder $S$, $s_i$, $s_j$ and $s_k$ are hyper-parameters to
    indicate iterator number of each module
Initialize the parameters of $D$, $G$, $S$ with random weights $\theta_d$, $\theta_g$ and $\theta_s$
for numbers of training iterations do
    for $s_i$ steps do
        Sample mini-batch of $n$ samples from accurate instances set presented as $x$
        Sample mini-batch of $m$ samples from unlabeled instances set presented as $c$
        Fix $G$ and $S$, update $D$ by ascending its stochastic gradient:
            $\nabla_{\theta_d}[\frac{1}{n}\sum_{u=1}^{n}\log D(x_u) + \frac{1}{m}\sum_{v=1}^{m}\log(1 - D(G(c_v)))]$
    end for
    for $s_j$ steps do
        Sample mini-batch of $m$ samples from unlabeled instances set presented as $c$
        Fix $D$ and $S$, update $G$ by descending its stochastic gradient:
            $\nabla_{\theta_g}\frac{1}{m}\sum_{v=1}^{m}\log(1 - D(G(c_v)))$
    end for
    for $s_k$ steps do
        Sample mini-batch of $n$ samples from accurate instances set presented as $x$
        Sample mini-batch of $m$ samples from unlabeled instances set presented as $c$
        Fix $D$ and $G$, update $S$ by descending its stochastic gradient:
            $\nabla_{\theta_s}[-\frac{1}{nz}\sum_{u=1}^{n}\sum_{j=1}^{z} y_{l_j}\log\hat{p}(y_j|x_u) - \frac{1}{mz}\sum_{v=1}^{m}\sum_{j=1}^{z} y_{g_j}\log\hat{p}(y_j|G(c_v))]$
    end for
end for
```

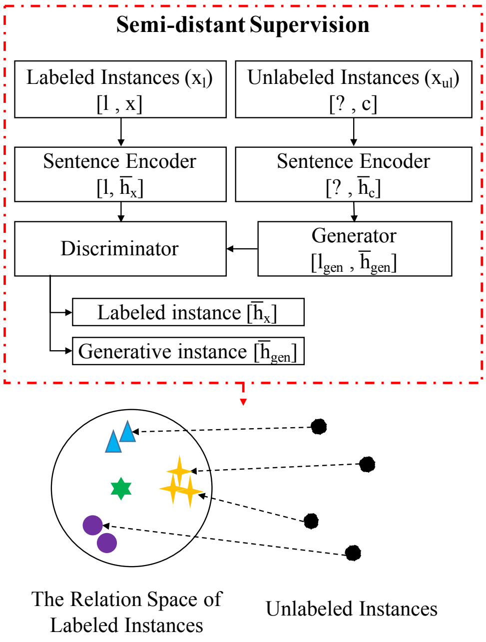


Figure 3: The architecture of GAN driven semi-distant supervision for relation extraction. $\bar{h}_x$ and $\bar{h}_c$ are relation representations of labeled and unlabeled instances respectively. $\bar{h}_{gen}$ is generated relation representation by the generator. l and l_{gen} represent accurate label and generated label respectively. The symbols in relation space represent labeled instances.

time. Sentence encoder is proposed as the third player, which extracts relation features from all the instances and produces a pre-trained relation representation for unlabeled instances. With the sentence encoder, we can control relation features contained in the generated representations.

Therefore, the discriminative module D will try to distinguish labeled data from generative data, while the generative module G makes $p(x_{gen}) \approx p(x_l)$. In addition, the sentence encoder S extracts relation features with all the training instances p_{all}. The training procedure is a three-player min-max game as the following equation,

$$\begin{aligned}\min_{S,G}\max_{D} V(S,D,G) = {} & E_{x\sim p_{all}}[\log S(x)] \\ & + E_{x\sim p_{x_l}}[\log D(x)] \\ & + E_{c\sim p_{x_{gen}}}[\log(1 - D(G(c)))],\end{aligned} \tag{3}$$

In generative adversarial training, the discriminative module is trained by maximizing the gap between labeled data and generative data with the following equation,

$$J_D(x, c, \theta_d) = \log D(x) + \log(1 - D(G(c))), \tag{4}$$

where x and c are instances from accurately labeled set (x_l) and unlabeled set (x_{ul}) respectively.

θ_d presents parameters for the discriminator. $D(x)$ and $D(G(c))$ are defined as follows,

$$D(x) = \sigma(W_d \bar{h}_x), \tag{5}$$

$$D(G(c)) = \sigma(W_d(\bar{h}_c + W_g)), \tag{6}$$

where W_d and W_g are variables for discriminative and generative modules respectively. σ is the sigmoid function. The generative module is trained to make the generated relation representations more similar to real ones by the following loss function, where θ_g presents parameters.

$$J_G(c, \theta_g) = \log(1 - D(G(c))) \tag{7}$$

Finally, we train our sentence encoder S by optimizing the following loss function,

$$\begin{aligned} J_S(x, c, \theta_s) = &-\frac{1}{z}\sum_{j=1}^{z} y_{l_j} \log \hat{p}(y_j|x) \\ &-\frac{1}{z}\sum_{j=1}^{z} y_{g_j} \log \hat{p}(y_j|G(c)), \end{aligned} \tag{8}$$

where y_g means a one-hot vector which labels the most possible relation for unlabeled instances generated by the sentence encoder. θ_s represents parameters of S. The complete training procedure for generative adversarial training is shown as Algorithm 1.

4 Experiments

The experiments are proposed to answer the following three questions, 1) Is the proposed semi-distant supervision method effective for the task of relation extraction? 2) Is the constructed dataset credible enough? 3) Is the generative adversarial training helpful to relation extraction and other semi-supervised tasks?

4.1 Experimental Settings

4.1.1 Dataset

We conduct experiments on a widely used dataset NYT (Riedel et al., 2010) and its new version Accurate-NYT (A-NYT). A-NYT is a credible dataset filtered by our data construction module. We follow the previous work (Lin et al., 2016) to partition training and testing sets for NYT and A-NYT. Besides, we apply sixteen real-world datasets[3] (Liu et al., 2017a) to further verify the effectiveness of our generative adversarial training strategy on the task of sentiment classification. The dataset details are shown in Table 2.

Dataset	Positive	Negative	Unlabeled	Classes
NYT	163,108	579,428	-	53
A-NYT	163,108	240,453	338,975	53
Books	1,000	1,000	2,000	2
Electronics	1,000	998	2,000	2
DVD	1,000	1,000	2,000	2
Kitchen	1,000	1,000	2,000	2
Apparel	1,000	1,000	2,000	2
Camera	999	998	2,000	2
Health	1,000	1,000	2,000	2
Music	1,000	1,000	2,000	2
Toys	1,000	1,000	2,000	2
Video	1,000	1,000	2,000	2
Baby	1,000	900	2,000	2
Magazine	1,000	970	2,000	2
Software	1,000	915	475	2
Sports	1,000	1,000	2,000	2
IMDB	994	1,006	2,000	2
MR	986	1,014	2,000	2

Table 2: Statistics of the datasets

4.1.2 Evaluation Metrics and Baselines

On the dataset NYT and A-NYT, we evaluate our method in the classical held-out evaluation. It evaluates our models by comparing relation facts discovered from the test sentences with those in Freebase. Specifically, we report both the aggregate Precision-Recall (PR) curves and Precision at top N predictions (P@N) in our experiments. For the other datasets, we compute the precision of all the predictions.

We adopt the following baselines for distant supervised relation extraction.

Zeng et al. (2015) extracted relation features with piecewise convolutional neural network (PCNN).

Lin et al. (2016) integrated PCNN with selective attention mechanism (PCNN+ATT).

Wu et al. (2017) added adversarial noise at the level of the word embeddings (PCNN+ATT+AT).

Liu et al. (2017b) relabeled the training instances dynamically by the relation extractor (PCNN+ATT+SL).

Qin et al. (2018a) designed a GAN to recognize true positive samples (PCNN+DSGAN).

Liu et al. (2018) shortened the training instances with the parser tree and pre-trained word embeddings with transfer learning, which is the latest state-of-the-art work.

Self-Training (ST) is a semi-supervised method that can be integrated with PCNN+ATT for unlabeled data, which generates relation types for unlabeled instances with the model itself.

---

[3] The datasets are Amazon product reviews and movie reviews.

4.1.3 Parameters

In our experiments, we use the $word2vec$ in the setting of Skip-gram to train the word embeddings on NYT set. To train our model efficiently, we iterate by randomly selecting a batch from the training set until convergence and apply sentence-level attention mechanism following the previous work (Lin et al., 2016). The parameter n and m are batch sizes for accurate and unlabeled datasets respectively. We update the gradient with adaptive moment estimation (Kingma and Ba, 2015). Furthermore, $L2$ regularization and dropout (Srivastava et al., 2014) are adopted to avoid overfitting. Finally, we use a grid search and cross-validation to determine the optional parameters as shown in Table 3. The hyper-parameters s_i, s_j and s_k are training steps for different modules of generative adversarial training. Since the other parameters have little effect on the results, we follow the settings as the previous work (Lin et al., 2016).

batch size (n, m)	50
s_i, s_j, s_k	2, 1, 2
filter number d	230
kernel size k	3
word dimension w	50
position dimension p	10
learning rate	0.001
dropout probability	0.5
L2 regularization strength	0.0001

Table 3: Parameter settings

4.2 Overall Performance of Semi-Distant Supervision

The overall performance of our method compared with baselines for distant supervised relation extraction is shown in Table 4. We can see that our semi-distant supervised method achieves much better results than the baselines on all metrics. The huge improvement comes from both the accurate dataset and the effective training strategy which leverages unlabeled instances properly.

4.3 Effect of Dataset A-NYT

In this section, we apply two previous methods Zeng et al. (2015) and Lin et al. (2016) on NYT and A-NYT. PR curves for NYT are reported in their papers, while PR curves for A-NYT come from our implementations of the two baselines. NYT and A-NYT share the same positive instances, while A-NYT set has less and credible negative instances. As shown in Figure 4(a), methods on A-NYT always obtain better performance. The huge gap between PR curves is caused by false negative instances in NYT, which are not used for training and testing in A-NYT. To prove that results on A-NYT are according to the actual situation, we do manual evaluations at P@Ns. As shown in Table 5, the huge bias caused by false negative instances on NYT is dramatically alleviated on the dataset A-NYT.

P@N	100	200	300	Mean	PR
Zeng et al. (2015)	72.3	69.7	64.1	68.7	0.33
Lin et al. (2016)	76.2	73.1	67.4	72.2	0.35
Wu et al. (2017)	81.0	74.5	71.7	75.7	0.34
Liu et al. (2017b)	87.0	84.5	77.0	82.8	0.34
Qin et al. (2018a)	78.0	75.5	72.3	75.3	0.35
Liu et al. (2018)	87.0	83.0	78.0	82.7	0.39
Our Method	**96.0**	**93.5**	**93.0**	**94.2**	**0.56**

Table 4: Overall performance at P@Ns(%) and PR curve areas

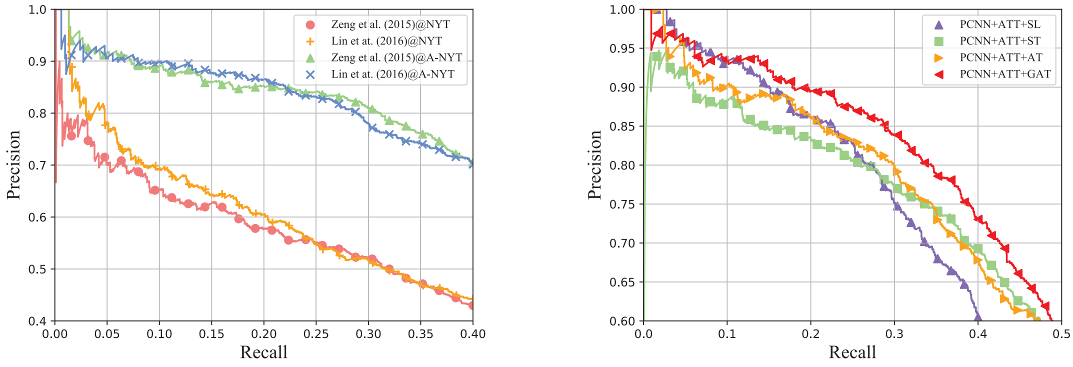
Evaluations	P@100	P@200	P@300
Automatic@NYT	76.2	73.1	67.4
Manual@NYT	96.0(+19.8)	95.5(+22.4)	91.0(+23.6)
Automatic@A-NYT	93.0	89.5	88.0
Manual@A-NYT	96.0(+3.0)	92.5(+3.0)	90.7(+2.7)

Table 5: P@Ns(%) of Lin et al. (2016) upon automatic and manual evaluations

4.4 Effect of Generative Adversarial Training for Relation Extraction

To further demonstrate the effectiveness of our training strategies, we compare Generative Adversarial Training (GAT) with other baselines on the partially labeled dataset A-NYT as shown in Figure 4(b). The figure gives the following insights, 1) PCNN+ATT+ST and PCNN+ATT+AT do not work well, which is caused by the low quality of unlabeled instances. 2) PCNN+ATT+SL works as well as our models at low recall rate because of its excellent ability to extract notable features. Unfortunately, it falls far behind all the baselines at high recall rate, which means it tends to converge to a local optimum. 3) Our model achieves solid PR curves at all range of recall rate.

Meanwhile, we propose a detailed comparison of baselines with P@Ns and PR curve areas as shown in Table 6. From the table, we can see that our training strategy achieves much better result than the other baselines, which indicates that abundant unlabeled instances are helpful to extract relations only if used appropriately. Going



(a) PR curves of baselines on NYT and A-NYT [4]

(b) PR curves of our model and baselines on A-NYT with all the labeled and unlabeled data

Figure 4: PR curves for the comparisons (Better view in color)

deeper in the table, PCNN+ATT+SL works well at top predictions but obtains the worst PR curve. Our semi-distant supervised model with adversarial generations is useful for leveraging unlabeled instances properly.

P@N	100	200	300	Mean	PR
Zeng et al. (2015)‡	91.0	88.5	87.6	89.0	0.513
Lin et al. (2016)‡	93.0	89.5	88.0	88.2	0.513
Qin et al. (2018a)‡	90.0	91.0	88.3	89.8	0.524
Liu et al. (2018)‡	93.0	93.5	91.3	92.6	0.503
PCNN+ATT+SL	**96.0**	93.0	90.6	93.2	0.466
PCNN+ATT+ST	92.0	88.0	85.3	88.4	0.519
PCNN+ATT+AT	95.0	92.0	88.6	91.9	0.526
PCNN+ATT+GAT	**96.0**	**93.5**	**93.0**	**94.2**	**0.558**

Table 6: P@Ns(%) and PR curve areas on A-NYT. Methods with ‡ do not use unlabeled data. PCNN+ATT+AT is a semi-supervised extension of original method (Wu et al. (2017)) to make the unlabeled instances consistent with their predictions.

4.5 Effect of Generative Adversarial Training for Sentiment Classification

To verify the expandability of generative adversarial training, we conduct additional experiments on the task of sentiment classification. We implement our model and three baselines based on the Long Short-Term Memory (LSTM) network[5]. The results are shown in Table 7, from which we see that, 1) Self-training obtains poor results compared with the basic LSTM model, which means they fail to utilize unlabeled data correctly. 2) Adversarial training improves the performance on only three of the datasets and performs poorly on others, which means they possibly rely on the quality of unlabeled data. 3) LSTM+GAT achieves better results than the baselines on most of the datasets because of generating high-quality representations for unlabeled sentences.

Dataset	LSTM	LSTM+ST	LSTM+AT	**LSTM+GAT**
Books	79.5	75.8	**80.5**	80.3
Elec.	80.5	77.5	**84.1**	81.5
DVD	81.7	75.8	78.6	**82.0**
Kitchen	78.0	79.3	**81.7**	81.3
Apparel	83.2	83.5	84.8	**85.2**
Camera	85.2	84.3	86.1	**86.8**
Health	84.5	84.4	81.7	**86.2**
Music	76.7	76.0	76.0	**80.3**
Toys	83.2	79.8	83.7	**84.8**
Video	81.5	79.7	80.4	**83.0**
Baby	84.7	84.3	83.0	**85.3**
Mag.	89.2	85.3	89.0	**89.5**
Soft.	84.7	84.1	83.3	**85.1**
Sports	81.7	79.8	82.3	**82.5**
IMDB	81.7	78.3	82.5	**82.8**
MR	72.7	71.8	72.3	**73.5**
Mean	81.8	80.0	81.9	**83.1**

Table 7: Precision for the sentiment classification task

5 Conclusions

In this paper, we propose a novel semi-distant supervision approach that is capable of jointly exploiting limited accurate and abundant unlabeled ones. We first construct a reliable dataset with a knowledge base and additional entity descriptions.

---

[4] Results on NYT are reported as their papers except Wu et al. (2017) and Qin et al. (2018a). Results of these two methods on NYT and all results on A-NYT are obtained by our implementations.

[5] Our generative adversarial training strategy is model-independent, meaning that it could be applied to other neural models.

With the dataset, the generative adversarial training strategy is proposed to deal with plenty of unlabeled instances, which generates valid relation representations. Our experiments show that the proposed approach achieves significant improvement over previous state-of-the-art baselines.

References

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1556–1567.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 601–610.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2216–2225.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y Chang. 2014. Distant supervision for relation extraction with matrix completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 839–849.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pages 5779–5786.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 2672–2680.

Xianpei Han and Le Sun. 2016. Global distant supervision for relation extraction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2950–2956.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2236–2245.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 541–550.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 3060–3066.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Processings of the third International Conference for Learning Representations (ICLR)*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 34–43.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2124–2133.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–10.

Tianyi Liu, Xinsong Zhang, Wanhao Zhou, and Weijia Jia. 2018. Neural relation extraction via inner-sentence noise reduction and transfer learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2195–2204.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017b. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1790–1795.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 1003–1011.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 2204–2212.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018a. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 496–505.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2137–2147.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 148–163.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, pages 1929–1958.

Sen Su, Ningning Jia, Xiang Cheng, Shuguang Zhu, and Ruiping Li. 2018. Exploring encoder-decoder model for distant supervised relation extraction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4389–4395.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 455–465.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 721–729.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1257–1266.

Guanying Wang, Wen Zhang, Ruoxu Wang, Yalin Zhou, Xi Chen, Wei Zhang, Hai Zhu, and Huajun Chen. 2018. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2246–2255.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1778–1783.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1753–1762.

Xinsong Zhang, Pengshuai Li, Weijia Jia, and Hai Zhao. 2018. Multi-labeled relation extraction with attentive capsule network. *arXiv preprint arXiv:1811.04354*.

A General Framework for Information Extraction using Dynamic Span Graphs

Yi Luan[†] **Dave Wadden**[†] **Luheng He**[‡] **Amy Shah**[†]
Mari Ostendorf[†] **Hannaneh Hajishirzi**[†*]
[†]University of Washington
[*]Allen Institute for Artificial Intelligence
[‡]Google AI Language
{luanyi, dwadden, amyshah, ostendor, hannaneh}@uw.edu
luheng@google.com

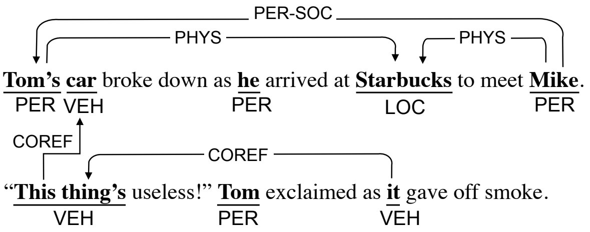
Abstract

We introduce a general framework for several information extraction tasks that share span representations using dynamically constructed span graphs. The graphs are constructed by selecting the most confident entity spans and linking these nodes with confidence-weighted relation types and coreferences. The dynamic span graph allows coreference and relation type confidences to propagate through the graph to iteratively refine the span representations. This is unlike previous multi-task frameworks for information extraction in which the only interaction between tasks is in the shared first-layer LSTM. Our framework significantly outperforms the state-of-the-art on multiple information extraction tasks across multiple datasets reflecting different domains. We further observe that the span enumeration approach is good at detecting nested span entities, with significant F1 score improvement on the ACE dataset.[1]

1 Introduction

Most Information Extraction (IE) tasks require identifying and categorizing phrase spans, some of which might be nested. For example, entity recognition involves assigning an entity label to a phrase span. Relation Extraction (RE) involves assigning a relation type between pairs of spans. Coreference resolution groups spans referring to the same entity into one cluster. Thus, we might expect that knowledge learned from one task might benefit another.

Most previous work in IE (e.g., (Nadeau and Sekine, 2007; Chan and Roth, 2011)) employs a pipeline approach, first detecting entities and then using the detected entity spans for relation extraction and coreference resolution. To avoid cascading errors introduced by pipeline-style systems, recent work has focused on coupling different IE tasks as in joint modeling of entities and relations (Miwa and Bansal, 2016; Zhang et al., 2017), entities and coreferences (Hajishirzi et al., 2013; Durrett and Klein, 2014), joint inference (Singh et al., 2013) or multi-task (entity/relation/coreference) learning (Luan et al., 2018a). These models mostly rely on the first layer LSTM to share span representations between different tasks and are usually designed for specific domains.


Tom's **car** broke down as **he** arrived at **Starbucks** to meet **Mike**.

"**This thing's** useless!" **Tom** exclaimed as **it** gave off smoke.

Figure 1: A text passage illustrating interactions between entities, relations and coreference links. Some relation and coreference links are omitted.

In this paper, we introduce a general framework Dynamic Graph IE (DyGIE) for coupling multiple information extraction tasks through shared span representations which are refined leveraging contextualized information from relations and coreferences. Our framework is effective in several domains, demonstrating a benefit from incorporating broader context learned from relation and coreference annotations.

Figure 1 shows an example illustrating the potential benefits of entity, relation, and coreference contexts. It is impossible to predict the entity labels for *This thing* and *it* from within-sentence context alone. However, the antecedent *car* strongly suggests that these two entities have a VEH type. Similarly, the fact that *Tom* is located at *Starbucks* and *Mike* has a relation to *Tom* provides support for

---

[1] Code and pre-trained models are publicly available at `https://github.com/luanyi/DyGIE`.

Proceedings of NAACL-HLT 2019, pages 3036–3046
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

the fact that *Mike* is located at *Starbucks*.

DYGIE uses multi-task learning to identify entities, relations, and coreferences through shared span representations using dynamically constructed span graphs. The nodes in the graph are dynamically selected from a beam of highly-confident mentions, and the edges are weighted according to the confidence scores of relation types or coreferences. Unlike the multi-task method that only shares span representations from the local context (Luan et al., 2018a), our framework leverages rich contextual span representations by propagating information through coreference and relation links. Unlike previous BIO-based entity recognition systems (Collobert and Weston, 2008; Lample et al., 2016; Ma and Hovy, 2016) that assign a text span to at most one entity, our framework enumerates and represents all possible spans to recognize arbitrarily overlapping entities.

We evaluate DYGIE on several datasets spanning many domains (including news, scientific articles, and wet lab experimental protocols), achieving state-of-the-art performance across all tasks and domains and demonstrating the value of coupling related tasks to learn richer span representations. For example, DYGIE achieves relative improvements of 5.7% and 9.9% over state of the art on the ACE05 entity and relation extraction tasks, and an 11.3% relative improvement on the ACE05 overlapping entity extraction task.

The contributions of this paper are threefold. 1) We introduce the dynamic span graph framework as a method to propagate global contextual information, making the code publicly available. 2) We demonstrate that our framework significantly outperforms the state-of-the-art on joint entity and relation detection tasks across four datasets: ACE 2004, ACE 2005, SciERC and the Wet Lab Protocol Corpus. 3) We further show that our approach excels at detecting entities with overlapping spans, achieving an improvement of up to 8 F1 points on three benchmarks annotated with overlapped spans: ACE 2004, ACE 2005 and GENIA.

2 Related Work

Previous studies have explored joint modeling (Miwa and Bansal, 2016; Zhang et al., 2017; Singh et al., 2013; Yang and Mitchell, 2016)) and multi-task learning (Peng and Dredze, 2015; Peng et al., 2017; Luan et al., 2018a, 2017a) as methods to share representational strength across related information extraction tasks. The most similar to ours is the work in Luan et al. (2018a) that takes a multi-task learning approach to entity, relation, and coreference extraction. In this model, the different tasks share span representations that only incorporate broader context indirectly via the gradients passed back to the LSTM layer. In contrast, DYGIE uses dynamic graph propagation to explicitly incorporate rich contextual information into the span representations.

Entity recognition has commonly been cast as a sequence labeling problem, and has benefited substantially from the use of neural architectures (Collobert et al., 2011; Lample et al., 2016; Ma and Hovy, 2016; Luan et al., 2017b, 2018b). However, most systems based on sequence labeling suffer from an inability to extract entities with overlapping spans. Recently Katiyar and Cardie (2018) and Wang and Lu (2018) have presented methods enabling neural models to extract overlapping entities, applying hypergraph-based representations on top of sequence labeling systems. Our framework offers an alternative approach, forgoing sequence labeling entirely and simply considering all possible spans as candidate entities.

Neural graph-based models have achieved significant improvements over traditional feature-based approaches on several graph modeling tasks. Knowledge graph completion (Yang et al., 2015; Bordes et al., 2013) is one prominent example. For relation extraction tasks, graphs have been used primarily as a means to incorporate pipelined features such as syntactic or discourse relations (Peng et al., 2017; Song et al., 2018; Zhang et al., 2018). Christopoulou et al. (2018) models all possible paths between entities as a graph, and refines pair-wise embeddings by performing a walk on the graph structure. All these previous works assume that the nodes of the graph (i.e. the entity candidates to be considered during relation extraction) are predefined and fixed throughout the learning process. On the other hand, our framework does not require a fixed set of entity boundaries as an input for graph construction. Motivated by state-of-the-art span-based approaches to coreference resolution (Lee et al., 2017, 2018) and semantic role labeling (He et al., 2018), the model uses a beam pruning strategy to dynamically select high-quality spans, and constructs a graph using the selected spans as nodes.

Many state-of-the-art RE models rely upon domain-specific external syntactic tools to construct dependency paths between the entities in a sentence (Li and Ji, 2014; Xu et al., 2015; Miwa and Bansal, 2016; Zhang et al., 2017). These systems suffer from cascading errors from these tools and are hard to generalize to different domains. To make the model more general, we combine the multitask learning framework with ELMo embeddings (Peters et al., 2018) without relying on external syntactic tools and risking the cascading errors that accompany them, and improve the interaction between tasks through dynamic graph propagation. While the performance of DyGIE benefits from ELMo, it advances over some systems (Luan et al., 2018a; Sanh et al., 2019) that also incorporate ELMo. The analyses presented here give insights into the benefits of joint modeling.

3 Model

Problem Definition The input is a document represented as a sequence of words D, from which we derive $S = \{s_1, \ldots, s_T\}$, the set of all possible within-sentence word sequence spans (up to length L) in the document. The output contains three structures: the entity types E for all spans S, the relations R for all span pairs $S \times S$ within the same sentence, and the coreference links C for all spans in S across sentences. We consider two primary tasks. First, *Entity Recognition* is the task of predicting the best entity type labels e_i for each span s_i. Second, *Relation Extraction* involves predicting the best relation type r_{ij} for all span pairs (s_i, s_j). We provide additional supervision by also training our model to perform a third, auxiliary task: *Coreference resolution*. For this task we predict the best antecedent c_i for each span s_i.

Our Model We develop a general information extraction framework (DyGIE) to identify and classify entities, relations, and coreference in a multi-task setup. DyGIE first enumerates all text spans in each sentence, and computes a locally-contextualized vector space representation of each span. The model then employs a *dynamic span graph* to incorporate global information into its span representations, as follows. At each training step, the model identifies the text spans that are most likely to represent entities, and treats these spans as nodes in a graph structure. It constructs confidence-weighted arcs for each node according to its predicted coreference and relation links with the other nodes in the graph. Then, the span representations are refined using broader context from gated updates propagated from neighboring relation types and co-referred entities. These refined span representations are used in a multi-task framework to predict entity types, relation types, and coreference links.

3.1 Model Architecture

In this section, we give an overview of the main components and layers of the DyGIE framework, as illustrated in Figure 2. Details of the graph construction and refinement process will be presented in the next section.

Token Representation Layer We apply a bidirectional LSTM over the input tokens. The input for each token is a concatenation of the character reprensetation, GLoVe (Pennington et al., 2014) word embeddings, and ELMo embeddings (Peters et al., 2018). The output token representations are obtained by stacking the forward and backward LSTM hidden states.

Span Representation Layer For each span s_i, its initial vector representation $\mathbf{g}_i^0$ is obtained by concatenating BiLSTM outputs at the left and right end points of s_i, an attention-based soft "head-word," and an embedded span width feature, following Lee et al. (2017).

Coreference Propagation Layer The propagation process starts from the span representations $\mathbf{g}_i^0$. At each iteration t, we first compute an *update vector* $\mathbf{u}_C^t$ for each span s_i. Then we use $\mathbf{u}_C^t$ to update the current representation $\mathbf{g}_i^t$, producing the next span representation $\mathbf{g}_i^{t+1}$. By repeating this process N times, the final span representations $\mathbf{g}_i^N$ share contextual information across spans that are likely to be antecedents in the coreference graph, similar to the process in (Lee et al., 2018).

Relation Propagation Layer The outputs $\mathbf{g}_i^N$ from the coreference propagation layer are passed as inputs to the relation propagation layer. Similar to the coreference propagation process, at each iteration t, we first compute the update vectors $\mathbf{u}_R^t$ for each span s_i, then use it to compute $\mathbf{g}_i^{t+1}$. Information can be integrated from multiple relation paths by repeating this process M times.

Final Prediction Layer We use the outputs of the relation graph layer $\mathbf{g}_i^{N+M}$ to predict the entity labels E and relation labels R. For entities, we pass $\mathbf{g}_i^{N+M}$ to a feed-forward network (FFNN) to

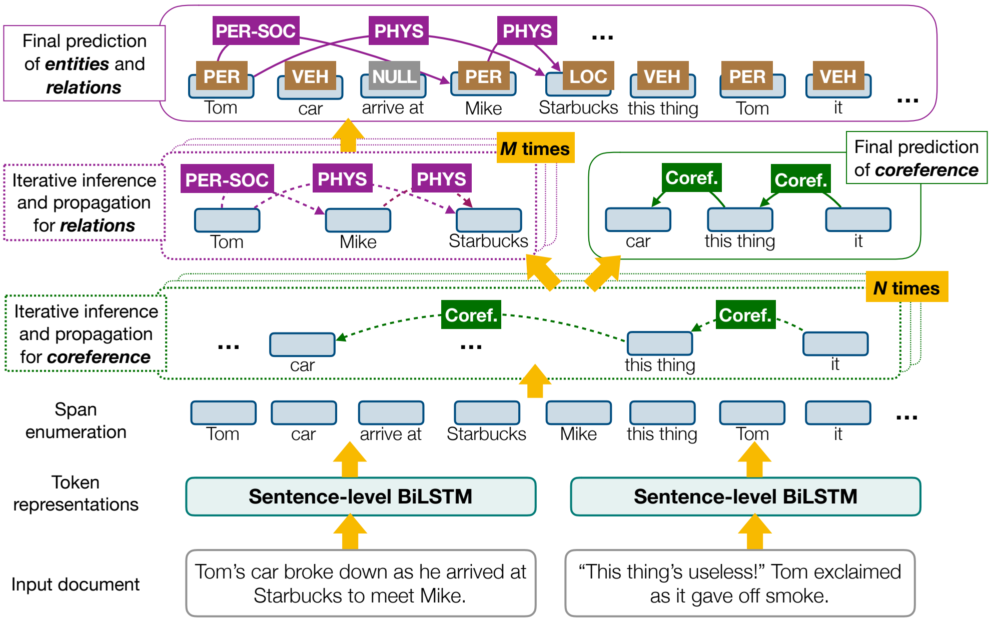


Figure 2: Overview of our DyGIE model. Dotted arcs indicate confidence weighted graph edges. Solid lines indicate the final predictions.

produce per-class scores $\mathbf{P}_E(i)$ for span s_i. For relations, we pass the concatenation of $\mathbf{g}_i^{N+M}$ and $\mathbf{g}_j^{N+M}$ to a FFNN to produce per-class relation scores $\mathbf{P}_R(i,j)$ between spans s_i and s_j. Entity and relation scores are normalized across the label space, similar to Luan et al. (2018a). For coreference, the scores between span pairs (s_i, s_j) are computed from the coreference graph layer outputs $(\mathbf{g}_i^N, \mathbf{g}_j^N)$, and then normalized across all possible antecedents, similar to Lee et al. (2018).

3.2 Dynamic Graph Construction and Span Refinement

The dynamic span graph facilitates propagating broader contexts through soft coreference and relation links to refine span representations. The nodes in the graph are spans s_i with vector representations $\mathbf{g}_i^t \in \mathbb{R}^d$ for the t-th iteration. The edges are weighted by the coreference and relation scores, which are trained according to the neural architecture explained in Section 3.1. In this section, we explain how coreference and relation links can update span representations.

Coreference Propagation Similar to (Luan et al., 2018a), we define a beam B_C consisting of b_c spans that are most likely to be in a coreference chain. We consider $\mathbf{P}_C^t$ to be a matrix of real values that indicate coreference confidence scores between these spans at the t-th iteration. $\mathbf{P}_C^t$ is of size $b_c \times K$, where K is the maximum number of antecedents considered. For the coreference graph, an edge in the graph is single directional, connecting the current span s_i with all its potential antecedents s_j in the coreference beam, where $j < i$. The edge between s_i and s_j is weighted by coreference confidence score at the current iteration $P_C^t(i,j)$. The span update vector $\mathbf{u}_C^t(i) \in \mathbb{R}^d$ is computed by aggregating the neighboring span representations $\mathbf{g}_j^t$, weighted by their coreference scores $P_C^t(i,j)$:

$$\mathbf{u}_C^t(i) = \sum_{j \in B_C(i)} P_C^t(i,j)\mathbf{g}_j^t \qquad (1)$$

where $B_C(i)$ is the set of K spans that are antecedents of s_i,

$$P_C^t(i,j) = \frac{\exp(V_C^t(i,j))}{\sum_{j' \in B_C(i)} \exp(V_C^t(i,j))} \qquad (2)$$

$V_C^t(i,j)$ is a scalar score computed by concatenating the span representations $[\mathbf{g}_i^t, \mathbf{g}_j^t, \mathbf{g}_i^t \odot \mathbf{g}_j^t]$, where $\odot$ is element-wise multiplication. The concatenated vector is then fed as input to a FFNN, similar to (Lee et al., 2018).

Relation Propagation For each sentence, we define a beam B_R consisting of b_r entity spans that are mostly likely to be involved in a relation. Unlike the coreference graph, the weights of relation edges capture different relation types. Therefore, for the t-th iteration, we use a tensor $\mathbf{V}_R^t \in \mathbb{R}^{b_R \times b_R \times L_R}$ to capture scores of each of the L_R relation types. In other words, each edge in the relation graph connects two entity spans s_i and s_j in the relation beam B_R. $\mathbf{V}_R^t(i,j)$ is a L_R-length vector of relation scores, computed with a FFNN with $[\mathbf{g}_i^t, \mathbf{g}_j^t]$ as the input. The relation update vector $\mathbf{u}_R^t(i) \in \mathbb{R}^d$ is computed by aggregating neighboring span representations on the relation graph:

$$\mathbf{u}_R^t(i) = \sum_{j \in B_\mathrm{R}} f(\mathbf{V}_R^t(i,j))\mathbf{A}_R \odot \mathbf{g}_j^t, \quad (3)$$

where $\mathbf{A}_R \in \mathbb{R}^{L_R \times d}$ is a trainable linear projection matrix, f is a non-linear function to select the most important relations. Because only a small number of entities in the relation beam are actually linked to the target span, propagation among all possible span pairs would introduce too much noise to the new representation. Therefore, we choose f to be the ReLU function to remove the effect of unlikely relations by setting the all negative relation scores to 0. Unlike coreference connections, two spans linked via a relation are not expected to have similar representations, so the matrix $\mathbf{A}_R$ helps to transform the embedding $\mathbf{g}_j^t$ according to each relation type.

Updating Span Representations with Gating To compute the span representations for the next iteration $t \in \{1, \ldots, N+M\}$, we define a gating vector $\mathbf{f}_x^t(i) \in \mathbb{R}^d$, where $x \in \{C, R\}$, to determine whether to keep the previous span representation $\mathbf{g}_i^t$ or to integrate new information from the coreference or relation update vectors $\mathbf{u}_x^t(i)$. Formally,

$$\begin{aligned} \mathbf{f}_x^t(i) &= g(\mathbf{W}_x^\mathrm{f}[\mathbf{g}_i^t, \mathbf{u}_x^t(i)]) & (4) \\ \mathbf{g}_i^{t+1} &= \mathbf{f}_x^t(i) \odot \mathbf{g}_i^t + (1 - \mathbf{f}_x^t(i)) \odot \mathbf{u}_x^t(i), \end{aligned}$$

where $\mathbf{W}_x^\mathrm{f} \in \mathbb{R}^{d \times 2d}$ are trainable parameters, and g is an element-wise sigmoid function.

3.3 Training

The loss function is defined as a weighted sum of the log-likelihood of all three tasks:

$$\sum_{(D,R^*,E^*,C^*) \in \mathcal{D}} \Big\{ \lambda_\mathrm{E} \log P(E^* \mid C, R, D) \quad (5)$$
$$+ \lambda_\mathrm{R} \log P(R^* \mid C, D) + \lambda_\mathrm{C} \log P(C^* \mid D) \Big\}$$

where E^*, R^* and C^* are gold structures of the entity types, relations and coreference, respectively. $\mathcal{D}$ is the collection of all training documents D. The task weights λ_E, λ_R, and λ_C are hyperparameters to control the importance of each task.

	Domain	Docs	Ent	Rel	Coref
ACE04	News	348	7	7	✓
ACE05	News	511	7	6	✗
SciERC	AI	500	6	7	✓
WLP	Bio lab	622	18	13	✗

Table 1: Datasets for joint entity and relation extraction and their statistics. *Ent*: Number of entity categories. *Rel*: Number of relation categories.

We use a 1 layer BiLSTM with 200-dimensional hidden layers. All the feed-forward functions have 2 hidden layers of 150 dimensions each. We use 0.4 variational dropout (Gal and Ghahramani, 2016) for the LSTMs, 0.4 dropout for the FFNNs, and 0.5 dropout for the input embeddings. The hidden layer dimensions and dropout rates are chosen based on the development set performance in multiple domains. The task weights, learning rate, maximum span length, number of propagation iterations and beam size are tuned specifically for each dataset using development data.

4 Experiments

DyGIE is a general IE framework that can be applied to multiple tasks. We evaluate the performance of DyGIE against models from two lines of work: combined entity and relation extraction, and overlapping entity extraction.

4.1 Entity and relation extraction

For the entity and relation extraction task, we test the performance of DyGIE on four different datasets: ACE2004, ACE2005, SciERC and the Wet Lab Protocol Corpus. We include the relation graph propagation layer in our models for all datasets. We include the coreference graph propagation layer on the data sets that have coreference annotations available.

Data All four data sets are annotated with entity and relation labels. Only a small fraction of entities ($< 3\%$ of total) in these data sets have a text span that overlaps the span of another entity. Statistics on all four data sets are displayed in Table 1.

The **ACE2004** and **ACE2005** corpora provide entity and relation labels for a collection of documents from a variety of domains, such as newswire and online forums. We use the same entity and relation types, data splits, and preprocessing as Miwa and Bansal (2016) and Li and Ji (2014). Following the convention established in this line of work, an entity prediction is considered correct

Dataset	System	Entity	Relation
ACE04	Bekoulis et al. (2018)	81.6	47.5
	Miwa and Bansal (2016)	81.8	48.4
	DyGIE	**87.4**	**59.7**
ACE05	Miwa and Bansal (2016)	83.4	55.6
	Zhang et al. (2017)	83.6	57.5
	Sanh et al. (2019)	87.5	62.7
	DyGIE	**88.4**	**63.2**
SciERC	Luan et al. (2018a)	64.2	39.3
	DyGIE	**65.2**	**41.6**
WLPC	Kulkarni et al. (2018)	78.0	*54.9
	DyGIE	**79.5**	**64.1**

Table 2: F1 scores on the joint entity and relation extraction task on each test set, compared against the previous best systems. * indicates relation extraction system that takes gold entity boundary as input.

if its type label and head region match those of a gold entity. We will refer to this version of the ACE2004 and ACE2005 data as ACE04 and ACE05. Since the domain and mention span annotations in the ACE datasets are very similar to those of OntoNotes (Pradhan et al., 2012), and OntoNotes contains significantly more documents with coreference annotations, we use OntoNotes to train the parameters for the auxiliary coreference task. The OntoNotes corpus contains 3493 documents, averaging roughly 450 words in length.

The **SciERC** corpus (Luan et al., 2018a) provides entity, coreference and relation annotations for a collection of documents from 500 AI paper abstracts. The dataset defines scientific term types and relation types specially designed for AI domain knowledge graph construction. An entity prediction is considered correct if its label and span match with a gold entity.

The **Wet Lab Protocol Corpus (WLPC)** provides entity, relation, and event annotations for 622 wet lab protocols (Kulkarni et al., 2018). A wet lab protocol is a series of instructions specifying how to perform a biological experiment. Following the procedure in Kulkarni et al. (2018), we perform entity recognition on the union of entity tags and event trigger tags, and relation extraction on the union of entity-entity relations and entity-trigger event roles. Coreference annotations are not available for this dataset.

Baselines We compare DyGIE with current state of the art methods in different datasets. Miwa and Bansal (2016) provide the current state of the art on ACE04. They construct a Tree LSTM using dependency parse information, and use the representations learned by the tree structure as features for relation classification. Bekoulis et al. (2018) use adversarial training as regularization for a neural model. Zhang et al. (2017) cast joint entity and relation extraction as a table filling problem and build a globally optimized neural model incorporating syntactic representations from a dependency parser. Similar to DyGIE, Sanh et al. (2019) and Luan et al. (2018a) use a multi-task learning framework for extracting entity, relation and coreference labels. Sanh et al. (2019) improved the state of the art on ACE05 using multi-task, hierarchical supervised training with a set of low level tasks at the bottom layers of the model and more complex tasks at the top layers of the model. Luan et al. (2018a) previously achieved the state of the art on SciERC and use a span-based neural model like our DyGIE. Kulkarni et al. (2018) provide a baseline for the WLPC data set. They employ an LSTM-CRF for entity recognition, following Lample et al. (2016). For relation extraction, they assume the presence of gold entities and train a maximum-entropy classifier using features from the labeled entities.

Results Table 2 shows test set F1 on the joint entity and relation extraction task. We observe that DyGIE achieves substantial improvements on both entity recognition and relation extraction across the four data sets and three domains, all in the realistic setting where no "gold" entity labels are supplied at test time. DyGIE achieves 7.1% and 7.0% relative improvements over the state of the art on NER for ACE04 and ACE05, respectively. For the relation extraction task, DyGIE attains 25.8% relative improvement over SOTA on ACE04 and 13.7% relative improvement on ACE05. For ACE05, the best entity extraction performance is obtained by switching the order between `CorefProp` and `RelProp` (`RelProp` first then `CorefProp`).

On SciERC, DyGIE advances the state of the art by 5.9% and 1.9% for relation extraction and NER, respectively. The improvement of DyGIE over the previous SciERC model underscores the ability of coreference and relation propagation to construct rich contextualized representations.

The results from Kulkarni et al. (2018) establish a baseline for IE on the WLPC. In that work, relation extraction is performed using gold entity boundaries as input. Without using any gold entity information, DyGIE improves on the baselines by 16.8% for relation extraction and 2.2% for NER.

	Domain	Docs	Ent	Overlap	Coref
ACE04-O	News	443	7	42%	✓
ACE05-O	News	437	7	32%	✗
GENIA	Biomed	1999	5	24%	✓

Table 3: Datasets for overlapping entity extraction and their statistics. *Ent*: Number of entity categories. *Overlap*: Percentage of sentences that contain overlapping entities.

On the OntoNotes data set used for the auxiliary coreference task with ACE05, our model achieves coreference test set performance of 70.4 F1, which is competitive with the state-of-the-art performance reported in Lee et al. (2017).

4.2 Overlapping Entity Extraction

There are many applications where the correct identification of overlapping entities is crucial for correct document understanding. For instance, in the biomedical domain, a *BRCA1 mutation carrier* could refer to a patient taking part in a clinical trial, while *BRCA1* is the name of a gene.

We evaluate the performance of DyGIE on overlapping entity extraction in three datasets: ACE2004, ACE2005 and GENIA. Since relation annotations are not available for these datasets, we include the coreference propagation layer in our models but not the relation layer.[2]

Data Statistics on our three datasets are listed in Table 3. All three have a substantial number ($>20\%$ of total) of overlapping entities, making them appropriate for this task.

As in the joint case, we evaluate our model on **ACE2004** and **ACE2005**, but here we follow the same data preprocessing and evaluation scheme as Wang and Lu (2018). We refer to these data sets as ACE04-O and ACE05-O. Unlike the joint entity and relation task in Sec. 4.1, where only the entity head span need be predicted, an entity prediction is considered correct in these experiments if both its entity label and its full text span match a gold prediction. This is a more stringent evaluation criterion than the one used in Section 4.1. As before, we use the OntoNotes annotations to train the parameters of the coreference layer.

The **GENIA** corpus (Kim et al., 2003) provides entity tags and coreferences for 1999 abstracts from the biomedical research literature. We only use the IDENT label to extract coreference clusters. We use the same data set split and preprocessing procedure as Wang and Lu (2018) for overlapping entity recognition.

Dataset	System	Entity F1
ACE04-O	Katiyar and Cardie (2018)	72.7
	Wang and Lu (2018)	75.1
	DyGIE	**84.7**
ACE05-O	Katiyar and Cardie (2018)	70.5
	Wang and Lu (2018)	74.5
	DyGIE	**82.9**
GENIA	Katiyar and Cardie (2018)	73.8
	Wang and Lu (2018)	75.1
	DyGIE	**76.2**

Table 4: Performance on the overlapping entity extraction task, compared to previous best systems. We report F1 of extracted entities on the test sets.

	Entity			Relation		
Model	P	R	F1	P	R	F1
DyGIE	87.4	86.7	**87.1**	56.2	60.9	58.4
`−CorefProp`	86.2	85.2	85.7	64.3	56.7	**60.2**
`−RelProp`	87.0	86.7	86.9	60.4	55.8	58.0
`Base`	86.1	85.7	85.9	59.5	55.7	57.6

Table 5: Ablations on the ACE05 development set with different graph propagation setups. `−CorefProp` ablates the coreference propagation layers, while `−RelProp` ablates the relation propagation layers. `Base` is the system without any propagation.

Baselines The current state-of-the-art approach on all three data sets is Wang and Lu (2018), which uses a segmental hypergraph coupled with neural networks for feature learning. Katiyar and Cardie (2018) also propose a hypergraph approach using a recurrent neural network as a feature extractor.

Results Table 4 presents the results of our overlapping entity extraction experiments on the different datsets. DyGIE improves 11.6% on the state of the art for ACE04-O and 11.3% for ACE05-O. DyGIE also advances the state of the art on GENIA, albeit by a more modest 1.5%. Together these results suggest that DyGIE can be utilized fruitfully for information extraction across different domains with overlapped entities, such as bio-medicine.

5 Analysis of Graph Propagation

We use the dev sets of ACE2005 and SciERC to analyze the effect of different model components.

5.1 Coreference and Relation Graph Layers

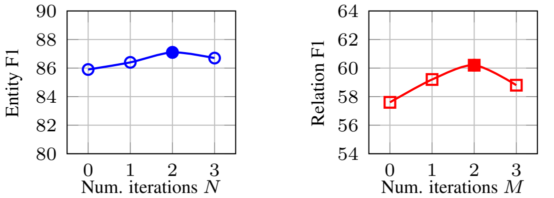
Tables 5 and 6 show the effects of graph propagation on entity and relation prediction accuracy,

---

[2] We use the pre-processed ACE dataset from previous work and relation annotation is not available.

	Entity			Relation		
Model	P	R	F1	P	R	F1
DyGIE	68.6	67.8	**68.2**	46.2	38.5	**42.0**
−`CorefProp`	69.2	66.9	68.0	42.0	40.5	41.2
−`RelProp`	69.1	66.0	67.5	43.6	37.6	40.4
`Base`	70.0	66.3	68.1	45.4	34.9	39.5

Table 6: Ablations on the SciERC development set on different graph progation setups. `CorefProp` has a much smaller effect on entity F1 compared to ACE05.



(a) Entity F1 with different number of `CorefProp` iterations N.

(b) Relation F1 with different number of `RelProp` iterations M.

Figure 3: F1 score of each layer on ACE development set for different number of iterations. $N = 0$ or $M = 0$ indicates no propagation is made for the layer.

where −`CorefProp` and −`RelProp` denote ablating the propagation process by setting $N = 0$ or $M = 0$, respectively. `Base` is the base model without any propagation. For ACE05, we observe that coreference propagation is mainly helpful for entities; it appears to hurt relation extraction. On SciIE, coreference propagation gives a small benefit on both tasks. Relation propagation significantly benefits both entity and relation extraction in both domains. In particular, there are a large portion of sentences with multiple relation instances across different entities in both ACE05 and SciERC, which is the scenario in which we expect relation propagation to help.

Since coreference propagation has more effect on entity extraction and relation propagation has more effect on relation extraction, we mainly focus on ablating the effect of coreference propagation on entity extraction and relation propagation on relation extraction in the following subsections.

5.2 Coreference Propagation and Entities

A major challenge of ACE05 is to disambiguate the entity class for pronominal mentions, which requires reasoning with cross-sentence contexts. For example, in a sentence from ACE05 dataset, "One of **[them]**$_{\text{PER}}$, from a very close friend of **[ours]**$_{\text{ORG}}$." It is impossible to identity whether *them* and *ours* is a person (*PER*) or organization (*ORG*) unless we have read previous sentences. We hypothesize that this is a context where coreference propagation can help. Table 7 shows the effect of the coreference layer for entity categorization of pronouns.[3] DyGIE has 6.6% improvement on pronoun performance, confirming our hypothesis.

Entity Perf. on Pronouns	P	R	F1
DyGIE	79.0	77.1	**78.0**
DyGIE−`CorefProp`	73.8	72.6	73.2

Table 7: Entity extraction performance on pronouns in ACE05. `CorefProp` significantly increases entity extraction F1 on hard-to-disambiguate pronouns by allowing the model to leverage cross-sentence contexts.

Looking further, Table 8 shows the impact on all entity categories, giving the difference between the confusion matrix entries with and without `CorefProp`. The frequent confusions associated with pronouns (*GPE/PER* and *PER/ORG*, where *GPE* is a geopolitical entity) greatly improve, but the benefit of `CorefProp` extends to most categories.

Of course, there are a few instances where `CorefProp` causes errors in entity extraction. For example, in the sentence "[They]$^{\text{ORG}}_{\text{PER}}$ might have been using Northshore...", DyGIE predicted *They* to be of *ORG* type because the most confident antecedent is *those companies* in the previous sentence: "The money was invested in *those companies*." However, *They* is actually referring to *these fund managers* earlier in the document, which belongs to *PER* category.

In the SciERC dataset, the pronouns are uniformly assigned with a *Generic* label, which explains why `CorefProp` does not have much effect on entity extraction performance.

The Figure 3a shows the effect of number of iterations for coreference propagation in the entity extraction task. The figure shows that coreference layer obtains the best performance on the second iteration ($N = 2$).

5.3 Relation Propagation Impact

Figure 4 shows relation scores as a function of number of entities in sentence for DyGIE and DyGIE without relation propagation on ACE05. The figure indicates that relation propagation achieves significant improvement in sentences with more entities, where one might expect that using broader context

---

[3] Pronouns included: `anyone, everyone, it, itself, one, our, ours, their, theirs, them, themselves, they, us, we, who`

	LOC	WEA	GPE	PER	FAC	ORG	VEH
LOC	5	0	-2	-1	2	-1	0
WEA	0	3	0	0	1	-3	-1
GPE	-3	0	31	**-26**	3	-7	0
PER	0	-2	-3	18	-1	**-26**	4
FAC	4	-1	2	-3	2	-5	1
ORG	0	0	0	-8	-1	6	0
VEH	0	-2	-1	2	5	-1	1

Table 8: Difference in the confusion matrix counts for ACE05 entity extraction associated with adding `CorefProp`.

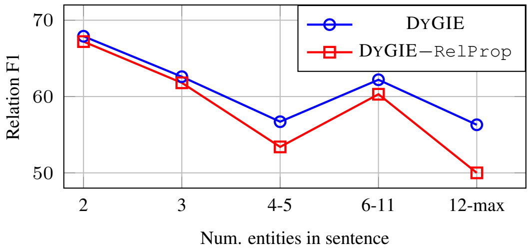


Figure 4: Relation F1 broken down by number of entities in each sentence. The performance of relation extraction degrades on sentences containing more entities. Adding relation propagation alleviates this problem.

could have more impact.

Figure 3b shows the effect of number of iterations for relation propagation in the relation extraction task. Our model achieves the best performance on the second iteration ($M = 2$).

6 Conclusion

We have introduced DyGIE as a general information extraction framework, and have demonstrated that our system achieves state-of-the art results on entity recognition and relation extraction tasks across a diverse range of domains. The key contribution of our model is the dynamic span graph approach, which enhance interaction across tasks that allows the model to learn useful information from broader context. Unlike many IE frameworks, our model does not require any preprocessing using syntactic tools, and has significant improvement across different IE tasks including entity, relation extraction and overlapping entity extraction. The addition of co-reference and relation propagation across sentences adds only a small computation cost to inference; the memory cost is controlled by beam search. These added costs are small relative to those of the baseline span-based model. We welcome the community to test our model on different information extraction tasks. Future directions include extending the framework to encompass more structural IE tasks such as event extraction.

Acknowledgments

This research was supported by the Office of Naval Research under the MURI grant N00014-18-1-2670, NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We also thank the anonymous reviewers and the UW-NLP group for their helpful comments.

References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2830–2836.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2018. A walk-based model on entity graphs for relation extraction. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, volume 2, pages 81–88.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Machine Learning Research*, 12(Aug):2493–2537.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Trans. Assoc. for Computational Linguistics (TACL)*, 2:477–490.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 289–299.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proc. Conf. North American Assoc. for Computational Linguistics (NAACL)*.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1:i180–2.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An annotated corpus for machine reading of instructions in wet lab protocols. In *NAACL-HLT*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. Conf. North American Assoc. for Computational Linguistics (NAACL)*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, volume 1, pages 402–412.

Yi Luan, Chris Brockett, Bill Dolan, Jianfeng Gao, and Michel Galley. 2017a. Multi-task learning for speaker-role adaptation in neural conversation models. In *Proc. IJCNLP*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018a. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2017b. Scientific information extraction with semi-supervised neural tagging. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. 2018b. The uwnlp system at semeval-2018 task 7: Neural relation extraction model with selectively incorporated concept embeddings. In *Proc. Int. Workshop on Semantic Evaluation (SemEval)*, pages 788–792.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 1105–1116.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 548–554.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Trans. Assoc. for Computational Linguistics (TACL)*, 5:101–115.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. *AAAI*.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proc. of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state lstm. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2226–2235.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *EMNLP*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 536–540.

Bishan Yang and Tom M Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1730–1740.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

OpenCeres: When Open Information Extraction Meets the Semi-Structured Web

Colin Lockard
Paul G. Allen School
University of Washington
lockardc@cs.washington.edu

Prashant Shiralkar
Amazon
shiralp@amazon.com

Xin Luna Dong
Amazon
lunadong@amazon.com

Abstract

Open Information Extraction (OpenIE), the problem of harvesting triples from natural language text whose predicate relations are not aligned to any pre-defined ontology, has been a popular subject of research for the last decade. However, this research has largely ignored the vast quantity of facts available in semi-structured webpages. In this paper, we define the problem of OpenIE from semi-structured websites to extract such facts, and present an approach for solving it. We also introduce a labeled evaluation dataset to motivate research in this area. Given a semi-structured website and a set of seed facts for some relations existing on its pages, we employ a semi-supervised label propagation technique to automatically create training data for the relations present on the site. We then use this training data to learn a classifier for relation extraction. Experimental results of this method on our new benchmark dataset obtained a precision of over 70%. A larger scale extraction experiment on 31 websites in the movie vertical resulted in the extraction of over 2 million triples.

1 Introduction

Knowledge extraction is the problem of extracting *(subject, predicate, object)* triples from unstructured or semi-structured data, where the subject and object are entities and the predicate indicates the relationship between them. In conventional information extraction (which we call "ClosedIE"), a closed set of potential predicates and their semantics are pre-defined in an ontology. Open Information Extraction (OpenIE) is an alternative approach that has no pre-defined ontology and instead represents the predicate with a string extracted from the source data. These extractions can capture a much vaster array of semantic relationships than ClosedIE and have been used to support many downstream use-cases, including question-answering, ontology discovery, embedding generation, fact checking, and summarization (Mausam, 2016). Previous OpenIE work has concentrated on raw text, with the aim to extract "open" triples from natural language sentences (Niklaus et al., 2018), with another line of work focused on extracting from webtables (Cafarella et al., 2008).

Semi-structured websites (e.g. IMDb) contain many pages displaying information in stand-alone fields in relatively consistent locations on each page, with entities and the relationship between them indicated via formatting features such as section headers and lists of key-value pairs. Figure 1 shows an example page and the triples it conveys. Semi-structured websites have recently been shown to be a rich target for IE; the Knowledge Vault large-scale web extraction experiment, which extracted from semi-structured websites, natural language text, webtables, and Semantic Web annotations, found that semi-structured websites contributed 75% of total extracted facts and 94% of high-confidence extractions (Dong et al., 2014); the Ceres system showed that one can automatically extract from semi-structured sites with a precision over 90% using distant supervision (Lockard et al., 2018). These works, however, all build on the tradition of semi-structured ClosedIE techniques (Kushmerick et al., 1997; Soderland, 1999; Gulhane et al., 2011; Furche et al., 2014).

Interestingly, we are not aware of any work that applies OpenIE on semi-structured sources, despite the great potential to identify new relationships and new knowledge triples. We investigated 8 movie websites from the Structured Web Data Extraction (SWDE) corpus (Hao et al., 2011) and found that the IMDb ontology can cover only 7% of semantically unique predicates on these sites.

The major challenges that distinguish natural language text and semi-structured data are the basic unit and the inherent structure of the data. In

Proceedings of NAACL-HLT 2019, pages 3047–3056
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

natural language text, each sentence is a unit; it typically consists of a subject, a verb, and an object, which corresponds naturally to the subject, predicate, and object of a knowledge triple. Similarly, in webtables, each table is a unit; its rows, columns, and cells also naturally correspond to subjects, predicates, and objects in triples.

In the semi-structured setting, the basic unit is the webpage, which may contain hundreds or thousands of entity mentions. There is no fixed layout between the subject entity, object entity, and their relation, which may be far apart on the webpage. For example, Figure 1 contains object strings that are below, to the left, and to the right of their corresponding predicates; even trickier, for object string "Uma Thurman", the correct predicate string "Cast" is much farther away than the incorrect one "Crew". Despite the challenges, semi-structured pages do provide inherent visual structure to help distinguish the subject of a page, and *(predicate, object)* pairs for the subject. In this paper we answer the following question: *Given semi-structured webpages in a website, how can we tell which field contains the subject, and which fields contain the (predicate, object) pairs for the subject through any visual or DOM-structured clue?*

This paper makes three contributions. Our first contribution is to formally define a new problem of OpenIE from semi-structured websites (Section 2). We created a benchmark[1] for this problem by enhancing the SWDE corpus (Hao et al., 2011); our benchmark contains a high accuracy and coverage set of ground truth extractions for 21 websites spanning three domains, comprising 855,748 labels across 27,641 pages (Section 4).

Our second contribution is OpenCeres, a solution for OpenIE on semi-structured websites (Section 3). Our solution is novel in three aspects. First, whereas ClosedIE techniques on semi-structured data focus on extracting objects for given predicates, we also identify predicate strings on the website that represent the relations. Second, while ClosedIE techniques can only learn extraction patterns for predicates where there exists seed knowledge, we identify unseen predicates by applying semi-supervised label propagation. Third, whereas most existing extraction techniques on semi-structured sites leverage only DOM patterns as evidence, we use visual aspects of the webpage for distant supervision and label propagation.

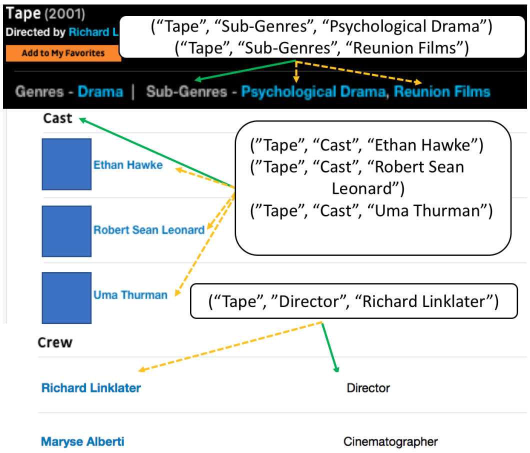


Figure 1: A cropped portion of the detail page from allmovie.com for the film *Tape* with some triples indicated. Solid green and dashed yellow arrows indicate predicate strings and objects respectively.

Our final contribution is a comprehensive evaluation on our new benchmark dataset and online websites (Section 5). Our proposed method obtained an F1 of 0.68, significantly higher than baseline systems, while extracting 7 times as many predicates as were present in the original ontology. In addition, we evaluate on a set of 31 movie websites, yielding 1.17 million extractions at a precision of 0.70. Our results inspire new directions for improvement as discussed in Section 7, and serve as a good baseline for future work.

2 Overview

2.1 Problem Definition

We propose the problem of OpenIE from semi-structured websites. A *semi-structured website* W consists of a set of *detail pages* that each contains information about a particular entity, the *topic entity* of the page. This information is typically populated from an underlying database into an HTML template to create the detail pages. The goal of semi-structured OpenIE is to recover all *(subject, predicate, object)* triples represented in these templatized fields, including the extraction of the string that semantically represents each predicate.

Relation objects are sometimes present without an explicit predicate string defining the relationship; since OpenIE requires extraction of a predicate string, we only consider the case where a meaningful predicate string exists on the page.

Definition 2.1 (Semi-Structured OpenIE) *Let* W *be a semi-structured website, with each page*

---

[1] `https://homes.cs.washington.edu/~lockardc/expanded_swde.html`

w_i *containing facts about a topic entity* t_i. *Semi-Structured OpenIE extracts a set of triples from* W *such that the subject, predicate, and object of each triple is a string value on a page in* W, *with the subject representing* t_i *and the predicate string semantically representing the relation between the subject and object as asserted by the webpage.*

Following the tradition of relation extraction, which considers only binary relationships, we do not consider extraction of compound value types (CVTs) (Freebase, 2018), which express multi-way relationships. In this work, we narrow our focus to Semi-structured OpenIE from a given domain, since we will rely on pre-existing knowledge about that domain to provide us with seed annotations. We leave the extension to the general semi-structured OpenIE problem for future work.

2.2 From ClosedIE to OpenIE

We first summarize the Ceres techniques proposed in (Lockard et al., 2018), which is the state-of-the-art for ClosedIE from semi-structured websites. Ceres learns a model capable of generalizing across variations in a website from training labels automatically generated by the distant supervision technique. The automatic annotation contains two steps. First, *topic annotation* annotates the topic name on the page. Second, *relation annotation* annotates each object field, where the relation is guessed as a relationship in the seed ontology that is valid between the topic and the object.

OpenIE needs to go beyond existing relations in the ontology, identifying relations not existing in seed knowledge. As such, it raises two challenges for the relation annotation step. First, in addition to annotating the objects, we also need to be able to identify the predicate fields in order to extract predicate strings. Second, in addition to annotating the predicates already in the seed knowledge, we also need to identify new predicates on a webpage.

Figure 2 shows the infrastructure of our OpenIE solution, OpenCeres. We propose a relation annotation method that is suitable for OpenIE (shown in the shaded blocks), and inherit other components from Ceres (Lockard et al., 2018). Our key intuition to solve this problem is that different predicates often share some visual features, such as being aligned vertically or horizontally, sharing the same font, size or color, and so on. Thus, if we can identify at least one *(predicate, object)* pair on the page, we can look for other similarly formatted pairs of nodes and assume that they also represent *(predicate, object)* pairs. Accordingly, we propose a three-stage process that combines distant supervision and semi-supervised learning.

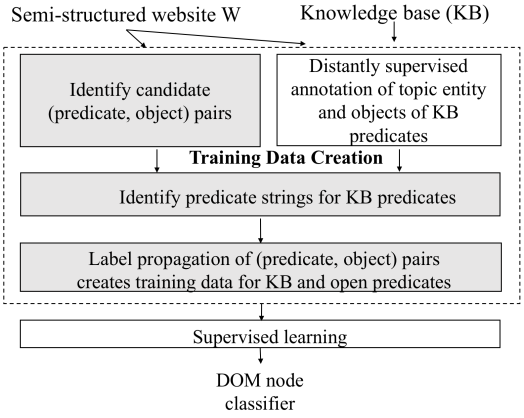


Figure 2: An overview of our proposed semi-structured OpenIE model learning process. Shaded areas indicate contributions of our process.

1. Predicate-object candidate generation: We first generate potential candidate *(predicate, object)* pairs, as described in Section 3.1. The search for these candidate pairs is quasilinear in the number of DOM nodes, thereby avoiding examination of every pair of DOM nodes.
2. Distant supervision: We then use a seed knowledge base (KB) to identify instances of *(predicate, object)* pairs appearing in the seed, where the predicate exists in a pre-defined ontology, as described in Section 3.2. For example, for the page in Figure 1, we would hope to identify *("Director", "Richard Linklater")* assuming that fact was in our KB.
3. Semi-supervised label propagation: We perform a semi-supervised label propagation step to identify pairs of nodes that are formatted similarly to the known *(predicate, object)* pairs as described in Section 3.3; these new *(predicate, object)* pairs give us training labels for new predicates. For example, in Figure 1, we should identify the pair *("Cinematographer", "Maryse Alberti")* since it is formatted similarly to our seed pair, even though the concept of *cinematographer* does not exist in our seed ontology.

3 Approach

We now describe the three key steps we use to generate training labels.

3.1 Candidate pair generation

Recall that for ClosedIE we only need to annotate objects; for OpenIE we need to additionally identify the predicates, which will then allow us to find other predicates not found in the seed KB. Our first step is thus to find all potential *(predicate, object)* pairs on a webpage for further annotation.

Determining which nodes should be checked as potential predicates for a given object is not trivial. On the one hand, there may be hundreds of nodes on the page, so considering all potential pairs of nodes would be computationally expensive. On the other hand, consider that a webpage about a movie may contain a long list of actors under the section header "Cast"; in Euclidean distance, the actor at the bottom of the list may be quite far away from the "Cast" string at the top of the list, so only searching nearby nodes may miss the predicate node.

To identify potential predicate nodes, we use our intuition that predicate strings should be more common across a website than their related object strings. Consider *("Language", "English")* as an example. Even though the object string "English" might be quite common on a site, it should be less frequent than its corresponding predicate string "Language". According to the site-wide frequency rankings, we consider a *(predicate, object)* pair a candidate pair only if the predicate appears more frequently than the object[2].

Following this intuition, we start by computing the frequency of each string found across all pages in the website, and create a ranked list. Second, for each DOM node, we create candidate pairs consisting of the node as object and the k-nearest higher ranking nodes as predicate . Third, to identify potential predicates that may be farther away but still represent a section header for the region of the page containing the object, we recursively travel up the DOM tree, and at each level we find the k-highest ranked candidate predicates paired with any candidate object in that DOM subtree. We create additional candidate pairs pairing those candidate predicates with all candidate objects in that subtree. Thus in total each candidate object is paired with up to dk candidate predicates, where d is the depth of the DOM.

For example, consider the object strings in the Cast section of Figure 1, with a k value of 1. Since "Cast" appears on every page, an initial candidate pair would be created for ("Cast", "Ethan Hawke"). If Hawke is mentioned more frequently across the site than the other actors, he would be paired as the potential predicate for their strings since they are closer to his name than "Cast", such as ("Ethan Hawke", "Robert Sean Leonard"). However, since Hawke and Leonard are in the same `<div>`, the recursive process would add the candidate pair ("Cast", "Robert Sean Leonard") since "Cast" would be the most highly ranked string associated with any object in that section.

3.2 Seed labels

In the second stage, given a webpage, the subject and objects we have identified on the page, and the candidate *(predicate, object)* pairs, we are now ready for distant supervision annotation to generate seed labels that are *(predicate, object)* pairs for the subject appearing in the seed knowledge.

We start with the Ceres object identification to generate a list of nodes containing object strings corresponding to KB predicates, and look up *(predicate, object)* pairs in the candidate list that contain the object node. We use lexical clues to we filter a candidate *(predicate, object)* pair if the predicate name is not semantically similar to the predicate in the ontology. There are multiple ways of doing this. One way is to compare the cosine similarity of word embeddings (such as FastText (Bojanowski et al., 2017)) representing the predicate string and the ontology predicate name and filter using a threshold. Another way is to manually compile a few terms for each predicate in our ontology, and filter a predicate if it does not contain any of the terms as a substring. Empirically we found using a manually compiled list, which takes about a minute per predicate, gives higher precision than using embeddings, though it limits us to the particular language of those terms. After the filtering step, we can fairly safely choose the *(predicate, object)* pair where the predicate is closest to the object in Euclidean distance.

3.3 Semi-supervised Label Propagation

In the third stage, given a set of *(predicate, object)* pairs on a webpage generated in the first stage, we aim at following visual clues to find other *(predicate, object)* pairs on the same page. These new candidate pairs serve as training labels for predicates that may not occur in the seed knowledge.

---

[2] We consider strings consisting of numeric values only as potential objects and not predicates, regardless of their frequency.

We apply semi-supervised learning, which typically resorts to a similarity graph where similar instances are connected by edges in the graph, and propagates existing labels to neighbor nodes. Our intuition is that *(predicate, object)* pairs should share similar formatting; we capture this intuition as we construct the graph.

Graph construction: Each vertex in the similarity graph represents a candidate pair, an edge connecting two vertices indicates that the two candidate pairs are similar, and the edge weight gives the level of similarity. We compute similarity between candidate pairs by visual clues[3], creating an edge between them if they have similar predicate formatting and simliar object formatting. Formatting similarity requires having the same font, font size, font weight, and text alignment, and being either vertically or horizontally aligned.

We then weight the edges by adding up similarities of the horizontal, vertical, and DOM relationship between predicate and object. Similarity of DOM relationship is 1 for exact match and 0 otherwise. Similarity of horizontal relationship is computed by measuring the distance between the predicate and the object in a (pred, obj) pair, and then taking the ratio of minimum distance and maximum distance[4]. We compute similarity of vertical relationship in a similar way, giving:

$$w_{i,j} = \mathbb{1}_{r_d(i)=r_d(j)} + \max\left(0, \frac{\min\left(r_h\left(i\right), r_h\left(j\right)\right)}{\max\left(r_h\left(i\right), r_h\left(j\right)\right)}\right) + \max\left(0, \frac{\min\left(r_v\left(i\right), r_v\left(j\right)\right)}{\max\left(r_v\left(i\right), r_v\left(j\right)\right)}\right)$$

where i and j are candidate pairs, r_d calculates the DOM path, r_h calculates the horizontal distance between candidate predicate and candidate object, and r_v calculates the vertical distance.

A sample graph for the webpage in Figure 1 is shown in Figure 3. The pair ("Director", "Richard Linklater") is connected to the pair ("Cinematographer", "Maryse Alberti") with a weight of 3, since they have identical values for all three relationships, while ("Sub-Genres:", "Psychological Drama") and ("Sub-Genres", "Reunion Films") have an edge weight of 2.1 since the latter's horizontal distance is ten times greater than the former.

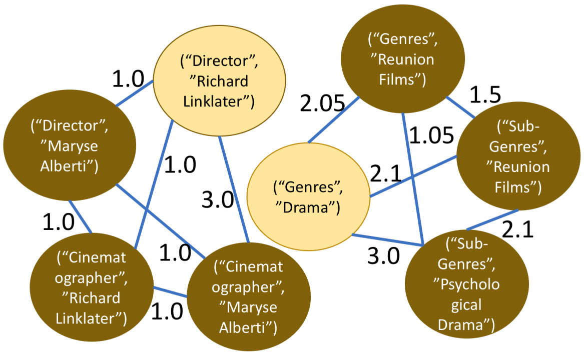


Figure 3: A portion of the graph corresponding to the webpage in Figure 1. Lighter nodes indicate seed pairs labeled by the Ceres process.

To speed up propagation, we keep only the 10 top-weighted edges for each pair. On average, on the dataset in Section 4, pages have 1,142 text fields resulting in 2,813 candidate pairs connected by 14,733 edges, far less than the 1.3 million candidate pairs (and corresponding increase in edges) that would result from a naive pairwise matching.

Label propagation: We use the MultiRankWalk label propagation algorithm (Lin and Cohen, 2010), which has been shown to be successful in very low-data situations. This allows us to propagate even when we only have a single seed label on a page. MultiRankWalk adapts the Personalized PageRank (PPR) algorithm for a classification setting, conducting a PageRank run for each class, with the personalization vector set to equally divide the restart probability among positive labeled examples of the class. The PageRank runs are conducted over the weighted graph constructed in the prior step. Each unlabeled vertex is assigned to the class whose PageRank run gives it the highest score. In our case we have two PPR runs: a positive run for labeled *(predicate, object)* candidates and a negative run for unlabeled candidates.

The results of this process are then used as training data to train a supervised Ceres (Lockard et al., 2018) extraction model.

4 Extended SWDE Benchmark

The Structured Web Data Extraction (SWDE) dataset has served as a benchmark for semi-structured web extraction, with webpages and ground truth extractions from 10 websites in each of 8 domains (Hao et al., 2011). However, the

---

[3] To harvest these features, we render the page using the headless Chrome browser and access element attributes with Selenium (https://www.seleniumhq.org/).

[4] In practice, there are multiple ways to calculate horizontal distance: Left side to left side, left side to right side, right to right, and right to left. The same holds for vertical distance. We calculate each possible ratio and use the one that gives the highest weight. In the case that the ratio is negative (e.g. one pair had predicate to the left of the object while the other pair had it to the right), we set it to 0.

ground truth in SWDE only covers a subset of the predicates found on each site, typically 3-4 predicates per domain.

We extend it as follows: Of the 8 domains, we kept 4 domains whose page topics are named entities. We extended their gold set to include extractions identifying all key-value semi-structured fields on the websites. Since not all SWDE websites can still be rendered in the browser (due to missing resources), we eliminated websites that we were unable to successfully render in the Chrome browser, resulting in 30 websites. We then attempted to create ground truth via a combination of wrapper induction based on manually-labeled training data (with an extractor implementation based on (Gulhane et al., 2011)), hand-crafted extraction rules, and manual cleanup of remaining errors. Eventually, we generated accurate labels for 21 sites in 3 domains.

This new extended benchmark includes both extracted object values as well as the predicate string that accompanies each value on the page. The statistics of the augmented dataset are shown in Table 1. We enhanced SWDE in two ways. First, SWDE on average contains 4,480 triples for 3 predicates from these 21 websites, whereas we have an average of 41K triples for 36 predicates. The number of predicates per website ranges from 5 to 272 (Hollywood features very fine-grained relationships like “Assistant Art Director”). Second, when multiple predicate strings may apply on the webpage, we list all of them in order of specificity. Taking Figure 1 as an example, we include both “Director” and “Crew” for a relation, considering the former to be more specific. To our knowledge, this is the first dataset that represents all key-value pairs found in semi-structured web data.

Domain	Site	# Predicates	# Labels
Movie	AllMovie	65	104,303
Movie	AMCTV	20	85,916
Movie	Hollywood	272	77,047
Movie	iHeartMovies	9	21,253
Movie	IMDb	36	152,880
Movie	Metacritic	18	43,450
Movie	RottenTomatoes	12	65,524
Movie	Yahoo	12	28,354
University	CollegeProwler	26	40,707
University	ECampusTours	17	18,448
University	Embark	67	46,431
University	MatchCollege	68	107,763
University	USNews	22	21,269
NBAPlayer	ESPN	22	6,757
NBAPlayer	FanHouse	16	6,656
NBAPlayer	FoxSports	15	6,157
NBAPlayer	MSNca	12	5,208
NBAPlayer	SI	12	6,082
NBAPlayer	Slam	13	5,453
NBAPlayer	USAToday	5	2,178
NBAPlayer	Yahoo	9	3,912

Table 1: Statistics of the augmented SWDE dataset.

5 Experimental Evaluation

5.1 Experimental Setup

Datasets: Our primary dataset is the augmented SWDE corpus described in Section 4. In addition, we used the set of 31[5] movie websites (comprising 433,000 webpages) found in CommonCrawl[6] from Lockard et al. (2018). To generate seed KBs for the distant supervision, we relied on the methodology from Lockard et al. (2018), using the IMDb database for the Movie domain, and using the original SWDE ground truth for websites CollegeBoard and ESPN to create a KB for the University and NBAPlayer domains respectively.

Implementations: We compared OpenCeres with two baselines. The three algorithms apply the same method to extract topic subjects but differ in how they extract *(predicate, object)* pairs.

1. *WEIR:* Proposed by Bronzi et al. (2013), the Web Extraction and Integration of Redundant data (WEIR) approach takes as input a set of websites in the same subject domain and makes use of overlap in observed entities across sites to learn extraction rules for predicates. The system is unsupervised, though it does require a dictionary of potential page topic entities for the domain to align pages between sites. WEIR also contains a method for automatically identifying predicate strings for the extraction rules it learned by finding strings that frequently occur nearby extracted objects in the HTML templates of sites in the domain.
2. *Colon Baseline:* Semi-structured pages frequently represent a *(predicate, object)* pair via a set of adjacent DOM nodes, with the predicate string ending in a colon and the object string either to its right or below it. This baseline starts with the *(predicate, object)* candidate pairs generated in Section 3.1,

---

[5] We removed the two sites on which Lockard et al. (2018) reported Ceres made no annotations.

[6] www.commoncrawl.org

identifies those where the predicate field ends with a colon, and extracts them as a predicate along with their closest candidate object either to their right or below.

3. *OpenCeres:* This implements our system exactly as described in Section 3, using the generated training data to train a Ceres extractor.

In addition, to understand the uper bound of OpenCeres, we implemented two versions using ground truth data for training seeds:

4. *OpenCeres-Gold:* This implements our system, but skips the label propagation step and replaces noisy seed labels (Section 3.2) with samples from ground truth triples. We sampled 25% of triples for each predicate, so this method is essentially ClosedIE Ceres with incomplete but clean training labels, giving an upper bound on the system's performance when no errors are introduced during training data generation and label propagation.
5. *OpenCeres-GoldProp:* This implements OpenCeres-Gold, but adds the label propagation step described in Section 3.3. Rather than sampling 25% of ground truth triples from all predicates, we instead sample p% of ground truth predicates for a site (with p varying from 10 to 100) and then sample 25% of the corresponding triples for each page. The process is run five times for each setting of p and the results are averaged.

Evaluation: Evaluation is tricky for semi-structured OpenIE because a page may contain multiple valid predicates for a relation. Recall that the SWDE benchmark data we generated (Section 4) lists all predicate strings that are valid, ranked in their order of specificity. We thus define two scores for an extracted triple.

- A *strict score* that requires an exact match between the extracted predicate and the most-specific predicate string in the ground truth.
- A *lenient score* that counts an extraction as correct if the extracted predicate matches any of the predicate strings in the ground truth.

For the SWDE dataset, where we have complete ground truth, we compute precision, recall, and F1. For the CommonCrawl dataset, where no ground truth exists, we sampled 500 extractions at each confidence threshold (giving a 4% margin of error) and manually scored them; since we cannot calculate true recall, we report precision and yield.

System	**Movie**		**NBA**		**University**	
	P	R	P	R	P	R
WEIR (Bronzi et al., 2013)	0.23	0.17	0.08	0.17	0.13	0.18
Colon Baseline	0.63	0.21	0.51	0.33	0.46	**0.31**
OpenCeres	**0.77**	**0.68**	**0.74**	**0.48**	**0.65**	0.29
OpenCeres-Gold	0.99	0.74	0.98	0.80	0.99	0.60

Table 2: Extraction precision and recall (lenient) on SWDE domains. OpenCeres on average improves over baseline by 36% on precision and by 88% on recall.

5.2 Results on SWDE

Overall results: Table 2 shows the precision and recall obtained via lenient scoring. Our results show that OpenCeres outperformed both baselines, achieving an average precision of 72% across the three domains, with an average recall of 48%. Comparing with OpenCeres-Gold on Movie, our precision is 22% lower, while recall is only 5% lower, showing that our label propagation is fairly effective in preserving recall, but introduces errors reducing precision. WEIR does not perform as well as ColonBaseline, showing that our *(predicate, object)* candidate identification technique works well.

Our recall is a robust 68% in the Movie domain, but is much lower in the other two domains. This is because we failed to make any extraction in 3 of the 5 University sites and 2 of the 8 NBA sites due to the inability to find a predicate string for the seed predicates. In some cases no predicate string existed, but in others the string was not in our lexicon. In fact, if we skip those websites where we extract nothing, our recall increases to 58% for NBA and 44% for University. Other recall misses occur when a page has some semi-structured fields that differ significantly in format from those found in our seed ontology, so they were too dissimilar for the label propagation to extend to them.

Details: We now deep dive to results of OpenCeres, shown in Table 3. First, our scoring under the "strict" rules is only slightly lower than under "lenient" rules, because the case that multiple predicates apply is not common and we are often able to find the most specific ones. Across all triples, the overall lenient F1 is 0.68 and strict F1 is 0.61. Second, at predicate-level, OpenCeres has an average precision of 74% and recall of 39%, showing that our method attains high precision for the new predicates it identifies. Third, through the label propagation technique, we are able to extract an average of 10.5 new predicates for every predicate in our seed ontology.

A sample of 100 erroneous OpenCeres extractions shows that 33% of errors are due to the presence of CVTs on the page. For example, the movie site Rotten Tomatoes contains a “Full Review” predicate that contains review date, writer, publication, and text; we extracted only the date, which arguably is not useful. Considering these extractions as correct will increase the precision to 81%. Among the errors, 22% were due to incoherent predicates such as “See More”, while 20% were due to incorrect extraction of a template string as an object of a correct predicate.

Label propagation: Figure 4 shows how the label propagation process successfully creates new training examples from a small number of seeds. While propagation does introduce some precison errors, when only 10% of predicates are given to OpenCeres-GoldProp as seeds (and only 25% of triples sampled for each predicate), training data recall is already nearly 50%. As the percentage of seed predicates rises, the seeds become more likely to capture the full variety of predicate formatting, and recall rises.

There are a number of reasons why the recall upper bound demonstrated by OpenCeres-Gold (and OpenCeres-GoldProp) is less than perfect. First, a small number of relations in the dataset have predicate or object strings that span multiple text fields (particularly in the University vertical); the implemented system can only extract a single text field, so these will be missed. Second, the Candidate Pair Identification algorithm has imperfect recall. Finally, because only 25% of ground truth triples were used for each page of training data, some true positive examples were sampled as negative examples for training, thereby lowering classification recall.

Parameter setting: Table 4 shows that Candidate Pair Identification has increasing recall in capturing true candidate pairs in the SWDE-Movie vertical with more neighbors considered, with a trade-off in increased runtime due to the creation of more pairs; we used $k = 5$ in our experiments.

5.3 Results on CommonCrawl

We now report results of ClosedIE and OpenIE extractions on the 31 CommonCrawl websites; the ClosedIE implementation is a subset of the OpenIE system, without the shaded components in Figure 2. Of these 31 websites, we successfully extracted from 22 sites using OpenIE, and failed to extract from 9 sites because of our inability to match their predicate strings to our lexicon for seed predicates (4 sites were in foreign languages while our lexicon is in English, on 3 sites the pages had no predicate strings labeling the seed object, and 2 sites used terms outside our lexicon).

	Movie	**NBA Player**	**University**
Triple-level F1	0.72 (0.65)	0.58 (0.58)	0.41 (0.36)
Pred-level Prec	0.55 (0.52)	0.86 (0.86)	0.81 (0.76)
Pred-level Rec	0.35 (0.32)	0.46 (0.46)	0.37 (0.35)
Pred-level F1	0.43 (0.40)	0.60 (0.60)	0.51 (0.48)
New:Existing-pred ratio	4.4 : 1	4.3 : 1	23.0 : 1

Table 3: Detailed results of OpenCeres using lenient scoring, with strict scoring results shown in parentheses.

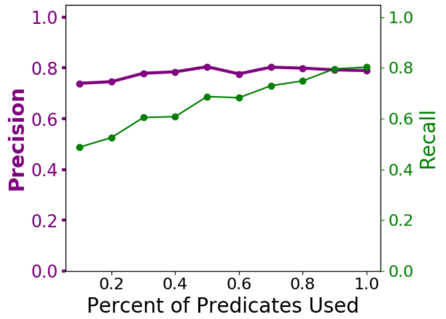


Figure 4: Precision and recall of the training data automatically created for the Movie domain by OpenCeres-GoldProp, after label propagation from seed data created by first sampling varying percents of the ground truth predicates for a site and then sampling a constant 25% of ground truth objects for each predicate.

Figure 5 shows the precision-yield curve of our ClosedIE and OpenIE extractions as we vary the confidence threshold. At a 0.5 confidence threshold, we extracted 2.3M triples at a precision of 0.58, where 1.17M (51%) have new predicates. A higher threshold of 0.8 yielded 1.17M extractions at a precision of 0.70, with 50% of extractions representing new predicates. The high percentage of extractions with new predicates shows the big promise of our method in enriching existing knowledge bases not only with new entities and new facts, but also with new relationships.

6 Related Work

In unstructured text, OpenIE was originally proposed by Banko et al. (2007), an approach extended by ReVerb (Fader et al., 2011) and Ollie (Mausam et al., 2012), which relied on syntactic

k	# candidates	recall
3	1,863	0.92
7	3,044	0.95
11	4,104	0.98

Table 4: Average number of candidate pairs produced by considering k-nearest higher ranking text fields for each candidate object on the SWDE-Movie dataset, along with recall over true pairs.

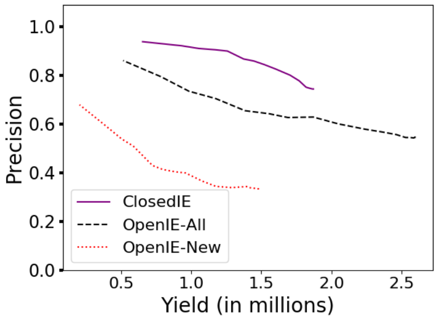


Figure 5: Extraction precision vs. number of extractions on the CommonCrawl dataset at various confidence thresholds; ClosedIE is the implementation of the Ceres system. OpenCeres-All shows all extractions from OpenCeres, while OpenCeres-New shows only the OpenCeres extractions with new predicates.

constraints to identify relation patterns. Our approach is influenced by Wu and Weld (2010), who aligned Wikipedia infobox contents to article text to automatically create training data for an extractor. Recent work on neural extraction models (Cui et al., 2018) has explored entirely supervised models learned from a modified version of the QA-SRL dataset (Stanovsky et al., 2018).

The line of research that has most closely examined the prospect of OpenIE-style extractions using webpage structure is the work on Webtables (Cafarella et al., 2008; Dalvi et al., 2012; Balakrishnan et al., 2015; Cafarella et al., 2018). This work specifically examines the identification of subjects, predicate, and object strings but is limited to fields in rows and columns created using HTML `<table>` tags. Extractions from webtables have recently been harnessed as a source of facts for question-answering systems (Pasupat and Liang, 2015; Krishnamurthy et al., 2017).

In extraction from semi-structured websites, the traditional approach is *wrapper induction*, in which a rule learning algorithm is applied to a set of labeled training examples (Kushmerick et al., 1997). Work in this line of research has achieved high accuracy from only a few labeled examples, but requires manually-annotated examples for each website (Gulhane et al., 2011). To remove this bottleneck, researchers have explored alternative ways to automatically create labeled data and learn models from such potentially noisy labels (Dalvi et al., 2011; Gentile et al., 2015; Furche et al., 2014; Lockard et al., 2018). However, these approaches cannot find triples for predicates that are not in the seed ontology.

The Roadrunner project (Crescenzi et al., 2001) does attempt to identify the objects of all relations represented on a site, but does not extract predicate strings. However, the WEIR project (Bronzi et al., 2013) extended this framework with a heuristic to harvest predicate strings based on words found in DOM nodes that form part of the path of the learned XPath extraction rule. This is the first work that could truly be considered an OpenIE approach to semi-structured extraction. However, as we show in our experiments, the constraints of their heuristic limit the recall of this approach.

7 Conclusions

We presented a new problem of Open Information Extraction from semi-structured websites, and are releasing a new set of over 855,000 ground truth extractions for 21 websites available in the SWDE corpus. We also proposed an algorithm for OpenIE that employs semi-supervised label propagation to discover new predicates based on a set of seed predicates in a known ontology. This method attained a 68% F1 score in OpenIE extractions on our benchmark. In addition, a large-scale evaluation on 31 CommonCrawl movie websites yielded extractions of over two million triples.

In the future, we would like to improve extraction by training a model to extract *(predicate, object)* pairs directly without having to train on particular predicates. Such a model could potentially be based on visual clues common across websites, so a single model could be applied to many sites.

Acknowledgments

We wish to thank Hannaneh Hajishirzi for helpful advice, Paolo Merialdo and Valter Crescenzi for help in running WEIR, and Andrew Bridges, Marc Landers, and Alexander Macdonald for their help in annotating data.

References

Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice. In *CIDR*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew G Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. 2013. Extraction and integration of partially overlapping web sources. *PVLDB*, 6:805–816.

Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *PVLDB*, 11:2140–2149.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yonghui Zhang. 2008. Webtables: exploring the power of tables on the web. *PVLDB*, 1:538–549.

Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118.

Lei Cui, Furu Wei, and Ming Zhou. 2018. Neural open information extraction. In *ACL*.

Bhavana Bharat Dalvi, William W. Cohen, and James P. Callan. 2012. Websets: extracting sets of entities from the web using unsupervised information extraction. In *WSDM*.

Nilesh N. Dalvi, Ravi Kumar, and Mohamed A. Soliman. 2011. Automatic wrappers for large scale web extraction. *PVLDB*, 4:219–230.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Freebase. 2018. Compound value types in Freebase API.

Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, Christian Schallhart, and Cheng Wang. 2014. Diadem: Thousands of websites to a single database. *PVLDB*, 7:1845–1856.

Anna Lisa Gentile, Ziqi Zhang, and Fabio Ciravegna. 2015. Early steps towards web scale information extraction with lodie. *AI Magazine*, 36:55–64.

Pankaj Gulhane, Amit Madaan, Rupesh R. Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeepkumar Satpal, Srinivasan H. Sengamedu, Ashwin Tengli, and Charu Tiwari. 2011. Web-scale information extraction with vertex. *ICDE*, pages 1209–1220.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. 1997. Wrapper induction for information extraction. In *IJCAI*.

Frank Lin and William W. Cohen. 2010. Semi-supervised classification of network data using very few labels. *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 192–199.

Colin Lockard, Xin Dong, Prashant Shiralkar, and Arash Einolghozati. 2018. Ceres: Distantly supervised relation extraction from the semi-structured web. *PVLDB*, 11:1084–1096.

Mausam. 2016. Open information extraction systems and downstream applications. In *IJCAI*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *COLING*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272.

Gabriel Stanovsky, Julian Michael, Luke S. Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL-HLT*.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *ACL*.

Structured Minimally Supervised Learning for Neural Relation Extraction

Fan Bai and **Alan Ritter**
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH
`{bai.313, ritter.1492}@osu.edu`

Abstract

We present an approach to minimally supervised relation extraction that combines the benefits of learned representations and structured learning, and accurately predicts sentence-level relation mentions given only proposition-level supervision from a KB. By explicitly reasoning about missing data during learning, our approach enables large-scale training of 1D convolutional neural networks while mitigating the issue of label noise inherent in distant supervision. Our approach achieves state-of-the-art results on minimally supervised sentential relation extraction, outperforming a number of baselines, including a competitive approach that uses the attention layer of a purely neural model.[1]

1 Introduction

Recent years have seen significant progress on tasks such as object detection, automatic speech recognition and machine translation. These performance advances are largely driven by the application of neural network methods on large, high-quality datasets. In contrast, traditional datasets for relation extraction are based on expensive and time-consuming human annotation (Doddington et al., 2004) and are therefore relatively small. Distant supervision (Mintz et al., 2009), a technique which uses existing knowledge bases such as Freebase or Wikipedia as a source of weak supervision, enables learning from large quantities of unlabeled text and is a promising approach for scaling up. Recent work has shown promising results from large-scale training of neural networks for relation extraction (Toutanova et al., 2015; Zeng et al., 2015).

There are, however, significant challenges due to the inherent noise in distant supervision. For example, Riedel et al. (2010) showed that, when learning using distant supervision from a knowledge base, the portion of mis-labeled examples can vary from 13% to 31%. To address this issue, another line of work has explored *structured* learning methods that introduce latent variables. An example is MultiR (Hoffmann et al., 2011), which is based on a joint model of relations between entities in a knowledge base and those mentioned in text. This structured learning approach has a number of advantages; for example, by integrating inference into the learning procedure it has the potential to overcome the challenge of missing facts by ignoring the knowledge base when mention-level classifiers have high confidence (Ritter et al., 2013; Xu et al., 2013). Prior work on structured learning from minimal supervision has leveraged sparse feature representations, however, and has therefore not benefited from learned representations, which have recently achieved state-of-the-art results on a broad range of NLP tasks.

In this paper, we present an approach that combines the benefits of structured and neural methods for minimally supervised relation extraction. Our proposed model learns sentence representations that are computed by a 1D convolutional neural network (Collobert et al., 2011) and are used to define potentials over latent relation mention variables. These mention-level variables are related to observed facts in a KB using a set of deterministic factors, followed by pairwise potentials that encourage agreement between extracted propositions and observed facts, but also enable inference to override these soft constraints during learning, allowing for the possibility of missing information. Because marginal inference is intractable in this model, a MAP-based approach to learning is applied (Taskar et al., 2004).

Our approach is related to recent work structured learning with end-to-end learned representa-

---

[1] Our code and data are publicly available on Github: `https://github.com/bflashcp3f/PCNN-NMAR`

Proceedings of NAACL-HLT 2019, pages 3057–3069
Minneapolis, Minnesota, June 2 - June 7, 2019.

tions, including Structured Prediction Energy Networks (SPENs) (Belanger and McCallum, 2016); the key differences are the application to minimally supervised relation extraction and the inclusion of latent variables with deterministic factors, which we demonstrate enables effective learning in the presence of missing data in distant supervision. Our proposed method achieves state-of-the-art results on minimally supervised sentential relation extraction, outperforming a number of baselines including one that leverages the attention layer of a purely neural model (Lin et al., 2016).

2 A Latent Variable Model for Neural Relation Extraction

In this section we present our model, which combines continuous representations with structured learning. We first review the problem setting and introduce notation, next we present our approach to extracting feature representations which is based on the piecewise convolutional neural network (PCNN) model of Zeng et. al. (2015) and includes positional embeddings (Collobert et al., 2011). Finally we describe how this can be combined with structured latent variable models that reason about overlapping relations and missing data during learning.

2.1 Assumptions and Problem Formulation

Given a set of sentences, $\mathbf{s} = s_1, s_2 \ldots, s_n$ that mention a pair of knowledge base entities e_1 and e_2 (dyad), our goal is to predict which relation, r, is mentioned between e_1 and e_2 in the context of each sentence, represented by a set of hidden variables, $\mathbf{z} = z_1, z_2, \ldots z_n$. Relations are selected from a fixed set drawn from a knowledge base, in addition to *NA* (no relation). Minimally supervised learning is more difficult than supervised relation extraction, because we do not have direct access to relation labels on the training sentences. Instead, during learning, we are only provided with information about what relations hold between e_1 and e_2 according to the KB. The problem is further complicated by the fact that most KBs are highly incomplete (this is the reason we want to extend them by extracting information from text in the first place), which effectively leads to false-negatives during learning. Furthermore, there are many overlapping relations between dyads, so it is easy for a model trained using minimal supervision from a KB to confuse these relationships. All of these issues are addressed to some degree by the structured learning approach that we present in Section 2.3. First, however we present our approach to feature representation based on convolutional neural networks.

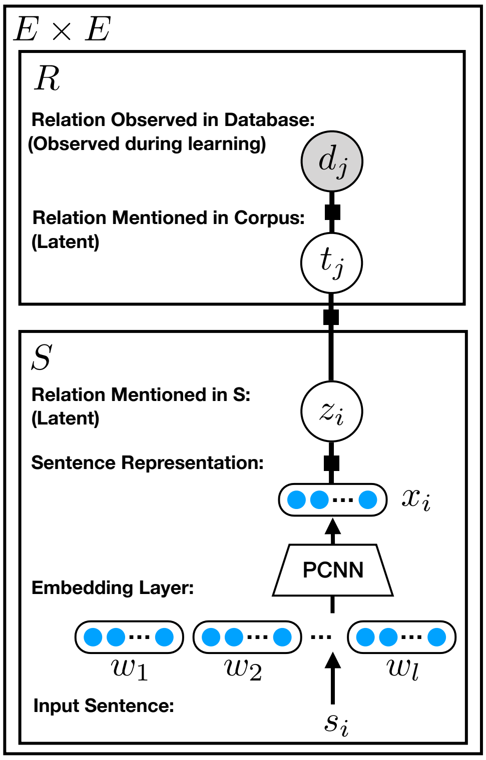


Figure 1: Plate representation of our proposed model. Plates represent replication; $E \times E$ is the number of entity pairs in the dataset, S is the number of sentences mentioning each entity pair and R is the number of relations. Arrows represent functions from input to output. Latent variables are represented as unshaded nodes. Factors over variables are represented as boxes.

2.2 Mention Representation

In the following section we review the Piecewise CNN (PCNN) architecture, first proposed by Zeng et. al. (2015), which is used as the basis for our feature representation.

Input Representation: A sentence, s_i consisting of l words is represented by two types of embeddings: word embeddings, E_i, and position embeddings, P_i relative to the entity pair. Following Lin et. al. (2016), word embeddings were initialized by running Word2Vec on the New York Times corpus and later fine-tuned; position embeddings encode the position of the word relative to KB entities, e_1 and e_2, mentioned in the sentence. The form of input sentence representation is $w_1, w_2, \cdots, w_l$, where $w_i \in \mathbb{R}^d$. The dimension of embedding at each word position is equal to the word embedding dimension plus two times the position embedding size (one position is encoded for each entity).

Convolution: Given an input sentence representation, we perform 1D convolution within a window of length l to extract local features. Assume we have d_f convolutional filters ($F = \{f_1, f_2, \cdots, f_{d_f}\}, f_i \in \mathbb{R}^{l\times d}$). The output of the i-th convolutional filter within the j-th window is:

$$c_{ij} = f_i \cdot w_{j-l+1:j} + b \quad (1 \leq j \leq m + l - 1)$$

Where b is a bias term. We use zero padding when the window slides out of the sentence boundaries.

Piecewise Max Pooling: The output of the convolutional layer c_i is separated into three parts (c_{i1}, c_{i2}, c_{i3}) using the positions of the two entities in the sentence. Max pooling over time is then applied to each of these parts, followed by an elementwise tanh. The final sentence vector is defined as follows:

$$[x]_{ik} = \tanh(\max_j(c_{ikj})) \quad (1 \leq i \leq d_f, 1 \leq k \leq 3)$$

2.3 Structured Minimally Supervised Learning

Our proposed model is based on the PCNN representations described above, in addition to a latent variable model that reasons about missing data and ambiguous relations during learning and is illustrated in Figure 1. The embedding for sentence i, is used to define a factor over the ith input sentence and latent relation mention variable z_i:

$$\phi_{\text{PCNN}}(s_i, z_i) = e^{x_i \cdot \theta_{z_i}}$$

where x_i is the representation for sentence s_i, as encoded by the piecewise CNN.

Another set of factors, ϕ_{OR}, link the sentence-level mention variables, z_i, to aggregate-level variables t_j, representing whether relation j is mentioned between e_1 and e_2 in text. This is modeled using a deterministic OR:

$$\phi_{\text{OR}}(\mathbf{z}, t_j) = \mathbf{1}_{\neg t_j \oplus \exists i: j = z_i}$$

where $\mathbf{1}_x$ is an indicator function that takes the value 1 when x is true. The choice of deterministic OR can be interpreted intuitively as follows: if a proposition is true according to t_j, then it must be extracted from at least one sentence in the training corpus, on the other hand, if it is false, no sentences in the corpus can mention it.

Finally, we incorporate a set of factors that penalize disagreement between observed relations in the KB, d_j, and latent variables t_j, which represent whether relation j was extracted from the text. The penalties for disagreement with the KB are hyperparameters that are adjusted on held-out development data and incorporate entity frequency information from the KB, to model the intuition that more popular entities are less likely to have missing facts:

$$\phi_{\text{A}}(t_j, d_j) = \begin{cases} e^{-\alpha_T}, & \text{if } t_j = 0 \text{ and } d_j = 1 \\ e^{-\alpha_D}, & \text{if } t_j = 1 \text{ and } d_j = 0 \\ 1, & \text{otherwise} \end{cases}$$

Putting everything together, the (unnormalized) joint distribution over $\mathbf{t}$, $\mathbf{d}$ and $\mathbf{z}$ conditioned on sentences $\mathbf{s}$ mentioning a dyad is defined as follows:

$$\begin{aligned} P(\mathbf{d}, \mathbf{t}, \mathbf{z}|\mathbf{s}) \propto & \prod_{i=1}^{|\mathbf{s}|} \phi_{\text{PCNN}}(s_i, z_i) \times \\ & \Big(\prod_{j=1}^{|\mathbf{r}|} \phi_{\text{OR}}(\mathbf{z}, t_j) \phi_{\text{A}}(t_j, d_j)\Big)^{\mu} \\ = & \exp(S_\theta(\mathbf{s}, \mathbf{z}, \mathbf{t}, \mathbf{d})) \end{aligned} \quad (1)$$

Here, μ is a tunable hyperparameter to adjust impact of the disagreement penalty, and $S_\theta(\cdot)$ is the model score for a joint configuration of variables, which corresponds to the log of the unnormalized probability.

A standard conditional random field (CRF) formulation would optimize model parameters, θ so as to maximize marginal probability of the observed KB relations, $\mathbf{d}$ conditioned on observed sentences, $\mathbf{s}$:

$$P(\mathbf{d}|\mathbf{s}) = \sum_{\mathbf{z},\mathbf{t}} P(\mathbf{d}, \mathbf{t}, \mathbf{z}|\mathbf{s})$$

Computing gradients with respect to $P(\mathbf{d}|\mathbf{s})$ (and marginalizing out $\mathbf{z}$ and $\mathbf{t}$) is computationally intractable, so instead we propose an approach that uses maximum-a-posteriori (MAP) parameter learning (Taskar et al., 2004) and is inspired by the latent structured SVM (Yu and Joachims, 2009).

Given a large text corpus in which a set of sentences, $\mathbf{s}$ mention a specific pair of entities (e_1, e_2) and a set of relations $\mathbf{d}$ hold between e_1 and e_2, our goal is to minimize the structured hinge loss: $L_{\text{SH}}(\theta) =$

$$\max\left\{ 0, \begin{array}{c} \max\limits_{\mathbf{z}_e^*,\mathbf{t}_e^*,\mathbf{d}_e^*} \left[S_\theta(\mathbf{s}, \mathbf{z}_e^*, \mathbf{t}_e^*, \mathbf{d}_e^*) + l_{\text{Ham}}(\mathbf{d}_e^*, \mathbf{d})\right] \\ - \max\limits_{\mathbf{z}_g^*,\mathbf{t}_g^*} \left[S_\theta(\mathbf{s}, \mathbf{z}_g^*, \mathbf{t}_g^*, \mathbf{d})\right] \end{array} \right\} \tag{2}$$

Where $l_{\text{Ham}}(\mathbf{d}_e^*, \mathbf{d})$ is the Hamming distance between the bit vector corresponding to the set of observed relations holding between (e_1, e_2) in the KB and those predicted by the model. Minimizing $L_{\text{SH}}(\theta)$ can be understood intuitively as adjusting the parameters so that configurations consistent with observed relations in the KB, $\mathbf{d}$, achieve a higher model score than those with a large hamming distance from the observed configuration. $\mathbf{z}_e^*$ corresponds to the most confusing configuration of the sentence-level relation mention variables (i.e. one that has a large score and also a large Hamming loss) and $\mathbf{z}_g^*$ corresponds to the best configuration that is consistent with the observed relations in the KB.

This objective can be minimized using stochastic subgradient descent. Fixing $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$ to their maximum values in Equation 2, subgradients with respect to the parameters can be computed as follows:

$$\nabla_\theta L_{\text{SH}}(\theta) = \begin{cases} \mathbf{0} & \text{if } L_{\text{SH}}(\theta) \leq 0, \\ \nabla_\theta S_\theta(\mathbf{s},\mathbf{z}_e^*,\mathbf{t}_e^*,\mathbf{d}_e^*) & \\ -\nabla_\theta S_\theta(\mathbf{s},\mathbf{z}_g^*,\mathbf{t}_g^*,\mathbf{d}) & \text{otherwise} \end{cases} \tag{3}$$

$$= \begin{cases} \mathbf{0} & \text{if } L_{\text{SH}}(\theta) \leq 0, \\ \sum_i \nabla_\theta \log \phi_{\text{PCNN}}(s_i, z_{e,i}^*) & \\ -\sum_i \nabla_\theta \log \phi_{\text{PCNN}}(s_i, z_{g,i}^*) & \text{otherwise} \end{cases} \tag{4}$$

Because the second factor of the product in Equation 1 does not depend on θ, it is straightforward to compute subgradients of the scoring function, $\nabla S_\theta(\cdot)$, with fixed values of $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$ using backpropagation (Equation 4).

Inference: The two inference problems, corresponding to maximizing over hidden variables in Equation 2 can be solved using a variety of solutions; we experimented with A* search over left-to-right assignments of the hidden variables. An admissible heuristic is used to upper-bound the maximum score of each partial hypothesis by maximizing over the unassigned PCNN factors, ignoring inconsistencies. This approach is guaranteed to find an optimal solution, but can be slow and memory intensive for problems with many variables. In preliminary experiments on development data, we found that local-search (Eisner and Tromble, 2006) using both relation type and mention search operators (Liang et al., 2010; Ritter et al., 2013) usually finds an optimal solution and also scales up to large training datasets; we use local search with 30 random restarts to compute argmax assignments for the hidden variables, $\mathbf{z}_g^*$ and $\mathbf{z}_e^*$, in all our experiments.

Bag-Size Adaptive Learning Rate: Since the search space of the MAP inference problem increases exponentially as the number of hidden variables goes up, it becomes more difficult to find the exact argmax solution using local search, leading to increased noise in the computed gradients. To mitigate the search-error problem in large bags of sentences, we dynamically modify the learning rate based on the number of sentences in each bag as follows:

$$\lambda_i = \begin{cases} \lambda, & \text{if } |\mathbf{s_i}| < \beta_1 \\ \lambda \times \frac{\beta_1}{|\mathbf{s}_i|}, & \text{if } \beta_1 \leq |\mathbf{s}_i| \leq \beta_2 \\ \lambda \times (\frac{\beta_1}{|\mathbf{s}_i|})^2, & \text{otherwise} \end{cases}$$

where λ_i is the learning rate for ith training entity pair and β_1/β_2 are two tunable bag-size thresholds. In Table 3 and Table 4, we see that this strategy significantly improves performance, especially when training on the larger NYTFB-280K dataset. We also experimented with this method for PCNN+ATT, but found that its performance did not improve.

3 Experiments

In Section 2, we presented an approach that combines the benefits of PCNN representations and structured learning with latent variables for minimally supervised relation extraction. In this section we present the details of our evaluation methodology and experimental results.

Datasets: We evaluate our models on the NYT-Freebase dataset (Riedel et al., 2010) which was created by aligning relational facts from Freebase with the New York Times corpus, and has been used in a broad range of prior work on minimally supervised relation extraction. Several versions of this dataset have been used in prior work; to facilitate the reproduction of prior results, we experiment with two versions of the dataset used by Riedel et. al. (2010) (henceforth NYTFB-68K) and Lin et. al. (2016) (NYTFB-280K). Statistics of these datasets are presented in Table 8. A more detailed discussion about the differences between

Dataset	NYTFB-68K (Riedel et. al. 2010)	NYTFB-280K (Lin et. al. 2016)
Entity pairs	67,946	280,275
Sentences	120,290	523,312

Table 1: Number of entity pairs and sentences in the training portion of Riedel's HELDOUT dataset (NYTFB-68K) and Lin's dataset (NYTFB-280K).

Window length l	3
Number of convolutional filters d_f	230
Word embedding dimension d_w	50
Position embedding dimension d_p	5
Batch size B	1

Table 2: Untuned hyperparameters in our experiments.

datasets used in prior work is also presented in Appendix B.

Hyperparameters: Following Lin et. al. (2016), we utilize word embeddings pre-trained on the NYT corpus using the word2vec tool, other parameters are initialized using the method described by Glorot and Bengio (2010). The Hoffmann et. al. sentential evaluation dataset is split into a development and test set and grid search on the development set was used to determine optimal values for the learning rate λ among $\{0.001, 0.01\}$, KB disagreement penalty scalar μ among $\{100, 200, \cdots, 2000\}$ and β_1/β_2 bag size threshold for the adaptive learning rate among $\{10, 15, \cdots, 40\}$. Other hyperparameters with fixed values are presented in Table 2.

Neural Baselines: To demonstrate the effectiveness of the our approach, we compare against colless universal schema (Verga et al., 2016) in addition to the PCNN+ATT model of Lin et. al. (2016). After training the Lin et. al. model to predict observed facts in the KB, we use its attention layer to make mention-level predictions as follows:

$$p(r_j|x_i) = \frac{\exp(r_j \cdot x_i)}{\sum_{k=1}^{n_r} \exp(r_k \cdot x_i)}$$

Where r_j indicates the vector representation of the jth relation.

Structured Baselines: In addition to initializing convolutional filters used in the $\phi_{\text{PCNN}}(\cdot)$ factors randomly and performing structured learning of representations as in Equation 4, we also experimented with variants of MultiR and DNMAR, which are based on the structured perceptron (Collins, 2002), using fixed sentence representations: both traditional sparse feature representations, in addition to pre-trained continuous representations generated using our best-performing reimplementation of PCNN+ATT. For the structured perceptron baselines, we also experimented with variants based on MIRA (Crammer and Singer, 2003), which we found to provide consistent improvements. More details are provided in Appendix A.

3.1 Sentential Evaluation

In this work, we are primarily interested in mention-level relation extraction. For our first set of experiments (Tables 3 and 4), we use the manually annotated dataset created by (Hoffmann et al., 2011). Note that sentences in the Hoffman et. al. dataset were selected from the output of systems used in their evaluation, so it is possible there are high confidence predictions made by our systems that are not present. Therefore, we further validate our findings, by performing a manual inspection of the highest confidence predictions in Table 5.

NYTFB-68K **Results:** As illustrated in Table 3, simply applying structured models (MultiR and DNMAR) with pre-trained sentence representations performs competitively. MIRA provides consistent improvements for both sparse and dense representations. PCNN+ATT outperforms most latent-variable models on the sentential evaluation, we found this result to be surprising as the model was designed for extracting proposition-level facts. Col-less universal schema does not perform very well in this evaluation; this is likely due to the fact that it was developed for the KBP slot filling evaluation (Ji et al., 2010), and only uses the part of a sentence between two entities as an input representation, which can remove important context. Our proposed model, which jointly learns sentence representations using a structured latent-variable model that allows for the possiblity of missing data, achieves the best overall performance; its improvements over all baselines were found to be statistically significant according to a paired bootstrap test (Efron and Tibshirani, 1994; Berg-Kirkpatrick et al., 2012).[2]

NYTFB-280K **Results:** When training on the larger dataset provided by Lin et. al. (2016), linguistic features are not available, so only neural representations are included in our evaluation. As illustrated in Table 4, PCNNNMAR also achieves the best performance when training on the larger

---

[2] p-value is less than 0.05.

Model	Name	DEV	TEST
Fixed Sentence Representations	MultiR_sparse (Hoffmann et al., 2011)	66.2	63.2
	MultiR_sparse_MIRA	75.3	71.6
	MultiR_continuous	74.2	68.7
	MultiR_continuous_MIRA	80.3	72.5
	DNMAR_sparse (Ritter et al., 2013)	77.9	70.1
	DNMAR_sparse_MIRA	77.5	72.1
	DNMAR_continuous	80.2	70.0
	DNMAR_continuous_MIRA	82.2	74.2
Jointly Learned Representations	PcnnNmar	82.4	83.9
	PcnnNmar (bag size adaptive learning rate)	**85.4**	**86.0**
Baselines	col-less universal schema (Verga et al., 2016)	63.4	61.1
	PCNN+ATT (Lin et al. (2016) code)	81.4	76.4
	PCNN+ATT (our reimplementation with parameter tuning)	83.6	78.4

Table 3: AUC of sentential evaluation precision / recall curves for all models trained on NytFb-68k. Continuous sentence representation works as well as human-engineered sentence representation, and MIRA consistently helps structured perceptron training. PCNN+ATT performs competitively while our PcnnNmar (AdapLR) is statistically significantly better (p-value of bootstrap is less than 0.05)

dataset; its improvements over the baselines are statistically significant. The AUC of most models decreases on the Hoffmann et. al. sentential dataset when training on NytFb-280k. This is not surprising, because the Hoffmann et. al. dataset is built by sampling sentences from positive predictions of models trained on NytFb-68k; changing the training data causes a difference in the ranking of high-confidence predictions for each model, leading to the observed decline in performance against the Hoffmann et. al. dataset. To further validate our findings, we also manually inspect the models' top predictions as described below.

Manual Evaluation: Because the Hoffmann et. al. sentential dataset does not contain the highest confidence predictions, we also manually inspected each model's top 500 predictions for the most frequent 4 relations, and report precision @ N to further validate our results. As shown in Table 5, for NytFb-68k, PCNN+ATT performs comparably on `/location/contains`[3] and `/person/company`, whereas our model has a considerable advantage on the other two relations. For NytFb-280k, our model performs consistently better on all four relations compared with PCNN+ATT. When training on the larger NytFb-280k dataset, we observe trend of increasing mention-level P@N for PcnnNmar, however the performance of PCNN+ATT appears to decrease. We investigate this phenomenon further below.

Performance at Extracting New Facts: To explain PCNN+ATT's drop in mention-level performance after training on the larger NytFb-280k dataset, our hypothesis is that the larger KB-supervised dataset not only contains more true positive training examples but also more false negative examples. This biases models toward predicting facts about popular entities, which are likely to exist in Freebase. To provide evidence in support of this hypothesis, we divide the manually annotated dataset from Hoffmann et. al. into two categories: mentions of facts found in Freebase, and those that are not; this distribution is presented in the Table 6. In Table 7, we present a breakdown of model performance on these two subsets. For PCNN+ATT, although the AUC of in-Freebase mentions on the test set increases after training on the larger NytFb-280k, its Out-Of-Freebase AUC on both dev and test sets drops significantly, which clearly illustrates the problem of increasing false negatives during training. In contrast, our model, which explicitly allows for the possibility of missing data in the KB during learning, has relatively stable performance on the two types of mentions, as the amount of weakly-supervised training data is increased.

3.2 Held-Out Evaluation

In Section 3.1, we evaluated the results of minimally supervised approaches to relation extraction by comparing extracted mentions against human judgments. An alternative approach, which has been used in prior work, is to evaluate a model's performance by comparing predictions against held out facts from a KB. Taken in isolation, this approach to evaluation can be misleading, because it penalizes models that extract many

---

[3] `/location/contains` is the most frequent relation in the Hoffmann et. al. dataset.

Model	Name	DEV	TEST
Fixed Sentence Representations	MultiR_continuous	72.4	66.7
	MultiR_continuous_MIRA	74.6	73.4
	DNMAR_continuous	73.1	68.0
	DNMAR_continuous_MIRA	75.6	68.7
Jointly Learned Representations	PCNNNMAR	78.1	75.4
	PCNNNMAR (bag size adaptive learning rate)	**82.9**	**83.1**
Baselines	col-less universal schema (Verga et al., 2016)	60.3	57.5
	PCNN+ATT (Lin et al. (2016) code)	67.9	72.1
	PCNN+ATT (our reimplementation with parameter tuning)	78.2	74.8

Table 4: AUC of sentential evaluation precision / recall curves for all models trained on NYTFB-280K. Our proposed PCNNNMAR (AdapLR) still performs the best, and the advantage over baselines is also statistically significant (p-value of bootstrap is less than 0.05).

Relation	N	PCNN+ATT	PCNNNMAR (AdapLR)
NYTFB-68K			
/location/contains	100	**1.00**	0.99
	500	0.97	**0.98**
/person/place_lived	100	0.76	**0.98**
	500	0.63	**0.78**
/person/nationality	100	0.62	**0.89**
	500	0.43	**0.54**
/person/company	100	0.98	0.98
	500	0.72	**0.78**
NYTFB-280K			
/location/contains	100	0.98	**0.99**
	500	0.82	**0.99**
/person/place_lived	100	0.58	**0.98**
	500	0.57	**0.84**
/person/nationality	100	0.70	**0.91**
	500	0.35	**0.56**
/person/company	100	0.59	**0.95**
	500	0.40	**0.68**

Table 5: Top: P@N of 4 most frequent relations for models trained on NYTFB-68K. Bottom: P@N of 4 most frequent relations for models trained on NYTFB-280K. Both models can perform well on `/location/contains` relation while PCNNNMAR (AdapLR) is consistently better over other relations.

Category	True	False	Total
DEV			
In-Freebase	102	180	282
Out-Of-Freebase	58	96	154
TEST			
In-Freebase	113	192	305
Out-Of-Freebase	41	99	140

Table 6: Top: Sentence distribution in Hoffmann et. al. (2011) sentential evaluation DEV dataset. Bottom: Sentence distribution in Hoffmann et. al. (2011) sentential evaluation TEST dataset. There are substantial Out-Of-Freebase mentions which are manually labelled as correct relational mentions.

Model	Dataset	InFB	OutFB
DEV			
PCNN+ATT	NYTFB-68K	78.2	89.6
	NYTFB-280K	77.1	77.0
	Change	-1.1	**-12.6**
PCNNNMAR(AdapLR)	NYTFB-68K	81.3	90.4
	NYTFB-280K	77.7	90.6
	Change	**-3.6**	+0.2
TEST			
PCNN+ATT	NYTFB-68K	78.7	75.9
	NYTFB-280K	81.9	56.8
	Change	+3.2	**-19.1**
PCNNNMAR(AdapLR)	NYTFB-68K	85.9	85.4
	NYTFB-280K	83.1	81.5
	Change	-2.8	**-3.9**

Table 7: Top: Comparison of AUCs of In-Freebase and Out-Of-Freebase mentions on sentential DEV set for PCNN+ATT and PCNNNMAR (AdapLR) with two datasets. Bottom: Comparison of AUCs of In-Freebase and Out-Of-Freebase mentions on sentential TEST set for PCNN+ATT and PCNNNMAR (AdapLR) with two datasets. PCNN+ATT has significant drops about Out-Of-Freebase mentions on both sentential DEV and TEST set after training on the larger NYTFB-280K which explains why its overall AUC performances go down while PCNNNMAR (AdapLR) does not have such problem.

new facts that do not already appear in the knowledge base. This is undesirable, because the whole point of an information extraction system is to extract *new* facts that are not already contained in a KB. Furthermore, sentential extraction has the benefit of providing clear provenance for extracted facts, which is crucial in many applications. Having mentioned these limitations of the held-out evaluation metrics, however, we now present results using this approach to facilitate comparison to prior work.

Figure 2 presents precision-recall curves against held out facts from Freebase comparing PCNNNMAR to several baselines and Figure 3 presents results on the larger NYTFB-280K dataset. All models perform better according to the held out evaluation metric when training on the larger dataset, which is consistent with our hypothesis, presented at the end of Section 3.1. Our structured model with learned representations, PCNNNMAR (AdapLR), has lower precision when recall is high. This also fits with our hypothesis, as systems that explicitly model missing data will extract many correct facts that do not appear in the KB, resulting in an under-estimate of precision according to this metric.

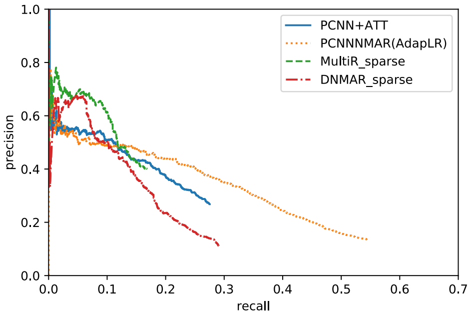


Figure 2: Held-out evaluation precision / recall curves for PCNN+ATT, MultiR, DNMAR and our proposed model PCNNNMAR (AdapLR) on NYTFB-68K.

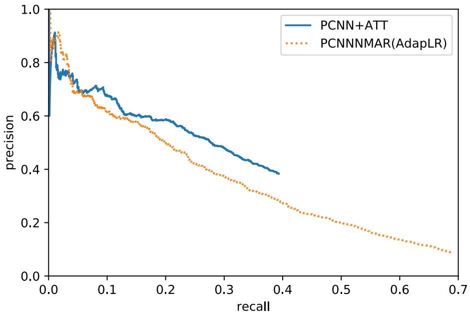


Figure 3: Held-out evaluation precision / recall curves for all NN-based models on NYTFB-280K.

4 Related Work

Knowledge Base Population: There is a long line of prior work on learning to extract relational information from text using minimal supervision. Early work on semantic bootstrapping (Hearst, 1992; Brin, 1998; Agichtein and Gravano, 2000; Carlson et al., 2010; Gupta and Manning, 2014; Qu et al., 2018), applied an iterative procedure to extract lexical patterns and relation instances. These systems tend to suffer from the problem of semantic drift, which motivated work on distant supervision (Craven et al., 1999; Snyder and Barzilay, 2007; Wu and Weld, 2007; Mintz et al., 2009), that explicitly minimizes standard loss functions, against observed facts in a knowledge base. The TAC KBP Knowledge Base Population task was a prominent shared evaluation of relation extraction systems (Ji et al., 2010; Surdeanu, 2013; Surdeanu et al., 2010, 2012). Recent work has explored a variety of new neural network architectures for relation extraction (Wang et al., 2016; Zhang et al., 2017; Yu et al., 2015), experimenting with alternative sentence representations in our framework is an interesting direction for future work. Recent work has also shown improved performance by incorporating supervised training data on the sentence level (Angeli et al., 2014; Beltagy et al., 2018), in contrast our approach does not make use of any sentence-level labels during learning and therefore relies on less human supervision. Finally, prior work has explored a variety of methods to address the issue of noise introduced during distant supervision (Wu et al., 2017; Yaghoobzadeh et al., 2017; Qin et al., 2018).

Another line of work has explored open-domain and unsupervised methods for IE (Yao et al., 2011; Ritter et al., 2012; Stanovsky et al., 2015; Huang et al., 2016; Weber et al., 2017). Universal schemas (Riedel et al., 2013) combine aspects of minimally supervised and unsupervised approaches to knowledge-base completion by applying matrix factorization techniques to multi-relational data (Nickel et al., 2011; Bordes et al., 2013; Chang et al., 2014). Rows of the matrix typically model pairs of entities, and columns represent relations or syntactic patterns (i.e., syntactic dependency paths observed between the entities).

Structured Learning with Neural Representations: Prior work has investigated the combination of structured learning with learned representations for a number of NLP tasks, including parsing (Weiss et al., 2015; Durrett and Klein, 2015; Andor et al., 2016), named entity recognition (Cherry and Guo, 2015; Ma and Hovy, 2016; Lample et al., 2016) and stance detection (Li et al., 2018). We are not aware of any previous work that has explored this direction on the task of minimally supervised relation extraction; we believe structured learning is particularly crucial when learning from minimal supervision to help address the issues of missing data and overlapping relations.

5 Conclusions

In this paper we presented a hybrid approach to minimally supervised relation extraction that combines the benefits of structured learning and learned representations. Extensive experiments show that by performing inference during the learning procedure to address the issue of noise in distant supervision, our proposed model achieves state-of-the-art performance on minimally supervised mention-level relation extraction.

Acknowledgments

Funding was provided by the National Science Foundation under Grant No. IIS-1464128, the Defense Advanced Research Projects Agency (DARPA) via the U.S. Army Research Office (ARO) and under Contract Number W911NF-17-C-0095 and the Office of the Director of National Intelligence (ODNI) and Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C0114, in addition to an Amazon Research Award and an NVIDIA GPU grant. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation here on.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992.

Iz Beltagy, Kyle Lo, and Waleed Ammar. 2018. Improving distant supervision with maxpooled attention and sentence-level supervision. *arXiv preprint arXiv:1810.12956*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta.

K. Chang, W. Yih, B. Yang, and C. Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Colin Cherry and Hongyu Guo. 2015. The unreasonable effectiveness of word representations for twitter named entity recognition. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 735–745.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3(Jan):951–991.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *LREC*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China. Association for Computational Linguistics.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Jason Eisner and Roy W Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Proceedings of the HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*.

Sonal Gupta and Christopher Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Chang Li, Aldo Porco, and Dan Goldwasser. 2018. Structured representation learning for online debate stance prediction. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Percy Liang, Michael I Jordan, and Dan Klein. 2010. Type-based mcmc. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–581. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

M. Nickel, V. Tresp, and H. Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning (ICML)*, pages 809–816.

Pengda Qin, Weiran XU, and William Yang Wang. 2018. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Meng Qu, Xiang Ren, Yu Zhang, and Jiawei Han. 2018. Weakly-supervised relation extraction by pattern-enhanced embedding learning. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 74–84.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics (TACL)*, 1:367–378.

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multilabel classification. In *IJCAI*, pages 1713–1718.

Gabriel Stanovsky, Ido Dagan, et al. 2015. Open ie as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *TAC*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California. Association for Computational Linguistics.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2017. Event representations with tensor-based compositions. *AAAI*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 323–333.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1183–1194.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

A MIRA

Prior work on minimally supervised structured learning has made use of sparse feature representations in combination with perceptron-style parameter updates. We found these updates result in poor performance on held-out development data, however, when using fixed, pre-trained continuous sentence representations. Perhaps this is not surprising because, intuitively, the margin of the dataset is likely to be smaller when using lower dimensional, continuous representations, leading to a larger mistake bound for convergence of the perceptron. To address this, we applied the the **M**argin **I**nfused **R**elaxation **A**lgorithm (Crammer and Singer, 2003), as described below. In Section 3.1, we show empirically that MIRA is crucial for achieving good performance when using continuous representations, and consistently improves performance when using sparse features as well.

As discussed above, we have $\hat{\mathbf{z}}^{\text{KB}}$ the most likely sentence extractions conditioned on the KB and $\hat{\mathbf{z}}$, the MAP assignment to $\mathbf{z}$, ignoring the KB. MIRA updates parameters of the PCNN factors as follows:

$$\theta_j \quad = \quad \theta_j + \tau \cdot \Big(F_j(x_i, \hat{z}_i^{\text{KB}}) - F_j(x_i, \hat{z}_i)\Big)$$

here τ is an adaptive learning rate that scales the update to the smallest step size that achieves 0 loss on each mention-level classification:

$$\tau \quad = \quad \min\left(C, \frac{1 - \theta \cdot \big(F(x_i, \hat{z}_i^{\text{KB}}) - F(x_i, \hat{z}_i)\big)}{2||x_i||^2}\right)$$

θ is the concatenation of parameters θ_j across relations j, and similarly $F(\cdot)$ is the concatenation of PCNN features across relations. C is a hyper-parameter that truncates large steps and helps to prevent overfitting.

B Differing Versions of the NYT-Freebase Corpus Used in Prior Work

We evaluate our models on the NYT-Freebase dataset (Riedel et al., 2010) which was created by aligning relational facts from Freebase with the New York Times corpus, and has been used in a broad range of prior work on minimally supervised relation extraction. Originally, Riedel et. al. created two separate datasets for their HELDOUT and MANUAL evaluations. In the HELDOUT dataset, Freebase entity pairs are divided into two parts, one for training and one for testing. Training dyads are aligned to the 2005-2006 portion of the NYT corpus while testing dyads are aligned to the year 2007. In the MANUAL evaluation data, **all** Freebase entity pairs are matched against the 2005-2006 articles and used as training instances. Testing data in the Riedel et. al. MANUAL evaluation consists of dyads found within sentences in the 2007 NYT articles, for which at least one entity does not appear in Freebase; their models' predictions on this data were annotated manually. The Riedel et. al. data splits ensure it is not possible to have overlapping train/test entity pairs in either the HELDOUT or MANUAL evaluation.

As neural models with many parameters typically benefit significantly from larger quantities of training data, Lin et. al. (2016) added training data from the Riedel et. al. MANUAL-TRAIN dataset into their training dataset. This modification of the training data leads to overlap in the entity pairs in the Lin et. al. training/test split. We found 11,424 entity pairs appearing in both training and test sets, however no sentences appear in both the training and test sets, as the matched NYT articles came from different time periods.[4] In all our evaluations we remove these overlapping entity pairs from the training set, to ensure the models are not simply memorizing KB facts that appear in the training data. Figure 4 shows that after removing these shared entity pairs from the training data, performance of the Lin et. al. PCNN+ATT model does not change very much when evaluating against held out facts from Freebase.

We name two versions of the NYT-Freebase dataset according to the number of training entity

---

[4] We downloaded the Lin et. al. (2016) dataset from the associated Github repository (`https://github.com/thunlp/NRE`) on June, 2017. The repository was updated in March and May 2018, addressing the overlapping-entity-pairs issue using the same approach described in our paper.

Dataset	NYTFB-68K (Riedel et. al. 2010)	NYTFB-280K (Lin et. al. 2016)
Entity pairs	67,946	280,275
Sentences	120,290	523,312
Distinct sent.	96,340	340,970
Relations	52	53

Table 8: Number of entity pairs and sentences in the training portion of Riedel's HELDOUT dataset (NYTFB-68K) and Lin's dataset (NYTFB-280K).

pairs they include. Table 8 shows that NYTFB-280K training set has around 4 times the number of sentences and entity pairs as NYTFB-68K, and the proportions of multi-sentence entity pairs in NYTFB-280K is higher. In Table 9, we can see that the distribution of relations in the two datasets are comparable, but NYTFB-280K has much more entity pairs for each relation. Also, Figure 5 tells us that NYTFB-280K has a wider bag-size range and more large training bags.

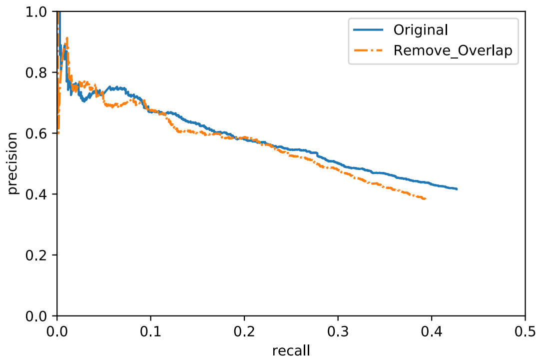


Figure 4: Held-out evaluation precision / recall curves for PCNN+ATT model on original NYTFB-280K and its shared-entity-pairs-removed version.

Relation	NYTFB-68K		NYTFB-280K	
	# EPs	percent	# EPs	percent
NA	63596	93.12	263372	93.52
/location/contains	2147	3.14	7760	2.76
/person/place_lived	581	0.85	2300	0.86
/person/nationality	436	0.64	2553	0.87
/person/place_of_birth	370	0.54	1400	0.49
/person/company	357	0.52	1417	0.50

Table 9: Distribution of the most frequent relations in the training set of NYTFB-68K and NYTFB-280K.

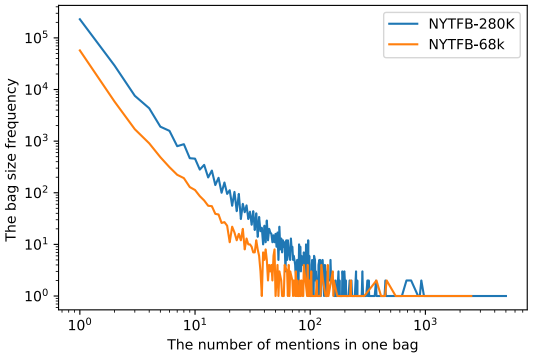


Figure 5: Distribution of bag size in the training set of the NYTFB-68K and NYTFB-280K.

Method		DEV	TEST
0/1 loss	normal	82.6	82.8
	AdapLR	83.9	81.3
relation-level	normal	83.9	83.1
	AdapLR	84.6	81.1
mention-level	normal	82.4	83.9
	AdapLR	**85.4**	**86.0**

Table 10: AUC of sentential evaluation precision / recall curves for PCNNNMAR with three loss functions trained on NYTFB-68K. Mention-level hamming loss has some advantages over other two loss functions.

C Variations on Structured Hinge Loss

Since we use the hinge loss as the loss function in our proposed PCNNNMAR model, the way that the hamming loss is calculated decides how we solve the argmax problem in loss-augmented search. In our experiments, we explore three ways to compute the loss: 0/1 loss, relation-level hamming loss and mention-level hamming loss. Table 10 shows that mention-level hamming loss has obvious advantage on AUC performance over other two methods. Although theoretically relation-level hamming loss should be better, it is really hard to find the exact argmax solution in loss-augmented inference with local search while we can easily get it with mention-level hamming loss.

Neural Machine Translation of Text from Non-Native Speakers

Antonios Anastasopoulos[†,1] **Alison Lui**[†,2] **Toan Q. Nguyen**[2] **David Chiang**[2]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Department of Computer Science and Engineering, University of Notre Dame
`aanastas@cs.cmu.edu` `{alui,tnguye28,dchiang}@nd.edu`

Abstract

Neural Machine Translation (NMT) systems are known to degrade when confronted with noisy data, especially when the system is trained only on clean data. In this paper, we show that augmenting training data with sentences containing artificially-introduced grammatical errors can make the system more robust to such errors. In combination with an automatic grammar error correction system, we can recover 1.0 BLEU out of 2.4 BLEU lost due to grammatical errors. We also present a set of Spanish translations of the JFLEG grammar error correction corpus, which allows for testing NMT robustness to real grammatical errors.

1 Introduction

Neural Machine Translation (NMT) is undeniably a success story: public benchmarks (Bojar et al., 2016) are dominated by neural systems, and neural approaches are the *de facto* option for industrial systems (Wu et al., 2016; Hassan Awadalla et al., 2018; Crego et al., 2016; Hieber et al., 2018). Even under low-resource conditions, neural models were recently shown to outperform traditional statistical approaches (Nguyen and Chiang, 2018).

However, there are still several shortcomings of NMT that need to be addressed: a (non-exhaustive) list of six challenges is discussed by Koehn and Knowles (2017), including out-of-domain testing, rare word handling, the wide-beam problem, and the large amount of data needed for learning. An additional challenge is robustness to noise, both during training and at inference time.

In this paper, we study the effect of a specific type of noise in NMT: grammatical errors. We primarily focus on errors that are made by non-native source-language speakers (as opposed to dialectal language, SMS or Twitter language). Not only is this linguistically important, but we believe that it would potentially have great social impact.

Our contributions are three-fold. First, we confirm that NMT is vulnerable to source-side noise when trained on clean data, losing up to 3.6 BLEU on our test set. This is consistent with previous work, yet orthogonal to it, since we use more realistic noise for our experiments. Second, we explore training methods that can deal with noise, and show that including noisy synthetic data in the training data makes NMT more robust to handling similar types of errors in test data. Combining this simple method with an automatic grammar correction system, we find that we can recover 1.5 BLEU. Third, we release Spanish translations of the JFLEG corpus,[1] a standard benchmark for English Grammar Error Correction (GEC) systems. We also release all other data and code used in this paper.

Our additional annotations on both the JFLEG corpus and the English WMT data will enable the evaluation of the robustness of NMT systems on realistic, natural noise: a robust system would ideally produce the same output when presented with either the original or the noisy source sentence. We hope that our datasets will become a benchmark for noise-robust NMT, because we believe that deployed systems should also be able to handle source-side noise.

2 Data

We focus on NMT from English to Spanish. We choose English to be our source-side language because there exist English corpora annotated with grammar corrections, which we can use as a

---

[†] Equal contribution. Work performed at the University of Notre Dame.

[1] Freely available at https://bitbucket.com/antonis/nmt-grammar-noise

Proceedings of NAACL-HLT 2019, pages 3070–3080
Minneapolis, Minnesota, June 2 - June 7, 2019.

source of natural noise. Moreover, since English is probably the most commonly spoken non-native language (Lewis et al., 2009), our work could be directly applicable to several translation applications. Our choice of Spanish as a target language enables us to have access to existing parallel data and easily create new parallel corpora (see below, §2.3).

For all experiments, we use the Europarl English-Spanish dataset (Koehn, 2005) as our training set. In the synthetic experiments of Section §2.2, we use the newstest2012 and newstest2013 as dev and test sets, respectively. Furthermore, to test our translation methods on real grammatical errors, we introduce a new collection of Spanish translations of the JFLEG corpus (§2.3).

2.1 Grammar Error Correction Corpora

To our knowledge, there are five publicly available corpora of non-native English that are annotated with corrections, which have been widely used for research in Grammar Error Correction (GEC). The NUS Corpus of Learner English (NUCLE) contains essays written by students at the National University of Singapore, corrected by two annotators using 27 error codes (Dahlmeier et al., 2013). It has become the main benchmark for GEC, as it was used in the CoNLL GEC Shared Tasks (Ng et al., 2013, 2014). Other corpora include the Cambridge Learner Corpus First Certificate in English FCE corpus (Yannakoudakis et al., 2011), which is only partially public, the LANG-8 corpus (Tajiri et al., 2012), which was harvested from online corrections, and the AESW 2016 Shared Task corpus, which contains corrections on texts from scientific journals.

The last corpus is the JHU FLuency-Extended GUG corpus (JFLEG) (Napoles et al., 2017). This corpus covers a wider range of English proficiency levels on the source side, and its correction annotations include extended fluency edits rather than just minimal grammatical ones. That way, the corrected sentence is not just grammatical, but also guaranteed to be fluent.

2.2 Synthetic grammar errors

Ideally, we would train a translation model to translate grammatically noisy language by training it on parallel data with grammatically noisy language. Since, to our knowledge, no such data exist in the quantities that would be needed, an alternative is to add synthetic grammatical noise to clean data. An advantage of this approach is that controlled introduction of errors allows for fine-grained analysis.

Error Type	Confusion Set
ART	{a, an, the, $\emptyset$}
PREP	{on, in, at, from, for, under, over, with, into, during, until, against, among, throughout,of, to, by, about, like, before, after, since, across, behind, but, out, up, down, off, $\emptyset$}
NN	{SG, PL}
SVA	{3SG, not 3SG, 2SG-Past, not 2SG-Past}

Table 1: Confusion sets for each grammar error type. The ART and PREP sets include an empty token ($\emptyset$) allowing for insertions and deletions. SG, PL, 2SG, and 3SG stand for singular, plural, second-person and third-person singular respectively.

This is a two-step process, similar to the methods used in the GEC literature for creating synthetic data based on confusion matrices (Rozovskaya et al., 2014; Rozovskaya and Roth, 2010; Xie et al., 2016; Sperber et al., 2017). First, we mimic the distribution of errors found in real data, and then introduce errors by applying rule-based transformations on automatic parse trees.

The first step involves collecting error statistics on real data. Conveniently, the NUCLE corpus has all corrections annotated with 27 error codes. We focus on five types of errors, with the last four being the most common in the NUCLE corpus:

- DROP: randomly deleting one character from the sentence.[2]
- ART: article/determiner errors
- PREP: preposition errors
- NN: noun number errors
- SVA: subject-verb agreement errors

Using the annotated training set of the NUCLE corpus, we compute error distribution statistics, resulting in confusion matrices for the cases outlined in Table 1. For ART and PREP errors, we obtain probability distributions that an article, determiner, or preposition is deleted, substituted with another member of the confusion set, or inserted in the beginning of a noun phrase. For NN errors,

---

[2] This error is not part of the NUCLE error list.

Dataset	Percentage of Errors Train	 Dev	 Test
sentences	2M	3K	3K
words	55M	74K	73K
DROP	100%	100%	100%
ART	96.4%	98.4%	99.8%
PREP	95.7%	95.9%	98.4%
NN	94.5%	91.0%	98.6%
SVA	93.1%	81.9%	82.0%
CLEAN+DROP	50%	50%	–
CLEAN+ART	48.2%	49.1%	–
CLEAN+PREP	47.8%	47.9%	–
CLEAN+NN	47.3%	45.5%	–
CLEAN+SVA	46.5%	41.0%	–
MIX-ALL	79.9%	77.8%	–

Table 2: Statistics on the original and synthetic En-Es datasets. Each (synthetic) sentence has exactly one introduced error, wherever possible. CLEAN+[ERROR] is the concatenation of the [ERROR] with the original clean dataset, while MIX-ALL includes six versions of each training sentence, one without errors and one for each error.

we obtain the probability of a noun being replaced with its singular or plural form. For SVA errors, the probability that a present tense verb is replaced with its third-person-singular (3SG) or not-3SG form. An additional SVA error that we included is the confusion between the appropriate form for the verb 'to be' in the past tense ('was' and 'were').

The second step involves applying the noise-inducing transformations using our collected statistics as a prior. We obtained parses for each sentence using the Berkeley parser (Petrov et al., 2006). The parse tree allows us to identify candidate error positions in each sentence (for example, the beginning of a noun phrase without a determiner, were one could be inserted). For each error type we introduced exactly *one* error per sentence, wherever possible, which we believe matches more realistic scenarios than previous work. It also allows for controlled analysis of the behaviour of the NMT system (see Section 4).

For each error and each sentence, we first identify candidate positions (based on the error type and the parse tree) and sample one of them based on the specific error distribution statistics. Then, we sample and introduce a specific error using the corresponding probability distribution from the confusion matrix. (In the case of DROP, NN, and SVA errors, we only need to sample the position and only insert/substitute the corresponding error.) If no candidate positions are found (for example, a sentence doesn't have a verb that can be substituted to produce a SVA error) then the sentence remains unchanged.

Following the above procedure, we added errors in our training, dev, and test set (henceforth referred to as [ERROR]). Basic statistics on our produced datasets can be found in Table 2, while example sentences are shown in Table 3. Furthermore, we created training and dev sets that mix clean and noisy data. The CLEAN+[ERROR] training sets are the concatenation of each [ERROR] with the clean data, effectively including a clean and a noisy version of each sentence pair.

We also created a training and dev dataset with mixed error types, in our attempt to study the effect of including all noise types during training. The MIX-ALL dataset includes each training pair six times: once with the original (clean) sentence as the source, and once for every possible error. We experimented with a mixed dataset that included each training sentence once, with the number of noisy sentences being proportional to the real error distributions of the NUCLE dataset, but obtained results similar to the [ERROR] datasets.

2.3 JFLEG-ES: Spanish translations of JFLEG

The JFLEG corpus consists of a dev and test set (no training set), with 747 and 754 English sentences, respectively, collected from non-native English speakers. Each sentence is annotated with four different corrections, resulting in four (fluent and grammatical) reference sentences. About 14% of the sentences do not include any type of error, with the source and references being equivalent.

We created translations of the JFLEG corpus that allow us to evaluate how well NMT fares compared to a human translator, when presented with noisy input. We will refer to the augmented JFLEG corpus as JFLEG-ES.

Two professional translators were tasked with producing translations for the dev and the test set, respectively. The translators were presented only with the original erroneous sentences; they did not have access to the correction annotations. They were asked to produce fluent, grammatical translations in European Spanish (to match the

Error Type	Example
ART	In October , Tymoshenko was sentenced to seven years in prison for entering into what was reported to be *a/*∅* disadvantageous gas deal with Russia. Its ratification would require *∅/*the* 226 votes. It is *a/*the* good result, which nevertheless involves a certain risk.
PREP	[. . .] the motion to revoke an article based *on/*in* which the opposition leader , Yulia Tymoshenko , was sentenced. Its ratification would require *∅/*for* 226 votes.
NN	Its ratification would require 226 *votes/*vote*. The *verdict/*verdicts* is not yet final ; the court will hear Tymoshenko 's appeal in December.
SVA	As a rule, Islamists *win/*wins* in the country; the question is whether they are the moderate or the radical ones. This cultural signature *accompanies/*accompany* the development of Moleskine;

Table 3: Example grammatical errors that were introduced in the En-Es WMT test set.

Spanish used in the Europarl corpus). There exist cases where a translator might choose to preserve a source-side error when producing the translation, such as translation of literary works where it's possible that grammar or fluency errors are intentional; however, our translators were explicitly asked not to do that. The exact instructions were as follows:

> Please translate the following sentences. Note that some sentences will have grammatical errors or typos in English. Don't try to translate the sentences word for word (e.g. replicate the error in Spanish). Instead, try to translate it as if it was a grammatical sentence, and produce a fluent grammatical Spanish sentence that captures its meaning.

3 Experiments

In this section, we provide implementation details and the results of our NMT experiments. For convenience, we will refer to each model with the same name as the dataset it was trained on; e.g. the MIX-ALL model will refer to the model trained on the MIX-ALL dataset.

3.1 Implementation Details

All data are tokenized, truecased, and split into subwords using Byte Pair Encoding (BPE) with 32,000 operations (Sennrich et al., 2016). We filter the training set to only contain sentences up to 80 words.

Our LSTM models are implemented using DyNet (Neubig et al., 2017), and our transformer models using PyTorch (Paszke et al., 2017). The transformer model uses 6 layers, 8 attention heads, the dimension for embeddings and positional feed-forward are 512 and 2048 respectively . The sub-layer computation sequence follows the guidelines from Chen et al. (2018). Dropout probability is set to 0.2 (also in the source embeddings, following Sperber et al. (2017)). We use the learning rate schedule in Vaswani et al. (2017) with warm-up steps of 24000 but only decay the learning rate until it reaches 10^{-5} as inspired by Chen et al. (2018). For testing, we select the model with the best performance on the dev set corresponding to the test set. At inference time, we use a beam size of 4 with length normalization (Wu et al., 2016) with a weight of 0.6.

3.2 Results

We report the results obtained with the transformer model, as they were consistently better than the LSTM one. All the result tables for the LSTM models can be found in the Appendix.

The performance of our systems on the synthetic WMT test sets, as measured by detokenized BLEU (Papineni et al., 2002), is summarized in Table 4. When the system is trained only on clean data (first row) and tested on noisy data, it unsurprisingly exhibits degraded performance. We observe significant drops in the range of 1.0–3.6 BLEU.

WMT Training Set	En-Es WMT Test Set						
	CLEAN	DROP	ART	PREP	NN	SVA	AVERAGE ± STDEV
CLEAN	**33.0**	29.6	31.3	**32.0**	29.3	**32.1**	31.2 ± 1.5
DROP	31	30.2	30.0	30.0	28.3	30.6	30.0 ± 0.9
ART	31.2	28.4	30.8	30.2	27.7	30.8	29.8 ± 1.4
PREP	30.4	27.8	29.3	30.3	27.4	29.9	29.2 ± 1.3
NN	30.4	27.9	28.9	29.5	29.8	29.8	29.4 ± 0.8
SVA	31.2	28.7	30.2	30.3	28.2	30.9	29.9 ± 1.2
CLEAN+DROP	**32.9**	**31.4**	**31.4**	31.8	29.5	**32.0**	31.5 ± 1.2
CLEAN+ART	**32.7**	29.7	**31.7**	31.7	28.8	**32.1**	31.1 ± 1.5
CLEAN+PREP	**32.7**	29.6	31.2	**32.2**	29.0	31.8	31.1 ± 1.5
CLEAN+NN	32.5	29.4	30.7	31.4	**31.0**	31.6	31.1 ± 1.0
CLEAN+SVA	32.5	29.6	31.2	31.5	29.0	**31.9**	30.9 ± 1.4
MIX-ALL	**32.7**	30.9	**31.4**	**32.0**	**30.6**	**32.0**	**31.6 ± 0.7**

Table 4: BLEU scores on the WMT test set without (CLEAN) and with synthetic grammar errors. The best performing models for each test set are **highlighted**. When training and test match (highlighted) we generally observe higher results. However, including all clean and noisy data in the training set (MIX-ALL) yields the best results across almost all datasets, with the highest average BLEU *and* the lowest variance.

The largest drop (more than 3.5 BLEU) is observed with NN errors in the source sentence. This is not unreasonable: nouns almost always carry content significant for translation. Especially when translating into Spanish, a noun number change can, and apparently does, also affect the rest of the sentence significantly, for example, by influencing the conjugation of a subsequent verb. The second-largest drop (more than 3.0 BLEU points) is observed in the case of DROP errors. This is also to be expected; typos produce out-of-vocabulary (OOV) words, which in the case of BPE are usually segmented to a most likely rarer subword sequence than the original correct word.

We find that a training regime that includes both clean and noisy sentences ([CLEAN+ERROR) results in better systems across the board. Importantly, these models manage to perform en par with the CLEAN model on the CLEAN test set. Since the original training set is part of the [CLEAN+ERROR training sets, this behavior is expected. We conclude, thus, that including the full clean dataset during training is important for performance on clean data – one cannot just train on noisy data.

The [CLEAN+ERROR] systems exhibit a notable pattern: their BLEU scores are generally similar to the CLEAN system on all test sets, except for the test set that matches their training set errors (highlighted in Table 4), where they generally obtain the best performance.

The MIX-ALL model is our best system on all test sets (except DROP) and on average. Unlike the [CLEAN+ERROR] systems, it outperforms the CLEAN model on *all* noisy test sets and not only on a specific one. On average, using the MIX-ALL training set leads to an improvement of 0.4 BLEU over the CLEAN model and 0.1 − 0.7 BLEU over the [CLEAN+ERROR] models. Furthermore, the MIX-ALL model exchibits the smallest performance standard deviation of all models, averaging over all test sets. This is another indication that our system is more robust to multiple source-side variations. We further explore this intuition in Section 4.

On the more realistic JFLEG-ES dev and test sets, we observe same trends but at a smaller scale, as shown in Table 5. Our MIX-ALL model generally achieves comparable results when presented with each of the four reference corrections of the test set (CORX columns). However, when we use the noisy source sentence as input (No CORR column) our MIX-ALL model obtains 1.4 BLEU improvements over the CLEAN model. The *difference* between the performance of the models when presented with clean and noisy input is another indicator for robustness. On the JFLEG-ES test set, the noisy source results in a −3.1 BLEU point drop for the CLEAN model, while the drop for our MIX-ALL model is smaller, at −1.7 BLEU points.

In addition, we experimented with using an automatic error-corrected source as input to our sys-

JFLEG-ES Dev

Training	Manual correction					No	Auto
	COR0	COR1	COR2	COR3	avg.	corr.	corr.
CLEAN	32.1	31.5	32.5	33.3	32.4	31.1	31.2
MIX-ALL	31.9	31.4	32.2	**32.9**	**32.1**	**32.2**	**31.6**

JFLEG-ES Test

Training	Manual correction					No	Auto
	COR0	COR1	COR2	COR3	avg.	corr.	corr.
CLEAN	**28.4**	**28.8**	**29.1**	**28.2**	**28.6**	26.2	**27.0**
MIX-ALL	27.7	28.1	28.1	27.5	27.8	**26.8**	**26.7**

Table 5: BLEU scores on the JFLEG-ES dev and test datasets. Our proposed MIX-ALL model is slightly behind the CLEAN model on manually corrected input (COR[0–3]). On noisy input (No corr.) the MIX-ALL outperforms the CLEAN model (26.8 > 26.2). Preprocessing the noisy input with a GEC model (Auto corr.) slightly improves results.

tem (column AUTO CORR of Table 5). We used the publicly available JFLEG outputs of the (almost) state-of-the-art model of Junczys-Dowmunt and Grundkiewicz (2016) as inputs to our NMT system.[3] This experiment envisions a pipeline where the noisy source is first automatically corrected and then translated. As expected, this helps the CLEAN model (by +1.1 BLEU), but our MIX-ALL training helps even further (by another +0.8 BLEU). Interestingly, the automatic GEC system only helps in the test set, while there are no improvements in the dev set. Naturally, since automatic GEC systems are imperfect, the performance of this pipeline still lags behind translating on clean data.

4 Analysis

We attempt an in-depth analysis of the impact of the different source-side error types on the behavior of our NMT system, when trained on clean data and tested on the artificial noisy data that we created.

ART Errors Table 6 shows the difference of the BLEU scores obtained on the sentences, broken down by the type of article error that was introduced. The first observation is that in all cases the difference is negative, meaning that we get higher BLEU scores when testing on clean data. Encouragingly, there is practically no difference when we substitute ‘a’ with ‘an’ or ‘an’ with ‘a’; the model seems to have learned very similar representations for the two indefinite articles, and as a result such an error has no impact on the produced output. However, we observe larger performance drops when substituting indefinite articles with the definite one and vice versa; since the target language makes the same article distinction as the source language, any article source error is propagated to the produced translation.

PREP Errors Due to the large number of prepositions, we cannot present a full analysis of preposition errors, but highlights are shown in Table 7. Deleting a correct preposition or inserting a wrong one leads to performance drops of 1.2 and 0.8 BLEU points for the CLEAN model, but drops of 0.4 and 0.7 for the MIX-ALL model.

NN and SVA Errors We found no significant performance difference between the different NN errors. Incorrectly pluralizing a noun has the same adverse effect as singularizing it, leading to performance reductions of over 4.0 and 3.5 BLEU points respectively. We observe a similar behavior with SVA errors: each error type leads to roughly the same performance degradation.

5 Related Work

The effect of noise in NMT was recently studied by Khayrallah and Koehn (2018), who explored noisy situations during training due to web-crawled data. This type of noise includes misaligned, mistranslated, or untranslated sentences which, when used during training, significantly degrades the performance of NMT. Unlike our

---

[3] This model has been recently surpassed by other systems, e.g. (Junczys-Dowmunt et al., 2018), but their outputs are not available online.

Correct article	Substituted article				
	a	an	the	∅	*all*
a	–	0	−2.0	−2.1	−2.1
an	0	–	−5.7	−7.3	−6.3
the	−4.1	−2.2	–	−1.7	−1.8
∅	−3.1	−3.7	−1.5	–	−1.7
all	−3.8	−3.4	−1.5	−1.8	−1.7

Table 6: Effect of article substitutions in test data (ART) relative to clean test data (CLEAN), broken down by substitution type. Different article substitutions have very different impacts on BLEU; changing an indefinite article to definite is especially damaging.

Substitution	model BLEU difference	
	CLEAN	MIX-ALL
in→with	−6.7	−1.7
on→for	−6.0	−0.1
to→on	−2.9	−0.5
in→ ∅	−1.8	−1.9
∅→for	−1.6	−0.6
∅→*any*	−1.2	−0.4
any→∅	−0.8	−0.7

Table 7: Effect of selected preposition substitutions in test data (PREP) relative to clean test data (CLEAN), for the CLEAN and MIX-ALL models. The MIX-ALL model handles most errors more efficiently.

work, they primarily focus on a setting where the training set is noisy but the test set is clean.

In addition, Heigold et al. (2018) evaluated the robustness of word embeddings against word scrambling noise, and showed that performance in downstream tasks like POS-tagging and MT is especially hurt. Sakaguchi et al. (2017a) studied word scrambling and the *Cmabrigde Uinervtisy (Cambridge University) effect*, where humans are able to understand the meaning of sentences with scrambled words, performing word recognition (word level spelling correction) with a semi-character RNN system.

Focusing only on character-level NMT models, Belinkov and Bisk (2018) showed that they exhibit degraded performance when presented with noisy test examples (both artificial and natural occurring noise). In line with our findings, they also showed that slightly better performance can be achieved by training on data artificially induced with the same kind of noise as the test set.

Sperber et al. (2017) proposed a noise-introduction system reminiscent of WER, based on insertions, deletions, and substitutions. An NMT system tested on correct transcriptions achieves a BLEU score of 55 (4 references), but tested on the ASR transcriptions it only achieves a BLEU score of 35.7. By introducing similar noise in the training data, they were able to make the NMT system slightly more robust. Interestingly, they found that the optimal amount of noise on the training data is smaller than the amount of noise on the test data.

The notion of linguistically plausible corruption is also explored by Li et al. (2017), who created adversarial examples with syntactic and semantic noise (reordering and word substitutions respectively). When training with these noisy datasets, they obtained better performance on several text classification tasks. Furthermore, in accordance with our results, their best system is the one that combines different types of noise.

We present a summary of relevant previous work in Table 8. *Synthetic* errors refer to noise introduced according an artificially created distribution, and *natural* errors refer to actual errorful text produced by humans. As for *semi-natural*, it refers to either noise introduced according to a distribution learned from data (as in our work), or to errors that are learned from data but introduced according to an artificial distribution (as is part of the work of Belinkov and Bisk (2018)).

We consider our work to be complementary to the works of Heigold et al. (2018); Belinkov and Bisk (2018), and Sperber et al. (2017). However, there are several important differences:

1. Belinkov and Bisk (2018) and Sperber et al. (2017) train their NMT systems on fairly small datasets: 235K (Fr-En), 210K (De-En), 122K (Cz-En), and 138K sentences (Es-En) respectively. Even though they use systems like Nematus (Sennrich et al., 2017) or XNMT (Neubig et al., 2018) which generally achieve nearly SOTA results, it is unclear whether their results generalize to larger training data. In contrast, we train our system on almost 2M sentences.
2. All three systems introduce somewhat unrealistic amounts of noise in the data. The natural noise of Belinkov and Bisk (2018) consists of word substitutions based on Wikipedia errors or corrected essays (in the

Work	Errors	Noise Types	NMT level	Languages
(Heigold et al., 2018)	synthetic	character swaps, character flips, word scrambling	char, BPE	De→En
(Sperber et al., 2017)	synthetic	ASR errors	word	Es→En
(Belinkov and Bisk, 2018)	synthetic	character swap, middle scramble, full scramble, keyboard typo	char, BPE	Fr,De,Cz→En
	semi-natural	word substitutions		
this work	semi-natural	grammar errors: article, preposition, noun number, verb agreement	BPE	En→Es
	natural	JFLEG corpus		

Table 8: Previous work on evaluating the effect of noise in NMT systems. Character swaps refer to neighboring character reordering (e.g. noise→nosie), while character flips refer to character substitutions (e.g. noise→noiwe).

Czech case) but they substitute all possible correct words with their erroneous version, ending up with datasets with more than 40% of the tokens being noisy. For that reason, we refer to it as *semi-natural* noise in Table 8. Meanwhile, Sperber et al. (2017) test on the outputs of an ASR system that has a WER of 41.3%. For comparison, in the JFLEG datasets, we calculated that only about 3.5%–5% of the tokens are noisy – the average Levenshtein distance of a corrected reference and its noisy source is 13 characters.

3. The word scrambling noise, albeit interesting, could not be claimed to be applicable to realistic scenarios, especially when applied to all words in a sentence. The solution Belinkov and Bisk (2018) suggested and Sperber et al. (2017) discussed is a character- or spelling-aware model for producing word- or subword-level embeddings. We suspect that such a solution would indeed be appropriate for dealing with typos and other character-level noise, but not for more general grammatical noise. Our method could potentially be combined with GloVe (Pennington et al., 2014) or fastText (Bojanowski et al., 2017) embeddings that can deal with slight spelling variations, but we leave this for future work.

On the other side, Grammar Error Correction has been extensively studied, with significant incremental advances made recently by treating GEC as an MT task: among others, Junczys-Dowmunt and Grundkiewicz (2016) used phrased-based MT, Ji et al. (2017) used hybrid character-word neural sequence-to-sequence systems, Sakaguchi et al. (2017b) used reinforcement learning, and Junczys-Dowmunt et al. (2018) combined several techniques with NMT to achieve the current state-of-the-art. Synthetic errors for training GEC systems have also been studied and applied with mixed success (Rozovskaya and Roth, 2010; Rozovskaya et al., 2014; Xie et al., 2016), while more recently Xie et al. (2018) used backtranslation techniques to add synthetic noise for GEC.

6 Conclusion

In this work, we studied the effect of grammatical errors in NMT. We not only confirmed previous findings, but also expanded on them, showing that *realistic human-like* noise in the form of specific grammatical errors also leads to degraded performance. We added synthetic errors on the English WMT training, dev, and test data (including dev and test sets for all WMT 18 evaluation pairs), and have released them along with the scripts necessary for reproducing them. We also produced Spanish translations of the JFLEG corpus, so that future NMT systems can be properly evaluated on real noisy data.

Acknowledgments

This material is based upon work generously supported by the National Science Foundation under grants 1464553 and 1761548. We are grateful to the anonymous reviewers for their useful comments.

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proc. ICLR*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 Conference on Machine Translation. In *Proc. WMT*, pages 131–198.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proc. ACL*, pages 76–86.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. SYSTRAN's pure neural machine translation systems. arXiv:1610.05540.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS Corpus of Learner English. In *Proc. BEA NLP*, pages 22–31.

Hany Hassan Awadalla, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, Will Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic Chinese to English news translation. ArXiv:1803.05567.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef Genabith. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proc. AMTA*, volume 1, pages 68–80.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2018. The SOCKEYE neural machine translation toolkit at AMTA 2018. *Proc. AMTA*, page 200.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proc. ACL*, pages 753–762.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proc. EMNLP*, pages 1546–1556.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proc. NAACL-HLT*, pages 595–606.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proc. WNMT*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit*, pages 79–86.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proc. WNMT*, pages 28–39.

M Paul Lewis, Gary F Simons, Charles D Fennig, et al. 2009. *Ethnologue: Languages of the world*, volume 16. SIL International.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proc. EACL*, pages 21–27.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proc. EACL*, pages 229–234.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, et al. 2018. XNMT: The eXtensible Neural Machine Translation toolkit. arXiv: 1803.00188.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proc. CoNLL*, pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proc. CoNLL*, pages 1–12.

Toan Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proc. NAACL HLT*, pages 334–343.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proc. NeurIPS Autodiff Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING-ACL*, pages 433–440.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proc. CoNLL*, pages 34–42.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proc. EMNLP*, pages 961–970.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017a. Robsut wrod reocginiton via semi-character recurrent neural network. In *Proc. AAAI*, pages 3281–3287.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017b. Grammatical error correction with neural reinforcement learning. In *Proc. IJCNLP*, pages 366–372.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. In *Proc. EACL*, pages 65–68.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences. In *Proc. IWSLT*.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proc. ACL*, pages 198–202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proc. NeurIPS*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. arXiv: 1603.09727.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proc. NAACL HLT*, pages 619–628.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proc. ACL-HLT*, pages 180–189.

A Results with LSTM models

Training Set	En-Es WMT Test Set						
	CLEAN	DROP	ART	PREP	NN	SVA	AVERAGE ± STDEV
CLEAN	**26.62**	24.08	25.35	25.63	23.34	**26.06**	25.18 ± 1.24
DROP	25.10	24.21	24.24	24.00	22.26	19.58	23.23 ± 2.02
ART	25.49	23.26	24.78	24.35	22.42	25.59	24.31 ± 1.26
PREP	25.49	22.99	24.39	25.22	22.78	25.07	24.32 ± 1.17
NN	25.35	23.04	23.06	24.15	24.73	24.61	24.16 ± 0.94
SVA	25.77	23.49	24.68	24.62	23.22	25.41	24.53 ± 1.01
CLEAN+DROP	**26.45**	**25.37**	25.59	25.59	23.64	25.92	25.43 ± 0.95
CLEAN+ART	**26.64**	24.60	**26.35**	26.08	23.69	26.48	**25.64 ± 1.21**
CLEAN+PREP	**26.60**	24.31	25.12	**26.30**	23.27	**26.14**	25.29 ± 1.31
CLEAN+NN	26.23	23.86	24.75	25.52	**25.20**	25.66	25.20 ± 0.82
CLEAN+SVA	**26.62**	24.22	25.49	25.86	23.79	**26.24**	25.37 ± 1.13
MIX-ALL	**26.60**	24.90	25.52	25.80	24.68	**26.03**	**25.59 ± 0.72**

Table 9: BLEU scores on the WMT test set without (CLEAN) and with synthetic grammar errors using an LSTM encoder-decoder model.

JFLEG-ES Dev

Training	Manual correction					No	Auto
	COR0	COR1	COR2	COR3	avg.	corr.	corr.
CLEAN	**28.3**	27.3	28.4	28.2	28.0	27.1	27.7
MIX-ALL	28.2	**27.5**	**28.8**	**29.1**	**28.4**	**27.4**	**28.2**

JFLEG-ES Test

Training	Manual correction					No	Auto
	COR0	COR1	COR2	COR3	avg.	corr.	corr.
CLEAN	**24.9**	**25.1**	**25.6**	**25.1**	**25.2**	22.8	23.5
MIX-ALL	24.8	25.0	25.3	25.0	25.0	**23.1**	**24.3**

Table 10: BLEU scores on the JFLEG-ES dev and test datasets with the LSTM encoder-decoder model.

Improving Domain Adaptation Translation with Domain Invariant and Specific Information

Shuhao Gu[1,2] **Yang Feng**[1,2]* **Qun Liu**[3]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2]University of Chinese Academy of Science
[3]Huawei Noah's Ark Lab, Hong Kong, China
[1,2]{gushuhao17g, fengyang}@ict.ac.cn
[3]qun.liu@huawei.com

Abstract

In domain adaptation for neural machine translation, translation performance can benefit from separating features into domain-specific features and common features. In this paper, we propose a method to explicitly model the two kinds of information in the encoder-decoder framework so as to exploit out-of-domain data in in-domain training. In our method, we maintain a private encoder and a private decoder for each domain which are used to model domain-specific information. In the meantime, we introduce a common encoder and a common decoder shared by all the domains which can only have domain-independent information flow through. Besides, we add a discriminator to the shared encoder and employ adversarial training for the whole model to reinforce the performance of information separation and machine translation simultaneously. Experiment results show that our method can outperform competitive baselines greatly on multiple data sets.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014; Gehring et al., 2017) has made great progress and drawn much attention recently. Most NMT models are based on the encoder-decoder architecture, where all the sentence pairs share the same set of parameters for the encoder and decoder which makes NMT models have a tendency towards overfitting to frequent observations (e.g., words, word co-occurrences, translation patterns), but overlooking special cases that are not frequently observed. However, in practical applications, NMT models usually need to perform translation for some specific domain with only a small quantity of in-domain training data but a large amount of out-of-domain data. Simply combining in-domain training data with out-of-domain data will lead to overfitting to the out-of-domain data. Therefore, some domain adaptation technique should be adopted to improve in-domain translation.

Fortunately, out-of-domain data still embodies common knowledge shared between domains. And incorporating the common knowledge from out-of-domain data can help in-domain translation. Britz et al. (2017) have done this kind of attempts and managed to improve in-domain translation. The common architecture of this method is to share a single encoder and decoder among all the domains and add a discriminator to the encoder to distinguish the domains of the input sentences. The training is based on adversarial learning between the discriminator and the translation , ensuring the encoder can learn common knowledge across domains that can help to generate target translation. Zeng et al. (2018) extend this line of work by introducing a private encoder to learn some domain specific knowledge. They have proven that domain specific knowledge is a complement to domain invariant knowledge and indispensable for domain adaptation. Intuitively, besides the encoder, the knowledge inferred by the decoder can also be divided into domain specific and domain invariant and further improvement will be achieved by employing private decoders.

In this paper, in order to produce in-domain translation with not only common knowledge but in-domain knowledge, we employ a common encoder and decoder among all the domains and also a private encoder and decoder for each domain separately. The differences between our method and the above methods are in two points: first, we employ multiple private encoders rather where all the domains only have one private encoder; sec-

---

*Corresponding Author

Proceedings of NAACL-HLT 2019, pages 3081–3091
Minneapolis, Minnesota, June 2 - June 7, 2019.

ond, we also introduce multiple private decoders contrast to no private decoder. This architecture is based on the consideration that out-of-domain data is far more than in-domain data and only using one private encoder and/or decoder has the risk of overfitting. Under the framework of our method, the translation of each domain is predicted on the output of both the common decoder and its private decoder. In this way, the in-domain private decoder has direct influence to the generation of in-domain translation and the out-of-domain decoder is used to help train the common encoder and decoder better which can also help in-domain translation. We conducted experiments on English→Chinese and English↔German domain adaptation tasks for machine translation under the framework of RNNSearch (Bahdanau et al., 2014) and Transformer (Vaswani et al., 2017) and get consistently significant improvements over several strong baselines.

2 Related Work

The task of domain adaptation for NMT is to translate a text in-domain for which only a small number of parallel sentences is available. The main idea of the work for domain adaptation is to introduce external information to help in-domain translation which may include in-domain monolingual data, meta information or out-of-domain parallel data.

To exploit in-domain monolingual data, Gülçehre et al. (2015) train a RNNLM on the target side monolingual data first and then use it in decoding. Domhan and Hieber (2017) further extend this work by training the RNNLM part and translation part jointly. Sennrich et al. (2015a) propose to conduct back translation for the monolingual target data so as to generate the corresponding parallel data. Zhang and Zong (2016) employs the self-learning algorithm to generate the synthetic large-scale parallel data for NMT training. To introduce meta information, Chen et al. (2016) use the topic or category information of the input text to assistant the decoder and Kobus et al. (2017) extend the generic NMT models, which are trained on a diverse set of data to, specific domains with the specialized terminology and style.

To make use of out-of-domain parallel data, Luong and Manning (2015) first train an NMT model with a large amount of out-of-domain data, then fine tune the model with in-domain data. Wang et al. (2017a) select sentence pairs from the out-of-domain data set according to their similarity to the in-domain data and then add them to the in-domain training data. Chu et al. (2017) construct the training data set for the NMT model by combining out-of-domain data with the over-sampled in-domain data. Wang et al. (2017b) combine the in-domain and out-of-domain data together as the training data but apply instance weighting to get a weight for each sentence pair in the out-of-domain data which is used in the parameter updating during back propagation. Britz et al. (2017) employ a common encoder to encode the sentences from both the in-domain and out-of-domain data and meanwhile add a discriminator to the encoder to make sure that only domain-invariant information is transferred to the decoder. They focus on the situation that the quantity of the out-of-domain data is almost the same as the in-domain data while our method can handle more generic situations and there is no specific demand for the ratio of the quantity between the in-domain and out-of-domain data. Besides, our method employs a private encoder-decoder for each domain which can hold the domain-specific features. In addition to the common encoder, Zeng et al. (2018) further introduce a domain-specific encoder to each domain together with a domain-specific classifier to ensure the features extracted by the domain-specific encoder is proper. Compared to our method, they focus on the encoder and do not distinguish the information in the decoder.

Adversarial Networks have achieved great success in some areas (Ganin et al., 2016; Goodfellow et al., 2014). Inspired by these work, we also employ a domain discriminator to extract some domain invariant features which has already shown its effectiveness in some related NLP tasks. Chen et al. (2017) use a classifier to exploit the shared information between different Chinese word segment criteria. Gui et al. (2017) tries to learn common features of the out-domain data and in-domain data through adversarial discriminator for the part-of-speech tagging problem. Kim et al. (2017) train a cross-lingual model with language-adversarial training to generate the general information across different languages for the POS tagging problem. All these work try to utilize a discriminator to distinguish invariant features across the divergence.

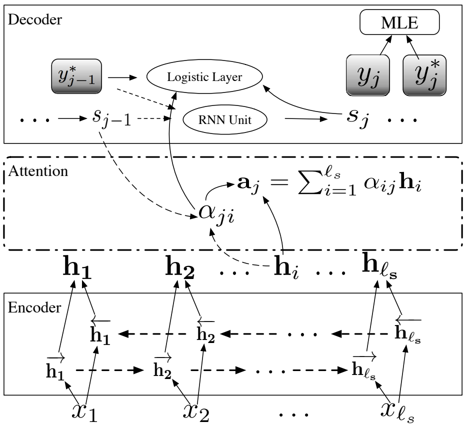


Figure 1: The architecture of the attention-based NMT.

3 RNN-based NMT model

Our method can be applied to both the RNN-based NMT model (Bahdanau et al., 2014) and self-attention-based NMT model (Vaswani et al., 2017). In this paper, we will introduce our method under the RNN-based framework and the application to the self-attention-based framework can be implemented in a similar way. Before introducing our method, we will first briefly describe the RNN-based NMT model with attention shown in Figure 1.

The encoder uses two GRUs to go through source words bidirectionally to get two hidden states $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ for the source word x_i, which are then concatenated to produce the final hidden states for x_i as follows

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i] \tag{1}$$

The attention layer aims to extract the source information which is most related to the generation of each target word. First it evaluates the correlation between the previous decoder hidden state $\mathbf{s}_{j-1}$ and each source hidden state $\mathbf{h}_i$ by

$$e_{ij} = \mathbf{v}_\alpha^T \tanh\left(\mathbf{W}_\alpha \mathbf{s}_{j-1} + \mathbf{U}_\alpha \mathbf{h}_i\right), \tag{2}$$

next calculates α_{ij} which is the correlation degree to each target hidden state $\mathbf{h}_i$, and then gets the attention $\mathbf{c}_j$. The formulation is as follows

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{i'=1}^{l_s}\exp(e_{i'j})}; \quad \mathbf{c}_j = \sum_{i=1}^{l_s} \alpha_{ij}\mathbf{h}_i \tag{3}$$

The decoder also employs a GRU to get the hidden state s_j for the target word y_j as

$$\mathbf{s}_j = g(y_{j-1}, \mathbf{s}_{j-1}, \mathbf{c}_j). \tag{4}$$

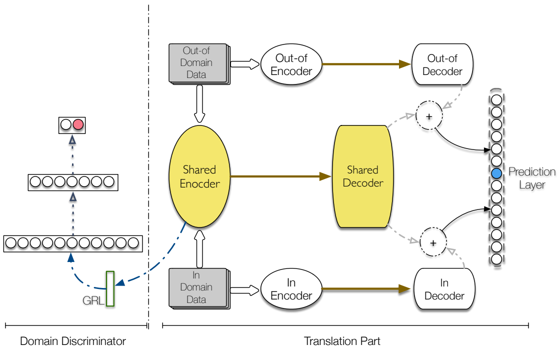


Figure 2: The architecture of the proposed method. GRL means the gradient reversal layer which will multiply a negative constant to the gradients during back-propagation.

Then the probability of the target word y_j is defined as follows

$$p(y_j|\mathbf{s}_j, y_{j-1}, \mathbf{c}_j) \propto \exp(y_j^{\mathsf{T}}\mathbf{W}_o\mathbf{t}_j) \tag{5}$$

where t_j is computed by

$$\mathbf{t}_j = \mathbf{U}_o\mathbf{s}_{j-1} + \mathbf{V}_o\mathbf{E}y_{j-1} + \mathbf{C}_o\mathbf{c}_j \tag{6}$$

4 The Proposed Method

Assume that we have two kinds of training data: out-of-domain and in-domain, and we want to get the translation for the in-domain input. The out-of-domain and in-domain data can be represented as

$$\begin{aligned} \mathbf{out} &= \{(\mathbf{x}^k, \mathbf{y}^{*k})\}_{k=1}^{N_{\mathbf{out}}} \sim \mathcal{D}_{\mathbf{out}}; \\ \mathbf{in} &= \{(\mathbf{x}^k, \mathbf{y}^{*k})\}_{k=1}^{N_{\mathbf{in}}} \sim \mathcal{D}_{\mathbf{in}} \end{aligned} \tag{7}$$

The main idea of our method is to extract domain invariant information from the out-of-domain data to improve in-domain translation. To this end, we employ a common encoder and a common decoder shared by both of the domains, and a private encoder and a private decoder for each domain. The main architecture given in Figure 2.

The working scenario of our method is as follows. When a sentence comes, it is inputted into the shared encoder and the private encoder of the corresponding domain simultaneously. Then the output of the shared encoder is fed into the shared decoder and the output of the private encoder into its corresponding private decoder. Finally, the shared decoder and the private decoder collaborate together to generate the current target word with a gate to decide the contribution ratio.

In addition, our method also introduce a discriminator to distinguish the domain of the input sentence based on the output of the shared encoder. When the discriminator cannot predict the domain of the input sentence, we can think the knowledge encoded in the shared encoder is domain invariant. This is achieved with a gradient reversal layer (GRL) so that the gradients are reversed during back-propagation. In this way, the adversarial training is performed between the translation and the discriminator.

4.1 The Translation Part

The Encoder

Our model has a shared encoder, an in-domain private encoder and an out-of-domain private encoder, where the shared encoder accepts input from the two domains. Given a sentence of domain p (p $\in \{\mathbf{in}, \mathbf{out}\}$), the shared encoder and the private encoder of domain p will roll the sentence as the encoder shown in Section 3 and the outputs of the shared encoder and the private encoder for word x_j are represented as $\mathbf{h}_j^{\mathrm{c}}$ and $\mathbf{h}_j^{\mathrm{p}}$ respectively.

The Attention Layer

As the output of the shared encoder is only fed to the shared decoder and the output of the private encoder of domain p only flows to the private decoder of domain p, we only need to calculate the attention of the shared decoder over the shared encoder and the attention of the private decoder of domain p over the private encoder of domain p. We calculate these two attentions as in Section 3 and denote them as $\mathbf{c}_j^{\mathrm{c}}$ and $\mathbf{c}_j^{\mathrm{p}}$ for the shared decoder and the private decoder, respectively.

The Decoder

We also maintain a shared decoder, an in-domain private decoder and an out-of-domain private decoder For a sentence of domain p (p $\in \{\mathbf{in}, \mathbf{out}\}$), the shared decoder and the private decoder of domain p act in the same way as shown in Equation 4 and Equation 6 and then produce the hidden states $\mathbf{s}_j^{\mathrm{c}}$ and $\mathbf{t}_j^{\mathrm{c}}$ for the shared decoder, and $\mathbf{s}_j^{\mathrm{p}}$ and $\mathbf{t}_j^{\mathrm{p}}$ for the private decoder.

To predict the target word y_j, $\mathbf{t}_j^{\mathrm{c}}$ and $\mathbf{t}_j^{\mathrm{p}}$ are weighted added to get $\mathbf{t}_j$ as

$$\begin{aligned} z_j &= \sigma(\mathbf{W}_z \mathbf{t}_j^{\mathrm{c}} + \mathbf{U}_z \mathbf{t}_j^{\mathrm{p}}); \\ \mathbf{t}_j &= z_j \cdot \mathbf{t}_j^{\mathrm{c}} + (1 - z_j) \cdot \mathbf{t}_j^{\mathrm{p}} \end{aligned} \tag{8}$$

Where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{W}_z$ and $\mathbf{U}_z$ are shared by in-domain and out-of-domain. Finally the probability of the target word y_j is computed with

$$P(y_j | \dots) \propto \exp(y_j^{\mathsf{T}} \mathbf{W}_o \mathbf{t}_j); \tag{9}$$

4.2 The Domain Discriminator

The domain discriminator acts as a classifier to determine the knowledge encoded in the shared encoder is from in-domain or from out-of-domain. When a well trained discriminator can't classify the domain properly, we can think the knowledge in the shared encoder is domain invariant (Ganin et al., 2016). As CNN has shown its effectiveness in some related classification tasks (Zhang et al., 2015; Yu et al., 2017), we construct our discriminator with CNN.

First, the input to the CNN is the representation of the whole source sentence which is got by concatenating the sequence of hidden states generated by the shared encoder as

$$\Pi_{1:I} = \mathbf{h}_1 \oplus \mathbf{h}_2 \oplus \cdots \oplus \mathbf{h}_I \tag{10}$$

where I is the length of the source sentence and $\mathbf{h}_1, ..., \mathbf{h}_I$ is the hidden state of the corresponding source word. $\oplus$ stands for the concatenation operation of the hidden states, and we can get the final source sentence representation $\Pi_{1:I} \in \mathbb{R}^{I \times m}$ where m is the dimension of the hidden state.

We then employ a kernel $w \in \mathbb{R}^{l \times m}$ to apply a convolutional operation to produce a new feature map:

$$\mathbf{f} = \rho(w \otimes \Pi_{1:I} + b_f) \tag{11}$$

where ρ is the ReLU activation function, $\otimes$ stands for the convolutional operation of the kernel and b is the bias term. A number of different kinds of kernels with different windows sizes are used in our work to extract different features at different scales. Next, we apply a max-over-time pooling operation over the feature maps to get a new feature map. To further improve the performance of the discriminator, following the work (Yu et al., 2017), we also add the highway architecture (Srivastava et al., 2015; Zhang et al., 2018) behind the pooled feature maps where we use a gate to control the information flow between the two layers. Finally, the combined feature map is fed into a fully connected network with a sigmoid activation function to make the final predictions:

$$p(d) \propto \exp(\mathbf{W}_d \cdot \mathbf{f} + b_d); \tag{12}$$

where d is the domain label of in-domain or out-of-domain.

5 Training

Our final loss considers the translation loss and the domain prediction loss. For the translation loss, we employ cross entropy to maximize the translation probability of the ground truth, so we have this loss as follows and the training objective is to minimize the loss.

$$\mathcal{L}_{\mathbf{MT}} = -\sum_{k=1}^{N_{\mathbf{in}}+N_{\mathbf{out}}} \sum_{j=1}^{J^k} \log p(y_j^{*k}) \tag{13}$$

where $N_{\mathbf{in}}$ and $N_{\mathbf{out}}$ are the number of training sentences for in-domain and out-of-domain data respectively, J^k is the length of the k-th ground truth sentence, and $p(y_j^{*k})$ is the predicted probability of the j-th word for the k-th ground truth sentence.

Note that we have three different encoders and three different decoders in total, including the shared encoder and decoder, the in-domain private encoder and decoder, and the out-of-domain private encoder and decoder, and all of them have their own parameters.

For the domain prediction loss, we also use cross-entropy to minimize the following loss

$$\mathcal{L}_{\mathfrak{D}} = -\sum_{k=1}^{N_{\mathbf{in}}+N_{\mathbf{out}}} \log p(d^{*k}) \tag{14}$$

where d^{*k} is the ground truth domain label of the k-th input sequence.

Then the final loss is defined as

$$\mathcal{L} = \mathcal{L}_{\mathbf{MT}} + \lambda \mathcal{L}_{\mathfrak{D}} \tag{15}$$

where λ is a hyper-parameter to balance the effects of the two parts of loss. We gradually tried λ from 0.1 to 2.5 and set it to 1.5 in our final experiments.

Borrowing ideas from Ganin et al. (2016), we introduce a special gradient reversal layer (GRL) between the shared encoder and the domain discriminator. During forward propagation, the GRL has no influence to the model, while during back-propagation training, it multiplies a certain negative constant to the gradients back propagated from the discriminator to the shared encoder. In this way, an adversarial learning is applied between the translation part and the discriminator.

At the beginning of the training, we just use the $\mathcal{L}_{\mathbf{MT}}$ to train the translation part on the combined data, including the shared encoder-decoder and the in-domain and out-of-domain private encoder-decoder. Then we use $\mathcal{L}_{\mathfrak{D}}$ to only train the domain discriminator until the precision of the discriminator reach 90% while the parameters of the shared encoder keep unupdated. Finally, we train the whole model with the complete loss $\mathcal{L}$ with all the parameters updated. In the training process, the sentences in each batch is sampled from in-domain and out-of-domain data at the same rate.

During testing, we just use the shared encoder-decoder and the private in-domain encoder-decoder to perform in-domain translation.

6 Experiments

We evaluated our method on the English→Chinese (En-Zh), English→German (En-De) and German→English (De-En) domain adaptation translation task.

6.1 Data Preparation

English→Chinese For this task, out-of-domain data is from the LDC corpus[1] that contains 1.25M sentence pairs. The LDC data is mainly related to the **News** domain. We chose the parallel sentences with the domain label **Laws** from the UM-Corpus (Tian et al., 2014) as our in-domain data. We chose 109K, 1K and 1K sentences from the UM-Corpus randomly as our training, development and test data. We tokenized and lowercased the English sentences with Moses[2] scripts. For the Chinese data, we performed word segmentation using Stanford Segmenter[3].

English→German For this task, the training data is from the Europarl corpus distributed for the shared domain adaptation task of WMT 2007 (Callison-Burch et al., 2007) where the out-of-domain data is mainly related to the **News** domain, containing about 1.25M sentence pairs, and in-domain data is mainly related to the **News Commentary** domain which is more informal compared to the news corpus, containing about 59.1K sentences. We also used the development set of the domain adaptation shared task. Finally, we tested our method on the NC test set of WMT

---

[1] https://www.ldc.upenn.edu/

[2] http://www.statmt.org/moses/

[3] https://nlp.stanford.edu/

2006 and WMT 2007. We tokenized and lowercased the corpora.

German→English For this task, out-of-domain corpus is from the WMT 2015 en-de translation task which are mainly **News** texts (Bojar et al., 2015) containing about 4.2M sentence pairs. For the in-domain corpus, we used the parallel training data from the IWSLT 2015 which is mainly from the the **TED talks** containing about 190K sentences. In addition, dev2012 and test2013/2014/2015 of IWSLT 2015 were selected as the development and test data, respectively. We tokenized and truecased the corpora.

Besides, 16K, 16K and 32K merging operations were performed to learn byte-pair encoding(BPE) (Sennrich et al., 2015b) on both sides of the parallel training data and sentences longer than 50, 50 and 80 tokens were removed from the training data, respectively.

6.2 Systems

We implemented the baseline and our model by PyTorch framework[4]. For the En-Zh and En-De translation task, batch size was set to 80 and vocabulary size was set to 25k which covers all the words in the training set. The source and target embedding sizes were both set to 256 and the size of the hidden units in the shared encoder-decoder RNNs was also set to 256. During experiments, we found that the shared encoder-decoder played a major role in the model and the size of the private encoder-decoder didn't influence the results too much. Thus we just set the size of the private encoder-decoder one-quarter of the shared encoder-decoder considering the training and decoding speed.

For the De-En translation task, batch size was set to 40 and vocabulary size was set to 35K in the experiment. The source and target embedding sizes were both set to 620 and the size of the hidden units in the shared encoder and decoder RNNs was set to 1000. As mentioned before, the size of the private encoder-decoder was just one-quarter of the shared encoder-decoder.

All the parameters were initialized by using uniform distribution over $[-0.1, 0.1]$. The adadelta algorithm was employed to train the model. We reshuffled the training set between epochs. Besides, the beam size was set to 10.

Contrast Methods We compared our model with the following models, namely:

- **In** : This model was trained only with the in-domain data.
- **Out + In** : This model was trained with both of the in-domain and out-of-domain data.
- **Sampler** (Chu et al., 2017) : This method over-sampled the in-domain data and concatenated it with the out-of-domain data.
- **Fine Tune** (Luong and Manning, 2015) : This model was trained first on the out-of-domain data and then fine-tuned using the in-domain data.
- **Domain Control (DC)** (Kobus et al., 2017) : This method extend word embedding with an arbitrary number of cells to encode domain information.
- **Discriminative Mixing (DM)** (Britz et al., 2017) : This method adds a discriminator on top of the encoder which is trained to predict the correct class label of the input data. The discriminator is optimized jointly with the translation part.
- **Target Token Mixing (TTM)** (Britz et al., 2017) : This method append a domain token to the target sequence.
- **Adversarial Discriminative Mixing(ADM)** (Britz et al., 2017) : This method is similar with our model which also add a discriminator to extract common features across domains. The biggest difference is that we add private parts to preserve the domain specific features. Besides we also applied a different training strategy as the section 5 describes so that our method can handle more generic situations.

Noting that our model has a private encoder-decoder which brings extra parameters, we just slightly extend the hidden size of the contrast model to make sure that the total parameter number of the contrast model is equal to the number of our model's translation part.

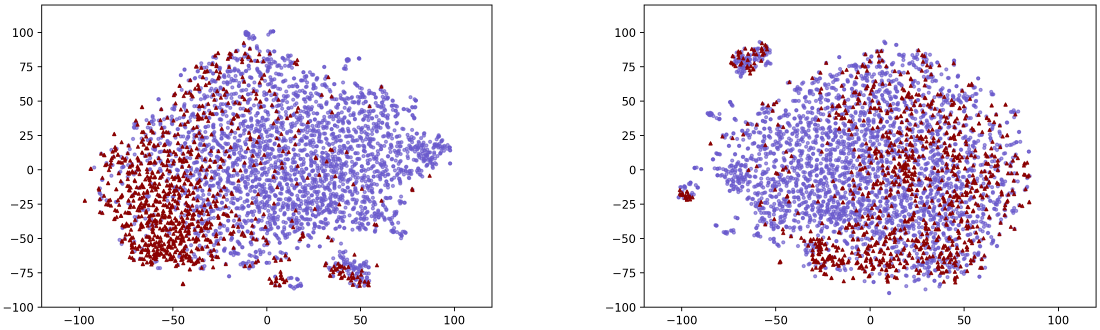
6.3 Main Results

The **En-Zh Experiments** Results are measured using char based 5-gram BLEU score (Papineni et al., 2002) by the *multi-bleu.pl* script. The main results are shown in Table 1. On both of the development set and test set, our model significantly outperforms the baseline models and other contrast models. Furthermore, we got the following conclusions:

First, the baseline model 'In' surpass the 'Out + In' model which shows that the NMT model tends to fit out-of-domain features if we directly include

---

[4] http://pytorch.org



(a) without discriminator

(b) full model

Figure 3: The shared encoder's hidden state of the two models. Data from the out-of-domain are presented as blue dots while data from the in-domain are presented as red triangles. There is an obvious separation of the results of the without discriminator model but the hidden states of the shared encoder of our full model are well-distributed.

En-Zh	dev	test	average
In	32.45	30.42	31.44
Out + In	30.37	28.76	29.57
Sampler	35.06	32.97	34.02
Fine Tune	35.02	33.36	34.19
DC	31.08	29.59	30.34
DM	30.98	29.73	30.36
TTM	31.77	30.11	30.94
ADM	31.23	29.88	30.56
our method	36.55**	34.84**	35.70

Table 1: Results of the en-zh translation experiments. The marks indicate whether the proposed methods were significantly better than the best performed contrast models(**: better at significance level $\alpha = 0.01$, *:$\alpha = 0.05$)(Collins et al., 2005)

out-of-domain data into the in-domain data, as the domain specific features embodied in out-of-domain data is in greater quantity than that in in-domain data. However, we also found that the model will over fit so soon if we only use the in-domain data so it is necessary to make use of the out-of-domain data to improve the translation performance.

Second, we found that when the in-domain data is much less than the out-of-domain data, some contrast methods for domain adaptation, such as DC, DM TTM and ADM, didn't perform well. They were worse than the baseline model 'in' and only slightly better than 'out + in'. These methods all try to take domain information into translation in their own ways which actually brings improvement compared with the 'out + in' model. However, as the out-of-domain data is much more than the in-domain data, the model will still tends to fit out-of-domain data and ignore the in-domain information which will degrade the final performance. Therefore, it is necessary to handle the in-domain data separately in some way. The 'Sampler' and 'Fine Tune' perform better because they receive much more information from the in-domain data compared with other methods, but they don't make use of the domain information when translating.

En-De	test06	test07	average
In	23.36	25.00	24.18
Out + In	20.69	22.43	21.56
Sampler	26.83	29.01	27.92
Fine Tune	27.02	29.19	28.11
our method	27.97*	30.67**	29.32

Table 2: Results of the WMT 07 en-de translation experiments.

Last, our model achieves the best performance among all the contrast models. The shared encoder extract the domain invariant features of the two domains with the help of the discriminator so that the shared part will be well trained using all the in-domain and out-of-domain data. At the meantime, we also consider the domain specific features and the private encoder-decoder can re-

De-En	test13	test14	test15
In	25.83	21.97	24.64
Out + In	26.45	23.21	25.85
Sampler	29.70	25.71	28.29
Fine Tune	30.48	26.55	28.62
Sennrich et al. (2015a)	28.20	24.40	26.70
Wang et al. (2017b)	28.58	24.12	-
our method	30.99	26.94	29.30*

Table 3: Results of the IWSLT 15 en-de experiments. The second part results were directly taken from their papers.

ceive enough information from the in-domain data to prevent the whole model from overfitting the out-of-domain features.

The **En-De Experiments** and **De-En Experiments** results are shown in the Table 2 and Table 3. Results are measured using word based 4-gram BLEU score (Papineni et al., 2002) by the *multi-bleu.pl* script. In these two experiments, we only compared our method with the baseline model and the competitive contrast methods 'Sampler' (Kobus et al., 2017) and 'Fine Tune' (Luong and Manning, 2015). Similar to the previous experiment results, our method still achieves the best performance compared to all contrast models, which demonstrates again that our model is effective and general to different language pairs and different domains.

6.4 Experiment Analysis

We made some some detailed analysis to empirically show the effectiveness of our model based on En-Zh translation task.

6.4.1 Ablation Study

In order to further understand the impact of the components of the proposed model, we performed some further studies by training multiple versions of our model by removing some components: The first model removed the domain discriminator but preserved the private part. The second one removed the private encoder-decoder but kept the domain discriminator. The last one just removed both of those two parts.

Results are shown in the Table 4. As expected, the best performance is obtained with the simultaneous use of all the tested elements. When we removed the private encoder-decoder, the result shows that the score was reduced by 0.79, which indicates that our private part can preserve some useful domain specific information which is abandoned by the shared encoder. When we removed the discriminator, the result was reduced by 0.76. This result supports our idea that modeling common features from out-of-domain data can benefit in-domain translation. When we removed both of the two components, we got the lowest score. The total result shows that every component of our model plays an important role in our model.

DCN	Private	dev	test	average
√	√	36.55	34.84	35.70
√	×	35.73	34.09	34.91
×	√	35.67	34.22	34.94
×	×	35.13	33.36	34.25

Table 4: Results of the ablation study. "DCN" means the discriminator and "Private" means the private encoder-decoder.

	Out-of-Domain	In-Domain	distance
- DCN	(12.9, 5.8)	(-38.9,-17.5)	56.8
Full Model	(-4.4, 3.0)	(13.2,-9.0)	21.3

Table 5: The average coordinates value and its distance of the the hidden states. '- DCN' is the model without domain discriminator.

6.4.2 Impact of the Discriminator

To verify whether the discriminator have learned the domain invariant knowledge, we did the following experiments using model **without discriminator** and our **full model** with the domain discriminator of the former subsection.

We sampled 3000 sentences randomly from the out-of-domain and 1000 sentences from the in-domain En-Zh parallel sentences as the test data. Then they were fed into the shared encoders of the two models to get the reshaped feature maps as the Equation 11 describes. Next, we used the t-Distributed Stochastic Neighbor Embedding(t-SNE)[5] technique to do a dimensionality reduction to the hidden state. Results are shown in the Figure 3. We also calculate the average value of the coordinates of each domain's hidden state. The results are shown in Table 5

---

[5] http://lvdmaaten.github.io/tsne/

En-Zh	test1	test2	test3
Out + In	22.31	18.82	17.59
Sampler	21.60	18.64	16.93
Fine Tune	13.18	11.94	11.55
our method	22.61	19.36	17.78

Table 6: Results of the out-of-domain translation task. The test sets are from the NIST test sets but we exchange the translation directions.

En-Zh	dev	test	average
In	32.61	30.33	31.47
Sampler	35.84	33.68	34.76
Fine Tune	36.01	34.03	35.02
our method	37.26**	35.39**	36.33

Table 7: Results of the En-Zh experiments based on the transformer model.

From the figure, we can find that here is an obvious separation in the results of the model without discriminator and the numerical analysis also support this point, which indicates that the shared encoder without the help of discriminator will treat the data from different domains differently. All the domain shared features and domain specific features are just mixed together. On the contrary, the output of the shared encoder of our full model is well distributed. This proves that the discriminator can help the shared encoder to extract domain invariant features which then help to improve the translation performance in in-domain.

6.4.3 Out-of-domain Translation Performance

Despite the fact that the purpose of our work is to improve the in-domain translation performance, the domain invariant features extracted from the training data are also beneficial to the out-of-domain translation performance. To prove this, we use the NIST 03 04 and 05 test sets which are mainly related to the **News** domain as our out-of-domain test set. Noting that the origin set was designed for the Zh-En translation task and each sentence has four English references, we just chose the first reference as the source side sentence for our En-Zh translation task. The results are shown in the Table 6 We can conclude from the results that the "Fine Tune" method suffered a catastrophic forgetting caused by parameter shift during the training process. On the contrary, our method can achieve a mild improvement on the out-of-domain compared to the baseline system.

6.4.4 Combined With Transformer Model

Transformer (Vaswani et al., 2017) is an efficient NMT architecture. To test the generality of our method, we also conducted relevant experiments based on the transformer model. We did the experiment based on the Fairseq code[6]. The implementation on this translation framework is similar with the way on the RNN based models. The encoder and decoder of our final model consist 3 sublayers. The number of the multi-head attention was set to 4 and the embedding dim was set to 256. We also compared with the 'Sampler' and 'Fine Tune' method based on transformer. The results are shown in 7. According to the table, our method still outperforms than other models, which can prove that our method has a good generality across different translation architecture.

7 Conclusions

In this paper, we present a method to make use of out-of-domain data to help in-domain translation. The key idea is to divide the knowledge into domain invariant and domain specific. The realization way is to employ a shared encoder-decoder to process domain invariant knowledge and a private encoder-decoder for each domain to process knowledge of the corresponding domain. In addition, a discriminator is added to the shared encoder and adversarial learning is applied to make sure the shared encoder can learn domain invariant knowledge. We conducted experiments on multiple data sets and get consistent significant improvements. We also verified via experiments that the shared encoder, the domain specific private encoder-decoder and the discriminator all make contribution to the performance improvements.

Acknowledgements

We thank the three anonymous reviewers for their comments, Jinchao Zhang, Wen Zhang for suggestions. This work was supported by the National Natural Science Foundation of China (NSFC) under the project NO.61876174, NO.61662077 and NO.61472428.

---

[6] https://fairseq.readthedocs.io/en/latest/index.html

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *WMT@EMNLP*.

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158. Association for Computational Linguistics.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420.

Caglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loıc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR, abs/1503.03535*, 15.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *LREC*, pages 1837–1842.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 560–566.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

Wen Zhang, Jiawei Hu, Yang Feng, and Qun Liu. 2018. Refining source representations with relation networks for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1292–1303.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Selective Attention for Context-aware Neural Machine Translation

Sameen Maruf[†][*] **André F. T. Martins**[‡] **Gholamreza Haffari**[†]
[†]Faculty of Information Technology, Monash University, VIC, Australia
[‡]Unbabel & Instituto de Telecomunicações, Lisbon, Portugal
[†]`{firstname.lastname}@monash.edu`
[‡]`andre.martins@unbabel.com`

Abstract

Despite the progress made in sentence-level NMT, current systems still fall short at achieving fluent, good quality translation for a full document. Recent works in context-aware NMT consider only a few previous sentences as context and may not scale to entire documents. To this end, we propose a novel and scalable top-down approach to hierarchical attention for context-aware NMT which uses sparse attention to selectively focus on relevant sentences in the document context and then attends to key words in those sentences. We also propose single-level attention approaches based on sentence or word-level information in the context. The document-level context representation, produced from these attention modules, is integrated into the encoder or decoder of the Transformer model depending on whether we use monolingual or bilingual context. Our experiments and evaluation on English-German datasets in different document MT settings show that our selective attention approach not only significantly outperforms context-agnostic baselines but also surpasses context-aware baselines in most cases.

1 Introduction

Neural machine translation has grown immensely in the past few years, from the simplistic RNN-based encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to the state-of-the-art Transformer architecture (Vaswani et al., 2017). Most of these models rely on the attention mechanism as a major component, which involves focusing on different parts of a sequence to compute new representations, and has proven to be quite effective in improving the translation quality (Vaswani et al., 2017). However, all of these models share the same inherent problem: the translation is still performed on a sentence-by-sentence basis, thus ignoring the long-range dependencies which may be useful when it comes to translating discourse phenomena.

More recently, **context-aware NMT** has been gaining significant traction from the MT community with majority of works coming out in the past two years. Most of these focus on using a few previous sentences as context (Jean et al., 2017; Wang et al., 2017; Tu et al., 2018; Voita et al., 2018; Zhang et al., 2018; Miculicich et al., 2018) and neglect the rest of the document. Only one existing work has endeavoured to consider the full document context (Maruf and Haffari, 2018), thus proposing a more generalised approach to document-level NMT. However, the model is restrictive as the document-level attention computed is sentence-based and static (computed only once for the sentence being translated). A more recent work (Miculicich et al., 2018) proposes to use a hierarchical attention network (HAN) (Yang et al., 2016) to model the contextual information in a structured manner using word-level and sentence-level abstractions; yet, it uses a limited number of past source and target sentences as context and is not scalable to entire document.

In this work, we propose a **selective attention** approach to first selectively focus on relevant sentences in the global document-context and then attend to key words in those sentences, while ignoring the rest.[1] Towards this goal, we use **sparse attention**, enabling an efficient and scalable use of the context. The intuition behind this is the way humans translate a sentence containing ambiguous words. They may look for sentences in the whole document which contain similar words and just focus on those for the translation. This attention,

---

[*] Work initiated during an internship at Unbabel.

[1] The term "selective attention" comes from cognitive science and is defined as the act of focusing on a particular object for a period of time while simultaneously ignoring irrelevant information that is also occurring (Dayan et al., 2000).

Proceedings of NAACL-HLT 2019, pages 3092–3102
Minneapolis, Minnesota, June 2 - June 7, 2019.

which we call Hierarchical Attention, is computed dynamically for each query word. Furthermore, we propose a Flat Attention approach which is based on either sentence or word-level information in the context. We integrate the document-level context representation, produced from these attention modules, into the encoder or decoder of the Transformer model depending on whether we consider monolingual (source-side) or bilingual (both source and target-side) context.

Our contributions are as follows: (i) we propose a novel and efficient top-down approach to hierarchical attention for context-aware NMT, (ii) we compare variants of selective attention with both context-agnostic and context-aware baselines, and (iii) we run experiments in both online (only past context) and offline (both past and future context) settings on three English-German datasets. Experiments show that our approach improves upon the Transformer by an overall +1.34, +2.06 and +1.18 BLEU for TED Talks, News-Commentary and Europarl, respectively. It also outperforms two recent context-aware baselines (Zhang et al., 2018; Miculicich et al., 2018) in majority of the cases.

2 Background

2.1 Neural Machine Translation

Generic NMT models are based on an encoder-decoder architecture (Bahdanau et al., 2015; Vaswani et al., 2017). The encoder reads the source sentence denoted by $\boldsymbol{x} = (x_1, x_2, ..., x_M)$ and maps it to a continuous representation $\boldsymbol{z} = (z_1, z_2, ..., z_M)$. Given $\boldsymbol{z}$, an attentional decoder generates the target translation $\boldsymbol{y} = (y_1, y_2, ..., y_N)$ one word at a time in a left-to-right fashion. The popular Transformer architecture (Vaswani et al., 2017) follows the same structure by using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder.

Encoder The encoder stack is composed of L identical layers, each containing two sub-layers. The first, a multi-head self-attention sub-layer, allows each position in the encoder to attend to all positions in the previous layer of the encoder, while the second, a feed-forward network, uses two linear transformations with a ReLU activation.

Decoder The decoder stack is also composed of L identical layers. In addition to the two sub-layers, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Masking is used in the self-attention sub-layer to prevent positions from attending to subsequent positions thus avoiding leftward flow of information.

2.2 Document-level Machine Translation

In general, the probability of a document translation $\boldsymbol{Y}$ given the source document $\boldsymbol{X}$ is given by:

$$P_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J} P_{\boldsymbol{\theta}}(\boldsymbol{y}^j|\boldsymbol{x}^j, \boldsymbol{D}_{-j}) \qquad (1)$$

where $\boldsymbol{y}^j$ and $\boldsymbol{x}^j$ denote the j^{th} target and source sentence, respectively, and $\boldsymbol{D}_{-j} = \{\boldsymbol{X}_{-j}, \boldsymbol{Y}_{-j}\}$ is the collection of all other sentences in the source and target document. Since generic NMT models translate one word at a time, Eq. 1 becomes:

$$P_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J}\prod_{n=1}^{N} P_{\boldsymbol{\theta}}(y_n^j|\boldsymbol{y}_{<n}^j, \boldsymbol{x}^j, \boldsymbol{D}_{-j}) \quad (2)$$

where y_n^j is the n^{th} word of the j^{th} target sentence and $\boldsymbol{y}_{<n}^j$ are the previously generated words.

Training The document-conditioned NMT model $P_{\boldsymbol{\theta}}(\boldsymbol{y}^j|\boldsymbol{x}^j, \boldsymbol{D}_{-j})$ is realised using a neural architecture and usually trained via a two-step procedure (Maruf and Haffari, 2018; Miculicich et al., 2018). The first step involves pre-training a standard sentence-level NMT model, and the second step involves optimising the parameters of the whole model, i.e., both the document-level and the sentence-level parameters.

Decoding To generate the best translation for a full document according to the document MT model, the problem of maximizing Eq. 1 is solved using a two-pass Iterative Decoding strategy (Maruf and Haffari, 2018): first, the translation of each sentence is initialised using the sentence-based NMT model; then, each translation is updated using the context-aware NMT model fixing the other sentences' translations.

3 Proposed Approach

The main goal of this paper is to have a document-level NMT model which is memory-efficient, scalable, and capable of listening to the entire document. To achieve this, we augment a sentence-level NMT model (the Transformer (Vaswani et al., 2017)) with an efficient hierarchical attention mechanism which has the ability to identify the key sentences in the document context and then attend to the key words within those sentences. As mentioned previously, we want to maximise $P_{\boldsymbol{\theta}}(\boldsymbol{y}^j|\boldsymbol{x}^j, \boldsymbol{D}_{-j})$, where we take $\boldsymbol{D}_{-j}$ to be either the monolingual source or bilingual source and target-side context in two settings: *offline*—the context comes from both past and future, and *online*—the context comes from only the past.

In this section, we show how to represent the document-level context using our Context Layer, how to regulate the information at the sentence and document-level using context gating and finally we present our integrated model.

3.1 Document-level Context Layer

The context $\boldsymbol{D}_{-j}$ is modeled via a single Document-level Context Layer comprising of two sub-layers: (i) a Multi-Head Context Attention sub-layer, and (ii) a Feed-Forward sub-layer, where the former consists of either a top-down Hierarchical Attention module or a Flat Attention module (explained shortly), and the latter is similar to the Feed-Forward network in the original Transformer architecture. Each sub-layer is followed by a layer normalisation.[2]

Let us now describe the attention modules which independently form the Multi-Head Context Attention sub-layer.

3.1.1 Hierarchical Attention

Our hierachical attention module H-Attention(Q_s, Q_w, K_s, K_w, V_w) (Figure 1) is a reformulation of the Scaled Dot-Product Attention of Vaswani et al. (2017). Here, we have five inputs consisting of two types of keys and queries, one each for the sentences and the words, while the values are based only on words in the context. The Hierarchical Attention module has four operations:

1. **Sentence-level Key Matching:** This is performed on a set of queries simultaneously, packed together into a matrix Q_s. The sentence-level keys are also packed into a matrix K_s. We will describe in §3.3 how Q_s and K_s are computed. The attention weights are computed as:

$$\alpha_s = \text{sparsemax}(Q_s {K_s}^T / \sqrt{d_k}) \qquad (3)$$

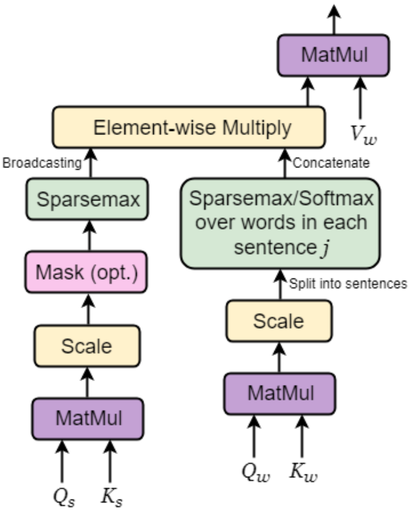


Figure 1: Hierarchical Context Attention module.

where d_k is the dimension of the keys, and α_s has dimensions equal to the total number of sentences in the document. We propose to use **sparsemax** (Martins and Astudillo, 2016), instead of softmax, as this gives us the intended selective attention behavior, that is identifying the key sentences that may potentially be relevant to the current sentence, hence making the model more efficient in compressing its memory. A softmax attention, on the other hand, can still assign low probability to sentences, forming a long-tail and absorbing significant probability mass, and it cannot fully *ignore* those sentences. An additive mask is used (before the *sparsemax* operation) based on whether we train for offline or online setting by masking out only the current sentence or current and future sentences, respectively.

2. **Word-level Key Matching:** Here the query and key matrices, Q_w and K_w, are word-level. We perform a word-level key matching for each sentence j in the document:

$$\alpha_w^j = \text{sparsemax}(Q_w {K_w^j}^T / \sqrt{d_k}) \qquad (4)$$

where α_w^j is the word-level attention vector for j^{th} sentence.[3] We can also use softmax, instead of sparsemax, for a coarser key matching. We explore the two variants in our experiments.

3. **Re-scaling attention weights:** The word-level attention is further re-weighted by the cor-

---

[2] We do not have residual connections after sub-layers in our Document-level Context Layer as we found them to have a deteriorating effect on the translation scores (also reported by Zhang et al. (2018)).

[3] This can be done for only the sentences with non-zero probabilities (obtained from the sentence-level key matching), however, we found it to be computationally expensive, as it required breaking down the batched matrices.

responding sentence-level attention (Nallapati et al., 2016) such that the probability of j^{th} sentence in a document is given by:

$$\alpha^{j}_{hier} = \alpha_s(j)\alpha^{j}_{w} \tag{5}$$

where $\alpha_s(j)$ is the attention weight for the j^{th} sentence obtained via Eq. 3 and α^{j}_{w} is as in Eq. 4. The re-weighting, thus, produces a scaled attention vector $\alpha_{hier} = \text{Concat}(\alpha^{1}_{hier}, ..., \alpha^{J}_{hier})$, each entry of which corresponds to the attention weight for a specific word in the document.

4. **Value Reading:** The set of word-level values is packed together into a matrix V_w and the matrix of outputs is given by $\alpha_{hier}V_w$. This multiplication, combined with sparsemax attention, allows to *prune* the hierarchy.

We further extend the MULTIHEAD attention function proposed by Vaswani et al. (2017) for our Hierarchical Attention module as:

$$\text{H-MULTIHEAD}(Q_s, K_s, Q_w, K_w, V_w) = \text{Concat}(head_1, ..., head_H)W^O$$

where $head_h = \text{H-Attention}(Q_sW_h^{Q_s}, Q_wW_h^{Q_w}, K_sW_h^{K_s}, K_wW_h^{K_w}, V_wW_h^{V_w})$, W's are parameter matrices and all (five) inputs are transformed using separate linear layers.

3.1.2 Flat Attention

Another way to model the context $\boldsymbol{D}_{-j}$ is via single-level attention by re-using the Scaled Dot-Product Attention in Vaswani et al. (2017),

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V \tag{6}$$

The attention[4] here is of two types: (i) *sentence-level* if K, V are computed for sentences in the document, or (ii) *word-level* if K, V are computed for words in the document. The former module is similar to the Memory Networks architecture of Maruf and Haffari (2018) in that it uses sentence-level information. However, there are two key differences: (i) we use MultiHead attention as in the Transformer architecture, and (ii) our context attention is dynamic such that we have a separate attention for each query word.

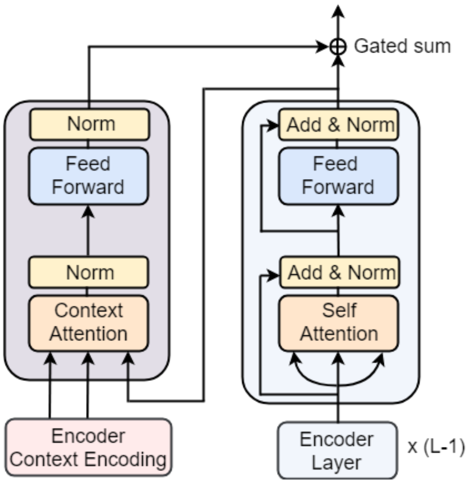


Figure 2: Encoder-side context integration.

3.2 Context Gating

As mentioned previously, the Multi-Head Context Attention sub-layer is part of the Context Layer (Figure 2), the output of which is fed into the Transformer architecture through context gating (Tu et al., 2018). For i^{th} word in source or target:

$$\gamma_i = \sigma(W_r r_i + W_d d_i) \tag{7}$$

$$\tilde{r}_i = \gamma_i \odot r_i + (1 - \gamma_i) \odot d_i \tag{8}$$

where W's are parameter matrices, r_i is the output of encoder or decoder stack for i^{th} word, d_i is the output from the context layer for i^{th} word and $\tilde{r}_i$ is the final hidden representation for the same.

3.3 Integrated Model

The context can be integrated into the encoder or decoder of the NMT model depending on if it is monolingual or bilingual.[5]

Monolingual context integration in Encoder We add the Document-level Context Layer alongside the encoder stack as shown in Figure 2. The Encoder Context Encoding block stores the keys and values produced from the pre-trained sentence-level NMT model. For word-level attention, the keys K_w and values V_w are composed of vector representations (from last encoder layer) of source words in the document, while for the sentence-level attention, the keys K_s and values V_s are composed of vector representations of sentences in the document where the vector representation of each sentence is an average of the word

---

[4] We plan to investigate sparse flat attention in future work.

[5] We do not integrate context into both encoder and decoder as it would have redundant information from the source (the context incorporated in the decoder is bilingual), in addition to increasing the complexity of the model.

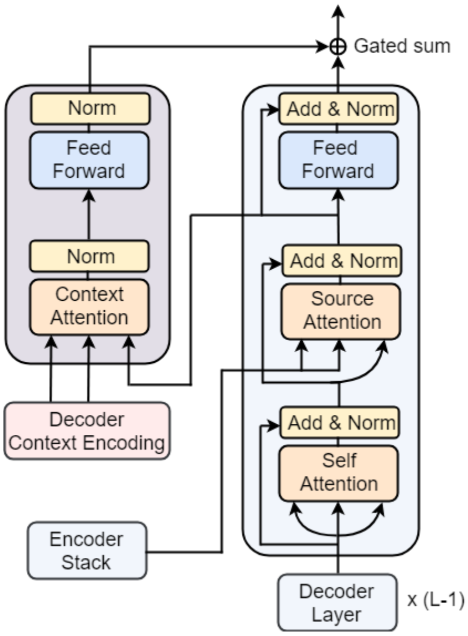


Figure 3: Decoder-side context integration.

representations in that sentence. The queries Q_w, Q_s are linear transformations of the output of the L^{th} encoder layer which are then matched with the corresponding keys and values stored in the Encoder Context Encoding block just described.

Bilingual context integration in Decoder We again add the Document-level Context Layer alongside the decoder stack as in Figure 3. However, instead of choosing the keys and values to be monolingual as in the encoder, we follow Tu et al. (2018) in choosing the key to match to the source-side context, while designing the value to match to the target-side context. Hence, the keys (in the Decoder Context Encoding block) are composed of context vectors from the Source Attention sub-layer, while the values are composed of the hidden representations of the target words, both from the last decoder layer. Again the keys K_w and K_s are either for individual target words or target sentences, and same goes for V_w and V_s. The queries Q_w, Q_s for the Context Layer come from the Source Attention sub-layer in the L^{th} layer of the decoder (Figure 3).

4 Experiments

4.1 Setup

Datasets We conduct experiments for English→German on three different domains: TED talks, News-Commentary and Europarl. These datasets are chosen based on their variance in genre, style and level of formality:

Domain	#Sentences	Document length
TED	0.21M/9K/2.3K	120.89/96.42/98.74
News	0.24M/2K/3K	38.93/26.78/19.35
Europarl	1.67M/3.6K/5.1K	14.14/14.95/14.06

Table 1: Training/development/test corpora statistics: number of sentences (K stands for thousands and M for millions), and average document length (in sentences).

- **TED** This corpus is from the IWSLT 2017 MT track (Cettolo et al., 2012) and contains transcripts of TED talks aligned at sentence level. Each talk is considered to be a document. We combine *tst2016-2017* into the test set and the rest are used for development.
- **News-Commentary** We obtain the sentence-aligned document-delimited News Commentary v11 corpus for training.[6] The WMT'16 *news-test2015* and *news-test2016* are used for development and testing, respectively.
- **Europarl** This dataset is extracted from Europarl v7 (Koehn, 2005). The source and target sentences are aligned using the links provided by Tiedemann (2012). Following Maruf and Haffari (2018), we use the *SPEAKER* tag as the document delimiter. Documents longer than 5 sentences are kept and the resulting corpus is randomly split into training, dev and test sets.

The corpora statistics are provided in Table 1. All datasets are tokenised and truecased using the Moses toolkit (Koehn et al., 2007), and split into subword units using a joint BPE model with 30K merge operations (Sennrich et al., 2016).

Models and Baselines For offline document MT, we have two context-agnostic baselines: (i) a modified version of RNNSearch (Bahdanau et al., 2015), which incorporates dropout on the output layer and improves the attention model by feeding the previously generated word, and (ii) the state-of-the-art Transformer architecture. For the online case, we again have the Transformer as a context-agnostic baseline and two context-aware baselines (Zhang et al., 2018; Miculicich et al., 2018).

All models are implemented in C++ using DyNet (Neubig et al., 2017). For RNNSearch, we modify the sentence-based NMT implementation in *mantis* (Cohn et al., 2016). The encoder is a single layer bidirectional GRU (Cho et al., 2014) and

---

[6] `www.casmacat.eu/corpus/news-commentary.html`

	Integration into Encoder						Integration into Decoder					
	TED		News		Europarl		TED		News		Europarl	
Model	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**
RNNSearch	19.24	40.81	16.51	36.79	26.26	44.14	19.24	40.81	16.51	36.79	26.26	44.14
Transformer	23.28	44.17	22.78	42.19	28.72	46.22	23.28	44.17	22.78	42.19	28.72	46.22
+Attention, sentence	24.47	**45.25**	**24.78**	43.90	29.60	46.98	**24.38**	44.82	24.67	43.82	29.67	**47.04**
word	**24.55**	44.89	24.55	43.75	29.63	46.94	24.27	44.95	24.23	43.44	29.68	46.93
+H-Attention, sparse-soft	24.23	44.81	24.76	44.10	**29.72**	47.03	24.19	44.94	**24.67**	**43.86**	**29.69**	46.97
sparse-sparse	24.27	45.07	24.66	**44.18**	29.64	**47.04**	24.14	**45.32**	24.49	43.49	29.59	47.02

Table 2: BLEU and Meteor scores for variants of our model and two context-agnostic baselines for offline document MT. **bold**: Best performance. All reported results for our model are significantly better than both baselines.

	Integration into Encoder						Integration into Decoder					
	TED		News		Europarl		TED		News		Europarl	
Model	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**	**BLEU**	**Meteor**
Zhang et al. (2018)	24.00	44.69	23.08	42.40	29.32	46.72	23.82	44.54	22.78	42.17	29.35	46.73
Miculicich et al. (2018)	**24.58**	**45.48**	**25.03**	44.02	28.60	46.09	24.39	45.23	24.38	43.58	29.58	46.91
Transformer	23.28	44.17	22.78	42.19	28.72	46.22	23.28	44.17	22.78	42.19	28.72	46.22
+Attention, sentence	24.38	45.01	24.46*	43.46*	29.59$^{\clubsuit}$	47.02$^{\clubsuit}$	24.29*	45.13*	**24.75**$^{\clubsuit}$	**44.03**$^{\clubsuit}$	29.56	46.84
word	24.22	45.05*	24.84*	**44.27***	29.67$^{\clubsuit}$	47.04$^{\clubsuit}$	24.02	44.79	24.17*	43.53*	**29.90**$^{\clubsuit}$	47.11$^{\clubsuit}$
+H-Attention, sparse-soft	24.34	45.05*	24.54*	43.66*	**29.75**$^{\clubsuit}$	**47.22**$^{\clubsuit}$	**24.62***	**45.32***	24.36*	43.67*	29.80*	**47.11**$^{\clubsuit}$
sparse-sparse	24.42	45.38*	24.73*	44.06*	29.39$^{\diamondsuit}$	46.78$^{\diamondsuit}$	24.43*	45.10*	24.58*	43.75*	29.64*	46.94*

Table 3: BLEU and Meteor scores for variants of our model and three baselines for online document MT. **bold**: Best performance. $\star$, $\diamondsuit$, $\clubsuit$: Statistically significantly better than our implementations of Zhang et al. (2018), Miculicich et al. (2018), or both. All reported results for our model are significantly better than the Transformer.

the decoder is a 2-layer GRU with embeddings and hidden dimensions set to 512. The dropout rate for the output layer is set to 0.2. For the Transformer, we use *Transformer-DyNet*[7] implementation and extend it for our context-aware NMT model.[8] The hidden dimensions and feed-forward layer size is set to 512 and 2048 respectively. We use 4 layers[9] each in the encoder and decoder with 8 attention heads and employ label smoothing with a value of 0.1. We also employ all four types of dropouts as in the original Transformer with a rate of 0.1 for the sentence-based model and 0.2 for our context-aware model.

For training all models, we use the default Adam optimiser (Kingma and Ba, 2015) with an initial learning rate of 0.0001 and employ early stopping. For our context-aware NMT model, we use a two-stage training strategy as described in §2.2. For inference, we use Iterative Decoding only when using the bilingual context. All experiments are run on a single Nvidia P100 GPU with 16GBs of memory.[10]

Evaluation Metrics For evaluation, we use BLEU (Papineni et al., 2002) and Meteor (Lavie and Agarwal, 2007) scores on tokenised text, and measure statistical significance with respect to the baselines, $p < 0.05$ (Clark et al., 2011).

4.2 Main Results

We divide our experiments into two parts: offline and online document MT.

Offline Document MT From the scores of the two context-agnostic baselines in Table 2, we can see that the Transformer beats the RNNSearch model in all cases by atleast +2.5 BLEU and +2.1 Meteor scores showing that our hyperparameter choice for the Transformer is indeed effective.

For the Encoder Context integration, our Hierarchical Attention models perform the (near) best for News and Europarl datasets with +1.98 and +1 BLEU and +1.99 and +0.82 Meteor improvements with respect to the Transformer. For TED talks, however, we find the Flat Attention based models (sentence and word-level) to be the best with +1.27 BLEU and +1.08 METEOR improvements. For Decoder Context integration, we find the Hierarchical Attention to be the best in majority of the cases both in terms of BLEU and Meteor.

---

[7] `https://github.com/duyvuleo/Transformer-DyNet`

[8] The code is available at `https://github.com/sameenmaruf/selective-attn`

[9] We found this configuration to be much more stable than using 6 layers with almost no difference in performance as reported by Xia et al. (2018).

[10] The experiments can also be run on GPUs with 10-12GBs of memory by reducing the batch size at the expense of increased computational cost.

Online Document MT From Table 3, all our models significantly outperform the context-agnostic baseline and are significantly better than Zhang et al. (2018) in majority cases. For Encoder Context integration, the HAN encoder (Miculicich et al., 2018) is the best for TED and News datasets, however, the results are statistically insignificant with respect to our best model. For Europarl, our Hierarchical Attention model performs significantly better than Miculicich et al. (2018) with a gain of +1.15 BLEU and +1.13 Meteor. For Decoder Context integration, our Hierachical Attention models are the winner in majority cases and our best models beat Miculicich et al. (2018) for all datasets based upon BLEU and Meteor. The main conclusion we draw from these results is that efficiently using the context information at hand is crucial when it comes to improving the performance of context-aware NMT. Furthermore, shorter pieces of text (e.g., the ones in Europarl) benefit more from using global context because their sentences may exhibit higher interdependency than those in a longer piece of text.

Offline vs. Online Document MT Let us compare the overall results for the offline and online document MT settings. For all datasets and model variants, we find the best BLEU and Meteor scores in Tables 2 and 3 (highlighted in bold) to be quite close to each other with those for the online setting slightly better. This is quite self-explanatory, because in essence, all of the datasets comprise of talks, speeches or commentaries, which are in fact produced in an online manner and hence we do not see drastic improvements in terms of BLEU and Meteor when conditioning on the future context. This, in our opinion, does not mean that we should never look into the future, but just that NMT models in general are highly subjective to data, and whether context-aware models benefit from future context is also dependent on that.

4.3 Analysis

Evaluation on Contrastive Pronoun Test Set It has been argued that evaluation metrics which quantify the overall translation quality are somewhat ill-equipped to assess how well models translate inter-sentential phenomena such as pronouns. Hence, we use a test suite of contrastive translations designed to measure accuracy of translating the English pronoun *it* to its German counterparts *es*, *er* and *sie* (Müller et al., 2018). We are interested to see if our global document-context models surpass the local context-aware baselines. Table 4 shows that not only our global-context models are quite effective but our Hierarchical Attention model is most useful when the antecedent is farther than three previous sentences. We also conclude that models for offline MT perform better when antecedent distance is greater than two.

Model	**antecedent distance**				
	0	1	2	3	>3
	Offline document MT				
RNNSearch	0.415	0.310	0.424	0.440	0.647
Transformer	0.586	0.308	0.437	0.48	0.642
+Attention, sentence	0.677	0.314	0.439	0.478	0.697
word	**0.686**	**0.347**	**0.464**	**0.511**	0.679
+H-Attention, sparse-soft	0.676	0.308	0.440	0.480	0.686
sparse-sparse	0.652	0.303	0.435	0.471	**0.701**
	Online document MT				
Zhang et al. (2018)	0.622	0.321	0.450	0.485	0.658
Miculicich et al. (2018)	0.722	0.326	0.451	0.471	0.661
Transformer	0.586	0.308	0.437	0.48	0.642
+Attention, sentence	**0.732**	**0.340**	**0.460**	0.485	0.661
word	0.690	0.317	0.444	0.487	0.683
+H-Attention, sparse-soft	0.692	0.329	0.446	0.464	0.656
sparse-sparse	0.711	0.317	0.437	**0.489**	**0.692**

Table 4: Accuracy on contrastive test set with regard to antecedent distance (in sentences) on TED Talks. Antecedent distance 0 means the pronoun occurs in the same sentence as the antecedent.

Subjective Evaluation We conduct a subjective evaluation to validate the benefit of exploiting document-level context. Three native German speakers were asked to choose the better (with ties allowed) of two translations for each of 18 documents (randomly sampled from Europarl test set). The two translations, one produced by the Transformer and the other by our Hierarchical Attention model, were evaluated in terms of: *adequacy* (Which translation expresses the meaning of the source text more adequately?) and *fluency* (Which text has better German?) (Läubli et al., 2018). Let a, b be number of ratings in favour of Transformer or our model, respectively, and t be number of ties, then number of successes $x = b + 0.5t$ and trials $n = a + b + t$. We test for statistically significant preference of our model over the Transformer by means of two-sided Sign Tests and find that our model is better than the Transformer both in terms of document-level adequacy ($x = 39$, $n = 54$, $p = 0.0015$) and fluency ($x = 38$, $n = 54$, $p = 0.0038$).

Model Complexity Model complexity is reported in Table 5. Our context-aware models introduce only 8% more parameters to the original

Model	#Params	Speed (words/sec.)	
		Training	Decoding
Zhang et al. (2018)	59.5M	3300	84.94
Miculicich et al. (2018)	54.8M	1650	76.90
Transformer	50M	5100	86.33
+Attention, sentence	53.7M	3750	83.84
+H-Attention, sparse-soft	54.2M	2600	74.11

Table 5: Model complexity for Encoder Context integration models (News-Commentary).

Transformer model. In comparison to the Transformer, our Hierarchical Attention model is slow in training, dropping the speed by almost 50%[11], but it is still almost 40% faster than Miculicich et al. (2018). At decoding time, our Hierarchical Attention model is almost equivalent to Miculicich et al. (2018) and only 13% slower than Zhang et al. (2018). Hence, attending to the whole document (instead of few previous sentences) does not add to the time complexity of the model on average.

Qualitative Analysis To analyse the effect of using sparse attention at both the sentence and word-level, we looked at the attention weights computed by *sparsemax*. Table 6 shows an example where our model helped generate a correct translation of the noun "thoughts" (highlighted in bold). The context sentences shown in the bottom box had the highest attention weights as assigned by sparsemax. It seems that this particular attention head focuses more on phrases like "words of sympathy", "support', "symbol of hope" which are related to the query "thoughts". Another example in Table 7 shows how our model correctly translates the pronoun "their". Upon looking at the words in the context sentences, it seems that this particular attention head focuses on the words related to the antecedent "Croatia's Serbian population" with most of the weight concentrated around neighbouring words in sentence s^{j-1}. It is evident from both examples that word-level sparsity is more prevalent in longer sentences in the context. The same holds for sparsity at sentence-level.

5 Related Work

The body of work in document-level MT can be broadly classified into two categories: conventional MT and neural MT.

Src: my **thoughts** are also with the victims .
Ref: meine **Gedanken** sind auch bei den Opfern .
Transformer: ich **denke** auch an die Opfer .
Zhang et al. (2018): ich **denke** auch an die Opfer .
Miculicich et al. (2018): ich **denke** auch an die Opfer .
Our Model: meine **Gedanken** sind auch bei den Opfern .

Head 2: Attention to related words *sympathy, support, hope*
s^{j-2}: (FR) Madam President , many things have already been said , but I would like to echo all the words of sympathy and support that have already been addressed to the peoples of Tunisia and Egypt .
s^{j+4}: it must implement a strong strategy towards these countries .
s^{j-1}: they are a symbol of hope for all those who defend freedom .

Table 6: Example of noun disambiguation. Source context sentences are ordered in decreasing probability mass. The intensity of color corresponds to the attention given to a specific word before rescaling.

Src: Croatia is **their** homeland , too .
Ref: Kroatien ist auch **ihre** Heimat .
Transformer: Kroatien ist auch **seine** Heimat .
Our Model: Kroatien ist auch **ihr** Heimatland .

Head 8: Attention to words related to the antecedent.
s^{j-1}: to name but a few , these include cooperation with the Hague Tribunal , efforts made so far in prosecuting corruption , restructuring the economy and finances and greater commitment and sincerity in eliminating the obstacles to the return of Croatia 's Serbian population .
s^{j-4}: by signing a border arbitration agreement with its neighbour Slovenia , the new Croatian Government has not only eliminated an obstacle to the negotiating process , but has also paved the way for the resolution of other issues .

Table 7: Example of pronoun disambiguation. Context sentences are ordered in decreasing probability mass.

Conventional Document-level MT These can further be classified into two main categories. The first, which use cache-based memories (Tiedemann, 2010; Gong et al., 2011) and the second, which focus on specific discourse phenomema like anaphora (Hardmeier and Federico, 2010), lexical cohesion (Xiong et al., 2013; Gong et al., 2015; Mascarell, 2017) and coreference (Miculicich Werlen and Popescu-Belis, 2017) to name a few. Most of these approaches are, however, restrictive as they mostly involve using handcrafted features similar to the conventional MT approaches.

Document-level Neural MT The works here can again be divided into two categories: *online*—use previous context only, and *offline*—use both past and future contexts. Most works fall into the former category, with those that use only a single

---

[11] DyNet implementation of *sparsemax* is CPU-based and only operates on column vectors. We believe a GPU-based matrix implementation would bring the speed much closer to our Word Attention model (training: 3100, decoding: 81.38).

previous sentence in the source (Jean et al., 2017; Tiedemann and Scherrer, 2017; Voita et al., 2018); one previous sentence both in source and target (Bawden et al., 2018); more than one previous source sentence (Wang et al., 2017; Zhang et al., 2018); or a few previous source and target sentences (Miculicich et al., 2018). Apart from fixing the context length, there are few works which use cache-based memories to store contextual information (Tu et al., 2018; Kuang et al., 2018) and use that to improve the MT system performance. A recent work (Maruf et al., 2018) reports promising results when using the complete history for translating online conversations.

For the offline setting, however, there is only one work that effectively uses the full document-context on both source and target-side using memory networks (Maruf and Haffari, 2018). The debate in document-level NMT today is mostly about how much of the previous context to use and there has been no comparison between the online and offline setting except using only one previous and following sentence (Voita et al., 2018).

Sparse Attention Sparse attention and its constrained variants have been used to address the coverage problem in NMT (Malaviya et al., 2018) by limiting the amount of attention that each source word can receive. Apart from NMT, sparse attention has been shown to yield promising results for NLP tasks of textual entailment (Martins and Astudillo, 2016) and summarization (Niculae and Blondel, 2017).

6 Conclusion

We have proposed a novel approach to hierarchical attention for context-aware NMT, based on sparse attention, which is both scalable and efficient. Experiments and evaluation on three English→German datasets in offline and online document MT settings show that our approach surpasses context-agnostic and two recent context-aware baselines. The qualitative analysis shows that the sparsity at sentence-level allows our model to identify key sentences in the document context and the sparsity at word-level allows it to focus on key words in those sentences allowing for an efficient compression of memory. In future work, we plan to dig deeper on the benefits of sparse attention in terms of better interpretability of context-aware NMT models.

Acknowledgments

The authors are grateful to the anonymous reviewers for their helpful comments and corrections. SM would like to thank her colleagues at Monash University: Veronika Kuchta, Harald Bögeholz and Hagen Lauer, for their help in the subjective evaluation. This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (`www.massive.org.au`), the European Research Council (ERC StG DeepSPIN 758969), the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal), and a Google Faculty Research Award to GH.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Short Papers)*, pages 176–181. Association for Computational Linguistics.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the North American Chapter of* *the Association for Computational Linguistics: Human Language Technologies*, pages 876–885. Association for Computational Linguistics.

Peter Dayan, Sham Kakade, and P. Read Montague. 2000. Learning and selective attention. *Nature Neuroscience*, 3:1218–1223.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919. Association for Computational Linguistics.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2015. Document-level machine translation evaluation with gist consistency and text cohesion. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 33–40, Lisbon, Portugal. Association for Computational Linguistics.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *International Workshop on Spoken Language Translation*, pages 283–289.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? In *arXiv:1704.05135*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Conference Proceedings: the 10th Machine Translation Summit*, pages 79–86. AAMT.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2018. Modeling coherence for neural machine translation with dynamic and topic caches. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 596–606. Association for Computational Linguistics.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? A case for document-level evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 370–376.

André F. T. Martins and Ramón Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1614–1623, New York, New York, USA. PMLR.

Sameen Maruf and Gholamreza Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284. Association for Computational Linguistics.

Sameen Maruf, André F. T. Martins, and Gholamreza Haffari. 2018. Contextual neural model for translating bilingual multi-speaker conversations. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 101–112, Brussels, Belgium. Association for Computational Linguistics.

Laura Mascarell. 2017. Lexical chains meet word embeddings in document-level statistical machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 99–109. Association for Computational Linguistics.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954.

Lesly Miculicich Werlen and Andrei Popescu-Belis. 2017. Using coreference links to improve spanish-to-english machine translation. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 30–40, Valencia, Spain. Association for Computational Linguistics.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 61–72, Belgium, Brussels. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of Conference on Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3338–3348. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54^{th} Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112. MIT Press.

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 8–15, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2816–2821. Association for Computational Linguistics.

Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-level dual learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5379–5388.

Deyi Xiong, Yang Ding, Min Zhang, and Chew Lim Tan. 2013. Lexical chain based cohesion models for document-level statistical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1563–1573. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 533–542. Association for Computational Linguistics.

On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models

Paul Michel[1], **Xian Li**[2], **Graham Neubig**[1], **Juan Miguel Pino**[2]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Facebook AI
`{pmichel1,gneubig}@cs.cmu.edu,`
`{xianl,juancarabina}@fb.com`

Abstract

Adversarial examples — perturbations to the input of a model that elicit large changes in the output — have been shown to be an effective way of assessing the robustness of sequence-to-sequence (seq2seq) models. However, these perturbations only indicate weaknesses in the model if they do not change the input so significantly that it legitimately results in changes in the expected output. This fact has largely been ignored in the evaluations of the growing body of related literature. Using the example of untargeted attacks on machine translation (MT), we propose a new evaluation framework for adversarial attacks on seq2seq models that takes the semantic equivalence of the pre- and post-perturbation input into account. Using this framework, we demonstrate that existing methods may not preserve meaning in general, breaking the aforementioned assumption that source side perturbations should not result in changes in the expected output. We further use this framework to demonstrate that adding additional constraints on attacks allows for adversarial perturbations that are more meaning-preserving, but nonetheless largely change the output sequence. Finally, we show that performing untargeted adversarial training with meaning-preserving attacks is beneficial to the model in terms of adversarial robustness, without hurting test performance. [1]

1 Introduction

Attacking a machine learning model with adversarial perturbations is the process of making changes to its input to maximize an adversarial goal, such as mis-classification (Szegedy et al., 2013) or mis-translation (Zhao et al., 2018). These attacks provide insight into the vulnerabilities of machine learning models and their brittleness to samples outside the training distribution. Lack of robustness to these attacks poses security concerns to safety-critical applications, *e.g.* self-driving cars (Bojarski et al., 2016).

Adversarial attacks were first defined and investigated for computer vision systems (Szegedy et al. (2013); Goodfellow et al. (2014); Moosavi-Dezfooli et al. (2016) inter alia), where the input space is continuous, making minuscule perturbations largely imperceptible to the human eye. In discrete spaces such as natural language sentences, the situation is more problematic; even a flip of a single word or character is generally perceptible by a human reader. Thus, most of the mathematical framework in previous work is not directly applicable to discrete text data. Moreover, there is no canonical distance metric for textual data like the ℓ_p norm in real-valued vector spaces such as images, and evaluating the level of semantic similarity between two sentences is a field of research of its own (Cer et al., 2017). This elicits a natural question: *what does the term "adversarial perturbation" mean in the context of natural language processing (NLP)*?

We propose a simple but natural criterion for adversarial examples in NLP, particularly untargeted[2] attacks on seq2seq models: *adversarial examples should be meaning-preserving on the source side, but meaning-destroying on the target side*. The focus on explicitly evaluating meaning preservation is in contrast to previous work on adversarial examples for seq2seq models (Belinkov and Bisk, 2018; Zhao et al., 2018; Cheng et al., 2018; Ebrahimi et al., 2018a). Nonetheless, this feature is extremely important; given two sentences with equivalent meaning, we would expect a good model to produce two outputs with

---

[1] A toolkit implementing our evaluation framework is released at `https://github.com/pmichel31415/teapot-nlp`.

[2] Here we use the term untargeted in the same sense as (Ebrahimi et al., 2018a): an attack whose goal is simply to decrease performance with respect to a reference translation.

Proceedings of NAACL-HLT 2019, pages 3103–3114
Minneapolis, Minnesota, June 2 - June 7, 2019.

equivalent meaning. In other words, any meaning-preserving perturbation that results in the model output changing drastically highlights a fault of the model.

A first technical contribution of this paper is to lay out a method for formalizing this concept of meaning-preserving perturbations (§2). This makes it possible to evaluate the effectiveness of adversarial attacks or defenses either using gold-standard human evaluation, or approximations that can be calculated without human intervention. We further propose a simple method of imbuing gradient-based word substitution attacks (§3.1) with simple constraints aimed at increasing the chance that the meaning is preserved (§3.2).

Our experiments are designed to answer several questions about meaning preservation in seq2seq models. First, we evaluate our proposed "source-meaning-preserving, target-meaning-destroying" criterion for adversarial examples using both manual and automatic evaluation (§4.2) and find that a less widely used evaluation metric (chrF) provides significantly better correlation with human judgments than the more widely used BLEU and METEOR metrics. We proceed to perform an evaluation of adversarial example generation techniques, finding that chrF does help to distinguish between perturbations that are more meaning-preserving across a variety of languages and models (§4.3). Finally, we apply existing methods for adversarial training to the adversarial examples with these constraints and show that making adversarial inputs more semantically similar to the source is beneficial for robustness to adversarial attacks and does not decrease test performance on the original data distribution (§5).

2 A Framework for Evaluating Adversarial Attacks

In this section, we present a simple procedure for evaluating adversarial attacks on seq2seq models. We will use the following notation: x and y refer to the source and target sentence respectively. We denote x's translation by model M as y_M. Finally, $\hat{x}$ and $\hat{y}_M$ represent an adversarially perturbed version of x and its translation by M, respectively. The nature of M and the procedure for obtaining $\hat{x}$ from x are irrelevant to the discussion below.

2.1 The Adversarial Trade-off

The goal of adversarial perturbations is to produce failure cases for the model M. Hence, the evaluation must include some measure of the *target similarity* between y and y_M, which we will denote $\mathrm{s}_{\mathrm{tgt}}(y, \hat{y}_M)$. However, if no distinction is being made between perturbations that preserve the meaning and those that don't, a sentence like "he's very *friendly*" is considered a valid adversarial perturbation of "he's very *adversarial*", even though its meaning is the opposite. Hence, it is crucial, when evaluating adversarial attacks on MT models, that the discrepancy between the original and adversarial input sentence be quantified in a way that is sensitive to meaning. Let us denote such a *source similarity* score $\mathrm{s}_{\mathrm{src}}(x, \hat{x})$.

Based on these functions, we define the *target relative score decrease* as:

$$\mathrm{d}_{\mathrm{tgt}}(y, y_M, \hat{y}_M) = \begin{cases} 0 \text{ if } \mathrm{s}_{\mathrm{tgt}}(y, \hat{y}_M) \geq \mathrm{s}_{\mathrm{tgt}}(y, y_M) \\ \frac{\mathrm{s}_{\mathrm{tgt}}(y, y_M) - \mathrm{s}_{\mathrm{tgt}}(y, \hat{y}_M)}{\mathrm{s}_{\mathrm{tgt}}(y, y_M)} \text{ otherwise} \end{cases} \tag{1}$$

The choice to report the *relative* decrease in $\mathrm{s}_{\mathrm{tgt}}$ makes scores comparable across different models or languages[3]. For instance, for languages that are comparatively easy to translate (*e.g.* French-English), $\mathrm{s}_{\mathrm{tgt}}$ will be higher in general, and so will the gap between $\mathrm{s}_{\mathrm{tgt}}(y, y_M)$ and $\mathrm{s}_{\mathrm{tgt}}(y, \hat{y}_M)$. However this does not necessarily mean that attacks on this language pair are more effective than attacks on a "difficult" language pair (*e.g.* Czech-English) where $\mathrm{s}_{\mathrm{tgt}}$ is usually smaller.

We recommend that both $\mathrm{s}_{\mathrm{src}}$ and $\mathrm{d}_{\mathrm{tgt}}$ be reported when presenting adversarial attack results. However, in some cases where a single number is needed, we suggest reporting the attack's *success* $\mathcal{S} := \mathrm{s}_{\mathrm{src}} + \mathrm{d}_{\mathrm{tgt}}$. The interpretation is simple: $\mathcal{S} > 1 \Leftrightarrow \mathrm{d}_{\mathrm{tgt}} > 1 - \mathrm{s}_{\mathrm{src}}$, which means that the attack has destroyed the target meaning ($\mathrm{d}_{\mathrm{tgt}}$) more than it has destroyed the source meaning ($1 - \mathrm{s}_{\mathrm{src}}$).

Importantly, this framework can be extended beyond strictly meaning-preserving attacks. For example, for targeted keyword introduction attacks (Cheng et al., 2018; Ebrahimi et al., 2018a), the same evaluation framework can be used if $\mathrm{s}_{\mathrm{tgt}}$ (resp. $\mathrm{s}_{\mathrm{src}}$) is modified to account for the presence (resp. absence) of the keyword (or its translation in the source). Similarly this can be extended to other

---

[3] Note that we do not allow negative $\mathrm{d}_{\mathrm{tgt}}$ to keep all scores between 0 and 1.

tasks by adapting s_{tgt} (*e.g.* for classification one would use the zero-one loss, and adapt the success threshold).

2.2 Similarity Metrics

Throughout §2.1, we have not given an exact description of the semantic similarity scores s_{src} and s_{tgt}. Indeed, automatically evaluating the semantic similarity between two sentences is an open area of research and it makes sense to decouple the definition of adversarial examples from the specific method used to measure this similarity. In this section, we will discuss manual and automatic metrics that may be used to calculate it.

2.2.1 Human Judgment

Judgment by speakers of the language of interest is the *de facto* gold standard metric for semantic similarity. Specific criteria such as adequacy/fluency (Ma and Cieri, 2006), acceptability (Goto et al., 2013), and 6-level semantic similarity (Cer et al., 2017) have been used in evaluations of MT and sentence embedding methods. In the context of adversarial attacks, we propose the following 6-level evaluation scheme, which is motivated by previous measures, but designed to be (1) symmetric, like Cer et al. (2017), (2) and largely considers meaning preservation but at the very low and high levels considers fluency of the output[4], like Goto et al. (2013):

How would you rate the similarity between the meaning of these two sentences?

0. The meaning is completely different or one of the sentences is meaningless
1. The topic is the same but the meaning is different
2. Some key information is different
3. The key information is the same but the details differ
4. Meaning is essentially equal but some expressions are unnatural
5. Meaning is essentially equal and the two sentences are well-formed English[a]

[a] Or the language of interest.

2.2.2 Automatic Metrics

Unfortunately, human evaluation is expensive, slow and sometimes difficult to obtain, for example in the case of low-resource languages. This makes automatic metrics that do not require human intervention appealing for experimental research. This section describes 3 evaluation metrics commonly used as alternatives to human evaluation, in particular to evaluate translation models.[5]

BLEU: (Papineni et al., 2002) is an automatic metric based on n-gram precision coupled with a penalty for shorter sentences. It relies on exact word-level matches and therefore cannot detect synonyms or morphological variations.

METEOR: (Denkowski and Lavie, 2014) first estimates alignment between the two sentences and then computes unigram F-score (biased towards recall) weighted by a penalty for longer sentences. Importantly, METEOR uses stemming, synonymy and paraphrasing information to perform alignments. On the downside, it requires language specific resources.

chrF: (Popović, 2015) is based on the character n-gram F-score. In particular we will use the chrF2 score (based on the F2-score — recall is given more importance), following the recommendations from Popović (2016). By operating on a sub-word level, it can reflect the semantic similarity between different morphological inflections of one word (for instance), without requiring language-specific knowledge which makes it a good one-size-fits-all alternative.

Because multiple possible alternatives exist, it is important to know which is the best stand-in for human evaluation. To elucidate this, we will compare these metrics to human judgment in terms of Pearson correlation coefficient on outputs resulting from a variety of attacks in §4.2.

3 Gradient-Based Adversarial Attacks

In this section, we overview the adversarial attacks we will be considering in the rest of this paper.

3.1 Attack Paradigm

We perform gradient-based attacks that replace one word in the sentence so as to maximize an adversarial loss function $\mathcal{L}_{\mathrm{adv}}$, similar to the substitution attacks proposed in (Ebrahimi et al., 2018b).

---

[4] This is important to rule out nonsensical sentences and distinguish between clean and "noisy" paraphrases (*e.g.* typos, non-native speech…). We did not give annotators additional instruction specific to typos.

[5] Note that other metrics of similarity are certainly applicable within the overall framework of §2.2.1, but we limit our examination in this paper to the three noted here.

Original	**Pourquoi** faire cela ?
English gloss	**Why** do this?
Unconstrained	**construisant** (English: building) faire cela ?
kNN	**interrogez** (English: interrogate) faire cela ?
CharSwap	**Puorquoi** (typo) faire cela ?
Original	Si seulement je pouvais me muscler **aussi** rapidement.
English gloss	If only I could build my muscle **this** fast.
Unconstrained	Si seulement je pouvais me muscler **etc** rapidement.
kNN	Si seulement je pouvais me muscler **plsu** (typo for "more") rapidement.
CharSwap	Si seulement je pouvais me muscler **asusi** (typo) rapidement.

Table 1: Examples of different adversarial inputs. The substituted word is highlighted.

3.1.1 General Approach

Precisely, for a word-based translation model M[6], and given an input sentence $w_1, \ldots, w_n$, we find the position i^* and word w^* satisfying the following optimization problem:

$$\underset{1\leq i\leq n, \hat{w}\in\mathcal{V}}{\arg\max}\ \mathcal{L}_{\text{adv}}(w_0, \ldots, w_{i-1}, \hat{w}, w_{i+1}, \ldots, w_n) \tag{2}$$

where $\mathcal{L}_{\text{adv}}$ is a differentiable function which represents our adversarial objective. Using the first order approximation of $\mathcal{L}_{\text{adv}}$ around the original word vectors $\mathbf{w}_1, \ldots, \mathbf{w}_n$[7], this can be derived to be equivalent to optimizing

$$\underset{1\leq i\leq n, \hat{w}\in\mathcal{V}}{\arg\max}\ [\hat{\mathbf{w}} - \mathbf{w}_i]^\intercal \nabla_{\mathbf{w}_i} \mathcal{L}_{\text{adv}} \tag{3}$$

The above optimization problem can be solved by brute-force in $\mathcal{O}(n|\mathcal{V}|)$ space complexity, whereas the time complexity is bottlenecked by a $|\mathcal{V}| \times d$ times $n \times d$ matrix multiplication, which is not more computationally expensive than computing logits during the forward pass of the model. Overall, this naive approach is sufficiently fast to be conducive to adversarial training. We also found that the attacks benefited from normalizing the gradient by taking its sign.

Extending this approach to finding the optimal perturbations for more than 1 substitution would require exhaustively searching over all possible combinations. However, previous work (Ebrahimi et al., 2018a) suggests that greedy search is a good enough approximation.

3.1.2 The Adversarial Loss $\mathcal{L}_{\text{adv}}$

We want to find an adversarial input $\hat{x}$ such that, assuming that the model has produced the correct output $y_1, \ldots, y_{t-1}$ up to step $t-1$ during decoding, the probability that the model makes an error at the next step t is maximized.

In the log-semiring, this translates into the following loss function:

$$\mathcal{L}_{\text{adv}}(\hat{x}, y) = \sum_{t=1}^{|y|} \log(1 - p(y_t \mid \hat{x}, y_1, \ldots, y_{t-1})) \tag{4}$$

3.2 Enforcing Semantically Similar Adversarial Inputs

In contrast to previous methods, which don't consider meaning preservation, we propose simple modifications of the approach presented in §3.1 to create adversarial perturbations at the word level that are more likely to preserve meaning. The basic idea is to restrict the possible word substitutions to similar words. We compare two sets of constraints:

kNN: This constraint enforces that the word be replaced only with one of its 10 nearest neighbors in the source embedding space. This has two effects: first, the replacement will be likely semantically related to the original word (if words close in the embedding space are indeed semantically related, as hinted by Table 1). Second, it ensures that the replacement's word vector is close enough to the original word vector that the first order assumption is more likely to be satisfied.

CharSwap: This constraint requires that the substituted words must be obtained by swapping

---

[6] Note that this formulation is also valid for character-based models (see Ebrahimi et al. (2018a)) and subword-based models. For subword-based models, additional difficulty would be introduced due to changes to the input resulting in different subword segmentations. This poses an interesting challenge that is beyond the scope of the current work.

[7] More generally we will use the bold $\mathbf{w}$ when talking about the embedding vector of word w

characters. Word internal character swaps have been shown to not affect human readers greatly (McCusker et al., 1981), hence making them likely to be meaning-preserving. Moreover we add the additional constraint that the substitution must not be in the vocabulary, which will likely be particularly meaning-destroying on the target side for the word-based models we test here. In such cases where word-internal character swaps are not possible or can't produce out-of-vocabulary (OOV) words, we resort to the naive strategy of repeating the last character of the word. The exact procedure used to produce this kind of perturbations is described in Appendix A.1. Note that for a word-based model, every OOV will look the same (a special `<unk>` token), however the choice of OOV will still have an influence on the output of the model because we use unk-replacement.

In contrast, we refer the base attack without constraints as **Unconstrained** hereforth. Table 1 gives qualitative examples of the kind of perturbations generated under the different constraints.

For subword-based models, we apply the same procedures at the subword-level on the original segmentation. We then de-segment and re-segment the resulting sentence (because changes at the subword or character levels are likely to change the segmentation of the resulting sentence).

4 Experiments

Our experiments serve two purposes. First, we examine our proposed framework of evaluating adversarial attacks (§2), and also elucidate which automatic metrics correlate better with human judgment for the purpose of evaluating adversarial attacks (§4.2). Second, we use this evaluation framework to compare various adversarial attacks and demonstrate that adversarial attacks that are explicitly constrained to preserve meaning receive better assessment scores (§4.3).

4.1 Experimental setting

Data: Following previous work on adversarial examples for seq2seq models (Belinkov and Bisk, 2018; Ebrahimi et al., 2018a), we perform all experiments on the IWSLT2016 dataset (Cettolo et al., 2016) in the {French,German,Czech}→English directions (`fr-en`, `de-en` and `cs-en`). We compile all previous IWSLT test sets before 2015 as validation data, and keep the 2015 and 2016 test sets as test data. The data is tokenized with the Moses tokenizer (Koehn et al., 2007). The exact data statistics can be found in Appendix A.2.

MT Models: We perform experiments with two common neural machine translation (NMT) models. The first is an LSTM based encoder-decoder architecture with attention (Luong et al., 2015). It uses 2-layer encoders and decoders, and dot-product attention. We set the word embedding dimension to 300 and all others to 500. The second model is a self-attentional Transformer (Vaswani et al., 2017), with 6 1024-dimensional encoder and decoder layers and 512 dimensional word embeddings. Both the models are trained with Adam (Kingma and Ba, 2014), dropout (Srivastava et al., 2014) of probability 0.3 and label smoothing (Szegedy et al., 2016) with value 0.1. We experiment with both word based models (vocabulary size fixed at 40k) and subword based models (BPE (Sennrich et al., 2016) with 30k operations). For word-based models, we perform `<unk>` replacement, replacing `<unk>` tokens in the translated sentences with the source words with the highest attention value during inference. The full experimental setup and source code are available at `https://github.com/pmichel31415/translate/tree/paul/pytorch_translate/research/adversarial/experiments`.

Automatic Metric Implementations: To evaluate both sentence and corpus level BLEU score, we first de-tokenize the output and use `sacreBLEU`[8] (Post, 2018) with its internal `intl` tokenization, to keep BLEU scores agnostic to tokenization. We compute METEOR using the official implementation[9]. ChrF is reported with the `sacreBLEU` implementation on detokenized text with default parameters. A toolkit implementing the evaluation framework described in §2.1 for these metrics is released at `https://github.com/pmichel31415/teapot-nlp`.

4.2 Correlation of Automatic Metrics with Human Judgment

We first examine which of the automatic metrics listed in §2.2 correlates most with human judgment for our adversarial attacks. For this experiment, we restrict the scope to the case of the

---

[8] `https://github.com/mjpost/sacreBLEU`

[9] `http://www.cs.cmu.edu/~alavie/METEOR/`

		LSTM			Transformer		
	Language pair	`cs-en`	`de-en`	`fr-en`	`cs-en`	`de-en`	`fr-en`
Word-based		Target RDChrF			Target RDChrF		
	Original chrF	45.68	49.43	57.49	47.66	51.08	58.04
	Unconstrained	25.38	25.54	25.59	25.24	25.00	24.68
	CharSwap	24.11	24.94	23.60	21.59	23.23	21.75
	kNN	15.00	15.59	15.22	20.74	19.97	18.59
		Source chrF			Source chrF		
	Unconstrained	70.14	72.39	74.29	69.03	71.93	73.23
	CharSwap	82.65	84.40	86.62	84.13	85.97	87.02
	kNN	78.08	78.11	77.62	74.94	77.92	77.88
Subword-based		Target RDChrF			Target RDChrF		
	Original chrF	48.30	52.42	59.08	49.70	54.01	59.65
	Unconstrained	25.79	26.03	26.96	23.97	25.07	25.28
	CharSwap	18.65	19.15	19.75	16.98	18.38	17.85
	kNN	15.00	16.26	17.12	19.02	18.58	18.63
		Source chrF			Source chrF		
	Unconstrained	69.32	72.12	73.57	68.66	71.51	72.65
	CharSwap	85.84	87.46	87.98	85.79	87.07	87.99
	kNN	76.17	77.74	78.03	73.05	75.91	76.54

Table 2: Target RDchrF and source chrF scores for all the attacks on all our models (word- and subword-based LSTM and Transformer).

LSTM model on `fr-en`. For the French side, we randomly select 900 sentence pairs $(x, \hat{x})$ from the validation set, 300 for each of the Unconstrained, kNN and CharSwap constraints. To vary the level of perturbation, the 300 pairs contain an equal amount of perturbed input obtained by substituting 1, 2 and 3 words. On the English side, we select 900 pairs of reference translations and translations of adversarial input $(y, \hat{y}_M)$ with the same distribution of attacks as the source side, as well as 300 (y, y_M) pairs (to include translations from original inputs). This amounts to 1,200 sentence pairs in the target side.

These sentences are sent to English and French speaking annotators to be rated according to the guidelines described in §2.2.1. Each sample (a pair of sentences) is rated by two independent evaluators. If the two ratings differ, the sample is sent to a third rater (an auditor and subject matter expert) who makes the final decision.

Finally, we compare the human results to each automatic metric with Pearson's correlation coefficient. The correlations are reported in Table 3. As evidenced by the results, chrF exhibits higher correlation with human judgment, followed by METEOR and BLEU. This is true both on the source side (x vs $\hat{x}$) and in the target side (y vs $\hat{y}_M$). We evaluate the statistical significance of this result using a paired bootstrap test for $p < 0.01$. Notably we find that chrF is significantly better than METEOR in French but not in English. This is not too unexpected because METEOR has access to more language-dependent resources in English (specifically synonym information) and thereby can make more informed matches of these synonymous words and phrases. Moreover the French source side contains more "character-level" errors (from CharSwap attacks) which are not picked-up well by word-based metrics like BLEU and METEOR. For a breakdown of the correlation coefficients according to number of perturbation and type of constraints, we refer to Appendix A.3.

Language	BLEU	METEOR	chrF
French	0.415	0.440	**0.586***
English	0.357	0.478*	**0.497**

Table 3: Correlation of automatic metrics to human judgment of adversarial source and target sentences. "*" indicates that the correlation is significantly better than the next-best one.

Thus, in the following, we report attack results both in terms of chrF in the source (s_{src}) and relative decrease in chrF (RDchrF) in the target (d_{tgt}).

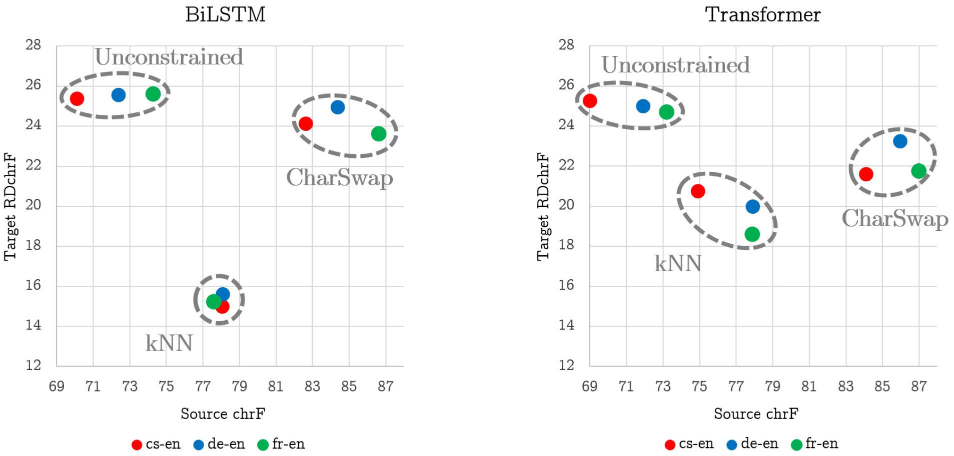


Figure 1: Graphical representation of the results in Table 2 for word-based models. High source chrF and target RDchrF (upper-right corner) indicates a good attack.

4.3 Attack Results

We can now compare attacks under the three constraints Unconstrained, kNN and CharSwap and draw conclusions on their capacity to preserve meaning in the source and destroy it in the target. Attacks are conducted on the validation set using the approach described in §3.1 with 3 substitutions (this means that each adversarial input is at edit distance at most 3 from the original input). Results (on a scale of 0 to 100 for readability) are reported in Table 2 for both word- and subword-based LSTM and Transformer models. To give a better idea of how the different variables (language pair, model, attack) affect performance, we give a graphical representation of these same results in Figure 1 for the word-based models. The rest of this section discusses the implication of these results.

Source chrF Highlights the Effect of Adding Constraints: Comparing the kNN and CharSwap rows to Unconstrained in the "source" sections of Table 2 clearly shows that constrained attacks have a positive effect on meaning preservation. Beyond validating our assumptions from §3.2, this shows that source chrF is useful to carry out the comparison in the first place[10]. To give a point of reference, results from the manual evaluation carried out in §4.2 show that 90% of the French sentence pairs to which humans gave a score of 4 or 5 in semantic similarity have a chrF > 78.

Successful attack (source chrF $= 80.89$, target RDchrF $= 84.06$)	
Original	Ils le réinvestissent directement en engageant plus de procès.
Adv. src	Ilss le réinvestissent dierctement en engagaent plus de procès.
Ref.	They plow it right back into filing more troll lawsuits.
Base output	They direct it directly by engaging more cases.
Adv. output	.. de plus.
Unsuccessful attack (source chrF $= 54.46$, target RDchrF $= 0.00$)	
Original	C'était en Juillet 1969.
Adv. src	C'étiat en Jiullet 1969.
Ref.	This is from July, 1969.
Base output	This was in July 1969.
Adv. output	This is. in 1969.

Table 4: Example of CharSwap attacks on the `fr-en` LSTM. The first example is a successful attack (high source chrF and target RDchrF) whereas the second is not.

Different Architectures are not Equal in the Face of Adversity: Inspection of the target-side results yields several interesting observations. First, the high RDchrF of CharSwap for word-based model is yet another indication of their known shortcomings when presented with words out of their training vocabulary, even with `<unk>`-replacement. Second, and perhaps more interestingly, Transformer models appear to be less robust to small embedding perturbations (kNN attacks) compared to LSTMs. Although the exploration of the exact reasons for this phenomenon is beyond the scope of this work, this is a good example that RDchrF can shed light on the different behavior of different architectures when confronted with adversarial input. Overall, we find that the Char-

---

[10] It can be argued that using chrF gives an advantage to CharSwap over kNN for source preservation (as opposed to METEOR for example). We find that this is the case for Czech and German (source METEOR is higher for kNN) but not French. Moreover we find (see A.3) that chrF correlates better with human judgement even for kNN.

Swap constraint is the only one that consistently produces attacks with > 1 average success (as defined in Section 2.1) according to Table 2. Table 4 contains two qualitative examples of this attack on the LSTM model in `fr-en`.

5 Adversarial Training with Meaning-Preserving Attacks

5.1 Adversarial Training

Adversarial training (Goodfellow et al., 2014) augments the training data with adversarial examples. Formally, in place of the negative log likelihood (NLL) objective on a sample x, y, $\mathcal{L}(x, y) = NLL(x, y)$, the loss function is replaced with an interpolation of the NLL of the original sample x, y and an adversarial sample $\hat{x}, y$:

$$\mathcal{L}'(x, y) = (1 - \alpha) NLL(x, y) + \alpha NLL(\hat{x}, y) \tag{5}$$

Ebrahimi et al. (2018a) suggest that while adversarial training improves robustness to adversarial attacks, it can be detrimental to test performance on non-adversarial input. We investigate whether this is still the case when adversarial attacks are largely meaning-preserving.

In our experiments, we generate $\hat{x}$ by applying 3 perturbations on the fly at each training step. To maintain training speed we do not solve Equation (2) iteratively but in one shot by replacing the argmax by top-3. Although this is less exact than iterating, this makes adversarial training time less than $2\times$ slower than normal training. We perform adversarial training with perturbations without constraints (Unconstrained-adv) and with the CharSwap constraint (CharSwap-adv). All experiments are conducted with the word-based LSTM model.

5.2 Results

Test performance on non-adversarial input is reported in Table 5. In keeping with the rest of the paper, we primarily report chrF results, but also show the standard BLEU as well.

We observe that when $\alpha = 1.0$, *i.e.* the model only sees the perturbed input during training[11], the Unconstrained-adv model suffers a drop in test performance, whereas CharSwap-adv's performance is on par with the original. This is likely attributable to the spurious training samples $(\hat{x}, y)$ where y is not an acceptable translation of $\hat{x}$ introduced by the lack of constraint. This effect disappears when $\alpha = 0.5$ because the model sees the original samples as well.

Language pair	`cs-en`	`de-en`	`fr-en`
Base	44.21 (22.89)	49.30 (28.61)	55.67 (35.28)
		$\alpha = 1.0$	
Unconstrained-adv	41.38 (21.51)	46.15 (27.06)	53.39 (33.96)
CharSwap-adv	43.74 (23.00)	48.85 (28.45)	55.60 (35.33)
		$\alpha = 0.5$	
Unconstrained-adv	43.68 (22.93)	48.60 (28.30)	55.55 (35.25)
CharSwap-adv	44.57 (23.66)	49.14 (28.66)	55.88 (35.63)

Table 5: chrF (BLEU) scores on the original test set before/after adversarial training of the word-based LSTM model.

Language pair	`cs-en`	`de-en`	`fr-en`
Base	24.11	24.94	23.60
		$\alpha = 1.0$	
Unconstrained-adv	25.99	26.24	25.67
CharSwap-adv	16.46	17.19	15.72
		$\alpha = 0.5$	
Unconstrained-adv	26.52	27.26	24.92
CharSwap-adv	20.41	20.24	16.08

Table 6: Robustness to CharSwap attacks on the validation set with/without adversarial training (RDchrF). Lower is better.

Not unexpectedly, Table 6 indicates that CharSwap-adv is more robust to CharSwap constrained attacks for both values of α, with 1.0 giving the best results. On the other hand, Unconstrained-adv is similarly or more vulnerable to these attacks than the baseline. Hence, we can safely conclude that adversarial training with CharSwap attacks improves robustness while not impacting test performance as much as unconstrained attacks.

6 Related work

Following seminal work on adversarial attacks by Szegedy et al. (2013), Goodfellow et al. (2014) introduced gradient-based attacks and adversarial training. Since then, a variety of attack (Moosavi-

---

[11] This setting is reminiscent of word dropout (Iyyer et al., 2015).

Dezfooli et al., 2016) and defense (Cissé et al., 2017; Kolter and Wong, 2017) mechanisms have been proposed. Adversarial examples for NLP specifically have seen attacks on sentiment (Papernot et al., 2016; Samanta and Mehta, 2017; Ebrahimi et al., 2018b), malware (Grosse et al., 2016), gender (Reddy and Knight, 2016) or toxicity (Hosseini et al., 2017) classification to cite a few.

In MT, methods have been proposed to attack word-based (Zhao et al., 2018; Cheng et al., 2018) and character-based (Belinkov and Bisk, 2018; Ebrahimi et al., 2018a) models. However these works side-step the question of meaning preservation in the source: they mostly focus on target side evaluation. Finally there is work centered around meaning-preserving adversarial attacks for NLP via paraphrase generation (Iyyer et al., 2018) or rule-based approaches (Jia and Liang, 2017; Ribeiro et al., 2018; Naik et al., 2018; Alzantot et al., 2018). However the proposed attacks are highly engineered and focused on English.

7 Conclusion

This paper highlights the importance of performing *meaning-preserving* adversarial perturbations for NLP models (with a focus on seq2seq). We proposed a general evaluation framework for adversarial perturbations and compared various automatic metrics as proxies for human judgment to instantiate this framework. We then confirmed that, in the context of MT, "naive" attacks do not preserve meaning in general, and proposed alternatives to remedy this issue. Finally, we have shown the utility of adversarial training in this paradigm. We hope that this helps future work in this area of research to evaluate meaning conservation more consistently.

Acknowledgments

The authors would like to extend their thanks to members of the LATTE team at Facebook and Neulab at Carnegie Mellon University for valuable discussions, as well as the anonymous reviewers for their insightful feedback. This research was partially funded by Facebook.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2016. The iwslt 2016 evaluation campaign.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *COLING*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Isao Goto, Ka-Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin Ka-Yin T'sou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *NII Test Collection for IR Systems*.

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

J. Zico Kolter and Eric Wong. 2017. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Xiaoyi Ma and Christopher Cieri. 2006. Corpus support for machine translation at ldc. In *Language Resources and Evaluation Conference*.

Leo X McCusker, Philip B Gough, and Randolph G Bias. 1981. Word recognition inside out and outside in. *Journal of Experimental Psychology: Human Perception and Performance*, 7(3):538.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 49–54. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Maja Popović. 2016. chrf deconstructed: beta parameters and n-gram weights. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 499–504.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Supplemental Material

A.1 Generating OOV Replacements with Internal Character Swaps

We use the following snippet to produce an OOV word from an existing word:

```python
def make_oov(
  word,
  vocab,
  max_scrambling,
):
  """Modify a word to make it OOV
  (while keeping the meaning)"""
  # If the word has >3 letters
  # try scrambling them
  L = len(word)
  if L > 3:
    # For a fixed number of steps
    for _ in range(max_scrambling):
      # Swap two adjacent letters
      # in the middle of the word
      pos = random.randint(1, L - 3)
      word = word[:pos]
      word += word[pos+1] + word[pos]
      word += word[pos+2:]
      # If we got an OOV already just
      # return it
      if word not in vocab:
        return word
  # If nothing worked, or the word is
  # too short for scrambling, just
  # repeat the last letter ad nauseam
  char = word[-1]
  while word in vocab:
    word = word + char
  return word
```

A.2 IWSLT2016 Dataset

See table 7 for statistics on the size of the IWSLT2016 corpus used in our experiments.

	#train	#valid	#test
`fr-en`	220.4k	6,824	2,213
`de-en`	196.9k	11,825	2,213
`cs-en`	114.4k	5,716	2,213

Table 7: IWSLT2016 data statistics.

A.3 Breakdown of Correlation with Human Judgement

We provide a breakdown of the correlation coefficients of automatic metrics with human judgment for source-side meaning-preservation, both in terms of number of perturbed words (Table 8) and constraint (Table 9). While those coefficients are computed on a much smaller sample size, and their differences are not all statistically significant with $p < 0.01$, they exhibit the same trend as the results from Table 3 (BLEU < METEOR < chrF).

# edits	BLEU	METEOR	chrF
1	0.351	0.352	**0.486***
2	0.403	0.424	**0.588***
3	0.334	0.393	**0.560***

Table 8: Correlation of automatic metrics to human judgment of semantic similarity between original and adversarial source sentences, broken down by number of perturbed words. "*" indicates that the correlation is significantly better than the next-best one.

Constraint	BLEU	METEOR	chrF
Unconstrained	0.274	0.572	**0.599**
CharSwap	0.274	0.319	**0.383**
kNN	0.534	0.584	**0.606**

Table 9: Correlation of automatic metrics to human judgment of semantic similarity between original and adversarial source sentences, broken down by type of constraint on the perturbation. "*" indicates that the correlation is significantly better than the next-best one.

In particular Table 8 shows that the good correlation of chrF with human judgment is not only due to the ability to distinguish between different number of edits.

Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction

Kazuma Hashimoto[*]
Salesforce Research
k.hashimoto@salesforce.com

Yoshimasa Tsuruoka
The University of Tokyo
tsuruoka@logos.t.u-tokyo.ac.jp

Abstract

A major obstacle in reinforcement learning-based sentence generation is the large action space whose size is equal to the vocabulary size of the target-side language. To improve the efficiency of reinforcement learning, we present a novel approach for reducing the action space based on dynamic vocabulary prediction. Our method first predicts a fixed-size small vocabulary for each input to generate its target sentence. The input-specific vocabularies are then used at supervised and reinforcement learning steps, and also at test time. In our experiments on six machine translation and two image captioning datasets, our method achieves faster reinforcement learning ($\sim$2.7x faster) with less GPU memory ($\sim$2.3x less) than the full-vocabulary counterpart. We also show that our method more effectively receives rewards with fewer iterations of supervised pre-training.

1 Introduction

Sentence generation with neural networks plays a key role in many language processing tasks, including machine translation (Sutskever et al., 2014), image captioning (Lin et al., 2014), and abstractive summarization (Rush et al., 2015). The most common approach for learning the sentence generation models is maximizing the likelihood of the model on the gold-standard target sentences. Recently, approaches based on reinforcement learning have attracted increasing attention to reduce the gap between training and test situations and to directly incorporate task-specific and more flexible evaluation metrics such as BLEU scores (Papineni et al., 2002) into optimization (Ranzato et al., 2016).

While reinforcement learning-based sentence generation is appealing, it is often too computationally demanding to be used with large training data. In reinforcement learning for sentence generation, selecting an action corresponds to selecting a word in the vocabulary V. The number of possible actions at each time step is thus equal to the vocabulary size, which often exceeds tens of thousands. Among such a large set of possible actions, at most N actions are selected if the length of the generated sentence is N, where we can assume $N \ll |V|$. In other words, most of the possible actions are not selected, and the large action space slows down reinforcement learning and consumes a large amount of GPU memory.

In this paper, we propose to accelerate reinforcement learning by reducing the large action space. The reduction of action space is achieved by predicting a small vocabulary for each source input. Our method first constructs the small input-specific vocabulary by selecting K (≤ 1000) relevant words, and then the small vocabulary is used at both training and test time.

Our experiments on six machine translation and two image captioning datasets show that our method enables faster reinforcement learning with less GPU memory than the standard full softmax method, without degrading the accuracy of the sentence generation tasks. Our method also works faster at test time, especially on CPUs. The implementation of our method is available at `https://github.com/hassyGo/NLG-RL`.

2 REINFORCE with Small Vocabularies

We first describe a neural machine translation model and an image captioning model as examples of sentence generation models. Machine translation is a text-to-text task, and image captioning is an image-to-text task. We then review how reinforcement learning is used, and present a simple and efficient method to accelerate the training.

---

[*] Work was done while the first author was working at the University of Tokyo.

Proceedings of NAACL-HLT 2019, pages 3115–3125
Minneapolis, Minnesota, June 2 - June 7, 2019.

2.1 Sentence Generation Models

Recurrent Neural Networks (RNNs) are widely used to generate sentences by outputting words one by one (Sutskever et al., 2014). To generate a sentence $Y = (y_1, y_2, \ldots, y_N)$, where N is its length, given a source input X, a hidden state $h_t \in \mathbb{R}^d$ is computed for each time step t (≥ 1) by using its previous information:

$$h_t = \mathrm{RNN}\left(h_{t-1}, e(y_{t-1}), s_{t-1}\right), \tag{1}$$

where $\mathrm{RNN}(\cdot)$ is an RNN function, $e(y_{t-1}) \in \mathbb{R}^d$ is a word embedding of y_{t-1}, and $s_{t-1} \in \mathbb{R}^d$ is a hidden state optionally used to explicitly incorporate the information about the source input X into the transition. We employ Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) for the RNN function. The task here is to predict the t-th word y_t by computing a target word distribution $p(y|y_{<t}, X) \in \mathbb{R}^{|V|}$, where $|V|$ represents the vocabulary size of the target language. $p(y|y_{<t}, X)$ is used to generate a sentence by either greedy/beam search or random sampling.

To learn the model parameters, the following cross entropy loss is usually employed:

$$L_c(Y_g, X) = -\sum_{t=1}^{N_g} \log p\left(y = y_t | y_{<t}, X\right), \tag{2}$$

where we assume that the target sentence Y_g is the gold sequence. Once we train the model, we can use it to generate unseen sentences.

Machine translation In the context of machine translation, the source input X corresponds to a source sentence $(x_1, x_2, \ldots, x_M)$ of length M. Each word x_i is also associated with a word embedding $\tilde{e}(x_i) \in \mathbb{R}^d$. We assume that a hidden state $\tilde{h}_i \in \mathbb{R}^{2d}$ is computed for each x_i by using a bi-directional RNN with LSTM units (Graves and Schmidhuber, 2005). That is, $\tilde{h}_i$ is the concatenation of x_i's d-dimensional hidden states $[\overrightarrow{h}_i; \overleftarrow{h}_i]$ computed by a pair of forward and backward RNNs. We set the initial hidden state of the sentence generator as $h_0 = \overrightarrow{h}_M + \overleftarrow{h}_1$. Following an attention mechanism proposed in Luong et al. (2015), s_t for predicting y_t is computed as follows:

$$s_t = \tanh\left(W_s\left[h_t; \sum_{i=1}^{M} a_i \tilde{h}_i\right] + b_s\right), \tag{3}$$

where $a_i = f(h_t, i, \tilde{h})$ is the global-attention function in Luong et al. (2015), $W_s \in \mathbb{R}^{d \times 3d}$ is a weight matrix, and $b_s \in \mathbb{R}^d$ is a bias vector. s_t is then used to compute the target word distribution:

$$p(y|y_{<t}, X) = \mathrm{softmax}(W_p s_t + b_p), \tag{4}$$

where $W_p \in \mathbb{R}^{|V| \times d}$ is a weight matrix, and $b_p \in \mathbb{R}^{|V|}$ is a bias vector.

Image captioning In the case of image captioning, the source input X corresponds to an image to be described. We assume that in our preprocessing step, each input image is fed into a convolutional neural network to extract its fixed-length feature vector $f \in \mathbb{R}^{d_f}$. More specifically, we use the pre-computed feature vectors provided by Kiros et al. (2014), and the feature vectors are never updated in any model training processes. The input feature vector is transformed into the initial hidden state $h_0 = \tanh\left(W_f f + b_f\right)$, where $W_f \in \mathbb{R}^{d \times d_f}$ is a weight matrix, and $b_f \in \mathbb{R}^d$ is a bias vector. In contrast to machine translation, we do not use s_{t-1} in Equation (1); more concretely, we do not use any attention mechanisms for image captioning. Therefore, we directly use the hidden state h_t to compute the target word distribution:

$$p(y|y_{<t}, X) = \mathrm{softmax}(W_p h_t + b_p), \tag{5}$$

where the weight and bias parameters are analogous to the ones in Equation (4).

For both of the tasks, we use the weight-tying technique (Inan et al., 2017; Press and Wolf, 2017) by using W_p as the word embedding matrix. That is, $e(y_t)$ is the y_t-th row vector in W_p, and the technique has shown to be effective in machine translation (Hashimoto and Tsuruoka, 2017) and text summarization (Paulus et al., 2018).

2.2 Applying Reinforcement Learning

One well-known limitation of using the cross entropy loss in Equation (2) is that the sentence generation models work differently at the training and test time. More concretely, the models only observe gold sequences at the training time, whereas the models have to handle unseen sequences to generate sentences at the test time.

To bridge the gap, reinforcement learning has started gaining much attention (Ranzato et al., 2016; Wu et al., 2016; Rennie et al., 2017; Zhang and Lapata, 2017; Paulus et al., 2018; Yang et al., 2018). In this work, we focus on the most popular method called REINFORCE (Williams, 1992).[1] In REINFORCE, the sentence generation model sets an initial state given a source input, and then iterates an action selection and its corresponding state transition. The action selection corresponds to randomly sampling a target word from Equation (4) and (5), and the state transition corresponds to the RNN transition in Equation (1).

Once a sentence is generated, an approximated loss function is defined as follows:

$$L_r(Y, X) = -\sum_{t=1}^{N} R_t \log p(y = y_t | y_{<t}, X), \quad (6)$$

where R_t is the reward at time step t, and the loss is approximated by the single example Y. R_t is used to evaluate how good the t-th action selection is. Unlike maximum likelihood training, the reward function can be defined by using task-specific evaluation scores like BLEU for machine translation. In this paper, we employ GLEU proposed by Wu et al. (2016), a variant of sentence-level BLEU. Following the implementation in Ranzato et al. (2016), we define $R_t = \mathrm{GLEU}(Y, Y_g) - b_t$, where b_t is a baseline value estimating the future reward from the next time step to reduce the variance of the gradients. To estimate b_t, we jointly train a linear regression model by minimizing $\|b_t - \mathrm{GLEU}(Y, Y_g)\|^2$, and b_t is computed as $b_t = \sigma(W_r \cdot s_t + b_r)$, where $W_r \in \mathbb{R}^d$ is a weight vector, b_r is a bias, $\sigma(\cdot)$ is the logistic sigmoid function, and in the case of image captioning, h_t is used instead of s_t.

Overall model training The reinforcement learning step is usually applied after pre-training the models with the cross entropy loss in Equation (2). At the REINFORCE phase, we define the following joint loss function:

$$L = \lambda L_c + (1 - \lambda) L_r, \quad (7)$$

where λ is a hyperparameter, and $\lambda = 0.0$ usually leads to unstable training (Wu et al., 2016).

2.3 Large Action-Space Reduction

The vocabulary size $|V|$ is usually more than ten thousands for datasets covering many sentences with a variety of topics. However, for example, at most 100 unique words are selected when generating a sentence of length 100. That is, the output length N is much smaller than the vocabulary size $|V|$, and this fact motivated us to reduce the large action space. Moreover, we have in practice found that REINFORCE runs several times slower than the supervised learning with the cross entropy loss.

To accelerate the training, we propose to construct a small action space for each source input. In other words, our method selects a small vocabulary V' of size K for each source input in advance to the model training. In this section, we assume that V' is given and represented with a sparse binary matrix $M_X \in \mathbb{R}^{K \times |V|}$, where there are only K non-zero elements at position (i, w_i) for $1 \leq i \leq K$. w_i is a unique word index in V. M_X is used to construct a small subset of the parameters in the softmax layer:

$$W'_p = M_X W_p, \quad b'_p = M_X b_p, \quad (8)$$

and $W'_p \in \mathbb{R}^{K \times d}$ and $b'_p \in \mathbb{R}^K$ are used instead of W_p and b_p in Equation (4) and (5). Therefore, in mini-batched processes with a mini-batch size B, our method constructs B different sets of (W'_p, b'_p).

Relationship to previous work Sampling-based approximation methods have previously been studied to reduce the computational cost at the large softmax layer in probabilistic language modeling (Ji et al., 2016; Zoph et al., 2016), and such methods are also used to enable one to train neural machine translation models on CPUs (Eriguchi et al., 2016). The construction of (W'_p, b'_p) in our method is similar to these softmax approximation methods in that they also sample small vocabularies either at the word level (Ji et al., 2016), sentence level (Hashimoto and Tsuruoka, 2017), or mini-batch level (Zoph et al., 2016). However, one significant difference is that the approximation methods work only at training time using the cross entropy loss, and full softmax computations are still required at test time. The difference is crucial because a sentence generation model needs to simulate its test-time behavior in reinforcement learning.

3 Target Vocabulary Prediction

The remaining question is how to construct the input-specific vocabulary V' for each source input X. This section describes our method to construct V' by using a vocabulary prediction model which is separated from the sentence generation models.

---

[1] We tried self critic (Rennie et al., 2017), but did not observe significant improvement over REINFORCE.

3.1 Input Representations

In the vocabulary prediction task, the input is the source X (source sentences or images) to be described, and the output is V'. We should be careful *not* to make the prediction model computationally expensive; otherwise the computational efficiency by our method would be canceled out.

To feed the information about X into our vocabulary prediction model, we define an input vector $v(X) \in \mathbb{R}^{d_v}$. For image captioning, we use the feature vector f described in Section 2.1: $v(X) = W_v f + b_v$, where $W_v \in \mathbb{R}^{d_v \times d_f}$ is a weight matrix, and $b_v \in \mathbb{R}^{d_v}$ is a bias vector. For machine translation, we employ a bag-of-embeddings representation: $v(X) = \frac{1}{M}\sum_{i=1}^{M} \tilde{e}_v(x_i)$, where the d_v-dimensional word embedding $\tilde{e}_v(x_i) \in \mathbb{R}^{d_v}$ is different from $\tilde{e}(x_i)$ used in the machine translation model. By using the different set of the model parameters, we avoid the situation that our vocabulary prediction model is affected during training the sentence generation models.

Relationship to previous work Vocabulary prediction has gained attention for training sequence-to-sequence models with the cross entropy loss (Weng et al., 2017; Wu et al., 2017), but not for reinforcement learning. Compared to our method, previous methods jointly train a vocabulary predictor by directly using source encoders as input to the predictor. One may expect joint learning to improve both of the vocabulary predictor and the sentence generator, but in practice such positive effects are not clearly observed. Weng et al. (2017) reported that the joint learning improves the accuracy of their machine translation models, but our preliminary experiments did not indicate such accuracy gain. Such a joint training approach requires the model to continuously update the vocabulary predictor during REINFORCE, because the encoder is shared. That is, the action space for each input changes during reinforcement learning, and we observed unstable training. Therefore, this work separately models the vocabulary predictor and focuses on the effects of using the small vocabularies for REINFORCE.

Another note is that Jean et al. (2015) and L'Hostis et al. (2016) also proposed to construct small vocabularies in advance to the cross entropy-based training. They suggest that the use of word alignment works well, but using the word alignment is not general enough, considering that there exist different types of source input. By contrast, our method can be straightforwardly applied to the two sentence generation tasks with the different input modalities (i.e. image and text).

3.2 Multi-Label Classification

Once the input representation $v(X)$ is computed, we further transform it by a single residual block (He et al., 2016): $r(X) = \mathrm{Res}\,(v(X)) \in \mathbb{R}^{d_v}$.[2] Then $r(X)$ is fed into a prediction layer:

$$o = \sigma\left(W_o r(X) + b_o\right), \tag{9}$$

where $W_o \in \mathbb{R}^{|V| \times d_v}$ is a weight matrix, and $b_o \in \mathbb{R}^{|V|}$ is a bias vector. The i-th element o_i corresponds to the probability that the i-th word in the target vocabulary V appears in the target sentence Y given its source X.

We use the training data for the sentence generations tasks to train the vocabulary predictor. For each X in the training data, we have its gold target sentence Y_g. We train the vocabulary predictor as a multi-label classification model by the following loss function:

$$-\sum_{i=1}^{|V|}\left(t_i \log o_i + (1 - t_i)\log(1 - o_i)\right), \tag{10}$$

where t_i is equal to 1.0 if the i-th word in V is included in Y_g, and otherwise t_i is 0.0. In practice, we apply the label smoothing technique (Szegedy et al., 2016) to the loss function.

We evaluate the accuracy of the vocabulary predictor by using a separate development split D:

$$\frac{\#\text{ of correctly predicted words in } D}{\#\text{ of words in } D}, \tag{11}$$

where we select the top-K predictions in Equation (9) for each source input X in D, and the evaluation metric is a recall score. We use the top-K words to construct the input-specific vocabularies V' for the sentence generation models, and we restrict that the recall is 100% for the training data.

4 Experimental Settings

We describe our experimental settings, and the details can be found in the supplemental material.

---

[2] We can use arbitrary types of hidden layers or even linear models like SVMs, but we found this one performed the best. We describe the details of this in the supplemental material.

Dataset	Size	$\lvert V\rvert$	$\max(N)$
En-De	100,000	24,482	50
En-Ja (100K)	100,000	23,536	50
En-Ja (2M)	1,998,821	70,668	100
En-Ja (2M, SW)	1,998,816	37,905	200
En-Vi	132,406	14,321	100
Ch-Ja	100,000	23,383	50
MS COCO	413,915	14,543	57
Flickr8K	30,000	4,521	38

Table 1: Statistics of the training datasets.

4.1 Datasets

We used machine translation datasets of four different language pairs: English-to-German (En-De), English-to-Japanese (En-Ja), English-to-Vietnamese (En-Vi), and Chinese-to-Japanese (Ch-Ja). For image captioning, we used two datasets: MS COCO (Lin et al., 2014) and Flickr8K. Table 1 summarizes the statistics of the training datasets, where the number of training examples ("Size"), the target vocabulary size ($|V|$), and the maximum length of the target sentences ($\max(N)$) are shown. For the machine translation datasets, we manually set $\max(N)$ and omitted training examples which violate the constraints.

En-De: We used 100,000 training sentence pairs from `news commentary` and `newstest2015` as our development set, following Eriguchi et al. (2017).

En-Ja: We used parallel sentences in ASPEC (Nakazawa et al., 2016) and constructed three types of datasets: En-Ja (100K), En-Ja (2M), and En-Ja (2M, SW). The 100K and 2M datasets were constructed with the first 100,000 and 2,000,000 sentence pairs, respectively. To test our method using subword units, we further preprocessed the 2M dataset by using the SentencePiece toolkit (Kudo and Richardson, 2018) to construct the En-Ja (2M, SW) dataset.

En-Vi: We used the pre-processed datasets provided by Luong and Manning (2015). Our development dataset is the `tst2012` dataset.

Ch-Ja: We constructed the Ch-Ja dataset by using the first 100,000 sentences from `ASPEC`.

MS COCO and Flickr8K: We used the pre-processed datasets provided by Kiros et al. (2014). We can also download the 4096-dimensional feature vectors f (i.e., $d_f = 4096$).

4.2 Settings of Vocabulary Prediction

We set $d_v = 512$ for all the experiments. We used `AdaGrad` (Duchi et al., 2011) to train the vocabulary predictor with a learning rate of 0.08 and a mini-batch size of 128. The model for each setting was tuned based on recall scores (with $K = 1000$) for the development split.

4.3 Settings of Sentence Generation

We set $d = 256$ with single-layer LSTMs for all the experiments, except for the En-Ja (2M) and (2M, SW) datasets. For the larger En-Ja datasets, we set $d = 512$ with two-layer LSTMs. We used stochastic gradient decent with momentum, with a learning rate of 1.0, a momentum rate of 0.75, and a mini-batch size of 128. The model for each setting was tuned based on BLEU scores for the development split. All of the models achieved the best BLEU scores for all the datasets within 15 to 20 training epochs. Each of the selected models with the best BLEU scores was used for the following REINFORCE step. For REINFORCE, we set $\lambda = 0.005$, and the learning rate was set to 0.01. The REINFORCE steps required around 5 epochs to significantly improve the BLEU scores.

4.4 Computational Resources and Mini-Batch Processing

We used a single GPU of `NVIDIA GeForce GTX 1080`[3] to run experiments for the En-De, En-Ja (100K), En-Vi, Ch-Ja, MS COCO, and Flickr8K datasets. For the En-Ja (2M) and En-Ja (2M, SW) datasets, we used a single GPU of `NVIDIA Tesla V100`[4] to speedup our experiments.

Mini-batch splitting It should be noted that our small softmax method can be run even on the single `GTX 1080` GPU for the larger translation datasets, whereas the full softmax method runs out of the GPU memory. A typical strategy to address such out-of-memory issues is to use multiple GPUs, but we have found that we need at most eight GPUs to conduct our experiments on the full softmax method with REINFORCE.[5] Moreover, using the multiple GPUs does not always speedup the training time. We instead employ another strategy to split the mini-batch at each training iteration. First, we sort the mini-batch examples according to the lengths of the source (or target) text, and then split the mini-batch into S sets of the training examples. For example, in our case the

---

[3] The GPU memory capacity is 11,178MiB.

[4] The GPU memory capacity is 16,152MiB (AWS p3).

[5] This also depends on the mini-batch size.

		Cross entropy		REINFORCE w/ cross entropy	
		Small softmax	Full softmax	Small softmax	Full softmax
Translation	En-De	11.09±0.51	10.84±0.37	12.13±0.33	11.73±0.23
	En-Ja (100K)	28.26±0.15	28.05±0.40	29.14±0.13	29.01±0.35
	En-Vi	24.56±0.14	24.53±0.18	24.98±0.11	24.92±0.09
	Ch-Ja	29.27±0.08	28.97±0.15	30.10±0.12	29.80±0.15
Image captioning	MS COCO	24.88±0.25	24.75±0.36	26.43±0.32	25.74±0.13
	Flickr8K	16.45±0.28	16.52±0.11	19.04±0.43	19.17±0.24

Table 2: BLEU scores for the development splits of the six datasets. "Small softmax" corresponds to our method.

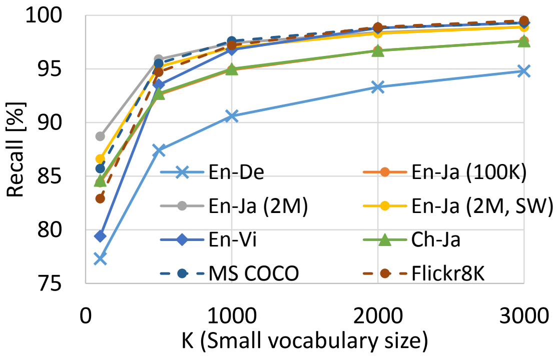


Figure 1: Recall scores of our vocabulary predictor.

mini-batch size is 128, and if S is set to 4, each of the smaller sets includes 32 training examples. We perform back-propagation for each set one by one, and at each step we delete the corresponding computational graphs to reduce the GPU memory consumption. Finally, the accumulated partial derivatives are used to update the model parameters. More details can be found in our Pytorch 0.4 implementation.

5 Results of Sentence Generation Tasks

5.1 Accuracy of Vocabulary Prediction

Figure 1 shows recall scores with respect to different values of the small vocabulary size K for each dataset. We can see that the recall scores reach 95% with $K = 1000$ for most of the datasets. One exception is the En-De dataset, and this is not surprising because a German vocabulary would become sparse by many compound nouns.

These results show that our vocabulary predictor works well for source inputs of different modalities (text and image) and their corresponding different target languages. Our method also works at the subword level as well as at the standard word level. For training the sentence generation models, we set $K = 500$ for the Flickr8K dataset and $K = 1000$ for the other datasets. Our empirical recommendation is $K = 1000$ if $|V|$ is larger than 10,000 and otherwise $K = 500$.

5.2 Accuracy of Sentence Generation

The goal of this paper is achieving efficient reinforcement learning for sentence generation to encourage future research, but before evaluating the efficiency of our method, we show that using the small vocabularies does not degrade the accuracy of the sentence generation models. Table 2 shows BLEU scores for the development splits of the four machine translation and two image captioning datasets. The BLEU scores are averaged over five different runs with different random seeds, and the standard deviations are also reported.

We can see in Table 2 that our method (Small softmax) keeps the BLEU scores as high as those of "Full softmax". For some datasets, the BLEU scores of our method are even better than those of the full softmax method. The trend is consistent in both of the cross entropy training phase and the REINFORCE phase. These results indicate that our method works well for different machine translation and image captioning datasets. We also confirmed that our experimental results are competitive with previously reported results when using the same training datasets; for example, our En-Vi test set result on `tst2013` is 27.87±0.21 (cf. 26.9 in Luong and Manning (2015)).

Better generation of rare words These BLEU scores suggest that our method for reinforcement learning has the potential to outperform the full softmax baseline. However, it is still unclear what is the potential advantage in terms of generation quality. We therefore analyzed the differences between output sentences of the small and full softmax methods, following Ott et al. (2018). Figure 2 shows the results of the En-De translation dataset,

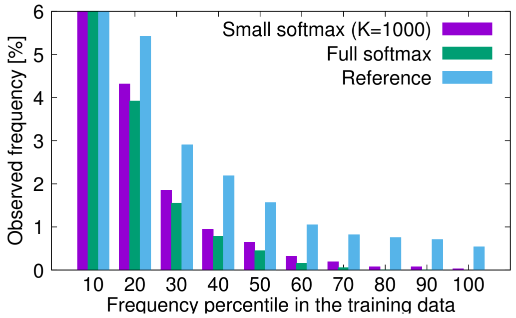


Figure 2: An analysis on the En-De translation results. The small and full softmax results are based on the REINFORCE setting, and "Reference" corresponds to the En-De reference translations of the development split.

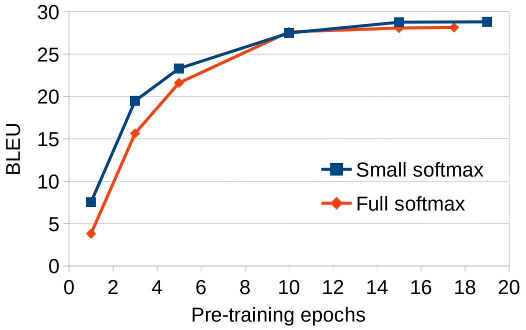


Figure 3: Effects of pre-training for REINFORCE.

	2M	2M, SW
Cross entropy	38.76	39.15
w/ beam search	39.88	40.35
REINFORCE w/ cross entropy	40.10	40.26
w/ beam search	40.36	40.38
w/ beam search (K= 500)	40.07	40.07
w/ beam search (K=2000)	40.30	40.50
w/ beam search (K=3000)	40.27	40.41

Table 3: BLEU scores for the development split of the En-Ja (2M) and En-Ja (2M, SW) datasets.

REINFORCE w/ cross entropy (K=1000)	40.16
w/ beam search	40.50
- Cross entropy (1.3M) w/ beam search (Hashimoto and Tsuruoka, 2017)	39.42
- Cross entropy (2M) w/ beam search (Oda et al., 2017b)	40.29
- Cross entropy (2M+1M back-trans.) w/ beam search (Morishita et al., 2017)	41.42

Table 4: BLEU scores for the En-Ja test split, where we use the En-Ja (2M, SW) dataset. The 95% confidence interval by bootstrap resampling (Noreen, 1989) is [39.61, 41.47] for our beam search result.

and we observed the same trend for all the other datasets. Each entry is computed as follows:

$$\frac{\#\text{ of output words in each percentile}}{\#\text{ of output words}}, \quad (12)$$

where the "10" percentile includes the top 10% of the most *frequent* words, and the "100" percentile includes the top 10% of the most *infrequent* words. We can see that our small softmax method better outputs rare words, and these results suggest that using input-specific vocabularies is useful in controlling action spaces for reinforcement learning.

Effectiveness with fewer pre-training steps We followed the standard practice that the models are pre-trained by maximum likelihood before starting reinforcement learning. However, such pre-training may have a negative effect in reinforcement learning. Consider the situation where the pre-training leads to zero cross-entropy loss. In this case, nothing will be learned during reinforcement learning because no exploratory action can be performed. Although pre-training in practice does not lead to zero cross-entropy loss, it can still overfit the data and result in very sharp output distributions, thereby hindering exploration in reinforcement learning. It is therefore important to consider a reinforcement learning setting with less or no pre-training (Liu et al., 2018). In Figure 3 for the En-Ja (100K) dataset, we show that the small softmax method works more effectively with fewer pre-training epochs. For this experiment, we set $\lambda = 0$ in Equation (7) to purely focus on REINFORCE. Using GLEU (or BLEU) scores gives sparse rewards, and thus the resulting BLEU scores are very low with fewer pre-training steps, but the small softmax method has the potential to work well if we can design more effective reward functions.

Results on larger datasets To see whether our method works in larger scales, Table 3 shows BLEU scores for the development split when using the En-Ja (2M) and En-Ja (2M, SW) datasets.[6] These results show that our method consistently works even on these larger datasets at the word and subword levels. In this table we also report how our method works with beam search, and the greedy-based BLEU scores are very close to those of beam search after the REINFORCE phase. When performing a beam search, we can optionally use different sizes of the small vocab-

---

[6] For the 2M dataset, the full softmax baseline achieves BLEU scores of 38.67 and 39.84 for the "Cross entropy" and "REINFORCE w/ cross entropy" settings, respectively.

			Training time [minutes/epoch]				GPU memory [MiB]			
			CE		REINFORCE w/ CE		CE		REINFORCE w/ CE	
	$\|V\|$	$\max(N)$	Small	Full	Small	Full	Small	Full	Small	Full
En-Ja (100K)	23,536	50	4.6	4.8	10.1	21.2	1,781	6,061	2,193	7,443
En-Ja (2M)	70,668	100	95.7	141.4	231.3	635.9	5,033	10,527	6,485	14,803
En-Vi	14,321	100	10.5	10.7	23.2	38.4	2,149	10,645	2,909	10,807
MS COCO	14,543	57	4.4	4.2	11.6	22.9	1,419	8,587	1,785	10,651
Flickr8K	4,521	38	0.3	0.3	0.8	0.9	911	2,031	1,031	3,197

Table 5: Training time, and maximum memory consumption on our GPU devices for the text generation models. For the full softmax baseline on the En-Ja (2M) experiments, the mini-batch splitting strategy (described in Section 4.4) is applied. CE: Cross-Entropy, Small: Small softmax (our proposed method), Full: Full softmax (the baseline).

ulary, but we observe that our method is robust to the changes, whereas Wu et al. (2017) reported that their dynamic vocabulary selection method is sensitive to such changes.

For reference, we report the test set results in Table 4. We cite BLEU scores from previously published papers which reported results of single models (i.e., without ensemble). Our method with greedy translation achieves a competitive score. It should be noted that Morishita et al. (2017) achieve a better score presumably because they used additional in-domain one million parallel sentences obtained by the back-translation technique (Sennrich et al., 2016).

6 Efficiency of the Proposed Method

This section discusses our main contribution: how efficient our method is in accelerating reinforcement learning for sentence generation.

6.1 Speedup at Training Time

We have examined the training-time efficiency of our method. Table 5 shows the training time [minutes/epoch] for five different datasets. We selected the five datasets to show results with different vocabulary sizes and different maximum sentence lengths, and we observed the same trend on the other datasets. The vocabulary size $|V|$ and the maximum sentence length $\max(N)$ are shown for each training dataset. In the training with the standard cross entropy loss, the speedup by our method is not impressive as long as the vocabulary size $|V|$ can be easily handled by the GPUs. We set $S = 2$ for the cross entropy training of the "Full softmax" method in the En-Ja (2M) setting, to reduce the GPU memory consumption as described in Section 4.4.

In the training with the REINFORCE algorithm, the speedup by our method is enhanced. In particular, in the En-Ja (2M) experiments, our method gains a factor of 2.7 speedup compared with the full softmax baseline ($S = 3$). For most of the experimental settings, the speedup significantly accelerates our research and development cycles when working on reinforcement learning for sentence generation tasks. One exception is the Flickr8K dataset whose original vocabulary size $|V|$ is already very small, and the lengths of the target sentences are short. In the supplementary material, we also show the test-time efficiency.

6.2 GPU Memory Consumption

Our method is also efficient in terms of GPU memory consumption at training time. Table 5 also shows the maximum GPU memory consumption during the training. These results show that our method easily fits in the memory of the single `GTX 1080` GPU, whereas "Full softmax" is very sensitive to the vocabulary size $|V|$ and the sentence lengths. In particular, we observe about 56% reduction in memory usage when using the En-Ja (2M) dataset. By saving the memory usage, one could try using larger models, larger mini-batches, larger vocabularies, and longer target sentences without relying on multiple GPUs.

Scalability of our method To further show the memory efficiency our our method, we measured the GPU memory consumption with a larger mini-batch size, 2048. We applied the mini-batch splitting strategy to both the small and full softmax methods to handle such a large mini-batch size. In the En-Ja (2M) experiments with REINFORCE, our small softmax method works with the large batch-size by setting $S = 6$, whereas the full softmax baseline needs $S = 40$. Aggressively splitting the mini-batch (i.e. using larger values of S) slows down the training time, and in that sense our method is much more efficient when we consider the larger mini-batch sizes. If we increase the mini-batch size to 4096, our small softmax method works with $S = 12$.

7 Related Work

Reducing the computational cost at the large softmax layer in language modeling/generation is actively studied (Jean et al., 2015; Ji et al., 2016; Eriguchi et al., 2016; L'Hostis et al., 2016; Zoph et al., 2016; Wu et al., 2017). Most of the existing methods try to reduce the vocabulary size by either negative sampling or vocabulary prediction. One exception is that Oda et al. (2017a) propose to predict a binary code of its corresponding target word. Although such a sophisticated method is promising, we focused on the vocabulary reduction method to apply policy-based reinforcement learning in a straightforward way.

As reported in this paper, one simple way to define a reward function for reinforcement learning is to use task-specific automatic evaluation metrics (Ranzato et al., 2016; Wu et al., 2016; Rennie et al., 2017; Zhang and Lapata, 2017; Paulus et al., 2018), but this is limited in that we can only use training data with gold target sentences. An alternative approach is to use a *discriminator* in generative adversarial networks (Goodfellow et al., 2014), and Yang et al. (2018) showed that REINFORCE with such a discriminator improves translation accuracy. However, Yang et al. (2018) only used the training data, and thus the potential of the generative adversarial networks is not fully realized. One promising direction is to improve the use of the generative adversarial networks for the sentence generation tasks by using our method, because our method can also accelerate the combination of REINFORCE and the discriminator.

8 Conclusion

This paper has presented how to accelerate reinforcement learning for sentence generation tasks by reducing large action spaces. Our method is as accurate as, is faster than, and uses less GPU memory than the standard full softmax counterpart, on sentence generation tasks of different modalities. In future work, it is interesting to use our method in generative adversarial networks to further improve the sentence generation models.

Acknowledgments

We thank anonymous reviewers for their fruitful comments. This work was supported by JST CREST Grant Number JPMJCR1513, Japan.

References

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural Machine Translation with Source-Side Latent Graph Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Proceedings of the 14th European Conference on Computer Vision*, pages 630–645.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *Proceedings of the 5th International Conference on Learning Representations*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1–10.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv*, cs.LG 1411.2539.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary Selection Strategies for Neural Machine Translation. *arXiv*, cs.CL 1610.00072.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. 2014. Microsoft COCO: Common Objects in Context. *arXiv*, cs.CV 1405.0312.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *Proceedings of the 6th International Conference on Learning Representations*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domain. In *International Workshop on Spoken Language Translation*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. NTT Neural Machine Translation Systems at WAT 2017. In *Proceedings of the 4th Workshop on Asian Translation*, pages 89–94.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian Scientific Paper Excerpt Corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation*.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Yusuke Oda, Philip Arthur, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. 2017a. Neural Machine Translation via Binary Code Prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 850–860.

Yusuke Oda, Katsuhito Sudoh, Satoshi Nakamura, Masao Utiyama, and Eiichiro Sumita. 2017b. A Simple and Strong Baseline: NAIST-NICT Neural Machine Translation System for WAT2017 English-Japanese Translation Task. In *Proceedings of the 4th Workshop on Asian Translation*, pages 135–139.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3956–3965.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A Deep Reinforced Model for Abstractive Summarization. In *Proceedings of the 6th International Conference on Learning Representations*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Marc?fAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical Sequence Training for Image Captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jon Shlens. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xin-Yu Dai, and Jiajun Chen. 2017. Neural Machine Translation with Word Predictions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 136–145.

Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256.

Yonghui Wu et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv*, cs.CL 1609.08144.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2017. Neural Response Generation with Dynamic Vocabularies. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1346–1355.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222.

Mitigating Uncertainty in Document Classification

Xuchao Zhang†‡, **Fanglan Chen**†, **Chang-Tien Lu**†, **Naren Ramakrishnan**†
†Discovery Analytics Center, Virginia Tech, Falls Church, VA, USA
‡NEC Laboratories America, Inc, Princeton, NJ, USA
†{xuczhang, fanglanc, ctlu, naren}@vt.edu, ‡xuczhang@nec-labs.com

Abstract

The uncertainty measurement of classifiers' predictions is especially important in applications such as medical diagnoses that need to ensure limited human resources can focus on the most uncertain predictions returned by machine learning models. However, few existing uncertainty models attempt to improve overall prediction accuracy where human resources are involved in the text classification task. In this paper, we propose a novel neural-network-based model that applies a new dropout-entropy method for uncertainty measurement. We also design a metric learning method on feature representations, which can boost the performance of dropout-based uncertainty methods with smaller prediction variance in accurate prediction trials. Extensive experiments on real-world data sets demonstrate that our method can achieve a considerable improvement in overall prediction accuracy compared to existing approaches. In particular, our model improved the accuracy from 0.78 to 0.92 when 30% of the most uncertain predictions were handed over to human experts in "20NewsGroup" data.

1 Introduction

Machine learning algorithms are gradually taking over from the human operators in tasks such as machine translation (Bahdanau et al., 2014), optical character recognition (Mithe et al., 2013), and face recognition (Parkhi et al., 2015). However, some real-world applications require higher accuracy than the results achieved by state-of-the-art algorithms, which makes it difficult to directly apply these algorithms in certain scenarios. For example, a medical diagnosis system (van der Westhuizen and Lasenby, 2017) is expected to have a very high accuracy to support correct decision-making for medical practitioners. Although domain experts can achieve a high performance in these challenging tasks, it is not always feasible to rely on limited and expensive human input for large-scale data sets. Therefore, if we have a model with 70% prediction accuracy, it is intuitive to ask what percentage of the data should be handed to domain experts to achieve an overall accuracy rate above 90%? To maximize the value of limited human resources while achieving desirable results, modeling uncertainty accurately is extremely important to ensure that domain experts can focus on the most uncertain results returned by machine learning models.

Most existing uncertainty models are based on Bayesian models, which are not only time-consuming but also unable to handle large-scale data sets. Deep Neural networks (DNNs) have attracted increasing attention in recent years and have been reported to achieve state-of-the-art performance in various machine learning tasks (Yang et al., 2016; Iyyer et al., 2014). However, unlike probabilistic models, DNNs are still at the early development stage in regards to providing the model uncertainty in their predictions. For those seeking to address the prediction uncertainty in DNNs, it is common to suffer from the following issues on the text classification task. Firstly, few researchers have sought to improve overall prediction performance when only limited human resources are available. Different from existing methods which focus on the value of uncertainty, this problem needs to get domain experts involved in emphasis on the order of the uncertain predictions. For example, the importance of distance between feature representations is neglected by the majority of existing models, but actually this is crucial for improving the order of uncertain predictions, especially during the pre-training of embedding vectors. Moreover, the methods proposed for continuous feature space cannot be applied to discrete text data. For example, adversarial train-

Proceedings of NAACL-HLT 2019, pages 3126–3136
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing is used in some uncertainty models (Goodfellow et al., 2014; Lakshminarayanan et al., 2017; Mandelbaum and Weinshall, 2017). However, due to its dependence on gradient-based methods to generate adversarial examples, the method is not applicable to discrete text data.

In order to simultaneously address all these problems in existing methods, the work presented in this paper adopts a DNN-based approach that incorporates a novel dropout-entropy uncertainty measurement method along with metric learning in the feature representation to handle the uncertainty problem in the document classification task. The study's main contributions can be summarized as follows:

- A novel DNN-based text classification model is proposed to achieve higher model accuracy with limited human input. In this new approach, a reliable uncertainty model learns to identify the accurate predictions with smaller estimated uncertainty.

- Metric learning in feature representation is designed to boost the performance of the dropout-based uncertainty methods in the text classification task. Specifically, the shortened intra-class distance and enlarged inter-class distance can reduce the prediction variance and increase the confidence for the accurate predictions.

- A new dropout-entropy method based on the Bayesian approximation property of Dropout in DNNs is presented. Specifically, we measure the model uncertainty in terms of the information entropy of multiple dropout-based evaluations combined with the de-noising mask operations.

- Extensive experiments on real-world data sets demonstrate that the effectiveness of our proposed approach consistently outperforms existing methods. In particular, the macro-F1 score can be increased from 0.78 to 0.92 by assigning 25% of the labeling work to human experts in a 20-class text classification task.

The rest of this paper is organized as follows. Section 2 reviews related work, and Section 3 provides a detailed description of our proposed model. The experiments on multiple real-world data sets are presented in Section 4. The paper concludes with a summary of the research in Section 5.

2 Related Work

The work related to this paper falls into two sub topics, described as follows.

2.1 Model Uncertainty

Existing uncertainty models are usually based on Bayesian models, which is Traditional Bayesian models such as Gaussian Process (GP), can measure uncertainty of model. However, as a nonparametric model, the time complexity of GP is increased by the size of data, which makes it intractable in many real world applications.

Conformal Prediction (CP) was proposed as a new approach to obtain confidence values (Vovk et al., 1999). Unlike the traditional underlying algorithm, conformal predictors provide each of the predictions with a measure of confidence. Also, a measure of “credibility serves as an indicator of how suitable the training data are used for the classification task (Shafer and Vovk, 2008). Different from Bayesian-based methods, CP approaches obtain probabilistically valid results, which are merely based on the independent and identically distributed assumption. The drawback of CP methods is their computational inefficiency, which renders the application CP not applicable for any model that requires long training time such as Deep Neural Networks.

With the recently heated research on DNNs, the associated uncertainty models have received a great deal of attention. Bayesian Neural Networks are a class of neural networks which are capable of modeling uncertainty (Denker and LeCun, 1990) (Hernández-Lobato and Adams, 2015). These models not only generate predictions but also provide the corresponding variance (uncertainty) of predictions. However, as the number of model parameters increases, these models become computationally more expensive (Wang and Yeung, 2016). Lee et al. proposed a computationally efficient uncertainty method that treats Deep Neural Networks as Gaussian Processes (Lee et al., 2017). Due to its kernel-based design, however, it is not straightforward to apply this to the deep network structures for text classification. Gal and Ghahramani used dropout in DNNs as an approximate Bayesian inference in deep Gaussian processes (Gal and Ghahramani, 2016) to mitigate the problem of representing uncertainty in deep learning without sacrificing the computational complexity. Dropout-based methods have also been extended to various tasks such as computer vision (Kendall and Gal, 2017), autonomous vehicle safety (McAllister et al., 2017) and medical decision making (van der Westhuizen and Lasenby, 2017).

However, few of these methods are specifically designed for text classification and lack of considerations on improving the overall accuracy in the scenario that domain experts can be involved in the process.

2.2 Metric Learning

Metric learning (Xing et al., 2003; Weinberger et al., 2006) algorithms design distance metrics that capture the relationships among data representations. This approach has been widely used in various machine learning applications, including image segmentation (Gong et al., 2013), face recognition (Guillaumin et al., 2009), document retrieval (Xu et al., 2012), and collaborative filtering (Hsieh et al., 2017). Weinberger et al. proposed a large margin nearest neighbor (LMNN) method (Weinberger et al., 2006) in learning a metric to minimize the number of class impostors based on pull and push losses. However, as yet there have been no report of work focusing specifically on mitigating prediction uncertainties. Mandelbaum and Weinshall (Mandelbaum and Weinshall, 2017) measured model uncertainty by the distance when comparing to the feature representations in training data, but this makes the uncertainty measurement inefficient because it requires an iteration over the entire training data set. To the best of our knowledge, we are the first to apply metric learning to mitigate model uncertainty in the text classification task. We also demonstrate that metric learning can be applied to dropout-based approaches to improve their prediction uncertainty.

3 Model

In this section, we propose a DNN-based approach to predict document categories with high confidence for the accurate predictions and high uncertainty for the inaccurate predictions. The overall architecture of the proposed model is presented in Section 3.1. The technical details for the metric loss and model uncertainty predictions are described in Sections 3.2 and 3.3, respectively.

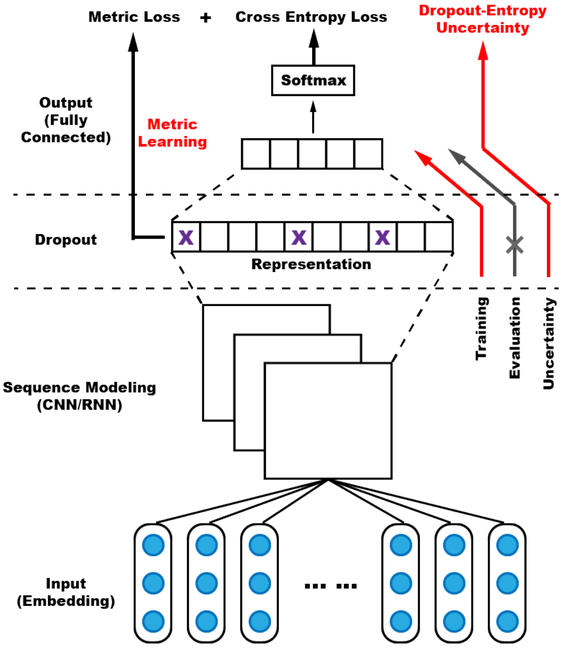


Figure 1: Overall Architecture of Proposed Model

3.1 Model Overview

In order to measure the uncertainty of the predictions for document classification task, we propose a neural-network-based model augmented with dropout-entropy uncertainty measurement and incorporating metric learning in its feature representation. The overall structure of the proposed model is shown in Figure 1. Our proposed model has four layers: 1) **Input Layer**. The input layer is represented by the word embeddings of each words in the document. By default, all word vectors are initialized by Glove (Pennington et al., 2014) pre-trained word vectors in Wikipedia with an embedding dimension of 200. 2) **Sequence Modeling Layer**. The sequence modeling layer extracts the feature representations from word vectors. This is usually implemented by Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN). In this paper, we focus on a CNN implementation with max pooling that utilizes 3 kernels with filter sizes of 3, 4 and 5, respectively. After that, a max pooling operation is applied on the output of sequence model. 3) **Dropout layer**. The convolutional layers usually contain a relatively small number of parameters compared to the fully connected layers. It is therefore reasonable to assume that CNN layers suffer less from overfitting, so Dropout is not usually used after CNN layers as it achieves only a trivial performance

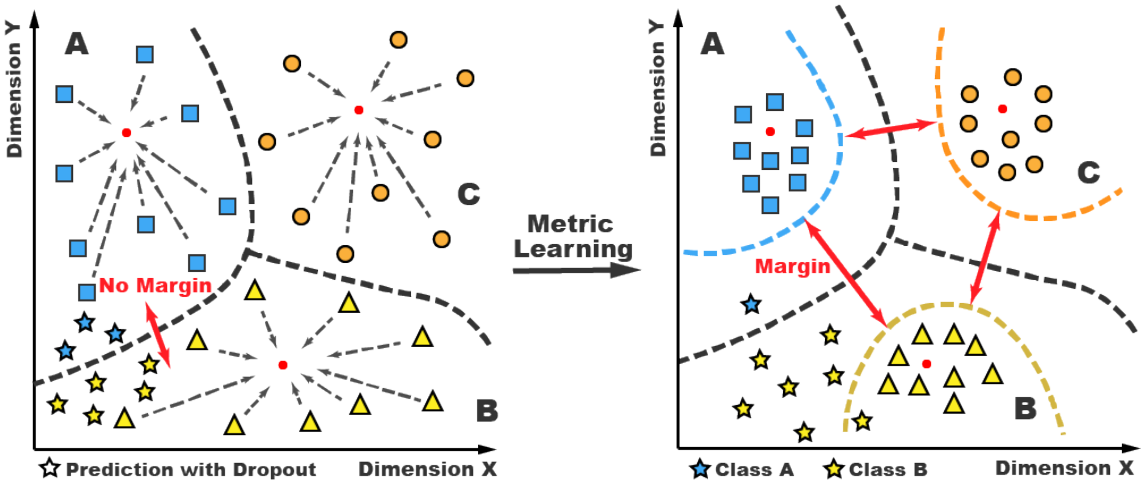


Figure 2: Feature representations with no metric learning (left) and metric learning (right).

improvement (Srivastava et al., 2014). However, since there is only one fully-connected layer in our model, we opted to add one Dropout layer after the CNN layer, not only to prevent overfitting, but also to measure prediction uncertainty (Gal and Ghahramani, 2016). The Dropout operation will be randomly applied to the activations during the training and uncertainty measurement phrases, but will not be applied to the evaluation phrase. 4) **Output layers**. The output is connected by a fully connected layer and the softmax. The loss function of our model is the combination of the cross entropy loss of the prediction and the metric loss of the feature representation. We regard the output of the Dropout layer as the representation of the document and deposit it into a metric loss function. The purpose here is to penalize large distance feature representations in the same class and small distance feature representations among different classes. The details of the metric loss function will be described in Section 3.2.

3.2 Metric Learning on Text Features

For uncertainty learning in text feature space, our purpose is to ensure the Euclidean distance between intra-class instances is much smaller than the inter-class instances. To achieve this, we use metric learning to train the desirable embeddings. Specifically, let $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ be the feature representations of instances i and j, respectively, then the Euclidean distance between them is defined as $\mathcal{D}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \frac{1}{d}\|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2^2$, where d is the dimension of the feature representation.

Suppose the data instances in the training data contain n classes and these are categorized into n subsets $\{S_k\}_{k=1}^{n}$, where S_k denotes the set of data instances belong to class k. Then the intra-class loss penalizes the large Euclidean distance between the feature representations in the same class, which can be formalized as Equation (1).

$$\mathcal{L}_{\text{intra}}(k) = \frac{2}{|S_k|^2 - |S_k|} \sum_{i,j \in S_k, i<j} \mathcal{D}(\boldsymbol{r}_i, \boldsymbol{r}_j) \tag{1}$$

where $|S_k|$ represents the number of elements in set S_k. The loss is the sum of all the feature distances between each possible pair in the same class set. Then, the loss is normalized by the number of unique pairs belonging to each class set.

The inter-class loss ensures large feature distances between different classes, which is formally defined as Equation (2).

$$\mathcal{L}_{\text{inter}}(p, q) = \frac{1}{|S_p| \cdot |S_q|} \sum_{i \in S_p, j \in S_q} \Big[m - \mathcal{D}(\boldsymbol{r}_i, \boldsymbol{r}_j) \Big]_+ \tag{2}$$

where m is a metric margin constant to distinguish between the intra- and inter-classes and $[z]_+ = \max(0, z)$ denotes the standard hinge loss. If the feature distance between instances from different classes is larger than m, the loss is zero. Otherwise, we use the value of m minus the distance as its penalty loss, with a larger m representing a larger inter-class distance. This parameter usually varies when we use different word embedding methods. In our experiment, we found that a small m is normally needed when the word embedding is initialized by a pre-trained word vector method such as Glove (Pennington et al., 2014); a larger m is required if word vectors are initialized randomly. The overall metric loss function is defined in Equation (3). This combines the intra-class loss and inter-class loss for all the classes.

$$\mathcal{L}_{\text{metric}} = \sum_{k=1}^{n} \left\{ \mathcal{L}_{\text{intra}}(k) + \lambda \sum_{i \neq k} \mathcal{L}_{\text{inter}}(k,i) \right\} \tag{3}$$

where λ is a pre-defined parameter to weight the importance of the intra- and inter-class losses. We set λ to 0.1 by default.

Figure 2 illustrates an example of a three-class feature representation in two dimensions. The left-hand figure shows the feature distribution trained with no metric learning. Obviously, the feature distance of the intra-class is large, sometimes even exceeding those of the inter-class distance near the decision boundary. However, the features trained by metric learning, shown in the right-hand figure, exhibit clear gaps between the inter-class predictions. This means the predictions with dropout are less likely to result in an inaccurate prediction and even reduce the variance of dropout prediction trials. The example shown in Figure 2 has eight dropout predictions, three of which are classified to an inaccurate class when no metric learning is applied compared to only one inaccurate prediction with metric learning.

3.3 Uncertainty Measurement

Bayesian models such as the Gaussian process (Rasmussen, 2004) provide a powerful tool to identify low-confidence regions of input space. Recently, Dropout (Srivastava et al., 2014), which is used in deep neural networks, has been shown to serve as a Bayesian approximation to represent the model uncertainty in deep learning (Gal and Ghahramani, 2016). Based on this work, we propose a novel information-entropy-based dropout method to measure the model uncertainty in combination with metric learning for text classification. Given an input data instance $\boldsymbol{x}^*$, we assume the corresponding output of our model is y^*. The output computed by our model incorporates a dropout mechanism in its evaluation mode, which means the activations of intermediate layers with Dropout are not reduced by a factor. When we repeat the process k times, we obtain the output vector $\boldsymbol{y}^* = \{y_1^*, \ldots, y_k^*\}$. Note that the outputs are not the same since the output here is generated by applying dropout after the feature representation layer in Figure 1.

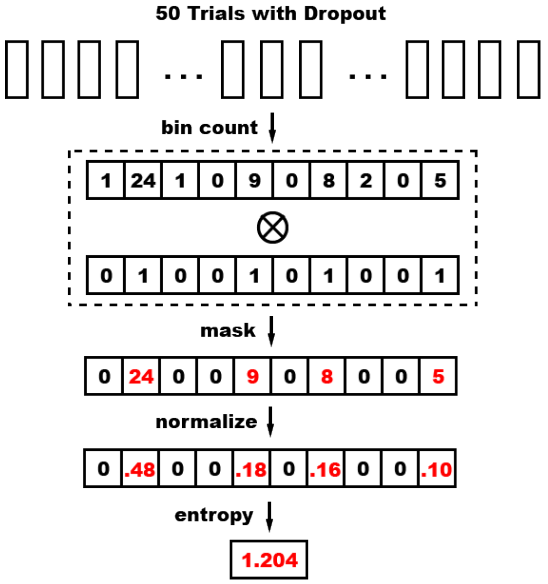


Figure 3: Example of the dropout-entropy method.

Given the output $\boldsymbol{y}^*$ of k trials with Dropout, our proposed uncertainty method has the following four steps, as shown in Figure 3: (1) **Bin count**. We use bin count to calculate the frequency of each class. For example, if the class 2 appears 24 times in the dropout output vector $\boldsymbol{y}^*$, the bin count for class 2 is 24. (2) **Mask**. We use the mask step to avoid random noises in the frequency vector. In this step, we set the largest m elements to have their original values and the remaining ones to zero. The value of m is usually chosen to be $2/3$ of the total class number when the total classes are over 10; otherwise, we just skip the step. (3) **Normalization**. We use the normalization step to calculate the probabilities of each class. (4) **Information entropy**. The information entropy is calculated by $u = -\sum_{i=1}^{c} p_k(i) \log p_k(i)$, where $p_k(i)$ represents the frequency probability of the i-th class in a total k trials and c is the number of classes. We use the entropy value as the uncertainty score here, in which the smaller the entropy value is, the more confident the model is in the output. Take the case in Figure 3 as an example. When the frequency of class 2 is 24, the entropy is 1.204. If the output of the 50 trials all belong to class 2, the entropy becomes 0.401, which means that the model is less uncertain about the predictive results.

4 Experiment

In this section, the performance of the proposed model uncertainty approach is evaluated on multiple real-world document classification data sets.

	Uncertainty Ratio (Micro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
PL-Variance	0.760	0.799(5.10%)	0.815(7.30%)	0.827(8.87%)	0.840(10.52%)
Distance	0.780	0.784(0.59%)	0.787(0.94%)	0.795(1.91%)	0.800(2.58%)
NNGP	0.637	0.659(3.44%)	0.670(5.04%)	0.678(6.44%)	0.689(8.11%)
Dropout	0.758	0.792(4.47%)	0.827(9.08%)	0.851(12.18%)	0.879(15.93%)
Dropout + Metric	0.781	0.823(5.38%)	0.863(10.53%)	0.892(14.31%)	0.921(18.05%)
DE	0.760	0.807(6.25%)	0.849(11.73%)	0.888(16.79%)	0.917(20.70%)
DE + Metric	0.781	**0.835(6.93%)**	**0.878(12.47%)**	**0.918(17.62%)**	**0.944(20.92%)**

	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
PL-Variance	0.751	0.789(5.05%)	0.806(7.24%)	0.818(8.87%)	0.830(10.49%)
Distance	0.773	0.777(0.48%)	0.779(0.81%)	0.786(1.76%)	0.789(2.15%)
NNGP	0.624	0.647(3.56%)	0.657(5.27%)	0.665(6.54%)	0.675(8.13%)
Dropout	0.749	0.781(4.22%)	0.813(8.46%)	0.833(11.10%)	0.860(14.74%)
Dropout + Metric	0.773	0.816(5.47%)	0.853(10.33%)	0.878(13.59%)	0.906(17.14%)
DE	0.752	0.796(5.96%)	0.835(11.05%)	0.872(16.04%)	0.900(19.70%)
DE + Metric	0.774	**0.826(6.70%)**	**0.866(11.97%)**	**0.904(16.87%)**	**0.929(20.02%)**

Table 1: Uncertainty Scores for the 20 NewsGroup Dataset (20 Categories)

After an introduction of the experiment settings in Section 4.1, we compare the performance achieved by the proposed method against those of existing state-of-the-art methods, along with an analysis of the parameter settings and metric learning in Section 4.2. Due to space limitation, the detailed experiment results on different sequence models can be accessed in the full version here[1]. The source code can be downloaded here[2].

4.1 Experimental Setup

In our experiments, all word vectors are initialized by pre-trained Glove (Pennington et al., 2014) word vectors, by default. The word embedding vectors are pre-trained in Wikipedia 2014 with a word vector dimension of 200. We trained all the DNN-based models with a batch size of 32 samples with a momentum of 0.9 and an initial learning rate of 0.001 using the Adam (Kingma and Ba, 2014) optimization algorithm.

4.1.1 Datasets and Labels

We conducted experiments on three publicly available datasets: 1) **20 Newsgroups**[3] (Lang, 1995): The data set is a collection of 20,000 documents, partitioned evenly across 20 different news groups; 2) **IMDb Reviews** (Maas et al., 2011): The data set contains 50,000 popular movie reviews with binary positive or negative labels from the IMDb website; and 3) **Amazon Reviews** (McAuley and Leskovec, 2013): The dataset is a collection of reviews from Amazon spanning the time period from May 1996 to July 2013. We used review data from the Sports and outdoors category, with 272,630 data samples and rating labels from 1 to 5.

For all three data sets, we randomly selected 70% of the data samples as the training set, 10% as the validation set and 20% as the test set.

4.1.2 Evaluation Metrics

In order to answer the question ”*What percentage of data should be transferred to domain experts to achieve an overall accuracy rate above 90%?*”, we measure the classification performance in terms of various uncertainty ratios. Specifically, assuming the entire testing set S has size n and an uncertainty ratio r, we can remove the most uncertain samples S_r from S based on the uncertainty ratio r, where the size of the uncertainty set S_r is $r \cdot n$. We assume the uncertain samples S_r handed to domain experts achieve 100% accuracy. If the uncertainty ratio r equals to 0, the model performs

---

[1] `https://xuczhang.github.io/papers/naacl19_uncertainty_full.pdf`

[2] `https://github.com/xuczhang/UncertainDC`

[3] `http://qwone.com/~jason/20Newsgroups/`

	Uncertainty Ratio (Accuracy, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
PL-Variance	0.878	0.911(3.69%)	0.937(6.70%)	0.955(8.71%)	0.970(**10.42%**)
Distance	0.884	0.893(0.95%)	0.892(0.91%)	0.893(1.04%)	0.895(1.24%)
Dropout	0.880	0.912(3.72%)	0.936(6.43%)	0.957(8.75%)	0.969(10.20%)
Dropout + Metric	0.884	0.917(3.73%)	**0.944**(6.78%)	**0.961**(8.70%)	**0.973**(10.11%)
DE	0.878	0.911(3.70%)	0.937(6.71%)	0.956(8.83%)	0.969(10.33%)
DE + Metric	0.883	**0.918(3.91%)**	**0.944(6.87%)**	**0.961(8.78%)**	**0.973**(10.20%)

	Uncertainty Ratio (F1 Score, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
PL-Variance	0.880	0.913(3.68%)	0.939(6.67%)	0.956(8.65%)	0.971(**10.34%**)
Distance	0.885	0.894(1.07%)	0.898(1.42%)	0.901(1.84%)	0.904(2.13%)
Dropout	0.881	0.914(3.70%)	0.938(6.41%)	0.958(8.67%)	0.971(10.13%)
Dropout + Metric	0.885	0.917(3.70%)	**0.944**(6.74%)	**0.961**(8.67%)	**0.974**(10.06%)
DE	0.880	0.913(3.67%)	0.939(6.67%)	0.957(8.77%)	0.970(10.25%)
DE + Metric	0.884	**0.918(3.88%)**	**0.944(6.83%)**	**0.961(8.73%)**	**0.974**(10.14%)

Table 2: Uncertainty Scores for the IMDb Dataset (2 Categories)

	Uncertainty Ratio (Accuracy, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
PL-Variance	0.700	0.738(5.43%)	0.764(9.14%)	0.784(1.20%)	0.801(14.4%)
Distance	0.697	0.699(0.29%)	0.702(0.72%)	0.704(1.00%)	0.705(1.15%)
Dropout	0.700	0.735(5.00%)	0.764(9.14%)	0.800(14.29%)	0.831(18.71%)
Dropout + Metric	0.710	0.746(5.07%)	0.779(9.72%)	0.815(14.79%)	0.847(19.30%)
DE	0.700	0.739(**5.57%**)	0.773(10.43%)	0.806(15.14%)	0.836(19.43%)
DE + Metric	0.724	**0.764**(5.52%)	**0.800(10.50%)**	**0.834(15.19%)**	**0.866(19.61%)**

Table 3: Uncertainty Scores for the Amazon Dataset (5 Categories)

without uncertainty measurement concerns.

For the binary classification task, we use the accuracy and F1-score to measure the classification performance based on the testing set $S \setminus S_r$ for different uncertainty ratios r. Similarly, for multi-class tasks, we use the micro-F1 and macro-F1 scores utilizing the same settings as for the binary classification.

4.1.3 Comparison Methods

The following methods are included in the performance comparison: 1) Penultimate Layer Variance (PL-Variance). Activations before the softmax layer in a deep neural network always reveal the uncertainty of the prediction (Zaragoza and d'Alche Buc, 1998). As a baseline method, we use the variance of the output of a fully connected layer in Figure 1 as the uncertainty weight. 2) Deep Neural Networks as Gaussian Processes (NNGP) (Lee et al., 2017). This approach applies a Gaussian process to perform a Bayesian inference for deep neural networks, with a computationally efficient pipeline being used to compute the covariance function of the Gaussian process. The default parameter settings in the source code[4] were applied in our experiments. 3) Distance-based Confidence (Distance)(Mandelbaum and Weinshall, 2017). This method assigns confidence scores based on the data embedding compared to the training data. We set its nearest neighbor parameter $k = 10$. 4) Dropout (Gal and Ghahramani, 2016). Here, dropout training in DNNs is treated as an approximation of Bayesian inference in deep Gaussian processes. We set the sample number T as 100 in our experiments. 5) Dropout + Metric. In

---

[4] https://github.com/brain-research/nngp

		Uncertainty Ratio (Micro F1, Improved Ratio)				
		0%	**10%**	**20%**	**30%**	**40%**
Random	**DE**	0.659	0.702(6.47%)	0.748(13.46%)	0.792(20.14%)	0.831(26.03%)
	DE + Metric	0.660	**0.705(6.85%)**	**0.752(13.92%)**	**0.802(21.57%)**	**0.845(28.04%)**
Glove	**DE**	0.760	0.807(6.25%)	0.849(11.73%)	0.888(16.79%)	0.917(20.70%)
	DE + Metric	0.781	**0.835(6.93%)**	**0.878(12.47%)**	**0.918(17.62%)**	**0.944(20.92%)**

Table 4: Embedding vs. No Pre-trained Embedding

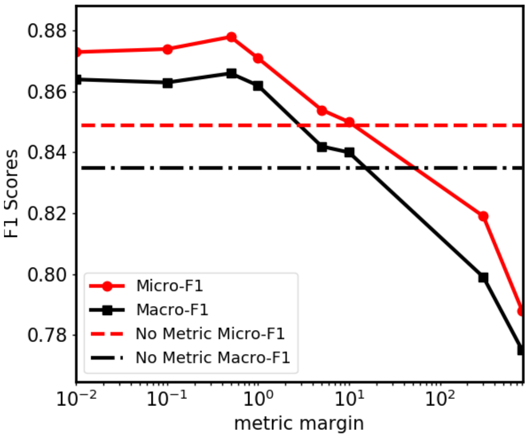


Figure 4: Prediction performance for different metric margin settings.

order to validate the effectiveness of our metric learning, we applied our proposed metric learning method to the Dropout method. The metric margin m and coefficient λ were set as 0.5 and 0.1, respectively. 6) Our proposed method. We evaluate our proposed method in two different settings, Dropout-Entropy alone (DE) and Dropout-Entropy with metric learning (DE + Metric). Here, we set the sample number $T = 100$, coefficient $\lambda = 0.1$ and the metric margin may vary from different data sets.

4.2 Experimental Results

This subsection presents the results of the uncertainty performance comparison and the analysis of the metric learning and parameter settings.

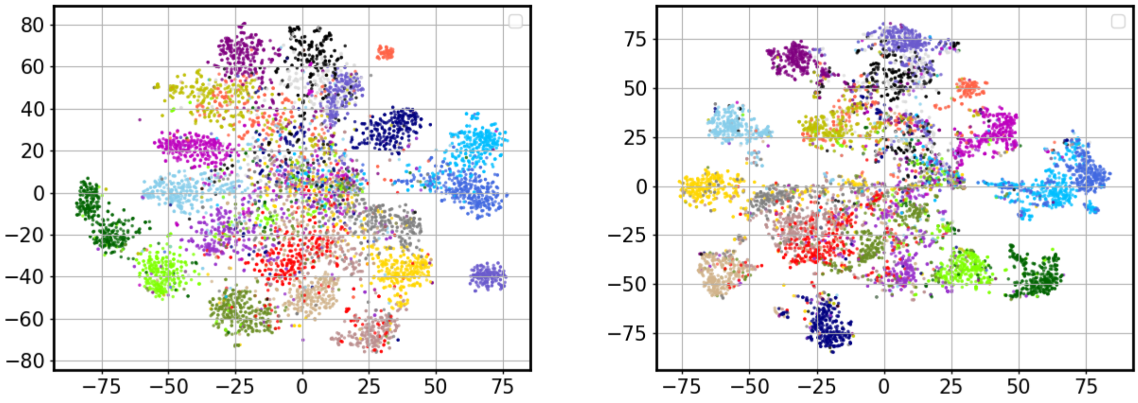
4.2.1 Uncertainty Results

Table 1 shows the Micro-F1 and Macro-F1 scores for ratios of uncertain predictions eliminated ranging from 10 to 40% for the 20NewsGroup data set. To demonstrate its effect, metric learning was also applied to the baseline method Dropout, and our proposed method DE. The improvement ratio compared to the results with no uncertainty elimination, shown in the 0% column, are presented after the F1 scores. Based on these result, we can conclude that: 1) Our proposed method, DE+Metric, significantly improves both the Micro- and Macro-F1 scores when a portion of uncertain predictions are eliminated. For example, the Micro-F1 improves from 0.78 to 0.92 when 30% of the uncertain predictions are eliminated. 2) Comparing the results obtained by DE and DE+Metric, metric learning significantly improves the results obtained for different uncertainty ratio settings. Similar results can be observed when comparing the Dropout and Dropout+Metric. For example, the Micro-F1 scores for Dropout+Metric are around 5% better than the Dropout method alone, boosting them from 0.851 to 0.892, with a 30% uncertainty ratio. 3) The DE method outperforms all the other methods when metric learning is not applied. Specifically, DE is around 4% better than the Dropout method in terms of the Micro-F1 score.

The results for IMDb and Amazon data sets are presented in Table 2 and Table 3. When comparing our proposed model's performance across three data sets, we found that the greater improvements are achieved on multi- instead of binary-class classification data sets. One possible explanation is that a comparatively large portion of multi-class features are close to the decision boundary in the feature space. Through the metric learning strategy of minimizing intra-class distance while maxmizing the inter-class instances, the feature distance between the inter-class predictions is enlarged and the quality of embeddings is greatly enhanced.

4.2.2 Analysis of Metric Learning

The impact of metric learning on feature representation is analyzed in this section. Figure 5 shows the 300-dimension feature representations



(a) No Metric Learning

(b) Metric Margin m = 10

Figure 5: Feature visualization of 20 NewsGroup testing data set in two dimensions by t-SNE algorithm.

for the 20 NewsGroup testing data set, with Figure 5(a) presenting the features trained without metric learning and Figure 5(b) those trained by metric learning with a margin parameter m=10. We used the t-SNE algorithm (Maaten and Hinton, 2008) to visualize the high dimensional features in the form of two dimensional images. From the results, we can clearly see that the distances between the inter-classes are significantly enlarged compared to the features trained without metric learning shown in Figure 5(a). This enlarged inter-class spacing means that dropout-based uncertainty methods have smaller prediction variances in case their dropout prediction trials are accurate.

4.2.3 Parameter Analysis

The impact of the metric margin and word embeddings are discussed in this section.

Metric Margin. Figure 4 shows the impact of metric margin parameters, ranging from 0 to 800 on the 20 NewsGroup data set with a 20% uncertainty ratio. From the results, we can conclude that: (1) The prediction performance is not sensitive to the point at which the metric margin parameter is set as long as its value is not extremely large. (2) Compared to the model trained with no metric learning, our methods consistently achieve better performance when the metric margin is set no larger than 10. When the metric margin is too large, however, the prediction cross-entropy loss is hard to minimize and thus dampens the overall prediction performance. (3) The results of Macro-F1 are similar to Micro-F1 with relatively small scores.

Impact of Word Embedding. We also analyzed the impact of our proposed methods on different word embedding initialization methods, including random and pre-trained Glove word vectors in 200 dimensions. Table 4 shows the results of Micro-F1 for the different uncertainty ratios. We can observe that: 1) The performance of Glove-based methods are around 15% better than that of the randomly initialized methods for different uncertainty ratios. 2) Metric learning based on a Glove initialization generally outperforms a random initialization. For instance, the F1 score of Glove rises by 0.29 when the uncertainty ratio is 20%, while for a random method it only increases by 0.04.

5 Conclusion

In this paper, a DNN-based model is proposed to address the uncertainty mitigation problem in the presence of human involvement in a text classification task. To achieve this, we proposed a dropout-entropy uncertainty measurement method with the metric learning for the feature representation. Extensive experiments on real-world data sets confirmed that our proposed approach dramatically outperforms competing methods, exhibiting a significant improvement in accuracy when a relatively small portion of the uncertainty predictions are handed over to domain experts.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John S. Denker and Yann LeCun. 1990. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, pages 853–859, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org.

Maoguo Gong, Yan Liang, Jiao Shi, Wenping Ma, and Jingjing Ma. 2013. Fuzzy c-means clustering with local information and kernel metric for image segmentation. *IEEE Transactions on Image Processing*, 22(2):573–584.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. 2009. Is that you? metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th international conference on*, pages 498–505. IEEE.

José Miguel Hernández-Lobato and Ryan Adams. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*, pages 193–201. International World Wide Web Conferences Steering Committee.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6405–6416.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2017. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Amit Mandelbaum and Daphna Weinshall. 2017. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*.

Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Vivian Weller. 2017. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Ravina Mithe, Supriya Indalkar, and Nilam Divekar. 2013. Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1):72–75.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. 2015. Deep face recognition. In *BMVC*, volume 1, page 6.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Glenn Shafer and Vladimir Vovk. 2008. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(Mar):371–421.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Volodya Vovk, Alexander Gammerman, and Craig Saunders. 1999. Machine-learning applications of algorithmic randomness.

Hao Wang and Dit-Yan Yeung. 2016. Towards bayesian deep learning: A survey. *arXiv preprint arXiv:1604.01662*.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. 2006. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480.

Jos van der Westhuizen and Joan Lasenby. 2017. Bayesian lstms in medicine. *arXiv preprint arXiv:1706.01242*.

Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. 2003. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528.

Zhixiang (Eddie) Xu, Minmin Chen, Kilian Q. Weinberger, and Fei Sha. 2012. From sbow to dcot marginalized encoders for text representation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1879–1884, New York, NY, USA. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Hugo Zaragoza and Florence d'Alche Buc. 1998. Confidence measures for neural network classifiers. In *Proceedings of the Seventh Int. Conf. Information Processing and Management of Uncertainty in Knowlegde Based Systems*.

Complexity-Weighted Loss and Diverse Reranking for Sentence Simplification

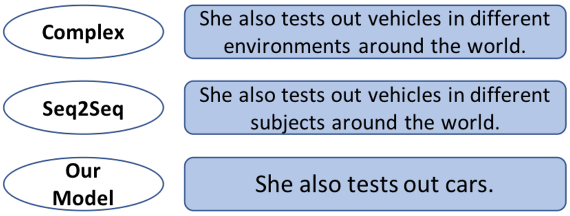
Reno Kriz*, João Sedoc*, Marianna Apidianaki△,
Carolina Zheng*, Gaurav Kumar*, Eleni Miltsakaki†,
and Chris Callison-Burch*
* Computer and Information Science Department, University of Pennsylvania
△ LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay & LLF, Univ. Paris Diderot
† Choosito, Inc.
`{rekriz,joao,gauku,carzheng,ccb}@seas.upenn.edu,`
`marianna@limsi.fr, eleni@choosito.com`

Abstract

Sentence simplification is the task of rewriting texts so they are easier to understand. Recent research has applied sequence-to-sequence (Seq2Seq) models to this task, focusing largely on training-time improvements via reinforcement learning and memory augmentation. One of the main problems with applying generic Seq2Seq models for simplification is that these models tend to copy directly from the original sentence, resulting in outputs that are relatively long and complex. We aim to alleviate this issue through the use of two main techniques. First, we incorporate content word complexities, as predicted with a leveled word complexity model, into our loss function during training. Second, we generate a large set of diverse candidate simplifications at test time, and rerank these to promote fluency, adequacy, and simplicity. Here, we measure simplicity through a novel sentence complexity model. These extensions allow our models to perform competitively with state-of-the-art systems while generating simpler sentences. We report standard automatic and human evaluation metrics.[1]

1 Introduction

Automatic text simplification aims to reduce the complexity of texts and preserve their meaning, making their content more accessible to a broader audience (Saggion, 2017). This process can benefit people with reading disabilities, foreign language learners and young children, and can assist non-experts exploring a new field. Text simplification has gained wide interest in recent years due to its relevance for NLP tasks. Simplifying text during preprocessing can improve the performance of syntactic parsers (Chandrasekar et al., 1996) and semantic role labelers (Vickrey and Koller, 2008; Woodsend and Lapata, 2014), and can improve the grammaticality (fluency) and meaning preservation (adequacy) of translation output (Štajner and Popovic, 2016).



She also tests out vehicles in different environments around the world.

She also tests out vehicles in different subjects around the world.

She also tests out cars.

Figure 1: Example comparison of a simplification generated by a standard Seq2Seq model vs. our model.

Most text simplification work has approached the task as a monolingual machine translation problem (Woodsend and Lapata, 2011; Narayan and Gardent, 2014). Once viewed as such, a natural approach is to use sequence-to-sequence (Seq2Seq) models, which have shown state-of-the-art performance on a variety of NLP tasks, including machine translation (Vaswani et al., 2017) and dialogue systems (Vinyals and Le, 2015).

One of the main limitations in applying standard Seq2Seq models to simplification is that these models tend to copy directly from the original complex sentence too often, as this is the most common operation in simplification. Several recent efforts have attempted to alleviate this problem using reinforcement learning (Zhang and Lapata, 2017) and memory augmentation (Zhao et al., 2018), but these systems often still produce outputs that are longer than the reference sentences. To avoid this problem, we propose to extend the generic Seq2Seq framework at both training and inference time by encouraging the model to choose simpler content words, and by effectively choosing an output based on a large set of can-

---

[1] Our code is available in our fork of Sockeye (Hieber et al., 2017) at https://github.com/rekriz11/sockeye-recipes.

Proceedings of NAACL-HLT 2019, pages 3137–3147
Minneapolis, Minnesota, June 2 - June 7, 2019.

didate simplifications. The main contributions of this paper can be summarized as follows:

- We propose a custom loss function to replace standard cross entropy probabilities during training, which takes into account the complexity of content words.
- We include a similarity penalty at inference time to generate more diverse simplifications, and we further cluster similar sentences together to remove highly similar candidates.
- We develop methods to rerank candidate simplifications to promote fluency, adequacy, and simplicity, helping the model choose the best option from a diverse set of sentences.

An analysis of each individual components reveals that of the three contributions, reranking simplifications at post-decoding stage brings about the largest benefit for the simplification system. We compare our model to several state-of-the-art systems in both an automatic and human evaluation settings, and show that the generated simple sentences are shorter and simpler, while remaining competitive with respect to fluency and adequacy. We also include a detailed error analysis to explain where the model currently falls short and provide suggestions for addressing these issues.

2 Related Work

Text simplification has often been addressed as a monolingual translation process, which generates a simplified version of a complex text. Zhu et al. (2010) employ a tree-based translation model and consider sentence splitting, deletion, reordering, and substitution. Coster and Kauchak (2011) use a Phrase-Based Machine Translation (PBMT) system with support for deleting phrases, while Wubben et al. (2012) extend a PBMT system with a reranking heuristic (PBMT-R). Woodsend and Lapata (2011) propose a model based on a quasi-synchronous grammar, a formalism able to capture structural mismatches and complex rewrite operations. Narayan and Gardent (2014) combine a sentence splitting and deletion model with PBMT-R. This model has been shown to perform competitively with neural models on automatic metrics, though it is outperformed using human judgments (Zhang and Lapata, 2017).

In recent work, Seq2Seq models are widely used for sequence transduction tasks such as machine translation (Sutskever et al., 2014; Luong et al., 2015), conversation agents (Vinyals and Le, 2015), summarization (Nallapati et al., 2016), etc. Initial Seq2Seq models consisted of a Recurrent Neural Network (RNN) that encodes the source sentence $\mathbf{x}$ to a hidden vector of a fixed dimension, followed by another RNN that uses this hidden representation to generate the target sentence $\mathbf{y}$. The two RNNs are then trained jointly to maximize the conditional probability of the target sentence given the source sentence, i.e. $P(\mathbf{y}|\mathbf{x})$. Other works have since extended this framework to include attention mechanisms (Luong et al., 2015) and transformer networks (Vaswani et al., 2017).[2] Nisioi et al. (2017) was the first major application of Seq2Seq models to text simplification, applying a standard encoder-decoder approach with attention and beam search. Vu et al. (2018) extended this framework to incorporate memory augmentation, which simultaneously performs lexical and syntactic simplification, allowing them to outperform standard Seq2Seq models.

There are two main Seq2Seq models we will compare to in this work, along with the statistical model from Narayan and Gardent (2014). Zhang and Lapata (2017) proposed DRESS (Deep REinforcement Sentence Simplification), a Seq2Seq model that uses a reinforcement learning framework at training time to reward the model for producing sentences that score high on fluency, adequacy, and simplicity. This work showed state-of-the-art results on human evaluation. However, the sentences generated by this model are in general longer than the reference simplifications. Zhao et al. (2018) proposed DMASS (Deep Memory Augmented Sentence Simplification), a multi-layer, multi-head attention transformer architecture which also integrates simplification rules. This work has been shown to get state-of-the-art results in an automatic evaluation, training on the WikiLarge dataset introduced by Zhang and Lapata (2017). Zhao et al. (2018), however, does not perform a human evaluation, and restricting evaluation to automatic metrics is generally insufficient for comparing simplification models. Our model, in comparison, is able to generate shorter and simpler sentences according to Flesch-Kincaid grade level (Kincaid et al., 1975) and human judgments, and provide a comprehensive analysis using human evaluation and a qualitative error analysis.

---

[2] For a detailed description of Seq2Seq models, please see (Sutskever et al., 2014).

3 Seq2Seq Approach

3.1 Complexity-Weighted Loss Function

Standard Seq2Seq models use cross entropy as the loss function at training time. This only takes into account how similar our generated tokens are to those in the reference simple sentence, and not the complexity of said tokens. Therefore, we first develop a model to predict word complexities, and incorporate these into a custom loss function.

3.1.1 Word Complexity Prediction

Extending the complex word identification model of Kriz et al. (2018), we train a linear regression model using length, number of syllables, and word frequency; we also include Word2Vec embeddings (Mikolov et al., 2013). To collect data for this task, we consider the Newsela corpus, a collection of 1,840 news articles written by professional editors at 5 reading levels (Xu et al., 2015).[3] We extract word counts in each of the five levels; in this dataset, we denote 4 as the original complex document, 3 as the least simplified re-write, and 0 as the most simplified re-write. We propose using Algorithm 1 to obtain the complexity label for each word w, where l_w represents the level given to the word, and c_{w_i} represents the number of times that word occurs in level i.

Algorithm 1 Word Complexity Data Collection

```
procedure DATA COLLECTION
    $l_w \leftarrow 4$
    for $i \in \{3, 0\}$ do
        if $c_{w_i} \geq 0.7 * c_{w_{i+1}}$ then
            if $c_{w_i} \geq 0.4 * c_{w_4}$ then
                $l_w \leftarrow i$
    return $l_w$
```

Here, we initially label the word with the most complex level, 4. If at least 70% of the instances of this word is preserved in level 3, we reassign the label as level 3; if the label was changed, we then do this again for progressively simpler levels.

As examples, Algorithm 1 labels "pray", "sign", and "ends" with complexity level 0, and "proliferation", "consensus", and "emboldened" with complexity level 4. We split the data extracted from Algorithm 1 into Train, Validation and Test sets (90%, 5% and 5%, respectively, and use them for training and evaluating the complexity prediction model. [4]

Model	Correlation	MSE
Frequency Baseline	-0.031	1.90
Length Baseline	0.344	1.51
LinReg	**0.659**	**0.92**

Table 1: Pearson Correlation and Overall Mean Squared Error (MSE) of the word-level complexity prediction model (LinReg). Comparison to length-based and frequency-based baselines.

We report the Mean Squared Error (MSE) and Pearson correlation on our test set in Table 1.[5] We compare our model to two baselines, which predict complexity using log Google n-grams frequency (Brants and Franz, 2006) and word length, respectively. For these baselines, we calculate the minimum and maximum values for words in the training set, and then normalize the values for words in the test set.

3.1.2 Loss Function

We propose a metric that modifies cross entropy loss to upweight simple words while downweighting more complex words. More formally, the probabilities of our simplified loss function can be generated by the process described in Algorithm 2. Since our word complexities are originally from 0 to 4, with 4 being the most complex, we need to reverse this ordering and add one, so that more complex words and non-content words are not given zero probability. In this algorithm, we denote the original probability vector as **CE**, our vocabulary as **V**, the predicted word complexity of a word v as $score_v$, the resulting weight for a word as w_v, and our resulting weights as **SCE**, which we then normalize and convert back to logits.

Here, α is a parameter we can tune during experimentation. Note that we only upweight simple content words, not stopwords or entities.

3.2 Diverse Candidate Simplifications

To increase the diversity of our candidate simplifications, we apply a beam search scoring modification proposed in Li et al. (2016). In standard

---

[3] Newsela is an education company that provides reading materials for students in elementary through high school. The Newsela corpus can be requested at https://newsela.com/data/

[4] Note that we also tried continuous rather than discrete labels for words by averaging frequencies, but found that this increased the noise in the data. For example, "the" and "dog" were incorrectly labeled as level 2 instead of 0, since these words are seen frequently across all levels.

[5] We report MSE results by level in the appendix.

Algorithm 2 Simplified Loss Function

```
procedure SIMPLIFIED LOSS
    CE ← softmax($logits_{CE}$)
    for $v \in \mathbf{V}$ do
        $score_v \leftarrow WordComplexity(v)$
        if $v$ is a content word then
            $w_v \leftarrow (4 - s_v) + 1$
        else
            $w_v \leftarrow 1$
    $w_v \leftarrow \left(\frac{w_v}{\sum_{v \in V} w_v}\right)^{\alpha}$ for $v \in \mathbf{V}$
    SCE ← CE · w
  return SCE
```

beam search with a beam width of b, given the b hypotheses at time $t-1$, the next set of hypotheses is generated by first selecting the top b candidate expansions from each hypothesis. These $b \times b$ hypotheses are then ranked by the joint probabilities of their sequence of output tokens, and the top b according to this ranking are chosen.

We observe that candidate expansions from a single parent hypothesis tend to dominate the search space over time, even with a large beam. To increase diversity, we apply a penalty term based on the rank of a generated token among the b candidate tokens from its parent hypothesis.

If Y^j_{t-1} is the j^{th} top hypothesis at time $t-1$, $j \in [1..b]$, and $y^{j,j'}_t$ is a candidate token generated from Y^j_{t-1}, where $j' \in [1..b]$ represents the rank of this particular token among its siblings, then our modified scoring function is as follows (here, δ is a parameter we can tune during experimentation):

$$S(Y^j_{t-1}, y^{j,j'}_t) = \log p(y^j_1, \ldots, y^j_{t-1}, y^{j,j'}_t | x) - j' * \delta \quad (1)$$

Extending the work of Li et al. (2016), to further increase the distance between candidate simplifications, we can cluster similar sentences after decoding. To do this, we convert each candidate into a document embedding using Paragraph Vector (Le and Mikolov, 2014), cluster the vector representations using k-means, and select the sentence nearest to the centroids. This allows us to group similar sentences together, and only consider candidates that are relatively more different.

Model	Correlation	MSE
Length Baseline	0.503	3.72
CNN (ours)	**0.650**	**1.13**

Table 2: Pearson Correlation and Overall Mean Squared Error (MSE) for the sentence-level complexity prediction model (CNN), compared to a length-based baseline.

3.3 Reranking Diverse Candidates

Generating diverse sentences is helpful only if we are able to effectively rerank them in a way that promotes simpler sentences while preserving fluency and adequacy. To do this, we propose three ranking metrics for each sentence i:

- **Fluency** (f_i): We calculate the perplexity based on a 5-gram language model trained on English Gigaword v.5 (Parker et al., 2011) using KenLM (Heafield, 2011).
- **Adequacy** (a_i): We generate Paragraph Vector representations (Le and Mikolov, 2014) for the input sentence and each candidate and calculate the cosine similarity.
- **Simplicity** (s_i): We develop a sentence complexity prediction model to predict the overall complexity of each sentence we generate.

To calculate sentence complexity, we modify a Convolutional Neural Network (CNN) for sentence classification (Kim, 2014) to make continuous predictions. We use aligned sentences from the Newsela corpus (Xu et al., 2015) as training data, labeling each with the complexity level from which it came.[6] As with the word complexity prediction model, we report MSE and Pearson correlation on a held-out test set in Table 2.[7]

We normalize each individual score between 0 and 1, and calculate a final score as follows:

$$score_i = \beta_f f_i + \beta_a a_i + \beta_s s_i \quad (2)$$

We tune these weights (β) on our validation data during experimentation to find the most appropriate combinations of reranking metrics. Examples of improvements resulting from the including each of our contributions are shown in Table 3.

4 Experiments

4.1 Data

We train our models on the Newsela Corpus. In previous work, models were mainly trained on the parallel Wikipedia corpus (PWKP) consisting of paired sentences from English Wikipedia

---

[6] We respect the train/test splits described in Section 4.1.

[7] We report MSE results by level in the appendix.

and Simple Wikipedia (Zhu et al., 2010), or the extended WikiLarge corpus (Zhang and Lapata, 2017). We choose to instead use Newsela, because it was found that 50% of the sentences in Simple Wikipedia are either not simpler or not aligned correctly, while Newsela has higher-quality simplifications (Xu et al., 2015).

As in Zhang and Lapata (2017), we exclude sentence pairs corresponding to levels 4-3, 3-2, 2-1, and 1-0, where the simple and complex sentences are just one level apart, as these are too close in complexity. After this filtering, we are left with 94,208 training, 1,129 validation, and 1,077 test sentence pairs; these splits are the same as Zhang and Lapata (2017). We preprocess our data by tokenizing and replacing named entities using CoreNLP (Manning et al., 2014).

4.2 Training Details

For our experiments, we use Sockeye, an open source Seq2Seq framework built on Apache MXNet (Hieber et al., 2017; Chen et al., 2015). In this model, we use LSTMs with attention for both our encoder and decoder models with 256 hidden units, and two hidden layers. We attempt to match the hyperparameters described in Zhang and Lapata (2017) as closely as possible; as such, we use 300-dimensional pretrained GloVe word embeddings (Pennington et al., 2014), and Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. We ran our models for 30 epochs.[8]

During training, we use our complexity-weighted loss function, with $\alpha = 2$; for our baseline models, we use cross-entropy loss. At inference time, where appropriate, we set the beam size $b = 100$, and the similarity penalty $\delta = 1.0$. After inference, we set the number of clusters to 20, and we compare two separate reranking weightings: one which uses fluency, adequacy, and simplicity (FAS), where $\beta_f = \beta_a = \beta_s = \frac{1}{3}$; and one which uses only fluency and adequacy (FA), where $\beta_f = \beta_a = \frac{1}{2}$ and $\beta_s = 0$.

4.3 Baselines and Models

We compare our models to the following baselines:

- **Hybrid** performs sentence splitting and deletion before simplifying with a phrase-based machine translation system (Narayan and Gardent, 2014).
- **DRESS** is a Seq2Seq model trained with reinforcement learning which integrates lexical simplifications (Zhang and Lapata, 2017).[9]
- **DMASS** is a Seq2Seq model which integrates the transformer architecture and additional simplifying paraphrase rules (Zhao et al., 2018).[10]

We also present results on several variations of our models, to isolate the effect of each individual improvement. **S2S** is a standard sequence-to-sequence model with attention and greedy search. **S2S-Loss** is trained using our complexity-weighted loss function and greedy search. **S2S-FA** uses beam search, where we rerank all sentences using fluency and adequacy (FA weights). **S2S-Cluster-FA** clusters the sentences before reranking using FA weights. **S2S-Diverse-FA** uses diversified beam search, reranking using FA weights. **S2S-All-FAS** uses all contributions, reranking using fluency, adequacy, and simplicity (FAS weights). Finally, **S2S-All-FA** integrates all modifications we propose, and reranks using FA weights.

5 Results

In this section, we compare the baseline models and various configurations of our model with both standard automatic simplification metrics and a human evaluation. We show qualitative examples where each of our contributions improves the generated simplification in Table 3.

5.1 Automatic Evaluation

Following previous work (Zhang and Lapata, 2017; Zhao et al., 2018), we use SARI as our main automatic metric for evaluation (Xu et al., 2016).[11] Specifically, SARI calculates how often a generated sentence correctly keeps, inserts, and deletes n-grams from the complex sentence, using the reference simple standard as the gold-standard, where $1 \leq n \leq 4$. Note that we do not use

---

[8] All non-default hyperparameters can be found in the Appendix.

[9] For Hybrid and DRESS, we use the generated outputs provided in Zhang and Lapata (2017). We made a significant effort to rerun the code for DRESS, but were unable to do so.

[10] For DMASS, we ran the authors' code on our data splits from Newsela, in collaboration with the first author to ensure an accurate comparison.

[11] To calculate SARI, we use the original script provided by (Xu et al., 2016).

Complex Sentence	Model 1	Model 1 Sentence	Model 2	Model 2 Sentence
Mary travels between two offices.	S2S	Mary is a professor at the park.	S2S-Loss	Mary goes between two offices.
Their fatigue changes their voices, but they're still on the freedom highway.	S2S	Their condition changes their voices, but they're still on the freedom highway.	S2S-FA	Their fatigue changes their voices.
Just until recently, the education system had banned Islamic headscarves in schools and made schoolchildren recite a pledge of allegiance.	S2S-FA	The education system had banned Islamic law.	S2S-Cluster-FA	Only until recently , the education system had banned Islamic hijab in schools.
Police used tear gas, dogs and clubs on the unarmed protesters.	S2S-FA	Police used tear gas and dogs on the unarmed protesters.	S2S-Diverse-FA	They used tear gas and dogs.

Table 3: Example sentences where each component of our model improved the output sentence, compared to a model that does not use that component.

Model	SARI	Oracle
Hybrid	33.27	–
DRESS	36.00	–
DMASS	34.35	–
S2S	36.32	–
S2S-Loss	36.03	–
S2S-FA	36.47	54.01
S2S-Cluster-FA	**37.22**	50.36
S2S-Diverse-FA	35.36	52.65
S2S-All-FAS	36.30	50.40
S2S-All-FA	**37.11**	50.40

Table 4: Comparison of our models to baselines and state-of-the-art models using SARI. We also include oracle SARI scores (Oracle), given a perfect reranker. S2S-All-FA is significantly better than the DMASS and Hybrid baselines using a student t-test ($p < 0.05$).

Model	Len	FKGL	TER	Ins	Edit
Complex	23.1	11.14	0	0	–
Hybrid	12.4	7.82	0.49	0.01	–
DRESS	14.4	7.60	0.44	0.07	–
DMASS	15.1	7.40	0.59	0.28	–
S2S	16.1	7.91	0.41	0.23	–
S2S-Loss	16.4	8.11	0.40	0.31	–
S2S-FA	7.6	6.42	0.73	0.01	7.28
S2S-Cluster-FA	9.1	6.49	0.68	0.05	7.55
S2S-Diverse-FA	7.5	5.97	0.78	0.07	8.22
S2S-All-FAS	9.1	5.37	0.68	0.05	7.56
S2S-All-FA	10.8	6.42	0.61	0.07	7.56
Reference	12.8	6.90	0.67	0.42	–

Table 5: Average sentence length, FKGL, TER score compared to input, and number of insertions. We also calculate average edit distance (Edit) between candidate sentences for applicable models.

BLEU (Papineni et al., 2002) for evaluation; even though it correlates better with fluency than SARI, Sulem et al. (2018) recently showed that BLEU often negatively correlates with simplicity on the task of sentence splitting. We also calculate oracle SARI, where appropriate, to show the score we could achieve if we had a perfect reranking model. Our results are reported in Table 4.

Our best models outperform previous state-of-the-art systems, as measured by SARI. Table 4 also shows that, when used separately, reranking and clustering result in improvements on this metric. Our loss and diverse beam search methods have more ambiguous effects, especially when combined with the former two; note however that including diversity before clustering does slightly improve the oracle SARI score.

We calculate several descriptive statistics on the generated sentences and report the results in Table 5. We observe that our models produce sentences that are much shorter and lower reading level, according to Flesch-Kincaid grade level (FKGL) (Kincaid et al., 1975), while making more changes to the original sentence, according to Translation Error Rate (TER) (Snover et al., 2006). In addition, we see that the customized loss function increases the number of insertions made, while both the diversified beam search and clustering techniques individually increase the distance between sentence candidates.

5.2 Human Evaluation

While SARI has been shown to correlate with human judgments on simplicity, it only weakly cor-

Model	Fluency	Adequacy	Simplicity	All
Hybrid	2.79*	2.76	2.88*	2.81*
DRESS	**3.50**	**3.11****	3.03	**3.21****
DMASS	2.59*	2.15*	2.50*	2.41*
S2S-All-FAS	3.35	2.50*	**3.11**	2.99
S2S-All-FA	3.38	2.66	**3.08**	3.04
Reference	3.82*	3.23*	3.29*	3.45*

Table 6: Average ratings of crowdsourced human judgments on fluency, adequacy and complexity. Ratings significantly different from S2S-All-FA are marked with * ($p < 0.05$); statistical significance tests were calculated using a student t-test. We provide 95% confidence intervals for each rating in the appendix.

relates with judgments on fluency and adequacy (Xu et al., 2016). Furthermore, SARI only considers simplifications at the word level, while we believe that a simplification metric should also take into account sentence structure complexity. We plan to investigate this further in future work.

Due to the current perceived limitations of automatic metrics, we also choose to elicit human judgments on 200 randomly selected sentences to determine the relative overall quality of our simplifications. For our first evaluation, we ask native English speakers on Amazon Mechanical Turk to evaluate the fluency, adequacy, and simplicity of sentences generated by our systems and the baselines, similar to Zhang and Lapata (2017). Each annotator rated these aspects on a 5-point Likert Scale. These results are found in Table 6.[12]

As we can see, our best models substantially outperform the Hybrid and DMASS systems. Note that DMASS performs the worst, potentially because the transformer model is a more complex model that requires more training data to work properly. Comparing to DRESS, our models generate simpler sentences, but DRESS better preserves the meaning of the original sentence.

To further investigate why this is the case, we know from Table 5 that sentences generated by our model are overall shorter than other models, which also corresponds to higher TER scores. Napoles et al. (2011) notes that on sentence compression, longer sentences are perceived by human annotators to preserve more meaning than shorter sentences, controlling for quality. Thus, the drop in human-judged adequacy may be related to our sentences' relatively short lengths.

To test that this observation also holds true for simplicity, we took the candidates generated by our best model, and after reranking them as before, we selected three sets of sentences:

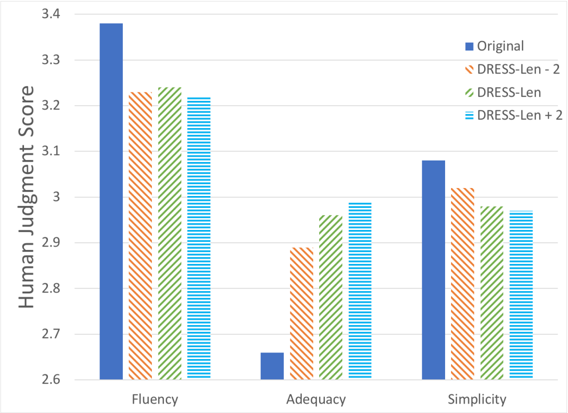


Figure 2: Effect of length on human judgments.

- **MATCH-Dress0**: Highest ranked sentence with length closest to that of DRESS (DRESS-Len); average length is 14.10.
- **MATCH-Dress+2**: Highest ranked sentence with length closest to (DRESS-Len + 2); average length is 15.32.
- **MATCH-Dress-2**: Highest ranked sentence with length closest to (DRESS-Len - 2); average length is 12.61.

The average fluency, adequacy, and simplicity from human judgments on these new sentences are shown in Figure 2, along with those ranked highest by our best model (Original). As expected, meaning preservation does substantially increase as we increase the average sentence length, while simplicity decreases. Interestingly, fluency also decreases as sentence length increases; this is likely due to our higher-ranked sentences having greater fluency, as defined by language model perplexity.

---

[12] We present the instructions for all of our human evaluations in the appendix.

6 Error Analysis

To gain insight in what aspects of the simplification process are challenging to our model, we present the most recurring types of errors from our test set.

6.1 Examples of Error Types

1. Long and complex sentences with multiple clauses
    (a) *Complex*: Turkey has long enshrined the secular ideals of founding father Mustafa Kemal Ataturk, particularly in an education system that until recently banned Islamic headscarves in schools and made schoolchildren begin the day reciting an oath of allegiance to Ataturk's legacy.
    Reference: Schools in Turkey had banned headscarves.
    Simple: They made schoolchildren to Ataturk's history.
    (b) *Complex*: And Wal-Mart, which imports more fruits and vegetables from Mexico than any other U.S. company, announced its effort to force improvements up and down its supply chain.
    Reference: Experts said Wal-Mart is an important company.
    Simple: Wal-Mart used more fruits and vegetables from the company.

2. Need for anaphora resolution
    (a) *Complex*: He is the creative director of Rethink Leisure & Entertainment , which is working on several projects in China and elsewhere in Asia .
    Reference: He is with Rethink Leisure & Entertainment.
    Simple: He is working on several projects in China.
    (b) *Complex*: Teachers there say Richie reads like a high school student.
    Reference: He reads like a high school student.
    Simple: Richie says he is a high school student.

3. Simplifying the wrong part of the sentence
    (a) *Complex*: Parks deliberately maintained her image as shy and proper, said Adrienne Cannon, an expert on African-American history.
    Reference: Adrienne Cannon studies African-American history.
    Simple: She is an expert on African-American history.
    (b) *Complex*: His father owned the home when the lava flowed slowly to the coast.
    Reference: His father still owned the home.
    Simple: The river cut slowly to the coast.

4. Poor substitution due to word embeddings proximity
    (a) *Complex*: Just until recently, the education system had banned Islamic **headscarves** in schools and made schoolchildren begin the day reciting a pledge of allegiance to Ataturk's legacy.
    Reference: Schools in Turkey had banned headscarves.
    Simple: Just until recently, the education officials had banned Islamic **flags**.
    (b) *Complex*: In Beijing kite circles, Fei is widely known as the elder **statesman**.
    Reference: In Beijing, Fei is widely known as an expert.
    Simple: In Beijing, Fei is considered a **doctor**.

5. Training confusion due to misalignments in training corpus
    (a) *Complex*: He heard the applause.
    Reference: The winning word was "magician."
    Simple: But he heard the song.
    (b) *Complex*: More than 300,000 students across the country signed up to participate.
    Reference: Rogers Middle School took part in the program.
    Simple: There were more than 300,000 students.

6. Acceptable simplifications that were not available in the training
    (a) *Complex*: He later tried to buy his freedom.
    Reference: Henson tried to buy his freedom later.
    Simple: The man tried to buy his freedom.
    (b) *Complex*: Middle school was a rough couple of years for Talmus' daughter, Lili Smith.
    Reference: She had a hard time in middle school.
    Simple: School was a rough couple of years.

6.2 Error Discussion

Attempting to rewrite very long and complex sentences resulted to consistent errors, as shown in 1a and 1b. This observation in combination with the examples of mis-alignments in the training corpus (5a and 5b) indicate that we either need to improve the alignments such the model can capture that the simplification process involves in many cases splitting a sentence and then simplifying or train to learn when to split first and then attempt rewriting.

The next two types of errors show failure in capturing discourse level meaning: a) errors due to failed pronoun resolution, shown in 2a and 2b, and b) errors due to the most important part of the sentence being left out, shown in 3b and 3b. In these cases, the sentences were not bad, but the information was assigned to the wrong referent, or important meaning was left out. In 4a and 4b, the substitution is clearly semantically related to the target, but changes the meaning. Finally, there were examples of acceptable simplifications, as in 6a and 6b, that were classified as errors because they were not in the gold data. We provide additional examples for each error category in the appendix.

To improve the performance of future models, we see several options. We can improve the original alignments within the Newsela corpus, particularly in the case where sentences are split. Prior to simplification, we can use additional context around the sentences to perform anaphora resolution; at this point, we can also learn when to perform sentence splitting; this has been done in the Hybrid model (Narayan and Gardent, 2014), but has not yet been incorporated into neural models. Finally, we can use syntactic information to ensure the main clause of a sentence is not removed.

7 Conclusion

In this paper, we present a novel Seq2Seq framework for sentence simplification. We contribute three major improvements over generic Seq2Seq models: a complexity-weighted loss function to encourage the model to choose simpler words; a similarity penalty during inference and clustering post-inference, to generate candidate simplifications with significant differences; and a reranking system to select the simplification that promotes both fluency and adequacy. Our model outperforms previous state-of-the-art systems using SARI, the standard metric for simplification. More importantly, while other previous models generate relatively long sentences, our model is able to generate shorter and simpler sentences, while remaining competitive regarding human-evaluated fluency and adequacy. Finally, we provide a qualitative analysis of where our different contributions improve performance, the effect of length on human-evaluated meaning preservation, and the current shortcomings of our model as insights for future research.

Generating diverse outputs from Seq2Seq models could be used in a variety of NLP tasks, such as chatbots (Shao et al., 2017), image captioning (Vijayakumar et al., 2018), and story generation (Fan et al., 2018). In addition, the proposed techniques can also be extremely helpful in leveled and personalized text simplification, where the goal is to generate different sentences based on who is requesting the simplification.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback on this work. We would also like to thank Devanshu Jain, Shyam Upadhyay, and Dan Roth for their feedback on the post-decoding aspect of this work, as well as Anne Cocos and Daphne Ippolito for their insightful comments during proofreading.

This material is based in part on research sponsored by DARPA under grant number HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

The work has also been supported by the French National Research Agency under project ANR-16-CE33-0013. This research was partially supported by João Sedoc's Microsoft Research Dissertation Grant. Finally, we gratefully acknowledge the support of NSF-SBIR grant 1456186.

References

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. In *LDC2006T13*, Philadelphia, Pennsylvania. Linguistic Data Consortium.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *CoRR*, abs/1512.01274.

Will Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, UK.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

J. Peter Kincaid, Robert P. Fishburne, Richard E. L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel ; research branch report 8-75.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Reno Kriz, Eleni Miltsakaki, Marianna Apidianaki, and Chris Callison-Burch. 2018. Simplification using paraphrases and context-based lexical substitution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 207–217, New Orleans, Louisiana. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages 1188–1196. JMLR.org.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A Simple, Fast Diverse Decoding Algorithm for Neural Generation. *CoRR*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97, Portland, Oregon. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 435–445, Baltimore, Maryland. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07. DVD. *Philadelphia: Linguistic Data Consortium*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Horacio Saggion. 2017. *Automatic Text Simplification*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA.

Sanja Štajner and Maja Popovic. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Bleu is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Brussels, Belgium. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Neural Information Processing Systems*, Montreal, Canada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Neural Information Processing Systems*, Long Beach, CA.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, Columbus, Ohio. Association for Computational Linguistics.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop*.

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence simplification with memory-augmented neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 79–85.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2014. Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, 51:133–164.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3(1):283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4(1):401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594. Association for Computational Linguistics.

Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. 2018. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 EMNLP Conference*, pages 3164–3173, Brussels, Belgium.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China.

Predicting Helpful Posts in Open-Ended Discussion Forums: A Neural Architecture

Kishaloy Halder[1,2] **Min-Yen Kan**[1,2] **Kazunari Sugiyama**[1]

[1]School of Computing, National University of Singapore
[2]Institute for Application of Learning Science and Educational Technology,
National University of Singapore
{kishaloy, kanmy, sugiyama}@comp.nus.edu.sg

Abstract

Users participate in online discussion forums to learn from others and share their knowledge with the community. They often start a thread with a question or by sharing their new findings on a certain topic. Unlike in Community Question Answering, where questions are mostly factoid based, we find that the threads in a forum are often open-ended (*e.g.*, asking for recommendations from others) without a definitive correct answer. We thus address the task of identifying helpful posts in a forum thread to help users comprehend long-running discussion threads, which often contain repetitive or irrelevant posts. We propose a recurrent neural network based architecture to model (i) the relevance of a post regarding the original post starting the thread, and (ii) the novelty it brings to the discussion, compared to the previous posts in the thread. Experimental results on five different types of online forum datasets show that our model significantly outperforms the state-of-the-art neural network models for text classification.

1 Introduction

Online discussion forums are widely used in many domains such as in generic web content[1], e-health[2], Massive Open Online Courses (MOOCs)[3], and e-commerce, among others. Users participate in these forums to gain knowledge from the collective wisdom of the community. Typically, users start a discussion thread by posting a question or asking others for opinions on a topic. Others then reply to threads relevant to their interests. Importantly, as these forums are indexed by search engines, they need to be discoverable by a wider audience — apart from just registered users — by enabling threads to be found in response to queries.

Due to the open nature of the forums and the various expertise level of users, the posts in the discussion threads vary in helpfulness. To address this, some websites provide actions for users to signal this, as in “Upvote” (`reddit`, `stackoverflow`) and “Highlight” (`coursera`). Such feedback is helpful for identifying important posts among the many. Such feedback rarely comes immediately following new post creation, affecting their visibility to the users (Singh et al., 2017). We can devise technology to proactively identify such helpful posts as they arrive, in a *helpfulness prediction task*, enabling users to efficiently assess relevance.

We observe that there is a key structural difference between online discussion forums and Community Question Answering (CQA) websites. Figure 1 shows the distribution of normalized helpful votes for the top-5 posts across a popular discussion forum (`reddit`), and a CQA website (`stackoverflow`[4]). In CQA, the vote distribution decays exponentially, indicating that usually there is a single correct answer with the largest number of votes (Omari et al., 2016). In contrast, votes for less helpful posts in discussion forums decay at a much lower rate, suggesting that discussion forum threads are more open-ended.

Table 1 shows a sample thread from `reddit` to understand the dynamics of online discussion. We observe the following two major differences compared to threads in CQA domain: (1) The first post (hereafter, *original post*) is not necessarily a question, but can be personal anecdotes or new findings on a certain topic, attracting more discussion. (2) Instead of searching for a single relevant answer as in CQA, discussion forum users find a post helpful

---

[1] https://www.reddit.com/
[2] https://www.healthboards.com/boards
[3] https://www.coursera.org
[4] https://www.kaggle.com/stackoverflow/stacksample/data

Proceedings of NAACL-HLT 2019, pages 3148–3157
Minneapolis, Minnesota, June 2 - June 7, 2019.

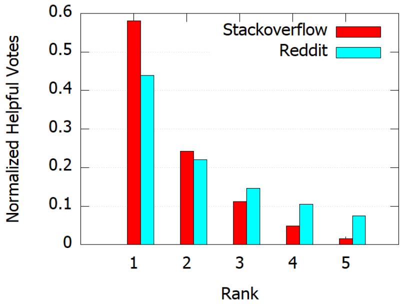


Figure 1: The helpful vote distribution for the top-5 posts across an online discussion forum (`reddit`), and the `stackoverflow` CQA website. The helpful votes decay at a slower rate for `reddit` compared to focused CQA.

Order	Post Text	Helpful?
Original post	I was working yesterday..and my back was bent over and when I got up I felt like I strained my back but now my mind is linking it to my kidney..	Yes
1	I have this and my doc has told me it's muscular and physio might help..	Yes
2	Kidney pain is usually constant and doesn't change when you move, or get better when you change position, from how I understand it..you'll be fine :)	Yes
3	If it happens only when you move there is a big chance it's a muscle spasm, this happens after some physical activities.	No

Table 1: A sample discussion thread from `reddit`. Helpful votes are provided by the website users.

when it introduces some *relevant* (with respect to the original post) and *novel* (*i.e.*, not presented in the earlier posts within the same thread) information. Motivated by these observations, we address helpfulness prediction by considering both the target post and its preceding posts.

We propose a novel neural architecture to predict the helpfulness of a post in a discussion thread. Our approach consists of two components: (1) modeling the *relevance* of a post and (2) determining the *novelty* with respect to the sequence of preceding posts. It combines the output from both components to predict the overall post helpfulness. As recurrent neural networks (RNNs) have shown good performance in sequence modeling tasks (Chung et al., 2014; Sutskever et al., 2014), we apply it to our architecture to model the (i) sequence of words in the post text, and the (ii) sequence of posts in a thread. Our model significantly outperforms other state-of-the-art models across experiments on five varied and large forum datasets. Our main contributions are:

- We reveal the key differences between posts in CQA and online discussion forums;
- We analyze the confounding factors behind the perceived helpfulness of posts in discussion forums. We observe that both *relevance* and *novelty* play important roles in determining the helpfulness of a post;
- We propose a novel neural network architecture to predict the helpfulness by using textual content of a target post as well as sequence of posts preceding it in the thread;
- We compare our model with current neural network classifiers and analyze the factors that influence our model's performance.

2 Related Work

To the best of our knowledge, predicting helpful posts in generic open-ended discussion forums has not been studied before. However, there is significant amounts of related work on similar directions; where researchers evaluate the *quality* (which may not correlate with perceived helpfulness by the community users) of posts in specific domains such as health (Oh et al., 2012; Oh and Worrall, 2013; Beloborodov et al., 2014) and online education (Chandrasekaran et al., 2015; Jenders et al., 2016). External medical resources and thesauri such as UMLS[5] have been used to identify patterns of helpfulness in health (Asghar et al., 2014). In MOOC platforms, apart from the textual content of the forums, additional signals such as user reputation (*e.g.*, average homework scores, number of courses taken) have been used to estimate post quality (Jenders et al., 2016). However, these techniques are tightly coupled with the target domain, and may not be generalizable to new domains.

CQA Answer Quality: Past work has also addressed the evaluation of answer quality in CQA sites (Jeon et al., 2006; Hong and Davison, 2009; Shah and Pomerantz, 2010; **?**; Omari et al., 2016; Li et al., 2015). Typically posed as a classification problem, these use both textual and non-textual feature-based approaches. Since it is quite common for popular questions to attract many potential answers, answer ranking based on perceived quality is another line of approach (Surdeanu et al., 2008; Bian et al., 2008; Wang et al., 2009). Closer to our approach, Omari et al. (2016)

---

[5] `https://www.nlm.nih.gov/research/umls/`

proposed a novelty-based greedy ranking algorithm that depends on a pre-trained parser to identify different propositions, useful for predicting helpfulness. Li et al. (2015) propose a few features for answer quality detection from academic QA sites such as ResearchGate[6]. However this approach does not generalize well since the method uses many website-specific signals such as *reputation scores* for users and their institutions. Additionally, their approach relies on human annotations to identify a few key conversational characteristics in the answers, keeping it from being applied to use cases where scalability and automation are key.

In the CQA answer quality evaluation literature, quality is often measured through the human evaluators' annotations during experimentation (Shah and Pomerantz, 2010; Oh et al., 2012; Omari et al., 2016). However, we are interested in modeling the "helpfulness" for actual discussion forum users (in term of "Upvotes") and not annotators following guidelines to mark answer quality, which might present other forms of bias.

Modeling Novelty in IR, such as search result diversification (Carbonell and Goldstein, 1998; Soboroff and Harman, 2005; Ziegler et al., 2005; Clarke et al., 2008), also constitutes prior art. Carbonell and Goldstein (1998) proposed maximal marginal relevance (MMR) to diversify the set of documents returned for a search query. Similar approaches were also used later in Multi-Document Summarization (MDS) tasks (Nallapati et al., 2017). These approaches address the problem either as a ranking task (ordering search results) or as a subset selection problem (MDS), where all documents are simultaneously made available. In contrast, in our discussion thread scenario, we need to model the discussion posts' sequential nature to understand the context of a later post and, in turn, determine its helpfulness.

Neural Network Based Models have also recently outperformed existing classifiers in many text classification tasks. They have been widely adopted as they induce useful features on their own, given sufficient data. Although there are differences, the problem of answer selection is relevant: the goal is to rank the potential answers to a target question from multiple candidate answers in order of their similarity (Yu et al., 2014; Wang and Nyberg, 2015; Severyn and Moschitti, 2015). However in our case, all posts in a thread are similar to the original post to an extent. Helpful posts are thus more difficult to identify; computing similarity is not viable as a single source solution.

Inspired by all these previous works, we propose a neural architecture to predict the helpfulness of posts in open-ended discussion forums. To make it generic and easily adaptable to multiple domains, we study the problem from a linguistic viewpoint, where we consider only the textual contents of the discussion threads.

3 Methods

We propose a neural network architecture to model post helpfulness (*cf* Figure 2a). Our architecture is end-to-end trainable, adaptable to different domains. The model comprises two components to analyze a target post's thread *relevance* and *novelty* with respect to its past k posts.

3.1 Text Encoder

This component takes a post text p which consists of words $(w_1, w_2, \ldots, w_n)$ as input and encodes it to a tensor $(\mathbf{h}^p)$ in two steps. We first use a word embedding initialized with GloVe[7] to transform all the words from the post text into finite d-dimensional vectors, *i.e.*, $w_i \mapsto \mathbb{R}^d$. Our experimental results show that the coverage of GloVe varies between $68 - 76\%$ on our datasets. To estimate the embeddings for the out-of-vocabulary words and reflect the domain dependence, we keep the embedding vectors trainable. In the second step, the sequence of words are provided to a gated recurrent unit (GRU) layer (Chung et al., 2014) to obtain a sequence of hidden vectors $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)$, where $\mathbf{h}_i \in \mathbb{R}^g$, and g is the output dimension of the GRU encoded tensor. The latent vector is defined as follows:

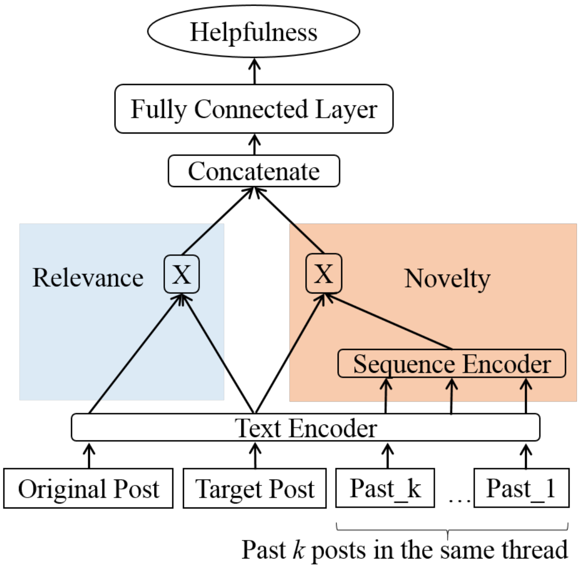
$$\mathbf{h}_i = \text{GRU}_{\text{text}}(\mathbf{h}_{i-1}, w_i).$$

The last vector in the sequence, $\mathbf{h}_n$, is considered as the encoded representation of a post text (*cf.* Figure 2c). For a post p, the GRU_{text} encoded representation is denoted as $\mathbf{h}^p$. We use a dropout layer after the GRU to prevent overfitting. In our model, note that there is only a single text encoder; all textual inputs — the target post, original post,

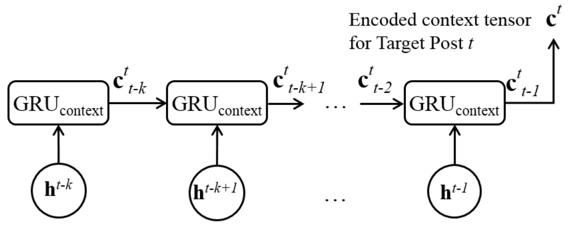
---

[6] `https://www.researchgate.net/`

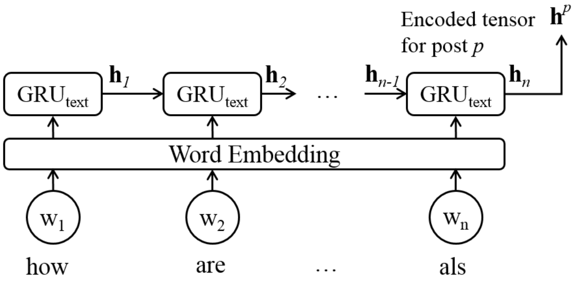
[7] `http://nlp.stanford.edu/data/glove.6B.zip`



(a) Overall network



(b) Sequence Encoder



(c) Text Encoder

Figure 2: Our neural architecture and its components. (a) Overall network architecture. Shaded component on the left captures relevance with respect to the original post; ones on the right measure the novelty compared to the past k posts. (b) Unrolled Sequence Encoder ($\text{GRU}_{\text{context}}$). (c) Unrolled Text Encoder (GRU_{text}).

and each of the past posts in the thread — are encoded using a single text encoder, since as all of them are essentially text posts of similar nature.

Alternative Architectures. We also tried stacking additional GRUs in our experiments, but we did not observe accuracy improvements. We also tried to replace GRU with LSTM (Long-Short Term Memory) (Hochreiter and Schmidhuber, 1997), resulting in similar performance at the cost of much longer training time due to the larger number of parameters.

3.2 Modeling Post's Relevance

The left component of Figure 2a captures the relevance of a target post with respect to the original post. It takes as input two GRU encoded tensors: one for the target post $\mathbf{h}^t$, the other for the original post $\mathbf{h}^o$. It computes their similarity defined as:

$$\mathbf{r}_t = \mathbf{h}^t \otimes \mathbf{h}^o,$$

where $\otimes$ denotes the element-wise multiplication. We also experimented with element-wise difference and cosine similarity, but found that multiplication works best. Our relevance modeling component is inspired from the architecture for answer sentence selection model (Yu et al., 2014).

3.3 Modeling Post's Novelty

In Figure 2a, the right component models the target post's novelty compared to the past k posts from the same thread. It takes the encoded tensors for the target post $\mathbf{h}^t$ as input, as well as the past k posts $(\mathbf{h}^{t-k}, \mathbf{h}^{t-k+1}, ..., \mathbf{h}^{t-1})$.

We first encode the context of the discussion by modeling the sequence of the past k posts. In order to achieve this, we use another GRU (labeled as Sequence Encoder in Figure 2a) to transform the sequence of k post tensors to a single context tensor $\mathbf{c}^t$ of equal dimension g. Each timestep i of this is defined as follows:

$$\mathbf{c}^t_i = \text{GRU}_{\text{context}}(\mathbf{c}^t_{i-1}, \mathbf{h}^{t-i}).$$

Similar to GRU_{text}, $\mathbf{c}^t_{t-1}$, the last vector in the sequence, is considered as the context representation $\mathbf{c}^t$ (as shown in Figure 2b).

To determine the novelty of the target post, we compute its similarity $\mathbf{n}_t$ with the discussion thread context represented by its context tensor:

$$\mathbf{n}_t = \mathbf{h}^t \otimes \mathbf{c}^t.$$

Importantly, instead of considering all the previous posts in the thread, we limit the context to the past k posts for two reasons:

1. Users may not recall the entire context of discussion while reading a post appearing much later in a long-running thread.

2. Users often arrive at a discussion thread through search engine queries. Since long threads are paginated, a user may arrive on a page in the

middle of the discussion thread, thus also missing the previous context.

We find empirical evidence for these assumptions later in our experiments (see Section 5). In tuning our model, we observed that increasing the context length beyond a threshold does not yield improvements.

3.4 Final Helpfulness Prediction

We combine the relevance tensor ($\mathbf{r}_t$) and novelty tensor ($\mathbf{n}_t$) and feed through a fully connected layer to make the final post helpfulness prediction:

$$\mathbf{x}_t = \mathbf{r}_t \oplus \mathbf{n}_t,$$

$$p(y|\mathbf{x}_t) = sigmoid(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{b}),$$

where $\oplus$ denotes concatenation; $\mathbf{x}_t$ is the concatenated tensor; y is the output label (0 or 1); $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector, respectively, learned for the fully connected layer. We use binary cross-entropy loss to train the model, optimizing with Adam (Kingma and Ba, 2014).

Alternative Architectures. We also investigated ensemble architectures. We fed the relevance and novelty tensors through two separate fully connected layers to obtain the binary predictions from both components concurrently, then merged the two predictions via a final fully connected layer for obtaining prediction. This approach fared worse compared to our concatenation-based model, possibly as our final concatenation model can exploit non-linear interactions between both components.

The actual post content is never presented to the fully connected layer so that it generalizes well. The final layer only gets to see the relevance, and novelty vectors, which we believe ameliorates the creation of overfitted (post-based or thread-based) features for the helpfulness prediction task.

4 Experiments

We first describe the datasets, evaluation metrics, and baseline models before our main results. We also conducted additional experiments to answer specific research questions about our model.

4.1 Datasets

We experiment with five real-world online discussion forums (Table 2) to validate model effectiveness. Typical of other research work, we also remove threads that have less than two posts.

1–2. **Reddit** is a popular platform for discussions on a wide-variety of topics on the web. We use a large number of discussion threads from a `reddit` data dump[8]. To diversify the datasets in terms of average thread length, we set different thresholds, and created two datasets: *Reddit_10+* ($\geq$ 10 posts) and *Reddit_3+* ($\geq$ 3 posts). Along with a chronologically ordered set of posts, `reddit` also has “Upvote” counts for every post.

Dataset	# Posts	# Threads	Avg # Posts / Thread	Avg # words / Post
1. Reddit_10+	200,006	9,744	20.52	29.45
2. Reddit_3+	200,016	28,763	6.95	30.58
3. Android Apps	11,643	2,077	5.60	56.53
4. Matrix	10,159	2,484	4.08	65.30
5. Travel	30,116	10,250	2.93	163.43

Table 2: Dataset statistics.

3–4. **Coursera** is a large MOOC platform, providing a discussion forum for the course participants. We select two courses with the largest number of posts: “Matrix-001” and “Android Apps 101-001” from a MOOC dataset (Chandrasekaran et al., 2015). Course participants can “vote” for a post if they find it helpful. We refer to these datasets as *Matrix* and *Android Apps*, hereafter.

5. **Travel Stack Exchange** is one of many QA websites in the Stack Exchange community. We use a data dump[9] of the website and refer to it as *Travel* dataset. In Travel Stack Exchange, a user can “Upvote” a post if she deems it helpful. Although not strictly a discussion forum, the threads in this forum appear to be less objective (by our vote distribution analysis, similar to Figure 1), compared to other CQA sites like `stackoverflow`.

4.2 Post Annotation and Evaluation Metrics

We use the user-provided feedback in form of “mark as helpful”, “like”, “upvote” actions as a proxy of the actual helpfulness of a post. Vote counts vary widely across posts and threads, (*i.e.*, 0 to 3,100 for the `reddit` dataset), making it infeasible to formulate the task as a regression problem. Following by prior published research (Cheng et al., 2014; Lo et al., 2017), we model it as a binary classification problem, and use the 80^{th} percentile expected value of helpful vote count across all the posts as the boundary between the two classes. We assume that a post is

---

[8] `https://files.pushshift.io/reddit/comments/`

[9] `https://archive.org/download/stackexchange/travel.stackexchange.com.7z`

helpful if it has received more helpful votes than the 80^{th} percentile, and *not helpful* otherwise.

Since our goal is to predict the helpful posts and the class distribution is inherently skewed from our definition, we evaluate the model performance in terms of prediction accuracy for only the positive, helpful class. We evaluate using standard precision, recall, and F_1 score across all datasets.

4.3 Baselines

Code for our model is publicly available[10] to aid the reproduction of our results. We experiment with the following state-of-the-art neural text classification methods:

1. BiLSTM (Sun et al., 2017): a stack of two layers of Bidirectional LSTM encoders on post text.

2. Stacked LSTM (Liu et al., 2016): a stack of two layers of LSTM encoders on the post text.

3. LSTM with Attention (Rocktäschel et al., 2016): an LSTM layer with hierarchical attention.

4. Answer Sentence Selection (Yu et al., 2014): a CNN model pioneered in a TREC QA[11] task.

5. Our Model (Relevance based): only the relevance component of our model.

6. Our Model (Novelty based): only the novelty component of our model.

We do not include traditional feature-based models as part of our reported baseline portfolio, as in our study, neural models have outperformed them as well, which is corroborated in recent studies (Kim, 2014). Additionally, such approaches are fragile, as we experiment with datasets from multiple domains with various discussion styles, and extracting hand crafted features for each is non-trivial and labour intensive. As a preliminary experiment, we tried with a traditional bag-of-words based model. However, we do not include it in the baseline portfolio given its poor performance on our datasets.

4.4 Training

We used the Keras[12] library with TensorFlow as the backend for model implementation. We split the dataset 80:10:10 for train, validation, and test, respectively, and perform 5-fold cross validation. We tuned the hyper-parameters via grid search on the validation set for all the models.

The rest of the parameters used follow standard values from the recent literature. We set word embedding dimension (d) to 100, vocabulary size to 100K, hidden dimension of GRU (g) to 128, batch size to 512, the dimension of the final fully connected layer to 128, and use 70% dropout. For the CNN-based Answer Sentence Selection baseline, we tuned the number and size of filters (128 and 3, respectively). The maximum length of post text was set according to average post length (in the training split) for each dataset.

4.5 Results

Table 3 shows the comparison of model performance over the five datasets. We observe that our full model consistently outperforms others in terms of F_1 across all datasets. Our novelty-based model gives the second best score in all datasets except for *Android Apps*. Comparing our novelty-based model against answer selection model, we observe that the helpfulness of a post depends on both its relevance to the original post and the novelty with respect to earlier posts in the same thread. The evaluation scores obtained by the state-of-the-art neural text classification models strongly support this observation. They consistently make less accurate prediction compared to the relevance- and/or novelty-based models. Among them, BiLSTM or LSTM with Attention model achieves the best performance, dependent on the dataset. We discuss the confounding factor affecting performance in Section 5.

We also observe that the prediction is more accurate when there is sufficient context to learn the dynamics of the discussion forums. In *Reddit_10+* and *Reddit_3+*, where both datasets average about 20 and 7 posts per thread respectively, we obtain an F_1 score of 0.40 to 0.51. In the other datasets, where the average thread length is much shorter ($\sim$ 3 to 5), we obtain relatively low F_1 scores of 0.34 to 0.38. Our model is more accurate in `reddit` datasets where threads are longer on average, indicative of more open-ended discussion centered on the original post.

4.6 Case Study

We now highlight a few corner cases successfully handled by our model.

Table 4 shows three target posts along with the original posts and their previous posts from dif-

---

[10] `https://github.com/WING-NUS/post-helpfulness`

[11] `http://trec.nist.gov/data/qa.html`

[12] `https://keras.io`

Model	1. **Reddit_10+**			2. **Reddit_3+**			3. **Android Apps**			4. **Matrix**			5. **Travel**		
	P	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**
BiLSTM (Sun et al., 2017)	0.23	0.23	0.23	0.23	0.22	0.22	**0.36**	0.32	0.34	0.29	0.35	0.32	0.28	**0.31**	0.29
Stacked LSTM (Liu et al., 2016)	0.24	0.21	0.22	0.23	0.20	0.21	0.34	0.29	0.31	0.32	0.29	0.31	0.23	0.26	0.25
LSTM w/ Attention (Rocktäschel et al., 2016)	0.24	0.21	0.23	0.24	0.21	0.22	0.34	0.27	0.30	0.30	0.36	0.33	0.25	0.26	0.25
Answer Selection (Yu et al., 2014)	0.28	0.27	0.27	0.31	0.32	0.32	0.28	0.21	0.24	0.33	0.34	0.33	0.30	**0.31**	0.31
Our Model (Relevance-based)	0.30	0.30	0.30	0.32	0.34	0.33	0.31	0.35	0.33	0.38	0.31	0.34	0.35	0.30	0.32
Our Model (Novelty-based)	**0.53**	0.38	0.44	**0.42**	0.27	0.33	0.33	0.24	0.28	**0.43**	0.27	0.33	**0.47**	0.27	**0.34**
Our Model (Full)	0.48	**0.53**	**0.51**	0.41	**0.39**	**0.40**	0.35	**0.40**	**0.38**	0.37	**0.37**	**0.37**	0.37	**0.31**	**0.34**

Table 3: (P)recision, (R)ecall and F$_1$ comparison of model performances across our five datasets representing three domains. Our model outperforms other state-of-the-art neural text classifiers consistently. Ablation study with Answer Selection, Relevance-based, and Novelty-based model shows that modelling both relevance, and novelty is important.

Original Post	**My fiancée and I are looking for a good Caribbean cruise in October and were wondering which islands are best to see and which Cruise line to take?..**	**I've had bouts of heart burn & this time its sticking around for a while. I ate something really spicy on Tuesday night & its Thursday & Im having heart burn on & off... Please help**	**In a few weeks' time, I will be visiting the US for 14 days. Coming from the EU, roaming is very expensive, so I am considering getting a temporary SIM card..**
Past Posts	Friends I am staying with are travelling with Royal Carribean on a cruise in October. They are starting from Miami..	You're probably fine. People get heartburn from time to time.. Eat bland food for a few days and that inflammation should subside..	There are many options you can have as far as mobile phone data prepaid plans are concerned. Since you need coverage along the route..
	The Princess Cruise line has a Caribbean cruise in the fall. It may start in November rather than October but could be suitable for your needs..	Heartburn can last a few days and its not always spicy food that triggers it. I assume youre concerned it might be a heart attack. If that was it you would know it by now.	You may want to check your existing phone plan. For example, quite a few providers in the UK offer free or cheap roaming with data included..
	There are plenty of options for the Caribbean in October regradless of it being in hurricane season..	Heartburn doesn't JUST occur from spicy food. If you're having it over multiple days, it could simply be other food. Fatty foods in particular cause it.	If your main goal is price, MetroPCS has no-contract 30 month plans which have unlimited calling US numbers, unlimited SMS, and unlimited data in the US..
Target Post	If you like to dress up and eat high-end food, the cruise line you want is not the one that caters to honeymooners on a tight budget or to families with small kids. If you like things to be..	Stay calm. Drink lots of water. Do you have an antacid you could take? Try to avoid spicy, acidic, caffeine, alcohol for a while..	`willmyphonework.net` is good for checking a phone's compatibility with the various networks. Suggestion before departure, print-out a list of the carriers your phone will work with hard copy is the way to go here..
Helpful?	Yes	No	No

Table 4: Illustration of different corner cases for helpfulness prediction. The target post needs to be both relevant to the original post, and novel compared to the previous posts in the thread in order to be helpful.

ferent datasets. In the first case, we observe that the target post introduces some relevant and novel information into the thread, and thus our model predicts it as helpful.

In the second example, we find that the target post is quite similar to some of the previous posts. Since it introduces less novelty in the discussion, our model predicts the target post as unhelpful, although relevant to the discussion topic. In the third example, the target post seems to be novel compared to the previous posts but it deviates from discussion topic in the original post. Hence, our model does not predict it as helpful.

These observations indicate that our model treats each of the two qualities of a target post, *i.e.*, relevance with the original post, and novelty compared to the previous discussion individually as necessary but not sufficient conditions. A target post needs both relevance and novelty so that our model predicts it as helpful.

5 Discussion

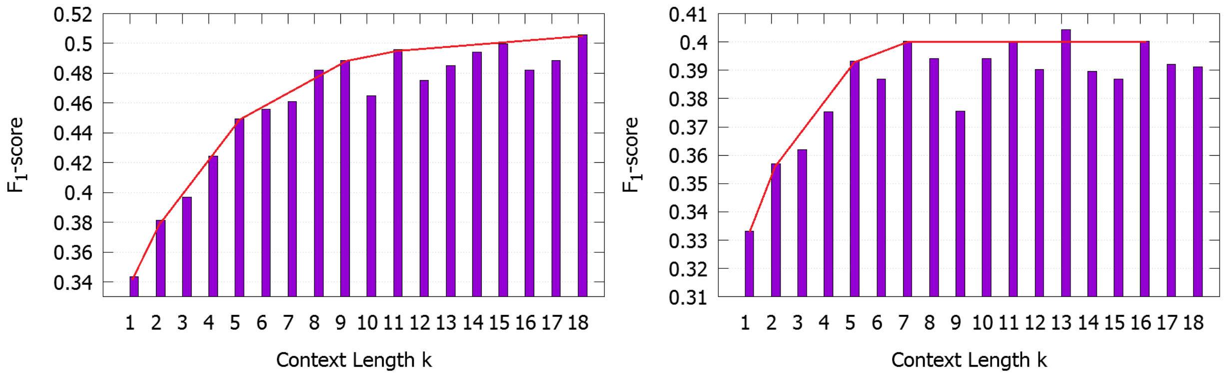
We now answer the following research questions (RQ) to further analyze prediction of helpful posts:

RQ1: How does the past context length influence model performance? The number of posts across threads varies widely, making it difficult to estimate the optimal value for past context length (k in Section 3.3). To understand the effect of k on model performance, we vary k ranging from 1 to 18 and report F$_1$ for the *Reddit_10+*, and *Reddit_3+* datasets in Figure 3. Interestingly, we observe that, the performance stops improving after a certain number of posts in both cases: k=11 and k=7 for *Reddit_10+*, and *Reddit_3+*, respectively.

Setting too low a k limits the number of past posts the model gets to see, underfitting the data. Large k gives modest performance gains but incurs significant increase in training cost. As discussed in Section 3.3, the entire context might be redundant to determine target posts' helpfulness in long threads.

We believe the context length analysis would be necessary to achieve optimal model performance while exploring other domains.

RQ2: Does the order of contextual posts matter? To investigate whether the order of the past posts matter in determining the helpfulness of a target post, instead of modeling the past posts by GRU$_{\text{context}}$ layer, we just use the average of the



(a) Reddit_10+

(b) Reddit_3+

Figure 3: Model performance while varying context length k for *Reddit_10+*, and *Reddit_3+* datasets. F$_1$ stabilizes after a certain context length in both cases. Trend line in red.

Context Modeling	Reddit_10+	Reddit_3+	Andriod Apps	Matrix	Travel
Average	0.40	0.35	0.36	0.36	0.33
$\text{GRU}_{\text{context}}$	**0.53**	**0.40**	**0.38**	**0.37**	**0.34**

Table 5: F$_1$ obtained by the model variation that uses the average tensor of the past post tensors as the context tensor, compared to our $\text{GRU}_{\text{context}}$ based model.

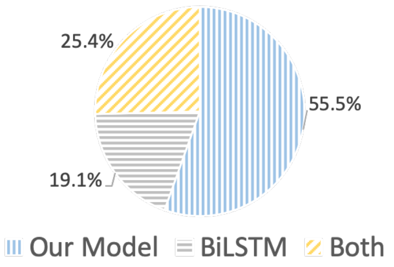


Figure 4: Correct prediction share of helpful posts for Reddit_all. Yellow: both models; blue: only our model; grey: only BiLSTM.

past post tensors to get the context tensor. Table 5 shows the F$_1$ achieved by this variation compared to our model.

We observe that the model performance significantly degrades when the order of the past posts are ignored and represented by an average. Crucially, we find that the datasets with longer threads suffer more compared to the ones with shorter threads. This observation indicates that the sequential nature of discussion is integral to model construction.

RQ3: What factors influence performance among the text classification models and our model? Table 3 shows that BiLSTM achieved better scores compared to the other neural text classification models. To better understand the modeling differences between the BiLSTM and our models, we focus on the cases where one model is correct but not the other (as illustrated for *Reddit_10+* in Figure 4). While both models can predict the correct class in 25.4% cases (in yellow), in the other cases (blue and grey), they differ.

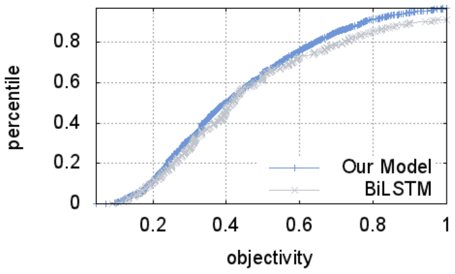


Figure 5: Thread objectivity score CDF. The blue curve shows threads where our model is correct and BiLSTM is not; vice versa for the grey.

We study the objectivity of the posts where such differences were observed. Without loss of generality, we define a metric called thread *objectivity* spread, in terms of the vote shares for the top-5 posts:

$$objectivity = \frac{max(vote(x)) - min(vote(x))}{\sum vote(x)},$$

where $x \in \{\text{top-5 posts}\}$ in the thread and $vote(x)$ gives the helpfulness score of post x. $objectivity$ is unit bound $[0, 1]$. While a high objectivity score indicates skewed helpfulness distribution in a thread, a low score indicates that there are multiple helpful answers in a thread; in other words, the thread is less objective in nature.

We analyze the cumulative distribution functions (CDFs) of objectivity spread scores for all threads belonging to the grey or blue wedge of Figure 4 (*cf.* Figure 5). We observe that the CDF for our model (blue) gives lower objectivity scores with 80^{th} percentile score of 0.64 for our model and 0.72 for BiLSTM, respectively. This indicates that our model performs better when the thread is more open-ended in nature.

6 Conclusion

We studied the problem of predicting helpfulness of posts in open-ended discussion forums. We found key differences in discussion forums compared to traditional CQA platforms: we observe that forum threads are often non-factoid and subjective in nature with many helpful answers. We hypothesize that post helpfulness crucially relies on two factors: (i) its relevance to the discussion thread and (ii) the novelty of the information introduced. We propose a generic and novel neural architecture using GRU encoders to embody this intuition. Our model outperforms state-of-the-art neural text classification baselines over a diverse set of forums representing three distinct domains. Through deeper analysis, we demonstrate that our model is able to encode the sequential nature of contextual posts, and capture the open-ended nature of discussion threads, thus achieving superior performance over other neural approaches.

We plan to apply our work towards building a notification system for incoming helpful posts. In the current work, we addressed the information need aspect present in the discussion forums in general. However, helpfulness might be conflated with other reasons such as humour, sentiment in certain domains. We would like to investigate those aspects in the future.

Acknowledgments

We acknowledge the support of NVIDIA Corporation for their donation of the Titan X GPU that facilitated this research. This research is supported by the Singapore National Research Foundation under its International Research Centre.

References

Muhammad Z Asghar, Aurangzeb Khan, Fazal M Kundi, Maria Qasim, Furqan Khan, Rahman Ullah, and Irfan U Nawaz. 2014. Medical Opinion Lexicon: an Incremental Model for Mining Health Reviews. *International Journal of Academic Research*, 6(1):295–302.

Alexander Beloborodov, Pavel Braslavski, and Marina Driker. 2014. Towards Automatic Evaluation of Health-Related CQA Data. In *Proc. of CLEF*, pages 7–18. Springer.

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the Right Facts in the Crowd: Factoid Question Answering over Social Media. In *Proc. of WWW*, pages 467–476.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proc. of SIGIR*, pages 335–336.

Muthu Kumar Chandrasekaran, Min-Yen Kan, Bernard C. Y. Tan, and Kiruthika Ragupathi. 2015. Learning Instructor Intervention from MOOC Forums: Early Results and Issues. In *Proc. of EDM*, pages 218–225.

Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can Cascades be Predicted? In *Proc. of WWW*, pages 925–936.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Proc. of NIPS 2014 Deep Learning and Representation Learning Workshop*.

Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and Diversity in Information Retrieval Evaluation. In *Proc. of SIGIR*, pages 659–666.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Liangjie Hong and Brian D Davison. 2009. A Classification-based Approach to Question Answering in Discussion Boards. In *Proc. of SIGIR*, pages 171–178.

Maximilian Jenders, Ralf Krestel, and Felix Naumann. 2016. Which Answer is Best?: Predicting Accepted Answers in MOOC Forums. In *Proc. of Q4APS*, pages 679–684.

Jiwoon Jeon, W Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A Framework to Predict the Quality of Answers with Non-Textual Features. In *Proc. of SIGIR*, pages 228–235.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. of EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Lei Li, Daqing He, Wei Jeng, Spencer Goodwin, and Chengzhi Zhang. 2015. Answer Quality Characteristics and Prediction on an Academic QA Site: A Case Study on ResearchGate. In *Proc. of WWW*, pages 1453–1458. ACM.

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling Interaction of Sentence Pair with Coupled-LSTMs. In *Proc. of EMNLP*, pages 1703–1712.

Caroline Lo, Justin Cheng, and Jure Leskovec. 2017. Understanding Online Collection Growth Over Time: A Case Study of Pinterest. In *Proc. of WWW 2017 Industry Track*, pages 545–554.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *Proc. of AAAI*, pages 3075–3081.

Sanghee Oh and Adam Worrall. 2013. Health Answer Quality Evaluation by Librarians, Nurses, and Users in Social Q&A. *Library & Information Science Research*, 35(4):288–298.

Sanghee Oh, Yong Jeong Yi, and Adam Worrall. 2012. Quality of Health Answers in Social Q&A. *Proc. of AIST*, 49(1):1–6.

Adi Omari, David Carmel, Oleg Rokhlenko, and Idan Szpektor. 2016. Novelty based Ranking of Human Answers for Community Questions. In *Proc. of SIGIR*, pages 215–224.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *Proc. of ICLR*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proc. of SIGIR*, pages 373–382.

Chirag Shah and Jefferey Pomerantz. 2010. Evaluating and Predicting Answer Quality in Community QA. In *Proc. of SIGIR*, pages 411–418.

Jyoti Prakash Singh, Seda Irani, Nripendra P Rana, Yogesh K Dwivedi, Sunil Saumya, and Pradeep Kumar Roy. 2017. Predicting the "Helpfulness" of Online Consumer Reviews. *Journal of Business Research*, 70:346–355.

Ian Soboroff and Donna Harman. 2005. Novelty Detection: The TREC Experience. In *Proc. of HLT-EMNLP*, pages 105–112.

Chengjie Sun, Yang Liu, Change Jia, Bingquan Liu, and Lei Lin. 2017. Recognizing Text Entailment via Bidirectional LSTM Model with Inner-Attention. In *Proc. of ICIC*, pages 448–457.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to Rank Answers on Large Online QA Collections. In *Proc. of ACL*, pages 719–727.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*, pages 3104–3112.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Proc. of ACL and IJCNLP*, pages 707–712.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking Community Answers by Modeling Question-Answer Relationships via Analogical Reasoning. In *Proc. of SIGIR*, pages 179–186.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *Proc. of NIPS 2014 Deep Learning and Representation Learning Workshop*.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proc. of WWW*, pages 22–32.

Text Classification with Few Examples using Controlled Generalization

Abhijit Mahabal
Google, Pinterest
amahabal@gmail.com

Jason Baldridge
Google
jridge@google.com

Burcu Karagol Ayan
Google
burcuka@google.com

Vincent Perot
Google
vperot@google.com

Dan Roth
UPenn
danroth@seas.upenn.edu

Abstract

Training data for text classification is often limited in practice, especially for applications with many output classes or involving many related classification problems. This means classifiers must generalize from limited evidence, but the manner and extent of generalization is task dependent. Current practice primarily relies on pre-trained word embeddings to map words unseen in training to similar seen ones. Unfortunately, this squishes many components of meaning into highly restricted capacity. Our alternative begins with sparse pre-trained representations derived from unlabeled parsed corpora; based on the available training data, we select features that offers the relevant generalizations. This produces task-specific semantic vectors; here, we show that a feed-forward network over these vectors is especially effective in low-data scenarios, compared to existing state-of-the-art methods. By further pairing this network with a convolutional neural network, we keep this edge in low data scenarios and remain competitive when using full training sets.

1 Introduction

Modern neural networks are highly effective for text classification, with convolutional neural networks (CNNs) as the de facto standard for classifiers that represent both hierarchical and ordering information implicitly in a deep network (Kim, 2014). Deep models pre-trained on language model objectives and fine-tuned to available training data have recently smashed benchmark scores on a wide range of text classification problems (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018).

Despite the strong performance of these approaches for large text classification datasets, challenges still arise with small datasets with few, possibly imbalanced, training examples per class. Labels can be obtained cheaply from crowd workers for some languages, but there are a nearly unlimited number of bespoke, challenging text classification problems that crop up in practical settings (Yu et al., 2018). Obtaining representative labeled examples for classification problems with many labels, like taxonomies, is especially challenging.

Text classification is a broad but useful term and covers classification based on topic, on sentiment, and even social status. As Systemic Functional Linguists such as Halliday (1985) point out, language carries many kinds of meanings. For example, words such as *ambrosial* and *delish* inform us not just of the domain of the text (*food*) and sentiment, but perhaps also of the age of the speaker. Text classification problems differ on the dimensions they distinguish along and thus in the words that help in identifying the class.

As Sachan et al. (2018) show, classifiers mostly focus on sub-lexicons; they memorize patterns instead of extending more general knowledge about language to a particular task. When there is low lexical overlap between training and test data, accuracy drops as much as 23.7%. When training data is limited, most meaning-carrying terms are never seen in training, and the sub-lexicons correspondingly poorer. Classifiers must generalize from available training data, possibly exploiting external knowledge, including representations derived from raw texts. For small training sizes, this requires moving beyond sub-lexicons.

Existing strategies for low data scenarios include treating labels as informative (Song and Roth, 2014; Chang et al., 2008) and using label-specific lexicons (Eisenstein, 2017), but neither is competitive when labeled data is plentiful. Instead, we seek classifiers that adapt to both low and high data scenarios.

People exploit parallelism among examples for generalization (Hofstadter, 2001; Hofstadter and

Proceedings of NAACL-HLT 2019, pages 3158–3167
Minneapolis, Minnesota, June 2 - June 7, 2019.

1.1	Kampuchea says **rice** crop in 1986 increased . . .	2.1	**Gamma ray** Bursters. What are they?
1.2	U.S. **sugar** policy may self-destruct . . .	2.2	Life on **Mars**
1.3	EC denies **maize** exports reserved for the U.S.. . .	2.3	Single launch **space stations**
1.4	U.S. **corn**, **sorghum** payments 50-50 cash/certs. . .	2.4	**Astronauts**—what does weighlessness feel like?
1.5	Canada **corn** decision unjustified. . .	2.5	**Satellite** around **Pluto** mission?

Table 1: **Left**: examples from the Reuters *Grains* class, showing semantic type cohesion (kinds of crops). **Right**: post headers from the *sci.space* newsgroup in 20 Newsgroups, showing topical cohesion (astronomical terms). **Bolded terms** are to draw the reader's attention to parallels among examples.

Sander, 2013). Consider Table 1, which displays five examples from a single class for two tasks. Bolded terms for each task are clearly related, and to a person, suggest abstractions that help relate other terms to the task. This helps with disambiguation: that the word *Pluto* is the planet and not Disney's character is inferred not just by within-example evidence (e.g. *mission*) but also by cross-example presence of *Mars* and *astronauts*.

Cross-example analysis also reveals the amount of generalization warranted. For a word associated with a label, word embeddings give us neighbors, which often are associated with that label. What they do not tell us is the extent this associated-with-same-label phenomenon holds; that depends on the granularity of the classes. Cross-example analysis is required to determine how neighbors at various distances are distributed among labels in the training data. This should allow us to include *barley* and *peaches* as evidence for a class like *Agriculture* but only *barley* for *Grains*.

Most existing systems ignore cross-example parallelism and thus miss out on a strong classification signal. We introduce a flexible method for controlled generalization that selects syntacto-semantic features from sparse representations constructed by Category Builder (Mahabal et al., 2018). Starting with sparse representations of words and their contexts, a tuning algorithm selects features with the relevant kinds and appropriate amounts of generalization, making use of parallelism among examples. This produces task-specific dense embeddings for new texts that can be easily incorporated into classifiers.

Our simplest model, CBC (Category Builder Classifier), is a feed-forward network that uses only CB embeddings to represent a document. For small amounts of training data, this simple model dramatically outperforms both CNNs and BERT (Devlin et al., 2018). When more data is available, both CNNs and BERT exploit their greater capacity and broad pre-training to beat CBC. We thus create CBCNN, a simple combination of CBC and the CNN that concatenates their pre-prediction layers and adds an additional layer. By training this model with a scheduled block dropout (Zhang et al., 2018) that gradually introduces the CBC sub-network, we obtain the benefits of CBC in low data scenarios while obtaining parity with CNNs when plentiful data is available. BERT still dominates when all data is available, suggesting that further combinations or ensembles are likely to improve matters further.

dataset	k	train/test/dev	size range
20NG	20	15076/1885/1885	513/810
reuters	8	6888/862/864	128/3128
spam	2	3344/1115/1115	436/2908
attack	2	10000/2000/2000	1126/8874

Table 2: Data sizes, and the disparity between the smallest and the largest class in training data. The k column indicates the number of classes in the task.

2 Evaluation Strategy

Our primary goal is to study classifier performance with limited data. To that end, we obtain learning curves on four standard text classification datasets (Table 2) based on evaluating predictions on the full test sets. At each sample size, we produce multiple samples and run several text classification methods multiple times, measuring the following:

- **Macro-F1 score**. Macro-F1 measures support for all classes better than accuracy, especially with imbalanced class distributions.
- **Recall for the rarest class**. Many measures like F1 and accuracy often mask performance on infrequent but high impact classes, such as detecting toxicity (Waseem and Hovy, 2016))
- **Degenerate solutions**. Complex classifiers with millions of parameters sometimes produce degenerate classifiers when provided very few training examples; as a result, they can skip some output classes entirely.

The datasets we chose for evaluation, while all multi-class, form a diverse set in terms of the number of classes and kinds of cohesion among examples in a single class. The former clearly affects training data needs, while the latter informs appropriate generalization.

- **20 Newsgroups** 20Newsgroups (20NG) contains documents from 20 different newsgroups with about 1000 messages from each. We randomly split the documents into an 80-10-10 train-dev-test split. The classes are evenly balanced.
- **Reuters R8.** The Reuters21578 dataset contains Reuters Newswire articles. Following several authors (Pinto and Rosso, 2007; Zhao et al., 2018, for example), we use only the eight most frequent labels. We begin with a given 80/10/10 split. Given that we focused on single-label classification, we removed items associated with two or more of the top eight labels (about 3% of examples). Classes are highly imbalanced. Of the 6888 training examples, 3128 are labeled *earn*, while only 228 examples are of class *interest* and only 128 are *ship*.
- **Wiki Comments Personal Attack.** The Wikipedia Detox project collected over 100k discussion comments from English Wikipedia and annotated them for presence of personal attack (Wulczyn et al., 2017). We randomly select 10k, 2k, and 2k items as train/dev/test. 11% are attacks.
- **Spam** The SMS Spam Collection v.1 has SMS labeled messages that were collected for mobile phone spam research (Hidalgo et al., 2012). Each of the 5574 messages is labeled as *spam* or *ham*.

3 Identifying Generalizing Features

In this section, we explicate the source of features, discuss the properties relevant to generalization by focusing on one feature in isolation, and present the overall feature selection method. The overview in Figure 1 displays the order of operations: identify *generalizing features* based on the training data (done once), and for each document to be classified, convert it to a vector, where each entry corresponds to a generalizing feature.

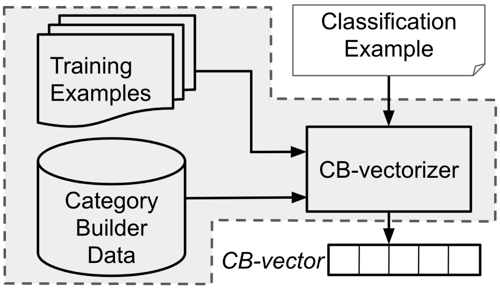


Figure 1: **Shaded Region:** We use Category Builder data (Mahabal et al., 2018) and identify generalizing features in training data, producing a *vectorizer*. This is done once. **Unshaded:** Given a document, the vectorizer produces a dense vector usable in deep networks.

Feature	Prototypical Supports
allergen as X	*pollen, dander, dust mites, soy, perfumes, milk, smoke, mildew*
liter of X	*water, petrol, milk, fluid, beer*
serve with X	*rice, sauce, salad, fries, milk*
flour mixture	*butter mixture, rubber spatula, dredged, creamed, medium speed, sifted, milk*
replacer	*colostrum, calves, whole milk, inulin, pasteurized, weaning*

Table 3: A few features (among hundreds) evoked by *milk*, with top n-grams in their support. **Above dashed line** (F_S) fit in tidy categories (here, *allergen*, *fluid*, and *food* are rough glosses). **Below dashed line** (F_C) are not describable by simple labels—the evoking terms have different parts of speech and instead display *situational coherence*, e.g. association with the process of mixing flour or with animal husbandry (a *replacer* is milk formula for calves).

3.1 Category Builder

Our source of generalizing features is Category Builder (CB) (Mahabal et al., 2018), which constructs a sparse vector space derived from parsed corpora (Erk, 2012). CB constructs features for *n-grams* (not just unigrams) that are the union of syntactic context features F_S and co-occurrence features F_C. Consider *milk*: an F_S feature is $gallon\xrightarrow{prep}of\xrightarrow{pobj}X$ and F_C features include *goat*, *cow*, *drink*, *spill*, etc. Table 3 provides other examples of features evoked by *milk*, along with other n-grams which evoke them. For present purposes, we can treat CB as a matrix with n-grams as rows and features in F_C and F_S as columns. The entries of CB are weights that give the association strength between an n-gram and a feature; these weights are an asymmetric variant of pointwise-mutual information (Mahabal et al., 2018).

3.2 Properties of Generalizing Features

Which features generalize well depends on the granularity of classes in a task. Useful features for generalization strike a balance between *breadth* and *specificity*. A feature that is evoked by many words provides generalization potential because the feature's overall support is likely to be distributed across both the training data and test data. However, this risks over-generalization, so a feature should also be sufficiently specific to be a precise indicator of a particular class.

A key aspect of choosing good features based on a limited training set is to resolve referential ambiguity (Quine, 1960; Wittgenstein, 1953) to the extent supported by the observed uses of the words. To illustrate, consider the *grains* class in the Reuters Newswire dataset. The word *wheat* can evoke the features at different levels of the taxonimical hierarchy: *triticum* (the wheat genus), *poaceae* (grass family), *spermatophyta* (seeded plants), *plantae* (plant kingdom), and *living thing*. The first among these has low breadth and is evoked only by *wheat*. The second is far more useful: specific and yet with a large support, including *maize* and *sorghum*. The final feature is too broad. In general, the most useful features for generalization are the intermediate features, also known as Basic Level Categories (Rosch et al., 1976).

Another important aspect of generalization comprises the *facets* of meaning. For example, the word *milk* has facets relating it to other liquids (e.g., *oil*, *kerosene*), foods (*cheese*, *pasta*), white things (*ivory*), animal products (*honey*, *eggs*), and allergens (*pollen*, *ragweed*). Along these axes, generalization can be more or less conservative; e.g., both *cheese* and *tears of a phoenix* are animal products, but the former is semantically closer to *milk*. Looking back at Table 3, the utility of individual features evoked by *milk* for tasks involving related topics varies; e.g., does the classification problem pertain to *food* or *animal husbandary*?

3.3 Focus on a single feature

A single *generalizing feature* is associated with many n-grams, each of which evokes it (with different strengths). Table 4 displays n-grams that evoke the feature *co-occurrence with **Saturn V***, as discovered by unsupervised analysis of a large corpus of web pages. The table further displays the interaction of this unsupervised feature with supervised data, specifically, with the label *sci.space* in 20NG, when using a size 320 training sample that contain only 18 *sci.space* documents. Counts for some evoking terms are shown within and outside this class, for both training and test data.

		Training		Testing	
n-gram	wt	C	$\overline{C}$	C	$\overline{C}$
apollo	8.93	1	1	5	1
launch pad	8.52	0	0	1	0
rocket	7.32	3	1	8	0
rockets	7.27	2	0	4	1
liftoff	6.92	1	0	1	0
space shuttle	6.27	0	0	4	0
space station	6.19	0	0	4	3
payload	4.23	0	0	5	0
shuttle	2.57	2	0	15	3
kennedy	2.30	1	0	1	4
capacity	1.95	0	1	0	4

Table 4: Some evoking n-grams associated with the CB feature *co-occurrence with **Saturn V*** and pivoting on the class *sci.space*. Counts for n-grams in training (sample size 320) and test data are shown, within *sci.space* (C) and outside ($\overline{C}$). Bolded n-grams are not seen in training but occur in test, providing generalization. The dashed line represents a threshold; higher scoring n-grams are more cohesive, and thresholding can make a feature cleaner by decreasing semantic drift.

Notation. We introduce some notation and explicate with Table 4. We have a labelled collection of training documents T. T_l is the training examples with label l. The *positive support set* $\Psi_l(f,t)$ is the set of n-grams in T_l evoking feature f with weight greater than t, here, $\{$*apollo, rocket, …, shuttle*$\}$ for t=2.3. The *positive support size* $\Lambda_l(f,2.3)$=$|\Psi_l(f,2.3)|$=5 and the *positive support weight* $\lambda_l(f,2.3)$ is the sum of counts of supports of f in l with weight greater than 2.3, here 1+3+2+1+2=9. Analogously the *negative support weight* $\lambda_{\bar{l}}(f,2.3)$ is the sum of counts from outside T_l; here, 1+1=2 since $\{$*apollo, rocket*$\}$ were seen outside *sci.space* once each.

What makes this feature (*words that have co-occurred with Saturn V*) well suited for *sci.space* is that many evoking words here are associated with the label *sci.space*. What confirms the benefit is the limited amount of negative support. Crucially, the bolded terms do not occur in the training data, but do occur in the test data. (We stress that we include these counts here only for this example; our methods do not access the test data for feature selection in our experiments.)

That said, we must limit potential noise from such features, so we seek **thresholded features** $\langle f, t \rangle$, as suggested by the dashed line in Table 4. Items below this line are prevented from evoking f. We choose the highest threshold such that dropped negative support exceeds dropped positive support. This is determined simply by going through all the supports of a feature, sorted by ascending weight, and checking the positive and negative support of all features with smaller versus greater weight given the class. The weight of the feature at this cusp is used as the threshold of the feature for this particular class. This $\langle f, t \rangle$ pair then forms one element of the CB-vector used as a feature for classification.

Given the labeled subsets of T and this feature thresholding algorithm, we produce a vectorizer that embeds documents. The values of a document's embedding *are not* directly associated with any class. Such association happens during training. Although *sci.space* accounts for just 6% of the documents, 75% of documents that contain an n-gram evoking the *Saturn V* feature are in that class. A classifier trained with such an embedding should learn to associate this feature with that class, and an unseen document containing the unseen-in-training term *space shuttle* stands a good chance to be classified as *sci.space*.

The feature displayed in Table 4 is useful for the 20NG problem because it contains a class related to space travel. This feature has no utility in spam classification or in sentiment classification, since, for those problems, seeing *rocket* in one class does not make it more likely that a document containing *space station* belongs to that same class. This example illustrates why a generalization strategy must incorporate both what we can learn from unsupervised data as well as (limited) labeled training data.

3.4 Overall feature selection

We now describe how we use the training data T to produce a set of features-and-threshold pairs; each chosen feature-with-threshold $\langle f, t \rangle$ will be one component in the CB-vectors provide to classifiers. Calculation of features for a single class is a three step process: (i) for each feature f, choose a threshold t (as discussed above) (ii) score the resultant $\langle f, t \rangle$ (iii) filter useless or redundant $\langle f, t \rangle$.

Given a label l and a feature f, we implicitly produce a table of supporting n-grams and their distribution within and outside l (e.g. as in Table 4). This involves computing the precision of a feature at a given threshold value, comparing it to the class probability and deciding whether to keep it.

Recall the positive support $\lambda_l(f, t)$ and negative support $\lambda_{\bar{f}}(f, t)$ defined previously. The **precision** of f at threshold t is $\mu_l(f, t) = \frac{\lambda_l(f,t)}{\lambda_l(f,t)+\lambda_{\bar{l}}(f,t)}$, (this is $\frac{9}{11}$ in the example of Table 4, with t=2.3). However, since we are often dealing with low counts, we smooth the precision toward the empirical class probability of l, $p(l) = \frac{|T_l|}{|T_l|+|T_{\bar{l}}|}$.

$$\tilde{\mu}_l(f, t) = \frac{\lambda_l(f, t) + p(l)\alpha}{\lambda_l(f, t) + \lambda_{\bar{l}}(f, t) + \alpha}$$

The score $S_l(f, t)$ is reduction in error rate of the smoothed precision relative to the base rate:

$$S_l(f, t) = \frac{\tilde{\mu}_l(f, t) - p(l)}{1 - p(l)}$$

We retain a thresholded feature if it is generalizing ($\Lambda_l(\langle f, t \rangle) > 1$), has better-than-chance precision (we use $S_l(\langle f, t \rangle) > 0.01$), and is not redundant (i.e., its positive support has one or more terms not present in positive supports of higher scoring features).

3.5 Creating the CB-vector

Each vector dimension corresponds to some $\langle f, t \rangle$. The evocation level of f is the sum of its evocation for the n-grams in the document d, $e_d(f) = \sum_{w \in d} CB(w, f)$. The vector entry is $\frac{e_d(f)}{t}$ when $e_d(f) >= t$, and is clipped to 0 otherwise.

4 Models

As benchmarks, we use a standard CNN with pretrained embeddings (Kim, 2014) and BERT (Devlin et al., 2018).[1] For CNN, we used 300 filters each of sizes 2, 3, 4, 5, and 6, fed to a hidden layer of 200 nodes after max pooling. Pretrained vectors provided by Google were used.[2] For BERT, we used the `run_classifier` script from GitHub and used the BERT-large-uncased model.

We use the pre-computed vocab-to-context association matrix provided as part of the open

---

[1] `https://github.com/google-research/bert`

[2] `https://code.google.com/archive/p/word2vec`

source Categorial Builder repository.[3] This contains 194,051 co-occurrence features (F_C) and 954,276 syntactic features (F_S).

CBC model. The *CB-vector* containing the derived features from the training dataset and Category Builder can be exploited in various ways with existing techniques. The simplest of these is to use a feed-forward network over the *CB-vector*. This model does not encode the tokens or any word order information—information which is highly informative in many classification tasks.

CBCNN model. Inspired by the combination of standard features and deep networks in Wide-and-Deep models (Cheng et al., 2016), we pair the CBC model with a standard CNN, concatenating their pre-prediction layers, and add an additional layer before the softmax prediction. In early experiments, this combined model performed worse than the CNN on larger data sizes, as the network above the CB-vector effectively stole useful signal from the CNN. To ensure that the more complex CNN side of the network had a chance to train, we employed a block dropout strategy (Zhang et al., 2018) with a schedule. During training, with some probability, all weights in the CB-vector are set to 0.5. The probability of hiding decreases from 1 to 0 using a parameterized hyperbolic tangent function $p_k = \frac{2}{e^{Cx}+1}$. Lower values of C lead to slower convergence to zero. The effect is that the CBC sub-network is introduced gradually, allowing the CNN to train while eventually taking advantage of the additional information.

The natural strategy of replacing with 0s (instead of 0.5 as above) was tried and also works, but less well, since the network has no way to distinguish between genuine absence of feature and hiding. In CB-vector, non-zero values are at least 1, and thus 0.5 does not suffer from this problem.

5 Experiments

Our primary goal is to improve generalization for low-data scenarios, but we also want our methods to remain competitive on full data.

5.1 Experimental setup

We compare different models across learning curves of increasing the training set sizes. We use training data sizes of $40, 80, \ldots, 5120$ as well as the entire available training data. For each training size, we produce three independent samples by uniformly sampling the training data and training each model three times at each size. The final value reported is the average of all nine runs. All models are implemented in Tensorflow. Batch sizes are between 5 and 64 depending on training size. Training stops after there is no macro-F1 improvement on development data for 1000 steps.

For evaluation, we focus primarily on macro-F1 and recall of the rarest class. The recall on the rarest class is especially important for imbalanced classification problems. For such problems, a model can obtain high accuracy by strongly preferring the majority class, but we seek models that effectively identify minority class labels. (This is especially important for active learning scenarios, where we expect the CB-vectors to help with in future.)

5.2 Results: low data scenarios

Figure 2 shows learning curves giving macro-F1 scores and rarest class recall for all four datasets. When very limited training data is available, the simple CBC model generally outperforms the CNN and BERT, except for the Spam dataset. The more powerful models eventually surpass CBC; however, the CBCNN model provides consistent strong performance at all dataset sizes by combining the generalization of CBC with the general efficacy of CNNs. Importantly, CBCNN provides massive error reductions with low data for 20NG and R8 (tasks with many labels).

Table 5's left half gives results for all models when using only 320 training examples. For 20NG, CNN's macro-F1 is just 43.9, whereas CBC and CBCNN achieve 61.7 and 62.4—the same as CNN performance with four times as much data. These models outperform CNN on R8 as well, reaching 83.7 vs CNN's 74.1, and also on the Wiki-attack dataset, achieving 80.6 vs CNNs 74.0. BERT fails to produce a solution for the two datasets with >2 labels, but does produce the best result for Spam—indicating an opportunity to more fully explore BERT's parameter settings for low data scenarios and to fruitfully combine CBC with BERT.

Rarest class recall is generally much better with less data when exploiting CB-features. For example, with 320 training examples for R8, CNNs reach 36.2 whereas CBCNN scores 76.2. Prediction quality with few training examples (especially getting good balance across all labels) also inter-

---

[3] `https://github.com/google/categorybuilder`

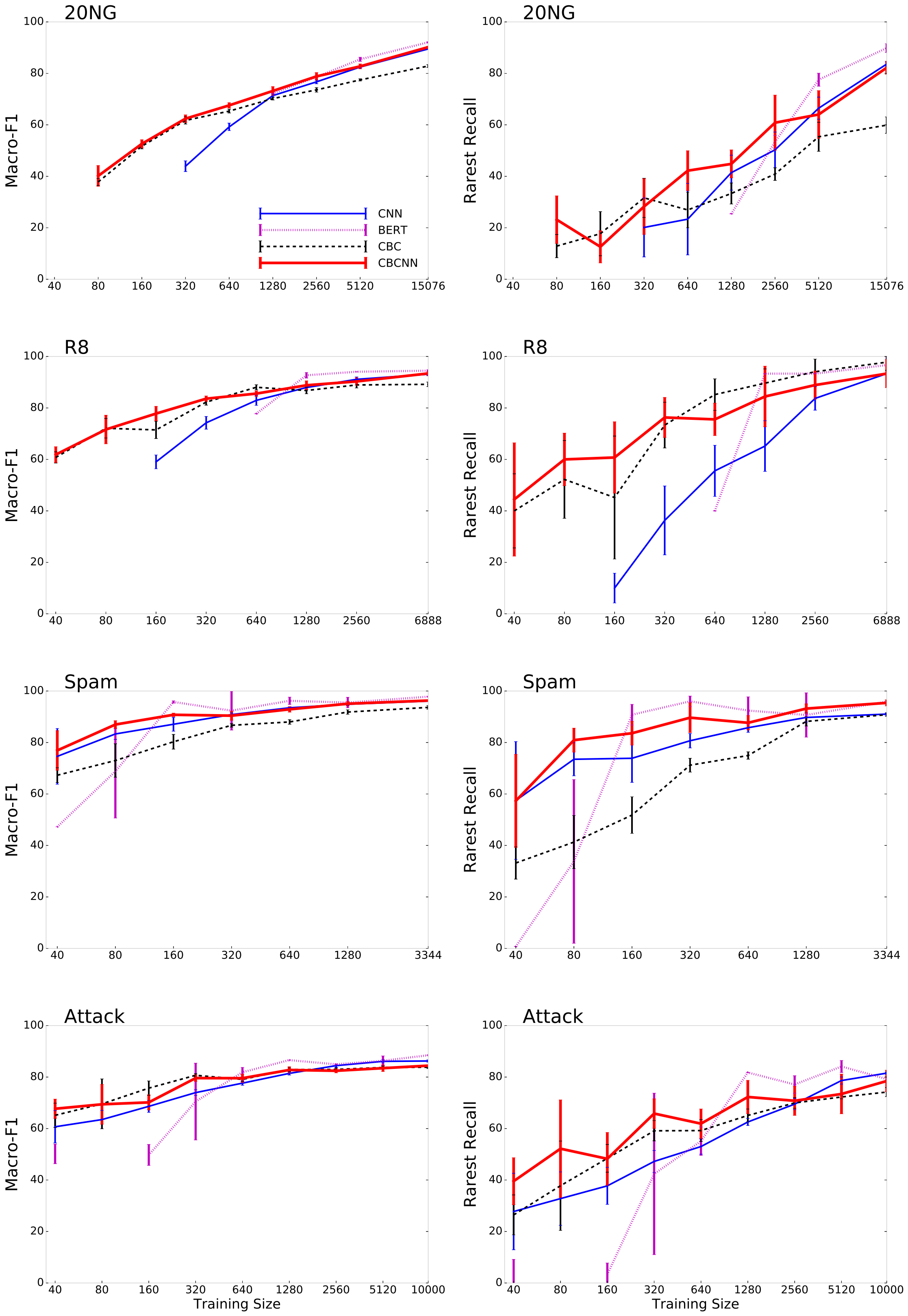


Figure 2: **Left:** F1 score by training size for 20NG, Reuters, SMS Spam, and Wiki-attack. Data is shown for non-degenerate models, and hence CNN and BERT start at higher sizes (see Table 6). **Right:** Recall for the rarest class for the same models.

		320 Training Examples				Full Training Data			
Data	k	CBC	CNN	CBCNN	BERT	CBC	CNN	CBCNN	BERT
20NG	20	61.7	43.9	**62.4**	—	82.9	89.5	90.2	**92.0**
R8	8	82.3	74.2	**83.6**	—	89.2	93.1	93.4	**94.4**
Spam	2	80.3	87.1	90.8	**95.7**	93.7	96.1	96.3	**97.8**
Attack	2	**80.7**	74.0	79.6	70.5	83.8	86.2	84.5	**88.2**

Table 5: Macro-F1 scores on all data sets when using 320 training examples (left) and when using all available training data (right). k is the number of classes. The CBCNN model provides the strongest overall performance across all data sizes. (Note that BERT produces degenerate solutions for the >2 class problems with 320 examples.)

Model	k	CBC	CNN	CBCNN	BERT
20NG	20	80	320	80	1280
R8	8	40	160	40	640
Spam	2	40	40	40	40
Attack	2	40	40	40	40

Table 6: Minimum training size at which a non-degenerate model was produced in any of 9 runs. With more classes, more data is needed by CNN and BERT to produce acceptable models. k is number of classes.

acts with other strategies for dealing with limited resources, such as active learning. For example, Baldridge and Osborne (2008) obtained stronger data utilization ratios with better base models and uncertainty sampling for Reuters text classification: better models pick better examples for annotation and thus require fewer new labeled data points to achieve a given level of performance.

Importantly, the CBC and CBCNN models take far less data to produce non-degenerate models (defined as a model which produces all output classes as predictions). CNN and BERT have a large number of parameters, and using these powerful tools with small training sets produces unstable results. Table 6 gives the minimum training set sizes at which each model produces at least one non-degenerate model. While it might be possible to ameliorate the instability of CNN and BERT with a wider parameter search and other strategies, nothing special needs to be done for CBC. It is likely that an approach which adaptively selects CBC or CBCNN and BERT would obtain the strongest result across all training set sizes.

For each dataset, among the 100 best features chosen (for training size 640), the breakdown of domain features (F_C) versus type features (F_S) is revealing. As expected, domain features are more important in a topical task such as 20NG (71% are F_C features), while the opposite is true for Spam (19%) and a toxicity dataset like Wiki Attack (23%). Reuters shows a fairly even balance between the two types of features (41%): it is useful for R8 to be topically coherent and also to hone in on fairly narrow groups of words that collectively cover a Basic Level Category.

5.3 Results: full data scenarios

Table 5 provides macro-F1 scores for all models when given all available training data. The CBC model performs well, but its (intentional) ignorance of the actual tokens in a document takes a toll when more labeled documents are available. The CNN benchmark, which exploits both word order and the tokens themselves, is a strong performer. The CBCNN model effectively keeps pace with the CNN—improving on 20NG and R8, though slipping on Wiki-Attack. BERT simply crushes all other models when there is sufficient training data, showing the impact of structured pre-training and consistent with performance across a wide range of tasks in Devlin et al. (2018).

6 Conclusion

We demonstrate an effective method for exploiting syntactically derived features from large external corpora and selecting the most useful of those given a small labeled text classification corpus. We show how to do this with the map provided by Category Builder n-grams to features, but other sources of well generalizing features have been exploited for text classification. These include topic models (Blei et al., 2003), ontologies such as WordNet (Bloehdorn and Hotho, 2004) and Wikipedia Category structure (Gabrilovich and Markovitch, 2009). It may be possible to use these other sources exactly as we use CB. Some of these sources have been manually curated, which makes them high quality but limits the size and facets. We have not yet explored their use because CB features seem to cover many of these sources' strengths—for example, F_C features are like topics, and F_S features like nodes in ontologies. Nonetheless, a combination may add value.

Our focus is on data scarce scenarios. However, it would be ideal to derive utility at both the small and large labeled data sizes. This will likely require models that can generalize with contextual features while also exploiting implicit hierarchical organization and order of the texts, e.g. as done by CNNs and BERT. The CBCNN model is one effective way to do this and we expect there could be similar benefits from combining CBC with BERT. Furthermore, approaches like AutoML (Zoph and Le, 2017) would likely be effective for exploring the design space of network architectures for representing and exploiting the information inherent in both signals.

Finally, although we focus on multi-class problems here—each example belongs to a single class—the general approach of selecting features should work for multi-label problems. Our confidence in this (unevaluated) claim stems from the observation that we select features one class at a time, treating that class and its complement as a binary classification problem.

Acknowledgements

We would like to thank our anonymous reviewers and the Google AI Language team, especially Rahul Gupta, Tania Bedrax-Weiss and Emily Pitler, for the insightful comments that contributed to this paper.

References

Jason Baldridge and Miles Osborne. 2008. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*, 14(2):191222.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *JMLR*, 3:993–1022.

Stephan Bloehdorn and Andreas Hotho. 2004. Boosting for text classification with semantic features. In *International Workshop on Knowledge Discovery on the Web*, pages 149–166. Springer.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, pages 7–10, New York, NY, USA. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Eisenstein. 2017. Unsupervised learning for lexicon-based classification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Evgeniy Gabrilovich and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498.

M. A. K. Halliday. 1985. *An introduction to functional grammar*. Arnold.

Jose Maria Gomez Hidalgo, Tiago A Almeida, and Akebo Yamakami. 2012. On the validity of a new SMS spam collection. In *Proceedings of the 11st ICMLA*, volume 2, pages 240–245. IEEE.

Douglas Hofstadter and Emmanuel Sander. 2013. *Surfaces and essences: Analogy as the fuel and fire of thinking*. Basic Books.

Douglas R Hofstadter. 2001. Analogy as the core of cognition. *The Analogical Mind: Perspectives from Cognitive Science*, pages 499–538.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Abhijit Mahabal, Dan Roth, and Sid Mittal. 2018. Robust handling of polysemy via sparse representations. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 265–275.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.

David Pinto and Paolo Rosso. 2007. On the relative hardness of clustering corpora. In *International Conference on Text, Speech and Dialogue*, pages 155–161. Springer.

Willard Van Orman Quine. 1960. *Word and object*. MIT press.

Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive psychology*, 8(3):382–439.

Devendra Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. 2018. Investigating the working of text classifiers. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2120–2131, Santa Fe, New Mexico, USA.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California.

Ludwig Wittgenstein. 1953. *Philosophical investigations*. John Wiley & Sons.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana.

Yuan Zhang, Jason Riesa, Daniel Gillick, Anton Bakalov, Jason Baldridge, and David Weiss. 2018. A fast, compact, accurate model for language identification of codemixed text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 328–337, Brussels, Belgium.

Taotao Zhao, Xiangfeng Luo, Wei Qin, Subin Huang, and Shaorong Xie. 2018. Topic detection model in a single-domain corpus inspired by the human memory cognitive process. *Concurrency and Computation: Practice and Experience*, 30(19):e4642.

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Appendix

Changes to the Category Builder Matrix

Category Builder uses two matrices: one mapping items to features ($M^{\mathcal{V}\rightarrow\mathcal{F}}$), the other mapping features to items ($M^{\mathcal{F}\rightarrow\mathcal{V}}$). Two are needed since the relationship is asymmetric: the feature *X is a star sign* is more strongly associated with the term *cancer* than vice versa, and the two matrices are thus not exact transposes of each other, although they almost are. For this current work, we just use one matrix, $M^{\mathcal{V}\rightarrow\mathcal{F}}$. For syntactic features F_S, we directly use the Category Builder rows. For F_C features, however, Category Builder replaced the corresponding submatrix in $M^{\mathcal{V}\rightarrow\mathcal{F}}$ with an identity matrix as described in (Mahabal et al., 2018). We obtain that part of the matrix by copying the corresponding rows from $(M^{\mathcal{F}\rightarrow\mathcal{V}})^T$.

This new matrix will be made available as part of the Category Builder project.

Reinforcement Learning Based Text Style Transfer without Parallel Training Corpus

Hongyu Gong[*] **Suma Bhat**[*] **Lingfei Wu**[†] **Jinjun Xiong**[†] **Wen-mei Hwu**[*]
[*]University of Illinois at Urbana-Champaign, USA
[†]T. J. Watson Research Center, IBM
[*]`{hgong6, spbhat2, w-hwu}@illinois.edu` [†]`{wuli, jinjun}@us.ibm.com`

Abstract

Text style transfer rephrases a text from a source style (e.g., informal) to a target style (e.g., formal) while keeping its original meaning. Despite the success existing works have achieved using a parallel corpus for the two styles, transferring text style has proven significantly more challenging when there is no parallel training corpus. In this paper, we address this challenge by using a reinforcement-learning-based generator-evaluator architecture. Our generator employs an attention-based encoder-decoder to transfer a sentence from the source style to the target style. Our evaluator is an adversarially trained style discriminator with semantic and syntactic constraints that score the generated sentence for style, meaning preservation, and fluency. Experimental results on two different style transfer tasks (sentiment transfer and formality transfer) show that our model outperforms state-of-the-art approaches. Furthermore, we perform a manual evaluation that demonstrates the effectiveness of the proposed method using subjective metrics of generated text quality.

1 Introduction

Text style transfer is the task of rewriting a piece of text to a particular style while retaining the meaning of the original text. It is a challenging task of natural language generation and is at the heart of many recent NLP applications, such as personalized responses in dialogue system (Zhou et al., 2017), formalized texts (Rao and Tetreault, 2018), cyberspace purification by rewriting offensive texts (Niu and Bansal, 2018; Santos et al., 2018), and poetry generation (Yang et al., 2018).

Recent works on supervised style transfer with a parallel corpus have demonstrated considerable success (Jhamtani et al., 2017b; Rao and Tetreault, 2018). However, a parallel corpus may not always be available for a transfer task. This has prompted studies on style transfer without parallel corpora. These hinge on the common idea of separating the content from the style of the text (Shen et al., 2017; Fu et al., 2018; Santos et al., 2018). This line of research first encodes the context via a style-independent representation, and then transfers sentences by combining the encoded content with style information. In addition, an appropriate training loss is chosen to change the style while preserving the content. However, these approaches are limited by their use of loss functions that must be differentiable with respect to the model parameters, since they rely on gradient descent to update the parameters. Furthermore, since focusing only on semantic and style metrics in style transfer, they ignore other important aspects of quality in text generation, such as language fluency.

In this paper, we propose a system trained using reinforcement-learning (RL) that performs text style transfer without accessing to a parallel corpus. Our model has a generator-evaluator structure with one generator and one evaluator with multiple modules. The generator takes a sentence in a source style as input and transfers it to the target style. It is an attention-based sequence-to-sequence model, which is widely used in generation tasks such as machine translation (Luong et al., 2015). More advanced model such as graph-to-sequence model can also exploited for this generation task (Xu et al., 2018b). The evaluator consists of a style module, a semantic module and a language model for evaluating the transferred sentences in terms of style, semantic content, and fluency, respectively. Feedback from each evaluator is sent to the generator so it can be updated to improve the transfer quality.

Our style module is a style discriminator built using a recurrent neural network, predicting the likelihood that the given input is in the target style.

Proceedings of NAACL-HLT 2019, pages 3168–3180
Minneapolis, Minnesota, June 2 - June 7, 2019.

We train the style module adversarially to be a target style classifier while regarding the transferred sentences as adversarial samples. An adversarial training renders style classification more robust and accurate. As for the semantic module, we used word movers' distance (WMD), a state-of-the-art unsupervised algorithm for comparing semantic similarity between two sentences (Kusner et al., 2015; Wu et al., 2018b), to evaluate the semantic similarity between input sentences in the source style and the transferred sentences in the target style. We also engaged a language model to evaluate the fluency of the transferred sentences.

Unlike prior studies that separated content from style to guarantee content preservation and transfer strength, we impose explicit semantic, style and fluency constraints on our transfer model. Moreover, employing RL allows us to use other evaluation metrics accounting for the quality of the transferred sentences, including non-differentiable ones.

We summarize our contributions below:

(1) We propose an RL framework for text style transfer. It is versatile to include a diverse set of evaluation metrics as the training objective in our model.

(2) Our model does not rely on the availability of a parallel training corpus, thus addressing the important challenge of lacking parallel data in many transfer tasks.

(3) The proposed model achieves state-of-the-art performance in terms of content preservation and transfer strength in text style transfer.

The rest of our paper is organized as follows: we discuss related works on style transfer in Section 2. The proposed text style transfer model and the reinforcement learning framework is introduced in Section 3. Our system is empirically evaluated on sentiment and formality transfer tasks in Section 4. We report and discuss the results in Section 5 and Section 6. The paper is concluded in Section 7.

2 Related Works

Text style transfer has been explored in the context of a variety of natural language applications, including sentiment modification (Zhang et al., 2018b), text simplification (Zhang and Lapata, 2017), and personalized dialogue (Zhou et al., 2017). Depending on whether the parallel corpus is used for training, two broad classes of style transfer methods have been proposed to transfer the text from the source style to the target style. We will introduce each line of research in the following subsections.

Style transfer with parallel corpus. Style transfer with the help of a style parallel corpus can be cast as a monolingual machine translation task. For this, a sequence-to-sequence (seq2seq) neural network has been successfully applied in a supervised setting. Jhamtani et al. transfer modern English to Shakespearean English by enriching a seq2seq model with a copy mechanism to replicate the source segments in target sentences (Jhamtani et al., 2017a).

Style transfer without parallel corpus. Scarce parallel data in many transfer tasks has prompted a recent interest in studying style transfer without a parallel corpus (e.g., (Zhang et al., 2018a)). Li et al. propose to delete words associated with the source style and replace them with similar phrases associated with the target style. Clearly, this approach is limited to transfers at the lexical level and may not handle structural transfer. Most existing unsupervised approaches share a core idea of disentangling content and style of texts. For a given source sentence, a style-independent content representation is firstly derived. Then, in combination with the target style, the content representation is used to generate the sentence following the target style.

Approaches to extract the content include variational auto-encoders (VAE) and cycle consistency. VAEs are commonly used to learn the hidden representation of inputs for dimensionality reduction, and have been found to be useful for representing the content of the source (Hu et al., 2017; Mueller et al., 2017; Shen et al., 2017; Fu et al., 2018). Cycle consistency is an idea borrowed from image style transfer for content preservation (Zhu et al., 2017). It proposes to reconstruct the input sentence from the content representation, by forcing the model to keep the information of the source sentence (Santos et al., 2018).

The transferred sentences are generated based on the content representation and the target style. One way to achieve this is with the use of a pre-trained style classifier. The classifier scores the transfer strength of the generated sentences and guides the model to learn the target text style (Santos et al., 2018; Prabhumoye et al., 2018). Another way is to learn the style embedding, which can be concatenated with the content embedding as the representation of the target sentence (Fu et al., 2018). The decoder then constructs the sentences from their hidden representations.

We note that previous works rely on gradient descent in their model training, and therefore their training losses (e.g., content and style loss) were limited to functions differentiable with respect to model parameters. Also, very few works consider other aspects of transfer quality beyond the content and the style of the generated sentences. This is in part due to their reliance on a differentiable training objective. We propose an RL-based style transfer system so that we can incorporate more general evaluation metrics in addition to preserving the semantic meaning of content and style transfer strength.

Reinforcement learning. RL has recently been applied to challenging NLP tasks (Yu et al., 2017). RL has advantages over supervised learning in that it supports non-differentiable training objectives and does not need annotated training samples. Benefits of using RL have been demonstrated in image captioning (Guo et al., 2018), sentence simplification (Zhang and Lapata, 2017), machine translation (Wu et al., 2018a) and essay scoring (Wang et al., 2018). A recent work on the task of sentiment transfer applied reinforcement learning to handle its BLEU score-based training loss (a non-differentiable function) (Xu et al., 2018a). Similar to the style transfer works discussed above, it also disentangled the semantics and the sentiment of sentences using a neutralization module and an emotionalization module respectively. Our work is different from these related works in that the semantic preservation and transfer strength are taken care of by the use of discriminators without explicitly separating content and style. An additional aspect that we focus here is the notion of fluency of the transferred sentences, which has not been explored before.

3 Model

Our style transfer system consists of the following modules: a generator, a style discriminator, a semantic module and a language model as shown in Fig. 1. We next describe the structure and function of each component. A closer view of our system is presented in Fig. 2.

Generator. The generator in our system takes a sentence in the source style as input and transfers it to the target style. For this, we use a recurrent encoder-decoder model combined with attention mechanism, which can handle variable-length input and output sequences (Sutskever et al., 2014; Cho et al., 2014). We could also leverage recently proposed more advanced encoder-decoder models (Xu et al., 2018b,c) to exploit rich syntactic information for this task, which we leave it as future work. Both the encoder and the decoder are recurrent neural layers with gated recurrent units (GRU). The encoder takes one word from the input at each time step, and outputs a hidden state vector $\bar{h}_s$ at time s. Similarly, the decoder outputs a hidden representation h_t at time t.

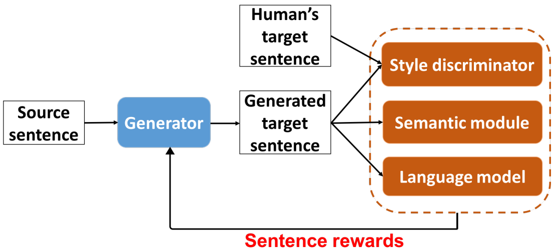


Figure 1: Model overview: the generator transfers the input source sentence to the generated target sentence. The generated sentences are collectively evaluated by the style discriminator, the semantic module and language module respectively. The style discriminator is adversarially trained with both human- and model-generated sentences. These three modules evaluate the generated sentences in terms of transfer strength, content preservation and fluency, and the rewards are sent to train the generator.

Suppose that the input sequence consists of T words $x = \{x_1, \ldots, x_T\}$, and the generated target sentence y is also a sequence of words $\{y_1, \ldots, y_{T'}\}$. We use $\text{vec}(\cdot)$ to denote the embedding of a word.

The gated recurrent unit dynamically updates its state h_t based on its previous state h_{t-1} and current input i_t. Its computation can be abstracted as $h_t = \text{GRU}(h_{t-1}, i_t)$. For the encoder, the input i_t is the embedding of the t-th input source word,

$$\bar{h}_t = \text{GRU}(\bar{h}_{t-1}, \text{vec}(x_t)). \tag{1}$$

For the decoder, the input to the recurrent unit is the embedding of the t-th generated target word.

$$h_t = \text{GRU}(h_{t-1}, \text{vec}(y_t)). \tag{2}$$

An attention mechanism is commonly adopted in text generation, such as machine translation (Bahdanau et al., 2015; Luong et al., 2015). We

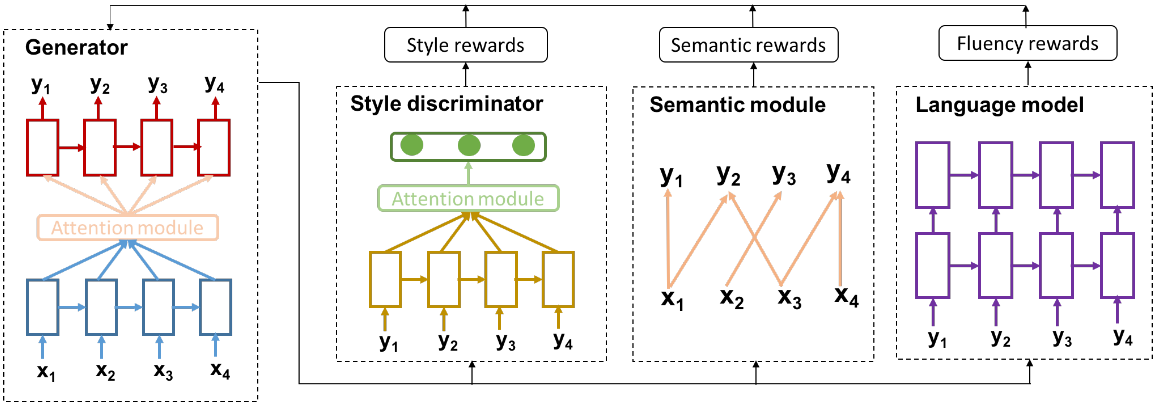


Figure 2: A detailed view of each component in the text style transfer system.

apply the attention mechanism to the decoding step so that the decoder learns to attend to source words and generates words. In this work, we use the attention mechanism similar to that used in (Luong et al., 2015). At the t-th decoding step, the attention $\alpha_t(s)$ is the weight of the s-th encoder state $\bar{h}(s)$.

The encoder hidden states are linearly weighted by the attention as the context vector at time t.

$$c_t = \sum \alpha_t(s)\bar{h}_s. \tag{3}$$

Combining the attention over the source sentence, the decoder produces a new hidden state $\tilde{h}_t$,

$$\tilde{h}_t = \tanh(W_c[c_t; h_t]). \tag{4}$$

The hidden vector $\tilde{h}_t$ is then used to predict the likelihood of the next word in the target sentence over the target vocabulary.

$$\mathbb{P}(y_t|y_{<t}, x) = \text{softmax}(W_s\tilde{h}_t). \tag{5}$$

where W_c and W_s are decoder parameters.

Style discriminator. The style discriminator evaluates how well the generated sentences are transferred to the target style. It is a classifier built on a bidirectional recurrent neural network with attention mechanism. The style discriminator is pre-trained to minimize the cross-entropy loss in the style classification task. This style classifier predicts the likelihood that an input sentence is in the target style, and the likelihood is taken as the style score of a sentence.

The pre-training does not guarantee that the neural network model will learn robust style patterns. So we resort to adversarial training as done in generative adversarial networks (GAN) (Yu et al., 2017; Wang and Lee, 2018). Accordingly, the style discriminator is later adversarially trained to distinguish the original (human-written) sentences from the model-generated ones so that the classifier learns to classify the text style well.

Semantic module. This evaluates how well the content from the input is preserved in the generated sentences. We use word mover's distance (WMD), which is the state-of-the-art approach (known for its robustness and efficiency) to measure the dissimilarity between the input and output sentences based on word embeddings (Kusner et al., 2015; Wu et al., 2018b). We take the negative of the WMD distance and divide it by the sequence length to yield the *semantic score* of a generated sentence. Previous works have also used cycle reconstruction loss to measure content preservation by reconstructing input sentences from generated sentences (Santos et al., 2018).

Language model. The style and the semantic modules do not guarantee the fluency of the transferred sentences. This fluency is achieved using a language model. The language model we use is a two-layer recurrent neural network pre-trained on the corpus in the target style so as to maximize the likelihood of the target sentences (Mikolov et al., 2010; Jozefowicz et al., 2016). The language model estimates the probability of input sentences. We take the logarithm of the probability and divide it by the sequence length as the *fluency score*.

3.1 Reinforcement Learning

The output sentences from the generator are sent to the semantic, style and language model modules for evaluation. These modules give feedback to the generator for the purpose of tuning it and to improve the quality of the generated sentences. We emphasize that despite the fact that our chosen evaluation metrics are not differentiable with respect to the generator parameters, they are still usable here. This is made possible by our use of the RL framework (the REINFORCE algorithm) to update the parameters of the generator (Williams, 1992).

In the RL framework, we define the state and the action for our style transfer task as follows. The state s_t at time t is the input source sequence and the first $t-1$ words that are already generated in the target sequence, i.e., $s_t = (X, Y_{1:t-1})$. The action a_t at time t is the t-th word to be generated in the output sequence, i.e., $a_t = y_t$. Suppose that the target vocabulary is V, and the maximum length of the decoder is T'. The generator G is parameterized with a parameter set θ, and we define the expected reward of the current generator as $J(G_\theta)$. The total expected reward is

$$J(G_\theta) = \sum_{t=1}^{T'} \mathbb{E}_{Y_{1:t-1} \sim G_\theta} [\sum_{y_t \in V} \mathbb{P}_\theta(y_t|s_t) Q(s_t, y_t)], \tag{6}$$

where $\mathbb{P}_\theta(y_t|s_t)$ is the likelihood of word y_t given the current state, and $Q(s_t, y_t)$ is the cumulative rewards that evaluate the quality of the sentences extended from $Y_{1:t}$. Suppose that $r(s_t, y_t)$ is the reward of word y_t at state s_t. The total reward, Q, is defined as the sum of the word rewards.

$$Q(s_t, y_t) = \sum_{\tau=t}^{T'} \gamma^{\tau-t} r(s_\tau, y_\tau), \tag{7}$$

where γ $(0 < \gamma < 1)$, is a discounting factor so that the future rewards have decreasing weights, since their estimates are less accurate.

If we only consider one episode, i.e., $Y_{1:t-1}$ has been given for every y_t, the reward $J(G_\theta)$ can be written as

$$J(G_\theta) = \sum_{t=1}^{T'} \sum_{y_t \in V} \mathbb{P}_\theta(y_t|s_t) Q(s_t, y_t). \tag{8}$$

Sequence sampling. By design, the three evaluation modules in Fig. 1 only evaluate complete sentences instead of single words or partial sentences. This means that we cannot obtain $r(s_t, y_t)$ directly from the evaluation modules at any time instance before the end of the sentence. One way around this problem is rolling out (Yu et al., 2017), where the generator 'rolls out' the given sub-sentence $Y_{1:t}$ at time step t to generate complete sentences by sampling the remaining part of the sentence $\{Y^n_{t+1:T'}\}$.

Previous works have adopted different sampling strategies, including Monte Carlo search, multinomial sampling and beam search. Starting from the given segment $Y_{1:t}$, Monte Carlo search explores the sub-sequence which leads to the best complete sentence (Yu et al., 2017). This leads to a good estimate of the sentence rewards but comes at significant computational cost. In many applications, the other two sampling strategies have been adopted for their efficiency. In multinomial sampling, each word y_τ $(t < \tau \leq T')$ is sampled from the vocabulary according to the likelihood $\mathbb{P}(y_\tau|s_\tau)$ predicted by the generator (ODonoghue et al., 2016; Chatterjee and Cancedda, 2010). The beam search process, on the other hand, keeps track of the k (a user-specified parameter) most likely words at each decoding step rather than just one word (Wu et al., 2018a). While this yields an accurate estimate of the reward for each action, multinomial sampling allows us to explore the diversity of generated texts with a potentially higher reward later on. This is the trade-off between exploitation and exploration in RL.

To balance the estimation accuracy and the generation diversity, we combine the ideas of beam search and multinomial sampling. Given a source sentence, we first generate a reference target sentence $Y^{\text{ref}}_{1:T}$ using beam search. To estimate the reward at each time step t, we draw samples of complete sentences $\{Y^l_{1:T'}\}$ by rolling out the subsequence $Y^{\text{ref}}_{1:t}$ using multinomial sampling. The evaluation scores of the sampled sentences are used as reward $r(y_t, s_t)$. More details about the sampling process are in Appendix.

Reward estimation. We estimate the reward as follows. We draw N samples of complete sentences starting from $Y_{1:t}$: $\{Y^{(n)}_{1:T'}\}^N_{n=1}$. The complete sentences are then fed into the three evaluation modules. Let f_{style} be the style score given by the style module, f_{semantic} be semantic score by the semantic module, and f_{lm} be the fluency score given by the language model.

We score the action y_t at state s_t by the average score of the complete sentences rolled out from $Y_{1:t}$. This action score is defined as the weighted sum of the scores given by the three modules.

$$f(s_t, y_t) = \frac{1}{N} \sum_{n=1}^{N} (\ \alpha \cdot f_{\text{style}}(Y^{(n)}_{1:T'}) + \\ \beta \cdot f_{\text{semantic}}(Y^{(n)}_{1:T'}, Y^{\text{real}}_{1:T'}) + \eta \cdot f_{\text{lm}}(Y^{(n)}_{1:T'})\ ), \tag{9}$$

where the hyperparameters α, β and η are positive.

In our experiments, we set $\alpha = 1.0$, $\beta = 0.5$ and $\eta = 0.5$ heuristically.

Given the scores from the evaluation modules, we define the reward $r(s_\tau, y_\tau)$ of word y_τ at state s_τ as

$$r(s_\tau, y_\tau) = \begin{cases} f(s_\tau, y_\tau) - f(s_{\tau-1}, y_{\tau-1}), & \tau > 1, \\ f(s_1, y_1), & \tau = 1. \end{cases} \tag{10}$$

We then obtain the discounted cumulative reward $Q(s_t, y_t)$ from the rewards $\{r(s_\tau, y_\tau)\}_{\tau > t}$ at each time step using Eq. 7.

The total reward of $J(G_\theta)$ can be derived from the cumulative rewards $\{Q(s_t, y_t)\}$ using Eq. 8. We define the generator loss L_θ as the negative of reward $J(G_\theta)$, $L_G(\theta) = -J(G_\theta)$.

According to Eq. 8, we can find the gradient $\nabla_\theta L_\theta$ of the generator loss as,

$$\nabla_\theta L_G(\theta) = -\sum_{t=1}^{T'} \nabla_\theta \mathbb{P}_\theta(y_t|s_t) Q(s_t, y_t). \tag{11}$$

3.2 Adversarial Training

The style discriminator is pre-trained on corpora in the source and target styles, and is used to evaluate the strength of style transfer. We note that this pre-training may not be sufficient for the style classifier to learn robust patterns and to provide accurate style evaluation. Indeed, in our experiments we found that even though the generator was trained to generate target sentences by maximizing the style rewards, the one-shot pre-training was insufficient to render the sentences in the target style.

Borrowing the idea of adversarial training proposed in GANs, we continuously trained the style discriminator using the generated target sentences. Toward this, we used a combination of a randomly sampled set of human-written target sentences $\{Y^{(k)}_{\text{human}}\}$ and model-generated sentences $\{Y^{(k)}_{\text{model}}\}$. Here the model-generated instances act as adversarial training samples, using which, the style discriminator was trained to distinguish the model outputs from human-written sentences. Let the discriminator D be parameterized by a parameter set ϕ. We define the prediction of the style discriminator, $D(Y)$, as the likelihood that the sentence Y is in the target style. The objective of this adversarial training amounts to minimizing the discriminator loss L_D:

$$L_D(\phi) = \frac{1}{K}\Big(-\sum_{k=1}^{K} \log(1 - D_\phi(Y^{(k)}_{\text{model}})) - \sum_{k=1}^{K} \log D_\phi(Y^{(k)}_{\text{human}}) \Big) . \tag{12}$$

4 Experiments

In this work, we considered two textual style transfer tasks, that of sentiment transfer (**ST**, involving negative and positive sentiments) and formality transfer (**FT**, involving informal and formal styles) using two curated datasets. We experimented with both transfer directions: positive-to-negative, negative-and-positive, informal-to-formal and formal-to-informal.

Dataset. For our experiments with style transfer we used a sentiment corpus and a formality corpus described below.

	Vocabulary	Type	Train	Dev	Test
Sentiment	9,640	Negative	176,878	25,278	50,278
		Positive	267,314	38,205	76,392
Formality	21,129	Informal	50,711	1,019	1,327
		Formal	50,711	1,019	1,019

Table 1: Data sizes of sentiment and formality transfer.

(1) Sentiment corpus. The sentiment corpus consists of restaurant reviews collected from the Yelp website (Shen et al., 2017). The reviews are classified as either negative or positive.

(2) Formality corpus. We use the Grammarly's Yahoo Answers Formality Corpus (GYAFC) (Rao and Tetreault, 2018), which is a collection of sentences posted in a question-answer forum (Yahoo Answers) and written in an informal style. In addition, these sentences have been manually rewritten in a formal style. We used the data from the section *family and relationships*. Note that even though the corpus is parallel, we did not use the parallel information.

Table 1 shows the train, dev and test data sizes as well as the vocabulary sizes of the corpora used in this work.

Model settings. The word embeddings used in this work were of dimension 50. They were first trained on the English WikiCorpus and then tuned on the training dataset. The width of the beam search (parameter k) was 8 during the RL and the inference stage.

Type	Source sentence	Transferred sentence
Negative-to-Positive	Crap fries , hard hamburger buns , burger tasted like crap !	Love you fries, burgers , always fun burger , authentic !
Positive-to-Negative	I was very impressed with this location .	I was very disappointed with this location .
Informal-to-Formal	It defenitely looks like he has feelings for u do u show how u feel u should ! !	It is like he is interested in you you should show how you feel .
Formal-to-Informal	I believe you 're a good man most likely she loves you quite a bit.	I think you 're a good man she kinda loves you .

Table 2: Example transferred sentences.

Pre-training. We pre-trained the generator, the style discriminator and the language model before the reinforcement learning stage. We discuss each of these steps below.

Generator pre-training. We pre-trained the generator to capture the target style from the respective target corpus. This pre-training occurred before setting up the reward from the evaluator to update its parameters in reinforcement learning. During pre-training, we used a set of target instances with a given instance serving as the input as well as the expected output. Using this set we trained the generator in a supervised manner with the cross-entropy loss as the training objective. Pre-training offered two immediate benefits for the generator: (1) the encoder and decoder learned to capture the semantics and the target style from the target corpus; (2) the generator had a good set of initial parameters that led to faster model training. This second aspect is a significant gain, considering that reinforcement learning is more time consuming than supervised learning.

Style discriminator pre-training. The style discriminator in our work was built using a bidirectional recurrent neural network. It was pre-trained using training corpora consisting of sentences in both the source and the target styles. We trained it to classify the style of the input sentences with the cross-entropy classification loss.

Language model pre-training. The language model was a two-layer recurrent neural network. Taking a target sentence $y = \{y_1, \ldots, y_{T'}\}$ as the input, the language model predicted the probability of the t-th word y_t given the previous subsequence $y_{1:t-1}$. The language model was pre-trained on the training corpus in target style to maximize the probability of y_t $(1 \leq t \leq T')$.

Baselines. We considered two state-of-the-art methods of unsupervised text style transfer that use non-parallel training corpus.

(1) Cross alignment model (CA). The CA model assumes that the text in the source and target style share the same latent content space (Shen et al., 2017). The style-independent content representation generated by its encoder is combined with available style information to transfer the sentences to the target style. We used their publicly available model for ST, and trained the model for FT separately with its default parameters.

(2) Multi-decoder seq2seq model (MDS). MDS consists of one encoder and multiple decoders (Fu et al., 2018). Similar to the cross alignment transfer, its encoder learns style-independent representations of the source, and the style specific decoder will rewrite sentences in the target style based on the content representation. We trained the model with its default parameters for both the tasks.

4.1 Evaluation

We used both automatic and human evaluation to validate our system in terms of content preservation, transfer strength and fluency.

4.1.1 Automatic evaluation

Aligning with prior work, we used the automatic metrics of content preservation, transfer and fluency that have been found to be well correlated with human judgments (Fu et al., 2018). For comparison, in Appendix, we also report our style and semantic metrics as provided by the evaluator.

Content preservation. A key requirement of the transfer process is that the original meaning be retained. Here we measure this by an embedding based sentence similarity metric s_{sem} proposed by (Fu et al., 2018). The embedding we used was based on the word2vec (CBOW) model (Mikolov et al., 2013). It was first trained on the English WikiCorpus and then tuned on the training dataset. Previous works used pre-trained GloVe embedding (Pennington et al., 2014), but we note that it does not have embeddings for Internet slang commonly seen in sentiment and formality datasets.

Transfer strength. The transfer strength s_{style} captures the degree to which the style transfer was

Sentiment	Negative-to-Positive				Positive-to-Negative			
Metric	Content	Style	Overall	Perplexity	Content	Style	Overall	Perplexity
CA	**0.894**	0.836	0.432	103.11	0.905	0.836	0.435	185.35
MDS	0.783	0.988	0.437	**98.89**	0.756	0.860	0.402	**156.98**
RLS	0.868	0.98	**0.460**	119.24	0.856	0.992	**0.459**	174.02
Formality	Informal-to-Formal				Formal-to-Informal			
Metric	Content	Style	Overall	Perplexity	Content	Style	Overall	Perplexity
CA	0.865	0.558	0.339	238.05	0.789	0.956	0.432	317.40
MDS	0.519	0.435	0.237	278.65	0.546	0.998	0.353	352.86
RLS	0.885	0.601	**0.358**	**208.33**	0.873	0.982	**0.462**	**267.78**

Table 3: Automatic evaluation of text style transfer systems on sentiment and formality transfer.

carried out and was quantified using a classifier. An LSTM-based classifier was trained for style classification on a training corpus (Fu et al., 2018). The classifier predicts the style of the generated sentences with a threshold of 0.5. The prediction accuracy is defined as the percentage of generated sentences that were classified to be in the target style. The accuracy was used to evaluate transfer strength, and the higher the accuracy is, the better the generated sentences fit in target style.

Overall score. We would like to point out that there is a trade-off between content preservation and transfer strength. This is because the outputs resulting from unchanged input sentences show the best content preservation while having poor transfer strength. Likewise, for given inputs, sentences sampled from the target corpora have the strongest transfer strength while barely preserving any content if at all. To combine the evaluation of semantics and style, we use the overall score s_{overall}, which is defined as a function of s_{sem} and s_{style}: $s_{\text{overall}} = \frac{s_{\text{sem}} * s_{\text{style}}}{s_{\text{sem}} + s_{\text{style}}}$ (Fu et al., 2018).

Fluency. This is usually evaluated with a language model in many NLP applications (Peris and Casacuberta, 2015; Tüske et al., 2018). We used a two-layer recurrent neural network with gated recurrent units as a language model, and trained it on the target style part of the corpus. The language model gives an estimation of perplexity (PPL) over each generated sentence. Given a word sequence of M words $\{w_1, \ldots, w_M\}$ and the sequence probability $p(w_1, \ldots, w_M)$ estimated by the language model, the perplexity is defined as:

$$\text{PPL} = p(w_1, \ldots, w_M)^{-\frac{1}{M}}. \qquad (13)$$

The lower the perplexity on a sentence, the more fluent the sentence is.

4.1.2 Human annotation

Noting the best overall score of our system in both directions of the tasks considered (to be discussed in the section that follows), we performed human annotations for content, style and fluency to validate the automatic scores. We chose a sample of 100 sentences generated by our system for each transfer task and collected three human judgments per sentence in each evaluation aspect. The annotation guidelines were:

Content preservation. Following the annotation scheme adopted by (Rao and Tetreault, 2018), we asked annotators to rate the semantic similarity between the original and transferred sentence on a scale from 1 to 6. Here "1" means completely dissimilar, "2" means dissimilar but on the same topic, "3" means dissimilar while sharing some content, "4" means roughly similar, "5" means almost similar, and "6" means completely similar.

Transfer strength. Annotators were given pairs of original and transferred sentences and were asked to decide which one was more likely to be in the target style. We define transfer strength to be the percentage of transferred sentences that were classified to be in the target style.

Fluency. Similar to the annotation of content, annotators scored sentences for fluency on a scale of 1(*not fluent*) to 6 (*perfectly fluent*).

5 Results

Some example sentences transferred by our system are shown in Table 2. More transferred sentences generated by our system and those by the baseline methods can be found in the Appendix. We first report the results of the automatic evaluation of our proposed system (denoted as "RLS") and the two baselines–the cross alignment model (CA) (Shen et al., 2017) and the multi-decoder seq2seq model (MDS) (Fu et al., 2018)–in Table 3.

Sentiment transfer. We notice that CA was the best in preserving content, MDS generated the most fluent target sentences and our model achieved the best trade-off between meaning and

Metric	Negative-to-positive	Positive-to-negative	Informal-to-formal	Formal-to-informal
Content (1-6)	5.19	5.20	4.96	5.33
Style accuracy	0.90	0.91	0.83	0.86
Fluency (1-6)	5.51	5.61	5.33	5.21

Table 4: Human judgments of transferred sentences

style with the highest overall score. Looking at the Overall score, it is notable that despite the differences in performance between the models studied here, each one performs similarly in both directions. This could be interpreted to mean that with respect to difficulty of transfer, style transfer is equivalent in both the directions for this task.

Formality transfer. For this task, we notice that our model outperforms the baselines in terms of content preservation, transfer strength and fluency with the best Overall score and perplexity. This suggests that our model is better at capturing formality characteristics compared to the baselines. We also note that the style strength of all models for informal-to-formal transfer is significantly lower than that for formal-to-informal transfer. This suggests that the informal-to-formal transfer is harder than the reverse. A plausible explanation is that informal sentences are more diverse and thus easier to generate than formal sentences. For example, informality can be achieved by multiple ways, such as by using an abbreviation (e.g., “u” used as “you”) and adding speech markers (e.g., “hey” and “ummm”), while formality is achieved in a more restricted manner.

Another challenge for informal-to-formal transfer is that informal data collected from online users usually contain non-negligible spelling errors such as “defenetely”, “htink” and “realy”. Words being the smallest semantic units in all the models considered here, these spelling errors could affect the transfer performance.

For each direction of transfer, we average the scores by annotators for each evaluation item, and report the results in Table 4. Our transferred sentences are shown to have good quality in content, style and fluency in subjective evaluations.

6 Discussion

To gain insights into the ways in which our approach performs the intended style transfer, we randomly sampled the generated sentences in the informal-to-formal transfer task. We found that the forms of rewriting can be broadly classified as: lexical substitution, word removal, word insertion and structural change. We show the following examples to these forms of re-writing, where the changed parts are highlighted.

(1) Lexical substitution. The informal sentence “I do **n’t** know what **u** mean” was transferred to “I do **not** know what **you** mean”;

(2) Word removal. The informal sentence “**And** I dont know what I should do” was rewritten as “I do not know what I should do”;

(3) Word insertion. In the example instance “depends on the woman” that was changed to “**It** depends on the woman”, we see that a subject was added to generate a complete formal sentence.

(4) Structural change. A small number of instances were also rewritten by making structural changes. For example, the informal sentence “**Just** tell them , **what are they gonna do** , slap you ??” was transferred to a formal version as “**You should** tell them , **they can not** slap you”. Other ways of style transfer by incorporating evaluation metrics of structural diversity are left for future work.

7 Conclusion

We proposed a reinforcement-learning-based text style transfer system that can incorporate any evaluation metric to enforce semantic, stylistic and fluency constraints on transferred sentences. We demonstrated its efficacy via automatic and human evaluations using curated datasets on two different style transfer tasks. We will explore and incorporate other metrics to improve other aspects of generated texts such as the structural diversity in the future work.

Acknowledgments

This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. We thank the NAACL anonymous reviewers for their constructive suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 663–670.

Tszhang Guo, Shiyu Chang, Mo Yu, and Kun Bai. 2018. Improving reinforcement learning based image captioning with natural language prior. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 751–756.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017a. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *EMNLP 2017*, 6:10.

Harsh Jhamtani, Varun Gangal, Eduard H. Hovy, and Eric Nyberg. 2017b. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *CoRR*, abs/1707.01161.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1865–1874.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:273–389.

Brendan ODonoghue, Rémi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. 2016. Pgq: Combining policy gradient and q. *arXiv preprint arXiv:1611.01626*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Álvaro Peris and Francisco Casacuberta. 2015. A bidirectional recurrent neural language model for machine translation. *Procesamiento del Lenguaje Natural*, 55:109–116.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876. Association for Computational Linguistics.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the yafc corpus: Corpus, benchmarks and metrics for formality style transfer. *arXiv preprint arXiv:1803.06535*.

Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. *arXiv preprint arXiv:1805.07685*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6830–6841.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zoltán Tüske, Ralf Schlüter, and Hermann Ney. 2018. Investigation on LSTM recurrent n-gram language models for speech recognition. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, pages 3358–3362.

Yau-Shian Wang and Hung-yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4187–4195.

Yucheng Wang, Zhongyu Wei, Yaqian Zhou, and Xuanjing Huang. 2018. Automatic essay scoring incorporating rating schema via reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 791–797.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018a. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621.

Lingfei Wu, Ian EH Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. 2018b. Word mover's embedding: From word2vec to document embedding. *arXiv preprint arXiv:1811.01713*.

Jingjing Xu, SUN Xu, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018a. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 979–988.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018b. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018c. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. *arXiv preprint arXiv:1808.07624*.

Cheng Yang, Maosong Sun, Xiaoyuan Yi, and Wenhao Li. 2018. Stylistic chinese poetry generation via unsupervised style disentanglement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3960–3969.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Ye Zhang, Nan Ding, and Radu Soricut. 2018a. Shaped: Shared-private encoder-decoder for text style adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1528–1538.

Yi Zhang, Jingjing Xu, Pengcheng Yang, and Xu Sun. 2018b. Learning sentiment memories for sentiment modification without parallel data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1108.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3400–3407.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251.

A Appendices

A.1 Sequence Sampling in Reinforcement Learning

The generator G transfers a source sentence X into a sentence in target style. In this work, we use beam search of width k to find a reference target sentence $Y^{\text{ref}}_{1:T'}$. In RL, we need to estimate the reward of each action y_t in the reference sentence

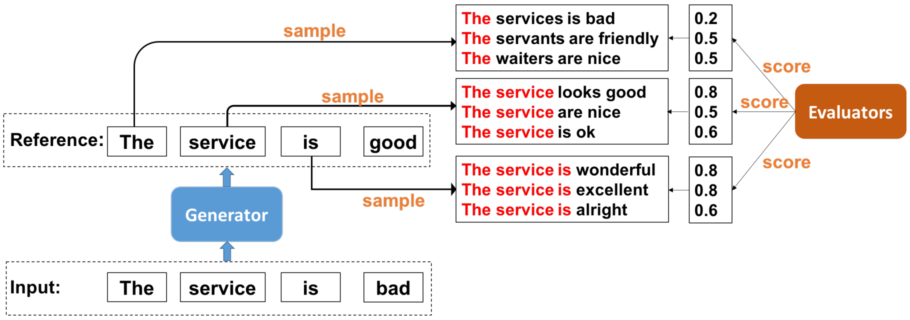


Figure 3: Sequence sampling: red words are sub-sequences in reference target sentence based on which the remaining sub-sequences are sampled. The sampled complete sentences are sent to the evaluator for scoring.

$Y^{\text{ref}}_{1:T'}$. Fig. 3 shows the sampling and scoring process.

Suppose that the reference target sentence $Y^{\text{ref}}_{1:T'}$ is “The service is good”. At the first time step, i.e., when $t = 1$, we start from the sub-sequence $Y^{\text{ref}}_{1:1}$, i.e., the sub-sentence ”The”. We use multinomial sampling to roll out “The” to complete sentences, which are “The service is bad”, “The servants are friendly” and “The waiters are nice” in Fig. 3. These sampled sentences are then sent to the evaluator for scoring in terms of style, content and fluency. Their scores are 0.2, 0.5 and 0.5 respectively, and we average them as the action score $f(y_1, s_1)$ of the first word y_1 at its state s_1. The score $f(y_1, s_1) = 0.4$ is sent back to the generator, which will be used to obtain the reward $r(y_1, s_1)$ as described in Eq. 10. Similarly when $t = 2$, we sample three complete sentences based on the sub-sentence “The service”: “The service looks good”, “The service are nice” and “The service is ok”.

A.2 Experiments

Automatic evaluation metrics. We reported the automatic evaluation results of all text style transfer systems in Table 3, where we used the evaluation metrics adopted by previous works (Fu et al., 2018; Santos et al., 2018). Here we report the style and semantic scores given by the evaluator in our system in Table 5. Recall that semantic score given by our evaluator was the negative of word movers’ distance between the generated sentence and the source sentence divided by the sentence length. The larger the semantic score was, the better the content was preserved in the generated sentence. As for the style evaluation, we used a bidirectional recurrent neural network as style classifier. It predicted the likelihood that an input sentence was in target style, which was taken as the *style score* of the generated sentences. Again, the larger the style score was, the better the generated sentence fitted in target style.

As shown in Table 5, the results given by the semantic and style modules of our evaluator are very similar to those given by Fu et al.. In sentiment transfer task, CA model does best in content preservation and MDS does best in transfer strength. As for FT, our model outperforms the two baselines in terms of semantic and style scores.

Examples and Analysis. We list some example transferred sentences given by our model and two baseline systems in Table 6. In the first example of negative-to-positive transfer, our model adheres to the topic of food service while baselines change to topic of food. Similarly in the first example of positive-to-negative transfer, our model preserves the topic of chicken while CA model talks about pizza and MDS model talks about customer service. Semantic similarity as explicit semantic constraints in our model is shown to be better at preserving the topic of source sentences.

There is still space to improve content preservation in all models. In the second example of informal-to-formal transfer, all transferred sentences miss the segment of “take a deep breathe” in the source sentence. In the second example of formal-to-informal transfer, the three transferred sentences miss part of source information. The source sentence is a rhetorical question, which truly means “people hardly understand the meaning behind their behavior”. This is a hard example, and all models do not capture its semantic meaning accurately.

FT task is more challenging compared with ST given that the sentence structure is more complicated with a larger vocabulary in the formality dataset.

Table 5: Semantic and style scores given by our evaluator on all systems.

	Sentiment				Formality			
	Negative-to-Positive		Positive-to-Negative		Informal-to-Formal		Formal-to-Informal	
Metric	Semantic	Style	Semantic	Style	Semantic	Style	Semantic	Style
CA	**-1.293**	0.806	**-1.346**	0.818	-1.212	0.646	-1.281	0.851
MDS	-1.412	**0.855**	-1.662	0.822	-1.508	0.568	-1.445	**0.878**
RLS	-1.315	0.846	-1.458	**0.847**	**-0.935**	**0.782**	**-0.903**	0.872

Table 6: Example transferred sentences of all systems.

Type	Transferred sentence
Negative-to-positive	Source: I ’ve noticed the food service sliding down hill quickly this year . CA: I have enjoyed the food here and this place is perfect . MDS: Food is the best staff . RLS: I ’ve noticed the food service was perfect this time .
	Source: The chicken tenders did n’t taste like chicken , wtf ? CA: The food tastes good , just like spicy ! MDS: And the food is the food in the food in well . RLS: . The chicken were like chicken, you can find what you want .
Positive-to-negative	Source: I recommend ordering the “ special chicken ” really good ! CA: I would give the pizza ... how they really really good ? MDS: They are the worst customer service . RLS: I would say chicken were very bad .
	Source: My experience was brief , but very good . CA: My experience was ok , but , very good . MDS: Worst , i would never go to going back . RLS: My experience was bad .
Informal-to-formal	Source: Well that is just the way it is i guess . CA: It is the best thing i think that is not . MDS: That is for the way . RLS: It is the way I think .
	Source: Like i said he already knows that you like him , so just take a deep breathe and ask him . CA: I think that she likes you , but perhaps you will get a relationship and and ask her . MDS: If you find him and i think that you have been in a relationship . RLS: I believe he knows that you like him, so go to ask him .
Formal-to-informal	Source: Well, if you are really attracted to this guy, then smile and speak nicely to him . CA: If you to tell her the way that is you and get married . MDS: The way of guys are not if you are not . RLS: Well , if you really like this guy , then smile to him .
	Source: Men are unintelligent! What person understands the meaning behind their behavior ? CA: Men are not of his meaning . MDS: Men are understands all men are not ? RLS: Men are stupid ! Why girl loves the mind ?

Adapting RNN Sequence Prediction Model to Multi-label Set Prediction

Kechen Qin **Cheng Li** **Virgil Pavlu** **Javed A. Aslam**
Khoury College of Computer Sciences
Northeastern University
`qin.ke@husky.neu.edu` `{chengli,vip,jaa}@ccs.neu.edu`

Abstract

We present an adaptation of RNN sequence models to the problem of multi-label classification for text, where the target is a *set of labels*, not a sequence. Previous such RNN models define probabilities for sequences but not for sets; attempts to obtain a set probability are after-thoughts of the network design, including pre-specifying the label order, or relating the sequence probability to the set probability in *ad hoc* ways.

Our formulation is derived from a principled notion of set probability, as the sum of probabilities of corresponding permutation sequences for the set. We provide a new training objective that maximizes this set probability, and a new prediction objective that finds the most probable set on a test document. These new objectives are theoretically appealing because they give the RNN model freedom to discover the best label order, which often is the natural one (but different among documents).

We develop efficient procedures to tackle the computation difficulties involved in training and prediction. Experiments on benchmark datasets demonstrate that we outperform state-of-the-art methods for this task.

1 Introduction

Multi-label text classification is an important machine learning task wherein one must predict a set of labels to associate with a given document; for example, a news article might be tagged with labels `sport`, `football`, `2018 world cup`, and `Russia`. Formally, we are given a set of label candidates $\mathcal{L} = \{1, 2, ..., L\}$, and we aim to build a classifier which maps a document x to a set of labels $\mathbf{y} \subset \mathcal{L}$. The label set $\mathbf{y}$ is typically written as a binary vector $\mathbf{y} \in \{0, 1\}^L$, with each bit y_ℓ indicating the presence or absence of a label.

Naively, one could predict each label independently without considering label dependencies. This approach is called Binary Relevance (Boutell et al., 2004; Tsoumakas and Katakis, 2007), and is widely used due to its simplicity, but it often does not deliver good performance. Intuitively, knowing some labels—such as `sport` and `football`—should make it easier to predict `2018 world cup` and then `Russia`. There are several methods that try to capture label dependencies by building a joint probability estimation over all labels $p(\mathbf{y} = (y_1, y_2, ..., y_L)|x)$ (Ghamrawi and McCallum, 2005; Read et al., 2009; Dembczynski et al., 2010; Li et al., 2016). The most popular approach, Probabilistic Classifier Chain (PCC) (Dembczynski et al., 2010) learns labels one-by-one in a predefined fixed order: for each label, it uses one classifier to estimate the probability of that label given all previous labels predictions, $p(y_l|y_1, ..., y_{l-1}, x)$. PCC's well known drawback is that errors in early probability estimations tend to affect subsequent predictions, and can become massive when the total number of label candidates L is large.

Recurrent neural network (RNN) is originally designed to output a sequential structure, such as a sentence (Cho et al., 2014). Recently, RNNs have also been applied to multi-label classification by mapping the label set to a sequence (Wang et al., 2016; Zhang et al., 2016; Jin and Nakayama, 2016; Wang et al., 2017b,a; Chen et al., 2018; Yang et al., 2018). In contrast to PCC where a binary decision is made for each label sequentially, RNN only predicts the positive labels explicitly and therefore its decision chain length is equal to the number of positive labels, not the number of all labels. This makes RNN suffer less from early estimation errors than PCC.

Both PCC and RNN rely heavily on label orders in training and prediction. In multi-label data, the labels are given as sets, not necessarily with natural orders. RNN defines a sequence probability,

Proceedings of NAACL-HLT 2019, pages 3181–3190
Minneapolis, Minnesota, June 2 - June 7, 2019.

while PCC defines set probability. Various ways of arranging sets as sequences have been explored: ordering alphabetically, by frequency, based on a label hierarchy, or according to some label ranking algorithm (Liu and Tsang, 2015). Previous experimental results show that which order to choose can have a significant impact on learning and prediction (Vinyals et al., 2016; Nam et al., 2017; Chen et al., 2018). In the above example, starting label predictions sequence with `Russia`, while correct, would make the other predictions very difficult.

Previous work has shown that it is possible to train an RNN on multi-label data without specifying the label order in advance. With special training objectives, RNN can explore different label orders and converge to some order automatically (Vinyals et al., 2016). In this paper we follow the same line of study: We consider how to adapt RNN sequence model to multi-label set prediction without specifying the label order. Specifically, we make the following contributions:

1. We analyze existing RNN models proposed for multi-label prediction, and show that existing training and prediction objectives are not well justified mathematically and have undesired consequences in practice.

2. We develop efficient approximate training and prediction methods. We propose new training and prediction objectives based on a principled notion of set probability. Our new formulation avoids the drawbacks of existing ones and gives the RNN model freedom to discover the best label order.

3. We crawl two new datasets for multi-label prediction task, and apply our method to them. We also test our method on two existing multi-label datasets. The experimental results show that our method outperforms state-of-the-art methods on all datasets. We release the datasets at `http://www.ccis.neu.edu/home/kechenqin`.

2 Mapping Sequences to Sets

In this section, we describe how existing approaches map sequences to sets, by writing down their objective functions using consistent notations. To review RNN designed for sequences, let $\mathbf{s} = (s_1, s_2, ..., s_T)$ be an input sequence of outcomes, in a particular order, where $s_t \in \{1, 2, ..., L\}$; the order is often critical to the datapoint. An RNN model defines a probability distribution over all possible output sequences given the input in the form $p(\mathbf{s} = (s_1, s_2, ..., s_T)|x) = \prod_{t=1}^{T} p(s_t|x, s_1, s_2, ..., s_{t-1})$. To train the RNN model, one maximizes the likelihood of the ground truth sequence.

At prediction time, one seeks to find the sequence with the highest probability $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$, and this is usually implemented approximately with a beam search procedure (Lowerre, 1976) (we modified into Algorithm 1). The sequence history is encoded with an internal memory vector h_t which is updated over time. RNN is also often equipped with the attention mechanism (Bahdanau et al., 2014), which in each timestep t puts different weights on different words (features) and thus effectively attends on a list of important words. The context vector c_t is computed as the weighted average over the dense representation of important words to capture information from the document. The context c_t, the RNN memory h_t at timestep t, and the encoding of previous label s_{t-1} are all concatenated and used to model the label probability distribution at time t as $p(s_t|x, s_1, s_2, ..., s_{t-1}) \sim \textit{softmax}(\phi(c_t, h_t, s_{t-1}))$, where ϕ is a non-linear function, and *softmax* is the normalized exponential function.

To apply RNN to multi-label problems, one approach is to map the given set of labels $\mathbf{y}$ to a sequence $\mathbf{s} = (s_1, s_2, ..., s_T)$, on training documents. This is usually obtained by writing the label set in a globally fixed order (e.g. by label frequency), as in PCC. Once the mapping is done, RNN is trained with the standard maximum likelihood objective (Nam et al., 2017):

$$\textit{maximize} \sum_{n=1}^{N} \log p(\mathbf{s}^{(n)}|x^{(n)}) \qquad (1)$$

where $x^{(n)}$ is the n-th document and N is the total number of documents in the corpus.

Vinyals et al. (2016) proposes to dynamically choose during training the sequence order deemed as most probable by the current RNN model:

$$\textit{maximize} \sum_{n=1}^{N} \max_{\mathbf{s} \in \pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)}) \qquad (2)$$

where the $\pi(\mathbf{y}^{(n)})$ stands for all permutations of the label set $\mathbf{y}^{(n)}$. This eliminates the need to manually specify the label order. However, as noticed

Methods	Training objectives	Prediction objectives
seq2seq-RNN	*maximize* $\sum_{n=1}^{N} \log p(\mathbf{s}^{(n)}\|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*)$, $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}\|x)$
Vinyals-RNN-max	*maximize* $\sum_{n=1}^{N} \max_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}\|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*)$, $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}\|x)$
Vinyals-RNN-uniform	*maximize* $\sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}\|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*)$, $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}\|x)$
Vinyals-RNN-sample	*maximize* $\sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}\|x^{(n)}) \log p(\mathbf{s}\|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*)$, $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}\|x)$
set-RNN (ours)	*maximize* $\sum_{n=1}^{N} \log \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}\|x^{(n)})$	$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}\|x)$

Table 1: Comparison between previous and our *set-RNN* training and prediction objectives.

by the authors, this objective cannot be used in the early training stages: the early order choice (often random) is reinforced by this objective and can be stuck upon permanently. To address this issue, Vinyals et al. (2016) also proposes two smoother alternative objectives to initialize the model training:

The authors suggest that one first consider many random orders for each label set in order to explore the space:

$$\textit{maximize} \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)}) \tag{3}$$

After that, one can sample sequences following the model predictive distribution instead of uniform distribution:

$$\textit{maximize} \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)}) \log p(\mathbf{s}|x^{(n)}) \tag{4}$$

In training, one needs to schedule the transition among these objectives, a rather tricky endeavor. At prediction time, one needs to find the most probable set. This is done by (approximately) finding the most probable sequence $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$ and treating it as a set $\hat{\mathbf{y}} = set(\mathbf{s}^*)$. With a large number of sequences, it is quite possible that the argmax has actually a low probability, which can lead to neglecting important information when we ignore sequences other than the top one.

3 Adapting RNN Sequence Prediction Model to Multi-label Set Prediction

We propose a new way of adapting RNN to multi-label set prediction, which we call *set-RNN*. We appreciate the RNN model structure (Rumelhart et al., 1988) (defines a probability distribution over all possible sequences directly) and introduce training and prediction objectives tailored for sets that take advantage of it, while making a clear distinction between the sequence probability $p(\mathbf{s}|x)$ and the set probability $p(\mathbf{y}|x)$. We define the set probability as the sum of sequences probabilities for all sequence permutations of the set, namely $p(\mathbf{y}|x) = \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$. Based on this formulation, an RNN also defines a probability distribution over all possible sets indirectly since $\sum_{\mathbf{y}} p(\mathbf{y}|x) = \sum_{\mathbf{y}} \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x) = \sum_{\mathbf{s}} p(\mathbf{s}|x) = 1$. (For this equation to hold, in theory, we should also consider permutations $\mathbf{s}$ with repeated labels, such as $(1, 2, 3, 1)$. But in practice, we find it very rare for RNN to actually generate sequences with repeated labels in our setup, and whether allowing repetition or not does not make much difference.)

In standard maximum likelihood training, one wishes to maximize the likelihood of given label sets, namely, $\prod_{n=1}^{N} p(\mathbf{y}^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)})$, or equivalently,

$$\textit{maximize} \sum_{n=1}^{N} \log \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)}) \tag{5}$$

3.1 How is our new formulation different?

This training objective (5) looks similar to the objective (3) considered in previous work (Vinyals et al., 2016), but in fact they correspond to very different transformations. Under the maximum likelihood framework, our objective (5) corresponds to the transformation $p(\mathbf{y}|x) = \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$, while objective (3) corresponds to the transformation $p(\mathbf{y}|x) = \prod_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$. The latter transformation does not define a valid probability distribution over $\mathbf{y}$ (i.e., $\sum_{\mathbf{y}} p(\mathbf{y}|x) \neq 1$), and it has an undesired consequence in practical model training: because of the multiplication operation, the RNN model has to assign equally high probabilities to all sequence permutations of the given label set in order to maximize the set probability. If only some sequence permutations receive high probabilities while others receive low probabilities, the set probability computed as the product of sequence probabilities will still be low. In other words, if for each document, RNN finds one good way of ordering relevant labels (such as hierarchically) and allocates most of the probability mass to the sequence in that order, the model still assigns low probabilities to the ground truth label sets and will be penalized heavily. As a consequence the model has little freedom in discovering and concentrating on some natural label order. In contrast, with our proposed training objective, in which the multiplication operation is replaced by the summation operation, it suffices to find only one reasonable permutation of the labels for each document. It is worth noting that different documents can have different label orders; thus our proposed training objective gives the RNN model far more freedom on label order. The other two objectives (2) and (4) proposed in (Vinyals et al., 2016) are less restrictive than (3), but they have to work in conjunction with (3) because of the self reinforcement issue. Our proposed training objective has a natural probabilistic interpretation, and does not suffer from self reinforcement issue. Thus it can serve as a stand alone training objective. Also, using Jensen's inequality, one can show that objective (3) is maximizing a lower bound on the log-likelihood, while objective (5) is maximizing it directly.

3.2 Training by Maximizing Set Probability

Training an RNN model with the proposed objective (5) requires summing up sequence (permutation) probabilities for a set $\mathbf{y}$, where $|\mathbf{y}|$ is the cardinality of the set. Thus evaluating this objective exactly can be intractable. We can approximate this sum by only considering the top K highest probability sequences produced by the RNN model. We introduce a variant of beam search for sets with width K and with the search candidates in each step restricted to only labels in the set (see Algorithm 1 with $ALL = 1$). This approximate inference procedure is carried out repeatedly before each batch training step, in order to find highest probability sequences for all training instances occurring in that batch. The overall training procedure is summarized in Algorithm 2.

3.3 Predicting the Most Probable Set

The transformation $p(\mathbf{y}|x) = \sum_{\mathbf{s} \in \pi(\mathbf{y})} p(\mathbf{s}|x)$ also naturally leads to a prediction procedure, which is different from the previous standard of directly using most probable sequence as a set. We instead aim to find the most likely set $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|x)$, which involves summing up probabilities for all of its permutations. To make it tractable, we propose a two-level beam search procedure. First we run standard RNN beam search (Algorithm 1 with $ALL = 0$) to generate a list of highest probability sequences. We then consider the label set associated with each label sequence. For each set, we evaluate its probability using the same approximate summation procedure as the one used during model training (Algorithm 1 with $ALL = 1$): we run our modified beam search to find the top few highest probability sequences associated with the set and sum up their

Algorithm 1: Beam_Search

```
Input : Instance $x$
        Subset of labels considered $G \subset \mathcal{L}$
        Boolean flag $ALL$: 1 if sequences must contain all $G$ labels; 0 if partial sequences are allowed
Output: A list of top sequences and the associated probabilities
Let $\mathbf{s}_1$,$\mathbf{s}_2$,...,$\mathbf{s}_K$ be the top $K$ sequences found so far. Initially, all $K$ sequences are empty. $\oplus$ means concatenation.
while true do
    // Step 1: Generate Candidate Sequences from each existing sequence $s_k \in K$ and all possible new labels $l \in G$:
    Expand all non-stopped sequences:
    $C = \{\mathbf{s}_k \oplus l | l \in G, STOP \notin s_k\}$
    Include stopped sequences:
    $C = C \cup \{\mathbf{s}_k | STOP \in s_k\}$
    Terminate non-stopped sequences:
    if $ALL == 0$ then
        $C = C \cup \{\mathbf{s}_k \oplus STOP | STOP \notin s_k\}$
    end
    // Step 2: Select highest probabilities sequences from candidate set $C$
    $K$ = topK-argmax$_k\{$prob$[s_k] | s_k \in C\}$
    if all top $K$ sequences end with $STOP$ or contain all labels in $G$ then
        Terminate the algorithm
    end
end
return sequence list $\mathbf{s}_1$,$\mathbf{s}_2$,...,$\mathbf{s}_K$ and the associated probabilities
```

Algorithm 2: Training method for set-RNN

```
Input  : Multi-label dataset
         $(x^{(n)}, \mathbf{y}^{(n)}), n = 1, 2, ..., N$
Output: Trained RNN model parameters
foreach batch do
    foreach $(x^n, \mathbf{y}^n)$ in the batch do
        Get top $K$ sequences :
        $\{\mathbf{s}_1^n, ..., \mathbf{s}_K^n, p(\mathbf{s}_1^n|x^n), ..., p(\mathbf{s}_K^n|x^n)\}$=
          = Beam_Search$(x^n, \mathbf{y}^n, ALL = 1)$
    end
    Update model parameters by maximizing
    $\sum_{(x^n,\mathbf{y}^n)\in \text{batch}} \log \sum_{\mathbf{s}\in\{\mathbf{s}_1^n,...,\mathbf{s}_K^n\}} p(\mathbf{s}|x^n)$
end
```

Algorithm 3: Prediction Method for set-RNN

```
Input  : Instance $x$
Output: Predicted label set $\hat{\mathbf{y}}$
Obtain $K$ highest probability sequences :
$\{\mathbf{s}_1, ..., \mathbf{s}_K\}$ = Beam_Search(x,$\mathcal{L}$, $ALL = 0$)
Map each sequence $\mathbf{s}_k$ to the corresponding
  set $\mathbf{y}_k$ and remove duplicate sets (if any)
foreach $\mathbf{y}_k$ do
    Get $K$ most probable sequences
      associated with $\mathbf{y}_k$ and their
      probabilities :
    $\{\mathbf{s'}_1, ..., \mathbf{s'}_K, p(\mathbf{s}'_1|x), ..., p(\mathbf{s}'_K|x)\}$=
          = Beam_Search(x,$\mathbf{y}_k$, $ALL = 1$)
    Set probability is approx by summing up :
      $p(\mathbf{y}_k|x) \approx \sum_{\mathbf{s}\in\{\mathbf{s}'_1,...,\mathbf{s}'_K\}} p(\mathbf{s}|x)$
end
$\hat{\mathbf{y}} = argmax_{\mathbf{y}_k}(p(\mathbf{y}_k|x))$
```

probabilities. Among these sets that we have evaluated, we choose the one with the highest probability as the prediction. The overall prediction procedure is summarized in Algorithm 3. As we shall show in case study, the most probable set may not correspond to the most probable sequence; these are certainly cases where our method has an advantage.

Both our method and the competitor state-of-the-art (Vinyals-RNNs) are at most K times slower than a vanilla-RNN, due to the time spent on dealing with K permutations per datapoint. Our proposed method is about as fast as the Vinyals-RNN methods, except for the Vinyals-RNN-uniform which is a bit faster (by a factor of 1.5) because its epochs do not run the additional forward pass.

4 Results and Analysis

4.1 Experimental Setup

We test our proposed set-RNN method on 4 real-world datasets, RCV1-v2, Slashdot, TheGuardian, and Arxiv Academic Paper Dataset (AAPD) (Yang et al., 2018). We take the public RCV1-v2 release[1] and randomly sample 50,000 documents. We crawl Slashdot and TheGuardian documents from their websites[2] and treat the official editor tags as ground truth. We also gather a list of user tags[3] for each document and treat them as additional features. For AAPD dataset, we follow the same train/test split as in (Yang et al., 2018). Table 2 contains statistics of these four datasets. Links to document, official editor tags, and user tags are avaliable at `http://www.ccis.neu.edu/home/kechenqin`.

Data	#Train	#Test	Cardinality	#Labels	Doc length
Slashdot	19,258	4,814	4.15	291	64
RCV1-v2	40,000	10,000	3.17	101	121
TheGuardian	37,638	9,409	7.41	1,527	505
AAPD	53,840	1,000	2.41	54	163

Table 2: Statistics of the datasets.

To process documents, we filter out stopwords and punctuations. Each document is truncated to have maximum 500 words for TheGuardian and AAPD, and 120 for Slashdot and RCV1-v2. Zero padding is used if the document contains less words than the maximum number. Numbers and out-of-vocabulary words are replaced with special tokens. Words, user tags and labels are all encoded as 300-dimensional vectors using WORD2VEC (Mikolov et al., 2013).

We implement RNNs with attention using TENSORFLOW-1.4.0 (Abadi et al., 2016). The dynamic function for RNNs is chosen to be Gated recurrent units (GRU) with 2 layers and at most 50 units in decoder. The size of the GRU unit is 300. We set dropout rate to 0.3, and train the model with Adam optimizer (Kingma and Ba, 2014) with learning rate 0.0005. Beam size is set to be 12 at both training and inference stages. We adopt *label-F1* (average F1 over labels) and *instance-F1*

---

[1] `http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

[2] Slashdot: `https://slashdot.org/` Note that there is another public Slashdot multi-label dataset (Read et al., 2009) but we do not use that one because it is quite small. TheGuardian: `https://www.theguardian.com`

[3] `www.zubiaga.org/datasets/socialbm0311/`

Methods	Slashdot		RCV1-v2		TheGuardian		AAPD			
	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	hamming-loss	micro-F1
BR	.271	.484	.486	.802	.292	.572	.529	.654	.0230	.685
BR-support	.247	.516	.486	.805	.296	.594	.545	.689	**.0228**	.696
PCC	.279	.480	.595	.818	-	-	.541	.688	.0255	.682
seq2seq-RNN	.270	.528	.561	.824	.331	.603	.510	.708	.0254	.701
Vinyals-RNN-uniform	.279	.527	.578	.826	.313	.567	.532	.721	.0241	.711
Vinyals-RNN-sample	.300	.531	.590	.828	.339	.597	.527	.706	.0259	.697
Vinyals-RNN-max	.293	.530	.588	.829	.343	.599	.535	.709	.0256	.700
Vinyals-RNN-max-direct	.226	.518	.539	.808	.313	.583	.490	.702	.0257	.694
SGM	-	-	-	-	-	-	-	-	.0245	.710
set-RNN	**.310**	**.538**	**.607**	**.838**	**.361**	**.607**	**.548**	**.731**	.0241	**.720**

Table 3: Comparison of different approaches. "-" means result not available. For *hamming loss*, the lower the value is, the better the model performs. For all other measures, the higher the better.

Methods	Slashdot		RCV1-v2		TheGuardian		AAPD	
	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1
seq2seq-RNN	.270→.269	.528→.528	.561→.561	.824→.824	.331→.336	.603→.603	.510→.511	.708→.709
Vinyals-RNN-uniform	.279→.288	**.527→.537**	.578→.587	.826→.833	**.313→.336**	**.567→.585**	**.532→.542**	.721→.724
Vinyals-RNN-sample	.300→.303	.531→.537	.590→.597	.828→.833	**.339→.351**	.597→.602	.527→.530	.706→.708
Vinyals-RNN-max	.293→.301	.530→.535	.588→.585	.829→.830	.343→.352	.599→.604	.535→.537	.709→.712
Vinyals-RNN-max-direct	.226→.228	.518→.519	.539→.538	.808→.808	.313→.316	.583→.584	.490→.490	.702→.701
set-RNN	**.297→.310**	**.528→.538**	**.593→.607**	.831→.838	**.349→.361**	**.595→.607**	.548→.548	.728→.731

Table 4: Predicting the most probable sequence vs. predicting the most probable set. Numbers before the arrow: predicting the most probable sequence. Numbers after the arrow: predicting the most probable set. We highlight scores which get significantly improved in bold (improvement is larger than 0.01).

(average F1 over instances) as our main evaluation metrics, as defined below:

$$\text{label-F1} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{2\sum_{n=1}^{N} y_\ell^{(n)} \hat{y}_\ell^{(n)}}{\sum_{n=1}^{N} y_\ell^{(n)} + \sum_{n=1}^{N} \hat{y}_\ell^{(n)}}$$

$$\text{instance-F1} = \frac{1}{N}\sum_{n=1}^{N} \frac{2\sum_{\ell=1}^{L} y_\ell^{(n)} \hat{y}_\ell^{(n)}}{\sum_{\ell=1}^{L} y_\ell^{(n)} + \sum_{\ell=1}^{L} \hat{y}_\ell^{(n)}}$$

where for each instance n, $y_\ell^{(n)} = 1$ if label ℓ is a given label in ground truth; $\hat{y}_\ell^{(n)} = 1$ if label ℓ is a predicted label.

We compare our method with the following methods:

- **Binary Relevance (BR)** (Tsoumakas and Katakis, 2007) with both independent training and prediction;
- **Binary Relevance with support inference (BR-support)** (Wang et al., 2018) which trains binary classifiers independently but imposes label constraints at prediction time by only considering label sets observed during training, namely $\hat{\mathbf{y}} = \arg\max_{\text{observed } \mathbf{y}} \prod_{\ell=1}^{L} p(y_\ell|x)$;
- **Probabilistic Classifier Chain (PCC)** (Dembczynski et al., 2010) which transforms the multi-label classification task into a chain of binary classification problems. Predictions are made with Beam Search.
- **Sequence to Sequence RNN (seq2seq-RNN)** (Nam et al., 2017) which maps each set to a sequence by decreasing label frequency and solves the multi-label task with an RNN designed for sequence prediction (see Table 1).
- **Vinyals-RNN-uniform, Vinyals-RNN-sample, and Vinyals-RNN-max** are three variants of RNNs proposed by (Vinyals et al., 2016). They are trained with different objectives that correspond to different transformations between sets and sequences. See Table 1 for a summary of their training objectives. Following the approach taken by (Vinyals et al., 2016), Vinyals-RNN-sample and Vinyals-RNN-max are initialized by Vinyals-RNN-uniform. We have also tested training Vinyals-RNN-max directly without having Vinyals-RNN-uniform as an initialization, and we name it as **Vinyals-RNN-max-direct**.
- **Sequence Generation Model (SGM)** (Yang et al., 2018) which trains the RNN model

similar to seq2seq-RNN but uses a new decoder structure that computes a weighted global embedding based on all labels as opposed to just the top one at each timestep.

In BR and PCC, logistic regressions with L1 and L2 regularizations are used as the underlying binary classifiers. seq2seq-RNN, PCC, and SGM rely on a particular label order. We adopt the decreasing label frequency order, which is the most popular choice.

4.2 Experimental Results

Table 3 shows the performance of different methods in terms of *label-F1* and *instance-F1*. The SGM results are taken directly from (Yang et al., 2018), and are originally reported only on AAPD dataset in terms of *hamming-loss* and *micro-F1*. Definitions of these two metrics can be found in (Koyejo et al., 2015).

Our method performs the best in all metrics on all datasets (except hamming loss on AAPD, see table 3). In general, RNN based methods perform better than traditional methods BR, BR-support and PCC. Among the Vinyals-RNN variants, Vinyals-RNN-max and Vinyals-sample work the best and have similar performance. However, they have to be initialized by Vinyals-RNN-uniform. Otherwise, the training gets stuck in early stage and the performance degrades significantly. One can see the clear degradation by comparing the Vinyals-RNN-max row (with initialization) with the Vinyals-RNN-max-direct row (without initialization). By contrast, our training objective in set-RNN does not suffer from this issue and can serve as a stable stand alone training objective.

On TheGuardian dataset, set-RNN performs slightly better than seq2seq-RNN in terms of instance-F1, but much better in terms of label-F1. It is known that instance-F1 is basically determined by the popular labels' performance while label-F1 is also sensitive to the performance on rare labels. Figure 1 shows that set-RNN predicts rare labels better than seq2seq-RNN.

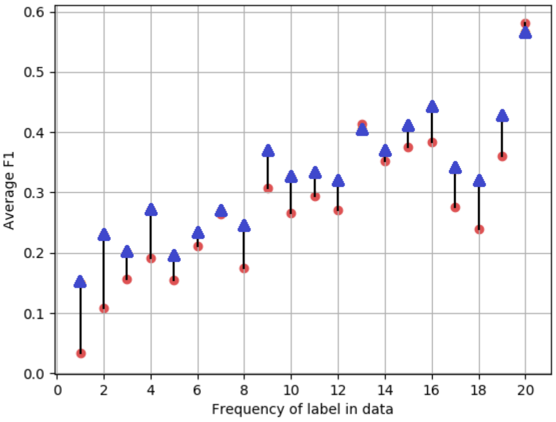


Figure 1: Average F1 over rare labels with the same frequency on TheGuardian dataset. Blue(Δ)=set-RNN, Red($\cdot$)=seq2seq-RNN.

Next we analyze how much benefit our new set prediction strategy brings in. For each RNN-based method, we test two prediction strategies: 1) finding the sequence with the highest probability and outputting the corresponding set (this is the default prediction strategy for all models except set-RNN); 2) outputting the set with the highest probability (this is the default prediction strategy for set-RNN). Table 4 shows how each method performs with these two prediction strategies. One can see that Vinyals-RNN-uniform and set-RNN benefit most from predicting the top set, Vinyals-RNN-sample, Vinyals-RNN-max and Vinyals-RNN-max-direct benefit less, and seq2seq RNN does not benefit at all. Intuitively, for the top-set prediction to be different from the top-sequence prediction, the model has to spread probability mass across different sequence permutations of the same set.

4.3 Analysis: Sequence Probability Dsitribution

Results in Table 4 motivates us to check how sharply (or uniformly) distributed the probabilities are over different sequence permutations of the predicted set. We first normalize these sequence probabilities related to the predicted set and then compute the entropy. To make predictions with different set sizes (and hence different number of sequence permutations) comparable, we further divide the entropy by the logarithm of number of sequences. Smaller entropy values indicate a sharper distributions. The results are shown in Figure 2.

seq2seq-RNN trained with fixed label order and standard RNN objective (1) generates very sharp sequence distributions. It basically only assigns probability to one sequence in the given order. The entropy is close to 0. In this case, predicting the set is no different than predicting the top sequence (see Table 4). On the other extreme is Vinyals-RNN-uniform, trained with objective (3), which spreads probabilities across many se-

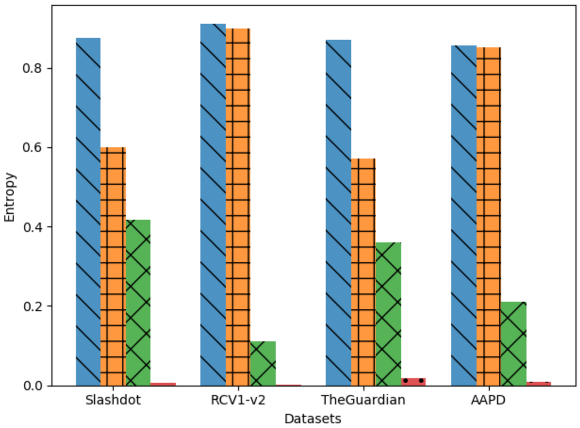


Figure 2: Entropy of sequence probability distribution for each model. Blue(\)=Vinyals-RNN-uniform, Orange(+)=set-RNN, Green(×)=Vinyals-RNN-max, Red(·)=seq2seq-RNN.

quences, and leads to the highest entropy among all models tested (the uniform distribution has the max entropy of 1). From Table 4, we see that by summing up sequence probabilities and predicting the most probable set, Vinyals-RNN-uniform's performance improves. But as discussed earlier, training with the objective (3) makes it impossible for the model to discover and concentrate on a particular natural label order (represented by a sequence). Overall Vinyals-RNN-uniform is not competitive even with the set-prediction enhancement. Between the above two extremes are Vinyals-RNN-max and set-RNN (we have omitted Vinyals-RNN-sample and Vinyals-RNN-max-direct here as they are similar to Vinyals-RNN-max). Both models are allowed to assign probability mass to a subset of sequences. Vinyals-RNN-max produces sharper sequence distributions than set-RNN, because Vinyals-RNN-max has the incentive to allocate most of the probability mass to the most probable sequence due to the max operator in its training objective (2). From Table 4, one can see that set-RNN clearly benefits from summing up sequence probabilities and predicting the most probable set while Vinyals-RNN-max does not benefit much. Therefore, the sequence probability summation is best used in both training and prediction, as in our proposed method.

Comparing 4 datasets in Table 4, we also see that Slashdot and TheGuardian, which have larger label cardinalities (therefore more permutations for one set potentially), benefit more from predicting the most probable set than RCV1 and AAPD, which have smaller label cardinalities.

5 Case Analysis

We further demonstrate how set-RNN works with two examples. In the first example from the RCV1-v2 dataset, the most probable set predicted by set-RNN (which is also the correct set in this example) does not come from the most probable sequence. Top sequences in decreasing probability order are listed in Table 5. The correct label set {forex, markets, equity, money markets, metals trading, commodity} has the maximum total probability of 0.161, but does not match the top sequence.

PROB	SEQUENCE
0.0236	equity, markets, money markets, forex
0.0196	**forex, markets, equity, money markets, metals trading, commodity**
0.0194	**equity, markets, forex, money markets, metals trading, commodity**
0.0159	**markets, equity, forex, money markets, metals trading, commodity**
0.0157	**forex, money markets, equity, metals trading, markets, commodity**
0.0153	**forex, money markets, markets, equity, metals trading, commodity**
0.0148	markets, equity, money markets, forex
0.0143	**money markets, equity, metals trading, commodity, forex, markets**
0.0123	**markets, money markets, equity, metals trading, commodity, forex**
0.0110	**markets, equity, forex, money markets, commodity, metals trading**
0.0107	**forex, markets, equity, money markets, commodity, metals trading**
0.0094	**forex, money markets, equity, markets, metals trading, commodity**

Table 5: The set-RNN predicted set (also the correct set) {forex, markets, equity, money markets, metals trading, commodity} has the max total probability of 0.161, but does not match the top sequence. Sequences for the correct set are in bold.

Next we demonstrate the issue with prescribing the sequence order in seq2seq-RNN with a TheGuardian example[4]. Figure 3 shows the predictions made by seq2seq-RNN and our method. In this particular example the top sequence agrees with the top set in our method's prediction so we can just analyze the top sequence. seq2seq-RNN predicts `Tate Modern` (incorrect but more popular label) while we predict `Tate Britain` (correct but less popular label). The seq2seq predicted sequence is in the decreasing label frequency order while our predicted sequence is not. In the training data, `Exhibition` is more frequent than `Tate Britain` and `Tate Modern`. If we arrange labels by decreasing frequency, `Exhibition` is immediately followed by `Tate Modern` 19 times, and by `Tate Britain` only 3 times. So it is far more likely to have `Tate Modern` than `Tate Britain` after `Exhibition`. However, at the set level, `Exhibition` and `Tate Modern` co-occurs 22 times while `Exhibition` and `Tate Britain`

---

[4] This document can be viewed at `http://www.guardian.co.uk/artanddesign/jonathanjonesblog/2009/apr/08/altermodernism-nicolas-bourriaud`

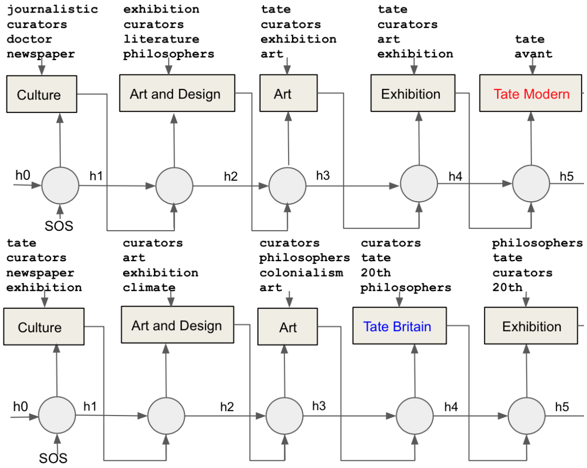


Figure 3: Top: best sequence by seq2seq-RNN; bottom: best sequence by set-RNN. Above models, at each time, we list the top unigrams selected by attention.

co-occurs 12 times, so the difference is not so dramatic. In this case, imposing the sequence order biases the probability estimation and leads to incorrect predictions.

6 Conclusion

In this work, we present an adaptation of RNN sequence models to the problem of multi-label classification for text. RNN only directly defines probabilities for sequences, but not for sets. Different from previous approaches, which either transform a set to a sequence in some pre-specified order, or relate the sequence probability to the set probability in some ad hoc way, our formulation is derived from a principled notion of set probability. We define the set probability as the sum of all corresponding sequence permutation probabilities. We derive a new training objective that maximizes the set probability and a new prediction objective that finds the most probable set. These new objectives are theoretically more appealing than existing ones, because they give the RNN model more freedom to automatically discover and utilize the best label orders.

Acknowledgements

We thank reviewers and Krzysztof Dembczyński for their helpful comments, Xiaofeng Yang for her help on writing, and Bingyu Wang for his help on proofreading. This work has been generously supported through a grant from the Massachusetts General Physicians Organization.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. 2018. Order-free RNN with visual attention for multi-label classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 279–286.

Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Proceedings of the 14th ACM international con-*

ference on Information and knowledge management, pages 195–200. ACM.

Jiren Jin and Hideki Nakayama. 2016. Annotation order matters: Recurrent image annotator for arbitrary length image tagging. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2452–2457.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Consistent multilabel classification. In *Advances in Neural Information Processing Systems*, pages 3321–3329.

Cheng Li, Bingyu Wang, Virgil Pavlu, and Javed Aslam. 2016. Conditional bernoulli mixtures for multi-label classification. In *International Conference on Machine Learning*, pages 2482–2491.

Weiwei Liu and Ivor Tsang. 2015. On the optimality of classifier chain for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 712–720.

Bruce T Lowerre. 1976. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J. Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5419–5429.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. *CoRR*, abs/1511.06391.

Bingyu Wang, Cheng Li, Virgil Pavlu, and Jay Aslam. 2018. A pipeline for optimizing f1-measure in multi-label text classification. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 913–918. IEEE.

Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. CNN-RNN: A unified framework for multi-label image classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2285–2294.

Jingya Wang, Xiatian Zhu, Shaogang Gong, and Wei Li. 2017a. Attribute recognition by joint recurrent learning of context and correlation. *CoRR*, abs/1709.08553.

Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. 2017b. Multi-label image recognition by recurrently discovering attentional regions. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 464–472.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. *CoRR*, abs/1806.04822.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, and Jianfeng Lu. 2016. Multi-label image classification with regional latent semantic dependencies. *CoRR*, abs/1612.01082.

Customizing Grapheme-to-Phoneme System for Non-Trivial Transcription Problems in Bangla Language

Sudipta Saha Shubha[1], **Nafis Sadeq**[1], **Shafayat Ahmed**[1], **Md. Nahidul Islam**[1], **Muhammad Abdullah Adnan**[1], **Md. Yasin Ali Khan**[2], and **Mohammad Zuberul Islam**[2]

[1]Bangladesh University of Engineering and Technology (BUET)
[2]Samsung R&D Institute, Bangladesh
`{sudipta,nafis,shafayat,nahid.rimon}@ra.cse.buet.ac.bd`
`adnan@cse.buet.ac.bd,{yasin.ali,m.zuberul}@samsung.com`

Abstract

Grapheme to phoneme (G2P) conversion is an integral part in various text and speech processing systems, such as: Text to Speech system, Speech Recognition system, etc. The existing methodologies for G2P conversion in Bangla language are mostly rule-based. However, data-driven approaches have proved their superiority over rule-based approaches for large-scale G2P conversion in other languages, such as: English, German, etc. As the performance of data-driven approaches for G2P conversion depend largely on pronunciation lexicon on which the system is trained, in this paper, we investigate on developing an improved training lexicon by identifying and categorizing the critical cases in Bangla language and include those critical cases in training lexicon for developing a robust G2P conversion system in Bangla language. Additionally, we have incorporated nasal vowels in our proposed phoneme list. Our methodology outperforms other state-of-the-art approaches for G2P conversion in Bangla language.

1 Introduction

Grapheme to phoneme (G2P) conversion provides a mapping between a word and its pronunciation. Such mapping provides opportunity for a non-native person to learn the correct pronunciation of words of a foreign language. Moreover, in modern Text to Speech (TTS) and Automatic Speech Recognition (ASR) systems, G2P conversion is an integral task.

The task of G2P conversion is generally language specific due to language specific conventions, rules, pronunciation constraints, etc. In this paper, we focus on Modern Standard Bangla. An example of G2P conversion in Bangla language: phonetic transcription of অনুশীলন (practice) is /o n u sh i l O n/. (Please refer to Table 1 for our phoneme symbols.)

The simplest means of G2P conversion is to build up a lexicon or dictionary containing the mapping from words to their corresponding pronunciations. However, it fails to provide pronunciations for unknown words and inclusion of newer words increases memory requirement. In another approach (Mosaddeque et al., 2006), there are predefined rules for the conversion of a word to its pronunciation. Though such rule-based approach can work for any word, the system becomes complex when it tries to formulate rules for incorporating all irregularities of pronunciation in a language.

Clearly, these approaches are not feasible for large-scale G2P conversion which is necessary in any modern TTS or ASR system. Data-driven machine learning approaches have great potential in such large-scale G2P conversion (Rao et al., 2015). In such an approach, a machine learning model predicts the phoneme conversion of a grapheme, being trained on a lexicon. A predominant work following such approach in Bangla language is by Google (Gutkin et al., 2016), where they train their system using $37K$ words and achieve word-level accuracy of 81.5%. However, a system trained on their lexicon will face several shortcomings, such as: কাদা(mud) and কাঁদা(to cry) are pronounced differently but will have same phoneme representation in their system as: /k a d a/. Similarly, পরী(fairy) and পড়ি(to read) are pronounced differently but will have same phoneme representation in their system as: /p o r i/. Moreover, G2P system trained on their lexicon performs poorly on our identified critical cases from the most frequent $100K$ words (Table 3).

Being motivated to increase the accuracy of grapheme to phoneme conversion in Bangla language, which will also perform well for critical inputs, we have developed a customized and robust G2P system for Bangla language.

Proceedings of NAACL-HLT 2019, pages 3191–3200
Minneapolis, Minnesota, June 2 - June 7, 2019.

Our major contributions are as follows:

(i) We identify and categorize the critical cases for grapheme to phoneme (G2P) conversion in Bangla language by analyzing the most frequent $100K$ words.

(ii) We enrich the training lexicon for developing a robust G2P conversion system in Bangla language that performs much better for critical cases compared to other state-of-the-art G2P systems.

(iii) We perform phonetic transcriptions considering nasal vowels as separate phonemes.

(iv) We perform extensive simulations on large-scale dataset and show that our methodology outperforms other state-of-the-art approaches for G2P conversion in Bangla language by providing word-level accuracy of 90.2%.

The rest of the paper is organized as follows: we discuss the previous works in Section 2, our phoneme list in Section 3, identification of critical cases and categorization of errors in Section 4, development of our system in Section 5, experimental results in Section 6, and conclusion and future works in Section 7.

2 Previous Works

The research works for G2P in English are quite extensive. Chen (2003) investigate machine learning based systems for G2P in English. They experiment with joint maximum entropy n-gram model, conditional maximum entropy model, etc. Yao and Zweig (2015) utilize bi-directional LSTM (Long Short Term Memory) recurrent neural network for G2P and achieve 5.45% PER on CMU dictionary (air, 2015). Thu et al. (2016) show comparisons among various machine learning algorithms for G2P in Burmese language. Joint sequence n-gram models aim to discover joint vocabulary consisting of graphemes and phonemes through the alignment of graphemes and phonemes. Bisani and Ney (2008) develop a joint-sequence model for G2P. Novak (2012), Novak et al. (2013) are other prominent works working on this model. Neural sequence to sequence models are popular for G2P conversion. Some prominent works on such models are: Caruana (1997), Jiampojamarn et al. (2007), Sutskever et al. (2014), Yao and Zweig (2015), Jiampojamarn et al. (2007), Rao et al. (2015), Yao and Zweig (2015), Schnober et al. (2016), Tsvetkov et al. (2016), He et al. (2016), Wu et al. (2016), Johnson et al. (2016), Toshniwal and Livescu (2016), and Vaswani et al. (2017).

Again, another line of research deals with G2P conversion for more than one language. Such works include: Mana et al. (2001), Kim and Snyder (2012), Deri and Knight (2016), and Milde et al. (2017).

Most of the works related to G2P conversion that are focused on Bangla language, follow rule-based approach. Rule-based approach of Mosaddeque et al. (2006) provides accuracy of 97.01% on a previously seen corpus containing 736 words, but the system's accuracy is 81.95% on an previously unobserved corpus containing 8399 words. This work was extended by Alam et al. (2011) describing 3880 rules with an accuracy of 89.48% on another corpus. Basu et al. (2009) discuss a rule-based approach considering several information: parts-of-speech, subsequent context, etc. Their work describes only 21 rules and provides an accuracy of 91.48% on a corpus of 9294 words. Ghosh et al. (2010) provide a heuristic for G2P that takes into account parts-of-speech, orthographic, and contextual information. Their work provides 70% accuracy on a corpus containing of 755 words. A prominent work for data driven G2P in Bangla language is by Google (Gutkin et al. (2016)). They develop a lexicon and achieve word-level accuracy of 81.5%. Chowdhury et al. (2017) use conditional random field for G2P in Bangla. They report 14.88% phoneme error rate on Google lexicon.

3 Phoneme List

Our Phoneme symbols are provided in Table 1. This table is a good reference for the 47 phoneme symbols that we have followed in this paper and their corresponding International Phonetic Alphabet (IPA) symbols. Throughout the paper, we use these 47 phoneme symbols, not the IPA symbols. There is disagreement between linguists whether nasal vowels should be considered as separate phonemes (Barman, 2009). We added nasal vowels in our phoneme list to differentiate between a word with its nasalized counterpart, such as the word কাঁদা(to cry) and কাদা(mud). Here, /a/, /e/, /u/, /i/, /o/, /O/, /E/, /an/, /en/, /un/, /in/, /on/, /On/, /En/ are normal vowels, /ew/, /ow/, /uw/, /iw/ are weak vowels, and the rest are consonants.

Phoneme	IPA	Phoneme	IPA	Phoneme	IPA	Phoneme	IPA
i	i	On	ɔ̃	D	ɖ	m	m
u	u	an	ã	Dh	ɖʱ	r	r
e	e	k	k	t	t	R	ɽ
o	o	kh	kʰ	th	tʰ	l	l
E	ɛ	g	g	d	d	h	ɦ
O	ɔ	gh	gʱ	dh	dʱ	s	s
a	a	c	ʧ	p	p	sh	ʃ
in	ĩ	ch	ʧʰ	ph	pʰ	iw	i̯
un	ũ	j	dʒ	b	b	ew	e̯
en	ẽ	jh	dʒʱ	bh	bʱ	ow	o̯
on	õ	T	ʈ	N	ŋ	uw	u̯
En	æ̃	Th	ʈʰ	n	n		

Table 1: Our Phoneme Symbols with Their Corresponding IPA Symbols

4 Identification and Categorization of Non-Trivial Cases for Transcription

We envision of developing a robust G2P system that will perform reasonably well on any word in Bangla language. A G2P system that performs well on the most frequent words, should also do well on other words. With this motivation, we focus on increasing accuracy on the most frequent words. Especially, we are concerned about those words that are among the most frequent words but non-trivial or critical for phonetic transcription, i.e., current state-of-the-art methodologies perform poorly on these critical words. We investigate on identifying and categorizing such non-trivial or critical cases so that future research works can give special focus on developing methods for improving phonetic transcriptions of these critical words.

4.1 Identifying the Most Frequent Words

To get a hold of contemporary usage of Bangla language, we do extensive crawling. We crawled 42 websites of various Bangla newspapers, blogs, e-book libraries, wikipedia, etc. covering various domains such as: politics, economics, sports, drama, novel, story, education, entertainment, general knowledge, history, etc. After data cleaning and data normalization, we had about $10M$ sentences. We counted how many times each of the unique words appeared in those sentences. We then consider the most frequent $100K$ words and aim to identify the critical cases for phonetic transcription among these most frequent words.

4.2 Identifying the Critical Cases for Transcription

After changing the Google lexicon (of size $60K$ (around)) according to our phoneme symbols (Table 1), we prepare 4 versions of Google's lexicon of size $12K$, $24K$, $40K$, and $60K$ respectively for identifying the critical cases for phonetic transcription. Algorithm 1 shows prefix comparing algorithm that we use for compressing a phonetic lexicon or dictionary of grapheme sequence to phoneme sequence. The algorithm matches the prefix of consecutive words (grapheme sequence) of a sorted dictionary (sorted according to ascending order of grapheme sequence of a word) and keeps a word (with its corresponding phoneme sequence) only if it does not share its prefix with any other words. We run the algorithm successively 3 times, i.e., we use the destination dictionary of one iteration as the source dictionary of next iteration. Each iteration produces a minimized version of the basic lexicon (Google lexicon). After 3 iterations, the dictionary does not get any more compressed. We find the phonetic transcriptions of each of the $100K$ most frequent words using models trained on each of the 4 versions of Google's lexicon (basic + 3 minimized). So, from 4 models (each model trained on a version of the basic Google lexicon), we get 4 sets of transcriptions for the most frequent $100K$ words. For most of the words (around $70K$ words), we observe that the phonetic transcriptions are exactly same in each of the 4 set. However, for the remaining $30K$ words (29105 words to be exact), we observe that at least one set provides different transcription. We take these $30K$ words to be the critical cases. Our intuition is that if two G2P systems: one trained on a smaller version of the basic lexicon, and another trained on a larger version of the basic lexicon provide the same transcription for a word, then the word is a trivial case for phonetic transcription. We then manually verify the phonetic transcriptions of these $30K$ words taking help from 3 linguists and following Chowdhury (2016), and consider these $30K$ words as critical cases for phonetic transcription.

4.3 Categorizing the Critical Cases

We categorize the critical cases into 7 categories and observe the distribution of the critical transcriptions into these 7 categories. These 7 cate-

Algorithm 1 Algorithm for Compressing a Dictionary or Lexicon

```
$sd$ ← sorted sourceDictionary
$dd$ ← sorted destinationDictionary
$a.grs$ ← grapheme sequence of lexicon entry $a$
add $sd[0]$ to $dd$
$i = 1$
while $i \neq$ length($sd$) do
    $pw = sd[i - 1]$
    $cw = sd[i]$
    if $length(pw.grs) \geq 3$ & $pw.grs$ is prefix of $cw.grs$ then
        continue
    else
        add $cw$ to $dd$
    $i \leftarrow i + 1$
```

gories capture most of the errors. The categories are:

- **Open Close Vowel Confusion:** G2P system provides pronunciation as close vowel that should be pronounced as open vowel ideally, and vice-versa. For example, correct phoneme of ব্যাঙ (frog) is /b E n g/, but if G2P system provides output /b e n g/, then it is an error under this category as in the place of open vowel (here, /E/), G2P system is giving close vowel (here, /e/).
- **Inherent Vowel Confusion:** G2P system does not provide inherent vowel as output where there should be an inherent vowel ideally. For example, correct phoneme of সকাল (morning) is /sh O k a l/, but if G2P system provides output /sh k a l/, then it is an error under this category as the output of G2P does not give inherent vowel (here, /O/).
- **Diphthong Confusion:** G2P system does not provide falling diphthong in output where there should be a falling diphthong ideally. Or, system does not provide rising diphthong in output where there should be a rising diphthong ideally. For example, correct phoneme of সই (friend) is /sh o iw/, but if G2P system provides output /sh o i/, then it is an error under this category as the output of G2P does not capture the falling diphthong (here, /o iw/).
- **s or sh Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /sh/, and vice-versa. For example, correct phoneme sequence of সংগঠন (organization) is /sh O N g O Th o n/, but if G2P system provides output /s O N g O Th o n/, then it is an error under this category as the output of G2P gives /s/ in place of /sh/.
- **s or ch Confusion:** G2P system provides /s/ in phonetic transcription, where there should be /ch/, and vice-versa. For example, correct phoneme sequence of ছাতা (umbrella) is /ch a t a/, but if G2P system provides output /s a t a/, then it is an error under this category as the output of G2P gives /s/ in place of /ch/.
- **Nasal Confusion:** G2P system does not provide any nasal vowel where there should be a nasal vowel, and vice-versa. For example, correct phoneme sequence of চাঁদ (moon) is /c an d/, but if G2P system provides output /c a d/, then it is an error under this category as the output of G2P gives /a/ in place of /an/.
- **Other Vowel Confusion:** G2P system provides completely different vowel than the corresponding vowel that should ideally be in that position of the phoneme sequence. Note that, in the other error categories, for each position in the phoneme sequence, the generated and ideal phonemes were somehow related. But in this category, at a specific position of the phoneme sequence, the generated and ideal phonemes are completely different. For example, correct phoneme sequence of অধ্যবসায় (perseverance) is /o d dh o b O sh a ew/, but if G2P system provides output /o d dh a b O sh a ew/, then it is an error under this category as the output of G2P gives /a/ in place of /o/ (fourth phoneme).

Algorithm 2 compares a machine-generated lexicon with a reference lexicon (manually verified), where both the lexicons have same grapheme sequences, but the corresponding phoneme sequences may be different. This algorithm counts how many errors of each category are there in the machine generated lexicon. The algorithm takes each entry of the generated lexicon and increases the count of the corresponding error category (if an error is present there).

We train attention mechanism based Transformer model on each of the 4 lexicons and get

Algorithm 2 Comparing a Generated Lexicon (gl) with Reference Lexicon (rl)

```
$N \leftarrow$ total number of entries in each lexicon
$A, B, C, D, E, F, G$ are Open Close Vowel, s or sh, s or ch, Nasal, Dipthong, Other Vowel, and Inherent confusions, respectively, all initially zero
$H$ denotes other errors not captured by the 7 categories, initially zero
vl and wl are lists of vowels and weak vowels respectively
$a.phs \leftarrow$ phoneme sequence of lexicon entry $a$
$i \leftarrow 0$
while $i \neq N$ do
    $g = gl[i].phs$
    $r = rl[i].phs$
    $M = \min(length(g), length(r))$
    $j \leftarrow 0$
    while $j \neq M$ do
        $x = g[j]$
        $y = r[j]$
        if $x = y$ then
            continue
        $(x, y) \leftarrow sorted(x, y)$
        $Total\_error = Total\_error + 1$
        if $ocConfusion(x, y)$ then
            $A \leftarrow A + 1$
        else if $(x, y) = (``s", ``sh")$ then
            $B \leftarrow B + 1$
        else if $(x, y) = (``ch", ``s")$ then
            $C \leftarrow C + 1$
        else if $x + ``n" = y$ then
            $D \leftarrow D + 1$
        else if $x$ in wl or $y$ in wl then
            $E \leftarrow E + 1$
        else if $x$ in vl and $y$ in vl then
            $F \leftarrow F + 1$
        else if $removeVowel(g) = removeVowel(r)$ then
            $G \leftarrow G + 1$
        else
            $H \leftarrow H + 1$
        $j \leftarrow j + 1$
    $i \leftarrow i + 1$
```

Algorithm 3 Procedure: ocConfusion (x, y) (Checks if open close vowel confusion)

```
$ocSet \leftarrow [(``O", ``o"), (``E", ``e"),
        (``On", ``on"), (``En", ``en")]$
if $(x, y)$ in $ocSet$ then
    return True
else
    return False
```

Algorithm 4 Procedure: removeVowel (phoneme_sequence)

```
return phoneme_sequence removing all vowels from it
```

4 G2P models. We find the phoneme representation of $30K$ critical cases using each of the 4 G2P models. Using Algorithm 2, we count the errors of each category for each of the 4 models. We report the results in Table 2 and Figure 1. Here, other error denotes the errors that are not captured by these 7 categories. We see from these results that most of the errors are under Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions.

5 Developing an Improved G2P System for Bangla Language

We develop an improved lexicon and use two machine learning based models trained on our lexicon to develop an improved G2P system for Bangla language.

5.1 Developing an Improved Lexicon

For developing an improved lexicon, we include with Google's lexicon the manually verified $30K$ non-trivial or critical entries. Also, we include the $70K$ entries in which all of the 4 models (trained on each of the 4 versions of Google lexicon) unanimously agreed. After removing the repeated entries with same grapheme sequence, our lexicon consists of around $90K$ entries. In case of repeated entries having same grapheme sequence, we carefully keep only the entry that has been manually verified.

5.2 G2P Models

We use Neural Sequence to Sequence models. In these models, a conditional distribution of a sequence (here, phoneme sequence) is learned conditioned on another sequence (here, grapheme sequence). We train two following Sequence to Sequence models on our lexicon for G2P conversion:

LSTM-RNN: This is a plain Sequence to Sequence model that incorporates an encoder and decoder mechanism. Recurrent Neural Network

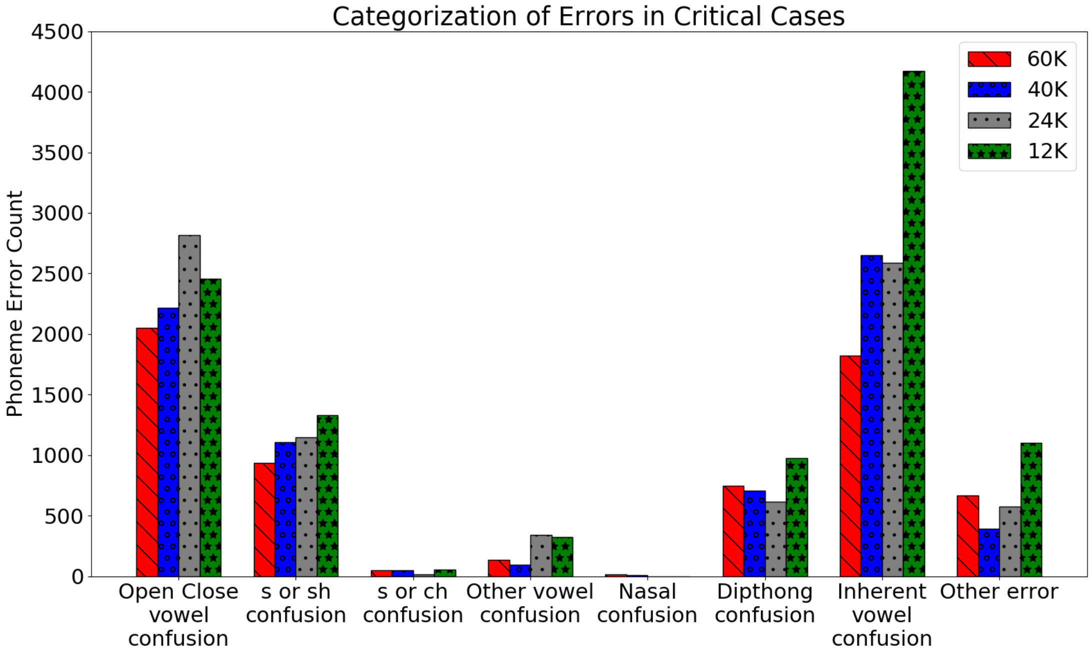


Figure 1: Categorization of errors in critical cases, here each of the $60K, 40K, 24K,$ and $12K$ denotes the model trained on that particular lexicon.

(RNN) is usually utilized in encoder and decoder design. For addressing vanishing gradient problem in RNN, Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is used. We follow Yao and Zweig (2015) for implementation.

Transformer Model: Transformer Model uses attention mechanism. Attention mechanism provides improvement upon plain Sequence to Sequence by easing the flow of information from source sequence to destination sequence. We follow Vaswani et al. (2017) for implementation.

We show the performance of both of these models in Section 6. We observe that Transformer Model provides higher token-level accuracy (lower Word Error Rate) than LSTM-RNN.

6 Experimental Results

We run extensive simulations and use two measures for evaluating the performances of the systems:

Word Error Rate (WER): For calculating Word Error Rate (WER), we use the following formula:

$$\text{WER} = \frac{E}{T}$$

where E denotes the number of words that have disagreement in their generated phoneme sequence and reference phoneme sequence, and T denotes the total number of words.

Phoneme Error Rate (PER): For calculating Phoneme Error Rate (PER), we use the following formula:

$$\text{PER} = \frac{I+S+D}{T}$$

where I, S, D denote respectively the total number of insertion, substitution, and deletion operations needed for all the words to align the generated phoneme sequence with the reference phoneme sequence for each word. T denotes the total number of phonemes present in all the words.

Our best performing model is Transformer Model. We use batch size of 4096. Our neural network has 3 hidden layers, each containing 256 nodes. We use a computer having 8GB RAM, Intel Core i7 CPU, and Nvidia Geforce 1050 GPU for running all of the simulations. For each model, we run the simulations for around $110K$ iterations taking around 5 hours.

6.1 Performance on Critical Cases

We report the experiment results of Google's lexicon on critical cases in Table 3. We do not report our lexicon here as critical cases are already included in our lexicon.

6.2 Performance Comparison In General

For comparing the performances of models trained on our lexicon and Google's lexicon, we randomly

Error Type	**60K**	**40K**	**24K**	**12K**
total error	6415	7222	8087	10400
Open Close Confusion (%)	32.0	30.7	34.8	23.6
Inherent Vowel Confusion (%)	28.4	36.7	32.0	40.1
s or sh confusion (%)	14.6	15.3	14.2	12.8
Dipthong confusion (%)	11.6	9.8	7.6	9.4
Other Vowel Confusion (%)	2.1	1.3	4.2	3.1
s or ch confusion (%)	0.8	0.7	0.2	0.5
Nasal Confusion (%)	0.2	0.1	0	0
Other Error (%)	10.4	5.4	7.1	10.6

Table 2: Error classification of $30K$ critical cases, here each of the four rightmost columns denotes the model trained on that particular lexicon.

Lexicon	**Model**	**WER (%)**	**PER (%)**
Google	LSTM-RNN	25.7	3.26
	Transformer Model	23.6	2.71

Table 3: Performance on Critical Cases

take 9000 entries from our manually verified $30K$ critical cases as test set. We use this test set for evaluating all the models. Though our actual lexicon contains these 9000 entries, we do not keep them in our lexicon while doing the experiments to fairly evaluate the performances of the lexicons. For both lexicons, we keep 90% of the lexicon in train set and remaining 10% in validation set. Table 4 shows the result. Models trained on our lexicon outperforms those trained on Google's lexicon by a significant margin. Moreover, Transformer Model performs better than LSTM-RNN.

Figure 2 and Table 5 categorize the errors of systems trained on 3 types of lexicons (Romanized version of our lexicon is discussed in section 6.3) by using Algorithm 2. Here, we report the results of Transformer Model only as it has been better performing that LSTM-RNN in our experiments. We observe most of the errors are related to Open Close vowel, s or sh, Diphthong, and Inherent Vowel confusions - this finding also conforms to Figure 1 and Table 2, which were error categorization of critical cases.

Lexicon	**Model**	**WER (%)**	**PER (%)**
Google	LSTM-RNN	17.1	2.32
	Transformer Model	14.8	1.88
Our Lexicon	LSTM-RNN	10.5	1.42
	Transformer Model	9.8	1.33

Table 4: Performance Comparison In General

Error Type	**Our Lexicon**	**Romanized Lexicon**	**Google Lexicon**
Total Error	1337	1406	2961
Inherent Vowel Confusion (%)	35.6	34.5	33.8
Open Close Confusion (%)	30.1	30.4	27.9
s or sh confusion (%)	13.3	13.0	12.1
Diphthong confusion (%)	10.7	9.8	13.1
Other Vowel Confusion (%)	1.9	4.3	2.7
s or ch confusion (%)	0.3	0.2	0.2
Nasal Confusion (%)	0.2	0.43	0.6
Other Error (%)	7.9	7.4	9.7

Table 5: Performance comparison on different error categories. Here each of the three rightmost columns denotes the model trained on that particular lexicon.

6.3 Effect of Romanization

For doing experiments on the effect of romanization on G2P conversion, we romanized all grapheme sequences in our lexicon to prepare romanized counterpart of our lexicon. During romanization, each grapheme symbol in Bangla is replaced with a single English letter except that if a consonant grapheme is not followed by a vowel grapheme, "O" was added after the romanized symbol of that consonant as the roman symbol for "অ", which is usually inherently pronounced in such cases. All the symbols used for romanization were completely disjoint to avoid any ambiguity in the lexicon. Figure 3 shows the WER and PER

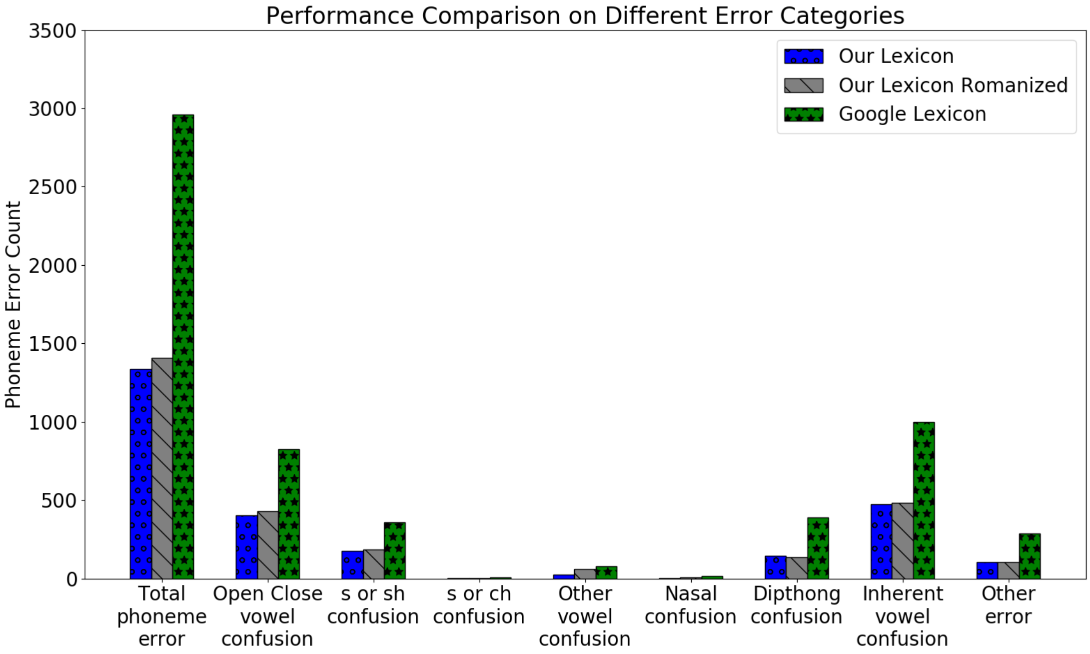


Figure 2: Performance Comparison on Different Error Categories

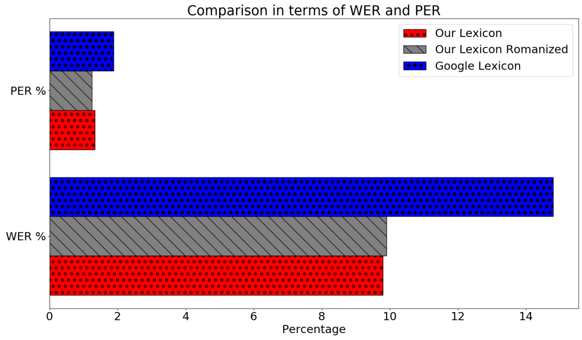


Figure 3: Comparison in terms of WER and PER

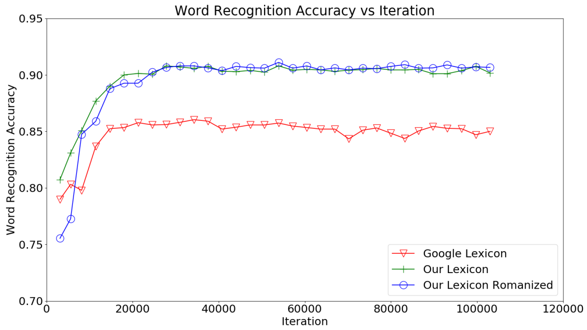


Figure 4: Word Recognition Accuracy vs Iteration

of systems trained on 3 types of lexicons. Here, we report the results of Transformer Model only as it has been better performing than LSTM-RNN in our experiments. Both versions of our lexicon perform better (lower WER and lower PER) than Google's lexicon. Also romanization does not significantly increase or decrease the performance.

Figures 4 and 5 show respectively the Word Recognition Accuracy (1−WER) and Phoneme Recognition Accuracy (1−PER) with respect to number of iterations run during simulation. Both versions of our lexicon perform better (higher Word Recognition Accuracy and higher Phoneme Recognition Accuracy) than Google's lexicon. Figure 6 shows Negative Log Perplexity vs number of iterations. Both versions of our lexicon provide higher negative log perplexity than Google's lexicon.

6.4 Effectiveness of Our Identified Critical Cases

In this section, we want to establish that the improved performance of our lexicon comes not only from the increase in number of training samples, but also due to the fact that the critical cases identified by our novel methodology have been added as training samples. For this, we prepare a new training lexicon by combining a portion of the Google lexicon with a portion of our identified critical cases. As we have kept $9K$ entries from the critical cases as our test set, we take the remaining $21K$ critical cases and combine them with the randomly taken $39K$ entries from Google lexicon to prepare a new lexicon of size $60K$. While taking entries from Google lexicon, we ensure that we do not take any repeated entry that has already been in the critical cases and added to the new lexicon.

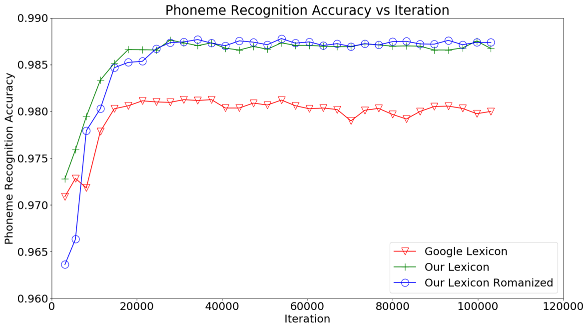


Figure 5: Phoneme Recognition Accuracy vs Iteration

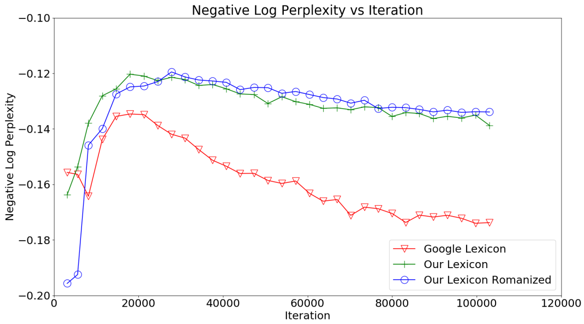


Figure 6: Negative Log Perplexity vs Iteration

Lexicon	**Model**	**WER (%)**	**PER (%)**
Google	LSTM-RNN	17.1	2.32
	Transformer Model	14.8	1.88
New Lexicon	LSTM-RNN	12.6	1.54
	Transformer Model	11.2	1.49

Table 6: Effectiveness of critical cases. Both lexicons are of size $60K$. New Lexicon consists of $21K$ critical cases and $39K$ entries from Google lexicon.

We then compare the performance of this new lexicon with the Google lexicon, both of which are of same size ($60K$), on our test set. The results are in Table 6. The results clearly show that even in the case of same sized lexicons, our identified critical cases can significantly improve the performance as evidenced by the lower WER and lower PER than those for the Google lexicon.

7 Conclusion and Future Works

In this paper, we have identified the critical cases, categorized the errors, and increased the current state-of-the-art accuracy in G2P conversion for Bangla language by developing an improved lexicon. In future, we will do classification of errors using unsupervised clustering approaches. Tables 2 and 5 both show that most of the errors occur in Open Close Vowel, s or sh, Diphthong, and Inherent Vowel confusion categories. In future, we will focus on each category to devise novel methodology for mitigating the errors of that category and in turn further increase the accuracy of G2P system in Bangla.

Acknowledgements

This research work is conducted at the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) and is supported by Samsung Research.

References

air. 2015. `https://svn.code.sf.net/p/cmusphinx/code/branches/cmudict/cmudict-0.7b`.

Firoj Alam, SM Murtoza Habib, and Mumit Khan. 2011. Bangla text to speech using festival. In *Conference on Human Language Technology for Development*, pages 154–161.

Binoy Barman. 2009. A contrastive analysis of english and bangla phonemics. *Dhaka University Journal of Linguistics*, 2(4):19–42.

Joyanta Basu, Tulika Basu, Mridusmita Mitra, and Shyamal Kr Das Mandal. 2009. Grapheme to phoneme (g2p) conversion for bangla. In *Speech Database and Assessments, 2009 Oriental COCOSDA International Conference on*, pages 66–71. IEEE.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Stanley F Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Eighth European Conference on Speech Communication and Technology*.

Jamil Chowdhury, editor. 2016. *Adhunik Bangla Ovidhan*. Bangla Academy, Bangla Academy, Dhaka - 1000.

Shammur Absar Chowdhury, Firoj Alam, Naira Khan, and Sheak RH Noori. 2017. Bangla grapheme to phoneme conversion using conditional random fields. In *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, pages 1–6. IEEE.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 399–408.

Krishnendu Ghosh, Ramu V Reddy, NP Narendra, S Maity, SG Koolagudi, and KS Rao. 2010. Grapheme to phoneme conversion in bengali for festival based tts framework. In *8th international conference on natural language processing (ICON)*. Macmillan Publishers.

Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat. 2016. Tts for low resource languages: A bangla synthesizer. In *LREC*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Young-Bum Kim and Benjamin Snyder. 2012. Universal grapheme-to-phoneme prediction over latin alphabets. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 332–343. Association for Computational Linguistics.

Franco Mana, Paolo Massimino, and Alberto Pacchiotti. 2001. Using machine learning techniques for grapheme to phoneme transcription. In *Seventh European Conference on Speech Communication and Technology*.

Benjamin Milde, Christoph Schmidt, and Joachim Köhler. 2017. Multitask sequence-to-sequence models for grapheme-to-phoneme conversion. *Proc. Interspeech 2017*, pages 2536–2540.

Ayesha Binte Mosaddeque, Naushad UzZaman, and Mumit Khan. 2006. Rule based automated pronunciation generator.

J. R. Novak. 2012. `https://github.com/AdolfVonKleist/Phonetisaurus`.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2013. Failure transitions for joint n-gram models and g2p conversion. In *INTERSPEECH*, pages 1821–1825.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. *arXiv preprint arXiv:1610.07796*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ye Kyaw Thu, Win Pa Pa, Yoshinori Sagisaka, and Naoto Iwahashi. 2016. Comparison of grapheme-to-phoneme conversion methods on a myanmar pronunciation dictionary. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 11–22.

Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 76–82. IEEE.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. *arXiv preprint arXiv:1605.03832*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Connecting Language and Knowledge with Heterogeneous Representations for Neural Relation Extraction

Peng Xu
Department of Computing Science
University of Alberta
Edmonton, Canada
pxu4@ualberta.ca

Denilson Barbosa
Department of Computing Science
University of Alberta
Edmonton, Canada
denilson@ualberta.ca

Abstract

Knowledge Bases (KBs) require constant updating to reflect changes to the world they represent. For general purpose KBs, this is often done through Relation Extraction (RE), the task of predicting KB relations expressed in text mentioning entities known to the KB. One way to improve RE is to use KB Embeddings (KBE) for link prediction. However, despite clear connections between RE and KBE, little has been done toward properly unifying these models systematically. We help close the gap with a framework that unifies the learning of RE and KBE models leading to significant improvements over the state-of-the-art in RE. The code is available at `https://github.com/billy-inn/HRERE`.

1 Introduction

Knowledge Bases (KBs) contain structured information about the world and are used in support of many natural language processing applications such as semantic search and question answering. Building KBs is a never-ending challenge because, as the world changes, new knowledge needs to be harvested while old knowledge needs to be revised. This motivates the work on the Relation Extraction (RE) task, whose goal is to assign a KB relation to a *phrase* connecting a pair of entities, which in turn can be used for updating the KB. The state-of-the-art in RE builds on neural models using distant (a.k.a. weak) supervision (Mintz et al., 2009) on large-scale corpora for training.

A task related to RE is that of Knowledge Base Embedding (KBE), which is concerned with representing KB entities and relations in a vector space for predicting missing links in the graph. Aiming to leverage the similarities between these tasks, Weston et al. (2013) were the first to show that *combining* predictions from RE and KBE models was beneficial for RE. However, the way in which they combine RE and KBE predictions is rather naive (namely, by adding those scores). To the best of our knowledge, there have been no systematic attempts to further unify RE and KBE, particularly during model *training*.

We seek to close this gap with HRERE (Heterogeneous REpresentations for neural Relation Extraction), a novel neural RE framework that learns language and knowledge representations *jointly*. Figure 1 gives an overview. HRERE's backbone is a bi-directional long short term memory (LSTM) network with multiple levels of attention to learn representations of text expressing relations. The knowledge representation machinery, borrowed from **ComplEx** (Trouillon et al., 2016), nudges the language model to agree with facts in the KB. Joint learning is guided by three loss functions: one for the language representation, another for the knowledge representation, and a third one to ensure these representations do not diverge. In effect, this contributes to HRERE's generalization power by preventing over-fitting by either model.

We build on state-of-the-art methods for learning the separate RE and KBE representations and on learning tools that allow us to scale to a moderately large training corpus. (We use a subset of Freebase with 3M entities as our KB.) We validate our approach on an established benchmark against state-of-the-art methods for RE, observing not only that our base model significantly outperforms previous methods, but also the fact that jointly learning the heterogeneous representations consistently brings in improvements. To the best of our knowledge, ours is the first principled framework to combine and jointly learn heterogeneous representations from both language and knowledge for the RE task.

Contributions. This paper describes and evaluates a novel neural framework for jointly learning

Proceedings of NAACL-HLT 2019, pages 3201–3206
Minneapolis, Minnesota, June 2 - June 7, 2019.

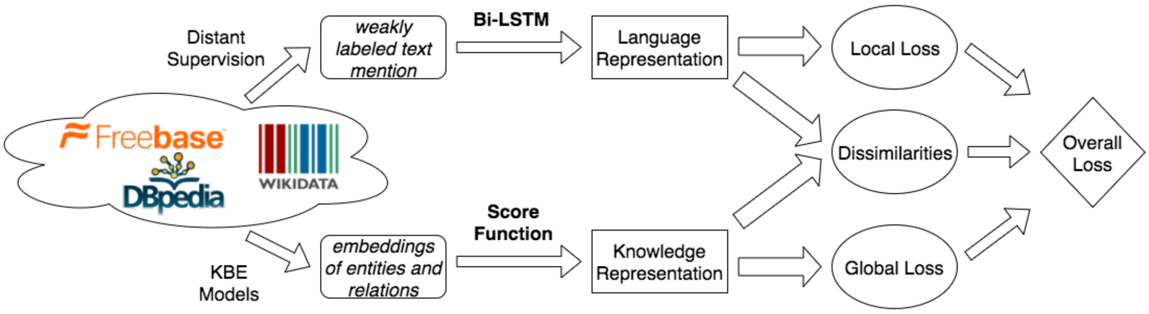


Figure 1: Workflow of the proposed framework.

representations for RE and KBE tasks that uses a cross-entropy loss function to ensure both representations are learned together, resulting in significant improvements over the current state-of-the-art for the RE task.

2 Related Work

Recent neural models have been shown superior to approaches using hand-crafted features for the RE task. Among the pioneers, Zeng et al. (2015) proposed a piecewise convolutional network with multi-instance learning to handle weakly labeled text mentions. Recurrent neural networks (RNN) are another popular architecture (Wu et al., 2017). Similar fast progress has been seen for the KBE task for representing entities and relations in KBs with vectors or matrices. Bordes et al. (2013) introduced the influential translation-based embeddings (TransE), while Yang et al. (2014) leveraged latent matrix factorization in their DistMult method. We build on ComplEx (Trouillon et al., 2016), which extends DistMult into the complex space and has been shown significantly better on several benchmarks.

Weston et al. (2013) were the first to connect RE and KBE models for the RE task. Their simple idea was to train the two models *independently* and only combine them at inference time. While they showed that combining the two models is better than using the RE model alone, newer and better models since then have obviated the net gains of such a simple strategy (Xu and Barbosa, 2018). We propose a much tighter integration of RE and KBE models: we not only use them for prediction, but also *train* them together, thus mutually reinforcing one another.

Recently, many methods have been proposed to use information from KBs to facilitate relation extraction. Sorokin and Gurevych (2017) considered other relations in the sentential context while predicting the target relation. Vashishth et al. (2018) utilized additional side information from KBs for improved RE. However, these methods didn't leverage KBE method to unify RE and KBE in a principled way. Han et al. (2018) used a mutual attention between KBs and text to perform better on both RE and KBE, but their method was still based on TransE (Bordes et al., 2013) which can not fully exploit the advantage of the information from KBs.

3 Background and Problem

The goal in the task of Relation Extraction is to predict a KB relation that holds for a pair of entities given a set of sentences mentioning them (or *NA* if no such relation exists). The input is a KB Ψ with relation set $\mathcal{R}_\Psi$, a set of relations of interest $\mathcal{R}$, $\mathcal{R} \subseteq \mathcal{R}_\Psi$, and an automatically labelled training dataset $\mathcal{D}$ obtained via distant supervision. Given a sentence mentioning entities h, t, the output is a relation $r \in \mathcal{R}$ that holds for h, t or the catch-all relation *NA* if no such r exists.

Knowledge Base and Distant Supervision. As customary, we denote a KB Ψ with relation scheme $\mathcal{R}_\Psi$ as a set of *triples* $\mathcal{T}_\Psi = \{(h, r, t) \in \mathcal{E}_\Psi \times \mathcal{R}_\Psi \times \mathcal{E}_\Psi\}$, where $\mathcal{E}_\Psi$ is the set of entities of interest. Distant supervision exploits the KB to automatically annotate sentences in a corpus containing mentions of entities with the relations they participate in. Formally, a labeled dataset for relation extraction consists of fact triples $\{(h_i, r_i, t_i)\}_{i=1}^N$ and a multi-set of extracted sentences for each triple $\{\mathcal{S}_i\}_{i=1}^N$, such that each sentence $s \in \mathcal{S}_i$ mentions both the head entity h_i and the tail entity t_i.

Problem Statement. Given an entity pair (h, t) and a set of sentences $\mathcal{S}$ mentioning them, the RE task is to estimate the probability of each relation in $\mathcal{R} \cup \{NA\}$. Formally, for each relation r, we want to predict $P(r \mid h, t, \mathcal{S})$.

In practice, the input set of sentences $\mathcal{S}$ can have arbitrary size. For the sake of computational efficiency, we normalize the set size to a fixed number T by splitting large sets and oversampling small ones. We also restrict the length of each sentence in the set by a constant L by truncating long sentences and padding short ones.

4 Methodology

We now go over the details of our framework outlined in Figure 1 for unifying the learning of the language and the knowledge representations used for relation extraction. In a nutshell, we use LSTM with attention mechanisms for language representation and we follow the approach of Trouillon et al. (2016) for KB embedding.

4.1 Language Representation

Input Representation. For each word token, we use pretrained word embeddings and randomly initialized position embeddings (Zeng et al., 2014) to project it into $(d_w + d_p)$-dimensional space, where d_w is the size of word embedding and d_p is the size of position embedding.

Sentence Encoder. For each sentence s_i, we apply a non-linear transformation to the vector representation of s_i to derive a feature vector $z_i = f(s_i; \theta)$ given a set of parameters θ. In this paper, we adopt bidirectional LSTM with d_s hidden units as $f(s_i; \theta)$ (Zhou et al., 2016).

Multi-level Attention Mechanisms. We employ attention mechanisms at both word-level and sentence-level to allow the model to softly select the most informative words and sentences during training (Zhou et al., 2016; Lin et al., 2016). With the learned language representation $\mathbf{s}_L$, the conditional probability $p(r|\mathcal{S}; \Theta^{(L)})$ is computed through a *softmax* layer, where $\Theta^{(L)}$ is the parameters of the model to learn language representation.

4.2 Knowledge Representation

Following the score function ϕ and training procedure of Trouillon et al. (2016), we can get the knowledge representations $e_h, w_r, e_t \in \mathbb{C}^{d_k}$. With the knowledge representations and the scoring function, we can obtain the conditional probability $p(r|(h,t); \Theta^{(G)})$ for each relation r:

$$p(r|(h,t); \Theta^{(G)}) = \frac{e^{\phi(e_h, w_r, e_t)}}{\sum_{r' \in \mathcal{R} \cup \{NA\}} e^{\phi(e_h, w_{r'}, e_t)}}$$

where $\Theta^{(G)}$ corresponds to the knowledge representations $e_h, w_r, e_t \in \mathbb{C}^{d_k}$. Since $NA \notin \mathcal{R}_\Psi$, we use a randomized complex vector as w_{NA}.

4.3 Connecting the Pieces

As stated, this paper seeks an elegant way of connecting language and knowledge representations for the RE task. In order to achieve that, we use separate loss functions (recall Figure 1) to guide the language and knowledge representation learning and a third loss function that ties the predictions of these models thus nudging the parameters towards agreement.

The cross-entropy losses based on the language and knowledge representations are defined as:

$$\mathcal{J}_L = -\frac{1}{N} \sum_{i=1}^{N} \log p(r_i | \mathcal{S}_i; \Theta^{(L)}) \quad (1)$$

$$\mathcal{J}_G = -\frac{1}{N} \sum_{i=1}^{N} \log p(r_i | (h_i, t_i); \Theta^{(G)}) \quad (2)$$

where N denotes the size of the training set. Finally, we use a cross-entropy loss to measure the dissimilarity between two distributions, thus connecting them, and formulate model learning as minimizing $\mathcal{J}_D$:

$$\mathcal{J}_D = -\frac{1}{N} \sum_{i=1}^{N} \log p(r_i^* | \mathcal{S}_i; \Theta^{(L)}) \quad (3)$$

where $r_i^* = \arg\max_{r \in \mathcal{R} \cup \{NA\}} p(r|(h_i, t_i); \Theta^{(G)})$.

4.4 Model Learning

Based on Eq. 1, 2, 3, we form the joint optimization problem for model parameters as

$$\min_{\Theta} \mathcal{J} = \mathcal{J}_L + \mathcal{J}_G + \mathcal{J}_D + \lambda \|\Theta\|_2^2 \quad (4)$$

where $\Theta = \Theta^{(L)} \cup \Theta^{(G)}$. The knowledge representations are first trained on the whole KB independently and then used as the initialization for the *joint* learning. We adopt the stochastic gradient descent with mini-batches and Adam (Kingma and Ba, 2014) to update Θ, employing different learning rates lr_1 and lr_2 on $\Theta^{(L)}$ and $\Theta^{(G)}$ respectively

4.5 Relation Inference

In order to get the conditional probability $p(r|(h,t),\mathcal{S};\Theta)$, we use the weighed average to combine the two distribution $p(r|\mathcal{S};\Theta^{(L)})$ and $p(r|(h,t);\Theta^{(G)})$:

$$\begin{aligned} p(r|(h,t),\mathcal{S};\Theta) &= \alpha * p(r|\mathcal{S};\Theta^{(L)}) \\ &+(1-\alpha) * p(r|(h,t);\Theta^{(G)}). \end{aligned} \tag{5}$$

where α is the combining weight of the weighted average. Then, the predicted relation $\hat{r}$ is

$$\hat{r} = \underset{r \in \mathcal{R} \cup \{NA\}}{\operatorname{argmax}} \; p(r|(h,t),\mathcal{S};\Theta). \tag{6}$$

5 Experiments

Datasets. We evaluate our model on the widely used **NYT** dataset (Riedel et al., 2010) by aligning Freebase relations mentioned in the New York Times Corpus. Articles from years 2005-2006 are used for training while articles from 2007 are used for testing. As our KB, we used a Freebase subset with the 3M entities with highest degree (i.e., participating in most relations). Moreover, to prevent the knowledge representation from memorizing the true relations for entity pairs in the test set, we removed all entity pairs present in the NYT.

Evaluation Protocol: Following previous work (Mintz et al., 2009), we evaluate our model using held-out evaluation which approximately measures the precision without time-consuming manual evaluation. We report both Precision/Recall curves and Precision@N (P@N) in our experiments, ignoring the probability predicted for the *NA* relation. Moreover, to evaluate each sentence in the test set as in previous methods, we append T copies of each sentence into $\mathcal{S}$ for each testing sample.

Word Embeddings: In this paper, we used the freely available 300-dimensional pre-trained word embeddings distributed by Pennington et al. (2014) to help the model generalize to words not appearing in the training set.

Hyperparameter Settings: For hyperparameter tuning, we randonly sampled 10% of the training set as a development set. All the hyperparameters were obtained by evaluating the model on the development set. With the well-tuned hyperparameter setting, we run each model five times on the whole training set and report the average P@N. For Precision/Recall curves, we just select the results from the first run of each model. For training, we set the iteration number over all the training data as 30. Values of the hyperparameters used in the experiments can be found in Table 1.

learning rate on $\Theta^{(L)}$	lr_1	5×10^{-4}
learning rate on $\Theta^{(K)}$	lr_2	1×10^{-5}
size of word position embedding	d_p	25
state size for LSTM layers	d_s	320
input dropout keep probability	p_i	0.9
output dropout keep probability	p_o	0.7
L2 regularization parameter	λ	0.0003
combining weight parameter	α	0.6

Table 1: Hyperparameter setting

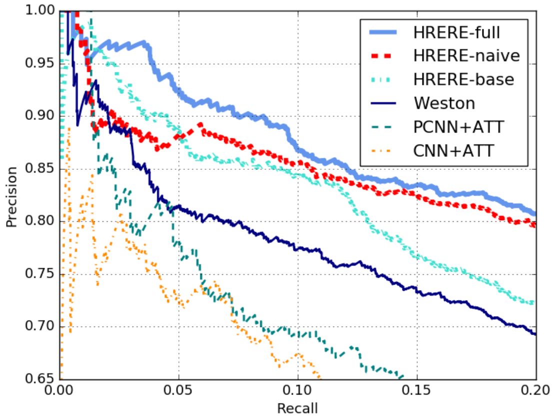


Figure 2: The Precision/Recall curves of previous state-of-the-art methods and our proposed framework.

P@N(%)	10%	30%	50%
Weston	79.3	68.6	60.9
HRERE-base	81.8	70.1	60.7
HRERE-naive	83.6	74.4	65.7
HRERE-full	**86.1**	**76.6**	**68.1**

Table 2: P@N of **Weston** and variants of our proposed framework.

Methods Evaluated. We study three variants of our framework: (1) HRERE-**base**: basic neural model with local loss $\mathcal{J}_L$ only; (2) HRERE-**naive**: neural model with both local loss $\mathcal{J}_L$ and global loss $\mathcal{J}_G$ but without the dissimilarities $\mathcal{J}_D$; (3) HRERE-**full**: neural model with both local and global loss along with their dissimilarities. We compare against two previous state-of-the-art neural models, **CNN+ATT** and **PCNN+ATT** (Lin et al., 2016). We also implement a baseline **Weston** based on the strategy following Weston et al. (2013), namely to combine the scores computed

Relation	Textual Mention	base	naive	full
contains	Much of the **middle east** tension stems from the sense that shiite power is growing, led by **Iran**.	0.311	0.864	**0.884**
place_of_birth	Sometimes I rattle off the names of movie stars from **Omaha**: Fred Astaire, **Henry Fonda**, Nick Nolte . . .	0.109	0.605	**0.646**
country	Spokesmen for **Germany** and Italy in Washington said yesterday that they would reserve comment until the report is formally released at a news conference in **Berlin** today.	0.237	0.200	**0.880**

Table 3: Some examples in NYT corpus and the predicted probabilities of the true relations.

with the methods stated in this paper directly without joint learning.

Analysis. Figure 2 shows the Precision/Recall curves for all the above methods. As one can see, HRERE-**base** significantly outperforms previous state-of-the-art neural models and **Weston** over the entire range of recall. However, HRERE-**base** performs worst compared to all other variants, while HRERE-**full** always performs best as shown in Figure 2 and Table 2. This suggests that introducing knowledge representation consistently results in improvements, which validates our motivating hypothesis. HRERE-**naive** simply optimizes both local and global loss at the same time without attempting to connect them. We can see that HRERE-**full** is not only consistently superior but also more stable than HRERE-**naive** when the recall is less than 0.1. One possible reason for the instability is that the results may be dominated by one of the representations and biased toward it. This suggests that (1) jointly learning the heterogeneous representations bring mutual benefits which are out of reach of previous methods that learn each independently; (2) connecting heterogeneous representations can increase the robustness of the framework.

Case Study. Table 3 shows two examples in the testing data. For each example, we show the relation, the sentence along with entity mentions and the corresponding probabilities predicted by HRERE-**base** and HRERE-**full**. The entity pairs in the sentence are highlighted with bold formatting.

From the table, we have the following observations: (1) The predicted probabilities of three variants of our model in the table match the observations and corroborate our analysis. (2) From the text of the two sentences, we can easily infer that *middle east contains Iran* and *Henry Fonda was born in Omaha*. However, HRERE-**base** fails to detect these relations, suggesting that it is hard for models based on language representations alone to detect implicit relations, which is reasonable to expect. With the help of KBE, the model can effectively identify implicit relations present in the text. (3) It may happen that the relation cannot be inferred by the text as shown in the last example. It's a common wrong labeled case caused by distant supervision. It is a case of an incorrectly labeled instance, a typical occurrence in distant supervision. However, the fact is obviously true in the KBs. As a result, HRERE-**full** gives the underlying relation according to the KBs. This observation may point to one direction of de-noising weakly labeled textual mentions generated by distant supervision.

6 Conclusion

This paper describes an elegant neural framework for jointly learning heterogeneous representations from text and from facts in an existing knowledge base. Contrary to previous work that learn the two disparate representations independently and use simple schemes to integrate predictions from each model, we introduce a novel framework using an elegant loss function that allows the proper connection between the the heterogeneous representations to be learned seamlessly during training. Experimental results demonstrate that the proposed framework outperforms previous strategies to combine heterogeneous representations and the state-of-the-art for the RE task. A closer inspection of our results show that our framework enables both independent models to enhance each other.

Acknowledgments

This work was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and a gift from Diffbot Inc.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. *arXiv preprint arXiv:1812.04361*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1779–1784.

Peng Xu and Denilson Barbosa. 2018. Investigations on knowledge base embedding for relation prediction and extraction. *arXiv preprint arXiv:1802.02114*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Emnlp*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Segmentation-free Compositional n-gram Embedding

Geewook Kim and **Kazuki Fukui** and **Hidetoshi Shimodaira**
Department of Systems Science, Graduate School of Informatics, Kyoto University
Mathematical Statistics Team, RIKEN Center for Advanced Intelligence Project
`{geewook, k.fukui}@sys.i.kyoto-u.ac.jp, shimo@i.kyoto-u.ac.jp`

Abstract

We propose a new type of representation learning method that models words, phrases and sentences seamlessly. Our method does not depend on word segmentation and any human-annotated resources (e.g., word dictionaries), yet it is very effective for noisy corpora written in unsegmented languages such as Chinese and Japanese. The main idea of our method is to ignore word boundaries completely (i.e., *segmentation-free*), and construct representations for all character n-grams in a raw corpus with embeddings of *compositional* sub-n-grams. Although the idea is simple, our experiments on various benchmarks and real-world datasets show the efficacy of our proposal.

1 Introduction

Most existing word embedding models (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017) take a sequence of words as their input. Therefore, the conventional models are dependent on word segmentation (Yang et al., 2017; Shao et al., 2018), which is a process of converting a raw corpus (i.e., a sequence of characters) into a sequence of segmented character n-grams. After the segmentation, the segmented character n-grams are assumed to be words, and each word's representation is constructed from distribution of neighbour words that co-occur together across the estimated word boundaries. However, in practice, this kind of approach has several problems. First, word segmentation is difficult especially when texts in a corpus are noisy or unsegmented (Saito et al., 2014; Kim et al., 2018). For example, word segmentation on social network service (SNS) corpora, such as Twitter, is a challenging task since it tends to include many misspellings, informal words, neologisms, and even emoticons. This problem becomes more severe in unsegmented languages, such as Chinese and Japanese, whose word boundaries are not explicitly indicated. Second, word segmentation has ambiguities (Luo et al., 2002; Li et al., 2003). For example, a compound word 線形代数学 (linear algebra) can be seen as a single word or sequence of words, such as 線形|代数学 (linear | algebra).

Word segmentation errors negatively influence subsequent processes (Xu et al., 2004). For example, we may lose some words in training corpora, leading to a larger Out-Of-Vocabulary (OOV) rate (Sun et al., 2005). Moreover, segmentation errors, such as segmenting きのう (yesterday) as き|のう (tree | brain), produce false co-occurrence information. This problem is crucial for most existing word embedding methods as they are based on distributional hypothesis (Harris, 1954), which can be summarized as: "*a word is characterized by the company it keeps*" (Firth, 1957).

To enhance word segmentation, some recent works (Junyi, 2013; Sato, 2015; Jeon, 2016) made rich resources publicly available. However, maintaining them up-to-date is difficult and it is infeasible for them to cover all types of words. To avoid the negative impacts of word segmentation errors, Oshikiri (2017) proposed a word embedding method called *segmentation-free word embedding* (`sembei`). The key idea of `sembei` is to directly embed frequent character n-grams from a raw corpus without conducting word segmentation. However, most of the frequent n-grams are non-words (Kim et al., 2018), and hence `sembei` still suffers from the OOV problems. The fundamental problem also lies in its extension (Kim et al., 2018), although it uses external resources to reduce the number of OOV. To handle OOV problems, Bojanowski et al. (2017) proposed a novel *compositional* word embedding method with subword modeling, called *subword-information skip-gram* (`sisg`). The key idea of `sisg` is to extend the notion of vocabulary to include subwords,

Proceedings of NAACL-HLT 2019, pages 3207–3215
Minneapolis, Minnesota, June 2 - June 7, 2019.

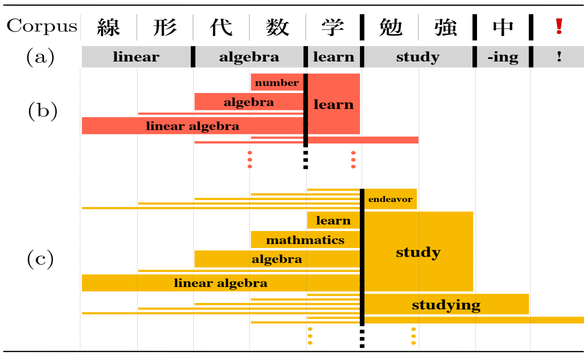


Figure 1: A Japanese tweet with manual segmentation. (a) is the segmentation result of a widely-used word segmenter which conventional word embedding methods are dependent on. (b) and (c) show the embedding targets and their co-occurrence information to be considered in our proposed method `scne` on the boundaries of 数|学 and 学|勉. Unlike conventional word embedding methods, `scne` considers all possible character n-grams on all boundaries (e.g., 線|形, 形|代, 代|数, $\cdots$) in the raw corpus without segmentation.

namely, substrings of words, for enriching the representations of words by the embeddings of its subwords. In `sisg`, the embeddings of OOV (or unseen) words are computed from the embedings of their subwords. However, `sisg` requires word segmentation as a prepossessing step, and the way of collecting co-occurrence information is dependent on the results of explicit word segmentation.

For solving the issues of word segmentation and OOV, we propose a simple but effective unsupervised representation learning method for words, phrases and sentences, called *segmentation-free compositional n-gram embedding* (`scne`). The key idea of `scne` is to train embeddings of character n-grams to compose representations of all character n-grams in a raw corpus, and it enables treating all words, phrases and sentences seamlessly (see Figure 1 for an illustrative explanation). Our experimental results on a range of datasets suggest that `scne` can compute high-quality representations for words and sentences although it does not consider any word boundaries and is not dependent on any human annotated resources.

2 Segmentation-free Compositional n-gram Embedding (`scne`)

Our method `scne` successfully combines a subword model (Zhang et al., 2015; Wieting et al., 2016; Bojanowski et al., 2017; Zhao et al., 2018) with an idea of character n-gram embedding (Oshikiri, 2017; Kim et al., 2018). In `scne`, the vector representation of a target character n-gram is defined as follows. Let $x_1 x_2 \cdots x_N$ be a raw unsegmented corpus of N characters. For a range $i, i+1, \ldots, j$ specified by index $t = (i,j)$, $1 \leq i \leq j \leq N$, we denote the substring $x_i x_{i+1} \cdots x_j$ as $\boldsymbol{x}_{(i,j)}$ or $\boldsymbol{x}_t$. In a training phase, `scne` first counts frequency of character n-grams in the raw corpus to construct n-gram set V by collecting M-most frequent n-grams with $n \leq n_{\max}$, where M and $n_{\max}$ are hyperparameters. For any target character n-gram $\boldsymbol{x}_{(i,j)} = x_i x_{i+1} \cdots x_j$ in the corpus, `scne` constructs its representation $v_{\boldsymbol{x}_{(i,j)}} \in \mathbb{R}^d$ by summing the embeddings of its sub-n-grams as follows:

$$v_{\boldsymbol{x}_{(i,j)}} = \sum_{s \in S(\boldsymbol{x}_{(i,j)})} z_s,$$

where $S(\boldsymbol{x}_{(i,j)}) = \{\boldsymbol{x}_{(i',j')} \in V \mid i \leq i' \leq j' \leq j\}$ consists of all sub-n-grams of target $\boldsymbol{x}_{(i,j)}$, and the embeddings of sub-n-grams $z_s \in \mathbb{R}^d$, $s \in V$ are model parameters to be learned. The objective of `scne` is similar to that of Mikolov et al. (2013),

$$\sum_{t \in \mathcal{D}} \left\{ \sum_{c \in \mathcal{C}(t)} \log \sigma\left(v_{\boldsymbol{x}_t}^\top u_{\boldsymbol{x}_c}\right) + \sum_{\tilde{s} \sim P_{\text{neg}}}^{k} \log \sigma\left(-v_{\boldsymbol{x}_t}^\top u_{\tilde{s}}\right) \right\},$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$, $\mathcal{D} = \{(i,j) \mid 1 \leq i \leq j \leq N, j-i+1 \leq n_{\text{target}}\}$, and $\mathcal{C}((i,j)) = \{(i',j') \mid \boldsymbol{x}_{(i',j')} \in V, j' = i-1 \text{ or } i' = j+1\}$. $\mathcal{D}$ is the set of indexes of all possible target n-grams in the raw corpus with $n \leq n_{\text{target}}$, where n_{target} is a hyperparameter. $\mathcal{C}(t)$ is the set of indexes of contexts of the target $\boldsymbol{x}_t$, that is, all character n-grams in V that are adjacent to the target (see Figures 1 and 2). The negative sampling distribution P_{neg} of $\tilde{s} \in V$ is proportional to its frequency in the corpus. The model parameters $z_s, u_{\tilde{s}} \in \mathbb{R}^d$, $s, \tilde{s} \in V$, are learned by maximizing the objective. We set $n_{\text{target}} = n_{\max}$ in our experiments.

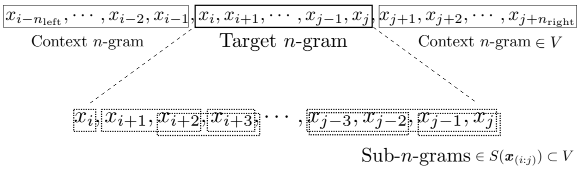


Figure 2: A graphical illustration of the proposed model trying to compute a representation for a character n-gram $\boldsymbol{x}_{(i,j)}$. The co-occurrence of $\boldsymbol{x}_{(i,j)}$ and its neighbouring context n-grams are used to train embeddings of *compositional* n-grams.

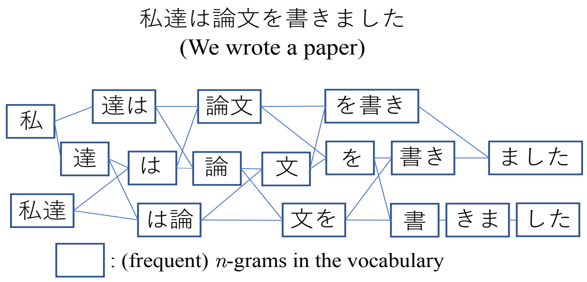


Figure 3: An example of a frequent n-gram lattice.

Although we examine frequent n-grams for simplicity, incorporating supervised word boundary information or *byte pair encoding* into the construction of *compositional* n-gram set would be an interesting future work (Kim et al., 2018; Sennrich et al., 2016; Heinzerling and Strube, 2018).

2.1 Comparison to Oshikiri (2017)

To avoid the problems of word segmentation, Oshikiri (2017) proposed *segmentation-free word embedding* (`sembei`) (Oshikiri, 2017) that considers the M-most frequent character n-grams as individual words. Then, a *frequent n-gram lattice* is constructed, which is similar to a word lattice used in morphological analysis (see Figure 3). Finally, the pairs of adjacent n-grams in the lattice are considered as target-context pairs and they are fed to existing word embedding methods, e.g., `skipgram` (Mikolov et al., 2013). Although `sembei` is simple, the frequent n-gram vocabulary tends to include a vast amount of non-words (Kim et al., 2018). Furthermore, its vocabulary size is limited to M, hence, `sembei` can not avoid the undesirable issue of OOV. The proposed `scne` avoids these problems by taking all possible character n-grams as embedding targets. Note that the target-context pairs of `sembei` are fully contained in those of `scne` (see Figure 1).

2.2 Comparison to Kim et al. (2018)

To overcome the problem of OOV in `sembei`, Kim et al. (2018) proposed an extension of `sembei` called *word-like n-gram embedding* (`wne`). In `wne`, the n-gram vocabulary is filtered to have more vaild words by taking advantage of a supervised probabilistic word segmenter. Although `wne` reduce the number of non-words, there is still the problem of OOV since its vocabulary size is limited. In addition, `wne` is dependent on word segmenter while `scne` does not.

2.3 Comparison to Bojanowski et al. (2017)

To deal with OOV words as well as rare words, Bojanowski et al. (2017) proposed *subword information skip-gram* (`sisg`) that enriches word embeddings with the representations of its subwords, i.e., sub-character n-grams of words. In `sisg`, a vector representation of a target word is encoded as the sum of the embeddings of its subwords. For instance, subwords of length $n = 3$ of the word *where* are extracted as `<wh, whe, her, ere, re>`, where "<","\>" are special symbols added to the original word to represent its left and right word boundaries. Then, a vector representation of *where* is encoded as the sum of the embeddings of these subwords and that of the special sequence `<where>`, which corresponds to the original word itself. Although `sisg` is powerful, it requires the information of word boundaries as its input, that is, semantic units need to be specified when encoding targets. Therefore, it cannot be directly applied to unsegmented languages. Unlike `sisg`, `scne` does not require such information. The proposed `scne` is much simpler, but due to its simpleness, the embedding target of `scne` should contains many non-words, which seems to be a problem (see Figure 1). However, our experimental results show that `scne` successfully captures the semantics of words and even sentences for unsegmented languages without using any knowledge of word boundaries (see Section 3).

3 Experiments

In this section, we perform two intrinsic and two extrinsic tasks at both word and sentence level, focusing on unsegmented languages. The implementation of our method is available on GitHub[1].

3.1 Common Settings

Baselines: We use `skipgram` (Mikolov et al., 2013), `sisg` (Bojanowski et al., 2017) and `sembei` (Oshikiri, 2017) as word embedding baselines. For sentence embedding, we first test simple baselines obtained by averaging the word vectors over a word-segmented sentence. In addition, we examine several recent successful sentence embedding methods, `pv-dbow`, `pv-dm` (Le and Mikolov, 2014) and `sent2vec` (Pagliardini et al., 2018) in an extrinsic task. Note that both `scne` and `sembei` have embeddings of frequent character n-grams as their model parameters, but

---

[1] `www.github.com/kdrl/SCNE`

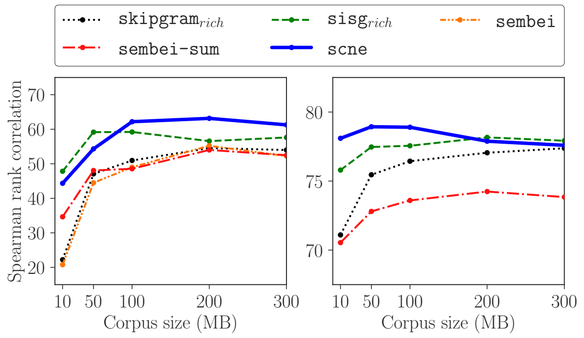


Figure 4: Word (left) and sentence (right) similarity tasks on portions of Chinese Wikipedia corpus.

the differences come from training strategies, such as embedding targets and the way of collecting co-occurrence information (see Section 2.1 for more details). For contrasting scne with sembei, we also propose a variant of sembei (denoted by sembei-sum) as one of baselines, which composes word and sentence embeddings by simply summing up the embeddings of their sub-n-grams which are learned by sembei.

Hyperparameters Tuning: To see the effect of rich resources for the segmentation-dependent baselines, we employ widely-used word segmenter with two settings: Using only a basic dictionary (*basic*) or using a rich dictionary together (*rich*). The dimension of embeddings is 200, the number of epochs is 10 and the number of negative samples is 10 for all the methods. The n-gram vocabulary size $M = 2 \times 10^6$ is used for sisg, sembei and scne. The other hyperparameters, such as learning rate and $n_{\max}$, are carefully adjusted via a grid search in the validation set. In the word similarity task, 2-fold cross validation is used for evaluation. In the sentence similarity task, we use the provided validation set. In the downstream tasks, vector representations are combined with a supervised logistic regression classifier. We repeat training and testing of the classifier 10 times, while the prepared dataset is randomly split into train (60%) and test (40%) sets at each time, and the hyperparameters are tuned by 3-fold cross validation in the train set. We adopt mean accuracy as the evaluation metric. See Appendix A.1 for more experimental details.

3.2 Word and Sentence Similarity

We measure the ability of models to capture semantic similarity for words and sentences in Chinese; see Appendix A.2 for the experiment in Japanese. Given a set of word pairs, or sentence pairs, and their human annotated similarity scores, we calculated Spearman's rank correlation between the cosine similarities of the embeddings and the scores. We use the dataset of Jin and Wu (2012) and Wang et al. (2017) for Chinese word and sentence similarity respectively. Note that the conventional models, such as skipgram, cannot provide the embeddings for OOV words, while the *compositional* models, such as sisg and scne, can compute the embeddings by using their sub-word modeling. In order to show comparable results, we use the null vector for these OOV words following Bojanowski et al. (2017).

	skipgram$_{rich}$	sisg$_{rich}$	sembei	sembei-sum	**scne**
Wiki.	51.0	59.2	49.0	48.6	**62.2**
SNS	41.3	47.0	38.9	41.5	**60.0**
Diff.	-9.7	-12.2	-10.1	-7.1	-2.2

Table 1: Spearman rank correlations of the word similarity task on two different Chinese corpora. Best scores are boldface and 2nd best scores are underlined.

Results: To see the effect of training corpus size, we train all models on portions of Wikipedia[2]. The results are shown in Figure 4. As it can be seen, the proposed scne is competitive with or outperforms the baselines for both word and sentence similarity tasks. Moreover, it is worth noting that scne provides high-quality representations even when the size of training corpus is small, which is crucial for practical real-world settings where rich data is not available. For a next experiment to see the effect of noisiness of training corpus, we test both noisy SNS corpus and the Wikipedia corpus[3] of the same size. The results are reported in Table 1. As it can be seen, the performance of segmentation-dependent methods (skipgram, sisg) are decreased greatly by the noisiness of the corpus, while scne degrades only marginally. The other two segmentation-free methods (sembei, sembei-sum) performed poorly. This shows the efficacy of our method in the noisy texts. On the other hand, in preliminary experiments on English (not shown), scne did not get better results than our segmentation-dependent baselines and it will be a future work to incorporate easily obtainable word boundary information into scne for segmented languages.

---

[2] We use 10, 50, 100, 300MB of Wikipedia from the head.

[3] We use 100MB of Sina Weibo posts for Chinese SNS corpus and 100MB of Chinese Wikipedia corpus.

	Wikipedia corpora						Noisy SNS corpora					
	Chinese		Japanese		Korean		Chinese		Japanese		Korean	
	All	Intersec.	**All**	Intersec.	**All**	Intersec.	**All**	Intersec.	**All**	Intersec.	**All**	Intersec.
`skipgram`$_{basic}$	8.9 (11)	81.0	7.8 (10)	75.7	11.5 (15)	**77.1**	2.9 (5)	58.2	2.9 (7)	41.4	2.4 (7)	35.4
`skipgram`$_{rich}$	9.5 (12)	81.0	16.7 (20)	**75.8**	11.9 (15)	76.9	3.0 (5)	58.2	4.1 (9)	40.9	2.5 (7)	34.2
`sisg`$_{basic}$	79.2 (100)	**82.3**	72.2 (100)	75.7	72.2 (100)	76.2	71.0 (100)	64.8	67.1 (100)	46.9	63.4 (100)	**39.8**
`sisg`$_{rich}$	79.5 (100)	82.2	73.3 (100)	74.7	72.4 (100)	76.6	70.8 (100)	64.9	67.5 (100)	46.0	63.3 (100)	37.7
`sembei`	21.8 (25)	79.0	18.2 (23)	70.1	14.2 (19)	41.8	4.5 (7)	59.6	4.9 (10)	41.9	5.0 (13)	33.7
`sembei-sum`	76.8 (100)	74.2	69.9 (100)	61.3	66.3 (100)	56.0	72.3 (100)	56.4	66.3 (100)	40.7	64.3 (100)	34.8
`scne` (Proposed)	**79.8 (100)**	81.5	**73.9 (100)**	74.0	**73.2 (100)**	73.9	**74.9 (100)**	**65.0**	**68.1 (100)**	**47.6**	**65.3 (100)**	38.2

Table 2: Noun category prediction accuracies (higher is better) and coverages [%] (in parentheses, higher is better).

	Segmentation-free	Chinese	Japanese	Korean
`pv-dbow`$_{basic}$		82.84	85.24	84.16
`pv-dbow`$_{rich}$		83.47	85.55	84.80
`pv-dm`$_{basic}$		76.96	80.67	66.35
`pv-dm`$_{rich}$		77.94	81.37	67.32
`sent2vec`$_{basic}$		85.09	87.12	82.31
`sent2vec`$_{rich}$		85.39	87.20	82.34
`skipgram`$_{basic}$		85.79	86.76	84.06
`skipgram`$_{rich}$		85.77	87.16	84.48
`sisg`$_{basic}$		85.67	87.22	84.34
`sisg`$_{rich}$		85.04	87.25	84.35
`sembei-sum`	✓	83.41	80.80	74.98
`scne`$_{n_{\max}=8}$	✓	87.07	87.42	84.15
`scne`$_{n_{\max}=16}$	✓	**87.76**	**88.03**	**86.74**

Table 3: Sentiment classification accuracies [%].

3.3 Noun Category Prediction

As a word-level downstream task, we conduct a noun category prediction on Chinese, Japanese and Korean[4]. Most settings are the same as those of Oshikiri (2017). Noun words and their semantic categories are extracted from Wikidata (Vrandečić and Krötzsch, 2014) with a predetermined semantic category set[5], and the classifier is trained to predict the semantic category of words from the learned word representations, where unseen words are skipped in training and treated as errors in testing. To see the effect of the noisiness of corpora, both noisy SNS corpus and Wikipedia corpus of the same size are examined as training corpora[6].

Results: The results are reported in Table 2. Since the set of covered nouns (i.e., non-OOV words) depends on the methods, we calculate accuracies in two ways for a fair comparison: Using all the nouns and using the intersection of the covered nouns. `scne` achieved the highest accuracies in all the settings when using all the nouns, and also performed well when using the intersection of the covered nouns, especially for the noisy corpora.

3.4 Sentiment Analysis

As a sentence-level evaluation, we perform sentiment analysis on movie review data. We use 101k, 56k and 200k movie reviews and their scores respectively from Chinese, Japanese and Korean movie review websites (see Appendix A.1.6 for more details). Each review is labeled as positive or negative by its rating score. Sentence embedding models are trained using the whole movie reviews as training corpus. Among the reviews, 5k positive and 5k negative reviews are randomly selected, and the selected reviews are used to train and test the classifiers as explained in Section 3.1.

Results: The results are reported in Table 3. The accuracies show that `scne` is also very effective in the sentence-level application. In this experiment, we observe that the larger $n_{\max}$ contributes to the performance improvement in sentence-level application by allowing our model to capture composed representations for longer phrases or sentences.

4 Conclusion

We proposed a simple yet effective unsupervised method to acquire general-purpose vector representations of words, phrases and sentences seamlessly, which is especially useful for languages whose word boundaries are not obvious, i.e., unsegmented languages. Although our method does not rely on any manually annotated resources or word segmenter, our extensive experiments show that our method outperforms the conventional approaches that depend on such resources.

Acknowledgments

We would like to thank anonymous reviewers for their helpful advice. This work was partially supported by JSPS KAKENHI grant 16H02789 to HS and 18J15053 to KF.

---

[4] Although Korean has spacing, word boundaries are not obviously determined by space.

[5] {food, song, music band name, manga, fictional character name, television series, drama, chemical compound, disease, taxon, city, island, country, year, business enterprise, public company, profession, university, language, book}

[6] For each language, we use 100MB of Wikipedia and SNS data as training corpora. For the SNS data, we use Sina Weibo for Chinese and Twitter for the rest.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146.

Daan van Esch. 2012. Leidon Weibo Corpus.

J.R. Firth. 1957. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis*, pages 1–32.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, pages 2989–2993. European Language Resource Association.

Heewon Jeon. 2016. NIA (National Information Society Agency) Dictionary.

Peng Jin and Yunfang Wu. 2012. SemEval-2012 Task 4: Evaluating Chinese Word Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 374–377. Association for Computational Linguistics.

Sun Junyi. 2013. Jieba.

Geewook Kim, Kazuki Fukui, and Hidetoshi Shimodaira. 2018. Word-like character n-gram embedding. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 148–152. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196. PMLR.

Mu Li, Jianfeng Gao, Changning Huang, and Jianfeng Li. 2003. Unsupervised Training for Overlapping Ambiguity Resolution in Chinese Word Segmentation. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing - Volume 17*, SIGHAN '03, pages 1–7. Association for Computational Linguistics.

Xiao Luo, Maosong Sun, and Benjamin K. Tsou. 2002. Covering Ambiguity Resolution in Chinese Word Segmentation Based on Contextual Information. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Takamasa Oshikiri. 2017. Segmentation-Free Word Embedding for Unsegmented Languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 767–772. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, and Yoshihiro Matsuo. 2014. Morphological Analysis for Japanese Noisy Text based on Character-level and Word-level Normalization. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1773–1782. Dublin City University and Association for Computational Linguistics.

Yuya Sakaizawa and Mamoru Komachi. 2018. Construction of a Japanese Word Similarity Dataset. In *Proceedings of the 11th Language Resources and Evaluation Conference*, pages 948–951. European Language Resource Association.

Toshinori Sato. 2015. Neologism dictionary based on the language resources on the Web for Mecab.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2018. Universal Word Segmentation: Implementation and Interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435.

Chengjie Sun, Chang-Ning Huang, Xiaolong Wang, and Mu Li. 2005. Detecting Segmentation Errors in Chinese Annotated Corpus. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 1–8. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledge Base. *Communications of the ACM*, 57:78–85.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2017. Exploiting Word Internal Structures for Generic Chinese Sentence Representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 298–303. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515. Association for Computational Linguistics.

Jia Xu, Richard Zens, and Hermann Ney. 2004. Do We Need Chinese Word Segmentation for Statistical Machine Translation? In *ACL SIGHAN Workshop 2004*, pages 122–128. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural Word Segmentation with Rich Pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing Word Embeddings using Bag of Subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 601–606. Association for Computational Linguistics.

A Appendices

A.1 Experimental Details

A.1.1 Hyperparameters Tuning

For `skipgram`, we performed a grid search over $(h, \gamma) \in \{1, 5, 10\} \times \{0.01, 0.025\}$, where h is the size of context window and γ is the initial learning rate. For `sisg`, we performed a grid search over $(h, \gamma, n_{\text{min}}, n_{\text{max}}) \in \{1, 5, 10\} \times \{0.01, 0.025\} \times \{1, 3\} \times \{4, 8, 12\}$, where h is the size of context window, γ is the initial learning rate, n_{min} is the minimum length of character n-gram and n_{max} is the maximum length of character n-gram. For `pv-dbow`, `pv-dm` and `sent2vec`, we performed a grid search over $(h, \gamma) \in \{5, 10\} \times \{0.01, 0.05, 0.1, 0.2, 0.5\}$, where h is the size of context window and γ is the initial learning rate. For `sembei` and `scne`, we used the initial learning rate 0.01 and $n_{\text{min}} = 1$. The maximum length of n-gram to consider n_{max} is grid searched over $\{4, 6, 8\}$ in the word and sentence similarity tasks. In the noun category prediction task, we used $n_{\text{max}} = 8$ for `sembei` and the n_{max} of `scne` is grid searched over $\{4, 6, 8\}$. For sentiment analysis task, we tested both $n_{\text{max}} = 8$ and $n_{\text{max}} = 16$ for `sembei` and `scne` to see the effect of large n_{max}. After carefully monitoring the loss curve and the performance in the word and sentence similarity tasks, we set the number of epochs 10 for all methods. In preliminary experiments, we also tested the number of epochs 20 for the word-segmentation-dependent baselines but there were no significant differences. In the two supervised downstream tasks, the learned vector representations are combined with the logistic regression classifier. The parameter C, which is the inverse of regularization strength of the classifier, is adjusted via a grid search over $C \in \{0.1, 0.5, 1, 5, 10\}$. Again, as explained in the main paper, the hyperparamters are grid searched on the determined validation set for all experiments.

A.1.2 Implementations

Here we provide the list of implementations of baselines which are used in our experiments. For `skipgram`[7], `sisg`[8], `sembei`[9], and `sent2vec`[10], we use the official implementations provided by the authors. Meanwhile, as for `pv-dbow` and `pv-dm`, we employ a widely-used implementation of Gensim library[11].

A.1.3 Word Segmenters and Word Dictionaries for Unsegmented Languages

Below we list the word segmentation tools and word dictionaries which are used in our experiments. We employed a widely-used word segmentation tool for each language.

For Chinese language, we used jieba[12] with its

---

[7] `https://code.google.com/archive/p/word2vec/`

[8] `https://github.com/facebookresearch/fastText`

[9] `https://github.com/oshikiri/w2v-sembei`

[10] `https://github.com/epfml/sent2vec`

[11] `https://radimrehurek.com/gensim/models/doc2vec.html`

[12] `https://github.com/fxsjy/jieba`

default dictionary[13] or with an extended dictionary[14], which fully supports both traditional and simplified Chinese characters.

For Japanese, we used MeCab[15] with its default dictionary called IPADIC[15] along with specially designed neologisms-extended dictionary called mecab-ipadic-NEologd[16]. Note that, because this extended dictionary *mecab-ipadic-NEologd* is specially designed to include many neologisms, there is a significant word coverage improvement by using this word dictionary as it can be seen in the Japanese noun category prediction task in the main paper.

For Korean, we used mecab-ko[17] with its default dictionary called mecab-ko-dic[18] along with another extended dictionary called NIADic[19].

A.1.4 Training Corpora

We prepared Wikipedia corpora and SNS corpora for Chinese, Japanese and Korean for our experiments. For the Wikipedia corpora, we used the first 10, 50, 100, 200 and 300MB of texts from the publicly available Wikipedia dumps[20]. The texts are extracted by using WikiExtractor tool[21]. For Chinese SNS corpus, we used 100MB of Leiden Weibo Corpus (van Esch, 2012) from the head. For Japanese and Korean SNS corpora, we collected Japanese and Korean tweets using Twitter Streaming API. We removed usernames and URLs from the SNS corpora. There were many informal words, emoticons and misspellings in the SNS corpora. We preserved them without preprocessing to see the effect of the noisiness of training corpora in our experiments.

A.1.5 Preprocess of Wikidata

For the noun category prediction task, we extracted noun words and their semantic categories from Wikidata (Vrandečić and Krötzsch, 2014) following Oshikiri (2017). We determined the semantic category set used in our experiments as follows: First, we collected Wikidata objects that have Chinese, Japanese, Korean and English labels. Next, we sorted the categories by the number of noun words, and removed categories (e.g., *Wikimedia category* or *Wikimedia template*) that do not represent any semantic category. We also removed out several categories that contain too many noun words (e.g., *human*) or too few noun words (e.g., *academic discipline*). Since there were several duplicated labels for different Wikidata objects, the number of nouns for each language is slightly different. Each category has at least 0.1k words and no more than 5k words. The numbers of extracted noun words that are used in our experiments were 22,468, 22,396 and 22,298 for Chinese, Japanese and Korean, respectively.

	skipgram$_{rich}$	sisg$_{rich}$	sembei	sembei-sum	**scne**
Wiki.	8.3	15.4	4.0	9.3	**24.1**
SNS	5.3	12.7	2.8	9.3	**23.0**
Diff.	-3.0	-2.7	-1.2	-0.0	-1.1

Table 4: Spearman rank correlations of the word similarity task on two different Japanese corpora.

A.1.6 Movie Review Datasets

In the main paper, three movie review datasets are used to evaluate the quality of sentence embeddings. We used 101,114, 55,837 and 200,000 movie reviews and their rating scores from Yahoo奇摩電影[22], Yahoo!映画[23] and Naver Movies[24] for Chinese, Japanese and Korean, respectively.

A.2 Additional Experiment on Japanese

In this section, we show the results of Japanese word similarity experiments. We use the datasets of Sakaizawa and Komachi (2018). It contains 4427 pairs of words with human similarity scores. We omit sentence similarity task since there is no public widely-used benchmark dataset for Japanese yet. Following the main paper, given a set of word pairs and their human annotated similarity scores, we calculated Spearman's rank correlation between the cosine similarities of the embeddings and the human scores. We use 2-fold

---

[13] `https://github.com/fxsjy/jieba/blob/master/jieba/dict.txt`

[14] `https://github.com/fxsjy/jieba/blob/master/extra_dict/dict.txt.big`

[15] `http://taku910.github.io/mecab/`

[16] `https://github.com/neologd/mecab-ipadic-neologd`

[17] `https://bitbucket.org/eunjeon/mecab-ko`

[18] `https://bitbucket.org/eunjeon/mecab-ko-dic`

[19] `https://github.com/haven-jeon/NIADic`

[20] `https://dumps.wikimedia.org/`

[21] `https://github.com/attardi/wikiextractor`

[22] `https://github.com/fychao/ChineseMovieReviews`

[23] `https://github.com/dennybritz/sentiment-analysis/tree/master/data`

[24] `https://github.com/e9t/nsmc`

cross validation for hyperparameters tuning. The same grid search is performed as explained in Section A.1.1. To see the effect of the noisiness of training corpora, we use two Japanese corpora, 100MB of Wikipedia corpus and 100MB of noisy SNS corpus (Twitter), which are also used in the Japanese noun category prediction task in the main paper. As seen in Table 4, the experiment results for Japanese are similar to those of Chinese in the main paper.

Exploiting Noisy Data in Distant Supervision Relation Classification

Kaijia Yang, Liang He, Xin-yu Dai, Shujian Huang, Jiajun Chen
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, 210023, China
{yangkj,heliang}@nlp.nju.edu.cn
{daixinyu,huangsj,chenjj}@nju.edu.cn

Abstract

Distant supervision has obtained great progress on relation classification task. However, it still suffers from noisy labeling problem. Different from previous works that underutilize noisy data which inherently characterize the property of classification, in this paper, we propose RCEND, a novel framework to enhance Relation Classification by Exploiting Noisy Data. First, an instance discriminator with reinforcement learning is designed to split the noisy data into correctly labeled data and incorrectly labeled data. Second, we learn a robust relation classifier in semi-supervised learning way, whereby the correctly and incorrectly labeled data are treated as labeled and unlabeled data respectively. The experimental results show that our method outperforms the state-of-the-art models.

1 Introduction

Relation classification plays a crucial role in natural language processing (NLP) tasks, such as question answering and knowledge base completion (Xu et al., 2016; Han et al., 2018a). The goal of relation classification is to predict relations of the target entity pair given a plain text. Traditional supervised learning methods (Zelenko et al., 2002; Bunescu and Mooney, 2005; Zhou et al., 2005) heavily rely on large scale annotated data which is time and labor consuming. Mintz et al. (2009) proposed distant supervision (DS) to automatically generate training data for relation classification based on the assumption that if two target entities have a relation in knowledge base (KB), sentences containing this entity pair might express the relation. For example, if a relational fact <Apple, *founder*, Steve Jobs> exists in KB, distant supervision will assign *founder* as the label of all sentences that contain "Apple" and "Steve Jobs" together.

Sentence	DS	Gold
S1:**Al Gore** was waiting to board a commercial flight from **Nashville** to Miami...	LivedIn	NA
S2:There were also performers who were born in **Louisiana** , including **Lucinda Williams**...	LivedIn	BornIn
S3:**Boggs** was married, had three young children and lived in **Brewster**	NA	LivedIn

Table 1: Examples of noisy labeling problem in distant supervision relation classification. S1 and S2 are heuristically labeled as *LivedIn* by DS, but neither of them mention the relation while S2 mentions the *BornIn* relation. S3 expresses the *LivedIn* relation but it is mislabeled as *NA* since no relation of the entity pair exist in KB.

However, it suffers from noisy labeling problem due to the irrelevance of aligned text and incompleteness of KB, which consists of **false positives** and **false negatives**. The false positives means that not all sentences containing two entities mention the relation in KB, such as S1 and S2 in Table 1. And the false negatives are sentences are mislabeled as no relation (*NA*) due to the absence of relational fact in KB even though they express the target relation, such as S3 in Table 1.

In order to reduce the impact of noisy data, previous works (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016; Han et al., 2018b) adopt Multi-Instance Learning (MIL) for relation classification. Recent studies (Feng et al., 2018; Qin et al., 2018b,a) introduce reinforcement learning (RL) and adversarial learning to filter out incorrectly labeled sentences and achieve significant improvements. However, there are two remaining challenges of noisy labeling problem.

- Most of these approaches focus on solving the false positives but overlook false nega-

Proceedings of NAACL-HLT 2019, pages 3216–3225
Minneapolis, Minnesota, June 2 - June 7, 2019.

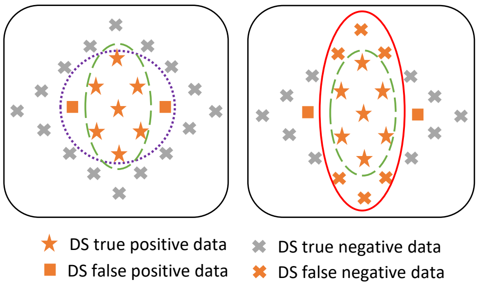


Figure 1: Illustration of false positive and false negative cases

tives. As illustrated in Figure 1, they concentrate on discovering the false positive instances[1] which are suppressed or removed at last and obtain a better decision boundary (green dashed line) than without consideration of false positive instances. Nevertheless, there are still a lot of false negative instances expressing similar semantic information with positive data. These instances also provide evidence for the target relation. The incorrect labels will weaken the discriminative capability of available features and confuse the model if they stay the same. However, when we remedy the label correctly, we indeed possess the optimal decision boundary (red solid line).

- There lacks an effective method to fully utilize noisy data of distant supervision. (Xu et al., 2013; Liu et al., 2017) apply methods such as pseudo-labels to directly correct the label of noisy data and Luo et al. (2017) design a dynamic transition matrix to model noise patterns. They still suffer from the drawback of error propagation during training.

To tackle the above challenges, we propose a novel framework exploiting noisy data to enhance distant supervision relation classification. We design an instance discriminator with reinforcement learning to recognize both false positive and false negative instances simultaneously, and further split the noisy dataset into two sets, representing correctly labeled and incorrectly labeled data respectively. Additionally, we learn a robust relation classifier applying a semi-supervised learning method, whereby the correctly and incorrectly labeled data are regarded as labeled and unlabeled data. On the one hand, we mitigate the side effect of incorrectly labeled data by recognizing them and treating them as unlabeled data. On the other hand, taking full advantage of the incorrectly labeled data in semi-supervised learning way facilitates robust property of model and improves generalization performance. Our contributions in this work are three-fold:

- We propose a deep reinforcement learning framework to discriminate both false-positive and false-negative instances simultaneously.
- We introduce a semi-supervised learning method to fully exploit the noisy data in distant supervision relation classification.
- We conduct experiments on a widely used benchmark dataset and the results show that our method achieves significant improvements as compared with strong baselines.

2 Related work

Many efforts based on supervised learning (Zelenko et al., 2002; Bunescu and Mooney, 2005; Zhou et al., 2005) have been devoted to relation classification. As is well-known, achieving a good performance while applying supervised learning paradigm requires a large amount of high-quality annotated data. To address the issue of data sparsity, Mintz et al. (2009) propose distant supervision to automatically annotate large scale training data, which inevitably results in noisy labeling problem.

To tolerate noisy instances in positive examples, most early approaches employ multi-instance learning framework, including multi-instance single-label learning (Riedel et al., 2010) and multi-instance multi-label learning (Hoffmann et al., 2011; Surdeanu et al., 2012). Recently, deep learning has also been introduced to propose an end-to-end convolutional neural network for relation classification (Zeng et al., 2014). In the sentences bag of one entity pair, Zeng et al. (2015) select the most reliable sentence, and Lin et al. (2016) propose attention schemes to de-emphasize unreliable sentences. Han et al. (2018b) incorporate hierarchical information of relations to enhance the attention scheme. But they fail to handle the issue where all sentences in one bag are mislabeled.

Feng et al. (2018); Qin et al. (2018b,a) further achieve improvement by using reinforcement

---

[1] In this paper, instance is the same as sentence

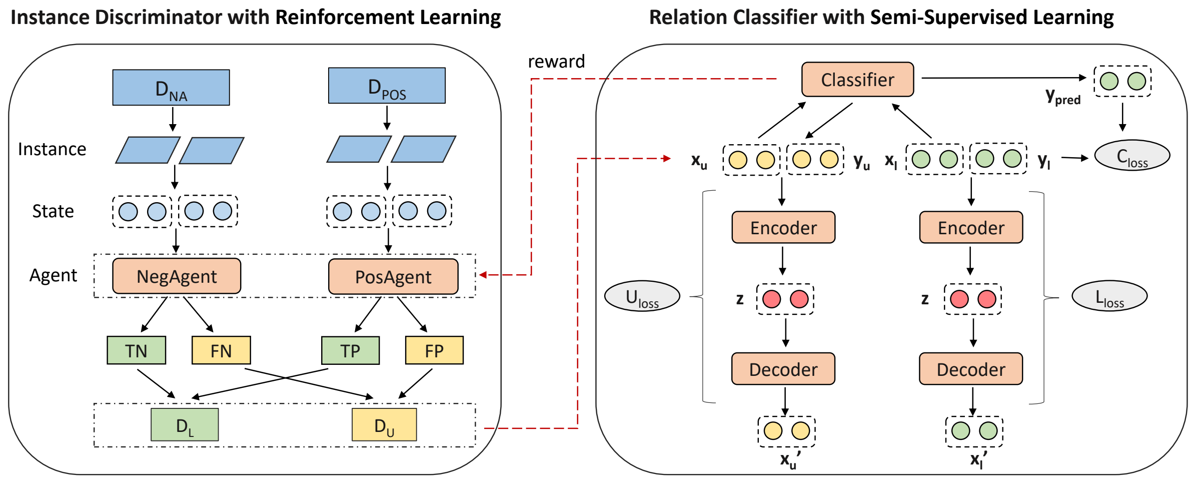


Figure 2: The framework of train process. Instance discriminator, consisting of PosAgent and NegAgent, aims to recognize false-positive (FP) and false-negative (FN) instances from positive dataset ($\mathcal{D}_{\text{POS}}$) and NA dataset ($\mathcal{D}_{\text{NA}}$) respectively. Afterward, true-positive (TP) and true-negative (TN) instances are split into labeled data ($\mathcal{D}_l$) while FP and FN instances are split into unlabeled data ($\mathcal{D}_u$). We adopt SemiVAE, which consists of an encoder, a decoder and a classifier, to train a robust relation classifier with semi-supervised learning utilizing the above data $\mathcal{D}_l$,$\mathcal{D}_u$. More details are introduced in Section 3.2 and 3.3.

learning and adversarial learning to explicitly remove incorrectly labeled sentences. However, they neglect the useful inherent information of those sentences which should be replaced correctly. In other words, they remove the noise rather than utilize it in the right way.

Furthermore, Xu et al. (2013) correct false negative instances by using pseudo-relevance feedback to expand the origin knowledge base. Liu et al. (2017) apply a dynamic soft-label instead of the immutable hard label produced by DS during the training process. Luo et al. (2017) design a transition matrix which characterizes the underlying noise pattern to correct noisy labels. They utilize the noisy data and address the false negative problem to some extent, but they still suffer from the drawback that errors may be propagated because the model is unable to correct its own mistakes.

In this work, we propose a unified framework for learning a discriminator to recognize both false-positive and false-negative instances with reinforcement learning, and utilizing the incorrectly labeled data as unlabeled data in semi-supervised learning way.

3 Methodology

In this section, we introduce our framework and the details of instance discriminator and relation classifier as follows.

3.1 Framework

In MIL paradigm, the entire instances are split into multiple entity-pair bags $\{B_{h_i,t_i}\}_{i=1}^{k}$. The sentences in $B_{h,t}$ mention both head entity h and tail entity t. Here we denote dataset as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where x_i is a sentence associated with the corresponding entity pair, y_i is a noisy relation label produced by distant supervision and n is the total number of sentences contained in each bag. As mentioned above, *NA* is a special relation which indicates the sentence does not express any relations in the KB. We define other relations in the KB as positive relations. Accordingly, we split the dataset into $\mathcal{D}_{\text{POS}}$ and $\mathcal{D}_{\text{NA}}$.

In the scenario of distant supervision, an ideal model is not only capable of capturing valid supervision information about correctly labeled data with less noise, but also leveraging information contained in incorrectly labeled data by correcting the noisy label implicitly.

As a result, we solve the task of distant supervision relation classification in two steps. As depicted in Figure 2, we design an instance discriminator to heuristically recognize false positive and false negative instances from the noisy distantly-supervised dataset with reinforcement learning. The correctly labeled instances discovered by the discriminator are split into labeled data while the incorrectly labeled ones are split into unlabeled data. The details of the instance discriminator will be introduced in Section 3.2. After scanning the entire noisy dataset, we train a robust classifier with semi-supervised learning utilizing above labeled data and unlabeled data. The details of the relation classifier will be introduced in Section 3.3. Meanwhile, the relation classifier provides rewards to the instance discriminator for updating parameters of its policy function.

3.2 Instance discriminator

We regard recognizing incorrectly labeled instances as a reinforcement learning problem. The instance discriminator acts as an agent interacting with the environment that consists of a noisy dataset and a relation classifier. The agent is parameterized with a policy network $\pi(a|s;\theta)$ which gives the probability distribution of action a at each state s and receives reward r from the relation classifier to update parameters θ. Note that *NA* indicates that there is no relation between two entities or that the relation is of no interest. The relation *NA* is very ambiguous since instances have no unified pattern. Thus we cannot decide whether a sentence belongs to *NA* only by the fact that it does not express any other positive relation. Under this consideration, we adopt two agents, PosAgent and NegAgent, to recognize false positive and false negative instances respectively. The definitions of the components in RL are introduced as follows.

State The state includes the semantic and syntactic information of current sentence and the relation label given by DS. We use a piecewise convolutional neural network (PCNN) (Zeng et al., 2015) to convert each sentence into real-valued vector $\boldsymbol{x}$ and build a class representation matrix $\boldsymbol{M}$ to represent each relation type. As we decide whether the current sentence is correctly labeled according to the similarity between the semantic meanings of sentence and relation, we only take the current sentence into consideration without sentences in early states. For PosAgent, state $\boldsymbol{s}_p$ is the concatenation of the current sentence vector $\boldsymbol{x}$ and corresponding relation embedding. For NegAgent, we represent state $\boldsymbol{s}_n$ by the vector of relational scores based on the representation of the current sentence $\boldsymbol{x}$.

$$\begin{aligned} \boldsymbol{s}_p &= [\boldsymbol{x}; \boldsymbol{M}_{[y]}] \\ \boldsymbol{s}_n &= \boldsymbol{M}\boldsymbol{x} + \boldsymbol{b} \end{aligned} \tag{1}$$

where y is relation label of the current sentence. $\boldsymbol{b} \in \mathbb{R}^{n_r}$ is a bias vector and n_r is the number of class.

Action We desire the agent to distinguish whether the current sentence is mislabeled or not. Therefore, the action of our agent is defined as $a_i \in \{0, 1\}$, where 0 indicates the sentence is incorrectly labeled and 1 indicates the sentence is correctly labeled.

Reward The reward function can reflect the advantage of redistributing the noisy data. As previously mentioned, the actions of our agent redistribute noisy data into labeled data and unlabeled data, corresponding to correctly labeled and incorrectly labeled instances. Therefore, the average likelihood of labeled data will be larger than unlabeled data when the agent makes correct actions. We define the difference of likelihood between them as the reward to evaluate the performance of our policy. Consequently, the reward is defined as follows:

$$r = \lambda(\frac{1}{|L|}\sum_{x \in L} p_\phi(y|x) - \frac{1}{|U|}\sum_{x \in U} p_\phi(y|x)) \tag{2}$$

where L and U is the subset of labeled data and unlabeled data respectively, and y is the relation label given by DS. $p_\phi(y|x)$ is calculated by the relation classifier from the semi-supervised learning framework. λ is used to scale the difference to a rational numeric range.

Training the Policy-based Agent

The objective of the agent is to maximize the expected reward of the actions sampled from the probability distribution. Given a mini-batch data B, following the policy, our agent produces a set of probability distributions of actions $\pi(a_i|s_i;\theta)$. Based on the actions, the agent achieves a performance-driven reward r. We use a policy gradient strategy to compute the gradient and update our agent referring to the policy gradient theorem (Sutton et al., 1999) and the REINFORCE algorithm (Williams, 1992). The parameters of the policy network are updated according to the following gradient:

$$\theta \leftarrow \theta + \alpha \sum_{i=1}^{|B|} \nabla_\theta r \log \pi(a_i|\boldsymbol{s}_i;\theta) \tag{3}$$

As the goal of our agent is to determine whether an annotated sentence expresses target relation with weak supervision, we need a relation classifier to compute the reward for updating the policy network. We first pre-train our classifier on the entire dataset with supervised learning until rough convergence. Then we pre-train the policy network by receiving rewards from the pre-trained classifier with the parameters frozen. The pre-training strategy is necessary as it saves time that would otherwise be spent training the model by trial and error. It is also widely used by other related works (Silver et al., 2016; Bahdanau et al., 2016). The training procedure for instance discriminator is summarized in Algorithm 1.

3.3 Relation Classifier

In order to reach the maximum utilization of noisy data, we train our relation classifier with semi-supervised learning. Below, we introduce PCNN and SemiVAE, the method we adopt for semi-supervised learning.

PCNN

We take the widely used CNN architecture (PCNN) (Zeng et al., 2015; Lin et al., 2016) to encode input sentences into low-dimensional vectors and predict their corresponding relation labels.

Given a sentence containing an entity pair, we represent the i-th word as $\boldsymbol{v}_i$ by concatenating its word embedding $\boldsymbol{w}_i$ and position embedding $\boldsymbol{p}_i$ which encodes the relative distances from it to two entities ($\boldsymbol{v}_i \in \mathbb{R}^d, w_i \in \mathbb{R}^{d_w}, p_i \in \mathbb{R}^{d_p}, d = d_w + d_p$).

Afterward, the convolution layer applies a kernel of the window size l to slide over the input sequence $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m\}$ and output the d_h-dimensional hidden embeddings $\boldsymbol{h}$, where $\boldsymbol{h} \in \mathbb{R}^{m \times d_c}$ and d_c is the number of feature maps.

Then, piecewise max-pooling is used to divide the hidden embeddings into three parts $\{\boldsymbol{h}^1, \boldsymbol{h}^2, \boldsymbol{h}^3\}$ by the position of head and tail entities. We perform max-pooling on each part respectively and get final embedding $\boldsymbol{x}$ by concatenating the pooling results, where $\boldsymbol{x} \in \mathbb{R}^{d_s} (d_s = d_c \times 3)$.

$$\boldsymbol{x} = [\max(\boldsymbol{h}^1); \max(\boldsymbol{h}^2); \max(\boldsymbol{h}^3)] \tag{4}$$

Finally, we formalize the probability of predicting y given sentence $\boldsymbol{x}$ as follows:

$$\begin{aligned} \boldsymbol{o} &= \boldsymbol{M}\boldsymbol{x} + \boldsymbol{b}, \\ p_\phi(y|\boldsymbol{x}) &= \frac{\exp{(o_y)}}{\sum_{k=1}^{n_r} \exp{(o_k)}} \end{aligned} \tag{5}$$

where $\boldsymbol{M} \in \mathbb{R}^{n_r \times d_s}$ is the class embeddings of each relation and $\boldsymbol{b} \in \mathbb{R}^{n_r}$ is a bias vector.

Semi-supervised VAE

SemiVAE, a semi-supervised learning method based on variational inference, is introduced and developed by (Kingma et al., 2014; Xu et al., 2017). The inference model consists of three components as follows. An encoder network $p_\varphi(z|x, y)$ encodes data x and label y into a latent variable z. The decoder network $p_\psi(x|z, y)$ is used to estimate the probability of generating x given z and categorical label y. Finally, classifier $p_\phi(y|x)$ predict the corresponding label y of x. We model both encoder and decoder by multilayer perceptron (MLP) and employ the PCNN model as the classifier in SemiVAE.

For the case of labeled data (x_l, y_l), the evidence lower bound is:

$$\begin{aligned} \log p_\psi(x_l, y_l) \geq & \mathbb{E}_{p_\varphi(z|x_l,y_l)}[\log p_\psi(x'_l|y_l, z)] \\ & + \log p_\psi(y_l) - D_{KL}(p_\varphi(z|x_l, y_l)||p(z)) \\ & = -\mathcal{L}(x_l, y_l) \end{aligned} \tag{6}$$

where first term represent the expectation of the conditional log-likelihood on latent variable z and the last term is Kullabck-Leibler divergence between the prior distribution $p(z)$ and the latent posterior $p_\phi(z|x_l, y_l)$.

For the case of unlabeled data x_u, the unobserved label y_u is obtained from the classifier in the inference model. The variational lower bound is:

$$\begin{aligned} \log p_\psi(x_u) \geq & \sum_y p_\phi(y_u|x_u)(-\mathcal{L}(x_u, y_u)) + \mathcal{H}(p_\phi(y_u|x_u)) \\ & = -\mathcal{U}(x_u) \end{aligned} \tag{7}$$

where $\mathcal{H}$ denotes the entropy of $p_\phi(y_u|x_u)$.

Since the classifier $p_\phi(y|x)$ is unable to learn directly from labeled data, a classification loss is introduced as:

$$\mathcal{C} = \mathbb{E}_{(x,y) \in D_l}[-\log p_\phi(y|x)] \tag{8}$$

To maximize the evidence lower bound of both labeled data and unlabeled data and minimize the classification loss, the objective is defined as:

$$\mathcal{J} = \sum_{(x,y) \in \mathcal{D}_l} \mathcal{L}(x, y) + \sum_{x \in \mathcal{D}_u} \mathcal{U}(x) + \beta \mathcal{C} \tag{9}$$

where $\mathcal{D}_l$ and $\mathcal{D}_u$ are labeled and unlabeled data respectively, β is a factor used to scale the classification loss of labeled data.

Algorithm 1 Reinforcement Learning Algorithm for Instance Discriminator.

```
Input: Origin dataset $\mathcal{D}_{\text{POS}} = \{(x_i, y_i)\}_{i=1}^{n}$.
    ( To be clear, we demonstrate the training procedure of PosAgent. NegAgent is trained in the same way. )
Output: Labeled data $\mathcal{D}_l$ , unlabeled data $\mathcal{D}_u$.
Initialize parameters of policy network as $\theta$.
for step $t = 1 \to T$ do
    Partition $\mathcal{D}_{\text{POS}}$ into minibatches of size $b_s$
    for minibatch $B \subset \mathcal{D}_{\text{POS}}$ do
        $B = \{(x_j, y_j)\}_{j=1}^{b_s}$
        Sample actions for each sentence in $B$: $a_j \sim \pi(a_j|s_j; \theta)$
        if $a_j == 0$ then
            Add $x_j$ to $\mathcal{D}_u$
        else
            Add $(x_j, y_j)$ to $\mathcal{D}_l$
        end if
        Calculate reward $r$ by Eq.(2)
        Update $\theta$ by Eq.(3)
    end for
end for
```

Algorithm 2 Semi-supervised Learning Algorithm for Relation Classifier.

```
Input: Labeled data $\mathcal{D}_l$, unlabeled data $\mathcal{D}_u$.
Initialize parameters of relation classifier as $\phi$.
for epoch $i = 1 \to N$ do
    Sample m data pair $(x_l, y_l)$ from $\mathcal{D}_l$
    Sample m data $x_u$ from $\mathcal{D}_u$ and predict their corresponding unobserved label $y_u$ via $p_\phi(y|x)$
    Update $\phi$ by Eq.(9)
end for
```

After our reinforcement learning process, we obtain an instance discriminator which possesses the capability of recognizing incorrectly labeled instances from the noisy dataset. Additionally, the entire DS dataset $\mathcal{D}$ is split into labeled data $\mathcal{D}_l$ and unlabeled data $\mathcal{D}_u$. Therefore, we utilize the above data to train SemiVAE model and obtain a robust relation classifier which explicitly learns from correctly labeled data and correct incorrectly labeled data implicitly. The training procedure for relation classifier is summarized in Algorithm 2.

4 Experiment

4.1 Datasets and Evaluation

We evaluate our model on a widely used dataset that is generated by aligning entity pairs from Freebase with New York Times corpus(NYT)[2] and developed by (Riedel et al., 2010). Entity mentions are recognized by the Stanford named entity recognizer (Finkel et al., 2005). The relation facts in Freebase are divided into two parts for training and testing respectively. The sentences from the corpus of the years 2005-2006 are used as the training instances, and sentences from 2007 are used as the testing instances. There are 52 positive relations and a special relation *NA*.

Batch size b_s	160
Word Dimension d_w	50
Position Dimension d_p	5×2
Convolution Filter Dimension d_c	230
Convolution Window Size l	3
Latent Variable Dimension d_z	100
Dropout p	0.5
Regulator λ, β	100, 2

Table 2: Hyperparameter settings

Following previous works, we evaluate our model on the held-out evaluation, which compares relation facts extracted from the test corpus with those in Freebase. We adopt aggregated precision/recall curves and precision@N (P@N) to illustrate the performance of our model.

4.2 Parameter Settings

We adopt the Adam (Kingma and Ba, 2014) optimizer to optimize our instance discriminator and relation classifier with learning rate 0.0001 and 0.001 respectively. We also apply dropout to prevent overfitting. More detailed hyperparameter settings are presented in Table 2.

4.3 Overall Evaluations Results

We adopt the following baselines with which we compare our model:

- **Mintz** (Mintz et al., 2009) is the original distantly supervised model. **MultiR** (Hoffmann et al., 2011) and **MIML**(Surdeanu et al., 2012) handle overlapping relation problem with graphical model in multi-instance and multi-instance multi-label framework. All above models are based on handcrafted feature.

- **PCNN+ONE** (Zeng et al., 2015) and **PCNN+ATT** (Lin et al., 2016) are both robust models to solve noisy labeling problem

---

[2] http://iesl.cs.umass.edu/riedel/ecml/

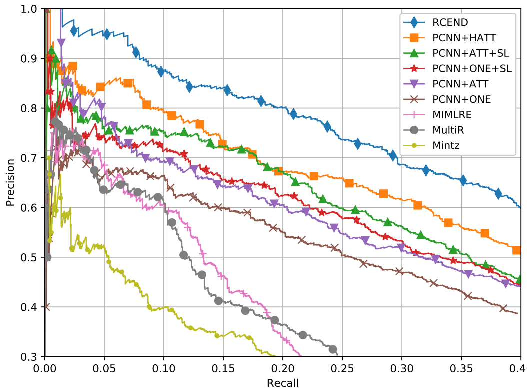


Figure 3: Precision-recall curves of our model and baselines.

P@N	100	200	300	Mean
PCNN+ONE	72.3	69.7	64.1	68.7
PCNN+ATT	76.2	73.1	67.4	72.2
PCNN+ONE+SL	84.0	81.0	74.0	79.7
PCNN+ATT+SL	87.0	84.5	77.0	82.8
PCNN+HATT	88.0	79.5	75.3	80.9
RCEND	**95.0**	**87.5**	**84.4**	**88.9**

Table 3: Top-N precision (P@N) of our model and baselines

based on the at-least-one assumption and selective attention. **PCNN+HATT** (Han et al., 2018b) is a attention-based method which employs hierarchical attention to exploit correlations among relations.

- **PCNN+ONE+SL** and **PCNN+ATT+SL**(Liu et al., 2017) use a soft-label method to alleviate the negative impact of the noisy labeling problem.

We compare our model with aforementioned baselines and the results are shown in Figure 3. From the overall result we can see that: (1) All feature-based models preform poorly as their features are derived from NLP tools, which will generate errors that propagate through in model. (2) PCNN+ONE and PCNN+ATT boost the performance because they reduce noise in the bag of entity pair by selecting the most confident sentence or de-emphasize the incorrectly labeled sentences with an attention mechanism. (3) When PCNN+ONE and PCNN+ATT use soft labels, they achieve an improvement. This indicates correcting the noisy label is helpful to relation classification in MIL scheme. (4) PCNN+HATT further enhances the performance as it incorporates hierarchical information of relations to improve the attention mechanism. (5) Our method RCEND achieves the best precision over the entire recall range compared with all baselines. The performance achieves further improvement when we regard the incorrectly labeled sentences as unlabeled data and adopt a semi-supervised learning method to train our model. It shows that exploiting noisy data with our method is beneficial to promote distant supervision relation classification.

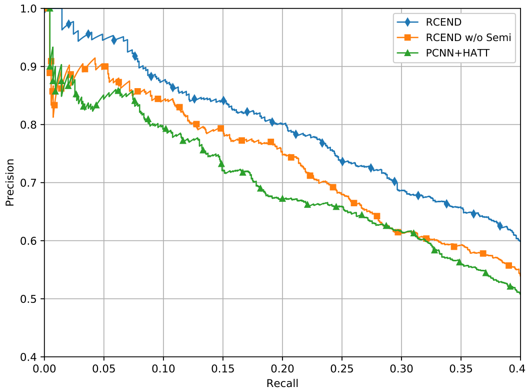


Figure 4: Precision-recall curves of our model with different settings.

P@N	100	200	300	Mean
RCEND	**95.0**	**87.5**	**84.4**	**88.9**
RCEND w/o Semi	90.0	84.6	79.1	84.6
RCEND(P)	87.1	82.1	80.1	83.3
RCEND(N)	89.1	85.1	81.1	85.1

Table 4: Top-N precision (P@N) of our model with different settings.

We also report the result of Precisions@N (100, 200, 300) in Table 3. We can see that our method outperforms the baselines on the precision values of top N triples extracted.

4.4 Impact of Unlabeled Data

To further verify the impact of the unlabeled data, we conduct experiments with both utilization and non-utilization of unlabeled data. The results are presented in Figure 4. Note that, the method RCEND w/o Semi is similar to the method proposed by (Feng et al., 2018), which only removes the incorrectly labeled sentences but does not fully utilize them. We can see that it achieves higher precision over the entire level of recall compared to PCNN+HATT, the best noise-tolerate method in MIL scheme, which shows that removing noise is better than dealing with them with soft attention weights. However, it is still unable to surpass our method. In Table 4, our method also shows notable improvement over RCEND w/o Semi. This demonstrates that fully utilizing noisy data is more advantageous than reducing them.

Type	Sentence	Predict	DS
FN	C1: [**Oliver O'Grady**] is now a silver-haired , twinkly-eyed resident of [**Ireland**] , where Ms. Berg often films him in parks ...	Nationality	NA
	C2: ... said [**John Allison**] , editor of [**Opera**] magazine , based in London.	EmployedBy	NA
	C3: [**Jean-Pierre Bacri**] is a famous writer , who is too self-centered to care about his lonely , overweight , 20-year-old daughter , [**Lolita Marilou Berry**]...	ChildrenOf	NA
FP	C4: they wanted to interview [**Bill Cosby**] after they met with a former Temple University employee who has accused him of groping her in his home in suburban [**Philadelphia**]	LivedIn	BornIn
	C5: "Without the fog , [**London**] wouldn't be a beautiful city."the French painter Claude Monet wrote to his wife , Alice , during one of his long visits to [**England**] from France.	NA	Capital
	C6: MTV Goes to Africa MTV opened its first local music channel in Africa this week , a step touted by the singer [**Lebo Mathosa**] , above , at an MTV event in [**Johannesburg**].	NA	DieIn

Table 5: Examples for case study. The first three sentences are examples of false negative case and the final three are examples of false positive case.

This can be partially explained due to the label rectification of the incorrectly labeled data during semi-supervised learning with correctly labeled data which improves the generalization performance.

4.5 Impact of False Positives and False Negatives

The goal of this experiment is to inspect whether the relation classifier is enhanced more through the utilization of false negatives or through the utilization of false positives. As depicted in Figure 5, RCEND(P) only recognizes the false positive sentences in $\mathcal{D}_{\text{POS}}$ by PosAgent and regards them as unlabeled data. Likewise, RCEND(N) only discovers and utilizes false negative sentences. RCEND(P) and RCEND(N) behave similarly and achieve further improvement when utilizing both false-positive and false-negative sentences, which implies that both of them are important and promote the ability of our relation classifier. And the results in Table 4 also show utilizing false negative data performs slightly better than false positives since false negative data might be predicted as positive relation and increase samples of the target relation to learn a more accurate decision boundary.

4.6 Case Study

We sample some examples of incorrectly labeled data which are regarded as unlabeled data during training. In Table 5, it can be seen that our discriminator recognizes both false positive and false negative instances. For example, though the fact (John Allison, *EmployedBy*, Opera) is absent in the KB due to the incompleteness of the KB, C2 expresses *EmployedBy* relation and provides evidence of target relation. Additionally, C4 is mislabeled as *BornIn* due to the relational fact (Bill Cosby, *BornIn*, Philadelphia), even though it mentions *LivedIn* relation. Further more, they are all predicted correctly by our relation classifier in the end which shows that our model indeed captures the valid information of noisy data and exploits them to enhance its ability.

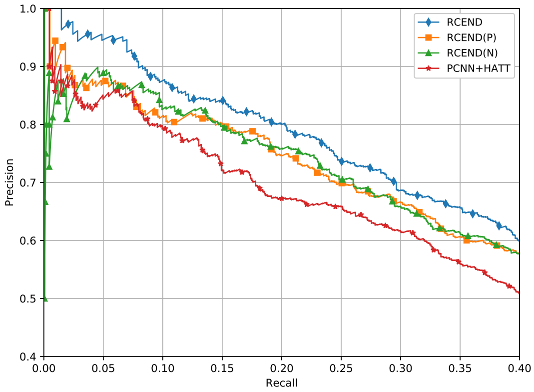


Figure 5: Precision-recall curves of our model with different settings.

5 Conclusion

In this paper, we proposed RCEND to fully exploit valid information of the noisy data in distant supervision relation classification. The instance discriminator is trained with reinforcement learning, which aims to recognize the instances mislabeled by distant supervision. We treat the correctly labeled instances as labeled data and incorrectly labeled ones as unlabeled data. Afterward, we adopt a semi-supervised learning method to learn a robust relation classifier to utilize the data. In this way, not only can our model reduce the side effect of noisy labels, but also adequately take advantage of valid information contained in noisy data. Experiments demonstrate that our model outperforms state-of-the-art baselines.

Acknowledgments

We would like to express gratitude to Robert Ridley and the anonymous reviewers for their valuable feedback on the paper. This work is supported by the National Natural Science Foundation of China (No. 61672277, U1836221), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *Computing Research Repository*, arXiv:1607.07086.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Jun Feng, Minlie Huang, Li Zhao, and Xiaoyan Yang, Yang amd Zhu. 2018. Reinforcement learning for relation classification from noisy data.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018a. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 4832–4839.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018b. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 541–550. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133. Association for Computational Linguistics.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1790–1795. Association for Computational Linguistics.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 430–439. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1003–1011. Association for Computational Linguistics.

Pengda Qin, Weiran XU, and William Yang Wang. 2018a. Dsgan: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 496–505. Association for Computational Linguistics.

Pengda Qin, Weiran XU, and William Yang Wang. 2018b. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2137–2147. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 455–465. Association for Computational Linguistics.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 665–670. Association for Computational Linguistics.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3358–3364.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of International Conference on Computational Linguistics*, pages 2335–2344.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics.

Misspelling Oblivious Word Embeddings

Bora Edizel
Facebook AI, London
`edizel@fb.edu`

Aleksandra Piktus
Facebook AI, London
`piktus@fb.com`

Piotr Bojanowski
Facebook AI, Paris
`bojanowski@fb.com`

Rui Ferreira[*]
Spotify, London
`ruif@spotify.com`

Edouard Grave
Facebook AI, Paris
`egrave@fb.com`

Fabrizio Silvestri
Facebook AI, London
`fsilvestri@fb.com`

Abstract

In this paper we present a method to learn word embeddings that are resilient to misspellings. Existing word embeddings have limited applicability to malformed texts, which contain a non-negligible amount of out-of-vocabulary words. We propose a method combining FastText with subwords and a supervised task of learning misspelling patterns. In our method, misspellings of each word are embedded close to their correct variants. We train these embeddings on a new dataset we are releasing publicly. Finally, we experimentally show the advantages of this approach on both intrinsic and extrinsic NLP tasks using public test sets.

1 Introduction

Word embeddings constitute a building block of many practical applications across NLP and related disciplines. Techniques such as Word2Vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014) have been extensively used in practice. One of their drawbacks, however, is that they cannot provide embeddings for words that have not been observed at training time, i.e. *Out-Of-Vocabulary* (OOV) words. In real-world tasks, the input text is often generated by people and misspellings, a common source of OOV words, are frequent (e.g. (Cucerzan and Brill, 2004) report that misspellings appear in up to 15% of web search queries). As a consequence, the quality of downstream applications of word embeddings in real-world scenarios diminishes.

Simply allowing the inclusion of misspellings into corpora and vocabularies in existing methodologies might not provide satisfactory results. The sparsity of misspellings would most likely prevent their embeddings from demonstrating any interesting properties. Trying to balance the representation of misspellings with the representation of correctly spelled variants in training data by artificially introducing misspelled variants for every word in the corpus would on the other hand cause up to an exponential growth in the size of the training data, making training of the models infeasible.

To address this deficiency, we propose Misspelling Oblivious (word) Embeddings (**MOE**), a new model combining FastText (Bojanowski et al., 2017) with a supervised task which embeds misspellings close to their correct variants. We carry out experiments on well established tasks and on their variants adapted to the misspellings problem. We also propose new methods of evaluating embeddings specifically designed to capture their quality on misspelled words. We train **MOE** embeddings on a new dataset we are releasing publicly. Finally, we experimentally show the advantages of this approach on both intrinsic and extrinsic NLP tasks using public test sets. Summarizing, we propose the following contributions:

- a novel problem and a non-trivial solution to building word embeddings resistant to misspellings;
- a novel evaluation method specifically suitable for evaluating the effectiveness of **MOE**;
- a dataset of misspellings [1] to train **MOE**.

The reminder of this paper is structured as follows. Section 2 gives an overview of the word embeddings literature. In Section 3.1 we introduce Word2Vec and FastText models. We introduce the **MOE** model in Section 3.2. Section 4 contains the descriptions of datasets we trained on and section 5 contains the description of experiments we conducted and their results. In Section 6 we present our conclusions and plans for further research.

---

[*] This work was carried out when the author was working as an employee at Facebook London.

[1] `https://bitbucket.org/bedizel/moe`

Proceedings of NAACL-HLT 2019, pages 3226–3234
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related Work

One of the first works to introduce the concept of a distributed representation for symbolic data was (Hinton, 1986). Later, the Information Retrieval community proposed techniques of embedding words into a vector space. Latent Semantic Indexing (Deerwester et al., 1990) was one of the most influential works in this area. (Bengio et al., 2003) introduced the first neural language model which jointly learned word embeddings. Although such a language model was outperforming the baselines, it was not practical because of long training time requirements. (Collobert and Weston, 2008) proposed new neural architectures for word embeddings and showed that pre-trained word embeddings can be very valuable for some downstream NLP tasks. Word2Vec (Mikolov et al., 2013b,a) got very popular both because of its effectiveness and its ability to train models on very large text corpora efficiently. (Levy and Goldberg, 2014) showed that Word2Vec's skip-gram with negative sampling model (SGNS) is implicitly equivalent to word co-occurrence matrix factorization. Besides neural approaches, (Pennington et al., 2014) proposed an SVD based architecture which gained a lot of attention because it allowed to effectively consider the popularity of each word in the model definition.

FastText (Bojanowski et al., 2017) is a popular, recent proposal in the area of word embeddings. FastText introduces subword-level features to the Word2Vec framework which enables building embeddings for OOV words (see details in Section 3.1). An alternative approach, also capable of yielding representations for OOV words, is MIMIK (Pinter et al., 2017). MIMICK learns a function from input words to embeddings by minimizing the distance between embeddings produced by a char-based approach and the pre-trained embeddings. As opposed to **MOE**, MIMICK does not support misspellings explicitly and it requires a set of pre-trained embeddings as input. We consider MIMICK to be a viable alternative to FastText which deserves future work exploring its performance on misspelled text.

3 Misspelling Oblivious Embeddings

3.1 The FastText Model

Our current work can be viewed as a generalization of FastText, which, in turn, extends the the skip-gram with negative sampling (SGNS) architecture, proposed as a part of the Word2Vec framework. In this section we will briefly discuss major additions to SGNS introduded by FastText.

Let V be a vocabulary of words and $T = w_1, w_2, \ldots, w_{|T|}$ be a text corpus, represented as a sequence of words from V. We define the context of a word $w_i \in V$ as $C_i = \{w_{i-l}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+l}\}$ for some l set as a hyperparameter. In the SGNS model, a word w_i is represented by a single embedding vector $\mathbf{v}_i$ equivalent to the input vector of a simple feed-forward neural network, trained by optimizing the following loss function:

$$L_{W2V} := \sum_{i=1}^{|T|} \sum_{w_c \in C_i} [\ell(s(w_i, w_c)) + \sum_{w_n \in N_{i,c}} \ell(-s(w_i, w_n))] \tag{1}$$

where ℓ denotes the logistic loss function $\ell(x) = log(1 + e^{-x})$ and $N_{i,c}$ is a set of negative samples drawn for the current word w_i and its context $w_c \in C_i$. s is the scoring function, which for SGNS is defined as the the dot product $\mathbf{v}_i^T \mathbf{u}_c$, where $\mathbf{u}_c$ is an output vector associated with the word w_c and $\mathbf{v}_i$ is an input vector associated with the word w_i. Therefore, $s\left(w_i, w_c\right) = \mathbf{v}_i^T \mathbf{u}_c$.

In FastText, we additionally embed subwords (also referred to as character n-grams) and use them to construct the final representation of w_i. Formally, given hyperparameters m and M denoting a minimum and a maximum length of an n-gram respectively, the FastText model embeds all possible character n-grams of the word such that $m \leq n \leq M$. E.g. given $m = 3$, $M = 5$ and the word *banana*, the set of n-grams we consider is $ban, ana, nan, bana, anan, nana, banan, anana$. Now, let $\mathcal{G}_{w_i}$ denote the set of all n-grams of a word $w_i \in V$ plus the word itself (e.g. $\mathcal{G}_{\text{banana}}$ is the set defined in the example above plus the word *banana* itself). Given $\mathcal{G}_{w_i}$, FastText scoring function for a word w_i and a context w_c is defined as follows:

$$s_{FT}(w_i, w_c) := \sum_{\mathbf{v}_g, g \in \mathcal{G}_{w_i}} \mathbf{v}_g{}^T \mathbf{u}_c \tag{2}$$

Therefore, the representation of w_i is expressed through the sum of the representations of each of the n-grams derived from w_i plus the representation of w_i itself. FastText optimizes the loss function in Eq.1, but uses the scoring function s_{FT} defined in Eq.2. An extensive experimentation has shown that FastText improves over the original Word2Vec skip-gram model. The loss function of FastText will be referred to as L_{FT} throughout the rest of this work.

3.2 The MOE Model

As was shown empirically in the FastText paper, the n-grams which impact the final representation of a word the most in FastText correspond to morphemes. Based on this observation, we hypothesize that although FastText can capture morphological aspects of text, it may not be particularly resistant to misspellings which can occur also withing the dominant morphemes. In this section, we present the architecture of our model - **MOE** or Misspelling Oblivious (word) Embeddings. **MOE** holds the fundamental properties of FastText and Word2Vec while giving explicit importance to misspelled words.

Loss Function. The loss function of **MOE** is a weighted sum of two loss functions: L_{FT} and L_{SC}. L_{FT} is the loss function of FastText which captures semantic relationships between words. L_{SC} or the spell correction loss aims to map embeddings of misspelled words close to the embeddings of their correctly spelled variants in the vector space. We define L_{SC} as follows:

$$L_{SC} := \sum_{(w_m,w_e)\in M} [\ell(\hat{s}(w_m, w_e)) + \sum_{w_n \in N_{m,e}} \ell(-\hat{s}(w_m, w_n))] \tag{3}$$

where M is a set of pairs of words (w_m, w_e) such that $w_e \in V$ is the expected (correctly spelled) word and w_m is its misspelling. $N_{m,e}$ is a set of random negative samples from $V \setminus \{w_m, w_e\}$. L_{SC} makes use of the logistic function $\ell(x) = log(1 + e^{-x})$ introduced in Section 3.1. The scoring function $\hat{s}$ is defined as follows:

$$\hat{s}(w_m, w_e) = \sum_{\mathbf{v}_g, g\in\hat{\mathcal{G}}_{w_m}} \mathbf{v}_g{}^T \mathbf{v}_e \tag{4}$$

where $\hat{\mathcal{G}}_{w_m} := \mathcal{G}_{w_m} \setminus \{w_m\}$. Therefore, the scoring function is defined as the dot product between the sum of input vectors of the subwords of w_m and the input vector of w_e. Formally, the term $\ell(\hat{s}(w_m, w_e))$ enforces predictability of w_e given w_m. Intuitively, optimizing L_{SC} pushes the representation of a misspelling w_m closer to the representation of the expected word w_e. It is also worth mentioning that embeddings for w_m and w_e share the same parameters set. The complete loss function of **MOE**, L_{MOE}, is defined as follows:

$$L_{MOE} := (1 - \alpha)L_{FT} + \alpha\frac{|T|}{|M|}L_{SC} \tag{5}$$

Optimizing the loss functions L_{FT} and L_{SC} concurrently is not a straightforward task. This is because two different loss functions iterate over two different datasets: the text corpus T, and the misspellings dataset M. The optimization process should be agnostic to the sizes of T and M in order to prevent results from being severely affected by those sizes. Therefore, we scale L_{SC} with the coefficient $|T|/|M|$. This way the importance of a single Stochastic Gradient Descent (SGD) update for L_{FT} becomes equivalent to a single SGD update for L_{SC}. Moreover, α is the hyperparameter which sets the importance of the spell correction loss L_{SC} with respect to L_{FT} thus making **MOE** a generalization of FastText.

4 Data

As mentioned in Section 3.2, **MOE** jointly optimizes two loss functions, each of which iterates over a separate dataset - a corpus of text for the FastText loss L_{FT} and a set of pairs (misspelling, correction) for the spell correction loss L_{SC}. In this section, we will briefly discuss how we obtain each of these datasets.

4.1 English text corpus

We use an English Wikipedia dump[2] as the text corpus T to optimize L_{FT}. The baseline FastText model is also trained on this dataset. Matt Mahoney's perl script[3] is used for pre-processing the raw Wikipedia dump. After pre-processing, the training corpus consists of $|T|= 4,341,233,424$ words. When generating the vocabulary V based on the corpus, we apply a frequency threshold of 5. After deduplication and thresholding, the size of the vocabulary for our corpus is $|V|= 2,746,061$ words. We also apply progressive subsampling of

---

[2] `dumps.wikimedia.org`

[3] `http://mattmahoney.net/dc/textdata`

frequent words in order to not assign too much importance to highly frequent words.

4.2 Misspelled data generation

The misspellings dataset M consists of a set of pairs (w_m, w_e), where $w_e \in V$ represents a (presumably correctly spelled) word from the vocabulary and w_m is a misspelling of w_e. Given the size of V, we opt for generating misspellings in an automated fashion, using an in-house misspellings generation script. The script is based on a simple error model, which captures the probability of typing a character p_m when a character p_e is expected (note that it's possible to have $p_m == p_e$), given a context of previously typed characters. The script is capable of generating misspellings of targeted edit distance for an input word w_i. In the reminder of this section, we'll discuss details of the script implementation.

Error model. In order to create the error model, we first mine query logs of a popular search engine[4] and identify cases where a query was manually corrected by a searcher. We then pivot on the modified character and for each such modification we save a triplet (c, p_m, p_e), where p_m is the pivot character before modification, p_e is the the target character after modification and c represents up to 3 characters preceding the pivot in the original query. E.g. given a query *hello worjd* corrected to *hello world*, we would generate four triplets: $[(wor, j, l), (or, j, l), (r, j, l), (\epsilon, j, l))]$, where ϵ represents an empty word. Similarly, we create triplets by pivoting on characters which have not been modified. After processing all available logs, we count each unique triplet. For each unique pair (c, p_m) of a context and a pivot, we then create a *target list* consisting of all possible targets p_e, each associated with its probability calculated based on counts. We then sort each target list in the order of decreasing probability.

Injecting misspellings. Let's consider a word $w_i \in V$ that we want to misspell. For each character $p \in w_i$, we take it's longest possible context c (up to 3 characters) and we look up the *target list* corresponding to (c, p). We then proceed along the *target list*, summing up the probabilities of subsequent targets until the sum is greater or equal to a randomly selected target probability $tp \in [0.0, 1.0]$. We then choose the corresponding target t as a replacement for p (note that in the majority of the cases $t == p$). We repeat this process for every word from V.

In order to respect real distributions of words in the text corpus T, we set the number of misspellings generated for each word $w_i \in V$ to be equal to the square root of the number of appearances of w_i in T. The total size of misspellings dataset generated in this fashion is $|M| = 20,068,964$ pairs. We make the dataset of misspellings publicly available at `https://bitbucket.org/bedizel/moe`.

5 Experiments

In this section, we describe the experimental set up used for training our models and the experiments we conducted.

5.1 Experimental setup

We use FastText[5] as a baseline for comparison since it can generate embeddings for OOV words which makes it potentially suitable for dealing with misspellings. We train the baseline model using the default hyperparameters provided by the authors. We consider character n-grams of lengths between $m = 3$ and $M = 6$, and we use 5 negative samples for each positive sample. Training **MOE** requires optimizing two loss functions L_{FT} and L_{SC} jointly. For optimizing L_{FT}, we use the same parameters as in the baseline. Additionally, to optimize L_{SC}, we experiment with 5 negative samples per positive sample. We sweep over a range of values for the coefficient combining the two losses: $\alpha \in \{0.01, 0.05, 0.1, 0.5, 0.25, 0.5, 0.75, 0.95, 0.99\}$. Both FastText and **MOE** are trained using Stochastic Gradient Descent with a linearly decaying learning rate for 5 epochs to learn vectors with 300 dimensions. We evaluate the performance of **MOE** on the following tasks: (intrinsic) Word Similarity, Word Analogy and Neighborhood Validity; (extrinsic) POS Tagging of English sentences.

We report the overlap between the misspellings seen at training time and misspellings present in tests in Table 1.

5.2 Intrisic Evaluation

We evaluate MOE on two classic intrinsic tasks, namely Word Similarity and Word Analogy and

---

[4] `https://www.facebook.com`

[5] `https://fasttext.cc/`

Test set	% of unseen
WS353 $r = 0.125$	25.05
WS353 $r = 0.250$	57.06
WS353 $r = 0.375$	64.37
RareWord $r = 0.125$	44.67
RareWord $r = 0.250$	70.18
RareWord $r = 0.375$	78.84
Word Analogies	50.71
Neighborhood Similarity	69.1

Table 1: Percentages of test misspellings unobserved at training time per test set. The r parameter indicates variants of respective word similarity test sets.

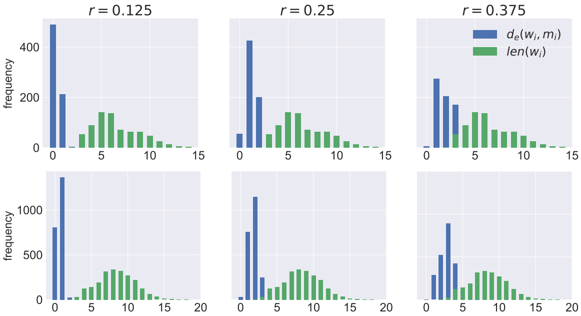


Figure 1: Distribution of edit distances $d_e(w_i, m_i)$ and lenghts of words $len(w_i)$ for WS353 variants (Top) and RW variants (Bottom).

on a novel intrinsic task evaluating the distance between vector embeddings of misspellings and their correctly spelled variants.

Word Similarity. In the word similarity task, we evaluate how well embeddings generated by **MOE** can capture the semantic similarity between two words. For this purpose, we use two datasets: (i) *WS353* (Finkelstein et al., 2001), and (ii) *Rare Words (RW)* (Luong et al., 2013). Both datasets contain pairs of words w_a, and w_b annotated with a real value in the range $[0, 10]$ representing the degree of similarity between w_a and w_b as perceived by human judges.

In order to evaluate how resilient our method is to spelling errors, for each pair of words (w_a, w_b) in the dataset, we provide a respective pair of misspellings (m_a, m_b). The misspellings are mined from search query logs of a real-world online search service. When desired misspellings are not available in the logs, we synthetically generate them using the same script we used to generate the set M (see Section 4 for details). We create 3 misspelled variants of both WS353 and RW datasets. In each variant we limit the ratio between the edit distance (Levenshtein, 1966) of the word and the misspelling $d_e(w_i, m_i)$ and the length of the word by a constant r, where $r \in \{0.125, 0.250, 0.375\}$, with $r = 0$ representing the original dataset. More precisely for each r we look for a misspelling which satisfies the following condition $d_e(w_i, m_i) = \lfloor r * len(w_i) \rfloor$. Effectively, if a word is too short to satisfy the condition, we preserve the original word (then $w_i = m_i$). Histograms in Figure 1 show the actual distribution of edit distances and lengths of words. As expected, edit distance increases steeply with the increase of r value. Edit distances are higher for the RW dataset since in average the length of words in RW is higher than on average length of words in WS353. Also, we observe that for $r = 0.125$, a significant portion of the words is not changed.

We conduct experiments for different values of the hyperparameter α which sets the trade-off between L_{FT} and L_{SC}, i.e. the importance assigned to semantic loss and misspelling loss. In the experiments, results corresponding to $\alpha = 0$ represents our baseline, FastText, since for $\alpha = 0$ the loss L_{MOE} is equal to L_{FT}.

We measure the Spearman's rank correlation (Spearman, 1904) between the distance of an input pair of words and the human judgment score both for the original and the misspelled pair. Figure 2 demonstrates the results of the word similarity task on the WS353 dataset. We observe that **MOE** is improving over FastText for WS353 variants with $r = 0.25$, and $r = 0.375$, and degrading performance when $r = 0$, and $r = 0.125$, where the majority of the words is not changed (see Figure 1 for the edit distance distribution). As we expected, larger values of α, corresponding to more attention given to misspellings during training, result in improvements for highly misspelled datasets.

For the RW dataset (Figure 2), we observe that for all the values of r, **MOE** improves over the FastText baseline when we set $\alpha = 0.05$. More specifically, when $r \in \{0, 0.125\}$ and when $\alpha <\approx 0.1$, the proposed method improves over the baseline. When the amount of misspellings is higher, i.e., $r \in \{0.25, 0.375\}$, **MOE** improves the results over the baseline for all of the α values. These results suggest that FastText may be a good baseline for dealing with low edit distance misspellings, however our model is better at capturing semantic relationships on higher edit distance misspellings. This is in line with our hypothesis presented in

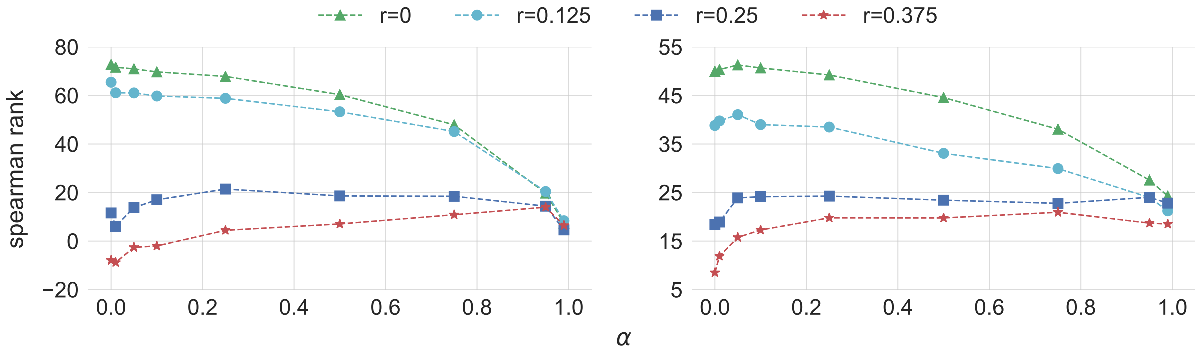


Figure 2: Experimental results for word similarity task for WS353 (Left) and RW (Right). $\alpha = 0$ values represent our baseline, FastText.

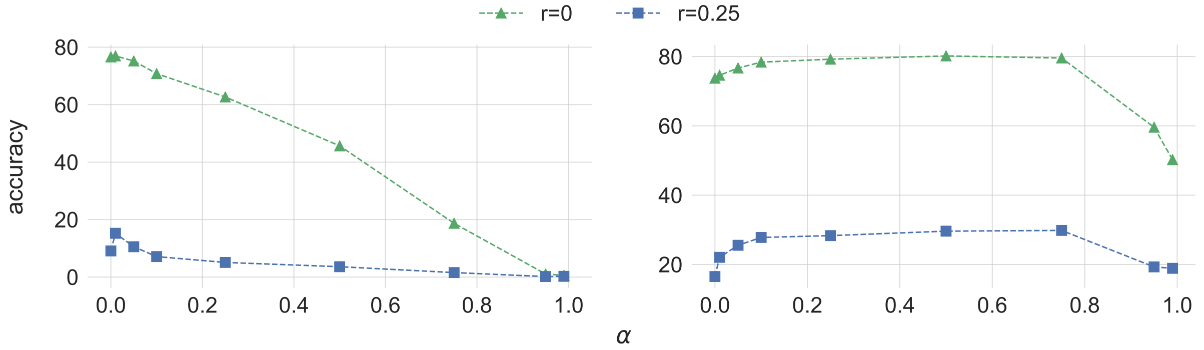


Figure 3: Experimental results for word analogy task, Semantic (Left) and Syntactic (Right). $\alpha = 0$ values represent our baseline, FastText.

Section 3.2.

Word Analogy. In addition to the word similarity, we also test the performance of **MOE** on the popular word analogy task introduced by (Mikolov et al., 2013a). This task attempts to measure how good the embeddings model is at preserving relationships between words. A single test sample from the word analogy dataset consists of four words A, B, C, D, forming two pairs - A, B and C, D, remaining in analogous relationships ("A is to B like C is to D"). There are two types of relationships: (i) *syntactic*, related to the structure of words; and (ii) *semantic*, related to their meanings. *banana, bananas, cat, cats* is an example of a syntactic test sample. In both pairs the fact that the second word is a plural version of the first constitutes a relationship between them. *Athens, Greece, Berlin, Germany* is an example of a semantic test sample. The relationship which is being tested in this case is that between the capital of a country and the country itself.

In addition to analyzing the canonical variant of the word analogies test, we also introduce a modification which is suitable specifically to the misspellings use-case. Given a line A, B, C, D from the original analogies dataset, we misspell the first pair of words, obtaining a line A', B', C, D, where A' is a misspelling of A and B' is a misspelling of B. We want to test if the misspelled pair A', B' preserves the relationship of the pair C, D. When generating misspellings we use a procedure similar to the one used for word similarities. We create one variant of the misspelled dataset, constraining the edit distance to $r = 0.25$.

Experimental results for the canonical version of the word analogy task, presented in Figure 3, show that **MOE** performs worse than FastText on the semantic analogy task. On the other hand, **MOE** performs better than the baseline on the syntactic analogies task. The results for the misspelled variant of the task show that, the overall performance of both the baseline and **MOE** is worse than on the canonical variant. For low values of $\alpha \in \{0.01, 0.05\}$, **MOE** outperforms the baseline on the semantic task, achieving an over 67% better score than FastText for $\alpha = 0.01$. **MOE** outperforms the baseline on the syntactic task for all tested values of α, improving by over 80% for $\alpha = 0.75$. For $\alpha = 0.01$, which achieved the best semantic result, the improvement on the syntactic task is over 33%.

The trends that we observe both in the canonical and the misspelled variant of the word analogies task seem to validate our choice of the loss function for the **MOE** model. It is clear that the FastText component of the loss is indispensable to learn the semantic relationships between words. In fact, it is the only component of the loss function which attempts to learn these relationships. Therefore, decreasing it's importance (by increasing the value of α) is reflected by a decay in the semantic analogies score. The spell-correction component of the loss function, on the other hand, leverages the relationship between correctly spelled words and their misspellings. As a side effect, it also adds additional subword information into the model. This explains our good performance on the syntactic analogies task. As our results on the misspelled variant of the task show, we improve over the baseline in understanding analogies on the misspelled words, which was one of the design principles for **MOE**.

Neighborhood Validity. One of the explicit objectives of **MOE** is to embed misspellings close to their correct variants in the vector space. In order to validate this hypothesis, we check where in the neighborhood of a misspelling the correct word is situated. Formally, for a pair (w_m, w_e) of a misspelling and its correction, we pick k nearest neighbors of the misspelling w_m in the embedding space using cosine similarity as a distance metric. We then evaluate the position of the correct word w_e within the neighborhood of w_m using two metrics:

- We use *MRR* (Voorhees et al., 1999) to score the neighborhood of the embeddings of misspellings (we assign a score of 0 if the correct word is not present).
- We also compute the *neighborhood coverage* defined as the percentage of misspellings for which the neighborhood contains the correct version.

The test set contains $5,910$ pairs (w_m, w_e) sampled from a collection of data coming from a real-world online service[6]. Figure 4 shows experimental results for Neighbor Validity task. We remind that $\alpha = 0$ denotes the FastText baseline.

The test results confirm our hypothesis. We observe that MRR increases when more importance is given to the L_{SC} component of the loss for any size of the neighborhood $k \in \{5, 10, 50, 100\}$. A similar trend can be observed for the neighborhood coverage task. We conclude that, on average, we're more likely to surface the correction using **MOE** than with FastText. What is more, whenever we are able to surface the correct version of a misspelled word, its position in the ranking is higher for **MOE** than for the FastText baseline.

5.3 Extrinsic Evaluation

POS Tagging. Finally, we evaluate **MOE** on a Part-of-Speech (POS) tagging task[7]. To assess the impact of misspellings we artificially inject misspellings in the dataset. We train **MOE** on three different dataset variants: a non-misspelled dataset, to verify that **MOE** does not jeopardize the performance on correct words; a dataset where 10% of words contain a misspelling, to simulate a realistic environment where some of the words are misspelled; and finally on a dataset where 100% of words contain misspellings, to simulate a highly distorted environment. We use a state-of-the-art POS tagger (Ma and Hovy, 2016) consisting of a Conditional Random Fields (CRF) model where embeddings of the words in a sentence constitute observations and the tags to assign constitute the latent variables. This model adds a dependency on both layers of a Bi-LSTM component to the tag variables in the CRF. We evaluate the F1 score of the system for the three dataset variants we describe above. We test two different representations as input to the CRF: FastText (our baseline), and **MOE** embeddings. Our results are reported in Table 2.

We make the following observations based on the results of our experiments. Firstly, in the two extreme cases of the 100% misspelled test and correct training and the correct test and 100% misspelled training, **MOE** improves the F1 by 2 and 3.5 points respectively with respect to the FastText baseline. When the test data is 100% misspelled, **MOE** always beats the baseline by up to 2.3 points of F1. Also, in this case the loss in F1 with respect to the case where both the training and the test are

---

[6] `www.facebook.com`

[7] `http://universaldependencies.org/conll17/data.html`

Test Data	100% Misspelled			Original		
Training Data	Original	100% Miss.	10% Miss.	Original	100% Miss.	10% Miss.
FastText, $\alpha = 0.0$	30.47	79.71	65.70	94.33	57.16	94.14
MOE, $\alpha = 0.01$	29.04	80.66	67.94	**94.55**	59.11	94.21
MOE, $\alpha = 0.05$	28.52	**81.17**	**68.92**	94.25	58.95	93.92
MOE, $\alpha = 0.1$	30.94	80.97	67.30	94.45	58.88	**94.29**
MOE, $\alpha = 0.25$	29.00	80.13	67.63	94.37	58.67	94.01
MOE, $\alpha = 0.5$	29.19	80.43	66.76	94.27	57.29	93.94
MOE, $\alpha = 0.75$	30.94	78.65	64.53	94.18	57.67	93.81
MOE, $\alpha = 0.95$	32.40	75.28	62.29	93.09	60.21	92.52
MOE, $\alpha = 0.99$	**32.57**	73.36	61.36	90.91	**60.62**	90.53

Table 2: Performance on POS tagging task for UPOS tags using CRF. The models were trained on 100 epochs with an early stop (small difference on validation error) mechanism enabled. Considering F1 score, we evaluate on 2 variants of test data: Original (correctly spelled) on the right hand side of the table and 100% misspelled on the left hand side.

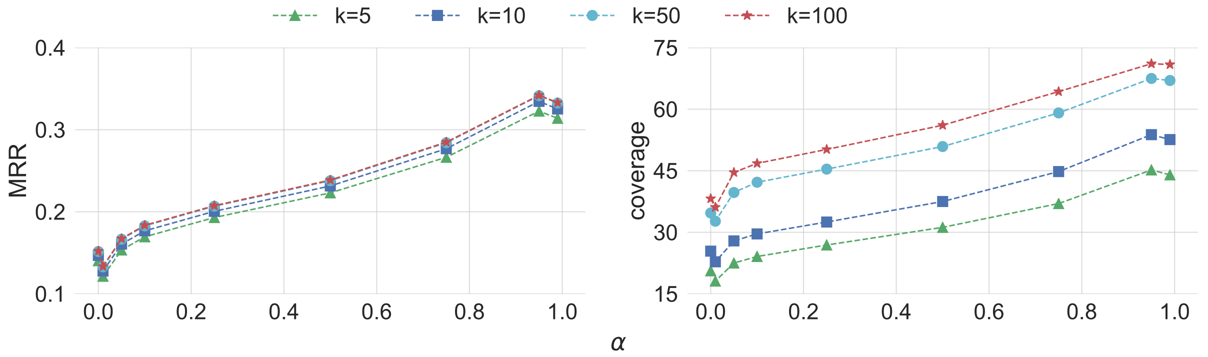


Figure 4: Experimental results for the Neighborhood Validity task. $\alpha = 0$ values represent our baseline, FastText. On the left hand side we present the resulting MRR scores. On the right hand size we present the results for the neighborhood coverage.

correct is much less then when the training data does not contain misspellings. To be remarked is the F1 score difference in the more realistic case consisting of training data that is 10% misspelled. In this case **MOE** attains a sensitive improvement of 2.3% points of F1. Finally, **MOE** does not reduce the effectiveness of the CRF POS Tagger with respect to the FastText baseline when neither the training nor the test set are misspelled. All in all, we have shown that **MOE** does not affect the effectiveness of the POS Tagger in the case of correctly misspelled words and improves sensitively the quality of the POS tagger on misspellings.

6 Conclusion and Future Work

One of the most urgent issues of word embeddings is that they are often unable to deal with malformed words which is a big limitation in the real-world applications. In this work, we proposed a novel model called **MOE**, which aims to solve a long-standing problem: generating high quality, semantically valid embeddings for misspellings.

In the experiments section, in the neighborhood validity task, we show that **MOE** maps embeddings of misspellings close to embedding of the corresponding correctly spelled word. Moreover, we show that **MOE** is performing significantly better than the FastText baseline for the word similarity task when misspellings are involved. For the canonical versions of the word similarity tasks, where misspellings are not involved, we show that **MOE** doesn't worsen the quality significantly for the WS353 dataset and improves over baseline for the RW dataset. In the word analogy task, **MOE** is able to preserve the quality of the semantic analogies similar to the baseline, while improving on the syntactic analogies. In the variant of the test where misspellings are involved, **MOE** outperforms the baseline on both semantic and syntactic questions. Finally, we have shown that **MOE** does not affect the effectiveness of the POS Tagger in the case of correctly spelled words and improves sensitively the quality of the POS tagger on misspellings.

In the future, we will test different ways of training embeddings for misspellings including the extension of the same technique to multi-lingual embeddings. We are going to test deep architectures to combine the n-grams in misspellings to better capture various interdependencies of n-grams and correct versions of words. Finally, we will assess the robustness of both character-based (Kim et al., 2016) and context-dependent embeddings (Devlin et al., 2018), (Peters et al., 2018) with respect to misspellings.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Learning Relational Representations by Analogy using Hierarchical Siamese Networks

Gaetano Rossiello[†], **Alfio Gliozzo**[‡], **Robert Farrell**[‡], **Nicolas Fauceglia**[‡], **Michael Glass**[‡]
[†]Department of Computer Science, University of Bari, Italy
[‡]IBM Research AI, Yorktown Heights, NY, US
`gaetano.rossiello@uniba.it`, `gliozzo@us.ibm.com`,
`robfarr@us.ibm.com`, `nicolas.fauceglia@ibm.com`, `mrglass@us.ibm.com`

Abstract

We address relation extraction as an analogy problem by proposing a novel approach to learn representations of relations expressed by their textual mentions. In our assumption, if two pairs of entities belong to the same relation, then those two pairs are analogous. Following this idea, we collect a large set of analogous pairs by matching triples in knowledge bases with web-scale corpora through distant supervision. We leverage this dataset to train a hierarchical siamese network in order to learn entity-entity embeddings which encode relational information through the different linguistic paraphrasing expressing the same relation. We evaluate our model in a one-shot learning task by showing a promising generalization capability in order to classify unseen relation types, which makes this approach suitable to perform automatic knowledge base population with minimal supervision. Moreover, the model can be used to generate pre-trained embeddings which provide a valuable signal when integrated into an existing neural-based model by outperforming the state-of-the-art methods on a downstream relation extraction task.

1 Introduction

The task of identifying semantic relationships between entities in unstructured textual corpora, namely Relation Extraction (RE), is often a prerequisite for many other natural language understanding tasks, e.g. automatic knowledge base population, question answering, etc. RE is commonly addressed as a classification task (Bunescu et al., 2005), where a model is trained to classify relation mentions in text among a predefined set of relation types. For instance, given the sentence "Robert Plant is the singer of the band Led Zeppelin", an effective RE system might extract the triple `memberOf`(ROBERT PLANT, LED ZEPPELIN), where `memberOf` is a relation label expressed by the linguistic context *"is the singer of the band"*.

Since a given relation can be expressed using different textual patterns surrounding entities, the state-of-the-art RE models which follow this approach need a considerable amount of examples for each relation to reach satisfactory performance. Distant supervision (Mintz et al., 2009) instead uses training examples from a knowledge base, guaranteeing a large amount of (popular) relation examples without human intervention, which can be used effectively by neural networks (Lin et al., 2016; Glass et al., 2018). However, even with this technique, approaching RE as a classification task presents several limitations: (1) distant supervision models are not accurate in extracting relations with a long-tailed distribution, because they typically have a small set of instances in knowledge bases; (2) in most domains, relation types are very specific and only a few examples of each relation are available; (3) these models cannot be applied to recognize new relation types not observed during training.

In this paper, we address RE from a different perspective by reducing it to an analogy problem. Our assumption states that if two pairs of entities, (A, B) and (C, D), have at least one relation in common r, then those two pairs are analogous. Viceversa, solving proportional analogies, such as $A : B = C : D$, consists of identifying the implicit relations shared between two pairs of entities. For example, ROME:ITALY=PARIS:FRANCE is a valid analogy because `capitalOf` is a relation in common.

Based on this idea, we propose an end-to-end neural model able to measure the degree of analogical similarity between two entity pairs, instead of predicting a confidence score for each relation type. An entity pair is represented through its

Proceedings of NAACL-HLT 2019, pages 3235–3245
Minneapolis, Minnesota, June 2 - June 7, 2019.

mentions in a textual corpus, sequences of sentences where entities in the pair co-occur. If a mention represents a specific relation type, then this relationship is expressed by the linguistic context surrounding the two entities. E.g., “Rome is the capital of Italy” or “The capital of France is Paris” referring to the example above. Thus, given two analogous entity pairs represented by their textual mentions sets as input, the model is trained to minimize the difference between the representations of relations having the same linguistic patterns. In other words, the model learns the different paraphrases expressing the same relation. In our research hypothesis, a model trained in such way is able to recognize analogies between unseen entity pairs belonging to new unseen relation types by: (1) generalizing over the sequence of words in the mentions; (2) projecting the sequence of words in the mentions into a vector space representing relational semantics. This approach poses several research questions: (RQ1) How to collect and organize a dataset for training? (RQ2) What kinds of models are effective for this task? (RQ3) How should the model be evaluated?

Knowledge bases, such as Wikidata or DBpedia, consist of large relational data sources organized in the form of triples, `predicate`(SUBJECT, OBJECT). We exploit this information to build a reliable set of analogous facts used as ground truth. Then, we adopt distant supervision to retrieve relation mentions in web-scale textual corpora by matching the subject-object entities which co-occur in the same sentences (Riedel et al., 2010; ElSahar et al., 2018; Glass and Gliozzo, 2018a). Through this technique we can train our model on millions of analogy examples without human supervision.

Since our goal is to train a model able to compute the relational similarity given two sets of textual mentions, we use siamese networks to learn discriminative features between those two instances (Hadsell et al., 2006). This kind of neural network has been used in both computer vision (Koch et al., 2015) and natural language processing (Mueller and Thyagarajan, 2016; Neculoiu et al., 2016) in order to map two similar instances close in a feature space. However, in our setting each instance consists of a set of mentions, therefore it is inherently a multi-instance learning task[1]. We propose a hierarchical siamese network with an attention mechanism at both word level (Yang et al., 2016) and at the set level (Ilse et al., 2018) in order to select the textual mention which better describes the relation. To the best of our knowledge, this is the first application of a siamese network by pairing sets of instances, so it can be considered a novelty of this work.

We evaluate the generalization capability of our model in recognizing unseen relation types through an one-shot relational classification task introduced in this paper. We train the parameters of the model on a subset of most frequent relations of one of three different distantly supervised datasets used in our experiments. Then, we evaluate it on the long-tailed relations of each dataset. During the test phase, only a single example for each unseen relation is provided. This example is not used to update the parameters of the model as in a classification task, but rather to produce the vector representation of the relation itself. Entity pairs having mention sets close to this representation are more likely to be analogous. The experiments show promising results of our approach on this task, compared with the recent deep models commonly used for encoding textual representations (Conneau et al., 2017). However, when the number of the unseen relation types increases, the performance of our model become far from the results obtained in the one-shot image classification (Koch et al., 2015), opening an interesting challenge for future work.

Finally, our model shows a transfer capability in other tasks through the use of its pre-trained vectors. Indeed, a branch of the hierarchical siamese network can be used to generate entity-entity representations given sets of mentions as input, that we call *analogy embeddings*. In our experiments, we integrate those representations into an existing end-to-end model based on convolutional networks (Glass and Gliozzo, 2018b), outperfoming the state-of-the-art systems on two shared datasets commonly used for distantly supervised relation extraction.

2 Related Work

Relation Extraction Several approaches have been proposed in the literature to address the problem of extracting relations from text with minimal supervision.

The bootstrapping method (Agichtein and Gra-

---

[1] Due to the weak supervision, the whole set of mentions is labeled, but each individual mention in the set is unlabeled.

vano, 2000) collects the textual patterns between a few example pairs of entities iteratively, and uses them to retrieve other pairs of entities from a corpus. This method is limited by the semantic drift issue since wrong patterns might be collected.

OpenIE (Mausam et al., 2012) is an unsupervised method for extracting triples from text, where the relations are linguistic phrases. The lack of a canonical form for the extracted relations makes this approach not suitable to populate knowledge bases with a fixed schema.

Universal schema (Riedel et al., 2013) addresses RE by combining the OpenIE and knowledge base relations through a matrix factorization technique typically adopted in the collaborative filtering approach of recommendation systems. The column-less (Toutanova et al., 2015) and row-less (Verga and McCallum, 2016) extensions of this method can handle unseen entity pairs and textual relations when combined (Verga et al., 2017).

The one-shot RE has been addressed by (Yuan et al., 2017), who adopt a siamese network to extract fine-grained relations which typically have few training examples. This model has two main limitations. Firstly, it works only by pairing two single mentions and is not able to handle a whole set of mentions referring to a relation instance. Our hierarchical siamese network overcomes this issue by using an attention mechanism at both word and mention level. Moreover, their one-shot evaluation mainly focuses on extracting the same relation types seen during training. Instead, the goal of our one-shot task is to evaluate the transfer capability in extracting unseen relation types across domains using a single pre-trained model.

Recently, (Levy et al., 2017) propose to reduce RE slot-filling to a question answering problem. The main idea is to build a set of question-answer pairs for the relations in knowledge bases and train a reading comprehension model using this dataset. This approach shows promising zero-shot capability in extracting unseen relation types. However, the schema querification phase requires a crowd-sourcing effort. Our method uses distant supervision, so it does not need any kind of manual annotations.

Word Analogy The analogy problem, from a computational linguistic perspective, was originally addressed by (Turney, 2006) who investigate several similarity measures for solving word analogy questions in the Scholastic Aptitude Test dataset. The authors provide an interesting argument regarding the different types of similarities, attributional and relational, and their use in solving word analogies. Attributional similarity, typical of the word vector space models, is useful for synonym detection, word sense disambiguation and so on. Instead, relational similarity is suitable for understanding analogies between two pairs of words. Our neural-based analogy approach is inspired by this finding.

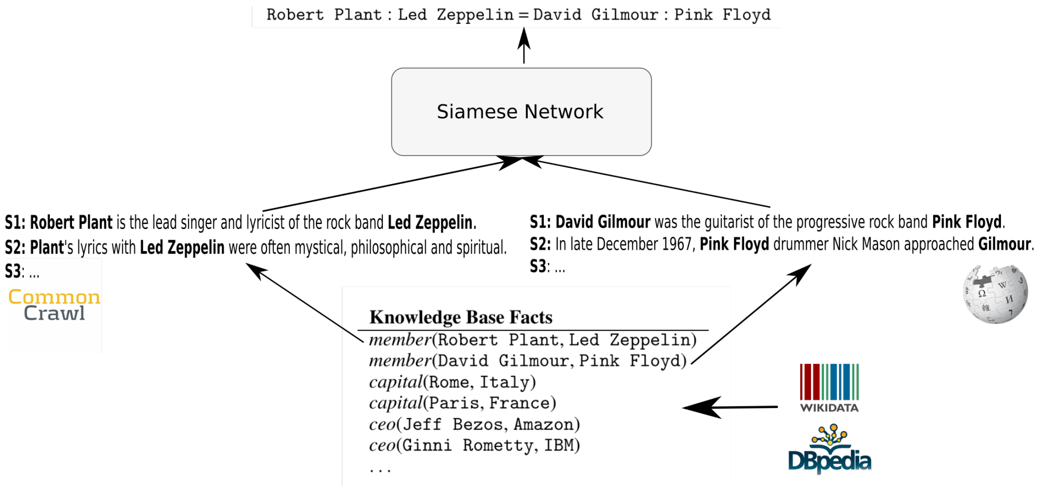
Recently, word analogies, namely the proportional analogy between two word pairs such as $a : b = c : d$, have been used by (Mikolov et al., 2013; Pennington et al., 2014) to show the capability of word embeddings to discover linguistic regularities in word contexts using vector offsets (e.g. $king - man + woman = queen$). The works in (Gladkova et al., 2016; Vylomova et al., 2016) explore the use of word vectors to model the semantic relations. The proportional analogy is also adopted by (Liu et al., 2017) as analogical inference in order to learn multi-relational embeddings which are evaluated on knowledge base completion benchmarks.

However, in order to apply word embedding models to proportional analogy, the model must have seen the words during training. This approach is unsuitable for computing the analogy between out-of-vocabulary words. Our approach overcomes this limitation, since it works by considering the contexts where the entities occur.

3 Learning Relations by Analogy

Given two pairs of entities, (A, B) and (C, D), their semantic relations can be expressed by their mentions in text, $(A, B) = \{S_i\}$ and $(C, D) = \{S_j\}$. Specifically, $\{S_i\}$ and $\{S_j\}$ are the sets of sentences where (A, B) and (C, D) co-occur in the same set. Two pairs of entities are analogous, $A : B = C : D$, iff their mentions sets, or part of them, express the same relation r. Knowledge bases, such as Wikidata, contain millions of trusted facts in form of triples, $r(A, B)$, namely pairs of entities in known relationships. We leverage these relational data sources as ground truth in order to collect a set of proportional analogy statements. Then, we build a dataset through the distant supervision technique by retrieving the mentions sets from web-scale corpora.

Our idea is to train a neural network to solve



S1: Robert Plant is the lead singer and lyricist of the rock band **Led Zeppelin**.
S2: Plant's lyrics with **Led Zeppelin** were often mystical, philosophical and spiritual.
S3: ...

S1: David Gilmour was the guitarist of the progressive rock band **Pink Floyd**.
S2: In late December 1967, **Pink Floyd** drummer Nick Mason approached **Gilmour**.
S3: ...

Knowledge Base Facts
member(Robert Plant, Led Zeppelin)
member(David Gilmour, Pink Floyd)
capital(Rome, Italy)
capital(Paris, France)
ceo(Jeff Bezos, Amazon)
ceo(Ginni Rometty, IBM)
...

Figure 1: Learning relations by analogy through matching the facts from knowledge bases with textual corpora.

the analogy problem between any two entity pairs in this dataset, as long as they are described by the textual contexts where they co-occur. In other words, this task is reduced to a binary classification of determining whether the relational similarity between the representations of two sets of mentions exceeds a threshold. The network is trained by feeding two sets of mentions related to two different pairs, and it is optimized to return a positive label if the two entity pairs are analogous, namely they share at least one relation, or a negative label otherwise.

Figure 1 provides an example of this process. For the relation `memberOf`, the entity pairs (ROBERT PLANT, LED ZEPPELIN) and (DAVID GILMOUR , PINK FLOYD) are sampled. The two entity pairs are converted into their respectively mentions sets gathered from a textual corpus, such as Wikipedia. Since these two entity pairs are analogous, the network is optimized to learn the representations of the two textual contexts to be close into the feature space. In fact, the first sentences of both pairs represent the concept of membership of a band, even if they are expressed using different words. Based on our assumption, the aim is to learn how to encode the relational representations through the different paraphrases of the same relation. However, the model also needs negative examples during the training phase. We randomly select an entity pair from a different relation for each positive example, such as (ROBERT PLANT, LED ZEPPELIN) and (PARIS, FRANCE). Since we cast the problem as a binary classification task, we create a balanced dataset of positive and negative examples.

Siamese neural networks (Bromley et al., 1993) are well suited to this task because they are specifically designed to compute the similarity between two instances. A siamese network has symmetric twin sub-networks which share the same parameters, but are joined by an energy function at the head. Weight sharing forces the two similar instances to be mapped to very close locations in feature space because both of the sub-networks are optimized using the same function. In computer vision, siamese architectures based on convolutional neural networks (Hadsell et al., 2006; Koch et al., 2015; Vinyals et al., 2016) have shown promising performance in learning highly discriminative features by pairing images that belong to the same class. Likewise, our hypothesis is that a siamese network trained by matching two distinct mentions sets that share the same relation is able to learn how to map patterns of words across the sentences containing the two pairs of entities so as to capture the semantics of the relation. For instance, given the example in Figure 1 an effective siamese network should determine that the patterns for *"is the lead singer"* and *"was the guitarist"* express the same relation, `memberOf`.

To train a siamese network based on our approach, we have to face the following challenges: (1) the language may be highly variable and the same relation expressed in a multitude of different ways; (2) the mentions set of an entity pair consists of several sentences, each of which might express different relations, hence this is a multi-instance learning problem; (3) distant supervision could provide a wrong labeling, namely sentences which do not express any specific relations.

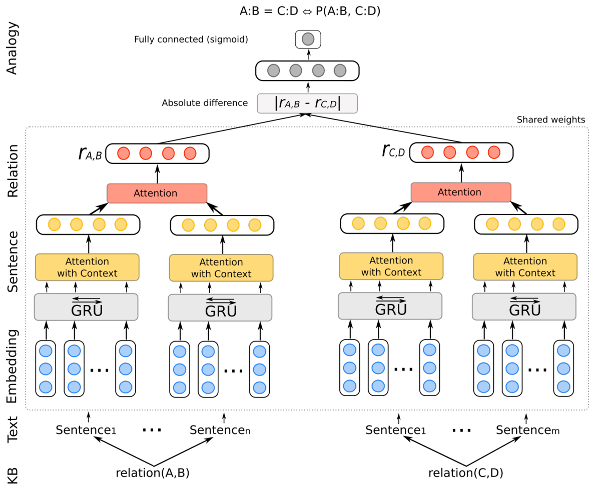


Figure 2: Hierarchical siamese network.

4 Hierarchical Siamese Network

To face these challenges, we propose a Hierarchical Siamese Network (HSN) architecture as shown in Figure 2. In the following paragraphs, we describe the details of each component.

Input Representation The HSN takes as input two entity pairs represented by their mentions sets. Since the twin sub-networks of the HSN are the same, we focus only on one of these. Given a triple $r(A, B)$ from a knowledge base, the relation r can be expressed through the set of sentences in a textual corpus where the the two entities co-occur: $r(A, B) = \{S_1, S_2, \ldots, S_n\}$, with $S_i = \{w_{i1}, w_{i2}, \ldots, w_{ik}\}$, where w_{ij} represents the j-th word in the sentences S_i, $\forall i \in 1 \leq i \leq n$ and $\forall j \in 1 \leq i \leq k$. The purpose of a sub-network is to learn a low-rank vector representation $r_{A,B}$ for the relation r expressed by the pair (A, B). This is done by hierarchically composing the word and sentence representations.

Gated Recurrent Unit for Sentence Encoder Given a sentence $S_i = \{w_{i1}, w_{i2}, \ldots, w_{ik}\}$, we map each one-hot word representation of w_{ij} into its word embedding $x_{ij} = Ew_{ij}$, where $E^{d,|V|}$ is a matrix of real-valued vectors of size d, and V is a (fixed) vocabulary. Word embeddings are designed to encode syntactic and semantic features of words and can be randomly initialized or pre-trained on large corpora. We use pre-trained GloVe (Pennington et al., 2014) embeddings for our purposes. We encode the whole sentence S_i into a low-rank representation by composing its constituent word embeddings. An effective way to perform such an encoding is using recurrent neural networks (RNN) which are able to compose word embeddings by taking into account their positions in the sentence conditioned on the previous words. For our model, this capability is critical in order to detect sequences of words which express a particular relation, such as *"is the capital of"*. We use a bidirectional GRU (Bahdanau et al., 2014) to gather the information from both directions for words. Formally, given $\overrightarrow{h_{ij}} = \overrightarrow{GRU}(x_{ij})$ and $\overleftarrow{h_{ij}} = \overleftarrow{GRU}(x_{ij})$, the hidden state $h_{ij} = [\overrightarrow{h_{ij}}, \overleftarrow{h_{ij}}]$ is a new dense representation of w_{ij} which encodes also the information of the whole sentence.

Word Attention with Context Vector However, only certain words in a sentence express the semantics of a relation, therefore we need a strategy to automatically identify them during the training. For example, the words *"singer"* and *"guitarist"* at both sides of Figure 1 are good candidates to express the relation `member`. We use the attention mechanism with a context vector proposed in (Yang et al., 2016) to reward such words which are important to the meaning of a relation and then aggregate their information in the sentence representation. In detail, $s_i = \sum_k \alpha_{ik} h_{ik}$, where $\alpha_{ij} = \frac{exp(u_{ij}^T u_w)}{\sum_k exp(u_{ik}^T u_w)}$, and $u_{ij} = tanh(W_w h_{ij} + b_w)$. The vector s_i represents the sentence S_i and is computed as the weighted sum of the GRU-based word vectors h_{ij} using the normalized attention weights α_{ij}. The parameters for the attention mechanism are the weights and biases W_w, b_w and u_w, the context vector, a global fixed vector which, independently from a specific word, represents a kind of query which helps to inform what is the most informative word for each analogy. The context vector u_w essentially works like a memory mechanism, as described in (Sukhbaatar et al., 2015; Kumar et al., 2016).

Attention for Multi-instance Relation Representation Once all sentences in the mentions set are encoded, the aim of the last layer of a sub-network is to produce the vector $r_{A,B}$ which represents the pair (A, B). However, while weak supervision guarantees a large amount of training data without any human intervention, wrongly labeled sentences inevitably occur. For instance, the S2 of the pair on the right side in the Figure 1 does not express the relation `member` precisely, therefore a wrong bias could propagate during the training phase. This issue is typically addressed through a multi-instance setting, where a model should identify the correct instance(s) from a bag. Recently, end-to-end neural architectures have been proposed to address this multi-instance classification problem (Wang et al., 2018; Feng and Zhou, 2017) by proposing several ways to aggregate the unlabeled instances, such as taking their average. Our goal is to have a model which is able to properly select the most relevant sentences by ascribing different weights to the encoded sentences. For this purpose, we adopt an attention mechanism at the sentence level. It is important to point out that the sentences in the mentions set do not have any temporal relationship, therefore we adapt the standard attention strategies as described in (Ilse et al., 2018). In detail, $r_{A,B} = \sum_i \alpha_i s_i$ is the embedding of the relation r given the pair (A, B), with $\alpha_i = \frac{exp(u_i^T u_s)}{\sum_k exp(u_k^T u_s)}$, $u_i = tanh(W_s s_i + b_s)$, where W_s and u_s are parameters.

Merging Layer and Training Strategy There are several ways to merge the output of the two sub-networks in order to learn the analogical similarity between them. For instance, (Hadsell et al., 2006) propose a contrastive loss with the aim of decreasing the distance between two instance representations. However, we adopt the strategy proposed in (Koch et al., 2015), in which the metric distance is induced by a fully-connected layer with a sigmoidal output unit on the absolute difference between the representations output by twin networks. Thus, given $r_{A,B}$ and $r_{C,D}$ the two relation embeddings which encode the whole mentions sets of the two entity pairs, we can compute the degree of analogy between them with $p = \sigma(W_r(|r_{A,B} - r_{C,D}|))$, where the parameters W_r measure the importance of each element of the difference vector, and they are learned in a end-to-end fashion, together with the relation representations. We build a training set by pairing the mentions sets of the entity-entity pairs from a KB, following the idea discussed in the next section. Thus, we can reduce the analogy task to a binary classification problem, so that $p = P((A, B), (C, D); \Theta)$ is equal to 1 if $A : B = C : D$, 0 otherwise. We learn Θ (all the parameters of HSN) using a gradient-based method which minimizes a cross-entropy loss function with the L2 regularization.

	NYT-FB	**CC-DBP**	**T-REX**
Knowledge Base	Freebase	DBpedia	Wikidata
Corpus	New York Times	Common Crawl	Wikipedia
# words	239,877	8,445,417	4,062,498
# entity pairs	375,846	6,876,913	6,413,452
# relations	57	298	685
avg. mentions	1.9	3.8	3.2
avg. sent. length	41	37	25

Table 1: Statistics of the distantly supervised datasets.

5 Experiments

Once the analogy model is trained, it has two different capabilities. First, the whole HSN architecture can be used as a binary classifier in order to infer if two entity pairs, expressed by their mentions sets, are analogous. Second, we can use its subnetwork before the merge layer as a feature extractor to generate entity-entity vectors given sets of sentences as input which can be used as pretrained analogy embeddings in other tasks. We evaluate our model on the one-shot relational classification and distantly supervised relation extraction benchmarks.

5.1 Datasets

In the entire experimental protocol we exploit three different datasets (see the supplemental material for details). **T-REX** (ElSahar et al., 2018) is a large scale alignment dataset between Wikipedia abstracts and Wikidata triples, having 685 unique relations. **NYT-FB** (Riedel et al., 2010) is a standard benchmark for distantly supervised relation extraction. The text of New York Times was processed with a named entity recognizer and the identified entities linked by name to Freebase. **CC-DBP** (Glass and Gliozzo, 2018a) is a web-scale KB population benchmark. It combines the text of Common Crawl with the entity-relation-entity triples from 298 frequent relations in DBpedia. Mentions of DBpedia entities are located in text by matching the preferred label. This task is similar to NYT-FB, but it has a much larger number of relations, triples and textual contexts. The statistics of the three datasets are summarized in Table 1. Aside from the difference in size and KB adopted, it worth noting also the difference in terms of corpus style of these datasets. For instance, T-REX has well-written textual mentions, because the sentences are extracted from Wikipedia. Conversely, CC-DBP and NYT-FB contain dirtier sentences which mean a high probability of incurring wrong labeling.

5.2 Training and Implementation Details

For both benchmarks, we use the same analogy model trained only once on a subset of the relations in T-REX. In detail, we discard all relations having less than 20 entity pairs, collecting 482 relations. We sort the relations by the number of instances, and we took the most frequent 60% of them to train the HSN. We use the remaining 20% of the relations for validation and the least frequent 20% as a corpus to implement one of the three one-shot classification tasks. For the validation and test partitions we randomly select only 20 entity pairs for each relation. This becomes a useful test set for the one-shot validation. To train the HSN, we select a balanced number of positive and negative examples out of the training split based on these rules: (1) for each relation, we randomly extract a set of 20 entity pairs; (2) out of this set, we generated all possibile combinations, $\binom{20}{2} = 190$, as positive pairing examples; (3) for each combination, we create a negative example by randomly selecting an entity pair from another relation. After these steps, we collect a bucket of 109,820 proportional analogy training examples.

We iterate this process throughout the training phase by selecting a different buckets at each iteration to prevent overfitting. The training is monitored by computing the binary accuracy over a fixed validation set, consisting of 36,480 analogy examples, built by adopting the same criteria described above. We initialize our word embedding layer with the pre-trained GloVe vectors consisting of 6B tokens with 50 dimensions. The word embedding weights are not updated during training. The number of mentions for each entity pair is fixed to 3, based on their average on T-REX (see Table 1).

5.3 One-shot Relational Classification

Task Given an unseen entity pair (A^t, B^t) and its mentions set $\langle A^t, B^t \rangle$, the one-shot relation classification task is to categorize this test pair (A^t, B^t) into one of N relation types, with the restriction that for each relation type $r_i, \forall i \in N$, we are given only one entity pair (A^i, B^i) together with its mentions set $\langle A^i, B^i \rangle$ as training. We can cast the one-shot classification in terms of a relational similarity as follows:

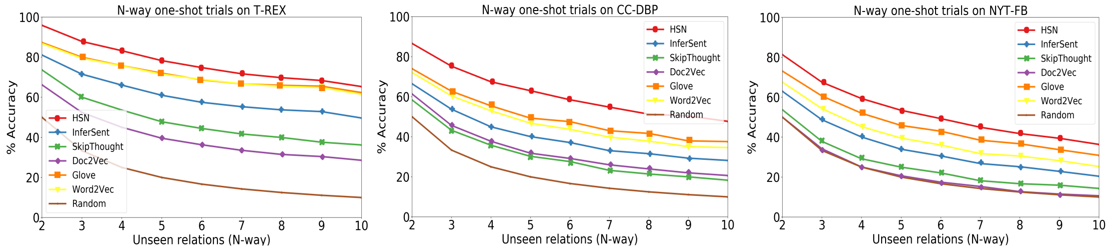
$$r_i = \arg\max_i sim_M(\langle A^t, B^t \rangle, \langle A^i, B^i \rangle) \quad (1)$$

where sim_M is a similarity score, using the method M, which measures the analogy between the train and test entity pairs through their mentions sets. We implemented sim_{HSN} using the HSN trained as described above. The two mentions sets are given as input to the network and their similarity is computed using the sigmoidal output of the last layer.

Baselines A method M should be robust in facing the *unseen* entity pairs used for testing. Training a standard RE model using just one example cannot provide a suitable baseline. Furthermore, since the two new entities that we want to classify might not be present in an existing knowledge graph, we could not apply relational embeddings (Bordes et al., 2013) as well. Thus, the use of the contexts surrounding the two entities in the mentions sets to compute the relational similarity score is needed. In other words, we cast the task of one-shot relational classification to a problem of measuring textual (i.e. mentions) similarity with the aim to prove that our pre-trained siamese model is able to grasp the semantics of relations better than the other pre-trained text representation models.

We implemented five baselines commonly used to encode textual representations. First, we use the pre-trained Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) embeddings. The score is given by the cosine similarity between the bag-of-means for the two entity pairs, averaging the word vectors in the mentions sets. We also adopt Doc2Vec (Le and Mikolov, 2014) to derive entity pair vectors, and comparing them using cosine similarity. For each entity pair, a pseudo-document embedding is created by concatenating its mentions sets. Finally, we compare HSM with the pre-trained Skip-Thought (Kiros et al., 2015) and InferSent (Conneau et al., 2017) sentence encoders, which are the state-of-the-art in computing textual similarity. An entity pair vector is obtained by averaging the embeddings of each sentence in the mentions set.

One-shot trials We follow the experimental setup described in (Koch et al., 2015) to create our one-shot benchmark. For each dataset, we select the 20% of less frequent relations sorted by the number of entity pairs, having at least 20 instances. Therefore, we collect three different one-shot test sets of 92, 55 and 11 unseen relation types for T-REX, CC-DBP and NYT-FB,



(a) T-REX (b) CC-DBP (c) NYT-FB

Figure 3: One-shot relational classification results for N-way unseen relation types.

respectively. The reason behind this criteria is to prevent the overlap of the semantic relation types between the train data and the different one-shot test sets. In fact, frequent relations, such as `location` or `birthPlace`, are common in all three datasets, and they are used to train our HSN. Moreover, using the long-tailed relations, such as `portOfRegistry`, we emulate a scenario where only a small set of relation examples are available, making more challenging the task. To evaluate the one-shot capabilities on N-way classes: (1) N different relation types are selected; (2) we sample one(shot) entity pair example for each of the selected N relation types; (3) we choose another entity pair used as test example from one of the N relation types. All the selections in these three steps are random. If the relation type returned by the Eq. 1 is equal to the relation type of the selected test example, then the one-shot trial is correct, otherwise it is incorrect. We repeated this operation k times for N from 2 to 10, for each of the three datasets. We choose k equal to 10,000, so that the random baseline converges to $100/N$, in order to create an unbiased testbed.

Results and Discussion The results are reported in Figure 3. Our model outperforms all the baselines on the test split of T-REX, reaching an accuracy range from 95.87% to 65.33% for N-way one-shot trials. This behavior remains constant also for the other two datasets, showing the solidity of HSN even though it has been trained on a different corpus using relations from an another ontology. This result confirms that our model is able to generalize on the linguistic contexts expressing relations, as well as the capability to learn how to transfer this information to other relations not observed before. The supplemental file reports some one-shot trial examples.

The lower accuracy on CC-DBP and NYT-FB might be caused by the different style of the corpora (Wikipedia vs. Web pages). Indeed, the test set of T-REX is build using the same corpus which HSN is trained on. The Wikipedia abstracts consist of well-written contents, typically the definition of one of the two entities in the pairs. Thus, T-REX can be considered an easier dataset compared with the other two.

Surprisingly, the average vectors using Word2Vec and GloVe obtain remarkable performance compared to state-of-the-art sentence encoders. This might be due to the way how these sentence models are trained. For instance, InferSent is trained using a natural language inference dataset, which might be not suitable to learn representations which represent relations in text. Instead, HSN is trained and optimized to learn and encode relational representations. However, this aspect deserves to be dealt with more deeply, as does the comparison of HSN on the shared textual similarity benchmarks; we think this is a clear path for future research.

5.4 Transfer Learning in Relation Extraction

We also evaluate the ability of the analogy model to provide low-rank representations for entity pairs which are useful for more traditional relation extraction tasks, where a corpus of text has to be processed and relevant relations in a predefined schema have to be recognized.

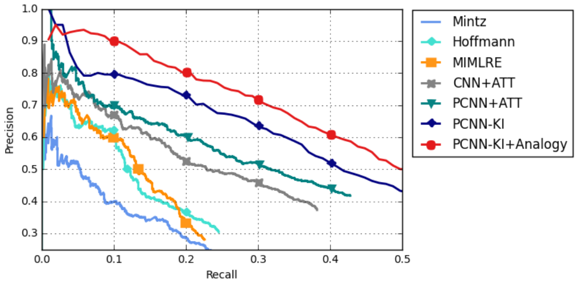


Figure 4: Precision-Recall curves on NYT-FB.

Relation / Score	Entity pair / Best mention	Entity pair / Best mention
doctoralStudent	Victor Weisskopf : Murray Gell-Mann	John Bardeen : Nick Holonyak
0.95	**Murray Gell-Mann**, one of the principal discoverers of the quarks, is one of the distinguished pupils of **Victor Weisskopf**.	Professor **Nick Holonyak** jr. was the first phd student of Nobel Prize winner **John Bardeen**.
approvedBy	Hundred Horse Chestnut : Guinness World Records	NCSA Open Source License : Open Source Initiative
0.83	**Guinness World Records** has listed **Hundred Horse Chestnut** for the record of "greatest tree girth eve".	**NCSA** was formally certified as an open-source license during a March 28, 2002 board meeting of the **Open Source Initiative**.
architecturalStyle	Rockefeller Center : Art Deco	St. Mark Basilica : Byzantine architecture
0.63	**Art Deco** mural "wisdom" hangs over the entrance to the **Rockefeller Center** and was designed and sculpted by artist Lee Lawrie.	**St. Mark's Basilica**, the cathedral of Venice, is one of the best known examples of **Byzantine architecture**.

Table 2: Three examples of relational similarity between two pairs of entities computed by our HSN. For each example, we report the unseen relation type, the mentions related to each pair, and the similarity score. We report only the mention having the highest attention weight. The examples show the ability of the analogy model in providing a high score to two mentions which represent the same relation, even if they are expressed using different textual contexts.

To this aim, we use the sub-network of our HSN before the merge layer, and we feed the mentions set of each entity pair of instances as found in the corpus to generate an analogy embedding as a vector of features. In detail, given a set of mentions referring to an entity pair (A, B) as input, the pre-trained HSN generates an embedding $r_{A,B}$ (see Figure 2) which represents the relation between those two entities. Then, we concatenate these embeddings to the penultimate layer of a relation extraction model, PCNN-KI (Glass and Gliozzo, 2018b), based on a convolutional neural network, which is the state-of-the-art for this benchmark. The final fully-connected layer uses the representation from HSN in combination with its own learned multi-instanced vector representation to predict a confidence score for each relation. During the training of this joint model, PCNN-KI+Analogy, we freeze our analogy embeddings in order to avoid the loose the *knowledge transfer* capability.

As for the one-shot setting described before, we use the same pre-trained the HSN on the T-REX and we used it as a feature extractor for entity pairs in both train-test standard splits of NYT-FB, as used in (Zeng et al., 2015; Lin et al., 2016). Figure 4 reports the results of our evaluation. The model which uses the features generated by HSN largely improve the performances of PCNN-KI, despite the HSN is trained on a different corpus and using a different KB. In the same chart, we also report a compared evaluation for other approaches proposed in the literature for the NYT-FB benchmark: PCNN+ATT (Lin et al., 2016), CNN+ATT (Zeng et al., 2015), MIML-RE (Surdeanu et al., 2012), Hoffmann (Hoffmann et al., 2011), Mintz (Mintz et al., 2009).

We run the evaluation also on CC-DBP, a larger dataset for distantly supervised RE, using the same train-test setting adopted in (Glass and Gliozzo, 2018b). As done for the NYT-FB dataset, we incorporate the analogy embeddings generated by the same HSN trained on the T-REX. The results confirm the improvements obtained by PCNN-KI model if it integrates our pre-trained embeddings (Table 3).

	AUC	F1
PCNN-KI	0.437	0.468
PCNN-KI+Analogy	**0.500**	**0.504**

Table 3: AUC and F1 results on CC-DBP.

6 Conclusion and Future Work

In this paper, we proposed a novel approach to learn representations of relations in text. Alignments between knowledge bases and textual corpora are used as ground truth in order to collect a set of analogies between entity pairs. We designed a hierarchical siamese network trained to recognize those analogies. The experiments showed the two main advantages of our approach. First, the model can generalize on new unseen relation types, obtaining promising results in one-shot learning compared with the state-of-the-art sentence encoders. Second, the model can generate low-rank representations can help existing neural-based models designed for other tasks. As future work, we plan to continue our investigation by extending the method with other ideas. For instance, the use of positional embeddings, as well as the use of placeholders replacing the entities in the textual mentions are promising future directions. Finally, we plan also to explore the use of analogy embeddings in other tasks, such as question answering and knowledge base population.

References

Eugene Agichtein and Luis Gravano. 2000. *Snowball*: extracting relations from large plain-text collections. In *ACM DL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *NIPS*.

Razvan C. Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *LREC*.

Ji Feng and Zhi-Hua Zhou. 2017. Deep MIML network. In *AAAI*.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *SRW@HLT-NAACL*.

Michael Glass and Alfio Gliozzo. 2018a. A dataset for web-scale knowledge base population. In *ESWC*.

Michael Glass and Alfio Gliozzo. 2018b. Discovering implicit knowledge with unary relations. In *ACL*.

Michael Glass, Alfio Gliozzo, Oktie Hassanzadeh, Nandana Mihindukulasooriya, and Gaetano Rossiello. 2018. Inducing implicit relations from text using distantly supervised deep nets. In *ISWC*.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *HLT*.

Maximilian Ilse, Jakub M. Tomczak, and Max Welling. 2018. Attention-based deep multiple instance learning. *CoRR*, abs/1802.04712.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *ICML*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Rep4NLP@ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML PKDD*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3).

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *AKBC@NAACL-HLT*.

Patrick Verga, Andrew McCallum, and Arvind Neelakantan. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *EACL*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *NIPS*.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *ACL*.

Xinggang Wang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu. 2018. Revisiting multiple instance neural networks. *Pattern Recognition*, 74.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*.

Jianbo Yuan, Han Guo, Zhiwei Jin, Hongxia Jin, Xianchao Zhang, and Jiebo Luo. 2017. One-shot learning for fine-grained relation extraction via convolutional siamese neural network. In *BigData*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

An Effective Label Noise Model for DNN Text Classification

Ishan Jindal[1], Daniel Pressel[2], Brian Lester[2], and Matthew Nokleby[1]
[1]Wayne State University, Detroit MI 48202
[2]Interactions Digital Roots, Ann Arbor MI 48104
{ishan.jindal, matthew.nokleby}@wayne.edu,
{dpressel, blester}@interactions.com

Abstract

Because large, human-annotated datasets suffer from labeling errors, it is crucial to be able to train deep neural networks in the presence of label noise. While training image classification models with label noise have received much attention, training text classification models have not. In this paper, we propose an approach to training deep networks that is robust to label noise. This approach introduces a non-linear processing layer (*noise model*) that models the statistics of the label noise into a convolutional neural network (CNN) architecture. The noise model and the CNN weights are learned jointly from noisy training data, which prevents the model from overfitting to erroneous labels. Through extensive experiments on several text classification datasets, we show that this approach enables the CNN to learn better sentence representations and is robust even to extreme label noise. We find that proper initialization and regularization of this noise model is critical. Further, by contrast to results focusing on large batch sizes for mitigating label noise for image classification, we find that altering the batch size does not have much effect on classification performance.

1 Introduction

Deep Neural Networks (DNNs) have led to significant advances in the fields of computer vision (He et al., 2016), speech processing (Graves et al., 2013) and natural language processing (Kim, 2014; Young et al., 2018; Devlin et al., 2018). To be effective, supervised DNNs rely on large amounts of carefully labeled training data. However, it is not always realistic to assume that example labels are clean. Humans make mistakes and, depending on the complexity of the task, there may be disagreement even among expert labelers. Further, samples drawn from the class conditional densities with overlapping supports gives rise to the label noise in training datasets. To support noisy labels in data, we need new training methods that can be used to train DNNs directly from the corrupted labels to significantly reduce human labeling efforts. Zhu and Wu (2004) perform an extensive study on the effect of label noise on classification performance of a classifier and find that noise in input features is less important than noise in training labels.

In this work, we add a *noise model* layer on top of our target model to account for label noise in the training set, following (Jindal et al., 2016; Sukhbaatar et al., 2014). We provide extensive experiments on several text classification datasets with artificially injected label noise. We study the effect of two different types of label noise; *Uniform label flipping (Uni)*, where a clean label is swapped with another label sampled uniformly at random; and *Random label flipping (Rand)* where a clean label is swapped with another label from the given number of labels sampled randomly over a unit simplex.

We also study the effect of different initialization, regularization, and batch sizes when training with noisy labels. We observe that proper initialization and regularization helps the noise model learn to be robust to even extreme amounts of noise. Finally, we use low-dimensional projections of the features of the training examples to understand the effectiveness of the noise model.

The rest of the paper is organized as follows. Section 2 discusses the various approaches in literature to handle label noise. In Section 3, we describe the problem statement along with the proposed approach. We describe the experimental setup and datasets in Section 4. We empirically evaluate the performance of the proposed approach along with the discussion in Section 5 and finally conclude our work in Section 6.

Proceedings of NAACL-HLT 2019, pages 3246–3256
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related Work

Learning from label noise is a widely studied problem in the classical machine learning setting. Earlier works (Brodley and Friedl, 1999; Rebbapragada and Brodley, 2007; Manwani and Sastry, 2013) consider learning from noisy labels for a wide range of classifiers including SVMs (Natarajan et al., 2013) and fisher discriminants (Lawrence, 2001). Traditional approaches handle label noise by detecting and eliminating the corrupted labels. More details about these approaches can be found in (Frénay and Verleysen, 2014).

Recently, DNNs have made huge gains in performance over traditional methods on large datasets with very clean labels. However large real-world datasets often contain label errors. A number of works have attempted to address this problem of learning from corrupted labels for DNNs. These approaches can be divided into two categories; attempts to mitigate the effect of label noise using auxiliary clean data, and attempts to learn directly from the noisy labels.

Presence of auxiliary clean data: This line of research exploits a small, clean dataset to correct the corrupted labels. For instance, Li et al. (2017) learn a teacher network with clean data to re-weight a noisy label with a soft label in the loss function. Similarly, Veit et al. (2017) use the clean data as a label correction network. One can use this auxiliary source of information to do inference over latent clean labels (Vahdat, 2017). Further, Yao et al. (2018) models the auxiliary trustworthiness of noisy image labels to alleviate the effect of label noise. Though these methods show very promising results, the absence of clean data in some situations might hinder the applicability of these methods.

Learning directly from noisy labels: This research directly learns from the noisy labels by designing a robust loss function, or by modeling the latent labels. For instance, Reed et al. (2014), apply bootstrapping to the loss function to have consistent label prediction for similar images. Similarly, Joulin et al. (2016) alleviate the label noise effect by adequately weighting the loss function using the sample number. Jiang et al. (2017) propose a sequential meta-learning model that takes in a sequence of loss values and outputs the weights for the labels. Ghosh et al. (2017) further explores the conditions on loss functions such that the loss function is noise tolerant.

A number of approaches learn the transition from latent labels to the noisy labels. For example, Mnih and Hinton (2012) propose a noise adaptation framework for symmetric label noise. Based on this work, several other works (Sukhbaatar et al., 2014; Jindal et al., 2016; Patrini et al., 2017; Han et al., 2018) account for the label noise by learning a noisy layer on top of a DNN where the learned transition matrix represents the label flip probabilities. Similarly, Xiao et al. (2015) propose a probabilistic image conditioned noise model. Azadi et al. (2015) proposed an image regularization technique to detect and discard the noisy labeled images. Other approaches include building two parallel classifiers (Misra et al., 2016) where one classifier deals with image recognition and the other classifier models humans reporting bias.

All of these approaches have targeted image classification. In this work, we propose a framework for learning from noisy labels for text classification using a DNN architecture. Similar to (Sukhbaatar et al., 2014; Jindal et al., 2016; Patrini et al., 2017), we append a non-linear processing layer on top of this architecture to model the label noise. This layer helps the base architecture to learn better representations, even in the presence of label noise. We empirically show that, for better classification performance, the knowledge of noise transition matrix is not needed. Instead, the process forces the DNN to learn better sentence representations.

3 Problem Statement

In a supervised text classification setting where $\mathbf{x}_i \in \mathbf{R}^d$ is a d-dimensional word embedding of the ith word in a sentence of length l (padded wherever necessary), we represent the sample as a temporal embedding matrix $\mathbf{X} \in \mathbf{R}^{d\times l}$ which belongs to one of the K classes. Let the noise-free training set be denoted by

$$\mathcal{D} = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \cdots, (\mathbf{X}_n, y_n)\},$$

where $y_i \in \{1, \ldots, K\}$ represents the category of ith sample, n is the total number of training samples, and there is an unknown joint distribution $p(\mathbf{X}, y)$ on the sample/label pairs. This temporal representation of a sample $\mathbf{X}$ is fed as input to a classifier on the training set $\mathcal{D}$ with sample categories y. However, as mentioned in Section 2, we cannot access the true noise-free samples labels and instead, observe noisy labels corrupted by an unknown noise distribution. Let this *noisy* training set be denoted by

$$\mathcal{D}' = \{(\mathbf{X}_1, y_1'), (\mathbf{X}_2, y_2'), \cdots, (\mathbf{X}_n, y_n')\},$$

where y_i' represents the corrupted label for the sentence $\mathbf{X}_i$. In this work, we suppose the label noise is *class-conditional*, where the noisy label y_i' depends only on the true label y_i, but not on the input $\mathbf{X}_i$ or any other labels y_j or y_j'. Under this model, the label noise is characterized by the conditional distribution

$$p(y' = i | y = j) = \Phi_{ij},$$

which we describe via the $K \times K$ column-stochastic matrix Φ_{ij}, parameterized by a matrix $\mathbf{Q} = \{\Phi_{ij}\}$.

In our experiments, we artificially inject label noise into the training and validation sets. We fix the noise distribution Φ_{ij} and, for a training sample, we generate a noisy label by drawing i.i.d from this noise distribution Φ_{ij}. However, we do not alter the test labels.

Though the proposed approach works for any noise distribution, for this study, we focus on two different types of label flip distributions. We use a noise model parameterized by the overall probability of a label error, denoted by $0 \leq p \leq 1$. For a noise level p, we set the noise distribution matrix

$$\Phi = (1-p)\mathbf{I} + \frac{p}{K}\mathbf{II}^K, \tag{1}$$

and we call it *Uniform label flip noise model*. Here, $\mathbf{I}$ represents the identity matrix and $\mathbf{II}$ denotes the all-ones matrix. Similarly, we describe the *random label flip noise model* as

$$\Phi = (1-p)\mathbf{I} + p\Delta, \tag{2}$$

where $\mathbf{I}$ is the identity matrix, and Δ is a matrix with zeros along the diagonal and remaining entries of each column are drawn uniformly and independently from the $K-1$-dimensional unit simplex. The label error probability for each class is p, while the probability distribution *within* the erroneous classes is drawn uniformly at random.

Our objective is to train a classifier on the noisy labeled sample categories on the training set $\mathcal{D}'$ such that it jointly makes accurate predictions of the true label y and learns the noise transition matrix simultaneously, given $\mathbf{X}$. For the noisy dataset $\mathcal{D}'$, it is straightforward to train a classifier that predicts the *noisy* labels using conditional distribution for the noisy labeled input sentence $\mathbf{X}$:

$$p(y' = \hat{y}'|\mathbf{X}) = \sum_i \Big(p(y' = \hat{y}'|y = \hat{y}_i) \\ p(y = \hat{y}_i|\mathbf{X})\Big). \tag{3}$$

One can learn the classifier associated with $p(y' = \hat{y}'|x)$ via standard training on the noisy set $\mathcal{D}'$. To predict the clean labels by learning the conditional distribution $p(y = \hat{y}_i|x)$ requires more effort, as we cannot extract the "clean" classifier from the noisy classifier when the label noise distribution is unknown.

3.1 Proposed Framework

We refer to the DNN model without the final layer as the *base model* or network *without noise model (WoNM)*. This model, along with the non-linear layer, is trained via back-propagation on the noisy training dataset. The non-linear processing layer in the *noise model* transforms the base model outputs to match the noisy labels during the forward pass better and presents the denoised labels to the base model during the backward pass. The noise layer is parameterized by a square matrix $\Psi \in \mathbf{R}^{K\times K}$). At test time, we remove this learned noise model and use the output of the base model as final predictions.

We refer to the base model parameters as Θ. The base model outputs a probability distribution over the number of K categories denoted as $p\left(y = \hat{y}_i|\mathbf{X};\Theta\right) \forall i \in \{1, 2, \cdots, K\}$. During the forward pass the noise model transforms this output to obtain the noisy labels as

$$p\left(y'|\mathbf{X};\Theta,\Psi\right) = \sigma\left(\Psi \times p\left(y|\mathbf{X};\Theta\right)\right), \tag{4}$$

where $\sigma(\cdot)$ represents the usual softmax operator. Note that both the equations (3) and (4) compute the probability distribution over noisy labels – our noise model does not learn a noise transition matrix. However, we assert that the knowledge of exact noise statistics is neither necessary nor sufficient for the better prediction results.

We learn the base model parameters Θ and the noise model parameters Ψ by maximizing the log likelihood (4) over all of the training samples, minimizing the *cross-entropy* loss:

$$\begin{aligned}\mathcal{L}(\Theta, \Psi; \mathcal{D}') &= -\frac{1}{n}\sum_{i=1}^{n}\log\left[p\left(y'|\mathbf{X}_i;\Theta,\Psi\right)\right] \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\left[\sigma\left(\Psi \times p\left(y|\mathbf{X}_i;\Theta\right)\right)\right]_{y_i}\end{aligned} \quad (5)$$

Similar to (Sukhbaatar et al., 2014), we initialize the noise model weights to the identity matrix. Since DNNs have high capacity, we may encounter the situation when the base model itself absorbs all the label noise and, thus, the noise model does not learn anything at all. In order to avoid this situation, and to prevent overfitting, we apply l_2 regularization to the noise model. However, we want the noise model to overfit the label noise. In the experiment section, we observe that with proper regularization and weight initialization the noise model absorbs most of the label noise. Finally, we train the entire network according to the following loss function:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\log\left[\sigma\left(\Psi \times p\left(y|\mathbf{X}_i;\Theta\right)\right)\right]_{y_i} + \frac{1}{2}\lambda||\Psi||_2^2. \quad (6)$$

Here, λ is a tuning parameter and we validate the value of λ by repeating the experiment multiple times with multiple λ values over different datasets and choose the one with better classification performance on the validation set for the respective datasets. A value of $\lambda = 0.01$ works best.

4 Datasets and Experimental Setup

In this section, we empirically evaluate the performance of the proposed approach for text classification and compare our results with the other methods.

4.1 General Setting

In all the experiments, we use a publicly-available deep learning library *Baseline* – a fast model development tool for NLP tasks (Pressel et al., 2018). For all the different datasets, we choose a commonly-used, high-performance model from (Kim, 2014) as a base model. To examine the robustness of the proposed approach, we intentionally flip the class labels with 0% to 70% label noise, in other words: $p \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$, and observe the effect of different types of label flipping, such as uniform (*Uni*) and random (*Rand*) label flipping, along with instance-dependent label noise. For all the experiments, we use early stopping based on validation set accuracy where the class labels in validation are also corrupted.

	Dataset	K	L	N	T	Type
Text Data	SST-2	2	19	76961	1821	Balanced
	TREC	6	10	5000	500	Not Balanced
	AG-News	4	38	110K	10K	Balanced
	DBpedia	14	29	504K	70K	Balanced

Table 1: Summary of text classification datasets; K: denotes the number of classes, L: represents the average length of sentence, N: denotes the number of training samples, T: represents the number of test samples, Type: describes whether the dataset is balanced.

We indicate the performance of a standard deep network *Without Noise model (WoNM)* on the noisy label dataset. We also plot the results for the stacked *Noise Model Without Regularization (NMWoRegu)* and stacked *Noise Model With Regularization (NMwRegu)*. Unless otherwise stated, in all the deep networks with the stacked noise model, we initialize the noise layer parameters as an identity matrix. We further analyze the effect of the noise layer initialization on the overall performance. We define *TDwRegu* as the stacked noise model with regularization, initialized with true injected noise distribution and *RandwRegu* as the stacked noise model with regularization, initialized randomly. We run all experiments five times and report the mean accuracy.

4.2 Datasets

Here, we describe all the text classification datasets used to evaluate the performance of the proposed approach. The base model architecture is the same for all datasets. For each set, we tune the number of filter windows and filter lengths using the development set. Along with the description, we also provide the hyper-parameters we selected for each. Table 1 summarizes the basic statistic of the datasets.

1. SST-2[1] (Socher et al., 2011): Stanford Sentiment Treebank dataset for predicting the sentiment of movie reviews. The classification task involves detecting positive or negative reviews. Using the base model with

---

[1] `http://nlp.stanford.edu/sentiment/`

SST-2		Batch Size	50							100							
		Label Flips	Random							Random							
	Noise%	Clean Labels	10	20	30	40	45	47	50	0	10	20	30	40	45	47	50
	WoNM	87.27%	83.29%	79.08%	73.42%	64.03%	58.1%	54.73%	49.7%	86.53%	81.44%	75.58%	71.88%	63.39%	57.12%	55.81%	52.32%
	TDwRegu01	86.88%	85.37%	84.92%	**83.29%**	**78.53%**	**74.01%**	51.95%	**49.5%**	**86.88%**	**84.88%**	**85.08%**	**82.41%**	**76.09%**	**70.10%**	**58.98%**	49.86%
	NMWoRegu	**87.28%**	**86.2%**	84.07%	81.29%	70.42%	62.27%	**55.76%**	48.42%	86.66%	84.72%	83.03%	78.2%	66.65%	61.32%	57.11%	**52.24%**
	NMwRegu001	86.08%	85.01%	83.82%	81.97%	73.18%	62.18%	55.63%	48.87%	**86.51%**	**85.26%**	84.37%	81.05%	69.54%	60.89%	56.8%	51.6%
	NMwRegu01	**87.78%**	**86.04%**	**85.04%**	**82.7%**	**77.43%**	**66.96%**	**61.5%**	**49.08%**	86.33%	85.17%	**85.10%**	**81.9%**	**76.2%**	**65.47%**	**58.92%**	**52.46%**
TREC		Batch Size	10														
		Label Flips	Uniform							Random							
	Noise%	Clean data	10	20	30	40	50	60	70	0	10	20	30	40	50	60	70
	WoNM	92.8%	87.6%	83.6%	75.87%	67.27%	57.4%	46.27%	42.8%	92.8%	85.93%	82.2%	74.0%	68.4%	53.53%	48.2%	31.47%
	TDwRegu01	50.87%	45.33%	45.4%	36.33%	25.87%	28.33%	16.87%	16.87%	50.87%	56.4%	36.8%	24.0%	25.47%	22.6%	18.8%	22.6%
	NMWoRegu	92.33%	88.07%	84.67%	76.4%	68.47%	58.4%	50.07%	41.33%	92.07%	85.87%	84.27%	72.47%	66.53%	50.13%	44.6%	33.0%
	NMwRegu001	92.47%	90.53%	88.07%	81.6%	73.47%	64.07%	55.87%	43.67%	92.4%	88.53%	86.4%	77.2%	67.67%	54.67%	47.93%	**34.87%**
	NMwRegu01	**92.73%**	**90.8%**	**89.53%**	**88.67%**	**84.93%**	**79.67%**	**69.67%**	**52.4%**	**92.7%**	**90.33%**	**90.6%**	**86.47%**	**83.07%**	**70.93%**	**65.2%**	33.4%
		Batch Size	50														
		Label Flips	Uniform							Random							
	Noise%	Clean Labels	10	20	30	40	50	60	70	0	10	20	30	40	50	60	70
	WoNM	92.8%	87.27%	83.07%	75.00%	69.13%	61.53%	50.13%	39.8%	92.8%	86.00%	81.2%	76.2%	64.07%	52.4%	47.4%	34.13%
	TDwRegu01	55.73%	50.4%	44.73%	39.6%	22.27%	25.67%	14.93%	21.00%	55.73%	45%	44.93%	27.73%	27.87%	22.6%	17.87%	22.6%
	NMWoRegu	92.6%	87.73%	83.33%	76.33%	70.67%	56.8%	48.2%	39.67%	92.60%	85.27%	83.00%	73.6%	65.8%	50.4%	45.93%	30.73%
	NMwRegu001	92.53%	90.73%	87.20%	82.53%	73.93%	65.07%	52.87%	44.60%	92.53%	88.%	87.2%	79.07%	71.2%	51.67%	49.00%	33.40%
	NMwRegu01	**92.53%**	**91.33%**	**90.27%**	**88.47%**	**83.87%**	**77.87%**	**68.73%**	**55.67%**	**92.53%**	**90.00%**	**90.2%**	**85.93%**	**82.6%**	**71.4%**	**67.33%**	**37.53%**
AG-News		Batch Size	100														
		Label Flips	Uniform							Random							
	Noise%	Clean Labels	10	20	30	40	50	60	70	0	10	20	30	40	50	60	70
	WoNM	92.31%	89.96%	87.42%	84.55%	79.96%	75.42%	68.78%	59.94%	92.31%	89.71%	86.11%	79.05%	76.04%	65.09%	45.79%	38.12%
	TDwRegu01	92.47%	92.25%	92.15%	92.04%	84.87%	77.56%	63.13%	47.83%	92.68%	92.09%	91.99%	61.81%	62.44%	70.26%	24.99%	38.12%
	NMWoRegu	91.94%	91.89%	91.21%	90.51%	89.29%	88.02%	86.25%	79.88%	91.97%	91.79%	91.00%	90.04%	88.82%	86.49%	**77.66%**	43.01%
	NMwRegu001	92.47%	92.21%	91.82%	91.21%	90.71%	89.61%	88.43%	85.32%	**92.62%**	92.14%	91.5%	91.07%	90.2%	88.68%	64.01%	55.11%
	NMwRegu01	**92.55%**	**92.23%**	**92.2%**	**91.98%**	**91.7%**	**91.23%**	**90.54%**	**89.78%**	92.57%	**92.23%**	**91.96%**	**91.69%**	**91.13%**	**90.77%**	76.64%	**62.04%**
		Batch Size	1024														
		Label Flips	Uniform							Random							
	Noise%	Clean Labels	10	20	30	40	50	60	70	0	10	20	30	40	50	60	70
	WoNM	92.42%	89.77%	87.04%	84.07%	79.77%	74.54%	67.59%	59.41%	92.29%	89.47%	85.78%	80.51%	75.99%	65.55%	45.50%	39.75%
	TDwRegu01	92.61%	92.37%	92.18%	92.07%	84.92%	62.74%	63.43%	47.59%	92.54%	92.34%	91.82%	53.81%	69.04%	48.88%	25.05%	46.9%
	NMWoRegu	92.16%	91.51%	90.80%	89.58%	85.58%	79.96%	70.79%	62.89%	92.22%	91.61%	90.33%	86.92%	82.61%	71.49%	48.96%	39.96%
	NMwRegu001	92.4%	92.13%	91.88%	91.46%	90.14%	89.07%	86.96%	80.94%	92.54%	91.87%	91.38%	90.42%	99.18%	86.78%	75.74%	50.11%
	NMwRegu01	**92.66%**	**92.2%**	**92.29%**	**92.09%**	**91.7%**	**91.24%**	**90.72%**	**89.88%**	**92.57%**	**92.11%**	**91.99%**	**91.57%**	**91.2%**	**90.5%**	**77.93%**	**61.12%**
DBpedia		Batch Size	512														
		Label Flips	Uniform							Random							
	Noise%	Clean Labels	30	50	70	75	80	85	90	0	30	50	70	75	80	85	90
	WoNM	99.01%	95.19%	89.59%	74.01%	67.73%	57.87%	47.48%	34.01%	99.01%	94.72%	86.08%	62.87%	53.13%	40.78%	26.6%	12.42%
	NMWoRegu	98.93%	95.07%	90.2%	78.32%	73.65%	66.24%	54.24%	40.9%	98.93%	93.55%	84.53%	25.96%	54.84%	42.96%	29.25%	12.97%
	NMwRegu001	**99.04%**	**98.94%**	**98.81%**	**98.61%**	**98.52%**	**98.33%**	**98.13%**	**97.53%**	**99.04%**	98.93%	98.82%	98.62%	**98.48%**	**98.33%**	**89.00%**	11.36%
	NMwRegu01	99.01%	98.89%	98.71%	98.45%	98.32%	98.10%	97.76%	97.15%	98.92%	**99.01%**	**98.88%**	**98.72%**	98.10%	97.67%	38.62%	**16.27%**
		Batch Size	1024														
		Label Flips	Uniform							Random							
	Noise%	Clean Labels	30	50	70	75	80	85	90	0	30	50	70	75	80	85	90
	WoNM	98.96%	97.93%	96.47%	90.49%	68.07%	59.78%	48.06%	55.29%	98.96%	94.75%	86.36%	63.75%	53.39%	40.87%	26.18%	11.9%
	NMWoRegu	98.87%	97.37%	95.71%	89.54%	72.79%	66.49%	55.27	60.7%	98.87%	93.96%	85.6%	44.85%	54.32%	42.21%	28.63%	12.51%
	NMwRegu001	**98.97%**	**98.9%**	**98.79%**	**98.53%**	**98.50%**	**98.32%**	**98.19%**	**97.27%**	**98.97%**	98.83%	98.51%	98.1%	**98.49%**	**98.32%**	**83.79%**	10.51%
	NMwRegu01	98.92%	98.79%	98.58%	98.26%	98.32%	98.09%	97.79%	96.54%	98.92%	**98.88%**	**98.72%**	**98.35%**	98.12%	97.72%	33.10%	**15.94%**

Table 2: Test performance for different text classification datasets

clean labels we obtain classification accuracy of 87.27%. For this dataset, the base model network architecture consists of an input and embedding layer + $[3, 4, 5]$ feature windows with 100 feature maps each and dropout rate 0.5 with batch size 50.

2. TREC[2] (Voorhees and Tice, 1999): A question classification dataset consisting of fact based questions divided into broad semantic categories. We use a six-class version of TREC dataset. For this dataset, the base model network architecture consists of an input and embedding layer + $[3]$ one feature windows with 100 feature maps and dropout rate 0.5 with batch size 10.

3. Ag-News[3] (Zhang et al., 2015): A large-scale, four-class topic classification dataset. It contains approx 110K training samples. For this dataset, the base model network architecture consists of Input layer + Embedding layer + $[3, 4, 5]$ feature windows with 200 feature maps and dropout rate 0.5 with batch size 100.

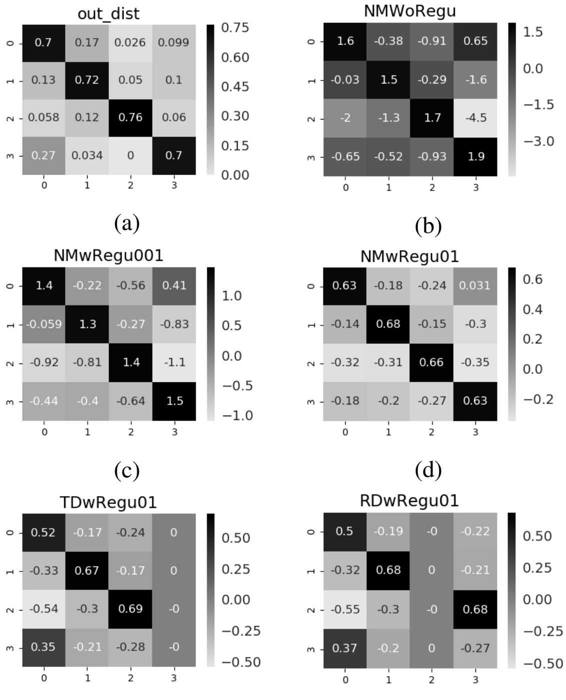
4. DBpedia[3] (Zhang et al., 2015): A large scale 14-class topic classification dataset containing $36K$ training samples per category. For this dataset, the base model network architecture consists of Input layer + Embedding layer + $[1, 2, 3, 4, 5, 7]$ feature windows with 400 feature maps each and dropout rate 0.5 with batch size 1024.

---

[2] `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

[3] `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

For all the datasets, we use Rectified Linear Units (ReLU) and fix the base model architecture. We use early stopping on dev. sets for all the datasets. We run all the experiments 5 times and report the average classification accuracy in Table 2. We train all the networks end-to-end via stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012) except for the DBpedia, where we use SGD. In order to improve base model performance, we initialize the word embedding layer with the publicly available *word2vec* word vectors (Mikolov et al., 2013) for all the datasets except for DBpedia, where we use *GloVe* embeddings (Pennington et al., 2014).



(a) (b) (c) (d) (e) (f)

Figure 1: AG-News Dataset: a) Input random label noise; (b-f) learned weight matrix learned by different noise models.

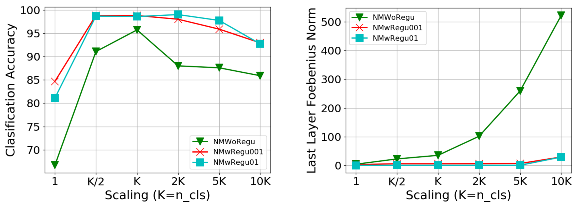
5 Results and Discussion

We evaluate the performance of our model in Table 2 for each datasets in the presence of uniform and random label noise and compare the performance with the base model (*WoNM*) as our baseline. For all datasets, the proposed approach is significantly better than the baseline for both random and uniform label noise. For all datasets, we observe a gain of approximately 30% w.r.t the baseline in the presence of extreme label noise. We do observe a drop in classification accuracy as we increase the percentage of label noise but even at the extreme label noise our method outperformed the baseline method. Interestingly, if we assume an oracle to determine prior knowledge of true noise distribution (*TDwRegu01*), it does not necessarily improve classification performance, especially for multi-class classification problems. For binary classification, using the SST-2 dataset, we did observe that the noise model initialized with the true noise distribution works better than all the other models. In addition to this, we also observe a slight performance gain for the proposed approach over the baseline with clean labels – perhaps due to label noise inherent in the datasets.

5.1 Effect of Different Regularizers

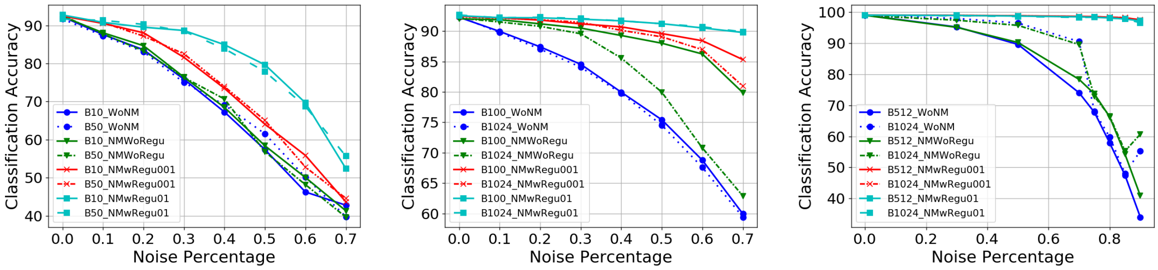
The *NMwRegu01* performs better in all cases for both types of label noise. We plot the weight matrix learned by all the noise models in all the noise regimes. For brevity, we only plot the weight matrix for AG-News datasets with 30% label noise in Fig.1. We find that l_2 regularization diffuses the diagonal weight elements and learns more smoothed off-diagonal elements which resemble the corresponding input label noise distribution in Fig. 1d. This also means that, without regularization, the noise model has less ability to diffuse the diagonal elements which leads to poor classification performance. Therefore, we use a regularizer (l_2) to diffuse the diagonal entries.

In some cases, especially for low label noise, we find the l_2 regularization with a small penalty works better than a large penalty since, for low label noise, learning a less diffuse noise is beneficial. The proposed approach scales to a large number of label categories, as evident from the experiments on DBpedia dataset in the last row of Table .2.

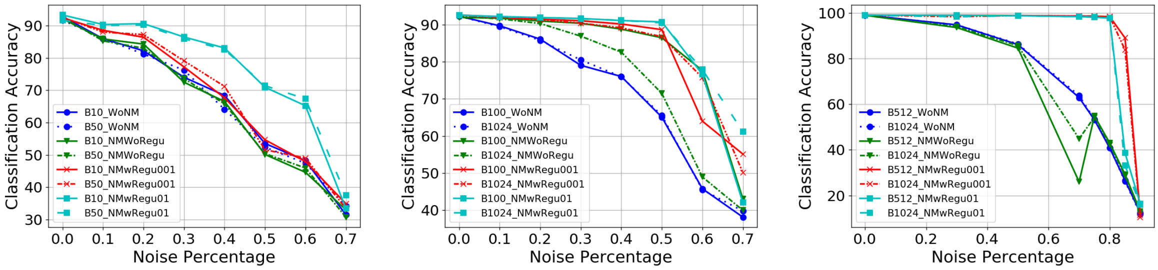


(a) Classification accuracy (b) Noise model norm

Figure 2: Effect of noise model initialization scaling on the classification performance



(a) TREC [Uniform] (b) AG-News [Uniform] (c) DBpedia [Uniform]



(d) TREC [Random] (e) AG-News [Random] (f) DBpedia [Random]

Figure 3: Effect of batch size on label noise classification for different datasets. [Best viewed in color]

5.2 Effect of Different Scaling Factors on Noise Layer Initialization

We initialize the noise model weights as identity matrices with gain equal to the number of classes (gain = K) for all experiments. We observe the effect of different gain values on the overall performance of the proposed network in Fig. 2 where on x-axis we vary the scaling factor – a function of number of classes in the dataset. We plot the classification performance for the DBpedia dataset with 50% random noise. For each noise model in Fig. 2a, we find that setting the gain to K works best and any other gain results in poor performance.

In Fig. 2b we plot the Frobenius norm of the learned noise model weights with respect to the different gain values. We find that, using the high gain initialization, the model learns a high noise model norm, resulting in poor classification performance. This finding provides support to the claim in (Liao et al., 2018) that “higher layer norm leads to highest test errors.”

5.3 Effect of Batch Size

We also observe the effect of different batch sizes on performance as described in (Rolnick et al., 2017). For all datasets, we do observe small performance gains for highly non-uniform noisy labels, for instance 70%, in Fig. 3 row 2. However, for uniform label flips, we do not observe performance gains with increasing batch size.

	TRB			TRPr		
Data(N%)	WoNM	Noisy	True	NMwRegu01	Noisy	True
SST2 (40%)	70.24	70.95	79.24	82.32	73.90	83.25
AG (70%)	59.70	52.44	79.18	90.33	86.27	89.4
AG (60%)	83.25	68.8	88.28	90.45	87.77	90.78
TREC (40%)	66.80	63.4	79.0	73.40	69.6	83.2
TREC (20%)	83.6	80.0	86.0	87.40	83.6	90.0

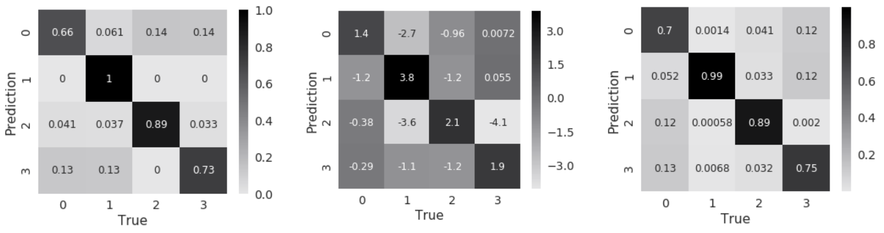
Table 3: SVM Classification

5.4 Instance Dependent Label Noise

We further investigate the performance of the proposed approach on instance-dependent label noise by flipping each class labels with different noise percentages as shown in Fig. 4a. For brevity, we present results on AG-News dataset in Fig. 4. On this type of label noise, the performance of the proposed approach is far better than the baseline with a performance improvement of $\sim 6\%$. The noise model learned by the proposed approach is shown in Fig. 4b and we show the normalized weight matrix in Fig. 4c. We observe that the learned noise model is able to capture the input label noise statistics and is highly correlated to the input noise distribution with Pearson Correlation Coefficient 0.988.

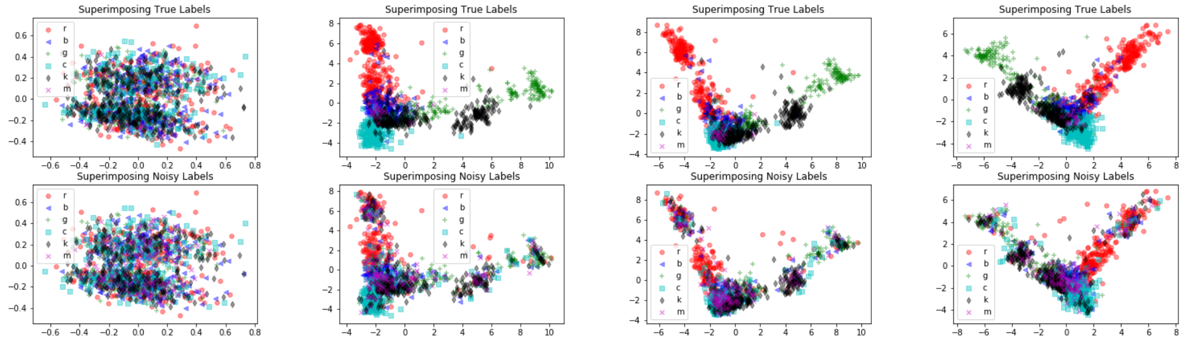
5.5 Understanding Noise Model

In order to further understand the noise model, we first train the base model and the proposed model on noisy labels. Afterward, we collect



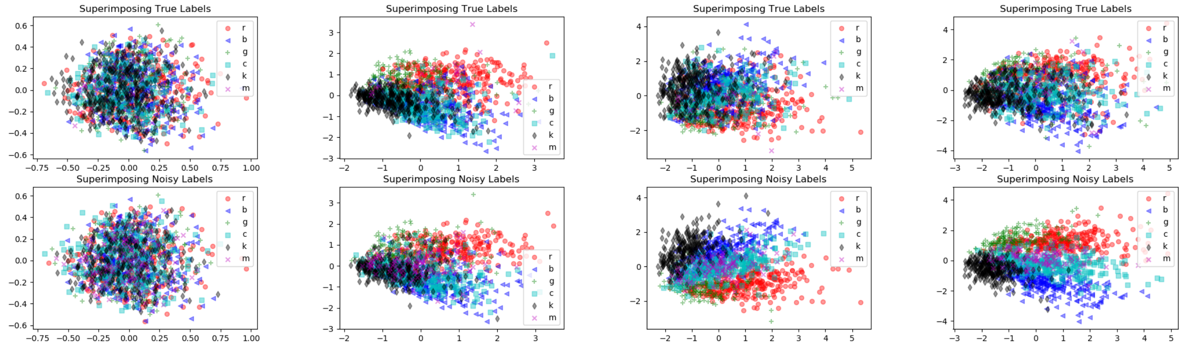
(a) (b) (c)

Figure 4: AG-News Dataset: a) input instance dependent label noise; b) learned weight matrix by proposed approach; c) column normalization of (b).



Iteration 0 Iteration 5 Iteration 10 Iteration 18

(a) Proposed model



Iteration 0 Iteration 5 Iteration 10 Iteration 18

(b) No noise model stacked

Figure 5: t-SNE visualization of the last layer activations of a base network before softmax for TREC Dataset with 50% corrupted labels; First row in (a) when the corresponding true labels are superimposed on the t-SNE data points; Second row in (a) when the noisy labels are superimposed onto the t-SNE data points. [Best viewed in color]

the last fully-connected layer's activations for all the training samples and treat them as the learned feature representation of the input sentence. We get two different sets of feature representations, one corresponding to the base model (*TRB*), and the other corresponding to the proposed model (*TRPr*). Given these learned feature representations – the artificially injected noisy labels and the true labels of the training data – we learn two different SVMs for each model, with and without noise. For the base model, for both SVMs, we use TRB representation as inputs and train the first SVM with the true labels as targets and the second SVM with the unreliable labels as targets. Similarly, we train two SVMs for the proposed model. After training, we evaluate the performance of all the learned SVMs on clean test data in Table 3, where the 1st column represents the corresponding model performance, "Noisy" and "True" column represents the SVM performance when trained on noisy and clean labels, respectively. We run these experiments for different datasets with different label noise.

The SVM, trained on TRB and noisy labels, is very close to the base model performance (3). This suggests that the base model is just fitting the noisy labels. On the other hand, when we train an SVM on the TRPr representations with true labels as targets, the SVM achieves the proposed model performance. This means that the proposed approach helps the base model to learn better feature representations even with the noisy targets, which suggest that this noise model is learning a label denoising operator.

We analyze the representation of training samples in feature domain by plotting the t-SNE embeddings (Van Der Maaten, 2014) of the TRB and TRPr. For brevity, we plot the t-SNE visualizations for TREC dataset with 50% label noise in Fig. 5 .

For each network, we show two different t-SNE plots. For example in Fig. 5a we plot two rows of t-SNE embeddings for the proposed model. In the first row of Fig. 5a, each training sample is represented by its corresponding true label, while in the second row (the noisy label plot) each training sample is represented by its corresponding noisy label. We observe that, as the learning process progresses, the noise model helps the base model to cluster the training samples in the feature domain. With each iteration, we can see the formation of clusters in Row 1. However, in Row 2, when the noisy labels are superimposed, the clusters are not well separated. This means that the noise model denoises the labels and presents the true labels to the base network to learn.

In Fig. 5b, we plot two rows of t-SNE embeddings of the TRB representations. It seems that the network directly learns the noisy labels. This provides further evidence to support (Zhang et al., 2016)'s finding that the deep network memorizes data without knowing of true labels. In Row 2 of Fig. 5b, we can observe that the network learns noisy features representations which can be well clustered according to given noisy labels.

6 Conclusion

In this work, we propose a framework to enable a DNN to learn better sentence representations in the presence of label noise for text classification tasks. To model the label noise, we append a nonlinear noise model on top of the standard DNN architecture. With proper initialization and regularization, the noise model is able to absorb most of the label noise and helps the base model to learn better sentence representations.

Acknowledgments

We thank the anonymous reviewers for their detailed and insightful comments. We would also like to thank Patrick Haffner, Sagnik Ray Choudhury, Yanjie Zhao and Amy Hemmeter for their valuable discussions with us during the course of this research.

References

Samaneh Azadi, Jiashi Feng, Stefanie Jegelka, and Trevor Darrell. 2015. Auxiliary image regularization for deep cnns with noisy labels. *arXiv preprint arXiv:1511.07069*.

Carla E Brodley and Mark A Friedl. 1999. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. 2018. Masking: A new perspective of noisy supervision. *arXiv preprint arXiv:1805.08193*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2017. Mentornet: Regularizing very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*.

Ishan Jindal, Matthew Nokleby, and Xuewen Chen. 2016. Learning deep networks from noisy labels with dropout regularization. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 967–972. IEEE.

Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. 2016. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, pages 67–84. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Neil D Lawrence. 2001. Estimating a kernel fisher discriminant in the presence of label noise. In *ICML*. Citeseer.

Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. 2017. Learning from noisy labels with distillation. In *ICCV*, pages 1928–1936.

Qianli Liao, Brando Miranda, Andrzej Banburski, Jack Hidary, and Tomaso Poggio. 2018. A surprising linear relationship predicts test performance in deep networks. *arXiv preprint arXiv:1807.09659*.

Naresh Manwani and PS Sastry. 2013. Noise tolerance under risk minimization. *IEEE transactions on cybernetics*, 43(3):1146–1151.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ishan Misra, C Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. 2016. Seeing through the human reporting bias: Visual classifiers from noisy humancentric labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2939.

Volodymyr Mnih and Geoffrey E Hinton. 2012. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Daniel Pressel, Sagnik Ray Choudhury, Brian Lester, Yanjie Zhao, and Matt Barta. 2018. Baseline: A library for rapid modeling, experimentation and development of deep learning algorithms targeting nlp. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 34–40. Association for Computational Linguistics.

Umaa Rebbapragada and Carla E Brodley. 2007. Class noise mitigation through instance weighting. In *European Conference on Machine Learning*, pages 708–715. Springer.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2014. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Arash Vahdat. 2017. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 6575–6583.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699.

Jiangchao Yao, Jiajie Wang, Ivor W Tsang, Ya Zhang, Jun Sun, Chengqi Zhang, and Rui Zhang. 2018. Deep learning from noisy image labels with quality embedding. *IEEE Transactions on Image Processing*.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Xingquan Zhu and Xindong Wu. 2004. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210.

Understanding Learning Dynamics Of Language Models with SVCCA

Naomi Saphra and **Adam Lopez**
Institute for Language, Cognition, and Computation
University of Edinburgh
`n.saphra@ed.ac.uk` `alopez@ed.ac.uk`

Abstract

Research has shown that neural models implicitly encode linguistic features, but there has been no research showing *how* these encodings arise as the models are trained. We present the first study on the learning dynamics of neural language models, using a simple and flexible analysis method called Singular Vector Canonical Correlation Analysis (SVCCA), which enables us to compare learned representations across time and across models, without the need to evaluate directly on annotated data. We probe the evolution of syntactic, semantic, and topic representations and find that part-of-speech is learned earlier than topic; that recurrent layers become more similar to those of a tagger during training; and embedding layers less similar. Our results and methods could inform better learning algorithms for NLP models, possibly to incorporate linguistic information more effectively.

1 Introduction

Large neural networks have a notorious capacity to memorize training data (Zhang et al., 2016), but their high accuracy on many NLP tasks shows that they nonetheless generalize. One apparent explanation for their performance is that they learn linguistic generalizations even without explicit supervision for those generalizations—for example, that subject and verb number agree in English (Linzen et al., 2016); that derivational suffixes attach to only specific parts of speech (Kementchedjhieva and Lopez, 2018); and that short segments of speech form natural clusters corresponding to phonemes (Alishahi et al., 2017). These studies tell us that neural models learn to implicitly represent linguistic categories and their interactions. But *how* do they learn these representations?

One clue comes from the inspection of multilayer models, which seem to encode lexical categories in lower layers, and more contextual categories in higher layers. For example, Blevins et al. (2018) found that a word's part of speech (POS) is encoded by lower layers, and the POS of its syntactic parent is encoded by higher layers; while Belinkov et al. (2018) found that POS is encoded by lower layers and semantic category is encoded by higher layers. More generally, the most useful layer for an arbitrary NLP task seems to depend on how "high-level" the task is (Peters et al., 2018). Since we know that lower layers in a multi-layer model converge to their final representations more quickly than higher layers (Raghu et al., 2017), it is likely that models learn local lexical categories like POS earlier than they learn higher-level linguistic categories like semantic class.

How and when do neural representations come to encode specific linguistic categories? Answers could explain why neural models work and help us improve learning algorithms. We investigate how representations of linguistic structure are learned over time in neural language models (LMs), which are central to NLP: on their own, they are used to produce contextual representations of words for many tasks (e.g. Peters et al., 2018); while *conditional* LMs power machine translation, speech recognition, and dialogue systems. We use a simple and flexible method, Singular Vector Canonical Correlation Analysis (SVCCA; Raghu et al., 2017), which allows us to compare representations from our LM at each epoch of training with representations of other models trained to predict specific linguistic categories. We discover that lower layers initially discover features shared by all predictive models, but lose these features as the LM explores more specific clusters. We demonstrate that different aspects of linguistic structure are learned at different rates within a single recurrent layer, acquiring POS tags early but continuing to learn global topic information later in training.

Proceedings of NAACL-HLT 2019, pages 3257–3267
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

2 Methods

Our experiments require a LM, tagging models, and a method to inspect the models: SVCCA.

2.1 Language model

We model the probability distribution over a sequence of tokens $x_1 \dots x_{|x|}$ with a conventional two-layer LSTM LM. The pipeline from input x_t at time step t to a distribution over x_{t+1} is described in Formulae (1)–(4). At time step t, input word x_t is embedded as (1) h_t^e, which is input to a two-layer LSTM, producing outputs (2) h_t^1 and (3) h_t^2 at these layers, along with cell states c_t^1 and c_t^2. A softmax layer converts h_t^2 to a distribution from which (4) $\hat{x}_{t+1}$ is sampled.

$$\begin{aligned} h_t^e &= \text{embedding}(x_t) && (1) \\ h_t^1, c_t^1 &= \text{LSTM}_1(h_t^e, h_{t-1}^1, c_{t-1}^1) && (2) \\ h_t^2, c_t^2 &= \text{LSTM}_2(h_t^1, h_{t-1}^2, c_{t-1}^2) && (3) \\ \hat{x}_{t+1} &\sim \text{softmax}(h_t^2) && (4) \end{aligned}$$

Each function can be thought of as a *representation* or *embedding* of its discrete input; hence h_t^e is a representation of x_t, and—due to the recursion in (2)—h_t^1 is a representation of $x_1 \dots x_t$.

2.2 Tagging models

To inspect our language model for learned linguistic categories, we will use a collection of tagging models, designed to mimic the behavior of our language model but predicting the next *tag* rather than the next word. Given $x_1 \dots x_{|x|}$, we model a corresponding sequence of tags $y_1 \dots y_{|x|}$ using a one-layer LSTM. (Our limited labeled data made this more accurate on topic tagging than another two-layer LSTM, so this architecture does not directly parallel the LM.)

$$\begin{aligned} h_t^{e\prime} &= \text{embedding}'(x_t) && (5) \\ h_t^{1\prime}, c_t^{1\prime} &= \text{LSTM}'(h_t^{e\prime}, h_{t-1}^{1}{}', c_{t-1}^{1}{}') && (6) \\ \hat{y}_{t+1} &\sim \text{softmax}'(h_t^{1\prime}) && (7) \end{aligned}$$

We will also discuss *input taggers*, which share this architecture but instead sample y_t, the tag of the most recently observed word.

2.3 SVCCA

SVCCA is a general method to compare the correlation of two vector representations. Let d_A and d_B be their dimensions. For N data points we have two distinct views, given by matrices $A \in \mathcal{R}^{N \times d_A}$ and $B \in \mathcal{R}^{N \times d_B}$. We project these views onto a shared subspace in two steps:

1. Use Singular Value Decomposition (SVD) to reduce matrices A and B to lower dimensional matrices A' and B', respectively. This is necessary because many dimensions in the representations are noisy, and in fact cancel each other out (Frankle and Carbin, 2018). SVD removes dimensions that were likely to be less important in the original representations from A and B, and in keeping with Raghu et al. (2017), we retain enough dimensions to keep 99% of the variance in the data.

2. Use Canonical Correlation Analysis (CCA) to project A' and B' onto a shared subspace, maximizing the correlation of the projections. Formally, CCA identifies vectors w, v to maximize $\rho = \frac{<w^\top A', v^\top B'>}{\|w^\top A'\| \|v^\top B'\|}$. We treat these w, v as new basis vectors, computing the top d_C (a hyperparameter) such basis vectors to form projection matrices $W \in \mathcal{R}^{d_C \times d_{A'}}, V \in \mathcal{R}^{d_C \times d_{A'}}$. The resulting projections WA' and VB' map onto a shared subspace where the representations of each datapoint from A' and B' are maximally correlated.

Intuitively, the correlation ρ will be high if both representations encode the same information, and low if they encode unrelated information. Figure 1 illustrates how we use SVCCA to compare representation h_t^2 of our language model with the recurrent representation of a tagger, $h_t^{1\prime}$. In practice, we run over all time steps in a test corpus, rather than a single time step as illustrated.

3 Experimental Setup

We trained our LM on a corpus of tokenized, lowercased English Wikipedia (70/10/20 train/dev/test split). To reduce the number of unique words in the corpus, we excluded any sentence with a word type appearing fewer than 100 times. Words appearing fewer than 100 times in the resulting training set are replaced with an unknown token. The resulting training set has over 227 million tokens of 20K types.

We train for 50 epochs to maximize cross-entropy, using a batch size of 40, dropout ratio of 0.2, and sequence length of 35. The optimizer is standard SGD with clipped gradients at 0.25,

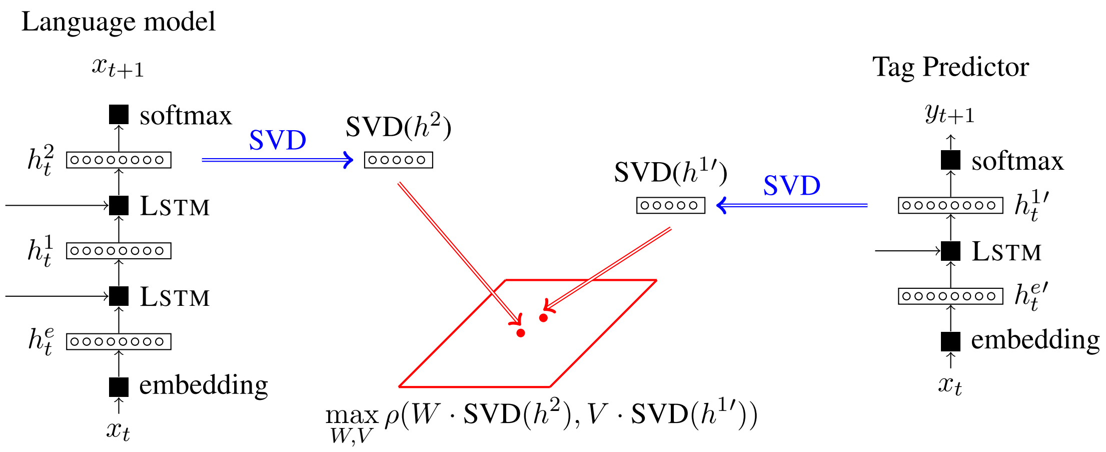


Figure 1: SVCCA used to compare the layer h^2 of a language model and layer $h^{1\prime}$ of a tagger.

Tag	These	cats	live	in	that	house	.
UDP POS	DET	NOUN	VERB	ADP	DET	NOUN	SYM
PTB POS	DT	NNS	VBP	IN	DT	NN	.
SEM (coarse)	DEM	ENT	EVE	ATT	DEM	ENT	LOG
SEM (fine)	PRX	CON	ENS	REL	DST	CON	NIL
topic	pets	pets	pets	pets	pets	pets	pets

Table 1: An annotated example sentence from the article *pets*, based on an example from Bjerva et al. (2016).

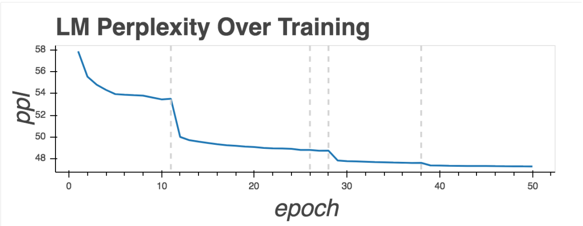


Figure 2: Test performance of the LM. Vertical dotted lines indicate when the optimizer rescales the step size.

with the learning rate quartered when validation loss increases. The result of training is shown in Figure 2, which illustrates the dips in loss when learning rate changes. All experiments on the LM throughout training are conducted by running the model at the end of each epoch in inference mode over the test corpus.

3.1 Taggers

To understand the representations learned by our LM, we compare them with the internal representations of tagging models, using SVCCA. Where possible, we use coarse-grained and fine-grained tagsets to account for effects from the size of the tagset. Table 1 illustrates our tagsets.

POS tagging For syntactic categories, we use POS tags, as in Belinkov et al. (2017). As a coarse-grained tagset, we use silver Universal Dependency Parse (UDP) POS tags automatically added to our Wikipedia corpus with spacy.[1] We also use a corpus of fine-grained human annotated Penn Treebank POS tags from the Groningen Meaning Bank (GMB; Bos et al., 2017).

Semantic tagging We follow Belinkov et al. (2018) in representing word-level semantic information with silver SEM tags (Bjerva et al., 2016). SEM tags disambiguate POS tags in ways that are relevant to multilingual settings. For example, the comma is not assigned a single tag as punctuation, but has distinct tags according to its function: conjunction, disjunction, or apposition. The 66 fine-grained SEM tag classes fall under 13 coarse-grained tags, and an ‘unknown’ tag.

Global topic For topic, we classify each word of the sequence by its source Wikipedia article; for example, every word in the wikipedia article on Trains is labeled “Trains”. This task assesses whether the network encodes the global topic of the sentence.

---

[1] https://spacy.io/

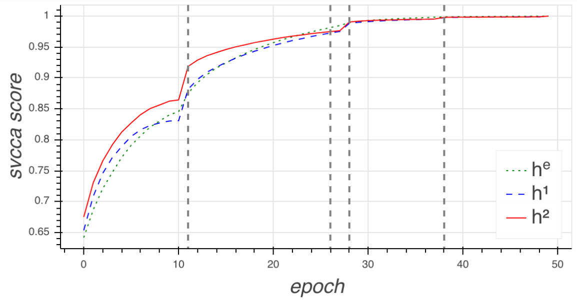


Figure 3: SVCCA score between representations at each epoch and from the final trained LM.

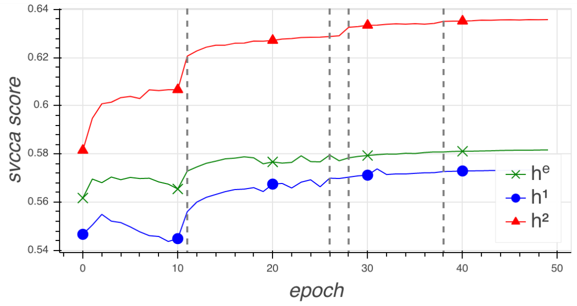


Figure 4: SVCCA score between the LM at each epoch and a LM with different initialization.

UDP silver POS and topic information use the same corpus, taken from the 100 longest articles in Wikipedia randomly partitioned in a 70/10/20 train/dev/test split. Each token is tagged with POS and with the ID of the source article. The corpus is taken from the LM training data, which may increase the similarity between the tag model and LM. Because both tag predictors are trained and tested on the same domain as the LM, they can be easily compared in terms of their similarity to the LM representation. Though the SEM corpus and the PTB corpus are different domains from the Wikipedia training data, we compare their activations on the same 191K-token 100-article test corpus.

Table 2 describes the training and validation corpus statistics for each tagging task. Note that topic and UDP POS both apply to the same en-wikipedia corpus, but PTB POS and SEM use two different unaligned sets from the GMB corpus.

4 Experiments, Results, and Analysis

A benefit of SVCCA is its flexibility: it can compute the correlation of a hidden representation to any other vector. Raghu et al. (2017) used it to understand learning dynamics by comparing a learned representation to snapshots of the same representation at different epochs during training. We use a similar experiment to establish the basic learning dynamics of our model. In our shallow 2-level model, activations at h^1 converge slightly after h^2 (Figure 3). This differs from the results of Raghu et al. (2017), who found that a 5-layer stacked LSTM LM exhibits faster convergence at lower layers, but this difference may be attributed to our much larger training data, which gives our model sufficient training data at early epochs.

Empirical upper bounds. Our main experiments will test the rate at which different linguistic categories are learned by different layers, but to interpret the results, we need to understand the behaviour of SVCCA for these models. In theory, SVCCA scores can vary from 0 for no correlation to 1 for perfect correlation. But in practice, these extreme cases will not occur. To establish an *empirical* upper bound on correlation, we compared the similarity at each epoch of training to the frozen final state of a LM with identical architecture but different initialization, trained on the same data (Figure 4).[2] The correlations increase over time as expected, but to a maximum near 0.64; we don't expect correlations between our LM and other models to exceed this value. We explore corresponding lower bounds in our main experiments below.

Correlations between different layers. Next we examine the correlation between different layers of the same model over time (Figure 5). We observe that, while over time correlation increases, in general closer layers are more similar, and they are less correlated than they are with the same layer of a differently initialized model. This supports the idea that we should compare recurrent layers with recurrent layers because their representations play similar roles within their respective architectures.

SVCCA vs. Diagnostic classifiers A popular method to analyze learned representations is to use a *diagnostic classifier* (Belinkov et al., 2017; Hupkes et al., 2018) or probe (Conneau et al., 2018), a separate model that is trained to predict a linguistic category of interest, y_t, from an arbitrary hidden layer h_t. Diagnostic classifiers are widely used (Belinkov et al., 2018; Giulianelli et al., 2018).

---

[2] This experiment is similar to the comparisons of randomly initialized models by Morcos et al. (2018).

tag	corpus	number of classes	token count train	token count dev	label $t+1$ acc	label $t+1$ ppl	label t acc	label t ppl	randomized acc	randomized ppl
UDP POS	wiki	17	665K	97K	50	4.3	93	1.2	21	8.9
PTB POS	GMB	36	943K	136K	51	4.7	95	1.18	14	18.0
SEM (coarse)	GMB	14	937K	132K	55	3.5	91	1.3	22	9.0
SEM (fine)	GMB	67	937K	132K	50	5.6	88	1.45	17	21.5
topic	wiki	100	665K	97K	36	19.1	37	16.3	5	81.6

Table 2: Tag predictor and tagger statistics. Accuracy and perplexity on $t+1$ are from the target tag predictor, on t are from the input tagger. Metrics obtained when training on randomly shuffled labels are provided as a low baseline. Accuracy is on the test set from the training domain (GMB or Wikipedia).

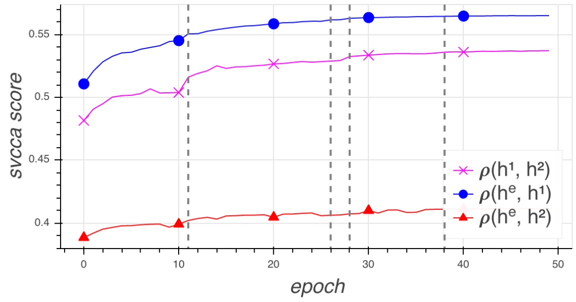


Figure 5: SVCCA score between different layers of the LM at each epoch. For example, $\rho(h^2, h^1)$ compares the activations h^2 with the activations h^1.

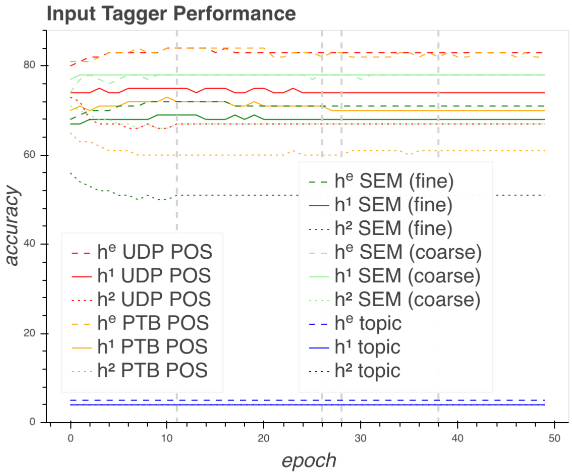


Figure 6: Learning dynamics interpreted with diagnostic classifiers labeling input word tag y_t.

But if a diagnostic classifier is trained on enough examples, then random embeddings as input representations often outperform any pretrained intermediate representation (Wieting and Kiela, 2019; Zhang and Bowman, 2018). This suggests that diagnostic classifiers may work simply by memorizing the association between an embedding and the most frequent output category associated with that embedding; since for many words their category is (empirically) unambiguous, this may give an inflated view of just how much a model “understands” about that category.

Our use of SVCCA below will differ from the use of diagnostic classifiers in an important way. Diagnostic classifiers use the intermediate representations of the LM as inputs to a tagger. A representation is claimed to encode, for example, POS if the classifier accurately predicts it—in other words, whether it can *decode* it from the representation. We will instead evaluate the *similarity* between the representations in an LM and in an independently-trained tagger. The intuition behind this is that, if the representation of our LM encodes a particular category, then it must be similar to the representation of model that is specifically trained to predict that category. A benefit of the approach is that similarity can be evaluated on *any* dataset, not only one that has been labeled with the linguistic categories of interest.

Another distinction from the typical use of diagnostic classifiers is that probes are usually used to decode tag information about the context or most recent *input* from the hidden state at the current step. Because the hidden representation at time t is meant to encode predictive information about the target word at time $t+1$, we treat it as encoding a prediction about the tag of the *target* word.

To understand the empirical strengths and weaknesses of these approaches, we compare the use of SVCCA and diagnostic classifiers in understanding learning dynamics. In other words, we ask: is our first conceptual shift (to SVCCA) necessary? To test this, we use the same model as Belinkov et al. (2017), which classifies an arbitrary representation using a ReLU followed by a softmax layer. To be consistent with Belinkov et al. (2017), we use y_t as their target label. We repeat their method in this manner (Figure 6) as well as applying our second modification, in which we instead target the label y_{t+1} (Figure 7).

We found the correlations to be relatively stable over the course of training. This is at odds with

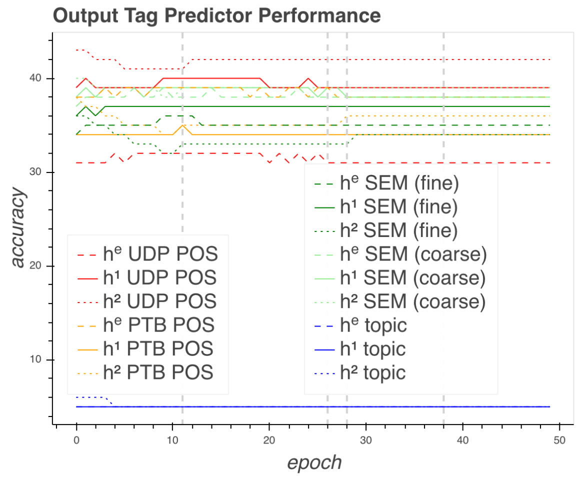


Figure 7: Learning dynamics interpreted with diagnostic classifiers labeling target word tag y_{t+1}.

the results in Figures 2 and 3, which suggest that representations change substantially during training in ways that materially affect the accuracy of the LM. This suggests that diagnostic classifiers are not illustrating improvements in word representations throughout training, and we conclude that they are ineffective for understanding learning dynamics. Our remaining experiments use only SVCCA.

4.1 SVCCA on Output Tag Prediction

We applied SVCCA to each layer of our LM with the corresponding layer of each tag predictor in order to find the correlation between the LM representation and the tag model representation at each level (Figure 8). To establish empirical lower bounds on correlation, we also trained our taggers on the same data with randomly shuffled labels, as in Zhang et al. (2016). These latter experiments, denoted by the dotted lines of Figure 8, show how much of the similarity between models is caused by their ability to memorize arbitrary associations. Note that the resulting scores are nonzero, likely because the linguistic structure of the input shapes representations even when the output is random, due to the memorization phase of training (Shwartz-Ziv and Tishby, 2017).

The strongest similarity at recurrent layers belongs to the most local property, the UDP POS tag. Both coarse- and fine-grained semantic tags, which rely on longer range dependencies, fall below UDP POS consistently. Topic, which is global to an entire document, is the least captured and the slowest to stabilize. Indeed, correlation with true

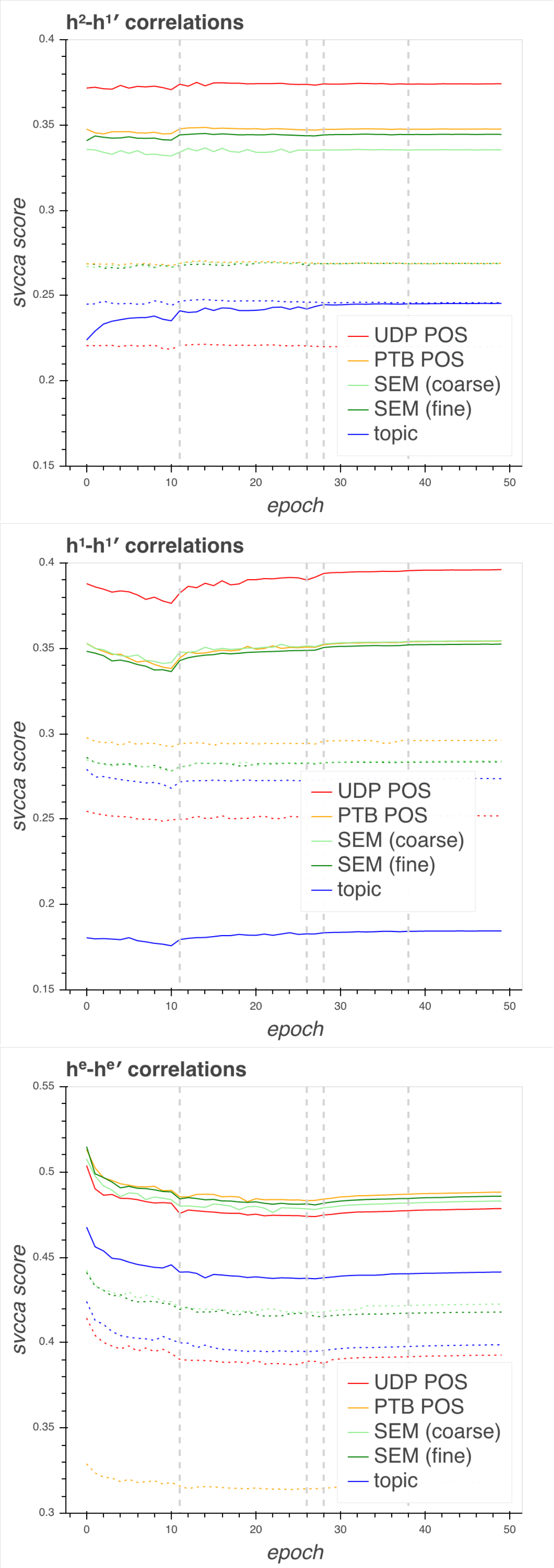


Figure 8: SVCCA correlation scores between the LM predicting x_{t+1} and the tag model predicting y_{t+1}. At the end of each epoch, we compare the current LM with the final tag model. Dotted lines use shuffled tags. Gray vertical lines mark when the step size is rescaled.

topic falls consistently *below* the score for a model trained on randomized topic tags, implying that early in training the model has removed the context necessary to identify topic (below even the inadequate contextual information memorized by a model with random labels), which depends on the general vocabulary in a sentence rather than a local sequence. Over time correlation rises, possibly because the model permits more long-distance context to be encoded. Khandelwal et al. (2018) found that LSTMs remember content words like nouns for more time steps than they remember function words like prepositions and articles. We hypothesize that the LM's slower stabilization on topic is related to this phenomenon, since it must depend on content words, and its ability to remember them increases throughout training.

The encoder layer exhibits very different patterns. Because the representation produced by the encoder layer is local to the word, the nuances that determine how a word is tagged in context cannot be learned. The encoder layers are all highly similar to each other, which suggests that the unigram representations produced by the encoder are less dependent on the particular end task of the neural network. Similarity between the encoders declines over time as they become more specialized towards the language modeling task. This decline points to some simple patterns which are learned for all language tasks, but which are gradually replaced by representations more useful for language modeling. This process may even be considered a naturally occurring analog to the common practice of initializing the encoder layer as word embeddings pretrained an unrelated task such as skipgram or CBOW (Mikolov et al., 2013). It seems that the 'easy' word properties, which immediately improve performance, are similar regardless of the particular language task.

At h^1, the correlation shows a clear initial decline in similarity for all tasks. This seems to point to an initial representation that relies on simple shared properties, which in the first stage of training is gradually dissolved before the layer begins to converge on a structure shared with each tag predictor. It may also be linked to the information bottleneck learning phases explored by Shwartz-Ziv and Tishby (2017). They suggest that neural networks learn by first maximizing the mutual information between the input and internal representation, then minimizing the mutual information between the internal representation and output. The network thus initially learns to effectively represent the input, then compresses this representation, keeping only the elements relevant to the output.[3] If the LM begins by maximizing mutual information with input, because the input is identical for the LM and tag models it may lead to these similar initial representations, followed by a decline in similarity as the compression narrows to properties specific to each task.

4.2 SVCCA on Input Tagging

Our second conceptual shift is to focus on output tag prediction—asking what a representation encodes about the *next* output word, rather than what it has encoded about words it has already observed in the input. What effect does this have? Since we already studied output tags in the previous set of experiments, here we consider input tags, in the style of most diagnostic classifier analysis (Figure 9). The learning dynamics are similar to those for tag prediction, but the UDP POS tagger decreases dramatically in all correlations while the GMB-trained taggers[4] often increase slightly. While the shapes of the lines are similar, UDP POS no longer consistently dominates the other tasks in recurrent layer correlation. Instead, we find the more granular PTB POS tags lead to the most similar representations.

5 Discussion and Conclusions

We find clear patterns in the encoding of linguistic structure with SVCCA, in contrast to the weaker results from a less responsive diagnostic classifier. Because SVCCA proves so much more sensitive than the diagnostic classifiers currently in use, we believe that future work on measuring the encoding of linguistic structure should use the similarity of individual modules from independently trained tag predictors rather than the performance of tag predictors trained on a particular representation.

This system should also be of interest because it is efficient. To train a diagnostic classifier, we must run a forward pass of the LM for each forward pass of the auxiliary model, while SVCCA only requires the LM to run on the test set. A fur-

---

[3] This memorization–compression learning pattern parallels the memorization–generalization of the first half of the U-shaped curve exhibited by human children learning irregular word forms. Kirov and Cotterell (2018) observe similar patterns when artificially modeling inflection.

[4] PTB POS, SEM (fine), and SEM (coarse)

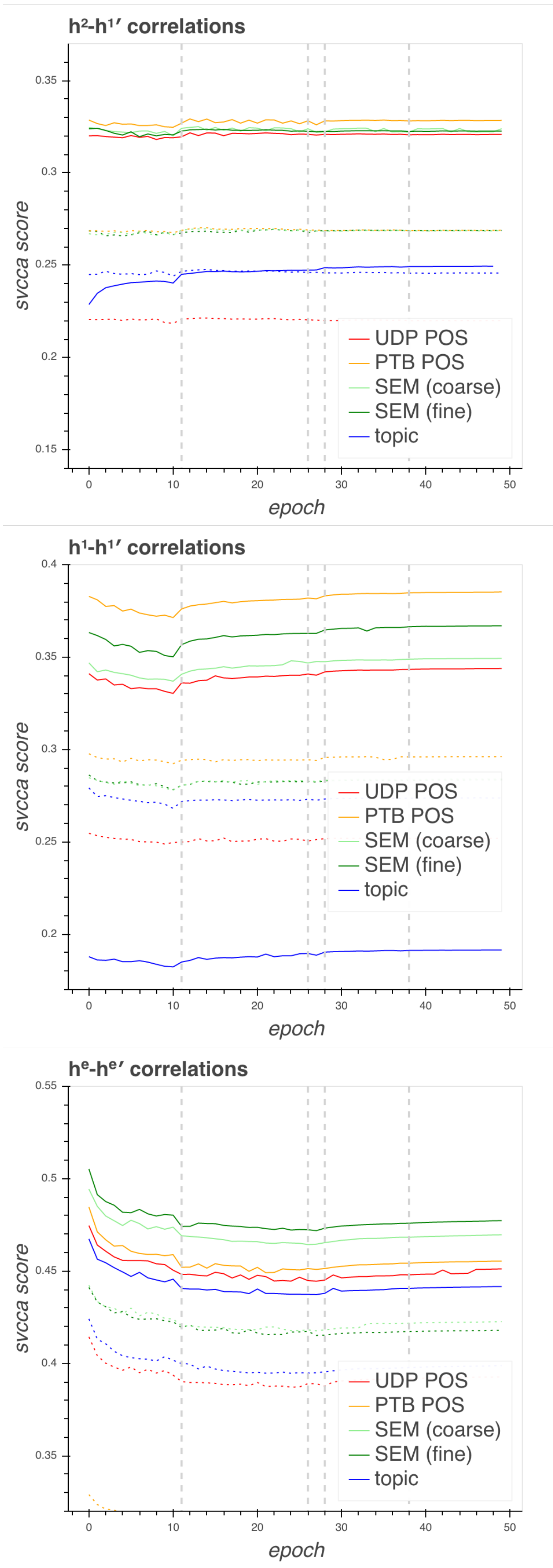


Figure 9: SVCCA correlation scores between LM activations when predicting x_{t+1} and tagger activations when labeling y_t. Dotted lines use shuffled tags. Gray vertical lines mark when the step size is rescaled.

ther efficiency gain is particular to studying learning dynamics: we train only one tagger and compare it to different versions of the LM over training, but for standard probing, we must train a new version of each layer's tagger at each epoch. Our SVCCA experiments in Figure 8 ran in hours, while the diagnostic classifier experiments in Figure 7 ran for days.

Our method holds another, more subtle advantage. Our analysis provides an alternative view of what it means for a model to *encode* some linguistic trait. The literature on analyzing neural networks includes a broad spectrum of interpretations about what it means to encode a property. At one end of the spectrum lies the purely informational view (e.g., mutual information; Noshad and Hero III, 2018). Mutual information is a very flexible view, but it requires us to compute theoretical information content, which in practice can only be estimated. Furthermore, information can be represented without being used, as shown by Vanmassenhove et al. (2017), who found that NMT systems often predicted tense according to a diagnostic classifier but did not produce the correct tense as output. The other end of the spectrum is focused on the structure of the representation space (e.g., the features and the property in question are linearly similar; Alishahi et al., 2017). Analyzing structural similarity should remedy the shortcomings of the informational view, but most intermediate representations are not targeted to extract the property in question through a linear transformation, and failing to be interpretable through such simple extraction should not be equated to a failure to encode that property.

Most of the literature on analyzing representations, by probing with a more complex architecture, seeks the flexibility of mutual information with the concreteness and tractability of the structural view – but instead obscures the strict information view without offering interpretable information about the structure, because the architecture of a diagnostic classifier affects its performance. It should not be surprising that representational quality as measured by such systems is a poor indicator of translation quality (Cífka and Bojar, 2018). SVCCA, in contrast, is a structural view that does not directly compare an activation that targets word prediction with a particular tag, but instead compares that activation with one targeting the prediction of the tag.

Let us consider a specific common probing method. What do we learn about the LM when a feedforward network cannot extract tag information directly from the embedding layer, but can from a recurrent layer? It may be tempting to conclude that tag information relies heavily on context, but consider some alternative explanations. If the embedding encodes the tag to be interpreted by a recurrent layer, a feedforward network may not be capable of representing the function to extract that tag because it does not have access to a context vector for aiding interpretation of the hidden layer. Perhaps its activation functions cover a different range of outputs. By directly comparing LSTM layers to LSTM layers and embedding layers to embedding layers, we respect the shape of their outputs and the role of each module within the network in our analysis.

The results of our analysis imply that early in training, representing part of speech is the natural way to get initial high performance. However, as training progresses, it increasingly benefits the model to represent categories with longer-range dependencies, such as topic.

6 Future Work

One direction for future work is exploring how generalization interacts with the correlations between LMs and tag predictors. It may be that a faithful encoding of a property like POS tag indicates that the LM is relying more on linguistic structure than on memorizing specific phrases, and therefore is associated with a more general model.

If these measurements of structure encoding are associated with more general models, we might introduce regularizers or other modifications that explicitly encourage correlation with a tagging task.

Combes et al. (2018) identified the phenomenon of *gradient starvation*, meaning that while frequent and unambiguous features are learned quickly in training, they slow down the learning of rarer features. For example, artificially brightening images according to their class leads to a delay in learning to represent the more complex natural class features. Although it is tempting to claim that semantic structure is learned using syntactic structure as natural scaffolding, it is possible that the simple predictive power of POS is acting as an attractor and starving semantic features that are rarer and more ambiguous. A possible direction for future work would be to explore which of these explanations is true, possibly by decorrelating particular aspects of linguistic structure from language modeling representations.

The techniques in this paper could be applied to better understand the high performance of a system like ELMo (Peters et al., 2018). Different layers in such a system are useful for different tasks, and this effect could be understood in terms of the gradual divergence between the layers and their respective convergence to representations geared toward a single task.

Acknowledgements

We thank Denis Emelin, Sameer Bansal, Toms Bergmanis, Maria Corkery, Sharon Goldwater, Sorcha Gilroy, Aibek Makazhanov, Yevgen Matusevych, Kate McCurdy, Janie Sinclair, Ida Szubert, Nikolay Bogoychev, Clara Vania, and the anonymous reviewers for helpful discussion and comments on drafts. We thank Matthew Summers for assistance with visualisations.

References

Afra Alishahi, Marie Barking, and Grzegorz Chrupała. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 368–378.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2018. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. *CoRR*, abs/1801.07772.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. In *COLING*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. *arXiv:1805.04218 [cs]*. ArXiv: 1805.04218.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In *Handbook of linguistic annotation*, pages 463–496. Springer.

Ondřej Cífka and Ondřej Bojar. 2018. Are bleu and meaning representation in opposition? *arXiv preprint arXiv:1805.06536*.

Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. 2018. On the Learning Dynamics of Deep Neural Networks. *arXiv:1809.06848 [cs, stat]*. ArXiv: 1809.06848.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv:1805.01070 [cs]*. ArXiv: 1805.01070.

Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Training Pruned Neural Networks. *arXiv:1803.03635 [cs]*. ArXiv: 1803.03635.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information. *arXiv:1808.08079 [cs]*. ArXiv: 1808.08079.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Yova Kementchedjhieva and Adam Lopez. 2018. *Indicatements* that character language models learn English morpho-syntactic units and regularities. In *Proc. of Workshop on Analyzing and interpreting neural networks for NLP*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. *arXiv:1805.04623 [cs]*. ArXiv: 1805.04623.

Christo Kirov and Ryan Cotterell. 2018. Recurrent neural networks in linguistic theory: Revisiting pinker and prince (1988) and the past tense debate. *arXiv preprint arXiv:1807.04783*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *arXiv:1611.01368 [cs]*. ArXiv: 1611.01368.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *arXiv:1806.05759 [cs, stat]*. ArXiv: 1806.05759.

Morteza Noshad and Alfred O. Hero III. 2018. Scalable Mutual Information Estimation using Dependence Graphs. *arXiv preprint arXiv:1801.09125*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. *arXiv:1706.05806 [cs, stat]*. ArXiv: 1706.05806.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*. ArXiv: 1703.00810.

Eva Vanmassenhove, Jinhua Du, and Andy Way. 2017. Investigating 'aspect'in nmt and smt: Translating the english simple past and present perfect. *Computational Linguistics in the Netherlands Journal*, 7:109–128.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. *arXiv preprint arXiv:1901.10444*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language Modeling Teaches You More Syntax than Translation Does: Lessons Learned Through Auxiliary Task Analysis. *arXiv:1809.10040 [cs]*. ArXiv: 1809.10040.

A Performance Out Of Domain

Because SEM tags and PTB POS tags were both trained on the GMB corpus, we present the SVCCA similarities on an in-domain GMB test corpus as well as the Wikipedia test corpus used elsewhere in the paper. The results are in Figures 10-11. In general correlations are higher using the original tagging domain, but not enough to contradict our earlier analysis. The shapes of the curves remain similar.

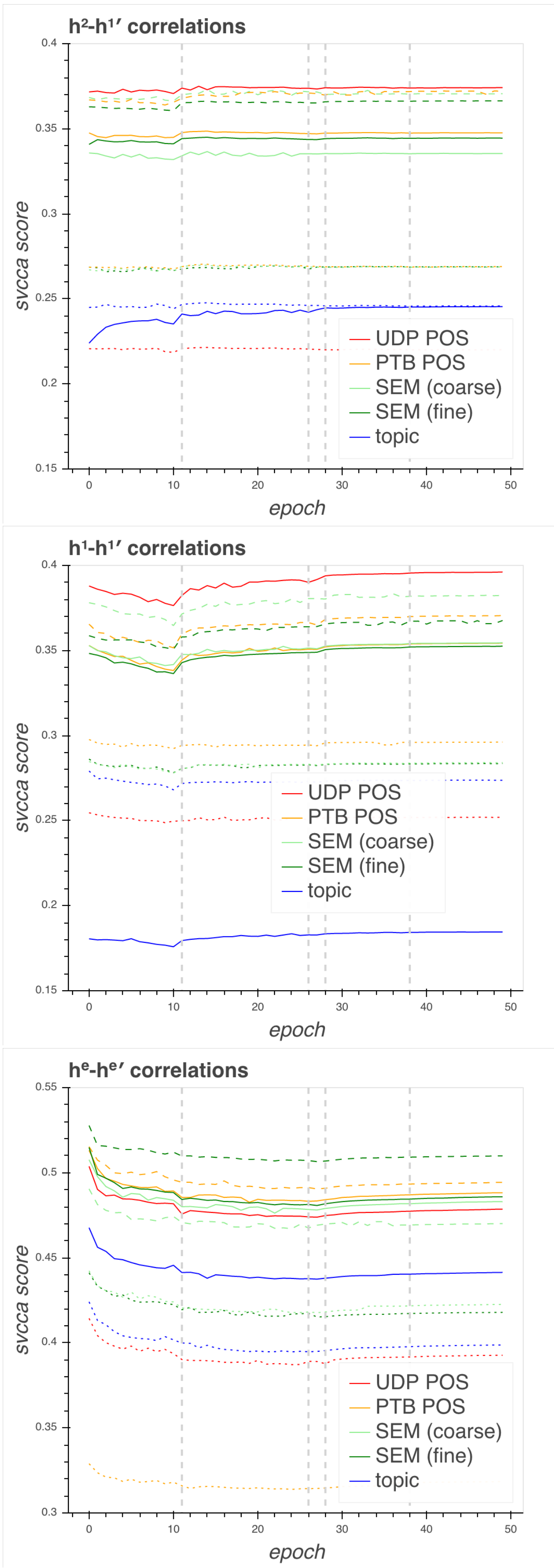


Figure 10: SVCCA correlation scores between LM and y_{t+1} tag predictor. Dotted lines use models trained on randomly shuffled the data. Dashed lines use GMB domain test data.

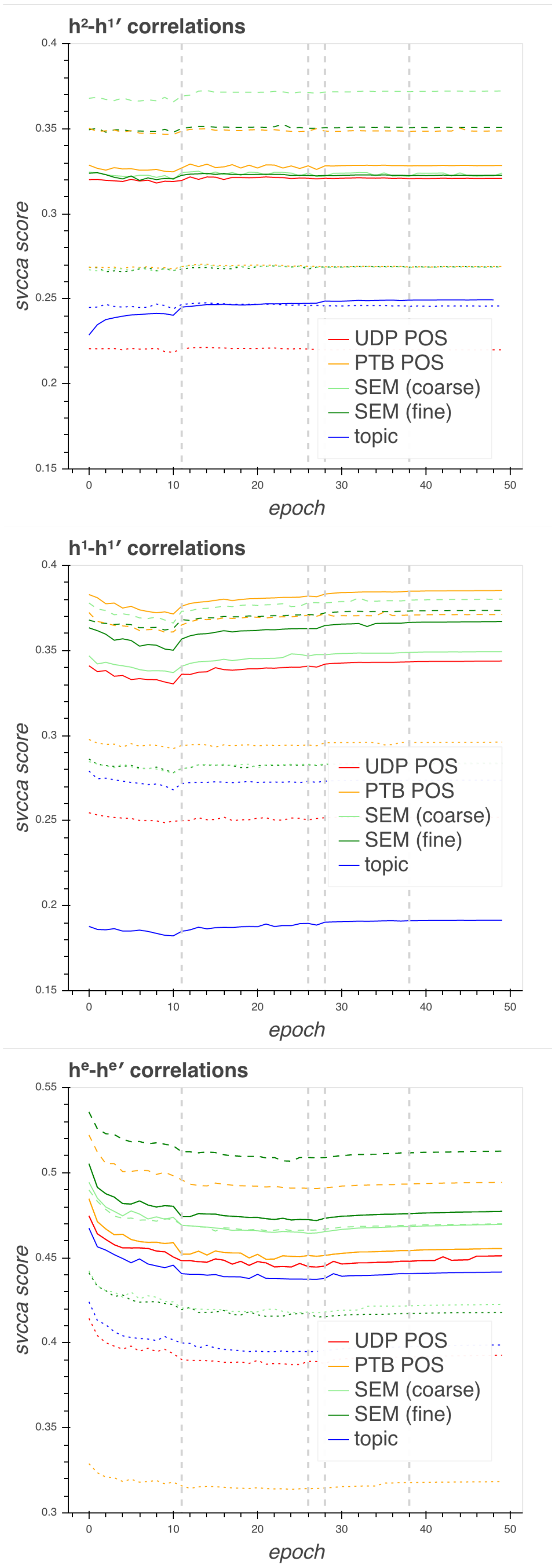


Figure 11: SVCCA correlation scores between LM and y_t tagger. Dotted lines use models trained on randomly shuffled the data. Dashed lines use GMB domain test data.

Using Large Corpus N-gram Statistics to Improve Recurrent Neural Language Models

Yiben Yang, Ji-Ping Wang
Department of Statistics
Northwestern University
Evanston, IL, 60208, USA
{yiben.yang,jzwang}@northwestern.edu

Doug Downey
Department of Electrical Engineering
& Computer Science
Northwestern University
Evanston, IL, 60208, USA
{d-downey}@northwestern.edu

Abstract

Recurrent neural network language models (RNNLM) form a valuable foundation for many NLP systems, but training the models can be computationally expensive, and may take days to train on a large corpus. We explore a technique that uses large corpus n-gram statistics as a regularizer for training a neural network LM on a smaller corpus. In experiments with the Billion-Word and Wikitext corpora, we show that the technique is effective, and more time-efficient than simply training on a larger sequential corpus. We also introduce new strategies for selecting the most informative n-grams, and show that these boost efficiency.

1 Introduction

Recurrent neural network models of language (RNNLMs) form a foundation for many natural language processing systems. However, the networks can be expensive to train: training a single model over several million tokens can take hours, and searching through the large hyperparameter space of RNNLMs often entails training and testing hundreds of different models. This makes it burdensome to experiment with new RNNLM architectures on large corpora, or to train RNNLMs for new textual domains.

RNNLMs are typically trained on sequential text. In this paper, we investigate how to efficiently augment the training of RNNLMs by regularizing the models to match n-gram statistics taken from a much larger corpus. The motivation is that large-corpus n-gram statistics may be informative to an RNNLM trained on a smaller sequential corpus, but unlike RNNLM training, n-gram statistics are inexpensive to compute even over large corpora. Moreover, the statistics only need to be computed once and can be re-used for training many different smaller-corpus RNNLMs.

Naively, regularizing an RNNLM to match a given set of n-gram statistics is non-trivial, because the marginal probabilities that n-gram statistics represent are not parameters of the RNNLM. In recent work, Noraset et al. (2018) showed that it was possible to regularize an RNNLM to match given n-gram statistics by training the network, when started from a zero state, to match each n-gram probability. However, the regularization approach in that work had tractability limitations—the time cost of the regularization was sufficiently high that using it was inferior to simply training the RNNLM on more sequential text.

In this paper, we present an efficient n-gram regularization technique and show that the technique can improve RNNLM training. Our method has three distinctions from previous work that provide efficiency. First, we prioritize regularizing only the n-grams that are most likely to improve the model, by focusing on cases where the RNNLM's sequential training corpus diverges significantly from the n-gram statistics. Secondly, we regularize the entire output softmax of the RNN to match given conditional n-gram statistics, which means we can impose a large number of statistical constraints using only one softmax evaluation. Finally, we use an ensemble of multiple loss functions in our regularizer, which provides an additional boost. In experiments, we show how n-gram regularization with these enhancements results in better models using the same amount of training time, compared to standard sequential training. We also plan to release our code base and to the research community.[1]

2 Methods

RNNLMs are trained to optimize a loss function L_d, which is defined as the average negative log-likelihood of a training corpus. We regularize the RNNLM to match n-gram statistics by introduc-

---

[1] https://github.com/yangyiben/Conditional-N-gram-Regularization

Proceedings of NAACL-HLT 2019, pages 3268–3273
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing another penalty term L_p to the loss function that captures how well the model matches n-gram statistics, giving a combined loss L:

$$L = L_d + \alpha L_p,$$

where α is a hyperparameter to control the regularization strength. We use the term *large corpus* to refer to the text utilized to compute the n-gram statistics. We expect the large corpus to be multiple orders of magnitude larger than the *small corpus* utilized for computing the RNN's sequential loss L_d. In the rest of this section, we first define what we mean by conditional n-gram statistics, and then present our regularization methods.

2.1 Conditional N-gram Statistics

An order-N n-gram is a sequence of N words. For a given corpus $\boldsymbol{c}$, we denote the kth distinct order-N n-gram $w_{k1}, w_{k2}, ..., w_{kN}$ as $\mathbf{w}_k^N$, and denote the corresponding order $N-1$ n-gram formed by the first $N-1$ words as $\mathbf{w}_k^{N-1}$. Here, in order to eliminate ambiguity, the notation $N-1$ is exclusively used to represent the prefix. For instance, $\mathbf{w}_k^3$ is the k-th trigram, while $\mathbf{w}_k^{4-1}$ is the trigram prefix of the k-th 4-gram. We define conditional n-gram statistics as the empirical conditional probabilities of observing each next word w_{kN} given the previous $N-1$ gram $\mathbf{w}_k^{N-1}$ for all ks :

$$\hat{P}(w_{kN}|\mathbf{w}_k^{N-1}) = \frac{count(\mathbf{w}_k^N)}{count(\mathbf{w}_k^{N-1})}, \forall \mathbf{w}_k^N \in \Omega(\boldsymbol{c}),$$

where w_{kN} and $\mathbf{w}_k^{N-1}$ are the Nth word and the corresponding previous $N-1$ gram of $\mathbf{w}_k^N$ respectively, and $\Omega(\boldsymbol{c})$ is the set of all unique $\mathbf{w}_k^N$s contained in the corpus $\boldsymbol{c}$. For a given RNNLM, the model conditional probability for some n-gram $\mathbf{w}_k^N$ is defined as:

$$P_\theta(w_{kN}|\mathbf{w}_k^{N-1}) = \mathbb{E}_h(P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h)),$$

where h is the model's hidden state prior to encountering the N-gram. However, it is difficult to express that expectation in terms of the model parameters, so we adopt the approach from Noraset et al. (2018), which has shown preliminary evidence of forming an effective regularizer:

$$\mathbb{E}_h(P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h)) \approx P_\theta(w_{kN}|\mathbf{w}_k^{N-1}, h_0),$$

where h_0 is a zero hidden state.

2.2 Conditional N-gram Regularization

We propose three forms of regularization loss functions that penalize the divergence of the model's conditional probabilities from the conditional n-gram statistics.

2.2.1 Mean Squared Log Probability Ratio

We denote our first penalty as L_p^{sq}, defined as:

$$L_p^{sq} = \frac{1}{\|R\|} \sum_{\mathbf{w}_k^N \in R} \left(log \frac{\hat{P}(w_{kN}|\mathbf{w}_k^{N-1})}{P_\theta(w_{kN}|\mathbf{w}_k^{N-1})} \right)^2,$$

where R is a set of n-grams $\mathbf{w}_k^N$ to regularize, and $\hat{P}$ and P_θ are conditional n-gram statistics and model conditional probabilities as defined in Section 2.1. This penalty is similar to that of (Noraset et al., 2018). However, instead of computing the loss with multiple forward passes for different $\mathbf{w}_k^N$s that have the same $\mathbf{w}_k^{N-1}$, we propose to only perform one forward pass for each $\mathbf{w}_k^{N-1}$, and regularize all subsequent Nth words. This makes our loss much more computationally efficient. As this penalty only accounts differences in point-wise probabilities for specific words w_N, it can be used for partially specified distributions where we only know the desired probabilities for some Nth words in a given context, but not the entire distribution.

2.2.2 Mean Kullback–Leibler Divergence

We denote our second penalty as L_p^{KL}, defined as:

$$P_{\mathbf{w}_k^{N-1}} = \hat{P}(w|\mathbf{w}_k^{N-1}), Q_{\mathbf{w}_k^{N-1}} = P_\theta(w|\mathbf{w}_k^{N-1})$$
$$L_p^{KL} = \frac{1}{\|R\|} \sum_{\mathbf{w}_k^{N-1} \in R} D_{KL}(P_{\mathbf{w}_k^{N-1}} || Q_{\mathbf{w}_k^{N-1}}),$$

where here R is a set of prefixes $\mathbf{w}_k^{N-1}$ to regularize. This penalty regularizes all possible subsequent words w_N, thus it only works for fully-specified reference distributions.

2.2.3 Combined Penalty

Because the above two penalty functions differ, we hypothesize that they may be complementary and propose a *combined* penalty $L_p^c = L_p^{sq} + L_p^{KL}$.

2.3 N-gram Selection Strategy

Note that we do one forward pass for each unique $\mathbf{w}_k^{N-1}$, so only the number of distinct prefixes $\mathbf{w}_k^{N-1}$ will significantly affect the computational cost of our regularization methods. Naively regularizing all unique prefixes in a large corpus usually requires a large number of forward passes, which could be expensive. We hypothesize that some prefixes are more useful than others, so we attempt to select the ones that will improve the model the most. We propose to select prefixes $\mathbf{w}_k^{N-1}$ that maximize the **E**xpected **L**og-likelihood **C**hange (ELC), defined as:

$$ELC(\mathbf{w}_k^{N-1}) = \sum \hat{P}(\mathbf{w}_k^N) log \frac{\hat{P}(w_{kN}|\mathbf{w}_k^{N-1})}{P_\theta(w_{kN}|\mathbf{w}_k^{N-1})}.$$

Ideally, P_θ would reflect the statistics of the RNNLM, updated during training, but these are expensive to obtain. Thus we propose to train a inexpensive n-gram model (Chen and Goodman, 1999) on the sequential corpus to serve as P_θ, and we use that to select a fixed set of n-grams to regularize.

3 Experiments

We now present our experiments measuring the effectiveness of conditional n-gram regularization.

3.1 Data and Settings

We experiment on a medium-size (2 layers with 650 hidden states) LSTM language model (Zaremba et al., 2014) over two corpora: Wikitext (Merity et al., 2016) and Google Billion-Word (Chelba et al., 2013) (1B). We adopt weight tying (Inan et al., 2016) and variational dropout (Gal and Ghahramani, 2016). All models are trained by SGD for 30 epochs with batch size 64 and truncated backpropagation (Mikolov et al., 2011) with 35 time steps. The learning rate starts at 20 and then is reduced to 5 at epoch 20. For the 1B corpus, we follow the same procedure in Yang et al. (2017) to generate training, validation and test sets, except that we use only the top 50K vocabulary terms. For the Wikitext corpus, we adopt the Wikitext-2 vocabulary. All of the RNNLM sequential training sets are small subsets sampled from the Wikitext-103 and 1B training corpora.

For each dataset, we use the whole training set as the large corpus for building our reference conditional n-gram statistics. In this study, we only consider conditional trigram regularization for all experiments. The regularization takes additional time during training. To enable a fair experiment, we equalize the training time between regularized models and the baselines, by providing the baselines with more sequential training data than the regularized models. The three proposed penalties are almost equally fast, thus they can be compared against the same baseline. Unlike RNNs, there are no hyperparameter settings or decisions involved in computing the n-gram counts, so this can be done once and re-used across the many RNN training runs that adequate hyperparameter search for RNNs often entails. Moreover, counting n-grams is fast. We approximate that it takes about one minute to obtain n-gram statistics from the Wikitext-103 training corpus, for example. Thus, we set up our experiments to equalize neural network training time, and we ignore the small one-time cost of computing n-gram statistics.

We fix the number of bigrams per batch to be 500, and employ the proposed strategy to select the top X most informative bigrams (X depends on the number of batches in sequential data). Finally, instead of tuning the regularization strength hyperparameter α for each setting, we fix α to be 1.0, 0.75 and 0.5 for the three sequential data sizes based on the heuristic that larger sequential data may need less regularization. More carefully tuning the regularization strength might yield somewhat better results for our methods. Also, using different regularization strengths in the combined penalty might further improve results.

3.2 Results

In Table 1, we compare the performance of our proposed methods against equal-time controlled baselines under different token sizes for both the Wikitext and 1B data sets. In the table, the numbers in the column headings indicate the token count of the sequential corpus used to train the regularized methods. The baselines train on larger corpora, to ensure an equal-time setting as described above. All regularized models outperform their baseline counterparts for all token sizes. Among them, the models regularized by the combined penalty consistently perform best. This illustrates that conditional n-gram regularization is effective at incorporating large-corpus statistics to improve an RNNLM trained on a relatively small corpus. Performing regularization using the combined selection strategy yields more accurate models compared to simply training on a larger sequential corpus.

In Figure 1, we plot validation perplexities after each training epoch for models trained on the 5M-token 1B corpus, against an equal-time baseline. The plot shows that the relative performance of the methods remains similar across training epochs.

Methods	Wikitext			Google 1B		
	500K	1M	2M	2.5M	5M	10M
baseline	161	110	76	110	90	81
sq_log	122	91	72	98	86	80
KL	137	100	77	100	86	78
combined	**112**	**86**	**69**	**94**	**83**	**77**

Table 1: Test perplexities of different methods and sequential training token sizes. sq_log: mean squared log probability ratio penalty. KL: mean Kullback-Leibler divergence penalty. Models with the combined penalty achieve the lowest perplexities.

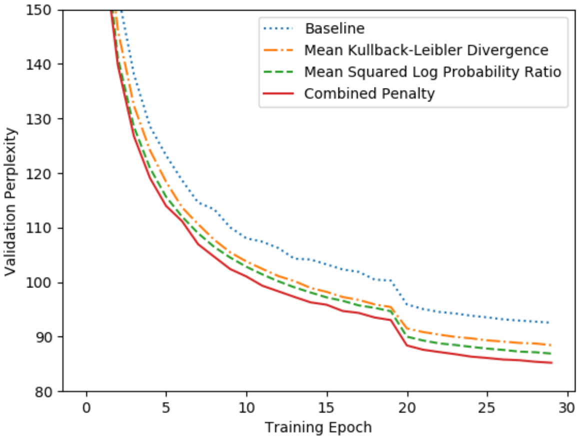


Figure 1: Validation Perplexities on the 5M-token 1B Dataset. The combined penalty performs best.

3.3 Analysis

In Table 2, we demonstrate how the number of regularized bigrams affects the performance. Here, the regularized models always train on wikitext-2 as a sequential corpus, and the baseline trains on larger corpora as the fractions of bigrams increases. The regularized model achieves 66 test perplexity on Wikitext-2 corpus, which is about 2.7 points worse than a state of the art mixture of softmax model (Yang et al., 2017) even though our model has fewer parameters (25M vs. 35M). Including more bigrams helps lower the perplexity, while it also demands extra computational time. ELC performs best when using less than 20% of the bigrams. Regularizing randomly selected n-grams does not outperform the equal-time baseline, indicating that not all n-grams are equally useful. In order to be time efficient, it is important to select informative n-grams, and ELC is an effective measure of informativeness.

% of total bigrams	baseline	random	ELC
0%	86	86	86
5%	76	83	**69**
10%	71	77	**67**
20%	**66**	71	67
40%	**61**	68	66

Table 2: Test perplexities of RNNLMs trained on Wikitext-2 regularized with different numbers of bigrams using the combined penalty. Baseline indicates an equal-time controlled baseline, random is a regularized model with bigrams selected randomly, and ELC indicates our proposed strategy.

In Table 3, we consider ensembling a standard RNN with a KN-smoothed trigram model. This achieves a ppl of 65, but requires 51M more parameters, whereas our regularization with the n-grams achieves most of the perplexity improvement at the cost of zero additional parameters. Further, somewhat surprisingly we find that an ensemble of our regularized RNNLM with the n-gram model achieves much better perplexity of 59.

Models	Test PPL
Unregularized RNNLM	86
Unregularized RNNLM + Trigram KN	65
Regularized RNNLM	67
Regularized RNNLM + Trigram KN	**59**

Table 3: Test perplexities of RNN-LMs trained with and without regularization on the Wikitext-2 corpus, ensembled with a KN-smoothed trigram model trained on the Wikitext-103 corpus.

Another possible way of efficiently utilizing a large corpus would involve training a Word2vec model on the large corpus, and using the pre-trained embeddings within a RNNLM trained on a small corpus. This approach can utilize larger corpora since training a Word2vec model is much faster than training a RNNLM. However, in our preliminary experiments with this approach, we did not observe any improvement when using word embeddings trained on a large corpus. Further experiments with variants of this approach are an item of future work.

4 Related Work

Chelba et al. (2017) trained large-order n-gram models using a recurrent neural network trained over limited context to produce the conditional probability estimates. Our regularizer is trained in a similar way, but by contrast we are focused on how the regularizer can be used in concert with standard sequential RNNLM training to improve the training procedure. We introduce n-gram selection techniques and distinct loss functions that increase the effectiveness of the combined training. Ganchev et al. (2010) presents a posterior regularization method for restricting posterior distributions of probabilistic models with latent variables to obey predefined constraints using the EM algorithm. This approach shares our goal of imposing constraints on probabilistic models, but we focus on RNNLMs, which do not estimate distributions over latent state variables and are not trained using EM. Finally, Mikolov et al. (2011), Józefowicz et al. (2016) and Chelba et al. (2013) trained ensembles of RNNLMs and KN-smoothed n-gram models, and showed that one can obtain a better model when ensembling RNNLMs with n-gram models. Our experiments show that compared to ensemble methods, conditional n-gram regularization achieves similar results at the cost of zero additional parameters, and can perform even better when combined with ensembling.

5 Conclusion and Future Work

In this paper, we have proposed methods to utilize using large corpus n-gram statistics to regularize RNNLMs trained on a smaller corpus. Our experiments demonstrate that the proposed regularization penalties are effective in improving model performance, and can be more time efficient than training RNNLMs on a larger sequential corpus. Selecting informative n-grams is shown to be important. In future work, we would like to obtain a better theoretical understanding of why starting the RNNLM from a zero state forms an effective n-gram regularizer. We would also like to extend our regularization approach to BiLSTMs (Peters et al., 2017) and Transformers (Alec Radford and Sutskever, 2018; Devlin et al., 2018).

Acknowledgments

This work was supported in part by NSF Grant IIS-1351029 and the Allen Institute for Artificial Intelligence.

References

Tim Salimans Alec Radford, Karthik Narasimhan and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI.*

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *CoRR*, abs/1703.10724.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531.

Thanapon Noraset, Doug Downey, and Lidong Bing. 2018. Estimating marginal probabilities of n-grams for recurrent neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2930–2935. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, abs/1711.03953.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Continual Learning for Sentence Representations Using Conceptors

Tianlin Liu
Department of Computer Science and
Electrical Engineering
Jacobs University Bremen
28759 Bremen, Germany
`t.liu@jacobs-university.de`

Lyle Ungar and **João Sedoc**
Department of Computer and
Information Science
University of Pennsylvania
Philadelphia, PA 19104
`{ungar, joao}@cis.upenn.edu`

Abstract

Distributed representations of sentences have become ubiquitous in natural language processing tasks. In this paper, we consider a continual learning scenario for sentence representations: Given a sequence of corpora, we aim to optimize the sentence encoder with respect to the new corpus while maintaining its accuracy on the old corpora. To address this problem, we propose to initialize sentence encoders with the help of corpus-independent features, and then sequentially update sentence encoders using Boolean operations of *conceptor* matrices to learn corpus-dependent features. We evaluate our approach on semantic textual similarity tasks and show that our proposed sentence encoder can continually learn features from new corpora while retaining its competence on previously encountered corpora.

1 Introduction

Distributed representations of sentences are essential for a wide variety of natural language processing (NLP) tasks. Although recently proposed sentence encoders have achieved remarkable results (e.g., (Yin and Schütze, 2015; Arora et al., 2017; Cer et al., 2018; Pagliardini et al., 2018)), most, if not all, of them are trained on *a priori* fixed corpora. However, in open-domain NLP systems such as conversational agents, oftentimes we are facing a dynamic environment, where training data are accumulated sequentially over time and the distributions of training data vary with respect to external input (Lee, 2017; Mathur and Singh, 2018). To effectively use sentence encoders in such systems, we propose to consider the following *continual sentence representation learning task*: Given a sequence of corpora, we aim to train sentence encoders such that they can continually learn features from new corpora while retaining strong performance on previously encountered corpora.

Toward addressing the continual sentence representation learning task, we propose a simple sentence encoder that is based on the summation and linear transform of a sequence of word vectors aided by matrix conceptors. Conceptors have their origin in reservoir computing (Jaeger, 2014) and recently have been used to perform continual learning in deep neural networks (He and Jaeger, 2018). Here we employ Boolean operations of conceptor matrices to update sentence encoders over time to meet the following desiderata:

1. *Zero-shot learning*. The initialized sentence encoder (no training corpus used) can effectively produce sentence embeddings.
2. *Resistant to catastrophic forgetting*. When the sentence encoder is adapted on a new training corpus, it retains strong performances on old ones.

The rest of the paper is organized as follows. We first briefly review a family of linear sentence encoders. Then we explain how to build upon such sentence encoders for continual sentence representation learning tasks, which lead to our proposed algorithm. Finally, we demonstrate the effectiveness of the proposed method using semantic textual similarity tasks.[1]

Notation We assume each word w from a vocabulary set V has a real-valued word vector $v_w \in \mathbb{R}^n$. Let $p(w)$ be the monogram probability of a word w. A corpus D is a collection of sentences, where each sentence $s \in D$ is a multiset of words (word order is ignored here). For a collection of vectors $Y = \{y_i\}_{i \in I}$, where $y_i \in \mathbb{R}^l$

---

[1] Our codes are available on GitHub `https://github.com/liutianlin0121/contSentEmbed`

Proceedings of NAACL-HLT 2019, pages 3274–3279
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

for i in an index set I with cardinality $|I|$, we let $[y_i]_{i\in I} \in \mathbb{R}^{l\times|I|}$ be a matrix whose columns are vectors $y_1, \cdots, y_{|I|}$. An identity matrix is denoted by $\mathbf{I}$.

2 Linear sentence encoders

We briefly overview "linear sentence encoders" that are based on linear algebraic operations over a sequence of word vectors. Among different linear sentence encoders, the smoothed inverse frequency (SIF) approach (Arora et al., 2017) is a prominent example – it outperforms many neural-network based sentence encoders on a battery of NLP tasks (Arora et al., 2017).

Derived from a generative model for sentences, the SIF encoder (presented in Algorithm 1) transforms a sequence of word vectors into a sentence vector with three steps. First, for each sentence in the training corpus, SIF computes a weighted average of word vectors (line 1-3 of Algorithm 1); next, it estimates a "common discourse direction" of the training corpus (line 4 of Algorithm 1); thirdly, for each sentence in the testing corpus, it calculates the weighted average of the word vectors and projects the averaged result away from the learned common discourse direction (line 5-8 of Algorithm 1). Note that this 3-step paradigm is slightly more general than the original one presented in (Arora et al., 2017), where the training and the testing corpus is assumed to be the same.

```
Algorithm 1: SIF sentence encoder.
Input : A training corpus $D$; a testing corpus $G$; parameter $a$, monogram
        probabilities $\{p(w)\}_{w\in V}$ of words
1 for sentence $s \in D$ do
2     $q_s \leftarrow \frac{1}{|s|}\sum_{w\in s}\frac{a}{p(w)+a}v_w$
3 end
4 Let $u$ be the first singular vector of $[q_s]_{s\in D}$.
5 for sentence $s \in G$ do
6     $q_s \leftarrow \frac{1}{|s|}\sum_{w\in s}\frac{a}{p(w)+a}v_w$
7     $f_s^{\text{SIF}} \leftarrow q_s - uu^\top q_s$.
8 end
Output: $\{f_s^{\text{SIF}}\}_{s\in G}$
```

Building upon SIF, recent studies have proposed further improved sentence encoders (Khodak et al., 2018; Pagliardini et al., 2018; Yang et al., 2018). These algorithms roughly share the core procedures of SIF, albeit using more refined methods (e.g., softly remove more than one common discourse direction).

3 Continual learning for linear sentence encoders

In this section, we consider how to design a linear sentence encoder for continual sentence representation learning. We observe that common discourse directions used by SIF-like encoders are estimated from the training corpus. However, incrementally estimating common discourse directions in continual sentence representation learning tasks might not be optimal. For example, consider that we are sequentially given training corpora of `tweets` and `news article`. When the first `tweets` corpus is presented, we can train a SIF sentence encoder using `tweets`. When the second `news article` corpus is given, however, we will face a problem on how to exploit the newly given corpus for improving the *trained* sentence encoder. A straightforward solution is to first combine the `tweets` and `news article` corpora and then train a new encoder from scratch using the combined corpus. However, this paradigm is not efficient or effective. It is not efficient in the sense that we will need to re-train the encoder from scratch every time a new corpus is added. Furthermore, it is not effective in the sense that the common direction estimated from scratch reflects a compromise between tweets and news articles, which might not be optimal for either of the stand-alone corpus. Indeed, it is possible that larger corpora will swamp smaller ones.

To make the common discourse learned from one corpus more generalizable to another, we propose to use the conceptor matrix (Jaeger, 2017) to characterize and update the common discourse features in a sequence of training corpora.

3.1 Matrix conceptors

In this section, we briefly introduce matrix conceptors, drawing heavily on (Jaeger, 2017; He and Jaeger, 2018; Liu et al., 2019). Consider a set of vectors $\{x_1, \cdots, x_n\}$, $x_i \in \mathbb{R}^N$ for all $i \in \{1, \cdots, n\}$. A conceptor matrix is a regularized identity map that minimizes

$$\frac{1}{n}\sum_{i=1}^{n}\|x_i - Cx_i\|_2^2 + \alpha^{-2}\|C\|_{\text{F}}^2. \qquad (1)$$

where $\|\cdot\|_{\text{F}}$ is the Frobenius norm and α^{-2} is a scalar parameter called *aperture*. It can be shown that C has a closed form solution:

$$C = \frac{1}{n}XX^\top(\frac{1}{n}XX^\top + \alpha^{-2}I)^{-1}, \qquad (2)$$

where $X = [x_i]_{i\in\{1,\cdots,n\}}$ is a data collection matrix whose columns are vectors from $\{x_1, \cdots, x_n\}$. In intuitive terms, C is a soft projection matrix on the linear subspace where the typical components of x_i samples lie. For convenience in notation, we may write $C(X, \alpha)$ to stress the dependence on X and α.

Conceptors are subject to most laws of Boolean logic such as NOT $\neg$, AND $\wedge$ and OR $\vee$. For two conceptors C and B, we define the following operations:

$$\neg C := \mathbf{I} - C, \qquad (3)$$

$$C \wedge B := (C^{-1} + B^{-1} - \mathbf{I})^{-1} \qquad (4)$$

$$C \vee B := \neg(\neg C \wedge \neg B) \qquad (5)$$

Among these Boolean operations, the OR operation $\vee$ is particularly relevant for our continual sentence representation learning task. It can be shown that $C \vee B$ is the conceptor computed from the union of the two sets of sample points from which C and B are computed. Note that, however, to calculate $C \vee B$, we only need to know two matrices C and B and do not have to access to the two sets of sample points from which C and B are computed.

3.2 Using conceptors to continually learn sentence representations

We now show how to sequentially characterize and update the common discourse of corpora using the Boolean operation of conceptors. Suppose that we are sequentially given M training corpora $D^1, \cdots, D^M$, presented one after another. Without using any training corpus, we first initialize a conceptor which characterizes the corpus-independent common discourse features. More concretely, we compute $C^0 := C([v_w]_{w\in Z}, \alpha)$, where $[v_w]_{w\in Z}$ is a matrix of column-wisely stacked word vectors of words from a stop word list Z and α is a hyper-parameter. After initialization, for each new training corpus D^i ($i = 1, \cdots, M$) coming in, we compute a new conceptor $C^{\text{temp}} := C([q_s]_{s\in D^i}, \alpha)$ to characterize the common discourse features of corpus D^i, where those q_s are defined in the SIF Algorithm 1. We can then use Boolean operations of conceptors to compute $C^i := C^{\text{temp}} \vee C^{i-1}$, which characterizes common discourse features from the new corpus as well as the old corpora. After all M corpora are presented, we follow the SIF paradigm and use C^M to remove common discourse features from (potentially unseen) sentences. The above outlined conceptor-aided (CA) continual sentence representation learning method is presented in Algorithm 2.

Algorithm 2: CA sentence encoder.

```
Input : A sequence of $M$ training corpora
        $\mathcal{D} = \{D^1, \cdots, D^M\}$; a testing
        corpus $G$; hyper-parameters $a$ and $\alpha$;
        word probabilities $\{p(w)\}_{w\in V}$; stop
        word list $Z$.
1  $C^0 \leftarrow C([v_w]_{w\in Z}, \alpha)$ .
2  for corpus index $i = 1, \cdots, M$ do
3      for sentence $s \in D^i$ do
4          $q_s \leftarrow \frac{1}{|s|}\sum_{w\in s}\frac{a}{p(w)+a}v_w$
5      end
6      $C^{\text{temp}} \leftarrow C([q_s]_{s\in D^i}, \alpha)$
7      $C^i \leftarrow C^{\text{temp}} \vee C^{i-1}$
8  end
9  for $s \in G$ do
10     $q_s \leftarrow \frac{1}{|s|}\sum_{w\in s}\frac{a}{p(w)+a}v_w$
11     $f_s^{\text{CA}} \leftarrow q_s - C^M q_s$
12 end
Output: $\{f_s^{\text{CA}}\}_{s\in G}$
```

A simple modification of Algorithm 2 yields a "zero-shot" sentence encoder that requires only pre-trained word embeddings and no training corpus: we can simply skip those corpus-dependent steps (line 2-8) and use C^0 in place of C^M in line 11 in Algorithm 2 to embed sentences. This method will be referred to as "zero-shot CA."

4 Experiment

We evaluated our approach for continual sentence representation learning using semantic textual similarity (STS) datasets (Agirre et al., 2012, 2013, 2014, 2015, 2016). The evaluation criterion for such datasets is the Pearson correlation coefficient (PCC) between the predicted sentence similarities and the ground-truth sentence similarities. We split these datasets into five corpora by their genre: news, captions, wordnet, forums, tweets (for details see appendix). Throughout this section, we use publicly available 300-

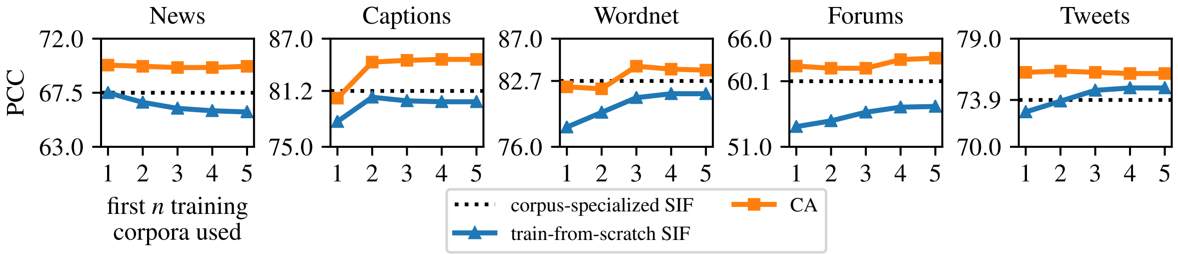


Figure 1: PCC results of STS datasets. Each panel shows the PCC results of a testing corpus (specified as a subtitle) as a function of increasing numbers of training corpora used. The setup of this experiment mimics (Zenke et al., 2017, section 5.1).

	News	Captions	WordNet	Forums	Tweets
av. train-from-scratch SIF	66.5	79.7	80.3	55.5	74.2
zero-shot CA	65.6	79.8	82.5	61.5	75.2
av. CA	**69.7**	**83.8**	**83.2**	**62.5**	**76.2**

Table 1: Time-course averaged PCC of train-from-scratch SIF and conceptor-aided (CA) methods, together with the result of zero-shot CA. Best results are in boldface and the second best results are underscored.

dimensional GloVe vectors (trained on the 840 billion token Common Crawl) (Pennington et al., 2014). Additional experiments with Word2Vec (Mikolov et al., 2013), Fasttext (Bojanowski et al., 2017), Paragram-SL-999 (Wieting et al., 2015) are in the appendix.

We use a standard continual learning experiment setup (cf. (Zenke et al., 2017, section 5.1)) as follows. We sequentially present the five training datasets in the order[2] of news, captions, wordnet, forums, and tweets, to train sentence encoders. Whenever a new training corpus is presented, we train a SIF encoder from scratch[3] (by combining all available training corpora which have been already presented) and then test it on each corpus. At the same time, we incrementally adapt a CA encoder[4] using the newly presented corpus and test it on each corpus. The lines of each panel of Figure 1 show the test results of SIF and CA on each testing corpus (specified as the panel subtitle) as a function of the number of training corpora used (the first n corpora of news, captions, wordnet, forums, and tweets for this experiment). To give a concrete example, consider the blue line in the first panel of Figure 1. This line shows the test PCC scores (y-axis) of SIF encoder on the news corpus when the number of training corpora increases (x-axis). Specifically, the left-most blue dot indicates the test result of SIF encoder on news corpus when trained on news corpus itself (that is, the first training corpus is used); the second point indicates the test results of SIF encoder on news corpus when trained on news and captions corpora (i.e., the first *two* training corpora are used); the third point indicates the test results of SIF encoder on news corpus when trained on news, captions, and wordnet corpora (that is, the first *three* training corpora are used), so on and so forth. The dash-lines in panels show the results of a corpus-specialized SIF, which is trained and tested on the same corpus, i.e., as done in (Arora et al., 2017, section 4.1). We see that the PCC results of CA are better and more "forgetting-resistant" than train-from-scratch SIF throughout the time course where more training data are incorporated. Consider, for example, the test result of news corpus (first panel) again. As more and more training corpora are used, the performance of train-from-scratch SIF drops with a noticeable slope; by contrast, the performance CA drops only slightly.

As remarked in the section 3.2, with a simple modification of CA, we can perform zero-shot sentence representation learning without using any training corpus. The zero-shot learning results are

---

[2] The order can be arbitrary. Here we ordered the corpora from the one with the largest size (news) to the smallest size (tweets). The results from reversely ordered corpora are reported in the appendix.

[3] We use $a = 0.001$ as in (Arora et al., 2017). The word frequencies are available at the GitHub repository of SIF.

[4] We used hyper-parameter $\alpha = 1$. Other parameters are set to be the same as SIF.

presented in Table 1, together with the time-course averaged results of CA and train-from-scratch SIF (i.e., the averaged values of those CA or SIF scores in each panel of Figure 1). We see that the averaged results of our CA method performs the best among these three methods. Somewhat surprisingly, the results yielded by zero-shot CA are better than the averaged results of train-from-scratch SIF in most of the cases.

We defer additional experiments to the appendix, where we compared CA against more baseline methods and use different word vectors other than GloVe to carry out the experiments.

5 Conclusions and future work

In this paper, we formulated a continual sentence representation learning task: Given a consecutive sequence of corpora presented in a time-course manner, how can we extract useful sentence-level features from new corpora while retaining those from previously seen corpora? We identified that the existing linear sentence encoders usually fall short at solving this task as they leverage on "common discourse" statistics estimated based on a priori fixed corpora. We proposed two sentence encoders (CA encoder and zero-shot CA encoder) and demonstrate their the effectiveness at the continual sentence representation learning task using STS datasets.

As the first paper considering continual sentence representation learning task, this work has been limited in a few ways – it remains for future work to address these limitations. First, it is worthwhile to incorporate more benchmarks such as GLUE (Wang et al., 2019) and SentEval (Conneau and Kiela, 2018) into the continual sentence representation task. Second, this work only considers the case of linear sentence encoder, but future research can attempt to devise (potentially more powerful) non-linear sentence encoders to address the same task. Thirdly, the proposed CA encoder operates at a corpus level, which might be a limitation if boundaries of training corpora are ill-defined. As a future direction, we expect to lift this assumption, for example, by updating the common direction statistics at a sentence level using Autoconceptors (Jaeger, 2014, section 3.14). Finally, the continual learning based sentence encoders should be applied to downstream applications in areas such as open domain NLP systems.

Acknowledgements

The authors thank anonymous reviewers for their helpful feedback. This work was partially supported by João Sedoc's Microsoft Research Dissertation Grant.

References

E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigaua, L. Uriaa, and J. Wiebeg. 2015. Semeval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation*, pages 252–263.

E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation*, pages 81–91.

E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California.

E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo. 2013. Sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 32–43.

E. Agirre, M. Diab, D. Cer, and A. Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. Arora, Y. Liang, and T. Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

A. Conneau and D. Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

X. He and H. Jaeger. 2018. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*.

H. Jaeger. 2014. Controlling recurrent neural networks by conceptors. Technical report, Jacobs University Bremen.

H. Jaeger. 2017. Using conceptors to manage neural long-term memories for temporal patterns. *Journal of Machine Learning Research*, 18(13):1–43.

M. Khodak, N. Saunshi, Y. Liang, T. Ma, B. Stewart, and S. Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. *In the Proceedings of ACL*.

S. Lee. 2017. Toward continual learning for conversational agents. Technical report, Microsoft Research AI - Redmond.

T. Liu, L. Ungar, and J. Sedoc. 2019. Unsupervised post-processing of word vectors via conceptor negation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-2019), Honolulu*.

V. Mathur and A. Singh. 2018. The rapidly changing landscape of conversational agents.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

M. Pagliardini, P. Gupta, and M. Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *Proceedings of the NAACL 2018*.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Z. Yang, C. Zhu, and W. Chen. 2018. Zero-training sentence embedding via orthogonal basis.

W. Yin and H. Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the NAACL HLT 2015*, pages 901–911.

F. Zenke, B. Poole, and S. Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995, International Convention Centre, Sydney, Australia. PMLR.

Relation Discovery with Out-of-Relation Knowledge Base as Supervision

Yan Liang[1], Xin Liu[1], Jianwen Zhang[2], and Yangqiu Song[1]
[1]Department of CSE, Hong Kong University of Science and Technology, HK
[2]Microsoft, USA
[1]{`yliangav, xliucr, yqsong`}`@cse.ust.hk`
[2]{`jiazhan`}`@microsoft.com`

Abstract

Unsupervised relation discovery aims to discover new relations from a given text corpus without annotated data. However, it does not consider existing human annotated knowledge bases even when they are relevant to the relations to be discovered. In this paper, we study the problem of how to use out-of-relation knowledge bases to supervise the discovery of unseen relations, where out-of-relation means that relations to discover from the text corpus and those in knowledge bases are not overlapped. We construct a set of constraints between entity pairs based on the knowledge base embedding and then incorporate constraints into the relation discovery by a variational auto-encoder based algorithm. Experiments show that our new approach can improve the state-of-the-art relation discovery performance by a large margin.

1 Introduction

Relation extraction has been widely used for many applications, such as knowledge graph construction (Dong et al., 2014), information retrieval (Liu et al., 2014), and question answering (Ravichandran and Hovy, 2002). Traditional supervised approaches require direct annotation on sentences with a relatively small number of relations (Roth and Yih, 2002; Kambhatla, 2004).[1] With the development of large-scale knowledge bases (KBs) such as Freebase (Bollacker et al., 2008), relation extraction has been extended to larger scales comparable to KBs using the distant supervision (Mintz et al., 2009). However, when the training corpus does not support the annotated relations showing in the KB, such approach could fail to find sufficient training examples. Distant supervision assumption can be violated by up to 31% for some relations when aligning to NYT corpus (Riedel et al., 2010). More importantly, either traditional supervised learning or distantly supervised learning cannot discover new relations unseen in the training phase.

Unsupervised relation discovery tries to overcome the shortcomings of supervised or distantly supervised learning approaches. Existing approaches either extract surface or syntactic patterns from sentences and use relation expressions as predicates (which result in many noisy relations) (Etzioni et al., 2004; Banko et al., 2007), or cluster the relation expressions based on the extracted triplets to form relation clusters (Yao et al., 2011, 2012; Marcheggiani and Titov, 2016). However, these approaches do not use existing high-quality and large-scale KBs when they are relevant to the relations to be discovered.

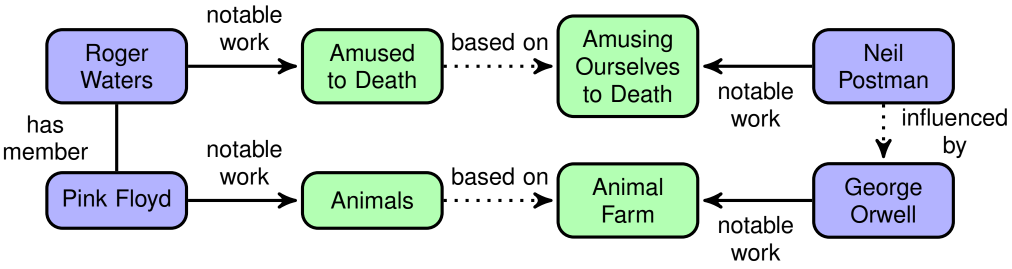
In this paper, we consider a new relation discovery problem where both the training corpus for relation clustering and a KB are available, but the relations in the training corpus and those in the KB are not overlapped. As shown in Figure 1, in the KB, we have entities `Pink Floyd`, `Animals`, etc., with some existing relations *notable_work* and *has_member* in the KB. However, when doing relation discovery, we can only get supporting sentences that suggest new relations *based_on* and *influenced_by*. This is a common and practical problem since predicates in KBs are limited to the annotator defined relations while the real relations in the world are always open and creative.

It is challenging when there is no overlapped relation between target relation clusters and the KB because in this case the KB is not a direct supervision. But if target relation clusters and the KB share some entities, we can use the shared entities as a bridge to introduce indirect supervision for the relation discovery problem. Specifically, we build constraints between pairs of tuples based on the

---

[1] We distinguish a relation (e.g., a predicate in a knowledge base) from the relation expression (e.g., the text surface between entities in a sentence) throughout the paper.

Proceedings of NAACL-HLT 2019, pages 3280–3290
Minneapolis, Minnesota, June 2 - June 7, 2019.



Amused to Death was inspried by Neil Postman's book **Amusing Ourselves to Death**.

Loosely based on George Orwell's **Animal Farm**, **Animals** describe various classes in society as different kinds of animals

Postman distinguishes the **Orwellian** vision of the future, from that offered by Aldous Huxley in Brave New World.

Figure 1: An illustration of our new relation discovery setting. The konwledge base contains relations *notable_work* and *has_member*. However, the training corpus to perform relation discovery only contains new relations *based_on* and *influenced_by*.

KB. For example, in Figure 1, when we cluster the *based_on* relation, we can evaluate the similarity between the tuple (`Animals`, `Animal Farm`) and the tuple (`Amused to Death`, `Amusing Ourselves to Death`) based on the KB. If the KB tells us these two pairs of tuples are close to each other, then we put a constraint to force our relation clustering algorithm to group them together.

We use the discrete-state variational autoencoder (DVAE) framework (Marcheggiani and Titov, 2016) as our base relation discovery model since this framework is flexible to incorporate different features and currently the state-of-the-art. We use KB embedding (Bordes et al., 2013) to obtain entity embeddings in the KB and use entity embeddings to evaluate the similarity between a pair of tuples. Then constraints are constructed and incorporated into the DVAE framework in a way inspired by the must-link and cannot-link based constrained clustering (Basu et al., 2004). We show that with no overlapped relations between the KB and the training corpus, we can improve the relation discovery by a large margin.

Our contributions are summarized as follows.

- We study a new prevalent but challenging task of relation discovery where the training corpus and the KB have no overlapped relation.
- We propose a new kind of indirect supervision to relation discovery which is built based on pairwise constraints between two tuples.
- We show promising results using existing relation discovery datasets to demonstrate the effectiveness of our proposed learning algorithm for the new relation discovery task.

The code we used to train and evaluate our models is available at `https://github.com/HKUST-KnowComp/RE-RegDVAE`.

2 Problem Definition

We use $\mathcal{X}$ to denote the set of all training sentences. $\mathcal{V}$ is the set of named entities that are recognized by an NER system in $\mathcal{X}$, and (e_1, e_2) is the pair of first and second entities in a given sentence $x \in \mathcal{X}$. $\mathcal{R}_\mathcal{X}$ is the set of relation labels for $\mathcal{X}$. In addition, there exists an external knowledge base $\mathcal{G}(\mathcal{E}_\mathcal{G}, \mathcal{T}_\mathcal{G})$, consisting of a set of entities $\mathcal{E}_\mathcal{G}$ and relations $\mathcal{R}_\mathcal{G}$ and triplets $\mathcal{T}_\mathcal{G}$ where a triplet consists of two entities with their relation.

Our model is a relation extractor to predict the underlying semantic relation $r \in \mathcal{R}_\mathcal{X}$ given sentences $\mathcal{X}$, with the help of $\mathcal{G}(\mathcal{E}_\mathcal{G}, \mathcal{T}_\mathcal{G})$. In particular, we focus on the challenging scenario where $\mathcal{R}_\mathcal{X} \cap \mathcal{R}_\mathcal{G} = \emptyset$.

3 Model

In this section, we first review the discrete-state variational autoencoder (DVAE) in §3.1. Then we introduce our new framework in §3.2.

3.1 DVAE for Relation Discovery

Assuming that we perform generative modeling, where each latent relation r follows a uniform prior distribution $p_u(r)$, we follow (Marcheggiani and Titov, 2016) to optimize a pseudo-likelihood:

$$\mathcal{L}(\theta) = \log \sum_{r \in \mathcal{R}_{\mathcal{X}}} p(e_i, e_{-i} | r, \theta) p_u(r) \tag{1}$$

$$\approx \sum_{i=1}^{2} \log \sum_{r \in \mathcal{R}_{\mathcal{X}}} p(e_i | e_{-i}, r, \theta) p_u(r), \tag{2}$$

where e_i and e_{-i} are entities, $i \in \{1, 2\}$ and e_{-i} denotes the complement $\{e_1, e_2\} \setminus \{e_i\}$. $p(e_i|e_{-i}, r, \theta)$ is the probability of one entity given another entity as well as the relation, where θ denotes the set of parameters. Note that this probability p is defined on the triplet (e_1, r, e_2) which is universal across different sentences containing the two entities.

The pseudo-likelihood $\mathcal{L}(\theta)$ can be lower-bounded based on Jensen's inequality through a variational posterior $q(r|x, \psi)$:

$$\begin{aligned}\mathcal{L}(\theta, \psi) = & \sum_{i=1}^{2} \sum_{r \in \mathcal{R}_{\mathcal{T}}} q(r|x, \psi) \log p(e_i | e_{-i}, r, \theta) \\ & + \alpha H \left[q(r|x, \psi) \right],\end{aligned} \tag{3}$$

where $q(r|x, \psi)$ predicts the relation based on the whole sentence x as an input and ψ as the set of parameters. H is the entropy to regularize the probability distribution q, and α is the hyper-parameter to balance the regularization strength.

This model consists of two components, an encoder $q(r|x, \psi)$ which encodes sentence features into a relation distribution, and a decoder $p(e_i|r, e_{-i}, \theta)$ which predicts an entity given the relation cluster and another entity. Both are modeled by softmax functions:

$$q(r|x, \psi) = \frac{\exp\left(\mathbf{w}_r^{\intercal} \mathbf{g}(x)\right)}{\sum_{r' \in \mathcal{R}_{\mathcal{X}}} \exp\left(\mathbf{w}_{r'}^{\intercal} \mathbf{g}(x)\right)}, \tag{4}$$

$$p(e_i | e_{-i}, r, \theta) = \frac{\exp\left(\phi(e_i, e_{-i}, r, \theta)\right)}{\sum_{e'_i \in \mathcal{V}} \exp\left(\phi(e'_i, e_{-i}, r, \theta)\right)}, \tag{5}$$

where $\psi = \{\mathbf{w}_r | r \in \mathcal{R}_{\mathcal{X}}\}$ and $\mathbf{g}(x)$ is a vector representation of sentence x, which can be high-dimensional one-hot feature encodings or low-dimensional sentence embeddings encoded by deep neural networks. $\phi(e_1, e_2, r, \theta)$ can be a general scoring function defined over triplets. We use the instantiation with the best performance shown by (Marcheggiani and Titov, 2016), which is a combination of bilinear model and selectional preference model:

$$\phi(e_1, e_2, r, \theta) = \mathbf{e}_1^{\intercal} \mathbf{C}_r \mathbf{e}_2 + [\mathbf{e}_1, \mathbf{e}_2]^{\intercal} \mathbf{r} \tag{6}$$

where $\theta = \{\mathbf{C}_r, \mathbf{r}, \mathbf{e}_i | r \in \mathcal{R}_{\mathcal{T}}, e_i \in \mathcal{V}\}$, $\mathbf{C}_r$ is a matrix, $\mathbf{r}$ is a vector for the relation r, $\mathbf{e}_1$ and $\mathbf{e}_2$ are the vectors for head and tail entities respectively, and $[\mathbf{e}_1, \mathbf{e}_2]$ is the concatenation of the vector representations of the two entities.

The DVAE model directly optimizes the variational lower bound by doing gradient ascent for θ and ψ jointly. Both encoder $q(r|x, \psi)$ and decoder $p(e_i|r, e_{-i}, \theta)$ are implemented as neural networks. Standard training techniques and tricks can be applied.

3.2 Knowledge Base Constraint

Our KB constraint framework can be summarized as a two-step procedure: KB constraints construction and regularization for the learning model. In the constraints construction step, a set of sentences is formed as a query to KB and retrieves a set of constraints back. Then in the regularization step, we apply the constraint to regularize posterior distributions of the relation extractor.

Conceptually, given a set of sentences $\mathcal{X}$, we want to bias the learning result: After the entities are linked to the KB, if KB inference indicates that some pairs should be in a relation based on a set of rules Υ, then the extractor should be constrained to output it. This constraint can be encoded into a feature function $Q(\mathcal{X})$ = "entity pairs in the same relation based on Υ" and put into the posterior regularization framework (Gillenwater et al., 2011). However, the computational complexity of the feature function is exponential since we need to traverse the KB to find Υ. We instead consider the must-link and cannot-link constraints (Basu et al., 2004), indicating respectively that a pair of sentences should be or should not be labeled as the same relation. For each pairwise constraint, the model assigns an associated cost of violating that constraint for the model regularization.

3.2.1 KB Constraints Construction

From the perspective of KB, a must-link constraint on sentences (x_1, x_2) exists if two pairs of entities $(p_1, p_2) = [(e_{1,1}, e_{1,2}), (e_{2,1}, e_{2,2})]$ are similar given the KB, where $(e_{i,1}, e_{i,2})$ is the entity pair

Euclidean L_2 distance	$d_{Euc}\left(q_1(r), q_2(r)\right) = \sqrt{\sum_r \lvert q_1(r) - q_2(r)\rvert^2}$
Kullback-Leibler (KL) divergence	$d_{KL}(q_1(r), q_2(r)) = \sum_r q_1(r) \log\left(\frac{q_1(r)}{q_2(r)}\right)$
Jensen-Shannon (JS) divergence	$d_{JS}(q_1(r), q_2(r)) = \frac{1}{2}\sum_r q_1(r) \log\left(\frac{2q_1(r)}{q_1(r)+q_2(r)}\right) + \frac{1}{2}\sum_r q_2(r) \log\left(\frac{2q_2(r)}{q_1(r)+q_2(r)}\right)$

Table 1: Cluster regularization with different distance or divergences.

belongs to sentence x_i. This motivates us to define a similarity score for a pair of entity pairs. Instead of modeling the common relation paths or logic rules, which is computationally infeasible, we compare them in the latent embedding space. In particular, we model the KB using the TransE (Bordes et al., 2013) model, where a relation is interpreted as a translation from the head entity to the tail entity, with a score function, $\mathbf{e}_1 + \mathbf{r} = \mathbf{e}_2$ for each gold triplet (e_1, r, e_2) in the KB. This operation is fast and the latent embeddings are expressive in many cases. Then we can reason the latent relation representation of a particular pair in vector space by $\mathbf{r}_i = \mathbf{e}_{i,2} - \mathbf{e}_{i,1}$, without the need for extra parameters. Here $\mathbf{r}_i$ is not necessarily a real relation between two entities in the KB but just reflects the geometric property. The penalty for violating a must-link constraint between a pair of sentences with a high KB score should be higher than those with low KB scores. This further inspires us to define a soft constraint penalty based on the similarity of latent KB relations.

Here, we use the adjusted cosine similarity (Sarwar et al., 2001) between two latent relations as a must-link confidence score

$$s^+(x_1, x_2) = [\cos(\mathbf{e}_{1,2} - \mathbf{e}_{1,1}, \mathbf{e}_{2,2} - \mathbf{e}_{2,1})]^+_{\gamma^+} \tag{7}$$

where $[x]^+_{\gamma^+} = x$ if $x > \gamma^+$ otherwise 0, $\gamma^+ \in [0, 1]$ is a threshold we defined to control the must-link scope, $e_{i,j}$ is named entity in x_i and $\mathbf{e}_{i,j}$ is its embedding. The similarity between $\mathbf{e}_{1,2} - \mathbf{e}_{1,1}$ and $\mathbf{e}_{2,2} - \mathbf{e}_{2,1}$ evaluates whether two sentences indicate similar relations according to the KB embedding.

We also define the cannot-link in a similar way, where two sentences cannot be in the same cluster with a confidence

$$s^-(x_1, x_2) = [\cos(\mathbf{e}_{1,2} - \mathbf{e}_{1,1}, \mathbf{e}_{2,2} - \mathbf{e}_{2,1})]^-_{\gamma^-} \tag{8}$$

where $[x]^-_{\gamma^-} = x$ if $x < -\gamma^-$ otherwise 0, and $\gamma^- \in [0, 1]$ is a threshold we defined to control the cannot-link scope. We simply set $\gamma^+ = \gamma^- = \gamma$.

3.2.2 Clustering Regularization

For each pair of sentences (x_1, x_2), the relation extractor will predict a clustering posterior $q_i(r|x_i, \psi)$, $i = 1, 2$, which can be computed based on Eq. (4). We regularize the clustering result on the probability distance between sentence pairs, using either Euclidean L_2 distance, Kullback-Leibler (KL) divergence, or Jensen-Shannon (JS) divergence. The computation of the distance or divergences can be found in Table 1.

Then the soft constraints introduced in §3.2.1 are applied on the corresponding distance to calculate the regularization terms:

$$D^+(x_1, x_2) = -d_*\left(q_1(r), q_2(r)\right) s^+(x_1, x_2), \tag{9}$$

$$D^-(x_1, x_2) = d_*\left(q_1(r), q_2(r)\right) |s^-(x_1, x_2)|, \tag{10}$$

for must and cannot links respectively, where d_* can be d_{Euc}, d_{KL}, or d_{JS}. Taking must-link constraint as an example, if the posterior distributions $q_1(r|x_1, \psi)$ and $q_2(r|x_2, \psi)$ are different from each other but KB suggests that these two sentences should be in the same cluster where $s^+(x_1, x_2)$ is large, then $d_*\left(q_1(r), q_2(r)\right)$ being large means there is a large cost when q_1 and q_2 being different. Then in the training phase, we want to reduce this cost given the constraint.

The constraints above are defined in a $|\mathcal{X}| \times |\mathcal{X}|$ space. It is almost impossible to enumerate all of the constraints. To make it trainable, we instead gather the constraints within a mini-batch. Since in different training epochs we randomly permute the training samples, it is possible to touch many pairs of sentences in practice.

3.3 Learning

The model parameters only exist in original autoencoder components (i.e., ψ and θ), which can be jointly optimized by maximizing the following objective function with L_2 regularization:

$$\begin{aligned}\mathcal{L}(\theta,\psi) = &\sum_{x\in X}\sum_{i=1}^{2}\sum_{r\in R_T} q(r|x,\psi)\log p(e_i|e_{-i},r,\theta)\\ &+\sum_{x\in X}\alpha H\left[q(r|x,\psi)\right]\\ &+\sum_{X_i\sim X}\sum_{(x_1,x_2)\in X_i}\beta D(x_1,x_2)\\ &+\lambda\|(\psi,\theta)\|_2,\end{aligned} \tag{11}$$

where α, β, γ, and λ are hyper-parameters to control the regularization strength. D can be D^+ or D^- depending on the cosine similarity between pairs. In practice, we apply annealing method over α in an exponential way:

$$\alpha_t = \alpha_0 \exp(-\eta t) \text{ and } \eta = \frac{\log(\alpha_0/\alpha_T)}{T},$$

where α_0 is the initial value, and α_T is the final value, t and T are the current and total training steps respectively. This method enables the extractor to explore more possibilities first and finally converge to a stable distribution.

It is difficult to directly compute the partition function in Eq. (5), as it requires to sum over $|\mathcal{V}|$. We use the same negative sampling method as (Marcheggiani and Titov, 2016) to substitute $\log p(e_i|e_{-i},r,\theta)$ in Eq. (11) with:

$$\begin{aligned}\log p(e_i|e_{-i},r,\theta) \approx &\log\sigma(\phi(e_i,e_{-i},r,\theta))\\ &+\sum_{e^{\text{neg}}\in\mathcal{N}}\log\sigma\left(-\phi(e^{\text{neg}},e_i,r,\theta)\right),\end{aligned}$$

where $\mathcal{N}$ is the set of randomly sampled entities in $\mathcal{V}$ and σ is the sigmoid function.

4 Experiments

In this section, we show the experimental results.

4.1 Dataset and Preprocessing

We evaluate our model in the context of unsupervised relation discovery and compare to the baseline model, DVAE (Marcheggiani and Titov, 2016) which is the current state-of-the-art of relation discovery. Distant supervision assumes that the relations should be aligned between the KB and the training text corpus, which is not available in our setting.

We tested our model on three different subsets of New York Times corpus (NYT) (Sandhaus and Evan, 2008).

	Data	**NYT122**	**NYT71**	**NYT27**
Text	# sentences	67,123	14,210	87,144
	# facts	9,207	2,274	8,559
	# entity pairs	20,939	3,539	36,714
	# entities	5,865	2,489	4,803
	# relations	122	71	27
KB	# triplets	401,490	456,146	439,507
	# entity pairs	331,008	373,875	354,960
	# entities	14,907	14,933	14,911
	# relations	705	1,009	1,031

Table 2: Statistics of datasets. # facts in the text corpus is the number of sentences with relation labels.

- The first one is widely used in unsupervised settings, which was developed by Yao et al. (2011) and has also been used by Marcheggiani and Titov (2016). This dataset contains articles 2000 to 2007, with named entities annotated and features processed (POS tagging, NER, and syntactic parsing). We use this dataset to compare with previous work directly (Marcheggiani and Titov, 2016).

- The second and third ones are usually applied by supervised models. So when they generated the data, they tended to focus on relations with more supporting sentences. The second one was developed by Zeng et al. (2017). The data is built by aligning Wikidata (Vrandečić, 2012) relations with NYT corpus, as a result of 99 possible relations. It is built to contain more updated facts and richer structures of relations, e.g., a larger number of relation/relation paths. We use this dataset to amplify the effects coming from relation paths in KB, as the data was used to train a path-based relation extraction model.

- The third one was developed by Riedel et al. (2010) and has also been used by Lin et al. (2016). This dataset was generated by aligning Freebase (Bollacker et al., 2008) relations with NYT in 2005-2007, and with 52 possible relations. We use this data to test the clustering result with a narrow relation domain.

We align these datasets against FB15K, which is a randomly sampled subset of Freebase developed by Bordes et al. (2013). For each of the datasets above, we hold out the triplets in FB15K that contains relations in corresponding text data, so that we ensure that KB cannot give any direct supervision on any relation labels. We then discard named

Model	**Metrics**							
	Prediction based on encoder				Prediction based on decoder			
	F1		NMI		F1		NMI	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
DVAE	0.417	0.011	0.339	0.009	0.419	0.011	0.337	0.014
RegDVAE (Euclidean at encoder)	**0.469**	0.014	**0.430**	0.020	**0.448**	0.020	**0.384**	0.020
RegDVAE (KL at encoder)	0.375	0.009	0.359	0.014	0.380	0.011	0.355	0.014
RegDVAE (JS at encoder)	0.435	0.038	0.370	0.042	0.409	0.012	0.336	0.005
RegDVAE (Euclidean at decoder)	0.416	0.019	0.329	0.017	0.350	0.012	0.201	0.054

Table 3: Comparison results on NYT122 with different prediction and regularization strategies (using encoder or decoder).

entities in text corpus if they are not shown in KB, so that we can directly test the influence of our KB constraint model. Finally, we only keep a single label for each sentence, and e_1, e_2 follow the occurrence order in the sentence. The resulting datasets contain 122, 71, and 27 relation labels respectively, so we name them as NYT122, NYT71, and NYT27. The statistics of the three datasets are shown in Table 2. For NYT71 and NYT27, we perform the same feature extraction as NYT122 shown in (Marcheggiani and Titov, 2016).

4.2 Implementation Details

All the model parameters are initialized randomly. The number of negative samples is set to 5, mini-batch size is set to 100 with 80 epochs. We optimize all the models using AdaGrad (Duchi et al., 2011) with initial learning rate at 0.5. For NYT122, we induce 40 relations clusters, with $\alpha_0 = 4$, $\alpha_T = 10^{-5}$, $\beta = 0.6$, and $\gamma = 0.9$. For NYT71, we induce 30 relations clusters, with $\alpha_0 = 2$, $\alpha_T = 10^{-4}$, $\beta = 0.8$, and $\gamma = 0.95$. For NYT27, we induce 20 relations clusters, with $\alpha_0 = 2$, $\alpha_T = 10^{-4}$, $\beta = 0.8$, and $\gamma = 0.3$. We train TransE as our KB embedding model with 50 dimensions and 1,000 epochs.

We report the average and standard deviation based on five different runs. We randomly split the data into validation:test=4:6. All the model selections were based on validation sets, and final evaluation results will be only based on test sets.

4.3 Evaluation and Discussion

As the scoring function, we use the B^3F_1 (Bagga and Baldwin, 1998) which has also been used by our baseline (Marcheggiani and Titov, 2016), and Normalized Mutual Information (NMI) (Strehl and Ghosh, 2002) metrics. Both are standard measures for evaluating clustering tasks.

Regularization and Prediction Strategies. We first report our results on NYT122 using different regularization and prediction settings, as this dataset was used by our baseline model DVAE.

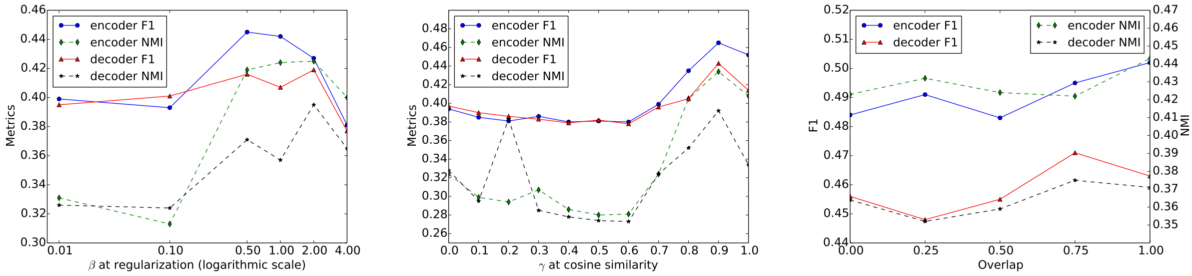
Note that both encoder and decoder components can make relation predictions. In fact, the way of using encoder $q(r|x, \psi)$ for each sentence is straightforward. Then based on the encoder, we predict relation on the basis of single occurrence of entity pair. When using the decoder, we need to re-normalize $p(e_i|r, e_{-i}, \theta)$ as $p(r|e_1, e_2, \theta)$ to make predictions. Based on the decoder, we make predictions for each unique entity pair. As a consequence, our constraints can be imposed on both encoder and decoder. The way of computing decoder probability distribution is the same as making predictions. So in this experiment, we report both results.

The results are shown in Table 3. From the table, we can see that regularization with Euclidean distance performs the best compared to KL and JS. Moreover, the regularization over encoder is better than the regularization over decoder. This may be because the way that we put constraints only over sampled sentences in a batch may hurt the regularization of decoder, since sampled unique pairs may be less than sample sentences. If we look at results comparing original DVAE prediction based on the encoder and the decoder, both result in similar F1 and NMI numbers. Thus, we can only conclude that currently in the way we do sampling, constraining over encoder is a better choice.

Comparison on Different Datasets. We also compare our algorithm on the three datasets with different baseline settings. In order to evaluate our model rigorously, besides the original DVAE model, we compare two additional augmented baseline models with the same hyper-parameter setting: DVAE with TransE embeddings appended to encoder input features (DVAE+E) and DVAE

Model	**NYT122**				**NYT71**				**NYT27**			
	F1		NMI		F1		NMI		F1		NMI	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Majority	0.355	-	0	-	0.121	-	0	-	0.549	-	0	-
DVAE	0.417	0.011	0.339	0.009	0.325	0.011	0.375	0.023	0.433	0.018	0.384	0.021
DVAE+E	0.385	0.021	0.341	0.043	0.339	0.021	0.418	0.022	0.396	0.034	0.381	0.039
DVAE+D	0.452	0.033	0.438	0.022	0.352	0.038	0.339	0.009	0.499	0.040	0.469	0.027
RegDVAE	0.469	0.014	0.430	0.020	0.377	0.020	0.466	0.036	0.587	0.005	0.451	0.005
RegDVAE+D	**0.499**	0.022	**0.497**	0.013	**0.432**	0.028	**0.589**	0.071	**0.665**	0.022	**0.562**	0.038

Table 4: Comparison of prediction results based on encoder using NYT122, NYT71, and NYT27 datasets with different KB regularization strategies.



(a) Parameter sensitivity analysis at β.

(b) Parameter sensitivity analysis at γ.

(c) Relation overlap ratios.

Figure 2: Comparison results on NYT122 with different parameters and relation overlaps. The predictions are based on either the encoder or the decoder.

with decoder entity vectors replaced by pre-trained KB embeddings (DVAE+D). For our method, we report RegDVAE with the best setting where we use Euclidean distance based constraints to regularize the encoder. Moreover, we report a setting with fixed embeddings in the decoder as the ones obtained from TransE (RegDVAE+D). This also makes sense since even though the TransE embeddings are not trained with the observation of the same relations as the text corpus, the embeddings already contain much semantic information about entities. Then by fixing the embeddings of entities in the decoder, we can significantly reduce the number of parameters that need to be trained. The results are shown in Table 4. As we can see that, RegDVAE+D can outperform the original DVAE by 8∼23 points on F1. DVAE+D is also good but may fail when there are a lot of out-of-sample entities in the training corpus.

Hyper-parameter Sensitivity. We have three hyper-parameters in our algorithm: α_0 for the regularization of encoder entropy, β for the regularization with our constraints, and γ for the threshold of KB based cosine similarities. Here, we test β and γ, since the sensitivity result of α_0 is the same as the original DVAE work (Marcheggiani and Titov, 2016). The sensitivity of β is shown in Figure 2(a). The results are good in a wide range from $\beta = 0.5$ to $\beta = 2$. The sensitivity of γ is shown in Figure 2(b). It reveals some interesting patterns. At the beginning when γ is small, it hurts the performance. After γ getting greater than 0.7, it improves the performance, which means that only very similar relations indicated by KB embeddings are useful relations as constraints. In addition, $\gamma = 1$ (meaning only finding identical relations) is worse than $\gamma = 0.9$, which means we indeed find some relations in our KB so that different triplets will be constrained.

KB Relation Overlap. Although we assume that there is no overlapped relation between the KB and the training text corpus, in practice, we may find a lot of applications that the relations are partially observed in KB. Thus, we also test a setting when the KB has different proportions of overlapped relations with training text corpus. In this case, we train different KB embeddings for different percentages of overlapped relations, and then apply the embeddings into the constraints. The results are shown in Figure 2(c). As we can see, in general, more overlapped relations will result in better performance. The best number can be better than the number without overlapped relation by about two points. This again verifies that the KB embedding is very robust and represent the semantic meanings of entities even with part of the

Contextual Sentence	Cluster	Similarity
. . . **Spain** will become the third country in **Europe**. . .	12	0.926
Portugal, with all that talent, goes home to **Europe**. . .	12	
Brazil, **Latin America** 's largest economy . . .	12	0.916
. . . **Argentina** was perhaps the most expensive country in **Latin America** for tourists.. . .	12	

Table 5: Examples for relation: /location/contained_by.

relations observed (Bordes et al., 2013).

Case Study. We also show some examples of entity pair similarities in Table 5. From the Table we can see that our target relation cluster is */location/contained_by*. In the first example, the similarity between entity pairs (`Spain, Europe`) and (`Portugal, Europe`) are high, which indicates the same cluster of pairs of sentences. The same constraint is applied in the second example, although there's no direct connection between (`Brazil, Latin America`), (`Argentina, Latin America`).

5 Related Work

Supervised and Distantly Supervised Relation Extraction. Traditional supervised relation extraction focuses on a limited number of relations (Roth and Yih, 2002; Kambhatla, 2004; Chan and Roth, 2010). Distant supervision uses KBs to obtain a lot of automatically annotated data (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Xu et al., 2013a; Zhang et al.; Zeng et al., 2015; Lin et al., 2016; Zeng et al., 2017). There are two important assumptions behind these models, namely multi-instance learning (Riedel et al., 2010) and multi-instance multi-label learning (Hoffmann et al., 2011; Surdeanu et al., 2012). Our setting is similar to multi-instance learning but we assume there is no overlapped relation between KB and training text corpus. Universal schema (Riedel et al., 2013; Verga et al., 2016; Toutanova et al., 2015; McCallum et al., 2017) can also exploit KB to help extract relations. It needs a lot of entity pairs in text to co-occur with KB triplets, which is under the same setting with distant supervision. Those surface patterns are pre-extracted and shown in the training phase, which makes it also a weakly supervised learning method.

Unsupervised Relation Extraction. Open Domain Information Extraction (Open-IE) assumes that every relation expression can represent a unique relation (Etzioni et al., 2004; Banko et al., 2007; Fader et al., 2011; Mausam et al., 2012; Xu et al., 2013b; Angeli et al., 2015). On the other hand, relation clustering approaches group all the related relation expressions to represent a relation (Lin and Pantel, 2001; Mohamed et al., 2011; Takamatsu et al., 2011; Yao et al., 2011, 2012; Nakashole et al., 2012a,b; Marcheggiani and Titov, 2016). Our setting is based on (Marcheggiani and Titov, 2016) but we also introduce KB as a different kind of weak and indirect supervision.

Knowledge Base Representation. Embedding based knowledge base representation learning methods (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Trouillon et al., 2016) represent entities and relations as vectors, denoted as $\mathbf{e}$ and $\mathbf{C}_r$ respectively such that for a distances function f, the value $f(\mathbf{e}_1, \mathbf{C}_r, \mathbf{e}_2)$ is maximized for all (e_1, r, e_2) facts. Among all these methods, TransE model has a favorable property that the translation operation can be easily recovered by entity vectors ($\mathbf{r}_{1,2} = \mathbf{e}_1 - \mathbf{e}_2$). With its simplicity and high performance, TransE is enough for demonstration. Though our method is not restricted to the representation form of KB, we leave it for future evaluation.

Constraints can be made more explainable by paths finding. For instance, the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010; Lao et al., 2011) uses random walk to perform multi-hop reasoning based on logic rules. Later on, reinforcement Learning (Toutanova et al., 2015; Xiong et al., 2017; Das et al., 2017; Chen et al., 2018) is used to search for paths more effectively. Though heuristics are used to further reduce the number of mined relations, it is still very costly to find the paths for KB with hundreds of relations, if not impossible.

Constraint Modeling. Originated from semi-supervised learning (Chapelle et al., 2006), must-link and cannot-link modeling has been well studied in machine learning community (Wagstaff et al., 2001; Basu et al., 2004, 2008). Such constraints were usually generated based on the ground truth labels of data. For document clustering, word constraints constructed based on WordNet similarities have been applied (Song et al., 2013) and entity constraints based on entity types in an external KB have been used (Wang et al., a, 2016), both being considered as a kind of indirect supervision based on side information. For triplet relation clustering, relation surface similarity and entity type constraints have been explored (Wang et al., b). However the above constraints are applied to a particular form of models, co-clustering models. Compared to existing approaches, our constraints are constructed based on more recently developed KB embeddings, which is more flexible and easy to incorporate into different models.

In natural language processing community, constraints based on background knowledge are also well studied. For example, constrained conditional models (CCM) (Chang et al., 2012) provides a very flexible framework to decouple learning and inference, where in the inference step, background knowledge can be incorporated as an ILP (integer linear programming) problem. Posterior regularization (PR) (Ganchev et al., 2010) generalizes this idea so that it uses a joint learning and inference framework to incorporate the background knowledge. Both CCM and PR have many applications including the application to relation extraction (Chan and Roth, 2010; Chen et al., 2011). Compared to these existing approaches, our constraints are derived from the general-purpose KB, which is quite different from their way of manually crafting some background knowledge as declarative rules.

It is very interesting that we are similar to the PR framework. Since we use a DVAE framework as the base algorithm, there is no traditional E-step and M-step in the variational inference. Instead, only q and p probabilities parameterized by neural networks are updated. In our framework, we can add constraints to either q or p probabilities (applying to p needs modification of normalization). It is the same that we draw a biased learning process when estimating the posteriors as PR does.

6 Conclusion

In this paper, we propose a new relation discovery setting where there is no overlapped relations between the training text corpus and the KB. We propose a new learning framework of KB regularization which uses must-link and cannot-link constraints derived based on similarities in the KB embedding space. Our method improves the results over all baseline models without harming the scalability. We believe this framework is as flexible as other constraint models to be applied to many applications when we think the semantics of entities and relations provided by the KB is useful.

Acknowledgments

This paper was supported by the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong. We thank Intel Corporation for supporting our deep learning related research. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*, pages 344–354.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *LREC*, pages 563–566.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, pages 2670–2676.

Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *KDD*, pages 59–68.

Sugato Basu, Ian Davidson, and Kiri Wagstaff. 2008. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *COLING*, pages 152–160.

Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431.

O. Chapelle, B. Schölkopf, and A. Zien, editors. 2006. *Semi-Supervised Learning*. MIT Press.

Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. 2011. In-domain relation discovery with meta-constraints via posterior regularization. In *ACL-HLT*, pages 530–540.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *NAACL-HLT*, pages 1823–1832.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *CoRR*, abs/1711.05851.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Oren Etzioni, Michael Cafarella, and Doug Downey. 2004. Webscale information extraction in knowitall (preliminary results). In *WWW*, pages 100–110.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2011. Posterior sparsity in unsupervised dependency parsing. *Journal of Machine Learning Research*, 12:455–490.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, pages 541–550.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *ACL - Poster and Demonstration*.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random walk inference and learning in A large scale knowledge base. In *EMNLP*, pages 529–539.

Dekang Lin and Patrick Pantel. 2001. DIRT – discovery of inference rules from text. In *KDD*, pages 323–328.

Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pages 705–714.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, pages 2124–2133.

Xitong Liu, Fei Chen, Hui Fang, and Min Wang. 2014. Exploiting entity relationship for query expansion in enterprise search. *Inf. Retr.*, 17(3):265–294.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistic*, 4:231–244.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534.

Andrew McCallum, Arvind Neelakantan, and Patrick Verga. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *EACL*, pages 613–622.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*, pages 1003–1011.

Thahir Mohamed, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2011. Discovering relations between noun categories. In *EMNLP*, pages 1447–1455.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012a. Discovering and exploring relations on the web. *PVLDB*, 5(12):1982–1985.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012b. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP*, pages 1135–1145.

Deepak Ravichandran and Eduard H. Hovy. 2002. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML PKDD*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*, pages 74–84.

Dan Roth and Wen-tau Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *COLING*.

Sandhaus and Evan. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295.

Yangqiu Song, Shimei Pan, Shixia Liu, Furu Wei, Michelle X. Zhou, and Weihong Qian. 2013. Constrained text coclustering with supervised and unsupervised constraints. *IEEE Trans. Knowl. Data Eng.*, 25(6):1227–1239.

Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles — A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*, pages 455–465.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2011. Probabilistic matrix factorization leveraging contexts for unsupervised relation extraction. In *PAKDD*, pages 87–99.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *NAACL-HLT*, pages 886–896.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *WWW*, pages 1063–1064.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584.

Chenguang Wang, Yangqiu Song, Ahmed El-Kishky, Dan Roth, Ming Zhang, and Jiawei Han. a. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224.

Chenguang Wang, Yangqiu Song, Dan Roth, Chi Wang, Jiawei Han, Heng Ji, and Ming Zhang. b. Constrained information-theoretic tripartite graph clustering to identify semantically similar relations. In *IJCAI*, pages 3882–3889.

Chenguang Wang, Yangqiu Song, Dan Roth, Ming Zhang, and Jiawei Han. 2016. World knowledge as indirect supervision for document clustering. *ACM Transactions on Knowledge Discovery from Data*, 11(2):13:1–13:36.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013a. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL*, pages 665–670.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013b. Open information extraction with tree kernels. In *NAACL-HLT*, pages 868–877.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *EMNLP*, pages 1456–1466.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *ACL*, pages 712–720.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *EMNLP*, pages 1768–1777.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. Towards accurate distant supervision for relational facts extraction. In *ACL (2)*, pages 810–815.

Corpora Generation for Grammatical Error Correction

Jared Lichtarge*, **Chris Alberti***, **Shankar Kumar***, **Noam Shazeer***, **Niki Parmar***, **Simon Tong***
Google Research
`{lichtarge,chrisalberti,shankarkumar,noam,nikip,simon}@google.com`

Abstract

Grammatical Error Correction (GEC) has been recently modeled using the sequence-to-sequence framework. However, unlike sequence transduction problems such as machine translation, GEC suffers from the lack of plentiful parallel data. We describe two approaches for generating large parallel datasets for GEC using publicly available Wikipedia data. The first method extracts source-target pairs from Wikipedia edit histories with minimal filtration heuristics, while the second method introduces noise into Wikipedia sentences via round-trip translation through bridge languages. Both strategies yield similar sized parallel corpora containing around 4B tokens. We employ an iterative decoding strategy that is tailored to the loosely supervised nature of our constructed corpora. We demonstrate that neural GEC models trained using either type of corpora give similar performance. Fine-tuning these models on the Lang-8 corpus and ensembling allows us to surpass the state of the art on both the CoNLL-2014 benchmark and the JFLEG task. We provide systematic analysis that compares the two approaches to data generation and highlights the effectiveness of ensembling.

1 Introduction

Much progress in the Grammatical Error Correction (GEC) task can be credited to approaching the problem as a translation task (Brockett et al., 2006) from an ungrammatical source language to a grammatical target language. This has enabled Neural Machine Translation (NMT) sequence-to-sequence (S2S) models and techniques to be applied to the GEC task (Tao et al., 2018b; Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018). However, the efficacy of NMT techniques is degraded for low-resource tasks (Koehn and Knowles, 2017). This poses difficulties for S2S approaches to GEC, as Lang-8, the largest publicly available parallel corpus, contains only ∼25M words (Mizumoto et al., 2011). Motivated by this data scarcity, we present two contrasting approaches to generating parallel data for GEC that make use of publicly available English language Wikipedia revision histories[1][2].

Our first strategy is to mine real-world errors. We attempt to accumulate source–target pairs from grammatical errors and their human-curated corrections gleaned from the Wikipedia revision histories. Unlike previous work (Grundkiewicz and Junczys-Dowmunt, 2014), we apply minimal filtering so as to generate a large and noisy corpus of ∼4B tokens (Table 1). As a consequence of such permissive filtering, the generated corpus contains a large number of real grammatical corrections, but also noise from a variety of sources, including edits with drastic semantic changes, imperfect corrections, ignored errors, and Wikipedia spam.

Our second strategy is to synthesize data by corrupting clean sentences. We extract target sentences from Wikipedia, and generate corre-

---

*Equal contribution. Listing order is random. Jared conducted systematic experiments to determine useful variants of the Wikipedia revisions corpus, pre-training and fine-tuning strategies, and iterative decoding. Chris implemented the ensemble and provided background knowledge and resources related to GEC. Shankar ran training and decoding experiments using round-trip translated data. Jared, Chris and Shankar wrote the paper. Noam identified Wikipedia revisions as a source of training data. Noam developed the heuristics for using the full Wikipedia revisions at scale and conducted initial experiments to train Transformer models for GEC. Noam and Niki provided guidance on training Transformer models using the *Tensor2Tensor* toolkit. Simon proposed using round-trip translations as a source for training data, and corrupting them with common errors extracted from Wikipedia revisions. Simon generated such data for this paper.

[1] `https://dumps.wikimedia.org/enwiki/latest/`

[2] Last accessed: December 15, 2017

Proceedings of NAACL-HLT 2019, pages 3291–3301
Minneapolis, Minnesota, June 2 - June 7, 2019.

sponding source sentences by translating the target into another language and back. This round-trip translation introduces relatively clean errors, so the generated corpus is much less noisy than the human-derived Wikipedia corpus. However, these synthetic corruptions, unlike human errors, are limited to the domain of errors that the translation models are prone to making. Both approaches benefit from the broad scope of topics in Wikipedia.

Corpus	# sentences	# words
Lang-8	1.9M	25.0M
WikEd	12M	292 M
Wikipedia Revisions	170M	4.1B
Round-Trip Translation	176M	4.1B

Table 1: Statistics computed over extant training sets for GEC (top) and corpora generated from Wikipedia in this work (bottom).

We train the Transformer sequence-to-sequence model (Vaswani et al., 2017) on data generated from the two schemes. Fine-tuning the models on the Lang-8 corpus gives us additional improvements which allow a single model to surpass the state-of-art on both the CoNLL-2014 and the JFLEG tasks. Finally, we explore how to combine the two data sources by comparing a single model trained on all the data to an ensemble of models.

2 Data Generation from Wikipedia Revision Histories

Wikipedia provides a dump of the revision histories of all Wikipedia pages. For each Wikipedia page, the dump contains chronological snapshots of the entire content of the page before and after every submitted edit; thus two consecutive snapshots characterize a single revision to the page. Because a small number of popular pages see disproportionate traffic, some pages grow very large. As we are interested in the edits between snapshots, and not identical content that is typically in higher proportion in the revision histories for the largest pages, we discard pages larger than 64Mb. To prevent remaining large pages from skewing the dataset towards their topics with their many revisions, we downsample consecutive revisions from individual pages, selecting only $\log_{1.5}(n)$ pairs for a page with a total of n revisions. This reduces the total amount of data 20-fold. Each remaining pair of consecutive snapshots forms a source–target pair. The process for extracting examples from a page's revision history is illustrated in Figure 1.

From the XML of each page in a source–target pair, we extract and align the text, removing non-text elements. We then probabilistically cut the aligned text, skipping over non-aligned sequences. Two cuts bound an example pair, for which the source sequence is provided by the older snapshot, and the target sequence by the newer snapshot.

Following extraction of the examples, we do a small amount of corruption and filtration in order to train a model proficient at both spelling and grammar correction. We probabilistically introduce spelling errors in the source sequences at a rate of 0.003 per character, randomly selecting deletion, insertion, replacement, or transposition of adjacent characters for each introduced error.

We throw out examples exceeding a maximum length of 256 word-pieces. The majority of examples extracted by this process have identical source and target. Since this is not ideal for a GEC parallel corpus, we downsample identity examples by 99% to achieve 3.8% identical examples in the final dataset. The data generation scripts we use have been opensourced[3].

In Figure 2, we show examples of extracted source–target pairs. While some of the edits are grammatical error corrections, the vast majority are not.

3 Data Generation from Round-trip Translations

As an alternative approach to extracting the edits from Wikipedia revisions, we extract sentences from the identity examples that were discarded during edit extraction, and generate a separate parallel corpus by introducing noise into those sentences using round-trip translation via a bridge language. Therefore, the original sentence from Wikipedia is the target sentence and output of the round-trip translation is the corresponding source sentence. The round trip translations introduce noise according to both the weaknesses of the translation models and the various inherent ambiguities of translation. We create a corrupted dataset using each bridge language. We use French (Fr), German (De), Japanese (Ja) and Russian (Ru) as bridge languages because they are high-resource languages and relatively dissim-

---

[3] `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/data_generators/wiki_revision.py`

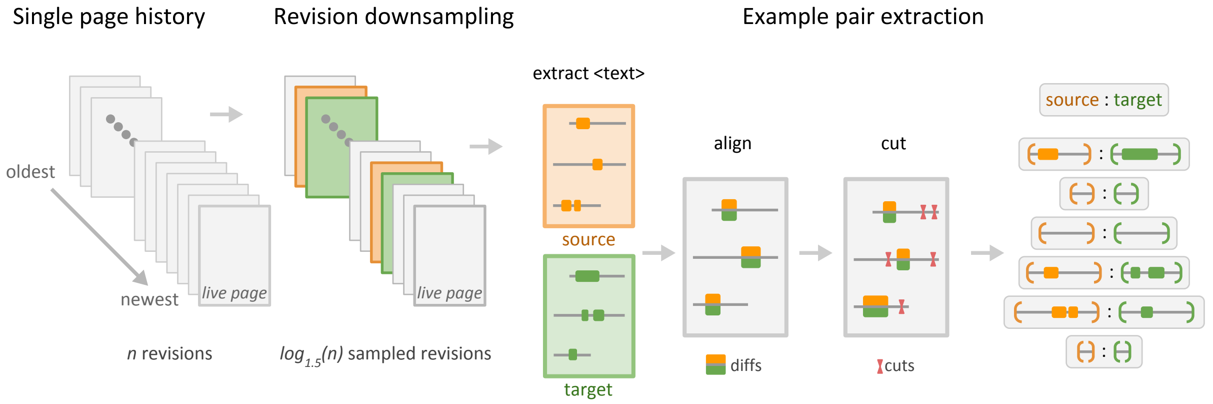


Figure 1: Process for extracting source–target pairs from revision history of a Wikipedia page. See Figure 2 for actual examples.


Examples drawn from Revisions

Artillery in 1941 and was medically dis-charged : Artillery in 1941 he was later medically discharged with

Special terms have been coined to denote many imfortant technical concepts in the game of Go. Such technical : Players of the game of Go often use jargon terms to describe situations on the board and surrounding the game. Such technical

The County of Fitzroy is a county in Queensland, : The County of Fitzroy is a county (a cadastral division) in Queensland,

Examples drawn from Round-Trip Translations

Aerolineas held a strong company through rthe 90's and they even aded Sydney as a goal for a little while. : Aerolineas kept on being a strong company thru the 90's and they even added Sydney as a destination for a little while.

What we no wcall "disco balls" was first used in nightclubs in the 1920s. : What we now call "disco balls" were first used in nightclubs in the 1920 's.

At the same time, she became a jounalist for news, such as "NHK News 7" and "Shutoken News 845". : At the same time, she became a newscaster for some news shows , such as "NHK News 7" and "Shutoken News 845".

Figure 2: Example source–target pairs from each corpus.

ilar from each other. Thus, we compute a total of four corrupted datasets. The translations are obtained using a competitive machine translation system (Wu et al., 2016).

These round trip translated sentence-pairs contained only a small fraction of identity translations compared to those that are present in real-world GEC corpora. To address this deficiency, we augment this corpus with 2.5% identity translations. Analogous to Section 2, we want the models to learn both spelling and grammar correction. Thus, we randomly corrupt single characters via insertion, deletion, and transposition, each with a probability of $0.005/3$. Round-trip translations do not contain some types of word and phrase errors (e.g., your/you're, should of/should have) and so we additionally corrupt the translated text by stochastically introducing common errors identified in Wikipedia. We first examine the Wikipedia revision histories to extract edits of up to three words whose source and target phrases are close in edit distance, and which do not contain numbers or capitalization. For each of the remaining edits (original, revised), we compute the probability that the user typed *original* when they intended to type *revised*:

$$P(\text{original}|\text{revised}) = \frac{C(\text{original}, \text{revised})}{C(\text{revised})},$$

where $C(x)$ refers to the counts of x in the corpus. We then probabilistically apply these rules to corrupt the translated text.

This process produces a parallel corpus identical in size to the Wikipedia Revision corpus, though with vastly different characteristics. Because the target sentences are Wikipedia sentences that were left unchanged for at least one Wikipedia revision, they are less likely to contain poor grammar, misspellings, or spam than the target sequences of the revisions data.

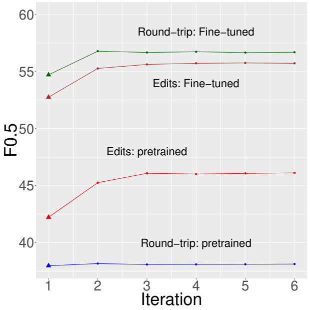


Figure 3: $F_{0.5}$ with iterative decoding on the CoNLL dev set. Triangles indicate performance with single-shot decoding.

Also, the errors introduced by round-trip translation are relatively clean, but they represent only a subset of the domain of real-world errors. In contrast, the Wikipedia data likely has good coverage of the domain of real-world grammatical errors, but is polluted by significant noise. Examples from both corpora are shown in Figure 2. Examples of round-trip translations for each bridge language are shown in Table 2.

4 Iterative Decoding

Many sentences that require grammatical correction contain multiple errors. As a result, it can be difficult to correct all errors in a single decoding pass. This is specifically a problem when using models trained on noisy parallel data such as Lang-8 where the target sentences still contain grammatical errors. Following other work on the GEC task (Dahlmeier and Ng, 2012a; Tao et al., 2018a), we employ an iterative decoding algorithm that allows the model to make multiple incremental corrections. This allows the model multiple chances to suggest individually high-confidence changes, accruing incremental improvements until it cannot find any more edits to make.

Our iterative decoding algorithm is presented in Algorithm 1. Given the source sentence S and a hypothesis H, Cost(H) refers to the negative log probability $-logP(H|S)$ using the sequence-to-sequence model. In each iteration, the algorithm performs a conventional beam search but is only allowed to output a rewrite (non-identity translation) if it has high confidence i.e., its cost is less than the cost of the identity translation times a prespecified threshold. Using iterative decoding allows a stricter threshold value than what is optimal for single-shot decoding, as a change ignored for being low confidence in one decoding iteration may be selected in the next.

Using incremental edits produces a significant improvement in performance over single-shot decoding for models trained on the Wikipedia revision data, a highly noisy corpus, while models trained on the relatively clean round-trip translation data see no improvment. All models finetuned on Lang-8 see improvement with iterative decoding (Figure 3, Table 3).

Algorithm 1: Iterative Decoding

```
Data: $I$, beam, threshold, MAXITER
Result: $\hat{T}$
for $i \in \{1, 2, ..., MAXITER\}$ do
    Nbestlist = Decode($I$, beam)
    $C_{\text{Identity}} = +\infty$
    $C_{\text{Non-Identity}} = +\infty$
    $H_{\text{Non-Identity}} = NULL$
    for $H \in Nbestlist$ do
        if $H = I$ then
            $C_{\text{Identity}} = \text{Cost}(H)$;
        else if $Cost(H) < C_{Non\text{-}Identity}$ then
            $C_{\text{Non-Identity}} = \text{Cost}(H)$
            $H_{\text{Non-Identity}} = H$
    end
        ▷ Rewrite if non-identity cost < identity cost
    if $C_{Non\text{-}Identity}/C_{Identity} < threshold$ then
        $\hat{T} = H_{\text{Non-Identity}}$        ▷ Output rewrite.
    else
        $\hat{T} = I$        ▷ Output identity.
    end
    $I = \hat{T}$        ▷ Input for next iteration.
end
```

In Table 4, we show an example of iterative decoding in action. The model continues to refine the input until it reaches a sentence that does not require any edits. We generally see fewer edits being applied as the model gets closer to the final result.

5 Model

In this work, we use the *Transformer* sequence-to-sequence model (Vaswani et al., 2017), using the *Tensor2Tensor* opensource implementation.[4] We use 6 layers for both the encoder and the decoder, 8 attention heads, embedding size $d_{\text{model}} = 1024$, a position-wise feed forward network at every layer of inner size $d_{ff} = 4096$, and Adafactor as optimizer with inverse squared root decay (Shazeer and Stern, 2018)[5]. The word tokens are split into subwords using a variant of the byte-pair encoding technique (Sennrich et al., 2016b), described in Schuster and Nakajima (2012).

We train the *Transformer* model for 5 epochs

---

[4] `https://github.com/tensorflow/tensor2tensor`

[5] We used the "transformer_clean_big_tpu" setting.

Original	“The Adventures of Patchhead“ makes its second and final appearance.
Bridge Language	
French	“The Adventures of Patchhead “ makes **his secnod** and final appearance.
German	“The Adventures of Patchhead” makes its second and **last** appearance.
Russian	“The Adventures of Patchhead” makes its second and **last apparance**.
Japanese	**“Patchhead Adventure” is the** final appearance **of the second time**.
Original	He is not so tolerant of the shortcomings of those outside his family.
Bridge Language	
French	He is not so tolerant of the **weaknesses** of those outside his family.
German	He is not so tolerant **to** the **defects** of **the** outside **of** his family.
Russian	He is not so tolerant of the shortcomings of those outside his family**,**.
Japanese	He is not so tolerant of the shortcomings of those outside his family.

Table 2: Example sentences generated via round-trip translation with introduced spelling errors.

Source	Decoding	CoNLL-2014			JFLEG
		Prec.	Rec.	$F_{0.5}$	GLEU$^+$
Revision	single-shot	60.4	19.2	42.2	54.5
	iterative	58.3	25.1	46.1	56.6
+finetune	single-shot	67.7	28.1	52.8	57.9
	iterative	64.5	36.2	55.8	62.0
RTT	single-shot	47.1	21.4	38.0	52.5
	iterative	47.1	21.4	38.0	52.5
+finetune	single-shot	66.7	31.8	54.7	59.0
	iterative	64.4	38.4	56.7	62.1

Table 3: Comparing iterative decoding to single-shot decoding for two models, trained on all Wikipedia revisions data and on all round-trip translation (RTT) data.

Original	this is nto the pizzza that i ordering
1st	this is **not** the **pizza** that **I** ordering
2nd	**T**his is not the pizza that I ordering
3nd	This is not the pizza that I **ordered**
4th	This is not the pizza that I ordered**.**
Final	This is not the pizza that I ordered.

Table 4: Iterative decoding on a sample sentence.

with a batch size of approximately 64,000 word pieces. While training on the Wikipedia corpora, we set the learning rate to 0.01 for the first 10,000 steps, then decrease it proportionally to the inverse square root of the number of steps after that.

We then finetune our models on Lang-8 for 50 epochs and use a constant learning rate of 3×10^{-5}. We stop the fine-tuning before the models start to overfit on a development set drawn from Lang-8.

6 Experiments

6.1 Evaluation

We report results on the CoNLL-2014 test set (Ng et al., 2014) and the JFLEG test set (Napoles et al., 2017; Heilman et al., 2014). Our initial experiments with iterative decoding showed that increasing beam sizes beyond 4 did not yield improvements in performance. Thus, we report all results using a beam size of 4. Our ensemble models are obtained by decoding with 4 identical *Transformers* trained and finetuned separately. Ensembles of neural translation systems are typically constructed by computing the logits from each individual system and combining them using either an arithmetic average (Sutskever et al., 2014) or a geometric average (Cromieres et al., 2016). Similar to Cromieres et al. (2016), we find that a geometric average outperforms an arithmetic average. Hence, we report results using only this scheme.

Following (Grundkiewicz and Junczys-Dowmunt, 2018; Junczys-Dowmunt et al., 2018), we preprocess JFLEG development and test sets with a spell-checking component but do not apply spelling correction to CoNLL sets. For CoNLL sets, we pick the best iterative decoding threshold and number of iterations on a subset of the CoNLL-2014 training set, sampled to have the same ratio of modified to unmodified sentences as the CoNLL-2014 dev set. For JFLEG, we pick the best decoding threshold on the JFLEG dev set.We report performance of our models by measuring $F_{0.5}$ with the M^2 scorer (Dahlmeier and Ng, 2012b) on the CoNLL-2014 dev and test sets, and the GLEU+ metric (Napoles et al., 2016) on the JFLEG dev and test sets. Table 5 reports statistics computed over the development and test sets.

Test/Dev Set	# sentences	# annotators	Metric
CoNLL-2014 dev	1345	1	M^2
CoNLL-2014 test	1312	2	M^2
JFLEG dev	754	4	GLEU
JFLEG test	747	4	GLEU

Table 5: Statistics for test/dev data.

6.2 Data from Wikipedia Revisions

In extracting examples from Wikipedia revision histories, we set a number of variables, selecting

Revision Dataset	CoNLL-2014			JFLEG
	Prec.	Rec.	$F_{0.5}$	GLEU$^+$
Revisions				
Default setting	62.7	24.3	47.7	56.9
Max-edit-28	57.3	28.0	47.4	57.1
Max-edit-6	58.3	25.7	46.5	56.1
Dwnsample-1.35	47.0	35.1	44.0	56.4
All	58.3	25.1	46.1	56.8
+ finetuning on Lang-8				
Default setting	68.8	32.3	56.1	61.7
Max-edit-28	59.6	40.9	54.6	61.8
Max-edit-6	65.5	37.1	56.8	61.6
Dwnsample-1.35	62.7	39.9	56.3	61.3
All	64.5	36.2	55.8	62.0
Lang-8 only	41.2	16.4	31.7	52.8

Table 6: Performance of the models trained on variants of data extracted from Wikipedia revision histories (top panel) and then fine-tuned on Lang-8 (bottom panel), and of a model trained only on Lang-8 with the same architecture.

rate of revision downsampling, and maximum edit distance. We generate four data sets using variations of these values: *Default setting* uses the default values described in Section 2, *Max-edit-28* and *Max-edit-6* correspond to maximum edit distance of 28 and 6 wordpieces respectively, and *Dwnsample-1.35* corresponds to a revision downsampling rate of $\log_{1.35}(n)$ for a page with a total of n revisions (whereas the default setting uses a rate of $\log_{1.5}(n)$). We train a fifth model on the union of the datasets. Table 6 shows that varying the data generation parameters led to modest variation in performance, but training on the union of the diverse datasets did not yield any benefit. Fine-tuning yields large improvements for all models. As a sanity check, we also trained a model only on Lang-8 with the same architecture. All pre-trained and fine-tuned models substantially outperform this Lang-8 only model, confirming the usefulness of pre-training.

6.3 Round Trip Translations

As for the Revision data, we train a model on each of the round-trip translation datasets, and a fifth model on the union of their data, then fine-tune all models. The results are shown in Table 7. Using Japanese as the bridge language gives the best performance on CoNLL-2014, even when compared to the model trained on all round-trip data. This is likely because the error patterns generated using Japanese round-trip translations are very similar to those in CoNLL-2014 set, created from non-native speakers of English (Ng et al., 2014). Pooling all round-trip translations dilutes this similarity and lowers performance on CoNLL-2014. However, the model trained on all data performs best on the JFLEG set, which has a different distribution of errors relative to CoNLL-2014 (Napoles et al., 2017). After fine-tuning, all round-trip models perform considerably better than the Lang-8 model.

Bridge Language	CoNLL-2014			JFLEG
	Precision	Recall	$F_{0.5}$	GLEU$^+$
Round-Trip Translations				
French	33.6	21.9	30.3	50.6
German	36.4	21.2	31.8	51.3
Russian	33.5	21.1	30.0	50.5
Japanese	35.7	51.3	38.1	46.2
All	38.1	27.1	35.2	52.1
+ finetuning on Lang-8				
French	57.9	39.9	53.1	60.9
German	56.4	42.1	52.8	61.5
Russian	60.8	32.5	51.7	59.9
Japanese	60.9	38.6	54.6	61.5
All	62.1	40.0	56.0	61.6
Lang-8 only	41.2	16.4	31.7	52.8

Table 7: Performance of the models trained on the round-trip translations (top panel) and fine-tuned on Lang8 (bottom panel) and of a model trained only on Lang-8 with the same architecture.

6.4 Combining Data Sources

Having generated multiple diverse datasets, we investigate strategies for utilizing combinations of data from multiple sources. For each corpus, we train a single model on all data and compare its performance to an ensemble of the 4 individually-trained models (Table 8). The ensemble clearly outperforms the single model for both types of data. We additionally train a single model on the union of all Revisions and Round-Trip Translated datasets reported on in Tables 6 and 7, which we compare to an ensemble of the 8 models trained individually on those datasets.

When Wikipedia edits are combined with the round-trip translations, the single-model performance remains unchanged on CoNLL-2014, while the ensemble shows an improvement. This suggests that when utilizing disparate sources of data, an ensemble is preferable to combining the data.

6.5 Comparison with Other systems

We compare the performance of our best individual system, trained on all revisions, the best ensemble of 8 models trained from both revisions and roundtrip translations on the CoNLL-2014 and JFLEG datasets (Table 9). We only report performance of models that use publicly available

Model	CoNLL-2014			JFLEG
	Precision	Recall	$F_{0.5}$	GLEU$^+$
Revisions				
All	64.5	36.2	55.8	62.0
Ensemble (4)	66.3	42.3	59.0	62.9
Round-Trip Translations				
All	62.1	40.0	56.0	61.6
Ensemble (4)	63.5	47.0	59.3	63.2
Revisions + Round-Trip Translations				
All	65.8	35.2	56.1	62.6
Ensemble (8)	66.7	43.9	60.4	63.3

Table 8: Combining datasets using either a single model trained on all data versus an ensemble of models. All models are fine-tuned on Lang-8.

Lang-8 and CoNLL datasets. Our single system trained on all revisions outperforms all previous systems on both datasets, and our ensemble improves upon the single system result[6].

7 Error Analysis

All models trained on Wikipedia-derived data are demonstrated to benefit significantly from finetuning on Lang-8 (Tables 6 and 7). In Table 10, we compare example corrections proposed by two Wikipedia-derived models to the corrections proposed by their fine-tuned counterparts. The changes proposed by the revisions-trained model often appear to be improvements to the original sentence, but fall outside the scope of GEC. Models finetuned on Lang-8 learn to make more conservative corrections.

The finetuning on Lang-8 can be viewed as an adaptation technique that shifts the model from the Wikipedia-editing task to the GEC task. On Wikipedia, it is common to see substantial edits that make the text more concise and readable, e.g. replacing “which is RFID for short” with “(RFID)”, or removing less important clauses like “Then we can see that”. But these are not appropriate for GEC as they are editorial style fixes rather than grammatical fixes. The models trained on round-trip translation seem to be make fewer drastic changes.

Table 11 reports $F_{0.5}$ across broad error categories for models trained from revisions and round-trip translations on the CoNLL-2014 test set. The error categories were tagged using the approach in Bryant et al. (2017). Although the overall $F_{0.5}$ of the 2 ensembles are similar, there are notable differences on specific categories. The ensemble using round-trip translation performs considerably better on prepositions and pronouns while the revision ensemble is better on morphology and orthography. Thus, each system may have advantages on specific domains.

8 Related Work

Progress in GEC has accelerated rapidly since the CoNLL-2014 Shared Task (Ng et al., 2014). Rozovskaya and Roth (2016) combined a Phrase Based Machine Translation (PBMT) model trained on the Lang-8 dataset (Mizumoto et al., 2011) with error specific classifiers. Junczys-Dowmunt and Grundkiewicz (2016) combined a PBMT model with bitext features and a larger language model. The first Neural Machine Translation (NMT) model to reach the state of the art on CoNLL-2014 (Chollampatt and Ng, 2018) used an ensemble of four convolutional sequence-to-sequence models followed by rescoring. The current state of the art ($F_{0.5}$ of 56.25 on CoNLL-2014) using publicly available Lang-8 and CoNLL data was achieved by Grundkiewicz and Junczys-Dowmunt (2018) with a hybrid PBMT-NMT system. A neural-only result with an $F_{0.5}$ of 56.1 on CoNLL-2014 was reported by Junczys-Dowmunt et al. (2018) using an ensemble of neural *Transformer* models (Vaswani et al., 2017), where the decoder side of each model is pretrained as a language model. From a modeling perspective, our approach can be viewed as a direct extension of this last work. Rather than pretraining only the decoder as a language model, we pretrain on a large amount of parallel data from either Wikipedia revision histories or from round-trip translations. While pretraining on out-of-domain data has been employed previously for neural machine translation (Luong and Manning, 2015), it has not been presented in GEC thus far, perhaps due to the absence of such large datasets. Tao et al. (2018b) apply iterative decoding, where two neural models, trained in left-to-right and right-to-left directions, are applied in an interleaved manner. Similar to their study, we find that iterative decoding can improve the performance of GEC.

Prior work (Brockett et al., 2006; Foster and Andersen, 2009; Rozovskaya and Roth,

---

[6] Using non-public sentences beyond the regular Lang-8 and CoNLL datasets, Tao et al. (2018b) recently obtained an $F_{0.5}$ of 61.3 on CoNLL-2014 and a GLEU of 62.4 on JFLEG. Using finetuning data beyond the standard datasets, we obtain an $F_{0.5}$ of 62.8 on CoNLL-2014 and a GLEU of 65.0 on JFLEG.

	Model	CoNLL-2014			JFLEG
		Precision	Recall	$F_{0.5}$	GLEU$^+$
Chollampatt and Ng (2018)	MLConv$_{embed}$	60.9	23.7	46.4	51.3
	Ensemble (4) +EO +LM +SpellCheck	65.5	33.1	54.8	57.5
Junczys-Dowmunt et al. (2018)	Single Transformer			53.0	57.9
	Ensemble (4)	63.0	38.9	56.1	58.5
	Ensemble (4) +LM	61.9	40.2	55.8	59.9
Grundkiewicz and Junczys-Dowmunt (2018)	Hybrid PBMT +NMT +LM	66.8	34.5	56.3	61.5
Best Single Model		65.5	37.1	56.8	61.6
Best Ensemble		66.7	43.9	**60.4**	**63.3**

Table 9: Comparison of recent state-of-the-art models (top) and our best single-system and ensemble models (bottom) on the CoNLL-2014 and JFLEG datsets. Only systems trained with publicly available Lang-8 and CoNLL datasets are reported.

Original	Recently, a **new coming** surveillance technology called radio-frequency identification **which is RFID for short** has caused heated discussions on whether it should be used to track people.
Revisions	Recently, a surveillance technology called radio frequency identification **(RFID)** has caused heated discussions on whether it should be used to track people.
+finetuning	Recently, a **new** surveillance technology called radio-frequency identification**, which is RFID for short,** has caused heated discussions on whether it should be used to track people.
Ensemble	Recently, a new coming surveillance technology called radio-frequency identification, which is RFID for short**,** has caused heated discussions on whether it should be used to track people.
Round-Trip	Recently, a new coming surveillance technology called radio-frequency identification which is RFID for short has caused heated discussions on whether it should be used to track people.
+finetuning	Recently, a **new upcoming** surveillance technology called radio-frequency identification which is RFID for short has caused heated discussions on whether it should be used to track people.
Ensemble	Recently, a **new** surveillance technology called radio-frequency identification which is RFID for short has caused heated discussions on whether it should be used to track people.
Original	**Then we can see that** the rising life **expectancies** can also be viewed as a challenge for **us** to face.
Revisions	**The** rising life **expectancy** can also be viewed as a challenge for **people** to face.
+finetuning	**Then we can see that** the rising life **expectancy** can also be viewed as a challenge for **us** to face.
Ensemble	Then we can see that the rising life expectancies can also be viewed as a challenge for us to face.
Round-Trip	Then we can see that the rising life expectancies can also be viewed as a challenge for us to face.
+finetuning	Then we can see that the rising life **expectancy** can also be viewed as a challenge for us to face.
Ensemble	Then we can see that the rising life expectancies can also be viewed as a challenge for us to face.

Table 10: Corrections from models trained on (a) Wikipedia revisions and (b) round-trip translations using Japanese as a bridge language, along with suggestions from their Lang-8 finetuned counterparts. Also shown are the corrections from the ensembles of 4 wikipedia models as well as 4 models trained on round trip translations. Example sentences are from the CoNLL-2014 dev set.

2010), (Felice et al., 2014; Xie et al., 2016; Rei et al., 2017) has investigated multiple strategies for generating artificial errors in GEC. Cahill et al. (2013) show that preposition corrections extracted from Wikipedia revisions improve the quality of a GEC model for correcting preposition errors. Back-translation (Sennrich et al., 2016a; Xie et al., 2018) addresses data sparsity by introducing noise into a clean corpus using a translation model trained in the clean to noisy direction. However, training such a reverse translation model also requires access to parallel data which is scarce for GEC. In contrast, round-trip translation attempts to introduce noise via bridge translations. Round-trip translations have been investigated for GEC. Madnani et al. Madnani et al. (2012) combine round-trip translations to generate a lattice from which the best correction is extracted using a language model. Désilets et al. (2009) use round-trip translations for correcting preposition errors. In contrast to these approaches, we employ round-trip translations for generating a large parallel training corpus for neural GEC models.

9 Discussion

Motivated by data scarcity for the GEC task, we present two contrasting approaches for generating large parallel corpora from the same publicly available data source. We believe both techniques offer promising research avenues for further development on the task.

We show that models trained exclusively on minimally filtered English Wikipedia revisions can already be valuable for the GEC task. This approach can be easily extended to the many other languages represented in Wikipedia, presenting an

Error Type	Revisions			Round-trip Translations		
	Pre-trained	Fine-tuned	Ensemble	Pre-trained	Fine-tuned	Ensemble
Adjective	16.9	29.4	36.6	14.4	27.8	37.9
Adverb	31.5	39.7	43.5	21.7	33.3	44.6
Determiner	31.3	57.2	59.4	27.4	57.7	59.5
Morphology	64.5	66.1	66.1	38.7	59.3	62.0
Noun	24.1	28.6	33.2	8.6	27.5	32.4
Orthography	69.4	57.1	69.6	19.2	58.6	57.9
Preposition	33.0	49.2	55.6	30.3	52.7	61.9
Pronoun	34.9	34.1	44.6	24.4	41.7	50.1
Punctuation	26.7	29.5	36.4	29.8	18.4	33.3
Spelling	60.6	69.2	66.7	51.0	58.5	62.5
Verb	36.1	47.1	43.2	20.7	45.2	43.2
Word Order	45.5	33.3	52.1	34.8	42.9	45.5

Table 11: $F_{0.5}$ across error categories on the CoNLL-2014 test set.

opportunity to extend GEC into languages that may have no extant GEC corpora. While we expect pre-training on Wikipedia to give us a reasonable model, it may be crucial to fine-tune this model on small amounts of clean, in-domain corpora to achieve good performance.

When extracting examples from the Wikipedia revisions, we implemented minimal filtration in pursuit of simplicity, and to produce a sufficiently large dataset. Implementing more complex filtration in order to reduce the noise in the generated dataset will likely be a productive avenue to increase the value of this approach. The performance achieved by the reported Wikipedia revisions-trained models, both with and without finetuning, may be used as a baseline by which to evaluate smaller, cleaner datasets drawn from Wikipedia revisions.

Round-trip translation takes advantage of the advanced state of the task of Machine Translation relative to GEC by leveraging extant translation models as a source of grammatical-style data corruption. In this work, we only experiment with producing English-language GEC corpora, but this technique can be extended to any of the many languages for which translation models exist. It would be useful to assess how the translation quality influences the performance of the resulting GEC model. In our experiments with round-trip translation, we used target sentences drawn from Wikipedia to maintain a reasonable comparability between the two techniques. However, there is no constraint preventing the application of round-trip translation to diverse data sources; any source of clean text can be turned into a parallel GEC corpus. This can be used to increase diversity in the generated data, or to generate domain-specific GEC corpora (e.g. patents).

We observe that pooling two diverse data sources used to train competitively performing models on the same task can degrade performance. This suggests that within datasets useful for a specific task, there may be greater value to be discovered in finding optimal partitions of the data for training models which can then be combined using ensembles. Prior work in combining diverse data sources includes addition of special tokens (Johnson et al., 2017) and meta-learning (Finn et al., 2017). We intend to compare ensembling with these alternatives.

We have opensourced the scripts used to extract example pairs from Wikipedia, which we hope will become a resource in the further development of models for GEC as well as other NLP tasks that rely on edit histories, such as sentence re-writing (Botha et al., 2018) and text simplification (Tonelli et al., 2016).

Acknowledgements

We thank Jayakumar Hoskere, Emily Pitler, Slav Petrov, Daniel Andor, Alla Rozovskaya and Antonis Anastasopoulos for helpful suggestions. We also thank Jayakumar Hoskere, Shruti Gupta and Anmol Gulati for providing various GEC resources that were used in this paper.

References

Jan Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from wikipedia edit history. In *Proceedings of EMNLP*.

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the*

44th annual meeting of the Association for Computational Linguistics, pages 249–256. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of ACL*.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust systems for preposition error correction using wikipedia revisions. In *Proceedings of NAACL*, pages 507–517. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. arXiv:1801.08831.

Fabian Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to wat 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A beam-search decoder for grammatical error correction. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better evaluation for grammatical error correction. In *Proceedings of NAACL*.

A. Désilets and Hermet M. 2009. Using automatic roundtrip translation to repair general errors in second language writing. In *Proceedings of MT Summit XII*.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

Jennifer Foster and Øistein E. Andersen. 2009. Generrate: Generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, EdAppsNLP '09, pages 82–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *International Conference on Natural Language Processing*, pages 478–490. Springer.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv:1804.05945*.

Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. 2014. Predicting grammaticality on an ordinal scale. In *Proceeedings of ACL*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5(1):339–351.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of EMNLP*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv:1804.05940*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Exploring grammatical error correction with not-so-crummy machine translation. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, NAACL HLT '12, pages 44–53, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated japanese error correction of second language learners. In *Proceedings of IJCNLP*, pages 147–155.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU without tuning. arXiv:1605.02592.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of EACL*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *CoNLL Shared Task*, pages 1–14.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of ACL*.

Michael Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *Proceedings of ICASSP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv:1804.04235*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

Ge Tao, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of ACL*.

Ge Tao, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammar error correction: An empirical study. arXiv:1807.01270.

Sara Tonelli, Alessio Palmero Aprosio, and Francesca Saltori. 2016. Simpitiki: a simplification corpus for italian. In *Proceedings of CLiC-it*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv:1603.09727*.

Ziang Xie Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of NAACL*.

Structural Supervision Improves Learning of Non-Local Grammatical Dependencies

Ethan Wilcox[1], **Peng Qian**[2], **Richard Futrell**[3], **Miguel Ballesteros**[4], and **Roger Levy**[2]

[1]Department of Linguistics, Harvard University, `wilcoxeg@g.harvard.edu`
[2]Department of Brain and Cognitive Sciences, MIT, `{pqian,rplevy}@mit.edu`
[3]Department of Language Science, UC Irvine, `rfutrell@uci.edu`
[4]IBM Research, MIT-IBM Watson AI Lab `miguel.ballesteros@ibm.com`

Abstract

State-of-the-art LSTM language models trained on large corpora learn sequential contingencies in impressive detail and have been shown to acquire a number of non-local grammatical dependencies with some success. Here we investigate whether supervision with hierarchical structure enhances learning of a range of grammatical dependencies, a question that has previously been addressed only for subject-verb agreement. Using controlled experimental methods from psycholinguistics, we compare the performance of word-based LSTM models versus two models that represent hierarchical structure and deploy it in left-to-right processing: Recurrent Neural Network Grammars (RNNGs) (Dyer et al., 2016) and a incrementalized version of the Parsing-as-Language-Modeling configuration from Charniak et al. (2016). Models are tested on a diverse range of configurations for two classes of non-local grammatical dependencies in English—*Negative Polarity* licensing and *Filler–Gap Dependencies*. Using the same training data across models, we find that structurally-supervised models outperform the LSTM, with the RNNG demonstrating best results on both types of grammatical dependencies and even learning many of the *Island Constraints* on the filler–gap dependency. Structural supervision thus provides data efficiency advantages over purely string-based training of neural language models in acquiring human-like generalizations about non-local grammatical dependencies.

1 Introduction

Long Short-Term Memory Recurrent Neural Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) have achieved state of the art language modeling performance (Jozefowicz et al., 2016) and have been shown to indirectly learn a number of non-local grammatical dependencies, such as subject-verb number agreement and filler-gap licensing (Linzen et al., 2016; Wilcox et al., 2018), although they fail to learn others, such as Negative Polarity Item and anaphoric pronoun licensing (Marvin and Linzen, 2018; Futrell et al., 2018). LSTMs, however, require large amounts of training data and remain relatively uninterpretable. One model that attempts to address both these issues is the Recurrent Neural Network Grammar (Dyer et al., 2016). RNNGs are generative models, which represent hierarchical syntactic structure and use neural control to deploy it in left-to-right processing. They can achieve state-of-the-art broad-coverage scores on language modeling and phrase structure parsing tasks, learn Noun Phrase headedness (Kuncoro et al., 2016), and outperform linear models at learning subject-verb number agreement (Kuncoro et al., 2018).

In this work, we comparatively evaluate LSTMs, RNNGs and a third model trained using syntactic supervision—similar to the Parsing-as-Language-Modeling configuration from Charniak et al. (2016)—by conducting side-by-side tests on two novel English grammatical dependencies, deploying methodology from psycholinguistics. In this paradigm, the language models are fed with hand-crafted sentences, designed to draw out behavior that belies whether they have learned the underlying syntactic dependency. For example, Linzen et al. (2016) and Kuncoro et al. (2018) assessed how well neural language models were able to learn subject-verb number agreement by feeding the prefix *The keys to the cabinet...* If the model assigns a relatively higher probability to the grammatical plural verb *are* than the ungrammatical singular *is* it can be said to have learned the agreement dependency. Here, we investigate two non-local dependencies that remain untested for RNNGs: Negative Polarity Item (NPI) licensing is the dependency between a negative licensor—

Proceedings of NAACL-HLT 2019, pages 3302–3312
Minneapolis, Minnesota, June 2 - June 7, 2019.

such as *not* or *none*—and a Negative Polarity Item such as *any* or *ever*. The filler–gap dependency is the dependency between a filler—such as *who* or *what*—and a gap, which is an empty syntactic position. Both dependencies have been shown to be learnable by LSTMs trained on large amounts of data (Wilcox et al., 2018; Marvin and Linzen, 2018). Here, we investigate whether, after controlling for size of the training data, explicit hierarchical representation results in learning advantages.

2 Methodology

2.1 Neural Language Models

Recurrent Neural Network LMs model a sentence in a purely sequential basis, without explicitly representing the latent syntactic structure. We use the LSTM architecture in Hochreiter and Schmidhuber (1997), deploying a 2-layer LSTM language model with hidden layer size 256, input embedding size 256, and dropout rate 0.3. We refer to this model as the "LSTM" model in the following sections.

Recurrent Neural Network Grammars (Dyer et al., 2016) predict joint probability of a sentence as well as its syntactic parse. RNNGs contain three sub-components, all of which are LSTMS: the *neural stack*, which keeps track of the current parse, the *output buffer*, which keeps track of previously-seen terminals and the *history of actions*. At each timestep the model can take three different actions: NT, which introduces a non-terminal symbol—such as a VP or NP—onto the stack; SHIFT, which places a terminal symbol onto the top of the stack, or REDUCE. REDUCE pops terminal symbols (words) off the stack until a non-terminal phrasal boundary is encountered; it then combines the terminals into a single representation via a bidirectional-LSTM and pushes the newly-reduced constituent back onto the stack. By reducing potentially unbounded constituents within the neural stack, the RNNG is able to create structural adjacency between co-dependent words that may be linearly distal. Following Dyer et al. (2016), we use 2-layer LSTMs with 256 hidden layer size for the stack-LSTM, action LSTM, and terminal LSTM, and dropout rate 0.3.

ActionLSTM: It is the combination of the neural stack and the REDUCE function that may give the RNNG an advantage over purely sequential models (such as LSTMS) or models that deploy syntactic supervision without explicit notions of compositionality. In order to assess the gains from explicitly modeling compositionality, we compare the previous two models against an incrementalized version of the Parsing-as-Language-Modeling configuration presented in Charniak et al. (2016). In this model, we strip an RNNG of its *neural stack* and *output buffer*, and train it to jointly predict the action sequence of a parse tree as well as the upcoming word. The action space of the model contains a set of non-terminal nodes (NT), terminal generations (GEN), as well as a (REDUCE) action, which functions only as a generic phrasal boundary marker. The model was trained using embedding size 256, dropout 0.3, and was able to achieve a parsing F1 score of 92.81 on the PTB, which is only marginally better than the performance of the original architecture on the same test set, as reported in Kuncoro et al. (2016). We will refer to this model as the "ActionLSTM" model in the following sections.

All three models are trained on the training-set portion of the English Penn Treebank standardly used in the parsing literature (PTB; sections 2-21), which consists of about **950,000 tokens** of English language news-wire text (Marcus et al., 1993). The RNNG and Action models get supervision from syntactic annotation–crucially, only constituent boundaries and major syntactic categories, with functional tags and empty categories stripped away—whereas the LSTM language model only uses the sequences of terminal words. We train the models until performance converges on the held-out PTB development-set data.

2.2 Psycholinguistic Assessment Paradigm

2.2.1 Surprisal

The **surprisal**, or negative log-conditional probability, $S(x_i)$ of a sentence's i^{th} word x_i, tells us how strongly x_i is expected in context and is also known to correlate with human processing difficulty (Smith and Levy, 2013; Hale, 2001; Levy, 2008). For sentences out of context, surprisal is:

$$S(x_i) = -\log p(x_i | x_1 \dots x_{i-1})$$

We investigate a model's knowledge of a grammatical dependency, which is the co-variance between an upstream *licensor* and a downstream *licensee*, by measuring the effect that an upstream licensor has on the surprisal of a downstream licensee. The idea is that grammatical licensors should set up an expectation for the licensee thus reducing its surprisal compared to minimal pairs in which the licensor is absent. We derive the word surprisal from the LSTM language model by directly computing the negative log value of the predicted conditional probability $p(x_i|x_1 \ldots x_{i-1})$ from the softmax layer.

Following the method in Hale et al. (2018) for estimating word surprisals from RNNG, we use word-synchronous beam search (Stern et al., 2017) to find a set of most likely incremental parses and sum their forward probabilities to approximate $P(x_1, \ldots x_i)$ and $P(x_1, \ldots x_{i-1})$ for computing the surprisal. We set the action beam size to 100 and word beam size to 10. We ensured that the correct incremental RNNG parses were present on the beam immediately before and throughout the material over which surprisal was calculated through manual spot inspection; the correct parse was almost always at the top of the beam.

2.2.2 Wh-Licensing Interaction

Unlike NPI, licensing, the filler—gap dependency is the covariance between a piece of extant material, a filler, and a piece of *absent* material, a gap. Here, we employ the methodology from Wilcox et al. (2018), which introduces the **Wh-Licensing Interaction**. To compute the wh-licensing interaction for a sentence, Wilcox et al. (2018) construct four variants, given in (1), that exhibit the four possible combinations of fillers and gaps for a specific syntactic position. The underscores are for presentational purposes only and were not included in experimental materials.

(1) a. I know that the lion devoured the gazelle at sunrise. [-FILLER -GAP]
b. *I know what the lion devoured the gazelle at sunrise. [+FILLER -GAP]
c. *I know that the lion devoured __ at sunrise. [-FILLER +GAP]
d. I know what the lion devoured __ at sunrise. [+FILLER +GAP]

If a filler sets up an expectation for a gap, then filled syntactic positions should be more surprising in the context of a filler than in a minimally-different, non-filler variants. We measure this expectation by calculating the difference of surprisal between (1-b) and (1-a). Similarly, if gaps require fillers to be licensed, transitions from transitive verbs to adjunct clauses that skip an obligatory argument should be less surprising in the context of a filler than in minimally-different, non-filler variants. We measure this expectation by computing the difference in surprisal between (1-c) and (1-d). Because the filler–gap dependency is a two-way interaction, the wh-licensing interaction consists of the difference of these two differences, which is given in (2).

(2) $(S(\text{1-b}) - S(\text{1-a})) - (S(\text{1-c}) - S(\text{1-d}))$

For basic filler—gap dependencies, we expect the presence of a filler to set up a global expectation for a gap, thus we measure the summed licensing interaction across the entire embedded clause, which we expect to be significantly above zero if the model is learning the dependency. Our experimental materials include only vocabulary items within the PTB, avoiding the need for Out of Vocabulary handling. We determine statistical significance using a mixed-effects linear regression model, using sum-coded conditions (Baayen et al., 2008). For within-model comparison we use surprisal as the dependent variable and experimental conditions as predictors; for between-model comparison, we use wh-licensing interaction as the dependent variable with model type and experimental conditions as predictors. All figures depict by-item means, with error bars representing 95% confidence intervals, computed by subtracting out the within-item means from each condition as advocated by Masson and Loftus (2003). The strength of a wh-licensing interaction can be interpreted as either its mean size in bits, or as its mean size normalized by its standard deviation across items. The latter is Cohen's d, rooted in signal-detection theory (**?**); because all our experiments involve similar number of items, it is roughly proportional to the size of wh-interaction relative to the size of the associated confidence interval.[1]

3 Negative Polarity Item Licensing

In English, Negative Polarity Items (NPIs), such as *any*, *ever* must be in the SCOPE of a negative LICENSOR such as *no*, *none*, or *not* (**?**Ladusaw, 1979). Crucially, the scope of a licensor is characterized structurally, not in purely linear terms; for present purposes, a sufficient approximation is that an NPI is in the proper scope of a licensor if it is c-commanded by it. Thus while *ever* in (3-b) and (3-d) is grammatical because it is licensed by *no* in the main-clause subject, *ever* is ungrammatical in (3-c) despite the linearly preceding *no*, be-

---

[1] All of our experiments were pre-registered online at `http://aspredicted.org/blind.php?x={xd9cw9, 3xv2du, jd384m, cy6zp6, 2hk4gf, zt73qt, f9pk9f, ab9f3h, yt6pi4}`

cause inside a subject-modifying relative clause is not a valid position for an NPI licensor; we call this a DISTRACTOR position.

(3) a. ***The** senator that supported the measure has ever found any support from her constituents.
b. **No** senator that supported the measure has ever found any support from her constituents.
c. ***The** senator that supported **no** measure has ever found any support from her constituents.
d. **No** senator that supported no measure has ever found any support from her constituents.

Learning of NPI licensing conditions by LSTM language models trained on large corpora has previously been investigated by Marvin and Linzen (2018) and Futrell et al. (2018). Futrell et al. found that the language models of both Gulordava et al. (2018) and Jozefowicz et al. (2016) (hereafter called 'Large Data LSTMs') learned a contingency between licensors and NPIs: the NPIs in examples like (3) were lower-surprisal when linearly preceded by negative licensors. However, both papers reported that these models failed to constrain the contingency along the correct structural lines: negative NPI surprisal was decreased at least as much by a preceding negative distractor as by a negative licensor.

Syntactic supervision might plausibly facilitate learning of NPI licensing conditions. We tested this following the method of Futrell et al. (2018), constructing 27 items on the design of in (3), with two variants: one included *ever* and omitting *any*, and one including *any* and omitting *ever*. Figure 1, left panel, shows the results. For the RNNG and the ActionLSTM, negative licensors and distractors alike reduced surprisal of both NPIs ($p < 0.05$ for the RNNG, $p < 0.001$ for the ActionLSTM). For the LSTM, negative licensors and distractors alike reduced surprisal of *ever* (both $p < 0.01$), but not *any*. This may seem surprising as *any* is considerably more frequent than *ever* (123 vs. 727 instances in the training data), but *any*'s non-NPI uses (e.g., *I will eat anything fried*) may complicate its learning.

From Figure 1 it is also apparent that the RNNG and ActionLSTM show signs of stronger NPI licensing effects from negation in the licensor position than in the distractor position, at least for *ever*. To quantify this, we follow Marvin and Linzen (2018) in computing item-mean classification accuracies, with classification being considered correct if the NPI is assigned higher probability in context for (3-b) than for (3-c). Results are shown in Figure 1, right panel. No

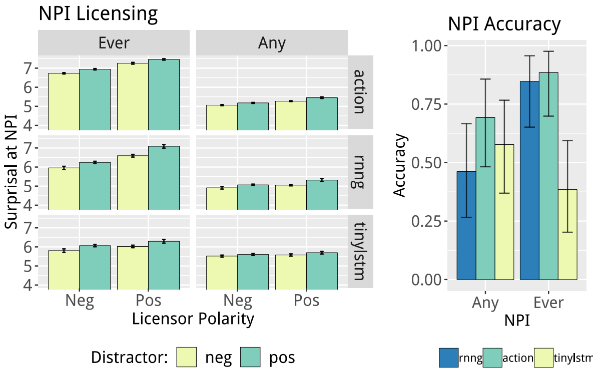


Figure 1: NPI Licensing at left: Y-axis shows surprisal at the NPI, x-axis indicates polarity of the c-commanding licensor, and color indicates distractor polarity. Licensing accuracy at right: Y-axis shows classification accuracy, x-axis indicates the NPI tested, and color indicates the model. Error bars represent 95% binomial confidence intervals.

model is significantly above chance for *any*, but for *ever* the syntactically supervised models perform much better: The RNNG reaches 85% performance, and the ActionLSTM 88%, both significantly above chance ($p < 0.001$ by binomial test for each), and are not significantly different from each other, but both better than the LSTM ($p < 0.01$ for the RNNG/LSTM; $p < 0.001$ for the ActionLSTM/LSTM by Fisher's exact test). To our knowledge this is the first demonstration of a language model learning the licensing conditions for an NPI without direct supervision.

Overall, we find that syntactic supervision facilitates the contingency of NPIs on a negative licensor in context, but is not sufficient for clean generalization of the structural conditions on NPI licensing with the training dataset used here.

4 Filler–Gap Dependencies

The dependency between a FILLER, which is a wh-word such as *who* or *what*, and a GAP, which is an empty syntactic position, is characterized by a number of properties, some of which were tested for large data LSTMs by Wilcox et al. (2018). Here we investigate the effect of syntactic supervision on filler–gap dependency learning. Syntactic annotation of the dependency itself is stripped from the training data (Figure 2), so syntactic supervision can play only an indirect facilitatory role for the models' neural learning mechanisms.

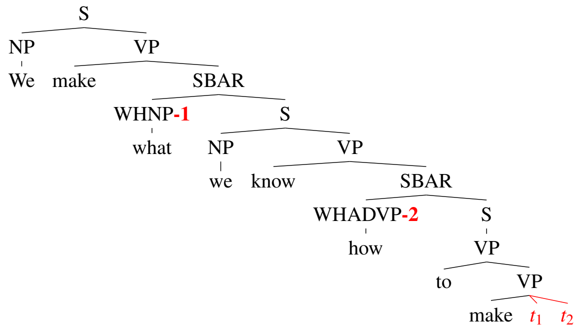


Figure 2: Example of filler–gap dependency representation in the Penn Treebank. Non-local dependency annotation indicated in bold, red font is stripped from the training data, so that the RNNG must learn about the filler-gap dependency purely through neural generalization.

Location of Gap	All Fillers	'Who'	'What'
All Positions	13907	1888	660
Subject Position	6632	1510	236
Object Position	2080	12	332
Indirect Object Position	57	0	6

Table 1: Filler—Gap Dependency Statistics for the Penn Treebank training data (used for both models).

4.1 Flexibility of Gap Position

The filler–gap dependency is flexible: a filler can license a gap in any of a number of syntactic positions, including the argument positions of subject, object, and indirect object, as illustrated in (4), as well as in other positions (e.g. the adjunct position for *how* in Figure 2).

(4) a. I know who __ introduced the accountant to the guests after lunch.
b. I know who the CEO introduced __ to the guests after lunch.
c. I know who the CEO introduced the accountant to __ after lunch.

These gap positions differ in frequency, however (Table 1): the majority (63.1%) are in some argument structure position, of which the vast majority (75.6%) are subject position (mostly subject-extracted relative clauses), 23.7% are object position, and 0.7% are indirect object position.

Using the wh-interaction measure described in Section 2.2, Wilcox et al. (2018) showed that large-data LSTMs learn filler–gap dependencies for all three argument positions, with the size of the wh-interaction generally largest for subject gaps and smallest for indirect-object gaps. Table 1 suggests that this gradation may reflect frequency of learning signal, with the dependency being learned more robustly the more frequent the extraction type. We applied the same method, adapting Wilcox et al.'s items to the smaller training dataset. The results can be seen in the upper-left panel of Figure 3.

All three models learn the filler-gap dependency for subject and object positions, and there is suggestive but inconclusive evidence for learning in the rare indirect object position. We see stronger dependency learning for more frequent gap types, as was found for large data LSTMs, and the supervised models show a much stronger wh-licensing effect than the LSTM.

4.2 Syntactic Hierarchy

As with NPIs, the filler–gap dependency is subject to a number of hierarchical, structural constraints. The most basic of these constraints is that the filler must be "above" the gap in the appropriate structural sense (to a first approximation, the filler must *c*-command the gap, though see e.g. **?** for qualifications). Hence *who* in (5-a) is a legitimate extraction from the relative clause, but (5-b) is ungrammatical as the gap is in the matrix clause, above the filler.

(5) a. The policeman who the criminal shot __ with his gun shocked the jury during the trial.
b. *The policeman who the criminal shot the politician with his gun shocked __ during the trial.

A model that properly generalizes this constraint on the filler–gap dependency should *not* show a wh-interaction for cases like (5-b): an undischarged *who* filler should not make the matrix-clause gap particularly more expected. As far as we are aware, no prior work has investigated this property of the filler–gap dependency in language models; we do so here. Because the context in (5) does not allow for an immediate *that* clause initiation for the –FILLER condition as in (1), we instantiate this condition by contrasting the +FILLER,+GAP condition of (5-b) with the variants in (6), where the *who* filler is immediately discharged as the RC verb's extracted subject:

(6) a. *The policeman who __ knows that the criminal shot the politician with his gun shocked __ during the trial. -FILLER,+GAP
b. *The policeman who the criminal shot the politician with his gun shocked the jury during the trial. +FILLER,–GAP
c. The policeman who __ knows that the criminal shot the politician with his gun shocked the jury during the trial. -FILLER,–GAP

We created 22 items following the templates of (5-a) (*Subject* condition) and (5-b) (*Matrix* condition); results are shown in the top-right panel of 3. The supervised models show a large

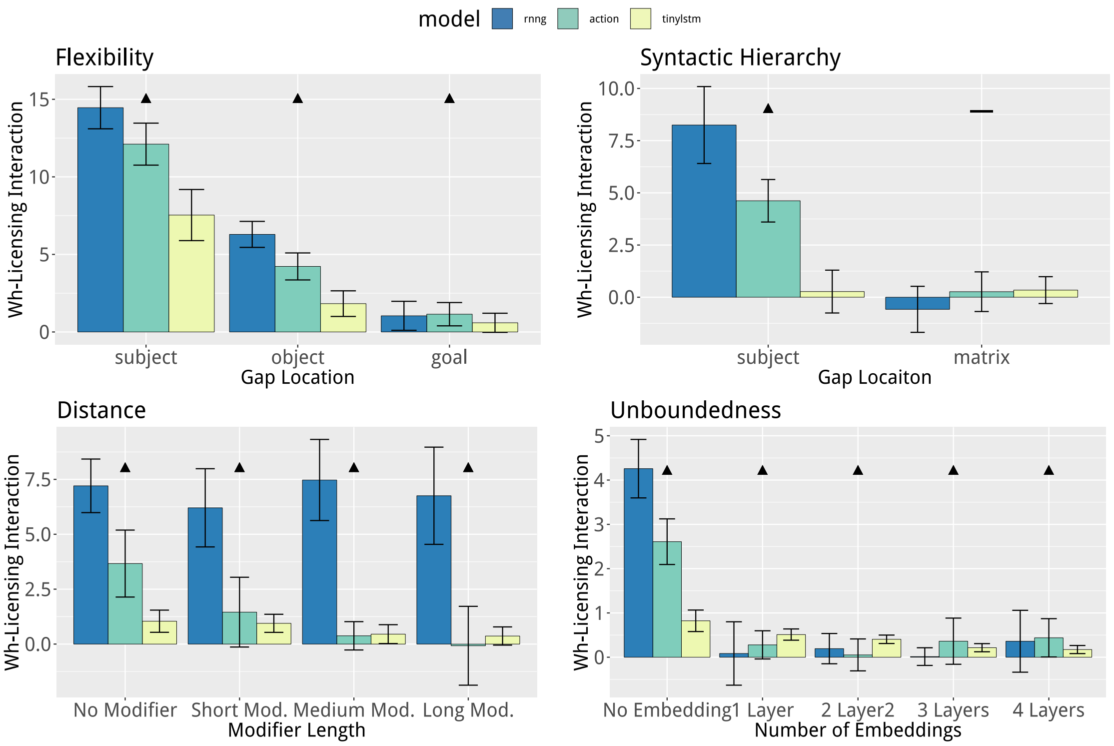


Figure 3: Model results for the basic properties of filler–gap licensing. "▲" indicates grammatical conditions in which models should display strong wh-licensing interaction, "–" indicates ungrammatical conditions in which models should display reduced wh-licensing interaction. The RNNG model significantly outperforms the LSTM model in 8/13 grammatical cases; the ActionLSTM model outperforms the LSTM model in 5/13 cases; and the RNNG outperforms the ActionLSTM model in 6/13 cases where strong licensing is expected.

wh-licensing interaction effect for a gap inside the subject-modifying relative clause—with the RNNG demonstrating more licensing interaction than the ActionLSTM—and neither model inappropriately generalizes this licensing effect to a matrix-clause gap. The LSTM shows no wh-licensing effects in either position, suggesting that syntactic supervision facilitates appropriately generalized filler-gap dependencies for subject-modifying relative clauses.[2]

4.3 Robustness to Intervening Material

For a model that learns human-like syntactic generalizations and maintains accurate phrase-like representations throughout a string, filler–gap dependencies should be robust to linearly intervening material that does not change the tree-structural relationship between the filler and the gap. Wilcox et al. (2018) found that the large-data RNNs described earlier exhibit a robust wh-interaction of this type, by introducing an optional postnominal modifier between filler and gap to sentence templates like (7), with no modification (7-a), short (3–5 word) modifiers (7-b), medium (6–8 word) modifiers (7-c), and long (8–12 word) modifiers (7-d).

(7) a. I know what your friend gave __ to Alex last weekend.
b. I know what your friend in the hat gave __ to Alex last weekend.
c. I know what your friend who you ate lunch with yesterday gave __ to Alex last weekend.
d. I know what your friend who recently took you on a walking tour of the city gave __ to Alex last weekend.

We adapted their materials for the small training dataset and tested our three models; results are shown in 3, bottom-left panel. The RNNG shows a robust licensing interaction that does not diminish with additional intervening material (all $d > 1.3$). The LSTM shows smaller wh-licensing interactions across the board; these are still substantial in the *No Modifier* and *Short Modifier* conditions ($d = 0.88, d = 0.98$, respectively), but are smaller in the *Medium Modifier* and *Long Modifier* conditions ($d = 0.45, d = 0.37$ respectively), suggesting less robustness to intervening material. The

---

[2] Results for the Larger Data LSTM models for the Hierarchy and Unboundedness experiments presented here can be found in the appendix.

ActionLSTM shows strong interactions in the *No Modifier* condition ($d = 1.02$), but weak interaction once any modifying material is introduced ($d < 0.4$ in all other conditions). This result is significant, as it indicates that RNNG is able to leverage the structural locality afforded by the neural stack to maintain robust gap expectancy.

4.4 Unboundedness

For humans, filler–gap dependencies are not only robust to linearly intervening material that does not change their tree-structural relationship, they can be STRUCTURALLY NON-LOCAL as well, propagating through intervening syntactic structures (subject to constraints examined in Section 5). For example, a filler can be extracted from multiply-nested complement clauses as in (8-b):

(8) a. I know who your aunt insulted __ at the party.
b. I know who the chauffeur said [$_S$ the hostess believed [$_S$ the butler reported [$_S$ her friend thinks your aunt insulted __ at the party.]]]

Humans show sensitivity to a single layer of sentential embedding when processing filler—gap dependencies in an offline 'complexity rating' task (Phillips et al., 2005). This may due to the relative frequency of single versus doubly-embedded filler—gap dependencies. In our training data there were 13,907 examples of filler—gap dependencies, however only 758 examples that spanned two layers of sentential embedding and 19 that spanned three layers. There were no instances of filler—gap dependencies spanning over more than three sentential embeddings, as in (8-b).

The unboundedness of filler–gap dependencies has not previously been tested for contemporary language models. To do this, we constructed 22 test items like (8), varying embedding depth within-item between zero, one, two, three, and four levels, and measured the resulting licensing interactions. The results are in Figure 3, bottom-right panel. No model's filler–gap dependency is perfectly robust to clausal embedding. The LSTM's wh-licensing interaction starts out small and diminishes with embedding depth. The RNNG and ActionLSTM show strong wh-licensing interaction in the unembedded condition but no significant wh-licensing interaction after even a single layer of embedding. Since these experimental materials are new, we also tested the large-data LSTMs on them, which exhibited much larger and more robust filler–dependency effects (Appendix B). Hence the syntactic supervision explored here is not sufficient to guarantee that learned filler–gap dependencies can be structurally unbounded.

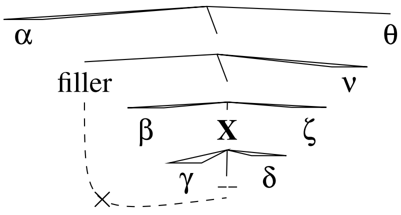


Figure 4: Anatomy of an island constraint. If node **X** is an island, then a filler outside **X** cannot associate with a gap inside **X**. For our analyses, successful learning of an island constraint implies that we should *not* see a wh-licensing interaction at the first part of the material δ immediately following the potential gap site.

5 Island Constraints

A crucial exception to the flexibility and unboundedness of filler–gap dependencies is that ISLAND CONSTRAINTS prevent association of a filler and a gap through certain types of syntactic nodes, illustrated in Figure 4 (Ross, 1967). Contemporary theories variously attribute island effects to grammatical rules, incremental processing considerations, or discourse-structural factors (Ambridge and Goldberg, 2008; Hofmeister and Sag, 2010; Sprouse and Hornstein, 2013). In our setting, a language model is sensitive to an island constraint if it *fails* to show a wh-licensing interaction between a filler and a gap that cross an island. Wilcox et al. (2018) found evidence that large-data LSTMs are sensitive to some island constraints (although see Chowdhury and Zamparelli (2018) for a contrasting view), but not to others. Here we investigate whether LSTMs would learn these from smaller training datasets, and if an RNNG's syntactic supervision provides a learning advantage for island constraints. In this section we measure the wh-licensing interaction in the material immediately following the potential gap site, which is guaranteed to implicate the model's (lack of) expectation for a gap inside the island, rather than throughout the entire embedded clause, which also implicates filler-driven expectations after the end of the island.

5.1 Adjunct Islands

Adjunct clauses block the filler–gap dependency. Wilcox et al. (2018) found evidence that large-data LSTMs are sensitive to adjunct islands, as evidenced by attenuated and often fully eliminated wh-licensing interactions for materials like (9-b)–(9-c) relative to (9-a) below. (In this and the subsequent subsections, the post-gap material used for wh-interaction computation is in **bold**.)

(9) a. The director discovered what the robbers stole __ **last night**. [Object]
b. *The director discovered what the security guard slept while the robbers stole __ **last night**. [Adj-back]
c. *The director discovered what, while the robbers stole __ **last night**, the security guard slept. [Adj-front]

We adapted these materials; results are in Figure 5, upper-left panel. The RNNG shows a strong licensing interaction in the baseline main-clause object extraction position, but no licensing interaction for a gap in an adjunct either at the back or front of the main clause. Because RNNGs failed our test for unboundedness of filler–gap dependency, however (Section 4.4), this result is inconclusive as to whether anything corresponding to an island constraint is learned. The LSTM and the ActionLSTM show no sign of filler–gap dependency attenuation from adjunct islands, in contrast to previous findings using the LSTM architecture on much larger training datasets.

5.2 Wh Islands

Embedded sentences introduced by *wh-* words are also islands; hence, (10-c) is anomalous but (10-a) and (10-b) are not.

(10)a. I know what the guide said the lion devoured __ **yesterday**. [Null Comp]
b. I know what the guide said that the lion devoured __ **yesterday**. [That Comp]
c. *I know what the guide said whether the lion devoured __ **yesterday**. [Wh- Comp]

Wilcox et al. (2018) found that the large-data LSTMs learned this island constraint: the wh-licensing interaction was eliminated or severely attenuated for the Wh-Complementizer variant but not for the other variants. Results for our three models are in Figure 5, top-right panel. These materials paint a slightly more optimistic picture than the results of Section 4.4 for the RNNG's ability to propagate a gap expectation from a filler down one level of clausal embedding. However, no models show an appreciable attenuation in the Wh- Comp condition that would suggest an island constraint-like generalization.

5.3 Complex Noun-Phrase Islands

Extractions from within clauses dominated by a lexical head noun are unacceptable; this is the Complex Noun Phrase Constraint. For example, (11-b) and (11-c) are unacceptable object extractions compared with (11-a); the same acceptability pattern holds for subject extractions.

(11)a. I know what the collector bought __ **last week**. [Argument extraction]
b. *I know what the collector bought the painting which depicted __ **last week**. [Wh- Complex NP]
c. *I know what the collector bought the painting that depicted __ **last week**. [That- Complex NP]

Wilcox et al. (2018) found that large-data LSTM behavior reflected this island constraint, with attenuated wh-licensing interactions for complex NPs like (11-b)–(11-c) and for analogous complex NPs involving subject extractions. Our results for adaptations of their materials are shown in Figure 5, bottom-left panel. All three models show attenuated wh-licensing interactions inside complex NPs in subject position, with the licensing interaction in the grammatical argument structure position greatest for the RNNG and ActionLSTM. This may be taken as an indication of Complex NP Constraint-like learning, but is inconclusive due to the models' general failure to propagate gap expectations into embedded clauses (Section 4.4).

5.4 Subject Islands

Prepositional phrases attaching to subjects are islands: this is the Subject Constraint, and accounts for the unacceptability of (12-d) compared to (12-c) (Huang, 1998).

(12)a. I know what the collector bought __ **yesterday**. [Obj Verbal-Arg]
b. I know what the collector bought a painting of __ **yesterday**. [Obj Prep-Arg]
c. I know what __ **sold** for a high price at auction. [Subj Verbal-Arg]
d. *I know what a painting of __ **sold** for a high price at auction. [Subj Prep-Arg]

Wilcox et al. (2018) found that the wh-licensing interactions of large-data LSTMs fail to distinguish between subject-modifying PPs, which cannot be extracted from, and object-modifying PPs, which can. Our results for adaptations of their materials can be seen in Figure 5, bottom right panel. The syntactically supervised models show a significant decrease between the verbal argument and prepositional argument conditions in subject position ($p < 0.001$ for RNNG; $p < 0.01$ for ActionLSTM), and no significant difference between the two conditions in object position (however, note that the licensing in object position is significantly less than the licensing in the grammatical, *Verbal Argument Subject* position, following the pattern in 4.1). LSTMs fare worse, showing a clear wh-

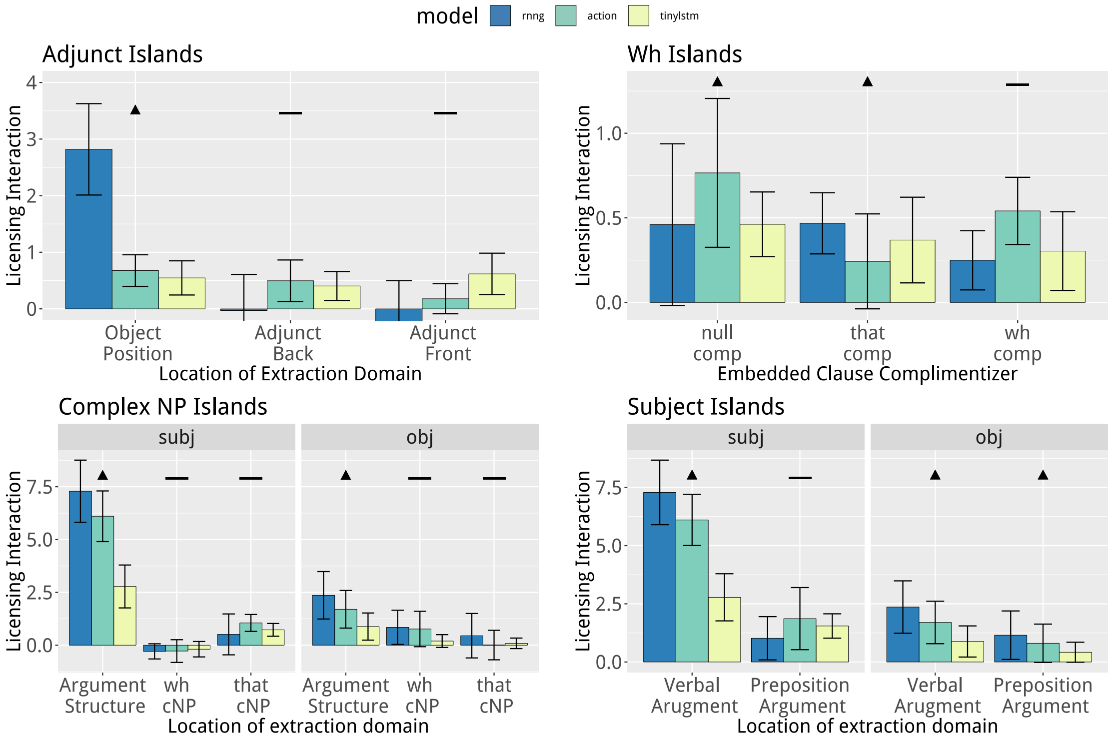


Figure 5: Model results for Syntactic Islands. "▲" indicates grammatical conditions in which models should display strong wh-licensing interaction, "–" indicates ungrammatical conditions in which models should display reduced wh-licensing interaction.

licensing interaction for subject-modifying PPs, which should be islands, and no wh-licensing interaction for object-modifying PPs.

6 Conclusion

In this paper we have argued that structural supervision provides advantages over purely string-based training of neural language models in acquiring more human-like generalizations about non-local grammatical dependencies. We have also demonstrated how the neural compositionality of the RNNG architecture can provide even further advantages, especially at maintaining expectations into structurally-local but linearly distant material. We compared RNNG, ActionLSTM and LSTM models using recently developed controlled experimental materials, and developed additional experimental materials to further test several characteristics of grammatical dependency learning for neural language models (Sections 4.2, 4.4). We found advantages for syntactic supervision in learning conditions for **Negative Polarity Item licensing** and a majority of tests involving **filler–gap dependencies**, showing particularly strong wh-licensing effects in tree-structurally-local contexts. On basic filler—gap dependency properties the RNNG significantly outperformed the LSTM in 8/13 and the ActionLSTM outperformed the LSTM on 5/13 cases where strong licensing interaction was expected. While the RNNG, and to some extent the ActionLSTM, exhibited more humanlike behavior than the LSTM for a number of **Island Constraints**, the tests were inconclusive due to the models' failure to propagate gap expectation into embedded clauses: island-like behavior may merely be sensitivity to general syntactic complexity, not the highly-specific syntactic arrangements that constitute the family of island constructions. Thus, major-category supervision does not provide enough information for the neural component to learn fully robust and human-like filler—gap dependencies from 1-million words alone. However, for some dependencies tested (i.e. NPIs) structural supervision on 1 million words provides better outcomes than even large-data LSTMs. Scaling the gains derived from structural supervision is a challenge for data-scarce NLP and is the basis for future work.

Acknowledgments

This work was supported by the MIT-IBM Watson AI Lab.

References

Ben Ambridge and Adele E Goldberg. 2008. The island status of clausal complements: Evidence in favor of an information structure explanation. *Cognitive Linguistics*, 19(3):357–389.

R Harald Baayen, Douglas J Davidson, and Douglas M Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412.

Eugene Charniak et al. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. Rnn simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. Rnns as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R Brennan. 2018. Finding syntax in human encephalography with beam search. *arXiv preprint arXiv:1806.04127*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Philip Hofmeister and Ivan A Sag. 2010. Cognitive constraints and island effects. *Language*, 86(2):366.

C-T James Huang. 1998. *Logical relations in Chinese and the theory of grammar*. Taylor & Francis.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv*, 1602.02410.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2016. What do recurrent neural network grammars learn about syntax? *arXiv preprint arXiv:1611.05774*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

William Ladusaw. 1979. *Polarity Sensitivity as Inherent Scope Relations*. Ph.D. thesis, University of Texas at Austin.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*.

Michael EJ Masson and Geoffrey R Loftus. 2003. Using confidence intervals for graphically based data interpretation. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 57(3):203.

Colin Phillips, Nina Kazanina, and Shani H Abada. 2005. Erp effects of the processing of syntactic long-distance dependencies. *Cognitive Brain Research*, 22(3):407–428.

John Robert Ross. 1967. Constraints on variables in syntax.

Nathaniel J Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.

Jon Sprouse and Norbert Hornstein. 2013. *Experimental syntax and island effects*. Cambridge University Press.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. *arXiv preprint arXiv:1707.08976*.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do rnn language models learn about filler-gap dependencies? *arXiv preprint arXiv:1809.00042*.

Appendix

We present results for two large data LSTM models on novel experiments described in the paper. The two models tested here are the 'BIG LSTM + CNN Inputs' from Jozefowicz et al. (2016)

(the ‘google’ model) and the highest-performing model presented in the supplementary materials of Gulordava et al. (2018) (the ‘Gulordava’ model). Both models where shown in (Wilcox et al., 2018) to represent filler—gap dependencies and some island constraints.

A Syntactic Hierarchy

We tested the two large LSTM models using our stimuli from the syntactic hierarchy experiment and measured the wh-licensing interaction across the entire embedded clause. The results of this experiment can be seen in Figure 6. Both models show significant licensing interaction in the grammatical *Subject* condition ($p < 0.001$), and a significant reduction in licensing interaction between the *Subject* and *Matrix* conditions ($p < 0.001$ in both models). Additionally, there is a significant licensing interaction in the *Matrix* condition for the Google model, but not so for the Gulordava model.

B Unboundedness

We tested the two large LSTM models from Wilcox et al. (2018) following the stimuli from our unboundedness experiment, with two variants, one that included gaps in *Object* position and one that included gaps in indirect object or *Goal* position. The results can be seen in Figure 7. For the Google model in *Object* position, we find a significant reduction of wh-licensing interaction across more than three layers of embedding ($p < 0.001$). For the Gulordava model, we find a significant reduction in wh-licensing interaction after only one layer of embedding ($p < 0.001$). In the *Goal* position: For the Google model, we find a significant reduction in licensing interaction after two layers of embedding ($p < 0.05$ for 2 layers, $p < 0.001$ for 3-4 layers). For the Gulordava model, we find no significant licensing interaction after one layer of embedding. These results indicate the larger LSTMs are able to thread gap expectation through embedded clauses.

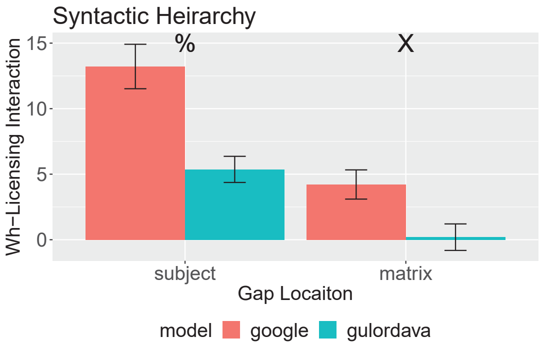


Figure 6: Syntactic Hierarchy. %s indicate conditions where we would expect a strong wh-licensing interaction, Xs where we expect low wh-licensing interaction.

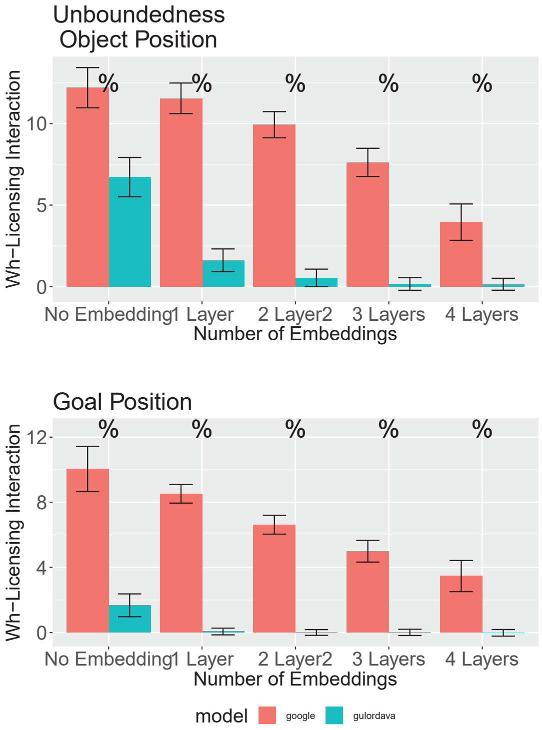


Figure 7: Unboundedness, %s indicate conditions where we expect a strong wh-licensing interaction.

Benchmarking Approximate Inference Methods for Neural Structured Prediction

Lifu Tu **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{lifu,kgimpel}@ttic.edu

Abstract

Exact structured inference with neural network scoring functions is computationally challenging but several methods have been proposed for approximating inference. One approach is to perform gradient descent with respect to the output structure directly (Belanger and McCallum, 2016). Another approach, proposed recently, is to train a neural network (an "inference network") to perform inference (Tu and Gimpel, 2018). In this paper, we compare these two families of inference methods on three sequence labeling datasets. We choose sequence labeling because it permits us to use exact inference as a benchmark in terms of speed, accuracy, and search error. Across datasets, we demonstrate that inference networks achieve a better speed/accuracy/search error trade-off than gradient descent, while also being faster than exact inference at similar accuracy levels. We find further benefit by combining inference networks and gradient descent, using the former to provide a warm start for the latter.[1]

1 Introduction

Structured prediction models commonly involve complex inference problems for which finding exact solutions is intractable (Cooper, 1990). There are generally two ways to address this difficulty. One is to restrict the model family to those for which inference is feasible. For example, state-of-the-art methods for sequence labeling use structured energies that decompose into label-pair potentials and then use rich neural network architectures to define the potentials (Collobert et al., 2011; Lample et al., 2016; Ma and Hovy, 2016, *inter alia*). Exact dynamic programming algorithms like the Viterbi algorithm can be used for inference.

The second approach is to retain computationally-intractable scoring functions but then use approximate methods for inference. For example, some researchers relax the structured output space from a discrete space to a continuous one and then use gradient descent to maximize the score function with respect to the output (Belanger and McCallum, 2016). Another approach is to train a neural network (an "inference network") to output a structure in the relaxed space that has high score under the structured scoring function (Tu and Gimpel, 2018). This idea was proposed as an alternative to gradient descent in the context of structured prediction energy networks (Belanger and McCallum, 2016).

In this paper, we empirically compare exact inference, gradient descent, and inference networks for three sequence labeling tasks. We train conditional random fields (CRFs) for sequence labeling with neural networks used to define the potentials. We choose a scoring function that permits exact inference via Viterbi so that we can benchmark the approximate methods in terms of search error in addition to speed and accuracy. We consider three families of neural network architectures to serve as inference networks: convolutional neural networks (CNNs), recurrent neural networks (RNNs), and sequence-to-sequence models with attention (seq2seq; Sutskever et al., 2014; Bahdanau et al., 2015). We also use multi-task learning while training inference networks, combining the structured scoring function with a local cross entropy loss.

Our empirical findings can be summarized as follows. Gradient descent works reasonably well for tasks with small label sets and primarily local structure, like part-of-speech tagging. However, gradient descent struggles on tasks with long-distance dependencies, even with small label set sizes. For tasks with large label set sizes, inference networks and Viterbi perform comparably,

---

[1] Code is available at `github.com/lifu-tu/BenchmarkingApproximateInference`

Proceedings of NAACL-HLT 2019, pages 3313–3324
Minneapolis, Minnesota, June 2 - June 7, 2019.

with Viterbi taking much longer. In this regime, it is difficult for gradient descent to find a good solution, even with many iterations.

In comparing inference network architectures, (1) CNNs are the best choice for tasks with primarily local structure, like part-of-speech tagging; (2) RNNs can handle longer-distance dependencies while still offering high decoding speeds; and (3) seq2seq networks consistently work better than RNNs, but are also the most computationally expensive.

We also compare search error between gradient descent and inference networks and measure correlations with input likelihood. We find that inference networks achieve lower search error on instances with higher likelihood (under a pretrained language model), while for gradient descent the correlation between search error and likelihood is closer to zero. This shows the impact of the use of dataset-based learning of inference networks, i.e., they are more effective at amortizing inference for more common inputs.

Finally, we experiment with two refinements of inference networks. The first fine-tunes the inference network parameters for a single test example to minimize the energy of its output. The second uses an inference network to provide a warm start for gradient descent. Both lead to reductions in search error and higher accuracies for certain tasks, with the warm start method leading to a better speed/accuracy trade-off.

2 Sequence Models

For sequence labeling tasks, given an input sequence $\boldsymbol{x} = \langle x_1, x_2, ..., x_{|\boldsymbol{x}|}\rangle$, we wish to output a sequence $\boldsymbol{y} = \langle \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{|\boldsymbol{x}|}\rangle \in \mathcal{Y}(\boldsymbol{x})$. Here $\mathcal{Y}(\boldsymbol{x})$ is the structured output space for $\boldsymbol{x}$. Each label $\mathbf{y}_t$ is represented as an L-dimensional one-hot vector where L is the number of labels.

Conditional random fields (CRFs; Lafferty et al., 2001) form one popular class of methods for structured prediction, especially for sequence labeling. We define our structured energy function to be similar to those often used in CRFs for sequence labeling:

$$E_\Theta(\boldsymbol{x}, \boldsymbol{y}) = \\ -\left(\sum_t \sum_{i=1}^{L} y_{t,i}\left(\mathbf{u}_i^\top f(\boldsymbol{x}, t)\right) + \sum_t \mathbf{y}_{t-1}^\top \mathbf{W} \mathbf{y}_t\right)$$

where $y_{t,i}$ is the ith entry of the vector $\mathbf{y}_t$. In the standard discrete-label setting, each $\mathbf{y}_t$ is a one-hot vector, but this energy is generalized to be able to use both discrete labels and continuous relaxations of the label space, which we will introduce below. Also, we use $f(\boldsymbol{x}, t) \in \mathbb{R}^d$ to denote the "input feature vector" for position t, $\mathbf{u}_i \in \mathbb{R}^d$ is a label-specific parameter vector used for modeling the local scoring function, and $\mathbf{W} \in \mathbb{R}^{L \times L}$ is a parameter matrix learned to model label transitions. For the feature vectors we use a bidirectional long short-term memory (BLSTM; Hochreiter and Schmidhuber, 1997), so this forms a BLSTM-CRF (Lample et al., 2016; Ma and Hovy, 2016).

For training, we use the standard conditional log-likelihood objective for CRFs, using the forward and backward dynamic programming algorithms to compute gradients. For a given input $\boldsymbol{x}$ at test time, prediction is done by choosing the output with the lowest energy:

$$\underset{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})}{\operatorname{argmin}}\, E_\Theta(\boldsymbol{x}, \boldsymbol{y})$$

The Viterbi algorithm can be used to solve this problem exactly for the energy defined above.

2.1 Modeling Improvements: BLSTM-CRF+

For our experimental comparison, we consider two CRF variants. The first is the basic model described above, which we refer to as BLSTM-CRF. Below we describe three additional techniques that we add to the basic model. We will refer to the CRF with these three techniques as BLSTM-CRF+. Using these two models permits us to assess the impact of model complexity and performance level on the inference method comparison.

Word Embedding Fine-Tuning. We used pretrained, fixed word embeddings when using the BLSTM-CRF model, but for the more complex BLSTM-CRF+ model, we fine-tune the pretrained word embeddings during training.

Character-Based Embeddings. Character-based word embeddings provide consistent improvements in sequence labeling (Lample et al., 2016; Ma and Hovy, 2016). In addition to pretrained word embeddings, we produce a character-based embedding for each word using a character convolutional network like that of Ma and Hovy (2016). The filter size is 3 characters and the character embedding dimensionality is 30. We use max pooling over the character sequence in the word and the resulting embedding is concatenated with the word embedding before being passed to the BLSTM.

Dropout. We also add dropout during training (Hinton et al., 2012). Dropout is applied before the character embeddings are fed into the CNNs, at the final word embedding layer before the input to the BLSTM, and after the BLSTM. The dropout rate is 0.5 for all experiments.

3 Gradient Descent for Inference

To use gradient descent (GD) for structured inference, researchers typically relax the output space from a discrete, combinatorial space to a continuous one and then use gradient descent to solve the following optimization problem:

$$\operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}_R(\boldsymbol{x})} E_\Theta(\boldsymbol{x}, \boldsymbol{y})$$

where $\mathcal{Y}_R$ is the relaxed continuous output space. For sequence labeling, $\mathcal{Y}_R(\boldsymbol{x})$ consists of length-$|\boldsymbol{x}|$ sequences of probability distributions over output labels. To obtain a discrete labeling for evaluation, the most probable label at each position is returned.

There are multiple settings in which gradient descent has been used for structured inference, e.g., image generation (Johnson et al., 2016), structured prediction energy networks (Belanger and McCallum, 2016), and machine translation (Hoang et al., 2017). Gradient descent has the advantage of simplicity. Standard autodifferentiation toolkits can be used to compute gradients of the energy with respect to the output once the output space has been relaxed. However, one challenge is maintaining constraints on the variables being optimized. Therefore, we actually perform gradient descent in an even more relaxed output space $\mathcal{Y}_{R'}(\boldsymbol{x})$ which consists of length-$|\boldsymbol{x}|$ sequences of vectors, where each vector $\mathbf{y}_t \in \mathbb{R}^L$. When computing the energy, we use a softmax transformation on each $\mathbf{y}_t$, solving the following optimization problem with gradient descent:

$$\operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}_{R'}(\boldsymbol{x})} E_\Theta(\boldsymbol{x}, \mathrm{softmax}(\boldsymbol{y})) \tag{1}$$

where the softmax operation above is applied independently to each vector $\mathbf{y}_t$ in the output structure $\boldsymbol{y}$.

4 Inference Networks

Tu and Gimpel (2018) define an **inference network** ("infnet") $\mathbf{A}_\Psi : \mathcal{X} \to \mathcal{Y}_R$ and train it with the goal that

$$\mathbf{A}_\Psi(\boldsymbol{x}) \approx \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}_R(\boldsymbol{x})} E_\Theta(\boldsymbol{x}, \boldsymbol{y})$$

where $\mathcal{Y}_R$ is the relaxed continuous output space as defined in Section 3. For sequence labeling, for example, an inference network $\mathbf{A}_\Psi$ takes a sequence $\boldsymbol{x}$ as input and outputs a distribution over labels for each position in $\boldsymbol{x}$. Below we will consider three families of neural network architectures for $\mathbf{A}_\Psi$.

For training the inference network parameters Ψ, Tu and Gimpel (2018) explored stabilization and regularization terms and found that a local cross entropy loss consistently worked well for sequence labeling. We use this local cross entropy loss in this paper, so we perform learning by solving the following:

$$\operatorname*{argmin}_{\Psi} \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} E_\Theta(\boldsymbol{x}, \mathbf{A}_\Psi(\boldsymbol{x})) + \lambda \ell_{\text{token}}(\boldsymbol{y}, \mathbf{A}_\Psi(\boldsymbol{x}))$$

where the sum is over $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ pairs in the training set. The token-level loss is defined:

$$\ell_{\text{token}}(\boldsymbol{y}, \mathbf{A}(\boldsymbol{x})) = \sum_{t=1}^{|\boldsymbol{y}|} \mathrm{CE}(\mathbf{y}_t, \mathbf{A}(\boldsymbol{x})_t) \tag{2}$$

where $\mathbf{y}_t$ is the L-dimensional one-hot label vector at position t in $\boldsymbol{y}$, $\mathbf{A}(\boldsymbol{x})_t$ is the inference network's output distribution at position t, and CE stands for cross entropy. We will give more details on how ℓ_{token} is defined for different inference network families below. It is also the loss used in our non-structured baseline models.

4.1 Inference Network Architectures

We now describe options for inference network architectures for sequence labeling. For each, we optionally include the modeling improvements described in Section 2.1. When doing so, we append "+" to the setting's name to indicate this (e.g., infnet+).

4.1.1 Convolutional Neural Networks

CNNs are frequently used in NLP to extract features based on symbol subsequences, whether words or characters (Collobert et al., 2011; Kalchbrenner et al., 2014; Kim, 2014;

Kim et al., 2016; Zhang et al., 2015). CNNs use filters that are applied to symbol sequences and are typically followed by some sort of pooling operation. We apply filters over a fixed-size window centered on the word being labeled and do not use pooling. The feature maps $f_n(\boldsymbol{x}, t)$ for $(2n+1)$-gram filters are defined:

$$f_n(\boldsymbol{x}, t) = g(\mathbf{W}_n[\mathbf{v}_{x_{t-n}}; ...; \mathbf{v}_{x_{t+n}}] + \mathbf{b}_n)$$

where g is a nonlinearity, $\mathbf{v}_{x_t}$ is the embedding of word x_t, and $\mathbf{W}_n$ and $\mathbf{b}_n$ are filter parameters. We consider two CNN configurations: one uses $n = 0$ and $n = 1$ and the other uses $n = 0$ and $n = 2$. For each, we concatenate the two feature maps and use them as input to the softmax layer over outputs. In each case, we use H filters for each feature map.

4.1.2 Recurrent Neural Networks

For sequence labeling, it is common to use a BLSTM that runs over the input sequence and produces a softmax distribution over labels at each position in the sequence. We use this "BLSTM tagger" as our RNN inference network architecture. The parameter H refers to the size of the hidden vectors in the forward and backward LSTMs, so the full dimensionality passed to the softmax layer is $2H$.

4.1.3 Sequence-to-Sequence Models

Sequence-to-sequence (seq2seq; Sutskever et al. 2014) models have been successfully used for many sequential modeling tasks. It is common to augment models with an attention mechanism that focuses on particular positions of the input sequence while generating the output sequence (Bahdanau et al., 2015). Since sequence labeling tasks have equal input and output sequence lengths and a strong connection between corresponding entries in the sequences, Goyal et al. (2018) used fixed attention that deterministically attends to the ith input when decoding the ith output, and hence does not learn any attention parameters. It is shown as follows:

$$P(\mathbf{y}_t \mid \mathbf{y}_{<t}, \boldsymbol{x}) = \text{softmax}(\mathbf{W}_s[\mathbf{h}_t, \mathbf{s}_t])$$

where $\mathbf{s}_t$ is the hidden vector at position t from a BLSTM run over $\boldsymbol{x}$, $\mathbf{h}_t$ is the decoder hidden vector at position t, and $\mathbf{W}_s$ is a parameter matrix. The concatenation of the two hidden vectors is used to produce the distribution over labels.

When using this inference network, we redefine the local loss to the standard training criterion for seq2seq models, namely the sum of the log losses for each output conditioned on the previous outputs in the sequence. We always use the previous predicted label as input (as used in "scheduled sampling," Bengio et al., 2015) during training because it works better for our tasks. In our experiments, the forward and backward encoder LSTMs use hidden dimension H, as does the LSTM decoder. Thus the model becomes similar to the BLSTM tagger except with conditioning on previous labeling decisions in a left-to-right manner.

We also experimented with the use of beam search for both the seq2seq baseline and inference networks and did not find much difference in the results. Also, as alternatives to the deterministic position-based attention described above, we experimented with learned local attention (Luong et al., 2015) and global attention, but they did not work better on our tasks.

4.2 Methods to Improve Inference Networks

To further improve the performance of an inference network for a particular test instance $\boldsymbol{x}$, we propose two novel approaches that leverage the strengths of inference networks to provide effective starting points and then use instance-level fine-tuning in two different ways.

4.2.1 Instance-Tailored Inference Networks

For each test example $\boldsymbol{x}$, we initialize an instance-specific inference network $\mathbf{A}_\Psi(\boldsymbol{x})$ using the trained inference network parameters, then run gradient descent on the following loss:

$$\underset{\Psi}{\operatorname{argmin}}\, E_\Theta(\boldsymbol{x}, \mathbf{A}_\Psi(\boldsymbol{x})) \tag{3}$$

This procedure fine-tunes the inference network parameters for a single test example to minimize the energy of its output. For each test example, the process is repeated, with a new instance-specific inference network being initialized from the trained inference network parameters.

4.2.2 Warm-Starting Gradient Descent with Inference Networks

Given a test example $\boldsymbol{x}$, we initialize $\boldsymbol{y} \in \mathcal{Y}_{R'}(\boldsymbol{x})$ using the inference network and then use gradient descent by solving Eq. 1 described in Section 3 to update $\boldsymbol{y}$. However, the inference network output is in $\mathcal{Y}_R(\boldsymbol{x})$ while gradient descent works with the more relaxed space $\mathcal{Y}_{R'}(\boldsymbol{x})$. So we simply use the logits from the inference network, which are the score vectors before the softmax operations.

5 Experimental Setup

We perform experiments on three tasks: Twitter part-of-speech tagging (POS), named entity recognition (NER), and CCG supersense tagging (CCG).

5.1 Datasets

POS. We use the annotated data from Gimpel et al. (2011) and Owoputi et al. (2013) which contains 25 POS tags. For training, we combine the 1000-tweet OCT27TRAIN set and the 327-tweet OCT27DEV set. For validation, we use the 500-tweet OCT27TEST set and for testing we use the 547-tweet DAILY547 test set. We use the 100-dimensional skip-gram embeddings from Tu et al. (2017) which were trained on a dataset of 56 million English tweets using `word2vec` (Mikolov et al., 2013). The evaluation metric is tagging accuracy.

NER. We use the CoNLL 2003 English data (Tjong Kim Sang and De Meulder, 2003). There are four entity types: PER, LOC, ORG, and MISC. There is a strong local dependency between neighboring labels because this is a labeled segmentation task. We use the BIOES tagging scheme, so there are 17 labels. We use 100-dimensional pretrained GloVe (Pennington et al., 2014) embeddings. The task is evaluated with micro-averaged F1 score using the `conlleval` script.

CCG. We use the standard splits from CCGbank (Hockenmaier and Steedman, 2002). We only keep sentences with length less than 50 in the original training data when training the CRF. The training data contains 1,284 unique labels, but because the label distribution has a long tail, we use only the 400 most frequent labels, replacing the others by a special tag $*$. The percentages of $*$ in train/development/test are 0.25/0.23/0.23%. When the gold standard tag is $*$, the prediction is always evaluated as incorrect. We use the same GloVe embeddings as in NER. Because of the compositional nature of supertags, this task has more non-local dependencies. The task is evaluated with per-token accuracy.

5.2 Training and Tuning

For the optimization problems mentioned below, we use stochastic gradient descent with momentum as the optimizer. Full details of hyperparameter tuning are in the appendix.

Local Baselines. We consider local (non-structured) baselines that use the same architectures as the inference networks but train using only the local loss ℓ_{token}.

Structured Baselines. We train the BLSTM-CRF and BLSTM-CRF+ models with the standard conditional log-likelihood objective. We tune hyperparameters on the development sets.

Gradient Descent for Inference. We use gradient descent for structured inference by solving Eq. 1. We randomly initialize $\boldsymbol{y} \in \mathcal{Y}_{R'}(\boldsymbol{x})$ and, for N iterations, we compute the gradient of the energy with respect to $\boldsymbol{y}$, then update $\boldsymbol{y}$ using gradient descent with momentum, which we found to generally work better than constant step size. We tune N and the learning rate via instance-specific oracle tuning, i.e., we choose them separately for each input to maximize performance (accuracy or F1 score) on that input. Even with this oracle tuning, we find that gradient descent struggles to compete with the other methods.

Inference Networks. To train the inference networks, we first train the BLSTM-CRF or BLSTM-CRF+ model with the standard conditional log-likelihood objective. The hidden sizes H are tuned in that step. We then fix the energy function and train the inference network $\mathbf{A}_{\Psi}$ using the combined loss from Section 4.

For instance-tailored inference networks and when using inference networks as a warm start for gradient descent, we tune the number of epochs N and the learning rate on the development set, and report the performance on the test set, using the same values of N and the learning rate for all test examples.

6 BLSTM-CRF Results

This first section of results uses the simpler BLSTM-CRF modeling configuration. In Section 7 below we present results with the stronger BLSTM-CRF+ configuration and also apply the same modeling improvements to the baselines and inference networks.

	Twitter POS Tagging			NER			CCG Supertagging		
	CNN	BLSTM	seq2seq	CNN	BLSTM	seq2seq	CNN	BLSTM	seq2seq
local baseline	89.6	88.0	88.9	79.9	85.0	85.3	90.6	92.2	92.7
infnet	**89.9**	89.5	89.7	82.2	85.4	86.1	91.3	92.8	**92.9**
gradient descent	89.1			84.4			89.0		
Viterbi	89.2			**87.2**			92.4		

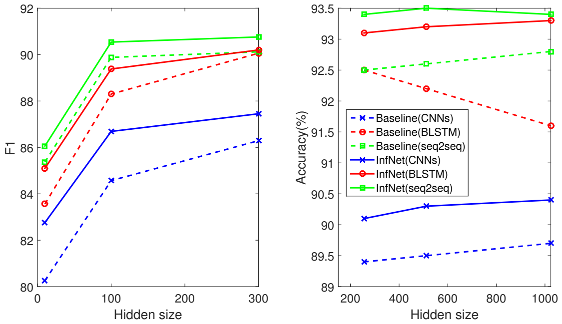
Table 1: Test results for all tasks. Inference networks, gradient descent, and Viterbi are all optimizing the BLSTM-CRF energy. Best result per task is in bold.

Table 1 shows test results for all tasks and architectures. The inference networks use the same architectures as the corresponding local baselines, but their parameters are trained with both the local loss and the BLSTM-CRF energy, leading to consistent improvements. CNN inference networks work well for POS, but struggle on NER and CCG compared to other architectures. BLSTMs work well, but are outperformed slightly by seq2seq models across all three tasks. Using the Viterbi algorithm for exact inference yields the best performance for NER but is not best for the other two tasks.

It may be surprising that an inference network trained to mimic Viterbi would outperform Viterbi in terms of accuracy, which we find for the CNN for POS tagging and the seq2seq inference network for CCG. We suspect this occurs for two reasons. One is due to the addition of the local loss in the inference network objective; the inference networks may be benefiting from this multi-task training. Edunov et al. (2018) similarly found benefit from a combination of token-level and sequence-level losses. The other potential reason is beneficial inductive bias with the inference network architecture. For POS tagging, the CNN architecture is clearly well-suited to this task given the strong performance of the local CNN baseline. Nonetheless, the CNN inference network is able to improve upon both the CNN baseline and Viterbi.

Hidden Sizes. For the test results in Table 1, we did limited tuning of H for the inference networks based on the development sets. Figure 1 shows the impact of H on performance. Across H values, the inference networks outperform the baselines. For NER and CCG, seq2seq outperforms the BLSTM which in turn outperforms the CNN.

Tasks and Window Sizes. Table 2 shows that CNNs with smaller windows are better for POS, while larger windows are better for NER and CCG. This suggests that POS has more local dependencies among labels than NER and CCG.



(a) NER (b) CCG Supertagging

Figure 1: Development results for inference networks with different architectures and hidden sizes (H).

		{1,3}-gram	{1,5}-gram
POS	local baseline	89.2	88.7
	infnet	**89.6**	89.0
NER	local baseline	84.6	85.4
	infnet	86.7	**86.8**
CCG	local baseline	89.5	90.4
	infnet	90.3	**91.4**

Table 2: Development results for CNNs with two filter sets ($H = 100$).

6.1 Speed Comparison

Asymptotically, Viterbi takes $\mathcal{O}(nL^2)$ time, where n is the sequence length. The BLSTM and our deterministic-attention seq2seq models have time complexity $\mathcal{O}(nL)$. CNNs also have complexity $\mathcal{O}(nL)$ but are more easily parallelizable. Table 3 shows test-time inference speeds for inference networks, gradient descent, and Viterbi for the BLSTM-CRF model. We use GPUs and a minibatch size of 10 for all methods. CNNs are 1-2 orders of magnitude faster than the others. BLSTMs work almost as well as seq2seq models and are 2-4 times faster in our experiments. Viterbi is actually faster than seq2seq when L is small, but for CCG, which has $L = 400$, it is 4-5 times slower. Gradient descent is slower than the others because it generally needs many iterations (20-50) for competitive performance.

	CNN	BLSTM	seq2seq	Viterbi	GD
POS	12500	1250	357	500	20
NER	10000	1000	294	360	23
CCG	6666	1923	1000	232	16

Table 3: Speed comparison of inference networks across tasks and architectures (examples/sec).

6.2 Search Error

We can view inference networks as approximate search algorithms and assess characteristics that affect search error. To do so, we train two LSTM language models (one on word sequences and one on gold label sequences) on the Twitter POS data. We also compute the difference in the BLSTM-CRF energies between the inference network output $\boldsymbol{y}_{inf}$ and the Viterbi output $\boldsymbol{y}_{vit}$ as the search error: $E_{\Theta}(\boldsymbol{x}, \boldsymbol{y}_{inf}) - E_{\Theta}(\boldsymbol{x}, \boldsymbol{y}_{vit})$. We compute the same search error for gradient descent.

For the BLSTM inference network, Spearman's ρ between the word sequence perplexity and search error is 0.282; for the label sequence perplexity, it is 0.195. For gradient descent inference, Spearman's ρ between the word sequence perplexity and search error is 0.122; for the label sequence perplexity, it is 0.064. These positive correlations mean that for frequent sequences, inference networks and gradient descent exhibit less search error. We also note that the correlations are higher for the inference network than for gradient descent, showing the impact of amortization during learning of the inference network parameters. That is, since we are learning to do inference from a dataset, we would expect search error to be smaller for more frequent sequences, and we do indeed see this correlation.

7 BLSTM-CRF+ Results

We now compare inference methods when using the improved modeling techniques described in Section 2.1 (i.e., the setting we called BLSTM-CRF+). We use these improved techniques for all models, including the CRF, the local baselines, gradient descent, and the inference networks. When training inference networks, both the inference network architectures and the structured energies use the techniques from Section 2.1. So, when referring to inference networks in this section, we use the name infnet+.

The results are shown in Table 4. With a more powerful local architecture, structured prediction is less helpful overall, but inference networks still improve over the local baselines on 2 of 3 tasks.

	POS	NER	CCG
local baseline	91.3	90.5	94.1
infnet+	91.3	90.8	94.2
gradient descent	90.8	89.8	90.4
Viterbi	90.9	91.6	94.3

Table 4: Test results with BLSTM-CRF+. For local baseline and inference network architectures, we use CNN for POS, seq2seq for NER, and BLSTM for CCG.

	F1
local baseline (BLSTM)	90.3
infnet+ (1-layer BLSTM)	90.7
infnet+ (2-layer BLSTM)	91.1
Viterbi	91.6

Table 5: NER test results (for BLSTM-CRF+) with more layers in the BLSTM inference network.

POS. As in the BLSTM-CRF setting, the local CNN baseline and the CNN inference network outperform Viterbi. This is likely because the CRFs use BLSTMs as feature networks, but our results show that CNN baselines are consistently better than BLSTM baselines on this task. As in the BLSTM-CRF setting, gradient descent works quite well on this task, comparable to Viterbi, though it is still much slower.

NER. We see slightly higher BLSTM-CRF+ results than several previous state-of-the-art results (cf. 90.94; Lample et al., 2016 and 91.37; Ma and Hovy, 2016). The stronger BLSTM-CRF+ configuration also helps the inference networks, improving performance from 90.5 to 90.8 for the seq2seq architecture over the local baseline. Though gradient descent reached high accuracies for POS tagging, it does not perform well on NER, possibly due to the greater amount of non-local information in the task.

While we see strong performance with infnet+, it still lags behind Viterbi in F1. We consider additional experiments in which we increase the number of layers in the inference networks. We use a 2-layer BLSTM as the inference network and also use weight annealing of the local loss hyperparameter λ, setting it to $\lambda = e^{-0.01t}$ where t is the epoch number. Without this annealing, the 2-layer inference network was difficult to train. The weight annealing was helpful for encouraging the inference network to focus more on the non-local information in the energy function rather than the token-level loss. As shown in Table 5, these changes yield an improvement of 0.4 in F1.

		Twitter POS Tagging		NER		CCG Supertagging	
	N	Acc. (↑)	Energy (↓)	F1 (↑)	Energy (↓)	Acc. (↑)	Energy (↓)
gold standard		100	-159.65	100	-230.63	100	-480.07
BLSTM-CRF+/Viterbi		90.9	-163.20	91.6	-231.53	94.3	-483.09
gradient descent	10	89.2	-161.69	81.9	-227.92	65.1	-412.81
	20	90.8	-163.06	89.1	-231.17	74.6	-414.81
	30	90.8	-163.02	89.6	-231.30	83.0	-447.64
	40	90.7	-163.03	89.8	-231.34	88.6	-471.52
	50	90.8	-163.04	89.8	-231.35	90.0	-476.56
	100	-	-	-	-	90.1	-476.98
	500	-	-	-	-	90.1	-476.99
	1000	-	-	-	-	90.1	-476.99
infnet+		91.3	-162.07	90.8	-231.19	94.2	-481.32
discretized output from infnet+		91.3	-160.87	90.8	-231.34	94.2	-481.95
instance-tailored infnet+	3	91.0	-162.59	91.3	-231.32	94.3	-481.91
	5	90.9	-162.81	91.2	-231.37	94.3	-482.23
	10	91.3	-162.85	91.5	-231.39	94.3	-482.56
infnet+ as warm start for gradient descent	3	91.4	-163.06	91.4	-231.42	94.4	-482.62
	5	91.2	-163.12	91.4	-231.45	94.4	-482.64
	10	91.2	-163.15	91.5	-231.46	94.4	-482.78

Table 6: Test set results of approximate inference methods for three tasks, showing performance metrics (accuracy and F1) as well as average energy of the output of each method. The inference network architectures in the above experiments are: CNN for POS, seq2seq for NER, and BLSTM for CCG. N is the number of epochs for GD inference or instance-tailored fine-tuning.

CCG. Our BLSTM-CRF+ reaches an accuracy of 94.3%, which is comparable to several recent results (93.53, Xu et al., 2016; 94.3, Lewis et al., 2016; and 94.50, Vaswani et al., 2016). The local baseline, the BLSTM inference network, and Viterbi are all extremely close in accuracy. Gradient descent struggles here, likely due to the large number of candidate output labels.

7.1 Speed, Accuracy, and Search Error

Table 6 compares inference methods in terms of both accuracy and energies reached during inference. For each number N of gradient descent iterations in the table, we tune the learning rate per-sentence and report the average accuracy/F1 with that fixed number of iterations. We also report the average energy reached. For inference networks, we report energies both for the output directly and when we discretize the output (i.e., choose the most probable label at each position).

Gradient Descent Across Tasks. The number of gradient descent iterations required for competitive performance varies by task. For POS, 20 iterations are sufficient to reach accuracy and energy close to Viterbi. For NER, roughly 40 iterations are needed for gradient descent to reach its highest F1 score, and for its energy to become very close to that of the Viterbi outputs. However, its F1 score is much lower than Viterbi. For CCG, gradient descent requires far more iterations, presumably due to the larger number of labels in the task. Even with 1000 iterations, the accuracy is 4% lower than Viterbi and the inference networks. Unlike POS and NER, the inference network reaches much lower energies than gradient descent on CCG, suggesting that the inference network may not suffer from the same challenges of searching high-dimensional label spaces as those faced by gradient descent.

Inference Networks Across Tasks. For POS, the inference network does not have lower energy than gradient descent with ≥ 20 iterations, but it does have higher accuracy. This may be due in part to our use of multi-task learning for inference networks. The discretization of the inference network outputs increases the energy on average for this task, whereas it decreases the energy for the other two tasks. For NER, the inference network reaches a similar energy as gradient descent, especially when discretizing the output, but is considerably better in F1. The CCG tasks shows the largest difference between gradient descent and the inference network, as the latter is much better in both accuracy and energy.

Instance Tailoring and Warm Starting. Across tasks, instance tailoring and warm starting lead to lower energies than infnet+. The improvements in energy are sometimes joined by improvements in accuracy, notably for NER where the gains range from 0.4 to 0.7 in F1.

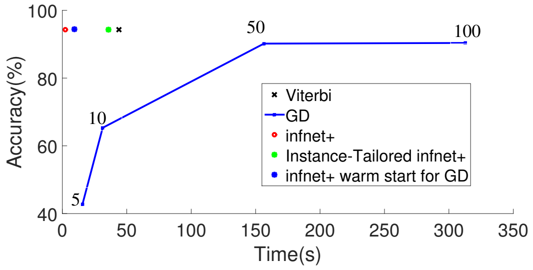


Figure 2: CCG test results for inference methods (GD = gradient descent). The x-axis is the total inference time for the test set. The numbers on the GD curve are the number of gradient descent iterations.

Warm starting gradient descent yields the lowest energies (other than Viterbi), showing promise for the use of gradient descent as a local search method starting from inference network output.

Wall Clock Time Comparison. Figure 2 shows the speed/accuracy trade-off for the inference methods, using wall clock time for test set inference as the speed metric. On this task, Viterbi is time-consuming because of the larger label set size. The inference network has comparable accuracy to Viterbi but is much faster. Gradient descent needs much more time to get close to the others but plateaus before actually reaching similar accuracy. Instance-tailoring and warm starting reside between infnet+ and Viterbi, with warm starting being significantly faster because it does not require updating inference network parameters.

8 Related Work

The most closely related prior work is that of Tu and Gimpel (2018), who experimented with RNN inference networks for sequence labeling. We compared three architectural families, showed the relationship between optimal architectures and downstream tasks, compared inference networks to gradient descent, and proposed novel variations.

We focused in this paper on sequence labeling, in which CRFs with neural network potentials have emerged as a state-of-the-art approach (Lample et al., 2016; Ma and Hovy, 2016; Strubell et al., 2017; Yang et al., 2018). Our results suggest that inference networks can provide a feasible way to speed up test-time inference over Viterbi without much loss in performance. The benefits of inference networks may be coming in part from multi-task training; Edunov et al. (2018) similarly found benefit from combining token-level and sequence-level losses.

We focused on structured prediction in this paper, but inference networks are useful in other settings as well. For example, it is common to use a particular type of inference network to approximate posterior inference in neural approaches to latent-variable probabilistic modeling, such as variational autoencoders (Kingma and Welling, 2013) and, more closely related to this paper, variational sequential labelers (Chen et al., 2018). In such settings, Kim et al. (2018) have found benefit with instance-specific updating of inference network parameters, which is related to our instance-level fine-tuning. There are also connections between structured inference networks and amortized structured inference (Srikumar et al., 2012) as well as methods for neural knowledge distillation and model compression (Hinton et al., 2015; Ba and Caruana, 2014; Kim and Rush, 2016).

Gradient descent is used for inference in several settings, e.g., structured prediction energy networks (Belanger and McCallum, 2016), image generation applications (Mordvintsev et al., 2015; Gatys et al., 2015), finding adversarial examples (Goodfellow et al., 2015), learning paragraph embeddings (Le and Mikolov, 2014), and machine translation (Hoang et al., 2017). Gradient descent has started to be replaced by inference networks in some of these settings, such as image transformation (Johnson et al., 2016; Li and Wand, 2016). Our results provide more evidence that gradient descent can be replaced by inference networks or improved through combination with them.

9 Conclusion

We compared several methods for approximate inference in neural structured prediction, finding that inference networks achieve a better speed/accuracy/search error trade-off than gradient descent. We also proposed instance-level inference network fine-tuning and using inference networks to initialize gradient descent, finding further reductions in search error and improvements in performance metrics for certain tasks.

Acknowledgments

We would like to thank Ke Li for suggesting experiments that combine inference networks and gradient descent, the anonymous reviewers for their feedback, and NVIDIA for donating GPUs used in this research.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations (ICLR)*.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on Machine Learning - Volume 48*, ICML'16, pages 983–992.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1171–1179.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 215–226.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12.

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artif. Intell.*, 42(2-3).

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *CoRR*, abs/1508.06576.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47, Portland, Oregon, USA.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Kartik Goyal, Graham Neubig, Chris Dyer, and Taylor Berg-Kirkpatrick. 2018. A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, Louisiana.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. Towards decoding as continuous optimisation in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156, Copenhagen, Denmark.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision (ECCV)*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the*

2016 Conference on Empirical Methods in Natural Language Processing, pages 1317–1327, Austin, Texas.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2678–2687, Stockholmsmssan, Stockholm Sweden.

Diederik Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California.

Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 702–716.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. DeepDream-a code example for visualizing neural networks. *Google Research*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Vivek Srikumar, Gourab Kundu, and Dan Roth. 2012. On amortizing inference cost for structured prediction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1114–1124, Jeju Island, Korea.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 265–275, Vancouver, Canada.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 232–237, San Diego, California.

Wenduan Xu, Michael Auli, and Stephen Clark. 2016. Expected F-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 210–220, San Diego, California.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657.

A Appendix

Local Baselines. We consider local (non-structured) baselines that use the same architectures as the inference networks but train using only the local loss ℓ_{token}. We tune the learning rate ($\{5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$). We train on the training set, use the development sets for tuning and early stopping, and report results on the test sets.

Structured Baselines. We train the BLSTM-CRF and BLSTM-CRF+ models with the standard conditional log-likelihood objective. We tune hyperparameters on the development sets. The tuned BLSTM hidden size H for BLSTM-CRF is 100 for POS/NER and 512 for CCG; for BLSTM-CRF+ the tuned hidden size is 100 for POS, 200 for NER, and 400 for CCG.

Gradient Descent for Inference. For the number of epochs N, we consider values in the set $\{5, 10, 20, 30, 40, 50, 100, 500, 1000\}$. For each N, we tune the learning rate over the set $\{1\text{e}^4, 5\text{e}^3, 1\text{e}^3, 500, 100, 50, 10, 5, 1\}$). These learning rates may appear extremely large when we are accustomed to choosing rates for empirical risk minimization, but we generally found that the most effective learning rates for structured inference are orders of magnitude larger than those effective for learning. To provide as strong performance as possible for the gradient descent method, we tune N and the learning rate via oracle tuning, i.e., we choose them separately for each input to maximize performance (accuracy or F1 score) on that input.

Inference Networks. To train the inference networks, we first train the BLSTM-CRF or BLSTM-CRF+ model with the standard conditional log-likelihood objective. The hidden sizes H are tuned in that step. We then fix the energy function and train the inference network $\mathbf{A}_\Psi$ using the combined loss from Section 4. We tune the learning rate over the set $\{5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$ for the inference network and the local loss weight λ over the set $\{0.2, 0.5, 1, 2, 5\}$. We use early stopping on the development sets and report the results on the test sets using the trained inference networks.

Evaluating and Enhancing the Robustness of Dialogue Systems: A Case Study on a Negotiation Agent

Minhao Cheng
University of California Los Angeles
mhcheng@cs.ucla.edu

Wei Wei
Google Cloud AI
wewei@google.com

Cho-Jui Hsieh
University of California Los Angeles
chohsieh@cs.ucla.edu

Abstract

Recent research has demonstrated that goal-oriented dialogue agents trained on large datasets can achieve striking performance when interacting with human users. In real world applications, however, it is important to ensure that the agent performs smoothly interacting with not only regular users but also those malicious ones who would attack the system through interactions in order to achieve goals for their own advantage. In this paper, we develop algorithms to evaluate the robustness of a dialogue agent by carefully designed attacks using adversarial agents. Those attacks are performed in both black-box and white-box settings. Furthermore, we demonstrate that adversarial training using our attacks can significantly improve the robustness of a goal-oriented dialogue system. On a case-study of the negotiation agent developed by (Lewis et al., 2017), our attacks reduced the average advantage of rewards between the attacker and the trained RL-based agent from 2.68 to -5.76 on a scale from -10 to 10 for randomized goals. Moreover, with the proposed adversarial training, we are able to improve the robustness of negotiation agents by 1.5 points on average against all our attacks.

1 Introduction

Crafting an intelligent agent to communicate in the dialogue system using natural languages has been a long-standing problem in AI. It requires designing an agent to understand, plan and generate natural language to achieve different goals such as question-answering, cooperation, negotiation etc (Vinyals and Le, 2015; Li et al., 2017; Serban et al., 2016; Dhingra et al., 2016; Serban et al., 2016). Inspired by recent successes in deep neural networks, (Lewis et al., 2017) has recently developed an end-to-end learning framework to train a recurrent neural network (RNN)-based negotiation agent in goal-oriented dialogue systems. This NN-based technique has been identified as one of the state-of-the-arts and has been applied to several other tasks (Bahdanau et al., 2014; Luong et al., 2015; Rush et al., 2015; Chan et al., 2016).

Although NN-based dialogue agents have shown convincing performance on several tasks, it is not clear whether they also work well when facing malicious users or agents. To answer this question, we study how to evaluate the robustness of a goal-oriented dialogue system. For simplicity, we consider a goal-oriented agent A that aims to maximize some score, and define the "robustness" of A as the worst-case performance under any feasible agent A'. We also call A' an **adversarial agent** that tries to "attack" A since it aims to minimize A's score. The problem of evaluating the robustness of A can then be solved by designing an adversarial agent to attack A. For instance, considering a negotiation agent that can decide when to make a deal, we say the agent is not robust if an adversarial agent can fool the target agent to make a deal with significant lower scores. Ideally, before deploying an agent into real systems, we need to ensure it performs smoothly under strong adversarial attacks.

The concept of adversarial agent is related to recent studies on adversarial examples for image classifiers—it has been shown that a carefully designed small perturbation can easily make neural networks mis-classify (Goodfellow et al., 2014; Szegedy et al., 2013; Moosavi Dezfooli et al., 2016; Carlini and Wagner, 2017; Cheng et al., 2019), and several recent works has extended these attacks to natural language processing models such as sentiment analysis (Gao et al., 2018; Yang et al., 2018) and machine translation (Ebrahimi et al., 2018; Cheng et al., 2018). However, all of the previous work consider attacking a static model, where except input im-

Proceedings of NAACL-HLT 2019, pages 3325–3335
Minneapolis, Minnesota, June 2 - June 7, 2019.

age/sentence there is no interaction between the attacker and the target model. Instead, we investigate a much more challenging problem, where there can be many turns of interactions between adversarial and target agents. This leads to several difficulties including 1) How to lead the target agent to a bad state and 2) how to force the target agent to make a wrong decision. Therefore, previous methods for attacking static models cannot be directly applied.

In this paper, we tackle the aforementioned challenges by proposing several novel ways to design an adversarial agent to evaluate the robustness of goal-oriented dialogue systems. We highlight our major contributions as follows:

- We propose a framework to generate adversarial agents in both black-box and white-box settings. To the best of our knowledge, this is the first work on crafting adversarial agents instead of adversarial examples in an interactive dialogue system.

- We conduct a series of studies on the negotiation agent proposed in (Lewis et al., 2017). We demonstrate that the proposed strategies can successfully attack existing negotiation agents to significantly reduce their average score. For instance, our attacks can reduce the average advantage of the RL-based negotiation agent from 2.68 to -5.76 on random problems with the total value of 10.

- We also show that through the proposed iterative adversarial training procedure, we could significantly improve the robustness of a goal-oriented agent against various attacks.

2 Related work

2.1 Goal-oriented dialogue agent

Goal-oriented dialogue systems aim at building a conversation model that is capable of accomplishing tasks through the interactions with human using natural language (Li et al., 2017; Eric and Manning, 2017; Wen et al., 2016; Wei et al., 2018; Bordes et al., 2016). Traditional approaches to learn a goal-oriented intelligent agent relies heavily on dialogue states annotated in the training data (Wen et al., 2016; Henderson et al., 2014). The use of state annotations allows a cleaner separation of the reasoning and natural language aspects of dialogues. However, it is very expensive to annotate every state in a large amount of training data. (Bordes et al., 2016) explores end-to-end goal orientated dialogue with a supervised model. And (He et al., 2017) uses task-specific rules to combine the task input and dialogue history into a more structured state representation. Recently, reinforcement learning has been widely used in dialogue systems to increase the agent versatility (Mordatch and Abbeel, 2017) and improve the agent's performance in goal-oriented tasks such as cooperative bot-bot dialogues (Das et al., 2017) and negotiation tasks (Lewis et al., 2017).

2.2 Adversarial examples in NLP applications

Algorithms have been proposed to craft adversarial sentences in NLP applications. (Papernot et al., 2016) uses Fast Gradient Sign method to generate adversarial example on RNN/LSTM based model. (Li et al., 2016) learns the importance of words by deleting them in sentiment analysis task and then use reinforcement learning to locate such words. (Samanta and Mehta, 2017) and (Liang et al., 2017) generate adversarial sequences by inserting or replacing existing words with typos and synonyms. (Gao et al., 2018) aims to attack sentiment classification models in a black-box setting. It develops some scoring functions to find the most important words to modify. (Jia and Liang, 2017) aims to fool the SQuAD reading comprehension system by adding crafted sentences. (Yang et al., 2018) proposes a greedy algorithm to swap the word/character and uses a Gumbel softmax function to reduce the computation. (Ebrahimi et al., 2018) aims to generate adversarial examples on character CNN model in machine translation problem by using Jacobian matrix to determine which word/character should be replaced or deleted. (Zhao et al., 2017) generated natural adversarial example using Generative Adversarial Networks (GANs). (Cheng et al., 2018) proposed a framework to conduct non-overlapping and targeted keyword attack on seq2seq model.

All the above-mentioned work focus on the static setting, i.e., the input does not depend on the model's output. However, in our work, one agent's input depends on the other agent's output, which makes the input undecidable in the beginning. Therefore, an adversarial sentence or example is not enough to conduct attack in dialogue systems. Instead, for the first time, we propose novel ways to construct a adversarial agent, which can bait the target agent to step to a wrong state and make a bad decision.

2.3 Defense against adversarial examples

Many defense algorithms have been proposed recently to enhance the robustness of classification models. Among them, adversarial training (Madry et al., 2018; Goodfellow et al., 2014) has become one of the most successful methods, which uses both clean and adversarial examples to train a robust model. (Wong and Kolter, 2018) proposed another kind of adversarial training to improve the verification lower bound of neural networks; (Liu and Hsieh, 2019) combines the idea of generative adversarial network (GAN) and adversarial training to further boost the robustness on test images. Another promising way to enhance robustness is by adding randomness to the model. (Liu et al., 2018) shows adding randomness to both input and intermediate layers of neural networks can improve robustness; (Liu et al., 2019; Ye and Zhu, 2018) show that combining Bayesian neural network (with randomized weights) and adversarial training can achieve state-of-the-art adversarial error under attacks. However, all the existing defense methods only work for static models (usually for classification tasks). In this paper, we propose an adversarial training algorithm for an agent using RL with an "adversarial agent". This is to our knowledge the first algorithm for improving the robustness of an agent.

3 Competitive Negotiation Dialogues

We use the negotiation agent developed in (Lewis et al., 2017) as the running example in this paper. Note that our algorithm can be generalized to other goal-oriented dialogue systems by designing a different scoring function according to the task.

In a competitive negotiation dialogue setting, two agents are negotiating with each other over a set of items. We adopt the same setting as (Lewis et al., 2017), in which case items can be categorized into either a ball, a hat or a book. Each agent is given the goal of the conversation (denoted by g), which contains the initial values and the quantities of each of the three items. Agents then negotiate to maximize the total value of their possessed items. Agents are allowed to negotiate up to a maximum of 10 turns. Scores will be granted to agents based on the total value of the items if they reach an agreement. If they choose not to agree, 0 score will be granted to both agents. A competitive negotiation dialogue example played by human and agent could be found in Table 1.

Input		
Human	3x**book** value 2 3x**hat** value 1 1x**ball** value 1	
Agent	3x**book** value 1 3x**hat** value 2 1x**ball** value 1	
Human	I'd like the books and the hats.	
Agent	I will take the books and hats.	
Human	I need the books and at least one other object.	
Agent	Ok, you can have one book and one ball	
Human	If I can have all books, I can leave you the rest.	
Agent	Deal	
Human	$\langle selection \rangle$	
Output		**Reward**
Human	3x**hat** 1x**ball**	7/10
Agent	3x**book**	6/10

Table 1: Competitive negotiation dialogue generated between agent and human.

We assess the robustness of a trained end-to-end negotiation agent used in (Lewis et al., 2017). In the negotiation chatbot setting, agents first chat using natural language and then make a selection based on what they have chatted with. We refer to the first phase as **negotiation phase** and the second phase as **decision phase**.

In the negotiation phase, conversation response at time t, x_t is generated word by word based on chat history $x_{0..t-1}$ and the goal of the conversation g. The conversation model is controlled by a speaking module θ and tokens are randomly sampled from probability distribution p_θ. This process continues recursively until an end-of-sentence token $\langle EOS \rangle$ or selection token $\langle selection \rangle$ token is generated. When $\langle EOS \rangle$ is encountered, the turn terminates and the conversation is handled to another agent. When $\langle selection \rangle$ is encountered, the negotiation phase terminates and the negotiation will reach the decision phase.

$$x_t \sim p_\theta(x_t | x_{0...t-1}, g) \tag{1}$$

In the decision phase, both agents will output a decision o based on a decision module probability distribution p'_θ. Agents' decisions will be based on conversation history $x_{0...T}$ up to the current time step T and the goal of the conversation g. Here O is a set of all legitimate selections, which is defined to be a space of where each selection must be greater or equal than 0 and the sum of selections for the same item must be equal to its original quantity. Since we only have a few items, it is possible to enumerate all the possibilities to get the set O.

$$o^* = \arg\max_{o \in O} \prod_i p'_\theta(o_i | x_{0...T}, g) \qquad (2)$$

Agents will then collect rewards (i.e. scores) from the environment (which will be 0 if they output conflicted decisions, e.g. the total number of items are different from the initial amount). It is important to keep the agent producing sentences that are correct both grammatically and semantically and keeping them competitive at the same time. Therefore, a common strategy is to train agents using supervised learning to learn natural language and to use reinforcement learning to optimize models' performance using on goal-oriented learning. We measure two statistics **score** and **agreement**. **score** is the average score for each agent (0-10). **agreement** is the percentage of dialogues where both agents agreed on the same decision. To measure the extent of success of our adversarial agent, we use **advantage** which is easy to compute directly from adversarial agent score minus target agent score, i.e. $S_{adv} - S_{ori}$.

4 Proposed Black-box Attack Algorithms

We first build our adversarial agent in black-box setting. Black-box setting in goal-oriented dialogue system is defined where the target agent is unknown to the attacker, but it is possible to make queries to obtain the final decision made by the target agent. To be noted, our aim is to test the robustness of the target agent. Therefore, in the decision phase we let adversarial agent chooses the complementary of target agent's choice, so those two agents will always reach agreement. The adversarial agent thus only has the speaking module and there is no decision network needed. In this section we proposed two adversarial agents in the black-box setting.

4.1 Reinforcement learning attack

Inspired by the procedure of goal-based reinforcement learning, we modified the reward function of our adversarial agent with the advantage instead of the score he got:

$$r^{adv} = S_{adv} - S_{ori} \qquad (3)$$

where S_{adv} and S_{ori} are adversarial agent score and target agent score respectively. After a complete dialogue has been generated, we update adversarial agent's parameters based on the outcome of the negotiation.

To learn the adversarial agent's speaking network by reinforcement learning, we denote the subset of tokens generated by the adversarial agent as X^{adv}. In the completed dialogue, γ is the discount factor that rewards actions at the end of the dialogue more strongly, and μ is a running average of completed dialogue rewards so far. We define the future reward R for an action $x_t \in X^{adv}$ as follows:

$$R(x_t) = \sum_{x_t \in X^{adv}} \gamma^{T-t}(r^{adv} - \mu). \qquad (4)$$

Then by a standard policy gradient algorithm, we could train our adversarial agent. Note that this attack doesn't require the knowledge on the target agent's structure/weights, and the experimental results demonstrate significant attack performance over regular agents.

4.2 Transfer attack

Transfer attack is a popular idea for attacking black-box models (Papernot et al., 2017). In dialogue systems, we can also consider the following transfer process: a sentence that leads to low r^{adv} in one dialogue might also lead to similar results in another dialogue. To implement this idea, we first collect a list of last sentences spoken by the adversarial agent from dialogues with high reward, denoted by L. In the conversations, we let our adversarial agent and the target agent negotiate n turns using the regular speaking module, and then plug in one sentence in L at the $(n + 1)$-th turn. Our experimental results show that this transfer attack does not work well in practice.

5 Proposed White-box Attack Algorithms

In the white-box setting, we assume that the attacker can access every part of the target agent, including the weights of both speaking and decision models, and the decision output in every dialogue. Similar to the black-box attacks, we let the adversarial agent choose the complementary of target agent's choices to ensure 100% agreement. By exploiting the knowledge of the target agent's model, white-box attacks can achieve much higher advantage than black-box attacks.

5.1 Force target agent to select at a fixed turn

To begin with, we consider a simplified strategy where we first let our adversarial agent and the target agent negotiate n turns using regular speaking module. For the $(n+1)$-th turn, we propose the following two ways to modify the output of regular speaking module to maximize the rewards of adversarial agent.

5.1.1 Reactive attack

The first strategy is that the adversarial agent produces a sentence that forces the target agent to say $\langle selection \rangle$. The conversation will then enter the decision phase. At the same time, the sentence produced by the adversarial agent should guide the target agent to make a bad selection that would be in favor of the adversarial agent. We call this method **reactive attack**.

We formulate this strategy as an optimization problem. Let $\hat{\mathbf{x}} = x_{t_n \ldots T-1}$ be the output sentence generated by adversarial agent in the speaking model after n -th turn. Specifically, we define $x_{0 \ldots T-1}$ as all the tokens in the dialogue history before $\langle selection \rangle$. $Z_r(x_{0 \ldots T-1})$ indicates the logit layer outputs for predicting x_T based on chat history $x_{0 \ldots T-1}$ in the speaking model. $Z_o(x_{0 \ldots T})$ indicates the logit layer outputs on conversation history $x_{0 \ldots T}$ in the decision model. Because we have a constraint to force the target agent to say the end-of-dialog token $\langle selection \rangle$, we could format this constraint as

$$[Z_r(x_{0 \ldots T-1})]_{k_{sel}} - \max_{i \neq k_{sel}} [Z_r(x_{0 \ldots T-1})]_i \geq 0 \quad (5)$$

where k_{sel} is the corresponding index of end-of-dialog token $\langle selection \rangle$.

At the same time, the score of output o should be in favor of our adversarial agent. Assume the original decision output is o',

$$L(\hat{\mathbf{x}}) = \max\{[Z_o(x_{0 \ldots T})]_{o'} - \max_{o \in \bar{O}} [Z_o(x_{0 \ldots T})]_o, -\kappa\} \quad (6)$$

where $\bar{O}$ is the set of outputs that score of adversarial agent is greater than target agent i.e. $\bar{O} = \{o \in \bar{O} | S_{adv}(o) > S_{ori}(o)\}$, and $\kappa \geq 0$ denotes the confidence margin parameter. Note that $\hat{\mathbf{x}}$ is a sub-sequence in $x_{0 \ldots T}$, so the right hand side of (6) is a function of $\hat{\mathbf{x}}$.

Combining these two equations together, we can get our final objective function:

$$\min_{\hat{\mathbf{x}}} \quad L(\hat{\mathbf{x}}) \quad (7)$$
$$\text{s.t. } [Z_r(x_{0 \ldots T-1})]_{k_{sel}} - \max_{i \neq k_{sel}} [Z_r(x_{0 \ldots T-1})]_i \geq 0$$

Eq (7) is a discrete optimization problem since $\hat{x}$ is the sentence produced by adversarial agent.

In this paper, we use a modified version of the greedy algorithm to optimize (7). Although the original algorithm proposed in (Yang et al., 2018) only considered the unconstrained discrete problem, we show that the following slightly modified version performs well for solving (7). At each iteration, we try to replace each word in $\hat{x}$ by the special token $\langle PAD \rangle$. A word that achieves minimal loss after swapping with $\langle PAD \rangle$ is then selected as the word to be replaced. Then we try to replace the selected word with each word in the vocabulary. For all the trials that satisfy the constraint, we choose the one with minimal loss and conduct the actual change. We run this procedure iteratively to minimize (7). In the experiments, we only replace two words in $\hat{x}$ to ensure the fluency and correctness of the adversarial sentences.

5.1.2 Preemptive attack

The other attack strategy is to produce a sentence to guide the target agent to lower its demand in the reply instead of making target agent say end-of-dialog token. And after the reply from target agent, the adversarial agent speaks the end-of-dialogue token to enter the decision phase. Similar to the reactive attack, adversarial agent's score should be greater than target agent's score in the decision phase. Clearly, this strategy is more challenging than the previous one because there is an intermediate sentence spoken by the target agent before end-of-dialogue. We call this preemptive attack.

Let $\hat{\mathbf{x}} = x_{t_n \ldots t_{n_T}}$ be the output sentence generated by adversarial agent in the speaking model after turn n, where t_n is the first word and t_{n_T} is the last word of the sentence. Similarly, we could formally turn the intuition into optimization problem as follows:

$$L(\hat{\mathbf{x}}) = \max\{[Z_o(x_{0 \ldots T})]_{o'} - \max_{o \in \bar{O}} [Z_o(x_{0 \ldots T})]_o, -\kappa\} \quad (8)$$

Since we do not need to force target agent to say end-of-dialogue, the problem becomes an unconstrained discrete optimization problem. We then

Algorithm 1 Arbitrary turn attack algorithm

```
**Input:** Target agent B, Input goal $g$
**Output:** Dialogue $x_{0...T}$, Agent score $S_{adv}$ and $S_{ori}$
while $\langle selection \rangle$ is not generated do
  Set the loss $L(\cdot)$ to be (7)
  Optimize the Loss $L(\cdot)$
  if $L(\cdot) < 0$ then
    Add the output into the dialogue
  else
    Set the loss $L(\cdot)$ in to be (8)
    Optimize the Loss $L(\cdot)$
    if $L(\cdot) < 0$ then
      Add the output into the dialogue
    else
      if Transfer Attack then
        Randomly add a sentence in $L$ (malicious sentences) into the dialogue.
      else
        Add the sentence generated by regular speaking model into the dialogue (delayed attack).
      end if
    end if
  end if
end while
Generate $o$ using dialogue $x_{0...T}$
Calculate $S_{adv}$ and $S_{ori}$
**Return:** $x_{0...T}$,$S_{adv}$,$S_{ori}$
```

directly apply the unconstrained version of greedy algorithm (Yang et al., 2018) to solve it.

5.2 Force target agent to select at arbitrary turn

While we could let our adversarial agent and the target agent negotiate n turns, it is still unknown which n should be chosen to get the best performance. In other words, we aim to not only know what to say but also when to say to fool the target agent.

We propose two strategies to force target agent to make bad decisions at arbitrary turn. The details are presented in Algorithm 1. When it is the turn for adversarial agent to speak, we first try to apply reactive and preemptive attacks. If both attacks couldn't make the loss $L(\cdot)$ less than 0, there are two strategies: 1) just output the sentence generated by the regular speaking module (delayed attack), and 2) conduct transfer attack. The comparisons can be found in the experiments.

6 Adversarial Training

Adversarial training is a popular method to improve the robustness of machine learning models (Miyato et al., 2016; Madry et al., 2018). In this section, we use the agents designed in the previous sections to improve the robustness of the target agent.

In standard adversarial training for neural network models (Goodfellow et al., 2014; Jia and Liang, 2017), adversarial examples (images or sentences) generated by an attack are added to the training procedure to refine the model. Since our setting is interactive and there is no fixed data used in selfplay, we should conduct training with *adversarial agents* instead of adversarial examples. Moreover, as pointed out by (Jia and Liang, 2017), training on the examples generated by a single attack will lead to over-fitting to a particular attack, so we should do adversarial training iteratively.

Taking the black-box RL agent as an example, we consider the following min-max formulation:

$$\min_{\theta^{ori}}\{\max_{\theta^{adv}} S_{adv} - S_{ori}\}, \tag{9}$$

where θ^{ori} is the weights for the target agent and θ^{adv} is the weights for the adversarial black-box agent. We solve (9) by the following alternating minimization procedure. At each iteration, we first update the target agent (θ^{ori}) using the standard policy gradient algorithm, and then use our RL algorithm in Section 4.1 to update adversarial agent to counter the target model. We iteratively conduct these updates until convergence. The experiments show that the adversarial training procedure can improve the robustness not only under RL attack but also under other white-box attacks.

7 Experimental Results

We perform extensive experiments on evaluating the robustness of the negotiation agents developed in (Lewis et al., 2017). Furthermore, we show that the robustness of negotiation agents can be significantly improved using the proposed adversarial training procedure. Our codes are publicly available at `https://github.com/cmhcbb/Robustness-of-Dialogue-systems`.

7.1 Experimental Setup

We use the code released by the authors (Lewis et al., 2017) and follow their instructions to get the target end-to-end negotiation agents. More specifically, we first train the model on 5808 dialogues, based on 2236 unique scenarios in supervised way to imitate the actions of human users. We call this model supervised model (SV agent). Then we use reinforcement learning to conduct goal-oriented training in order to maximize the agent' reward. The second model is called the reinforcement learning model (RL agent). As a result, when doing selfplay between RL agent and SV agent, we could get RL agent with 5.86 perplexity, 89.57% agreement and 7.23 average score, while SV agent achieves 5.47 perplexity and 4.55 average score. These numbers are similar to the numbers reported in (Lewis et al., 2017).

To evaluate the robustness of these agents, we conduct all the proposed attacks on both supervised model (SV agent) and reinforcement learning model (RL agent). The successfulness of an attack is measured by average score advantage and positive advantage rate (PAR). Average score advantage is defined by averaged adversarial agent's score minus average target agent's score. The value is in the region of $[-10, 10]$ since the total values are controlled to be 10 for both sides, and a larger advantage indicates a more successful attack. Also, we define positive advantage rate (PAR) as the ratio of dialogues that the adversarial agent gets a higher score than the target agent. We will see that most attacks developed in this paper will improve both average score advantage and PAR. Note that this is the first work on attacking a goal-oriented dialogue agent so there is no previous method that could be included in the comparisons.

7.2 Results on Black-box Attacks

As introduced in Section 4, we have two black-box attacks: reinforcement learning attack (RL attack) and Transfer attack.

RL Attack. In the reinforcement learning attack, we use a learning rate of 0.1, clip gradients above 1.0, and set the discount factor $\gamma = 0.95$ in (4). We train the adversarial agent for 4 epochs on all scenarios. From Table 2, we observe that with 100% agreement rate, our adversarial agent could gain 2.32 score advantage against the RL agent and 4.25 advantage against the SV agent. Also, our agent achieves a relatively high positive advantage rate at 84.45% and 69.35% respectively. We show some adversarial dialogues played by adversarial agent and target agent in Table 3. It shows that RL agent is able to identify the weak point of target agent by saying ”take book you get rest”, which could easily let the agent accept the deal and make a bad selection that is inconsistent with the context of dialogue.

Transfer attack. In transfer attack, we first let our adversarial agent speak the sentence generated by the speaking model with target agent for 3 turns. If the end-of-dialog token has never been mentioned, in the 4th turn, the adversarial agent speaks the sentence generated by our RL agent. The detailed results are shown in Table 2. We observe that the transfer attack is not successful—only -0.13 and -1.189 score advantage are achieved. We found that transferring sentences between dialogues is not successful because the item values and conversation histories are quite different between dialogues.

7.3 Results on White-box Attacks

We conduct the white-box attacks introduced in Section 5.

Force target agent to select at a fixed turn. There are two types of algorithms (reactive attack and preemptive attack) introduced in Section 5.1. The detailed results are shown in Table 2. We observe that the reactive attack could achieve better results than black-box method with 5.40 score advantage against SV agent and 4.98 score advantage against RL agent. On the other hand, preemptive attack is not that successful—it gets 2.81 advantage against SV agent and 0.77 score advantage against RL agent. Furthermore, we have included some adversarial dialogues played by white-box adversarial agent and target agent in Table 4. From these examples, we could see that white-box adversarial agent could generate the adversarial sentences, slightly unnatural however still readable, that could fool the target agent to make terrible decisions.

Force target agent to select at arbitrary turn. To determine when should we begin the attack, we design combinations of reactive attack, preemptive attack and transfer attack or delayed attack in Section 5.2. Here, we conduct experiments to validate the effectiveness of these two attack combinations. From Table 2, the combinations achieve better results than all the previous attacks. The best result is achieved by the combination of reactive attack, preemptive attack and delayed attack

	vs SV agent			vs RL agent		
Model	**PAR%**	**Score(advantage)**	**Agreement%**	**PAR%**	**Score(advantage)**	**Agreement%**
RL model(w/o attack)	75.79	7.23 vs 4.55 (2.68)	89.57	44.70	5.05 vs 5.00 (0.05)	76.36
Transfer attack	44.43	6.41 vs 6.54 (-0.13)	100	36.10	5.65 vs 6.84 (-1.19)	100
RL attack	84.45	8.28 vs 4.03 (4.25)	100	69.35	7.11 vs 4.79 (2.32)	100
Reactive attack	87.00	8.83 vs 3.43 (5.40)	100	90.23	8.72 vs 3.77 (4.95)	100
Preemptive attack	71.86	7.76 vs 4.95 (2.81)	100	69.23	6.78 vs 6.01 (0.77)	100
RA+PA+DA	84.33	8.79 vs 2.96 (5.83)	100	86.93	8.73 vs 2.95 (5.78)	100
RA+PA+TA	83.12	8.67 vs 3.05 (5.62)	100	89.74	8.62 vs 2.92 (5.70)	100

Table 2: Negotiation task evaluation with different adversarial agent on 2000 randomly generated scenarios, against the supervised model and reinforcement learning model. The maximum socre is 10. When agents failed to agree, all agents get 0 score. PAR stands for positive advantage rate. RA+PA+DA stands for the combination of reactive attack, preemptive attacka and delayed attack. RA+PA+TA stands for the combination of reactive attack, preemptive attacka and transfer attack.

Input		
Adv agent	1x**book** value 1 4x**hat** value 1 1x**ball** value 5	
RL agent	1x**book** value 2 4x**hat** value 1 1x**ball** value 4	
Adv agent	i want the hats and 2 balls	
RL agent	i need the balls and the hat	
Adv agent	take book you get rest	
RL agent	deal	
Adv agent	$\langle selection \rangle$	
Output		**Reward**
Adv agent	4x**hat** 1x**ball**	9/10
RL agent	1x**book**	2/10

Table 3: Dialogue example generated by black-box RL attack agent against RL agent.

Input		
Adv agent	1x**book** value 0 1x**hat** value 1 3x**ball** value 3	
RL agent	1x**book** value 1 1x**hat** value 0 3x**ball** value 3	
Adv agent	i would like the balls and the hat	
RL agent	i need the balls and the book	
Adv agent	i need the balls and fine book	
RL agent	$\langle selection \rangle$	
Output		**Reward**
Adv agent	1x**hat** 1x**book** 3x**ball**	10/10
RL agent		0/10

Table 4: Dialogue example generated by reactive attack agent against RL agent.

Input		
Adv agent	1x**book** value 4 2x**hat** value 1 2x**ball** value 2	
RL agent	1x**book** value 8 2x**hat** value 0 2x**ball** value 1	
RL agent	i would like the book and the hat.	
Adv agent	i want reasonable balls and book	
RL agent	$\langle selection \rangle$	
Output		**Reward**
Adv agent	1x**book** 2x**ball**	8/10
RL agent	2x**hat**	0/10

Table 5: Dialogue example generated by RA+PA+DA attack agent against RL agent.

(RA+PA+DA), which gets 5.83 advantage against SV agent and 5.78 score advantage against RL agent, with very high positive advantage rates at 84.33% and 86.93% respectively. We have included some adversarial dialogues played by this adversarial agent and the target agent in Table 5. We observe that with the delayed attack, the adversarial agent can decide **when to attack**, thus achieves much better performance than attacking at a fixed turn.

7.4 Adversarial Training

Using the algorithm proposed in Section 6, we conduct adversarial training using the black-box RL attack model. The results are shown in Table 6. First, we observe that the adversarial trained model achieves much better performance against black-box RL attack; the advantage of RL attack drops from 2.32 to -1.8. Moreover, the model achieves consistently better performance against other white-box attacks. For instance, the advantage of the strongest RA+PA+DA attack is reduced from 5.78 to 3.98.

	vs advtrain model		
Model	**PAR%**	**Score(advantage)**	**Agreement%**
RL model(w/o attack)	48.67	6.51 vs 6.64 (-0.13)	91.75
Transfer attack	23.05	4.93 vs 7.59 (-2.66)	100
RL attack	62.61	5.71 vs 7.51 (-1.80)	100
Reactive attack	80.76	8.83 vs 4.31 (4.52)	100
Preemptive attack	34.39	5.64 vs 7.41 (-1.77)	100
RA+PA+DA	73.96	8.05 vs 4.07 (3.98)	100
RA+PA+TA	73.45	8.06 vs 4.13 (3.93)	100

Table 6: Negotiation task evaluation with different adversarial agent on 2000 randomly generated scenarios, against adversarial trained model.

7.5 Analysis and Discussions

RL agents are more robust than SV agents. From Table 2, we could see that all the attack methods perform better when facing SV agents than RL agents. It is because that SV agents only learn to mimic human's action and are trained only on human data. Therefore, it is reasonable that RL agents are more robust than SV agents.

The importance of arbitrary turns. In reactive attack and preemptive attack, we begin our attack at the n-th turn and we set $n = 2$ in the experiments. Here we show the results with different n in Table 7. We observe that the performance of white-box attacks are quite consistent with different choices of n. This probably indicates that there the best n varies for different cases. Therefore, if we could change the n from case to case adaptively, which is done by delayed attack, we could see a performance boost.

n	**PAR%**	**Score(advantage)**	**Agreement%**
1	94.02	8.84 vs 3.32 (5.52)	100
2	90.23	8.72 vs 3.77 (4.95)	100
3	92.02	8.81 vs 3.62 (5.19)	100
4	90.35	8.71 vs 3.87 (4.84)	100

Table 7: Negotiation task evaluation with different choices of n against RL model.

Adversarial training helps to improve the robustness. We then try to investigate the robustness of the adversarial trained model. We found that in the original model, it is easy for an attacker to find a sentence to quickly end the dialogue. However, after adversarial training, it becomes much harder to find such sentences. Moreover, although we only conduct adversarial training on black-box RL model, the adversarial trained model still achieves better performance against other white-box attacks. This indicates that the adversarial trained model could probably detect the slight unnaturalness of those sentences and thus have a better reading comprehension ability.

8 Conclusion

In this paper, we develop adversarial agents to evaluate the robustness of a goal-oriented dialogue system. Our experimental results show that the current NN-based models are not robust against our adversarial agents. Furthermore, by iterative adversarial training using our black-box RL agent, we can significantly improve the robustness of the dialogue system.

Acknowledgments

We acknowledge the support by NSF IIS1719097, Intel faculty award, Google Cloud and AITRICS.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2019. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 31–36.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *arXiv preprint arXiv:1801.04354*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *arXiv preprint arXiv:1704.07130*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2443–2453.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385.

Xuanqing Liu and Cho-Jui Hsieh. 2019. Adv-gan: Generator, discriminator, and adversarial attacker. In *CVPR*.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. 2019. Ad v-bnn: Improved adversarial defense through robust bayesian neural network. In *ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, EPFL-CONF-218057.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 49–54. IEEE.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Eric Wong and J Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. 2018. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *arXiv preprint arXiv:1805.12316*.

Nanyang Ye and Zhanxing Zhu. 2018. Bayesian adversarial learning. In *NIPS*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.

Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications

Pouya Pezeshkpour
University of California
Irvine, CA
pezeshkp@uci.edu

Yifan Tian
University of California
Irvine, CA
yifant@uci.edu

Sameer Singh
University of California
Irvine, CA
sameer@uci.edu

Abstract

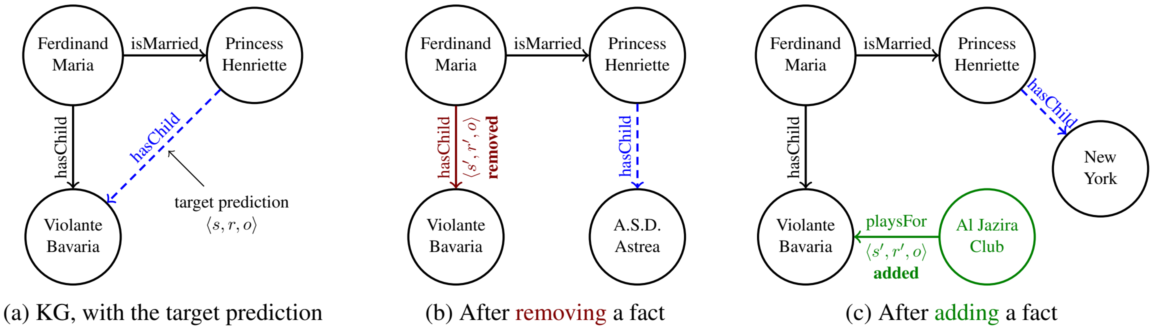
Representing entities and relations in an embedding space is a well-studied approach for machine learning on relational data. Existing approaches, however, primarily focus on improving accuracy and overlook other aspects such as robustness and interpretability. In this paper, we propose adversarial modifications for link prediction models: identifying the fact to add into or remove from the knowledge graph that changes the prediction for a target fact after the model is retrained. Using these single modifications of the graph, we identify the most influential fact for a predicted link and evaluate the sensitivity of the model to the addition of fake facts. We introduce an efficient approach to estimate the effect of such modifications by approximating the change in the embeddings when the knowledge graph changes. To avoid the combinatorial search over all possible facts, we train a network to *decode* embeddings to their corresponding graph components, allowing the use of gradient-based optimization to identify the adversarial modification. We use these techniques to evaluate the robustness of link prediction models (by measuring sensitivity to additional facts), study interpretability through the facts most responsible for predictions (by identifying the most influential neighbors), and detect incorrect facts in the knowledge base.

1 Introduction

Knowledge graphs (KG) play a critical role in many real-world applications such as search, structured data management, recommendations, and question answering. Since KGs often suffer from incompleteness and noise in their facts (links), a number of recent techniques have proposed models that *embed* each entity and relation into a vector space, and use these embeddings to predict facts. These dense representation models for link prediction include tensor factorization [Nickel et al., 2011, Socher et al., 2013, Yang et al., 2015], algebraic operations [Bordes et al., 2011, 2013b, Dasgupta et al., 2018], multiple embeddings [Wang et al., 2014, Lin et al., 2015, Ji et al., 2015, Zhang et al., 2018], and complex neural models [Dettmers et al., 2018, Nguyen et al., 2018]. However, there are only a few studies [Kadlec et al., 2017, Sharma et al., 2018] that investigate the quality of the different KG models. There is a need to go beyond just the accuracy on link prediction, and instead focus on whether these representations are robust and stable, and what facts they make use of for their predictions.

In this paper, our goal is to design approaches that minimally change the graph structure such that the prediction of a target fact changes the most after *the embeddings are relearned*, which we collectively call *Completion Robustness and Interpretability via Adversarial Graph Edits* (CRIAGE). First, we consider perturbations that remove a neighboring link for the target fact, thus identifying the *most influential* related fact, providing an explanation for the model's prediction. As an example, consider the excerpt from a KG in Figure 1a with two observed facts, and a target *predicted fact* that Princes Henriette is the parent of Violante Bavaria. Our proposed graph perturbation, shown in Figure 1b, identifies the existing fact that Ferdinal Maria is the father of Violante Bavaria as the one when removed and model retrained, will change the prediction of Princes Henriette's child. We also study attacks that add a new, fake fact into the KG to evaluate the robustness and sensitivity of link prediction models to small additions to the graph. An example attack for the original graph in Figure 1a, is depicted in Figure 1c. Such perturbations to the the training data are from a family of adversarial modifications that have been applied to other machine learning tasks, known as *poisoning* [Biggio et al., 2012, Corona et al., 2013, Biggio

Proceedings of NAACL-HLT 2019, pages 3336–3347
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) KG, with the target prediction (b) After removing a fact (c) After adding a fact

Figure 1: **Completion Robustness and Interpretability via Adversarial Graph Edits (CRIAGE):** Change in the graph structure that changes the prediction of the *retrained* model, where (a) is the original sub-graph of the KG, (b) removes a neighboring link of the target, resulting in a change in the prediction, and (c) shows the effect of adding an attack triple on the target. These modifications were identified by our proposed approach.

et al., 2014, Zügner et al., 2018].

Since the setting is quite different from traditional adversarial attacks, search for link prediction adversaries brings up unique challenges. To find these minimal changes for a target link, we need to identify the fact that, when added into or removed from the graph, will have the biggest impact on the predicted score of the target fact. Unfortunately, computing this change in the score is expensive since it involves retraining the model to recompute the embeddings. We propose an efficient estimate of this score change by approximating the change in the embeddings using Taylor expansion. The other challenge in identifying adversarial modifications for link prediction, especially when considering addition of fake facts, is the combinatorial search space over possible facts, which is intractable to enumerate. We introduce an *inverter* of the original embedding model, to *decode* the embeddings to their corresponding graph components, making the search of facts tractable by performing efficient gradient-based continuous optimization.

We evaluate our proposed methods through following experiments. First, on relatively small KGs, we show that our approximations are accurate compared to the true change in the score. Second, we show that our additive attacks can effectively reduce the performance of state of the art models [Yang et al., 2015, Dettmers et al., 2018] up to 27.3% and 50.7% in Hits@1 for two large KGs: WN18 and YAGO3-10. We also explore the utility of adversarial modifications in explaining the model predictions by presenting rule-like descriptions of the most influential neighbors. Finally, we use adversaries to detect errors in the KG, obtaining up to 55% accuracy in detecting errors.

2 Background and Notation

In this section, we briefly introduce some notations, and existing relational embedding approaches that model knowledge graph completion using dense vectors. In KGs, facts are represented using triples of subject, relation, and object, $\langle s, r, o\rangle$, where $s, o \in \xi$, the set of entities, and $r \in \mathcal{R}$, the set of relations. To model the KG, a scoring function $\psi : \xi \times \mathcal{R} \times \xi \rightarrow \mathbb{R}$ is learned to evaluate whether any given fact is true. In this work, we focus on *multiplicative* models of link prediction[1], specifically DistMult [Yang et al., 2015] because of its simplicity and popularity, and ConvE [Dettmers et al., 2018] because of its high accuracy. We can represent the scoring function of such methods as $\psi(s, r, o) = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) \cdot \mathbf{e}_o$, where $\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o \in \mathbb{R}^d$ are embeddings of the subject, relation, and object respectively. In DistMult, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) = \mathbf{e}_s \odot \mathbf{e}_r$, where $\odot$ is element-wise multiplication operator. Similarly, in ConvE, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ is computed by a convolution on the concatenation of $\mathbf{e}_s$ and $\mathbf{e}_r$.

We use the same setup as Dettmers et al. [2018] for training, i.e., incorporate binary cross-entropy loss over the triple scores. In particular, for subject-relation pairs (s, r) in the training data G, we use binary $y_o^{s,r}$ to represent negative and positive facts. Using the model's probability of truth as $\sigma(\psi(s, r, o))$ for $\langle s, r, o\rangle$, the loss is defined as:

$$\mathcal{L}(G) = \sum_{(s,r)} \sum_{o} y_o^{s,r} \log(\sigma(\psi(s, r, o))) + (1 - y_o^{s,r}) \log(1 - \sigma(\psi(s, r, o))). \quad (1)$$

Gradient descent is used to learn the embeddings $\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o$, and the parameters of $\mathbf{f}$, if any.

---

[1] As opposed to *additive* models, such as TransE [Bordes et al., 2013a], as categorized in Sharma et al. [2018].

3 Completion Robustness and Interpretability via Adversarial Graph Edits (CRIAGE)

For adversarial modifications on KGs, we first define the space of possible modifications. For a target triple $\langle s, r, o\rangle$, we constrain the possible triples that we can remove (or inject) to be in the form of $\langle s', r', o\rangle$ i.e s' and r' may be different from the target, but the object is not. We analyze other forms of modifications such as $\langle s, r', o'\rangle$ and $\langle s, r', o\rangle$ in appendices A.1 and A.2, and leave empirical evaluation of these modifications for future work.

3.1 Removing a fact (CRIAGE-Remove)

For explaining a target prediction, we are interested in identifying the observed fact that has the most influence (according to the model) on the prediction. We define *influence* of an observed fact on the prediction as the change in the prediction score if the observed fact was not present when the embeddings were learned. Previous work have used this concept of influence similarly for several different tasks [Kononenko et al., 2010, Koh and Liang, 2017]. Formally, for the target triple $\langle s, r, o\rangle$ and observed graph G, we want to identify a neighboring triple $\langle s', r', o\rangle \in G$ such that the score $\psi(s, r, o)$ when trained on G and the score $\overline{\psi}(s, r, o)$ when trained on $G-\{\langle s', r', o\rangle\}$ are maximally different, i.e.

$$\underset{(s',r')\in\text{Nei}(o)}{\operatorname{argmax}} \quad \Delta_{(s',r')}(s, r, o) \tag{2}$$

where $\Delta_{(s',r')}(s, r, o) = \psi(s, r, o)-\overline{\psi}(s, r, o)$, and $\text{Nei}(o) = \{(s', r')|\langle s', r', o\rangle \in G\}$.

3.2 Adding a new fact (CRIAGE-Add)

We are also interested in investigating the robustness of models, i.e., how sensitive are the predictions to small additions to the knowledge graph. Specifically, for a target prediction $\langle s, r, o\rangle$, we are interested in identifying a single fake fact $\langle s', r', o\rangle$ that, when added to the knowledge graph G, changes the prediction score $\psi(s, r, o)$ the most. Using $\overline{\psi}(s, r, o)$ as the score after training on $G \cup \{\langle s', r', o\rangle\}$, we define the adversary as:

$$\underset{(s',r')}{\operatorname{argmax}} \, \Delta_{(s',r')}(s, r, o) \tag{3}$$

where $\Delta_{(s',r')}(s, r, o) = \psi(s, r, o) - \overline{\psi}(s, r, o)$. The search here is over any possible $s' \in \xi$, which is often in the millions for most real-world KGs, and $r' \in \mathcal{R}$. We also identify adversaries that *increase* the prediction score for specific false triple, i.e., for a target fake fact $\langle s, r, o\rangle$, the adversary is $\operatorname{argmax}_{(s',r')} -\Delta_{(s',r')}(s, r, o)$, where $\Delta_{(s',r')}(s, r, o)$ is defined as before.

3.3 Challenges

There are a number of crucial challenges when conducting such adversarial attack on KGs. First, evaluating the effect of changing the KG on the score of the target fact ($\overline{\psi}(s, r, o)$) is expensive since we need to update the embeddings by retraining the model on the new graph; a very time-consuming process that is at least linear in the size of G. Second, since there are many candidate facts that can be added to the knowledge graph, identifying the most promising adversary through search-based methods is also expensive. Specifically, the search size for unobserved facts is $|\xi| \times |\mathcal{R}|$, which, for example in YAGO3-10 KG, can be as many as $4.5M$ possible facts for a single target prediction.

4 Efficiently Identifying the Modification

In this section, we propose algorithms to address mentioned challenges by (1) approximating the effect of changing the graph on a target prediction, and (2) using continuous optimization for the discrete search over potential modifications.

4.1 First-order Approximation of Influence

We first study the addition of a fact to the graph, and then extend it to cover removal as well. To capture the effect of an adversarial modification on the score of a target triple, we need to study the effect of the change on the vector representations of the target triple. We use $\mathbf{e}_s$, $\mathbf{e}_r$, and $\mathbf{e}_o$ to denote the embeddings of s, r, o at the solution of $\operatorname{argmin} \mathcal{L}(G)$, and when considering the adversarial triple $\langle s', r', o\rangle$, we use $\overline{\mathbf{e}_s}$, $\overline{\mathbf{e}_r}$, and $\overline{\mathbf{e}_o}$ for the new embeddings of s, r, o, respectively. Thus $\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r}, \overline{\mathbf{e}_o}$ is a solution to $\operatorname{argmin} \mathcal{L}(G \cup \{\langle s', r', o\rangle\})$, which can also be written as $\operatorname{argmin} \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$. Similarly, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ changes to $\mathbf{f}(\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r})$ after retraining.

Since we only consider adversaries in the form of $\langle s', r', o\rangle$, we only consider the effect of the attack on $\mathbf{e}_o$ and neglect its effect on $\mathbf{e}_s$ and $\mathbf{e}_r$. This assumption is reasonable since the adversary is connected with o and directly affects its embedding when added, but it will only have a secondary, negligible effect on $\mathbf{e}_s$ and $\mathbf{e}_r$, in comparison to its effect on $\mathbf{e}_o$. Further, calculating the effect of the attack on $\mathbf{e}_s$ and $\mathbf{e}_r$ requires a third order derivative of the loss, which is not practical ($O(n^3)$ in the number of parameters). In other words, we assume that $\overline{\mathbf{e}_s} \simeq \mathbf{e}_s$ and $\overline{\mathbf{e}_r} \simeq \mathbf{e}_r$. As a result, to calculate the effect of the attack, $\overline{\psi}(s,r,o) - \psi(s,r,o)$, we need to compute $\overline{\mathbf{e}_o} - \mathbf{e}_o$, followed by:

$$\overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) \tag{4}$$

where $\mathbf{z}_{s,r} = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$. We now derive an efficient computation for $\overline{\mathbf{e}_o} - \mathbf{e}_o$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$ over $\mathbf{e}_o$ is:

$$\nabla_{e_o}\mathcal{L}(\overline{G}) = \nabla_{e_o}\mathcal{L}(G) - (1-\varphi)\mathbf{z}_{s',r'} \tag{5}$$

where $\mathbf{z}_{s',r'} = \mathbf{f}(\mathbf{e}'_s, \mathbf{e}'_r)$, and $\varphi = \sigma(\psi(s', r', o))$. At convergence, after retraining, we expect $\nabla_{e_o}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_o}\mathcal{L}(\overline{G})$ to get:

$$\begin{aligned} 0 \simeq & -(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'} + \\ & (H_o + \varphi(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'}\mathbf{z}_{s',r'})(\overline{\mathbf{e}_o} - \mathbf{e}_o) \end{aligned} \tag{6}$$

where H_o is the $d \times d$ Hessian matrix for o, i.e., second order derivative of the loss w.r.t. $\mathbf{e}_o$, computed sparsely. Solving for $\overline{\mathbf{e}_o} - \mathbf{e}_o$ gives us, $\overline{\mathbf{e}_o} - \mathbf{e}_o =$:

$$(1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\mathsf{T}}_{s',r'}.$$

In practice, H_o is positive definite, making $H_o + \varphi(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'}\mathbf{z}_{s',r'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\begin{aligned} &\overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) \qquad (7) \\ &= \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\mathsf{T}}_{s',r'}). \end{aligned}$$

Calculating this expression is efficient since H_o is a $d \times d$ matrix (d is the embedding dimension), and $\mathbf{z}_{s,r}, \mathbf{z}_{s',r'} \in \mathbb{R}^d$. Similarly, we estimate the score change of $\langle s, r, o\rangle$ after *removing* $\langle s', r', o\rangle$ as:

$$-\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\mathsf{T}}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\mathsf{T}}_{s',r'}).$$

4.2 Continuous Optimization for Search

Using the approximations provided in the previous section, Eq. (7) and (4.1), we can use brute force enumeration to find the adversary $\langle s', r', o\rangle$. This approach is feasible when removing an observed triple since the search space of such modifications is usually small; it is the number of observed facts that share the object with the target. On the other hand, finding the most influential unobserved fact to add requires search over a much larger space of all possible unobserved facts (that share the object). Instead, we identify the most influential unobserved fact $\langle s', r', o\rangle$ by using a gradient-based algorithm on vector $\mathbf{z}_{s',r'}$ in the embedding space (reminder, $\mathbf{z}_{s',r'} = \mathbf{f}(\mathbf{e}'_s, \mathbf{e}'_r)$), solving the following continuous optimization problem in $\mathbb{R}^d$:

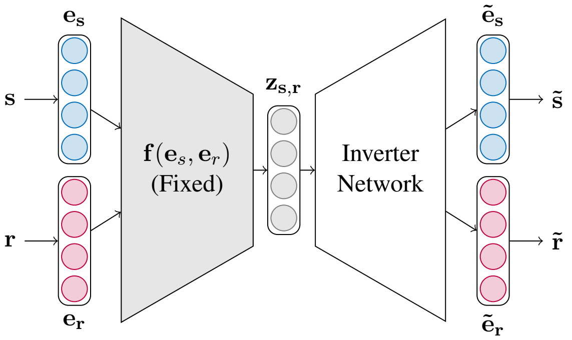


Figure 2: **Inverter Network** The architecture of our inverter function that translate $\mathbf{z}_{s,r}$ to its respective $(\tilde{s}, \tilde{r})$. The encoder component is fixed to be the encoder network of DistMult and ConvE respectively.

$$\underset{\mathbf{z}_{s',r'}}{\operatorname{argmax}}\, \Delta_{(s',r')}(s,r,o). \tag{8}$$

After identifying the optimal $\mathbf{z}_{s',r'}$, we still need to generate the pair (s', r'). We design a network, shown in Figure 2, that maps the vector $\mathbf{z}_{s',r'}$ to the entity-relation space, i.e., translating it into (s', r'). In particular, we train an auto-encoder where the encoder is fixed to receive the s and r as one-hot inputs, and calculates $\mathbf{z}_{s,r}$ in the same way as the DistMult and ConvE encoders respectively (using trained embeddings). The decoder is trained to take $\mathbf{z}_{s,r}$ as input and produce s and r, essentially inverting $\mathbf{f}$ and the embedding layers. As our decoder, for DistMult, we pass $\mathbf{z}_{s,r}$ through a linear layer and then use two other linear layers for the subject and the relation separately, providing one-hot vectors as $\tilde{s}$ and $\tilde{r}$. For ConvE, we pass $\mathbf{z}_{s,r}$ through a deconvolutional layer, and then use the same architecture as the DistMult decoder. Although we could use maximum inner-product search [Shrivastava and Li, 2014] for DistMult instead of our defined inverter function, we are looking for a general approach that works across multiple models.

We evaluate the performance of our inverter networks (one for each model/dataset) on correctly recovering the pairs of subject and relation from the test set of our benchmarks, given the $\mathbf{z}_{s,r}$. The accuracy of recovered pairs (and of each argument)

	WordNet		YAGO	
	DistMult	ConvE	DistMult	ConvE
Recover s	93.4	96.1	97.2	98.1
Recover r	91.3	95.3	99.0	99.6
Recover $\{s, r\}$	89.5	94.2	96.4	98.0

Table 1: **Inverter Functions Accuracy**, we calculate the accuracy of our inverter networks in correctly recovering the pairs of subject and relation from the test set of our benchmarks.

	# Rels	#Entities	# Train	#Test
Nations	56	14	1592	200
Kinship	26	104	4,006	155
WN18	18	40,943	141,442	5000
YAGO3-10	37	123,170	1,079,040	5000

Table 2: **Data Statistics** of the benchmarks.

is given in Table 1. As shown, our networks achieve a very high accuracy, demonstrating their ability to invert vectors $\mathbf{z}_{s,r}$ to $\{s, r\}$ pairs.

5 Experiment Setup

Datasets To evaluate our method, we conduct several experiments on four widely used KGs. To validate the accuracy of the approximations, we use smaller sized Kinship and Nations KGs for which we can make comparisons against more expensive but less approximate approaches. For the remaining experiments, we use YAGO3-10 and WN18 KGs, which are closer to real-world KGs in their size and characteristics (see Table 2).

Models We implement all methods using the same loss and optimization for training, i.e., AdaGrad and the binary cross-entropy loss. We use validation data to tune the hyperparameters and use a grid search to find the best hyperparameters, such as regularization parameter, and learning rate of the gradient-based method. To capture the effect of our method on link prediction task, we study the change in commonly-used metrics for evaluation in this task: mean reciprocal rank (MRR) and Hits@K. Further, we use the same hyperparameters as in Dettmers et al. [2018] for training link prediction models for these knowledge graphs.

Influence Function We also compare our method with influence function (IF) [Koh and Liang, 2017]. The influence function approximates the effect of upweighting a training sample on the loss for a specific test point. We use IF to approximate the change in the loss after removing a triple as:

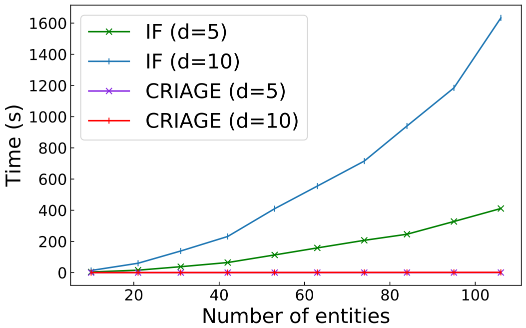


Figure 3: **Influence function vs CRIAGE**. We plot the average time (over 10 facts) of influence function (IF) and CRIAGE to identify an adversary as the number of entities in the Kinship KG is varied (by randomly sampling subgraphs of the KG). Even with small graphs and dimensionality, IF quickly becomes impractical.

$$\mathcal{I}_{\text{up,loss}}(\langle s', r', o\rangle, \langle s, r, o\rangle) = \\ -\nabla_\theta \mathcal{L}(\langle s, r, o\rangle, \hat{\theta})^\intercal H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(\langle s', r', o\rangle, \hat{\theta}) \quad (9)$$

where $\langle s', r', o\rangle$ and $\langle s, r, o\rangle$ are training and test samples respectively, $\hat{\theta}$ represents the optimum parameters and $\mathcal{L}(\langle s, r, o\rangle, \hat{\theta})$ represents the loss function for the test sample $\langle s, r, o\rangle$. Influence function does not scale well, so we only compare our method with IF on the smaller size KGs.

6 Experiments

We evaluate CRIAGE by (6.1) comparing CRIAGE estimate with the actual effect of the attacks, (6.2) studying the effect of adversarial attacks on evaluation metrics, (6.3) exploring its application to the interpretability of KG representations, and (6.4) detecting incorrect triples.

6.1 Influence Function vs CRIAGE

To evaluate the quality of our approximations and compare with influence function (IF), we conduct leave one out experiments. In this setup, we take all the neighbors of a random target triple as candidate modifications, remove them one at a time, retrain the model each time, and compute the *exact* change in the score of the target triple. We can use the magnitude of this change in score to rank the candidate triples, and compare this *exact* ranking with ranking as predicted by: CRIAGE-Remove, influence function with and without Hessian matrix, and the original model score (with the intuition that facts that the model is most confident of will have

Methods	Nations				Kinship			
	Adding		Removing		Adding		Removing	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Ranking Based on Score	0.03	0.02	-0.01	-0.01	-0.09	-0.06	0.01	0.01
Influence Function without Hessian	0.15	0.12	0.12	0.1	0.77	0.71	0.77	0.71
CRIAGE (Brute Force)	0.95	0.84	0.94	0.85	0.99	0.97	0.99	0.95
Influence Function	0.99	0.95	0.99	0.96	0.99	0.98	0.99	0.98

Table 3: **Ranking modifications by their impact on the target**. We compare the *true* ranking of candidate triples with a number of approximations using ranking correlation coefficients. We compare our method with influence function (IF) with and without Hessian, and ranking the candidates based on their score, on two KGs ($d = 10$, averaged over 10 random targets). For the sake of brevity, we represent the Spearman's ρ and Kendall's τ rank correlation coefficients simply as ρ and τ.

the largest impact when removed). Similarly, we evaluate CRIAGE-Add by considering 200 random triples that share the object entity with the target sample as candidates, and rank them as above.

The average results of Spearman's ρ and Kendall's τ rank correlation coefficients over 10 random target samples is provided in Table 3. CRIAGE performs comparably to the influence function, confirming that our approximation is accurate. Influence function is slightly more accurate because they use the complete Hessian matrix over all the parameters, while we only approximate the change by calculating the Hessian over $\mathbf{e}_o$. The effect of this difference on scalability is dramatic, constraining IF to very small graphs and small embedding dimensionality ($d \leq 10$) before we run out of memory. In Figure 3, we show the time to compute a single adversary by IF compared to CRIAGE, as we steadily grow the number of entities (randomly chosen subgraphs), averaged over 10 random triples. As it shows, CRIAGE is mostly unaffected by the number of entities while IF increases quadratically. Considering that real-world KGs have tens of thousands of times more entities, making IF unfeasible for them.

6.2 Robustness of Link Prediction Models

Now we evaluate the effectiveness of CRIAGE to successfully *attack* link prediction by adding false facts. The goal here is to identify the attacks for triples in the test data, and measuring their effect on MRR and Hits@ metrics (ranking evaluations) after conducting the attack and retraining the model.

Since this is the first work on adversarial attacks for link prediction, we introduce several baselines to compare against our method. For finding the adversarial fact to add for the target triple $\langle s, r, o\rangle$, we consider two baselines: 1) choosing a random fake fact $\langle s', r', o\rangle$ (**Random Attack**); 2) finding (s', r') by first calculating $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ and then feeding $-\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ to the decoder of the inverter function (**Opposite Attack**). In addition to CRIAGE-Add, we introduce two other alternatives of our method: (1) **CRIAGE-FT**, that uses CRIAGE to *increase* the score of fake fact over a test triple, i.e., we find the fake fact the model ranks second after the test triple, and identify the adversary for them, and (2) **CRIAGE-Best** that selects between CRIAGE-Add and CRIAGE-FT attacks based on which has a higher estimated change in score.

All-Test The result of the attack on all test facts as targets is provided in the Table 4. CRIAGE-Add outperforms the baselines, demonstrating its ability to effectively attack the KG representations. It seems DistMult is more robust against random attacks, while ConvE is more robust against designed attacks. CRIAGE-FT is more effective than CRIAGE-Add since changing the score of a fake fact is easier than of actual facts; there is no existing evidence to support fake facts. We also see that YAGO3-10 models are more robust than those for WN18. Looking at sample attacks (provided in Appendix A.4), CRIAGE mostly tries to change the *type* of the target object by associating it with a subject and a relation for a different entity type.

Uncertain-Test To better understand the effect of attacks, we consider a subset of test triples that 1) the model predicts correctly, 2) difference between their scores and the negative sample with the highest score is minimum. This "Uncertain-Test" subset contains 100 triples from each of the original

	Models	**YAGO3-10**				**WN18**			
		All-Test		**Uncertain-Test**		**All-Test**		**Uncertain-Test**	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
DistMult	DistMult	0.458	37 (0)	1.0	100 (0)	0.938	93.1 (0)	1.0	100 (0)
	+ Adding Random Attack	0.442	34.9 (-2.1)	0.91	87.6 (-12.4)	0.926	91.1 (-2)	0.929	90.4 (-9.6)
	+ Adding Opposite Attack	0.427	33.2 (-3.8)	0.884	84.1 (-15.9)	0.906	87.3 (-5.8)	0.921	91 (-9)
	+ CRIAGE-Add	0.379	29.1 (-7.9)	0.71	58 (-42)	0.89	86.4 (-6.7)	0.844	81.2 (-18.8)
	+ CRIAGE-FT	0.387	27.7 (-9.3)	0.673	50.5 (-49.5)	0.86	79.2 (-13.9)	0.83	74.5 (-25.5)
	+ CRIAGE-Best	0.372	26.9 (-10.1)	0.658	49.3 (-50.7)	0.838	77.9 (-15.2)	0.814	72.7 (-27.3)
ConvE	ConvE	0.497	41.2 (0)	1.0	100 (0)	0.94	93.3 (0)	1.0	100 (0)
	+ Adding Random Attack	0.474	38.4 (-2.8)	0.889	83 (-17)	0.921	90.1 (-3.2)	0.923	89.7 (-10.3)
	+ Adding Opposite Attack	0.469	38 (-3.2)	0.874	81.9 (-18.1)	0.915	88.9 (-4.4)	0.908	88.1 (-11.9)
	+ CRIAGE-Add	0.454	36.9 (-4.3)	0.738	61.5 (-38.5)	0.897	87.8 (-5.5)	0.895	87.6 (-12.4)
	+ CRIAGE-FT	0.441	33.2 (-8)	0.703	57.4 (-42.6)	0.865	80 (-13.3)	0.874	79.5 (-20.5)
	+ CRIAGE-Best	0.423	31.9 (-9.3)	0.677	54.8 (-45.2)	0.849	79.1 (-14.2)	0.858	78.4 (-21.6)

Table 4: **Robustness of Representation Models**, the effect of adversarial attack on link prediction task. We consider two scenario for the target triples, 1) choosing the whole test dataset as the targets (All-Test) and 2) choosing a subset of test data that models are uncertain about them (Uncertain-Test).

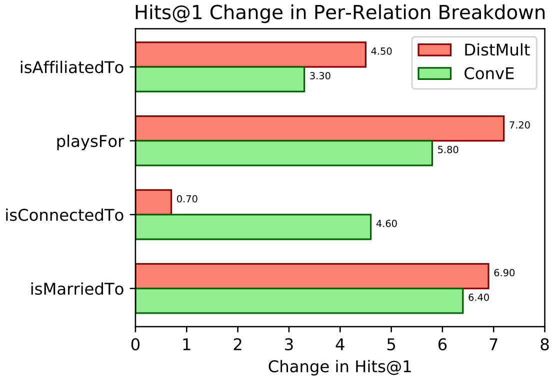


Figure 4: Per-Relation Breakdown showing the effect of CRIAGE-Add on different relations in YAGO3-10.

test sets, and we provide results of attacks on this data in Table 4. The attacks are much more effective in this scenario, causing a considerable drop in the metrics. Further, in addition to CRIAGE significantly outperforming other baselines, they indicate that ConvE's confidence is much more robust.

Relation Breakdown We perform additional analysis on the YAGO3-10 dataset to gain a deeper understanding of the performance of our model. As shown in Figure 4, both DistMult and ConvE provide a more robust representation for isAffiliatedTo and isConnectedTo relations, demonstrating the confidence of models in identifying them. Moreover, the CRIAGE affects DistMult more in playsFor and isMarriedTo relations while affecting ConvE more in isConnectedTo relations.

Rule Body, $R_1(a,c) \wedge R_2(c,b) \Rightarrow$	**Target,** $R(a,b)$
Common to both	
isConnectedTo$(a,c)\wedge$ isConnectedTo(c,b)	isConnectedTo
isLocatedIn$(a,c)\wedge$ isLocatedIn(c,b)	isLocatedIn
isAffiliatedTo$(a,c)\wedge$ isLocatedIn(c,b)	wasBornIn
isMarriedTo$(a,c)\wedge$ hasChild(c,b)	hasChild
only in DistMult	
playsFor$(a,c)\wedge$ isLocatedIn(c,b)	wasBornIn
dealsWith$(a,c)\wedge$ participatedIn(c,b)	participatedIn
isAffiliatedTo$(a,c)\wedge$ isLocatedIn(c,b)	diedIn
isLocatedIn$(a,c)\wedge$ hasCapital(c,b)	isLocatedIn
only in ConvE	
influences$(a,c)\wedge$ influences(c,b)	influences
isLocatedIn$(a,c)\wedge$ hasNeighbor(c,b)	isLocatedIn
hasCapital$(a,c)\wedge$ isLocatedIn(c,b)	exports
hasAdvisor$(a,c)\wedge$ graduatedFrom(c,b)	graduatedFrom
Extractions from DistMult [Yang et al., 2015]	
isLocatedIn$(a,c) \wedge$ isLocatedIn(c,b)	isLocatedIn
isAffiliatedTo$(a,c) \wedge$ isLocatedIn(c,b)	wasBornIn
playsFor$(a,c) \wedge$ isLocatedIn(c,b)	wasBornIn
isAffiliatedTo$(a,c) \wedge$ isLocatedIn(c,b)	diedIn

Table 5: **Extracted Rules** for identifying the most influential link. We extract the patterns that appear more than 90% times in the neighborhood of the target triple. The output of CRIAGE-Remove is presented in red.

6.3 Interpretability of Models

To be able to understand and interpret why a link is predicted using the opaque, dense embeddings, we need to find out which part of the graph was most influential on the prediction. To provide such explanations for each predictions, we identify the most influential fact using CRIAGE-Remove. Instead of focusing on individual predictions, we aggregate the explanations over the whole dataset for each relation using a simple rule extraction technique: we

Methods	$\langle s', r', o\rangle$ **Noise**		$\langle s', r, o\rangle$ **Noise**	
	Hits@1	Hits@2	Hits@1	Hits@2
Random	19.7	39.4	19.7	39.4
Lowest	16	37	26	47
CRIAGE	42	62	55	76

Table 6: **Error Detection Accuracy** in the neighborhood of 100 chosen samples. We choose the neighbor with the least value of $\Delta_{(s',r')}(s, r, o)$ as the incorrect fact. This experiment assumes we know each target fact has exactly one error.

find simple patterns on subgraphs that surround the target triple and the removed fact from CRIAGE-Remove, and appear more than 90% of the time. We only focus on extracting length-2 horn rules, i.e., $R_1(a, c) \wedge R_2(c, b) \Rightarrow R(a, b)$, where $R(a, b)$ is the target and $R_2(c, b)$ is the removed fact.

Table 5 shows extracted YAGO3-10 rules that are common to both models, and ones that are not. The rules show several interesting inferences, such that hasChild is often inferred via married parents, and isLocatedIn via transitivity. There are several differences in how the models reason as well; DistMult often uses the hasCapital as an intermediate step for isLocatedIn, while ConvE *incorrectly* uses isNeighbor. We also compare against rules extracted by Yang et al. [2015] for YAGO3-10 that utilizes the structure of DistMult: they require domain knowledge on types and cannot be applied to ConvE. Interestingly, the extracted rules contain all the rules provided by CRIAGE, demonstrating that CRIAGE can be used to accurately interpret models, including ones that are not interpretable, such as ConvE. These are preliminary steps toward interpretability of link prediction models, and we leave more analysis of interpretability to future work.

6.4 Finding Errors in Knowledge Graphs

Here, we demonstrate another potential use of adversarial modifications: finding erroneous triples in the knowledge graph. Intuitively, if there is an error in the graph, the triple is likely to be inconsistent with its neighborhood, and thus the model should put least trust on this triple. In other words, the error triple should have the least influence on the model's prediction of the training data. Formally, to find the incorrect triple $\langle s', r', o\rangle$ in the neighborhood of the train triple $\langle s, r, o\rangle$, we need to find the triple $\langle s', r', o\rangle$ that results in the *least* change $\Delta_{(s',r')}(s, r, o)$ when removed from the graph.

To evaluate this application, we inject random triples into the graph, and measure the ability of CRIAGE to detect the errors using our optimization. We consider two types of incorrect triples: 1) incorrect triples in the form of $\langle s', r, o\rangle$ where s' is chosen randomly from all of the entities, and 2) incorrect triples in the form of $\langle s', r', o\rangle$ where s' and r' are chosen randomly. We choose 100 random triples from the observed graph, and for each of them, add an incorrect triple (in each of the two scenarios) to its neighborhood. Then, after retraining DistMult on this noisy training data, we identify error triples through a search over the neighbors of the 100 facts. The result of choosing the neighbor with the least influence on the target is provided in the Table 6. When compared with baselines that randomly choose one of the neighbors, or assume that the fact with the lowest score is incorrect, we see that CRIAGE outperforms both of these with a considerable gap, obtaining an accuracy of 42% and 55% in detecting errors.

7 Related Work

Learning relational knowledge representations has been a focus of active research in the past few years, but to the best of our knowledge, this is the first work on conducting adversarial modifications on the link prediction task.

Knowledge graph embedding There is a rich literature on representing knowledge graphs in vector spaces that differ in their scoring functions [Wang et al., 2017, Goyal and Ferrara, 2018, Fooshee et al., 2018]. Although CRIAGE is primarily applicable to multiplicative scoring functions [Nickel et al., 2011, Socher et al., 2013, Yang et al., 2015, Trouillon et al., 2016], these ideas apply to additive scoring functions [Bordes et al., 2013a, Wang et al., 2014, Lin et al., 2015, Nguyen et al., 2016] as well, as we show in Appendix A.3.

Furthermore, there is a growing body of literature that incorporates an extra types of evidence for more informed embeddings such as numerical values [Garcia-Duran and Niepert, 2017], images [Oñoro-Rubio et al., 2017], text [Toutanova et al., 2015, 2016, Tu et al., 2017], and their combinations [Pezeshkpour et al., 2018]. Using CRIAGE, we can gain a deeper understanding of these methods, especially those that build their embeddings wit hmultiplicative scoring functions.

Interpretability and Adversarial Modification There has been a significant recent interest in conducting an adversarial attacks on different machine learning models [Biggio et al., 2014, Papernot et al., 2016, Dong et al., 2017, Zhao et al., 2018a,b, Brunet et al., 2018] to attain the interpretability, and further, evaluate the robustness of those models. Koh and Liang [2017] uses influence function to provide an approach to understanding black-box models by studying the changes in the loss occurring as a result of changes in the training data. In addition to incorporating their established method on KGs, we derive a novel approach that differs from their procedure in two ways: (1) instead of changes in the loss, we consider the changes in the scoring function, which is more appropriate for KG representations, and (2) in addition to searching for an attack, we introduce a gradient-based method that is much faster, especially for "adding an attack triple" (the size of search space make the influence function method infeasible). Previous work has also considered adversaries for KGs, but as part of training to improve their representation of the graph [Minervini et al., 2017, Cai and Wang, 2018].

Adversarial Attack on KG Although this is the first work on adversarial attacks for link prediction, there are two approaches [Dai et al., 2018, Zügner et al., 2018] that consider the task of adversarial attack on graphs. There are a few fundamental differences from our work: (1) they build their method on top of a path-based representations while we focus on embeddings, (2) they consider node classification as the target of their attacks while we attack link prediction, and (3) they conduct the attack on small graphs due to restricted scalability, while the complexity of our method does not depend on the size of the graph, but only the neighborhood, allowing us to attack real-world graphs.

8 Conclusions

Motivated by the need to analyze the robustness and interpretability of link prediction models, we present a novel approach for conducting adversarial modifications to knowledge graphs. We introduce CRIAGE, completion robustness and interpretability via adversarial graph edits: identifying the fact to add into or remove from the KG that changes the prediction for a target fact. CRIAGE uses (1) an estimate of the score change for any target triple after adding or removing another fact, and (2) a gradient-based algorithm for identifying the most influential modification. We show that CRIAGE can effectively reduce ranking metrics on link prediction models upon applying the attack triples. Further, we incorporate the CRIAGE to study the interpretability of KG representations by summarizing the most influential facts for each relation. Finally, using CRIAGE, we introduce a novel automated error detection method for knowledge graphs. We have release the open-source implementation of our models at: `https://pouyapez.github.io/criage`.

Acknowledgements

We would like to thank Matt Gardner, Marco Tulio Ribeiro, Zhengli Zhao, Robert L. Logan IV, Dheeru Dua and the anonymous reviewers for their detailed feedback and suggestions. This work is supported in part by Allen Institute for Artificial Intelligence (AI2) and in part by NSF awards #IIS-1817183 and #IIS-1756023. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 2014.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI Conference on Artificial Intelligence*, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, 2013a.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, 2013b.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. Understanding the origins of bias in word embeddings. *arXiv preprint arXiv:1810.03611*, 2018.

Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 2013.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, 2018.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *AAAI Conference on Artificial Intelligence*, 2018.

Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

David Fooshee, Aaron Mood, Eugene Gutman, Mohammadamin Tavakoli, Gregor Urban, Frances Liu, Nancy Huynh, David Van Vranken, and Pierre Baldi. Deep learning for chemical reaction prediction. *Molecular Systems Design & Engineering*, 2018.

Alberto Garcia-Duran and Mathias Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*, 2017.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 2018.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 2010.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI Conference on Artificial Intelligence*, 2015.

P Minervini, T Demeester, T Rocktäschel, and S Riedel. Adversarial sets for regularising neural link predictors. In *Uncertainty in Artificial Intelligence (UAI)*, 2017.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2016.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International conference on machine learning (ICML)*, 2011.

Daniel Oñoro-Rubio, Mathias Niepert, Alberto García-Durán, Roberto González-Sánchez, and Roberto J López-Sastre. Representation learning for visual-relational knowledge graphs. *arXiv preprint arXiv:1709.02314*, 2017.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. Embedding multimodal relational data for knowledge base completion. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Aditya Sharma, Partha Talukdar, et al. Towards understanding the geometry of knowledge graph embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Neural Information Processing Systems (NeurIPS)*, 2014.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Neural Information Processing Systems (NeurIPS)*, 2013.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. Compositional learning of embeddings for relation paths in knowledge base and text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, 2016.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. Cane: Context-aware network embedding for relation modeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*, 2014.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations (ICLR)*, 2015.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. Knowledge graph embedding with hierarchical relation structure. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. Data poisoning attacks on multitask relationship learning. In *AAAI Conference on Artificial Intelligence*, 2018a.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018b.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

A Appendix

We approximate the change on the score of the target triple upon applying attacks other than the $\langle s', r', o\rangle$ ones. Since each relation appears many times in the training triples, we can assume that applying a single attack will not considerably affect the relations embeddings. As a result, we just need to study the attacks in the form of $\langle s, r', o\rangle$ and $\langle s, r', o'\rangle$. Defining the scoring function as $\psi(s, r, o) = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) \cdot \mathbf{e}_o = \mathbf{z}_{s,r} \cdot \mathbf{e}_o$, we further assume that $\psi(s, r, o) = \mathbf{e}_s \cdot \mathbf{g}(\mathbf{e}_r, \mathbf{e}_o) = \mathbf{e}_s \cdot \mathbf{x}_{r,o}$.

A.1 Modifications of the Form $\langle s, r', o'\rangle$

Using similar argument as the attacks in the form of $\langle s', r', o\rangle$, we can calculate the effect of the attack, $\overline{\psi}(s, r, o) - \psi(s, r, o)$ as:

$$\overline{\psi}(s, r, o) - \psi(s, r, o) = (\overline{\mathbf{e}_s} - \mathbf{e}_s)\mathbf{x}_{s,r} \tag{10}$$

where $\mathbf{x}_{s,r} = \mathbf{g}(\mathbf{e}_r, \mathbf{e}_o)$.

We now derive an efficient computation for $(\overline{\mathbf{e}_s} - \mathbf{e}_s)$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s, r', o'\rangle)$ over $\mathbf{e}_s$ is:

$$\nabla_{e_s}\mathcal{L}(\overline{G}) = \nabla_{e_s}\mathcal{L}(G) - (1 - \varphi)\mathbf{x}_{r',o'} \tag{11}$$

where $\mathbf{x}_{r',o'} = \mathbf{g}(\mathbf{e}'_r, \mathbf{e}'_o)$, and $\varphi = \sigma(\psi(s, r', o'))$. At convergence, after retraining, we expect $\nabla_{e_s}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_s}\mathcal{L}(\overline{G})$ to get:

$$\begin{aligned} 0 \simeq & -(1 - \varphi)\mathbf{x}^{\mathsf{T}}_{r',o'} + \\ & (H_s + \varphi(1 - \varphi)\mathbf{x}^{\mathsf{T}}_{r',o'}\mathbf{x}_{r',o'})(\overline{\mathbf{e}_s} - \mathbf{e}_s) \end{aligned} \tag{12}$$

where H_s is the $d \times d$ Hessian matrix for s, i.e. second order derivative of the loss w.r.t. $\mathbf{e}_s$, computed sparsely. Solving for $\overline{\mathbf{e}_s} - \mathbf{e}_s$ gives us:

$$\begin{aligned} & \overline{\mathbf{e}_s} - \mathbf{e}_s = \\ & (1 - \varphi)(H_s + \varphi(1 - \varphi)\mathbf{x}^{\mathsf{T}}_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^{\mathsf{T}}_{r',o'} \end{aligned}$$

In practice, H_s is positive definite, making $H_s + \varphi(1 - \varphi)\mathbf{x}^{\mathsf{T}}_{r',o'}\mathbf{x}_{r',o'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\begin{aligned} & \overline{\psi}(s, r, o) - \psi(s, r, o) = \mathbf{x}_{r,o}(\overline{\mathbf{e}_s} - \mathbf{e}_s) = \\ & ((1 - \varphi)(H_s + \varphi(1 - \varphi)\mathbf{x}^{\mathsf{T}}_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^{\mathsf{T}}_{r',o'})\mathbf{x}_{r,o}. \end{aligned} \tag{13}$$

A.2 Modifications of the Form $\langle s, r', o\rangle$

In this section we approximate the effect of attack in the form of $\langle s, r', o\rangle$. In contrast to $\langle s', r', o\rangle$ attacks, for this scenario we need to consider the change in the $\mathbf{e}_s$, upon applying the attack, in approximation of the change in the score as well. Using previous results, we can approximate the $\overline{\mathbf{e}_o} - \mathbf{e}_o$ as:

$$\begin{aligned} & \overline{\mathbf{e}_o} - \mathbf{e}_o = \\ & \qquad (1 - \varphi)(H_o + \varphi(1 - \varphi)\mathbf{z}^{\mathsf{T}}_{s,r'}\mathbf{z}_{s,r'})^{-1}\mathbf{z}^{\mathsf{T}}_{s,r'} \end{aligned} \tag{14}$$

	Target Triple	CRIAGE-Add
DistMult	Brisbane Airport, isConnectedTo, Boulia Airport	Osman Ozköylü, isPoliticianOf, Boulia Airport
	Jalna District, isLocatedIn, India	United States, hasWonPrize, India
	Quincy Promes, wasBornIn, Amsterdam	Gmina Krzeszyce, hasGender, Amsterdam
	Princess Henriette, hasChild, Violante Bavaria	Al Jazira Club, playsFor, Violante Bavaria
ConvE	Brisbane Airport, isConnectedTo, Boulia Airport	Victoria Wood, wasBornIn, Boulia Airport
	National Union(Israel), isLocatedIn, Jerusalem	Sejad Halilović, isAffiliatedTo, Jerusalem
	Robert Louis, influences, David Leavitt	David Louhoungou, hasGender, David Leavitt
	Princess Henriette, hasChild, Violante Bavaria	Jonava, isAffiliatedTo, Violante Bavaria

Table 7: Top adversarial triples for target samples.

and similarly, we can approximate $\overline{\mathbf{e}_s} - \mathbf{e}_s$ as:

$$\overline{\mathbf{e}_s} - \mathbf{e}_s = (1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal \tag{15}$$

where H_s is the Hessian matrix over $\mathbf{e}_s$. Then using these approximations:

$$\mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) = \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^\intercal \mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^\intercal)$$

and:

$$(\overline{\mathbf{e}_s} - \mathbf{e}_s)\mathbf{x}_{r,\bar{o}} = ((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal)\mathbf{x}_{r,\bar{o}}$$

and then calculate the change in the score as:

$$\begin{aligned}&\overline{\psi}(s,r,o) - \psi(s,r,o) = \\ &\mathbf{z}_{s,r}.(\overline{\mathbf{e}_o} - \mathbf{e}_o) + (\overline{\mathbf{e}_s} - \mathbf{e}_s).\mathbf{x}_{r,\bar{o}} = \\ &\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^\intercal \mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^\intercal) + \\ &((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal)\mathbf{x}_{r,\bar{o}}\end{aligned} \tag{16}$$

A.3 First-order Approximation of the Change For TransE

In here we derive the approximation of the change in the score upon applying an adversarial modification for TransE [Bordes et al., 2013a]. Using similar assumptions and parameters as before, to calculate the effect of the attack, $\overline{\psi}(s,r,o)$ (where $\psi(s,r,o) = |\mathbf{e}_s + \mathbf{e}_r - \mathbf{e}_o|$), we need to compute $\overline{\mathbf{e}_o}$. To do so, we need to derive an efficient computation for $\overline{\mathbf{e}_o}$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$ over $\mathbf{e}_o$ is:

$$\nabla_{e_o}\mathcal{L}(\overline{G}) = \nabla_{e_o}\mathcal{L}(G) + (1-\varphi)\frac{\mathbf{z}_{s',r'} - \mathbf{e}_o}{\psi(s',r',o)} \tag{17}$$

where $\mathbf{z}_{s',r'} = \mathbf{e}'_s + \mathbf{e}'_r$, and $\varphi = \sigma(\psi(s',r',o))$. At convergence, after retraining, we expect $\nabla_{e_o}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_o}\mathcal{L}(\overline{G})$ to get:

$$0 \simeq (1-\varphi)\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} + (H_o - H_{s',r',o})(\overline{\mathbf{e}_o} - \mathbf{e}_o) \tag{18}$$

$$H_{s',r',o} = (1-\varphi)\varphi\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal(\mathbf{z}_{s',r'} - \mathbf{e}_o)}{\psi(s',r',o)^2} + \frac{1-\varphi}{\psi(s',r',o)} - (1-\varphi)\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal(\mathbf{z}_{s',r'} - \mathbf{e}_o)}{\psi(s',r',o)^3} \tag{19}$$

where H_o is the $d \times d$ Hessian matrix for o, i.e., second order derivative of the loss w.r.t. $\mathbf{e}_o$, computed sparsely. Solving for $\overline{\mathbf{e}_o}$ gives us:

$$\overline{\mathbf{e}_o} = -(1-\varphi)(H_o - H_{s',r',o})^{-1}\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} + \mathbf{e}_o \tag{20}$$

Then, we compute the score change as:

$$\begin{aligned}\overline{\psi}(s,r,o) &= |\mathbf{e}_s + \mathbf{e}_r - \overline{\mathbf{e}_o}| \\ &= |\mathbf{e}_s + \mathbf{e}_r + (1-\varphi)(H_o - H_{s',r',o})^{-1} \\ &\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} - \mathbf{e}_o|\end{aligned} \tag{21}$$

Calculating this expression is efficient since H_o is a $d \times d$ matrix.

A.4 Sample Adversarial Attacks

In this section, we provide the output of the CRIAGE-Add for some target triples. Sample adversarial attacks are provided in Table 7. As it shows, CRIAGE-Add attacks mostly try to change the *type* of the target triple's object by associating it with a subject and a relation that require a different entity types.

Instructions for NAACL-HLT 2019 Proceedings

Anonymous NAACL submission

Abstract

This document contains the instructions for preparing a camera-ready manuscript for the proceedings of NAACL-HLT 2019. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used for both papers submitted for review and for final versions of accepted papers. Authors are asked to conform to all the directions reported in this document.

1 Credits

This document has been adapted from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Michell and Stephanie Lukin, 2017/2018 (NA)ACL bibtex suggestions from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, those from ACL 2010 by Jing-Shing Chang and Philipp Koehn, those for ACL 2008 by JohannaD. Moore, Simone Teufel, James Allan, and Sadaoki Furui, those for ACL 2005 by Hwee Tou Ng and Kemal Oflazer, those for ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats. Those versions were written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

2 Introduction

The following instructions are directed to authors of papers submitted to NAACL-HLT 2019 or accepted for publication in its proceedings. All authors are required to adhere to these specifications. Authors are required to provide a Portable Document Format (PDF) version of their papers. **The proceedings are designed for printing on A4 paper.**

3 General Instructions

Manuscripts must be in two-column format. Exceptions to the two-column format include the title, authors' names and complete addresses, which must be centered at the top of the first page, and any full-width figures or tables (see the guidelines in Subsection 3.6). **Type single-spaced.** Start all pages directly under the top margin. See the guidelines later regarding formatting the first page. The manuscript should be printed single-sided and its length should not exceed the maximum page limit described in Section 5. Pages are numbered for initial submission. However, **do not number the pages in the camera-ready version**.

By uncommenting `\aclfinalcopy` at the top of this document, it will compile to produce an example of the camera-ready formatting; by leaving it commented out, the document will be anonymized for initial submission. When you first create your submission on softconf, please fill in your submitted paper ID where `***` appears in the `\def\aclpaperid{***}` definition at the top.

The review process is double-blind, so do not include any author information (names, addresses) when submitting a paper for review. However, you should maintain space for names and addresses so that they will fit in the final (accepted) version. The NAACL-HLT 2019 LaTeX style will create a titlebox space of 2.5in for you when `\aclfinalcopy` is commented out.

The author list for submissions should include all (and only) individuals who made substantial contributions to the work presented. Each author

Proceedings of NAACL-HLT 2019, pages 3348–3354
Minneapolis, Minnesota, June 2 - June 7, 2019.

listed on a submission to NAACL-HLT 2019 will be notified of submissions, revisions and the final decision. No authors may be added to or removed from submissions to NAACL-HLT 2019 after the submission deadline.

3.1 The Ruler

The NAACL-HLT 2019 style defines a printed ruler which should be presented in the version submitted for review. The ruler is provided in order that reviewers may comment on particular lines in the paper without circumlocution. If you are preparing a document without the provided style files, please arrange for an equivalent ruler to appear on the final output pages. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler. (LaTeX users may uncomment the `\aclfinalcopy` command in the document preamble.)

Reviewers: note that the ruler measurements do not align well with lines in the paper – this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. In most cases one would expect that the approximate location will be adequate, although you can also use fractional references (*e.g.*, the first paragraph on this page ends at mark 108.5).

3.2 Electronically-available resources

NAACL-HLT provides this description in LaTeX2e (`naaclhlt2019.tex`) and PDF format (`naaclhlt2019.pdf`), along with the LaTeX2e style file used to format it (`naaclhlt2019.sty`) and an ACL bibliography style (`acl_natbib.bst`) and example bibliography (`naaclhlt2019.bib`). These files are all available at `http://naacl2019.org/downloads/naaclhlt2019-latex.zip`. We strongly recommend the use of these style files, which have been appropriately tailored for the NAACL-HLT 2019 proceedings.

3.3 Format of Electronic Manuscript

For the production of the electronic manuscript you must use Adobe's Portable Document Format (PDF). PDF files are usually produced from LaTeX using the *pdflatex* command. If your version of LaTeX produces Postscript files, you can convert these into PDF using *ps2pdf* or *dvipdf*. On Windows, you can also use Adobe Distiller to generate PDF.

Please make sure that your PDF file includes all the necessary fonts (especially tree diagrams, symbols, and fonts with Asian characters). When you print or create the PDF file, there is usually an option in your printer setup to include none, all or just non-standard fonts. Please make sure that you select the option of including ALL the fonts. **Before sending it, test your PDF by printing it from a computer different from the one where it was created.** Moreover, some word processors may generate very large PDF files, where each page is rendered as an image. Such images may reproduce poorly. In this case, try alternative ways to obtain the PDF. One way on some systems is to install a driver for a postscript printer, send your document to the printer specifying "Output to a file", then convert the file to PDF.

It is of utmost importance to specify the **A4 format** (21 cm x 29.7 cm) when formatting the paper. When working with `dvips`, for instance, one should specify `-t a4`. Or using the command `\special{papersize=210mm,297mm}` in the latex preamble (directly below the `\usepackage` commands). Then using `dvipdf` and/or `pdflatex` which would make it easier for some.

Print-outs of the PDF file on A4 paper should be identical to the hardcopy version. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs as soon as possible.

3.4 Layout

Format manuscripts two columns to a page, in the manner these instructions are formatted. The exact dimensions for a page on A4 paper are:

- Left and right margins: 2.5 cm
- Top margin: 2.5 cm
- Bottom margin: 2.5 cm
- Column width: 7.7 cm
- Column height: 24.7 cm
- Gap between columns: 0.6 cm

Papers should not be submitted on any other paper size. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs above as soon as possible.

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word “Abstract”	12 pt	bold
section titles	12 pt	bold
document text	11 pt	
captions	10 pt	
abstract text	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 1: Font guide.

3.5 Fonts

For reasons of uniformity, Adobe’s **Times Roman** font should be used. In LaTeX2e this is accomplished by putting

```
\usepackage{times}
\usepackage{latexsym}
```

in the preamble. If Times Roman is unavailable, use **Computer Modern Roman** (LaTeX2e’s default). Note that the latter is about 10% less dense than Adobe’s Times Roman font.

3.6 The First Page

Center the title, author’s name(s) and affiliation(s) across both columns. Do not use footnotes for affiliations. Do not include the paper ID number assigned during the submission process. Use the two-column format only when you begin the abstract.

Title: Place the title centered at the top of the first page, in a 15-point bold font. (For a complete guide to font sizes and styles, see Table 1) Long titles should be typed on two lines without a blank line intervening. Approximately, put the title at 2.5 cm from the top of the page, followed by a blank line, then the author’s names(s), and the affiliation on the following line. Do not use only initials for given names (middle initials are allowed). Do not format surnames in all capitals (*e.g.*, use “Mitchell” not “MITCHELL”). Do not format title and section headings in all capitals as well except for proper names (such as “BLEU”) that are conventionally in all capitals. The affiliation should contain the author’s complete address, and if possible, an electronic mail address. Start the body of the first page 7.5 cm from the top of the page.

Command	Output	Command	Output
`{\"a}`	ä	`{\c c}`	ç
`{\^e}`	ê	`{\u g}`	ğ
``{\`i}``	ì	`{\l}`	ł
`{\.I}`	İ	`{\~n}`	ñ
`{\o}`	ø	`{\H o}`	ő
`{\'u}`	ú	`{\v r}`	ř
`{\aa}`	å	`{\ss}`	ß

Table 2: Example commands for accented characters, to be used in, *e.g.*, BibTeX names.

The title, author names and addresses should be completely identical to those entered to the electronical paper submission website in order to maintain the consistency of author information among all publications of the conference. If they are different, the publication chairs may resolve the difference without consulting with you; so it is in your own interest to double-check that the information is consistent.

Abstract: Type the abstract at the beginning of the first column. The width of the abstract text should be smaller than the width of the columns for the text in the body of the paper by about 0.6 cm on each side. Center the word **Abstract** in a 12 point bold font above the body of the abstract. The abstract should be a concise summary of the general thesis and conclusions of the paper. It should be no longer than 200 words. The abstract text should be in 10 point font.

Text: Begin typing the main body of the text immediately after the abstract, observing the two-column format as shown in the present document. Do not include page numbers.

Indent: Indent when starting a new paragraph, about 0.4 cm. Use 11 points for text and subsection headings, 12 points for section headings and 15 points for the title.

3.7 Sections

Headings: Type and label section and subsection headings in the style shown on the present document. Use numbered sections (Arabic numerals) in order to facilitate cross references. Number subsections with the section number and the subsection number separated by a dot, in Arabic numerals. Do not number subsubsections.

Citations: Citations within the text appear in parentheses as (Gusfield, 1997) or, if the author’s name appears in the text itself, as Gusfield (1997). Using the provided LaTeX style, the former is accomplished using `\cite` and the latter with `\shortcite` or `\newcite`. Collapse multi-

output	natbib	previous ACL style files
(Gusfield, 1997)	`\citep`	`\cite`
Gusfield (1997)	`\citet`	`\newcite`
(1997)	`\citeyearpar`	`\shortcite`

Table 3: Citation commands supported by the style file. The citation style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

ple citations as in (Gusfield, 1997; Aho and Ullman, 1972); this is accomplished with the provided style using commas within the `\cite` command, *e.g.*, `\cite{Gusfield:97,Aho:72}`. Append lowercase letters to the year in cases of ambiguities. Treat double authors as in (Aho and Ullman, 1972), but write as in (Chandra et al., 1981) when more than two authors are involved. Collapse multiple citations as in (Gusfield, 1997; Aho and Ullman, 1972). Also refrain from using full citations as sentence constituents.

We suggest that instead of

> "(Gusfield, 1997) showed that ..."

you use

> "Gusfield (1997) showed that ..."

If you are using the provided LaTeX and BibTeX style files, you can use the command `\citet` (cite in text) to get "author (year)" citations.

If the BibTeX file contains DOI fields, the paper title in the references section will appear as a hyperlink to the DOI, using the hyperref LaTeX package. To disable the hyperref package, load the style file with the `nohyperref` option:

```
\usepackage[nohyperref]{naaclhlt2019}
```

Digital Object Identifiers: As part of our work to make ACL materials more widely used and cited outside of our discipline, ACL has registered as a CrossRef member, as a registrant of Digital Object Identifiers (DOIs), the standard for registering permanent URNs for referencing scholarly materials. As of 2017, we are requiring all camera-ready references to contain the appropriate DOIs (or as a second resort, the hyperlinked ACL Anthology Identifier) to all cited works. Thus, please ensure that you use BibTeX records that contain DOI or URLs for any of the ACL materials that you reference. Appropriate records should be found for most materials in the current ACL Anthology at `http://aclanthology.info/`.

As examples, we cite (Goodman et al., 2016) to show you how papers with a DOI will appear in the bibliography. We cite (Harper, 2014) to show how papers without a DOI but with an ACL Anthology Identifier will appear in the bibliography.

As reviewing will be double-blind, the submitted version of the papers should not include the authors' names and affiliations. Furthermore, self-references that reveal the author's identity, *e.g.*,

> "We previously showed (Gusfield, 1997) ..."

should be avoided. Instead, use citations such as

> "Gusfield (1997) previously showed ... "

Any preliminary non-archival versions of submitted papers should be listed in the submission form but not in the review version of the paper. NAACL-HLT 2019 reviewers are generally aware that authors may present preliminary versions of their work in other venues, but will not be provided the list of previous presentations from the submission form.

Please do not use anonymous citations and do not include when submitting your papers. Papers that do not conform to these requirements may be rejected without review.

References: Gather the full set of references together under the heading **References**; place the section before any Appendices. Arrange the references alphabetically by first author, rather than by order of occurrence in the text. By using a .bib file, as in this template, this will be automatically handled for you. See the `\bibliography` commands near the end for more.

Provide as complete a citation as possible, using a consistent format, such as the one for *Computational Linguistics* or the one in the *Publication Manual of the American Psychological Association* (American Psychological Association, 1983). Use of full names for authors rather than initials is preferred. A list of abbreviations for common computer science journals can be found in the ACM *Computing Reviews* (for Computing Machinery, 1983).

The LaTeX and BibTeX style files provided roughly fit the American Psychological Association format, allowing regular citations, short citations and multiple citations as described above.

- Example citing an arxiv paper: (Rasooli and Tetreault, 2015).
- Example article in journal citation: (Ando and Zhang, 2005).
- Example article in proceedings, with location: (Borschinger and Johnson, 2011).
- Example article in proceedings, without location: (Andrew and Gao, 2007).

See corresponding .bib file for further details.

Submissions should accurately reference prior and related work, including code and data. If a piece of prior work appeared in multiple venues, the version that appeared in a refereed, archival venue should be referenced. If multiple versions of a piece of prior work exist, the one used by the authors should be referenced. Authors should not rely on automated citation indices to provide accurate references for prior and related work.

Appendices: Appendices, if any, directly follow the text and the references (but see above). Letter them in sequence and provide an informative title: **Appendix A. Title of Appendix**.

3.8 Footnotes

Footnotes: Put footnotes at the bottom of the page and use 9 point font. They may be numbered or referred to by asterisks or other symbols.[1] Footnotes should be separated from the text by a line.[2]

3.9 Graphics

Illustrations: Place figures, tables, and photographs in the paper near where they are first discussed, rather than at the end, if possible. Wide illustrations may run across both columns. Color illustrations are discouraged, unless you have verified that they will be understandable when printed in black ink.

Captions: Provide a caption for every illustration; number each one sequentially in the form: "Figure 1. Caption of the Figure." "Table 1. Caption of the Table." Type the captions of the figures and tables below the body, using 10 point text. Captions should be placed below illustrations. Captions that are one line are centered (see Table 1). Captions longer than one line are left-aligned (see Table 2). Do not overwrite the default caption sizes. The naaclhlt2019.sty file is compatible with the caption and subcaption packages; do not add optional arguments.

3.10 Accessibility

In an effort to accommodate people who are color-blind (as well as those printing to paper), grayscale readability for all accepted papers will be encouraged. Color is not forbidden, but authors should ensure that tables and figures do not rely solely on color to convey critical distinctions. A simple criterion: All curves and points in your figures should be clearly distinguishable without color.

4 Translation of non-English Terms

It is also advised to supplement non-English characters and terms with appropriate transliterations and/or translations since not all readers understand all such characters and terms. Inline transliteration or translation can be represented in the order of: original-form transliteration "translation".

5 Length of Submission

The NAACL-HLT 2019 main conference accepts submissions of long papers and short papers. Long papers may consist of up to eight (8) pages of content plus unlimited pages for references. Upon acceptance, final versions of long papers will be given one additional page – up to nine (9) pages of content plus unlimited pages for references – so that reviewers' comments can be taken into account. Short papers may consist of up to four (4) pages of content, plus unlimited pages for references. Upon acceptance, short papers will be given five (5) pages in the proceedings and unlimited pages for references. For both long and short papers, all illustrations and tables that are part of the main text must be accommodated within these page limits, observing the formatting instructions given in the present document. Papers that do not conform to the specified length and formatting requirements are subject to be rejected without review.

NAACL-HLT 2019 does encourage the submission of additional material that is relevant to the reviewers but not an integral part of the paper. There are two such types of material: appendices,

---

[1] This is how a footnote should appear.

[2] Note the line separating the footnotes from the text.

which can be read, and non-readable supplementary materials, often data or code. Do not include this additional material in the same document as your main paper. Additional material must be submitted as one or more separate files, and must adhere to the same anonymity guidelines as the main paper. The paper must be self-contained: it is optional for reviewers to look at the supplementary material. Papers should not refer, for further detail, to documents, code or data resources that are not available to the reviewers. Refer to Appendix A and Appendix B for further information.

Workshop chairs may have different rules for allowed length and whether supplemental material is welcome. As always, the respective call for papers is the authoritative source.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

Preparing References:

Include your own bib file like this: `\bibliographystyle{acl_natbib}` `\bibliography{naaclhlt2019}`

where `naaclhlt2019` corresponds to a naaclhlt2019.bib file.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Benjamin Borschinger and Mark Johnson. 2011. A particle filter algorithm for Bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 10–18, Canberra, Australia.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, page 1. Dublin City University and Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

A Appendices

Appendices are material that can be read, and include lemmas, formulas, proofs, and tables that are not critical to the reading and understanding of the paper. Appendices should be **uploaded as supplementary material** when submitting the paper for review. Upon acceptance, the appendices come after the references, as shown here. Use `\appendix` before any appendix section to switch the section numbering over to letters.

B Supplemental Material

Submissions may include non-readable supplementary material used in the work and described in the paper. Any accompanying software and/or data should include licenses and documentation of research review as appropriate. Supplementary material may report preprocessing decisions, model parameters, and other details necessary for the replication of the experiments reported in the paper. Seemingly small preprocessing decisions can sometimes make a large difference in performance, so it is crucial to record such decisions to precisely characterize state-of-the-art methods.

Nonetheless, supplementary material should be supplementary (rather than central) to the paper. **Submissions that misuse the supplementary material may be rejected without review.** Supplementary material may include explanations or details of proofs or derivations that do not fit into the paper, lists of features or feature templates, sample inputs and outputs for a system, pseudo-code or source code, and data. (Source code and data should be separate uploads, rather than part of the paper).

The paper should not rely on the supplementary material: while the paper may refer to and cite the supplementary material and the supplementary material will be available to the reviewers, they will not be asked to review the supplementary material.

Transferable Neural Projection Representations

Chinnadhurai Sankar
Mila, Université de Montréal*
Montreal, QC, Canada
`chinnadhurai@gmail.com`

Sujith Ravi
Google Research
Mountain View, CA, USA
`sravi@google.com`

Zornitsa Kozareva
Google
Mountain View, CA, USA
`zornitsa@kozareva.com`

Abstract

Neural word representations are at the core of many state-of-the-art natural language processing models. A widely used approach is to pretrain, store and look up word or character embedding matrices. While useful, such representations occupy huge memory making it hard to deploy on-device and often do not generalize to unknown words due to vocabulary pruning.

In this paper, we propose a skip-gram based architecture coupled with Locality-Sensitive Hashing (LSH) projections to learn efficient dynamically computable representations. Our model does not need to store lookup tables as representations are computed on-the-fly and require low memory footprint. The representations can be trained in an unsupervised fashion and can be easily transferred to other NLP tasks. For qualitative evaluation, we analyze the nearest neighbors of the word representations and discover semantically similar words even with misspellings. For quantitative evaluation, we plug our transferable projections into a simple LSTM and run it on multiple NLP tasks and show how our transferable projections achieve better performance compared to prior work.

1 Introduction

Pre-trained word representations are at the core of many neural language understanding models. Among the most popular and widely used word embeddings are word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and ELMO (Peters et al., 2018). The biggest challenge with word embedding is that they require lookup and a large memory footprint, as we have to store one entry (d-dim vector) per word and it blows up.

In parallel, the tremendous success of deep learning models and the explosion of mobile, IoT devices coupled together with the growing user privacy concerns have led to the need for deploying deep learning models on-device for inference. This has led to new research in compressing large and complex deep learning models for low power on-device deployment. Recently, (Ravi and Kozareva, 2018) developed an on-device neural text classification model. They proposed to reduce the memory footprint of large neural networks by replacing the input word embeddings with projection based representations. (Ravi and Kozareva, 2018) used n-gram features to generate binary LSH (Charikar, 2002) randomized projections on the fly surpassing the need to store word emebdding tables and reducing the memory size. The projection models reduce the memory occupied by the model from $O(|V|)$ to $O(n_{\mathbb{P}})$, where $|V|$ refers to the vocabulary size and $n_{\mathbb{P}}$ refers to number of projection operations (Ravi, 2017). Two key advantages of the projection based representations over word embeddings are: (1) they are fixed and have low memory size; (2) they can handle out of vocabulary words. However, the projections in (Ravi and Kozareva, 2018) are static and currently do not leverage pre-training on large unsupervised corpora, which is an important property to make the projections transferable to new tasks.

In this paper, we propose to combine the best of both worlds by learning transferable neural projection representations over randomized LSH projections. We do this by introducing new neural architecture inspired by the skip gram model of (Mikolov et al., 2013) and combined with a deep MLP plugged on top of LSH projections. In order to make this model train better, we introduce new regularizing loss function, which minimizes the cosine similarities of the words within a mini-batch. The loss function is critical for generalization.

In summary, our model (1) requires a fixed and low memory footprint, (2) can handle out of vo-

---

[*] Work done during internship at Google.

Proceedings of NAACL-HLT 2019, pages 3355–3360
Minneapolis, Minnesota, June 2 - June 7, 2019.

cabulary words and misspellings, (3) captures semantic and syntactic properties of words; (4) can be easily plugged to other NLP models and (5) can support training with data augmentation by perturbing characters of input words. To validate the performance of our approach, we conduct a qualitative analysis of the nearest neighbours in the learned representation spaces and a quantitative evaluation via similarity, language modeling and NLP tasks.

2 Neural Projection Model

We propose a novel model (NP-SG) to learn compact neural representations that combines the benefit of representation learning approaches like skip-gram model with efficient LSH projections that can be computed on-the-fly.

2.1 Vanilla Skip-Gram Model

In the skip-gram model (Mikolov et al., 2013), we learn continuous distributed representations for words in a large fixed vocabulary, $\mathbb{V}$ to predict the context words surrounding them in documents. We maintain an embedding look up table, $v(w) \in \mathbb{R}^d$ for every word, $w \in \mathbb{V}$.

For each word, w_t in the training corpus of size T, the set of context words $\mathbb{C}_t = \{w_{t-W_t}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+W_t}\}$ includes W_t words to the left and right of w_t respectively. W_t is the window size randomly sampled from the set $\{1, 2, \ldots, N\}$, where N is the maximum window size. Given a pair of words, $\{w_c, w_t\}$, the probability of w_c being within the context window of w_t is given by equation 1.

$$\begin{aligned} \mathrm{P}(w_c|w_t) &= \sigma(v'(w_c)^\intercal v(w_t)) \\ &= \frac{1}{1+\exp(-v'(w_c)^\intercal v(w_t))} \end{aligned} \tag{1}$$

where v, v' are input and context embedding look up tables.

2.2 Neural Projection Skip-Gram (NP-SG)

In the neural projection approach, we replace the input embedding look up table, $v(w)$ in equation 1 with a deep n-layer MLP over the binary projection, $\mathbb{P}(w)$ as shown equation 2.

$$v_{\mathbb{P}}(w) = \mathbb{N}(f_n(\mathbb{P}(w))) \tag{2}$$

where $v_{\mathbb{P}}(w) \in \mathbb{R}^d$, f_n is a n-layer deep neural network encoder with $ReLU$ non-linear activations after each layer except for the last layer as shown in Figure 1. $\mathbb{N}$ refers to a normalization applied to the final layer of f_n. We experimented with Batch-normalization, L2-normalization and layer normalization; batch-normalization works the best.

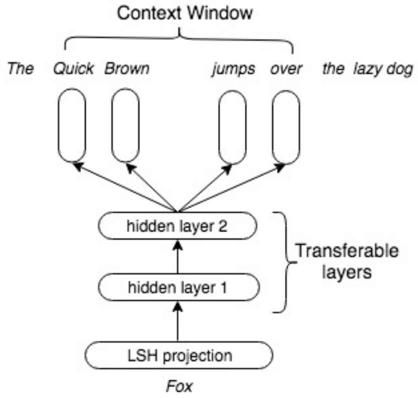


Figure 1: Neural Projection Skip-gram (NP-SG) model

The binary projection $\mathbb{P}(w)$ is computed using locality-sensitive projection operations (Ravi, 2017) which can be performed on-the-fly (i.e., without any embedding look up) to yield a fixed, low-memory footprint binary vector. Unlike (Ravi and Kozareva, 2018) which uses *static* projections to encode the entire input text and learn a classifier, NP-SG creates a *trainable* deep projection representation for words using LSH projections over character-level features combined with contextual information learned via the skip-gram architecture.

2.3 Training NP-SG Model

We follow a similar approach as Mikolov et al. (2013) and others for training our neural projection skip-gram model (NP-SG). We define the training objective to maximize the probability of predicting the context words given the current word. Formally, the model tries to learn the word embeddings by maximizing the objective, $J(\theta)$ known as negative sampling (NEG), given by equation 3.

$$J(\theta) = \sum_{t=1}^{T} \sum_{w_c \in \mathbb{C}_t} J_{w_t, w_c}(\theta) \tag{3}$$

$$\begin{aligned} J_{w_t,w_c}(\theta) &= \log(\mathrm{P}(w_c|w_t)) \\ &+ \sum_{i=1, w_i \sim \mathrm{P}_n(w)}^{k} \log(1 - \mathrm{P}(w_i|w_t)) \end{aligned} \tag{4}$$

where k is the number of randomly sampled words from the training corpus according to the noise distribution, $\mathrm{P}_n(w) \propto U(w)^{3/4}$, where $U(w)$ is the unigram distribution of the training corpus.

Model improvements: Training an NP-SG model as is, though efficient, may not lead to highly discriminative representations. During training, we noticed that the word representations, $v_{\mathbb{P}}(w)$ were getting projected in a narrow sub-space where the cosine similarities of all the words in the dataset were too close to 1.0. This made the convergence slower and led to poor generalization.

2.4 Discriminative NP-SG Models

To encourage the word representations to be more spaced out in terms of the cosine similarities, we introduce an additional explicit regularizing L2-loss function. With the assumption that the words in each mini-batch are randomly sampled, we add a L2-loss over the cosine similarities between all the words within a mini-batch, as shown in equation 6.

$$Loss = J(\theta) + \mathrm{L}_2^{cs}(\mathbf{w}_{mb}) \tag{5}$$

$$\mathrm{L}_2^{cs}(\mathbf{w}_{mb}) = \lambda \cdot \| \{\mathrm{CS}(w_i, w_j) \,|\, i, j \in [0, mb)\} \|_2^2 \tag{6}$$

where $\mathrm{CS}(w_i, w_j)$ refers to the cosine similarity between w_i and w_j, mb refers to the mini-batch size and $\mathbf{w}_{mb}$ refers to the words in the mini-batch. We enforce this using a simple outerproduct trick. We extract the cosine-similarities between all the words within a mini-batch in a single shot by computing the outer-product of the L_2 row normalized word representations corresponding to each mini-batch $\hat{v}_{\mathbb{P}}(\mathbf{w}_{mb})$, as shown in equation 7.

$$\mathrm{L}_2^{cs}(\mathbf{w}_{mb}) = \frac{\lambda}{2} \cdot \| \,\mathrm{Flatten}(\hat{v}_{\mathbb{P}}(\mathbf{w}_{mb}) \cdot \hat{v}_{\mathbb{P}}(\mathbf{w}_{mb})^{\intercal}) \,\|_2^2 \tag{7}$$

2.5 Improved NP-SG Training

Since the NP-SG model does not have a fixed vocabulary size, we can be flexible and leverage a lot more information during training compared to standard skip-gram models which require vocabulary pruning for feasibility.

To improve training for NP-SG model, we augment the dataset with inputs words after applying character level perturbations to them. The perturbations are such a way that they are commonly occurring misspellings in documents. We mainly experiment with three types of pertub operation APIs (Gao et al., 2018).

- *insert(word, n)* : We randomly choose *n* chars from the character vocabulary and insert them randomly into the input *word*. We ignore the locations of first and last character in the word for the *insert* operation. Example transformation: $sample \rightarrow samnple$.
- *swap(word, n)* : We randomly swap the location of two characters in the word *n* times. As with the *insert* operation, we ignore the first and last character in the word for the *swap* operation. Example transformation: $sample \rightarrow sapmle$.
- *duplicate(word, n)* : We randomly duplicate a character in the word by *n* times. Example transformation: $sample \rightarrow saample$.

We would like to note that the perturbation operations listed above are not exhaustive and we plan to experiment with more operations in the future.

3 Training Setup

3.1 Dataset

We train our skipgram models on the wikipedia data XML dump, ***enwik9***[1]. We extract the normalized English text from the XML dump using the Matt Mahoneys pre-processing perl script[2]. We fix the vocabulary to the top $100k$ frequently occurring words. We sub-sample words in the training corpus, dropping them with probability, $\mathrm{P}(w) = 1 - \sqrt{t/freq(w)}$, where $freq(w)$ is the frequency of occurrence of w in the corpus and we set the threshold, t to 10^{-5}. We perturb the input words with a probability of 0.4 using a randomly chosen perturbation described in Section 2.5.

3.2 Implementation Details

We fix the number of random projections to 80 and the projection dimension to 14. We use a 2-layer MLP (sizes: $[2048, 100]$) regularized with dropout (with probability of 0.65) and weight decay (regularization parameter of 0.0005) to transform the binary random projections to continuous word representation. For the vanilla skipgram model, we fix the embedding size to 100. For both models, we use 25 negative samples for the NEG loss. We learn the parameters using the Adam optimizer (Kingma and Ba, 2014) with a default learning rate of 0.001, clipping the gradients which have a norm larger than 5.0. We initialize the weights of the MLP using Xavier initialization, and output embeddings

---

[1] http://mattmahoney.net/dc/enwik9.zip

[2] http://mattmahoney.net/dc/textdata

Dataset	*SG (10M)*	*NP-SG* (w/oOP)	*NP-SG* (1M)	*NP-SG* (2M)	*NP-SG* (4M)
EN-MTurk-287	0.5409	0.0107	**0.5629**	**0.5517**	**0.5494**
EN-WS-353-ALL	0.5930	0.0710	0.4891	0.5215	0.5370
EN-WS-353-REL	0.5359	0.0203	0.4956	**0.5746**	**0.5671**
EN-WS-353-SIM	0.6242	0.1043	0.4994	0.5116	0.5111
EN-RW-STANFORD	0.1505	0.0401	0.0184	0.0375	0.0835
EN-VERB-143	0.2452	0.0730	0.1333	0.1500	0.2108

Table 1: Similarity Tasks: # of params, $100k$ vocabulary size for skipgram baseline, 100 embedding size.

uniformly random in the range $[-1.0, 1.0]$. We use a batch size of 1024 in all our experiments. We found that $\lambda = 0.01$ for the outerproduct loss to be working better after experimenting with other values. Training time for our model was around 0.85 times that of the skipgram model. Both the models were trained for 10 epochs.

4 Experiments

We show both qualitative and quantitative evaluation on multiple tasks for the NP-SG model.

4.1 Qualitative Evaluation and Results

Table 2 shows the nearest neighbors produced by NP-SG for select words. Independent of whether it is an original or misspelled word, our NP-SG model accurately retrieves relevant and semantically similar words.

Word	*Nearest neighbours*
king	reign, throne, kings, knights, vii, regent
kingg	vii, younger, peerage, iv, tiberius, frederick
woman	man, young, girl, child, girls, women
wwoamn	man, herself, men, couple, herself, alive
city	town, village, borough, township, county
ciity	town, village, borough, county, unorganized
time	few, times, once, entire, prominence, since
tinme	times, once, takes, taken, another, only
zero	two, three, seven, one, eight, four
zzero	two, three, five, six, seven, four

Table 2: Sampled nearest neighbors for NP-SG.

4.2 Quantitative Evaluation and Results

We evaluate our NP-SG model on similarity, language modeling and text classification tasks. Similarity tests the ability to capture words, while language modeling and classification warrant the ability to transfer the neural projections.

4.2.1 Similarity Task

We evaluate our NP-SG word representations on 4 different widely used benchmark datasets for measuring similarities.

Dataset: *MTurk-287* (Radinsky et al., 2011) has 287 pairs of words and was constructed by crowdsourcing the human similarity ratings using Amazon Mechanical Turk. *WS353* (Finkelstein et al., 2001) has 353 pairs of similar English words rated by humans and is further split into *WS353-SIM*. *WS353-REL* (Agirre et al., 2009) captures different types of similarities and relatedness. *RW-STANFORD* (Luong et al., 2013) has 2034 rare word pairs sampled from different frequency bins. **Evaluation:** For all the datasets, we compute the Spearmans rank correlation coefficient between the rankings computed by skip-gram models (baseline SG and NP-SG) and the human rankings. We use cosine similarity metric to measure word similarity. **Results:** Table 1 shows that NP-SG, with significantly smaller number of parameters comes close to the skip-gram model (SG) and even outperforms it with 2.5x-10x compression. NP-SG gets better representations even with misspellings which cannot be handled by vanilla SG.

It is interesting to note that the vanilla skip-gram model does well on *WS353-SIM* compared to *WS353-REL*. This behavior is reversed in our NP-SG model, which indicates that it captures meronym-holonym relationships better than the vanilla skip-gram model. Although NP-SG handles out of vocabulary words in the form of misspellings, it needs further improvement for rare word similarity task. We plan to improve it by including context word n-gram features in the LSH projection function, allowing NP-SG to also leverage information from the context words in the case of rare words and provide word sense disambiguation.

4.2.2 Language Modeling

We applied NP-SG to language modeling task on the Penn Treebank (PTB)(Taylor et al., 2003) dataset. We consider a single layer LSTM with hidden size of 2048 for the language model task. With the input embedding size of 200, we observed a perplexity of ≈ 120 on the test set after training for 5 epochs. We replace the input embeddings in the LSTM with transferable encoder layer of the NP-SG model. We train the LSTMs with and without pretrained initializations. Since we observed convergence issues with the single layer NP-SG representation, we considered 2-layer MLP with layer sizes (1024, 256) for the NP-SG representations. We found that while the model without pretrained NP-SG layer got stuck at a perplexity of around 300, the pretrained version converged to a perplexity of 140, comparable to the embedding based network. We leave the analysis of the impact of the deeper NP-SG layers to the future work.

4.2.3 Text Classification

For the text classification evaluations, we used two different tasks and datasets. For the dialog act classification task, we used the MRDA dataset from the ICSI Meeting Recorder Dialog Act Corpus (Adam et al., 2003). MRDA is a multiparty dialog annotated with 5 dialog act tags. For the question classification task, we used the TREC dataset (Lin and Katz, 2006). The task is given a question to predict the most relevant category.

We trained a single layer LSTM (hidden size: 256) with and without the pretrained NP-SG layers. Overall, we observed accuracy improvements of $+5.7\%$ and $+3.75\%$ compared to baseline models without pretrained NP-SG initializations on TREC and MRDA respectively.

5 Conclusion

In this paper, we introduced a new neural architecture (NP-SG), which learns transferable word representations that can be efficiently and dynamically computed on device without any embedding look up. We proposed an unsupervised method to train the new architecture and learn more discriminative word representations. We compared the new model with a skip-gram approach and showed qualitative and quantitative comparisons on multiple language tasks. The evaluations show that our NP-SG model learns better representations even with misspellings and reaches competitive results with skip-gram on similarity tasks, even outperforming with 2.5x-10x fewer parameters.

Acknowledgments

The authors would like to thank the Google Expander team for many helpful discussions.

References

Janin Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The icsi meeting corpus. In *Proceedings of the 5TH SIGdial Workshop on Discourse and Dialogue*, pages 364–367.

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 19–27.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 406–414.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pages 50–56.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jimmy J. Lin and Boris Katz. 2006. Building a reusable test collection for question answering. *JASIST*, 57(7):851–861.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference* *on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 337–346.

Sujith Ravi. 2017. Projectionnet: Learning efficient on-device deep networks using neural projections. *CoRR*, abs/1708.00630.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 804–810.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The penn treebank: An overview.

Semantic Role Labeling with Associated Memory Network

Chaoyu Guan[†], **Yuhao Cheng**[†], **Hai Zhao**[†*]
[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[†]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
`{351549709,cyh958859352}@sjtu.edu.cn`
`zhaohai@cs.sjtu.edu.cn`

Abstract

Semantic role labeling (SRL) is a task to recognize all the predicate-argument pairs of a sentence, which has been in a performance improvement bottleneck after a series of latest works were presented. This paper proposes a novel syntax-agnostic SRL model enhanced by the proposed associated memory network (AMN), which makes use of inter-sentence attention of label-known associated sentences as a kind of memory to further enhance dependency-based SRL. In detail, we use sentences and their labels from train dataset as an associated memory cue to help label the target sentence. Furthermore, we compare several associated sentences selecting strategies and label merging methods in AMN to find and utilize the label of associated sentences while attending them. By leveraging the attentive memory from known training data, Our full model reaches state-of-the-art on CoNLL-2009 benchmark datasets for syntax-agnostic setting, showing a new effective research line of SRL enhancement other than exploiting external resources such as well pre-trained language models.

1 Introduction

Semantic role labeling (SRL) is a task to recognize all the predicate-argument pairs of a given sentence and its predicates. It is a shallow semantic parsing task, which has been widely used in a series of natural language processing (NLP) tasks, such as information extraction (Liu et al., 2016) and question answering (Abujabal et al., 2017).

Generally, SRL is decomposed into four classification subtasks in pipeline systems, consisting of predicate identification, predicate disambiguation, argument identification, and argument classification. In recent years, great attention (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017, 2018a,b) has been turned to deep learning method, especially Long Short-term Memory (LSTM) network for learning with automatically extracted features. (Zhou and Xu, 2015) proposed the first end-to-end recurrent neural network (RNN) to solve the SRL task. (Marcheggiani et al., 2017) studied several predicate-specified embedding and decoding methods. (He et al., 2017) delivered a full study on the influence of RNN training and decoding strategies. Whether to use the syntactic information for SRL is also studied actively (He et al., 2017, 2018b).

Since the recent work of (Marcheggiani et al., 2017), which surprisingly shows syntax-agnostic dependency SRL for the first time can be rival of syntax-aware models, SRL has been more and more formulized into standard sequence labeling task on a basis of keeping syntax unavailable. A series of work on SRL received further performance improvement following this line through further refining neural model design (He et al., 2018a). Different from all previous work, we propose to introduce an associated memory network which builds memory from known data through the inter-sentence attention to enhance syntax-agnostic model even further.

Inspired by the observation that people always refer to other similar problems and their solutions when dealing with a problem they have never seen, like query in their memory, we want to utilize similar known samples which include the associated sentences and their annotated labels to help model label target sentence. To reach such a goal, we adopt a memory network component, and use inter-sentence attention to fully exploit the information in memory.

---

[*]Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. U1836222 and No. 61733011) and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of NAACL-HLT 2019, pages 3361–3371
Minneapolis, Minnesota, June 2 - June 7, 2019.

Based on Memory Network (Weston et al., 2014; Sukhbaatar et al., 2015), (Miller et al., 2016) proposed Key-Value Memory Network (KV-MemNN) to solve Question Answering problem and gain large progress. Our proposed method is similar to KV-MemNN, but with a different definition of key-value and different information distilling process. Thus, we propose a carefully designed inter-sentence attention mechanism to handle it.

Recently, there are also some attempts to make use of attention mechanism in SRL task. (Tan et al., 2018; Strubell et al., 2018) focus on self-attention, which only uses the information of the input sentence as the source of attention. (Cai et al., 2018) makes use of biaffine attention (Dozat and Manning, 2017) for decoding in SRL, which was the current state-of-the-art (SOTA) in CoNLL-2009 benchmark as this work was embarking. Different from all previous work, we utilize inter-sentence attention to help model leverage associated information from other known sentences in the memory.

To our best knowledge, this is the first time to use memory network in the SRL task. Our evaluation on CoNLL-2009 benchmarks shows that our model outperforms or reaches other syntax-agnostic models on English, and achieves competitive results on Chinese, which indicates that memory network learning from known data is indeed helpful to SRL task.

There are several SRL annotation conventions, such as PropBank (Bonial et al., 2012) and FrameNet (Baker et al., 1998). This paper focuses on the former convention. Under PropBank convention, there are two role representation forms, which are span-based SRL, such as CoNLL 2005 and CoNLL 2012 shared tasks, and dependency-based SRL, such as CoNLL 2009 shared task. The former uses span to represent argument, while the latter uses the headword of the span to represent the argument. As the latter has been more actively studied due to dependency style SRL for convenient machine learning, we will focus on dependency SRL only in this work.

Given a sentence **S**, the goal of dependency SRL task is to find all the predicate-argument pairs (p, a). The following shows an example sentence with semantic role labels marked in subscripts.

She$_{A0}$ has **lost**$_{v}$ **it**$_{A1}$ **just**$_{ARGM-MNR}$ as quickly.

Here, *v* means the predicate, *A0* means the agent, *A1* means the patient and *ARGM-MNR* means how an action *v* is performed.

In the rest of this paper, we will describe our model in Section 2. Then, the experiment set-up and results are given in Section 3. Related works about SRL and attention mechanism will be given in Section 4. Conclusions and future work are drawn in Section 5.

2 Model

An SRL system usually consists of four pipeline modules: predicate identification and disambiguation, argument identification and classification. Following most of previous work, we focus on the last two steps in standard SRL task: argument identification and classification. The predicate identification subtask is not needed in CoNLL-2009 shared task[1], and we follow previous work (He et al., 2018b) to handle the predicate disambiguation subtask. This work will only focus on the argument labeling subtask through sequence labeling formalization. We first describe our base model in Section 2.1. Then we introduce the proposed associated memory network including the inter-sentence attention design and label merging strategies in Section 2.2. The full model architecture is shown in Figure 1.

2.1 Base Model

Word Embedding

We use the concatenation of the following embeddings as the representation for every word. (1) Random-initialized word embedding $x_i^{re} \in \mathbb{R}^{d_{re}}$ (2) GloVe (Pennington et al., 2014) word embedding $x_i^{pe} \in \mathbb{R}^{d_{pe}}$ pre-trained on 6B tokens (3) Random-initialized part-of-speech (POS) tag embedding $x_i^{pos} \in \mathbb{R}^{d_{pos}}$ (4) Random-initialized lemma embedding $x_i^{le} \in \mathbb{R}^{d_{le}}$ (5) Contextualized word embedding derived by applying fully connected layer on ELMo embedding $x_i^{ce} \in \mathbb{R}^{d_{ce}}$ (Peters et al., 2018), and (6) Random-initialized predicate specified flag embedding $x_i^{pred} \in \mathbb{R}^{d_{pred}}$. The final representation of each word is:

$$x_i = x_i^{re} \circ x_i^{pe} \circ x_i^{pos} \circ x_i^{le} \circ x_i^{ce} \circ x_i^{pred}$$

where $\circ$ stands for concatenation operator.

---

[1] In CoNLL-2009 task, the predicates information is already identified when testing.

BiLSTM Encoder

LSTM network is known to handle the dependency over long sentence well, and can effectively model the context information when encoding. Therefore, we leverage a stacked BiLSTM network $LSTM_e$ to be our encoder. It takes word embedding sequence $\mathbf{x} = [x_i]_{i=1}^{n_\mathbf{S}}$ of sentence $\mathbf{S} = [w_i]_{i=1}^{n_\mathbf{S}}$ as input ($n_\mathbf{S}$ is the length of sentence), and outputs two different hidden states $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ for word w_i by processing the sequence in forward and backward directions. The final contextual representation of word w_i is the concatenation of two hidden states $h_i = \overrightarrow{h_i} \circ \overleftarrow{h_i}$.

Then, we use a final softmax layer after the BiLSTM encoding to predict the label of each word.

2.2 Associated Memory Network

Using the base model as backbone, we introduce an associated memory network (AMN) component for further performance improvement. The proposed AMN memorizes known associated sentences and their labels, then the useful clue in the memory will be delivered to the SRL module through an inter-sentence mechanism. AMN processing includes three steps, associated sentence selection, inter-sentence attention and label merging.

Associated Sentence Selection

We aim to utilize the associated sentences and their labels to help our model label the target sentences. For the sake of fairness, we only use the sentences in train dataset as our source. However, it is impossible to attend all the sentences in train dataset because of the extremely high computational and memory cost. Therefore, we propose a filter to select the most useful sentences from the given dataset (train dataset in this paper) when given the label-unknown sentence **S**.

The filter algorithm is straightforward. First, We compute the distance of every two sentences. Then, we sort all the sentences in train dataset according to their distances with the target sentence **S**, and select top m sentences $\{\mathbf{A}_j\}_{j=1}^{m}$ with the minimum distances and their label sequences $\{\mathbf{L}_j\}_{j=1}^{m}$ as our associated attention. m is the memory size.

As for the computation of distance between two sentences, we formally consider three types of distances, which are *edit distance (ED)*, *word moving distance (WMD)* and *smooth inverse frequency distance (SD)*, plus *random distance (RD)* as baseline. These distances are defined as follows,

- *edit distance* This method uses the edit distance of the POS tag sequences of two sentences as the distance value.
- *word moving distance* Following (Kusner et al., 2015), this method takes word moving distance of two sentences[2].
- *smooth inverse frequency distance* Following (Arora et al., 2017), we use Euclidean distance between the SIF embedding of two sentences as the distance value.
- *random distance* This method returns a random value for distance computation thus lead to selecting sentences randomly in the train dataset.

Inter-sentence Attention

This part aims to attain the inter-sentence attention matrix, which can be also regarded as the core memory part of the AMN. The input sentence **S** and associated sentences $\{\mathbf{A}_j\}_{j=1}^{m}$ first go through a stacked BiSLTM network $LSTM_a$ to encode the sentence-level information to each word representation[3]:

$$\mathbf{S}' = LSTM_a(\mathbf{S})$$

$$\mathbf{A}'_j = LSTM_a(\mathbf{A}_j) \;\; j \in \{1, 2, ..., m\}$$

where $\mathbf{S}' = [x'_i]_{i=1}^{n_\mathbf{S}}$ and $\mathbf{A}'_j = [x'_{j,k}]_{k=1}^{n_j}$ are the lists of new word representations, with each word representation is a vector $x' \in \mathbb{R}^{d_a}$, where d_a is the size of hidden state in $LSTM_a$.

Then, for each associated sentence $\mathbf{A}'_j$, we multiply it with the input sentence representation $\mathbf{S}'$ to get the raw attention matrix M_j^{raw}.

$$M_j^{raw} = \mathbf{S}'\mathbf{A}_j'^T$$

Every element $M_j^{raw}(i,k) = x'_i \cdot x'^T_{j,k}$ can be regarded as an indicator of similarity between the i^{th} word in input sentence $\mathbf{S}'$ and the k^{th} word in associated sentence $\mathbf{A}'_j$.

Finally, we perform softmax operation on every row in M_j^{raw} to normalize the value so that it can

---

[2] In this paper, we use relaxed word moving distance (rwmd) for efficiency

[3] Here we abuse the symbol $\mathbf{S}$ and $\mathbf{A}_j$ for meaning both the word sequence $[w_i]$ and the embedded sequence $[x_i]$

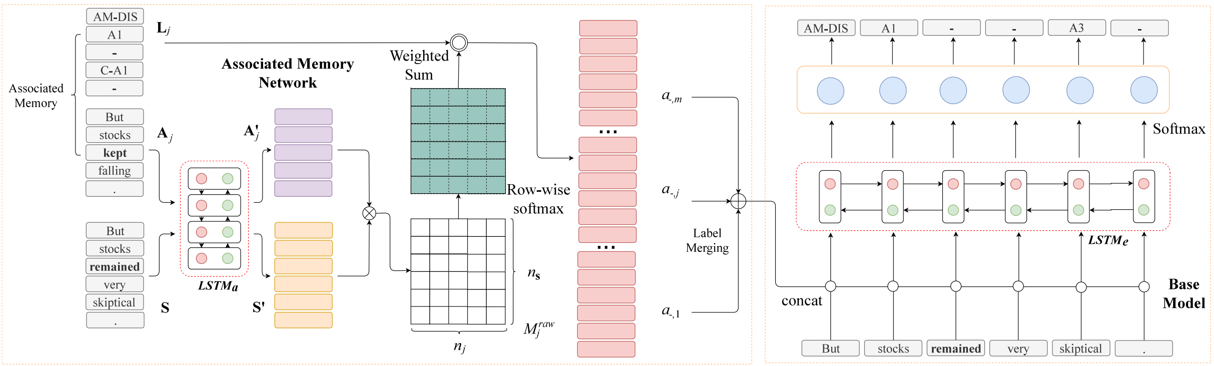


Figure 1: Semantic role labeling with associated memory network, where S is the input sentence with its length n_{S}. A_j is the j^{th} associated sentence of S with its label sequence L_j and its length n_j. S' and A'_j are the result of $LSTM_1$ with S and A_j as input respectively. d_{ae} is the dimension of argument embedding. M_j^{raw} is the raw attention matrix of A_j and S. $a_{-,j} = [a_{1,j}, a_{2,j}, ..., a_{n_{\mathrm{S}},j}]$ is the associated-sentence-specified attention embedding.

be considered as probability from input sentence $\mathbf{S}$ to associated sentence $\mathbf{A}_j$.

$$\alpha_{i,j} = f([M_j^{raw}(i,1)..., M_j^{raw}(i,n_j)])$$

$$M_j = [\alpha_{1,j}, \alpha_{2,j}, ..., \alpha_{n_{\mathbf{S}},j}]$$

where $f(\cdot)$ stands for softmax function. $\alpha_{i,j}$ can be regarded as probability vector indicating the similarity between the i^{th} word in sentence $\mathbf{S}$ and every word in the associated sentence $\mathbf{A}'_j$.

Label Merging

In order to utilize the labels $\{\mathbf{L}_j\}_{j=1}^m$ of the associated sentences during decoding, a label merging needs to be done.

We use randomly initialized argument embedding $x^{ae} \in \mathbb{R}^{d_{ae}}$ to embed each argument label. Therefore, the label sequence $\mathbf{L}_j$ of associated sentence $\mathbf{A}_j$ can be written as $\mathbf{L}_j = [x_{j,k}^{ae}]_{k=1}^{n_j}$.We treat the probability vector $\alpha_{i,j}$ as weight to sum all the elements in L_j to get the associated-sentence-specified argument embedding $a_{i,j}$, which represents the attention embedding of word $w_i \in \mathbf{S}$ calculated from the j^{th} associated sentence $\mathbf{A}_j$ and label $\mathbf{L}_j$.

$$a_{i,j} = \alpha_{i,j} \cdot \mathbf{L}_j^T = \sum_{k=1}^{n_j} \alpha_{i,j}(k) x_{j,k}^{ae}$$

Because the associated sentences are different, the overall contributions of these argument embeddings should be different. We let the model itself learn how to make use of these argument embeddings. Following attention combination mechanism from (Libovickỳ and Helcl, 2017), we consider four ways to merge the label information.

1) Concatenation All the associated argument embedding are concatenated as the final attention embeddings.

$$a_i = a_{i,1} \circ a_{i,2} \circ ... \circ a_{i,m}$$

2) Average The average value of all the associated argument embeddings is used as the final attention embedding.

$$a_i = \frac{1}{m} \sum_{j=1}^m a_{i,j}$$

3) Weighted Average The weighted average of all the associated argument embedding is used as the final attention embedding. We calculate the mean value of every raw similarity matrix M_j^{raw} to indicate the similarity between input sentence $\mathbf{S}$ and associated sentence $\mathbf{A}_j$, and we use the softmax function to normalize them to get a probability vector β indicating the similarity of input sentence $\mathbf{S}$ towards all the associated sentences $\{\mathbf{A}_j\}_{j=1}^m$.

$$\beta = f([g(M_1^{raw}), ..., g(M_m^{raw})])$$

where $f(\cdot)$ stands for softmax function and $g(\cdot)$ represents the mean function. Then, we use the probability vector β as weight to sum all the associated-sentence-specified attention embedding $a_{i,j}$ to get the final attention embedding a_i of the i^{th} word w_i in input sentence $\mathbf{S}$.

$$a_i = \sum_{j=1}^m \beta(j) a_{i,j}$$

4) Flat This method does not use $a_{i,j}$ information. First, we concatenate all the raw similarity matrix M_j^{raw} along the row.

$$M^{raw} = [M_1^{raw}, M_2^{raw}, ..., M_m^{raw}]$$

Then, we perform softmax operation on every row in M^{raw} to normalize the value so that it can be considered as probability from input sentence $\mathbf{S}$ to all associated sentences $\mathbf{A}_j$.

$$\gamma_i = f([M_{i,1}^{raw}, M_{i,2}^{raw} ..., M_{i,n_{all}}^{raw}])$$

Name	**Meaning**	**Value**
d_{re}	random word embedding	100
d_{pe}	pre-trained word embedding	100
d_{pos}	POS embedding	32
d_{le}	lemma embedding	100
d_{ce}	contextualized embedding	128
d_{pred}	flag embedding	16
d_{ae}	argument embedding	128
m	memory size	4
k_e	#$LSTM_e$ layers	2
k_a	#$LSTM_a$ layers	3
d_e	$LSTM_e$ hidden state	512
d_a	$LSTM_a$ hidden state	512
r_d	dropout rate	0.1
l_r	learning rate	0.001

Table 1: Hyper-parameter settings (signal #x means number of x).

where $f(\cdot)$ stands for softmax operation. $n_{all} = \sum_{j=1}^{m} n_j$ is the total length of all m associated sentences.

We also concatenate the associated label information, and use γ_i as weight to sum the concatenated label sequence as final attention embedding.

$$\mathbf{L} = [\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_j], \quad a_i = \gamma_i \cdot \mathbf{L}^T$$

After we have the final attention embedding a_i, we concatenate it with word embedding x_i as the input of the BiLSTM encoder $LSTM_e$.

3 Experiments

We conduct experiments on CoNLL-2009 (Hajič et al., 2009) English and Chinese dataset. We use the standard training, development and test data split provided by CoNLL-2009 shared task. The word lemma, word POS are the predicted ones given in CoNLL-2009 dataset. Adam optimizer (Kingma and Ba, 2014) is used for training to minimize the categorical cross entropy loss. All the hyper-parameters we use are listed in Table 1. All parameters are learned during training, and are randomly initialized except the pre-trained GloVe (Pennington et al., 2014) word embeddings.

For English, We independently determine the best distance calculating method and the best merging method one after another. First, we select a distance according to the results on development set and then we determine the merging method with the selected distance method. At last we explore the impact of memory size. For Chinese, we obtain the result with similar parameters as for the best model in English. The English and Chinese GloVe word embeddings are both trained on Wikipedia. The pretrained English ELMo model is from (Peters et al., 2018), and the Chinese one is from (Che et al., 2018), which is hosted at (Fares et al., 2017). The model is trained for maximum 20 epochs for the nearly best model based on development set results. We re-run our model using different initialized parameters for 4 times and report the average performance[4].

System (syntax-aware single)	P	R	F_1
(Zhao et al., 2009a)	-	-	86.2
(Zhao et al., 2009c)	-	-	85.4
(FitzGerald et al., 2015)	-	-	86.7
(Roth and Lapata, 2016)	88.1	85.3	86.7
(Marcheggiani and Titov, 2017)	89.1	86.8	88.0
(He et al., 2018b)	89.7	**89.3**	89.5
(Li et al., 2018)	**90.3**	89.3	**89.8**
System (syntax-aware ensemble)	P	R	F_1
(FitzGerald et al., 2015)	-	-	87.7
(Roth and Lapata, 2016)	90.3	85.7	87.9
(Marcheggiani and Titov, 2017)	**90.5**	**87.7**	**89.1**
System (syntax-agnostic single)	P	R	F_1
(Marcheggiani et al., 2017)	88.7	86.8	87.7
(He et al., 2018b)	89.5	87.9	88.7
(Cai et al., 2018)	89.9	**89.2**	**89.6**
(Li et al., 2018)	89.5	87.9	88.7
Ours (+ AMN + ELMo)	**90.0**	**89.2**	**89.6**

Table 2: Results on CoNLL-2009 English in-domain (WSJ) test set.

System (syntax-aware single)	P	R	F_1
(Zhao et al., 2009a)	-	-	74.6
(Zhao et al., 2009c)	-	-	73.3
(FitzGerald et al., 2015)	-	-	75.2
(Roth and Lapata, 2016)	76.9	73.8	75.3
(Marcheggiani and Titov, 2017)	78.5	75.9	77.2
(He et al., 2018b)	**81.9**	76.9	79.3
(Li et al., 2018)	80.6	**79.0**	**79.8**
System (syntax-aware ensemble)	P	R	F_1
(FitzGerald et al., 2015)	-	-	75.5
(Roth and Lapata, 2016)	79.7	73.6	76.5
(Marcheggiani and Titov, 2017)	**80.8**	**77.1**	**78.9**
System (syntax-agnostic single)	P	R	F_1
(Marcheggiani et al., 2017)	79.4	76.2	77.7
(He et al., 2018b)	**81.7**	76.1	78.8
(Cai et al., 2018)	79.8	78.3	79.0
Ours (+ AMN + ELMo)	80.0	**79.4**	**79.7**

Table 3: Results on CoNLL-2009 English out-of-domain (Brown) test set.

3.1 Results

For the predicate disambiguation, we use the same one from (He et al., 2018b) with the precisions

---

[4] Our implementation is publicly available at `https://github.com/Frozenmad/AMN_SRL`.

System (syntax-aware single)	P	R	F_1
(Zhao et al., 2009a)	80.4	75.2	77.7
(Roth and Lapata, 2016)	83.2	75.9	79.4
(Marcheggiani and Titov, 2017)	84.6	80.4	82.5
(He et al., 2018b)	84.2	**81.5**	82.8
(Li et al., 2018)	**84.8**	81.2	**83.0**
System (syntax-agnostic single)	P	R	F_1
(Marcheggiani et al., 2017)	83.4	79.1	81.2
(He et al., 2018b)	84.5	79.3	81.8
(Cai et al., 2018)	84.7	**84.0**	**84.3**
Ours (+ AMN + ELMo)	**85.0**	82.6	83.8

Table 4: Results on CoNLL-2009 Chinese test set.

of 95.01% and 95.58% on development and test sets. We compare our full model (using *edit distance* and *average method*) with the reported state-of-the-art models on both English and Chinese dataset. The results are in Tables 2, 3 and 4.

For English in-domain test, our model outperforms the syntax-agnostic model in (He et al., 2018b), whose architecture is quite similar to our base model. Our model achieves 89.6% in F_1 score, which is the same with current SOTA syntax-agnostic model (Cai et al., 2018). Besides, our result is competitive with existing syntax-aware and better than ensemble models.

The advantage is more salient on English out-of-domain test set. The F_1 score of our model is 79.7%, which is 0.7% higher than the current SOTA syntax-agnostic model (Cai et al., 2018). The result is also competitive with the best syntax-aware model (Li et al., 2018). The comparisons show that the proposed model has a greater generalization ability.

For Chinese, starting with the similar parameters as for the best model in English, we find that attending 5 associated sentences shows a better result on Chinese. Our model achieves 83.8% F_1 score, outperforming (He et al., 2018b) with an improvement of 2.0% in F_1 score. Our result is also competitive with that of (Cai et al., 2018).

Note that our method is not conflict with the one in (Cai et al., 2018), which leverages biaffine attention (Dozat and Manning, 2017) for decoding. However, due to experiment cycle, we are not able to combine these two methods together. We will leave the combination as future work.

In the following part, we conduct several ablation studies on our model. All the experiments are re-run 2-4 times and the average values are reported on CoNLL-2009 English development set.

System	P	R	F_1
WMD (Kusner et al., 2015)	**89.1**	87.1	88.1
SD (Arora et al., 2017)	88.5	**87.5**	88.0
RD	**89.1**	87.2	88.1
Base Model	88.7	86.9	87.8
ED	89.0	**87.5**	**88.3**

Table 5: Ablations about distance on CoNLL-2009 English development set. ED means edit distance, WMD means word moving distance, SD means SIF distance, RD means random distance.

System	P	R	F_1
Concatenation	88.9	86.6	87.7
Average	**89.0**	**87.5**	**88.3**
Weighted Average	88.7	87.4	88.1
Flat	88.4	86.9	87.7
None	88.7	86.9	87.8

Table 6: Ablations about label merging method on CoNLL-2009 English development set.

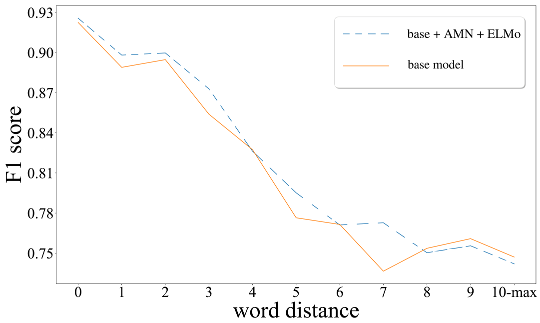


Figure 2: The comparison of our full model and base model with distance increases.

3.2 Ablation on Distance Method

Table 5 shows the performance of different distance calculating methods. All models use *average method* for label merging, and the memory size m is set to 4. It can be observed from Table 5 that edit distance performs best among all the distance calculating methods, with 88.3% F_1 score. All the distance calculating methods have surpassed the base model, showing that the proposed AMN is effective. Note that even the random distance model performs better than the base model, with an improvement of 0.3% in F_1 score, which shows that the proposed AMN can effectively extract useful information from even poorly related sentences. Besides, associated sentence selection methods based on word embeddings like WMD and SD have similar performance with random distance (RD), which shows simple word embedding may not be good enough signal indicator to measure semantic structure similarity in SRL task. On the contrary, we may also try to explain why even the random distance selection may work to some extent. As sentences always have core arguments label such as A0, A1 and A2, associated sentences even from random selection may also have such labels, which makes them helpful to enhance SRL over these labels. This may explain why our model with randomly selected associated sentences can distinguish core arguments better.

3.3 Ablation on Label Merging Method

Table 6 shows the performance of different label merging methods. All models use *edit distance* with 4 associated sentences. The result shows that *Average* label merging strategy gives the best performance, achieving 88.3% in F_1 score with an improvement of 0.5% compared to the baseline model.

Note that our weighted average model does not outperform the average model, which is a surprise to us. We speculate that the current weight calculation method needs to be more improved to fit the concerned task.

3.4 ELMo vs. AMN

Table 7 compares the performance contribution from ELMo and AMN. Our model can achieve better performance only using informative clue from training set in terms of AMN design, rather than focusing on external resource like ELMo. However, even though our baseline SRL has been enhanced by ELMo, it can still receive extra performance improvement from the propose AMN. Note that our enhancement from the proposed AMN keeps effective when ELMo is included (a 0.5% enhancement on baseline over the 0.3% enhancement on ELMo baseline)

3.5 Ablation on Memory Size

We show the effect of different memory size in Figure 3. Note that more associated sentences means more cost on time and space. We test memory size m from 2 to 6 (which reaches the limit under experiment setting in 11G GPU). We also fit the measured points with a linear function (the blue line in Figure 3). The performance of our model has a general trend of increasing when the memory size becomes larger, which shows the potential of the proposed AMN.

System (syntax-aware)	P	R	F_1
(He et al., 2018b)	86.8	85.8	86.3
(He et al., 2018b) + ELMo	87.7	87.0	87.3
(Li et al., 2018)	87.7	86.7	87.2
(Li et al., 2018) + ELMo	**89.2**	**87.6**	**88.4**
Ours (syntax-agnostic)	P	R	F_1
Base	86.9	85.0	86.0
Base + AMN	86.9	85.6	86.3
Base + ELMo	88.7	86.9	87.8
Ours + AMN + ELMo	**89.0**	**87.5**	**88.3**

Table 7: AMN vs. ELMo, the performance comparison on English development set.

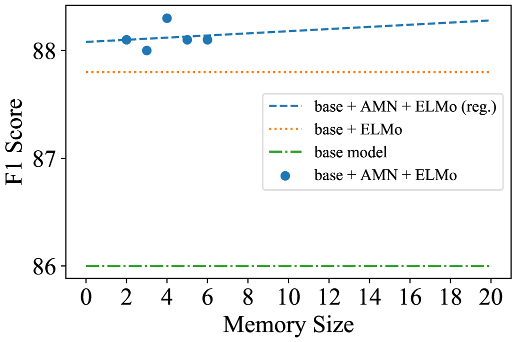
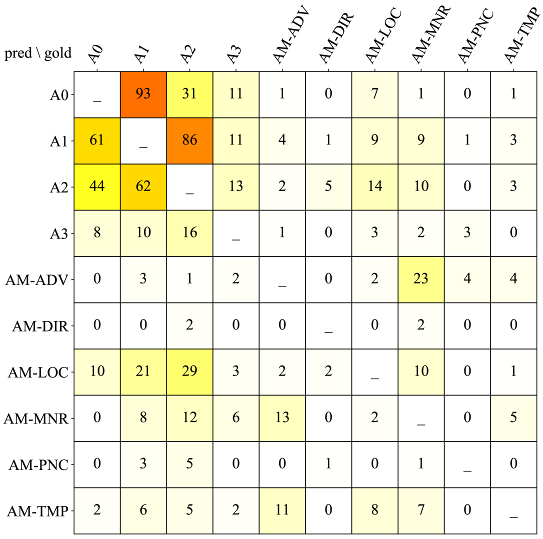


Figure 3: Model performance on English development set with different memory sizes, in which *base+AMN+ELMo* (*reg.*) indicates the general trend of our base model enhanced by the AMN when the memory size is enlarged.

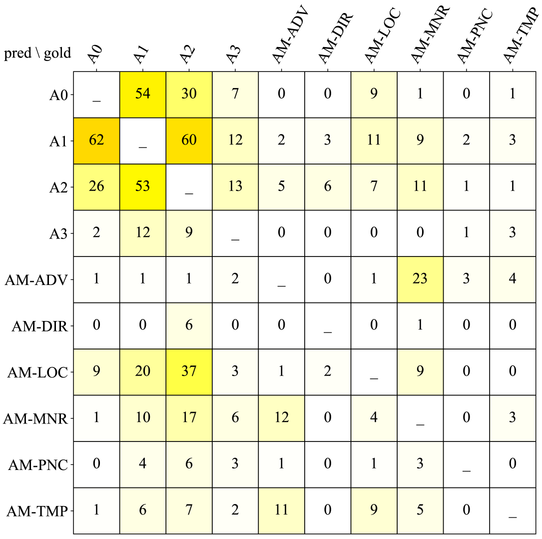
3.6 Analysis on Confusion Matrix

To further understand the advance of the proposed method, we conduct an error type break down analysis. Figures 4 and 5 show the confusion matrices of labeling errors in the baseline model and our model on development set, respectively. We only show the main and most informative type of arguments. Every number in these figures stands for the times of occurrence. Comparing these two confusion matrixes shows that the proposed model makes fewer mistakes between core arguments such as $A0$, $A1$, and $A2$. AMN indeed helps when labeling them. It is also noted that, as in (He et al., 2017; Tan et al., 2018), the model still easily confuses ARG2 with AM-DIR, AM-LOC and AM-MNR.



pred \ gold	A0	A1	A2	A3	AM-ADV	AM-DIR	AM-LOC	AM-MNR	AM-PNC	AM-TMP
A0	_	93	31	11	1	0	7	1	0	1
A1	61	_	86	11	4	1	9	9	1	3
A2	44	62	_	13	2	5	14	10	0	3
A3	8	10	16	_	1	0	3	2	3	0
AM-ADV	0	3	1	2	_	0	2	23	4	4
AM-DIR	0	0	2	0	0	_	0	2	0	0
AM-LOC	10	21	29	3	2	2	_	10	0	1
AM-MNR	0	8	12	6	13	0	2	_	0	5
AM-PNC	0	3	5	0	0	1	0	1	_	0
AM-TMP	2	6	5	2	11	0	8	7	0	_

Figure 4: Confusion matrix for labeling errors in base model.



pred \ gold	A0	A1	A2	A3	AM-ADV	AM-DIR	AM-LOC	AM-MNR	AM-PNC	AM-TMP
A0	_	54	30	7	0	0	9	1	0	1
A1	62	_	60	12	2	3	11	9	2	3
A2	26	53	_	13	5	6	7	11	1	1
A3	2	12	9	_	0	0	0	0	1	3
AM-ADV	1	1	1	2	_	0	1	23	3	4
AM-DIR	0	0	6	0	0	_	0	1	0	0
AM-LOC	9	20	37	3	1	2	_	9	0	0
AM-MNR	1	10	17	6	12	0	4	_	0	3
AM-PNC	0	4	6	3	1	0	1	3	_	0
AM-TMP	1	6	7	2	11	0	9	5	0	_

Figure 5: Confusion matrix for labeling errors in proposed model.

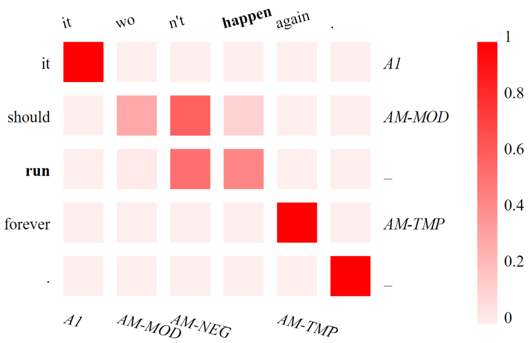


Figure 6: Visualization of similar matrix M. The input sentence is at the left of the matrix, with its golden argument label at the right. The associated sentence is at the top of the matrix, with its golden argument label at the bottom. Their predicate is bolded.

3.7 Analysis of Performance on Distance

We compare the performance concerning with the distance of argument and predicate on our best model and base model in Figure 2, from which we can observe that our model performs better nearly at any distance.

3.8 Case Study on AMN

To explore how the AMN works in the model, we visualize the similarity matrix M of some sentences from development set in Figure 6. The input sentence is

it$_{A1}$ **should**$_{AM-MOD}$ **run**$_{v}$ **forever**$_{AM-TMP}$.

And the associated sentence is

it$_{A1}$ **wo**$_{AM-MOD}$ **n't**$_{AM-NEG}$ **happen**$_{v}$ **again**$_{AM-TMP}$.

The current predicates are **run**, **happen** respectively. The visualization shows that inter-sentence attention can find and align the word in the similar context correctly, which shows that the proposed AMN is reasonable and effective.

4 Related Works

Early attempts (Pradhan et al., 2005; Zhao et al., 2009a,b, 2013; Roth and Woodsend, 2014) to the SRL task were mainly linear classifiers. The main focus was how to find proper feature templates that can best describe the sentences. (Pradhan et al., 2005) utilized a SVM classifier with rich syntactic features. (Toutanova et al., 2008) took the structural constraint into consideration by using a global reranker. (Zhao et al., 2009c) adopted a maximum entropy model with large scale feature template selection. (Roth and Woodsend, 2014) explored the distributional word representations as new feature to gain more powerful models.

Recently, a great attention has been paid on neural networks. (Zhou and Xu, 2015) proposed an end-to-end model using stacked BiLSTM network combined with CRF decoder without any syntactic input. (Marcheggiani et al., 2017) explored the predicate-specified encoding and decoding and also provided a syntax-agnostic LSTM model. (He et al., 2017) followed (Zhou and Xu, 2015) and analyzed all popular methods for initialization and regularization in LSTM network.

By considering that our approach also bor-rows power from the memory, the proposed inter-sentence attention in our AMN shares features with memory networks, which was proposed in (Weston et al., 2014) with motivation that memory may reduce the long-term forgetting issues. (Sukhbaatar et al., 2015) and (Miller et al., 2016) later further improved this work. However, we use quite different mechanisms to store the memory, and the effectiveness of our model needs a carefully designed attention mechanism to handle the sequence-level information distilling.

Attention mechanism was first used by (Bahdanau et al., 2014) in machine translation. Recently, (Tan et al., 2018) and (Strubell et al., 2018) proposed to use self-attention mechanism in SRL task. (Cai et al., 2018) leveraged the biaffine attention (Dozat and Manning, 2017) for better decoding performance. Different from all the existing work, we instead introduce an inter-sentence attention to further enhance the current state-of-the-art SRL.

5 Conclusions and Future Work

This paper presents a new alternative improvement on strong SRL baselines. We leverage memory network which seeks power from known data, the associated sentences, and thus is called associated memory network (AMN). The performance of our model on CoNLL-2009 benchmarks shows that the proposed AMN is effective on SRL task.

As to our best knowledge, this is the first attempt to use memory network in SRL task. There is still a large space to explore along this research line. For example, our weighted average method may need more carefully improved. Our model can be built over the biaffine attention which has been verified effective in (Cai et al., 2018)[5], and the encoder in our model can be improved with more advanced forms such as Transformer (Vaswani et al., 2017). At last, as this work is done on a basis of quite limited computational resources, only one piece of nVidia 1080Ti (11G graphic memory), much plentiful available computational resource will greatly enable us to explore more big model setting (i.e., larger memory size m) for more hopefully better performance improvement.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 1191–1200.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING)*, volume 1.

Claire Bonial, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. 2012. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntax-agnostic over syntax-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (CoNLL)*, pages 2753–2765.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 271–276.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970.

---

[5] As this paper is submitting, we get to know the work (Li et al., 2019), which has taken both strengths of biaffine and ELMo. We leave the verification of our proposed method over this new strong baseline in the future.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task*, pages 1–18.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 364–369.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 473–483.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 2061–2071.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 957–966.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2401–2411.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Jindřich Libovickỳ and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 196–202.

Feifan Liu, Jinying Chen, Abhyuday Jagannatha, and Hong Yu. 2016. Learning for biomedical information extraction: Methodological review of recent advances. *arXiv preprint arXiv:1606.07993*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*, pages 411–420.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1400–1409.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 2227–2237.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 581–588.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1192–1202.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 407–413.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5027–5038.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, volume 2, pages 2440–2448.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4929–4936.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 6000–6010.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–66.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of nombank and propbank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 30–39.

Hai Zhao, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009c. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task*, pages 55–60.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, volume 1, pages 1127–1137.

Better, Faster, Stronger Sequence Tagging Constituent Parsers

David Vilares
Universidade da Coruña, CITIC
Departamento de Computación
A Coruña, Spain
david.vilares@udc.es

Mostafa Abdou, Anders Søgaard
University of Copenhagen
Department of Computer Science
Copenhagen, Denmark
{abdou,soegaard}@di.ku.dk

Abstract

Sequence tagging models for constituent parsing are faster, but less accurate than other types of parsers. In this work, we address the following weaknesses of such constituent parsers: (a) high error rates around closing brackets of long constituents, (b) large label sets, leading to sparsity, and (c) error propagation arising from greedy decoding. To effectively close brackets, we train a model that learns to switch between tagging schemes. To reduce sparsity, we decompose the label set and use multi-task learning to jointly learn to predict sublabels. Finally, we mitigate issues from greedy decoding through auxiliary losses and sentence-level fine-tuning with policy gradient. Combining these techniques, we clearly surpass the performance of sequence tagging constituent parsers on the English and Chinese Penn Treebanks, and reduce their parsing time even further. On the SPMRL datasets, we observe even greater improvements across the board, including a new state of the art on Basque, Hebrew, Polish and Swedish.[1]

1 Introduction

Constituent parsing is a core task in natural language processing (NLP), with a wide set of applications. Most competitive parsers are slow, however, to the extent that it is prohibitive of downstream applications in large-scale environments (Kummerfeld et al., 2012). Previous efforts to obtain speed-ups have focused on creating more efficient versions of traditional shift-reduce (Sagae and Lavie, 2006; Zhang and Clark, 2009) or chart-based parsers (Collins, 1997; Charniak, 2000). Zhu et al. (2013), for example, presented a fast shift-reduce parser with transitions learned by a SVM classifier. Similarly, Hall et al. (2014) introduced a fast GPU implementation for Petrov and Klein (2007), and Shen et al. (2018) significantly improved the speed of the Stern et al. (2017) greedy top-down algorithm, by learning to predict a list of syntactic distances that determine the order in which the sentence should be split.

In an alternative line of work, some authors have proposed new parsing paradigms that aim to both reduce the complexity of existing parsers and improve their speed. Vinyals et al. (2015) proposed a machine translation-inspired sequence-to-sequence approach to constituent parsing, where the input is the raw sentence, and the 'translation' is a parenthesized version of its tree. Gómez-Rodríguez and Vilares (2018) reduced constituent parsing to sequence tagging, where only n tagging actions need to be made, and obtained one of the fastest parsers to date. However, the performance is well below the state of the art (Dyer et al., 2016; Stern et al., 2017; Kitaev and Klein, 2018a).

Contribution We first explore different factors that prevent sequence tagging constituent parsers from obtaining better results. These include: high error rates when long constituents need to be closed, label sparsity, and error propagation arising from greedy inference. We then present the technical contributions of the work. To effectively close brackets of long constituents, we combine the relative-scale tagging scheme used by Gómez-Rodríguez and Vilares (2018) with a secondary top-down absolute-scale scheme. This makes it possible to train a model that learns how to switch between two encodings, depending on which one is more suitable at each time step. To reduce label sparsity, we recast the constituent-parsing-as-sequence-tagging problem as multi-task learning (MTL) (Caruana, 1997), to decompose a large label

---

[1] After this paper was submitted, Kitaev and Klein (2018b) have improved our results using their previous self-attentive constituent parser (Kitaev and Klein, 2018a) and BERT representations (Devlin et al., 2018) as input to their system. We will acknowledge these results in the Experiments section.

Proceedings of NAACL-HLT 2019, pages 3372–3383
Minneapolis, Minnesota, June 2 - June 7, 2019.

space and also obtain speed ups. Finally, we mitigate error propagation using two strategies that come at no cost to inference efficiency: auxiliary tasks and policy gradient fine-tuning.

2 Preliminaries

We briefly introduce preliminaries that we will build upon in the rest of this paper: encoding functions for constituent trees, sequence tagging, multi-task learning, and reinforcement learning.

Notation We use w=$[w_0, w_1, ..., w_n]$ to refer to a raw input sentence and bold style lower-cased and math style upper-cased characters to refer to vectors and matrices, respectively (e.g. $\mathbf{x}$ and $\mathbf{W}$).

2.1 Constituent Parsing as Sequence Tagging

Gómez-Rodríguez and Vilares (2018) define a linearization function of the form $\Phi_{|w|} : T_{|w|} \rightarrow L^{(|w|-1)}$ to map a phrase structure tree with $|w|$ words to a sequence of labels of length $|w| - 1$.[2] For each word w_t, the function generates a label $l_t \in L$ of the form l_t=(n_t, c_t, u_t), where:

- n_t encodes the number of ancestors in common between between w_t and w_{t+1}. To reduce the number of possible values, n_t is encoded as the relative variation in the number of common ancestors with respect to n_{t-1}.
- c_t encodes the lowest common ancestor between w_t and w_{t+1}.
- u_t contains the unary branch for w_t, if any.

Figure 1 explains the encoding with an example.

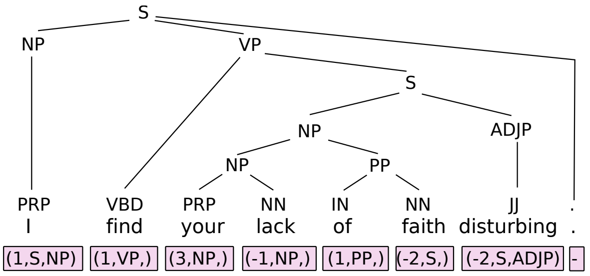


Figure 1: A constituent tree linearized as by Gómez-Rodríguez and Vilares (2018).

2.2 Sequence Tagging

Sequence tagging is a structured prediction task that generates an output label for every input token. Long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) are a popular architecture for such tasks, often giving state-of-the-art performance (Reimers and Gurevych, 2017; Yang and Zhang, 2018).

Tagging with LSTMs In LSTMs, the prediction for the ith element is conditioned on the output of the previous steps. Let LSTM$_\theta(\mathbf{x}_{1:n})$ be a parametrized function of the network, where the input is a sequence of vectors $\mathbf{x}_{1:n}$, its output is a sequence of hidden vectors $\mathbf{h}_{1:n}$. To obtain better contextualized hidden vectors, it is possible to instead use bidirectional LSTMS (Schuster and Paliwal, 1997). First, a LSTM$^l_\theta$ processes the tokens from left-to-right and then an independent LSTM$^r_\theta$ processes them from right-to-left. The ith final hidden vector is represented as the concatenation of both outputs, i.e. BILSTM$_\theta(\mathbf{x}, i)$ = LSTM$^l_\theta(\mathbf{x}_{[1:i]}) \circ$ LSTM$^r_\theta(\mathbf{x}_{[|\mathbf{x}|:i]})$. BILSTMs can be stacked in order to obtain richer representations. To decode the final hidden vectors into discrete labels, a standard approach is to use a feed-forward network together with a softmax transformation, i.e. $P(y|\mathbf{h}_i) = softmax(W \cdot \mathbf{h}_i + \mathbf{b})$. We will use the BILSTM-based model by Yang and Zhang (2018), for direct comparison against Gómez-Rodríguez and Vilares (2018), who use the same model. As input, we will use word embeddings, PoS-tag embeddings and a second word embedding learned by a character-based LSTM layer. The model is optimized minimizing the categorical cross-entropy loss, i.e. $\mathcal{L} = -\sum log(P(y|\mathbf{h}_i))$. The architecture is shown in Figure 2.

2.3 Multi-task Learning

Multi-task learning is used to solve multiple tasks using a single model architecture, with task-specific classifier functions from the outer-most representations (Caruana, 1997; Collobert and Weston, 2008). The benefits are intuitive: sharing a common representation for different tasks acts as a generalization mechanism and allows to address them in a parallel fashion. The *hard-sharing* strategy is the most basic MTL architecture, where the internal representation is fully shared across all tasks. The approach has proven robust for a number of NLP tasks (Bingel and Søgaard, 2017) and comes with certain guarantees if a common, op-

---

[2] They (1) generate a dummy label for the last word and (2) pad sentences with a beginning- and end-of-sentence tokens.

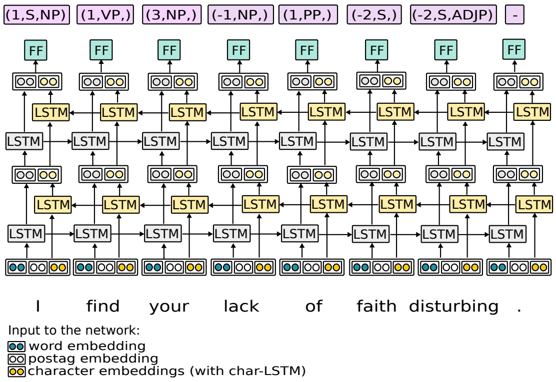


Figure 2: The baseline architecture used in this work. The input to the network is a concatenation of word embeddings, PoS-tag embeddings and a second word embedding learned by a character-based LSTM layer.

timal representation exists (Baxter, 2000). Dong et al. (2015) use it for their multilingual machine translation system, where the encoder is a shared gated recurrent neural network (Cho et al., 2014) and the decoder is language-specific. Plank et al. (2016) also use a hard-sharing setup to improve the performance of BILSTM-based PoS taggers. To do so, they rely on *auxiliary tasks*, i.e, tasks that are not of interest themselves, but that are co-learned in a MTL setup with the goal of improving the network's performance on the main task(s). We will introduce auxiliary tasks for sequence tagging constituent parsing later on in this work. A MTL architecture can also rely on *partial sharing* when the different tasks do not fully share the internal representations (Duong et al., 2015; Rei, 2017; Ruder et al., 2019) and recent work has also shown that hierarchical sharing (e.g. low-level task outputs used as input for higher-level ones) could be beneficial (Søgaard and Goldberg, 2016; Sanh et al., 2018).

2.4 Policy Gradient Fine-tuning

Policy gradient (PG) methods are a class of reinforcement learning algorithms that directly learn a parametrized policy, by which an agent selects actions based on the gradient of a scalar performance measure with respect to the policy. Compared to other reinforcement learning methods, PG is well-suited to NLP problems due to its appealing convergence properties and effectiveness in high-dimensional spaces (Sutton and Barto, 2018).

Previous work on constituent parsing has employed PG methods to mitigate the effect of exposure bias, finding that they function as a model-agnostic substitute for dynamic oracles (Fried and Klein, 2018). Similarly, Le and Fokkens (2017) apply PG methods to Chen and Manning (2014)'s transition-based dependency parser to reduce error propagation. In this work, we also employ PG to fine-tune models trained using supervised learning. However, our setting (sequence tagging) has a considerably larger action space than a transition parser. To deal with that, we will adopt a number of variance reduction and regularization techniques to make reinforcement learning stable.

3 Methods

We describe the methods introduced in this work, motivated by current limitations of existing sequence tagging models, which are first reviewed. The source code can be found as a part of `https://github.com/aghie/tree2labels`.

3.1 Motivation and Analysis

For brevity, we limit this analysis to the English Penn Treebank (PTB) (Marcus et al., 1993). We reproduced the best setup by Gómez-Rodríguez and Vilares (2018), which we are using as baseline, and run the model on the development set. We below show insights for the elements of the output tuple (n_t, c_t, u_t), where n_t is the number of levels in common between w_t and w_{t+1}, c_t is the non-terminal symbol shared at that level, and u_t is a leaf unary chain located at w_t.

High error rate on closing brackets We first focus on predicting relative tree levels (n_t). See Figure 3 for F-scores over n_t labels. The sparsity on negative n_ts is larger than for the positive ones, and we see that consequently, the performance is also significantly worse for negative n_t values, and performance worsens with higher negative values. This indicates that the current model cannot effectively identify the end of long constituents. This is a known source of error for shift-reduce or chart-based parsers, but in the case of sequence tagging parsers, the problem seems particularly serious.

Sparsity The label space is large and sparse: the output labels are simply the possible values in the tuple (n_t, c_t, u_t). An analysis over the PTB training set shows a total of 1423 labels, with 58% of them occurring 5 or less times. These infrequent cases might be difficult to predict, even if some of the elements of the tuple are common.

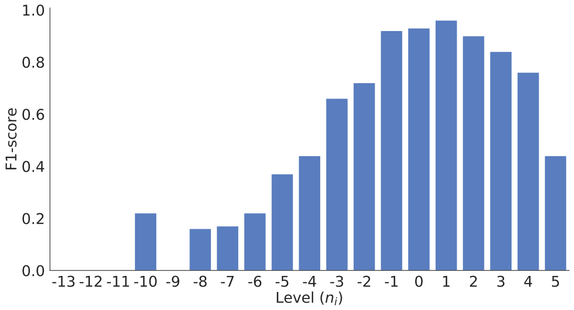


Figure 3: F-score for n_t labels on the PTB dev set using Gómez-Rodríguez and Vilares (2018).

Greedy decoding Greedy decoding is prone to issues such as error propagation. This is a known source of error in transition-based dependency parsing (Qi and Manning, 2017); in contrast with graph-based parsing, in which parsing is reduced to global optimization over edge-factored scores (McDonald et al., 2005).

In the case of BILSTM-based sequence tagging parsers, for a given word w_t, the output label as encoded by Gómez-Rodríguez and Vilares (2018) only reflects a relation between w_t and w_{t+1}. We hypothesize that even if the hidden vector representations are globally contextualized over the whole sequence, the intrinsic locality of the output label also turns into error propagation and consequently causes a drop in the performance. These hypotheses will be tested in §4. In particular, we will evaluate the impact of the different methods intended to perform structured inference (§3.4).

3.2 Dynamic Encodings

Gómez-Rodríguez and Vilares (2018) encode the number of common ancestors n_t, from the output tuple (n_t, c_t, u_t), as the variation with respect to n_{t-1}. We propose instead to encode certain elements of a sentence using a secondary linearization function. The aim is to generate a model that can dynamically switch between different tagging schemes at each time step t to select the one that represents the relation between w_t and w_{t+1} in the most effective way.

On the one hand, the relative-scale encoding is effective to predict the beginning and the end of short constituents, i.e. when a short constituent must be predicted ($|n_t| \leq 2$). On the other hand, with a relative encoding scheme, the F-score was low for words where the corresponding n_t has a large negative value (as showed in Figure 3). This matches a case where a long constituent must be closed: w_t is located at a deep level in the tree and will only (probably) share a few ancestors with w_{t+1}. These configurations are encoded in a more sparse way by a relative scheme, as the n_t value shows a large variability and it depends on the depth of the tree in the current time step. We can obtain a compressed representation of these cases by using a *top-down absolute scale* instead, as any pair of words that share the same m top levels will be equally encoded. The absolute scale becomes however sparse when predicting deep levels. Figure 4 illustrates the strengths and weaknesses of both encodings with an example, and how a dynamically encoded tree helps reduce variability on n_t values.

In our particular implementation, we will be using the following setup:

- $\Phi_{|w|} : T_{|w|} \rightarrow L^{|w|-1}$, the relative-scale encoding function, is used by default.
- $\Omega_{|w|} : T_{|w|} \rightarrow L'^{|w|-1}$ is the secondary linearization function that maps words to labels according to a top-down absolute scale. Ω is used iff: (1) $\Omega(w_{[t:t+1]}) = (n'_t, c'_t, u'_t)$ with $n'_t \leq 3$, i.e. w_t and w_{t+1} share at most the three top levels, and (2) $\Phi(w_{[t:t+1]}) = (n_t, c_t, u_t)$ with $n_t \leq -2$, i.e. w_t is at least located two levels deeper in the tree than w_{t+1}.[3]

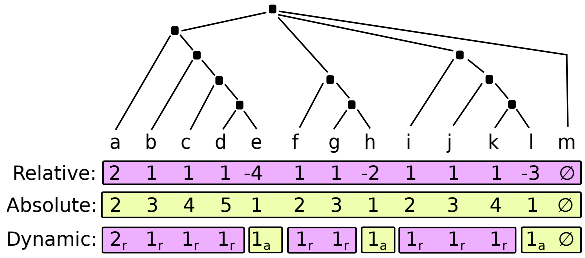


Figure 4: A synthetic constituent tree where n_t is encoded using a relative scheme, a top-down absolute scale, and an ideal dynamic combination. The relative scheme is appropriate to open and close short constituents, but becomes sparse when encoding the large ones, e.g. n_t for the tokens 'e', 'h' and 'l'. The opposite problem is observed for the top-down absolute scheme (e.g. tokens from 'a' to 'd'). The dynamic linearization combines the best of both encodings (we use the subscript 'r' to denote the labels coming from the relative encoding, and 'a' from the absolute one).

---

[3] The values were selected based on the preliminary experiments of Figure 3.

3.3 Decomposition of the label space

We showed that labels of the form $(n_t, c_t, u_t) \in L$ are sparse. An intuitive approach is to decompose the label space into three smaller sub-spaces, such that $n_i \in N$, $c_i \in C$ and $u_i \in U$. This reduces the output space from potentially $|N| \times |C| \times |U|$ labels to just $|N| + |C| + |U|$. We propose to learn this decomposed label space through a multi-task learning setup, where each of the subspaces is considered a different task, namely task$_N$, task$_C$ and task$_U$. The final loss is now computed as $\mathcal{L} = \mathcal{L}_n + \mathcal{L}_c + \mathcal{L}_u$.

We relied on a hard-sharing architecture, as it has been proved to reduce the risk of overfitting the shared parameters (Baxter, 1997). A natural issue that arises is that the prediction of labels from different label sub-spaces could be interdependent to a certain extent, and therefore a hierarchical sharing architecture could also be appropriate. To test this, in preliminary experiments we considered variants of hierarchical sharing architectures. We fed the output of the task$_U$ as input to task$_N$ and/or task$_C$. Similarly, we tested whether it was beneficial to feed the output of task$_N$ into task$_C$, and viceversa. However, all these results did not improve those of the hard-sharing model. In this context, in addition to a generalization mechanism, the shared representation could be also acting as way to keep the model aware of the potential interdependencies that might exist between subtasks.

3.4 Mitigating Effects of Greedy Decoding

We propose two ways to mitigate error propagation arising from greedy decoding in constituent parsing as sequence tagging: auxiliary tasks and policy gradient fine-tuning. Note that we want to optimize bracketing F-score *and* speed. For this reason we do *not* explore approaches that come at a speed cost in testing time, such as beam-search or using conditional random fields (Lafferty et al., 2001) on top of our LSTM.

Auxiliary tasks Auxiliary tasks force the model to take into account patterns in the input space that can be useful to solve the main task(s), but that remain ignored due to a number of factors, such as the distribution of the output label space (Rei, 2017). In a similar fashion, we use auxiliary tasks as a way to force the parser to pay attention to aspects beyond those needed for greedy decoding. We propose and evaluate two separate strategies:

1. Predict partial labels n_{t+k} that are k steps from the current time step t. This way we can jointly optimize at each time step a prediction for the pairs $(w_t, w_{t+1}), \ldots, (w_{t+k}, w_{t+k+1})$. In particular, we will experiment both with previous and upcoming n_k's, setting $|k|$=1.

2. Predict the syntactic distances presented by Shen et al. (2018), which reflect the order a sentence must be split to obtain its constituent tree using a top-down parsing algorithm (Stern et al., 2017). The algorithm was initially defined for binary trees, but its adaptation to *n*-ary trees is immediate: leaf nodes have a split priority of zero and the ancestors' priority is computed as the maximum priority of their children plus one. In this work, we use this algorithm in a sequence tagging setup: the label assigned to each token corresponds to the syntactic distance of the lowest common ancestor with the next token. This is illustrated in Figure 5.

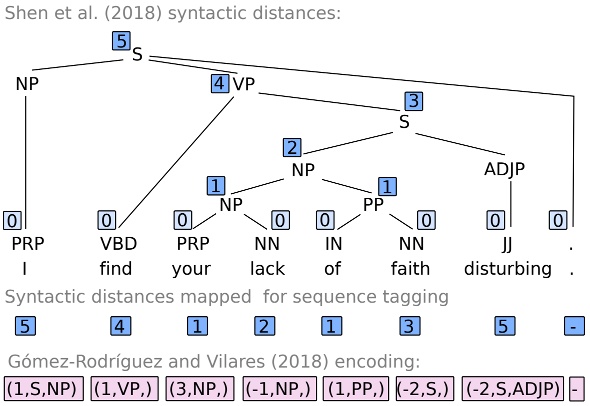


Figure 5: A constituent with syntactic distances attached to each non-terminal symbol, according to Shen et al. (2018). Distances can be used for sequence tagging, providing additional information to our base encoding (Gómez-Rodríguez and Vilares, 2018)

The proposed auxiliary tasks provide different types of contextual information. On the one hand, the encoding of the n_ts by Gómez-Rodríguez and Vilares (2018) only needs to know about w_t and w_{t+1} paths to generate the label for the time step t. On the other hand, to compute the syntactic distance of a given non-terminal symbol, we need to compute the syntactic distances of its subtree, providing a more global, but also sparser context. For training, the loss coming from the auxiliary task(s) is weighted by β=0.1, i.e, the final loss is computed as $\mathcal{L} = \mathcal{L}_n + \mathcal{L}_c + \mathcal{L}_u + \beta \sum_a \mathcal{L}_a$.

Policy gradient fine-tuning Policy gradient training methods allow us to fine-tune our models with a tree-level objective, optimizing directly for bracketing F-score. We start off with a converged supervised model as our initial policy. The sequence labeling model can be seen as a functional approximation of the policy π parametrized by θ, which at timestep t selects a label l_t=(n_t, c_t, u_t)[4] given the current state of the model's parameters, s_t. The agent's reward, R_{tree}, is then derived from the bracketing F-score. This can be seen as a variant of the REINFORCE algorithm (Williams, 1992) where the policy is updated by gradient ascent in the direction of:

$$\Delta_\theta log\pi(l_t|s_t;\theta)R_{tree} \quad (1)$$

Baseline and Variance Reduction We use as baseline a copy of a pre-trained model where the parameters are frozen. The reward used to scale the policy gradient can then be seen as an estimate of the advantage of an action l_t in state s_t over the baseline model. This is equivalent to $R_{tree}-B_{tree}$, where R_{tree} is the bracketing F-score of a sequence sampled from the current policy and B_{tree} is the the tree-level F-score of the sequence greedily predicted by the baseline. To further reduce the variance, we standardize the gradient estimate Δ_θ using its running mean and standard deviation for all candidates seen in training so far. In initial experiments without these augmentations, we observed that fine-tuning with vanilla PG often led to a deterioration in performance. To encourage exploration away from the converged supervised model's policy, we add the entropy of the policy to the objective function (Williams and Peng, 1991). Moreover, following Lillicrap et al. (2015), we optionally add noise sampled from a noise process N to the policy. The gradient of our full fine-tuning objective function takes the following form:

$$\begin{aligned}\Delta_\theta(log\pi(l_t|s_t;\theta) + N)(R_{tree} - B_{tree})\\ + \beta\Delta_\theta H(\pi(s_t;\theta) + N) \quad (2)\end{aligned}$$

where H is the entropy and β controls the strength of the entropy regularization term.

4 Experiments

We now review the impact of the proposed techniques on a wide variety of settings.

Datasets We use the English Penn Treebank (PTB) (Marcus et al., 1993) and the Chinese Penn Treebank (CTB) (Xue et al., 2005). For these, we use the same predicted PoS tags as Dyer et al. (2016). We also provide detailed results on the SPMRL treebanks (Seddah et al., 2014),[5] a set of datasets for constituent parsing on morphologically rich languages. For these, we use the predicted PoS tags provided together with the corpora. To the best of our knowledge, we provide the first evaluation on the SPMRL datasets for sequence tagging constituent parsers.

Metrics We report bracketing F-scores, using the EVALB and the EVAL-SPMRL scripts. We measure the speed in terms of sentences per second.

Setup We use NCRFpp (Yang and Zhang, 2018), for direct comparison against Gómez-Rodríguez and Vilares (2018). We adopt bracketing F-score instead of label accuracy for model selection and report this performance as our second baseline. After 100 epochs, we select the model that fared best on the development set. We use GloVe embeddings (Pennington et al., 2014) for our English models and `zzgiga` embeddings (Liu and Zhang, 2017) for the Chinese models, for a more homogeneous comparison against other parsers (Dyer et al., 2016; Liu and Zhang, 2017; Fernández-González and Gómez-Rodríguez, 2018). ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018) could be used to improve the precision, but in this paper we focus on keeping a good speed-accuracy tradeoff. For SPMRL, no pretrained embeddings are used, following Kitaev and Klein (2018a). As a side note, if we wanted to improve the performance on these languages we could rely on the CoNLL 2018 shared task pretrained word embeddings (Zeman et al., 2018) or even the multilingual BERT model[6]. Our models are run on a single CPU[7] (and optionally on a consumer-grade GPU for further comparison) using a batch size of 128 for testing. Additional hyperparameters can be found in Appendix A.

4.1 Results

Table 1 contrasts the performance of our models against the baseline on the PTB development set.

---

[4] 3 different labels in the MTL setting.

[5] Except for Arabic, for which we do not have the license.

[6] `https://github.com/google-research/bert/blob/master/multilingual.md`

[7] Intel Core i7-7700 CPU 4.2 GHz

Model	F-score	(+/-)	Sents/s
Gómez and Vilares (2018)	89.70	-	109
Our baseline	89.77	(+0.07)	111
+ DE	90.22	(+0.52)	111
+ MTL	90.38	(+0.68)	130
aux(n_{t+1})	90.41	(+0.71)	130
aux(n_{t-1})	90.57	(+0.87)	130
aux(distances)	90.55	(+0.85)	130
+ PG	**90.70**	(+1.00)	130

Table 1: Results on the PTB dev set, compared against Gómez-Rodríguez and Vilares (2018). DE refers to dynamic encoding and MTL to a model that additionally casts the problem as multi-task learning. Each auxiliary task is added separately to the baseline with DE and MTL. Policy gradient fine-tunes the model that includes the best auxiliary task.

To show that the model which employs dynamic encoding is better (+0.52) than the baseline when it comes to closing brackets from long constituents, we compare their F-scores in Figure 6. When we recast the constituent-parsing-as-sequence-tagging problem as multi-task learning, we obtain both a higher bracketing F-score (+0.68) and speed (1.17x faster). Fusing strategies to mitigate issues from greedy decoding also leads to better models (up to +0.87 when adding an auxiliary task[8] and up to +1.00 if we also fine-tune with PG). Note that including auxiliary tasks and PG come at a time cost in training, but not in testing, which makes them suitable for fast parsing.

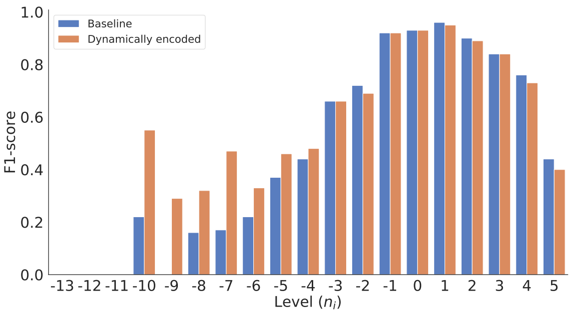


Figure 6: F-score for n_ts on the PTB dev set, obtained by the Gómez-Rodríguez and Vilares (2018) baseline (in blue, first bar for each n_t, already shown in Figure 3) and our model with dynamically encoded trees (in orange, second bar).

Table 2 replicates the experiments on the CTB and the SPMRL dev sets. The dynamic encoding improves the performance of the baseline on large treebanks, e.g. German, French or Korean, but causes some drops in the smaller ones, e.g. Swedish or Hebrew. Overall, casting the problem as multitask learning and the strategies used to mitigate error propagation lead to improvements.

For the experiments on the test sets we select the models that summarize our contributions: the models with dynamic encoding and the multi-task setup, the models including the best auxiliary task, and the models fine-tuned with policy gradient.

Tables 3, 4 and 5 compare our parsers against the state of the art on the PTB, CTB and SPMRL test sets. Gómez-Rodríguez and Vilares (2018) also run experiments without character embeddings, to improve speed without suffering from a big drop in performance. For further comparison, we also include them as additional results (shadowed). In a related line, Smith et al. (2018) show that for dependency parsing two out of three embeddings (word, postag and characters) can suffice.

4.2 Discussion

The results across the board show that the dynamic encoding has a positive effect on 6 out of 10 treebanks. Casting the constituent-parsing-as-sequence-labeling problem as MTL surpasses the baseline for all tested treebanks (and it leads to better parsing speeds too). Finally, by mitigating issues from greedy decoding we further improve the performance of all models that include dynamic encodings and multi-task learning.

On the PTB, our models are both faster and more accurate than existing sequence tagging or sequence-to-sequence models, which already were among the fastest parsers (Gómez-Rodríguez and Vilares, 2018; Vinyals et al., 2015). We also outperform other approaches that were not surpassed by the original sequence tagging models in terms of F-score (Zhu et al., 2013; Fernández-González and Martins, 2015). On the CTB our techniques also have a positive effect. The baseline parses 70 sents/s on the CTB, while the full model processes up to 120. The speed up is expected to be larger than the one obtained for the PTB because the size of the label set for the baseline is bigger, and it is reduced in a greater proportion when the constituent-parsing-as-sequence-labeling problem is cast as MTL.

On the SPMRL corpora, we provide the first evaluation of sequence labeling constituent parsers, to verify if these perform well on mor-

---

[8] We observed that adding more than one auxiliary task did not translate into a clear improvement. We therefore chose the auxiliary task that performed the best in the development set.

Model	CTB	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Our baseline	88.57	87.93	81.09	87.83	89.27	88.85	83.51	92.60	80.11
+DE	88.37	87.91	81.16	**88.81**	89.03	88.70	83.92	93.35	79.57
+MTL	88.57	89.41	81.70	88.52	**92.72**	89.73	84.10	93.81	82.83
aux(n_{t+1})	88.73	89.65	81.95	88.64	92.65	89.69	84.09	93.86	82.82
aux(n_{t-1})	88.48	89.47	81.77	88.58	92.53	89.71	84.13	93.87	82.74
aux(distances)	88.51	89.48	82.02	88.68	92.66	89.80	84.20	93.83	83.12
+PG	**89.01**	**89.73**	**82.13**	88.80	92.66	**89.86**	**84.45**	**93.93**	**83.15**

Table 2: Results on the CTB and SPMRL dev sets

Model	Sents/s	Hardware	F-score
Vinyals et al. (2015)	120	Many CPU	88.30
Coavoux and Crabbé (2016)	168	1 CPU	88.60
Fernández and Martins (2018)	41	1 CPU	90.20
Zhu et al. (2013)	90	1 CPU	90.40
Dyer et al. (2016)	17	1 CPU	91.20
Stern et al. (2017)	76	16 CPU	91.77
Shen et al. (2018)	111	1 GPU	91.80
Kitaev and Klein (2018a) (single model)	213	2 GPU	93.55
Kitaev and Klein (2018a) (with ELMo)	71	2 GPU	95.13
Kitaev and Klein (2018b) (ensemble and BERT)	-	-	*95.77*
Gómez and Vilares (2018)	115	1 CPU	90.00
Our baseline	115	1 CPU	90.06
+DE	115	1 CPU	90.19
+MTL	132	1 CPU	90.36
+ best aux	132	1 CPU	90.59
+PG	132	1 CPU	90.60
+PG	942	1 GPU	90.60
+PG (no char emb)	149	1 CPU	90.50
+PG (no char emb)	1267	1 GPU	90.50

Table 3: Comparison on the PTB test set. Kitaev and Klein (2018b) are results published after this work was submitted (italics represent the cases where they obtain a new state of the art on the corresponding language).

phologically rich languages. We then evaluated whether the proposed techniques can generalize on heterogeneous settings. The tendency observed for the original tagging models by Gómez-Rodríguez and Vilares (2018) is similar to the one for the PTB and CTB: they improve other fast parsers, e.g. Coavoux and Crabbé (2016), in 5 out of 8 treebanks and Fernández-González and Martins (2015) in 7 out of 8, but their performance is below more powerful models. When incorporating the techniques presented in this work, we outperform the original sequence tagging models on all datasets. We outperform the current best model for Basque, Hebrew and Polish (Kitaev and Klein, 2018a) and for Swedish (Björkelund et al., 2014), which corresponds to the four smallest treebanks among the SPMRL datasets. This indicates that even if sequence tagging models are conceptually simple and fast, they can be very suitable when little training data is available. This is also of special interest in terms of research for low-resource languages. Again, casting the problem as MTL reduces the parsing time for all tested treebanks, as reflected in Table 6. Finally, for treebanks such as French, designing methods to handle multi-word expressions could lead to better results, getting closer to other parsers (Coavoux and Crabbé, 2017).

Model	F-score
Zhu et al. (2013)	83.2
Dyer et al. (2016)	84.6
Liu and Zhang (2017)	86.1
Shen et al. (2018)	86.5
Fernández and Gómez-Rodríguez (2018)	86.8
Gómez and Vilares (2018)	84.1
Our baseline	83.90
+DE	83.98
+MTL	84.24
+best aux	85.01
+PG	85.61
+PG (no char emb)	83.93

Table 4: Comparison on the CTB test set

5 Conclusion

We have explored faster and more precise sequence tagging models for constituent parsing. We proposed a multitask-learning architecture that employs dynamic encodings, auxiliary tasks, and policy gradient fine-tuning. We performed experiments on the English and Chinese Penn Treebanks, and also on the SPMRL datasets. Our models improve current sequence tagging parsers on all treebanks, both in terms of performance and speed. We also report state-of-the-art results for the Basque, Hebrew, Polish, and Swedish datasets. The methods presented in this work are specifically designed for constituent parsing. However, it seems natural to apply some of these to other NLP tagging tasks, e.g. using multi-task learning to predict sub-level morphological information for morphologically-rich part-of-speech tagging.

Model	**Basque**	**French**	**German**	**Hebrew**	**Hungarian**	**Korean**	**Polish**	**Swedish**	**Avg**
Fernández-González and Martins (2015)	85.90	78.75	78.66	88.97	88.16	79.28	91.20	82.80	84.21
Coavoux and Crabbé (2016)	86.24	79.91	80.15	88.69	90.51	85.10	92.96	81.74	85.67
Björkelund et al. (2014) (ensemble)	88.24	82.53	81.66	89.80	91.72	83.81	90.50	85.50	86.72
Coavoux and Crabbé (2017)	88.81	82.49	85.34	89.87	92.34	86.04	93.64	84.00	87.82
Kitaev and Klein (2018a)	89.71	**84.06**	**87.69**	90.35	**92.69**	**86.59**	93.69	84.35	**88.64**
Kitaev and Klein (2018b) (with BERT)	*91.63*	*87.42*	*90.20*	*92.99*	*94.90*	*88.80*	*96.36*	*88.86*	*91.40*
Baseline	89.54	80.56	84.05	88.83	90.42	83.33	92.48	83.67	86.61
+DE	89.56	80.69	84.64	88.80	90.02	83.67	93.20	83.40	86.75
+MTL	90.90	80.98	84.94	91.99	90.63	83.91	93.80	86.26	87.92
+best aux	**91.23**	81.27	84.95	**92.03**	90.60	83.67	93.84	86.62	88.02
+PG	91.18	81.37	84.88	**92.03**	90.65	84.01	**93.93**	**86.71**	88.10
+PG (no char emb)	90.14	81.38	85.09	92.11	90.34	83.46	93.87	86.65	87.88

Table 5: Comparison on the test SPMRL datasets (except Arabic). Kitaev and Klein (2018b) are results published after this work was submitted (italics represent the cases where they obtain a new state of the art on the corresponding language).

Dataset	**Baseline speed**	**Full speed (increase)**	**Full (no char) speed (increase)**
Basque	179	223 (1.25x)	257 (1.44x)
French	76	91 (1.20x)	104 (1.37x)
German	70	100 (1.43x)	108 (1.54x)
Hebrew	44	102 (2.32x)	115 (2.61x)
Hungarian	93	134 (1.44x)	150 (1.61x)
Korean	197	213 (1.08x)	230 (1.17x)
Polish	187	253 (1.35x)	278 (1.49x)
Swedish	98	158 (1.61x)	187 (1.81x)

Table 6: Comparison of speeds on the SPMRL datasets

Acknowlegments

DV has received support from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). MA and AS are funded by a Google Focused Research Award.

References

Jonathan Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169. Association for Computational Linguistics.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morpho-syntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 97–102.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 331–336.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the* *eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Faster Shift-Reduce Constituent Parsing with a Non-Binary, Bottom-Up Strategy. *ArXiv e-prints*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533. Association for Computational Linguistics.

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 469–476. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324. Association for Computational Linguistics.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nikita Kitaev and Dan Klein. 2018a. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018b. Multilingual constituency parsing with self-attention and pretraining. *CoRR*, abs/1812.11760.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Minh Le and Antske Fokkens. 2017. Tackling error propagation through reinforcement learning: A case of greedy dependency parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 677–687. Association for Computational Linguistics.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. *To appear at AAAI 2019*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 691–698, Stroudsburg, PA, USA. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109. Dublin City University.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180. Association for Computational Linguistics.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018. An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Oriol Vinyals, Łũkasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Daniel Zeman, Jan Haji, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443. Association for Computational Linguistics.

A Appendices

For the BILSTM-based model, we essentially follow the configuration of the baseline (Gómez-Rodríguez and Vilares, 2018) for an homogenous comparison. We detail the hyperparameters in Table 7.[9]

Hyperparameter	**Value**
BILSTM size	800
# BILSTM layers	20
optimizer	SGD
loss	cat. cross-entropy
learning rate	0.2
decay (linear)	0.05
momentum	0.9
dropout	0.5
word emb size	100
features size	20
character emb size	50
batch size training	8
training epochs	100
batch size test	128
PG finetuning Hyperparameter	**Value**
# samples	8
learning rate	0.0005
entropy regularization coefficient	0.01
variance reduction burn-in # of examples	1000
layers frozen	word & char embeddings
noise process initial stddev	0.1
noise process desired action stddev	0.5
noise process adaptation coefficient	1.05

Table 7: Additional hyperparameters of the base model and Policy Gradient fine-tuning

---

[9] Note that the noise sampling is only used for Swedish in the final models based on development set results with and without it.

CAN-NER: Convolutional Attention Network for Chinese Named Entity Recognition

Yuying Zhu*
Nankai University
Tianjin, China
yuyzhu@mail.nankai.edu.cn

Guoxin Wang
Microsoft Research Asia
Beijing, China
guow@microsoft.com

Abstract

Named entity recognition (NER) is a common task in Natural Language Processing (NLP), but it remains more challenging in Chinese because of its lack of natural delimiters. Therefore, Chinese Word Segmentation (CWS) is usually necessary as the first step for Chinese NER. However, models based on word-level embeddings and lexicon features often suffer from segmentation errors and out-of-vocabulary (OOV) problems. In this paper, we investigate a **C**onvolutional **A**ttention **N**etwork (**CAN**) for Chinese NER, which consists of a character-based convolutional neural network (CNN) with local-attention layer and a gated recurrent unit (GRU) with global self-attention layer to capture the information from adjacent characters and sentence contexts. Moreover, differently from other approaches, CAN-NER does not depend on any external resources like lexicons and employing small-size char embeddings makes CAN-NER more practical for real systems scenarios. Extensive experimental results show that our approach outperforms state-of-the-art methods without word embedding and external lexicon resources on different domains datasets.

1 Introduction

Named Entity Recognition (NER) aims at identifying text spans which are associated with a specific semantic entity type such as person (PER), organization (ORG), location (LOC), and geopolitical entity (GPE). NER has received constant research attention as it is the first step in a wide range of downstream Natural Language Processing (NLP) tasks, e.g., entity linking (Gupta et al., 2017), relation extraction (Miwa and Bansal, 2016), event extraction (Chen et al., 2015), and coreference resolution (Fragkou, 2017). The standard approach in existing state-of-the-art models for English NER treats the problem as a word-by-word sequence labeling task and makes full use of the Recurrent Neural Network (RNN) and Conditional Random Field (CRF) to capture context information at the word level (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2018). These models for English NER predict a tag for each word assuming that words can be separated clearly by explicit word separators, e.g., blank spaces. As the Chinese language has no natural delimiters, it would be intuitive to apply Chinese Word Segmentation (CWS) first to get word boundaries and then use a word-level sequence labeling model similar to the English NER models. However, word boundaries can be ambiguous in Chinese, which leads to the possibility that entity boundaries do not match word boundaries. For example, the term “西藏自治区 (Tibet Autonomous Region)” is a GPE-type entity in NER, but it could be segmented as a single word or as two words “西藏 (Tibet)” and “自治区 (autonomous region)” separately, depending on different granularity of segmentation tools. Most of the time, however, it is hard to determine the correct granularity for word segmentation. Also, as shown in Figure 1, different segmentation can lead to different sentence meanings in Chinese, which could even result in different named entities. Obviously, if entity boundaries are

Sentence:
南京市长江大桥

Segmentation 1:
南京市 | 长江大桥
Nanjing City, Yangtze River Bridge
Location, Location

Segmentation 2:
南京 | 市长 | 江大桥
Nanjing, Mayor, Jiang Daqiao
Location, Title, Person

Figure 1: Entity Ambiguity with Word Segmentation.

---

[*] This work was performed when the first author was an intern at Microsoft Research Asia.

Proceedings of NAACL-HLT 2019, pages 3384–3393
Minneapolis, Minnesota, June 2 - June 7, 2019.

mistakenly detected in segmentation, it will negatively affect entity tagging in word-based NER models. Furthermore, most recent neural network-based Chinese NER models rely heavily on word-level embeddings and external lexicon sets (Huang et al., 2017; Zhang and Yang, 2018). The quality of such models strongly relies on the different word embedding representations and lexicon features. Moreover, word-based models tend to suffer from OOV issues as Chinese words can be very diverse and named entities are an important source of OOV words. Other potential limitations are as follows: (1) Dependency on word embeddings increases model size and makes the fine-tuning process more costly during training (while negatively affecting latency in testing/decoding); (2) It is hard to learn word representation correctly without enough labeled utterances for named entities are usually rarer proper nouns. (3) Large lexicons are very costly for real NER systems as they greatly increase memory usage and latency in feature extraction (matching), which makes models inefficient; (4) It is very costly to remove noise from large lexicons and any update to pre-trained word embeddings or lexicons requires model retraining. Meanwhile, character-level embedding by itself can only carry limited information due to losing word and word-sequence information. For instance, the character “拍” in words “球拍” (bat) and “拍卖” (auction) has very different meanings. How to better integrate segmentation-related information and exploit local context information is the key feature in a character-based model. Zhang and Yang (2018) leverage lexicons to add all the embeddings of candidate word segmentation to their last character embeddings as soft features, and construct a convolutional neural network (CNN) to encode characters as word-level information. Cao et al. (2018) propose a multi-task architecture to learn NER tagging and Chinese word segmentation together, with each part using a character-based Bi-LSTM. In this paper, we propose a convolutional attention layer to capture the implicit relations within adjacent characters, in which the position features from word segmentation are soft hints for character combinations. With the segmentation vector softly concatenating into character embedding, the convolutional attention layer is able to group implicitly meaning-related characters and help bypass the impact of segmentation errors. A BiGRU structure with a global self-attention layer on the whole sentence is utilized to capture sentence-level dependencies. Extensive experimental results show that our approach outperforms state-of-the-art methods without relying on external resources (e.g. word embedding, external lexicon) across different corpora. The main contributions of this paper can be summarized as follows:

- We first combine CNNs with the local-attention mechanism to enhance the ability of the model to capture implicitly local context relations among character sequences. Compared with experimental results against a baseline with a regular CNN layer, our Convolutional Attention layer leads to substantial performance improvements.

- We introduce a character-based Chinese NER model that consists of combined CNN with local attention and BiGRU with global self-attention layers. Our model achieves state-of-the-art F1-scores without using any external resources like word embeddings and lexicon resources, which make it very practical for real-world NER systems.

2 Methodology

We utilize BiGRU-CRF as our basic model structure. Our model considers multi-level context features in three layers: i) convolutional attention layer, ii) GRU layer, and iii) global attention layer. The whole architecture of our proposed model is illustrated in Figure 2.

2.1 Formulation

In the Chinese NER task, we denote an input sentence as $\mathbf{X}_i = \{x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,\tau}\}$, where $x_{i,\tau} \in \mathbb{R}^{d_e}$ represents the τ-th character in sentence $\mathbf{X}_i$ and d_e is the dimension of the input embeddings. Correspondingly, we denote the sentence label sequence as $\mathbf{Y}_i = \{y_{i,1}, y_{i,2}, y_{i,3}, ..., y_{i,\tau}\}$, where $y_{i,\tau} \in \mathcal{Y}$ belongs to the set of all possible labels. The objective is learning a function $f_\theta : \mathbf{X} \mapsto \mathbf{Y}$ to obtain the entity types including the ‘O’ type for all the characters in the input text. In the following text, we take one instance as the example and therefore omit subindex i in the formula.

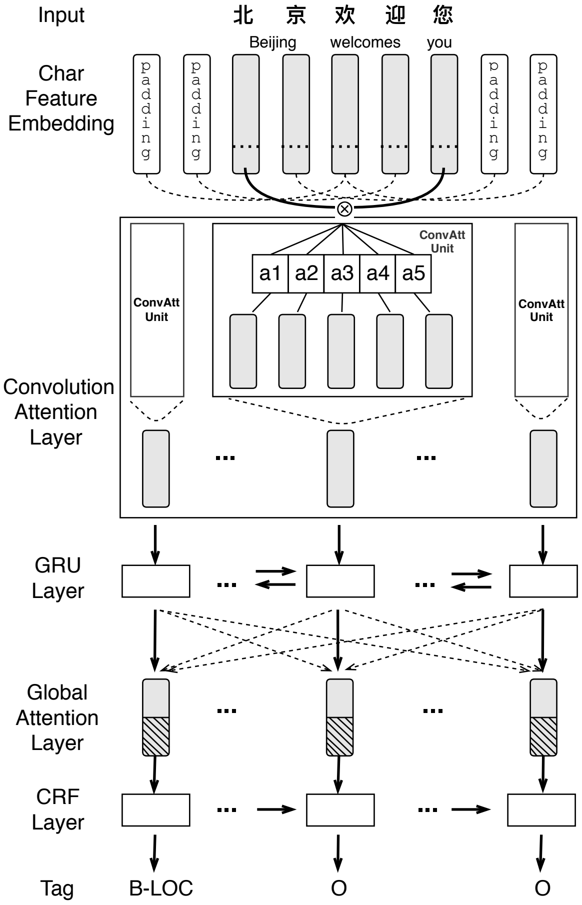


Figure 2: Overall model architecture. A convolutional attention layer is constructed to encode both character- and word-level information. The BiGRU-CRF layer is extended by a global self-attention layer to capture long sequential sentence-level relations.

2.2 Convolutional Attention Layer

The convolutional attention layer aims to encode the sequence of input characters and implicitly group meaning-related characters in the local context. The input representation for each character is constructed as $x = [x_{ch}; x_{seg}]$, where $x_{ch} \in \mathbb{R}^{d_{ch}}$ and $x_{seg} \in \mathbb{R}^{d_{seg}}$ are character embedding and segmentation mask, respectively. The segmentation information is encoded by BMES scheme (Wang and Xu, 2017).

For every window in the CNN, whose window size is k, we first concatenate a position embedding to each character embedding, helping to keep sequential relations in the local window context. The dimension of the position embedding equals to the window size k with the initial values of 1 at the position where the character lies in the window and 0 at other positions. So, the dimension of the concatenated embedding is $d_e = d_{ch} + d_{pos} + d_{seg}$. We then apply local attention inside the window to capture the relations between the center character and each context token, followed by a CNN with sum-pooling layer. We set the hidden dimension as d_h. For the j-th character, the local attention takes all the concatenated embeddings $x_{j-\frac{k-1}{2}}, ...x_j, ..., x_{j+\frac{k-1}{2}}$ in the window as its input and outputs k hidden vectors $h_{j-\frac{k-1}{2}}, ..., h_j, ..., h_{j+\frac{k-1}{2}}$. The hidden vectors are calculated as follows:

$$h_m = \alpha_m x_m, \tag{1}$$

where $m \in \{j - \frac{k-1}{2}, ..., j + \frac{k-1}{2}\}$ and α_m is the attention weight, which is calculated as:

$$\alpha_m = \frac{\exp s(x_j, x_m)}{\sum_{n \in \{j-\frac{k-1}{2}, ..., j+\frac{k-1}{2}\}} \exp s(x_j, x_n)}. \tag{2}$$

The score function s is defined as follows:

$$s(x_j, x_k) = v^\top \tanh(W_1 x_j + W_2 x_k), \tag{3}$$

where $v \in \mathbb{R}^{d_h}$ and $W_1, W_2 \in \mathbb{R}^{d_h, d_e}$.

The CNN layer contains d_h kernels on a context window of k tokens as:

$$h_j^c = \sum_k [W^c * h_{j-\frac{k-1}{2}:j+\frac{k-1}{2}} + b^c], \tag{4}$$

where $W^c \in \mathbb{R}^{k \times d_h \times d_e}$ and $b^c \in \mathbb{R}^{k \times d_h}$. The $*$ operation denotes element-wise product and $h_{j-\frac{k-1}{2}:j+\frac{k-1}{2}}$ means a concatenation of the hidden states $h_{j-\frac{k-1}{2}}, ..., h_{j+\frac{k-1}{2}}$, both of which are calculated at the first dimension. Finally sum-pooling is also conducted on the first dimension.

2.3 BiGRU-CRF with Global Attention

After extracting the local context features by the convolutional attention layer, we feed them into a BiGRU-CRF based model to predict final label for each character. This layer models the sequential sentence information and it is calculated as follows:

$$h_j^r = \text{BiGRU}(h_{j-1}^r, h_j^c; W^r, U^r), \tag{5}$$

where h_j^c is the output of the convolutional attention layer, h_{j-1}^r is the previous hidden state for the BiGRU layer, and $W^r, U^r \in \mathbb{R}^{d_h \times d_h}$ are its parameters.

A global self-attention layer is utilized to better handle sentence-level information, as:

$$h_j^g = \sum_{s=1}^{n} \alpha_{j,s}^g h_s^r \tag{6}$$

where $j = 1, ..., \tau$ denotes all characters in a sentence instance and $\alpha^g_{j,s}$ is calculated as:

$$\alpha^g_{j,s} = \frac{\exp s(h^r_j, h^r_s)}{\sum_{n \in \{1,...,\tau\}} \exp s(h^r_j, h^r_n)}. \tag{7}$$

The score function s is similar to Equation 3 with different parameters $v^g \in \mathbb{R}^{d_h}$ and $W^g_1, W^g_2 \in \mathbb{R}^{d_h, d_h}$ instead.

Finally, a standard CRF layer is used at the top of the concatenation of the output of the BiGRU and global attention layers, which is denoted as $H_\tau = [h^r_\tau; h^g_\tau]$. Given the predicted tag sequence $\mathbf{Y} = \{y_1, y_2, y_3, ..., y_\tau\}$, the probability of the ground-truth label sequence is computed by:

$$P(\mathbf{Y}|\mathbf{X}) = \frac{exp(\sum_i(\mathbf{W}^{y_i}_{\text{CRF}} H_i + b^{(y_{i-1},y_i)}_{\text{CRF}}))}{\sum_{y'} exp(\sum_i(\mathbf{W}^{y'_i}_{\text{CRF}} H_i + b^{(y'_{i-1},y'_i)}_{\text{CRF}}))}, \tag{8}$$

where y' denotes an arbitrary label sequence, $\mathbf{W}^{y_i}_{\text{CRF}}$ and $b^{(y_{i-1},y_i)}_{\text{CRF}}$ are trainable parameters. In decoding, we use the Viterbi algorithm to get the predicted tag sequence.

2.4 Training

For training, we exploit log-likelihood objective as the loss function. Given a set of training examples $\{(\mathbf{X}_i, \mathbf{Y}_i)\}|^K_{i=1}$, the loss function L can be defined as follows:

$$L = \sum_{i=1}^{K} logP(\mathbf{Y}_i|\mathbf{X}_i) \tag{9}$$

In the training phase, at each iteration, we first shuffle all the training instances, and then feed them to the model with batch updates. We use the AdaDelta (Zeiler, 2012) algorithm to optimize the final objective with all the parameters as described in Section 3.1.

Dataset	Type	Train	Test	Dev
OntoNotes	Sentences	15.7k	4.3k	4.3k
	Chars	491.9k	208.1k	200.5k
	Entities	13.4k	7.7k	6.95k
MSRA	Sentences	46.4k	4.4k	-
	Chars	2169.9k	172.6k	-
	Entities	74.8k	6.2k	-
Weibo	Sentences	1.4k	0.27k	0.27k
	Chars	73.8k	14.8k	14.5k
	Entities	1.89k	0.42k	0.39k
Resume	Sentences	3.8k	0.48k	0.46k
	Chars	124.1k	15.1k	13.9k
	Entities	1.34k	0.15k	0.16k

Table 1: Statistics of each dataset

3 Experiments

To demonstrate the effectiveness of our proposed model, we have run multiple experiments on Chinese NER datasets covering different domains. This section describes the details of each dataset, settings, and results in our experiments. Standard precision (P), recall (R) and F1-score (F1) are used as evaluation metrics.

3.1 Experimental Settings

Data We use four datasets in our experiments. For the news domain, we experiment on OntoNotes 4 (Weischedel et al., 2011) and MSRA NER dataset from SIGHAN Bakeoff 2006 (Levow, 2006). For the social media domain, we adopt the same annotated Weibo corpus as Peng and Dredze (2015) which is extracted from Sina Weibo[1]. For more variety in test domains, we also use a Chinese Resume dataset (Zhang and Yang, 2018) collected from Sina Finance[2].

The Weibo dataset is annotated with four entity types: PER (Person), ORG (Organization), LOC (Location), and GPE (Geo-Political Entity); and it includes both named and nominal mentions. This corpus is already divided into training, development, and test sets. The Chinese Resume dataset is annotated with eight types of named entities: CONT (Country), EDU (Educational Institution), LOC, PER, ORG, PRO (Profession), RACE (Ethnicity/Background), and TITLE (Job Title). OntoNotes 4 is annotated with four named entity categories: PER, ORG, LOC, and GPE. We follow the same data split method of Che et al. (2013) over OntoNotes 4. Lastly, the MSRA 2006 dataset contains three annotated named entities: ORG, PER and LOC. A development subset is

Models	NE	NM	Overall
Peng and Dredze (2015)	51.96	61.05	56.05
Peng and Dredze (2016)*	55.28	62.97	58.99
He and Sun (2017a)	50.60	59.32	54.82
He and Sun (2017b)*	54.50	62.17	58.23
Cao et al. (2018)	54.34	57.35	58.70
Zhang and Yang (2018)	53.04	62.25	58.79
Baseline	49.02	58.80	53.80
Baseline + CNN	53.86	58.05	55.91
CAN-NER Model	**55.38**	**62.98**	**59.31**

Table 2: Weibo NER results

---

[1] http://www.weibo.com/

[2] http://finance.sina.com.cn/stock/index.html

not available for the MSRA dataset. The detailed statistics of each datasets are shown in Table 1.

Gold segmentation is unavailable for Weibo, Chinese Resume, and MSRA test sections. We follow Zhang and Yang (2018) to automatically segment these by using the model described in Yang et al. (2017). We treat NER as a sequential labeling problem and adopt the BIOES tagging style since it has been shown to produce better results than straight BIO (Yang et al., 2018b).

Hyper-parameter settings For hyper-parameter configuration, we adjust them according to the performance on the described development sets for Chinese NER. We set the character embedding size, hidden sizes of CNN and BiGRU to 300 dims. After comparing experimental results with different CNN window sizes, we set the window size as 5. Adadelta is used for optimization, with an initial learning rate of 0.005. The character embeddings used in our experiments are from Li et al. (2018), which is trained by Skip-Gram with Negative Sampling (SGNS) on Baidu Encyclopedia.

3.2 Experimental Results

In this section, we describe the experimental results of our proposed model and previous state-of-the-art methods on four datasets: Weibo, Chinese Resume, OntoNotes 4, and MSRA. We propose two baselines for comparison, and show the CAN-NER model results. In the experiment results table, we use Baseline to represent a pure BiGRU + CRF model; and Baseline + CNN to indicate the base model with a CNN layer.

3.2.1 Weibo Dataset

Here we compare our proposed model with the latest models on the Weibo dataset.[3] Table 2 shows the F1-scores for named entities (NE), nominal entities (NM, excluding named entities), and both (Overall). We observe that our proposed model achieves state-of-the-art performance.

Existing state-of-the-art systems include Peng and Dredze (2016), He and Sun (2017b), Cao et al. (2018) and Zhang and Yang (2018), which leverage rich external data like cross-domain data, semi-supervised data, and lexicons, or joint-train NER and Chinese Word Segmentation (CWS).[4] In the first block of Table 2, we report the performance of the latest models. Peng and Dredze (2015) propose a model that jointly trains embeddings with NER and it achieves a F1-score of 56.05% on overall performance. The model (Peng and Dredze, 2016) that jointly trains NER and CWS reaches a F1-score of 58.99%. He and Sun (2017b) propose a unified model to exploit cross-domain and semi-supervised data, which improves the F1-score from 54.82% to 58.23% compared with the model proposed by He and Sun (2017a). Cao et al. (2018) use an adversarial transfer learning framework to incorporate task-shared word boundary information from CWS and achieves a F1-score of 58.70%. Zhang and Yang (2018) leverage a lattice structure to integrate lexicon information into their model and achieve a F1-score of 58.79%.

In the second block of Table 2, we give the results of our baselines and proposed models. While the BiGRU + CRF baseline only achieves a F1-score of 53.80%, adding a normal CNN layer as featurizer improves the score to 55.91%. Replacing the CNN with our convolutional attention layer greatly improves the F1-score to 59.31%, which outperforms other models. The improvement demonstrates the effectiveness of our proposed model.

Models	P	R	F1
Zhang and Yang (2018)^{1†}	94.53	94.29	94.41
Zhang and Yang (2018)^{2‡}	94.07	94.42	94.24
Zhang and Yang (2018)^{3}	94.81	94.11	94.46
Baseline	93.71	93.74	93.73
Baseline + CNN	94.36	**94.85**	94.60
CAN-NER Model	**95.05**	94.82	**94.94**

Table 3: Results on Chinese Resume Dataset. For models proposed by Zhang and Yang (2018), 1 represents the char-based LSTM model, 2 indicates the word-based LSTM model and 3 is the Lattice model.

3.2.2 Chinese Resume Dataset

The Chinese Resume test results are shown in Table 3. Zhang and Yang (2018) released the Chinese Resume dataset and they achieve a F1-score of 94.46%. It can be seen that our proposed baseline (CNN + BiGRU + CRF) outperforms Zhang and Yang (2018) with F1-score of 94.60%.

---

[3] In Table 2,3, 4 and 5, we use $*$ to denote a model with external labeled data for semi-supervised learning. † denotes that the model use external lexicon data. Zhang and Yang (2018) with ‡ is the char-based model in the paper.

[4] The results of Peng and Dredze (2015, 2016) are taken from Peng and Dredze (2017)

Models	P	R	F1
Yang et al. (2016)	65.59	71.84	68.57
Yang et al. (2016)*	72.98	**80.15**	**76.40**
Che et al. (2013)*	**77.71**	72.51	75.02
Wang et al. (2013)*	76.43	72.32	74.32
Zhang and Yang (2018)†	76.35	71.56	73.88
Zhang and Yang (2018)‡	74.36	69.43	71.81
Baseline	70.67	71.64	71.15
Baseline + CNN	72.69	71.51	72.10
CAN-NER Model	75.05	72.29	73.64

Table 4: Results on OntoNotes

Adding our convolutional attention leads a further improvement and achieves state-of-the-art F1-score of 94.94%, which further demonstrates the effectiveness of our proposed model.

3.2.3 OntoNotes Dataset

Table 4 shows comparisons on the OntoNotes 4 dataset. The first block in the table lists the performance of previous methods for Chinese NER. Yang et al. (2016) propose a model combining neural and discrete feature, e.g., POS tagging features, CWS features and orthographic features, improving the F1-score from 68.57% to 76.40%. Leveraging bilingual data, Che et al. (2013) and Wang et al. (2013) achieves F1-scores of 74.32% and 73.88% respectively. Zhang and Yang (2018)‡ is a recent model that uses a character-based model with bichar and softword.

The second block of Table 4 shows the results of our baselines and proposed model. Consistently with observations on the Weibo and Resume datasets, our Convolutional Attention layer leads to a substantial increment on F1-score. Our proposed model achieves a competitive F1-score of 73.64% among character-based model without using external data (e.g., Zhang and Yang (2018)‡).

3.2.4 MSRA Dataset

Table 5 shows experiment results on the MSRA 2006 dataset. Chen et al. (2006), Zhang et al. (2006), and Zhou et al. (2013) leverage rich handcrafted features and Lu et al. (2016) exploit multi-prototype embedding features. Dong et al. (2016) introduce radical features into LSTM-CRF. Cao et al. (2018) make use of Adversarial Transfer Learning and global self-attention to improve model performance. Yang et al. (2018a) propose a character-based CNN-BiLSTM-CRF model to incorporate stroke embeddings and generate n-gram features. Zhang and Yang (2018) introduce a lattice structure to incorporate lexicon information into the neural network, which actually includes word embedding information. Although this model achieves state-of-the-art F1-score at 93.18%, it leverages external lexicon data and thus the result is dependent on the quality of the lexicon. At the bottom section of the table, we can see that Baseline + CNN already outperforms most previous methods. Compared with Zhang and Yang (2018), our char-based method achieves a competitive F1-score of 92.97% without any additional lexicon data and word embedding information. Moreover, CAN-NER model achieves state-of-the-art result among the character-based models.

Models	P	R	F1
Chen et al. (2006)	91.22	81.71	86.20
Zhang et al. (2006)*	92.20	90.18	91.18
Zhou et al. (2013)	91.86	88.75	90.28
Lu et al. (2016)	-	-	87.94
Dong et al. (2016)	91.28	90.62	90.95
Cao et al. (2018)	91.30	89.58	90.64
Yang et al. (2018a)	92.04	91.31	91.67
Zhang and Yang (2018)†	**93.57**	**92.79**	**93.18**
Baseline	92.54	88.20	90.32
Baseline + CNN	92.57	92.11	92.34
CAN-NER Model	**93.53**	92.42	**92.97**

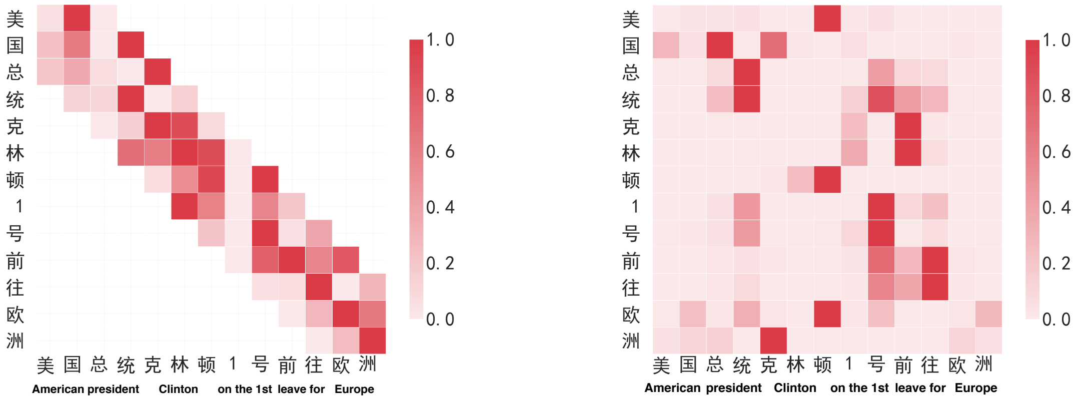
Table 5: Results on MSRA dataset

3.3 Discussion

This section discusses the model effectiveness and the experimental results.

3.3.1 Effectiveness of Convolutional Attention and Global Self-Attention

As shown in Tables 2, 3, and 5, our proposed model's performance demonstrates the effectiveness of the Convolutional Attention Network. To better evaluate the effect of the Attention Mechanism, we visualize the normalized attention weights α_m^l for each window from Eq. 2, as in Figure 3a. Each row of the matrix represents location attention weights in each window. For example, the third row indicates that the relationship between center character "总" and contexts "美 国 总 统 克". We can see from the Figure 3a that the word-level features can be extracted through the local attention. In the context, the center character "美" tends to have a stronger connection with



(a) Local attention.

(b) Global self-attention.

Figure 3: Attention visualization. The left-side image shows the normalized Convolutional Attention weights in each window in a sentence. The right-side indicates the global self-attention weights for the whole sentence. In both pictures, the x-axis represents context, while the y-axis represents the input query in the attention mechanism.

its related character “国”, which means they have a higher probability of forming the Chinese word “美国 (American)”. Also for characters “克”, “林”, and “顿”, they tend to have a strong connection because “克林顿” means “Clinton”. Characters “欧” and “洲” also have strong connections, as seen in Figure 3a, because “欧洲” represents “Europe” in Chinese. Therefore, both experiment results and visualization verifies that the Convolutional Attention is effective in obtaining phrase-level information between adjacent characters.

In Figure 3b, we visualize the global self-attention matrix. From the picture, we can find that global self-attention can capture the sentence context information from the long-distance relationship of words to overcome the limitation of Recurrent Neural Networks. For the word “克林顿 (Clinton)”, the global self-attention learns the dependencies with “前往 (leave for)” and “1号 (on the 1st)”. Distinguished by the red color, “克林顿 (Clinton)” has a stronger connection with “前往 (leave for)” than with “1号 (on the 1st)”, which matches the expectation that the predicate in a sentence provides more information to the subject than adverbs of time.

3.3.2 Results Analysis

Our proposed model outperforms previous work on th eWeibo and Chinese Resume datasets and reaches competitive results on both MSRA and OntoNotes 4 datasets without using any external resources. The experiments results demonstrate the effectiveness of our proposed model, especially among char-based models. The performance improvement after adding Convolutional Attention Layer and Global Attention Layer verifies that our model can capture the relationship between character and its local context, as well as the relationship between word and global context. However, although we can obtain comparable or better results to other models that utilize no external resources, we find that our model performance on the OntoNotes 4 dataset still has room for improvement (2.76% F1-score gap to the best model that leverages additional data). This may be explained by specific discrete features and external resources (e.g., other labeled data or lexicons) having a more positive influence on this specific dataset, while CAN-NER cannot learn enough information from only the training set. However, we were not able to identify the precise contributors to the gap based on the available corresponding resources.

4 Related Work

4.1 Neural Network Models

Neural networks, such as LSTM and CNN, have been shown to outperform conventional machine learning methods without requiring handcrafted features. Collobert et al. (2011) describe a CNN-CRF model that reaches competitive results compared to the best statistical models at the time. More recently, the LSTM-CRF architecture has become a quasi-standard on NER tasks. Huang et al. (2015) employed BiLSTM to extract word-level context information and Lample et al. (2016) further introduced a hierarchy structure by incorporating BiLSTM-based character embeddings. Multiple recent works integrating word-level information and character-level information have been found to achieve improved performance (dos Santos et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016; Chen et al., 2019). Moreover, external knowledge has also been exploited for NER, as has character-level knowledge, both pre-trained (Peters et al., 2017) and co-trained (Liu et al., 2018). More recently, large-scale pre-trained language representations with deep language models have been proposed to help improve the performance of downstream NLP tasks. E.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018).

4.2 Attention Mechanism

Also, Attention Mechanisms have shown very good performance on a variety of tasks including machine translation, machine comprehension, and related NLP tasks (Vaswani et al., 2017; Seo et al., 2016; Tan et al., 2018a). In language understanding, Shen et al. (2018) exploit self-attention to learn long range dependencies. Rei et al. (2016) proposed a model employing an attention mechanism to combine the character-based representation with the word embedding instead of simply concatenating them. This method allows the model to dynamically decide which source of information to use for each word, and therefore outperforming the concatenation method used in previous work. More recently, Tan et al. (2018b) and Cao et al. (2018) employ self-attention to directly capture the global dependencies of the inputs for NER tasks and demonstrate the effectiveness of self-attention in Chinese NER.

4.3 Chinese NER

Multiple previous efforts have tried to address the Chinese language challenge of not having explicit word boundaries. Traditional models depended on hand-crafted features and CRFs-based models (He and Wang, 2008; Mao et al., 2008) and character-based LSTM-CRF models have been applied to Chinese NER to utilize both character- and radical-level representations (Dong et al., 2016). Peng and Dredze (2015) applied character positional embeddings and proposed a jointly trained model for embeddings and NER. To better integrate word boundary information into Chinese NER model, Peng and Dredze (2016) co-trained NER and word segmentation to improve performance in both tasks. He and Sun (2017b) unified cross-domain learning and semi-supervised learning to obtain information from out-of-domain corpora and in-domain unannotated text. Instead of performing word segmentation first, Recently, Zhang and Yang (2018) proposed constructing a word-character lattice by matching words in texts with a lexicon to avoid segmentation errors. Cao et al. (2018) use an adversarial network to jointly train Chinese NER task and Chinese Word Segmentation tasks to extract task-shared word boundary information. Also, Yang et al. (2018c) leverage character-level BiLSTM to extract higher-level features from crowd-annotations.

5 Conclusion

In this paper, we propose CAN-NER, a Convolutional Attention Network model to improve Chinese NER performance and preclude word embedding and additional lexicon dependencies; thus making the model more efficient and robust. In our model, we implement local-attention CNN and Bi-GRU with the global self-attention structure to capture word-level features and context information with char-level features. Extensive experiments show that our model outperforms the state-of-art systems on the different domain datasets.

Acknowledgements

We'd like to thank our colleague Börje Karlsson for his contribution and support in this work, as well as thank our colleagues Haoyan Liu, Zijia Lin, and the anonymous reviewers for their valuable feedback.

References

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 182–192.

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 52–62.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 173–176.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Börje F. Karlsson. 2019. GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of AAAI 2019*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 167–176.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics*, 4(1):357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence*, 6(4):325–346.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690.

Hangfeng He and Xu Sun. 2017a. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 713–718.

Hangfeng He and Xu Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI Conference on Artificial Intelligence*, pages 3216–3222.

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Shen Huang, Xu Sun, and Houfeng Wang. 2017. Addressing domain adaptation for chinese word segmentation with global recurrent structure. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 1, pages 184–193.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model.

Yanan Lu, Yue Zhang, and Dong-Hong Ji. 2016. Multi-prototype chinese character embedding. In *LREC*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1064–1074.

Xinnian Mao, Yuan Dong, Saike He, Sencheng Bao, and Haila Wang. 2008. Chinese word segmentation and named entity recognition based on conditional random fields. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1105–1116.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 149.

Nanyun Peng and Mark Dredze. 2017. Supplementary results for named entity recognition on chinese social media with an updated dataset. Technical report.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018a. Deep semantic role labeling with self-attention. In *AAAI Conference on Artificial Intelligence*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018b. Deep semantic role labeling with self-attention. In *AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Chunqi Wang and Bo Xu. 2017. Convolutional neural network with word embeddings for chinese word segmentation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 1, pages 163–172.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *AAAI Conference on Artificial Intelligence*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Fan Yang, Jianhu Zhang, Gongshen Liu, Jie Zhou, Cheng Zhou, and Huanrong Sun. 2018a. Five-stroke based cnn-birnn-crf network for chinese named entity recognition. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 184–195. Springer.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018b. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. *In CICLing*.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 839–849.

YaoSheng Yang, Meishan Zhang, Wenliang Chen, Wei Zhang, Haofen Wang, and Min Zhang. 2018c. Adversarial learning for chinese ner from crowd annotations. In *AAAI Conference on Artificial Intelligence*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 158–161.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1554–1564.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.

Decomposed Local Models for Coordinate Structure Parsing

Hiroki Teranishi[1] **Hiroyuki Shindo**[1] **Yuji Matsumoto**[1,2]
[1]Nara Institute of Science and Technology
[2]RIKEN Center for Advanced Intelligence Project (AIP)
`{teranishi.hiroki.sw5, shindo, matsu}@is.naist.jp`

Abstract

We propose a simple and accurate model for coordination boundary identification. Our model decomposes the task into three subtasks during training; finding a coordinator, identifying inside boundaries of a pair of conjuncts, and selecting outside boundaries of it. For inference, we make use of probabilities of coordinators and conjuncts in the CKY parsing to find the optimal combination of coordinate structures. Experimental results demonstrate that our model achieves state-of-the-art results, ensuring that the global structure of coordinations is consistent.

1 Introduction

Coordination is a frequently occurring structure that consists of conjuncts joined by a coordinator word. Since conjunct spans are one of the major ambiguities, identifying them is difficult, even for humans. For instance, in the sentence "*Toshiba's line of portables, for example, features the T-1000, which is in the same weight class* <u>*but*</u> *is much slower* <u>*and*</u> *has less memory,* <u>*and*</u> *the T-1600, which also uses a 286 microprocessor,* <u>*but*</u> *which weighs almost twice as much* <u>*and*</u> *is three times the size*," we cannot find correct conjuncts for each coordinator at a glance. The presence of coordination makes a sentence more ambiguous and longer, resulting in errors in syntactic parsing.

To identify the conjuncts of a given coordinator, previous studies have explored two properties of coordinate structures: (1) similarity – conjuncts tend to be similar; (2) replaceability – conjuncts can be replaced. Ficler and Goldberg (2016b) combine the syntactic parser and neural networks to compute the similarity and replaceability features of conjuncts. Teranishi et al. (2017) also exploit the two properties without deploying any syntactic parser, and achieve state-of-the-art results. Although both approaches outperform the similarity-based approaches (Shimbo and Hara, 2007; Hara et al., 2009), they cannot handle more than two conjuncts in a coordination, and multiple coordinations in a sentence at one time. Hence, their systems may produce coordinations that conflict with each other. In contrast, Hara et al. (2009) define production rules for coordination in order to output consistent coordinate structures.

Here, we propose a new framework for coordination boundary identification. We generalize a scoring function that takes a pair of spans with a coordinator and returns a higher score when the two spans appear to be coordinated. Using this function in the CKY parsing with production rules for coordination, our system produces globally consistent coordinations in a given sentence. To obtain such a function, we decompose the task into three independent subtasks – finding a coordinator, identifying the inner boundaries of a pair of conjuncts and delineating its outer boundaries. We use three different neural networks for the tasks, and the networks are trained on the basis of their local decisions. Our method is inspired by recent successes with locally-trained models for structured inference problems such as constituency parsing (Teng and Zhang, 2018) and dependency parsing (Dozat and Manning, 2017) without globally-optimized training. Experimental results reveal that our model outperforms existing systems and our strong baseline, an extension of Teranishi et al. (2017), and ensures that the global structure of the coordinations is consistent.

In summary, our contributions include the following:

- We propose a simple framework that trains a generalized scoring function of a pair of conjuncts and uses it for inference.
- We decompose the task and use three local models that interoperate for the CKY parsing.

Proceedings of NAACL-HLT 2019, pages 3394–3403
Minneapolis, Minnesota, June 2 - June 7, 2019.

- We establish a system that can accommodate more than two conjuncts in a sentence.
- Our system outperforms existing ones, particularly because it produces globally consistent coordinate structures.

2 Coordination Boundary Identification

2.1 Coordinate Structure

A *coordinate structure* or *coordination* is a syntactic structure in which two or more elements, known as *conjuncts*, are linked by *coordinator*(s). In addition to coordinating words, such as "and," "or," or "but," some punctuation marks function secondarily to connect two conjuncts. We refer to those punctuation marks as *sub-coordinator*s. Sub-coordinators cannot independently conjoin phrases to form a coordinate structure. The presence of a coordination is usually signaled by the appearance of a coordinator; however, coordinating words do not always lead to coordinations. For instance, "but" is not a coordinator when it functions as a preposition. In this paper, we refer to a word that can be a coordinator or sub-coordinator as a *coordinator key*.

2.2 Task Definition and Difficulties

The task of coordination boundary identification is to find conjunct spans of a given coordinating word. If a coordinating word does not act as a coordinator, a system must return NONE; denoting the absence of a coordinate structure. The difficulties in this task arise when there are multiple coordinate structures in a sentence or more than two conjuncts in a single coordinate structure. If there is more than one coordinate structure in a sentence, each coordinate structure must be isolated from the others or integrated into the other(s). In other words, coordinate structures cannot be partially overlapped. When there are more than two conjuncts in a coordinate structure, it has to be ascertained whether the punctuation marks are sub-coordinators that bring one more conjunct, and if so, which coordinate structure they belong to. Thus, we must identify how many conjuncts a coordinate structure contains and the location of those conjuncts in the coordinate structure — whether it is nested in or isolated from other coordinate structures.

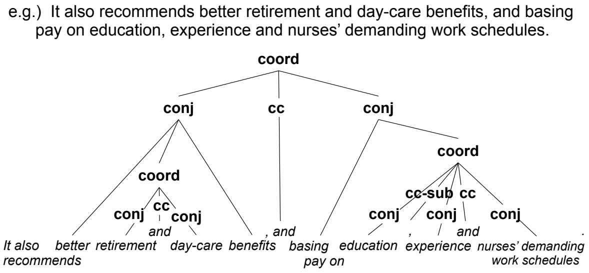


Figure 1: An example of a coordinate tree.

2.3 Coordinate Structures as a Tree

Invoking Shimbo and Hara (2007), we use a tree to represent the coordinate structures in a sentence. We call this tree a **coordinate tree**. Figure 1 shows an example of a coordinate tree. Tree structures are particularly suitable because the ranges of coordinate structures are always consistent, and conjuncts are shown as nodes without being limited by the frequency of their occurrence. Our system produces a coordinate tree using the CKY algorithm and then retrieves well-formed coordinate structures from the tree. In this work, we focus on how to learn the scoring function that assigns higher scores to probable pairs of conjuncts for the CKY parsing.

3 Proposed Method

Our proposed model consists of three parts: a coordinator classifier and the inner and outer-boundary scoring models. Figure 2 is the overview of our framework. The coordinator classifier is a binary classifier that ascertains whether a word functions as a coordinator or not. The inner-boundary scoring model computes the score for a pair of conjuncts on the basis of their boundaries that are in proximity to a coordinator. This means that the model produces a score based on the end of the left conjunct and the beginning of the right conjunct. Similarly, the outer-boundary scoring model assigns a score to a pair of the beginning of the left conjunct and the end of the right conjunct. Using the inner and outer-boundary scoring models, our model calculates all possible combinations of the four boundaries, and then produces their probabilities. Given the local probabilities, we run the CKY algorithm to find the globally optimal coordinate structures in the sentence. In this section, we formulate our model based on the details of the neural networks' architecture; afterward, we describe the parsing method.

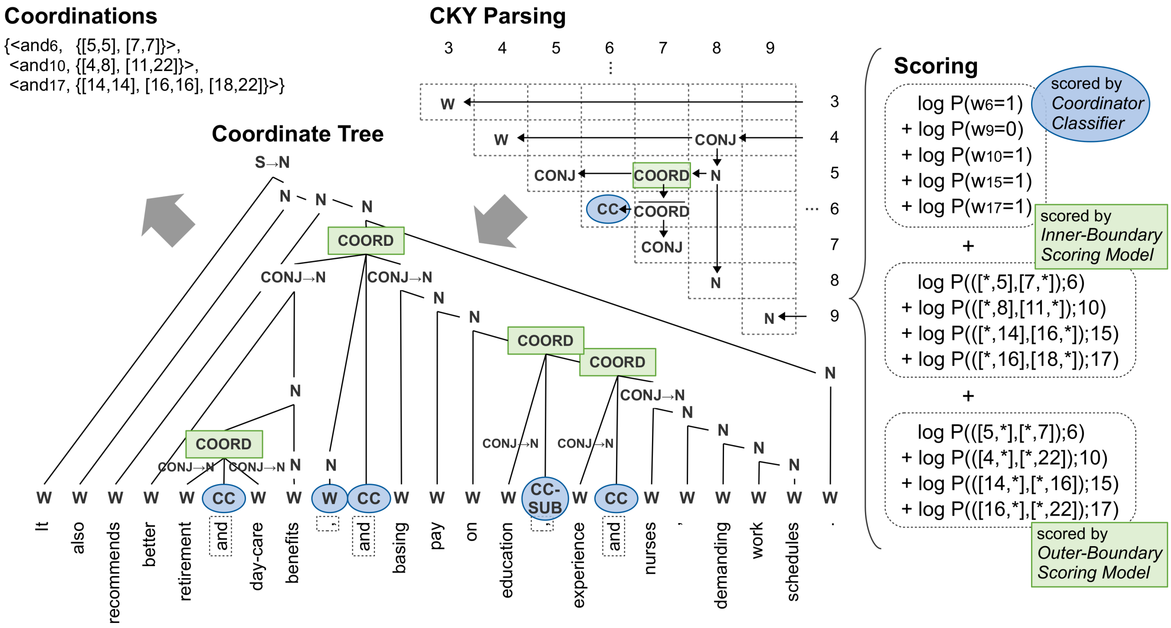


Figure 2: Overview of the proposed framework for coordination boundary identification. The scores of circular nodes are assigned by the coordinator classifier, and the scores of rectangular nodes are assigned by the inner and outer-boundary scoring models.

3.1 Model

Given a sentence that consists of N words $w_{1:N} = w_1, \ldots, w_N$ with the corresponding part-of-speech (POS) tags $p_{1:N} = p_1, \ldots, p_N$, our model outputs a set of coordinate structures $\{\langle c, \{[b_1, e_1], \ldots, [b_n, e_n]\}\rangle\}(n \geq 2)$ where c is a coordinator and $[b_k, e_k]$ is the k-th conjunct spanning from the b_k-th word to the e_k-th word. Although we cannot know the number of coordinate structures and conjuncts in each coordinate structure, we can use coordinator keys as clues to find pairs of conjuncts. Our model tries to find pairs of conjuncts, rather than coordinate structures, in a sentence.

$$\begin{aligned} X &= \{w_{1:N}, p_{1:N}, C\} \\ C &= \{t | w_t \in \mathbb{S}_{cc} \cup \mathbb{S}_{\textit{sub-cc}}\} \\ Y &= \{\langle y_t^{ckey}, y_t^{pair}\rangle | t \in C\} \end{aligned} \tag{1}$$

where y_t^{ckey} is a label that indicates whether w_t is the actual coordinator ($y_t^{ckey} = 1$) or not ($y_t^{ckey} = 0$), and y_t^{pair} is a pair of conjunct spans. $y_t^{pair} = \varnothing$ when $y_t^{ckey} = 0$. When $t = 1$ or $t = N$, $y_t^{ckey} = 0$ because it does not form a coordinate structure within the sentence. In this paper, we define $\mathbb{S}_{cc}$ and $\mathbb{S}_{\textit{sub-cc}}$ as {"and", "or", "but", "nor", "and/or"} and {",", ";", ":"}, respectively. We use two different models to identify inner and outer boundaries of y_t^{pair}, because enumerating all possible inner and outer boundaries of y_t^{pair} requires time complexity $\mathcal{O}(N^2) + \mathcal{O}(N^2) = \mathcal{O}(N^2)$, whereas enumerating all possible y_t^{pair} requires time complexity $\mathcal{O}(N^4)$[1].

Coordinator Classifier

The coordinator classifier is a binary classifier that predicts the label of a coordinator key.

$$P(y_t^{ckey} | w_t, \theta) = \mathrm{softmax}(f_{ckey}(w_t)) \tag{2}$$

The training loss of the binary classification is computed by the following equation:

$$\ell_\theta^{ckey}(X, Y) = - \sum_{\langle y_t^{ckey}, y_t^{pair}\rangle \in Y} \log P(y_t^{ckey} | w_t, \theta) \tag{3}$$

Inner-Boundary Scoring Model

The inner-boundary scoring model assigns a score to a pair of conjunct spans on the basis of inner boundaries. We use b^l, e^l, b^r, e^r to denote the beginning of a left conjunct, the end of the left conjunct, the beginning of a right conjunct, and the end of the right conjunct, respectively. The score

---

[1] For division of four boundaries, "two beginnings and two ends" or "left span and right span" can be chosen instead. In preliminary experiments, "left span and right span" models perform poorly, and "two beginnings and two ends" models perform well, but worse than "inner and outer-boundary" models.

of the inner-boundary pair (e^l, b^r) for a coordinator key w_t is calculated as follows:

$$\textsc{Score}_{\theta}^{inner}(e^l, b^r, w_t) = f_{inner}(e^l, b^r, w_t) \tag{4}$$

The probabilities of the inner boundaries are normalized distributions over all possible inner boundary pairs:

$$\begin{aligned} I_{w_t} = \{&(1, t+1), (1, t+2), \ldots, (1, N), \\ &(2, t+1), \ldots, (t-1, N)\} \end{aligned} \tag{5}$$

$$\begin{aligned} &P(y_t^{pair} = ([*, e^l], [b^r, *]) | w_t, \theta) = \\ &\frac{\exp\left(\textsc{Score}_{\theta}^{inner}(e^l, b^r, w_t)\right)}{\sum\limits_{(e'^l, b'^r) \in I_{w_t}} \exp\left(\textsc{Score}_{\theta}^{inner}(e'^l, b'^r, w_t)\right)} \end{aligned} \tag{6}$$

$$\begin{aligned} &\ell_{\theta}^{inner}(X, Y) = \\ &\quad - \sum_{\langle y_t^{ckey}, y_t^{pair} \rangle \in Y} y_t^{ckey} \log P(y_t^{pair} | w_t, \theta) \end{aligned} \tag{7}$$

The term $y_t^{ckey} \log P(y_t^{pair} | w_t, \theta)$ means the cross-entropy loss is activated only for positive coordinator keys ($y_t^{ckey} = 1$) and is disabled otherwise ($y_t^{ckey} = 0$).

Outer-Boundary Scoring Model

Similarly to the inner-boundary scoring model, we define the probability $P(y_t^{pair} = ([b^l, *], [*, e^r]) | w_t, \theta)$ based on the set of all the outer-boundary pairs O_{w_t}; the loss is defined as ℓ_{θ}^{outer} using the scoring function $\textsc{Score}_{\theta}^{outer}(b^l, e^r, w_t) = f_{outer}(b^l, e^r, w_t)$. Note that I_{w_t} and O_{w_t} are identical because their possible pairs are the same. Based on the inner pair probability $P(y_t^{pair} = ([*, e^l], [b^r, *]) | w_t, \theta)$ and the outer pair probability $P(y_t^{pair} = ([b^l, *], [*, e^r]) | w_t, \theta)$, the most probable pair is produced by:

$$\begin{aligned} y_t^{pair} = &\mathop{\arg\max}_{(\hat{e}^l, \hat{b}^r)} P(([*, \hat{e}^l], [\hat{b}^r, *]) | w_t, \theta) \\ &\cup \mathop{\arg\max}_{(\hat{b}^l, \hat{e}^r)} P(([\hat{b}^l, *], [*, \hat{e}^r]) | w_t, \theta) \end{aligned} \tag{8}$$

3.2 CKY Parsing

Our three models predict coordinators including sub-coordinators, and the inner and outer boundaries of their coordinating conjuncts. Such local predictions may cause conflicts between different coordinate structures. Furthermore, two conjuncts linked by a sub-coordinator must be embedded in another coordinate structure formed by a coordinator. To overcome these limitations, we use the CKY algorithm to find the optimal coordinations in a sentence. In particular, we define the CFG rules to produce a coordinate tree, as used in Hara et al. (2009). Our CFG rules, distinct from those of Hara et al. (2009) [2], are shown in Table 1. Based on these rules, we can map a coordinate tree to the one-to-one corresponding syntactic tree, covering 99.5% coordinations in the Penn Treebank [3].

Non-terminals			
COORD	Coordination		
CONJ	Conjunct		
CC	Coordinating conjunction		
CC-SUB	Sub-coordinator		
W	Word		
N	Non-coordination		
S	Sentence		
Rules for coordinations			
(1)	COORD	→	CONJ N? CC N? CONJ
(2)	COORD	→	CONJ CC-SUB COORD
(3)	CONJ	→	COORD
(4)	CONJ	→	N
Rules for non-coordinations			
(5)	S	→	COORD
(6)	S	→	N
(7)	N	→	COORD N
(8)	N	→	W COORD
(9)	N	→	W N
(10)	N	→	W
Rules for pre-terminals			
(11)	CC	→	(and|or|but|nor|and/or)
(12)	CC-SUB	→	(,|;|:)
(13)	W	→	*

Table 1: Production rules for coordinate trees. (. . . | . . .) matches one of the elements and "*" matches any word. "?" indicates zero or one occurrence of the preceding element.

3.2.1 Scoring

We give scores only to coordination nodes denoted as COORD, and pre-terminals. When scoring pre-terminals, we assign $\log P(w_k = 1)$ to CC and CC-SUB, and $\log(P(w_k = 0))$ to W if $w_k \in \mathbb{S}_{cc} \cup \mathbb{S}_{sub\text{-}cc}$, otherwise 0. When scoring the CO-

---

[2] Our rules can produce coordinate structures that contain arbitrary length phrase(s) around coordinators, while conjuncts always appear next to coordinators in their rules.

[3] Most of the non-derivable coordinations are in the form like "A and B and C" where a coordinating word is regarded as a sub-coordinator. Even so, this expression can be parsed as a nested coordinate structure by the rules.

ORD, we take the left conjunct and the right conjunct which are linked by the CC. Thus, in the rule (2), the conjunct pair linked by a CC-SUB is the incoming CONJ and the leftmost CONJ in the child COORD. Using a coordinator and its pair of conjuncts, we assign $\log P(([i,j],[l,m])) = \log P(([*,j],[l,*])) + \log P(([i,*],[*,m]))$ to the COORD. The best scoring coordinate tree can be found efficiently using dynamic programming with time complexity $\mathcal{O}(N^3)$.

3.3 Neural Network Models

We use neural networks as instantiations of f_{ckey}, f_{inner}, and f_{outer} that we have introduced in this section.

Encoder

To get sentence-level representations for a sequence of words and POS tags, we use bidirectional long short-term memories (BiLSTMs) (Hochreiter and Schmidhuber, 1997).

$$\mathbf{h}_{1:N} = \mathrm{BiLSTMs}(f_{input}(w_{1:N}, p_{1:N})) \qquad (9)$$

The dimensionality of each resulting vector $\mathbf{h}_t$ is $2d^{hidden}$. For the BiLSTMs inputs, we use f_{input} to map words and POS tags onto their representations. We can use different word representations including a pretrained word model, ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) or character-level LSTMs/convolutional neural networks (CharCNNs). We demonstrate the differences between the different choices in Section 4. The entire network consisting of f_{input} and BiLSTMs is referred to as the encoder; it is shared by the three neural networks in the higher layer.

Coordinator Classifier

We use a linear transformation of the sentence-level representation of a coordinator key for f_{ckey}.

$$f_{ckey}(w_t) = \mathbf{W}^{ckey}\mathbf{h}_t + \mathbf{b}^{ckey} \qquad (10)$$

where $\mathbf{W}^{ckey} \in \mathbb{R}^{2\times 2d^{hidden}}$ and $\mathbf{b}^{ckey} \in \mathbb{R}^2$ are the model parameters of the classifier.

Inner-Boundary Scoring Model

From the sentence-level representations produced by the encoder, the inner-boundary scoring model concatenates two representations of inner boundaries, and then feeds the produced vector into a multilayered perceptron (MLP).

$$\begin{aligned} & f_{inner}(e^l, b^r, w_t) = \\ & \quad \mathbf{w}_2^{in}\,\mathrm{ReLU}(\mathbf{W}_1^{in}[\mathbf{h}_{e^l}; \mathbf{h}_{b^r}] + \mathbf{b}_1^{in}) + \mathrm{b}_2^{in} \end{aligned} \qquad (11)$$

where $\mathbf{W}_1^{in} \in \mathbb{R}^{d^{in}\times 4d^{hidden}}$, $\mathbf{b}_1^{in} \in \mathbb{R}^{d^{in}}$, $\mathbf{w}_2^{in} \in \mathbb{R}^{d^{in}}$ and $\mathrm{b}_2^{in} \in \mathbb{R}^1$ are the parameters of the inner-boundary scoring model.

Outer-Boundary Scoring Model

Using sentence-level representations, the outer-boundary scoring model takes two vectors that are calculated by subtracting the adjacent vectors to the coordinator from the boundary vectors. These subtraction operations are intended to capture the semantic distance and relatedness between two spans (Teranishi et al., 2017). The model then passes the vector to a MLP.

$$\begin{aligned} & f_{feature}(b^l, e^r, w_t, \mathbf{h}_{1:N}) = \\ & \quad [\mathbf{h}_{b^l} - \mathbf{h}_{t+1}; \mathbf{h}_{e^r} - \mathbf{h}_{t-1}] \end{aligned} \qquad (12)$$

$$\begin{aligned} & f_{outer}(b^l, e^r, w_t) = \\ & \quad \mathbf{w}_2^{out}\,\mathrm{ReLU}(\mathbf{W}_1^{out}\,\mathbf{r}) + \mathbf{b}_1^{out}) + \mathrm{b}_2^{out} \\ & \quad \mathbf{r} = f_{feature}(b^l, e^r, w_t, \mathbf{h}_{1:N}) \end{aligned} \qquad (13)$$

where $\mathbf{W}_1^{out} \in \mathbb{R}^{d^{out}\times 4d^{hidden}}$, $\mathbf{b}_1^{out} \in \mathbb{R}^{d^{out}}$, $\mathbf{w}_2^{out} \in \mathbb{R}^{d^{out}}$ and $\mathrm{b}_2^{out} \in \mathbb{R}^1$ are the parameters of the outer-boundary scoring model.

3.4 Learning

To train the set of parameter θ of our neural networks, we minimize the following loss function:

$$\begin{aligned} L(\theta) = \sum_{(X,\hat{Y})\in D} \Big(& \ell_\theta^{ckey}(X,\hat{Y}) \\ & + \ell_\theta^{inner}(X,\hat{Y}) \\ & + \ell_\theta^{outer}(X,\hat{Y}) \Big) \end{aligned} \qquad (14)$$

where D is a set of pairs of a sentence and its correct coordinate structures in a training dataset. Thus, our submodels are trained jointly.

Why local training?

Instead of learning the scoring functions on the basis of local decisions, we can directly train our models combined with the CKY parsing using a structured max-margin objective between the scores of the best predicted and gold trees. In preliminary experiments, however, such a global training requires careful hyperparameter tuning and is hard to optimize stably, resulting in slightly better performance than the method of Teranishi et al. (2017).

4 Experiments

4.1 Settings

4.1.1 Datasets

We use the coordination-annotated Penn Treebank (Ficler and Goldberg, 2016a) (PTB) and Genia Treebank beta (Kim et al., 2003) (GENIA). Unlike the evaluation by Teranishi et al. (2017) and Ficler and Goldberg (2016b), we strip the PTB of all quotation marks (“) and (”) to normalize irregular coordinations such as ⟨... *“Daybreak,” “Daywatch,” “Newsday,” and “Newsnight,”* ... ⟩. We follow the standard train/development/test split on the PTB. For the GENIA, we do not apply the preprocessing described above. We evaluate the model through a five-fold cross-validation, as in Hara et al. (2009).

4.1.2 Model

We use pretrained word vectors, POS tags, and character vectors produced by the CharCNN (Ma and Hovy, 2016), regarded as the *default*. We also investigate the performance of the model, using three different word representations for the encoder: (1) pretrained word embeddings; GloVe (Pennington et al., 2014) for the PTB, BioASQ (Tsatsaronis et al., 2012) for the GENIA, (2) contextualized sentence embeddings; ELMo, (3) randomly initialized word vectors. For the PTB, POS tags are obtained using the Stanford POS Tagger (Toutanova et al., 2003) with 10-way jackknifing. For the GENIA, we use the gold POS tags, as in Hara et al. (2009). To optimize the model parameters, we use Adam (Kingma and Ba, 2015). Other hyperparameters are described in Appendix A.

4.1.3 Baseline Model

We adopt our implementation of Teranishi et al. (2017) as the baseline. The original model of Teranishi et al. (2017) predicts the beginning and the end of a coordinate structure, and then splits it into conjuncts by commas. Their model decides the boundary of a coordinate structure individually, which may cause conflicts with that of other coordinate structure(s). Thus, we extend their model to find the best combination of coordinate structures, greedily choosing most probable boundaries without conflicts[4]. For the baseline model, we use the same encoder as that of our default model. Hereinafter, we refer to this baseline model as *Teranishi+17:+ext*.

4.1.4 Evaluation

We evaluate the systems on the basis of the ability to predict conjunct spans with the precision, recall, and F1 measures on the PTB. To compare the performance of our model with Teranishi et al. (2017), we adjudge the predicted conjuncts correct based on the following metrics.

- **whole**: matches at the beginning of the first conjunct and the end of the last conjunct.
- **outer**: matches in the first conjunct and the last conjunct.
- **inner**: matches in the two conjuncts adjacent to the coordinator.
- **exact**: matches in all the conjuncts.

In addition, we pay particular attention to the evaluation of NP coordination.

For the GENIA, we measure the recall values of coordinate structures by the aforementioned metrics; previous studies, on the other hand, evaluated their systems based only on the whole metric. Also, we evaluate the performance of our model based on syntactic categories.

4.2 Results

Tables 2 and 3 show the experimental results on the PTB and GENIA datasets. On the PTB, our model outperforms the baseline and existing methods for all metrics. We cannot compare its performance with that of existing methods because of its use of the preprocessing for quotation marks; nevertheless, our model achieves significant improvements. Our model is more accurate than the baseline because ours learns both the inner and outer boundaries of conjunct pairs including those of sub-coordinators, while the baseline learns only the coordination boundaries. On the GENIA, our model also outperforms the baseline on the exact metric. While our model has some limitations when it comes to predicting the beginning and the end of coordinations, it performs better on the inner metric. In contrast, Teranishi+17:+ext achieves the best results on the whole metric, whereas it performs poorly on the other metrics. This performance reflects the differences between the algorithms of the two systems. Our model builds a coordinate tree in a bottom-up manner and predicts inner conjuncts accurately. On the other hand, the baseline model predicts the entire span of a coordinate structure and splits them into conjuncts in a top-down fashion. That is why the base-

---

[4] We did not add the constraint to situate a nested coordination in the parent conjunct.

		Development						Test					
		All			NP			All			NP		
		P	R	F	P	R	F	P	R	F	P	R	F
Ours	whole	78.60	78.41	78.51	79.26	78.71	78.98	76.88	77.16	77.02	78.75	78.50	78.62
	outer	77.18	77.00	77.09	78.57	78.03	78.30	75.33	75.61	75.47	77.95	77.70	77.83
	inner	79.19	79.00	79.10	80.64	80.09	80.36	77.60	77.88	77.74	80.19	79.93	80.06
	exact	**76.95**	**76.76**	**76.85**	**78.11**	**77.57**	**77.84**	**75.33**	**75.61**	**75.47**	**77.95**	**77.70**	**77.83**
Teranishi+17 :+ext	whole	78.78	77.94	78.36	78.52	77.80	78.16	77.36	76.52	76.94	78.72	78.34	78.53
	outer	74.49	73.70	74.09	76.67	75.97	76.32	72.03	71.24	71.63	75.36	75.00	75.17
	inner	76.04	75.23	75.63	77.82	77.11	77.47	74.14	73.33	73.74	77.44	77.07	77.25
	exact	74.13	73.34	73.74	76.21	75.51	75.86	71.48	70.70	71.08	75.20	74.84	75.01
Teranishi+17*	whole	75.92	72.87	74.36	77.90	75.05	76.45	-	-	-	-	-	-
	outer	72.48	69.57	70.99	76.24	73.45	74.82	-	-	-	-	-	-
	inner	74.07	71.10	72.56	77.43	74.59	75.99	73.46	72.16	72.81	75.87	74.76	75.31
	exact	72.11	69.22	70.63	75.77	72.99	74.35	-	-	-	-	-	-
Ficler+16*	inner	72.34	72.25	72.29	75.17	74.82	74.99	72.81	72.61	72.7	76.91	75.31	76.1

Table 2: Evaluation per coordination by the different metrics. Preprocessing for quotation marks are not reported in "Teranishi+17" and "Ficler+16".

		NP	VP	ADJP	S	PP	UCP	SBAR	ADVP	Others	All
	#	2317	465	321	188	167	60	56	21	3	3598
Ours	whole	59.30	65.16	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.31
	outer	59.21	64.94	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.22
	inner	70.60	67.74	81.61	55.31	55.68	53.33	69.64	90.47	33.33	69.51
	exact	**59.21**	**64.94**	**78.19**	**53.19**	**55.68**	**48.33**	**66.07**	**90.47**	**0.00**	**61.22**
Teranishi+17 :+ext	whole	67.19	63.65	76.63	53.19	61.67	35.00	78.57	85.71	33.33	66.31
	outer	57.14	54.83	72.27	8.51	55.68	28.33	57.14	85.71	0.00	55.22
	inner	57.61	54.83	72.27	8.51	55.68	28.33	57.14	85.71	0.00	55.53
	exact	57.14	54.83	72.27	8.51	**55.68**	28.33	57.14	85.71	**0.00**	55.22
Teranishi+17	whole	66.59	63.87	78.50	52.65	53.89	50.00	78.57	85.71	33.33	65.98
Ficler+16	whole	65.08	71.82	74.76	17.02	56.28	51.66	91.07	80.95	33.33	64.14
Hara+09	whole	64.2	54.2	80.4	22.9	59.9	36.7	51.8	85.7	66.7	61.5

Table 3: Recall with Genia Treebank beta. The numbers in the rows "Teranishi+17," "Ficler+16" and "Hara+09" are taken from their papers.

line model cannot predict coordinated clauses labeled as "S," that are likely to be longer and to contain non-coordinating commas. The shortcoming of our model is that our bottom-up parsing may cause errors due to wrong decisions in the early stage of the parsing; this is observed as poor performance in the whole metric.

4.3 Analysis

Complete match in a sentence

We investigate the ability of our system to predict all the coordinate structures in a sentence precisely. We categorize sentences into the following four groups[5].

All: All sentences that have any coordinate structure.

- **Simple**: Sentences that have only one coordinate structure consisting of two conjuncts.
- **Complex**: Sentences that are categorized as Consecutive and/or Multiple.
  - **Consecutive**: Sentences that have a coordinate structure consisting of more than two conjuncts.
  - **Multiple**: Sentences that have multiple coordinate structures.

Sentences categorized as "All" are the union of the mutually exclusive sets of Simple and Complex.

Table 4 shows complete match rates on the PTB. Both on the development and test sets, our system records significant gain, in comparison to Teranishi+17:+ext, on Simple coordination sentences. It might be because the inner and outer-boundary scoring models learn to predict four boundaries of two spans, whereas the baseline model predicts only two outer boundaries on Simple coordination sentences. Since an appositive or adverbial phrase can appear between a coordinator and its conjunct, the assumption that two conjuncts must be next to

---

[5] *Consecutive* and *Multiple* both contain sentences that are Consecutive and Multiple.

Model	Sentence	Development	Test
Ours	All	489 / 673 = **72.65**	619 / 873 = **70.90**
	- Simple	378 / 481 = **78.58**	476 / 609 = **78.16**
	- Complex	111 / 192 = **57.81**	143 / 264 = **54.16**
	- Consecutive	41 / 66 = **62.12**	56 / 96 = **58.33**
	- Multiple	79 / 146 = **54.10**	96 / 197 = **48.73**
Teranishi+17 :+ext	All	468 / 673 = 69.53	577 / 873 = 66.09
	- Simple	358 / 481 = 74.42	444 / 609 = 72.90
	- Complex	110 / 192 = 57.29	133 / 264 = 50.37
	- Consecutive	40 / 66 = 60.60	48 / 96 = 50.00
	- Multiple	78 / 146 = 53.42	92 / 197 = 46.70

Table 4: Complete match rates of coordinations per sentence.

	All (exact)			All (inner)
	P	R	F	F
default	76.95	76.76	76.85	79.10
-POS tags	71.59	71.34	71.47	74.42
-CharCNNs	76.41	76.41	76.41	78.53
-GloVe	75.05	75.23	75.14	77.03
+ELMo	76.35	76.17	76.26	78.15
concat feature	74.85	74.41	74.63	76.64

Table 5: Performance comparison between different settings of the proposed models.

a coordinator fails and causes errors. Our system also outperforms Teranishi+17:+ext on Consecutive and Multiple coordination sentences. Teranishi+17:+ext predicts a coordination span, and then splits it into conjunct spans. Therefore, it can mistakenly segment coordinations when false sub-coordinators appear in a sentence. In contrast, our approach ascertains whether sub-coordinating words are true sub-coordinators; thus, it can lead to more robust production of Consecutive sentences.

What helps for Coordination Parsing?

We conduct an ablation study for our model. Table 5 shows the results. Without the POS tags, the model performs poorly. It is worthy of note that the pretrained word embedding is beneficial information for the task. On the other hand, the use of contextual embedding, ELMo, does not improve performance. We deduce that POS tags and morphological information, and not contextual word senses, are clues for shorter and similar coordinations such as NP coordinations. For the feature extraction function of the outer-boundary scoring model, the *concat* function that performs the same function as the inner-boundary scoring model does not achieve competitive advantage. The feature function described as Eq. 12 is designed to capture the similarity and replaceability of two spans; while the *concat* function has only the contextual information of the outer boundaries of a pair.

5 Related Work

5.1 Similarity-based Approaches

For the coordination identification task in Japanese, Kurohashi and Nagao (1994) used a chart to find the highest similarity pair of conjuncts using dynamic programming. Hogan (2007) developed a generative parsing model for coordinated noun phrases, incorporating symmetry in conjunct structures and head words. Shimbo and Hara (2007) proposed a discriminative model that computes scores based on the syntactic and morphological features assigned to edges and nodes in a sequence alignment. While their method focused on non-nested coordinations, Hara et al. (2009) extended their work to accommodate nested coordinations using CFG rules. A consistent global structure of coordinations is produced using discriminative functions based on the similarity of conjuncts with dynamic programming. Our concept of the CKY parsing is borrowed from their work; however, a key difference of our approach lies in how it computes the score of conjuncts and trains the score function. Hanamoto et al. (2012) used dual decomposition to combine HPSG parsing with the discriminative model developed by Hara et al. (2009).

5.2 Non Similarity-based Approaches

Kawahara and Kurohashi (2008) focused on resolving the ambiguities of coordinate structures without the use of any similarities. Their method relied on the dependency relations surrounding the conjuncts and the generative probabilities of phrases. Yoshimoto et al. (2015) extended the Eisner algorithm by adding new rules to accommodate coordinations during dependency parsing.

5.3 Coordination Boundary Identification using Neural Networks

Ficler and Goldberg (2016b) used neural networks for the coordination boundary identification task. They incorporated the replaceability property between conjuncts, in addition to the similarity property, in the computation of a score for a pair of conjuncts. They first used a binary classifier for coordinating words; then, they extracted probable candidate pairs of conjuncts using the Berkeley Parser (Petrov et al., 2006); afterward, they assigned scores to the pairs using neural networks. However, the shortcoming of their work is that it is highly dependent on the external parser. The work of Teranishi et al. (2017) developed an end-to-end model, as opposed to the pipeline approach of Ficler and Goldberg (2016b). They also used similarity and replaceability feature representations without information from a syntactic parser. While Ficler and Goldberg (2016b) cut off improbable pairs of conjuncts ahead of training, Teranishi et al. (2017) calculated scores for all possible pairs of the beginning and the end of coordinate structures instead of conjuncts. We apply the same strategy to the inner-boundary pairs and the outer-boundary pairs because assigning low probabilities to improbable inner and outer pairs makes the model robust for the CKY parsing.

6 Conclusion

We proposed a simple and accurate model for coordination boundary identification. Our system decomposes this task into three subtasks, and uses three different neural networks to tackle them. For inference, the CKY algorithm is applied using the CFG rules in order to produce globally consistent coordinate structures in a sentence. Experimental results demonstrated that our locally-trained models interoperate to obtain the optimal combination of coordinate structures and outperform existing systems and the strong baseline. Through empirical analysis, we found that our system performs better than the baseline in complete matches of sentences that contain more than two conjuncts and/or multiple coordinations.

Acknowledgments

This work was partly supported by JST CREST Grant Number JPMJCR1513 and JSPS KAKENHI Grant Number 18K18109. We are grateful to the anonymous reviewers for their helpful insights and comments.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Jessica Ficler and Yoav Goldberg. 2016a. Coordination annotation extension in the penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, Berlin, Germany. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016b. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32, Austin, Texas. Association for Computational Linguistics.

Atsushi Hanamoto, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Coordination structure analysis using dual decomposition. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 430–438, Avignon, France. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 680–687, Prague, Czech Republic. Association for Computational Linguistics.

Daisuke Kawahara and Sadao Kurohashi. 2008. Coordination disambiguation without any similarities. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 425–432, Manchester, UK. Coling 2008 Organizing Committee.

Jin Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 610–619, Prague, Czech Republic. Association for Computational Linguistics.

Zhiyang Teng and Yue Zhang. 2018. Two local models for neural constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 119–132. Association for Computational Linguistics.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 264–272, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Tsatsaronis, Michael Schroeder, Georgios Paliouras, Yannis Almirantis, Ion Androutsopoulos, Eric Gaussier, Patrick Gallinari, Thierry Artieres, Michael R Alvers, Matthias Zschunke, and Axel-Cyrille Ngonga Ngomo. 2012. Bioasq: A challenge on large-scale biomedical semantic indexing and question answering. In *AAAI Fall Symposium Series: Information Retrieval and Knowledge Discovery in Biomedical Text*.

Akifumi Yoshimoto, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. 2015. Coordination-aware dependency parsing (preliminary report). In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 66–70, Bilbao, Spain. Association for Computational Linguistics.

A Hyperparameters

Name	Value
Dimention of the word embeddings (GloVe)	100
Dimention of the word embeddings (BioASQ)	200
Dimention of the POS tag embeddings	50
Dimention of the character embeddings in the CharCNNs	10
Window size of the the CharCNNs	5
Dimention of the produced representation from the CharCNNs	50
Dimension of the LSTM hidden vector d^{hidden}	512
Number of BiLSTMs layers	2
MLP units in the hidden layer d^{in}	1024
MLP units in the hidden layer d^{out}	1024
Dropout ratio (all)	0.50
Initial learning rate	0.001
Regularization term λ (PTB)	0.0
Regularization term λ (GENIA)	0.0001
Gradient clipping threshold	5.0

Table 6: The final hyperparameters used in the experiments.

Multi-task Learning for Japanese Predicate Argument Structure Analysis

Hikaru Omori
Tokyo Metropolitan University
omori-hikaru@ed.tmu.ac.jp

Mamoru Komachi
Tokyo Metropolitan University
komachi@tmu.ac.jp

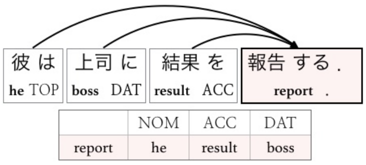
Abstract

An *event-noun* is a noun that has an argument structure similar to a predicate. Recent works, including those considered state-of-the-art, ignore event-nouns or build a single model for solving both Japanese predicate argument structure analysis (PASA) and event-noun argument structure analysis (ENASA). However, because there are interactions between predicates and event-nouns, it is not sufficient to target only predicates. To address this problem, we present a multi-task learning method for PASA and ENASA. Our multi-task models improved the performance of both tasks compared to a single-task model by sharing knowledge from each task. Moreover, in PASA, our models achieved state-of-the-art results in overall F1 scores on the NAIST Text Corpus. In addition, this is the first work to employ neural networks in ENASA.

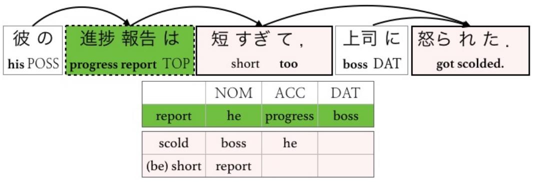
1 Introduction

Japanese predicate argument structure analysis (PASA) examines semantic structures between the predicate and its arguments in a text. The identification of the argument structure such as "who did what to whom?" is useful for natural language processing that requires deep analysis of complicated sentences such as machine translation and recognizing textual entailment. PASA is a task targeted at predicates such as verbs and adjectives. However, there are also many nouns that have event-related arguments in a sentence. We call these nouns that refer to events *event-nouns*, for example, a verbal noun (*sahen* nouns) such as *houkoku* "report" or a deverbal noun (nominalized forms of verbs) such as *sukui* "rescue."

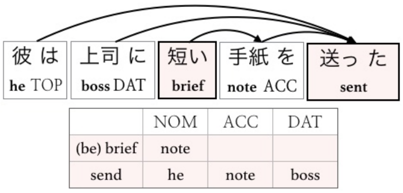
Figure 1 shows examples of PASA and event-noun argument structure analysis (ENASA). In the NAIST Text Corpus (Iida et al., 2007), both predicates and event-nouns have one of three core case roles, *nominative* (NOM), *accusative* (ACC), and *dative* (DAT) as an argument. According to Iida et al. (2007), predicates have almost no argument in the same *bunsetsu*[1] phrase. However, in the case of event-nouns, approximately half of the accusative and dative arguments appear in the same bunsetsu phrase. Accordingly, although PASA and ENASA are semantically highly related, they are syntactically different tasks. However, most previous studies focused on predicates only; hence, there are few studies that focus



(a) He reports the result to his boss.



(b) His progress report was too short; hence, he got scolded by his boss.



(c) He sent a brief note to his boss.

Figure 1: Examples of PASA and ENASA. The edges denote dependency paths.

---

[1] Functional chunk in Japanese. It consists of one or more content words (noun, verb, adjective, etc.) followed by zero or more function words (postposition, auxiliary verb, etc.). A verb phrase in Japanese thus cannot bear noun arguments in the same bunsetsu.

Proceedings of NAACL-HLT 2019, pages 3404–3414
Minneapolis, Minnesota, June 2 - June 7, 2019.

on event-nouns (Komachi et al., 2007; Taira et al., 2008). To identify the semantic units of a sentence and to correctly understand syntactic relations, it is not sufficient to target only PASA.

Thus, we propose a multi-task learning model that effectively leverages ENASA and improves PASA. Our proposed model is based on an end-to-end multilayer bi-directional recurrent neural network (RNN) used in recent works, and the model has networks that distinguish task-independent information and task-specific information.

In summary, the main contributions of this work are the following:

1. This is the first attempt to design a multi-task learning framework for PASA and ENASA, and we show that our models improve the performance of both tasks.
2. Although our model is a simple model that does not consider the interactions between multiple predicates, it achieves a state-of-the-art result on the NAIST Text Corpus (NTC) in PASA by combining syntactic information as one of the features.
3. For ENASA, this is the first work to employ neural networks to effectively incorporate PASA.

2 Related Work

2.1 Japanese PASA and ENASA Approaches

Many machine learning-based methods have been studied in Japanese PASA. Traditional models take pointwise approaches that construct independent models for each core case role (NOM, ACC, DAT). Taira et al. (2008) proposed a supervised model that learns features of each case using decision lists and support vector machines. Imamura et al. (2009) proposed a model that combines a maximum entropy model with a language model trained from large-scale newspaper articles. Hayashibe et al. (2011) designed three models exploiting argument position and type and determined the maximum likelihood output using pairwise comparison.

However, the joint approach that optimizes the scores of all predicate-argument pairs in a sentence simultaneously showed better results than the pointwise approach. Yoshikawa et al. (2011) proposed a model that considers dependency between multiple predicate-argument relations using Markov logic networks. Ouchi et al. (2015) jointly optimized the combinations among multiple predicates and arguments in a sentence using a bipartite graph.

Except for (Taira et al., 2008), these studies focused on the analysis of predicates while there are few studies that focus on event-nouns. Komachi et al. (2007) decomposed ENASA into two tasks: *event-hood* determination and argument identification; they proposed a supervised method using lexico-syntactic patterns. Event-hood determination is the most important characteristic that semantically differentiates ENASA from PASA. It is a task to determine whether a noun refers to an event (e.g., *houkoku* can refer to either “to report” or the outcome of reporting action, “a report”). Since the previous ENASA models adopted the pointwise approach with a single model, they did not explore the effective features in each task. In contrast, our models simultaneously optimize three core case roles. Moreover, the proposed models allow us to distinguish between task-shared and task-specific features using multi-task learning.

2.2 PASA using neural networks

Some neural models have achieved higher performance than traditional machine learning models in Japanese PASA. Shibata et al. (2016) replaced Ouchi et al. (2015)’s scoring function with feed forward neural networks. Matsubayashi and Inui (2017) represented a dependency path between a predicate and its argument with path embeddings and showed that even the local model without multiple predicates can outperform a global model.

Moreover, some end-to-end models have been proposed in Japanese PASA. Ouchi et al. (2017) proposed an end-to-end model based on the model using eight-layer bi-directional long short-term memory (LSTM) proposed by Zhou and Xu (2015) and considered the interaction of multiple predicates simultaneously using a Grid RNN. Matsubayashi and Inui (2018) combined self-attention with Ouchi et al. (2017)’s model to directly capture interaction among multiple predicate-arguments. In particular, the model improved the performance of arguments that have no syntactic dependency with predicates and achieved a state-of-the-art result on Japanese PASA.

2.3 Semantic Role Labeling

Semantic role labeling (SRL) is a similar task to Japanese PASA. Recently, several end-to-end models using neural networks showed high performance in English SRL (Zhou and Xu, 2015; He et al., 2017; Tan et al., 2018). Strubell et al. (2018) proposed a multi-task learning model that jointly learned dependency parsing, part-of-speech tagging, predicate detection, and SRL based on multi-head self-attention. Ouchi et al. (2018) proposed a span-based SRL model using bi-directional LSTMs and achieved state-of-the-art results. The authors scored all possible spans for each label and selected correct spans satisfying constraints when decoding. In terms of the event-noun research, Gerber and Chai (2010) used pointwise mutual information (PMI) as a feature for 10 event-nouns with high frequency and identified semantic roles using a logistic regression model.

There were several LSTM models that also achieved high accuracy gains in Chinese SRL (Wang et al., 2015; Roth and Lapata, 2016; Sha et al., 2016; Marcheggiani et al., 2017; Qian et al., 2017). For event-nouns, Li et al. (2009) showed that combining effective features in verbal SRL with nominal SRL can improve results. Although the authors did not demonstrate that verbal SRL also improves performance in combination with nominal SRL, we show that our model improves performance in both PASA and ENASA.

3 Japanese PASA and ENASA

3.1 Task Description

Japanese predicate (event-noun) argument structure analysis is a task to extract arguments for certain predicates (event-nouns) and assign three case labels, NOM, ACC and DAT (Iida et al., 2007). Arguments are divided into four categories (Taira et al., 2008) according to the positions with their predicates (event-nouns).

Dep Arguments depend on their predicate (event-noun), or a predicate (event-noun) depends on its arguments.

Zero Arguments and their predicate (event-noun) are in the same sentence, but the arguments are omitted by zero anaphora. Therefore, they have no direct dependency.

Inter-zero Zero anaphoric arguments and their predicate (event-noun) are not in the same sentence.

Bunsetsu Arguments and their event-noun are in the same bunsetsu.

A sentence $w = w_1, w_2, \cdots, w_T$ and a predicate (event-noun) $p = p_1, p_2, \cdots, p_q$ are given as input. Iida et al. (2006), Imamura et al. (2009), and Sasano and Kurohashi (2011) also analyze Inter-zero, which is a difficult task because the whole document must be searched. Following existing research (Ouchi et al., 2015, 2017; Matsubayashi and Inui, 2017, 2018; Taira et al., 2008), we only focus on three categories where arguments and their predicate (event-noun) are in the same sentence. In addition, we exclude the Bunsetsu category from the PASA evaluation following Ouchi et al. (2017) and Matsubayashi and Inui (2018).

3.2 End-to-end Single Model

Our single model is based on an end-to-end approach (Zhou and Xu, 2015; Ouchi et al., 2017; Matsubayashi and Inui, 2018). Additionally, we add new features. Figure 2 shows the network architecture of our base model.

3.2.1 Input Layer

Each word $w_t \in [w_1, \cdots, w_T]$ is converted to a feature representation $\mathbf{x}_t \in [\mathbf{x}_1, \cdots, \mathbf{x}_T]$ at the input layer. We use six types of features. The feature representation $\mathbf{x}_t$ is defined as follows:

$$\mathbf{x}_t = \mathbf{x}_t^{\mathrm{as}} \oplus \mathbf{x}_t^{\mathrm{posi}} \oplus \mathbf{x}_t^{\mathrm{dep}} \oplus \mathbf{x}_t^{\mathrm{type}} \oplus \mathbf{x}_p^{\mathrm{task}} \quad (1)$$

where ($\oplus$) indicates concatenation of vectors.

Argument Structure Predicate (event-noun) w_p and argument candidates w_t are converted to the vectors $\mathbf{x}_t^{\mathrm{as}} \in \mathbb{R}^{2d_{\mathrm{w}}}$ by the word embedding matrix.

Position This is a feature that represents the positional relation between w_p and w_t. The feature is calculated by subtracting the word index of argument candidates from the word index of predicates (event-nouns). We use two types of units to represent relative position: word unit p_t^{word} and bunsetsu unit p_t^{bunsetsu}, which are converted to the word positional vector $\mathbf{p}_t^{\mathrm{word}} \in \mathbb{R}^{d_{\mathrm{p}}}$ and the bunsetsu positional vector $\mathbf{p}_t^{\mathrm{bunsetsu}} \in \mathbb{R}^{d_{\mathrm{p}}}$, respectively, by the word and bunsetsu positional embed-

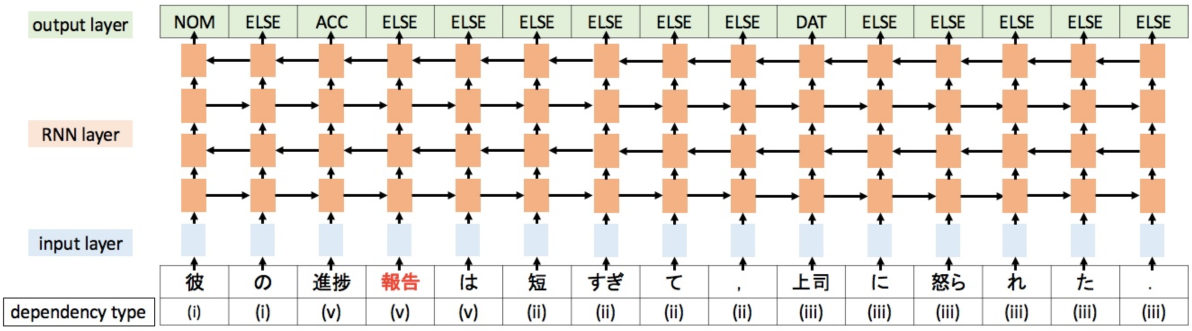


Figure 2: End-to-end single model.

ding matrices. We concatenate these two vectors and obtain the positional vectors $\mathbf{x}_t^{\text{posi}} \in \mathbb{R}^{2d_\text{p}}$.

Dependency This is a feature that represents the dependency relation between w_p and w_t. We set five types of dependency relations:

i). Argument candidates depend on the predicate (event-noun).

ii). The predicate (event-noun) depends on the argument candidates.

iii). No dependency relations between the predicate (event-noun) and argument candidates.

iv). The predicate and candidate arguments are in the same bunsetsu.

v). The event-noun and candidate arguments are in the same bunsetsu.

The dependency relation d_t is converted to the dependency vector $\mathbf{x}_t^{\text{dep}} \in \mathbb{R}^{d_\text{d}}$ by the dependency relation embedding matrix. The dependency type in Figure 2 shows how to make dependency features in Figure 1b as an example. We define the dependency type from the syntactic information annotated in the NTC.

In previous work, dependency features are used differently from our study. Imamura et al. (2009) used a binary feature that represents whether or not there is a dependency relation between the predicate and its arguments. We employ more fine-grained relation types to adapt to event-nouns. Matsubayashi and Inui (2017) represented the interactions between a predicate and its arguments using path embedding. In contrast, we define different types for a predicate and event-noun to distinguish event-nouns from predicates and learn embeddings to find the associated latent structures.

Event-hood Type This is a binary feature to flag all predicates (event-nouns) in a sentence inspired by Matsubayashi and Inui (2018). The purpose of this feature is to prevent predicates from becoming arguments and to help some event-nouns become arguments. The event-hood type vector $\mathbf{x}_t^{\text{type}} \in \mathbb{R}^2$ of a candidate indicates [0,1] if the candidate is a predicate, [1,0] if the candidate is an event-noun, and [0,0] otherwise. The predicate and event-noun are annotated in the NTC.

Task Label This is a binary feature vector $\mathbf{x}_p^{\text{task}} \in \mathbb{R}^1$ that indicates 1 if the task is predicate argument structure analysis; otherwise, 0.

3.2.2 RNN Layer

We use the gated recurrent unit (GRU) (Cho et al., 2014) for RNN. The RNN layers are made up of L layers of stacked bi-directional GRU. Additionally, we apply the residual connections (He et al., 2016) following Ouchi et al. (2017); Matsubayashi and Inui (2018). At each time step t, the hidden state $\mathbf{h}_t^l \in \mathbb{R}^{d_\text{h}}$ in the $l \in [1, \cdots, L]$-th layer is calculated as follows:

$$\mathbf{h}_t^l = \begin{cases} g^l(\mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^l) & (l = \text{odd}) \\ g^l(\mathbf{h}_t^{l-1}, \mathbf{h}_{t+1}^l) & (l = \text{even}) \end{cases} \quad (2)$$

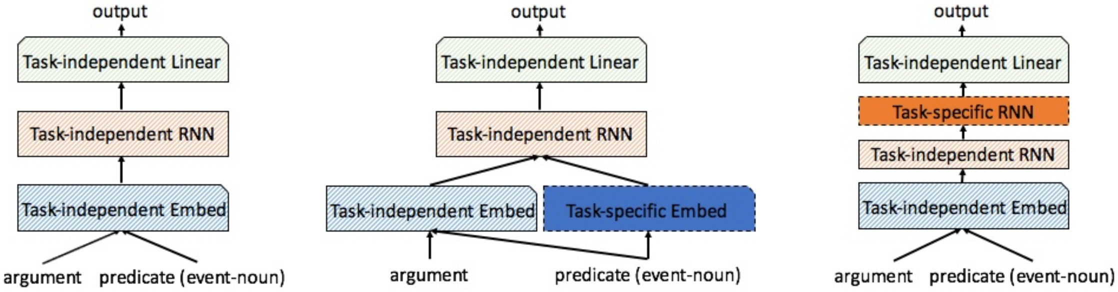
where $g^l(\cdot)$ denotes the l-th layer GRU function. In addition, $\mathbf{h}_t^0 = \mathbf{x}_t$.

3.2.3 Output Layer

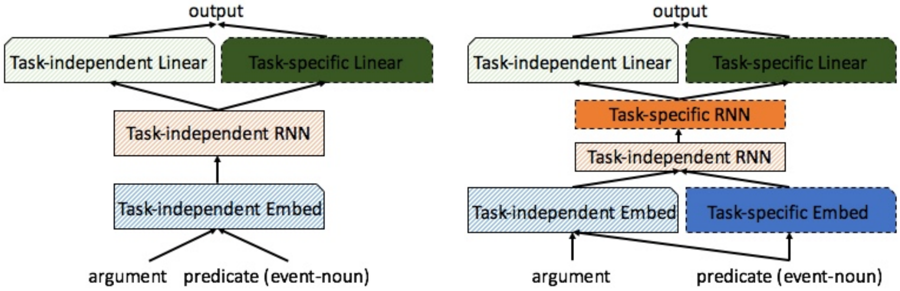
In the output layer, we input each hidden state $\mathbf{h}_t^L$. Then, we obtain the output vector $\mathbf{o}_t$ using the softmax function:

$$\mathbf{o}_t = \text{softmax}(\mathbf{W}_\text{o}\mathbf{h}_t^L + \mathbf{b}_\text{o}) \quad (3)$$

where $\mathbf{W}_\text{o} \in \mathbb{R}^{4 \times d_\text{h}}$ is the parameter matrix, and $\mathbf{b}_\text{o} \in \mathbb{R}^4$ is the bias term. The output vector represents the probability for each argument candidate



(a) Single (b) Multi-input (c) Multi-RNN



(d) Multi-output (e) Multi-ALL

Figure 3: Proposed models: (a) Single, (b) Multi-input, (c) Multi-RNN, (d) Multi-output, (e) Multi-ALL.

over four labels, [NOM, ACC, DAT, ELSE]. ELSE denotes that the candidate argument does not have a case label. In testing, the maximum probability label is selected as the output label. We train the model using the cross-entropy loss function.

4 Multi-task Model

Multi-task learning has been successfully applied to various natural language processing tasks (Collobert et al., 2011; Søgaard and Goldberg, 2016; Luong et al., 2016; Hashimoto et al., 2017; Liu et al., 2017; Stoyanov et al., 2018; Marasovic and Frank, 2018; Strubell et al., 2018). One of the advantages of multi-task learning is that it learns better representation, which is robust against task-dependent noise by increasing training data. In this paper, we introduce multi-task learning to PASA and ENASA for the first time. We propose three methods to extend the end-to-end single model to the multi-task learning model in the input layer, RNN layer, and output layer. Figure 3 shows the proposed models. Our final model combines all three methods (Figure 3e).

4.1 Multi Input Layer

Even if the surface form is the same, the contexts are different for predicates and event-nouns. For example, the event-noun *houkoku* "report" in Figure 1b has an argument in the same bunsetsu unlike predicates. Moreover, the event-noun also has a nominative argument role for the predicate *mijikai* "short". Therefore, given this, we prepare a task-specific word embedding matrix that addresses the task-specific distribution of words. The predicate is converted to PASA-specific vectors $\mathbf{x}_t^{\mathrm{p}} \in \mathbb{R}^{d'_{\mathrm{w}}}$ by the PASA-specific predicate embedding matrix. Similarly, the event-noun is converted to ENASA-specific vectors $\mathbf{x}_t^{\mathrm{n}} \in \mathbb{R}^{d'_{\mathrm{w}}}$ by the ENASA-specific event-noun embedding matrix. These matrices are randomly initialized and can be learned during training.

The feature vector $\overline{\mathbf{x}}_t$ is defined as follows:

$$\overline{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t \oplus \mathbf{x}_t^{\mathrm{p}} & (\mathrm{PASA}) \\ \mathbf{x}_t \oplus \mathbf{x}_t^{\mathrm{n}} & (\mathrm{ENASA}) \end{cases} \tag{4}$$

4.2 Multi RNN Layer

Previous work (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) proposed hierarchical multi-task learning models that exploited features obtained from easy tasks for difficult tasks. These studies showed that performance improves when low-layer RNN representations are trained in easy tasks and high-layer RNN are leveraged for difficult tasks. Therefore, we construct a network that hierarchically overlaps a task-specific RNN on a task-independent RNN. Lower RNN layers learn task-independent knowledge representations. Then, the task-specific RNN adjusts the representations for each task. At each time step t, the hidden state $\mathbf{m}_t^{l'} \in \mathbb{R}^{d'_\mathrm{h}}$ in the $l' \in [1, \cdots, L']$-th layer is calculated as follows:

$$\mathbf{m}_t^{l'} = \begin{cases} g^{l'}(\mathbf{m}_t^{l'-1}, \mathbf{m}_{t-1}^{l'}) & (l' = \mathrm{odd}) \\ g^{l'}(\mathbf{m}_t^{l'-1}, \mathbf{m}_{t+1}^{l'}) & (l' = \mathrm{even}) \end{cases} \quad (5)$$

$$g^{l'}(\cdot) = \begin{cases} g_\mathrm{p}^{l'}(\cdot) & (\mathrm{PASA}) \\ g_\mathrm{n}^{l'}(\cdot) & (\mathrm{ENASA}) \end{cases} \quad (6)$$

where $g^{l'}(\cdot)$, $g_\mathrm{p}^{l'}(\cdot)$, and $g_\mathrm{n}^{l'}(\cdot)$ denote the l'-th layer GRU functions. In addition, $\mathbf{m}_t^0 = \mathbf{h}_t^L$.

4.3 Multi Output Layer

The position of arguments is different with respect to predicates and event-nouns. For example, predicates seldom have arguments in the same bunsetsu. In contrast, event-nouns often have arguments in the same bunsetsu, compound nouns, for example. Therefore, it is intuitive and natural to divide the output layer into task-independent and task-specific layers. The task-specific output vectors are calculated as follows:

$$\mathbf{o}_t^\mathrm{p} = \mathbf{W}_\mathrm{o}^\mathrm{p}\mathbf{h}_t + \mathbf{b}_\mathrm{o}^\mathrm{p} \quad (7)$$

$$\mathbf{o}_t^\mathrm{n} = \mathbf{W}_\mathrm{o}^\mathrm{n}\mathbf{h}_t + \mathbf{b}_\mathrm{o}^\mathrm{n} \quad (8)$$

$$\mathbf{g}_t = \sigma(\mathbf{W}_\mathrm{g}\mathbf{h}_t + \mathbf{b}_\mathrm{g}) \quad (9)$$

where $\mathbf{W}_\mathrm{o}^\mathrm{p}, \mathbf{W}_\mathrm{o}^\mathrm{n}, \mathbf{W}_\mathrm{g} \in \mathbb{R}^{4 \times d_\mathrm{h}}$ are the parameter matrices, and $\mathbf{b}_\mathrm{o}^\mathrm{p}, \mathbf{b}_\mathrm{o}^\mathrm{n}, \mathbf{b}_\mathrm{g} \in \mathbb{R}^4$ are the bias terms. $\mathbf{h}_t$ is the hidden state of the last layer. We combine task-specific output vectors $\mathbf{o}_t^\mathrm{p}, \mathbf{o}_t^\mathrm{n}$ with task-independent output vector $\mathbf{o}_t$ by the gate $\mathbf{g}_t$.

$$\mathbf{c}_t = \begin{cases} \mathbf{g}_t \odot \mathbf{o}_t + (1 - \mathbf{g}_t) \odot \mathbf{o}_t^\mathrm{p} & (\mathrm{PASA}) \\ \mathbf{g}_t \odot \mathbf{o}_t + (1 - \mathbf{g}_t) \odot \mathbf{o}_t^\mathrm{n} & (\mathrm{ENASA}) \end{cases} \quad (10)$$

$$\overline{\mathbf{o}}_t = \mathrm{softmax}(\mathbf{c}_t) \quad (11)$$

where $(\odot)$ denotes the element-wise product. The output vector $\overline{\mathbf{o}}_t$ represents the probability of [NOM, ACC, DAT, ELSE].

5 Experiments

5.1 Dataset and Setting

We use NTC 1.5 for our experiments. We divide the dataset into training, development, and test sets in the same way as Taira et al. (2008). We use morphological and syntactic information, such as the word boundaries, the bunsetsu boundaries and the dependency relations provided in the NTC.

For the development and test sets, if there are two or more arguments annotated with the same case label in a sentence, we set an argument that only has a dependency relation with a predicate as a correct answer and assign the ELSE label to other arguments. If there is no dependency relation, we set an argument with the shortest distance $|w_p - w_t|$ as a correct answer. If the distance is equal, an argument on the left side of a predicate is considered a correct answer.

In NTC 1.5, if there is a predicate phrase, such as "verbal noun + *suru*," *suru* is annotated as a predicate word. We consider the verbal noun as the predicate word at the preprocessing step to match the surface of a predicate with that of an event-noun. Take the predicate *houkoku-suru* "to report" and an event-noun *houkoku* "report" as an example. Although w_p before preprocessing are *suru* and *houkoku*, w_p are unified to *houkoku* after preprocessing.

the dimension of word embeddings d_w	300
the dimension of position embeddings d_p	16
the dimension of dependency embeddings d_d	16
the dimension of hidden states d_h	300
the number of GRU layers L	4
the dimension of task-specific word embeddings d'_w	16
the dimension of task-specific hidden states d'_h	300
the number of task-specific GRU layers L'	2
dropout rate	0.4
batch size	8
gradient clipping	4

Table 1: Hyperparameters.

5.2 Hyperparameters

We use pre-trained embeddings[2] for the initial values of the word embedding matrix. The initial values of the other embedding matrices are sampled according to a uniform distribution of [-0.25,0.25]. We convert words appearing more than once in the training set into word vectors and the remaining words into the unknown word vector. We adopt AdaDelta ($\epsilon = 10^{-6}, \ \rho = 0.95$) as the optimization method. We set the number of epochs to 20 and evaluate the model with the highest F1 scores on the development set. Table 1 shows the hyperparameters.

5.3 Results

We evaluate each model with the NTC 1.5 test. The experimental results for the argument structure analysis of predicates and event-nouns are shown in Tables 2 and 3.

---

[2] http://www.asahi.com/shimbun/medialab/word_embedding

			Dep				*Zero*			
Method	ALL	SD	ALL	NOM	ACC	DAT	ALL	NOM	ACC	DAT
Ouchi+ 17	81.42		88.17	88.75	93.68	64.38	47.12	50.65	32.35	7.52
M&I 17	83.50	±0.17	89.89	91.19	95.18	61.90	51.79	54.69	41.8	17
M&I 18	83.94	±0.12	90.26	90.88	94.99	67.57	**55.55**	**57.99**	**48.9**	**23**
Single	83.62	±0.17	90.09	90.45	94.84	69.77	51.87	54.73	43.48	11.40
Multi-input	83.88	±0.11	90.27	90.65	95.12	69.86	53.01	55.82	44.68	10.77
Multi-RNN	83.91	±0.23	90.17	90.58	95.07	67.94	53.31	55.85	45.71	9.97
Multi-output	83.77	±0.20	90.13	90.68	94.89	68.16	53.93	56.73	43.79	9.45
Multi-ALL	83.82	±0.10	90.15	90.68	95.06	67.56	53.50	56.37	45.36	8.70
Multi-RNN+`DEP`	84.55	±0.11	90.69	91.28	95.25	70.07	51.56	54.29	42.67	1.85
Multi-output+`DEP`	84.73	±0.11	90.82	**91.46**	95.29	70.69	52.29	55.14	42.15	1.81
Multi-ALL+`DEP`	**84.75**	±0.16	**90.88**	91.40	**95.37**	**71.02**	52.35	55.10	42.54	2.32
M&I 17 (ens. of 5)	84.07		90.24	91.59	95.29	62.61	53.66	56.47	44.7	16
M&I 18 (ens. of 10)	85.34		91.26	91.84	95.57	70.8	**58.07**	**60.21**	**52.5**	**26**
Multi-RNN+`DEP` (ens. of 5)	85.85		91.61	92.11	**95.87**	72.63	53.41	55.96	46.10	0
Multi-output+`DEP` (ens. of 5)	85.83		91.52	92.12	95.69	72.72	54.35	57.02	45.95	0
Multi-ALL+`DEP` (ens. of 5)	**86.01**		**91.63**	**92.15**	95.80	**72.95**	54.99	57.84	45.20	0

Table 2: F1 scores on the PASA test set. Single is a base model without multi-task learning.

			Dep				*Zero*				*Bunsetsu*			
Method	ALL	SD	ALL	NOM	ACC	DAT	ALL	NOM	ACC	DAT	ALL	NOM	ACC	DAT
Taira+ 08 on NTC 1.4				68.01	62.46	56.05		36.19	20.46	6.62		78.93	77.96	58.13
Single	66.21	±0.15	74.64	76.06	74.54	51.28	46.05	49.67	33.36	13.63	78.24	76.67	81.75	48.55
Multi-input	67.89	±0.42	75.62	76.63	75.78	**57.17**	49.07	52.81	36.95	19.39	**79.35**	77.31	**83.31**	51.03
Multi-RNN	67.96	±0.44	75.86	76.90	76.33	54.46	48.67	52.18	**38.47**	18.89	79.08	77.24	82.89	50.93
Multi-output	67.96	±0.17	76.25	**77.18**	**76.90**	54.97	48.74	52.48	36.09	**19.64**	79.02	77.00	83.04	50.60
Multi-ALL	**68.00**	±0.41	75.90	77.16	76.05	53.00	**49.66**	**53.37**	37.64	14.46	79.05	77.32	82.61	**51.83**
Multi-ALL+`DEP`	67.68	±0.39	**75.95**	**77.18**	76.11	55.26	47.57	51.21	35.14	15.65	79.06	**77.44**	82.66	51.10
Multi-ALL (ens. of 5)	**71.14**		**78.63**	**79.66**	**78.83**	58.29	**52.49**	**56.41**	**39.02**	16.42	**81.90**	**80.25**	**85.21**	**56.29**
Multi-ALL+`DEP` (ens. of 5)	69.90		77.86	78.89	78.16	**58.46**	49.36	53.10	36.36	17.23	81.16	79.74	84.57	52.99

Table 3: F1 scores on the ENASA test set.

Predicate Argument Structure Analysis The first set of rows in Table 2 shows the results of previous models. Ouchi+ 17 is the model from the Multi-Seq model in (Ouchi et al., 2017). M&I 17 is the model in (Matsubayashi and Inui, 2017). M&I 18 is the model from the MP-POOL-SELFATT model in (Matsubayashi and Inui, 2018).

The second set of rows in Table 2 shows the results of the proposed models. These models do not use the dependency feature. Compared with the single model, all multi-task learning models improved the overall F1 scores. Among them, Multi-RNN improved the overall F1 score from the single model by 0.29 points. In previous work, Ouchi et al. (2017); Matsubayashi and Inui (2018) see improvements of 0.27 and 0.55 F1 points in their baseline models by considering multiple predicate-argument interactions. Therefore, we show that multi-task learning with ENASA achieved comparable effects as these studies in PASA.

The third set of rows shows the results of proposed models using all features including the dependency feature. Multi-ALL+`DEP` achieved the best F1 score among all the models including previous state-of-the-art models. In particular, the dependency feature was effective for Dep arguments. On the other hand, the performance for Zero arguments was poor. This result suggests that the dependency feature causes the model to optimize mainly for Dep arguments since Dep arguments are more numerous than Zero arguments.

The fourth set of rows shows the results of ensemble models. Overall, our proposed model outperformed the previous ensemble model by 0.67 points in the overall F1 score. Moreover, our models are simple models that independently analyze each predicate in a sentence. Nevertheless, our models achieved higher results than Ouchi et al. (2017); Matsubayashi and Inui (2018). Although recent works have researched the method whereby multiple predicate-argument interactions are considered simultaneously, how to use syntactic in-

社会党 の 山花 貞夫 ・ 新 民主 連合 会長 [NOM] は 十 七 日 午前 、 新た に 結成 [PRED] する
Social Party of Yamahana Sadao ・ new democratic federation president NOM 1 7 day a.m. , newly organize
新 会派 [ACC] に 参加 する 同 党 所属 国会 議員 二 十 四 人 の 「 日本 社会党 ・ 護憲
new faction ACC participate the said party belonging Diet member 2 1 4 people of “ Japan Social Party ・ constitution protection
民主 連合 」 から の 離党 届 を 森井 忠良 国対 委員 長 に 提出 し た 。
democratic federation ” from secession notification ACC Morii Tadayoshi National Diet committee chair DAT submitted .
New Democratic Federation President Yamada Sadao, a member of Social Democratic Party of Japan, submitted to Mori Tadayoshi, National Diet Committee Chairperson, a notice of resignation from a party “Japan Social Party-Pro-constitution Democratic Federation” from 24 Diet members of SDP who will join a newly organized new faction in the morning of the 17th.

(a) predicate: 結成 “organize,” NOM: 会長 “president,” ACC: 会派 “faction.”

問題 の 行方 [ACC] を 左右 [PRED] する カギ [NOM] は 「 電子 レビュー 」 の よう な もの かも しれ ない 。
Problem of whereabouts ACC determine key NOM “ electronic review ” like may .
The key to determine whereabouts of the problem may be like an “electronic review.”

(b) predicate: 左右 “determine,” NOM: カギ “key,” ACC: 行方 “whereabouts.”

事態 [ACC] 打開 [EN] の カギ [NOM] を 握る の は 、 セルビア 人 勢力 に 影響 力 を 持つ ロシア 及び 新 ユーゴスラビア だろ う 。
situation break of key ACC hold NOM , Serbs peopel Influence DAT affect power ACC have Russia and new Yugoslavia will .
Russia and New Yugoslavia, which are affecting Serb influence, will hold the key to break the situation.

(c) event-noun: 打開 “break,” NOM: カギ “key,” ACC: 事態 “situation.”

行政 、 銀行 、 農協 系統 の 経営 トップ [NOM] が 責任 [ACC] 回避 [PRED] する の は なぜ か 。
administration , bank , agricultural coorperative system of management top NOM responsibility avoid NOM why .
Why do top managements of administration, bank and agricultural cooperative avoid responsibility?

(d) predicate: 回避 “avoid,” NOM: トップ “top,” ACC: 責任 “responsibility.”

私 は 、 紙幣 を 改めて 見る と 、 なるほど 千 円 札 の 左下 [DAT] に 直径 五 ミリ ほど の 丸 [ACC] が 一つ 押さ [PRED] れ て ある 。
I NOM , bill ACC again look when , I see 1000 yen bill of lower left DAT diameter 5 mm about of circle NOM one push PASSIVE
When I look at the bill again, I see, a circle about 5 mm in diameter is pushed at the lower left of 1000 yen bill.

(e) predicate: 押す “push,” ACC: 丸 “circle,” DAT: 左下 “lower left.”

ロージズ 博士 は 、 アポ E [NOM] が 発病 [DAT] に 果たす [PRED] 役割 [ACC] の 解明 [ACC/FALSE] に 取り組ん で おり 、
PERSON doctor NOM , Apolipoprotein E NOM pathogenesis DAT play role of solution DAT tackle ,
「 発病 を 予防 する 薬 の 開発 を めざし て いる 」 と いう 。
“ onset ACC prevent drug of develop ACC aim ” .
Dr. PERSON is working on elucidating the role Apolipoprotein E plays in the pathogenesis, and says “We aim to develop drugs to prevent onset”.

(f) predicate: 果たす “play,” NOM: E, ACC: 役割 “role,” DAT: 発病 “pathogenesis.”

Figure 4: Examples of analysis errors on the PASA test set

formation in the end-to-end model is a subject for future work.

Event-noun Argument Structure Analysis The first set of rows in Table 3 shows the results of a previous model in event-noun argument structure analysis. Taira+ 08 is the model from (Taira et al., 2008). Since its scores are from NTC 1.4, the model cannot be directly compared to our models. Compared with the single model, all multi-task models improved the overall F1 scores. However, Multi-ALL+DEP compared unfavorably with Multi-ALL even though it was the best PASA architecture. Therefore, this implies that the dependency type feature between the predicate and its argument is not effective in ENASA.

5.4 Analysis

In Figure 4, we compare the PASA results from test sets for each model. In Examples (a), (b) and (d), the single model failed to predict correct arguments but the Multi-RNN model correctly predicted arguments. In Example (a), the single model incorrectly predicted that arguments do not exist in this sentence. Comparing the training set of each task, although the number of event-nouns is approximately one-third of the number of predicates, the number of *kessei* 結成 “organize (event-nouns)” is approximately twice the number of *kessei* 結成 “organize (predicates).” Accordingly, we showed that the Multi-RNN model effectively leverages the information of event-nouns using multi-task learning.

In Example (b), the single model incorrectly predicted that the NOM argument does not exist, but the multi-RNN predicted the correct arguments. Comparing the training set, there is *sayuu* 左右 "determine (predicate)" but not *sayuu* 左右 "determine (event-noun)." However, there are some *kagi* カギ "key (arguments of predicates)" in the PASA training set, and there is one *kagi* カギ "key (argument of event-noun)" in the ENASA training set (Example (c)). Moreover, in Example (c), *dakai* 打開 "break (event-noun)" depends on *kagi* カギ "key" like *sayuu* 左右 "determine (predicate)" in Example (b); however, no predicate depends on *kagi* カギ "key" in the training set. Accordingly, the Multi-RNN model also leverages the arguments of event-nouns and the positional relations between event-nouns and their arguments.

Example (d) is an interesting case in which a predicate *kaihi* 回避 "avoid" and its argument *sekinin* 責任 "responsibility" are located in the same bunsetsu. Although this argument type (Bunsetsu) is excluded from the evaluation target in PASA, it is common as a compound noun in ENASA. Therefore, the single model wrongly predicted that the ACC argument does not exist, but multi-RNN was able to predict the answer using the specific knowledge of event-nouns.

In contrast, in Example (e), the single model correctly predicted the answer, but the multi-RNN model failed to predict the correct arguments. Multi-RNN incorrectly predicted that the DAT argument does not exist in this sentence. However, *ni* に, a postpositional particle located after an argument, often indicates a dative case. Nevertheless, multi-RNN often predicted a wrong DAT argument by ignoring *ni* に. Therefore, for DAT analysis, the information of event-nouns adversely affects PASA.

In Example (f), the Multi-ALL+`DEP` model correctly predicted the answer, but the Multi-ALL model failed. Specifically, Multi-ALL+`DEP` correctly predicted that the ACC argument is *yakuwari* 役割 "role," which is dependent on *hatasu* 果たす "play." However, the Multi-ALL incorrectly predicted that the ACC argument is *kaimei* 解明 "solution." Similarly, Multi-ALL without syntactic information made many mistakes, including attributive modification, such as Figure 1c. Table 4 shows the results of the two PASA models for attributive modification instances. Multi-ALL+`DEP` considerably outperformed Multi-ALL for all cases using dependency features. Therefore, these results suggest that the dependency type feature is effective for PASA with respect to attributive modifications.

	ALL	NOM	ACC	DAT
Multi-ALL	80.31	83.37	72.16	19.48
Multi-ALL+`DEP`	81.83	84.67	74.41	28.31

Table 4: F1 scores on the PASA test set with respect to attributive modifications.

6 Conclusion

We design a multi-task learning model for predicate and event-noun argument structure analysis. The experiment results show that the multi-task models outperform the single-task model on the NAIST Test Corpus for both tasks. Moreover, our model achieves a state-of-the-art result for PASA. In addition, this is the first work to employ neural networks for ENASA. In future work, we plan to consider multiple predicates and event-nouns.

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of ACL*, pages 1583–1592.

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of EMNLP*, pages 1923–1933.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of IJCNLP*, pages 201–209.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of ACL*, pages 473–483.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of COLING/ACL*, pages 625–632.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop*, pages 132–139.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of ACL-IJCNLP*, pages 85–88.

Mamoru Komachi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Learning based argument structure analysis of event-nouns in Japanese. In *Proceedings of PACLING*, pages 120–128.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving nominal SRL in Chinese language with verbal SRL information and automatic predicate recognition. In *Proceedings of EMNLP*, pages 1280–1288.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of ACL*, pages 1–10.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of ICLR*.

Ana Marasovic and Anette Frank. 2018. SRL4ORL: Improving opinion role labeling using multi-task learning with semantic role labeling. In *Proceedings of NAACL*, pages 583–594.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*, pages 411–420.

Yuichiroh Matsubayashi and Kentaro Inui. 2017. Revisiting the design issues of local models for Japanese predicate-argument structure analysis. In *Proceedings of IJCNLP*, pages 128–133.

Yuichiroh Matsubayashi and Kentaro Inui. 2018. Distance-free modeling of multi-predicate interactions in end-to-end Japanese predicate argument structure analysis. In *Proceedings of COLING*, pages 94–106.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of ACL-IJCNLP*, pages 961–970.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Neural modeling of multi-predicate interactions for Japanese predicate argument structure analysis. In *Proceedings of ACL*, pages 1591–1600.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

Feng Qian, Lei Sha, Baobao Chang, LuChen Liu, and Ming Zhang. 2017. Syntax aware LSTM model for semantic role labeling. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 27–32.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of IJCNLP*, pages 758–766.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2016. Capturing argument relationship for Chinese semantic role labeling. In *Proceedings of EMNLP*, pages 2011–2016.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for Japanese predicate argument structure analysis. In *Proceedings of ACL*, pages 1235–1244.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*, pages 231–235.

Veselin Stoyanov, Heng Ji, Ying Lin, and Shengqi Yang. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of ACL*, pages 799–809.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, pages 5027–5038.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of EMNLP*, pages 523–532.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proceedings of EMNLP*, pages 1626–1631.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with Markov logic. In *Proceedings of IJCNLP*, pages 1125–1133.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, pages 1127–1137.

Domain adaptation for part-of-speech tagging of noisy user-generated text

Luisa März and **Dietrich Trautmann** and **Benjamin Roth**
CIS, University of Munich (LMU) Munich, Germany
`{luisa.maerz, dietrich, beroth}@cis.lmu.de`

Abstract

The performance of a Part-of-speech (POS) tagger is highly dependent on the domain of the processed text, and for many domains there is no or only very little training data available. This work addresses the problem of POS tagging noisy user-generated text using a neural network. We propose an architecture that trains an out-of-domain model on a large newswire corpus, and transfers those weights by using them as a prior for a model trained on the target domain (a data-set of German Tweets) for which there is very little annotations available. The neural network has two standard bidirectional LSTMs at its core. However, we find it crucial to also encode a set of task-specific features, and to obtain reliable (source-domain and target-domain) word representations. Experiments with different regularization techniques such as early stopping, dropout and fine-tuning the domain adaptation prior weights are conducted. Our best model uses external weights from the out-of-domain model, as well as feature embeddings, pretrained word and sub-word embeddings and achieves a tagging accuracy of slightly over 90%, improving on the previous state of the art for this task.

1 Introduction

Part-of-speech (POS) tagging is a prerequisite for many applications and necessary for a wide range of tools for computational linguists. The state-of-the art method to implement a tagger is to use neural networks (Ma and Hovy, 2016; Yang et al., 2018). The performance of a POS tagger is highly dependent on the domain of the processed text and the availability of sufficient training data (Schnabel and Schütze, 2014). Existing POS taggers for canonical German text already achieve very good results around 97% accuracy, e.g. (Schmid, 1999; Plank et al., 2016). When applying these trained models to out-of-domain data the performance decreases drastically.

One of the domains where there is not enough data is online conversational text in platforms such as Twitter, where the very informal language exhibits many phenomena that differ significantly from canonical written language.

In this work, we propose a neural network that combines a character-based encoder and embeddings of features from previous non-neural approaches (that can be interpreted as an inductive bias to guide the learning task). We further show that the performance of this already effective tagger can be improved significantly by incorporating external weights using a mechanism of domain-specific L2-regularization during the training on in-domain data. This approach establishes state-of-the-art results of 90.3% accuracy on the German Twitter corpus of Rehbein (2013).

2 Related Work

The first POS tagging approach for German Twitter data was conducted by Rehbein (2013) and reaches an accuracy of 88.8% on the test set using a CRF. They use a feature set with eleven different features and an extended version of the STTS (Schiller et al., 1999) as a tagset. Gimpel et al. (2011) developed a tagset for English Twitter data and report results of 89.37% on their test set using a CRF with different features as well. POS tagging for different languages using a neural architecture was successfully applied by Plank et al. (2016). The data comes from the Universal Dependencies project[1] and mainly contains German newspaper texts and Wikipedia articles.

The work of Barone et al. (2017) investigates different regularization mechanisms in the field of domain adaptation. They use the same L2 regularization mechanism for neural machine translation, as we do for POS tagging.

---

[1] `http://universaldependencies.org`

Proceedings of NAACL-HLT 2019, pages 3415–3420
Minneapolis, Minnesota, June 2 - June 7, 2019.

3 Data

3.1 Tagset

The Stuttgart-Tübingen-TagSet (STTS, Schiller et al. (1999)) is widely used as the state-of-the-art tagset for POS tagging of German. Bartz et al. (2013) show that the STTS is not sufficient when working with textual data from online social platforms, as online texts do not have the same characteristics as formal-style texts, nor are identical to spoken language. Online conversational text often contains contracted forms, graphic reproductions of spoken language such as prolongations, interjections and grammatical inconsistencies as well as a high rate of misspellings, omission of words etc.

For POS tagging we use the tagset of Rehbein (2013), where (following Gimpel et al. (2011)) additional tags are provided to capture peculiarities of the Twitter corpus. This tagset provides tags for @-mentions, hashtags and URLs. They also provide a tag for non-verbal comments such as ***Trommelwirbel*** (drum-roll). Additional, complex tags for amalgamated word forms were used (see Gimpel et al. (2011)). Overall the tagset used in our target domain contains 15 tags more than the original STTS.

3.2 Corpora

Two corpora with different domains are used in this work. One of them is the TIGER corpus and the other is a collection of German Twitter data.

The texts in the TIGER corpus (Brants et al., 2004) are taken from the Frankfurter Rundschau newspaper and date from 1995 over a period of two weeks. The annotation of the corpus was created semi automatically. The basis for the annotation of POS tags is the STTS. The TIGER corpus is one of the standard corpora for German in NLP and contains 888.505 tokens.

The Twitter data was collected by Rehbein (2013) within eight months in 2012 and 2013. The complete collection includes 12.782.097 distinct tweets, from which 1.426 tweets were randomly selected for manual annotation with POS tags. The training set is comparably small and holds 420 tweets, whereas the development and test set hold around 500 tweets each (overall 20.877 tokens). Since this is the only available German annotated Twitter corpus, we use it for this work.

3.3 Pretrained word vectors

The usage of pretrained word embeddings can be seen as a standard procedure in NLP to improve the results with neural networks (see Ma and Hovy (2016).

3.4 FastText

FastText[2] provides pretrained sub-word embeddings for 158 different languages and allows to obtain word vectors for out-of-vocabulary words. The pretrained vectors for German are based on Wikipedia articles and data from Common Crawl[3]. We obtain 97.988 different embeddings for the tokens in TIGER and the Twitter corpus of which 75.819 were already contained in Common Crawl and 22.171 were inferred from sub-word units.

3.5 Word2Vec

Spinningbytes[4] is a platform for different applications in NLP and provides several solutions and resources for research. They provide word embeddings for different text types and languages, including Word2Vec (Mikolov et al., 2013) vectors pretrained on 200 million German Tweets. Overall 17.030 word embeddings form the Spinningbytes vectors are used (other words are initialized all-zero).

3.6 Character level encoder

Lample et al. (2016) show that the usage of a character level encoder is expedient when using bidirectional LSTMs. Our implementation of this encoder follows Hiroki Nakayama (2017)[5], where character embeddings are passed to a bidirectional LSTM and the output is concatenated to the word embeddings.

4 Experiments

This section describes the proposed architecture of the neural network and the conditional random field used in the experiments. For comparison of the results we also experiment with jointly training on a merged training set, which contains the Twitter and the TIGER training sets.

---

[2] `https://fasttext.cc`

[3] `https://commoncrawl.org`

[4] `https://www.spinningbytes.com`

[5] `https://github.com/Hironsan/anago`

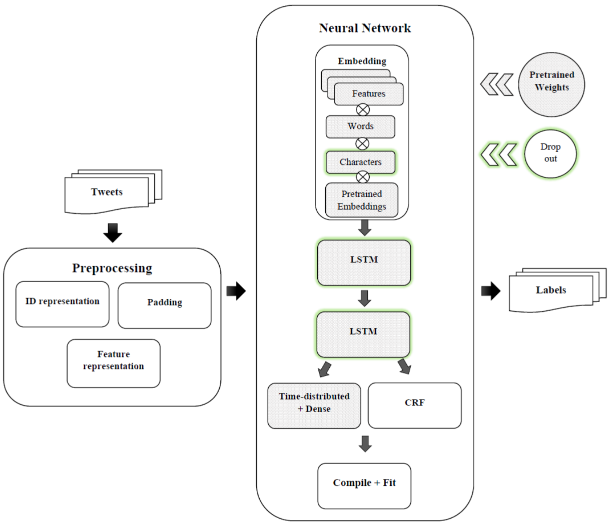


Figure 1: Final architecture of the neural model. Layers that are passed pretrained weights are hatched in gray. Dropout affected layers are highlighted in green.

4.1 Methods

4.1.1 Conditional random field baseline

The baseline CRF of Rehbein (2013) achieves an accuracy of 82.49%. To be comparable with their work we implement a CRF equivalent to their baseline model. Each word in the data is represented by a feature dictionary. We use the same features as Rehbein proposed for the classification of each word. These are the lowercased word form, word length, number of uppercase letters, number of digits and occurrence of a hashtag, URL, @-mention or symbol.

4.1.2 Neural network baseline

The first layer in the model is an embedding layer. The next layers are two bidirectional LSTMs. The baseline model uses softmax for each position in the final layer and is optimized using Adam core with a learning rate of 0.001 and the categorical crossentropy as the loss function.

4.1.3 Extensions of the neural network

The non neural CRF model benefits from different features extracted from the data. Those features are not explicitely modeled in the neural baseline model, and we apply a feature function for the extended neural network. We include the features used in the non-neural CRF for hashtags and @-mentions. In addition, we capture orthographic features, e.g., whether a word starts with a digit or an upper case letter. Typically, manually defined features like these are not used in neural networks, as a neural network should take over feature engineering completely. Since this does not work optimally, especially for smaller data sets, we have decided to give the neural network this type of information as well. Thus we combine the advantages of classical feature engineering and neural networks. This also goes along with the observations of Plank et al. (2018) and Sagot and Martínez Alonso (2017), who both show that adding conventional lexical information improves the performance of a neural POS tagger. All words are represented by their features and for each feature type an embedding layer is set up within the neural network in order to learn vectors for the different feature expressions. Afterwards all the feature embeddings are added together. As the next step we use the character level layer mentioned in section 3.6 (Lample et al., 2016). The following vector sequences are concatenated at each position and form the input to the bidirectional LSTMs:

- Feature embedding vector
- character-level encoder
- FastText vectors
- Word2Vec vectors

4.1.4 Domain Adaptation and regularization

We train the model with the optimal setting on the TIGER corpus, i.e., we prepare the TIGER data just like the Twitter data and extract features, include a character level layer and use pretrained embeddings. We extract the weights $\hat{W}$ that were optimized with TIGER. The prior weights $\hat{W}$ are used during optimization as a regularizer for the weights W used in the final model (trained on the Twitter data). This is achieved by adding the penalty term R_W, as shown in Equation 1, to the objective function (cross-entropy loss).

$$R_W = \lambda||W - \hat{W}||_2^2 \qquad (1)$$

The regularization is applied to the weights of the two LSTMs, the character LSTM, to all of the embedding layers and to the output layer.

As a second regularization mechanism we include dropout for the forward and the backward LSTM layers. We also add 1 to the bias of the forget gate at initialization, since this is recommended in Jozefowicz et al. (2015). Additionally, we use early stopping. Since the usage of different regularization techniques worked well in the experiments of Barone et al. (2017), we also tried the combination of different regularizers in this work. Figure 1 shows the final architecture of our model.

4.2 NCRF++

We also report results obtained by training the sequence labelling tagger of Yang and Zhang (2018), NCRF++. They showed that their architecture produces state-of-the-art models across a wide range of data sets (Yang et al., 2018) so we used this standardized framework to compare it with our model.

5 Results

5.1 Experimental Results

Table 1 shows the results on the Twitter test set. The feature-based baseline CRF outperforms the baseline of the neural net with more than 20 percentage points. After adding the feature information, the performance of the neural baseline is improved by 13 percentage points, which is understandable, because many German POS tags are case sensitive.

experiment	**accuracy**
baseline crf	0.831
baseline neural model	0.634
neural model	
+features	0.768
+character embeddings	0.796
+pretrained word vectors	0.845
+l2 domain adaptation	0.896
+dropout	**0.903**
neural model joint training	0.894
final CRF of Rehbein 2013	0.888
NCRF++ system	0.887

Table 1: Results on the test set using the time-distributed layer.

The model's performance increases by another 3 percentage points if the character level layer is used. Including the pretrained embeddings, FastText and Word2Vec vectors, the accuracy is 84.5%, which outperforms the CRF baseline.

Figure 2 shows the impact of domain adaptation and fine-tuning the prior weight. The value of the λ parameter in the regularization formula 1 can control the degree of impact of the weights on the training. Excluding the pretrained weights means that λ is 0. We observe an optimal benefit from the out-of-domain weights by using a λ value of 0.001. This is in line with the observations of Barone et al. (2017) for transfer-learning for machine translation.

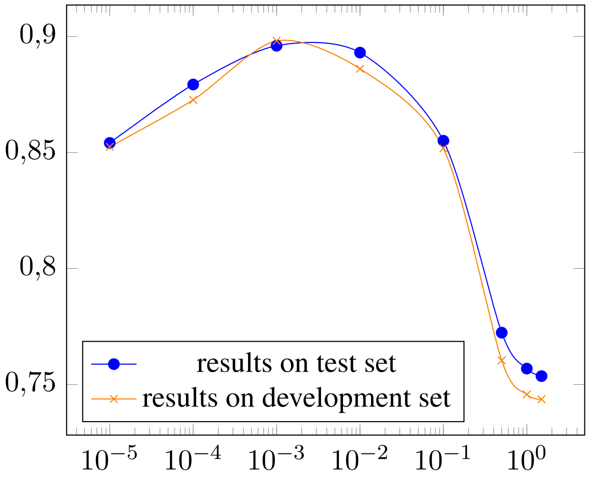


Figure 2: Influence of fine-tuning on the results on dev and test set in accuracy (y-axis). The x-axis corresponds to the different λ values.

Overall the addition of the L2 fine-tuning can improve the tagging outcome by 5 percentage points, compared to not doing domain adaptation. A binomial test shows that this improvement is significant. This result confirms the intuition that the tagger can benefit from the pretrained weights. On top of fine-tuning different dropout rates were added to both directions of the LSTMs for the character level layer and the joint embeddings. A dropout rate of 75% is optimal in our scenario, and it increases the accuracy by 0.7 percentage points.

The final 90.3% on the test set outperform the results of Rehbein (2013) by 1.5 percentage points.Our best score also outperforms the accuracy obtained with the NCRF++ model. This shows that for classifying noisy user-generated text, explicit feature engineering is beneficial, and that the usage of domain adaptation is expedient in this context. Joint training, using all data (out-of-domain and target domain), can obtain an accuracy score of 89.4%, which is about 1 percentage point worse than using the same data with domain adaptation. The training setup for the joint training is the same as for the other experiments and includes all extensions except for the domain adaptation.

5.2 Error Analysis

The most frequent error types in all our systems were nouns, proper nouns, articles, verbs, adjec-

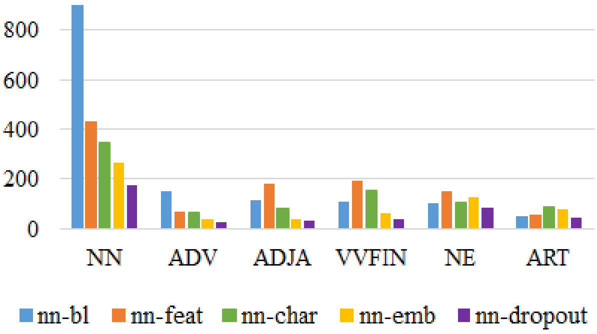


Figure 3: Total number of errors for the six most frequent POS-tags and different experimental settings

tives and adverbs as pictured in figure 3. By including the features the number of errors can be reduced drastically for nouns. Since we included a feature that captures upper and lower case, and nouns as well as proper nouns are written upper case in German, the model can benefit from that information. The pretrained word embeddings also help classifying nouns, articles, verbs, adjectives and adverbs. Only the errors with proper nouns increase slightly. Compared to only including the features, the model can benefit from adding both, the character level layer and the pretrained word vectors, while the results for tagging proper nouns and articles are still slightly worse than the baseline. In contrast the final experimental setup can optimize the results for every POS tag compared to the baseline, see figure 3. Slightly in case of articles and proper nouns, but markedly for the other tags. A comparison of the baseline errors and the errors of the final system shows that Twitter specific errors, e.g. with @-mentions or URLs, can be reduced drastically. Only hashtags still pose a challenge for the tagger. In the gold standard words with hashtags are not always tagged as such, but sometimes are classified as proper nouns. This is due to the fact that the function of the token in the sentence is the one of a proper noun. Thus the tagger has decision problems with these hashtags. Other types of errors, such as confusion of articles or nouns, are not Twitter-specific issues, but are often a problem with POS tagging and can only be fixed by general improvement of the tagger.

6 Conclusion

We present a deep learning based fine-grained POS tagger for German Twitter data using both domain adaptation and regularization techniques. On top of an efficient POS tagger we implemented domain adaptation by using a L2-norm regularization mechanism, which improved the model's performance by 5 percentage points. Since this performance is significant we conclude that fine-tuning and domain adaptation techniques can successfully be used to improve the performance when training on a small target-domain corpus.

Our experiments show that the combination of different regularization techniques is recommendable and can further optimize already efficient systems.

The advantage of our approach is that we do not need a large annotated target-domain corpus, but only pretrained weights. Using a pretrained model as a prior for training on a small amount of data is done within minutes and therefore very practicable in real world scenarios.

References

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494. Association for Computational Linguistics.

Thomas Bartz, Michael Beisswenger, and Angelika Storrer. 2013. Optimierung des stuttgart-tübingen-tagset für die linguistische annotation von korpora zur internetbasierten kommunikation: Phänomene, herausforderungen, erweiterungsvorschläge. *JLCL*, 28:157–198.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International*

Conference on Machine Learning - Volume 37, ICML'15, pages 2342–2350. JMLR.org.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bidirectional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Barbara Plank, Sigrid Klerke, and Zeljko Agic. 2018. The best of both worlds: Lexical resources to improve low-resource part-of-speech tagging. *CoRR*, abs/1811.08757.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *CoRR*, abs/1604.05529.

Ines Rehbein. 2013. Fine-grained pos tagging of german tweets. In *Language Processing and Knowledge in the Web*, pages 162–175, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 25–31. Association for Computational Linguistics.

Anne Schiller, Simone Teufel, Christine Stckert, and Christine Thielen. 1999. Guidelines fr das tagging deutscher textcorpora mit stts (kleines und groes tagset). Seminararbeit, University of Stuttgart, University of Tbingen.

H. Schmid. 1999. *Improvements in Part-of-Speech Tagging with an Application to German*, pages 13–25. Springer Netherlands, Dordrecht.

Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Neural Chinese Address Parsing

Hao Li and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
hao_li@mymail.sutd.edu.sg
luwei@sutd.edu.sg

Pengjun Xie and **Linlin Li**
DAMO Academy
Alibaba Group
chengchen.xpj@alibaba-inc.com
linyan.lll@alibaba-inc.com

Abstract

This paper introduces a new task – Chinese address parsing – the task of mapping Chinese addresses into semantically meaningful chunks. While it is possible to model this problem using a conventional sequence labelling approach, our observation is that there exist complex dependencies between labels that cannot be readily captured by a simple linear-chain structure. We investigate neural structured prediction models with latent variables to capture such rich structural information within Chinese addresses. We create and publicly release a new dataset consisting of 15,000 Chinese addresses, and conduct extensive experiments on the dataset to investigate the model effectiveness and robustness. We release our code and data at http://statnlp.org/research/sp.

Figure 1: Two example Chinese addresses and the expected structures after parsing. Each chunk is underlined with its corresponding label in blue.

1 Introduction

Addresses play an important role in modern society. They are typically used as identifiers to locations and entities in the world that can be used to facilitate various social activities, such as business correspondences, meetings and events. Recent research efforts show that systems that perform *address parsing*, the task of automatically parsing addresses into semantically meaningful structures, can be useful for tasks such as building e-commerce or product recommendation systems (Jia et al., 2017; Avvenuti et al., 2018). Due to historical reasons, the English addresses come with a standardized format, mostly written in order from most specific to most general. Meaningful chunks in an English address are also separated by punctuation or the new-line symbols. Such characteristics make parsing English addresses a relatively easy task.

However, addresses written in eastern Asian languages such as Chinese present several unique challenges. Unlike English addresses, Chinese addresses are typically written in the form of a consecutive sequence of Chinese characters (possibly intermixed with digits and English letters). Figure 1 presents two example Chinese addresses and their desired output structures after parsing – chunks annotated with their labels indicating semantics (such as province, road, etc). The Chinese addressing system is also different from that of English. Though it is generally believed that the system uses the opposite ordering – starting from most general (e.g., province) and ending with most specific (e.g., room no.), in practice it can be observed that the format is far less rigorous than expected. The lack of rigor also leads to other issues – the addresses may come with incomplete, redundant or even inaccurate information, as we can see from the second example listed in Figure 1. Such unique challenges make the design of an effective Chinese address parser non-trivial.

Proceedings of NAACL-HLT 2019, pages 3421–3431
Minneapolis, Minnesota, June 2 - June 7, 2019.

Label	Order ID	Unique#	Train	Dev	Test	Interpretation	Example
COUNTRY	20	2	41	15	13	*name of a country*	中国(China)
PROVINCE	19	65	3,794	1,317	1,265	*name of a province*	浙江省(Zhejiang Province)
CITY	18	377	4,824	1,662	1,613	*name of a city*	北京市(Beijing)
DISTRICT	17	1137	5,881	2,027	1,921	*name of a district in a city*	朝阳区(Chaoyang District)
DEVZONE	16	297	330	118	107	*name of an economic development zone*	下沙开发区(Xiasha Development Zone)
TOWN	15	2,382	3,972	1,300	1,308	*name of a town or a boulevard*	乔司镇(Qiaosi Street)
COMMUNITY	14	1,867	1,279	415	416	*name of a community or a village*	荆山社区(Jingshan Community)
ROAD	13	5,037	5,410	1,788	1,801	*name of a road*	中山路(Zhongshan Road)
SUBROAD	12	486	333	109	130	*name of a lane*	丹心巷(Danxin Road)
ROADNO	11	2,676	4,316	1,435	1,401	*road number*	4-5号(#4-#5)
SUBROADNO	10	215	170	68	75	*road number for a subroad*	8号(#8)
POI	9	8,662	6,312	2,093	2,122	*name of the point of interest*	萧山医院(Xiaoshan Hospital)
SUBPOI	8	1,642	1,435	461	487	*name of the second point of interest*	西三苑(Xisan Sub-residence)
HOUSENO	7	1,309	2,993	978	943	*house number*	3幢(Block #3)
CELLNO	6	295	1,134	388	358	*cell number*	1单元(Unit #1)
FLOORNO	5	258	1,119	346	331	*floor number*	5层(Level 5)
ROOMNO	4	3,245	2,702	883	824	*room number*	402室(Room 402)
PERSON	3	903	650	207	208	*name of the third point of interest or a person*	大厅(the hall)
ASSIST	2	264	718	207	241	*a phrase for indicating relative position*	对面(opposite)
REDUNDANT	1	1,009	3,517	1,208	1,137	*redundant characters as well as repeated characters*	-,!
OTHERINFO	0	5	4	0	3	*a chunk which cannot be assigned any label above*	
Total Instances			8,957	2,985	2,985		
Total Chunks		32,133	50,934	17,024	16,704		

Table 1: Statistics of different labels in our *Chinese Address* corpus.

Parsing a Chinese address into semantically meaningful structures can be regarded as a special type of chunking task (Abney, 1991), where we need to perform address-specific Chinese word segmentation (Xue, 2003; Peng et al., 2004; Zhao et al., 2006) while assigning a semantic label to each chunk. However, existing models designed for chunking may not be readily applicable in this task. Our observations show that there are a few characteristics associated with the task. We found that while generally there exists certain ordering information among the chunks of different labels in the addresses, such ordering information is better preserved among the chunks that appear at the beginning of the addresses. For the chunks appearing towards the end of the addresses, chunks of different types often appear in more flexible order.

On top of the above observations, we propose a specific model based on neural networks for the task of Chinese address parsing. The model is able to encode the regular patterns among chunks that appear at the beginning of a Chinese address, while flexibly capturing the irregular patterns and rich dependencies among the chunks of different types that appear towards the end of the address. This is achieved by designing a novel structured representation integrating both a linear structure and a latent-variable tree structure.

Our main contributions in this work can be summarized as follows:

- We create and publicly release a new corpus consisting of 15K Chinese address entries fully annotated with chunk boundaries and address labels. To the best of our knowledge, this is the first and largest annotated Chinese address corpus.
- We introduce a novel neural approach to Chinese address parsing with latent variables to flexibly capture both prior ordering information and rich dependencies among labels.
- Through extensive experiments, we demonstrate the effectiveness of our approach. The experimental results show that our approach outperforms several baselines significantly.

2 Data

In this work, we created a *Chinese Address* corpus. To do so, we crawled a large number of publicly available addresses from the Chinese websites including online business directory websites (e.g., `b2b.huangye88.com`), social media websites (e.g., `www.dianping.com`), and an online API service translating a geo-location to a Chinese address (`lbs.amap.com`). In order to protect privacy, we discarded sensitive addresses (such as those involving military locations) and randomly altered the digits in the collected addresses.

Due to the lack of Chinese address standard format as well as complicated and different writing preferences in different regions (e.g., people living in southern China prefer the word “弄” as the suffix of the name of a lane or sub-road over the word “胡同” which is widely used in northern China), we create an annotation guideline[1] by summarizing different writing preferences. We proposed 21 chunk labels listed in Table 1. The meaning

---

[1] The annotation guideline can be found at `http://statnlp.org/research/sp`.

of most labels can be inferred from their names. We hire 3 annotators to annotate chunk boundaries and chunk labels for each Chinese address following the annotation guideline. In order to maintain high annotation quality, we also hire 2 additional quality controllers to sample 20 sentences from each batch of 1,000 annotated sentences for human evaluation. Re-annotation for that batch will be performed should the accuracy of human evaluation fall below 95%.

We randomly split the annotated data into 3 portions following the ratio of 60%, 20%, and 20%, yielding training, development, and test sets. The complete statistics of our data can be found in Table 1. From the table we can observe that the chunk label POI (point of interest) occurs most frequently. Indeed, such a label has a high level of importance. This is because location-based information can be extracted from such chunks, which is crucial for recommendation services (Gao et al., 2015; Xie et al., 2016). In addition, we report the number of distinctive chunks (unique#) that appear in the data for each label, from which we can see our corpus has a good coverage on PROVINCE, CITY, and DISTRICT[2].

We empirically assign each label a order ID indicating its level of specificity. For example, the label COUNTRY is used for describing a country, and is the most general concept. It is thus assigned the order ID 20, which is the highest among all labels. As another example, the label PERSON gets assigned an order ID 3, as it is used to describe one of the most specific concepts. Such order ID information will be useful later when designing our models for Chinese address parsing.

3 Approach

Our objective is to design a model for parsing Chinese addresses into semantically meaningful structures in the form of consecutive chunks, where each chunk is assigned a label as described in the previous section. As we have mentioned before, we believe there exist Chinese address-specific characteristics associated with address texts that can be exploited in designing a parsing model. Specifically, we argue there are two types of structured information within Chinese addresses that can be exploited when designing our parser – the *latent tree structures* and the *regular chain structures*. The former is used for capturing rich dependencies among chunks that appear towards the end of each address. The latter is used for capturing the structural patterns associated with chunks appearing at the beginning of each address.

3.1 Latent Tree Structures

We focus our discussions on the latent tree structures first. Given a consecutive sequence of labeled chunks, we can construct a binary tree structure whose yield exactly corresponds to the sequence of labeled chunks. We build the latent tree structures to capture complex dependencies based on the observation that chunks appearing towards the end of a given address do not follow a rigorous order. For instance, as we can see in the second example in Figure 1, chunks towards the end of the address consist of some labels related to numbers as well as the label REDUNDANT. These labels are either optional or do not follow some regular patterns in terms of order, which makes capturing dependencies among labels challenging.

We first introduce *auxiliary labels* based on the set of *original labels* we defined in the previous section. Such auxiliary labels are assigned to the internal nodes within a parse tree. Specifically, for each original label X, we introduce the auxiliary label $\overline{\text{X}}$. For example, the auxiliary label for ROAD would be $\overline{\text{ROAD}}$.

We now illustrate how a latent tree is constructed from a sequence of labeled chunks. These chunks will be regarded as a sequence of leaf nodes, each of which contains the corresponding chunk boundary and chunk type information. To simplify the construction process, we focus on building a specific type of binary trees with each non-leaf node containing at least 1 leaf node as one of its child.[3] We start the process by selecting any chunk first as one leaf node. Next we take a chunk that is either on the left or on the right of the selected chunk as its binary sibling node, and create a parent node by assigning the two selected leaf nodes as child nodes. To determine the label of the newly created parent node, we choose the auxiliary label based on the label with a higher order ID between the two of the child nodes. The newly created parent node will replace the 2 child nodes in the sequence and now the parent node becomes a selected node. We repeat this construction pro-

---

[2] We found these numbers are comparable with statistics on `www.stats.gov.cn`.

[3] Preliminary results show that considering arbitrary binary trees would lead to slightly worse results for our task.

cess until there is only 1 node left in the sequence. Note that the construction process makes use of the label order information.

Figure 2 shows an example tree that the gold chunks correspond to. From the example we can see that the non-leaf node label $\overline{\text{POI}}$ that appears twice has connections to other non-leaf node labels such as $\overline{\text{ROADNO}}$ and $\overline{\text{POI}}$. Such tree structures will allow us to capture rich Chinese address-specific structural information among labels. Since there are many latent trees corresponding to the given address consisting of consecutive labeled chunks, the model is facilitated to learn such complicated patterns, which is potentially beneficial for the address parsing task.

3.2 Regular Chain Structures

The latent tree structures allow complex dependencies between different chunks to be captured within a Chinese address. Such dependencies would be helpful when there exist irregular patterns within an address. However, if we believe there are regular patterns among the labeled chunks, using an alternative assumption on the dependencies to properly capture such patterns may be more desirable. For instance, the first example in Figure 1 illustrates a common regular pattern at the beginning of the address, which is the order of (PROVINCE, CITY, DISTRICT). This motivates us to employ an alternative representation for capturing dependencies within chunks that appear at the beginning of the addresses, which are believed to exhibit more regular patterns.

Specifically, we employ a chain structure to capture the dependencies between adjacent labeled chunks. For example, given a sequence of chunks, we may always consider a right-branching tree structure to connect all these chunks. The resulting structure will be able to capture first-order dependencies between adjacent labeled chunks, which allows the regular orders among the labels to be learned. For example, consider the first two chunks that appear within the address as illustrated in Figure 2. The first two chunks form a right-branching tree structure. The construction process for such chain structures is similar to that of the latent trees, except that there is a single fixed (right-branching tree) structure for given labeled chunks.

Based on the observation that regular patterns appear mostly at the beginning of an address, we define the space $\mathcal{H}(x, y, sp)$ that consists of all latent tree structures that are consistent with the input character sequence x, the gold labeled chunks y and sp which determines the split point. Formally, we define the split point of a given address as specified by sp as the left boundary of the rightmost chunk whose label order ID is larger than or equal to sp. The split point divides the chunks into two groups – those appearing on the left of sp will form a chain structure while those on the right will form a tree structure where the correct construction is latent. Both structures are then merged to form a single representation, which is used for building our address parsing model.

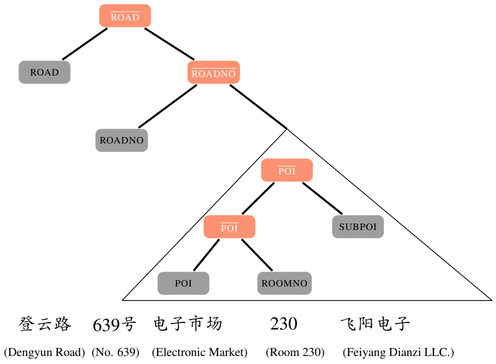


Figure 2: An example latent tree for given gold chunks where $sp = \text{POI}$. The English translation is listed below each chunk. Leaf nodes are in gray, and the internal nodes are in pink (labeled with white auxiliary labels). The tree structure within the triangle is latent – we show one of the many possible structures for illustration only.

Notice that when sp is set to -1 (denoted as $sp = \text{LAST}$), the split point is on the right of the last chunk. In this case the latent structured space $\mathcal{H}(x, y, sp)$ consists of only one single right-branching tree. On the other hand, when sp is set to its maximal value 20, the label order ID of COUNTRY (denoted as $sp = \text{COUNTRY}$), the latent structured space does not contain any structure that involves a partial regular chain component. Different values of sp leads to different interpolations between the two types of structural assumptions, resulting in different variants of our models. We will discuss the effect of different sp values in the experiments section.

3.3 Chunk Representation

A parse tree corresponds to a collection of labeled chunks as leaves. We adopt a bi-directional LSTM over a given input to compute the span-level representation. At each position i in the original input consisting of a sequence of characters, we use $\mathbf{f}_i$ and $\mathbf{b}_i$ to denote the outputs of forward LSTM and backward LSTM respectively. We use $\mathbf{c}_{i,j} = [\mathbf{f}_j - \mathbf{f}_i; \mathbf{b}_i - \mathbf{b}_j]$ to denote the vector representation of the span covering characters from position i to position j (Wang and Chang, 2016). Motivated by Stern et al. (2017), we define the label score as follows:

$$\mathbf{s}(i,j) = F(\mathbf{c}_{i,j})$$

where F is a 2-layer feed-forward neural network with output dimension being the number of chunk labels. In addition, we denote the score of the span with a specific label l as the value of the l-th element in the vector $\mathbf{s}(i,j)$:

$$s(i,j,l) = [\mathbf{s}(i,j)]_l \tag{1}$$

3.4 Model

Inspired by Stern et al. (2017), we build a chart-based parsing model. Unlike that work, however, our model involves latent structures as mentioned in Section 3.1. For a given sequence of labeled chunks, our model considers all possible constituent trees whose yield are exactly the labeled chunks.

Consider a tree $\mathbf{t}$ that can be represented by a set of labeled spans, where each span is uniquely defined by the boundary (i,j) and the label l:

$$\mathbf{t} := \{(i_n, j_n, l_n) : n = 1, \ldots, |\mathbf{t}|\}. \tag{2}$$

The score of the tree $\mathbf{t}$ can be defined as follows:

$$\mathcal{S}(\mathbf{t}) = \sum_{(i,j,l)\in\mathbf{t}} [s(i,j,l)] \tag{3}$$

Similar to (Stern et al., 2017), we use a CKY-style algorithm to calculate the score of the optimal sub-tree that spans the interval (i,j) recursive using the following formula:

$$\begin{aligned}\pi(i,j) = \max_l[s(i,j,l)] + \\ \max_k \Big\{ \max_l[s(i,k,l)] + \pi(k,j), \\ \pi(i,k) + \max_l[s(k,j,l)]\Big\}\end{aligned} \tag{4}$$

The base case is when the text span (i,j) corresponds to a leaf node (a chunk) in the tree; in this case we have: $\pi(i,j) = \max_l[s(i,j,l)]$.

3.5 Training and Decoding

Inspired by the structural support vector machines with latent variables (Yu and Joachims, 2009), we employ a (per instance) hinge loss during training:

$$L = \max_{\mathbf{t}\in\mathcal{H}(x)}[0, \boldsymbol{\Delta}(\mathbf{t},\mathbf{t}^*) + \mathcal{S}(\mathbf{t}) - \mathcal{S}(\mathbf{t}^*)] \tag{5}$$

where $\mathcal{H}(x)$ refers to the set of all possible trees for the given input x, and $\mathbf{t}^*$ denotes the best tree in the latent space $\mathcal{H}(x, y, sp)$:

$$\mathbf{t}^* = \max_{\mathbf{t}'\in\mathcal{H}(x,y,c)} \mathcal{S}(\mathbf{t}') \tag{6}$$

Here $\boldsymbol{\Delta}(\mathbf{t},\mathbf{t}^*)$ represents the Hamming loss on labeled spans, measuring the similarity between the predicted tree and the best latent tree that corresponds to the gold chunks.

During decoding, we aim to obtain the best tree as the prediction $\hat{\mathbf{t}}$ for a new address x' among all the possible trees:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}\in\mathcal{H}(x')}[\mathcal{S}(\mathbf{t})] \tag{7}$$

The yield of the predicted tree $\hat{\mathbf{t}}$ gives us the list of labeled chunks.[4]

4 Experimental Setup

We call our model Address Parser with Latent Trees (**APLT**). We conducted experiments based on different settings of the sp values, leading to many model variants. We describe baselines, model hyperparameters as well as evaluation metrics in this section.

Baselines To understand the effectiveness of our models, we build the following baselines:

- **ℓCRF** is the standard first-order linear CRF model (Lafferty et al., 2001) with discrete features for sequence labeling tasks.
- **sCRF** is based on the standard semi-Markov CRF (Sarawagi and Cohen, 2004) with discrete features[5].
- **LSTM** is the standard bi-directional LSTM model for sequence labeling tasks.

---

[4] In some cases, it is possible to predict a tree with one or more leaf chunks labeled with auxiliary labels (e.g., $\overline{\text{ROAD}}$). We have a post-processing step that converts such labels into their corresponding original labels (e.g., ROAD).

[5] See the supplementary material for details on the features for **ℓCRF** and **sCRF**. For **sCRF** (and **LSTM-sCRF**), maximal chunk length is set to 36, which is the length of the longest chunk appearing in the training set.

- **LSTM-ℓCRF** is proposed by Lample et al. (2016) which is the state-of-the-art for many sequence labeling tasks
- **LSTM-sCRF** is based on segmental recurrent neural network (Kong et al., 2016) which is the neural network version of semi-Markov CRF (Sarawagi and Cohen, 2004).
- **TP** is a transition-based parser for chunking based on Lample et al. (2016), which makes use of the *stack LSTM* (Dyer et al., 2015) to encode the representation of the stack.

Hyperparameters We conducted all the experiments based on our *Chinese Address* corpus. We pre-trained Chinese character embeddings based on the Chinese Gigaword corpus (Graff and Chen, 2005), using the skip-gram model with hierarchical softmax implemented within the *word2vec* toolkit (Mikolov et al., 2013) where we set the sample rate to 10^{-5} and embedding size to 100.

We use a 2-layer LSTM (for both directions) with a hidden dimension of 200. For optimization, we adopt the Adam (Kingma and Ba, 2014) optimizer to optimize the model with batch size 1 and dropout rate 0.4. We randomly replace the low frequency words with the *UNK* token and normalize all numbers by replacing each digit (including Chinese characters representing numbers from 0-9) to 0. We train our model for a maximal of 30 epochs and select the model parameters based on the F_1 score after each epoch on the development set. The selected model is then applied to the test set for evaluation. Our model, as well as the baseline neural models, are implemented using DyNet (Neubig et al., 2017). All the neural weights are initialized following the default initialization method used in DyNet.

Evaluation Metrics We use the standard evaluation metrics from the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000), reporting precision ($P.$), recall ($R.$) and F_1 percentage scores.

5 Result and Discussion

5.1 Main Results

We present our main results in Table 2, where we report the overall performance as well as specific results on the POI label. For our model, we report results for sp=20, -1 as two special cases – the former learns latent tree structures only and the latter assumes a single right-branching tree. We also report results for sp=7 which is selected based on the optimal results on the development set.

Model	POI			OVERALL		
	$P.$	$R.$	F_1	$P.$	$R.$	F_1
ℓCRF	69.76	72.68	71.19	87.78	85.33	86.53
sCRF	74.95	77.14	76.03	88.64	87.36	87.99
LSTM	70.11	76.90	73.35	85.63	88.11	86.85
LSTM-ℓCRF	77.94	75.62	76.76	88.83	88.88	88.86
LSTM-sCRF	77.80	77.84	77.82	89.21	88.52	88.86
TP	77.61	75.67	76.63	88.80	88.75	88.77
APLT $sp = 20$ (COUNTRY)	**80.36**	78.46	79.40	90.10	88.64	89.37
APLT $sp = -1$ (LAST)	79.64	78.89	79.26	90.06	89.07	89.56
APLT $sp = 7$ (HOUSENO)	79.75	**79.26**	**79.51**	**90.65**	**89.21**	**89.93**

Table 2: Main results.

Among all the baselines, **LSTM-ℓCRF** performs better than **LSTM** and **TP**, which is consistent with the finding reported in (Lample et al., 2016). The two models **LSTM-ℓCRF** and **LSTM-sCRF** both achieve similar results, which is also consistent with the finding reported in (Liu et al., 2016). The two non-neural models **ℓCRF** and **sCRF** perform substantially worse than their neural counterparts, which we believe is mainly due to the use of only handcrafted features in such systems. All these baseline models are capable of encoding transition patterns between neighboring chunks, which can partially capture certain structural information. However, certain Chinese address-specific structural information is not explicitly captured in such models.

Our model **APLT** (sp=7) achieves the best overall results, as well as the best results when evaluated on POI only. Compared with the strongest baselines **LSTM-ℓCRF** and **LSTM-sCRF**, **APLT** (sp=7) outperforms them significantly by more than 1 F_1 point overall ($p < 10^{-5}$)[6]. Furthermore, the **APLT** (sp=7) model obtains the best F_1 scores among all the models on POI. Note that our **APLT** model is able to learn richer dependencies among labels including label order information, regular patterns and irregular patterns among labels. Overall, the model **APLT** (sp=7) also outperforms both **APLT** (sp=-1) ($p < 0.05$) and **APLT** (sp=20) ($p < 0.005$) significantly. Such a result implies the importance of capturing the various Chinese address-specific structural information mentioned above within our model.

To understand the results better, we conduct detailed analysis of our results. Table 3 shows the F_1 scores of each label as well as the percentage of each label in the test data among four

---

[6] We perform the bootstrap resampling significant test.

Label	%	LSTM ℓCRF	LSTM sCRF	APLT sp=-1	APLT sp=7
Overall	100	88.86	88.86	89.56	**89.93**
POI	12.70	76.76	77.82	79.26	**79.51**
DISTRICT	11.55	95.04	95.04	95.46	**96.12**
ROAD	10.78	94.76	94.33	**95.28**	95.03
CITY	9.66	96.25	95.99	96.80	**97.03**
ROADNO	8.39	**95.32**	95.06	94.37	94.74
TOWN	7.83	92.07	92.05	**93.09**	92.90
PROVINCE	7.57	97.91	97.69	98.30	**98.42**
REDUNDANT	6.81	83.54	82.43	84.80	**85.98**
HOUSENO	5.65	90.62	90.07	89.83	**91.30**
ROOMNO	4.93	**91.15**	90.60	90.47	91.11
SUBPOI	2.92	57.70	**60.63**	60.58	59.41
COMMUNITY	2.49	74.79	75.06	76.53	**76.58**
CELLNO	2.14	92.29	91.01	**92.35**	90.78
FLOORNO	1.98	**98.03**	97.01	96.96	97.59
ASSIST	1.44	77.64	77.73	73.78	**78.32**
PERSON	1.25	61.58	61.68	61.69	**63.92**
SUBROAD	0.78	77.11	73.36	**80.00**	75.81
DEVZONE	0.64	63.85	63.11	**66.67**	64.13
SUBROADNO	0.45	70.50	63.24	**71.01**	67.65
COUNTRY	0.08	**96.00**	**96.00**	**96.00**	**96.00**
OTHERINFO	0.02	0.00	0.00	0.00	0.00

Table 3: F_1 score comparison on test data for each label among 4 models as well as the percentage of each label in the gold data.

models **LSTM-ℓCRF**, **LSTM-sCRF**, **APLT** (sp=-1) and **APLT** (sp=7). Note that the results for the top 4 labels POI, DISTRICT, ROAD and CITY, which take up 45% of total chunks, all get improved when using our **APLT** models. Moreover, it achieves better or comparable $F1$ scores on 15 labels in the table among the total 21 labels, especially on POI, DISTRICT, REDUNDANT, COMMUNITY and PERSON with at least 1 point improvement in F_1. Interestingly, our models perform worse than **LSTM-ℓCRF** on labels such as ROADNO, ROOMNO, and FLOORNO, which are mostly related to numbers. We note that, however, chunks with such labels do not constitute a large proportion of all chunks. Results suggest that our models somehow learned to focus on optimization performance for chunks with more prominent labels such as POI and DISTRICT.

5.2 Effectiveness of Structural Information

In order to investigate how tree structures affect the final performance, we also conducted experiments with different values for sp, which is used for determining the split point. Figure 3 shows the moving-averaged F_1 scores on the test set obtained when choosing sp around specific values (a similar distribution can be observed on the development set). From the bottom (COUNTRY,20) to the top (LAST,-1) along y axis, the lower the sp is, the more constraints are applied to the latent space $\mathcal{H}(x, y, sp)$. Note that when sp=-1 (LAST), the gold input only corresponds to a single right-branching tree. We exclude the following labels: REDUNDANT, ASSIST and OTHERINFO, because we found these labels may appear at any place within a given address, which make them unsuitable for determining the split point.

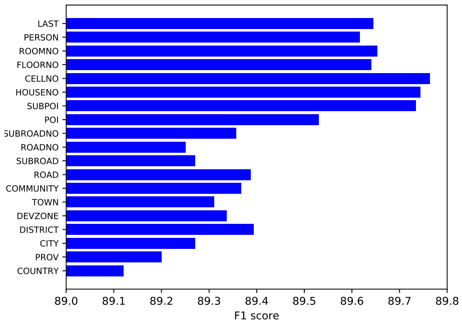


Figure 3: Effect of sp.

From Figure 3 we can observe that the F_1 score generally increases as we decrease sp, starting from COUNTRY(with order ID 20). The performance reaches the maximum when the sp is set to a value within the range [SUBPOI, CELLNO]. This observation implies that there does exist ordering information among labels, and introducing more constraints on the latent space will have the benefit of modeling the regular patterns around the beginning part of a given address. After reaching the best value, as we further decrease sp, the performance drops slightly and oscillates around the range [FLOORNO, LAST]. From here we can observe that the latent trees are able to help capture irregular patterns within labels that appear towards the end of the address. Overall, these results suggest the importance of designing a model like ours that is capable of capturing Chinese address-specific characteristics.

5.3 Error Analysis

We conduct error analysis on two strongest baselines **LSTM-ℓCRF** and **LSTM-sCRF** as well as two best-performing **APLT** models respectively. We examined the list of top-10 labels with most errors for each model, and found most of the errors come from labels such as POI, SUBPOI and REDUNDANT – this implies they are the most challenging labels for this task. We also found labels such as ROOMNO appear in the list for **APLT** models, but not for the **LSTM-ℓCRF** model, showing that **APLT** models are still not good at handling numbers as we discussed above.

There are two major types of errors. The type-

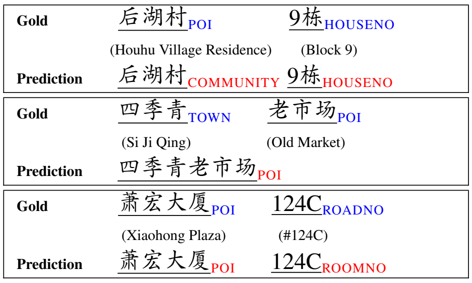


Figure 4: Example outputs from **APLT**(*sp*=7).

Length	%	LSTM ℓCRF	LSTM *s*CRF	APLT *sp*=−1	APLT *sp*=7
1	9.39	**92.16**	92.14	91.65	91.77
2	**23.73**	86.69	86.13	87.16	**87.77**
3	**44.60**	92.26	92.04	93.03	**93.51**
4	**13.31**	86.49	87.48	88.05	**88.18**
5	3.70	74.57	76.41	77.55	**79.43**
6	2.14	68.88	70.19	70.87	**73.73**
7	1.16	64.61	68.14	67.59	**68.22**
≥8	1.97	63.31	62.57	**63.33**	60.19

Table 4: Results for different chunk lengths.

I error refers to the case where the boundary of a chunk is predicted correctly but not its label. The type-II error is the case where even the boundary of a predicted chunk is incorrect. We found that **APLT** (*sp*=−1) and **APLT** (*sp*=7) produce less type-I errors (45.04% and 42.95% respectively) than **LSTM-ℓCRF** and **LSTM-*s*CRF** (49.87% and 47.26% respectively). Moreover, we find that **APLT** (*sp*=7) model produces the least number of type-I errors as well as type-II errors.

Looking into the type-I errors of both two **APLT** models, we find chunks with label POI are often incorrectly labeled as COMMUNITY, which is a major source of errors (9% of total errors). As a typical example, we show a partial prediction in Figure 4, where our model fails to recognize "后湖村"(Houhu Village Residence) as a POI. Here the character "村"(Village) is a common suffix for the name of either a village or a residence, hence the confusion.

The second example in Figure 4 demonstrates another typical kind of errors produced by our models around the POI labels. Here, "四季青(Si Ji Qing)" is actually the name of a town. However, as most names of towns end with "镇(Town)" as the suffix, our models as well as baseline models all fail to identify the correct chunk boundaries.

We also investigate the errors around the number labels. We choose to look into the results on ROADNO because it is the fifth most popular label in the test data. Based on the error analysis, we found that many chunks of label ROADNO were incorrectly assigned other types of number labels. As we can see from the third example in Figure 4, the ROADNO "124C" is incorrectly predicted as a ROOMNO. Indeed, this chunk does look like a room number, though in fact it refers to a road within a "plaza" (大厦) rather than an office within a "building" (another interpretation of 大厦). From these examples we can observe that many ambiguities may not be easily resolved without further background knowledge.

5.4 Robustness Analysis

We analyze the model robustness by assessing the performance on chunks of different lengths for each of the four models discussed above. We group chunks into 8 categories based on their lengths and present the results in Table 4 where the distribution information is also included. As we can see, all the models achieve at least a F_1 score of 86 when considering chunks whose lengths are less than 5. As the length increases, the performance of all models drop gradually. For chunks whose lengths are at least 8, the F_1 score is around 60-63 for all models. Considering chunks whose lengths are either 2, 3, or 4 only (such chunks constitute over 80% of total chunks), we can observe that **APLT** (*sp*=7) outperforms two baselines significantly by more than 1 point for each category. These results demonstrate the robustness of our model when handling chunks of different lengths.

Comparing the two **APLT** models, we can see the model **APLT** (*sp*=7) outperforms **APLT** (*sp*=−1) for each chunk category, except for chunks whose lengths are greater than or equal to 8. These two models differ in their latent spaces. **APLT** (*sp*=7) with a richer latent space appears to be better at handling chunks with short or medium lengths.

In addition, we conducted a further experiment to understand how each model is able to handle new chunks – the chunks that appear in the test set (according to the gold labels) but do not appear in the training set. We found empirically there are 31% of the chunks in the test set that are new chunks. Such an experiment allows us to assess the robustness of each model when new data is available. We report the accuracy for the new chunks in Table 5. As we can see, two **APLT** models outperform two baselines, indicating our **APLT** models appear to be better at handling new chunks. We believe this is due to the tree models

LSTM	**LSTM**	**APLT**	**APLT**
ℓ**CRF**	s**CRF**	$sp{=}-1$	$sp{=}7$
80.17	79.92	80.94	80.94

Table 5: Accuracy on test data for the new chunks.

that we used, which are capable of capturing complex dependencies among chunks.

6 Related Work

While the Chinese address parsing task is new, it is related to the following traditional tasks within the field of natural language processing (NLP) – chunking, named entity recognition, word segmentation and parsing. We briefly survey research efforts which are most related to our task below.

Chunking as a fundamental task in NLP has been investigated for decades (Abney, 1991). Chunking for Chinese can typically be regarded as a sequence labeling problem solvable by models such as conditional random fields (Chen et al., 2006; Tan et al., 2005; Zhou et al., 2012), hidden Markov models (Li et al., 2003), support vector machines (Tan et al., 2004) and the maximum entropy model (Wu et al., 2005). Our task can also be regarded as a chunking task where we need to assign an address-specific label to each chunk.

Named entity recognition (NER) is another fundamental task close to chunking within the field of NLP, which focuses on the extraction of semantically meaningful entities from the text. The state-of-the-art approach by Lample et al. (2016) employs a LSTM-CRF model. Ma and Hovy (2016) proposed a LSTM-CNNs-CRF model that utilizes convolutional neural networks (CNNs) to extract character-level features besides word-level features. Zhai et al. (2017) suggested a neural chunking model based on pointer networks (Vinyals et al., 2015) to resolve the issue of being difficult to use chunk-level features such as the length of the chunk for segmentation. Zhang and Yang (2018) tackled the problem of Chinese NER by deploying a lattice LSTM leveraging lexicons.

Another task closely related to our task is the Chinese word segmentation task which at least dates back to the 1990s (Sproat et al., 1994). The segmentation task is typically casted as a character-based sequence labeling problem (Xue, 2003) which can be solved by CRF based models (Peng et al., 2004; Zhao et al., 2006), their latent-variable variants (Sun et al., 2009), or max-margin based models (Zhang and Clark, 2007). Recently, Zhang et al. (2016) proposed a neural transition-based segmentation approach by encoding both words and characters as well as the history action sequence. Yang et al. (2017) suggested to perform segmentation with a neural transition-based method with rich pre-training.

Constituent parsing is another line of work that is related to our task. The state-of-the-art approaches to parsing include transition-based models (Dyer et al., 2016) and chart-based models (Stern et al., 2017; Kitaev and Klein, 2018).Our model is motivated by the latter approaches, where we additionally introduce latent variables for capturing complex dependencies among chunks.

7 Conclusion

In this work, we introduce a new task – *Chinese address parsing*, which is to segment a given Chinese address text into chunks while assigning each chunk a semantically meaningful label. We create and publish a *Chinese address* corpus that consists of 15K fully labeled Chinese addresses. We identify interesting characteristics associated with the task and design a novel neural parsing model with latent variables for this task, which is able to capture Chinese address-specific structural information. We conduct extensive experiments and compare our approach with strong baselines through detailed analysis. We show that our proposed model outperforms baseline approaches significantly, due to its ability in capturing rich structural information present in the Chinese addresses.

Future work includes leveraging external knowledge bases to disambiguate chunks and entities that appear within Chinese addresses, as well as designing algorithms that are able to capture longer-range dependencies among chunks using alternative structures.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments on this work. This work is done under a collaborative agreement between SUTD and Alibaba on an Alibaba Innovative Research (AIR) Program funded by Alibaba, where Alibaba provided data. We appreciate Alibaba's generosity in the agreement that makes it possible for us to make all data and code in this research publicly available upon acceptance of this paper. This work is also partially supported by SUTD project PIE-SGP-AI-2018-01.

References

Steven P Abney. 1991. Parsing by chunks. In *Principle-based parsing*, pages 257–278. Springer.

Marco Avvenuti, Stefano Cresci, Leonardo Nizzoli, and Maurizio Tesconi. 2018. Gsp (geo-semantic-parsing): Geoparsing and geotagging with machine learning on top of linked data. In *Proc. of ESWC*.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *Proc. of ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2015. Content-aware point of interest recommendation on location-based social networks. In *Proc. of AAAI*.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN*, 1:58563–58230.

Ya-Hui Jia, Wei-Neng Chen, Tianlong Gu, Huaxiang Zhang, Huaqiang Yuan, Ying Lin, Wei-Jie Yu, and Jun Zhang. 2017. A dynamic logistic dispatching system with set-based particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proc. of ACL*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proc. of ICLR*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL*.

Heng Li, Jonathan J Webster, Chunyu Kit, and Tianshun Yao. 2003. Transductive hmm based chinese text chunking. In *Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on*, pages 257–262. IEEE.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proc. of IJCAI*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proc. of ACL*.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*.

Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1994. A stochastic finite-state word-segmentation algorithm for chinese. In *Proc. of ACL*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proc. of ACL*.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proc. of NAACL*.

Yongmei Tan, Tianshun Yao, Qing Chen, and Jingbo Zhu. 2004. Chinese chunk identification using svms plus sigmoid. In *Proc. of IJCNLP*.

Yongmei Tan, Tianshun Yao, Qing Chen, and Jingbo Zhu. 2005. Applying conditional random fields to chinese shallow parsing. In *Proc. of CICLing*.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proc. of CoNLL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. of NIPS*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proc. of ACL*.

Shih-Hung Wu, Cheng-Wei Shih, Chia-Wei Wu, Tzong-Han Tsai, and Wen-Lian Hsu. 2005. Applying maximum entropy to robust chinese shallow parsing. In *Proc. of IJCLCLP*.

Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *Proc. of CIKM*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, 8(1):29–48.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proc. of ACL*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proc. of ICML*.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Proc. of AAAI*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proc. of ACL*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proc. of ACL*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proc. of ACL*.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proc. of PACLIC*.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2012. Exploiting chunk-level features to improve phrase chunking. In *Proc. of EMNLP*.

Learning Hierarchical Discourse-level Structure for Fake News Detection

Hamid Karimi
Computer Science and Engineering
Michigan State University
`karimiha@msu.edu`

Jiliang Tang
Computer Science and Engineering
Michigan State University
`tangjili@msu.edu`

Abstract

On the one hand, nowadays, fake news articles are easily propagated through various online media platforms and have become a grand threat to the trustworthiness of information. On the other hand, our understanding of the language of fake news is still minimal. Incorporating hierarchical discourse-level structure of fake and real news articles is one crucial step toward a better understanding of how these articles are structured. Nevertheless, this has rarely been investigated in the fake news detection domain and faces tremendous challenges. First, existing methods for capturing discourse-level structure rely on annotated corpora which are not available for fake news datasets. Second, how to extract out useful information from such discovered structures is another challenge. To address these challenges, we propose **H**ierarchical **D**iscourse-level **S**tructure for **F**ake news detection. HDSF learns and constructs a discourse-level structure for fake/real news articles in an automated and data-driven manner. Moreover, we identify insightful structure-related properties, which can explain the discovered structures and boost our understating of fake news. Conducted experiments show the effectiveness of the proposed approach. Further structural analysis suggests that real and fake news present substantial differences in the hierarchical discourse-level structures.

1 Introduction

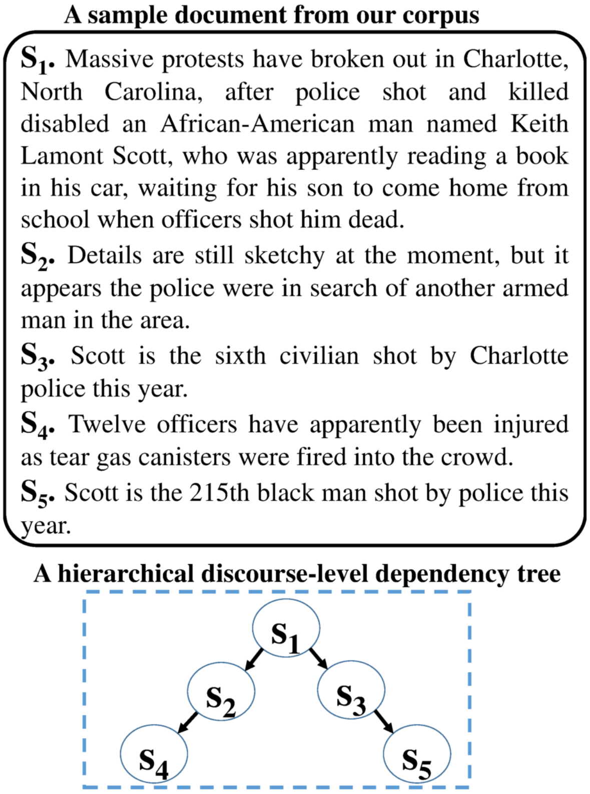
In this work, we focus on detecting fake news articles (hereafter referred to as *documents*) based on their contents. Many existing linguistic approaches for fake news detection (Feng et al., 2012; Pennebaker et al., 2015; Ott et al., 2011) overlook a crucial linguistic aspect of fake/real news documents i.e., the hierarchical discourse-level structure. Usually, in a document, discourse units (e.g., sentences) are organized in a hierarchical structure e.g., a tree. The importance of considering the hierarchical discourse-level structure for fake news detection is three-fold. First, previous studies (Bachenko et al., 2008; Rubin and Lukoianova, 2015) explored discourse-level structure in fake news detection and discovered that the way two discourse units of a document are connected could be quite revealing and insightful about its truthfulness. For instance, (Rubin and Lukoianova, 2015) applied Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) and noted that fake stories lack "evidence" as a defined inter-discourse relation. Second, fake news is typically produced by connecting disjoint pieces of news and unlike well-established journalism (e.g., New York Times) fake news production lacks a meticulous editorial board. Therefore, by incorporating the hierarchical discourse-level structure, we can investigate the coherence of fake/real news documents (we will show this later). Third, a substantial number of studies have shown that using hierarchical structures yields a better document representation in various downstream tasks whose predictions depend on the entire text (Bhatia et al., 2015; Morey et al., 2018; Li et al., 2014b). Since typically fake news detection is considered as a classification problem based on the entire text, applying discourse analysis has the potential to advance fake news detection (this will be verified later).

On the other hand, incorporating the hierarchical structure at the discourse level for fake news detection faces tremendous challenges. First, many existing methods incorporating structural discourse (Li et al., 2014a; Bhatia et al., 2015) (not for fake news detection though) rely on annotated corpora such as Penn Discourse Treebank (Prasad et al., 2007). Constructing and annotating such corpora is an arduous and costly process. Incor-

Proceedings of NAACL-HLT 2019, pages 3432–3442
Minneapolis, Minnesota, June 2 - June 7, 2019.

porating hierarchical structure is even more difficult for fake news detection as there exists virtually no available annotated discourse corpus in this domain. Therefore, we need to learn the discourse-level structure in a data-driven and automated manner. Second, how to use the hierarchical discourse-level structure and extract insightful information that can boost our understanding of fake news is another challenge.

In this study, we embrace the opportunities and challenges and propose **H**ierarchical **D**iscourse-level **S**tructure for **F**ake news detection (HDSF) framework. HDSF in an automated manner learns a hierarchical structure for a given document through an approach based on the dependency parsing at the sentence level (i.e., sentences are discourse units). As an example, Figure 1 illustrates a hierarchical structure of a document i.e., a dependency tree of sentences. Sentences represent discourse units and two sentences are connected by a directed link where a sentence at the *head* of the link *semantically depends* on a sentence at the *tail* of the link e.g., in Figure 1, the sentence s_4 depends on the sentence s_2.



A sample document from our corpus

S_1. Massive protests have broken out in Charlotte, North Carolina, after police shot and killed disabled an African-American man named Keith Lamont Scott, who was apparently reading a book in his car, waiting for his son to come home from school when officers shot him dead.

S_2. Details are still sketchy at the moment, but it appears the police were in search of another armed man in the area.

S_3. Scott is the sixth civilian shot by Charlotte police this year.

S_4. Twelve officers have apparently been injured as tear gas canisters were fired into the crowd.

S_5. Scott is the 215th black man shot by police this year.

A hierarchical discourse-level dependency tree

Figure 1: An illustration of the hierarchical discourse-level structure of a document using a dependency tree

Our key contributions are summarized as follow.

- To the best of our knowledge, we are the first to study automatic document structure learning for fake news detection.
- We propose the framework HDSF that automatically and in an end-to-end manner learns structurally rich representations for fake and real news documents.
- We identify a set of structure-related properties delineating meaningful structural differences between fake and real news documents.

The rest of the paper is organized as follows. In Section 2, we formally define the problem. Section 3 describes the proposed framework. In Section 4, we introduce several structure-related properties to analyze the hierarchical structures. Section 5 presents the experiments and discussions followed by an overview of the related work in Section 6. We conclude the paper in Section 7 and shed light on a few future directions.

2 Notations and Problem Statement

Following the previous work (Allcott and Gentzkow, 2017; Shu et al., 2017), we define the fake news as follows.

Definition. We define a news document *fake* if its content is verified to be false and *real* otherwise.

Let's briefly introduce some notations. Suppose we have a corpus $\mathcal{D}$ of fake and real news documents. Let a document $d \in \mathcal{D}$ contain k sentences $s_1, s_2, \cdots s_k$. Suppose a sentence $s_j \in d$ ($1 \leq j \leq k$) includes words $W_j = \{w_1, w_2, \cdots, w_{T_j}\}$ where T_j denotes the number of words in sentence s_j. Additionally, binary labels Y (i.e., *fake* or *real* labels) hold ground-truth labels associated with documents in $\mathcal{D}$.

Given the corpus of fake/real news documents (i.e., $\mathcal{D}$), we aim to learn model $\mathcal{M}$ that can automatically learn hierarchical and structurally rich representations for documents in $\mathcal{D}$. Meanwhile, given binary labels Y, the model $\mathcal{M}$ uses the hierarchical representations to automatically predict the labels of unseen news documents.

3 The Proposed Framework

To incorporate the discourse-level structure for fake news detection, we propose the framework HDSF illustrated in Figure 2. It provides three components: the *Hierarchical Discourse-level*

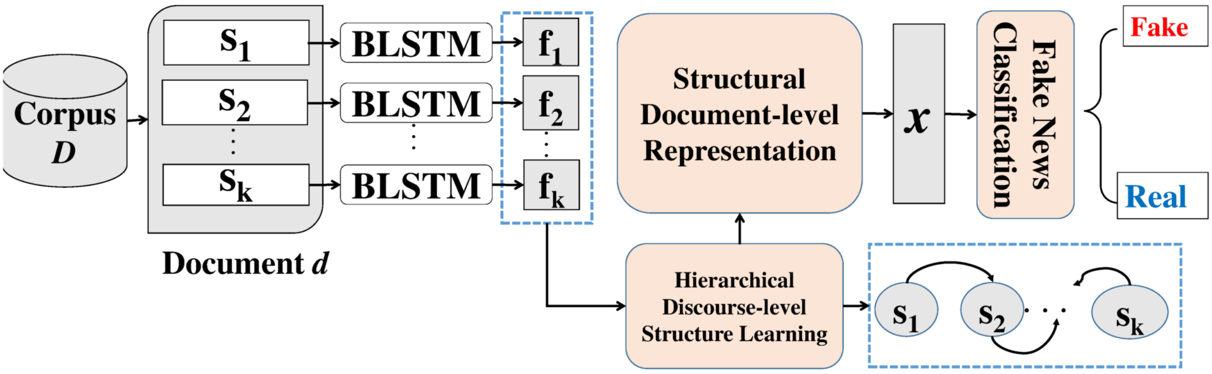


Figure 2: The proposed framework Hierarchical Discourse-level Structure for Fake news detection (HDSF)

Structure Learning automatically learns a proper structure for a given document, the *Structural Document-level Representation* yields a numerical and unified representation for the entire document, which is used by the *Fake News Classification* component to predict the label of a document.

3.1 Hierarchical Discourse-level Structure Learning

In this component, we aim to construct a hierarchical structure between discourse units (i.e., sentences) without relying on an annotated corpus. To achieve this, we use a dependency parsing approach (Liu and Lapata, 2018; Li et al., 2014a; Kim et al., 2017) and represent the hierarchical structure of a document as a dependency tree (see Figure 1 for an example of a dependency tree). In dependency parsing of discourse units, we mainly need to identify if a discourse unit semantically depends on another one. If so, a *parent-child* (or *tail-head*) link in the dependency tree can be established. Therefore, as long as the semantic dependencies between discourse units are identified, we can construct a discourse-level dependency tree without an annotated corpus. In Section 3.1.1, we describe a method to discover the semantic dependencies between discourse units in an automated manner. Afterward, in Section 3.1.2, we utilize these dependencies to construct the dependency tree of a document.

3.1.1 Learning Inter-Discourse Dependencies

Since discourse units are defined as sentences, we first need to get a fixed representation for each sentence. To this end, we utilize Bi-directional Long Short-Term Memory (BLSTM) network (Schuster and Paliwal, 1997). We represent each word in a sentence s_j by a fixed-size word embedding, and further the BLSTM network at each time step $t \in [1, T_j]$ executes the following functions[1]:

$$\begin{aligned} \overrightarrow{h}_t &= \mathcal{F}(\overrightarrow{h}_{t-1}; w_{t-1}) \\ \overleftarrow{h}_t &= \mathcal{F}(\overleftarrow{h}_{t-1}; w_{T_j-t+1}) \end{aligned} \qquad (1)$$

where $\mathcal{F}$ is the LSTM function (Hochreiter and Schmidhuber, 1997), and $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are outputs of the forward and backward LSTM networks at time step t, respectively. Then, a fixed representation for a sentence s_j, denoted as f_j, is defined as the average of the last output of forward and backward LSTM networks:

$$f_j = \frac{[\overrightarrow{h}_{T_j} + \overleftarrow{h}_{T_j}]}{2} \qquad (2)$$

Similarly, we apply the BLSTM network to all sentences of a document and obtain a sequence of sentential representations i.e., $f_1, f_2, \cdots, f_k$ (see Figure 2).

As mentioned before, in dependency parsing, we need to identify the dependency between two discourse units in an automated manner. To do this, the HDSF framework learns and optimizes an inter-sentential attention matrix $\mathbf{A} \in \mathbb{R}^{k \times k}$. The entry (m, n) of $\mathbf{A}$ holds the probability of the sentence s_m being the parent of the sentence s_n where $1 \leq m, n \leq k$ and $m \neq n$. In other words, $\mathbf{A}$ contains parent-child probabilities and is computed as follows.

$$\begin{aligned} u_m &= \mathcal{G}(\mathbf{W} \times f_m + \mathbf{b}) \\ u_n &= \mathcal{G}(\mathbf{W} \times f_n + \mathbf{b}) \\ \mathbf{A}[m, n] &= \frac{e^{(u_m \odot u_n)}}{\sum_{i=1}^{k} e^{\sum (u_i \odot u_n)}} \end{aligned} \qquad (3)$$

---

[1] Note that the backward LSTM (i.e., the lower part of Eq. 1) takes a sequence of words in a reverse order.

where $\mathcal{G}$ is a non-linear activation function, $\mathbf{W}$ is some weight matrix, $\mathbf{b}$ is a bias vector, and $\odot$ denotes the dot product operator. Further, since we need a root node in a dependency tree, we compute the probability of a sentence s_j being the root node, denoted as r_j, as follows.

$$\begin{aligned} u_j &= \mathcal{G}(\mathbf{W} \times f_j + \mathbf{b}) \\ r_j &= \frac{e^{\sum_{\forall y} u_j[y]}}{\sum_{i=1}^{k} e^{\sum_{\forall y} u_i[y]}} \end{aligned} \tag{4}$$

where $u_j[y]$ is the y-th element of vector u_j. Similarly, we calculate the root probabilities for all sentences and obtain the array of root probabilities denoted as $\mathbf{r} = \{r_1, r_2, \cdots, r_k\}$ where $0 \leq r_j \in \mathbf{r} \leq 1$.

3.1.2 Discourse Dependency Tree Construction

We use the learned matrix of inter-sentential parent-child probabilities i.e., $\mathbf{A}$ (Eq. 3) as well as the array of root probabilities i.e., $\mathbf{r}$ (Eq. 4) and propose a greedy algorithm, illustrated in Algorithm 1, to construct the discourse dependency tree of a document. A sentence with the maximum value in $\mathbf{r}$ is considered the root node and is inserted into the tree (line 5). Then, at each iteration, the algorithm finds the maximum entry in a block of the matrix $\mathbf{A}$ whose rows correspond to the rows of current nodes added to the tree (i.e., nodes V in line 7) and its columns correspond to columns of the rest of nodes (i.e., nodes $N \backslash V$ in line 7). Note that the columns of the current nodes are excluded because their parents have already been identified and also each node should have exactly one parent (except the root which has no parent). Assume the search in line 7 results in the entry (p, c) of $\mathbf{A}$ where $1 \leq p, c \leq k$ and $p \neq c$. Then, the sentence s_c is added as the child node of the sentence s_p (line 8). Algorithm 1 continues until all sentences of a document are added to the tree $\mathcal{T}$.

To fix the idea of discourse dependency tree construction algorithm, we present a step-by-step execution of this algorithm demonstrated in Figure 3. In Step 0, the sentence s_1 is added as the root of the tree $\mathcal{T}$ since it has the maximum value in the array of the root probabilities. Next in Step 1, the algorithm searches for the maximum probability value in the row s_1 while the column s_1 is excluded. The maximum value is this block is 0.4 and corresponds to entry (s_1,s_2). Therefore, the sentence s_2 is added as the child node of the sentence s_1. In a similar fashion, the algorithm continues until all 6 sentences are added to the tree.

Algorithm 1: The proposed algorithm for discourse dependency tree construction

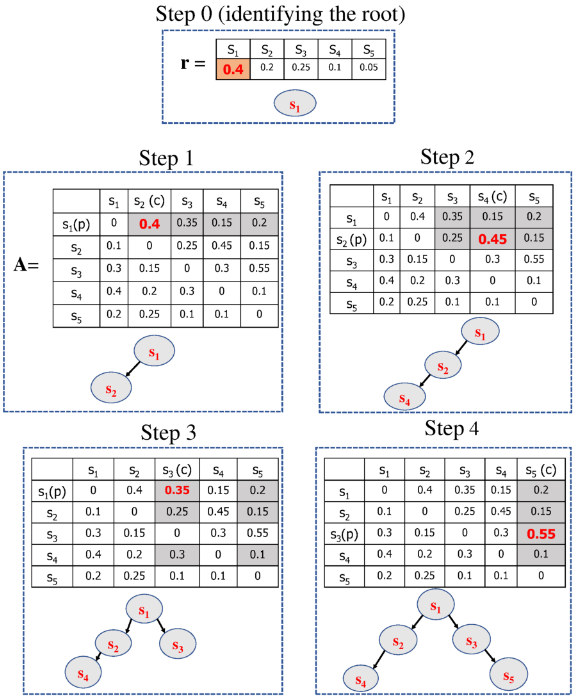
```
    Input: $\mathbf{A}$; $\mathbf{r}$
    Output: Discourse dependency tree $\mathcal{T}$
 1  $\mathcal{T} = empty$
 2  $N = \{s_1, s_2, \cdots, s_k\}$ // All nodes
 3  $V = \{\}$ // Set of current nodes
 4  Add $N[argmax(\mathbf{r})]$ to $V$
 5  $\mathcal{T}.root = N[argmax(\mathbf{r})]$ // Adding the root
 6  while $|V| \neq k$ do
 7      $p, c = argmax(\mathbf{A}[V, N \backslash V])$ // Search block
 8      $\mathcal{T}.link(N[c], N[p])$ // Child-parent link
 9      $V.add(N[c])$ // Adding the child node
10  end
11  return $\mathcal{T}$
```

3.2 Structural Document-level Representation

We use a similar method presented in (Liu and Lapata, 2018) to extract a structurally rich representation for the entire document. First, for each sentence (i.e., a discourse unit), we obtain a structurally-aware representation. To achieve this, we take into account the parent-child probabilities as well as the root probabilities as follows.

$$\begin{aligned} \mathbf{p}_j &= r_j \times \mathbf{e}_{root} + \sum_{z=1}^{k} \mathbf{A}[z, j] \times f_z \\ \mathbf{c}_j &= \sum_{z=1}^{k} \mathbf{A}[j, z] \times f_j \\ \mathbf{g}_j &= \mathcal{G}(\mathbf{W}[\mathbf{p}_j || \mathbf{c}_j || f_j] + \mathbf{b}) \end{aligned} \tag{5}$$

where $\mathbf{p}_j$ and $\mathbf{c}_j$ are two context vectors taking into account possible parents and children of a sentence s_j, respectively, $\mathbf{e}_{root}$ denotes a special root embedding vector, $||$ denotes vector concatenation operator, and $\mathbf{g}_j$ is a *structurally-aware representation for sentence* s_j. Finally, to extract a structurally rich representation for the entire doc-



Step 0 (identifying the root)

$\mathbf{r} =$	s1	s2	s3	s4	s5
	0.4	0.2	0.25	0.1	0.05

Step 1

A=	s1	s2 (c)	s3	s4	s5
s1(p)	0	**0.4**	0.35	0.15	0.2
s2	0.1	0	0.25	0.45	0.15
s3	0.3	0.15	0	0.3	0.55
s4	0.4	0.2	0.3	0	0.1
s5	0.2	0.25	0.1	0.1	0

Step 2

	s1	s2	s3	s4 (c)	s5
s1	0	0.4	0.35	0.15	0.2
s2 (p)	0.1	0	0.25	**0.45**	0.15
s3	0.3	0.15	0	0.3	0.55
s4	0.4	0.2	0.3	0	0.1
s5	0.2	0.25	0.1	0.1	0

Step 3

	s1	s2	s3 (c)	s4	s5
s1(p)	0	0.4	**0.35**	0.15	0.2
s2	0.1	0	0.25	0.45	0.15
s3	0.3	0.15	0	0.3	0.55
s4	0.4	0.2	0.3	0	0.1
s5	0.2	0.25	0.1	0.1	0

Step 4

	s1	s2	s3	s4	s5 (c)
s1	0	0.4	0.35	0.15	0.2
s2	0.1	0	0.25	0.45	0.15
s3(p)	0.3	0.15	0	0.3	**0.55**
s4	0.4	0.2	0.3	0	0.1
s5	0.2	0.25	0.1	0.1	0

Figure 3: An illustration of a step-by-step execution of Algorithm 1 for the sample document presented in Figure 1. The search block at each iteration has been highlighted. Note the letter ‘p’ for the *parent node* and ‘c’ for the *child node*.

ument, we average all of its $\mathbf{g}_j$ vectors:

$$x = \frac{1}{k}\sum_{j=1}^{k}\mathbf{g}_j \tag{6}$$

where x denotes *the structurally rich document-level representation*.

3.3 Fake News Classification

We hypothesize that the document-level structurally rich representations (Eq. 6) offer a discriminatory power to detect fake news documents. Therefore, as shown in Figure 2, we employ a binary classification for fake news detection formulated as follows.

$$\begin{aligned} c_i &= -\big(y_i log(p^i_f) + (1-y_i)log(p^i_r)\big) \\ \mathcal{L}(\theta) &= \sum_{\forall d_i \in \mathcal{D}} c_i \end{aligned} \tag{7}$$

where y_i is the ground-truth label of a document $d_i \in \mathcal{D}$, c_i is the cross-entropy loss value, p^i_f and p^i_r are probabilities of d_i being fake or real, respectively. p^i_f and p^i_r are obtained from the representation x (Eq. 6) fed to a fully connected layer followed by the softamx function. In Eq. 7, θ denotes the framework’s parameters and $\mathcal{L}$ is the total loss of the model. Since the entire framework (Eq. 1 to 7) is fully differentiable, we utilize the error backpropagation (Hecht-Nielsen, 1992) to calculate the gradients followed by the stochastic gradient descent (Bottou, 2010) to update and optimize the parameters. Note that the discourse dependency tree construction algorithm presented in Section 3.1.2 is employed post hoc. Hence, the gradient calculation over $argmax$ operator (line 7 of Algorithm 1) is not performed.

4 Structure-related Properties

Incorporating discourse-level structure offer a discriminatory power distinguishing fake and real documents, which will be verified in Section 5.3. We expect more than this discriminatory power. We expect to identify insightful and interpretable information from extracted structures which can delineate intrinsic differences between fake and real news documents. To meet this expectation, we define three fundamental properties of constructed discourse dependency trees. Note that we leave the definitions of more advanced properties as one future work. We seek to fulfill two goals by defining the structure-related properties. First, we intend to highlight the way that fake and real news documents are different. Second, we intend to leverage these properties to shed light on the *coherence* of fake and real news documents. Coherence has been the subject of many studies (Barzilay and Lapata, 2008; Lin et al., 2011; Guinaudeau and Strube, 2013; Li and Hovy, 2014) and is concerned with how constituents of a document (e.g., discourse units) are linked together in a way that the entire text creates a clear mental picture to the readers (Storrer, 2002). Notwithstanding its importance, coherence has not been investigated in the fake news domain in a large-scale and systematic fashion. Aiming at filling this gap, we naturally connect the defined structure-related properties with the coherence of news documents.

Property 1 (Number of Leaf Nodes) *This property, denoted as* P_l*, defines the normalized number of leaf nodes in a discourse dependency tree:*

$$\mathsf{P}_l = \frac{l}{log(k)} \tag{8}$$

where l is the number of leaf nodes (i.e., sentences) in the discourse dependency tree of a document. Recall that k is the total number of sentences in a document.

The intuition behind defining Property 1 is as follows. According to the description of the dependency tree in Section 3.1, leaf nodes are isolated discourse units and no other discourse units depend on them. Thus, the more the number of leaf nodes is, the less inter-linked the discourse units will be, and vice versa. Therefore, Property 1 is likely to indicate the coherence of a document – the higher P_l, the more isolated sentences and the less coherent the document. Also, for a document with k sentences $\mathsf{P}_l \in [\frac{1}{log(k)}, \frac{k-1}{log(k)}]$.

Property 2 (Preorder Difference) *This property, denoted as* P_t*, defines the normalized positional difference between the preorder traversal of a document's discourse dependency tree and its original sentential sequential order:*

$$\mathsf{P}_t = \frac{\sum_{j=1}^{k} |s_j^{position} - j|}{log(k)} \tag{9}$$

where $s_j^{position}$ denotes the position of a sentence s_j in the preorder traversal[1] of dependency tree of a document e.g., $s_3^{position} = 4$ in Figure 1. The position of s_j in the original sequential order is simply j[2]. The preorder traversal of the tree in Figure 1 is the sequence $\{s_1, s_2, s_4, s_3, s_5\}$ and the sentential sequential order is $\{s_1, s_2, s_3, s_4, s_5\}$. Therefore, according to the definition of Property 2:
$\mathsf{P}_t = \frac{|1-1|+|2-2|+|4-3|+|3-4|+|5-5|}{log(5)} \approx 2.86.$

The preorder traversal of a document's discourse dependency tree takes into consideration the organization of a document respect to the dependencies between sentences. Then, the purpose of Property 2 is to measure how much the organization of a document, captured through the preorder traversal, deviates from its sentential sequential order. Sentence order is highly related to the coherence of a document where the displaced order of sentences in a document makes it less coherent (Li and Hovy, 2014). Thus, intuitively, the less the value of Property 2 for a document is, the more coherent that document should be. Also, for a document with k sentences $\mathsf{P}_t \in [\frac{k-1}{log(k)}, \frac{(k^2-1)}{2log(k)}]$ if k is odd and $\mathsf{P}_t \in [\frac{k-1}{log(k)}, \frac{k^2}{2log(k)}]$ if k is even.

Property 3 (Parent-Child Distance) *This property, denoted as* P_c*, defines the normalized sum of positional distances between child nodes and their parents when they are considered in the original sequential order:*

$$\mathsf{P}_c = \frac{\sum_{\forall c,p \in \mathcal{T}} |c_{position} - p_{position}|}{log(k)} \tag{10}$$

where $c_{position}$ and $p_{position}$ denote the positions of a child node c and a parent node p, respectively, in the original sentential sequential order. For instance, in our running example, the parent node s_3 has $p_{position} = 3$ (i.e., it is the third sentence) and its child node s_5 has $c_{position} = 5$. Therefore, their parent-child distance is $|5-3| = 2$. Following a similar calculation for other parent-child pairs, we have $\mathsf{P}_c = \frac{1+2+2+2}{log(5)} \approx 10$.

Similar to Property 2, Property 3 pertains to the organization of a document and takes into consideration the deviation from sentential sequential order. Intuitively speaking, usually, we expect that a child node and its parent to be close to each other in the original sequential order. Consequently, the less value of this property is, the more coherent a document is likely to be. The range of Property 3 in a document containing k sentences is $\mathsf{P}_c \in [\frac{k-1}{log(k)}, \frac{k(k-1)}{2log(k)}]$.

5 Experiments

To verify the performance of the proposed framework HDSF, we conduct a set of experiments. We seek to answer the following research questions:

1. How does the proposed framework perform on fake news detection?
2. How do the defined structure-related properties describe the fake and real news documents?

In this section, we first describe the datasets followed by presenting the experimental settings. Afterward, we evaluate the performance of HDSF compared to several representative baselines. Finally, we present a structural analysis of the fake/real news documents.

5.1 Datasets

We utilize five available fake news datasets in this study. The first two datasets are collected by (Shu

---

[1] The subtrees are ordered based on *when* they are added as the child nodes of a parent node in Algorithm 1. That is why we can compute preorder traversal.

[2] We assume that the sentences in the sequential order are numbered incrementally from 1 to k.

et al., 2017) and include online articles whose veracities have been identified by experts in BuzzFeed[1] and PolitiFact[2]. For the next two datasets, we utilize two available online fake news datasets provided by *kaggle.com*[3] [4]. Finally, we include the dataset constructed and shared by McIntire[5]. Since the proposed framework HDSF is a general-purpose framework investigating discourse-level structures of fake/real news documents based on their textual contents, we do not restrict HDSF to a particular source of data and therefore combine all datasets. Similar to the previous work (Shu et al., 2017), we balance the dataset to avoid a trivial solution as well as ensuring a fair performance comparison. In total, we have 3360 fake and 3360 real documents.

5.2 Experimental Settings

First, we pre-process the documents by removing numbers, non-English characters, stop-words (e.g., ‘with’), and converting all characters to lower case. We randomly select 134 documents as the development set, (67 from each class) and 134 documents (67 from each class) as the test set. The remaining 6452 documents are used for training. The development set is used for tuning the hyper-parameters. We initialize the word embeddings from the Google news pre-trained word2vec embeddings (Mikolov et al., 2013)[6]. LeakyReLU (Xu et al., 2015) is used as the non-linear activation function and the number of hidden units in the BLSTM network is set to 100. Each simulation is run for 200 steps with a random mini-batch size of 40 documents. The learning rate starts at 0.01 with the decay rate of 0.9 after every 50 steps. We use the ADAM optimizer (Kingma and Ba, 2014) to optimize the parameters. The PyTorch package[7] is utilized for the implementation and the code and data are publicly available in `https://github.com/hamidkarimi/HDSF`.

5.3 Comparison Results

To answer the research question (1), we compare the performance of HDSF with the following representative baselines.

N-grams. In this baseline method, we extract and combine unigrams, bigrams, and trigrams features and use SVM (Support Vector Machines) (Scholkopf and Smola, 2001) for classification.

LIWC (Pennebaker et al., 2015). LIWC (Linguistic Inquiry and Word Count) offers a set of rich psycholinguistic features for a written document. We extract 94 features for each document and use SVM for classification.

RST (Rubin and Lukoianova, 2015). We extract a set of RST relations (Mann and Thompson, 1988) using the implementation of the method proposed by (Ji and Eisenstein, 2014). Then, we vectorize the relations and employ SVM for classification. This baseline takes into account the hierarchical structure of documents via RST.

BiGRNN-CNN (Ren and Zhang, 2016). A CNN (Convolutional Neural Network) is applied at the sentence-level on word embeddings and a BiGRNN (Bi-Directional Gated Recurrent Neural Network) extracts features from a sequence of extracted sentential features. This baseline takes into consideration a two-level sequential structure for a document.

LSTM[w+s]. In this baseline, we apply an LSTM network on a sequence of word embeddings belonging to a sentence and then apply another LSTM on a sequence of extracted sentential features. LSTM[w+s] also considers a two-level sequential structure for a document.

LSTM[s]. This method is similar to LSTM[w+s] except that the mean of word embeddings in a sentence is used instead of applying an LSTM network. LSTM[s] considers a single sequential structure for a document.

We use accuracy as the metric of performance evaluation given that the dataset is fully balanced. Table 1 shows the comparison results on the test set and we make the following observations:

- N-grams achieve a better performance than LIWC. In line with the previous study (Ott et al., 2011), this shows that for fake news detection, taking into account the context of a document as n-grams do is more effective than employing the existing pre-defined dictionaries as LIWC does.

---

[1] `https://www.buzzfeed.com`

[2] `http://www.politifact.com/`

[3] `https://www.kaggle.com/mrisdal/fake-news/data`

[4] `https://www.kaggle.com/jruvika/fake-news-detection`

[5] `https://github.com/GeorgeMcIntire/fake_real_news_dataset`

[6] Out-of-vocabulary words are initialized randomly.

[7] `https://pytorch.org/`

Method	Accuracy (%)
N-grams	72.37
LIWC	70.26
RST	67.68
BiGRNN-CNN	77.06
LSTM[w+s]	80.54
LSTM[s]	73.63
HDSF	**82.19**

Table 1: Comparison results

- Most of the time, methods wherein a document's structure is somehow taken into account outperform n-grams and LIWC. This observation shows that for fake news detection, the content's structure plays an important role.

- The poor performance of RST is because of the following reasons: a) using RST without an annotated corpus is not very effective, and b) RST relations are extracted using auxiliary tools optimized for other corpora which cannot be applied effectively to the fake news corpus in hand. Note that annotating RST for our corpus is extremely unscalable and time-consuming.

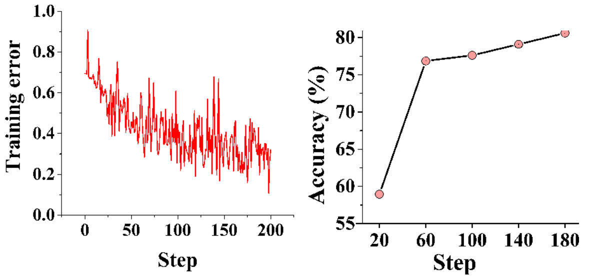
- The proposed framework HDSF significantly outperforms all other methods. This observation shows that hierarchical discourse-level representations are effectively rich for fake news prediction.

5.4 The Inspection of HDSF

To further verify the working of the HDSF framework, we inspect HDSF in more detail. Figure 4a demonstrates the training error during model optimization. As we can observe from this figure, the error is decreasing as the training process proceeds. Furthermore, Figure 4b demonstrates the accuracy on the development set during the training and it is monotonically increasing as the training goes on. Hence, based on these figures, we can ensure the framework is getting optimized and *learns* to classify fake news documents correctly.

5.5 Structural Analysis for Fake/Real Documents

In this section, we compute the average values of structure-related properties, presented in Section 4, for the fake/real news documents belonging to the test set. Figure 5 shows the results. We make the two key observations based on this figure:



(a) The training error (b) The accuracy on dev set

Figure 4: The Inspection of HDSF

- There is a significant difference in all three properties for fake news documents vs. real news documents. *This observation shows the fact that structures of fake news documents at the discourse-level are substantially different from those of real ones.*

- Noticeably, in all three properties, the real news documents show less value than the fake news documents. As described in Section 4, all three properties are closely connected to the coherence of a document. *Therefore, real news documents indicate more degree of coherence.*

6 Related Work

Content-based fake news detection has been the subject of many linguistic research endeavors. (DePaulo et al., 2003) investigated fake stories introduced insightful cues in fake stories and highlighted 'unusual' language in such stories. N-grams and Part-of-Speech (POS) tags are fundamental features of a text which have been utilized for fake news detection (Ahmed et al., 2017; Ott et al., 2013). Also, LIWC (Pennebaker et al., 2015) has been employed to investigate the role of individual words in a document for deception detection (Ott et al., 2011). Since POS tags, n-grams, and LIWC features are considered as 'shallow' and hand-crafted features, deep neural networks have been utilized for fake news detection where features are extracted automatically (Wang, 2017; Ren and Zhang, 2016; Volkova et al., 2017; Karimi et al., 2018). In this study, we also utilized an automated feature extraction instead of relying on hand-craft features.

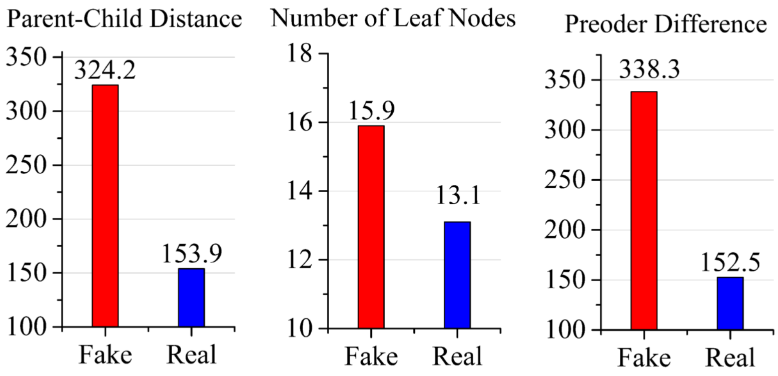


Figure 5: Average values of the proposed structure-related properties for fake and real news documents

Syntax-based approaches have been employed to take into account the hierarchical structure of textual documents for fake news detection (Feng et al., 2012; Pérez-Rosas and Mihalcea, 2015). One caveat of syntax-based approaches is their reliance on auxiliary parsing tools which might propagate error in later part of a developed model. Moreover, generating syntactic production rules in an automated manner is a complicated process.

Another way of incorporating structure is discourse-level parsing (Mann and Thompson, 1988; Li et al., 2014a; Ren and Zhang, 2016) which has seldom been explored for fake news detection. The noticeable exception is the approach developed by (Rubin and Lukoianova, 2015). They extracted a set of RST relations in fake and real documents and vectorized them using a Vector Space Model (VSM) method (Strehl and Ghosh, 2000). Unlike (Rubin and Lukoianova, 2015), in this work, we proposed an automated and data-driven discourse-level parsing approach which used neither any annotated corpus nor any external tool.

7 Conclusion and Future Work

In this work, we looked into fake news detection from a new perspective. We hypothesized that hierarchical discourse-level structure of news documents offers a discriminatory power for fake news detection. To investigate this hypothesis, we proposed a new framework HDSF, which can automatically extract discourse-level structures of real/fake news documents represented by dependency trees while does not rely on an annotated corpus. Moreover, we defined a set of insightful properties describing tree structures. Conducted experiments confirmed the power of our approach where it outperformed representative baselines. More importantly, we highlighted noticeable differences between structures of fake and real news documents. These differences also indicated less coherency in the fake news documents.

The new perspective pursued in this paper can be continued in several directions. First, we intend to define more advanced properties from the discourse dependency trees. Second, investigating the hierarchical structure at the word-level will be an exciting research inquiry. Finally, unsupervised discourse-level structure extraction of fake/real news documents is a worthwhile research topic.

Acknowledgments

This work is supported by the National Science Foundation (NSF) under grant numbers IIS-1714741, IIS-1715940, IIS-1845081, CNS-1815636, and a grant from Criteo Faculty Research Award. We would like to thank Mr. Tyler Derr and other members of the Data Science and Engineering (DSE) lab at Michigan State University[1] for their constructive comments.

References

Hadeer Ahmed, Issa Traore, and Sherif Saad. 2017. Detection of online fake news using n-gram analysis and machine learning techniques. In *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pages 127–138. Springer.

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

---

[1] `http://dse.cse.msu.edu/`

Joan Bachenko, Eileen Fitzpatrick, and Michael Schonwetter. 2008. Verification and implementation of language-based deception indicators in civil and criminal narratives. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 41–48. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bella M DePaulo, James J Lindsay, Brian E Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *Psychological bulletin*, 129(1):74.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 93–103.

Robert Hecht-Nielsen. 1992. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. 2018. Multi-source multi-class fake news detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014a. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014b. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 25–35.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association of Computational Linguistics*, 6:63–75.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on rst discourse parsing and evaluation. *Computational Linguistics*, (Just Accepted):1–54.

Myle Ott, Claire Cardie, and Jeffrey T Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 497–501.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

Verónica Pérez-Rosas and Rada Mihalcea. 2015. Experiments in open domain deception detection. In *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing, pages 1120–1125.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The penn discourse treebank 2.0 annotation manual.

Yafeng Ren and Yue Zhang. 2016. Deceptive opinion spam detection using neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 140–150.

Victoria L Rubin and Tatiana Lukoianova. 2015. Truth and deception at the rhetorical structure level. *Journal of the Association for Information Science and Technology*, 66(5):905–917.

Bernhard Scholkopf and Alexander J Smola. 2001. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Kai Shu, Suhang Wang, and Huan Liu. 2017. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*.

Angelika Storrer. 2002. Coherence in text and hypertext. *Document Design*, 3(2):156–168.

Alexander Strehl and Joydeep Ghosh. 2000. Impact of similarity measures on web-page clustering.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion

Mor Geva [*]
Tel Aviv University
morgeva@mail.tau.ac.il

Eric Malmi
Google AI
emalmi@google.com

Idan Szpektor
Google AI
szpektor@google.com

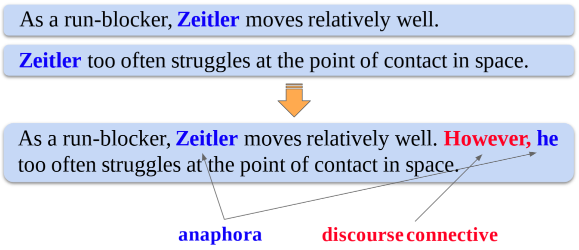
Jonathan Berant [†]
Tel Aviv University,
Allen Institute for AI
joberant@cs.tau.ac.il

Abstract

Sentence fusion is the task of joining several independent sentences into a single coherent text. Current datasets for sentence fusion are small and insufficient for training modern neural models. In this paper, we propose a method for automatically-generating fusion examples from raw text and present DiscoFuse, a large scale dataset for discourse-based sentence fusion. We author a set of rules for identifying a diverse set of discourse phenomena in raw text, and decomposing the text into two independent sentences. We apply our approach on two document collections: Wikipedia and Sports articles, yielding 60 million fusion examples annotated with discourse information required to reconstruct the fused text. We develop a sequence-to-sequence model on DiscoFuse and thoroughly analyze its strengths and weaknesses with respect to the various discourse phenomena, using both automatic as well as human evaluation. Finally, we conduct transfer learning experiments with WebSplit, a recent dataset for text simplification. We show that pretraining on DiscoFuse substantially improves performance on WebSplit when viewed as a sentence fusion task.

1 Introduction

Sentence fusion is the task of combining several independent sentences into a single coherent text (Barzilay and McKeown, 2005). Sentence fusion is important in many NLP applications, including retrieval-based dialogue (Song et al., 2018; Yan and Zhao, 2018), text summarization (Barzilay and McKeown, 2005; Bing et al., 2015) and question answering (Li et al., 2018; Marsi and Krahmer, 2005). Such systems retrieve multiple sentences from different sources, documents or paragraphs, and use them to construct a coherent text.


As a run-blocker, **Zeitler** moves relatively well.

Zeitler too often struggles at the point of contact in space.

As a run-blocker, **Zeitler** moves relatively well. **However, he** too often struggles at the point of contact in space.

Figure 1: Example for two independent sentences, and their fusion. The modifications applied are pronominalization (blue) and connective insertion (red).

Sentence fusion is challenging because it requires understanding the discourse semantics between the input sentences. Consider the example in Figure 1: a coherent fusion of the sentences requires understanding that the second sentence *contrasts* the first one, in order to insert the discourse connective *"However"*. In addition, the gender and syntactic role of the entity *"Zeitler"* needs to be inferred to insert the pronoun *"he"*.

Prior work on sentence fusion (Barzilay and McKeown, 2005; Turner and Charniak, 2005; Filippova, 2010; Elsner and Santhanam, 2011; Thadani and McKeown, 2013; Bing et al., 2015; Chali et al., 2017) utilized very small amounts of labeled data, which are insufficient to train modern neural models. In this work, we propose a method for automatically generating sentence fusion examples at scale from raw text corpora.

To this end, we go over sentences and contiguous pairs of sentences in a corpus, and apply a set of manually-constructed rules, which identify the occurrence of prevalent fusion operations. The rules specify how to modify the sentences such that they are "unfused" into two independent sentences. E.g., in Figure 1 one rule will delete the discourse connective *"However"*, and another will replace the pronoun *"he"* with the named entity *"Zeitler"*.

---

[*] Work done during internship at Google AI.

[†] Work done at Google AI.

Proceedings of NAACL-HLT 2019, pages 3443–3455
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

In the generated examples, the original fused text becomes the target, and the unfused sentences (generated by rules) are the input. Importantly, sentence fusion models trained on our data cannot simply learn to invert rule application, because information is lost and can be recovered only by understanding the text semantics . As mentioned, learning to insert *"However"* in Figure 1 requires inferring that the sentences contrast. We cover a wide range of fusion phenomena such as inserting discourse connectives in various positions of the sentences, anaphora and cataphora identification, and sentence merging through coordination, relative clauses and apposition.

We applied our method on two large document collections, Wikipedia and sports articles from the Web, resulting in two datasets of 16 million and 44 million examples respectively. We call the combined dataset DISCOFUSE. We extensively analyze the quality of our dataset with crowdsourcing, and find that workers understand the text after splitting in 85% of the cases, and the other 15% are due to either the original text being unclear or errors in rule application.

We trained a state-of-the-art sequence-to-sequence model (Vaswani et al., 2017) and analyzed the fusion phenomena in which the model struggles. We found that the model succeeds in fusing sentences through structural constructions such as apposition or relative clauses, but performs badly when fusion involves inserting a particular discourse connective, or selecting pronominals.

Last, we performed transfer learning by training on DISCOFUSE and then fine-tuning on a smaller dataset from a different distribution. To this end, we utilize WEBSPLIT, a recent dataset for sentence splitting (Narayan et al., 2017; Aharoni and Goldberg, 2018), viewing WEBSPLIT as a sentence fusion task. We found that pre-training on DISCOFUSE substantially improves the performance of a fusion model in this setup.

To conclude, our contributions are:

1. DISCOFUSE: a dataset of 60 million sentence fusion examples from two different corpora.
2. A method for automatically generating sentence fusion examples from raw text.
3. Automatic and human evaluation of the Transformer model on the fusion task.
4. A transfer learning setting in which model performance improves when pre-trained with DISCOFUSE.

The DISCOFUSE dataset is publicly available at: `https://discofuse.page.link/data`.

2 Background

Existing fusion datasets are small, which is perhaps why only few works have explored the application of supervised models to sentence fusion (Elsner and Santhanam, 2011; Thadani and McKeown, 2013). McKeown et al. (2010) introduced a human-generated corpus of 3,000 examples. Elsner and Santhanam (2011) extracted around 300 fusion examples from pre- and post-editing news articles. Thadani and McKeown (2013) constructed 1,858 examples from summarization tasks. Such datasets are too small to train modern data-hungry neural models.

Related to sentence fusion is its "inverse" task of sentence splitting. Collados (2013) automatically constructed a Spanish simplification dataset by splitting single sentences into several simpler ones. Recently, two larger datasets for text splitting were released (Botha et al., 2018; Narayan et al., 2017; Aharoni and Goldberg, 2018). However, using these datasets for the "mirror" task of sentence fusion is problematic. First, sentence splitting often involves removing content from the original sentence for simplification, and this content is impossible to recover in the fusion direction. Second, these datasets do not focus on discourse and thus prominent discourse phenomena may be missed. Last, our new dataset is more than an order of magnitude larger than the above sentence splitting datasets.

Another related line of recent work focused on predicting discourse connectives between sentences and automatically generating examples from raw text (Liu et al., 2016; Malmi et al., 2018). We substantially expand over those works by handling more diverse linguistic phenomena, such as connectives in single sentences, generating anaphora and cataphora constructions, relative clauses, coordination and more, which are all represented in a single dataset. Moreover, our dataset is 20x larger compared to prior work, allowing us to examine in depth long-tail scenarios.

3 The DISCOFUSE Dataset

We next describe our process for building DISCOFUSE, which contains 60 million sentence fusion examples from two different document collections: Wikipedia and Web articles about sports.

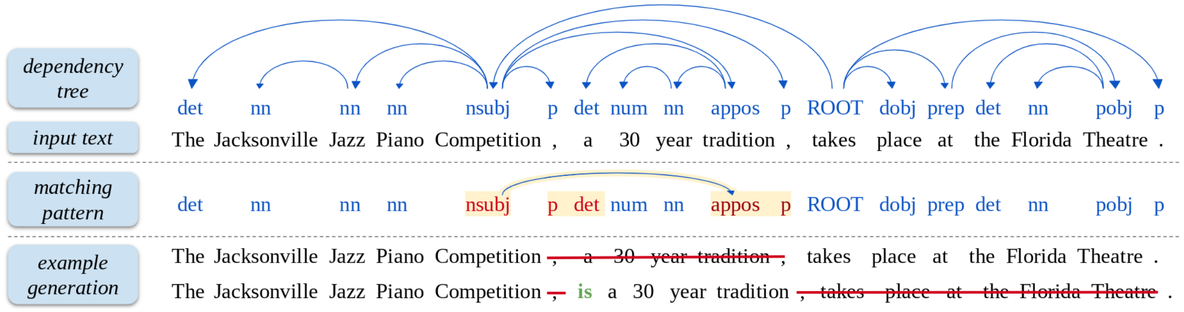


Figure 2: Example generation rule for apposition. Given an input text and its dependency tree, we check for a match with the apposition pattern. We then use the dependency tree to split the sentence and create a new example.

3.1 Example Generation

DiscoFuse contains union-fusion examples, *i.e.* fusing sentences without loss of content (Marsi and Krahmer, 2005). To automatically extract examples, we manually crafted a list of text splitting rules. Our rule-set covers 9 fusion phenomena, including handling discourse connectives, coordination and relative clauses, and entity resolution for anaphora and cataphora constructions. For entity resolution, both anaphoric pronouns (*"she"*, *"they"*, *"his"*) and anaphoric nominals (*"the team"*, *"the man"*) are considered, based on the output of a coreference system. The covered phenomena are summarized in Table 1 and a detailed description is given in Appendix A.1.

Given a text t consisting of one or two consecutive sentences, each of our rules addresses a specific discourse phenomenon and has two parts: (a) conditions for matching the phenomenon in t, and (b) operations over a dependency tree annotated with coreference resolution. Applying the operations generates a fusion example $(x = (s_1, s_2), t)$, in which (s_1, s_2) are two independent sentences originating from t, but stripped from the discourse phenomenon that tied them in t.

Figure 2 gives an example of a rule for the apposition structure. The rule is applied to the sentence *"The Jacksonville Jazz Piano Competition, a 30 year tradition, takes place at the Florida Theatre"*. First, the input is matched to the rule's condition. In this case, the condition is a single clause surrounded by two commas, which has a determiner as its first token and includes an apposition with incoming edge from a preceding token to the clause. Once matched, an example is generated. For this rule, the first sentence is created by removing the apposition clause, and the second sentence by removing the part after the clause and inserting the appropriate *"be"* verb (*"is"*). Generation examples for all 9 rule types are provided in Table 1.

As explained in Section 1, solving sentence fusion involves more than just reverse-engineering the generation rules. The model needs to decide whether to insert a discourse connective with the right semantics, whether to merge the input sentences, and what syntactic construction (relative clause, coordination, apposition) is most appropriate in the given context.

Last, often several discourse phenomena occur in a single text t. Thus, we allow combining anaphora rules with one of the following rule types: discourse connective, inner connective and sentence coordination, which cover frequent combinations in our texts.

3.2 Building the DiscoFuse Dataset

To create DiscoFuse we retrieved the latest Wikipedia release and crawled the Web for several million sports articles. Documents were annotated with dependency trees and coreference resolution using Google Cloud Natural Language.[1]

We considered each sentence and pair of consecutive sentences in each document as candidates, applying the example generation process described in Section 3.1. Additionally, we added as examples sentence pairs from the original corpus that did not match any rule, that is $(s_1, s_2) = t$, so that a trained model would also learn when not to change the input. We filtered out examples with sentence length $\leq$ 6 tokens, and examples with non-ASCII characters. This process resulted in $44,177,443$ sports examples and $16,642,323$ Wikipedia examples. We randomly split these examples into 98% train, 1% dev, and 1% test sets,

---

[1] https://cloud.google.com/natural-language/

Phenomenon	**Example**
Discourse connective	(A) Hebden Bridge is a popular place to live .
	(B) **However ,** space is limited due to the steep valleys and lack of flat land .
	(a) Hebden Bridge is a popular place to live .
	(b) Space is limited due to the steep valleys and lack of flat land .
Anaphora	(A) **Rider** entered the weekend averaging 23.0 points , good for 10th in the league .
	(B) **He** said those numbers mean little because of the Hawks ' 11 - 18 record.
	(a) **Rider** entered the weekend averaging 23.0 points , good for 10th in the league .
	(b) **Rider** said those numbers mean little because of the Hawks ' 11 - 18 record.
Forward connective	(A) **Although** the friendship somewhat healed years later , it was a devastating loss to Croly .
	(a) The friendship somewhat healed years later .
	(b) It was a devastating loss to Croly .
Inner connective	(A) Open workouts are held every Sunday **unless** the gym is closed for a holiday or other special events .
	(a) Open workouts are held every Sunday .
	(b) The gym is closed for a holiday or other special events .
Cataphora	(A) **Stating** that the proponents were unlikely to succeed in this appeal , **Walker** rejected the stay request on October 23 .
	(a) **Walker stated** that the proponents were unlikely to succeed in this appeal .
	(b) **Walker** rejected the stay request on October 23 .
Sentence coordination	(A) The time of the autumn floods came **, and** the hundred streams poured into the Yellow River .
	(a) The time of the autumn floods came .
	(b) The hundred streams poured into the Yellow River .
Verb phrase coordination	(A) The Sharks **started** the year 0 - 4 **, yet recovered** to claim sixth spot .
	(a) The Sharks **started** the year 0 - 4 .
	(b) The Sharks **recovered** to claim sixth spot .
Relative clause	(A) Kubler **, who** retired from cycling in 1957 **,** remained a revered figure in the wealthy alpine nation .
	(a) Kubler remained a revered figure in the wealthy alpine nation .
	(b) Kubler retired from cycling in 1957 .
Apposition	(A) The frigidarium **,** the last stop in the bathhouse **,** was where guests would cool off in a large pool .
	(a) The frigidarium was where guests would cool off in a large pool .
	(b) The frigidarium is the last stop in the bathhouse .

Table 1: Generated fusion examples for different phenomena. The input text is marked in uppercase blue, and the generated sentence pair is marked in lowercase red. We show in **boldface** parts that allow us to detect the target phenomenon.

making sure that each document contributes examples to only one of the split sets.

Like prior work (Malmi et al., 2018), we observed a skewed distribution of discourse phenomena in the data. Specifically, examples with anaphora or the connectives *"and"* and *"but"* constitute 99.7% of Sports and 59% of Wikipedia examples. Such a skewed distribution is likely to bias models and will fail to elucidate the ability of models to capture a wide range of linguistic phenomena. Therefore, we constructed a version of DISCOFUSE by down-sampling examples containing *"and"* and *"but"* or anaphora. The down-sampled dataset contains 12,080,513 Sports examples and 4,581,352 Wikipedia examples.

The resulting distributions of discourse types and most common connectives in the two parts of DISCOFUSE are provided in Appendix A.2. We will release both the original and the down-sampled versions of DISCOFUSE.

Rater Selection	SPORTS (%)	WIKIPEDIA (%)
Yes	83.4	86.0
No majority	10.9	8.9
No	5.7	5.1

Table 2: Rater evaluation understandability of the text after splitting. For each example, the majority of 5 raters was taken as the final rater selection.

4 DISCOFUSE Quality Evaluation

To assess the quality of the generated fusion examples in DISCOFUSE, we randomly selected 500 examples from each of the development sets of the Wikipedia and the Sports parts. We then conducted a crowdsourcing experiment in which each example was rated by 5 proficient English speakers, limiting each rater to at most 6 items. Each rater was presented with the two independent sentences in the example and was asked to indicate whether the text is understandable. If the rater answered "yes", she was then asked to characterize the relation between the sentences and how she would fuse them. We next detail the results.

Reason	Example
Original text unclear	(A) UPDATE: Peat falls because footwork and quickness. (a) UPDATE: Peat falls. (b) Footwork and quickness.
Missing context	(A) We were right on the heels of Spurs, although Everton were closing in. (a) We were right on the heels of Spurs. (b) Everton were closing in.
Bad rule generation	(A) He told reporters after the game his reaction was because he missed a wide-open Randall Cobb in the end zone. (a) He told reporters after the game his reaction was. (b) He missed a wide-open Randall Cobb in the end zone.

Table 3: Examples for three possible reasons for not understanding the text. In each example, (A) is the original text and (a) and (b) are the two sentences generated by our rules.

Error	Generated sentence
Extra comma	The space behind the fence in right field is blocked off **,** .
Missing determiner	*My* internist sent me for a mammogram and sonogram .
Bad pronoun replacement	**Lions** have a 3 - 1 record overall against Lions .

Table 4: Examples of grammatical errors introduced by our rules. The **red boldface** text was incorrectly inserted and the *blue italic* text was incorrectly removed.

4.1 Example Text Clarity

Raters were asked whether they can understand the text after the example is split. Table 2 summarizes this evaluation. Most examples were marked as understandable by the raters ("yes") – 86% of Wikipedia examples and 83.4% of Sports examples. The rest either had no majority of rater votes or were marked as not understandable.

To shed light on the possible reasons for obscurity, we analyzed 70 random examples that were not marked as understandable by the majority of raters. In 29 examples (41%) the original text was unclear and for 17 examples a broader context was needed (24%). In the remaining 24 examples (34%), our rules generated sentences with grammatical or semantic errors. Examples for these cases are in Table 3.

Additionally, we analyzed 100 random examples for grammatical errors, and found that our rules did not introduce any errors in 79 examples. For 15 examples, the errors did not modify the meaning of the text nor caused semantic errors. The detected grammatical errors include extra commas, missing determiners and bad pronoun replacements, and are demonstrated in Table 4.

4.2 Fusion Evaluation

Next, we evaluated agreement on the fusion task for the 847 examples marked as understandable in Section 4.1. Because there are many ways in which sentences can be fused, one cannot expect raters to produce the original text t verbatim. Instead, we analyzed three central decisions and estimated whether people agree on those: (a) whether to merge the two sentences into a single one or keep them separate; (b) whether there are entities in the text that should be replaced with nominal or pronominal anaphors or cataphors; and (c) which discourse connective to add (if any).

For the last question, we presented raters with one connective from each of the four coarse-grained senses for discourse connectives defined by the PDTB (Prasad et al., 2008): *comparison*, *expansion*, *contingency* and *temporal*, as well as a *no-connective* option. If the original text in the example includes a connective, we provided it as one of the options.

We observed a strong bias among raters towards refraining from performing any changes. E.g., while only 38% of the examples did not contain a connective in t, the raters chose not to add a connective in 69.2% of the cases. Similarly, only in 29.1% of the examples the two sentences were not merged into a single one, while the raters chose not to merge in 53.1% of the examples. Similar behavior was also observed by Malmi et al. (2018) and Rohde et al. (2016).

We further looked at the agreement between the rater majority and the 'gold' fusion decision. This analysis is shown in Table 5. Agreement on merging the input sentences into one is almost random (52%), since usually both options are valid. Consensus on whether to add an anaphor is higher, but not very high (63%), especially in sentences when the anaphor in t is a nominal rather than a pronoun. Finally, there is higher agreement on selecting the connective category (57%), for which the random baseline is 20%.

As mentioned, raters tend to keep the sentences unchanged. But in cases where raters agree to add a connective, agreement figures increase substantially. Specifically, when it is clear that a connective is needed, there is also high agreement for picking the right one (76%), for deciding whether

Decision Type	All Examples (%)	Examples with connectives (%)
Single / pair	52.0	70.1
Anaphor	63.4	70.1
Connective category	57.0	76.4
# Examples	847	271

Table 5: Average agreement for each fusion decision between the gold annotation and rater majority on examples marked as understandable by the raters. The right column considers only examples in which both the 'gold' and rater majority agreed that a connective should be added.

Sports	Full		Sampled	
	SARI	*Exact*	*SARI*	*Exact*
DfSport	**81.9**	**42.3%**	**83.9**	**50.6%**
DfWiki	77.8	31.7%	80.1	40.1%
DfS+W	80.7	38.3%	82.9	47.0%
Copy	40.0	1.1%	40.4	3.8%

Wikipedia	Full		Sampled	
	SARI	*Exact*	*SARI*	*Exact*
DfSport	80.0	41.5%	80.0	41.9%
DfWiki	**83.1**	**47.6%**	**84.5**	**51.1%**
DfS+W	82.8	46.7%	83.7	49.2%
Copy	40.3	1.0%	39.6	2.1%

Table 6: *Exact* and *SARI* scores of DfSport, DfWiki, DfS+W and Copy, on the test sets of DiscoFuse before (*Full*) and after down-sampling (*Sampled*).

Discourse phenomena	**DfSport**		**DfWiki**	
	Exact	*SARI*	*Exact*	*SARI*
Apposition	94.8%	99.3	94.7%	99.6
Relative clause	84.3%	95.3	76.9%	92.6
Cataphora	79.9%	92.8	84.2%	95.8
Verb phrase coordination	58.4%	88.0	58.7%	88.8
None (control)	55.3%	73.2	54.2%	72.7
Anaphora	52.1%	83.7	47.7%	81.5
Inner connective	49.9%	83.9	51.6%	85.5
Sentence coordination	35.6%	80.9	31.7%	79.4
Inner connective + anaphora	32.6%	82.5	37.2%	83.7
Forward connective	27.5%	80.2	34.6%	82.8
Sentence coordination + anaphora	20.5%	80.9	16.3%	78.3
Discourse connective	14.2%	65.6	29.1%	73.4
Discourse connective + anaphora	2.5%	73.0	8.0%	72.1

Table 7: In-domain evaluation with breakdown by discourse phenomena. Performance of DfSport and DfWiki on the sports and Wikipedia development sets.

to add an anaphor (70%), and for deciding whether to merge the sentences or not (70%).

5 Supervised Neural Fusion Models

Using DiscoFuse, we trained a Transformer seq2seq model (Vaswani et al., 2017) that reads the input sentence pair and generates a fused text. We report model performance on the test-set using automatic metrics as well as human evaluation. We also provide detailed analysis of the different phenomena captured by this model.

5.1 Experimental Settings

We tokenized all texts using byte-pair-encoding (Sennrich et al., 2015) and compared the following three Transformer models :

- **DfSport** - trained on the sports portion of DiscoFuse after down-sampling.
- **DfWiki** - trained on the Wikipedia portion of DiscoFuse after down-sampling.
- **DfS+W** - trained on a 50%-50% mixture of the sports and Wikipedia portions of DiscoFuse after down-sampling.

All models share the same network architecture, based on the best practices discussed by Popel and Bojar (2018). We tuned parameters to select the best learning and dropout rates for each model with respect to the Exact Match objective (described in Section 5.2). Network architecture and hyper-parameters are in Appendix A.3. As a baseline, we also tested a model called Copy, which simply concatenates the two input sentences.

5.2 Automatic Evaluation Results

We evaluated model performance using two automatic metrics. The first is Exact Match (*Exact*) to see how often the model generates the exact same text as the gold fusion. The second is *SARI* (Xu et al., 2016), which computes the set of added, removed, and kept n-grams in the model output, comparing the output both with the gold text and the input text. Then it computes the F_1 scores for these three sets and averages the scores. We compute *SARI* on up to 4-grams, as in Xu et al. (2016). We refrained from using metrics like BLEU because in fusion there is large overlap between the input sentences and their fused version, and such metrics do not capture well fine-grained differences of only a single word.

We note that our definition of *SARI*[2] slightly differs from the one given by Xu et al. (2016) in

---

[2] Our *SARI* implementation is available at: `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/sari_hook.py`

two aspects: (i) We define $\frac{0}{0} = 1$ when computing precision and recall, otherwise *SARI* could be less than 1 even if the output matches the gold text exactly. (ii) Instead of considering only the precision of deleted n-grams, we use F_1 for all three sets. Otherwise, *SARI* will give high scores to models that merely copy everything in the input, without even trying to infer what to delete.

Table 6 summarizes the results. When training and testing on the same domain, either Sports or Wikipedia, *SARI* score is a little above 80 points for the full dataset. Yet *Exact* is not high, around 42% for Sports and 47% for Wikipedia, showing that in the majority of the examples the model's fusion differs from the gold. Tested on the down-sampled test-set, performance increases significantly for *Exact*, especially on Sports, where discourse phenomena is more skewed.

We next turn to cross-domain evaluation. When applying a model trained on one domain to the other domain performance drops. This shows that the discourse phenomena distribution differs between the domains, indicating that transfer learning is not trivial even with these large datasets. This is especially evident when applying DFWIKI to Sports, where *Exact* falls from 42% to 32% on the full dataset and from 50% to 40% on the down-sampled one. Interestingly, when learning on the mixed training set, performance on both domains is close to in-domain performance, showing that the model has the capacity to handle both domains.

Finally, we take advantage of the provided annotation of the different discourse phenomena within each example in DISCOFUSE. We conducted a detailed analysis of in-domain model performance by discourse type, presented in Table 7. Results show that structural discourse types, such as apposition and relative clause, are easier to learn with both high exact match and *SARI* scores. While differences with respect to *SARI* scores are not large between phenomena, exact match varies more. Anaphora and verb phrase coordination are more challenging, but still require matching of the same noun (the named entity or the subject). On the other hand, discourse types that involve connective prediction, such as sentence coordination and discourse connective, require semantic understanding, and performance is significantly lower. In addition, when two discourse types are required for fusion, performance drops dramatically.

		# Examples	Detection
SPORTS	output = gold	525	50%
	output != gold	475	65%
	total	1000	**57%**
WIKIPEDIA	output = gold	528	50%
	output != gold	472	61%
	total	1000	**55%**

Table 8: Human detection (*Detection*) percentage for DFSPORT and DFWIKI on 1000 samples from each of the Sports and Wikipedia development sets. We report *Detection* for cases when model output differed from the gold, and cases when they were identical.

5.3 Human Evaluation Results

As our second experiment we employed crowd-sourcing to test how distinguishable the fusion model outputs are from the gold fused texts. Concretely, we present raters an independent sentence pair from DISCOFUSE and two fused versions - the gold version and one generated by a model. Raters were asked to detect the gold version. For each example, we took the majority of 5 raters as the final choice. This experiment mitigates the difficulties of automatic text generation evaluation, where many outputs are valid for a single input.

We sampled 1000 random examples from each development set of the two domains and applied the in-domain model to both. The raters were presented only with examples where the model output was different from the gold fusion, and assumed 50% detection accuracy otherwise.

Table 8 depicts the results. Out of cases when model output differed from the gold, raters were able to identify the human version in 65% of Sports examples and 61% of Wikipedia examples. Looking at the entire set, humans were able to identify the human version in 57% (Sports) and 55% (Wikipedia) of the cases. This shows that our Transformer model, applied over a dataset of millions of examples, is able to learn good fusions in general. Nevertheless, models are still far from perfect – human accuracy is clearly better than random and this improvement is statistically significant at a level of $p < 10^{-5}$ for Sports and $p < 10^{-3}$ for Wikipedia.

5.4 Alignment-based Analysis

We next present an analysis of the types of errors our models produce. To this end, we sampled 40K examples of DFSPORT and DFWIKI outputs on Sports and Wikipedia development sets. We then automatically aligned predicted sequences to gold

Connective	DfSport accuracy	DfWiki accuracy	Top 3 connectives
and	50.9	53.7	and, but, ⟨other⟩
but	42.8	43.7	but, ⟨other⟩, and
because	61.5	60.7	because, ⟨other⟩, but
although	35.1	33.2	although, ⟨other⟩, but
so	50.6	50.2	so, but, and
or	70.5	72.1	or, and, ⟨other⟩
however	28.3	26.7	⟨other⟩, however, but
while	70.1	70.6	while, ⟨other⟩, but
so that	64.3	63.0	so that, ⟨other⟩, because
unless	68.9	67.0	unless, because, ⟨other⟩
for example	26.9	28.1	⟨other⟩, for example, however

Table 9: Alignment-based connective prediction accuracy for the most common connectives. When a model did not add a connective, the token ⟨other⟩ is used.

sequences and looked at the differences between aligned words. The trained models successfully learned to copy most of the input text, and thus errors due to alignment problems are rare.

We start by considering the semantic relation between the input sentences. Table 9 displays model accuracy in predicting the most common connectives in DiscoFuse, as well as the top connectives predicted in this slot. We observe that when the model predicts a wrong connective, that connective is often reasonable, e.g., predicting *"but"* instead of *"and"* or *"however"*. Moreover, a second source of error is not adding a connective at all. It is also clear that some connectives, like *"however"*, *"although"* and *"for example"*, are harder to learn.

We also analyzed the models' ability to correctly infer pronoun anaphors including gender, possessive and plurality. Figure 3 shows the pronoun confusion matrix for DfWiki,[3] where lines refer to gold pronouns and columns to the generated pronoun in the same position. The clear diagonal shows that in most cases, the model successfully outputs the correct pronoun. However, the ⟨other⟩ column indicates that occasionally the model tends not to replace the entity in the input with a pronoun anaphor. In addition, the model seems to struggle with possession and plural 3rd person (*"it"*, *"its"*, *"they"*, *"their"*, *"theirs"*).

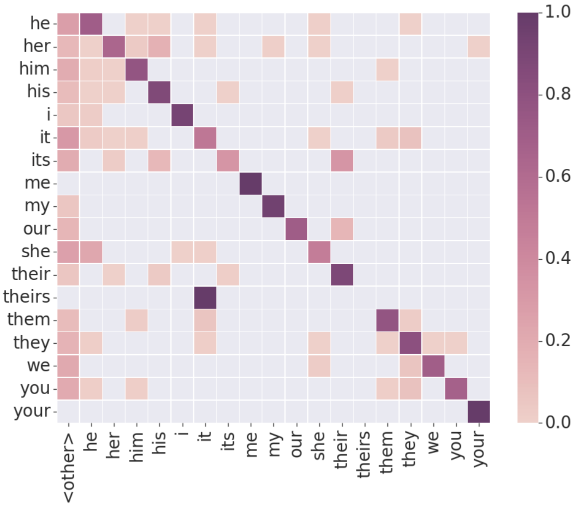


Figure 3: DfWiki outputs versus the gold pronouns. Rows refer to gold pronouns and columns refer to aligned model outputs at the gold pronoun position. Values in each row are normalized to 1. Column ⟨other⟩ refers to model outputs that are not pronouns.

6 Transfer Learning Experiment

With the DiscoFuse approach we can collect a large amount of examples automatically. Still, these examples only reflect the manual rules that identify discourse phenomena. We wanted to see if DiscoFuse covers enough cases such that a trained model would be helpful for testing on fusion datasets generated by different approaches.

6.1 Experimental Settings

In this experiment, we looked at the recently released WebSplit dataset 1.0 (Narayan et al., 2017). It consists of examples $(t, \{s_i\}_{i=1}^{n})$, where t is a sentence that verbalizes the same set of RDF triples as $\{s_i\}_{i=1}^{n}$. We note that WebSplit was originally developed for sentence splitting, from t to $\{s_i\}_{i=1}^{n}$, but here we view its examples for the reverse fusion task: from $\{s_i\}_{i=1}^{n}$ to t. We only considered examples where $\{s_i\}_{i=1}^{n}$ corresponds to exactly two simpler sentences ($n = 2$). This leaves us with 135K training, 8K validation, and 8K test samples.

We tokenized the data using byte-pair-encoding (Sennrich et al., 2015) and compared three models: (i) The Copy baseline that concatenates the two input sentences, (ii) a model trained on WebSplit alone, and (iii) a model pre-trained on DfWiki and fine-tuned on WebSplit.

For the last two models, we use the CopyNet architecture (Gu et al., 2016), which is similar

---

[3] Results for DfSport are very similar.

Training data	*SARI*	Keep	Add	Delete
COPY	18.1	52.9	0.5	0.9
WEBSPLIT	40.5	44.6	7.8	**69.3**
DFWIKI + WEBSPLIT	**44.2**	**54.8**	**10.4**	67.5

Table 10: Fusion results on WEBSPLIT, measured by *SARI* and the F1 scores that compose it.

to state-of-the-art models for the splitting task on WEBSPLIT (Narayan et al., 2017; Botha et al., 2018). While the Transformer outperformed this model on our main experiments, here it overfit on the small training set of WEBSPLIT. The training details are provided in Appendix A.3.

6.2 Results

Table 10 shows the results of the experiment. Similarly to Section 5, we measured the model performance using *SARI*. Pre-training with DFWIKI improves *SARI* score by 9% compared to using WEBSPLIT alone. In particular, the F1 of the 'kept' and 'added' n-grams is significantly higher, by 23% and 33% respectively. Specifically, 'added' tokens refer also to correctly choosing discourse connectives, to which the large-scale examples in DISCOFUSE were likely helpful.

We note that even with pre-training, the SARI 'add' score is only 10.4. This is probably due to the large amount of paraphrasing done in WEBSPLIT, which makes it problematic for fusion evaluation (see also Section 2). For example:

> **Sentence 1:** *Bolt , a comic character AKA Larry Bolatinsky , was created by Paris Cullins and Ernie Colon .*
>
> **Sentence 2:** *Paris Cullins is a United States national .*
>
> **Gold:** *Larry Bolatinsky is the alternative name for the comic book character Bolt , which was created by Ernie Colon and the American Paris Cullins .*

Correctly inferring the added terms (shown in red) requires paraphrasing knowledge that is outside the scope of DISCOFUSE.

7 Conclusions

We presented DISCOFUSE, a large-scale dataset for sentence fusion that was generated by applying a rule-based method. It contains millions of examples from two domains, annotated with multiple discourse phenomena.

We used DISCOFUSE to build supervised neural models for sentence fusion and conducted fine-grained analyses of the results. Currently, our models fuse only two sentences together. We would like to expand them to more input sentences in future work.

We also demonstrated DISCOFUSE's usefulness in a transfer learning setup on a different fusion test-set, hoping it would facilitate research on text fusion in data-scarce domains.

Acknowledgments

We thank Marta Recasens, Jan Botha, and Sascha Rothe from Google AI for helpful discussions. This research was supported by the Yandex Initiative in Machine Learning and by The Israel Science Foundation grant 942/16. This work was completed in partial fulfillment for the Ph.D degree of the first author.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and stronger baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597. Association for Computational Linguistics.

Jan A Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from wikipedia edit history. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yllias Chali, Moin Tanvee, and Mir Tafseer Nayeem. 2017. Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 418–424.

José Camacho Collados. 2013. Splitting complex sentences for natural language processing applications: Building a simplified spanish corpus. *Procedia-Social and Behavioral Sciences*, 95:464–472.

Micha Elsner and Deepak Santhanam. 2011. Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 54–63. Association for Computational Linguistics.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics.

J. Gu, Z. Lu, H. Li, and V. O. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*.

Yutong Li, Nicholas Gekakis, Qiuze Wu, Boyue Li, Khyathi Chandu, and Eric Nyberg. 2018. Extraction meets abstraction: Ideal answer generation for biomedical questions. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 57–65.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*, pages 2750–2756.

Eric Malmi, Daniele Pighin, Sebastian Krause, and Mikhail Kozhevnikov. 2018. Automatic Prediction of Discourse Connectives. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320. Association for Computational Linguistics.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. K. Joshi, and B. L. Webber. 2008. The Penn discourse treebank 2.0. In *LREC*.

Hannah Rohde, Anna Dickinson, Nathan Schneider, Christopher NL Clark, Annie Louis, and Bonnie Webber. 2016. Filling in the blanks in understanding discourse adverbials: Consistency, conflict, and context-dependence in a crowdsourced elicitation task. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 49–58.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Yiping Song, Rui Yan, Cheng-Te Li, Jian-Yun Nie, Ming Zhang, and Dongyan Zhao. 2018. An ensemble of retrieval-based and generation-based human-computer conversation systems. In *Proceedings of IJCAI-ECAI, 2018*.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1410–1418.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Rui Yan and Dongyan Zhao. 2018. Coupled context modeling for deep chit-chat: towards conversations between human and computer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2574–2583. ACM.

A Supplemental Material

A.1 Generation Rules

In this section we provide technical details of the generation rules used to create DiscoFuse. For the sake of clarity, we provide a simplified version of the rules, that does not include edge cases and minor implementation details. The discourse connectives we considered in the rules were selected from the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) and are listed in Table 12.

Given an input text, it is encoded with 3 lists: Z is the token list, Z_t is a list of POS tags, Z_l is a list of dependency labels (see Table 11). In addition, all entities mentioned in the text are extracted and stored such that for two token lists Z, R, the set $m(Z, R)$ holds all the mention pairs of the same entity in the two lists. Each rule is designed for a specific discourse phenomenon and contains two parts. First, a set of conditions is applied to the input lists to detect whether the phenomenon occurs in the text. If a discourse pattern has been identified, a short sequence of simple operations is applied to the input, yielding a new sentence pair. Table 13 summarizes the operations in use, which allow insertion and deletion of tokens and splitting of the input text.

Table 14 provides the technicalities of each rule, i.e. the detection conditions of the discourse structure, and the sequence of operations for generating a new sentence pair from it. A detailed example for two-rule execution process is given in Table 15.

As mentioned, the rules are simplified for clarity. However, we note two special cases where morphological modifications are required to produce text without grammatical errors. First, in some cases of forward connective and cataphora, the tense change of a verb is required when splitting the input sentence. For instance, in the cataphora example in Table 1, we change the verb *“stating”* to have a past tense – *“stated”*. Likewise, occasionally a *“be”* verb needs to be inserted when splitting a single sentence, as demonstrated in Figure 2. In our rules, we choose which *“be”* verb to insert based on the tense and perspective of the rest of the sentence.

A.2 DiscoFuse Data Distribution

Figure 4 and Table 16 show the distributions of discourse types and most common connectives in the two parts of DiscoFuse.

Analyzing the dataset reveals significant differences in discourse phenomena between the two types of documents (Figure 4). E.g., coordination is very common in Wikipedia while anaphora is dominant in Sports. Likewise, the distribution of discourse connectives is quite different (Table 16).

A.3 Neural Models Parameters

The models DfSport, DfWiki, DfS+W share the same Transformer network architecture, that was originally proposed by Vaswani et al. (2017). During training, we split the samples to buckets

Notation	Definition
Token list Z	A list of tokens $\{z^{(1)}, \ldots, z^{(\lvert Z\rvert)}\}$
Z_t	The list of POS tags of Z, where $Z_t^{(i)}$ is the tag of $z^{(i)}$ for every $i = 1, \ldots, \lvert Z\rvert$.
Z_l	The list of dependency labels of Z, where $Z_l^{(i)}$ is the label of incoming edge of $z^{(i)}$ for every $i = 1, \ldots, \lvert Z\rvert$.
$m(Z, R)$	A set of mention pairs in $Z, R : \{\langle S_Z, S_R\rangle \mid S_R \prec R$ and $S_Z \prec Z$ are mentions of the same entity$\}$
$S \prec Z$	S is a span in Z, such that $\exists i \in 1, \ldots, \lvert Z\rvert - \lvert S\rvert + 1 : \forall j = 0, \ldots, \lvert S\rvert - 1 : s_{1+j} = z_{i+j}$
$S \sqsubset Z$	S is a prefix of Z, such that $\forall i = 1, \ldots, \lvert S\rvert : s_i = z_i$
$i \curvearrowright_U j$	There is an edge from the ith token to the jth token in the dependency tree of Z.
$\mathcal{C}_b$	A set of backward connectives.
$\mathcal{C}_s$	A set of intra-sentence connectives, which are either forward connectives or conjunctions.
$\mathcal{C}_f$	A set of forward connectives.
$\mathcal{C}_c$	A set of coordinating conjunctions.
$\mathcal{P}_r$	A set of relative pronouns.
$\mathcal{V}$	A set of POS tags for verbal phrases.

Table 11: Notation and definitions for Table 14 of generation rules. Z, R, S_Z, S_R are token lists and $1 \leq i, j \leq |Z|$ are indices. The full lists of connectives and POS tags are provided in Table 16.

Set	Values
$\mathcal{C}_b$	”accordingly”, ”additionally”, ”afterward”, ”alternatively”, ”although ,”, ”and”, ”as a result ,”, ”because of that”, ”because of this”, ”besides ,”, ”but”, ”by comparison ,”, ”by contrast ,”, ”by doing this ,”, ”by then”, ”consequently”, ”conversely”, ”else,”, ”finally ,”, ”for example”, ”for instance”, ”further ,”, ”furthermore”, ”hence ,”, ”however”, ”in contrast ,”, ”in fact ,”, ”in other words”, ”in particular ,”, ”in short ,”, ”in sum ,”, ”in the end ,”, ”in turn ,”, ”indeed ,”, ”instead ,”, ”lest”, ”likewise ,”, ”meantime ,”, ”in the meantime ,”, ”meanwhile ,”, ”moreover”, ”nevertheless”, ”next ,”, ”nonetheless”, ”on the contrary ,”, ”on the other hand”, ”or ,”, ”otherwise ,”, ”overall ,”, ”plus ,”, ”rather ,”, ”regardless ,”, ”similarly ,”, ”simultaneously”, ”specifically ,”, ”still ,”, ”then ,”, ”thereafter ,”, ”thereby ,”, ”therefore”, ”though ,”, ”thus ,”, ”ultimately ,”, ”whereas”, ”yet ,”, ”now ,”, ”second ,”, ”third ,”, ”basically ,”, ”this ,”, ”eventually ,”, ”obviously ,”, ”again ,”, ”fortunately ,”, ”luckily ,”, ”meaning ,”, ”interestingly ,”, ”anyway ,”, ”clearly ,”
$\mathcal{C}_s$	”because”, ”, because”, ”hence”, ”, while”, ”whereas”, ”, although”, ”although”, ”and although”, ”unless”, ”now that”, ”, now that”, ”so that”, ”, so that”, ”meaning”, ”, meaning”
$\mathcal{C}_f$	although, since, in addition to, aside from
$\mathcal{C}_c$	and, but, or, nor, yet, so, for
$\mathcal{P}_r$	who, which, whose, whom
$\mathcal{V}$	VB, VBD, VBG, VBN, VBP, VBZ

Table 12: Connectives and POS tags used in our detection rules. A preceding comma is allowed for conjunctions in $\mathcal{C}_c$. For the connectives *“although”* and *“since”* in $\mathcal{C}_f$, we do not allow a following comma.

Operation	Description
DELETE(X, i, n)	Delete a sequence of n tokens from X, starting from index i.
PREPEND(X, Y)	Attach the list Y at the beginning of X.
REPLACE(X, Y, Z)	Replace every occurrence of Y in X with Z, in a non-overlapping manner.
SPLIT(X, i)	Split X into two token lists $V = \{x_1, \ldots, x_{i-1}\}, W = \{x_i, \ldots, x_{\lvert X\rvert}\}$.
TRIM(X)	Delete all tokens in X after the first punctuation token, e.g. period, comma, etc.

Table 13: Operations upon token lists, which are used for generation of sentence pairs (Table 14). The arguments X, Y, Z are token lists and the arguments i, n are integers.

Phenomenon	**Input**	**Detection**	**Generation**
Discourse connective	(A,B)	$\exists S \in \mathcal{C}_b\,,\ i \in [1,5] : S \sqsubset \{b^{(i)}, \ldots, b^{(\lvert B\rvert)}\}$	$\textsc{Delete}(B, i, \lvert S\rvert)$
Anaphora	(A,B)	$m(B,A) \neq \emptyset$	$\textsc{Replace}(B, S_B, S_A)$ $\forall\langle S_B,\ S_A\rangle \in m(B,A)$
Forward connective	Z	$\exists S \in \mathcal{C}_f : S \sqsubset Z$ $\exists i : \lvert S\rvert + 1 < i < \lvert Z\rvert \wedge z^{(i)} = \text{",''}$	$A, B \leftarrow \textsc{Split}(Z, i)$ $\textsc{Delete}(A, 1, \lvert S\rvert)$ $\textsc{Delete}(B, 1, 1)$
Inner connective	Z	$\exists S \in \mathcal{C}_s : S \prec Z$	$A, B \leftarrow \textsc{Split}(Z, i)$ $\textsc{Delete}(B, 1, \lvert S\rvert)$ $\textsc{Trim}(B)$
Cataphora	Z	$\exists i : 1 < i < \lvert Z\rvert \wedge z^{(i)} = \text{",''}$ $Z_t^{(1)} = \text{VBG} \wedge Z_l^{(1)} = \text{vmod} \wedge Z_t^{(i+2)} \in \mathcal{V}$	$A, B \leftarrow \textsc{Split}(Z, i)$ $\textsc{Prepend}(A, \{b_1\})$ $\textsc{Delete}(B, 1, 1)$
Sentence coordination	Z	$\exists i < j \leq i + 5 : Z_l^{(i)} = \text{cc} \ \wedge\ Z_l^{(j)} = \text{conj}$ $\exists i < k < j : Z_l^{(k)} \in \{\text{nsubj, nsubjpass}\}$	$A, B \leftarrow \textsc{Split}(Z, i)$ $\textsc{Delete}(B, 1, 1)$
Verb phrase coordination	Z	$\exists i, j : i < j \leq i + 5 : Z_l^{(i)} = \text{cc} \ \wedge\ Z_l^{(j)} = \text{conj} \ \wedge\ Z_t^{(j)} \in \mathcal{V}$ $\exists k : k \curvearrowright_Z j \ \wedge\ Z_l^{(k)} = \text{root}$	$A, B \leftarrow \textsc{Split}(Z, i)$ $\textsc{Delete}(B, 1, 1)$ $\textsc{Prepend}(B, \{a_1, \ldots, a_{k-1}\})$
Relative clause	Z	$\exists i, j : 1 < i < j < \lvert Z\rvert \wedge z^{(i)} = z^{(j)} = \text{",''}$ $z^{(i+1)} \in \mathcal{P}_r$	$A, B \leftarrow \textsc{Split}(Z, j)$ $\textsc{Delete}(B, 1, 1)$ $\textsc{Delete}(A, i, 2)$ $\textsc{Prepend}(B, \{a_r, \ldots, a_{i-1}\})^*$
Apposition	Z	$\exists i, j : 1 < i < j < \lvert Z\rvert \wedge z^{(i)} = z^{(j)} = \text{",''}$ $Z_l^{(i+1)} \in \{\text{det, poss}\} \ \wedge\ \exists k : i < k < j, Z_l^{(k)} = \text{appos}$	$A, B \leftarrow \textsc{Split}(Z, j)$ $\textsc{Delete}(B, 1, 1)$ $\textsc{Delete}(A, i, 2)$ $\textsc{Prepend}(B, \{a_r, \ldots, a_{i-1}\})^*$

Table 14: Generation rules for sentence pairs. The rules apply for token lists Z, A, B, where Z represents a single sentence and A, B either represent two consecutive sentences or two consecutive sentence parts. *For the rules of relative clause and apposition, r is the index of the leftmost child in the dependency sub-tree of $a^{(i)}$.

1. Input
$Z = \{$Ruiz ordered his first shot to be retaken because Brazilian players entered the penalty area before his kick .$\}$

↓

2. Inner connective
Detection
For $S =$ "because" it holds that $S \in \mathcal{C}_s$ and $S \prec Z$.

Generation

$\textsc{Split}(Z, i)$	$A = \{$Ruiz ordered his first shot to be retaken .$\}$ $B = \{$Because Brazilian players entered the penalty area before his kick .$\}$
$\textsc{Delete}(B, 1, \lvert S\rvert)$	$B = \{$ Brazilian players entered the penalty area before his kick .$\}$
$\textsc{Trim}$(B)	*no effect at this case*

↓

3. Anaphora
Detection
$m(B, A) = \{(\text{his, Ruiz})\} \neq \emptyset$

Generation

$\textsc{Replace}$ $(B, \text{his, Ruiz})$	$B = \{$ Brazilian players entered the penalty area before Ruiz 's kick .$\}$

↓

4. Output sentence pair
$A = \{$Ruiz ordered his first shot to be retaken . $\}$
$B = \{$Brazilian players entered the penalty area before Ruiz 's kick . $\}$

Table 15: Detailed two-rule execution example. We show in red parts of the input that are used for detection or modified during execution. The input token list Z is of a single sentence. First, the rule for inner connective is applied, splitting Z into two sentences A, B, without the connective *"because"*. Then, applying the anaphora rule, the pronoun *"his"* in B is replaced with the entity it refers to in A, to obtain a new sentence pair.

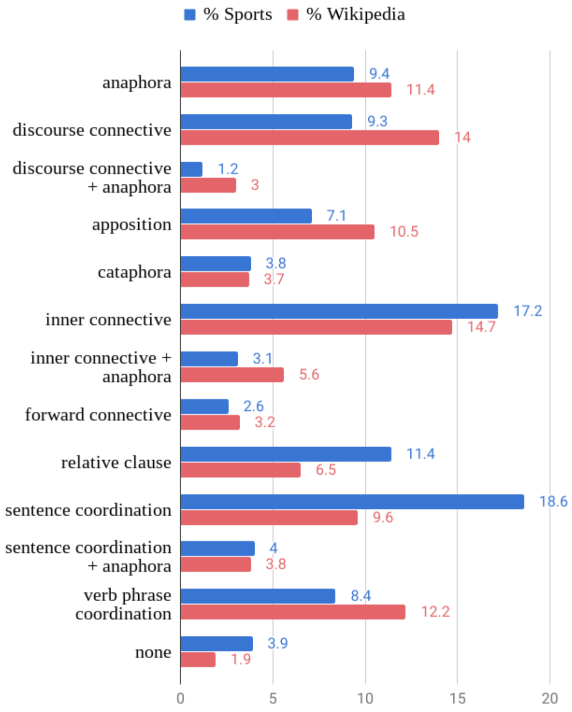


Figure 4: Discourse type distribution of the sports and Wikipedia portions of DiscoFuse after down-sampling.

by their text length, and use different batch size between 60-100 for each bucket. Further configuration details and the hyperparameters used for training of each model are provided in Table 17.

In our transfer learning experiment, we trained two CopyNet models (Gu et al., 2016): a model

SPORTS	%	WIKIPEDIA	%
and	12.0	and	12.5
but	10.9	but	10.7
because	8.1	although	8.4
although	5.4	however	8.2
so	4.9	because	7.7
or	4.8	so that	2.1
however	3.5	while	2.0
while	2.4	or	1.8
so that	2.2	so	1.2
unless	2.2	for example	1.0

Table 16: Most common connectives in DISCOFUSE after down-sampling. Percentages are with respect to the entire dataset, including examples without a connective.

	DFSPORT	DFWIKI	DFS+W
number of hidden layers	7	7	7
hidden dimension	1024	1024	1024
filter size	2048	2048	2048
number of heads	16	16	16
beam width	4	4	4
attention dropout rate	0.1	0.2	0.2
ReLU dropout rate	0.4	0.3	0.2
learning rate	0.14	0.07	0.11

Table 17: Parameters and hyperparameters of the models DFSPORT, DFWIKI, DFS+W.

number of encoder layers	3
number of decoder layers	1
hidden dimension	128
beam width	20
scheduled sampling probability	0.2
dropout rate	0.2
learning rate	0.001
learning rate decay	0.98

Table 18: Parameters and hyperparameters of the CopyNet models used for transfer learning.

trained on WEBSPLIT alone, and a model pretrained on DFWIKI and finetuned on WEBSPLIT. The first model was trained for 200,000 steps on WEBSPLIT, whereas the second model was pretrained for 1 million steps on DFWIKI and then finetuned for 100,000 steps on WEBSPLIT. Again, the samples were split to buckets by their text length, with batch sizes between 25-125 for each bucket. The final test scores were computed with the parameters that maximize the validation *SARI* score during training. The network architecture and hyperparameters were shared between the models and not optimized during training. They are listed in Table 18.

Linguistically-Informed Specificity and Semantic Plausibility for Dialogue Generation

Wei-Jen Ko[1] **Greg Durrett**[1] **Junyi Jessy Li**[2]
[1] Department of Computer Science
[2] Department of Linguistics
The University of Texas at Austin
wjko@utexas.edu, gdurrett@cs.utexas.edu, jessy@austin.utexas.edu

Abstract

Sequence-to-sequence models for open-domain dialogue generation tend to favor generic, uninformative responses. Past work has focused on word frequency-based approaches to improving specificity, such as penalizing responses with only common words. In this work, we examine whether specificity is solely a frequency-related notion and find that more linguistically-driven specificity measures are better suited to improving response informativeness. However, we find that forcing a sequence-to-sequence model to be more specific can expose a host of other problems in the responses, including flawed discourse and implausible semantics. We rerank our model's outputs using externally-trained classifiers targeting each of these identified factors. Experiments show that our final model using linguistically motivated specificity and plausibility reranking improves the informativeness, reasonableness, and grammatically of responses.

1 Introduction

Since the pioneering work in machine translation (Sutskever et al., 2014), sequence-to-sequence (SEQ2SEQ) models have led much recent progress in open-domain dialogue generation, especially single-turn generation where the input is a prompt and the output is a response. However, SEQ2SEQ methods are known to favor universal responses, e.g., "I don't know what you are talking about" (Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a). These responses tend to be "safe" responses to many input queries, yet they usually fail to provide useful information.

One promising line of research tackling this issue is to improve the specificity of responses, building on the intuition that generic responses frequently appear in the training data or consist of frequent words (Yao et al., 2016; Zhang et al., 2018b; Liu et al., 2018). However, past work in sentence specificity—the "quality of belonging or relating uniquely to a particular subject"[1]—has shown that word frequency is only one aspect of specificity, and that specificity involves a wide range of phenomena including word usage, sentence structure (Louis and Nenkova, 2011; Li and Nenkova, 2015; Lugini and Litman, 2017) and discourse context (Dixon, 1987; Lassonde and O'Brien, 2009). Frequency-based specificity also does not exactly capture "the amount of information" as an information-theoretic concept. Hence, in dialogue generation, we can potentially make progress by incorporating more linguistically driven measures of specificity, as opposed to relying solely on frequency.

We present a sequence-to-sequence dialogue model that factors out specificity and explicitly conditions on it when generating a response. The decoder takes as input categorized values of several specificity metrics, embeds them, and uses them at each stage of decoding. During training, the model can learn to associate different specificity levels with different types of responses. At test time, we set the specificity level to its maximum value to force specific responses, which we found to be most beneficial. We integrate linguistic (Ko et al., 2019), information-theoretic, and frequency-based specificity metrics to better understand their roles in guiding response generation.

The second component of our model is designed to make the more specific responses more *semantically plausible*. In particular, we found that forcing a SEQ2SEQ model to be more specific exposes problems with plausibility as illustrated in Table 1. As sentences become more specific and contain more information, intra-response consistency

---

[1] Definition from the Oxford Dictionary

Proceedings of NAACL-HLT 2019, pages 3456–3466
Minneapolis, Minnesota, June 2 - June 7, 2019.

Conflicting	i understand. i am not sure if i can afford a babysitter, i am a millionaire
Wrong connective	i am an animal phobic, but i do not like animals
Wrong pronoun	my mom was a social worker, he was an osteopath.
Wrong noun	cool. i work at a non profit organization that sells the holocaust.
Repeating	my favorite food is italian, but i also love italian food, especially italian food.

Table 1: Examples of different types of implausible responses on the PersonaChat dataset generated from our system that maximizes specificity only.

problems become evident, making the overall response implausible or unreasonable in real life. Our inspection discovered that ∼30% of specific responses suffer from a range of problems from semantic incompatibility to flawed discourse. To improve the plausibility of responses, we propose a reranking method based on four external classifiers, each targeting a separate aspect of linguistic plausibility. These classifiers are learned on synthetically generated examples, and at test time their responses are used to rerank proposed responses and mitigate the targeted issues.

Using both automatic and human evaluation, we find that linguistic-based specificity is more suitable than frequency-based specificity for generating informative and topically relevant responses, and learning from different types of specificity metrics leads to further improvement. Our plausibility reranking method not only successfully improved the semantic plausibility of responses, but also improved their informativeness, relevance, and grammaticality.

Our system is available at `https://git.io/fjkDd`.

2 Related work

Generic responses is a recognized problem in dialogue generation. Li et al. (2016a) maximized mutual information in decoding or reranking, which practically looks like penalizing responses that are common under a language model. Zhou et al. (2017) promoted diversity by training latent embeddings to represent different response mechanisms. Shao et al. (2017) trained and reranked responses segment by segment with a glimpse model to inject diversity. Another angle is to promote prompt-response coherence using techniques such as LDA (Baheti et al., 2018; Xing et al., 2017). Cosine similarity between prompt and response has also been used for coherence (Xu et al., 2018b; Baheti et al., 2018). Wu et al. (2018) learn a small vocabulary of words that may be relevant during decoding and generates responses with this vocabulary.

Several works tackle the problem by directly controlling response specificity in terms of word and response frequency. IDF and response frequency have been used as rewards in reinforcement learning (Yao et al., 2016; Li et al., 2016d). Some methods adjusted sample weights in the training data, using a dual encoding model (Lison and Bibauw, 2017) or sentence length and frequency in the corpus (Liu et al., 2018). Zhang et al. (2018b) proposed a Gaussian mixture model using frequency-based specificity values. Their approach involves ensembling the context probability and a specificity probability, whereas our approach conditions on both in a single model.

Prediction of **sentence specificity** following the dictionary definition and pragmatically cast as "level of detail" was first proposed by Louis and Nenkova (2011), who related specificity to discourse relations. Sentence specificity predictors have since been developed (Louis and Nenkova, 2011; Li and Nenkova, 2015; Lugini and Litman, 2017; Ko et al., 2019). Insights from these feature-rich systems and hand-code analysis (Li et al., 2016e) showed that sentence specificity encompasses multiple phenomena, including referring expressions, concreteness of concepts, gradable adjectives, subjectivity and syntactic structure.

Researchers have noticed that distributional semantics largely fail to capture **semantic plausibility**, especially in terms of discrete properties (e.g., negation) (Kruszewski et al., 2016) and physical properties (Wang et al., 2018). Kruszewski et al. (2016) created a dataset building on synthetically generated sentences for negation plausibility.

Methodology-wise, Li et al. (2016b) trained embeddings for different speakers jointly with the dialogue context. Huang et al. (2018) learned embeddings of emotions; we learn embeddings of specificity metrics. Targeting multiple factors this way is broadly similar to the approach of Holtzman et al. (2018), who used multiple cooperative discriminators to model repetition, entailment, relevance, and lexical style in generation. Our approach additionally leverages synthetic synthetic sentences targeting a range of plausibility issues and trains discriminators for reranking.

3 Generating specific responses

Our main framework (Figure 1) is an attention-based SEQ2SEQ model (Section 3.1) augmented with the ability to jointly learn embeddings from a target metric (e.g., specificity) with the response (Section 3.2). We then integrate frequency-based, information-theoretic and linguistic notions of specificity (Section 3.3) as well as coherence (Section 3.4).

3.1 Base framework

Our model is based on a SEQ2SEQ model (Sutskever et al., 2014) consisting of an encoder and decoder, both of which are LSTMs (Hochreiter and Schmidhuber, 1997). We apply attention (Bahdanau et al., 2015) on the decoder. The encoder LSTM takes word embeddings x_i in the prompt sentence as input. The hidden layer and cell state of the decoder are initialized with the final encoder states. During training, the decoder takes the embedding of the previous word in the gold response as input; during testing, it uses the previous generated word. We denote both as y_{i-1}:

$$h_i^d, c_i^d = LSTM(y_{i-1}, [\hat{h}_i^d; c_{i-1}^d]) \qquad (1)$$

where $\hat{h}_i^d$ is the output of the attention mechanism, given the decoder hidden state.

During training, we minimize the negative log likelihood of responses Y given prompts X.

3.2 Conditioning on specificity

In the base model, uninformative responses are preferred partially because these are common in the training data. We want to be able to fit the training data while at the same time recognizing that we do not want to generate such responses at test time. Our approach, shown in Figure 1, involves conditioning on an explicit specificity level during both training and test time. This explicit conditioning allows us to model specificity orthogonally to response content, so we can control it at test time. We represent specificity as a collection of real valued metrics that can be estimated for each sentence independently of the dialogue system. To direct the model to generate more specific responses from multiple specificity metrics, we learn embeddings of various specificity levels for each metric jointly with the model.

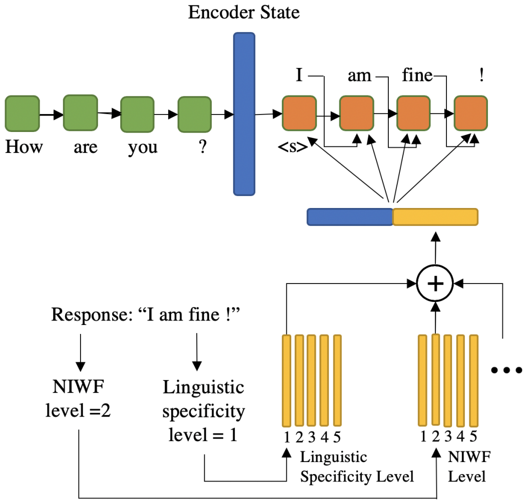


Figure 1: Structure of our model. The decoder explicitly conditions on embeddings of various specificity measures. At train time, these factor out specificity from generation; at test time, maxing these out encourages the model to generate specific responses. NIWF is defined in Section 3.3.

In particular, for each metric m, we rank the responses in the training data according to that metric and divide it into $K = 5$ levels of equal size. For each level, we learn an embedding e_k^m, $k \in \{1, 2, ...K\}$. During training, for each sentence pair in the training set, the response is classified to level l_m for metric m. We take the sum of embeddings across all metrics $e = \sum_{m=1}^{N} e_{l_m}^m$ and feed it into the decoder at every time step, where N is the number of metrics. The decoder becomes

$$h_i^d, c_i^d = LSTM(y_{i-1}, [\hat{h}_i^d; c_{i-1}^d; e]) \qquad (2)$$

During testing, we specify a level for each metric and calculate e based on those levels. In practice, the level of specificity varies with the larger context of dialogue discourse, however for the purpose of avoiding generic responses and improving specificity in single-turn dialogue generation, and examining various metrics of specificity, we use the level that maximizes specificity at test time (which we show in Section 5.3 is better the uninformative "median" level).[2]

---

[2] For the purposes of this work, we want an agent that is highly specific and keeps the conversation going. Learning the ideal specificity for a given response is something we leave for future work.

3.3 Specificity metrics

Normalized inverse word frequency (NIWF) Used in Zhang et al. (2018b), NIWF is the maximum of the Inverse Word Frequency (IWF) of all the words in a response, normalized to 0-1:

$$\max(IWF) = \max\left(\frac{\log(1+|Y|)}{f_w}\right) \quad (3)$$

where f_w denotes the number of responses in the corpus that contain the word w, and $|Y|$ is the number of responses in the corpus. Taking a maximum reflects the assumption that a response is specific as long as it has at least some infrequent word.

Perplexity per word (PPW) Perplexity is the exponentiation of the entropy, which estimates the expected number of bits required to encode the sentence (Brown et al., 1992; Goodman, 2001). Thus perplexity is a direct measure of the amount of information in the sentence in information theory; it has also been used as a measure of linguistic complexity (Gorin et al., 2000). To compute perplexity, we train a neural language model (Mikolov et al., 2011) on all gold responses and calculate cross-entropy of each sentence. To represent the amount of information per-token and to prevent the model to simply generate long sentences, we normalize perplexity by sentence length.

Linguistically-informed specificity We use the system developed by Ko et al. (2019), which estimates specificity as a real value. This system adopts a pragmatic notion of specificity—level of details in text—that is originally derived using sentence pairs connected via the INSTANTIATION discourse relation (Louis and Nenkova, 2011). With this relation, one sentence explains in further detail of the content in the other; the explanatory sentence is shown to demonstrate properties of specificity towards particular concepts, entities and objects, while the other sentence is much more general (Li and Nenkova, 2016). We use this particular system since other specificity predictors are trained on news with binary specificity labels (Li and Nenkova, 2015). Ko et al. (2019) is an unsupervised domain adaptation system that predicts continuous specificity values, and was evaluated to be close to human judgments across several domains. We retrain their system using the gold responses in our data as unlabeled sentences in the unsupervised domain adaptation component.

3.4 Coherence

Prior work has shown that the universal response problem can be mitigated by improving the coherence between prompt and response (Zhang et al., 2018a; Xu et al., 2018b; Baheti et al., 2018). We introduce two methods to improve coherence upon the base model, and analyze specificity on top.

For better interactions between decoder embeddings and the prompt, we feed the final encoder state into every time step of the decoder, instead of only the first token. Thus the decoder becomes

$$h_i^d, c_i^d = LSTM(y_{i-1}, [\hat{h}_i^d; c_{i-1}^d; e; h_f]). \quad (4)$$

Furthermore, Zhang et al. (2018a) showed that responses ranked higher by humans are more similar to the prompt sentence vector. Thus we compute the cosine similarity between input and response representations. This is computed by the weighted average of all word embeddings in the sentence, where the weight of each word is its inverse document frequency. Our model additionally conditions on an embedding of this measure so that coherence is factored out in our model as well as specificity. During testing, we condition on the highest level of our similarity metric in order to generate maximally coherent responses (Xu et al., 2018b).

4 Semantic plausibility

While injecting specificity encourages the model to generate more specific responses, we discovered that it exposes a series of issues that together, severely impact the semantic plausibility of generated responses. This is the case even when responses are considered independently without the prompt context. To have a better understanding of the problem, we first present manual analysis on generated responses with improved specificity. We then present a reranking method to improve the semantic plausibility of responses.

4.1 Data analysis

We manually inspected 200 responses generated from our full model on the PersonaChat dataset (Zhang et al., 2018c). We evaluated the responses independent of the input prompt and found that ∼33% of the sentences are semantically implausible; some of them shown in Table 1.

We found three major types of errors. The most common type is a wrong word that is not compatible with the context, making the phrase unreasonable (*cool . i work at a non profit organization that* *sells the* ***holocaust***), meaningless (*i like to* ***dance*** *battles*), or unnatural (*yeah , but i am more of a* ***game*** *worm . i am a pro ball player)*. These make up about 45% of the implausible cases.

About 30% of the problematic sentences contain incompatible phrases. Different phrases in the response are contradictory (*i understand. i am not sure if i can afford a babysitter, i am a millionaire*) or repetitive (*my favorite food is italian, but i also love italian food, especially italian food.*).

The third problem (∼15%) is that phrases are connected by a wrong discourse connective (*i am an animal phobic,* ***but*** *i do not like animals*). This and the previous problem reveal that even when the model generates sensible phrases, proper discourse relations between them are not captured.

Other notable errors include cohesion, such as wrong determiners or pronouns (*my mom was a social worker,* ***he*** *was an osteopath.*) and inappropriate prepositional phrases (*hello , i am winding down* ***to*** *the morning .*)

This semantic implausibility may come from two sources. First, since specific responses tend to be longer, it is easier to have internal consistency issues where parts of the sentence are incompatible with each other. Second, regardless of the specificity metric, word frequency in specific responses tend to be lower than that in generic responses. Learning meaningful representations for infrequent words is a known challenge (Gong et al., 2018) hence low-quality representations may increase the probability of the sentence being implausible.

4.2 Reranking

To mitigate semantic plausibility issues, we propose a reranking method so that more plausible sentences are ranked higher among the candidates. We use classifiers targeting various types of errors using synthetically generated data. Specifically, we train four classifiers that distinguish true response sentences from the dataset and negative sentences we create that reflect a specific type of semantic implausibility:

Phrase compatibility: We split all the training data into phrases by splitting sentences on punctuation or discourse connectives. To create a negative sentence given a gold response, we pick a random phrase in the true response and replace it with a random phrase in another random true response.

Content word plausibility: We replace a randomly selected content word (noun, verb, adjective, adverb) in the gold response with another random word with the same part-of-speech in the training set.

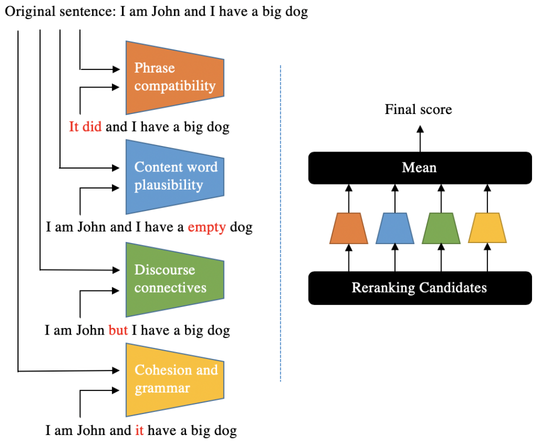


Figure 2: Reranking models to encourage plausibility. Four types of errors are synthetically applied to the data and classifiers are trained to differentiate each transformed sentence from the original. The mean score under these classifiers is then used as a feature to rerank system outputs.

Discourse connectives: We replace a discourse connective in the gold response (if one exists) with a random connective.

Cohesion and grammar: We replace a randomly selected function word in the gold response with another random function word of the same part-of-speech. For pronouns and determiners, these negative sentences would likely be incohesive; with other word categories such as prepositions, this will target grammatically.

One word or phrase is replaced in each synthetic sentence. We train one classifier $\theta_j, j \in \{1,2,3,4\}$ for each of the categories above.[3] The classifiers take word embeddings as input and predict if the response is real or generated. Each classifier consists of a bi-directional LSTM with a projection layer and max pooling (Conneau et al., 2017), followed by 3 fully connected layers. The posterior probabilities of these classifiers reflect how confident the classifiers are that the sentence is synthetic and prone to be implausible, hence we prefer sentences with lower posterior probabilities. During reranking, we feed each candidate sentence c into the classifiers and aggregate the posterior probabilities from these classifiers by taking the

---

[3] We compare with using one classifier lumping all negative sentences in the experiments.

mean $\frac{1}{4}\sum_{k=1}^{4} P(synthetic|c,\theta_k)$.

At test time, to encourage diversity, we repeat inference multiple times to generate different candidate sentences, and each time dropout is applied to different nodes in the network. Compared with diverse decoding (Li et al., 2016c), we observed during development that sentences generated by different dropouts tend to have diverse semantics (hence more likely to have different plausibility levels). On the contrary, sentences from diversity decoding often have similar structure and phrases across candidates. We also experimented with reinforcement learning, using policy gradient with the reranking scores as reward. However, during development, we observed that this method produced shorter, less informative sentences compared to reranking.

5 Experiments

5.1 Evaluation metrics

Automatic evaluation of dialogue generation systems is a known challenge. Prior work has shown that commonly used metrics for overall quality in other generation tasks such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005) and perplexity have poor correlations with human judgment (Liu et al., 2016; Tao et al., 2018)[4] or are model-dependent (Liu et al., 2016). Therefore, we adopt several metrics that evaluate multiple aspects of responses, and also conduct human evaluation for each result we present.

We use the following automatic evaluation metrics: (1) **distinct-1** and **distinct-2** (Li et al., 2016a), which evaluates response *diversity*. They respectively calculate the number of distinct unigrams and bigrams, divided by the total number of words in all responses; (2) linguistically-informed *specificity* (**spec**) (Ko et al., 2019); (3) **cosine similarity** between input and response representations, which captures *coherence* (Zhang et al., 2018a).

We follow standards from prior work for human evaluation (Li et al., 2017; Zhang et al., 2018a,b; Xu et al., 2018a). We select 250 prompt-response pairs, and asked 5 judges from MechanicalTurk to rate the responses for each prompt. We evaluate whether the responses are **informative** (Ko et al., 2019; Wu et al., 2018; Shao et al., 2017) and **on topic** with the prompt (Shen et al., 2018; Xu et al., 2018b; Xing et al., 2017), on a scale of 1-5. Average scores are reported. In addition, we evaluate **plausibility** by asking judges whether they think the given response sentence without the prompt can reasonably be uttered, following instructions from Kruszewski et al. (2016). The percentage of plausible ratings are reported.

5.2 Experiment setup

Data We use two datasets in this work: (1) *OpenSubtitles* (Tiedemann, 2009), a collection of movie subtitles widely used in open-domain dialogue generation. We sample 4,173,678 pairs for training and 5,000 pairs for testing from the movie subtitles dataset. Following Li et al. (2017), we remove all pairs with responses shorter than 5 words to improve the quality of the generated responses. (2) *PersonaChat* (Zhang et al., 2018c), a chit-chat dataset collected via crowdsourcing. This is a multi-turn dataset, but we only consider single turn generation in this work. We don't use the personas and false candidate replies. There are 122,458 prompt-response pairs for training and 14,602 pairs for testing. For validation, for reasons described in Section 5.1, we opt for human evaluation of overall response quality on a validation set of 60 prompt-response pairs from PersonaChat.

Settings We use LSTMs with hidden layers of size 500, Adam optimizer (Kingma and Ba, 2015) with learning rate 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$, dropout rate 0.2 for both training and testing, metric embedding dimension 300 and 5 training epochs. We train randomly initialized word embeddings of size 500 for the dialog model and use 300 dimentional GloVe (Pennington et al., 2014) embeddings for reranking classifiers. We generate 15 candidates for reranking per input sentence. To train the 4 reranking classifiers, we use 375,996 positive sentences on Opensubtitles and 110,221 on PersonaChat. We generate one negative sentence per word or phrase in the positive sentences.

Since specificity is the focus of this study, during testing, we use the embedding of the highest specificity level (5) for NIWF and the linguistically informed specificity predictor. For PPW, we observe that the perplexity of generated sentences does not increase beyond the median level (3) during development, hence we use the median level. For comparison, we also report results when all metric levels are set to be the median (level 3).

---

[4] Although Tao et al. (2018) proposed an unsupervised metric, their code is not available.

		human evaluation			automatic metrics			
	Model	Informative	On topic	Plausible	Dist-1	Dist-2	Spec.	Cos.Sim
Opensubtitles	Seq2seq	3.10	3.18	82.4	0.0349	0.138	0.133	0.638
	+coherence	2.59	3.25	90.0	**0.0538***	**0.217***	0.148	0.728*
	+specificity	3.29	**3.40***	72.4	0.0422	0.191*	**0.351***	**0.740***
	+plausibility	**3.53***	**3.39***	76.0	0.0524*	**0.217***	0.342*	0.711*
	MMI-Anti	3.29	3.07	**90.4**	0.0477	0.184	0.136	0.583
	Zhang	2.30	2.34	48.8	0.0305	0.161	0.163	0.648
PersonaChat	Seq2seq	3.01	2.84	86.9	0.0035	0.0126	0.381	0.703
	+coherence	2.71	3.16	**90.0**	0.0097	0.0428	0.279	**0.734***
	+specificity	3.31*	2.92	65.2	0.0135	0.0458	**0.491***	0.669
	+plausibility	**3.39***	**3.45***	71.7	**0.0184***	0.0679	0.483*	0.650
	MMI-Anti	2.79	3.23	79.0	0.0071	0.0289	0.200	0.706
	Zhang	3.07	3.06	44.9	0.0134	**0.0998**	0.400	0.662

Table 2: Results for the base SEQ2SEQ model and each component. (*) denotes significant improvement ($p < 0.05$ with paired bootstrap resampling) over our three baselines and benchmarks (*Seq2seq, MMI-Anti, Zhang*) according to paired bootstrap resampling. Our full reranking model performed the best in informativeness and topic relevance. A drop in plausibility is unavoidable as responses become more specific; the reranking model mitigates this.

Prompt (PersonaChat) thank you . also , the food is amazing . fries are the best thing ever .
Seq2seq yes it is . do you have a favorite color ? mine is blue .
Ours(Spec) yes , i am a vegetarian so i eat a lot of meat and bread puffs .
Ours(Spec+Plaus) i love italian food , but i am a vegan so i eat a lot of curries
MMI-Anti i agree . i am a vegetarian .
Zhang i agree . i love to eat and eat .

Prompt (PersonaChat) i work as kindergarten teacher at a school so pretty busy
Seq2seq what do you do for a living ? i work in a factory
Ours(Spec) i work at a grocery store and i work for a pharmaceutical company
Ours(Spec+Plaus) i work as a medical file clerk at a local hospital .
MMI-Anti what do you do for work ?
Zhang that is cool . i am a teacher and i love to be able to write .

Prompt (Opensubtitles) The countryside in spring Appears to the lovers in a new light . The air is already warm .
Seq2seq The snow is in the air .
Ours(Spec) It is the night of the dawn of the sunset .
Ours(Spec+Plaus) The light is rising from the skies , and the sun is shining .
MMI-Anti The light of the sun is rising .
Zhang The weather in the sky is a , .

Table 3: Example responses.

5.3 Results

We will first discuss results for the overall architecture, then dive into specificity and plausibility.

Overall architecture We evaluate our model against the base SEQ2SEQ for each component: coherence, specificity embeddings, and plausibility reranking (using the mean of all four classifiers). We also benchmark with the MMI-Anti model using mutual information (Li et al., 2016a), as well as Zhang et al. (2018b)'s model that incorproates a Gaussian kernel layer to control for specificity. We ran Zhang's code on our data and set $s = 1$ for PersonaChat and $s = 0.8$ for Opensubtitles when testing.[5] Significance tests are done via Paired Bootstrap Resampling (Berg-Kirkpatrick et al., 2012).

Table 2 shows that for both datasets, our full model with plausibility reranking (according to average posterior of the four classifiers) generates the most informative, relevant and plausible responses. Examples from our full model and the baselines are shown in Table 3.

Incorporating specificity led to more interesting responses, with 6-10% improvement in informativeness and 3-7% improvement in topic relevance. Since the system is trained without any semantics or common sense knowledge, this led to a drop in semantic plausibility. Plausibility reranking successfully mitigates this issue by improving plausibility by 3.6-6.5%. Although responses from MMI-Anti tend to be more plausible than directly using specificity, these responses are not useful if they are even less informative or relevant than the SEQ2SEQ baseline. Zhang et al. (2018b)'s model performed reasonably on PersonaChat but failed on OpenSubtitles.[6] One reason may be that OpenSubtitles is much more diverse in terms of topic and vocabulary, which makes their approach of estimating specificity independent of dialogue

---

[5] We observed that a higher s on Opensubtitles will result in many grammatical errors.

[6] Their original evaluation was on Chinese Weibo data.

Model	Informative		On topic	
Ours(Spec)	3.31		2.92	
-Linguistic	**3.11***	(-0.20)	**3.10***	(+0.18)
-NIWF	3.23	(-0.08)	3.20*	(+0.28)
-PPW	3.19*	(-0.12)	3.35*	(+0.43)

Table 4: Effect of excluding each specificity metric on PersonaChat. Delta against *Ours(Spec)* are included in parenthesis and (*) denotes significant delta ($p < 0.05$). Excluding linguistically informed specificity led to the greatest drop in informativeness and the slightest increase in topic relevance.

Model	Reranking	Inform.	Topic	Plaus.
Ours(Spec)	—	3.31	2.92	65.2
Ours(Spec +Plausibility)	1-classifier	3.26	3.36*	68.0*
	Max	**3.58***	3.35*	70.0*
	Mean	3.39	**3.45***	**71.7***
	+CoLA	3.45*	3.22*	68.5*
	CoLA	3.36	3.20*	58.0

Table 5: Comparison of different reranking methods on PersonaChat: training a single classifier, using max/mean posterior from four classifiers, and using CoLA. (*) denotes significant improvement over *Ours(Spec)* ($p < 0.05$). Learning multiple classifiers from synthetic data is the most effective.

context less effective. Indeed, we observe unstable word specificity learned across different training rounds and notable grammatical issues on OpenSubtitles. On the contrary, our joint approach gave stable performance on both datasets.

On PersonaChat, our coherence component led to improvements in topic relevance and cosine similarity, while specificity improved topic relevance and diversity, which is an intuitive result. On OpenSubtitles, coherence led to increased diversity while specificity led to a decrease. We looked into this and found that length trade-off is at play since the Distinct measures normalize by length of all generated responses: coherence led to diverse but short responses while specificity increased length. On human evaluation, they complement each other and using both gave better overall results. While reranking clearly did improve plausibility, there is also notable improvement in informativeness. This shows that informativeness is not only a frequency-only issue, or even a specificity-only issue, and that semantic plausibility plays an important role. Since the automatic metrics do not capture plausibility information in the sentence, it is unsurprising that they did not improve with plausibility added in.

We also study the effect of maxing out specificity and coherence levels at test time vs. using an uninformative level (median). Using median significantly improved informativeness and diversity (distinct-2) on PersonaChat by 0.90 and 0.53, and did not improve topic relevance. Similar but insignificant improvements are observed on OpenSubtitles. On the other hand, using the maximum levels led to significant improvements over the baseline or the median level on all metrics.

Specificity We now dive into a more detailed analysis for each specificity metric on PersonaChat. Table 4 shows human evaluation of informativeness, topic relevance and plausibility for the non-reranking model minus one specificity metric. Notably, excluding the linguistic based metric resulted in the largest drop in informativeness and relevance. Frequency based NIWF has the least impact on informativeness, indicating that specificity in dialogue is a multi-faceted issue and that the linguistically-informed notion is the most suitable. If none of the specificity metrics are included, topic relevance scores improve. This is because increasing specificity leads to fewer generic responses, yet they are more likely to be judged "on topic" by humans.

Plausibility We compare several different settings for plausibility reranking. Table 5 shows three ways of using the synthetically generated sentences discussed in Section 4: (1) **1-classifier**, which trains one classifier to distinguish true responses vs. all generated ones; (2) **Max**, which trains separate classifiers and take the maximum posterior probability (recall that higher posterior means less plausible responses); (3) **Mean**, which trains separate classifiers and averages the posterior probability. For all classifiers, at least 72% of the responses ranked top 50% on a balanced test set are true responses.

All three reranking methods helped, however, using one classifier is less effective than training and aggregating separate classifiers for each type of semantic implausibility. The latter not only improved plausibility but also informativeness and topic relevance. Using Max vs Mean yields comparable results in terms of plausibility, although Max improves informativeness more while Mean improves topic relevance more.

We also experimented with training an additional classifier (of the same architecture) on

Seq2seq	Ours(Spec)	Ours(Spec +Plausibility)	MMI -Anti	Zhang
82.4	78.7	82.6	90.0	61.5

Table 6: Percentage of sentences judged grammatical on OpenSubtitles.

the Corpus of Linguistic Acceptability (Warstadt et al., 2018), a dataset consisting of linguistically acceptable vs. unacceptable sentences. However, looking at results from PersonaChat, reranking using CoLA did not improve plausibility although is of slight help for informativeness and topic relevance. Combining CoLA with the other four classifiers decreased plausibility.

Grammaticality Finally, since the function word substitution aspect of our synthetic sentences is related to grammar, we also conduct human evaluation of grammaticality on OpenSubtitles. We did not evaluate on PersonaChat because almost all generate responses of our model we inspected are grammatically correct. Here annotators are asked to judge whether a sentence is grammatical vs. not. Results are shown in Table 6.

Informative and interesting responses that are the result of increasing specificity also made the model more prone to grammatical errors, but adding reranking completely mitigated this issue and grammaticality results are the same as the base model that generates much shorter, canned universal responses. MMI gave the best grammaticality; however, these response are not useful if they are even less informative or relevant than the SEQ2SEQ baseline. Zhang et al. (2018b)'s model generated more complicated sentences, but has worse grammar. Again we suspect that this is because of the lack of interaction between specificity estimates and dialogue context in their model.

6 Conclusion

We presented a new method to incorporate specificity information and semantic plausibility in SEQ2SEQ models. We showed that apart from frequency-based specificity metrics explored in prior work, information-theoretic and linguistically informed specificity improve the specificity of the responses. We proposed a reranking method aimed at improving the semantic plausibility of specific responses. Results showed that our method improved human ratings on informativeness, plausibility and grammaticality on both open domain and chit-chat datasets.

Acknowledgments

This work was partially supported by the NSF Grant IIS-1850153, and an Amazon Alexa Graduate Fellowship. We thank the anonymous reviewers for their helpful feedback.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *EMNLP*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *EMNLP*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Peter Dixon. 1987. The processing of organizational and component step information in written directions. *Journal of memory and language*, 26(1):24–35.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In *NIPS*.

Joshua Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15:403–434.

A.L. Gorin, J.H. Wright, G. Riccardi, A. Abella, and T. Alonso. 2000. Semantic information processing of spoken language. In *ATR Workshop on Multi-Lingual Speech Communication*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *ACL*.

Chenyang Huang, Osmar R. ZaIane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *NAACL*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Wei-Jen Ko, Greg Durrett, and Junyi Jessy Li. 2019. Domain agnostic real-valued specificity prediction. In *AAAI*.

Germán Kruszewski, Denis Paperno, Raffaella Bernardi, and Marco Baroni. 2016. There is no logical negation here, but there are alternatives: Modeling conversational negation with distributional semantics. *Computational Linguistics*, 42(4):637–660.

Karla A Lassonde and Edward J O'Brien. 2009. Contextual specificity in the activation of predictive inferences. *Discourse Processes*, 46(5):426–438.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *ACL*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016c. A simple, fast diverse decoding algorithm for neural generation. In *arXiv CS.CL*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016d. Deep reinforcement learning for dialogue generation. In *EMNLP*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *AAAI*.

Junyi Jessy Li and Ani Nenkova. 2016. The instantiation discourse relation: A corpus analysis of its properties and improved detection. In *NAACL*.

Junyi Jessy Li, Bridget O'Daniel, Yi Wu, Wenli Zhao, and Ani Nenkova. 2016e. Improving the annotation of sentence specificity. In *LREC*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Pierre Lison and Serge Bibauw. 2017. Not all dialogues are created equal: Instance weighting for neural conversational models. In *SIGDIAL*.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Yahui Liu, Victoria, Jun Gao, Xiaojiang Liu, Jian Yao, and Shuming Shi. 2018. Towards less generic responses in neural conversation models:a statistical re-weighting method. In *EMNLP*.

Annie Louis and Ani Nenkova. 2011. Automatic identification of general and specific sentences by leveraging discourse annotations. In *IJCNLP*.

Luca Lugini and Diane Litman. 2017. Predicting specificity in classroom discussion. In *Workshop on Innovative Use of NLP for Building Educational Applications*.

Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černockỳ. 2011. RNNLM - recurrent neural network language modeling toolkit. In *ASRU 2011 Demo Session*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *EMNLP*.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. In *AAAI*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*.

Jörg Tiedemann. 2009. News from OPUS - a collection of multilingual parallel corpora with tools and interfaces. In *RANLP*.

Su Wang, Greg Durrett, and Katrin Erk. 2018. Modeling semantic plausibility by injecting world knowledge. In *NAACL*.

Alex Warstadt, Amanpreet Singh, and Sam Bowman. 2018. Neural network acceptability judgments. In *arXiv CS.CL*.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural response generation with dynamic vocabularies. In *AAAI*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *AAAI*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018a. DP-GAN: Diversity-promoting generative adversarial network for generating informative and diversified text. In *EMNLP*.

Xinnuo Xu, Ondrej Dusek, Ioannis Konstas, and Verena Rieser. 2018b. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *EMNLP*.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, and Kam-Fai Wong. 2016. An attentional neural conversation model with improved specificity. In *arXiv CS.CL*.

Hainan Zhang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2018a. Reinforcing coherence for sequence to sequence model in dialogue generation. In *IJCAI*.

Ruqing Zhang, Jiafeng Guo, Yixing Fan, YanyanLan, Jun Xu, and Xueqi Cheng. 2018b. Learning to control the specificity in neural response generation. In *ACL*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018c. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *AAAI*.

Learning to Describe Unknown Phrases with Local and Global Contexts

Shonosuke Ishiwatari† **Hiroaki Hayashi**‡ **Naoki Yoshinaga**§ **Graham Neubig**‡
Shoetsu Sato† **Masashi Toyoda**§ **Masaru Kitsuregawa**¶§
† The University of Tokyo ‡ Carnegie Mellon University
§ Institute of Industrial Science, the University of Tokyo ¶ National Institute of Informatics
†§¶`{ishiwatari,ynaga,shoetsu,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp`
‡`{hiroakih, gneubig}@cs.cmu.edu`

Abstract

When reading a text, it is common to become stuck on unfamiliar words and phrases, such as polysemous words with novel senses, rarely used idioms, internet slang, or emerging entities. If we humans cannot figure out the meaning of those expressions from the immediate *local* context, we consult dictionaries for definitions or search documents or the web to find other *global* context to help in interpretation. Can machines help us do this work? Which type of context is more important for machines to solve the problem? To answer these questions, we undertake a task of describing a given phrase in natural language based on its local and global contexts. To solve this task, we propose a neural description model that consists of two context encoders and a description decoder. In contrast to the existing methods for non-standard English explanation (Ni and Wang, 2017) and definition generation (Noraset et al., 2017; Gadetsky et al., 2018), our model appropriately takes important clues from *both* local and global contexts. Experimental results on three existing datasets (including WordNet, Oxford and Urban Dictionaries) and a dataset newly created from Wikipedia demonstrate the effectiveness of our method over previous work.

1 Introduction

When we read news text with emerging entities, text in unfamiliar domains, or text in foreign languages, we often encounter expressions (words or phrases) whose senses we do not understand. In such cases, we may first try to figure out the meanings of those expressions by reading the surrounding words (*local* context) carefully. Failing to do so, we may consult dictionaries, and in the case of polysemous words, choose an appropriate meaning based on the context. Learning novel word senses via dictionary definitions is known to be more effective than contextual guessing (Fraser, 1998; Chen, 2012). However, very often, hand-crafted dictionaries do not contain definitions of expressions that are rarely used or newly created. Ultimately, we may need to read through the entire document or even search the web to find other occurances of the expression (*global* context) so that we can guess its meaning.

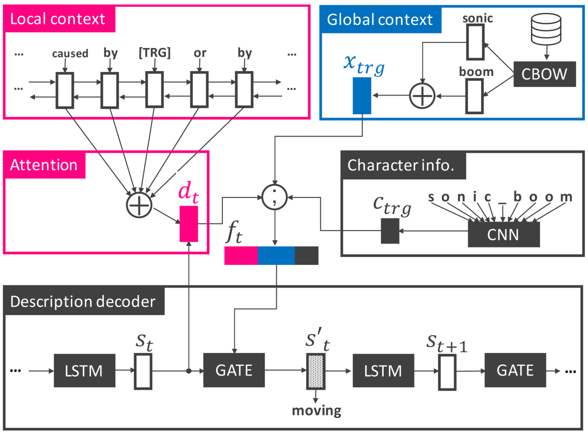


Figure 1: **Lo**cal & **G**lobal **C**ontext-**a**ware **D**escription generator (**LOG-CaD**).

Can machines help us do this work? Ni and Wang (2017) have proposed a task of generating a definition for a phrase given its local context. However, they follow the strict assumption that the target phrase is newly emerged and there is only a single local context available for the phrase, which makes the task of generating an accurate and coherent definition difficult (perhaps as difficult as a human comprehending the phrase itself). On the other hand, Noraset et al. (2017) attempted to generate a definition of a word from an embedding induced from massive text (which can be seen as global context). This is followed by Gadetsky et al. (2018) that refers to a local context to disambiguate polysemous words by choosing relevant dimensions of their word embeddings. Al-

Proceedings of NAACL-HLT 2019, pages 3467–3476
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

though these research efforts revealed that both local and global contexts are useful in generating definitions, none of these studies exploited both contexts directly to describe unknown phrases.

In this study, we tackle the task of describing (defining) a phrase when given its local and global contexts. We present **LOG-CaD**, a neural description generator (Figure 1) to directly solve this task. Given an unknown phrase without sense definitions, our model obtains a phrase embedding as its global context by composing word embeddings while also encoding the local context. The model therefore combines both pieces of information to generate a natural language description.

Considering various applications where we need definitions of expressions, we evaluated our method with four datasets including WordNet (Noraset et al., 2017) for general words, the Oxford dictionary (Gadetsky et al., 2018) for polysemous words, Urban Dictionary (Ni and Wang, 2017) for rare idioms or slang, and a newly-created Wikipedia dataset for entities.

Our contributions are as follows:

- We propose **a general task of defining unknown phrases given their contexts**. This task is a generalization of three related tasks (Noraset et al., 2017; Ni and Wang, 2017; Gadetsky et al., 2018) and involves various situations where we need definitions of unknown phrases (§ 2).
- We propose a method for **generating natural language descriptions for unknown phrases with local and global contexts** (§ 3).
- As a benchmark to evaluate the ability of the models to describe entities, we **build a large-scale dataset** from Wikipedia and Wikidata for the proposed task. We release our dataset and the code[1] to promote the reproducibility of the experiments (§ 4).
- The proposed method achieves **the state-of-the-art performance** on our new dataset and the three existing datasets used in the related studies (Noraset et al., 2017; Ni and Wang, 2017; Gadetsky et al., 2018) (§ 5).

2 Context-aware Phrase Description Generation

In this section, we define our task of describing a phrase in a specific context. Given an undefined phrase $X_{trg} = \{x_j, \cdots, x_k\}$ with its context $X = \{x_1, \cdots, x_I\}$ $(1 \leq j \leq k \leq I)$, our task is to output a description $Y = \{y_1, \cdots, y_T\}$. Here, X_{trg} can be a word or a short phrase and is included in X. Y is a definition-like concrete and concise sentence that describes the X_{trg}.

For example, given a phrase "sonic boom" with its context "the shock wave may be caused by *sonic boom* or by explosion," the task is to generate a description such as "sound created by an object moving fast." If the given context has been changed to "this is the first official tour to support the band's latest studio effort, 2009's *Sonic Boom*," then the appropriate output would be "album by Kiss."

The process of description generation can be modeled with a conditional language model as

$$p(Y|X, X_{trg}) = \prod_{t=1}^{T} p(y_t|y_{<t}, X, X_{trg}). \quad (1)$$

3 LOG-CaD: Local & Global Context-aware Description Generator

In this section, we describe our idea of utilizing local and global contexts in the description generation task, and present the details of our model.

3.1 Local & global contexts

When we find an unfamiliar phrase in text and it is not defined in dictionaries, how can we humans come up with its meaning? As discussed in Section 1, we may first try to figure out the meaning of the phrase from the immediate context, and then read through the entire document or search the web to understand implicit information behind the text.

In this paper, we refer to the explicit contextual information included in a given sentence with the target phrase (i.e., the X in Eq. (1)) as "local context," and the implicit contextual information in massive text as "global context." While both local and global contexts are crucial for humans to understand unfamiliar phrases, are they also useful for machines to generate descriptions? To verify this idea, we propose to incorporate both local and global contexts to describe an unknown phrase.

---

[1] `https://github.com/shonosuke/ishiwatari-naacl2019`

3.2 Proposed model

Figure 1 shows an illustration of our **LOG-CaD** model. Similarly to the standard encoder-decoder model with attention (Bahdanau et al., 2015; Luong and Manning, 2016), it has a context encoder and a description decoder. The challenge here is that the decoder needs to be conditioned not only on the local context, but also on its global context. To incorporate the different types of contexts, we propose to use a gate function similar to Noraset et al. (2017) to dynamically control how the global and local contexts influence the description.

Local & global context encoders We first describe how to model local and global contexts. Given a sentence X and a phrase X_{trg}, a bi-directional LSTM (Gers et al., 1999) encoder generates a sequence of continuous vectors $\boldsymbol{H} = \{\boldsymbol{h}_1 \cdots, \boldsymbol{h}_I\}$ as

$$\boldsymbol{h}_i = \text{Bi-LSTM}(\boldsymbol{h}_{i-1}, \boldsymbol{h}_{i+1}, \boldsymbol{x}_i), \quad (2)$$

where $\boldsymbol{x}_i$ is the word embedding of word x_i. In addition to the local context, we also utilize the global context obtained from massive text. This can be achieved by feeding a phrase embedding $\boldsymbol{x}_{trg}$ to initialize the decoder (Noraset et al., 2017) as

$$\boldsymbol{y}_0 = \boldsymbol{x}_{trg}. \quad (3)$$

Here, the phrase embedding $\boldsymbol{x}_{trg}$ is calculated by simply summing up all the embeddings of words that consistute the phrase X_{trg}. Note that we use a randomly-initialized vector if no pre-trained embedding is available for the words in X_{trg}.

Description decoder Using the local and global contexts, a description decoder computes the conditional probability of a description Y with Eq. (1), which can be approximated with another LSTM as

$$\begin{aligned} \boldsymbol{s}_t &= \text{LSTM}(\boldsymbol{y}_{t-1}, \boldsymbol{s}'_{t-1}), && (4)\\ \boldsymbol{d}_t &= \text{ATTENTION}(\boldsymbol{H}, \boldsymbol{s}_t), && (5)\\ \boldsymbol{c}_{trg} &= \text{CNN}(X_{trg}), && (6)\\ \boldsymbol{s}'_t &= \text{GATE}(\boldsymbol{s}_t, \boldsymbol{x}_{trg}, \boldsymbol{c}_{trg}, \boldsymbol{d}_t), && (7)\\ p(y_t|y_{<t}, X_{trg}) &= \text{softmax}(\boldsymbol{W}_{s'}\boldsymbol{s}'_t + \boldsymbol{b}_{s'}), && (8) \end{aligned}$$

where $\boldsymbol{s}_t$ is a hidden state of the decoder LSTM ($\boldsymbol{s}_0 = \vec{0}$), and $\boldsymbol{y}_{t-1}$ is a jointly-trained word embedding of the previous output word y_{t-1}. In what follows, we explain each equation in detail.

Attention on *local* context Considering the fact that the local context can be relatively long (e.g., around 20 words on average in our Wikipedia dataset introduced in Section 4), it is hard for the decoder to focus on important words in local contexts. In order to deal with this problem, the ATTENTION$(\cdot)$ function in Eq. (5) decides which words in the local context X to focus on at each time step. $\boldsymbol{d}_t$ is computed with an attention mechanism (Luong and Manning, 2016) as

$$\begin{aligned} \boldsymbol{d}_t &= \sum_{i=1}^{T} \alpha_i \boldsymbol{h}_i, && (9)\\ \alpha_i &= \text{softmax}(\boldsymbol{U}_h \boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{U}_s \boldsymbol{s}_t), && (10) \end{aligned}$$

where $\boldsymbol{U}_h$ and $\boldsymbol{U}_s$ are matrices that map the encoder and decoder hidden states into a common space, respectively.

Use of character information In order to capture the surface information of X_{trg}, we construct character-level CNNs (Eq. (6)) following (Noraset et al., 2017). Note that the input to the CNNs is a sequence of words in X_{trg}, which are concatenated with special character "_," such as "sonic_boom." Following Noraset et al. (2017), we set the CNN kernels of length 2-6 and size $10, 30, 40, 40, 40$ respectively with a stride of 1 to obtain a 160-dimensional vector $\boldsymbol{c}_{trg}$.

Gate function to control local & global contexts In order to capture the interaction between the local and global contexts, we adopt a GATE$(\cdot)$ function (Eq. (7)) which is similar to Noraset et al. (2017). The GATE$(\cdot)$ function updates the LSTM output $\boldsymbol{s}_t$ to $\boldsymbol{s}'_t$ depending on the global context $\boldsymbol{x}_{trg}$, local context $\boldsymbol{d}_t$, and character-level information $\boldsymbol{c}_{trg}$ as

$$\begin{aligned} \boldsymbol{f}_t &= [\boldsymbol{x}_{trg}; \boldsymbol{d}_t; \boldsymbol{c}_{trg}] && (11)\\ \boldsymbol{z}_t &= \sigma(\boldsymbol{W}_z[\boldsymbol{f}_t; \boldsymbol{s}_t] + \boldsymbol{b}_z), && (12)\\ \boldsymbol{r}_t &= \sigma(\boldsymbol{W}_r[\boldsymbol{f}_t; \boldsymbol{s}_t] + \boldsymbol{b}_r), && (13)\\ \tilde{\boldsymbol{s}}_t &= \tanh(\boldsymbol{W}_s[(\boldsymbol{r}_t \odot \boldsymbol{f}_t); \boldsymbol{s}_t] + \boldsymbol{b}_s), && (14)\\ \boldsymbol{s}'_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{s}_t + z_t \odot \tilde{\boldsymbol{s}}_t, && (15) \end{aligned}$$

where $\sigma(\cdot)$, $\odot$ and ; denote the sigmoid function, element-wise multiplication, and vector concatenation, respectively. $\boldsymbol{W}_*$ and $\boldsymbol{b}_*$ are weight matrices and bias terms, respectively. Here, the update gate $\boldsymbol{z}_t$ controls how much the original hidden state $\boldsymbol{s}_t$ is to be changed, and the reset gate $\boldsymbol{r}_t$ controls how much the information from $\boldsymbol{f}_t$ contributes to word generation at each time step.

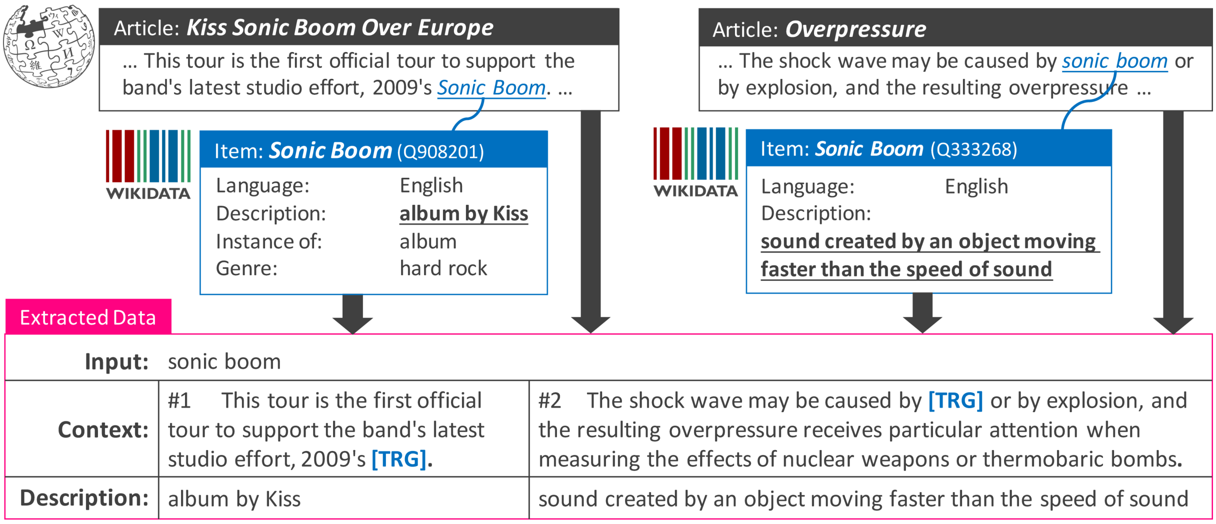


Figure 2: Context-aware description dataset extracted from Wikipedia and Wikidata.

4 Wikipedia Dataset

Our goal is to let machines describe unfamiliar words and phrases, such as polysemous words, rarely used idioms, or emerging entities. Among the three existing datasets, WordNet and Oxford dictionary mainly target the words but not phrases, thus are not perfect test beds for this goal. On the other hand, although the Urban Dictionary dataset contains descriptions of rarely-used phrases, the domain of its targeted words and phrases is limited to Internet slang.

In order to confirm that our model can generate the description of entities as well as polysemous words and slang, we constructed a new dataset for context-aware phrase description generation from Wikipedia[2] and Wikidata[3] which contain a wide variety of entity descriptions with contexts. The overview of the data extraction process is shown in Figure 2. Each entry in the dataset consists of (1) a phrase, (2) its description, and (3) context (a sentence).

For preprocessing, we applied Stanford Tokenizer[4] to the descriptions of Wikidata items and the articles in Wikipedia. Next, we removed phrases in parentheses from the Wikipedia articles, since they tend to be paraphrasing in other languages and work as noise. To obtain the contexts of each item in Wikidata, we extracted the sentence which has a link referring to the item through all the first paragraphs of Wikipedia articles and replaced the phrase of the links with a special token **[TRG]**. Wikidata items with no description or no contexts are ignored. This utilization of links makes it possible to resolve the ambiguity of words and phrases in a sentence without human annotations, which is a major advantage of using Wikipedia. Note that we used only links whose anchor texts are identical to the title of the Wikipedia articles, since the users of Wikipedia sometimes link mentions to related articles.

5 Experiments

We evaluate our method by applying it to describe words in WordNet[5] (Miller, 1995) and Oxford Dictionary,[6] phrases in Urban Dictionary[7] and Wikipedia/Wikidata.[8] For all of these datasets, a given word or phrase has an inventory of senses with corresponding definitions and usage examples. These definitions are regarded as ground-truth descriptions.

Datasets To evaluate our model on the word description task on WordNet, we followed Noraset et al. (2017) and extracted data from WordNet using the `dict-definition`[9] toolkit. Each entry in the data consists of three elements: (1) a word, (2) its definition, and (3) a usage example of the

---

[2] https://dumps.wikimedia.org/enwiki/20170720/

[3] https://dumps.wikimedia.org/wikidatawiki/entities/20170802/

[4] https://nlp.stanford.edu/software/tokenizer.shtml

[5] https://wordnet.princeton.edu/

[6] https://en.oxforddictionaries.com/

[7] https://www.urbandictionary.com/

[8] https://www.wikidata.org

[9] https://github.com/NorThanapon/dict-definition

Corpus	#Phrases	#Entries	Phrase length	Context length	Desc. length
WordNet					
Train	7,938	13,883	1.00	5.81	6.61
Valid	998	1,752	1.00	5.64	6.61
Test	1,001	1,775	1.00	5.77	6.85
Oxford Dictionary					
Train	33,128	97,855	1.00	17.74	11.02
Valid	8,867	12,232	1.00	17.80	10.99
Test	8,850	12,232	1.00	17.56	10.95
Urban Dictionary					
Train	190,696	411,384	1.54	10.89	10.99
Valid	26,876	57,883	1.54	10.86	10.95
Test	26,875	38,371	1.68	11.14	11.50
Wikipedia					
Train	151,995	887,455	2.10	18.79	5.89
Valid	8,361	44,003	2.11	19.21	6.31
Test	8,397	57,232	2.10	19.02	6.94

Table 1: Statistics of the word/phrase description datasets.

Corpus	Domain	Inputs	Cov. emb.
WordNet	General	words	100.00%
Oxford Dictionary	General	words	83.04%
Urban Dictionary	Internet slang	phrases	21.00%
Wikipedia	Proper nouns	phrases	26.79%

Table 2: Domains, expressions to be described, and the coverage of pre-trained embeddings of the expressions to be described.

	Global	Local	I-Attn.	LOG-CaD
# Layers of Enc-LSTMs	-	2	2	2
Dim. of Enc-LSTMs	-	600	600	600
Dim. of Attn. vectors	-	300	300	300
Dim. of input word emb.	300	-	300	300
Dim. of char. emb.	160	160	-	160
# Layers of Dec-LSTMs	2	2	2	2
Dim. of Dec-LSTMs	300	300	300	300
Vocabulary size	10k	10k	10k	10k
Dropout rate	0.5	0.5	0.5	0.5

Table 3: Hyperparameters of the models

word. We split this dataset to obtain Train, Validation, and Test sets. If a word has multiple definitions/examples, we treat them as different entries. Note that the words are mutually exclusive across the three sets. The only difference between our dataset and theirs is that we extract the tuples only if the words have their usage examples in WordNet. Since not all entries in WordNet have usage examples, our dataset is a small subset of Noraset et al. (2017).

In addition to WordNet, we use the Oxford Dictionary following Gadetsky et al. (2018), the Urban Dictionary following Ni and Wang (2017) and our Wikipedia dataset described in the previous section. Table 1 and Table 2 show the properties and statistics of the four datasets, respectively.

To simulate a situation in a real application where we might not have access to global context for the target phrases, we did not train domain-specific word embeddings on each dataset. Instead, for all of the four datasets, we use the same pre-trained CBOW[10] vectors trained on Google news corpus as global context following previous work (Noraset et al., 2017; Gadetsky et al., 2018). If the expression to be described consists of multiple words, its phrase embedding is calculated by simply summing up all the CBOW vectors of words in the phrase, such as "sonic" and "boom." (See Figure 1). If pre-trained CBOW embeddings are unavailable, we instead use a special **[UNK]** vector (which is randomly initialized with a uniform distribution) as word embeddings. Note that our pre-trained embeddings only cover 26.79% of the words in the expressions to be described in our Wikipedia dataset, while it covers all words in WordNet dataset (See Table 2). Even if no reliable word embeddings are available, all models can capture the character information through character-level CNNs (See Figure 1).

Models We implemented four methods: (1) **Global** (Noraset et al., 2017), (2) **Local** (Ni and Wang, 2017) with CNN, (3) **I-Attention** (Gadetsky et al., 2018), and our proposed model, (4) **LOG-CaD**. The **Global** model is our reimplementation of the best model (S + G + CH) in Noraset et al. (2017). It can access the global context of a phrase to be described, but has no ability to read the local context. The **Local** model is the reimplementation of the best model (dual encoder) in Ni and Wang(2017). In order to make a fair comparison of the effectiveness of local and global contexts, we slightly modify the original implementation by Ni and Wang(2017); as the character-level encoder in the **Local** model, we adopt CNNs that are exactly the same as the other two models instead of the original LSTMs.

The **I-Attention** is our reimplementation of the best model (S + I-Attention) in Gadetsky

---

[10] `GoogleNews-vectors-negative300.bin.gz` at `https://code.google.com/archive/p/word2vec/`

Model	WordNet	Oxford	Urban	Wikipedia
Global	24.10	15.05	6.05	44.77
Local	22.34	17.90	9.03	52.94
I-Attention	23.77	17.25	10.40	44.71
LOG-CaD	**24.79**	**18.53**	**10.55**	**53.85**

Table 4: BLEU scores on four datasets.

Model	**Annotated score**
Local	2.717
LOG-CaD	**3.008**

Table 5: Averaged human annotated scores on Wikipedia dataset.

et al.(2018). Similar to our model, it uses both local and global contexts. Unlike our model, however, it does not use character information to predict descriptions. Also, it cannot directly use the local context to predict the words in descriptions. This is because the **I-Attention** model indirectly uses the local context only to disambiguate the phrase embedding $\boldsymbol{x}_{trg}$ as

$$\boldsymbol{x}'_{trg} = \boldsymbol{x}_{trg} \odot \boldsymbol{m}, \tag{16}$$

$$\boldsymbol{m} = \sigma(\boldsymbol{W}_m \frac{\sum_{i=1}^{I} \mathrm{FFNN}(\boldsymbol{h}_i)}{I} + \boldsymbol{b}_m). \tag{17}$$

Here, the $\mathrm{FFNN}(\cdot)$ function is a feed-forward neural network that maps the encoded local contexts $\boldsymbol{h}_i$ to another space. The mapped local contexts are then averaged over the length of the sentence X to obtain a representation of the local context. This is followed by a linear layer and a sigmoid function to obtain the soft binary mask $\boldsymbol{m}$ which can filter out the unrelated information included in global context. Finally, the disambiguated phrase embedding $\boldsymbol{x}'_{trg}$ is then used to update the decoder hidden state as

$$\boldsymbol{s}_t = \mathrm{LSTM}([\boldsymbol{y}_{t-1}; \boldsymbol{x}'_{trg}], \boldsymbol{s}_{t-1}). \tag{18}$$

All four models (Table 3) are implemented with the PyTorch framework (Ver. 1.0.0).[11]

Automatic Evaluation Table 4 shows the BLEU (Papineni et al., 2002) scores of the output descriptions. We can see that the **LOG-CaD** model consistently outperforms the three baselines in all four datasets. This result indicates that using both local and global contexts helps describe the unknown words/phrases correctly. While the **I-Attention** model also uses local and global contexts, its performance was always lower than the **LOG-CaD** model. This result shows that using local context to predict description is more effective than using it to disambiguate the meanings in global context.

Input:	waste	
Context:	#1	#2
	if the effort brings no compensating gain it is a **waste**	We **waste** the dirty water by channeling it into the sewer
Reference:	useless or profitless activity	to get rid of
Global:	to give a liquid for a liquid	
Local:	a state of being assigned to a particular purpose	to make a break of a wooden instrument
I-Attention:	a person who makes something that can be be be done	to remove or remove the contents of
LOG-CaD:	a source of something that is done or done	to remove a liquid

Table 6: Descriptions for a word in WordNet.

Input:	daniel o'neill	
Context:	#1	#2
	after being enlarged by publisher **daniel o'neill** it was reportedly one of the largest and most prosperous newspapers in the united states.	in 1967 he returned to belfast where he met fellow belfast artist **daniel o'neill**.
Reference:	american journalist	irish artist
Global:	american musician	
Local:	american publisher	british musician
I-Attention:	american musician	american musician
LOG-CaD:	american writer	british musician

Table 7: Descriptions for a phrase in Wikipedia.

In particular, the low BLEU scores of **Global** and **I-Attention** models on Wikipedia dataset suggest that it is necessary to learn to ignore the noisy information in global context if the coverage of pre-trained word embeddings is extremely low (see the third and fourth rows in Table 2). We suspect that the Urban Dictionary task is too difficult and the results are unreliable considering its extremely low BLEU scores and high ratio of unknown tokens in generated descriptions.

---

[11] `http://pytorch.org/`

Input:	q			
Context:	#1	#2	#3	#4
	q-lets and co. is a filipino and english informative children 's show on **q** in the philippines .	she was a founding producer of the cbc radio one show " **q** " .	the q awards are the uk 's annual music awards run by the music magazine " **q** " .	charles fraser-smith was an author and one-time missionary who is widely credited as being the inspiration for ian fleming 's james bond quartermaster **q** .
Reference:	philippine tv network	canadian radio show	british music magazine	fictional character from james bond
Global:	american rapper			
Local:	television channel	television show show	magazine	american writer
I-Attention:	american rapper	american rapper	american rapper	american rapper
LOG-CaD:	television station in the philippines	television program	british weekly music journalism magazine	[unk] [unk]

Table 8: Descriptions for a word in Wikipedia.

Manual Evaluation To compare the proposed model and the strongest baseline in Table 4 (i.e., the **Local** model), we performed a human evaluation on our dataset. We randomly selected 100 samples from the test set of the Wikipedia dataset and asked three native English speakers to rate the output descriptions from 1 to 5 points as: 1) completely wrong or self-definition, 2) correct topic with wrong information, 3) correct but incomplete, 4) small details missing, 5) correct. The averaged scores are reported in Table 5. Pair-wise bootstrap resampling test (Koehn, 2004) for the annotated scores has shown that the superiority of **LOG-CaD** over the **Local** model is statistically significant ($p < 0.01$).

Qualitative Analysis Table 6 shows a word in the WordNet, while Table 7 and Table 8 show the examples of the entities in Wikipedia as examples. When comparing the two datasets, the quality of generated descriptions of Wikipedia dataset is significantly better than that of WordNet dataset. The main reason for this result is that the size of training data of the Wikipedia dataset is 64x larger than the WordNet dataset (See Table 1).

For all examples in the three tables, the **Global** model can only generate a single description for each input word/phrase because it cannot access any local context. In the Wordnet dataset, only the **I-Attention** and **LOG-CaD** models can successfully generate the concept of "remove" given the context #2. This result suggests that considering both local and global contexts are essential to generate correct descriptions. In our Wikipedia dataset, both the **Local** and **LOG-CaD** models can describe the word/phrase considering its local context. For example, both the **Local** and **LOG-CaD** models could generate "american" in the description for "daniel o'neill" given "united states" in context #1, while they could generate "british" given "belfast" in context #2. A similar trend can also be observed in Table 8, where **LOG-CaD** could generate the locational expressions such as "philippines" and "british" given the different contexts. On the other hand, the **I-Attention** model could not describe the two phrases, taking into account the local contexts. We will present an analysis of this phenomenon in the next section.

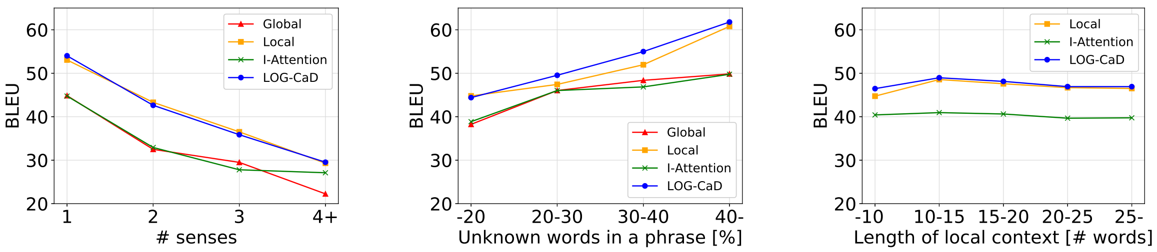
6 Discussion

In this section, we present analyses on how the local and global contexts contribute to the description generation task. First, we discuss how the local context helps the models to describe a phrase. Then, we analyze the impact of global context under the situation where local context is unreliable.

6.1 How do the models utilize *local* contexts?

Local context helps us (1) disambiguate polysemous words and (2) infer the meanings of unknown expressions. Can machines also utilize the local context? In this section, we discuss the two roles of local context in description generation.

Considering that the pre-trained word embeddings are obtained from word-level co-occurrences in massive text, more information is mixed up into a single vector as the more senses the word has. While Gadetsky et al. (2018) de-



(a) Number of senses of the phrase.

(b) Unknown words ratio in the phrase.

(c) Length of the local context.

Figure 3: Impact of various parameters of a phrase to be described on BLEU scores of the generated descriptions.

signed the **I-Attention** model to filter out unrelated meanings in the global context given local context, they did not discuss the impact of the number of senses has on the performance of definition generation. To understand the influence of the ambiguity of phrases to be defined on the generation performance, we did an analysis on our Wikipedia dataset. Figure 3(a) shows that the description generation task becomes harder as the phrases to be described become more ambiguous. In particular, when a phrase has an extremely large number of senses, (i.e., #senses ≥ 4), the **Global** model drops its performance significantly. This result indicates that the local context is necessary to disambiguate the meanings in global context.

As shown in Table 2, a large proportion of the phrases in our Wikipedia dataset includes unknown words (i.e., only 26.79% of words in the phrases have their pre-trained embeddings). This fact indicates that the global context in this dataset is not fully reliable. Then our next question is, how does the lack of information from global context affect the performance of phrase description? Figure 3(b) shows the impact of unknown words in the phrases to be described on the performance. As we can see from the result, the advantage of **LOG-CaD** and **Local** models over **Global** and **I-Attention** models becomes larger as the unknown words increases. This result suggests that we need to fully utilize local contexts especially in practical applications where the phrases to be defined have many unknown words. Here, Figure 3(b) also shows a counterintuitive phenomenon that BLEU scores increase as the ratio of unknown words in a phrase increase. This is mainly because unknown phrases tend to be person names such as writers, actors, or movie directors. Since these entities have fewer ambiguities in categories, they can be described in extremely short sentences that are easy for all four models to decode (e.g., "finnish writer" or "american television producer").

6.2 How do the models utilize *global* contexts?

As discussed earlier, local contexts are important to describe unknown expressions, but how about global contexts? Assuming a situation where we cannot obtain much information from local contexts (*e.g.*, infer the meaning of "boswellia" from a short local context "Here is a boswellia"), global contexts should be essential to understand the meaning. To confirm this hypothesis, we analyzed the impact of the length of local contexts on BLEU scores. Figure 3(c) shows that when the length of local context is extremely short ($l \leq 10$), the **LOG-CaD** model becomes much stronger than the **Local** model. This result indicates that not only local context but also global context help models describe the meanings of phrases.

7 Related Work

In this study, we address a task of describing a given phrase with its context. In what follows, we explain existing tasks that are related to our work.

Our task is closely related to word sense disambiguation (WSD) (Navigli, 2009), which identifies a pre-defined sense for the target word with its context. Although we can use it to solve our task by retrieving the definition sentence for the sense identified by WSD, it requires a substantial amount of training data to handle a different set of meanings of each word, and cannot handle words (or senses) which are not registered in the dictionary. Although some studies have attempted to detect novel senses of words for given contexts (Erk, 2006; Lau et al., 2014), they do not provide definition sentences. Our task avoids these difficulties in WSD by directly generating descriptions for phrases or words. It also allows us to flexibly tailor a fine-grained definition for the specific context.

Paraphrasing (Androutsopoulos and Malakasiotis, 2010; Madnani and Dorr, 2010) (or text simplification (Siddharthan, 2014)) can be used to rephrase words with unknown senses. However, the target of paraphrase acquisition are words/phrases with no specified context. Although a few studies (Connor and Roth, 2007; Max, 2009; Max et al., 2012) consider subsentential (context-sensitive) paraphrases, they do not intend to obtain a definition-like description as a paraphrase of a word.

Recently, Noraset et al. (2017) introduced a task of generating a definition sentence of a word from its pre-trained embedding. Since their task does not take local contexts of words as inputs, their method cannot generate an appropriate definition for a polysemous word for a specific context. To cope with this problem, Gadetsky et al. (2018) proposed a definition generation method that works with polysemous words in dictionaries. They presented a model that utilizes local context to filter out the unrelated meanings from a pre-trained word embedding in a specific context. While their method use local context for disambiguating the meanings that are mixed up in word embeddings, the information from local contexts cannot be utilized if the pre-trained embeddings are unavailable or unreliable. On the other hand, our method can fully utilize the local context through an attentional mechanism, even if the reliable word embeddings are unavailable.

The most related work to this paper is Ni and Wang (2017). Focusing on non-standard English phrases, they proposed a model to generate the explanations solely from local context. They followed the strict assumption that the target phrase was newly emerged and there was only a single local context available, which made the task of generating an accurate and coherent definition difficult. Our proposed task and model are more general and practical than Ni and Wang (2017); where (1) we use Wikipedia, which includes expressions from various domains, and (2) our model takes advantage of global contexts if available.

Our task of describing phrases with its context is a generalization of the three tasks (Noraset et al., 2017; Ni and Wang, 2017; Gadetsky et al., 2018), and the proposed method utilizes both local and global contexts of an expression in question.

8 Conclusions

This paper sets up a task of generating a natural language description for an unknown phrase with a specific context, aiming to help us acquire unknown word senses when reading text. We approached this task by using a variant of encoder-decoder models that capture the given local context with the encoder and global contexts with the decoder initialized by the target phrase's embedding induced from massive text. We performed experiments on three existing datasets and one newly built from Wikipedia and Wikidata. The experimental results confirmed that the local and global contexts complement one another and are both essential; global contexts are crucial when local contexts are short and vague, while the local context is important when the target phrase is polysemous, rare, or unseen.

As future work, we plan to modify our model to use multiple contexts in text to improve the quality of descriptions, considering the "one sense per discourse" hypothesis (Gale et al., 1992). We will release the newly built Wikipedia dataset and the experimental codes for the academic and industrial communities at `https://github.com/shonosuke/ishiwatari-naacl2019` to facilitate the reproducibility of our results and their use in various application contexts.

Acknowledgements

The authors are grateful to Thanapon Noraset for sharing the details of his implementation of the previous work. We also thank the anonymous reviewers for their careful reading of our paper and insightful comments, and the members of Kitsuregawa-Toyoda-Nemoto-Yoshinaga-Goda laboratory in the University of Tokyo for proofreading the draft.

This work was partially supported by Grant-in-Aid for JSPS Fellows (Grant Number 17J06394) and Commissioned Research (201) of the National Institute of Information and Communications Technology of Japan.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38:135–187.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

Yuzhen Chen. 2012. Dictionary use and vocabulary learning in the context of reading. *International Journal of Lexicography*, 25(2):216–247.

Michael Connor and Dan Roth. 2007. Context sensitive paraphrasing with a global unsupervised classifier. In *Proceedings of the 18th European Conference on Machine Learning (ECML)*, pages 104–115.

Katrin Erk. 2006. Unknown word sense detection as outlier detection. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pages 128–135.

Carol A. Fraser. 1998. The role of consulting a dictionary in reading and vocabulary learning. *Canadian Journal of Applied Linguistics*, 2(1-2):73–89.

Artyom Gadetsky, Ilya Yakubovskiy, and Dmitry Vetrov. 2018. Conditional generators of words definitions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Short Papers*, pages 266–271.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language, HLT*, pages 233–237.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, pages 850–855.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395.

Jey Han Lau, Paul Cook, Diana McCarthy, Spandana Gella, and Timothy Baldwin. 2014. Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 259–270.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1054–1063.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Aurélien Max. 2009. Sub-sentencial paraphrasing by contextual pivot translation. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 18–26.

Aurélien Max, Houda Bouamor, and Anne Vilnat. 2012. Generalizing sub-sentential paraphrase acquisition across original signal type of text pairs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 721–731.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM COMPUTING SURVEYS*, 41(2):1–69.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard English words and phrases. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 413–417.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 3259–3266.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165(2):259–298.

Mining Discourse Markers for Unsupervised Sentence Representation Learning

Damien Sileo[3,1], Tim Van De Cruys[2], Camille Pradel[1], and Philippe Muller[3]

[1]Synapse Développement, Toulouse, France
[2]IRIT, CNRS, France
[3]IRIT, University of Toulouse, France
damien.sileo@synapse-fr.com

Abstract

Current state of the art systems in NLP heavily rely on manually annotated datasets, which are expensive to construct. Very little work adequately exploits unannotated data – such as discourse markers between sentences – mainly because of data sparseness and ineffective extraction methods. In the present work, we propose a method to automatically discover sentence pairs with relevant discourse markers, and apply it to massive amounts of data. Our resulting dataset contains 174 discourse markers with at least 10K examples each, even for rare markers such as *coincidentally* or *amazingly*. We use the resulting data as supervision for learning transferable sentence embeddings. In addition, we show that even though sentence representation learning through prediction of discourse markers yields state of the art results across different transfer tasks, it is not clear that our models made use of the semantic relation between sentences, thus leaving room for further improvements. Our datasets are publicly available [1]

1 Introduction

An important challenge within the domain of natural language processing is the construction of adequate semantic representations for textual units – from words over sentences to whole documents. Recently, numerous approaches have been proposed for the construction of vector-based representations for larger textual units, especially sentences. One of the most popular frameworks aims to induce sentence embeddings as an intermediate representation for predicting relations between sentence pairs. For instance, similarity judgements (paraphrases) or inference relations have been used as prediction tasks, and the resulting embeddings perform well in practice, even when the representations are transfered to other semantic tasks (Conneau et al., 2017). However, the kind of annotated data that is needed for such supervised approaches is costly to obtain, prone to bias, and arguably fairly limited with regard to the kind of semantic information captured, as they single out a narrow aspect of the entire semantic content.

Unsupervised approaches have also been proposed, based on sentence distributions in large corpora in relation to their discourse context. For instance, Kiros et al. (2015) construct sentence representations by trying to reconstruct neighbouring sentences, which allows them to take into account different contextual aspects of sentence meaning. In the same vein, Logeswaran et al. (2016) propose to predict if two sentences are consecutive, even though such local coherence can be straightforwardly predicted with relatively shallow features (Barzilay and Lapata, 2008). A more elaborate setting is the prediction of the semantic or rhetorical relation between two sentences, as is the goal of discourse parsing. A number of annotated corpora exist, such as RST-DT (Carlson et al., 2001) and PDTB (Prasad et al., 2008), but in general the available data is fairly limited, and the task of discourse relation prediction is rather difficult. The problem, however, is much easier when there is a marker that makes the semantic link explicit (Pitler et al., 2008), and this observation has often been used in a semi-supervised setting to predict discourse relations in general (Rutherford and Xue, 2015). Building on this observation, one approach to learn sentence representations is to predict such markers or clusters of markers explicitly (Jernite et al., 2017; Malmi et al., 2018; Nie et al., 2017). Consider the following sentence pair:

I live in Paris. But I'm often abroad.

The discourse marker *but* highlights an opposition between the first sentence (the speaker

---

[1] https://github.com/synapse-developpement/Discovery

Proceedings of NAACL-HLT 2019, pages 3477–3486
Minneapolis, Minnesota, June 2 - June 7, 2019.

s1	Paul Prudhomme's Louisiana Kitchen created a sensation when it was published in 1984.
c	happily,
s2'	This family collective cookbook is just as good

Table 1: Sample from our *Discovery* dataset

lives in Paris) and the second sentence (the speaker is often abroad). The marker can thus be straightforwardly used as a label between sentence pairs. In this case, the task is to predict c = *but* (among other markers) for the pair (*I live in Paris*, *I'm often abroad*). Note that discourse markers can be considered as noisy labels for various semantic tasks, such as entailment (c = *therefore*), subjectivity analysis (c = *personally*) or sentiment analysis (c = *sadly*). More generally, discourse markers indicate how a sentence contributes to the meaning of a text, and they provide an appealing supervision signal for sentence representation learning based on language use.

A wide variety of discourse usages would be desirable in order to learn general sentence representations. Extensive research in linguistics has resulted in elaborate discourse marker inventories for many languages.[2] These inventories were created by manual corpus exploration or annotation of small-scale corpora: the largest annotated corpus, the English PDTB consists of a few tens of thousand examples, and provides a list of about 100 discourse markers, organized in a number of categories.

Previous work on sentence representation learning with discourse markers makes use of even more restricted sets of discourse markers, as shown in table 2. Jernite et al. (2017) use 9 categories as labels, accounting for 40 discourse markers in total. It should be noted that the aggregate labels do not allow for any fine-grained distinctions; for instance, the TIME label includes both *now* and *next*, which is likely to impair the supervision. Moreover, discourse markers may be ambiguous; for example *now* can be used to express contrast. On the other hand, Nie et al. (2017) make use of 15 discourse markers, 5 of which are accounting for more than 80% of their training data. In order to ensure the quality of their examples, they only select pairs matching a dependency pattern manually specified for each marker. As such, both of these studies use a restricted or impoverished set of discourse markers; they also both use the BookCorpus dataset, whose size ($4.7M$ sentences that contain a discourse marker, according to Nie et al., 2017) is prohibitively small for the prediction of rare discourse markers.

In this work we use web-scale data in order to explore the prediction of a wide range of discourse markers, with more balanced frequency distributions, along with application to sentence representation learning. We use English data for the experiments, but the same method could be applied to any language that bears a typological resemblance with regard to discourse usage, and has sufficient amounts of textual data available (e.g. German or French). Inspired by recent work (Dasgupta et al., 2018; Poliak et al., 2018; Levy et al., 2018; Glockner et al., 2018) on the unexpected properties of recent manually labelled datasets (e.g. SNLI), we will also analyze our dataset to check whether labels are easy to guess, and whether the proposed model architectures make use of high-level reasoning for their predictions. Our contributions are as follows:

- we propose a simple and efficient method to discover new discourse markers, and present a curated list of 174 markers for English;
- we provide evidence that many connectives can be predicted with only simple lexical features;
- we investigate whether relation prediction actually makes use of the relation between sentences;
- we carry out extensive experiments based on the Infersent/SentEval framework.

2 Discovering discourse markers

2.1 Rationale

Our goal is thus to capture semantic aspects of sentences by means of distributional observations. For our training signal, we aim at something more evolved than just plain contextual co-occurrence,

---

[2] See for instance a sample of language on the Textlink project website: `http://www.textlink.ii.metu.edu.tr/dsd-view`

author	discourse markers / classes	classes	markers
Jernite et al. (2017)	ADDITION, CONTRAST, TIME, RESULT, SPECIFIC, COMPARE, STRENGTH, RETURN, RECOGNIZE	9	40
Nie et al. (2017)	*and, but, because, if, when, before, though, so, as, while, after, still, also, then, although*	15	15
current work	*later, often, understandably, gradually, or, ironically, namely, . . .*	174	174

Table 2: Discourse markers or classes used by previous work on unsupervised representation learning

but simpler than a full-fledged encoder-decoder *à la* Skip-Thought. In that respect, discourse relations are an interesting compromise, if we can reliably extract them in large quantities. This objective is shared with semi-supervised approaches to discourse relation prediction, where automatically extracted explicit instances feed a model targetting implicit instances (Marcu and Echihabi, 2002; Sporleder and Lascarides, 2008; Pitler and Nenkova, 2009; Rutherford and Xue, 2015). In this perspective, it is important to collect unambiguous instances of potential discourse markers. To do so, previous work used heuristics based on specific constructs, especially syntactic patterns for intra-sentential relations, based on a fixed list of manually collected discourse markers. Since we focus on sentence representations, we limit ourselves to discourse arguments that are well-formed sentences, thus also avoiding clause segmentation issues.

Following a heuristic from Rutherford and Xue (2015), also considered by Malmi et al. (2018) and Jernite et al. (2017), we collect pairs of sentences (s_1, s_2) where s_2 starts with marker c. We only consider the case where c is a single word, as detecting longer adverbial constructions is more difficult. We remove c from the beginning of s_2 and call the resulting sentence s_2'. Malmi et al. (2018) make use of a list of the 80 most frequent discourse markers in the PDTB in order to extract suitable sentence pairs. We stay faithful to Rutherford and Xue (2015)'s heuristic, as opposed to Malmi et al. (2018); Jernite et al. (2017): if s_2 starts with c followed by a comma, and c is an adverbial or a conjunction, then it is a suitable candidate. By limiting ourselves to sentences that contain a comma, we are likely to ensure that s_2' is meaningful and grammatical. As opposed to all the cited work mentioned above, we do not restrict the pattern to a known list of markers, but try to collect new reliable cues.

This pattern is decisively restrictive, since discourse markers often appear at the clausal level (e.g. *I did it but now I regret it*). But clauses are not meant to be self contained, and it is not obvious that they should be included in a dataset for *sentence* representation learning. At the same time, one could easily think of cases where c is not a discourse marker, e.g. (s_1, s_2)= ("It's cold.", "Very, very cold."). However, these uses might be easily predicted with shallow language models. In the next section, we use the proposed method for the discovery of discourse markers, and we investigate whether the resulting dataset leads to improved model performance.

2.2 Methodology

We use sentences from the Depcc corpus (Panchenko et al., 2017), which consists of English texts harvested from commoncrawl web data. We sample 8.5 billion consecutive sentence pairs from the corpus. We keep 53% of sentence pairs that contain between 3 and 32 words, have a high probability of being English ($> 75\%$) using FastText langid from Grave et al. (2018), have balanced parentheses and quotes, and are mostly lowercase. We use NLTK (Bird et al., 2009) as sentence tokenizer and NLTK PerceptronTagger as part of speech tagger for adverb recognition. In addition to our automatically discovered candidate set, we also include all (not necessarily adverbial) PDTB discourse markers that are not induced by our method. Taking this into account, 3.77% of sentence pairs contained a discourse marker candidate, which is about 170M sentence pairs. An example from the dataset is shown in table 1. We only keep pairs in which the discourse marker occurs at least 10K times. We also subsample pairs so that the maximum occurrence count of a discourse marker is 200K. The resulting dataset con-

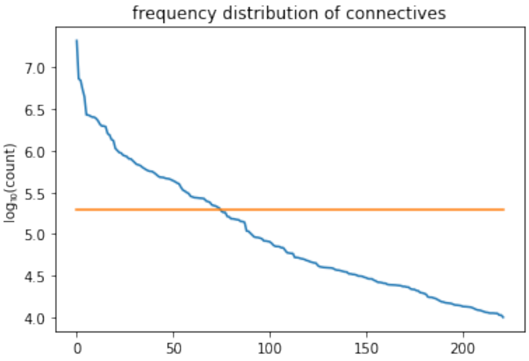


Figure 1: Frequency distribution of candidate discourse markers; the horizontal line indicates the subsampling threshold.

tains 19M pairs.

We discovered 243 discourse marker candidates. Figure 1 shows their frequency distributions. As expected, the most frequent markers dominate the training data, but when a wide range of markers is included, the rare ones still contribute up to millions of training instances. Out of the 42 single word PDTB markers that precede a comma, 31 were found by our rule. Some markers are missing because of NLTK errors, which mainly result from morphological issues.[3]

2.3 Controlling for shallow features

As previously noted, some candidates discovered by our rule may not be actual discourse markers. In order to discard them, we put forward the hypothesis that actual discourse markers cannot be predicted with shallow lexical features. Inspired by Gururangan et al. (2018), we use a Fasttext classifier (Joulin et al., 2016) in order to predict c from s_2'. The Fasttext classifier predicts labels from an average of word embeddings fed to a linear classifier. We split the dataset in 5 folds, and we predict markers for each fold, while training on the remaining folds. We use a single epoch, randomly initialized vectors of size 100 (that can be unigrams, bigrams or trigrams) and a learning rate of 0.5.

In addition, we predict c from the concatenation of s_1 and s_2' (using separate word representations for each case). One might assume that the prediction of c in this case relies on the interaction between s_1 and s_2; however, the features of s_1 and s_2 within Fasttext's setup only interact additively, which means that the classification most likely relies on individual cues in the separate sentences, rather than on their combination. In order to test this hypothesis, we introduce a *random shuffle* operation: for each example (s_1, s_2', c), s_2' is replaced by a random sentence from a pair that is equally linked by c (we perform this operation separately in train and test sets).

Table 3 indicates that shallow lexical features indeed yield relatively high prediction rates. Moreover, the shuffle operation indeed increases accuracy, which corroborates the hypothesis that classification with shallow features relies on individual cues from separate sentences, rather than their combination.

features	accuracy (%)
majority rule	1.2
s2	18.6
s1-s2'	21.9
s1-s2' (shuffled)	24.8

Table 3: Accuracy when predicting candidate discourse markers using shallow lexical features

Tables 4 and 5 show the least and most predictable discourse markers, and the corresponding recognition rate with lexical features.

candidate marker	accuracy (%)
evidently,	0.0
frequently,	0.0
meantime,	0.0
truthfully,	0.0
supposedly,	0.1

Table 4: Candidate discourse markers that are the most difficult to predict from shallow features

candidate marker	accuracy (%)
defensively,	65.5
afterward	71.1
preferably,	71.9
this,	72.7
very,	90.7

Table 5: Candidate discourse markers that are the easiest to predict from shallow features. This shows candidates that are unlikely to be interesting discourse cues.

---

[3] For instance, *lovely* is tagged as an adverb because of its suffix, while *besides* was never tagged as an adverb

Interestingly, the two most predictable candidates are not discourse markers. Upon inspection of harvested pairs, we noticed that even legitimate discourse markers can be guessed with relatively simple heuristics in numerous examples. For example, $c =$ *thirdly* is very likely to occur if s_1 contains *secondly*. We use this information to optionally filter out such simple instances, as described in the next section.

2.4 Dataset variations

In the following, we call our method *Discovery*. We create several variations of the sentence pairs dataset. In *DiscoveryHard*, we remove examples where the candidate marker was among the top 5 predictions in our Fasttext shallow model and keep only the 174 candidate markers with a frequency of at least $10k$. Instances are then sampled randomly so that each marker appears exactly $10k$ times in the dataset.

Subsequently, the resulting set of discourse markers is also used in the other variations of our dataset. *DiscoveryBase* designates the dataset for which examples predicted with the Fasttext model were not removed. In order to measure the extent to which the model makes use of the relation between s_1 and s_2', we also create a *DiscoveryShuffled* dataset, which is the *DiscoveryBase* dataset subjected to the *random shuffle* operation described previously. To isolate the contribution of our discovery method, the dataset *DiscoveryAdv* discards all discourse markers from PDTB that were not found by our method. Also, in order to measure the impact of label diversity, *Discovery10* uses $174k$ examples for each of the 10 most frequent markers,[4] thus totalling as many instances as *DiscoveryBase*. Finally, *DiscoveryBig* contains almost twice as many instances as *DiscoveryBase*, i.e. $20k$ instances for each discourse marker (although, for a limited number of markers, the number of instances is slightly lower due to data sparseness).

3 Evaluation of sentence representation learning

3.1 Setup

Our goal is to evaluate the effect of using our various training datasets on sentence encoding, given encoders of equivalent capacity and similar setups. Thus, we follow the exact setup of Infersent (Conneau et al., 2017), also used in the Dissent (Malmi et al., 2018) model: we learn to encode sentences into h with a bi-directional LSTM sentence encoder using element-wise max pooling over time. The dimension size of h is 4096. Word embeddings are fixed GloVe embeddings with 300 dimensions, trained on Common Crawl 840B.[5] A sentence pair (s_1, s_2) is represented with $[h_1, h_2, h_1 \odot h_2, |h_2 - h_1|]$,[6] which is fed to a softmax in order to predict a marker c. Our datasets are split in 90% train, 5% validation, and 5% test. Optimization is done with SGD (learning rate is initialized at 0.1, decayed by 1% at each epoch and by 80% if validation accuracy decreases; learning stops when learning rate is below 10^{-5} and the best model on training task validation loss is used for evaluation; gradient is clipped when its norm exceeds 5). Once the sentence encoder has been trained on a base task, the resulting sentence embeddings are tested with the SentEval library (Conneau et al., 2017).

We evaluate the different variations of our dataset we described above in order to analyze their effect, and compare them to a number of existing models. Table 7 displays the tasks used for evaluation. For further analysis, table 9 displays the result of *Linguistic Probing* using the method by Conneau et al. (2018). Although these tasks are primarily designed for understanding the content of embeddings, they also focus on aspects that are desirable to perform well in general semantic tasks (e.g. prediction of tense, or number of object).

3.2 Results

Table 6 gives an overview of transfer learning evaluation, also comparing to other supervised and unsupervised approaches. Note that we outperform DisSent on all tasks except TREC[7] with less than half the amount of training examples. In addition, our approach is arguably simpler and faster.

MTL (Subramanian et al., 2018) only achieves stronger results than our method on the MRPC and SICK tasks. The MTL model uses $124M$ training examples with an elaborate multi-task setup, training on $45M$ sentences with manual translation, $1M$ pairs from SNLI/MNLI, $4M$ parse trees of sentences, and $74M$ consecutive sentence pairs.

---

[4] They are: *however, hence, moreover, additionally, nevertheless, furthermore, alternatively, again, next, therefore*

[5] https://nlp.stanford.edu/projects/glove/

[6] $h_1 \odot h_2 = (h_{11}.h_{21}, .., h_{1i}.h_{2i}, ...)$

[7] This dataset is composed of questions only, which are underrepresented in our training data.

	N	MR	CR	SUBJ	MPQA	SST2	TREC	SICK-R	SICK-E	MRPC	AVG
InferSent	1.0	81.1	86.3	92.4	90.2	84.6	88.2	88.4	86.1	76.2	85.9
MTL	124	82.5	**87.7**	94	90.9	83.2	93	**88.8**	**87.8**	**78.6**	**87.4**
SkipThought	74	76.5	80.1	93.6	87.1	82	92.2	85.8	82.3	73	83.6
QuickThought	174	81.3	84.5	**94.6**	89.5	-	92.4	87.1	-	75.9	-
DisSent	4.7	80.1	84.9	93.6	90.1	84.1	**93.6**	84.9	83.7	75	85.6
DiscoveryBase	1.7	82.5	86.3	94.2	90.5	85.2	91.8	85.7	84	75.8	86.2
DiscoveryHard	1.7	81.6	86.5	93.9	90.5	84.8	90	85.4	83.2	76.5	85.8
Discovery10	1.7	81.2	85.1	93.7	90.2	83	90	85.9	83.8	75.8	85.4
DiscoveryAdv	1.4	81.4	85.8	93.8	90.5	83.4	92	86	84.3	75.7	85.9
DiscoveryShuffled	1.7	81.4	86.1	94.1	90.9	**85.3**	90.4	85.6	83.6	75.4	85.9
DiscoveryBig	3.4	**82.6**	87.4	94.5	**91.0**	85.2	93.4	86.4	84.8	76.6	86.9

Table 6: SentEval evaluation results with our models trained on various datasets. The first two models are supervised, the other ones unsupervised. All scores are accuracy percentages, except SICK-R, which is Pearson correlation percentage. InferSent is from Conneau et al. (2017), MTL is the multi-task learning based model from Subramanian et al. (2018). Evaluation tasks are described in table 7, and N denotes the number of examples for each dataset (in millions). Dissent is from Nie et al. (2017), QuickThought is from Logeswaran and Lee (2018) with fixed embeddings configuration. The best result per task appears in bold, the best result for unsupervised setups is underlined.

The model also fine-tunes word embeddings in order to achieve a higher capacity. It is therefore remarkable that our model outperforms it on many tasks. Besides, MTL is not a direct competitor to our approach since its main contribution is its multi-task setup, and it could benefit from using our training examples.

Our best model rivals (and indeed often outperforms) QuickThought on all tasks, except relatedness (SICK-R). QuickThought's training task is to predict whether two sentences are contiguous, which might incentivize the model to perform well on a relatedness task. We also outperform InferSent on many tasks except entailment and relatedness. Entailment prediction is the explicit training signal for Infersent.

To help the analysis of our different model variations, table 8 displays the test scores on each dataset for the original training task. It also shows the related PDTB implicit relation prediction scores. The PDTB is annotated with a hierarchy of relations, with 5 classes at level 1 (including the EntRel relation), and 16 at level 2 (with one relation absent from the test). It is interesting to see that this form of simple semi-supervised learning for implicit relation prediction performs quite well, especially for fine-grained relations, as the best model slightly beats the best current dedicated model, listed at 40.9% in Xue et al. (2017).

DiscoveryHard scores lower on its training task than *DiscoveryBase*, and it also performs worse on transfer learning tasks. This makes sense, since lexical features are important to solve the evaluation tasks. Our initial hypothesis was that more difficult instances might force the model to use higher-level reasoning, but this does not seem to be the case. More surprisingly, preventing the encoders to use the relationship between sentences, as in *DiscoveryShuffled*, does not substantially hurt the transfer performance, which remains on average higher than Nie et al. (2017). Additionally, our models score well on linguistic probing tasks. They outperform Infersent on all tasks, which seems to contradict the claim that SNLI data allows for learning of *universal* sentence representations (Conneau et al., 2017). And a final interesting outcome is that the diversity of markers (e.g. using *DiscoveryBase* instead of *Discovery10*) seems to be important for good performance on those tasks, since *Discovery10* has the worst overall performance on average.

name	N	task	C
MR	11k	sentiment (movie reviews)	2
CR	4k	sentiment (product reviews)	2
SUBJ	10k	subjectivity/objectivity	2
MPQA	11k	opinion polarity	2
TREC	6k	question-type	6
SST	70k	sentiment (movie reviews)	2
SICK-E	10k	entailment	3
SICK-R	10k	relatedness	3
MRPC	4k	paraphrase detection	2
$PDTB_5$	17k	implicit discourse relation (coarse)	5
$PDTB_{16}$	17k	implicit discourse relation (fine)	15

Table 7: Transfer evaluation tasks. N is the number of training examples and C is number of classes for each task.

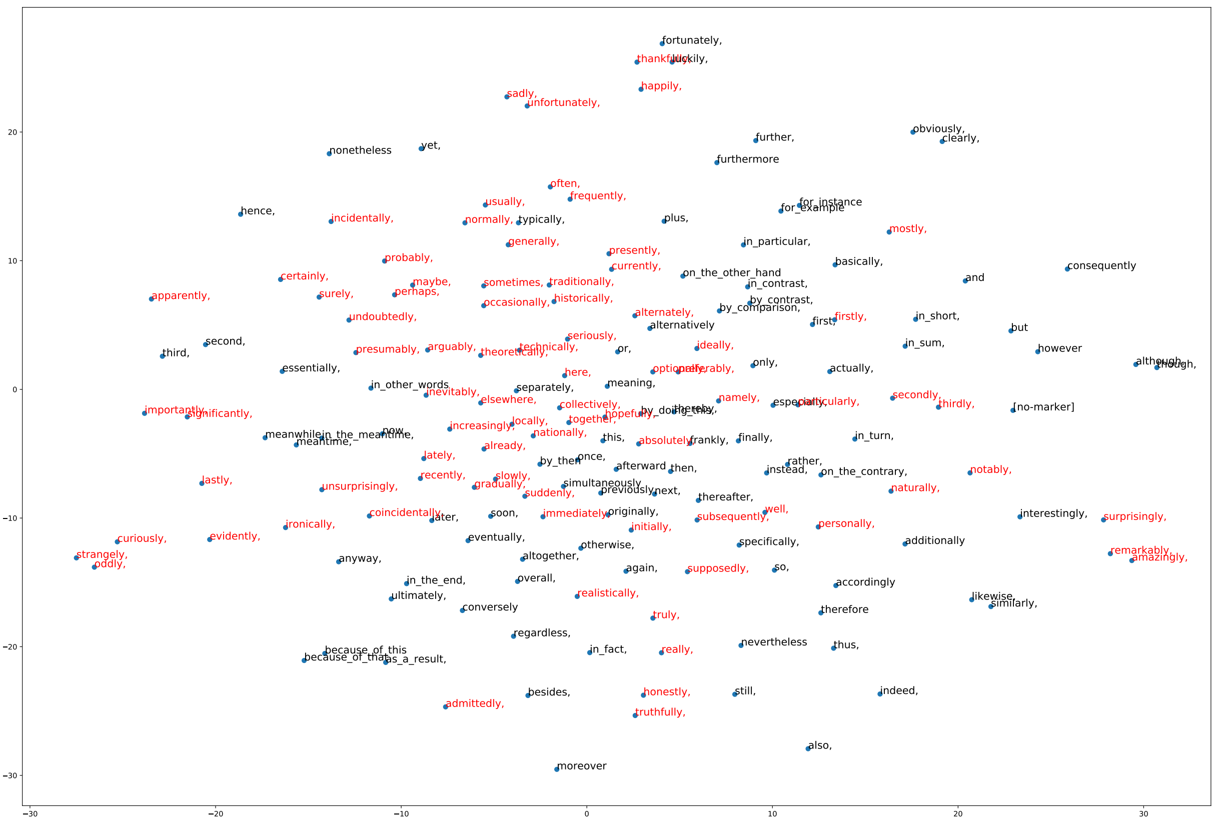


Figure 2: TSNE visualization of the softmax weights from our *DiscoveryBig* model for each discourse marker, after unit norm normalization. Markers discovered by our method (e.g. absent from PDTB annotations) are colored in red.

	PDTB$_5$ coarse	PDTB$_{16}$ fine	$\mathcal{T}$
InferSent	46.7	34.2	-
DisSent	48.9	36,9	-
DiscoveryBase	**52.5**	40.0	20.6
DiscoveryHard	50.7	39.8	9.3
Discovery10	48.3	37.7	51.9
DiscoveryAdv	49.7	37.6	26.1
DiscoveryShuffled	51.0	39.5	11.5
DiscoveryBig	51.3	**41.3**	22.2

Table 8: Test results (accuracy) on implicit discursive relation prediction task (PDTB relations level 1 and 2, i.e coarse-grained and fine-grained) and training tasks $\mathcal{T}$. Note that scores for $\mathcal{T}$ are not comparable since the test set changes for each version of the dataset.

3.3 Visualisation

The softmax weights learned during the training phase can be interpreted as embeddings for the markers themselves, and used to visualize their relationships. Figure 2 shows a TSNE (van der Maaten and Hinton, 2008) plot of the markers' representations. Proximity in the feature space seems to reflect semantic similarity (e.g. *usually/normally*). In addition, the markers we discovered, colored in red, blend with the PDTB markers (depicted in black). It would be interesting to cluster markers in order to empirically define discourse relations, but we leave this for future work.

4 Related work

Though discourse marker prediction in itself is an interesting and useful task (Malmi et al., 2017), discourse markers have often been used as a training cue in order to improve implicit relation prediction (Marcu and Echihabi, 2001; Sporleder and Lascarides, 2005; Zhou et al., 2010; Braud and Denis, 2016). This approach has been extended to general representation learning by Jernite et al. (2017)—although with empirically unconvincing results, which might be attributed to an inappropriate training/evaluation set-up, or the use of a limited number of broad categories instead of actual discourse markers. Nie et al. (2017) used the more standard InferSent framework and obtained better results, although they were still outperformed by QuickThought (Logeswaran and Lee, 2018),

	BShift	CoordInv	Depth	ObjNum	SubjNum	OddM	Tense	TC	WC	AVG
InferSent	56.5	65.9	37.5	79.9	84.3	53.2	87	78.1	95.2	70.8
SkipThought	**69.5**	69	39.6	83.2	86.2	54.5	**90.3**	82.1	79.6	72.7
QuickThought	56.8	70	40.2	79.7	83	55.3	86.2	80.7	90.3	71.4
DiscoveryBase	63.1	70.6	45.2	83.8	87.2	57.3	89.1	83.2	94.7	74.9
DiscoveryHard	62.7	70.4	44.5	83.4	**88.1**	57.3	89.5	82.8	94.1	74.8
Discovery10	61.3	69.7	42.9	81.8	86.7	55.8	87.8	81.4	**96.1**	73.7
DiscoveryAdv	61.5	70	43.9	82.6	86.2	56.2	89.1	82.8	**96.1**	74.3
DiscoveryShuffled	62.6	**71.4**	45.3	**84.3**	88	**58.3**	89.3	82.8	93.4	75
DiscoveryBig	63.3	**71.4**	**46.0**	84.1	87.8	57.1	89.4	**84.2**	96	**75.5**

Table 9: Accuracy of various models on linguistic probing tasks using logistic regression on SentEval. BShift is detection of token inversion. CoordInv is detection of clause inversion. ObjNum/SubjNum is prediction of the number of object resp. subject. Tense is prediction of the main verb tense. Depth is prediction of parse tree depth. TC is detection of common sequences of constituents. WC is prediction of words contained in the sentence. OddM is detection of random replacement of verbs/nouns by other verbs/nouns. AVG is the average score of those tasks for each model. For more details see Conneau et al. (2018). SkipThought and Infersent results come from Perone et al. (2018), QuickThought results come from Brahma (2018).

which uses a much simpler training task. Both of these rely on pre-established lists of discourse markers provided by the PDTB, and both perform a manual annotation for each marker—Nie et al. (2017) uses dependency patterns, while Jernite et al. (2017) uses broad discourse categories. Our work is the first to automatically discover discourse markers from text.

More generally, various automatically extracted training signals have been used for unsupervised learning tasks. Hashtags (Felbo et al., 2017) have been sucessfully exploited in order to learn sentiment analysis from unlabelled tweets, but their availability is mainly limited to the microblogging domain. Language modeling provides a general training signal for representation learning, even though there is no obvious way to derive sentence representations from language models. BERT (Devlin et al., 2018) currently holds the best results in transfer learning based on language modeling, but it relies on sentence pair classification in order to compute sentence embeddings, and it makes use of a simple sentence contiguity detection task (like QuickThought); this task does not seem challenging enough since BERT reportedly achieves 98% detection accuracy. Phang et al. (2018) showed that the use of SNLI datasets yields significant gains for the sentence embeddings from Radford (2018), which are based on language modeling.

For the analysis of our models, we draw inspiration from critical work on Natural Language Inference datasets (Dasgupta et al., 2018; Levy et al., 2018). Gururangan et al. (2018); Poliak et al. (2018) show that baseline models that disregard the hypothesis yield good results on SNLI, which suggests that the model does not perform the high level reasoning we would expect in order to predict the correct label. They attribute this effect to bias in human annotations. In this work, we show that this issue is not inherent to human labeled data, and propose the *shuffle* perturbation in order to measure to what extent the relationship between sentences is used.

5 Conclusion

In this paper, we introduce a novel and efficient method to automatically discover discourse markers from text, and we use the resulting set of candidate markers for the construction of an extensive dataset for semi-supervised sentence representation learning. A number of dataset variations are evaluated on a wide range of transfer learning tasks (as well as implicit discourse recognition) and a comparison with existing models indicates that our approach yields state of the art results on the bulk of these tasks. Additionally, our analysis shows that removing 'simple' examples is detrimental to transfer results, while preventing the model to exploit the relationship between sentences has a negligible effect. This leads us to believe that, even though our approach reaches state of the art results, there is still room for improvement: models that adequately exploit the relationship between sentences would be better at leveraging the supervision of our dataset, and could yield even better sentence representations. In future work, we also aim to increase the coverage of our method. For instance, we can make use of more lenient patterns that capture an even wider range of discourse markers, such as multi-word markerse.

References

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, 34(1):1–34.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, volume 43.

Siddhartha Brahma. 2018. Unsupervised learning of sentence representations using sequence consistency. *CoRR*, abs/1808.04217.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 203–213, Austin, Texas. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurovsky. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the SIGDIAL 2001 Workshop, The 2nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, Saturday, September 1, 2001 to Sunday, September 2, 2001, Aalborg, Denmark*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating Compositionality in Sentence Embeddings. (2011).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bjarke Felbo, Alan Mislove, Anders Sogaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, (3):1–6.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. Cite arxiv:1607.01759.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Omer Levy, Samuel R Bowman, and Noah A Smith. 2018. Annotation Artifacts in Natural Language Inference Data. *Proceedings of NAACL-HLT 2018*, pages 107–112.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. pages 1–16.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2016. Sentence Ordering using Recurrent Neural Networks. pages 1–15.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Eric Malmi, Daniele Pighin, Sebastian Krause, and Mikhail Kozhevnikov. 2017. Automatic Prediction of Discourse Connectives.

Eric Malmi, Daniele Pighin, Sebastian Krause, and Mikhail Kozhevnikov. 2018. Automatic Prediction of Discourse Connectives. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Daniel Marcu and Abdessamad Echihabi. 2001. An unsupervised approach to recognizing discourse relations. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, (July):368.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368–375.

Allen Nie, Erin D. Bennett, and Noah D. Goodman. 2017. DisSent: Sentence Representation Learning from Explicit Discourse Relations.

Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone P Ponzetto, and Chris Biemann. 2017. Building a Web-Scale Dependency-Parsed Corpus from Common Crawl. pages 1816–1823.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore, Short Papers*, pages 13–16.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90. Coling 2008 Organizing Committee.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics*, (1):180–191.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). Http://www.lrec-conf.org/proceedings/lrec2008/.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 799–808.

Caroline Sporleder and Alex Lascarides. 2005. Exploiting linguistic cues to classify rhetorical relations. In *Proceedings of Recent Advances in Natural Langauge Processing (RANLP)*, Bulgaria.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: an assessment. *Natural Language Engineering*, 14(3):369–416.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *International Conference on Learning Representations*.

Nianwen Xue, Vera Demberg, and Attapol Rutherford. 2017. A systematic study of neural discourse models for implicit discourse relation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 281–291.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Coling 2010: Posters*, pages 1507–1514. Coling 2010 Organizing Committee.

How Large a Vocabulary Does Text Classification Need? A Variational Approach to Vocabulary Selection

Wenhu Chen†, **Yu Su**¶, **Yilin Shen**‡, **Zhiyu Chen**†, **Xifeng Yan**†, **William Wang**†
University of California, Santa Barbara†
Ohio State University, Columbus¶
Samsung Research, Moutain View‡
{wenhuchen,zhiyuchen,xyan,william}@cs.ucsb.edu
su.809@osu.edu yilin.shen@samsung.com

Abstract

With the rapid development in deep learning, deep neural networks have been widely adopted in many real-life natural language applications. Under deep neural networks, a predefined vocabulary is required to vectorize text inputs. The canonical approach to select predefined vocabulary is based on the word frequency, where a threshold is selected to cut off the long tail distribution. However, we observed that such a simple approach could easily lead to under-sized vocabulary or oversized vocabulary issues. Therefore, we are interested in understanding how the end-task classification accuracy is related to the vocabulary size and what is the minimum required vocabulary size to achieve a specific performance. In this paper, we provide a more sophisticated variational vocabulary dropout (VVD) based on variational dropout to perform vocabulary selection, which can intelligently select the subset of the vocabulary to achieve the required performance. To evaluate different algorithms on the newly proposed vocabulary selection problem, we propose two new metrics: Area Under Accuracy-Vocab Curve and Vocab Size under X% Accuracy Drop. Through extensive experiments on various NLP classification tasks, our variational framework is shown to significantly outperform the frequency-based and other selection baselines on these metrics.

1 Introduction

Over the past decade, deep neural networks have become arguably the most popular model choice for a vast number of natural language processing (NLP) tasks and have constantly been delivering state-of-the-art results. Because neural network models assume continuous data, to apply a neural network on any text data, the first step is to vectorize the discrete text input with a word embedding matrix through look-up operation, which in turn assumes a pre-defined vocabulary set. For many NLP tasks, the vocabulary size can easily go up to the order of tens of thousands, which potentially makes the word embedding the largest portion of the trainable parameters. For example, a document classification task like AG-news (Zhang et al., 2015) can include up to 60K unique words, with the embedding matrix accounting for 97.6% of the trainable parameters (Table 1), which leads to under-representation of the neural networks' own parameters.

Cutoff Freq.	Vocab	Remain Vocab	#Emb	#CNN	#Emb Ratio
1	60K	100%	15M	0.36M	97.6%
5	40K	21.7%	10M	0.36M	95.6%
10	24K	13%	6M	0.36M	94.3%
20	14K	9.4%	3.5M	0.36M	90%
100	4K	2.7%	1M	0.36M	73%

Table 1: Illustration of the frequency-based vocabulary selection heuristic on a typical CNN-based document classification model (Section 4.1). #Emb is the number of parameters in the word embedding matrix (256 dimensions), and #CNN is that in the CNN model.

Intuitively, using the full or very large vocabulary are neither economical, as it limits model applicability on computation- or memory-constrained scenarios (Yogatama et al., 2015; Faruqui et al., 2015), nor necessary, as many words may contribute little to the end task and could have been safely removed from the vocabulary. Therefore, how to select the best vocabulary is a problem of both theoretical and practical interests. Somewhat surprisingly, this *vocabulary selection* problem is largely under-addressed in the literature: The *de facto* standard practice is to do

Proceedings of NAACL-HLT 2019, pages 3487–3497
Minneapolis, Minnesota, June 2 - June 7, 2019.

frequency-based cutoff (Luong et al., 2015; Kim, 2014), and only retain the words more frequent than a certain threshold (Table 1). Although this simple heuristic has demonstrated strong empirical performance, its task-agnostic nature implies that likely it is not the optimal strategy for many tasks (or any task). Task-aware vocabulary selection strategies and a systematic comparison of different strategies are still lacking.

In this work, we present the first systematic study of the vocabulary selection problem. Our study will be based on text classification tasks, a broad family of NLP tasks including document classification (DC), natural language inference (NLI), natural language understanding in dialog systems (NLU), etc. Specifically, we aim to answer the following questions:

1. *How important a role does the vocabulary selection algorithm play in text classification?*
2. *How to dramatically reduce the vocabulary size while retaining the accuracy?*

The rest of the paper is organized as follows: We first formally define the vocabulary selection problem (subsection 2.1) and present a quantitative study on classification accuracy with different vocabulary selections to showcase its importance in the end task (subsection 2.2). We also propose two new metrics for evaluating the performance of vocabulary selection in text classification tasks (subsection 2.3). We then propose a novel, task-aware vocabulary selection algorithm called *Varitional Vocabulary Dropout (VVD)* (section 3) which draws on the idea of variational dropout (Kingma et al., 2015): If we learn a dropout probability p_w for each given word w in the vocabulary $\mathbb{V}$ during the model training on a given task, the learned dropout probabilities p_w will imply the importance of word w to the end task and can, therefore, be leveraged for vocabulary selection. We propose to infer the latent dropout probabilities under a Bayesian inference framework. During test time, we select the sub vocabulary $\hat{\mathbb{V}}$ by only retaining words with dropout probability lower than a certain threshold. For any words deselected using VVD, we will simply regard them as a special token with null vector representation $[0, 0, \cdots, 0]$. Please note that our proposed algorithm needs to re-train a word embedding matrix, thus it is tangential to the research of pre-trained word embedding like Word2Vec (Mikolov et al., 2013) or Glove (Pennington et al., 2014) though we can use them to initialize our embedding.

We conduct comprehensive experiments to evaluate the performance of VVD (section 4) on different end classification tasks. Specifically, we compare against an array of strong baseline selection algorithms, including the frequency-based algorithm (Luong et al., 2015), TF-IDF algorithm (Ramos et al., 2003), and structure lasso algorithm (Friedman et al., 2010), and demonstrate that it can consistently outperform these competing algorithms by a remarkable margin. To show that the conclusions are widely held, our evaluation is based on a wide range of text classification tasks and datasets with different neural networks including Convolutional Neural Network (CNN) (Kim, 2014), Bi-directional Long-Short Term Memory (BiLSTM) (Bahdanau et al., 2014) and Enhanced LSTM (ESIM) (Chen et al., 2017). In summary, our contributions are three-fold:

1. We formally define the vocabulary selection problem, demonstrate its importance, and propose new evaluation metrics for vocabulary selection in text classification tasks.
2. We propose a novel vocabulary selection algorithm based on variational dropout by re-formulating text classification under the Bayesian inference framework. The code will be released in Github[1].
3. We conduct comprehensive experiments to demonstrate the superiority of the proposed vocabulary selection algorithm over a number of strong baselines.

2 Vocabulary Selection

2.1 Problem Definition

We now formally define the problem setting and introduce the notations for our problem. Conventionally, we assume the neural classification model vectorizes the discrete language input into a vector representation via an embedding matrix $W \in \mathbb{R}^{V*D}$, where V denotes the size of the vocabulary, and D denotes the vector dimension. The embedding is associated with a pre-defined word-to-index dictionary $\mathbb{V} = \{w_i : i | 1 \leq i \leq V\}$ where w_i denotes a literal word corresponding to i_{th} row in the embedding matrix. The embedding matrix W covers the subset of a vocabulary of interests for a particular NLP task, note that the value of V

---

[1] `https://github.com/wenhuchen/Variational-Vocabulary-Selection.git`

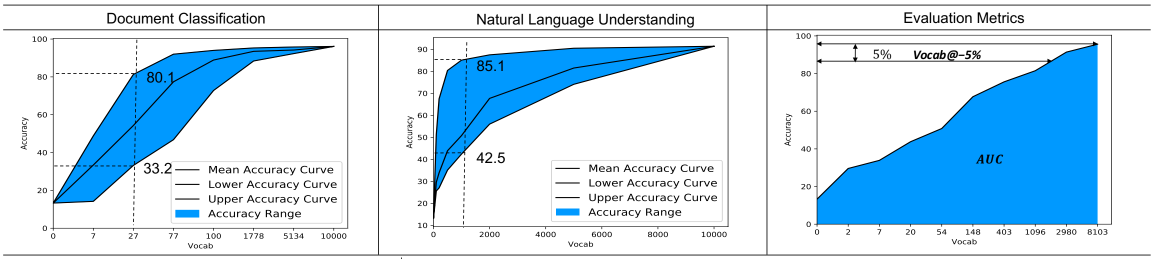


Figure 1: Monte-Carlo simulation on vocabulary selection. Left: CNN-based document classification on AG-news dataset. Middle: Natural language understanding on Snips dataset. Right: Metrics for vocabulary selection.

is known to be very large due to the rich variations in human languages. Here we showcase the embedding matrix size of a popular text classification model[2] on AG-news dataset (Zhang et al., 2015) in Table 1. From which we can easily observe that the embedding matrix is commonly occupying most of the parameter capacity, which could be the bottleneck in many real-world applications with limited computation resources.

In order to alleviate such redundancy problem and make embedding matrix as efficient as possible, we are particularly interested in discovering the minimum row-sized embedding $\hat{W}$ to achieve nearly promising performance as using the full row-sized embedding W. More formally, we define the our problem as follows:

$$\begin{aligned} &\underset{\hat{W},\hat{\theta}}{\operatorname{argmin}} \#Row(\hat{W}) \\ s.t. \quad & Acc(f_{\hat{\theta}}(x;\hat{W}),y) - Acc(f_{\theta}(x;W),y) \leq \epsilon \end{aligned} \tag{1}$$

where #Row is a the number of rows in the matrix $\hat{W}$, f_θ is the learned neural model with parameter θ to predict the class given the inputs x, Acc is the function which measure accuracy between model prediction and y (reference output), and ϵ is the tolerable performance drop after vocabulary selection. It is worth noting that here θ includes all the parameter set of the neural network except embedding matrix W. For each vocabulary selection algorithm $\mathcal{A}$, we propose to draw its characteristic curve $Acc(f_{\hat{\theta}}(x;\hat{W}),y) = g_{\mathcal{A}}(\#Row(\hat{W}))$ to understand the relationship between the vocabulary capacity and classification accuracy, which we call as (characteristic) accuracy-vocab curve throughout our paper.

2.2 Importance of Vocabulary Selection

In order to investigate the importance of the role played by the vocabulary selection algorithm, we design a Monte-Carlo simulation strategy to approximate accuracy's lower bound and upper bound of a given vocabulary size reached by a possible selection algorithm $\mathcal{A}$. More specifically, for a given vocabulary size of $\hat{V}$, there exist $\binom{V}{\hat{V}}$ algorithms which can select distinct vocabulary subset $\hat{\mathbb{V}}$ from the full vocabulary $\mathbb{V}$. Directly enumerating these possibilities are impossible, we instead propose to use a Monte-Carlo vocabulary selection strategy which can randomly pick vocabulary subset $\hat{\mathbb{V}}$ to simulate the possible selection algorithms by running it N times. After simulation, we obtain various point estimations $(Acc_1, \cdots, Acc_N|\hat{V})$ at each given $\hat{V}$ and depict the point estimates in Figure 1 to approximately visualize the upper and lower bound of the accuracy-vocab curve. From Figure 1, we can easily observe that the accuracy range under a limited-vocabulary is extremely large, when the budget $\hat{V}$ increases, the gap gradually shrinks. For example, for document classification with a budget of 1000, a selection algorithm $\mathcal{A}$ can yield a potential accuracy ranging from 42.5 to 85.1, while for natural language understanding task with a budget of 27, a selection algorithm $\mathcal{A}$ can yield a potential accuracy ranging from 33.2 to 80.1. Such a Monte-Carlo simulation study has demonstrated the significance of vocabulary selection strategy in NLP tasks and also implicate the enormous potential of an optimal vocabulary selection algorithm.

2.3 Evaluation Metrics

In order to evaluate how well a given selection algorithm $\mathcal{A}$ performs, we propose evaluation metrics as depicted in Figure 1 by quantitatively studying its characteristic accuracy-vocab curve. These metrics namely Area Under Curve (AUC) and Vocab@-X% separately measure the vocabulary selection performance globally and locally.

---

[2] `https://github.com/dennybritz/cnn-text-classification-tf`

Specifically, AUC computes enclosed area by the curve, which gives an overview of how well the vocabulary selection algorithm performs. In comparison, Vocab@-X% computes the minimum vocabulary size required if X% performance drop is allowed, which straightforwardly represents how large vocabulary is required to achieve a given accuracy. For the local evaluation metric, we mainly consider Vocab@-3% and Vocab@-5%. However, we observe that directly computing AUC lays too much emphasis on the large-vocabulary region, thus unable to represent an algorithm's selection capability under the low-vocabulary conditions. Therefore, we propose to take the logarithm of the vocabulary size and then compute the normalized enclosed area by:

$$AUC = \frac{\int_{\hat{V}} Acc(\log(\hat{V})) d \log(\hat{V})}{\int_{\hat{V}} Acc(V) d \log(\hat{V})} \tag{2}$$

It is worth noting that Vocab@-X% takes value from range $[0, V]$ with smaller values indicate better performance. Since AUC is normalized by Acc(V), it takes value from range $[0, 1]$ regardless of the classification error.

3 Our Method

Inspired by DNN dropout (Srivastava et al., 2014; Wang and Manning, 2013), we propose to tackle the vocabulary selection problem from word-level dropout perspective, where we assume each word w_i (an integer index) is associated with its characteristic dropout rate p_i, which represents the probability of being replaced with an empty placeholder, specifically, higher dropout probability indicates less loss suffered from removing it from the vocabulary. Hence, the original optimization problem in Equation 1 can be thought of as inferring the latent dropout probability vector $\mathbf{p} = [p_1, \cdots, p_V]$. The overview of our philosophy is depicted in Figure 2, where we associate with each row of the embedding matrix a dropout probability and then re-train the complete system, which grasps how much contribution each word from the vocabulary makes to the end NLP task and remove those "less contributory" words from the vocabulary without hurting the performance.

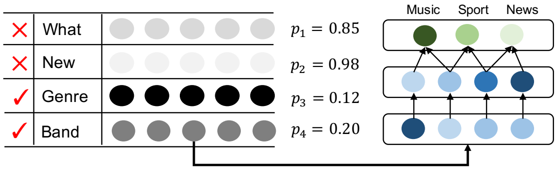


Figure 2: Variational dropout in classification models, "New" and "What" can be safely removed without harming performance due to large dropout probability.

3.1 Bernouli Dropout

Here we first assume that the neural network vectorizes the discrete inputs with an embedding matrix W to project given words x into vector space $\mathbb{R}^D$, and then propose to add random dropout noise into the embedding input to simulate the dropout process as follows:

$$E(x|\mathbf{b}) = (\mathbf{b} \odot OneHot(x)) \cdot W \tag{3}$$

where $OneHot$ is a function to transform a word x into its one-hot form $OneHot(x) \in \mathbb{R}^V$, and $\mathbf{b} \in \mathbb{R}^V$ is the Bernouli dropout noise with $b_i \sim Bern(1 - p_i)$. The embedding output vector $E(x|\mathbf{b})$ is computed with a given embedding matrix W under a sampled Bernouli vector $\mathbf{b}$. In order to infer the latent Bernouli distribution with parameters $\mathbf{p}$ under the Bayesian framework where training pairs $(\mathbf{x} = x_1 \cdots x_n, y)$ are given as the evidence, we first define an objective function as $\mathcal{L}(f_\theta(\mathbf{x}), y)$ and then derive its lower bound as follows (with $\bar{\mathbf{p}} = 1 - \mathbf{p}$):

$$\begin{aligned}
&\log \mathcal{L}(f_\theta(\mathbf{x}), y) = \log \int_{\mathbf{b}} \mathcal{L}(f_\theta(E(\mathbf{x}|\mathbf{b})), y) \mathcal{P}(\mathbf{b}) d\mathbf{b} \\
\geq &\mathop{\mathbb{E}}_{\mathbf{b} \sim Bern(\bar{\mathbf{p}})} [\log \mathcal{L}(f_\theta(E(\mathbf{x}|\mathbf{b})), y)] - KL(Bern(\bar{\mathbf{p}}) || \mathcal{P}(\mathbf{b})) \\
= &\mathcal{L}(W; \theta)
\end{aligned}$$

where $\mathcal{P}(\mathbf{b})$ is the prior distribution, and $Bern(\bar{\mathbf{p}})$ denotes the Bernouli approximate posterior with parameter $\mathbf{p}$. Here we use $E(\mathbf{x})$ as the simplied form of $\{E(x_1), \cdots, E(x_n)\}$, we separate the text classification model's parameters θ with the embedding parameters W and assume the classification model f_θ directly takes embedding E as input.

3.2 Gaussian Relaxation

However, the Bernouli distribution is hard to reparameterize, where we need to enumerate 2^V different values to compute the expectation over the stochastic dropout vector $\mathbf{b}$. Therefore, we follow Wang and Manning (2013) to use a continuous Gaussian approximation, where the Bernouli noise $\mathbf{b}$ is replaced by a Gaussian noise $\mathbf{z}$:

$$E(x|z) = (\mathbf{z} \odot OneHot(x)) \cdot W \tag{4}$$

where $\mathbf{z} \in \mathbb{R}^V$ follows Gaussian distribution $z_i \sim \mathcal{N}(1, \alpha_i = \frac{p_i}{1-p_i})$. It is worth noting that

α and p are one-to-one corresponded, and α is a monotonously increasing function of p. For more details, please refer to Wang and Manning (2013). Based on such approximation, we can use α as dropout criteria, e.g. throw away words with α above a certain given threshold α_T. We further follow Louizos et al. (2017); Kingma et al. (2015); Molchanov et al. (2017) to re-interpret the input noise as the intrinsic stochasticity in the embedding weights B itself as follows:

$$E(x|z) = \mathit{OneHot}(x) \cdot B \tag{5}$$

where $B \in \mathbb{R}^{V*D}$ follows a multi-variate Gaussian distribution $B_{ij} \sim \mathcal{N}(\mu_{ij} = W_{ij}, \sigma_{ij}^2 = \alpha_i W_{ij}^2)$, where the random weights in each row has a tied variance/mean ratio α_i. Thus, we re-write the evidence lower bound as follows:

$$\begin{aligned}&\log \mathcal{L}(f_\theta(\mathbf{x}), y)) = \log \int_B \mathcal{L}(f_\theta(E(\mathbf{x}|z)), y))\mathcal{P}(B)dB \\ \geq &\mathop{\mathbb{E}}_{B\sim\mathcal{N}(\mu,\sigma)}[\log \mathcal{L}(f_\theta(E(\mathbf{x}|z)), y)] - KL(\mathcal{N}(\mu,\sigma)||\mathcal{P}(B)) \\ = &\mathcal{L}(B,\theta)\end{aligned}$$

where $\mathcal{P}(B)$ is the prior distribution and $\mathcal{N}(\mu, \sigma)$ denotes the Gaussian approximate posterior with parameters μ and σ. $\mathcal{L}(B, \theta)$ is used as the relaxed evidence lower bound of marginal log likelihood $\log \mathcal{L}(f_\theta(\mathbf{x}), y))$. Here, we follow Kingma et al. (2015); Louizos et al. (2017) to choose the prior distribution $\mathcal{P}(B)$ as the "improper log-scaled uniform distribution" to guarantee that the regularization term $D_{KL}(\mathcal{N}(\mu, \sigma)||\mathcal{P}(B))$ only depends on dropout ratio α, i.e. irrelevant to μ. Formally, we write the prior distribution as follows:

$$\mathcal{P}(\log |B_{ij}|) = \mathit{const} \rightarrow \mathcal{P}(|B_{ij}|) \propto \frac{1}{|B_{ij}|} \tag{6}$$

Since there exists no closed-form expression for such KL-divergence, we follow Louizos et al. (2017) to approximate it by the following formula with minimum variance:

$$\begin{aligned}D_{KL} = -k_1\sigma(k_2 + k_3 \log \alpha) + \frac{1}{2}\log(1 + \frac{1}{\alpha}) + k_1 \\ k_1 = 0.63576 \quad k_2 = 1.87320 \quad k_3 = 1.48695\end{aligned} \tag{7}$$

By adopting the improper log-uniform prior, more weights are compressed towards zero, and the KL-divergence is negatively correlated with dropout ratio α. Intuitively, the dropout ratio α_i is an redundancy indicator for i_{th} word in the vocabulary, with larger α_i meaning less performance loss caused by dropping i_{th} word. During training, we use re-parameterization trick (Kingma and Welling, 2013) to sample embedding weights from the normal distribution to reduce the Monte-Carlo variance in Bayesian training.

3.3 Vocabulary Selection

After optimization, we can obtain the dropout ratio α_i associated with each word w_i. We propose to select vocabulary subset based on the dropout ratio by using a threshold α_T. Therefore, the remaining vocabulary subset is described as follows:

$$\hat{\mathbb{V}} = \{w_i \in V | \alpha_i < \alpha_T\} \tag{8}$$

where we use $\hat{\mathbb{V}}$ to denote the subset vocabulary of interest, by adjusting α_T we are able to control the selected vocabulary size.

4 Experiments

We compare the proposed vocabulary selection algorithm against several strong baselines on a wide range of text classification tasks and datasets.

4.1 Datasets & Architectures

The main datasets we are using are listed in Table 2, which provides an overview of its description and capacities. Specifically, we follow (Zhang et al., 2015; Goo et al., 2018; Williams et al., 2018) to pre-process the document classification datasets, natural language understanding dataset and natural language inference dataset. We exactly replicate their experiment settings to make our method comparable with theirs. Our models is implemented with TensorFlow (Abadi et al., 2015). In order to evaluate the generalization ability of VVD selection algorithm in deep learning architectures, we study its performance under different established architectures (depicted in Figure 3). In natural language understanding, we use the most recent attention-based model for intention tracking (Goo et al., 2018), this model first uses BiLSTM recurrent network to leverage left-to-right and right-to-left context information to form the hidden representation, then computes self-attention weights to aggregate the hidden representation and predicts user intention. In document classification, we mainly follow the CNN architecture (Kim, 2014) to extract n-gram features and then aggregate these features to predict document category. In natural language inference, we follow the popular ESIM architecture (Williams et al., 2018; Chen et al., 2017) us-

Datasets	Task	Description	#Class	#Train	#Test
ATIS-flight (Tur et al., 2010)	NLU	Classify Airline Travel dialog	21	4,478	893
Snips (Coucke et al., 2018)		Classify inputs to personal voice assistant	7	13,084	700
AG-news (Zhang et al., 2015)	DC	Categories: World, Sports, etc	4	120,000	7,600
DBPedia (Lehmann et al., 2015)		Categories: Company, Athlete, Album, etc	14	560,000	70,000
Sogou-news (Zhang et al., 2015)		Categories: Sports, Technology, etc	5	450,000	60,000
Yelp-review (Zhang et al., 2015)		Categories: Review Ratings (1-5)	5	650,000	50,000
SNLI (Bowman et al., 2015)	NLI	Entailment: Contradict, Neutral,Entail	3	550,152	10,000
MNLI (Williams et al., 2018)		Multi-Genre Entailment	3	392,702	10,000

Table 2: An overview of different datasets under different classification tasks including description and sizes.

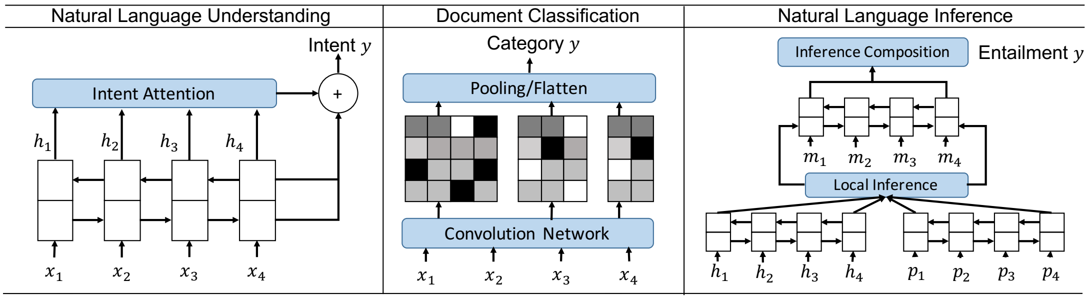


Figure 3: The neural network architecture overview of different NLP tasks.

ing the Github implementation[3]. In this structure, three main components input encoding, local inference modeling, and inference composition are used to perform sequential inference and composition to simulate the interaction between premises and hypothesis. Note that, we do not apply the syntax-tree based LSTM proposed in (Chen et al., 2017) because we lost the parse tree (Klein and Manning, 2003) after the vocabulary compression, instead, we follow the simpler sequential LSTM framework without any syntax parse as input. Besides, the accuracy curve is obtained using the publicly available test split rather than the official online evaluation because we need to evaluate lots of times at different vocabulary capacity.

4.2 Baselines

Here we mainly consider the following baselines:

Frequency-based (task-agnostic) This approach is already extensively talked about in section 1, its basic idea is to rank the word based on its frequency and then set a threshold to cut off the long tail distribution.

TF-IDF (task-agnostic) This algorithm views the vocabulary selection as a retrieval problem (Ramos et al., 2003), where term frequency is viewed as the word frequency and document frequency is viewed as the number of sentences where such word appears. Here we follow the canonical TF-IDF approach to compute the retrieval score as follows:

$$tfidf(w, D) = tf(w)^{\lambda} * (\log \frac{N}{n_w})^{1-\lambda} \tag{9}$$

where $tf(w)$ denotes the word frequency, λ is the balancing factor, N denotes the number of sentences and n_w denotes the number of sentences in which w appears. We rank the whole vocabulary based on the $tfidf$ and cut off at given threshold.

Group Lasso (task-aware) This baseline aims to find intrinsic sparse structures (Liu et al., 2015; Park et al., 2016; Wen et al., 2016) by grouping each row of word embedding. The regularization objective is described as follows, which aims at finding the row-wise sparse structure:

$$\mathcal{L}_{reg} = \sum_i (\sum_j W_{ij}^2)^{\frac{1}{2}} \tag{10}$$

After optimized with the above regularization, we use a threshold-based selection strategy on the row-norm of embedding matrix, the selected vocabulary is described as $\hat{\mathbb{V}} = \{w_i \in \mathbb{V} | ||W_i||_2 > \beta_T\}$, where β_T is the threshold.

4.3 Experimental Results

Here we demonstrate our results in natural language understanding, document classification, and

---

[3] `https://github.com/coetaur0/ESIM`

Datasets / Reported Accuracy	Accuracy	Vocab	Methods	AUC	Vocab@-3%	Vocab@-5%
Snips / 96.7 (Liu and Lane, 2016)	95.9	11000	Frequency	77.4	81	61
	95.9		TF-IDF	77.6	81	62
	95.6		Group Lasso	82.1	77	52
	96.0		VVD	**82.5**	**52**	**36**
ATIS-Flight / 94.1 (Goo et al., 2018)	93.8	724	Frequency	70.1	33	28
	93.8		TF-IDF	70.5	34	28
	93.8		Group Lasso	72.9	30	26
	94.0		VVD	**74.8**	**29**	**26**
AG-news / 91.1 (Zhang et al., 2015)	91.6	61673	Frequency	67.1	2290	1379
	91.6		TF-IDF	67.8	2214	1303
	91.2		Group Lasso	68.3	1867	1032
	91.6		VVD	**70.5**	**1000**	**673**
DBPedia / 98.3 (Zhang et al., 2015)	98.4	563355	Frequency	69.7	1000	743
	98.4		TF-IDF	71.7	1703	804
	97.9		Group Lasso	71.9	768	678
	98.5		VVD	**72.2**	**427**	**297**
Sogou-news / 95.0 (Zhang et al., 2015)	93.7	254495	Frequency	70.9	789	643
	93.7		TF-IDF	71.3	976	776
	93.6		Group Lasso	73.4	765	456
	94.0		VVD	**75.5**	**312**	**196**
Yelp-review / 58.0 (Zhang et al., 2015)	56.3	252712	Frequency	74.0	1315	683
	56.3		TF-IDF	74.1	1630	754
	56.5		Group Lasso	75.4	934	463
	57.4		VVD	**77.9**	**487**	**287**
SNLI / 86.7 (Williams et al., 2018)	84.1	42392	Frequency	72.2	2139	1362
	84.1		TF-IDF	72.8	2132	1429
	84.6		Group Lasso	73.6	1712	1093
	85.5		VVD	**75.0**	**1414**	**854**
MNLI / 72.3 (Williams et al., 2018)	69.2	100158	Frequency	78.5	1758	952
	69.2		TF-IDF	78.7	1656	934
	70.1		Group Lasso	79.2	1466	711
	71.2		VVD	**80.1**	**1323**	**641**

Table 3: Experimental Results on various NLP tasks and datasets on the proposed metrics in subsection 2.3. Bold accuracy means the result is statistically significantly better than the competitors.

natural language inference separately in Table 3. From these tables, first of all, we can observe that VVD is able to maintain or even improve the reported accuracy on DC and NLU tasks, the accuracy of VVD is reported under dropping out the words with dropout rate larger than 0.95. The exception is in NLI (Williams et al., 2018), where the common approach uses GloVe (Pennington et al., 2014) for initialization, and we use random initialization, which makes our model fall slightly behind. It is worth noting that Frequency-based/TF-IDF methods are based on the model trained with cross entropy, while both Group-Lasso and VVD modify the objective function by adding additional regularization. It can be seen that VVD is performing very similar to the baseline models on DC and NLU tasks, while consistently outperforming the baseline methods (with random initialized embedding) on more challenging NLI and Yelp-Review tasks, that said, VVD can also be viewed as a generally effective regularization technique to sparsify features and alleviate the over-fitting problem in NLP tasks. In terms of the vocabulary selection capability, our proposed VVD is demonstrated to outperform the competing algorithms in terms of both AUC and Vocab@-X% metrics consistently over different datasets as shown in Table 3. In order to better understand the margin between VVD and frequency-based method, we plot their accuracy-vocab curves in Figure 4, from which we can observe that the accuracy curves start from nearly the same accuracy with the full vocabulary, by gradually decreasing the budget $\hat{V}$, VVD decreases at a much lower rate than the competing algorithms, which clearly reflects its superiority under limited-budget scenario. From the empirical result, we can conclude that: 1) the retrieval-based selection algorithm can yield

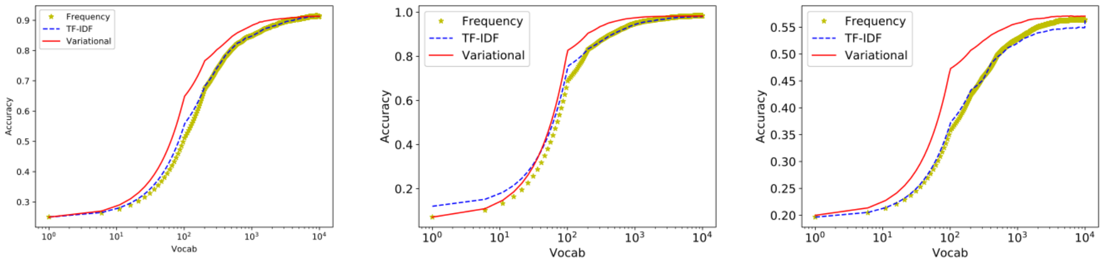


Figure 4: The accuracy-vocab curve of VVD, TF-IDF and frequency-based baseline, the datasets used are AG-news, DBPedia and Yelp-review respectively.

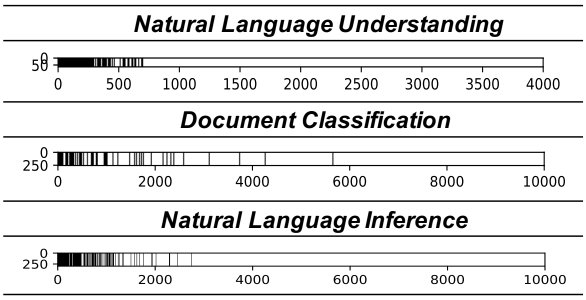


Figure 5: The vocabulary selection spectrum of our proposed VVD algorithm on different NLP tasks.

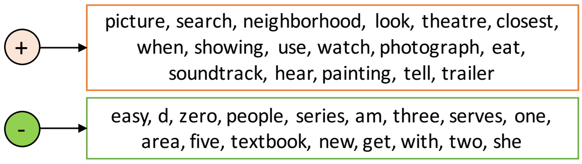


Figure 6: The vocabulary selected (+)/unselected (-) by VVD compared to frequency-based baseline.

marginal improvement over the AUC metric, but the vocab@-X% metric deteriorates. 2) group-lasso and VVD algorithm directly considers the connection between each word and end classification accuracy; such task-awareness can greatly in improving both evaluation metrics. Here we show that NLU datasets are relatively simpler, which only involves detecting key words from human voice inputs to make decent decisions, a keyword vocabulary within 100 is already enough for promising accuracy. For DC datasets, which involve better inner-sentence and inter-sentence understanding, hundred-level vocabulary is required for most cases. NLI datasets involve more complicated reasoning and interaction, which requires a thousand-level vocabulary.

Figure 7: The vocabulary cloud of Snips NLU dataset.

Case Study To provide an overview of what words are selected, we depict the selection spectrum over different NLP tasks in Figure 5, from which we observe that most of the selected vocabulary are still from the high-frequency area to ensure coverage, which also explains why the frequency-based algorithm is already very strong. Furthermore, we use the Snips dataset (Coucke et al., 2018) to showcase the difference between the vocabularies selected by VVD and by frequency-based baseline. The main goal of this dataset is to understand the speaker's intention such as "BookRestaurant", "PlayMusic", and "SearchLocalEvent". We show the selected/unselected words by our algorithm in Figure 6 under a vocabulary budget of 100, it is observed that many non-informative but frequent functional words like "get", "with", and "five" are unselected while more task-related but less frequent words like "neighborhood", "search", "theatre" are selected. More vividly, we demonstrate the word cloud of the selected vocabulary of Snips (Coucke et al., 2018) in Figure 7.

4.4 Discussion

Here we will talk about some potential issues posed when training and evaluating VVD.

Training Speed Due to the stochasticity of VVD, the training of text classification takes longer than canonical cross entropy objective. More importantly, we observe that with the increase the full vocabulary size, the convergence time of VVD also increases sub-linearly but the convergence time of Cross Entropy remains quite consistent. We conjecture that this is due to the fact that the VVD algorithm has a heavier burden to infer the drop out the probability of the long tail words. Therefore, we propose to use a two-step vocabulary reduction to dramatically decrease VVD's training time, in the first step, we cut off the rare words without having any harm on the final accuracy, then we continue training with VVD on the shrunk vocabulary. By applying such a hybrid methodology, we are able to decrease the training time dramatically.

Evaluation Speed As we know, at each vocabulary point, the network needs to perform once evaluation on the whole test set. Therefore, it is not practical to draw each vocabulary size from 1 to V and perform V times of evaluation. Given the limited computational resources, we need to sample some vocabulary size and estimate the area under curve relying on only these points. Uniformly sampling the data points are proved wasteful, since when the accuracy curve will converge to a point very early, most of the sampled point is actually getting equivalent accuracy. Therefore, we propose to increase the interval exponentially to cover more samples at extremely low vocabulary size. For example, given the total vocabulary of 60000, the interval will be split into 1, 2, 4, 8, 24, 56, ..., 60K. Using such sampling method achieve a reasonably accurate estimation of ROC with only $O(log(|V|))$ sample points, which is affordable under many cases.

5 Related Work

Neural Network Compression In order to better apply the deep neural networks under limited-resource scenarios, much recent research has been performed to compress the model size and decrease the computation resources. In summary, there are mainly three directions, weight matrices approximation (Le et al., 2015; Tjandra et al., 2017), reducing the precision of the weights (Hubara et al., 2017; Han et al., 2015) and sparsification of the weight matrix (Wen et al., 2016). Another group of sparsification relies on the Bayesian inference framework (Molchanov et al., 2017; Neklyudov et al., 2017; Louizos et al., 2017). The main advantage of the Bayesian sparsification techniques is that they have a small number of hyperparameters compared to pruning-based methods. As stated in (Chirkova et al., 2018), Bayesian compression also leads to a higher sparsity level (Molchanov et al., 2017; Neklyudov et al., 2017; Louizos et al., 2017). Our proposed VVD is inspired by these predecessors to specifically tackle the vocabulary redundancy problem in NLP tasks.

Vocabulary Reduction An orthogonal line of research for dealing similar vocabulary redundancy problem is the character-based approaches to reduce vocabulary sise (Kim et al., 2016; Zhang et al., 2015; Costa-Jussà and Fonollosa, 2016; Lee et al., 2017), which decomposes the words into its characters forms for better handling open world inputs. However, these approaches are not applicable to character-free languages like Chinese and Japanese. Moreover, splitting words into characters incurs potential lose of word-level surface form, and thus needs more parameters at the neural network level to recover it to maintain the end task performance (Zhang et al., 2015), which contradicts with our initial motivation of compressing the neural network models for computation- or memory-constrained scenarios.

6 Conclusion

In this paper, we propose a vocabulary selection algorithm which can find sparsity in the vocabulary and dynamically decrease its size to contain only the useful words. Through our experiments, we have empirically demonstrated that the commonly adopted frequency-based vocabulary selection is already a very strong mechanism, further applying our proposed VVD can further improve the compression ratio. However, due to the time and memory complexity issues, our algorithm and evaluation are more suitable for classification-based application. In the future, we plan to investigate broader applications like summarizaion, translation, question answering, etc.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by NSF 1528175. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Nadezhda Chirkova, Ekaterina Lobacheva, and Dmitry Vetrov. 2018. Bayesian compression for natural language processing. *arXiv preprint arXiv:1810.10927*.

Marta R. Costa-Jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1491–1500.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2010. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 753–757.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *International Conference on Learning Representations*.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL*, 5:365–378.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. 2015. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 685–689.

Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2498–2507.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. 2017. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784.

Jongsoo Park, Sheng Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey. 2016. Faster cnns with direct sparse convolutions and guided pruning. *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4451–4458.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 19–24. IEEE.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *international conference on machine learning*, pages 118–126.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 87–96.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Subword-based Compact Reconstruction of Word Embeddings

Shota Sasaki[1,2], **Jun Suzuki**[2,1], **Kentaro Inui**[2,1]
[1]RIKEN AIP, [2]Tohoku University
`{sasaki.shota, jun.suzuki, inui}@ecei.tohoku.ac.jp`

Abstract

The idea of *subword-based* word embeddings has been proposed in the literature, mainly for solving the out-of-vocabulary (OOV) word problem observed in standard *word-based* word embeddings. In this paper, we propose a method of reconstructing pre-trained word embeddings using subword information that can effectively represent a large number of subword embeddings in a considerably small fixed space. The key techniques of our method are twofold: memory-shared embeddings and a variant of the key-value-query self-attention mechanism. Our experiments show that our reconstructed subword-based embeddings can successfully imitate well-trained word embeddings in a small fixed space while preventing quality degradation across several linguistic benchmark datasets, and can simultaneously predict effective embeddings of OOV words. We also demonstrate the effectiveness of our reconstruction method when we apply them to downstream tasks[1].

1 Introduction

Pre-trained word embeddings (or embedding vectors), especially those trained on a vast amount of text data, such as the Common Crawl (CC) corpus[2], are now considered as highly beneficial, fundamental language resources. Typical examples of large, well-trained word embeddings are those trained on the CC corpus with 600 billion tokens by fastText (Bojanowski et al., 2017) and with 840 billion tokens by GloVe (Pennington et al., 2014), which we refer to as `fastText.600B`[3] and `GloVe.840B`[4], respectively. In fact, we often leverage such word embeddings to further improve the task performance of many natural language processing (NLP) tasks, such as constituency parsing (Suzuki et al., 2018; Gómez-Rodríguez and Vilares, 2018), discourse parsing (Yu et al., 2018), semantic parsing (Groschwitz et al., 2018; Dong and Lapata, 2018), and semantic role labeling (Strubell et al., 2018).

Despite their significant impact on the NLP community, well-trained word embeddings still have several disadvantages. In this paper, we focus on two issues surrounding well-trained word embeddings: i) the massive memory requirement and ii) the inapplicability of out-of-vocabulary (OOV) words. It is crucial to address such issues, especially when applying them to real-world open systems. The total number of embeddings (i.e., the total memory requirement of such word embeddings) often becomes unacceptably large, especially in limited-memory environments, including GPUs, since the vocabulary size is more than 2 million words, which require at least 2 gigabytes (GB) of memory for storage.

One possible solution is to merely discard (less important) words from the vocabulary, which can straightforwardly reduce the memory requirement. However, such a naive method can cause another well-known drawback regarding the inapplicability of OOV words. The applicability of OOV words is highly desirable in real systems since input words can be uncontrollably diverse. Therefore, there is a trade-off between the number of embedding vectors and the applicability of OOV words; thus, our goal is to investigate and develop a method that simultaneously has less memory requirement and high applicability of OOV words, which are both desirable properties for word embeddings in real-world open systems.

Recently, methods that leverage subword information have been proposed and have become popular for overcoming the OOV word issue. Con-

---

[1] Our code and reconstructed subword-based word embeddings trained from `GloVe.840B` and `fastText.600B` are available: `https://github.com/losyer/compact_reconstruction`

[2] `http://commoncrawl.org`

[3] `https://fasttext.cc/docs/en/english-vectors.html`

[4] `https://nlp.stanford.edu/projects/glove/`

Proceedings of NAACL-HLT 2019, pages 3498–3508
Minneapolis, Minnesota, June 2 - June 7, 2019.

ceptually, the subword-based approach can cover all the words that can be constructed by a combination of subwords. Thus, the subword-based approach can greatly mitigate (or solve) the OOV word issue. We extend this approach to simultaneously enabling a reduction in the total number of embedding vectors through the reconstruction of word embeddings by subwords. The key techniques of our approach are twofold: memory-shared embeddings and a variant of the key-value-query (KVQ) self-attention mechanism (Vaswani et al., 2017). That is, our approach reconstructs well-trained word embeddings using a limited number of embedding vectors that are shared by all the subwords with an effective weighting calculated by the self-attention mechanism.

In our experiments, we show that our reconstructed subword-based embeddings can successfully imitate well-trained word embeddings, such as `fastText.600B` and `GloVe.840B`, in a small fixed space while preventing quality degradation across several linguistic benchmark datasets from word similarity and analogy tasks. We also demonstrate the effectiveness of our reconstructed embeddings for representing the embeddings of OOV words. Lastly, we confirm the performance of our reconstructed embeddings on several downstream tasks from the named entity recognition task and the textual entailment task.

2 Related Work

The OOV word issue is one of the widely discussed topics in word embedding research, which several researches have recently attempted to solve. For example, methods that leverage subword information, such as character N-grams (including character unigrams) (Bojanowski et al., 2017; Pinter et al., 2017; Zhao et al., 2018) and morphological features (Luong et al., 2013), have recently been discussed as means of constructing word embeddings that consider the applicability of OOV words. Moreover, Pilehvar and Collier (2017) have proposed a method called `SemLand`, which induces OOV word embeddings by leveraging external resources. Bahdanau et al. (2017) and Herbelot and Baroni (2017) have also proposed methods that estimate OOV word embeddings using an additional LSTM and leveraging a small additional dataset, respectively.

Among them, the study most closely related to ours is that of Zhao et al. (2018). Their basic idea is to reconstruct each pre-trained word embedding using a bag-of-character N-grams. We refer to their method as '`BoS`'. The motivation for reconstructing pre-trained word embeddings and utilizing character N-grams in our approach is substantially the same, however, an essential difference from `BoS` is that we additionally consider jointly reducing the total number of embedding vectors.

Another study that shares the same motivation and goal is that of Pinter et al. (2017). Their method, referred to as `MIMICK`, utilizes only character information instead of character N-grams by mixing it with more sophisticated neural networks, i.e., LSTM (Hochreiter and Schmidhuber, 1997). `MIMICK` can produce a more compact model than the original word embeddings. The important difference between their method and ours is that our method only consists of the subword embeddings, whereas their method consists of the character embeddings and several transformation matrices for calculating LSTMs. We compare their method with ours in our experiments and empirically show the effectiveness of our approach.

Moreover, Bojanowski et al. (2017) have proposed a method called `fastText`, which also incorporates character N-gram embeddings in addition to word embeddings. However, they did not explicitly prove the effectiveness of OOV word embeddings. Thus, it is still unclear how well the combination of character N-grams can reconstruct appropriate embeddings for OOV words. In addition, their method trains word embeddings from a corpus, which is not a reconstruction setting we discuss in this paper. Therefore, their method is orthogonal to ours.

We often aim to reduce memory consumption of word embeddings in the real-world since they require relatively large memory. Suzuki and Nagata (2016) proposed a parameter reduction method for word embeddings by using machine learning techniques. Our method can also be interpreted as a kind of parameter reduction method based on the subword features. However, their method only considers the model shrinkage, and does not utilize any subword information nor consider the OOV issue.

To summarize, none of the previous studies have attempted to simultaneously achieve a smaller number of embedding vectors and higher applicability of OOV words. Thus, in this paper, we report the first attempt to investigate how we can simultaneously achieve them.

Additionally, deep contextualized pre-trained language models, such as ELMo (Peters et al., 2018), have recently been proposed as alternatives to the pre-trained word embeddings to further improve task performances. However, ELMo still takes advantage of `Glove.840B` to achieve its state-of-the-art performance. This fact implies that we can still combine the pre-trained word embeddings with strong pre-trained language models; thus, the importance of word embeddings in the literature remains unchanged even though stronger pre-trained models have been established.

3 Reconstruction of Word Embeddings Using Subwords

In this section, we explain a formal task definition that we tackle in this paper.

3.1 Preliminaries

Notation rules: In this paper, we use the following notation rules unless otherwise specified. First, a lower-case bold letter, e.g., $\boldsymbol{v}$ and $\boldsymbol{e}$, represents a column vector, and an upper-case bold letter, e.g., $\boldsymbol{V}$ and $\boldsymbol{E}$, represents a matrix. Then, $\|\boldsymbol{v}\|_p$ represents L_p-norm of the given vector $\boldsymbol{v}$. Next, let a lower-case letter, e.g., z or i, be a scalar variable or index, and an upper-case letter, e.g., C or H, indicate a scalar but hyper-parameter during the training. Here, we introduce the notation $\boldsymbol{V}[i]$ to represent the i-th column vector in the matrix $\boldsymbol{V}$ to simplify the representation. Moreover, an upper-case letter in a calligraphy form, e.g., $\mathcal{W}$ and $\mathcal{S}$, denotes a set, and the absolute value of a set, such as $|\mathcal{W}|$ and $|\mathcal{S}|$, indicates the number of instances in the corresponding set. Finally, the Greek letter, such as Φ and η, indicates a function.

Words and their embedding: Let $\mathcal{W}$ be a vocabulary, i.e., a set of words. Let $\zeta(\cdot)$ represent a mapping function from a word to the corresponding index of the word, namely,

$$\zeta(\cdot) : \mathcal{W} \rightarrow \mathcal{I}_w \text{ where } \mathcal{I}_w = \{1, \dots, |\mathcal{W}|\}. \quad (1)$$

In this paper, we always assume that $\zeta(\cdot)$ is a bijective function; thus, each word has its own unique index between 1 and $|\mathcal{W}|$. This also implies that the relation $|\mathcal{W}| = |\mathcal{I}_w|$ always holds.

Let $\boldsymbol{e}_w$ be a D-dimensional embedding vector for the word $w \in \mathcal{W}$, and let $\boldsymbol{E}$ denote an embedding matrix for all words in $\mathcal{W}$, where $\boldsymbol{E} = \mathbb{R}^{D\times|\mathcal{I}_w|}$. Then, we assume that the following relation always holds between $\boldsymbol{e}_w$ and $\boldsymbol{E}$:

$$\boldsymbol{e}_w = \boldsymbol{E}[z_e] \quad \text{where} \quad z_e = \zeta(w). \quad (2)$$

Therefore, the i-th column vector in the matrix $\boldsymbol{E}$ represents the word embedding of the corresponding word w that satisfies $i = \zeta(w)$.

Subwords and their embedding: Let $\mathcal{S}$ be a vocabulary for all pre-defined subwords obtained from the words in $\mathcal{W}$. Let $\eta_v(\cdot)$ represent a mapping function from a subword to the corresponding index of the subword, that is,

$$\eta_v(\cdot) : \mathcal{S} \rightarrow \mathcal{I}_s \text{ where } \mathcal{I}_s = \{1, \dots, |\mathcal{S}|\}. \quad (3)$$

Similar to ζ in Eq. 1, $\eta_v(\cdot)$ is generally defined as a bijective function[5]. In this case, each subword has its own unique index between 1 and $|\mathcal{S}|$, and the relation $|\mathcal{S}| = |\mathcal{I}_s|$ always holds.

Here, we introduce $\boldsymbol{v}_s$ as a D-dimensional vector for the subword $s \in \mathcal{S}$ and $\boldsymbol{V}$ as an embedding matrix for all subwords in $\mathcal{S}$, where $\boldsymbol{V} = \mathbb{R}^{D\times|\mathcal{I}_s|}$. Then, we also assume the following relation between $\boldsymbol{v}_s$ and $\boldsymbol{V}$:

$$\boldsymbol{v}_s = \boldsymbol{V}[z_v] \quad \text{where} \quad z_v = \eta_v(s). \quad (4)$$

Therefore, the j-th column vector in the matrix $\boldsymbol{V}$ represents the subword embedding of the corresponding subword s that satisfies $j = \eta_v(s)$.

Word to subword mapping: Additionally, we introduce a (abstract) function $\phi(\cdot)$ that maps a word $w \in \mathcal{W}$ to a list of subwords contained in the word w. We can define $\phi(\cdot)$ in detail from several choices. For example, if we define $\phi(\cdot)$ to extract all the character bi-grams appearing in a given word and $w =$ 'higher', then we obtain a list of total seven distinct subword indices of '⟨w⟩h', 'hi', 'ig', 'gh', 'he', 'er', 'r⟨/w⟩' as the return value of $\phi(w)$, where '⟨w⟩' and '⟨/w⟩' are special characters that represent the beginning and end of a word, respectively.

3.2 Task definition

Conceptually, we aim to reconstruct all the embeddings in $\boldsymbol{E}$ using $\boldsymbol{V}$ and a pre-defined subword mixing function $\tau(\cdot)$. Formally, our reconstruction problem is represented as a minimization problem of the following form:

$$\hat{\boldsymbol{V}} = \underset{\boldsymbol{V}}{\arg\min} \big\{\Psi(\boldsymbol{E}, \boldsymbol{V}, \tau)\big\}, \quad (5)$$

---

[5] We redefine η_v as a surjective function in Section 5.2.

where $\Psi(\cdot)$ is a loss function used to calculate the total reconstruction loss between $\boldsymbol{E}$ and $\boldsymbol{V}$.

As a brief summary, our goal is to find $\hat{\boldsymbol{V}}$ at which the loss function $\Psi(\cdot)$ is minimized from the machine learning perspective. Note that the previous study, i.e., `BoS`, also utilized the above formulation for the reconstruction problem. Moreover, `MIMICK` can also be considered to utilize this formulation if $\boldsymbol{V}$ consists of all the single characters.

Subword mixing function $\tau(\cdot)$: The role of the function $\tau(\cdot)$ is to calculate an alternative embedding of the word embedding $\boldsymbol{e}_w$ using a list of subwords contained in the given word w. One of the most popular definitions of $\tau(\cdot)$ is to simply sum-up all the obtained subwords as follows:

$$\tau_{\texttt{sum}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} \boldsymbol{v}_s. \tag{6}$$

In fact, a subword mixing function of this form was utilized in the previous studies, such as `fastText` and `BoS`.

Loss function $\Psi(\cdot)$: First, to improve readability, we introduce $\hat{\boldsymbol{v}}_w$ as a short notation of $\tau(\boldsymbol{V}, w)$, namely $\hat{\boldsymbol{v}}_w = \tau(\boldsymbol{V}, w)$. There are also several possible choices for the definition of the loss function $\Psi(\cdot)$. Here, we consider utilizing a squared loss function $\Psi_{\texttt{lsq}}$, which can be written as the summation of the squared losses over an individual embedding vector $\boldsymbol{e}_w$:

$$\Psi(\boldsymbol{E}, \boldsymbol{V}, \tau) = \sum_{w \in \mathcal{W}} C_w \left\| \boldsymbol{e}_w - \hat{\boldsymbol{v}}_w \right\|_2^2, \tag{7}$$

where C_w is a weight factor for each word. Intuitively, $\Psi(\cdot)$ is used to calculate the weighted sum of the L_2-norm distances between the reference vector $\boldsymbol{e}_w$ and a vector calculated by the subword mixing function $\tau(\cdot)$.

4 Reconstruction with Model Shrinkage

In this section, we briefly explains the background that we still need to consider the number of embedding vectors in the subword-based approach. Table 1 shows the statistics of the total number of embedding vectors and the total memory requirement in several different settings. As shown in row `(a)`, the original `fastText.600B` word embeddings consist of 2 million words, which require 2.2 GB to store them since each word has a $D = 300$ dimensional vector. If we consider using all the character N-grams obtained from all the words (row `(e)`), then surprisingly, the memory requirement becomes approximately 25GB, which is too large for practical use. Therefore, it is crucial to technically reduce the memory requirement.

ID	Setting	# of vecs	mem. (GB)
`(a)`	original `fastText.600B`	2.0 M	2.2 GB
`(b)`	char N-gram $N = 1, 2, 3$	0.2 M	0.3 GB
`(c)`	$N = 3, 4, 5, 6$	6.2 M	7.1 GB
`(d)`	$N = 1$ to 6	6.3 M	7.2 GB
`(e)`	$N = 1$ to ∞	21.8 M	24.9 GB

Table 1: Statistics for each setting: The columns '# of vecs' and 'mem.' represent the number of embedding vectors and the memory requirement, respectively. M denotes one million, and we consider that a real value requires 4 bytes of storage in the calculation of the memory requirement.

A practical approach is to partially take advantage of a certain range of smaller N-grams, such as $N = 1$ to 3 (row `(b)`) or $N = 3$ to 6 (row `(c)`). However, smaller subword settings, such as row `(b)`, might markedly degrade the performance from the original word embeddings. Therefore, it is necessary to discover a better balance between the memory requirement (or the total number of embedding vectors) and performance.

5 Modifications to Improve Performance

In this section, we describe several modifications to simultaneously achieve the purpose of smaller number of embedding vectors but higher performance with the applicability of OOV words. To do so, we incorporate several techniques in the baseline word embedding reconstruction approach explained in Section 3. Roughly speaking, we enhance the mapping function $\eta_v(\cdot)$ and the subword mixing function $\tau(\cdot)$.

5.1 Frequent subwords: Modification of $\eta_v(\cdot)$

We take advantage of the top-F frequent subwords that can be counted from the words in vocabulary $\mathcal{W}$ as a subword vocabulary instead of all possible subwords $\mathcal{S}$. Let $\mathcal{S}_F$ represent the set of the top-F frequent subwords, where $\mathcal{S}_F \subseteq \mathcal{S}$. Then, we define a new mapping function $\eta_{v,F}(\cdot)$ as follows:

$$\eta_{v,F}(\cdot) : \mathcal{S}_F \rightarrow \mathcal{I}_{s,F} \text{ where } \mathcal{I}_{s,F} = \{1, \dots, |\mathcal{S}_F|\}. \tag{8}$$

5.2 Memory sharing: Modification of $\eta_v(\cdot)$

In the baseline method, we assumed that the mapping function $\eta_v(\cdot)$ is a bijective function as de-

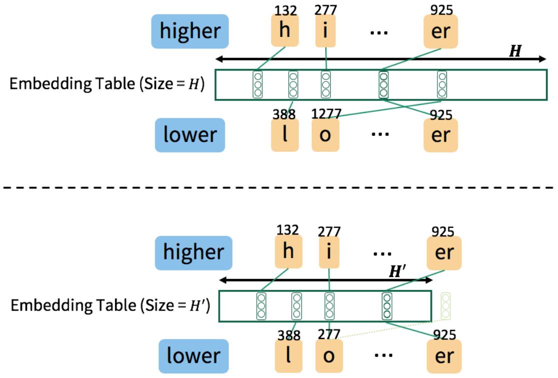


Figure 1: Intuitive idea of our memory-shared embeddings by hashing. Here, we assume $H' < H$.

scribed in Section 3.1. Again, this means that each subword has its own unique subword index. Here, we modify $\eta_v(\cdot)$ as a surjective function by introducing the following new mapping function $\eta_{v,H}(\cdot)$ as a replacement of $\eta_v(\cdot)$ in Eq. 3:

$$\eta_{v,H}(\cdot) : \mathcal{S} \rightarrow \mathcal{I}_{s,H} \;\text{ where }\; \mathcal{I}_{s,H} = \{1, \ldots, H\}. \tag{9}$$

Here, we assume $H < |\mathcal{S}|$. This mapping function $\eta_{v,H}(\cdot)$ implies that each subword is mapped to an index, but the index is not unique and may be shared between other subwords. Therefore, the number of subword embeddings can also be reduced to H by sharing the embeddings; thus, the subword embedding matrix $\boldsymbol{V}$ can also be reduced from $\boldsymbol{V} = \mathbb{R}^{D \times |\mathcal{S}|}$ to $\boldsymbol{V} = \mathbb{R}^{D \times H}$. Here, if we assume the relation $H \ll |\mathcal{S}|$, then we can greatly reduce the total embedding size.

There are several possible choices for the definition of the mapping function $\eta_{v,H}(\cdot)$. We select a simple hash function for $\eta_{v,H}(\cdot)$. This means that subword embeddings are randomly shared over the subwords. One large merit of using simple hash functions is that we require no external mapping structure of every (subword, subword index) pair, which also matches our goal of reducing the memory requirement in actual use cases. Figure 1 illustrates an intuitive idea of memory-shared embeddings by hashing.

5.3 Combination of $\eta_{v,F}(\cdot)$ and $\eta_{v,H}(\cdot)$

Also $\eta_{v,F}(\cdot)$ and $\eta_{v,H}(\cdot)$ can be combined step by step. First, we reduce the subword vocabulary $\mathcal{S}$ to top-F frequent subwords $\mathcal{S}_F$ as described in Section 5.1. Second, we apply our memory sharing method to only $\mathcal{S}_F$ in contrast to applying it to $\mathcal{S}$ in Section 5.2. Here, we define a new mapping function $\eta_{v,FH}(\cdot)$ as follows:

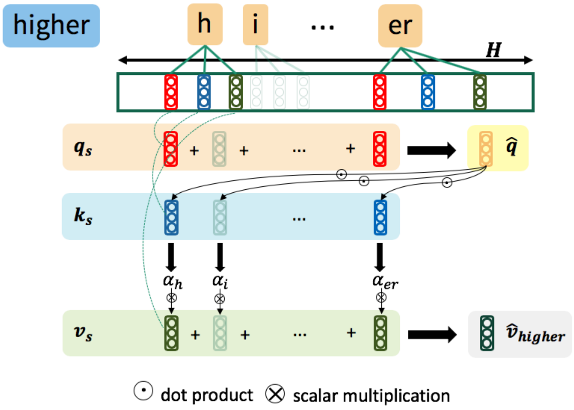


Figure 2: Illustration of how our KVQ self-attention operation calculates each word embedding.

$$\eta_{v,FH}(\cdot) : \mathcal{S}_{\mathcal{F}} \rightarrow \mathcal{I}_{s,H} \;\text{ where }\; \mathcal{I}_{s,H} = \{1, \ldots, H\}. \tag{10}$$

5.4 Attention operation: Modification of $\tau(\cdot)$

Previous researches such as `fastText` and `BoS` treat $\tau(\boldsymbol{V}, w)$ as a summation of all subword embeddings described by Eq. 6. However, the summation is less expressive and it may lack capability in a memory-sharing setting since subwords share their embeddings randomly. One possible improvement is to handle the importance of each subword based on a given word during the calculation of $\Phi(\boldsymbol{V}, w)$.

A simple approach to deal with this phenomenon is to incorporate a “context-dependent” weighting factor for each subword in a given word. Thus, we consider the following subword mixing function $\tau_{\text{kvq}}(\boldsymbol{V}, w)$ as:

$$\tau_{\text{kvq}}(\boldsymbol{V}, w) = \sum_{s \in \phi(w)} a_{s,w} \boldsymbol{v}_s \tag{11}$$

where $a_{s,w}$ represents a context-dependent weighting factor of the subword s, where the “context” here means all the subwords obtained from word w.

To calculate $a_{s,w}$, we first introduce $\boldsymbol{k}_s$ and $\boldsymbol{q}_s$, which are similarly defined to $\boldsymbol{v}_s$ in Eq. 4, namely, $\boldsymbol{k}_s = \boldsymbol{V}[z_k]$, where $z_k = \eta_k(s)$ and $\boldsymbol{q}_s = \boldsymbol{V}[z_q]$, where $z_q = \eta_q(s)$. Similar to $\eta_v(\cdot)$ in Eq. 3, $\eta_k(\cdot)$ and $\eta_q(\cdot)$ are two distinct mapping functions that map a given subword s into a subword index. Then, we introduce a key-value-query (KVQ) self-attention operation inspired by

abbre.	data size	number of OOV data `fastText.600B`	`GloVe.840B`
Word similarity estimation (**WordSim**)			
MEN	3,000	0	0
M&C	30	0	0
MTurk	287	0	0
RW	2,034	37	36
R&G	65	0	0
SCWS	2,003	2	2
SLex	998	0	0
WSR	252	0	0
WSS	203	0	0
Word analogy estimation (**Analogy**)			
GL	19,544	0	0
MSYN	8,000	1000	1000

Table 2: Evaluation datasets used in our experiments. MEM (Bruni et al., 2014), M&C (Miller and Charles, 1991), MTurk (Radinsky et al., 2011), RW (Luong et al., 2013), R&G (Rubenstein and Goodenough, 1965), SCWS (Huang et al., 2012), SLex (Hill et al., 2014), WSR and WSS (Agirre et al., 2009), GL (Mikolov et al., 2013a), and MSYN (Mikolov et al., 2013b).

method	hyper-parameters		$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{S}\rvert$	size (GB)
`fastText.600B`			2M	–	2.23GB
`SUM-F`	$F = 0.5$M	-	2M	0.5M	0.59GB
`SUM-H`	-	$H = 0.5$M	2M	21.8M	0.59GB
`KVQ-H`	-	$H = 0.5$M	2M	21.8M	0.59GB
`SUM-FH`	$F = 1.0$M	$H = 0.5$M	2M	1.0M	0.59GB
`KVQ-FH`	$F = 1.0$M	$H = 0.5$M	2M	1.0M	0.59GB
`SUM-F`	$F = 0.2$M	-	2M	0.2M	0.23GB
`SUM-H`	-	$H = 0.2$M	2M	21.8M	0.23GB
`KVQ-H`	-	$H = 0.2$M	2M	21.8M	0.23GB
`SUM-FH`	$F = 1.0$M	$H = 0.2$M	2M	1.0M	0.23GB
`KVQ-FH`	$F = 1.0$M	$H = 0.2$M	2M	1.0M	0.23GB

Table 3: Statistics for our methods.

`Transformer` (Vaswani et al., 2017), that is,

$$a_{s,w} = \frac{\exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_s)}{\sum_{s' \in \phi(w)} \exp(Z\hat{\boldsymbol{q}} \cdot \boldsymbol{k}_{s'})}, \tag{12}$$

where Z is a scaling hyper-parameter. and $\hat{\boldsymbol{q}} = \sum_{s \in \phi(w)} \boldsymbol{q}_s$. Figure 2 illustrates how our KVQ self-attention operation calculates each word embedding.

6 Experiments

6.1 Evaluation of model shrinkage

This section describes our experiments for evaluating the performance of the model shrinkage.

6.1.1 Settings

Evaluation data: Table 2 shows a summary of the evaluation datasets used in our experiments. We conducted experiments on well-studied linguistic benchmark datasets, i.e., nine for word similarity (**WordSim**) tasks and two for word analogy (**Analogy**) tasks. In this evaluation, we discarded data in the evaluation datasets if at least one of the words in the data was an OOV word. Note that this is the standard evaluation criterion used in the previous studies. By following this criterion, we investigate the effectiveness in terms of model shrinkage since we can fairly compare the performance with the original (word-based) word embeddings.

Pre-trained word embeddings: For the reconstruction target, we selected `fastText.600B`[6]. Note that it achieved the state-of-the-art performance on the WordSim and Analogy datasets (Bojanowski et al., 2017). The hyper-parameters $D = 300$ and $|\mathcal{W}| = 2$M were automatically obtained from the properties of `fastText.600B`.

Hyper-parameters for training: We took advantage of a N-grams' range of $N = 3$ to 30. We adopted Adam (Kingma and Ba, 2014) as our optimization algorithm to minimize Eq. 5. We set the following hyper-parameters for Adam: $\alpha = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. We leveraged a mini-batch training, whose size was 200, and trained each method for 300 epochs. For C_w in Eq. 7, we utilized the occurrence information calculated from a large external corpus. Then, we set $Z = \sqrt{D}$ for all experiments.

Comparison: We compared the following five distinct settings of subword-based reconstruction of word embeddings.

1. **SUM-F**: Select $\eta_{v,F}(\cdot)$ in Eq. 8 (Section 5.1) for the subword mapping function and $\tau_{\mathtt{sum}}(\cdot)$ in Eq. 6 for the subword mixing function.
2. **SUM-H**: As in the first setting but substitute $\eta_{v,F}(\cdot)$ with $\eta_{v,H}(\cdot)$ in Eq. 9 (Section 5.2).
3. **KVQ-H**: As in the second setting but substitute $\tau_{\mathtt{sum}}(\cdot)$ in Eq. 6 with $\tau_{\mathtt{kvq}}(\boldsymbol{V}, w)$ in Eq. 11 (Section 5.4).
4. **SUM-FH**: As in the second setting but substitute $\eta_{v,H}(\cdot)$ with $\eta_{v,FH}(\cdot)$ in Eq. 10 (Section 5.3).
5. **KVQ-FH**: As in the third setting but substitute $\eta_{v,H}(\cdot)$ with $\eta_{v,FH}(\cdot)$ in Eq. 10 (Section 5.3).

6.1.2 Results

Figures 3 show the performance/model size (or performance/number of embedding vectors)

---

[6] Additionally, we also conducted the same experiments using `GloVe.840B` instead of `fastText.600B`. See Appendix A for the results.

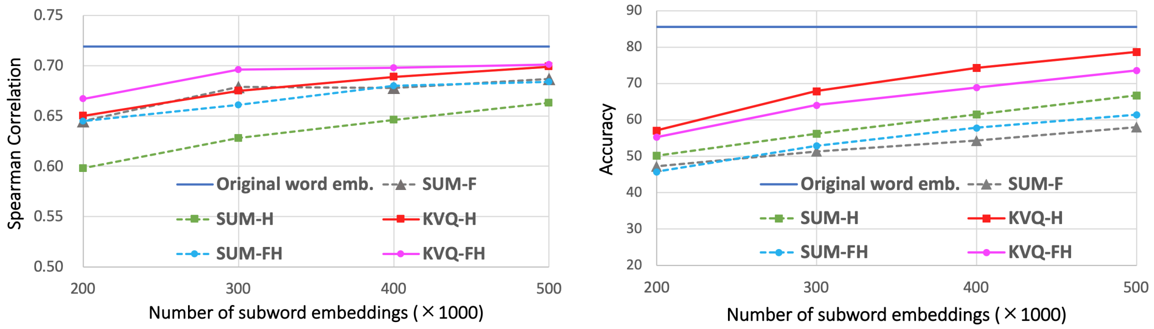


Figure 3: Performance/model size curves for WordSim (left) and Analogy (right). The x-axis represents the number of subword embeddings. The y-axis represents the performance evaluated by macro-average of Spearman's rho (left) and micro-average accuracy (right), respectively.

			WordSim										Analogy		
method	hyper-parameters		MEN	MC	MTurk	RW	R&G	SCWS	Slex	WSR	WSS	Macro	GL	MSYN	Micro
`fastText.600B`			.815	.850	.735	.572	.871	.684	.471	.640	.835	.719	84.9	87.8	85.6
`SUM-F`	$F = 0.5$M	-	.768	.829	.746	.566	.817	.668	.402	.578	.806	.687	53.3	71.2	58.0
`SUM-H`	-	$H = 0.5$M	.740	.824	.708	.533	.811	.652	.367	.536	.797	.663	61.9	79.9	66.7
`KVQ-H`	-	$H = 0.5$M	789	**.848**	.734	.562	**.836**	**.687**	.423	.597	.811	.699	**76.4**	**85.2**	**78.7**
`SUM-FH`	$F = 1.0$M	$H = 0.5$M	.772	.802	.743	**.569**	.813	.664	.397	.578	.816	.684	56.4	75.4	61.4
`KVQ-FH`	$F = 1.0$M	$H = 0.5$M	**.799**	.812	**.749**	.564	.816	.684	**.425**	**.630**	**.826**	**.701**	70.9	81.2	73.6
`SUM-F`	$F = 0.2$M	-	.731	.809	.710	.526	.777	.642	.369	.482	.762	.645	40.3	66.2	47.2
`SUM-H`	-	$H = 0.2$M	.660	.762	.671	.476	.756	.600	.310	.414	.732	.598	42.9	70.4	50.2
`KVQ-H`	-	$H = 0.2$M	**.773**	**.820**	.694	.523	.793	.652	.360	.500	.778	.650	**50.8**	**74.6**	**61.9**
`SUM-FH`	$F = 1.0$M	$H = 0.2$M	.719	.801	**.734**	.525	.798	.635	.334	.490	.767	.645	39.3	63.9	45.8
`KVQ-FH`	$F = 1.0$M	$H = 0.2$M	.754	.805	.731	**.542**	**.800**	**.667**	**.385**	**.523**	**.797**	**.667**	49.1	72.7	55.3

Table 4: Results of model shrinkage experiments by reconstructing the `fastText.600B` embeddings. Each dataset in WordSim and Analogy was evaluated by Spearman's rho and accuracy, respectively. 'Macro' and 'Micro' represent the macro-average of Spearman's rho over all WordSim datasets and the micro-average of accuracy over all Analogy datasets.

curves for WordSim and Analogy, respectively. Each plot is the macro-average of Spearman's rho (WordSim) or micro-average of accuracy (Analogy) of all evaluation datasets. Moreover, Table 3 shows the statistics of each setting and Table 4 show a summary of the detailed results for all datasets. Overall, we observed a consistent tendency that the methods using KVQ obtained the best performance compared with the methods using SUM. Notably, `KVQ-H` significantly outperformed `SUM-F`, `SUM-H` and `SUM-FH` by more than 10 points in terms of the micro-average accuracy of all Analogy datasets when the model size was 0.5M. The difference between `KVQ-H` and `SUM-H` shows that the approach of memory-shared embeddings is particularly useful when we combined it with the KVQ operation. Moreover, in some cases, we observed that `KVQ-H` achieved the performance of the original word embeddings `fastText.600B` when $H = 0.5$M. This means that `KVQ-H` with $H = 0.5$M with successfully reduced the model size nearly fourfold compared with the original word embeddings while maintaining the original performance.

6.1.3 Analysis

Time efficiency of SUM and KVQ: We investigated the difference in time efficiency between SUM and KVQ. We calculated the time computing word embeddings from subword embeddings using each operation. SUM and KVQ took 5.7×10^{-6} and 1.6×10^{-5} seconds per word, respectively, i.e., KVQ took 2.8 times longer than SUM. However, the calculation speed per word is sufficiently high to be negligible in the real applications since other operations such as calculating deep neural networks may take much longer.

6.2 Experiments of OOV word embeddings

This section describes our experiments for evaluating the performance of OOV word embeddings.

6.2.1 Settings

Evaluation data: We used the identical nine WordSim datasets used in Section 6.1.

method	hyper-parameters		Macro
`Random`	-	-	.110
`SUM-F`	$F = 0.5$M	-	**.640**
`SUM-H`	-	$H = 0.5$M	.609
`KVQ-FH`	-	$H = 0.5$M	.598
`SUM-FH`	$F = 1.0$M	$H = 0.5$M	.626
`KVQ-FH`	$F = 1.0$M	$H = 0.5$M	.636
`SUM-F`	$F = 0.2$M	-	.611
`SUM-H`	-	$H = 0.2$M	.552
`KVQ-H`	-	$H = 0.2$M	.566
`SUM-FH`	$F = 1.0$M	$H = 0.2$M	.609
`KVQ-FH`	$F = 1.0$M	$H = 0.2$M	**.614**

Table 5: Results of (synthetic) OOV word experiments on WordSim by reconstructing the `fastText.600B` embeddings. The performance was evaluated by Spearman's rho.

method	hyper-parameters		$\|\mathcal{W}\|$	$\|\mathcal{S}\|$	RW
`Random`	-	-	0.16M	-	.452
`MIMICK`	-	-	0.16M	<1K	.201
`BoS`	-	-	0.16M	0.53M	.46*
`SUM-F`	$F = 0.04$M	-	0.16M	0.04M	.513
`SUM-H`	-	$H = 0.04$M	0.16M	2.03M	.485
`KVQ-H`	-	$H = 0.04$M	0.16M	2.03M	.509
`SUM-FH`	$F = 0.50$M	$H = 0.04$M	0.16M	0.50M	.488
`KVQ-FH`	$F = 0.50$M	$H = 0.04$M	0.16M	0.50M	**.522**
`fastText`	-	-	0.16M	0.53M	.48*

Table 6: Results of OOV experiments on the Stanford Rare Word dataset. * indicates the values reported by Zhao et al. (2018). Note that `fastText` learned subword embeddings from an English Wikipedia dump since this method is not a reconstruction method.

Preparation of training data: As we showed in Table 2, the numbers of OOV problems for `fastText.600B` and `GloVe.840B` are indeed very small. This is because their vocabulary sizes exceeds 2 million words, and the words contained in the evaluation datasets tend to be 'non-rare words' in general. Therefore, it is difficult to precisely evaluate the effectiveness of the estimation of OOV word embeddings.

To overcome this difficulty, we artificially made our reconstruction problem much more difficult, namely, we discarded the words contained in the evaluation datasets from the vocabulary $\mathcal{W}$ for training. This means that all the problems in the evaluation datasets now became OOV problems. In other words, the number of OOV data in Table 2 in this setting always matches to the evaluation data size, such as 2034 for RW.

Other settings: We used the same experimental settings as used in Section 6.1 unless otherwise specified. For example, we used `fastText.600B` as the reconstruction target and the same training hyper-parameters.

6.2.2 Results

Table 5 shows the results of the (synthetic) OOV word experiments. First, we observed that the performance of the `Random` baseline's was nearly equal to zero across all the datasets. This means that there is no correlation between the `Random` and human-annotated scores. Importantly, the performances of `KVQ-FH`, `SUM-FH` and `SUM-F` were significantly improved by 44-53 points from that of `Random`. This result indicates that `KVQ-FH`, `SUM-FH` and `SUM-F` successfully predicted the OOV word embeddings.

However, we also observed no significant difference between `KVQ-FH`, `SUM-FH` and `SUM-F` for both the $H = 0.5$M and $H = 0.2$M settings.

6.2.3 Comparison with previous studies

As we discussed in Section 2, several closely related methods have also tackled to solve the OOV word issue, such as `MIMICK` and `BoS`. We aim to directly compare our approach with these methods to investigate whether it can outperform them. However, these methods (or available authors' codes) do not work on the large-vocabulary settings employed in Sections 6.1 and 6.2. Thus, as an alternative, we strictly followed the experimental settings described in (Zhao et al., 2018) and compared the performance under fair conditions.

Evaluation data: In their evaluation setting, they evaluated the OOV word performance over RW shown in Table 2. They included all the words appearing in RW as the evaluation data, in contrast to discarding the OOV data as in Section 6.1.

Pre-trained word embeddings: The target word embeddings for the reconstruction were the embeddings trained on Google News with 100 billion tokens[7] that were pre-cleaned by (Zhao et al., 2018). The resultant embeddings consist of 0.16M lower-cased word embeddings.

Comparison: We compared our approach with the following related methods:

1. `Random`: the performance when we used random vectors for OOV words.
2. `MIMICK`[8] (Pinter et al., 2017).
3. `BoS`[9] (Zhao et al., 2018).

Other settings: The shared memory size H was set to 0.04M since the vocabulary of this setting

---

[7] `https://code.google.com/archive/p/word2vec/`

[8] `https://github.com/yuvalpinter/Mimick`

[9] `https://github.com/jmzhao`

method	hyper-parameters		K	size (GB)	F1
`fastText.600B`	-	-	20	2.23GB	90.3
`KVQ-FH`	F = 1.0M	H = 0.5M	100	0.59GB	**90.4**
`KVQ-FH`	F = 1.0M	H = 0.2M	100	0.23GB	89.3
`GloVe.840B`	-	-	10	2.45GB	**90.8**
`KVQ-FH`	F = 1.0M	H = 0.5M	100	0.59GB	90.6
`KVQ-FH`	F = 1.0M	H = 0.2M	50	0.23GB	90.2

Table 7: Results of the NER experiments on the CoNLL-2003 dataset.

method	hyper-parameters		K	size (GB)	Acc
`fastText.600B`	-	-	10	2.23GB	87.8
`KVQ-FH`	F = 1.0M	H = 0.5M	20	0.59GB	**88.0**
`KVQ-FH`	F = 1.0M	H = 0.2M	10	0.23GB	87.6
`GloVe.840B`	-	-	1	2.45GB	**88.3**
`KVQ-FH`	F = 1.0M	H = 0.5M	10	0.59GB	87.8
`KVQ-FH`	F = 1.0M	H = 0.2M	20	0.23GB	87.6

Table 8: Results of the TE experiments on the SNLI dataset.

was relatively very small compared with that in our experiments, i.e., 2M vs 0.16M.

Results: Table 6 shows a comparison with the related methods. All our reconstruction methods outperformed `BoS`, which was the previous state-of-the-art method, with substantial improvements by 2-6 points. Moreover, `KVQ-FH` achieved the best performance in this comparison.

6.3 Evaluation on downstream tasks

To investigate the effectiveness of our reconstucted embeddings in downstream tasks, we evaluated them in the named entity recognition (NER) and the textual entailment (TE) tasks.

6.3.1 Settings

Evaluation data: We used the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003) for an NER experiment and the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) for a TE experiment.

Other settings: We used `fastText.600B` and `GloVe.840B` as the target word embeddings for the reconstruction. For our reconstruction embeddings, we calculated the embeddings of all the words in the datasets, thus there exist no OOV words when using our methods.

We used AllenNLP[10] to train base NER and TE models. We basically used the provided hyper-parameter values in their repository for both training and testing. Additionally, we added one hyper-parameter K to re-scale embeddings (i.e., multiply all the elements in the embeddings by K) since we learned that the re-scaling may significantly affect the overall performance of downstream tasks in certain situation. We search K from $[1, 5, 10, 20, 50, 100]$ on the validation set of each dataset.

6.3.2 Results

Tables 7 and 8 show the comparison between the original (large) embeddings and our reconstructed (small) embeddings. When $H = 0.5$M, the performances of our reconstructed embeddings are equivalent to or even better than the ones of the original embeddings. One might be surprised the improved results by `KVQ-FH` since the model sizes of `KVQ-FH` were relatively very small comparing with the original embeddings. However, this may be a reasonable observation since our method additionally offered the embeddings of OOV words that cannot be handled by the original embeddings. Moreover, even when $H = 0.2$M, i.e. the model size was approximately ten times smaller, the degradation was less than 1.0, which is considered to be sufficiently acceptable for real-world systems if we can significantly reduce the model (system) size.

7 Conclusion

We discussed and investigated an approach that reconstructs subword-based word embeddings in a reduced memory space. We demonstrated that memory-shared embeddings with the KVQ self-attention operation significantly outperformed the conventional summation-based approach, such as `BoS`. Moreover, our best setting successfully reduced the number of embedding vectors to approximately ten times smaller than that of the original word embeddings while maintaining an acceptable performance loss on the downstream tasks. We also confirmed the effectiveness of our approach in terms of the applicability of OOV words. We believe that our reconstructed subword-based word embeddings can be better alternatives of `fastText.600B` and `GloVe.840B` because they require less memory requirement and have high applicability of OOV words.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 15H01702 and JST CREST Grant Number JPMJCR1513, Japan.

---

[10] `https://allennlp.org/`

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 19–27.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *J. Artif. Int. Res.*, 49(1):1–47.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 731–742.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1314–1324.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. Amr dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1831–1841.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 304–309.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *arXiv preprint arXiv:1408.3456*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 873–882.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 746–751.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes*, 6(1):1–28.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Represqentation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 2227–2237.

Mohammad Taher Pilehvar and Nigel Collier. 2017. Inducing embeddings for rare and unseen words by leveraging lexical resources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 388–393.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 102–112.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 337–346.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Commun. ACM*, 8(10):627–633.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5027–5038.

Jun Suzuki and Masaaki Nagata. 2016. Learning compact neural word embeddings by parameter space sharing. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2046–2052.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An empirical study of building a strong baseline for constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 612–618.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL)*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural rst parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 559–570.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 601–606.

Bayesian Learning for Neural Dependency Parsing

Ehsan Shareghi,[♣] **Yingzhen Li,**[♠] **Yi Zhu,**[♣] **Roi Reichart,**[◇] **Anna Korhonen**[♣]
[♣] Language Technology Lab, DTAL, University of Cambridge
[♠]Microsoft Research Cambridge, [◇]Technion, IIT
[♣]`{es776, yz568, alk23}@cam.ac.uk`
[♠]`Yingzhen.Li@microsoft.com`, [◇]`roiri@technion.ac.il`

Abstract

While neural dependency parsers provide state-of-the-art accuracy for several languages, they still rely on large amounts of costly labeled training data. We demonstrate that in the small data regime, where uncertainty around parameter estimation and model prediction matters the most, Bayesian neural modeling is very effective. In order to overcome the computational and statistical costs of the approximate inference step in this framework, we utilize an efficient sampling procedure via stochastic gradient Langevin dynamics to generate samples from the approximated posterior. Moreover, we show that our Bayesian neural parser can be further improved when integrated into a multi-task parsing and POS tagging framework, designed to minimize task interference via an adversarial procedure. When trained and tested on 6 languages with less than $5k$ training instances, our parser consistently outperforms the strong BiLSTM baseline (Kiperwasser and Goldberg, 2016). Compared with the BiAFFINE parser (Dozat et al., 2017) our model achieves an improvement of up to 3% for Vietnamese and Irish, while our multi-task model achieves an improvement of up to 9% across five languages: Farsi, Russian, Turkish, Vietnamese, and Irish.

1 Introduction

Dependency parsing is essential for many Natural Language Processing (NLP) tasks (Levy and Goldberg, 2014; Angeli et al., 2015; Toutanova et al., 2016; Hadiwinoto and Ng, 2017; Marcheggiani et al., 2017). While earlier work on dependency parsing required careful feature engineering (McDonald et al., 2005b; Koo et al., 2008), this has become less of a concern in recent years with the emergence of deep neural networks (Kiperwasser and Goldberg, 2016; Dozat et al., 2017). Nonetheless, an accurate parser still requires a large amount of labeled data for training, which is costly to obtain, while the lack of data often causes overfitting and poor generalization.

Several approaches for parsing in the small data regime have been proposed. These include augmenting input data with pretrained embedding (Dozat et al., 2017; Che et al., 2018), leveraging unannotated data via semi-supervised learning (Corro and Titov, 2018), predicting based on a pool of high probability trees (Niculae et al., 2018; Keith et al., 2018), and transferring annotation or model across languages (Agic et al., 2016; Lacroix et al., 2016; Rasooli and Collins, 2017). Despite the empirical success of these approaches, an inherent problem still holds: The maximum likelihood parameter estimation (MLE) in deep neural networks (DNNs) introduces statistical challenges at both estimation (training), due to the risk of overfitting, and at test time as the model ignores the uncertainty around the estimated parameters. When training data is small these challenges are more pronounced.

The Bayesian paradigm provides a statistical framework which addresses both challenges by (i) including prior knowledge to guide the learning in the absence of sufficient data, and (ii) predicting under the full posterior distribution of model parameters which offers the desired degree of uncertainty by exploring the posterior space during inference. However, this solution comes with a high computational cost, specifically in DNNs, and is often replaced by regularization techniques such as dropout (Srivastava et al., 2014) as well as ensemble learning and prediction averaging (Liu et al., 2018; Che et al., 2018).

Bayesian neural networks (BNNs) have attracted some attention (Welling and Teh, 2011; Hernández-Lobato and Adams, 2015; Li et al., 2016; Gong et al., 2018). Yet, its current application to NLP is limited to language modeling (Gan et al., 2017), and BNNs have not been developed for structured

Proceedings of NAACL-HLT 2019, pages 3509–3519
Minneapolis, Minnesota, June 2 - June 7, 2019.

prediction tasks such as dependency parsing.

In this paper we aim to close this gap and propose the first BNN for dependency parsing (BNNP). To address the costs of inference step, we apply an efficient sampling procedure via stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011). At training, samples from the posterior distribution of the parser parameters are generated via controlled noise injection to the maximum a posteriori (MAP) gradient update. The generated samples are then used during the inference step to create multiple viable parses based on which the final dependency parse is generated.

Another means of directing a model towards more accurate predictions in the small data regime is via multi-task learning (Caruana, 1997). Such a framework allows models for multiple tasks to reinforce each other towards more accurate joint solutions, which is particularly useful when training data is scarce. We hence present a multi-task framework where our BNNP is integrated with a POS tagger through an adversarial procedure designed to guide the two models towards improved joint solutions.

Our experiments with monolingual and delexicalized cross-lingual parsing using the Universal Dependency treebank (Zeman et al., 2017) demonstrate the effectiveness of our approach. Particularly, our BNNP consistently outperforms the single task BiLSTM baseline (Kiperwasser and Goldberg, 2016), while outperforming the BiAFFINE parser (Dozat et al., 2017) by up to 3% on Vietnamese and Irish. Additionally, our multi-task model achieves an improvement of up to 9% over the BiAFFINE parser for five low-resource languages: Farsi, Russian, Turkish, Vietnamese, and Irish.

2 BiLSTM Dependency Parsing

Our parser extends the graph-based BiLSTM parser of Kiperwasser and Goldberg (2016). We briefly review their work and its notable extensions, and then discuss our extension of their architecture and the limitations of MLE training.

Model Given an input word sequence $w_{1:n}$,[1] and its corresponding POS tag sequence $p_{1:n}$, each input word is encoded as a d-dimensional representation,

$$w_i = e(w_i) \circ e(p_i) \circ \hat{e}(w_i),$$

where $e(.)$ denotes trainable embedding, $\hat{e}$ denotes pretrained external embedding, and $\circ$ denotes concatenation. The context dependent representation of each word is generated through a BiLSTM,

$$v_i = \overrightarrow{\text{LSTM}}_{\zeta}(w_{1:i}) \circ \overleftarrow{\text{LSTM}}_{\phi}(w_{n:i}).$$

Here, ζ and ϕ denote the LSTMs' parameters. The resulting sequence of l-dimensional vectors, $v_{1:n}$, is then used for computing the ARC-SCORE matrix $(n \times n)$ where cell ARC-SCORE(i, j) is computed as,

$$V \times \tanh\Bigg(\underbrace{W^{(arc-head)} v_i}_{\text{head specialized } w_i} + \underbrace{W^{(arc-mod)} v_j}_{\text{modifier specialized } w_j} + b \Bigg),$$

indicating the likelihood of having the arc from w_i to w_j. Here, V is $(1 \times h)$, b is $(h \times 1)$, and $W^{(.)}$ is a $(h \times l)$ matrix used for creating specialized versions of words in $w_{1:n}$. The h-dimensional specialized representation of w_i as a head or a modifier is generated via multiplication with $W^{(arc-head)}$ and $W^{(arc-mod)}$, respectively. Following a first-order dependence assumption between the arcs, the parsing is done via a dynamic programming solution that finds the dependency parse T^* such that,

$$T^* = \underset{T}{\text{argmax}} \Bigg(\text{SCORE}(T) = \sum_{(i,j) \in T} \text{ARC-SCORE}(i, j) \Bigg).$$

Next, given T^*, for each arc $(i, j) \in T^*$ the LABEL-SCORE(i, j, r) is computed as:

$$\Bigg(V' \times \tanh(\underbrace{W'^{(arc-head)} v_i}_{\text{head specialized } w_i} + \underbrace{W'^{(arc-mod)} v_j}_{\text{modifier specialized } w_j} + b') \Bigg)[r],$$

where r is a dependency relation type, $r \in \{r_z\}_{z=1}^{\mathbf{r}}$, V' is $(\mathbf{r} \times h)$, b' is $(\mathbf{r} \times 1)$, and $W'^{(.)}$ is $(h \times l)$. The label for each dependency arc (i, j) is then chosen by a max operation.

Dozat et al. (2017) proposed an extension of the BiLSTM parser by replacing the non-linear transformation of ARC-SCORE and LABEL-SCORE with a linear transformation (BiAFFINE). This was further extended by Che et al. (2018) who utilized contextualized word embeddings (Peters et al., 2018). Both extensions showed success in dependency parsing shared tasks (Zeman et al., 2017, 2018).

Our Neural Parser Architecture Our network architecture extends the BiLSTM model with an additional BiLSTM layer and input signals. While our

---

[1] We overload the notation w_i, and it refers to both a word and its d-dimensional representation.

architecture is not the core contribution of this paper, we aim to implement our BNNP on a strong architecture. In §5.3 we demonstrate the contribution of these additions to our final results.

In our BNNP, each word is represented as,

$$w_i = e(w_i) \circ e(p_i) \circ \hat{e}(w_i) \circ e(c_i),$$

where, similar to the BiAFFINE parser, $e(c_i)$ is a character-level representation of the word w_i, generated by a BiLSTM:

$$e(c_i) = \overrightarrow{\text{LSTM}}_\alpha(c_{1:|w_i|}) \circ \overleftarrow{\text{LSTM}}_\beta(c_{|w_i|:1}).$$

We consider a maximum of two layers of BiLSTMs, denoted as *1st* and *2nd*.[2] The output of the *2nd* BiLSTM is denoted as,

$$u_i = \overrightarrow{\text{LSTM}}_\kappa(v_{1:i}) \circ \overleftarrow{\text{LSTM}}_\delta(v_{n:i}),$$

where v_i is the encoding generated by the *1st* layer of the BiLSTMs.

Learning For structured prediction, the objective function is to maximize the margin between the gold structure $\mathbb{T}$ and the other structures. To achieve this, the cost augmented hinge loss is used, where the fixed margin m is replaced by a cost function, $\text{COST}(\mathbb{T}, T)$, sensitive to the local mistakes (here, each local mistake has a constant cost of 1), and computed as:

$$\mathcal{L}_{arc} = \max\Big(0, \text{COST}(\mathbb{T}, T) + \text{SCORE}(T) - \text{SCORE}(\mathbb{T})\Big)$$

For label prediction, hinge loss with $m = 1$ is used (denoted by $\mathcal{L}_{rel}$). We refer to the parser loss as:

$$\mathcal{L}_{parse} = \mathcal{L}_{arc} + \mathcal{L}_{rel}.$$

Beyond MLE Training The point-estimate of DNN parameters is computationally efficient, but ignores the uncertainty around model parameters during learning. This results in an overconfidence over model predictions during the inference phase. A common generic practice to incorporate a degree of uncertainty is to consider an ensemble of models. Indeed, for dependency parsing ensemble learning has shown to improve accuracy (Surdeanu and Manning, 2010; Kuncoro et al., 2016; Che et al., 2018). However, ensembles are computationally demanding due to the large number of participating models. The de-facto approach to overcome this has been to randomly perturb the structure of the network for each training instance by switching off connections between the nodes, a practice known as dropConnect (Wan et al., 2013), or eliminating the nodes entirely, which is known as dropout (Srivastava et al., 2014).

Hinton et al. (2012) demonstrated that dropping out a node with probability ρ from a neural model $\mathcal{M}$ at training, and scaling the same node's output at test time (in the fully connected $\mathcal{M}$), performs very well in practice. While this scaling trick avoids the need for running multiple models at test time, it lacks theoretical guarantees when applied to DNNs. Gal and Ghahramani (2016) showed that dropout can be casted as integrating out the posterior of model parameters, and in this regard is a form of Bayesian approximation. Still, dropout is not as effective when applied to the small data regime (Goodfellow et al., 2016). We hence propose a fully Bayesian approach.

3 Bayesian Neural Parsing

The Bayesian paradigm is appealing in its ability to capture uncertainty around parameters, avoid overfitting, and incorporate prior knowledge to compensate for the lack of sufficient training data in resource lean settings. Dependency parsing may be a natural candidate for Bayesian modeling since a typical sentence can have multiple viable parses corresponding to different grammatical ambiguities, and considering these possibilities have shown to improve predictive accuracy (Niculae et al., 2018; Keith et al., 2018). In the same spirit, Bayesian inference can be interpreted as a statistically grounded means to generate this pool of high quality trees.

In detail, Bayesian inference for parsing computes the *predictive distribution* $P(T|s, \mathcal{D})$[3] of the dependency parse tree T for an input sentence s given the training data $\mathcal{D}$ by *posterior averaging*:

$$P(T|s, \mathcal{D}) = \int P(T|s, \theta) P(\theta|\mathcal{D}) d\theta,$$

where $P(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$ denotes the posterior distribution of the BNNP parameters θ given observations $\mathcal{D}$. Here the prior distribution $p(\theta)$ is a standard Gaussian $\mathcal{N}(0, I)$, and, by assuming i.i.d training datapoints, the likelihood term

---

[2] Stacking BiLSTMs is believed to be helpful. In our case, the addition of a third layer led to overfitting.

[3] When $P(T|s, \theta)$ is replaced by the unnormalized score from the first-order arc-factored model, $P(T|s, \mathcal{D})$ can be interpreted as the expected score assigned to a dependency tree under the full posterior distribution of model parameters.

is $P(\mathcal{D}|\theta) = \prod_{x_i \in D} P(x_i|\theta)$ where $x_i = (s_i, T_i)$ is a training instance.

The above integral is intractable, however an empirical estimate can be computed using $\mathcal{K}$ samples from the posterior,

$$P(T|s,\mathcal{D}) \approx \frac{1}{\mathcal{K}} \sum_{\{\theta_k \sim P(\theta|D)\}_{k=1}^{\mathcal{K}}} P(T|s,\theta_k). \tag{1}$$

Therefore, efficient generation of posterior samples is the key for applying the Bayesian treatment. Since the posterior is intractable in DNNs, additional machineries such as Markov Chain Monte Carlo (MCMC) (Robert and Casella, 2010) are required for sampling. In this paper we explore an efficient class of Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) methods, designed for NNs by adjusting stochastic gradient descent with properly scaled Gaussian noise to generate posterior samples.

3.1 Stochastic Gradient Langevin Dynamics

We first consider the Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) sampler to generate posterior samples. Intuitively, this mechanism guides SGD to explore the posterior distribution rather than finding the maximum a posteriori (MAP) solution. More concretely, at each time step t we first compute a stochastic estimate of the gradient for the negative log-posterior distribution on a mini-batch b_t:

$$\nabla_{\theta_t}\mathcal{L} \approx -\left(\nabla_{\theta_t}\log P(\theta_t) + \frac{|\mathcal{D}|}{|b_t|}\sum_{x_i \in b_t}\nabla_{\theta_t}\log P(x_i|\theta_t)\right),$$

and then update the BNNP parameters by SGD with noise injection,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta_t}{2}\nabla_{\theta_t}\mathcal{L} + \xi_t, \quad \xi_t \sim \mathcal{N}(0, \eta_t \mathbf{I}) \tag{2}$$

Theoretically, the distribution of θ_t converges to the true posterior $p(\theta|\mathcal{D})$ when $b_t = \mathcal{D}$, $t \to \infty$ and $\eta_t \to 0$ (see Teh et al. (2016) for SGLD convergence rate proofs and analysis).

In practice, the learning rate $\eta_t = a(b+t)^{-\gamma}$, starting high and decaying, plays a crucial role in eqn. 2 dynamics. Intuitively, while the variance of the injected noise is η_t, the variance of the gradient is of $\mathcal{O}(\eta_t^2)$. In the earlier stages of the optimization, the $\frac{\eta_t}{2}\nabla_\theta \mathcal{L}$ term is dominant as gradients are of greater impact and the learning rate is high, simulating stochastic gradient descent. As the optimization proceeds, the impact of the gradient shrinks and the injected Gaussian noise becomes the dominant term, transitioning from SGD to Langevin Monte Carlo (Neal, 2011).

Since $\eta_t \to 0$ might cause slow mixing in practice, we also set a minimum threshold for the learning rate to allow mixing at later stages. SGLD and SGD have the same time complexity as sampling occurs along with parameter updates with a negligible overhead of drawing ξ_ts.

3.2 Preconditioned SGLD

As SGLD relies on a single learning rate along all dimensions of θ, it has the same potential inefficiencies of SGD for optimizing functions with different curvatures along each dimension. Inspired by more advanced optimization techniques that adjust to the geometry of parameter space, Li et al. (2016) proposed preconditioning of SGLD (similar to RMSprop (Tieleman and Hinton, 2012)) which scales the gradients with respect to the weighted average of the gradients along each dimension, such that a unified learning rate is sufficient. The proposed preconditioner diagonal matrix is computed as follows,

$$G(\theta_t) = \mathrm{diag}(1 \oslash (\lambda 1 + \sqrt{V(\theta_t)})),$$
$$V(\theta_t) = \alpha V(\theta_{t-1}) + (1-\alpha)\nabla_{\theta_t}\mathcal{L} \odot \nabla_{\theta_t}\mathcal{L},$$

where $\odot, \oslash$ are element-wise matrix product and division, and λ, α are hyperparameters controlling the extremes of the curvature, and weighting average of previous and current gradients, respectively. The updated preconditioned eqn. 2 is,

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta_t}{2}G(\theta_t)\nabla_{\theta_t}\mathcal{L} + \xi_t, \quad \xi_t \sim \mathcal{N}(0, \eta_t G(\theta_t)).$$

Here, replacing the preconditioner matrix $G(\theta_t)$ with an identity matrix will recover the SGLD update formulation. In case of a diagonal preconditioner, it has the same time complexity of SGLD, although the constant factors in the complexity figure can be slightly higher due to additional computations involved in $G(\theta_t)$ and $V(\theta_t)$.

While SGLD is designed to explore different modes of the posterior distribution, we found (§5.4) that in practice we still require dropout to facilitate modes exploration via random perturbation of the model structure.

3.3 Prediction

The generated posterior samples can be used to compute the approximate expectation of eqn. 1 by scoring all plausible parses via explicit model averaging.

As this solution is not computationally feasible, we use the sampled parameters and follow a procedure that minimizes the Bayes risk (MBR) (Goodman, 1996). Given each sampled parameter, first we generate the maximum scoring parse using the arc-factored decomposition (McDonald et al., 2005a) and dynamic programming (Eisner, 1996). This can be done concurrently for all samples, resulting in a running time identical to the non-Bayesian approach. For each labelled edge, we replace its score in the ARC-SCORE matrix with its occurrence count in the collection of sampled trees and infer the final tree using counts as scores. The predicted structure is then passed to the label predictor, which assigns labels to the edges (§2). This decoding approach, while selecting the global structure with the highest probability under the approximate posterior, could potentially allow for additional corrections of the highest scoring tree in the pool of samples (Shareghi et al., 2015; Kuncoro et al., 2016).

4 Multi-Task Learning

Multi-task learning (Caruana, 1997) lends itself as a natural choice for low-resource settings as it aims at leveraging the commonality between tasks to improve their performance in the absence of sufficient amount of training data. This framework hence naturally complements Bayesian modeling in dealing with the challenges of the small data regime.

We couple our BNNP with POS tagging due to the strong connection between the two tasks (Rush et al., 2010) and the availability of joint training data in several languages (Zeman et al., 2017). While multi-task frameworks have shown success in some areas (Reichart et al., 2008; Finkel and Manning, 2009; Liu et al., 2016; Malca and Reichart, 2018), in our case we found that our two tasks interfered with each other and degraded the parser performance (see similar findings for other tasks at Søgaard and Goldberg (2016); Plank and Alonso (2017)).

To minimize task interference, an approach shown effective (Ganin and Lempitsky, 2015; Kim et al., 2017; Chen et al., 2017; ZareMoodi and Haffari, 2018) is to implicitly guide the update signals during training via an adversarial procedure that avoids shared parameters contamination. We adapt this idea to our multi-task learning.

Architecture Given an input word, w_i, the score for a POS tag p is computed as $\tanh(W''v_i + b'')[p]$, where $p \in \{p_z\}_{z=1}^{\mathbf{p}}$, W'' is $(\mathbf{p} \times l)$, and b'' is $(\mathbf{p} \times 1)$. To train the POS tagger, the cross-entropy loss is used (denoted by $\mathcal{L}_{pos}$). Taking w_is as input, the *shared* BiLSTM strictly encodes a task-agnostic l-dimensional representation of the input (explained in the next subsection),

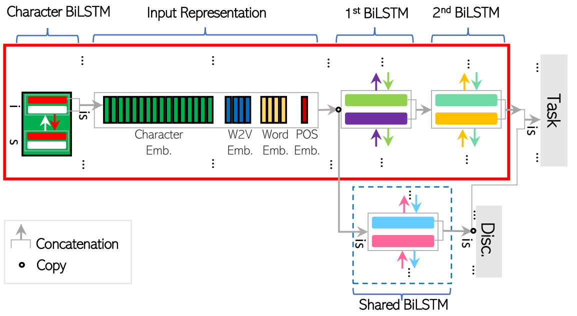


Figure 1: The **MULTI TASK** architecture flow for a single word in a sequence.

$$s_i = \overrightarrow{\text{LSTM}}_\rho(w_{1:i}) \circ \overleftarrow{\text{LSTM}}_\chi(w_{n:i}).$$

The input to each task is then defined as the concatenation of task-agnostic and task-specific representations. We considered a basic architecture where both tasks have their separate parameters (identical number of layers, dimensions, etc.) and they only share the *shared* BiLSTM (denoted as **MULTI TASK** in Table 1). [4]

Adversarial Training The *shared* BiLSTM output, s_i, is meant to encode a task-agnostic representation of the input word w_i. In order to enforce this criterion, we apply an adversarial training procedure. The shared representation, s_i, is forwarded to a task discriminator (denoted as *Disc.* in Figure 1) through a gradient reversal layer. The task discriminator predicts the task identity for each word in the input via a linear transformation $(W'''s_i + b''')[\tau]$ followed by a softmax, where $\tau \in \{\texttt{parser}, \texttt{tagger}\}$ and W''' and b''' are $(2 \times l)$ and (2×1), respectively. To train the discriminator, a sum of the cross-entropy losses for $w_{1:n}$ is used (denoted by $\mathcal{L}_{disc}$). As the parameters of the discriminators are being updated, the gradient signals to minimize the discriminator's error are backpropagated with an opposite sign to the *shared* BiLSTM layer, which adversarially encourages the *shared* BiLSTM to fool the discriminator.

Our training schedule alternates between the two modes, in one mode optimizing the shared and task-specific parameters based on $\mathcal{L}_{parse}$ and $\mathcal{L}_{pos}$ (in

---

[4] We also tried layer-wise placements of tasks (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) and the results were slightly worse. Details are omitted for space reason.

random order), while in the other mode optimizing $\mathcal{L}_{disc}$ which includes the shared and discriminator-specific parameters.

5 Experiments and Results

We experiment with mono-lingual and cross-lingual dependency parsing using the treebanks of the CoNLL 2017 shared task on parsing to Universal Dependencies (UD) (Zeman et al., 2017).[5]

5.1 Experimental Setup

We use the UDPipe baseline outputs for segmentation and POS tagging of the raw test data (released along with the raw test data). While segmentation and POS errors substantially impact the quality of the final predicted parse, their exploration is beyond our scope. Our evaluation metric is Labeled Attachment Score (LAS), computed by the shared task evaluation script. Statistical significance, when mentioned, is computed over 20 runs, via the Kolmogorov-Smirnov test (Reimers and Gurevych, 2017; Dror et al., 2018) with $p = 0.01$.

Mono-Lingual Experiments We experiment with Persian (fa), Korean (ko), Russian (ru), Turkish (tr), Vietnamese (vi) and Irish (ga), all with less than $5k$ training sentences (Table 1). For comparison we report the scores published by the top system of the CoNLL 2017 shared task, BiAFFINE (Dozat et al., 2017), noting the following differences between their input and output and ours. The BiAFFINE parser: (i) uses the UDPipe outputs for segmentation but corrects POS errors before parsing, (ii) includes both language specific and universal POS tags in the input layer while we only include the universal POS tags, and (iii) applies post-process correction for non-projective languages.

Cross-Lingual Experiments We use the English (en), French (fr), Russian (ru), and Persian (fa) datasets of the UD treebanks as our training and test data, with the addition of 3 languages for which we did not have any training data: Kurmanji (kmr), Buriat (bxr), and Northern Sami (sme). We also report the results for each language, where the combination of training datasets for the rest of the languages (marked as +) was used for training. The cross-lingual experiments are done on delexicalized parses after replacing the words with their Universal POS tags.

Models and Baselines - Single-Task We consider the following models: **BASE** is the BiLSTM model of Kiperwasser and Goldberg (2016); **BASE++** extends BASE by having 2 layers of BiLSTMs and using 1 layer of character level BiLSTM (§2); **+SHARED** includes an additional BiLSTM (dashed box in Figure 1). We included this to provide a fair comparison (in terms of the number of parameters) with the multi-task experiments but we apply a higher dropout rate to resolve overfitting; **ENSEMBLE** denotes a collection of 9 +SHARED models each randomly initialized (Reimers and Gurevych, 2017) and trained for MLE with MBR (§3.3) applied for prediction; **MAP** denotes the +SHARED model optimized for MAP instead of MLE; **+SGLD** denotes Bayesian learning and prediction (§3.1), and **+PRECOND** denotes preconditioned SGLD (§3.2). **+SGLD** and **+PRECOND** are applied to the +SHARED model.

Models and Baselines - Multi-Task We consider the same models as in the single-task setup, with the following changes. **BASEMT** is the variant of **MULTI TASK** architecture where the *shared* BiLSTM is removed and all components except for decoders are shared between the two tasks. **+SHARED** and **+ADV** denote the results without and with the adversarial training (§4), respectively.

5.2 Hyperparameters

Training is done for 330 epochs with early stopping, using successive[6] mini-batches of 5000 words,[7] with drop-out rate 0.33 unless stated otherwise. SGLD is only applied for learning the parameters of the parser, with learning rate decaying from 0.01 to 0.0001 ($\gamma = 0.5$), and the preconditioned matrix hyperparameters are $\lambda = 10^{-5}, \alpha = 0.99$. The rest of the parameters are updated using Adam (Kingma and Ba, 2014) with default DyNet parameters (Neubig et al., 2017). The size and selection of the samples used in Bayesian inference are tuned on Irish (see §6.1 of the main paper). In all experiments, training stops based on loss convergence of a single model on the dev set. We also tried GRUs and our results were substantially worse than LSTMs.

We consider the following sizes within the models: **BASE** is 1 layer of BiLSTM with 125 hidden

---

[5] For train and dev sets (1-1983), test set (1-2184), and pretrained embeddings (1-1989) see: `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/{1-1983,1-2184,1-1989}`

[6] We found that random batches perform slightly worse.

[7] When the 5000 words limit lands in the middle of a sentence, the entire sentence is included.

units, 100-D word embeddings, 25-D POS embeddings, and dropout rate of 0.33; **BASE++** extends BASE by having 2 layers of BiLSTMs with 200-D hidden units, 100-D external word embeddings, and using 1 layer of character level BiLSTM with 200-D hidden units to generate 100-D character embeddings (§2); **+SHARED** includes an additional BiLSTM with 200-D hidden units in BASE++ to provide a fair comparison (in terms of the number of parameters) with the multi-task experiments but we apply a higher dropout rate of 0.66 to resolve overfitting.

5.3 Results

Mono-Lingual Parsing Table 1 summarizes the training data statistics and the LAS results of the various models, with bold-font marking cases where a model outperforms the BiAFFINE parser.

	fa	ko	ru	tr	vi	ga
TRAIN(sen.)	4798	4400	3850	3685	1400	566
Labeled Attachment Score (LAS)						
BiAFFINE	86.31	**82.49**	83.65	62.79	42.13	70.06
BASE	80.97	64.76	75.45	52.64	39.36	62.50
SINGLE TASK						
BASE++	83.15	76.70	79.44	58.92	41.03	66.58
+SHARED	83.11	76.32	79.62	58.72	41.10	66.42
ENSEMBLE	84.12	77.28	80.17	59.36	41.89	68.12
MAP	83.59	76.61	79.78	59.13	41.33	66.95
+SGLD	84.98	78.91	80.86	60.53	**43.12**	69.51
+PRECOND	85.76	79.83	81.9	61.71	**44.52**	**70.91**
MULTI TASK						
BASEMT	81.54	75.78	78.61	57.12	40.08	60.51
+SHARED	81.03	75.08	78.64	57.09	40.04	65.24
+ADV	84.93	78.12	81.23	60.66	**43.11**	69.89
ENSEMBLE	85.01	78.31	81.56	60.92	**43.3**	70.00
MAP	85.02	78.26	81.59	60.72	**43.25**	**70.36**
+SGLD	85.89	79.50	83.06	62.13	**44.57**	**72.52**
+PRECOND	**86.75**	80.97	**84.51**	**63.24**	**45.96**	**74.12**

Table 1: Mono-Lingual results. Models that outperform the BiAFFINE parser are highlighted in bold.

1. SINGLE TASK Comparison between BASE++ and +SHARED reveals that the additional BiLSTM in the +SHARED does not improve the results. As expected, model averaging in ENSEMBLE improved the results over the +SHARED model. Additionally, the MAP results show slight improvements compared with +SHARED. Both of these findings suggest that the quality of predictions is likely to improve if some notion of uncertainty (via averaging, or prior insertion) is included.

For Bayesian solutions, both +SGLD and +PRECOND consistently (and in a statistically significant manner) outperform the ENSEMBLE models and the strong BiLSTM baselines on all languages, while PRECOND outperforms BiAFFINE by up to 3% on two languages (vi, and ga). The consistency of improvements as we incorporate richer means of capturing the uncertainty suggests that these gains are independent of our specific choice of neural architecture and that they might also hold for future parsing architectures.

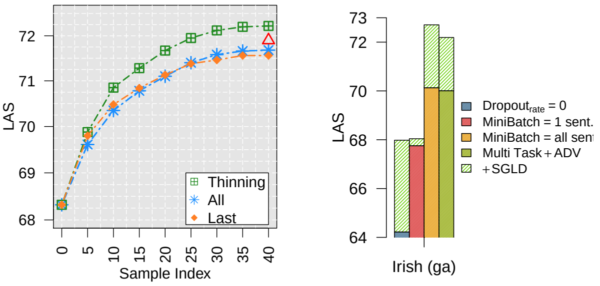
2. MULTI TASK As explained earlier (§4), comparing the parser performances under +SHARED models in single and multi task settings indicates that parser quality was degraded with the inclusion of a POS tagging task, possibly due to interference between the tasks. This issue was alleviated with the inclusion of task discriminator and the adversarial training. The +ADV model consistently improves the results (in a statistically significant manner), while outperforming the **SINGLE TASK (+SHARED)** by 2.2% for Persian (fa), and 5.2% for Irish (ga). This is an indicator that low-resource languages could potentially benefit more from multi-task learning. Other variants of +ADV are also reported in Table 1 and similar patterns to the single task setting are observed.

Next, we test the effectiveness of Bayesian learning and inference using $\mathcal{K} = 9$ thinned samples (§5.4). For all languages in the multi-task setting the gain from including +SGLD in +ADV is statistically significant. However, the gain decays as the amount of training data increases: from 3.8% on Irish (ga) to 1.1% on Persian (fa). This verifies our expectation that Bayesian learning helps parameter estimation and prediction in the small data regime. Compared to +SGLD, +PRECOND provides further improvements (all are statistically significant) of 2.2% on Irish and 1% on Persian, showing a similar generic negative correlation with data size.

To summarize our mono-lingual results, the Bayesian framework shows its merit in improving predictive quality, and multi-task learning introduces new and informative signal via a related task which allows for better parameter estimation. The integration of a Bayesian parser into a multi-task learning framework outperforms the BiAFFINE parser on up to 5 languages, while we observe decreasing improvements as the data size grows. Also, since our Bayesian approach builds on the BASE model, it is subject to the shortcomings of this model: For instance, on Korean (ko) the difference between BASE and BiAFFINE is too large to be closed without further adjustment of model design,

	sen.	en	fr	ru	fa	kmr	bxr	sme
en	13k	81.04	64.34	44.61	24.57	27.62	24.97	33.42
		+0.94	+1.34	+0.99	+1.05	+1.21	+1.39	+1.16
fr	15k	57.71	79.14	47.72	29.23	28.06	18.04	29.72
		+0.89	+1.21	+0.82	+1.02	+1.39	+1.02	+1.01
ru	3k	52.25	62.30	77.55	46.47	27.80	39.29	32.40
		+3.26	+2.58	+3.41	+2.61	+3.14	+2.75	+3.36
fa	5k	37.53	50.94	43.41	78.28	45.29	21.65	23.94
		+1.30	+1.97	+1.82	+1.01	+1.66	+1.36	+1.41
+	36k	82.79	80.92	77.19	78.25	43.26	20.57	35.00
		+0.78	+1.16	+0.87	+0.78	+0.96	+0.94	+0.82

Table 2: Cross-Lingual Parsing Evaluation. LAS of an ensemble of $\mathcal{K} = 9$ **SINGLE TASK** (BASE++) models, and the performance gain by the Bayesian **MULTI TASK** (+PRECOND), placed as subscript.



(a) Performance of different sampling strategies.

(b) Dropout, SGLD, and mini-batch size interdependence.

Figure 2: Ablation analysis for Irish. Similar patterns are observed for the other languages.

or its inputs and outputs.

Cross-Lingual Delexicalized Parsing We tested our most successful mono-lingual model, **MULTI TASK**(+PRECOND), in the cross-lingual setup against an ensemble of $\mathcal{K} = 9$ randomly initialized **SINGLE TASK** (BASE++) models. Table 2 summarizes the training data size (rounded up) and the LAS results of both models.

As expected, the ensemble models perform best when the training and test data come from the same language or language family (as in Buryat to Russian and Kurmanji to Persian). Showing a similar pattern to mono-lingual parsing, in all cases the performance gain of the Bayesian **MULTI TASK** (+PRECOND) setting, reported as under-text in the table, consistently outperforming the ENSEMBLE (in a statistically significant manner) and the impact decays with the training set size.

5.4 Ablation Analysis

We aim to answer two questions: (i) how many of the generated samples are used during inference? and (ii) how much the success of SGLD depends on other sources of noise during training?

1. UTILIZING SAMPLES Once the training stops, different strategies could be applied to utilize the collection of $\mathcal{M}$ posterior samples to recompute ARC-SCORE matrix (see §3.3). An extreme option is including all samples (denoted by *All*), another option is to only use the last sample (denoted by *Last*). An intermediate alternative is to choose $\mathcal{K}$ samples with interval $\lfloor \frac{\mathcal{M}}{\mathcal{K}} \rfloor$, an approach known as *Thinning* (Gan et al., 2017).

Comparing the results in Figure 2a, *Thinning* consistently outperforms the other strategies. Interestingly, when we compared *Thinning* with a variant (denoted by Δ in Figure 2a) that just uses the last 5 samples , it appears that *Thinning* which includes earlier samples is still superior. This is likely to be an indicator that a larger (and more diverse) set of trees can potentially improve MBR decoding. In practice, we use *Thinned* samples.

2. DROPOUT, SGLD, AND MINI-BATCH We would next like to better understand the interdependence between SGLD and two existing sources of noise: dropout and mini-batch size. Results are reported in Figure 2b. In all four stacked bars, the top segment demonstrates the gain when including +SGLD. The right-most stacked bar illustrates the performance of **MULTI TASK (+ADV)** without (bottom part of the bar) and with +SGLD. This is our reference and in the following three experiments, we make a single change to this model while keeping everything else fixed.

In the left-most stacked bar, we exclude dropout. This significantly hurts the performance, an indication that the random perturbation of model by dropout can provide a positive complimentary effect for better exploration of the posterior modes via SGLD. Comparing the top parts of all stacked bars (representing the gain with +SGLD) reveals that this gain is at its maximum when dropout is switched off. An interpretation can be that in the absence of dropout, SGLD becomes the main component for countering overfitting.

In the second stacked bar from the right, we remove the noise caused by mini-batching and use the entire dataset as one batch. This slightly improves the results, although this improvement is not statistically significant. Based on eqn. 2, full training data gradient updates cause $\nabla_\theta \mathcal{L} \to 0$ in fewer steps. Consequently, the learning rate (which controls the variance of the noise) is at a higher value when the gradient is close to zero, giving a better chance for the sampler to mix via ξ_t.

In the second stacked bar from left, parameters are updated for each training sentence. In this setting the learning rate, η_t, is consumed much faster and can potentially reach zero on the MAP solution, discarding the effect of the injected noise and +SGLD. Hence, the gain for including SGLD is at its minimum. We speculate that adjusting the decaying speed of the learning rate (which we left untouched) could allow for this extreme case to perform better.

6 Conclusion

We proposed a Bayesian framework for neural dependency parsing (BNNP). We employ efficient SG-MCMC approximate inference mechanisms through stochastic gradient Langevin dynamics to generate posterior samples during optimization. The collected samples are then used via a minimum Bayes risk parsing algorithm to generate the final parse tree. In mono-lingual and cross-lingual experiments in the small data regime, where Bayesian learning in expected to be most effective, our BNNP consistently outperformed the strong BiLSTM baselines. Moreover, when integrating the BNNP into a multi-task learning framework, utilized to prevent task interference, we outperformed the BiAFFINE parser (best system of the CoNLL17 shared task) on 5 low-resource languages by up to 9% LAS.

In future work, we intend to investigate other types of priors over the network parameters (e.g., sparse priors (Lobacheva et al., 2017)). We would also like to explicitly quantify the uncertainty captured in our framework under different sampling strategies or MCMC-SG methods (e.g., similar to McClure and Kriegeskorte (2016); Teye et al. (2018)).

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909). The first author would like to thank Costanza Conforti, Victor Prokhorov, and Gamal Crichton for their comments on the presentation of this work. The authors would like to thank the anonymous reviewers for their helpful suggestions.

References

Zeljko Agic, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *TACL*.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of ACL*.

Rich Caruana. 1997. Multitask learning. *Machine learning*.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of CoNLL*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of ACL*.

Caio Corro and Ivan Titov. 2018. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *arXiv preprint arXiv:1807.09875*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of CoNLL*.

Roten Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of ACL*.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. 2017. Scalable bayesian learning of recurrent neural networks for language modeling. In *Proceedings of ACL*.

Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of ICML*.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. 2018. Meta-learning for stochastic gradient MCMC. *CoRR*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL*.

Christian Hadiwinoto and Hwee Tou Ng. 2017. A dependency-based neural reordering model for statistical machine translation. In *Proceedings of AAAI*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of EMNLP*.

José Miguel Hernández-Lobato and Ryan P. Adams. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of ICML*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

Katherine A. Keith, Su Lin Blodgett, and Brendan T. O'Connor. 2018. Monte carlo syntax marginals for exploring and using dependency parses. In *Proceedings of NAACL*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of EMNLP*.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of NAACL*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. 2016. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of AAAI*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A. Smith. 2018. Parsing tweets into universal dependencies. In *Proceedings of NAACL*.

Ekaterina Lobacheva, Nadezhda Chirkova, and Dmitry P. Vetrov. 2017. Bayesian sparsification of recurrent neural networks. *CoRR*.

Rivka Malca and Roi Reichart. 2018. Neural transition based parsing of web queries: An entity based approach. In *Proceedings of EMNLP*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*.

Patrick McClure and Nikolaus Kriegeskorte. 2016. Representation of uncertainty in deep neural networks through sampling. *CoRR*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Ryan T. McDonald, Koby Crammer, and Fernando C. N. Pereira. 2005b. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Radford M Neal. 2011. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Vlad Niculae, André F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. Sparsemap: Differentiable sparse structured inference. In *Proceedings of ICML*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Barbara Plank and Héctor Martínez Alonso. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of EACL*.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *TACL*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of EMNLP*.

Christian P. Robert and George Casella. 2010. *Monte Carlo Statistical Methods*.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi S. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*.

Ehsan Shareghi, Gholamreza Haffari, Trevor Cohn, and Ann Nicholson. 2015. Structured prediction of sequences and trees using infinite contexts. In *Proceedings of ECML*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of NAACL*.

Yee Whye Teh, Alexandre H. Thiery, and Sebastian J. Vollmer. 2016. Consistency and fluctuations for stochastic gradient langevin dynamics. *JMLR*.

Mattias Teye, Hossein Azizpour, and Kevin Smith. 2018. Bayesian uncertainty estimation for batch normalized deep networks. In *Proceedings of ICML*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of ACL*.

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of ICML*.

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of ICML*.

Poorya ZareMoodi and Gholamreza Haffari. 2018. Neural machine translation for bilingually scarce scenarios: a deep multi-task learning approach. In *Proceedings of NAACL*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, et al. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of CoNLL*.

AutoSeM: Automatic Task Selection and Mixing in Multi-Task Learning

Han Guo **Ramakanth Pasunuru** **Mohit Bansal**
UNC Chapel Hill
`{hanguo, ram, mbansal}@cs.unc.edu`

Abstract

Multi-task learning (MTL) has achieved success over a wide range of problems, where the goal is to improve the performance of a primary task using a set of relevant auxiliary tasks. However, when the usefulness of the auxiliary tasks w.r.t. the primary task is not known a priori, the success of MTL models depends on the correct choice of these auxiliary tasks and also a balanced mixing ratio of these tasks during alternate training. These two problems could be resolved via manual intuition or hyper-parameter tuning over all combinatorial task choices, but this introduces inductive bias or is not scalable when the number of candidate auxiliary tasks is very large. To address these issues, we present AutoSeM, a two-stage MTL pipeline, where the first stage automatically selects the most useful auxiliary tasks via a Beta-Bernoulli multi-armed bandit with Thompson Sampling, and the second stage learns the training mixing ratio of these selected auxiliary tasks via a Gaussian Process based Bayesian optimization framework. We conduct several MTL experiments on the GLUE language understanding tasks, and show that our AutoSeM framework can successfully find relevant auxiliary tasks and automatically learn their mixing ratio, achieving significant performance boosts on several primary tasks. Finally, we present ablations for each stage of AutoSeM and analyze the learned auxiliary task choices.

1 Introduction

Multi-task Learning (MTL) (Caruana, 1997) is an inductive transfer mechanism which leverages information from related tasks to improve the primary model's generalization performance. It achieves this goal by training multiple tasks in parallel while sharing representations, where the training signals from the auxiliary tasks can help improve the performance of the primary task. Multi-task learning has been applied to a wide range of natural language processing problems (Luong et al., 2015; Pasunuru and Bansal, 2017; Hashimoto et al., 2017; Ruder et al., 2017b; Kaiser et al., 2017; McCann et al., 2018). Despite its impressive performance, the design of a multi-task learning system is non-trivial. In the context of improving the primary task's performance using knowledge from other auxiliary tasks (Luong et al., 2015; Pasunuru and Bansal, 2017), two major challenges include selecting the most relevant auxiliary tasks and also learning the balanced mixing ratio for synergized training of these tasks. One can achieve this via manual intuition or hyper-parameter tuning over all combinatorial task choices, but this introduces human inductive bias or is not scalable when the number of candidate auxiliary tasks is considerable. To this end, we present AutoSeM, a two-stage Bayesian optimization pipeline to this problem.

In our AutoSeM framework[1], the first stage addresses automatic task selection from a pool of auxiliary tasks. For this, we use a non-stationary multi-armed bandit controller (MAB) (Bubeck et al., 2012; Raj and Kalyani, 2017) that dynamically alternates among task choices within the training loop, and eventually returns estimates of the utility of each task w.r.t. the primary task. We model the utility of each task as a Beta distribution, whose expected value can be interpreted as the probability of each task making a non-negative contribution to the training performance of the primary task. Further, we model the observations as Bernoulli variables so that the posterior distribution is also Beta-distributed. We use Thompson sampling (Chapelle and Li, 2011; Russo et al., 2018) to trade off exploitation and exploration.

The second stage then takes the auxiliary tasks

---

[1] We make all our code and models publicly available at: `https://github.com/HanGuo97/AutoSeM`

Proceedings of NAACL-HLT 2019, pages 3520–3531
Minneapolis, Minnesota, June 2 - June 7, 2019.

selected in the first stage and automatically learns the training mixing ratio of these tasks, through the framework of Bayesian optimization, by modeling the performance of each mixing ratio as a sample from a Gaussian Process (GP) to sequentially search for the optimal values (Rasmussen, 2004; Snoek et al., 2012). For the covariance function in the GP, we use the Matern kernel which is parameterized by a smoothness hyperparameter so as to control the level of differentiability of the samples from GP. Further, following Hoffman et al. (2011), we use a portfolio of optimistic and improvement-based policies as acquisition functions (Shahriari et al., 2016) for selecting the next sample point from the GP search space.

We conduct several experiments on the GLUE natural language understanding benchmark (Wang et al., 2018), where we choose each of RTE, MRPC, QNLI, CoLA, and SST-2 as the primary task, and treat the rest of the classification tasks from the GLUE benchmark as candidate auxiliary tasks. Results show that our AUTOSEM framework can successfully find useful auxiliary tasks and automatically learn their mixing ratio, achieving significant performance boosts on top of strong baselines for several primary tasks, e.g., 5.2% improvement on QNLI, 4.7% improvement on RTE, and 2.8%/0.8% improvement on MRPC.

We also ablate the usefulness of our two stages of auxiliary task selection and automatic mixing ratio learning. The first ablation removes the task selection stage and instead directly performs the second GP mixing ratio learning stage on all auxiliary tasks. The second ablation performs the task selection stage (with multi-armed bandit) but replaces the second stage Gaussian Process with manual tuning on the selected tasks. Our 2-stage model performs better than both these ablations, showing that both of our stages are crucial. Further, we also discuss the learned auxiliary task choices in terms of their intuitive relevance w.r.t. the corresponding primary task.

2 Related Work

Multi-task learning (Caruana, 1998), known for improving the generalization performance of a task with auxiliary tasks, has successfully been applied to many domains of machine learning, including natural language processing (Collobert and Weston, 2008; Girshick, 2015; Luong et al., 2015; Pasunuru and Bansal, 2017; Pasunuru et al., 2017), computer vision (Misra et al., 2016; Kendall et al., 2017; Dai et al., 2016), and reinforcement learning (Teh et al., 2017; Parisotto et al., 2015; Jaderberg et al., 2016). Although there are many variants of multi-task learning (Ruder et al., 2017b; Hashimoto et al., 2017; Luong et al., 2015; McCann et al., 2018), our goal is to improve the performance of a primary task using a set of relevant auxiliary tasks, where different tasks share some common model parameters with alternating mini-batches optimization, similar to Luong et al. (2015).

To address the problem of automatic shared parameter selection, Ruder et al. (2017a) automatically learned the latent multi-task sharing architecture, and Xiao et al. (2018) used a gate mechanism that filters the feature flows between tasks. On the problem of identifying task relatedness, Ben-David and Schuller (2003) provided a formal framework for task relatedness and derived generalization error bounds for learning of multiple tasks. Bingel and Søgaard (2017) explored task relatedness via exhaustively experimenting with all possible two task tuples in a non-automated multi-task setup. Other related works explored data selection, where the goal is to select or reorder the examples from one or more domains (usually in a single task) to either improve the training efficiency or enable better transfer learning. These approaches have been applied in machine translation (van der Wees et al., 2017), language models (Moore and Lewis, 2010; Duh et al., 2013), dependency parsing (Søgaard, 2011), etc. In particular, Ruder and Plank (2017) used Bayesian optimization to select relevant training instances for transfer learning, and Tsvetkov et al. (2016) applied it to learn a curriculum for training word embeddings via reordering data. Graves et al. (2017) used the bandit approach (Exp3.S algorithm) in the context of automated curriculum learning, but in our work, we have two stages with each stage addressing a different problem (automatic task selection and learning of the training mixing ratio). Recently, Sharma and Ravindran (2017) used multi-armed bandits (MAB) to learn the choice of hard vs. easy domain data selection as input feed for the model. Guo et al. (2018) used MAB to effectively switch across tasks in a dynamic multi-task learning setup. In our work, we use MAB with Thompson Sampling for the novel paradigm of automatic auxiliary task selection;

and next, we use a Matern-kernel Gaussian Process to automatically learn an exact (static) mixing ratio (i.e., relatedness ratio) for the small number of selected tasks.

Many control problems can be cast as a multi-armed bandits problem, where the goal of the agent is to select the arm/action from one of the N choices that minimizes the regrets (Bubeck et al., 2012). One problem in bandits learning is the trade-off between exploration and exploitation, where the agent needs to make a decision between taking the action that yields the best payoff on current estimates or exploring new actions whose payoffs are not yet certain. Many previous works have explored various exploration and exploitation strategies to minimize regret, including Boltzmann exploration (Kaelbling et al., 1996), adversarial bandits (Auer et al., 2002b), UCB (Auer et al., 2002a), and information gain using variational approaches (Houthooft et al., 2016). In this work, for task selection, we use Thompson Sampling (Russo et al., 2018; Chapelle and Li, 2011), an algorithm for sequential decision making problems, which addresses a broad range of problems in a computationally efficient manner and is therefore enjoying wide use.

Gaussian Process (GP) is a non-parametric Bayesian approach, and it can capture a wide variety of underlying functions or relations between inputs and outputs by taking advantage of the full information provided by the history of observations and is thus very data-efficient (Rasmussen, 2004; Shahriari et al., 2016; Schulz et al., 2018). Gaussian Processes have been widely used as a black-box optimizer and hyper-parameter optimization (Snoek et al., 2012; Brochu et al., 2010; Knudde et al., 2017; Cully et al., 2018; Swersky et al., 2013; Golovin et al., 2017). In our work, we use Gaussian Process for automatic learning of the multi-task mixing ratio in our stage-2 among the selected tasks from stage-1.

3 Models

We will first introduce our baseline model and its integration for multiple classification tasks in a multi-task learning (MTL) setup. Next, we will introduce our AutoSem framework, an automatic way of selecting auxiliary tasks and learning their optimal training mixing ratio w.r.t. the primary task, via a Beta-Bernoulli bandit with Thompson Sampling and a Gaussian Process framework.

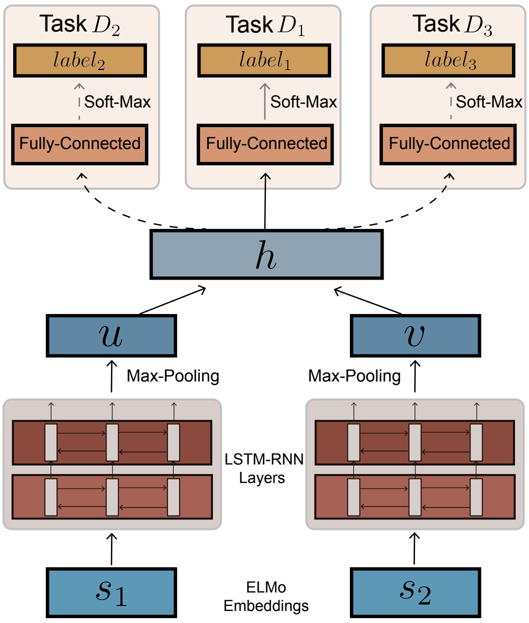


Figure 1: Overview of our baseline model where we use different projection layers for each task during MTL, while sharing rest of the model parameters.

3.1 Bi-Text Classification Model

Let s_1 and s_2 be the input sentence pair in our classification task, where we encode these sentences via bidirectional LSTM-RNN, similar to that of Conneau et al. (2017). Next, we do max-pooling on the output hidden states of both encoders where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the outputs from the max-pooing layer for s_1 and s_2 respectively. Later, we map these two representations ($\boldsymbol{u}$ and $\boldsymbol{v}$) into a single rich dense representation vector $\boldsymbol{h}$:

$$\boldsymbol{h} = [\boldsymbol{u}; \boldsymbol{v}; \boldsymbol{u} \star \boldsymbol{v}; |\boldsymbol{u} - \boldsymbol{v}|] \qquad (1)$$

where $[;]$ represents the concatenation and $\boldsymbol{u} \star \boldsymbol{v}$ represents the element-wise multiplication of $\boldsymbol{u}$ and $\boldsymbol{v}$. We project this final representation $\boldsymbol{h}$ to label space to classify the given sentence pair (see Fig. 1). We also use ELMo (Peters et al., 2018) representations for word embeddings in our model. For this, we extract the three ELMo layer representations for each of the sentence pair and use their weighted sum as the ELMo output representation, where the weights are trainable.

3.2 Multi-Task Learning

In this work, we focus on improving a task (primary task) by allowing it to share parameters with related auxiliary tasks via multi-task learning (MTL). Let $\{D_1, ..., D_N\}$ be a set of N tasks, where we set D_1 to be the primary task and the rest of them as auxiliary tasks. We can extend

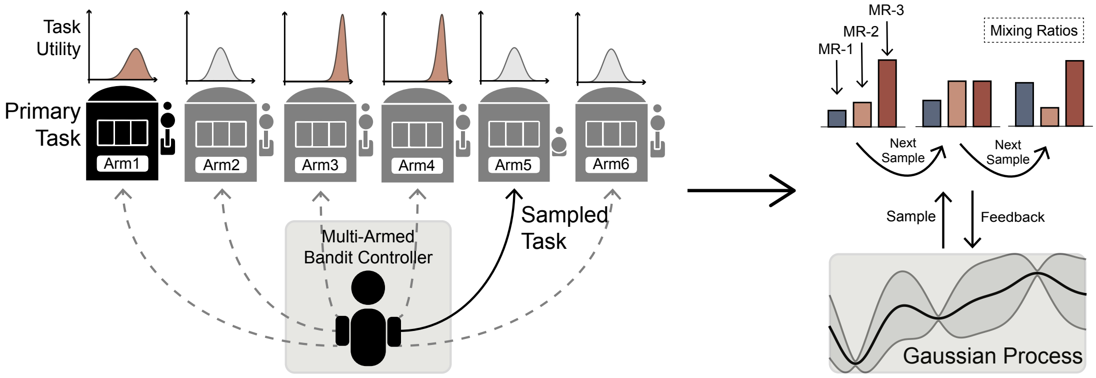


Figure 2: Overview of our AUTOSEM framework. **Left**: the multi-armed bandit controller used for task selection, where each arm represents a candidate auxiliary task. The agent iteratively pulls an arm, observes a reward, updates its estimates of the arm parameters, and samples the next arm. **Right**: the Gaussian Process controller used for automatic mixing ratio (MR) learning. The GP controller sequentially makes a choice of mixing ratio, observes a reward, updates its estimates, and selects the next mixing ratio to try, based on the full history of past observations.

our single-task learning baseline (see Sec. 3.1) into multi-task learning model by augmenting the model with N projection layers while sharing the rest of the model parameters across these N tasks (see Fig. 1). We employ MTL training of these tasks in alternate mini-batches based on a mixing ratio η_1:η_2:..η_N, similar to previous work (Luong et al., 2015), where we optimize η_i mini-batches of task i and go to the next task.

In MTL, choosing the appropriate auxiliary tasks and properly tuning the mixing ratio can be important for the performance of multi-task models. The naive way of trying all combinations of task selections is hardly tractable. To solve this issue, we propose AUTOSEM, a two-stage pipeline in the next section. In the first stage, we automatically find the relevant auxiliary tasks (out of the given $N-1$ options) which improve the performance of the primary task. After finding the relevant auxiliary tasks, in the second stage, we take these selected tasks along with the primary task and automatically learn their training mixing ratio.

3.3 Automatic Task Selection: Multi-Armed Bandit with Thompson Sampling

Tuning the mixing ratio for N tasks in MTL becomes exponentially harder as the number of auxiliary tasks grows very large. However, in most circumstances, only a small number of these auxiliary tasks are useful for improving the primary task at hand. Manually searching for this optimal choice of relevant tasks is intractable. Hence, in this work, we present a method for automatic task selection via multi-armed bandits with Thompson Sampling (see the left side of Fig. 2).

Let $\{a_1, ..., a_N\}$ represent the set of N arms (corresponding to the set of tasks $\{D_1, ..., D_N\}$) of the bandit controller in our multi-task setting, where the controller selects a sequence of actions/arms over the current training trajectory to maximize the expected future payoff. At each round t_b, the controller selects an arm based on the noisy value estimates and observes rewards r_{t_b} for the selected arm. Let $\theta_k \in [0, 1]$ be the utility (usefulness) of task k. Initially, the agent begins with an independent prior belief over θ_k. We take these priors to be Beta-distributed with parameters α_k and β_k, and the prior probability density function of θ_k is:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1}(1 - \theta_k)^{\beta_k - 1} \quad (2)$$

where Γ denotes the gamma function. We formulate the reward $r_{t_b} \in \{0, 1\}$ at round t_b as a Bernoulli variable, where an action k produces a reward of 1 with a chance of θ_k and a reward of 0 with a chance of $1 - \theta_k$. The true utility of task k, i.e., θ_k, is unknown, and may or may not change over time (based on stationary vs. non-stationary of task utility). We define the reward as whether sampling the task k improves (or maintains) the validation metric of the primary task,

$$r_{t_b} = \begin{cases} 1, & \text{if } R_{t_b} \geq R_{t_b - 1} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where R_{t_b} represents the validation performance of the primary task at time t_b. With our reward setup above, the utility of each task (θ_k) can be intuitively interpreted as the probability that multi-task learning with task k can improve (or maintain) the performance of the primary task. The conjugacy properties of the Beta distribution assert that the posterior distribution is also Beta with parameters that can be updated using a simple Bayes rule, which is defined as follows (Russo et al., 2018),

$$\begin{aligned} p(\theta_k|r) &\propto \text{Bern}_\theta(r)\text{Beta}_{\alpha,\beta}(\theta_k) \\ &\propto \text{Beta}_{\alpha+r,\beta+1-r}(\theta_k) \end{aligned} \tag{4}$$

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_k, \beta_k), & \text{if } x^s_{t_b} \neq k \\ (\alpha_k, \beta_k)+(r_{t_b}, 1 - r_{t_b}), & \text{if } x^s_{t_b} = k \end{cases} \tag{5}$$

where $x^s_{t_b}$ is the sampled task at round t_b. Finally, at the end of the training, we calculate the expected value of each arm as follows:

$$\mathbb{E}_p[\theta_k] = \frac{\alpha_k}{\alpha_k + \beta_k} \tag{6}$$

Here, the expectation measures the probability of improving (or maintaining) the primary task by sampling this task. To decide the next action to take, we apply Thompson Sampling (Russo et al., 2018; Chapelle and Li, 2011) to trade off exploitation (maximizing immediate performance) and exploration (investing to accumulate new information that might improve performance in the future). In Thompson Sampling (Russo et al., 2018), instead of taking action k that maximizes the expectation (i.e., $\arg\max_k \mathbb{E}_p[\theta_k]$), we randomly sample the primary task improvement probability $\hat{\theta}_k$ from the posterior distribution $\hat{\theta}_k \sim p(\theta_k)$, and take the action k that maximizes the sampled primary task improvement probability, i.e., $\arg\max_k \hat{\theta}_k$. At the end of the training, the task selection can proceed either via a threshold on the expectation, or take the top-K tasks, and run stage-2 using the selected task subset as auxiliary tasks (details in Sec. 3.4).

Stronger Prior for Primary Task Note that at the beginning of training, model performance is usually guaranteed to improve from the initial random choices. This causes issues in updating arm values because less useful tasks will be given high arm values when they happen to be sampled at the beginning. To resolve this issue, we initially set a slightly stronger prior/arm-value in favor of the arm corresponding to the primary task. Intuitively, the bandit will then sample the primary model more often at the beginning, and then start exploring auxiliary tasks when the primary model's performance stabilizes (as the arm value of the primary model will start decreasing because sampling it in later rounds produces smaller additional improvements).

Algorithm 1 BernThompson$(N, \alpha, \beta, \gamma, \alpha_0, \beta_0)$

```
for $t_b = 1, 2, \ldots$ do
    # sample model:
    for $k = 1, \ldots, N$ do
        Sample $\hat{\theta}_k \sim \text{Beta}(\alpha_k, \beta_k)$
    end for
    # select and apply action:
    $x^s_{t_b} \leftarrow \arg\max_k \hat{\theta}_k$
    Apply $x^s_{t_b}$ and observe $r_{t_b}$
    # non-stationarity
    for $k = 1, \ldots, N$ do
        $\hat{\alpha}_k = (1 - \gamma)\alpha_k + \gamma\alpha_0$
        $\hat{\beta}_k = (1 - \gamma)\beta_k + \gamma\beta_0$
        if $k \neq x^s_{t_b}$ then
            $(\alpha_k, \beta_k) \leftarrow (\hat{\alpha}_k, \hat{\beta}_k)$
        else
            $(\alpha_k, \beta_k) \leftarrow (\hat{\alpha}_k, \hat{\beta}_k)+(r_{t_b}, 1 - r_{t_b})$
        end if
    end for
end for
```

Non-Stationary Multi-Armed Bandit Also note that the intrinsic usefulness of each task varies throughout the training (e.g., the primary task might be more important at the beginning, but not necessarily at the end), and thus the agent faces a non-stationary system. In such cases, the agent should always be encouraged to explore in order to track changes as the system drifts. One simple approach to inject non-stationarity is to discount the relevance of previous observations. Thus we introduce a tunable decay ratio γ, and modify Eq. 3.3 as follows:

$$(\alpha_k, \beta_k) = \begin{cases} (\hat{\alpha}_k, \hat{\beta}_k), & \text{if } k \neq x^s_{t_b} \\ (\hat{\alpha}_k, \hat{\beta}_k)+(r_{t_b}, 1 - r_{t_b}), & \text{if } k = x^s_{t_b} \end{cases} \tag{7}$$

where $\hat{\alpha}_k = (1-\gamma)\alpha_k+\gamma\alpha_0$ and $\hat{\beta}_k = (1-\gamma)\beta_k+\gamma\beta_0$, and γ controls how quickly uncertainty is injected into the system (α_0, β_0 are parameters of the prior). Algorithm 1 presents the Thompson Sampling algorithm with a Beta-Bernoulli MAB.

3.4 Automatic Mixing Ratio Learning via Gaussian Process

The right side of Fig. 2 illustrates our Gaussian Process controller for automatic learning of the MTL training mixing ratio (see definition in Sec. 3.2). Given the selected auxiliary tasks from the previous section, the next step is to find a proper mixing ratio of training these selected tasks along with the primary task.[2] Manual tuning of this mixing ratio via a large grid search over the hyperparameter values is very time and compute expensive (even when the number of selected auxiliary tasks is small, e.g., 2 or 3). Thus, in our second stage, we instead apply a non-parametric Bayesian approach to search for the approximately-optimal mixing ratio. In particular, we use a 'Gaussian Process' to sequentially search for the mixing ratio by trading off exploitation and exploration automatically. Next, we describe our Gaussian Process approach in detail.

A Gaussian Process (Rasmussen, 2004; Snoek et al., 2012; Shahriari et al., 2016), GP(μ_0, k), is a non-parametric model that is fully characterized by a mean function $\mu_0 : \mathcal{X} \mapsto \mathbb{R}$ and a positive-definite kernel or covariance function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ denote any finite collections of n points, where each $\boldsymbol{x}_i$ represents a choice of the mixing ratio (i.e., the ratio η_1:η_2:..η_N described in Sec. 3.2), and $f_i = f(\boldsymbol{x}_i)$ is the (unknown) function values evaluated at $\boldsymbol{x}_i$ (true performance of the model given the selected mixing ratio). Let $y_1, y_2, ..., y_n$ be the corresponding noisy observations (the validation performance at the end of training). In the context of GP Regression (GPR), $\boldsymbol{f} = \{f_1, ..., f_n\}$ are assumed to be jointly Gaussian (Rasmussen, 2004), i.e., $\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K})$, where, $\boldsymbol{m}_i = \mu_0(\boldsymbol{x}_i)$ is the mean vector, and $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the covariance matrix. Then the noisy observations $\boldsymbol{y} = y_1, ..., y_n$ are normally distributed around $\boldsymbol{f}$ as follows: $\boldsymbol{y}|\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I})$.

Given $\mathcal{D} = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{n_0}, y_{n_0})$, the set of random initial observations, where $\boldsymbol{x}_i$ represents a mixing ratio and y_i represents the corresponding model's validation performance. Next, we model the GP based on these initial observations as described above. We sample a next point $\boldsymbol{x}_{n_0+1}$ (a mixing ratio in our case) from this GP and get its corresponding model performance y_{n_0+1}, and update the GP again by now considering the $n_0 + 1$ points (Rasmussen, 2004). We continue this process for a fixed number of steps. Next, we will discuss how we perform the sampling (based on acquisition functions) and the kernels used for calculating the covariance.

Acquisition Functions Here, we describe the acquisition functions for deciding where to sample next. While one could select the points that maximize the mean function, this does not always lead to the best outcome (Hoffman et al., 2011). Since we also have the variance of the estimates along with the mean value of each point $\boldsymbol{x}_i$, we can incorporate this information into the optimization. In this work, we use the GP-Hedge approach (Hoffman et al., 2011; Auer et al., 1995), which probabilistically chooses one of three acquisition functions: probability of improvement, expected improvement, and upper confidence bound. Probability of improvement acquisition functions measure the probability that the sampled mixing ratio $\boldsymbol{x}_i$ leads to an improvement upon the best observed value so far (τ), $\mathbb{P}(f(\boldsymbol{x}_i) > \tau)$. Expected improvement additionally incorporates the amount of improvement, $\mathbb{E}[(f(\boldsymbol{x}_i) - \tau)\mathbb{I}(f(\boldsymbol{x}_i) > \tau)]$. The Gaussian Process upper confidence bound (GP-UCB) algorithm measures the optimistic performance upper bound of the sampled mixing ratio (Srinivas et al., 2009), $\mu_i(\boldsymbol{x}_i) + \lambda\sigma_i(\boldsymbol{x}_i)$, for some hyper-parameter λ.

Matern Kernel The covariance function (or kernel) defines the nearness or similarity of two points in the Gaussian Process. Here, we use the automatic relevance determination (ARD) Matern kernel (Rasmussen, 2004), which is parameterized by $\nu > 0$ that controls the level of smoothness. In particular, samples from a GP with such a kernel are differentiable $\lfloor \nu - 1 \rfloor$ times. When ν is half-integer (i.e. $\nu = p + 1/2$ for non-negative integer p), the covariance function is a product of an exponential and a polynomial of order p. In the context of machine learning, usual choices of ν include $3/2$ and $5/2$ (Shahriari et al., 2016).

4 Experiment Setup

Datasets: We evaluate our models on several datasets from the GLUE benchmark (Wang et al., 2018): RTE, QNLI, MRPC, SST-2, and CoLA. For all these datasets, we use the standard splits provided by Wang et al. (2018). For dataset details, we refer the reader to the GLUE paper.[3]

---

[2] Note that ideally Gaussian Process can also learn to set the mixing ratio of less important tasks to zero, hence allowing it to essentially also perform the task selection step. However, in practice, first applying our task selection Thompson-Sampling model (Sec. 3.3) allows GP to more efficiently search the mixing ratio space for the small number of filtered auxiliary tasks, as shown in results of Sec. 6.1.

[3] We did not include the remaining tasks as primary tasks, because STS-B is a regression task; MNLI is a very large dataset and does not benefit much from MTL with other tasks in the GLUE benchmark; and QQP and WNLI have dev/test discrepancies and adversarial label issues as per the GLUE

Models	RTE	MRPC	QNLI	CoLA	SST-2
BiLSTM+ELMo (Single-Task) (Wang et al., 2018)	50.1	69.0/80.8	69.4	**35.0**	90.2
BiLSTM+ELMo (Multi-Task) (Wang et al., 2018)	55.7	76.2/83.5	66.7	27.5	89.6
Our Baseline	54.0	75.7/83.7	74.0	30.8	91.3
Our AutoSeM	**58.7**	**78.5/84.5**	**79.2**	32.9	**91.8**

Table 1: Test GLUE results of previous work, our baseline, and our AutoSeM MTL framework. We report accuracy and F1 for MRPC, Matthews correlation for CoLA, and accuracy for all others.

Training Details: We use pre-trained ELMo[4] to obtain sentence representations as inputs to our model (Peters et al., 2018), and the Gaussian Process implementation is based on Scikit-Optimize[5], and we adopt most of the default configurations. We use accuracy as the validation criterion for all tasks. For all of our experiments except QNLI and SST-2, we apply early stopping on the validation performance plateau.[6] The set of candidate auxiliary tasks consists of all 2-sentence classification tasks when the primary task is a classification of two sentences, whereas it consists of all two-sentence and single-sentence classification tasks when the primary task is a classification of a single sentence.[7] Since the utility estimates from the multi-armed bandit controller are noisy, we choose the top two tasks based on expected task utility estimates, and include additional tasks if their utility estimate is above 0.5. All the results reported are the aggregate of the same experiment with two runs (with different random seeds) unless explicitly mentioned.[8] We use a two-layer LSTM-RNN with hidden size of 1024 for RTE and 512 for the rest of the models, and use Adam Optimizer (Kingma and Ba, 2014). The prior parameters of each task in stage-1 are set to be $\alpha_0 = 1$, $\beta_0 = 1$, which are commonly used in other literature. For stage-1, the bandit controller iteratively selects batches of data from different tasks during training to learn the approximate importance of each auxiliary task (Graves et al., 2017). In stage-2 (Gaussian Process), we sequentially draw samples of mixing ratios and evaluate each sample after full training (Snoek et al., 2012). Without much tuning, we used approximately 200 rounds for the stage-1 bandit-based approach, where each round consist of approximately 10 mini-batches of optimization. For stage-2, we experimented with 15 and 20 as the number of samples to draw and found that 15 samples for MRPC and 20 samples for the rest of the tasks work well. This brings the total computational cost for our two-stage pipeline to be approximately (15+1)x and (20+1)x, where x represents the time taken to run the baseline model for the given task. This is significantly more efficient than a grid-search based manually-tuned mixing ratio setup (which would scale exponentially with the number of tasks).

5 Results

5.1 Baseline Models

Table 1 shows the results of our baseline and previous works (Wang et al., 2018). We can see that our single-task baseline models achieve stronger performance on almost all tasks in comparison to previous work's single-task models.[9] Next, we present the performance of our AutoSeM framework on top of these strong baselines.

5.2 Multi-Task Models

Table 1 also presents the performance of our AutoSeM framework-based MTL models. As can be seen, our MTL models improve significantly (see Table 3 for standard deviations) upon their corresponding single-task baselines for all tasks, and achieve strong improvements as compared to the fairly-comparable[9] multi-task results of previous work (Wang et al., 2018).[10] During the task

---

website's FAQ: `https://gluebenchmark.com/faq`

[4] `https://allennlp.org/elmo`

[5] `https://scikit-optimize.github.io`

[6] In our initial experiments, we found early stopping on larger datasets led to sub-optimal performance, and hence we used a pre-specified maximum number of steps instead.

[7] We made this design decision because there are only two single-sentence tasks in GLUE, so we mix them with 2-sentence tasks to allow more auxiliary choices.

[8] We use the average of validation results across runs as the tuning criterion, and use the ensemble of models across runs for reporting the test results.

[9] Note that we do not report previous works which *fine-tune* large external language models for the task (e.g., OpenAI-GPT and BERT), because they are not fairly comparable w.r.t. our models. Similarly, we report the non-attention based best GLUE models (i.e., BiLSTM+ELMo) for a fair comparison to our non-attention baseline. Our approach should ideally scale to large pre-training/fine-tuning models like BERT, given appropriate compute resources.

[10] Note that even though the performance improvement gaps of Wang et al. (2018) (MTL vs. baseline) and our improvements (AutoSeM vs. our improved baseline) are similar, these are inherently two different setups. Wang et al. (2018) MTL is based on a 'one model for all' setup (Kaiser et al., 2017; McCann et al., 2018), whereas our approach in-

selection stage of our AUTOSEM framework, we observe that MultiNLI is chosen as one of the auxiliary tasks in all of our MTL models. This is intuitive given that MultiNLI contains multiple genres covering diverse aspects of the complexity of language (Conneau et al., 2017). Also, we observe that WNLI is sometimes chosen in the task selection stage; however, it is always dropped (mixing ratio of zero) by the Gaussian Process controller, showing that it is not beneficial to use WNLI as an auxiliary task (intuitive, given its small size). Next, we discuss the improvements on each of the primary tasks and the corresponding auxiliary tasks selected by AUTOSEM framework.

RTE: Our AUTOSEM approach achieves stronger results w.r.t. the baseline on RTE (58.7 vs. 54.0). During our task selection stage, we found out that QQP and MultiNLI tasks are important for RTE as auxiliary tasks. For the second stage of automatic mixing ratio learning via Gaussian Process, the model learns that a mixing ratio of 1:5:5 works best to improve the primary task (RTE) using related auxiliary tasks of QQP and MultiNLI.

MRPC: AUTOSEM here performs much better than the baseline on MRPC (78.5/84.5 vs. 75.7/83.7). During our task selection stage, we found out that RTE and MultiNLI tasks are important for MRPC as auxiliary tasks. In the second stage, AUTOSEM learned a mixing ratio of 9:1:4 for these three tasks (MRPC:RTE:MultiNLI).

QNLI: Again, we achieve substantial improvements with AUTOSEM w.r.t. baseline on QNLI (79.2 vs. 74.0). Our task selection stage learned that WNLI and MultiNLI tasks are best as auxiliary tasks for QNLI. We found that the Gaussian Process further drops WNLI by setting its mixing ratio to zero, and returns 20:0:5 as the best mixing ratio for QNLI:WNLI:MultiNLI.

CoLA: We also observe a strong performance improvement on CoLA with our AUTOSEM model w.r.t. our baseline (32.9 vs. 30.8). During our task selection stage, we found out that MultiNLI and WNLI tasks are important for CoLA as auxiliary tasks. In the second stage, GP learns to drop WNLI, and found the mixing ratio of 20:5:0 for CoLA:MultiNLI:WNLI.

SST-2: Here also our AUTOSEM approach performs better than the baseline (91.8 vs. 91.3). The task selection stage chooses MultiNLI, MRPC, and WNLI as auxiliary tasks and the stage-2 Gaussian Process model drops MRPC and WNLI by setting their mixing ratio to zero (learns ratio of 13:5:0:0 for SST-2:MultiNLI:MRPC:WNLI).

Name	Validation	Test
Baseline	78.3	75.7/83.7
w/o Stage-1	80.3	76.3/83.8
w/o Stage-2	80.3	76.7/83.8
Final MTL	81.2	78.5/84.5

Table 2: Ablation results on the two stages of our AUTOSEM framework on MRPC.

6 Analysis

6.1 Ablation on MTL stages

In this section, we examine the usefulness of each stage of our two-stage MTL pipeline.[11]

Removing Stage-1: The purpose of the Beta-Bernoulli MAB in stage-1 is to find useful auxiliary tasks for the given primary task. Here, to understand its importance, we remove the task selection part, and instead directly run the Gaussian Process (GP) model on all tasks (see 'w/o Stage-1' row in Table 2). We can see that by removing the task selection stage, the Gaussian Process model can still outperform the baseline, indicating the usefulness of the GP, but the large mixing ratio search space causes the GP to be unable to efficiently find the best mixing ratio setting.

Removing Stage-2: Given the selected tasks from stage-1, the goal of the Gaussian Process in stage-2 is to efficiently find the approximately-optimal mixing ratio. To examine its usefulness, we replace the Gaussian Process controller by manually tuning a grid of mixing ratios, where the number of tuning experiments equals to the number of steps used in the Gaussian Process model (for a fair comparison). Table 2 shows the results by removing stage-2. We can see that a grid search over hyper-parameters can improve upon the baseline, indicating the usefulness of stage-1 task selection, but a reasonable-sized fair-comparison grid search (i.e., not exhaustive over all ratio values) is not able to match our stage-2 GP process that leverages prior experimental results to more efficiently find the best setting.

---

terpretably chooses the 2-3 tasks that are most beneficial for the given primary task. Also see Sec. 4 for comparison of training speeds for these two setups.

[11] We present this ablation only on MRPC for now, because GP stage-2 takes a lot of time without the task selection stage.

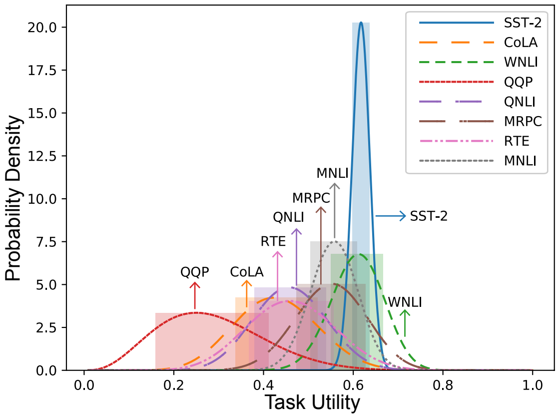


Figure 3: Visualization of task utility estimates from the multi-armed bandit controller on SST-2 (primary task). The x-axis represents the task utility, and the y-axis represents the corresponding probability density. Each curve corresponds to a task and the bar corresponds to their confidence interval.

6.2 Stability of MTL Models

In this section, we provide the mean and standard deviation of our baseline and multi-task models (over three runs) on the validation set. Note that the test set is hidden, so we cannot do these studies on it. As seen in Table 3, our multi-task models clearly surpass the performance of baseline models w.r.t. standard deviation gaps, in all tasks.

6.3 Visualization of Task Selection

In Fig. 3, we show an example of the task utility estimates from the stage-1 multi-armed bandit controller (Eq. 3.3) on SST-2. The x-axis represents the task utility, and the y-axis represents the probability density over task utility. Each curve represents a task (the blue curve corresponds to the primary task, SST-2, and the rest of the curves correspond to auxiliary tasks), and the width of the bars represents the confidence interval of their estimates. We can see that the bandit controller gives the highest (and most confident) utility estimate for the primary task, which is intuitive given that the primary task should be the most useful task for learning itself. Further, it gives 2-3 tasks moderate utility estimates (the corresponding expected values are around 0.5), and relatively lower utility estimates for the remaining tasks (the corresponding expected values are lower than 0.5).

6.4 Educated-Guess Baselines

Name	RTE	MRPC	QNLI	CoLA	SST-2
BASELINES					
Mean Std	58.6 0.94	78.3 0.31	74.9 0.30	74.6 0.44	91.4 0.36
MULTI-TASK MODELS					
Mean Std	62.0 0.62	81.1 0.20	76.0 0.18	75.7 0.18	91.8 0.29

Table 3: Validation-set performance mean and standard deviation (based on three runs) of our baselines and Multi-task models in accuracy.

We additionally experimented with 'educated-guess' baseline models, where MTL is performed using manual intuition mixtures that seem a priori sensible.[12] For example, with MRPC as the primary task, our first educated-guess baseline is to choose other similar paraphrasing-based auxiliary tasks, i.e., QQP in case of GLUE. This MRPC+QQP model achieves 80.8, whereas our AUTOSEM framework chose MRPC+RTE+MultiNLI and achieved 81.2. Furthermore, as our second educated-guess baseline, we added MultiNLI as an auxiliary task (in addition to QQP), since MultiNLI was helpful for all tasks in our MTL experiments. This educated-guess MRPC+QQP+MultiNLI model achieves 80.9 (vs. 81.2 for our AUTOSEM model). This suggests that our AUTOSEM framework (that automatically chose the seemingly less-related RTE task for MRPC) is equal or better than manual intuition based educated-guess models.

7 Conclusion

We presented the AUTOSEM framework, a two-stage multi-task learning pipeline, where the first stage automatically selects the relevant auxiliary tasks for the given primary task and the second stage automatically learns their optimal mixing ratio. We showed that AUTOSEM performs better than strong baselines on several GLUE tasks. Further, we ablated the importance of each stage of our AUTOSEM framework and also discussed the intuition of selected auxiliary tasks.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), ONR (N00014-18-1-2871), Google, Facebook, Baidu, Salesforce, and Nvidia. The views contained in this article are those of the authors and not of the funding agency.

---

[12] These educated-guess models replace our stage-1 automatic auxiliary task section with manual intuition task-mixtures; but we still use our stage-2 Gaussian Process for mixing ratio learning, for fair comparison.

References

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *focs*, page 322. IEEE.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002b. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Shai Ben-David and Reba Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*.

Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

A. Cully, K. Chatzilygeroudis, F. Allocati, and J.-B. Mouret. 2018. Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization. *The Journal of Open Source Software*, 3(26):545.

Jifeng Dai, Kaiming He, and Jian Sun. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 678–683.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. 2017. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. *arXiv preprint arXiv:1806.07304*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Matthew D Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336. Citeseer.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 3.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. 2017. GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv preprint – arXiv:1711.03845*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224. Association for Computational Linguistics.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. *arXiv preprint arXiv:1704.07489*.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *NFiS@EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Vishnu Raj and Sheetal Kalyani. 2017. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*.

Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017a. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017b. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Eric Schulz, Maarten Speekenbrink, and Andreas Krause. 2018. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Sahil Sharma and Balaraman Ravindran. 2017. Online multi-task learning using active sampling. In *ICLR*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. *arXiv preprint arXiv:1605.03852*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. *arXiv preprint arXiv:1708.00712*.

Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated multi-task network for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 726–731.

Studying the Inductive Biases of RNNs with Synthetic Variations of Natural Languages

Shauli Ravfogel[1] **Yoav Goldberg**[1,2] **Tal Linzen**[3]
[1]Computer Science Department, Bar Ilan University
[2]Allen Institute for Artificial Intelligence
[3]Department of Cognitive Science, Johns Hopkins University
{shauli.ravfogel, yoav.goldberg}@gmail.com, tal.linzen@jhu.edu

Abstract

How do typological properties such as word order and morphological case marking affect the ability of neural sequence models to acquire the syntax of a language? Cross-linguistic comparisons of RNNs' syntactic performance (e.g., on subject-verb agreement prediction) are complicated by the fact that any two languages differ in multiple typological properties, as well as by differences in training corpus. We propose a paradigm that addresses these issues: we create synthetic versions of English, which differ from English in one or more typological parameters, and generate corpora for those languages based on a parsed English corpus. We report a series of experiments in which RNNs were trained to predict agreement features for verbs in each of those synthetic languages. Among other findings, (1) performance was higher in subject-verb-object order (as in English) than in subject-object-verb order (as in Japanese), suggesting that RNNs have a recency bias; (2) predicting agreement with both subject and object (polypersonal agreement) improves over predicting each separately, suggesting that underlying syntactic knowledge transfers across the two tasks; and (3) overt morphological case makes agreement prediction significantly easier, regardless of word order.

1 Introduction

The strong performance of recurrent neural networks (RNNs) in applied natural language processing tasks has motivated an array of studies that have investigated their ability to acquire natural language syntax without syntactic annotations; these studies have identified both strengths (Linzen et al., 2016; Giulianelli et al., 2018; Gulordava et al., 2018; Kuncoro et al., 2018; van Schijndel and Linzen, 2018; Wilcox et al., 2018) and limitations (Chowdhury and Zamparelli, 2018; Marvin and Linzen, 2018; Wilcox et al., 2018).

Most of the work so far has focused on English, a language with a specific word order and relatively poor morphology. Do the typological properties of a language affect the ability of RNNs to learn its syntactic regularities? Recent studies suggest that they might. Gulordava et al. (2018) evaluated language models on agreement prediction in English, Russian, Italian and Hebrew, and found worse performance on English than the other languages. In the other direction, a study on agreement prediction in Basque showed substantially *worse* average-case performance than reported for English (Ravfogel et al., 2018).

Existing cross-linguistic comparisons are difficult to interpret, however. Models were inevitably trained on a different corpus for each language. The constructions tested can differ across languages (Gulordava et al., 2018). Perhaps most importantly, any two natural languages differ in a number of typological dimensions, such as morphological richness, word order, or explicit case marking. This paper proposes a controlled experimental paradigm for studying the interaction of the inductive bias of a neural architecture with particular typological properties. Given a parsed corpus for a particular natural language (English, in our experiments), we generate corpora for synthetic languages that differ from the original language in one of more typological parameters (Chomsky, 1981), following Wang and Eisner (2016). In a synthetic version of English with a subject-object-verb order, for example, sentence (1-a) would be transformed into (1-b):

(1) a. The man eats the apples.
b. The man the apples eats.

Proceedings of NAACL-HLT 2019, pages 3532–3542
Minneapolis, Minnesota, June 2 - June 7, 2019.

Original		they say the broker took them out for lunch frequently . *(they, broker: subjects; say, took: verbs; them: object)*
Polypersonal agreement		they saykon the broker tookkarker them out for lunch frequently . *(kon: plural subject; kar: singular subject; ker: plural object)*
Word order variation	SVO	they say the broker took out frequently them for lunch .
	SOV	they the broker them took out frequently for lunch say .
	VOS	say took out frequently them the broker for lunch they.
	VSO	say they took out frequently the broker them for lunch .
	OSV	them the broker took out frequently for lunch they say .
	OVS	them took out frequently the broker for lunch say they *(they, broker: subjects; say, took: verbs; them: object)*
Case systems	Unambiguous	theykon saykon the brokerkar tookkarker theyker out for lunch frequently . *(kon: plural subject; kar: singular subject; ker: plural object)*
	Syncretic	theykon saykon the brokerkar tookkarkar theykar out for lunch frequently . *(kon: plural subject; kar: plural object/singular subject)*
	Argument marking	theyker sayker the brokerkin tookkerkin theyker out for lunch frequently . (*ker: plural argument; kin: singular argument*)

Figure 1: The sentences generated in our synthetic languages based on an original English sentence. All verbs in the experiments reported in the paper carried subject and object agreement suffixes as in the polypersonal agreement experiment; we omitted these suffixes from the word order variation examples in the table for ease of reading.

We then train a model to predict the agreement features of the verb; in the present paper, we focus on predicting the plurality of the subject and the object (that is, whether they are singular or plural). The subject plurality prediction problem for (1-b), for example, can be formulated as follows:

(2) The man the apples ⟨singular/plural subject?⟩.

We illustrate the potential of this approach in a series of case studies. We first experiment with polypersonal agreement, in which the verb agrees with both the subject and the object (§3). We then manipulate the order of the subject, the object and the verb (§4), and experiment with overt morphological case (§5). For a preview of our synthetic languages, see Figure 1.

2 Setup

Synthetic Language Generation We used an expert-annotated corpus, to avoid potential confounds between the typological parameters we manipulated and possible parse errors in an automatically parsed corpus. As our starting point, we took the English Penn Treebank (Marcus et al., 1993), converted to the Universal Dependencies scheme (Nivre et al. 2016) using the Stanford converter (Schuster and Manning, 2016). We then manipulated the tree representations of the sentences in the corpus to generate parametrically modified English corpora, varying in case systems, agreement patterns, and order of core elements. For each parametric version of English, we recorded the verb-argument relations within each sentence, and created a labeled dataset. We exposed our models to sentences from which one of the verbs was omitted, and trained them to predict the plurality of the arguments of the unseen verb. The following paragraph describes the process of collecting verb-argument relations; a detailed discussion of the parametric generation process for agreement marking, word order and case marking is given in the corresponding sections. We have made our synthetic language generation code publicly available.[1]

Argument Collection We created a labeled agreement prediction dataset by first collecting verb-arguments relations from the parsed corpus. We collected nouns, proper nouns, pronouns, adjectives, cardinal numbers and relative pronouns connected to a verb (identified by its part-of-speech tag) with an *nsubj*, *nsubjpass* or *dobj* dependency edge, and record the plurality of those arguments. Verbs that were the head of a clausal complement without a subject (*xcomp* dependencies) were excluded. We recorded the plurality of the dependents of the verb regardless of whether the tense and person of the verb condition agreement in English (that is, not only in third-person

---

[1] https://github.com/Shaul1321/rnn_typology

Prediction task	Subject accuracy	Object accuracy	Object recall
Subject	94.7 ± 0.3	-	-
Object	-	88.9 ± 0.26	81.8 ± 1.4
Joint	95.7 ± 0.23	90.0 ± 0.1	85.4 ± 2.3

Table 1: Results of the polypersonal agreement experiments. "Joint" refers to multitask prediction of subject and object plurality.

	Singular	Plural
Subject	-kar	-kon
Object	-kin	-ker
Indirect Object	-ken	-kre

Table 2: Case suffixes used in the experiments. Verbs are marked by a concatenation of the suffixes of their corresponding arguments.

present-tense verbs). For relative pronouns that function as subjects or objects, we recorded the plurality of their referent; for instance, in the phrase *Treasury bonds, which pay lower interest rates*, we considered the verb *pay* to have a plural subject.

Prediction Task We experimented both with prediction of one of the arguments of the verb (subject or object), and with a joint setting in which the model predicted both arguments of each verb. Consider, for example, the prediction problem (3) (the verb in the original sentence was *gave*):

(3) The state ⟨verb⟩ CenTrust 30 days to sell the Rubens .

In the joint prediction setting the system is expected to make the prediction ⟨subject: singular, object: plural⟩. For each argument, the model predicts one of three categories: SINGULAR, PLURAL or NONE. The NONE label was used in the object prediction task for intransitive verbs, which do not have an object; it was never used in the subject prediction task.

Model We used bidirectional LSTMs with 150 hidden units. The bidirectional LSTM's representation of the left and right contexts of the verb was fed into a multilayer perceptron (MLP) with two hidden layers of sizes 100 and 50. We used independent MLPs to predict subject and object plurality. To capture morphological information, words were represented as the sum of the word embedding and embeddings of the character n-grams that made up the word.[2]

The model (including the embedding layer) was trained end-to-end using the Adam optimizer (Kingma and Ba, 2014). For each of the experiments described in the paper, we trained four models with different random initializations; we report averaged results alongside standard deviations.

3 Polypersonal Agreement

In languages with polypersonal agreement, verbs agree not only with their subject (as in English), but also with their direct object. Consider the following Basque example:[3]

(4) *Kutxazain-ek bezeroa-ri liburu-ak*
cashier-PL.ERG customer-SG.DAT book-PL.ABS
eman dizkiote
gave they-them-to-her/him
The cashiers gave the books to the customer.

Information about the grammatical role of certain constituents in the sentence may disambiguate the function of others; most trivially, if a word is the subject of a given verb, it cannot simultaneously be its object. The goal of the present experiment is to determine whether jointly predicting both object and subject plurality improves the overall performance of the model.

Corpus Creation In sentences with multiple verbs, agreement markers on verbs other than the prediction target could plausibly help predict the features on the target verb. In a preliminary experiment, we did not observe clear differences between different verb marking schemes (e.g., avoiding marking agreement on verbs other than the prediction target). We thus opted for full marking in all experiments: verbs are modified with suffixes that encode the number of all their arguments (see Figure 1). The suffixes we used for verbs are a concatenation of the respective case suffixes of their arguments (Table 2). For consistency, we remove plurality markers from English

---

[2] Specifically, let E_t and E_{ng} be word and n-gram embedding matrices, and let t_w and NG_w be the word and the set of all n-grams of lengths 1 to 5, for a given word w. The final vector representation of w, e_w, is given by $e_w = E_t[t] + \sum_{ng \in NG_w} E_{ng}[ng]$.

[3] The verb in Basque agrees with the indirect object as well. In preliminary experiments, the recall of models trained on indirect object prediction was very low, due to the small number of indirect objects in the training corpus; we therefore do not include this task.

verbs before adding our suffixes (for example, by replacing *has* with *have*).

Single Task Results The basic results are summarized in Table 1. Recall is calculated as the proportion of the sentences with a direct object for which the model predicted either SINGULAR or PLURAL, but not NONE. Since all verbs included in the experiment had a subject, subject recall was 100% and is therefore not reported.

Plurality prediction accuracy was higher for subjects than objects. Recall for object prediction was 81.8%, indicating that in many sentences the model was unable to identify the direct object. The lower performance on object plurality prediction is likely to be due to the fact that only about third of the sentences contain a direct object. This hypothesis is supported by the results of a preliminary experiment, in which the model was trained only on transitive sentences (with a direct object). Transitive-only training led to a reversal of the pattern: object plurality was predicted with higher accuracy than subject plurality. We conjecture that this is due to the fact that most noun modifiers in English follow the head, making the head of the object, which in general determines the plurality of the phrase, closer on average to the verb than the head of the subject (see Table 3 below).

The accuracy we report for subject prediction, 94.7%, is lower than the accuracy of over 99% reported by Linzen et al. (2016). This may be due to one of several reasons. First, our training set was smaller: ∼35,000 sentences in our treebank corpus compared to ∼121,000 in their automatically parsed corpus. Second, sentences in the Wall Street Journal corpus may be more syntactically complex on average than sentences in Wikipedia, making it more challenging to identify the verb's arguments. Finally, we predicted agreement in all tenses, whereas Linzen et al. (2016) limited their study to the present tense (where English does in fact show agreement); it may be the case that sentences with past tense verbs are on average more complex than those with present tense verbs, regardless of the corpus.

Multitask Training Accuracy was higher in the joint setting: polypersonal agreement prediction is easier for the model. Subject prediction accuracy rose from 94.7% to 95.7%, object precision was slightly higher (90.0% compared to 88.9%), and object recall was significantly higher, increasing from 81.8% to 85.4%. We hypothesize that supervision signals from the prediction of both arguments lead to more robust abstract syntactic representations that transfer across the two tasks (Enguehard et al., 2017); for example, the model may be better able to identify the head of a noun phrase, regardless of whether it is the subject or the object. These findings suggest that when training on an auxiliary agreement prediction task in order to improve a language model's syntactic performance, additional supervision—in the form of predicting both subject and object—may be beneficial.

4 Order of Core Elements

Languages vary in the typical order of the core elements of a clause: the subject, the object and the verb (Dryer, 2013). For example, whereas in English the canonical order is Subject-Verb-Object (SVO, *The priests are reading the book*), in Irish it is Verb-Subject-Object (VSO, Dillon and Ó Cróinin 1961):

(5)	Léann	[na	sagairt]	[na	leabhair].
	read.PRES	the.PL	priest.PL	the.PL	book.PL

'The priests are reading the books.'

While there are six possible orderings of these three elements, in most human languages the subject precedes both the object and the verb: about 86.5% of the languages use either SOV or SVO orders, 9% of the languages use VOS order, and OVS and OSV languages are extremely rare (Tomlin, 1986).

To test whether RNNs have inductive biases favoring certain word orders over others, we created synthetic versions of English with all six possible orders of core elements. While natural languages often allow at least a limited degree of word order flexibility, our experiments used a simplified setting in which word order was either completely fixed (e.g., always SVO) or fully flexible, where one of the six orders was selected uniformly at random for each sentence in the corpus (the same order is used for all of the clauses of the sentence).

4.1 Corpus Creation

Given a dependency parse for a sentence, we modulated the order of the subject and object nodes with respect to their verb. When changing the position of an argument node, we moved the entire subtree rooted in that node, including verbs and other arguments in this subtree. In the permutation process, we moved to the subject position not only nominal subjects (*nsubj* and *nsubjpass* edges in UD), but also clausal subjects (*csubj* edges). Similarly, we moved to the object position not only nominal objects (*dobj* edge), but also clausal complements (*ccomp* and *xcomp*).

We kept negations, adverbial modifiers, particles and auxiliaries in their original position with respect to the verb. Other non-core dependents of the verb (i.e. not the subject or the object), such as prepositional phrases, were placed according to their original position relative to the verb. For instance, in the clause *the broker took them out for lunch*, the phrase *for lunch* appeared directly following the verb and the arguments of the subtree in which it resides (*took*, *them*, *the broker*) in all word orders, reflecting its original position relative to the verb *took* (see Figure 1). Relative pronouns and complementizers remained in their original position.[4]

In all experiments in this section, we trained the model to jointly predict the plurality of the subject and the object. For consistency across the object and subject plurality prediction tasks, we used the polypersonal agreement markers on all verbs in the sentence (except, of course, for the prediction target, which was withheld completely). For example, in the OVS version of the sentence presented in Figure 1, the input was (6), where *kon* marks the fact that *say* has a plural subject:

(6) them ⟨verb⟩ out frequently the broker for lunch say*kon* they .

4.2 Results

Performance varied significantly across word orders (Table 3). Subject plurality prediction accuracy was inversely correlated with the frequency of attractors (intervening nouns of the opposite plurality) in the language: accuracy was lowest for subject prediction in the VOS and SOV languages, in which objects intervene between the subject and the verb (Figure 2). The degraded performance in these languages is consistent with the attraction effects found in previous studies of agreement in natural languages (Linzen et al., 2016; Gulordava et al., 2018), and support the hypothesis that RNNs have an inductive bias favoring dependencies with recent elements; we test this hypothesis in a more controlled way in §4.3.

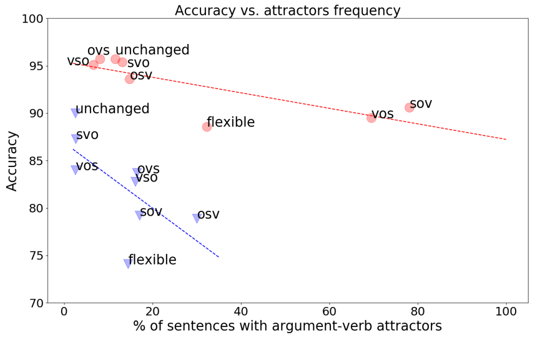


Figure 2: Subject and object plurality prediction accuracy as a function of the percentage of sentences with attractors that are arguments of the verb. Red circles represent subject prediction and blue triangles represent object prediction. R^2: 0.61 for subject, 0.43 for object.

Attractors affected object prediction accuracy as well. The highest accuracy among the synthetic languages was in the SVO language and the worst performance observed in the OSV language. As in §3, subjects were easier to predict than objects, likely because all verbs in the training set had a subject, but only 35% had an object.

Flexible word order was especially challenging for the model, with a subject plurality prediction accuracy of 88.6%, object plurality prediction accuracy of 74.1%, and object recall of 60.2%. This does not necessarily bear on the RNNs' inductive biases: flexible word order without case marking would make it difficult for any learner to infer syntactic relations. Without overt cues, the model must resort to selectional restrictions (e.g., in *the apples ate the man*, the only plausible subject is *the man*), but those are difficult to learn from a small corpus. What's more, some sentences are truly ambiguous when there are no case markers or word order cues; this happens for example when both arguments are animate, as in *the lawyer saw the doctor* (Gibson et al., 2013; Ettinger et al., 2018).

4.3 Withholding Direct Objects in Training

The previous experiments suggested that the RNN has a tendency to identify the more recent argument as the subject, leading to attraction effects

---

[4] For example, the result of transforming (i) to VSO word order was (ii) rather than (iii):

(i) But these are not the differences that make headlines.

(ii) But are these not the differences that make headlines.

(iii) But are these not the differences make headlines that.

	Subject		Object		
Order	% Attractors	Accuracy	% Attractors	Accuracy	Recall
Unchanged	11.56	95.7 ± 0.23	2.55	90.0 ± 0.1	85.4 ± 2.37
SVO	13.16	95.4 ± 0.41	2.6	87.3 ± 0.23	80.0 ± 2.61
SOV	78.12	90.6 ± 0.37	17.04	79.2 ± 0.78	63.3 ± 4.62
VOS	69.50	89.5 ± 0.54	2.57	84.0 ± 0.39	77.8 ± 3.68
VSO	6.65	95.1 ± 0.12	16.09	82.8 ± 0.7	70.0 ± 1.91
OSV	14.81	93.6 ± 0.23	30.00	78.9 ± 0.17	63.5 ± 4.59
OVS	8.13	95.7 ± 0.37	16.42	83.7 ± 0.32	72.8 ± 1.58
Flexible	32.24	88.6 ± 0.43	14.44	74.1 ± 0.70	60.2 ± 3.24

Table 3: Subject and object plurality prediction for different word orders (recall for the subject is 100% and is not indicated). The % attractors columns indicate the percentage of sentences containing verb-argument attractors. The number are averaged over four runs and the error interval represents the standard deviation.

caused by the object. We conjectured that this is due to the fact that many verbs are intransitive, that is, have a subject but not an object. The clauses in which those verbs appear provide ambiguous evidence: they are equally compatible with a generalization in which the subject is the *most recent* core element before the verb, and with a generalization in which the subject is the *first* core constituent of the clause. Attraction effects suggest that the inductive bias of the RNN leads it to adopt the incorrect recency-based generalization. To test this hypothesis in a controlled way, we adopt the "poverty of the stimulus" paradigm (Wilson, 2006; Culbertson and Adger, 2014; McCoy et al., 2018): we withhold all evidence that disambiguates these two hypotheses (namely, all transitive sentences), and test how the RNN generalizes to the withheld sentence type.

We used the SOV and VOS corpora described before; in both of these languages, the object intervened between the subject and the verb, potentially causing agreement attraction. Crucially, we train only on sentences *without* a direct object, and test on the following three types of sentences:

1. Sentences with an object of the opposite plurality from the subject (object attractor).

2. Sentences with an object of the same plurality as the subject (non-attractor object).[5]

3. Sentences without an object, but with one or more nouns of the opposite plurality intervening between the subject and the verb (non-object attractor); e.g., *The gap between winners and losers will grow* is intransitive, but the plural words *winners* and *losers*, which are a part of a modifier of the subject, may serve as attractors for the singular subject *gap*.

The results are shown in Table 4. Withholding direct objects during training dramatically degraded the performance of the model on sentences with an object attractor: the accuracy decreased from 90.6% for the model trained on the full SOV corpus (Table 3) to 60.0% for the model trained only on intransitive sentences from the same corpus. There was an analogous drop in performance in the case of VOS (89.5% compared to 48.3%). By contrast, attractors that were not core arguments, or objects that were not attractors, did not hurt performance in a comparable way. This suggests that in our poverty of the stimulus experiments RNNs were able to distinguish between core and non-core elements, but struggled on instances in which where the object directly preceded the verb (the instances that were withheld in training). This constitutes strong evidence for the RNN's recency bias: our models extracted the generalization that subjects directly precede the verb, even though the data were equally compatible with the generalization that the subject is the first core argument in the clause.

These findings align with the results of Khandelwal et al. (2018), who demonstrated that RNN language models are more sensitive to perturbations in recent input words compared with perturbations to more distant parts of the input. While in their case the model's recency preference can

---

[5] When the object is a noun-noun compound, it is considered a non-attractor if its head is not of the opposite plurality of the subject, regardless of the plurality of other elements. This can only make the task harder compared with the alternative of considering compound objects such as "screen displays" as attractors for plural subjects.

	Object (attractor)	Object (non attractor)	Non-object attractor
SOV	60.3 ± 3.7	92.8 ± 0.3	79.2 ± 3
VOS	48.3 ± 2.3	94.0 ± 2.3	83.1 ± 1.1

Table 4: Subject prediction accuracy in the "poverty of the stimulus" paradigm of Section 4.3, where transitive sentences were withheld during training. Numbers are averaged over four runs and the error interval represents the standard deviation.

be a learned property (since recent information is more relevant for the task of next-word prediction), our experiment focuses on the inherent inductive biases of the model, as the cues that are necessary for differentiating between the two generalizations were absent in training.

4.4 Discussion

Our reordering manipulation was limited to core element (subjects, objects and verbs). Languages also differ in word order inside other types of phrases, including noun phrases (e.g., does an adjective precede or follow the noun?), adpositional phrases (does the language use prepositions or postpositions?), and so on. Greenberg (1963) pointed out correlations between head-modifier orders across phrase categories; while a significant number of exceptions exist, these correlations have motivated proposals for a language-wide setting of a Head Directionality Parameter (Stowell, 1981; Baker, 2001). In future work, we would like to explore whether consistent reordering across categories improves the model's performance.

In practice, even languages with a relatively rigid word order almost never enforce this order in every clause. The order of elements in English, for example, is predominately SVO, but constructions in which the verb precedes the subject do exist, e.g., *Outside were three police officers*. Other languages are considerably more flexible than English (Dryer, 2013). Given that word order flexibility makes the task more difficult, our setting is arguably simpler than the task the model would face when learning a natural language.

The fact that the agreement dependency between the subject and the verb was more challenging to establish in the SOV order compared to the SVO order is consistent with the hypothesis that SVO languages make it easier to distinguish the subject from the object (Gibson et al., 2013); indeed, to compensate for this issue, SOV languages more frequently employ case marking (Matthew Dryer, quoted in Gibson et al. 2013).

There was not a clear relationship between the prevalence of a particular word order in the languages of the world and the difficulty that our models experienced with that order. The model performed best on the OVS word order, which is present in a very small number of languages ($\sim$1%). SOV languages were more difficult for our RNNs to learn than SVO languages, even though SOV languages are somewhat *more* common (Dryer, 2013). These results weakly support functional explanations of these typological tendencies; such explanations appeal to communicative efficiency considerations rather than learning biases (Maurits et al., 2010). Of course, since the inductive biases of humans and RNNs are likely to be different in many respects, our results do not rule out the possibility that the distribution of word orders is driven by a human learning bias after all.

5 Overt Morphological Case Systems

The vast majority of noun phrases in English are not overtly marked for grammatical function (case), with the exception of pronouns; e.g., the first-person singular pronoun is *I* when it is a subject and *me* when it is an object. Other languages mark case on most nouns. Consider, for example, the following example from Russian:[6]

(7) a. ya kupil knig-u.
I bought book-OBJECT
'I bought the book.'
b. knig-a ischezla.
book-SUBJECT disappeared
'The book disappeared.'

Overt case marking reduces ambiguity and facilitates parsing languages with flexible word order. To investigate the influence of case on agreement prediction—and on the ability to infer sentence structure—we experimented with different case systems. In all settings, we used "fused" suffixes, which encode both plurality and grammatical function. We considered three case systems (see Figure 1):

1. An unambiguous case system, with a unique

---

[6] The standard grammatical term for these cases are nominative (for subject) and accusative (for object); we use SUBJECT and OBJECT for clarity.

Case system	Flexible word order		VOS		OVS	
	Subject A	Object A/R	Subject A	Object A/R	Subject A	Object A/R
Unambiguous	99.2 ± 0.5	98.7 ± 0.2 $/98.0 \pm 0.5$	98.9 ± 0.2	99.5 ± 0.1 $/99.1 \pm 0.1$	99.5 ± 0.2	98.6 ± 0.3 $/98.4 \pm 0.6$
Syncretic	99.3 ± 0.2	93.6 ± 0.4 $/88.9 \pm 1.7$	99.1 ± 0.2	97.1 ± 0.2 $/95.0 \pm 1.1$	99.4 ± 0.1	97.8 ± 0.2 $/97.4 \pm 1.2$
Argument marking	96.0 ± 0.3	86.1 ± 0.9 $/79.7 \pm 4.9$	96.9 ± 0.1	93.6 ± 0.1 $/89.8 \pm 2.4$	99.6 ± 0.1	96.8 ± 0.1 $/95.5 \pm 0.5$

Table 5: Accuracy (A) and recall (R) in predicting subject and object agreement with different case systems.

suffix for each combination of number and grammatical function.

2. A partially syncretic (ambiguous) case system, in which the same suffix was attached to both singular subjects and plural objects (modeled after Basque).

3. A fully syncretic case system (argument marking only): the suffix indicated only the plurality of the argument, regardless of its grammatical function (cf. subject/object syncretism in Russian neuter nouns).

In the typological survey reported in Baerman and Brown (2013), 62% of the languages had no or minimal case marking, 20% had syncretic case systems, and 18% had case systems with no syncretism.

Corpus Creation The suffixes we used are listed in Table 2. We only attached the suffix to the head of the relevant argument; adjectives and other modifiers did not carry case suffixes. The same suffix was used to mark plurality/case on noun and the agreement features on the verb; e.g., if the verb *eat* had a singular subject and plural object, it appeared as eat*karker* (the singular subject suffix was *kar* and the plural object suffix was *ker*). We stripped off plurality and case markers from the original English noun phrases before adding these suffixes.

Setup We evaluated the interaction between different case marking schemes and three word orders: flexible word order and the two orders on which the model achieved the best (OVS) and worst (VOS) subject prediction accuracy. We train one model for each combination of case system and word order. We jointly predicted the plurality of subject and the object.

Results and Analysis The results are summarized in Table 5. Unambiguous case marking dramatically improved subject and object plurality prediction compared with the previous experiments; accuracy was above 98% for all three word orders. Partial syncretism hurt performance somewhat relative to the unambiguous setting (except with flexible word order), especially for object prediction. The fully syncretic case system, which marked only the plurality of the head of each argument, further decreased performance. At the same time, even this limited marking scheme was helpful: accuracy in the most challenging setting, flexible word order (subject: 96.0%; object: 86.1%), was not very different from the results on unmodified English (95.7% and 90.0%). This contrasts with the poor results on the flexible setting without cases (subject: 88.6%; object: 60.2%). On the rigid orders, a fully syncretic system still significantly improved agreement prediction. The moderate effect of case syncretism on performance suggests that most of the benefits of case marking stems from the overt marking of the heads of all arguments.

Overall, these results are consistent with the observation that languages with explicit case marking tend to allow a more flexible word orders compared with languages such as English that make use of word order to express grammatical function of words.

6 Related Work

Our approach of constructing synthetic languages by parametrically modifying parsed corpora for natural languages is closely inspired by Wang and Eisner (2016) (see also Wang and Eisner 2017). While they trained a model to mimic the POS tags order-statistics of the target language, we manually modified the parsed corpora; this allows us to control for selected parameters, at the expense of reducing generality.

Simpler synthetic languages (not based on natural corpora) have been used in a number of recent studies to examine the inductive biases of different neural architectures (Bowman et al., 2015; Lake and Baroni, 2018; McCoy et al., 2018). In another recent study, Cotterell et al. (2018) measured the ability of RNN and n-gram models to perform character-level language modeling in a sample of languages, using a parallel corpus; the main typological property of interest in that study was morphological complexity. Finally, a large number of studies, some mentioned in the introduction, have used syntactic prediction tasks to examine the generalizations acquired by neural models (see also Bernardy and Lappin 2017; Futrell et al. 2018; Lau et al. 2017; Conneau et al. 2018; Ettinger et al. 2018; Jumelet and Hupkes 2018).

7 Conclusions

We have proposed a methodology for generating parametric variations of existing languages and evaluating the performance of RNNs in syntactic feature prediction in the resulting languages. We used this methodology to study the grammatical inductive biases of RNNs, assessed whether certain grammatical phenomena are more challenging for RNNs to learn than others, and began to compare these patterns with the linguistic typology literature.

In our experiments, multitask training on polypersonal agreement prediction improved performance, suggesting that the models acquired syntactic representations that generalize across argument types (subjects and objects). Performance varied significantly across word orders. This variation was not correlated with the frequency of the word orders in the languages of the world. Instead, it was inversely correlated with the frequency of attractors, demonstrating a recency bias. Further supporting this bias, in a poverty-of -the-stimulus paradigm, where the data were equally consistent with two generalizations—first, the generalization that the subject is the first argument in the clause, and second, the generalization that the subject is the most recent argument preceding the verb—RNNs adopted the recency-based generalization. Finally, we found that overt case marking on the heads of arguments dramatically improved plurality prediction performance, even when the case system was highly syncretic.

Agreement feature prediction in some of our synthetic languages is likely to be difficult not only for RNNs but for many other classes of learners, including humans. For example, agreement in a language with very flexible word order and without case marking is impossible to predict in many cases (see §4.2), and indeed such languages are very rare. In future work, a human experiment based on the agreement prediction task can help determine whether the difficulty of our languages is consistent across humans and RNNs.

Acknowledgements

This work is supported by the Israeli Science Foundation (grant number 1555/15) and by Theo Hoffenberg, the founder & CEO of Reverso.

References

Matthew Baerman and Dunstan Brown. 2013. Case syncretism. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Mark C. Baker. 2001. *The atoms of language: The mind's hidden rules of grammar*. Basic Books, New York.

Jean-Philippe Bernardy and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *LiLT (Linguistic Issues in Language Technology)*, 15.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

Shammur Absar Chowdhury and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* , pages 536–541. Association for Computational Linguistics.

Jennifer Culbertson and David Adger. 2014. Language learners privilege structured meaning over surface frequency. *Proceedings of the National Academy of Sciences*, page 201320525.

Myles Dillon and Donncha Ó Cróinin. 1961. *Teach Yourself Irish*. The English Universities Press Ltd., London.

Matthew S. Dryer. 2013. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 3–14.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801. Association for Computational Linguistics.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. RNNs as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Edward Gibson, Steven T. Piantadosi, Kimberly Brink, Leon Bergen, Eunice Lim, and Rebecca Saxe. 2013. A noisy-channel account of crosslinguistic word-order variation. *Psychological Science*, 24(7):1079–1088.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248. Association for Computational Linguistics.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of language*, pages 73–113. MIT Press, Cambridge, MA.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1195–1205.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? on the ability of lstms to understand negative polarity items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231. Association for Computational Linguistics.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 284–294.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint 1412.6980*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 2879–2888.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1247.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Luke Maurits, Danielle J. Navarro, and Amy Perfors. 2010. Why are some word orders more common than others? A uniform information density account. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 1585–1593. Curran Associates, Inc.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: Hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2093—2098.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Shauli Ravfogel, Francis Tyers, and Yoav Goldberg. 2018. Can LSTM learn to capture agreement? the case of Basque. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 98–107. Association for Computational Linguistics.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Timothy Angus Stowell. 1981. *Origins of phrase structure*. Ph.D. thesis, Massachusetts Institute of Technology.

Rudolf S. Tomlin. 1986. *Basic word order: functional principles*. Croom Helm, London.

Marten van Schijndel and Tal Linzen. 2018. Modeling garden path effects without explicit hierarchical syntax. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2600–2605, Austin, TX. Cognitive Science Society.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with supervised learning. *Transactions of the Association for Computational Linguistics*, 5:147–161.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221. Association for Computational Linguistics.

Colin Wilson. 2006. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science*, 30:945–982.

Attention is not Explanation

Sarthak Jain
Northeastern University
`jain.sar@husky.neu.edu`

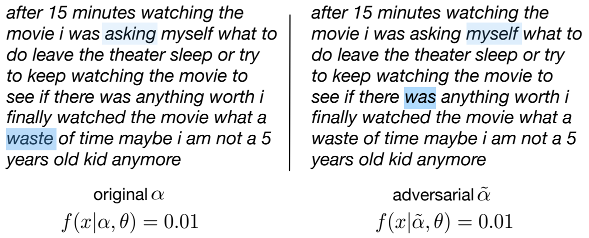
Byron C. Wallace
Northeastern University
`b.wallace@northeastern.edu`

Abstract

Attention mechanisms have seen wide adoption in neural NLP models. In addition to improving predictive performance, these are often touted as affording transparency: models equipped with attention provide a distribution over attended-to input units, and this is often presented (at least implicitly) as communicating the relative importance of inputs. However, it is unclear what relationship exists between attention weights and model outputs. In this work we perform extensive experiments across a variety of NLP tasks that aim to assess the degree to which attention weights provide meaningful "explanations" for predictions. We find that they largely do not. For example, learned attention weights are frequently uncorrelated with gradient-based measures of feature importance, and one can identify very different attention distributions that nonetheless yield equivalent predictions. Our findings show that standard attention modules do not provide meaningful explanations and should not be treated as though they do. Code to reproduce all experiments is available at `https://github.com/successar/AttentionExplanation`.

1 Introduction and Motivation

Attention mechanisms (Bahdanau et al., 2014) induce conditional distributions over input units to compose a weighted context vector for downstream modules. These are now a near-ubiquitous component of neural NLP architectures. Attention weights are often claimed (implicitly or explicitly) to afford insights into the "inner-workings" of models: for a given output one can inspect the inputs to which the model assigned large attention weights. Li *et al.* (2016) summarized this commonly held view in NLP: "Attention provides an important way to explain the workings of neural models". Indeed, claims that attention provides interpretability are common in the literature, e.g., (Xu et al., 2015; Choi et al., 2016; Lei et al., 2017; Martins and Astudillo, 2016; Xie et al., 2017).[1]



after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

Figure 1: Heatmap of attention weights induced over a negative movie review. We show observed model attention (left) and an adversarially constructed set of attention weights (right). Despite being quite dissimilar, these both yield effectively the same prediction (0.01).

Implicit in this is the assumption that the input units (e.g., words) accorded high attention weights are responsible for model outputs. But as far as we are aware, this assumption has not been formally evaluated, and our findings here suggest that it is problematic. More specifically, we empirically investigate the relationship between attention weights, inputs, and outputs. Assuming attention provides an explanation for model predictions, we might expect the following properties to hold. (i) Attention weights should correlate with feature importance measures (e.g., gradient-based measures); (ii) Alternative (or *counterfactual*) attention weight configurations ought to yield corresponding changes in prediction (and if they do not then are equally plausible as explanations). We report that neither property is consistently observed by standard attention mechanisms in the context of text classification, question answering (QA), and Natural Language Inference (NLI) tasks when RNN encoders are used.

---

[1] We do not intend to single out any particular work; indeed one of the authors has himself presented (supervised) attention as providing interpretability (Zhang et al., 2016).

Proceedings of NAACL-HLT 2019, pages 3543–3556
Minneapolis, Minnesota, June 2 - June 7, 2019.

Consider Figure 1. The left panel shows the original attention distribution α over the words of a particular movie review using a standard attentive BiLSTM architecture for sentiment analysis. It is tempting to conclude from this that the token *waste* is largely responsible for the model coming to its disposition of 'negative' ($\hat{y} = 0.01$). But one can construct an alternative attention distribution $\tilde{\alpha}$ (right panel) that attends to entirely different tokens yet yields an essentially identical prediction (holding all other parameters of f, θ, constant).

Such counterfactual distributions imply that explaining the original prediction by highlighting attended-to tokens is misleading insofar as alternative attention distributions would have yielded an equivalent prediction (e.g., one might conclude from the right panel that model output was due primarily to *was* rather than *waste*). Further, the attention weights in this case correlate only weakly with gradient-based measures of feature importance ($\tau_g = 0.29$). And arbitrarily permuting the entries in α yields a median output difference of 0.006 with the original prediction.

These and similar findings call into question the view that attention provides meaningful insight into model predictions. We thus caution against using attention weights to highlight input tokens "responsible for" model outputs and constructing just-so stories on this basis, particularly with complex encoders.

Research questions and contributions. We examine the extent to which the (often implicit) narrative that attention provides model *transparency*[2] holds across tasks by exploring the following empirical questions.

1. To what extent do induced attention weights correlate with measures of feature importance – specifically, those resulting from gradients and *leave-one-out* methods?

2. Would alternative attention weights (and hence distinct heatmaps/"explanations") necessarily yield different predictions?

Our findings with respect to these questions (assuming a BiRNN encoder) are summarized as follows: (1) Only weakly and inconsistently, and, (2) No; it is very often possible to construct *adversarial* attention distributions that yield effectively equivalent predictions as when using the originally induced attention weights, despite attending to entirely different input features. Further, randomly permuting attention weights often induces only minimal changes in output.

2 Preliminaries and Assumptions

We consider exemplar NLP tasks for which attention mechanisms are commonly used: classification, natural language inference (NLI), and question answering.[3] We adopt the following general modeling assumptions and notation.

We assume model inputs $\mathbf{x} \in \mathbb{R}^{T \times |V|}$, composed of one-hot encoded words at each position. These are passed through an embedding matrix $\mathbf{E}$ which provides dense (d dimensional) token representations $\mathbf{x}_e \in \mathbb{R}^{T \times d}$. Next, an encoder **Enc** consumes the embedded tokens in order, producing T m-dimensional hidden states: $\mathbf{h} = \textbf{Enc}(\mathbf{x}_e) \in \mathbb{R}^{T \times m}$. We predominantly consider a Bi-RNN as the encoder module, but for completeness we also analyze convolutional and (unordered) 'average embedding' variants.[4]

A similarity function ϕ maps $\mathbf{h}$ and a query $\mathbf{Q} \in \mathbb{R}^m$ (e.g., hidden representation of a question in QA, or the hypothesis in NLI) to scalar scores, and attention is then induced over these: $\hat{\boldsymbol{\alpha}} = \text{softmax}(\phi(\mathbf{h}, \mathbf{Q})) \in \mathbb{R}^T$. In this work we consider two common similarity functions: *Additive* $\phi(\mathbf{h}, \mathbf{Q}) = \mathbf{v}^T \tanh(\mathbf{W_1 h} + \mathbf{W_2 Q})$ (Bahdanau et al., 2014) and *Scaled Dot-Product* $\phi(\mathbf{h}, \mathbf{Q}) = \frac{\mathbf{hQ}}{\sqrt{m}}$ (Vaswani et al., 2017), where $\mathbf{v}, \mathbf{W_1}, \mathbf{W_2}$ are model parameters.

Finally, a dense layer **Dec** with parameters $\boldsymbol{\theta}$ consumes a weighted instance representation and yields a prediction $\hat{y} = \sigma(\boldsymbol{\theta} \cdot h_\alpha) \in \mathbb{R}^{|\mathcal{Y}|}$, where $h_\alpha = \sum_{t=1}^{T} \hat{\alpha}_t \cdot h_t$; σ is an output activation function; and $|\mathcal{Y}|$ denotes the label set size.

3 Datasets and Tasks

For **binary text classification**, we use:

Stanford Sentiment Treebank (SST) (Socher et al., 2013). 10,662 sentences tagged with sentiment on a scale from 1 (most negative) to 5 (most positive). We filter out neutral instances and dichotomize the remaining sentences into positive (4, 5) and negative (1, 2).

---

[2] Defined as per (Lipton, 2016); we are interested in whether attended-to features are responsible for outputs.

[3] While attention is perhaps *most* common in seq2seq tasks like translation, our impression is that interpretability is not typically emphasized for such tasks, in general.

[4] In the latter case, h_t is the embedding of token t after being passed through a linear layer and ReLU activation.

Dataset	$\|V\|$	*Avg. length*	*Train size*	*Test size*	*Test performance*
SST	16175	19	3034 / 3321	863 / 862	0.81
IMDB	13916	179	12500 / 12500	2184 / 2172	0.88
ADR Tweets	8686	20	14446 / 1939	3636 / 487	0.61
20 Newsgroups	8853	115	716 / 710	151 / 183	0.94
AG News	14752	36	30000 / 30000	1900 / 1900	0.96
Diabetes (MIMIC)	22316	1858	6381 / 1353	1295 / 319	0.79
Anemia (MIMIC)	19743	2188	1847 / 3251	460 / 802	0.92
CNN	74790	761	380298	3198	0.64
bAbI (Task 1 / 2 / 3)	40	8 / 67 / 421	10000	1000	1.0 / 0.65 / 0.64
SNLI	20982	14	182764 / 183187 / 183416	3219 / 3237 / 3368	0.78

Table 1: Dataset characteristics. For train and test size, we list the cardinality for each class, where applicable: 0/1 for binary classification (top), and 0 / 1 / 2 for NLI (bottom). Average length is in tokens. Test metrics are F1 score, accuracy, and micro-F1 for classification, QA, and NLI, respectively; all correspond to performance using a BiLSTM encoder. We note that results using convolutional and average (i.e., non-recurrent) encoders are comparable for classification though markedly worse for QA tasks.

IMDB Large Movie Reviews Corpus (Maas et al., 2011). Binary sentiment classification dataset containing 50,000 polarized (positive or negative) movie reviews, split into half for training and testing.

Twitter Adverse Drug Reaction dataset (Nikfarjam et al., 2015). A corpus of ∼8000 tweets retrieved from Twitter, annotated by domain experts as mentioning adverse drug reactions.

20 Newsgroups (Hockey vs Baseball). Collection of ∼20,000 newsgroup correspondences, partitioned (nearly) evenly across 20 categories. We extract instances belonging to *baseball* and *hockey*, which we designate as 0 and 1, respectively, to derive a binary classification task.

AG News Corpus (Business vs World).[5] 496,835 news articles from 2000+ sources. We follow (Zhang et al., 2015) in filtering out all but the top 4 categories. We consider the binary classification task of discriminating between *world* (0) and *business* (1) articles.

MIMIC ICD9 (Diabetes) (Johnson et al., 2016). A subset of discharge summaries from the MIMIC III dataset of electronic health records. The task is to recognize if a given summary has been labeled with the ICD9 code for diabetes (or not).

MIMIC ICD9 (Chronic vs Acute Anemia) (Johnson et al., 2016). A subset of discharge summaries from MIMIC III dataset (Johnson et al., 2016) known to correspond to patients with anemia. Here the task to distinguish the type of anemia for each report – *acute* (0) or *chronic* (1).

For **Question Answering (QA)**:

CNN News Articles (Hermann et al., 2015). A corpus of cloze-style questions created via automatic parsing of news articles from CNN. Each instance comprises a paragraph-question-answer triplet, where the answer is one of the anonymized entities in the paragraph.

bAbI (Weston et al., 2015). We consider the three tasks presented in the original bAbI dataset paper, training separate models for each. These entail finding (i) a single supporting fact for a question and (ii) two or (iii) three supporting statements, chained together to compose a coherent line of reasoning.

Finally, for **Natural Language Inference** (NLI):

The *SNLI dataset* (Bowman et al., 2015). 570k human-written English sentence pairs manually labeled for balanced classification with the labels *neutral*, *contradiction*, and *entailment*, supporting the task of natural language inference (NLI). In this work, we generate an attention distribution over premise words conditioned on the hidden representation induced for the hypothesis.

We restrict ourselves to comparatively simple instantiations of attention mechanisms, as described in the preceding section. This means we do not consider recently proposed 'BiAttentive' architectures that attend to tokens in the respective inputs, conditioned on the other inputs (Parikh et al., 2016; Seo et al., 2016; Xiong et al., 2016).

Table 1 provides summary statistics for all datasets, as well as the observed test performances for additional context.

4 Experiments

We run a battery of experiments that aim to examine empirical properties of learned attention weights and to interrogate their interpretability and transparency. The key questions are: *Do*

---

[5] `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

		Gradient (BiLSTM) τ_g		Gradient (Average) τ_g		Leave-One-Out (BiLSTM) τ_{loo}	
Dataset	Class	Mean ± Std.	Sig. Frac.	Mean ± Std.	Sig. Frac.	Mean ± Std.	Sig. Frac.
SST	0	0.34 ± 0.21	0.48	0.61 ± 0.20	0.87	0.27 ± 0.19	0.33
	1	0.36 ± 0.21	0.49	0.60 ± 0.21	0.83	0.32 ± 0.19	0.40
IMDB	0	0.44 ± 0.06	1.00	0.67 ± 0.05	1.00	0.34 ± 0.07	1.00
	1	0.43 ± 0.06	1.00	0.68 ± 0.05	1.00	0.34 ± 0.07	0.99
ADR Tweets	0	0.47 ± 0.18	0.76	0.73 ± 0.13	0.96	0.29 ± 0.20	0.44
	1	0.49 ± 0.15	0.85	0.72 ± 0.12	0.97	0.44 ± 0.16	0.74
20News	0	0.07 ± 0.17	0.37	0.79 ± 0.07	1.00	0.06 ± 0.15	0.29
	1	0.21 ± 0.22	0.61	0.75 ± 0.08	1.00	0.20 ± 0.20	0.62
AG News	0	0.36 ± 0.13	0.82	0.78 ± 0.07	1.00	0.30 ± 0.13	0.69
	1	0.42 ± 0.13	0.90	0.76 ± 0.07	1.00	0.43 ± 0.14	0.91
Diabetes	0	0.42 ± 0.05	1.00	0.75 ± 0.02	1.00	0.41 ± 0.05	1.00
	1	0.40 ± 0.05	1.00	0.75 ± 0.02	1.00	0.45 ± 0.05	1.00
Anemia	0	0.47 ± 0.05	1.00	0.77 ± 0.02	1.00	0.46 ± 0.05	1.00
	1	0.46 ± 0.06	1.00	0.77 ± 0.03	1.00	0.47 ± 0.06	1.00
CNN	Overall	0.24 ± 0.07	0.99	0.50 ± 0.10	1.00	0.20 ± 0.07	0.98
bAbI 1	Overall	0.25 ± 0.16	0.55	0.72 ± 0.12	0.99	0.16 ± 0.14	0.28
bAbI 2	Overall	−0.02 ± 0.14	0.27	0.68 ± 0.06	1.00	−0.01 ± 0.13	0.27
bAbI 3	Overall	0.24 ± 0.11	0.87	0.61 ± 0.13	1.00	0.26 ± 0.10	0.89
SNLI	0	0.31 ± 0.23	0.36	0.59 ± 0.18	0.80	0.16 ± 0.26	0.20
	1	0.33 ± 0.21	0.38	0.58 ± 0.19	0.80	0.36 ± 0.19	0.44
	2	0.31 ± 0.21	0.36	0.57 ± 0.19	0.80	0.34 ± 0.20	0.40

Table 2: Mean and std. dev. of correlations between gradient/leave-one-out importance measures and attention weights. *Sig. Frac.* columns report the fraction of instances for which this correlation is statistically significant; note that this largely depends on input length, as correlation does tend to exist, just weakly. Encoders are denoted parenthetically. These are representative results; exhaustive results for all encoders are available to browse online.

learned attention weights agree with alternative, natural measures of feature importance? And, *Had we attended to different features, would the prediction have been different*?

More specifically, in Section 4.1, we empirically analyze the correlation between gradient-based feature importance and learned attention weights, and between 'leave-one-out' (LOO) measures and the same. In Section 4.2 we then consider counterfactual (to those observed) attention distributions. Under the assumption that attention weights are explanatory, such counterfactual distributions may be viewed as alternative potential explanations; if these do not correspondingly change model output, then the original attention weights do not provide unique explanation for predictions, i.e., attending to other features could have resulted in the same output.

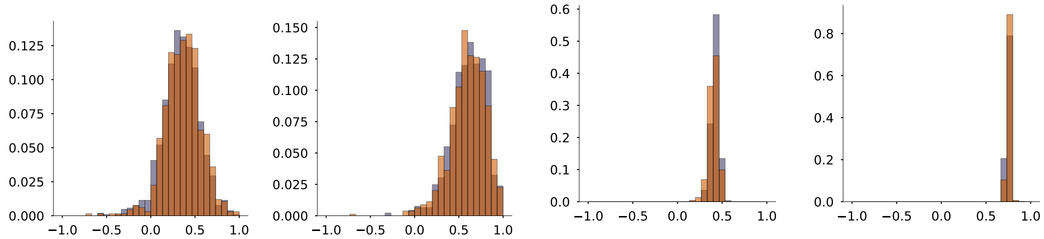
To generate counterfactual attention distributions, we first consider randomly permuting observed attention weights and recording associated changes in model outputs (4.2.1). We then propose explicitly searching for "adversarial" attention weights that maximally differ from the observed attention weights (which one might show in a heatmap and use to explain a model prediction), and yet yield an effectively equivalent prediction (4.2.2). The latter strategy also provides a useful potential metric for the reliability of attention weights as explanations: we can report a measure quantifying how different attention weights can be for a given instance without changing the model output by more than some threshold ϵ.

All results presented below are generated on test sets. We present results for *Additive* attention below. The results for *Scaled Dot Product* in its place are comparable. We provide a web interface to interactively browse the (very large set of) plots for all datasets, model variants, and experiment types: `https://successar.github.io/AttentionExplanation/docs/`.

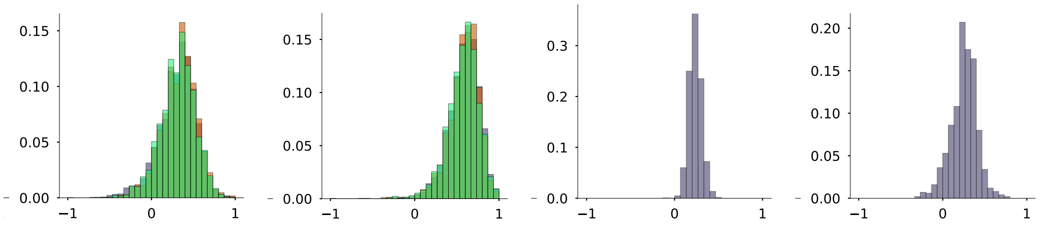
In the following sections, we use Total Variation Distance (TVD) as the measure of change between output distributions, defined as follows. $\text{TVD}(\hat{y}_1, \hat{y}_2) = \frac{1}{2}\sum_{i=1}^{|\mathcal{Y}|} |\hat{y}_{1i} - \hat{y}_{2i}|$. We use the Jensen-Shannon Divergence (JSD) to quantify the difference between two attention distributions: $\text{JSD}(\alpha_1, \alpha_2) = \frac{1}{2}\text{KL}[\alpha_1 || \frac{\alpha_1+\alpha_2}{2}] + \frac{1}{2}\text{KL}[\alpha_2 || \frac{\alpha_1+\alpha_2}{2}]$.

4.1 Correlation Between Attention and Feature Importance Measures

We empirically characterize the relationship between attention weights and corresponding feature importance scores. Specifically we measure correlations between attention and: (1) gradient based measures of feature importance (τ_g), and, (2) differences in model output induced by leaving features out (τ_{loo}). While these measures are themselves insufficient for interpretation of neu-



(a) SST (BiLSTM) (b) SST (Average) (c) Diabetes (BiLSTM) (d) Diabetes (Average)



(e) SNLI (BiLSTM) (f) SNLI (Average) (g) CNN-QA (BiLSTM) (h) BAbI 1 (BiLSTM)

Figure 2: Histogram of Kendall τ between attention and gradients. Encoder variants are denoted parenthetically; colors indicate predicted classes. Exhaustive results are available for perusal online. Best viewed in color.

ral model behavior (Feng et al., 2018), they do provide measures of individual feature importance with known semantics (Ross et al., 2017). It is thus instructive to ask whether these measures correlate with attention weights.

The process we follow to quantify this is described in Algorithm 1. We denote the input resulting from removing the word at position t in $\mathbf{x}$ by $\mathbf{x}_{-t}$. Note that we disconnect the computation graph at the attention module so that the gradient does not flow through this layer.

Algorithm 1 Feature Importance Computations

```
$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x}),\ \hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \alpha)$
$g_t \leftarrow |\sum_{w=1}^{|V|} \mathbb{1}[\mathbf{x}_{tw} = 1] \frac{\partial y}{\partial \mathbf{x}_{tw}}|\ , \forall t \in [1, T]$
$\tau_g \leftarrow \text{Kendall-}\tau(\alpha, g)$
$\Delta\hat{y}_t \leftarrow \text{TVD}(\hat{y}(\mathbf{x}_{-t}), \hat{y}(\mathbf{x}))\ , \forall t \in [1, T]$
$\tau_{loo} \leftarrow \text{Kendall-}\tau(\alpha, \Delta\hat{y})$
```

Table 2 reports summary statistics of Kendall τ correlations for each dataset. Full distributions are shown in Figure 2, which plots histograms of τ_g for every data point in the respective corpora. (Corresponding plots for τ_{loo} are similar and the full set can be browsed via the online supplement.) We plot these separately for each class: orange (🟧) represents instances predicted as positive, and purple (🟪) those predicted to be negative. For SNLI, colors 🟪, 🟧 and 🟩 code for contradiction, entailment, and neutral respectively.

In general, observed correlations are modest (recall: 0 indicates no correspondence, 1 implies perfect concordance) for the BiRNN encoder. The centrality of observed densities hovers around or below 0.5 in most of the corpora considered. Moreover, as per Table 2, correlation is sufficiently weak that a statistically significant correlation between attention weights and feature importance scores (both gradient and feature erasure based) cannot consistently be established across corpora.

In contrast, gradients in "average" embedding based models show very high degree of correspondence with attention weights – on average across corpora, correlation between LOO scores and attention weights is ∼0.375 points higher for this encoder, compared to the BiLSTM. These results suggest that, in general, attention weights do not strongly or consistently agree with such feature importance scores in models with contextualized embeddings. This is problematic for the view of attention weights as explanatory, given the face validity of input gradient/erasure based explanations (Ross et al., 2017; Li et al., 2016). On some datasets — notably the MIMIC tasks, and to a lesser extent the QA corpora — this correlation is consistently significant but remains relatively weak. This could be attributed to increased length of documents for these datasets providing stronger signal to standard hypothesis testing methods.

For reference we report correlations between gradients and LOO scores in the Appendix and online materials; these are consistently stronger than the correlation between attention weights and either feature importance score for the recurrent (BiLSTM) encoder. These exhibit, on average, a (i) 0.2 and (ii) ~ 0.25 greater correlation with each other than BiLSTM attention and (i) LOO and (ii) gradient scores.

4.2 Counterfactual Attention Weights

We next consider *what-if* scenarios corresponding to alternative (counterfactual) attention weights. The idea is to investigate whether the prediction would have been different, had the model emphasized (attended to) different input features. More precisely, suppose $\hat{\alpha} = \{\hat{\alpha}_t\}_{t=1}^{T}$ are the attention weights induced for an instance, giving rise to model output $\hat{y}$. We then consider counterfactual distributions over y, under alternative α.

We experiment with two means of constructing such distributions. First, we simply scramble the original attention weights $\hat{\alpha}$, re-assigning each value to an arbitrary, randomly sampled index (input feature). Second, we generate an *adversarial attention distribution*: this is a set of attention weights that is maximally distinct from $\hat{\alpha}$ but that nonetheless yields an equivalent prediction (i.e., prediction within some ϵ of $\hat{y}$).

4.2.1 Attention Permutation

To characterize model behavior when attention weights are shuffled, we follow Algorithm 2.

Algorithm 2 Permuting attention weights

```
$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x}),\ \hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \hat{\alpha})$
for $p \leftarrow 1$ to 100 do
    $\alpha^p \leftarrow \text{Permute}(\hat{\alpha})$
    $\hat{y}^p \leftarrow \text{Dec}(\mathbf{h}, \alpha^p)$        ▷ Note : $\mathbf{h}$ is not changed
    $\Delta\hat{y}^p \leftarrow \text{TVD}[\hat{y}^p, \hat{y}]$
end for
$\Delta\hat{y}^{med} \leftarrow \text{Median}_p(\Delta\hat{y}^p)$
```

Figure 3 depicts the relationship between the maximum attention value in the original $\hat{\alpha}$ and the median induced change in model output ($\Delta\hat{y}^{med}$) across instances in the respective datasets. Colors again indicate class predictions, as above.

We observe that there exist many points with small $\Delta\hat{y}^{med}$ despite large magnitude attention weights. These are cases in which the attention weights might suggest explaining an output by a small set of features (this is how one might reasonably read a heatmap depicting the attention weights), but where scrambling the attention makes little difference to the prediction.

In some cases, such as predicting ICD codes from notes using the MIMIC dataset, one can see different behavior for the respective classes. For the *Diabetes* task, e.g., attention behaves intuitively for at least the positive class; perturbing attention in this case causes large changes to the prediction. We again conjecture that this is due to a few tokens serving as high precision indicators for the positive class; in their absence (or when they are not attended to sufficiently), the prediction drops considerably. However, this is the exception rather than the rule.

4.2.2 Adversarial Attention

We next propose a more focused approach to counterfactual attention weights, which we will refer to as *adversarial attention*. The intuition is to explicitly seek out attention weights that differ as much as possible from the observed attention distribution and yet leave the prediction effectively unchanged. Such adversarial weights violate an intuitive property of explanations: shifting model attention to very different input features should yield corresponding changes in the output. Alternative attention distributions identified adversarially may then be viewed as equally plausible explanations for the same output.

Operationally, realizing this objective requires specifying a value ϵ that defines what qualifies as a "small" difference in model output. Once this is specified, we aim to find k adversarial distributions $\{\alpha^{(1)}, ..., \alpha^{(k)}\}$, such that each $\alpha^{(i)}$ maximizes the distance from original $\hat{\alpha}$ but does not change the output by more than ϵ. In practice we simply set this to 0.01 for text classification and 0.05 for QA datasets.[6]

We propose the following optimization problem to identify adversarial attention weights.

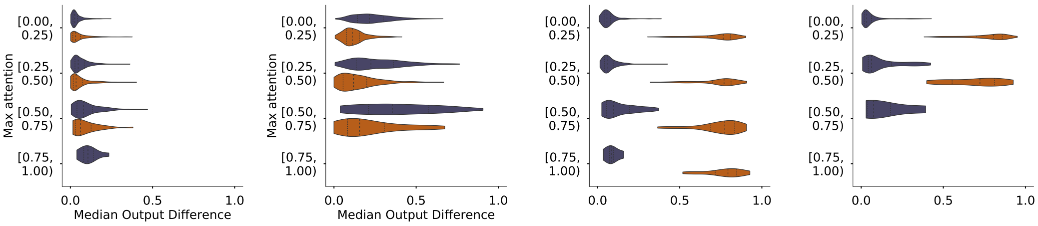
$$\begin{aligned} &\underset{\alpha^{(1)},...,\alpha^{(k)}}{\text{maximize}} && f(\{\alpha^{(i)}\}_{i=1}^{k}) \\ &\text{subject to} && \forall i\ \text{TVD}[\hat{y}(\mathbf{x}, \alpha^{(i)}), \hat{y}(\mathbf{x}, \hat{\alpha})] \leq \epsilon \end{aligned} \tag{1}$$

Where $f(\{\alpha^{(i)}\}_{i=1}^{k})$ is:

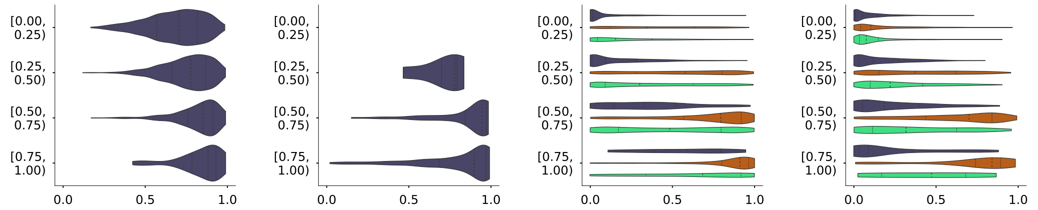
$$\sum_{i=1}^{k} \text{JSD}[\alpha^{(i)}, \hat{\alpha}] + \frac{1}{k(k-1)} \sum_{i<j} \text{JSD}[\alpha^{(i)}, \alpha^{(j)}] \tag{2}$$

---

[6] We make the threshold slightly higher for QA because the output space is larger and thus small dimension-wise perturbations can produce comparatively large TVD.



(a) SST (BiLSTM) (b) SST (CNN) (c) Diabetes (BiLSTM) (d) Diabetes (CNN)



(e) CNN-QA (BiLSTM) (f) bAbI 1 (BiLSTM) (g) SNLI (BiLSTM) (h) SNLI (CNN)

Figure 3: Median change in output ($\Delta\hat{y}^{med}$) (x-axis) densities in relation to the max attention ($\max\hat{\alpha}$) (y-axis) obtained by randomly permuting instance attention weights. Encoders denoted parenthetically. Plots for all corpora and using all encoders are available online.

Algorithm 3 Finding adversarial attention weights

```
$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x})$, $\hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \hat{\alpha})$
$\alpha^{(1)}, ..., \alpha^{(k)} \leftarrow$ Optimize Eq 1
for $i \leftarrow 1$ to $k$ do
    $\hat{y}^{(i)} \leftarrow \text{Dec}(\mathbf{h}, \alpha^{(i)})$        ▷ $\mathbf{h}$ is not changed
    $\Delta\hat{y}^{(i)} \leftarrow \text{TVD}[\hat{y}, \hat{y}^{(i)}]$
    $\Delta\alpha^{(i)} \leftarrow \text{JSD}[\hat{\alpha}, \alpha^{(i)}]$
end for
$\epsilon$-max JSD $\leftarrow \max_i \mathbb{1}[\Delta\hat{y}^{(i)} \leq \epsilon]\Delta\alpha^{(i)}$
```

In practice we maximize a relaxed version of this objective via the Adam SGD optimizer (Kingma and Ba, 2014): $f(\{\alpha^{(i)}\}_{i=1}^{k}) + \frac{\lambda}{k}\sum_{i=1}^{k}\max(0, \text{TVD}[\hat{y}(\mathbf{x}, \alpha^{(i)}), \hat{y}(\mathbf{x}, \hat{\alpha})] - \epsilon)$.[7]

Equation 1 attempts to identify a set of new attention distributions over the input that is as far as possible from the observed α (as measured by JSD) and from each other (and thus diverse), while keeping the output of the model within ϵ of the original prediction. We denote the output obtained under the i^{th} adversarial attention by $\hat{y}^{(i)}$. Note that the JS Divergence between any two categorical distributions (irrespective of length) is bounded from above by 0.69.

One can view an attentive decoder as a function that maps from the space of latent input representations and attention weights over input words Δ^{T-1} to a distribution over the output space $\mathcal{Y}$. Thus, for any output $\hat{y}$, we can define how likely each attention distribution α will generate the output as inversely proportional to $\text{TVD}(y(\alpha), \hat{y})$.

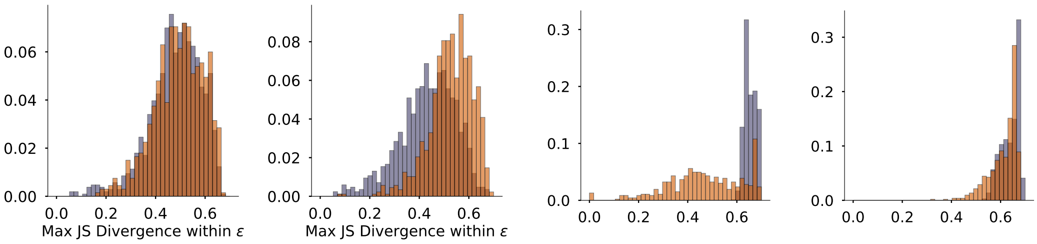
Figure 4 depicts the distributions of max JSDs realized over instances with adversarial attention weights for a subset of the datasets considered. Colors again indicate predicted class. Mass toward the upper-bound of 0.69 indicates that we are frequently able to identify maximally different attention weights that hardly budge model output. We observe that one can identify adversarial attention weights associated with high JSD for a significant number of examples. This means that is often the case that quite different attention distributions over inputs would yield essentially the same (within ϵ) output.

In the case of the *diabetes* task, we again observe a pattern of low JSD for positive examples (where evidence is present) and high JSD for negative examples. In other words, for this task, if one perturbs the attention weights when it is inferred that the patient is diabetic, this does change the output, which is intuitively agreeable. However, this behavior again is an exception to the rule.

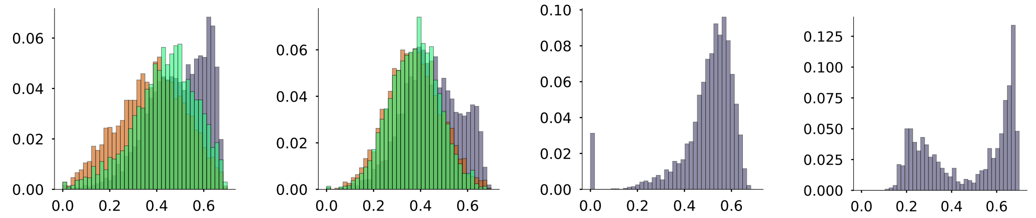
We also consider the relationship between max attention weights (indicating strong emphasis on a particular feature) and the dissimilarity of identified adversarial attention weights, as measured via JSD, for adversaries that yield a prediction within ϵ of the original model output. Intuitively, one might hope that if attention weights are peaky, then counterfactual attention weights that are very different but which yield equivalent predictions

---

[7] We set $\lambda = 500$.



(a) SST (BiLSTM) (b) SST (CNN) (c) Diabetes (BiLSTM) (d) Diabetes (CNN)



(e) SNLI (BiLSTM) (f) SNLI (CNN) (g) CNN-QA (BiLSTM) (h) BAbI 1 (BiLSTM)

Figure 4: Histogram of maximum adversarial JS Divergence (ϵ-max JSD) between original and adversarial attentions over all instances. In all cases shown, $|\hat{y}^{adv} - \hat{y}| < \epsilon$. Encoders are specified in parantheses. Best viewed in color.

would be more difficult to identify.

Figure 5 illustrates that while there is a negative trend to this effect, it is realized only weakly. Put another way: there exist many cases (in all datasets) in which despite a high attention weight, an alternative and quite different attention configuration over inputs yields effectively the same output. In light of this, presenting a heatmap implying that a particular set of features is primarily responsible for an output would seem to be misleading.

5 Related Work

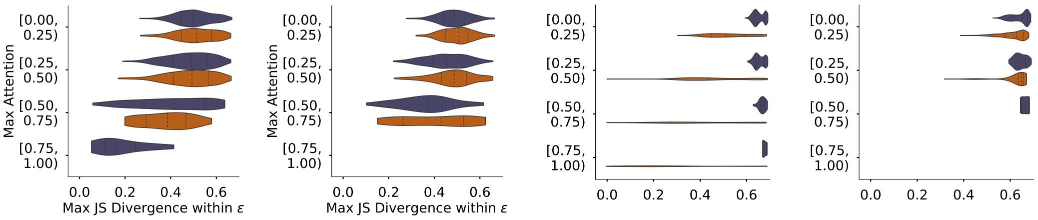
We have focused on attention mechanisms and the question of whether they afford transparency, but a number of interesting strategies unrelated to attention mechanisms have been recently proposed to provide insights into neural NLP models. These include approaches that measure feature importance based on gradient information (Ross et al., 2017; Sundararajan et al., 2017) (aligned with the gradient-based measures that we have used here), and methods based on *representation erasure* (Li et al., 2016), in which dimensions are removed and then the resultant change in output is recorded (similar to our experiments with removing tokens from inputs, albeit we do this at the input layer).

Comparing such importance measures to attention scores may provide additional insights into the working of attention based models (Ghaeini et al., 2018). Another novel line of work in this direction involves explicitly identifying explanations of black-box predictions via a causal framework (Alvarez-Melis and Jaakkola, 2017). We also note that there has been complementary work demonstrating correlation between human attention and induced attention weights, which was relatively strong when humans agreed on an explanation (Pappas and Popescu-Belis, 2016). It would be interesting to explore if such cases present explicit 'high precision' signals in the text (for example, the positive label in diabetes dataset).

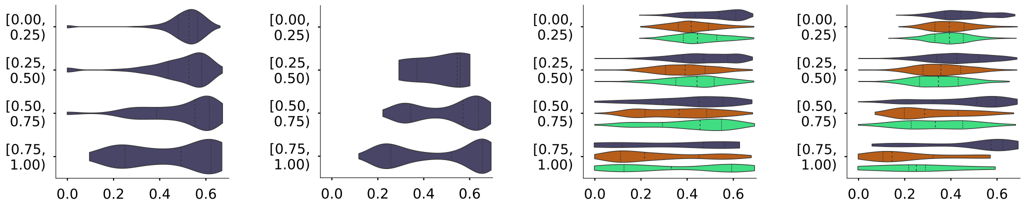
More specific to attention mechanisms, recent promising work has proposed more principled attention variants designed explicitly for interpretability; these may provide greater transparency by imposing *hard*, *sparse* attention. Such instantiations explicitly select (modest) subsets of inputs to be considered when making a prediction, which are then by construction responsible for model output (Lei et al., 2016; Peters et al., 2018). *Structured attention* models (Kim et al., 2017) provide a generalized framework for describing and fitting attention variants with explicit probabilistic semantics. Tying attention weights to human-provided rationales is another potentially promising avenue (Bao et al., 2018). We hope our work motivates further development of these methods, resulting in attention variants that both improve predictive performance and provide insights into model predictions.

6 Discussion and Conclusions

We have provided evidence that correlation between intuitive feature importance measures (in-



(a) SST (BiLSTM) (b) SST (CNN) (c) Diabetes (BiLSTM) (d) Diabetes (CNN)



(e) CNN-QA (BiLSTM) (f) bAbI 1 (BiLSTM) (g) SNLI (BiLSTM) (h) SNLI (CNN)

Figure 5: Densities of maximum JS divergences (ϵ-max JSD) (x-axis) as a function of the max attention (y-axis) in each instance for obtained between original and adversarial attention weights.

cluding gradient and feature erasure approaches) and learned attention weights is weak when using a BiRNN encoder (Section 4.1). We also established that counterfactual attention distributions — which would tell a different story about why a model made the prediction that it did — often have no effect on model output (Section 4.2).

These results suggest that while attention modules consistently yield improved performance on NLP tasks, their ability to provide transparency for model predictions is (in the sense of pointing to inputs responsible for outputs) questionable. More generally, how one is meant to interpret the 'heatmaps' of attention weights placed over inputs that are commonly presented is unclear. These seem to suggest a story about how a model arrived at a particular disposition, but the results here indicate that the relationship between this and attention is not obvious, at least for RNN encoders.

There are important **limitations** to this work and the conclusions we can draw from it. We have reported the (generally weak) correlation between learned attention weights and various alternative measures of feature importance, e.g., gradients. We do not imply that such alternative measures are necessarily ideal or should be considered 'ground truth'. While such measures do enjoy a clear intrinsic (to the model) semantics, their interpretation for non-linear neural networks can nonetheless be difficult for humans (Feng et al., 2018). Still, that attention consistently correlates poorly with *multiple* such measures ought to give pause to practitioners. That said, exactly how strong such correlations 'should' be to establish reliability as explanation is an admittedly subjective question.

We note that the counterfactual attention experiments demonstrate the existence of alternative heatmaps that yield equivalent predictions; thus one cannot conclude that the model made a particular prediction *because* it attended over inputs in a specific way. But these adversarial weights may themselves be unlikely under the attention module parameters. Further, it may be that multiple plausible explanations exist, complicating interpretation. We would maintain that in such cases the model should highlight all plausible explanations, but one may instead view a model that provides 'sufficient' explanation as reasonable.

An additional limitation is that we have only considered a handful of attention variants, selected to reflect common module architectures for the respective tasks included in our analysis. Alternative attention specifications may yield different conclusions; and indeed we hope this work motivates further development of principled attention mechanisms (or encoders). Finally, we have limited our evaluation to tasks with unstructured output spaces, i.e., we have not considered seq2seq tasks, which we leave for future work. However we believe interpretability is more often a consideration in, e.g., classification than in translation.

Acknowledgements

We thank Zachary Lipton for insightful feedback on a preliminary version of this manuscript.

This work was supported by the Army Research Office (ARO), award W911NF1810328.

References

David Alvarez-Melis and Tommi S Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. *arXiv preprint arXiv:1808.09367*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512.

Shi Feng, Eric Wallace, Alvin Grissom II, Pedro Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretation difficult. In *Empirical Methods in Natural Language Processing*.

Reza Ghaeini, Xiaoli Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4952–4957.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Tao Lei et al. 2017. *Interpretable neural models for natural language processing*. Ph.D. thesis, Massachusetts Institute of Technology.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Azadeh Nikfarjam, Abeed Sarker, Karen OâĂŹConnor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681.

Nikolaos Pappas and Andrei Popescu-Belis. 2016. Human versus machine attention in document classification: A dataset with crowdsourced annotations. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 94–100.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Ben Peters, Vlad Niculae, and André FT Martins. 2018. Interpretable structure induction via sparse attention. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 365–367.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on* *empirical methods in natural language processing*, pages 1631–1642.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. *arXiv preprint arXiv:1704.05908*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Ye Zhang, Iain Marshall, and Byron C Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 2016, pages 795–804.

Appendices

A Model details

For all datasets, we use spaCy for tokenization. We map out of vocabulary words to a special `<unk>` token and map all words with numeric characters to 'qqq'. Each word in the vocabulary was initialized to pretrained embeddings. For general domain corpora we used either (i) FastText Embeddings (SST, IMDB, 20News, and CNN) trained on Simple English Wikipedia, or, (ii) GloVe 840B embeddings (AGNews and SNLI). For the MIMIC dataset, we learned word embeddings using Gensim over all discharge summaries in the corpus. We initialize words not present in the vocabulary using samples from a standard Gaussian $\mathcal{N}(\mu = 0, \sigma^2 = 1)$.

A.1 BiLSTM

We use an embedding size of 300 and hidden size of 128 for all datasets except bAbI (for which we use 50 and 30, respectively). All models were regularized using ℓ_2 regularization ($\lambda = 10^{-5}$) applied to all parameters. We use a sigmoid activation functions for binary classification tasks, and a softmax for all other outputs. We trained the model using maximum likelihood loss using the Adam Optimizer with default parameters in PyTorch.

A.2 CNN

We use an embedding size of 300 and 4 kernels of sizes [1, 3, 5, 7], each with 64 filters, giving a final hidden size of 256 (for bAbI we use 50 and 8 respectively with same kernel sizes). We use ReLU activation function on the output of the filters. All other configurations remain same as BiLSTM.

A.3 Average

We use the embedding size of 300 and a projection size of 256 with ReLU activation on the output of the projection matrix. All other configurations remain same as BiLSTM.

B Further details regarding attentional module of gradient

In the gradient experiments, we made the decision to cut-off the computation graph at the attention module so that gradient does not flow through this layer and contribute to the gradient feature importance score. For the sake of gradient calculation this effectively treats the attention as a separate input to the network, independent of the input. We argue that this is a natural choice to make for our analysis because it calculates: *how much does the output change as we perturb particular inputs (words) by a small amount, while paying the same amount of attention to said word as originally estimated and shown in the heatmap*?

C Correlations between Feature Importance measures

A question one might have here is how well correlated LOO and gradients are with *one another*. We report such results in their entirety on the paper website, and we summarize their correlations relative to those realized by attention in a BiLSTM model with LOO measures in Figure 6. This reports the mean differences between (i) gradient

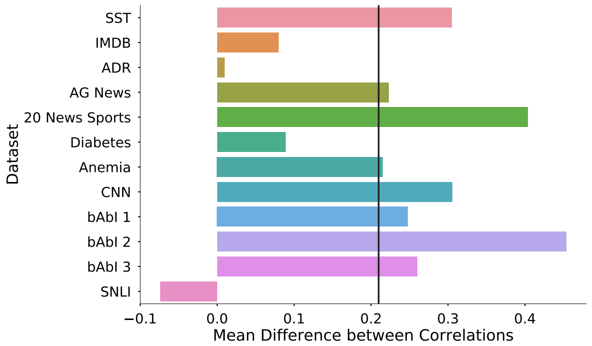


Figure 6: Mean difference in correlation of (i) LOO vs. Gradients and (ii) Attention vs. LOO scores using BiLSTM Encoder + Tanh Attention. On average the former is more correlated than the latter by >0.2 τ_{loo}.

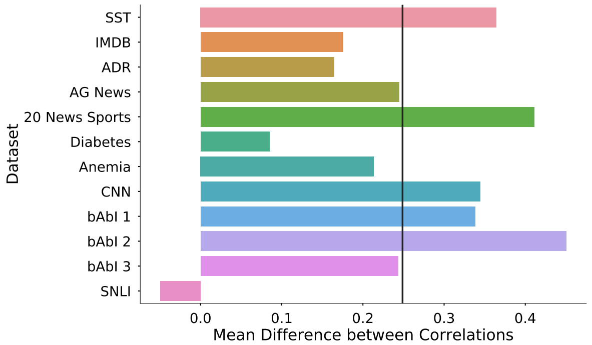


Figure 7: Mean difference in correlation of (i) LOO vs. Gradients and (ii) Attention vs. Gradients using BiLSTM Encoder + Tanh Attention. On average the former is more correlated than the latter by ~0.25 τ_g.

and LOO correlations, and (ii) attention and LOO correlations. As expected, we find that these exhibit, in general, considerably higher correlation with one another (on average) than LOO does with attention scores. (The lone exception is on SNLI.) Figure 7 shows the same for gradients and attention scores; the differences are comparable. In the *ADR* and *Diabetes* corpora, a few high precision tokens indicate (the positive) class, and in these cases we see better agreement between LOO/gradient measures with attention; this is consistent with Figure 4 which shows that it is difficult for the BiLSTM variant to find adversarial attention distributions for *Diabetes*.

A potential issues with using Kendall τ as our metric here is that (potentially many) irrelevant features may add noise to the correlation measures. We acknowledge that this as a shortcoming of the metric. One observation that may mitigate this concern is that we might expect such noise to depress the LOO and gradient correlations to the same extent as they do the correlation between attention and feature importance scores; but as per Figure 7, they do not. We also note that the correlations between the attention weights on top of feedforward (projection) encoder and LOO scores are much stronger, on average, than those between BiLSTM attention weights and LOO. This is shown in Figure 8. Were low correlations due simply to noise, we would not expect this.[8]

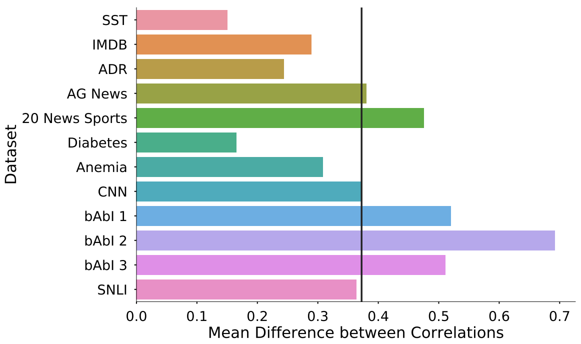


Figure 8: Difference in mean correlation of attention weights vs. LOO importance measures for (i) Average (feed-forward projection) and (ii) BiLSTM Encoders with Tanh attention. Average correlation (vertical bar) is on average ~0.375 points higher for the simple feedforward encoder, indicating greater correspondence with the LOO measure.

D Graphs

To provide easy navigation of our (large set of) graphs depicting attention weights on various datasets/tasks under various model configuration we have created an interactive interface to browse these results, accessible at: `https://successar.github.io/AttentionExplanation/docs/`.

E Adversarial Heatmaps

SST

Original: reggio falls victim to relying on the very digital technology that he fervently scorns creating a meandering inarticulate and ultimately disappointing film

Adversarial: reggio falls victim to relying on the very digital technology that he fervently scorns creating a meandering inarticulate and ultimately disappointing film $\Delta\hat{y}$: *0.005*

IMDB

---

[8] The same contrast can be seen for the gradients, as one would expect given the direct gradient paths in the projection network back to individual tokens.

Original: fantastic movie one of the best film noir movies ever made bad guys bad girls a jewel heist a twisted morality a kidnapping everything is here jean has a face that would make bogart proud and the rest of the cast is is full of character actors who seem to to know they're onto something good get some popcorn and have a great time

Adversarial: fantastic movie one of the best film noir movies ever made bad guys bad girls a jewel heist a twisted morality a kidnapping everything is here jean has a face that would make bogart proud and the rest of the cast is is full of character actors who seem to to know they're onto something good get some popcorn and have a great time $\Delta\hat{y}$: *0.004*

20 News Group - Sports

Original:i meant to comment on this at the time there ' s just no way baserunning could be that important if it was runs created would n ' t be nearly as accurate as it is runs created is usually about qqq qqq accurate on a team level and there ' s a lot more than baserunning that has to account for the remaining percent .

Adversarial:i meant to comment on this at the time there ' s just no way baserunning could be that important if it was runs created would n ' t be nearly as accurate as it is runs created is usually about qqq qqq accurate on a team level and there ' s a lot more than baserunning that has to account for the remaining percent . $\Delta\hat{y}$: *0.001*

ADR

Original:meanwhile wait for DRUG and DRUG to kick in first co i need to prep dog food etc . co omg <UNK> .

Adversarial:meanwhile wait for DRUG and DRUG to kick in first co i need to prep dog food etc . co omg <UNK> . $\Delta\hat{y}$: *0.002*

AG News

Original:general motors and daimlerchrysler say they # qqq teaming up to develop hybrid technology for use in their vehicles . the two giant automakers say they have signed a memorandum of understanding

Adversarial:general motors and daimlerchrysler say they # qqq teaming up to develop hybrid technology for use in their vehicles . the two giant automakers say they have signed a memorandum of understanding . $\Delta\hat{y}$: *0.006*

SNLI

Hypothesis:a man is running on foot

Original Premise Attention:a man in a gray shirt and blue shorts is standing outside of an old fashioned ice cream shop named sara 's old fashioned ice cream , holding his bike up , with a wood like table , chairs , benches in front of him .

Adversarial Premise Attention:a man in a gray shirt and blue shorts is standing outside of an old fashioned ice cream shop named sara 's old fashioned ice cream , holding his bike up , with a wood like table , chairs , benches in front of him . $\Delta\hat{y}$: *0.002*

Babi Task 1

Question: Where is Sandra ?

Original Attention:John travelled to the garden . Sandra travelled to the garden

Adversarial Attention:John travelled to the garden . Sandra travelled to the garden $\Delta\hat{y}$: *0.003*

CNN-QA

Question:federal education minister @placeholder visited a @entity15 store in @entity17 , saw cameras

Original:@entity1 , @entity2 (@entity3) police have arrested four employees of a popular @entity2 ethnic - wear chain after a minister spotted a security camera overlooking the changing room of one of its stores . federal education minister @entity13 was visiting a @entity15 outlet in the tourist resort state of @entity17 on friday when she discovered a surveillance camera pointed at the changing room , police said . four employees of the store have been arrested , but its manager – herself a woman – was still at large saturday , said @entity17 police superintendent @entity25 . state authorities launched their investigation right after @entity13 levied her accusation . they found an overhead camera that the minister had spotted and determined that it was indeed able to take photos of customers using the store 's changing room , according to @entity25 . after the incident , authorities sealed off the store and summoned six top officials from @entity15 , he said . the arrested staff have been charged with voyeurism and breach of privacy , according to the police . if convicted , they could spend up to three years in jail , @entity25 said . officials from @entity15 – which sells ethnic garments , fabrics and other products – are heading to @entity17 to work with investigators , according to the company . " @entity15 is deeply concerned and shocked at this allegation , " the company said in a statement . " we are in the process of investigating this internally and will be cooperating fully with the police . "

Adversarial:@entity1 , @entity2 (@entity3) police have arrested four employees of a popular @entity2 ethnic - wear chain after a minister spotted a security camera overlooking the changing room of one of its stores . federal education minister @entity13 was visiting a @entity15 outlet in the tourist resort state of @entity17 on friday when she discovered a surveillance camera pointed at the changing room , police said . four employees of the store have been arrested , but its manager – herself a woman – was still at large saturday , said @entity17 police superintendent @entity25 . state authorities launched their investigation right after @entity13 levied her accusation . they found an overhead camera that the minister had spotted and determined that it was indeed able to take photos of customers using the store 's changing room , according to @entity25 . after the incident , authorities sealed off the store and summoned six top officials from @entity15 , he said . the arrested staff have been charged with voyeurism and breach of privacy , according to the police . if convicted , they could spend up to three years in jail , @entity25 said . officials from @entity15 – which sells ethnic garments , fabrics and other products – are heading to @entity17 to work with investigators , according to the company . " @entity15 is deeply concerned and shocked at this allegation , " the company said in a statement . " we are in the process of investigating this internally and will be cooperating fully with the police . " $\Delta\hat{y}$: *0.005*

Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning

Prithviraj Ammanabrolu
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
raj.ammanabrolu@gatech.edu

Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
riedl@cc.gatech.edu

Abstract

Text-based adventure games provide a platform on which to explore reinforcement learning in the context of a combinatorial action space, such as natural language. We present a deep reinforcement learning architecture that represents the game state as a knowledge graph which is learned during exploration. This graph is used to prune the action space, enabling more efficient exploration. The question of which action to take can be reduced to a question-answering task, a form of transfer learning that pre-trains certain parts of our architecture. In experiments using the TextWorld framework, we show that our proposed technique can learn a control policy faster than baseline alternatives. We have also open-sourced our code at https://github.com/rajammanabrolu/KG-DQN.

1 Introduction

Natural language communication can be used to affect change in the real world. Text adventure games, in which players must make sense of the world through text descriptions and declare actions through natural language, can provide a stepping stone toward more real-world environments where agents must communicate to understand the state of the world and indirectly affect change in the world. Text adventure games are also useful for developing and testing reinforcement learning algorithms that must deal with the partial observability of the world (Narasimhan et al., 2015; He et al., 2016).

In text adventure games, the agent receives an incomplete textual description of the current state of the world. From this information, and previous interactions with the world, a player must determine the next best action to take to achieve some quest or goal. The player must then compose a textual description of the action they intend to make and receive textual feedback of the effects of the action. Formally, a text-based game is a partially observable Markov decision process (POMDP), represented as a 7-tuple of $\langle S, T, A, \Omega, O, R, \gamma \rangle$ representing the set of environment states, conditional transition probabilities between states, words used to compose text commands, observations, observation conditional probabilities, reward function, and the discount factor respectively (Côté et al., 2018).

In text-based games, the agent never has access to the true underlying world state and has to reason about how to act in the world based only on the textual observations. Additionally, the agent's actions must be expressed through natural language commands, ensuring that the action space is combinatorially large. Thus, text-based games pose a different set of challenges than traditional video games. Text-based games require a greater understanding of previous context to be able to explore the state-action space more effectively. Such games have historically proven to be difficult to play for AI agents, and the more complex variants such as *Zork* still remain firmly out of the reach of existing approaches.

We introduce three contributions to text-based game playing to deal with the combinatorially large state and action spaces. First, we show that a state representation in the form of a *knowledge graph* gives us the ability to effectively prune an action space. A knowledge graph captures the relationships between entities as a directed graph. The knowledge graph provides a persistent memory of the world over time and enables the agent to have a prior notion of what actions it should not take at a particular stage of the game.

Our second contribution is a deep reinforcement learning architecture, *Knowledge Graph DQN* (KG-DQN), that effectively uses this state rep-

Proceedings of NAACL-HLT 2019, pages 3557–3565
Minneapolis, Minnesota, June 2 - June 7, 2019.

resentation to estimate the Q-value for a state-action pair. This architecture leverages recent advances in graph embedding and attention techniques (Guan et al., 2018; Veličković et al., 2018) to learn which portions of the graph to pay attention to given an input state description in addition to having a mechanism that allows for natural language action inputs. Finally, we take initial steps toward framing the POMDP as a question-answering (QA) problem wherein a knowledge-graph can be used to not only prune actions but to answer the question of what action is most appropriate. Previous work has shown that many NLP tasks can be framed as instances of question-answering and that we can transfer knowledge between these tasks (McCann et al., 2017). We show how pre-training certain parts of our KG-DQN network using existing QA methods improves performance and allows knowledge to be transferred from different games.

We provide results on ablative experiments comparing our knowledge-graph based approach approaches to strong baselines. Results show that incorporating a knowledge-graph into a reinforcement learning agent results in converges to the highest reward more than 40% faster than the best baseline. With pre-training using a question-answering paradigm, we achieve this fast convergence rate while also achieving high quality quest solutions as measured by the number of steps required to complete the quests.

2 Related Work

A growing body of research has explored the challenges associated with text-based games (Bordes et al., 2010; Narasimhan et al., 2015; He et al., 2016; Fulda et al., 2017; Haroush et al., 2018; Côté et al., 2018; Tao et al., 2018). Narasimhan et al. (2015) attempts to solve parser-based text games by encoding the observations using an LSTM. This encoding vector is then used by an action scoring network that determines the scores for the action verb and each of the corresponding argument objects. The two scores are then averaged to determine Q-value for the state-action pair. He et al. (2016) present the Deep Reinforcement Relevance Network (DRRN) which uses two separate deep neural networks to encode the state and actions. The Q-value for a state-action pair is then computed by a pairwise interaction function between the two encoded representations. Both of these methods are not conditioned on previous observations and so are at a disadvantage when dealing with complex partially observable games. Additionally, neither of these approaches prune the action space and so end up wasting trials exploring state-action pairs that are likely to have low Q-values, likely leading to slower convergence times for combinatorially large action spaces.

Haroush et al. (2018) introduce the Action Eliminating Network (AEN) that attempts to restrict the actions in each state to the top-k most likely ones, using the emulator's feedback. The network learns which actions should not be taken given a particular state. Their work shows that reducing the size of the action space allows for more effective exploration, leading to better performance. Their network is also not conditioned on previous observations.

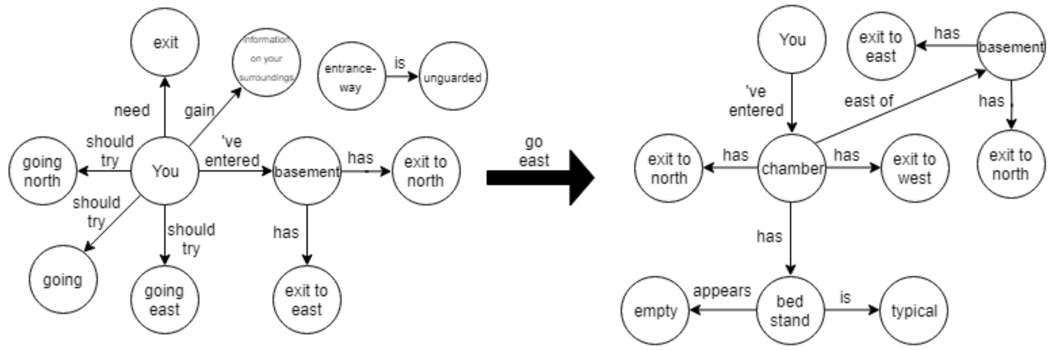
Knowledge graphs have been demonstrated to improve natural language understanding in other domains outside of text adventure games. For example, Guan et al. (2018) use commonsense knowledge graphs such as *ConceptNet* (Speer and Havasi, 2012) to significantly improve the ability of neural networks to predict the end of a story. They represent the graph in terms of a knowledge context vector using features from *ConceptNet* and graph attention (Veličković et al., 2018). The state representation that we have chosen as well as our method of action pruning builds on the strengths of existing approaches while simultaneously avoiding the shortcomings of ineffective exploration and lack of long-term context.

3 Knowledge Graph DQN

In this section we introduce our knowledge graph representation, action pruning and deep Q-network architecture.

3.1 Knowledge Graph Representation

In our approach, our agent learns a knowledge graph, stored as a set of RDF triples, i.e. 3-tuples of $\langle subject, relation, object\rangle$. These triples are extracted from the observations using Stanford's Open Information Extraction (OpenIE) (Angeli et al., 2015). OpenIE is not optimized to the regularities of text adventure games and there are a lot of relations that can be inferred from the typical structure of descriptive texts. For example, from a phrase such as "There is an exit to the north" one can infer a *has* relation between the current



You've entered a basement. You try to gain information on your surroundings by using a technique you call "looking." You need an unguarded exit? You should try going east. You don't like doors? Why not try going north, that entranceway is unguarded.

You've entered a chamber. You can see a bed stand. The bed stand is typical. The bed stand appears to be empty. There is an exit to the north. Don't worry, it is unblocked. There is an unblocked exit to the west.

Figure 1: Graph state update example given two observations

location and the direction of the exit. These additional rules fill in the information not provided by OpenIE. The resultant knowledge graph gives the agent what essentially amounts to a mental map of the game world.

The knowledge graph is updated after every agent action (see Figure 1). The update rules are defined such that there are portions of the graph offering short and long-term context. A special node—designated "you"—represents the agent and relations out of this node are updated after every action with the exception of relations denoting the agent's inventory. Other relations persist after each action. We intend for the update rules to be applied to text-based games in different domains and so only hand-craft a minimal set of rules that we believe apply generally. They are:

- Linking the current room type (e.g. "basement", "chamber') to the items found in the room with the relation "has", e.g. $\langle chamber, has, bed\ stand\rangle$
- Extracting information regarding entrances and exits and linking them to the current room, e.g. $\langle basement, has, exit\ to\ north\rangle$
- Removing all relations relating to the "you" node with the exception of inventory every action, e.g. $\langle you, have, cubical\ key\rangle$
- Linking rooms with directions based on the action taken to move between the rooms, e.g. $\langle chamber, east\ of, basement\rangle$ after the action "go east" is taken to go from the basement to the chamber

All other RDF triples generated are taken from OpenIE.

3.2 Action Pruning

The number of actions available to an agent in a text adventure game can be quite large: $A = \mathcal{O}(|V| \times |O|^2)$ where V is the number of action verbs, and O is the number of distinct objects in the world that the agent can interact with, assuming that verbs can take two arguments. Some actions, such as movement, inspecting inventory, or observing the room, do not have arguments.

The knowledge graph is used to prune the combinatorially large space of possible actions available to the agent as follows. Given the current state graph representation G_t, the action space is pruned by ranking the full set of actions and selecting the top-k. Our action scoring function is:

- +1 for each object in the action that is present in the graph; and
- +1 if there exists a valid directed path between the two objects in the graph.

We assume that each action has at most two objects (for example inserting a key in a lock).

3.3 Model Architecture and Training

Following Narasimhan et al. (2015), all actions A that will be accepted by the game's parser are available to the agent at all times. When playing the game, the agent chooses an action and receives an observation o_t from the simulator, which is a textual description of current game state. The state graph G_t is updated according to the given observation, as described in Section 3.1.

We use the Q-Learning technique (Watkins and Dayan, 1992) to learn a control policy $\pi(a_t|s_t)$, $a_t \in A$, which gives us the probability of taking action a_t given the current state s_t. The policy is determined by the Q-value of a particular state-action pair, which is updated using the Bellman equation (Sutton and Barto, 2018):

$$Q_{t+1}(s_{t+1},a_{t+1}) = E[r_{t+1} + \gamma \max_{a \in A_t} Q_t(s,a) | s_t, a_t] \quad (1)$$

where γ refers to the discount factor and r_{t+1} is the observed reward. The policy is thus to take the action that maximizes the Q-value in a particular state, which will correspond to the action that maximizes the reward expectation given that the agent has taken action a_t at the current state s_t and followed the policy $\pi(a|s)$ after.

The architecture in Figure 2 is responsible for computing the representations for both the state s_t and the actions $a^{(i)} \in A$ and coming to an estimation of the Q-value for a particular state and action. During the forward activation, the agent uses the observation to update the graph G_t using the rules outlined in Section 3.2.

The graph is then embedded into a single vector $\mathbf{g_t}$. We use Graph Attention (Veličković et al., 2018) with an attention mechanism similar to that described in Bahdanau et al. (2014). Formally, the Multi-headed Graph Attention component receives a set of node features $H = \{\mathbf{h_1}, \mathbf{h_2}, \dots, \mathbf{h_N}\}$, $\mathbf{h_i} \in \mathrm{I\!R}^{\mathrm{F}}$, where N is the number of nodes and F the number of features in each node, and the adjacency matrix of G_t. Each of the node features consist of the averaged word embeddings for the tokens in that node, as determined by the preceding graph embedding layer. The attention mechanism is set up using self-attention on the nodes after a learnable linear transformation $W \in \mathrm{I\!R}^{2\mathrm{F}\times\mathrm{F}}$ applied to all the node features:

$$e_{ij} = LeakyReLU(\mathbf{p} \cdot W(\mathbf{h_i} \oplus \mathbf{h_j})) \quad (2)$$

where $\mathbf{p} \in \mathrm{I\!R}^{2\mathrm{F}}$ is a learnable parameter. The attention coefficients α_{ij} are then computed by normalizing over the choices of $k \in \mathcal{N}$ using the softmax function. Here $\mathcal{N}$ refers to the neighborhood in which we compute the attention coefficients. This is determined by the adjacency matrix for G_t and consists of all third-order neighbors of a particular node.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}} exp(e_{ik})} \quad (3)$$

Multi-head attention is then used, calculating multiple independent attention coefficients. The resulting features are then concatenated and passed into a linear layer to determine $\mathbf{g_t}$:

$$\mathbf{g_t} = f(W_g(\|_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j)) + b_g) \quad (4)$$

where k refers to the parameters of the k^{th} independent attention mechanism, W_g and b_g the weights and biases of this component's output linear layer, and $\|$ represents concatenation.

Simultaneously, an encoded representation of the observation $\mathbf{o_t}$ is computed using a Sliding Bidirectional LSTM (SB-LSTM). The final state representation $\mathbf{s_t}$ is computed as:

$$\mathbf{s_t} = f(W_l(\mathbf{g_t} \oplus \mathbf{o_t}) + b_l) \quad (5)$$

where W_l, b_l represent the final linear layer's weights and biases and $\mathbf{o_t}$ is the result of encoding the observation with the SB-LSTM.

The entire set of possible actions A is pruned by scoring each $a \in A$ according to the mechanism previously described using the newly updated G_{t+1}. We then embed and encode all of these action strings using an LSTM encoder (Sutskever et al., 2014). The dashed lines in Figure 2 denotes non-differentiable processes.

The final Q-value for a state-action pair is:

$$Q(\mathbf{s_t}, \mathbf{a_t}) = \mathbf{s_t} \cdot \mathbf{a_t} \quad (6)$$

This method of separately computing the representations for the state and action is similar to the approach taken in the DRRN (He et al., 2016).

We train the network using experience replay (Lin, 1993) with prioritized sampling (cf., (Moore and Atkeson, 1993)) and a modified version of the ϵ-greedy algorithm (Sutton and Barto, 2018) that we call the ϵ_1, ϵ_2-greedy learning algorithm. The experience replay strategy finds paths in the game, which are then stored as transition tuples in a experience replay buffer D. The ϵ_1, ϵ_2-greedy algorithm explores by choosing actions randomly from A with probability ϵ_1 and from A_t with a probability ϵ_2. The second threshold is needed to account for situations where an action must be chosen to advance the quest for which the agent has no prior in G_t. That is, action pruning may remove actions essential to quest completion because those actions involve combinations of entities that have not been encountered before.

We then sample a mini-batch of transition tuples consisting of $\langle \mathbf{s_k}, \mathbf{a_k}, r_{k+1}, \mathbf{s_{k+1}}, \mathbf{A_{k+1}}, p_k \rangle$ from

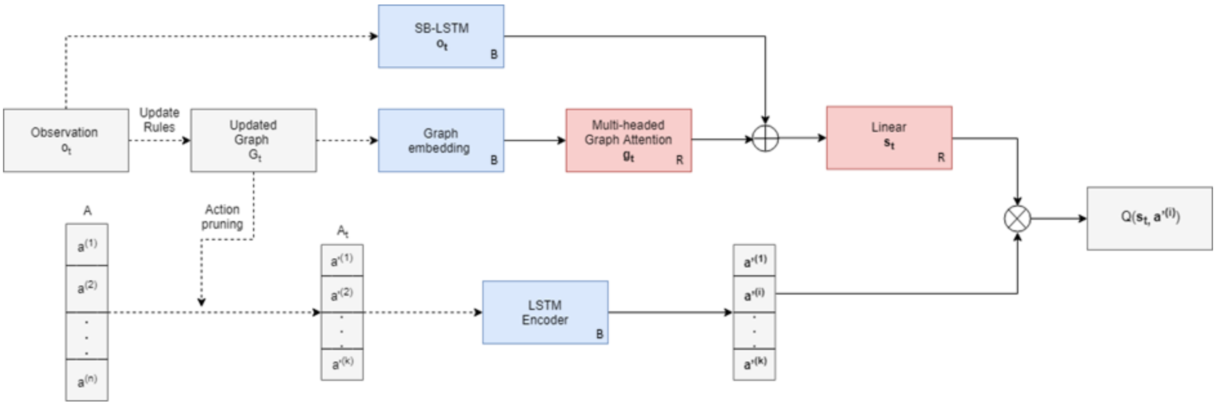


Figure 2: KG-DQN architecture, blue shading (or the symbol 'B') indicates components that can be pre-trained and red (or the symbol 'R') indicates no pre-training. The solid lines indicate gradient flow for learnable components.

D and compute the temporal difference loss as:

$$\begin{aligned} L(\theta) =& r_{k+1} + \\ & \gamma \max_{\mathbf{a} \in \mathbf{A_{k+1}}} Q(\mathbf{s_t}, \mathbf{a}; \theta) - Q(\mathbf{s_t}, \mathbf{a_t}; \theta) \end{aligned} \quad (7)$$

Replay sampling from D is done by sampling a fraction ρ from transition tuples with a positive reward and $1 - \rho$ from the rest. As shown in (Narasimhan et al., 2015), prioritized sampling from experiences with a positive reward helps the deep Q-network more easily find the sparse set of transitions that advance the game. The exact training mechanism is described in Algorithm 1.

4 Game Play as Question Answering

Previous work has shown that many NLP tasks can be framed as instances of question-answering and that in doing so, one can transfer knowledge between these tasks (McCann et al., 2017). In the abstract, an agent playing a text adventure game can be thought of as continuously asking the question "What is the right action to perform in this situation?" When appropriately trained, the agent may be able to answer the question for itself and select a good next move to execute. Treating the problem as question-answering will not replace the need for exploration in text-adventure games. However, we hypothesize that it will cut down on the amount of exploration needed during testing time, theoretically allowing it to complete quests faster; one of the challenges of text adventure games is that the quests are puzzles and even after training, execution of the policy requires a significant amount of exploration.

To teach the agent to answer the question of what action is best to take given an observation, we use an offline, pre-training approach. The data for the pre-training approach is generated using an oracle, an agent capable of finishing a game perfectly in the least number of steps possible. Specifically, the agent knows exactly what action to take given the state observation in order to advance the game in the most optimal manner possible. Through this process, we generate a set of traces consisting of state observations and actions such that the state observation provides the context for the implicit question of "What action should be taken?" and the oracle's correct action is the answer. We then use the DrQA (Chen et al., 2017) question-answering technique to train a paired question encoder and an answer encoder that together predict the answer (action) from the question (text observation). The weights from the SB-LSTM in the document encoder in the DrQA system are then used to initialize the weights of the SB-LSTM. Similarly, embedding layers of both the graph and the LSTM action encoder are initialized with the weights from the embedding layer of same document encoder. Since the DrQA embedding layers are initialized with GloVe, we are transferring word embeddings that are tuned during the training of the QA architecture.

The game traces used to train the question-answering come from a set of games of the same domain but have different specific configurations of the environment and different quests. We use the TextWorld framework (Côté et al., 2018), which uses a grammar to generate random worlds and quests. The types of rooms are the same, but their relative spatial configuration, the types of objects, and the specific sequence of actions needed to complete the quest are different each time. This

	Small	Large
Rooms	10	20
Total objects	20	40
Quest length	5	10
Branching factor	143	562
Vocab size	746	819
Average words per obs.	67.5	94.0
Average new RDF triples per obs.	7.2	10.5

Table 1: Generated game details.

means that the agent cannot simply memorize quests. For pre-training to work, the agent must develop a general question-answering competence that can transfer to new quests. Our approach to question-answering in the context of text adventure game playing thus represents a form of transfer learning.

5 Experiments

We conducted experiments in the TextWorld framework (Côté et al., 2018) using their "home" theme. TextWorld uses a grammar to randomly generate game worlds and quests with given parameters. Games generated with TextWorld start with a zero-th observation that gives instructions for the quest; we do not allow our agent to access this information. The TextWorld API also provides a list of *admissible* actions at each state—the actions that can be performed based on the objects that are present. We do not allow our agent to access the admissible actions.

We generated two sets of games with different random seeds, representing different game difficulties, which we denote as *small* and *large*. Small games have ten rooms and quests of length five and large games have twenty rooms and quests of length ten. Statistics on the games are given in Table 1. Quest length refers to the number of actions that the agent is required to perform in order to finish the quest; more actions are typically necessary to move around the environment and find the objects that need to be interacted with. The branching factor is the size of the action set A for that particular game.

The reward function provided by TextWorld is as follows: +1 for each action taken that moves the agent closer to finishing the quest; -1 for each action taken that extends the minimum number of steps needed to finish the quest from the current stage; 0 for all other situations. The maximum achievable reward for the small and large sets of games are 5 and 10 respectively. This allows for a large amount of variance in quest quality—as measured by steps to complete the quest—that receives maximum reward.

	EM	Precision	Recall	F1
Small	46.20	56.57	63.38	57.94
Large	34.13	52.53	64.72	55.06

Table 2: Pre-training accuracy.

The following procedure for pre-training was done separately for each set of games. Pre-training of the SB-LSTM within the question-answering architecture is conducted by generating 200 games from the same TextWorld theme. The QA system was then trained on data from walkthroughs of a randomly-chosen subset of 160 of these generated games, tuned on a dev set of 20 games, and evaluated on the held-out set of 20 games. Table 2 provides details on the Exact Match (EM), precision, recall, and F1 scores of the QA system after training for the small and large sets of games. Precision, recall, and F1 scores are calculated by counting the number of tokens between the predicted answer and ground truth. An Exact Match is when the entire predicted answer matches with the ground truth. This score is used to tune the model based on the dev set of games.

A random game was chosen from the test-set of games and used as the environment for the agent to train its deep Q-network on. Thus, at no time did the QA system see the final testing game prior to the training of the KG-DQN network.

We compare our technique to three baselines:

- Random command, which samples from the list of admissible actions returned by the TextWorld simulator at each step.
- LSTM-DQN, developed by Narasimhan et al. (2015).
- Bag-of-Words DQN, which uses a bag-of-words encoding with a multi-layer feed forward network instead of an LSTM.

To achieve the most competitive baselines, we used a randomized grid search to choose the best hyperparameters (e.g., hidden state size, γ, ρ, final ϵ, update frequency, learning rate, replay buffer size) for the BOW-DQN and LSTM-DQN baselines.

We tested three versions of our KG-DQN:

1. Un-pruned actions with pre-training

Algorithm 1 ϵ_1, ϵ_2-greedy learning algorithm for KG-DQN

```
for episode=1 to $M$ do
    Initialize action dictionary $A$ and graph $G_0$
    Reset the game simulator
    Read initial observation $o_1$
    $G_1 \leftarrow updateGraph(G_0, o_1); A_1 \leftarrow pruneActions(A, G_0)$        ▷ Section 3.2
    for step $t$=1 to $T$ do
        if $random() < \epsilon_1$ then
            if $random() < \epsilon_2$ then
                Select random action $a_t \in A$
            else
                Select random action $a_t \in A_t$
        else
            Compute $Q(\mathbf{s_t}, \mathbf{a^{(i)}}; \theta)$ for $a^{(i)} \in A$ for network parameters $\theta$        ▷ Section 3.3, Eq. 6
            Select $a_t$ based on $\pi(a|s_t)$
        Execute action $a_t$ in the simulator and observe reward $r_t$
        Receive next observation $ot + 1$
        $G_{t+1} \leftarrow updateGraph(G_t, o_{t+1}); A_{t+1} \leftarrow pruneActions(A, G_{t+1})$        ▷ Section 3.1
        Compute $\mathbf{s_{t+1}}$ and $\mathbf{A_{t+1}} = \{\mathbf{a'^{(i)}}$ for all $a'^{(i)} \in A\}$        ▷ Section 3.3
        Set priority $p_t = 1$ if $r_t > 0$, else $p_t = 0$
        Store transition $(\mathbf{s_t}, \mathbf{a_t}, r_t, \mathbf{s_{t+1}}, \mathbf{A_{t+1}}, p_t)$ in replay buffer $D$
        Sample mini-batch of transitions $(\mathbf{s_k}, \mathbf{a_k}, r_k, \mathbf{s_{k+1}}, \mathbf{A_{k+1}}, p_k)$ from $D$, with fraction $\rho$ having $p_k = 1$
        Set $y_k = r_k + \gamma \max_{\mathbf{a} \in \mathbf{A_{k+1}}} Q(\mathbf{s_t}, \mathbf{a}; \theta)$, or $y_k = r_k$ if $s_{k+1}$ is terminal
        Perform gradient descent step on loss function $L(\theta) = (y_k - Q(\mathbf{s_t}, \mathbf{a_t}; \theta))^2$
```

2. Pruned actions without pre-training
3. Pruned actions with pre-training (full)

Our models use 50-dimensional word embeddings, 2 heads on the graph attention layers, mini-batch size of 16, and perform a gradient descent update every 5 steps taken by the agent.

All models are evaluated by observing the (a) time to reward convergence, and (b) the average number of steps required for the agent to finish the game with $\epsilon = 0.1$ over 5 episodes after training has completed. Following Narasimhan et al. (2015) we set ϵ to a non-zero value because text adventure games, by nature, require exploration to complete the quests. All results are reported based on multiple independent trials. For the large set of games, we only perform experiments on the best performing models found in the small set of games. Also note that for experiments on large games, we do not display the entire learning curve for the LSTM-DQN baseline, as it converges significantly more slowly than KG-DQN. We run each experiment 5 times and average the results.

Additionally, human performance on the both the games was measured by counting the number of steps taken to finish the game, with and without instructions on the exact quest. We modified Textworld to give the human players reward feedback in the form of a score, the reward function itself is identical to that received by the deep reinforcement learning agents. In one variation of this experiment, the human was given instructions on the potential sequence of steps that are required to finish the game in addition to the reward in the form of a score and in the other variation, the human received no instructions.

6 Results and Discussion

Recall that the number of steps required to finish the game for the oracle agent is 5 and 10 for the small and large maps respectively. It is impossible to achieve this ideal performance due to the structure of the quest. The player needs to interact with objects and explore the environment in order to figure out the exact sequence of actions required to finish the quest. To help benchmark our agent's performance, we observed people unaffiliated with the research playing through the same TextWorld "home" quests as the other models. Those who did not receive instructions on how to finish the quest never finished a single quest and gave up after an average of 184 steps on the small map and an average of 190 steps on the large map. When given instructions, human players completed the quest on the large map in an average of 23 steps, finishing the game with the maximum reward possible. Also note that none of the deep reinforcement learning agents received instructions.

On both small and large maps, all versions of KG-DQN tested converge faster than baselines (see Figure 3 for the small game and Figure 4 for the large game). We don't show BOW-DQN because it is strictly inferior to LSTM-DQN in all situations). KG-DQN converges 40% faster than baseline on the small game; both KG-DQN and the LSTM-DQN baseline reaches the maximum reward of five. On the large game, no

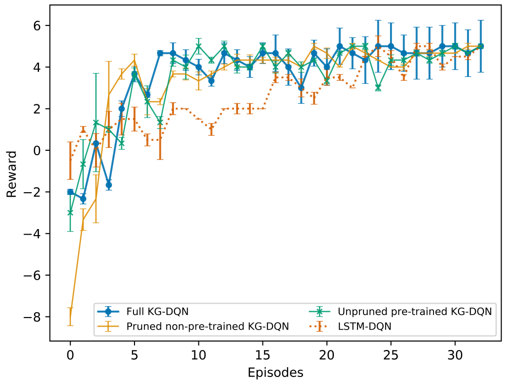


Figure 3: Reward learning curve for select experiments with the small games. Best viewed in color.

Model	Steps
Random Command	319.8
BOW-DQN	83.1 ± 8.0
LSTM-DQN	72.4 ± 4.6
Unpruned, pre-trained KG-DQN	131.7 ± 7.7
Pruned, non-pre-trained KG-DQN	97.3 ± 9.0
Full KG-DQN	73.7 ± 8.5

Table 3: Average number of steps (and standard deviation) taken to complete the small game.

agents achieve the maximum reward of 10, and the LSTM-DQN requires more than 300 episodes to converge at the same level as KG-DQN. Since all versions of KG-DQN converge at approximately the same rate, we conclude that the knowledge graph—i.e., persistent memory—is the main factor helping convergence time since it is the common element across all experiments.

After training is complete, we measure the number of steps each agent needs to complete each quest. Full KG-DQN requires an equivalent number of steps in the small game (Table 3) and in the large game (Table 4). Differences between LSTM-DQN and full KG-DQN are not statistically significant, $p = 0.199$ on an independent T-test. The ablated versions of KG-DQN—unpruned KG-DQN and non-pre-trained KG-DQN—require many more steps to complete quests. TextWorld's reward function allows for a lot of exploration of the environment without penalty so it is possible for a model that has converged on reward to complete quests in as few as five steps or in many hundreds of steps. From these results, we conclude that the pre-training using our question-answering paradigm is allowing the agent to find a general understanding of how to pick good actions even when the agent has never seen the final

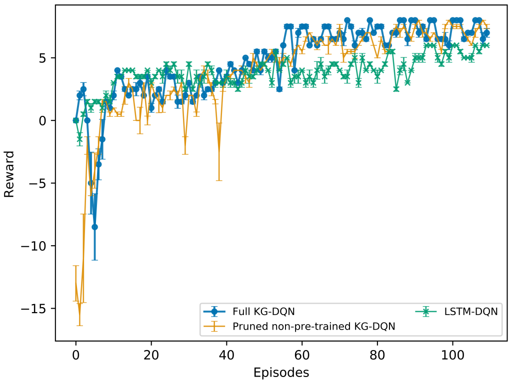


Figure 4: Reward learning curve for select experiments with the large games. Best viewed in color.

Model	Steps
Random Command	2054.8
LSTM-DQN	260.3 ± 4.5
Pruned, non-pre-trained KG-DQN	340 ± 6.4
Full KG-DQN	265.9 ± 9.4

Table 4: Average number of steps (and standard deviation) taken to complete the large game.

test game. LSTM-DQN also learns how to choose actions efficiently, but this knowledge is captured in the LSTM's cell state, whereas in KG-DQN this knowledge is made explicit in the knowledge graph and retrieved effectively by graph attention. Taken together, KG-DQN converges faster without loss of quest solution quality.

7 Conclusions

We have shown that incorporating knowledge graphs into an deep Q-network can reduce training time for agents playing text-adventure games of various lengths. We speculate that this is because the knowledge graph provides a persistent memory of the world as it is being explored. While the knowledge graph allows the agent to reach optimal reward more quickly, it doesn't ensure a high quality solution to quests. Action pruning using the knowledge graph and pre-training of the embeddings used in the deep Q-network result in shorter action sequences needed to complete quests.

The insight into pre-training portions of the agent's architecture is based on converting text-adventure game playing into a question-answering activity. That is, at every step, the agent is asking—and trying to answer—what is the most important thing to try. The pre-training acts as a form of transfer learning from different, but related games. However, question-answering alone cannot solve the text-adventure playing problem because there will always be some trial and error required.

By addressing the challenges of partial observability and combinatorially large action, spaces through persistent memory, our work on playing text-adventure games addresses a critical need for reinforcement learning for language. Text-adventure games can be seen as a stepping stone toward more complex, real-world tasks; the human world is one of partial understanding through communication and acting on the world using language.

References

Gabor Angeli, Johnson Premkumar, Melvin Jose, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. 2010. Towards understanding situated natural language. In *Proceedings of the 2010 International Conference on Artificial Intelligence and Statistics*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. TextWorld : A Learning Environment for Text-based Games. In *Proceedings of the ICML/IJCAI 2018 Workshop on Computer Games*, page 29.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045.

Jian Guan, Yansen Wang, and Minlie Huang. 2018. Story Ending Generation with Incremental Encoding and Commonsense Knowledge. *arXiv:1808.10113v1*.

Matan Haroush, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2018. Learning How Not to Act in Text-Based Games. In *Workshop Track at ICLR 2018*, pages 1–4.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep Reinforcement Learning with a Natural Language Action Space. In *Association for Computational Linguistics (ACL)*.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Carnegie Mellon University.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2017. The Natural Language Decathlon : Multitask Learning as Question Answering. *arXiv:1806.08730*.

Andrew W. Moore and Christopher G. Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language Understanding for Text-based Games Using Deep Reinforcement Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robert Speer and Catherine Havasi. 2012. Representing General Relational Knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. In *Proceedings of the 2018 NeurIPS Workshop on Wordplay: Reinforcement and Language Learning in Text-based Games*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations (ICLR)*.

Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning*, 8(3):279–292.

Information Aggregation for Multi-Head Attention with Routing-by-Agreement

Jian Li[1] **Baosong Yang**[2] **Zi-Yi Dou**[3] **Xing Wang**[4] **Michael R. Lyu**[1] **Zhaopeng Tu**[4*]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong
{jianli,lyu}@cse.cuhk.edu.hk
[2]University of Macau
nlp2ct.baosong@gmail.com
[3]Carnegie Mellon University
zdou@andrew.cmu.edu
[4]Tencent AI Lab
{brightxwang,zptu}@tencent.com

Abstract

Multi-head attention is appealing for its ability to jointly extract different types of information from multiple representation subspaces. Concerning the information aggregation, a common practice is to use a concatenation followed by a linear transformation, which may not fully exploit the expressiveness of multi-head attention. In this work, we propose to improve the information aggregation for multi-head attention with a more powerful *routing-by-agreement* algorithm. Specifically, the routing algorithm iteratively updates the proportion of how much a part (i.e. the distinct information learned from a specific subspace) should be assigned to a whole (i.e. the final output representation), based on the agreement between parts and wholes. Experimental results on linguistic probing tasks and machine translation tasks prove the superiority of the advanced information aggregation over the standard linear transformation.

1 Introduction

Attention model becomes a standard component of the deep learning networks, contributing to impressive results in machine translation (Bahdanau et al., 2015; Luong et al., 2015), image captioning (Xu et al., 2015), speech recognition (Chorowski et al., 2015), among many other applications. Its superiority lies in the ability of modeling the dependencies between representations without regard to their distance. Recently, the performance of attention is further improved by multi-head mechanism (Vaswani et al., 2017), which parallelly performs attention functions on different representation subspaces of the input sequence. Consequently, different attention heads are able to capture distinct linguistic properties of the input, which are embedded in different subspaces (Raganato and Tiedemann, 2018). Subsequently, a linear transformation is generally employed to aggregate the partial representations extracted by different attention heads (Vaswani et al., 2017; Ahmed et al., 2018).

Most existing work focus on extracting informative or distinct partial-representations from different subspaces (e.g. Lin et al., 2017; Li et al., 2018), while few studies have paid attention to the aggregation of the extracted partial-representations. Arguably, information extraction and aggregation are both crucial for multi-head attention to generate an informative representation. Recent studies in multimodal learning show that a straightforward linear transformation for fusing features in different sets of representations usually limits the extent of abstraction (Fukui et al., 2016; Ben-Younes et al., 2017). A natural question arises: *whether the straightforward linear transformation is expressive enough to fully capture the rich information distributed in the extracted partial-representations?*

In this work, we provide the first answer to this question. We propose to empirically validate the importance of information aggregation in multi-head attention, by comparing the performance of the standard linear function and advanced aggregation functions on various tasks. Specifically, we cast information aggregation as the *assigning-parts-to-wholes* problem (Hinton et al., 2011), and investigate the effectiveness of the routing-by-agreement algorithm – an appealing alternative to solving this problem (Sabour et al., 2017; Hinton et al., 2018). The routing algorithm iteratively updates the proportion of how much a part should be assigned to a whole, based on the agreement between parts and wholes. We leverage the routing algorithm to aggregate the information distributed in the extracted partial-representations.

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was mainly conducted when Jian Li, Baosong Yang, and Zi-Yi Dou were interning at Tencent AI Lab.

Proceedings of NAACL-HLT 2019, pages 3566–3575
Minneapolis, Minnesota, June 2 - June 7, 2019.

We evaluate the performance of the aggregated representations on both linguistic probing tasks as well as machine translation tasks. The probing tasks (Conneau et al., 2018) consists of 10 classification problems to study what linguistic properties are captured by input representations. Probing analysis show that our approach indeed produces more informative representation, which embeds more syntactic and semantic information. For translation tasks, we validate our approach on top of the advanced TRANSFORMER model (Vaswani et al., 2017) on both WMT14 English⇒German and WMT17 Chinese⇒English data. Experimental results show that our approach consistently improves translation performance across languages while keeps the computational efficiency.

The main contributions of this paper can be summarized as follows:

- To our best knowledge, this is the first work to demonstrate the necessity and effectiveness of advanced information aggregation for multi-head attention.
- Our work is among the few studies (*cf.* (Gong et al., 2018; Zhao et al., 2018; Dou et al., 2019)) which prove that the idea of capsule networks can have promising applications on natural language processing tasks.

2 Background

Attention mechanism aims at modeling the relevance between representation pairs, thus a representation is allowed to build a direct relation with another representation. Instead of performing a single attention function, Vaswani et al. (2017) found it is beneficial to capture different context features with multiple individual attention functions, namely multi-head attention. Formally, attention function maps a sequence of query $\mathbf{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_J\}$ and a set of key-value pairs $\{\mathbf{K}, \mathbf{V}\} = \{(\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_M, \mathbf{v}_M)\}$ to outputs, where $\mathbf{Q} \in \mathbb{R}^{J \times d}$, $\{\mathbf{K}, \mathbf{V}\} \in \mathbb{R}^{M \times d}$. More specifically, multi-head attention model first transforms $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ into H subspaces with different, learnable linear projections:

$$\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h = \mathbf{Q}\mathbf{W}_h^Q, \mathbf{K}\mathbf{W}_h^K, \mathbf{V}\mathbf{W}_h^V, \quad (1)$$

where $\{\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h\}$ are respectively the query, key, and value representations of the h-th head. $\{\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V\} \in \mathbb{R}^{d \times \frac{d}{H}}$ denote parameter matrices associated with the h-th head, where d represents the dimensionality of the model hidden states. Furthermore, H attention functions are applied in parallel to produce the output states $\{\mathbf{O}_1, \dots, \mathbf{O}_H\}$, among them:

$$\mathbf{O}_h = \textsc{Att}(\mathbf{Q}_h, \mathbf{K}_h)\mathbf{V}_h, \quad (2)$$

where $\mathbf{O}_h \in \mathbb{R}^{J \times \frac{d}{H}}$, $\textsc{Att}(\cdot)$ is an attention model. In this work, we use scaled dot-product attention (Luong et al., 2015), which achieves similar performance with its additive counterpart (Bahdanau et al., 2015) while is much faster and more space-efficient in practice (Vaswani et al., 2017).

Finally, the H output states are concatenated and linearly transformed to produce the final state:

$$\begin{aligned} \text{Concat:} \quad \widehat{\mathbf{O}} &= [\mathbf{O}_1, \dots, \mathbf{O}_H], && (3) \\ \text{Linear:} \quad \mathbf{O} &= \widehat{\mathbf{O}}\mathbf{W}^O, && (4) \end{aligned}$$

where $\mathbf{O} \in \mathbb{R}^{J \times d}$ denotes the final output states, $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ is a trainable matrix.

As shown in Equations 3 and 4, the conventional multi-head attention uses a straightforward concatenation and linear mapping to aggregate the output representations of multiple attention heads. We argue that this straightforward strategy may not fully exploit the expressiveness of multi-head attention, which can benefit from advanced information aggregation by exploiting the intrinsic relationship among the learned representations.

3 Related Work

Our work synthesizes two strands of research work, namely *multi-head attention* and *information aggregation*.

3.1 Multi-Head Attention

Multi-head attention has shown promising empirical results in many NLP tasks, such as machine translation (Vaswani et al., 2017; Domhan, 2018), semantic role labeling (Strubell et al., 2018), and subject-verb agreement task (Tang et al., 2018). The strength of multi-head attention lies in the rich expressiveness by using multiple attention functions in different representation subspaces.

Previous work show that multi-head attention can be further enhanced by encouraging individual attention heads to extract distinct information. For example, Lin et al. (2017) introduce a penalization term to reduce the redundancy of attention weights among different attention heads. Li et al. (2018) propose disagreement regularizations to encourage different attention heads to capture distinct features, and Yang et al. (2019) model the interactions among attention heads. Shen et al. (2018) explicitly use multiple attention heads to model different dependencies of the same word pair, and Strubell et al. (2018) employ different attention heads to capture different linguistic features. Our approach is complementary to theirs, since they focus on extracting distinct information while ours aims at effectively aggregating the extracted information. Our study shows that information aggregation is as important as information extraction for multi-head attention.

3.2 Information Aggregation

Information aggregation in multi-head attention (e.g. Equations 3 and 4) aims at composing the partial representations of the input captured by different attention heads to a final representation. Recent work shows that representation composition benefits greatly from advanced functions beyond simple concatenation or mean/max pooling. For example, Fukui et al. (2016) and Ben-Younes et al. (2017) succeed on fusing multi-modal features (e.g., visual features and textual features) more effectively via employing the higher-order bilinear pooling instead of vector concatenation or element-wise operations. In NLP tasks, Peters et al. (2018) aggregate layer representations with linear combination, and Dou et al. (2018) compose deep representations with layer aggregation and multi-layer attention mechanisms.

Recently, the routing-by-agreement algorithm, which origins from the capsule networks (Hinton et al., 2011), becomes an appealing alternative to representation composition. The majority of existing work on capsule networks has focused on computer vision tasks, such as MNIST tasks (Sabour et al., 2017; Hinton et al., 2018), CIFAR tasks (Xi et al., 2017), and object segmentation task (LaLonde and Bagci, 2018). The applications of capsule networks in NLP tasks, however, have not been widely investigated to date. Zhao et al. (2018) testify capsule networks on text classification tasks and Gong et al. (2018) propose to aggregate a sequence of vectors via dynamic routing for sequence encoding. Dou et al. (2019) use routing-by-agreement strategies to aggregate layer representations dynamically. Inspired by these successes, we apply the routing algorithms to multi-head attention on both linguistic probing and machine translation tasks, which demonstrates the necessity and effectiveness of advanced information aggregation for multi-head attention.

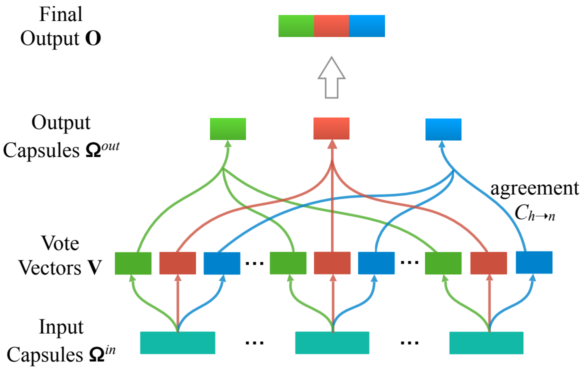


Figure 1: Illustration of routing-by-agreement.

4 Approach

In this work, we cast information aggregation in multi-head attention as the problem of *assigning-parts-to-wholes*. Specifically, each attention head extracts different linguistic properties of the same input (Raganato and Tiedemann, 2018), and the goal of information aggregation is to compose the partial representations extracted by different heads to a whole representation. An appealing solution to this problem is the *routing-by-agreement* algorithm, as shown in Figure 1.

The routing algorithm consists of two layers: *input capsules* and *output capsules*. The input capsules are constructed from the transformation of the partial representations extracted by different attention heads. For each output capsule, each input capsule proposes a distinct "voting vector", which represents the proportion of how much the information is transformed from this input capsule (i.e parts) to the corresponding output capsule (i.e. wholes). The proportion is iteratively updated based on the agreement between the voting vectors and the output capsule. Finally, all output capsules are concatenated to form the final representation.

4.1 Routing-by-Agreement

Mathematically, the input capsules $\mathbf{\Omega}^{in} = \{\mathbf{\Omega}^{in}_1, \ldots, \mathbf{\Omega}^{in}_H\}$ with $\mathbf{\Omega}^{in} \in \mathbb{R}^{n\times d}$ are constructed from the outputs of multi-head attention:

$$\mathbf{\Omega}^{in}_h = f_h(\widehat{\mathbf{O}}), \tag{5}$$

where $f_h(\cdot)$ is a distinct non-linear transformation function associated with the input capsule

Algorithm 1 Iterative Simple Routing.

```
1: procedure ROUTING($\mathbf{V}$, $T$):
2:     $\forall \mathbf{V}_{h\to *}$: $B_{h\to n} = 0$
3:     for $T$ iterations do
4:         $\forall \mathbf{V}_{h\to *}$: $C_{h\to n} = \frac{\exp(B_{h\to n})}{\sum_{n'=1}^{N} \exp(B_{h\to n'})}$
5:         $\forall \mathbf{\Omega}_n^{out}$: compute $\mathbf{\Omega}_n^{out}$ by Eq. 7
6:         $\forall \mathbf{V}_{h\to *}$: $B_{h\to n}$ += $\mathbf{\Omega}_n^{out} \cdot \mathbf{V}_{h\to n}$
    return $\mathbf{\Omega}$
```

Algorithm 2 Iterative EM Routing.

```
1: procedure EM ROUTING($\mathbf{V}$, $T$):
2:     $\forall \mathbf{V}_{h\to *}$: $C_{l\to n} = 1/N$
3:     for $T$ iterations do
4:         $\forall \mathbf{\Omega}_n^{out}$: M-STEP($\mathbf{V}$, $C$)        ▷ hold $C$ constant, adjust $(\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n, A_n)$
5:         $\forall \mathbf{V}_{h\to *}$: E-STEP($\mathbf{V}$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $A$)    ▷ hold $(\boldsymbol{\mu}, \boldsymbol{\sigma}, A)$ constant, adjust $C_{h\to *}$
6:     $\forall \mathbf{\Omega}_n^{out}$: $\mathbf{\Omega}_n^{out} = A_n * \boldsymbol{\mu}_n$
    return $\mathbf{\Omega}$
```

$\mathbf{\Omega}_h^{in}$. Given N output capsules, each input capsule $\mathbf{\Omega}_h^{in}$ propose N "vote vectors" $\mathbf{V}_{h\to *} = \{\mathbf{V}_{h\to 1}, \dots, \mathbf{V}_{h\to N}\}$, which is calculated by

$$\mathbf{V}_{h\to n} = \mathbf{\Omega}_h^{in} \mathbf{W}_{h\to n}, \tag{6}$$

Each output capsule $\mathbf{\Omega}_n^{out}$ is calculated as the normalization of its total input, which is a weighted sum over all "vote vectors" $\mathbf{V}_{*\to n}$:

$$\mathbf{\Omega}_n^{out} = \frac{\sum_{h=1}^{H} C_{h\to n} \mathbf{V}_{h\to n}}{\sum_{h=1}^{H} C_{h\to n}}, \tag{7}$$

The weight $C_{h\to n}$ with $\sum_n C_{h\to n} = 1$ measures the agreement between vote vector $\mathbf{V}_{h\to n}$ and output capsule $\mathbf{\Omega}_n^{out}$, which is determined by the iterative routing as described in the next section. Note that $\sum_{h=1}^{H} C_{h\to n}$ is not necessarily equal to 1. After the routing process, following Gong et al. (2018), we concatenate the N output capsules to form the final representation: $\mathbf{O} = [\mathbf{\Omega}_1^{out}, \dots, \mathbf{\Omega}_N^{out}]$. To make the dimensionality of the final output be consistent with that of hidden layer (i.e. d), we set the dimensionality of each output capsule be $\frac{d}{N}$.

4.2 Routing Mechanisms

In this work, we explore two representative routing mechanisms, namely *simple routing* (Sabour et al., 2017) and *EM routing* (Hinton et al., 2018), which differ at how the agreement weights $C_{h\to n}$ are calculated.

4.2.1 Simple Routing

Algorithm 1 lists a straightforward implementation of routing mechanism. $B_{h\to n}$ measures the degree that the input capsule $\mathbf{\Omega}_h^{in}$ should be coupled to the output capsule $\mathbf{\Omega}_n^{out}$, which is initialized as all 0 (Line 2). The agreement weights $C_{h\to n}$ are then iteratively refined by measuring the agreement between vote vector $\mathbf{V}_{h\to n}$ and output capsule $\mathbf{\Omega}_n^{out}$ (Lines 4-6), which is implemented as a simple scalar product $\mathbf{\Omega}_n^{out} \cdot \mathbf{V}_{h\to n}$ (Line 6).

To represent the probability that the output capsule $\mathbf{\Omega}_n^{out}$ is activated, Sabour et al. (2017) use a non-linear "squashing" function:

$$\mathbf{\Omega}_n^{out} = \frac{||\mathbf{\Omega}_n^{out}||^2}{1 + ||\mathbf{\Omega}_n^{out}||^2} \frac{\mathbf{\Omega}_n^{out}}{||\mathbf{\Omega}_n^{out}||}, \tag{8}$$

The scalar product $\mathbf{\Omega}_n^{out} \cdot \mathbf{V}_{h\to n}$ saturates at 1, which makes it insensitive to the difference between a quite good agreement and a very good agreement. In response to this problem, Hinton et al. (2018) propose a novel Expectation-Maximization (EM) routing algorithm.

4.2.2 EM Routing

Comparing with simple routing, EM routing has two modifications. First, it explicitly assigns an activation probability A to represent the probability of whether each output capsule is activated, rather than the length of vector calculated by a squashing function (Equation 8). Second, it casts the routing process as fitting a mixture of Gaussians using EM algorithm, where the output capsules play the role of Gaussians and the means of the input capsules play the role of the datapoints. Accordingly, EM routing can better estimate the agreement by allowing activated output capsules to receive a cluster of similar votes.

Algorithm 2 lists the EM routing, which iteratively adjusts the means, variances, and activation probabilities $(\boldsymbol{\mu}, \boldsymbol{\sigma}, A)$ of the output capsules, as well as the agreement weights C of the input capsules (Lines 4-5). The representation of output capsule $\mathbf{\Omega}_n^{out}$ is calculated as

$$\mathbf{\Omega}_n^{out} = A_n * \boldsymbol{\mu}_n = A_n * \frac{\sum_{h=1}^{H} C_{h\to n} \mathbf{V}_{h\to n}}{\sum_{h=1}^{H} C_{h\to n}}, \tag{9}$$

The EM algorithm alternates between an E-step and an M-step. The E-step determines, for each datapoint (i.e. input capsule), the probability of agreement (i.e. C) between it and each of the Gaussians (i.e. output capsules). The M-step holds the agreement weights constant, and for each Gaussian (i.e. output capsule) consists of finding the mean of these weighted datapoints (i.e. input capsules) and the variance about that mean.

M-Step for each Gaussian (i.e. $\mathbf{\Omega}_n^{out}$) consists of finding the mean $\boldsymbol{\mu}_n$ of the votes from input capsules and the variance $\boldsymbol{\sigma}_n$ about that mean:

$$\boldsymbol{\mu}_n = \frac{\sum_{h=1}^{H} C_{h\to n}\mathbf{V}_{h\to n}}{\sum_{h=1}^{H} C_{h\to n}}, \quad (10)$$

$$(\boldsymbol{\sigma}_n)^2 = \frac{\sum_{h=1}^{H} C_{h\to n}(\mathbf{V}_{h\to n}-\boldsymbol{\mu}_n)^2}{\sum_{h=1}^{H} C_{h\to n}}. \quad (11)$$

The incremental cost of using an active capsule $\mathbf{\Omega}_n^{out}$ is

$$\chi_n = \sum_i \big(\log(\boldsymbol{\sigma}_n^i) + \frac{1+\log(2\pi)}{2}\big)\sum_{h=1}^{H} C_{h\to n},$$

where $\boldsymbol{\sigma}_n^i$ denotes the i-th dimension of the variance vector $\boldsymbol{\sigma}_n$. The activation probability of capsule $\mathbf{\Omega}_n^{out}$ is calculated by

$$A_n = logistic\big(\lambda(\beta_A - \beta_\mu \sum_{h=1}^{H} C_{h\to n} - \chi_n)\big),$$

where β_A is a fixed cost for coding the mean and variance of $\mathbf{\Omega}_n^{out}$ when activating it, β_μ is another fixed cost per input capsule when not activating it, and λ is an inverse temperature parameter set with a fixed schedule. We refer the readers to (Hinton et al., 2018) for more details.

E-Step adjusts the assignment probabilities $C_{h\to *}$ for each input $\mathbf{\Omega}_h^{in}$. First, we compute the negative log probability density of the vote $\mathbf{V}_{h\to n}$ from $\mathbf{\Omega}_h^{in}$ under the Gaussian distribution fitted by the output capsule $\mathbf{\Omega}_n^{out}$ it gets assigned to:

$$P_{h\to n} = \sum_i \frac{1}{\sqrt{2\pi(\boldsymbol{\sigma}_n^i)^2}} \exp(-\frac{(\mathbf{V}_{h\to n}^i - \boldsymbol{\mu}_n^i)^2}{2(\boldsymbol{\sigma}_n^i)^2}).$$

Again, i denotes the i-th dimension of the vectors $\{\mathbf{V}_{h\to n}, \boldsymbol{\mu}_n, \boldsymbol{\sigma}_n\}$. Accordingly, the agreement weight is re-normalized by

$$C_{h\to n} = \frac{A_n P_{h\to n}}{\sum_{n'=1}^{N} A_{n'} P_{h\to n'}}. \quad (12)$$

5 Experiments

In this section, we evaluate the performance of our proposed models on both linguistic probing tasks and machine translation tasks.

5.1 Linguistic Probing Tasks

5.1.1 Setup

Tasks Recently, Conneau et al. (2018) designed 10 probing tasks to study what linguistic properties are captured by input representations. A probing task is a classification problem that focuses on simple linguistic properties of sentences. 'SeLen' is to predict the length of sentences in terms of number of words. 'WC' tests whether it is possible to recover information about the original words given its sentence embedding. 'TrDep' checks whether an encoder infers the hierarchical structure of sentences. In 'ToCo' task, sentences should be classified in terms of the sequence of top constituents immediately below the sentence node. 'Bshif' tests whether two consecutive tokens within the sentence have been inverted. 'Tense' asks for the tense of the main-clause verb. 'SubNm' focuses on the number of the subject of the main clause. 'ObjNm' tests for the number of the direct object of the main clause. In 'SOMO', some sentences are modified by replacing a random noun or verb with another noun or verb and the classifier should tell whether a sentence has been modified. 'CoIn' benchmark contains sentences made of two coordinate clauses. Half of the sentences are inverted the order of the clauses and the task is to tell whether a sentence is intact or modified. We conduct probing tasks to study whether the routing-based aggregation benefits multi-head attention to produce more informative representation.

Data and Models The models on each classification task are trained and examined using the open-source dataset provided by Conneau et al. (2018), where each task is assigned 100k sentences for training and 10k sentences for validating and testing. Each of our probing model consists of 3 encoding layers followed by a MLP classifier. For each encoding layer, we employ a multi-head self-attention block and a feed-forward block as in TRANSFORMER-BASE, which have achieved promising results on several NLP tasks (Dehghani et al., 2018; Devlin et al., 2018). The mean of the top encoding layer is served as the sentence

Model	Surface		Syntactic			Semantic				
	SeLen	WC	TrDep	ToCo	BShif	Tense	SubNm	ObjNm	SOMO	CoIn
BASE	**97.22**	97.92	44.48	84.44	49.30	84.20	87.66	82.94	50.24	68.77
SIMPLE	97.10	**98.85**	43.37	86.15	49.87	**88.22**	87.25	85.07	48.77	69.12
EM	96.26	98.75	**47.72**	**87.00**	**51.82**	88.17	**89.97**	**86.40**	**51.55**	**69.86**

Table 1: Classification accuracies on 10 probing tasks of evaluating the linguistic properties ("Surface", "Syntectic", and "Semantic") learned by sentence encoder. "BASE" denotes the standard linear transformation, "SIMPLE" is the simple routing algorithm, and "EM" is the EM routing algorithm.

representation passed to the classifier. The difference between the compared models merely lies in the aggregation mechanism of multiple attention heads: "BASE" uses a standard concatenation and linear transformation, "SIMPLE" and "EM" are assigned simple routing and EM routing algorithms, respectively. For routing algorithms, the number of output capsules and routing iterations are empirically set to 512 and 3.

5.1.2 Results

Table 1 lists the classification accuracies of the three models on the 10 probing tasks. We highlight the best accuracies in bold. Several observations can be made here.

First, *routing-based models produce more informative representation.* The representation produced by encoders with routing-based aggregation outperforms that by the baseline in most tasks, proving that routing mechanisms indeed aggregate attention heads more effectively. The only exception is the sentence length classification task ('SeLen'), which is consistent with the conclusion in (Conneau et al., 2018): as a model captures deeper linguistic properties, it will tend to forget about this superficial feature.

Second, *EM routing outperforms simple routing by embedding more syntactic and semantic information.* As shown in the last row, EM routing for multi-head aggregation consistently achieves best performances on most syntactic and semantic tasks. Especially on task 'TrDep', 'Tense' and 'ObjNm', EM routing-based model surpasses the baseline more than 3 points, demonstrating that EM routing benefits multi-head attention to capture more syntax structure and sentence meaning. Simple routing, however, underperforms the baseline model in some cases such as 'TrDep' and 'SubNm'. We attribute the superiority of EM routing to generating more accurate agreement weights with the Gaussian estimation.

5.2 Machine Translation Tasks

5.2.1 Setup

Data We conduct experiments on the widely-used WMT2014 English⇒German (En⇒De) and WMT2017 Chinese⇒English (Zh⇒En) machine translation tasks. For the En⇒De task, the dataset consists of 4.6M sentence pairs. We use newstest2013 as the development set and newstest2014 as the test set. For the Zh⇒En task, we use all of the available parallel data, consisting of about 20.6M sentence pairs. We use newsdev2017 as the development set and newstest2017 as the test set. We employ byte-pair encoding (BPE) (Sennrich et al., 2016) with 32K merge operations for both language pairs. We use the case-sensitive 4-gram NIST BLEU score (Papineni et al., 2002) as evaluation metric, and bootstrap resampling (Koehn, 2004) for statistical significance test.

Models We implement the proposed approaches on top of the advanced TRANSFORMER model (Vaswani et al., 2017). We follow Vaswani et al. (2017) to set the configurations and have reproduced their reported results on the En⇒De task. The *Base* and *Big* models differ at hidden size (512 vs. 1024) and number of attention heads (8 vs. 16). All the models are trained on eight NVIDIA P40 GPUs where each is allocated with a batch size of 4096 tokens.

TRANSFORMER consists of three attention components: encoder-side self-attention, decoder-side self-attention and encoder-decoder attention, all of which are implemented as multi-head attention. For the information aggregation in multi-head attention, we replace the standard linear transformation with the proposed routing mechanisms. We experimentally set the number of iterations to 3 and the number of output capsules as model hidden size, which outperform other configurations during our investigation.

#	Applying Aggregation to ...			Routing	# Para.	Speed		BLEU	△
1	*Enc-Self*	*Enc-Dec*	*Dec-Self*			Train	Decode		
2	×	×	×	n/a	88.0M	1.92	1.67	27.31	–
3	✓	×	×	Simple	+12.6M	1.23	1.66	27.98	+0.67
4	✓	×	×	EM	+12.6M	1.20	1.65	28.28	+0.97
5	×	✓	×	EM	+12.6M	1.20	1.21	27.94	+0.63
6	×	×	✓	EM	+12.6M	1.21	1.21	28.15	+0.84
7	✓	✓	×	EM	+25.2M	0.87	1.20	28.45	+1.14
8	✓	✓	✓	EM	+37.8M	0.66	0.89	28.47	+1.16

Table 2: Effect of information aggregation on different attention components, i.e., encoder self-attention ("*Enc-Self*"), encoder-decoder attention ("*Enc-Dec*"), and decoder self-attention ("*Dec-Self*"). "# Para." denotes the number of parameters, and "Train" and "Decode" respectively denote the training speed (steps/second) and decoding speed (sentences/second).

#	Layers	# Para.	Train	BLEU
1	None	88.0M	1.92	27.31
2	[1-6]	100.6M	1.20	28.28
3	[4-6]	94.3M	1.54	28.26
4	[1-3]	94.3M	1.54	28.27
5	[1,2]	92.2M	1.67	28.26
6	[6]	90.1M	1.88	27.68
7	[1]	90.1M	1.88	27.75

Table 3: Evaluation of different layers in the encoder, which are implemented as multi-head self-attention with the EM routing based information aggregation. "1" denotes the bottom layer, and "6" the top layer.

5.2.2 Component Analysis

Table 2 lists the results on the En⇒De translation task with TRANSFORMER-BASE. As seen, the proposed routing mechanism outperforms the standard aggregation in all cases, demonstrating the necessity of advanced aggregation functions for multi-head attention.

Routing Mechanisms (Rows 3-4) We first apply simple routing and EM routing to encoder self-attention. Both strategies perform better than the standard multi-head aggregation (Row 1), verifying the effectiveness of the non-linear aggregation mechanisms. Specifically, the two strategies require comparable parameters and computational speed, but EM routing achieves better performance on translation qualities. Considering the training speed and performance, *EM routing* is used as the default multi-head aggregation method in subsequent experiments.

Effect on Attention Components (Rows 4-8) Concerning the individual attention components (Rows 4-6), we found that the encoder and decoder self-attention benefit more from the routing-based information aggregation than the encoder-decoder attention. This is consistent with the finding in (Tang et al., 2018), which shows that self-attention is a strong semantic feature extractor. Encouragingly, applying EM routing in the encoder (Row 4) significantly improve the translation quality with almost no decrease in decoding speed, which matches the requirement of online MT systems. We find that this is due to the auto-regressive generation schema, modifications on the decoder influence the decoding speed more than the encoder.

Compared with individual attention components, applying routing to multiple components (Rows 7-8) marginally improves translation performance, at the cost of a significant decrease of the training and decoding speeds. Possible reasons include that the added complexity makes the model harder to train, and the benefits enjoyed by different attention components are overlapping to some extent. To balance translation performance and efficiency, we only apply EM routing to aggregate multi-head self-attention at the *encoder* in subsequent experiments.

Encoder Layers As shown in Row 4 of Table 2, applying EM routing to all encoder layers significantly decreases the training speed by 37.5%, which is not acceptable since TRANSFORMER is best known for both good performance and quick training. We expect applying to fewer layers can alleviate the training burden. Recent studies show that different layers of NMT encoder can capture different levels of syntax and semantic features (Shi et al., 2016; Peters et al., 2018). There-

System	Architecture	En⇒De		Zh⇒En	
		# Para.	BLEU	# Para.	BLEU
Existing NMT systems					
(Wu et al., 2016)	GNMT	n/a	26.30	n/a	n/a
(Gehring et al., 2017)	ConvS2S	n/a	26.36	n/a	n/a
(Vaswani et al., 2017)	Transformer-Base	65M	27.3	n/a	n/a
	Transformer-Big	213M	28.4	n/a	n/a
(Hassan et al., 2018)	Transformer-Big	n/a	n/a	n/a	24.2
Our NMT systems					
this work	Transformer-Base	88M	27.31	108M	24.13
	+ Effective Aggregation	92M	$28.26^{\Uparrow}$	112M	$24.68^{\Uparrow}$
	Transformer-Big	264M	28.58	304M	24.56
	+ Effective Aggregation	297M	$28.96^{\uparrow}$	337M	$25.00^{\uparrow}$

Table 4: Comparing with existing NMT systems on WMT14 English⇒German ("En⇒De") and WMT17 Chinese⇒English ("Zh⇒En") tasks. "$\uparrow$ / $\Uparrow$": significantly better than the baseline counterpart ($p < 0.05/0.01$).

fore, an investigation to study whether EM routing works for multi-head attention at different layers is highly desirable.

As shown in Table 3, we respectively employ EM routing for multi-head attention at the high-level three layers (Row 3) and low-level three layers (Row 4). The translation quality marginally drop while parameters are fewer and training speeds are quicker. This phenomena verifies that it is unnecessary to apply the proposed model to all layers. We further reduce the applied layers to low-level two (Row 5), the above phenomena still holds. However, a big drop on translation quality occurs when the number of layer is reduced to 1 (Rows 6-7).

Accordingly, to balance translation performance and efficiency, we only apply EM routing for multi-head aggregation at the *low-level two layers of the encoder*, which we term "Effective Aggregation" in the following sections.

5.2.3 Main Results

In this section, we validate the proposed "Effective Aggregation" for multi-head attenion on both WMT17 Zh⇒En and WMT14 En⇒De translation tasks. The results are listed in Table 4. Our implementations of both Transformer-Base and Transformer-Big outperform the reported NMT systems on the same data and match the strong results of Transformer reported in previous works, which we believe make the evaluation convincing. Incorporating the effective aggregation consistently and significantly improves translation performance for both base and big Transformer models across language pairs, demonstrating the efficiency and universality of our proposed multi-head aggregation mechanism.

Moreover, it is encouraging to see that Transformer-Base with effective aggregation strategy even achieves comparable performances to that of Transformer-Big, with about two thirds fewer parameters, which further demonstrates that our performance gains are not simply brought by additional parameters.

6 Conclusion

In this work, we provide first empirical validation on the importance of information aggregation for multi-head attention. Instead of the conventional linear transformation, we propose to aggregate the partial representations learned by multiple attention heads via *routing-by-agreement*. The routing algorithm iteratively updates the proportion of how much a partial representation should be assigned to the final output representation, based on the agreement between parts and wholes.

Experimental results across 10 linguistic probing tasks reveal that our EM routing-based model indeed produces more informative representation, which benefits multi-head attention to capture more syntactic and semantic information. In addition, our approach on various machine translation tasks consistently and significantly outperforms the strong Transformer baseline. Extensive analysis further suggests that only applying EM routing to low-level two layers of the encoder can best balance the translation performance and computational efficiency.

Future work includes combining our information aggregation techniques together with other advanced information extraction models for multi-head attention (Li et al., 2018). We expect that the two kinds of approaches can complement each other to further improve the expressiveness of multi-head attention.

Acknowledgments

Jian Li and Michael R. Lyu were supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14210717 of the General Research Fund), and Microsoft Research Asia (2018 Microsoft Research Asia Collaborative Research Award). We thank the anonymous reviewers for their insightful comments and suggestions.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2018. Weighted Transformer Network for Machine Translation. *arXiv preprint arXiv:1711.02132*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. 2017. Mutan: Multimodal Tucker Fusion for Visual Question Answering. In *ICCV*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based Models for Speech Recognition. In *NIPS*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What You Can Cram into A Single $&!#∗ Vector: Probing Sentence Embeddings for Linguistic Properties. In *ACL*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal Transformers. *arXiv preprint arXiv:1807.03819*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Tobias Domhan. 2018. How Much Attention Do You Need? A Granular Analysis of Neural Machine Translation Architectures. In *ACL*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation. In *AAAI*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information Aggregation via Dynamic Routing for Sequence Encoding. In *COLING*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving Human Parity on Automatic Chinese to English News Translation. *arXiv preprint arXiv:1803.05567*.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming Auto-encoders. In *ICANN*.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix Capsules with EM Routing. In *ICLR*.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *EMNLP*.

Rodney LaLonde and Ulas Bagci. 2018. Capsules for object segmentation. *arXiv preprint arXiv:1804.04241*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for Automatic Evaluation of Machine Translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

Alessandro Raganato and Jörg Tiedemann. 2018. An Analysis of Encoder Representations in Transformer-Based Machine Translation. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic Routing Between Capsules. In *NIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding. In *AAAI*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does String-based Neural MT Learn Source Syntax? In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Edgar Xi, Selina Bing, and Yang Jin. 2017. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional self-attention networks. In *NAACL*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. In *EMNLP*.

Context-Dependent Semantic Parsing over Temporally Structured Data

Charles Chen and Razvan Bunescu
School of Electrical Engineering and Computer Science, Ohio University
`lc971015@ohio.edu, bunescu@ohio.edu`

Abstract

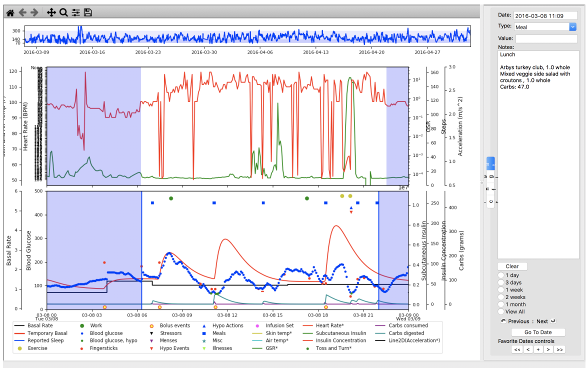
We describe a new semantic parsing setting that allows users to query the system using both natural language questions and actions within a graphical user interface. Multiple time series belonging to an entity of interest are stored in a database and the user interacts with the system to obtain a better understanding of the entity's state and behavior, entailing sequences of actions and questions whose answers may depend on previous factual or navigational interactions. We design an LSTM-based encoder-decoder architecture that models context dependency through copying mechanisms and multiple levels of attention over inputs and previous outputs. When trained to predict tokens using supervised learning, the proposed architecture substantially outperforms standard sequence generation baselines. Training the architecture using policy gradient leads to further improvements in performance, reaching a sequence-level accuracy of 88.7% on artificial data and 74.8% on real data.

1 Introduction and Motivation

Wearable sensors are being increasingly used in medicine to monitor important physiological parameters. Patients with type I diabetes, for example, wear a sensor inserted under the skin which provides measurements of the interstitial blood glucose level (BGL) every 5 minutes. Sensor bands provide a non-invasive solution to measuring additional physiological parameters, such as temperature, skin conductivity, heart rate, and acceleration of body movements. Patients may also self-report information about discrete life events such as meals, sleep, or stressful events, while an insulin pump automatically records two types of insulin interventions: a continuous stream of insulin called the basal rate, and discrete self-administered insulin dosages called boluses. The data acquired from sensors and patients accumulates rapidly and leads to a substantial data overload for the health provider.

To help doctors more easily browse the wealth of generated patient data, we built a graphical user interface (GUI) that displays the various time series of measurements corresponding to a patient. As shown in Figure 1, the GUI displays the data corresponding to one day, whereas buttons allow the user to move to the next or previous day. While the graphical interface was enthusiastically received by doctors, it soon became apparent that the doctor-GUI interaction could be improved substantially if the tool also allowed for natural language (NL) interactions. Most information needs are highly contextual and local. For example, if the blood glucose spiked after a meal, the doctor would often want to know more details about the meal or about the bolus that preceded the meal. The doctor often found it easier to express their queries in natural language (e.g. "show me how much he ate", "did he bolus before that"), resulting in a sub-optimal situation where the doctor would ask this type of *local questions* in English while a member of our team would perform the clicks required to answer the question, e.g. click on the meal event, to show details such as amount of carbohydrates. Furthermore, there were also *global questions*, such as "How often does the patient go low in the morning and the evening", whose answers would require browsing the entire patient history in the worst case, which would be very inefficient. This motivated us to start work on a new system component that would allow the doctor to interact using both natural language queries and direct actions within the GUI. A successful solution to the task described in this paper has the potential for applications in many areas of medicine where sensor data and life events are pervasive. Intelligent user interfaces for the proposed task will also benefit the exploration and interpretation of data in other domains such as experimental

Proceedings of NAACL-HLT 2019, pages 3576–3585
Minneapolis, Minnesota, June 2 - June 7, 2019.



Date: 2016-03-08 11:09
Type: Meal
Value:
Notes:
Lunch

Arbys turkey club, 1.0 whole
Mixed veggie side salad with
croutons , 1.0 whole
Carbs: 47.0

Figure 1: GUI window displaying 1 day worth of data.

physics, where large amounts of time series data are generated from high-throughput experiments.

2 Task Definition

Given an input from the user (a NL query or a direct GUI interaction), the aim is to parse it into a logical form representation that can be run by an inference engine in order to automatically extract the answer from the database. Table 1 shows sample inputs paired with their logical forms. For each input, the examples also show relevant previous inputs from the interaction sequence. In the following sections we describe a number of major features that, on their own or through their combination, distinguish this task from other semantic parsing tasks.

2.1 Time is essential

All events and measurements in the knowledge base are organized in time series. Consequently, many queries contain time expressions, such as the relative “midnight" or the coreferential “then", and temporal relations between relevant entities, expressed through words such as “after" or “when". This makes processing of temporal relations essential for a good performance. Furthermore, the GUI serves to anchor the system in time, as most of the information needs expressed in local questions are relative to the day shown in the GUI, or the last event that was clicked.

2.2 GUI interactions vs. NL questions

The user can interact with the system 1) directly within the GUI (e.g. mouse clicks); 2) through natural language questions; or 3) through a combination of both, as shown in Examples 1 and 2 in Table 1. Although the result of every direct interaction with the GUI can also be obtained using natural language questions, sometimes it can be more convenient to use the GUI directly, especially when all events of interest are in the same area of the screen and thus easy to move the mouse or hand from one to the other. For example, a doctor interested in what the patient ate that day can simply click on the blue squares at the top of the bottom pane in Figure 1, one after another. Sometimes a click can be used to anchor the system at a particular time during the day, after which the doctor can ask short questions implicitly focused on that region in time. An example of such hybrid behavior is shown in Example 2, where a click on a Bolus event is followed by a question about a snack, which implicitly should be the meal right after the bolus.

Example 1

Click on Exercise event at 9:29am.
$Click(e) \wedge e.type = \text{Exercise} \wedge e.time = \text{9:29am}$

Click on Miscellaneous event at 9:50am
$Click(e) \wedge e.type = \text{Misc} \wedge e.time = \text{9:50am}$

Q_1: What was she doing mid afternoon
when her heart rate went up?
$Answer(e) \wedge Behavior(e_1.value, \text{Up})$
$\wedge Around(e.time, e_1.time)$
$\wedge e.type == DiscreteType$
$\wedge e1.type == HeartRate$
$\wedge e1.time == MidAfternoon()$

Q_2: What time did that start?
$Answer(e(-1).time)$

Example 2

Click on Bolus at 8:03pm.
$Click(e) \wedge e.type = \text{Bolus} \wedge e.time = \text{8:03pm}$

Q_3: What did she eat for her snack?
$Answer(e.food) \wedge e.kind == Snack$

Example 3

Click on Exercise at 7:52pm.
$Click(e) \wedge e.type = \text{Exercise} \wedge e.time = \text{7:52pm}$

Q_4: What did she do then?
$Answer(e(-1).kind)$

Q_5: Did she take a bolus before then?
$Answer(Any(d.type == Bolus$
$\wedge Before(d.time, e(-1).time)))$

Example 4

Q_6: What is the first day they have heart rate reported?
$Answer(e.date)$
$\wedge Order(e, 1, Sequence(d, d.type == HeartRate))$

Example 5

Q_7: Is there another day he goes low in the morning?
$Answer(Any(Hypo(d1) \wedge x! = CurrentDate$
$\wedge x.type == Date \wedge d1.time == Morning(x))$

Table 1: Examples of interactions and logical forms.

2.3 Factual queries vs. GUI commands

Most of the time, doctors have information needs that can be satisfied by clicking on an event shown in the GUI or by asking factual questions about a particular event of interest from that day. In contrast, a different kind of interaction happens when the doctor wants to change what is shown in the tool, such as toggling on/off particular time series (e.g. “toggle on heart rate"), or navigating to a different day (e.g. “go to next day", “look at the previous day"). Sometimes, a question may be a combination of both, as in “What is the first day they have a meal without a bolus?", for which the expectation is that the system navigates to that day and also clicks on the meal event to show additional information and anchor the system at the time of that meal.

2.4 Sequential dependencies

The user interacts with the system through a sequence of questions or clicks. The logical form of a question, and implicitly its answer, may depend on the previous interaction with the system. Examples 1 to 3 in Table 1 are all of this kind. In example 1, the pronoun “that" in question 2 refers to the answer to question 1. In example 2, the snack refers to the meal around the time of the bolus event that was clicked previously – this is important, as there may be multiple snacks that day. In example 3, the adverb “then" in question 5 refers to the time of the event that is the answer of the previous question. As can be seen from these examples, sequential dependencies can be expressed as coreference between events from different questions. Coreference may also happen within questions, as in question 4 for example. Overall, solving coreferential relations will be essential for good performance.

3 Semantic Parsing Datasets

To train and evaluate semantic parsing approaches, we created two datasets of sequential interactions: a dataset of real interactions (Section 3.1) and a much larger dataset of artificial interactions (Section 3.2).

3.1 Real Interactions

We recorded interactions with the GUI in real time, using data from 9 patients, each with around 8 weeks worth of time series data. In each recording session, the tool was loaded with data from one patient and the physician was instructed to explore the data in order to understand the patient behavior as usual, by asking NL questions or interacting directly with the GUI. Whenever a question was asked, a member of our study team found the answer by navigating in and clicking on the corresponding event. After each session, the question segments were extracted manually from the speech recordings, transcribed, and timestamped. All direct interactions (e.g. mouse clicks) were recorded automatically by the tool, timestamped, and exported into an XML file. The sorted list of questions and the sorted list of mouse clicks were then merged using the timestamps as key, resulting in a chronologically sorted list of questions and GUI interactions. Mouse clicks were automatically translated into logical forms, whereas questions were parsed into logical forms manually.

Event Types
Physiological Parameters:
BGL, BasalRate, TemporaryBasal, Carbs, GSR, InfusionSet, AirTemperature, SkinTemperature, HeartRate, StepCount.
Life Events:
FingerSticks, Bolus, Hypo, HypoAction, Misc, Illness, Meal, Exercise, ReportedSleep, Wakeup, Work, Stressors.
Constants
Up, Down, On, Off, Monday, Tuesday, ..., Sunday.
Functions
$Interval(t_1, t_2)$, $Before(t)$, $After(t)$, ...
return corresponding intervals (default lengths).
$Morning([d])$, $Afternoon([d])$, $Evening([d])$, ...
return corresponding intervals for day d.
$WeekDay(d)$:
return the day of the week of date d.
$Sequence(var, statements)$:
return a chronologically ordered sequence of possible values for var that satisfy $statements$.
$Count(var[, statements])$:
returns the number of possible values for var that satisfy $statements$.
Predicates
$Answer(e)$, $Click(e)$
$Morning(t)$, $Afternoon(t)$, $Evening(t)$, ...
$Overlap(t_1, t_2)$, $Before(t_1, t_2)$, $Around(t_1, t_2)$, ...
$Behavior(variable, direction)$:
whether $variable$ increases, if $direction$ is Up, (or decrease if $direction$ is $Down$).
$High(variable)$, $Low(variable)$:
whether $variable$ has some low value.
$Order(event, ordinal, sequence[, attribute])$:
whether the $event$ is at place $ordinal$ in $sequence$
Commands
DoClick, DoToggle, DoSetDate, DoSetTime, ...

Table 2: Vocabulary for logical forms.

A snapshot of the vocabulary for logical forms is shown in Table 2, showing the Event Types, Constants, Functions, Predicates, and Commands. Every life event or physiological measurement stored in the database is represented in the logical forms as an event object e with 3 major attributes: $e.type$, $e.date$, and $e.time$. Depending on its type, an event object may contain additional fields. For example, if $e.type = BGL$, then it has an attribute $e.value$. If $e.type = Meal$, then it has attributes $e.food$ and $e.carbs$. We use $e(-i)$ to represent the event appearing in the i^{th} previous logical form (LF). Thus, to reference the event mentioned in the previous LF, we use $e(-1)$, as shown for question Q_5. If more than one event appears in the previous LF, we use an additional index j to match the event index in the previous LF. Coreference between events is represented simply using the equality operator, e.g. $e = e(-1)$.The dataset contains logical forms for 237 interactions: 74 mouse clicks and 163 NL queries.

3.2 Artificial Interactions

The number of annotated real interactions is too small for training an effective semantic parsing model. To increase the number of training examples, we designed and implemented an artificial data generator that simulates user-GUI interactions, with sentence *templates* defining the skeleton of each entry in order to maintain high-quality sentence structure and grammar. This approach is similar to (Weston et al., 2015), with the difference that we need a much higher degree of variation such that the machine learning model does not memorize all possible sentences, and consequently a much richer template database. We therefore implemented a *template language* with recursive grammar, that can be used to define as many templates and generate as many data examples as desired. We used the same vocabulary as for the real interactions dataset. To generate contextual dependencies (e.g. event coreference), the implementation allows for more complex *combo templates* where a sequence of templates are instantiated together. A more detailed description of the template language and the simulator implementation is given in (Chen et al., 2019) and Appendix A, together with illustrative examples. The simulator was used to generate 1,000 interactions and their logical forms: 312 mouse clicks and 688 NL queries.

4 Baseline Models for Semantic Parsing

This section describes two baseline models: a standard LSTM encoder-decoder for sequence generation *SeqGen* (Section 4.1) and its attention-augmented version *SeqGen+Att2In* (Section 4.2). This last model will be used later in Section 5 as a component in the context-dependent semantic parsing architecture.

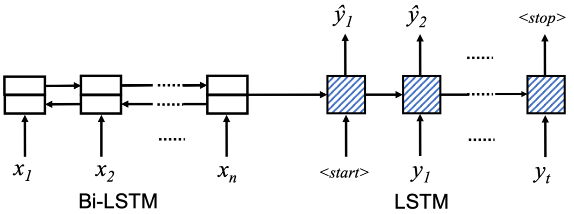


Figure 2: The *SeqGen* model takes a sequence of interactions as input $X = x_1, \ldots, x_n$ and encodes it with a Bi-LSTM (left). The decoder LSTM (right) generates a logical form $\hat{Y} = \hat{y}_1, \ldots, \hat{y}_T$.

4.1 *SeqGen*

As shown in Figure 2, the sequence-generation model *SeqGen* uses Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units in an encoder-decoder architecture (Bahdanau et al., 2017; Cho et al., 2014), composed of a bi-directional LSTM for the encoder over the input sequence X and an LSTM for the decoder of the output LF sequence Y. We use $Y_t = y_1, \ldots, y_t$ to denote the sequence of output tokens up to position t. We use $\hat{Y}$ to denote the generated logical form.

The initial state $\boldsymbol{s}_0$ is created by running the bi-LSTM encoder over the input sequence X and concatenating the last hidden states. Starting from the initial hidden state $\boldsymbol{s}_0$, the decoder produces a sequence of states $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T$, using embeddings $e(y_t)$ to represent the previous tokens in the sequence. A softmax is used to compute token probabilities at each position as follows:

$$\begin{aligned} p(y_t|Y_{t-1}, X) &= softmax(\boldsymbol{W}_h \boldsymbol{s}_t) \qquad (1) \\ \boldsymbol{s}_t &= h(\boldsymbol{s}_{t-1}, e(y_{t-1})) \end{aligned}$$

The transition function h is implemented by the LSTM unit.

4.2 *SeqGen+Att2In*

This model (Figure 3) is similar to *SeqGen*, except that it attends to the current input (NL query or mouse click) during decoding. Equation 2 defines the corresponding attention mechanism *Att2In* used to create the context vector $\boldsymbol{d}_t$:

$$e_{tj} = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a \boldsymbol{f}_j + \boldsymbol{U}_a \boldsymbol{s}_{t-1}) \qquad (2)$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{m} \exp(e_{tk})}, \quad \boldsymbol{d}_t = \boldsymbol{c}_t = \sum_{j=1}^{n} \alpha_{tj} \boldsymbol{f}_j$$

Here $\boldsymbol{f}_j$ is the j-th hidden states for Bi-LSTM corresponding to $\boldsymbol{x}_j$ and α_{tj} is an attention weight.

Both the context vector $\boldsymbol{d}_t$ and $\boldsymbol{s}_t$ are used to predict the next token $\hat{y}_t$ in the logical form:

$$\hat{y}_t \sim \text{softmax}(\boldsymbol{W}_h \boldsymbol{s}_t + \boldsymbol{W}_d \boldsymbol{d}_t)$$

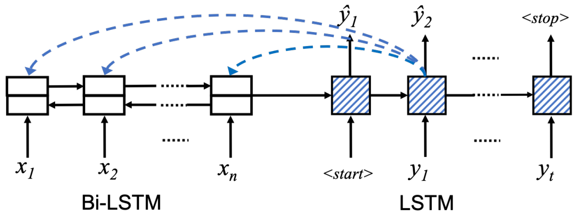


Figure 3: The *SeqGen+Att2In* model augments the *SeqGen* model with an attention mechanism. At each decoding step t, it attends to all input tokens in order to compute a context vector $\boldsymbol{d}_t$.

5 Context-Dependent Semantic Parsing

In Figure 4 we show our proposed semantic parsing model, *SP+Att2All+Copy* (*SPAAC*). Similar to the baseline models, we use a bi-directional LSTM to encode the input and another LSTM as the decoder. Context-dependency is modeled using two types of mechanisms: *attention* and *copying*. The attention mechanism (Section 5.1) is comprised of 3 models: *Att2HisIn* attending to the previous input, *Att2HisLF* attending to the previous logical form, and the *Att2In* introduced in Section 4.2 that attends to the current input. The copying mechanism (Section 5.2) is comprised of two models: one for handling unseen tokens, and one for handling coreference to events in the current and previous logical forms.

5.1 Attention Mechanisms

At decoding step t, the *Att2HisIn* attention model computes the context vector $\hat{\boldsymbol{c}}_t$ as follows:

$$\hat{e}_{tk} = \boldsymbol{v}_b^T \tanh(\boldsymbol{W}_b \boldsymbol{r}_k + \boldsymbol{U}_b \boldsymbol{s}_{t-1}) \qquad (3)$$

$$\beta_{tk} = \frac{\exp(\hat{e}_{tk})}{\sum_{l=1}^{m^2} \exp(\hat{e}_{tl})}, \quad \hat{\boldsymbol{c}}_t = \sum_{k=1}^{n} \beta_{tk} \cdot \boldsymbol{r}_k$$

where $\boldsymbol{r}_k$ is the encoder hidden state corresponding to $\boldsymbol{x}_k$ in the previous input X^{-1}, $\hat{\boldsymbol{c}}_t$ is the context vector, and β_{tk} is an attention weight.

Similarly, the *Att2HisLF* model computes the context vector $\tilde{\boldsymbol{c}}_t$ as follows:

$$\tilde{e}_{tj} = \boldsymbol{v}_c{}^T \tanh(\boldsymbol{W}_c \boldsymbol{l}_j + \boldsymbol{U}_c \boldsymbol{s}_{t-1}) \qquad (4)$$

$$\gamma_{tj} = \frac{\exp(\tilde{e}_{tj})}{\sum_{j=1}^{n} \exp(\tilde{e}_{tj})}, \quad \tilde{\boldsymbol{c}}_t = \sum_{j=1}^{n} \gamma_{tj} \cdot \boldsymbol{l}_j$$

where $\boldsymbol{l}_j$ is the j-th hidden state of the decoder for the previous logical form Y^{-1}.

The context vector used in the decoder is comprised of the context vectors from the three attention models *Att2In*, *Att2HisIn* and *Att2HisLF*:

$$\boldsymbol{d}_t = concat(\boldsymbol{c}_t, \hat{\boldsymbol{c}}_t, \tilde{\boldsymbol{c}}_t) \qquad (5)$$

5.2 Copying Mechanisms

In order to handle out-of-vocabulary (OOV) tokens and coreference (REF) between entities in the current and the previous logical forms, we add two special tokens *OOV* and *REF* to the vocabulary. Inspired by the copying mechanism in (Gu et al., 2016), we train the model to learn which token in the current input $X = \{x_j\}$ is an OOV by minimizing the following loss:

$$L_{oov}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{X.l} \log p_o(O_j | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) \qquad (6)$$

where $X.l$ is the length of current input, $Y.l$ is the length of the current logical form, $\boldsymbol{s}_j^X$ is the LSTM state for x_j and $\boldsymbol{s}_t^Y$ is the LSTM state for y_t, $O_j \in \{0, 1\}$ is a label indicating whether x_j is an OOV. We use logistic regression to compute the OOV probability, i.e. $p_o(O_j = 1 | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_o^T [\boldsymbol{s}_j^X, \boldsymbol{s}_t^Y])$.

Similarly, to solve coreference, the model is trained to learn which entity in the previously generated logical form $\hat{Y}^{-1} = \{\hat{y}_j\}$ is coreferent with the entity in the current logical form by minimizing the following loss:

$$L_{ref}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{\hat{Y}^{-1}.l} \log p_r(R_j | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) \qquad (7)$$

where $\hat{Y}^{-1}.l$ is the length of the previous generated logical form, $Y.l$ is the length of the current logical form, $\boldsymbol{s}_j^{\hat{Y}^{-1}}$ is the LSTM state at position j in $\hat{Y}^{-1}$ and $\boldsymbol{s}_t^Y$ is the LSTM state for position t in Y, and $R_j \in \{0, 1\}$ is a label indicating whether $\hat{y}_j$ is an entity referred by y_t in the next logical form Y. We use logistic regression to compute the coreference probability, i.e. $p_r(R_j = 1 | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_r^T [\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y])$.

Finally, we use "Teacher forcing" (Williams and Zipser, 1989) to train the model to learn which token in the vocabulary (including special tokens *OOV* and *REF*) should be generated, by minimizing

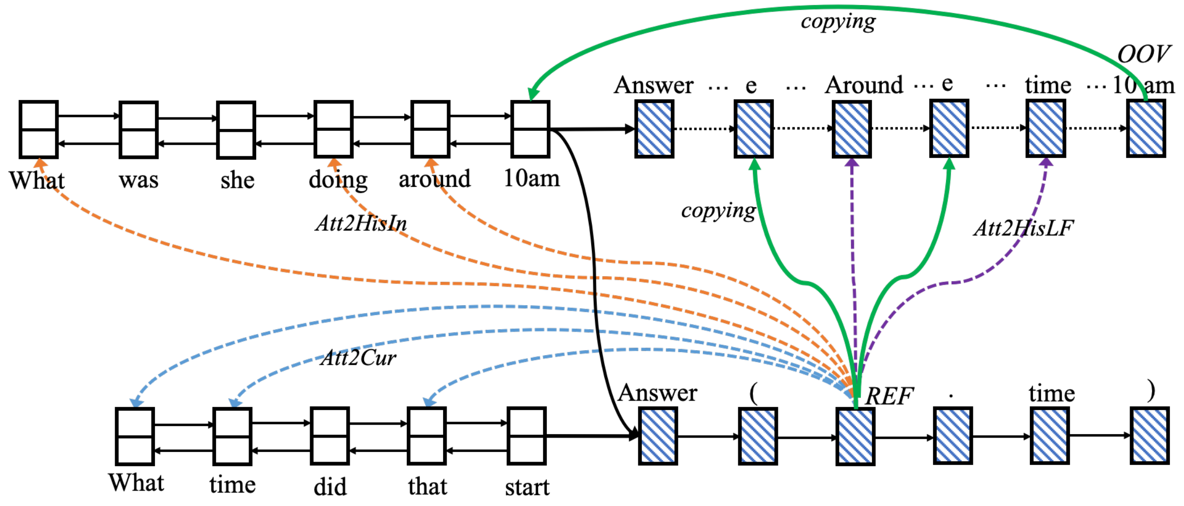


Figure 4: Context-dependent semantic parsing architecture. We use a Bi-LSTM (left) to encode the input and a LSTM (right) as the decoder. We show only parts of the LF to save space. The complete generated LF at time T-1 is Y^{-1} = [*Answer, (, e,),* $\wedge$, *Around, (, e, ., time, OOV,),* $\wedge$, *e, ., type,* ==, *DiscreteType*]. The token *10am* is copied from the input to replace the generated *OOV* token (solid green arrow). The complete generated LF at time T is Y = [*Answer, (, REF, ., time,)*]. The entity token *e* is copied from the previous LF to replace the generated *REF* token (solid green arrow). Orange dash arrows attend to historical input. Blue dash arrows attend to current input. Purple dash arrows attend to previous logical form.

the following token generation loss:

$$L_{gen}(Y) = -\sum_{t=1}^{Y.l} \log p(y_t|Y_{t-1}, X) \qquad (8)$$

where $Y.l$ is the length of the current logical form.

5.3 Supervised Learning: *SPAAC-MLE*

The supervised learning model *SPAAC-MLE* is obtained by training the semantic parsing architecture from Figure 4 to minimize the sum of the 3 negative log-likelihood losses:

$$L_{MLE}(Y) = L_{gen}(Y) + L_{oov}(Y) + L_{ref}(Y) \quad (9)$$

At inference time, beam search is used to generate the LF sequence (Ranzato et al., 2015; Wiseman and Rush, 2016). During inference, if the generated token at position t is *OOV*, we copy the token from the current input X that has the maximum OOV probability, i.e. $\arg\max_j p_o(O_j = 1|\boldsymbol{s}_j^X, \boldsymbol{s}_t^Y)$. Similarly, if the generated entity token at position t is *REF*, we copy the entity token from the previous LF Y^{-1} that has the maximum coreference probability, i.e. $\arg\max_j p_r(R_j = 1|\boldsymbol{s}_j^{Y^{-1}}, \boldsymbol{s}_t^Y)$.

5.4 Reinforcement Learning: *SPAAC-RL*

All models described in this paper are evaluated using sequence-level accuracy, a discrete metric where a generated logical form is considered to be correct if it is equivalent with the ground truth logical form. This is a strict evaluation measure in the sense that it is sufficient for a token to be wrong to invalidate the entire sequence. At the same time, there can be many generated sequences that are correct, e.g. any reordering of the clauses from the ground truth sequence is correct. The large number of potentially correct generations can lead MLE-trained models to have sub-optimal performance (Paulus et al., 2017; Rennie et al., 2017; Zeng et al., 2016; Norouzi et al., 2016). Furthermore, although "teacher forcing" (Williams and Zipser, 1989) is widely used for training sequence generation models, it leads to *exposure bias* (Ranzato et al., 2015): the network has knowledge of the ground truth LF tokens up to the current token during training, but not during testing, which can lead to propagation of errors at generation time.

Like Paulus et al. (2017), we address these problems by using policy gradient to train a token generation policy that aims to directly maximize sequence-level accuracy. We use the self-critical policy gradient training algorithm proposed by Rennie et al. (2017). We model the sequence generation process as a sequence of actions taken according to a policy, which takes an action (token $\hat{y}_t$) at each step t as a function of the current state (history $\hat{Y}_{t-1}$), according to the probability $p(\hat{y}_t|\hat{Y}_{t-1})$. The algorithm uses this probability to define two policies: a greedy, baseline policy π^b

that takes the action with the largest probability, i.e. $\pi^b(\hat{Y}_{t-1}) = \arg\max_{\hat{y}_t} p(\hat{y}_t|\hat{Y}_{t-1})$; and a sampling policy π^s that samples the action according to the same distribution, i.e. $\pi^s(\hat{Y}_{t-1}) \propto p(\hat{y}_t|\hat{Y}_{t-1})$.

The baseline policy is used to generate a sequence $\hat{Y}^b$, whereas the sampling policy is used to generate another sequence $\hat{Y}^s$. The reward $R(\hat{Y}^s)$ is then defined as the difference between the sequence-level accuracy (A) of the sampled sequence $\hat{Y}^s$ and the baseline sequence $\hat{Y}^b$. The corresponding self-critical policy gradient loss is:

$$\begin{aligned} L_{RL} &= -R(\hat{Y}^s) \times L_{MLE}(\hat{Y}^s) \\ &= -\Big(A(\hat{Y}^s) - A(\hat{Y}^b)\Big) \times L_{MLE}(\hat{Y}^s) \end{aligned} \quad (10)$$

Thus, minimizing the RL loss is equivalent to maximizing the likelihood of the sampled $\hat{Y}^s$ if it obtains a higher sequence-level accuracy than the baseline $\hat{Y}^b$.

6 Experimental Evaluation

All models are implemented in Tensorflow using dropout to deal with overfitting. For both datasets, 10% of the data is put aside for validation. After tuning on the artificial validation data, the feed-forward neural networks dropout rate was set to 0.5 and the LSTM units dropout rate was set to 0.3. The word embeddings had dimensionality of 64 and were initialized at random. Optimization is performed with the Adam algorithm. For each dataset, we use five-fold cross evaluation, where the data is partitioned into five folds, one fold is used for testing and the other folds for training. The process is repeated five times to obtain test results on all folds. We use an early-stop strategy on the validation set. The number of gradient updates is typically more than 20,000. All the experiments are performed on a single NVIDIA GTX1080 GPU.

The models are trained and evaluated on the artificial interactions first. To evaluate on real interactions, the models are pre-trained on the entire artificial dataset and then fine-tuned using real interactions. *SPAAC-RL* is pre-trained with MLE loss to provide more efficient policy exploration. We use sequence level accuracy as evaluation metric for all models: a generated sequence is considered correct if and only if all the generated tokens match the ground truth tokens.

We report experimental evaluations of the proposed models *SPAAC-MLE* and *SPAAC-RL* and baseline models *SeqGen*, *SeqGen+Att2In* on the Real and Artificial Interactions Datasets in Table 3. We also report examples generated by the SPAAC models in Tables 4 and 5.

Models	Artificial	Real
SeqGen	51.8	22.2
SeqGen+Att2In	72.7	35.4
SPAAC-MLE	**84.3**	**66.9**
SPAAC-RL	**88.7**	**74.8**

Table 3: Sequence-level accuracy on the 2 datasets.

Well the Finger Stick is 56. T&MLE&RL: $e.type == Fingerstick \wedge e.value == 56$ It looks like she suspended her pump. T&MLE&RL: $Suspended(e) \wedge around(e.time, e(-1).time)$
Let's look at the next day. T&MLE&RL: $DoSetDate(currentdate + 1)$
See if he went low. T&MLE&RL: $Answer(any(e, hypo(e)))$
Let's see what kind of exercise that is, where the steps are high? T&RL:$Answer(e.kind) \wedge e.type == exercise$ $\wedge around(e.time, e_1.time) \wedge e_1.type == \boldsymbol{stepcount}$ $\wedge \boldsymbol{high(e_1.value)}$ MLE: $Answer(e.kind) \wedge e.type == exercise$ $\wedge around(e.time, e_1.time) \wedge e_1.type == \boldsymbol{exercise}$ $\wedge \boldsymbol{e_1.type == exercise}$
Click on the exercise. T&RL:$\boldsymbol{DoClick}(e) \wedge e.type == exercise$ MLE: $\boldsymbol{Answer}(e) \wedge e.type == exercise$

Table 4: Examples generated by *SPAAC-MLE* and *SPAAC-RL* using real interactions. T: true logical forms. MLE: logical forms by *SPAAC-MLE*. RL: logical forms by *SPAAC-RL*.

6.1 Discussion

The results in Table 3 demonstrate the importance of modeling context-dependency, as the two *SPAAC* models outperform the baselines on both datasets. The RL model also obtains substantially better accuracy than the MLE model. The improvement in performance over the MLE model for the real data is statistically significant at $p = 0.05$ in a one-tailed paired t-test.

Analysis of the generated logical forms revealed that one common error made by *SPAAC-MLE* is the generation of incorrect event types. Some of these errors are fixed by the current RL model. However, there are instances where even the RL-trained model outputs the wrong event type. By comparing

Does he always get some sleep around 4:30pm? T&MLE&RL: $Answer(cond(around(x, 4:30pm) => any(e.type == reportedsleep \wedge e.time == x)))$
Is it the first week of the patient? T&MLE&RL: $Answer(week(currentdate) == x) \wedge order(x, 1, sequence(e, e.type == week))$
Does she ever get some rest around 5:37pm? T&MLE&RL: $Answer(any(e.type == reportedsleep \wedge around(e.time, 5:37pm)))$
When is the first time he changes his infusion set? T&MLE&RL: $Answer(e.date) \wedge order(e, 1, sequence(e, e.type == infusionset))$
How many months she has multiple exercises? T&RL: $Answer(count(x, count(e, e.type == exercise \wedge e.date == x) > 1 \wedge x.type == \boldsymbol{month}))$ MLE: $Answer(count(x, count(e, e.type == exercise \wedge e.date == x) > 1 \wedge x.type == \boldsymbol{week}))$
Toggle so we can see fingersticks. T&RL: $DoToggle(on, \boldsymbol{fingersticks})$ MLE: $DoToggle(on, \boldsymbol{bgl})$

Table 5: Examples generated by *SPAAC-MLE* and *SPAAC-RL* using artificial interactions. T: true logical forms. MLE: logical forms generated by *SPAAC-MLE*. RL: logical forms generated by *SPAAC-RL*.

the sampled logical forms $\hat{Y}^s$ and the generated baseline logical forms $\hat{Y}^b$, we found that sometimes the sampled tokens for event types are the same as those in the baseline. An approach that we plan to investigate in future work is to utilize more advanced sampling methods to generate $\hat{Y}^s$, in order to achieve a better balance between exploration and exploitation.

7 Related Work

Question Answering has been the topic of recent research (Yih et al., 2014; Dong et al., 2015; Andreas et al., 2016; Hao et al., 2017; Abujabal et al., 2017; Chen and Bunescu, 2017). Semantic parsing, which maps text in natural language to meaning representations in formal logic, has emerged as an important component for building QA systems, as in (Liang, 2016; Jia and Liang, 2016a; Zhong et al., 2017). Context-dependent processing has been explored in complex, interactive QA (Harabagiu et al., 2005; Kelly and Lin, 2007) and semantic parsing (Zettlemoyer and Collins, 2009; Artzi and Zettlemoyer, 2011; Iyyer et al., 2017; Suhr et al., 2018; Long et al., 2016). Although these approaches take into account sequential dependencies between questions or sentences, the setting in our work has a number of significant distinguishing features, such as the importance of time – data is represented naturally as multiple time series of events – and the anchoring on a graphical user interface that also enables direct interactions through mouse clicks and a combination of factual queries and interface commands.

Dong and Lapata (2016) use an attention-enhanced encoder-decoder architecture to learn the logical forms from natural language without using hand-engineered features. Their proposed Seq2Tree architecture can capture the hierarchical structure of logical forms. Jia and Liang (2016b) train a sequence-to-sequence RNN model with a novel attention-based copying mechanism to learn the logical forms from questions. The copying mechanism has been investigated by Gu et al. (2016) and Gulcehre et al. (2016) in the context of a wide range of NLP applications. These semantic parsing models considered sentences in isolation. In contrast, generating correct logical forms in our task required modeling sequential dependencies between logical forms. In particular, coreference is modeled between events mentioned in different logical forms by repurposing the copying mechanism originally used for modeling out-of-vocabulary tokens.

8 Conclusion

We introduced a new semantic parsing setting in which users can query a system using both natural language and direct interactions (mouse clicks) within a graphical user interface. Correspondingly, we created a dataset of real interactions and a much larger dataset of artificial interactions. The correct interpretation of a natural language query often requires knowledge of previous interactions with the system. We proposed a new sequence generation architecture that modeled this context dependency through multiple attention models and a copying mechanism for solving coreference. The proposed architecture is shown to outperform standard LSTM encoder-decoder architectures that are context agnostic. Furthermore, casting the sequence generation process in the framework of reinforcement learning alleviates the exposure bias and leads to substantial improvements in sequence-level accuracy.

The two datasets and the implementation of the systems presented in this paper are made publicly available at `https://github.com/charleschen1015/SemanticParsing`. The data visualization GUI is available under the name OHIOT1DMVIEWER at `http://smarthealth.cs.ohio.edu/nih.html`.

Acknowledgments

This work was partly supported by grant 1R21EB022356 from the National Institutes of Health. We would like to thank Frank Schwartz and Cindy Marling for contributing real interactions, Quintin Fettes and Yi Yu for their help with recording and pre-processing the interactions, and Sadegh Mirshekarian for the design of the artificial data generation. We would also like to thank the anonymous reviewers for their useful comments.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. Quint: Interpretable question answering over knowledge bases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–66.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*, pages 421–432. Association for Computational Linguistics.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Charles Chen and Razvan Bunescu. 2017. An exploration of data augmentation and rnn architectures for question ranking in community question answering. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 442–447.

Charles Chen, Sadegh Mirshekarian, Razvan Bunescu, and Cindy Marling. 2019. From physician queries to logical forms for efficient exploration of patient data. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 371–374. IEEE.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan. 2005. Experiments with interactive question-answering. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 205–214. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

Robin Jia and Percy Liang. 2016a. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016b. Data recombination for neural semantic parsing. In *Proceedings of* *the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 12–22.

Diane Kelly and Jimmy Lin. 2007. Overview of the TREC 2006 ciQA task. In *ACM SIGIR Forum*, volume 41, pages 107–116. ACM.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1456–1465.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195. IEEE.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2238–2249.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 643–648.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Structural Scaffolds for Citation Intent Classification in Scientific Publications

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, Field Cady
Allen Institute for Artificial Intelligence
`{armanc,waleeda,madeleinev,fieldc}@allenai.org`

Abstract

Identifying the intent of a citation in scientific papers (e.g., *background information, use of methods, comparing results*) is critical for machine reading of individual publications and automated analysis of the scientific literature. We propose structural scaffolds, a multitask model to incorporate structural information of scientific papers into citations for effective classification of citation intents. Our model achieves a new state-of-the-art on an existing ACL anthology dataset (ACL-ARC) with a 13.3% absolute increase in F1 score, without relying on external linguistic resources or hand-engineered features as done in existing methods. In addition, we introduce a new dataset of citation intents (SciCite) which is more than five times larger and covers multiple scientific domains compared with existing datasets. Our code and data are available at: `https://github.com/allenai/scicite`.

1 Introduction

Citations play a unique role in scientific discourse and are crucial for understanding and analyzing scientific work (Luukkonen, 1992; Leydesdorff, 1998). They are also typically used as the main measure for assessing impact of scientific publications, venues, and researchers (Li and Ho, 2008). The nature of citations can be different. Some citations indicate direct use of a method while some others merely serve as acknowledging a prior work. Therefore, identifying the intent of citations (Figure 1) is critical in improving automated analysis of academic literature and scientific impact measurement (Leydesdorff, 1998; Small, 2018). Other applications of citation intent classification are enhanced research experience (Moravcsik and Murugesan, 1975), information retrieval (Ritchie, 2009), summarization (Cohan and Goharian, 2015), and studying evolution of scientific fields (Jurgens et al., 2018).

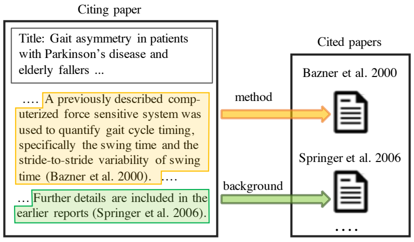


Figure 1: Example of citations with different intents (BACKGROUND and METHOD).

In this work, we approach the problem of citation intent classification by modeling the language expressed in the citation context. A citation context includes text spans in a citing paper describing a referenced work and has been shown to be the primary signal in intent classification (Teufel et al., 2006; Abu-Jbara et al., 2013; Jurgens et al., 2018). Existing models for this problem are feature-based, modeling the citation context with respect to a set of predefined hand-engineered features (such as linguistic patterns or cue phrases) and ignoring other signals that could improve prediction.

In this paper we argue that better representations can be obtained directly from data, sidestepping problems associated with external features. To this end, we propose a neural multitask learning framework to incorporate knowledge into citations from the structure of scientific papers. In particular, we propose two auxiliary tasks as *structural scaffolds* to improve citation intent prediction:[1] (1) predicting the section title in which the citation occurs and (2) predicting whether a sentence needs a citation. Unlike the primary task of citation intent prediction, it is easy to collect large

---

[1] We borrow the scaffold terminology from Swayamdipta et al. (2018) in the context of multitask learning.

Proceedings of NAACL-HLT 2019, pages 3586–3596
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

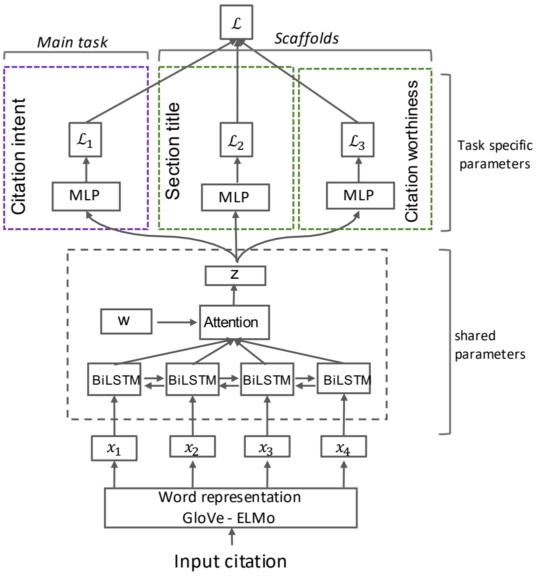


Figure 2: Our proposed scaffold model for identifying citation intents. The main task is predicting the citation intent (top left) and two scaffolds are predicting the section title and predicting if a sentence needs a citation (citation worthiness).

amounts of training data for scaffold tasks since the labels naturally occur in the process of writing a paper and thus, there is no need for manual annotation. On two datasets, we show that the proposed neural scaffold model outperforms existing methods by large margins.

Our contributions are: *(i)* we propose a neural scaffold framework for citation intent classification to incorporate into citations knowledge from structure of scientific papers; *(ii)* we achieve a new state-of-the-art of 67.9% F1 on the ACL-ARC citations benchmark, an absolute 13.3% increase over the previous state-of-the-art (Jurgens et al., 2018); and *(iii)* we introduce SciCite, a new dataset of citation intents which is at least five times as large as existing datasets and covers a variety of scientific domains.

2 Model

We propose a neural multitask learning framework for classification of citation intents. In particular, we introduce and use two structural scaffolds, auxiliary tasks related to the structure of scientific papers. The auxiliary tasks may not be of interest by themselves but are used to inform the main task. Our model uses a large auxiliary dataset to incorporate this structural information available in scientific documents into the citation intents. The overview of our model is illustrated in Figure 2.

Let C denote the citation and $\mathbf{x}$ denote the citation context relevant to C. We encode the tokens in the citation context of size n as $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathcal{R}^{d_1}$ is a word vector of size d_1 which concatenates non-contextualized word representations (GloVe, Pennington et al., 2014) and contextualized embeddings (ELMo, Peters et al., 2018), i.e.:

$$\mathbf{x}_i = \left[\mathbf{x}_i^{\text{GloVe}}; \mathbf{x}_i^{\text{ELMo}}\right]$$

We then use a bidirectional long short-term memory (Hochreiter and Schmidhuber, 1997) (BiLSTM) network with hidden size of d_2 to obtain a contextual representation of each token vector with respect to the entire sequence:[2]

$$\mathbf{h}_i = \left[\overrightarrow{\text{LSTM}}(\mathbf{x}, i); \overleftarrow{\text{LSTM}}(\mathbf{x}, i)\right],$$

where $\mathbf{h} \in \mathcal{R}^{(n, 2d_2)}$ and $\overrightarrow{\text{LSTM}}(\mathbf{x}, i)$ processes $\mathbf{x}$ from left to write and returns the LSTM hidden state at position i (and vice versa for the backward direction $\overleftarrow{\text{LSTM}}$). We then use an attention mechanism to get a single vector representing the whole input sequence:

$$\mathbf{z} = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i, \quad \alpha_i = \text{softmax}(\mathbf{w}^\top \mathbf{h}_i),$$

where $\mathbf{w}$ is a parameter served as the query vector for dot-product attention.[3] So far we have obtained the citation representation as a vector $\mathbf{z}$. Next, we describe our two proposed structural scaffolds for citation intent prediction.

2.1 Structural scaffolds

In scientific writing there is a connection between the structure of scientific papers and the intent of citations. To leverage this connection for more effective classification of citation intents, we propose a multitask framework with two structural scaffolds (auxiliary tasks) related to the structure of scientific documents. A key point for our proposed scaffolds is that they do not need any additional manual annotation as labels for these tasks occur naturally in scientific writing. The structural scaffolds in our model are the following:

---

[2] In our experiments BiGRUs resulted in similar performance.

[3] We also experimented BiLSTMs without attention; we found that BiLSTMs/BiGRUs along with attention provided best results. Other types of attention such as additive attention result in similar performance.

Citation worthiness. The first scaffold task that we consider is "citation worthiness" of a sentence, indicating whether a sentence needs a citation. The language expressed in citation sentences is likely distinctive from regular sentences in scientific writing, and such information could also be useful for better language modeling of the citation contexts. To this end, using citation markers such as "[12]" or "Lee et al (2010)", we identify sentences in a paper that include citations and the negative samples are sentences without citation markers. The goal of the model for this task is to predict whether a particular sentence needs a citation.[4]

Section title. The second scaffold task relates to predicting the section title in which a citation appears. Scientific documents follow a standard structure where the authors typically first introduce the problem, describe methodology, share results, discuss findings and conclude the paper. The intent of a citation could be relevant to the section of the paper in which the citation appears. For example, method-related citations are more likely to appear in the methods section. Therefore, we use the section title prediction as a scaffold for predicting citation intents. Note that this scaffold task is different than simply adding section title as an additional feature in the input. We are using the section titles from a larger set of data than training data for the main task as a proxy to learn linguistic patterns that are helpful for citation intents. In particular, we leverage a large number of scientific papers for which the section information is known for each citation to automatically generate large amounts of training data for this scaffold task.[5]

Multitask formulation. Multitask learning as defined by Caruana (1997) is an approach to inductive transfer learning that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It requires the model to have at least some sharable parameters between the tasks. In a general setting in our model, we have a main task $Task^{(1)}$ and $n-1$ auxiliary tasks $Task^{(i)}$. As shown in Figure 2, each scaffold task will have its task-specific parameters for effective classification and the parameters for the lower layers of the network are shared across tasks. We use a Multi Layer Perceptron (MLP) for each task and then a softmax layer to obtain prediction probabilites. In particular, given the vector $\mathbf{z}$ we pass it to n MLPs and obtain n output vectors $\mathbf{y}^{(i)}$:

$$\mathbf{y}^{(i)} = \mathrm{softmax}(\mathrm{MLP}^{(i)}(\mathbf{z}))$$

We are only interested in the output $\mathbf{y}^{(1)}$ and the rest of outputs $(\mathbf{y}^{(2)}, ..., \mathbf{y}^{(n)})$ are regarding the scaffold tasks and only used in training to inform the model of knowledge in the structure of the scientific documents. For each task, we output the class with the highest probability in $\mathbf{y}$. An alternative inference method is to sample from the output distribution.

2.2 Training

Let $\mathcal{D}_1$ be the labeled dataset for the main task $Task^{(1)}$, and $\mathcal{D}_i$ denote the labeled datasets corresponding to the scaffold task $Task^{(i)}$ where $i \in \{2, ..., n\}$. Similarly, let $\mathcal{L}_1$ and $\mathcal{L}_i$ be the main loss and the loss of the auxiliary task i, respectively. The final loss of the model is:

$$\mathcal{L} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_1} \mathcal{L}_1(\mathbf{x},\mathbf{y}) + \sum_{i=2}^{n} \lambda_i \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_i} \mathcal{L}_i(\mathbf{x},\mathbf{y}), \quad (1)$$

where λ_i is a hyper-parameter specifying the sensitivity of the parameters of the model to each specific task. Here we have two scaffold tasks and hence n=3. λ_i could be tuned based on performance on validation set (see §4 for details).

We train this model jointly across tasks and in an end-to-end fashion. In each training epoch, we construct mini-batches with the same number of instances from each of the n tasks. We compute the total loss for each mini-batch as described in Equation 1, where $\mathcal{L}_i$=0 for all instances of other tasks $j{\neq}i$. We compute the gradient of the loss for each mini-batch and tune model parameters using the AdaDelta optimizer (Zeiler, 2012) with gradient clipping threshold of 5.0. We stop training the model when the development macro F1 score does not improve for five consecutive epochs.

3 Data

We compare our results on two datasets from different scientific domains. While there has been a long history of studying citation intents, there are only a few existing publicly available datasets on

---

[4] We note that this task may also be useful for helping authors improve their paper drafts. However, this is not the focus of this work.

[5] We also experimented with adding section titles as additional feature to the input, however, it did not result in any improvements.

Intent cateogry	Definition	Example
Background information	The citation states, mentions, or points to the background information giving more context about a problem, concept, approach, topic, or importance of the problem in the field.	Recent evidence suggests that co-occurring alexithymia may explain deficits [12]. Locally high-temperature melting regions can act as permanent termination sites [6-9]. One line of work is focused on changing the objective function (Mao et al., 2016).
Method	Making use of a method, tool, approach or dataset	Fold differences were calculated by a mathematical model described in [4]. We use Orthogonal Initialization (Saxe et al., 2014)
Result comparison	Comparison of the paper's results/findings with the results/findings of other work	Weighted measurements were superior to T2-weighted contrast imaging which was in accordance with former studies [25-27] Similar results to our study were reported in the study of Lee et al (2010).

Table 1: The definition and examples of citation intent categories in our SciCite.

Dataset	Categories (distribution)	Source	#papers	#instances
ACL-ARC	Background (0.51) Extends (0.04) Uses (0.19) Motivation (0.05) Compare/Contrast (0.18) Future work (0.04)	Computational Linguistics	186	1,941
SciCite	Background (0.58) Method (0.29) Result comparison (0.13)	Computer Science & Medicine	6,627	11,020

Table 2: Characteristics of SciCite compared with ACL-ARC dataset by Jurgens et al. (2018)

the task of citation intent classification. We use the most recent and comprehensive (ACL-ARC citations dataset) by Jurgens et al. (2018) as a benchmark dataset to compare the performance of our model to previous work. In addition, to address the limited scope and size of this dataset, we introduce SciCite, a new dataset of citation intents that addresses multiple scientific domains and is more than five times larger than ACL-ARC. Below is a description of both datasets.

3.1 ACL-ARC citations dataset

ACL-ARC is a dataset of citation intents released by Jurgens et al. (2018). The dataset is based on a sample of papers from the ACL Anthology Reference Corpus (Bird et al., 2008) and includes 1,941 citation instances from 186 papers and is annotated by domain experts in the NLP field. The data was split into three standard stratified sets of train, validation, and test with 85% of data used for training and remaining 15% divided equally for validation and test. Each citation unit includes information about the immediate citation context, surrounding context, as well as information about the citing and cited paper. The data includes six intent categories outlined in Table 2.

3.2 SciCite dataset

Most existing datasets contain citation categories that are too fine-grained. Some of these intent categories are very rare or not useful in meta analysis of scientific publications. Since some of these fine-grained categories only cover a minimal percentage of all citations, it is difficult to use them to gain insights or draw conclusions on impacts of papers. Furthermore, these datasets are usually domain-specific and are relatively small (less than 2,000 annotated citations).

To address these limitations, we introduce SciCite, a new dataset of citation intents that is significantly larger, more coarse-grained and general-domain compared with existing datasets. Through examination of citation intents, we found out many of the categories defined in previous work such as motivation, extension or future work, can be considered as background information providing more context for the current research topic. More interesting intent categories are a direct use of a method or comparison of results. Therefore, our dataset provides a concise annotation scheme that is useful for navigating research topics and machine reading of scientific papers. We consider three intent categories outlined in Table 1: BACKGROUND, METHOD and RESULTCOMPARISON. Below we describe data collection and annotation details.

3.2.1 Data collection and annotation

Citation intent of sentence extractions was labeled through the crowdsourcing platform Figure Eight.[6] We selected a sample of papers from the Semantic Scholar corpus,[7] consisting of papers in general computer science and medicine domains. Citation contexts were extracted using science-

---

[6] `https://www.figure-eight.com/platform/`

[7] `https://semanticscholar.org/`

parse.[8] The annotators were asked to identify the intent of a citation, and were directed to select among three citation intent options: METHOD, RESULTCOMPARISON and BACKGROUND. The annotation interface also included a dummy option OTHER which helps improve the quality of annotations of other categories. We later removed instances annotated with the OTHER option from our dataset (less than 1% of the annotated data), many of which were due to citation contexts which are incomplete or too short for the annotator to infer the citation intent.

We used 50 test questions annotated by a domain expert to ensure crowdsource workers were following directions and disqualify annotators with accuracy less than 75%. Furthermore, crowdsource workers were required to remain on the annotation page (five annotations) for at least ten seconds before proceeding to the next page. Annotations were dynamically collected. The annotations were aggregated along with a confidence score describing the level of agreement between multiple crowdsource workers. The confidence score is the agreement on a single instance weighted by a trust score (accuracy of the annotator on the initial 50 test questions).

To only collect high quality annotations, instances with confidence score of $\leq$0.7 were discarded. In addition, a subset of the dataset with 100 samples was re-annotated by a trained, expert annotator to check for quality, and the agreement rate with crowdsource workers was **86%**. Citation contexts were annotated by 850 crowdsource workers who made a total of 29,926 annotations and individually made between 4 and 240 annotations. Each sentence was annotated, on average, 3.74 times. This resulted in a total 9,159 crowdsourced instances which were divided to training and validation sets with 90% of the data used for the training set. In addition to the crowdsourced data, a separate test set of size 1,861 was annotated by a trained, expert annotator to ensure high quality of the dataset.

3.3 Data for scaffold tasks

For the first scaffold (citation worthiness), we sample sentences from papers and consider the sentences with citations as positive labels. We also remove the citation markers from those sentences such as numbered citations (e.g., [1]) or name-year combinations (e.g, Lee et al (2012)) to not make the second task artificially easy by only detecting citation markers. For the second scaffold (citation section title), respective to each test dataset, we sample citations from the ACL-ARC corpus and Semantic Scholar corpus[9] and extract the citation context as well as their corresponding sections. We manually define regular expression patterns mappings to normalized section titles: "introduction", "related work", "method", "experiments", "conclusion". Section titles which did not map to any of the aforementioned titles were excluded from the dataset. Overall, the size of the data for scaffold tasks on the ACL-ARC dataset is about 47K (section title scaffold) and 50K (citation worthiness) while on SciCite is about 91K and 73K for section title and citation worthiness scaffolds, respectively.

4 Experiments

4.1 Implementation

We implement our proposed scaffold framework using the AllenNLP library (Gardner et al., 2018). For word representations, we use 100-dimensional GloVe vectors (Pennington et al., 2014) trained on a corpus of 6B tokens from Wikipedia and Gigaword. For contextual representations, we use ELMo vectors released by Peters et al. (2018)[10] with output dimension size of 1,024 which have been trained on a dataset of 5.5B tokens. We use a single-layer BiLSTM with a hidden dimension size of 50 for each direction[11]. For each of scaffold tasks, we use a single-layer MLP with 20 hidden nodes , ReLU (Nair and Hinton, 2010) activation and a Dropout rate (Srivastava et al., 2014) of 0.2 between the hidden and input layers. The hyperparameters λ_i are tuned for best performance on the validation set of the respective datasets using a 0.0 to 0.3 grid search. For example, the following hyperparameters are used for the ACL-ARC. Citation worthiness saffold: λ_2=0.08, λ_3=0, section title scaffold: λ_3=0.09, λ_2=0; both scaffolds: λ_2=0.1, λ_3=0.05. Batch size is 8 for ACL-ARC dataset and 32 for SciCite dataset (recall that SciCite is larger than ACL-ARC). We

---

[8] `https://github.com/allenai/science-parse`

[9] `https://semanticscholar.org/`

[10] `https://allennlp.org/elmo`

[11] Experiments with other types of RNNs such as BiGRUs and more layers showed similar or slightly worst performance

use Beaker[12] for running the experiments. On the smaller dataset, our best model takes approximately 30 minutes per epoch to train (training time without ELMo is significantly faster). It is known that multiple runs of probabilistic deep learning models can have variance in overall scores (Reimers and Gurevych, 2017)[13]. We control this by setting random-number generator seeds; the reported overall results are average of multiple runs with different random seeds. To facilitate reproducibility, we release our code, data, and trained models.[14]

4.2 Baselines

We compare our results to several baselines including the model with state-of-the-art performance on the ACL-ARC dataset.

- *BiLSTM Attention (with and without ELMo).* This baseline uses a similar architecture to our proposed neural multitask learning framework, except that it only optimizes the network for the main loss regarding the citation intent classification ($\mathcal{L}_1$) and does not include the structural scaffolds. We experiment with two variants of this model: with and without using the contextualized word vector representations (ELMo) of Peters et al. (2018). This baseline is useful for evaluating the effect of adding scaffolds in controlled experiments.

- *Jurgens et al. (2018).* To make sure our results are competitive with state-of-the-art results on this task, we also compare our model to Jurgens et al. (2018) which has the best reported results on the ACL-ARC dataset. Jurgens et al. (2018) incorporate a variety of features, ranging from pattern-based features to topic-modeling features, to citation graph features. They also incorporate section titles and relative section position in the paper as features. Our implementation of this model achieves a macro-averaged F1 score of 0.526 using 10-fold cross-validation, which is in line with the highest reported results in Jurgens et al. (2018): 0.53 using leave-one-out cross validation. We were not able to use leave-one-out cross validation in our experiments since it is impractical to re-train each variant of our deep learning models thousands of times. Therefore, we opted for a standard setup of stratified train/validation/test data splits with 85% data used for training and the rest equally split between validation and test.

	Model	macro F1
Baselines	BiLSTM-Attn	51.8
	BiLSTM-Attn w/ ELMo	54.3
	Previous SOTA (Jurgens et al., 2018)	54.6
This work	BiLSTM-Attn + section title scaffold	56.9
	BiLSTM-Attn + citation worthiness scaffold	56.3
	BiLSTM-Attn + both scaffolds	63.1
	BiLSTM-Attn w/ ELMo + both scaffolds	**67.9**

Table 3: Results on the ACL-ARC citations dataset.

4.3 Results

Our main results for the ACL-ARC dataset (Jurgens et al., 2018) is shown in Table 3. We observe that our scaffold-enhanced models achieve clear improvements over the state-of-the-art approach on this task. Starting with the 'BiLSTM-Attn' baseline with a macro F1 score of 51.8, adding the first scaffold task in 'BiLSTM-Attn + section title scaffold' improves the F1 score to 56.9 (Δ=5.1). Adding the second scaffold in 'BiLSTM-Attn + citation worthiness scaffold' also results in similar improvements: 56.3 (Δ=4.5). When both scaffolds are used simultaneously in 'BiLSTM-Attn + both scaffolds', the F1 score further improves to 63.1 (Δ=11.3), suggesting that the two tasks provide complementary signal that is useful for citation intent prediction.

The best result is achieved when we also add ELMo vectors (Peters et al., 2018) to the input representations in 'BiLSTM-Attn w/ ELMo + both scaffolds', achieving an F1 of 67.9, a major improvement from the previous state-of-the-art results of Jurgens et al. (2018) 54.6 (Δ=13.3). We note that the scaffold tasks provide major contributions on top of the ELMo-enabled baseline (Δ=13.6), demonstrating the efficacy of using structural scaffolds for citation intent prediction. We note that these results were obtained without using hand-curated features or additional linguistic resources as used in Jurgens et al. (2018). We also experimented with adding features used in Jurgens et al. (2018) to our best model and not only we did not see any improvements, but we observed

---

[12] Beaker is a collaborative platform for reproducible research (`https://github.com/allenai/beaker`)

[13] Some CuDNN methods are non-deterministic and the rest are only deterministic under the same underlying hardware. See `https://docs.nvidia.com/deeplearning/sdk/pdf/cuDNN-Developer-Guide.pdf`

[14] `https://github.com/allenai/scicite`

	Model	macro F1
Baselines	BiLSTM-Attn	77.2
	BiLSTM-Attn w/ ELMo	82.6
	Previous SOTA (Jurgens et al., 2018)	79.6
This work	BiLSTM-Attn + section title scaffold	77.8
	BiLSTM-Attn + citation worthiness scaffold	78.1
	BiLSTM-Attn + both scaffolds	79.1
	BiLSTM-Attn w/ ELMo + both scaffolds	**84.0**

Table 4: Results on the SciCite dataset.

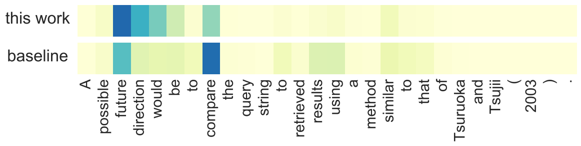
at least 1.7% decline in performance. This suggests that these additional manual features do not provide the model with any additional useful signals beyond what the model already learns from the data.

Table 4 shows the main results on SciCite dataset, where we see similar patterns. Each scaffold task improves model performance. Adding both scaffolds results in further improvements. And the best results are obtained by using ELMo representation in addition to both scaffolds. Note that this dataset is more than five times larger in size than the ACL-ARC, therefore the performance numbers are generally higher and the F1 gains are generally smaller since it is easier for the models to learn optimal parameters utilizing the larger annotated data. On this dataset, the best baseline is the neural baseline with addition of ELMo contextual vectors achieving an F1 score of 82.6 followed by Jurgens et al. (2018), which is expected because neural models generally achieve higher gains when more training data is available and because Jurgens et al. (2018) was not designed with the SciCite dataset in mind.

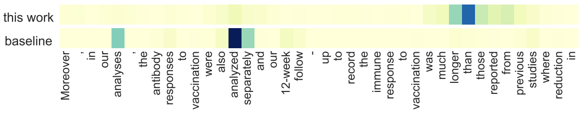
The breakdown of results by intent on ACL-ARC and SciCite datasets is respectively shown in Tables 5 and 6. Generally we observe that results on categories with more number of instances are higher. For example on ACL-ARC, the results on the Background category are the highest as this category is the most common. Conversely, the results on the FutureWork category are the lowest. This category has the fewest data points (see distribution of the categories in Table 2) and thus it is harder for the model to learn the optimal parameters for correct classification in this category.

4.4 Analysis

To gain more insight into why the scaffolds are helping the model in improved citation intent classification, we examine the attention weights assigned to inputs for our best proposed model



(a) Example from ACL-ARC: Correct label is FutureWork. Our model correctly predicts it while baseline predicts Compare.



(b) Example from SciCite: Correct label is ResultComparison; our model correctly predicts it, while baseline considers it as Background.

Figure 3: Visualization of attention weights corresponding to our best scaffold model compared with the best baseline neural baseline model without scaffolds.

('BiLSTM-Attn w/ ELMo + both scaffolds') compared with the best neural baseline ('BiLSTM-Attn w/ ELMO'). We conduct this analysis for examples from both datasets. Figure 3 shows an example input citation along with the horizontal line and the heatmap of attention weights for this input resulting from our model versus the baseline. For first example (3a) the true label is FutureWork. We observe that our model puts more weight on words surrounding the word "future" which is plausible given the true label. On the other hand, the baseline model attends most to the words "compare" and consequently incorrectly predicts a Compare label. In second example (3b) the true label is ResultComparison. The baseline incorrectly classifies it as a Background, likely due to attending to another part of the sentence ("analyzed seprately"). Our model correctly classifies this instance by putting more attention weights on words that relate to comparison of the results. This suggests that the our model is more successful in learning optimal parameters for representing the citation text and classifying its respective intent compared with the baseline. Note that the only difference between our model and the neural baseline is inclusion of the structural scaffolds. Therefore, suggesting the effectiveness the scaffolds in informing the main task of relevant signals for citation intent classification.

Error analysis. We next investigate errors made by our best model (Figure 4 plots classification errors). One general error pattern is that the model has more tendency to make false positive errors in the Background category likely due to this category dominating both datasets. It's interesting that for the ACL-ARC dataset some prediction

Category (# instances)	Background (71)			Compare (25)			Extension (5)			Future (5)			Motivation (7)			Use (26)			Average (Macro)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BiLSTM-Attn	78.6	77.5	78.0	44.8	52.0	48.1	50.0	40.0	44.4	33.3	40.0	36.4	50.0	28.6	36.4	65.4	65.4	65.4	53.7	50.6	51.5
BiLSTM-Attn w/ ELMo	76.5	87.3	81.6	59.1	52.0	55.3	66.7	40.0	50.0	33.3	40.0	36.4	50.0	28.6	36.4	69.6	61.5	65.3	59.2	51.6	54.2
Previous SOTA (Jurgens et al., 2018)	75.6	87.3	81.1	70.6	48.0	57.1	66.7	40.0	50.0	50.0	20.0	28.6	75.0	**42.9**	**54.6**	51.6	61.5	56.1	64.9	49.9	54.6
BiLSTM-Attn+section title scaffold	77.2	85.9	81.3	53.8	56.0	54.9	**100.0**	40.0	57.1	33.3	40.0	36.4	50.0	28.6	36.4	**81.8**	**69.2**	**75.0**	66.0	53.3	56.9
BiLSTM-Attn+citation worthiness scaffold	77.1	90.1	83.1	59.1	52.0	55.3	**100.0**	40.0	57.1	28.6	40.0	33.3	50.0	28.6	36.4	81.0	65.4	72.3	66.0	52.7	56.3
BiLSTM-Attn+both scaffolds	**77.6**	**93.0**	**84.6**	65.0	52.0	57.8	**100.0**	**60.0**	**75.0**	40.0	40.0	40.0	75.0	**42.9**	54.5	72.7	61.5	66.7	71.7	58.2	63.1
BiLSTM-Attn+both scaffolds /w ELMo	75.9	**93.0**	83.5	**80.0**	**64.0**	**71.1**	75.0	**60.0**	66.7	**75.0**	**60.0**	**66.7**	**100.0**	28.6	44.4	**81.8**	**69.2**	**75.0**	**81.3**	**62.5**	**67.9**

Table 5: Detailed per category classification results on ACL-ARC dataset.

Category (# instances)	Background (1,014)			Method (613)			Result (260)			Average (Macro)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BiLSTM-Attn	82.2	83.2	82.7	80.7	74.4	77.4	67.1	76.2	71.4	76.7	77.9	77.2
BiLSTM-Attn w/ ELMo	**86.6**	87	86.8	87.2	79.1	83.0	71.5	**85.8**	78.0	81.8	**84.0**	82.6
Previous SOTA (Jurgens et al., 2018)	77.9	**92.9**	84.7	**91.5**	63.1	74.7	79.1	77.3	78.2	82.8	77.8	79.2
BiLSTM-Attn + section title scaffold	81.3	86.0	83.6	85.3	68.8	76.2	66.8	81.9	73.6	77.8	78.9	77.8
BiLSTM-Attn + citation worthiness scaffold	82.9	84.8	83.8	84.6	73.2	78.5	65.4	80.0	72.0	77.6	79.3	78.1
BiLSTM-Attn + both scaffolds	85.4	80.8	83.0	78.6	80.4	79.5	69.8	80.8	74.9	77.9	80.7	79.1
BiLSTM-Attn w/ ELMo + both scaffolds	85.4	90.3	**87.8**	89.5	**80.8**	**84.9**	**79.3**	79.6	**79.5**	**84.7**	83.6	**84.0**

Table 6: Detailed per category classification results on the SciCite dataset.

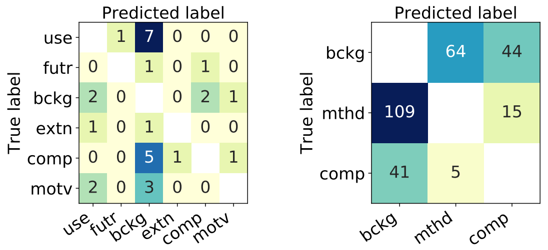
Example	True	Prediction
Our work is inspired by the latent left-linking model in (CITATION) and the ILP formulation from (CITATION).	Motivation	Use
ASARES is presented in detail in (CITATION) .	Use	Background
The advantage of tuning similarity to the application of interest has been shown previously by (CITATION).	Compare	Background
One possible direction is to consider linguistically motivated approaches , such as the extraction of syntactic phrase tables as proposed by (CITATION).	FutureWork	Background
After the extraction, pruning techniques (CITATION) can be applied to increase the precision of the extraction.	Background	Use

Table 7: A sample of model's classification errors on ACL-ARC dataset

errors are due to the model failing to properly differentiate the Use category with Background. We found out that some of these errors would have been possibly prevented by using additional context. Table 7 shows a sample of such classification errors. For the citation in the first row of the table, the model is likely distracted by "model in (citation)" and "ILP formulation from (citation)" deeming the sentence is referring to the use of another method from a cited paper and it misses the first part of the sentence describing the motivation. This is likely due to the small number of training instances in the Motivation category, preventing the model to learn such nuances. For the examples in the second and third row, it is not clear if it is possible to make the correct prediction without additional context. And similarly in the last row the instance seems ambiguous without accessing to additional context. Similarly as shown in Figure 4a two of FutureWork labels are wrongly classified. One of them is illustrated in the forth row of Table 7 where perhaps additional context could have helped the model in identifying the correct label. One possible way to prevent this type of errors, is to provide the model with an additional input, modeling the extended surrounding context. We experimented with encoding the extended surrounding context using a BiLSTM and concatenating it with the main citation context vector (z), but it resulted in a large decline in overall performance likely due to the overall noise introduced by the additional context. A possible future work is to investigate alternative effective approaches for incorporating the surrounding extended context.

5 Related Work

There is a large body of work studying the intent of citations and devising categorization systems (Stevens and Giuliano, 1965; Moravcsik and Murugesan, 1975; Garzone and Mercer, 2000; White, 2004; Ahmed et al., 2004; Teufel et al., 2006; Agarwal et al., 2010; Dong and Schäfer, 2011). Most of these efforts provide citation categories that are too fine-grained, some of which rarely occur in papers. Therefore, they are hardly useful for automated analysis of scientific publications. To address these problems and to unify previous



(a) ACL-ARC (test size: 139) (b) SciCite (test size: 1,861)

Figure 4: Confusion matrix showing classification errors of our best model on two datasets. The diagonal is masked to bring focus only on errors.

efforts, in a recent work, Jurgens et al. (2018) proposed a six category system for citation intents. In this work, we focus on two schemes: (1) the scheme proposed by Jurgens et al. (2018) and (2) an additional, more coarse-grained general-purpose category system that we propose (details in §3). Unlike other schemes that are domain-specific, our scheme is general and naturally fits in scientific discourse in multiple domains.

Early works in automated citation intent classification were based on rule-based systems (e.g., (Garzone and Mercer, 2000; Pham and Hoffmann, 2003)). Later, machine learning methods based on linguistic patterns and other hand-engineered features from citation context were found to be effective. For example, Teufel et al. (2006) proposed use of “cue phrases”, a set of expressions that talk about the act of presenting research in a paper. Abu-Jbara et al. (2013) relied on lexical, structural, and syntactic features and a linear SVM for classification. Researchers have also investigated methods of finding cited spans in the cited papers. Examples include feature-based methods (Cohan et al., 2015), domain-specific knowledge (Cohan and Goharian, 2017), and a recent CNN-based model for joint prediction of cited spans and citation function (Su et al., 2018). We also experimented with CNNs but found the attention BiLSTM model to work significantly better. Jurgens et al. (2018) expanded all pre-existing feature-based efforts on citation intent classification by proposing a comprehensive set of engineered features, including boostrapped patterns, topic modeling, dependency-based, and metadata features for the task. We argue that we can capture necessary information from the citation context using a data driven method, without the need for hand-engineered domain-dependent features or external resources. We propose a novel scaffold neural model for citation intent classification to incorporate structural information of scientific discourse into citations, borrowing the “scaffold” terminology from Swayamdipta et al. (2018) who use auxiliary syntactic tasks for semantic problems.

6 Conclusions and future work

In this work, we show that structural properties related to scientific discourse can be effectively used to inform citation intent classification. We propose a multitask learning framework with two auxiliary tasks (predicting section titles and citation worthiness) as two scaffolds related to the main task of citation intent prediction. Our model achieves state-of-the-art result (F1 score of 67.9%) on the ACL-ARC dataset with 13.3 absolute increase over the best previous results. We additionally introduce SciCite, a new large dataset of citation intents and also show the effectiveness of our model on this dataset. Our dataset, unlike existing datasets that are designed based on a specific domain, is more general and fits in scientific discourse from multiple scientific domains.

We demonstrate that carefully chosen auxiliary tasks that are inherently relevant to a main task can be leveraged to improve the performance on the main task. An interesting line of future work is to explore the design of such tasks or explore the properties or similarities between the auxiliary and the main tasks. Another relevant line of work is adapting our model to other domains containing documents with similar linked structured such as Wikipedia articles. Future work may benefit from replacing ELMo with other types of contextualized representations such as BERT in our scaffold model. For example, at the time of finalizing the camera ready version of this paper, Beltagy et al. (2019) showed that a BERT contextualized representation model (Devlin et al., 2018) trained on scientific text can achieve promising results on the SciCite dataset.

Acknowledgments

We thank Kyle Lo, Dan Weld, and Iz Beltagy for helpful discussions, Oren Etzioni for feedback on the paper, David Jurgens for helping us with their ACL-ARC dataset and reproducing their results, and the three anonymous reviewers for their comments and suggestions. Computations on `beaker.org` were supported in part by credits from Google Cloud.

References

Amjad Abu-Jbara, Jefferson Ezra, and Dragomir Radev. 2013. Purpose and polarity of citation: Towards nlp-based bibliometrics. In *NAACL-HLT*.

Shashank Agarwal, Lisha Choubey, and Hong Yu. 2010. Automatically classifying the role of citations in biomedical articles. In *AMIA Annual Symposium Proceedings*, volume 2010, page 11. American Medical Informatics Association.

Tanzila Ahmed, Ben Johnson, Charles Oppenheim, and Catherine Peck. 2004. Highly cited old papers and the reasons why they continue to be cited. part ii., the 1953 watson and crick article on the structure of dna. *Scientometrics*, 61(2):147–156.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan R. Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *LREC*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *EMNLP*.

Arman Cohan and Nazli Goharian. 2017. Contextualizing citations for scientific summarization using word embeddings and domain knowledge. In *SIGIR*.

Arman Cohan, Luca Soldaini, and Nazli Goharian. 2015. Matching citation text and cited spans in biomedical literature: a search-oriented approach. In *HLT-NAACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Cailing Dong and Ulrich Schäfer. 2011. Ensemble-style self-training on citation classification. In *IJCNLP*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Mark Garzone and Robert E Mercer. 2000. Towards an automated citation classifier. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 337–346. Springer.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*, 6:391–406.

Loet Leydesdorff. 1998. Theories of citation? *Scientometrics*.

Zhi Li and Yuh-Shan Ho. 2008. Use of citation per publication as an indicator to evaluate contingent valuation research. *Scientometrics*.

Terttu Luukkonen. 1992. Is scientists' publishing behaviour rewardseeking? *Scientometrics*.

Michael J Moravcsik and Poovanalingam Murugesan. 1975. Some results on the function and quality of citations. *Social studies of science*, 5(1):86–92.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Son Bao Pham and Achim Hoffmann. 2003. A new approach for scientific citation classification using cue phrases. In *Australasian Joint Conference on Artificial Intelligence*, pages 759–771. Springer.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *EMNLP*.

Anna Ritchie. 2009. Citation context analysis for information retrieval. Technical report, University of Cambridge, Computer Laboratory.

Henry Small. 2018. Characterizing highly cited method and non-method papers using citation contexts: The role of uncertainty. *Journal of Informetrics*, 12(2):461 – 480.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mary Elizabeth Stevens and Vincent Edward Giuliano. 1965. *Statistical Association Methods for Mechanized Documentation: Symposium Proceedings, Washington, 1964*, volume 269. US Government Printing Office.

Xuan Su, Animesh Prasad, Min-Yen Kan, and Kazunari Sugiyama. 2018. Neural multi-task learning for citation function and provenance. *CoRR*, abs/1811.07351.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke S. Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *EMNLP*.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *EMNLP*, EMNLP '06, pages 103–110, Stroudsburg, PA, USA. Association for Computational Linguistics.

Howard D White. 2004. Citation analysis and discourse analysis revisited. *Applied linguistics*, 25(1):89–116.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

`pair2vec`: Compositional Word-Pair Embeddings for Cross-Sentence Inference

Mandar Joshi[†] **Eunsol Choi**[†] **Omer Levy**[‡] **Daniel S. Weld**[†] **Luke Zettlemoyer**[†‡]

[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,eunsol,weld,lsz}@cs.washington.edu

[‡] Facebook AI Research, Seattle
{omerlevy,lsz}@fb.com

Abstract

Reasoning about implied relationships (e.g. paraphrastic, common sense, encyclopedic) between pairs of words is crucial for many cross-sentence inference problems. This paper proposes new methods for learning and using embeddings of word *pairs* that implicitly represent background knowledge about such relationships. Our pairwise embeddings are computed as a compositional function on word representations, which is learned by maximizing the pointwise mutual information (PMI) with the contexts in which the two words co-occur. We add these representations to the cross-sentence attention layer of existing inference models (e.g. BiDAF for QA, ESIM for NLI), instead of extending or replacing existing word embeddings. Experiments show a gain of 2.7% on the recently released SQuAD 2.0 and 1.3% on MultiNLI. Our representations also aid in better generalization with gains of around 6-7% on adversarial SQuAD datasets, and 8.8% on the adversarial entailment test set by Glockner et al. (2018).

1 Introduction

Reasoning about relationships between pairs of words is crucial for cross sentence inference problems such as question answering (QA) and natural language inference (NLI). In NLI, for example, given the premise "*golf is prohibitively expensive*", inferring that the hypothesis "*golf is a cheap pastime*" is a contradiction requires one to know that *expensive* and *cheap* are antonyms. Recent work (Glockner et al., 2018) has shown that current models, which rely heavily on unsupervised single-word embeddings, struggle to learn such relationships. In this paper, we show that they can be learned with word *pair* vectors (`pair2vec`[1]), which are trained unsupervised, and which significantly improve performance when added to existing cross-sentence attention mechanisms.

X	**Y**	**Contexts**
hot	*cold*	with **X** and **Y** baths too **X** or too **Y** neither **X** nor **Y**
Portland	*Oregon*	in **X**, **Y** the **X** metropolitan area in **Y** **X** International Airport in **Y**
crop	*wheat*	food **X** are maize, **Y**, etc dry **X**, such as **Y**, more **X** circles appeared in **Y** fields
Android	*Google*	**X** OS comes with **Y** play the **X** team at **Y** **X** is developed by **Y**

Table 1: Example word pairs and their contexts.

Unlike single-word representations, which typically model the co-occurrence of a target word x with its context c, our word-pair representations are learned by modeling the three-way co-occurrence between words (x, y) and the context c that ties them together, as seen in Table 1. While similar training signals have been used to learn models for ontology construction (Hearst, 1992; Snow et al., 2005; Turney, 2005; Shwartz et al., 2016) and knowledge base completion (Riedel et al., 2013), this paper shows, for the first time, that large scale learning of pairwise embeddings can be used to directly improve the performance of neural cross-sentence inference models.

More specifically, we train a feedforward network $R(x, y)$ that learns representations for the individual words x and y, as well as how to compose them into a single vector. Training is done by maximizing a generalized notion of the pointwise mutual information (PMI) among x, y, and their context c using a variant of negative sampling (Mikolov et al., 2013a). Making $R(x, y)$ a

---

[1] `https://github.com/mandarjoshi90/pair2vec`

Proceedings of NAACL-HLT 2019, pages 3597–3608
Minneapolis, Minnesota, June 2 - June 7, 2019.

compositional function on individual words alleviates the sparsity that necessarily comes with embedding pairs of words, even at a very large scale.

We show that our embeddings can be added to existing cross-sentence inference models, such as BiDAF++ (Seo et al., 2017; Clark and Gardner, 2018) for QA and ESIM (Chen et al., 2017) for NLI. Instead of changing the word embeddings that are fed into the encoder, we add the pretrained *pair* representations to *higher layers* in the network where cross sentence attention mechanisms are used. This allows the model to use the background knowledge that the pair embeddings implicitly encode to reason about the likely relationships between the pairs of words it aligns.

Experiments show that simply adding our word-pair embeddings to existing high-performing models, which already use ELMo (Peters et al., 2018), results in sizable gains. We show 2.72 F1 points over the BiDAF++ model (Clark and Gardner, 2018) on SQuAD 2.0 (Rajpurkar et al., 2018), as well as a 1.3 point gain over ESIM (Chen et al., 2017) on MultiNLI (Williams et al., 2018). Additionally, our approach generalizes well to adversarial examples, with a 6-7% F1 increase on adversarial SQuAD (Jia and Liang, 2017) and a 8.8% gain on the Glockner et al. (2018) NLI benchmark. An analysis of `pair2vec` on word analogies suggests that it complements the information in single-word representations, especially for encyclopedic and lexicographic relations.

2 Unsupervised Pretraining

Extending the distributional hypothesis to word pairs, we posit that similar word *pairs* tend to occur in similar contexts, and that the contexts provide strong clues about the likely relationships that hold between the words (see Table 1). We assume a dataset of (x, y, c) triplets, where each instance depicts a word pair (x, y) and the context c in which they appeared. We learn two compositional representation functions, $R(x, y)$ and $C(c)$, to encode the pair and the context, respectively, as d-dimensional vectors (Section 2.1). The functions are trained using a variant of negative sampling, which tries to embed word pairs (x, y) close to the contexts c with which they appeared (Section 2.2).

2.1 Representation

Our word-pair and context representations are both fixed-length vectors, composed from individual words. The word-pair representation function $R(x, y)$ first embeds and normalizes the individual words with a shared lookup matrix E_a:

$$\mathbf{x} = \frac{E_a(x)}{\|E_a(x)\|} \qquad \mathbf{y} = \frac{E_a(y)}{\|E_a(y)\|}$$

These vectors, along with their element-wise product, are fed into a four-layer perceptron:

$$R(x, y) = MLP^4(\mathbf{x}, \mathbf{y}, \mathbf{x} \circ \mathbf{y})$$

The context $c = c_1 ... c_n$ is encoded as a d-dimensional vector using the function $C(c)$. $C(c)$ embeds each token c_i with a lookup matrix E_c, contextualizes it with a single-layer Bi-LSTM, and then aggregates the entire context with attentive pooling:

$$\begin{aligned}
\mathbf{c}_i &= E_c(c_i) \\
\mathbf{h_1} ... \mathbf{h_n} &= \text{BiLSTM}(\mathbf{c}_1 ... \mathbf{c}_n) \\
w &= \text{softmax}_i(\mathbf{k}\mathbf{h_i}) \\
C(c) &= \sum_i w_i \mathbf{W}\mathbf{h}_i
\end{aligned}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{k} \in \mathbb{R}^d$. All parameters, including the lookup tables E_a and E_c, are trained.

Our representation is similar to two recently-proposed frameworks by Washio and Kato (2018a,b), but differs in that: (1) they use dependency paths as context, while we use surface form; (2) they encode the context as either a lookup table or the last state of a unidirectional LSTM. We also use a different objective, which we discuss next.

2.2 Objective

To optimize our representation functions, we consider two variants of negative sampling (Mikolov et al., 2013a): bivariate and multivariate. The original bivariate objective models the two-way distribution of context and (monolithic) word pair co-occurrences, while our multivariate extension models the three-way distribution of word-word-context co-occurrences. We further augment the multivariate objective with typed sampling to upsample harder negative examples. We discuss the impact of the bivariate and multivariate objectives (and other components) in Section 4.3.

Bivariate Negative Sampling Our objective aspires to make $R(x, y)$ and $C(c)$ similar (have high inner products) for (x, y, c) that were observed together in the data. At the same time, we wish

Bivariate	$J_{2NS}(x,y,c) = \log\sigma\left(R(x,y)\cdot C(c)\right) + \sum_{i=1}^{k_c}\log\sigma\left(-R(x,y)\cdot C(c_i^N)\right)$
Multivariate	$J_{3NS}(x,y,c) = J_{2NS}(x,y,c) + \sum_{i=1}^{k_x}\log\sigma\left(-R(x_i^N,y)\cdot C(c)\right) + \sum_{i=1}^{k_y}\log\sigma\left(-R(x,y_i^N)\cdot C(c)\right)$

Table 2: The bivariate and multivariate negative sampling objectives. The superscript ^{N} marks randomly sampled components, with k_* being the negative sample size per instance. The equations present per-instance objectives.

to keep our pair vectors *dis*-similar from random context vectors. In a straightforward application of the original (bivariate) negative sampling objective, we could generate a negative example from each observed (x, y, c) instance by replacing the original context c with a randomly-sampled context c^N (Table 2, J_{2NS}).

Assuming that the negative contexts are sampled from the empirical distribution $P(\cdot,\cdot,c)$ (with $P(x,y,c)$ being the portion of (x,y,c) instances in the dataset), we can follow Levy and Goldberg (2014) to show that this objective converges into the pointwise mutual information (PMI) between the word pair and the context.

$$R(x,y)\cdot C(c) = \log\frac{P(x,y,c)}{k_c P(x,y,\cdot)P(\cdot,\cdot,c)}$$

This objective mainly captures co-occurrences of monolithic pairs and contexts, and is limited by the fact that the training data, by construction, only contains pairs occurring within a sentence. For better generalization to cross-sentence tasks, where the pair distribution differs from that of the training data, we need a multivariate objective that captures the full three-way (x,y,c) interaction.

Multivariate Negative Sampling We introduce negative sampling of target words, x and y, in addition to negative sampling of contexts c (Table 2, J_{3NS}). Our new objective also converges to a novel multivariate generalization of PMI, different from previous PMI extensions that were inspired by information theory (Van de Cruys, 2011) and heuristics (Jameel et al., 2018).[2] Following Levy and Goldberg (2014), we can show that when replacing target words in addition to contexts, our objective will converge[3] to the optimal value in Equation 1:

$$R(x,y)\cdot C(c) = \log\frac{P(x,y,c)}{Z_{x,y,c}} \quad (1)$$

where $Z_{x,y,c}$, the denominator, is:

$$\begin{aligned} Z_{x,y,c} = {} & k_c P(\cdot,\cdot,c)P(x,y,\cdot) \\ & + k_x P(x,\cdot,\cdot)P(\cdot,y,c) \\ & + k_y P(\cdot,y,\cdot)P(x,\cdot,c) \end{aligned} \quad (2)$$

This optimal value deviates from previous generalizations of PMI by having a linear mixture of marginal probability products in its denominator. By introducing terms such as $P(x,\cdot,c)$ and $P(\cdot,y,c)$, the objective penalizes spurious correlations between words and contexts that disregard the other target word. For example, it would assign the pattern "*X is a Y*" a high score with (*banana*, *fruit*), but a lower score with (*cat*, *fruit*).

Typed Sampling In multivariate negative sampling, we typically replace x and y by sampling from their unigram distributions. In addition to this, we also sample uniformly from the top 100 words according to cosine similarity using distributional word vectors. This is done to encourage the model to learn relations between specific instances as opposed to more general types. For example, using *California* as a negative sample for *Oregon* helps the model to learn that the pattern "*X is located in Y*" fits the pair (*Portland*, *Oregon*), but not the pair (*Portland*, *California*). Similar adversarial constraints were used in knowledge base completion (Toutanova et al., 2015) and word embeddings (Li et al., 2017).[4]

3 Integrating `pair2vec` into Models

We first present a general outline for incorporating `pair2vec` into attention-based architectures, and then discuss changes made to BiDAF++ and ESIM. The key idea is to inject our pairwise representations into the attention layer by reusing the cross-sentence attention weights. In addition to attentive pooling over single word representations, we also pool over cross-sentence word pair embeddings (Figure 1).

---

[2] See supplementary material for their exact formulations.

[3] A full proof is provided in the supplementary material.

[4] Applying typed sampling also changes the value to which our objective will converge, and will replace the unigram probabilities in Equation (2) to reflect the type-based distribution.

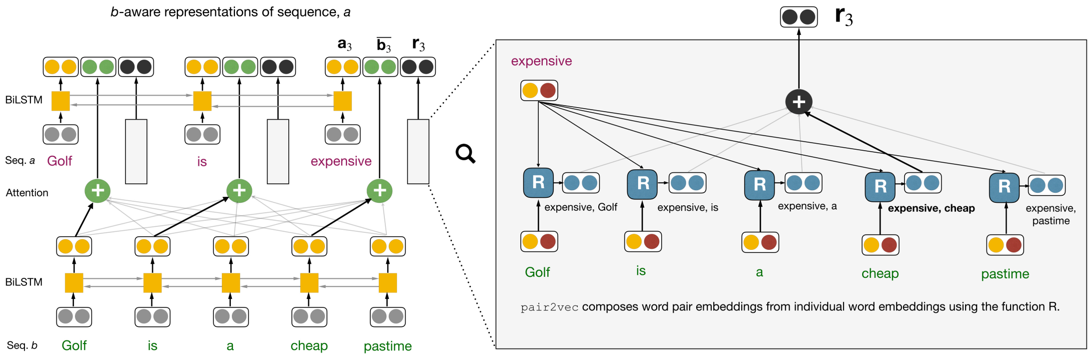


Figure 1: A typical architecture of a cross-sentence inference model (left), and how `pair2vec` is added to it (right). Given two sequences, a and b, existing models create b-aware representations of words in a. For any word a_i, this typically involves the BiLSTM representation of word a_i ($\mathbf{a}_i$), and an attention-weighted sum over b's BiLSTM states with a_i as the query ($\overline{\mathbf{b}}_i$). To these, we add the word-pair representation of a_i and each word in b, weighted by attention ($\mathbf{r}_i$). Thicker attention arrows indicate stronger word pair alignments (e.g. *cheap*, *expensive*).

3.1 General Approach

Pair Representation We assume that we are given two sequences $a = a_1 ... a_n$ and $b = b_1 ... b_m$. We represent the word-pair embeddings between a and b using the pretrained `pair2vec` model as:

$$\mathbf{r}_{i,j} = \left[\frac{R(a_i, b_j)}{\|R(a_i, b_j)\|}; \frac{R(b_j, a_i)}{\|R(b_j, a_i)\|} \right] \tag{3}$$

We include embeddings in both directions, $R(a_i, b_j)$ and $R(b_j, a_i)$, because the many relations can be expressed in both directions; e.g., hypernymy can be expressed via "*X is a type of Y*" as well as "*Y such as X*". We take the L_2 normalization of each direction's pair embedding because the heavy-tailed distribution of word pairs results in significant variance of their norms.

Base Model Let $\mathbf{a}_1 ... \mathbf{a}_n$ and $\mathbf{b}_1 ... \mathbf{b}_m$ be the vector representations of sequences a and b, as produced by the input encoder (e.g. ELMo embeddings contextualized with model-specific BiLSTMs). Furthermore, we assume that the base model computes soft word alignments between a and b via co-attention (4, 5), which are then used to compute b-aware representations of a:

$$s_{i,j} = f_{att}(\mathbf{a}_i, \mathbf{b}_j) \tag{4}$$

$$\alpha = \text{softmax}_j(s_{i,j}) \tag{5}$$

$$\overline{\mathbf{b}}_i = \sum_{j=0}^{m} \alpha_{i,j} \mathbf{b}_j \tag{6}$$

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i\right] \tag{7}$$

The symmetric term $\mathbf{b}_j^{inf}$ is defined analogously. We refer to $\mathbf{a}^{inf}$ and $\mathbf{b}^{inf}$ as the inputs to the *inference* layer, since this layer computes some function over aligned word pairs, typically via a feed-forward network and LSTMs. The inference layer is followed by aggregation and output layers.

Injecting `pair2vec` We conjecture that the inference layer effectively learns word-pair relationships from training data, and it should, therefore, help to augment its input with `pair2vec`. We augment $\mathbf{a}_i^{inf}$ (7) with the pair vectors $\mathbf{r}_{i,j}$ (3) by concatenating a weighted average of the pair vectors $r_{i,j}$ involving a_i, where the weights are the same $\alpha_{i,j}$ computed via attention in (5):

$$\mathbf{r}_i = \sum_j \alpha_{i,j} \mathbf{r}_{i,j} \tag{8}$$

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i; \mathbf{r}_i\right] \tag{9}$$

The symmetric term $\mathbf{b}_j^{inf}$ is defined analogously.

3.2 Question Answering

We augment the inference layer in the BiDAF++ model with `pair2vec`. BiDAF++ is an improved version of the BiDAFNoAnswer (Seo et al., 2017; Levy et al., 2017) which includes self-attention and ELMo embeddings from Peters et al. (2018). We found this variant to be stronger than the baselines presented in Rajpurkar et al. (2018) by over 2.5 F1. We use BiDAF++ as a baseline since its architecture is typical for QA systems, and, until recently, was state-of-the-art on SQuAD 2.0 and other benchmarks.

BiDAF++ Let $\mathbf{a}$ and $\mathbf{b}$ be the outputs of the passage and question encoders respectively (in place of the standard $\mathbf{p}$ and $\mathbf{q}$ notations). The inference layer's inputs $\mathbf{a}_i^{inf}$ are defined similarly to the generic model's in (7), but also contain an aggregation of the elements in $\mathbf{a}$, with better-aligned elements receiving larger weights:

$$\mu = \text{softmax}_i(\max_j s_{i,j}) \tag{10}$$

$$\hat{\mathbf{a}}_i = \sum_i \mu_i \mathbf{a}_i \tag{11}$$

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i; \mathbf{a}_i \circ \overline{\mathbf{b}}_i; \hat{\mathbf{a}}\right] \tag{12}$$

In the later layers, $\mathbf{a}^{inf}$ is recontextualized using a BiGRU and self attention. Finally a prediction layer predicts the start and end tokens.

BiDAF++ with `pair2vec` To add our pair vectors, we simply concatenate $\mathbf{r}_i$ (3) to $\mathbf{a}_i^{inf}$ (12):

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i; \mathbf{a}_i \circ \overline{\mathbf{b}}_i; \hat{\mathbf{a}}; \mathbf{r}_i\right] \tag{13}$$

3.3 Natural Language Inference

For NLI, we augment the ESIM model (Chen et al., 2017), which was previously state-of-the-art on both SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) benchmarks.

ESIM Let $\mathbf{a}$ and $\mathbf{b}$ be the outputs of the premise and hypothesis encoders respectively (in place of the standard $\mathbf{p}$ and $\mathbf{h}$ notations). The inference layer's inputs $\mathbf{a}_i^{inf}$ (and $\mathbf{b}_j^{inf}$) are defined similarly to the generic model's in (7):

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i; \mathbf{a}_i \circ \overline{\mathbf{b}}_i; \mathbf{a}_i - \overline{\mathbf{b}}_i\right] \tag{14}$$

In the later layers, $\mathbf{a}^{inf}$ and $\mathbf{b}^{inf}$ are projected, recontextualized, and converted to a fixed-length vector for each sentence using multiple pooling schemes. These vectors are then passed on to an output layer, which predicts the class.

ESIM with `pair2vec` To add our pair vectors, we simply concatenate $\mathbf{r}_i$ (3) to $\mathbf{a}_i^{inf}$ (14):

$$\mathbf{a}_i^{inf} = \left[\mathbf{a}_i; \overline{\mathbf{b}}_i; \mathbf{a}_i \circ \overline{\mathbf{b}}_i; \mathbf{a}_i - \overline{\mathbf{b}}_i; \mathbf{r}_i\right] \tag{15}$$

A similar augmentation of ESIM was recently proposed in KIM (Chen et al., 2018). However, their pair vectors are composed of WordNet features, while our pair embeddings are learned directly from text (see further discussion in Section 6).

4 Experiments

For experiments on QA (Section 4.1) and NLI (Section 4.2), we use our full model which includes multivariate and typed negative sampling. We discuss ablations in Section 4.3

Benchmark		BiDAF	+ `pair2vec`	Δ
SQuAD 2.0	EM	65.66	68.02	+2.36
	F1	68.86	71.58	+2.72
AddSent	EM	37.50	44.20	+6.70
	F1	42.55	49.69	+7.14
AddOneSent	EM	48.20	53.30	+5.10
	F1	54.02	60.13	+6.11

Table 3: Performance on SQuAD 2.0 and adversarial SQuAD (AddSent and AddOneSent) benchmarks, with and without `pair2vec`. All models have ELMo.

Benchmark	ESIM	+ `pair2vec`	Δ
Matched	79.68	81.03	+1.35
Mismatched	78.80	80.12	+1.32

Table 4: Performance on MultiNLI, with and without `pair2vec`. All models have ELMo.

Data We use the January 2018 dump of English Wikipedia, containing 96M sentences to train `pair2vec`. We restrict the vocabulary to the 100K most frequent words. Preprocessing removes all out-of-vocabulary words in the corpus. We consider each word pair within a window of 5 in the preprocessed corpus, and subsample[5] instances based on pair probability with a threshold of $5 \cdot 10^{-7}$. We define the context as one word each to the left and right, and all the words in between each pair, replacing both target words with placeholders X and Y (see Table 1). More details can be found in the supplementary material.

4.1 Question Answering

We experiment on the SQuAD 2.0 QA benchmark (Rajpurkar et al., 2018), as well as the adversarial datasets of SQuAD 1.1 (Rajpurkar et al., 2016; Jia and Liang, 2017). Table 3 shows the performance of BiDAF++, with ELMo , before and after adding `pair2vec`. Experiments on SQuAD 2.0 show that our pair representations improve performance by 2.72 F1. Moreover, adding `pair2vec` also results in better generalization on the adversarial SQuAD datasets with gains of 7.14 and 6.11 F1.

4.2 Natural Language Inference

We report the performance of our model on MultiNLI and the adversarial test set from Glockner et al. (2018) in Table 5. We outperform the

---

[5] Like in `word2vec`, subsampling reduces the size of the dataset and speeds up training. For this, we define the word pair probability as the product of unigram probabilities.

Model	Accuracy
Rule-based Models	
WordNet Baseline	85.8
Models with GloVe	
ESIM (Chen et al., 2017)	77.0
KIM (Chen et al., 2018)	87.7
ESIM + `pair2vec`	**92.9**
Models with ELMo	
ESIM (Peters et al., 2018)	84.6
ESIM + `pair2vec`	**93.4**

Table 5: Performance on the adversarial NLI test set of Glockner et al. (2018).

Model	EM (Δ)	F1 (Δ)
`pair2vec` (Full Model)	69.20	72.68
Composition: 2 Layers	68.35 (-0.85)	71.65 (-1.03)
Composition: Multiply	67.10 (-2.20)	70.20 (-2.48)
Objective: Bivariate NS	68.63 (-0.57)	71.98 (-0.70)
Unsupervised: Pair Dist	67.07 (-2.13)	70.24 (-2.44)
No `pair2vec` (BiDAF)	66.66 (-2.54)	69.90 (-2.78)

Table 6: Ablations on the Squad 2.0 development set show that argument sampling as well as using a deeper composition function are useful.

ESIM + ELMo baseline by 1.3% on the matched and mismatched portions of the dataset.

We also record a gain of 8.8% absolute over ESIM on the Glockner et al. (2018) dataset, setting a new state of the art. Following standard practice (Glockner et al., 2018), we train all models on a combination of SNLI (Bowman et al., 2015) and MultiNLI. Glockner et al. (2018) show that with the exception of KIM (Chen et al., 2018), which uses WordNet features, several NLI models fail to generalize to this setting which involves lexical inference. For a fair comparison with KIM on the Glockner test set, we replace ELMo with GLoVE embeddings, and still outperform KIM by almost halving the error rate.

4.3 Ablations

Ablating parts of `pair2vec` shows that all components of the model (Section 2) are useful. We ablate each component and report the EM and F1 on the development set of SQuAD 2.0 (Table 6). The full model, which uses a 4-layer MLP for $R(x, y)$ and trains with multivariate negative sampling, achieves the highest F1 of 72.68.

We experiment with two alternative composition functions, a 2-layer MLP (*Composition: 2 Layers*) and element-wise multiplication (*Composition: Multiply*), which yield significantly smaller gains over the baseline BiDAF++ model. This demonstrates the need for a deep composition function. Eliminating sampling of target words (x, y) from the objective (*Objective: Bivariate NS*) results in a drop of 0.7 F1, accounting for about a quarter of the overall gain. This suggests that while the bulk of the signal is mined from the pair-context interactions, there is also valuable information in other interactions as well.

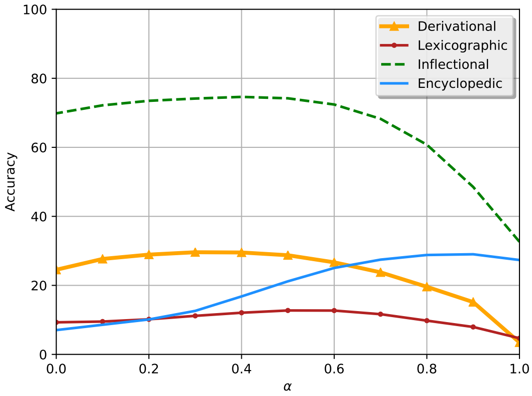


Figure 2: Accuracy as a function of the interpolation parameter α (see Eq. (16)). The α=0 configuration relies only on fastText (Bojanowski et al., 2017), while α=1 reflects `pair2vec`.

We also test whether specific pre-training of word *pair* representations is useful by replacing `pair2vec` embeddings with the vector offsets of pre-trained word embeddings (*Unsupervised: Pair Dist*). We follow the PairDistance method for word analogies (Mikolov et al., 2013b), and represent the pair (x, y) as the L2 normalized difference of single-word vectors: $(\mathbf{x} - \mathbf{y})/\|\mathbf{x} - \mathbf{y}\|$. We use the same fastText (Bojanowski et al., 2017) word vectors with which we initialized `pair2vec` before training. We observe a gain of only 0.34 F1 over the baseline.

5 Analysis

In Section 4, we showed that `pair2vec` adds information complementary to single-word representations like ELMo. Here, we ask what this extra information is, and try to characterize which word relations are better captured by `pair2vec`. To that end, we evaluate performance on a word analogy dataset with over 40 different relation types (Section 5.1), and observe how `pair2vec` fills hand-crafted relation patterns (Section 5.2).

Relation	`3CosAdd`	`+pair2vec`	α^*
Country:Capital	1.2	86.1	0.9
Name:Occupation	1.8	44.6	0.8
Name:Nationality	0.1	42.0	0.9
UK City:County	0.7	31.7	1.0
Country:Language	4.0	28.4	0.8
Verb 3pSg:Ved	49.1	61.7	0.6
Verb Ving:Ved	61.1	73.3	0.5
Verb Inf:Ved	58.5	70.1	0.5
Noun+less	4.8	16.0	0.2
Substance Meronym	3.8	14.5	0.6

Table 7: The top 10 analogy relations for which interpolating with `pair2vec` improves performance. α^* is the optimal interpolation parameter for each relation.

5.1 Quantitative Analysis: Word Analogies

Word Analogy Dataset Given a word pair (a, b) and word x, the word analogy task involves predicting a word y such that $a : b :: x : y$. We use the Bigger Analogy Test Set (BATS, Gladkova et al., 2016) which contains four groups of relations: encyclopedic semantics (e.g., person-profession as in *Einstein-physicist*), lexicographic semantics (e.g., antonymy as in *cheap-expensive*), derivational morphology (e.g., noun forms as in *oblige-obligation*), and inflectional morphology (e.g., noun-plural as in *bird-birds*). Each group contains 10 sub-relations.

Method We interpolate `pair2vec` and `3CosAdd` (Mikolov et al., 2013b; Levy et al., 2014) scores on fastText embeddings, as follows:

$$\begin{aligned}\text{score}(y) = {} & \alpha \cdot \cos(\mathbf{r}_{a,b}, \mathbf{r}_{x,y}) \\ & + (1-\alpha) \cdot \cos(\mathbf{b} - \mathbf{a} + \mathbf{x}, \mathbf{y}) \end{aligned} \quad (16)$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{x}$, and $\mathbf{y}$ represent fastText embeddings[6] and $\mathbf{r}_{a,b}$, $\mathbf{r}_{x,y}$ represent the `pair2vec` embedding for the word pairs (a, b) and (x, y), respectively; α is the linear interpolation parameter. Following prior work (Mikolov et al., 2013b), we return the highest-scoring y in the entire vocabulary, excluding the given words a, b, and x.

Results Figure 2 shows how the accuracy on each category of relations varies with α. For all four groups, adding `pair2vec` to `3CosAdd` results in significant gains. In particular, the biggest relative improvements are observed for encyclopedic (356%) and lexicographic (51%) relations.

Table 7 shows the specific relations in which `pair2vec` made the largest absolute impact. The gains are particularly significant for relations where fastText embeddings provide limited signal. For example, the accuracy for *substance meronyms* goes from 3.8% to 14.5%. In some cases, there is also a synergistic effect; for instance, in *noun+less*, `pair2vec` alone scored 0% accuracy, but mixing it with `3CosAdd`, which got 4.8% on its own, yielded 16% accuracy.

These results, alongside our experiments in Section 4, strongly suggest that `pair2vec` encodes information complementary to that in single-word embedding methods such as fastText and ELMo.

5.2 Qualitative Analysis: Slot Filling

To further explore how `pair2vec` encodes such complementary information, we consider a setting similar to that of knowledge base completion: given a Hearst-like context pattern c and a single word x, predict the other word y from the entire vocabulary. We rank candidate words y based on the scoring function in our training objective: $R(x, y) \cdot C(c)$. We use a fixed set of example relations and manually define their predictive context patterns and a small set of candidate words x.

Table 8 shows the top three y words. The model embeds (x, y) pairs close to contexts that reflect their relationship. For example, substituting *Portland* in the city-state pattern ("*in X, Y.*"), the top two words are *Oregon* and *Maine*, both US states with cities named Portland. When used with the city-city pattern ("*from X to Y.*"), the top two words are *Salem* and *Astoria*, both cities in Oregon. The word-context interaction often captures multiple relations; for example, *Monet* is used to refer to the painter (*profession*) as well as his paintings.

As intended, `pair2vec` captures the three-way word-word-context interaction, and not just the two-way word-context interaction (as in single-word embeddings). This profound difference allows `pair2vec` to complement single-word embeddings with additional information.

6 Related Work

Pretrained Word Embeddings Many state-of-the-art models initialize their word representations using pretrained embeddings such as `word2vec` (Mikolov et al., 2013a) or ELMo (Peters et al., 2018). These representations are typically trained using an interpretation of the Distributional Hy-

---

[6] The fastText embeddings in the analysis were retrained using the same Wikipedia corpus used to train `pair2vec` to control for the corpus when comparing the two methods.

Relation	**Context**	**X**	**Y (Top 3)**
Antonymy/Exclusion	either X or Y	accept hard	*reject*, *refuse*, recognise *soft*, *brittle*, *polished*
Hypernymy	including X and other Y	copper google	ones, *metals*, *mines* apps, *browsers*, *searches*
Hyponymy	X like Y	cities browsers	solaris, *speyer*, *medina* *chrome*, *firefox*, *netscape*
Co-hyponymy	, X , Y ,	copper google	*malachite*, *flint*, ivory *microsoft*, *bing*, *yahoo*
City-State	in X , Y .	portland dallas	*oregon*, *maine*, *dorset* *tx*, *texas*, va
City-City	from X to Y .	portland dallas	*salem*, *astoria*, ogdensburg *denton*, *allatoona*, *addison*
Profession	X , a famous Y ,	ronaldo monet	*footballer*, portuguese, *player* *painter*, painting, butterfly

Table 8: Given a context c and a word x, we select the top 3 words y from the entire vocabulary using our scoring function $R(x, y) \cdot C(c)$. The analysis suggests that the model tends to rank correct matches (italics) over others.

pothesis (Harris, 1954) in which the bivariate distribution of target words and contexts is modeled. Our work deviates from the word embedding literature in two major aspects. First, our goal is to represent word *pairs*, not individual words. Second, our new PMI formulation models the *trivariate* word-word-context distribution. Experiments show that our pair embeddings can complement single-word embeddings.

Mining Textual Patterns There is extensive literature on mining textual patterns to predict relations between words (Hearst, 1992; Snow et al., 2005; Turney, 2005; Riedel et al., 2013; Van de Cruys, 2014; Toutanova et al., 2015; Shwartz and Dagan, 2016). These approaches focus mostly on relations between pairs of nouns (perhaps with the exception of VerbOcean (Chklovski and Pantel, 2004)). More recently, they have been expanded to predict relations between unrestricted pairs of words (Jameel et al., 2018; Espinosa Anke and Schockaert, 2018), assuming that each word-pair was observed together during pretraining. Washio and Kato (2018a,b) relax this assumption with a compositional model that can represent any pair, as long as each word appeared (individually) in the corpus.

These methods are evaluated on either intrinsic relation prediction tasks, such as BLESS (Baroni and Lenci, 2011) and CogALex (Santus et al., 2016), or knowledge-base population benchmarks, e.g. FB15 (Bordes et al., 2013). To the best of our knowledge, our work is the first to integrate pattern-based methods into modern high-performing semantic models and evaluate their impact on complex end-tasks like QA and NLI.

Integrating Knowledge in Complex Models Ahn et al. (2016) integrate Freebase facts into a language model using a copying mechanism over fact attributes. Yang and Mitchell (2017) modify the LSTM cell to incorporate WordNet and NELL knowledge for event and entity extraction. For cross-sentence inference tasks, Weissenborn et al. (2017), Bauer et al. (2018), and Mihaylov and Frank (2018) dynamically refine word representations by reading assertions from ConceptNet and Wikipedia abstracts. Our approach, on the other hand, relies on a relatively simple extension of existing cross-sentence inference models. Furthermore, we do not need to dynamically retrieve and process knowledge base facts or Wikipedia texts, and just pretrain our pair vectors in advance.

KIM (Chen et al., 2017) integrates word-pair vectors into the ESIM model for NLI in a very similar way to ours. However, KIM's word-pair vectors contain only hand-engineered word-relation indicators from WordNet, whereas our word-pair vectors are automatically learned from unlabeled text. Our vectors can therefore reflect relation types that do not exist in WordNet (such as *profession*) as well as word pairs that do not have a direct link in WordNet (e.g. *bronze* and *statue*); see Table 8 for additional examples.

7 Conclusion and Future Work

We presented new methods for training and using word *pair* embeddings that implicitly represent background knowledge. Our pair embeddings are computed as a compositional function of the individual word representations, which is learned by maximizing a variant of the PMI with the contexts in which the the two words co-occur. Experiments on cross-sentence inference benchmarks demonstrated that adding these representations to existing models results in sizable improvements for both in-domain and adversarial settings.

Published concurrently with this paper, BERT (Devlin et al., 2018), which uses a masked language model objective, has reported dramatic gains on multiple semantic benchmarks including question-answering, natural language inference, and named entity recognition. Potential avenues for future work include multitasking BERT with `pair2vec` in order to more directly incorporate reasoning about word pair relations into the BERT objective.

Acknowledgments

We would like to thank Anna Rogers (Gladkova), Qian Chen, Koki Washio, Pranav Rajpurkar, and Robin Jia for their help with the evaluation. We are also grateful to members of the UW and FAIR NLP groups, and anonymous reviewers for their thoughtful comments and suggestions.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '11, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668. Association for Computational Linguistics.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855. Association for Computational Linguistics.

Tim Van de Cruys. 2011. Two multivariate generalizations of pointwise mutual information. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 16–20. Association for Computational Linguistics.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Luis Espinosa Anke and Steven Schockaert. 2018. Seven: Augmenting word embeddings with unsupervised relation vectors. In *Proceedings of the*

27th International Conference on Computational Linguistics, pages 2653–2665. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't. In *Proceedings of the NAACL-HLT SRW*, pages 47–54, San Diego, California, June 12-17, 2016. ACL.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655. Association for Computational Linguistics.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, Nantes, France.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018. Unsupervised learning of distributional relation vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open ie propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2177–2185, Cambridge, MA, USA. MIT Press.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*, pages 333–342.

Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers, and Xiaoyong Du. 2017. Investigating different syntactic context types and context representations for learning word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2421–2431. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016. The CogALex-V shared task on the corpus-based identification of semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 69–79. The COLING 2016 Organizing Committee.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon, CogALex@COLING 2016, Osaka, Japan, December 12, 2016*, pages 24–29.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1499–1509.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1136–1141, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koki Washio and Tsuneaki Kato. 2018a. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1123–1133. Association for Computational Linguistics.

Koki Washio and Tsuneaki Kato. 2018b. Neural latent relational analysis to capture lexical semantic relations in a vector space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*. Association for Computational Linguistics.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural NLU systems. *arXiv preprint arXiv:1706.02596*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446. Association for Computational Linguistics.

A Implementation Details

Hyperparameters For both word pairs and contexts, we use 300-dimensional word embeddings initialized with FastText (Bojanowski et al., 2017). The context representation uses a single-layer BiLSTM with a hidden layer size of 100. We use 2 negative context samples and 3 negative argument samples for each pair-context tuple.

For pre-training, we used stochastic gradient descent with an initial learning rate of 0.01. We reduce the learning rate by a factor of 0.9 if the loss does not decrease for 300K steps. We use a batch size of 600, and train for 12 epochs.[7]

For both end-task models, we use AllenNLP's implementations (Gardner et al., 2018) with default hyperparameters; we did not change any setting before or after injecting `pair2vec`. We use 0.15 dropout on our pretrained pair embeddings.

B Multivariate Negative Sampling

B.1 Relation to Multivariate PMI

In this appendix, we elaborate on mathematical details of multivariate negative sampling to support our claims in Section 2.2, and also discuss its relation to other PMI multivariate formulations (Table 9).

B.2 Global Objective

Equation (Table 2, J_{3NS}) in Section 2.2 characterizes the local objective for each data instance. To understand the mathematical properties of this objective, we must first describe the global objective in terms of the entire dataset. However, this

---

[7] On Titan X GPUs, the training takes about a week.

(Van de Cruys, 2011)	$SI_1(x,y,c)$	$\log \frac{P(x,y,\cdot)P(x,\cdot,c)P(\cdot,y,c)}{P(x,\cdot,\cdot)P(\cdot,y,\cdot)P(\cdot,\cdot,c)P(x,y,c)}$
	$SI_2(x,y,c)$	$\log \frac{P(x,y,c)}{P(x,\cdot,\cdot)P(\cdot,y,\cdot)P(\cdot,\cdot,c)}$
(Jameel et al., 2018)	$SI_3(x,y,c)$	$\log \frac{P(x,y,c)}{P(x,y,\cdot)P(\cdot,\cdot,c)}$
	$SI_4(x,y,c)$	$\log \frac{P(x,y,c)P(\cdot,\cdot,c)}{P(x,\cdot,c)P(\cdot,y,c)}$
(This Work)	$MVPMI(x,y,c)$	$\log \frac{P(x,y,c)}{k_c P(\cdot,\cdot,c)P(x,y,\cdot)+k_x P(x,\cdot,\cdot)P(\cdot,y,c)+k_y P(\cdot,y,\cdot)P(x,\cdot,c)}$

Table 9: Multivariate generalizations of PMI.

cannot be done by simply summing the local objective for each (x, y, c), since each such example may appear multiple times in our dataset. Moreover, due to the nature of negative sampling, the number of times an (x, y, c) triplet appears as a positive example will almost always be different from the number of times it appears as a negative one. Therefore, we must determine the frequency in which each triplet appears in each role.

We first denote the number of times the example (x, y, c) appears in the dataset as $\#(x, y, c)$; this is also the number of times (x, y, c) is used as a positive example. We observe that the expected number of times (x, y, c) is used as a corrupt x example is $k_x P(x, \cdot, \cdot)\#(\cdot, y, c)$, since (x, y, c) can only be created as a corrupt x example by randomly sampling x from an example that already contained y and c. The number of times (x, y, c) is used as a corrupt y or c example can be derived analogously. Therefore, the global objective of our trenary negative sampling approach is:

$$J_{3NS}^{\text{Global}} = \sum_{(x,y,c)} J_{3NS}^{x,y,c} \tag{17}$$

$$\begin{aligned} J_{3NS}^{x,y,c} &= \#(x,y,c) \cdot \log \sigma\left(S_{x,y,c}\right) \\ &+ Z'_{x,y,c} \cdot \log \sigma\left(-S_{x,y,c}\right) \end{aligned} \tag{18}$$

$$\begin{aligned} Z'_{x,y,c} &= k_x P(x,\cdot,\cdot)\#(\cdot,y,c) J_{-}^{x,y,c} \\ &+ k_y P(\cdot,y,\cdot)\#(x,\cdot,c) J_{-}^{x,y,c} \\ &+ k_c P(\cdot,\cdot,c)\#(x,y,\cdot) J_{-}^{x,y,c} \end{aligned} \tag{19}$$

$$S_{x,y,c} = R(x,y) \cdot C(c) \tag{20}$$

With the global objective, we can now ask what is the optimal value of $S_{x,y,c}$ (20) by comparing the partial derivative of (17) to zero. This derivative is in fact equal to the partial derivative of (18), since it is the only component of the global objective in which $R(x, y) \cdot C(c)$ appears:

$$0 = \frac{\partial J_{3NS}^{\text{Global}}}{\partial S_{x,y,c}} = \frac{\partial J_{3NS}^{x,y,c}}{\partial S_{x,y,c}}$$

The partial derivative of (18) can be expressed as:

$$0 = \#(x,y,c) \cdot \sigma\left(-S_{x,y,c}\right) - Z'_{x,y,c} \cdot \sigma\left(S_{x,y,c}\right)$$

which can be reformulated as:

$$S_{x,y,c} = \log \frac{\#(x,y,c)}{Z'_{x,y,c}}$$

By expanding the fraction by $1/\#(\cdot,\cdot,\cdot)$ (i.e. dividing by the size of the dataset), we essentially convert all the frequency counts (e.g. $\#(x, y, z)$) to empirical probabilities (e.g. $P(x, y, z)$), and arrive at Equation (1) in Section 2.2.

B.3 Other Multivariate PMI Formulations

Previous work has proposed different multivariate formulations of PMI, shown in Table 9. Van de Cruys (2011) presented specific interaction information (SI_1) and specific correlation (SI_2). In addition to those metrics, Jameel et al. (2018) experimented with SI_3, which is the bivariate PMI between (x, y) and c, and with SI_4. Our formulation deviates from previous work, and, to the best of our knowledge, cannot be trivially expressed by one of the existing metrics.

Submodular Optimization-based Diverse Paraphrasing and its Effectiveness in Data Augmentation

Ashutosh Kumar[*1] **Satwik Bhattamishra**[*2 †] **Manik Bhandari**[1] **Partha Talukdar**[1]
[1] Indian Institute of Science, Bangalore, India
[2] Birla Institute of Technology and Science, Pilani, India
{ashutosh, ppt}@iisc.ac.in, {satwik55, mbbhandarimanik}@gmail.com

Abstract

Inducing diversity in the task of paraphrasing is an important problem in NLP with applications in data augmentation and conversational agents. Previous paraphrasing approaches have mainly focused on the issue of generating semantically similar paraphrases, while paying little attention towards diversity. In fact, most of the methods rely solely on top-k beam search sequences to obtain a set of paraphrases. The resulting set, however, contains many structurally similar sentences. In this work, we focus on the task of obtaining highly diverse paraphrases while not compromising on paraphrasing quality. We provide a novel formulation of the problem in terms of monotone submodular function maximization, specifically targeted towards the task of paraphrasing. Additionally, we demonstrate the effectiveness of our method for data augmentation on multiple tasks such as intent classification and paraphrase recognition. In order to drive further research, we have made the source code available.

1 Introduction

Paraphrasing is the task of rephrasing a given text in multiple ways such that the semantics of the generated sentences remain unaltered. Paraphrasing *Quality* can be attributed to two key characteristics - *fidelity* which measures the semantic similarity between the input text and generated text, and *diversity*, which measures the lexical dissimilarity between generated sentences.

Many previous works (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2018) address the task of obtaining semantically similar paraphrases. While it is essential to produce paraphrases with high fidelity, it is equally important, and in many cases desirable, to produce lexically diverse ones. Diversity in paraphrase generation finds applications in text simplification (Nisioi et al., 2017; Xu et al., 2015), document summarization (Li et al., 2009; Nema et al., 2017), QA systems (Fader et al., 2013; Bernhard and Gurevych, 2008), data augmentation (Zhang et al., 2015; Wang and Yang, 2015), conversational agents (Li et al., 2016) and information retrieval (Anick and Tipirneni, 1999).

SOURCE	– how do i increase body height ?
REFERENCE	– what do i do to increase my height ?
BEAM SEARCH	– how do i increase my height ? – how do i increase my body height ? – how do i increase the height ? – how would i increase my body height ?
DIPS (OURS)	– how could i increase my height ? – what should i do to increase my height ? – what are the fastest ways to increase my height ? – is there any proven method to increase height ?

Table 1: Sample paraphrases generated by beam search and DiPS (our method). DiPS offers lexically diverse paraphrases without compromising on fidelity.

To obtain a set of multiple paraphrases, most of the current paraphrasing models rely solely on top-k beam search sequences. The resulting set, however, contains many structurally similar sentences with only minor, word level changes. There have been some prior works (Li and Jurafsky, 2016; Elhamifar et al., 2012) which address the notion of diversity in NLP, including in sequence learning frameworks (Song et al., 2018; Vijayakumar et al., 2018). Although Song et al. (2018) address the issue of diversity in the scenario of neural conversation models using determinantal point processes (DPP), it could be naturally used for paraphrasing. On similar lines, subset selection based on Simultaneous Sparse Recovery (SSR) (Elhamifar et al., 2012) can also be easily adapted for the same task.

Though these methods are helpful in maximizing diversity, they are restrictive in terms of re-

---

[*] Equal Contribution

[†] This research was conducted during the author's internship at the Indian Institute of Science, Bangalore.

Proceedings of NAACL-HLT 2019, pages 3609–3619
Minneapolis, Minnesota, June 2 - June 7, 2019.

taining fidelity with respect to the source sentence. Addressing the task of diverse paraphrasing through the lens of monotone submodular function maximization (Fujishige, 2005; Krause and Golovin; Bach et al., 2013) alleviates this problem and also provides a few additional benefits. Firstly, the submodular objective offers better flexibility in terms of controlling diversity as well as fidelity. Secondly, there exists a simple greedy algorithm for solving monotone submodular function maximization (Nemhauser et al., 1978), which guarantees the diverse solution to be almost as good as the optimal solution. Finally, many submodular programs are fast and scalable to large datasets.

Below, we list the main contributions of our paper.

1. We introduce **Di**verse **P**araphraser using **S**ubmodularity (DiPS). DiPS maximizes a novel submodular objective function specifically targeted towards paraphrasing.
2. We perform extensive experiments to show the effectiveness of our method in generating structurally diverse paraphrases without compromising on fidelity. We also compare against several possible diversity inducing schemes.
3. We demonstrate the utility of diverse paraphrases generated via DiPS as data augmentation schemes on multiple tasks such as intent and question classification.

We have made DiPS's source code available at https://github.com/malllabiisc/DiPS

2 Related Work

Paraphrasing a given sentence is an important problem and numerous approaches have been proposed to address it. Recently *sequence-to-sequence* based data-driven deep learning models have been proposed, which try to address the limitations of earlier traditional rule-based (McKeown, 1983) methods. Prakash et al. (2016) employ residual connections in LSTM to enhance the traditional sequence-to-sequence model. Gupta et al. (2018) provide a VAE (Kingma and Welling, 2013) based framework to improve the quality of generated paraphrases. Li et al. (2018) propose a reinforcement learning based model which uses pointer-generator (See et al., 2017) for generating paraphrases and an evaluator based on (Parikh et al., 2016) to penalize non-paraphrastic generations. Several other works (Cao et al., 2017; Iyyer et al., 2018) exist for paraphrasing, though they have either been superseded by newer models or are not-directly applicable to our settings. However, most of these methods focus on the issue of generating semantically similar paraphrases, while paying little attention towards diversity.

Diversity in paraphrasing models was first explored by (Gupta et al., 2018) where they propose to generate variations based on different samples from the latent space in a deep generative framework. Although diversity in paraphrasing models has not been explored extensively, methods have been proposed to address diversity in other NLP tasks (Li et al., 2016, 2015; Gimpel et al., 2013). Diverse beam search proposed by (Vijayakumar et al., 2018) generates k-diverse sequences by dividing the candidate subsequences at each time step into several groups and penalizing subsequences which are similar to prior groups. The most relevant to our approach is the method proposed by (Song et al., 2018) for neural conversation models where they incorporate diversity by using DPP to select diverse subsequences at each time step. Although their work is addressed in the scenario of neural conversation models, it could be naturally adapted to paraphrasing models and thus we use it as a baseline.

Submodular functions have been applied to a wide variety of problems in machine learning (Iyer and Bilmes, 2013; Jegelka and Bilmes, 2011; Krause and Guestrin, 2011; Kolmogorov and Zabih, 2002) and have recently attracted much attention in several NLP tasks including document summarization (Lin and Bilmes, 2011), data selection in machine translation (Kirchhoff and Bilmes, 2014) and goal-oriented chatbot training (Dimovski et al., 2018). However, their application to sequence generation is largely unexplored.

Data augmentation is a technique for increasing the size of labeled training sets by leveraging task specific transformations which preserve class labels. While the technique is ubiquitous in the vision community (Krizhevsky et al., 2012; Ratner et al., 2017), data-augmentation in NLP is largely under-explored. Most current augmentation schemes involve thesaurus based synonym replacement (Zhang et al., 2015; Wang and Yang, 2015), and replacement by words with paradigmatic relations (Kobayashi, 2018). Both of these

Algorithm 1: Greedy selection for submodular optimization (Cardinality constraint)

```
Input: Ground Set: $V$
       Budget: $k$
       Submodular Function: $\mathcal{F}$
$S \leftarrow \emptyset$
$N \leftarrow V$
while $|S| < k$ do
    $x^* \leftarrow \text{argmax}_{x \in N} \mathcal{F}(S \cup \{x\})$
    $S \leftarrow S \cup \{x^*\}$
    $N \leftarrow N \setminus \{x^*\}$
end
return $S$
```

Algorithm 2: DiPS

```
Input: Input Sentence: $S_{in}$
       Max. decoding length: $T$
       Submodular objective: $\mathcal{F}$
       No. of paraphrases required: $k$
Process $S_{in}$ using the encoder of SEQ2SEQ
Start the decoder with input symbol sos
$t \leftarrow 0$; $P \leftarrow \emptyset$
while $t < T$ do
    Generate top $3k$ most probable subsequences
    $P \leftarrow$ Select $k$ based on $\text{argmax}_{X \subseteq V^{(t)}} \mathcal{F}(X)$
      using Algorithm 1
    $t = t + 1$
end
return $P$
```

approaches try to boost the generalization abilities of downstream classification models through word-level substitutions. However, they are inherently restrictive in terms of the diversity they can offer. Our work offers a data-augmentation scheme via high quality paraphrases.

3 Background: Submodularity

Let V = $\{v_1, \ldots, v_n\}$ be a set of objects, which we refer to as the ground set, and $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ be a set function which works on subsets S of V to return a real value. The task is to find a subset S of bounded cardinality say $|S| \leq k$ that maximizes the function $\mathcal{F}$, i.e., $\text{argmax}_{S \subseteq V} \mathcal{F}(S)$. In general, solving this problem is intractable. However, if the function $\mathcal{F}$ is monotone non-decreasing submodular, then although the problem is still NP-complete, there exists a greedy algorithm (Algorithm 1) (Nemhauser et al., 1978) that finds an approximate solution which is guaranteed to be within 0.632 of the optimal solution.

Submodular functions are set functions $\mathcal{F} : 2^V \rightarrow \mathbb{R}$, where 2^V denotes the power set of ground set V. Submodular functions satisfy the following equivalent properties of *diminishing returns*: $\forall X, Y \subseteq V$ with $X \subseteq Y$, and $\forall s \in V \setminus Y$, we have the following.

$$\mathcal{F}(X \cup \{s\}) - \mathcal{F}(X) \geq \mathcal{F}(Y \cup \{s\}) - \mathcal{F}(Y) \quad (1)$$

In other words, the *value* addition due to incorporation of s decreases as the subset grows from X to Y. Equivalently, $\forall X, Y \subseteq V$, we have,

$$\mathcal{F}(X) + \mathcal{F}(Y) \geq \mathcal{F}(X \cup Y) + \mathcal{F}(X \cap Y)$$

In case the above inequalities are equalities, the function $\mathcal{F}$ is said to be modular. Let $\mathcal{F}(s|X) \triangleq \mathcal{F}(X \cup \{s\}) - \mathcal{F}(X)$. Therefore, $\mathcal{F}$ is submodular if $\mathcal{F}(s|X) \geq \mathcal{F}(s|Y)$ for $X \subseteq Y$.

The second criteria which the function needs to satisfy for Algorithm 1 to be applicable is of monotonicity. A set function $\mathcal{F}$ is said to be monotone non-decreasing if $\forall X \subseteq Y, \mathcal{F}(X) \leq \mathcal{F}(Y)$.

Submodular functions are relevant in a large class of real-world applications, therefore making them extremely useful in practice. Additionally, submodular functions share many commonalities with convex functions, in the sense that they are closed under a number of standard operations like mixtures (non-negative weighted sum of submodular functions), truncation and some restricted compositions.

The above properties will be useful when defining the submodular objective for obtaining high quality paraphrases.

4 Methodology

Similar to Prakash et al. (2016); Gupta et al. (2018); Li et al. (2018), we formulate the task of paraphrase generation as a sequence-to-sequence learning problem. Previous SEQ2SEQ based approaches depend entirely on the standard cross-entropy loss to produce semantically similar sentences and greedy decoding during generation. However, this does not guarantee lexical variety in the generated paraphrases. To incorporate some form of diversity, most prior approaches rely solely on top-k beam search sequences. The k-best list generated by standard beam search are a poor surrogate for the entire search space (Finkel et al., 2006). In fact, most of the sentences in the resulting set are structurally similar, differing only by punctuations or minor morphological variations.

While being similar in the encoding scheme, our work adopts a different approach for the final decoding. We propose a framework which organi-

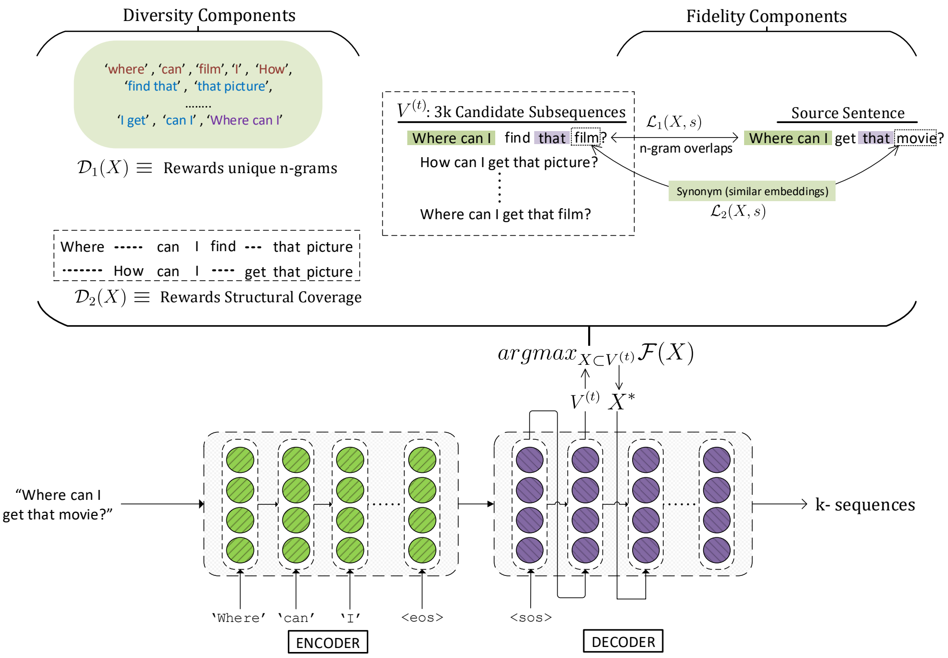


Figure 1: Overview of DiPS during decoding to generate k paraphrases. At each time step, a set of N sequences ($V^{(t)}$) is used to determine $k < N$ sequences (X^*) via submodular maximization . The above figure illustrates the motivation behind each submodular component. Please see Section 4 for details.

cally combines a sentence encoder with a diversity inducing decoder.

4.1 Overview

Our approach is built upon SEQ2SEQ framework. We first feed the tokenized source sentence to the encoder. The task of the decoder is to take as input the encoded representation and produce the respective paraphrase. To achieve this, we train the model using standard cross entropy loss between the generated sequence and the target paraphrase. Upon completion of training, instead of using greedy decoding or standard beam search, we use a modified decoder where we incorporate a submodular objective to obtain high quality paraphrases. Please refer to Figure 1 for an overview of the proposed method.

During the generation phase, the encoder takes the source sentence as input and feeds its representation to the decoder to initiate the decoding process. At each time-step t, we consider N most probable subsequences since they are likely to be well-formed. Based on optimization of our submodular objective, a subset of size $k < N$ are selected and sent as input to the next time step $t + 1$ for further generation. The process is repeated until desired output length T or <eos> token, whichever comes first.

4.2 Monotone Submodular Objectives

We design a parameterized class of submodular functions tailored towards the task of paraphrase generation. Let $V^{(t)}$ be the ground set of possible subsequences at time step t. We aim to determine a set $X \subseteq V^{(t)}$ that retains certain *fidelity* as well as *diversity*. Hence, we model our submodular objective function as follows:

$$X^* = \underset{X \subseteq V^{(t)}}{\operatorname{argmax}} \mathcal{F}(X) \quad s.t. \; |X| \leq k \qquad (2)$$

where k is our budget (desired number of paraphrases) and $\mathcal{F}$ is defined as:

$$\mathcal{F}(X) = \lambda \mathcal{L}(X, s) + (1 - \lambda)\mathcal{D}(X) \qquad (3)$$

Here s is the source sentence, $\mathcal{L}(X, s)$ and $\mathcal{D}(X)$ measure *fidelity* and *diversity*, respectively. $\lambda \in [0, 1]$ is the trade-off coefficient. This formulation clearly brings out the trade-off between the two key characteristics.

Fidelity

It is imperative to design functions that exploit the decoder search space to maximize the semantic similarity between the generated and the source sentence. To achieve this we build upon a known class of monotone submodular functions (Stobbe and Krause, 2010)

$$f(X) = \sum_{i \in U} \mu_i \phi_i(m_i(X)) \tag{4}$$

where U is the set of features to be defined later, $\mu_i \geq 0$ is the feature weight, $m_i(X) = \sum_{x \in X} m_i(x)$ is non-negative modular function and ϕ_i is a non-negative non-decreasing concave function. Based on the analysis of concave functions in (Kirchhoff and Bilmes, 2014), we use the simple square root function as ϕ ($\phi(a) = \sqrt{a}$) in both of our fidelity objectives defined below.

We consider two complementary notions of sentence similarity namely syntactic and semantic. To capture syntactic information we define the following function:

$$\mathcal{L}_1(X, s) = \mu_1 \sqrt{\sum_{x \in X} \sum_{n=1}^{N} \beta^n \left| x_{\text{n-gram}} \cap s_{\text{n-gram}} \right|} \tag{5}$$

where $|x_{\text{n-gram}} \cap s_{\text{n-gram}}|$ represents the number of overlapping n-grams between the source and the candidate sequence x for different values of $n \in \{1, \ldots, N\}$(we use $N = 3$). Since longer n-gram overlaps are more valuable, we set $\beta > 1$. This function inherently increases the BLEU score between the source and the generated sentences.

We address the semantic aspect of fidelity by devising a function based on the word embeddings of source and generated sentences. We define embedding based similarity between two sentences as,

$$\mathcal{S}(x, s) = \frac{1}{|x|} \sum_{w_i \in x} \underset{w_j \in s}{\text{argmax}} \ \psi(\mathbf{v}_{w_i}, \mathbf{v}_{w_j}) \tag{6}$$

where $\mathbf{v}_{w_i}$ is the word embedding for token w_i and $\psi(\mathbf{v}_{w_i}, \mathbf{v}_{w_j})$ is the gaussian radial basis function (rbf)[1]. For each word in the candidate sequence x, we find the best matching word in the source sentence using word level similarity. Using the above mentioned measure for embedding similarity we use the following submodular function:

$$\mathcal{L}_2(X, s) = \mu_2 \sqrt{\sum_{x \in X} \mathcal{S}(x, s)} \tag{7}$$

This function helps increase the semantic homogeneity between the source and generated sequences. The above defined functions (Equation 5,7) are compositions of non-decreasing concave functions and modular functions. Thus, staying in the realm of the class of monotone submodular functions mentioned in Equation 4, we define fidelity function $\mathcal{L}(X, s) = \mathcal{L}_1(X, s) + \mathcal{L}_2(X, s)$

Diversity

Ensuring high fidelity often comes at the cost of producing sequences that only slightly differ from each other. To encourage diversity in the generation process it is desirable to reward sequences with higher number of distinct n-grams as compared to others in the ground set $V^{(t)}$. Accordingly, we propose to use the following function:

$$\mathcal{D}_1(X) = \mu_3 \sum_{n=1}^{N} \beta^n \left| \bigcup_{x \in X} x_{n-gram} \right| \tag{8}$$

For $\beta = 1$, $\mathcal{D}_1(X)$ denotes the number of distinct n-grams present in the set X. Since shorter n-grams contribute more towards diversity, we set $\beta < 1$, thereby giving more value to shorter n-grams. It is easy to see that this function is monotone non-decreasing as the number of distinct n-grams can only increase with the addition of more sequences. To see that $\mathcal{D}_1(X)$ is submodular, consider adding a new sequence to two sets of sequences, one a subset of the other. Intuitively, the increment in the number of distinct n-grams when adding a new sequence to the smaller set should be larger than the increment when adding it to the larger set, as the distinct n-grams in the new sequence might have already been covered by the sequences in the larger set.

Apart from distinct n-gram overlaps, we also wish to obtain sequence candidates that are not only diverse, but also cover all major structural variations. It is reasonable to expect sentences that are structurally different to have lower degree of word/phrase alignment as compared to sentences with minor lexical variations. Edit distance (Levenshtein) is a widely accepted measure to determine such dissimilarities between two sentences. To incorporate this notion of diversity, a formulation in terms of edit distance seems like a natural fit for the problem. To do so, we use the coverage function which measures the similarity of the candidate sequences X with the ground set $V^{(t)}$. The

---

[1] We find gaussian rbf to work better than other similarity metrics such as cosine similarity

coverage function is naturally monotone submodular and is defined as:

$$\mathcal{D}_2(X) = \mu_4 \sum_{x_i \in V^{(t)}} \sum_{x_j \in X} \mathcal{R}(x_i, x_j) \tag{9}$$

where $\mathcal{R}(x_i, x_j)$ is an alignment based similarity measure between two sequences x_i and x_j given by:

$$\mathcal{R}(x_i, x_j) = 1 - \frac{\texttt{EditDistance}(x_i, x_j)}{|x_i| + |x_j|} \tag{10}$$

Note that $\mathcal{R}(x_i, x_j)$ will always lie in the range $[0, 1]$.

Evidently, this method allows flexibility in terms of controlling diversity and fidelity. Our goal is to strike a balance between these two to obtain high-quality generations.

5 Experiments

5.1 Datasets

In this section we outline the datasets used for evaluating our proposed method. We specify the actual splits in Table 2. Based on the task, we categorize them into the following:

1. **Intrinsic evaluation**: To demonstrate the efficacy of our method on fidelity and diversity, we use the standard *Quora question pair*[2] dataset and the *Twitter URL paraphrasing* (Lan et al., 2017) dataset.

   We train and evaluate the paraphrase generation model on a subset of *Quora question pair* dataset which we refer to as *Quora-Div*. This subset comprises only positive examples (pairs which have been annotated as paraphrases).

   We additionally perform in-domain data augmentation for the task of paraphrase recognition. For that, we augment sentences generated through different paraphrasing model as positive samples to the *Quora-PR* dataset. *Quora-PR* is a subset of *Quora question pair* dataset which contains positive as well as negative examples.

2. **Data-augmentation**: We exhibit the importance of samples generated through our method on the task of Data-augmentation using three primary datasets. *SNIPS (Coucke et al., 2018)*, *Yahoo-L31*[3] is used for intent-classification and *TREC* (Li and Roth, 2002) is used for question classification. Each dataset is balanced in terms of the number of samples per classes.

Dataset	Task	Train	Val.	Test	Classes
Quora-Div	Intrinsic	120K	20K	5K	N/A
Twitter	Intrinsic	100K	15K	3K	N/A
Quora-PR	Intrinsic	40K	10K	40K	2
DATA-AUGMENTATION					
SNIPS	Intent	10k	1k	700	7
Yahoo-L31	Intent	4K	1K	1K	2
TREC	Question	1K	200	500	6

Table 2: Dataset Statistics

5.2 Baseline

Several models have sought to increase diversity, albeit with different goals and techniques. However, majority of the prior works in this area have focused on the task of producing diverse responses in dialog systems (Li et al., 2016; Ritter et al., 2011) and not paraphrasing. Given the lack of relevant baselines, we compare our model against the following methods:

1. **SBS**: Decoder which performs standard beam search during generation.
2. **DBS**: An alternative of beam search to incorporate diversity. (Vijayakumar et al., 2018)
3. **DPP**: Decoder based on Determinantal Point Processes (Kulesza et al., 2012)
4. **SSR**[4]: Decoder based on Subset Selection using Simultaneous Sparse Recovery (Elhamifar et al., 2016)

   We additionally, evaluate against the following paraphrase generation models:

5. **VAE-SVG**: VAE based generative framework for paraphrase generation. (Gupta et al., 2018)
6. **RbM**: Deep Reinforcement learning based paraphrase generation model. (Li et al., 2018)

Note that the first four baselines are trained using the same SEQ2SEQ network and differ only in the decoding phase.

---

[2] https://www.kaggle.com/c/quora-question-pairs

[3] https://webscope.sandbox.yahoo.com/

[4] Exact formulation of the SSR and DPP can be found in the supplementary section.

Model	Quora-Div			Twitter		
	BLEU↑	METEOR↑	TERp↓	BLEU↑	METEOR↑	TERp↓
SBS	33.1	28.2	55.6	51.1	23.5	67.9
DBS (Vijayakumar et al., 2018)	30.9	28.3	57.5	47.1	22.1	69.0
VAE-SVG (Gupta et al., 2018)	33.4	25.6	63.2	46.7	25.2	67.1
RbM (Li et al., 2018)	29.4	29.5	62.5	47.7	29.3	68.7
DPP	30.5	27.9	57.3	44.8	21.4	71.4
SSR	28.7	26.8	58.7	41.3	20.0	74.4
DiPS (Ours)	**35.1**	**29.7**	**53.2**	**55.3**	**30.1**	**63.5**

Table 3: Results on Quora-Div and Twitter dataset. Higher↑ BLEU and METEOR score is better whereas lower↓ TERp score is better. Please see Section 6 for details.

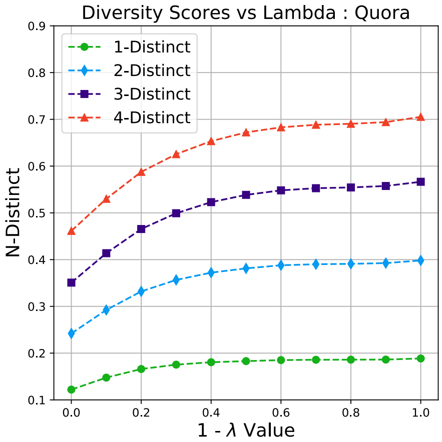


Figure 2: Effect of varying the trade-off coefficient λ in DiPS on various diversity metrics on the Quora dataset.

5.3 Intrinsic Evaluation

1. **Fidelity**: To evaluate our method for fidelity of generated paraphrases, we use three machine translation metrics which have been shown to be suitable for paraphrase evaluation task (Wubben et al., 2010): BLEU (Papineni et al., 2002)(upto bigrams), METEOR (Banerjee and Lavie, 2005) and TERPlus (Snover et al., 2009).
2. **Diversity**: We report degree of diversity by calculating the number of distinct n-grams (n $\in \{1, 2, 3, 4\}$). The value is scaled by the number of generated tokens to avoid favoring long sequences.

In addition to fidelity and diversity, we evaluate the efficacy of our method by using the generated paraphrases as augmented samples in the task of paraphrase recognition on the *Quora-PR* dataset. We perform experiments with multiple augmentation settings for the following classifiers:

1. **LogReg**: Simple Logistic Regression model. We use a set of hand-crafted features, the details of which can be found in the supplementary.
2. **SiameseLSTM**: Siamese adaptation of LSTM to measure quality between two sentences (Mueller and Thyagarajan, 2016)

We also perform ablation testing to highlight the importance of each submodular component. Details can be found in the supplementary section.

5.4 Data-Augmentation

We evaluate the importance of using high quality paraphrases in two downstream classification tasks namely intent-classification and question-classification. Our original generation model is trained on *Quora-Div* question pairs. Since intent-classification and question-classification contain questions, this setting seems like a good fit to perform transfer learning. We perform experiments on the following standard classifier models:

1. **LogRegDA**: Simple logistic regression model trained using hand-crafted features. For details, please refer to the supplementary section.
2. **LSTM**: Single layered LSTM classification model.

In addition to SBS and DBS, we use the following data-augmentation baselines for comparison:

1. **SynRep** : Simple synonym replacement
2. **ContAug**: Data-augmentation scheme based on replacement of words with similar paradigmatic relations. (Kobayashi, 2018)

5.5 Setup

We train our SEQ2SEQ model with attention (Bahdanau et al., 2014) for up to 50 epochs using the adam optimizer (Kingma and Ba, 2014) with initial learning rate set to 2e-4. During the generation phase, we follow standard beam search till

	Quora-Div				Twitter			
Model	**1-distinct**	**2-distinct**	**3-distinct**	**4-distinct**	**1-distinct**	**2-distinct**	**3-distinct**	**4-distinct**
SBS	12.8	24.8	35.3	46.6	20.0	30.9	38.1	44.6
VAE-SVG (Gupta et al., 2018)	15.8	22.5	27.6	31.8	19.3	28.2	33.3	37.2
DBS (Vijayakumar et al., 2018)	17.9	33.7	44.8	54.9	25.8	40.7	48.2	53.9
DPP	17.1	34.4	49.1	62.6	25.6	41.4	51.1	59.0
SSR	16.6	32.8	47.1	60.7	26.6	43.7	54.0	62.4
DiPS (Ours)	**18.1**	**37.2**	**52.3**	**65.3**	**28.3**	**46.6**	**56.7**	**64.5**

Table 4: Results on Quora-Div and Twitter dataset. Higher distinct scores imply better lexical diversity. Please see Section 6 for details.

	LogRegDA			LSTM		
Model	**YahooL31**	**TREC**	**SNIPS**	**YahooL31**	**TREC**	**SNIPS**
NoAug	62.7	82.2	93.4	64.8	94.2	94.7
SBS	63.6	84.6	93.8	65.4	94.4	94.7
DBS	63.3	84.2	94.1	65.6	95.2	96.1
SynRep	63.7	85.2	93.9	65.3	93.6	95.5
ContAug	63.8	86.0	95.3	66.3	95.8	96.4
DiPS(Ours)	**64.9**	**86.6**	**96.0**	**66.7**	**96.4**	**97.1**

Table 5: Accuracy scores of two classification models on various data-augmentation schemes. Please see Section 6 for details

the number of generated tokens is nearly half the source sequence length (token level) to avoid possibly erroneous sentences. We then apply submodular maximization stochastically with probability p at each time step. Since each candidate subsequence is extended by a single token at every time-step, information added might not necessarily be useful as our submodular components work on sentence level. This approach is time efficient and avoids redundant computations.

For each augmentation setting, we randomly select sentences from the training data and generate its paraphrases. We then add them in the training data with the same label as that of the source sentence. We evaluate the performance on different classification models in terms of accuracy.

Based on the formulation of the objective function, it should be clear that diversity would attain maximum value at (or around) $\lambda = 0$ albeit at the cost of fidelity. This is certainly not a desirable property for paraphrasing systems. To address this, we perform hyperparameter tuning for λ value by analyzing the trade-off between diversity and fidelity based on varying λ values. In practice, diversity metric attains saturation at certain λ range (usually 0.2 - 0.5). This behaviour can be seen in Figure 2. Corresponding plot for Twitter, the effect of λ on fidelity and additional details about the hyperparameters can be found in the supplementary.

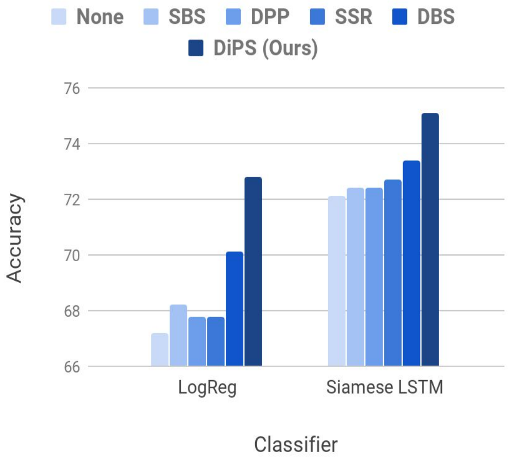


Figure 3: Comparison of accuracy scores of two paraphrase recognition models using different augmentation schemes (Quora-PR). Both LogReg and SiameseLSTM achieve the highest boost in performance when augmented with samples generated using DiPS

6 Results

Our experiments were geared towards answering the following primary questions:

Q1. Is DiPS able to generate diverse paraphrases without compromising on fidelity? (Section 6.1)

Q2. Are paraphrase generated by DiPS useful in data-augmentation? (Section 6.2)

6.1 Intrinsic Evaluation

We compare our method against recent paraphrasing models as well as multiple diversity inducing schemes. DiPS outperforms these baseline models in terms of fidelity metrics namely BLEU, METEOR and TERp. A high METEOR score and a low TERp score indicate the presence of not only exact words but also synonyms and semantically similar phrases. Notably, our model is not only able to achieve substantial gains over other diversity inducing schemes but is also able to do so without compromising on fidelity. Diversity and fidelity scores are reported in Table 4 and Table 3, respectively.

As described in Section 5.3, we evaluate the accuracy of paraphrase recognition models when provided with training data augmented using different schemes. It is reasonable to expect that high quality paraphrases would tend to yield better results on in-domain paraphrase recognition task. We observe that using the paraphrases generated by DiPS helps in achieving substantial gains in accuracy over other baseline schemes. Figure 3 showcases the effect of using paraphrases generated by our method as compared to other competitive paraphrasing methods.

6.2 Data-augmentation

Data Augmentation results for intent and question classification are shown in Table 5. While, SBS does not offer much lexical variability, DBS offers high diversity at the cost of fidelity. SynRep and ContAug are augmentation schemes which are limited by the amount of structural variations they can offer. DiPS on the other hand provides generation having high structural variations without compromising on fidelity. The boost in accuracy scores on both the types of classification models is indicative of the importance of using high quality paraphrases for data-augmentation.

7 Conclusion

In this paper, we have proposed DiPS, a model which generates high quality paraphrases by maximizing a novel submodular objective function designed specifically for paraphrasing. In contrast to prior works which focus exclusively either on fidelity or diversity, a submodular function based approach offers a large degree of freedom to control fidelity and diversity. Through extensive experiments on multiple standard datasets, we have demonstrated the effectiveness of our approach over numerous baselines. We observe that the diverse paraphrases generated are not only interesting and meaning preserving, but are also helpful in data augmentation. We showcase that using multiple settings on the task of intent and question classification. We hope that our approach will be useful not only for paraphrase generation and data augmentation, but also for other NLG problems in conversational agents and text summarization.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This research is supported in part by the Ministry of Human Research Development (Government of India), Amazon and travel gifts from Microsoft Research (MSR) India and the Tata Trusts.

References

Peter G Anick and Suresh Tipirneni. 1999. The paraphrase search assistant: terminological feedback for iterative information seeking. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 153–159. ACM.

Francis Bach et al. 2013. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Delphine Bernhard and Iryna Gurevych. 2008. Answering learners' questions by retrieving question paraphrases from social q&a sites. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 44–52. Association for Computational Linguistics.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Mladen Dimovski, Claudiu Musat, Vladimir Ilievski, Andreea Hossman, and Michael Baeriswyl. 2018. Submodularity-inspired data selection for goal-oriented chatbot training based on sentence embeddings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4019–4025. International Joint Conferences on Artificial Intelligence Organization.

Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. 2016. Dissimilarity-based sparse subset selection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2182–2197.

Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. 2012. Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery. In *Advances in Neural Information Processing Systems*, pages 19–27.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1608–1618.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

S Fujishige. 2005. Submodular functions and optimization. *Annals of Discrete Mathematics*, 58.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI Conference on Artificial Intelligence*.

Rishabh K Iyer and Jeff A Bilmes. 2013. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1875–1885.

Stefanie Jegelka and Jeff Bilmes. 2011. Submodularity beyond submodular energies: coupling edges in graph cuts.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Katrin Kirchhoff and Jeff Bilmes. 2014. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 131–141.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 452–457.

Vladimir Kolmogorov and Ramin Zabih. 2002. What energy functions can be minimized via graph cuts? In *European conference on computer vision*, pages 65–81. Springer.

Andreas Krause and Daniel Golovin. Submodular function maximization.

Andreas Krause and Carlos Guestrin. 2011. Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):32.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th international conference on World wide web*, pages 71–80. ACM.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity.

Preksha Nema, Mitesh M Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1063–1072.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming*, 14(1):265–294.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems*, pages 3239–3249.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or hter?: exploring different human judgments with a tunable mt metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268. Association for Computational Linguistics.

Yiping Song, Rui Yan, Yansong Feng, Yaoyuan Zhang, Dongyan Zhao, and Ming Zhang. 2018. Towards a neural conversation model with diversity net using determinantal point processes. In *AAAI*.

Peter Stobbe and Andreas Krause. 2010. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems*, pages 2208–2216.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. AAAI Conference on Artificial Intelligence.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *INLGC*, pages 203–207. ACL.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Let's Make Your Request More Persuasive: Modeling Persuasive Strategies via Semi-Supervised Neural Nets on Crowdfunding Platforms

Diyi Yang[*], **Jiaao Chen**[*], **Zichao Yang**, **Dan Jurafsky**, **Eduard Hovy**
Georgia Institute of Technology, Carnegie Mellon University, Stanford University
`diyi.yang@cc.gatech.edu`
`{jiaaoc, zichaoy, hovy}@andrew.cmu.edu`
`jurafsky@stanford.edu`

Abstract

Modeling what makes a request persuasive—eliciting the desired response from a reader—is critical to the study of propaganda, behavioral economics, and advertising. Yet current models can't quantify the persuasiveness of requests or extract successful persuasive strategies. Building on theories of persuasion, we propose a neural network to quantify persuasiveness and identify the persuasive strategies in advocacy requests. Our semi-supervised hierarchical neural network model is supervised by the number of people persuaded to take actions and partially supervised at the sentence level with human-labeled rhetorical strategies. Our method outperforms several baselines, uncovers persuasive strategies—offering increased interpretability of persuasive speech—and has applications for other situations with document-level supervision but only partial sentence supervision.

1 Introduction

Crowdfunding platforms are a popular way to raise funds for projects. For example, Kiva, a peer-to-peer lending platform, has crowd-funded more than a million loans, totaling over $1 billion since 2005. Kickstarter, another online crowdfunding platform, successfully funded 110,270 projects with a total of over 2 billion dollars. Yet most projects still suffer from low success rates. How can we help requesters craft persuasive and successful pitches to convince others to take actions?

Persuasive communication has the potential to shape and change people's attitudes and behaviors (Hovland et al., 1953), and has been widely researched in various fields such as social psychology, marketing, behavioral economics, and political campaigning (Shrum et al., 2012). One of the most influential theories in the advertising literature is Chaiken's systematic-heuristic dual processing theory, which suggests that people process persuasive communication by evaluating the quality of arguments or by relying on inferential rules. Some such heuristic rules are commonly used in consumer behaviors; commercial websites may highlight the limited availability of their items "*In high demand - only 2 left on our site!*" or emphasize the person in authority "*Speak to our head of sales—he has over 15 years' experience selling properties*" to attract potential consumers. Although numerous studies on persuasion have been conducted (Chaiken, 1980), we still know little about the way how persuasion functions in the wild and how it can be modeled computationally.

In this work, we utilize neural-network based methods to computationally model persuasion in requests from crowdfunding websites. We build on theoretical models of persuasion to operationalize persuasive strategies and ensure generalizability. We propose to identify the persuasive strategy employed in each sentence in each request. However, constructing a large dataset with persuasion strategies labeled at the sentence level is time-consuming and expensive. Instead, we propose to use a small amount of hand-labeled sentences together with a large number of requests automatically labeled at the document level by the number of persuaded support actions. Our model is a semi-supervised hierarchical neural network that identifies the persuasive strategies employed in each sentence, where the supervision comes from the overall persuasiveness of the request. We propose that the success of requests could have substantive explanatory power to uncover their persuasive strategies. We also introduce an annotated corpus with sentence-level persuasion strategy labels and document-level persuasiveness labels, to facilitate future work on persuasion. Experiments

---

[*] Equal contribution. This work was done when the first two authors were students at CMU.

Proceedings of NAACL-HLT 2019, pages 3620–3630
Minneapolis, Minnesota, June 2 - June 7, 2019.

show that our semi-supervised model outperforms several baselines. We then apply this automated model to unseen requests from different domains and obtain nuanced findings of the importance of different strategies on persuasion success. Our model can be useful in any situation in which we have exogenous document-level supervision, but only small amounts of expensive human-annotated sentence labels.

2 Related Work

Computational argumentation has received much recent attention (Ghosh et al., 2016; Stab and Gurevych, 2017; Peldszus and Stede, 2013; Stab et al., 2018; Ghosh et al., 2014). Most work has either identified the arguments in news articles (Sardianos et al., 2015) or user-generated web content (Habernal and Gurevych, 2017; Musi et al., 2018), or classified argument components (Zhang and Litman, 2015) into claims and premises, supporting and opposing claims, or backings, rebuttals and refutations . For example, Stab and Gurevych (2014) proposed structural, lexical, syntactic and contextual features to identify convincing components of Web arguments including claim, major claim, and premise. Similarly, Zhang and Litman (2015) studied student essay revisions and classified a set of argumentative actions associated with successful writing such as warrant/reasoning/backing, rebuttal/reservation, and claims/ideas. Habernal and Gurevych (2016) investigated the persuasiveness of arguments in any given argument pair using bidirectional LSTM. Hidey et al., (2017) utilized the persuasive modes—ethos, logos, pathos—to model premises and the semantic types of argument components in an online persuasive forum.

While most computational argumentation focuses on the relational support structures and factual evidence to make claims, **persuasion** focuses more on language cues aimed at shaping, reinforcing and changing people's opinions and beliefs. How language changes people's attitudes and behaviors have received less attention from the computational community than argumentation, although there have been important preliminary work (Persing and Ng, 2017; Carlile et al., 2018). Farra et al., (2015) built regression models to predict essay scores based on features extracted from opinion expressions and topical elements. Chatterjee et al., (2014) used verbal descriptors and para-verbal markers of hesitation to predict speakers' persuasiveness on website housing videos of product reviews. When looking at persuasion in the context of online forum discussions (Wei et al., 2016), Tan et al., (2016) found that on the Change My View subreddit, interaction dynamics such as the language interplay between opinion holders and other participants provides highly predictive cues for persuasiveness. Using the same dataset, Wel et al., (2016) extracted a set of textual information and social interaction features to identify persuasive posts.

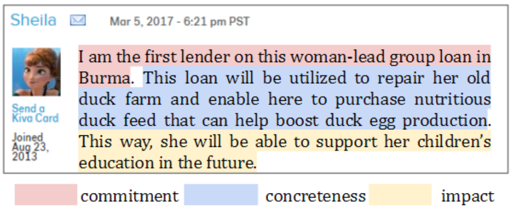
Recently, Pryzant et al., (2017) introduced a neural network with an adversarial objective to select text features that are predictive of some outcomes but decorrelated with others and further analyzed the narratives highlighted by such text features. Further work extended the model to induce narrative persuasion lexicons predictive of enrollment from course descriptions and sales from product descriptions (Pryzant et al., 2018a), and the efficacy of search advertisements (Pryzant et al., 2018b). Similar to their settings, we use the outcomes of a persuasive description to supervise the learning of persuasion tactics, and our model can similarly induce lexicons associated with successful narrative persuasion by examining highly attentional words associated with persuasion outcomes. Our work differs both in our semi-supervised method and also because we explicitly draw on the theoretical literature to model the persuasion strategy for each sentence in requests, allowing requests to have multiple persuasion strategies; our induced lexicons can thus be very specific to different persuasion strategies.

Other lines of persuasion work predict the success of requests on peer-to-peer lending or crowdfunding platforms, and mainly exploit request attributes like project description (Greenberg et al., 2013), project videos (Dey et al., 2017), and social predictors such as the number of backers (Etter et al., 2013) or specific types of project updates (Xu et al., 2014). Among them, only a few investigated the effect of language on the success of requests. Althoff et al., (2014) studied donations in Random Acts of Pizza on Reddit, using the social relations between recipient and donor plus linguistic factors to predict the success of these altruistic requests. Based on a corpus of 45K crowd-funded projects, Mitra and Gilbert (2014) found that 9M phrases commonly present in crowd-funding have reasonable predictive power in accounting for variance around successful funding, suggesting that language does exhibit some general principles of persuasion. Although this prior work offers predictive and insightful models, most studies chose their persuasion labels or variables without reference to a taxonomy of persuasion techniques nor to a principled method of choosing them. Some exceptions include Yang and Kraut (2017), Dey et al., (2017), and Rosenthal and McKeown (2017). For example, Yang and Kraut (2017) looked at the effectiveness of a set of persuasive cues in Kiva requests and found that certain heuristic cues are positively correlated with lenders' contribution.

Inspired by these prior work, we operationalize persuasive strategies based on theories of persuasion and aim to learn local structures/labels of sentences based on the global labels of paragraphs/requests. Our task is different from most previous work on semi-supervised learning for NLP (Liang, 2005; Yang et al., 2017) that focuses on the setting with partial data labels. While in computer vision, there is a lot of prior work in using image global labels to uncover local pixel level labels and bounding boxes of objects (Oquab et al., 2015; Pinheiro and Collobert, 2015), the investigation of this task in NLP, to the best of our knowledge, is novel and could potentially have much broader applications.

3 Research Context

We situate this research within the team forums of Kiva[1], the largest peer-to-peer lending website. These self-organized lending teams are built around common interests, school affiliation or location. In such teams, members can post messages in their team discussion board to persuade other members to lend to a particular borrower. One such message is shown in Figure 1. A borrower, Sheila, posted a message on Kiva to request loans for woman-led group. As highlighted in the figure, she made use of several persuasion strategies such as commitment, concreteness, and impact to render her request more persuasive. We define the *persuasiveness* score of a request message as the number of team members (in log-scale) who read the message and make loans to the mentioned borrower. We then regard this overall persuasiveness of messages as high-level supervision for training our model to determine which persuasion strategy is used in each sentence inside each message.



I am the first lender on this woman-lead group loan in Burma. This loan will be utilized to repair her old duck farm and enable here to purchase nutritious duck feed that can help boost duck egg production. This way, she will be able to support her children's education in the future.

Figure 1: An anonymized advocating message that persuaded 5 members to lend to the mentioned borrower. Persuasion strategies are highlighted.

4 Persuasion Strategies

Numerous studies have investigated the basic principles that govern getting compliance from people (Cialdini and Garde, 1987; Petty et al., 1983). In this work, we utilized Chiaken's 1980 systemic-heuristic model of social information processing, which suggests that people process persuasive requests by assessing the quality of arguments (systematic processing) or by relying on heuristic rules (heuristic processing). Building on that, we first borrow several commonly used heuristic principles (Cialdini and Garde, 1987) that are also suitable for our context as below.

- **Scarcity** states that people tend to value an item more as soon as it becomes rare, distinct or limited. For example, take the use of 'expire'in this message: *"This loan is going to expire in 35 mins..."*.
- The principle of **Emotion** says that making messages full of emotional valence and arousal affect (e.g., describing a miserable situation or a happy moment) can make people care and act, e.g., "*The picture of widow Bunisia holding one of her children in front of her meager home brings tears to my eyes..*", similar to *Sentiment* and *Politeness* used by Althoff et al., (2014) and Tan et al., (2016), and *Pathos* used by Hidey et al., (2017).
- **Commitment** states that once we make a choice, we will encounter pressures that cause us to respond in ways that justify our earlier decision, and to convince others that we have made the correct choice. Here it could be mentioning their contribution in the message, e.g., *"I loaned to her already."*
- **Social Identity** refers to people's self-concept of their membership in a social

---

[1] `https://www.kiva.org/`

group, and people have an affinity for their groups over others, similar to *name mentions* in Rosenthal and McKeown (2017). Thus if a loan request comes from their own groups, they are more likely to contribute, such as *"For those of you in our team who love bread, here is a loan about bakery."*

- **Concreteness** refers to providing concrete facts or evidence, such as "*She wishes to have a septic tank and toilet, and is 51% raised and needs $825*", similar to *Claim* and *Evidence* (Zhang et al., 2016; Stab and Gurevych, 2014)), *Evidentiality* (Althoff et al., 2014), and *Logos* (Hidey et al., 2017).

We also propose a new strategy to capture importance or impact on these requests:

- **Impact and Value** emphasizes the importance or bigger impact of this loan, such as *"... to grow organic rice. Then, she can provide better education for her daughter"*.

Note that other persuasion tactics such as *Reciprocity* — "feel obligated to return something after receiving something of value from another" — and *Authority* — "comply with the requests of authority in an unthinking way to guide their decisions"— are also widely used in persuasive communication. However, in this context, we did not observe enough instances of them.

5 Semi-supervised Neural Net

Given a message $M = \{S_0, S_1, ..., S_L\}$ consisting of L sentences that the author posted to advocate for a loan, our task is to predict the persuasion strategies p_i employed in each sentence S_i, $i \in [0, L]$. However, purely constructing a large-scale dataset that contains such labels of sentence-level persuasion strategy is often time-consuming and expensive. Instead, we propose to utilize a small amount of labeled and a large amount of unlabeled data. We design a semi-supervised hierarchical neural network to identify the persuasive strategies employed in each sentence, where the supervision comes from the sentence-level labels g in a small portion of data and the overall persuasiveness scores y of messages. The overall architecture of our method is shown in Figure 2.

5.1 Sentence Encoder

Given a sentence S_i with words $w_{i,j}$, $j \in [0, l]$ and l is the sentence length, a GRU (Bahdanau et al., 2014) is used to incorporate contextual cues of words into hidden state $h_{i,j}$. This GRU reads the sentence S_i from $w_{i,1}$ to $w_{i,l}$ and encodes each word $w_{i,j}$ with its context into hidden state $h_{i,j}$:

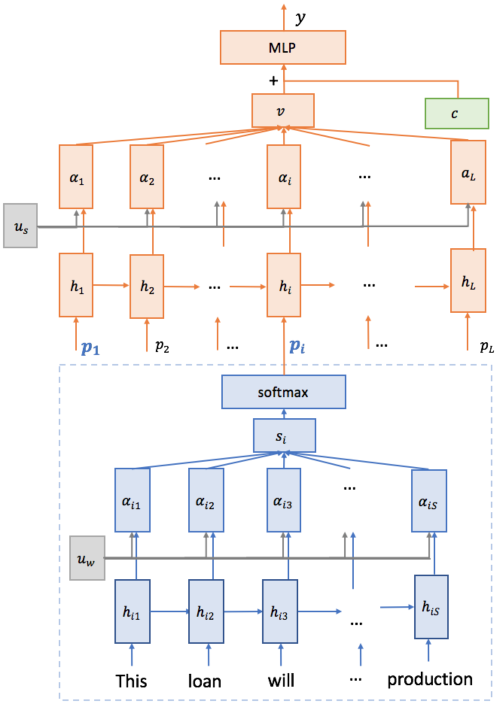


Figure 2: The overall model architecture. The blue part describes the sentence encoder. Sentences with labels of persuasion strategies are highlighted with dark blue like p_1. The orange part shows the document encoder.

$$h_{i,j} = \text{GRU}(W_e w_{i,j}, h_{i,j-1}), j \in [0, l]. \quad (1)$$

where W_e is the word embedding matrix. To learn the characteristic words associated with the persuasive strategy in a sentence, we apply an attention mechanism (Bahdanau et al., 2014; Yang et al., 2016). The representation of those words are then aggregated to form the sentence vector s_i. We formulated this word level attention as follows:

$$u_{i,j} = \tanh(W_w h_{i,j} + b_w) \quad (2)$$

$$\alpha_{i,j} = \frac{\exp(u_{i,j}^\intercal u_w)}{\sum_k \exp(u_{i,k}^\intercal u_w)} \quad (3)$$

$$s_i = \sum_j \alpha_{i,j} h_{i,j} \quad (4)$$

where u_w is a context vector that queries the characteristic words associated with different persuasion strategies. It is randomly initialized and jointly learned from data.

5.2 Latent Persuasive Strategies

We assume that each sentence instantiates only one type of persuasion strategy. For example, a sentence *"She is 51% raised and needs $825 in 3 days"* employs *Scarcity*, trying to emphasize limited time availability. We propose to use the high level representation of each sentence to predict the latent variable:

$$p_i = \text{softmax}(W_v s_i + b_v) \tag{5}$$

5.3 Document Encoder

After obtaining the sentence vector p_i, we can get a document vector in a similar way:

$$h_i = \text{GRU}(p_i, h_{i-1}), i \in [0, L] \tag{6}$$

where L denotes the number of sentences in a message. Similarly, we introduce an attention mechanism to measure the importance of each sentence and its persuasion strategy via a context vector u_s:

$$u_i = \tanh(W_s h_i + b_s) \tag{7}$$

$$\alpha_i = \frac{\exp(u_i^\intercal u_s)}{\sum_k \exp(u_k^\intercal u_s)} \tag{8}$$

$$v = \sum_i \alpha_i h_i \tag{9}$$

5.4 Semi-Supervised Learning Objective

The document vector v is a high-level representation of the document and can be used as a set of features for predicting $\tilde{y}$, the persuasiveness of a message, i.e., how many team members will make loans to the project mentioned in this message. We also include a context vector c to further assist the prediction of making loans. For instance, c could represent the number of team members in a team, the total amount of money contributed by this team in the past, etc.

$$\tilde{y} = W_f[v, c] + b_f \tag{10}$$

We then can use the mean squared error between the predicted and ground truth persuasiveness as training loss. To take advantage of the labeled subset that has sentence level annotation of persuasive strategies, we reformulate this problem as a semi-supervised learning task:

$$l = \gamma \sum_{d \in C_L} (y_d - \tilde{y_d})^2 - \beta \sum -g_i \log p_i \tag{11}$$

$$+ (1-\gamma) \sum_{d' \in C_U} (y_{d'} - \tilde{y_{d'}})^2 \tag{12}$$

Here, C_L refers to the document corpus with sentence level persuasion labels. C_U denotes those without any sentence labels. g_i refers to the persuasion strategy in sentence S_i, and p_i is predicted by our model. γ and β are used as re-weight factors to trade off the penalization and reward introduced by different components.

6 Experiment

6.1 Dataset

Our collaboration with Kiva provided us access to all public data dumps of the team discussion forums on Kiva. Here we only focused on messages that have explicit links because in most cases, members need to include the loan link to better direct others to a specific loan or borrower. After removing messages that do not contain any links, we obtained 41,666 messages that contain loan advocacy. We used Amazon's Mechanical Turk (MTurk) to construct a reliable, hand-coded dataset to obtain the persuasion strategy label for each sentence. To increase annotation quality, we required Turkers to have a United States location with 98% approval rate for their previous work on MTurk. Since messages often contain different numbers of sentences, which might be associated with different sets of persuasion strategies, we sampled 200 messages for each fixed message length ranging from one sentence to six sentences, in order to guarantee that our hand-coded dataset reasonably represents the data. Messages with at most six sentences accounted for 89% percentages among all messages in our corpus. Each sentence in a message was labeled by two Mechanical Turk Master Workers [2]. To assess the reliability of the judges' ratings, we computed the intra-class correlation (ICC), and obtained an overall ICC score of 0.524, indicating moderate agreement among annotators (Cicchetti, 1994). The distribution for each persuasion strategy in the annotated corpus is described in the blue line in Figure 3. We assigned a persuasion label to a sentence if two annotators gave consistent labels for it, and filtered out sentences that annotators disagreed on the label.

In the final annotated corpus, there were 1200 messages, with 2898 sentences. The average number of sentences is 2.4 and the average number of words per sentence is 17.3. For predicting the persuasive strategy in each sentence, we randomly

---

[2] `https://www.mturk.com/worker/help:What-is-a-Mechanical-Turk-Master-Worker`

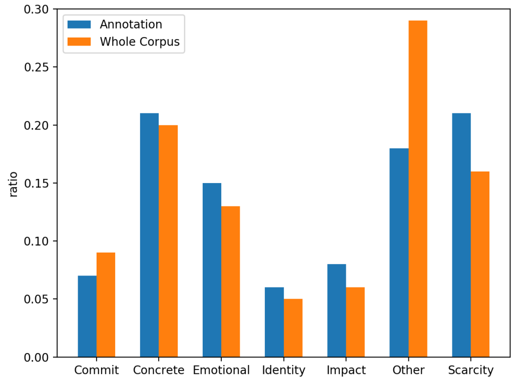


Figure 3: The distribution of each persuasion strategy in the annotation corpus and in the whole unlabeled corpus after prediction.

split 80% of this annotated corpus as the training set (2271 sentences in 1060 messages), 10% as the validation set (322 sentences in 70 messages), and 10% as the testing set (305 sentences in 70 messages). To further utilize supervision from the persuasiveness score of each message, we merged 1060 documents with sentence labels and 40,466 unlabeled messages, using it as the final training set for training semi-supervised models.

6.2 Model Setup and Baselines

We split documents into sentences and tokenize each sentence using Stanford's CoreNLP (Manning et al., 2014). Words appearing less than 5 times were replaced with a special UNK token. We trained the hyperparameters of the models on the validation set using Adam (Kingma and Ba, 2014). Specifically, we set the word embedding dimension to be 128, where the word embeddings are initialized randomly, and GRU dimension to be 256. The learning rate is set to be 5e-5. The balancer γ is the ratio of labeled data in a batch of training data. The balancer β is selected via grid search, searching in a set of (5, 10, 20, 50, 100), resulting in β=10.

We propose several baselines to predict the sentence level persuasion strategies for comparison with our model. (1) ***SVM + BoW*** is a *SVM* classifier with *RBF* kernel using bag-of-words features (one-hot). (2) ***GRU*** uses the hidden state at the last word as features to classify persuasive strategies, a special case of our *SH Net* model without the supervision from the overall persuasiveness scores. (3) ***bi-GRU*** uses bi-directional GRU.

H Net is a hierarchical GRU for classifying strategies with the supervision from the overall persuasiveness scores as shown in Figure 2, but it only adopts all the annotated messages. We denote our semi-supervised hierarchical model as **SH Net** (Semi Hierarchical Net), which utilizes both annotated messages and unlabeled corpus. **Semi-Att Net** builds on **SH Net** by incorporating both word-level and sentence-level attention. In addition to the textual cues in the advocation message, persuasive requests also depend on the context. We introduced a set of contextual descriptors into our semi-supervised hierarchical network, denoted as *SH-Att Plus Net*. Such features include the number of borrowers in this message, the number of team members in a team, the total amount of money contributed by this team, the number of messages ever posted in the discussion board of this team, and the amount of money requested in this loan.

6.3 Results

We evaluated the baselines and our hierarchical neural network models using accuracy, macro-averaged F1 score, macro-averaged precision and macro-averaged recall, as well as RMSE for evaluating the message level persuasiveness score prediction. As we can see in Table 1, when predicting the persuasive strategies (6 types of persuasive strategies plus an *Other* strategy), BoW + SVM gives a performance of 0.347 and a macro F1 of 0.229. A direct neural network *GRU* boosted the accuracy to 0.518, demonstrating the effectiveness of neural networks for sentence classification. When bi-directional contextual information is used, the sentence level prediction performance is 52.1%. Our hierarchical neural network achieved an accuracy of 48.2% and a macro F1 of 0.432. When incorporating the whole corpus of unlabeled messages, our semi-supervised neural network achieved an accuracy of 56.1% (16.4% improvement over *H Net*). This indicates that our semi-supervised model effectively takes advantage of the supervision from the small amount of labeled data and the overall persuasiveness scores. Moreover, we noticed that this semi-supervised neural network not only helps predict the sentence level persuasion strategies, but also assist the prediction of messages' overall persuasiveness with a 9% RMSE decrease. *Semi-Att* outperformed *SH Net* with an accuracy of 56.9%, and a macro F1 score of 0.518. Although the improvement from attention is minor (but significant), it's important for visualizing associations between words, persuasion strategies and persua-

	Evaluating Sentence Level Strategies				Doc Level
Model	**Accuracy**	**Macro F1**	**Macro Precision**	**Macro Recall**	**RMSE**
SVM (RBF) + BoW	0.347	0.229	0.364	0.167	-
GRU	0.518	0.479	0.479	0.479	-
bi-GRU	0.521	0.440	0.445	0.436	-
Hierarchical Net (H Net)	0.482	0.432	0.430	0.432	1.15
Semi Net (SH Net)*	0.561	0.513	0.504	0.522	1.05
Semi-Att Net*	**0.569**	**0.518**	**0.512**	**0.534**	1.04
Semi-Att Plus Net	0.552	0.513	0.515	0.512	**0.87**

Table 1: Results of different models. * indicates that the model is significantly better than the one above it.

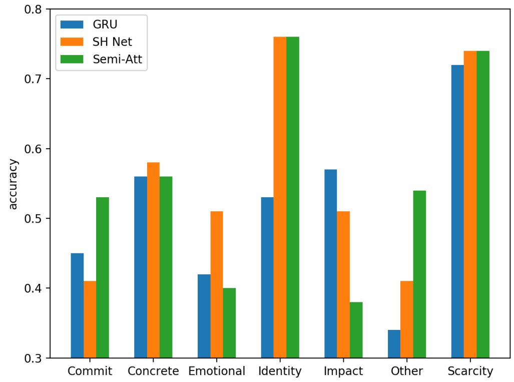


Figure 4: The accuracy for each persuasion strategy evaluated via *GRU*, *SH Net* and *Semi-Att*.

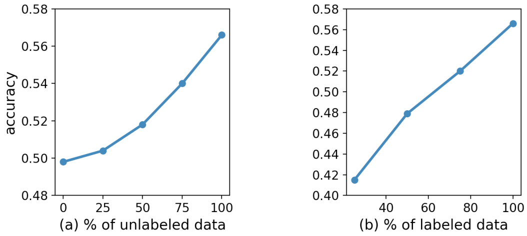


Figure 5: Model performances with different portion of unlabeled data (a) and labeled data (b).

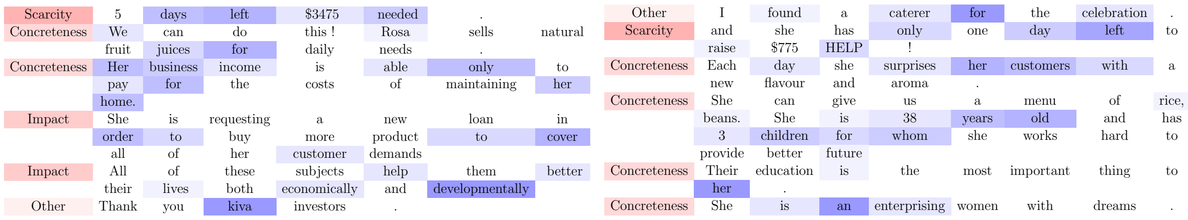
sion outcomes. Interestingly, incorporating contextual descriptors did not help the prediction of persuasion strategies. However, such contextual information strongly predicted the overall persuasiveness, decreasing RMSE to 0.84 from 1.04.

Strategy-Level Performance: We also report the accuracy per persuasion strategy category via *Semi-Att*, *SH Net* and simple *GRU* in Figure 4. It seems that overall neural models are better at capturing persuasion strategies such as concreteness, identity and scarcity. This might be because people are concrete by using specific terms such as numbers or entities that are easy to model. Similar principles might also occur for social identity and scarcity where the use of words such as "*we*", "*our*" and "*expire*", "*left*" can reveal a lot about the persuasion strategies.

Strategy	**Top Ranked Keywords**
Commitment	joined, lenders, loaning, lend, loan just, join, loaned, made, lent
Concreteness	women, married, old, heads, year-old money, sells, years, business, number
Emotion	hard, thank, better, grief, great maybe, help, please, thanks, happy
Identity	promotion, shall, captain, form, number spirits, lenders, member, team
Impact	improve, new, better, products, money to, use, business, more, order
Scarcity	minutes, there's, now, soon, go expire, hours, days, number, left

Table 2: Top ranked keywords for persuasion strategies

Different Percentage of Labeled Data: To figure out the importance of supervision from massages' overall persuasiveness scores, we experiment on *SH Net* with all the labeled messages. To this end, we include all the labeled messages, and vary the percentage of unlabeled corpus from 0%, 25%, 50%, 75%, to 100%, in Figure 5 (a). We found that as the amount of unlabeled messages increases, the accuracy of sentence level prediction increases as well, which further validates the effectiveness of the semi-supervised setting for persuasion strategy prediction. Similarly, to investigate the predictive power introduced by the sentence level labels, we also vary the percentage of labeled messages from 25%, 50%, 75%, to 100% when including the whole unlabeled corpus, as shown in Figure 5 (b). As expected, having more training data about sentence-level annotation increases the prediction performance. Overall, these experiments demonstrate the effectiveness of semi-supervised models for predicting sentence level persuasion strategies. This enables us



Scarcity	5	days	left	$3475	needed	.	
Concreteness	We	can	do	this !	Rosa	sells	natural
	fruit	juices	for	daily	needs	.	
Concreteness	Her	business	income	is	able	only	to
	pay	for	the	costs	of	maintaining	her
	home.						
Impact	She	is	requesting	a	new	loan	in
	order	to	buy	more	product	to	cover
	all	of	her	customer	demands		
Impact	All	of	these	subjects	help	them	better
	their	lives	both	economically	and	developmentally	
Other	Thank	you	kiva	investors	.		

(a) Predicted persuasiveness: 2.56 (after natural logarithm)

Other	I	found	a	caterer	for	the	celebration	.
Scarcity	and	she	has	only	one	day	left	to
	raise	$775	HELP	!				
Concreteness	Each	day	she	surprises	her	customers	with	a
	new	flavour	and	aroma	.			
Concreteness	She	can	give	us	a	menu	of	rice,
	beans.	She	is	38	years	old	and	has
	3	children	for	whom	she	works	hard	to
	provide	better	future					
Concreteness	Their	education	is	the	most	important	thing	to
	her	.						
Concreteness	She	is	an	enterprising	women	with	dreams	.

(b) Predicted persuasiveness: 1.75

Figure 6: Attention Visualization. Left-most columns in red represent sentence-level attention and the remaining columns in blue are word-level attention.

to obtain sentence level labels for any given paragraphs by using a small amount of labeled data.

6.4 Visualization

To validate whether our semi-supervised model captures characteristic words and sentences in requests, we visualize the attention in a sentence in Figure 6. We show the predicted persuasion label for each sentence in a message in red (left-most columns in Figure 6(a) and 6(b)), with the color scale indicating its learned attention weight. Word-level attention is highlighted in blue (remaining columns). As we can see in Figure 6(b), our model places emphasis on *Scarcity*, and highlights words such as "*left*" and "*day*" that carry the scarcity meaning. Similarly, in the second message— *5 days left* 3475 *needed*—our model first labeled the sentence as *Scarcity*, and then picked words such as "*days*" and "*left*". Sentences predicted as *Concreteness* seem to contain specific entities and concepts such as "*business*", "*her*", and "*home*". For *Impact*, our model accurately localizes words like "*in order to*" and "*cover*".

To demonstrate that our model can learn representative words associated with different persuasion strategies, we show the 10 highest-scoring words from sentences with different labels in Table 2. Interestingly, *Commitment* is highly associated with words such as "*made*" and "*loan*". Explicit mentions of "*thanks*" and "*hard*" were found in sentences with *Emotional* labels. Sentences that emphasize their "*team*" as a whole were labeled as *Identity*. Overall, this validates that our model is able to select informative words associated with different persuasion strategies.

For further illustration, we visualized the attention weight distributions of different persuasion strategies. Since the number of sentences inside each message is intertwined with attention weights, we only plotted the distributions for messages with two or three sentences in Figure 7. We observed that *Scarcity*, *Identity*, and *Impact* seem to play a relatively more important role for influencing the success of requests, whereas *Emotional* language, *Commitment* and *Concreteness* seem to concentrate more on the lower weight ranges.

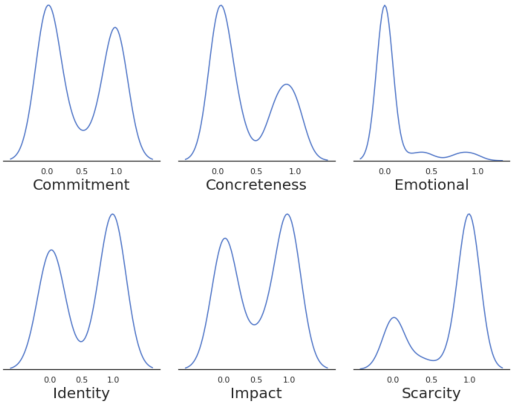


Figure 7: Attention weight distributions of persuasion strategies in requests with 2-3 sentences.

7 Importance of Persuasive Strategies

After applying the semi-supervised hierarchical neural network to the unlabeled 40552 messages, we obtained their sentence-level persuasive strategies usages. We showed the distribution of each persuasive strategy in the whole corpus in Figure 3, as described by the orange line. To further investigate how important each persuasive strategy is for convincing others to make loans, in this section, we present results on which of them are predictive via linear regression. All variables are standardized before entering the regression model. We controlled for the number of team members in a team, the total amount of money contributed by this team, the number of messages posted in the discussion board of this team. Since those variables are highly correlated with each other, we av-

Persuasion Strategy	**Kiva** (Coef.)	**RAOP** (Odds ratio)
Concreteness	0.041***	1.111***
Commitment	-0.015**	1.062
Emotional	0.030***	1.145***
Identity	0.087***	1.104**
Impact/Value	0.024***	1.084*
Scarcity	-0.076***	1.118***

Table 3: The influences of different persuasive strategies on request success on Kiva and RAOP. Here, p<0.001:***; p<0.01:**; p<0.05:*.

eraged them into a single variable to capture these team level attributes. We also controlled for the amount of money the borrower requested. We represented each message as a 6-dimensional vector to capture the amount of each persuasive strategy, which is calculated by selecting the maximum probability associated with each strategy from all sentences in this message. The persuasive strategy features significantly improve the model fit, as indicated by a 11.8% improvement in adjusted R-squared from 0.152 to 0.170. To demonstrate the generalizability of our persuasion strategies and the resulted semi-supervised model, we also applied our *Semi-Att* model to 5671 textual requests for pizza from the Reddit community "Random Acts of Pizza" (RAOP). Specifically, we used the data released by Althoff et al., (2014) where each request asked for a free pizza and the outcome whether its author received a pizza or not was provided in the dataset. Via *Semi-Att*, we were able to obtain the persuasive strategy used in each sentence of each request. Similarly, we built a logistic regression model to predict whether a request will receive the pizza or not, controlling for the community age of the requester, the number of subreddits the requester participated in, his/her number of posts as well as the votes (upvotes - downvotes) this requester had received.

As shown in the column of *RAOP* in Table 3, concreteness is significantly correlated with success on both datasets. This demonstrated that providing more evidence might help readers know the situation better, consistent with the effect of *Evidentiality* in Althoff et al., (2014). Similarly, making the request full of emotions (β=0.030, Odds ratio (OR) =1.145), mentioning the similarity between potential readers and the requester (β=0.087, *OR*=1.104), and talking about the potential impact and value for others (β=0.024, *OR*=1.084) are all significantly associated with increases in the persuasiveness of these requests across two contexts. In contrast, highlighting the urgency of the requests and emphasizing existing contribution to loans (β=-0.015) negatively correlate with request success (β=-0.076) on Kiva, confirming prior work (Yang and Kraut, 2017). This communicates to us that some of those loans might have expired before others read the request and took action given the limited time available, or it could be that members thought their actions might not help if the remaining money needed is high and the time left is low, different from the "limited-time offer" tactics widely used in commercial advertising. To sum up, the two analyses demonstrated that certain persuasive strategies such as *Identity* and *Impact* are consistently effective across two datasets, whereas *Scarcity* and *Commitment* contribute differently and need to be used with caution for different contexts.

8 Conclusion

In this work, we operationalized a set of persuasive strategies widely used in micro-lending platforms based on theories of persuasion, and developed an annotated corpus for identifying persuasion strategies. We designed a semi-supervised hierarchical neural network to identify the persuasive strategies contained in loan requests. Results show that our model improves accuracy considerably. We also showed how different persuasive strategies contribute to request success. In the future, we plan to build a richer taxonomy of persuasion strategies and incorporate additional neural architectures such as variational autoencoders to better represent sentences in each message to further assist the modeling of persuasiveness. Beyond the text, images and even audios may provide additional insights on the successes of persuasive requests. Our model also has important applications to other domains, such as in computational advertisements, micro-funding platforms and political campaigns.

Acknowledgement

The authors would like to thank Jason Eisner for his help at the brainstorm stage and insightful followup suggestions, and the anonymous reviewers for their helpful comments. Diyi Yang was supported by Facebook Fellowship.

References

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2014. How to ask for a favor: A case study on the success of altruistic requests. In *Proceedings of ICWSM*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Winston Carlile, Nishant Gurrapadi, Zixuan Ke, and Vincent Ng. 2018. Give me more feedback: Annotating argument persuasiveness and related attributes in student essays. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 621–631.

Shelly Chaiken. 1980. Heuristic versus systematic information processing and the use of source versus message cues in persuasion. *Journal of personality and social psychology*, 39(5):752.

Moitreya Chatterjee, Sunghyun Park, Han Suk Shim, Kenji Sagae, and Louis-Philippe Morency. 2014. Verbal behaviors and persuasiveness in online multimedia content. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 50–58.

Robert B Cialdini and Nathalie Garde. 1987. *Influence*, volume 3. A. Michel.

Domenic V Cicchetti. 1994. Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. *Psychological assessment*, 6(4):284.

Sanorita Dey, Brittany Duff, Karrie Karahalios, and Wai-Tat Fu. 2017. The art and science of persuasion: Not all crowdfunding campaign videos are the same. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 755–769. ACM.

Vincent Etter, Matthias Grossglauser, and Patrick Thiran. 2013. Launch hard or go home!: Predicting the success of kickstarter campaigns. In *Proceedings of the First ACM Conference on Online Social Networks*, COSN '13, pages 177–182, New York, NY, USA. ACM.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 549–554.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48.

Michael D Greenberg, Bryan Pardo, Karthic Hariharan, and Elizabeth Gerber. 2013. Crowdfunding support tools: predicting success & failure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1815–1820. ACM.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *ACL*, pages 1589–1599, Berlin, Germany. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21.

Carl I Hovland, Irving L Janis, and Harold H Kelley. 1953. Communication and persuasion; psychological studies of opinion change.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tanushree Mitra and Eric Gilbert. 2014. The language that gets people to give: Phrases that predict success on kickstarter. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 49–61. ACM.

Elena Musi, Debanjan Ghosh, and Smaranda Muresan. 2018. Changemyview through concessions: Do concessions increase persuasion? *arXiv preprint arXiv:1806.03223*.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694.

Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Isaac Persing and Vincent Ng. 2017. Why can't you convince me? modeling weaknesses in unpersuasive arguments. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4082–4088. AAAI Press.

Richard E Petty, John T Cacioppo, and David Schumann. 1983. Central and peripheral routes to advertising effectiveness: The moderating role of involvement. *Journal of consumer research*, 10(2):135–146.

Pedro O Pinheiro and Ronan Collobert. 2015. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1713–1721.

Reid Pryzant, Young-Joo Chung, and Dan Jurafsky. 2017. Predicting sales from the language of product descriptions. In *Proceedings of the SIGIR 2017 Workshop on eCommerce (ECOM 17)*.

Reid Pryzant, Kelly Shen, Dan Jurafsky, and Stefan Wagner. 2018a. Deconfounded lexicon induction for interpretable social science. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1615–1625.

Reid Pryzant, Kazoo Sone, and Sugato Basu. 2018b. Interpretable neural architectures for attributing an ad's performance to its writing style. In *EMNLP Workshop on BlackboxNLP*.

Sara Rosenthal and Kathleen Mckeown. 2017. Detecting influencers in multiple online genres. *ACM Trans. Internet Technol.*, 17(2):12:1–12:22.

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66.

LJ Shrum, Min Liu, Mark Nespoli, and Tina M Lowrey. 2012. Persuasion in the marketplace. *The SAGE Handbook of Persuasion*, page 314.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Christian Stab, Tristan Miller, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources using attention-based neural networks. *arXiv preprint arXiv:1802.05758*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 613–624, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200, Berlin, Germany. Association for Computational Linguistics.

Anbang Xu, Xiao Yang, Huaming Rao, Wai-Tat Fu, Shih-Wen Huang, and Brian P. Bailey. 2014. Show me the money!: An analysis of project updates during crowdfunding campaigns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 591–600, New York, NY, USA. ACM.

Diyi Yang and Robert E Kraut. 2017. Persuading teammates to give: Systematic versus heuristic cues for soliciting loans. *Proceedings of the ACM on Human-Computer Interaction*, 1:114.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Fan Zhang and Diane Litman. 2015. Annotation and classification of argumentative writing revisions. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 133–143.

Fan Zhang, Diane Litman, and Katherine Forbes-Riley. 2016. Inferring discourse relations from pdtb-style discourse labels for argumentative revision classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2615–2624.

Recursive Routing Networks: Learning to Compose Modules for Language Understanding

Ignacio Cases[1], **Clemens Rosenbaum**[2], **Matthew Riemer**[3], **Atticus Geiger**[1],
Tim Klinger[3], **Alex Tamkin**[1], **Olivia Li**[1], **Sandhini Agarwal**[1],
Joshua D. Greene[4], **Dan Jurafsky**[1], **Christopher Potts**[1], **Lauri Karttunen**[1]
[1]Stanford University [2]University of Massachusetts Amherst
[3]IBM Research [4]Harvard University
`cases@stanford.edu, cgbr@cs.umass.edu`

Abstract

We introduce Recursive Routing Networks (RRNs), which are modular, adaptable models that learn effectively in diverse environments. RRNs consist of a set of functions, typically organized into a grid, and a meta-learner decision-making component called the *router*. The model jointly optimizes the parameters of the functions and the meta-learner's policy for routing inputs through those functions. RRNs can be incorporated into existing architectures in a number of ways; we explore adding them to word representation layers, recurrent network hidden layers, and classifier layers. Our evaluation task is natural language inference (NLI). Using the MULTINLI corpus, we show that an RRN's routing decisions reflect the high-level genre structure of that corpus. To show that RRNs can learn to specialize to more fine-grained semantic distinctions, we introduce a new corpus of NLI examples involving implicative predicates, and show that the model components become fine-tuned to the inferential signatures that are characteristic of these predicates.

1 Introduction

Human cognition has an extraordinary ability to modularize, decomposing problems and solving them by re-composing elements from prior solutions, and this ability is nowhere more evident than in language understanding (Partee, 1984; Janssen, 1997). Most machine learning architectures lack this modularity, which limits their ability to generalize and leaves them susceptible to *catastrophic interference* (McCloskey and Cohen, 1989) – forgetting past skills when acquiring new ones.

We propose to address this need for modularity by applying Routing Networks (Rosenbaum et al., 2017) to natural language understanding. Routing Networks are self-organizing networks with two components (Figure 1): a set of function blocks which can be applied to transform the input, and a router which makes decisions about which function block to apply next. Here we introduce Recursive Routing Networks (RRNs), in which there is a single set of composable functions, recursively chosen by the router. RRNs can be applied to different components of modern language understanding architectures with full end-to-end training. The model jointly optimizes the parameters of selected sub-functions and the meta-learner's policy for how to route inputs through those functions. As a result, individual sub-functions specialize to specific inputs, and paths through the grid of sub-functions can similarly be trained to reflect specific concepts and capabilities.

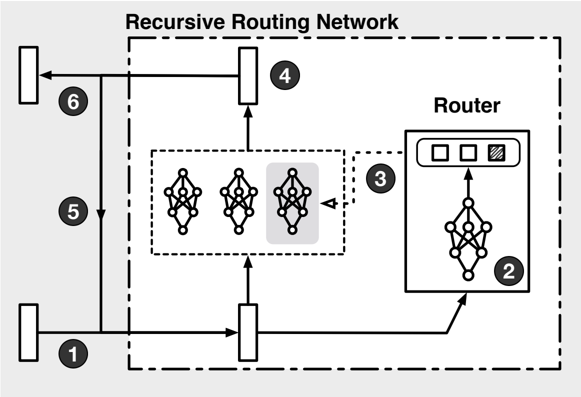


Figure 1: Given a premise–hypothesis pair x, e.g., *t managed to do s* and *t did s* (1), the model needs to learn to predict *entails* (6). The router (2) estimates the value of applying each of the available sub-functions to the input. Given that *manage* has certain semantic properties, the router may select (3) a function f_1 specialized to them. The module is then applied (4), yielding $f_1(x)$. This process repeats (5), now using $f_1(x)$, selecting and applying another sub-function, and so on, until the router is confident in its prediction (6).

RRNs share many intuitions with other modular methods like: Neural Module Networks (Andreas et al., 2015, 2016; Hu et al., 2017), which learn to construct a neural network from pre-defined modules; the Compositional Recursive Learner (Chang et al., 2018), a closely related approach that uti-

Proceedings of NAACL-HLT 2019, pages 3631–3648
Minneapolis, Minnesota, June 2 - June 7, 2019.

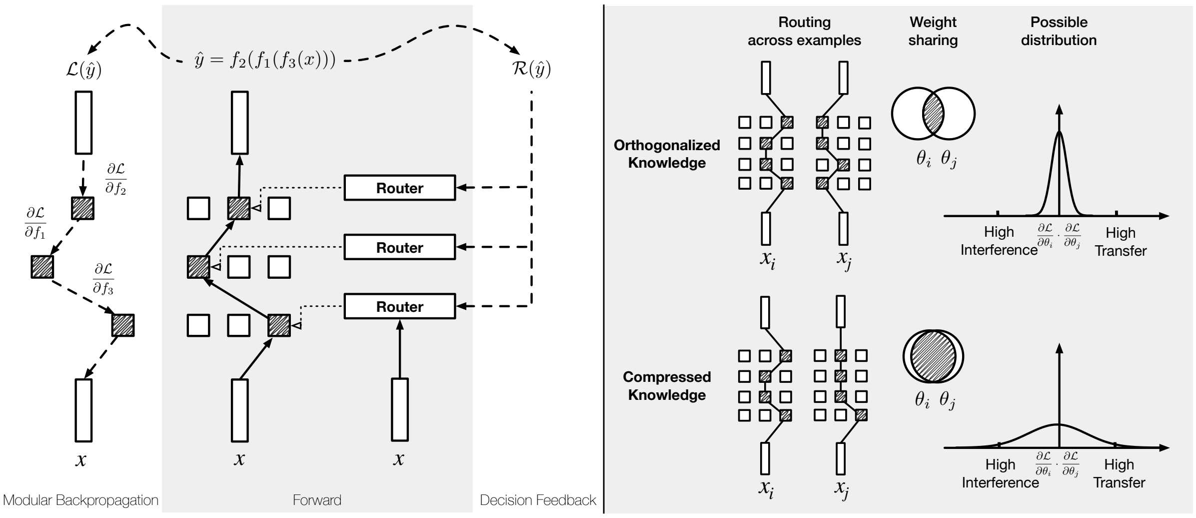


Figure 2: Left: **RRN learning.** Forward (grey background) and backward passes (white background) during the training of a rollout of an RRN with three blocks (width) and limited to depth three (height). In this unrolled version, a *path* is the sequence of function blocks selected by the router. Right: **The dilemma of weight sharing and the transfer–interference trade-off.** When examples are learned using largely separate weights, there is a thin possible distribution of gradient dot products that limits interference, which unfortunately also limits transfer. This is beneficial for unrelated examples, but frustrates the learning of related ones. Conversely, when examples are learned using largely shared weights, there is a wide possible distribution of gradient dot products that allows for both high magnitude transfer and interference. This is beneficial for related examples, but maximizes interference when examples are unrelated. With RRNs, we provide our network with an unprecedented degree of leverage to learn to navigate this trade-off.

lizes curriculum learning to solve arithmetic and vision problems; and ModularNetworks (Kirsch et al., 2018), which extend Routing Networks with an EM-like training approach.

We show how to incorporate RRNs into different neural components (word representation layers, recurrent network hidden layers, classifier layers), and we study their application to natural language inference (NLI), in which premise–hypothesis pairs are labeled for whether the premise entails, contradicts, or is neutral with respect to the hypothesis. We chose this task because reasoning in natural language involves context-sensitive interpretation of words and sentences as well as compositional structure. We make use of the MULTINLI corpus (Williams et al., 2018), which includes text from multiple genres that we expect to condition linguistic senses in complex ways. Our experiments show that RRNs learn policies and components that reflect this genre structure, which leads to superior performance.

We also introduce a new corpus of NLI examples involving implicative constructions like *manage to*, *be able to*, and *fail to* (Karttunen, 1971, 2012). This corpus follows the design of many recent NLI corpora, but with the added challenges of reasoning about implicatives, which have logical *signatures* that interact compositionally with each other and with surrounding semantic operators. Our experiments show that trained RRN model components become fine-tuned to these signatures. Finally, we introduce an extension of the framework which leverages a *Dispatcher* for situations where meta-information is not available at test time, obtaining very promising results.

2 Background and Motivation

RRNs are a natural extension of the Routing Networks introduced by Rosenbaum et al. (2017), themselves part of a larger class of conditional computation models (Bengio et al., 2015). Routing Networks combine trainable modules with a meta-learner called a *router*, which is typically trained using reinforcement learning, though any hard-decision making algorithm could be used. Given an example, the router selects a module and applies it, yielding an activation that can be the input to another routing iteration. As Rosenbaum et al. observe, this can be modelled as a recursive process where the router is applied repeatedly to define complex paths through the modules. The final routed output is passed to additional layers or interpreted as the output of the network (Figure 1).

We are motivated to consider Routing Networks, and in particular recursive variants, in order to address the transfer–interference trade-off

(Riemer et al., 2019). Vanilla neural networks latently attempt to solve a very difficult problem of deciding when to orthogonalize and compress knowledge (Figure 2, right). When two examples are learned with the same weights, there is high potential for transfer as well as interference. This is good for related examples because it maximizes the potential for transfer. However, weight sharing is bad for unrelated examples, as it increases the likelihood of interference. In contrast, when two examples are learned using different weights, there is low potential for transfer and interference. This is beneficial for unrelated examples, but limits the potential for learning about commonalities between related ones. RRNs extend vanilla neural networks by granting them the leverage to navigate this trade-off by making global functional decisions at the module level. As a result, RRNs explicitly make decisions to compress or orthogonalize knowledge between examples by deciding whether to share specific weights.

Thus far, hard-selection routing has not been applied to language domains. Zaremoodi et al. (2018) introduced a *soft* version of routing that falls within a larger class of Mixtures of Experts (MoE) models (Jacobs et al., 1991). However, MoE models differ from RRNs in two crucial ways. First, MoE models generally do not consider the recursive application of functions. The promise is that we can compose functions to reflect the compositional aspects of a problem. Imagine we have a sentence encoding and we want to answer a particular question. We can now condition the router on the question, so that it applies exactly the functions required that translate the encoding to extract the answer. Second, MoE models do not allow for nearly the level of specialization as those based on routing, because they do not eliminate weight sharing across modules and instead only gate the sharing. In practice, this still leads to significant interference across tasks.

Composition of modules has been widely explored for question answering, starting with Andreas et al. (2015). Andreas et al. (2016) learn to assemble a deep neural network on-the-fly from a pre-specified inventory of neural modules using tree-structured layouts based on linguistic analysis. Chang et al. (2018) also explore routing, but with problems described in pseudo-language rather than natural language.

3 Recursive Routing Networks

We now provide a formal presentation of RRNs and show how to incorporate them into existing neural architectures.

3.1 The Routing Paradigm

Formally, the router bases its decision on the tuple $\langle \mathfrak{f}(x), m \rangle$, where x is the sample, $\mathfrak{f}(x) = f_i \circ \cdots \circ f_k(x)$ is the composition of all applied functions from the set of all available functions $\mathcal{F} = \{f_1, \ldots, f_b\}$, and m is a vector that contains meta-information that can be utilized by the router – for example, an embedding for its genre or semantic classification. The router consists of a policy $\pi(\mathcal{F} | \langle \mathfrak{f}(x), m \rangle)$ that determines which of the functions in $\mathcal{F}$ to apply for a given state $\langle \mathfrak{f}(x), m \rangle$. Once a new function f_k is selected, the latest activation (and, thereby, the state) gets updated to $f_k \circ \mathfrak{f}(x)$, and the process repeats.[1]

We focus on the recursive case where the set $\mathcal{F}$ is generally the same for each decision. Apart from being the more general formulation, this form of recursivity allows more weight sharing and better compositional generalization. While there is no necessary upper limit to the number of selections, we found that limiting the number to a maximum of d makes the learning more stable. Consequently, the total number of possible routing paths available is b^d.

Decision Making One of the most important design choices for the router concerns the meta-learning algorithm. Routing is limited to *hard* decision-making algorithms, in particular a stochastic reparameterization using the Gumbel-Softmax function (Jang et al., 2016; Maddison et al., 2016) and reinforcement learning (RL) algorithms. As with other recent approaches to compositional architectures with hard selections (Bengio et al., 2015; Shazeer et al., 2017; Kirsch et al., 2018), routing networks can suffer from *module collapse*, a lack of diversity in the router's decision making. This general problem, common in architectures that jointly train decision making and functions, stems from an early over-estimation of the value of specific functions. This leads to these functions being selected and trained more than others, until these functions are indeed so good

---

[1] Some architectures have constraints that prevent repeated selection of the same functions in $\mathcal{F}$. In these cases, the router has to be restricted to select from only a compatible subset of $\mathcal{F}$.

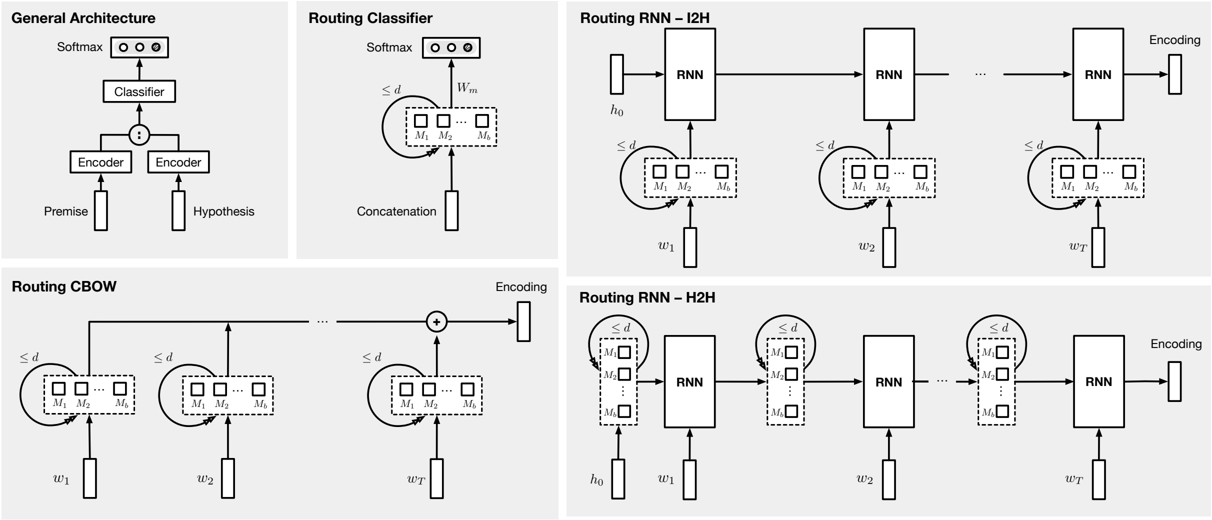


Figure 3: Top left: Our general NLI architecture. Top center: Routing the classifier. Lower left: Word representation routing of a CBOW encoder. Top right: Routing an RNN input transformation. Bottom right: Routing an RNN hidden transformation.

that others are not considered. To mitigate this, we start with a strategy proposed by Rosenbaum et al. (2017): exclusively conditioning the router on meta-information provided by the datasets and storing the router's policy in a table. We found that this stabilizes early training. During later training, we can replace the meta-information hard assignment rule with a meta-information "guesser". We implement this guesser with what Rosenbaum et al. call a *Dispatcher*, an approximation-based sub-policy that estimates the meta-information for each sample and passes it to the corresponding policies (Section 5).

Losses and Training Once the routing procedure terminates, the selected modules are trained using standard optimization techniques, such as backpropagation with stochastic gradient descent (SGD). The decision-making algorithm is either also trained with SGD (Gumbel reparameterization) or trained using reinforcement learning. As we focus on classification problems, our core loss function $\mathcal{L}_{class}(y, \hat{y})$ is the standard cross-entropy loss. We backpropagate this loss along the function path chosen by the router for input x.

Reinforcement Learning Rewards For RL training of the router, we define the reward r for the reinforcement learner to be the sum of the negative classification loss and an additional regularization reward that encourages diversity in the selection of modules:

$$r = -\mathcal{L}_{class}(y, \hat{y}) + r_{reg} \tag{1}$$

For discussion of this reward, see Appendix A.1.

3.2 Architectures

If not otherwise mentioned, we assume that the routed modules are all fully connected layers with the same input and output dimensions. We furthermore assume that the router is always selecting from the same set of functions when defining paths through the routing layers, as this straightforwardly allows recursion in the relevant sense.

Routing Classifiers The simplest method we explore involves routing the layers of a classifier. To do this, we define a fixed set of b fully connected layers (FC), each of the same dimensionality, and we allow the router to choose any path through this set up to a maximum length d (Figure 3, top center). The final activation produced by the chosen path is then densely connected to a non-routed output layer. To further model each example's dependence on its class, we hard-select this output layer based on the meta-information label m for x:

$$W_m \left(\mathrm{FC}_{k_1} \circ \cdots \circ \mathrm{FC}_{k_d}(x)\right) + b_m \tag{2}$$

Routing RNN Encoders In routing RNNs we focus on their two core transformations: from the hidden state at time step t_k to time step t_{k+1}, and from the input to the hidden state. We have designed routing architectures for both of them. In this paper, we start with LSTMs, but the techniques are straightforward to adapt to other cell types.

Figure 3, top right, shows an architecture where we route the input-to-hidden (I2H) transformation. Similarly, Figure 3, bottom right, shows an architecture where we route the hidden-to-hidden

(H2H) transformation. While these transformations are often designed as single fully connected layers, we allow a recursive application of d steps from a selection of b modules to be applied. The selections for the transformations f, i, o, c are tied to be the same. The corresponding transformation (in form of their weight matrices $W_{f/i/o/c}$ for I2H and $U_{f/i/o/c}$ for H2H) becomes:

$$\begin{aligned} &K = \mathrm{FC}_k \circ \cdots \circ \mathrm{FC}_m(x_t), \\ &\forall K \in \{W_f, W_i, W_o, W_c, U_f, U_i, U_o, U_c\} \end{aligned} \tag{3}$$

Routing CBOW Encoders We also experiment with routing a continuous bag of words (CBOW) encoding. As routing after the main addition is just routing the classifier, we instead add a word-level transformation before the addition (Figure 3, bottom left). This transformation can again be routed recursively (with up to d steps through b modules). The entire CBOW model can be defined as (with $w_1, \ldots, w_t$ as premise or hypothesis):

$$\begin{aligned} \mathrm{CBOW}_R(w_1, \ldots, w_t) = \\ \textstyle\sum_{i=1}^{t} \mathrm{FC}_{k_1} \circ \cdots \circ \mathrm{FC}_{k_d}(w_i) \end{aligned} \tag{4}$$

Routing Transformers The Transformers model (Vaswani et al., 2017) was recently pretrained as a language model (Radford et al., 2018) achieving impressive results on several NLI datasets. Since the corpus for pretraining is not available, we instead use the parameter-files distributed by the authors.[2] The encoding consists of twelve Transformer-blocks. Each block consists of a convolutional attention layer followed by two convolutional layers (along with several dropout and layer-norm layers). This allows routing at different levels of granularity, of which we investigate two: routing entire blocks and routing the attention step within each block. As we use pre-defined parameters, we cannot apply the blocks or the attention layers recursively. Furthermore, we have to add routing in the fine-tuning phase of using Transformers by creating b copies of each routed module (the depth is necessarily $d = 12$). When fine-tuning, the router then diversifies the initially identical modules.

4 Experiments

We now evaluate the routing architectures introduced in the previous section, along with minimally different non-routing baselines. Our experiments focus on NLI, using the MULTINLI corpus (Williams et al., 2018) and a new English Corpus of Implicatives, the *Stanford Corpus of Implicatives* (SCI). In both, each example is a premise/hypothesis pair labeled with one of *entails*, *contradicts*, or *permits*. What is special about MULTINLI and SCI in the current context is that the examples also have meta-information that can be used to guide the router: a genre label for MULTINLI and an implicative signature for SCI. We expect our routing models to leverage this information during policy and parameter learning.

For all models (except Transformers), we adopt the architecture in Figure 3, top left, in which the premise and hypothesis are processed separately, and the final representations of each are concatenated and fed into the classifier layers. We explore two methods for input processing: (i) pretrained word representations (GloVe; Pennington et al. 2014) and (ii) pretrained contextualized representations (ELMo; Peters et al. 2018). For all non-routed models, we provide explicit access to the genre label via special keywords put in front of each sentence before encoding. The routed models use this label as meta-information.

Unless otherwise specified, we used embedding and hidden dimensions of 300. The classifier consists of three fully connected layers with input and output dimensions of 600 (also when routed). The final layer projects from 600 to the output dimension, 3. The classifier nonlinearities are ReLUs. We train the modules using Adam (Kingma and Ba, 2014) with a learning rate of $1e{-}3$ and the router using SGD with a learning rate of $3e{-}4$ (for additional details, see Appendix A.5). For Transformers, all hyperparameters are determined by the published parameter files.

In experiments with different decision making algorithms (see Appendix A.5), we found that the Gumbel-Softmax reparameterization performed 10% worse on average, with *much* higher variance. QLearning was consistently as good as other more complex RL algorithms, so we report only our QLearning experiments. As Routing Networks have more parameters than their non-routed counterparts, we ran experiments with larger non-routed networks. We found that this did not affect performance, only resulting in more overfitting.

---

[2] https://github.com/openai/finetune-transformer-lm

			MULTINLI	SCI			
Encoding	Embedding	Routing	match	joint	disjoint	mismatch	nested*
CBOW	Glove	None	61.94±0.13	57.26±0.18	55.68±0.47	53.41±0.86	50.98±0.92
	Elmo	None	59.12±0.59	58.28±0.67	57.99±0.90	56.30±1.28	
	Glove	Classifier	51.99±0.17	62.33±0.80	61.17±0.28	51.55±1.61	
	Elmo	Classifier	55.36±0.35	64.29±0.99	61.60±0.61	55.50±0.87	
	Glove	WP	64.38±0.24	*74.95*±0.64	*73.91*±0.54	68.69±0.93	
	Glove	WP+D	**65.84**±0.12	**75.56**±0.77	**74.87**±0.49	**71.08**±0.52	**75.43**±0.29
	Elmo	WP	61.28±0.25	65.91±4.19	61.67±1.30	57.34±0.52	
RNN	Glove	None	**66.53**±0.21	67.04±2.36	63.52±2.00	*59.32*±2.87	57.69±2.75
	Elmo	None	66.13±0.19	65.76±2.16	61.89±2.41	*59.02*±1.92	
	Glove	Classifier	65.60±0.33	**71.02**±0.85	**67.82**±0.81	**60.58**±1.35	
	Elmo	Classifier	59.49±0.55	*68.98*±2.99	*67.23*±2.40	52.96±1.82	
	Glove	I2H	47.79±0.47	53.25±0.28	56.57±1.83	53.99±0.55	
	Elmo	I2H	49.12±1.01	56.13±1.04	56.23±0.17	53.36±0.26	
	Glove	H2H	62.34±0.25	54.89±0.93	53.08±0.57		
	Elmo	H2H	64.04±0.19	59.34±4.41	56.57±1.83	57.61±0.55	
Transformers (pretrained)		None	76.12	88.12	88.07	85.64	
		Classifier	76.50	86.05	86.68	73.45	
		Attention	**76.63**	87.91	**88.45**	**85.95**	
		Block	75.87	**88.22**	87.88	85.33	

Table 1: Results for MULTINLI and SCI with different baselines and their routed versions. We report average accuracy with confidence intervals over five runs with different seeds. For Transformers, we found that finetuning was highly volatile. We therefore report test results from the best-of-5 train models. *All results for nested SCI were computed by fine-tuning the same network previously trained on joint. 'WP' stands for Word Projection, '+D' for Dispatching, 'I2H' for Input-to-Hidden routing, and 'H2H' for Hidden-to-Hidden routing. Italics mark scores whose confidence intervals overlap with the best scores.

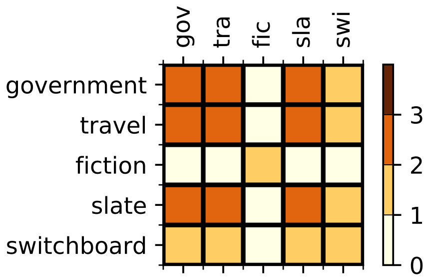


Figure 4: Path (module selection) overlap for MULTINLI between genres with the CBOW GloVe WP model. The diagonal represents the number of function blocks applied for a genre. A maximum of three means that two genres would be routed through the exact same functions.

4.1 Experiments with MULTINLI

The MULTINLI corpus contains 392,702 training examples, 10K dev examples, and 10K test examples. The examples come from 5 genres: fiction, government reports, the *Slate* website, the Switchboard corpus (Godfrey and Holliman, 1997), and Berlitz travel guides. We treat these genre labels as meta-information for the model (Section 3.1).

Our MULTINLI results are given in Table 1, and the learning dynamics in Figure 5. The best model combines the Transformer base model with routing in the attentional layers. Our methods for routing the RNN seem to be less successful, but word-representation routing offers clear benefits with the CBOW base model. Interestingly, the baseline (non-routed) models perform at the same level as the very similar models without genre labels evaluated by Williams et al. (2018). It seems that these models are not able to take advantage of the meta-information. In contrast, RRNs seem to provide the space needed to condition linguistic senses on these labels.

Our hypothesis is that routing will not only lead to better performance on diverse tasks like MULTINLI, but also that the paths – i.e., the sequence of functions selected by the router – followed by the network will reflect high-level task structure. Figure 4 suggests that this is the case for MULTINLI. Here we show the degree of path-overlap for all pairs of genres. As we might expect, government (the 9/11 report), Slate (current affairs), and travel cluster together, as distinct from the two more isolated genres (Switchboard; spoken language) and fiction (mostly from the 20th century).

4.2 Experiments with Implicatives

Karttunen (1971) discovered that implicative constructions, such as *manage* and *waste chance* (e.g. *They wasted their chance to win*), have *signatures*, which characterize the inferences they support in positive and negative contexts. This makes them

the order compelled him to appear as a witness	**entails**	he appeared as a witness
we have missed an opportunity to examine the art market today	**contradicts**	we have examined the art market today
Mr Odinga had not been forced to change his plans	**permits**	Mr Odinga had changed his plans

Table 2: Examples from SCI randomly chosen from the validation set. Each row contains a triplet formed by a premise (left column), a hypothesis (right column), and a label specifying one of the three possible relations (*entails*, *contradicts*, *permits*) holding between premise and hypothesis. The last row contains an example of a probabilistic implicative (see the main text).

particularly informative for NLI predictions.

For instance, the positive sentence *Joan managed to solve the problem* entails *Joan solved the problem*, and the negative sentence *Joan didn't manage to solve the problem* contradicts *Joan solved the problem*, so we say that the verb *manage* has the signature + | − (MacCartney and Manning, 2009; Karttunen, 2016). In contrast, *waste chance* has the opposite signature, since *they wasted the chance to befriend him* contradicts *they befriended him*, and *they didn't waste the chance to befriend him* entails *they befriended him*. There are seven implicative signatures: six were previously known (Karttunen, 1971, 2012), and we found an additional one (+ | +; e.g. *take no time to*). See Appendix C for additional details.

Signatures are compositional: when two or more implicative constructions are composed in a sentence, they create a *nested implicative construction* whose signature is determined by the signatures of the individual verbs (Nairn et al., 2006). For example, *John managed to remember to get the keys* entails *John got the keys*, where the nested implicative *manage to remember* has the overall signature + | −. We also see a more limited form of compositionality inside phrasal implicatures; their signatures are often largely determined by the lexical semantic family of their constituent words. Therefore, signatures make implicatives ideal for evaluating different degrees of compositional generalization with RRNs, as they provide valuable meta-information (Section 3.1).

Statistic	SCI
Validated pairs	41.15%
Unanimous gold label	94.93%
Gold matches author	98.61%
Gold does not match author	1.39%
No gold label	0
Fleiss κ coefficient	0.95
entails	0.32
contradicts	0.30
permits	0.38

Table 3: SCI statistics. Top: Percentage of validated pairs and basic agreement. Bottom: Fleiss κ coefficients and proportion of all assignments made to the corresponding label.

4.2.1 A Corpus of Implicative Constructions

Our SCI dataset contains ≈10K premise–hypothesis pairs. All seven signature types are represented, in addition to pragmatic signatures, which have inferential biases (see Appendix C).

We provide SCI[3] in three versions for all single and phrasal implicatives.[4] In the *joint* version, the underlying distribution of implicatives is shared across train, validation, and test splits. In the *disjoint* version, a different subset of implicatives is used in train from those used in validation and test. This allows us to test generalization to unseen constructions. Although disjoint with respect to constructions, the constructions are carefully distributed so that all the underlying signatures and most lexical items are represented in all splits. The lexical items that make up implicative constructions overlap between the splits. For example, *take vow* appears only in training and validation, while *make vow* only appears in test. The last version is *mismatch*, where different subsets of the signatures are present in training/validation and test (see appendix C for more details).

Data collection for SCI proceeded as follows: we collected at least 12, and most often 20, seed premises from examples found in Google Books and on the web. For each example, six hypotheses were created by expert annotators.[5] These six examples were constructed in two steps: first, the premise was taken to be the seed sentence, and three hypotheses were created to exhaust the label space in relation to the premise. Second, the premise was taken to be the *negation* of the seed sentence, and a different set of three hypotheses was created to also exhaust the label space with respect to this negated sentence. For example, one of the seeds for *manage* was *I managed to see who it was*. From this, annotators produced by hand a

---

[3] https://nlp.stanford.edu/projects/sci/

[4] We provide nested constructions as a separate extension, with the exception of a few nested implicatives that were used in the development version of this corpus, for which the results are presented here. In the final version of the corpus, single and phrasal implicatives will be separated from nested implicatives. See the Appendix for more details.

[5] Native speakers of English trained in semantics at the undergraduate level, with expertise in implicatives.

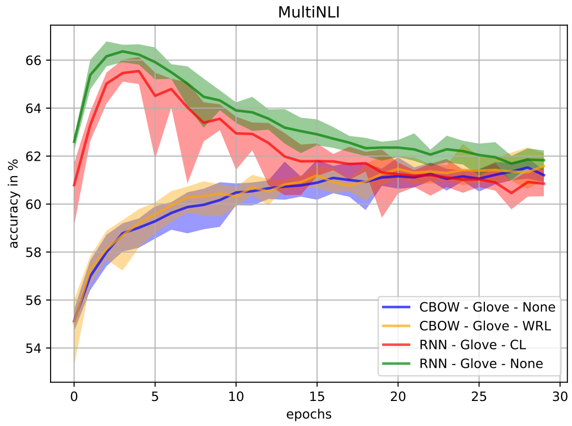


Figure 5: Learning on MULTINLI with GloVe inputs.

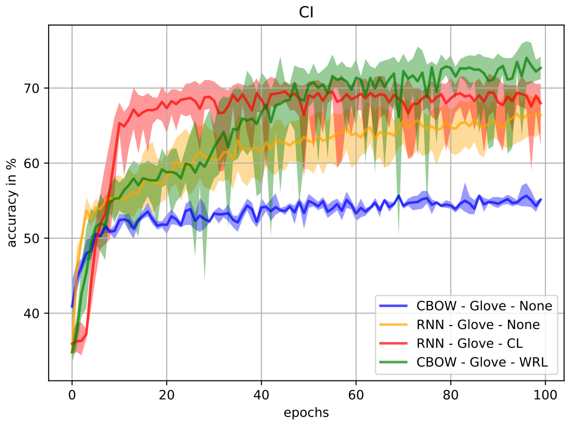


Figure 7: Learning on SCI (joint) with GloVe inputs.

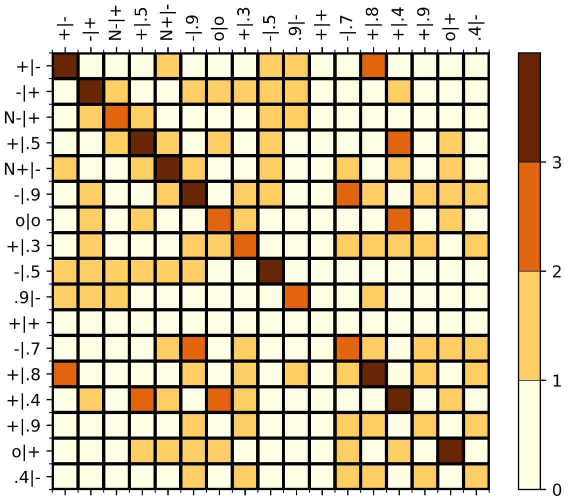


Figure 6: The path-overlap between different signatures on SCI, using the CBOW GloVe WP model, for $b = 4$, $d = 3$. N+|− and N−|+ are nested signatures.

negated seed, *I did not manage to see who it was.* Then three hypotheses were generated for each of these two premises. Finally, each example was labeled by the author of the hypothesis examples.

Validation was performed on a randomly selected subset of constructions. Two additional votes were cast by different linguists, resulting in three label votes for each example in the subset. The gold label was then defined to be the majority class from this set of votes. The Fleiss κ shows high inter-annotator agreement (Table 3).

4.2.2 Implicatives Results and Analysis

For each example in SCI, we use its associated implicative signature as the meta-information label. This serves as a subtler kind of semantic information than genre. The learning dynamics of different models are shown in Figure 7. As Table 1 shows, RRNs lead to considerable gains in performance over most benchmarks. Recursive routing of the classifier helps consistently with all models. Even simple word-level routing yields major improvements for CBOW. As with MULTINLI, we believe that the modules allow the conditioning of linguistic senses. Routing at the word-level for sequential models seems suboptimal here. The training accuracies ($> 99.5\%$) suggest that the problem is overfitting. Intuitively, routing can assign too many exclusive parameters to each class; and as the samples within these classes are similar to begin with, remembering them can be easier for the network than for non-routed networks.

We expect examples with similar signatures to be routed along similar paths. Figure 6 summarizes the path-overlap between different signatures. Some similar paths involve reversals of polarity, which calls for further scrutiny. However, many of these similarities make intuitive sense. For example, `+|.4` and `+|.5` are highly similar, as are `-|.7` and `-|.9`. In addition, the isolation of the unusual signature `+|+` (limited to just two constructions) seems expected, as does the affinity of the nested signatures `N+|-` and `N-|+` to their unnested counterparts. However, as shown in Table A3, some signatures only contain very few samples, resulting in highly noisy routing paths.

4.3 Navigating the Transfer–Interference Trade-off

We now seek to provide intuitive explanation for how RRNs help in navigating the transfer–interference trade-off. We have characterized the trade-off in terms of controlling weight sharing between examples. While baseline networks can learn to navigate this trade-off if they are optimized for the appropriate objective (Riemer et al., 2019), in general these models do not have available supervision on how to do this and so optimize greedily for the current example. As such, early experiments with the baseline, non-routed models show that examples with neutral signatures, such as ○ | ○ and ○ | +, are the first to be learned. We noticed that baseline sequence models learned to classify *take vow* much earlier in the training process, as the word *vow* is lexicalized through training on *make vow* examples. However, its performance decreases over time as *take* is lexicalized through training on examples of other implicatives with different signatures, such as *take chance*. This behavior is the characteristic outcome of catastrophic interference: learning *take chance* results in decreased performance on *take vow* as the two examples interfere. This is a particularly revealing instance as these two phrasal verbs are similar on the surface, but have quite different semantic properties, as reflected by their signatures. However, given that RRNs are able to route them differently (Figure 6), interference is less likely. It is also possible that routing helps with the transfer related to similar examples, an avenue we want to explore in the future.

5 Prediction without Meta-Information

We have shown that high-quality meta-information can be extremely useful. Unfortunately, it is often not available at test-time. To compensate, we evaluated an RRN extension in which an additional neural network module is trained to assign examples to meta-information classes (genres for MULTINLI; signatures for SCI). When we trained this model jointly, we found that it was unstable and did not perform well. However, if this model is introduced after RRN training on examples with known meta-information, then the results are extremely promising. We call this *Dispatcher Training* ('+D'), borrowing similar terminology from Rosenbaum et al. (2017). Table 1 includes an initial evaluation of this variant with a CBOW base model and WP routing. As we can see, accuracy actually *increases* by a small amount over WP alone.

Additionally, having +D allows the network to generalize better to unseen examples and unseen patterns. Consider the relative performance drop for CBOW WP+D from the full joint SCI dataset (75.56%) to disjoint (−0.69%) and from disjoint (74.87%) to mismatch (−3.79%), and compare this to the plain WP version: full (74.95%) to disjoint (−1.04%), and disjoint (73.91%) to mismatch (−5.22%)

6 Conclusion

This paper introduced Recursive Routing Networks and showed how to incorporate them into a variety of different neural architectures; we explored a range of possibilities for this, and the techniques generalize to other options straightforwardly. Our evaluations focused on NLI. We showed in particular that our RRNs can effectively leverage the meta-information in the MULTINLI corpus and in our new corpus focused on implicatives; not only do RRNs use this information to achieve superior accuracy, but they also learn sub-structure that reflects this high-level information, and our Dispatcher variant extends the framework to situations where the relevant meta-information is not available for testing. We believe exploring more powerful variants of dispatching is an interesting avenue for future work, as is pretraining routing models on language model tasks using large corpora. It is our hope that these lessons extend to other richly compositional, context-sensitive language understanding tasks.

Acknowledgements

We thank George Supaniratisai, Arun Chaganty, Kenny Xu and Abi See for valuable discussions, and the anonymous reviewers for their useful suggestions. Clemens Rosenbaum was a recipient of an IBM PhD Fellowship while working on this publication. We acknowledge the Office of the Vice Provost for Undergraduate Education at Stanford for the summer internships for Atticus Geiger, Olivia Li and Sandhini Agarwal. This research is based in part upon work supported by the Stanford Data Science Initiative, by the NSF under Grant No. BCS-1456077, by the NSF Award IIS-1514268, and by the Air Force Research Laboratory and DARPA under agreement number FA8750-18-2-0126. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2015. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554. Association for Computational Linguistics.

Rebekah Baglini and Itamar Francez. 2016. The implications of managing. *Journal of Semantics*, 33:541–560.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2015. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*.

Michael B Chang, Abhishek Gupta, Sergey Levine, and Thomas L Griffiths. 2018. Automatically composing representation transformations as a means for generalization. *arXiv preprint arXiv:1807.04640*.

Michael L. Geis and Arnold M. Zwicky. 1971. On invited inferences. *Linguistic Inquiry*, 2(4):561–566.

John J. Godfrey and Ed Holliman. 1997. Switchboard-1 release 2. Linguistic Data Consortium, Catalog #LDC97S62.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. *CoRR*, abs/1704.05526.

Robert A. Jacobs, Michael I. Jordan, and Andrew G. Barto. 1991. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Theo M. V. Janssen. 1997. Compositionality. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. MIT Press and North-Holland, Cambridge, MA and Amsterdam.

Lauri Karttunen. 1971. Implicative verbs. *Language*, 47(2):340–358.

Lauri Karttunen. 2012. Simple and phrasal implicatives. In **SEM 2012*, pages 124–131. Association for Computational Linguistics, Montréal, Canada.

Lauri Karttunen. 2016. Presupposition: What went wrong? In Mary Moroney, Carol-Rose Little, Jacob Collard, and Dan Burgdorf, editors, *Semantics and Linguistic Theory (SALT) 26*, pages 705–731. Cornell University, Ithaca, NY.

Lauri Karttunen and Stanley Peters. 1979. Conventional implicature. In Choon-Kyu Oh and David A. Dinneen, editors, *Syntax and Semantics, Volume 11: Presupposition*, pages 1–56. Academic Press, New York.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Louis Kirsch, Julius Kunze, and David Barber. 2018. Modular networks: Learning to decompose neural computation. *arXiv preprint arXiv:1811.05249*.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *The 8th International Conference on Computational Semantics (IWCS-8)*, pages 140–156. University of Tilburg, Tilburg, Netherlands.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109 – 165.

Rowan Nairn, Lauri Karttunen, and Cleo Condoravdi. 2006. Computing relative polarity for textual inference. In Johan Bos and Alexander Koller, editors, *Inference in Computational Semantics (ICoS-5)*, pages 67–76. University of Manchester, Manchester, UK.

Barbara H. Partee. 1984. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*, pages 281–311. Foris, Dordrecht.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Continual learning by maximizing transfer and minimizing interference. *ICLR*.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 656–661.

Appendices

A Architecture Details

A.1 The reward design

Rosenbaum et al. (2017) found that it was helpful in incentivizing different routing strategies to add a regularization reward to each action made. This reward is constructed by computing the average count of choosing a particular module, multiplied by a scaling factor. We also use a similar reward. In particular, we compute it for module m_k and action a as follows:

$$r_c(m_k) = \begin{cases} (1-\alpha) * r'_c(m_k) + \alpha & \text{if } k = a \\ r'_c(m), & \text{otherwise} \end{cases}$$

As we want this reward to sum to 1 over all modules, we normalize. Additionally, we want to scale this reward to stay constant even for larger routing depths. The result is therefore scaled as such:

$$r_c = \frac{r_c}{d \sum r_c}$$

A.2 Losses

To make the backpropagation step to train the network more efficient, we combine the losses

$$\mathcal{L}_{total}(x, \hat{y}) = \mathcal{L}_{class}(x, \hat{y}) + \mathcal{L}_{RL}(x, r) \qquad (5)$$

where $\mathcal{L}_{RL}(x, r)$ is the loss defined by the chosen RL algorithm.

A.3 Routing RNNs

We modify LSTMs as originally defined by Hochreiter and Schmidhuber (1997):

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \qquad (6)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \qquad (7)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \qquad (8)$$

$$\begin{aligned} c_t = {} & f_t \circ c_{t-1} + i_t \circ \\ & \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \end{aligned} \qquad (9)$$

$$h_t = o_t \circ \sigma_h(c_t) \qquad (10)$$

However, analogous architectures can be implemented for GRUs or different LSTMs. Here, either architecture has two core transformations – one from the hidden state at time step t_k to time step t_{k+1}, and one from the input (in our case, word embeddings) to the hidden state. We have designed architectures that route either.

The resulting equations (6), (7), and (8) for the input-to-hidden routing model illustrated in Figure 3, top right, become (with g being a placeholder for f, i, o):

$$\begin{aligned} g_t = {} & \sigma_g(\text{FC}_{g,k_d} \circ \text{FC}_{g,k_{d-1}} \circ \cdots \text{FC}_{g,k_d} x_t \\ & + U_g h_{t-1} + b_g) \end{aligned} \qquad (11)$$

Similarly, the equations (6), (7), and (8) for the hidden-to-hidden routing model in Figure 3, bottom right, become (with g being a placeholder for f, i, o):

$$\begin{aligned} g_t = {} & \sigma_g(W_g x_t + \text{FC}_{g,k_d} \circ \text{FC}_{g,k_{d-1}} \circ \cdots \\ & \text{FC}_{g,k_d} h_{t-1} + b_g) \end{aligned} \qquad (12)$$

A.4 Routing Transformers

The following describes the two different approaches for routing transformers. When routing entire transformer blocks (TB), we have:

$$\mathcal{F}_{k_t} = \{\text{TB}_{t,1}, \text{TB}_{t,2}, \ldots, \text{TB}_{t,b}\} \qquad (13)$$

However, as the routing functions for routing the attention layers are interleaved with the other block-layers (BL), the basic routing chain changes to:

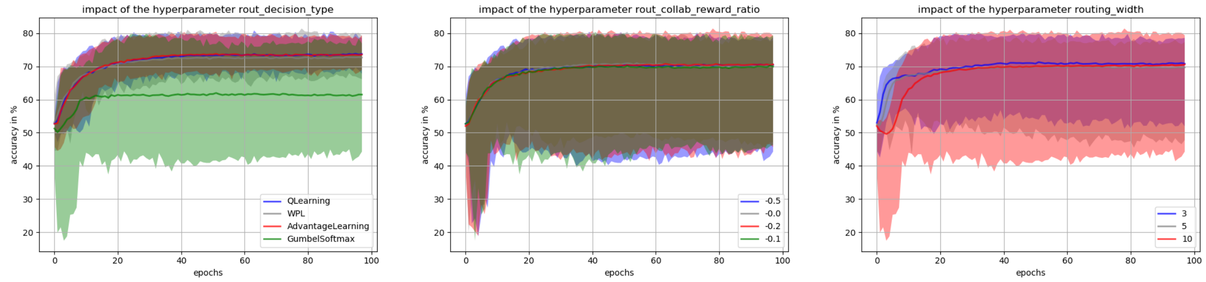
$$f_R(x) = f_{k_1} \circ \text{BL} \circ \ldots \circ \text{BL} \circ f_{k_d}(x) \circ \text{BL} \qquad (14)$$

and the routing module sets are (with AL as attention layers):

$$\mathcal{F}_{k_t} = \{\text{AL}_{t,1}, \text{AL}_{t,2}, \ldots, \text{AL}_{t,b}\} \qquad (15)$$

A.5 Hyperparameter Search

We found that training the router with Adam can become unstable and stay at random baseline performance. In contrast, we found that using Adam for the modules increased performance by around 1%. When training a routing utilizing the Gumbel-Softmax reparameterization, we tested Adam and SGD exclusively, and settled for Adam. We found the module learning rate of $1e{-}3$ and the router learning rate of $3e{-}4$ by performing sweeps over the learning rate from $1e{-}2$ to $1e{-}4$ for the module training and from $1e{-}2$ to $1e{-}5$ for the router training, roughly by decrements of factor 3. In general, we found a branching size of $b = 3$ to work best (although performance for $b = 5$ was nearly identical), after sweeps for $b \in \{2, 3, 5, 10, 15, 20\}$. As we also specify the maximum recursion depth, we generally allow up to $d = 3$ recursive steps. Higher ds add to training time, but neither increase nor decrease performance significantly, as the router (nearly) never learns to apply more than three transformations anyway. Figure A1 shows the effects of the value of the decision making algorithm, the value of the regularization reward, and of the routing width. Most notably, Gumbel-Softmax routing performs



(a) Effect of the decision-making algorithm.

(b) Effect of the regularization reward value.

(c) Effect of the routing width.

Figure A1: Encoder Routing Strategies.

on average 10% worse than RL algorithms, with variance that is nearly an order of magnitude larger. Also, larger routing width has similar expected performance but increases the variance. The value of the regularization reward has little effect on expected performance.

B Implicatives Experiments

Routed models that achieve good performance on SCI show a strong indication of proper learning of entailment and contradiction, and some difficulties with neutral cases, as indicated by the confusion matrix in Table A1. An inspection of common errors of the CBOW WP model shows a number of cases related to the probabilistic component of the signature. For example, *Jerry got a chance to pitch* was marked as contradicting *Jerry did not manage to pitch*, when the gold label is indeed *permits*. In this case, if the premise is true then there is a small chance that the author would regard the hypothesis as false and therefore predict *contradicts* as the model did. Other errors involve coreference, such as the prediction of entailment between *you took the time to review my presentation* and *you reviewed his presentation*.

Relation	entailment	contradiction	neutral	Totals
entailment	35	2	10	47
contradiction	2	33	5	40
neutral	7	6	28	41
Totals:	44	41	43	128

Table A1: Confusion Matrix.

C Implicatives Corpus

Implicative constructions yield a judgment about the truth of their complement clause under at least one polarity. Some are simple verbs like *manage* and *fail*, while some are phrasal constructions like *meet one's duty* and *waste a chance*.

C.1 Signatures

The signature of an implicative indicates the relation between the matrix clause and complement clause. The symbol + indicates an entailment of the complement clause, the symbol − indicates an entailment of the negation of the complement clause, and the symbol *o* indicates no entailment relation to the complement clause. A signature consists of a left symbol, which indicates the entailment relation to the complement clause when the matrix clause is in a positive environment, and the right symbol, which indicates the entailment relation to the complement clause when the matrix clause is in a negative environment.

For example, in the notation of MacCartney and Manning (2009) and Karttunen (2016), the signature of *manage* is + | −. This means a sentence with *manage* entails its complement in positive contexts, and contradicts its complement in negative contexts. As such, *Joan didn't manage to solve the problem* contradicts *Joan solved the problem.*

Fail and *lack foresight* have the implicative signature − | +. *NP failed/lacked the foresight to VP* contradicts *NP VPed* and entails *NP didn't VP*. This is negative entailment under positive polarity (−), positive entailment under negative polarity (+).

Two-way implicatives like *manage* and *fail* yield an entailment under positive and negative polarity. One-way implicatives such as *force* (+ | ○), *prevent* (− | ○), *be able* (○ | −), and *hesitate* (○ | +) yield an entailment, + or −, only under one polarity and no entailment (○) under the other.[6]

---

[6] In addition to the entailment properties that we focus on in this study, most implicative constructions also have another component of meaning, sometimes called *presupposition* (Karttunen, 1971; Baglini and Francez, 2016), some-

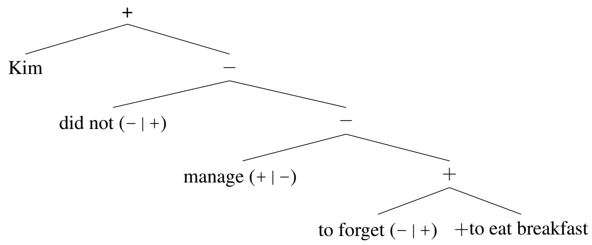


Figure A2: Computation of the inference *Kim ate breakfast* entailed by the premise *Kim did not manage to forget to eat breakfast*.

There are seven known implicative signatures, six of them documented earlier (Karttunen, 1971, 2012). In the course of collecting data for this study, we came across a new implicative construction that had been missed in all previous studies. We code *take no time* and *waste no time* as + | + because, in the vast majority of cases we looked at, the presence or absence of negation makes no difference. *I wasted no time to jump on board* and *I didn't waste no time to jump on board* both mean that I jumped on board quickly.

C.2 Pragmatic Implicatives

In cases where there is no logical entailment, one-way implicatives may give rise to an *invited inference* (Geis and Zwicky, 1971). *I was able to speak but just didn't know what to say* is not a contradiction. But if the speaker leaves out the *but*-clause, she probably means that not only was she able to speak, but she in fact did speak. There is no invited inference in cases like *Bill Clinton had a chance to kill Bin Laden before the September 11 attacks*, because it is common knowledge that the attempt to do so failed. Insofar as these cases depend on an encyclopedic knowledge about the world, as opposed to a dictionary/lexical knowledge, we use the terminology *pragmatic implicatives* for this type of verbs.

The likelihood of invited inferences varies greatly depending on the construction. They are very likely with *be able*, less likely with *have chance*, and very unlikely with *hesitate*. To better match actual usage, we assign to *be able* not the semantically correct ○ | − signature but `.9` | −, reflecting our estimation that there is a `.9` probability that the author is using the construction as if it were a two-way + | − implicative. We code *have chance* as `.4` | − and *hesitate* as ○ | +.

C.3 Nested Implicatives

Implicative constructions can be nested inside of other implicative constructions, as in *Stan managed to fail to propose to Carol*. The meaning of nested implicatives is compositional; the signature of a nested implicative can be determined algorithmically by the signatures of the single implicatives that compose it (Nairn et al. 2006; see Figure A2). Since *manage* has the signature + | − and *fail* is − | +, the nested constructions *manage to fail* and *fail to manage* are both − | +. Thus, the example above entails that Stan did not propose to Carol, as does *Stan failed to manage to propose to Carol*. The entailments are the same, but the implications associated with *manage* and *fail* differ depending on the nesting.

Some implicative constructions – e.g., *manage* and *fail*, *remember*, and *forget* – can be nested in one another. In many cases, only one order makes sense. It is easy to find examples of *happen to turn out* – e.g., *it happened to turn out to be a success* – but a search for *turn out to happen* sentences does not turn up any. One can even find triply nested implicative constructions, such as *don't forget to remember not to forget to watch the finale of series 2 tonight*, but they are rare.

The single-word, phrasal implicatives, and (a few) nested implicatives used in our data set are listed in Table A3. The release version of SCI will contain additional doubly-nested constructions (the complete list is in Table A4), and will have additional splits.

C.4 Corpus Versions and Extension

The three different versions of the dataset described in the article are *joint*, *disjoint*, and *mismatch*. The *disjoint* test set contains the implicatives listed in Table A2, and thus the implicatives listed there are not contained in train and validation.

The *mismatch* version of the SCI corpus contains all signatures in validation and test, but only a restricted set for train. Table A5 contains the subsets available on each split.

---

times *conventional implicature* (Karttunen and Peters, 1979). For example, *Marco failed to show up for work* implicates that Marco tried or was expected to show up for work. The *NatLog* system (MacCartney and Manning, 2009) computes entailments of implicatives but no presuppositions. No current NLP system does. The difference between entailment and presupposition/conventional implicature can be made with the question *Did Marco fail to show up for work?* This question asks whether Marco showed up for work or not; *fail* indicates that the speaker thinks that Marco tried or was expected to come to work.

In addition to the versions, we included all the nested constructions as a separate extension to the dataset using the same format. This expansion can then be easily used as a test set for generalization to longer sequences.

Figure A3 show the distribution of lengths, counted in word tokens, for premises and hypotheses. Figure A4 shows the distribution of lengths for both premises and hypotheses.

Construction	Sign	Examples
break promise	− | +	90
exploit opportunity	+ | −	120
fail obligation	− | +	60
fulfill promise	+ | −	90
grab occasion	+ | −	120
make vow	○ | ○	91
manage	+ | −	90
meet duty	+ | −	90
miss occasion	− | +	120
neglect	− | +	91
refrain	− | +	120
require	+ | . 3	120
seize opportunity	+ | −	120
take opportunity	+ | −	91
use occasion	+ | −	120
waste chance	− | +	91
waste no time	+ | +	76
waste opportunity	− | +	91
Constructions: 18	Total	1791

Table A2: Disjoint dataset in test set in SCI.

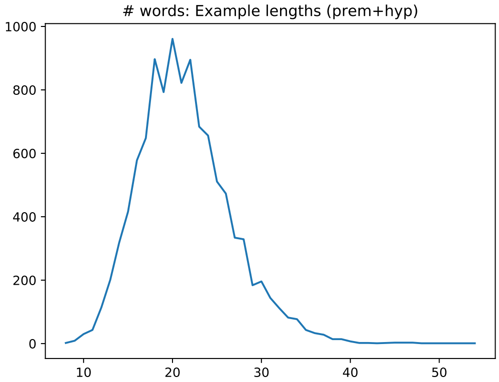


Figure A3: Distribution of sentence lengths in number of words for all verbs in SCI.

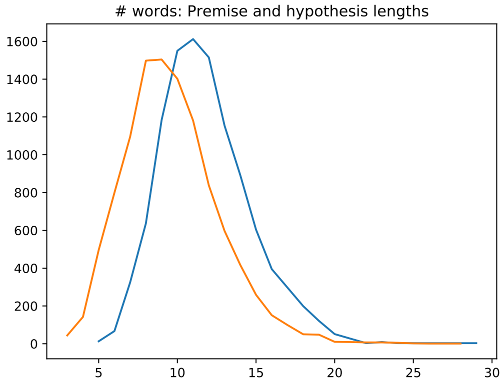


Figure A4: Distribution of lengths for premises (blue) and hypothesis (orange) in SCI.

Construction	Sign	Examples	Construction	Sign	Examples
be forced	+ | −	128	keep promise	+ | −	90
be kept from	− | .9	132	lack foresight	− | +	91
be made	+ | .5	126	lose opportunity	− | +	90
be obstructed from	− | .9	126	make	+ | .5	126
be prevented	− | .7	87	make promise	○ | ○	91
be required	+ | .3	120	make vow	○ | ○	91
be stopped	− | .5	120	manage	+ | −	90
bother	+ | −	91	manage to miss opportunity	− | +	120
breach contract	− | +	120	meet duty	+ | −	90
breach promise	− | +	120	meet obligation	+ | −	90
break pledge	− | +	73	meet promise	+ | −	88
break promise	− | +	90	miss chance	− | +	90
bring oneself	+ | −	120	miss occasion	− | +	120
cause	+ | .5	120	miss opportunity	− | +	91
coerce	+ | .9	120	neglect	− | +	91
compel	+ | .4	120	neglect duty	− | +	120
convince	+ | .8	120	neglect occasion	− | +	120
dare	+ | −	91	obey order	+ | −	91
disobey order	− | +	87	overlook opportunity	− | +	120
exploit occasion	+ | −	120	prevent	− | .7	60
exploit opportunity	+ | −	120	proceed	+ | −	120
fail	− | +	91	refrain	− | +	120
fail obligation	− | +	60	refuse to take advantage	− | +	114
fail to take advantage	− | +	120	remember	+ | −	91
follow order	+ | −	90	require	+ | .3	120
force oneself	+ | −	120	seize occasion	+ | −	120
forget	− | +	90	seize opportunity	+ | −	120
fulfill promise	+ | −	90	stop	− | .5	120
get chance	.9 | −	90	succeed	+ | −	120
get	+ | −	119	take advantage of opportunity	+ | −	120
grab occasion	+ | −	120	take chance	+ | −	91
grab opportunity	+ | −	120	take no time	+ | +	76
happen	+ | −	92	take occasion	+ | −	120
have chance	.4 | −	89	take opportunity	+ | −	91
have courage	+ | −	91	take time	+ | −	88
have	+ | .5	120	take vow	○ | ○	91
have foresight	+ | −	90	turn out	+ | −	153
have gall	+ | −	90	use occasion	+ | −	120
have time	.9 | −	90	use opportunity	+ | −	120
have wherewithal	+ | −	90	venture	+ | −	120
hazard	+ | −	120	waste chance	− | +	91
hesitate	○ | +	91	waste money	+ | −	91
ignore duty	− | +	120	waste no time	+ | +	76
ignore opportunity	+ | −	120	waste occasion	− | +	120
jump on occasion	+ | −	120	waste opportunity	− | +	91
keep from	− | .9	132	waste time	+ | −	91
			Constructions: 92	Total	9671

Table A3: Implicatives in joint SCI.

Construction	Sign	Examples
be able to bring oneself	+ | −	160
fail to have courage	− | +	66
fail to have means	− | +	36
fail to have wherewithal	− | +	18
fail to manage	− | +	90
fail to meet obligation	− | +	54
fail to obey order	− | +	51
fail to take advantage	− | +	120
forget to remember	− | +	48
manage to fail	− | +	54
manage to forget	− | +	18
manage to have gall	+ | −	18
manage to have power	+ | −	18
manage to have strength	+ | −	54
manage to have wherewithal	+ | −	18
manage to miss opportunity	− | +	166
manage to remember	+ | −	53
manage to turn out	+ | −	36
refuse to take advantage	− | +	114
remember to forget	− | +	359
remember to take time	+ | −	54
take advantage of opportunity	+ | −	120
Constructions: 22	Total	1725

Table A4: Nested Implicatives collected to appear in the release version of SCI.

<table>
<tr><th>Split</th><th>Signatures</th></tr>
<tr><td>train</td><td>-|+, +|-, o|-, o|o, +|.5, .4|-, -|.7</td></tr>
<tr><td>dev/test</td><td>-|+, +|-, o|-, o|o, +|.3, .4|-, +|.4, +|.5, -|.5, -|.7, +|.8, +|.9, -|.9, .9|-</td></tr>
</table>

Table A5: Mismatch Splits in SCI.

Structural Neural Encoders for AMR-to-text Generation

Marco Damonte **Shay B. Cohen**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
`m.damonte@sms.ed.ac.uk`
`scohen@inf.ed.ac.uk`

Abstract

AMR-to-text generation is a problem recently introduced to the NLP community, in which the goal is to generate sentences from Abstract Meaning Representation (AMR) graphs. Sequence-to-sequence models can be used to this end by converting the AMR graphs to strings. Approaching the problem while working directly with graphs requires the use of graph-to-sequence models that encode the AMR graph into a vector representation. Such encoding has been shown to be beneficial in the past, and unlike sequential encoding, it allows us to explicitly capture reentrant structures in the AMR graphs. We investigate the extent to which reentrancies (nodes with multiple parents) have an impact on AMR-to-text generation by comparing graph encoders to tree encoders, where reentrancies are not preserved. We show that improvements in the treatment of reentrancies and long-range dependencies contribute to higher overall scores for graph encoders. Our best model achieves 24.40 BLEU on LDC2015E86, outperforming the state of the art by 1.1 points and 24.54 BLEU on LDC2017T10, outperforming the state of the art by 1.24 points.

1 Introduction

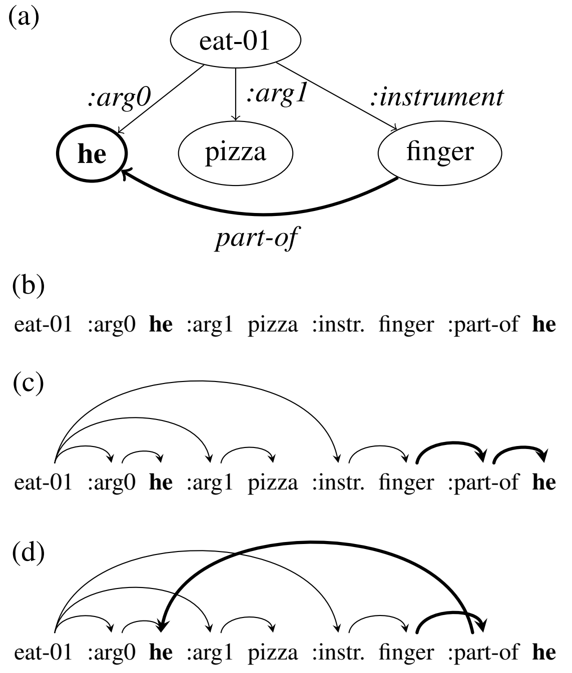


Figure 1: (a) AMR for the sentence *He ate the pizza with his fingers* and different input representations: (b) sequential; (c) tree-structured; (d) graph-structured. The nodes and edges in bold highlight a reentrancy.

Abstract Meaning Representation (AMR; Banarescu et al. 2013) is a semantic graph representation that abstracts away from the syntactic realization of a sentence, where nodes in the graph represent concepts and edges represent semantic relations between them. AMRs are graphs, rather than trees, because co-references and control structures result in nodes with multiple parents, called reentrancies. For instance, the AMR of Figure 1(a) contains a reentrancy between *finger* and *he*, caused by the possessive pronoun *his*. AMR-to-text generation is the task of automatically generating natural language from AMR graphs.

Attentive encoder/decoder architectures, commonly used for Neural Machine Translation (NMT), have been explored for this task (Konstas et al., 2017; Song et al., 2018; Beck et al., 2018). In order to use sequence-to-sequence models, Konstas et al. (2017) reduce the AMR graphs to sequences, while Song et al. (2018) and Beck et al. (2018) directly encode them as graphs. Graph encoding allows the model to explicitly encode reentrant structures present in the AMR graphs. While central to AMR, reentrancies are often hard to treat both in parsing and in generation. Previous work either removed them from the graphs, hence obtaining sequential (Konstas et al.,

Proceedings of NAACL-HLT 2019, pages 3649–3658
Minneapolis, Minnesota, June 2 - June 7, 2019.

2017) or tree-structured (Liu et al., 2015; Takase et al., 2016) data, while other work maintained them but did not analyze their impact on performance (e.g., Song et al., 2018; Beck et al., 2018). Damonte et al. (2017) showed that state-of-the-art parsers do not perform well in predicting reentrant structures, while van Noord and Bos (2017) compared different pre- and post-processing techniques to improve the performance of sequence-to-sequence parsers with respect to reentrancies. It is not yet clear whether explicit encoding of reentrancies is beneficial for generation.

In this paper, we compare three types of encoders for AMR: 1) sequential encoders, which reduce AMR graphs to sequences; 2) tree encoders, which ignore reentrancies; and 3) graph encoders. We pay particular attention to two phenomena: reentrancies, which mark co-reference and control structures, and long-range dependencies in the AMR graphs, which are expected to benefit from structural encoding. The contributions of the paper are two-fold:

- We present structural encoders for the encoder/decoder framework and show the benefits of graph encoders not only compared to sequential encoders but also compared to tree encoders, which have not been studied so far for AMR-to-text generation.
- We show that better treatment of reentrancies and long-range dependencies contributes to improvements in the graph encoders.

Our best model, based on a graph encoder, achieves state-of-the-art results for both the LDC2015E86 dataset (24.40 on BLEU and 23.79 on Meteor) and the LDC2017T10 dataset (24.54 on BLEU and 24.07 on Meteor).

2 Input Representations

Graph-structured AMRs AMRs are normally represented as rooted and directed graphs:

$$\begin{aligned}&G_0 = (V_0, E_0, L),\\ &V_0 = \{v_1, v_2, \ldots, v_n\},\\ &root \in V_0,\end{aligned}$$

where V_0 are the graph vertices (or nodes) and $root$ is a designated root node in V_0. The edges in the AMR are labeled:

$$\begin{aligned}&E_0 \subseteq V_0 \times L \times V_0,\\ &L = \{\ell_1, \ell_2, \ldots, \ell_{n'}\}.\end{aligned}$$

Each edge $e \in E_0$ is a triple: $e = (i, label, j)$, where $i \in V_0$ is the parent node, $label \in L$ is the edge label and $j \in V_0$ is the child node.

In order to obtain unlabeled edges, thus decreasing the total number of parameters required by the models, we replace each labeled edge $e = (i, label, j)$ with two unlabeled edges: $e_1 = (i, label), e_2 = (label, j)$:

$$\begin{aligned}&G = (V, E),\\ &V = V_0 \cup L = \{v_1, \ldots, v_n, \ell_1, \ldots, \ell_{n'}\},\\ &E \subseteq (V_0 \times L) \cup (L \times V_0).\end{aligned}$$

Each unlabeled edge $e \in E$ is a pair: $e = (i, j)$, where one of the following holds:

1. $i \in V_0$ and $j \in L$;
2. $i \in L$ and $j \in V_0$.

For instance, the edge between *eat-01* and *he* with label *:arg0* of Figure 1(a) is replaced by two edges in Figure 1(d): an edge between *eat-01* and *:arg0* and another one between *:arg0* and *he*. The process, also used in Beck et al. (2018), tranforms the input graph into its equivalent Levi graph (Levi, 1942).

Tree-structured AMRs In order to obtain tree structures, it is necessary to discard the reentrancies from the AMR graphs. Similarly to Takase et al. (2016), we replace nodes with $n > 1$ incoming edges with n identically labeled nodes, each with a single incoming edge.

Sequential AMRs Following Konstas et al. (2017), the input sequence is a linearized and anonymized AMR graph. Linearization is used to convert the graph into a sequence:

$$\begin{aligned}&x = x_1, \ldots, x_N,\\ &x_i \in V.\end{aligned}$$

The depth-first traversal of the graph defines the indexing between nodes and tokens in the sequence. For instance, the root node is x_1, its leftmost child is x_2 and so on. Nodes with multiple parents are visited more than once. At each visit, their labels are repeated in the sequence, effectively losing reentrancy information, as shown in Figure 1(b).

Anonymization removes names and rare words with coarse categories to reduce data sparsity. An alternative to anonymization is to employ a copy mechanism (Gulcehre et al., 2016), where the models learn to copy rare words from the input itself. In this paper, we follow the anonymization approach.

3 Encoders

In this section, we review the encoders adopted as building blocks for our tree and graph encoders.

3.1 Recurrent Neural Network Encoders

We reimplement the encoder of Konstas et al. (2017), where the sequential linearization is the input to a bidirectional LSTM (BiLSTM; Graves et al. 2013) network. The hidden state of the BiLSTM at step i is used as a context-aware word representation of the i-th token in the sequence:

$$e_{1:N} = \mathrm{BiLSTM}(x_{1:N}),$$

where $e_i \in \mathbb{R}^d$, d is the size of the output embeddings.

3.2 TreeLSTM Encoders

Tree-Structured Long Short-Term Memory Networks (TreeLSTM; Tai et al. 2015) have been introduced primarily as a way to encode the hierarchical structure of syntactic trees (Tai et al., 2015), but they have also been applied to AMR for the task of headline generation (Takase et al., 2016). TreeLSTMs assume tree-structured input, so AMR graphs must be preprocessed to respect this constraint: reentrancies, which play an essential role in AMR, must be removed, thereby transforming the graphs into trees.

We use the Child-Sum variant introduced by Tai et al. (2015), which processes the tree in a bottom-up pass. When visiting a node, the hidden states of its children are summed up in a single vector which is then passed into recurrent gates.

In order to use information from both incoming and outgoing edges (parents and children), we employ bidirectional TreeLSTMs (Eriguchi et al., 2016), where the bottom-up pass is followed by a top-down pass. The top-down state of the root node is obtained by feeding the bottom-up state of the root node through a feed-forward layer:

$$h^{\downarrow}_{\text{root}} = \tanh(W_r h^{\uparrow}_{\text{root}} + b),$$

where $h^{\uparrow}_i$ is the hidden state of node $x_i \in V$ for the bottom-up pass and $h^{\downarrow}_i$ is the hidden state of node x_i for the top-down pass.

The bottom up states for all other nodes are computed with an LSTM, with the cell state given by their parent nodes:

$$h^{\downarrow}_i = \mathrm{LSTM}(h^{\uparrow}_{p(i)}, h^{\uparrow}_i),$$

where $p(i)$ is the parent of node x_i in the tree. The final hidden states are obtained by concatenating the states from the bottom-up pass and the top-down pass:

$$h_i = \left[h^{\downarrow}_i; h^{\uparrow}_i\right].$$

The hidden state of the root node is usually used as a representation for the entire tree. In order to use attention over all nodes, as in traditional NMT (Bahdanau et al., 2015), we can however build node embeddings by extracting the hidden states of each node in the tree:

$$e_{1:N} = h_{1:N},$$

where $e_i \in \mathbb{R}^d$, d is the size of the output embeddings.

The encoder is related to the TreeLSTM encoder of Takase et al. (2016), which however encodes labeled trees and does not use a top-down pass.

3.3 Graph Convolutional Network Encoders

Graph Convolutional Network (GCN; Duvenaud et al. 2015; Kipf and Welling 2016) is a neural network architecture that learns embeddings of nodes in a graph by looking at its nearby nodes. In Natural Language Processing, GCNs have been used for Semantic Role Labeling (Marcheggiani and Titov, 2017), NMT (Bastings et al., 2017), Named Entity Recognition (Cetoli et al., 2017) and text generation (Marcheggiani and Perez-Beltrachini, 2018).

A graph-to-sequence neural network was first introduced by Xu et al. (2018). The authors review the similarities between their approach, GCN and another approach, based on GRUs (Li et al., 2015). The latter recently inspired a graph-to-sequence architecture for AMR-to-text generation (Beck et al., 2018). Simultaneously, Song et al. (2018) proposed a graph encoder based on LSTMs.

The architectures of Song et al. (2018) and Beck et al. (2018) are both based on the same core computation of a GCN, which sums over the embeddings of the immediate neighborhood of each node:

$$h_i^{(k+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} W_{(j,i)}^{(k)} h_j^{(k)} + b^{(k)}\right),$$

where $h_i^{(k)}$ is the embeddings of node $x_i \in V$ at layer k, σ is a non-linear activation function, $\mathcal{N}(i)$ is the set of the immediate neighbors of x_i, $W_{(j,i)}^{(k)} \in \mathbb{R}^{m \times m}$ and $b^{(k)} \in \mathbb{R}^m$, with m being the size of the embeddings.

It is possible to use recurrent networks to model the update of the node embeddings. Specifically, Beck et al. (2018) uses a GRU layer where the gates are modeled as GCN layers. Song et al. (2018) did not use the activation function σ and perform an LSTM update instead.

The systems of Song et al. (2018) and Beck et al. (2018) further differ in design and implementation decisions such as in the use of edge label and edge directionality. Throughout the rest of the paper, we follow the traditional, non-recurrent, implementation of GCN also adopted in other NLP tasks (Marcheggiani and Titov, 2017; Bastings et al., 2017; Cetoli et al., 2017). In our experiments, the node embeddings are computed as follows:

$$h_i^{(k+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} W_{\mathrm{dir}(j,i)}^{(k)} h_j^{(k)} + b^{(k)}\right), \quad (1)$$

where $\mathrm{dir}(j,i)$ indicates the direction of the edge between x_j and x_i (i.e., outgoing or incoming edge). The hidden vectors from the last layer of the GCN network are finally used to represent each node in the graph:

$$e_{1:N} = h_1^{(K)}, \ldots, h_N^{(K)},$$

where K is the number of GCN layers used, $e_i \in \mathbb{R}^d$, d is the size of the output embeddings.

To regularize the models we apply dropout (Srivastava et al., 2014) as well as edge dropout (Marcheggiani and Titov, 2017). We also include highway connections (Srivastava et al., 2015) between GCN layers.

While GCN can naturally be used to encode graphs, they can also be applied to trees by removing reentrancies from the input graphs. In the experiments of Section 5, we explore GCN-based models both as graph encoders (reentrancies are maintained) as well as tree encoders (reentrancies are ignored).

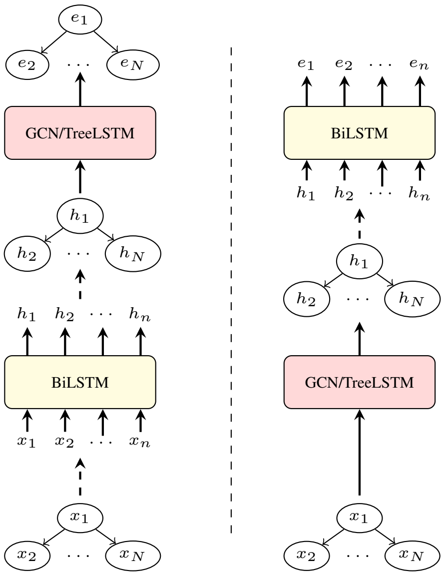


Figure 2: Two ways of stacking recurrent and structural models. Left side: structure on top of sequence, where the structural encoders are applied to the hidden vectors computed by the BiLSTM. Right side: sequence on top of structure, where the structural encoder is used to create better embeddings which are then fed to the BiLSTM. The dotted lines refer to the process of converting the graph into a sequence or vice-versa.

4 Stacking Encoders

We aimed at stacking the explicit source of structural information provided by TreeLSTMs and GCNs with the sequential information which BiLSTMs extract well. This was shown to be effective for other tasks with both TreeLSTMs (Eriguchi et al., 2016; Chen et al., 2017) and GCNs (Marcheggiani and Titov, 2017; Cetoli et al., 2017; Bastings et al., 2017). In previous work, the structural encoders (tree or graph) were used on top of the BiLSTM network: first, the input is passed through the sequential encoder, the output of which is then fed into the structural encoder. While we experiment with this approach, we also propose an alternative solution where the BiLSTM network is used on top of the structural encoder: the input embeddings are refined by exploiting the explicit structural information given by the graph. The refined embeddings are then fed into the BiLSTM networks. See Figure 2 for a graphical representation of the two approaches. In our experiments, we found this approach to be more effective. Compared to models that interleave structural and recurrent components such as the systems of Song et al. (2018) and Beck et al. (2018), stacking the components allows us to test for their contributions more easily.

4.1 Structure on Top of Sequence

In this setup, BiLSTMs are used as in Section 3.1 to encode the linearized and anonymized AMR. The context provided by the BiLSTM is a sequential one. We then apply either GCN or TreeLSTM on the output of the BiLSTM, by initializing the GCN or TreeLSTM embeddings with the BiLSTM hidden states. We call these models SeqGCN and SeqTreeLSTM.

4.2 Sequence on Top of Structure

We also propose a different approach for integrating graph information into the encoder, by swapping the order of the BiLSTM and the structural encoder: we aim at using the structured information provided by the AMR graph as a way to refine the original word representations. We first apply the structural encoder to the input graphs. The GCN or TreeLSTM representations are then fed into the BiLSTM. We call these models GCNSeq and TreeLSTMSeq.

The motivation behind this approach is that we know that BiLSTMs, given appropriate input embeddings, are very effective at encoding the input sequences. In order to exploit their strength, we do not amend their output but rather provide them with better input embeddings to start with, by explicitly taking the graph relations into account.

5 Experiments

We use both BLEU (Papineni et al., 2002) and Meteor (Banerjee and Lavie, 2005) as evaluation metrics.[1] We report results on the AMR dataset LDC2015E86 and LDC2017T10. All systems are implemented in PyTorch (Paszke et al., 2017) using the framework OpenNMT-py (Klein et al., 2017). Hyperparameters of each model were tuned on the development set of LDC2015E86. For the GCN components, we use two layers, ReLU activations, and tanh highway layers. We use single layer LSTMs. We train with SGD with the initial learning rate set to 1 and decay to 0.8. Batch size is set to 100.[2]

We first evaluate the overall performance of the models, after which we focus on two phenomena that we expect to benefit most from structural encoders: reentrancies and long-range dependencies. Table 1 shows the comparison on the development split of the LDC2015E86 dataset between sequential, tree and graph encoders. The sequential encoder (Seq) is a re-implementation of Konstas et al. (2017). We test both approaches of stacking structural and sequential components: structure on top of sequence (SeqTreeLSTM and SeqGCN), and sequence on top of structure (TreeLSTMSeq and GCNSeq). To inspect the effect of the sequential component, we run ablation tests by removing the RNNs altogether (TreeLSTM and GCN). GCN-based models are used both as tree encoders (reentrancies are removed) and graph encoders (reentrancies are maintained).

Input	Model	BLEU	Meteor
Seq	Seq	21.40	22.00
Tree	SeqTreeLSTM	21.84	22.34
	TreeLSTMSeq	22.26	22.87
	TreeLSTM	22.07	22.57
	SeqGCN	21.84	22.21
	GCNSeq	**23.62**	**23.77**
	GCN	15.83	17.76
Graph	SeqGCN	22.06	22.18
	GCNSeq	**23.95**	**24.00**
	GCN	15.94	17.76

Table 1: BLEU and Meteor (%) scores on the development split of LDC2015E86.

For both TreeLSTM-based and GCN-based models, our proposed approach of applying the structural encoder before the RNN achieves better scores. This is especially true for GCN-based models, for which we also note a drastic drop in performance when the RNN is removed, highlighting the importance of a sequential component. On the other hand, RNN layers seem to have less impact on TreeLSTM-based models. This outcome is not unexpected, as TreeLSTMs already use LSTM gates in their computation.

The results show a clear advantage of tree and graph encoders over the sequential encoder. The best performing model is GCNSeq, both as a tree and as a graph encoder, with the latter obtaining the highest results.

Table 2 shows the comparison between our best sequential (Seq), tree (GCNSeq without reentrancies, henceforth called Tree) and graph en-

---

[1] We used the evaluation script available at `https://github.com/sinantie/NeuralAmr`.

[2] Our code is available at `https://github.com/mdtux89/OpenNMT-py-AMR-to-text`.

Model	**BLEU**	**Meteor**
LDC2015E86		
SEQ	21.43	21.53
TREE	23.93	23.32
GRAPH	**24.40**	**23.60**
Konstas et al. (2017)	22.00	-
Song et al. (2018)	23.30	-
LDC2017T10		
SEQ	22.19	22.68
TREE	24.06	23.62
GRAPH	**24.54**	**24.07**
Beck et al. (2018)	23.30	-

Table 2: Scores on the test split of LDC2015E86 and LDC2017T10. TREE is the tree-based GCNSEQ and GRAPH is the graph-based GCNSEQ.

# reentrancies	**# dev sents.**	**# test sents.**
0	619	622
1-5	679	679
6-20	70	70

Table 3: Counts of reentrancies for the development and test split of LDC2017T10

coders (GCNSEQ with reentrancies, henceforth called GRAPH) on the test set of LDC2015E86 and LDC2017T10. We also include state-of-the-art results reported on these datasets for sequential encoding (Konstas et al., 2017) and graph encoding (Song et al., 2018; Beck et al., 2018).[3] In order to mitigate the effects of random seeds, we train five models with different random seeds and report the results of the median model, according to their BLEU score on the development set (Beck et al., 2018). We achieve state-of-the-art results with both tree and graph encoders, demonstrating the efficacy of our GCNSeq approach. The graph encoder outperforms the other systems and previous work on both datasets. These results demonstrate the benefit of structural encoders over purely sequential ones as well as the advantage of explicitly including reentrancies. The differences between our graph encoder and that of Song et al. (2018) and Beck et al. (2018) were discussed in Section 3.3.

Model	**Number of reentrancies**		
	0	1-5	6-20
SEQ	42.94	31.64	23.33
TREE	+0.63	+1.41	+0.76
GRAPH	**+1.67**	**+1.54**	**+3.08**

Table 4: Differences, with respect to the sequential baseline, in the Meteor score of the test split of LDC2017T10 as a function of the number of reentrancies.

5.1 Reentrancies

Overall scores show an advantage of graph encoder over tree and sequential encoders, but they do not shed light into how this is achieved. Because graph encoders are the only ones to model reentrancies explicitly, we expect them to deal better with these structures. It is, however, possible that the other models are capable of handling these structures implicitly. Moreover, the dataset contains a large number of examples that do not involve any reentrancies, as shown in Table 3, so that the overall scores may not be representative of the ability of models to capture reentrancies. It is expected that the benefit of the graph models will be more evident for those examples containing more reentrancies. To test this hypothesis, we evaluate the various scenarios as a function of the number of reentrancies in each example, using the Meteor score as a metric.[4]

Table 4 shows that the gap between the graph encoder and the other encoders is widest for examples with more than six reentrancies. The Meteor score of the graph encoder for these cases is 3.1% higher than the one for the sequential encoder and 2.3% higher than the score achieved by the tree encoder, demonstrating that explicitly encoding reentrancies is more beneficial than the overall scores suggest. Interestingly, it can also be observed that the graph model outperforms the tree model also for examples with no reentrancies, where tree and graph structures are identical. This suggests that preserving reentrancies in the training data has other beneficial effects. In Section 5.2 we explore one: better handling of long-range dependencies.

---

[3] We run comparisons on systems without ensembling nor additional data.

[4] For this analysis we use Meteor instead of BLEU because it is a sentence-level metric, unlike BLEU, which is a corpus-level metric.

5.1.1 Manual Inspection

In order to further explore how the graph model handles reentrancies differently from the other models, we performed a manual inspection of the models' output. We selected examples containing reentrancies, where the graph model performs better than the other models. These are shown in Table 5. In Example (1), we note that the graph model is the only one that correctly predicts the phrase *he finds out*. The wrong verb tense is due to the lack of tense information in AMR graphs. In the sequential model, the pronoun is chosen correctly, but the wrong verb is predicted, while in the tree model the pronoun is missing. In Example (2), only the graph model correctly generates the phrase *you tell them*, while none of the models use *people* as the subject of the predicate *can*. In Example (3), both the graph and the sequential models deal well with the control structure caused by the *recommend* predicate. The sequential model, however, overgenerates a wh-clause. Finally, in Example (4) the tree and graph models deal correctly with the possessive pronoun to generate the phrase *tell your ex*, while the sequential model does not. Overall, we note that the graph model produces a more accurate output than sequential and tree models by generating the correct pronouns and mentions when control verbs and co-references are involved.

5.1.2 Contrastive Pairs

For a quantitative analysis of how the different models handle pronouns, we use a method to inspect NMT output for specific linguistic analysis based on contrastive pairs (Sennrich, 2017). Given a reference output sentence, a contrastive sentence is generated by introducing a mistake related to the phenomenon we are interested in evaluating. The probability that the model assigns to the reference sentence is then compared to that of the contrastive sentence. The accuracy of a model is determined by the percentage of examples in which the reference sentence has a higher probability than the contrastive sentence.

We produce contrastive examples by running CoreNLP (Manning et al., 2014) to identify co-references, which are the primary cause of reentrancies, and introducing a mistake. When an expression has multiple mentions, the antecedent is repeated in the linearized AMR. For instance, the linearization of Figure 1(b) contains the token *he* twice, which instead appears only once in the sentence. This repetition may result in generating the token *he* twice, rather than using a pronoun to refer back to it. To investigate this possible mistake, we replace one of the mentions with the antecedent (e.g., *John ate the pizza with his fingers* is replaced with *John ate the pizza with John fingers*, which is ungrammatical and as such should be less likely).

An alternative hypothesis is that even when the generation system correctly decides to predict a pronoun, it selects the wrong one. To test for this, we produce contrastive examples where a pronoun is replaced by either a different type of pronoun (e.g., *John ate the pizza with his fingers* is replaced with *John ate the pizza with him fingers*) or by the same type of pronoun but for a different number (*John ate the pizza with their fingers*) or different gender (*John ate the pizza with her fingers*). Note from Figure 1 that the graph-structured AMR is the one that more directly captures the relation between *finger* and *he*, and as such it is expected to deal better with this type of mistakes.

From the test split of LDC2017T10, we generated 251 contrastive examples due to antecedent replacements, 912 due to pronoun type replacements, 1840 due to number replacements and 95 due to gender replacements.[5] The results are shown in Table 6. The sequential encoder performs surprisingly well at this task, with better or on par performance with respect to the tree encoder. The graph encoder outperforms the sequential encoder only for pronoun number and gender replacements. Future work is required to more precisely analyze if the different models cope with pronomial mentions in significantly different ways. Other approaches to inspect phenomena of co-reference and control verbs can also be explored, for instance by devising specific training objectives (Linzen et al., 2016).

5.2 Long-range Dependencies

When we encode a long sequence, interactions between items that appear distant from each other in the sequence are difficult to capture. The problem of long-range dependencies in natural language is well known for RNN architectures (Bengio et al., 1994). Indeed, the need to solve this problem motivated the introduction of LSTM models, which are known to model long-range dependencies better than traditional RNNs.

---

[5] The generated contrastive examples are available at `https://github.com/mdtux89/OpenNMT-py`.

(1)	REF	i dont tell him but **he finds out** ,
	Seq	i did n't tell him but **he was out** .
	Tree	i do n't tell him but **found out** .
	Graph	i do n't tell him but **he found out** .
(2)	REF	if **you tell people** they can help you ,
	Seq	if **you tell him** , you can help you !
	Tree	if **you tell person_name_0 you** , you can help you .
	Graph	if **you tell them** , you can help you .
(3)	REF	**i 'd recommend** you go and see your doctor too .
	Seq	**i recommend** you go to see your doctor who is going to see your doctor .
	Tree	**you recommend** going to see your doctor too .
	Graph	**i recommend** you going to see your doctor too .
(4)	REF	(you) **tell your ex** that all communication needs to go through the lawyer .
	Seq	(you) **tell** that all the communication go through lawyer .
	Tree	(you) **tell your ex** , tell your ex , the need for all the communication .
	Graph	(you) **tell your ex** the need to go through a lawyer .

Table 5: Examples of generation from AMR graphs containing reentrancies. REF is the reference sentence.

Model	**Antec.**	**Type**	**Num.**	**Gender**
Seq	96.02	97.70	94.89	94.74
Tree	96.02	96.38	93.70	92.63
Graph	96.02	96.49	95.11	95.79

Table 6: Accuracy (%) of models, on the test split of LDC201T10, for different categories of contrastive errors: antecedent (Antec.), pronoun type (Type), number (Num.), and gender (Gender).

# max length	**# dev sents.**	**# test sents.**
0-10	292	307
11-50	350	297
51-250	21	18

Table 7: Counts of longest dependencies for the development and test split of LDC2017T10

Model	**Max dependency length**		
	0-10	11-50	51-200
Seq	50.49	36.28	24.14
Tree	-0.48	+1.66	+2.37
Graph	**+1.22**	**+2.05**	**+3.04**

Table 8: Differences, with respect to the sequential baseline, in the Meteor score of the test split of LDC2017T10 as a function of the maximum dependency length.

Because the nodes in the graphs are not aligned with words in the sentence, AMR has no notion of distance between the nodes taking part in an edge. In order to define the length of an AMR edge, we resort to the AMR linearization discussed in Section 2. Given the linearization of the AMR $x_1, \ldots, x_N$, as discussed in Section 2, and an edge between two nodes x_i and x_j, the length of the edge is defined as $|j - i|$. For instance, in the AMR of Figure 1, the edge between *eat-01* and *:instrument* is a dependency of length five, because of the distance between the two words in the linearization *eat-01 :arg0 he :arg1 pizza :instrument*. We then compute the maximum dependency length for each AMR graph.

To verify the hypothesis that long-range dependencies contribute to the improvements of graph models, we compare the models as a function of the maximum dependency length in each example. Longer dependencies are sometimes caused by reentrancies, as in the dependency between *:part-of* and *he* in Figure 1. To verify that the contribution in terms of longer dependencies is complementary to that of reentrancies, we exclude sentences with reentrancies from this analysis. Table 7 shows the statistics for this measure. Results are shown in Table 8. The graph encoder always outperforms both the sequential and the tree encoder. The gap with the sequential encoder increases for longer dependencies. This indicates that longer dependencies are an important factor in improving results for both tree and graph encoders, especially for the latter.

6 Conclusions

We introduced models for AMR-to-text generation with the purpose of investigating the difference between sequential, tree and graph encoders. We showed that encoding reentrancies improves overall performance. We observed bigger benefits when the input AMR graphs have a larger number of reentrant structures and longer dependencies. Our best graph encoder, which consists of a GCN wired to a BiLSTM network, improves over the state of the art on all tested datasets. We inspected the differences between the models, especially in terms of co-references and control structures. Further exploration of graph encoders is left to future work, which may result crucial to improve performance further.

Acknowledgments

The authors would like to thank the three anonymous reviewers and Adam Lopez, Ioannis Konstas, Diego Marcheggiani, Sorcha Gilroy, Sameer Bansal, Ida Szubert and Clara Vania for their help and comments. This research was supported by a grant from Bloomberg and by the H2020 project SUMMA, under grant agreement 688139.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Linguistic Annotation Workshop*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of EMNLP*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *Proceedings of ACL*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Alberto Cetoli, Stefano Bragaglia, Andrew O'Harney, and Marc Sloan. 2017. Graph convolutional networks for named entity recognition. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of ACL*.

Marco Damonte, Shay B Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. *Proceedings of EACL*.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NIPS*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of ACL*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. *Proceedings of ACL*.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems*. University of Calcutta.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. *Proceedings of NAACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL*.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. *Proceedings of INLG*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*.

Rik van Noord and Johan Bos. 2017. Dealing with coreference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS Workshop*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of EACL*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of ACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of EMNLP*.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Multilingual Prediction of Alzheimer's Disease Through Domain Adaptation and Concept-Based Language Modelling

Kathleen C. Fraser[1], **Nicklas Linz**[2], **Bai Li**[3], **Kristina Lundholm Fors**[4], **Frank Rudzicz**[3,5],
Alexandra König[6], **Jan Alexandersson**[2], **Philippe Robert**[6], **Dimitrios Kokkinakis**[4]
[1]National Research Council Canada, Ottawa, Canada
[2]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
[3]University of Toronto and Vector Institute, Toronto, Canada
[4]University of Gothenburg, Gothenburg, Sweden
[5]St. Michael's Hospital, Toronto, Canada
[6]University of Côte d'Azur, Nice University Hospital, and INRIA, Nice, France
`kathleen.fraser@nrc-cnrc.gc.ca`, `nicklas.linz@dfki.de`,
`bai@cs.toronto.edu`, `kristina.lundholmfors@gu.se`,
`frank@cs.toronto.edu`, `alexandra.konig@inria.fr`,
`jan.alexandersson@dfki.de`, `probert@unice.fr`,
`dimitrios.kokkinakis@gu.se`

Abstract

There is growing evidence that changes in speech and language may be early markers of dementia, but much of the previous NLP work in this area has been limited by the size of the available datasets. Here, we compare several methods of domain adaptation to augment a small French dataset of picture descriptions ($n = 57$) with a much larger English dataset ($n = 550$), for the task of automatically distinguishing participants with dementia from controls. The first challenge is to identify a set of features that transfer across languages; in addition to previously used features based on *information units*, we introduce a new set of features to model the order in which information units are produced by dementia patients and controls. These concept-based language model features improve classification performance in both English and French separately, and the best result (AUC = 0.89) is achieved using the multilingual training set with a combination of information and language model features.

1 Introduction

According to the World Health Organisation, the largest global challenge facing the world today is the rapid increase of the population aged over 65 years. It is projected to increase from 524 million in 2010 to 1.5 billion in 2050, with the largest increase in the developing world (Suzman and Beard, 2011). This demographic trend has profound societal implications; for example, the number of persons affected by dementia will increase worldwide from 46 million in 2015 to 131.5 million in 2050 (Prince et al., 2015). The most common underlying condition causing dementia is Alzheimer's disease (AD). Although no cure to this neurodegenerative disease has been found, experts agree that intervention in early stages is crucial to delay onset (Dubois et al., 2016).

AD is characterised by a global impairment of cognitive functioning, with specific deficits in episodic memory, executive functioning, perceptual speed and language (Bäckman et al., 2005; Weiner et al., 2008).

Machine learning experiments using speech and language for the detection of dementia or related disorders have been conducted in many languages, including English (Roark et al., 2011; Mirheidari et al., 2016; Fraser et al., 2016; Asgari et al., 2017), French (Tröger et al., 2017; König et al., 2018), German (Weiner et al., 2016), Hungarian (Szatloczki et al., 2015; Vincze et al., 2016), Spanish (Meilán et al., 2014), Greek (Satt et al., 2013), Swedish (Lundholm Fors et al., 2018; Fraser et al., 2018a), Japanese (Shibata et al., 2016), Portuguese (Aluísio et al., 2016), and Mandarin Chinese (Lai et al., 2009). Most studies acknowledge that small data sets are a limitation and describe the difficulties in gathering more data, including the challenges in patient recruitment, the expense of running clinically-based studies, and the manual effort required for transcription and annotation.

Here, we consider whether it could be possible to increase the amount of available data by augmenting a corpus in one language with data from another language, and thus improve predictive per-

Proceedings of NAACL-HLT 2019, pages 3659–3670
Minneapolis, Minnesota, June 2 - June 7, 2019.

formance without the need for new data collection. Specifically, we consider augmenting a relatively small French dataset with a much larger English one. The two aims of this study are: (1) to identify a set of features that are both useful for the detection of dementia and that we expect to transfer across different languages, and (2) to improve classification results on the French dataset by augmenting the training set with English data.

2 Background and Related Work

2.1 Narrative Analysis in AD

One way to assess language is through narrative speech, such as that elicited by the *Cookie Theft Picture* (CTP) task (Goodglass et al., 2000). In this task, participants are asked to describe the content of a line drawing of a kitchen scene, where a boy can be seen standing on a stool, trying to reach a cookie jar, while a woman is preoccupied washing dishes. In this study, we analyse CTP narratives, due to the widespread use of the task in multiple languages.

Narrative speech can be analysed on a number of levels, including phonology, morphology, syntax, semantics, and pragmatics. Here, our goal is to extract features that both predict AD and are likely to transfer across different languages. Although other studies have used acoustic features for this task (Meilán et al., 2014; König et al., 2018), there are well-documented differences in the phonology and prosody of French and English (Bertrán, 1999; Vaissière, 2002). Syntax and morphology also differ across languages, and the degree to which they are impaired in mild to moderate AD is unclear (Taler and Phillips, 2008). Pragmatic ability in AD may be disrupted (Chapman et al., 1998; Boschi et al., 2017); however, the CTP is not ideally suited for assessing pragmatics.

Instead, we focus on the semantic level, with the assumption that while the specific vocabulary will be different across languages, the underlying meanings or semantic concepts expressed should be the same. Features relating to semantic content are also motivated by the AD literature. Cuetos et al. (2007) reported a significant reduction in semantic units produced by pre-clinical AD participants, relative to controls, on the CTP task. Croisile et al. (1996) studied CTP descriptions from French participants, and found that the AD descriptions were shorter and less informative than the control descriptions. They measured information content by scoring the narratives against a gold standard list of 23 expected "information units", which have been widely used in subsequent research.

2.2 NLP for AD Classification

Several recent studies have used NLP and machine learning to analyse speech samples from people with dementia and other cognitive disorders. Most relevant here, are those which focus on picture description tasks in English or French.

DementiaBank[1] is a large database of CTP narratives from AD patients and controls, containing primarily English data. A number of recent papers report classification results on this corpus (Prud'hommeaux and Roark, 2015; Fraser et al., 2016; Al-Hameed et al., 2016; Yancheva and Rudzicz, 2016; Sirts et al., 2017). Language analysis of English-language CTP data from other sources has also been used to differentiate between different underlying pathologies in AD (Rentoumi et al., 2014), and variants of frontotemporal lobar degeneration (Pakhomov et al., 2010).

In French, picture description was one of multiple tasks used to elicit speech for the classification of participants with mild cognitive impairment and AD reported by König et al. (2015) and König et al. (2018), although only acoustic processing was used.

2.3 Multi- and Cross-Lingual NLP

There has been very little prior work on multilingual or cross-lingual dementia classification. Rentoumi et al. (2018) presented preliminary results suggesting that some language features from CTP samples could transfer across Greek and English, but did not report classification results. Fraser et al. (2018b) studied a related task of detecting mild cognitive impairment (MCI), and found that classification results could be improved in both English and Swedish by incorporating multilingual topic modelling into the feature extraction pipeline; however, they did not consider multilingual classification directly.

More generally, multilingual NLP is an active and growing area of research. Some approaches to improving classifier performance on a resource-poor *target* language by leveraging a resource-rich *source* language include: translate the target language to the source language (or vice versa) and

---

[1] `https://dementia.talkbank.org/`

train a unilingual classifier (Wan, 2009); extract features from the two languages separately and then use domain adaptation techniques to train a classifier for the target language (Blitzer et al., 2006; Prettenhofer and Stein, 2010); or determine a common representation for both languages and then extract features from the combined corpus to train a multilingual classifier (Ammar et al., 2016). In the extreme case, one can also consider purely cross-lingual classification, in which the classifier is trained solely on the source language, but tested on the target language.

We use a supervised domain adaptation approach, similar to that of Daumé III (2007), by considering each language to be a different domain. In related (though not multilingual) work, Masrani et al. (2017) also used this approach to adapt a dataset of AD narratives to their MCI classification task.

2.4 Class-Based Language Modelling

In contrast to the previous work on AD classification, we measure not only which information units are mentioned, but also the order in which they are mentioned. Our approach has some similarity to class-based language models (Brown et al., 1992), in which words are first grouped into classes (or clusters), and then the language model is trained on the classes rather than the individual words. One benefit to this approach is improved generalisability (Hoidekr et al., 2006), and another is the ability of classes to span different languages (Täckström et al., 2012).

3 Methodology

3.1 Data

Data were taken from two corpora: a small French dataset ($n = 57$), collected at the Memory Clinic and Research Centre of the University Hospital Nice, and the Pitt subcorpus of DementiaBank, containing 550 English samples[2]. Detailed information about the protocols for each study can be found in Tröger et al. (2017) and Becker et al. (1994). In both cases, ethics approval for the data collection was obtained from the local governing bodies.

The demographics for the participants in each language are shown in Table 1. In both studies, the participants were asked to perform the CTP task in their respective languages. In English, the image was shown on paper and speech was digitally recorded, while in the French study, the image was displayed on a tablet and speech was recorded via the tablet microphone.

	English		French	
	HC	AD	HC	AD
N	241	309	25	33
Gender	154F/87M	189F/120M	19F/6M	22F/11M
Age	64.8 (7.7)	71.4 (8.4)	75.4 (7.0)	79.2 (6.6)
Education	14.2 (2.6)	12.8 (3.0)	14.0 (2.6)	11.3 (4.0)
MMSE (/30)	29.1 (1.1)	19.8 (5.7)	28.6 (1.4)	18.9 (3.9)

Table 1: Demographics of participants, where AD indicates Alzheimer's disease, and HC indicates healthy control. The Mini Mental State Examination (MMSE) is global measure of cognitive status.

3.2 Features

The English and French audio samples were manually transcribed using the CHAT protocol (MacWhinney, 2014). A set of pre-defined information units found in the CTP was determined as an extension to Croisile et al. (1996), and is given in Table 2a. Mentions of information units were determined using keyword-spotting (based on manually-constructed word lists specific to each language), and used to translate the full narratives to sequences of information units. As an example, the English *A boy is standing on a stool* and French *Le garçon est sur un tabouret* would both be mapped to the sequence BOY STOOL.

Features relating to the occurrence of each distinct information unit comprise the *info* feature set, described in Table 2b. Additionally, new features are derived from language models build on the sequence of information units. To this end, concept-based language models are trained for English and French in a leave-one-out fashion, using the `kenlm` framework (Heafield, 2011). Models up to 5-grams were constructed. For each participant, two language models are constructed for each n: one trained on the healthy control (HC) population and one trained on the AD population. The participant is left out of the model built on their associated diagnostic group. The trained language models are then applied to the held-out participant's sequence of information units and various language model (*LM*) features are extracted (Table 2c).

---

[2] In this analysis, we included all participants in the Dementia subfolder, regardless of specific diagnosis, to maximize the size of the source data.

Actions STEAL, FALL, WASH, OVERFLOW, GIRL'S ACTION, WOMAN'S INDIFFERENCE
Actors BOY, GIRL, CHILD(REN), WOMAN
Places KITCHEN, EXTERIOR
Objects COOKIE, JAR, STOOL, SINK, DISHCLOTH, WATER, WINDOW, CUPBOARD, DISH, CURTAIN, COUNTER

(a) Information units.

has_*unit* Binary feature indicating presence or absence of each information unit (23 features)
ratio_*unit* For each information unit, the number of times that unit was mentioned, divided by the total number of words in the original narrative (23 features)
unique_concept_density Total number of information units which were mentioned at least once, divided by the total number of words in the original narrative (1 feature)
unique_concept_efficiency Total number of information units which were mentioned at least once, divided by the duration of the sample in seconds (1 feature)
total_concept_density Total number of words referring to information units, divided by the total number of words in the original narrative (1 feature)
total_concept_efficiency Total number of words referring to information units, divided by the duration of the sample in seconds (1 feature)

(b) *info* features

perplexity_*class*_*n*-gram The perplexity assigned to the sample by each of the eight language models, where $n = 2, 3, 4, 5$, and the models are trained on data from either the AD or HC class. (8 features)
score_*class*_*n*-gram The log probability assigned to the sample by each of the eight language models. (8 features)
max_perplexity_*class*_*n*-gram The maximum perplexity, computed over all *n*-grams in a sample, for each of the eight language models. (8 features)
min_score_*class*_*n*-gram The minimum log probability, computed over all *n*-grams in a sample, for each of the eight language models. (8 features)

(c) *LM* features

Table 2: Top, the information units extracted from CTP narratives. Bottom, the *info* and *LM* features that are computed from the resulting sequence of information units.

3.3 Unilingual Classification

To evaluate the performance of the three proposed feature sets (*info*, *LM*, and *info*+*LM*), we first train classifiers to distinguish between HC and AD participants within a given language. To examine the importance of certain features, we restrict ourselves to more explainable linear models, namely logistic regression (LR) and linear support vector machines (SVM) (Pedregosa et al., 2011). In both cases, we use L_1 regularisation to promote sparsity in the feature weights.

Area under the Receiver-Operator curve (AUC) is reported as the evaluation parameter. Due to the small size of the French dataset, we use leave-pair-out cross validation (LPO-CV), which has been shown to produce an unbiased estimate for AUC on small datasets (Airola et al., 2009), and has also been used in related work (Roark et al., 2011). However, since LPO-CV is computationally very costly, we instead use 10-fold cross-validation (10-CV) for English, making sure that any samples for a given participant occur in either the training set or the test set, but not both. For LPO-CV we compute AUC and its standard deviation as described by Roark et al. (2011); for 10-CV we compute the AUC in each test fold and then report the average and standard deviation over folds.

Feature scaling and hyper-parameter optimisation is done on the training set in each fold. Features are scaled using Maximum-Absolute Scaling to preserve the binary nature of the *info* features. For both SVMs and LR, C was optimised between $C \in [10^{-4}, ..., 10^{4}]$ using a grid search.

3.4 Multilingual Classification

Our goal is to improve classification in French, by incorporating training data from English. To this end, we examine multiple ways to combine data from both English and French in the training set.

We first consider *domain adaptation*, where we treat French as the target domain and English as the source domain. We implement the AUGMENT method of Daumé III (2007), which involves augmenting the feature space with source-specific, target-specific, and combined versions of all the original features, allowing the classifier to assign a higher weight to the combined version when that feature transfers well across domains, while also retaining source- and target-specific information where appropriate.

We consider as well as the baseline methods described in Daumé III (2007): WEIGHT, in which the samples from the source domain are assigned reduced weights in the model; PRED, in which the prediction made by the source classifier is used as an additional feature in the target model; LININT, in which the predictions from the source and target models are linearly interpolated; and ALL, in which target and source data are simply combined in a single training set. Due to the limited size of our data, we do not optimise the weighting factors in WEIGHT and LININT, but rather assume the two languages should be given equal importance, and use a weighting factor of 0.1 in WEIGHT (since the English data is 10 times the size of the French data), and 0.5 in LININT.

Another option is to combine the French

and English datasets before extracting features. Specifically, we first replace the word-level transcripts with the sequence of information units, and then combine the two datasets and train the language models over the multilingual corpus, thus generating *multilingual language models*.

3.5 Cross-Lingual Classification

To understand how well a trained classification model in one language could be applied to another, we also perform cross-lingual experiments. For this, we train language and classification models in one language and test it on the other.

4 Results

The results of the classification experiments are presented in Figure 1.

4.1 Unilingual Classification

In French, for both LR and SVM, using *LM* features leads to higher AUC than the *info* features, and the combination of features is more effective than either feature set alone. In the English case, the *LM* and *info* features lead to equivalent performance individually, but the AUC is again marginally improved when the feature sets are combined, suggesting that they are capturing at least somewhat complementary information.

4.2 Domain Adaptation Results

For French, the *LM* features generally do not benefit from domain adaptation, with equivalent or poorer AUC relative to the unilingual case. The best result with the *LM* features is achieved in the AUGMENT scenario, where the classifier can select the French *LM* features only (although this result holds only for the SVM classifier). In contrast, the *info* features do benefit from the additional data available through domain adaptation, and lead to better results than the unilingual baseline. The best overall result of AUC = 0.89 is achieved by combining the feature types in the ALL configuration.

For English, we do not expect to see much benefit from including the (much smaller) French dataset. The WEIGHT adaptation technique is not feasible when the source data is smaller than the target data, and the LININT technique performs poorly, as it assigns too much importance the smaller and out-of-domain dataset. However, we do see marginal improvements using ALL and AUGMENT, reflecting the value of increasing the training set size by roughly 10%. The best result of AUC = 0.84 is achieved in the ALL condition, using the combined feature set.

4.3 Multilingual LM Results

Using the multilingual LM does not affect the *info* features, and therefore Figure 1 shows only the *LM* and *info+LM* results. Clearly, the multilingual LM approach does not work well here. Unlike in domain adaptation, combining the datasets using this method assumes that information units will be produced in the same order in the two languages. While French and English are similar in this respect, there are many possible counter-examples, such as *cookie jar* (COOKIE JAR) versus *boîte à biscuits* (JAR COOKIE).

4.4 Cross-Lingual Classification

When training entirely on English data and testing on French, the results using *info* and *info+LM* features are significantly improved over the unilingual baseline, while the *LM* results are reduced, once again indicating that the *info* features transfer better across languages. The results are very similar to those using the ALL technique for domain adaptation, suggesting that in that case, model training is dominated by the English data.

To further explore the similarity in performance in the ALL and cross-lingual cases, we examine the effect of incrementally increasing the amount of English data in the training set, when testing on French data. Figure 2 displays the classification performance of SVM and LR classifiers trained either using the ALL method of domain adaptation or cross-lingually with increasing amounts (10% at a time) of the English data. Considering first the ALL method (red and blue), at $x = 0$ there is no English data, and so we recover the French unilingual baseline. As we increase the amount of English data in the training set, performance slowly increases, eventually reaching the values reported in Figure 1. Considering next the cross-lingual case (yellow and green), we see that training on only 10% of the English data (55 samples) results in much poorer AUC values. However, each further 10% increases the classification performance. At 80% of English data (440 samples) the multi- and cross-lingual cases converge in performance. Thus, it would appear that domain adaptation is more data-efficient, as we achieve close to optimal results with a smaller proportion of English data,

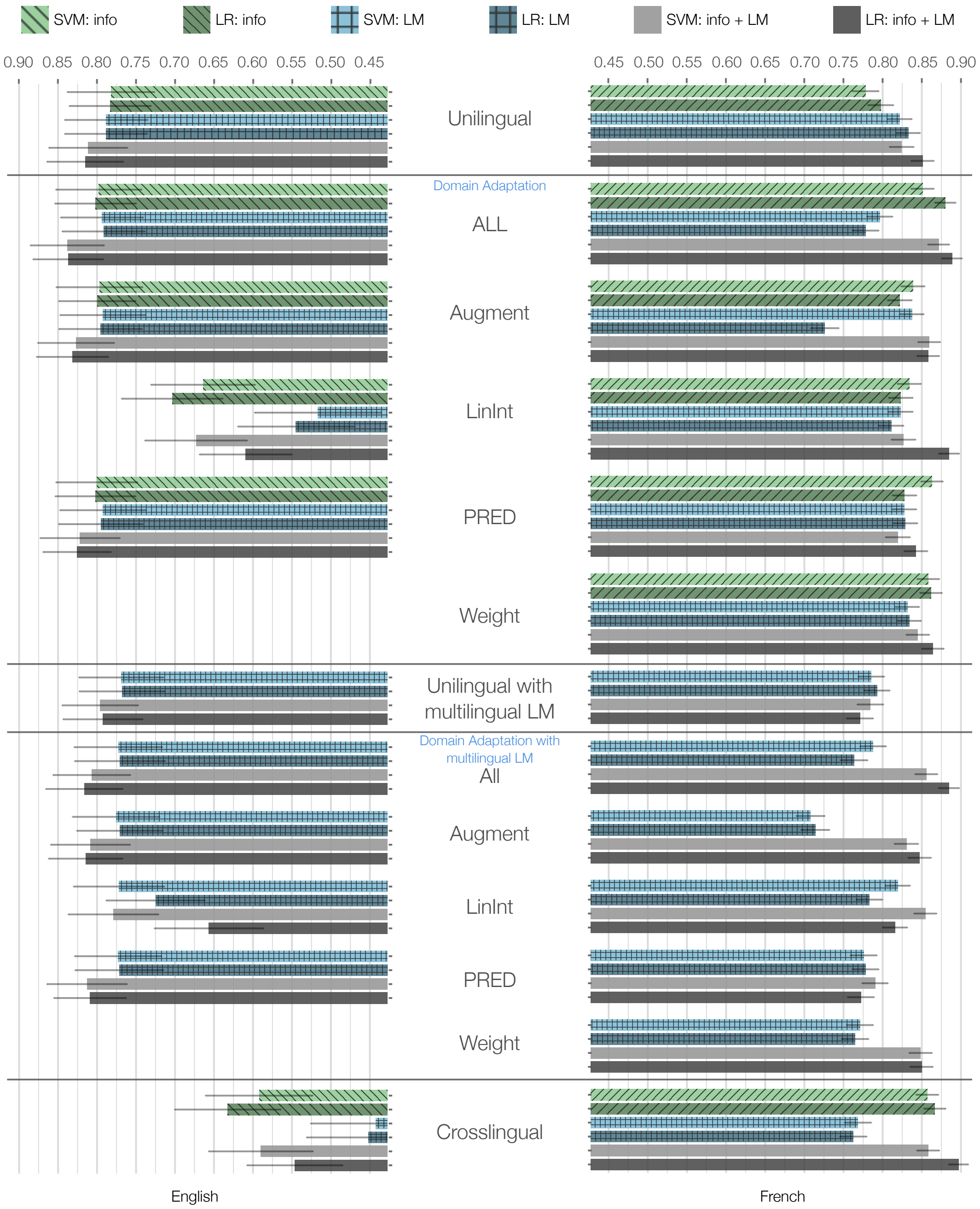


Figure 1: Results of uni-, multi- and cross-lingual classification experiments. Left panel displays results for English, right panel for French. Labels in the middle indicate the classification scenario and method of domain adaptation. Colours indicate the feature set and classifier. Bars indicate the AUC; error bars represent standard deviation.

but that the cross-lingual approach can be equally effective, given a large enough corpus.

4.5 Feature Analysis

Finally, we examine the features to determine which features are most useful to the task of dementia detection, and to compare the selected features in the unilingual and multilingual cases. Figure 3 shows the median absolute value of the weights assigned to each feature, for English and French, in the unilingual and multilingual ALL condition. The L_1 regularisation serves to set many feature weights to zero.

As a high-level observation, in both the uni- and multilingual cases, relatively more *info* features are selected, and relatively fewer *LM* features. Of the *LM* features that are selected, those which relate to the maximum perplexity or minimum probability appear to be more useful. These features

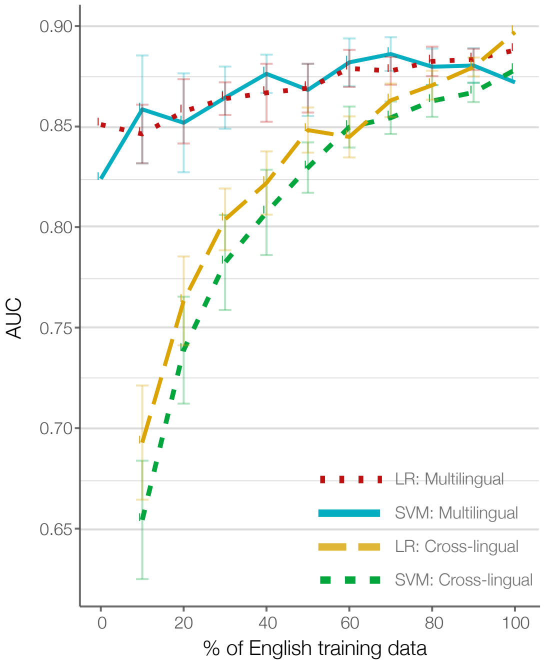


Figure 2: *AUC* as a function of the amount of English data used in the training set, for both multi- and cross-lingual cases. Error bars indicate 95% confidence intervals.

capture locally anomalous speech patterns, relative to either the AD or control language models.

In the unilingual case, the French models show a preference for the binary "has" features (indicating whether or not an information unit has been mentioned). Only 4 of the "ratio" features and none of the density or efficiency features have a median value greater than zero. However, these features *are* relevant to the task, and potentially more generalisable (e.g., total concept efficiency differs between the French AD and HC groups with $p < 0.001$ on a t-test, and represents an aggregate score rather than depending on the presence or absence of a single information unit). Such features are selected more often in the multilingual case, and lead to improved performance. One explanation for this could be that in the small French training set, spurious correlations due to noise can overpower the real signal, and lead to less relevant features being assigned high weights, while correlated (but perhaps actually more relevant) features are suppressed. By increasing the size of the training set with English data, the signal-to-noise ratio is improved, and a better set of features is selected.

Generally, the feature values (not shown) support the intuition that controls mention more of the information units in the image (higher "has" feature values), convey information more efficiently, with fewer off-topic words (higher density and efficiency scores), and organize the narrative in a more predictable way (narratives have lower perplexity and higher probability) than the AD participants. Again, these trends are more apparent in the English data than the French data, likely due to the relatively larger number of samples.

5 Discussion

One perhaps surprising result of this study was that naively combining features in the ALL condition led to better results than the AUGMENT algorithm. However, this is in line with the original findings of Daumé III (2007), where he identified a set of tasks where AUGMENT performed sub-optimally: specifically, those cases where training on source-only data was better than training on target-only data. This is precisely the case we have here, as training cross-lingually (on English source data) leads to better results than training unilingually (on French target data). The explanation offered by Daumé III is, "If the domains are so similar that a large amount of source data outperforms a small amount of target data, then it is unlikely that blowing up the feature space will help." In some sense, then, these results are confirmation that we have indeed identified a set of features over which the two languages (i.e. domains) are very similar.

The fact that the ALL configuration is optimal in both French and English has an added practical benefit: since there is no distinction between source and target features, the resulting classifier is language-agnostic. This means that test data could come from either language, in a hypothesized future screening application.

While our goal in this paper was not to push the state-of-the-art on the DementiaBank dataset, we do find that our best English result (AUC=0.84, which corresponds to an accuracy of 75% and F_1 score of 0.77) is comparable to the other published results on this dataset (Prud'hommeaux and Roark, 2015; Yancheva and Rudzicz, 2016; Sirts et al., 2017; Fraser et al., 2016; Hernández-Domínguez et al., 2018). There are no previously published results on the French dataset.

6 Conclusion and Future Work

In this work, we have shown that there are features which can both distinguish AD patients from healthy controls with a high degree of accuracy,

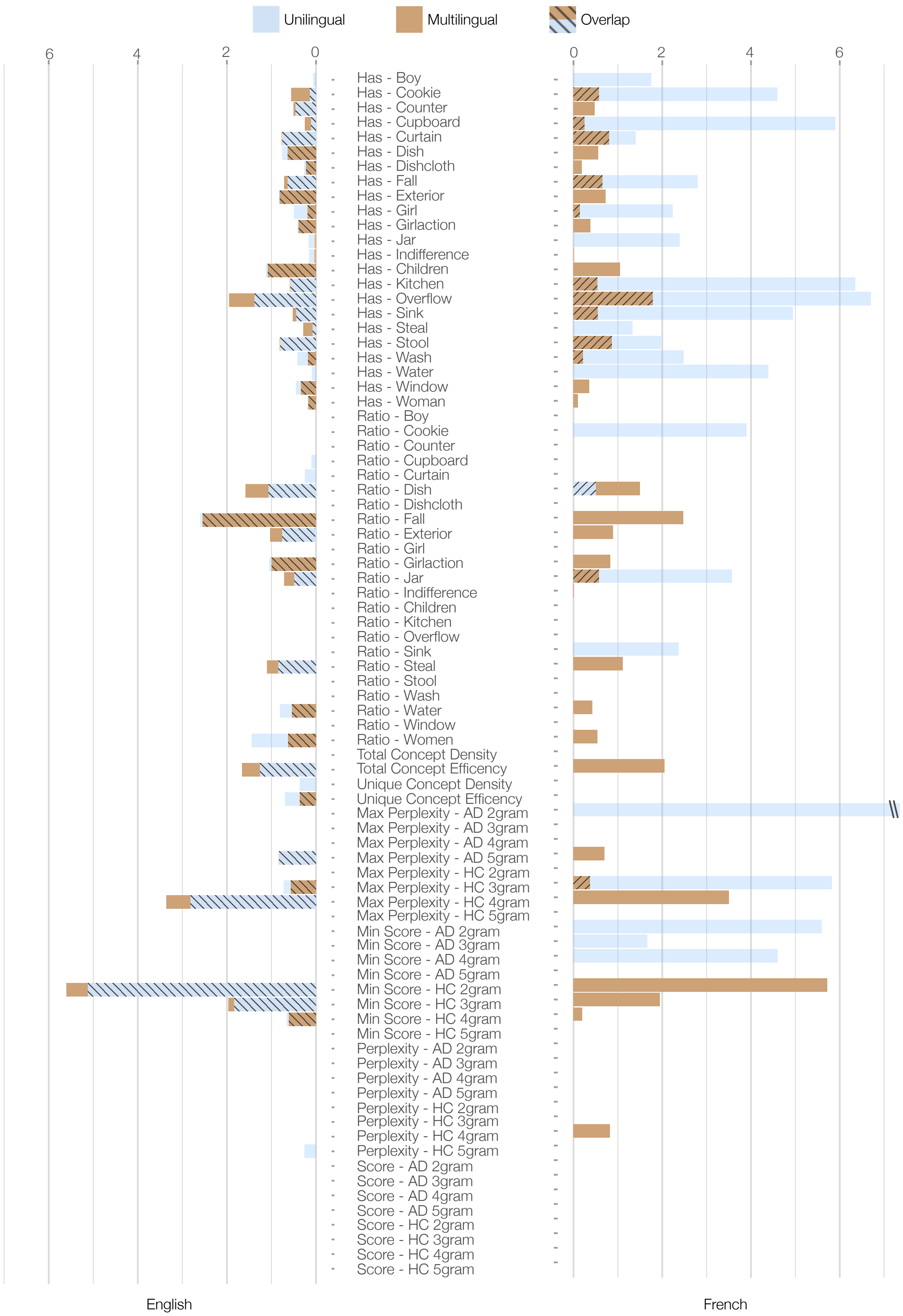


Figure 3: Visualisation of feature weights for uni- and multilingual experiments. Median feature importances over LPO- and 10-CV are displayed. The left panel displays the English and the right panel the French data sets. Unilingual experiments are given in blue and multilingual in yellow.

and also generalize across languages. By incorporating a large English dataset, we were able to improve the AUC on the French dataset from 0.85 to 0.89. We also developed a new set of features for this task, using concept-based language modelling, which improved AUC from 0.80 to 0.85 in the unilingual case, and 0.88 to 0.89 in the multilingual case.

Future work will involve extending the set of features involved, incorporating data from other languages, and testing whether similar techniques can be effective for detecting earlier stages of cognitive decline, such as MCI. Other work from our group has also begun to explore the use of unsupervised methods and out-of-domain data sources (Li et al., 2019).

Technical challenges aside, collaborations of this nature can be difficult due to the sensitive nature of the data, and the need to respect ethical guidelines and participant consent when sharing and storing data. With this in mind, we recommend to other researchers working in similar domains to consider from the outset whether their data could eventually be shared, and to make suitable provisions in their ethics protocols and participant consent forms. We look to DementiaBank as a model for this kind of data-sharing and openness, and hope that researchers can continue to find ways to share resources of this nature.

Acknowledgements

This research was partially funded by the Riksbankens Jubileumsfond – The Swedish Foundation for Humanities and Social Sciences, grant no: NHS 14-1761:1, and the EIT Digital Wellbeing Activity 17074, ELEMENT. The French data was collected during the ELEMENT project and the FP7 Dem@Care project (grant number 288199). The original acquisition of the DementiaBank data was supported by NIH grants AG005133 and AG003705 to the University of Pittsburgh, and the data archive is supported by NIHNIDCD grant R01-DC008524 to Carnegie Mellon University.

References

Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. 2009. A comparison of AUC estimators in small-sample studies. In *Machine Learning in Systems Biology*, pages 3–13.

Sabah Al-Hameed, Mohammed Benaissa, and Heidi Christensen. 2016. Simple and robust audio-based detection of biomarkers for Alzheimer's disease. In *7th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, pages 32–36.

Sandra Aluísio, Andre Cunha, and Carolina Scarton. 2016. Evaluating progression of Alzheimer's disease by regression and classification methods in a narrative language test in Portuguese. In *International Conference on Computational Processing of the Portuguese Language*, pages 109–114. Springer.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. 2017. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 3(2):219–228.

Lars Bäckman, Sari Jones, Anna-Karin Berger, Erika Jonsson Laukka, and Brent J Small. 2005. Cognitive impairment in preclinical Alzheimer's disease: A meta-analysis. *Neuropsychology*, 19(4):520–531.

James T. Becker, François Boiler, Oscar L. Lopez, Judith Saxton, and Karen L. McGonigle. 1994. The natural history of Alzheimer's disease: Description of study cohort and accuracy of diagnosis. *Archives of Neurology*, 51(6):585–594.

Antonio Pamie Bertrán. 1999. Prosodic typology: On the dichotomy between stress-timed and syllable-timed languages. *Language Design: Journal of Theoretical and Experimental Linguistics*, 2:103–130.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Veronica Boschi, Eleonora Catricala, Monica Consonni, Cristiano Chesi, Andrea Moro, and Stefano F Cappa. 2017. Connected speech in neurodegenerative language disorders: A review. *Frontiers in psychology*, 8:269.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Sandra Bond Chapman, Amy Peterson Highley, and Jennifer L Thompson. 1998. Discourse in fluent aphasia and Alzheimer's disease: Linguistic and pragmatic considerations. *Journal of Neurolinguistics*, 11(1-2):55–78.

Bernard Croisile, Bernadette Ska, Marie-Josee Brabant, Annick Duchene, Yves Lepage, Gilbert Aimard, and Marc Trillet. 1996. Comparative study of oral and written picture description in patients with Alzheimer's disease. *Brain and Language*, 53(1):1–19.

Fernando Cuetos, Juan Carlos Arango-Lasprilla, Claramónika Uribe, Claudia Valencia, and Francisco Lopera. 2007. Linguistic changes in verbal expression: A preclinical marker of Alzheimer's disease. *Journal of the International Neuropsychological Society*, 13(3):433–439.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 256–263.

Bruno Dubois, Harald Hampel, Howard H Feldman, Philip Scheltens, Paul Aisen, Sandrine Andrieu, Hovagim Bakardjian, Habib Benali, Lars Bertram, Kaj Blennow, et al. 2016. Preclinical Alzheimer's disease: Definition, natural history, and diagnostic criteria. *Alzheimer's & Dementia*, 12(3):292–323.

Kathleen C. Fraser, Kristina Lundholm Fors, Marie Eckström, Charalampos Themistokleous, and Dimitrios Kokkinakis. 2018a. Improving the sensitivity and specificity of MCI screening with linguistic information. In *Proceedings of the 2nd Workshop on Resources and processing of linguistic, paralinguistic and extra-linguistic data from people with various forms of cognitive/psychiatric impairments (RaPID)*, pages 19–26.

Kathleen C Fraser, Kristina Lundholm Fors, and Dimitrios Kokkinakis. 2018b. Multilingual word embeddings for the assessment of narrative speech in mild cognitive impairment. *Computer Speech & Language*, 53:121–139.

Kathleen C. Fraser, Jed A. Meltzer, and Frank Rudzicz. 2016. Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

Harold Goodglass, Edith Kaplan, and Barbara Barresi. 2000. *Boston Diagnostic Aphasia Examination*. Lippincott Williams & Wilkins.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT)*, pages 187–197.

Laura Hernández-Domínguez, Sylvie Ratté, Gerardo Sierra-Martínez, and Andrés Roche-Bergua. 2018. Computer-based evaluation of Alzheimer's disease and mild cognitive impairment patients during a picture description task. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 10:260–268.

Jan Hoidekr, Josef V Psutka, Ales Prazák, and Josef Psutka. 2006. Benefit of a class-based language model for real-time closed-captioning of TV ice-hockey commentaries. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 2064–2067.

Alexandra König, Aharon Satt, Alex Sorin, Ran Hoory, Alexandre Derreumaux, Renaud David, and Phillippe H Robert. 2018. Use of speech analyses within a mobile application for the assessment of cognitive impairment in elderly people. *Current Alzheimer Research*, 15(2):120–129.

Alexandra König, Aharon Satt, Alexander Sorin, Ron Hoory, Orith Toledo-Ronen, Alexandre Derreumaux, Valeria Manera, Frans Verhey, Pauline Aalten, Phillipe H Robert, and Renaud David. 2015. Automatic speech analysis for the assessment of patients with predementia and Alzheimer's disease. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 1(1):112–124.

Yi-hsiu Lai, Hsiu-hua Pai, et al. 2009. To be semantically-impaired or to be syntactically-impaired: Linguistic patterns in Chinese-speaking persons with or without dementia. *Journal of Neurolinguistics*, 22(5):465–475.

Bai Li, Yi-Te Hsu, and Frank Rudzicz. 2019. Detecting dementia in Mandarin Chinese using transfer learning from a parallel corpus. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Kristina Lundholm Fors, Kathleen C. Fraser, and Dimitrios Kokkinakis. 2018. Automated syntactic analysis of language abilities in persons with mild and subjective cognitive impairment. In *Proceedings of the Medical Informatics Europe (MIE) Conference*, pages 705–709.

Brian MacWhinney. 2014. *The CHILDES project: Tools for analyzing talk, Volume I: Transcription format and programs*. Psychology Press.

Vaden Masrani, Gabriel Murray, Thalia Shoshana Field, and Giuseppe Carenini. 2017. Domain adaptation for detecting mild cognitive impairment. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 248–259.

Juan José G Meilán, Francisco Martínez-Sánchez, Juan Carro, Dolores E López, Lymarie Millian-Morell, and José M Arana. 2014. Speech in Alzheimer's disease: Can temporal and acoustic parameters discriminate dementia? *Dementia and Geriatric Cognitive Disorders*, 37(5-6):327–334.

Bahman Mirheidari, Daniel Blackburn, Markus Reuber, Traci Walker, and Heidi Christensen. 2016. Diagnosing people with dementia using automatic conversation analysis. In *Proceedings of Interspeech*, pages 1220–1224. ISCA.

Serguei VS Pakhomov, Glenn E Smith, Dustin Chacon, Yara Feliciano, Neill Graff-Radford, Richard Caselli, and David S Knopman. 2010. Computerized analysis of speech and language to identify psycholinguistic correlates of frontotemporal lobar degeneration. *Cognitive and Behavioral Neurology*, 23(3):165.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the Association for Computational Linguistics (ACL)*, pages 1118–1127.

Martin Prince, Anders Wimo, Maelenn Guerchet, Gemma-Claire Ali, Yu-Tzu Wu, and Matthew Prina. 2015. World Alzheimer report 2015: The global impact of dementia. *Alzheimer's Disease International (ADI)*.

Emily Prud'hommeaux and Brian Roark. 2015. Graph-based word alignment for clinical language evaluation. *Computational Linguistics*, 41(4):549–578.

Vassiliki Rentoumi, George Paliouras, Dimitra Arfani, Katerina Fragkopoulou, Spyridoula Varlokosta, and Peter Garrard. 2018. Automatic detection of linguistic indicators as a means of early prediction of Alzheimer's and of related dementias: A cross-linguistics analysis. *Alzheimer's & Dementia: The Journal of the Alzheimer's Association*, 14:1302.

Vassiliki Rentoumi, Ladan Raoufian, Samrah Ahmed, Celeste A. de Jager, and Peter Garrard. 2014. Features and machine learning classification of connected speech samples from patients with autopsy proven Alzheimer's disease with and without additional vascular pathology. *Journal of Alzheimer's Disease*, 42(S3):S3–S17.

Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffery Kaye. 2011. Spoken language derived measures for detecting mild cognitive impairment. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2081–2090.

Aharon Satt, Alexander Sorin, Orith Toledo-Ronen, Oren Barkan, Ioannis Kompatsiaris, Athina Kokonozi, and Magda Tsolaki. 2013. Evaluation of speech-based protocol for detection of early-stage dementia. In *Proceedings of Interspeech*, pages 1692–1696.

Daisaku Shibata, Shoko Wakamiya, Ayae Kinoshita, and Eiji Aramaki. 2016. Detecting Japanese patients with Alzheimer's disease based on word category frequencies. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 78–85.

Kairit Sirts, Olivier Piguet, and Mark Johnson. 2017. Idea density for predicting Alzheimer's disease from transcribed speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 322–332.

Richard Suzman and John Beard. 2011. Global health and aging. *National Institute on Aging and National Institutes of Health, WHO*, 11-7737.

Greta Szatloczki, Ildiko Hoffmann, Veronika Vincze, Janos Kalman, and Magdolna Pakaski. 2015. Speaking in Alzheimer's disease, is that an early sign? Importance of changes in language abilities in Alzheimer's disease. *Frontiers in Aging Neuroscience*, 7:1–7.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 477–487.

Vanessa Taler and Natalie A Phillips. 2008. Language performance in Alzheimer's disease and mild cognitive impairment: A comparative review. *Journal of Clinical and Experimental Neuropsychology*, 30(5):501–556.

Johannes Tröger, Nicklas Linz, Jan Alexandersson, Alexandra König, and Philippe Robert. 2017. Automated speech-based screening for Alzheimer's disease in a care service scenario. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 292–297.

Jacqueline Vaissière. 2002. Cross-linguistic prosodic transcription: French versus English. In N.B. Volskaya, N.D. Svetozarova, and P.A. Skrelin, editors, *Problems and Methods of Experimental Phonetics. In honour of the 70th anniversary of Pr. L.V. Bondarko*, pages 147–164. St Petersburg State University Press.

Veronika Vincze, Gábor Gosztolya, László Tóth, Ildikó Hoffmann, Gréta Szatlóczki, Zoltán Bánréti, Magdolna Pákáski, and János Kálmán. 2016. Detecting mild cognitive impairment by exploiting linguistic information from transcripts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 181–187.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 235–243.

Jochen Weiner, Christian Herff, and Tanja Schultz. 2016. Speech-based detection of Alzheimer's disease in conversational German. In *Proceedings of Interspeech*, pages 1938–1942.

Myron F Weiner, Katherine E Neubecker, Mary E Bret, and Linda S Hynan. 2008. Language in Alzheimer's disease. *The Journal of Clinical Psychiatry*, 69(8):1223.

Maria Yancheva and Frank Rudzicz. 2016. Vector-space topic models for detecting Alzheimer's disease. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2337–2346.

Ranking and Selecting Multi-Hop Knowledge Paths to Better Predict Human Needs

Debjit Paul
Research Training Group AIPHES
Institute for Computational Linguistics
Heidelberg University
paul@cl.uni-heidelberg.de

Anette Frank
Research Training Group AIPHES
Institute for Computational Linguistics
Heidelberg University
frank@cl.uni-heidelberg.de

Abstract

To make machines better understand sentiments, research needs to move from polarity identification to understanding the reasons that underlie the expression of sentiment. Categorizing the goals or needs of humans is one way to explain the expression of sentiment in text. Humans are good at understanding situations described in natural language and can easily connect them to the character's psychological needs using commonsense knowledge. We present a novel method to extract, rank, filter and select multi-hop relation paths from a commonsense knowledge resource to interpret the expression of sentiment in terms of their underlying human needs. We efficiently integrate the acquired knowledge paths in a neural model that interfaces context representations with knowledge using a gated attention mechanism. We assess the model's performance on a recently published dataset for categorizing human needs. Selectively integrating knowledge paths boosts performance and establishes a new state-of-the-art. Our model offers interpretability through the learned attention map over commonsense knowledge paths. Human evaluation highlights the relevance of the encoded knowledge.

1 Introduction

Sentiment analysis and emotion detection are essential tasks in human-computer interaction. Due to its broad practical applications, there has been rapid growth in the field of sentiment analysis (Zhang et al., 2018). Although state-of-the-art sentiment analysis can detect the polarity of text units (Hamilton et al., 2016; Socher et al., 2013), there has been limited work towards explaining the reasons for the expression of sentiment and emotions in texts (Li and Hovy, 2017). In our work, we aim to go beyond the detection of sentiment, toward explaining sentiments. Such explanations can range from detecting overtly expressed explanations or reasons for sentiments towards specific aspects of, e.g., products or films, as in user reviews to the explanation of the underlying reasons for emotional reactions of characters in a narrative story. The latter requires understanding of stories and modeling the mental state of characters. Recently, Ding and Riloff (2018) proposed to categorize affective events with categories based on human needs, to provide explanations of people's attitudes towards such events. Given an expression such as *I broke my leg*, they categorize the reason for the expressed negative sentiment as being related to a need concerning 'health'.

In this paper we focus on the *Modelling Naive Psychology of Characters in Simple Commonsense Stories* dataset of Rashkin et al. (2018), which contains annotations of a fully-specified chain of motivations and emotional reactions of characters for a collection of narrative stories. The stories are annotated with labels from multiple theories of psychology (Reiss, 2004; Maslow, 1943; Plutchik, 1980) to provide explanations for the emotional reactions of characters.

Similar to Ding and Riloff (2018), we hypothesize that emotional reactions (joy, trust, fear, etc.) of characters can be explained by (dis)satisfaction of their psychological needs. However, predicting categories of human needs that underlie the expression of sentiment is a difficult task for a computational model. It requires not only detecting surface patterns from the text, but also requires commonsense knowledge about how a given situation may or may not satisfy specific human needs of a character. Such knowledge can be diverse and complex, and will typically be implicit in the text. In contrast, human readers can make use of relevant information from the story and associate it with their knowledge about human interaction, desires and human needs, and thus will be able

Proceedings of NAACL-HLT 2019, pages 3671–3681
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

to infer underlying reasons for emotions indicated in the text. In this work, we propose a computational model that aims to categorize human needs of story characters by integrating commonsense knowledge from ConceptNet (Speer and Havasi, 2012). Our model aims to imitate human understanding of a story, by (i) learning to select relevant words from the text, (ii) extracting pieces of knowledge from the commonsense inventory and (iii) associating them with human need categories put forth by psychological theories. Our assumption is that by integrating commonsense knowledge in our model we will be able to overcome the lack of textual evidence in establishing relations between expressed emotions in specific situations and the inferable human needs of story characters. In order to provide such missing associations, we leverage the graph structure of the knowledge source. Since these connections can be diverse and complex, we develop a novel approach to extract and rank multi-hop relation paths from ConceptNet using graph-based methods.

Our contributions are: (i) We propose a novel approach to extract and rank multi-hop relation paths from a commonsense knowledge resource using graph-based features and algorithms. (ii) We present an end-to-end model enhanced with attention and a gated knowledge integration component to predict human needs in a given context. To the best of our knowledge, our model is the first to advance commonsense knowledge for this task. (iii) We conduct experiments that demonstrate the effectiveness of the extracted knowledge paths and show significant performance improvements over the prior state-of-the-art. (iv) Our model provides interpretability in two ways: by selecting relevant words from the input text and by choosing relevant knowledge paths from the imported knowledge. In both cases, the degree of relevance is indicated via an attention map. (v) A small-scale human evaluation demonstrates that the extracted multi-hop knowledge paths are indeed relevant. Our code is made publicly available.[1]

2 Related Work

Sentiment Analysis and Beyond. Starting with Pang et al. (2002), sentiment analysis and emotion detection has grown to a wide research field. Researchers have investigated polarity classification, sentiment and emotion detection and classification (Tang et al., 2015; Yin et al., 2017; Li et al., 2017) on various levels (tokens, phrases, sentences or documents), as well as structured prediction tasks such as the identification of holders and targets (Deng and Wiebe, 2015) or sentiment inference (Choi et al., 2016). Our work goes beyond the analysis of overtly expressed sentiment and aims at identifying goals, desires or needs underlying the expression of sentiment. Li and Hovy (2017) argued that the goals of an opinion holder can be categorized by human needs. There has been work related to goals, desires, wish detection (Goldberg et al., 2009; Rahimtoroghi et al., 2017). Most recently, Ding and Riloff (2018) propose to categorize affective events into physiological needs to explain people's motivations and desires. Rashkin et al. (2018) published a dataset for tracking emotional reactions and motivations of characters in stories. In this work, we use this dataset to develop a knowledge-enhanced system that 'explains' sentiment in terms of human needs.

Integrating Structured Knowledge into Neural NLU Systems. Neural models aimed at solving NLU tasks have been shown to profit from the integration of knowledge, using different methods: Xu et al. (2017) show that injecting loosely structured knowledge with a recall-gate mechanism is beneficial for conversation modeling; Mihaylov and Frank (2018) and Weissenborn et al. (2017) propose integration of commonsense knowledge for reading comprehension: the former explicitly encode selected triples from ConceptNet using attention mechanisms, the latter enriches question and context embeddings by encoding triples as mapped statements extracted from ConceptNet. Concurrently to our work, Bauer et al. (2018) proposed a heuristic method to extract multi-hop paths from ConceptNet for a reading comprehension task. They construct paths starting from concepts appearing in the question to concepts appearing in the context, aiming to emulate multi-hop reasoning. Tamilselvam et al. (2017) use ConceptNet relations for aspect-based sentiment analysis. Similar to our approach, Bordes et al. (2014) make use of knowledge bases to obtain longer paths connecting entities appearing in questions to answers in a QA task. They also provide a richer representation of answers by building subgraphs of entities appearing in answers. In contrast, our work aims to provide information about missing links

---

[1] `https://github.com/debjitpaul/Multi-Hop-Knowledge-Paths-Human-Needs`

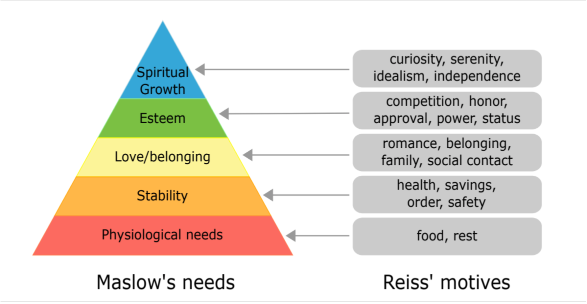


Figure 1: Maslow and Reiss: Theories of Psychology as presented in Rashkin et al. (2018).

between sentiment words in a text and underlying human needs by extracting relevant multi-hop paths from structured knowledge bases.

3 Selecting and Ranking Commonsense Knowledge to Predict Human Needs

Our task is to automatically predict human needs of story characters given a story context. In this task, following the setup of Rashkin et al. (2018), we *explain* the probable reasons for the expression of emotions by predicting appropriate categories from two theories of psychology: *Hierarchy of needs* (Maslow, 1943) and *basic motives* (Reiss, 2002). The task is defined as a multi-label classification problem with five coarse-grained (Maslow) and 19 fine-grained (Reiss) categories, respectively (see Fig. 1).[1] We start with a Bi-LSTM encoder with self-attention as a baseline model, to efficiently categorize human needs. We then show how to select and rank multi-hop commonsense knowledge paths from ConceptNet that connect textual expressions with human need categories. Finally, we extend our model with a gated knowledge integration mechanism to incorporate relevant multi-hop commonsense knowledge paths for predicting human needs. An overview of the model is given in Figure 2. We now describe each component in detail.

3.1 A Bi-LSTM Encoder with Attention to Predict Human Needs

Our Bi-LSTM encoder takes as input a sentence S consisting of a sequence of tokens, denoted as $w_1^s, w_2^s, ..., w_n^s$, or $w_{1:n}^s$ and its preceding context Cxt, denoted as $w_1^{cxt}, w_2^{cxt}, ..., w_m^{cxt}$, or $w_{1:m}^{cxt}$. As further input we read the name of a story character, which is concatenated to the input sentence. For this input the model is tasked to predict appropriate human need category labels $z \in Z$, according to a predefined inventory.

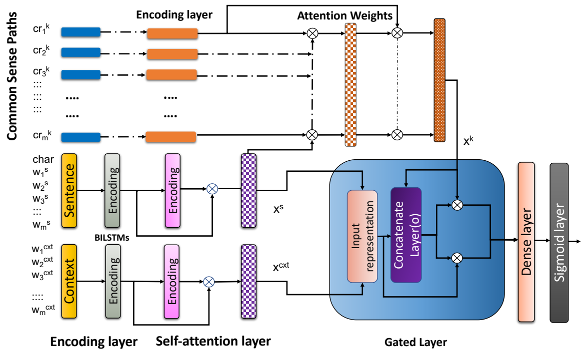


Figure 2: Attention over multi-hop knowledge paths.

Embedding Layer: We embed each word from the sentence and the context with a contextualized word representation using character-based word representations (ELMo) (Peters et al., 2018). The embedding of each word w_i in the sentence and context is represented as e_i^s and e_i^{cxt}, respectively.

Encoding Layer: We use a single-layer Bi-LSTM (Hochreiter and Schmidhuber, 1997) to obtain sentence and context representations h^s and h^{cxt}, which we form by concatenating the final states of the forward and backward encoders.

$$h^s = BiLSTM(e_{1:n}^s); h^{cxt} = BiLSTM(e_{1:m}^{cxt}) \tag{1}$$

A Self-Attention Layer allows the model to dynamically control how much each token contributes to the sentence and context representation. We use a modified version of self-attention proposed by Rei and Søgaard (2018), where both input representations are passed through a feedforward layer to generate scalar values for each word in context v_i^{cxt} and sentence v_i^s (cf. (2-5)).

$$a_i^s = ReLU(W_i^s h_i^s + b_i^s), \tag{2}$$

$$a_i^{cxt} = ReLU(W_i^{cxt} h_i^{cxt} + b_i^{cxt}) \tag{3}$$

$$v_i^s = W_v^s{}_i a_i^s + b_{vi}^s \tag{4}$$

$$v_i^{cxt} = W_v^{cxt}{}_i a_i^{cxt} + b_v^{cxt}{}_i \tag{5}$$

where, $W^s, b^s, W^{cxt}, b^{cxt}, W_v^s, W_v^{cxt}$ are trainable parameters. We calculate the soft attention weights for both sentence and context:

$$\widetilde{v_i} = \frac{1}{1 + exp(-v_i)}; \quad \hat{v}_i = \frac{\widetilde{v_i}}{\sum_{k=1}^{N} \widetilde{v}_k} \tag{6}$$

---

[1] Details about the labels are given in the Supplement.

where, $\widetilde{v}_i$ is the output of the sigmoid function, therefore $\widetilde{v}_i$ is in the range [0,1] and $\hat{v}_i$ is the normalized version of $\widetilde{v}_i$. Values $\hat{v}_i$ are used as attention weights to obtain the final sentence and context representations x^s and x^{cxt}, respectively:

$$x^s = \sum_{i=1}^{N} \hat{v}_i{}^s h_i^s \tag{7}$$

$$x^{cxt} = \sum_{i=1}^{M} \hat{v}_i{}^{cxt} h_i^{cxt} \tag{8}$$

with N and M the number of tokens in S and Cxt. The output of the self-attention layer is generated by concatenating x^s and x^{cxt}. We pass this representation through a FF layer of dimension Z:

$$y = ReLU(W_y[x^s; x^{cxt}] + b_y) \tag{9}$$

where W_y, b_y are trainable parameters and ';' denotes concatenation of two vectors. Finally, we feed the output layer y to a logistic regression layer to predict a binary label for each class $z \in Z$, where Z is the set of category labels for a particular psychological theory (Maslow/Reiss, Fig. 1).

3.2 Extracting Commonsense Knowledge

To improve the prediction capacity of our model, we aim to leverage external commonsense knowledge that connects expressions from the sentence and context to human need categories. For this purpose we extract multi-hop commonsense knowledge paths that connect words in the textual inputs with the offered human need categories, using as resource ConceptNet (Speer and Havasi, 2012), a large commonsense knowledge inventory. Identifying contextually relevant information from such a large knowledge base is a non-trivial task. We propose an effective two-step method to extract multi-hop knowledge paths that associate concepts from the text with human need categories: (i) collect all potentially relevant knowledge relations among concepts and human needs in a subgraph for each input sentence; (ii) rank, filter and select high-quality paths using graph-based local measures and graph centrality algorithms.

3.2.1 Construction of Sub-graphs

ConceptNet is a graph $G = (V, E)$ whose nodes are concepts and edges are relations between concepts (e.g. CAUSES, MOTIVATEDBY). For each sentence S we induce a subgraph $G' = (V', E')$ where V' comprises all concepts $c \in V$ that appear in S and the directly preceding sentence in context Cxt. V' also includes all concepts $c \in V$ that correspond to one of the human need categories in our label set Z. Fig. 3 shows an example. The sub-graph is constructed as follows:

Shortest Paths: In a first step, we find all shortest paths p' from ConceptNet that connect any concept $c_i \in V'$ to any other concept $c_j \in V'$ and to each human needs concept $z \in Z$. We further include in V' all the concepts $c \in V$ which are contained in the above shortest paths p'.

Neighbours: To better represent the meaning of the concepts in V', we further include in V' all concepts $c \in V$ that are directly connected to any $c \in V'$ that is not already included in V'.

Sub-graph: We finally construct a connected sub-graph $G' = (V', E')$ from V' by defining E' as the set of all ConceptNet edges $e \in E$ that directly connect any pair of concepts $(c_i, c_j) \in V'$.

Overall, we obtain a sub-graph that contains relations and concepts which are supposed to be useful to "explain" *why* and *how strongly* concepts c_i that appear in the sentence and context are associated with any of the human needs $z \in Z$.

3.2.2 Ranking and Selecting Multi-hop Paths

We could use all possible paths p contained in the sub-graph G', connecting concepts c_i from the text and human needs concepts z contained in G', as additional evidence to predict suitable human need categories. But not all of them may be relevant. In order to select the most relevant paths, we propose a two-step method: (i) we score each vertex with a score (*Vscore*) that reflects its importance in the sub-graph and on the basis of the vertices' *Vscores* we determine a path score *Pscore*, as shown in Figure 3; (ii) we select the top-k paths with respect to the computed path score (*Pscore*) .

(i) Vertex Scores and Path Scores: We hypothesize that the most useful commonsense relation paths should include vertices that are *important* with respect to the entire extracted subgraph. We measure the importance of a vertex using different local graph measures: the *closeness centrality measure, page rank* or *personalized page rank*.

Closeness Centrality (CC) (Bavelas, 1950) reflects how close a vertex is to all other vertices in the given graph. It measures the average length of the shortest paths between a given vertex v_i and all other vertices in the given graph G'. In a connected graph, the closeness centrality $CC(v_i)$ of a vertex $v_i \in G'$ is computed as

$$Vscore_{CC}(v_i) = \frac{\mid V' \mid}{\sum_j d\left(v_j, v_i\right)} \tag{10}$$

where $\mid V' \mid$ represents the number of vertices in the graph G' and $d(v_j, v_i)$ represents the length of the shortest path between v_i and v_j. For each path we compute the normalized sum of $Vscore_X$ of all vertices v_j contained in the path, for any measure $X \in \{CC, PR, PPR\}$.

$$Pscore_X = \frac{\sum_j Vscore_X(v_j)}{N} \tag{11}$$

We rank the paths according to their $Pscore_{CC}$, assuming that relevant paths will contain vertices that are close to the center of the sub-graph G'.

PageRank (PR) (Brin and Page, 1998) is a graph centrality algorithm that measures the relative importance of a vertex in a graph. The PageRank score of a vertex $v_i \in G'$ is computed as:

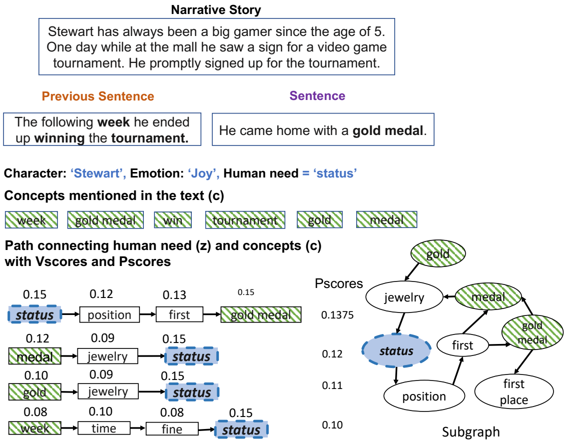
$$Vscore_{PR}(v_i) = \alpha \sum_j u_{ji} \frac{v_j}{L_j} + \frac{1-\alpha}{n} \tag{12}$$

where $L_j = \sum_i u_{ji}$ is the number of neighbors of vertex j, α is a damping factor representing the probability of jumping from a given vertex v_i to another random vertex in the graph and n represents the number of vertices in G'. We calculate $Pscore_{PR}$ using Eq. 11 and order the paths according to their $Pscore_{PR}$, assuming that relevant paths will contain vertices with high relevance, as reflected by a high number of incoming edges.

Personalized PageRank (PPR) (Haveliwala, 2002) is used to determine the importance of a vertex with respect to a certain topic (set of vertices). Instead of assigning equal probability for a random jump $\frac{1-\alpha}{n}$, PPR assigns stronger probability to certain vertices to prefer topical vertices. The PPR score of a vertex $v \in G'$ is computed as:

$$Vscore_{PPR}(v_i) = \alpha \sum_j u_{ji} \frac{v_j}{L_j} + (1-\alpha)\, T \tag{13}$$

where $T = \frac{1}{|T_j|}$ if nodes v_i belongs to topic T_j and otherwise $T = 0$. In our setting, T_j will contain concepts from the text and human needs, to assign them higher probabilities. We calculate $Pscore_{PPR}$ using Eq. 11 and order the paths according to their scores, assuming that relevant paths should contain vertices holding importance with respect to vertices representing concepts from the text and human needs.



Stewart has always been a big gamer since the age of 5. One day while at the mall he saw a sign for a video game tournament. He promptly signed up for the tournament.

The following **week** he ended up **winning** the **tournament.**

He came home with a **gold medal**.

Figure 3: Illustration of commonsense path selection. Top: Context and sentence, Bottom: Selected knowledge paths with *Vscores* and *Pscores* (left) and the corresponding subgraph. Concepts from the text are marked with green dashed lines; blue boxes show the human need label *status* assigned to *Stewart*.

(ii) Path Selection: We rank knowledge paths based on their *Pscore* using the above relevance measures, and construct ranked lists of paths of two types: (i) paths connecting a human needs concept $z \in Z$ to a concept mentioned in the text (p_{c-z}) [2] and (ii) paths connecting concepts in the text (p_{c-c}) [3]. Ranked lists of paths are constructed individually for concepts that constitute the start or endpoint of a path: a human needs concept for p_{c-z} or any concept from the text for p_{c-c}.

Figure 3 illustrates an example where the character *Stewart* felt *joy* after winning a gold medal. The annotated human need label is *status*. We show the paths selected by our algorithm that connect concepts from the text and the human need *status*. We select the top-k paths of type p_{c-z} for each human need to capture relevant knowledge about human needs in relation to concepts in the text. Similarly, we select the top-k paths of type p_{c-c} for each c_i to capture relevant knowledge about the text (not shown in Fig. 3).

3.3 Extending the Model with Knowledge

We have seen how to obtain a ranked list of commonsense knowledge paths from a subgraph extracted from ConceptNet that connect concepts from the textual input and possible human needs categories that are the system's classification tar-

---

[2] p_{c-z} denotes path connecting a human needs concept $z \in Z$ and a concept c mentioned in the text.

[3] p_{c-c} denotes path connecting a concept c and another concept c mentioned in the text.

gets. Our intuition is that the extracted commonsense knowledge paths will provide useful evidence for our model to *link the content expressed in the text to appropriate human need categories*. Paths that are selected by the model as a relevant connection between the input text and the labeled human needs concept can thus provide *explanations* for emotions or goals expressed in the text *in view of a human needs category*. We thus integrate these knowledge paths into our model, (i) to help the model making correct predictions and (ii) to provide explanations of emotions expressed in the text in view of different human needs categories. For each input, we represent the extracted ranked list of n commonsense knowledge paths p as a list $cr^{k,1}, cr^{k,2}, ..., cr^{k,n}$, where each $cr^{k,i}_{1:l}$ represents a path consisting of concepts and relations, with l the length of the path. We embed all concepts and relations in $cr^{k,i}_{1:l}$ with pretrained GloVe (Pennington et al., 2014) embeddings.

Encoding Layer: We use a single-layer BiLSTM to obtain encodings ($h^{k,i}$) for each knowledge path

$$h^{k,i} = BiLSTM(e^{k,i}_{1:n}) \tag{14}$$

where h^k represents the output of the BiLSTM for the knowledge path and i its the ranking index.

Attention Layer: We use an attention layer, where each encoded commonsense knowledge path interacts with the sentence representation x^s to receive attention weights ($\hat{h}^{k,i}$):

$$\widetilde{h}^{k,i} = \sigma(x^s h^{k,i}), \quad \hat{h}^{k,i} = \frac{\widetilde{h}^{k,i}}{\sum_{i=1}^{N} \widetilde{h}^{k,i}} \tag{15}$$

In Eq. 15, we use sigmoid to calculate the attention weights, similar to Eq. 6. However, this time we compute attention to highlight which knowledge paths are important for a given input representation (x^s being the final state hidden representation over the input sentence, Eq. 7). To obtain the sentence-aware commonsense knowledge representation x^k, we pass the output of the attention layer through a feedforward layer. W_k, b_k are trainable parameters.

$$x^k = ReLU(W_k(\sum_{i=1}^{N} \hat{h}^{k,i} h^{k,i}) + b_k) \tag{16}$$

3.4 Distilling Knowledge into the Model

In order to incorporate the selected and weighted knowledge into the model, we concatenate the sentence x^s, context x^{cxt} and knowledge x^k representation and pass it through a FF layer.

$$o_i = ReLU(W_z[x^s_i; x^{cxt}_i; x^k_i] + b_z) \tag{17}$$

Classification	**Train**	**Dev**	**Test**
Reiss	5432	1469	5368
Reiss without *belonging* class	5431	1469	5366
Maslow	6873	1882	6821

Table 1: Dataset Statistics: nb. of instances (sentences with annotated characters and human need labels).

We employ a gating mechanism to allow the model to selectively incorporate relevant information from commonsense knowledge x^k and from the joint input representation y_i (see Eq. 9) *separately*. We finally pass it to a logistic regression classifier to predict a binary label for each class z in the set Z of category labels

$$z_i = \sigma(W_{\widetilde{y}_z}(o_i \odot y_i + o_i \odot x^k_i) + b_{\widetilde{y}_z}) \tag{18}$$

where $\odot$ represents element-wise multiplication, $b_{\widetilde{y}_z}$, $W_{\widetilde{y}_z}$ are trainable parameters.

4 Experimental Setup

Dataset: We evaluate our model on the *Modeling Naive Psychology of Characters in Simple Commonsense Stories (MNPCSCS)* dataset (Rashkin et al., 2018). It contains narrative stories where each sentence is annotated with a character and a set of human need categories from two inventories: Maslow's (with five coarse-grained) and Reiss's (with 19 fine-grained) categories (Reiss's labels are considered as subcategories of Maslow's). The data contains the original worker annotations. Following prior work we select the annotations that display the "majority label" i.e., categories voted on by ≥ 2 workers. Since no training data is available, similar to prior work we use a portion of the devset as training data, by performing a random split, using 80% of the data to train the classifier, and 20% to tune parameters. Data statistics is reported in Table 1.

Rashkin et al. (2018) report that there is low annotator agreement i.a. between the *belonging* and the *approval* class. We also find high co-occurrence of the *belonging, approval* and *social contact* classes, where *belonging* and *social contact* both pertain to the Maslow class *Love/belonging* while *approval* belongs to the

Maslow class *Esteem*. This indicates that *belonging* interacts with *Love/belonging* and *Esteem* in relation to social contact. We further observed during our study that in the Reiss dataset the number of instances annotated with the *belonging* class is very low (no. of instances in training is 24, and in dev 5). The performance for this class is thus severely hampered, with 4.7 F_1 score for BiLSTM+Self-Attention and 7.1 F_1 score for BiLSTM+Self-Attention+Knowledge. After establishing benchmark results with prior work (cf. Table 2, including *belonging*), we perform all further experiments with a reduced Reiss dataset, by eliminating the *belonging* class from all instances. This impacts the overall number of instances only slightly: by *one* instance for training and *two* instances for test, as shown in Table 1.

Training: During training we minimize the weighted binary cross entropy loss,

$$L = \sum_{z=1}^{Z} w_z y_z log\widetilde{y}_z + (1 - w_z)(1 - y_z)log(1 - \widetilde{y}_z) \tag{19}$$

$$w_z = \frac{1}{1 - exp^{-\sqrt{P(y_z)}}} \tag{20}$$

where Z is the number of class labels in the classification tasks and w_z is the weight. $P(y_z)$ is the marginal class probability of a positive label for z in the training set.

Embeddings: To compare our model with prior work we experiment with pretrained GloVe (100d) embeddings (Pennington et al., 2014). Otherwise we used GloVe (300d) and pretrained ELMo embeddings (Peters et al., 2018) to train our model.

Hyperparameters for Knowledge Inclusion: We compute ranked lists of knowledge paths of two types: p_{c-z} and p_{c-c}. We use the top-3 p_{c-z} paths for each z using our best ranking strategy (Closeness Centrality + Personalized PageRank) in our best system results (Tables 2, 3, 5), and also considered paths p_{c-c} (top-3 per pair) when evaluating different path selection strategies (Table 4).

Evaluation Metrics: We predict a binary label for each class using a binary classifier so the prediction of each label is conditionally independent of the other classes given a context representation of the sentence. In all prediction tasks we report the micro-averaged Precision (P), Recall (R) and F_1 scores by counting the number of positive instances across all of the categories. All reported results are averaged over five runs. More information on the dataset, metrics and all other training details are given in the Supplement.

		Reiss			Maslow		
Model	**WE**	**P**	**R**	**F1**	**P**	**R**	**F1**
BiLSTM$^\diamond$	G_{100d}	18.35	27.61	22.05	31.29	33.85	32.52
CNN$^\diamond$	G_{100d}	18.89	31.22	23.54	27.47	41.01	32.09
REN$^\diamond$	G_{100d}	16.79	22.20	19.12	26.24	42.14	32.34
NPN$^\diamond$	G_{100d}	13.13	26.44	17.55	24.27	44.16	31.33
BM	G_{100d}	25.08	28.25	26.57	47.65	60.98	53.54
BM + K$^\clubsuit$	G_{100d}	**28.47**	**39.13**	**32.96**	**50.54**	**64.54**	**56.69**
BM	ELMo	29.50	**44.28**	35.41$_{\pm 0.23}$	53.86	**67.23**	59.81$_{\pm 0.23}$
BM + K$^\clubsuit$	ELMo	**31.74**	43.51	**36.70**$_{\pm 0.14}$	**57.90**	66.07	**61.72**$_{\pm 0.11}$
BM*	ELMo	31.45	44.29	37.70			
BM + K$^{\star\clubsuit}$	ELMo	36.76	42.53	39.44			

Table 2: Multi-label Classification Results: $^\diamond$: results in Rashkin et al.; *: w/o *belonging*; BM: BiLSTM+Self-Att.; +K:w/ knowledge, $^\clubsuit$:ranking method CC+PPR.

5 Results

Our experiment results are summarized in Table 2. We benchmark our baseline BiLSTM+Self-Attention model (BM, BM w/ knowledge) against the models proposed in Rashkin et al. (2018): a BiLSTM and a CNN model, and models based on the recurrent entity network (REN) (Henaff et al., 2016) and neural process networks (NPN) (Bosselut et al., 2017). The latter differ from the basic encoding models (BiLSTM, CNN) and our own models by explicitly modeling entities. We find that our baseline model BM outperforms all prior work, achieving new state-of-the-art results. For Maslow we show improvement of 21.02 pp. F_1 score. For BM+K this yields a boost of 6.39 and 3.15 pp. F_1 score for Reiss and Maslow, respectively. When using ELMo with BM we see an improvement in recall. However, adding knowledge on top improves the precision by 2.24 and 4.04 pp. for Reiss and Maslow. In all cases, injecting knowledge improves the model's precision and F_1 score.

Table 2 (bottom) presents results for the reduced dataset, after eliminating Reiss' label *belonging*. Since *belonging* is a rare class, we observe further improvements. We see the same trend: adding knowledge improves the precision of the model.

5.1 Model Ablations

To obtain better insight into the contributions of individual components of our models, we perform an ablation study (Table 3). Here and in all later experiments we use richer (300d) GloVe embeddings and the dataset w/o *belonging*. We show results including and not including self-attention

WE	Atten	K	Gated	P	R	F1
G$_{300d}$	-	-	-	23.31	34.69	27.89
G$_{300d}$	✓	-	-	26.09	35.59	30.11
G$_{300d}$	✓	✓	-	27.99	37.73	32.14
G$_{300d}$	✓	✓	✓	**28.65**	**39.42**	**33.19**
ELMo	-	-	-	32.35	42.66	36.80
ELMo	✓	-	-	31.45	44.29	37.70
ELMo	✓	✓	-	32.65	**45.60**	38.05
ELMo	✓	✓	✓	**36.76**	42.53	**39.44**

Table 3: Model ablations for Reiss Classification on *MNPCSCS* dataset w/o *belonging*.

Path	Ranking	P	R	F1
S+M(P_{c-z}+ P_{c-c})	None	32.51	42.70	36.90
S+M(P_{c-z}+ P_{c-c})	Random	31.63	43.35	36.57
Single Hop(P_{c-z})	CC + PPR	33.00	44.63	37.94
S+M(P_{c-c} + P_{c-z})	CC + PPR	35.30	44.11	39.21
S+M(P_{c-z})	CC	33.45	**47.93**	39.40
S+M(P_{c-z})	PR	35.51	42.82	38.82
S+M(P_{c-z})	PPR	36.23	43.09	39.34
S+M(P_{c-z})	CC + PPR	**36.76**	42.53	**39.44**

Table 4: Results for different path selection strategies on *MNPCSCS* w/o *belonging*; S+M:Single+Multi hop.

and knowledge components. We find that using self-attention over sentences and contexts is highly effective, which indicates that learning how much each token contributes helps the model to improve performance. We observe that integrating knowledge improves the overall F_1 score and yields a gain in precision with ELMo. Further, integrating knowledge using the gating mechanism we see a considerable increase of 3.58 and 1.74 pp. F_1 score improvement over our baseline model for GloVe and ELMo representations respectively.

5.2 Commonsense Path Selection

We further examine model performance for (i) different variants of selecting commonsense knowledge, including (ii) the effectiveness of the relevance ranking strategies discussed in §3.2.2. In Table 4, rows 3-4 use our best ranking method: CC+PPR; rows 5-8 show results when using the top-3 ranked p_{c-z} paths for each human need z with different ranking measures. *None* shows results when no selection is applied to the set of extracted knowledge paths (i.e., using all possible paths from p_{c-z} and p_{c-c}). *Random* randomly selects 3 paths for each human need from the set of paths used in *None*. This yields only a slight drop in performance. This suggests that not every path is relevant. We evaluate the performance when only considering single-hop paths (now top-3 ranked using CC+PPR) *(Single-Hop)*. We see an improvement over random paths and no selection, but not important enough. In contrast, using both single and multi-hop paths in conjunction with relevance ranking improves the performance considerably (rows 4-8). This demonstrates that multi-hop paths are informative. We also experimented with p_{c-c}+p_{c-z}. We find improvement in recall, however the overall performance decreases by 0.2 F_1 score compared to paths p_{c-z} ranked using CC + PPR. Among different ranking measures *precision* for Personalized PageRank performs best in comparison with CC and PR in isolation, and recall for CC in isolation is highest. Combining CC and PPR yields the best results among the different ranking strategies (rows 5-8).

6 Analysis

6.1 Performance per Human Need Categories

We examined the model performance on each category (cf. Figure 4). The model performs well for basic needs like *food*, *safety*, *health*, *romance*, etc. We note that inclusion of knowledge improves the performance for most classes (only 5 classes do not profit from knowledge compared to only using ELMo), especially for labels which are rare like *honor, idealism, power.* We also found that the annotated labels can be subjective. For instance, *Tom lost his job* is annotated with *order* while our model predicts *savings*, which we consider to be correct. Similar to Rashkin et al. (2018) we observe that preceding context helps the model to better predict the characters' needs, e.g., *Context: Erica's [..] class had a reading challenge [..]. If she was able to read 50 books [..] she won a pizza party!; Sentence: She read a book every day for the entire semester* is annotated with *competition*. Without context the predicted label is *curiosity*, however when including context, the model predicts *competition, curiosity*. We measure the models performance when applying it only to the first sentence of each story (i.e., without the context). As shown in Table 5, also in this setting the inclusion of knowledge improves the performance.

Model	WE	P	R	F1
BM	ELMo	33.39	45.15	38.39
BM+K	ELMo	36.36	44.02	39.83

Table 5: Multi-label classification on *MNPCSCS* w/o *belonging class* and w/o context (1^{st} sentence only)

.

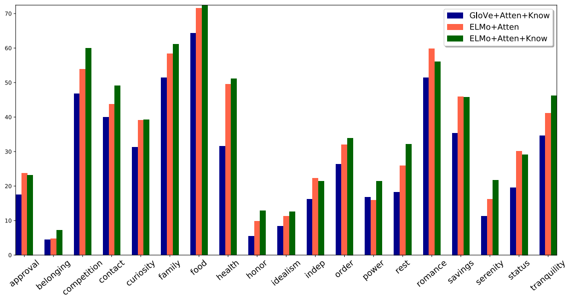


Figure 4: Best model's performance per human needs (F_1 scores) for Reiss on *MNPCSCS* dataset.

Context: Timmy had to renew his driver's license. He went to his local DMV. He waited in line for nearly 2 hours. He took a new picture for his driver's license.
Sentence: He drove back home after an exhausting day.
True Label: *rest*
Predicted Label (BM): *status, approval, order*
Predicted Label (BM+K): *rest*

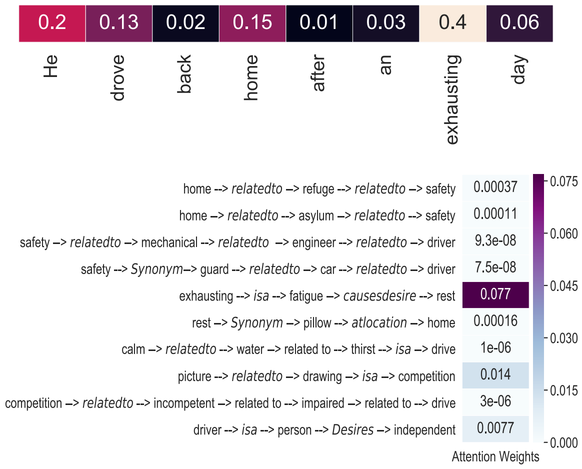


Figure 5: Interpreting the attention weights on sentence representation and selected commonsense paths.

6.2 Human Evaluation of Extracted Paths

We conduct human evaluation to test the effectiveness and relevance of the extracted commonsense knowledge paths. We randomly selected 50 sentence-context pairs with their gold labels from the devset and extracted knowledge paths that contain the gold label (using CC+PPR for ranking). We asked three expert evaluators to decide whether the paths are relevant to provide information about the missing links between the concepts in the sentence and the human need (gold label). The inter-annotator agreement had a Fleiss' κ= 0.76. The result for this evaluation shows that in 34% of the cases computed on the basis of majority agreement, our algorithm was able to select a relevant commonsense path. More details about the human evaluation are given in the Supplement.

6.3 Interpretabilty

Finally we study the learned attention distributions of the interactions between sentence representation and knowledge paths, in order to interpret how knowledge is employed to make predictions. Visualization of the attention maps gives evidence of the ability of the model to capture relevant knowledge that connects human needs to the input text. The model provides interpretability in two ways: by selecting tokens from the input text using Eq.6 and by choosing knowledge paths from the imported knowledge using Eq.15 as shown in Figure 5. Figure 5 shows an example where including knowledge paths helped the model to predict the correct human need category. The attention map depicts which exact paths are selected to make the prediction. In this example, the model correctly picks up the token *"exhausting"* from the input sentence and the knowledge path *"exhausting is a fatigue causes desire rest"*. We present more examples of extracted knowledge and its attention visualization in the Supplement.

7 Conclusion

We have introduced an effective new method to rank multi-hop relation paths from a commonsense knowledge resource using graph-based algorithms. Our end-to-end model incorporates multi-hop knowledge paths to predict human needs. Due to the attention mechanism we can analyze the knowledge paths that the model considers in prediction. This enhances transparency and interpretability of the model. We provide quantitative and qualitative evidence of the effectiveness of the extracted knowledge paths. We believe our relevance ranking strategy to select multi-hop knowledge paths can be beneficial for other NLU tasks. In future work, we will investigate structured and unstructured knowledge sources to find explanations for sentiments and emotions.

Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We thank NVIDIA Corporation for donating GPUs used in this research. We thank Éva Mújdricza-Maydt, Esther van den Berg and Angel Daza for evaluating the paths, and Todor Mihaylov for his valuable feedback.

References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230.

Alex Bavelas. 1950. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks. *CoRR*, abs/1711.05313.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

Eunsol Choi, Hannah Rashkin, Luke Zettlemoyer, and Yejin Choi. 2016. Document-level sentiment inference with social, faction, and discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 333–343.

Lingjia Deng and Janyce Wiebe. 2015. Joint prediction for entity/event-level sentiment analysis using probabilistic soft logic models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 179–189.

Haibo Ding and Ellen Riloff. 2018. Human needs categorization of affective events using labeled and unlabeled data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1919–1929.

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.

Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *CoRR*, abs/1612.03969.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

Jiwei Li and Eduard Hovy. 2017. Reflections on sentiment/opinion analysis. In *A Practical Guide to Sentiment Analysis*, pages 41–59. Springer.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Abraham Harold Maslow. 1943. A theory of human motivation. *Psychological review*, 50(4):370.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Theories of emotion*, 1(4):3–31.

Elahe Rahimtoroghi, Jiaqi Wu, Ruimin Wang, Pranav Anand, and Marilyn Walker. 2017. Modelling protagonist goals and desires in first-person narrative. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 360–369.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling naive psychology of characters in simple commonsense stories. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2289–2299.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 293–302.

Steven Reiss. 2002. *Who am I?: 16 basic desires that motivate our actions define our persona*. Penguin.

Steven Reiss. 2004. Multifaceted nature of intrinsic motivation: The theory of 16 basic desires. *Review of general psychology*, 8(3):179.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Robert Speer and Catherine Havasi. 2012. Representing General Relational Knowledge in ConceptNet 5. In *LREC*, pages 3679–3686.

Srikanth Tamilselvam, Seema Nagar, Abhijit Mishra, and Kuntal Dey. 2017. Graph Based Sentiment Aggregation using ConceptNet Ontology. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 525–535.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic Integration of Background Knowledge in Neural NLU Systems. *arXiv preprint arXiv:1706.02596*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into conversation modeling via recall-gate LSTM. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3506–3513. IEEE.

Yichun Yin, Yangqiu Song, and Ming Zhang. 2017. Document-level multi-aspect sentiment classification as machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2054.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

NLP Whack-A-Mole: Challenges in Cross-Domain Temporal Expression Extraction

Amy L. Olex, Luke G. Maffey, and Bridget T. McInnes
Department of Computer Science
Virginia Commonwealth University
401 East Main Street, Richmond VA 23298
`alolex,maffeyl,btmcinnes@vcu.edu`

Abstract

Incorporating domain knowledge is vital in building successful natural language processing (NLP) applications. Many times, cross-domain application of a tool results in poor performance as the tool does not account for domain-specific attributes. The clinical domain is challenging in this aspect due to specialized medical terms and nomenclature, shorthand notation, fragmented text, and a variety of writing styles used by different medical units. Temporal resolution is an NLP task that, in general, is domain-agnostic because temporal information is represented using a limited lexicon. However, domain-specific aspects of temporal resolution are present in clinical texts. Here we explore parsing issues that arose when running our system, a tool built on Newswire text, on clinical notes in the THYME corpus. Many parsing issues were straightforward to correct; however, a few code changes resulted in a cascading series of parsing errors that had to be resolved before an improvement in performance was observed, revealing the complexity of temporal resolution and rule-based parsing. Our system now outperforms current state-of-the-art systems on the THYME corpus with little change in its performance on Newswire texts.

1 Introduction

Temporal resolution is required for comprehending many types of communication, including written texts. This is especially true in clinical texts as patient narratives revolve around when an event happened, such as when a symptom occurred or the frequency a drug was administered (Lee et al., 2017; Sun et al., 2013b). Understanding the temporal component in texts is vital for many NLP systems (Tissot et al., 2015) to accurately interpret a patient narrative (Sun et al., 2013b).

Some temporal expressions could be considered domain agnostic as there are limited ways to represent information about time, such as formatted dates or days of the week. However, there are many lexical variations of these standard tokens. Additionally, vague temporal expressions, relative times, and event durations require contextual or implicit knowledge of the subject area for resolution (Sun et al., 2013b). Clinical texts include all these types of temporal expressions, and also contain domain-specific challenges to temporal expression identification and normalization, such as differentiating between dosage and time. Additionally, clinical texts frequently use repeated phrases such as "At this time" that are infrequently used in the general domain. These phrases are vague, relative, and require contextual knowledge of the subject area and the time of events to be resolved (Sun et al., 2013b).

In this work we focus on identification of temporal expressions in clinical texts using Chrono– a hybrid system that normalizes temporal expressions into the SCATE Schema (Bethard and Parker, 2016). Originally designed on general domain Newswire texts, we evaluate Chrono's performance on the clinical THYME corpus (Styler et al., 2014) "out-of-the-box" with no modifications, perform an error analysis, algorithm updates, and then re-evaluate on THYME. This analysis reveals six aspects of temporal expression extraction that should be considered when using a general domain tool in the clinical domain.

2 Related Work

State-of-the-art temporal expression extraction and normalization tools have emerged from temporal parsing challenges such as TempEval (Verhagen et al., 2007, 2010; UzZaman et al., 2013) and i2b2 (Sun et al., 2013a). Strategies utilized by these tools range from rule-based (SUTime (Chang and Manning, 2012), HeidelTime

Proceedings of NAACL-HLT 2019, pages 3682–3692
Minneapolis, Minnesota, June 2 - June 7, 2019.

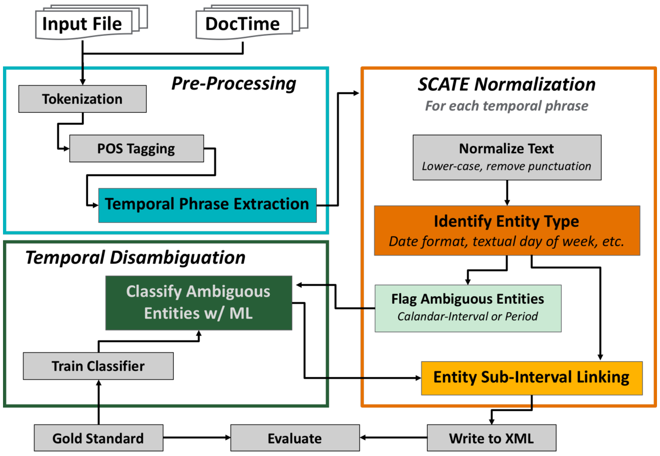


Figure 1: Overview of Chrono Workflow

(Strötgen and Gertz, 2010), NavyTime (Chambers, 2013), GUTime (Verhagen et al., 2005)) to machine learning (TRIPS and TRIOS (UzZaman and Allen, 2010), ClearTK (Bethard, 2013)) and hybrid approaches (ManTIME (Filannino et al., 2013)). For general domain texts, machine learning systems like ClearTK perform well at identifying temporal expression spans; however, rule-based and hybrid systems have better performance when taking temporal expression normalization into account (UzZaman et al., 2013).

When applied to clinical text in the 2012 i2b2 Challenge, high-ranking general domain systems SUTime, GUTime, and HeidelTime had reduced performance (Sun et al., 2013a) as compared to systems built specifically for this data (Sohn et al., 2013; Kovaevi et al., 2013; Xu et al., 2013). Regardless of the performance on general domain texts, modifications had to be made to the state-of-the-art systems to recognize clinical temporal expressions and achieve improved performance. For example, three teams utilized HeidelTime with two teams incorporating additional rules and machine learning modules on top of the default system, which achieved better performance in the 2012 i2b2 Challenge than HeidelTime with no modifications.

In addition to temporal challenges, other systems have been developed for general domain temporal parsing that utilize machine learning and complex grammars (Lee et al., 2014; Angeli et al., 2012) and rule-based methods referencing a central knowledge base (Llorens et al., 2012). SynTime (Zhong et al., 2017) takes a simplistic approach by defining a layer of syntactic token types that rules are applied to instead of processing the raw tokens. For temporal expression extraction, SynTime out-performs HeidelTime and SUTime, however, it does not attempt normalization. All these systems were built and trained on general domain texts, such as TimeBank (Pustejovsky et al., 2003) and WikiWars (Mazur and Dale, 2010) and may require adjustments to accurately capture clinical temporal expressions. In addition, these systems normalize expressions into the ISO-TimeML (Pustejovsky et al., 2010) representation, which is unable to represent expressions that don't map to a single calendar unit or are relative to an event instead of a temporal unit–both of which are frequent in clinical texts. The SCATE schema is able to faithfully represent these types of expressions, but normalization requires a more detailed approach to annotate fine-grained temporal components that are not captured by TimeML (Bethard and Parker, 2016). In this work we adapt Chrono, a novel SCATE normalization system, to the clinical domain and describe the challenges encountered when normalizing to the SCATE Schema.

3 Methods

Chrono is a hybrid rule-based and machine learning system built to identify temporal expressions in the AQUAINT corpus of Newswire texts (Graff, 2002) followed by normalization into the SCATE Schema for SemEval 2018 Task 6 (Laparra et al., 2018). Chrono consists of 3 main modules: 1) Temporal Phrase Extraction, 2) SCATE Normalization, and 3) Temporal Disambiguation (Figure 1). Briefly, the Temporal Phrase Extraction module identifies temporal/numeric tokens using a series of hierarchical rules and regular expressions. Temporal phrases are extracted based on consecutive tagged temporal/numeric tokens. Next, the SCATE Normalization module normalizes temporal phrases into the SCATE Schema using additional rule-based logic and regular expressions to identify specific temporal entities within each phrase, and links related sub-intervals. Finally, machine learning is used in the Temporal Disambiguation module as a sub-module of SCATE Normalization to disambiguate certain SCATE entities. Details on the specific rules implemented by Chrono for SemEval 2018 can be found in the systems description paper (Olex et al., 2018), and Chrono can be downloaded from https://github.com/AmyOlex/Chrono.

4 THYME Corpus

The THYME corpus consists of de-identified clinical notes and pathology reports for colon and brain cancer patients. For this work, we utilized the subset of the THYME colon cancer documents that have associated SCATE annotations in the Anafora XML format from SemEval 2018 Task 6 (Laparra et al., 2018). The Training Corpus includes 22 clinical notes and 13 pathology reports along with their gold standard Anafora XML annotations. The Evaluation Corpus includes 92 clinical notes and 49 pathology reports with the annotations withheld. In this work, Chrono is first run on the THYME Evaluation Corpus before modifications are made, then the THYME Training Corpus is used to identify problem areas in need of improvement. Finally, Chrono is run on the Evaluation Corpus again after making improvements. Data in the Evaluation Corpus remained hidden through the entire process.

5 Evaluation

Evaluation of Chrono's performance on the Training Corpus utilized python scripts provided by AnaforaTools[†] that compare Anafora XML (Chen and Styler, 2013) annotation files. All metrics reported exclude the "Event" entity because event identification is currently not implemented by Chrono, and was not included in the SemEval Task. Chrono's annotation of the Evaluation Corpus was uploaded to the Post-Evaluation submission system for SemEval 2018 Task 6, and overall Precision, Recall, and F1 measures are reported in Tables 1 and 3.

6 Results and Discussion

This section first discusses Chrono's "out-of-the-box" performance on the THYME Evaluation Corpus prior to any code changes. The next section presents parsing issues encountered using the Training Corpus that fall into six main categories: 1) lexical, 2) entity frequency, 3) numeric disambiguation, 4) machine learning training data, 5) writing style, and 6) document structure. While fixing some of these issues was straightforward, more complex issues resulted in debugging an error cascade before performance increased. Finally, a discussion of Chrono's improved performance on the THYME Evaluation Corpus is presented.

6.1 Out-of-the-Box Performance on THYME

Chrono's performance decreased significantly on the THYME Evaluation Corpus out-of-the-box with an F1 of 0.35, precision of 0.49, and recall of 0.27 (Table 1). This is due to Chrono having only been trained on Newswire text, thus, it saw a limited number of temporal expression examples.

Chrono's performance on the THYME Training Corpus resulted in an F1 of 0.314 when considering all entity properties (100% Correct Entity), and an F1 of 0.468 when only considering correct token span (Span Only). The higher Span Only result indicates that Chrono is identifying more correct entities than the 100% Correct Entity score indicates, but it is not assigning all the properties correctly. With the AnaforaTools evaluation script we are able to look at the performance on each SCATE entity individually to identify specific entities that significantly impact performance.

Dataset	System	Precision	Recall	F1
THYME Eval	Chrono	0.49	0.27	0.35
THYME Eval	Laparra et. al.	0.52	0.63	0.57
Newswire Eval	Chrono	0.61	0.50	0.55
Newswire Eval	Laparra et. al.	0.58	0.46	0.51
THYME Train	Chrono 100%	0.439	0.244	0.314
THYME Train	Chrono Span Only	0.696	0.352	0.468

Table 1: Baseline performance, excluding "Event", on THYME Training and Evaluation corpora using SVM.

6.2 NLP Whack-A-Mole - Resolving Cross-Domain NLP Challenges

Addressing cross-domain parsing issues felt synonymous to playing the arcade game of Whack-A-Mole, where as one issue was fixed another popped up. Several code improvements resulted

---

[†] https://github.com/bethard/anaforatools

in a cascading series of other code bugs and/or logical issues that needed resolution prior to realizing a performance improvement. This section describes these adventures in code improvement, which identify six primary challenges encountered in cross-domain application of temporal expression extraction. The following examples relay how complex and interconnected temporal expression extraction can be, and demonstrate the need to go beyond basic pattern identification and dictionary look-up strategies to including contextual and semantic information in order to capture all types of temporal expressions.

6.2.1 Lexical Diversity

Different domains are expected to differ in their lexicon. For example, the clinical domain contains many specialized medical terms and clinical jargon that is not encountered in general domain texts (Meystre et al., 2008). This is also true for a temporal lexicon. Originally trained on the Newswire corpus, Chrono's lexicon was limited to examples found in this domain; however, by expanding Chrono's temporal lexicon the performance on several SCATE entities increased.

Performance on the SCATE entity "Modifier" improved after refining the lexicon to include missed terms such as "nearly", "almost", "mid", "over", "early", and "beginning", and removing terms that should be annotated with other entities such as "this", "next", and "last". These descriptive temporal tokens are commonly used in clinical texts to describe various events in the patient narrative such as when symptoms occur or patient histories. The PartOfDay entity was also augmented with the terms "bedtime", "eve", and "midnight" as these, and similar terms, are frequently utilized in clinical notes for medication instructions, such as "take one pill at bedtime". Significant improvement in performance was observed after these additions, with an F1 increase of 0.117 for PartOfDay, and an F1 increase of 0.241 for Modifier.

Patient records revolve around temporal information, such as conveying medication instructions, describing symptom time lines, and outlining patients' histories. We found that temporal phrases associated with these events, like "at that time", "take one-time daily", "in four weeks time", "since that time", etc., were ubiquitous. All of these expressions include the token "time", which is annotated as a Period entity in the SCATE Schema. This token, along with others found frequently in clinical text such as "/min" and "/week" that are most commonly used as short-hand for conveying medication frequency, were not included in Chrono's temporal lexicon. This resulted in poor performance for the CalendarInterval and Period SCATE entities. The addition of 15 terms that were not present in the Newswire corpus significantly improved performance for these phrases. This result indicates that commonly used tokens have domain-specific frequencies. For example, the token "time" was used on average 0.32 times per document in the Newswire corpus and just over 4 times per document in the THYME corpus (Table 2).

6.2.2 Frequent Frequency

The frequency for some lexical terms, like "time", in clinical texts is understandable as certain concepts that convey a patient's narrative may be utilized over and over again. However, it is interesting that this observation also applies at the temporal entity level. For example, the initial build of Chrono excluded the SCATE entity Frequency because it is highly complex to parse and did not appear regularly in the Newswire corpus (0.12 times per document on average, Table 2). However, in the THYME corpus, the Frequency entity appeared on average 8.9 times per document–a 72-fold increase–which had a major impact on Chrono's performance. In clinical texts, phrases specifying frequency such as "2 time per day" or "once a day" are abundant as they are routinely used for specifying medication or symptom frequency. This increase in clinical usage extends to all but two temporal entities, with Frequency having the second highest fold change next to Event (Table 2).

6.2.3 Disambiguating Dosage

Clinical text commonly contains non-temporal numerical information representing lab test results or medication dosage along with their frequency. The majority of these instances in the THYME corpus were not identified as temporal because their values and formats were distinct. However, Chrono confused a few occurrences of medication dosage with a 24-hour time instance. For example, in the phrase "Vitamin D-3 1000 unit tablet" the "1000" was incorrectly assigned the 24-hour time value of 10am. In the current implementation of Chrono, if a 4-digit dose falls within the correct year range (1500 to 2050) or 24-hour time it will

Entity	Chrono Implements	Newswire Avg Freq	Clinical Avg Freq
AMPM-Of-Day	Y	0.06	1.26
After	Y	0.25	2.29
Before	Y	0.44	0.91
Between	N	0.28	1.11
Calendar-Interval	Y	1.83	6.80
Day-Of-Month	Y	2.84	8.66
Day-Of-Week	Y	1.33	1.29
Event	N	0.91	151.97
Every-Nth	N	0	0.09
Frequency	N	0.12	8.91
Hour-Of-Day	Y	1.15	1.46
Intersection	Y	0.11	1.60
Last	Y	2.80	3.86
Minute-Of-Hour	Y	1.12	1.31
Modifier	Y	0.42	1.31
Month-Of-Year	Y	3.31	9.77
Next	Y	0.72	0.80
NotNormalizable	N	0.06	0.06
NthFromStart	Y	0.30	0
Number	Y	1.17	13.66
Part-Of-Day	Y	0.19	0.91
Part-Of-Week	Y	0.04	0
Period	Y	1.64	4.97
Season-Of-Year	Y	0.07	0.03
Second-Of-Minute	Y	0.67	0.17
Sum	N	0.01	0.03
This	Y	1.43	2.60
Time-Zone	Y	0.44	0
Two-Digit-Year	Y	0.98	0.23
Union	N	0.02	0.03
Year	Y	1.67	9.91

Table 2: The average frequency per document of each SCATE Entity for the Newswire (81 documents) and THYME (35 documents) training corpora. The "Chrono Implements" column indicates whether or not Chrono identifies a given entity (Y=yes, N=no).

be annotated as such. A fix for this issue has yet to be implemented in Chrono, as it has a low rate of occurrence, but may include rules to identify dosage amounts such as "mg" and machine learning methods to disambiguate 4-digit numbers.

Another example of the need to disambiguate numerical values is found in the clinical phrase "Carotid pulses are 4/4". Without context, the "4/4" could be interpreted as the date "April 4th". This instance did not cause an issue with Chrono because a 2- or 4-digit year is required for a phrase to be identified as a formatted date. While this strategy worked for this example, it could become a problem when parsing files that contain year-less formatted dates. Thus, future improvements will include a numerical disambiguation module to aid in determining if a numerical phrase is temporal.

6.2.4 Cross-Domain Machine Learning Training Data

Supervised machine learning (ML) methods require the use of annotated training data in order to generate a predictive model. Naturally, training data is chosen from the domain of the task as it is the most relevant. Chrono utilizes ML to disambiguate the SCATE entities Period and CalendarInterval. First, rule-based logic identifies if an entity is a possible Period or CalendarInterval, but it is hard to tell which one without considering context. Then the ML module decides which class the entity should be labeled. The training data for this task was initially from the Newswire corpus, but this performed poorly on clinical texts with an overall F1 of 0.544. To incorporate domain-specific contextual elements, Chrono was re-trained using just the THYME corpus, which improved performance to an F1 of 0.577. We then generated a model that utilized both the Newswire and THYME data, which performed slightly better, giving an F1 of 0.578. As temporal expressions can be domain-agnostic, it makes sense that training on cross-domain data would generate a more robust and generalizable model; therefore, we chose to use the cross-domain model.

6.2.5 Lexical Variation

An advantage of processing clinical texts is that you are introduced to a variety of writing styles and preferences from different departments and medical personnel, where each may represent the same temporal concept differently. This results in lexical variations of concepts, for example, the concept of "Monday" can be represented as "M", "Mon.", or, "monday", and a temporal reasoning system must be able to identify that these all refer to the same day. The following sub-sections discuss issues associated with variation in formatted dates, times, and long temporal phrases.

Variation in Formatted Dates/Times: There are a number of standard formats to convey dates and times, of which only a few were identified in the Newswire corpus and implemented in Chrono. Clinical texts introduced additional variability in date and time formats that Chrono was unable to handle correctly. For example, the date format "21-SEP-2009" contains a mixture of letters and numbers needing to be interpreted. Chrono uses regular expressions to identify formatted dates and times; however, the expression restricted all components to be digits, so dates with alphanumeric characters were not captured. Editing the regular expression to allow for alphanumeric characters fixed the capturing issue, but resulted in an error downstream where other methods expected a numeric month to be returned. Ultimately, a custom function was written to convert months represented as text to integers as existing conversion packages were not versatile enough to accommodate all lexical variations of these entities.

Similarly, hour and minute formats such as "5:45 PM" were not being recognized correctly because Chrono's regular expression looked specifically for the format found in the Newswire corpus that contained seconds (hh:mm:ss). Debugging formatted time expressions proved to be a challenge because Chrono utilizes three different modules to parse out this data. First, a module to identify the hours, minutes, and seconds, followed by a module to identify AMPM entities, and finally, a module to link sub-intervals where both MinuteOfHour and AMPM entities are sub-intervals of HourOfDay. Interestingly, the performance of HourOfDay for the Span Only evaluation had an F1 score of 0.941 both before and after improvements, indicating that Chrono was actually identifying most of the hours correctly, but was missing specific SCATE properties.

Punctuation - To Include or Not to Include? Part of the HourOfDay parsing issue stemmed from temporal phrases at the end of a sentence, such as "2:04 AM.", where the period ended up being part of the "AM" string. Initially, Chrono looked for AMPM entities without considering punctuation unlike the MonthOfYear parsing, which specifically accounts for punctuation such as "Dec.". Thus, the "AM." in the example was never identified, so the HourOfDay entity "2" would be lacking the subinterval link to the AMPM entity. To resolve this, Chrono was modified to utilize regular expressions in parsing out AMPM entities with and without surrounding punctuation.

One dilemma arose when considering the variants of an AMPM entity. For example, valid AMPM entity strings include "AM", "am", "A.M.", and "a.m."; however, "AM." may not be considered a valid representation of an AMPM entity. Thus, Chrono specifically includes the period in the span only if there is a period after each letter in strings (e.g. "A.M."), otherwise, the period is not included in the span. Implementing this fix resulted in a significant performance improvement for the AMPM entity and, oddly, a decrease in HourOfDay performance.

Where have the Minutes Gone? While the HourOfDay entity was performing well in the Span Only evaluation, the MinuteOfHour entity performed poorly in both Span Only and 100% Correct Entity evaluations. This was a result of Chrono looking for an HourOfDay in two different methods–one that identified formatted times and another that first looked for an AMPM entity and, if found, searched for an upstream HourOfDay. The majority of time expressions in THYME were formatted as "hh:mm" followed by an "AM" or "PM" which resulted in HourOfDay being identified by AMPM parsing and not the formatted time method. The AMPM method was designed to identify the pattern found frequently in Newswire texts (e.g. "5 PM"), which doesn't include second or minute parsing. To fix this issue the formatted time method was adjusted to allow for the "hh:mm" format, so now the HourOfDay and MinuteOfHour entities are being identified and appropriate sub-intervals are annotated. However, this code improvement resulted in another decrease in performance of the HourOfDay entity.

Too Many Hours of the Day! The expected result of fixing the AMPM entity and formatted time parsing was increased performance on AMPM, MinuteOfHour, and HourOfDay entity parsing because the AMPM and MinuteOfHour sub-interval links were now identified correctly. However, HourOfDay performance actually became worse due to predicting too many HourOfDay entities. Further investigation revealed that every temporal phrase that included an AMPM entity had duplicate HourOfDay entities annotated (the same hour was annotated twice), one with the correct AMPM and MinuteOfHour sub-interval links and the other with no sub-interval links. This issue stemmed from a combination of the hierarchical parsing of formatted dates/times and inadvertently excluding a check to see if an HourOfDay entity already existed when parsing AMPM entities.

In Chrono, all temporal phrases are interrogated by all modules. To ensure only one entity of each type is identified in each temporal phrase Chrono implements a flag system. For example, in the phrase “Monday at 3:05 PM.” there is one DayOfWeek, one HourOfDay, one MinuteOfHour, and one AMPM entity. This phrase is first parsed by the formatted date/time module to identify the HourOfDay “3” and the MinuteOfHour “05” entity. Following is the identification of the “PM” AMPM entity; however, if this module finds an AMPM entity it then proceeds to look for an HourOfDay entity preceeding the AMPM substring. However, an HourOfDay had already been identified, and the AMPM module neglected checking this. Fixing this double parsing issue was straightforward as the AMPM module just needed to check if the HourOfDay flag had been set for the given temporal phrase. This error resulted in some initially puzzling results where the HourOfDay performance kept decreasing with every “improvement”, and ended up identifying twice as many HourOfDay entities as it should have. Different modules may be required for parsing different date/time formats, so it is important to ensure that all modules are consistently coded. It is also important to keep in mind that some formats are more frequent in one domain than another. This issue had not appeared when using the Newswire corpus because the majority of the AMPM entities were accompanied by the shorter format of “5 PM”, or contained the full “hh:mm:ss” format, whereas in the clinical domain the specification of hour and minutes, such as “3:05 PM”, was ubiquitous throughout the corpus.

Stop words splitting temporal phrases: Chrono was initially unable to handle stop words that connected temporal entities into a single phrase, which limited its performance on the THYME corpus due to the use of long temporal expressions in clinical texts. Chrono identified temporal phrases by looking for consecutive temporal and/or numeric tokens. If a stop word was identified (e.g. “is”, “of”, “at”, etc), the temporal phrase would be terminated–in some cases prematurely. For example, the phrase “beginning of this month on September 1” was originally separated into 3 temporal phrases: “beginning”, “this month”, and “September 1”. Other examples of temporal phrases that were incorrectly split include “2005 in April” and “October 14, 2010 at 02:07 PM”, which were both separated into two phrases. While individual temporal entities were identified correctly, the correct sub-intervals for each entity were unable to be assigned because Chrono only links sub-intervals within a single phrase. To fix this, code was added to tag “linking” words in the temporal phrase extraction module. Now, if a linking token is identified while constructing a temporal phrase it is ignored and the phrase is extended. This allows Chrono to correctly identify longer temporal phrases and results in correct assignment of sub-intervals, which brought the 100% Entity performance closer to Span Only.

Unexpected Effects of Longer Temporal Phrases: The inclusion of stop words in temporal phrases was a major upgrade to Chrono resulting in sub-intervals of longer phrases being correctly assigned. However, this had an unintended result that initially lowered the overall F1 scores for Calendar-Interval and Period entities. Investigating changes in performance revealed Calendar-Interval and Period entities that were correct were now incorrectly annotated with a link to a Number entity. This happened for phrases like “four times a day” or “one time a day”, which are highly frequent expressions in clinical notes as they are part of instructions for taking medications. This behavior resulted from Chrono’s parsing strategy for identifying associated numbers with SCATE entities where Chrono naively looked for a number token in the sub-string of characters preceding an annotated entity. This parsing strategy worked well for Newswire text as the majority of associated numbers appeared in formats similar to “2 weeks ago”, or “5 days”. Previously, Chrono assigned expressions like “four times a day” to two temporal phrases: “four times” and “day”. Thus, the Calendar-Interval “day” was correctly identified with no Number link. After including the stop words in the temporal phrases the first number in the phrase (e.g. “four”) was incorrectly associated with the Period or Calendar-Interval entity. Chrono’s number parsing strategy also became an issue with other frequent clinical phrases such as “one-time daily” where the number “one” was incorrectly associated with the Calendar-Interval “daily”. To fix this issue, Chrono’s definition of where a number had to be located in order to be linked to a SCATE entity was restricted to the immediately preceding token instead of the full preceding sub-string. This restriction works well for the THYME and Newswire corpora; however, may not work well with expressions such as “2 full weeks from now” where the Period “weeks” should be annotated with the Number “2”.

6.2.6 Document Design

Sentence Boundaries: An interesting temporal parsing issue appears in clinical texts regarding sentence tokenization due to item lists in the clinical record. Initially, Chrono did not tokenize on sentences as temporal phrases spanning sentence boundaries were not an issue in the Newswire corpus. However, clinical records in the THYME corpus contained entries like the following:

> “...my notes from December.
>
> 2. Ulcerative colitis...”

Where the top sentence ends with the temporal entity “December” followed by a numbered list item. Since Chrono did not consider sentence boundaries, this line break was removed in the preprocessing phase and the “2” that numbers the list item was parsed as a DayOfMonth associated with “December”. To resolve this issue, Chrono was updated to identify sentence boundaries. In Temporal Phrase Extraction, Chrono no longer allows a single temporal phrase to span sentence boundaries; however, the Temporal Disambiguation module still ignores these boundaries.

Metadata: Domain agnostic rules and procedures can be developed to identify many temporal expressions in written text, but metadata presents additional challenges in that it is inherently domain-specific, and can even be document type specific within the same domain. For example, pathology reports and clinical encounters with a physician can have their metadata formatted in different ways. In dealing with metadata the first question is if one wants to parse the metadata at all. A good reason to do so would be to gather contextual information that is not explicitly written in the text, like identifying the document creation date to disambiguate references to days of the week, etc. The gold standard SCATE annotations do contain dates from the metadata sections, so it is necessary for Chrono to identify these entities. Two issues arose when working on this problem: 1) How to identify a temporal token using whitespace tokenization when the metadata line contains little whitespace, and 2) whether or not to include the word “date” as a temporal token.

In the THYME corpus metadata is formatted as “[start_date=12/02/2010, rev_date=12/02/2010]”. Using whitespace tokenization this line is split into two tokens–both marked as temporal as they contain formatted date strings. However, in the Temporal Phrase Extraction module this line is considered a single phrase because it is composed of two consecutive temporal tokens. This causes an issue as Chrono assumes there is only one of each SCATE entity type in a phrase; thus, initially Chrono only annotated one of the two dates in the metadata line. To resolve this, Chrono now converts all equal signs to spaces prior to whitespace tokenization, thereby separating the metadata text to four tokens. While this fix resolved the issue of parsing metadata dates, an equal sign could be useful information, so a more sophisticated approach will be required in the future.

The second issue with parsing metadata information arose when updating the lexicon of known temporal tokens. The word “date” is temporal, but had not been included in the initial lexicon of Chrono. Including “date” as a temporal token resulted in identifying the metadata line as a single temporal phrase again as it was now a consecutive sequence of four temporal tokens: “start_date”, “12/02/2010”, “rev_date”, and “12/02/2010”. As “start_date” and “rev_date” are just labels they should not be considered temporal entities. Some mentions of “date” were valid temporal expressions, but there were few of them. Thus, we decided to continue to exclude this token. To be applicable to different domains, more sophisticated methods to parse metadata will need to be implemented to resolve issues with temporal labels and other special characters seen in metadata text.

6.3 Improved Performance

Improvements made to Chrono using the THYME Training Corpus lead to a 0.27 and 0.24 increase in precision and recall, respectively, with a 0.26 increase in F1 measure for the Evaluation Corpus (Table 3). This resulted in Chrono being the top performing system for SCATE Normalization. Chrono’s performance on the Training Corpus improved similarly with a precision of 0.881 in the Span Only evaluation and 0.729 for the 100% Correct Entity. This indicates that Chrono is identifying the correct location of many entities, but it is having trouble setting all the properties correctly.

When designing a rule-base system it is possible to develop rules that overfit or are tailored to the training corpus (i.e. Newswire texts). Overfitting rules results in good performance on the

Dataset	System	Precision	Recall	F1
THYME Eval	Chrono	0.76	0.51	0.61
THYME Eval	Laparra et. al.	0.52	0.63	0.57
Newswire Eval	Chrono	0.57	0.54	0.55
Newswire Eval	Laparra et. al.	0.58	0.46	0.51
THYME Train	Chrono 100%	0.729	0.478	0.578
THYME Train	Chrono Span Only	0.881	0.575	0.696

Table 3: Improved performance on THYME Corpora using SVM, excluding "Event".

training domain and poor performance on the testing domain, similar to Chrono's performance on the THYME corpus. However, when rules are adjusted to incorporate another domain it is expected that the performance in the training domain go down, indicating that it was overfitting the training domain. To see if this happened with Chrono, we re-evaluated our final model on the Newswire corpus. The results showed an insignificant 0.01 drop in F1 due to a 0.05 drop in Precision and a 0.04 increase in Recall, which indicates that Chrono is now more compatible with cross-domain application. Since we do not see a major drop in performance on the Newswire corpus we can conclude the original rules did not overfit the Newswire domain, but rather they were incomplete and required expansion to improve performance in the clinical domain.

7 Conclusions and Future Work

In conclusion, clinical domain texts posed additional challenges that were either not present in the Newswire corpus, or not frequent enough to prioritize highly when initially building Chrono. Application to the THMYE Training Corpus brought these limitations to light, such as the consistent use of temporal expressions that utilize frequency, highly repeated temporal phrases, dosage values being annotated as temporal expressions, and additional lexical elements. As temporal information is relatively domain agnostic, improvements made to Chrono for THYME should improve performance on other domains. An advantage of utilizing clinical texts is that it encounters a variety of writing styles from different practitioners who may prefer specifying temporal information in different ways. Additionally, different medical forms, such as pathology reports versus clinical notes, have specific ways to convey dates. Thus, the range of temporal expressions Chrono now identifies has been significantly expanded due to the variety incorporated in the clinical texts.

While Chrono's performance on SCATE Normalization has improved, there are still many areas for further development. These include identifying frequency, disambiguating dosage versus 4-digit year or 24-hour time, implementing more sophisticated approaches to parsing metadata, and performing a more detailed investigation at the entity level to identify which SCATE properties are being missed or incorrectly assigned in order to bring the 100% Correct Entity performance closer to the Span Only performance. These updates will require the implementation of additional rule-sets as well as the addition of machine learning modules and more complex contextual parsing. One approach to augmenting current rule sets is the automated generation of regular expressions (Redd et al., 2015) based on annotated gold standards, which has the potential to expand Chrono's capabilities without time-consuming human review of missed expressions. Finally, Chrono outputs normalized temporal expressions in the SCATE schema format, which limits our ability to evaluate its performance on corpora in other domains. Currently, only select subsets of the AQUAINT and THYME corpora are annotated with SCATE, and the complete conversion of TimeML to the SCATE schema is difficult as TimeML lacks details required by SCATE. Thus, implementation of a method to convert SCATE XML to the standard TimeML format will allow Chrono to be evaluated on additional cross-domain corpora and classic benchmark temporal corpora such as i2b2 (Sun et al., 2013a), TempEval (Verhagen et al., 2007, 2010; UzZaman et al., 2013), and Clinical TempEval (Bethard et al., 2015).

The process of improving Chrono brought to light several aspects of cross-domain application of temporal parsing: 1) lexical differences, 2) the frequency of temporal entity usage, 3) disambiguating numerical phrases, 4) appropriate machine learning data, 5) lexical variation of concepts, and 6) differences in document structure. While the concept of time is the same regardless of the domain, its representation can vary. Thus, temporal parsing provides a good backdrop for determining the challenges of cross-domain application, which is difficult for many NLP applications. The aspects of cross-domain application discussed herein provide a foundation for designing adaptable NLP tools that can be utilized across domains.

References

Gabor Angeli, Christopher Manning, and Daniel Jurafsky. 2012. Parsing Time: Learning to Interpret Time Expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 446–455, Montral, Canada. Association for Computational Linguistics.

Steven Bethard. 2013. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14, Atlanta, Georgia, USA. Association for Computational Linguistics.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 Task 6: Clinical TempEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 806–814, Denver, Colorado. Association for Computational Linguistics.

Steven Bethard and Jonathan Parker. 2016. A Semantically Compositional Annotation Scheme for Time Normalization. In *LREC*.

Nathanael Chambers. 2013. NavyTime: Event and Time Ordering from Raw Text. Technical report, NAVAL ACADEMY ANNAPOLIS MD.

Angel X. Chang and Christopher D. Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740.

Wei-Te Chen and Will Styler. 2013. Anafora: A Web-based General Purpose Annotation Tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia. Association for Computational Linguistics.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. ManTIME: Temporal expression identification and normalization in the TempEval-3 challenge. *arXiv:1304.7942 [cs]*. ArXiv: 1304.7942.

David Graff. 2002. The AQUAINT Corpus of English News Text LDC2002t31. Philadelphia: Linguistic Data Consortium, 2002.

Aleksandar Kovaevi, Azad Dehghan, Michele Filannino, John A. Keane, and Goran Nenadic. 2013. Combining rules and machine learning for extraction of temporal expressions and events from clinical narratives. *Journal of the American Medical Informatics Association*, 20(5):859–866.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. SemEval 2018 Task 6: Parsing Time Normalizations. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 88–96, New Orleans, Louisiana. Association for Computational Linguistics.

Hee-Jin Lee, Yaoyun Zhang, Jun Xu, Cui Tao, Hua Xu, and Min Jiang. 2017. Towards practical temporal relation extraction from clinical notes: An analysis of direct temporal relations. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1272–1275.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent Semantic Parsing for Time Expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Baltimore, Maryland. Association for Computational Linguistics.

Hector Llorens, Leon Derczynski, Robert Gaizauskas, and Estela Saquete. 2012. TIMEN: An Open Temporal Expression Normalisation Resource. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA). Event-place: Istanbul, Turkey.

Pawet Mazur and Robert Dale. 2010. WikiWars: A New Corpus for Research on Temporal Expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 913–922, Stroudsburg, PA, USA. Association for Computational Linguistics. Event-place: Cambridge, Massachusetts.

Stephane M. Meystre, Guergana K. Savova, Karin C. Kipper-Schuler, and John F. Hurdle. 2008. Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research. *Yearbook of Medical Informatics*, 17(1):128–144.

Amy Olex, Luke Maffey, Nicholas Morgan, and Bridget McInnes. 2018. Chrono at SemEval-2018 Task 6: A System for Normalizing Temporal Expressions. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 97–101, New Orleans, Louisiana. Association for Computational Linguistics.

James Pustejovsky, Patrick Hanks, Roser Saur, Andrew See, R Gaizauskas, A Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics*, pages 647–656.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of the Seventh International Confer- ence on Language Resources and Evaluation (LREC10)*, page 4, Valletta, Malta.

Douglas Redd, YiJun Shaoa, Jing Yang, Guy Divita, and Qing Zeng-Treitler. 2015. Automated Learning of Temporal Expressions. *Studies in Health Technology and Informatics*, 216:639–642.

Sunghwan Sohn, Kavishwar B. Wagholikar, Dingcheng Li, Siddhartha R. Jonnalagadda, Cui Tao, Ravikumar Komandur Elayavilli, and Hongfang Liu. 2013. Comprehensive temporal information detection from clinical text: medical events, time, and TLINK identification. *Journal of the American Medical Informatics Association*, 20(5):836–842.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 321–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

William F. Styler, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, and James Pustejovsky. 2014. Temporal Annotation in the Clinical Domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013a. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. *Journal of the American Medical Informatics Association : JAMIA*, 20(5):806–813.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013b. Temporal reasoning over clinical text: the state of the art. *Journal of the American Medical Informatics Association*, 20(5):814–819.

Hegler Tissot, Angus Roberts, Leon Derczynski, Genevieve Gorrell, and Marcus Didonet Del Fabro. 2015. Analysis of temporal expressions annotated in clinical notes. In *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11)*.

Naushad UzZaman and James Allen. 2010. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 276–283, Uppsala, Sweden. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen, Inderjeet Mani, Roser Sauri, Robert Knippen, Seok Bae Jang, Jessica Littman, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating Temporal Annotation with TARSQI. In *Proceedings of the ACL 2005 on Interactive Poster and Demonstration Sessions*, ACLdemo '05, pages 81–84, Stroudsburg, PA, USA. Association for Computational Linguistics. Event-place: Ann Arbor, Michigan.

Marc Verhagen, Roser Saur, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yan Xu, Yining Wang, Tianren Liu, Junichi Tsujii, and Eric I.-Chao Chang. 2013. An end-to-end system to identify temporal relation in discharge summaries: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):849–858.

Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 420–429, Vancouver, Canada. Association for Computational Linguistics.

Document-Level N-ary Relation Extraction with Multiscale Representation Learning

Robin Jia[1*] **Cliff Wong**[2] **Hoifung Poon**[2]
[1] Stanford University, Stanford, California, USA
[2] Microsoft Research, Redmond, Washington, USA
`robinjia@cs.stanford.edu, {cliff.wong,hoifung}@microsoft.com`

Abstract

Most information extraction methods focus on binary relations expressed within single sentences. In high-value domains, however, n-ary relations are of great demand (e.g., drug-gene-mutation interactions in precision oncology). Such relations often involve entity mentions that are far apart in the document, yet existing work on cross-sentence relation extraction is generally confined to small text spans (e.g., three consecutive sentences), which severely limits recall. In this paper, we propose a novel multiscale neural architecture for document-level n-ary relation extraction. Our system combines representations learned over various text spans throughout the document and across the subrelation hierarchy. Widening the system's purview to the entire document maximizes potential recall. Moreover, by integrating weak signals across the document, multiscale modeling increases precision, even in the presence of noisy labels from distant supervision. Experiments on biomedical machine reading show that our approach substantially outperforms previous n-ary relation extraction methods.

1 Introduction

Knowledge acquisition is a perennial challenge in AI. In high-value domains, it has acquired new urgency in recent years due to the advent of big data. For example, the dramatic drop in genome sequencing cost has created unprecedented opportunities for tailoring cancer treatment to a tumor's genetic composition (Bahcall, 2015). Despite this potential, operationalizing personalized medicine is difficult, in part because it requires painstaking curation of precision oncology knowledge from biomedical literature. With tens of millions of papers on PubMed, and thousands more added every day,[1] we are sorely in need of automated methods to accelerate manual curation.

> "*We next expressed ALK F1174L, ALK F1174L/L1198P, ALK F1174L/G1123S, and ALK F1174L/G1123D in the original SH-SY5Y cell line.*"
>
> (. . . 15 sentences spanning 3 paragraphs . . .)
>
> "*The 2 mutations that were only found in the neuroblastoma resistance screen (G1123S/D) are located in the glycine-rich loop, which is known to be crucial for ATP and ligand binding and are the first mutations described that induce resistance to TAE684, but not to PF02341066.*"

Figure 1: Two examples of drug-gene-mutation relations from a biomedical journal paper. The relations are expressed across multiple paragraphs, requiring document-level extraction.

Prior work in machine reading has made great strides in sentence-level binary relation extraction. However, generalizing extraction to n-ary relations poses new challenges. Higher-order relations often involve entity mentions that are far away in the document. Recent work on n-ary relation extraction has begun to explore cross-sentence extraction (Peng et al., 2017; Wang and Poon, 2018), but the scope is still confined to short text spans (e.g., three consecutive sentences), even though a document may contain hundreds of sentences and tens of thousands of words. While this already increases the yield compared to sentence-level extraction, it still misses many relations. For example, in Figure 1, the drug-gene-mutation relations between `PF02341066`, `ALK`, `G1123S(D)` (`PF02341066` can treat cancers with mutation `G1123S(D)` in gene `ALK`) can only be extracted by substantially expanding the scope. High-value information, such as latest medical findings, might only be mentioned once in the corpus. Maximiz-

---

[*] Work done as an intern at Microsoft Research.

[1] `ncbi.nlm.nih.gov/pubmed`

Proceedings of NAACL-HLT 2019, pages 3693–3704
Minneapolis, Minnesota, June 2 - June 7, 2019.

ing recall is thus of paramount importance.

In this paper, we propose a novel multiscale neural architecture for document-level n-ary relation extraction. By expanding extraction scope to the entire document, rather than restricting relation candidates to co-occurring entities in a short text span, we ensure maximum potential recall. To combat the ensuing difficulties in document-level extraction, such as low precision, we introduce multiscale learning, which combines representations learned over text spans of varying scales and for various subrelations (Figure 2). This approach deviates from past methods in several key regards.

First, we adopt an entity-centric formulation by making a single prediction for each entity tuple occurring in a document. Previous n-ary relation extraction methods typically classify individual mention tuples, but this approach scales poorly to whole documents. Since each entity can be mentioned many times in the same document, applying mention-level methods leads to a combinatorial explosion of mention tuples. This creates not only computational challenges but also learning challenges, as the vast majority of these tuples do not express the relation. Our entity-centric formulation alleviates both of these problems.

Second, for each candidate tuple, prior methods typically take as input the contiguous text span encompassing the mentions. For document-level extraction, the resulting text span could become untenably large, even though most of it is unrelated to the relation of interest. Instead, we allow discontiguous input formed by multiple discourse units (e.g., sentence or paragraph) containing the given entity mentions.

Finally, while an n-ary relation might not reside within a discourse unit, its subrelations might. In Figure 1, the paper first mentions a gene-mutation subrelation, then discusses a drug-mutation subrelation in a later paragraph. By including subrelations in our modeling, we can predict n-ary relations even when all n entities never co-occur in the same discourse unit.

With multiscale learning, we turn the document view from a challenge into an advantage by combining weak signals across text spans and subrelations. Following recent work in cross-sentence relation extraction, we conduct thorough evaluation in biomedical machine reading. Our approach substantially outperforms prior n-ary relation extraction methods, attaining state-of-the-art results on a large benchmark dataset recently released by a major cancer center. Ablation studies show that multiscale modeling is the key to these gains.[2]

2 Document-Level N-ary Relation Extraction

Prior work on relation extraction typically formulates it as a mention-level classification problem. Let $e_1, \ldots, e_n$ be entity mentions that co-occur in a text span T. Relation extraction amounts to classifying whether a relation R holds for $e_1, \ldots, e_n$ in T. For the well-studied case of binary relations within single sentences, $n = 2$ and T is a sentence.

In high-value domains, however, there is increasing demand for document-level n-ary relation extraction, where $n > 2$ and T is a full document that may contain hundreds of sentences. For example, a molecular tumor board needs to know if a drug is relevant for treating cancer patients with a certain mutation in a given gene. We can help the tumor board by extracting such ternary interactions from biomedical articles. The mention-centric view of relation extraction does not scale well to this general setting. Each of the n entities may be mentioned many times in a document, resulting in a large number of candidate mention tuples, even though the vast majority of them are irrelevant to the extraction task.

In this paper, we adopt an *entity-centric* formulation for document-level n-ary relation extraction. We use upper case for entities ($E_1, \cdots, E_n$) and lower case for mentions ($e_1, \cdots, e_n$). We define an n-ary relation candidate to be an $(n+1)$-tuple $(E_1, \ldots, E_n, T)$, where each entity E_i is mentioned at least once in the text span T. The relation extraction model is given a candidate $(E_1, \ldots, E_n, T)$ and outputs whether or not the tuple expresses the relation R.[3] Deciding what information to use from the various entity mentions within T is now a modeling question, which we address in the next section.

3 Our Approach: Multiscale Representation Learning

We present a general framework for document-level n-ary relation extraction using multiscale

---

[2] Our code and data will be available at `hanover.azurewebsites.net`

[3] It is easy to extend our approach to situations where k mutually exclusive relations $R_1, \ldots, R_k$ must be distinguished, resulting in a $(k+1)$-way classification problem.

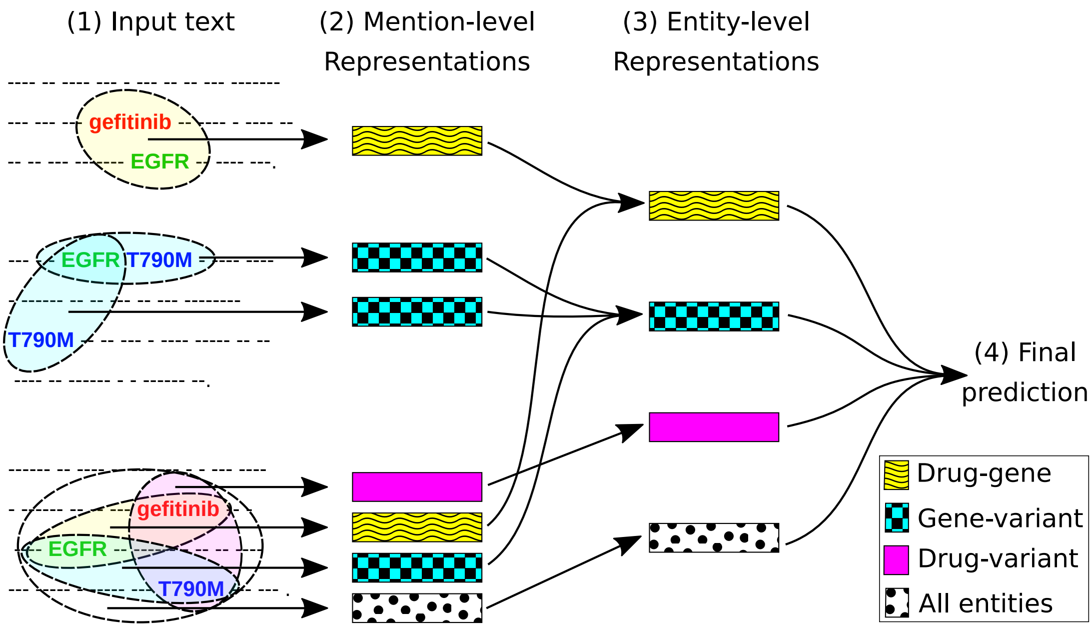


Figure 2: Multiscale representation learning for document-level n-ary relation extraction, an entity-centric approach that combines mention-level representations learned across text spans and subrelation hierarchy. (1) Entity mentions (e.g., gefitinib, a drug; EGFR, a gene; T790M, a variant) are identified from text, and mentions that co-occur within a discourse unit (e.g., paragraph) are isolated. (2) Within each discourse unit, mention-level representations are computed for each tuple of entity mentions. These representations may correspond to the entire n-ary relation or subrelations over subsets of entities (drug-variant, drug-gene, gene-variant). (3) At the document scale, mention-level representations for both the n-ary relation and its subrelations are combined into entity-level representations. (4) Entity-level representations are used to predict the relation.

representation learning. Given a document with text T and entities $E_1, \ldots, E_n$, we first build mention-level representations for groups of these entities whenever they co-occur within the same discourse unit. We then aggregate these representations across the whole document, yielding entity-level representations for each subset of entities. Finally, we predict whether $E_1, \ldots, E_n$ participate in the relation based on the concatenation of these entity-level representations. These steps are depicted in Figure 2.

3.1 Mention-level Representation

Let the full document T be composed of discourse units $T_1, \ldots, T_m$ (e.g., different paragraphs). Let T_j be one such discourse unit, and suppose $e_1, \ldots, e_n$ are entity mentions of $E_1, \ldots, E_n$ that co-occur in T_j. We construct a contextualized representation for mention tuple $(e_1, \ldots, e_n)$ in T_j. In this paper, we use a standard approach by applying a bi-directional LSTM (BiLSTM) to T_j, concatenating the hidden states for each mention, and feeding this through a single-layer neural network. We denote the resulting vector as $r(R, e_1, \ldots, e_n, T_j)$ for the relation R.

3.2 Entity-level Representation

Let $M(R, E_1, \ldots, E_n, T)$ denote the set of all mention tuples $(e_1, \ldots, e_n)$ and discourse units T_j within T such that each e_i appears in T_j. We can create an entity-level representation $r(R, E_1, \ldots, E_n, T)$ of the n entities by combining mention-level representations using an aggregation operator $\mathcal{C}$:

$$\underset{(e_1,\ldots,e_n,T_j)\in M(R,E_1,\ldots,E_n,T)}{\mathcal{C}} r(R, e_1, \ldots, e_n, T_j)$$

A standard choice for $\mathcal{C}$ is max pooling, which works well if it is pretty clear-cut whether a mention tuple expresses a relation. In practice, however, the mention tuples could be ambiguous and less than certain individually, yet collectively express a relation in the document. This motivates us to experiment with logsumexp, the smooth version of max, where

$$\mathrm{logsumexp}(x_1, \ldots, x_k) = \log \sum_{i=1}^{k} \exp(x_i).$$

This facilitates accumulating weak signals from individual mention tuples, and our experiments show that it substantially improves extraction accuracy compared to max pooling.

3.3 Subrelations

For higher-order relations (i.e., larger n), it is less likely that they will be completely contained within a discourse unit. Often, the relation can be decomposed into subrelations over subsets of entities, each of which is more likely to be expressed in a single discourse unit. This motivates us to construct entity-level representations for subrelations as well. The process is straightforward. Let R_S be the $|S|$-ary subrelation over entities $E_{S_1}, \cdots, E_{S_{|S|}}$, where $S \subseteq \{1, \dots, n\}$ and $|S|$ denotes its size. We first construct mention-level representations $r(R_S, e_{S_1}, \cdots, e_{S_{|S|}}, T)$ for R_S and its relevant entity mentions, then combine them into an entity-level representation $r(R_S, E_{S_1}, \cdots, E_{S_{|S|}}, D)$ using the chosen aggregation operator $\mathcal{C}$. We do this for every $S \subseteq \{1, \dots, n\}$ with $|S| \geq 2$ (including the whole set, which corresponds to the full relation R). This gives us an entity-level representation for each subrelation of arity at least 2, or equivalently, each subset of entities of size at least 2.

3.4 Relation Prediction

To make a final prediction, we first concatenate all of the entity-level representations $r(R_S, E_{S_1}, \dots, E_{S_{|S|}}, D)$ for all $S \subseteq \{1, \dots, n\}$ with $|S| \geq 2$. The concatenated representation is fed through a two-layer feedforward neural network followed by a softmax function to predict the relation type.

It is possible that for some subrelations R_S, all $|S|$ entities do not co-occur in any discourse unit. When this happens, we set $r(R_S, E_{S_1}, \dots, E_{S_{|S|}})$ to a bias vector which is learned separately for each R_S. This ensures that the concatenation is done over a fixed number of vectors, e.g., 4 for a tenary relation (three binary subrelations and the main relation). Importantly, this strategy enables us to make meaningful predictions for relation candidates even if all n entities never co-occur in the same discourse unit; such candidates would never be generated by a system that only looks at single discourse units in isolation.

3.5 Document Model

Our document model is actually a family of representation learning methods, conditioned on the choice of discourse units, subrelations, and aggregation operators. In this paper, we consider sentences and paragraphs as possible discourse units. We explore $\max$ and $\operatorname{logsumexp}$ as aggregation operators. Moreover, we explore ensemble prediction as an additional aggregation method. Specifically, we learn a restricted multiscale model by limiting the text span to a single discourse unit (e.g., a paragraph); the model still combines representations across mentions and subrelations. At test time, given a full document with m discourse units, we obtain independent predictions $p_1, \dots, p_m$ for each discourse unit. We then combine these probabilities using an ensemble operator $\mathcal{P}$. A natural choice for $\mathcal{P}$ is max, though we also experiment with noisy-or:

$$\mathcal{P}(p_1, \cdots, p_k) = 1 - \prod_{i=1}^{k} (1 - p_i).$$

It is also possible to ensemble multiple models that operate on different discourse units, using this same operator.

Our model can be trained using standard supervised or indirectly supervised methods. In this paper, we focus on distant supervision, as it is a particularly potent learning paradigm for high-value domains. Our entity-centric formulation is particularly well aligned with distant supervision, as distant supervision at the entity level is significantly less noisy compared to the mention level, so we don't need to deploy sophisticated denoising strategies such as multi-instance learning (Hoffmann et al., 2011).

	Sentence level	Paragraph level	Document level
Text Units	2,326	3,687	3,362
Pos. Examples	2,222	4,906	8,514
Neg. Examples	2,849	13,371	323,584

Table 1: Statistics of our training corpus using PMC-OA articles and distant supervision from CIVIC, GDKD, and OncoKB. "Text Units" refers to the number of distinct sentences, paragraphs, and documents that contain a candidate triple of drug, gene, mutation.

	Development	Test
Documents	118	225
Annotated facts	701	1,324
Paragraphs per document	101	105
Sentences per document	314	320
Words per document	6,871	7,010

Table 2: Statistics of the CKB evaluation corpus.

4 Experiments

4.1 Biomedical Machine Reading

We validate our approach on a standard biomedical machine reading task: extracting drug-gene-mutation interactions from biomedical articles (Peng et al., 2017; Wang and Poon, 2018). We cast this task as binary classification: given a drug, gene, mutation, and document in which they are mentioned, determine whether the document asserts that the mutation in the gene affects response to the drug. For training, we use documents from the PubMed Central Open Access Subset (PMC-OA)[4]. For distant supervision, we use three existing knowledgebases (KBs) with hand-curated drug-gene-mutation facts: CIVIC,[5] GDKD (Dienstmann et al., 2015), and OncoKB (Chakravarty et al., 2017). Table 1 shows basic statistics of this training data. Past methods using distant supervision often need to up-weight positive examples, due to the large proportion of negative candidates. Interestingly, we found that our document model was robust to this imbalance, as re-weighting had little effect and we didn't use it in our final results.

Evaluating distant supervision methods is challenging, as there is often no gold-standard test set, especially at the mention level. Prior work thus resorts to reporting sample precision (estimated proportion of correct system extractions) and absolute recall (estimated number of correct system extractions). This requires subsampling extraction results and manually annotating them. Subsampling variance also introduces noise in the estimate.

Instead, we used CKB CORE™, a public subset of the Clinical Knowledgebase (CKB)[6] (Patterson et al., 2016), as our gold-standard test set. CKB CORE™ contains document-level annotation of drug-gene-mutation interactions manually curated by The Jackson Laboratory (JAX), an NCI-designated cancer center. It is a high-quality KB containing facts from a few hundred PubMed articles for 86 genes, with minimal overlap with the three KBs we used for distant supervision. To avoid contamination, we removed CKB entries whose documents were used in our training data, and split the rest into a development and test set. See Table 2 for statistics. We tuned hyperparameters and thresholds on the development set, and report results on the test set.

4.2 Implementation Details

We conducted standard preprocessing and entity linking, similar to Wang and Poon (2018) (see Section A.1). Following standard practice, we masked all entities of the same type with a dummy token, to prevent the classifier from simply memorizing the facts in distant supervision. Wang and Poon (2018) observed that many errors stemmed from incorrect gene-mutation association. We therefore developed a simple rule-based system that predicts which gene-mutation pairs are valid (see Section A.2). We removed candidates that contained a gene-mutation pair that was not predicted by the rule-based system.

4.3 Main Results

We evaluate primarily on area under the precision recall curve (AUC).[7] We also report maximum recall, which is the fraction of true facts for which a candidate was generated. Finally, we report precision, recall, and F1, using a threshold tuned to maximize F1 on the CKB development set.

We compared our multiscale system (MULTISCALE) with three restricted variants (SENTLEVEL, PARALEVEL, DOCLEVEL). SENTLEVEL and PARALEVEL restricted training and prediction to single discourse units (i.e., sentences and paragraphs), and produced a document-level prediction by applying the ensemble operator over individual discourse units. DOCLEVEL takes the whole document as input, with each paragraph as a discourse unit. MULTISCALE further combined SENTLEVEL, PARALEVEL, and DOCLEVEL using the ensemble operator. For additional details about the models, see Section A.3. We also compared MULTISCALE with DPL (Wang and Poon, 2018), the prior state of the art in cross-sentence n-ary relation extraction. DPL classifies drug-gene-mutation interactions within three consecutive sentences using the same model architecture as Peng et al. (2017), but incorporates additional indirect supervision such as data programming and joint inference. We used the DPL code from the authors and produced a document-level prediction similarly using the ensemble operator. In the base version, we used max as the ensemble operator. We also evaluated the effect when we

---

[4] `www.ncbi.nlm.nih.gov/pmc`

[5] `civic.genome.wustl.edu`

[6] `ckbhome.jax.org`

[7] We compute area using average precision, which is similar to a right Riemann sum. This avoids errors introduced by the trapezoidal rule, which may overestimate area.

System	AUC	Max Recall	Precision	Recall	F1
Base versions					
DPL	24.4	53.8	27.3	42.3	33.2
SentLevel	22.4	36.6	39.3	34.7	36.9
ParaLevel	33.1	58.9	36.5	44.6	40.1
DocLevel	36.7	**79.0**	45.4	38.5	41.7
MultiScale	**37.3**	**79.0**	41.8	43.3	**42.5**
+ Noisy-Or					
DPL	31.5	53.8	33.3	41.5	36.9
SentLevel	25.3	36.6	39.3	35.3	37.2
ParaLevel	35.6	58.9	44.3	40.6	42.4
DocLevel	36.7	**79.0**	45.4	38.5	41.7
MultiScale	**39.7**	**79.0**	48.1	38.9	**43.0**
+ Noisy-Or + Gene-mutation filter					
DPL	39.1	52.6	50.5	47.8	49.1
SentLevel	29.0	35.5	63.3	34.2	44.4
ParaLevel	42.1	57.2	50.6	50.7	50.7
DocLevel	42.9	**74.4**	49.3	46.6	47.9
MultiScale	**47.5**	**74.4**	52.6	53.0	**52.8**

Table 3: Comparison of our multiscale system with restricted variants and DPL (Wang and Poon, 2018) on CKB.

used noisy-or as the ensemble operator, as well as when we applied the gene-mutation filter during postprocessing.

Table 3 shows the results on the CKB test set. In all scenarios, our full model (MultiScale) substantially outperforms the prior state-of-the-art system (DPL). For example, in the best setting, using both noisy-or and the gene-mutation filter, the full model improves over DPL by 8.4 AUC points. Multiscale learning is the key to this performance gain, with MultiScale substantially outperforming more restricted variants. Not surprisingly, expanding extraction scope from sentences to paragraphs resulted in the biggest gain, already surpassing DPL. Conducting end-to-end learning over a document-level representation, as in DocLevel, is beneficial compared to ensembling over predictions for individual discourse units (SentLevel, ParaLevel), especially in the base version. Interestingly, MultiScale still attained significant gain over DocLevel with an ensemble over SentLevel and ParaLevel, suggesting that the document-level representation can still be improved. In addition to prediction accuracy, the document-level models also have much more room to grow, as maximum recall is about 20 absolute points higher in MultiScale and DocLevel, compared to ParaLevel or DPL.[8]

The ensemble operator had a surprisingly large effect, as shown by the gain when it was changed from max (base version) to noisy-or. This suggests that combining weak signals across multiple scales can be quite beneficial. Our handcrafted gene-mutation filter also improved all systems substantially, corroborating the analysis of Wang and Poon (2018). In particular, without the filter, it is hard for the document-level models to achieve high precision, so they sacrifice a lot of recall to get good F1 scores. Using the filter helps them attain significantly higher recall while maintaining respectable precision.

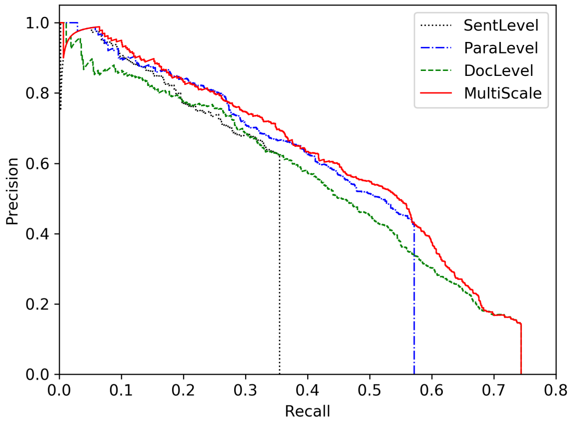


Figure 3: Precision-recall curves on CKB (with noisy-or and gene-mutation filter). MultiScale attained generally better precision than ParaLevel, and higher maximum recall like DocLevel.

Figure 3 shows the precision-recall curves for the four models (with noisy-or and gene-mutation filter). DocLevel has higher maximum recall than ParaLevel, but generally lower precision at the same recall level. By ensembling all three variants, MultiScale achieves the best combi-

---

[8] The difference in actual recall is less pronounced, as we chose thresholds to maximize F1 score. We expect actual recall to increase significantly as document-level models improve, whereas the other models are closer to their ceiling.

System	AUC	MR	P	R	F1
MultiScale	**47.5**	74.4	52.6	53.0	**52.8**
– SentLevel	47.0	74.4	43.0	55.8	48.6
– ParaLevel	45.9	74.4	48.8	49.8	49.3
– DocLevel	42.4	57.2	59.6	44.4	50.9

Table 4: Results on CKB when removing either SentLevel, ParaLevel, or DocLevel from the ensemble computed by MultiScale. MR=max recall, P=precision, R=recall.

System	AUC	P	R	F1
SentLevel	28.3	62.7	35.1	45.0
ParaLevel	38.1	47.4	52.2	49.7
DocLevel	41.1	48.2	45.6	46.9
MultiScale	43.7	45.7	51.2	48.3

Table 5: Results on CKB after replacing `logsumexp` with max (with noisy-or and gene-mutation filter). P=precision, R=recall. Max recall same as in Table 3.

nation: it generally improves precision while capturing more cross-paragraph relations. This can also be seen in Table 4, where we ablate each of the three variants used by MultiScale. All three variants in the ensemble contributed to overall performance.

We use logsumexp as the aggregation operator to combine mention-level representations into an entity-level one. If we replace it with max pooling, the performance drops substantially across the board, as shown in Table 5. For example, MultiScale lost 3.8 absolute points in AUC. Such difference is also observed in Verga et al. (2018). As in comparing ensemble operators, this demonstrates the benefit of combining weak signals using a multiscale representation.

4.4 Cross-sentence and Cross-paragraph Extractions

Compared to standard sentence-level extraction, our method can extract relations among entities that never co-occur in the same sentence or even paragraph. Figure 4 shows the proportion of correctly predicted facts by MultiScale that are expressed across paragraph or sentence boundaries. MultiScale can substantially improve the recall by making additional cross-sentence and cross-paragraph extractions. We manually inspected twenty correct cross-paragraph extractions (with the chosen threshold for the precision/recall numbers in Table 3) and found that our model was able to handle some interesting linguistic phenomena. Often, a paper would first describe the mutations present in a patient cohort, and later describe the effects of drug treatment. There are also instances of bridging anaphora, for example via cell lines. One paper first stated the gene and mutation for a cell line "*The **FLT3**-inhibitor resistant cells Ba/F3-ITD+691, Ba/F3-ITD+842, . . . , which harbored FLT-ITD plus F691L, **Y842C**, . . . mutations. . .*", and later stated the drug effect on the cell line "***E6201** also demonstrated strong anti-proliferative effects in **FLT3**-inhibitor resistant cells. . . such as Ba/F3-ITD+691, Ba/F3-ITD+842 . . .*".

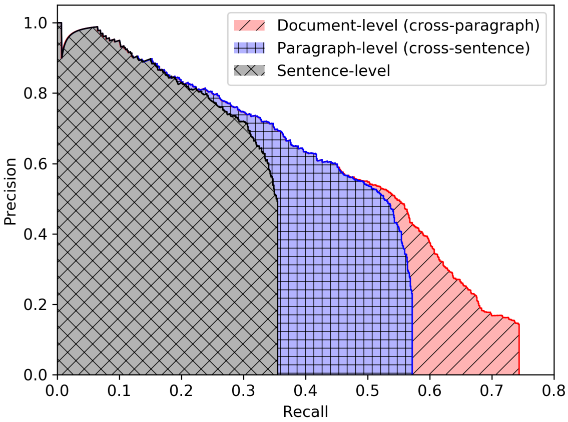


Figure 4: Breakdown of MultiScale recall based on whether entities in a correctly extracted fact occurred within a single sentence, cross-sentence but within a single paragraph, or only cross-paragraph. Adding cross-sentence and cross-paragraph extractions is important for high recall.

System	AUC	MR	P	R	F1
Base version					
SentDrugMut	31.0	40.8	60.0	40.7	48.5
SentDrugGene	17.9	64.2	31.4	27.9	29.5
ParaDrugMut	39.9	57.7	49.3	50.3	49.8
ParaDrugGene	19.9	68.9	32.1	18.9	23.8
+ Noisy-Or					
SentDrugMut	32.6	40.8	61.1	39.4	47.9
SentDrugGene	23.5	64.2	36.3	34.2	35.2
ParaDrugMut	42.0	57.7	49.9	51.5	50.7
ParaDrugGene	26.1	68.9	46.1	29.5	36.0

Table 6: Results of subrelation decomposition baselines on CKB, with the gene-mutation filter. MR=max recall, P=precision, R=recall.

4.5 Subrelation Decomposition

As a baseline, we also consider a different document-level strategy where we decompose the n-ary relation into subrelations of lower arity, train independent classifiers for them, then join the subrelation predictions into one for the n-ary relation. We found that with distant supervision, the gene-mutation subrelation classifier was too noisy. Therefore, we focused on training drug-gene and drug-mutation classifiers, and joined each with the rule-based gene-mutation predictions to make ternary predictions. Table 6 shows the results on CKB. The paragraph-level drug-mutation model is quite competitive, which benefits from the fact that the gene-mutation associations in a document are unique. This is not true in general n-ary relations. Still, it trails MULTISCALE by a large margin in predictive accuracy, and with an even larger gap in the potential upside (i.e., maximum recall). The drug-gene model has higher maximum recall, but much worse precision. This low precision is expected, as it is usually not valid to assume that if a drug and gene interact, then all possible mutations in the gene will have an effect on the drug response.

4.6 Error Analysis

While much higher compared to other systems, the maximum recall for MULTISCALE is still far from 100%. For over 20% of the relations, we can't find all three entities in the document. In many cases, the missing entities are in figures or supplements, beyond the scope of our extraction. Some mutations are indirectly referenced by well-known cell lines. There are also remaining entity linking errors (e.g., due to missing drug synonyms).

We next manually analyzed some sample prediction errors. Among 50 false positive errors, we found a significant portion of them were actually true mentions in the paper but were excluded by curators due to additional curation criteria. For example, CKB does not curate a fact referenced in related work, or if they deem the empirical evidence as insufficient. This suggests the need for even higher-order relation extraction to cover these aspects. We also inspected 50 sample false negative errors. In 40% of the cases, the textual evidence is vague and requires corroboration from a table or figure. In most of the remaining cases, there is direct textual evidence, though they require cross-paragraph reasoning (e.g., bridging anaphora). While MULTISCALE was able to process such phenomena sometimes, there is clearly much room to improve.

5 Related Work

N-ary relation extraction Prior work on n-ary relation extraction generally follows Davidsonian semantics by reducing the n-ary relation to n binary relations between the reified relation and its arguments, a.k.a. *slot filling*. For example, early work on the Message Understanding Conference (MUC) dataset aims to identify event participants in news articles (Chinchor, 1998). More recently, there has been much work in extracting semantic roles for verbs, as in semantic role labeling (Palmer et al., 2010), as well as properties for popular entities, as in Wikipedia Infobox (Wu and Weld, 2007) and TAC KBP[9]. In biomedicine, the BioNLP Event Extraction Shared Task aims to extract genetic events such as expression and regulation (Kim et al., 2009). These approaches typically assume that the whole document refers to a single coherent event, or require an event anchor (e.g., verb in semantic role labeling and trigger word in event extraction). We instead follow recent work in cross-sentence n-ary relation extraction (Peng et al., 2017; Wang and Poon, 2018; Song et al., 2018), which does not have these restrictions.

Document-level relation extraction Most information extraction work focuses on modeling and prediction within sentences (Surdeanu and Ji, 2014). Duan et al. (2017) introduces a pre-trained document embedding to aid event detection, but their extraction is still at the sentence level. Past work on cross-sentence extraction often relies on explicit coreference annotations or the assumption of a single event in the document (Wick et al., 2006; Gerber and Chai, 2010; Swampillai and Stevenson, 2011; Yoshikawa et al., 2011; Koch et al., 2014; Yang and Mitchell, 2016). Recently, there has been increasing interest in general cross-sentence relation extraction (Quirk and Poon, 2017; Peng et al., 2017; Wang and Poon, 2018), but their scope is still limited to short text spans of a few consecutive sentences. These methods all extract relations at the mention level, which does not scale to whole documents due to the combinatorial explosion of relation candidates. Wu et al. (2018b) applies manually crafted rules to heavily filter the candidates. We instead adopt an entity-centric approach and combine mention-level representations to create an entity-level representation for extraction. Mintz et al. (2009) aggregates mention-level features into entity-level ones within a document, but they only consider

---

[9] `http://www.nist.gov/tac/2016/KBP/ColdStart/index.html`

binary relations within single sentences. Kilicoglu (2016) used hand-crafted features to improve cross-sentence extraction, but they focus on binary relations, and their documents are limited to abstracts, which are substantially shorter than the full-text articles we consider. Verga et al. (2018) applies self-attention to combine the representations of all mention pairs into an entity pair representation, which can be viewed a special case of our framework. Their work is also limited to binary relations and abstracts, rather than full documents.

Multiscale modeling Deep learning on long sequences can benefit from multiscale modeling that accounts for varying scales in the discourse structure. Prior work focuses on generative learning such as language modeling (Chung et al., 2017). We instead apply multiscale modeling to discriminative learning for relation extraction. In addition to modeling various scales of discourse units (sentence, paragraph, document), we also combine mention-level representations into an entity-level one, as well as sub-relations of the n-ary relation. McDonald et al. (2005) learn $\binom{n}{2}$ pairwise relation classifiers, then construct maximal cliques of related entities, which also bears resemblance to our subrelation modeling. However, our approach incorporates the entire subrelation hierarchy, provides a principled end-to-end learning framework, and extracts relations from the whole document rather than within single sentences.

Distant supervision Distant supervision has emerged as a powerful paradigm to generate large but potentially noisy labeled datasets (Craven et al., 1999; Mintz et al., 2009). A common denoising strategy applies multi-instance learning by treating mention-level labels as latent variables (Hoffmann et al., 2011). Noise from distant supervision increases as extraction scope expands beyond single sentences, motivating a variety of indirect supervision approaches (Quirk and Poon, 2017; Peng et al., 2017; Wang and Poon, 2018). Our entity-centric representation and multiscale modeling provide an orthogonal approach to combat noise by combining weak signals spanning various text spans and subrelations.

6 Conclusion

We propose a multiscale, entity-centric approach for document-level n-ary relation extraction. We vastly increase maximum recall by scoring document-level candidates. Meanwhile, we preserve precision with a multiscale approach that combines representations learned across the subrelation hierarchy and text spans of various scales. Our method substantially outperforms prior cross-sentence n-ary relation extraction approaches in the high-value domain of precision oncology.

Our document-level view opens opportunities for multimodal learning by integrating information from tables and figures (Wu et al., 2018a). We used the ternary drug-gene-mutation relation as a running example in this paper, but knowledge bases often store additional fields such as effect (sensitive or resistance), cancer type (solid tumor or leukemia), and evidence (human trial or cell line experiment). It is straightforward to apply our method to such higher-order relations. Finally, it will be interesting to validate our approach in a real-world assisted-curation setting, where a machine reading system proposes candidate facts to be verified by human curators.

Acknowledgements

We thank Sara Patterson and Susan Mockus for guidance on precision oncology knowledge curation and CKB data, Hai Wang for help in running experiments with deep probabilistic logic, and Tristan Naumann, Rajesh Rao, Peng Qi, John Hewitt, and the anonymous reviewers for their helpful comments. R.J. is supported in part by an NSF Graduate Research Fellowship under Grant No. DGE-114747.

References

Orli Bahcall. 2015. Precision medicine. *Nature*, 526:335.

Debyani Chakravarty, Jianjiong Gao, Sarah Phillips, Ritika Kundra, Hongxin Zhang, Jiaojiao Wang, Julia E. Rudolph, Rona Yaeger, Tara Soumerai, Moriah H. Nissan, Matthew T. Chang, Sarat Chandarlapaty, Tiffany A. Traina, Paul K. Paik, Alan L. Ho, Feras M. Hantash, Andrew Grupe, Shrujal S. Baxi, Margaret K. Callahan, Alexandra Snyder, Ping Chi, Daniel C. Danila, Mrinal Gounder, James J. Harding, Matthew D. Hellmann, Gopa Iyer, Yelena Y. Janjigian, Thomas Kaley, Douglas A. Levine, Maeve Lowery, Antonio Omuro, Michael A. Postow, Dana Rathkopf, Alexander N. Shoushtari, Neerav Shukla, Martin H. Voss, Ederlinda Paraiso, Ahmet Zehir, Michael F. Berger, Barry S. Taylor, Leonard B. Saltz, Gregory J. Riely, Marc Ladanyi, David M. Hyman, Jos Baselga, Paul Sabbatini, David B. Solit, and Nikolaus Schultz. 2017. Oncokb: A precision oncology knowledge base. *JCO Precision Oncology*.

Nancy Chinchor. 1998. Overview of MUC-7/MET-2. Technical report, Science Applications International Corporation, San Diego, CA.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *ICLR*.

M. Craven, J. Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, pages 77–86.

Rodrigo Dienstmann, In Sock Jang, Brian Bot, Stephen Friend, and Justin Guinney. 2015. Database of genomic biomarkers for cancer drugs and clinical targetability in solid tumors. *Cancer Discovery*, 5.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *IJCNLP*.

Matthew Gerber and Joyce Y. Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the Forty-Eighth Annual Meeting of the Association for Computational Linguistics*.

R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*, pages 541–550.

Halil Kilicoglu. 2016. Inferring implicit causal relationships in biomedical literature. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Junichi Tsujii. 2009. Overview of bionlp-09 shared task on event extraction. In *Proceedings of the BioNLP Workshop 2009*.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mitchell Koch, John Gilmer, Stephen Soderland, and S. Daniel Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1891–1901. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings of the Forty-Third Annual Meeting on Association for Computational Linguistics*.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics (ACL)*, pages 1003–1011.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1).

Sarah E. Patterson, Rangjiao Liu, Cara M. Statz, Daniel Durkin, Anuradha Lakshminarayana, and Susan M. Mockus. 2016. The clinical trial landscape in oncology and connectivity of somatic mutational profiles to targeted therapies. *Human Genomics*, 10.

Nanyun Peng, Hoifung Poon, Chris Quirk, and Kristina Toutanova Wen tau Yih. 2017. Cross-sentence N-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Sampo Pyysalo, F Ginter, Hans Moen, T Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of Languages in Biology and Medicine*.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the Fifteenth Conference on European chapter of the Association for Computational Linguistics*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph state LSTM. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mihai Surdeanu and Heng Ji. 2014. Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*, pages 1–15, Gaithersburg, Maryland, USA.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting relations within and across sentences. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *NAACL*.

Hai Wang and Hoifung Poon. 2018. Deep probabilistic logic: A unifying framework for indirect supervision. In *EMNLP*.

Michael Wick, Aron Culotta, and Andrew McCallum. 2006. Learning field compatibilities to extract database records from unstructured text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*.

S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. R'e. 2018a. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of SIGMOD 2018*.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher R. 2018b. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*.

Katsumasa Yoshikawa, Sebastian Riedel, Tsutomu Hirao, Masayuki Asahara, and Yuji Matsumoto. 2011. Coreference based event-argument relation extraction on biomedical text. *Journal of Biomedical Semantics*, 2(5).

A Appendices

A.1 Preprocessing

Full-text documents in this study were obtained from PMC. The text was first tokenized using NLTK[10], then entities were extracted using a combination of regular expressions and dictionary lookups. To identify mutation mentions, we applied a regular expression rule for missense mutations. To identify gene mentions, we used dictionary lookup from the HUGO Gene Nomenclature Committee (HGNC)[11] dataset. To identify drug mentions, we used dictionary lookup from a curated list of drugs and their synonyms. For our training set, our list of drugs consists of all the drugs present in the distant supervision knowledge bases and selected cancer-related drugs from DrugBank[12] (770 drugs total). For our test set, our drug dictionary consists of all the drugs in CKB (1119 drugs).

A.2 Gene-mutation Rule-based System

Here we describe our rule-based system for linking mutations and genes within a document. We first generate a global mapping of mutations to sets of genes by combining publicly-available mutation-gene datasets (COSMIC[13], COSMIC Cell Lines Project[14], CIViC[15], and OncoKB[16]). We then augment this mapping by finding the gene that most frequently co-occurs with each mutation in all of PubMed Central (PMC) full-text articles based on three high-precision rules:

1. Gene and mutation are in the same token (e.g., ”EGFR-T790M”)
2. Gene token is followed by mutation token (e.g., ”EGFR T790M”)
3. Gene token is followed by a token of any single character and then followed by mutation token (e.g., ”EGFR - T790M”)

For each mutation, we start with the first rule, and find all text matches for a gene with that mutation and rule. If we found at least one match, we add the gene that occurred in the most matches to the global map. Otherwise, we repeat with the next rule.

Each mutation in the global mutation-gene map is mapped to more than 20 genes on average. However, within the context of a document, each mutation is (usually) associated with just a single gene. Given a document containing a mutation, we associate that mutation with the gene that (1) is in the global mutation-gene map for that mutation, and (2) appears closest to any mutation mention in the document.

To associate genes for the remaining mutations, we apply two recall-friendly regular expression rules within that document:

4. Mutation is in same sentence as “*GENE* mut”
5. Mutation is in same paragraph as “*GENE* mutation”

We choose the first gene in the document that satisfies one of the two rules, in the above order. If there is still no matching gene at this point, the most frequent gene in the document is selected for that mutation.

A.3 Model Details

We used 200-dimensional word vectors, initialized with word2vec vectors trained on a biomedical text corpus (Pyysalo et al., 2013). We updated

---

[10] https://www.nltk.org/

[11] https://www.genenames.org/

[12] https://www.drugbank.ca/

[13] https://cancer.sanger.ac.uk/cosmic

[14] https://cancer.sanger.ac.uk/cell_lines

[15] https://civicdb.org

[16] http://oncokb.org/

these vectors during training. At each step, our BiLSTM received as input a concatenation of the word vectors and a 100-dimensional embedding of the index of the current discourse unit within the document. Following Vaswani et al. (2017), we used sinusoidal embeddings to represent these indices. We used a single-layer bidirectional LSTM with a 200-dimensional hidden state. Mention-level representations were 400-dimensional and computed from BiLSTM hidden states using a single linear layer followed by the $\tanh$ activation function. For the final prediction layer, we used a two-layer feedforward network with 400 hidden units and ReLU activation function. We train using the Adam optimizer (Kingma and Ba, 2014) with learning rate of 1×10^{-5}. During training, we consider each document to be a single batch, which allows us to reuse computation for different relation candidates in the same document.

Inferring Which Medical Treatments Work from Reports of Clinical Trials

Eric Lehman
Northeastern University
`lehman.e@northeastern.edu`

Jay B. DeYoung
Northeastern University
`deyoung.j@northeastern.edu`

Regina Barzilay
MIT
`regina@csail.mit.edu`

Byron C. Wallace
Northeastern University
`b.wallace@northeastern.edu`

Abstract

How do we know if a particular medical treatment actually works? Ideally one would consult all available evidence from relevant clinical trials. Unfortunately, such results are primarily disseminated in natural language scientific articles, imposing substantial burden on those trying to make sense of them. In this paper, we present a new task and corpus for making this unstructured evidence actionable. The task entails inferring reported findings from a full-text article describing a randomized controlled trial (RCT) with respect to a given intervention, comparator, and outcome of interest, e.g., inferring if an article provides evidence supporting the use of *aspirin* to reduce *risk of stroke*, as compared to *placebo*.

We present a new corpus for this task comprising 10,000+ prompts coupled with full-text articles describing RCTs. Results using a suite of models — ranging from heuristic (rule-based) approaches to attentive neural architectures — demonstrate the difficulty of the task, which we believe largely owes to the lengthy, technical input texts. To facilitate further work on this important, challenging problem we make the corpus, documentation, a website and leaderboard, and code for baselines and evaluation available at `http://evidence-inference.ebm-nlp.com/`.

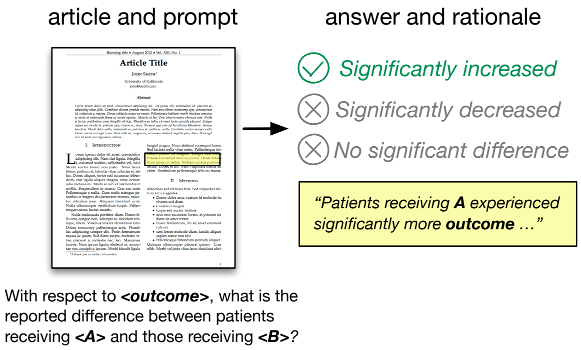


Figure 1: The task. Given a treatment **A**, a comparator **B**, and an **outcome**, infer the reported relationship between **A** and **B** with respect to **outcome**, and provide evidence supporting this from the text.

1 Introduction

Biomedical evidence is predominantly disseminated in unstructured, natural language scientific manuscripts that describe the conduct and results of randomized control trials (RCTs). The published evidence base is vast and expanding (Bastian et al., 2010): at present more than 100 reports of RCTs are published every day, on average. It is thus time-consuming, and often practically impossible, to sort through all of the relevant published literature to robustly answer questions such as: *Does infliximab reduce dysmenorrhea (pain) scores, relative to placebo*?

Given the critical role published reports of trials play in informing evidence-based care, organizations such as the Cochrane collaboration and groups at evidence-based practice centers (EPCs) are dedicated to manually synthesizing findings, but struggle to keep up with the literature (Tsafnat et al., 2013). NLP can play a key role in automating this process, thereby mitigating costs and keeping treatment recommendations up-to-date with the evidence as it is published.

In this paper, we consider the task of inferring whether a given treatment is effective with respect to a specified outcome. Typically, this assessment is done relative to other treatment options (i.e., comparators). We assume the model is provided with a *prompt* that specifies an intervention, a comparator, and an outcome, along with a full-text article. The model is then to infer the reported findings with respect to this prompt (Figure 1). From a healthcare perspective, this inference task is an essential step for automating extraction of actionable evidence from trial reports.

Proceedings of NAACL-HLT 2019, pages 3705–3717
Minneapolis, Minnesota, June 2 - June 7, 2019.

From an NLP standpoint, the proposed task can be seen as an instance of natural language inference (Bowman et al., 2015), viewing the article and prompt as the premise and hypothesis, respectively. However, the problem differs in a few important ways from existing NLP formulations. First, the inputs: prompts are brief (∼13.5 words on average), but articles are long (∼4200 words). Further, only a few snippets of the article will be relevant to the label for a given prompt. Second, prompts in this domain are structured, and include only a few types of key information: interventions, comparators, and outcomes. Methods that exploit this regularity are likely to be more accurate than generic inference algorithms.

Another interesting property of this task is that the target for an article depends on the interventions and outcome specified by a given prompt. Most articles report results for multiple interventions and outcomes: 67% of articles in our corpus are associated with two or more prompts that have *different* labels, e.g., indicating that a specific treatment was comparatively effective for one outcome but not for another. As a concrete example from our corpus, *infliximab* was reported as realizing *no significant difference* with respect to *dysmenorrhea*, compared to a *placebo*. But *infliximab* was associated with a *significant increase* in *pain killer intake*, again compared to *placebo*. Generally positive words in an article (e.g., "improved") will confuse inference models that fail to account for this. One may view these as built-in "adversarial" examples (Jia and Liang, 2016) for the task.

A key sub-problem is thus identifying snippet(s) of evidence in an article *relevant to a given input prompt*. Attention mechanisms (Bahdanau et al., 2014) conditioned on prompts would seem a natural means to achieve this, and we do find that these achieve predictive gains, but they are modest. Existing attention variants seem to struggle to consistently attend to relevant evidence, even when explicitly pretrained using marked rationales. This corpus can facilitate further research in attention variants designed for lengthy inputs (Choi et al., 2017; Yang et al., 2016).

In sum, our contributions are threefold. We: (1) formulate a novel task (*evidence inference*) that is both practically important and technically challenging; (2) Provide a new publicly-available corpus comprising 10,000+ evidence "prompts", answers, supporting evidence spans, and associated full-text articles (`http://evidence-inference.ebm-nlp.com`) all manually annotated by medical doctors; (3) Develop baseline algorithms to establish state-of-the-art performance and highlight modeling challenges posed by this new task.

2 Annotation

The specialized nature of this task necessitates adequate domain knowledge. We thus recruited medical doctors (MDs) via the Upwork platform to perform annotation. Annotators were assigned to one of three mutually exclusive groups, responsible for: (1) prompt generation, (2) prompt and article annotation, and (3) verification. Figure 2 depicts the annotation process schematically; we describe these steps in more detail below.

It is important to note that annotation was performed on full-texts, not just abstracts. Evidence relevant to a particular clinical question is quite often only available in the full text. Indeed, in our dataset, the relevant evidence span was marked in the abstract only 40.5% of the time.

2.1 Prompt Generation

This first task entails generating questions (or "prompts") that are answerable on the basis of a corresponding full-text article describing an RCT. Such prompts concern the comparison of specific interventions with respect to a particular outcome. Specifically, these questions ask whether an article reports that the specified intervention was found (in the described trial) to be significantly more effective than a comparator treatment, with respect to the outcome of interest.

Prompt creators were instructed to identify a snippet, in a given full-text article, that reports a relationship between an intervention, comparator, and outcome. Generators were also asked to provide answers and accompanying rationales to the prompts that they provided; such supporting evidence is important for this task and domain.

As a concrete example, an example generated prompt for a trial described in (Marre et al., 2009) specifies *Proinsulin : insulin ratio* as the outcome of interest, *liraglutide (1.8 mg) plus glimepiride* as the intervention, and *rosiglitazone plus glimepiride* as the comparator. Liraglutide and rosiglitazone are both drugs that can be used to treat type 2 diabetes. In this case, use of the intervention (liraglutide) was reported to *signifi-*

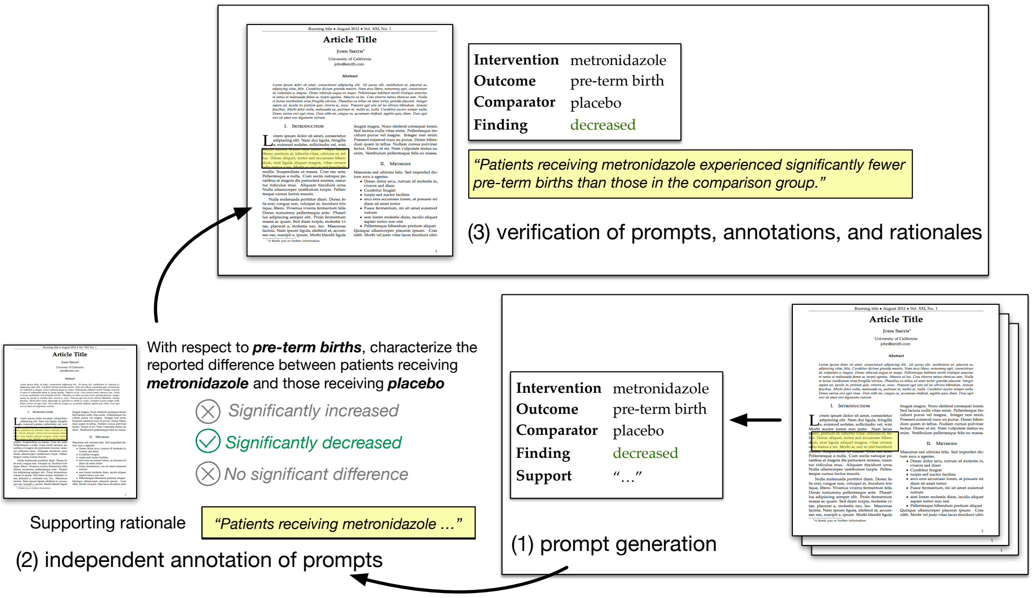


Figure 2: Schematic of the annotation process, performed by qualified (MD) annotators for all articles and prompts.

cantly decrease the proinsulin to insulin ratio, as supported by the following evidence snippet extracted by the prompt creator: "Reductions in the proinsulin : insulin ratio were greater with both liraglutide 1.2 and 1.8 mg compared with either rosiglitazone or placebo."

Trials typically report results for multiple outcomes, and often for more than two interventions. As discussed above, results for these will often differ. For instance, *postprandial plasma glucose* was another outcome reported in the aforementioned trial report, and *placebo plus glimepeiride* was considered as another comparator. Therefore, we instructed prompt generators to create multiple prompts for each full-text article. On average, this yielded 4.19 distinct prompts per article.[1]

Articles may be deemed *invalid* for a few reasons, chiefly for not describing RCTs.[2] Of 3525 articles considered, 1106 were marked invalid (31.4%). The prompt generators provided valid answers and rationales in 95.9% and 97.8% of cases, respectively, as per the verifier.

To summarize: prompt creation entails specifying answerable clinical questions, along with answers to these and supporting rationales (evidence snippets from the text). This task is the most laborious step in the annotation process.

2.2 Prompt Annotation

For this task, annotators were asked to answer prompts on the basis of a particular article. More specifically, given an evidence prompt articulating an intervention, comparator, and outcome (generated as described above), the task was to determine whether the associated article reports results indicating that the intervention *significantly increased*, *significantly decreased*, or realized *no significant difference*, relative to the comparator and with respect to the outcome. The annotator was also asked to mark a snippet of text supporting their response. Annotators also had the option to mark prompts as *invalid*, e.g., if the prompt did not seem answerable on the basis of the article.

Annotations collected in this step are redundant with the classification and rationale independently provided by the prompt generator in the preceding step; this is by design to ensure robust, high-quality annotations.

2.3 Verification

The final task in our annotation process entails a worker verifying the prompts and responses generated in the previous two steps. The verifier is here responsible for checking both whether the prompt (i.e., question) is valid and can be answered from the text, and whether the responses provided are accurate. Verifiers also assess whether the associated supporting evidence provided is reasonable.

Verification is a relatively easy task, because the verifier is directly provided all information rele-

---

[1] We restricted generators to creating at most five prompts for a given article; prior to imposing this constraint, annotators would sometimes generate >10 prompts per article.

[2] We used the RobotReviewer RCT classifier, which improves upon the standard MEDLINE RCT filter (Marshall et al., 2018), but some false positives remain.

vant to making a quality judgment. Nonetheless, this step decidedly improved data quality: 3.8% of prompts, 6.7% of answers, and 7.1% of rationales (supporting evidence snippets) were marked as invalid. All invalid prompts were removed from the corpus; so too were all prompts for which the verifier rejected all answers or all rationales.

2.4 Task Refinement

In an initial pilot round, we acquired annotations on 10 articles, yielding 93 prompts. Three medical doctors (MDs) were tasked with answering these prompts, achieving an agreement of 0.58 (Krippendorf's α). To improve this poor agreement, we provided personalized feedback that addressed systematic issues we observed. Following this feedback, the MDs were asked to re-examine the same set of prompts and update their responses if they felt it appropriate to do so. This resulted in a much improved agreement of α=0.84.

To verify that this agreement held beyond the specific set with respect to which we provided feedback, we subsequently assigned an additional 113 prompts to the annotators. As measured over these 113 prompts, the three annotators exhibited relatively high agreement between themselves and with the prompt generator (Krippendorf's α of 0.75 and 0.80, respectively).

3 Dataset Statistics

We hired 16 doctors from Upwork and split them at random into groups: 10 for prompt generation, 3 for annotation, and 3 for verification.[3] In total, we have acquired 10,137 annotated prompts for 2,419 unique articles. For each of these prompts, we have at least two independent sets of labels and associated rationales (supporting snippets).

We additionally calculated agreement between prompt generators, annotators and verifiers using Krippendorf's α. To calculate this, we converted the verifier's binary labels of valid or not to the label with which they agreed. This yields $\alpha = 0.88$. Removing the verifier annotations from the calculation results in $\alpha = 0.86$.

Intervention, outcome, and comparator strings contain on average 5.1, 5.3, and 3.4 tokens, respectively. Articles comprise a mean of 4.2k tokens.

We provide additional details concerning the dataset in the Appendix.

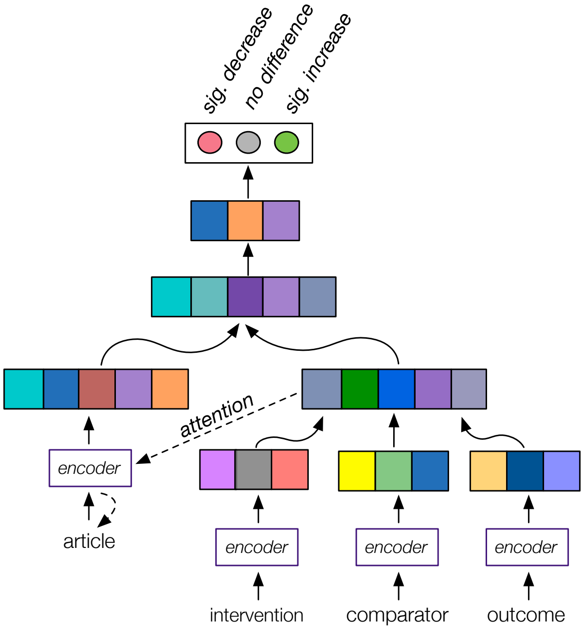


Figure 3: A generalized illustration of the neural network we propose for evidence inference.

4 Models

We experimented with a suite of models to establish performance on this task, which we explain below in increasing order of complexity.

4.1 Baselines

Majority. Predict the most common class, i.e., *no significant difference*.

Heuristics. This entails two parts: (1) finding the sentence that contains the answer, and (2) interpreting the sentence that possesses the evidence. The first step of this process is achieved through locating the sentence that has the most overlap with words in the outcome, intervention, and comparator. Afterwards, we search for reported p-values within the identified sentence, and evaluate whether they seem significant. We provide a detailed description of the heuristics model in Section A of the Appendix.

Logistic Regression. A standard logistic regression model trained on top of binary bag-of-words representations of articles and intervention, comprator and outcome (ICO) frames — these are concatenated to form inputs. We use a vocabulary size of 20k (based on frequency of occurrence), thus yielding an input size of 80k.

Neural Network Variants. We encode the intervention, comparator, and outcome strings accompanying a prompt into vectors i, c, and o, respec-

---

[3] One doctor from verification was moved to annotation in order to increase productivity.

	Train	Dev	Test	Total
Number of prompts	8168	1004	965	10137
Number of articles	1931	248	240	2419
Label counts (-1 / 0 / 1)	1981 / 3619 / 2568	232 / 448 / 324	215 / 403 / 347	2428 / 4470 / 3239

Table 1: Corpus statistics. Labels -1, 0, 1 indicate *significantly decreased*, *no significant difference* and *significantly increased*, respectively.

tively. Similarly, we encode the article itself into a vector a. We experimented with several encoder options, including simple bag-of-words style encoding (i.e., averaging constituent word vectors) and RNNs. For the latter, we specifically pass a Gated Recurrent Unit (Cho et al., 2014), or GRU, over inputs, yielding hidden states for each article token.

In preliminary experiments we found that simple averages over token embeddings worked well for encoding prompts (i, c and o), likely because they tend to be quite short. But this encoding works terribly for articles, due to their length. Therefore, we use a GRU to encode articles (uni-directional, as bi-directional added complexity without improving results).

In the simplest neural model variant, we simply concatenate the encoded article and ICO frame into a vector $[a; i; c; o]$ which is then passed through a feedforward network with a single hidden layer to allow interactions between the prompt and article text.[4] As discussed in detail below, we experiment with a variety of attention mechanisms imposed over article tokens.

4.2 Finding the Evidence

Exploiting the spans of evidence marked as supporting assessments should improve the predictive performance of models. An additional advantage of modeling this explicitly is that models will then be able to provide *rationales* for decisions (Lei et al., 2016; Zhang et al., 2016; Zaidan et al., 2007), i.e., snippets of text that support predictions. We therefore experiment with model variants that classify input tokens as being relevant evidence (or not) prior to performing inference.

We consider both *pipeline* and *joint* instantiations of such models. In the former type, the model first identifies spans in the text and then passes these forward to an independent component that makes predictions on the basis of these. In models of the latter type, evidence span tagging and document-level inference is performed end-to-end. Evidence snippets are not restricted to sentence boundaries (i.e., are token-wise), but we also consider model variants that relax evidence span tagging to a sentence labeling task (classifying sentences as containing *any* evidence tokens, or not). In either case, which spans are relevant will depend on the prompt assessed. Thus, we consider and contrast variants that condition evidence span prediction on the input prompt.

We consider both linear and neural models. For the former, we train two logistic regression models over bag-of-words input representations. The first predicts whether or not a given sentence contains any evidence tokens. Document predictions are then made via a second (independent) logistic regression model that consumes aggregate bag-of-words representations of only those sentences predicted to contain evidence. This is a linear model, and thus does not accommodate interactions; we therefore consider only an unconditioned version.

For our pipeline neural model, we first induce vector representations of article sentences via a GRU, and these are then passed through a binary classification layer. To allow interactions between the input prompt and the sentence being classified, we also consider a conditioned variant in which the sentence classification model is provided the induced vector representations of the prompt elements alongside the sentence vector.

For end-to-end models, we capitalize on attention mechanisms that learn to focus on (contextualized hidden representations of) individual article tokens prior to making a prediction. We consider several variants of attention, and we explore directly pretraining these using the marked evidence spans available in the training data.

The simplest attention module we consider is unconditioned; we simply learn weights W that scores hidden states h_a output from the article encoder. Concretely,

$$\alpha = \text{softmax}\{w_\alpha \cdot H_a\} \qquad (1)$$

where $w_\alpha \in \mathcal{R}^{1\times d}$ and $H_a \in \mathcal{R}^{d\times|a|}$, denoting hidden size by d and article length by $|a|$.[5]

---

[4] We use a linear hidden layer; experiments adding a non-linearity (ReLU) did not affect results.

[5] We have elided bias terms for presentation.

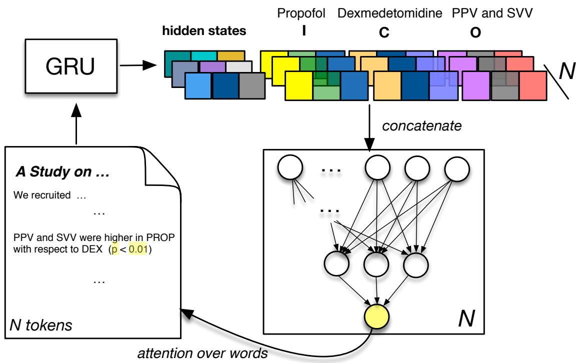


Figure 4: The proposed conditional attention variant. ICO frame embeddings are concatenated to hidden state vectors from the GRU and fed through an MLP to induce attention weights.

However, the text span relevant to a classification will depend on the prompt under consideration. We thus also consider a conditioned variant of attention. In this version we concatenate the i, c, and o vectors induced by our encoders to the hidden states. Abusing notation a bit, denote the matrix in which we concatenate the i, c, and o vectors to each column in H_a by $[H_a; i; c; o]$. We then consider an attention variant that passes this concatenated representation through a single hidden layer to score tokens (Figure 4).

$$\alpha = \text{softmax}\{v_\alpha^t \cdot \tanh(W_\alpha \cdot [H_a; i; c; o])\} \quad (2)$$

We consider two ways of converting the provided evidence spans into targets: (1) Imposing a uniform distribution over marked evidence tokens; (2) Setting the target for all marked evidence tokens as 1. In both cases we treat the absence of annotations on a token as an implicit negative target (0). It is important to note that the model will see the same article multiple times during training with different evidence span targets, one for each prompt in the train set. The snippet of text that supports a particular assessment naturally depends on the prompt under consideration. Unconditioned attention variants will thus, by construction, be unable to attend exclusively to the relevant spans of text for across all prompts.

When training with binary targets, we consider two specifications: one in which the outputs of the attention model are independent (per-token) sigmoids indicating whether or not a word belongs to an evidence span, and another in which attention weights are normalized via a softmax over tokens. The latter is standard, although per-token attention has been previously proposed (Kim et al., 2017).

5 Experimental Details

5.1 Development Setting

During model development, we used 90% of the train set for training, and the remaining 10% as to monitor performance over epochs. To iteratively assess and refine models during this development phase, we used the standardized validation set. All decisions regarding final experiments to run were made using this validation set, prior to evaluating models on the held-out test set of articles. Results reported in this paper are on the final test set. Note that we report averages for neural models (over five runs) to mitigate noise due to random initialization and fitting.

5.2 Training Details and Hyperparameters

All neural variants were trained up to 50 epochs with a patience of 10 epochs. We monitored performance during training on a nested development set and retained the model that achieved the highest F1 score on this. For GRU encoders we used 32 hidden units. All models for the primary task were trained with batch sizes of 32.

We initialized word embeddings to pretrained word vectors induced over a large set of PubMed abstracts (Pyysalo et al., 2013). Given the modest training dataset size, we did not fine-tune these.

5.3 Attention pretraining Details

We use the manually marked supporting snippets as explicit, intermediate supervision for pretraining the attention mechanisms described in 4.2. More specifically, we pretrain the attentional model components for both conditioned and unconditioned attention variants.

Concretely, we minimize token-wise binary cross entropy loss with respect to one of the two token-wise targets delineated in the preceding Section. We normalize loss per batch by the number of constituent tokens, using batch sizes of 16.[6] We monitor token-wise AUC with respect to the reference evidence span annotations marked in the held-out validation set mention in Section 5.1. We retained the model that achieved the best AUC measured over fifty epochs of attention pretraining (again with a patience of ten) and used these weights as initialization values for fine-tuning the end-to-end inference network.[7]

---

[6] Memory constraints precluded larger batches.

[7] We also experimented with 'freezing' attention module parameters during fine-tuning, but this performed poorly.

Model	Precision	Recall	F1	Evidence token AUC / mass
Majority	0.138	0.333	0.195	–
Heuristics	0.431	0.389	0.354	0.682 / 0.025
Logistic regression	0.409	0.400	0.388	–
Pipeline logistic regression	0.452	0.429	0.423	0.523 / 0.012
Pipeline neural network	0.422	0.405	0.402	0.847 / 0.080
+ Conditioned	0.426	0.420	0.417	0.863 / 0.062
End-to-end neural network	0.471	0.439	0.440	–
+ Attention	0.528	0.507	0.508	0.759 / 0.047
+ Pretrain attention†	0.527	0.507	0.505	0.880 / 0.129
+ Conditional attention	0.522	0.504	0.505	0.706 / 0.059
+ Pretrain conditional attention†	0.531	0.519	0.520	0.836 / 0.125

Table 2: Summary results on the evidence inference task test set, averaged over five independent runs (with independent random initialization values). Metrics are macro-averages over classes. Evidence token AUC and mass (last column) quantify identification of relevant supporting tokens. Models in the top two rows perform no learning; the second two correspond to linear models; the rest are neural model variants. † indicates 'token-wise' attention pretraining; we report results for alternative attention losses in the Appendix.

Class	Precision	Recall	F1
Sig. decreased	0.448	0.334	0.380
No sig. diff.	0.586	0.636	0.610
Sig. increased	0.556	0.585	0.572

Table 3: Average per-class test performance of best overall model (*pretrain conditional attention*).

We trained all models with the Adam optimizer using the parameters suggested by (Kingma and Ba, 2014). We trained using PyTorch (Paszke et al., 2017), `v 1.0.1.post2`.[8] Code for our models and to reproduce our results is available at: `https://github.com/jayded/evidence-inference`.

5.4 Pipeline Model Training Details

Pipeline models first attempt to identify sentences containing evidence. To train these, we generalize token-wise annotations to sentences such that a sentence is labeled 1 if it contains any evidence tokens, and 0 otherwise. We then trained the sentence tagging models described above with these labels, monitoring loss on a nested validation set and retaining the best observed model over 50 epochs. The document-level model subsequently consumes only sentences tagged as relevant.

6 Results

6.1 Main Task Results

Results on the main task for proposed model variants are reported in Table 2. These are averages over five independent runs, to ensure relatively robust measures of model performance.[9] The best performing model exploits pretrained conditional attention. For the leaderboard we assume a single set of model predictions. To generate these we evaluated models on the test set using the versions that realized the strongest observed performance on the validation set over the aforementioned five runs/initializations. The best performing model (and hence current leader) is the variant that uses pretrained, conditional attention, which aligns with the average results in Table 2. Table 4 reports the results here, along with a more standard attentive architecture for context.

Model	Precision	Recall	F1
NN + Attention	0.518	0.503	0.505
NN + pretrain cond. attn.	**0.533**	**0.530**	**0.531**

Table 4: Leaderboard results.

To highlight the importance of identifying relevant spans to inform predictions, we present results achieved when these are provided directly to models via an 'oracle' prior to prediction in Table 6. Access to this oracle yields a 20+ point jump in F1, indicating that accurately extracting the relevant evidence is critical. Below (Section 6.2) we attempt to elucidate how well (or poorly) attention mechanisms fail to find supporting evidence.

A natural question that arises in NLP tasks in which the output depends on both a document and a question (here, a prompt) is: how much does the latter in fact influence model predictions (Kaushik and Lipton, 2018)? We explore this in Table 5. Relying only on the prompt (ignoring the article completely) achieves surprisingly strong performance, outperforming a vanilla neural model (sans attention). This is not entirely unreasonable, as certain intervention types will tend to correlate with significant vs insignificant findings, i.e., the prompt

---

[8] This is a nightly build, used due to a dependence on recently introduced RNN utilities.

[9] These models exhibit a fair amount of variance; we report ranges over the validation set in the Appendix.

Model	Precision	Recall	F1
Best NN	0.531	0.519	0.520
NN (no attention)	0.471	0.439	0.440
- prompt	0.344	0.340	0.324
- article	0.489	0.468	0.472

Table 5: Average results achieved (macro-averages over five runs) by the neural model when it is provided only the article or only the prompt. We reproduce results for the best model from Table 2 and the vanilla (no attention) end-to-end neural network for context.

Model	Precision	Recall	F1
Heuristics	0.492	0.457	0.453
Logistic regression	0.732	0.734	0.731
Neural network	0.740	0.739	0.739

Table 6: Average results achieved when models are provided directly with the reference evidence spans.

itself contains signal. The neural model without attention is likely simply unable to extract meaningful signal from lengthy articles, and so induced representations merely add noise. By contrast, ignoring the prompt severely degrades performance.

6.2 Analyzing Attention

To provide a sense of how well models are able to identify relevant evidence (i.e., tokens in the supporting snippets marked by annotators), we report token AUCs and *evidence masses* for all models that assign scores to words. The former captures how well models discriminate evidence from non-evidence tokens in general; the latter measures the relative amount of attention payed to evidence tokens. Concretely we calculate attention mass as a sum of the normalized attention scores assigned to words that belong to reference evidence spans. Thus, e.g., if the evidence token mass were 1, this would mean the model attended to *only* relevant evidence, ignoring all other tokens. We also experimented with optimizing for this directly during attention pretraining (see Appendix).

Aside from the *Majority* and *LR* baselines, all of the models explored generate scores encoding token relevance, either explicitly or implicitly. Attentive neural variants induce these by scoring contextualized representations of tokens t, h_t for relevance. Pipelined models score sentences, not tokens. For comparison across models, we assign the probability predicted for a given sentence to all of the words that it contains. Note that the maximum evidence token AUC achievable when selecting a sentence is ~0.92.

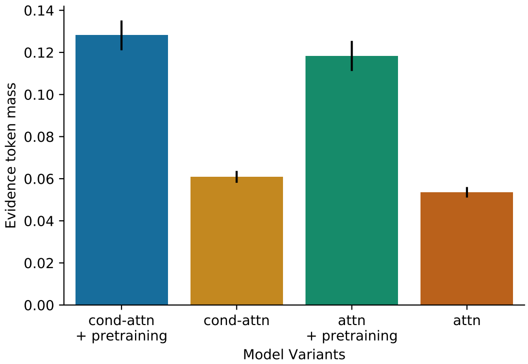


Figure 5: Evidence token masses achieved by models on the validation set, after training.

Qualitatively, we observe that attention weights often, though not always, square with intuition. In a test example wherein the intervention is *propofol*, the comparator *dexmedetomidine*, and the outcome *Stroke Volume Variation (SVV) and Pulse Pressure Variation (PPV)*, the conditioned, pretrained attention model focuses on tokens ‘svv’, ‘versus’ (suggesting comparison), and p-value indicators (‘p’, ‘01’). This is not surprising given the reasonably high evidence token AUC achieved by this model.

Overall, despite conditioning and pretraining attention mechanisms, end-to-end models remain over 20 points behind the oracle variant (Table 6. This suggests that the model is failing to sufficiently attend to the relevant evidence. Figure 5 supports this conjecture. This shows the total evidence mass realized by models on the validation set (after training for the final task of evidence inference). Even pretrained models assign $<$14% of total attention mass to actual evidence tokens.

We also explore how token-level discriminative performance varies between pretrained and end-to-end variants (without explicit attention training), and how this changes as learning progresses. Figure 6 plots evidence token AUC over epochs (on the validation set) for attentive model variants (conditioned and unconditioned). We show curves for the case where we use explicit supervision (pretraining; dotted lines) and where relevance is learned only indirectly via the downstream evidence inference objective (no pretraining; solid lines). Interestingly, evidence token AUC reaches maximum values during pretraining (shown as negative epochs) for supervised attention variants, and declines precipitously when the training objective transitions to the downstream task. This suggests a kind of catastrophic forgetting introduced due to shifting objectives.

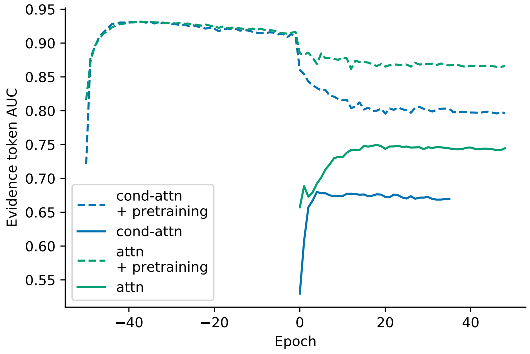


Figure 6: Validation evidence token AUCs during training. 'pretraining' epochs are depicted as 'negative' for the two explicitly supervised attention variants. Note that we use early stopping, so not all models run for the same number of epochs.

7 Related Work

The proposed task is situated at the intersection of information extraction (Cardie, 1997), natural language inference (Bowman et al., 2015), evidence mining (Rinott et al., 2015) and question answering (Harabagiu et al., 2000; Hovy et al., 2000). However, our focus on inferring results from lengthy clinical trial reports pertaining to particular prompts constitutes a unique problem, as discussed in the Introduction.

Prior systems have attempted to extract information from articles describing RCTs. For example, ExaCT (Kiritchenko et al., 2010) attempts to extract variables describing clinical trials from articles, and ACRES (Summerscales et al., 2011) ingests extracts key variables from abstracts. Blake and Lucic (2015; 2012) considered the problem of automatically extracting interventions and outcomes in sentences that report direct comparisons. And Mihaila *et al.* (2013) have proposed annotating and extracting casual statements from biomedical literature. Classifying the *modality* of statements in scientific literature has also been investigated (Thompson et al., 2008); this relates to identifying evidence.

None of these prior efforts attempted to infer the *findings* concerning the extracted interventions and outcomes, as we do here.

8 Conclusions and Future Work

We have presented the task of inferring the polarity of comparative results reported in articles describing clinical trials with respect to interventions and outcomes of interest. Such models would render the unstructured evidence currently buried in manuscripts actionable, in turn potentially informing evidence-based care. In addition to the practical import of this problem, the task poses core NLP challenges related to processing lengthy, technical texts, and performing conditional inference over entities within these.

Our baseline results establish both the feasibility and difficulty of the task. Very simple baselines (e.g., rule-based methods) perform quite poorly, and modern neural architectures achieve the best results, currently. When models are provided with reference evidence spans from an oracle, they achieve dramatically improved performance. This demonstrates that the key challenge concerns conditionally identifying relevant snippets to inform predictions; attention mechanisms would seem to provide a natural means of allowing the model to learn to focus, and we indeed found that (supervised) attention provides some predictive gains, but these are relatively modest.

The gap between the model that directly consumes only relevant evidence snippets (Table 6) and the best performing end-to-end model is over 20 points in F1. Further, ignoring the article entirely (relying only on the prompt) degrades performance by only $\sim$5 points in F1, again suggesting that even the pretrained, conditioned attention variant is not making good use of the relevant evidence contained in articles. The evidence token mass metrics also support this: The best models we have proposed consistently place only $\sim$10-15% of the attention mass on tokens actually marked as containing relevant evidence.

We are simply not learning to attend well, even with explicit pretraining and conditioning. This motivates a key future research direction: designing more sophisticated attention mechanisms that (conditionally) identify spans of evidence pertinent to a given prompt. We hope this corpus and task provides opportunity to pursue such models.

Acknowledgements

This work was supported by NSF CAREER Award 1750978.

We also acknowledge ITS at Northeastern for providing high performance computing resources that have supported this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Hilda Bastian, Paul Glasziou, and Iain Chalmers. 2010. Seventy-five trials and eleven systematic reviews a day: how will we ever keep up? *PLoS Med*, 7(9):e1000326.

Catherine Blake and Ana Lucic. 2015. Automatic endpoint detection to support the systematic review process. *Journal of biomedical informatics*, 56:42–56.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642. ACL.

Claire Cardie. 1997. Empirical methods in information extraction. *AI Magazine*, 18(4):65.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Eunsol Choi, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the Association for Computational Linguistics*. ACL.

Sanda M Harabagiu, Marius A Paşca, and Steven J Maiorano. 2000. Experiments with open-domain textual question answering. In *Proceedings of the Association for Computational Linguistics*, pages 292–298. ACL.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question answering in webclopedia. In *TREC*, volume 52, pages 53–56.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12–22. Association for Computational Linguistics.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. *arXiv preprint arXiv:1808.04926*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Svetlana Kiritchenko, Berry de Bruijn, Simona Carini, Joel Martin, and Ida Sim. 2010. ExaCT: automatic extraction of clinical trial characteristics from journal publications. *BMC medical informatics and decision making*, 10(1):56.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

M Marre, J Shaw, M Brändle, WM Wan Bebakar, Nor Azmi Kamaruddin, J Strand, M Zdravkovic, TD Le Thi, S Colagiuri, and LEAD-1 SU study group. 2009. Liraglutide, a once-daily human glp-1 analogue, added to a sulphonylurea over 26 weeks produces greater improvements in glycaemic and weight control compared with adding rosiglitazone or placebo in subjects with type 2 diabetes (lead-1 su). *Diabetic Medicine*, 26(3):268–278.

Iain J Marshall, Anna Noel-Storr, Joël Kuiper, James Thomas, and Byron C Wallace. 2018. Machine learning for identifying randomized controlled trials: An evaluation and practitioner's guide. *Research synthesis methods*.

Claudiu Mihăilă, Tomoko Ohta, Sampo Pyysalo, and Sophia Ananiadou. 2013. Biocause: Annotating and analysing causality in the biomedical domain. *BMC bioinformatics*, 14(1):2.

Dae Hoon Park and Catherine Blake. 2012. Identifying comparative claim sentences in full-text scientific articles. In *Proceedings of the workshop on detecting structure in scholarly discourse*, pages 1–9. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. *Proceedings of Languages in Biology and Medicine*.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450.

Rodney L Summerscales, Shlomo Argamon, Shangda Bai, Jordan Hupert, and Alan Schwartz. 2011. Automatic summarization of results from clinical trials. In *Proceedings of the International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 372–377. IEEE.

Paul Thompson, Giulia Venturi, John McNaught, Simonetta Montemagni, and Sophia Ananiadou. 2008. Categorising modality in biomedical texts. In *Proceedings of the LREC 2008 Workshop on Building and Evaluating Resources for Biomedical Text Mining*, pages 27–34.

Guy Tsafnat, Adam Dunn, Paul Glasziou, and Enrico Coiera. 2013. The automation of systematic reviews: Would lead to best currently available evidence at the push of a button. *BMJ: British Medical Journal*, 346(7891):8–8.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Omar Zaidan, Jason Eisner, and Christine D Piatko. 2007. Using" annotator rationales" to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. Citeseer.

Ye Zhang, Iain J. Marshall, and Byron C. Wallace. 2016. Rationale-Augmented Convolutional Neural Networks for Text Classification. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 795–804. Associaton for Computational Linguistics (ACL).

Appendix

A Description of Heuristics Baseline

We describe the heuristics implemented in our rule-based baseline model. These can be broken into two stages: (1) finding the sentence that contains the answer, and (2) interpreting the sentence that possesses the evidence. The variant that consumes the evidence spans directly (thus "cheating") uses only rules defined for (2). In both cases we use a 'points' based approach, where simple rules assign points to potential labels.

As an attempt to identify evidence spans, we first split the article into sentences and tokenize these. Each sentence is then assigned a ranking based on the number of words in the outcome, intervention, and comparator that it contains; each map is associated with one point. The sentence with the highest number of points is then designated as being most likely to contain the evidence of interest.

Once this sentence is selected, we use simple checks to try and identify a reported p-value. In particular we search for the following three distinct forms: "$p = X$", "$p > X$", and "$p < X$"; we use a simple RegEx to find instances of these, ignoring whitespace.

If we identify "$p = X$", we then attempt to identify the comparator and intervention from the prompt in the sentence, as the aim is to extract the p-value closest to the intervention or comparator (sentences may contain multiple p-values). If the p-value found is greater than 0.05, we add a point for a label of *no significant difference*. If the p-value is smaller than 0.05, we add a point for a label of *significant difference* (both significantly increases or significantly decreases). If neither the intervention nor the comparator is found in the selected sentence, then we look at all the p-values that it contains and sum points accordingly.

In the case of "$p > X$", we add a point to the label of *no significant difference*, as a p-value greater than some X likely corresponds to a statistically non-significant result.

In the case of "p < 0.05", we add a point to a label of significant difference, as a p-value smaller than some X likely corresponds to varying effects for both the intervention and comparator with respect to the outcome. If there is a tie between points pointing to a significant difference (in either direction) and *no significant difference*, then we return the latter, because it is the majority category.

Otherwise, we assume the result is significant, and next attempt to infer the reported direction of the effect. To do so, we count occurrences of synonyms of the word "increase" in the sentence and compare that with the number of occurrences of synonyms of "decrease". Synonyms are retrieved using WordNet via NLTK. If there are more synonyms of "increase", we return the label *significantly increase*; if there are more synonyms of "decrease", we return *significantly decrease*. If the number of occurrences is equal, we return *significantly increase*, as this designation is slightly more frequent than *significantly decrease*.

B Annotation Costing Details

Generating each prompt cost an average of 99 cents. We thus paid about $4.14 to complete prompt generation for each article. Doctors hired for prompt generation were paid an average of $19.05 an hour.

Prompts took an average of 2.54 minutes to complete. Annotators (again MDs) were paid an average of $13.88 dollars per hour, with each prompt costing 58 cents.

Verification cost a mere 27 cents per prompt. Verifiers were paid, on average, $16.25 an hour.

C Additional Dataset Details

Each prompt has at least two rows, one of which corresponds to the prompt generator's answer, and the other for an annotator's. These rows can be distinguished using the UserID column, in which prompt generator answers are marked with a value of '0', whereas annotator UserIDs are values greater than '0'. Each row also includes the verifier's response, which denotes the validity of that row's answer and rationale. We additionally include the offsets of where the rationale occurs in the text [10]. These offsets are calculated through FuzzyWuzzy, due to the frequent differences in encoding between rationales and extracted XML text.

D Preprocessing Details

To process the PubMed central XML documents, we iterate through by section, and parse subsequent sub-sections. Afterwards, we use an HTML parser library to remove all tags, as these might distract models. We removed all <p> tags that remained due to malformed XML documents.

E Attention (and Attention Pretraining) Variants

On the validation set we explored several substantiations of attention and associatd objectives, in the case of pre-training. These include:

1. **Tokenwise**. This attention variant aims to maximize per-token evidence predictions independently, i.e., no softmax is imposed over attention activations, and so attention weights may range between 0 and 1. During the forward pass, however, we do normalize these weights prior to inducing the context vector.

2. **Balanced tokenwise attention**. This is the same as above, except that during training we construct samples composed of an equal number of evidence and non-evidence tokens (the latter far outnumbering the former, in general).

3. **Evidence mass attention**. Here we attempt to directly optimize total quantity of attention mass placed on evidence tokens.

During pre-training, we run for some number of epochs over available marked evidence spans (we used a maximum of 50, with an early-stopping criterion using a patience of 10). This requires a metric to determine the 'best' observed set of attention weights during pre-training (as measured on a nested validation set). For this we considered a few options for performance measuring criterion over tokens, including evidence mass and entropy. We ultimately settled on monitoring evidence token AUC as a proxy for attention performance over tokens.

F Validation Results and Variances

For completeness in Table 7 we report means and ranges of all model variants explored on the validation dataset (taken over five runs). We include in these the attention variants explained above.

For pretraining variants, trained on the observed evidence spans for 50 epochs prior to shifting objectives to the downstream evidence inference task. We then must select which set of weights, over these 50 epochs, to use for initialization following pretraining. We considered three metrics to score attentional components during pretraining: token-level AUC (overall discriminatory power of the network, with respect to evidence/not-evidence words); entropy (with the intuition that we seek peaky distributions); and evidence token mass (such that most attention is placed on evidence tokens).

We settled on using the tokenwise pretrained variant along with token AUC as our pretraining criterion for the test experiments, but we report full results on the validation dataset in Table 7.

On the test set, we only considered using AUC as the criterion.

---

[10] Offset indices are based on the XML text post preprocessing

Model	Precision	Recall	F1	Token AUC / Mass
+ Attn.	0.499 (0.470, 0.524)	0.499 (0.461, 0.505)	0.483 (0.460, 0.506)	0.741 / 0.047
+ Pretrain attn. [AUC] (Tokenwise attn.)	0.514 (0.500, 0.529)	0.514 (0.484, 0.506)	0.497 (0.485, 0.508)	0.882 / 0.131
+ Pretrain attn. [Entropy] (Tokenwise attn)	0.487 (0.446, 0.516)	0.487 (0.432, 0.493)	0.472 (0.433, 0.491)	0.821 / 0.117
+ Pretrain attn. [Evidence Mass] (Tokenwise attention)	0.501 (0.482, 0.525)	0.501 (0.480, 0.512)	0.493 (0.472, 0.513)	0.861 / 0.128
+ Pretrain attn. [AUC] (Tokenwise attn. balanced)	0.507 (0.500, 0.523)	0.507 (0.485, 0.516)	0.500 (0.486, 0.517)	0.803 / 0.062
+ Pretrain attn. [Entropy] (Tokenwise attn. balanced)	0.489 (0.446, 0.499)	0.489 (0.418, 0.488)	0.470 (0.415, 0.491)	0.704 / 0.046
+ Pretrain attn. [Evidence Mass] (Tokenwise attn. balanced)	0.502 (0.499, 0.522)	0.502 (0.465, 0.522)	0.484 (0.464, 0.522)	0.770 / 0.055
+ Pretrain attn. [AUC] (Max evidence attn.)	0.448 (0.440, 0.470)	0.448 (0.416, 0.467)	0.433 (0.411, 0.467)	0.456 / 0.003
+ Pretrain attn. [Entropy] (Max evidence attn.)	0.458 (0.440, 0.471)	0.458 (0.410, 0.472)	0.436 (0.405, 0.471)	0.413 / 0.000
+ Pretrain attn. [Evidence Mass] (Max evidence attn.)	0.519 (0.514, 0.560)	0.519 (0.491, 0.543)	0.508 (0.492, 0.547)	0.661 / 0.044
+ Cond. attn.	0.509 (0.510, 0.516)	0.509 (0.478, 0.518)	0.498 (0.477, 0.513)	0.667 / 0.055
+ Pretrain cond. attn. [AUC] (Tokenwise attn.)	0.508 (0.500, 0.516)	0.508 (0.488, 0.514)	0.501 (0.488, 0.513)	0.815 / 0.120
+ Pretrain cond. attention [Entropy] (Tokenwise attn.)	0.494 (0.452, 0.525)	0.494 (0.450, 0.521)	0.482 (0.435, 0.517)	0.813 / 0.171
+ Pretrain cond. attn. [Evidence Mass] (Tokenwise attn.)	0.521 (0.509, 0.529)	0.521 (0.501, 0.524)	0.509 (0.499, 0.526)	0.807 / 0.137
+ Pretrain cond. attn. [AUC] (Tokenwise attn. balanced)	0.508 (0.506, 0.520)	0.508 (0.489, 0.514)	0.503 (0.489, 0.515)	0.766 / 0.051
+ Pretrain cond. attn. [Entropy] (Tokenwise attn. balanced)	0.466 (0.430, 0.500)	0.466 (0.425, 0.477)	0.454 (0.425, 0.479)	0.620 / 0.032
+ Pretrain cond. attn. [Evidence Mass] (Tokenwise attn. balanced)	0.516 (0.520, 0.516)	0.516 (0.487, 0.514)	0.504 (0.487, 0.515)	0.770 / 0.064
+ Pretrain cond. attn. [AUC] (Max evidence attn.)	0.495 (0.444, 0.506)	0.495 (0.436, 0.499)	0.482 (0.436, 0.500)	0.617 / 0.042
+ Pretrain cond. attn. [Entropy] (Max evidence attn.)	0.442 (0.428, 0.464)	0.442 (0.414, 0.463)	0.435 (0.413, 0.460)	0.462 / 0.004
+ Pretrain cond. attn. [Evidence Mass] (Max evidence attn.)	0.493 (0.474, 0.503)	0.493 (0.474, 0.504)	0.481 (0.470, 0.497)	0.636 / 0.039

Table 7: Means and (min, max) values across five runs over the validation set for attention variants and associated (selection criteria). The latter is the metric used to pick the best set of weights after 50 pretraining epochs with which to seed the end-task training process.

Decay-Function-Free Time-Aware Attention to Context and Speaker Indicator for Spoken Language Understanding

Jonggu Kim
Computer Science and Engineering,
Pohang University of Science and
Technology (POSTECH)
Pohang, Republic of Korea
`jgkimi@postech.ac.kr`

Jong-Hyeok Lee
Computer Science and Engineering,
Pohang University of Science and
Technology (POSTECH)
Pohang, Republic of Korea
`jhlee@postech.ac.kr`

Abstract

To capture salient contextual information for spoken language understanding (SLU) of a dialogue, we propose time-aware models that automatically learn the latent time-decay function of the history without a manual time-decay function. We also propose a method to identify and label the current speaker to improve the SLU accuracy. In experiments on the benchmark dataset used in Dialog State Tracking Challenge 4, the proposed models achieved significantly higher F1 scores than the state-of-the-art contextual models. Finally, we analyze the effectiveness of the introduced models in detail. The analysis demonstrates that the proposed methods were effective to improve SLU accuracy individually.

1 Introduction

Spoken language understanding (SLU) is a component that understands the user's utterance of a dialogue system. Given an utterance, SLU generates a structured meaning representation of the utterance; i.e., a semantic frame. SLU can be decomposed into several subtasks such as domain identification, intent prediction and slot filling; these subtasks can be jointly assigned using a single model (Hakkani-Tür et al., 2016; Liu and Lane, 2016; Chen et al., 2016b). The accuracy of SLU is important for the dialogue system to generate an appropriate response to a user.

To improve the accuracy of SLU, much work has used contextual information of dialogues to alleviate the ambiguity of recognition of the given utterance. In SLU, selecting important history information is crucial, and it directly influences the improvement of SLU accuracy. To summarize this history, content-aware models (Chen et al., 2016a; Kim et al., 2017) similar to attention models in machine translation (Bahdanau et al., 2014) have been proposed. However, content-aware models are likely to select the wrong history when the histories are similar in content. To alleviate this problem, time-aware models (Chen et al., 2017; Su et al., 2018a,b) which pay attention to recent previous utterances by using the temporal distance between a previous utterance and a current utterance are being considered; the models are based on mathematical formulas, time-decay functions, which are formulated by human, and decomposed into trainable parameters.

However, the previous time-aware models may not be sufficiently accurate. In the models, either a single time-decay function is used or a limited number of time-decay functions are linearly combined; these manual functions may not be sufficiently flexible to learn an optimal time-decay function.

In this paper, we propose flexible and effective time-aware attention models to improve SLU accuracy. The proposed models do not need any manual time-decay function, but learn a time-decay tendency directly by introducing a trainable distance vector, and therefore have good SLU accuracy. The proposed models do not use long short-term memory (LSTM) to summarize histories, and therefore use fewer parameters than previous time-aware models. We also propose current-speaker modeling by using a speaker indicator that identifies the current speaker.

To the best of our knowledge, this is the first method that shows improvement by considering the identity of the current speaker. This information may be helpful for modeling multi-party conversations in addition to human-human conversations.

Prediction of the semantic label of the current utterance even using a conventional time-aware model can be difficult. (Figure 1). The nearest utterance is "Right.", but it is not the most rele-

Proceedings of NAACL-HLT 2019, pages 3718–3726
Minneapolis, Minnesota, June 2 - June 7, 2019.

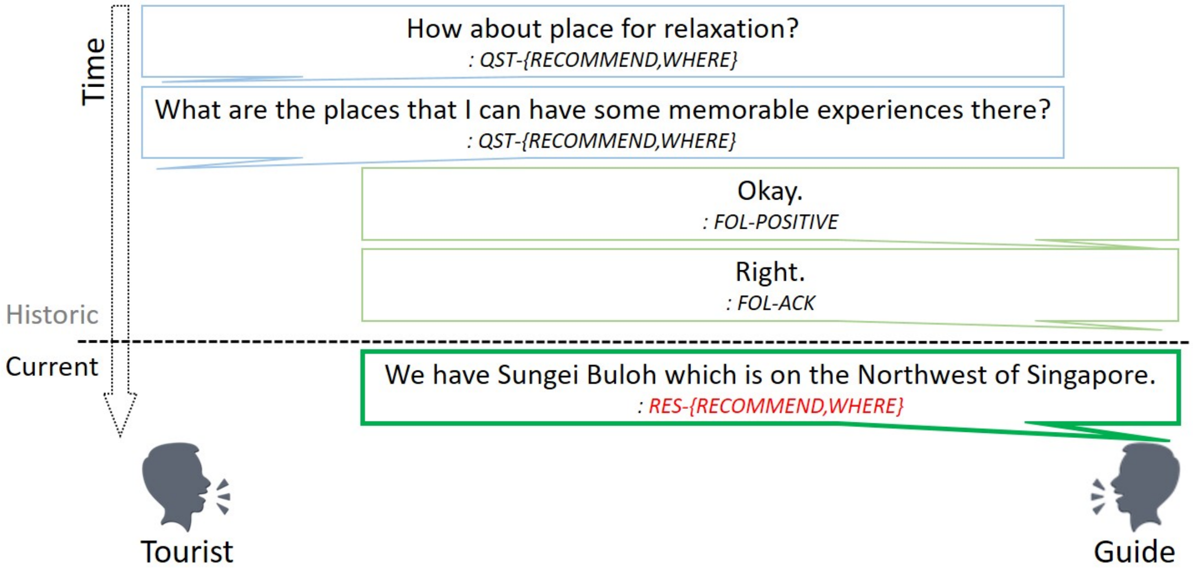


Figure 1: An example of utterances with their semantic labels (speech acts combined with associated attributes) from DSTC 4. The semantic labels are italicized.

vant utterance to the current utterance; the most relevant utterance is "What are the places that I can have some memorable experiences there?". If we do not know the current speaker is *Guide*, we cannot easily assess the relative importance of the nearest histories of the two speakers. We believe that the proposed 'speaker indicator' can help our model to identify such information.

In experiments on the Dialog State Tracking Challenge 4 (DSTC 4) dataset, the proposed models achieved significantly higher accuracy than the state-of-the-art contextual models for SLU. Also, we examine how the proposed methods affect the SLU accuracy in detail. This result shows that the proposed methods were effective to improve SLU accuracy individually. Our contributions are as follows:

- We propose a decay-function-free time-aware attention model that automatically learn the latent time-decay function of the history without a manual time-decay function. The proposed model achieves a new state-of-the-art F1 score.
- We propose a current-speaker modeling method that uses a speaker indicator to identify the current speaker. We present how to incorporate speaker indicator in the proposed attention model for further improvement of SLU accuracy.
- We propose a model that is aware of content as well as time, which also achieved a higher F1 score than the state-of-the-art contextual models.
- We analyze the effectiveness of proposed methods in detail.

Our source code to reproduce the experimental results is available at `https://github.com/jgkimi/Decay-Function-Free-Time-Aware`.

2 Related Work

Joint semantic frame parsing has the goal of learning intent prediction and slot filling jointly. By joint learning, the model learns their shared features, and this ability is expected to improve the accuracy on both tasks. A model based on bidirectional LSTM for joint semantic frame parsing (Hakkani-Tür et al., 2016) is trained on the two tasks in sequence, by adding an intent label to the output of the final time-step of LSTM. Similarly, an attention-based LSTM predicts slot tags for each time-step, then feeds the hidden vectors and their soft-aligned vectors to a fully-connected layer for intent prediction (Liu and Lane, 2016). Knowledge-guided joint semantic frame parsing (Chen et al., 2016b) incorporates syntax or semantics-level parsing information into a model by using a recurrent neural network (RNN) for joint semantic frame parsing.

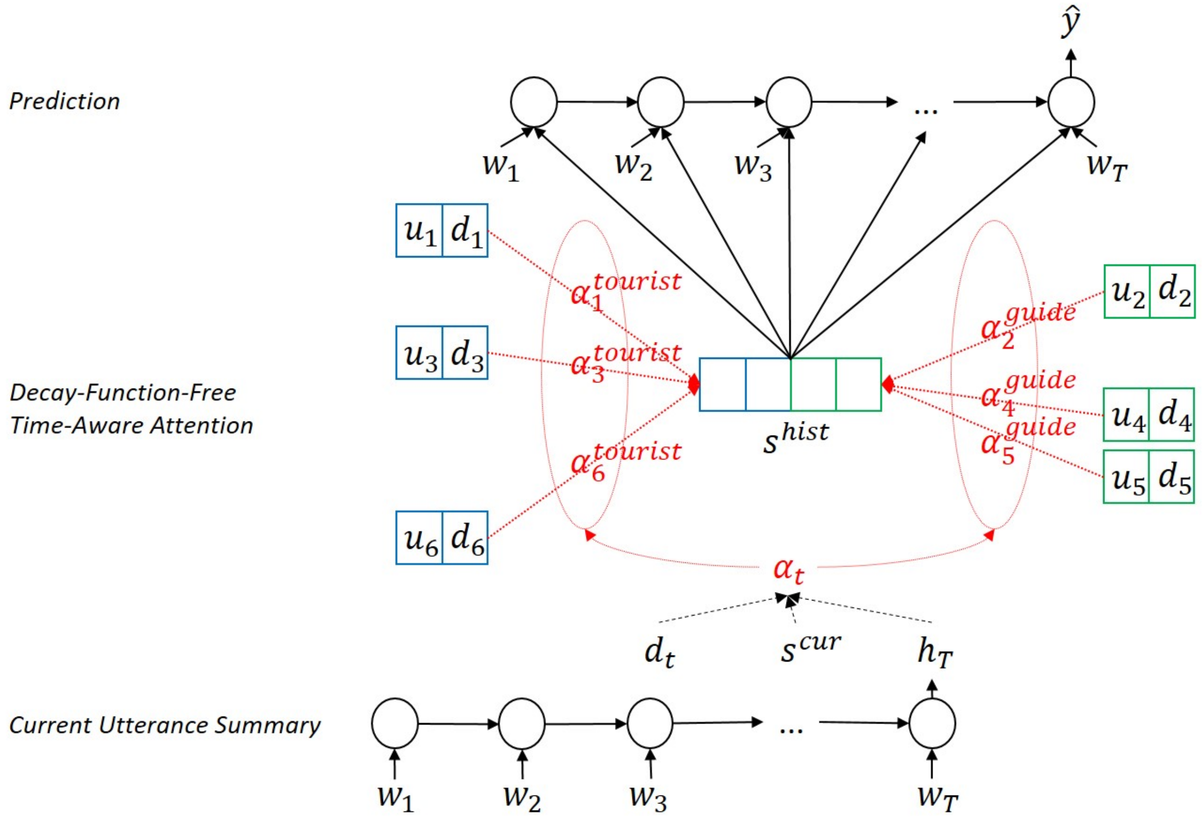


Figure 2: Overall architecture of the decay-function-free time-aware attention with speaker indicator (role-level). $w_1, ..., w_T$ are word vectors of the current utterance, d_t is the t^{th} distance vector, u_t is the t^{th} utterance vector, s^{cur} is the current speaker indicator, h_T is the current utterance summary vector and α_t is the importance of the t^{th} historic utterance. For simplicity, we represent bidirectional LSTM layers as unidirectional LSTM layers.

Other research on SLU uses context information. A model based on support vector machine and a hidden Markov model uses contextual information to show the importance of contextual information in SLU tasks, intent prediction and slot detection (Bhargava et al., 2013). RNN-based models can exploit context to classify domains (Xu and Sarikaya, 2014), and have been combined with previously-estimated intent and slot labels to predict domain and intent (Shi et al., 2015). A memory network that contains historic utterance vectors encoded by RNN has been used to select the most relevant history vector by multiplicative soft-attention (Chen et al., 2016a); the selected vector is fed to an RNN-based slot tagger as context information.

A memory network can be regarded as use of content-based similarity between the current utterance and previous utterances. A memory network can be separated to capture historic utterances for each speaker independently (Kim et al., 2017), and a contextual model can use different LSTM layers to separately encode a history summary for each speaker (Chi et al., 2017). For another task, addressee and response selection in multi-party conversations, a distinct RNN-based encoder for each speaker-role (sender, addressee, or observer) has been used to generate distinct history summaries (Zhang et al., 2018).

Recent work on contextual SLU has introduced time information of contexts into models because content-based attention may cause a wrong choice that introduce noises. The reciprocal of temporal distance between a current utterance and contexts can be used as a time-decay function, and the function can be decomposed into trainable parameters (Chen et al., 2017). Similarly, a universal time-aware attention model (Su et al., 2018a) has been proposed; it is a trainable linear combination of three distinct (convex, linear and concave) time-decay functions . An extension of this model is a context-sensitive time-decay attention (Su et al., 2018b) that generates its parameters from the current utterance by using a fully-connected layer so that the content information of the current utterance is also considered in the attention.

3 Proposed Model

We propose a time-aware model that includes a speaker indicator (Figure 2). In addition, we propose a content-and-time-aware model that includes a speaker indicator. The models are trained in an end-to-end way, in which every model parameter is automatically learned based on a downstream SLU task. The objective of the proposed models are to optimize the conditional probability of labels of SLU, given the current utterance $p(\hat{y}|x)$, by minimizing the cross-entropy loss.

The following description of the proposed model considers three steps: current utterance summary, context summary, and prediction.

3.1 Current Utterance Summary

To select salient parts of contextual histories, the current utterance is used. To summarize a current utterance matrix U that consists of words w_i as a vector (i.e., $U = \{w_1 , w_2,..., w_T\}$), U is fed to bidirectional LSTMs, and the final hidden vector $h_T \in \mathbb{R}^{dim}$ is taken as a current utterance summary.

3.2 Decay-Function-Free Time-Aware Attention

In this subsection, we introduce a decay-function-free time-aware attention model. To summarize contexts, we use a time difference (distance) between a historic utterance and the current utterance; this distance represents the interval between the historic utterance and the current utterance. We use the distance of the t^{th} history from the current utterance as an index to select a dense vector from a distance-embedding matrix $D \in \mathbb{R}^{dim\times|D|}$, then use the vector as the t^{th} distance vector d_t.

To compute the importance α_t of the t^{th} history, both in the sentence-level attention and in the role-level attention, our time-aware attention uses the current utterance summary h_T and the history distance d_t simultaneously and additively:

$$\alpha_t = w_{att}^T \sigma\big(h_T + d_t + b_{att}\big), \qquad (1)$$

where w_{att}^T is the transpose of a trainable weight vector for the attention, b_{att} is a trainable bias vector for the attention, and σ is the hyperbolic tangent function.

Computing a time-aware context summary vector s_{time}^{hist} depends on whether the role-level or sentence-level attention is considered. For the role-level attention, we use the softmax operation applied to all α_t of the same speaker, either a guide or a tourist, to obtain a role-level probabilistic importance α_t^{role} of t^{th} history. We then multiply α_t^{role} by t^{th} history vector, which is a concatenation of the corresponding intent-dense vector u_t and the distance vector d_t. We use the element-wise sum of the vectors of the same speaker to construct two summary vectors s_{time}^{guide} and $s_{time}^{tourist}$. Finally, s_{time}^{guide} and $s_{time}^{tourist}$ are concatenated to form a time-aware history summary vector s_{time}^{hist} as:

$$\alpha_t^{role} = softmax_{role}(\alpha_t), \qquad (2)$$

$$s_{time}^{role} = \sum_t \alpha_t^{role}(u_t \oplus d_t), \qquad (3)$$

$$s_{time}^{hist} = s_{time}^{guide} \oplus s_{time}^{tourist}, \qquad (4)$$

where $\oplus$ represents a concatenation operation.

For the sentence-level attention to obtain a sentence-level probabilistic importance α_t^{sent} of t^{th} history, we use the softmax operation applied to all α_t regardless of the speaker, then multiply α_t^{sent} by the t^{th} history vector, which is a concatenation of the corresponding intent dense vector u_t and the distance vector d_t. We use the element-wise sum of the vectors to construct a time-aware summary vector s_{time}^{hist} as:

$$\alpha_t^{sent} = softmax_{sent}(\alpha_t), \qquad (5)$$

$$s_{time}^{hist} = \sum_t \alpha_t^{sent}(u_t \oplus d_t). \qquad (6)$$

Then, s_{time}^{hist} is used as a context summary s^{hist} in the prediction step.

3.3 Decay-Function-Free Content-and-Time-Aware Attention

Although a time-aware attention model is powerful by itself, content can be considered at the same time to improve accuracy. We propose another contextual attention model that is aware of content, in addition to time. This model is called content-and-time-aware attention. The model uses an importance value β_t for the t^{th} history. To compute β_t, we uses the trainable parameters w_{att} and b_{att} of the time attention as:

$$\beta_t = w_{att}^T \sigma\big(h_T + u_t + b_{att}\big), \qquad (7)$$

where u_t is the intent dense vector of t^{th} history, and σ is the hyperbolic tangent function.

Then, β_t is used in the same way as α_t, but independently. s^{hist}_{time} is computed as in the previous subsection, β_t is used to compute s^{hist}_{cont} for the role-level attention as:

$$\beta^{role}_t = softmax_{role}(\beta_t), \tag{8}$$

$$s^{role}_{cont} = \sum_t \beta^{role}_t (u_t \oplus d_t), \tag{9}$$

$$s^{hist}_{cont} = s^{guide}_{cont} \oplus s^{tourist}_{cont}. \tag{10}$$

To compute s^{hist}_{cont} for the sentence-level attention, β_t is used as:

$$\beta^{sent}_t = softmax_{sent}(\beta_t), \tag{11}$$

$$s^{hist}_{cont} = \sum_t \beta^{sent}_t (u_t \oplus d_t). \tag{12}$$

Finally, the time-aware history summary s^{hist}_{time} and the content-aware history summary s^{hist}_{cont} are concatenated to generate a history summary s^{hist} regardless of the attention level:

$$s^{hist} = s^{hist}_{time} \oplus s^{hist}_{cont}. \tag{13}$$

3.4 Speaker Indicator

Speaker indicator is a trainable vector $s^{cur} \in \mathbb{R}^{dim}$ which indicates the identity of the current speaker; i.e., either a tourist or a guide in DSTC 4. An embedding lookup method is used after a speaker embedding matrix $S \in \mathbb{R}^{dim \times |S|}$ is defined. The speaker embedding matrix is randomly initialized before the model is trained.

To use speaker indicator s^{cur} in the proposed attentions, Eq. 1 is rewritten as:

$$\alpha_t = w^T_{att}\sigma\big(h_T + d_t + s^{cur} + b_{att}\big), \tag{14}$$

and Eq. 7 is rewritten as:

$$\beta_t = w^T_{att}\sigma\big(h_T + u_t + s^{cur} + b_{att}\big). \tag{15}$$

3.5 Prediction

To predict the true label in spoken language understanding, our model consumes the current utterance U again. We use another bidirectional LSTM layer which is distinct from that of the current utterance summary. To prepare for t^{th} input v_t of the LSTM layer, we concatenate t^{th} word vector w_t of the current utterance U with the history summary vector s^{hist}:

$$v_t = w_t \oplus s^{hist}. \tag{16}$$

Then, we feed each v_t to the LSTM layer sequentially, and the final hidden vector of the LSTM layer is used as an input of a feed-forward layer to predict the true label $\hat{y}$.

4 Experiments

To test the proposed models, we conducted language-understanding experiments on a dataset of human-human conversations.

4.1 Dataset and Settings

We conducted experiments on the DSTC 4 dataset which consists of 35 dialogue sessions on touristic information for Singapore; they were collected from Skype calls of three tour guides with 35 tourists. The 35 dialogue sessions total 21 h, and include 31,034 utterances and 273,580 words (Kim et al., 2016). DSTC 4 is a suitable benchmark dataset for evaluation, because all of the dialogues have been manually transcribed and annotated with speech acts and semantic labels at each turn level. a semantic label consists of a speech act and associated attribute(s). The speaker information (guide and tourist) is also provided. Human-human dialogues contain rich and complex human behaviors and bring much difficulty to all tasks that are involved in SLU. We used the same training dataset, the same test dataset and the same validation set as in the DSTC 4 competition: 14 dialogues as the training dataset, 6 dialogues as the validation dataset, and 9 dialogues as the test dataset.

We used Adam (Kingma and Ba, 2015) as the optimizer in training the model. We set the batch size to 256, and used pretrained 200-dimensional word embeddings GloVe (Pennington et al., 2014). We applied 30 training epochs with early stopping. We set the size dim of every hidden layer to 128, and the context length to 7. We used the ground truth intents (semantic labels) to form an intent-dense vector like previous work. To evaluate SLU accuracy, we used the F1 score, which is the harmonic mean of precision and recall. To validate the significance of improvements, we used a one-tailed t-test. We ran each model ten times, and report their average scores.

As baseline models, we used the state-of-the-art contextual models, and most accurate participant of DSTC 4 (DSTC 4 - Best) (Kim et al., 2016). For comparison with our models, we used the scores

Model	F1 score Sent.-Level	 Role-Level
DSTC 4 - Best	*61.4*	
No Context	65.06	
LSTM-Used Context Summary without Attention	72.15	
LSTM-Used Content-Aware Attention	71.27	71.84
Speaker Role Modeling (Chi et al., 2017)	*66.8*	*70.1*
Convex Time-Aware Attention (Chen et al., 2017)	*74.6*	*74.2*
Universal Time-Aware Attention (Su et al., 2018a)	74.22	74.12
Universal Content + Time Attention (Su et al., 2018a)	74.40	74.33
Context-Sensitive Time Attention (Su et al., 2018b)	74.20	73.53
Decay-Function-Free Time-Aware Attention	75.58**	75.58**
with Speaker Indicator	75.95**	**76.56****
Decay-Function-Free Content-and-Time-Aware Attention	75.59**	75.30**
with Speaker Indicator	76.11**	76.14**

Table 1: SLU accuracy on DSTC 4. *: $p < 0.05$; **, $p < 0.01$ compared to all the baseline models. Italicized scores are reported in the references. Model names are described in the text.

reported in the papers[1]. We ran three additional baseline models in which the prediction stage is the same: (1) 'No Context' uses no context summary; (2) 'LSTM-Used Context Summary without Attention' uses the context summary of bidirectional LSTM without an attention mechanism, and (3) 'LSTM-Used Content-Aware Attention' uses context summary of bidirectional LSTM after content-aware attention is applied to histories, as in previous approaches.

4.2 Results

We conducted an experiment to compare the proposed models with the baseline models in the SLU accuracy (Table 1). All of the proposed models achieved significant improvements compared to all the baseline models.

We conducted an experiment to identify details of how possible combinations of the proposed methods affect the SLU accuracy (Table 2). In addition to the combinations of the proposed methods, we tested another content-and-time-aware attention method (Content x Time) which computes attention values using both intent and distance at a time, and shares the values to compare with the proposed content-and-time-aware attention.

5 Discussion

In the first subsection, we analyze the effectiveness of the decay-function-free time-aware attention and decay-function-free content-and-time-aware attention by comparison with others. In the next subsection, we analyze the effectiveness of the proposed methods in their possible combinations. We also analyze the effectiveness of the use of a distance vector in the history representation under the various conditions. Finally, we analyze attention weights of the proposed models in a qualitative way to convince of the effectiveness of them.

We also conducted an experiment to show the effectiveness of the use of a distance vector in the history representation under the same condition as in the role-level attention (Table 3). Although we propose to use both intent and distance by concatenating them as a history representation, intent can be used alone; this approach is more intuitive than using both intent and distance.

5.1 Comparison with Baseline Models

In Table 1, Decay-Function-Free Time-Aware Attention and Decay-Function-Free Content-and-Time-Aware Attention achieved significantly higher F1 scores that all baseline models. Especially, the role-level Decay-Function-Free Time-Aware Attention with speaker indicator achieved an F1 score of 76.56 % (row 11), which is a state-of-the-art SLU accuracy.

---

[1] Su et al. (2018a) and Su et al. (2018b) specified that they used different training/valid/test datasets that had been randomly selected from the whole DSTC 4 data with different rates for the experiments. Therefore, we do not use the reported score in our comparison, but produced the results under the same conditions by using the open-source code.

Attention Type	F1 score Sent.-Level	 Role-Level
no attention	70.49	70.43
Content	73.03	72.87
Content	73.05	72.68
Time	75.58	75.58
Time	75.95**	**76.56****
Content + Time	75.59	75.30
Content + Time	75.83	75.96**
Content + **Time**	75.94*	75.97**
Content + Time	76.11**	76.14**
Content x Time	75.59	75.50
Content x Time	75.63	75.64

Table 2: SLU accuracy of possible combinations of the proposed methods. "no attention": sum of all history vectors without calculating α, "Content" is content-aware attention (Decay-Function-Free Content-Aware Attention), "Time" is the proposed time-aware attention (Decay-Function-Free Time-Aware Attention), "Content + Time" is the proposed content-and-time-aware attention (Decay-Function-Free Content-and-Time-Aware Attention), "Content x Time" is variant content-and-time-aware attention (Decay-Function-Free Inseparate Content-and-Time-Aware Attention). *: $p < 0.05$; **, $p < 0.01$ compared to the same attention without speaker indicator. An attention type in bold is the speaker-involved part.

Attention Type	F1 score Intent only	 Int. & Dist.
no attention	70.20	70.43
Content	71.09	72.87**
Content	71.26	72.68**
Time	75.17	75.58**
Time	75.11	**76.56****
Content + Time	75.04	75.30**
Content + Time	75.62	75.96*
Content + **Time**	75.13	75.97**
Content + Time	75.67	76.14**
Content x Time	75.08	75.50**
Content x Time	75.03	75.64**

Table 3: SLU accuracy of possible combinations of the proposed methods in role-level attention with different history representations. Int. used intent vector; Dist. used distance vector. *: $p < 0.05$; **, $p < 0.01$ compared to using intent only. Other codes are as in Table 2.

5.2 Detailed Analysis on Proposed Methods

The proposed methods had good SLU accuracy (Table 2). Every time-aware attention with and without speaker indicator (rows 4 to 11) improved the F1 score compared to the content-aware attention with and without speaker indicator (rows 2 and 3) and to no attention (row 1). This result means that the proposed time-aware attention was effective to improve the SLU accuracy. Any of the content-and-time-aware attention with or without speaker indicator (rows 6 to 11) did not improve the F1 score compared to the time-aware attention with and without speaker indicator (rows 4 and 5). This result means that incorporating content could not make further improvement of the accuracy. Also, without speaker indicator, all the time-aware attention (rows 4, 6 and 10) achieved similar F1 scores.

Use of speaker indicator also showed tendencies. It did not significantly improve the SLU accuracy of Decay-Function-Free Content-Aware Attention (rows 2 and 3) or Decay-Function-Free Inseparate Content-and-Time-Aware Attention (rows 10 and 11), but did improve the accuracy of the proposed models, Decay-Function-Free Time-Aware Attention (rows 4 and 5) and Decay-Function-Free Content-and-Time-Aware Attention (rows 6 to 9). Decay-Function-Free Content-and-Time-Aware Attention with speaker indicator (rows 7 to 9) were more accurate than Decay-Function-Free Inseparate Content-and-Time-Aware Attention with speaker indicator (row 11). This result means that using speaker indicator, separation of content and time improved the accuracy. The improvement in the role-level tended to be greater than that in the sentence-level. The improvement was greatest when speaker indicator was involved in the proposed role-level Decay-Function-Free Time-Aware Attention (row 5).

5.3 Effectiveness of Use of Distance in History Representation

In all models, the use of both intent and distance vectors significantly achieved higher F1 than the use of an intent vector only (Table 3). The results indicate that distance embeddings are helpful both for attention and for the history representation. Decay-Function-Free Time-Aware Attention achieved the biggest improvement (row 5) among all the models.

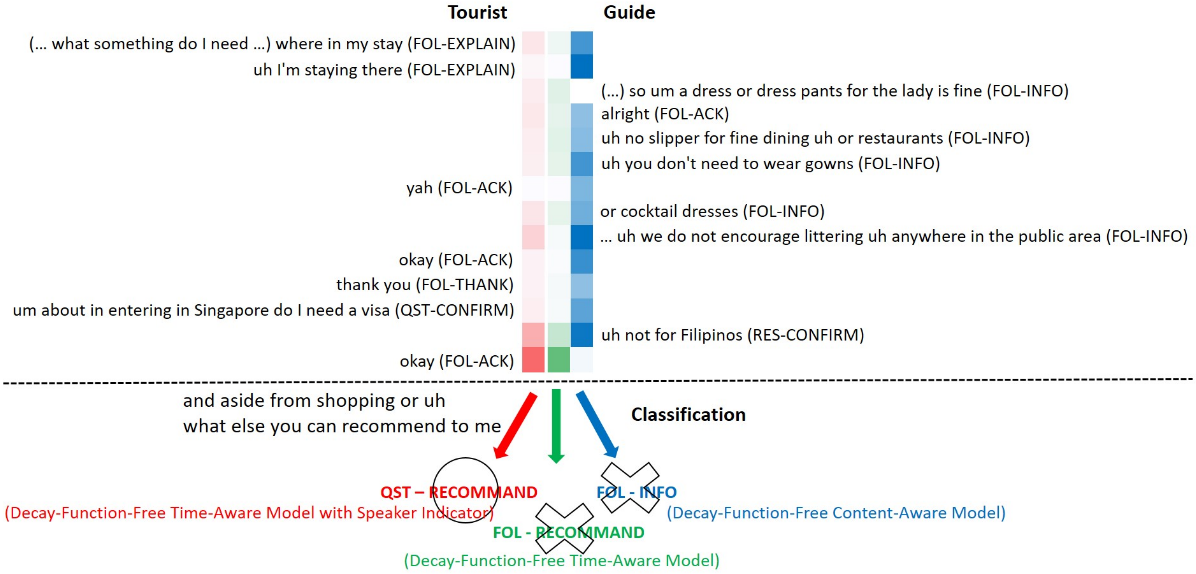


Figure 3: The visualization of the attention weights of the proposed models and the baseline content-aware model. Color gradient indicates intensity given a single datum after training. The color gradient at the left side indicates attention intensities of Decay-Function-Free Time-Aware Model with Speaker Indicator, the color gradient in the center indicates attention intensities of Decay-Function-Free Time-Aware Model without Speaker Indicator, and the color gradient at the right side indicates attention intensities of Decay-Function-Free Content-Aware Model.

5.4 Qualitative Analysis

To assess whether the proposed time-aware attention and speaker indicator can learn a time-decay tendency of the history effectively, we inspected the weights trained in Decay-Function-Free Time-Aware Attention with and without the speaker indicator. We also inspected Decay-Function-Free Content-Aware Attention to compare with them. We observed (Figure 3) that the weights of the proposed models were trained well compared to Decay-Function-Free Content-Aware Attention. The proposed time-aware attention with/without speaker indicator tended to pay attention to recent histories, whereas the content-aware attention does not. As a result, Decay-Function-Free Time-Aware Attention with speaker indicator could generate the true label, *QST-RECOMMEND*, by avoiding noisy contextual information like "uh I'm staying there (FOL-EXPLAIN)" or "... uh we do not encourage littering uh anywhere in the public area (FOL-INFO)".

6 Conclusion

In this paper, we propose decay-function-free time-aware attention models for SLU. These models summarize contextual information by taking advantage of temporal information without a manual time-decay function. We also propose a current-speaker detector that identifies the current speaker. In experiments on the DSTC 4 benchmark dataset, the proposed models achieved a state-of-the-art SLU accuracy. Detailed analysis of effectiveness of the proposed methods demonstrated that the proposed methods increase the accuracy of SLU individually.

Acknowledgments

We would like to thank the reviewers for their insightful and constructive comments on this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

A. Bhargava, A. Celikyilmaz, D. Hakkani-Tür, and R. Sarikaya. 2013. Easy contextual intent prediction and slot detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8337–8341.

P. Chen, T. Chi, S. Su, and Y. Chen. 2017. Dynamic time-aware attention to speaker roles and contexts for spoken language understanding. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 554–560.

Yun-Nung Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016a. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech 2016*, pages 3245–3249.

Yun-Nung (Vivian) Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. 2016b. Syntax or semantics? knowledge-guided joint semantic frame parsing. IEEE Workshop on Spoken Language Technology (SLT 2016).

Ta-Chung Chi, Po-Chun Chen, Shang-Yu Su, and Yun-Nung Chen. 2017. Speaker role contextual modeling for language understanding and dialogue policy learning. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 163–168.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech 2016*, pages 715–719.

Seokhwan Kim, Rafael E Banchs Luis Fernando DHaro, Jason D Williams, and Matthew Henderson. 2016. The fourth dialog state tracking challenge. In *Proceedings of IWSDS*.

Young-Bum Kim, Sungjin Lee, and Ruhi Sarikaya. 2017. Speaker-sensitive dual memory networks for multi-turn slot tagging. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 541–546.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference on Learning Representations (ICLR)*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*, pages 685–689.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Y. Shi, K. Yao, H. Chen, Y. Pan, M. Hwang, and B. Peng. 2015. Contextual spoken language understanding using recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5271–5275.

Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. 2018a. How time matters: Learning time-decay attention for contextual spoken language understanding in dialogues. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. 2018b. Learning context-sensitive time-decay attention for role-based dialogue modeling. *arXiv preprint arXiv:1809.01557*.

P. Xu and R. Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140.

Rui Zhang, Honglak Lee, Lazaros Polymenakos, and Dragomir Radev. 2018. Addressee and response selection in multi-party conversations with speaker interaction rnns. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Dialogue Act Classification with Context-Aware Self-Attention

Vipul Raheja   **Joel Tetreault**
Grammarly
`firstname.lastname@grammarly.com`

Abstract

Recent work in Dialogue Act classification has treated the task as a sequence labeling problem using hierarchical deep neural networks. We build on this prior work by leveraging the effectiveness of a context-aware self-attention mechanism coupled with a hierarchical recurrent neural network. We conduct extensive evaluations on standard Dialogue Act classification datasets and show significant improvement over state-of-the-art results on the Switchboard Dialogue Act (SwDA) Corpus. We also investigate the impact of different utterance-level representation learning methods and show that our method is effective at capturing utterance-level semantic text representations while maintaining high accuracy.

1 Introduction

Dialogue Acts (DAs) are the functions of utterances in dialogue-based interaction (Austin, 1975). A DA represents the meaning of an utterance at the level of illocutionary force, and hence, constitutes the basic unit of linguistic communication (Searle, 1969). DA classification is an important task in Natural Language Understanding, with applications in question answering, conversational agents, speech recognition, etc. Examples of DAs can be found in Table 1. Here we have a conversation of 7 utterances between two speakers. Each utterance has a corresponding label such as *Question* or *Backchannel*.

Speaker	Utterance	DA label
A	Okay.	Other
A	Um, what did you do this weekend?	Question
B	Well, uh, pretty much spent most of my time in the yard.	Statement
B	[Throat Clearing]	Non Verbal
A	Uh-Huh.	Backchannel
A	What do you have planned for your yard?	Question
B	Well, we’re in the process of, revitalizing it.	Statement

Table 1: A snippet of a conversation sample from the SwDA Corpus. Each utterance has a corresponding dialogue act label.

Early work in this field made use of statistical machine learning methods and approached the task as either a structured prediction or text classification problem (Stolcke et al., 2000; Ang et al., 2005; Zimmermann, 2009; Surendran and Levow, 2006). Many recent studies have proposed deep learning models for the DA classification task with promising results (Lee and Dernoncourt, 2016; Khanpour et al., 2016; Ortega and Vu, 2017). However, most of these approaches treat the task as a text classification problem, treating each utterance in isolation, rendering them unable to leverage the conversation-level contextual dependence among utterances. Knowing the text and/or the DA labels of the previous utterances can assist in predicting the current DA state. For instance, in Table 1, the *Answer* or *Statement* dialog acts often follow *Question* type utterances.

This work draws from recent advances in NLP such as self-attention, hierarchical deep learning models, and contextual dependencies to produce a dialogue act classification model that is effective across multiple domains. Specifically, we propose a hierarchical deep neural network to model different levels of utterance and dialogue act semantics, achieving state-of-the-art performance on the Switchboard Dialogue Act Corpus. We demonstrate how performance can improve by leveraging context at different levels of the model: previous labels for sequence prediction (using a CRF), conversation-level context with self-attention for utterance representation learning, and character embeddings at the word-level. Finally, we explore different ways to learn effective utterance repre-

Proceedings of NAACL-HLT 2019, pages 3727–3733
Minneapolis, Minnesota, June 2 - June 7, 2019.

sentations, which serve as the building blocks of our hierarchical architecture for DA classification.

2 Related Work

A full review of all DA classification methods is outside the scope of the paper, thus we focus on two main classes of approaches which have dominated recent research: those that treat DA classification as a text classification problem, where each utterance is classified in isolation, and those that treat it as a sequence labeling problem.

Text Classification: Lee and Dernoncourt (2016) build a vector representation for each utterance, using either a CNN or RNN, and use the preceding utterance(s) as context to classify it. Their model was extended by Khanpour et al. (2016) and Ortega and Vu (2017). Shen and Lee (2016) used a variant of the attention-based encoder for the task. Ji et al. (2016) use a hybrid architecture, combining an RNN language model with a latent variable model.

Sequence Labeling: Kalchbrenner and Blunsom (2013) used a mixture of sentence-level CNNs and discourse-level RNNS to achieve state-of-the-art results on the task. Recent works (Li and Wu, 2016; Liu et al., 2017) have increasingly employed hierarchical architectures to learn and model multiple levels of utterance and DA dependencies. Kumar et al. (2018), Chen et al. (2018) and Tran et al. (2017) used RNN-based hierarchical neural networks, using different combinations of techniques like last-pooling or attention mechanism to encode sentences, coupled with CRF decoders. Chen et al. (2018) achieved the highest performance to date on the two datasets for this task.

Our work extends these hierarchical models and leverages a combination of techniques proposed across these prior works (CRF decoding, contextual attention, and character-level word embeddings) with self-attentive representation learning, and is able to achieve state-of-the-art performance.

3 Model

The task of DA classification takes a conversation C as input, which is a varying length sequence of utterances $U = \{u_1, u_2, ...u_L\}$. Each utterance $u_i \in U$, in turn, is a sequence of varying lengths of words $\{w_i^1, w_i^2, ..., w_i^{N_i}\}$, and has a corresponding target label $y_i \in Y$. Hence, each conversation (i.e. a sequence of utterances) is mapped to a corresponding sequence of target labels $Y = \{y_1, y_2, ..., y_L\}$, which represents the DAs associated with the corresponding utterances.

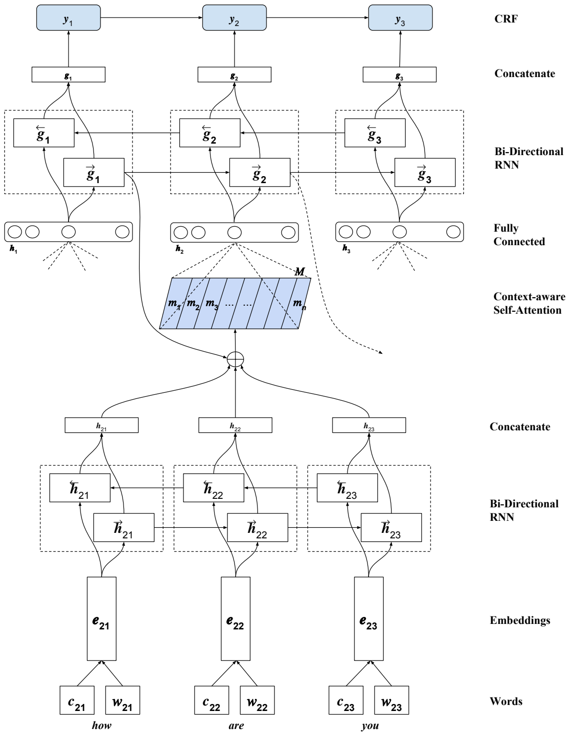


Figure 1: Model Architecture

Figure 1 shows the overall architecture of our model, which involves three main components: (1) an utterance-level RNN that encodes the information within the utterances at the word and character-level; (2) a context-aware self-attention mechanism that aggregates word representations into utterance representations; and (3) a conversation-level RNN that operates on the utterance encoding output of the attention mechanism, followed by a CRF layer to predict utterance labels. We describe them in detail below.

3.1 Utterance-level RNN

For each word in an utterance, we combine two different word embeddings: GloVe (Pennington et al., 2014) and pre-trained ELMo representations (Peters et al., 2018) with fine-tuned task-specific parameters, which have shown superior performance in a wide range of tasks. The word embedding is then concatenated with its CNN-based 50-D character-level embedding (Chiu and Nichols, 2016; Ma and Hovy, 2016) to get the complete word-level representations. The motivation behind incorporating subword-level information is to infer the lexical features of utterances and named entities better.

The word representation layer is followed by a bidirectional GRU (Bi-GRU) layer. Concatenating the forward and backward outputs of the Bi-GRU generates the utterance embedding that serves as input to the utterance-level context-aware self-attention mechanism which learns the final utterance representation.

3.2 Context-aware Self-attention

Self-attentive representations encode a variable-length sequence into a fixed size, using an attention mechanism that considers different positions within the sequence. Inspired by Tran et al. (2017), we use the previous hidden state from the conversation-level RNN (Section 3.3), which provides the context of the conversation so far, and combine it with the hidden states of all the constituent words in an utterance, into a self-attentive encoder (Lin et al., 2017), which computes a $2D$ representation of each input utterance. We follow the notation originally presented in Lin et al. (2017) to explain our modification of their self-attentive sentence representation below.

An utterance u_i, which is a sequence of n words $\{w_i^1, w_i^2, ...w_i^n\}$, is mapped into an embedding layer, resulting in a d-dimensional word embedding for every word. It is then fed into a bidirectional-GRU layer, whose hidden state outputs are concatenated at every time step.

$$\overrightarrow{h_i^j} = \overrightarrow{GRU}(w_i^j, \overrightarrow{h_i^{j-1}}) \qquad (1)$$

$$\overleftarrow{h_i^j} = \overleftarrow{GRU}(w_i^j, \overleftarrow{h_i^{j+1}}) \qquad (2)$$

$$\mathbf{h_i^j} = concat(\overrightarrow{h_i^j}, \overleftarrow{h_i^j}) \qquad (3)$$

$$H_i = \{\mathbf{h_i^1}, \mathbf{h_i^2}, ...\mathbf{h_i^n}\} \qquad (4)$$

H_i represents the n GRU outputs of size $2u$ (u is the number of hidden units in a unidirectional GRU).

The contextual self-attention scores are then computed as follows:

$$S_i = W_{s2}tanh(W_{s1}H_i^T + W_{s3}\overrightarrow{g_{i-1}} + \mathbf{b}) \qquad (5)$$

Here, W_{s1} is a weight matrix with a shape of $d_a \times 2u$, W_{s2} is a matrix of parameters of shape $r \times d_a$, where r and d_a are hyperparameters we can set arbitrarily, and W_{s3} is a parameter matrix of shape $d_a \times k$ for the conversational context, where k is another hyperparameter that is the size of a hidden state in the conversation-level RNN (size of $\overrightarrow{g_{i-1}}$), and $\mathbf{b}$ is a vector representing bias. Equation 5 can then be treated as a 2-layer MLP with bias, with d_a hidden units, W_{s1}, W_{s2} and W_{s3} as weight parameters. The scores S_i are mapped into a probability matrix A_i by means of a softmax function:

$$A_i = softmax(S_i) \qquad (6)$$

which is then used to obtain a 2-d representation M_i of the input utterance, using the GRU hidden states H_i according to the attention weights provided by A_i as follows:

$$M_i = A_iH_i \qquad (7)$$

This 2-d representation is then projected to a 1-d embedding (denoted as $\mathbf{h_i}$), using a fully-connected layer. The conversation-level GRU then operates over this 1-d utterance embedding, and hence, we can represent $\mathbf{g_i}$ as:

$$\overrightarrow{g_i} = \overrightarrow{GRU}(\mathbf{h_i}, \overrightarrow{g_{i-1}}) \qquad (8)$$

$$\overleftarrow{g_i} = \overrightarrow{GRU}(\mathbf{h_i}, \overrightarrow{g_{i+1}}) \qquad (9)$$

$$\mathbf{g_i} = concat(\overrightarrow{g_i}, \overleftarrow{g_i}) \qquad (10)$$

$\mathbf{g_i}$ then provides the conversation-level context used to learn the attention scores and 2-d representation (M_{i+1}) for the next utterance in the conversation ($\mathbf{h_{i+1}}$).

3.3 Conversation-level RNN

The utterance representation $\mathbf{h_i}$ from the previous step is passed on to the conversation-level RNN, which is another bidirectional GRU layer used to encode utterances across a conversation. The hidden states $\overrightarrow{g_i}$ and $\overleftarrow{g_i}$ (Figure 1) are then concatenated to get the final representation $\mathbf{g_i}$ of each utterance, which is further propagated to a linear chain CRF layer. The CRF layer considers the correlations between labels in context and jointly decodes the optimal sequence of utterance labels for a given conversation, instead of decoding each label independently.

4 Data

We evaluate the classification accuracy of our model on the two standard datasets used for DA classification: the Switchboard Dialogue Act Corpus (SwDA) (Jurafsky et al., 1997) consisting of 43 classes, and the 5-class version of the ICSI Meeting Recorder Dialogue Act Corpus (MRDA) introduced in (Ang et al., 2005). For both datasets,

Dataset	|C|	|V|	Train	Validation	Test
MRDA	5	12k	78k	16k	15k
SwDA	43	20k	193k	23k	5k

Table 2: Number of utterances by dataset. |C| denotes number of classes and |V| is the vocabulary size.

we use the train, validation and test splits as defined in Lee and Dernoncourt (2016).

Table 2 shows the statistics for both datasets. They are highly imbalanced in terms of class distribution, with the DA classes `Statement-non-opinion` and `Acknowledge/Backchannel` in SwDA and `Statement` in MRDA making up over 50% of the labels in each set.

5 Results

5.1 Dialogue Act Classification

We compare the classification accuracy of our model against several other recent methods (Table 3).[1] Four approaches (Chen et al., 2018; Tran et al., 2017; Ortega and Vu, 2017; Shen and Lee, 2016) use attention in some form to model the conversations, but none of them have explored self-attention for the task. The last three use CRFs in the final layer of sequence labeling. Only one other method (Chen et al., 2018) uses character-level word embeddings. All models and their variants were trained ten times and we report the average test performance. Our model outperforms state-of-the-art methods by 1.6% on SwDA, the primary dataset for this task, and comes within 0.6% on MRDA. It also beats a TF-IDF GloVe baseline (described in Section 5.2) by 16.4% and 12.2%, respectively.

The improvements that the model is able to make over the other methods are significant, however, the gains on MRDA still fall short of the state-of-the-art by 0.6%. This can mostly be attributed to the conversation/context lengths and label noise at the conversation level. Conversations in MRDA (1493 utterances on average) are significantly longer than in SwDA (271 utterances on average). In spite of having nearly 12% the number of labels (5 vs 43) compared to SwDA, MRDA has 6 times the normalized label entropy in its data. Consequently, due to the noise in label dependencies, and hence, in the inherent conversational structure, the model is not able to yield as big of a gain on the MRDA as it does on the SwDA. Consequently, learning long-range dependencies is a challenge because of noisier and longer path lengths in the network. This is illustrated in Figures 2 and 3, which show for every class, the variation between the entropy of the previous label in a conversation, and the accuracy of that class. MRDA was found to have a high negative correlation[2] (-0.68) between previous label entropy and accuracy, indicating the impact of label noise, which was compounded by longer conversations. On the other hand, SwDA was found to have a low positive correlation (+0.22), which could be compensated by significantly shorter conversations.

Model	SwDA	MRDA
TF-IDF GloVe	66.5	78.7
Kalchbrenner and Blunsom (2013)	73.9	-
Lee and Dernoncourt (2016)	73.9	84.6
Khanpour et al. (2016)	75.8	86.8
Ji et al. (2016)	77.0	-
Shen and Lee (2016)	72.6	-
Li and Wu (2016)	79.4	-
Ortega and Vu (2017)	73.8	84.3
Tran et al. (2017)	74.5	-
Kumar et al. (2018)	79.2	90.9
Chen et al. (2018)	81.3	**91.7**
Our Method	**82.9**	91.1
Human Agreement	84.0	-

Table 3: DA Classification Accuracy

5.2 Utterance Representation Learning

One of the primary motivations for this work was to investigate whether one can improve performance by learning better representations for utterances. To address this, we retrained our model by replacing the utterance representation learning (utterance-level RNN + context-aware self-attention) component with various sentence representation learning methods (either pre-training them or learning jointly), and feeding them into the conversation-level recurrent layers in the hierarchical model, so that the performance is indicative of the quality of utterance representations.

There are three main categories of utterance representation learning approaches: (i) the baseline which uses a TF-IDF weighted sum of GloVe word embeddings; (ii) pre-trained on cor-

---

[1] Contemporaneous to this submission, (Li et al., 2018; Wan et al., 2018; Ravi and Kozareva, 2018) proposed different approaches for the task. We do not focus on them here per NAACL 2019 guidelines, however note that our system outperforms the first two. (Ravi and Kozareva, 2018) bypasses the need for complex networks with huge parameters but its overall accuracy is 4.2% behind our system, despite being 0.2% higher on SwDA.

[2] Pearson's r

Method	SwDA	MRDA
Baseline		
TF-IDF GloVe	66.5	78.7
Pre-trained on Corpus		
Skip Thought Vectors	72.6	82.8
Paragraph vectors	72.5	82.6
Joint Learning		
RNN-Encoder	74.8	85.7
Bi-RNN-LastState	76.2	85.4
Bi-RNN-MaxPool	77.6	86.7
CNN	76.9	84.5
Bi-RNN + Attention	80.1	87.7
+ Context	81.8	89.2
Bi-RNN + Self-attention	81.1	88.6
+ Context	82.9	91.1

Table 4: Performance of utterance representation methods when integrated with the hierarchical model

pus, where we first learn utterance representations on the corpus using Skip-Thought Vectors (Kiros et al., 2015) and Paragraph Vectors (Le and Mikolov, 2014), and then use them with the rest of the model; (iii) jointly learned with the DA classification task. Table 4 describes the performance of different utterance representation learning methods when combined with the overall architecture on both datasets.

Introducing the word-level attention mechanism (Yang et al., 2016) enables the model to learn better representations by attending to more informative words in an utterance, resulting in better performance (Bi-RNN + Attention). The self-attention mechanism (Bi-RNN + Self-attention) leads to even greater overall improvements. Adding context information (previous recurrent state of the conversation) boosts performance significantly.

A notable aspect of our model is how contextual information is leveraged at different levels of the sequence modeling task. The combination of conversation-level contextual states for utterance-representation learning (+ Context) and a CRF at the conversation level to further inform conversation sequence modeling, leads to a collective performance improvement. This is particularly pronounced on the SwDA dataset: the two variants of the context-aware attention models (Bi-RNN + Attention + Context and Bi-RNN + Self-attention + Context) have significant performance gains.

6 Conclusion

We developed a model for DA classification with context-aware self-attention, which significantly outperforms earlier models on the commonly-used SwDA dataset and is very close to state-of-the-art on MRDA. We experimented with different utterance representation learning methods and showed that utterance representations learned at the lower levels can impact the classification performance at the higher level. Employing self-attention, which has not previously been applied to this task, enables the model to learn richer, more effective utterance representations for the task.

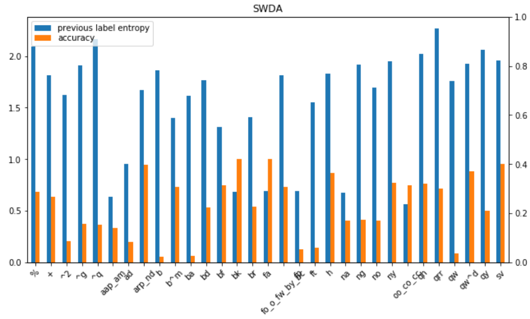


Figure 2: Previous Label Entropy vs. Accuracy on the SwDA Dataset

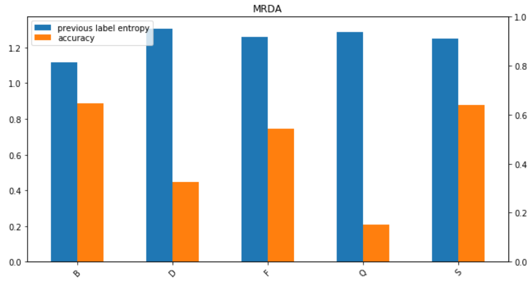


Figure 3: Previous Label Entropy vs. Accuracy on the MRDA Dataset

As future work, we would like to experiment with other attention mechanisms such as multi-head attention (Vaswani et al., 2017), directional self-attention (Shen et al., 2018a), block self-attention (Shen et al., 2018b), or hierarchical attention (Yang et al., 2016), since they have been shown to address the limitations of vanilla attention and self-attention by either incorporating information from different representation subspaces at different positions to capture both local and long-range context dependencies, encoding temporal order information, or by attending to context dependencies at different levels of granularity.

Acknowledgements

The authors would like to thank Dimitris Alikaniotis, Maria Nadejde and Courtney Napoles for their insightful discussions, and the anonymous reviewers for their helpful comments.

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '05*, pages 1061–1064.

John Langshaw Austin. 1975. *How to do things with words*. Oxford university press.

Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. 2018. Dialogue act recognition via crf-attentive structured network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 225–234. ACM.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 332–342. Association for Computational Linguistics.

Dan Jurafsky, Liz Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado at Boulder Technical Report 97-02.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126. Association for Computational Linguistics.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021. The COLING 2016 Organizing Committee.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, and Sachindra Joshi. 2018. Dialogue act sequence labeling using hierarchical encoder with CRF. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 3440–3447.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, pages 1188–1196.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520. Association for Computational Linguistics.

Ruizhe Li, Chenghua Lin, Matthew Collinson, Xiao Li, and Guanyi Chen. 2018. A dual-attention hierarchical recurrent neural network for dialogue act classification. *CoRR*, abs/1810.09154.

Wei Li and Yunfang Wu. 2016. Multi-level gated recurrent neural network for dialog act classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1970–1979. The COLING 2016 Organizing Committee.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations 2017 (Conference Track)*.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using context information for dialog act classification in dnn framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Daniel Ortega and Ngoc Thang Vu. 2017. Neural-based context representation learning for dialog act classification. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 247–252. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long Papers), pages 2227–2237. Association for Computational Linguistics.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 887–893. Association for Computational Linguistics.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Sheng-syun Shen and Hung-yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. In *INTERSPEECH 2016, 17th Annual Conference of the International Speech Communication Association*, pages 2716–2720.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5446–5455.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *International Conference on Learning Representations*.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3).

Dinoj Surendran and Gina-Anne Levow. 2006. Dialog act tagging with support vector machines and hidden markov models. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing*.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2017. A hierarchical neural model for learning sequences of dialogue acts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 428–437. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yao Wan, Wenqiang Yan, Jianwei Gao, Zhou Zhao, Jian Wu, and Philip S. Yu. 2018. Improved dynamic memory network for dialogue act classification with adversarial training. *CoRR*, abs/1811.05021.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics.

Matthias Zimmermann. 2009. Joint segmentation and classification of dialog acts using conditional random fields. In *INTERSPEECH 2009, 10th Annual Conference of the International Speech Communication Association*, pages 864–867.

A Supplementary Material

A.1 Training & Hyperparameters

All hyperparameters were selected by tuning one hyperparameter at a time while keeping the others fixed. Validation splits were used for the tuning process. The final set of hyperparameters were then used to train two different models, one each on SwDA and MRDA training splits. Table 5 lists the range of values for each parameter that we experimented with, and the final value that was chosen. Dropout was applied to the utterance embeddings. Early stopping was used on the validation set with a patience of 15 epochs.

Hyperparams	**Range of values**	**Final value**
Word Embeddings	GloVe 100D GloVe 200D GloVe 300D	GloVe 300D + ELMo 1024D
	Word2vec 300D Word2vec 200D	
	ELMo 1024D	
	GloVe 300D + ELMo 1024D	
	Word2Vec 300D + ELMo 1024D	
Sentence GRU Size (u)	20 - 300	50
Utterance GRU Size (k)	20 - 600	100
Learning Rate	0.01 - 2.0	0.015
Dropout	0.1 - 0.8	0.3
Optimizer	SGD, RMSProp, Adam	Adam

Table 5: Hyperparameter space and tuned values

Affect-Driven Dialog Generation

Pierre Colombo[1*], **Wojciech Witon**[1*], **Ashutosh Modi**[1], **James Kennedy**[1], **Mubbasir Kapadia**[1,2]
[1]Disney Research, [2]Rutgers University
{pierre.colombo, wojtek.witon}@disneyresearch.com
ashutosh.modi@disneyresearch.com
james.kennedy@disneyresearch.com
mubbasir.kapadia@rutgers.edu

Abstract

The majority of current systems for end-to-end dialog generation focus on response quality without an explicit control over the affective content of the responses. In this paper, we present an affect-driven dialog system, which generates emotional responses in a controlled manner using a continuous representation of emotions. The system achieves this by modeling emotions at a word and sequence level using: (1) a vector representation of the desired emotion, (2) an affect regularizer, which penalizes neutral words, and (3) an affect sampling method, which forces the neural network to generate diverse words that are emotionally relevant. During inference, we use a reranking procedure that aims to extract the most emotionally relevant responses using a human-in-the-loop optimization process. We study the performance of our system in terms of both quantitative (BLEU score and response diversity), and qualitative (emotional appropriateness) measures.

1 Introduction

Recent breakthroughs in deep learning techniques have had an impact on end-to-end conversational systems (Chen et al., 2017). Current research is mainly focused on functional aspects of conversational systems: keyword extraction, natural language understanding, and pertinence of generated responses (Ilievski et al., 2018). Although these aspects are indeed key features for building a commercial system, most existing solutions lack social intelligence. Conversational systems could benefit from incorporating social intelligence by: (1) avoiding interaction problems that may arise when the system does not understand the user's request (e.g., inappropriate responses that cause user anger) (Maslowski et al., 2017), and (2) building rapport with the user (Strohkorb et al., 2016). Our method makes such conversational systems more social by outputting responses expressing emotion in a controlled manner, without sacrificing grammatical correctness, coherence, or relevance.

Existing sequence-to-sequence (seq2seq) architectures, either recurrent- (Sordoni et al., 2015; Serban et al., 2015), attention- (Vaswani et al., 2017) or convolutional neural network (CNN)-based (Fan et al., 2018), do not provide a straightforward way to generate emotionally relevant output in a controlled manner. We introduce EMOTIonal CONversational System (EMOTICONS), which generates emotion-specific responses. It is based on novel contributions presented in this paper which fall in two main categories: explicit models which allow a controlled emotion-based response generation (e.g., methods based on emotion embeddings, affective sampling, and affective re-ranking), and implicit models with no direct control over the desired emotion (i.e., affective regularizer). We show that EMOTICONS outperforms both the system proposed by Zhou et al. (2018) (current state of the art for our task) and the vanilla seq2seq in terms of BLEU score (Papineni et al., 2002) (improvement up to 7.7%) and response diversity (improvement up to 52%). Additionally, we qualitatively evaluate the emotional content of the generated text (see example responses in Table 1). The user study (22 people) demonstrates that EMOTICONS is able to generate grammatically correct, coherent, emotionally-rich text in a controlled manner.

2 Related Work

Sequence-to-sequence (seq2seq) models have attracted a lot of attention in the past few years, especially in the fields of Neural Machine Translation (Sutskever et al., 2014; Bahdanau et al., 2014) and Neural Dialogue Generation (Sordoni et al., 2015; Vinyals and Le, 2015; Serban et al.,

---

[*] Both authors contributed equally to this work.

Proceedings of NAACL-HLT 2019, pages 3734–3743
Minneapolis, Minnesota, June 2 - June 7, 2019.

	Input	Good to see you again.	Be careful, I have a knife.	You're the only one who can help us.
	Baseline	It's good to see you.	Don't worry about it.	There's a lot of people here.
EMOTICONS	Fear	What are you doing here?	I'm just trying to get out of here.	What are you going to do with me?
	Anger	I'm not here.	I'm going to kill you.	You're not the only one who can help me.
	Joy	Good to see you.	I'm going to marry you.	I can protect you.
	Surprise	You want to talk to me?	I'm just going to the party.	You've got to be kidding me.

Table 1: Example responses from the baseline (seq2seq) model and the four EMOTICONS models with different emotions.

2015). Prior work has focused on designing architectures that lead to the best performance in terms of BLEU (Papineni et al., 2002) and Perplexity scores. Most seq2seq models are based on gated recurrent neural networks, either Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) (Serban et al., 2015), but in general it is difficult to conclude which gating mechanism performs better (Chung et al., 2014). In our model, we use GRU because it has fewer parameters to optimize, and it is faster to train.

In order to overcome the problem of generating trivial or mundane responses, there have been developments in inference techniques for encoder-decoder systems. Use of beam search has been shown to improve the general quality of generated answers, while Maximum Mutual Information (MMI) (Li et al., 2016) has improved the diversity of generated answers, leading to more meaningful output. We build on these techniques during affective inference.

Emotion-based (affective) dialog generation systems have received increasing attention in the past few years. Huang et al. (2018) use emotion tokens (special "words" in a dictionary representing specific emotions) at either the encoder or decoder side, forcing the decoder to output a sentence with one specific emotion. Zhou et al. (2018) build their system using external and internal memory, where the former forces the network to generate emotional words, and the latter measures how emotional a generated sequence is compared to a target sequence. Lubis et al. (2018) modeled emotions in Valence-Arousal (VA) space for response generation. We extend this idea by using a Valence-Arousal-Dominance (VAD) Lexicon (Mohammad, 2018), as it has been shown by Broekens (2012) that the third dimension (Dominance) is useful for modeling affect. Asghar et al. (2017) used the VAD Lexicon, but they let the neural network choose the emotion to generate (by maximizing or minimizing the affective dissonance) and their system cannot generate different emotional outputs for the same input, nor generate a specified emotion.

3 System Architecture

Our system (see overview in Figure 1) is divided into three main components: (1) Emotion Labeling – automatic labeling of sentences according to the emotional content they express, using an emotion classifier (§3.2.1); labeling of words with VAD Lexicon values (§4.2), (2) Affective Training – training of two seq2seq networks, which use an encoder-decoder setting. The first network is trained with prompt-response pairs (S-T), whereas the second (used during Affective Inference) is trained with reversed pairs (T-S), (3) Affective Inference – generation of many plausible responses, which are re-ranked based on emotional content.

3.1 Preliminaries

Let $V = \{w_1, w_2, \ldots, w_{|V|}\}$ be a vocabulary, and $\mathcal{X} = (x_1, x_2, \ldots, x_{|\mathcal{X}|})$ a sequence of words (e.g. a sentence). We denote $E_{\mathcal{X}} \in \mathbb{R}^6$ as an emotion vector representing a probability distribution over six emotions associated with the sequence $\mathcal{X}$:

$$E_{\mathcal{X}} = \begin{bmatrix} p_{\text{anger}} \\ p_{\text{surprise}} \\ p_{\text{joy}} \\ p_{\text{sadness}} \\ p_{\text{fear}} \\ p_{\text{disgust}} \end{bmatrix}$$

Note that in this work we focus on six basic emotions proposed by Paul Ekman (Ekman et al., 1983) but the techniques we develop are general and can be extended to a more fine grained list of emotions. $\mathcal{X}$ can be an input sequence, candidate response, final response, or target response (denoted respectively as $\mathcal{S}$, $\mathcal{R}_C$, $\mathcal{R}_{\text{final}}$, $\mathcal{R}_0$). We introduce E_0, which during training, is the representation of the emotion of the target response ($\mathcal{R}_0$). During testing, E_0 indicates a desired emotion for the final response ($\mathcal{R}_{\text{final}}$), and can be set manually. For

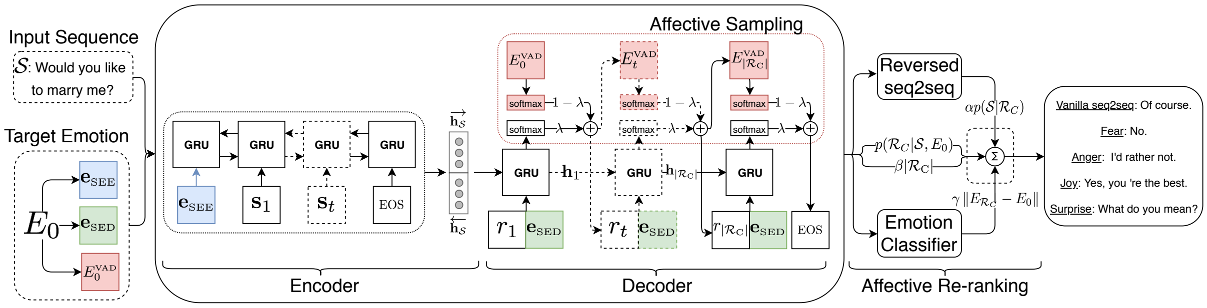


Figure 1: System overview: The input sequence and the target emotion (automatically extracted from the target response using the emotion classifier during training, and set by the user during inference) are fed into the seq2seq. The generated candidate responses are re-ranked based on the output of the reversed seq2seq, the length, and the emotional content.

example, in the case of ‘anger’, E_0 would be a one-hot vector with 1 at the first position, and 0 elsewhere.

In our work, we extend the standard seq2seq model (Sutskever et al., 2014), that predicts the final response $\mathcal{R}_{\text{final}} = \text{argmax}_{\mathcal{R}_C}\, p(\mathcal{R}_C|S)$. The proposed affective system aims to extend the inference mechanism by incorporating emotions encoded in E_0:

$$\mathcal{R}_{\text{final}} = \underset{\mathcal{R}_C}{\text{argmax}}\, p(\mathcal{R}_C|S, E_0) \tag{1}$$

3.2 Affect Modeling

We extend the standard seq2seq architecture by including emotion-specific information during the training and the inference. A critical challenge in both generating and evaluating responses is a reliable assessment of emotional state. We use two representations of emotion: (1) a categorical representation with six emotions (anger, surprise, joy, sadness, fear, disgust), and (2) a continuous representation in a VAD space. The latter uses a VAD Lexicon introduced by Mohammad (2018), where each of 20k words is mapped to a 3D vector of VAD values, ranging from 0 (lowest) to 1 (highest) ($\mathbf{v} \in [0,1]^3$). Valence measures the positivity/negativity, Arousal the excitement/calmness, and Dominance the powerfulness/weakness of the emotion expressed by a word. This expands the work of Lubis et al. (2018), who modeled emotions only in VA space. In the following sections we describe different versions of the proposed model.

3.2.1 Emotion Classifier

Affective training requires E_0, the emotion representation of the target sequence. In order to label all sentences of the corpus with E_0, we use an Emotion Classifier by Witon et al. (2018). The classifier predicts a probability distribution over class of six emotions. The classifier predictions for Cornell Movie-Dialogs Corpus (Cornell) have been shown to be highly correlated with human predictions (Witon et al., 2018).

3.2.2 Sequence-Level Explicit Encoder Model (SEE)

To explicitly generate responses with emotion, this version of the model includes an emotion embedding at the encoder side. We feed the encoder with $\mathcal{S}' = (\mathbf{e}_{\text{SEE}}, s_1, s_2, \ldots, s_{|S|})$, where $\mathbf{e}_{\text{SEE}} = \mathbf{A}_{SEE}E_0$ is an Emotion Embedding ($\mathbf{e}_{\text{SEE}} \in \mathbb{R}^3$), and $\mathbf{A}_{SEE} \in \mathbb{R}^{3\times 6}$ is a mapping (learned during training) from E_0 into an emotion embedding space.

3.2.3 Sequence-Level Explicit Decoder Model (SED)

Another way of forcing an emotional output is to explicitly indicate the target emotion at every step in decoding along with other inputs. Formally, the GRU hidden state at time t is calculated as $\mathbf{h}_t = f(\mathbf{h}_{t-1}, r'_t)$ with $r'_t = [r_{t-1}; \mathbf{e}_{\text{SED}}]$, where $\mathbf{e}_{\text{SED}}$ is defined similarly as $\mathbf{e}_{\text{SEE}}$. It is worth noting that $\mathbf{A}_{SEE}$ and $\mathbf{A}_{SED}$ are different, which implies that the emotion embedding spaces they map to are also different. Compared to a similar approach introduced by Huang et al. (2018), our solution enables the desired emotional content, E_0, to be provided in a continuous space.

3.2.4 Word-Level Implicit Model (WI)

To model the word-level emotion carried by each sequence, we introduce an Affective Regularizer (AR), which expresses the affective distance between $\mathcal{R}_{\text{final}}$ and $\mathcal{R}_0$, in the VAD space. It forces the neural network to prefer words in the vocabulary that carry emotions in terms of VAD. Mathematically, we extend the regular Negative Log Likelihood (NLL) loss with an affective regularizer, $\mathcal{L}_{\text{AR}}$:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{AR}} \\ &= -\log p(\mathcal{R}_{\text{final}}|\mathcal{S}) + \mu \mathcal{L}_{\text{VAD}}(\mathcal{R}_{\text{final}}, \mathcal{R}_0)\end{aligned}$$

$$\mathcal{L}_{\text{VAD}}(\mathcal{R}_{\text{final}}, \mathcal{R}_0) = \left\| \sum_{t=1}^{|\mathcal{R}_{\text{final}}|} \frac{\mathbf{E}^{\text{VAD}}\mathbf{s}_t}{|\mathcal{R}_{\text{final}}|} - \sum_{t=1}^{|\mathcal{R}_0|} \frac{\mathbf{e}^{\text{VAD}}_{r_{0_t}}}{|\mathcal{R}_0|} \right\|,$$

where $\mathbf{s}_t = \text{softmax}(\mathbf{h}_t)$ ($\mathbf{s}_t \in \mathbb{R}^{|V|}$) is a confidence of the system of generating words $w_1, \ldots, w_{|V|}$ at time t and $\mu \in \mathbb{R}$. $\mathbf{e}^{\text{VAD}}_x \in \mathbb{R}^3$ is a 3D vector representing emotion associated with a word x in VAD space (note that $\mathbf{e}^{\text{VAD}}_x$ is constant with respect to t), and $\mathbf{E}^{\text{VAD}} \in \mathbb{R}^{3\times|V|}$ is a matrix containing $\mathbf{e}^{\text{VAD}}_{w_v}$ for all $|V|$ words in the vocabulary:

$$\mathbf{E}^{\text{VAD}} = \left[\, \mathbf{e}^{\text{VAD}}_{w_1} ; \; \ldots \; ; \mathbf{e}^{\text{VAD}}_{w_{|V|}} \right]$$

Intuitively, the regularizer penalizes the deviation of the emotional content of the generated response, $\mathcal{R}_{\text{final}}$, from the desired response, $\mathcal{R}_0$. The emotional information carried by $\mathcal{R}_{\text{final}}$ is the weighted sum of emotion representations $\mathbf{e}^{\text{VAD}}_{w_i}$ for all words w_i in the vocabulary, where the weights are determined by the confidence $\mathbf{s}_t$.

3.2.5 Word-Level Explicit Model (WE)

Sequential word generation allows sampling of the next word, based on the emotional content of the current incomplete sequence. If some words in a sequence do not express the target emotion E_0, other words can compensate for this by changing the final affective content, e.g., in a sentence "I think that the cat really loves me!", the first 6 words are neutral, whereas the end of the sentence make it clearly express joy. We incorporate this observation by explicitly generating the next word using an Adaptive Affective Sampling Method:

$$\log p(\mathcal{R}_C|\mathcal{S}, E_0) = \sum_{t=1}^{|\mathcal{R}_C|} \log p(r_t|r_{<t}, \mathbf{e}_{r_{<t}}, \mathbf{h}_{\mathcal{S}}, E_0),$$

$$\begin{aligned}p(r_t|r_{<t}, \mathbf{e}_{r_{<t}}, \mathbf{h}_{\mathcal{S}}, E_0) = {}& \lambda \,\text{softmax}\, g(\mathbf{h}_t) + \\ & (1-\lambda)\,\text{softmax}\, v(E^{\text{VAD}}_t),\end{aligned}$$

where $g(\mathbf{h}_t)$ is a linear mapping from GRU hidden state $\mathbf{h}_t$ to an output vector of size $|V|$, and $0 \leq \lambda \leq 1$ is learned during training. The first term in Equation 3.2.5 is responsible for generating words according to a language model preserving grammatical correctness of the sequence, whereas the second term forces generation of words carrying emotionally relevant content. $E^{\text{VAD}}_t \in \mathbb{R}^3$ is a vector representing the remaining emotional content needed to match a goal (E^{VAD}_0) after generating all words up to time t. It is updated every time a new word r_t with an associated emotion vector $\mathbf{e}^{\text{VAD}}_{r_t}$ is generated:

$$E^{\text{VAD}}_t = E^{\text{VAD}}_{t-1} - \mathbf{e}^{\text{VAD}}_{r_{t-1}}$$

$$E^{\text{VAD}}_0 = \begin{cases} \sum_{t=1}^{|\mathcal{R}_0|} \mathbf{e}^{\text{VAD}}_{r_{0t}}, & \text{training} \\ \mathbf{M}_{\text{VAD}} E_0 \cdot \text{max}_{\text{length}}, & \text{inference} \end{cases}$$

where $\mathbf{e}^{\text{VAD}}_{r_{0t}}$ is an emotion vector associated with words r_{0_t} in the target sequence, $\text{max}_{\text{length}}$ is a maximum length set for the seq2seq model, and $\mathbf{M}_{\text{VAD}} \in \mathbb{R}^{3\times6}$ is a mapping from six-dimensional emotion space into VAD space (every emotion has a VAD vector as introduced by Hoffmann et al. (2012), scaled to a range [0, 1]):

$$\mathbf{M}_{\text{VAD}} = \begin{array}{cccccc|c} \text{anger} & \text{surprise} & \text{joy} & \text{sadness} & \text{fear} & \text{disgust} & \\ 0 & 1 & 1 & 0 & 0 & 0 & V \\ 1 & 1 & 1 & 0 & 1 & 0.5 & A \\ 1 & 0 & 1 & 0 & 0 & 0.5 & D \end{array}$$

$v(E^{\text{VAD}}_t)$ is a vector, whose i-th component measures the potential remaining emotional content of the sequence in the case of choosing the i-th word w_i:

$$v(E^{\text{VAD}}_t) = -\begin{bmatrix} \left\|E^{\text{VAD}}_t - \mathbf{e}_{w_0}\right\| \\ \cdots \\ \left\|E^{\text{VAD}}_t - \mathbf{e}_{w_{|V|}}\right\| \end{bmatrix}$$

In the following, we set a constant $\lambda = 1$ after generating the first $\text{max}_{\text{length}}/2$ words, as this setting ensures that the first generated words carry the right emotional content, while not sacrificing the grammatical correctness of the whole response. This leads to an improvement in performance.

3.3 Affective Inference

The methods described in the previous sections aim to improve the seq2seq training/sampling procedure. We hypothesize that a good inference strategy is crucial for generating diverse and emotion-specific responses. As Li et al. (2016) suggest, traditional objective functions, i.e., likelihood of a response given an input, can be improved by using an N-best list and MMI during inference. We build upon this idea; our hypothesis is that by generating B diverse sequences and re-ranking the responses, we are more likely to infer one best emotion-specific response. The B-best list is found using Beam Search of size B with length normalization.

In the MMI-bidi setting, Li et al. (2016) rank all responses found during beam search based on a score calculated as:

$$\mathcal{R}_{\text{final}} = \underset{\mathcal{R}_C}{\operatorname{argmax}}\, p(\mathcal{R}_C|\mathcal{S}) + \alpha p(\mathcal{S}|\mathcal{R}_C) + \beta|\mathcal{R}_C|, \tag{2}$$

where $p(\mathcal{S}|\mathcal{R}_C)$ is a model with the same architecture as $p(\mathcal{R}_C|\mathcal{S})$ trained on reversed prompt-response pairs (T-S), and $|R_C|$ is the length of the candidate response, R_C. We modify this objective in the following form:

$$\begin{aligned}\mathcal{R}_{\text{final}} = \underset{\mathcal{R}_C}{\operatorname{argmax}}\, & p(\mathcal{R}_C|\mathcal{S}, E_0) + \alpha p(\mathcal{S}|\mathcal{R}_C) \\ & + \beta|\mathcal{R}_C| - \gamma \left\| E_{\mathcal{R}_C} - E_0 \right\|, \end{aligned} \tag{3}$$

where the last term penalizes the deviation of the emotional content, $E_{\mathcal{R}_C}$, of the candidate response, $\mathcal{R}_C$, from the desired emotional content, E_0. The task is to find optimal values of parameters α, β and γ, which give the best responses in terms of grammatical correctness, diversity (α, β) and emotional content (γ) (see §5 and §6).

4 Model Training

In this section, we describe corpora used for training, the baseline models and the training procedure for the models presented in §3.

4.1 Corpora

Cornell contains around 10K movie characters and around 220K dialogues (Danescu-Niculescu-Mizil and Lee, 2011).

OpenSubtitles2018 is a collection of translated movie subtitles with 3.35G sentence fragments (Tiedemann, 2009). It has been filtered to get pairs of consecutive sequences (containing between 5 and 30 words), with respective timestamps within an interval of 5 seconds, that are part of a conversation of at least 4 turns. The filtered dataset contains 2.5M utterances.

Preprocessing Each dataset is tokenized using the spaCy[1] tokenizer, converted to lowercase, and non-ASCII symbols are removed. To restrain the vocabulary size and correct the typos, we use a default vocabulary of fixed size 42K words from spaCy. Each word in the dataset is then compared with the vocabulary using the difflib library[2] in Python (algorithm based on the Levenshtein distance), and mapped to the most similar word in the vocabulary. If no word with more than 90% of similarity is found, the word is considered a rare word or a typo, and is mapped to the out-of-vocabulary (OOV) word. For Cornell, less than 1% of the unigrams are OOV.

4.2 Affective Dictionary

The VAD lexicon may not have all the words in the vocabulary. Based on the word similarity (using difflib library), each word of the vocabulary is assigned a VAD value of the most similar word in the VAD lexicon. If no word with more than 90% of similarity is found, a "neutral" VAD value ($\mathbf{v} = [0.5, 0.5, 0.5]$) is assigned.

4.3 Baselines

We compare our work to two different baselines: a vanilla seq2seq and the ECM introduced by Zhou et al. (2018). For the external memory we use our affective dictionary and train the model using the default parameters provided by authors.

4.4 Training Details

All the hyper-parameters have been optimized on the validation set using BLEU score (Papineni et al., 2002). For the encoder, we use two-layer bidirectional GRUs (hidden size of 256). The final hidden states from both directions are concatenated and fed as an input to the decoder of one-layer unidirectional GRUs (hidden size of 512). The embedding layer is initialized with pre-trained word vectors of size 300 (Mikolov et al., 2018), trained with subword information (on Wikipedia 2017, UMBC web-base corpus and statmt.org news dataset), and updated during training. We use ADAM optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 for learning $p(\mathcal{R}_C|\mathcal{S}, E_0)$ (resp. 0.01 for $p(\mathcal{S}|\mathcal{R}_C)$), which is updated by using a scheduler with a patience of 20 epochs and a decreasing rate of 0.5. The gradient norm is clipped to 5.0, weight decay is set to $1e^{-5}$, and dropout (Srivastava et al., 2014) is set to 0.2. The maximum sequence length is set to 20 for Cornell and to 30 for OpenSubtitles. The models have been trained on 94%, validated on 1%, and tested on 5% of the data.

---

[1] https://spacy.io

[2] https://docs.python.org/3/library/difflib.html

	Model	C distinct-1	C distinct-2	OS distinct-1	OS distinct-2	C BLEU	OS BLEU
No re-rank	Baseline	0.0305	0.1402	0.0175	0.1205	0.0096	0.094
	ECM	0.0310	0.1412	0.0180	0.1263	0.0099	0.099
	SEE	0.0272	0.1331	0.0170	0.1100	0.0110	0.093
	SED	0.0303	0.1502	0.0189	0.1231	0.0128	0.103
	WI	0.0316	0.1480	0.0175	0.1235	0.0129	0.100
	WE	0.0310	0.1400	0.0195	0.1302	0.0098	0.095
	WI + WE	**0.0342**	**0.1530**	**0.0198**	**0.1300**	**0.0108**	**0.105**
		(+12.1%)	**(+9.1%)**	**(+13.1%)**	**(+7.9%)**	**(+12.5%)**	**(+11.7%)**
Re-rank	MMI$_{baseline}$	0.0379	0.1473	0.0200	0.1403	0.0130	0.105
	EMOTICONS$_{\gamma=0}$	**0.0406**	**0.2030**	**0.0305**	**0.1431**	**0.0140**	**0.110**
		(+7.1%)	**(+37.8%)**	**(+52.5%)**	**(+2.0%)**	**(+7.7%)**	**(+4.8%)**

Table 2: Quantitative results: Results for all proposed models trained on Cornell (C) and OpenSubtitles (OS). distinct-1 and distinct-2 count the number of distinct unigrams and bigrams, respectively, normalized by the total number of generated tokens in 200 candidate responses. The performance boost is computed with respect to the vanilla seq2seq model.

5 Quantitative Evaluation for Model Selection

To evaluate language models, we use BLEU score (computed using 1- to 4-grams), as it has been shown to correlate well with human judgment (Agarwal and Lavie, 2008). Perplexity does not provide a fair comparison across the models: during the training of the baseline seq2seq model, we minimize the cross entropy loss (logarithm of perplexity), whereas in other models (e.g., WI) we aim to minimize a different loss not directly related to perplexity (cross entropy extended with the affective regularizer). Having more diverse responses makes the affective re-ranking more efficient, to evaluate diversity we count the number of distinct unigrams (distinct-1) and bigrams (distinct-2), normalized by the total number of generated tokens.

The performance of different models introduced in §3 are presented in Table 2. MMI$_{\text{bas.}}$ refers to a system that re-ranks responses based on Equation 2, where both $p(\mathcal{R}_C|\mathcal{S})$ and $p(\mathcal{S}|\mathcal{R}_C)$ are baseline seq2seq models. EMOTICONS is a system based on Equation 3, where $p(\mathcal{R}_C|\mathcal{S}, E_0)$ is computed using a composition of Word-Level Implicit Model (WI) and Word-Level Explicit Model (WE), and $p(\mathcal{S}|\mathcal{R}_C)$ is computed using WI (as we are not interested in explicitly using the input emotion). We optimize α and β on the validation set using BLEU score, since Li et al. (2016) have shown that adding MMI during inference improves the BLEU score. We set $\gamma = 0$ and find optimal values $\alpha_{opt} = 50.0$ and $\beta_{opt} = 0.001$ using grid search.

Improving BLEU score and diversity was not the goal of our work, but the observed improvement (after adding emotions) shows that the different systems are able to extract and use emotional patterns to improve the general language model.

5.1 Response Diversity

From Table 2, we observe that for both Cornell and OpenSubtitles datasets, SED, WI, and WE models outperform the vanilla seq2seq and the ECM for at least one of the two distinct measures. SEE has the worst performance overall and does not compete with either the baseline, nor with SED. This is expected according to the results reported by Huang et al. (2018). It seems that the model is not able to capture the information carried by the additional emotion embedding token – it is treated as just one additional word among 20 others. SED makes better use of the emotion information, as it is used at each time step during decoding. In addition, it is more natural to use these features during the decoding, since the emotion embedding represents the desired emotion of the response. The combination of WI and WE performs best in terms of distinct-1 and distinct-2 measures among all models without re-ranking, yielding an improvement of up to 13.1%. It suggests that the word level emotion models suit the seq2seq architecture better. During training, both models are encouraged not only to match the target words, but also to promote less frequent words that are close to the target words in terms of VAD values (affective regularizer and affective sampling), fostering the model to generate more diverse responses.

As expected, by adding MMI, we observe an improvement in diversity, but the relative improvement for OpenSubtitles ($\text{MMI}_{\text{bas.}}$) is smaller than the one reported by Li et al. (2016). This could originate from the different data filtering and beam search strategy, and the fact the hyper-parameter optimization has been performed on Cornell. EMOTICONS is a combination of WI + WE (best performing model) for $p(\mathcal{R}_C|\mathcal{S}, E_0)$ and WI for $p(\mathcal{S}|\mathcal{R}_C)$, it is better than $\text{MMI}_{\text{bas.}}$ (up to 52.5% gain in distinct-1).

It is worth noting that we observe higher scores in terms of diversity for the reversed model $p(\mathcal{S}|\mathcal{R}_C)$ compared to the normal model $p(\mathcal{R}_C|\mathcal{S}, E_0)$, while training on Cornell. We can explain this using the data distribution: distinct-2 is higher for the questions than for the answers (0.167 and 0.154 for Cornell, respectively).

5.2 Response Quality

Table 2 shows that, in general, introducing emotional features into the process of generating responses does not reduce the BLEU score. To reduce the potential negative impact of choosing inappropriate first words in the sequence, we compute the BLEU score on the result of beam search of size 200. For example, if the first word is "I", the seq2seq models tend to generate a response "I don't know" with high probability, due to the high number of appearances of such terms in the training set. In certain cases, like WI and SED, we observe an improvement. Such an improvement is expected, since our model takes into account additional (affective) information from the target sequence during response generation.

6 Human-in-the-Loop Hyper-Parameter Estimation

The quantitative evaluation shows that EMOTICONS outperforms the baseline while adding the emotional features during response generation. The re-ranking phase did not take into account the affective term ($\gamma = 0$ in Equation 3). Setting a different value would not necessarily improve any of the available metrics (e.g., BLEU score, diversity), as they do not explicitly take into account affective content in their definition. In this section, we describe an optimization procedure, relying on human judgment, for finding the optimal value of γ.

6.1 Experiment Description

We asked annotators to evaluate (using an AffectButton) the generated responses. We use AffectButton (Broekens and Brinkman, 2013), a reliable affective tool for assigning emotions, which, to our knowledge, has never been used for estimating the emotional content of the generated responses. In our experiment, the AffectButton lets users choose a facial expression from a continuous space (see Figure 3), that best matches the emotional state associated with the sequence, which is then mapped into the VAD space. In order to conduct the experiment, we chose a pool of 12 annotators, who annotated a total of 400 sequences. The prompts were randomly chosen from the test set of Cornell, among the 200 sequences that create the most diverse responses in terms of distinct-2. The more diverse the responses are, the more likely we are to select a response carrying a desired emotion. The responses for the prompts were generated using EMOTICONS where the target emotion was either fear, anger, joy, or surprise; the four corners of the AffectButton. γ was randomly chosen among 20 uniformly sampled values in $[0, 10]$.

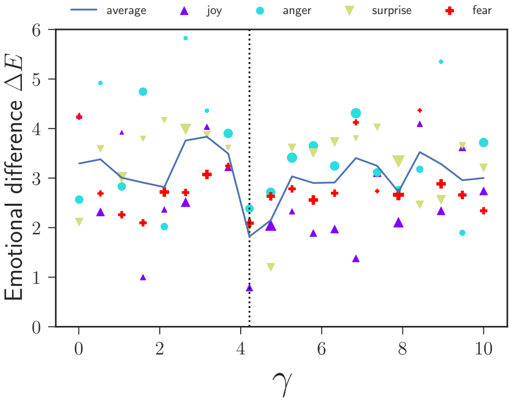


Figure 2: Hyper-parameter optimization: For different values of γ, users assign a face to the generated response. Each point represents an average ΔE of annotations for each emotion. ΔE is the difference between the VAD representation of the face assigned by the user and the desired emotion for the response. The size is proportional to the number of collected annotations. ΔE is at a minimum for $\gamma_{\text{opt}} = 4.2$.

6.2 Experiment Results

In Figure 2, we present the difference between the VAD value according to the face assigned by the user, and the desired emotion for the response. The average curve presents a global minimum at $\gamma_{opt} = 4.2$. The system does not perform equally well at generating different emotions according to the human judgment. On average, we observe lower values for joy compared to anger in Figure 2. This phenomenon is expected, as in the re-ranking

	Model	Grammatical Correctness	User Preference Total	User Preference Majority Vote
	$\text{MMI}_{\text{bas.}}$	83 %	39	8
EMOTICONS	Fear	82 %	96	37
	Anger	80 %		
	Joy	84 %		
	Surprise	79 %		

Table 3: User study results: *Grammatical Correctness* shows the ratio of grammatically correct sentences among all generated responses, whereas *User Preference* shows the number of times each model was preferred by the users.

process $E_{\mathcal{R}_C}$ is estimated using the emotion classifier (Witon et al., 2018) which detects joy more accurately than anger (77% versus 57%), surprise (62%) and fear (69%).

7 Qualitative Evaluation

In this section, we qualitatively evaluate the emotional content and correctness of the responses generated by $\text{EMOTICONS}_{\gamma=\gamma_{opt}}$ compared to the ones from $\text{MMI}_{\text{bas.}}$ through a user study. It consists of three different experiments which measure grammatical correctness, user preference, and emotional appropriateness. For all experiments, we chose prompts from the test set of Cornell, for which the most diverse responses were created by $\text{MMI}_{\text{bas.}}$ in terms of distinct-2. We test EMOTICONS by generating responses according to four emotions: fear, anger, joy, and surprise (beam size of 200).

7.1 Grammatical Correctness

In this experiment, we used 40 prompts. For each prompt, we generated 5 sentences (4 for EMOTICONS, and 1 for $\text{MMI}_{\text{bas.}}$) that were presented in a random order to 3 native English speakers. They assigned either 0 (sentence grammatically incorrect), or 1 (sentence grammatically correct) for all sentences. To measure the agreement across annotators, we calculate Fleiss' $\kappa = 0.4128$, which corresponds to "moderate agreement". Our model does not substantially sacrifice the grammatical correctness of the responses (see Table 3).

7.2 User Preference

In this setting, we quantify how likely the user is going to prefer the response generated by EMOTICONS compared to the one generated by $\text{MMI}_{\text{bas.}}$. We asked 18 annotators to choose their favorite response to the input query among eight proposed answers (top four responses coming from the MMI_{bas} and 4 coming from EMOTICONS with the four different target emotions). Each of 45 sentences were annotated by three different annotators. Results of the experiment (Table 3) indicate that users strongly prefer EMOTICONS over $\text{MMI}_{\text{bas.}}$.

(a) Bas. (b) Fear (c) Anger (d) Joy (e) Surprise

Figure 3: Emotional faces: AffectButton presents faces according to average VAD vectors (in parentheses) obtained for the (a) $\text{MMI}_{\text{bas.}}$ ($[0.47, 0.98, 0.36]$), and for the four EMOTICONS models with different target emotions: (b) Fear ($[0.2, 0.95, 0.38]$), (c) Anger ($[0.54, 0.92, 0.65]$), (d) Joy ($[0.68, 0.97, 0.66]$), and (e) Surprise ($[0.37, 0.97, 0.52]$).

7.3 Emotional Appropriateness

In this experiment, we show that our model is able to generate emotions in a controlled manner. For each of the 5 models, 22 users assign a face via the AffectButton. We generate responses for 120 different prompts. We keep the responses that were annotated with a VAD vector with the norm greater than 2, corresponding to those expressing strong emotions. We compute the average VAD vectors for the annotated sequences for each model, with corresponding AffectButton faces (Figure 3). The majority of user-assigned faces have a high arousal value, which can be explained by the fact that users tend to click in one of the four corners of the AffectButton. The majority of the faces represent an accurate portrayal of the desired emotion. The poor performance of EMOTICONS at expressing surprise comes from the fact that (1) users often mismatch surprise with joy, leading to a neutral dominance value, and (2) surprise is one of the most difficult emotions to judge (see §6).

8 Conclusion

We have presented EMOTICONS, a system that can generate responses with controlled emotions. The flexibility of the presented solution allows it to be used in any kind of neural architecture as long it fits the encoder-decoder framework. Currently, EMOTICONS does not generate different emotions equally well. Future work could include incorporating contextual information that would help EMOTICONS to better capture emotional content.

Acknowledgments

We would like to thank anonymous reviewers for their insightful comments. Mubbasir Kapadia has been funded in part by NSF IIS-1703883, NSF S&AS-1723869, and DARPA SocialSim-W911NF-17-C-0098.

References

Abhaya Agarwal and Alon Lavie. 2008. Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proc. of the Third Workshop on Statistical Machine Translation*, pages 115–118.

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2017. Affective neural response generation. *CoRR*, abs/1709.03968.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Joost Broekens. 2012. In defense of dominance: PAD usage in computational representations of affect. *International Journal of Synthetic Emotions*, 3(1):33–42.

Joost Broekens and Willem-Paul Brinkman. 2013. Affectbutton: A method for reliable and valid affective self-report. *International Journal of Human-Computer Studies*, 71(6):641–667.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proc. of the Workshop on Cognitive Modeling and Computational Linguistics, ACL*.

Paul Ekman, Robert W Levenson, and Wallace V Friesen. 1983. Autonomic nervous system activity distinguishes among emotions. *Science*, 221(4616):1208–1210.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *CoRR*, abs/1805.04833.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holger Hoffmann, Andreas Scheck, Timo Schuster, Steffen Walter, Kerstin Limbrecht, Harald C Traue, and Henrik Kessler. 2012. Mapping discrete emotions into the dimensional space: An empirical approach. In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3316–3320. IEEE.

Chenyang Huang, Osmar Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2 (Short Papers), pages 49–54.

Vladimir Ilievski, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2018. Goal-oriented chatbot dialog management bootstrapping with transfer learning. In *Proc. of IJCAI*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*, pages 110–119.

Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. 2018. Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach. In *Proc. of AAAI Conference on Artificial Intelligence*.

Irina Maslowski, Delphine Lagarde, and Chloé Clavel. 2017. In-the-wild chatbot corpus: from opinion analysis to interaction problem detection. In *Proc. of ICNLSP*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proc. of the International Conference on Language Resources and Evaluation*.

Saif M. Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. In *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *CoRR*, abs/1506.06714.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sarah Strohkorb, Chien-Ming Huang, Aditi Ramachandran, and Brian Scassellati. 2016. Establishing sustained, supportive human-robot relationships: Building blocks and open challenges. In *AAAI Spring Symposium on Enabling Computing Research in Socially Intelligent Human-Robot Interaction*, volume 2123, page 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of Advances in neural information processing systems*, pages 3104–3112.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of Advances in neural information processing systems*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Wojciech Witon, Pierre Colombo, Ashutosh Modi, and Mubbasir Kapadia. 2018. Disney at IEST 2018: Predicting emotions using an ensemble. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 248–253.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proc. of The AAAI Conference on Artificial Intelligence*, pages 730–738.

Multi-Level Memory for Task Oriented Dialogs

Revanth Reddy[1][*], **Danish Contractor**[2], **Dinesh Raghu**[2], **Sachindra Joshi**[2]
[1]Indian Institute of Technology, Madras [2]IBM Research AI, New Delhi
g.revanthreddy111@gmail.com
{dcontrac,diraghu1,jsachind}@in.ibm.com

Abstract

Recent end-to-end task oriented dialog systems use memory architectures to incorporate external knowledge in their dialogs. Current work makes simplifying assumptions about the structure of the knowledge base (such as the use of triples to represent knowledge) and combines dialog utterances (context), as well as, knowledge base (KB) results, as part of the same memory. This causes an explosion in the memory size, and makes reasoning over memory, harder. In addition, such a memory design forces hierarchical properties of the data to be fit into a triple structure of memory. This requires the memory reader to learn how to infer relationships across otherwise connected attributes.

In this paper we relax the strong assumptions made by existing architectures and use separate memories for modeling dialog context and KB results. Instead of using triples to store KB results, we introduce a novel multi-level memory architecture consisting of cells for each query and their corresponding results. The multi-level memory first addresses queries, followed by results and finally each key-value pair within a result. We conduct detailed experiments on three publicly available task oriented dialog data sets and we find that our method conclusively outperforms current state-of-the-art models. We report a 15-25% increase in both entity F1 and BLEU scores.

1 Introduction

Task oriented dialog systems are designed to complete a user specified goal, or service an information request using natural language exchanges. Unlike open domain end-to-end neural dialog models, task oriented systems rely on external knowledge sources, outside of the current conversation context, to return a response (Henderson et al., 2014a; Su et al., 2016; Bordes and Weston, 2017a; Eric and Manning, 2017; El Asri et al., 2017). For instance, in the example shown in Table 1, a dialog agent giving tour package recommendations needs to be able to first query an external knowledge source to determine packages that meet a user's requirement, and then respond accordingly.

Origin	**Dest.**	**Hotel**	**Price**	**Cat.**	**Rating**	**...**
Dallas	Mannheim	Regal Resort	$2800	5.0	8.98	...
Toronto	Calgary	Amusement	$1864.4	4.0	6.91	...
Dallas	Mannheim	Vertex Inn	$3592.8	3.0	7.15	...
Dallas	Santos	Globetrotter	$2000	3.0	8.37	..
Dallas	Mannheim	Starlight	$4018.1	4.0	8.56	
...	...	...	...	...	...	...

Role	**Turn**	**Utterance**
Agent	1	how can i help
User	2	i've got a few days off from aug 26-aug 31. I'm not flexible on this, but i want to somehow treat myself with an 8 day trip. I'm leaving Dallas and I want to check out Munich
Agent	2	sorry we dont have any package for Munich
User	3	how about Mannheiem?
Agent	3	would a 5 day trip suffice
User	4	sure dude
Agent	4	A 5 star hotel called the Regal resort, it has free wifi and a spa
User	5	dates?
Agent	5	starts on august 27th until the 30th
User	6	ok that could work. I would like to see my options in Santos.
Agent	6	there is Hotel Globetrotter in Santos. it has 3 stars and comes with breakfast and wifi, it leaves on august 25th and returns on 30th, all for $2000
User	7	ahh i can't leave until august 26th though. I guess i'll just have to go to Mannheim then.
Agent	7	the Regal resort package costs $2800 with economy class.
User	8	yeah i will book it

Table 1: A goal oriented dialog based on the Frames dataset (El Asri et al., 2017) along with an external knowledge source with each row containing a tour package.

In order to enable end-to-end goal oriented dialog tasks, current state of the art methods use neural memory architectures to incorporate external knowledge (Su et al., 2016; Eric and Manning, 2017; Madotto et al., 2018). As can be seen in Table 1, agent responses may also include entity values present only in the dialog context (eg: "Munich" in the Agent response in Turn 2). In order to support such utterances, models also include tokens from the input dialog context in the same memory (Madotto et al., 2018).

---

[*] Work done during internship at IBM Research AI

Proceedings of NAACL-HLT 2019, pages 3744–3754
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

Existing memory based architectures for task oriented dialog suffer from multiple limitations. First, the creation of a shared memory for copying values from dialog context, as well as the knowledge base (KB) results, forces the use of a common memory reader for two different types of data. This makes the task of reasoning over memory, harder – not only does the memory reader need to determine the right entries from a large memory (since each word from context also occupies a memory cell), it also needs to learn to distinguish between the two forms of data (context words and KB results) stored in the same memory.

Subject	Relation	Object	Subject	Relation	Object
Vertex Inn	Price	\$3592.8	Vertex Inn	Category	3.0
Regal Resort	Price	\$2800	Regal Resort	Rating	8.98
Regal Resort	Category	5.0	Starlight	Price	\$4018.1
Starlight	Rating	8.56	Starlight	Category	4.0
...	...	...	...	...	...

Table 2: Results from Dallas to Mannheim stored in the form of triples.

Second, all current neural memory architectures store results, returned by a knowledge source, in the form of triples (eg. $subject - relation - object$). This modeling choice makes it hard for the memory reader to infer relationships across otherwise connected attributes. For instance, consider the example triple store in Table 2 showing results for a query executed for packages between "Dallas" and "Mannheim". If the user asks the dialog agent to check the price of stay at a 5 star hotel, the memory reader needs to infer that the correct answer is $2800 by learning that the price, category and hotel need to be linked inorder to return an answer (shown in blue).

Lastly, current models treat conversations as a sequential process, involving the use of KB results from only the most recent information request/query. In contrast, in real world dialogs such as the one shown in Table 1, the agent may have to refer to results (*to Mannheim*) from a previously executed query (see Turn 7). Thus, at each turn, the system has to memorize all the information exchanged during the dialog, and infer the package being referred to, by the user. In order to support such dialogs, the memory needs to store results of all queries executed during the course of the dialog. The problem of inferring over such results (which may be from multiple queries) is exacerbated when memory is represented in the form of triples.

In this paper, we present our novel multi-level memory architecture that overcomes the limitations of existing methods: (i) We separate the memory used to store tokens from the input context and the results from the knowledge base. Thus, we learn different memory readers for context words as well for knowledge base entities (ii) Instead of using a $subj - rel - obj$ store, we develop a novel multi-level memory architecture which encodes the natural hierarchy exhibited in knowledge base results by storing queries and their corresponding results and values at each level. We first attend on the queries, followed by the results in each query to identify the result being referred to, by the user. We then attend on the individual entries in the result to determine which value to copy in the response. Figure 1c shows our multi-level memory storing the results from queries executed as part of the dialog in Table 1.

Our paper makes the following contributions:

1. We propose the use of separate memories for copying values from context and KB results. Thus, the model learns separate memory readers for each type of data.

2. Our novel multi-level memory for KB results, models the queries, results and their values in their natural hierarchy. As our experiments show, the separation of memory as well as our multi-level memory architecture, both, contribute to significant performance improvements.

3. We present detailed experiments demonstrating the benefit of our memory architecture along with model ablation studies. Our experiments on three publicly available datasets (**CamRest676** (Su et al., 2016), **InCar assistant** (Eric and Manning, 2017), **Maluuba Frames** (El Asri et al., 2017)) show a substantial improvement of 15-25 % in both entity F1 scores, and BLEU scores as compared to existing state of the art architectures. To the best of our knowledge, we are the first to attempt end-to-end modeling of task oriented dialogs with non-sequential references as well as multiple queries, as seen in the Maluuba Frames dataset. A human evaluation on model outputs also shows our model is preferred by users over existing systems such as KVRet (Eric and Manning, 2017) and Mem2Seq (Madotto et al., 2018).

2 Related work

Recent methods, such as (Vinyals and Le, 2015; Serban et al., 2016, 2017), proposed for end-to-end learning of dialogs were aimed at modeling open-domain dialogs. While they can be used for learning task oriented dialogs, they are not well suited to interface with a structured KB. To better adapt them to handle task oriented dialogs: 1) (Bordes and Weston, 2017b) proposed a memory network based architecture to better encode KB tuples and perform inferencing over them and 2) (Madotto et al., 2018) incorporated copy mechanism to enable copying of words from the past utterances and words from KB while generating responses. All successful end-to-end task oriented dialog networks (Eric and Manning, 2017; Bordes and Weston, 2017b; Madotto et al., 2018) make assumptions while designing the architecture: 1) KB results are assumed to be a triple store, 2) KB triples and past utterances are forced to be represented in a shared memory to enable copying over them. Both these assumptions make the task of inferencing much harder. Any two fields linked directly in the KB tuple are now linked indirectly by the subject of the triples. Further, placing the KB results and the past utterances in same memory forces the architecture to encode them using a single strategy. In contrast, our work uses two different memories for past utterances and KB results. The decoder is equipped with the ability to copy from both memories, while generating the response. The KB results are represented using a multi-level memory which better reflects the natural hierarchy encoded by sets of queries and their corresponding result sets.

Memory architectures have also been found to be helpful in other tasks such as question answering. Work such as (Xu et al., 2016) defines a hierarchical memory architecture consisting of sentence level memory followed by word memory for a QA task while (Chandar et al., 2016) defines a memory structure that speeds up loading and inferencing over large knowledge bases. Recent work by (Chen et al., 2018) uses a variational memory block along with a hierarchical encoder to improve diversity of open domain dialog responses.

3 Multi-Level Memory Network

In this section, we describe our end-to-end model for task oriented dialogues. Our model[1] (Figure 1a) consists of: (i) a hierarchical encoder which encodes the current input context consisting of the user and agent utterances (ii) a multi-level memory that maintains the queries and knowledge base results seen so far in the course of the dialogue, and (iii) copy augmented sequence decoder that uses separate context and multi-level memory. The queries and their corresponding results are maintained in a multi-level memory. The decoder uses a gating mechanism for memory selection while generating a response.

3.1 Encoder

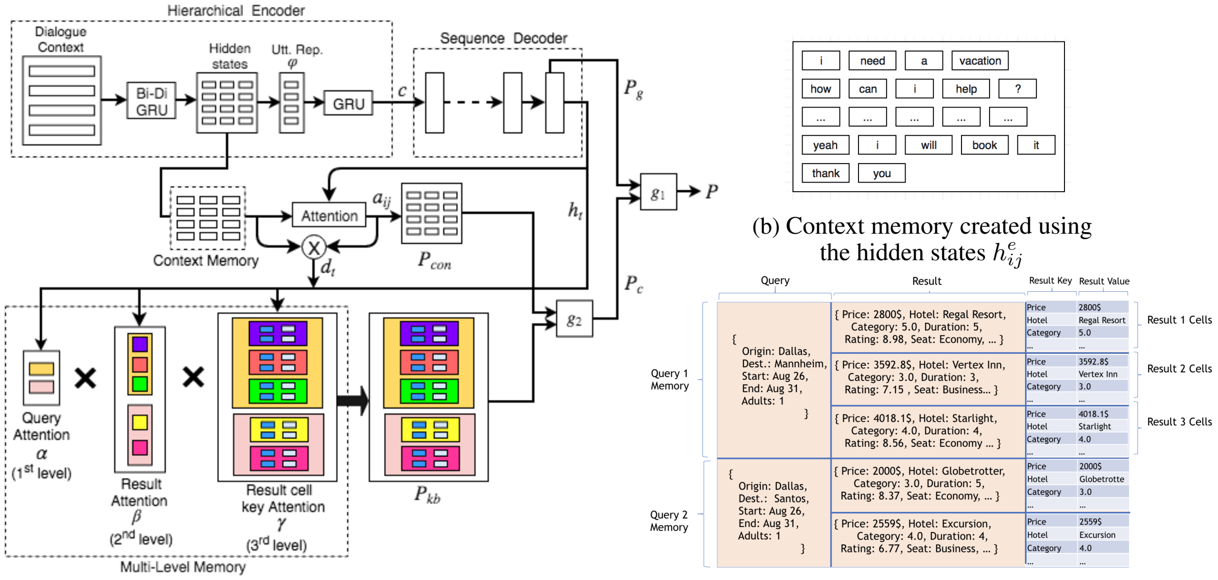
Our model uses a standard hierarchical encoder as proposed by (Sordoni et al., 2015). The encoder takes a sequence of utterances as input. For the t^{th} turn, the dialogue context can be represented as $(c_1, c_2, ...c_{2t-1})$, which consists of t user utterances and $t-1$ system utterances. Each utterance c_i is further a sequence of words $(w_{i1}, w_{i2}, ...w_{im})$. We first embed each word w_{ij} using a word embedding function ϕ^{emb} that maps each word to a fixed-dimensional vector. We then generate utterance representations, $\varphi(c_i)$ using a single layer bi-directional GRU. h^e_{ij} denotes the hidden state of word w_{ij} in the bi-directional GRU. The input representation c is generated by passing each utterance representation $\varphi(c_i)$ through another single layer GRU.

3.2 Multi-level Memory

Motivation: Current approaches break down KB results by flattening them into (*subj-rel-obj*) triples. However, converting KB results into triples leads to loss of relationship amongst attributes in the result set. This makes the reasoning over memory difficult as model now has to infer relationships when retrieving values from memory. Instead, we use a multi-level memory which keeps the natural hierarchy of results intact (without breaking them into triples). We store the queries and their corresponding results and individual values at different *levels*. We first attend on the queries and then on the results for each query to identify which result the user is referring to. This also enables us to handle user requests that refer to results from a previously executed query. We propose that a representation of all the values in the result, and not just one of the values (designated as *subj*), should be used while attending over a result in KB. We attend on this compound representation of the result before attending on the individual key-value pairs in each result, to determine which value to copy into the generated response.

---

[1] Code is available at Multi-Level Memory



(a) Architecture of our model with multi-level memory attention

(b) Context memory created using the hidden states h^e_{ij}

(c) Expanded view of the multi-level KB memory corresponding to example in Table 1

Figure 1: Model architecture (a) along with schematic representation of context memory (b) and multi-level KB memory (c)

3.2.1 Memory Representation

Let $q_1, q_2, ...q_k$ be the queries fired to the knowledge base till the current state of the dialogue. Every query q_i is a set of key-value pairs $\{k^{q_i}_a : v^{q_i}_a, 1 < a < n_{q_i}\}$, corresponding to the query's slot and argument where n_{q_i} is the number of slots in query q_i. For example, after the user utterance at Turn 3 in Table 1, the query fired by the system on the knowledge base would be $\{$*'origin':'Dallas','destination':'Manheim','Start': 'Aug 26', 'end': 'Aug 31', 'Adults':1*$\}$. The execution of a query on an external knowledge base, returns a set of results. Let r_{ij} be the j^{th} result of query q_i. Each result r_{ij} is also a set of slot-value pairs $\{k^{r_{ij}}_a : v^{r_{ij}}_a, 1 < a < n_{r_{ij}}\}$ where $n_{r_{ij}}$ is the number of attributes in result r_{ij}. A visualization of the memory with queries and their corresponding results can be seen in Figure 1c.

The first level of memory contains the query representations. Each query q_i is represented by q^v_i = Bag of words over the word embeddings of values ($v^{q_i}_a$) in q_i. The second level of memory contains the result representations. Representation of each result r_{ij} is given by r^v_{ij} = Bag of words over the word embeddings of values ($v^{r_{ij}}_a$) in r_{ij}. The third level of memory contains the result cells which have the key-value pairs ($k^{r_{ij}}_a : v^{r_{ij}}_a$) of the results. The values ($v^{r_{ij}}_a$) which are to be copied into the system response are thus present in the final level of memory . We now describe how we apply attention over the context and multi-level memory.

3.3 Decoder

The model generates the agent response word-by-word; a word at time step t is either generated from the decode vocabulary or is a value copied from one of the two memories (knowledge base or context memory). A soft gate g_1 controls whether a value is generated from vocabulary or copied from memory. Another gate g_2 determines which of the two memories is used to copy values.

3.3.1 Generating words:

Let the hidden state of the decoder at time t be h_t.

$$h_t = GRU(\phi^{emb}(y_{t-1}), h_{t-1}) \tag{1}$$

The hidden state h_t is used to apply attention over the input context memory. Attention is applied over the hidden states of the input bi-directional (BiDi) GRU encoder using the "*concat*" scheme as given in (Luong et al., 2015). The attention for the jth word in the ith utterance is given by:

$$a_{ij} = \frac{exp(w_1^T tanh(W_2 tanh(W_3[h_t, h^e_{ij}])))}{\sum_{ij} exp(w_1^T tanh(W_2 tanh(W_3[h_t, h^e_{ij}])))} \tag{2}$$

The attention scores a_{ij} are combined to create an attended context representation d_t,

$$d_t = \sum_{i,j} a_{i,j} h^e_{ij} \tag{3}$$

and similar to (Luong et al., 2015), the decoder word-generation distribution is given by :

$$P_g(y_t) = softmax(W_1[h_t, d_t] + b_1) \tag{4}$$

3.3.2 Copying words from context memory:

The input context memory is represented using the hidden states h^e_{ij} of the input Bi-Di GRU encoder. Similar to (Gulcehre et al., 2016), the attention scores a_{ij}, are used as the probability scores to form the copy distribution $P_{con}(y_t)$ over the input context memory.

$$P_{con}(y_t = w) = \sum_{ij:w_{ij}=w} a_{ij} \tag{5}$$

3.3.3 Copying entries from KB memory:

The context representation d_t, along with the hidden state of decoder h_t, is used to attend over the multi-level memory. The first level attention, $\alpha_.$, is applied over the queries $q_.$.

$$\alpha_i = \frac{exp(w_2^T tanh(W_4[d_t, h_t, q^v_i]))}{\sum_i exp(w_2^T tanh(W_4[d_t, h_t, q^v_i]))} \tag{6}$$

The second level attention, $\beta_{i.}$, is the attention over the results $r_{i.}$ of query q_i.

$$\beta_{ij} = \frac{exp(w_3^T tanh(W_5[d_t, h_t, r^v_{ij}]))}{\sum_j exp(w_3^T tanh(W_5[d_t, h_t, r^v_{ij}]))} \tag{7}$$

The product of first level attention and second level attention is the attention over results of all the queries in the multi-level memory. The weighted sum of the first level attention, second level attention and result representations gives us the attended memory representation, m_t.

$$m_t = \sum_i \sum_j \alpha_i \beta_{ij} r^v_{ij} \tag{8}$$

Each result is further composed of multiple result cells. On the last level of memory, which contains the result cells, we apply key-value attention similar to (Eric and Manning, 2017). The key of the result cell is the word embedding of the slot, $k^{r_{ij}}_a$, in the result. The attention scores, $\gamma_{ij.}$, for the keys represent the attention over the result cells of each result r_{ij}.

$$\gamma_{ijl} = \frac{exp(w_4^T tanh(W_6[d_t, h_t, \phi^{emb}(k^{r_{ij}}_l)]))}{\sum_l exp(w_4^T tanh(W_6[d_t, h_t, \phi^{emb}(k^{r_{ij}}_l)]))} \tag{9}$$

The product of first level attention α_i, second level attention β_{ij} and third level attention γ_{ijl} gives the final attention score of the value $v^{r_{ij}}_l$ in the KB memory. These final attention scores when combined (Eq. 10), form the copy distribution, $P_{kb}(y_t)$, over the values in KB memory.

$$P_{kb}(y_t = w) = \sum_{ijl:v^{r_{ij}}_l = w} \alpha_i \beta_{ij} \gamma_{ijl} \tag{10}$$

3.3.4 Decoding

Similar to (Gulcehre et al., 2016), we combine the generate and copy distributions - we use gate g_2 (Eq. 11) to obtain the copy distribution $P_c(y_t)$ (Eq. 12) by combining $P_{kb}(y_t)$ and $P_{con}(y_t)$.

$$g_2 = sigmoid(W_7[h_t, d_t, m_t] + b_2) \tag{11}$$

$$P_c(y_t) = g_2 P_{kb}(y_t) + (1 - g_2) P_{con}(y_t) \tag{12}$$

Finally, we use gate g_1 to obtain the final output distribution $P(y_t)$, by combining generate distribution $P_g(y_t)$ and copy distribution $P_c(y_t)$ as shown below:

$$g_1 = sigmoid(W_8[h_t, d_t, m_t] + b_3) \tag{13}$$

$$P(y_t) = g_1 P_g(y_t) + (1 - g_1) P_c(y_t) \tag{14}$$

We train our model by minimizing the cross entropy loss $-\sum_{t=1}^{T} log(P(y_t))$.

4 Experiments

4.1 Datasets

We present our experiments using three real world publicly available multi-turn task oriented dialogue datasets: the InCar assistant (Eric and Manning, 2017), CamRest (Su et al., 2016) and the Maluuba Frames dataset (El Asri et al., 2017). All three datasets contain human-human task oriented dialogues which were collected in a Wizard-of-Oz (Wen et al., 2017) setting.

(i) **InCar assistant dataset** consists of 3031 multi-turn dialogues in three distinct domains: calendar scheduling, weather information retrieval, and point-of-interest navigation. Each dialogue has it's own KB information provided and thus, the system does not have to make any queries.

(ii) **CamRest dataset**, consists of 676 human-to-human dialogues set in the restaurant reservation domain. There are three queryable slots (food, price range, area) that users can specify. This dataset has currently been used for evaluating slot-tracking systems. Recent work by (Lei et al., 2018) uses an end-to-end network without a KB and substitutes slot values with placeholders bearing the slot names in agent responses. However, we formatted the data to evaluate end-to-end systems by adding API call generation from the slot values so that restaurant suggestion task can proceed from the KB results.

(iii) **Maluuba Frames dataset**, consists of 1369 dialogues developed to study the role of memory in task oriented dialogue systems. The dataset is set in the domain of booking travel packages which involves flights and hotels. In contrast to the previous two datasets, this dataset contains dialogs that require the agent to remember all information presented previously as well as support results from multiple queries to the knowledge base. A user's preferences may change as the dialogue proceeds, and can also refer to previously presented queries (non-sequential dialog). Thus, to store multiple queries, we require 3 levels in our multi-level memory as compared to 2 levels in the other datasets, since they don't have more than one query. We do not use the dialogue frame annotations and use only the raw text of the dialogues. We map ground-truth queries to API calls that are also required to be generated by the model. Recent work has used this dataset only for frame tracking (Schulz et al., 2017) and dialogue act prediction (Peng et al., 2017; Tang et al., 2018). To the best of our knowledge we are the first to attempt the end-to-end dialog task using this dataset. Table 3 summarizes the statistics of the datasets.

	InCar	CamRest	Maluuba Frames
Train Dialogs	2425	406	1095
Val Dialogs	302	135	137
Test Dialogs	304	135	137
Avg. no. of turns	2.6	5.1	9.4
Avg length. of sys. resp.	8.6	11.7	14.8
Avg no. of sys. entities	1.6	1.7	2.9
Avg no. of queries	0	1	2.4
Avg no. of KB entries	66.1	13.5	141.2

Table 3: Statistics for 3 different datasets.

4.2 KB API Call Generation

In this section, we briefly describe how the knowledge base queries are generated as API calls as part of the model response. The InCar assistant dataset has a fixed KB for each dialogue whereas the CamRest and Maluuba datasets require queries to be fired on a global KB. Queries in CamRest dataset can have 3 slots namely cuisine, area and pricerange, whereas those in Maluuba can have 8 slots, which are destination, origin, start date, end date, budget, duration, number of adults and children. Any query that is to be fired on the KB is expected to be generated by the model as an API call, by considering a fixed ordering of slots in the generated response. For eg., in CamRest dataset, ApiCall(area=south, pricerange=cheap) would be generated by the model as *api_call dontcare south cheap*, with *dontcare* meaning that the user does not have any preference for cuisine and, *south*, *cheap* being the user constraints for area and pricerange respectively. Therefore, the task of API call generation typically involves copying relevant entities that are present in dialog context.

4.3 Training

Our model is trained end-to-end using Adam optimizer (Kingma and Ba, 2014) with a learning rate of $2.5e^{-4}$. The batch-size is sampled from [8,16]. We use pre-trained Glove vectors (Pennington et al., 2014) with an embedding size of 200. The GRU hidden sizes are sampled from [128, 256]. We tuned the hyper-parameters with grid search over the validation set and selected the model which gives best entity F1.

4.4 Evaluation Metrics

4.4.1 BLEU

We use the commonly used BLEU metric (Papineni et al., 2002) to study the performance of our systems as it has been found to have strong correlation (Sharma et al., 2017) with human judgments in task-oriented dialogs.

4.4.2 Entity F1

To explicitly study the behaviour of different memory architectures, we use the entity $F1$ to measure how effectively values from the knowledge base are used in the dialog. To compute the entity F1, we micro-average the precision and recall over the entire set of system responses to compute the micro F1[2]. For the InCar Assistant dataset, we compute a per-domain entity F1 as well as the aggregated entity F1. Since our model does not have slot-tracking by design, we evaluate

---

[2] We observe that (Madotto et al., 2018) reports the micro average of recall as the micro F1.

	InCar					CamRest		Maluuba Frames	
Model	BLEU	F1	Calendar F1	Weather F1	Navigate F1	BLEU	F1	BLEU	F1
Attn seq2seq (Luong et al., 2015)	11.3	28.2	36.9	35.7	10.1	7.7	25.3	3.7	16.2
Ptr-UNK (Gulcehre et al., 2016)	5.4	20.4	22.1	24.6	14.6	5.1	40.3	5.6	25.8
KVRet (Eric and Manning, 2017)	13.2	48.0	62.9	47.0	41.3	13.0	36.5	10.7	31.7
Mem2Seq (Madotto et al., 2018)	11.8	40.9	61.6	39.6	21.7	14.0	52.4	7.5	28.5
Multi-level Memory Model (MM)	**17.1**	**55.1**	**68.3**	**53.3**	**44.5**	**15.9**	**61.4**	**12.4**	**39.7**

Table 4: Comparison of our model with baselines

on entity F1 instead of the slot-tracking accuracy as in (Henderson et al., 2014b; Wen et al., 2017)

4.5 Baselines

We experiment with the following baseline models for comparing the performance of our Multi-Level Memory architecture:

Attn seq2seq[3] (Luong et al., 2015): A model with simple attention over the input context at each time step during decoding.

Ptr-UNK[3] (Gulcehre et al., 2016): The model augments a sequence-to-sequence architecture with attention-based copy mechanism over the encoder context.

KVRet (Eric and Manning, 2017): The model uses key value knowledge base in which the KB is represented as triples in the form of $subject - relation - object$. This model does not support copying words from context. The sum of word embeddings of $subject$, $relation$ is used as the key of the corresponding $object$.

Mem2Seq[3] (Madotto et al., 2018): The model uses a memory networks based approach for attending over dialog history and KB triples. During decoding, at each time step, the hidden state of the decoder is used to perform multiple hops over a single memory which contains both dialog history and the KB triples to get the pointer distribution used for generating the response.

4.6 Results

Table 4 shows the performance of our model against our baselines. We find that our multi- level memory architecture comprehensively beats all existing models, thereby establishing new state-of-the- art benchmarks on all three datasets. Our model outperforms each baseline on both BLEU and entity F1 metrics.

InCar: On this dataset, we show entity F1 scores for each of the scheduling, weather and navigation domains. Our model has the highest F1 scores across all the domains. It can be seen that our model strongly outperforms Mem2Seq on each domain. A detailed study reveals that the use of triples cannot handle cases when a user queries with *non-subject* entries or in cases when the response requires inferencing over multiple entries. In contrast, our model is able to handle such cases since we use a compound representation of entire result (bag of words over values) while attending on that result.

CamRest: Our model achieves the highest BLEU and entity F1 scores on this dataset. From Table 4, we see that simpler baselines like Ptr-UNK show competitive performance on this dataset because, as shown in Table 3, CamRest dataset has relatively fewer KB entries. Thus, a simple mechanism for copying from context results in good entity F1 scores.

	InCar		**CamRest**		**Maluuba**	
	Ctxt.	KB	Ctxt.	KB	Ctxt.	KB
Mem2Seq	66.2	25.3	63.7	36.5	17.7	8.9
Multi-level Mem.	81.6	37.5	70.1	53.4	27.2	14.6

Table 5: Percentage (%) of category-wise (context vs KB) ground truth entities correctly captured in generated response. Abbreviation Ctxt denotes context.

Maluuba Frames: The Maluuba Frames dataset was introduced for the frame tracking task. Here, a dialog frame is a structured representation of the current dialog state. Instead of explicitly modeling the dialog frames, we use the context representation d_t to directly attend on the Multi-level memory. As Table 3 shows, this dataset contains significantly longer contexts as well as larger number of entities, as compared to the other two datasets. In addition, unlike other datasets, it also contains non-linear dialog flows where a user may refer to previously executed queries and results. The complexity of this dataset is reflected in the relatively lower BLEU and F1 scores as compared to other datasets.

4.7 Analysis

4.7.1 Entity source-wise performance

To further understand the effect of separating context memory from KB memory and using a multi-

---

[3] We use the implementation provided by (Madotto et al., 2018) at https://github.com/HLTCHKUST/Mem2Seq

	InCar					CamRest		Maluuba Frames	
Model	BLEU	F1	Calendar F1	Weather F1	Navigate F1	BLEU	F1	BLEU	F1
Unified Context and KB memory (Mem2Seq)	11.8	40.9	61.6	39.6	21.7	14.0	52.4	7.5	28.5
Separate Context and KB Memory	14.3	44.2	56.9	54.1	24.0	14.3	55.0	12.1	36.5
+Replace KB Triples with Multi-level memory	17.1	55.1	68.3	53.3	44.5	15.9	61.4	12.4	39.7

Table 6: Model ablation study : Effect of (i) separate memory and (ii) multi-level memory design.

level memory for KB, Table 5 shows the percentage of ground-truth entities, according to their category, which were also present in the generated response. For example, on the InCar dataset, out of the 930 entities in ground-truth response that were to be copied from the KB, our model was able to copy 37.5% of them into the generated response. From Table 5, it can be seen that our model is able to correctly copy a significantly larger number of entities from both, KB and context, as compared to the recent Mem2Seq model in all datasets.

4.7.2 Model ablation study

We report results from ablation studies on all three datasets. Table 6 shows the incremental benefit obtained from individual components used in our model. We investigate the gains made by (i) Using separate memory for context and KB triples (ii) Replacing KB triples with a Multi-level memory. We use the recent Mem2Seq model for comparison with a unified context and KB memory model.

As can be seen from Table 6, the separation of context memory and KB memory leads to a significant improvement in BLEU and F1 scores on all datasets. This validates our hypothesis that storing context words and KB results in a single memory confuses the memory reader. The use of a multi-level memory instead of triples leads to further gains. This suggests, better organization of KB result memory by keeping the natural hierarchy intact is beneficial.

4.7.3 Error Analysis

We analyzed the errors made by our dialog model on 100 dialog samples in test set of Maluuba Frames. We observed that the errors can be divided into five major classes: (i) Model outputs wrong KB result entry due to incorrect attention (27%), (ii) Model returns package details instead of asking for more information from the user (16%), (iii) Model incorrectly captures user intent (13%), (iv) Model makes an error due to non-sequential nature of dialog (22%). In such errors, our model either generates an API call for a result already present in memory, or our model asks for a query-slot value that was already provided by the user, (v) Data specific characteristics such as insufficient samples for certain classes of utterances (eg: more than one package returned) or returning different, but meaningful package attributes as compared to ground-truth data, contribute to 22% of the errors.

	CamRest			**Maluuba**		
	Info.	Lang.	MRR	Info.	Lang.	MRR
KVRet	2.49	4.38	0.57	2.42	3.31	0.64
Mem2Seq	2.48	3.72	0.51	1.78	2.55	0.50
Our Model	**3.62**	**4.48**	**0.76**	**2.45**	**3.93**	**0.69**

Table 7: Informational accuracy and language quality scores using a human study.

4.8 Human Evaluation

We also conducted a blind user study that compared outputs from our model, Mem2Seq and KVRet systems. We used 96 randomly selected examples from each test split of Maluuba and CamRest datasets resulting in a total of 192 examples. Our study was split across 8 users who were provided with results fetched from the KB, current dialog context, gold response and the outputs of each of the models. Model outputs were shuffled in each example and users were asked to score each output between 1 (lowest) to 5 (highest) in terms of its accuracy of information in response and the quality of language. The results of this study are presented in Table 7. We also report the MRR (mean-reciprocal rank) for model preference along with other scores.

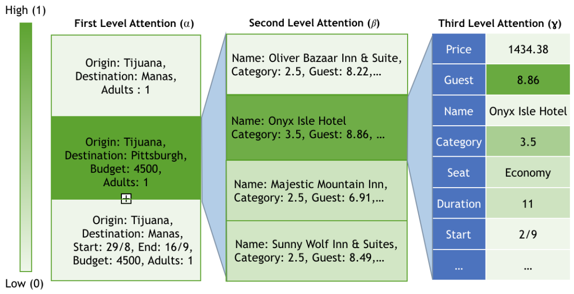
It can be seen that our model consistently ranks high for both information accuracy and language quality as well as reports a higher MRR. To further understand the quality of model performance, we asked the human evaluators whether their best ranked model output was a useful response. We saw that the evaluators agreed in 76.04% and 58.33% of the cases for CamRest and Maluuba datasets respectively. We observe that the results from human evaluation go hand-in-hand with automatic evaluation and reinforce our claim that separating context, KB memory and using a multi-level representation for the KB memory are useful for improving dialog modeling.

Role	Turn	Utterance
Agent	1	hello ! how can i help you today ?
User	2	*i just need to dream a bit. i have 4500 dollars but no vacation days. i'm wondering what a hypothetical trip to Manas would be like*
Agent	2	i can help with that, where are you departing from
User	3	*i'm in Tijuana, not departing any time soon unfortunately though*
...	...	...
Agent	5	`[API_CALL manas tijuana 29/8 16/9 4500 ...]`
...	...	...
User	7	*that's fine, what about pittsburgh*
Agent	7	`[API_CALL pittsburgh tijuana ...  4500 ...]`
...	...	...
Agent	8	that i do have! would you be satisfied with a 3.5 star hotel ?
User	9	*as of now i'm satisfied with a tent and a backpack .* *what are the hotel details?*
Agent	9	the hotel is situated near a park, comes with free parking, wifi and breakfast ! does this sound good
User	10	*sounds quite nice, what about the guest rating ?*
Gold		the guest rating is impressive : **8.86**
Mem2Seq		this is the only available with the same
KVRet		it's a 2.5 star hotel near a park, museum and airport.it has a guest rating of 8.22
Our Model		the onyx isle hotel has a **8.86** guest rating and offers free parking, breakfast and wifi.

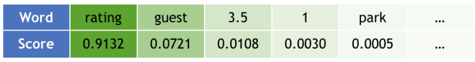
Table 8: Comparing the responses generated by various models on an example in test set of Maluuba Frames.

4.9 Attention Visualization

Analyzing the attention weights is a useful way to understand how the model is inferencing over the memory to copy entities from it. Table 8 shows an example of a dialog from the Maluuba Frames dataset and the outputs generated by different models. Here, the user first wants to know about packages to 'Manas' and then requests for trips to 'Pittsburgh'. Later, the user becomes interested in the 3.5 star hotel in Pittsburgh which was suggested by the agent and wants to know its guest rating. It can be seen from Table 8 that our model outputs the correct guest rating (8.86) of the hotel. Mem2Seq fails to understand the context and generates an irrelevant response. KVRet generates a readable response but points to the guest rating of a different hotel.



(a) Attention over the multi-level KB memory



Word	rating	guest	3.5	1	park	...
Score	0.9132	0.0721	0.0108	0.0030	0.0005	...

(b) Decreasing order of attention scores over words in dialogue context

Figure 2: Visualization of attention over memory while generating the word '8.86' for the example in Table 8.

The attention over the memory while generating the word '8.86' for this example is shown in Fig 2. Fig 2a shows that the query with destination as 'Pittsburgh' gets the highest attention and among the results of this query, the package with the 3.5 star rated hotel gets highest attention. Within this result, the model gives highest score to the result cell with guest rating as the key. To further understand why the correct result hotel gets higher attention, Fig 2b shows the attention scores over the words in context memory. The context representation d_t captures the important words (3.5, guest, rating) in context which are in-turn used to apply attention over the multi-level memory.

Lastly, studying the values of the gates g_1 (prob. of generating from vocabulary) and g_2 (prob. of copying from KB), we found that gate g_1 had a probability value of *0.08* thereby driving the model to copy from memory instead of generating from output vocabulary and gate g_2, with a probability value of *0.99*, was responsible for selecting KB memory over context memory.

5 Conclusion

In this paper, we presented an end-to-end trainable novel architecture with multi-level memory for task oriented dialogues. Our model separates the context and KB memory and combines the attention on them using a gating mechanism. The multi-level KB memory reflects the natural hierarchy present in KB results. This also allows our model to support non-sequential dialogs where a user may refer to a previously suggested result. We find that our model beats existing approaches by 15-25% on both entity F1 and BLEU scores, establishing state-of-the-art results on three publicly available real-world task oriented dialogue datasets. In a user study comparing outputs from our system against recent models, we found that our model consistently scored higher for both language quality as well as correctness of information in the response. We also present the benefits of each of our design choices by performing an ablation study. In future work, we would like to incorporate better modeling of latent dialog frames so as to improve the attention signal on our multi-level memory. As our error analysis suggests, nearly 22% of the errors could possibly be reduced by improved modeling of the dialog context. We believe that model performance can also be improved by capturing user intent better in case of non-sequential dialog flow.

References

Antoine Bordes and Jason Weston. 2017a. Learning end-to-end goal-oriented dialog. *ICLR*, abs/1605.07683.

Antoine Bordes and Jason Weston. 2017b. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *CoRR*, abs/1605.07427.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1653–1662, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *CoRR*, abs/1704.00057.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1437–1447.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hannes Schulz, Jeremie Zumer, Layla El Asri, and Shikhar Sharma. 2017. A frame tracking model for memory-enhanced dialogue systems. *arXiv preprint arXiv:1706.01690*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Jian Su, Xavier Carreras, and Kevin Duh, editors. 2016. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics.

Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li, and Tony Jebara. 2018. Subgoal discovery for hierarchical dialogue policy learning. *arXiv preprint arXiv:1804.07855*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics.

Jiaming Xu, Jing Shi, Yiqun Yao, Suncong Zheng, Bo Xu, and Bo Xu. 2016. Hierarchical memory networks for answer selection on unknown words. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2290–2299.

Topic Spotting using Hierarchical Networks with Self Attention

Pooja Chitkara[1], **Ashutosh Modi**[1],
Pravalika Avvaru[1], **Sepehr Janghorbani**[1,2], **Mubbasir Kapadia**[1,2]
[1]Disney Research, [2]Rutgers University
`pchitkar@andrew.cmu.edu`
`ashutosh.modi@disneyresearch.com`
`pavvaru@andrew.cmu.edu`
`sepehr.janghorbani@rutgers.edu`
`mubbasir.kapadia@rutgers.edu`

Abstract

Success of deep learning techniques have renewed the interest in development of dialogue systems. However, current systems struggle to have consistent long term conversations with the users and fail to build rapport. Topic spotting, the task of automatically inferring the topic of a conversation, has been shown to be helpful in making a dialog system more engaging and efficient. We propose a hierarchical model with self attention for topic spotting. Experiments on the Switchboard corpus show the superior performance of our model over previously proposed techniques for topic spotting and deep models for text classification. Additionally, in contrast to offline processing of dialog, we also analyze the performance of our model in a more realistic setting i.e. in an online setting where the topic is identified in real time as the dialog progresses. Results show that our model is able to generalize even with limited information in the online setting.

1 Introduction

Recently, a number of commercial conversation systems have been introduced e.g. Alexa, Google Assistant, Siri, Cortana, etc. Most of the available systems perform well on goal-oriented conversations which spans over few utterances in a dialogue. However, with longer conversations (in open domains), existing systems struggle to remain consistent and tend to deviate from the current topic during the conversation. This hinders the establishment of long term social relationship with the users (Dehn and Van Mulken, 2000). In order to have coherent and engaging conversations with humans, besides other relevant natural language understanding (NLU) techniques (Jokinen and McTear, 2009), a system, while responding, should take into account the topic of the current conversation i.e. Topic Spotting.

Topic spotting has been shown to be important in commercial dialog systems (Bost et al., 2013; Jokinen et al., 2002) directly dealing with the customers. Topical information is useful for speech recognition systems (Iyer and Ostendorf, 1999) as well as in audio document retrieval systems (Hazen et al., 2007; Hazen, 2011). Importance of topic spotting can be gauged from the work of Alexa team (Guo et al., 2018), who have proposed topic based metrics for evaluating the quality of conversational bots. The authors empirically show that topic based metrics correlate with human judgments.

Given the importance of topical information in a dialog system, this paper proposes self attention based hierarchical model for predicting topics in a dialog. We evaluate our model on Switchboard (SWBD) corpus (Godfrey et al., 1992) and show that our model supersedes previously applied techniques for topic spotting. We address the evaluative limitations of the current SWBD corpus by creating a new version of the corpus referred as SWBD2. We hope that SWBD2 corpus would provide a new standard for evaluating topic spotting models. We also experiment with an online setting where we examine the performance of our topic classifier as the length of the dialog is varied and show that our model can be used in a real time dialog system as well.

2 Related Work

Topic spotting is the task of detecting the topic of a dialog (Hazen et al., 2007). Topic spotting has been an active area of research over the past few decades both in the NLP community as well as in the speech community. In this section we briefly outline some of the main works in this area. For a detailed survey of prior research in this area, the reader is referred to Hazen (2011).

Most of the methods proposed for topic spotting use features extracted from transcribed text as in-

Proceedings of NAACL-HLT 2019, pages 3755–3761
Minneapolis, Minnesota, June 2 - June 7, 2019.

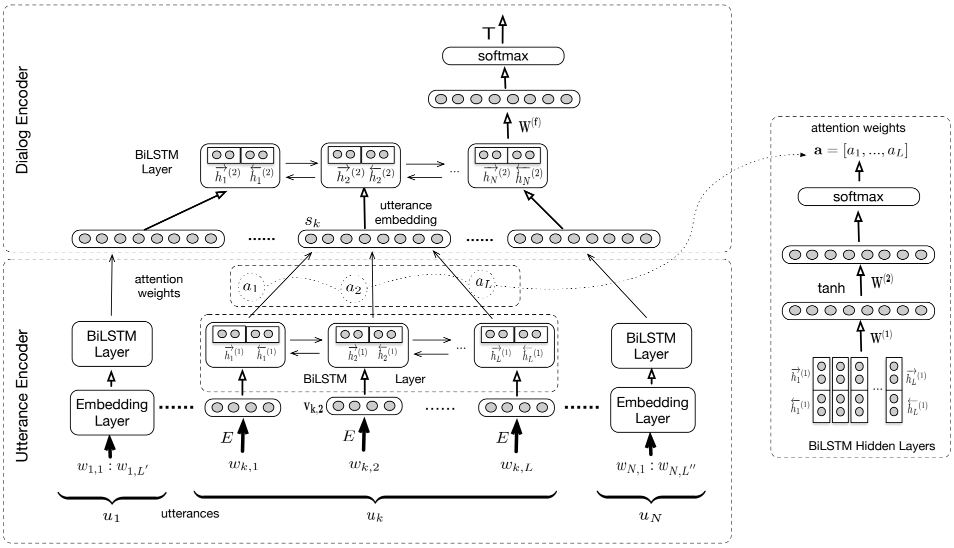


Figure 1: Model Architecture

put to a classifier (typically Naïve Bayes or SVM). Extracted features include: Bag of Words (BoW), TF-IDF (Sparck Jones, 1972; Schütze et al., 2008), n-grams, and word co-occurrences (Hazen, 2011; Myers et al., 2000). Some approaches (in addition to word co-occurrences features) incorporate background world knowledge using Wikipedia (Gupta and Ratinov, 2007). In our work, we do not explicitly extract the features but learn these during training. Moreover, unlike previous approaches, we explicitly model the dependencies between utterances via self attention mechanism and hierarchical structure.

Topic spotting has been explored in depth in the speech processing community (see for example, Wright et al. (1996); Kuhn et al. (1997); Nöth et al. (1997); Theunissen (2002)). Researchers in this community have attempted to predict the topic directly from the audio signals using phoneme based features. However, the performance of word based models supersedes those of audio models (Hazen et al., 2007).

Recently, there has been lot of work in deep learning community for text classification (Kalchbrenner et al., 2014; Zhang et al., 2015; Lai et al., 2015; Lin et al., 2015; Tang et al., 2015). These deep learning models use either RNN-LSTM based neural networks (Hochreiter and Schmidhuber, 1997) or CNN based neural networks (Kim, 2014) for learning representation of words/sentences. We follow similar approach for topic spotting. Our model is related to the Hierarchical Attention Network (HN-ATT) model proposed by Yang et al. (2016) for document classification. HN-ATT models the document hierarchically by composing words (with weights determined by first level of attention mechanism) to get sentence representations and then combines the sentence representations with help of second level attention to get document representation which is then used for classification.

The aim of this paper is not to improve text classification but to improve topic spotting. Topic spotting and text classification differ in various aspects. We are among the first to show the use of hierarchical self attention (HN-SA) model for topic spotting. It is natural to consider applying text classification techniques for topic spotting. However, as we empirically show in this paper, text classification techniques do not perform well in this setting. Moreover, for the dialog corpus simple BoW approaches perform better than more recently proposed HN-ATT model (Yang et al., 2016).

3 Hierarchical Model with Self Attention

We propose a hierarchical model with self attention (HN-SA) for topic spotting. We are given a topic label for each dialog and we want to learn a model mapping from space of dialogues to the space of topic labels. We learn a prediction model by minimizing Negative Log Likelihood ($\mathcal{NLL}$) of the data.

3.1 Model Architecture

We propose a hierarchical architecture as shown in Figure 1. An *utterance encoder* takes each utterance in the dialog and outputs the corresponding utterance representation. A *dialog encoder* processes the utterance representations to give a compact vector representation for the dialog which is used to predict the topic of the dialog.

Utterance Encoder: Each utterance in the dialog is processed sequentially using single layer Bidirectional Long Short Term Memory (BiLSTM) (Dyer et al., 2015) network and self-attention mechanism (Vaswani et al., 2017) to get the utterance representation. In particular, given an utterance with one-hot encoding for the tokens, $u_k = \{\mathbf{w_{k,1}}, \mathbf{w_{k,2}}, ..., \mathbf{w_{k,L}}\}$, each token is mapped to a vector $\mathbf{v_{k,i}} = \mathbf{E}\mathbf{w_{k,i}} \;\; ; i = 1, 2, ...L$ using pretrained embeddings (matrix $\mathbf{E}$).

Utterance representation ($\mathbf{s_k} = \mathbf{a}^T\mathbf{H^{(1)}}$) is the weighted sum of the forward and backward direction concatenated hidden states at each step of the BiLSTM ($\mathbf{H}^{(1)} = [\mathbf{h_1^{(1)}}, ..., \mathbf{h_L^{(1)}}]^T$ where $\mathbf{h_i^{(1)}} = [\overrightarrow{\mathbf{h_i}}^{(1)} : \overleftarrow{\mathbf{h_i}}^{(1)}] = \mathbf{BiLSTM}(\mathbf{v_{k,i}})$). The weights of the combination ($\mathbf{a} = \text{softmax}(\mathbf{h_a^{(2)}})$) are determined using self-attention mechanism proposed by Vaswani et al. (2017) by measuring the similarity between the concatenated hidden states ($\mathbf{h_a^{(2)}} = \mathbf{W_a^{(2)}}\mathbf{h_a^{(1)}} + \mathbf{b_a^{(2)}}$ and $\mathbf{h_a^{(1)}} = \tanh(\mathbf{W_a^{(1)}}\mathbf{H}^{(1)} + \mathbf{b_a^{(1)}})$) at each step in the utterance sequence. Self-attention computes the similarity of a token in the context of an utterance and thus, boosts the contribution of some keywords to the classifier. It also mitigates the need for a second layer of attention at a dialog level reducing the number of parameters, reducing the confusion of the classifier by not trying to reweigh individual utterances and reducing the dependence on having all utterances (full future context) for an accurate prediction. A simple LSTM based model (HN) and HN-ATT perform worse than the model using self attention (§5), indicating the crucial role played by self-attention mechanism.

Dialog Encoder: Utterance embeddings (representations) are sequentially encoded by a second single layer BiLSTM to get the dialog representation ($\mathbf{h_k^{(2)}} = [\overrightarrow{\mathbf{h_k}}^{(2)} : \overleftarrow{\mathbf{h_k}}^{(2)}] = \mathbf{BiLSTM}(\mathbf{s_k}) \;\; ; k = 1, 2, ...N$). Bidirectional concatenated hidden state corresponding to the last utterance (i.e. last step of BiLSTM) is used for making a prediction via a linear layer followed by softmax activation ($p(\mathsf{T}|\mathsf{D}) = \text{softmax}(\mathbf{h_D})$ where $\mathbf{h_D} = \mathbf{W_f}\mathbf{h_N^{(2)}}$).

	# Dialogues		# Topics		Avg. # Utterances	
	SWBD	**SWBD2**	**SWBD**	**SWBD2**	**SWBD**	**SWBD2**
Train	1024	877	66	42	192.27	194.33
Dev	112	49	48	33	180.52	177.02
Test	19	98	12	42	237.58	201.97

Table 1: Corpus statistics for both versions of SWBD

4 Experimental Setup

As in previous work (§2), we use Switchboard (SWBD) (Godfrey et al., 1992) corpus for training our model. SWBD is a corpus of human-human conversations, created by recording (and later transcribing) telephonic conversations between two participants who were primed with a topic. Table 1 gives the corpus statistics. Topics in SWBD range over a variety of domains, for example, politics, health, sports, entertainment, hobbies, etc., making the task of topic spotting challenging.

Dialogues in the test set of the original SWBD cover a limited number of topics (12 vs 66). The test set is not ideal for evaluating topic spotting system. We address this shortcoming by creating a new split and we refer to this version of the corpus as *SWBD2*. The new split provides opportunity for more rigorous evaluation of a topic spotting system. SWBD2 was created by removing infrequent topics ($<$ 10 dialogues) from the corpus and then randomly moving dialogues between the train/development set and the test set, in order to have instances of each topic in the test set. The majority class baseline in SWBD2 is around 5%.

In transcribed SWBD corpus some punctuation symbols such as #, ?, have special meanings and non-verbal sounds have been mapped to special symbols e.g. <Laughter>. To preserve the meanings of special symbols we performed minimal preprocessing. Dialog Corpora is different from text classification corpora (e.g. product reviews). If we roughly equate a dialog to a document and an utterance to a sentence, dialogs are very long documents with short sentences. Moreover, the vocabulary distribution in a dialog corpus is fundamentally different, e.g. presence of back-channel words like 'uhm' and 'ah'.

Model Hyper-parameters: We use GloVe embeddings (Pennington et al., 2014) with dimensionality of 300. The embeddings are updated during training. Each of the LSTM cell in the utterance and dialog encoder uses hidden state of dimension 256. The weight matrices in the attention network have dimension of 128. The hyper-parameters were found by experimenting with the

Models	**SWBD**	**SWBD2**
BoW + Logsitic	78.95	87.76
BoW + SVM	73.68	**90.82**
Bigram + SVM	52.63	79.59
BoW + TF-IDF + Logistic	52.63	81.63
nGram + Logistic	52.63	78.57
nGram + TF-IDF + Logistic	57.89	87.76
Bag of Means + Logistic	78.95	87.76
Avg. Skipgram + Logistic	26.32	59.18
Doc2Vec + SVM	73.68	86.73
HN	31.58	54.08
HN-ATT (Yang et al., 2016)	73.68	85.71
CNN (Kim, 2014)	84.21	93.87
HN-SA (our model)	**89.47**	**95.92**

Table 2: Accuracy (in %) of our model and other text classification models on both versions of SWBD.

development set. We trained the model by minimizing the cross-entropy loss using Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001. The learning rate was reduced by half when development set accuracy did not change over successive epochs. Model took around 30 epochs to train.

5 Experiments and Results

We compare the performance of our model (Table 2) with traditional Bag of Words (BoW), TF-IDF, and n-grams features based classifiers. We also compare against averaged Skip-Gram (Mikolov et al., 2013), Doc2Vec (Le and Mikolov, 2014), CNN (Kim, 2014), Hierarchical Attention (HN-ATT) (Yang et al., 2016) and hierarchical network (HN) models. HN it is similar to our model HN-SA but without any self attention.

Analysis: As is evident from the experiments on both the versions of SWBD, our model (HN-SA) outperforms traditional feature based topic spotting models and deep learning based document classification models. It is interesting to see that simple BoW and n-gram baselines are quite competitive and outperform some of the deep learning based document classification model. Similar observation has also been reported by Mesnil et al. (2014) for the task of sentiment analysis. The task of topic spotting is arguably more challenging than document classification. In the topic spotting task, the number of output classes (66/42 classes) is much more than those in document classification (5/6 classes), which is done mainly on the texts from customer reviews. Dialogues in SWBD have on an average 200 utterances and are much longer texts than customer reviews. Additionally, the number of dialogues available for training the model is significantly lesser than customer reviews. We further investigated the performance on SWBD2 by examining the confusion matrix of the model. Figure 2 shows the heatmap of the normalized confusion matrix of the model on SWBD2. For most of the classes the classifier is able to predict accurately. However, the model gets confused between the classes which are semantically close (w.r.t. terms used) to each other, for example, the model gets confused between pragmatically similar topics e.g. HOBBIES vs GARDENING, MOVIES vs TV PROGRAMS, RIGHT TO PRIVACY vs DRUG TESTING.

Online Setting: In an online conversational system, a topic spotting model is required to predict the topic accurately and as soon as possible during the dialog. We investigated the relationship between dialog length (in terms of number of utterances) and accuracy. This would give us an idea about how many utterances are required to reach a desirable level of accuracy. For this experiment, we varied the length of the dialogues from the test set that was available to the model for making prediction. We created sub-dialogues of length starting with $1/32$ of the dialog length and increasing it in multiples of 2, up to the full dialog. Figure 3 shows both the absolute accuracy and the accuracy relative to that on the full dialog. With just a few (3.125%) initial utterances available, the model is already 72% confident about the topic. This may be partly due to the fact that in a discussion, the first few utterances explicitly talk about the topic. However, as we have seen, since SWBD covers many different topics which are semantically close to each other but are assigned distinct classes, it is equally challenging to predict the topic with the same model. By the time the system has processed half the dialog in SWBD2 it is already within 99% accuracy of the full system. The experiment shows the possibility of using the model in an online setting where the model predicts the topic with high confidence as the conversation progresses.

6 Conclusion and Future Work

In this paper we presented a hierarchical model with self attention for topic spotting. The model outperforms the conventional topic spotting techniques as well as deep learning techniques for text classification. We empirically show that the proposed model can also be used in an online setting. We also introduced a new version of SWBD corpus: SWBD2. We hope that it will serve as the new standard for evaluating topic spotting models.

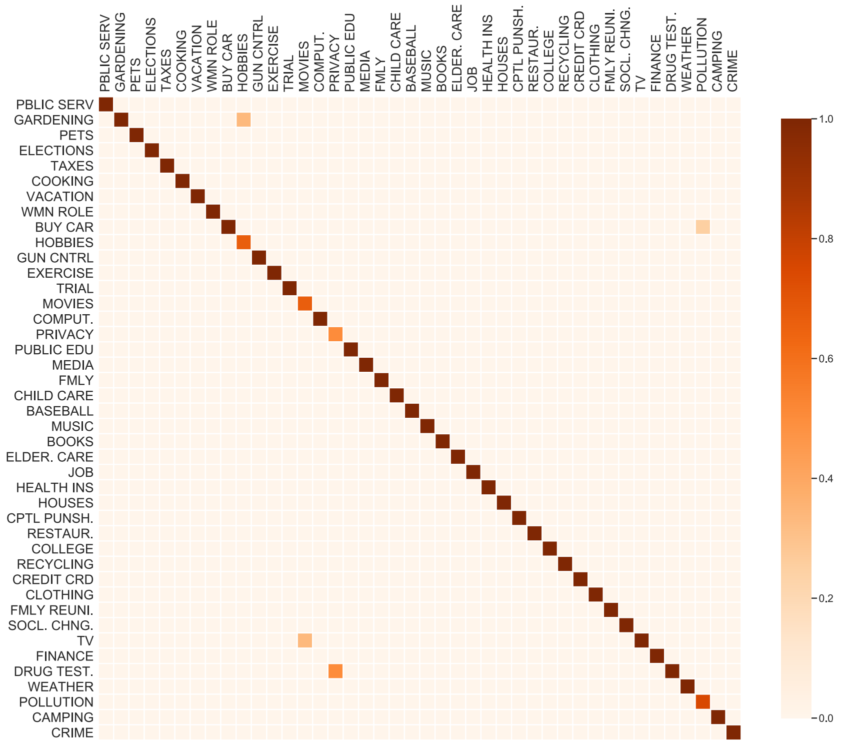


Figure 2: Normalized Confusion Matrix in form of heatmap for model predictions on SWBD2. Vertical axis is the target class.

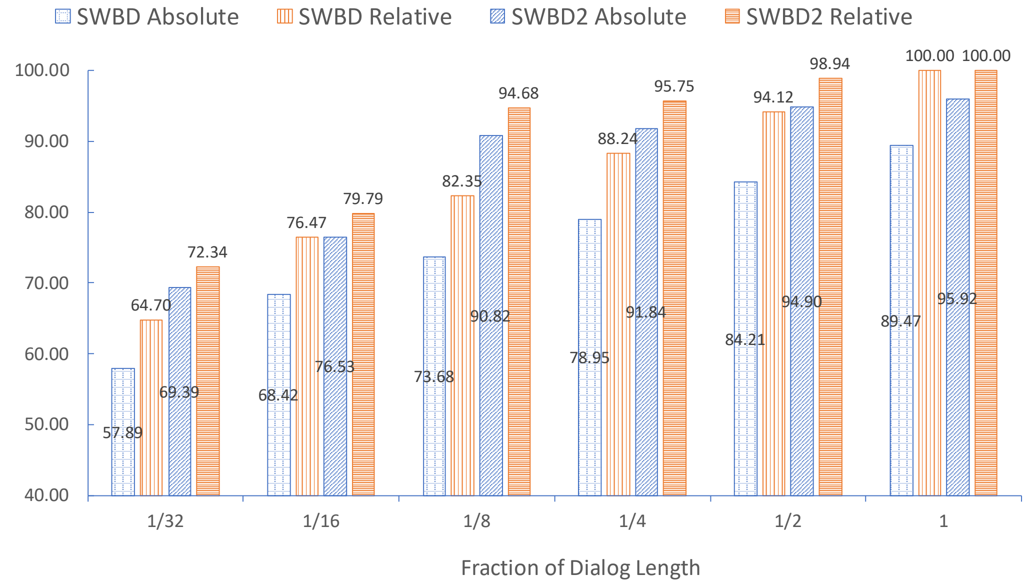


Figure 3: Effect of Dialog Length on Accuracy. Plot shows both the absolute accuracy and relative accuracy (w.r.t. full model) for different fractions of the data.

Moving forward, we would like to explore a more realistic multi-modal topic spotting system. Such a system should fuse two modalities: audio and transcribed text to make topic predictions.

Acknowledgments

We would like to thank anonymous reviewers for their insightful comments. Mubbasir Kapadia has been funded in part by NSF IIS-1703883, NSF S&AS-1723869, and DARPA SocialSim-W911NF-17-C-0098.

References

Xavier Bost, Marc El-Beze, and Renato De Mori. 2013. Multiple topic identification in telephone conversations. In *Interspeech*.

Doris M Dehn and Susanne Van Mulken. 2000. The impact of animated interface agents: a review of empirical research. *International journal of human-computer studies*, 52(1):1–22.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *ICASSP-92*.

Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. 2018. Topic-based evaluation for conversational bots. *CoRR*, abs/1801.03622.

Rakesh Gupta and Lev Ratinov. 2007. Topic spotting in dialogues using knowledge transfer. In *NIPS Workshop on Learning Problem Design*.

Timothy J Hazen. 2011. Topic identification. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 319–356.

Timothy J Hazen, Fred Richardson, and Anna Margolis. 2007. Topic identification from audio recordings using word and phone recognition lattices. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 659–664. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rukmini M Iyer and Mari Ostendorf. 1999. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on speech and audio processing*.

Kristiina Jokinen, Antti Kerminen, Mauri Kaipainen, Tommi Jauhiainen, Graham Wilcock, Markku Turunen, Jaakko Hakulinen, Jukka Kuusisto, and Krista Lagus. 2002. Adaptive dialogue systems-interaction with interact. In *Proceedings of the 3rd SIGdial workshop on Discourse and dialogue-2002,Volume 2*.

Kristiina Jokinen and Michael McTear. 2009. Spoken dialogue systems. synthesis lectures on human language technologies. *Morgan and Claypool*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Roland Kuhn, Peter Nowell, and Caroline Drouin. 1997. Approaches to phoneme-based topic spotting: An experimental comparison. In *ICASSP*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *EMNLP*.

Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. 2014. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Kary Myers, Michael Kearns, Satinder Singh, and Marilyn A Walker. 2000. A boosting approach to topic spotting on subdialogues. *Family Life*, 27(3):1.

Elmar Nöth, Stefan Harbeck, Heinrich Niemann, and Volker Warnke. 1997. A frame and segment based approach for topic spotting. In *Fifth European Conference on Speech Communication and Technology*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge University Press.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. pages 1422–1432. The Association for Computational Linguistics.

Marthinus Wilhelmus Theunissen. 2002. *Phonene-based topic spotting on the switchboard corpus*. Ph.D. thesis.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Jerry H Wright, Michael J Carey, and Eluned S Parris. 1996. Statistical models for topic identification using phoneme substrings. In *ICASSP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Top-Down Structurally-Constrained Neural Response Generation with Lexicalized Probabilistic Context-Free Grammar

Wenchao Du
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
`wenchaod@cs.cmu.edu`

Alan W Black
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
`awb@cs.cmu.edu`

Abstract

We consider neural language generation under a novel problem setting: generating the words of a sentence according to the order of their first appearance in its lexicalized PCFG parse tree, in a depth-first, left-to-right manner. Unlike previous tree-based language generation methods, our approach is both (i) top-down and (ii) explicitly generating syntactic structure at the same time. In addition, our method combines neural model with symbolic approach: word choice at each step is constrained by its predicted syntactic function. We applied our model to the task of dialog response generation, and found it significantly improves over sequence-to-sequence baseline, in terms of diversity and relevance. We also investigated the effect of lexicalization on language generation, and found that lexicalization schemes that give priority to content words have certain advantages over those focusing on dependency relations.

1 Introduction

Neural encoder-decoder architectures have shown promise and become very popular for natural language generation. Over the past few years, there has seen a surging interest in sequence-to-sequence learning for dialog response generation using neural encoder-decoder models (Vinyals and Le, 2015; Serban et al., 2017). Typically, an encoder encodes conversational context (source side) information into vector representations, and a decoder auto-regressively generates word tokens conditioned on the source vectors and previously generated words.

Two problems arise with the standard left-to-right decoding mechanism. First, no future information is available at any step of the decoding process, while the study of linguistic dependency structure shows that certain words depend on the others that come right to them. Second, preceding words define the context for following words in left-to-right, auto-regressive language models, while linguistic theories may prefer other hierarchies (e.g., adjectives modifying nouns, adverbs modifying verbs). Psycho-linguistics studies also suggest that human may first generate the abstract representation of the things to say, and then linearize them into sentences (Dell et al., 1999).

Therefore, it is appealing to consider language generation in alternative orders. This poses a greater challenge because a mechanism in extra to word generation is needed for deciding the position of each word. Some recent works adopt a syntax-free approach to address this problem. (Mehri and Sigal, 2018) proposed a middle-out decoder that starts from the middle of sentences and finishes the rest in forward and backward directions. (Mou et al., 2016) and (Li and Sun, 2018) start with one or two predicted keywords and generate the rest of sentences in a similar fashion. Others incorporate tree structures without syntactic relations and categories. (Zhou et al., 2018) canonicalizes the dependency structures of sentences into ternary trees, and generate only the words top-down. Yet another line of work aim to model the full syntactic trees. (Gū et al., 2018) generates phrase structures and part-of-speech tags along with words for machine translation. (Dyer et al., 2016) generates shift-reduce action sequences of context-free grammars in addition to words for language model and parsing. But words are still generated in left-to-right order in their approaches.

In the domain of dialog, we believe language generation can benefit from alternative orders, for the same reasons argued earlier. On the other hand, in human conversations, the structure of utterances usually correspond with dialog states (e.g., wh-noun or wh-adverb phrases are more

Proceedings of NAACL-HLT 2019, pages 3762–3771
Minneapolis, Minnesota, June 2 - June 7, 2019.

likely to be used in a request state), so modelling phrase structures can potentially help capturing discourse level information. In order to be able to generate complete syntactic trees, while be flexible about word generation order at the same time, the use of lexicalized grammar becomes a natural choice.

2 Related Work

Recent years has seen works in language model and generation through alternative orders. (Zhang et al., 2016) developed a top-down neural architecture for language model that alternates between four LSTM decoders according to given dependency relations. (Ford et al., 2018) proposed a two-stage language model, of which the first stage is a language model that generates templates, and the second stage is a translation model that fills in the blanks. Word generation order varies with the choice of words that are generated at different stages.

Language generation with tree structures has been explored more thoroughly for neural machine translation. (Eriguchi et al., 2017) and (Aharoni and Goldberg, 2017) generate CFG trees in bracketed form. (Wu et al., 2017) generates the sequence of transitions to form dependency trees. More recent works have focused on explicitly generating tree structures (Wang et al., 2018; Gū et al., 2018).

Regarding neural architectures for tree generation in the field of natural language processing, (Dong and Lapata, 2016) and (Yin and Neubig, 2017) use a single decoder with parent-feeding mechanism to generate logical forms and programming codes. (Gū et al., 2018) applied the doubly-recurrent neural networks of (Alvarez-Melis and Jaakkola, 2016) with attention mechanism to machine translation. Their model uses two decoders, of which one memorizes the ancestors, and the other remembers the siblings. (Wang et al., 2018) also uses two decoders, but one for generating words and the other for generating syntax trees.

In the domain of dialog response generation, the use of syntactic structures is under-studied. (Mou et al., 2016) and (Li and Sun, 2018) considered starting with keywords, and finish the sentences in forward and backward directions. Their models in principle are not tree-structured. The closest thing to our knowledge is by (Zhou et al., 2018). They proposed to convert dependency trees to ternary trees, but ignore the type of dependency relations. In other words, they modelled on trees of which the nodes and edges have no labels. The key difference between their approach and ours is that we generate syntax trees with labels, and word choices are also constrained by the labels.

3 Design Choices

We first consider the following three requirements when generating an L-PCFG syntax tree:

Deciding the structure of children. Several mechanisms have been proposed for deciding the structure of children of each node in the context of tree generation. One of them decide tree topology by using a sequence model to generate children one by one and predict stopping tokens (Alvarez-Melis and Jaakkola, 2016). Then there is a simpler approach that treats each combination of the labels of the children as one token, and predict such tokens when generating the parent node (Yin and Neubig, 2017). For language generation, we adopt the second approach and predict the combination of the labels of children, i.e. the rules, for two reasons: (i) the space of grammar rules is generally sparse even when its dimensionality is exponential of the number of labels, and (ii) with sequential generation of labels, as in the first approach, it is hard to enforce the labels of the children to form a valid grammar rule.

Deciding the heir of a node. Recall the definition of lexicalized PCFG: let W, N, R be the sets of lexicons, labels, and rules, where each rule is one of the following forms:

- $X(h) \to h$
- $X(h) \to Y_1(h_1) \dots Y_k(h_k)$ such that there exists i, $h_i = h$.

where $X, Y_1, \dots, Y_k \in N$, $h, h_1, \dots, h_k \in W$. We do not restrict ourselves to Chomsky Normal Form, and rules can have any number of children. The i^{th} children in the second case is called the *heir*. The key difficulty to top-down generation of lexicalized PCFG parse tree is deciding which child would be the heir. One way is to make explicit decision to select the child by adding a switch variable, at the cost of increasing the complexity of the problem. Instead, we make a change to the second case above, and simplify the problem by restricting the rules to be of the following form:

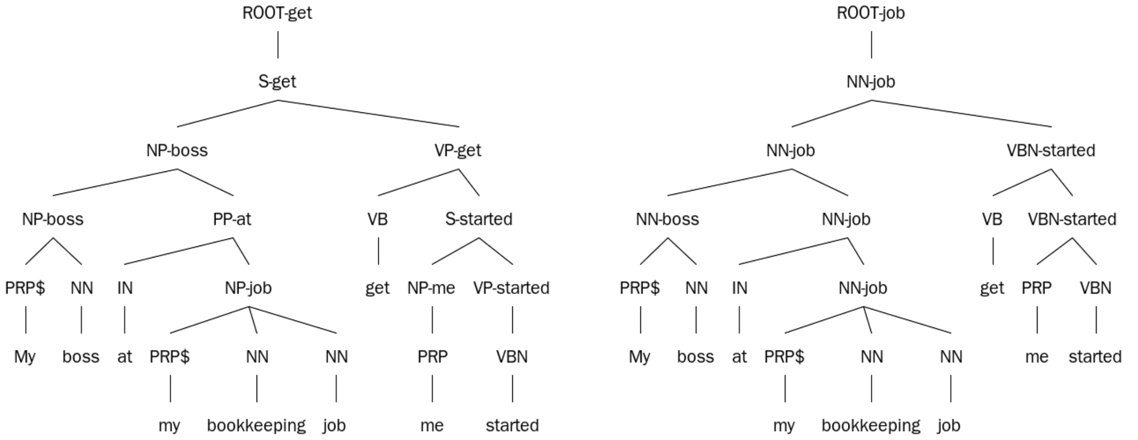


Figure 1: The first tree is the result of Stanford parser. The second one is obtained by performing content-based lexicalization on the first tree. All labels are replaced by part-of-speech tags of heirs. Unary rules NP-me → PRP and VP-started → VBN are removed.

- $X(h) \rightarrow Y_1(h_1)\dots Y_k(h_k)$ such that there exists i, $h_i = h$ and $Y_i = X$.

In other words, the heir would inherit both the lexicalization and the label of the parent (with the possible exception that the root node may produce children that are not labeled with “root”). When generating a parent node and its children, we restrict the choice of rules to those containing the label of the parent, so the heir can be inferred from the chosen rules by looking for the child that has the same label as its parent (in case there are multiple children that have the same label, we choose the rightmost one; other heuristics are possible). Note that under such restriction we end up with parse trees in which all labels are part-of-speech tags.

Sequentialization of a tree. Previous works adopt various construction orders of trees. (Zhang et al., 2016), (Zhou et al., 2018), (Alvarez-Melis and Jaakkola, 2016), and (Gū et al., 2018) generate trees through level-order traversal (breadth-first, left-to-right), whereas (Yin and Neubig, 2017) and (Wang et al., 2018) generate trees through pre-order traversal (depth-first, left-to-right). (Kuncoro et al., 2018) also experimented with bottom-up and left-corner construction orders for language model. While finding the optimal order of generating trees is beyond the scope of this work, we follow (Yin and Neubig, 2017) and (Wang et al., 2018), and generate lexicalized PCFG syntax trees through pre-order traversal.

4 Methods

4.1 Definitions

In this paper, we give the following graph-theoretic definition of L-PCFG syntax trees. Let W and N be the sets of lexicons and labels. Let $R = \bigcup_{k=1}^{\infty} N^k$ be the set of production rules. Then an L-PCFG syntax tree T is an *ordered tree* defined by the triple of vertices $v \in V \subset W \times N \times R \times \mathbb{N}$, edges $e \in E \subset V \times V$, and bijection $f : V \times \mathbb{N}^+ \rightarrow V$ such that $(v, f(v, j)) \in E$, where j range from 1 to the number of children of v. The fourth coordinate of v is the index of its heir, that is,

$$(w, n, r, i) = f((w_0, n_0, r_0, i_0), i_0) \\ \Longrightarrow w = w_0, n_0 = n$$

We say a node $v = (w, n, r, i)$ is a *leaf* if r is unary:

$$\text{v is a leaf} \iff r \in N$$

The parent of v_k is denoted by $v_{p(k)} = (w_{p(k)}, m_{p(k)}, n_{p(k)}, i_{p(k)})$.

4.2 Generation Procedure

Following previous work, we sequentialize L-PCFG parse trees and generate its content in an auto-regressive manner. When generating k^{th} node, we predict the lexicalization w_k and the rule r_k. The pre-order history available when generating k^{th} node is $n_1 \cdots n_{k-1}, w_1 \cdots w_{k-1}, r_1 \cdots r_{k-1}$, denoted by

H_k. The label "ROOT" is given at the start of generation. The label of the k^{th} node is inferred from the production rule of its parent and the order of k^{th} node among its siblings, and is used as input together with H_k. When a leaf node is reached, the program backtraces until it finds an ancestor that has unfinished child, and proceeds to the first such child.

We factor the joint probability of w_k and r_k into two component: a word model and a syntax model, as follow:

$$P(w_k, r_k \,|n_k, H_k) = \\ P(w_k \mid n_k, H_k) \cdot P(r_k \mid n_k, H_k)$$

The details of both models are given in the following sections.

4.3 Lexicalization Schemes

We parse the responses in training corpus using the lexicalized parser by (Klein and Manning, 2003) (which we call Stanford parser for the rest of this paper). We then replace the label of each node with that of their heir in a bottom-up manner. Unary rules at non-leaf nodes are removed as they become redundant given our definition of lexiclaized PCFG.

Stanford parser lexicalizes PCFG phrase structures by looking for the most likely combination of phrase structure and dependency structure. While their approach is optimized for parsing, the syntax trees lexicalized this way has a drawback for the purpose of generation. Empirically, their parser tends to lexicalize the first few nodes with auxiliary verbs or common verbs (e.g. be, must), and in some cases prefer function words over content words (e.g. in preposition phrases). We hypothesize that choosing content words over functions, or infrequent words over frequent words as lexicalization heads will help making the generation more specific and meaningful. Hence, we consider two alternative lexicalization schemes:

Content-based lexicalization. We rank words according to their part-of-speech in the sentence by the following order: nouns $>$ verbs $=$ adjectives $>$ adverbs $>$ everything else. If two words have the same rank, we give priority to the rightmost one. See Figure 1 for an example.

Frequency-based lexicalization. We ignore part-of-speech information and rank all words by their frequencies. We regard less frequent words as more important.

4.4 Encoding Tree Histories

To represent the state of a tree node by encoding its pre-order history H_k, we use 3 LSTMs to memorize the lexical and grammatical contents in H_k.

Encoding lexicalization history. We use 2 LSTMs to encode the lexicalization history, i.e. $w_1 \cdots w_{k-1}$: a *surface* decoder, L_s, which takes the lexicalization of the leaves in the history as inputs; and an *ancestor* decoder, L_a, which is given the lexicalization of the ancestors of the current node. This is another form of doubly-recurrent neural networks. Different from (Alvarez-Melis and Jaakkola, 2016), we chose to encode leaves instead of siblings. Denote the lexicalizaiton of leaves and ancestors in H_k by $\{w_{l(k)_i}\}$ and $\{w_{a(k)_i}\}$. We show that for an L-PCFG syntax tree, $\{w_{l(k)_i}\}$ and $\{w_{a(k)_i}\}$ sufficiently cover the lexical content of H_k:

Proposition. For any index set $I_k \subset \{1, 2, \dots, k-1\}$, if $w_n \in \{w_j\}_{j \in I_k}$ for all $n < k$, then $\{l(k)_i\} \bigcup \{a(k)_i\} \subset I_k$, i.e. $\{l(k)_i\}$ and $\{a(k)_i\}$ together is the minimal index set to cover $w_1 \cdots w_{k-1}$.

Encoding syntactic history. We encode the previous labels and rules using the full history with a single LSTM, L_g. At step k, the input to L_g is the concatenation of the embeddings of n_k, the rule of parent node $r_{p(k)}$, and depth of the node d. The depths of nodes deeper than 10 are rounded down to 10.

4.5 Encoding Source and Attention Mechanism

We adopt attention mechanism into our architecture for response generation. We use a one-layered LSTM to encode the dialog history, which is the concatenation of the last few utterances. The initial hidden states of L_s, L_a, and L_g is computed from the last hidden state of source encoder using 2 fully-connected layers with rectified linear activation. At time step t, the concatenation of the hidden states of L_s and L_a at step $t-1$ is used as the key for querying the source. The attention weights are the inner products of the key and the hidden states at source side, normalized by softmax function. The weighted sum of source hidden states results in the attention context, $c(k)$

4.6 Decoding

Denote the hidden states of L_s, L_a, L_g at node k as $h_s(k), h_a(k), h_g(k)$. Denote the softmax func-

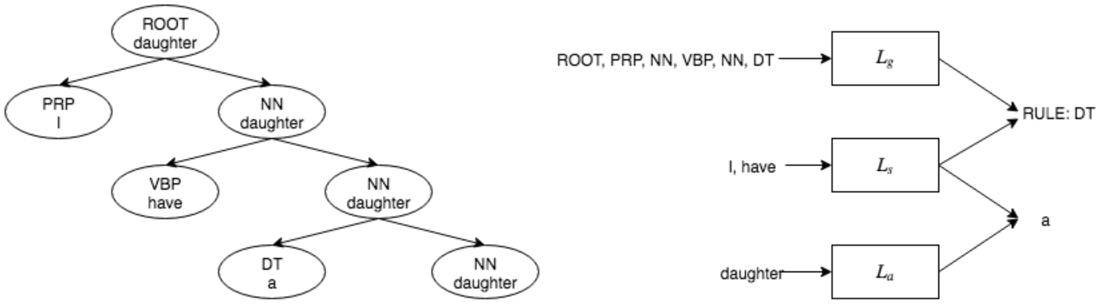


Figure 2: Demonstration of inputs and outputs at node DT. The sequence of inputs to each encoder are shown in the graph. The inputs to L_g is a sequence of labels, rules of parents, and tree depths (only labels are shown). L_s and L_a are used for predicting the word for DT. L_s and L_g are used for predicting the rule for DT. "RULE: DT" indicates DT will be a leaf node since the number of symbols is 1. In this tree, words are generated in the order: daughter - I - have - a.

tion by σ. $E_w \in \mathbb{R}^{|W| \times d_w}$ and $E_r \in \mathbb{R}^{|P| \times d_r}$ are embedding matrices for words, labels, and rules. $A_w \in \mathbb{R}^{n_w \times d_w}$ and $A_r \in \mathbb{R}^{n_p \times d_r}$ are weight matrices (n_w, n_p are the dimensions of input neurons). We use weight tying (Press and Wolf, 2017) to limit the search space for parameters.

Word prediction. To decode for w_k, we use the the hidden states of surface decoder L_s and ancestor decoder L_a. If v_k is a heir, then

$$P(w_k \mid n_k, H_k) = \begin{cases} 1 & w_k = w_{p(k)} \\ 0 & w_k \neq w_{p(k)} \end{cases}$$

Otherwise, the probability of w_k is given by:

$$P(w_k \, | n_k, H_k) = \sigma(tanh([h_s(k); h_a(k); c(k)] \, A_w) E_w^T)$$

At decoding time, we impose an additional constraint that w_k be a valid word for label n_k, to enforce grammaticality. This is estimated from the co-occurrence of w_k and n_k in the tagged training corpus. We only use those words whose frequency of co-occurrence with the given label is above a certain threshold.

Rule prediction. The probability of r_k is given by

$$P(r_k \, | n_k, H_k) = \sigma(tanh([h_s(k); h_g(k); c(k)] \, A_r) E_r^T)$$

Given the definition of L-PCFG syntax tree, we only consider rules that contain n_k at decoding time. There is one exception: at "ROOT" node, only unary rules are considered, and they do not have to contain the label "ROOT".

Hence, we train our architecture by minimizing the negative log-likelihood of words and rules:

$$\begin{aligned} & -\log P(T) = \\ & \quad -\frac{1}{|W(T)|} \sum_{\substack{k \\ v_k \neq f(v_{p(k)}, i_{p(k)})}} \log P(w_k \mid n_k, H_k) \\ & \quad -\frac{1}{|T|} \sum_k \log P(r_k \mid n_k, H_k) \end{aligned}$$

where $|W(T)|$ is the number of non-heir nodes (or the number of words in the original sentence), and $|T|$ is the number of nodes in T. Note that the log probability of each word in the sentence appears exactly once in the above equation. At test time, we conduct beam search and use the same equation to score each generation for selecting words and rules.

In our experiments, we use unlexicalized PCFG as an additional baseline. We still replace the labels of each node with their heirs' in the parse tree returned from Stanford parser, but words are generated only at leaf nodes. This baseline has syntactic structure while generating words from left to right. We use it as a test against top-down generation of words with syntax.

4.7 Training Details

All models are implemented using PyTorch. The hidden size of all LSTM encoders and decoders are 512. The size of embeddings of words, labels, rules, and tree depth are 300. We trained our models using stochastic gradient descent (SGD) with momentum and exponential learning rate decay. Dropout is applied to the input and output layer of LSTMs.

	Stand.	Depend.	Content
Nouns	8.21	7.70	6.25
Verbs	9.56	7.07	7.03
Adjectives	7.38	7.82	7.77
Adverbs	6.38	6.99	7.29
Other	5.64	6.23	6.62

Table 1: Average absolute positions of different type of words.

	Stand.	Depend.	Freq.
1	.0262	.0095	.0002
2	.0149	.0274	.0238
3	.0147	.0116	.0156
4	.0133	.0145	.0147
5	.0134	.0131	.0146

Table 2: Average frequency of first five words in different generation orders.

5 Experiments and Analysis

5.1 Data

We evaluate our model for dialog response generation on Persona dataset ((Zhang et al., 2018)). Each person is give a list of persona descriptions in simple sentences, and they are required to converse according to the given persona. We use last 3 utterances for each response as source. We prepend persona descriptions to source. We use global attention over persona descriptions to compute context vectors. During pre-processing, we truncate all trailing punctuations.

5.2 Positional Statistics

We measure how early do each type of words appear in different generation orders – standard left-to-right order, dependency-based lexicalization (as in Stanford parser), and content-based lexicalization. The earlier a word appear, the less context there is for predicting it. As shown in Table 1, content-based lexicalization can make nouns and verbs appear much earlier, while delaying function words.

To verify frequency-based lexicalization is making infrequent words appear earlier, we show the average frequencies of the first five words. The first few words are more important since they decide the context for generating the following words. In Table 2, the first two words of each parse tree under frequency-based lexicalization are much less frequent.

	Seq2seq	Ours
Standard	3.682	N/A
Dependency	4.015	3.964
Content	4.115	3.865
Frequency	4.088	3.827

Table 3: Perplexities.

5.3 Perplexity

We compare per word likelihood given different generation orders and architectures. For left-to-right sequence decoder, the non-standard generation orders are obtained by linearizing L-PCFG parse trees in pre-order traversal, and words of heirs are not repeated in the linearization. Note that our word model generates word without using rules and labels as inputs to its networks. As can be seen from Table 3, alternative word generation orders all make it harder for standard left-to-right sequence decoder to learn to predict the next word. On the other hand, using a doubly-recurrent architecture, specifically the surface decoder and the ancestor decoder, can improve perplexity scores for top-down word generation over the left-to-right decoder. While our word model with top-down word generation orders has higher perplexity scores than simple model with standard generation order, we emphasize that perplexity is not an appropriate measure for generation tasks.

5.4 Evaluation Metrics

Word Overlap Based Metrics. We use BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) scores as automatic evaluation metrics. While the reliability of such metrics has been criticized (Liu et al., 2016), there is also evidence that for task-oriented domains, these metrics correlate with human judgment to a certain extent (Sharma et al., 2017).

Word Embedding Based Metrics. We evaluate the semantic similarity between generated responses and human responses/persona by the cosine distance of their sentence embeddings. We use the word averaging approach by (Arora et al., 2016) to embed the responses, which has been demonstrated to be very good at capturing lexical level semantic similarity. The normalizing singular vector is obtained from the responses in training set.

	Human	Baseline	PCFG	L-PCFG (Dependency)	L-PCFG (Content)	L-PCFG (Frequency)
Length	10.35	7.21	9.39	8.96	**11.04**	**9.74**
BLEU	N/A	0.1926	0.2041	0.2038	**0.2093**	0.2028
ROUGE-L	N/A	**0.1639**	0.1448	0.1571	0.1565	**0.1624**
METEOR	N/A	0.0718	0.0704	0.0721	**0.0777**	0.0739
Cos. Sim. to targets	N/A	**0.0913**	0.0385	0.0824	**0.0964**	0.0883
Cos. Sim. to last utterance	0.1108	0.0880	0.0714	0.0921	**0.1113**	0.0971
Cos. Sim. to persona	0.1489	0.0206	0.0455	**0.0796**	0.0691	0.0605
Distinct uni-gram	6327	678	725	**891**	813	**874**
Distinct bi-gram	44376	2802	3368	**5306**	**5427**	3680
Distinct tri-gram	77884	4844	6061	**10348**	**10787**	6441
Inertia	11771	4385	5027	**6319**	3756	3959
BLEU to training set	0.4334	0.8471	**0.5320**	0.5874	**0.5402**	0.6331
ROUGE to training set	0.4728	0.8970	**0.5837**	0.6701	0.6226	0.7175

Table 4: Evaluation results on Persona dataset.

Context	Human	Seq2seq	L-PCFG
i am great . i just got back from the club	this is my favorite time of the year season wise	that is cool what do you do for a living	awesome ! i am getting ready to go to the club
sure i like tv , what do you watch ?	really anything , what about you ?	i watch a lot of tv	i watch lot of tv movies . i like to watch tv
oh . tell me something about yourself .	well i do not like heights very much and i love animals . what about you ?	i am an accountant . what do you do	i am trying to learn how to work with animals
i live in texas . i love riding my bike here .	are you a christian ? i am jewish	i have never been to the west coast . do you have any hobbies	i do too ! i wish i was there so i can do that for school

Table 5: Examples of generated responses.

Novelty and diversity. We measure word overlapping between generated responses and the responses in training set using BLEU and ROUGE as a proxy for novelty. The responses in training set with most common words with generated responses are used as references. For diversity, we count the number of distinct n-grams. In addition, we perform a k-means clustering on the sentence embeddings of responses into 10 clusters, and measure average squared Euclidean distance between members of each cluster (Inertia). The larger the number, the harder it is to separate embeddings into 10 clusters, thus the greater the diversity.

5.5 Main Results

5.5.1 Quantitative Analysis

Evaluation results are shown in Table 4. For metrics that are not measured using ground truth response as reference, we consider the closer to the number for human responses the better. We first look at measures for overall generation quality. We can see modelling syntactic structures is capable of generating longer responses. BLEU scores are positively correlated with lengths. While syntactic models do better on BLEU, and slightly better on METEOR than sequence-to-sequence baseline, they are generally not on par with the baseline in terms of ROUGE-L, except for frequency-based lexicalization. Among grammar models, lexicalized grammars out-performed unlexicalized grammar.

Relevance is measured using cosine similarities with the previous utterance and persona. Syntactic models with lexicalized grammar beat the baseline in terms of relevance. Furthermore, content-based lexicalization is much more on topic with the last source utterance than dependency-based

	$L_g + L_s$	$L_g + L_s + L_a$	L_g
Lengths	11.04	7.95	8.46
Distinct uni-gram	813	383	376
Distinct bi-gram	5427	1763	1775
Distinct tri-gram	10787	3061	3522
Inertia	3756	2049	2346
BLEU on training	0.5402	0.8056	0.6612
ROUGE on training	0.6226	0.8543	0.7417

Table 6: Ablation studies.

and frequence-based lexicalization. Dependency-based lexicalization is best at being adherent to personas than the other two lexicalization schemes.

All syntactic models generate more novel responses than sequence-to-sequence baseline, as reflected in the last two rows in 4. This is consistent with the observation that sequence-to-sequence model exhibits retrieval-like behaviour, selecting what is most common in the training corpus. Syntactic models also have larger vocabularies. As for cluster analysis, unlexicalized grammar model and dependency-based lexicalized grammar model have better diversity than sequence-to-sequence model; content-based and frequency-based lexicalization have slightly smaller inertia than the baseline.

5.5.2 Qualitative Analysis

We present a few examples generated by sequence-to-sequence baseline and L-PCFG model. There is a clear difference of how left-to-right decoder and L-PCFG tree decoder do conjunctions. Most of the time, standard LSTM decoder combine sentences with periods, while tree decoders learn to use conjunction words, or even clauses.

We also performed an error analysis on the generated responses by L-PCFG, and in Table 7 we selected the most peculiar and representative. These examples are all syntactically fine, but they do not follow the convention of the language or common sense. The first example contains the most common errors in the responses generated by L-PCFG: misuse of prepositions and determiners. It can be fixed by replacing “as a” with “for”. The error of other two examples have even less to do with syntax. The second one misuses the verb “be”, which is probably caused by the high frequency of the word in the corpus. The error of the third example is beyond surface level. Note that phrases such as “cooking as a dinner” and “be a dog” never appear in the corpus. It is clear that L-PCFG models are learning to make compositions of words and phrases, unlike standard LSTM decoder, which seems to only memorize word combinations.

i am doing well . just finished cooking as a dinner
i am sure it is nice . i am going to be a dog
i like to ride my horses on my bike

Table 7: Some peculiar examples generated by L-PCFG models.

5.6 Ablation Studies

We perform ablation studies on our architecture, with content-based lexicalization. Specifically, we consider two alternative ways of making rule prediciton. The first one takes only the hidden state of L_g for predicting rules, hence making the prediction of rules entirely independent from words. The second one takes both the hidden states of L_s and L_a, together with L_g, for predicting rules, in which way future lexical information is used for the construction of syntax trees.

For rule prediction using the hidden states of both surface decoder and ancestor decoder, we noticed a significant drop in diversity in generated responses. The model over-predicts “what do you do for a living” for than 50% of the time, and the lengths of responses tend to be shorter.

For the other choice in which rule prediction is independent from words, the results are closer to the original model, but there are still some decreases in lengths and lexical diversity. Upon manual inspection, we found that this model behaved more like sequence-to-sequence model. There are less compound sentences, and more conjunctions of simple sentences by end punctuation marks.

The proportion of simple sentences are also larger.

6 Conclusion

We consider the problem of generating natural language in alternative orders and with syntactic tree structures, with the use of lexicalized grammar. By incorporating syntactic structures, our models are capable of generating longer sentences. By changing lexicalizaion schemes and making content words appear earlier in generation process, our models are able to make word choices that are more relevant to source. Furthermore, incorporating syntax facilitates response generation with richer vocabularies and more complex structures. On the other hand, as shown in our error analysis, there is still room for improvement on discourse and pragmatics level.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation (Award # 1722822). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation, and no official endorsement should be inferred.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 132–140.

David Alvarez-Melis and Tommi S Jaakkola. 2016. Tree-structured decoding with doubly-recurrent neural networks.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Gary S Dell, Franklin Chang, and Zenzi M Griffin. 1999. Connectionist models of language production: Lexical access and grammatical encoding. *Cognitive Science*, 23(4):517–542.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 72–78.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. 2018. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2942–2946.

Jetic Gū, Hassan S Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 401–413.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Jingyuan Li and Xiao Sun. 2018. A syntactically constrained bidirectional-asynchronous approach for emotional conversation generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 678–683.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132. Association for Computational Linguistics.

Shikib Mehri and Leonid Sigal. 2018. Middle-out decoding. In *Advances in Neural Information Processing Systems*, pages 5519–5530.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 698–707.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 440–450.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320.

Ganbin Zhou, Ping Luo, Rongyu Cao, Yijun Xiao, Fen Lin, Bo Chen, and Qing He. 2018. Tree-structured neural machine for linguistics-aware sentence generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5722–5729. AAAI Press.

A Appendices

A.1 Proof sketch for *Proposition* in Section 4.4

We prove by contradiction. Suppose there is a node n_k, whose lexicalization is not in the leaves before node k, nor in the ancestors of node n_k. There must be a leaf l inheriting the lexicalization of node n_k. Since n_k is not an ancestor of node k, l must be a leaf before k, so we have a contradiction.

What do Entity-Centric Models Learn? Insights from Entity Linking in Multi-Party Dialogue

Laura Aina* **Carina Silberer*** **Ionut-Teodor Sorodoc***
Matthijs Westera* **Gemma Boleda**

Universitat Pompeu Fabra
Barcelona, Spain
`{firstname.lastname}@upf.edu`

Abstract

Humans use language to refer to entities in the external world. Motivated by this, in recent years several models that incorporate a bias towards learning entity representations have been proposed. Such entity-centric models have shown empirical success, but we still know little about why.

In this paper we analyze the behavior of two recently proposed entity-centric models in a referential task, Entity Linking in Multi-party Dialogue (SemEval 2018 Task 4). We show that these models outperform the state of the art on this task, and that they do better on lower frequency entities than a counterpart model that is not entity-centric, with the same model size. We argue that making models entity-centric naturally fosters good architectural decisions. However, we also show that these models do not really build entity representations and that they make poor use of linguistic context. These negative results underscore the need for model analysis, to test whether the motivations for particular architectures are borne out in how models behave when deployed.

1 Introduction

Modeling reference to entities is arguably crucial for language understanding, as humans use language to talk about things in the world. A hypothesis in recent work on referential tasks such as co-reference resolution and entity linking (Haghighi and Klein, 2010; Clark and Manning, 2016; Henaff et al., 2017; Aina et al., 2018; Clark et al., 2018) is that encouraging models to learn and use entity representations will help them better carry out referential tasks. To illustrate, creating an entity representation with the relevant information upon reading *a woman* should make it easier to resolve a pronoun mention like *she*.[1] In the mentioned work, several models have been proposed that incorporate an explicit bias towards entity representations. Such **entity-centric** models have shown empirical success, but we still know little about what it is that they effectively learn to model. In this analysis paper, we adapt two previous entity-centric models (Henaff et al., 2017; Aina et al., 2018) for a recently proposed referential task and show that, despite their strengths, they are still very far from modeling entities.[2]

JOEY TRIBBIANI (183):
"*. . . see Ross, because I think you love her .*"
335 183 335 306

Figure 1: Character identification: example.

The task is character identification on multi-party dialogue as posed in SemEval 2018 Task 4 (Choi and Chen, 2018).[3] Models are given dialogues from the TV show *Friends* and asked to link entity mentions (nominal expressions like *I*, *she* or *the woman*) to the characters to which they refer in each case. Figure 1 shows an example, where the mentions *Ross* and *you* are linked to entity 335, mention *I* to entity 183, etc. Since the TV series revolves around a set of entities that recur over many scenes and episodes, it is a good benchmark to analyze whether entity-centric models learn and use entity representations for referential tasks.

Our contributions are three-fold: First, we adapt two previous entity-centric models and show that they do better on lower frequency entities

---

*denotes equal contribution.

[1] Note the analogy with traditional models in formal linguistics like Discourse Representation Theory (Kamp and Reyle, 2013).

[2] Source code for our model, the training procedure and the new dataset is published on `https://github.com/amore-upf/analysis-entity-centric-nns`.

[3] `https://competitions.codalab.org/competitions/17310`.

Proceedings of NAACL-HLT 2019, pages 3772–3783
Minneapolis, Minnesota, June 2 - June 7, 2019.

(a significant challenge for current data-hungry models) than a counterpart model that is not entity-centric, with the same model size. Second, through analysis we provide insights into how they achieve these improvements, and argue that making models entity-centric fosters architectural decisions that result in good inductive biases. Third, we create a dataset and task to evaluate the models' ability to encode entity information such as gender, and show that models fail at it. More generally, our paper underscores the need for the analysis of model behavior, not only through ablation studies, but also through the targeted probing of model representations (Linzen et al., 2016; Conneau et al., 2018).

2 Related Work

Modeling. Various memory architectures have been proposed that are not specifically for entity-centric models, but could in principle be employed in them (Graves et al., 2014; Sukhbaatar et al., 2015; Joulin and Mikolov, 2015; Bansal et al., 2017). The two models we base our results on (Henaff et al., 2017; Aina et al., 2018) were explicitly motivated as entity-centric. We show that our adaptations yield good results and provide a closer analysis of their behavior.

Tasks. The task of entity linking has been formalized as resolving entity mentions to referential entity entries in a knowledge repository, mostly Wikipedia (Bunescu and Paşca, 2006; Mihalcea and Csomai, 2007 and much subsequent work; for recent approaches see Francis-Landau et al., 2016; Chen et al., 2018). In the present entity linking task, only a list of entities is given, without associated encyclopedic entries, and information about the entities needs to be acquired from scratch through the task; note the analogy to how a human audience might get familiar with the TV show characters by watching it. Moreover, it addresses multi-party dialogue (as opposed to, typically, narrative text), where speaker information is crucial. A task closely related to entity linking is *coreference resolution*, i.e., predicting which portions of a text refer to the same entity (e.g., *Marie Curie* and *the scientist*). This typically requires clustering mentions that refer to the same entity (Pradhan et al., 2011). Mention clusters essentially correspond to entities, and recent work on coreference and language modeling has started exploiting an explicit notion of entity (Haghighi and Klein, 2010; Clark and Manning, 2016; Yang et al., 2017). Previous work both on entity linking and on coreference resolution (cited above, as well as Wiseman et al., 2016) often presents more complex models that incorporate e.g. hand-engineered features. In contrast, we keep our underlying model basic since we want to systematically analyze how certain architectural decisions affect performance. For the same reason we deviate from previous work to entity linking that uses a specialized coreference resolution module (e.g., Chen et al., 2017).

Analysis of Neural Network Models. Our work joins a recent strand in NLP that systematically analyzes what different neural network models learn about language (Linzen et al., 2016; Kádár et al., 2017; Conneau et al., 2018; Gulordava et al., 2018b; Nematzadeh et al., 2018, a.o.). This work, like ours, has yielded both positive and negative results: There is evidence that they learn complex linguistic phenomena of morphological and syntactic nature, like long distance agreement (Gulordava et al., 2018b; Giulianelli et al., 2018), but less evidence that they learn how language relates to situations; for instance, Nematzadeh et al. (2018) show that memory-augmented neural models fail on tasks that require keeping track of inconsistent states of the world.

3 Models

We approach character identification as a classification task, and compare a baseline LSTM (Hochreiter and Schmidhuber, 1997) with two models that enrich the LSTM with a memory module designed to learn and use entity representations. LSTMs are the workhorse for text processing, and thus a good baseline to assess the contribution of this module. The LSTM processes text of dialogue scenes one token at a time, and the output is a probability distribution over the entities (the set of entity IDs are given).

3.1 Baseline: BILSTM

The BILSTM model is depicted in Figure 2. It is a standard bidirectional LSTM (Graves et al., 2005), with the difference with most uses of LSTMs in NLP that we incorporate speaker information in addition to the linguistic content of utterances.

The model is given chunks of dialogue (see Appendix for hyperparameter settings such as the chunk size). At each time step i, one-hot vectors for token $\mathbf{t}_i$ and speaker entities $\mathbf{s}_i$ are embedded

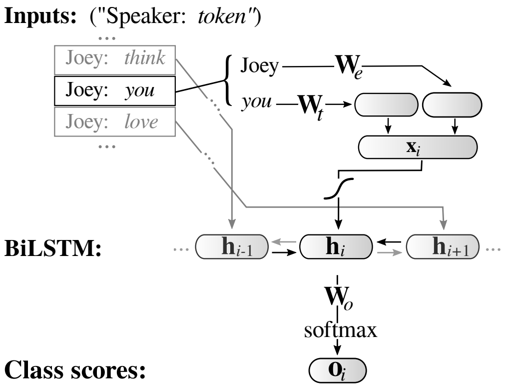


Figure 2: BILSTM applied to "...think you love..." as spoken by Joey (from Figure 1), outputting class scores for mention "you" (bias b_o not depicted).

via two distinct matrices $\mathbf{W}_t$ and $\mathbf{W}_e$ and concatenated to form a vector $\mathbf{x}_i$ (Eq. 1, where $\|$ denotes concatenation; note that in case of multiple simultaneous speakers S_i their embeddings are summed).

$$\mathbf{x}_i = \mathbf{W}_t\, \mathbf{t}_i \,\|\, \sum_{\mathbf{s} \in S_i} \mathbf{W}_e\, \mathbf{s} \tag{1}$$

The vector $\mathbf{x}_i$ is fed through the nonlinear activation function tanh and input to a bidirectional LSTM. The hidden state $\overrightarrow{\mathbf{h}_i}$ of a *uni*directional LSTM for the i^{th} input is recursively defined as a combination of that input with the LSTM's previous hidden state $\overrightarrow{\mathbf{h}}_{i-1}$. For a *bi*directional LSTM, the hidden state $\mathbf{h}_i$ is the concatenation of the hidden states $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$ of two unidirectional LSTMs which process the data in opposite directions (Eqs. 2-4).

$$\overrightarrow{\mathbf{h}_i} = \mathrm{LSTM}(\tanh(\mathbf{x}_i), \overrightarrow{\mathbf{h}}_{i-1}) \tag{2}$$

$$\overleftarrow{\mathbf{h}_i} = \mathrm{LSTM}(\tanh(\mathbf{x}_i), \overleftarrow{\mathbf{h}}_{i+1}) \tag{3}$$

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} \,\|\, \overleftarrow{\mathbf{h}_i} \tag{4}$$

For every entity mention $\mathbf{t}_i$ (i.e., every token[4] that is tagged as referring to an entity), we obtain a distribution over all entities, $\mathbf{o}_i \in [0,1]^{1 \times N}$, by applying a linear transformation to its hidden state $\mathbf{h}_i$ (Eq. 5), and feeding the resulting $\mathbf{g}_i$ to a softmax classifier (Eq. 6).

$$\mathbf{g}_i = \mathbf{W}_o\, \mathbf{h}_i + \mathbf{b}_o \tag{5}$$

$$\mathbf{o}_i = \mathrm{softmax}(\mathbf{g}_i) \tag{6}$$

Eq. 5 is where the other models will diverge.

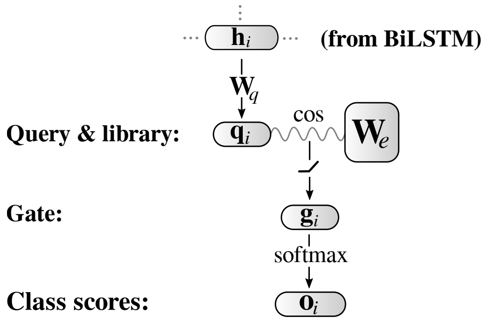


Figure 3: ENTLIB; everything before $\mathrm{h_i}$, omitted here, is the same as in Figure 2.

3.2 ENTLIB (Static Memory)

The ENTLIB model (Figure 3) is an adaptation of our previous work in Aina et al. (2018), which was the winner of the SemEval 2018 Task 4 competition. This model adds a simple memory module that is expected to represent entities because its vectors are tied to the output classes (accordingly, Aina et al., 2018, call this module *entity library*). We call this memory 'static', since it is updated only during training, after which it remains fixed.

Where BILSTM maps the hidden state $\mathbf{h}_i$ to class scores $\mathbf{o}_i$ with a single transformation (plus softmax), ENTLIB instead takes two steps: It first transforms $\mathbf{h}_i$ into a 'query' vector $\mathbf{q}_i$ (Eq. 7) that it will then use to query the entity library. As we will see, this mechanism helps dividing the labor between representing the context (hidden layer) and doing the prediction task (query layer).

$$\mathbf{q}_i = \mathbf{W}_q\, \mathbf{h}_i + \mathbf{b_q} \tag{7}$$

A weight matrix $\mathbf{W}_e$ is used as the entity library, which is the same as the speaker embedding in Eq. 1: the query vector $\mathbf{q}_i \in \mathbb{R}^{1 \times k}$ is compared to each vector in $\mathbf{W}_e$ (cosine), and a *gate* vector $\mathbf{g}_i$ is obtained by applying the ReLU function to the cosine similarity scores (Eq. 8).[5] Thus, the query extracted from the LSTM's hidden state is used as a soft pointer over the model's representation of the entities.

$$\mathbf{g}_i = \mathrm{ReLU}(\cos(\mathbf{W}_e, \mathbf{q}_i)) \tag{8}$$

As before, a softmax over $\mathbf{g}_i$ then yields the distribution over entities (Eq. 6). So, in the ENTLIB

---

[4] For multi-word mentions this is done only for the last token in the mention.

[5] In Aina et al. (2018), the gate did not include the ReLU nonlinear activation function. Adding it improved results.

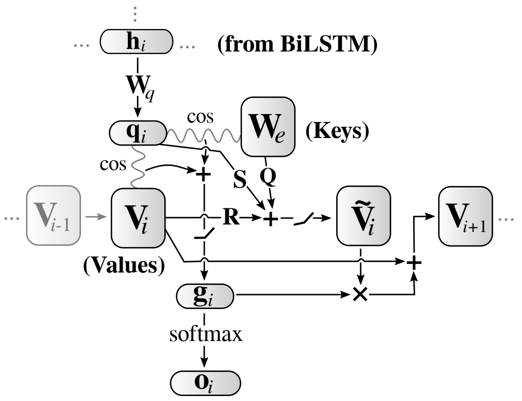


Figure 4: ENTNET; everything before $\mathrm{h_i}$, omitted here, is the same as in Figure 2.

model Eqs. 7 and 8 together take the place of Eq. 5 in the BILSTM model.

Our implementation differs from Aina et al. (2018) in one important point that we will show to be relevant to model less frequent entities (training also differs, see Section 4): The original model did not do parameter sharing between speakers and referents, but used two distinct weight matrices.

Note that the contents of the entity library in ENTLIB do not change during forward propagation of activations, but only during backpropagation of errors, i.e., during training, when the weights of $\mathbf{W}_e$ are updated. If anything, they will encode permanent properties of entities, not properties that change within a scene or between scenes or episodes, which should be useful for reference. The next model attempts to overcome this limitation.

3.3 ENTNET (Dynamic Memory)

ENTNET is an adaptation of *Recurrent Entity Networks* (Henaff et al., 2017, Figure 4) to the task. Instead of representing each entity by a single vector, as in ENTLIB, here each entity is represented jointly by a context-invariant or 'static' *key* and a context-dependent or 'dynamic' *value*. For the keys the entity embedding $\mathbf{W}_e$ is used, just like the entity library of ENTLIB. But the values $\mathbf{V}_i$ can be dynamically updated throughout a scene.

As before, an entity query $\mathbf{q}_i$ is first obtained from the BILSTM (Eq. 7). Then, ENTNET computes gate values $\mathbf{g}_i$ by estimating the query's similarity to both keys and values, as in Eq. 9 (replacing Eq. 8 of ENTLIB).[6] Output scores $\mathbf{o}_i$ are computed as in the previous models (Eq. 6).

$$\mathbf{g}_i = \mathrm{ReLU}(\cos(\mathbf{W}_e, \mathbf{q}_i) + \cos(\mathbf{V}_i, \mathbf{q}_i)) \quad (9)$$

The values $\mathbf{V}_i$ are initialized at the start of every scene ($i = 0$) as being identical to the keys ($\mathbf{V}_0 = \mathbf{W}_e$). After processing the i^{th} token, new information can be added to the values. Eq. 10 computes this new information $\tilde{\mathbf{V}}_{i,j}$, for the j^{th} entity, where $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{S}$ are learned linear transformations and PReLU denotes the parameterized rectified linear unit (He et al., 2015):

$$\tilde{\mathbf{V}}_{i,j} = \mathrm{PReLU}(\mathbf{Q}\mathbf{W}_{ej} + \mathbf{R}\mathbf{V}_{i,j} + \mathbf{S}\mathbf{q}_i) \quad (10)$$

This information $\tilde{\mathbf{V}}_{i,j}$, multiplied by the respective gate $\mathbf{g}_{i,j}$, is added to the values to be used when processing the next ($i + 1^{\text{th}}$) token (Eq. 11), and the result is normalized (Eq. 12):

$$\mathbf{V}_{i+1,j} = \mathbf{V}_j + \mathbf{g}_{i,j} * \tilde{\mathbf{V}}_{i,j} \quad (11)$$

$$\mathbf{V}_{i+1,j} = \frac{\mathbf{V}_{i+1,j}}{\|\mathbf{V}_{i+1,j}\|} \quad (12)$$

Our adaptation of the Recurrent Entity Network involves two changes. First, we use a biLSTM to process the linguistic utterance, while Henaff et al. (2017) used a simple multiplicative mask (we have natural dialogue, while their main evaluation was on bAbI, a synthetic dataset). Second, in the original model the gates were used to retrieve and output information about the query, whereas we use them directly as output scores because our task is referential. This also allows us to tie the keys to the characters of the Friends series as in the previous model, and thus have them represent entities (in the original model, the keys represented entity types, not instances).

4 Character Identification

The training and test data for the task span the first two seasons of *Friends*, divided into scenes and episodes, which were in turn divided into utterances (and tokens) annotated with speaker identity.[7] The set of all possible entities to refer to is given, as well as the set of mentions to resolve. Only the dialogues and speaker information are available (e.g., no video or descriptive text). Indeed,

---

[6] Two small changes with respect to the original model (motivated by empirical results in the hyperparameter search) are that we compute the gate using cosine similarity instead of dot product, and the obtained similarities are fed through a ReLU nonlinearity instead of sigmoid.

[7] The dataset also includes automatic linguistic annotations, e.g., PoS tags, which our models do not use.

models	#par	all (78) F$_1$	all (78) Acc	main (7) F$_1$	main (7) Acc
SemEv-1st	-	41.1	74.7	79.4	77.2
SemEv-2nd	-	13.5	68.6	83.4	82.1
biLSTM	3.4M	34.4	74.6	**85.0**	83.5
EntLib	3.3M	49.6*	**77.6***	84.9	**84.2**
EntNet	3.4M	**52.5***	77.5*	84.8	83.9

Table 1: Model parameters and results on the character identification task. First block: top systems at SemEval 2018. Results in the second block marked with * are statistically significantly better than biLSTM at $p < 0.001$ (approximate randomization tests, Noreen, 1989).

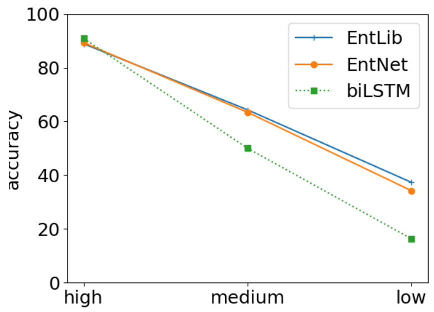


Figure 5: Accuracy on entities with high (>1000), medium (20–1000), and low (<20) frequency.

one of the most interesting aspects of the SemEval data is the fact that it is dialogue (even if scripted), which allows us to explore the role of speaker information, one of the aspects of the extralinguistic context of utterance that is crucial for reference. We additionally used the publicly available 300-dimensional word vectors that were pre-trained on a Google News corpus with the word2vec Skip-gram model (Mikolov et al., 2013a) to represent the input tokens. Entity (speaker/referent) embeddings were randomly initialized.

We train the models with backpropagation, using the standard negative log-likelihood loss function. For each of the three model architectures we performed a random search (> 1500 models) over the hyperparameters using cross-validation (see Appendix for details), and report the results of the best settings after retraining without cross-validation. The findings we report are representative of the model populations.

Results. We follow the evaluation defined in the SemEval task. Metrics are macro-average F$_1$-score (which computes the F$_1$-score for each entity separately and then averages these over all entities) and accuracy, in two conditions: *All entities*, with 78 classes (77 for entities that are mentioned in both training and test set of the SemEval Task, and one grouping all others), and *main entities*, with 7 classes (6 for the main characters and one for all the others). Macro-average F$_1$-score on all entities, the most stringent, was the criterion to define the leaderboard.

Table 1 gives our results in the two evaluations, comparing the models described in Section 3 to the best performing models in the SemEval 2018 Task 4 competition (Aina et al., 2018; Park et al., 2018). Recall that our goal in this paper is not to optimize performance, but to understand model behavior; however, results show that these models are worth analyzing, as that they outperform the state of the art. All models perform on a par on main entities, but entity-centric models outperform biLSTM by a substantial margin when all characters are to be predicted (the difference between EntLib and EntNet is not significant).

The architectures of EntLib and EntNet help with lower frequency characters, while not hurting performance on main characters. Indeed, Figure 5 shows that the accuracy of biLSTM rapidly deteriorates for less frequent entities, whereas EntLib and EntNet degrade more gracefully. Deep learning approaches are data-hungry, and entity mentions follow the Zipfian distribution typical of language, with very few high frequency and many lower-frequency items, such that this is a welcome result. Moreover, these improvements do not come at the cost of model complexity in terms of number of parameters, since all models have roughly the same number of parameters ($3.3 - 3.4$ million).[8]

Given these results and the motivations for the model architectures, it would be tempting to conclude that encouraging models to learn and use entity representations helps in this referential task. However, a closer look at the models' behavior reveals a much more nuanced picture.

Figure 6 suggests that: (1) models are quite good at using speaker information, as the best performance is for first person pronouns and determiners (*I*, *my*, etc.); (2) instead, models do not seem to be very good at handling other contextual information or entity-specific properties, as the worst

---

[8] See Appendix for a computation of the models' parameters.

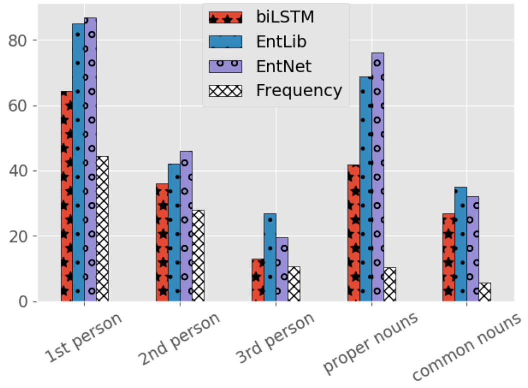


Figure 6: F_1-score (*all entities* condition) of the three models, per mention type, and token frequency of each mention type.

performance is for third person mentions and common nouns, which require both;[9] (3) EntLib and EntNet behave quite similarly, with performance boosts in (1) and smaller but consistent improvements in (2). Our analyses in the next two sections confirm this picture and relate it to the models' architectures.

5 Analysis: Architecture

We examine how the entity-centric architectures improve over the biLSTM baseline on the reference task, then move to entity representations (Section 6).

Shared speaker/referent representation. We found that an important advantage of the entity-centric models, in particular for handling low-frequency entities, lies in the integrated representations they enable of entities both in their role of speakers and in their role of referents. This explains the boost in first person pronoun and proper noun mentions, as follows.

Recall that the integrated representation is achieved by parameter sharing, using the same weight matrix $\mathbf{W}_e$ as speaker embedding and as entity library/keys. This enables entity-centric models to learn the linguistic rule "a first person pronoun (*I, me*, etc.) refers to the speaker" regardless of whether they have a meaningful representation of this particular entity: It is enough that speaker representations are distinct, and they are because they have been randomly initialized. In contrast, the simple biLSTM baseline needs to independently learn the mapping between speaker embedding and output entities, and so it can only learn to resolve even first-person pronouns for entities for which it has enough data.

model type	main	all
biLSTM	0.39	0.02
EntLib	0.82	0.13
EntNet	0.92	0.16
#pairs	21	22155

Table 2: RSA correlation between speaker/referent embeddings W_e and token embeddings W_t of the entities' names, for main entities vs. all entities (right)

For proper nouns (character names), entity-centric models learn to align the token embeddings with the entity representations (identical to the speaker embeddings). We show this by using Representation Similarity Analysis (RSA) (Kriegeskorte et al., 2008), which measures how topologically similar two different spaces are as the Spearman correlation between the pair-wise similarities of points in each space (this is necessary because entities and tokens are in different spaces). For instance, if the two spaces are topologically similar, the relationship of entities 183 and 335 in the entity library will be analogous to the relationship between the names *Joey* and *Ross* in the token space. Table 2 shows the topological similarities between the two spaces, for the different model types.[10] This reveals that in entity-centric models the space of speaker/referent embeddings is topologically very similar to the space of token embeddings restricted to the entities' names, and more so than in the biLSTM baseline. We hypothesize that entity-centric models can do the alignment better because referent (and hence speaker) embeddings are closer to the error signal, and thus backpropagation is more effective (this again helps with lower-frequency entities).

Further analysis revealed that in entity-centric models the beneficial effect of weight sharing between the speaker embedding and the entity representations (both $\mathbf{W}_e$) is actually restricted to first-person pronouns. For other expressions, having

---

[9] 1st person: *I*, *me*, *my*, *myself*, *mine*; 2nd person: *you*, *your*, *yourself*, *yours*; 3rd person: *she*, *her*, *herself*, *hers*, *he*, *him*, *himself*, *his*, *it*, *itself*, *its*.

[10] As an entity's name we here take the proper noun that is most frequently used to refer to the entity in the training data. Note that for the *all entities* condition the absolute values are lower, but the space is much larger (over 22K pairs). Also note that this is an instance of slow learning; models are not encoding the fact that a proper noun like *Rachel* can refer to different people.

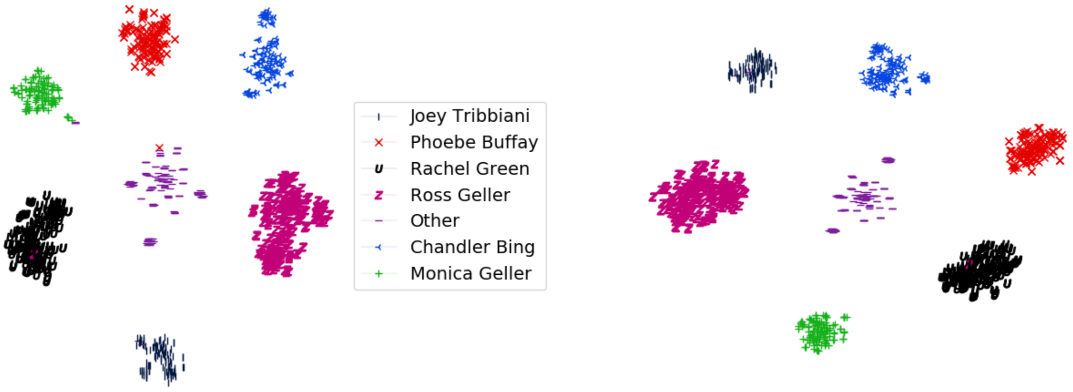


Figure 7: ENTLIB, 2D TSNE projections of the activations for first-person mentions in the test set, colored by the entity referred to. The mentions cluster into entities already in the hidden layer h_i (left graph; query layer q_i shown in the right graph). Best viewed in color.

Figure 8: ENTLIB, 2D TSNE projections of the activations for mentions in the test set (excluding first person mentions), colored by the entity referred to. While there is already some structure in the hidden layer h_i (left graph), the mentions cluster into entities much more clearly in the query q_i (right graph). Best viewed in color.

two distinct matrices yielded almost the same performance as having one (but still higher than the BILSTM, thanks to the other architectural advantage that we discuss below).

In the case of first-person pronouns, the speaker embedding given as input corresponds to the target entity. This information is already accessible in the hidden state of the LSTM. Therefore, mentions cluster into entities already at the hidden layer h_i, with no real difference with the query layer q_i (see Figure 7).

Advantage of query layer. The entity querying mechanism described above entails having an extra transformation after the hidden layer, with the query layer $\mathbf{q}$. Part of the improved performance of entity-centric models, compared to the BILSTM baseline, is due not to their bias towards 'entity representations' per se, but due to the presence of this extra layer. Recall that the BILSTM baseline maps the LSTM's hidden state $\mathbf{h}_i$ to output scores $\mathbf{o}_i$ with a single transformation. Gulordava et al. (2018a) observe in the context of Language Modeling that this creates a tension between two conflicting requirements for the LSTM: keeping track of contextual information across time steps, and encoding information useful for prediction in the current timestep. The intermediate query layer $\mathbf{q}$ in entity-centric models alleviates this tension. This explains the improvements in context-dependent mentions like common nouns or second and third pronouns.

We show this effect in two ways. First, we compare the average mean similarity s of mention pairs $T_e = \{(t_k, t_{k'}) | \; t_k \rightarrow e \wedge k \neq k'\}$ referring to the same entity e in the hidden layer (Eq. 13) and the

biLSTM	EntLib		EntNet	
$\mathbf{h}_i$	$\mathbf{h}_i$	$\mathbf{q}_i$	$\mathbf{h}_i$	$\mathbf{q}_i$
0.34	0.24	0.48	0.27	0.60

Table 3: Average cosine similarity of mentions with the same referent.

query layer.[11]

$$s = \frac{1}{|E|}\sum_{e\in E}\frac{1}{|T_e|}\sum_{(t_k,t_{k'})\in T_e}\cos(h_{t_k},h_{t_{k'}}) \quad (13)$$

Table 3 shows that, in entity-centric models, this similarity is lower in the hidden layer $\mathbf{h}_i$ than in the case of the biLSTM baseline, but in the query layer $\mathbf{q}_i$ it is instead much higher. The hidden layer thus is representing other information than referent-specific knowledge, and the query layer can be seen as extracting referent-specific information from the hidden layer. Figure 8 visually illustrates the division of labor between the hidden and query layers. Second, we compared the models to variants where the cosine-similarity comparison is replaced by an ordinary dot-product transformation, which converts the querying mechanism into a simple further layer. These variants performed almost as well on the reference task, albeit with a slight but consistent edge for the models using cosine similarity.

No dynamic updates in EntNet. A surprising negative finding is that EntNet is not using its dynamic potential on the referential task. We confirmed this in two ways. First, we tracked the values $\mathbf{V}_i$ of the entity representations and found that the pointwise difference in $\mathbf{V}_i$ at any two adjacent time steps i tended to zero. Second, we simply switched off the update mechanism during testing and did not observe any score decrease on the reference task. EntNet is thus only using the part of the entity memory that it shares with EntLib, i.e., the keys $\mathbf{W_e}$, which explains their similar performance.

This finding is markedly different from Henaff et al. (2017), where for instance the BaBI tasks could be solved only by dynamically updating the entity representations. This may reflect our different language modules: since our LSTM module already has a form of dynamic memory, unlike the simpler sentence processing module in Henaff et al. (2017), it may be that the LSTM takes this burden off of the entity module. An alternative is that it is due to differences in the datasets. We leave an empirical comparison of these potential explanations for future work, and focus in Section 6 on the static entity representations $\mathbf{W}_e$ that EntNet essentially shares with EntLib.

This person is {a/an/the} <PROPERTY> [and {a/an/the} <PROPERTY>]{0,2}.
This person is the brother of Monica Geller.
This person is a paleontologist and a man.

Figure 9: Patterns and examples (in italics) of the dataset for information extraction as entity linking.

6 Analysis: Entity Representations

The foregoing demonstrates that entity-centric architectures help in a reference task, but not that the induced representations in fact contain meaningful entity information. In this section we deploy these representations on a new dataset, showing that they do not—not even for basic information about entities such as gender.

Method. We evaluate entity representations with an information extraction task including attributes and relations, using information from an independent, unstructured knowledge base—the Friends Central Wikia.[12] To be able to use the models as is, we set up the task in terms of entity linking, asking models to solve the reference of natural language descriptions that uniquely identify an entity. For instance, given *This person is the brother of Monica Geller.*, the task is to determine that *person* refers to *Ross Geller*, based on the information in the sentence.[13] The information in the descriptions was in turn extracted from the Wikia. We do not retrain the models for this task in any way—we simply deploy them.

We linked the entities from the Friends dataset used above to the Wikia through a semi-automatic procedure that yielded 93 entities, and parsed the Wikia to extract their attributes (*gender* and *job*) and relations (e.g., *sister*, *mother-in-law*; see Appendix for details). We automatically generate the natural language descriptions with a simple pattern (Figure 9) from combinations of properties that uniquely identify a given entity within the set of Friends characters.[14] We

---

[11] For the query layer, Eq. 13 is equivalent, with $\cos(q_{t_k}, q_{t_{k'}})$.

[12] `http://friends.wikia.com`.

[13] The referring expression is the whole DP, *This person*, but we follow the method in Aina et al. 2018 of asking for reference resolution at the head noun.

[14] Models require inputting a speaker; we use speaker UN-

model	description	gender	job
Random	1.5	50	20
biLSTM	0.4	-	-
EntLib	2.2	55	27
EntNet	1.3	61	24

Table 4: Results on the attribute and relation prediction task: percentage accuracy for natural language descriptions, mean reciprocal rank of characters for single attributes (lower is worse).

consider unique descriptions comprising at most 3 properties. Each property is expressed by a noun phrase, whereas the article is adapted (definite or indefinite) depending on whether that property applies to one or several entities in our data. This yields 231 unique natural language descriptions of 66 characters, created on the basis of overall 61 relation types and 56 attribute values.

Results. The results of this experiment are negative: The first column of Table 4 shows that models get accuracies near 0.

A possibility is that models do encode information in the entity representations, but it doesn't get used in this task because of how the utterance is encoded in the hidden layer, or that results are due to some quirk in the specific setup of the task. However, we replicated the results in a setup that does not encode whole utterances but works with single attributes and relations. While the methodological details are in the Appendix, the 'gender' and 'job' columns of Table 4 show that results are a bit better in this case but models still perform quite poorly: Even in the case of an attribute like gender, which is crucial for the resolution of third person pronouns (*he/she*), the models' results are quite close to that of a random baseline.

Thus, we take it to be a robust result that entity-centric models trained on the SemEval data do not learn or use entity information—at least as recoverable from language cues. This, together with the remainder of the results in the paper, suggests that models rely crucially on speaker information, but hardly on information from the linguistic context.[15] Future work should explore alternatives such as pre-training with a language modeling task, which could improve the use of context.

7 Conclusions

Recall that the motivation for entity-centric models is the hypothesis that incorporating entity representations into the model will help it better model the language we use to talk about them. We still think that this hypothesis is plausible. However, the architectures tested do not yet provide convincing support for it, at least for the data analyzed in this paper.

On the positive side, we have shown that framing models from an entity-centric perspective makes it very natural to adopt architectural decisions that are good inductive biases. In particular, by exploiting the fact that both speakers and referents are entities, these models can do more with the same model size, improving results on less frequent entities and emulating rule-based behavior such as "a first person expression refers to the speaker". On the negative side, we have also shown that they do not yield operational entity representations, and that they are not making good use of contextual information for the referential task.

More generally, our paper underscores the need for model analysis to test whether the motivations for particular architectures are borne out in how the model actually behaves when it is deployed.

Acknowledgments

We gratefully acknowledge Kristina Gulordava and Marco Baroni for the feedback, advice and support. We are also grateful to the anonymous reviewers for their valuable comments. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154), and from the Spanish Ramón y Cajal programme (grant RYC-2015-18907). We are grateful to the NVIDIA Corporation for the donation of GPUs used for this research. We are also very grateful to the Pytorch developers. This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

KNOWN.

[15] Note that 44% of the mentions in the dataset are first person, for which linguistic context is irrelevant and the models only need to recover the relevant speaker embedding to succeed. However, downsampling first person mentions did not improve results on the other mention types.

References

Laura Aina, Carina Silberer, Ionut-Teodor Sorodoc, Matthijs Westera, and Gemma Boleda. 2018. AMORE-UPF at SemEval-2018 Task 4: BiLSTM with Entity Library. In *Proc. of SemEval*.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. 2017. RelNet: End-to-End Modeling of Entities & Relations. *CoRR*, abs/1706.07179.

Razvan Bunescu and Marius Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proc. of EACL*.

Henry Y. Chen, Ethan Zhou, and Jinho D. Choi. 2017. Robust Coreference Resolution and Entity Linking on Dialogues: Character Identification on TV Show Transcripts. In *Proc. of CoNLL 2017*.

Hui Chen, Baogang Wei, Yonghuai Liu, Yiming Li, Jifang Yu, and Wenhao Zhu. 2018. Bilinear Joint Learning of Word and Entity Embeddings for Entity Linking. *Neurocomputing*, 294:12–18.

Jinho D Choi and Henry Y Chen. 2018. SemEval 2018 Task 4: Character Identification on Multiparty Dialogues. In *Proc. of SemEval*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural Text Generation in Stories Using Entity Representations as Context. In *Proc. of NAACL*.

Kevin Clark and Christopher D. Manning. 2016. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *Proc. of ACL*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What You Can Cram into a Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties. In *Proc. of ACL*.

Nick Craswell. 2009. Mean Reciprocal Rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. In *Proc. of NAACL:HLT*.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information. In *Proc. of EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Proc. of ICANN*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *CoRR*, abs/1410.5401.

Kristina Gulordava, Laura Aina, and Gemma Boleda. 2018a. How to Represent a Word and Predict it, too: Improving Tied Architectures for Language Modelling. In *Proc. of EMNLP*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018b. Colorless Green Recurrent Networks Dream Hierarchically. In *Proc. of NAACL*.

Aria Haghighi and Dan Klein. 2010. Coreference Resolution in a Modular, Entity-Centered Model. In *Proc. of NAACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. of ICCV*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the World State with Recurrent Entity Networks. In *Proc. of ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Armand Joulin and Tomas Mikolov. 2015. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In *Proc. of NIPS*.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of Linguistic Form and Function in Recurrent Neural Networks. *Computational Linguistics*, 43(4):761–780.

Hans Kamp and Uwe Reyle. 2013. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42. Springer Science+Business Media.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. 2008. Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience. *Frontiers in Systems Neuroscience*, 2:4.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *TACL*, 4(1):521–535.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proc. of CIKM*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. of NIPS*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proc. of NAACL*.

Aida Nematzadeh, Kaylee Burns, Erin Grant, Alison Gopnik, and Tom Griffiths. 2018. Evaluating Theory of Mind in Question Answering. In *Proc. of EMNLP*.

E.W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley.

Cheon-Eum Park, Heejun Song, and Changki Lee. 2018. KNU CI System at SemEval-2018 Task4: Character Identification by Solving Sequence-Labeling Problem. In *Proc. of SemEval*.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proc. of CoNLL*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-End Memory Networks. In *Proc. of NIPS*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning Global Features for Coreference Resolution. In *Proc. of NAACL:HLT*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-Aware Language Models. In *Proc. of EMNLP*.

A Appendices

A.1 Hyperparameter search

Besides the LSTM parameters, we optimize the token embeddings $\mathbf{W}_t$, the entity/speaker embeddings $\mathbf{W}_e$, as well as $\mathbf{W}_o$, $\mathbf{W}_q$, and their corresponding biases, where applicable (see Section 3). We used five-fold cross-validation with early stopping based on the validation score. We found that most hyperparameters could be safely fixed the same way for all three types. Specifically, our final models were all trained in batch mode using the Adam optimizer (Kingma and Ba, 2014), with each batch covering 25 scenes given to the model in chunks of 750 tokens. The token embeddings ($\mathbf{W}_t$) are initialized with the 300-dimensional word2vec vectors, $\mathbf{h}_i$ is set to 500 units, and entity (or speaker) embeddings ($\mathbf{W}_e$) to k = 150 units.With this hyperparameter setting, ENTLIB has fewer parameters than BILSTM: the linear map $\mathbf{W}_o$ of the latter (500 × 401) is replaced by the query extractor $\mathbf{W}_q$ (500 × 150) followed by (non-parameterized) similarity computations. This holds even if we take into account that the entity embedding $\mathbf{W}_e$ used in both models contains 274 entities that are never speakers and that are, hence, used by ENTLIB but not by BILSTM.

Our search also considered different types of activation functions in different places, with the architecture presented above, i.e., tanh before the LSTM and ReLU in the gate, robustly yielding the best results. Other settings tested—randomly initialized token embeddings, self-attention on the input layer, and a uni-directional LSTM—did not improve performance.

We then performed another random search ($>$ 200 models) over the remaining hyperparameters: learning rate (sampled from the logarithmic interval 0.001–0.05), dropout before and after LSTM (sampled from 0.0–0.3 and 0.0–0.1, respectively), weight decay (sampled from 10^{-6}–10^{-2}) and penalization, i.e., whether to decrease the relative impact of frequent entities by dividing the loss for an entity by the square root of its frequency. This paper reports the best model of each type, i.e., BILSTM, ENTLIB, and ENTNET, after training on all the training data without cross-validation for 20, 80 and 80 epochs respectively (numbers selected based on tendencies in training histories). These models had the following parameters:

	BILSTM	ENTLIB	ENTNET
learning rate:	0.0080	0.0011	0.0014
dropout pre	0.2	0.2	0.0
dropout post:	0.0	0.02	0.08
weight decay:	1.8e-6	4.3e-6	1.0e-5
penalization:	no	yes	yes

B Attribute and relation extraction

B.1 Details of the dataset

We performed a two-step procedure to extract all the available data for the SemEval characters. First, using simple word overlap, we automatically mapped the 401 SemEval names to the characters in the database. In a second, manual step, we corrected these mappings and added links that were not found automatically due to name alternatives, ambiguities or misspellings (e.g., SemEval *Dana* was mapped to *Dana Keystone*, and *Janitor* to *The Zoo Employee*). In total, we found 93 SemEval entities in Friends Central, and we extracted their attributes (gender and job) and their mutual relationships (relatives).

Model	**Gender** (93;2)		**Occupation**	**Relatives**
	(wo)man	(s)he	(24;17)	(56;24)
RANDOM	.50	.50	.20	.16
ENTLIB	.55	.58	.27	.22
ENTNET	**.61**	.56	.24	**.26**

Table 5: Results on the attribute prediction task (mean reciprocal rank; from 0 (worst) to 1 (best)). The number of considered test items and candidate values, respectively, are given in the parentheses. For gender, we used *(wo)man* and *(s)he* as word cues for the values (fe)male.

B.2 Alternative setup

We use the same models, i.e. ENTLIB and ENTNET trained on Experiment 1, and (without further training) extract representations for the entities from them. The former are directly obtained from the entity embedding $\mathbf{W}_e$ of each model.

In the **attribute prediction** task, we are given an attribute (e.g., *gender*), and all its possible values $\mathcal{V}$ (e.g., $\mathcal{V} = \{$*woman, man* $\}$). We formulate the task as, given a character (e.g., *Rachel*), producing a ranking of the possible values in descending order of their similarity to the character, where similarity is computed by measuring the cosine of the angle between their respective vector representations in the entity space. We obtain representations of attributes values, in the same space as the entities, by inputting each attribute value as a separate utterance to the models, and extracting the corresponding entity query ($\mathbf{q}_i$). Since the models also expect a speaker for each utterance, we set it to either *all entities*, *main entities*, a *random* entity, or *no* entity (i.e., speaker embedding with zero in all units), and report the best results.

We evaluate the rankings produced for both tasks in terms of mean reciprocal rank (Craswell, 2009), scoring from 0 to 1 (from worst to best) the position of the target labels in the ranking. The two first columns Table 5 presents the results. Our models generally perform poorly on the tasks, though outperforming a random baseline. Even in the case of an attribute like gender, which is crucial for the resolution of third person pronouns, the models' results are still very close to that of the random baseline.

Instead, the task of **relation prediction** is to, given a pair of characters (e.g., *Ross* and *Monica*), predict the relation R which links them (e.g., *sister, brother-in-law, nephew*; we found 24 relations that applied to at least two pairs). We approach this following the vector offset method introduced by Mikolov et al. (2013b) for semantic relations between words. This leverages on regularities in the embedding space, taking the embeddings of pairs that are connected by the same relation to have analogous spatial relations. For two pairs of characters (a, b) and (c, d) which bear the same relation R, we assume $\mathbf{a} - \mathbf{b} \approx \mathbf{c} - \mathbf{d}$ to hold for their vector representations. For a target pair (a, b) and a relation R, we then compute the following measure:

$$\mathrm{s}_{rel}((a,b),R) = \frac{\sum_{(x,y)\in R} \cos(\mathbf{a}-\mathbf{b}, \mathbf{x}-\mathbf{y})}{|R|} \tag{14}$$

Equation (14) computes the average relational similarity between the target character pair and the exemplars of that relation (excluding the target itself), where the relational similarity is estimated as the cosine between the vector differences of the two pairs of entity representations respectively. Due to this setup, we restrict to predicting relation types that apply to at least two pairs of entities. For each target pair (a, b), we produce a rank of candidate relations in descending order of their scores s_{rel}. Table 5 contains the results, again above baseline but clearly very poor.

Continuous Learning for Large-scale Personalized Domain Classification

Han Li[1], **Jihwan Lee**[2], **Sidharth Mudgal**[2], **Ruhi Sarikaya**[2], and **Young-Bum Kim**[2]

[1]University of Wisconsin, Madison
`hanli@cs.wisc.edu`
[2]Amazon Alexa AI
`{jihwl, sidmsk, rsarikay, youngbum}@amazon.edu`

Abstract

Domain classification is the task of mapping spoken language utterances to one of the natural language understanding domains in intelligent personal digital assistants (IPDAs). This is a major component in mainstream IPDAs in industry. Apart from official domains, thousands of third-party domains are also created by external developers to enhance the capability of IPDAs. As more domains are developed rapidly, the question of how to continuously accommodate the new domains still remains challenging. Moreover, existing continual learning approaches do not address the problem of incorporating personalized information dynamically for better domain classification. In this paper, we propose CoNDA, a neural network based approach for domain classification that supports incremental learning of new classes. Empirical evaluation shows that CoNDA achieves high accuracy and outperforms baselines by a large margin on both incrementally added new domains and existing domains.

1 Introduction

Domain classification is the task of mapping spoken language utterances to one of the natural language understanding (NLU) domains in intelligent personal digital assistants (IPDAs), such as Amazon Alexa, Google Assistant, and Microsoft Cortana, etc. (Sarikaya, 2017). Here a domain is defined in terms of a specific application or functionality such as weather, calendar or music, which narrows down the scope of NLU. For example, given an utterance "*Ask Uber to get me a ride*" from a user, the appropriate domain would be one that invokes the "Uber" app.

Traditionally IPDAs have only supported dozens of well-separated domains, where each is defined in terms of a specific application or functionality such as calendar and weather (Sarikaya et al., 2016; Tur and De Mori, 2011; El-Kahky et al., 2014). In order to increase the domain coverage and extend the capabilities of the IPDAs, mainstream IPDAs released tools to allow third-party developers to build new domains. Amazons Alexa Skills Kit, Googles Actions and Microsofts Cortana Skills Kit are examples of such tools. To handle the influx of new domains, large-scale domain classification methods like SHORTLISTER (Kim et al., 2018b) have been proposed and have achieved good performance.

As more new domains are developed rapidly, one of the major challenges in large-scale domain classification is how to quickly accommodate the new domains without losing the learned prediction power on the known ones. A straightforward solution is to simply retraining the whole model whenever new domains are available. However, this is not desirable since retraining is often time consuming. Another approach is to utilize continual learning where we dynamically evolve the model whenever a new domain is available. There is extensive work on the topic of continual learning, however there is very little on incrementally adding new domains to a domain classification system.

To mitigate this gap, in this paper we propose the CoNDA solution for continuous domain adaptation. Given a new domain, we keep all learned parameters, but only add and update new parameters for the new domain. This enables much faster model updates and faster deployment of new features to customers. To preserve the learned knowledge on existing domains to avoid the notorious catastrophic forgetting problem (Kemker et al., 2018), we propose cosine normalization for output prediction and domain embedding regularization for regularizing the new domain embedding. Also, we summarize the data for existing domains by sampling exemplars, which will be used together

Proceedings of NAACL-HLT 2019, pages 3784–3794
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

with the new domain data for continuous domain adaptation. This is shown to further alleviate the overfitting on the new domain data. Empirical evaluation on real data with 900 domains for initial training and 100 for continuous adaptation shows that CoNDA out performs the baselines by a large margin, achieving 95.6% prediction accuracy on average for the 100 new domains and 88.2% accuracy for all seen domains after 100 new domains have been accommodated (only 3.6% lower than the upperbound by retraining the model using all domain data). To summarize, we make the following contributions in this paper:

- We introduce the problem of continuous domain adaptation for large-scale personalized domain classification.
- We describe CoNDA, a new solution for continuous domain adaptation with Cosine normalization, domain embedding regularization and negative exemplar sampling techniques. Our solution advances the research in continuous domain adaptation.
- We conduct extensive experiments showing that CoNDA achieves good accuracy on both new and existing domains, and outperforms the baselines by a large margin.

2 Background and Problem Definition

2.1 Domain Classification

Domain classification is the task of mapping spoken language utterances to one of the NLU domains in IPDAs. A straightforward solution to tackle this problem is to ask users to explicitly mention the domain name with a specific invocation pattern. For example, for the utterance "*Ask Uber to get me a ride*", the invocation pattern is "*Ask* $\{$*domain*$\}$ *to* $\{$*perform action*$\}$". While it makes things much simpler for the domain classifier, this significantly limits natural interaction with IPDAs as users need to remember the domain names as well as the invocation pattern. To address this limitation, *name-free domain classification* methods were developed for more user friendly interactions, and have been getting more attention recently. We specifically focus on the name-free scenario in this paper.

2.2 The Shortlister System

To our knowledge, the state-of-the-art for name-free domain classification is SHORTLISTER (Kim et al., 2018b), which leverages personalized user provided information for better classification performance. Specifically, it contains three main modules.

The first module is the LSTM-based encoder to map an utterance to a dimension-fixed vector representation. Given an utterance, each word is first represented as dense vectors using word embeddings, then a bidirectional LSTM (Graves and Schmidhuber, 2005) is be used to encode the full utterance.

The second module is the personalized domain summarization module. For each utterance from an IPDA user, a list of domains have been enabled by the user. These enabled domains can be viewed as user-specific personalized information. It has been shown that the domain classification accuracy can be significantly improved by leveraging information about enabled domains (Kim et al., 2018b). To represent the domain enablement information, first each enabled domain is mapped to a fixed-dimensional embedding, then a summarization vector is generated by taking an attention weighted sum (Luong et al., 2015) over the enabled domain embeddings.

Once the utterance representation and the enabled domain summarization are calculated, we concatenate the two vectors as the final representation. Then the third module, a feed-forward network, is used to predict the confidence score with a sigmoid function for each domain.

2.3 Continuous Domain Adaptation

As more new domains are developed, a major challenges in large-scale domain classification is quickly accommodating the new domains into the live production domain classification model without having to perform a full retrain. We refer to this problem as *Continuous Domain Adaptation* (CDA). In this paper, we specifically focus on the case of purely online learning where new domains where added one by one, since in practice we want to quickly integrate a new domain into the system as soon as it becomes available. We formally define the problem below.

Definition 1 *(Online continuous domain adaptation) Given a collection of* k *domains* $S_k = \{s_1, s_2, \ldots, s_k\}$, *suppose we have a dataset* $\mathcal{D}_k$ *defined on* S_k *where each item is a triple* (u, s, E) *with the utterance* $u \in U$ *(the set for all possible utterances), the ground-truth domain* $s \in S_k$, *and* *the enabled domains $E \subseteq S_k$. Denote $\mathcal{P}(S_k)$ as the powerset of S_k, a model $M_k : U \times \mathcal{P}(S_k) \rightarrow S_k$ has been trained on $\mathcal{D}_k$ for domain classification with the accuracy $M_k(\mathcal{D}_k)$. At some point, a new domain s_{k+1} is available with the corresponding dataset $D_{k+1} = \{(u, s_{k+1}, E) \mid E \subseteq S_{k+1}\}$ with $S_{k+1} = S_k \cup \{s_{k+1}\}$. Taking advantage of D_{k+1}, the continuous adaptation for s_{k+1} is to update M_k to $M_{k+1} : U \times \mathcal{P}(S_{k+1}) \rightarrow S_{k+1}$ so that the model can make predictions for s_{k+1}, with the goal of maximizing $M_{k+1}(D_{k+1})$ and minimizing $M_k(\mathcal{D}_k) - M_{k+1}(\mathcal{D}_k)$.*

3 The CoNDA Solution

We introduce CoNDA (**Co**ntinuous **N**eural **D**omain **A**daptation), a variation of SHORTLISTER that is capable of handling online CDA decribed in Definition 1. Similar to SHORTLISTER, it has three main modules.

The first module is the LSTM-based utterance encoder which shares the same architecture as the one used in SHORTLISTER, that maps an input utterance into a dense vector. After the training on the initial k-domain data $\mathcal{D}_k$, we freeze all parameters (i.e., the word embedding lookup and the bi-LSTM parameters) of this module from changing for the subsequent online domain adaptation tasks. Usually the value of k is large enough (hundreds or even thousands in real-world, at least 100 in our experiments), thus it is safe to assume that the parameters have been tuned sufficiently well to encode utterances from all existing and future domains. In this work we treat new words in the new domains as unknown and leave the problem of vocabulary expansion as future work.

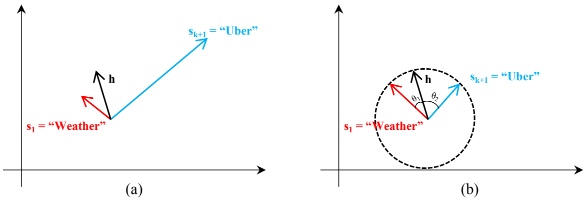
The second module is the personalized domain summarization module which will map the enabled domains of an input utterance to a dense vector representation. It is also similar to the one in SHORTLISTER, except we will evolve the module as we are adding new domains. Specifically, given dataset $\mathcal{D}_k$ on k domains for initial training, a domain embedding table $T_k \in R^{k \times d_s}$ will be learned where d_s is the size of the domain embeddings. When a new domain s_{k+1} is available, we expand T_k to $T_{k+1} \in R^{(k+1) \times d_s}$ by: (1) freezing the learned embeddings for all known domains; (2) adding a new row $t_{k+1} \in R^{d_s}$ to T_k as the domain embedding for s_{k+1} and updating the new parameters t_{k+1} using all available training data at hand (i.e., the dataset D_{k+1} and the negative samples which will be discussed later in this section). We repeat this procedure whenever a new domain is available. To avoid over-fitting on t_{k+1}, we introduce a new regularization term into the loss function. We describe the details in Section 3.2.

The third module is a two-layer feed-forward network as the classifier. The first layer $f^{(1)} : R^{d_u+d_s} \rightarrow R^{d_h}$ maps the concatenation of the utterance embedding (in size d_u) and domain summarization (in size d_s) into fix-sized hidden representation (in size d_h) using a fully connected layer followed by SELU activation (Klambauer et al., 2017), which is identical to the one in SHORTLISTER. Then the prediction layer $f^{(2)} : R^{d_h} \rightarrow R^k$ maps the hidden representation to the final domain prediction scores. Unlike SHORTLISTER where the final prediction score is the dot product of the weight vector and the hidden representation, we choose to use the cosine score of the two, referred to as *cosine normalization*. To support online CDA when a new domain is available, we apply a similar approach to the domain embedding expansion described above to expand the prediction layer. Specifically, denote $W_k^{(2)} \in R^{k \times d_h}$ be the weights for the prediction layer that has been trained on the initial k domains. To adapt the new domain d_{k+1}, we expand $W_k^{(2)}$ to $W_{k+1}^{(2)} \in R^{(k+1) \times d_h}$ by first freezing all learned parameters and adding a new row of learnable parameters $w_{k+1} \in R^{d_h}$ to $W_k^{(2)}$.

As each time we only add one new domain, all training utterances during the update will have the same label. Thus, it's easy to overfit the new data such that catastrophic forgetting occurs. Inspired by (Rebuffi et al., 2017), we also propose a negative sampling procedure to leverage (limited) information on the known domains to alleviate the catastrophic forgetting problem. For the rest of the section, we will first talk about cosine normalization, and then domain embedding regularization, and finally negative sampling.

3.1 Cosine Normalization

As mentioned above, we use the cosine similarity of the weights and the hidden representation vector instead of the linear dot product in the prediction layer. Formally, let $f_k^{(2)} : R^{d_h} \rightarrow [-1, 1]^k$ be the prediction layer for k domains with parameters $W_k^{(2)} \in R^{k \times d_h}$. Given an input hidden representation $h \in R^{d_h}$ from $f^{(1)}$, the score for the i-th



(a) (b)

Figure 1: Cosine Example.

domain under cosine normalization is:

$$f_{k,i}^{(2)}(h) = \cos\left(h, W_{k,i}^{(2)}\right) = \frac{h \cdot W_{k,i}^{(2)}}{\|h\| \left\|W_{k,i}^{(2)}\right\|} \tag{1}$$

To understand why cosine is better in the case of online CDA, let's first see the problem with the dot-product method. Suppose we are accommodating s_{k+1} with dataset D_{k+1}, because we train the new parameters w_{k+1} only on D_{k+1} where all utterances have the same domain s_{k+1}, the model can easily get good training performance on $M_{k+1}(D_{k+1})$ by simply maximizing the values in w_{k+1} such that the dot product of the hidden representation with w_{k+1} is larger than the dot product with any other $w_i, 1 \leq i \leq k$. Effectively this leads to the model predicting domain s_{k+1} for any given utterance. Using cosine normalization instead as described in Eq. 1 removes the incentive to maximize the vector length of w_{k+1}.

Example 1 *Suppose M_k has been initially trained on $\mathcal{D}_k$, and domain s_1="Weather". Given an utterance u = "What's the weather today?", M_k correctly classifies u into s_1. Now a new domain s_{k+1}="Uber" is coming and we evolve M_k to M_{k+1}. As the norm of the weights w_{k+1} could be much larger than w_1 in the prediction layer, even if the hidden representation h of u is closer to s_1 in direction, M_{k+1} will classifier u into s_{k+1} as it has a higher score, shown in Figure 1.a. However if we measure the cosine similarity, M_{k+1} will classify u correctly because we now care more about the directions of the vectors, and the angle θ_1 between h and s_1 is smaller (representing higher similarity) than the angle θ_2 between h and s_{k+1}, as shown in Figure 1.b.*

As we use the cosine normalization, all prediction scores are mapped into the range [-1, 1]. Therefore it's not proper to use log-Sigmoid loss function as in SHORTLISTER. So accompanying with the cosine normalization, the following hinge loss function has been used instead:

$$\begin{aligned}\mathcal{L}_{hinge} = &\sum_{i=1}^{n} y_i \max\{\Delta_{pos} - o_i, 0\} \\ &+ \sum_{i=1}^{n} (1 - y_i) \max\{o_i - \Delta_{neg}, 0\}\end{aligned} \tag{2}$$

where n is the number of all domains, o_i is the predicted score for each domain, y is a n-dimensional one-hot vector with 1 in the ground-truth label and 0 otherwise. Δ_{pos} and Δ_{neg} are the hinge thresholds for the true and false label predictions respectively. The reason we use hinge loss here is that it can be viewed as another way to alleviate the overfitting on new data, as the restrictions are less by only requiring the prediction for the ground-truth to be above Δ_{pos} and false domain predictions below Δ_{neg}. Our experiments show that this helps the model get better performance on the seen domains.

3.2 Domain Embedding Regularization

In this section, we introduce the regularizations on the domain embeddings used in the personalized domain summarization module. Recall that given an utterance u with h^u as the hidden representation from the encoder and its enabled domains E, personalized domain summarization module first compares u with each $s_i \in E$ (by calculating the dot product of h^u and the domain embedding t_i of s_i) to get a score a_i, then gets the weight $c_i = \exp(a_i)/\sum_{a_j} \exp(a_j)$ for domain s_i, and finally computes the personalized domain summary as $\sum_{e_i \in E} c_i \cdot t_i$. We observed that after training on the initial dataset D_k, the domain embedding vectors tend to roughly cluster around a certain (random) direction in the vector space. Thus, when we add a new domain embedding s_{k+1} to this personalization module, the model tends to learn to move this vector to a different part of the vector space such that its easier to distinguish the new domain from all other domains. Moreover, it also increases the ℓ_2 norm of the new domain embedding t_{k+1} to win over all other domains.

Example 2 *Suppose a similar scenario to Example 1 where we have s_1 = "Weather" in S_k and a new domain s_{k+1} = "Uber". As most utterances in D_{k+1} have s_{k+1} as an enabled domain, it's easy for the model to learn to enlarge the norm of the new domain embedding t_{k+1} as well as make it close to the context of ride sharing, so that t_{k+1}*

can dominate the domain summarization. Then coordinating with the new weights w_{k+1} in the prediction layer $f^{(2)}_{k+1}$, the network can easily predict high scores s_{k+1} and fit the dataset D_{k+1}. However, when we have utterances belonging to s_1 with s_{k+1} as an enabled domain, s_{k+1} may still dominate the summarization which makes the prediction layer tends to cast those utterances to s_{k+1}. We don't observe this on the initial training on $\mathcal{D}_k$ because s_{k+1} was not visible at that time, thus cannot be used as an enabled domain. And it's even worse if s_1 is similar to s_{k+1} in concept. For example if s_1 = "Lyft", in this case the utterances of the two domains are also similar, making the dot product of t_{k+1} and the hidden representations of the s_1's utterances even larger.

To alleviate this problem, we add a new domain embedding regularization term in the loss function to constrain the new domain embedding vector length and force it to direct to a similar area where the known domains are heading towards, so that the new domain will not dominate the domain summarization. Specifically,

$$\mathcal{L}_{der} = \sum_{i=1}^{k} \lambda_i \max\{\Delta_{der} - \cos(t_{k+1}, t_i), 0\} + \frac{\lambda_{norm}}{2} \|t_{k+1}\|^2 \quad (3)$$

We call the first part of Eq. 3 on the right hand side as the *domain similarity loss* where we ask the new domain embedding t_{k+1} to be similar to known domain t_i's controlled by a Cosine-based hinge loss. As we may not need t_{k+1} to be similar to all seen domains, a coefficient λ_i is used to weight the importance each similarity loss term. In this paper we encourage t_{k+1} to be more similar to the ones sharing similar concepts (e.g. "Uber" and "Lyft"). We assume all training data are available to us, and measure the similarity of two domains by comparing their average of utterance hidden representations.

Specifically, denote $\varphi : U \rightarrow R^{d_u}$ as the LSTM-encoder that will map an utterance to its hidden representation with dimension d_u. For each domain $s_i \in S_{k+1}$, we first calculate the average utterance representation on D_i

$$\widetilde{h}_i = \sum_{(u,s_i,e)\in D_i} \frac{\varphi(u)}{|D_i|} \quad (4)$$

Then we set $\lambda_i = \lambda_{dsl} \max\{\cos(\widetilde{h}_i, \widetilde{h}_{k+1}), 0\}$ with λ_{dsl} as a scaling factor.

Combining Eq. 2 and 3, the final loss function for optimization is: $\mathcal{L}_{total} = \mathcal{L}_{hinge} + \mathcal{L}_{der}$

3.3 Sampling Negative Exemplars

So far we developed our method by training only on the new data D_{k+1}, and use regularizations to prevent overfitting. However, in many real applications all of the training data, not only D_{k+1}, is actually available, but it's not affordable to retrain the full model using all data. Inspired by (Rebuffi et al., 2017), we can select a set of exemplars from the previously trained data to further improve continual adaptation.

Suppose we are handling the new domain s_{k+1} with D_{k+1}, and all data trained previously is $\mathcal{D}_k$ on k domains S_k. For each known $s_i \in S_k$, we pick N utterances from D_i as the exemplars for s_i. Denote P_i be the exemplar set for s_i and $P = \bigcup_{i=1}^{k} P_i$ be the total exemplar set. To generate each P_i, we pick the top-N utterances that are closest to the average of the utterance hidden representation. Specifically, following Eq. 4, we first get the average representation $\widetilde{h}_i$, then P_i is defined as follow:

$$P_i =_{P_i \subseteq D_i, |P_i|=N} \sum_{(u,s_i,e)\in P_i} \cos\left(\varphi(u), \widetilde{h}_i\right) \quad (5)$$

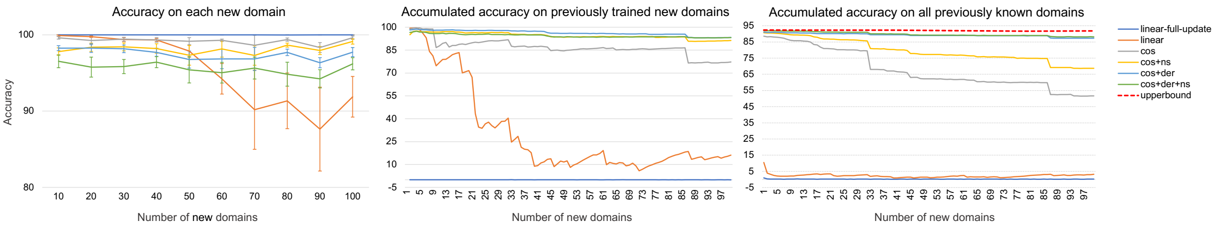
If multiple candidates satisfying Eq. 5 for P_i, we randomly pick one as P_i to break the tie. Once the domain adaptation for s_{k+1} is done, we similarly generate P_{k+1} and merge it to P. We repeat this procedure for negative sampling whenever a new domain is coming later.

As we add more new domains, the exemplar set P also grows. For some new domain D_{k+1}, we may have $|P| \gg |D_{k+1}|$. In this case, the prediction accuracy on the new domain data could be very low as the model will tend to not making mistakes on P rather than fitting D_{k+1}. To alleviate this problem, when $|P| > |D_{k+1}|$, we select a subset $P' \subseteq P$ with $|P'| = |D_{k+1}|$, and P' will be used as the final exemplar set to train together with D_{k+1}. To generate P', we just randomly sample a subset from P, since it was observed to be effective in our experiments.

4 Empirical Evaluation

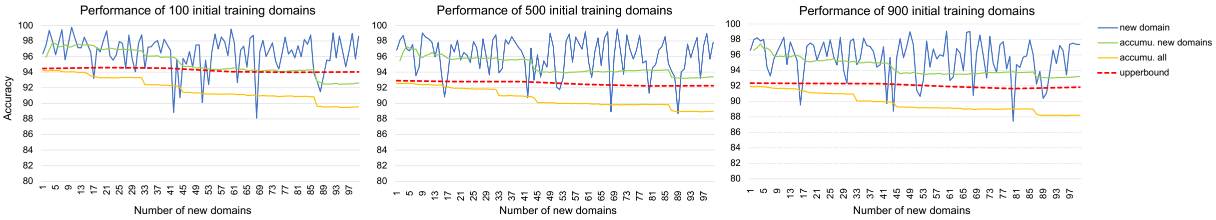
4.1 Experiment Setup

Dataset: We use a dataset defined on 1000 domains for our experiments which has 2.53M utterances, and we split them into two parts. The first



(a) (b) (c)

Figure 2: Overall evaluation. (a) shows the accuracy for new domains. (b) shows the accumulated accuracy for previous new domains that have been adapted to the model so far. (c) shows the accumulated accuracy for all known domains including the ones used for initial training and all previously adapted new domains.



(a) (b) (c)

Figure 3: The model performance on different number of initial training domains. The red dashed line shows the upperbound of the accumulated accuracy, which is generated by retraining the model on all domains seen so far.

part contains 900 domains where we use it for the initial training of the model. It has 2.06M utterances, and we split into training, development and test sets with ratio of 8:1:1. We refer to this dataset as "InitTrain". The second part consists of 100 domains and is used for the online domain adaptation. It has 478K utterances and we split into training, development and test sets with the same 8:1:1 ratio. We refer to this dataset as "IncTrain".

Training Setup: We implement the model in PyTorch (Paszke et al., 2017). All of the experiments are conducted on an Amazon AWS p3.16xlarge[1] cluster with 8 Tesla V100 GPUs. For initial training, we train the model for 20 epochs with learning rate 0.001, batch size 512. For the continuous domain adaptation, we add the new domains in a random order. Each domain data will be trained independently one-by-one for 10 epochs, with learning rate 0.01 and batch size 128. For both training procedures, we use Adam as the optimizer. The development data is used to pick the best model in different epoch runs. We evaluate the classification accuracy on the test set.

4.2 Overall Performance

We first talk about the overall performance. In our experiments we select two baselines. The first one `linear-full-update` which simply extends SHORTLISTER by adding new parameters for new domains and conducting full model updating. The second `linear` is similar to the first baseline except that we freeze all trained parameters and only allow new parameter updating. Both the two baselines update the model with D_{k+1} dataset only. To show the effectiveness of each component of CONDA, we choose four variations. The first one is `cos` where we apply the Cosine Normalization (CosNorm). The second one `cos+der` applies CosNorm with the domain embedding regularization. The third one `cos+ns` uses both CosNorm and negative exemplars. And the last one `cos+der+ns` is the combination of all three techniques, which is our CONDA model. For hyperparameters, we pick $\Delta_{pos} = 0.5, \Delta_{neg} = 0.3, \Delta_{der} = 0.1, \lambda_{dsl} = 5$, and $\lambda_{norm} = 0.4$.

Figure 2 shows the accuracy for new domain adaptations. From the figure, here are the main observations. First, without any constraints, `linear-full-update` can easily overfits the new data to achieve 100% accuracy as shown in Figure 2(a), but it causes catastrophic forgetting such that the accuracy on seen domains is (almost) 0 as shown in Figure 2(b) and (c). By freezing the all trained parameters, the catastrophic forgetting problem is a bit alleviated for `linear`, but the accuracy on the seen domains is still very low as we add more new domains. Second, `cos` produces much better accuracy on seen domains with a bit

---

[1] https://aws.amazon.com/ec2/instance-types/p3/

lower accuracy on each new domain, showing the effectiveness of the Cosine normalization. Third, as we add more regularizations to the model, we get better accuracy on the seen domains (Figure 2 (b) and (c)), at the cost of sacrificing a bit on the new domain accuracy (Figure 2 (a)). Also, `cos+der+ns` (the CoNDA model) achieves the best performance, with an average of 95.6% accuracy for each new domain and 88.2% accuracy for all previously seen domains after we add 100 new ones, which is only 3.6% lower than the upperbound (by retraining the model on the whole dataset). These demonstrate the superiority of our method.

4.3 Micro-benchmarks

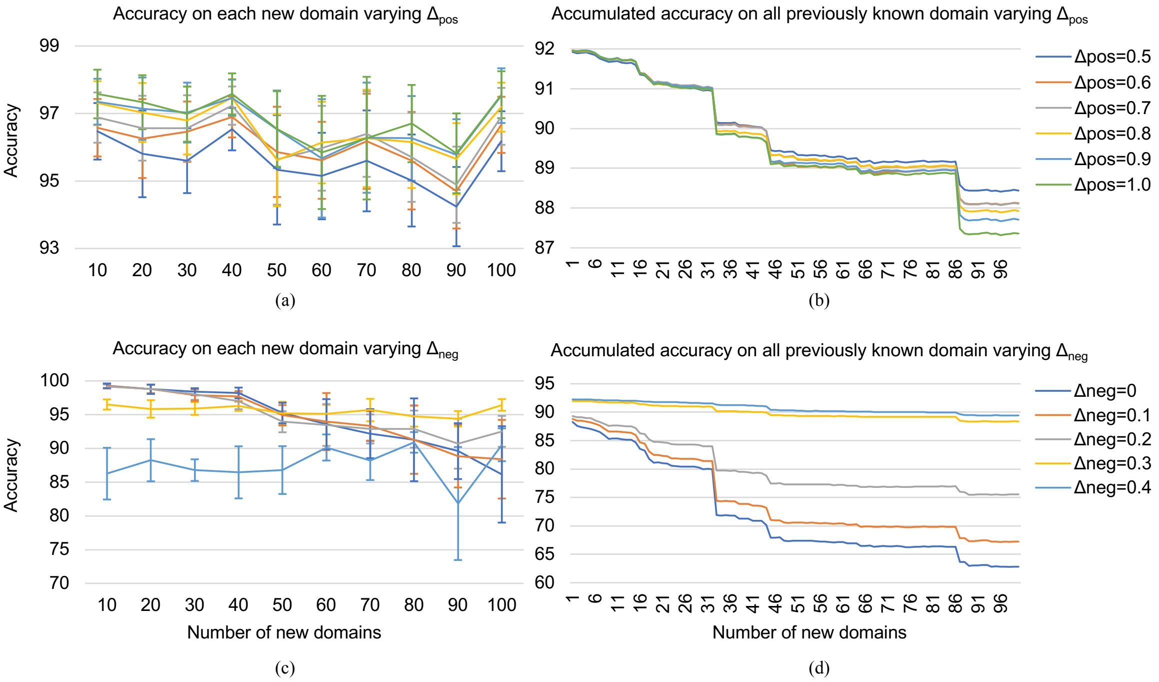
Using Different Number of Initial Domains: We vary the number of domains for initial training to see if it will have a big impact on the model performance. Specifically, we pick 100 and 500 domains from InitTrain, and use the same IncTrain data for domain adaptation. Figure 3 compares the model performance on these three different number (i.e., 100, 500, 900) of initial training domains. From the figure we can see that the curves share a similar pattern regardless of the number of initial domains, showing that our model is stable to the number of domains used for initial training.

Varying the hinge loss thresholds: We vary the classification hinge loss thresholds Δ_{pos} and Δ_{neg} to see how it will affect the performance. Specifically, we fix $\Delta_{neg} = 0.3$ and vary Δ_{pos} from 0.5 to 1.0, and fix $\Delta_{pos} = 0.5$ and vary Δ_{neg} from 0 to 0.4, respectively. For both of the them we use 0.1 as the step size. Figure 4 shows the model performance. From the figures, we summarize the following observations. First, as we increase Δ_{pos}, on average the accuracy on each new domain gets better (Figure 4(a)), but we loss performance on all seen domains (Figure 4(b)). This is in accord with our intuition that a larger Δ_{pos} puts more constraint on the new domain predictions such that it tends to overfit the new data and exacerbates catastrophic forgetting on existing domains. Second, as we increase Δ_{neg}, on average the accuracy on each new domain gets worse (Figure 4(c)), but we get better performance on existing domains. This is because a larger Δ_{neg} narrows down the prediction "margin" between positive and negative domains (similar to decreasing Δ_{pos}), so that less constraint has been put onto predictions to alleviate overfitting on the new domain data.

Varying the domain similarity loss threshold: We vary the threshold Δ_{der} to see how it will affect the model performance. Specifically, we vary Δ_{der} from 0 to 0.5 with step size 0.1, and Figure 5 shows the model performance. As we increase Δ_{der}, the performance on the new domains gets worse, and the drop is significant when Δ_{der} is large. On the other hand, the accumulated accuracy on seen domains increases when we start to increase Δ_{der}, and drops when Δ_{der} is too large. This means we when we start to make the new domain embeddings to be similar to the existing ones, we alleviate the problem that the new domain dominates the domain summarization. Thus the accuracy on existing domains improves at the cost of sacrificing some accuracy on the new domains. However, if we continue to increase Δ_{der} to make it very similar to some of existing domains, the new domain will compete with some existing ones so that we loss accuracy on both new and existing domains.

Varying the weights for domain similarity loss: To see how the weighted domain similarity loss will affect the performance, we compare it against the plain version without the utterance similarity weights. Specifically, we set each $\lambda_i = \lambda_{dsl}$ having the same value. And our experiments show that the plain version gets the average accuracy 94.1% on the new domains, which is 1.5% lower than the weighted version, and 88.7% accumulated accuracy on all domains after adding 100 new domains, which is 0.5% higher than the weighted version. This means we can get a bit higher accumulated accuracy at the cost of sacrificing more new domain accuracy. In real applications, the decision to whether use weighted domain similarity loss should be made by trading off the importance of the new and existing domains.

Varying the number of used negative exemplars: As we mentioned before, we down-sample the negative exemplar set P to reduce the impact on new domain performance. To see if it's necessary, we compare it against the one without down-sampling. Our experiments show that without down-sampling, the model achieves 87.5% new domain accuracy on average which is 8.1% lower than the down-sampling version, and 87.2% accumulated accuracy on all domains which is 1.0% lower than the down-sampling one.



(a) (b) (c) (d)

Figure 4: Model performance by varying the hinge loss thresholds. (a) and (b) shows accuracy on new domain and accumulated accuracy for all seen domains respectively by varying Δ_{pos}. Similarly, (c) and (d) shows the accuracy performance by varying Δ_{neg}.

Prediction Method	**100 initial domains**	**500 initial domains**	**900 initial domains**
Linear	95.2	93.9	93.4
Cosine	94.5	92.9	92.4

Table 1: Linear dot product versus Cosine normalization on initial training for different number of domains.

This means down-sampling effectively improve the model performance.

Effect of Cosine normalization in initial training: We have shown Cosine normalization with hinge loss works better than linear dot product with sigmoid loss (used in SHORTLISTER) for CDA. Here we compare the two on the regular training setting where we train the model from scratch on a large dataset. Specifically, we compare the initial training performance on 100, 500, and 900 domains which are the same as we used earlier. Table 1 shows the accuracy numbers. From the table we see that Linear works better than Cosine by 0.7-1.0% across different number of domains. Though the difference is not large, this means Linear could be a better option than Cosine when we train the model from scratch.

Varying the order of the new domains: To see if the incoming order of the new domains will affect the performance, we generate two different orders apart from the one used in overall evaluation. The first one sorts the new domains on the number of utterances in the decreasing order, and the second in the increasing order. Denote these three orders as "random", "decreasing", and "increasing", and we conduct domain adaptation on these orders. Our experiments show that they achieve 95.6%, 95.5%, and 95.6% average accuracy on new domains respectively, and 88.2%, 88.2%, and 88.1% accumulated accuracy on all domains after accommodating all 100 new domains. This indicates that there is no obvious difference on model performance, and our model is insensitive to the order of the new domains.

Using more new domains: We also experimented with adding a large number of new domains to see the limit of CONDA. Figure 6 shows the results by continuously adapting 900 new domains one-by-one. From the figure we can see that at the early stage of the new domain adaptation (e.g., first 200 new domains), we get high new domain accuracy with little performance decrease on the existing domains. After that, the new domain performance becomes more unstable with violent oscillation, and the existing domain accuracy decreases more quickly. This suggests that we cannot run the new domain adaptation forever, and

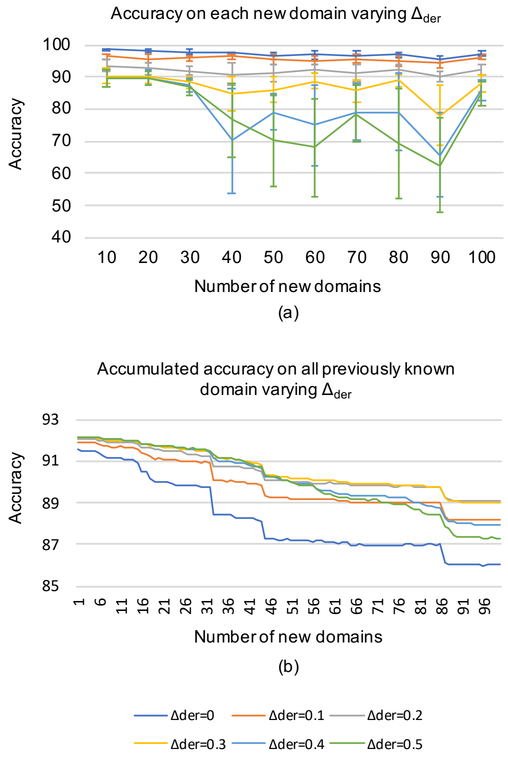


Figure 5: Model performance by varying Δ_{der}. (a) shows the accuracy on new domains, and (b) shows the accumulated accuracy for all seen domains.

after adapting a certain number of new domains (e.g., 200 new domains), it's more preferable to train the whole model from scratch.

5 Related Work

Domain Classification: Traditional domain classifiers were built on simple linear models such as Multinomial logistic regression or Support Vector Machines (Tur and De Mori, 2011). They were typically limited to a small number of domains which were designed by specialists to be well-separated. To support large-scale domain classification, (Kim et al., 2018b) proposed SHORTLISTER, a neural-based model. (Kim et al., 2018a) extended SHORTLISTER by using additional contextual information to rerank the predictions of SHORTLISTER. However, none of them can continuously accommodate new domains without full model retrains.

Continuous Domain Adaptation: To our knowledge, there is little work on the topic of continuous domain adaptation for NLU and IPDAs. (Kim et al., 2017) proposed an attention-based method for continuous domain adaptation, but it introduced a separate model for each domain and therefore is difficult to scale.

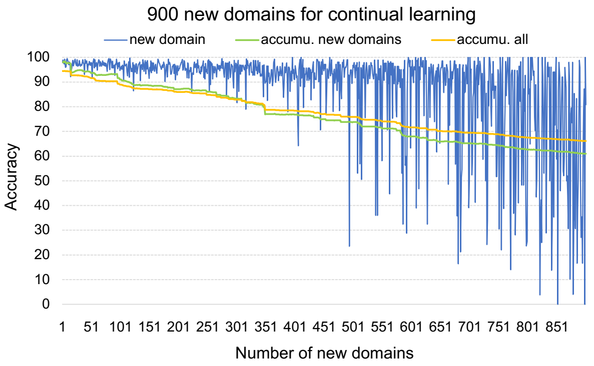


Figure 6: Using 900 new domains for continual learning.

Continual Learning: Several techniques have been proposed to mitigate the catastrophic forgetting (Kemker et al., 2018). Regularization methods add constraints to the network to prevent important parameters from changing too much (Kirkpatrick et al., 2017; Zenke et al., 2017). Ensemble methods alleviate catastrophic forgetting by explicitly or implicitly learning multiple classifiers and using them to make the final predictions (Dai et al., 2009; Ren et al., 2017; Fernando et al., 2017). Rehearsal methods use data from existing domains together with the new domain data being accommodated to mitigate the catastrophic forgetting (Robins, 1995; Draelos et al., 2017; Rebuffi et al., 2017). Dual-memory methods introduce new memory for handling the new domain data (Gepperth and Karaoguz, 2016). Among the existing techniques, our model is most related to the regularization methods. However, unlike existing work where the main goal is to regularize the learned parameters, we focus on regularizations on the newly added parameters. Our model also shares similar ideas to (Rebuffi et al., 2017) on the topic of negative exemplar sampling.

6 Conclusion and Future Work

In this paper, we propose CONDA for continuous domain adaptation. By using various normalization and regularizations, our model achieves high accuracy on both the accommodated new domains and the existing known domains, and outperforms the baselines by a large margin. For future work, we consider extending the model to handle unknown words. Also, we want to find a more principled way to down sample the negative exemplars.

References

Wenyuan Dai, Ou Jin, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2009. Eigentransfer: a unified framework for transfer learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 193–200. ACM.

Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Jonathan A Cox, Craig M Vineyard, Kristofor D Carlson, William M Severa, Conrad D James, and James B Aimone. 2017. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 526–533. IEEE.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4067–4071. IEEE.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Alexander Gepperth and Cem Karaoguz. 2016. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Young-Bum Kim, Dongchan Kim, Joo-Kyung Kim, and Ruhi Sarikaya. 2018a. A scalable neural shortlisting-reranking approach for large-scale domain classification in natural language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HTL 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*.

Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018b. Efficient large-scale neural domain classification with personalized attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2214–2224.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proc. CVPR*.

Boya Ren, Hongzhi Wang, Jianzhong Li, and Hong Gao. 2017. Life-long learning based on dynamic combination model. *Applied Soft Computing*, 56:398–404.

Anthony Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 391–397. IEEE.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International*

Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pages 3987–3995.

Cross-Lingual Transfer Learning for Multilingual Task Oriented Dialog

Sebastian Schuster[*]
Stanford Linguistics
`sebschu@stanford.edu`

Sonal Gupta
Facebook Conversational AI
`sonalgupta@fb.com`

Rushin Shah
Facebook Conversational AI
`rushinshah@fb.com`

Mike Lewis
Facebook AI Research
`mikelewis@fb.com`

Abstract

One of the first steps in the utterance interpretation pipeline of many task-oriented conversational AI systems is to identify user intents and the corresponding slots. Since data collection for machine learning models for this task is time-consuming, it is desirable to make use of existing data in a high-resource language to train models in low-resource languages. However, development of such models has largely been hindered by the lack of multilingual training data. In this paper, we present a new data set of 57k annotated utterances in English (43k), Spanish (8.6k) and Thai (5k) across the domains weather, alarm, and reminder. We use this data set to evaluate three different cross-lingual transfer methods: (1) translating the training data, (2) using cross-lingual pre-trained embeddings, and (3) a novel method of using a multilingual machine translation encoder as contextual word representations. We find that given several hundred training examples in the the target language, the latter two methods outperform translating the training data. Further, in very low-resource settings, multilingual contextual word representations give better results than using cross-lingual static embeddings. We also compare the cross-lingual methods to using monolingual resources in the form of contextual ELMo representations and find that given just small amounts of target language data, this method outperforms all cross-lingual methods, which highlights the need for more sophisticated cross-lingual methods.

1 Introduction

One of the first steps in many conversational AI systems that are used to parse utterances in personal assistants is the identification of what the user intends to do (the *intent*) as well as the arguments of the intent (the *slots*) (Mesnil et al., 2013; Liu and Lane, 2016). For example, for a request such as *Set an alarm for tomorrow at 7am*, a first step in fulfilling such a request is to identify that the user's intent is to set an alarm and that the required time argument of the request is expressed by the phrase *tomorrow at 7am*.

Given these properties of the task, the problem can be stated as a joint sentence classification (for intent classification) and sequence labeling (for slot detection) task and therefore naturally lends itself to using a biLSTM-CRF sequence labeling model (Lample et al., 2016; Vu, 2016) where the biLSTM layer is also used as the input for a projection layer for intent detection.

These models are very powerful and given enough training data, they achieve very high accuracy on the intent classification as well as the slot detection task. However, given the requirement of large amounts of labeled training data, developing a conversational AI system for many new languages is a very resource-intensive task and clearly not feasible to be done for the more than 6,500 languages that are currently spoken around the world.

For this reason, one would like to make use of methods that enable transfer learning from a high-resource language to a low-resource language. However, the development of sophisticated cross-lingual transfer methods for intent detection and slot filling has so far been hindered by the lack of multilingual data sets that have been annotated according to the same guidelines.[1] In this work, we therefore present a novel data set containing a large number of English utterances (the high-resource data) as well as a smaller set of utterances in Spanish and in Thai (the low-resource data), which were annotated according to the same annotation scheme. This data allows the systematic in-

---

[*] Work carried out during an internship at Facebook.

[1] Upadhyay et al. (2018) collected such a dataset but to the best of our knowledge, their data is not publicly available.

Proceedings of NAACL-HLT 2019, pages 3795–3805
Minneapolis, Minnesota, June 2 - June 7, 2019.

Domain	Number of utterances English	 Spanish	 Thai	Intent types	Slot types
Alarm	9,282/1,309/2,621	1,184/691/1,011	777/439/597	6	2
Reminder	6,900/943/1,960	1,207/647/1,005	578/336/442	3	6
Weather	14,339/1,929/4,040	1,226/645/1,027	801/460/653	3	5
Total	30,521/4,181/8,621	3,617/1,983/3,043	2,156/1,235/1,692	12	11

Table 1: Summary statistics of the data set. The three values for the number of utterances correspond to the number of utterances in the training, development, and test splits. Note that the slot type *datetime* is shared across all three domains and therefore the total number of slot types is only 11.

vestigation of cross-lingual transfer learning methods from high-resource languages to low-resource languages.

We use this data set to explore different strategies to make use of training data from a high-resource language to improve intent and slot detection models for other languages. We investigate two previously proposed strategies for cross-lingual transfer, namely using cross-lingual pre-trained embeddings (XLU embeddings; see Ruder et al., 2017 for a review) as well as automatically translating the English training data to the target language. Further, we present a novel technique that uses a bidirectional neural machine translation encoder as cross-lingual contextual word representations. We compare the cross-lingual transfer methods to models that are only trained on the target language data.

Across the two languages and the various transfer methods, we find that joint training on the high-resource and the low-resource target language improves results on the target language. We further find that the optimal choice of transfer method depends on the size of the available training data in the target language: In the zero-shot case where no target language data is available, translating the training data gives the best results. However, if a small amount of training data is available, we find that jointly training on the high-resource and low-resource data works better than training on translated data.

We release the data at `https://fb.me/multilingual_task_oriented_data`.

2 Data

We originally collected a data set of around 43,000 English utterances across the domains ALARM, REMINDER, and WEATHER. Data collection proceeded in three steps. First, native English speakers were asked to produce utterances for each intent, e.g., provide examples of how they would ask for the weather. In a second step, two annotators would label the intent and the spans corresponding to slots for each utterance. As a third step, if annotators disagreed on the annotation of an utterance, a third annotator who corresponded with the authors of the guidelines adjudicated between the two annotations.

For the Spanish and Thai data, native speakers of the target language translated a sample of the English utterances. These translated utterances were then also annotated by two annotators. For Spanish, if annotators disagreed, a third annotator who was bilingual in Spanish and English adjudicated these disagreements in communication with the guideline authors. Unfortunately, for Thai, we did not have a bilingual speaker available and hence we decided to discard all utterances for which the annotators disagreed. We hope to rectify this for future data collection efforts.

We believe this data presents a great opportunity to investigate cross-lingual semantic models and to the best of our knowledge, this is the first parallel data set for a word tagging task that has been annotated according to the same guidelines across multiple languages.

Table 1 contains several summary statistics of the data set. Note that the percentage of training examples as compared to development and test examples is much higher for the English data than for the Thai and Spanish data. We decided for a more even split for the latter two languages so that we had a sufficiently large data set for model selection and evaluation.

3 NLU models

The intent detection and slot-filling model consists of two parts: It first uses a sentence classification model to identify the domain of the user utterance (in our case, ALARM, REMINDER, or WEATHER), and then uses a domain-specific model to jointly predict the intent and slots. Figure 1 shows the ba-

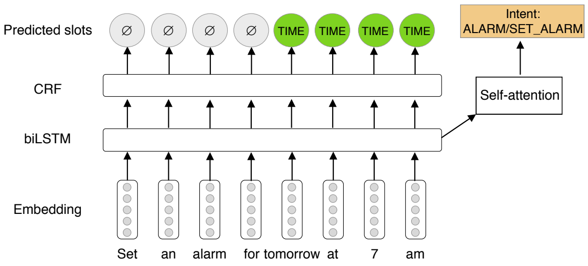


Figure 1: Slot and intent model architecture. Word embeddings are passed through a biLSTM layer which is shared across the slot detection and intent prediction tasks.

sic architecture of the joint intent-slot prediction model. It first embeds the utterance using an embedding matrix and then passes the word vectors to a biLSTM layer. For intent classification, we use a self-attention layer (Lin et al., 2017) over the hidden states of the biLSTM as input to a softmax projection layer; for slot detection, we pass for each word the concatenation of the forward and backward hidden states through a softmax layer, and then pass the resulting label probability vectors through a CRF layer for final predictions.

In our experiments, we vary how the tokens are embedded:

- **Zero embeddings**: We train the parameters of a 0-initialized embedding matrix that contains each word that appears in the training data.

- **XLU embeddings**: We embed the tokens through lookup in a pre-trained cross-lingual embedding matrix and concatenate these embeddings with tuned zero embeddings. Here, we follow Dozat et al. (2017) in having a fixed pre-trained embedding matrix combined with tuneable zero-embeddings.

- **Encoder embeddings**: We embed tokens by passing the entire utterance through a pre-trained biLSTM sentence encoder and using the hidden states of the top layer as input. We keep the parameters of the pre-trained encoder fixed and concatenate them with tuneable zero-embeddings. (See Section 4 for a detailed description of the different encoders.)

4 Encoder models

As mentioned in the previous section, some of our models use a pre-trained biLSTM encoder to generate contextual word embeddings. In all our experiments, we use a bidirectional LSTM encoder with two layers. Overall, we compare three strategies for training these encoders. The motivation for choosing these strategies is to investigate whether there is a benefit of using multilingual embeddings.

- **CoVe**: Following McCann et al. (2017), we train a neural machine translation model to translate from the low-resource language (Spanish or Thai) to English.

- **Multilingual CoVe**: We train a neural machine translation model to translate from the low-resource language to English and from English to the low-resource language. We encode the translation direction using target language-specific start tokens in the decoder (Yu et al., 2018a). In this model, the encoder does not have access to the target language and therefore we expect it to learn to encode phrases with similar meanings into similar vector spaces across languages.

- **Multilingual CoVe w/ autoencoder:** We train a bidirectional neural machine translation model and combine it with an autoencoder objective. For the language pair Spanish-English, that means given a Spanish input sentence we train the model to generate either an English translation or to reproduce the Spanish sentence depending on the start token in the decoder. Likewise, given an English sentence, we train the model to output either a Spanish translation or to reproduce the English sentence depending on the start token in the decoder. The motivation for this approach is that using the joint translation and autoencoder objective might lead to more general representations since the decoder has to be capable to output sentences in either language independent of what the source language was, and unlike in the previous model the source language does not determine the target language. We train an analogous model for the Thai-English language pair.

Spanish	Epoch	es→en	en→es	es→es	en→en
CoVe (unidirectional)	81	8.50	-	-	-
Mult. CoVe	98	8.27	6.90	-	-
Mult. CoVe + autoencoder	282	9.15	7.29	1.15	1.14
Thai	Epoch	th→en	en→th	th→th	en→en
CoVe (unidirectional)	12	13.06	-	-	-
Mult. CoVe	35	12.73	17.00	-	-
Mult. CoVe + autoencoder	92	11.76	16.31	1.12	1.13

Table 2: Perplexities on validation set for different encoder models for the Spanish-English and Thai-English language pairs. A hyphen means that an encoder was not trained for the corresponding language pair.

For Spanish, for which pre-trained ELMo (Peters et al., 2018) embeddings are available, we also evaluate the use of the ELMo embeddings by Che et al. (2018); Fares et al. (2017). Note that the ELMo encoder and the CoVe encoder are trained to encode only the low-resource language and therefore neither of them are multilingual encoders.

Implementation details We train all models using a wrapper around the fairseq (Gehring et al., 2016, 2017) sequence-to-sequence models. We use 300d randomly initialized word vectors as input to the first embedding layer. Each direction in each hidden layer has 512 dimensions which results in a total encoder output dimension of 1024. For the machine translation models, we further use dot-product attention (Luong et al., 2015) and to improve efficiency, we limit the output space of the softmax to 30 translation candidates as determined by word alignments as well as the 2,000 most frequent words (L'Hostis et al., 2016).

Data For the Spanish models, we use two copies[2] of Europarl v7 (Koehn, 2005), every eighth sentences of the Paracrawl data,[3] and the newstest2008-2011 data. For model selection, we use the newstest2012-2013 data. For the Thai models, we use 10 copies of the IWSTL training data (Cettolo et al., 2012) as well as the OpenSubtitles data (Lison et al., 2018) for training, and the IWSTL development and test data for model selection. We use the 20,000 most common words in the training data as the vocabulary. For the multilingual models, we take the union of the vocabulary from both languages. We tokenize the data using an in-house rule-based (for English and Spanish) and dictionary-based (for Thai) tokenizer. We further lowercase all data and remove all duplicates within a data set. We discard all sentences whose length exceeds 100 tokens.

Training details We train the models using stochastic gradient descent with an initial learning rate of 0.5. We decrease the learning rate by 1% after an epoch whenever perplexity on the validation data is higher than for the epoch with the lowest perplexity. We train all models for up to 100 epochs, except for the Spanish bidirectional MT model with an autoencoder which we trained for 300 epochs since it took considerably longer to converge. For multilingual models, we choose the model that has the lowest average perplexity on both translation tasks.

Table 2 shows the perplexities for the different models. In general, the translation perplexities are very similar independent of whether we train a unidirectional MT system or a bidirectional MT system, except for the Spanish bidirectional MT model with an autoencoder which even after 300 epochs still yields higher perplexities on the validation data than the other translation models.[4]

5 Cross-lingual learning

In our first set of experiments, we explore the following baselines and strategies for training models in Spanish and Thai given a large amount of English training data and a small amount of Spanish and Thai training data.

- **Target only**: Using only the low-resource target language data.

---

[2] We upsample the Europarl (for Spanish) and IWSLT (for Thai) data since these data sets are presumably of higher quality than the largerly automatically mined Paracrawl and OpenSubtitles data.

[3] https://paracrawl.eu, the version that was used in the WMT 2018 task

[4] We hypothesize that the slow convergence as well as the lower performance might be caused by the fact that the sentences in the Spanish-English parallel data are much longer than in the Thai-English data which might make it harder to learn good universal sentence representations.

English	Embedding type	Exact match	Domain acc.	Intent acc.	Slot F1
Target only	-	90.91	-	99.11	94.81
Spanish	Embedding type	Exact match	Domain acc.	Intent acc.	Slot F1
Target only	-	72.94	99.43	97.26	80.95
Target only	XLU embeddings	72.90	99.47	96.90	80.99
Target only	CoVe	73.93	99.52	97.43	81.51
Target only	Mult. CoVe	74.13	99.55	97.61	81.64
Target only	Mult. CoVe + auto	73.05	99.51	97.13	81.22
Target only	ELMo	74.81	99.53	96.64	82.96
Translate train	-	72.49	**99.65**	**98.47**	80.60
Cross-lingual	XLU embeddings	75.39	99.52	97.68	83.00
Cross-lingual	CoVe	75.17	99.55	97.81	82.55
Cross-lingual	Mult. CoVe	75.20	99.56	97.82	82.49
Cross-lingual	Mult. CoVe + auto	74.68	99.59	97.90	82.13
Cross-lingual	ELMo	**75.96**	99.47	97.51	**83.38**
Thai	Embedding type	Exact match	Domain acc.	Intent acc.	Slot F1
Target only	-	79.80	99.31	95.13	87.26
Target only	CoVe	84.84	99.36	96.60	90.63
Target only	Mult. CoVe	84.66	99.37	96.75	90.20
Target only	Mult. CoVe + auto	84.79	**99.41**	96.59	90.51
Translate train	-	73.37	99.37	**97.41**	80.38
Cross-lingual	CoVe	84.49	99.29	96.87	90.60
Cross-lingual	Mult. CoVe	85.76	99.39	96.98	91.22
Cross-lingual	Mult. CoVe + auto	**86.12**	99.33	96.87	**91.51**

Table 3: Results using the full training data averaged over 5 training runs. The *translate train* models are trained on the union of translated English and target language data; the *cross-lingual* models are trained on English and target language data.

- **Target only with encoder embeddings**: Using only the low-resource language training data and using pre-trained encoder embeddings.
- **Translate train**: Combining the target training data with the English data that has been automatically translated to the target language. The slot annotations are projected via the attention weights (Yarowsky et al., 2001). We translate the data using the Facebook neural machine translation system.
- **Cross-lingual with XLU embeddings**: Joint training on the English and target language data with pre-trained MUSE (Conneau et al., 2017) cross-lingual embeddings. Since MUSE embeddings are not available for Thai, we only evaluate this method for Spanish.
- **Cross-lingual with encoder embeddings**: Joint training on the English and target language data using pre-trained encoder embeddings.

Implementation details We implement all classification and sequence labeling models within the PyText framework (Aly et al., 2018). We train models for 20 epochs and select the model that performs best on the development set. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.01. We use dropout of 0.3 in the BiLSTM and we set the size of the self-attention layer to 128 dimensions.

Evaluation We evaluate our models according to four metrics: Domain accuracy, which measures the accuracy of the domain classification task; intent accuracy, which measures the accuracy of identifying the correct intent; slot F1, which is the geometric mean of the slot precision and slot recall; and the exact match metric, which indicates the number of utterances for which the domain, intent, and all slots were correctly identified. Exact match is thus the strictest metric of all. We micro-average all metrics across domains.

Results and discussion Table 3 shows the results for all evaluated models. While we get

Spanish	Embedding type	Exact match	Domain acc.	Intent acc.	Slot F1
Translate train	-	**54.95**	**88.70**	**85.39**	**72.87**
Cross-lingual	-	0.63	37.74	36.17	5.50
Cross-lingual	XLU embeddings	4.01	38.24	36.94	17.50
Cross-lingual	CoVe	1.37	39.42	37.13	5.35
Cross-lingual	mult. CoVe	10.56	59.29	53.34	22.50
Cross-lingual	mult. CoVe + auto	9.28	59.25	53.89	19.25
Cross-lingual	ELMo	0.18	35.98	35.36	2.53

Thai	Embedding type	Exact match	Domain acc.	Intent acc.	Slot F1
Translate train	-	**45.59**	**98.11**	**95.85**	**55.43**
Cross-lingual	-	0.20	39.36	39.11	3.44
Cross-lingual	CoVe	5.82	66.75	54.24	8.84
Cross-lingual	mult. CoVe	15.37	73.84	66.35	32.52
Cross-lingual	mult. CoVe + auto	20.84	81.95	70.70	35.62

Table 4: Zero-shot results averaged over 5 training runs. All models were trained only on the English data. In the case of the *translate train* models, the English data was automatically translated into the target language.

slightly different results for the two languages, there are several consistent patterns. For Spanish, we observe that adding contextual word representations to the *target only* model, consistently improves results. Not surprisingly since the ELMo embeddings were trained on a large monolingual corpus, the model that uses these embeddings outperforms all other *target only* models.

If we turn to the cross-lingual models for Spanish, the results indicate that the translation method works well for domain and intent classification but less so for slot detection, presumably due to noisy projection of the slot annotations. For slot detection, we get the best results using the ELMo embeddings which outperform the XLU embeddings as well as the bidirectional MT encoder in terms of exact match and slot F_1. Similarly as in the *target only* setting, the model with multilingual CoVe embeddings combined with the autoencoder performs worse than the other CoVe encoders. Overall, however, the choice of embeddings seems to have only a relatively small impact on the performance of the cross-lingual models. Importantly, however, we see improvements across all metrics as compared to training only on the target language data.

We observe similar results for Thai. The translation approach again yields the worst results for slot detection and we again see a consistent improvement from cross-lingual training as compared to training only on Thai data. And we again only observe small differences depending on the type of embeddings in the cross-lingual training scenario, but in this case, the models with the multilingual CoVe encoders outperform the model with the monolingual encoder.

Table 3 also shows the results for English. Not surprisingly, since we have an order of magnitude more data for English, the model trained and tested on English data still performs better than any of the evaluated methods for the other two languages. However, the gap between the numbers for English and the numbers for the other two languages does get considerably smaller for the cross-lingual models. Prima facie, the results also indicate that the models perform better for Thai than for Spanish. However, this is potentially an artifact of the data. As we mentioned above, we had to discard some of the Thai utterances for which the annotations disagreed with the annotations of their English translations and it is possible that we discarded a disproportionate number of more complex utterances which in return made parsing the Thai utterances easier.

In summary, the results from both languages suggest that pre-trained word representations as well as cross-lingual training improve results over training only on target language data without any pre-trained embeddings. The choice of embeddings, however, seems to matter less, and the overall performance seems to depend only very little on whether we use dynamic or static word representations or whether we use monolingual or bilingual word representations.

However, interestingly, for Spanish for which we compared more types of word representations than for Thai, the cross-lingual model with monolingual ELMo embeddings provided the best results. This potentially indicates that the benefit of cross-lingual training comes from sharing the biLSTM layer or the CRF layer and that embedding the tokens of the high-resource and the low-resource language in a similar space is not as important. At the same time, considering that we are getting relatively good results for both languages if we only train on the target language data, the potential of cross-lingual training might be limited in this case. To investigate whether the embedding type matters in more extreme low-resource scenarios, we also performed a series of zero-shot and low-resource experiments, which we describe in the next section.

6 Zero-shot learning and learning curves

As mentioned in the previous section, from the results on the full data, it is not entirely clear whether there is an advantage of using cross-lingual embeddings. We therefore conducted additional experiments with even smaller training sets in the target language: the case where we no annotated data in the target language exists (zero-shot) and the case where a very limited amount of training data in the target language exists. If there is no advantage of using cross-lingual embeddings over using monolingual embeddings, we expect to see similar results for all models independent of the training data size. On the other hand, if the multilingual CoVe encoder actually embeds phrases with similar meanings in the two languages in a similar vector space, we would expect the model with multilingual CoVe embeddings to perform much better in the zero-shot and very low-resource scenarios than any of the models with monolingual embeddings. Further, we can also investigate whether there is an advantage of translating the training data over other methods in extremely low-resource scenarios.

Experiments We used the same models with the same parameters as in the previous section. In the zero-shot case, we only use English data for training and model selection. For the learning curve experiments, we sample 10, 50, 100, or 200 utterances from each domain for the target language for training and model selection and upsample the target language data so that it roughly matches the size of the English data. For the zero-shot results, we present the average numbers across 5 runs. For the learning curve experiments, since we introduced another random factor by randomly sampling the training and model selection data, we repeat this process 10 times and report the average as well as the minimum and maximum values of the exact match metric for these experiments.

Results and discussion Table 4 shows the zero-shot results. These results indicate that using the multilingual CoVe embeddings works better than not using any encoder embeddings or using monolingual CoVe embeddings. This is true for the sentence-level domain and intent classification tasks as well as for slot detection. The Spanish results also suggest that in the zero-shot case, the multilingual encoder embeddings lead to better results than the XLU embeddings. However, also the models that use cross-lingual embeddings perform very poorly and contrary to the case where we have some training data in the target language, in the zero-shot scenario, the translation method works considerably better than any other of the transfer methods.

The results for different training set sizes are shown in Figure 2. These results generally confirm the patterns that we observed in the experiments with all available training data: cross-lingual training improves the results over training only on the target language (to a much bigger extent when there is much less target language training data available) and using pre-trained word representations leads to further improvements. We further observe that cross-lingual learning seems to lead to much more stable training which can be seen in the much smaller ranges of results as compared to the models trained only on the target language. Also in the extremely low-resource scenarios, the choice of embeddings seems to have very little effect. Lastly, as these plots show, the translation approach works better when there is very little training data available but the performance quickly plateaus and once there are are several hundred target language training examples available, joint training on multiple languages leads to better results.

Considering all results together, we find a consistent advantage of using cross-lingual training across all languages, training set sizes and embedding types. We further observe that the choice of embedding type has little effect as long as some form of pre-trained embeddings is being used. These facts together suggest that the main advantage of cross-lingual training comes from sharing the biLSTM and CRF layer across languages.

Figure 2: Results for different training set sizes. The top and the bottom of the error bars correspond to the highest and lowest value of the exact match metric among the 10 runs.

This is in line with the results by de Lhoneux et al. (2018) who found for cross-lingual training of dependency parsers, sharing the MLP layer for parser decisions improved results for all language pairs that they considered, whereas sharing of lower-level parameters only led to improvements for a limited set of language pairs.

7 Related work

Cross-lingual sequence labeling The task of cross-lingual and multilingual sequence labeling has gained a lot of attention recently. Yang et al. (2017) used shared character embeddings for cross-lingual transfer, and Lin et al. (2018) used shared character and sentence embeddings that were trained in a multitask setting for part-of-speech tagging and named entity recognition. Upadhyay et al. (2018) used cross-lingual embeddings for training multilingual slot-filling systems. Xie et al. (2018) used a similar model for NER but they first “translated” the high-resource training data by replacing each token with the token in the target language that was closest in vector space, and they further used character embeddings and a self-attention mechanism. Yu et al. (2018b) investigated using character-based language models for NER in several languages but did not do any cross-lingual learning. Plank and Agić (2018) used cross-lingual embeddings, projected annotations, and dictionaries for zero-shot cross-lingual part-of-speech tagging.

Cross-lingual sentence representations Recently, there was also a lot of work of using cross-lingual sequence encoders for sentence classifications using either multilingual MT encoders similar to ours (e.g., Eriguchi et al., 2018; Yu et al., 2018a; Singla et al., 2018) or training encoders and then aligning their vector spaces after pretraining (Conneau et al., 2018; Schuster et al., 2019). Even more recently, Lample and Conneau (2019) and Mulcaire et al. (2019) showed that it is also possible to directly train contextual word representations jointly on multiple languages.

Cross-lingual transfer for other tasks Apart from tasks such as slot filling and NER, cross-lingual transfer learning has also been investigated a lot for syntactic tasks, and in particular for part-of-speech tagging and dependency parsing. Early work trained part-of-speech taggers for individual languages and then trained delexicalized dependency parsers (e.g., Zeman and Resnik, 2008; McDonald et al., 2011). Further, a lot of syntactic and semantic parsing models recently successfully incorporated parameter sharing for training parsers in closely related langauges (Duong et al., 2015; Ammar et al., 2016; Susanto and Lu, 2017; Smith et al., 2018; de Lhoneux et al., 2018). In the domain of dialog managers, Mrkšić et al. (2017) and Chen et al. (2018) presented methods for cross-lingual transfer for dialog state tracking.

8 Conclusion and future work

In this paper, we presented a new multilingual intent and slot filling data set for task oriented dialog of around 57,000 utterances and evaluated the performance of different methods for cross-lingual transfer learning, including a novel method using cross-lingual contextual word representations. For both investigated languages, we consistently found that cross-lingual learning improves results as compared to only training on limited amounts of target language data, and our results suggest that the choice of multilingual or monolingual embeddings has only a small impact on the overall performance.

Despite the range of models that we considered in this paper, we only scratched at the surface of possible cross-lingual (embedding) models, and hence there are many future directions of this work. First, except for the Spanish ELMo embeddings, we did not use any character embeddings in any of our experiments or models. This presumably makes sense for the English-Thai transfer learning case since these two languages use different writing systems but given the results by Lin et al. (2018) and Yang et al. (2017), we would expect additional improvements by sharing character embeddings for languages with similar writing systems.

Second, one could try to include a specific learning objective to embed translations into a similar vector space as used by Yu et al. (2018a) and Conneau et al. (2018) for multilingual sentence representations.

As yet another extension, one could combine the approaches of multilingual CoVe embeddings and monolingual ELMo (or BERT, Devlin et al., 2018) embeddings and jointly train an encoder with a language model and an MT objective, which would potentially combine the benefit of training a model on large monolingual corpora while at the same time aligning the vector spaces of the two languages. A similar approach worked well for cross-lingual NLI inference on the XNLI data set (Conneau et al., 2018) as well as for unsupervised machine translation (Lample and Conneau, 2019).

We hope that our data set will facilitate research in these directions and ultimately lead to improved natural language understanding models for low-resource languages.

Acknowledgements

We thank the three anonymous reviewers for their thoughtful comments. We also thank Luke Zettlemoyer and Christopher Manning for valuable feedback throughout this project, and Maria Sumner and Neghesti Tinsew for their help with collecting the annotations and improving their consistency.

References

Ahmed Aly, Kushal Lakhotia, Shicong Zhao, Mrinal Mohit, Barlas Oguz, Abhinav Arora, Sonal Gupta, Christopher Dewan, Stef Nelson-Lindall, and Rushin Shah. 2018. PyText: A seamless path from NLP research to production. ArXiv preprint.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Wenhu Chen, Jianshu Chen, Yu Su, Xin Wang, Dong Yu, Xifeng Yan, and William Yang Wang. 2018. XL-NBT: A cross-lingual neural belief tracking framework. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 414–424.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. In *Proceedings of ICLR 2018*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Holger Schwenk, Ves Stoyanov, Adina Williams, and Samuel R. Bowman. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 2475–2485.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. ArXiv preprint.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*, pages 845–850.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. ArXiv preprint.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A Convolutional Encoder Model for Neural Machine Translation. ArXiv preprint.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. ArXiv preprint.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*, pages 260–270.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. ArXiv preprint.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4992–4997.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. ArXiv preprint.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 799–809.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR 2017*.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. OpenSubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *Proceedings of the 11th edition of the Language Resources and Evaluation Conference (LREC 2018)*, pages 1364–1369.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Proceedings of Interspeech 2016*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421.

Bryan McCann, James Bradbury, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 62–72.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Interspeech 2013*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018)*.

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. ArXiv preprint.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*.

Karan Singla, Dogan Can, and Shrikanth Narayanan. 2018. A multi-task approach to learning multilingual representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 214–220.

Aaron Smith, Bernd Bohnet, and Miryam de Lhoneux. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Raymond Hendy Susanto and Wei Lu. 2017. Neural architectures for multilingual semantic parsing. In *Proceedings of the 55th Annual Meeting ofthe Association for Computational Linguistics (ACL 2017)*, pages 38–44.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2018. (Almost) zero-shot cross-lingual spoken language understanding. In *Proceedings of the IEEE ICASSP 2018*.

Ngoc Thang Vu. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. In *Proceedings of Interspeech 2016*, pages 3250–3254.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 369–379.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of ICLR 2017*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Katherin Yu, Haoran Li, and Barlas Oguz. 2018a. Multilingual seq2seq training with similarity loss for cross-lingual document classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 175–179.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018b. On the strength of character language models for multilingual named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 3073–3077.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Evaluating Coherence in Dialogue Systems using Entailment

Nouha Dziri[*] **Ehsan Kamalloo**[*] **Kory W. Mathewson** **Osmar Zaiane**

Department of Computing Science
University of Alberta
{dziri,kamalloo,korym,zaiane}@cs.ualberta.ca

Abstract

Evaluating open-domain dialogue systems is difficult due to the diversity of possible correct answers. Automatic metrics such as BLEU correlate weakly with human annotations, resulting in a significant bias across different models and datasets. Some researchers resort to human judgment experimentation for assessing response quality, which is expensive, time consuming, and not scalable. Moreover, judges tend to evaluate a small number of dialogues, meaning that minor differences in evaluation configuration may lead to dissimilar results. In this paper, we present interpretable metrics for evaluating topic coherence by making use of distributed sentence representations. Furthermore, we introduce calculable approximations of human judgment based on conversational coherence by adopting state-of-the-art entailment techniques. Results show that our metrics can be used as a surrogate for human judgment, making it easy to evaluate dialogue systems on large-scale datasets and allowing an unbiased estimate for the quality of the responses.

1 Introduction

Recently, we have witnessed a big success in the capability of computers to seemingly understand natural language text and to generate plausible responses to conversations (Serban et al., 2016; Xing et al., 2017; Sordoni et al., 2015; Li et al., 2016; Serban et al., 2017; Devlin et al., 2018; Radford et al., 2018). A challenging task of building dialogue systems lies in evaluating the quality of their responses. Typically, evaluating goal-oriented dialogue systems is done via human-generated judgment like a task completion test or user satisfaction score (Walker et al., 1997; Möller et al., 2006). However, the task of evaluating open-ended dialogue systems is not well defined as there is no clear explicit goal for conversations. Indeed, dialog systems are ultimately created to satisfy the user's need which can be associated with how entertaining and engaging the conversation was. It is unclear how to define a metric that can account comprehensibly for the semantic meaning of the responses. Moreover, grasping the underlying meaning of text has always been fraught with difficulties, which are essentially attributed to the complexities and ambiguities in natural language. Generally, a good dialogue can be described as an exchange of information that sustain coherence through a train of thoughts and a flow of topics. Therefore, a plausible way to evaluate open-ended dialogue systems is to measure the consistency of the responses. For example, a neural dialogue system can respond to the utterance "*Do you like animals?*" by "*Yes, I have three cats*", thereafter replies to "*How many cats do you have*" by "*I don't have cats.*". Here, we can notice that the dialogue system failed to provide a coherent answer and instead generated an inconsistent response.

In this work, we characterize the consistency of dialogue systems as a natural language inference (NLI) (Dagan et al., 2006) problem. In particular, NLI is focused on recognizing whether a hypothesis is inferred from a premise. In dialogue systems, we cast a generated response as the hypothesis and the conversation history as the premise, projecting thus the automatic evaluation into an NLI task. In other words, we propose directly calculable approximations of human evaluation grounded on conversational coherence and affordance by using state-of-the-art entailment techniques. For this purpose, we build a synthesized inference data from conversational corpora. The intuition behind this choice is motivated by the fact that utterances in a human conversation tend to follow a consistent and coherent flow where each utterance can be inferred from the previous interactions. We train the state-of-the-art infer-

---

[*] Equal Contribution

Proceedings of NAACL-HLT 2019, pages 3806–3812
Minneapolis, Minnesota, June 2 - June 7, 2019.

ence models on our conversational inference data and then the learned models are used to evaluate the coherence in a given conversation. Finally, we fare our proposed evaluation method against existing automated metrics. The results highlight the capability of inference models to automatically evaluate dialogue coherence. The source code and the dataset are available at `https://github.com/nouhadziri/DialogEntailment`

2 Related Work

Evaluating open-ended dialogue systems has drawn the attention of several researchers in recent years. Unfortunately, word-overlapping metrics such as BLEU have been shown to correlate weakly with human evaluation, which in turn, introduces bias against certain models (Liu et al., 2016). Many studies have been proposed to improve the quality of automated metrics. In particular, Lowe et al. (Lowe et al., 2017) introduced an automatic evaluation system called ADEM which learns to score responses from an annotated dataset of human responses scores. However, such system is heavily biased towards the training data and struggles with generalization capabilities on unseen datasets. Further, collecting an annotated gold standard of human judgment is very expensive and thus, ADEM is less flexible and extensible. Venkatesh et al. (Venkatesh et al., 2018) introduced a framework for evaluating the quality of the conversations based on topical diversity, coherence, engagement and conversational depth and showed that these metrics conform with human evaluation. However, a big part of their metrics relies on human labels, which makes the evaluation system not scalable. Recently, Welleck et al. (Welleck et al., 2018) investigated the use of NLI models (e.g., ESIM (Chen et al., 2016) and InferSent (Conneau et al., 2017)) to measure consistency in dialogue systems. They built a Dialogue NLI dataset which consists of sentence pairs labeled as entailment, neutral, or contradiction. The utterances are derived from a two-agent persona-based dialogue dataset. To annotate the dataset, they used human annotation from Amazon Mechanical Turk. In this work, we propose a method that employs NLI approaches to detect coherence in dialogue systems. The proposed evaluation procedure does not require human labels, making progress towards scalable and autonomous evaluation systems.

3 Natural Language Inference

Reasoning about the semantic relationship between two utterances is a fundamental part of text understanding. In this setting, we consider inference about entailment as a useful testing bed for the evaluation of coherence in dialogue systems. The success of NLI models[1] allows us to frame automated dialogue evaluation as an entailment problem. More specifically, given a conversation history H and a generated response r, the goal is to understand whether the premise-hypothesis pair (H, r) is entailing, contradictory, or neutral.

3.1 Coherence in Dialogue Systems

The essence of neural response generation models is designed by maximizing the likelihood of the target response given source utterances. Therefore, a dialogue generation task can be formulated as a next utterance prediction problem. In particular, the model predicts a response u_{i+1} given a conversation history $(u_1, ..., u_i)$. One key factor for a successful conversation is having coherence across multiple turns. A machine's response can be considered as incoherent when it contradicts directly its previous utterances or follows an illogical reasoning throughout the whole conversation. Inconsistency can be clearly identified when it corresponds to logical discrepancy between two facts. For example, when you indicate clearly during the conversation that you have cats but when you get asked "*How many cats do you have*", you answer by "*I don't have cats.*". Nevertheless, in general, inconsistency can be less explicitly recognizable as it may describe an error between what the person has said and what she/he truly believes given her/his personality and background information.

To detect dialogue incoherence, we consider two prominent models that have shown promising results in commonsense reasoning: the Enhanced Sequential Inference Model (ESIM) (Chen et al., 2016) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018):

ESIM (Chen et al., 2016): employs a Bi-LSTM model (Graves and Schmidhuber, 2005) to encode the premise and the hypothesis. Also, it explores the effectiveness of syntax for NLI by encoding syntactic parse trees of premise and hypothesis through Tree-LSTM (Zhu et al., 2015). Then, the

---

[1] Recent models have achieved high accuracy in Stanford NLI corpus (Bowman et al., 2015) (90.1%) and GLUE Benchmark (Wang et al., 2018) (86.7%)

input encoding part is followed by a matrix attention layer, a local inference layer, another BiLSTM inference composition layer, and finally a pooling operation before the output layer. We further boost ESIM with by incorporating contextualized word embeddings, namely ELMo (Peters et al., 2018), into the inference model.

BERT (Devlin et al., 2018): exploits a multi-layer Bidirectional Transformers model (Vaswani et al., 2017) to learn pre-trained universal representations of text using only a plain text corpus from Wikipedia. BERT has achieved state-of-the art results on various natural language understanding tasks and has been shown to handle strongly long-range dependencies in text. BERT can be fine-tuned to achieve several tasks by solely adding a small layer to the core model. In this work, we adopted BERT to the task of NLI.

Overall, the goal of the above models is to learn a function G_{NLI} that predicts one of three categories (i.e., entailment, contradiction or neutral) given premise-hypothesis pairs.

4 Inference Corpus for Dialogues

To train the inference models, we build a synthesized dataset geared toward evaluating consistency in dialogue systems. To this end, the Persona-Chat conversational data (Zhang et al., 2018) is used to form the basis of our conversational inference data. The continuity of utterances in human conversation facilitates the use of entailment in the dialogue domain. Typically, when we interact with one another, we tend to reference information from previous utterances to engage with the interlocutor. This is why we build our synthetic inference dataset upon a dialogue corpus. The Persona-Chat corpus is a crowd-sourced dataset where two people converse with each other based on a set of randomly assigned persona. To build an inference corpus, we need to find three different labels (i.e., *entailment*, *contradiction*, and *neutral*). For this purpose, we map an appropriate and on topic response to the *entailment* label. Consequently, the *entailment* instances are derived from the utterances in the conversations. For *contradiction*, grammatically-impaired sentences are constructed by randomly choosing words from the conversation. We also added randomly drawn contradictory instances from the MultiNLI corpus (Williams et al., 2018) to account for meaningful inconsistencies. Finally, random utterances from other conversations or generic responses such as "*I don't know*" comprise the *neutral* instances. Following this approach, we build a corpus of 1.1M premise-hypothesis pairs, namely **InferConvAI**. Table 1 summarizes the statistics of InferConvAI.

	Train	**Dev**	**Test**
#entailment	218.2K	12.2K	1.4K
#neutral	579.5K	28.0K	3.1K
#contradiction	261.9K	9.8K	1.1K
Total	1.1M	50.2K	5.6K

Table 1: Distribution of labels in the InferConvAI corpus.

5 Experiments

In this section, we focus on the task of evaluating the next utterance given the conversation history. We used the following models to generate responses. These models were trained on the conversational datasets, using optimization, until convergence:

- Seq2Seq with attention mechanism (Bahdanau et al., 2015): predicts the next response given the previous utterance using an encoder-decoder model.
- HRED (Serban et al., 2016): extends the Seq2Seq model by adding a context-RNN layer that accounts for contextual information.
- TA-Seq2Seq (Xing et al., 2017): extends the Seq2Seq model by biasing the overall distribution towards leveraging topic words in the response.
- THRED (Dziri et al., 2018): builds upon TA-Seq2Seq model by levering topic words in the response in a multi-turn dialogue system.

The training was conducted on two datasets: OpenSubtitles (Tiedemann, 2012) and Reddit (Dziri et al., 2018). Due to lack of resources, we randomly sampled 6M dialogues as training data from each dataset, 700K dialogues as development data, and 40K dialogues as test data. Each dialogue corresponds to three turn exchanges. To evaluate accurately the quality of the generated responses, we recruited five native English speakers. The judges annotated 150 dialogues from Reddit

Method	Reddit	OpenSubtitles
ESIM + ELMo	0.526	0.455
BERT	**0.553**	**0.498**

Table 2: Accuracy of inference models on InferConvAI.

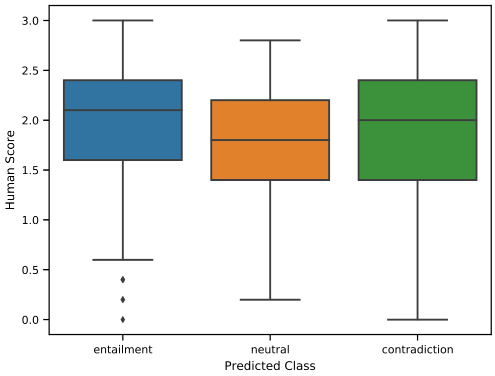


Figure 1: BERT predictions for each class vs. human scores. The labels in the horizontal axis are (from left to right): entailment, neutral, contradiction.

Method	Pearson	
	Reddit	OpenSubtitles
SS(H_{-2})$_{\text{BERT}}$	-0.204	-0.290
SS(H_{-2})$_{\text{ELMo}}$	-0.146	-0.365
SS(H_{-2})$_{\text{USE}}$	-0.248	-0.314
SS(H_{-1})$_{\text{BERT}}$	-0.214	-0.337
SS(H_{-1})$_{\text{ELMo}}$	-0.178	**-0.404**
SS(H_{-1})$_{\text{USE}}$	**-0.287**	-0.320
A$_{\text{BERT}}$	0.135	0.131
A$_{\text{ELMo}}$	0.085	0.162
A$_{\text{word2vec}}$	0.037	0.196
G$_{\text{BERT}}$	0.208	0.132
G$_{\text{ELMo}}$	0.037	0.072
G$_{\text{word2vec}}$	-0.033	0.015
E$_{\text{BERT}}$	0.162	0.144
E$_{\text{ELMo}}$	0.035	0.116
E$_{\text{word2vec}}$	-0.065	0.118

Table 3: The Pearson Correlation between different metrics and human judgments with p-value < 0.001. The semantic similarity (SS) metric is measured with respect to the most recent utterance H_{-1} and the most recent two utterances H_{-2} in the conversation history. We adopt different embedding algorithms to compute the word vectors: ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), word2vec (Mikolov et al., 2013) and Universal Sentence Encoder (USE) (Cer et al., 2018).

and 150 dialogues from OpenSubtitles. All subjects have informed consent as required from the Ethics Review Board at the University of Alberta. Due to lack of space, we will omit an exhaustive description of the human evaluation process and refer readers to (Dziri et al., 2018) as we conducted the same evaluation procedure.

5.1 NLI in Dialogues

In this section, we evaluate the performance of the state-of-the-art entailment models on predicting a score for the generated utterances. In particular, the conversation history H is treated as a hypothesis, whereas the generated response r acts as a premise. We pick two state-of-the-art NLI models (i.e., ESIM (Chen et al., 2016) and BERT (Devlin et al., 2018)). These models were trained on the InferConvAI dataset. During evaluation, we use our test dialogue corpus from Reddit and OpenSubtitles, in which the majority vote of the 4-scale human rating constitutes the labels. The results are illustrated in Table 2. Both models reach reasonable performance in this setting, while BERT outperforms ESIM. Note that this experiment examines the generalization capabilities of these inference models as the test datasets are drawn from an entirely different distribution than the training corpus. Figure 1 illustrates the performance of BERT for each class with respect to the human scores. The test utterances that are predicted as *entailment* tend to be rated higher than other utterances, exhibiting that the entailment models correlate quite well with what humans perceive as a coherent response. Another observation is that the inference models often classify acontextual and off-topic responses as *neutral* and the annotators typically dislike these types of responses. This contributes to the lower scores of *neutral*-detected responses compared to responses predicted as *contradiction*.

5.2 Automated Metrics

5.2.1 Word-level Metrics

We consider as evaluation metrics baselines three textual similarity metrics (Liu et al., 2016) based on word embeddings: Average (A), Greedy (G), and Extrema (E). These word-level embedding metrics have been proven to correlate with human judgment marginally better than other world-overlap metrics (e.g., BLEU, ROUGE and METEOR) (Liu et al., 2016). One critical flaw of these embedding metrics is that they assume that

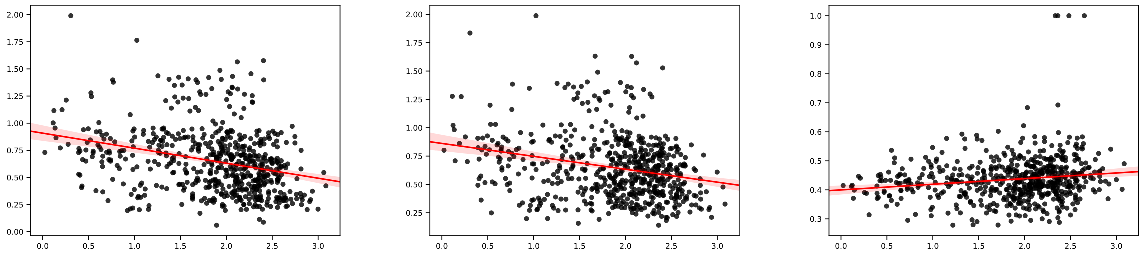


Figure 2: Scatter plots illustrating correlation between human judgment and the automated metrics on the Reddit test dataset. In order to better visualize the density of the points, we added stochastic noise generated by Gaussian distribution $\mathcal{N}(0, 0.1)$ to the human ratings (i.e., horizontal axis) at the cost of lowering correlation, as done in (Lowe et al., 2017). From left to right: SS_{USE} w.r.t. the second most recent utterance (H_{-2}), SS_{USE} w.r.t. the most recent utterance (H_{-1}), and $\text{Extrema}_{\text{BERT}}$

each word is independent of the other words in the sentence. Further, the sentence is treated as a bag-of-words, disregarding words order and dependencies that are known to be substantial for understanding the semantic of a sentence. The correlation of these metrics with human judgment is showcased in Table 3. We can notice that the three metrics A, G and E correlate weakly with human judgment in both datasets, demonstrating the need for a well-designed automated metric that provides an accurate evaluation of dialogues.

5.2.2 Semantic Similarity

The Semantic Similarity (SS) metric was suggested by (Dziri et al., 2018). It measures the distance between the generated response and the utterances in the conversation history. The intuition of this metric revolves around capturing good and consistent responses by showing whether the machine-generated responses maintain the topic of the conversation. In this project, we measured SS with respect to two different utterances, the conversation history H and the most recent utterance H_{-1}. The conversation history is formed by concatenating the two most recent utterances. We report the Pearson coefficient of this metric with human judgment in Table 3. The SS metric is expected to have a negative correlation as the higher human ratings correspond to the lower semantic distance. The results demonstrate that SS metrics correlate better than word-level metrics as they make use of word interactions to represent utterances. Moreover, the Universal Sentence Encoder (USE) (Cer et al., 2018) model performs better on Reddit, whereas the ELMo embeddings achieve higher correlation on OpenSubtitles. This arguably underlines that deep contextualized word representations can manage better complex characteristics of natural language (e.g., syntax and semantics). The SS metric, which requires no pretraining, reaches a Pearson correlation of -0.404 with respect to the most recent utterance on OpenSubtitles. Such correlation can be compared with a correlation of 0.436 achieved by ADEM (Lowe et al., 2017) which required large amounts of training data and computation. Moreover, in order to investigate whether the results in Table 3 are in line with human evaluation, we visualized the correlation between the human ratings and SS metric as scatter plots in Figure 2.

6 Conclusion

Evaluating dialogue systems has been heavily investigated, but researchers are still on the quest for a strong and reliable metric that highly conforms with human judgment. Existing automated metrics show poor correlation with human annotations. In this paper, we present a novel paradigm for evaluating the coherence of dialogue systems by using state-of-the-art entailment techniques. We aim at building a system that does not require human annotation, which in turn, can lead to a scalable evaluation approach. While our results illustrate that the proposed approach correlates reasonably with human judgment and provide an unbiased estimate for the response quality, we believe that there is still room for improvement. For instance, measuring the engagingness of the conversation would be helpful in improving evaluating different dialogue strategies.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly

learning to align and translate. *international conference on learning representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nouha Dziri, Ehsan Kamalloo, Kory W Mathewson, and Osmar Zaiane. 2018. Augmenting neural response generation with context-aware topical attention. *arXiv preprint arXiv:1811.01063*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 994–1003.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1116–1126.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Sebastian Möller, Roman Englert, Klaus Engelbrecht, Verena Hafner, Anthony Jameson, Antti Oulasvirta, Alexander Raake, and Norbert Reithinger. 2006. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *Ninth International Conference on Spoken Language Processing*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*, volume 2012, pages 2214–2218.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, Rahul Goel, Shaohua Yang, and Anirudh Raju. 2018. On evaluating and comparing conversational agents. *arXiv preprint arXiv:1801.03625*.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. Association for Computational Linguistics.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2018. Dialogue natural language inference. *arXiv preprint arXiv:1811.00671*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1112–1122.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generatio. In *AAAI*, pages 3351–3357.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.

On Knowledge Distillation from Complex Networks for Response Prediction

Siddhartha Arora[1,2] and **Mitesh M. Khapra**[1,2] and **Harish G. Ramaswamy**[1,2]
[1]Department of Computer Science and Engineering, Indian Institute of Technology Madras
[2]Robert Bosch Centre for Data Science and AI (RBC-DSAI),
Indian Institute of Technology Madras
`{sidarora,miteshk,hariguru}@cse.iitm.ac.in`

Abstract

Recent advances in Question Answering have lead to the development of very complex models which compute rich representations for query and documents by capturing all pairwise interactions between query and document words. This makes these models expensive in space and time, and in practice one has to restrict the length of the documents that can be fed to these models. Such models have also been recently employed for the task of predicting dialog responses from available background documents (e.g., Holl-E dataset). However, here the documents are longer, thereby rendering these complex models infeasible except in select restricted settings. In order to overcome this, we use standard simple models which do not capture all pairwise interactions, but learn to emulate certain characteristics of a complex teacher network. Specifically, we first investigate the conicity of representations learned by a complex model and observe that it is significantly lower than that of simpler models. Based on this insight, we modify the simple architecture to mimic this characteristic. We go further by using knowledge distillation approaches, where the simple model acts as a student and learns to match the output from the complex teacher network. We experiment with the Holl-E dialog data set and show that by mimicking characteristics and matching outputs from a teacher, even a simple network can give improved performance.

1 Introduction

The advent of large scale datasets for QA has lead to the development of increasing complex neural models with specialized components for (i) encoding the query (ii) encoding the document(s) (iii) capturing interactions between document and query words and (iv) generating/extracting the correct answer span from the given document (Seo et al., 2016; Hu et al., 2017; Yu et al., 2018). While these models give state-of-the-art performance on a variety of datasets, they have very high space and time complexity. This is a concern, and in practice, it is often the case that one has to resort to restricting the maximum length of the input document such that the model can run with reasonable resources (say, a single 12GB Tesla K80 GPU).

Such complex span prediction models are also being adapted for other NLP tasks such as dialog response prediction (Moghe et al., 2018), which is the focus of this work. In particular, we refer to the *Holl-E* dataset where the task is to extract the next response from a document which is relevant to the conversation (see Figure 1). This setup is very similar to QA wherein the input is {context, document} as opposed to {query, document} and the correct response span needs to be extracted from the given document. Given this similarity, it is natural to adopt existing QA models (Seo et al., 2016; Yu et al., 2018) for this task. However, the documents in *Holl-E* dataset are longer, and the authors specifically report that they were unable to run these models when the entire document was given as input. Hence, they report results only in constrained oracle settings where the document is trimmed such that the response still lies in the shortened document. The above situation suggests that there is clearly a trade-off needed. On one hand, we want to harness the power of these complex models to achieve better performance and on the other hand we want to be able to run them with reasonable compute resources without arbitrarily trimming the input document.

This trade-off situation naturally leads to the following question: *Is it possible to build a simple model, with low memory and compute requirements, that copy desirable characteristics from complex models?* To answer this, we start with a relatively simple model with very basic components for encoding query, document and capturing interactions. Once these interactions are cap-

Proceedings of NAACL-HLT 2019, pages 3813–3822
Minneapolis, Minnesota, June 2 - June 7, 2019.

Source Doc: ...comes in. As soon as the door is open, the Bride's fist crashes into Vernita's face. A savage fight follows, first with fists, then with knives.... The fight pauses ... At this point Vernita is introduced as a member of the Deadly Vipers, codename Copperhead. ...

Sample Conversation:
Prober (S1): Which is your favourite character in this?
Responder (S2): My favorite character was Copperhead because she was kicking butt.
Prober (S3): Oh my goodness I agree, because the fight with Vernita was the best in the whole movie.
Responder (S4): It's starts off action packed because as soon as the door is open, the Bride's fist crashes into Vernita's face. A savage fight follows, first with fists, then with knives.
Prober (S5): And it gets better when we find out they are both assassins.
Responder (S6): And a group of them, Vernita is introduced as a member of the Deadly Vipers, codename Copperhead.

Figure 1: Sample Conversation from the Holl-E Dataset. Note that the underlined responses directly correspond to spans in the background document.

tured, the model computes a final representation which is then fed to a decoder to predict the correct span in the document. This recipe is very similar to BiDAF (Seo et al., 2016), QANeT (Yu et al., 2018) but the main difference is that these models use much more complex encoder and interaction components to arrive at the final representation. As expected, the performance of this model is poor when compared to BiDAF, QANeT. The aim now is to improve the performance of this model by carefully analysing or learning from complex models. Given that the complex model differs in the manner in which the final representation is computed, one hypothesis is that it learns richer final representations than the simple model. Indeed, on investigation, we found that the final representations learned by complex models are diverse for different inputs (context, document pairs) as compared to the simple model. Based on this insight, we propose a modification to the simple model which increases the diversity of the embeddings, thereby improving the performance.

While this insight obtained by manual investigation is useful, there is clearly scope for learning by exploring other characteristics of the model. One principled way of doing this is to use knowledge distillation (Hinton et al., 2015) where the simple model acts as a student and learns to mimic the probability distributions predicted by a teacher. In other words, instead of simply maximizing the log likelihood of the training data, the simple model now gets additional signals from the teacher which act as hints while training.

Our experiments, using the *Holl-E* dataset show that by (i) improving the conicity (Chandrahas et al., 2018) of the representations learned by the simple model and (ii) mimicking the outputs of the complex teacher model the simple model can give improved performance with fewer compute and memory requirements. In particular, when compared to a standalone simple model the student model shows an improvement of 3.4% (compare SAM-mul-train (LD) and SAM-add-topk (LD) entries in Table 2 and Table 3 respectively).

2 Related Work

Over the past few years neural sequence prediction models which take a question as input and predict the corresponding answer span in a given document have evolved rapidly. Such models have also been adapted for dialog response prediction in the context of the *Holl-E* dataset (Moghe et al., 2018). These models typically differ in the components used for capturing interactions between query and document, capturing interactions between sentences in a document and refining the query/document representation over multiple passes (Shen et al., 2017; Dhingra et al., 2017; Sordoni et al., 2016).

In particular, a co-attention network which computes the importance of every query word w.r.t. every document word and the importance of every document word w.r.t. every query word is an important component in most state of the art models (Hermann et al., 2015; Kadlec et al., 2016; Cao et al., 2016; Xiong et al., 2016; Seo et al., 2016; Gong and Bowman, 2017; Dhingra et al., 2017; Wang et al., 2017; Shen et al., 2017; Trischler et al., 2016; Group and Asia, 2017; Tan et al., 2017; Sordoni et al., 2016). Similarly, some models (Group and Asia, 2017; Seo et al., 2016; Hu et al., 2017) contain a self-attention network which computes the importance of every document word w.r.t. every other document word. In general, the most successful models (for example, BiDAF (Seo et al., 2016), QANeT (Yu et al., 2018)) use a combination of these components which capture all pairwise interactions and are thus computationally very expensive. As a result, in practice, these models are not suitable for longer documents.

We now quickly review existing works which use the idea of knowledge distillation to build compact models (Cheng et al., 2017). For example, Ba and Caruana (2014); Hinton et al. (2015); Lopez-Paz et al. (2016); Chen et al. (2017) train a shallow student network using soft targets (or class probabilities) generated by an expensive teacher instead of the hard targets present in the training data. Romero et al. (2015) extend this idea to train a student model using the intermediate representations learned by the teacher model which act as additional *hints*. This idea of *Knowledge Distillation* has also been tried in the context of pruning networks for multiple object detection (Chen et al., 2017), speech recognition (Wong and Gales, 2016). In the context of reading comprehension or span prediction, Hu et al. (2018) have very recently shown that we can distill knowledge from an ensemble of models into a single model. However, unlike our work, the single model itself is a complex model (Hu et al., 2017) containing an expensive self attention network and a RL agent. To the best of our knowledge, ours is the first work which tries to build a simple span prediction model by distilling knowledge from a complex model.

3 Models For Response Prediction

We view a conversation as sequence of utterances by a *prober* and a *responder*. The response prediction (RP) model aims to predict the utterance by the responder based on a source document, when given the query (prober's most recent utterance) and the history (past utterance by the prober and responder). See Figure 1 for an example.

We denote the lengths of source document, query, prober history and responder history as T, I, J, K. The LSTMs/GRUs used all have the same number of cells, denoted by d. In particular, the document length T is of the order of a few thousands and the query/history lengths I, J, K are of the order of a few hundreds. Contrast this with QA tasks, where T is only of the order of a few hundreds, and the query length $(I+J+K)$ is of the order of a few tens.

3.1 BiDAF for RP

BiDAF (Seo et al., 2016) is an extremely popular model used for span prediction in reading comprehension based question answering problems. We can frame the problem of response prediction as one of question answering by concatenating the query, prober history, and responder history into a single "question". BiDAF has proven to be hugely successful in QA tasks, but has a large number of parameters (about 2.5 million) and consumes a large amount of computational space and time during training and prediction.

We use the BiDAF model as a guiding post while constructing our model, and in particular focus on the so called *query to context attention*, which is a vector (denoted by $\widetilde{\mathbf{h}}$) that indicates the weighted sum of the most important words in the source document, with respect to the query and histories.

3.2 QANeT for RP

QANeT (Yu et al., 2018) is another recent model used for span prediction in QA tasks and specifically targets better space and time efficiency than BiDAF. Despite this, it still has a large number of parameters (about 1.3 million) and still consumes a large amount of computational space and time during training and prediction. The QANeT model can also be modified for response prediction in a similar manner to BiDAF.

3.3 Simple Attention Model for RP

We now describe the simple attention model that we aim to learn. In a fashion similar to that of BiDAF and QANeT architectures, the simple model also operates in 3 distinct layers. See Figure 2 for an overview into the model.

3.3.1 Word Embedding Layer

The words from the source document, the utterances by the prober and the responder are all encoded using standard GloVe embeddings (Pennington et al., 2014).

3.3.2 Contextual Embedding Layer

In the next layer we encode the query (prober's most recent utterance) using a BiGRU/BiLSTM, and encode the previous utterances of the prober and responder in a query sensitive manner.

Query Encoder: Embedded query words are passed through BiGRU where final state $q_I \in \mathbb{R}^{2d}$ acts as query representation.

Query Sensitive History Summariser: The history of the prober and responder are passed through a BiGRU to get context sensitive vectors $\mathbf{h}_j^{\mathrm{P}} \in \mathbb{R}^{2d}$ and $\mathbf{h}_k^{\mathrm{R}} \in \mathbb{R}^{2d}$ for $j \in [J]$ and $k \in [K]$.

These vectors are combined to get vectors $\mathbf{h}^{\mathrm{P}}$ and $\mathbf{h}^{\mathrm{R}}$. This process of combining uses the query representation $\mathbf{q}_I$, and hence $\mathbf{h}^{\mathrm{P}}$ and $\mathbf{h}^{\mathrm{R}}$ can

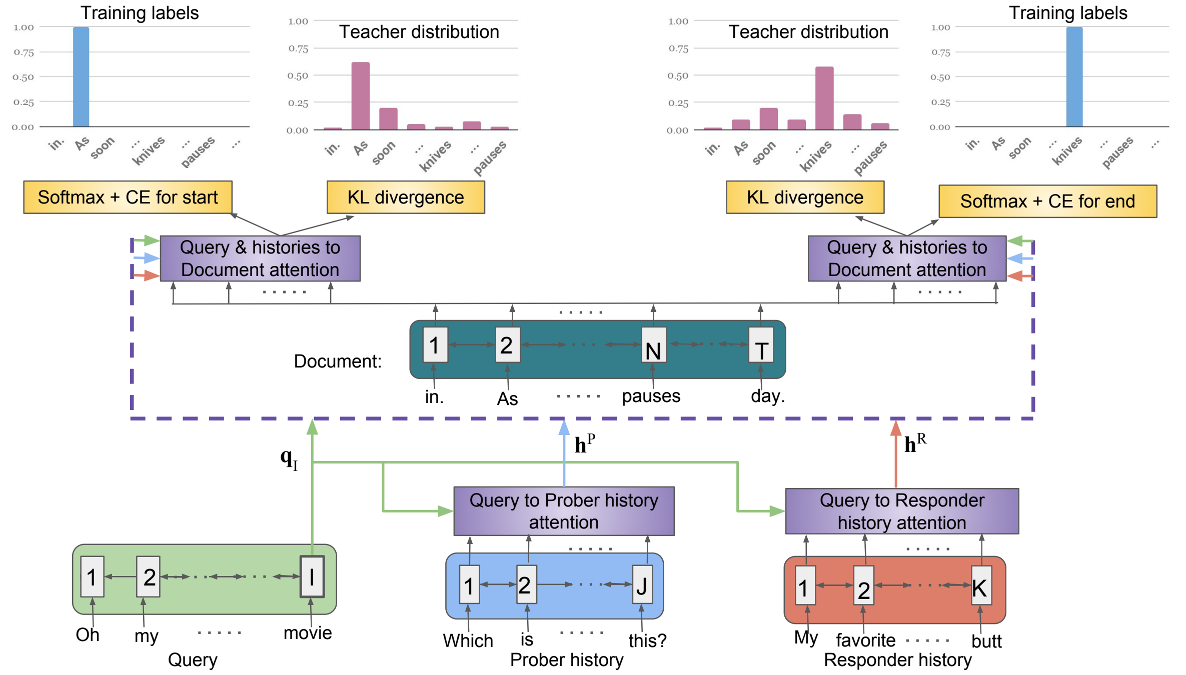


Figure 2: A Simple Attention Model with (1) Query Encoder, Prober History Encoder, Responder History Encoder, and Document Encoder (2) Query to Prober History Attention, Query to Responder History Attention, and Query & histories to Document Attention (3) Training Labels and Teacher Distribution.

be viewed as query-aware representations of the prober and responder history. The equations for $\mathbf{h}^{\text{P}}$ are given below. The vector $\mathbf{h}^{\text{R}}$ is also calculated in a similar manner.

$$e_j = \text{mul}^{W^{\text{P}}}(\mathbf{h}^{\text{P}}_j, \mathbf{q}_I)$$
$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{e})$$
$$\mathbf{h}^{\text{P}} = \sum_j \alpha_j \mathbf{h}^{\text{P}}_j$$

where $\text{mul}^W(\mathbf{v}_0, \mathbf{v}_1) = \mathbf{v}_0^\top W \mathbf{v}_1$ is a parameterized multiplicative way of capturing the interaction between two vectors.

3.3.3 Span Prediction Layer

The source document is finally used in this layer to predict the start and end indices of the response. The GloVe embedded words of the source document are passed through a BiGRU to get context sensitive vectors $\mathbf{u}_t \in \mathbb{R}^{2d}$, for all $t \in [T]$. Each index t gets a score s_t based on the interaction between $\mathbf{u}_t$ and the query/history vectors $\mathbf{q}_I, \mathbf{h}^{\text{P}}, \mathbf{h}^{\text{R}}$. The scores s_t are normalized by a softmax and is taken to be the prediction of the starting word index.

$$s_t = \text{mul}^{W^{\text{strt}}}(\mathbf{u}_t, \mathbf{q}_I, \mathbf{h}^{\text{P}}, \mathbf{h}^{\text{R}})$$
$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{s})$$

where $\text{mul}^W(\mathbf{u}_0, \ldots, \mathbf{u}_a) = \mathbf{u}_0^\top \left(\sum_{i=1}^a W^i \mathbf{u}_i\right)$

A similar method is used for the prediction of the ending word index as well.

4 Bridging The Gap Between Simple and Complex Models

We performed several experiments on the Holl-E dataset and observed that the complex models (QANeT and BiDAF) perform better than the simple attention model described in Section 3.3. However, they take significantly more time and memory for training and inference. In fact, for the examples with longer source documents, both BiDAF and QANeT run into memory issues when training. During prediction, the memory issues in QANeT and BiDAF can be sidestepped by breaking the source document into multiple chunks and taking the highest scoring span.

In the rest of this section we study several approaches to nudge the simple attention model to take parameters that make it have similar behaviour as the complex models, and check if the so nudged model demonstrates better performance on the Holl-E dataset.

4.1 Diversity of Embeddings

We observed that for the simple attention model, the response predictions at different points in the

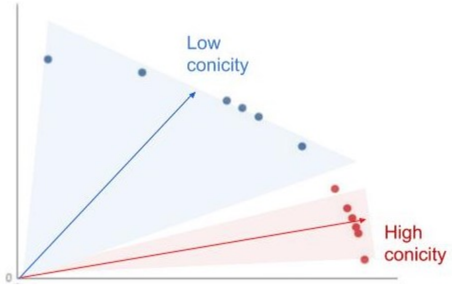


Figure 3: Conicity (*i.e.*, average cosine similarity of vectors with the mean of all the vectors) of two sets of vectors with standard deviation 0.1 (red) and 1.3 (blue)

same conversation are often the same, even though the "right response" is different in those points. For example, consider the conversation in Figure 1. We expect the trained model to be such that

$$\text{Pred\_span}(SD, S_1, S_2, S_3) = \text{span}_{SD}(S_4)$$

where SD is the source document, and S_i are the the utterances in the conversation. Similarly, we expect the trained model to be such that

$$\text{Pred\_span}(SD, S_3, S_4, S_5) = \text{span}_{SD}(S_6)$$

However we often find that our simple model predicts the same span for both the cases above, which is wrong (unless S_4 and S_6 are the same.)

We hypothesize this as being due to the context sensitive embeddings of the history not depending strongly on the query, and hence the span prediction model picks up most information from the source document. To support this point of view we measured the diversity of the context-to-query vectors $\widetilde{\mathbf{h}}$ of the BiDAF model for several examples grouped by conversation. In more detail we computed the *conicity* (Chandrahas et al., 2018) of vectors $\widetilde{\mathbf{h}}(SD, S_1, S_2, S_3), \widetilde{\mathbf{h}}(SD, S_3, S_4, S_5), \ldots$ for every conversation in the test set and averaged it over all conversations. (See Figure 3 for an overview on conicity). This average conicity was observed to be about 0.6 (see Table 6), which, according to Chandrahas et al. (2018), is low (low conicity implies high diversity).

We observe similar behaviour for QANeT as well. The average conicity of the row sums of the *similarity matrix* grouped by conversation was also observed to be about 0.6 (see Table 6).

On the other hand, for our simple attention model, the average conicity of the vectors $\mathbf{h}^{\mathrm{R}}$ and $\mathbf{h}^{\mathrm{P}}$, when computed in a similar fashion as mentioned above were generally high (about 0.8) (see Table 6).

Based on these observations we hypothesize that decreasing the conicity of the vectors $\mathbf{h}^{\mathrm{R}}$ and $\mathbf{h}^{\mathrm{P}}$ would improve the performance of the simple attention model. In particular, we propose to change the multiplicative method of combining vectors into an additive method instead.

In particular we propose to replace the function mul in our simple model with the function add defined as follows:

$$\mathrm{add}^{W}(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_a) = \mathbf{w}^{\top} \tanh\left(\sum_{i=0}^{a} W^i \mathbf{v}_i\right)$$

where the vector $\mathbf{w}$ and the matrices W^i parameterize the mode of combining the input vectors. This is motivated by Chandrahas et al. (2018) who show that using additive model in embedding of entities in knowledge graphs gives consistently better diversity than using multiplicative models.

4.2 Standard Knowledge Distillation from Complex Models

While borrowing high level ideas from complex models, like increasing diversity of the learned representation can help to some extent, one can push this further to distill the learned complex model (Hinton et al., 2015) into the simple attention model. To achieve this, we train a teacher model (BiDAF or QANeT) on the training set and use it to make predictions on the same training set. The simple attention model would minimise the sum of two loss functions: 1) Cross entropy loss of the predicted start and end indices with the train labels of the start and end indices, 2) KL-divergence of the predicted start and end indices from the teacher prediction of the same. The loss on a single training sample is given below

$$\mathrm{D}(\mathbf{p}_{\mathrm{b}}^{\mathrm{T}}||\mathbf{p}_{\mathrm{b}}^{\mathrm{S}}) + \mathrm{D}(\mathbf{y}_b||\mathbf{p}_{\mathrm{b}}^{\mathrm{S}}) + \mathrm{D}(\mathbf{p}_{\mathrm{e}}^{\mathrm{T}}||\mathbf{p}_{\mathrm{e}}^{\mathrm{S}}) + \mathrm{D}(\mathbf{y}_e||\mathbf{p}_{\mathrm{e}}^{\mathrm{S}}) \tag{1}$$

where D denotes the KL divergence, $\mathbf{p}_{\mathrm{b}}^{\mathrm{T}}, \mathbf{p}_{\mathrm{e}}^{\mathrm{T}}$ denote the predicted begin index and end index distribution of the teacher model, and $\mathbf{p}_{\mathrm{b}}^{\mathrm{S}}, \mathbf{p}_{\mathrm{e}}^{\mathrm{S}}$ denote the predicted begin and end index distribution of the student model and $\mathbf{y}_{\mathrm{b}}, \mathbf{y}_{\mathrm{S}}$ denote the true begin and end index in one-hot vector form.

4.3 Top-k Based Knowledge Distillation

Another variant of knowledge distillation is as follows. We do not view the teacher predicted distribution for all indices with importance and just take the top few predicted indices. In particular the loss on a single training sample is given below

$$\mathrm{D}(\widetilde{\mathbf{p}}_{\mathrm{b}}^{\mathrm{T}}||\widetilde{\mathbf{p}}_{\mathrm{b}}^{\mathrm{S}}) + \mathrm{D}(\mathbf{y}_b||\mathbf{p}_{\mathrm{b}}^{\mathrm{S}}) + \mathrm{D}(\widetilde{\mathbf{p}}_{\mathrm{e}}^{\mathrm{T}}||\widetilde{\mathbf{p}}_{\mathrm{e}}^{\mathrm{S}}) + \mathrm{D}(\mathbf{y}_e||\mathbf{p}_{\mathrm{e}}^{\mathrm{S}}) \tag{2}$$

where $\widetilde{\mathbf{p}}_{\mathrm{b}}^{\mathrm{T}}, \widetilde{\mathbf{p}}_{\mathrm{e}}^{\mathrm{T}}$ gives just the (normalised) probability of the top-k predictions of teacher model on the begin and end indices. Similarly $\widetilde{\mathbf{p}}_{\mathrm{b}}^{\mathrm{S}}, \widetilde{\mathbf{p}}_{\mathrm{e}}^{\mathrm{S}}$ gives the student predictions for the begin and end indices restricted to the top-k entries given by the teacher model.

4.4 Other Knowledge Distillation Approaches

As the teacher model is already trained, and the main objective in knowledge distillation is to have the student model mimic the teacher model, there is no need to restrict the objective terms 1 and 3 in equation 1 to only the training data. Hence by hallucinating conversations and documents we can get more terms in the objective and has an effect similar to data augmentation.

Another possible way to take advantage of teacher models is to extract more information than simply the predicted spans for each training example from the teacher models. In particular one easy way to extract piece of information is the gradient of the model output with respect to the input for the teacher model. The so called Sobolev training (Czarnecki et al., 2017) exploits this information and adds two more extra terms to the objective in (1).

$$||\nabla\mathbf{p}_{\mathrm{b}}^{\mathrm{T}} - \nabla\mathbf{p}_{\mathrm{b}}^{\mathrm{S}}||^2 + ||\nabla\mathbf{p}_{\mathrm{e}}^{\mathrm{T}} - \nabla\mathbf{p}_{\mathrm{e}}^{\mathrm{S}}||^2$$

The gradients are all taken with respect to the model input, which would be the source document, the query and the histories.

5 Experiments

In this section, we describe the setup used for our experiments and discuss the results.

5.1 Experiment Setup

We perform experiments using the Holl-E conversation dataset (Moghe et al., 2018) which contains crowdsourced conversations from the movie domain. Every conversation in this dataset is associated with background knowledge comprising of plot details (from Wikipedia), reviews and comments (from Reddit). Every alternate utterance in the conversation is generated by copying and/or modifying sentences from this unstructured background knowledge. We refer the reader again to Figure 1 for a sample from this dataset.

We use the same train, test and validation splits as provided by the authors of the original paper (Moghe et al., 2018). For each chat in the training data, the authors construct training triplets of the form {document, context, response} where the number of train, test and validations triplets are 34486, 4388 and 4318 respectively. The context contains (i) the query (the prober's most recent utterance) and (ii) the history (past 2 utterances by the prober and the responder) as described earlier. The task then is to train a model which can predict the response given the document and the context. At test time, the model is shown document, context and predicts the response.

As mentioned earlier, the authors of Holl-E found that BiDAF and QANeT run into memory issues when evaluated on their dataset. Hence, they propose two setups (i) long document (LD) setup and (ii) short document (SD) setup. In the long document setup, the authors do not trim the document from which the response needs to be predicted. In the short document setup, the authors trim the document to 256 words such that the span containing the response is contained in the trimmed document. This enables them to evaluate BiDAF and QANeT on the trimmed document. We also report experiments using both the LD and SD setup.

As mentioned above complex models (BiDAF and QANeT) face memory issues on training set with long documents. So for all situations where we need predictions from complex models for long documents, we use a BiDAF/QANeT model trained on short document examples, and the prediction on the long document is made by splitting the long documents into chunks and feeding it to the trained BiDAF/QANeT model. The final predicted span is the largest scoring span across all chunks.

For all models, we considered the following hyperparameters and tuned them using the validation set. We tried batch sizes of 32 and 64 and the following GRU sizes: 64, 100, 128. We experimented with 1, 2 and 3 layers of GRU. We used pre-trained publicly available Glove word embeddings [1] of 100 dimensions. The best performance

---

[1] `https://nlp.stanford.edu/projects/glove/`

	SAM, SD	SAM, LD	BiDAF, SD	BiDAF, LD	QANeT, SD	QANeT, LD
Memory	540MB	1.3GB	11GB	11GB	3GB	3GB
Time	30 secs	43 sec	347 secs	710 secs	90 secs	150 secs

Table 1: 'Inference' Memory and Time usage for different models. Here SAM, SD and LD refers to Simple Attention Model, short document and long document respectively.

Model	LD	SD
SAM-mul-train	36.49	40.08
SAM-add-train	36.96	41.30
BiDAF	38.30*	45.54
QANeT	38.10*	47.67

Table 2: Performance (F1 Scores) of different baseline models on SD (short document) and LD (long document) test set.

Model Details	BiDAF	QANeT
SAM-mul	36.50	37.03
SAM-add	37.14	37.28
SAM-add-topk	37.29	**37.73**
SAM-add-aug	37.17	37.01
SAM-add-ensemble	37.30	
SAM-add-both	36.80	

Table 3: F1 Scores for different variants of simple attention model on long documents test set.

was with the batch size of 32, 2 layers of GRU with hidden size 64. We used Adam (Kingma and Ba, 2014) optimizer with initial learning rate set to 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We performed L2 weight decay with decay rate set to 0.001.

5.2 Model Variant Details

The models that we experiment with are listed below:

1. **SAM-add-Train** : The simple attention model with additive interactions and no teacher terms in the objective (Only terms 2 and 4 in Eqn. (1)).
2. **SAM-add-Teach** : The simple attention model with additive interactions and only knowledge distillation terms in the objective (Only terms 1 and 3 in Eqn. (1)).
3. **SAM-add** : The simple attention model with additive interactions and both knowledge distillation terms and training data terms in the objective (all terms in 1).
4. **SAM-add-topk** : The simple attention model with additive interactions and knowledge distillation applied to the top-k indices and training data terms in the objective (all terms in 2).
5. **SAM-add-aug** : The SAM-add model, where the teacher terms are evaluated on hallucinated data in addition to training data. The hallucinated data are derived from the original training set by reordering the words in the source document, query and histories.
6. **SAM-add-grad** : The SAM-add model, with extra terms in the loss penalising the deviation of the gradient of the simple model from the gradient of the teacher model.
7. **SAM-add-both** : Same as the SAM-add model, but has 6 terms instead of the 4 terms in Equation 1. The extra two terms arise from using **both** QANeT and BiDAF instead of just one.
8. **SAM-add-ensemble** : Same as the SAM-add model, but the teacher predictions $\mathbf{p}^{\mathrm{T}}$ are set as the average of the QANeT and BiDAF predictions.

All the "add" models above also have a "mul" variant where the additive interaction add is replaced by a multiplicative interaction mul.

5.3 Results and Discussion

The F1-scores of the various models we train are given in Table 2, Table 3 and Table 4. A summary of the space and time complexity of prediction with the simple model and the complex models is given in Table 1. The training times and parameter counts of the models are given in Table 5 We draw several conclusions and inferences from these results and make some comments below.

Efficient Training with Simple Model: From Table 5, we observe that simple attention model has 5 to 10 times less parameters than QANeT and BiDAF. The training time of the simple model is also significantly lesser than that of the complex models.

Efficient Prediction with Simple Model: From Table 1, we observe that the simple model takes significantly less memory and time during prediction as well. The complex models run out of memory on the large document test set, but a prediction can still be made with a trained BiDAF or QANeT

Model Details	BiDAF	QANeT
SAM-add-teach	20.23	-
SAM-mul	40.81	40.76
SAM-add	42.05	41.89
SAM-add-topk	41.71	42.01
SAM-add-aug	41.65	41.62
SAM-add-ensemble	41.74	
SAM-add-both	**42.32**	
SAM-add-grad	41.37	41.72

Table 4: F1 Scores for different variants of simple attention model on short document test set.

	SAM	BiDAF	QANeT
Parameters	0.25 M	2.5 M	1.3 M
Train Time	307 secs	5880 secs	2213 secs

Table 5: Comparison of parameters (in Million) and training time (seconds) per epoch for different models.

Model	Conicity
Complex Models	0.6
SAM-mul	0.8
SAM-add	0.7

Table 6: Comparison of conicity between variants of simple-attention model and complex models.

model by splitting the source document into manageable chunks.

Conicity of Multiplicative models vs Additive Models: As noted before we use two distinct methods to capture the interaction between a group of vectors : an additive mechanism add, and a multiplicative mechanism mul. As mentioned earlier, an additive model for capturing interactions has been hypothesized to increase diversity. This is true in our case as well: the conicity of the $\mathbf{h}^{\mathrm{P}}$ and $\mathbf{h}^{\mathrm{R}}$ vectors goes down from about 0.8 to 0.7 (compare SAM-mul and SAM-add entries in Table 6) when using the additive model instead of the multiplicative model.

F1 scores of Multiplicative models vs Additive Models: In addition to improving diversity, using the additive model add instead of the multiplicative model mul increases F1-scores all across the board. We have not reported scores for certain multiplicative variants because their performance is significantly worse.

F1 scores of Simple Model with Knowledge Distillation: We observe that using a teacher model for knowledge distillation using the objective in (1) almost always improves the performance of the simple model.

Importance of training labels: The objective in knowledge distillation (Equation (1)) involves both the training labels and the teacher predicted distribution. Even though the teacher predicted distribution also incorporates the training data, removing the training data term from the objective of knowledge distillation worsens performance significantly.

Top-k Distillation: The knowledge distillation approach based on top-k predicted indices results in the best simple model for long document examples (see Table 3). The value of k was chosen to be 50 for the short document case and it is 20 for the long document case.

Add-Both Knowledge Distillation: Learning from multiple teachers could lead to better performance hence we trained the student model with two teachers (BiDAF+QANeT). Here the objective function of student is to minimize KL Divergence between predictions for both teachers. We achieved best results with this technique on short document (SD) test set.

Data Augmentation and Gradient Distillation: While the data augmentation and gradient distillation methods hold a lot of promise, in the experiments that we conducted, we did not see a significant improvement.

QANeT Teachers vs BiDAF Teachers: Using either QANeT or BiDAF as a teacher doesn't seem to make any difference in the performance of the student models (compare the two columns in Table 3 and 4).

6 Conclusion

In this work, we address the trade-off between simple models on one hand which have low memory and compute requirements and complex models on the other hand which give better performance but are computationally expensive. We propose a middle ground by training a simple model to mimic the characteristics of a complex model. In particular, we make observations from a complex model which learns very diverse representations for different inputs and suitably modify the simple model to learn similar diverse representations. We go further, by using knowledge distillation techniques to improve the simple model by training it to match the outputs from the complex model. We experimented with the Holl-E conversation dataset and showed that by mimicking characteristics of the teacher a simple model can give improved performance.

Acknowledgements

We thank Department of Computer Science and Engineering, and Robert Bosch Center for Data Sciences and Artificial Intelligence, IIT Madras (RBC-DSAI) for providing us with adequate compute resources. Lastly, we thank Ananya Sai and Shweta Bhardwaj for valuable discussions and reviewing intial drafts of this paper.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems 27*, pages 2654–2662.

Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. 2016. Attsum: Joint learning of focusing and summarization with neural attention. In *COLING*, pages 547–556. ACL.

Chandrahas, Aditya Sharma, and Partha P. Talukdar. 2018. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 122–131.

Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. 2017. Learning efficient object detection models with knowledge distillation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 742–751. Curran Associates, Inc.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. 2017. Sobolev training for neural networks. *CoRR*, abs/1706.04859.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846.

Yichen Gong and Samuel R. Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *CoRR*, abs/1704.07415.

Natural Language Computing Group and Microsoft Research Asia. 2017. R-net: Machine reading comprehension with self-matching networks. *ACL*, abs/1606.02245.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *CoRR*, abs/1705.02798.

Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. 2018. Attention-guided answer distillation for machine reading comprehension. In *EMNLP*, pages 2077–2086. Association for Computational Linguistics.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

D. Lopez-Paz, B. Schölkopf, L. Bottou, and V. Vapnik. 2016. Unifying distillation and privileged information. In *International Conference on Learning Representations*.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2322–2332.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *In Proceedings of ICLR*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. ReasoNet: Learning to Stop Reading in Machine Comprehension. In *KDD*, pages 1047–1055. ACM.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *CoRR*, abs/1706.04815.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *EMNLP*, pages 128–137. The Association for Computational Linguistics.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 189–198.

Jeremy H. M. Wong and Mark J. F. Gales. 2016. Sequence student-teacher training of deep neural networks. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 2761–2765.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *CoRR*, abs/1804.09541.

Cross-lingual Multi-Level Adversarial Transfer to Enhance Low-Resource Name Tagging

Lifu Huang[1], **Heng Ji**[1], **Jonathan May**[2]
[1] Computer Science Department, Rensselaer Polytechnic Institute
`{huangl7,jih}@rpi.edu`
[2] Information Sciences Institute, University of Southern California
`jonmay@isi.edu`

Abstract

We focus on improving name tagging for low-resource languages using annotations from related languages. Previous studies either directly project annotations from a source language to a target language using cross-lingual representations or use a shared encoder in a multitask network to transfer knowledge. These approaches inevitably introduce noise to the target language annotation due to mismatched source-target sentence structures. To effectively transfer the resources, we develop a new neural architecture that leverages multi-level adversarial transfer: (1) word-level adversarial training, which projects source language words into the same semantic space as those of the target language without using any parallel corpora or bilingual gazetteers, and (2) sentence-level adversarial training, which yields language-agnostic sequential features. Our neural architecture outperforms previous approaches on CoNLL data sets. Moreover, on 10 low-resource languages, our approach achieves up to 16% absolute F-score gain over all high-performing baselines on cross-lingual transfer without using any target-language resources.[1]

1 Introduction

Low-resource language name tagging is an important but challenging task. An effective solution is to perform cross-lingual transfer, by leveraging the annotations from high-resource languages. Most of these efforts achieve cross-lingual annotation projection based on bilingual parallel corpora combining with automatic word alignment (Yarowsky et al., 2001; Wang et al., 2013; Fang and Cohn, 2016; Ehrmann et al., 2011; Ni et al., 2017), bilingual gazetteers (Feng et al., 2017; Zirikly and Hagiwara, 2015), cross-lingual word embedding (Fang and Cohn, 2017; Wang et al., 2017; Huang et al., 2018), or cross-lingual Wikification (Kim et al., 2012; Nothman et al., 2013; Tsai et al., 2016; Pan et al., 2017), but these resources are still only available for dozens of languages. Recent efforts on multi-task learning model each language as one single task while all the tasks share the same encoding layer (Yang et al., 2016, 2017; Lin et al., 2018). These methods can transfer knowledge via the shared encoder without using bilingual resources. However, different languages usually have different underlying sequence structures, as shown in Figure 1. Without an explicit constraint, the encoder is not guaranteed to extract language-independent sequential features. Moreover, when the size of annotated resources is not balanced, the encoder is likely to be biased toward the resource-dominant language.

NED: Sedert het begin^{1} van de Europese integratie^{2} is het mededingingsbeleid^{3} van groot belang^{4} voor de Europese Unie^{5}.

ENG: The European Union^{5}' s competition policy^{3} has been of central importance^{4} since European integration^{2} began^{1}.

ESP: La política de competencia^{3} de la Unión Europea^{5} ha sido de central importancia^{4} desde que se inició^{1} la integración europea^{2}.

Figure 1: Example of parallel sentences between English (ENG), Spanish (ESP) and Dutch (NED) from Europarl Parallel Corpus (Koehn, 2005). The information units with the same color and superscript are aligned.

Considering these challenges, we develop a new neural architecture which can effectively transfer resources from source languages to improve target language name tagging. Our neural architecture is built upon a state-of-the-art sequence tagger: bi-directional long short-term memory as input to conditional random fields (Bi-LSTM-CRF) (Lample et al., 2016; Huang et al., 2015; Ma and

---

[1] Our programs will be released at `https://github.com/wilburOne/AdversarialNameTagger`

Proceedings of NAACL-HLT 2019, pages 3823–3833
Minneapolis, Minnesota, June 2 - June 7, 2019.

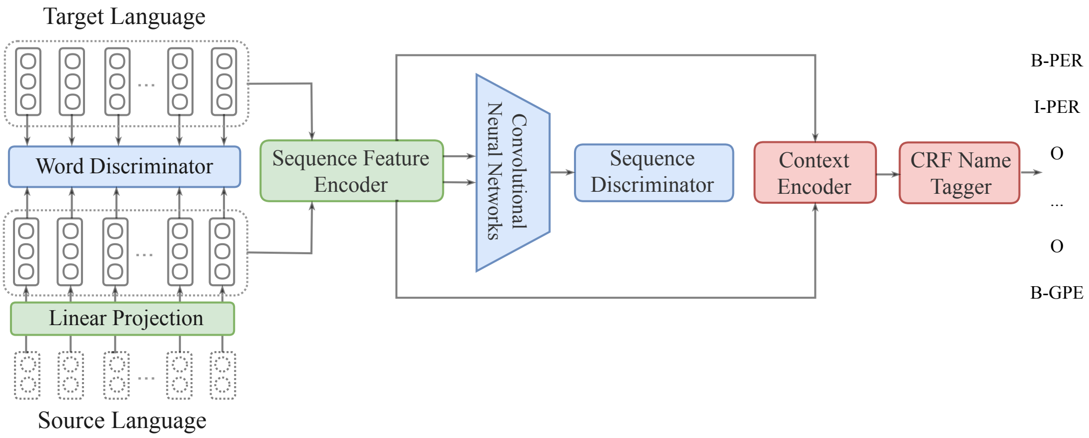


Figure 2: Architecture overview.

Hovy, 2016), integrated with multi-level adversarial transfer: (1) word level adversarial transfer, similar to Conneau et al. (2017), applying a projection function on the source language and a discriminator to distinguish each word of the target language from that of the source language, resulting in a bilingual shared semantic space; (2) sentence-level adversarial transfer, where a discriminator is trained to distinguish each sentence of the target language from that of the source language,[2] and a sequence encoder is applied to each sentence of both languages to prevent the discriminator from correctly predicting the source of each sentence, yielding language-agnostic sequential features. These features can better facilitate the resource transfer from the source language to the target language.

Our contributions are twofold: (1) without requiring any parallel corpora or bilingual gazetteers, the multi-level adversarial approach can efficiently transfer annotated resources from the source language to the target language and improve target language name tagging; (2) In addition to outperforming previous high-performing baselines on CoNLL data sets, we also evaluate cross-lingual name tagging on 10 low-resource languages and achieve up to 16% absolute F-score gain over all baselines when there is no annotated resource for the target language.

2 Approach

2.1 Approach Overview

Figure 2 shows the overview of our neural architecture. It consists of three components:

Cross-lingual word embedding learning with adversarial training: Given pre-trained monolingual word embeddings for a target language t and a source language s, we first apply a mapping function to each word representation from s, then feed both the projected source word representations and the target word representations to a word discriminator to predict the language of each word. If the discriminator cannot distinguish the language of t from the projection of s, then we consider t and the projection of s to be in a shared space.

Language-agnostic sequential feature extraction: For each sentence of t and s, we apply a sequence encoder to extract sequential features, and a Convolutional Neural Network (CNN) (Krizhevsky et al., 2012) based sequence discriminator to predict the language source of each sentence. The sequence encoder is trained to prevent the sequence discriminator from correctly predicting the language of each sentence, such that it finally extracts language-agnostic sequential features.

Language-independent name tagger The language-agnostic sequential features from both t and s are further fed into a context encoder to better capture and refine contextual information and a conditional random field (CRF) (Lafferty et al., 2001) based name tagger.

Next we show the details of each component in our architecture.

2.2 Word-level Adversarial Transfer

To better leverage the resources from the source language, our first step is to construct a shared se-

---

[2] For the name tagging task, 'sequence' always means 'sentence.'

mantic space where the words from the source and target languages are semantically aligned. Without requiring any bilingual gazetteers, recent efforts (Zhang et al., 2017b; Conneau et al., 2017; Chen and Cardie, 2018) explore unsupervised approaches to learn cross-lingual word embeddings and achieve comparable performance to supervised methods. Following these studies, we perform word-level adversarial training to automatically align word representations from s and t.

Formally, assume we are given pre-trained monolingual word embeddings $\mathbf{V}_t = \{\mathbf{v}_1^t, \mathbf{v}_2^t, ..., \mathbf{v}_N^t\} \in \mathbb{R}^{N \times d_t}$ for t, and $\mathbf{V}_s = \{\mathbf{v}_1^s, \mathbf{v}_2^s, ..., \mathbf{v}_M^s\} \in \mathbb{R}^{M \times d_s}$ for s, where $\mathbf{v}_i^t$ and $\mathbf{v}_j^s$ are the vector representations of words w_i^t and w_i^s from t and s, N and M denote the vocabulary sizes, d_t and d_s denote the embedding dimensionality of t and s respectively. We then apply a mapping function f to project s into the same semantic space as t:

$$\widetilde{\mathbf{V}}_s = f(\mathbf{V}_s) = \mathbf{V}_s\mathbf{U} \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{d_s \times d_t}$ is the transformation matrix. $\widetilde{\mathbf{V}}_s \in \mathbb{R}^{M \times d_t}$ are the projected word embeddings for s, and $\Theta_f = \{\theta_f\}$ denotes the set of parameters to be optimized for f.

Similar to Xing et al. (2015), Conneau et al. (2017), and Chen and Cardie (2018), we constrain the transformation matrix $\mathbf{U}$ to be orthogonal with singular value decomposition (SVD) to reduce the parameter search space:

$$\mathbf{U} = \mathbf{A}\mathbf{B}^\top \text{, with } \mathbf{A}\Sigma\mathbf{B}^\top = \text{SVD}(\widetilde{\mathbf{V}}_s\mathbf{V}_s^\top) \tag{2}$$

To automatically optimize the mapping function f without using extra bilingual signals, we introduce a multi-layer perceptron D as a word discriminator, which takes word embeddings of t and projected word embeddings of s as input features and outputs a single scalar. $D(w_i^*)$ represents the probability of w_i^* coming from t. The word discriminator is trained by minimizing the binary cross-entropy loss:

$$\begin{aligned} L_{dis}^w = &- \frac{1}{I_{t;s}} \cdot \sum_{i=0}^{I_{t;s}} \Big(y_i \cdot \log(D(w_i^*)) \\ &+ (1 - y_i) \cdot \log(1 - D(w_i^*)) \Big) , \\ y_i =& \delta_i(1 - 2\epsilon) + \epsilon , \end{aligned}$$

where $\delta_i = 1$ when w_i^* is from t and $\delta_i = 0$ otherwise. $I_{t;s}$ represents the number of words sampled from the vocabulary of t and s together. ϵ is a smoothed value added to the positive and negative labels. $\Theta_{dis} = \{\theta_D\}$ is the parameter set.

The mapping function f and word discriminator D are two adversarial players, thus we flip the word labels and optimize f by minimizing the following loss:

$$\begin{aligned} L_f^w = &- \frac{1}{I_{t;s}} \cdot \sum_{i=0}^{I_{t;s}} \Big((1 - y_i) \cdot \log(D(w_i^*)) \\ &+ y_i \cdot \log(1 - D(w_i^*)) \Big) , \\ y_i =& \delta_i(1 - 2\epsilon) + \epsilon \end{aligned}$$

Following the standard training procedures of deep adversarial networks (Goodfellow et al., 2014), we train the word discriminator and the mapping function successively with stochastic gradient descent (SGD) (Bottou, 2010) to minimize L_{dis}^w and L_f^w. Similar to Conneau et al. (2017), after word-level adversarial training, we also adopt a refinement step to construct a bilingual dictionary for the top-k most frequent words in the source language[3] based on $\widetilde{\mathbf{V}}_s$ and $\mathbf{V}_t$, and further optimize $\mathbf{U}$ with Equation 2 in a supervised way.

2.3 Sentence-level Adversarial Transfer

Once s is projected into the same semantic space as t, we can regard both sentences as coming from one unified language and directly project annotations from s to t. However, name tagging not only relies on word level features, but also on sequential contextual features for entity type classification. Without constraints, the sequence encoder can only extract sequential features for both t and s based on their final training signals while these features are not necessarily beneficial to the target language. Thus, we further design sentence level adversarial transfer to encourage the encoder to extract language-agnostic sequential features.

Given a sentence $x^t = \{w_1^t, w_2^t, ...\}$ from t and a sentence $x^s = \{w_1^s, w_2^s, ...\}$ from s, we first use $\mathbf{V}_t$ and $\widetilde{\mathbf{V}}_s$ to initialize a vector representation for each w_i^t and w_i^s. We also apply a character-based CNN (denoted as CharCNN) (Kim et al., 2016) for each language to compose a word representation from its characters. For each word, we

---

[3] We set k=15,000 in our experiment.

concatenate its word representation and character based representation. Then we feed the sequence of vector representations into a weight sharing Bi-LSTM encoder E to obtain sequential features $\mathbf{H}_t = \{\mathbf{h}_1^t, \mathbf{h}_2^t, ...\}$ and $\mathbf{H}_s = \{\mathbf{h}_1^s, \mathbf{h}_2^s, ...\}$ for x^t and x^s respectively. The parameter set of optimizing both language-dependent CharCNN and the sequence encoder can be denoted as $\Theta_e = \{\theta_{\text{CharCNN}_t}, \theta_{\text{CharCNN}_s}, \theta_E\}$.

Based on these sequential features, we use a sequence discriminator to predict the language source of each sentence. Given a sentence x^* and its sequential features $\mathbf{H} = \{\mathbf{h}_1^*, \mathbf{h}_2^*, ...\}$ from E, we first apply a language-independent CNN with max-pooling to get an overall vector representation for x^*, then feed it into another multi-layer perceptron, $\tilde{D}$, to predict the probability that x^* comes from language t. The sequence discriminator is trained by minimizing the following binary cross-entropy loss:

$$
\begin{aligned}
L_{dis}^x = &- \frac{1}{\tilde{I}_{t;s}} \cdot \sum_{i=0}^{\tilde{I}_{t;s}} \Big(\tilde{y}_i \cdot \log(\tilde{D}(x_i^*)) \\
&+ (1 - \tilde{y}_i) \cdot \log(1 - \tilde{D}(x_i^*)) \Big) , \\
\tilde{y}_i = &\tilde{\delta}_i(1 - 2\eta) + \eta ,
\end{aligned}
$$

where $\tilde{\delta}_i = 1$ if the sentence x_i^* is from t and $\tilde{\delta}_i = 0$ otherwise. $\tilde{I}_{t;s}$ represents the number of sentences sampled from the whole data set of t and s. η is another smoothed value for sequence labels. $\Theta_{\widetilde{dis}} = \{\theta_{\text{CNN}}, \theta_{\tilde{D}}\}$ denotes the parameter set for optimizing the sequence discriminator.

The sequence encoder E and the sequence discriminator $\tilde{D}$ are two adversarial players and E is optimized by trying to fool $\tilde{D}$ to correctly predict the language source of each sentence. Thus we flip the sequence labels and optimize E by minimizing the following loss:

$$
\begin{aligned}
L_e^x = &- \frac{1}{\tilde{I}_{t;s}} \cdot \sum_{i=0}^{\tilde{I}_{t;s}} \Big((1 - \tilde{y}_i) \cdot \log(\tilde{D}(x_i^*)) \\
&+ \tilde{y}_i \cdot \log(1 - \tilde{D}(x_i^*)) \Big) , \\
\tilde{y}_i = &\tilde{\delta}_i(1 - 2\eta) + \eta
\end{aligned}
$$

Algorithm 1 Multi-level Adversarial Training for Improving Target Language Name Tagging

Input: Monolingual pre-trained word embeddings $\mathbf{V}_t$ for target language t, and $\mathbf{V}_s$ for source language s. Annotated sentence set Δ_t for t and Δ_s for related language s.

```
for $iter = 1$ to $word\_epoch$ do
    for $a = 1$ to $word\_dis\_steps$ do
        sample a batch of words $\mathbf{b}_t \sim \mathbf{V}_t, \mathbf{b}_s \sim \mathbf{V}_s$
        $loss = L_{dis}^w([\mathbf{b}_t, f(\mathbf{b}_s)])$
        update $\Theta_{dis}$ to minimize $loss$
    sample a batch of words $\mathbf{b}_t^{'} \sim \mathbf{V}_t, \mathbf{b}_s^{'} \sim \mathbf{V}_s$
    $loss^{'} = L_f^w([\mathbf{b}_t^{'}, f(\mathbf{b}_s^{'})])$
    update $\Theta_f$ to minimize $loss^{'}$
build a parallel dictionary with $\mathbf{V}_t$ and $f(\mathbf{V}_s)$ and refine
    projected word embeddings $\widetilde{\mathbf{V}}_s = f(V_s)$
for $iter = 1$ to $seq\_epoch$ do
    sample a batch of sentences $\tilde{\mathbf{b}}_t \sim \Delta_t, \tilde{\mathbf{b}}_s \sim \Delta_s$
    extract sequential features from $\tilde{\mathbf{b}}_t, \tilde{\mathbf{b}}_s$ with $E$
    $loss = L_{dis}^x([E(\tilde{\mathbf{b}}_t), E(\tilde{\mathbf{b}}_s)])$
    update $\Theta_e, \Theta_{\widetilde{dis}}$ to minimize $loss$
    for $g = 1$ to $seq\_tagger\_steps$ do
        sample a batch of sequences $\tilde{\mathbf{b}}_t^{'} \sim \Delta_t, \tilde{\mathbf{b}}_s^{'} \sim \Delta_s$
        $loss^{'} = L_e^x([E(\tilde{\mathbf{b}}_t^{'}), E(\tilde{\mathbf{b}}_s^{'})]) + L_{crf}([\tilde{\mathbf{b}}_t^{'}, \tilde{\mathbf{b}}_s^{'}])$
        update $\Theta_e, \Theta_c$ to minimize $loss^{'}$
```

2.4 Name Tagger Training

With the language-agnostic sequential features from E, we can directly combine all annotated training data from both t and s to train the name tagger for t. To do so, we feed the sequential features from E to another Bi-LSTM encoder E_c to refine the context information for each token, and use a CRF output layer to render predictions for each token, which can effectively capture dependencies among name tags (e.g., an “inside-organization” token cannot follow a “beginning-person” token).

Specifically, given an input sentence $x = \{w_1, w_2, ...w_n\}$, we extract language-agnostic sequential features with E, and further obtain a new sequence of contextual features $\widetilde{\mathbf{H}} = \{\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, ..., \tilde{\mathbf{h}}_n\}$ with E_c. Then we a apply a linear layer ℓ to further convert each $\tilde{\mathbf{h}}_i$ to a score vector $\mathbf{y}_i$, in which each dimension denotes the predicted score for a tag (the starting, inside or outside of a name mention with a pre-defined entity type). Then we feed the sequence of score vectors $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\}$ into the CRF layer. The score of a sequence of tags $\mathbf{Z} = \{z_1, z_2, ..., z_n\}$ is defined as:

$$
Score(x, \mathbf{Y}, \mathbf{Z}) = \sum_{i=1}^{n} (R_{z_{i-1}, z_i} + Y_{i, z_i})
$$

where R is a transition matrix and $R_{p,q}$ denotes the binary score of transitioning from tag p to tag q.

$Y_{i,z}$ represents the unary score of assigning tag z to the i-th word.

Given the annotated sequence of tags $\mathbf{Z}$, the CRF loss is:

$$L_{crf} = \log \sum_{\mathbf{Z}' \in \tilde{\mathbf{Z}}} e^{Score(x,\mathbf{Y},\mathbf{Z}')} - Score(x, \mathbf{Y}, \mathbf{Z})$$

where $\tilde{\mathbf{Z}}$ is the set of all possible tagging paths. The parameter set for optimizing the name tagger can be denoted as $\Theta_c = \{\theta_{E_c}, \theta_\ell, \theta_{\text{CRF}}\}$.

We jointly optimize the sequence encoder E, the context encoder E_c and the CRF together by minimizing the loss $L' = L_e^x + L_{crf}$, and successively minimize L_{dis}^x and L' with SGD. The end-to-end training for our neural architecture is described in Algorithm 1.

3 Experiment

3.1 Data and Experimental Setup

We evaluate our methods from multiple settings. We first evaluate our architecture on 10 low-resource languages from the DARPA LORELEI project. The annotations are released by the Linguistic Data Consortium (LDC).[4] Each dataset has four predefined name types: person (PER), organization (ORG), location (LOC) and geo-political entity (GPE). For each target low-resource language, we choose a source language if they are from the same language family or use the same script. To show the impact of resource transfer between distinct languages, we also use English as a source language for each target low-resource language. We create the English annotated resource by combining the TAC-KBP 2015 English Entity Discovery and Linking (Ji et al., 2015) data set and the Automatic Content Extraction (ACE2005) data set.[5] To avoid the impact of parameter initialization, we perform 5-fold cross validation. For each experiment, we run twice and get the averaged F-score. Table 1 shows the statistics of each data set.

Language	# of Sents	# of Tokens	# of Names
Amharic (am)	4,770	71,399	3,891
Tigrinya (ti)	5,023	95,364	6,201
Arabic (ar)	4,781	80,715	4,937
Farsi (fa)	3,855	72,629	3,966
Oromo (om)	2,987	52,876	4,985
Somali (so)	3,453	78,400	5,571
Swahili (sw)	4,155	96,902	6,044
Yoruba (yo)	1,599	46,084	2,016
Uyghur (ug)	3,961	60,999	2,575
Uzbek (uz)	11,135	177,816	10,937
English (en)	17,936	388,120	23,938

Table 1: Data set statistics for each low-resource language.

We also evaluate our approach on high-resource languages. We use Dutch (nl) and Spanish (es) data sets from the CoNLL 2002 (Tjong Kim Sang, 2002) shared task as target languages, and use English (en) data from the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) shared task as the source language. All the data sets have four pre-defined name types: PER, ORG, LOC and miscellaneous (MISC). Table 2 shows the statistics of these data sets.

For fair comparison, we use the same pre-trained word embeddings of English, Dutch and Spanish as Lin et al. (2018), while for each low-resource language we train their word embeddings using the documents from their LDC packages with FastText.[6] Table 3 lists the key hyper-parameters we used in our experiments.

3.2 Baselines

We compare our methods with three categories of baseline methods:[7]

- **Monolingual Name Tagging** Using monolingual annotations only, the current state-of-the-art name tagging model is the Bi-LSTM-CRF network (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016).[8]

- **Multi-task Learning** Lin et al. (2018) apply multi-task learning to boost name tagging performance by introducing additional annotations from source languages using a weight sharing context encoder across multiple languages.

- **Language Universal Representations** We apply word adversarial transfer only to project the source language into the same semantic space as the target language, then train the name tagger on the annotations of source and target languages. Word-Adv^1 refers to the approach which is directly trained on the combination of the anno-

---

[4] The annotations are from: am (LDC2016E87), ti (LDC2017E39), ar (LDC2016E89), fa (LDC2016E93), om (LDC2017E27), so (LDC2016E91), sw (LDC2017E64), yo (LDC2016E105), ug (LDC2016E70), uz (LDC2016E29)

[5] The data sets are LDC2015E103 and LDC2006T06

[6] `https://fasttext.cc/`

[7] All the baselines are trained for 100 epochs

[8] For each word, we also combine its word embedding with a CharCNN based representation.

Language	Resource	Train	Dev	Test
English (en)	source language	204,567 (23,499)	51,578 (5,942)	46,666 (5,648)
Dutch (nl)	target language	202,931 (13,344)	37,761 (2,616)	68,994 (3,941)
Spanish (es)	target language	264,715 (18,797)	52,923 (4,351)	51,533 (3,558)

Table 2: CoNLL data set statistics: # of tokens and # of names (between parentheses).

Parameter Name	Value
Monolingual Embedding Size	100
CharCNN Filter Size	25
CharCNN Filter Widths	[2, 3]
LSTM Hidden Size	100
Droupout Rate	0.5
Smoothing Value ϵ for Word Discriminator	0.1
Word Adversarial Training Epochs	5
Smoothing Value η for Sequence Discriminator	0.3
Sequence Adversarial & Name Tagging Training Epochs	60
# of Steps for Sequence Tagging Training	5
Batch Size	20
Initial Learning Rate	0.01
Optimizer	SGD

Table 3: Hyper-parameters.

target (source)	Cross-lingual Word-Adv^1	Multitask Learning	Our Approach
am (ti)	15.19	19.72	**26.86**
ti (am)	16.20	9.06	**29.36**
ar (fa)	1.53	3.52	**13.83**
fa (ar)	2.59	0.91	**11.14**
om (so)	4.66	3.40	**14.14**
so (om)	4.12	2.98	**20.02**
sw (yo)	7.20	5.60	**18.25**
yo (sw)	13.07	6.14	**23.73**

Table 4: Cross-lingual transfer when the target language has no resources (F-score %).

tations, while Word-Adv^2 refers to the baseline that is first trained on the target language annotations and then further tuned on the related language annotations.

3.3 Cross-lingual Transfer with Zero Target Language Annotated Resource

We first evaluate our approach on a cross-lingual transfer setting without using any annotated training data from the target language. We conduct experiments on 8 low-resource languages. Among those, some pairs, such as Amharic (am) and Tigrinya (ti), Oromo (om) and Somali (so), or Yoruba (yo) and Swahili (sw), are from the same language family and are closely related, while some are not, such as Arabic (ar) and Farsi (fa). Since our approach requires some unlabeled sentences from the target language to train the sentence-level discriminator, we entirely remove the annotations from the annotated data set of the target language. Table 4 presents the results.

Our approach significantly outperforms the previous methods on all languages. Specifically, compared with the Word-Adv^1 baseline, which only performs word-level adversarial transfer, our approach achieves 10% absolute F-score gain on average, which demonstrates the effectiveness of the sentence-level adversarial transfer. In addition, compared with Lin et al. (2018), who only apply a shared context-encoder to transfer the knowledge, our approach not only includes a language-sharing encoder, but also performs multi-level adversarial training to encourage the semantic alignment of words from both languages and a sequence encoder to extract language-agnostic sequential features.

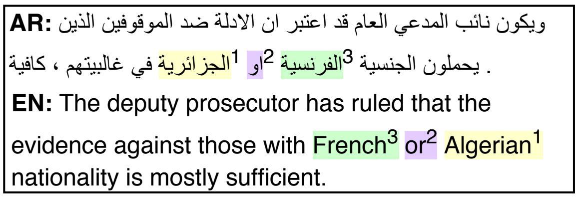
Here we use some Arabic (Farsi) examples to further show the effectiveness of each level of adversarial training in our architecture. Without using any annotated training data from Arabic, both our approach and the Word-Adv^1 baseline successfully identify الفرنسية (*French*) as a *GPE* from the Arabic (ar) sentence in Figure 3, since with word-level adversarial training, the semantics of الفرنسية is well aligned with the *GPE* names in Farsi annotated data, such as فرانسه (*France*), روسیه (*Russia*) and آلمان (*Germany*). However, both the Word-Adv^1 and Lin et al. (2018) baselines fail to identify الجزائرية (*Algerian*) as a *GPE* since its top ranked similar words in Farsi include مذاکرات (*negotiations*), دوحه (*Doha*) and توافقنامه (*agreement*). With sentence-level adversarial training, our approach successfully captures language-agnostic sequential features, such as "او (*or*) *usually connects two names with the same type*", thus our approach successfully identifies الجزائرية (*Algerian*) as a *GPE* name.

3.4 Cross-lingual Transfer for Low-Resource Languages

We also investigate the impact of cross-lingual transfer when the target languages have some annotated resources. For each target low-resource language, we explore the use of a related low-resource language vs. using the high-resource En-

target (related)	Monolingual Bi-LSTM-CRF	Cross-lingual Embedding Word-Adv^1	Word-Adv^2	Multitask Learning	Our Approach Multi-Adversarial
am (ti)	72.23	72.15	72.01	72.35	**73.98**
ti (am)	74.68	74.43	74.83	74.71	**74.93**
ar (fa)	48.92	48.37	47.90	47.53	**49.76**
fa (ar)	64.35	63.93	64.43	63.21	**65.09**
om (so)	76.37	76.43	76.19	76.18	**77.19**
so (om)	77.63	77.31	77.13	77.99	**78.15**
sw (yo)	77.01	77.31	77.85	**77.86**	76.28
yo (sw)	68.97	68.89	69.62	70.12	**70.59**
ug (uz)	68.73	68.53	68.29	68.39	**69.46**
uz (ug)	74.59	74.21	74.74	74.56	**75.37**
am (en)	72.23	72.43	71.63	72.22	**73.35**
ti (en)	74.68	74.61	74.69	74.68	**74.80**
ar (en)	48.92	48.50	47.91	47.40	**50.08**
fa (en)	**64.35**	64.04	64.25	63.44	63.92
om (en)	76..27	76.68	76.53	76.2	**77.29**
so (en)	77.63	76.67	77.88	77.88	**78.21**
sw (en)	77.01	77.52	76.84	**77.89**	77.01
yo (en)	68.97	69.21	69.46	70.43	**70.88**
ug (en)	68.73	68.14	68.79	68.69	**69.06**
uz (en)	74.59	73.95	74.46	74.48	**74.75**

Table 5: Cross-lingual transfer when the target language has resources (F-score %).



AR: ويكون نائب المدعي العام قد اعتبر ان الادلة ضد الموقوفين الذين
يحملون الجنسية الفرنسية^3 او^2 الجزائرية^1 في غالبيتهم ، كافية .

EN: The deputy prosecutor has ruled that the evidence against those with French^3 or^2 Algerian^1 nationality is mostly sufficient.

Figure 3: Example of an Arabic (ar) name tagging output with Farsi (fa) annotated training data only.

glish as our source language. Table 5 shows the performance on 10 low-resource languages.

Comparing cross-lingual embedding based baselines to the monolingual baseline, we observe that for most low-resource languages, directly adding the annotations from the source language to the target language slightly hurts the model. This suggests that when the training data for the target language is not enough, the model will be very sensitive to noise. The multitask learning based baseline (Lin et al., 2018) performs better than the monolingual baseline only when the target and source languages are very close, such as Amharic (am) and Tigrinya (ti), or Swahili (sw) and Yoruba (yo).

By introducing annotated training data from English, the performance of all the baselines becomes worse than the monolingual baseline. Since the script and sequence structure of English is very different from these low-resource languages, the addition of English to the limited target language training data yields a considerably noisy corpus. However, by forcing the sequence encoder to extract language-agnostic features, our approach still achieves better performance than the monolingual baseline for most languages. All of these experiments demonstrate that our approach is more effective in leveraging annotations from other languages to improve target language name tagging.

3.5 Cross-lingual Transfer for High Resource Languages

Language	Model	F-score
Dutch	Lampie et al. (2016)	81.74
	Yang et al. (2017)	85.19
	Lin et al. (2018)	85.71
	Gillick et al. (2016)	82.84
	Word-Adv^1	85.87
	Word-Adv^2	86.43
	Our Model (Bi-LSTM)	**86.87**
Spanish	Lample et al. (2016)	85.75
	Yang et al. (2017)	85.77
	Lin et al. (2018)	85.02
	Gillick et al. (2016)	82.95
	Word-Adv^1	85.92
	Word-Adv^2	85.84
	Our Model (Bi-LSTM)	**86.41**

Table 6: Comparison on cross-lingual transfer for Dutch and Spanish with various baselines: monolingual baseline (Lample et al. (2016)), multitask baselines (Yang et al. (2017) and Lin et al. (2018)), language universal representation baselines (Gillick et al. (2016), Word-Adv^1, Word-Adv^2).

We finally investigate the results when both the source and target languages are all high-resource languages. Table 6 presents the performance on Dutch and Spanish while using English as the source language. Our approach significantly outperforms all the other approaches even when the size of the annotated training data for the target language is huge. We notice that our approach achieves larger improvement on Dutch than Spanish. The reason may be that, compared with Spanish, Dutch is much closer to English (Cutler and Pasveer, 2006). Both English and Dutch are from the same *West Germanic* branch of the *Indo-European* language family while Spanish is from the *Italic* branch.

3.6 Impact of Annotation Size from Source and Target Languages

We use Amharic as the target language and Tigrinya as the source language to show the impact of the size of their annotations. Specifically, to explore the impact of the size of target language annotations, we use 0, 10%, 50%, or 100% annotated training data from Amharic. Similarly, to show the effect of the size of source language annotations, for each experiment, we also gradually add 0, 20%, 50%, or 100% annotated training data from Tigrinya. For all experiments, we use the same dev and test set of Amharic. As Figure 4 shows, as we gradually add annotations from the source or target language, the performance can always be improved. When the size of target language annotations is small, such as 400 sentences, we can achieve 5%-30% F-score gain by adding about 4,000 sentences from the source language. When the size of target language annotations is over 2,000 sentences, the improvement is about 2% if we add in about 4,000 sentences from source language annotations.

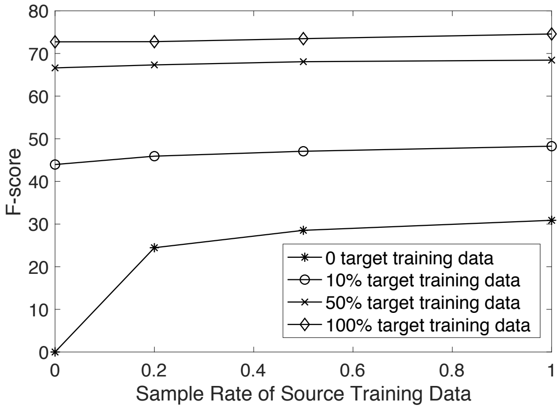


Figure 4: The impact of the size of annotations from source and target languages on Amharic name tagging.

4 Related Work

Name tagging methods based on sequence labeling have been widely studied in recent years. Huang et al. (2015) and Lample et al. (2016) propose an effective Bi-LSTM-CRF architecture; the Bi-LSTM encodes previous and following contexts, and the CRF is used for tag prediction. Other studies incorporate a character-level CNN (Ma and Hovy, 2016), global contexts (Zhang et al., 2018), or language models (Liu et al., 2018; Peters et al., 2017, 2018; Devlin et al., 2018) to improve name tagging. In addition, several approaches (Zhang et al., 2016a, 2017a; Al-Badrashiny et al., 2017) attempt to incorporate hand-crafted linguistic features into a Bi-LSTM-CRF to improve low-resource name tagging performance.

Recent attempts on cross-lingual transfer for name tagging can be divided into two categories: the first projects annotations from a source language to a target language via parallel corpora (Yarowsky et al., 2001; Wang and Manning, 2013; Wang et al., 2013; Zhang et al., 2016b; Fang and Cohn, 2016; Ehrmann et al., 2011; Enghoff et al., 2018; Ni et al., 2017), a bilingual gazetteer (Feng et al., 2017; Zirikly and Hagiwara, 2015), Wikipedia anchor links (Kim et al., 2012; Nothman et al., 2013; Tsai et al., 2016; Pan et al., 2017), and language universal representations, including Unicode bytes (Gillick et al., 2016) and cross-lingual word embeddings (Fang and Cohn, 2017; Wang et al., 2017; Huang et al., 2018; Xie et al., 2018). The second is based on multitask learning via a weight sharing encoder (Yang et al., 2016, 2017; Lin et al., 2018). Compared to these studies, our approach not only automatically learns cross-lingual word embeddings without requiring any parallel resources, but also carefully extracts language-agnostic sequential features, yielding better performance.

Adversarial training has also been extensively studied and applied for cross-lingual and cross-domain transfer. Several studies (Barone, 2016; Zhang et al., 2017c,b; Conneau et al., 2017; Chen and Cardie, 2018) explore adversarial training to automatically induce bilingual and multilingual word representations without using any parallel corpora or bilingual gazetteers. Adversarial training is also applied to extract language-agnostic (Chen et al., 2016; Zou et al., 2018; Wang and Pan, 2018; Kim et al., 2017a; Muis et al., 2018; Cao et al., 2018) and domain-agnostic features (Kim et al., 2017b; Ganin et al., 2016; Tzeng et al., 2017; Chen et al., 2017; Li et al., 2017; Fu et al., 2017; Bousmalis et al., 2016; Shi et al., 2018) for cross-lingual and cross-domain adaptation. Compared with these methods, our approach combines both word-level and sentence-level adversarial training.

5 Conclusions and Future Work

We design a new neural architecture which integrates multi-level adversarial transfer into a Bi-LSTM-CRF to improve low-resource name tagging. With word-level adversarial training, it can automatically project the source language into a shared semantic space with the target language without requiring any comparable data or bilingual gazetteers. Moreover, considering the different underlying sequential structures among various languages, we further design a sentence-level adversarial transfer to encourage the sequence encoder to extract language-agnostic features. The experiments show that our approach achieves the state-of-the-art on both CoNLL data sets and 10 low-resource languages. In the future, we will further explore selecting the feature-consistent annotations from the source language and add to the target language, and explore unsupervised pretrained cross-lingual language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018; Lample and Conneau, 2019) for cross-lingual low resource name tagging.

Acknowledgments

This research is based upon work supported in part by U.S. DARPA LORELEI Program # HR0011-15-C-0115, and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract # FA8650-17-C-9116, and ARL NS-CTA No. W911NF-09-2-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Mohamed Al-Badrashiny, Jason Bolton, Arun Tejasvi Chaganty, Kevin Clark, Craig Harman, Lifu Huang, Matthew Lamm, Jinhao Lei, Di Lu, Xiaoman Pan, et al. 2017. Tinkerbell: Cross-lingual cold-start knowledge base construction. In *Proceedings of TAC 2017*.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. *Proceedings of ACL 2016*.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT 2010*.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of NIPS 2016*.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of EMNLP 2018*, pages 182–192.

Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan Ting Hsu, Jianlong Fu, and Min Sun. 2017. Show, adapt and tell: Adversarial training of cross-domain image captioner. In *Proceedings of ICCV 2017*.

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Proceedings of EMNLP 2018*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Anne Cutler and Dennis Pasveer. 2006. Explaining cross-linguistic differences in effects of lexical stress on spoken-word recognition. In *3rd International Conference on Speech Prosody*. TUD press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of RANLP 2011*, pages 118–124.

Jan Vium Enghoff, Sren Harrison, and Zeljko Agi. 2018. Low-resource named entity recognition via multi-source projection: Not quite there yet? In *The 4th Workshop on Noisy User-generated Text*.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. *Proceedings of CoNLL 2016*.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of ACL 2017*.

Xiaocheng Feng, Lifu Huang, Bing Qin, Ying Lin, Heng Ji, and Ting Liu. 2017. Multi-level cross-lingual attentive neural architecture for low resource name tagging. *Tsinghua Science and Technology*, pages 633–645.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of IJCNLP 2017*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of NAACL-HLT 2016*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS 2014*.

Lifu Huang, Kyunghyun Cho, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Multi-lingual common semantic space construction via cluster-consistent word embedding. *Proceedings of EMNLP 2018*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. In *Proceedings of TAC 2015*.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017a. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of EMNLP 2017*.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of ACL 2012*, pages 694–702.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017b. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of ICML 2017*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI 2016*, pages 2741–2749.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS 2012*, pages 1097–1105.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT 2016*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of IJCAI 2017*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of ACL 2018*, volume 1, pages 799–809.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI 2018*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL 2016*.

Aldrian Obaja Muis, Naoki Otani, Nidhi Vyas, Ruochen Xu, Yiming Yang, Teruko Mitamura, and Eduard Hovy. 2018. Low-resource cross-lingual event type detection via distant supervision with minimal effort. In *Proceedings COLING 2018*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *Proceedings of ACL 2017*.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, pages 151–175.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of ACL 2017*, pages 1946–1958.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL 2018*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Ge Shi, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao, and Heyan Huang. 2018. Genre separation network with adversarial training for cross-genre relation extraction. In *Proceedings of EMNLP 2018*.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL 2002*, pages 1–4.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL 2003*.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of CoNLL 2016*.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of CVPR 2017*.

Dingquan Wang, Nanyun Peng, and Kevin Duh. 2017. A multi-task learning approach to adapting bilingual word embeddings for cross-lingual named entity recognition. In *Proceedings of IJCNLP 2017*.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of ACL 2013*, pages 1073–1082.

Mengqiu Wang and Christopher D Manning. 2013. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *arXiv preprint arXiv:1310.1597*.

Wenya Wang and Sinno Jialin Pan. 2018. Transition-based adversarial network for cross-lingual aspect extraction. In *Proceedings of IJCAI 2018*, pages 4475–4481.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of EMNLP 2018*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL 2015*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001*.

Boliang Zhang, Di Lu, Xiaoman Pan, Ying Lin, Halidanmu Abudukelimu, Heng Ji, and Kevin Knight. 2017a. Embracing non-traditional linguistic resources for low-resource language name tagging. In *Proceedings of IJCNLP 2017)*, pages 362–372.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016a. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of NAACL 2016*, pages 249–259.

Boliang Zhang, Spencer Whitehead, Lifu Huang, and Heng Ji. 2018. Global attention for name tagging. In *Proceedings of CoNLL 2018*, pages 86–96.

Dongxu Zhang, Boliang Zhang, Xiaoman Pan, Xiaocheng Feng, Heng Ji, and XU Weiran. 2016b. Bitext name tagging for cross-lingual entity annotation projection. In *Proceedings of COLING 2016*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of ACL 2017*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017c. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of EMNLP 2017*.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *Proceedings of ACL 2015*.

Bowei Zou, Zengzhuang Xu, Yu Hong, and Guodong Zhou. 2018. Adversarial feature adaptation for cross-lingual relation classification. In *Proceedings of COLING 2018*.

Unsupervised Extraction of Partial Translations for Neural Machine Translation

Benjamin Marie **Atsushi Fujita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
`{bmarie, atsushi.fujita}@nict.go.jp`

Abstract

In neural machine translation (NMT), monolingual data are usually exploited through a so-called back-translation: sentences in the target language are translated into the source language to synthesize new parallel data. While this method provides more training data to better model the target language, on the source side, it only exploits translations that the NMT system is already able to generate using a model trained on existing parallel data. In this work, we assume that new translation knowledge can be extracted from monolingual data, without relying at all on existing parallel data. We propose a new algorithm for extracting from monolingual data what we call partial translations: pairs of source and target sentences that contain sequences of tokens that are translations of each other. Our algorithm is fully unsupervised and takes only source and target monolingual data as input. Our empirical evaluation points out that our partial translations can be used in combination with back-translation to further improve NMT models. Furthermore, while partial translations are particularly useful for low-resource language pairs, they can also be successfully exploited in resource-rich scenarios to improve translation quality.

1 Introduction

Neural machine translation (NMT) systems usually require a large quantity of high-quality bilingual parallel data for training. However, for most language pairs, we do not have such resources, or only in very small quantities, mainly because they are costly to produce.

On the other hand, monolingual corpora are readily available in large quantity for many languages. Previous work has proposed various strategies to integrate monolingual data into NMT systems and has confirmed their usefulness to improve NMT systems. The so-called *back-translation* of monolingual data (Sennrich et al., 2016) is undoubtedly the most prevalent one. This approach simply uses a target-to-source MT system to translate monolingual data in the target language into the source language. The produced new synthetic parallel corpus can be used together with the original parallel data to increase the size of the training data, and eventually to improve NMT systems significantly and consistently. However, on the source side, the synthetic data only contain data that can be generated by the back-translation system trained on some existing parallel data.

Previous work has also studied the extraction of translation pairs of source and target sentences from monolingual data in their respective languages. They have been shown to be useful to train better statistical machine translation (SMT) systems, especially in low-resource conditions. Existing methods on sentence pair extraction mainly rely on the availability of comparable corpora as the source of accurate sentence pairs (Abdul Rauf and Schwenk, 2011), or on the robustness of SMT against noise (Goutte et al., 2012) because sentence pairs extracted from unrelated monolingual corpora tend to be noisy (Tillmann and Xu, 2009; Marie and Fujita, 2017). Most of them also require pre-trained accurate translation models, those of SMT systems for instance, that we may not have in low-resource conditions. Moreover, unlike SMT, NMT has been shown to deal very poorly with noisy training data and still largely underperforms SMT for low-resource language pairs (Koehn and Knowles, 2017) for which comparable corpora are usually not available. Even without an accurate translation model, we still have the possibility of extracting sentence pairs from unrelated source and target monolingual data. However, this is very challenging since we have no guarantee that there are sentence pairs actually retrievable from a given

Proceedings of NAACL-HLT 2019, pages 3834–3844
Minneapolis, Minnesota, June 2 - June 7, 2019.

Source sentence	der Mann wurde festgenommen .
Partial translation	a **man was arrested** at the scene **.**
Post-processed	UNKPP **man was arrested** UNKPP UNKPP UNKPP **.**

Figure 1: Example of a source sentence in our German monolingual data, its best partial translation found in our English monolingual data by our algorithm, and its post-processed version. The bold tokens translate tokens of the source sentence.

pair of source and target monolingual corpora.

In this work, we assume (i) that a given pair of monolingual corpora contain sentence pairs that are at least *partial translations*, i.e., pairs of source and target sentences containing phrases (sequences of tokens) that are translations of each other and (ii) that such pairs can help train better NMT systems. On these assumptions, we propose a new algorithm that extracts partial translations from trillions of candidate sentence pairs without any supervision. Relying on an unsupervised phrase table, our algorithm identifies phrases in a source sentence that have likely translations in a target sentence. The extracted partial translations often contain unrelated parts besides aligned phrases. Therefore, we also apply a simple but very effective post-processing to make such noisy sentence pairs exploitable for a target-to-source NMT model, as exemplified in Figure 1.

We report on significant improvements in translation quality for two language pairs and under different experimental conditions when using our extracted partial translations to train NMT systems. While our method is especially designed to provide new training data for low-resource language pairs, we also observed significant improvements over a strong NMT system trained on large quantity of parallel data. Furthermore, we demonstrate the complementarity of our approach with back-translation.

2 Extraction of Partial Translations

The whole framework for extracting partial translations is presented in Figure 2. To extract partial translations, we first induce a phrase table that contains phrases in the source language paired with their most probable translations in the target language (Section 2.1). These phrase pairs are collected from the same monolingual data from which we extract partial translations. Given the induced phrase table, we search for sentence pairs that are the most likely partial translations in the monolingual data (Section 2.2). Finally, the extracted sentence pairs are post-processed (Section 2.3).

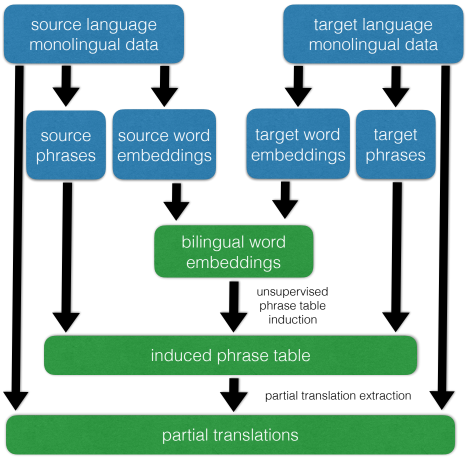


Figure 2: The framework for extracting partial translations from monolingual data.

2.1 Unsupervised Phrase Table Induction

Recent methods addressed the task of finding word translations from monolingual data without any supervision (Lample et al., 2018a; Artetxe et al., 2018a). On the other hand, Marie and Fujita (2018) presented a method for inducing phrase tables from monolingual data using a weakly-supervised framework. To make our approach useful in as many translation tasks as possible, including very low-resource scenarios, we propose a fully unsupervised version of the method in Marie and Fujita (2018). Using phrases instead of only single tokens promotes the extraction of partial translations containing longer sequences of tokens, rather than those with potentially more but discontinuously translated tokens.

Regarding all n-grams of tokens in the monolingual data as phrases and searching the translations for each phrase can be extremely costly. Therefore, we extract meaningful source and target phrases from their respective monolingual data, following the work by Marie and Fujita (2018).[1] Beside extracting phrases, we also train word embeddings on the same source and target monolingual data, independently. Both source and target embedding spaces are aligned in the same space to make word embeddings bilingual, without any supervision (Artetxe et al., 2018a). Using these

---

[1] See Section 3.1 of Marie and Fujita (2018) for the details.

bilingual word embeddings, we compute bilingual phrase embeddings for each of the extracted phrases through the element-wise addition of the embeddings of constituent words of the phrases.

Given the source and target phrase sets, we take their Cartesian product to generate all possible pairs of source and target phrases and compute cosine similarity of each pair using their phrase embeddings. Each pair is also associated with a translation probability (Lample et al., 2018b):

$$p(t_j|s_i) = \frac{\exp\left(\beta \cos(emb(t_j), emb(s_i))\right)}{\sum_k \exp\left(\beta \cos(emb(t_k), emb(s_i))\right)}, \tag{1}$$

where t_j is the j-th phrase in the target phrase list and s_i the i-th phrase in the source phrase list, β a parameter to tune the peakiness of the distribution[2] (Smith et al., 2017), $\cos(\cdot,\cdot)$ cosine similarity between two phrase embeddings, and $emb(\cdot)$ a function returning the bilingual embedding of a given phrase. In practice, to keep the set of phrase pairs to score manageable, we first filter the 300k most frequent phrases in each of the source and target phrase sets.[3]

Retrieved phrase pairs may have a very low translation probability, especially when dealing with distant languages and/or noisy monolingual data. Therefore, we keep only the n-best target phrases for each source phrase, according to Eq. (1). While maintaining the coverage by our phrase table of the source monolingual data the same, we ensure that the phrase translations for each source phrase are the most accurate among all the collected target phrases.

2.2 Partial Translation Extraction

NMT architectures expect parallel sentence pairs as training data, even if we have accurate phrase pairs, they cannot be used directly for training.

Algorithm 1: Partial Translation Extraction

```
Input      : $pt$: a phrase table,
             $M_s$: source monolingual data, and
             $M_t$: target monolingual data
Output     : $P_{\text{translations}}$: a set of the $m_{\text{top}}$-best partial
             translations
Parameters: $m$: the number of target sentence to
             retrieve for each source sentence for the
             PBFD algorithm, and
             $m_{\text{top}}$: the size of $P_{\text{translations}}$
1  $P_{\text{translations}} \leftarrow \{\}$ ;
2  foreach sentence $S$ in $M_s$ do
3      $P_s \leftarrow$ Intersection($pt$, $S$) ;
4      $B_t \leftarrow$ Bow($P_s$, $pt$) ;
5      $T_{\text{candidates}} \leftarrow \{\}$ ;
6      foreach sentence $T$ in $M_t$ do
7          if $\exists w \in T$: $w \in B_t$ then
8              $T_{\text{candidates}} \leftarrow T_{\text{candidates}} \cup T$ ;
9          end
10     end
11     $T_m \leftarrow$ Prune($m$, $T_{candidates}$) ;
12     $T_{\text{best}} \leftarrow$ PBFD($pt$, $S$, $T_m$) ;
13     $P_{\text{translations}} \leftarrow P_{\text{translations}} \cup (T_{\text{best}}, S)$ ;
14 end
15 $P_{\text{translations}} \leftarrow$ GetBest($m_{top}$, $P_{translations}$) ;
16 return $P_{\text{translations}}$
```

Therefore, we propose an algorithm for extracting sentence pairs from the monolingual data that matches the best possible combinations of phrase pairs from the induced phrase table. The pseudocode of this algorithm is presented in Algorithm 1. For each source sentence S (l.2), the algorithm first selects from the phrase table pt all the phrase pairs P_s whose source side appears in S (l.3). It then creates a bag of target words B_t via collecting all the words from the target phrases of P_s (l.4). Subsequently, we keep the m-best target sentences T_m (ll.5–11) according to the following score:

$$F_t(S,T) = \frac{2c_s c_t}{c_s + c_t}, \tag{2}$$

where $c_s = k_t/len(S)$, $c_t = k_t/len(T)$, with S and T respectively a source and a target sentence, k_t the number of tokens in T that are covered by B_t, and $len(\cdot)$ a function that returns the number of tokens in a sentence. With the harmonic mean F_t between c_s and c_t, the algorithm searches for target sentences containing as many words translating source tokens as possible while penalizing the retrieval of very long target sentences that may also contain many tokens having no counterparts in the source sentence.

Then, the algorithm re-ranks the target sentences in T_m with the phrase-based forced decoding (PBFD) algorithm (Zhang et al., 2017) (l.12). PBFD searches for the best combination of phrase pairs covering S and each target sentence in T_m, using the phrase translation probability computed by Eq. (1). However, the original PBFD algorithm penalizes sentence pair with words that are not covered by any phrase pair. It tends to favor very short sentences that are potentially less exploitable for NMT systems, or not even be sentences as it may happen when dealing with noisy

---

[2] We set $\beta = 30$ since it gives consistently good results in our preliminary experiments.

[3] This means that we still have to compute the cosine similarity for 90 billion phrase pairs (300k×300k). This can be done very efficiently on GPU: `https://github.com/facebookresearch/faiss`.

monolingual data.[4] Therefore, we use a slightly modified version which does not penalize uncovered words on the target side in order to favor the extraction of longer target sentences that may contain more translated tokens. Finally, we retain only $P_{\text{translations}}$: the m_{top} sentence pairs with the highest PBFD scores (l.15).

2.3 Post-Processing of Partial Translations

Since the extracted sentence pairs are only partial translations, incorporating them as they are into the training data for NMT may mislead the training of the model due to their noisiness. Since our PBFD algorithm does not penalize target words not covered by any phrase pair, the target side of our partial translations contains longer sentences than the source side, potentially with many words unaligned with any word in the source sentences. Nonetheless, the back-translation approach has proven that NMT can be trained on noisy data at the source side and fluent sentences at the target side. Following this, we use partial translations only to train target-to-source NMT systems, as for back-translated data. It means that the source and target languages of our extraction algorithm becomes respectively the target and source languages of the NMT system.

We want the NMT system to give as much attention as possible to the phrases of the source sentence that have a translation in the target sentence, while ignoring as much as possible the remaining tokens of the source sentence that are likely to be noise, i.e., not translated by the target sentence. Dropping these unaligned tokens is not an appropriate solution since it would produce unlikely sequences of translated tokens. Then, to make the decoder paying its attention to the translated part in the source sentence, we simply replaced all the tokens not covered by the best combination of phrase pairs found by the PBFD algorithm with a made-up token, `UNKPP`.[5]

We expect this post-processing step to particularly well suit the training of a Transformer NMT model (Vaswani et al., 2017), because it can easily learn to pay no attention to `UNKPP` and more attention to correctly translated tokens thanks to the multi-head attention mechanism. Moreover, the Transformer model does not memorize complete sequences and makes time steps independent, unlike recurrent neural networks (RNN). It uses instead positional encodings and has a better ability in linking important features from the entire sequence (Chen et al., 2018), which may make easier the learning from noisy sequences, such as the ones created by introducing `UNKPP`.

We show in Section 4 that using `UNKPP` tokens instead of dropping uncovered tokens leads to a better model. Nonetheless, a better strategy that we leave for future work could be to apply some forced-decoding, with an SMT system for instance, that translates in the source sentence the parts of the target sentence that are not translated, while preserving the partial translations detected by our algorithm in the source sentence.

3 Experiments

We experimented on three language pairs with different degrees of relatedness between the languages of each pair: English→German (en→de), English→Turkish (en→tr), and Bengali→Malay (bn→ms). While our approach is dedicated to improve translation quality for low-resource language pairs, we included the en→de pair for a detailed analysis on the impact of using partial translations in addition to much more training data. bn→ms is expected to be an extremely difficult translation task, because only small quantity of parallel data are available to start with (Section 3.1) and Bengali and Malay are very distant languages which makes very difficult the training of useful unsupervised bilingual word embeddings (Søgaard et al., 2018), which is a key element for inducing the phrase table.

3.1 Data

To train baseline NMT systems, we used for en→de and en→tr 100k parallel sentences randomly extracted from the parallel data provided for the WMT18 News Translation Task,[6] except the ParaCrawl corpus for en→de. For bn→ms, we used the 18k sentence pairs released by the Asian Language Treebank (ALT) project (Riza et al., 2016).[7] As validation and test data for en→de and en→tr, we used Newstest2016 and Newstest2017, respectively, provided by WMT18. For bn→ms,

---

[4] This includes equations, rows of a table, titles, etc.

[5] We made this token different from the usual token reserved for unknown word in the vocabulary, since they are of a different nature.

[6] `http://www.statmt.org/wmt18/translation-task.html`

[7] `http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/`

we used the official development and test data provided by the ALT project.

We chose this amount of training data for en→de and en→tr, since it suits our need for a low-resource translation task while we can still use the available parallel data for further analysis. Moreover, we needed enough parallel data to train an NMT model that can produce useful back-translated data. In our preliminary experiment, we found out that 100k sentence pairs satisfy the minimum amount to train NMT models for useful back-translation. As such, we did not succeed in training useful models for bn→ms, due to the difficulty of the task, but still decided to report on the results to provide insights and matters for future work.

As for monolingual data, we used the English (239M lines) and German (237M lines) NewsCrawl corpora provided by WMT18, and the NewsCrawl and Common Crawl corpora for Turkish (104M lines). We extracted monolingual data ourselves from the Common Crawl project[8] for Bengali (5.3M lines) and Malay (4.6M lines).

3.2 Methods and Systems

To build an unsupervised phrase table, we first extracted 300k most frequent phrases of up to 6 tokens from the entire monolingual data. We also trained 200-dimensional word embeddings on the same data with `fasttext`.[9] For each source phrase, we retained only the 1-best target phrase in the induced phrase table.

The search for partial translations was performed only for a random sample of 1M lines from the monolingual data for each target language. For each of these lines, we searched for the best partial translation with our algorithm in up to 10M lines randomly extracted from the monolingual data for the source language. Then, to maintain a 1:1 ratio with the parallel data, we retained the 100k best partial translations for en→de and en→tr,[10] and 18k best partial translations for bn→ms.

All our NMT systems, including baselines, were the Transformer model (Vaswani et al., 2017) trained with Marian (Junczys-Dowmunt et al., 2018). Note that we fine-tuned the hyperparameters for training on our validation data for each language pair in order to get best possible baseline systems. We then apply the same hyperparameters in all the experiments for the given language pair. To train systems with partial translations (`partial`), we simply mixed them with the original parallel data during training. We also evaluated the systems using back-translated (`backtr`) and copied[11] (`copy`) (Currey et al., 2017) data, separately mixed with the original parallel data. Note that these data were generated from the same target sentences sampled for extracting partial translations: `partial`, `backtr`, and `copy` had the same target side but different source side.[12] Our NMT systems were evaluated with detokenized BLEU-cased.

Training data	en→de	en→tr	bn→ms
baseline	7.1	9.3	**6.1**
`backtr`	9.1	11.4	5.4
`copy`	9.0	11.2	6.0
`partial`	9.9	10.4	5.5
`backtr+copy`	11.3	11.6	5.5
`backtr+partial`	11.5	11.6	4.5
`backtr+copy+partial`	**11.9**	**12.2**	5.8

Table 1: BLEU scores of NMT systems. All the evaluated systems used the parallel data used to train the baseline system, and other synthetic parallel data generated by back-translation (`backtr`), copy (`copy`), or from extracted partial translation (`partial`).

3.3 Results

The results of our experiments are presented in Table 1. For en→de and en→tr, the baseline systems resulted in a poor translation quality below 10 BLEU points. This highlights how critical it is to get more training data to better train an NMT model for low-resource translation tasks.

Adding 100k synthetic parallel sentences generated by back-translation (`backtr`) improved translation quality by 2.0 and 2.1 BLEU points for en→de and en→tr, respectively. Surprisingly, the simplest `copy` method brought improvements similar to `backtr`. Furthermore, we also observed the complementarity of `backtr` and `copy` (`backtr+copy`), with 4.2 and 2.3 BLEU points of improvements for en→de and en→tr, re-

---

[8] `http://commoncrawl.org/`

[9] `https://fasttext.cc/`

[10] The extraction of 100k partial translations from 10^{13} sentence pairs (1M×10M) required around 26 hours of computation using 100 CPUs.

[11] The `copy` approach simply copies the target sentences to the source side. This method surprisingly offers good results in low-resource conditions, and a good complementarity with back-translation, for languages with some orthographic similarity.

[12] With some source sentences that may be identical.

spectively, over the baseline system. To verify that it is not the consequence of just giving more weight to the target monolingual data that may be in-domain, we also trained `backtr+backtr` but did not observe any improvements over `backtr`.

For en→de, the system using our extracted partial translations (`partial`) outperformed `backtr` and `copy` by 0.8 and 0.9 BLEU points, respectively, and the baseline system by 2.8 BLEU points. For en→tr, `partial` also significantly outperformed the baseline system, by 1.1 BLEU points. However, `backtr` and `copy` brought larger improvements. We can explain the difference between en→de and en→tr by the fact that Turkish is more distant from English than German. It makes unsupervised bilingual word embeddings more difficult to train for en→tr and are consequently significantly less accurate (Søgaard et al., 2018). Extraction of accurate and useful partial translations from monolingual data is a more difficult task for en→tr.

While these three kinds of synthetic parallel data, `backtr`, `copy`, and `partial`, present the same target sentences, we found out that mixing all of them with the original parallel led to the best system (`backtr+copy+partial`). This result shows the complementarity of these three datasets thanks to the diversity of source sides generated by different means. For instance, `partial` provides new original translations that were not generated by back-translation.

As expected, bn→ms is a very difficult task for NMT due to the small size of the training data. We were not able to train an NMT model that can generate useful back-translations. The `copy` method was also unhelpful since Bengali and Malay have different writing systems. Our partial translations did not help either, presumably due to the difficulty in unsupervised learning of bilingual word embeddings. Note also that we used much less monolingual data for this language pair to train the word embeddings. This last result is disappointing, but it confirmed that unsupervised bilingual word embeddings are still far from being useful for truly low-resource and distant language pairs.

4 Analysis

4.1 Impact of the Phrase Table

We induced the phrase table from the 300k most frequent source and target phrases. An obvious and actually simpler alternative is the use of words instead of phrases. Using 300k words instead of 300k phrases, for instance, results in a similar cost for phrase table induction but involves a larger vocabulary introducing a better coverage of the monolingual data. However, involving more, and consequently less frequent, words means that it also introduces words for which the word embedding will be noisier.

Training data	en→de	en→tr
baseline	7.1	9.3
1-best target word	8.6	9.9
1-best target phrase	**9.9**	**10.4**
2-best target phrases	7.6	9.5
5-best target phrases	7.5	9.3

Table 2: BLEU scores of NMT systems based on a phrase table induced from source and target single words, or using the 1-best (as in our default configuration), 2-best, or 5-best target phrase translations for each source phrase.

Another important decision that we made when inducing the phrase table was to take only the 1-best target phrase for each source phrase. Involving noisier translations would have a larger impact on our algorithm by inflating the size of B_t that would contain more and noisier target tokens, resulting in unworthily high F_t scores for target sentences containing these tokens that are nonetheless less likely to be the translations of the corresponding source phrase.

Table 2 presents results obtained when using single words instead of phrases to induce the phrase table, and when used the 1-, 2-, or 5-best target phrases for each source phrase. For en→de, using words instead of phrases still led to useful partial translations, with an improvement of 1.5 BLEU points. However, it was 1.3 BLEU points lower than when using phrases. The use of more than one target phrase for each source phrase in the induced phrase table resulted in the retrieval of noisier partial translations that are much less useful to train better NMT models. We observed the similar tendencies for en→tr.

One of the strongest assumption of this work is that partial translations bring translation knowledge complementary to the manually created parallel data used to the train the NMT system. This new knowledge is unbiased by the existing parallel data since we induce the phrase table without using any given parallel data. On the other hand, we can train a phrase table on the parallel data used

Training data	en→de	bn→ms
baseline	7.1	6.1
induced phrase table	**9.9**	5.5
standard phrase table	9.4	**6.3**

Table 3: BLEU scores of NMT systems based on a phrase table induced from monolingual data or based on a standard phrase table trained on the same parallel data also used to train the baseline system.

Training data	en→de	en→tr
baseline	7.1	9.3
`original`	6.2	7.7
`dropped`	8.8	10.0
`partial`	**9.9**	**10.4**

Table 4: BLEU scores of NMT systems trained on partial translations with (`dropped`, `partial`) and without (`original`) post-processing.

to train the NMT system and use it to extract partial translations. Owing to the supervision, we can expect such a phrase table to be much more accurate than the induced phrase table. However, it would introduce a strong bias that encourages the retrieval of partial translations similar to the parallel data that are already used to train the system. Consequently, the extracted partial translations may be less useful than those extracted with an induced phrase table.

To test the above assumption, we trained a standard phrase table on the given parallel data, extracted partial translations using it, and evaluated their impact on the translation quality. Table 3 shows the results for en→de and bn→ms. The results for en→de supports our hypothesis: using a standard phrase table for extracting partial translations has led to a drop of 0.5 BLEU points compared to the use of an induced phrase table. In contrast, for bn→ms, standard phrase table achieved significantly better results than using the induced phrase table and brought a slight improvement over the baseline system. We speculate that the standard phrase table trained only on 18k sentence pairs is not strong enough to bias the extraction of partial translations.

4.2 Impact of Post-Processing

By replacing unaligned tokens, identified by the PBFD algorithm, with a made-up token `UNKPP`, we aimed to guide the decoder to ignore them during training. This section explores the impact of this post-processing, through comparing the translation quality of NMT systems trained on partial translations without any post-processing (`original`), post-processed by removing unaligned source tokens (`dropped`), and our proposed method that replaces unaligned tokens with `UNKPP` (`partial`).

Table 4 presents our results. Without any post-processing, using partial translations brought a significant drop of translation quality of 0.9 BLEU points for en→de, and 1.6 BLEU points for en→tr, from the baseline system trained only on parallel data. This is expected since the partial translations can be very noisy with many unaligned tokens for which the Transformer model will still try to learn a translation in the target sentence. Removing them helps significantly with, for instance for en→de, a 1.7 BLEU points of improvements over the baseline system. However, their removal hides the existence of unaligned tokens and produces sequences of tokens that are unlikely in the source language, presumably misleading the training of the NMT model. Indeed, replacing them with a made-up token further improved the translation quality by 1.1 BLEU points.

4.3 Impact of Noisier Partial Translations

The baseline systems evaluated in Section 3 were trained on 100k parallel data and augmented with the same amount of back-translated sentences and partial translations. Our algorithm retrieved the best partial translation for each one of the 1M source sentences, and then ranked them according to the PBFD score in order to select the most accurate ones. In fact, partial translations at lower rank contained more unaligned tokens, as shown in Figure 3, and also incorrectly aligned tokens. Using these noisier partial translations may disturb the training of the NMT system. In contrast, we can easily increase the quantity of the back-translated data of a similar quality.

To verify this assumption, we evaluated NMT models for en→de and en→tr trained on different quantities of original parallel data, back-translated data, and partial translations.[13] The results are presented in Table 5. As expected, using more back-translated data was much more helpful than using more partial translations. For en→de, in combination with 100k parallel sentences, using

---

[13] We did not oversample the original parallel data to match the size of original and synthetic parallel data, as commonly performed in multilingual NMT (Johnson et al., 2017).

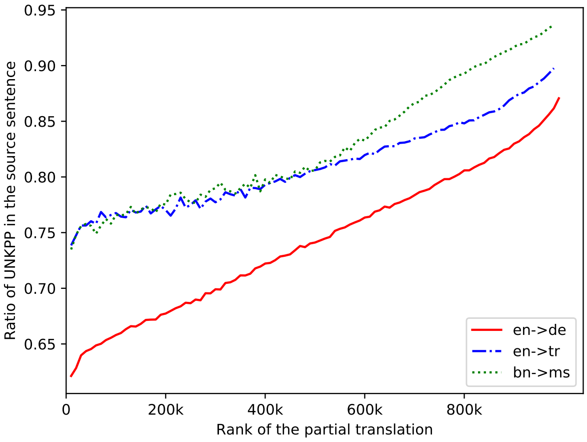


Figure 3: Ratio of `UNKPP` in the source sentence according to the rank of the partial translation. To smooth the curves, we drew the averaged ratio for every batch of 10k consecutively ranked partial translations.

para.	`backtr`	`partial`	en→de	en→tr
100k			7.1	9.3
100k	300k		13.2	13.3
100k		300k	10.8	10.7
100k	300k	300k	**13.6**	**13.7**
100k	1M		16.9	17.9
100k		1M	12.0	11.8
100k	1M	1M	**17.5**	**18.3**
all			26.2	13.6
all	1M		27.7	18.6
all		1M	26.4	14.7
all	1M	1M	**28.2**	**19.0**

Table 5: BLEU scores of NMT systems trained on different combinations of original parallel data (para.), back-translated data (`backtr`), and partial translations (`partial`). "all" denotes the use of all the parallel data provided by the WMT18 News Translation Task: around 5.6M and 207k sentence pairs for en→de and en→tr, respectively.

300k back-translated data achieved better results than using 300k partial translations, with an advantage of 2.4 BLEU points, in contrast to our observation with 100k additional data (Table 1). The gap was even more significant when using 1M additional data: back-translated data achieved 4.9 BLEU points higher than using partial translations. Nonetheless, for en→de, over the configuration using 100k partial translations (9.9 absolute BLEU points), using 300k and 1M partial translations improved by 0.9 and 2.1 BLEU points, respectively. We observed a similar tendency for en→tr. Searching for partial translations in more monolingual data would result in the extraction of a larger number of more accurate partial translations, and presumably help obtain even better NMT models.

Mixing partial translations, back-translated data, and parallel data remains our best configuration. We consistently obtained better results than when using only either partial translations or back-translated data as training data. Even when using the full parallel data of 5.6M sentence pairs and 1M back-translated data[14] for en→de, partial translations still brought an improvement of 0.5 BLEU points. Our best system reached 28.2 BLEU points.[15] This last result confirms that using partial translations as additional training data has also the potential to improve a state-of-the-art NMT system, while it is much more effective in low-resource scenarios.

5 Related Work

There are various methods for extracting sentence pairs from monolingual corpora. However, most of them rely on the availability of document-level information, in comparable corpora for instance, and usually for one specific domain, to efficiently extract accurate sentence pairs (Abdul Rauf and Schwenk, 2011). Other methods extract sentence pairs from completely unrelated monolingual corpora (Tillmann and Xu, 2009; Marie and Fujita, 2017). However, they still rely on an existing accurate translation model trained on large parallel data, introducing a strong bias in the retrieval of sentence pairs. Unlike existing methods, our algorithm for retrieving partial translations is efficient enough to work on large unrelated monolingual data without relying on any document-level information, and also fully unsupervised. Without any bias toward some existing parallel data, it is very suitable for low-resource scenarios.

Previous work has also exploited monolingual data in the target language for improving NMT systems (Sennrich et al., 2016; Currey et al., 2017; Hoang et al., 2018). As demonstrated in this paper, our approach is complementary to previous work, since partial translations can introduce novel information into training. To the best of our knowledge, this is the first work to propose a method for extracting sentence pairs from source and target

---

[14] For this experiment, the NMT system for back-translation was also trained on the full parallel data.

[15] Only 0.1 BLEU points lower than the best reported result at WMT17 for this task. The winning systems used much more back-translated data and an ensemble of several models for decoding.

unrelated monolingual corpora that can be used to train better NMT systems, without requiring any modification of current NMT model architecture.

Wang et al. (2017) proposed a method to train an RNN-based NMT system on partially aligned translations only. However, this method cannot straightforwardly be applied to the state-of-the-art Transformer architecture. In contrast, our proposed method does not assume a particular architecture of NMT, nor requires any modifications of the NMT implementation. In addition, they assume not only that a phrase table is given for their low-resource language pairs, but also that the phrase pairs in the given phrase table are very accurate. We rather focus on augmenting the training data, without assuming phrase pairs of high accuracy. Training NMT only on our extracted partial translations could also be worth investigating.

As confirmed in Section 4.1, the quality of induced phrase table nevertheless affects the usefulness of resulting partial translations. Recent advances in unsupervised MT (Artetxe et al., 2018b; Lample et al., 2018b) have shown that we can obtain phrase tables of better quality through iterating generation of synthetic parallel data and (pseudo-)training of a phrase table on such data. We plan to evaluate whether better phrase tables result in more useful partial translations.

6 Conclusion

We presented a new algorithm for extracting partial translations from unrelated monolingual corpora. Our algorithm is fully unsupervised, i.e., it does not rely on any existing human-made bilingual data, making itself suitable for low-resource language pairs. We demonstrated that very noisy partial translations can be transformed into useful training data for NMT systems with a simple post-processing. While we designed our method specifically for low-resource scenarios, we also showed that partial translations are useful for further improving a state-of-the-art NMT system trained on large parallel data and back-translated synthetic parallel data.

In our future work, we will study the impact of using more partial translations of better quality to train NMT systems. We assume that we can collect better partial translations by searching in more monolingual data. Moreover, we also observed that the top-ranked sentence pairs extracted by our algorithm may be translations of very good quality. We will study the possibility of using such sentence pairs as development data to enable the tuning of unsupervised SMT and NMT systems (Lample et al., 2018b). We will also analyze whether our partial translations are useful because of their noisy nature, since noisy synthetic data have recently been proven useful in some specific configurations (Edunov et al., 2018).

Acknowledgments

We would like to thank the reviewers for their useful comments and suggestions, and Jingyi Zhang for providing the implementation of the phrase-based forced decoding for our experiments. A part of this work was conducted under the program “Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology” of the Ministry of Internal Affairs and Communications (MIC), Japan.

References

Sadaf Abdul Rauf and Holger Schwenk. 2011. Parallel sentence generation from comparable corpora for improved SMT. *Machine Translation*, 25(4):341–375.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3632–3642, Brussels, Belgium. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In

Proceedings of the Second Conference on Machine Translation, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500. Association for Computational Linguistics.

Cyril Goutte, Marine Carpuat, and George Foster. 2012. The impact of sentence alignment errors on phrase-based machine translation performance. In *Proceedings of the 10th Biennial Conference of the Association for Machine Translation in the Americas*, San Diego, USA.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049. Association for Computational Linguistics.

Benjamin Marie and Atsushi Fujita. 2017. Efficient extraction of pseudo-parallel sentences from raw monolingual data using word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 392–398, Vancouver, Canada. Association for Computational Linguistics.

Benjamin Marie and Atsushi Fujita. 2018. Phrase table induction using monolingual data for low-resource statistical machine translation. *ACM Transaction on Asian and Low-Resource Language Information Processing*, 17(3):16:1–16:25.

Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Rapid Sun, Sethserey Sam, Sophe ap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian Language Treebank. In *Proceedings of the 2016 Conference of the Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Technique (O-COCOSDA)*, pages 1–6, Bali, Indonesia.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788, Melbourne, Australia. Association for Computational Linguistics.

Christoph Tillmann and Jian-ming Xu. 2009. A simple sentence-level extraction algorithm for comparable data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 93–96, Boulder, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Yining Wang, Yang Zhao, Jiajun Zhang, Chengqing Zong, and Zhengshan Xue. 2017. Towards neural machine translation with partially aligned corpora. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 384–393, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2017. Improving neural machine translation through phrase-based forced decoding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 152–162, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Low-Resource Syntactic Transfer with Unsupervised Source Reordering

Mohammad Sadegh Rasooli
Facebook AI
Menlo Park, CA, USA
`rasooli@fb.com`

Michael Collins
Department of Computer Science
Columbia University
`mcollins@cs.columbia.edu`

Abstract

We describe a cross-lingual transfer method for dependency parsing that takes into account the problem of word order differences between source and target languages. Our model only relies on the Bible, a considerably smaller parallel data than the commonly used parallel data in transfer methods. We use the concatenation of projected trees from the Bible corpus, and the gold-standard treebanks in multiple source languages along with cross-lingual word representations. We demonstrate that reordering the source treebanks before training on them for a target language improves the accuracy of languages outside the European language family. Our experiments on 68 treebanks (38 languages) in the Universal Dependencies corpus achieve a high accuracy for all languages. Among them, our experiments on 16 treebanks of 12 non-European languages achieve an average UAS absolute improvement of 3.3% over a state-of-the-art method.

1 Introduction

There has recently been a great deal of interest in cross-lingual transfer of dependency parsers, for which a parser is trained for a target language of interest using treebanks in other languages. Cross-lingual transfer can eliminate the need for the expensive and time-consuming task of treebank annotation for low-resource languages. Approaches include annotation projection using parallel data sets (Hwa et al., 2005; Ganchev et al., 2009), direct model transfer through learning of a delexicalized model from other treebanks (Zeman and Resnik, 2008; Täckström et al., 2013), treebank translation (Tiedemann et al., 2014), using synthetic treebanks (Tiedemann and Agić, 2016; Wang and Eisner, 2016), using cross-lingual word representations (Täckström et al., 2012; Guo et al., 2016; Rasooli and Collins, 2017) and using cross-lingual dictionaries (Durrett et al., 2012).

Recent results from Rasooli and Collins (2017) have shown accuracies exceeding 80% on unlabeled attachment accuracy (UAS) for several European languages.[1] However non-European languages remain a significant challenge for cross-lingual transfer. One hypothesis, which we investigate in this paper, is that word-order differences between languages are a significant challenge for cross-lingual transfer methods. The main goal of our work is therefore to reorder gold-standard source treebanks to make those treebanks syntactically more similar to the target language of interest. We use two different approaches for source treebank reordering: 1) reordering based on dominant dependency directions according to the projected dependencies, 2) learning a classifier on the alignment data. We show that an ensemble of these methods with the baseline method leads to higher performance for the majority of datasets in our experiments. We show particularly significant improvements for non-European languages.[2]

The main contributions of this work are as follows:

- We propose two different syntactic reordering methods based on the dependencies projected using translation alignments. The first model is based on the dominant dependency direction in the target language according to the projected dependencies. The second model learns a reordering classifier from the small set of aligned sentences in the Bible parallel data.

---

[1] Specifically, Table 9 of Rasooli and Collins (2017) shows 13 datasets, and 11 languages, with UAS scores of over 80%; all of these datasets are in European languages.

[2] Specifically, performance of our method gives an improvement of at least 2.3% absolute scores in UAS on 11 datasets in 9 languages—Coptic, Basque, Chinese, Vietnamese, Turkish, Persian, Arabic, Indonesian Hebrew—with an average improvement of over 4.5% UAS.

Proceedings of NAACL-HLT 2019, pages 3845–3856
Minneapolis, Minnesota, June 2 - June 7, 2019.

- We run an extensive set of experiments on 68 treebanks for 38 languages. We show that by just using the Bible data, we are able to achieve significant improvements in non-European languages. Our ensemble method is able to maintain a high accuracy in European languages.

- We show that syntactic transfer methods can outperform a supervised model for cases in which the gold-standard treebank is very small. This indicates the strength of these models when the language is truly low-resource.

Unlike most previous work for which a simple delexicalized model with gold part-of-speech tags are used, we use lexical features and automatic part-of-speech tags. Our final model improves over two strong baselines, one with annotation projection and the other one inspired by the non-neural state-of-the-art model of Rasooli and Collins (2017). Our final results improve the performance on non-European languages by an average UAS absolute improvement of 3.3% and LAS absolute improvement of 2.4%.

2 Related Work

There has recently been a great deal of research on dependency parser transfer. Early work on direct model transfer (Zeman and Resnik, 2008; McDonald et al., 2011; Cohen et al., 2011; Rosa and Zabokrtsky, 2015; Wang and Eisner, 2018a) considered learning a delexicalized parser from one or many source treebanks. A number of papers (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Ammar et al., 2016; Wang and Eisner, 2017) have considered making use of topological features to overcome the problem of syntactic differences across languages. Our work instead reorders the source treebanks to make them similar to the target language before training on the source treebanks.

Agić (2017) use part-of-speech sequence similarity between the source and target language for selecting the source sentences in a direct transfer approach. Ponti et al. (2018) preprocess source trees to increase the isomorphy between the source and the target language dependency trees. They apply their method on a simple delexicalized model and their accuracy on the small set of languages that they have tried is significantly worse than ours in all languages. The recent work by Wang and Eisner (2018b) reorders delexicalized treebanks of part-of-speech sequences in order to make it more similar to the target language of interest. The latter work is similar to our work in terms of using reordering. Our work is more sophisticated by using a full-fledged parsing model with automatic part-of-speech tags and every accessible dataset such as projected trees and multiple source treebanks as well as cross-lingual word embeddings for all languages.

Previous work (Täckström et al., 2012; Duong et al., 2015; Guo et al., 2015, 2016; Ammar et al., 2016) has considered using cross-lingual word representations. A number of authors (Durrett et al., 2012; Rasooli and Collins, 2017) have used cross-lingual dictionaries. We also make use of cross-lingual word representations and dictionaries in this paper. We use the automatically extracted dictionaries from the Bible to translate words in the source treebanks to the target language. One other line of research in the delexicalized transfer approach is creating a synthetic treebank (Tiedemann and Agić, 2016; Wang and Eisner, 2016, 2018b).

Annotation projection (Hwa et al., 2005; Ganchev et al., 2009; McDonald et al., 2011; Ma and Xia, 2014; Rasooli and Collins, 2015; Lacroix et al., 2016; Agić et al., 2016) is another approach in parser transfer. In this approach, supervised dependencies are projected through word alignments and then used as training data. Similar to previous work (Rasooli and Collins, 2017), we make use of a combination of projected dependencies from annotation projection in addition to partially translated source treebanks. One other approach is treebank translation (Tiedemann et al., 2014) for which a statistical machine translation system is used to translate source treebanks to the target language. These models need a large amount of parallel data for having an accurate translation system.

Using the Bible data goes back to the work of Diab and Finch (2000) and Yarowsky et al. (2001). Recently there has been more interest in using the Bible data for different tasks, due to its availability for many languages (Christodouloupoulos and Steedman, 2014; Agić et al., 2015, 2016; Rasooli and Collins, 2017). Previous work (Östling and Tiedemann, 2017) has shown that the size of the Bible dataset does not provide a reliable machine translation model. Previous work in the context of machine translation (Bisazza and Federico, 2016; Daiber et al., 2016) presumes the availability of a parallel data that is often much larger than the Bible data.

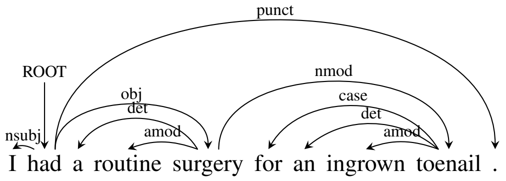
3 Baseline Model

Our model trains on the concatenation of projected dependencies $\mathcal{P}$ and all of the source treebanks $\mathcal{T}_1 \dots \mathcal{T}_k$. The projected data is from the set of projected dependencies for which at least 80% of words have projected dependencies or there is a span of length $l \geq 5$ such that all words in that span achieve a projected dependency. This is the same as the definition of dense structures $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 5}$ by Rasooli and Collins (2015).

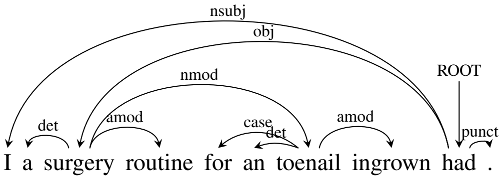
We use our reimplementation of the state-of-the-art neural biaffine graph-based parser of Dozat and Manning (2016)[3]. Because many words in the projected dependencies do not have a head assignment, the parser ignores words without heads during training. Inspired by Rasooli and Collins (2017), we replace every word in the source treebanks with its most frequent aligned translation word from the Bible data in the target language. If that word does not appear in the Bible, we use the original word. That way, we have a code-switched data for which some of the words are being translated. In addition to fine-tuning the word embeddings, we use the fixed pre-trained cross-lingual word embeddings using the training approach of Rasooli and Collins (2017) using the Wikipedia data and the Bible dictionaries.

4 Approach

Before making use of the source treebanks $\mathcal{T}_1 \dots \mathcal{T}_k$ in the training data, we reorder each tree in the source treebanks to be syntactically more similar to the word order of the target language. In general, for a head h that has c modifiers $m_1 \dots m_c$, we decide to put each of the dependents m_i on the left or right of the head h. After placing them in the correct side of the head, the order in the original source sentence is preserved. Figure 1 shows a real example of an English tree that is reordered for the sake of Persian as the target language. Here we see that we have a verb-final sentence, with nominal modifiers following the head noun. If one aims to translate this English sentence word by word, the reordered sentence gives a very good translation without any change in the sentence.



(a) Original tree.



(b) Persian-specific reordered tree.

Figure 1: An example of a gold-standard English tree that is reordered to look similar to the Persian syntactic order.

As mentioned earlier, we use two different approaches for source treebank reordering: 1) reordering based on dominant dependency directions according to the projected dependencies, 2) learning a classifier on the alignment data. We next describe these two methods.

4.1 Model 1: Reordering Based on Dominant Dependency Direction

The main goal of this model is to reorder source dependencies based on dominant dependency directions in the target language. We extract dominant dependency directions according to the projected dependencies $\mathcal{P}$ from the alignment data, and use the information for reordering source treebanks.

Let the tuple $\langle i, m, h, r \rangle$ show the dependency of the m'th word in the i'th projected sentence for which the h'th word is the parent with the dependency label r. $\langle i, m, \texttt{NULL}, \texttt{NULL} \rangle$ shows an unknown dependency for the m'th word: this occurs when some of the words in the target sentence do not achieve a projected dependency. We use the notations $h(i, m)$ and $r(i, m)$ to show the head index and dependency label of the m'th word in the i'th sentence.

Definition 1 *Dependency direction:* $d(i, m)$ *shows the dependency direction of the* m*'th*

---

[3] `https://github.com/rasoolims/universal-parser`

modifier word in the i'th sentence:

$$d(i,m) = \begin{cases} 1 & \texttt{if } h(i,m) > m \\ -1 & \texttt{otherwise} \end{cases}$$

Definition 2 *Dependency direction proportion: Dependency direction proportion of each dependency label l with direction $d \in \{-1, 1\}$ is defined as:*

$$\alpha^{(\mathcal{P})}(l,d) = \frac{\sum_{i=1}^{|\mathcal{P}|} \sum_{m=1}^{|\mathcal{P}^{(i)}|} \mathbb{I}(r(i,m) = l \,\&\, d(i,m) = d)}{\sum_{i=1}^{|\mathcal{P}|} \sum_{m=1}^{|\mathcal{P}^{(i)}|} \mathbb{I}(r(i,m) = l)}$$

Definition 3 *Dominant dependency direction: For each dependency label l, we define the dominant dependency direction $\lambda^{(\mathcal{P})}(l) = d$ if $\alpha^{(\mathcal{P})}(l,d) > 0.75$. In cases where there is no dominant dependency direction, $\lambda^{(\mathcal{P})}(l) = 0$.*

We consider the following dependency labels for extracting dominant dependency direction information: nsubj, obj, iobj, csubj, ccomp, xcomp, obl, vocative, expl, dislocated, advcl, advmod, aux, cop, nmod, appos, nummod, acl, amod. We find the direction of other dependency relations, such as most of the function word dependencies and other non-core dependencies such as conjunction, not following a fixed pattern in the Universal Dependencies corpus.

Reordering condition Given a set of projections $\mathcal{P}$, we calculate the dominant dependency direction information for the projections $\lambda^{(\mathcal{P})}$. Similar to the projected dependencies, we extract *supervised* dominant dependency directions from the gold-standard source treebank $\mathcal{D}$: $\lambda^{(\mathcal{D})}$. When we encounter a gold-standard dependency relation $\langle i, m, h, r \rangle$ in a source treebank $\mathcal{D}$, we change the direction if the following condition holds:

$$\lambda^{(\mathcal{D})}(r) \neq \lambda^{(\mathcal{P})}(r) \ \texttt{and} \ \lambda^{(\mathcal{P})}(r) = -d(i,m)$$

In other words, if the source and target languages do not have the same dominant dependency direction for r and the dominant direction of the target language is the reverse of the current direction, we change the direction of that dependency. Reordering multiple dependencies in a gold standard tree then results in a reordering of the full tree, as for example in the transformation from Figure 1a to Figure 1b.

4.2 Model 2: Reordering Classifier

We now describe our approach for learning a reordering classifier for a target language using the alignment data. Unlike the first model for which we learn concrete rules, this model learns a reordering classifier from automatically aligned data. This model has two steps; the first step prepares the training data from the automatically aligned parallel data, and the second step learns a classifier from the training data.

4.2.1 Preparing Training Data from Alignments

The goal of this step is to create training data for the reordering classifier. This data is extracted from the concatenation of parallel data from all source languages translated to the target language. Given a parallel dataset $(e^{(i)}, f^{(i)})$ for $i = 1 \ldots n$ that contains pairs of source and target sentences $e^{(i)}$ and $f^{(i)}$, the following steps are applied to create training data:

1. **Extracting reordering mappings from alignments:** We first extract intersected word alignments for each source-target sentence pair. This is done by running the Giza++ alignments (Och and Ney, 2003) in both directions. We ignore sentence pairs that more than half of the source words do not get alignment. We create a new mapping $\mu^{(i)} = \mu_1^{(i)} \ldots \mu_{s_i}^{(i)}$ that maps each index $1 \leq j \leq s_i$ in the original source sentence to a unique index $1 \leq \mu_j^{(i)} \leq s_i$ in the reordered sentence.

2. **Parsing source sentences:** We parse each source sentence using the supervised parser of the source language. We use the mapping $\mu^{(i)}$ to come up with a reordered tree for each sentence. In cases for which the number of non-projective arcs in the projected tree increase compared to the original tree, we do not use the sentence in the final training data.

3. **Extracting classifier instances:** We create a training instance for every modifier word $\langle i, m, h, r \rangle$. The decision about the direction of each dependency can be made based on the following condition:

$$d^*(i,m) = \begin{cases} 1 & \texttt{if } \mu_h^{(i)} > \mu_m^{(i)} \\ -1 & \texttt{otherwise} \end{cases}$$

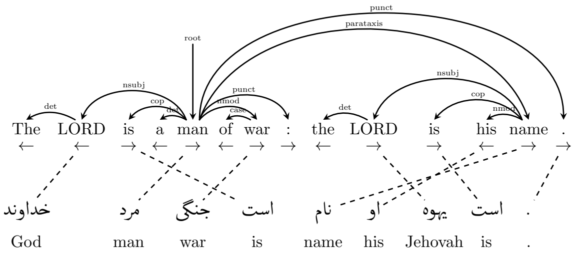


Figure 2: A reordering example from the Bible for English-Persian language pair. The Persian words are written from left to right for the ease of presentation. The arrows below the English words show the new dependency direction with respect to the word alignments to the Persian side. The reordered sentence would be "The LORD a man of war is : his name the LORD is .".

> In other words, we decide about the new order of a dependency according to the mapping $\mu^{(i)}$.

Figure 2 shows an example for the data preparation step. As shown in the figure, the new directions for the English words are decided according to the Persian alignments.

4.2.2 Classifier

The reordering classifier decides about the new direction of each dependency according to the recurrent representation of the head and dependent words. For a source sentence $e^{(i)} = e_1^{(i)} \dots e_{s_i}^{(i)}$ that belongs to a source language $\mathcal{L}$, we first obtain its recurrent representation $\eta^{(i)} = \eta_1^{(i)} \dots \eta_{s_i}^{(i)}$ by running a deep (3 layers) bi-directional LSTM (Hochreiter and Schmidhuber, 1997), where $\eta_j^{(i)} \in \mathbb{R}^{d_h}$. For every dependency tuple $\langle i, m, h, r \rangle$, we use a multi-layer Perceptron (MLP) to decide about the new order $dir \in \{-1, 1\}$ of the m'th word with respect to its head h:

$$p(\mathtt{dir}|i,m,h,r) = \mathtt{softmax}_{\mathtt{dir}}(W\phi(i,m,h,r))$$

where $W \in \mathbb{R}^{2 \times d_\phi}$ and $\phi(i,m,h,r) \in \mathbb{R}^{d_\phi}$ is as follows:

$$\phi(i,m,h,r) = \mathtt{relu}(Hq(i,m,h,r) + B)$$

where `relu` is the rectified linear unit activation (Nair and Hinton, 2010), $H \in \mathbb{R}^{d_\phi \times d_q}$, $B \in \mathbb{R}^{d_\phi}$, and $q(i,m,h,r) \in \mathbb{R}^{d_q}$ is as follows:

$$q(i,m,h,r) = [\eta_m^{(i)}; \eta_h^{(i)}; R[r]; \Lambda[\mathbb{I}(h > m)]; L[\mathcal{L}]]$$

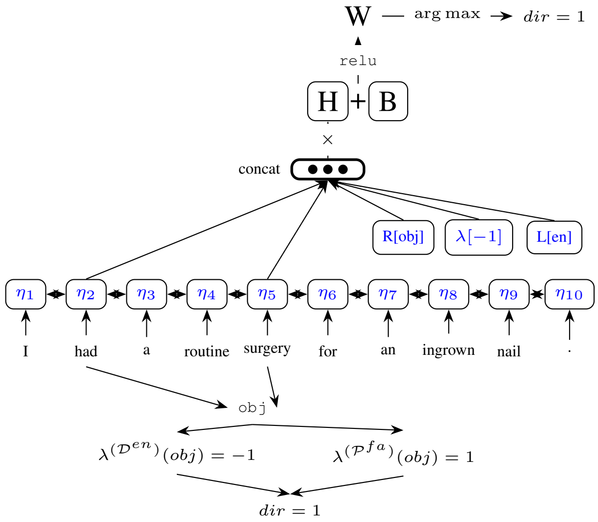


Figure 3: Two different approaches for reordering the dependency order for the example in Figure 1. The reordering classifier is shown on top, for the dependency relation between the words "had" and "surgery" with an "obj" relation. At the bottom, the reordering model based on dominant dependency direction information is shown.

where $\eta_m^{(i)}$ and $\eta_h^{(i)}$ are the recurrent representations for the modifier and head words respectively, R is the dependency relation embedding dictionary that embeds every dependency relation to a $\mathbb{R}^{d_r}$ vector, Λ is the direction embedding for the original position of the head with respect to its head and embeds each direction to a 2-dimensional vector, and L is the language embedding dictionary that embeds the source language id $\mathcal{L}$ to a $\mathbb{R}^{d_L}$ vector.

The input to the recurrent layer is the concatenation of two input vectors. The first vector is the sum of the fixed pre-trained cross-lingual embeddings, and randomly initialized word vector. The second vector is the part-of-speech tag embeddings.

Figure 3 shows a graphical depiction of the two reordering models that we use in this work.

5 Experiments

Datasets and Tools We use 68 datasets from 38 languages in the Universal Dependencies corpus version 2.0 (Nivre et al., 2017). The languages are Arabic (ar), Bulgarian (bg), Coptic (cop), Czech (cs), Danish (da), German (de), Greek (el), English (en), Spanish (es), Estonian (et), Basque (eu), Persian (fa), Finnish (fi), French (fr), Hebrew (he), Hindi (hi), Croatian (hr), Hungarian (hu), Indonesian (id), Italian (it), Japanese (ja), Korean (ko), Latin (la), Lithuanian (lt), Latvian (lv), Dutch (nl), Norwegian (no), Polish (pl), Portuguese (pt), Romanian (ro), Russian (ru), Slovak (sk), Slovene (sl), Swedish (sv), Turkish (tr), Ukrainian (uk), Vietnamese (vi), and Chinese (zh).

We use the Bible data from Christodouloupoulos and Steedman (2014) for the 38 languages. We extract word alignments using Giza++ default model (Och and Ney, 2003). Following Rasooli and Collins (2015), we obtain intersected alignments and apply soft POS consistency to filter potentially incorrect alignments. We use the Wikipedia dump data to extract monolingual data for the languages in order to train monolingual embeddings. We follow the method of Rasooli and Collins (2017) to use the extracted dictionaries from the Bible and monolingual text from Wikipedia to create cross-lingual word embeddings. We use the UDPipe pretrained models (Straka and Straková, 2017) to tokenize Wikipedia, and a reimplementation of the Perceptron tagger of Collins (2002)[4] to achieve automatic POS tags trained on the training data of the Universal Dependencies corpus (Nivre et al., 2017). We use word2vec (Mikolov et al., 2013)[5] to achieve embedding vectors both in monolingual and cross-lingual settings.

Supervised Parsing Models We trained our supervised models on the union of all datasets in a language to obtain a supervised model for each language. It is worth noting that there are two major changes that we make to the neural parser of Dozat and Manning (2016) in our implementation[6] using the Dynet library (Neubig et al., 2017): first, we add a one-layer character BiLSTM to represent the character information for each word. The final character representation is obtained by concatenating the forward representation of the last character and the backward representation of the first character. The concatenated vector is summed with the randomly initialized as well as fixed pre-trained cross-lingual word embedding vectors. Second, inspired by Weiss et al. (2015), we maintain the moving average parameters to obtain more robust parameters at decoding time.

We excluded the following languages from the set of source languages for *annotation projection* due to their low supervised accuracy: Estonian, Hungarian, Korean, Latin, Lithuanian, Latvian, Turkish, Ukrainian, Vietnamese, and Chinese.

Baseline Transfer Models We use two baseline models: 1) Annotation projection: This model only trains on the projected dependencies. 2) Annotation projection + direct transfer: To speed up training, we sample at most thousand sentences from each treebank, comprising a training data of about 37K sentences.

5.1 Reordering Ensemble Model

We noticed that our reordering models perform better in non-European languages, and perform slightly worse in European languages. We use the following ensemble model to make use of all of the three models (annotation projection + direct transfer, and the two reordering models), to make sure that we always obtain an accurate parser.

The ensemble model is as follows: given three output trees for the i'th sentence $\langle i_j, m, h_j, r_j\rangle$ for $j = 1, 2, 3$ in the target language $\mathcal{L}$, where the first tuple ($j = 1$) belongs to the baseline model, the second ($j = 2$) and third ($j = 3$) belong to the two reordering models, we weight each dependency edge with respect to the following conditions:

$$\omega(m,h,r) = z(m,h,r)\cdot\sum_{j=1}^{3} c(j,\mathcal{L})\cdot\mathbb{I}(\langle i_j, m, h, r\rangle)$$

where $c(j,\mathcal{L})$ is a coefficient that puts more weight on the first or the other two outputs depending on the target language family:

$$c(j,\mathcal{L}) = \begin{cases} 2 & \texttt{if } j = 1 \,\&\, \mathcal{L}\texttt{ is European} \\ 2 & \texttt{if } j > 1 \,\&\, \\ & \qquad \mathcal{L}\texttt{ is not European} \\ 1 & \texttt{otherwise} \end{cases}$$

and $z(m,h,r)$ is a simple weighting depending on the dominant order information:

$$z(m,h,r) = \begin{cases} 1 & \text{if } dir(\langle m,h\rangle) = -\lambda^{(\mathcal{P})}(r) \\ 3 & \text{if } dir(\langle m,h\rangle) = \lambda^{(\mathcal{P})}(r) \\ 2 & \text{otherwise } (\lambda^{(\mathcal{P})}(r) = 0) \end{cases}$$

---

[4] `https://github.com/rasoolims/SemiSupervisedPosTagger`

[5] `https://github.com/dav/word2vec`

[6] `https://github.com/rasoolims/universal-parser`

Variable	Notation	Size
Word embedding	d_w	100
POS embedding	d_p	100
Bi-LSTM	d_h	400
Dep. relation embedding	d_r	50
Language ID embedding	d_L	50
Hidden layer	d_ϕ	200
Number of BiLSTM layers	–	3
Mini-batch size (tokens)	–	~ 1000

Table 1: Parameter values in the reordering classifier model.

The above coefficients are modestly tuned on the Persian language as our development language. We have not seen any significant change in modifying the numbers: instead, the fact that an arc with a dominant dependency direction is regarded as a more valuable arc, and the baseline should have more effect in the European languages suffices for the ensemble model.

We run the Eisner first-order graph-based algorithm (Eisner, 1996) on top of the edge weights ω to extract the best possible tree.

5.2 Parameters

We run all of the transfer models with 4000 mini-batches, in which each mini-batch contains approximately 5000 tokens. We follow the same parameters as in Dozat and Manning (2016) and use a dimension of 100 for character embeddings. For the reordering classifier, we use the Adam algorithm (Kingma and Ba, 2014) with default parameters to optimize the log-likelihood objective. We filter the alignment data to keep only those sentences for which at least half of the source words have an alignment. We randomly choose 1% of the reordering data as our heldout data for deciding when to stop training the reordering models. Table 1 shows the parameter values that we use in the reordering classifier.

5.3 Results

Table 2 shows the results on the Universal Dependencies corpus (Nivre et al., 2017). As shown in the table, the algorithm based on dominant dependency directions improves the accuracy on most of the non-European languages and performs slightly worse than the baseline model in the European languages. The ensemble model, in spite of its simplicity, improves over the baseline in most of the languages, leading to an average UAS improvement of 0.9 for all languages and 3.3 for non-European languages. This improvement is very significant in many of the non-European languages; for example, from an LAS of 37.6 to 52.7 in Coptic, from a UAS of 44.9 to 53.7 in Basque, from a UAS of 40.6 to 47.0 in Chinese. Our model also outperforms the supervised models in Ukrainian and Latvian. That is an interesting indicator that for cases that the training data is very small for a language (37 sentences for Ukrainian, and 153 sentences for Latvian), our transfer approach outperforms the supervised model.

6 Analysis

In this section, we briefly describe our analysis based on the results in the ensemble model and the baseline. For some languages such as Coptic, the number of dense projected dependencies is too small (two trees) such that the parser gives a worse learned model than a random baseline. For some other languages, such as Norwegian and Spanish, this number is too high (more than twenty thousand trees), such that the baseline model performs very well.

The dominant dependency direction model generally performs better than the classifier. Our manual investigation shows that the classifier kept many of the dependency directions unchanged, while the dominant dependency direction model changed more directions. Therefore, the dominant direction model gives a higher recall with the expense of losing some precision. The training data for the reordering classifier is very noisy due to wrong alignments. We believe that the dominant direction model, besides its simplicity, is a more robust classifier for reordering, though the classifier is helpful in an ensemble setting.

Our detailed analysis show that we are able to improve the head dependency relation for the three most important head POS tags in the dependency grammar. We see that this improvement is more consistent for all non-European languages. Table 3 shows the differences in parsing f-score of dependency relations for adjectives, nouns and verbs as the head. As we see in the Table, we are able to improve the head dependency relation for the three most important head POS tags in the dependency grammar. We see that this improvement is more consistent for all non-European languages. We skip the details of those analysis due to space

Dataset	Baselines				Reordering								Supervised	
	Projection		Direct+Proj		Dominant		Classifier		Ensemble		**Difference**			
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Coptic	2.0	0.4	58.5	37.6	69.1	52.7	65.5	50.9	69.6	52.7	11.1	15.1	86.9	80.1
Basque	39.5	22.0	44.9	29.0	53.7	34.0	48.6	32.2	53.7	34.4	8.8	5.4	81.9	75.9
Chinese	23.6	10.8	40.6	17.8	47.3	25.4	45.4	23.5	47.0	25.6	6.4	7.8	81.1	74.8
Vietnamese	44.6	26.8	51.2	33.6	55.3	34.5	50.4	34.2	55.1	34.5	4.0	0.9	66.2	56.7
Turkish_pud	44.7	19.9	46.6	24.5	50.3	26.7	42.6	22.0	49.9	26.3	3.4	1.8	56.7	31.7
Persian	54.4	46.2	61.8	53.0	64.3	54.7	63.0	53.4	65.1	55.4	3.3	2.4	87.8	83.6
Arabic_pud	60.3	44.2	65.2	50.5	68.2	52.0	66.5	51.4	68.3	52.3	3.2	1.8	71.9	58.8
Indonesian	59.9	42.8	72.1	56.0	73.6	56.5	72.9	56.8	74.6	56.7	2.5	0.6	84.8	77.4
Turkish	44.6	23.9	46.6	29.3	48.9	30.6	44.9	26.6	49.0	30.0	2.4	0.7	64.2	52.5
Hebrew	63.1	46.9	70.4	55.4	72.4	54.9	71.6	55.7	72.7	55.4	2.3	0.0	88.2	82.4
Arabic	49.5	36.8	58.9	46.8	60.8	48.3	59.2	46.9	61.2	48.8	2.3	2.0	85.6	78.9
Japanese	54.8	38.9	65.2	46.5	65.9	46.8	64.1	44.8	66.6	46.8	1.4	0.3	94.5	92.7
Japanese_pud	58.6	44.1	66.8	51.5	67.4	51.5	64.7	48.4	67.9	51.9	1.1	0.4	94.7	93.5
Korean	34.3	17.3	43.0	24.8	43.5	23.8	43.6	26.4	44.1	24.7	1.1	-0.2	76.2	69.9
Hindi_pud	53.4	43.3	58.2	47.6	58.3	47.5	58.8	48.5	58.9	48.2	0.6	0.6	70.2	55.6
Lithuanian	60.6	42.5	66.6	49.5	63.7	46.8	64.6	46.0	67.2	49.9	0.6	0.4	54.8	40.0
Czech_cac	33.9	14.8	76.2	66.9	76.3	66.7	75.2	65.8	76.7	67.4	0.5	0.6	92.1	88.3
Czech_cltt	13.7	5.1	69.4	59.7	69.7	59.5	66.6	57.8	70.0	60.3	0.5	0.6	88.9	84.9
French_partut	81.6	75.2	84.3	77.8	84.9	78.4	84.4	78.1	84.8	78.4	0.5	0.5	90.0	85.1
Croatian	70.6	59.9	79.4	69.9	79.3	69.5	77.9	67.7	79.9	70.1	0.5	0.2	86.8	80.4
Greek	62.3	47.2	75.9	63.9	75.4	63.1	74.7	62.5	76.4	64.1	0.4	0.2	88.0	84.4
Russian_pud	75.7	65.8	81.1	72.2	80.9	72.2	79.9	70.7	81.5	72.7	0.4	0.5	86.5	74.1
German	71.4	62.3	75.4	67.1	75.6	67.1	75.5	66.4	75.8	67.3	0.4	0.2	85.9	81.2
French	80.2	72.9	83.0	75.9	82.9	75.9	83.3	75.9	83.4	76.2	0.4	0.3	90.4	86.9
Czech	33.9	14.5	74.6	65.3	74.1	64.4	73.0	63.7	75.0	65.8	0.4	0.5	92.5	89.1
Finnish_pud	64.1	52.5	67.2	55.0	66.8	55.0	67.3	55.1	67.5	55.5	0.4	0.5	81.6	74.5
Dutch	59.2	48.2	68.5	55.2	69.6	55.9	68.3	54.4	68.8	55.4	0.4	0.1	83.5	76.6
Russian	68.9	59.4	75.1	63.9	75.4	64.1	74.5	63.4	75.5	64.3	0.4	0.4	85.7	77.9
Latin_ittb	56.4	42.5	63.0	49.2	63.2	49.5	62.4	48.7	63.3	49.7	0.4	0.4	89.5	86.5
Norwegian_nynorsk	72.5	62.9	76.4	68.1	76.5	68.0	76.1	67.3	76.8	68.4	0.3	0.3	91.3	88.8
Ukrainian	55.1	36.9	64.3	46.1	64.5	45.7	61.7	42.2	64.6	45.9	0.3	-0.2	43.3	22.1
Bulgarian	80.4	69.4	83.8	73.8	84.0	73.8	83.1	73.0	84.1	73.9	0.3	0.1	90.9	86.0
English_lines	75.6	66.5	77.8	69.0	78.9	69.9	77.0	68.2	78.1	69.2	0.3	0.3	85.8	80.5
Finnish_ftb	63.9	46.5	66.0	48.3	65.8	47.6	65.7	48.1	66.3	48.4	0.3	0.1	81.1	74.4
Russian_syntagrus	69.4	57.5	73.9	62.2	73.8	61.8	73.2	61.2	74.2	62.3	0.3	0.1	91.3	88.3
Finnish	60.6	48.7	64.6	51.9	63.5	51.2	63.7	51.1	64.8	52.0	0.2	0.1	80.9	73.5
Hungarian	58.3	41.1	67.8	49.0	67.8	48.9	65.8	47.4	68.0	49.1	0.2	0.1	78.2	69.8
Czech_pud	35.7	16.6	77.5	69.3	76.7	67.6	76.2	67.7	77.7	69.4	0.2	0.2	89.9	84.4
Dutch_lassysmall	61.8	52.1	73.9	63.4	73.8	62.8	73.0	61.9	74.0	63.3	0.2	0.0	91.3	87.3
Slovenian_sst	58.4	44.1	61.7	47.7	61.6	47.7	61.6	47.4	61.9	48.0	0.2	0.3	70.6	63.6
English_pud	73.5	65.5	75.9	69.3	77.1	69.9	74.5	67.7	76.0	69.4	0.2	0.2	88.3	84.2
German_pud	74.1	65.3	77.8	68.9	77.7	68.5	76.9	67.4	78.0	68.8	0.1	0.0	85.9	79.0
Polish	77.6	64.7	79.9	67.9	79.7	67.5	79.5	67.2	80.1	68.0	0.1	0.1	89.4	83.3
Swedish_lines	77.2	67.7	81.1	71.6	80.7	71.1	80.1	70.4	81.3	71.7	0.1	0.1	86.9	81.5
English	70.1	61.6	72.8	64.6	73.5	65.2	71.6	63.5	72.9	64.8	0.1	0.3	88.2	84.8
Spanish	78.5	68.0	83.1	73.8	83.2	73.8	82.3	72.8	83.2	73.9	0.1	0.1	89.3	83.9
Swedish	75.3	67.0	79.0	70.9	78.8	70.9	78.2	70.0	79.1	71.0	0.1	0.1	86.7	82.3
English_partut	72.0	65.3	77.4	71.1	78.0	71.1	76.3	69.9	77.5	71.2	0.1	0.1	88.4	83.0
Swedish_pud	75.9	67.4	80.5	72.1	80.2	72.0	79.2	71.0	80.6	72.1	0.1	0.0	84.0	77.6
Italian	81.3	74.4	85.0	79.0	85.4	79.5	84.4	78.1	85.1	79.1	0.1	0.0	92.1	89.5
Romanian	72.8	59.0	76.8	64.2	76.2	63.7	75.3	63.2	76.8	64.3	0.1	0.1	89.6	83.5
Estonian	63.1	40.8	66.7	46.0	65.6	45.8	65.5	45.2	66.7	46.1	0.1	0.2	71.6	60.7
Portuguese	62.6	50.7	84.1	76.9	83.7	76.6	83.4	76.2	84.2	77.1	0.0	0.2	90.6	85.6
Portuguese_br	60.6	47.7	81.3	71.2	80.8	70.8	80.8	70.4	81.4	71.3	0.0	0.2	91.6	89.0
Norwegian_bokmaal	78.0	70.5	80.5	73.2	80.6	73.4	79.7	72.1	80.5	73.2	0.0	0.0	92.1	89.7
French_pud	81.0	72.8	83.7	75.7	84.2	76.2	83.3	75.2	83.7	75.7	0.0	0.0	89.1	83.8
Spanish_pud	81.3	70.9	84.3	75.6	84.6	76.0	83.6	74.6	84.3	75.7	0.0	0.1	89.1	80.8
Latvian	59.0	43.6	63.3	47.2	62.1	45.6	60.7	44.7	63.3	47.0	0.0	-0.2	71.3	61.2
Italian_pud	83.8	76.0	87.3	81.3	87.5	81.3	86.5	79.9	87.3	81.2	0.0	-0.1	91.9	88.4
French_sequoia	79.1	73.0	82.2	76.4	81.6	75.8	81.9	76.0	82.2	76.4	0.0	0.0	90.4	86.7
Latin	49.2	33.6	53.9	36.2	51.3	33.3	54.0	35.5	53.9	35.4	0.0	-0.8	67.2	54.5
Slovene	76.4	67.6	82.1	74.2	81.3	73.0	81.3	73.3	82.0	74.2	-0.1	0.0	88.9	85.4
Spanish_ancora	77.7	66.2	82.4	72.7	82.0	72.2	81.4	71.3	82.3	72.5	-0.1	-0.3	91.1	87.0
Danish	70.7	61.7	75.7	67.4	75.3	66.7	74.6	66.2	75.6	67.2	-0.1	-0.2	83.1	79.3
Portuguese_pud	63.5	51.8	82.7	75.8	82.5	75.8	82.0	74.8	82.6	75.7	-0.2	-0.1	86.4	78.5
Latin_proiel	59.2	46.2	61.5	47.4	60.9	47.1	60.2	46.0	61.3	47.2	-0.2	-0.2	80.9	75.4
Slovak	73.6	63.8	78.7	71.0	78.0	69.8	77.1	68.7	78.5	70.7	-0.2	-0.3	83.5	77.9
Hindi	58.7	47.2	63.7	50.0	62.3	49.0	62.6	49.3	62.7	49.4	-1.0	-0.6	94.2	90.4
Avg. All	62.0	49.7	71.2	59.3	71.7	59.6	70.6	58.7	**72.1**	**60.0**	0.9	0.7	83.9	77.3
Avg. Non-EU	46.6	32.0	57.1	40.9	60.1	43.1	57.8	41.9	**60.4**	**43.3**	3.3	2.4	80.3	72.2

Table 2: Dependency parsing results, in terms of unlabeled attachment accuracy (UAS) and labeled attachment accuracy (LAS) after ignoring punctuations, on the Universal Dependencies v2 test sets (Nivre et al., 2017) using supervised part-of-speech tags. The results are sorted by their "difference" between the ensemble model and the baseline. The rows for non-European languages are highlighted with cyan. The rows that are highlighted by pink are the ones that the transfer model outperforms the supervised model. For all of the non-European datasets except "hi", our model outperforms significantly better in terms of UAS with $p < 0.001$ using McNemar's test.

data	ADJ		NOUN		VERB	
	Base	Ens.	Base	Ens.	Base	Ens.
ar	40.4	46.7	70.6	72.5	55.3	58.8
ar_pud	32.3	39.7	73.2	75.9	67.2	70.1
bg	70.6	71.1	85.8	86.2	86.2	86.5
cop	0.0	0.0	63.4	75.7	64.6	76.4
cs	64.8	64.9	77.9	78.5	76.5	76.7
cs_cac	66.0	65.7	79.7	80.5	77.3	77.6
cs_cltt	55.9	56.5	76.9	77.7	68.3	68.9
cs_pud	71.2	70.9	79.4	80	80.2	80.3
da	70.9	71.2	79.5	79.3	79.5	79.6
de	65.7	66.7	81.3	81.5	75.8	76.3
de_pud	61.3	62.4	81.5	81.5	81.0	81.2
el	64.3	64.8	79.8	80.5	75.6	75.8
en	77.7	78.8	70.6	70.4	81.0	81.3
en_lines	74.4	74.7	78.3	78.5	82.2	82.8
en_partut	71.9	72.1	76.6	76.7	82.6	82.7
en_pud	69.5	70.6	75.4	75.5	81.2	81.6
es	75.6	74.6	88.0	88.4	80.6	80.9
es_ancora	71.3	71.4	87.4	87.4	83.0	82.9
es_pud	66.5	66.3	89.0	89.1	83.2	83.2
et	59.5	59.6	59.6	59.5	75.4	75.5
eu	31.1	35.4	37.6	47.9	52.4	61.2
fa	46.2	51.6	68.7	70.7	53.7	59.7
fi	65.8	66.3	61.8	62.5	70.5	70.5
fi_ftb	64.7	65.5	64.7	65.1	69.2	69.5
fi_pud	58.1	59.4	63.8	64.1	74.6	74.8
fr	74.1	74.6	87.3	87.5	81.9	82.7
fr_partut	72.2	72.9	88.4	88.8	83.1	83.8
fr_pud	71.3	71.1	88.7	88.8	81.0	81.1
fr_sequoia	72.0	72.0	86.5	86.6	82.2	82.0
he	64.7	69.1	75.6	77.8	68.1	70.6
hi	22.3	23.5	75.9	74.9	57.5	57.9
hi_pud	48.1	49.3	67.8	67.9	56.6	58.7
hr	72.3	71.8	82.2	82.4	83.1	83.8
hu	42.5	43.3	71.8	72.5	73.6	73.7
id	63.2	67.3	70.7	74.5	78.0	79.7
it	61.4	63.3	89.1	89.1	85.2	85.4
it_pud	71.7	72.0	90.7	90.7	87.1	87.2
ja	52.8	59.5	73.1	74.6	65.1	66.5
ja_pud	60.4	65.4	71.5	72.6	66.7	68.3
ko	55.7	52.9	23.5	24.3	52.4	54.3
la	35.1	35.6	43.8	44.4	58.8	58.5
la_ittb	57.9	57.4	65.5	66.5	63.5	63.6
la_proiel	55.2	55.4	61.8	61.6	64.3	64.1
lt	54.0	57.1	70.8	72.2	69.7	69.7
lv	58.7	60.2	57.0	57.2	70.3	70.6
nl	57.7	61.3	81.9	81.7	66.4	67.2
nl_lassysmall	46.4	47.6	79.8	80.0	75.4	75.3
no_bokmaal	76.0	75.9	83.4	83.4	84.2	84.4
no_nynorsk	69.7	70.5	81.4	81.8	79.6	79.9
pl	66.1	67.2	79.2	79.4	85.2	85.2
pt	72.1	73.3	88.7	88.8	82.9	82.7
pt_br	39.5	39.5	88.2	88.2	77.8	77.7
pt_pud	61.4	60.5	89.0	88.8	81.2	81.2
ro	55.6	56.4	79.3	79.5	80.3	80.3
ru	52.3	53.1	77.9	78.5	79.8	80.0
ru_pud	64.4	64.5	83.1	83.7	81.9	82.4
ru_syntagrus	57.3	56.9	78.7	79.2	74.3	74.6
sk	69.5	69.5	80.5	80.3	84.0	83.5
sl	73.6	72.6	83.3	83.4	85.2	85.2
sl_sst	60.6	61.6	69.4	69.6	67.4	67.1
sv	77.0	76.9	82.8	82.9	81.1	81.4
sv_lines	78.7	78.9	85.1	85.1	83.1	83.3
sv_pud	77.2	77.3	83.4	83.4	83.8	84.1
tr	42.3	46.8	49.4	47.9	48.0	51.5
tr_pud	43.2	46.7	50.0	52.0	49.5	53.5
uk	49.3	48.8	64.1	64.6	71.9	72.3
vi	31.5	35.7	50.6	56.5	55.1	58.3
zh	47.7	52.1	47.5	56.4	43.0	45.7

Table 3: Unlabeled attachment f-score of POS tags as heads for the baseline and the reordering ensemble model. As shown in the table, our model improves over the baseline in most cases.

limitations. More thorough analysis can be found in (Rasooli, 2019, Chapter 6).

For a few number of languages such as Vietnamese, the best model, even though improves over a strong baseline, still lacks enough accuracy to be considered as a reliable parser in place of a supervised model. We believe that more research on those language will address the mentioned problem. Our current model relies on supervised part-of-speech tags. Future work should study using transferred part-of-speech tags instead of supervised tags, leading to a much more realistic scenario for low-resource languages.

We have also calculated the POS trigram cosine similarity between the target language gold standard treeebanks, and the three source training datasets (original, and the two reordered datasets). In all of the non-European languages, the cosine similarity of the reordered datasets improved with different values in the range of $(0.002, 0.02)$. For Czech, Portuguese, German, Greek, English, Romanian, Russian, and Slovak, both of the reordered datasets slightly decreased the trigram cosine similarity. For other languages, the cosine similarity was roughly the same.

7 Conclusion

We have described a cross-lingual dependency transfer method that takes into account the problem of word order differences between the source and target languages. We have shown that applying projection-driven reordering improves the accuracy of non-European languages while maintaining the high accuracies in European languages. The focus of this paper is primarily of dependency parsing. Future work should investigate the effect of our proposed reordering methods on truly low-resource machine translation.

Acknowledgements

We deeply thank the anonymous reviewers for their useful feedback and comments.

References

Željko Agić. 2017. Cross-lingual parser selection for low-resource languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 1–10. Association for Computational Linguistics.

Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If all you have is a bit of the Bible: Learning POS taggers for truly low-resource languages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Pro-*

cessing (Volume 2: Short Papers), volume 2, pages 268–272.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Arianna Bisazza and Marcello Federico. 2016. A survey of word reordering in statistical machine translation: Computational models and language phenomena. *Computational linguistics*, 42(2):163–205.

Christos Christodouloupoulos and Mark Steedman. 2014. A massively parallel corpus: The Bible in 100 languages. *Language Resources and Evaluation*, pages 1–21.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

Joachim Daiber, Miloš Stanojević, and Khalil Sima'an. 2016. Universal reordering via linguistic typology. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3167–3176.

Mona Diab and Steve Finch. 2000. A statistical word-level translation model for comparable corpora. In *Content-Based Multimedia Information Access-Volume 2*, pages 1500–1508. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122, Beijing, China. Association for Computational Linguistics.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11, Jeju Island, Korea. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377, Suntec, Singapore. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California. Association for Computational Linguistics.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 1337–1348, Baltimore, Maryland. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, et al. 2017. Universal Dependencies 2. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Robert Östling and Jörg Tiedemann. 2017. Neural machine translation for low-resource languages. *arXiv preprint arXiv:1708.05729*.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018. Isomorphic transfer of syntactic structures in cross-lingual NLP. In *Proceedings of ACL*, Melbourne, Australia. Association for Computational Linguistics.

Mohammad Sadegh Rasooli. 2019. *Cross-Lingual Transfer of Natural Language Processing Systems*. Ph.D. thesis, Columbia University.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338, Lisbon, Portugal. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association of Computational Linguistics*, 5(1):279–293.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. Klcpos3 - a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 477–487. Association for Computational Linguistics.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140, Ann Arbor, Michigan. Association for Computational Linguistics.

Jörg Tiedemann and Željko Agić. 2016. Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 55:209–248.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with supervised learning. *Transactions of the Association for Computational Linguistics*, 5:147–161.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics*, 6:667–685.

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867, Lisbon, Portugal. Association for Computational Linguistics.

Revisiting Adversarial Autoencoder for Unsupervised Word Translation with Cycle Consistency and Improved Training

Tasnim Mohiuddin[⋆] and **Shafiq Joty**[⋆†]
[⋆]Nanyang Technological University, Singapore
[†]Salesforce Research Asia, Singapore
{mohi0004,srjoty}@ntu.edu.sg

Abstract

Adversarial training has shown impressive success in learning bilingual dictionary without any parallel data by mapping monolingual embeddings to a shared space. However, recent work has shown superior performance for non-adversarial methods in more challenging language pairs. In this work, we revisit adversarial autoencoder for unsupervised word translation and propose two novel extensions to it that yield more stable training and improved results. Our method includes regularization terms to enforce cycle consistency and input reconstruction, and puts the target encoders as an adversary against the corresponding discriminator. Extensive experimentations with European, non-European and low-resource languages show that our method is more robust and achieves better performance than recently proposed adversarial and non-adversarial approaches.

1 Introduction

Learning cross-lingual word embeddings has been shown to be an effective way to transfer knowledge from one language to another for many key linguistic tasks including machine translation, named entity recognition, part-of-speech tagging, and parsing (Ruder et al., 2017). While earlier efforts solved the associated word alignment problem using large parallel corpora (Luong et al., 2015), broader applicability demands methods to relax this requirement since acquiring a large corpus of parallel data is not feasible in most scenarios. Recent methods instead use embeddings learned from monolingual data, and learn a linear mapping from one language to another with the underlying assumption that two embedding spaces exhibit similar geometric structures (*i.e.,* approximately *isomorphic*). This allows the model to learn effective cross-lingual representations without expensive supervision (Artetxe et al., 2017).

Given monolingual word embeddings of two languages, Mikolov et al. (2013a) show that a linear mapping can be learned from a seed dictionary of 5000 word pairs by minimizing the sum of squared Euclidean distances between the mapped vectors and the target vectors. Subsequent works (Xing et al., 2015; Artetxe et al., 2016, 2017; Smith et al., 2017) propose to improve the model by normalizing the embeddings, imposing an orthogonality constraint on the mapper, and modifying the objective function. While these methods assume some supervision in the form of a seed dictionary, recently fully unsupervised methods have shown competitive results. Zhang et al. (2017a,b) first reported encouraging results with *adversarial training*. Conneau et al. (2018) improved this approach with post-mapping refinements, showing impressive results for several language pairs. Their learned mapping was then successfully used to train a fully unsupervised neural machine translation system (Lample et al., 2018a,b).

Although successful, adversarial training has been criticized for not being stable and failing to converge, inspiring researchers to propose *non-adversarial* methods more recently (Xu et al., 2018a; Hoshen and Wolf, 2018; Alvarez-Melis and Jaakkola, 2018; Artetxe et al., 2018b). In particular, Artetxe et al. (2018b) show that the adversarial methods of Conneau et al. (2018) and Zhang et al. (2017a,b) fail for many language pairs.

In this paper, we revisit adversarial training and propose a number of key improvements that yield more robust training and improved mappings. Our main idea is to learn the cross-lingual mapping in a projected latent space and add more constraints to guide the unsupervised mapping in this space. We accomplish this by proposing a novel *adversarial autoencoder* framework (Makhzani et al., 2015), where adversarial mapping is done at the (latent) code space as opposed to the original embedding

Proceedings of NAACL-HLT 2019, pages 3857–3867
Minneapolis, Minnesota, June 2 - June 7, 2019.

space (Figure 1). This gives the model the flexibility to automatically induce the required geometric structures in its latent code space that could potentially yield better mappings. Søgaard et al. (2018) recently find that the *isomorphic* assumption made by most existing methods does not hold in general even for two closely related languages like English and German. In their words *"approaches based on this assumption have important limitations"*. By mapping the latent vectors through adversarial training, our approach therefore departs from the isomorphic assumption.

In our adversarial training, not only the mapper but also the target encoder is trained to fool the discriminator. This forces the discriminator to improve its discrimination skills, which in turn pushes the mapper to generate indistinguishable translation. To guide the mapping, we include two additional constraints. Our first constraint enforces *cycle consistency* so that code vectors after being translated from one language to another, and then translated back to their source space remain close to the original vectors. The second constraint ensures reconstruction of the original input word embeddings from the back-translated codes. This grounding step forces the model to retain word semantics during the mapping process.

We conduct a series of experiments with six different language pairs (in both directions) comprising European, non-European, and low-resource languages from two different datasets. Our results show that our model is more robust and yields significant gains over Conneau et al. (2018) for all translation tasks in all evaluation measures. Our method also gives better initial mapping compared to other existing methods (Artetxe et al., 2018b). We also perform an extensive ablation study to understand the contribution of different components of our model. The study reveals that cycle consistency contributes the most, while adversarial training of the target encoder and post-cycle reconstruction also have significant effect. We have released our source code at `https://ntunlpsg.github.io/project/unsup-word-translation/`

The remainder of this paper is organized as follows. After discussing related work in Section 2, we present our unsupervised word translation approach with adversarial autoencoder in Section 3. We describe our experimental setup in Section 4, and present our results with in-depth analysis in Section 5. Finally, we summarize our findings with possible future directions in Section 6.

2 Related Work

In recent years a number of methods have been proposed to learn bilingual dictionary from monolingual word embeddings.[1] Many of these methods use an initial seed dictionary. Mikolov et al. (2013a) show that a linear transformation can be learned from a seed dictionary of 5000 pairs by minimizing the squared Euclidean distance. In their view, the key reason behind the good performance of their model is the similarity of geometric arrangements in vector spaces of the embeddings of different languages. For translating a new source word, they map the corresponding word embedding to the target space using the learned mapping and find the nearest target word. In their approach, they found that simple linear mapping works better than non-linear mappings with multilayer neural networks.

Xing et al. (2015) enforce the word vectors to be of unit length during the learning of the embeddings and modify the objective function for learning the mapping to maximize the cosine similarity instead of using Euclidean distance. To preserve length normalization after mapping, they enforce the *orthogonality* constraint on the mapper. Instead of learning a mapping from the source to the target embedding space, Faruqui and Dyer (2014) use a technique based on Canonical Correlation Analysis (CCA) to project both source and target embeddings to a common low-dimensional space, where the correlation of the word pairs in the seed dictionary is maximized. Artetxe et al. (2016) show that the above methods are variants of the same core optimization objective and propose a closed form solution for the mapper under orthogonality constraint. Smith et al. (2017) find that this solution is closely related to the orthogonal *Procrustes* solution. In their follow-up work, Artetxe et al. (2017) obtain competitive results using a seed dictionary of only 25 word pairs. They propose a *self-learning framework* that performs two steps iteratively until convergence. In the first step, they use the dictionary (starting with the seed) to learn a linear mapping, which is then used in the second step to induce a new dictionary.

A more recent line of research attempts to eliminate the seed dictionary totally and learn the map-

---

[1] see (Ruder et al., 2017) for a nice survey

ping in a purely unsupervised way. This was first proposed by Miceli Barone (2016), who initially used an adversarial network similar to Conneau et al. (2018), and found that the mapper (which is also the encoder) translates everything to a single embedding, known commonly as the *mode collapse* issue (Goodfellow, 2017). To preserve diversity in mapping, he used a decoder to reconstruct the source embedding from the *mapped embedding*, extending the framework to an adversarial autoencoder. His preliminary qualitative analysis shows encouraging results but not competitive with methods using bilingual seeds. He suspected issues with training and with the isomorphic assumption. In our work, we successfully address these issues with an improved model that also relaxes the isomorphic assumption. Our model uses two separate autoencoders, one for each language, which allows us to put more constraints to guide the mapping. We also distinguish the role of an encoder from the role of a mapper. The encoder projects embeddings to latent code vectors, which are then translated by the mapper.

Zhang et al. (2017a) improved adversarial training with orthogonal parameterization and cycle consistency. To aid training, they incorporate additional techniques like noise injection which works as a regularizer. For selecting the best model, they rely on sharp drops of the discriminator accuracy. In their follow-up work (Zhang et al., 2017b), they minimize Earth-Mover's distance between the distribution of the transformed source embeddings and the distribution of the target embeddings. Conneau et al. (2018) show impressive results with adversarial training and refinement with the Procrustes solution. Instead of using the adversarial loss, Xu et al. (2018a) use Sinkhorn distance and adopt cycle consistency inspired by the CycleGAN (Zhu et al., 2017). We also incorporate cycle consistency along with the adversarial loss. However, while all these methods learn the mapping in the original embedding space, our approach learns it in the latent code space considering both the mapper and the target encoder as adversary. In addition, we use a postcycle reconstruction to guide the mapping.

A number of non-adversarial methods have also been proposed recently. Artetxe et al. (2018b) learn an initial dictionary by exploiting the structural similarity of the embeddings and use a robust self-learning algorithm to improve it iteratively. Hoshen and Wolf (2018) align the second moment of word distributions of the two languages using principal component analysis (PCA) and then refine the alignment iteratively using a variation of the Iterative Closest Point (ICP) method used in computer vision. Alvarez-Melis and Jaakkola (2018) cast the problem as an optimal transport problem and exploit the Gromov-Wasserstein distance which measures how similarities between pairs of words relate across languages.

3 Approach

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$ be two sets consisting of n and m word embeddings of d-dimensions for a source and a target language, respectively. We assume that $\mathcal{X}$ and $\mathcal{Y}$ are trained independently from monolingual corpora. Our aim is to learn a mapping $f(x)$ in an unsupervised way (*i.e.,* no bilingual dictionary given) such that for every x_i, $f(x)$ corresponds to its translation in $\mathcal{Y}$. Our overall approach follows the same sequence of steps as Conneau et al. (2018):

(i) Induction of seed dictionary through adversarial training.
(ii) Iterative refinement of the initial mapping through the Procrustes solution.
(iii) Apply CSLS for nearest neighbor search.

We propose a novel adversarial autoencoder model to learn the initial mapping for inducing a seed dictionary in step (i), and we adopt existing refinement methods for steps (ii) and (iii).

3.1 Adversarial Autoencoder for Initial Dictionary Induction

Our proposed model (Figure 1) has two **autoencoders**, one for each language. Each autoencoder comprises an encoder $E_{\mathcal{X}}$ (res. $E_{\mathcal{Y}}$) and a decoder $D_{\mathcal{X}}$ (res. $D_{\mathcal{Y}}$). The encoders transform an input x (res. y) into a latent code z_x (res. z_y) from which the decoders try to reconstruct the original input. We use a linear encoder and l_2 **reconstruction loss**

$$z_{x_i} = \theta_{E_{\mathcal{X}}} x_i; \qquad \hat{x}_i = \theta_{D_{\mathcal{X}}} z_{x_i} \qquad (1)$$

$$\mathcal{L}_{\text{autoenc}_{\mathcal{X}}}(\theta_{E_{\mathcal{X}}}, \theta_{D_{\mathcal{X}}}) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \qquad (2)$$

where $\theta_{E_{\mathcal{X}}} \in \mathbb{R}^{c \times d}$ and $\theta_{D_{\mathcal{X}}} \in \mathbb{R}^{d \times c}$ are the parameters of the encoder and the decoder for d-dimensional word embedding and c-dimensional

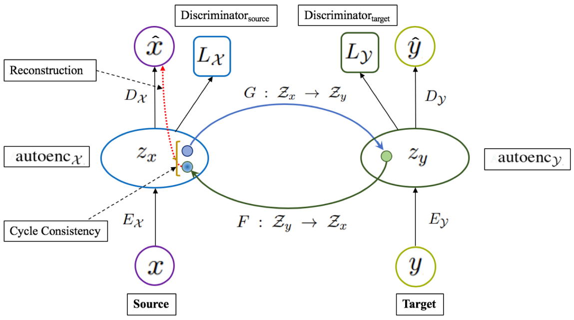


Figure 1: Our proposed adversarial autoencoder framework for unsupervised word translation.

code vector.[2] The encoder, decoder and the reconstruction loss for the other autoencoder (autoenc$_{\mathcal{Y}}$) is similarly defined.

Let $q(z_x|x)$ and $q(z_y|y)$ be the encoding distributions of the two autoencoders. We use *adversarial training* to find a mapping between $q(z_x|x)$ and $q(z_y|y)$. This is in contrast with most existing methods (*e.g.,* Conneau et al. (2018); Artetxe et al. (2017)) that directly map the distribution of the source word embeddings $p(x)$ to the distribution of the target $p(y)$. As Søgaard et al. (2018) pointed out, the *isomorphism* does not hold in general between the word embedding spaces of two languages. Mapping the latent codes gives our model more flexibility to induce the required semantic structures in its code space that could potentially yield more accurate mappings.

As shown in Figure 1, we include two linear **mappings** $G : \mathcal{Z}_x \rightarrow \mathcal{Z}_y$ and $F : \mathcal{Z}_y \rightarrow \mathcal{Z}_x$ to project the code vectors (samples from $q(.|.)$) from one language to the other. In addition, we have two language **discriminators**, $L_{\mathcal{X}}$ and $L_{\mathcal{Y}}$. The discriminators are trained to discriminate between the mapped codes and the encoded codes, while the mappers and encoders are jointly trained to fool their respective discriminator. This results in a **three-player** game, where the discriminator tries to identify the origin of a code, and the mapper and the encoder act together to prevent the discriminator to succeed by making the mapped vector and the encoded vector as similar as possible.

Discriminator Loss Let $\theta_{L_{\mathcal{X}}}$ and $\theta_{L_{\mathcal{Y}}}$ denote the parameters of the two discriminators, and W_G and W_F are the mapping weight matrices. The loss for the source discriminator $L_{\mathcal{X}}$ can be written as

$$\begin{aligned}\mathcal{L}_{L_{\mathcal{X}}}(\theta_{L_{\mathcal{X}}}|W_F, \theta_{E_{\mathcal{X}}}) = &-\frac{1}{m}\sum_{j=1}^{m}\log P_{L_{\mathcal{X}}}(\text{src} = 0|F(z_{y_j})) \\ &-\frac{1}{n}\sum_{i=1}^{n}\log P_{L_{\mathcal{X}}}(\text{src} = 1|z_{x_i}) \quad (3)\end{aligned}$$

where $P_{L_{\mathcal{X}}}(\text{src}|z)$ is the probability according to $L_{\mathcal{X}}$ to distinguish whether z is coming from the source encoder ($\text{src} = 1$) or from the target-to-source mapper F ($\text{src} = 0$). The discrimination loss $\mathcal{L}_{L_{\mathcal{Y}}}(\theta_{L_{\mathcal{Y}}}|W_G, \theta_{E_{\mathcal{Y}}})$ is similarly defined for the target discriminator $L_{\mathcal{Y}}$ using G and $E_{\mathcal{Y}}$.

Our discriminators have the same architecture as Conneau et al. (2018). It is a feed-forward network with two hidden layers of size 2048 and Leaky-ReLU activations. We apply dropout with a rate of 0.1 on the input to the discriminators. Instead of using 1 and 0, we also apply a smoothing coefficient ($s = 0.2$) in the discriminator loss.

Adversarial Loss The mappers and encoders are trained jointly with the following adversarial loss to fool their respective discriminators.

$$\begin{aligned}\mathcal{L}_{\text{adv}}(W_F, \theta_{E_{\mathcal{X}}}|\theta_{L_{\mathcal{X}}}) = &-\frac{1}{m}\sum_{i=1}^{m}\log P_{L_{\mathcal{X}}}(\text{src} = 1|F(z_{y_j})) \\ &-\frac{1}{n}\sum_{i=1}^{n}\log P_{L_{\mathcal{X}}}(\text{src} = 0|z_{x_i}) \quad (4)\end{aligned}$$

The adversarial loss for mapper G and encoder $E_{\mathcal{Y}}$ is similarly defined. Note that we consider both the mapper and the target encoder as generators. This is in contrast to existing adversarial methods, which do not use any autoencoder in the target side. The mapper and the target encoder team up to fool the discriminator. This forces the discriminator to improve its skill and vice versa for the generators, forcing them to produce indistinguishable codes through better mapping.

Cycle Consistency and Reconstruction The adversarial method introduced above maps a "bag" of source embeddings to a "bag" of target embeddings, and in theory, the mapper can match the target language distribution. However, mapping at the bag level is often insufficient to learn the individual word level mappings. In fact, there exist infinite number of possible mappings that can match the same target distribution. Thus to learn better mappings, we need to enforce more constraints to our objective.

The first form of constraints we consider is **cycle consistency** to ensure that a source code z_x

---

[2] We also experimented with a non-linear encoder, but it did not work well.

translated to the target language code space, and translated back to the original space remains unchanged, *i.e.,* $z_x \rightarrow G(z_x) \rightarrow F(G(z_x)) \approx z_x$. Formally, the cycle consistency loss in one direction:

$$\mathcal{L}_{\text{cyc}}(W_G, W_F) = \frac{1}{n} \sum_{i=1}^{n} \|z_{x_i} - F(G(z_{x_i}))\| \tag{5}$$

The loss in the other direction ($z_y \rightarrow F(z_y) \rightarrow G(F(z_y)) \approx z_y$) is similarly defined. In addition to cycle consistency, we include another constraint to guide the mapping further. In particular, we ask the decoder of the respective autoencoder to reconstruct the original input from the back-translated code. We compute this **post-cycle reconstruction loss** for the source autoencoder as follows:

$$\mathcal{L}_{\text{rec}}(\theta_{E_\mathcal{X}}, \theta_{D_\mathcal{X}}, W_G, W_F) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - D_\mathcal{X}(F(G(z_{x_i})))\|^2 \tag{6}$$

The reconstruction loss at the target autoencoder is defined similarly. Apart from improved mapping, both cycle consistency and reconstruction lead to more stable training in our experiments. Specifically, they help our training to converge and get around the *mode collapse* issue (Goodfellow, 2017). Since the model now has to translate the mapped code back to the source code and reconstruct the original word embedding, the generators cannot get away by mapping all source codes to a single target code.

Total Loss The total loss for mapping a batch from source to target is

$$\mathcal{L}_{\text{src}\rightarrow\text{tar}} = \mathcal{L}_{\text{adv}} + \lambda_1 \mathcal{L}_{\text{cyc}} + \lambda_2 \mathcal{L}_{\text{rec}} \tag{7}$$

where λ_1 and λ_2 control the relative importance of the three loss components. Similarly we define the total loss for mapping in the opposite direction $\mathcal{L}_{\text{tar}\rightarrow\text{src}}$. The complete objective of our model is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{src}\rightarrow\text{tar}} + \mathcal{L}_{\text{tar}\rightarrow\text{src}} \tag{8}$$

3.2 Training and Dictionary Construction

We present the training procedure of our model and the overall word translation process in Algorithm 1. We first pre-train the autoencoders separately on monolingual embeddings (Step 1). This pre-training is required to induce word semantics (and relations) in the latent code space.

We start adversarial training (Step 2) by updating the discriminators for $n\_critics$ (5) times, each time with a random batch. Then we update the generators (the mapper and target encoder) on the adversarial loss. The mappers then go through two more updates, one for cycle consistency and another for post-cycle reconstruction. The autoencoders (encoder-decoder) in this stage get updated only on the post-cycle reconstruction loss. We also apply the orthogonalization update to the mappers following Conneau et al. (2018) with $\beta = 0.01$.

Algorithm 1: Unsupervised word translation with cycle-consistent adversarial autoencoder

```
Input : Two sets of word embeddings: $\mathcal{X}$ and $\mathcal{Y}$
// Initial autoencoder training
1. Train autoenc$_\mathcal{X}$ and autoenc$_\mathcal{Y}$ separately for some
   epochs on monolingual embeddings (Eq. 2);
// Adversarial training
2. for n_epochs do
    for n_iterations do
        // Critic update
        for n_critics do
            (i) Sample a batch from $\mathcal{X}$ and $\mathcal{Y}$
            (ii) Update discriminators ($L_\mathcal{X}$, $L_\mathcal{Y}$) (Eq. 3)
        end
        (a) Sample a batch from $\mathcal{X}$ as source and $\mathcal{Y}$ as
            target
        (b) Update mapper $G$ and encoder $E_\mathcal{Y}$ on
            adversarial loss to fool $L_\mathcal{Y}$ (Eq. 4)
        (c) Update mappers $G$ and $F$ on cycle
            consistency loss (Eq. 5)
        (d) Update mappers ($G$, $F$) and autoenc$_\mathcal{X}$ on
            post-cycle reconstruction loss (Eq. 6)
        // Orthogonalize the mapper
        (e) Update weight matrices of mapper $G$ and $F$
            using:
                $W_G \leftarrow (1+\beta)W_G - \beta(W_G W_G^T)W_G$
                $W_F \leftarrow (1+\beta)W_F - \beta(W_F W_F^T)W_F$
        (f) Sample a batch from $\mathcal{Y}$ as source and $\mathcal{X}$ as
            target and update accordingly (symmetric to
            (b) -(e) steps).
    end
    Use validation criterion to save the best model.
end
// Iterative Procrustes/fine-tuning
3. Load the best model.
for n_iterations do
    (a) Build a synthetic dictionary (using source
        encoder, source-to-target mapper, and CSLS)
    (b) Apply the Procrustes solution on the dictionary.
end
// Test
4. Test the model on gold bilingual dictionary.
```

Our training setting is similar to Conneau et al. (2018), and we apply the same pre- and post-processing steps. We use stochastic gradient descent (SGD) with a batch size of 32, a learning rate of 0.1, and a decay of 0.98.

For selecting the best model, we use the *unsupervised validation criterion* proposed by Conneau et al. (2018), which correlates highly with the mapping quality. In this criterion, $10,000$ most frequent source words along with their nearest neighbors in the target space are considered. The average cosine similarity between these pseudo translations is considered as the validation metric.

The initial bilingual dictionary induced by adversarial training (or any other unsupervised method) is generally of lower quality than what could be achieved by a supervised method. Conneau et al. (2018) and Artetxe et al. (2018b) propose fine-tuning methods to refine the initial mappings. Similar to Conneau et al. (2018)), we fine-tune our initial mappings (G and F) by iteratively solving the *Procrustes* problem and applying a dictionary induction step. This method uses singular value decomposition or SVD of $\mathcal{Z}_y^T\mathcal{Z}_x$ to find the optimal mappings G (similarly SVD($\mathcal{Z}_x^T\mathcal{Z}_y$) for F) given the approximate alignment of words from the previous step. For generating synthetic dictionary in each iteration, we only consider the translation pairs that are mutual nearest neighbors. In our fine-tuning, we run five iterations of this process. For finding the nearest neighbors, we use the Cross-domain Similarity Local Scaling (CSLS) which works better in mitigating the hubness problem (Conneau et al., 2018).

4 Experimental Settings

Following the tradition, we evaluate our model on **word translation** (*a.k.a.* **bilingual lexicon induction**) task, which measures the accuracy of the predicted dictionary to a gold standard dictionary.

4.1 Datasets

We evaluate our model on two different datasets. The first one is from Conneau et al. (2018), which consists of FastText monolingual embeddings of (d =) 300 dimensions (Bojanowski et al., 2017) trained on Wikipedia monolingual corpus and gold dictionaries for 110 language pairs.[3] To show the generality of different methods, we consider **European**, **non-European** and **low-resource** languages. In particular, we evaluate on English (En) from/to Spanish (Es), German (De), Italian (It), Arabic (Ar), Malay (Ms), and Hebrew (He).

We also evaluate on the more challenging dataset of Dinu et al. (2015) and its subsequent extension by Artetxe et al. (2018a). We will refer to this dataset as **Dinu-Artexe** dataset. From this dataset, we choose to experiment on English from/to Italian and Spanish. English and Italian embeddings were trained on WacKy corpora using CBOW (Mikolov et al., 2013b), while the Spanish embeddings were trained on WMT News Crawl. The CBOW vectors are also of 300 dimensions.

4.2 Baselines

We compare our method with the **unsupervised** models of Conneau et al. (2018), Artetxe et al. (2018b), Alvarez-Melis and Jaakkola (2018), Xu et al. (2018a), and Hoshen and Wolf (2018).

To evaluate how our unsupervised method compares with methods that rely on a bilingual seed dictionary, we follow Conneau et al. (2018), and compute a **supervised** baseline that uses the Procrustes solution directly on the seed dictionary (5000 pairs) to learn the mapping function, and then uses CSLS to do the nearest neighbor search. We also compare with the supervised approaches of Artetxe et al. (2017, 2018a), which to our knowledge are the state-of-the-art supervised systems. For some of the baselines, results are reported from their papers, while for the rest we report results by running the publicly available codes on our machine.

For training our model on European languages, the weight for cycle consistency (λ_1) in Eq. 7 was always set to 5, and the weight for post-cycle reconstruction (λ_2) was set to 1. For non-European languages, we use different values of λ_1 and λ_2 for different language pairs. [4] The dimension of the code vectors in our model was set to 350.

5 Results

We present our results on European languages on the datasets of Conneau et al. (2018) and Dinu et al. (2015) in Tables 1 and 3, while the results on non-European languages are shown in Table 2. Through experiments, our goal is to assess:

(i) Does the unsupervised mapping method based on our proposed adversarial autoencoder model improve over the best existing adversarial method of Conneau et al. (2018) in terms of mapping accuracy and convergence (Section 5.1)?

(ii) How does our unsupervised mapping method compare with other unsupervised and supervised approaches (Section 5.2)?

---

[3] `https://github.com/facebookresearch/MUSE`

[4] We did not tune the λ values much, rather used our initial observation. Tuning λ values might yield even better results.

	En-Es		En-De		En-It	
	→	←	→	←	→	←
Supervised (Procrustes-CSLS)	82.4	83.9	75.3	72.7	78.1	78.1
Unsupervised Baselines						
Artetxe et al. (2018b)	82.2	84.4	74.9	74.1	78.8	79.5
Alvarez and Jaakkola (2018)	81.7	80.4	71.9	72.8	78.9	75.2
Xu et al. (2018b)	79.5	77.8	69.3	67.0	73.5	72.6
Hoshen and Wolf (2018)	82.1	84.1	74.7	73.0	77.9	77.5
Conneau et al. (2018) (paper)	81.7	83.3	74.0	72.2	-	-
Conneau et al. (2018) (code)	82.3	83.7	74.2	72.6	78.3	78.1
Our Unsupervised Approach						
Adversarial autoencoder +						
Conneau et al. (2018) Refinement	82.6	84.4	**75.5**	73.9	78.8	78.5
Artetxe et al. (2018b) Refinement	**82.7**	**84.7**	75.4	**74.3**	**79.0**	**79.6**

Table 1: Word translation accuracy (P@1) on European languages on the dataset of Conneau et al. (2018) using fastText embeddings (trained on Wikipedia). '-' indicates the authors did not report the number.

	En-Ar		En-Ms		En-He	
	→	←	→	←	→	←
Supervised Baselines						
Artetxe et al. (2017)	24.8	43.3	38.8	41.6	32.7	51.1
Artetxe et al. (2018a)	36.2	52.9	51.2	47.7	43.6	56.8
Supervised (Procrus-CSLS)	34.5	49.7	47.3	46.6	39.2	54.1
Unsupervised Baselines						
Hoshen and Wolf (2018)	34.4	49.3	**	**	36.5	52.3
Artetxe et al. (2018b)	36.1	48.7	54.0	**55.4**	43.8	**57.5**
Conneau et al. (2018) (code)	29.3	47.6	46.2	**	36.8	53.1
Our Unsupervised Approach						
Adversarial autoencoder +						
Conneau et al. (2018) Refinement	33.6	49.7	49.5	44.3	40.0	54.9
Artetxe et al. (2018b) Refinement	**36.3**	**52.6**	**54.1**	51.7	**44.0**	57.1

Table 2: Word translation accuracy (P@1) on non-European and low-resource languages on the dataset of Conneau et al. (2018) using fastText embeddings. ** indicates the model failed to converge.

(iii) Which components of our adversarial autoencoder model attribute to improvements (Section 5.3)?

5.1 Comparison with Conneau et al. (2018)

Since our approach follows the same steps as Conneau et al. (2018), we first compare our proposed model with their model on European (Table 1), non-European and low-resource languages (Table 2) on their dataset. In the tables, we present the numbers that they reported in their paper (Conneau et al. (2018) (paper)) as well as the results that we get by running their code on our machine (Conneau et al. (2018) (code)). For a fair comparison with respect to the quality of the learned mappings (or induced seed dictionary), here we only consider the results of our approach that use the refinement procedure of Conneau et al. (2018).

In Table 1, we see that our **Adversarial autoencoder + Conneau et al. (2018) Refinement** outperforms Conneau et al. (2018) in all the six translation tasks involving European language pairs, yielding gains in the range 0.3 - 1.3%. Our method is also superior to theirs for the non-European and low-resource language pairs in Table 2. Here our method gives more gains ranging from 1.8 to 4.3%. Note specifically that Malay (Ms) is a low-resource language, and the FastText contains word vectors for only 155K Malay words. We found their model to be very fragile for En from/to Ms, and does not converge at all for Ms→En. We ran their code 10 times for Ms→En but failed every time. Compared to that, our method is more robust and converged most of the time we ran.

If we compare our method with Conneau et al. (2018) on the dataset of (Dinu et al., 2015; Artetxe et al., 2017) in Table 3, we see here also our method performs better than their method in all the four translation tasks involving European language pairs. In this dataset, our method shows more robustness compared to their method. For example, their method had difficulties in converging for En from/to Es translations; for En→Es, it converges only 2 times out of 10 attempts, while for Es→En it did not converge a single time in 10 attempts. Compared to that, our method was more robust, converging 4 times out of 10 attempts.

In Section 5.3, we compare our model with Conneau et al. (2018) more rigorously by evaluating them with and without fine-tuning and measuring their performance on P@1, P@5, and P@10.

5.2 Comparison with Other Methods

In this section, we compare our model with other state-of-the-art methods that do not follow the same procedure as us and Conneau et al. (2018). For example, Artetxe et al. (2018b) do the initial mapping in the similarity space, then they apply a different self-learning method to fine-tune the embeddings, and perform a final refinement with symmetric re-weighting. Instead of mapping from source to target, they map both source and target embeddings to a common space.

Let us first consider the results for European language pairs on the dataset of Conneau et al. (2018) in Table 1. Our **Adversarial autoencoder + Conneau et al. (2018) Refinement** performs better than most of the other methods on this dataset, achieving the highest accuracy for 4 out of 6 translation tasks. For De→En, our result is very close to the best system of Artetxe et al. (2018b) with only 0.2% difference.

	En-It		En-Es	
	→	←	→	←
Supervised Baselines				
Artetxe et al. (2017)	39.7	33.8	32.4	27.2
Artetxe et al. (2018a)	45.3	38.5	37.2	29.6
Procrustes-CSLS	44.9	38.5	33.8	29.3
Unsupervised Baselines				
Artetxe et al. (2018b)	**47.9**	42.3	**37.5**	31.2
Conneau et al. (2018) (paper)	45.1	38.3	-	-
Conneau et al. (2018) (code)	44.9	38.7	34.7	**
Our Unsupervised Approach				
Adversarial autoencoder +				
Conneau et al. (2018) Refinement	45.3	39.4	35.2	29.9
Artetxe et al. (2018b) Refinement	47.6	**42.5**	37.4	**31.9**

Table 3: Word translation accuracy (P@1) on the English-Italian and English-Spanish language pairs of Dinu-Artetxe dataset (Dinu et al., 2015; Artetxe et al., 2017). All methods use CBOW embeddings. ** indicates the model failed to converge; '-' indicates the authors did not report the number.

	En-It		En-Es	
	→	←	→	←
Dinu-Artetxe Dataset	**	**	**	**
Conneau Dataset	01.2	01.6	04.7	05.1

Table 4: Conneau et al. (2018) refinement applied to the initial mappings of Artetxe et al. (2018b). ** indicates the model failed to converge.

On the dataset of Dinu et al. (2015); Artetxe et al. (2017) in Table 3, our **Adversarial autoencoder + Conneau et al. (2018) Refinement** performs better than other methods except Artetxe et al. (2018b). On average our method lags behind by about 2%. However, as mentioned, they follow a different refinement and mapping methods. For non-European and low-resource language pairs in Table 2, our **Adversarial autoencoder + Conneau et al. (2018) Refinement** exhibits better performance than others in one translation task, where the model of Artetxe et al. (2018b) performs better in the rest. One important thing to notice here is that other unsupervised models (apart from ours and Artetxe et al. (2018b)) fail to converge in one or more language pairs.

We notice that the method of Artetxe et al. (2018b) gives better results than other baselines, even in some translation tasks they achieve the highest accuracy. To understand whether the improvements of their method are due to a better initial mapping or better post-processing, we conducted two additional experiments. In our first experiment, we use their method to induce the initial seed dictionary and then apply iterative Procrustes solution (same refinement procedure of Conneau et al. (2018)) for refinement. Table 4 shows the results. Surprisingly, on both datasets their initial mappings fail to produce any reasonable results. So we suspect that the main gain in (Artetxe et al., 2018b) comes from their fine-tuning method, which they call *robust self learning*. In our second experiment, we use the initial dictionary induced by our adversarial training and then apply their refinement procedure. Here for most of the translation tasks, we achieve better results; see the model **Adversarial autoencoder + Artetxe et al. (2018b) Refinement** in Tables 1 - 3. These two experiments demonstrate that the quality of the initial dictionary induced by our model is far better than that of Artetxe et al. (2018b).

5.3 Model Dissection

We further analyze our model by dissecting it and measuring the contribution of each novel component that is proposed in this work. We achieve this by *incrementally* removing a new component from the model and evaluating it on different translation tasks. In order to better understand the contribution of each component, we evaluate each model by measuring its **P@1**, **P@5**, and **P@10 with fine-tuning** and **without fine-tuning**. In case of **without fine-tuning**, the models apply the CSLS neighbor search directly on the mappings learned from the adversarial training, *i.e.,* no Procrustes solution based refinement is done after the adversarial training. This setup allows us to compare our model directly with the adversarial model of Conneau et al. (2018), putting the effect of fine-tuning aside.

Table 5 presents the ablation results for En-Es, En-De, and En-It in both directions. The first row (**Conneau-18**) presents the results of Conneau et al. (2018) that uses adversarial training to map the *word embeddings*. The next row shows the results of **our full** model. The subsequent rows incrementally detach one component from our model. For example, **- Enc. adv** denotes the variant of our model where the target encoder is not trained on the adversarial loss ($\theta_{E_{\mathcal{X}}}$ in Eq. 4); **- - Recon** excludes the post-cycle reconstruction loss from **- Enc. adv**, and **- - - Cycle** excludes the cycle consistency from **- - Recon**. Thus, **- - - Cycle** is a variant of our model that uses only adversarial loss to learn the mapping. However, it is important

	En→Es			Es→En			En→De			De→En			En→It			It→En		
	P@1	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**
	Without Fine-Tuning																	
Conneau-18	65.3	73.8	80.6	66.7	78.3	80.8	61.5	70.1	78.2	60.3	70.2	77.0	64.8	75.3	79.4	63.8	77.1	81.8
Our (full)	71.8	81.1	85.7	72.7	81.5	83.8	64.9	74.4	81.8	63.1	71.3	79.8	68.2	78.9	83.7	67.5	77.6	82.1
- Enc. adv	70.5	79.7	83.5	71.3	80.4	83.3	63.7	73.5	79.3	62.6	70.5	79.0	67.6	77.3	82.7	66.2	78.3	82.5
- - Recon	70.1	78.9	83.4	70.8	81.1	83.4	63.1	73.8	80.5	62.2	71.7	78.7	66.9	79.7	82.1	64.8	78.6	82.1
- - - Cycle	66.8	76.5	82.1	67.2	79.9	82.7	61.4	69.7	77.8	60.1	69.8	76.5	65.3	75.1	78.9	64.4	77.6	81.7
	With Fine-Tuning																	
Conneau-18	82.3	90.8	93.2	83.7	91.9	93.5	74.2	89.0	91.5	72.6	85.7	88.8	78.3	88.4	91.1	78.1	88.2	90.6
Our (full)	82.6	91.8	93.5	84.4	92.3	94.3	75.5	90.1	92.9	73.9	86.5	89.3	78.8	89.2	91.9	78.5	88.9	91.1
- Enc. adv	82.5	91.6	93.5	84.3	92.1	94.3	75.4	89.7	92.7	73.5	86.3	89.2	78.4	89.0	91.8	78.1	88.7	91.0
- - Recon	82.5	91.6	93.4	84.1	92.2	94.3	75.3	89.4	92.6	73.2	85.9	89.0	78.2	89.1	91.9	78.2	88.8	91.2
- - - Cycle	82.4	91.0	93.1	83.6	92.2	94.0	74.3	89.7	92.6	72.7	86.1	89.1	77.8	89.2	91.8	77.4	88.3	90.8

Table 5: Ablation study of our adversarial autoencoder model on the dataset of Conneau et al. (2018).

to note that in contrast to Conneau et al. (2018), our mapping is performed at the code space.

As we compare our full model with the model of Conneau et al. (2018) in the *without fine-tuning* setting, we notice large improvements in all measures across all datasets: 5.1 - 7.3% in En→Es, 3 - 6% in Es→En, 3.4 - 4.3% in En→De, 1 - 3% in De→En, 3.4 - 4.3% in En→It, and 0.3 - 3.7% in It→En. These improvements demonstrate that our model finds a better mapping compared to Conneau et al. (2018). Among the three components, the cycle consistency is the most influential one across all languages. Training the target encoder adversarially also gives a significant boost. The reconstruction has less impact. If we compare the results of - - - *Cycle* with *Conneau-18*, we see sizeable gains for En-Es in both directions. This shows the benefits of mapping at the code level.

Now let us turn our attention to the results with fine-tuning. Here also we see gains across all datasets for our model, although the gains are not as verbose as before (about 1% on average). However, this is not surprising as it has been shown that *iterative fine-tuning* with Procrustes solution is a robust method that can recover many errors made in the initial mapping (Conneau et al., 2018). Given a good enough initial mapping, the measures converge nearly to the same point even though the differences were comparatively more substantial initially; for example, notice that the scores are very similar for P@5 and P@10 measures after fine-tuning.

6 Conclusions

We have proposed an adversarial autoencoder framework to learn the cross-lingual mapping of monolingual word embeddings of two languages in a completely unsupervised way. In contrast to the existing methods that directly map word embeddings, our method first learns to transform the embeddings into latent code vectors by pretraining an autoencoder. We apply adversarial training to map the distributions of the source and target code vectors. In our adversarial training, both the mapper and the target encoder are treated as generators that act jointly to fool the discriminator. To guide the mapping further, we include constraints for cycle consistency and post-cycle reconstruction.

Through extensive experimentations on six different language pairs comprising European, non-European and low-resource languages from two different data sources, we demonstrate that our method outperforms the method of Conneau et al. (2018) for all translation tasks in all measures (P@{1,5,10}) across all settings (with and without fine-tuning). Comparison with other existing methods also shows that our method learns better mapping (not considering the fine-tuning). With an ablation study, we further demonstrated that the cycle consistency is the most important component followed by the adversarial training of target encoder and the post-cycle reconstruction. In future work, we plan to incorporate knowledge from the similarity space in our adversarial framework.

Acknowledgments

The authors would like to thank the funding support from MOE Tier-1 (Grant M4011897.020).

References

David Alvarez and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890. Association for Computational Linguistics.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *ICLR, Workshop track*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471. Association for Computational Linguistics.

Ian J. Goodfellow. 2017. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160.

Yedid Hoshen and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159. Association for Computational Linguistics.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. 2015. Adversarial autoencoders. *CoRR*, abs/1511.05644.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126. Association for Computational Linguistics.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Sebastian Ruder, Ivan Vulic, and Anders Sogaard. 2017. A survey of cross-lingual word embedding models.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations (ICLR)*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *HLT-NAACL*, pages 1006–1011. The Association for Computational Linguistics.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018a. Unsupervised cross-lingual transfer of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474. Association for Computational Linguistics.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018b. Unsupervised cross-lingual transfer of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945. Association for Computational Linguistics.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

Addressing Word-order Divergence in Multilingual Neural Machine Translation for Extremely Low Resource Languages

Rudra Murthy V[†], **Anoop Kunchukuttan**[‡], **Pushpak Bhattacharyya**[†]
[†] Center for Indian Language Technology (CFILT)
Department of Computer Science and Engineering
IIT Bombay, India.
[‡]Microsoft AI & Research, Hyderabad, India.
`{rudra,pb}@cse.iitb.ac.in`, `ankunchu@microsoft.com`

Abstract

Transfer learning approaches for Neural Machine Translation (NMT) trains a NMT model on an assisting language-target language pair (parent model) which is later fine-tuned for the source language-target language pair of interest (child model), with the target language being the same. In many cases, the assisting language has a different word order from the source language. We show that divergent word order adversely limits the benefits from transfer learning when little to no parallel corpus between the source and target language is available. To bridge this divergence, we propose to pre-order the assisting language sentences to match the word order of the source language and train the parent model. Our experiments on many language pairs show that bridging the word order gap leads to major improvements in the translation quality in extremely low-resource scenarios.

1 Introduction

Transfer learning for multilingual Neural Machine Translation (NMT) (Zoph et al., 2016; Dabre et al., 2017; Nguyen and Chiang, 2017) attempts to improve the NMT performance on the *source* to *target* language pair (child task) using an *assisting source* language (assisting to target language translation is the parent task). Here, the parent model is trained on the assisting and target language parallel corpus and the trained weights are used to initialize the child model. If source-target language pair parallel corpus is available, the child model can further be fine-tuned. The weight initialization reduces the requirement on the training data for the source-target language pair by transferring knowledge from the parent task, thereby improving the performance on the child task.

However, the divergence between the source and the assisting language can adversely impact the benefits obtained from transfer learning. Multiple studies have shown that transfer learning works best when the languages are related (Zoph et al., 2016; Nguyen and Chiang, 2017; Dabre et al., 2017). Zoph et al. (2016) studied the influence of language divergence between languages chosen for training the parent and the child model, and showed that choosing similar languages for training the parent and the child model leads to better improvements from transfer learning.

Several studies have tried to address the *lexical divergence* between the source and the target languages either by using Byte Pair Encoding (BPE) as basic input representation units (Nguyen and Chiang, 2017) or character-level NMT system (Lee et al., 2017) or bilingual embeddings (Gu et al., 2018). However, the effect of *word order divergence* and its mitigation has not been explored. In a practical setting, it is not uncommon to have source and assisting languages with different word order. For instance, it is possible to find parallel corpora between English (SVO word order) and some Indian (SOV word order) languages, but very little parallel corpora between Indian languages. Hence, it is natural to use English as an assisting language for inter-Indian language translation.

To address the word order divergence, we propose to pre-order the assisting language sentences (SVO) to match the word order of the source language (SOV). We consider an extremely resource-constrained scenario, where there is no parallel corpus for the child task. From our experiments, we show that there is a significant increase in the translation accuracy for the unseen source-target language pair.

2 Related Work

To the best of our knowledge, no work has addressed word order divergence in transfer learning

Proceedings of NAACL-HLT 2019, pages 3868–3873
Minneapolis, Minnesota, June 2 - June 7, 2019.

for multilingual NMT. However, some work exists for other NLP tasks in a multilingual setting. For Named Entity Recognition (NER), Xie et al. (2018) use a self-attention layer after the Bi-LSTM layer to address word-order divergence for Named Entity Recognition (NER) task. The approach does not show any significant improvements, possibly because the divergence has to be addressed before/during construction of the contextual embeddings in the Bi-LSTM layer. Joty et al. (2017) use adversarial training for cross-lingual question-question similarity ranking. The adversarial training tries to force the sentence representation generated by the encoder of similar sentences from different input languages to have similar representations.

Pre-ordering the source language sentences to match the target language word order has been found useful in addressing word-order divergence for Phrase-Based SMT (Collins et al., 2005; Ramanathan et al., 2008; Navratil et al., 2012; Chatterjee et al., 2014). For NMT, Ponti et al. (2018) and Kawara et al. (2018) have explored pre-ordering. Ponti et al. (2018) demonstrated that by reducing the syntactic divergence between the source and the target languages, consistent improvements in NMT performance can be obtained. On the contrary, Kawara et al. (2018) reported drop in NMT performance due to pre-ordering. Note that these works address source-target divergence, not divergence between source languages in multilingual NMT scenario.

3 Proposed Solution

Consider the task of translating for an extremely low-resource language pair. The parallel corpus between the two languages, if available may be too small to train an NMT model. Similar to Zoph et al. (2016), we use transfer learning to overcome data sparsity between the source and the target languages. We choose *English* as the assisting language in all our experiments. In our resource-scarce scenario, we have no parallel corpus for training the child model. Hence, at test time, the source language sentence is translated using the parent model after performing a word-by-word translation from source to the assisting language using a bilingual dictionary.

Since the source language and the assisting language (English) have different word order, we hypothesize that it leads to inconsistencies in the contextual representations generated by the encoder for the two languages. Specifically, given an English sentence (SVO word order) and its translation in the source language (SOV word order), the encoder representations for words in the two sentences will be different due to different contexts of synonymous words. This could lead to the attention and the decoder layers generating different translations from the same (parallel) sentence in the source or assisting language. This is undesirable as we want the knowledge to be transferred from the parent model (assisting source→ target) to the child model (source→target).

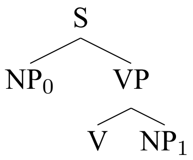
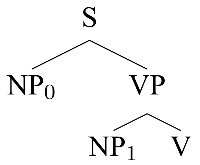
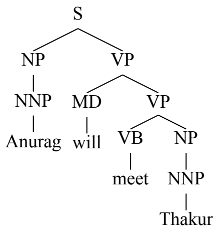
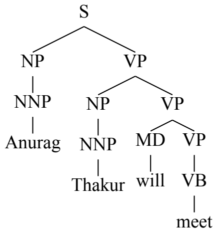
Before Reordering	**After Reordering**
	
	

Table 1: Example showing transitive verb before and after reordering (Adapted from Chatterjee et al. (2014))

In this paper, we propose to pre-order English sentences (assisting language sentences) to match the source language word-order and train the parent model on the pre-ordered corpus. Table 1 shows one of the pre-ordering rules (Ramanathan et al., 2008) used along with an example sentence illustrating the effect of pre-ordering. This will ensure that context of words in the parallel source and assisting language sentences are similar, leading to consistent contextual representations across the source languages. Pre-ordering may also be beneficial for other word order divergence scenarios (*e.g.,* SOV to SVO), but we leave verification of these additional scenarios for future work.

4 Experimental Setup

In this section, we describe the languages experimented with, datasets used, the network hyper-parameters used in our experiments.

Languages: We experimented with English → Hindi translation as the parent task. English is the assisting source language. Bengali, Gujarati, Marathi, Malayalam and Tamil are the source languages, and translation from these to Hindi constitute the child tasks. Hindi, Bengali, Gujarati and Marathi are Indo-Aryan languages, while Malayalam and Tamil are Dravidian languages. All these languages have a canonical SOV word order.

Datasets: For training English-Hindi NMT systems, we use the IITB English-Hindi parallel corpus (Kunchukuttan et al., 2018) ($1.46M$ sentences from the training set) and the ILCI English-Hindi parallel corpus ($44.7K$ sentences). The ILCI (Indian Language Corpora Initiative) multilingual parallel corpus (Jha, 2010)[1] spans multiple Indian languages from the health and tourism domains. We use the 520-sentence dev-set of the IITB parallel corpus for validation. For each child task, we use $2K$ sentences from ILCI corpus as test set.

Network: We use OpenNMT-Torch (Klein et al., 2018) to train the NMT system. We use the standard encoder-attention-decoder architecture (Bahdanau et al., 2015) with input-feeding approach (Luong et al., 2015). The encoder has two layers of bidirectional LSTMs with 500 neurons each and the decoder contains two LSTM layers with 500 neurons each. We use a mini-batch of size 50 and a dropout layer. We begin with an initial learning rate of 1.0 and continue training with exponential decay till the learning rate falls below 0.001. The English input is initialized with pre-trained *fastText* embeddings (Grave et al., 2018) [2].

English and Hindi vocabularies consists of $0.27M$ and $50K$ tokens appearing at least 2 and 5 times in the English and Hindi training corpus respectively. For representing English and other source languages into a common space, we translate each word in the source language into English using a bilingual dictionary (we used *Google Translate* to get single word translations). In an end-to-end solution, it would be ideal to use bilingual embeddings or obtain word-by-word translations *via* bilingual embeddings (Xie et al., 2018). However, publicly available bilingual embeddings for English-Indian languages are not good enough for obtaining good-quality, bilingual representations (Smith et al., 2017; Jawanpuria et al., 2019) and publicly available bilingual dictionaries have limited coverage. The focus of our study is the influence of word-order divergence on Multilingual NMT. We do not want bilingual embeddings quality or bilingual dictionary coverage to influence the experiments, rendering our conclusions unreliable. Hence, we use the above mentioned large-coverage bilingual dictionary.

Language	BLEU			LeBLEU		
	No Pre-Order	Pre-Ordered		No Pre-Order	Pre-Ordered	
		HT	G		HT	G
Bengali	6.72	8.83	**9.19**	37.10	41.50	**42.01**
Gujarati	9.81	**14.34**	13.90	43.21	47.36	**47.60**
Marathi	8.77	10.18	**10.30**	40.21	41.49	**42.22**
Malayalam	5.73	6.49	**6.95**	33.27	33.69	**35.09**
Tamil	4.86	**6.04**	6.00	29.38	30.77	**31.33**

Table 2: Transfer learning results for *X-Hindi* pair, trained on *English-Hindi* corpus and sentences from *X* word translated to English.

Language	No Pre-Order	Pre-Ordered	
		HT	G
Bengali	1324	1139	1146
Gujarati	1337	1190	1194
Marathi	1414	1185	1178
Malayalam	1251	1067	1059
Tamil	1488	1280	1252

Table 3: Number of UNK tokens generated by each model on the test set.

Pre-ordering: We use *CFILT-preorder*[3] for pre-reordering English sentences. It contains two pre-ordering configurations: (1) *generic* rules (G) that apply to all Indian languages (Ramanathan et al., 2008), and (2) *hindi-tuned* rules (HT) which improves generic rules by incorporating improvements found through error analysis of English-Hindi reordering (Patel et al., 2013). The Hindi-tuned rules improve translation for other English to Indian language pairs too (Kunchukuttan et al., 2014).

5 Results

We experiment with two scenarios: (a) an extremely resource scarce scenario with no parallel corpus for child tasks, (b) varying amounts of parallel corpora available for child task.

5.1 No Parallel Corpus for Child Task

The results from our experiments are presented in the Table 2. We report BLEU scores and LeBLEU[4]

---

[1] The corpus is available on request from `http://tdil-dc.in/index.php?lang=en`

[2] `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

[3] `https://github.com/anoopkunchukuttan/cfilt_preorder`

[4] LeBLEU (Levenshtein Edit BLEU) is a variant of BLEU that does a soft-match of reference and output words based

English	the treatment of migraine is done in two ways								
Gujarati (Original)	માઇગ્રેનની	સારવાર	બે	રીતે	કરી	શકાય	છે.		
Gujarati (Word Translate)	migraine	treatment	two	the way	doing be	done	is there .		
Hindi (Reference)	माइग्रेन	का	ट्रीटमेंट	दो	तरह	से	किया	जाता	है ।
(Word Translate)	migraine	of	treatment	two	kind	from	did	go	is .
No Pre-Order	<unk>	उपचार	दो	प्रकार	से	किया	जाता	है	।
		upachAra	do	prakAra	se	kiyA	jAtA	hai	.
	<unk>	treatment	two	kind	from	did	go	is	.
Pre-ordered (HT)	माइग्रेन	का	उपचार	दो	तरह	से	किया	जाता	है।
	mAigrena	kA	upachAra	do	prakAra	se	kiyA	jAtA	hai.
	migraine	of	treatment	two	kind	from	did	go	is .

Table 4: Sample Hindi translation generated by the Gujarati-Hindi NMT model. Text in red indicates phrase dropped by the no pre-ordered model.

scores. We observe that both the pre-ordering models significantly improve the translation quality over the no-preordering models for all the language pairs. The results support our hypothesis that word-order divergence can limit the benefits of multilingual translation. Thus, reducing the word order divergence improves translation in extremely low-resource scenarios.

An analysis of the outputs revealed that pre-ordering significantly reduced the number of UNK tokens (placeholder for unknown words) in the test output (Table 3). We hypothesize that due to word order divergence between English and Indian languages, the encoder representation generated is not consistent leading to decoder generating unknown words. However, the pre-ordered models generate better encoder representations leading to lesser number of UNK tokens and better translation, which is also reflected in the BLEU scores and Table 4.

5.2 Parallel Corpus for Child Task

We study the impact of child task parallel corpus on pre-ordering. To this end, we fine-tune the parent task model with the child task parallel corpus. Table 5 shows the results for *Bengali-Hindi*, *Gujarati-Hindi*, *Marathi-Hindi*, *Malayalam-Hindi*, and *Tamil-Hindi* translation. We observe that pre-ordering is beneficial when almost no child task corpus is available. As the child task corpus increases, the model learns the word order of the source language; hence, the non pre-ordering models perform almost as good as or sometimes better than the pre-ordered ones. The non pre-ordering model is able to forget the word-order of English and learn the word order of Indian languages. We attribute this behavior of the non pre-ordered model to the phenomenon of catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999) which enables the model to learn the word-order of the source language when sufficient child task parallel corpus is available.

We also compare the performance of the fine-tuned model with the model trained only on the available source-target parallel corpus with randomly initialized weights (No Transfer Learning). Transfer learning, with and without pre-ordering, is better compared to training only on the small source-target parallel corpus.

6 Conclusion

In this paper, we show that handling word-order divergence between the source and assisting languages is crucial for the success of multilingual NMT in an extremely low-resource setting. We show that pre-ordering the assisting language to match the word order of the source language significantly improves translation quality in an extremely low-resource setting. If pre-ordering is not possible, fine-tuning on a small source-target parallel corpus is sufficient to overcome word order divergence. While the current work focused on Indian languages, we would like to validate the hypothesis on a more diverse set of languages. We

---

on edit distance, hence it can handle morphological variations and cognates (Virpioja and Grönroos, 2015).

Corpus Size	No Transfer Learning	No Pre-Order	Pre-Ordered	
			HT	G
Bengali				
-	-	6.72	8.83	**9.19**
500	0.0	11.40	**11.49**	11.00
1000	0.0	13.71	**13.84**	13.62
2000	0.0	16.41	**16.79**	16.01
3000	0.0	17.44	**18.42**†	17.82
4000	0.0	18.86	**19.17**	18.66
5000	0.07	19.58	**20.15**†	19.82
10000	1.87	22.50	**22.92**	22.53
Gujarati				
-	-	9.81	**14.34**	13.90
500	0.0	17.27	17.11	**17.75**
1000	0.0	21.68	**22.12**	21.45
2000	0.0	25.34	**25.73**	25.63
3000	0.29	27.48	27.77	**27.83**
4000	0.82	29.20	29.49	**29.51**
5000	0.0	29.87	**31.09**†	30.58†
10000	1.52	33.97	**34.25**	34.08
Marathi				
-	-	8.77	10.18	**10.30**
500	0.0	12.84	**13.61**†	12.97
1000	0.0	15.62	15.75	**16.10**†
2000	0.0	18.59	**19.10**	18.67
3000	0.0	20.51	**20.76**	20.29
4000	0.24	**21.78**	21.77	21.39
5000	0.29	22.21	22.41	**22.73**†
10000	7.90	25.16	**25.88**	25.36
Malayalam				
-	-	5.73	6.49	**6.95**
500	0.0	5.40	5.54	**6.17**†
1000	0.0	7.34	7.36	**7.63**
2000	0.0	8.24	**8.66**†	8.31
3000	0.0	9.11	9.30	**9.31**
4000	0.0	9.65	**9.91**	9.87
5000	0.03	10.26	**10.47**	10.28
10000	0.0	**11.96**	11.85	11.63
Tamil				
-	-	4.86	**6.04**	6.00
500	0.0	5.49	**5.85**†	5.59
1000	0.0	7.04	7.23	**7.44**†
2000	0.0	8.83	8.84	**9.24**
3000	0.0	9.80	**10.04**	9.56
4000	0.0	9.69	**10.59**†	10.25†
5000	0.03	10.84	**10.93**	10.69
10000	0.0	12.71	**13.05**	12.69

Table 5: Transfer learning results (BLEU) for *Indian Language-Hindi* pair, fine-tuned with varying number of *Indian Language-Hindi* parallel sentences. †Indicates statistically significant difference between *Pre-ordered* and *No Pre-ordered* results using paired bootstrap resampling (Koehn, 2004) for a p-value less than 0.05. *No Transfer Learning* model refers to training the model on varying number of *Indian Language-Hindi* parallel sentences with randomly initialized weights.

would also like to explore alternative methods to address word-order divergence which do not require expensive parsing of the assisting language corpus. Further, use of pre-ordering to address word-order divergence for multilingual training of other NLP tasks can be explored.

Acknowledgements

We would like to thank Raj Dabre for his helpful suggestions and comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations, ICLR 2015*.

Rajen Chatterjee, Anoop Kunchukuttan, and Pushpak Bhattacharyya. 2014. Supertag Based Pre-ordering in Machine Translation. In *Proceedings of the 11th International Conference on Natural Language Processing, ICON 2014*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, ACL 2005*.

Raj Dabre, Tetsuji Nakagawa, and Hideto Kazawa. 2017. An Empirical Study of Language Relatedness for Transfer Learning in Neural Machine Translation. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal Neural Machine Translation for Extremely Low Resource Languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), NAACL 2018*.

Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, and Bamdev Mishra. 2019. Learning Multilingual Word Embeddings in Latent Metric Space: A Geometric Approach. *Transactions of the Association for Computational Linguistics, TACL*.

Girish Nath Jha. 2010. The TDIL program and the Indian Langauge Corpora Intitiative (ILCI). In *Proceedings of the Seventh conference on International Language Resources and Evaluation, LREC 2010*.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language Learning with Adversarial Neural Networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning, CoNLL 2017*.

Yuki Kawara, Chenhui Chu, and Yuki Arase. 2018. Recursive Neural Network Based Preordering for English-to-Japanese Machine Translation. In *Proceedings of ACL 2018, Student Research Workshop*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, and Alexander Rush. 2018. OpenNMT: Neural Machine Translation Toolkit. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi Parallel Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018*.

Anoop Kunchukuttan, Abhijit Mishra, Rajen Chatterjee, Ritesh Shah, and Pushpak Bhattacharyya. 2014. Shata-Anuvadak: Tackling Multiway Translation of Indian Languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics, TACL*, 5.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. volume 24 of *Psychology of Learning and Motivation*, pages 109 – 165. Academic Press.

Jiri Navratil, Karthik Visweswariah, and Ananthakrishnan Ramanathan. 2012. A Comparison of Syntactic Reordering Methods for English-German Machine Translation. In *Proceedings of COLING 2012, The 24th International Conference on Computational Linguistics, COLING 2012*.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), IJCNLP 2017*.

Raj Nath Patel, Rohit Gupta, Prakash B. Pimpale, and Sasikumar M. 2013. Reordering rules for English-Hindi SMT. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018. Isomorphic transfer of syntactic structures in cross-lingual nlp. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2018*.

Ananthakrishnan Ramanathan, Jayprasad Hegde, Ritesh M. Shah, Pushpak Bhattacharyya, and Sasikumar M. 2008. Simple Syntactic and Morphological Processing Can Help English-Hindi Statistical Machine Translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Aligning the fastText vectors of 78 languages.

Sami Virpioja and Stig-Arne Grönroos. 2015. LeBLEU: N-gram-based Translation Evaluation Score for Morphologically Complex Languages. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell. 2018. Neural Cross-Lingual Named Entity Recognition with Minimal Resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*.

Massively Multilingual Neural Machine Translation

Roee Aharoni[*]
Bar Ilan University
Ramat-Gan
Israel
roee.aharoni@gmail.com

Melvin Johnson and **Orhan Firat**
Google AI
Mountain View
California
melvinp,orhanf@google.com

Abstract

Multilingual neural machine translation (NMT) enables training a single model that supports translation from multiple source languages into multiple target languages. In this paper, we push the limits of multilingual NMT in terms of the number of languages being used. We perform extensive experiments in training massively multilingual NMT models, translating up to 102 languages to and from English within a single model. We explore different setups for training such models and analyze the trade-offs between translation quality and various modeling decisions. We report results on the publicly available TED talks multilingual corpus where we show that massively multilingual many-to-many models are effective in low resource settings, outperforming the previous state-of-the-art while supporting up to 59 languages. Our experiments on a large-scale dataset with 102 languages to and from English and up to one million examples per direction also show promising results, surpassing strong bilingual baselines and encouraging future work on massively multilingual NMT.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Sutskever et al., 2014) is the current state-of-the-art approach for machine translation in both academia (Bojar et al., 2016, 2017, 2018) and industry (Wu et al., 2016; Hassan et al., 2018). Recent works (Dong et al., 2015; Firat et al., 2016a; Ha et al., 2016; Johnson et al., 2017) extended the approach to support multilingual translation, i.e. training a single model that is capable of translating between multiple language pairs.

Multilingual models are appealing for several reasons. First, they are more efficient in terms of the number of required models and model parameters, enabling simpler deployment. Another benefit is transfer learning; when low-resource language pairs are trained together with high-resource ones, the translation quality may improve (Zoph et al., 2016; Nguyen and Chiang, 2017). An extreme case of such transfer learning is zero-shot translation (Johnson et al., 2017), where multilingual models are able to translate between language pairs that were never seen during training.

While very promising, it is still unclear how far one can scale multilingual NMT in terms of the number of languages involved. Previous works on multilingual NMT typically trained models with up to 7 languages (Dong et al., 2015; Firat et al., 2016b; Ha et al., 2016; Johnson et al., 2017; Gu et al., 2018) and up to 20 trained directions (Cettolo et al., 2017) simultaneously. One recent exception is Neubig and Hu (2018) who trained many-to-one models from 58 languages into English. While utilizing significantly more languages than previous works, their experiments were restricted to many-to-one models in a low-resource setting with up to 214k examples per language-pair and were evaluated only on four translation directions.

In this work, we take a step towards practical "universal" NMT – training massively multilingual models which support up to 102 languages and with up to one million examples per language-pair simultaneously. Specifically, we focus on training "English-centric" many-to-many models, in which the training data is composed of many language pairs that contain English either on the source side or the target side. This is a realistic setting since English parallel data is widely available for many language pairs. We restrict our experiments to Transformer models (Vaswani et al., 2017) as they were shown to be very effective in recent benchmarks (Ott et al., 2018), also in

---

[*] Work carried out during an internship at Google AI.

Proceedings of NAACL-HLT 2019, pages 3874–3884
Minneapolis, Minnesota, June 2 - June 7, 2019.

the context of multilingual models (Lakew et al., 2018; Sachan and Neubig, 2018).

We evaluate the performance of such massively multilingual models while varying factors like model capacity, the number of trained directions (tasks) and low-resource vs. high-resource settings. Our experiments on the publicly available TED talks dataset (Qi et al., 2018) show that massively multilingual many-to-many models with up to 58 languages to-and-from English are very effective in low resource settings, allowing to use high-capacity models while avoiding overfitting and achieving superior results to the current state-of-the-art on this dataset (Neubig and Hu, 2018; Wang et al., 2019) when translating into English.

We then turn to experiment with models trained on 103 languages in a high-resource setting. For this purpose we compile an English-centric in-house dataset, including 102 languages aligned to-and-from English with up to one million examples per language pair. We then train a single model on the resulting 204 translation directions and find that such models outperform strong bilingual baselines by more than 2 BLEU averaged across 10 diverse language pairs, both to-and-from English. Finally, we analyze the trade-offs between the number of involved languages and translation accuracy in such settings, showing that massively multilingual models generalize better to zero-shot scenarios. We hope these results will encourage future research on massively multilingual NMT.

2 Low-Resource Setting: 59 Languages

2.1 Experimental Setup

The main question we wish to answer in this work is how well a single NMT model can scale to support a very large number of language pairs. The answer is not trivial: on the one hand, training multiple language pairs together may result in transfer learning (Zoph et al., 2016; Nguyen and Chiang, 2017). This may improve performance as we increase the number of language pairs, since more information can be shared between the different translation tasks, allowing the model to learn which information to share. On the other hand, adding many language pairs may result in a bottleneck; the model has a limited capacity while it needs to handle this large number of translation tasks, and sharing all parameters between the different languages can be sub-optimal (Wang et al., 2018) especially if they are not from the same typological language family (Sachan and Neubig, 2018).

We begin tackling this question by experimenting with the TED Talks parallel corpus compiled by Qi et al. (2018)[1], which is unique in that it includes parallel data from 59 languages. For comparison, this is significantly "more multilingual" than the data available from all previous WMT news translation shared task evaluations throughout the years – the latest being Bojar et al. (2016, 2017, 2018), which included 14 languages so far.[2]

We focus on the setting where we train "English-centric" models, i.e. training on all language pairs that contain English in either the source or the target, resulting in 116 translation directions. This dataset is also highly imbalanced, with language pairs including between 3.3k to 214k sentence pairs for training. Table 9 in the supplementary material details the languages and training set sizes for this dataset. Since the dataset is already tokenized we did not apply additional preprocessing other than applying joint subword segmentation (Sennrich et al., 2016) with 32k symbols.

Regarding the languages we evaluate on, we begin with the same four languages as Neubig and Hu (2018) – Azerbeijani (Az), Belarusian (Be), Galician (Gl) and Slovak (Sk). These languages present an extreme low-resource case, with as few as 4.5k training examples for Belarusian-English. In order to better understand the effect of training set size in these settings, we evaluate on four additional languages that have more than 167k training examples each – Arabic (Ar), German (De), Hebrew (He) and Italian (It).

2.2 Model Details

Using the same data, we trained three massively multilingual models: a many-to-many model which we train using all 116 translation directions with 58 languages to-and-from English, a one-to-many model from English into 58 languages, and a many-to-one model from 58 languages into English. We follow the method of Ha et al. (2016); Johnson et al. (2017) and add a target-language

---

[1] `github.com/neulab/word-embeddings-for-nmt`

[2] Chinese, Czech, English, Estonian, Finnish, French, German, Hindi, Hungarian, Latvian, Romanian, Russian, Spanish, Turkish. According to `http://www.statmt.org/wmtXX`

prefix token to each source sentence to enable many-to-many translation. These different setups enable us to examine the effect of the number of translation tasks on the translation quality as measured in BLEU (Papineni et al., 2002). We also compare our massively multilingual models to bilingual baselines and to two recently published results on this dataset (Neubig and Hu (2018); Wang et al. (2019)).

Regarding the models, we focused on the Transformer in the "Base" configuration. We refer the reader to Vaswani et al. (2017) for more details on the model architecture. Specifically, we use 6 layers in both the encoder and the decoder, with model dimension set at 512, hidden dimension size of 2048 and 8 attention heads. We also applied dropout at a rate of 0.2 in the following components: on the sum of the input embeddings and the positional embeddings, on the output of each sub-layer before added to the previous layer input (residual connection), on the inner layer output after the ReLU activation in each feed-forward sub-layer, and to the attention weight in each attention sub-layer. This results in a model with approximately 93M trainable parameters. For all models we used the inverse square root learning rate schedule from Vaswani et al. (2017) with learning-rate set at 3 and 40k warmup steps. All models are implemented in Tensorflow-Lingvo (Shen et al., 2019).

In all cases we report test results for the checkpoint that performed best on the development set in terms of BLEU. For the multilingual models we create a development set that includes examples we uniformly sample from a concatenation of all the individual language pair development sets, resulting in 13k development examples per model. Another important detail regarding multilingual training is the batching scheme. In all of our multilingual models we use heterogeneous batching, where each batch contains examples which are uniformly sampled from a concatenation of all the language pairs the model is trained on. Specifically, we use batches of 64 examples for sequences shorter than 69 tokens and batches of 16 examples for longer sequences. We did not use oversampling as the dataset is relatively small.

2.3 Results

We use tokenized BLEU in order to be comparable with Neubig and Hu (2018). Table 1 shows the results of our experiments when evaluating on the same language pairs as they did. The results under "Neubig & Hu 18" are their bilingual baselines and their best many-to-one models. Their many-to-one models use similar-language-regularization, i.e. fine-tuning a pre-trained many-to-one model with data from the language pair of interest together with data from a language pair that has a typologically-similar source language and more training data (i.e. Russian and Belarusian, Turkish and Azerbaijani). The results under "Ours" are our many-to-one and many-to-many models we trained identically in terms of model architecture and hyper-parameters.

We first note that our many-to-many model out-

	Az-En	Be-En	Gl-En	Sk-En	Avg.
# of examples	5.9k	4.5k	10k	61k	20.3k
Neubig & Hu 18					
baselines	2.7	2.8	16.2	24	11.42
many-to-one	11.7	18.3	29.1	28.3	21.85
Wang et al. 18	11.82	18.71	30.3	28.77	22.4
Ours					
many-to-one	11.24	18.28	28.63	26.78	21.23
many-to-many	**12.78**	**21.73**	**30.65**	**29.54**	**23.67**

Table 1: X→En test BLEU on the TED Talks corpus, for the language pairs from Neubig and Hu (2018)

	Ar-En	De-En	He-En	It-En	Avg.
# of examples	213k	167k	211k	203k	198.5k
baselines	27.84	30.5	**34.37**	33.64	31.59
many-to-one	25.93	28.87	30.19	32.42	29.35
many-to-many	**28.32**	**32.97**	33.18	**35.14**	**32.4**

Table 2: X→En test BLEU on the TED Talks corpus, for language pairs with more than 167k examples

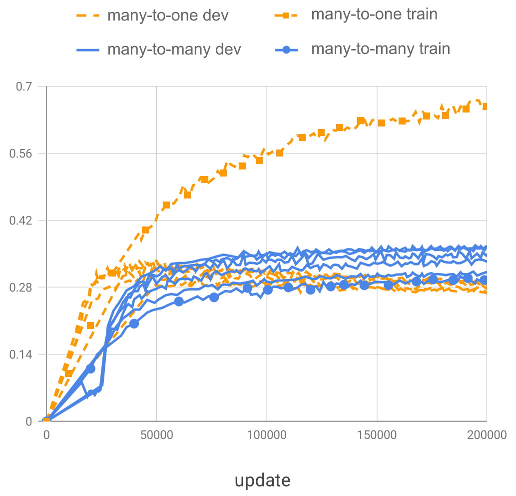


Figure 1: Development BLEU on {It,Ro,Nl,De,Ar}→En vs. training BLEU for the many-to-one and many-to-many models. Best viewed in color.

performs all other models when translating into English, with 1.82 BLEU improvement (when averaged across the four language pairs) over the best fine-tuned many-to-one models of Neubig and Hu (2018) and 2.44 BLEU improvement over our many-to-one model when averaged across the four low-resource language pairs (Table 1). This is surprising as it uses the same X→En data, model architecture and capacity as our many-to-one model, while handling a heavier burden since it also supports 58 *additional* translation tasks (*from* English *into* 58 languages). Our models also outperform the more complex models of Wang et al. (2019) which use ”Soft Decoupled Encoding” for the input tokens, while our models use a simple subword segmentation.

One possible explanation is that the many-to-one model overfits the English side of the corpus as it is multi-way-parallel: in such setting the English sentences are overlapping across the different language pairs, making it much easier for the model to memorize the training set instead of generalizing (when enough capacity is available). On the other hand, the many-to-many model is trained on additional target languages other than English, which can act as regularizers for the X→En tasks, reducing such overfitting.

To further illustrate this, Figure 1 tracks the BLEU scores on the individual development sets during training for Italian (It), Romanian (Ro), Dutch (Nl), German (De) and Arabic (Ar) into English (left), together with BLEU scores on a subset of the training set for each model. We can see that while the many-to-one model degrades in performance on the development set, the many-to-many model still improves. Note the large gap in the many-to-one model between the training set BLEU and the development set BLEU, which points on the generalization issue that is not present in the many-to-many setting. We also note that our many-to-one model is on average 0.75 BLEU behind the best many-to-one models in Neubig and Hu (2018). We attribute this to the fact that their models are fine-tuned using similar-language-regularization while our model is not.

We find an additional difference between the results on the resource-scarce languages (Table 1) and the higher-resource languages (Table 2). Specifically, the bilingual baselines outperform the many-to-one models only in the higher-resource setting. This makes sense as in the low-resource setting the baselines have very few training examples to outperform the many-to-one models, while in the higher resource setting they have access to more training data. This corroborates the results of Gu et al. (2018) that showed the sensitivity of such models to similar low resource conditions and the improvements gained from using many-to-one models (however with much fewer language pairs).

	En-Az	En-Be	En-Gl	En-Sk	Avg.
# of examples	5.9k	4.5k	10k	61k	20.3k
baselines	2.16	2.47	3.26	5.8	3.42
one-to-many	**5.06**	**10.72**	**26.59**	**24.52**	**16.72**
many-to-many	3.9	7.24	23.78	21.83	14.19

	En-Ar	En-De	En-He	En-It	Avg.
# of examples	213k	167k	211k	203k	198.5k
baselines	12.95	23.31	23.66	30.33	22.56
one-to-many	**16.67**	**30.54**	**27.62**	**35.89**	**27.68**
many-to-many	14.25	27.95	24.16	33.26	24.9

Table 3: En→X test BLEU on the TED Talks corpus

Table 3 shows the results of our massively multilingual models and bilingual baselines when evaluated out-of-English. In this case we see an opposite trend: the many-to-many model performs worse than the one-to-many model by 2.53 BLEU on average. While previous works (Wang et al., 2018; Sachan and Neubig, 2018) discuss the phenomena of quality degradation in English-to-many settings, this shows that increasing the number of *source* languages also causes additional degradation in a many-to-many model. This degradation may be due to the English-centric setting: since most of the translation directions the model is trained on are into English, this leaves less capacity for the other target languages (while still performing better than the bilingual baselines on all 8 language pairs). We also note that in this case the results are consistent among the higher and lower resource pairs – the one-to-many model is better than the many-to-many model, which outperforms the bilingual baselines in all cases. This is unlike the difference we saw in the X→ En experiments since here we do not have the multi-way-parallel overfitting issue.

2.4 Discussion

From the above experiments we learn that NMT models can scale to 59 languages in a low-resource, imbalanced, English-centric setting, with the following observations: (1) massively multilingual many-to-many models outperform many-to-one and bilingual models with similar capacity and identical training conditions when averaged over 8 language pairs into English. We attribute this improvement over the many-to-one models to the multiple target language pairs which may act as regularizers, especially in this low-resource multi-way-parallel setting that is prone to memorization. (2) many-to-many models are inferior in performance when going out-of-English in comparison to a one-to-many model. We attribute this to English being over-represented in the English-centric many-to-many setting, where it appears as a target language in 58 out of 116 trained directions, which may harm the performance on the rest of the target languages as the model capacity is limited.[3]

It is important to stress the fact that we compared the different models under *identical training conditions* and did not perform extensive hyper-parameter tuning for each setting separately. However, we believe that such tuning may improve performance even further, as the diversity in each training batch is very different between the different settings. For example, while the baseline model batches include only one language in the source and one language in the target, the many-to-many model includes 59 languages in each side with a strong bias towards English. These differences may require tailored hyper-parameter choices for each settings (i.e. different batch sizes, learning rate schedules, dropout rates etc.) which would be interesting to explore in future work.

In the following experiments we investigate whether these observations hold using (1) an even larger set of languages, and (2) a much larger, balanced training corpus that is not multi-way-parallel.

3 High-Resource Setting: 103 Languages

3.1 Experimental Setup

In this setting we scale the number of languages and examples per language pair further when training a single massively multilingual model. Since we are not aware of a publicly available resource for this purpose, we construct an in-house dataset. This dataset includes 102 language pairs which we "mirror" to-and-from English, with up to one million examples per language pair. This results in 103 languages in total, and 204 translation directions which we train simultaneously. More details about this dataset are available in Table 4, and Table 10 in the supplementary material details all the languages in the dataset.[4]

Similarly to our previous experiments, we compare the massively multilingual models to bilingual baselines trained on the same data. We tokenize the data using an in-house tokenizer and then apply joint subword segmentation to achieve an open-vocabulary. In this setting we used a vocabulary of 64k subwords rather than 32k. Since the dataset contains 24k unique characters, a 32k symbol vocabulary will consist of mostly characters, thereby increasing the average sequence length. Regarding the model, for these experiments we use a larger Transformer model with 6 layers in both the encoder and the decoder, model dimension set to 1024, hidden dimension size of 8192, and 16 attention heads. This results in a model with approximately 473.7M parameters.[5] Since the model and data are much larger in this case, we used a dropout rate of 0.1 for our multilingual models and tuned it to 0.3 for our baseline models as it improved the translation quality on the development set.

We evaluate our models on 10 languages from different typological families: *Semitic* – Arabic (Ar), Hebrew (He), *Romance* – Galician (Gl), Italian (It), Romanian (Ro), *Germanic* – German (De), Dutch (Nl), *Slavic* – Belarusian (Be), Slovak (Sk) and *Turkic* – Azerbaijani (Az) and Turkish (Tr). We evaluate both to-and-from English, where each language pair is trained on up to one million examples. As in the previous experiment, we report test results from the model that performed best in terms of BLEU on the development set.

# of language pairs	102
examples per pair	
min	63,879
max	1,000,000
average	940,087
std. deviation	188,194
total # of examples	95,888,938

Table 4: Training set details for the 103 langauges corpus, X→En data.

---

[3] This issue may be alleviated by over-sampling the non-English-target pairs, but we leave this for future work.

[4] The average number of examples per language pair is 940k, as for 13 out of the 102 pairs we had less than one million examples available.

[5] This is larger than the Transformer "Big" configuration, which includes approximately 213M trained parameters.

	Ar	Az	Be	De	He	It	Nl	Ro	Sk	Tr	Avg.
baselines	23.34	16.3	21.93	30.18	31.83	**36.47**	36.12	34.59	25.39	27.13	28.33
many-to-one	**26.04**	**23.68**	**25.36**	35.05	**33.61**	35.69	**36.28**	36.33	28.35	**29.75**	**31.01**
many-to-many	22.17	21.45	23.03	**37.06**	30.71	35.0	36.18	**36.57**	**29.87**	27.64	29.97

Table 5: X→En test BLEU on the 103-language corpus

	Ar	Az	Be	De	He	It	Nl	Ro	Sk	Tr	Avg.
baselines	10.57	8.07	15.3	23.24	19.47	31.42	28.68	27.92	11.08	15.54	19.13
one-to-many	**12.08**	**9.92**	**15.6**	**31.39**	**20.01**	**33**	**31.06**	**28.43**	**17.67**	**17.68**	**21.68**
many-to-many	10.57	9.84	14.3	28.48	17.91	30.39	29.67	26.23	18.15	15.58	20.11

Table 6: En→X test BLEU on the 103-language corpus

3.2 Results

Table 5 describes the results when translating into English. First, we can see that both multilingual models perform better than the baselines in terms of average BLEU. This shows that massively multilingual many-to-many models can work well in realistic settings with millions of training examples, 102 languages and 204 jointly trained directions to-and-from English. Looking more closely, we note several different behaviors in comparison to the low-resource experiments on the TED Talks corpus. First, the many-to-one model here performs better than the many-to-many model. This shows that the previous result was indeed due to the pathologies of the low-resource dataset; when the training data is large enough and not multi-way-parallel there is no overfitting in the many-to-one model, and it outperforms the many-to-many model in most cases while they are trained identically.

One particular outlier in this case is German-to-English, where the many-to-one model is 2 BLEU points below the many-to-many model. We examine the BLEU score of this language pair on its dedicated German-English development set during training in the many-to-one model and find that it highly fluctuates. We then measure the performance on the test set for this language pair by choosing the best checkpoint on the dedicated German-English development set (instead of on the mixed multilingual development set) and find it to be 38.07, which is actually *higher* in 1 BLEU than the best result of the many-to-many model. This shows that while training many languages together, there is no "silver bullet": some languages may suffer from severe interference during training (i.e. a reduction of 3 BLEU in this case, from 38.07 to 35.05) while other languages continue to improve with more updates.

Table 6 describes the results when translating out-of-English. Again, both of the massively multilingual models perform better than the baselines when averaged across the 10 evaluated language pairs, while handling up to 102 languages to-and-from English and 204 translation tasks simultaneously. In this case the results are similar to those we observed on the TED talks corpus, where the one-to-many model performs better than the many-to-many model. Again, this advantage may be due to the one-to-many model handling a smaller number of tasks while not being biased towards English in the target side like the many-to-many model.

4 Analysis

The above results show that massively multilingual NMT is indeed possible in large scale settings and can improve performance over strong bilingual baselines. However, it was shown in a somewhat extreme case with more than 100 languages trained jointly, where we saw that in some cases the joint training may harm the performance for some language pairs (i.e. German-English above). In the following analysis we would like to better understand the trade-off between the number of languages involved and the translation accuracy while keeping the model capacity and training configuration fixed.

4.1 Multilinguality & Supervised Performance

We first study the effect of varying the number of languages on the translation accuracy in a supervised setting, where we focus on many-

	Ar-En	En-Ar	Fr-En	En-Fr	Ru-En	En-Ru	Uk-En	En-Uk	Avg.
5-to-5	**23.87**	**12.42**	**38.99**	**37.3**	29.07	**24.86**	**26.17**	16.48	**26.14**
25-to-25	23.43	11.77	38.87	36.79	**29.36**	23.24	25.81	**17.17**	25.8
50-to-50	23.7	11.65	37.81	35.83	29.22	21.95	26.02	15.32	25.18
75-to-75	22.23	10.69	37.97	34.35	28.55	20.7	25.89	14.59	24.37
103-to-103	21.16	10.25	35.91	34.42	27.25	19.9	24.53	13.89	23.41

Table 7: Supervised performance while varying the number of languages involved

	Ar-Fr	Fr-Ar	Ru-Uk	Uk-Ru	Avg.
5-to-5	1.66	4.49	3.7	3.02	3.21
25-to-25	1.83	**5.52**	**16.67**	4.31	7.08
50-to-50	**4.34**	4.72	15.14	**20.23**	**11.1**
75-to-75	1.85	4.26	11.2	15.88	8.3
103-to-103	2.87	3.05	12.3	18.49	9.17

Table 8: Zero-Shot performance while varying the number of languages involved

to-many models. We create four subsets of the in-house dataset by sub-sampling it to a different number of languages in each subset. In this way we create four additional English-centric datasets, containing 5, 25, 50 and 75 languages each to-and-from English. We make sure that each subset contains all the languages from the next smaller subsets – i.e. the 25 language subset contains the 5 language subset, the 50 language subset contains the 25 language subset and so on. We train a similar-capacity large Transformer model (with 473.7M parameters) on each of these subsets and measure the performance for each model on the 8 supervised language pairs from the smallest subset – {Arabic, French, Russian, Ukrainian}↔English. In this way we can analyze to what extent adding more languages improves or harms translation quality while keeping the model capacity fixed, testing the capacity vs. accuracy "saturation point".

Table 7 shows the results of this experiment, reporting the test results for the models that performed best on the multilingual development set. We can see that in most cases the best results are obtained using the 5-to-5 model, showing that there is indeed a trade off between the number of languages and translation accuracy when using a fixed model capacity and the same training setup. One may expect that the gaps between the different models should become smaller and even close with more updates, as the models with more languages see less examples per language in each batch, thus requiring more updates to improve in terms of BLEU. However, in our setting these gaps did not close even after the models converged, leaving 2.73 average BLEU difference between the 5-to-5 and the 103-to-103 model.

4.2 Multilinguality & Zero-Shot Performance

We then study the effect of the number of languages on zero-shot translation accuracy. Since we find zero-shot accuracy as an interesting measure for model generalization, we hypothesize that by adding more languages, the model is forced to create a more generalized representation to better utilize its capacity, which may improve zero-shot performance. We choose four language pairs for this purpose: Arabic↔French which are distant languages, and Ukrainian↔Russian which are similar. Table 8 shows the results of our models on these language pairs. For Arabic↔French the BLEU scores are very low in all cases, with the 50-to-50 and 25-to-25 models being slightly better than rest on Ar-Fr and Fr-Ar respectively. On Russian↔Ukrainian we see clear improvements when increasing the number of languages to more than five.

Figure 2 further illustrates this, showing the better generalization performance of the massively multilingual models under this zero-shot setting. While the zero-shot performance in this case is low and unstable for the 5-to-5 and 25-to-25 mod-

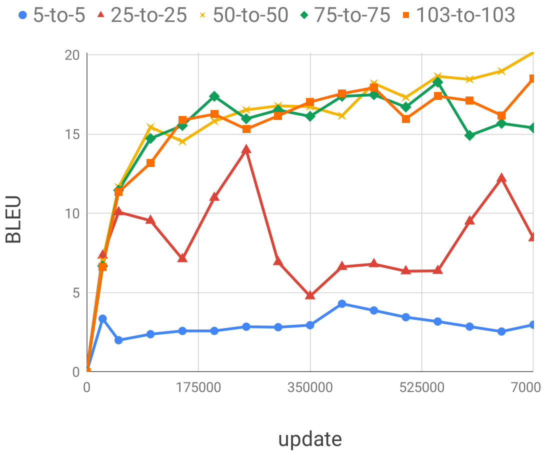


Figure 2: Zero-shot BLEU during training for Ukranian to Russian

els, it is much better for the 50-to-50, 75-to-75 and 103-to-103 models. Given these results we can say that the balance between capacity and generalization here favors the mid range 50-to-50 model, even when using models with more than 473M trained parameters. This may hint at the necessity of even larger models for such settings, which is a challenging avenue for future work. We also note that our 103 language corpus includes up to one million examples per language pair – while in real-world MT deployments, systems are trained on much more examples per pair. This again emphasizes the need for better techniques for training such massively multilingual models as we may already be hitting the capacity barrier in our setting.

5 Related Work

Dong et al. (2015) extended the NMT model of Bahdanau et al. (2014) to one-to-many translation (from English into 4 languages) by adding a dedicated decoder per target language, showing improvements over strong single-pair baselines. Firat et al. (2016a,b) proposed many-to-many models (with up to 6 languages) by using separate encoders and decoders per language while sharing the attention mechanism. They also introduced the notion of zero-resource translation, where they use synthetic training data generated through pivoting to train translation directions without available training data. Ha et al. (2016) and Johnson et al. (2017) proposed to use a shared encoder-decoder-attention model for many-to-many translation (with up to 7 languages in the latter). In order to determine the target language in such scenarios they proposed adding dedicated target-language symbols to the source. This method enabled zero-shot translation, showing the ability of the model to generalize to unseen pairs.

Recent works propose different methods for parameter sharing between language pairs in multilingual NMT. Blackwood et al. (2018) propose sharing all parameters but the attention mechanism and show improvements over sharing all parameters. Sachan and Neubig (2018) explore sharing various components in self-attentional (Transformer) models. Lu et al. (2018) add a shared "interlingua" layer while using separate encoders and decoders. Zaremoodi et al. (2018) utilize recurrent units with multiple blocks together with a trainable routing network. Platanios et al. (2018) propose to share the entire network, while using a contextual parameter generator that learns to generate the parameters of the system given the desired source and target languages. Gu et al. (2018) propose a "Universal Language Representation" layer together with a Mixture-of-Language-Experts component to improve a many-to-one model from 5 languages into English.

While the mentioned studies provide valuable contributions to improving multilingual models, they apply their models on only up to 7 languages (Johnson et al., 2017) and 20 trained directions (Cettolo et al., 2017) in a single model, whereas we focus on scaling NMT to much larger numbers of languages and trained directions. Regarding massively multilingual models, Neubig and Hu (2018) explored methods for rapid adaptation of NMT to new languages by training multilingual models on the 59-language TED Talks corpus and fine-tuning them using data from the new languages. While modeling significantly more languages than previous studies, they only train many-to-one models, which we show are inferior in comparison to our proposed massively multilingual many-to-many models when evaluated into English on this dataset.

Tiedemann (2018) trained an English-centric many-to-many model on translations of the bible including 927 languages. While this work pointed to an interesting phenomena in the latent space learned by the model where it clusters representations of typologically-similar languages together, it did not include any evaluation of the produced translations. Similarly, Malaviya et al. (2017) trained a many-to-English system including 1017 languages from bible translations, and used it to infer typological features for the different languages (without evaluating the translation quality). In another relevant work, Artetxe and Schwenk (2018) trained an NMT model on 93 languages and used the learned representations to perform cross-lingual transfer learning. Again, they did not report the performance of the translation model learned in that massively multilingual setting.

6 Conclusions and Future Work

We showed that NMT models can successfully scale to 102 languages to-and-from English with 204 trained directions and up to one million examples per direction. Such models improve the translation quality over similar single-pair baselines when evaluated to and from English by more than 2 BLEU when averaged over 10 diverse language pairs in each case. We show a similar result on the low-resource TED Talks corpus with 59 languages and 116 trained directions. We analyze the trade-offs between translation quality and the number of languages involved, pointing on capacity bottlenecks even with very large models and showing that massively multilingual models can generalize better to zero-shot settings.

We hope this work will encourage future research on massively multilingual NMT, enabling easier support for systems that can serve more people around the globe. There are many possible avenues for future work, including semi-supervised learning in such settings, exploring ways to reduce the performance degradation when increasing the number of languages, or using such models for multilingual transfer learning (McCann et al., 2017; Eriguchi et al., 2018; Artetxe and Schwenk, 2018). Understanding and improving zero-shot performance in such scenarios is also a promising direction for future work.

Acknowledgments

We would like to thank the Google Brain and Google Translate teams for their useful inputs and discussions. We would also like to thank the entire Lingvo development team for their foundational contributions to this project.

References

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Graeme Blackwood, Miguel Ballesteros, and Todd Ward. 2018. Multilingual neural machine translation with task-specific attention. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, et al. 2016. Findings of the 2016 conference on machine translation. In *ACL 2016 FIRST CONFERENCE ON MACHINE TRANSLATION (WMT16)*, pages 131–198.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*.

Mauro Cettolo, Federico Marcello, Bentivogli Luisa, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. 2017. Overview of the iwslt 2017 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-shot cross-lingual classification using multilingual neural machine translation. *arXiv preprint arXiv:1809.04686*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Hany Hassan, Anthony Aue, Chang Chen, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Melvin Johnson, Mike Schuster, Quoc V Le, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association of Computational Linguistics*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA.

Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. 2018. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA.

Yichao Lu, Phillip Keung, Faisal Ladhak, Vikas Bhardwaj, Shaonan Zhang, and Jason Sun. 2018. A neural interlingua for multilingual machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proc. IJCNLP*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ye Qi, Sachan Devendra, Felix Matthieu, Padmanabhan Sarguna, and Neubig Graham. 2018. When and why are pre-trained word embeddings useful for neural machine translation. In *HLT-NAACL*.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, et al. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jörg Tiedemann. 2018. Emerging language spaces learned from massively multilingual corpora. *arXiv preprint arXiv:1802.00273*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. In *International Conference on Learning Representations*.

Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. 2018. Three strategies to improve one-to-many multilingual translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

A Supplementary Material

Language	Train set size
Arabic	214111
Hebrew	211819
Russian	208458
Korean	205640
Italian	204503
Japanese	204090
Chinese-Taiwan	202646
Chinese-China	199855
Spanish	196026
French	192304
Portuguese-Brazil	184755
Dutch	183767
Turkish	182470
Romanian	180484
Polish	176169
Bulgarian	174444
Vietnamese	171995
German	167888
Persian	150965
Hungarian	147219
Serbian	136898
Greek	134327
Croatian	122091
Ukrainian	108495
Czech	103093
Thai	98064
Indonesian	87406
Slovak	61470
Swedish	56647
Portuguese	51785
Danish	44940
Albanian	44525
Lithuanian	41919
Macedonian	25335
Finnish	24222
Burmese	21497
Armenian	21360
French-Canadian	19870
Slovenian	19831
Hindi	18798
Norwegian	15825
Kannada	13193
Estonian	10738
Kurdish	10371
Galician	10017
Marathi	9840
Mongolian	7607
Esperanto	6535
Tamil	6224
Urdu	5977
Azerbaijani	5946
Bosnian	5664
Chinese	5534
Malay	5220
Basque	5182
Bengali	4649
Belarusian	4509
Kazakh	3317

Table 9: Language pairs in the TED talks dataset (58 languages, paired with English) with the train-set size for each pair.

Languages
Afrikaans
Albanian
Amharic
Arabic
Armenian
Azerbaijani
Basque
Belarusian
Bengali
Bosnian
Bulgarian
Burmese
Catalan
Cebuano
Chichewa*
Chinese
Corsican*
Croatian
Czech
Danish
Dutch
Esperanto
Estonian
Finnish
French
Frisian
Galician
Georgian
German
Greek
Gujarati
Haitian Creole
Hausa*
Hawaiian*
Hebrew
Hindi
Hmong*
Hungarian
Icelandic
Igbo
Indonesian
Irish
Italian
Japanese
Javanese
Kannada
Kazakh
Khmer
Korean
Kurdish
Kyrgyz
Laothian
Latin
Latvian
Lithuanian
Luxembourgish*
Macedonian
Malagasy
Malay
Malayalam
Maltese
Maori
Marathi
Mongolian
Nepali
Norwegian
Pashto
Persian
Polish
Portuguese
Punjabi
Romanian
Russian
Samoan*
Scots Gaelic*
Serbian
Sesotho
Shona*
Sindhi*
Sinhalese
Slovak
Slovenian
Somali
Spanish
Sundanese
Swahili
Swedish
Tagalog
Tajik*
Tamil
Telugu
Thai
Turkish
Ukrainian
Urdu
Uzbek
Vietnamese
Welsh
Xhosa
Yiddish
Yoruba*
Zulu

Table 10: Language pairs in the in-house dataset (102 languages, paired with English). For languages marked with * we had less than 1M examples, while for the rest we used exactly 1M.

A Large-Scale Comparison of Historical Text Normalization Systems

Marcel Bollmann
Department of Computer Science
University of Copenhagen, Denmark
`marcel@di.ku.dk`

Abstract

There is no consensus on the state-of-the-art approach to historical text normalization. Many techniques have been proposed, including rule-based methods, distance metrics, character-based statistical machine translation, and neural encoder–decoder models, but studies have used different datasets, different evaluation methods, and have come to different conclusions. This paper presents the largest study of historical text normalization done so far. We critically survey the existing literature and report experiments on eight languages, comparing systems spanning all categories of proposed normalization techniques, analysing the effect of training data quantity, and using different evaluation methods. The datasets and scripts are made publicly available.

1 Introduction[1]

Spelling variation is one of the key challenges for NLP on historical texts, affecting the performance of tools such as part-of-speech taggers or parsers and complicating users' search queries on a corpus. *Normalization* is often proposed as a solution; it is commonly defined as the mapping of historical variant spellings to a single, contemporary "normal form" as exemplified in Figure 1.

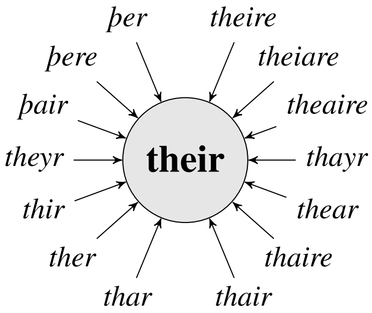


Figure 1: Historical text normalization exemplified: mapping variant spellings from historical English texts to their normalization *'their'*

Automatic normalization of historical texts has a long history, going back to at least Fix (1980). Earlier approaches often rely on hand-crafted algorithms tailored to one specific language, while more recent approaches have focused on supervised machine learning, particularly character-based statistical machine translation (SMT) and its neural equivalent (NMT). However, no clear consensus has emerged about the state of the art for this task, with papers either reporting an advantage for NMT (Hämäläinen et al., 2018), SMT (Domingo and Casacuberta, 2018), or language-specific algorithms (Schneider et al., 2017). Moreover, the quantity of annotated training data varies considerably between studies, making it difficult to obtain practical recommendations for new projects seeking to use normalization techniques.

Contributions This paper aims to provide the most comprehensive evaluation and analysis of historical text normalization systems so far. Motivated by a systematic review of previous work on this topic (Sec. 2), only publicly available normalization systems covering a wide range of proposed techniques are selected (Sec. 3) and evaluated across a diverse collection of historical datasets covering eight languages (Sec. 4). This is followed by a detailed analysis of the effect of training data quantity and a critical discussion of evaluation methods for assessing normalization quality (Sec. 5).

The datasets and code are made freely available whenever possible,[2] along with detailed instructions on how to reproduce the experiments.

---

[1] This work largely builds upon the author's doctoral thesis (Bollmann, 2018), the research for which was carried out at Ruhr-Universität Bochum, Germany.

[2] `https://github.com/coastalcph/histnorm`; one dataset could not be included due to licensing restrictions.

Proceedings of NAACL-HLT 2019, pages 3885–3898
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 A Brief Survey of Automatic Historical Text Normalization

The following overview is broadly organized by categories that each represent a conceptually or methodically different approach.

2.1 Substitution Lists

The conceptually simplest form of normalization is to look up each historical variant in a pre-compiled list that maps it to its intended normalization. This approach can go by many names, such as *lexical substitution*, *dictionary lookup*, *wordlist mapping*, or *memorization*. While it does not generalize in any way to variants that are not covered by the list, it has proven highly effective as a component in several normalization systems, such as the semi-automatic VARD tool (Rayson et al., 2005; Baron and Rayson, 2008) or the fully automatic Norma tool (Bollmann, 2012).

2.2 Rule-based Methods

Rule-based approaches try to encode regularities in spelling variants—e.g., historical $\langle v \rangle$ often representing modern $\langle u \rangle$—in the form of replacement rules, typically including context information to discriminate between different usages of a character. Some of the earliest approaches to normalization are rule-based, with rules being created manually for one particular language, such as Old Icelandic (Fix, 1980) or Old German (Koller, 1983).

VARD 2 uses "letter replacement rules" to construct normalization candidates, but is not necessarily concerned with precision due to its interactive nature (Baron and Rayson, 2008). Bollmann et al. (2011) describe a supervised learning algorithm to automatically derive context-aware replacement rules from training data, including "identity rules" that leave a character unchanged, then apply one rule to each character of a historical word form to produce a normalization. Porta et al. (2013) model phonological sound change rules for Old Spanish using finite-state transducers; Etxeberria et al. (2016) describe a similarly motivated model that can be trained in a supervised manner.

Rule-based methods are also commonly found when the goal is not to produce a single best normalization, but to cluster a group of spelling variants (Giusti et al., 2007) or to retrieve occurrences of variant spellings given a modern form in an information retrieval (IR) scenario (Ernst-Gerlach and Fuhr, 2006; Koolen et al., 2006).

2.3 Distance-based Methods

Approaches using edit distance measures (such as Levenshtein distance; Levenshtein, 1966) are most commonly found in an IR context, since measures that compare two word forms are a natural fit for matching a search term with relevant word forms in a historical document (e.g., Robertson and Willett, 1993). Weighted variants of distance measures can be used to assign lower costs to more likely edit operations (Kempken et al., 2006; Hauser and Schulz, 2007).

In a normalization context, distance measures can be used to compare historical variants to entries in a contemporary full-form lexicon (Kestemont et al., 2010; Jurish, 2010a). Norma includes a distance-based component whose edit weights can be learned from a training set of normalizations (Bollmann, 2012). Pettersson et al. (2013a) find a similar approach to be more effective than hand-crafted rules on Swedish. Sometimes, the line between distance-based and rule-based methods get blurred; Adesam et al. (2012) use the Levenshtein algorithm to derive "substitution rules" from training data, which are then used to link up historical Swedish forms with lexicon entries; van Halteren and Rem (2013) describe a comparable approach for Dutch.

Furthermore, distance measures also lend themselves to unsupervised approaches for clustering historical variants of the same modern form, where identifying the precise modern form is not necessarily required (Amoia and Martínez, 2013; Barteld et al., 2015).

2.4 Statistical Models

In a probabilistic view of the normalization task, the goal is to optimize the probability $p(t|s)$ that a contemporary word form t is the normalization of a historical word form s. This can be seen as a *noisy channel model*, which has been used for normalization by, e.g., Oravecz et al. (2010) and Etxeberria et al. (2016).

More commonly, *character-based statistical machine translation (CSMT)* has been applied to the normalization task. Instead of translating a sentence as a sequence of tokens, these approaches "translate" a historical word form as a sequence of characters. This has been found to be very effective for a variety of historical languages, such as Spanish (Sánchez-Martínez et al., 2013), Icelandic and Swedish (Pettersson et al., 2013b), Slovene (Scherrer and Erjavec, 2013, 2016; Ljubešić et al., 2016), as well as Hungarian, German, and English (Pettersson, 2016), where it is usually found to outperform previous approaches.

Pettersson et al. (2014) find that a CSMT system often performs best in a comparison with a filtering method and a distance-based approach on five different languages. Schneider et al. (2017) compare VARD 2 to CSMT on English and find that VARD 2 performs slightly better. Domingo and Casacuberta (2018) evaluate both word-based and character-based models and find that SMT outperforms a neural network model.

2.5 Neural Models

Neural network architectures have become popular for a variety of NLP tasks, and historical normalization is no exception. *Character-based neural machine translation (CNMT)* is the logical neural equivalent to the CSMT approach, and has first been used for normalization of historical German (Bollmann et al., 2017; Korchagina, 2017) using encoder–decoder models with long short-term memory (LSTM) units. Robertson and Goldwater (2018) present a more detailed evaluation of this architecture on five different languages. Hämäläinen et al. (2018) evaluate SMT, NMT, an edit-distance approach, and a rule-based finite-state transducer, and advocate for a combination of these approaches to make use of their individual strengths; however, they restrict their evaluation to English.

Other neural architectures have rarely been used for normalization so far. Al Azawi et al. (2013) and Bollmann and Søgaard (2016) frame the normalization task as a sequence labelling problem, labelling each character in the historical word form with its normalized equivalent. Kestemont et al. (2016) use convolutional networks for lemmatization of historical Dutch. Overall, though, the encoder–decoder model with recurrent layers is the dominant approach.

2.6 Beyond Token-Level Normalization

The presented literature almost exclusively focuses on models where the input is a single token. In theory, it would be desirable to include context from the surrounding tokens, as some historical spellings can have more than one modern equivalent depending on the context in which they are used (e.g., historical *ther* could represent *their* or *there*). Remarkably few studies have attempted this so far: Jurish (2010b) uses hidden Markov models to select between normalization candidates; Mitankin et al. (2014) use a language model in a similar vein; Ljubešić et al. (2016) experiment with "segment-level" input, i.e., a string of several historical tokens as input to a normalizer. Since this area is currently very underexplored, it warrants a deeper investigation that goes beyond the scope of this paper.

3 Experimental Setup

Systems The selection of normalization systems follows two goals: (i) to include at least one system for each major category as identified in Sec. 2; and (ii) to use only freely available tools in order to facilitate reproduction and application of the described methods. To that effect, this study compares the following approaches:

- **Norma**[3] (Bollmann, 2012), which combines substitution lists, a rule-based normalizer, and a distance-based algorithm, with the option of running them separately or combined. Importantly, it implements supervised learning algorithms for all of these components and is not restricted to a particular language.
- **cSMTiser**[4] (Ljubešić et al., 2016; Scherrer and Ljubešić, 2016), which implements a normalization pipeline using character-based statistical machine translation (CSMT) using the Moses toolkit (Koehn et al., 2007).
- **Neural machine translation (NMT)**, in the form of two publicly available implementations: (i) the model by Bollmann (2018), also used in Bollmann et al. (2018);[5] and (ii) the model by Tang et al. (2018).[6]

Two systems were chosen for the NMT approach as they use very different hyperparameters, despite both using comparable neural encoder–decoder models: Bollmann (2018) uses a single LSTM layer with dimensionality 300 in the encoder and decoder, while Tang et al. (2018) use six vanilla RNN cells with dimensionality 1024.

---

[3] `https://github.com/comphist/norma`

[4] `https://github.com/clarinsi/csmtiser`

[5] I reimplemented the model here using the XNMT toolkit (Neubig et al., 2018).

[6] `https://github.com/tanggongbo/normalization-NMT`; their model uses the deep transition architecture of Sennrich et al. (2017, Sec. 2.3.1) as implemented by Marian (Junczys-Dowmunt et al., 2018).

Dataset/Language		Time Period	Genre	Tokens		
				TRAIN	DEV	TEST
DE_A	German (Anselm)	14th–16th c.	Religious	233,947	45,996	45,999
DE_R	German (RIDGES)	1482–1652	Science	41,857	9,712	9,587
EN	English	1386–1698	Letters	147,826	16,334	17,644
ES	Spanish	15th–19th c.	Letters	97,320	11,650	12,479
HU	Hungarian	1440–1541	Religious	134,028	16,707	16,779
IS	Icelandic	15th c.	Religious	49,633	6,109	6,037
PT	Portuguese	15th–19th c.	Letters	222,525	26,749	27,078
SL_B	Slovene (Bohorič)	1750–1840s	Diverse	50,023	5,841	5,969
SL_G	Slovene (Gaj)	1840s–1899	Diverse	161,211	20,878	21,493
SV	Swedish	1527–1812	Diverse	24,458	2,245	29,184

Table 1: Historical datasets used in the experiments

Datasets Table 1 gives an overview of the historical datasets. They are taken from Bollmann (2018) and represent the largest and most varied collection of datasets used for historical text normalization so far, covering eight languages from different language families—English, German, Hungarian, Icelandic, Spanish, Portuguese, Slovene, and Swedish—as well as different text genres and time periods. Furthermore, most of these have also been used in previous work, such as the English, Hungarian, Icelandic, and Swedish datasets (e.g., Pettersson et al., 2014; Pettersson, 2016; Robertson and Goldwater, 2018; Tang et al., 2018) and the Slovene datasets (e.g., Ljubešić et al., 2016; Scherrer and Erjavec, 2016; Etxeberria et al., 2016; Domingo and Casacuberta, 2018).

Additionally, contemporary datasets are required for the rule-based and distance-based components of Norma, as they expect a list of valid target word forms to function properly. For this, we want to choose resources that are readily available for many languages and are reliable, i.e., consist of carefully edited text. Here, I choose a combination of three sources:[7] (i) the normalizations in the training sets, (ii) the Europarl corpus (Koehn, 2005), and (iii) the parallel Bible corpus by Christodouloupoulos and Steedman (2015). The only exception is Icelandic, which is not covered by Europarl; here, we can follow Pettersson (2016) instead by using data from two specialized resources, the BÍN database (Bjarnadóttir, 2012) and the MÍM corpus (Helgadóttir et al., 2012). This way, we obtain full-form lexica of 12k–64k word types from the Bible corpus, 55k–268k types from Europarl, and 2.8M types from the Icelandic resources.

Preprocessing The most important preprocessing decisions[8] are (i) to lowercase all characters and (ii) to remove all punctuation-only tokens. Both capitalization and punctuation often cannot be handled correctly without considering token context, which all current normalization models do not do. Furthermore, their usage can be very erratic in historical texts, potentially distorting the evaluation; e.g., when a text uses punctuation marks according to modern conventions, their normalization is usually trivial, resulting in artificial gains in normalization accuracy that other texts do not get. At the same time, most previous work has not followed these same preprocessing guidelines, making a direct comparison more difficult. This work tries to make up for this by evaluating many different systems, effectively reproducing some of these previous results instead.

4 Evaluation

All models are trained and evaluated separately for each dataset by calculating word accuracy over all tokens. In particular, there is no step to discriminate between tokens that require normalization and those that do not; all word forms in the datasets are treated equally.

For Norma, all components are evaluated both separately and combined, as the former gives us insight into the performance of each individual

---

[7] Detailed descriptions of the data extraction procedure can be found in the Supplementary Material.

[8] The full preprocessing steps can be found in the Supplementary Material.

Method	**Dataset**									
	DE_A	DE_R	EN	ES	HU	IS	PT	SL_B	SL_G	SV
Identity	*30.63*	*44.36*	*75.29*	*73.40*	*17.53*	*47.62*	*65.19*	*40.74*	*85.38*	*58.59*
Maximum	*94.64*	*96.46*	*98.57*	*97.40*	*98.70*	*93.46*	*97.65*	*98.71*	*98.96*	*98.97*
Norma, Lookup	83.86	82.15	92.45	92.51	74.58	82.84	91.67	81.76	93.90	83.80
Norma, Rule-based	76.48	82.52	90.85	88.59	78.73	83.72	86.33	86.09	91.63	85.23
Norma, Distance-based	58.92	73.30	83.92	84.41	62.38	69.95	77.28	71.02	88.20	76.03
Norma (Combined)	88.02	86.55	94.60	94.41	86.83	*86.85	94.19	89.45	91.44	87.12
cSMTiser	88.82	*88.06	*95.21	*95.01	*91.63	*87.10	*95.09	*93.18	*95.99	**91.13**
cSMTiser$_{\text{+LM}}$	86.69	*88.19	**95.24**	**95.02**	**91.70**	*86.83	**95.18**	**93.30**	**96.01**	*91.11
NMT (Bollmann, 2018)	89.16	*88.07	94.80	*94.83	91.17	86.45	94.64	91.61	95.19	90.27
NMT (Tang et al., 2018)	**89.64**	**88.22**	94.95	*94.84	*91.65	**87.31**	94.51	92.60	*95.85	90.39
†SMT (Pettersson et al., 2014)	–	–	94.3–	–	80.1–	71.8–	–	–	–	92.9–
†NMT (Tang et al., 2018)	–	–	94.69	–	91.69	87.59	–	–	–	91.56

Table 2: Word accuracy of different normalization methods on the test sets of the historical datasets, in percent; best result for each dataset in bold; results marked with an asterisk (*) are not significantly different from the best result using McNemar's test at $p < 0.05$. † indicates scores that were not (re)produced here, but reported in previous work; they might not be strictly comparable due to differences in data preprocessing (cf. Sec. 3). Additionally, *Identity* shows the accuracy when leaving all word forms unchanged, while *Maximum* gives the theoretical maximum accuracy with purely token-level methods.

component, while the latter is reported to produce the best results (Bollmann, 2012). For cSMTiser, the authors suggest using additional monolingual data to improve the language model; the contemporary datasets are used for this purpose and the model is trained both without and with this additional data; the latter is denoted cSMTiser$_{\text{+LM}}$. For NMT, the model by Bollmann (2018) is evaluated using an ensemble of five models; the model by Tang et al. (2018) is trained on character-level input using the default settings provided by their implementation.[9]

To illustrate how challenging the normalization task is on different datasets, we can additionally look at the *identity baseline*—i.e., the percentage of tokens that do *not* need to be normalized—as well as the *maximum accuracy* obtainable if each word type was mapped to its most frequently occurring normalization. The latter gives an indication of the extent of ambiguity in the datasets and the disadvantage of not considering token context (cf. Sec. 2.6).

Results Table 2 shows the results of this evaluation. The extent of spelling variation varies greatly between datasets, with less than 15% of tokens requiring normalization (SL_G) to more than 80% (HU). The maximum accuracy is above 97% for most datasets, suggesting that we can obtain high normalization accuracy in principle even without considering token context.

For the normalization systems, we observe significantly better word accuracy with SMT than NMT on four of the datasets, and non-significant differences on five others. There is only one dataset (DE_A) where the NMT system by Tang et al. (2018) gets significantly better word accuracy than other systems. This somewhat contradicts the results from Tang et al. (2018), who find NMT to usually outperform the SMT baseline by Pettersson et al. (2014). However, note that the results for the cSMTiser system are often significantly better than reported in previous work: e.g., on Hungarian, cSMTiser obtains 91.7% accuracy, but only 80.1% with the SMT system from Pettersson et al. (2014).

Overall, the deep NMT model by Tang et al. (2018) consistently outperforms the shallow one by Bollmann (2018). cSMTiser seems to benefit from the added contemporary data for language modelling, though the effect is not significant on any individual dataset. Finally, while Norma does produce competitive results on sev-

---

[9] This is the "Att-RNN" setting reported in their paper; due to the high computational demands of the model, it was not feasible to run experiments with multiple configurations.

Method	Dataset									
	DE$_\text{A}$	DE$_\text{R}$	EN	ES	HU	IS	PT	SL$_\text{B}$	SL$_\text{G}$	SV
Norma, Lookup	0.41	0.31	**0.38**	**0.35**	0.43	**0.38**	**0.39**	**0.44**	0.44	0.29
Norma, Rule-based	0.40	0.33	0.43	0.39	0.38	0.40	0.45	0.47	0.46	0.32
Norma, Distance-based	0.42	0.34	0.46	0.44	0.41	0.44	0.50	0.52	**0.39**	0.38
Norma (Combined)	0.41	0.33	0.45	0.42	0.34	0.42	0.51	0.51	0.42	0.31
cSMTiser	**0.37**	**0.26**	0.39	0.41	**0.26**	0.40	0.50	0.53	0.56	0.24
cSMTiser$_\text{+LM}$	0.39	0.27	0.39	0.42	0.27	0.41	0.50	0.53	0.56	0.24
NMT (Bollmann, 2018)	0.38	**0.26**	0.39	0.43	0.27	0.40	0.48	0.47	0.51	**0.23**
NMT (Tang et al., 2018)	0.38	0.27	**0.38**	0.42	**0.26**	0.41	0.46	0.50	0.56	0.24

(a) CER$_\text{I}$: character error rate on the subset of incorrect normalizations (lower is better)

Method	DE$_\text{A}$	DE$_\text{R}$	EN	ES	HU	IS	PT	SL$_\text{B}$	SL$_\text{G}$	SV
Norma, Lookup	8.47	16.72	8.33	27.81	2.86	–	4.57	–	–	7.42
Norma, Rule-based	12.82	26.73	8.48	26.33	12.44	–	5.65	–	–	15.38
Norma, Distance-based	7.71	19.10	7.26	28.62	10.93	–	5.58	–	–	12.18
Norma (Combined)	16.97	28.94	**9.86**	43.55	20.00	–	**11.13**	–	–	**20.75**
cSMTiser	**17.92**	**34.67**	8.27	**43.82**	**20.44**	–	6.09	–	–	17.73
cSMTiser$_\text{+LM}$	12.23	33.83	8.46	42.93	20.40	–	6.60	–	–	17.58
NMT (Bollmann, 2018)	17.24	33.65	7.96	39.22	18.30	–	5.92	–	–	16.65
NMT (Tang et al., 2018)	16.34	34.19	9.43	40.99	19.84	–	6.46	–	–	18.93

(b) Stemming accuracy: percentage of incorrect normalizations with correct word stems (higher is better)

Table 3: Evaluations on the subset of incorrect normalizations only; best results for each dataset in bold. Note that this subset is different for each system, so for comparisons between systems, these numbers should be considered in conjunction with word accuracy scores from Table 2.

eral datasets (particularly in the "combined" setting), it is generally significantly behind the SMT and NMT methods.

5 Analysis

5.1 Measuring Normalization Quality

While word accuracy is easily interpretable, it is also a very crude measure, as it classifies predictions as correct/incorrect without considering the type of error(s) made by the model. *Character error rate (CER)* has sometimes been suggested as a complement to address this issue, but I believe this is not very insightful: For any normalization system that achieves a reasonably high word accuracy, CER will highly correlate with accuracy simply because CER equals zero for any word that is accurately normalized.[10] At the same time, there is a need for a more fine-grained way to assess the normalization quality. Consider the following example from the Hungarian dataset with its predicted normalization from the NMT system by Bollmann (2018):

(1) ORIG yduewzewlendewk
GOLD üdvözülendőek
PRED üdvözülendők

Here, the prediction matches the correct target form almost perfectly, but would be counted as incorrect since it misses an insertion of the letter ⟨*e*⟩ towards the end. In this vein, it will be treated the same by the word accuracy measure as a prediction that, e.g., had left the original form unchanged.

CER$_\text{I}$ One alternative is to consider character error rate *on the subset of incorrect normalizations only.* This way, CER becomes a true complement to word accuracy by assessing the *magnitude of error* that a normalization model makes when it is not perfectly accurate. The results of this measure, denoted CER$_\text{I}$, are shown in Table 3a. The lowest CER$_\text{I}$ score is often achieved

---

[10] When comparing word accuracy scores in Table 2 with the same configurations evaluated using CER, they correlate with Pearson's $r \approx -0.96$.

by Norma's lookup module, which leaves historical word forms unchanged if they are not in its lookup wordlist learned during training. This suggests that the incorrect predictions made by other systems are often *worse* than just leaving the historical spelling unchanged.

Stemming Another problem of CER is that all types of errors are treated the same: a one-letter difference in inflection, such as *king – kings* or *came – come*, would be treated identically to an error that changes the meaning of the word (*bids – beds*) or results in a non-word (*creature – cryature*). I propose an approach that, to the best of my knowledge, has not been used in normalization evaluation before: measure accuracy on *word stems,* i.e., process both the reference normalization and the prediction with an automatic stemming algorithm and check if both stems match. For this evaluation, I choose the Snowball stemmer (Porter, 2001) as it contains stemming algorithms for many languages (including the ones represented here except for Icelandic and Slovene) and is publicly available.[11]

Table 3b shows the accuracy on word stems, again only evaluated on the subset of incorrect normalizations, as this better highlights the differences between settings. This evaluation reveals some notable differences between *datasets*: For example, while the English and Spanish datasets have very comparable accuracy scores overall (cf. Tab. 2), they show very different characteristics in the stemming evaluation; for English, only up to 9.86% of incorrect predictions show the correct word stem, while for Spanish the number is up to 43.82%. Examining predictions on the dev set, many of the incorrectly predicted cases in Spanish result from mistakes in placement of diacritics, such as *ésta – está* or *envíe – envié*; the stemming algorithm removes diacritics and can therefore match these instances. Overall, this gives an indication that the errors made on the Spanish dataset are less severe than those on English, despite comparable word accuracy scores and a usually higher CER_I for Spanish.

This case study shows that stemming can be a useful tool for error analysis in normalization models and reveal characteristics that neither word accuracy nor CER alone can show.

5.2 Effect of Training Data Quantity

Supervised methods for historical text normalization have been evaluated with highly varying amounts of training data: e.g., Domingo and Casacuberta (2018) train a normalizer for 17th century Spanish on 436k tokens; Etxeberria et al. (2016) use only 8k tokens to train a normalizer for Basque. Even in the evaluation in Sec. 4, training set sizes varied between 24k and 234k tokens, depending on the dataset. Furthermore, many research projects seeking to use automatic normalization techniques cannot afford to produce training data in high quantity. All of this raises the question how different normalization systems perform with varying amounts of training data, and whether reasonable normalization results can be achieved in a low-resource scenario.

Methodology All models are retrained on varying subsets of the training data, with sizes ranging from 100 tokens to 50,000 tokens. However, the lower the training set size is, the higher the potential variance when training on it, since random factors such as the covered spelling variants or vocabulary are more likely to impact the results. Therefore, I choose the following approach: For each dataset and training size, up to ten different training splits are extracted,[12] and a separate model is trained on each one. Each model is then evaluated on the respective development dataset, and only the average accuracy across all splits is considered.

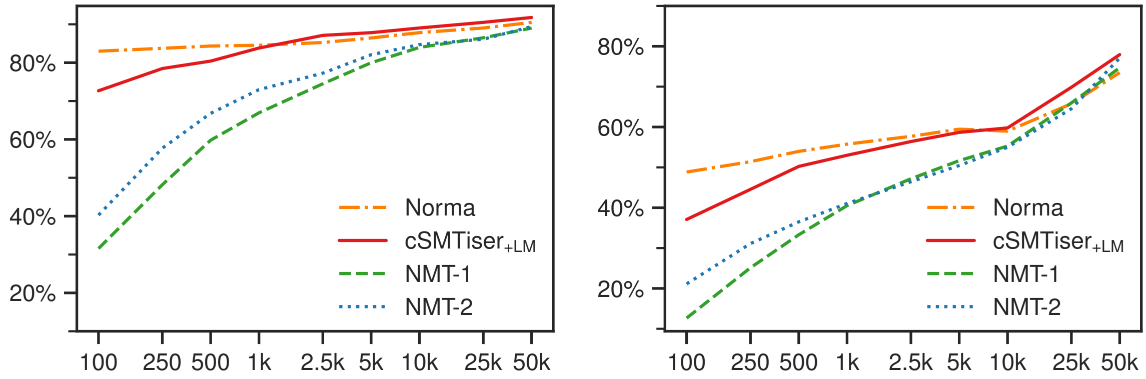
Results Figure 2 shows two learning curves that are representative for most of the datasets.[13] They reveal that Norma (in the "combined" setting) performs best in extremely low-resource scenarios, but is overtaken by the SMT approach as more training data becomes available; usually already around 500–1000 tokens. The NMT models have a steeper learning curve, needing more training data to become competitive. Extrapolating this trend, it is conceivable that the NMT models would simply need more training data than our current datasets provide in order to consistently outperform the SMT approach. On the other hand, there appears to be no correlation between the size of the training set (cf. Tab. 1) and the relative per-

---

[11] `http://snowballstem.org/`

[12] The ten training splits consist of chunks of n tokens that are spaced equidistantly across the full training set; for larger n, the number of chunks is reduced so that no splits overlap to more than 50%.

[13] Plots for all datasets can be found in the Appendix.



(a) English

(b) Hungarian

Figure 2: Word accuracy on the development sets for different amounts of training data (note that the x-axis is log-scaled); NMT-1 is the model by Bollmann (2018), NMT-2 is the model by Tang et al. (2018).

formance of NMT vs. SMT (cf. Tab. 2) in the experiments. Since I am not aware of larger datasets for the historical normalization task, this remains an open question for now.

A remarkable result is that very small amounts of training data can already be helpful for the normalization task. The English dataset has comparatively little spelling variation to begin with: leaving all words unnormalized already results in an accuracy of 75.5%. Still, with as little as 100 tokens for training, applying the Norma tool raises the accuracy above 83%. For Hungarian, the same amount of training data raises the accuracy from 17.8% (unnormalized) to around 50%. It would be interesting to further compare these results with fully unsupervised methods.

5.3 Out-of-Vocabulary Words

Robertson and Goldwater (2018) highlight the importance of evaluating separately on seen vs. unseen tokens, i.e., tokens that have also been in the training set (in-vocabulary) and those that have not (out-of-vocabulary), as well as comparing to a naive memorization baseline. These numbers are presented in Table 4. For unseen tokens (Tab. 4b), the accuracy scores follow generally the same trend as in the full evaluation of Tab. 2; i.e., SMT performs best in most cases. For seen tokens (Tab. 4a), however, Norma's lookup component—which implements naive memorization—obtains the highest score on nine datasets.

These observations suggest a new normalization strategy: apply the naive lookup on the subset of in-vocabulary tokens and the SMT/NMT models on the subset of out-of-vocabulary tokens only. Table 5 shows the results of this strategy.[14] On nine datasets, it performs better than always using the learned models (as in Tab. 2), and this difference is statistically significant on five of them. These results support the claim from Robertson and Goldwater (2018) that "learned models should typically only be applied to *unseen* tokens."

6 Conclusion

This paper presented a large study of historical text normalization. Starting with a systematic survey of the existing literature, four different systems (based on supervised learning) were evaluated and compared on datasets from eight different languages. On the basis of these results, we can extract some practical recommendations for projects seeking to employ normalization techniques:

1. to use the Norma tool when only little training data (<500 tokens) is available;

2. to use cSMTiser otherwise, ideally with additional data for language modelling; and

3. to make use of the naive memorization/lookup technique for in-vocabulary tokens when possible.

Furthermore, the qualitative analysis (in Sec. 5.1) should encourage authors evaluating normalization systems to use task-motivated approaches, such as evaluation on word stems, to provide

---

[14] The non-lookup components of Norma are not included in this evaluation since "Norma (Combined)" effectively implements such a strategy already.

Method	**Dataset**									
	DE$_{\text{A}}$	DE$_{\text{R}}$	EN	ES	HU	IS	PT	SL$_{\text{B}}$	SL$_{\text{G}}$	SV
Norma, Lookup/Combined	**92.36**	**93.66**	**97.46**	**96.59**	**96.81**	**89.51**	**97.04**	**97.15**	**98.17**	97.61
Norma, Rule-based	80.34	89.26	93.17	89.98	88.42	86.21	88.77	93.61	96.10	92.10
Norma, Distance-based	60.79	78.85	86.77	85.89	67.04	70.99	78.93	77.49	94.54	84.17
cSMTiser	92.18	93.25	97.10	96.33	96.33	89.27	96.80	96.73	98.09	**97.66**
cSMTiser$_{\text{+LM}}$	90.52	93.45	97.15	96.42	96.33	88.99	96.82	96.90	98.07	**97.66**
NMT (Bollmann, 2018)	91.91	93.29	97.19	96.37	96.18	88.67	96.72	95.92	97.56	97.01
NMT (Tang et al., 2018)	92.25	93.41	97.19	96.27	96.43	89.34	96.60	96.84	97.89	96.90

(a) In-vocabulary/seen tokens

Norma, Lookup	3.91	19.92	30.42	46.72	3.28	29.40	28.25	18.07	68.07	39.85
Norma, Rule-based	40.27	46.06	62.06	72.97	47.66	63.73	57.50	54.99	64.59	63.37
Norma, Distance-based	41.33	43.25	48.63	67.78	47.43	61.64	57.79	44.23	49.82	50.15
Norma (Combined)	47.25	48.13	59.18	69.93	54.83	65.52	60.62	57.57	50.71	53.73
cSMTiser	57.25	59.96	**71.78**	**80.22**	76.59	69.70	74.69	**78.49**	83.30	**70.35**
cSMTiser$_{\text{+LM}}$	50.69	59.76	71.70	79.24	**76.86**	69.55	**75.91**	78.40	**83.56**	70.26
NMT (Bollmann, 2018)	63.24	59.83	65.17	77.57	75.13	68.66	70.00	73.75	80.87	68.78
NMT (Tang et al., 2018)	**65.09**	**60.16**	67.15	78.75	76.33	**71.04**	69.90	75.04	83.46	69.66

(b) Out-of-vocabulary/unseen tokens

Table 4: Word accuracy for seen/unseen tokens separately (cf. Sec. 5.3); best results for each dataset in bold.

Method	**Dataset**									
	DE$_{\text{A}}$	DE$_{\text{R}}$	EN	ES	HU	IS	PT	SL$_{\text{B}}$	SL$_{\text{G}}$	SV
Best without lookup	**89.64*	**88.22*	*95.24*	*95.02*	*91.70*	**87.31*	*95.18*	*93.30*	**96.01*	***91.13***
cSMTiser	88.98	*88.41	**95.54**	**95.25**	*92.00	*87.31	*95.31	**93.52**	*96.06	*91.09
cSMTiser$_{\text{+LM}}$	88.35	*88.37	*95.53	*95.17	**92.07**	*87.30	**95.39**	*93.50	**96.10**	*91.07
NMT (Bollmann, 2018)	89.56	*88.38	95.05	95.03	91.66	*87.20	94.93	92.60	95.71	90.72
NMT (Tang et al., 2018)	**89.74**	**88.45**	95.19	*95.13	*91.94	**87.46**	94.92	92.85	*96.08	*90.93

Table 5: Word accuracy for the "lookup on *seen* tokens, learned models on *unseen* tokens" strategy, following Robertson and Goldwater (2018) (cf. Sec. 5.3), compared to the best result without this strategy (according to Table 2). Best result for each dataset in bold; results marked with an asterisk (*) are not significantly different from the best result using McNemar's test at $p < 0.05$.

deeper insight into the properties of their models and datasets.

Detailed information on how to train and apply all of the evaluated techniques is made available online at `https://github.com/coastalcph/histnorm`.

Acknowledgments

I would like to thank my PhD supervisor, Stefanie Dipper, for her continuous support over many years that culminated in the doctoral thesis on which this paper is based; the acknowledgments in that thesis largely extend to this paper as well. Many thanks to Anders Søgaard for many helpful discussions and for supporting the follow-up experiments conducted for this paper. Further thanks go to the anonymous reviewers whose helpful suggestions have largely been incorporated here.

I gratefully acknowledge the donation of a Titan Xp GPU by the NVIDIA Corporation that was used for a substantial part of this research.

References

Yvonne Adesam, Malin Ahlberg, and Gerlof Bouma. 2012. *bokstaffua, bokstaffwa, bokstafwa, bokstaua, bokstawa…* Towards lexical link-up for a corpus of Old Swedish. In *Proceedings of the 11th Conference on Natural Language Processing (KONVENS 2012), LThist 2012 workshop*, pages 365–369, Vienna, Austria.

Mayce Al Azawi, Muhammad Zeshan Afzal, and Thomas M. Breuel. 2013. Normalizing historical orthography for OCR historical documents using LSTM. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing (HIP '13)*, pages 80–85, Washington, DC.

Marilisa Amoia and José Manuel Martínez. 2013. Using comparable collections of historical texts for building a diachronic dictionary for spelling normalization. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 84–89. Association for Computational Linguistics.

Alistair Baron and Paul Rayson. 2008. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*.

Fabian Barteld, Ingrid Schröder, and Heike Zinsmeister. 2015. Unsupervised regularization of historical texts for POS tagging. In *Proceedings of the Workshop on Corpus-Based Research in the Humanities (CRH)*, pages 3–12, Warsaw, Poland.

Kristín Bjarnadóttir. 2012. The database of modern Icelandic inflection (Beygingarlýsing íslensks nútímamáls). In *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages*, pages 13–18.

Marcel Bollmann. 2012. (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, pages 3–12, Lisbon, Portugal.

Marcel Bollmann. 2018. Normalization of historical texts with neural network models. *Bochumer Linguistische Arbeitsberichte*, 22.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 332–344. Association for Computational Linguistics.

Marcel Bollmann, Florian Petran, and Stefanie Dipper. 2011. Rule-based normalization of historical texts. In *Proceedings of the International Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139. The COLING 2016 Organizing Committee.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24. Association for Computational Linguistics.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the Bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

Miguel Domingo and Francisco Casacuberta. 2018. Spelling normalization of historical documents by using a machine translation approach. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 129–137.

Andrea Ernst-Gerlach and Norbert Fuhr. 2006. Generating search term variants for text collections with historic spellings. In *Proceedings of the 28th European Conference on Information Retrieval Research (ECIR 2006)*, Lecture Notes in Computer Science, pages 49–60, Berlin. Springer.

Izaskun Etxeberria, Iñaki Alegria, Larraitz Uria, and Mans Hulden. 2016. Evaluating the noisy channel model for the normalization of historical texts: Basque, Spanish and Slovene. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1064–1069, Paris, France. European Language Resources Association (ELRA).

Hans Fix. 1980. Automatische Normalisierung – Vorarbeit zur Lemmatisierung eines diplomatischen altisländischen Textes. In Paul Sappler and Erich Straßner, editors, *Maschinelle Verarbeitung altdeutscher Texte. Beiträge zum dritten Symposion, Tübingen 17.–19. Februar 1977*, pages 92–100. Niemeyer, Tübingen.

Rafael Giusti, Arnaldo Candido Jr, Marcelo Muniz, Lívia Cucatto, and Sandra Aluísio. 2007. Automatic detection of spelling variation in historical corpus: An application to build a Brazilian Portuguese spelling variants dictionary. In *Proceedings of the Corpus Linguistics Conference (CL2007)*, Birmingham, UK.

Hans van Halteren and Margit Rem. 2013. Dealing with orthographic variation in a tagger-lemmatizer for fourteenth century Dutch charters. *Language Resources and Evaluation*, 47(4):1233–1259.

Mika Hämäläinen, Tanja Säily, Jack Rueter, Jörg Tiedemann, and Eetu Mäkelä. 2018. Normalizing Early English letters to present-day English spelling. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 87–96. Association for Computational Linguistics.

Andreas W. Hauser and Klaus U. Schulz. 2007. Unsupervised learning of edit distance weights for retrieving historical spelling variations. In *Proceedings of the First Workshop on Finite-State Techniques and Approximate Search (FSTAS 2007)*, pages 1–6, Borovets, Bulgaria.

Sigrún Helgadóttir, Ásta Svavarsdóttir, Eiríkur Rögnvaldsson, Kristín Bjarnadóttir, and Hrafn Loftsson. 2012. The tagged Icelandic corpus (MíM). In *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages*, pages 67–72.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Bryan Jurish. 2010a. Comparing canonicalizations of historical German text. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 72–77, Uppsala, Sweden. Association for Computational Linguistics.

Bryan Jurish. 2010b. More than words: using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39.

Sebastian Kempken, Wolfram Luther, and Thomas Pilz. 2006. Comparison of distance measures for historical spelling variants. In Max Bramer, editor, *Artificial Intelligence in Theory and Practice*, pages 295–304. Springer, Boston, MA.

Mike Kestemont, Walter Daelemans, and Guy De Pauw. 2010. Weigh your words—memory-based lemmatization for Middle Dutch. *Literary and Linguistic Computing*, 25(3):287–301.

Mike Kestemont, Guy de Pauw, Renske van Nie, and Walter Daelemans. 2016. Lemmatization for variation-rich languages using deep learning. *Digital Scholarship in the Humanities*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Mercello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Gerhard Koller. 1983. Ein maschinelles Verfahren zur Normalisierung altdeutscher Texte. In Dietmar Peschel, editor, *Germanistik in Erlangen*, volume 31 of *Erlanger Forschungen*, pages 611–620. Universitätsbund Erlangen-Nürnberg, Erlangen.

Marijn Koolen, Frans Adriaans, Jaap Kamps, and Maarten de Rijke. 2006. A cross-language approach to historic document retrieval. In *Proceedings of the 28th European Conference on Information Retrieval Research (ECIR 2006)*, Lecture Notes in Computer Science, pages 407–419, Berlin. Springer.

Natalia Korchagina. 2017. Normalizing medieval German texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 12–17. Linköping University Electronic Press.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Nikola Ljubešić, Katja Zupan, Darja Fišer, and Tomaž Erjavec. 2016. Normalising Slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, volume 16 of *Bochumer Linguistische Arbeitsberichte*, pages 146–155, Bochum, Germany.

Petar Mitankin, Stefan Gerdjikov, and Stoyan Mihov. 2014. An approach to unsupervised historical text normalisation. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage (DATeCH '14)*, Madrid, Spain.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The extensible neural machine translation toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston.

Csaba Oravecz, Bálint Sass, and Eszter Simon. 2010. Semi-automatic normalization of Old Hungarian codices. In *Proceedings of the ECAI 2010 Workshop on Language Technology for Cultural Heritage*, pages 55–59.

Eva Pettersson. 2016. *Spelling Normalisation and Linguistic Analysis of Historical Text for Information*

Extraction. Doctoral dissertation, Uppsala University, Department of Linguistics and Philology, Uppsala.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2013a. Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 163–179. Linköping University Electronic Press.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2014. A multilingual evaluation of three spelling normalisation methods for historical text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013b. An SMT approach to automatic annotation of historical text. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013*, NEALT Proceedings Series 18/Linköping Electronic Conference Proceedings 87, pages 54–69. Linköping University Electronic Press.

Jordi Porta, José-Luis Sancho, and Javier Gómez. 2013. Edit transducers for spelling variation in Old Spanish. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013*, NEALT Proceedings Series 18/Linköping Electronic Conference Proceedings 87, pages 70–79. Linköping University Electronic Press.

Martin Porter. 2001. Snowball: A language for stemming algorithms.

Paul Rayson, Dawn Archer, and Nicholas Smith. 2005. VARD versus Word: A comparison of the UCREL variant detector and modern spell checkers on English historical corpora. In *Proceedings of the Corpus Linguistics Conference CL2005*, Birmingham, UK. University of Birmingham.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 720–725. Association for Computational Linguistics.

Alexander M. Robertson and Peter Willett. 1993. A comparison of spelling-correction methods for the identification of word forms in historical text databases. *Literary and Linguistic Computing*, 8(3):143–152.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C. Carrasco. 2013. An open diachronic corpus of historical Spanish: annotation criteria and automatic modernisation of spelling. *CoRR*, abs/1306.3692.

Yves Scherrer and Tomaž Erjavec. 2013. Modernizing historical slovene words with character-based SMT. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pages 58–62. Association for Computational Linguistics.

Yves Scherrer and Tomaž Erjavec. 2016. Modernising historical Slovene words. *Natural Language Engineering*, 22(6):881–905.

Yves Scherrer and Nikola Ljubešić. 2016. Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, volume 16 of *Bochumer Linguistische Arbeitsberichte*, pages 248–255, Bochum, Germany.

Gerold Schneider, Eva Pettersson, and Michael Percillier. 2017. Comparing rule-based and SMT-based spelling normalisation for English historical texts. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 40–46. Linköping University Electronic Press.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399. Association for Computational Linguistics.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An evaluation of neural machine translation models on historical spelling normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331. Association for Computational Linguistics.

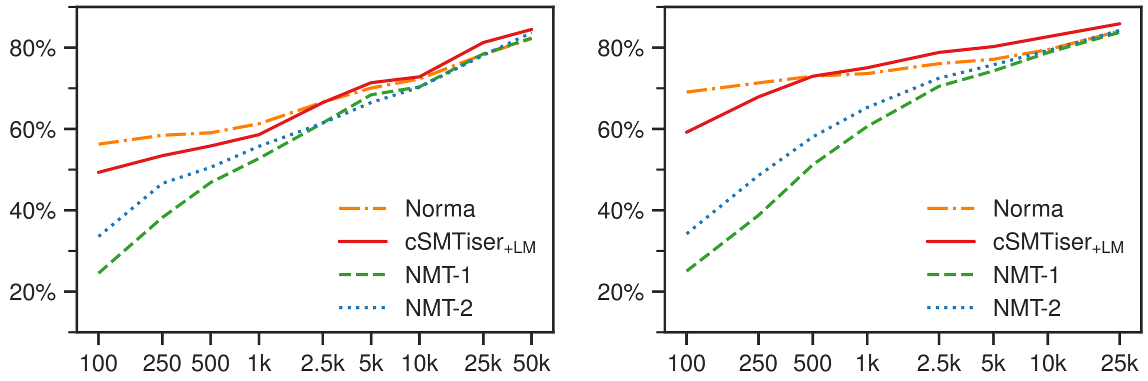
A Appendix

Figures 3 and 4 show plots of the learning curves for all of the datasets.

A.1 Preprocessing

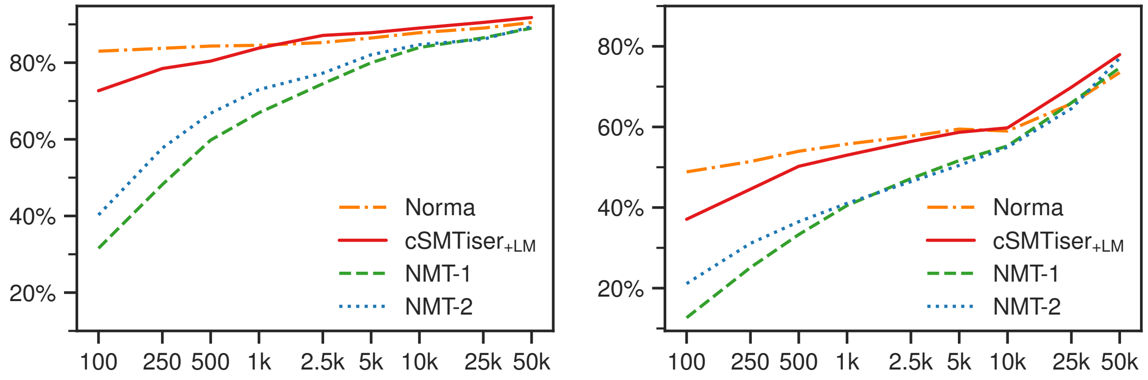
The full preprocessing steps of all datasets (both historical and contemporary) comprise of:

1. lowercasing all tokens;
2. filtering out pairs where either the historical token or the reference normalization is empty;
3. filtering out pairs where either the historical token or the reference normalization consists *only* of punctuation marks, defined as characters that belong to one of the Unicode "Punctuation" categories;



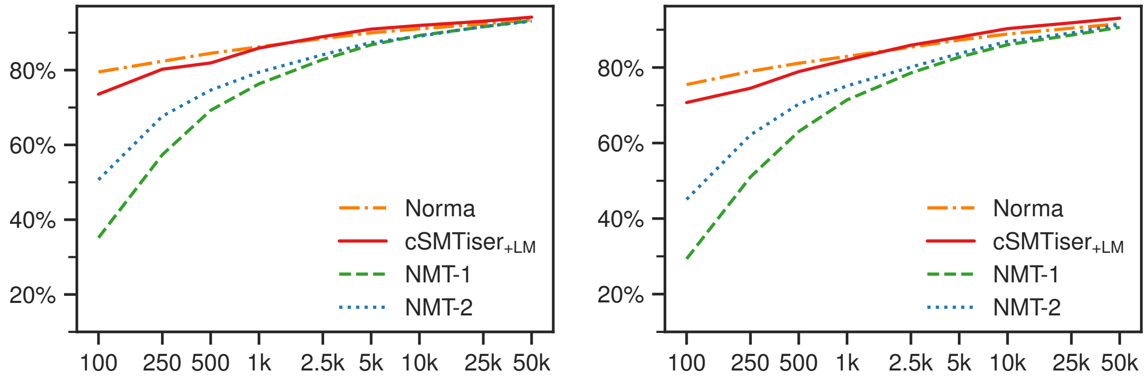
(a) German (Anselm)

(b) German (RIDGES)



(c) English

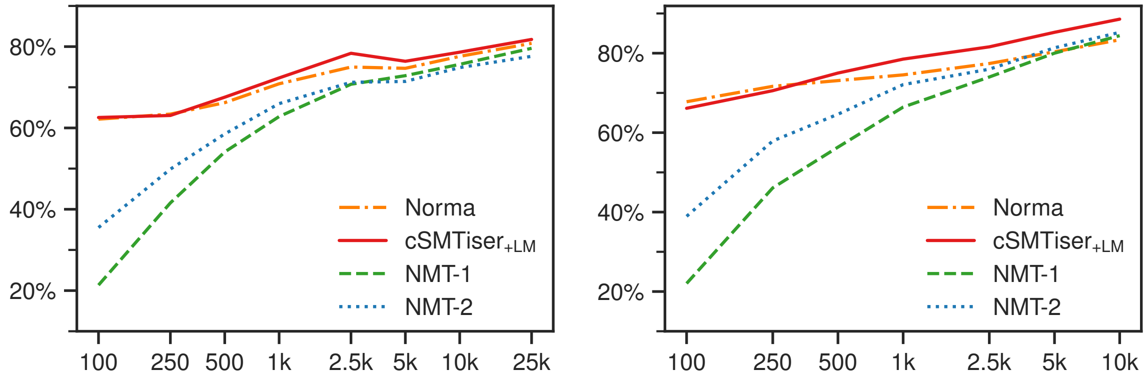
(d) Hungarian



(e) Spanish

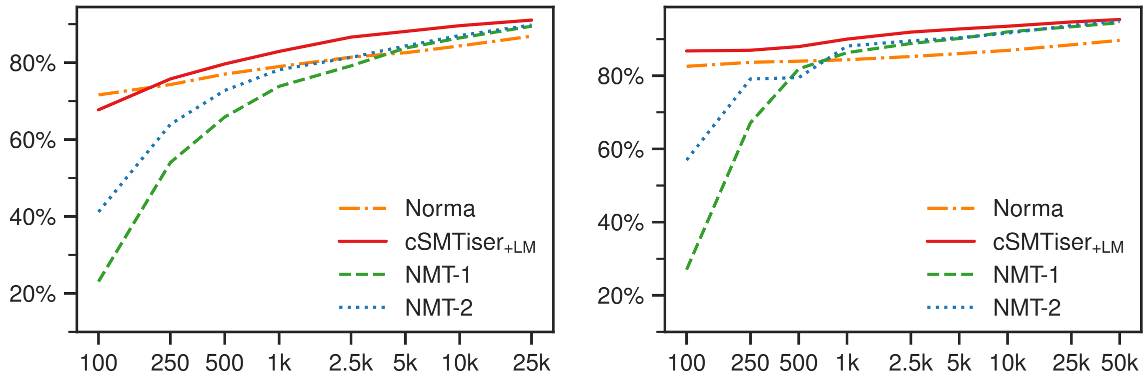
(f) Portuguese

Figure 3: Word accuracy on the development sets for different amounts of training data (note that the x-axis is log-scaled); NMT-1 is the model by Bollmann (2018), NMT-2 is the model by Tang et al. (2018).



(a) Icelandic

(b) Swedish



(c) Slovene (Bohorič)

(d) Slovene (Gaj)

Figure 4: Word accuracy on the development sets (continued); NMT-1 is the model by Bollmann (2018), NMT-2 is the model by Tang et al. (2018).

4. replacing all digits with zeroes *iff* the digits in the historical token and the reference normalization match;

5. replacing actual space characters in either the historical token or the reference normalization with a special symbol that does not otherwise occur in the dataset; and

6. performing Unicode normalization according to the NFC standard.

Additionally, the preprocessing script can also be found in the Supplementary Material or at `https://github.com/coastalcph/histnorm`.

Combining Discourse Markers and Cross-lingual Embeddings for Synonym–Antonym Classification

Michael Roth[1], **Shyam Upadhyay**[2]
[1]Institute for Natural Language Processing, University of Stuttgart
[2]Department of Computer and Information Science, University of Pennsylvania
`rothml@ims.uni-stuttgart.de`, `shyamupa@seas.upenn.edu`

Abstract

It is well-known that distributional semantic approaches have difficulty in distinguishing between synonyms and antonyms (Grefenstette, 1992; Padó and Lapata, 2003). Recent work has shown that supervision available in English for this task (e.g., lexical resources) can be transferred to other languages via cross-lingual word embeddings. However, this kind of transfer misses monolingual distributional information available in a target language, such as contrast relations that are indicative of antonymy (e.g., *hot*...*while*...*cold*). In this work, we improve the transfer by exploiting monolingual information, expressed in the form of co-occurrences with discourse markers that convey contrast. Our approach makes use of less than a dozen markers, which can easily be obtained for many languages. Compared to a baseline using only cross-lingual embeddings, we show absolute improvements of 4–10% F_1-score in Vietnamese and Hindi.

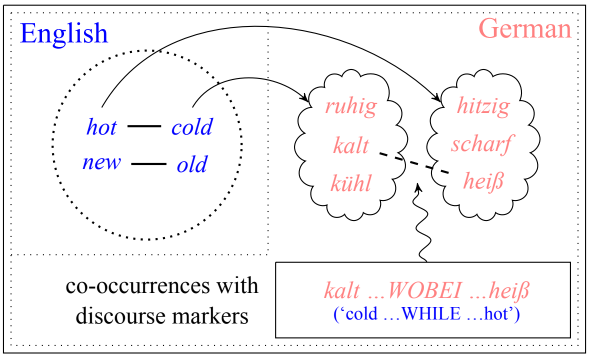


Figure 1: Supervision for distinguishing antonyms from synonyms can be derived using discourse markers. Here, antonyms available in English (denoted by solid edge between *hot* and *cold*) are translated to German via cross-lingual word embeddings. Using co-occurrences with discourse markers indicative of antonymy (shown in box), we can identify pairs of words in the n-best translations (clouds) that are also antonymous (e.g., dashed edge between *kalt* and *heiß*).

1 Introduction

Recent work has shown that monolingual word embeddings in different languages can be aligned in an unsupervised manner (Artetxe et al., 2018; Kementchedjhieva et al., 2018). The resulting *cross-lingual embeddings* can be used to share supervision for lexical classification tasks across languages, when annotated data is not available in one language. For instance, a model for distinguishing lexical relations such as hypernymy and meronymy can be transferred to other languages (Glavaš and Vulić, 2018). However, this kind of transfer, using only cross-lingual embeddings, misses useful monolingual information available in the target language.

In this paper, we consider one lexical classification task, namely the distinction between synonyms and antonyms, which is important for downstream applications such as contradiction detection (Harabagiu et al., 2006; Marneffe et al., 2008; Voorhees, 2008) and machine translation (Marton et al., 2011). To facilitate better transfer, we propose to use monolingual information in the form of word co-occurrences in contrast relations, in addition to cross-lingual embeddings (see Figure 1). In particular, we utilize the fact that discourse markers conveying contrast are more likely to be surrounded by antonyms than synonyms (e.g., *hot*...*while*...*cold*), as shown by Roth and Schulte im Walde (2014).

Our analysis reveals that (1) fine-grained semantic information that is required to distinguish synonyms from antonyms is insufficiently preserved cross-lingually in word embeddings, but (2) such information can be recovered (at least partially) by relying on linguistic intuitions about contrast relations in discourse.

Proceedings of NAACL-HLT 2019, pages 3899–3905
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

2 Related Work

The identification of paradigmatic relations such as synonymy and antonymy has been a task of interest for more than a decade. Early work focused on the identification of instances of a single relation: for example, Charles and Miller (1989) investigated co-occurrences as an indicator for antonymy, Hearst (1992) introduced a pattern-based approach to identify hypernymy, and Edmonds and Hirst (2002) applied distributional statistics to identify synonymy.

Beyond the *identification* of instances of a particular relation, more recent approaches attempt to *distinguish* relations such as synonymy and antonymy, based on lexico-syntactic patterns, distributional information, or combinations of both (Lin et al., 2003; Shwartz and Dagan, 2016). As supervision for weighting different features, most recent work makes use of lexical resources and/or lists of affix patterns that indicate contrast morphologically (Yih et al., 2012; Mohammad et al., 2013; Ono et al., 2015).

Supervision for lexical classification tasks is not available in all languages. To overcome this difficulty, some approaches (Mrkšić et al., 2017; Glavaš and Vulić, 2018) combine resources for English and cross-lingual word embeddings to distinguish lexical relations in other languages. That is, they train and test one model *across* different languages. In contrast, we propose to first use unsupervised translation techniques to *transfer supervision* into a target language, given resources available in English. More specifically, our approach transfers supervision using a combination of unsupervised cross-lingual embeddings (Artetxe et al., 2018) and word co-occurrences with discourse markers that indicate contrast. In doing so, our approach generalizes the discourse-marker based model for relation classification by Roth and Schulte im Walde (2014) to a cross-lingual setting.

3 Our Approach

In this work, we address the task of distinguishing synonyms and antonyms cross-lingually: given a word pair (a, b) in a target language, a model trained in a different source language determines whether this constitutes a synonym or an antonym pair. Our approach consists of two main ingredients: a *translation module* that creates training data in a target language (see Figure 2), and a *classification module* that uses this data in a supervised fashion. Our focus is on the first part, which we describe in more detail below. For classification, we re-use the LexNet model by Shwartz and Dagan (2016), which is a feed-forward network that uses distributional word representations as input.

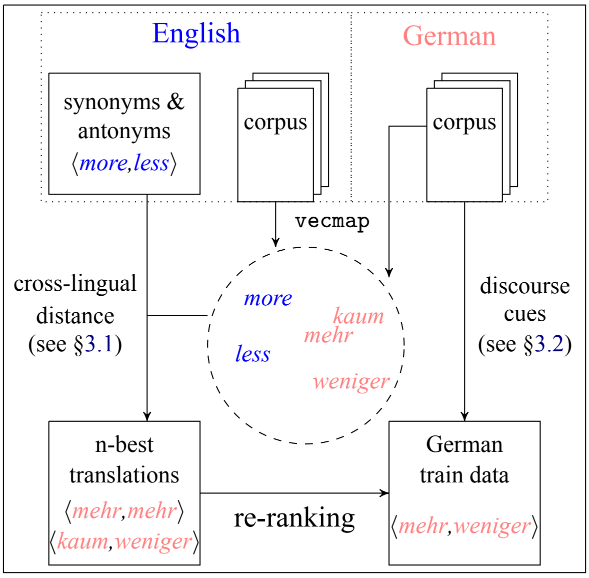


Figure 2: In our approach, we create training data in multiple languages by translating English data via cross-lingual word embeddings and discourse cues.

3.1 Cross-lingual Embeddings for Translation

Our transfer approach addresses a general setting that only assumes availability of monolingual data. Accordingly, we induce a translation dictionary of synonyms and antonyms in an unsupervised way, instead of relying on manually crafted dictionaries. We achieve this by finding n-best translations of English words in a cross-lingual word embedding space. We create this space using `vecmap`, a method that aligns and maps embedding spaces in two languages (Artetxe et al., 2018). Formally, we define the n^{th} best translation $w_t^{(n)}$ of a source word w_s to be the n^{th} nearest neighbor in the joint semantic space (based on cosine similarity).

3.2 Re-ranking n-best Translations

While 1$^{\text{st}}$ best translations are often accurate, erroneous translations occur for example when two related words in a source language are close to the same words in the target language (e.g., *less* and *more* share the same nearest neighbor in Figure 2: *mehr* 'more'). In such cases, we aim to use monolingual information about lexical relations in the form of discourse cues. Specifically, given two antonymous words a_s and b_s (henceforth $a_s \perp b_s$)

Language	Corpus size (in words)	Data size (in pairs)	Ratio (#syn:#ant)
English (train)	2.1 B	10 932	50:50
German (dev)	0.7 B	811	52:48
Hindi	0.5 B	1 000	50:50
Vietnamese	0.2 B	353	45:55

Table 1: Data statistics of the classification datasets as well as the corpora used to create word representations. Note that the data used here consist of only synonyms and antonyms (approximately in equal proportions).

in the source language, we want to choose translations $a_t^{(i)}$ and $b_t^{(j)}$ that are most likely antonymous as well. We achieve this by re-ranking translations $a_t^{(i)}$ and $b_t^{(j)}$ such that the probability $\Pr(a_t^{(i)} \perp b_t^{(j)})$ of them appearing in contrast relations is maximized. We only assume a set of discourse markers M that indicate contrast as knowledge about the target language. We approximate $\Pr(a_t^{(i)} \perp b_t^{(j)})$ by computing the pointwise mutual information (PMI) between two terms, conditioned on co-occurrence *around* a discourse marker $m \in M$:

$$\Pr(a_t^{(i)} \perp b_t^{(j)}) \sim \min_{m \in M} \log \frac{N(m)\, n(a_t^{(i)}, b_t^{(j)}|m)}{n(a_t^{(i)}|m)\, n(b_t^{(j)}|m)}$$

where $N(m)$ is the frequency of discourse marker m in the corpus, $n(w|m)$ is the number of times term w occurs with m, and $n(w_1, w_2|m)$ is the number of co-occurrences $w_1 ... m ... w_2$. For simplicity, we count all joint occurrences within a sentence, regardless of sentence length and distance between words. In practice, the set M is constructed by manually translating eight discourse markers that frequently indicate contrast relations according to the Penn Discourse Treebank 2.0 (Prasad et al., 2007): *although*, *by comparison*, *by contrast*, *however*, *nevertheless*, *nonetheless*, *though*, and *thus*.

4 Experiments

Our experiments test the performance of an off-the-shelf system for lexical relation classification under different cross-lingual settings. In particular, we evaluate the performance of unsupervised cross-lingual word embeddings and assess the benefits of translation and re-ranking, taking into account word co-occurrences in contrast relations.

Experimental setup. For our experiments, we use four languages: English, German, Hindi and Vietnamese. The antonymy and synonymy dataset by Nguyen et al. (2017) is used for training and estimating an upper bound in English. For cross-lingual development and hyper-parameter selection, we use the German dataset by Glavaš and Vulić (2018). Specifically, we select the number of hidden layers $\{0, \underline{1}\}$, learning rate $\{\underline{0.001}, \ldots, 0.1\}$, dropout rate $\{0.0, \underline{0.5}\}$, and whether to include information on paths between words {yes, $\underline{\text{no}}$}.[1] As examples of under-resourced languages, we evaluate on the Vietnamese dataset ViCon by Nguyen et al. (2018) and on a new Hindi dataset, which we crawled from `hindi2dictionary.com`.[2] Note that annotated data in these languages is held out exclusively for testing.

The task addressed here is to distinguish synonymous from antonymous words. Accordingly, we consider only word pairs for training and testing that are marked as antonyms or synonyms, even if the original dataset also contains unrelated words or other relations. Statistics of all considered word pairs, the ratio between synonyms and antonyms, as well as sizes of the text corpora used in our experiments are given in Table 1.

Models. We test our proposed approach against two baselines: **NoTrans** and **BestTrans**. **NoTrans** simply uses the training data in English and allows us to test in how far semantic properties related to the distinction between synonymy and antonymy are preserved cross-lingually. **BestTrans** uses 1st best translations of the English training data in a target language, as described in Section 3.1.

Our own approach, henceforth called **LingTrans**, is based on n-best translations and exploits word co-occurrences in contrast relations for re-ranking, as described in Section 3.2.

All three models take cross-lingual word embeddings as input. We created these as follows. First, the unsupervised morphological analyzer of Xu et al. (2018) is used to lemmatize the monolingual corpora for the respective languages—the German Wikipedia, the Vietnamese portion of the LORELEI pack (Strassel and Tracey, 2016), and the HindEnCorp corpus (Bojar et al., 2014). This step ensures that we can compare the (stems of)

---

[1] Since we do not assume access to syntactic paths in the target languages, we only experimented with lexical surface paths and found them to consistently degrade performance.

[2] To verify the correctness of the crawled data, we asked a native speaker to validate all instances. Since synonym/antonym labels were derived automatically, validation was only performed to remove noisy instances.

LexNet-crosslingual	de (dev)	hi	vi
NoTrans	59.9	54.6	42.2
BestTrans	62.2	59.2	46.9
LingTrans	**64.2***	**63.5***	**56.8****
LexNet-monolingual (en)		70.1	
+ path embeddings		**74.3**	

Table 2: Macro-averaged F_1-scores for crosslingual synonymy/antonymy distinction in German (de), Hindi (hi), and Vietnamese (vi). Significant differences from NoTrans are marked by asterisk(s) (*$p<0.1$, **$p<0.01$). Monolingual results in English (en) are only shown for comparison.

word tokens in text to those of the word forms that actually appear in the synonymy/antonymy datasets. Note that, while we use an unsupervised morphological analyzer, a stemmer can also be used, if available for that language. Next, we created monolingual embeddings for each language using fastText (Bojanowski et al., 2017). Finally, we applied the unsupervised variant of `vecmap` (Artetxe et al., 2018) to compute alignments and cross-lingual mappings.

The word and discourse marker co-occurrence counts required for our approach are computed on the same monolingual corpora used for training monolingual embeddings.

Results. Table 2 shows macro-averaged F_1-scores for crosslingual synonymy/antonymy distinction (top part) as well as monolingual results in English for comparison (bottom part). The monolingual results show that word embeddings provide appropriate information for classification in most instances, achieving an F_1-score of 70.1. The comparatively low cross-lingual results by **NoTrans** indicate that aligning and mapping embedding spaces does not preserve all semantic properties relevant to the distinction between synonymy and antonymy. The use of first-best translations in **BestTrans** alleviates this issue partially and improves F_1 by up to 4.6 points. Our intuition regarding discourse cues in **LingTrans** leads to further improvements of up to 9.9 additional points in F_1 and considerably closes the gap between monolingual and cross-lingual results (56.8−64.2 vs. 70.1 F_1). Based on an approximate randomization test over the respective test items (Yeh, 2000), we find the improvements of our proposed approach **LingTrans** over **NoTrans** to be significant in Vietnamese ($p<0.01$), German and Hindi ($p<0.1$). In contrast, there is no significant difference in performance between **BestTrans** and **NoTrans**, confirming that both translation *and* reranking are required to achieve consistent gains.

5 Analysis

We examine how far our proposed approach affects performance on the task of synonymy–antonymy distinction and discuss remaining shortcomings.

A first observation concerns the general variance of results across languages. In addition to differences in terms of available corpus sizes, we observe different challenges in each language and dataset. Notably, each dataset is created with its own linguistically motivated definitions of antonymy and synonymy, some of which are more relaxed than others. For example, Nguyen et al. (2018) consider "words which are strongly associated but highly dissimilar to each other" as antonyms and "words that are highly similar in meaning" as synonyms.

The performance of our baseline **NoTrans** suggests that the distinction between synonyms and antonyms in the crosslingual space is harder for Vietnamese than for the other languages. At least partially, this is due to errors and noise from preprocessing (i.e., lemmatization). Also, Vietnamese belongs to a different language family than English. Consequently, information relevant to the synonymy–antonymy distinction may be distributed differently in the target language space than in the source language. In Hindi, some errors specific to the data involve synonyms that denote the same mythological deity (e.g., हनुमान / बजरंगबली 'Hanuman/Bajrangbali') and thus require factual knowledge for classification.

Even for our development language German, results suggest that there is room for improvement. We examine the improvements achieved and remaining errors in German in more detail below.

Classification improvements. The main improvement of our **LingTrans** approach over **BestTrans** is reflected in an increased number of correctly classified antonyms, mostly including antonymous adjectives (e.g., *ständig* 'steadily' vs. *sporadisch* 'sporadically') but also mutually exclusive nouns (e.g., *Privatgelände* 'private grounds' vs. *Öffentlichkeit* 'public'). Compared to the **NoTrans** baseline, we find **BestTrans** to substantially increase the number of correctly classified synonyms (by almost 20%). In particular, this affects words derived from the same stem (*unzählig/zahllos* 'countless'), synonymous verbs (*pfuschen/schummeln* 'cheat') as well as (near-)synonymous adjectives (*verfügbar/verwendbar* 'available/usable').

Translation improvements. The observed quantitative improvements are in line with qualitative improvements that we see in the automatically generated training data. For example, *more* and *less* have the same 1st-best translation in German according to the cross-lingual space (*mehr* 'more'); as illustrated in Figure 2, reranking is required to find the correct translation for the latter word (i.e., *weniger* 'less'). To quantify translation improvements of our proposed reranking method, we count the number of words overlapping between the automatically translated data and data available in the target languages. Compared to the **BestTrans** baseline, we find our **LingTrans** approach to improve overlap for German, Hindi, and Vietnamese by 2%, 5% and 5%, respectively. Despite the increase in word overlap, it is worth noting that the overlap between training and testing in terms of actual data instances (i.e., pairs of words) remains constant between 0% and 1%. A partial explanation for the improved results could thus be that our reranking approach lets us find translations that are *generally* more likely to have an antonym (or synonym)—regardless of which other word they are presented with. A similar observation was made by Levy et al. (2015) regarding hypernymy classification, in which they found some words to be generally more likely to be "category words". Whether similar biases exist in synonymy–antonymy classification will be subject of future work.

Remaining errors. We categorized a randomly selected subset of 30 mis-classifications. We found 30% of the errors to be related to sparsity. In these cases, at least one word was infrequent or the word pair never co-occurred in the corpus (e.g., *unversöhnbar/unvereinbar* 'unforgiving/irreconcilable'). 27% of the errors can be attributed to problems with the test data because it contains "synonyms" that are only indirectly related (e.g., *parieren/ausweichen* 'parry/dodge').

The 42% remaining cases involve a number of challenging antonyms and synonyms, but also a few remarkably easy cases. The latter include morphologically marked antonyms (e.g., *Richtigkeit* 'correctness' vs. *Unrichtigkeit* 'incorrectness') and instances of synonyms that involve two identical words (e.g., *Verwandtschaft/Verwandtschaft* 'kinship'), which presumably is an artifact of the automatic translation step used in creating the German dataset (see Glavaš and Vulić (2018)). Both types of errors could be identified by simple heuristics but are incorrectly classified by the model. On the other hand, hard cases involve words of different linguistic register, which are expected to be distributed differently and therefore hard to capture by distributional methods (e.g., *mittellos/verarmt* 'penniless/impoverished').

In summary, we find substantial improvement through our transfer method, compared to the baselines. However, further improvements could be achieved, for example, taking into account register (e.g., formal vs. colloquial) and morphological marking (e.g., negation affixes).

6 Conclusion

We proposed to combine unsupervised cross-lingual embeddings and discourse cues to generate supervision for distinguishing synonyms and antonyms in under-resourced languages. Compared to a baseline that uses only cross-lingual embeddings, we showed that the use of a small set of discourse markers indicating contrast can yield absolute improvements of up to 10% F_1-score. The simplicity of our approach allows to easily incorporate other features (e.g., morphological marking) and it can be extended to further languages or lexical relations. For example, discourse markers that indicate specification and instantiation relations (e.g. *specifically*, *for instance*) could be used to detect hypernymy (cf. Roth and Schulte im Walde (2014)). Beyond classification, another direction for future work is to extend our approach to distinguish synonyms and antonyms from unrelated word pairs. An interesting direction to pursue would be to use multiple related languages, to aid lexical relation classification in an under-resourced language, instead of transferring supervision from a single language (English).

Acknowledgements

This research was supported by the German Research Foundation (DFG EXC 285), and by Contract HR0011-15-2-0025 and Agreement HR0011-15-2-0023 with the US DARPA.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A Robust Self-learning Method for Fully Unsupervised Cross-lingual Mappings of Word Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 789–798. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Walter G Charles and George A Miller. 1989. Contexts of Antonymous Adjectives. *Applied psycholinguistics*, 10(3):357–375.

Philip Edmonds and Graeme Hirst. 2002. Near-Synonymy and Lexical Choice. *Computational linguistics*, 28(2):105–144.

Goran Glavaš and Ivan Vulić. 2018. Discriminating between Lexico-Semantic Relations with the Specialization Tensor Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 181–187. Association for Computational Linguistics.

Gregory Grefenstette. 1992. Finding Semantic Similarity in raw text: The Deese Antonyms. In *Fall Symposium Series, Working Notes, Probabilistic Approaches to Natural Language*, pages 61–65.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, Contrast and Contradiction in Text Processing. In *AAAI*, volume 6, pages 755–762.

M. A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA. Association for Computational Linguistics.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes Analysis for Better Bilingual Dictionary Induction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 211–220. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976. Association for Computational Linguistics.

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying Synonyms among Distributionally Similar Words. In *IJCAI*, volume 3, pages 1492–1493.

Marie-Catherine Marneffe, Anna N Rafferty, and Christopher D Manning. 2008. Finding Contradictions in Text. In *Proceedings of ACL-08: HLT*, pages 1039–1047.

Yuval Marton, Ahmed El Kholy, and Nizar Habash. 2011. Filtering Antonymous, Trend-contrasting, and Polarity-dissimilar Distributional Paraphrases for Improving Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 237–249. Association for Computational Linguistics.

Saif M Mohammad, Bonnie J Dorr, Graeme Hirst, and Peter D Turney. 2013. Computing Lexical Contrast. *Computational Linguistics*, 39(3):555–590.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic Specialization of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 76–85. Association for Computational Linguistics.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2018. Introducing Two Vietnamese Datasets for Evaluating Semantic Models of (Dis-)Similarity and Relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 199–205. Association for Computational Linguistics.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word Embedding-based Antonym Detection using Thesauri and Distributional Information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989.

Sebastian Padó and Mirella Lapata. 2003. Constructing Semantic Space Models from Parsed Corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Sapporo, Japan. Association for Computational Linguistics.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The Penn Discourse Treebank 2.0 annotation manual. Technical report.

Michael Roth and Sabine Schulte im Walde. 2014. Combining Word Patterns and Discourse Markers for Paradigmatic Relation Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 524–530, Baltimore, Maryland. Association for Computational Linguistics.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29, Osaka, Japan. The COLING 2016 Organizing Committee.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI Language Packs: Data, Tools, and Resources for Technology Development in Low Resource Languages. In *LREC*.

Ellen M Voorhees. 2008. Contradictions and Justifications: Extensions to the Textual Entailment Task. *Proceedings of ACL-08: HLT*, pages 63–71.

Hongzhi Xu, Mitchell Marcus, Charles Yang, and Lyle Ungar. 2018. Unsupervised Morphology Learning with Statistical Paradigms. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 44–54, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics, Volume 2*, pages 947–953.

Wen-tau Yih, Geoffrey Zweig, and John C Platt. 2012. Polarity Inducing Latent Semantic Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1212–1222. Association for Computational Linguistics.

Context-Aware Cross-Lingual Mapping

Hanan Aldarmaki[1] and **Mona Diab**[1,2]
[1]The George Washington University
[2]AWS, Amazon AI
`aldarmaki@gwu.edu,diabmona@amazon.com`

Abstract

Cross-lingual word vectors are typically obtained by fitting an orthogonal matrix that maps the entries of a bilingual dictionary from a source to a target vector space. Word vectors, however, are most commonly used for sentence or document-level representations that are calculated as the weighted average of word embeddings. In this paper, we propose an alternative to word-level mapping that better reflects sentence-level cross-lingual similarity. We incorporate context in the transformation matrix by directly mapping the averaged embeddings of aligned sentences in a parallel corpus. We also implement cross-lingual mapping of deep contextualized word embeddings using parallel sentences with word alignments. In our experiments, both approaches resulted in cross-lingual sentence embeddings that outperformed context-independent word mapping in sentence translation retrieval. Furthermore, the sentence-level transformation could be used for word-level mapping without loss in word translation quality.

1 Introduction

Cross-lingual word vector models aim to embed words from multiple languages into a shared vector space to enable cross-lingual transfer and dictionary expansion (Upadhyay et al., 2016). One of the most common and effective approaches for obtaining bilingual word embeddings is by fitting a linear transformation matrix on the entries of a bilingual seed dictionary (Mikolov et al., 2013). This approach is versatile and scalable: multilingual embeddings can be obtained by mapping the vector spaces of multiple languages into a shared target language, typically English. In addition, imposing an orthogonality constraint on the mapping ensures that the original pair-wise distances are preserved after the transformation and results in better word translation retrieval (Artetxe et al., 2016; Smith et al., 2017).

While word vector spaces tend to be globally consistent across language variations (Aldarmaki et al., 2018), individual words like homographs with unrelated senses (e.g. 'bank', 'coast') and phrasal verbs ('stand up', 'stand out') are likely to behave less consistently in multilingual vector spaces due to their different usage distributions. Consequently, using such words in the alignment dictionary may result in suboptimal overall mapping. We propose two approaches to counteract this effect by incorporating sentential context in the mapping process without explicit word sense disambiguation or additional linguistic resources. The first approach is based on the recently proposed contextualized embeddings from language models, ELMo (Peters et al., 2018). Using a parallel corpus with word-alignments, we extract contextualized embeddings to construct a context-aware dictionary for mapping. The second approach is to learn a transformation between sentence embeddings rather than individual word embeddings. Since these embeddings include context that spans full sentences, we surmise that a mapping learned at this level would be more robust to individual word misalignments.

We used a constrained set of parallel sentences ranging from one hundred to a million sentences for alignment. We then evaluated the resultant mappings on sentence translation retrieval among English, Spanish, and German as test languages. Our results show that context-aware mappings significantly outperform context-independent cross-lingual word mappings using reasonably-sized parallel corpora, particularly when using contextualized word embeddings. In addition, when averaging static word embeddings, the sentence-level mapping can still be used for word-level mapping without loss in word translation quality.

Proceedings of NAACL-HLT 2019, pages 3906–3911
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related Work

For cross-lingual alignment, we follow the popular approach of fitting a linear transformation matrix between word vector spaces that are independently trained for each language. Aligning monolingual word vector spaces using a seed dictionary was originally proposed in Mikolov et al. (2013). In Artetxe et al. (2016) and Smith et al. (2017), it was shown that imposing an orthogonality constraint on the transformation leads to better word translation quality. Recently, contextualized word embeddings were proposed, where a sequential neural network is trained as a language model and then used to extract context-sensitive word representations from the hidden states (Peters et al., 2018). We use parallel text in order to align independently-trained contextualized embeddings across languages. Schuster et al. (2019) independently proposed a cross-lingual alignment approach for contextualized embeddings without the use of parallel text.

3 Approach

3.1 Orthogonal Bilingual Mapping

Given a dictionary of source to target pairs $\langle x, y \rangle$ and matrix representations X and Y whose columns are vector representations of the corresponding dictionary items, we seek to find an orthogonal transformation matrix R that minimizes the distances between the transformed vectors in RX and Y. Formally,

$$R = \arg\min_{\hat{R}} \|\hat{R}X - Y\| \quad \text{s. t.} \quad \hat{R}^T \hat{R} = I \qquad (1)$$

where $\|.\|$ denotes the Frobenius norm. The orthogonality constraint ensures that pair-wise distances in the original source vector space are preserved after the transformation. As shown in (Schönemann, 1966), the solution can be found by singular value decomposition of YX^T

$$YX^T = U\Sigma V^T$$

Then,

$$R = UV^T \qquad (2)$$

The resultant transformation, R, can then be used to transform additional vectors in the source vector space. The quality of the transformation depends on the size and accuracy of the initial dictionary, and it is typically evaluated on word translation precision using nearest neighbor search (Smith et al., 2017).

3.2 Mapping of Contextualized Embeddings

Word embeddings in a given language tend to have similar structures as their translations in a target language (Aldarmaki et al., 2018), which enables orthogonal mappings of word vector spaces to generalize well across various languages. However, items in bilingual dictionaries typically refer to specific word senses. In a given dictionary pair, the source word may have multiple senses that are not consistent with its aligned target translation (and vise versa), which could result in suboptimal global mappings. Intuitively, better mappings could be obtained using sense-disambiguated word embeddings, which could be approximated from context. ELMo (Embeddings from Language Models) is a recently-proposed deep model for obtaining contextualized word embeddings, which are calculated as the hidden states of a bi-LSTM network trained as a language model (Peters et al., 2018). The network can be used in lieu of static word embeddings within other models, which yields better performance in a range of tasks, including word sense disambiguation. Sentence embeddings can be obtained from ELMo by averaging the contextualized word embeddings (Perone et al., 2018).

Since ELMo generates dynamic, context-dependent vectors, we cannot use a simple word-level dictionary to map the model across languages. Instead, we use a parallel corpus with word alignments, i.e using an IBM Model (Brown et al., 1993), to extract a dynamic dictionary of aligned contextualized word embeddings. Depending on the size of the parallel corpus, a large dictionary can be extracted to learn an orthogonal mapping as described in Section 3.1, which is then applied post-hoc on newly generated contextualized embeddings.

3.3 Sentence-Level Mapping

An alternative general approach for obtaining a context-aware mapping is to learn sentence-level transformations. Intuitively, a sentence is less ambiguous than stand-alone words since the words are interpreted within a specific context, so a mapping learned at the sentence-level is likely to be less sensitive to individual word inconsistencies. Therefore, we learn the mapping as described in

Section 3.1 using a dictionary of aligned sentence embeddings. Over a large parallel corpus, the aggregate mapping can yield a more optimal global solution compared to word-level mapping. This approach can be applied using any model capable of generating monolingual sentence embeddings. In this work, we use the average of word vectors in each sentence, where the word vectors are either static or contextualized. For inference, monolingual sentence embeddings are generated first, then mapped to the target space using the sentence-level transformation matrix.[1]

4 Experiments

We used skip-gram with subword information, i.e FastText (Bojanowski et al., 2017), for the static word embeddings, and ELMo for contextualized word embeddings. Sentence embeddings were calculated from ELMo as the arithmetic average of the contextualized embeddings [2]. For FastText, we applied weighted averaging using smooth inverse frequency (Arora et al., 2017), which works better for sentence similarity compared to other averaging schemes (Aldarmaki and Diab, 2018).

4.1 Data and Processing

We trained and aligned all models using the same monolingual and parallel datasets. For monolingual training, we used the 1 Billion Word benchmark (Chelba et al., 2014) for English, and equivalent subsets of $\sim$400 million tokens from WMT'13 (Bojar et al., 2013) news crawl data. We trained monolingual ELMo and FastText with default parameters. We used the WMT'13 common-crawl data for cross-lingual mapping, and the WMT'13 test sets for evaluating sentence translation retrieval. For all datasets, the only preprocessing we performed was tokenization.

4.2 Evaluation Framework

We evaluated the cross-lingual mapping approaches on sentence translation retrieval, where we calculate the accuracy of retrieving the correct translation from the target side of a test parallel corpus using nearest neighbor search with cosine similarity. To assess the minimum bilingual data requirements of each approach and measure how the various models respond to additional data, we split the training parallel corpus into smaller subsets of increasing sizes, starting from 100 to a million sentences (we double the size at each step). Data splits and evaluation scripts are available at `https://github.com/h-aldarmaki/sent_translation_retrieval`.

4.3 Alignment Schemes

For ELMo, word embeddings need to be calculated from context, so we extracted a dictionary of contextualized words from the parallel corpora by first applying word-level alignments using Fast_Align (Dyer et al., 2013). We then calculated the contextualized embeddings for source and target sentences, and extracted a dictionary from the aligned words that have a one-to-one alignment (i.e. we excluded phrasal alignments). Since this can result in a very large dictionary, we capped the number of dictionary words at 1M for efficiency. For a fair comparison with FastText word-level mapping, we extracted a dictionary from word alignment probabilities using the same parallel sets. For each word in the source language, we extracted its translation as the word with the maximum alignment probability if the maximum was unique[3]. As a baseline, we used static dictionaries from (Conneau et al., 2017) to obtain word-level mappings (`dict`). All alignments were performed from the source languages to English.

4.3.1 Results

Sentence translation retrieval results in all language directions are shown in Figure 1 (note the x-axis denotes the size of the alignment corpus in log scale). The arrows indicate the translation direction from source to target, with en for English, es for Spanish, and de for German. For clarity, the legend shows the average accuracies in the final step (1M).

Overall, ELMo word alignment resulted in the highest sentence translation retrieval accuracies, even with small amounts of training data; it exceeded the static dictionary baseline at around 2K parallel sentences. Sentence-level mapping outperformed word-level mapping only when additional parallel data were used (over 50K sentences).

---

[1] Since we use vector averaging, it doesn't matter whether we apply the learned transformation to the word embeddings before averaging, or to the sentence embeddings after averaging.

[2] We found that using the arithmetic average for ELMo yields better results than weighted averaging.

[3] Using other dictionary pairs generally resulted in lower performance.

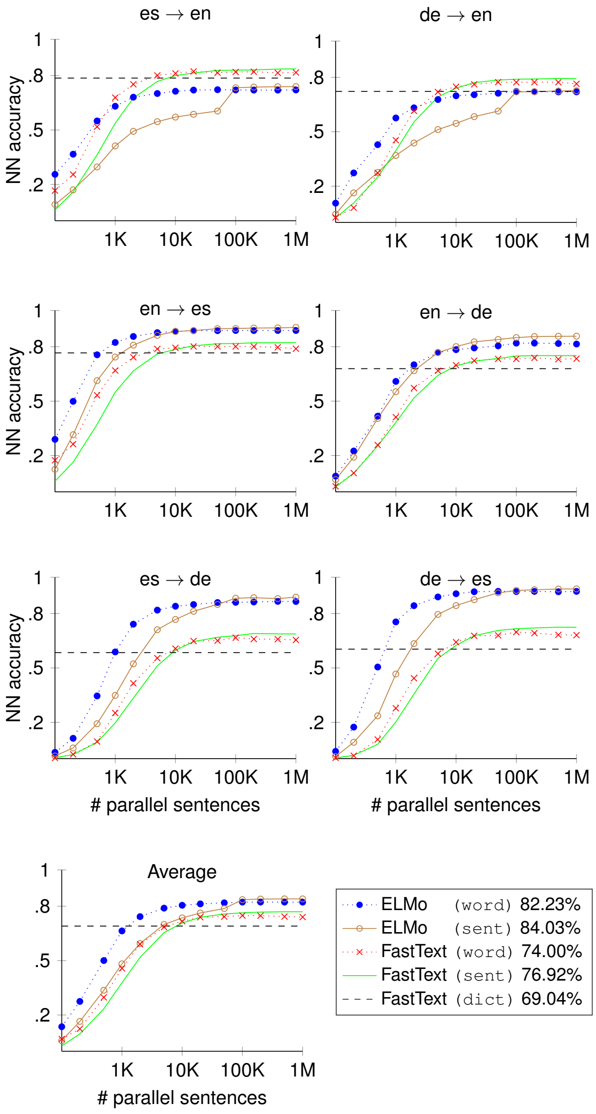


Figure 1: Nearest neighbor sentence translation accuracy as a function of (log) parallel corpus size. `(word)` refers to word-level mapping, `(sent)` to sentence-level mapping, and `(dict)` refers to the baseline (using a static dictionary for mapping). The legend shows the average accuracies of each model using 1M parallel sentences.

With 1M sentences, sentence-level mapping of FastText yielded an increase of ~3% in all directions. Sentence-level ELMo underperformed in the → *en* directions until we used 100K sentences, where we observed a sharp increase in accuracy compared to the previous step of 50K sentences. For ELMo, we note particular improvements in zero-shot translation retrieval between the source languages: *es* and *de*, where ELMo-based models performed much higher than FastText. The opposite is true for the → *en* directions, although the difference is not as notable. This is an interesting observation and may indicate that contextualized dictionaries result in a more balanced mapping, while context-independent embeddings overfit the mapping to the specific direction used for alignment.

Language pair		Mapping level	
		word	sentence
From source language to en:			
es-en	k=1	**56.46**	54.43
	k=5	**70.93**	68.97
de-en	k=1	**50.00**]	47.85
	k=5	**63.45**	62.69
From en to source language:			
en-es	k=1	56.98	**57.52**
	k=5	**72.68**	72.15
en-de	k=1	42.32	**43.27**
	k=5	**63.99**	62.84
Translation between source languages:			
de-es	k=1	36.14	**37.07**
	k=5	53.72	**54.85**
es-de	k=1	31.55	**34.22**
	k=5	51.37	**52.07**
Average	k=1	45.58	**45.73**
	k=5	**62.69**	62.26

Table 1: Word translation precision at k (%) using k nearest neighbor search, with $k \in \{1, 5\}$.

Language pair	Mapping level	
	word	sentence
en-es	0.6280	**0.6362**
en-de	**0.6480**	0.6476
es-de	0.6349	**0.6383**
Average	0.6370	**0.6407**

Table 2: The harmonic mean of Pearson and Spearman correlations with human judgment on the SemEval'17 cross-lingual word similarity task.

4.3.2 Word-level Evaluation

Cross-lingual word embeddings are typically evaluated in word-translation retrieval: the precision of correctly retrieving a translation from the vocabulary of another language. Since this is a context-free task, we evaluated the performance of static word embeddings, FastText, using word vs. sentence mapping (with 1M parallel sentences). The transformation matrix learned at the sentence level is used to transform the word embeddings.

We used the dictionaries from (Conneau et al., 2017). We also evaluated on the SemEval'17 cross-lingual word similarity task (Camacho-Collados et al., 2017), which is measured using the average of Pearson and Spearman correlation coefficients against human judgements. As shown in Tables 1 and 2, the mapping learned at the sentence-level yields equivalent performance to word-level mapping. While word-level mapping was slightly better in translating from source languages (German and Spanish) to English, the sentence-level mapping was better when translating between the source languages. In the word similarity task, sentence-level mappings performed slightly better in two out of the three cases. Overall, the performance of both models are comparable, which indicates that a single transformation matrix learned at the sentence-level can be used for both word and sentence-level tasks.

5 Discussion and Conclusions

We introduced alternatives to the popular word mapping approach that incorporate context in the mapping process. Given parallel corpora, context-aware mappings were learned by mapping aligned contextualized word embeddings or directly mapping the parallel sentence embeddings. Experimental results showed significant gains in sentence translation retrieval using contextualized mappings compared to context-independent word mapping. While word-level mappings worked better with smaller parallel corpora, the performance of sentence-level mapping continued to increase with additional data until it outperformed word-level mapping. In future work, we will explore the viability of the sentence mapping approach on other sentence embedding models.

References

Hanan Aldarmaki and Mona Diab. 2018. Evaluation of unsupervised compositional representations. *Proceedings of the 27th International Conference on Computational Linguistics*.

Hanan Aldarmaki, Mahesh Mohan, and Mona Diab. 2018. Unsupervised word mapping using structural similarities in monolingual embeddings. *Transactions of the Association of Computational Linguistics*, 6.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the eighth workshop on statistical machine translation*, pages 1–44.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv preprint arXiv:1902.09492*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Polyglot Contextual Representations Improve Crosslingual Transfer

Phoebe Mulcaire♡ **Jungo Kasai**♡ **Noah A. Smith**♡♢
♡Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
♢Allen Institute for Artificial Intelligence, Seattle, WA, USA
`{pmulc,jkasai,nasmith}@cs.washington.edu`

Abstract

We introduce Rosita, a method to produce multilingual contextual word representations by training a single language model on text from multiple languages. Our method combines the advantages of contextual word representations with those of multilingual representation learning. We produce language models from dissimilar language pairs (English/Arabic and English/Chinese) and use them in dependency parsing, semantic role labeling, and named entity recognition, with comparisons to monolingual and non-contextual variants. Our results provide further evidence for the benefits of polyglot learning, in which representations are shared across multiple languages.

1 Introduction

State-of-the-art methods for crosslingual transfer make use of multilingual word embeddings, and much research has explored methods that align vector spaces for words in different languages (Faruqui and Dyer, 2014; Upadhyay et al., 2016; Ruder et al., 2017). On the other hand, *contextual* word representations (CWR) extracted from language models (LMs) have advanced the state of the art beyond what was achieved with word type representations on many monolingual NLP tasks (Peters et al., 2018). Thus, the question arises: can *contextual* word representations benefit from *multilinguality*?

We introduce a method to produce multilingual CWR by training a single "polyglot" language model on text in multiple languages. As our work is a multilingual extension of ELMo (Peters et al., 2018), we call it Rosita (after a bilingual character from *Sesame Street*). Our hypothesis is that, although each language is unique, different languages manifest similar characteristics (e.g., morphological, lexical, syntactic) which can be exploited by training a single model with data from multiple languages (Ammar, 2016). Previous work has shown this to be true to some degree in the context of syntactic dependency parsing (Ammar et al., 2016), semantic role labeling (Mulcaire et al., 2018), named entity recognition (Xie et al., 2018), and language modeling for phonetic sequences (Tsvetkov et al., 2016) and for speech recognition (Ragni et al., 2016). Recently, de Lhoneux et al. (2018) showed that parameter sharing between languages can improve performance in dependency parsing, but the effect is variable, depending on the language pair and the parameter sharing strategy. Other recent work also reported that concatenating data from different languages can hurt performance in dependency parsing (Che et al., 2018). These mixed results suggest that while crosslingual transfer in neural network models is a promising direction, the best blend of polyglot and language-specific elements may depend on the task and architecture. However, we find overall contextual representations from polyglot language models succeed in a range of settings, even where multilingual word type embeddings do not, and are a useful technique for crosslingual transfer.

We explore crosslingual transfer between highly dissimilar languages (English→Chinese and English→Arabic) for three core tasks: Universal Dependency (UD) parsing, semantic role labeling (SRL), and named entity recognition (NER). We provide some of the first work using polyglot LMs to produce contextual representations,[1] and the first analysis comparing them to monolingual LMs for this purpose. We also introduce an LM variant which takes multilingual word embedding input as well as character input, and explore its

---

[1] Contemporaneous work uses polyglot LMs for natural language inference and machine translation (Lample and Conneau, 2019).

Proceedings of NAACL-HLT 2019, pages 3912–3918
Minneapolis, Minnesota, June 2 - June 7, 2019.

applicability for producing contextual word representations. Our experiments focus on comparisons in three dimensions: monolingual vs. polyglot representations, contextual vs. word type embeddings, and, within the contextual representation paradigm, purely character-based language models vs. ones that include word-level input.

Previous work has shown that contextual representations offer a significant advantage over traditional word embeddings (word type representations). In this work, we show that, on these tasks, polyglot character-based language models can provide benefits on top of those offered by contextualization. Specifically, even when crosslingual transfer with word type embeddings hurts target language performance relative to monolingual models, polyglot *contextual* representations can improve target language performance relative to monolingual versions, suggesting that polyglot language models tie dissimilar languages in an effective way.

In this paper, we use the following terms: *crosslingual transfer* and *polyglot learning*. While crosslingual transfer is often used in situations where target data are absent or scarce, we use it broadly to mean any method which uses one or more source languages to help process another target language. We also draw a sharp distinction between multilingual and polyglot models. Multilingual learning can happen independently for different languages, but a polyglot solution provides a single model for multiple languages, e.g., by parameter sharing between languages in networks during training.

2 Polyglot Language Models

We first describe the language models we use to construct multilingual (and monolingual) CWR.

2.1 Data and Preprocessing

Because the Universal Dependencies treebanks we use for the parsing task predominantly use Traditional Chinese characters and the Ontonotes data for SRL and NER consist of Simplified Chinese, we train separate language models for the two variants. For English we use text from the Billion Word Benchmark (Chelba et al., 2013), for Traditional Chinese, wiki and web data provided for the CoNLL 2017 Shared Task (Ginter et al., 2017), for Simplified Chinese, newswire text from Xinhua,[2] and for Arabic, newswire text from AFP.[3] We use approximately 60 million tokens of news and web text for each language.

We tokenized the language model training data for English and Simplified Chinese using Stanford CoreNLP (Manning et al., 2014). The Traditional Chinese corpus was already pre-segmented by UDPipe (Ginter et al., 2017; Straka et al., 2016). We found that the Arabic vocabulary from AFP matched both the UD and Ontonotes data reasonably well without additional tokenization. We also processed all corpora to normalize punctuation and remove non-text.

2.2 Models and Training

We base our language models on the ELMo method (Peters et al., 2018), which encodes each word with a character CNN, then processes the word in context with a word-level LSTM.[4] Following Che et al. (2018), who used 20 million words per language to train monolingual language models for many languages, we use the same hyperparameters used to train the monolingual English language model from Peters et al. (2018), except that we reduce the internal LSTM dimension from 4096 to 2048.

For each target language dataset (Traditional Chinese, Simplified Chinese, and Arabic), we produce:

- a monolingual language model with character CNN (MONOCHAR) trained on that language's data;
- a polyglot LM (ROSITACHAR) trained with the same code, on that language's data with an additional, equal amount of English data;
- a modified polyglot LM (ROSITAWORD), described below.

The ROSITAWORD model concatenates a 300 dimensional word type embedding, initialized with multilingual word embeddings, to the character CNN encoding of the word, before passing this combined vector to the bidirectional LSTM.

---

[2] `catalog.ldc.upenn.edu/LDC95T13`

[3] `catalog.ldc.upenn.edu/LDC2001T55`

[4] A possible alternative is BERT (Devlin et al., 2018), which uses a bidirectional objective and a transformer architecture in place of the LSTM. Notably, one of the provided BERT models was trained on several languages in combination, in a simple polyglot approach (see `https://github.com/google-research/bert/blob/master/multilingual.md`). Our initial exploration of multilingual BERT models raised sufficient questions about preprocessing that we defer exploration to future work.

The idea of this word-level initialization is to bias the model toward crosslingual sharing; because words with similar meanings have similar representations, the features that the model learns are expected to be at least partially language-agnostic. The word type embeddings used for these models, as well as elsewhere in the paper, are trained on our language model training set using the fastText method (Bojanowski et al., 2017), and target language vectors are aligned with the English ones using supervised MUSE[5] (Conneau et al., 2018). See appendix for more LM training details.

3 Experiments

All of our task models (UD, SRL, and NER) are implemented in AllenNLP, version 0.7.2 (Gardner et al., 2018).[6] We generally follow the default hyperparameters and training schemes provided in the AllenNLP library regardless of language. See appendix for the complete list of our hyperparameters. For each task, we experiment with five types of word representations: in addition to the three language model types (MONOCHAR, ROSITACHAR, and ROSITAWORD) described above, we show results for the task models trained with monolingual and polyglot non-contextual word embeddings.

After pretraining, the word representations are fine-tuned to the specific task during task training. In non-contextual cases, we fine-tune by updating word embeddings directly, while in contextual cases, we only update coefficients for a linear combination of the internal representation layers for efficiency (Peters et al., 2018). In order to properly evaluate our models' generalization ability, we ensure that sentences in the test data are excluded from the data used to train the language models.

3.1 Universal Dependency Parsing

We use a state-of-the-art graph-based dependency parser with BiLSTM and biaffine attention (Dozat and Manning, 2017). Specifically, the parser takes as input word representations and 100-dimensional fine-grained POS embeddings following Dozat and Manning (2017). We use the same UD treebanks and train/dev./test splits as the CoNLL 2018 shared task on multilingual dependency parsing (Zeman et al., 2018). In particular, we use the GUM treebank for English,[7] GSD for Chinese, and PADT for Arabic. For training and validation, we use the provided gold POS tags and word segmentation.

For each configuration, we run experiments five times with random initializations and report the mean and standard deviation. For testing, we use the CoNLL 2018 evaluation script and consider two scenarios: (1) gold POS tags and word segmentations and (2) predicted POS tags and word segmentations from the system outputs of Che et al. (2018) and Qi et al. (2018).[8] The former scenario enables us to purely assess parsing performance; see column 3 in Table 1 for these results on Chinese and Arabic. The latter allows for a direct comparison to the best previously reported parsers (Chinese, Che et al., 2018; Arabic, Qi et al., 2018). See Table 2 for these results.

As seen in Table 1, the Universal Dependencies results generally show a significant improvement from the use of CWR. The best results for both languages come from the ROSITACHAR LM and polyglot task models, showing that polyglot training helps, but that the word-embedding initialization of the ROSITAWORD model does not necessarily lead to a better final model. The results also suggest that combining ROSITACHAR LM and polyglot task training is key to improve parsing performance. Table 2 shows that we outperform the state-of-the-art systems from the shared task competition. In particular, our LMs even outperform the Harbin system, which uses monolingual CWR and an ensemble of three biaffine parsers.

3.2 Semantic Role Labeling

We use a strong existing model based on BIO tagging on top of a deep interleaving BiLSTM with highway connections (He et al., 2017). The SRL model takes as input word representations and 100-dimensional predicate indicator embeddings following He et al. (2017). We use a standard PropBank-style, span-based SRL dataset for English, Chinese, and Arabic: Ontonotes (Pradhan et al., 2013). Note that Ontonotes provides annotations using a single shared annotation scheme for

---

[5] For our English/Chinese and English/Arabic data, their unsupervised method yielded substantially worse results in word translation.

[6] We make our multilingual fork available at `https://github.com/pmulcaire/rosita`

[7] While there are several UD English corpora, we choose the GUM corpus to minimize domain mismatch.

[8] System outputs for all systems are available at `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2885`

vectors (lang.)	task lang.	UD LAS	SRL F_1	NER F_1
fastText (CMN)	CMN	85.15$_{\pm0.12}$	69.79	76.31
fastText (CMN+ENG)	CMN+ENG	84.92$_{\pm0.28}$	70.82	76.05
MonoChar (CMN)	CMN	87.55$_{\pm0.25}$	74.14	78.18
RositaChar (CMN+ENG)	CMN	87.16$_{\pm0.08}$	74.24	**78.29**
RositaChar (CMN+ENG)	CMN+ENG	**87.75**$_{\pm0.16}$	74.69	77.68
RositaWord (CMN+ENG)	CMN	86.50$_{\pm0.17}$	**74.84**	77.19
RositaWord (CMN+ENG)	CMN+ENG	86.37$_{\pm0.35}$	74.69	77.16
Best prior work	CMN	–	62.83	75.63
fastText (ARA)	ARA	82.58$_{\pm0.51}$	50.50	71.60
fastText (ARA+ENG)	ARA+ENG	82.67$_{\pm0.46}$	54.82	71.45
MonoChar (ARA)	ARA	84.98$_{\pm0.18}$	**59.55**	75.02
RositaChar (ARA+ENG)	ARA	84.98$_{\pm0.12}$	58.69	75.56
RositaChar (ARA+ENG)	ARA+ENG	**85.24**$_{\pm0.13}$	59.29	**76.19**
RositaWord (ARA+ENG)	ARA	84.34$_{\pm0.20}$	58.34	74.02
RositaWord (ARA+ENG)	ARA+ENG	84.24$_{\pm0.13}$	59.47	72.79
Best prior work	ARA	–	48.68	68.02

Table 1: LAS for UD parsing, F_1 for SRL, and F_1 for NER, with different input representations. For UD, each number is an average over five runs with different initialization, with standard deviation. SRL/NER results are from one run. The "task lang." column indicates whether the UD/SRL/NER model was trained on annotated text in the target language alone, or a blend of English and the target language data. RositaWord LMs use as word-level input the same multilingual word vectors as fastText models. The best prior result for Ontonotes Chinese NER is in Shen et al. (2018); the others are from Pradhan et al. (2013).

LM type	task lang.	LAS
Harbin (Che et al., 2018) CMN		76.77
Harbin (non-ensemble) CMN		75.55
RositaChar	CMN	77.40
RositaChar	CMN+ENG	**77.63**
Stanford (Qi et al., 2018) ARA		77.06
RositaChar	ARA	77.79
RositaChar	ARA+ENG	**78.02**

Table 2: LAS (F_1) comparison to the winning systems for each language in the CoNLL 2018 shared task for UD. We use predicted POS and the segmentation of the winning system for that language. The RositaChar LM variant was selected based on development performance in the gold-segmentation condition.

English, Chinese, and Arabic, which can facilitate crosslingual transfer. For Chinese and English we simply use the provided surface form of the words. The Arabic text in Ontonotes has diacritics to indicate vocalization which do not appear (or only infrequently) in the original source or in our language modeling data. We remove these for better consistency with the language model vocabulary. We use gold predicates and the CoNLL 2005 evaluation script for the experiments below to ensure our results are comparable to prior work. See column 4 in Table 1 for results on the CoNLL-2012 Chinese and Arabic test sets.

The SRL results confirm the advantage of CWR. Unlike the other two tasks, multilingual word type embeddings are better than monolingual versions in SRL. Perhaps relatedly, models using RositaWord are more successful here, providing the highest performance on Chinese. One unusual result is that the model using the MonoChar LM is most successful for Arabic. This may be linked to the poor results on Arabic SRL overall, which are likely due to the much smaller size of the corpus compared to Chinese (less than 20% as many annotated predicates) and higher proportion of language-specific tags. Such language-specific tags in Arabic could limit the effectiveness of shared English-Arabic representations. Still, polyglot methods' performance is only slightly behind.

3.3 Named Entity Recognition

We use the state-of-the-art BiLSTM-CRF NER model with the BIO tagging scheme (Peters et al., 2017). The network takes as input word representations and 128-dimensional character-level embeddings from a character LSTM. We again use the Ontonotes dataset with the standard data splits. See the last column in Table 1 for results on the CoNLL-2012 Chinese and Arabic test sets. As with most other experiments, the NER results show a strong advantage from the use of contextual representations and a smaller additional advantage from those produced by polyglot LMs.

4 Discussion

Overall, our results show that polyglot language models produce very useful representations. While Universal Dependency parsing, Arabic SRL, and Chinese NER show models using contextual representations outperform those using word type representations, the advantage from polyglot training in some cases is minor. However, Chinese SRL and Arabic NER show strong improvement both from contextual word representations and from polyglot training. Thus, while the benefit of crosslingual transfer appears to be somewhat variable and task dependent, polyglot training is helpful overall for contextual word representations. Notably, the ROSITACHAR LM does not involve any direct supervision of tying two languages together, such as bilingual dictionaries or parallel corpora, yet is still most often able to learn the most effective representations. One explanation is that it automatically learns crosslingual connections from unlabeled data alone. Another possibility, though, is that the additional data provided in polyglot training produces a useful regularization effect, improving the target language representations without crosslingual sharing (except that induced by shared vocabulary, e.g., borrowings, numbers, or punctuation). Nevertheless, the success of polyglot language models is worth further study.

5 Conclusion

We presented a method for using polyglot language models to produce multilingual, contextual word representations, and demonstrated their benefits, producing state-of-the-art results in multiple tasks. These results provide a foundation for further study of polyglot language models and their use as unsupervised components of multilingual models.

Acknowledgments

The authors thank Mark Neumann for assistance with the AllenNLP library and the anonymous reviewers for their helpful feedback. This research was funded in part by NSF grant IIS-1562364, the Funai Overseas Scholarship to JK, and the NVIDIA Corporation through the donation of a GeForce GPU.

References

Waleed Ammar. 2016. *Towards a Universal Analyzer of Natural Languages*. Ph.D. thesis, Carnegie Mellon University.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. arXiv:1312.3005.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proc. of ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*, pages 462–471.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. arXiv:1803.07640.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*, pages 473–483.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A method for stochastic optimization. In *Proc. of ICLR*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. arxiv:1901.07291.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Sgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proc. of EMNLP*, pages 4992–4997.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of ACL, System Demonstrations*, pages 55–60.

Phoebe Mulcaire, Swabha Swayamdipta, and Noah A Smith. 2018. Polyglot semantic role labeling. In *Proc. of ACL*, volume 2, pages 667–672.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proc. of CoNLL*, pages 143–152.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170.

Anton Ragni, Edgar Dakin, Xie Chen, Mark J. F. Gales, and Kate Knill. 2016. Multi-language neural network language models. In *Proc. of INTERSPEECH*, volume 08-12-September-2016, pages 3042–3046.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. arXiv:1706.04902.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *Proc. ICLR*.

Milan Straka, Jan Hajic, and Jana Strakov. 2016. UDPipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proc. of LREC*, pages 4290–4297.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proc. of NAACL-HLT*, pages 1357–1366.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proc. of ACL*, pages 1661–1670.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proc. of EMNLP*, pages 369–379.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arxiv:1212.5701.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21.

A Supplementary Material

In this supplementary material, we provide hyperparameters used in our models for easy replication of our results.

A.1 Language Models

Character CNNs	
Char embedding size	16
(# Window Size, # Filters)	(1, 32), (2, 32), (3, 68), (4, 128), (5, 256), 6, 512), (7, 1024)
Activation	Relu
Word-level LSTM	
LSTM size	2048
# LSTM layers	2
LSTM projection size	256
Use skip connections	Yes
Inter-layer dropout rate	0.1
Training	
Batch size	128
Unroll steps (Window Size)	20
# Negative samples	64
# Epochs	10
Adagrad (Duchi et al., 2011) lrate	0.2
Adagrad initial accumulator value	1.0

Table 3: Language Model Hyperparameters.

Seen in Table 3 is a list of hyperparameters for our language models. We generally follow Peters et al. (2018) and use their publicly available code for training.[9] For character only models, we halve the LSTM and projection sizes to expedite training and to compensate for the greatly reduced training data—their hyperparameters were tuned on around 30M sentences, while we used less than 3M sentences (60-70M tokens) per language.

A.2 UD Parsing

For UD parsing, we generally follow the hyperparameters provided in AllenNLP (Gardner et al., 2018). See a list of hyperparameters in Table 4.

A.3 Semantic Role Labeling

For SRL, we again follow the hyperparameters given in AllenNLP (Table 5). The one exception is that we used 4 layers of alternating BiLSTMs instead of 8 layers to expedite the training process.

A.4 Named Entity Recognition

We again use the hyperparameter configurations provided in AllenNLP. See Table 6 for details.

Input	
POS embedding size	100
Input dropout rate	0.3
Word-level BiLSTM	
LSTM size	400
# LSTM layers	3
Recurrent dropout rate	0.3
Inter-layer dropout rate	0.3
Use Highway Connection	Yes
Multilayer Perceptron, Attention	
Arc MLP size	500
Label MLP size	100
# MLP layers	1
Activation	Relu
Training	
Batch size	80
# Epochs	80
Early stopping	50
Adam (Kingma and Ba, 2015) lrate	0.001
Adam β_1	0.9
Adam β_2	0.999

Table 4: UD Parsing Hyperparameters.

Input	
Predicate indicator embedding size	100
Word-level Alternating BiLSTM	
LSTM size	300
# LSTM layers	4
Recurrent dropout rate	0.1
Use Highway Connection	Yes
Training	
Batch size	80
# Epochs	80
Early stopping	20
Adadelta (Zeiler, 2012) lrate	0.1
Adadelta ρ	0.95
Gradient clipping	1.0

Table 5: SRL Hyperparameters.

Char-level LSTM	
Char embedding size	25
Input dropout rate	0.5
LSTM size	128
# LSTM layers	1
Word-level BiLSTM	
LSTM size	200
# LSTM layers	3
Inter-layer dropout rate	0.5
Recurrent dropout rate	0.5
Use Highway Connection	Yes
Multilayer Perceptron	
MLP size	400
Activation	tanh
Training	
Batch size	64
# Epochs	50
Early stopping	25
Adam (Kingma and Ba, 2015) lrate	0.001
Adam β_1	0.9
Adam β_2	0.999
L2 regularization coefficient	0.001

Table 6: NER Hyperparameters.

---

[9] `https://github.com/allenai/bilm-tf`

Typological Features for Multilingual Delexicalised Dependency Parsing

Manon Scholivet **Franck Dary** **Alexis Nasr** **Benoit Favre** **Carlos Ramisch**
Aix-Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
`firstname.lastname@lis-lab.fr`

Abstract

The existence of universal models to describe the syntax of languages has been debated for decades. The availability of resources such as the Universal Dependencies treebanks and the World Atlas of Language Structures make it possible to study the plausibility of universal grammar from the perspective of dependency parsing. Our work investigates the use of high-level language descriptions in the form of typological features for multilingual dependency parsing. Our experiments on multilingual parsing for 40 languages show that typological information can indeed guide parsers to share information between similar languages beyond simple language identification.

1 Introduction

Human languages may share some syntactic features, but differ on others. For example, some languages tend to place the subject before the verb (e.g., English) whereas others favour the reverse order (e.g., Arabic), and some do not exhibit a clear preference (e.g., Polish). These features can be viewed as the *parameters* of a language's syntax (Greenberg, 1963; Chomsky, 1995).

When training a multilingual parser, it could be interesting to explicitly represent these parameters, and to integrate them into the parsing model. If a successful strategy to do so was found, then, a parser could be trained simultaneously on several languages whose syntactic parameters have been explicitly represented. Such parser could then use a single model to parse texts in any language with known syntactic parameters.

In theory, if we had at our disposal a set of parameters that completely describes the syntax of languages as well as treebanks that explore the whole space of parameters and their values, then such a universal parser could be designed. To make such a program realistic, though, several issues have to be addressed. In this paper, we propose to study the feasibility of learning such multilingual parser by addressing some of these issues.

The first one is the choice of syntactic parameters that will be used (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015). In our work, we approximate these parameters by extracting syntactic information from the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013). [1] A language is represented by a vector containing the values it selects in the WALS. This vector plays the role of the parameters mentioned above.

The second issue is the design of a unified scheme for representing syntax. Our natural choice is the Universal Dependencies (UD) initiative. [2] UD specifically proposes a set of universal dependency relations, part-of-speech tags and morphological features (Nivre et al., 2016). The UD treebanks are available for many languages, annotated according to common guidelines.

The third issue is the lexicon. UD proposes a common language for describing languages' morpho-syntax, but we do not dispose of a "universal lexicon" to which we can map the lexical units of different languages. The solution adopted in this work is to resort to delexicalised parsing (Zeman and Resnik, 2008). This technique consists in ignoring the lexicon when training a parser. Such impoverishment of the data leads to less accurate parsers, but offers a simple solution to the lexicon issue. Using an alternative solution for representing words in different languages, such as multilingual word embeddings, would have introduced in our experimental setting some biases that are difficult to assess and would have prevented

---

[1] `https://wals.info/`

[2] `http://universaldependencies.org`

Proceedings of NAACL-HLT 2019, pages 3919–3930
Minneapolis, Minnesota, June 2 - June 7, 2019.

to measure the precise influence of the typological features on the behaviour of the parser.

The fourth issue concerns the parser, which must be language independent and produce syntactic trees based on combinations of parameter values and sentential configurations. We use a transition-based parser with a multi-layer perceptron classifier (Chen and Manning, 2014), responsible for proposing how parameter values match observable patterns in the data.

Our research hypotheses are: (a) features derived from the WALS enable cross-lingual sharing in multilingual parsing, and (b) these features do more than acting as mere language identifiers. Our main contributions are to reassess the utility of the WALS as informant of typological features of parsed languages, to evaluate their benefit in a controlled multilingual setting with full supervision, and to perform a set of analyses to better understand how they interact with the parser model. In addition to multilingual parsing, our method is suitable for zero-shot learning for under-resourced languages (Ammar et al., 2016; Guo et al., 2015).

After discussing related work (Sec. 2), we describe UD (Sec. 3), the WALS (Sec. 4) and our parser (Sec. 5). The experimental setup (Sec. 6) precedes our results (Sec. 7), analyses (Sec. 8) and conclusions (Sec. 9).

2 Related Work

Our work is at the intersection of three trends in the multilingual dependency parsing literature. The first is *transfer parsing*, when a parser is trained on a language (or a collection of languages) and tested on another one. The second is *delexicalised parsing*, which aims at abstracting away from the lexicon in order to neutralise genre, domain and topic biases which are heavily marked in the treebanks' vocabulary. The third trend is the use of a handcrafted *typological resources*, such as the WALS, in multilingual NLP methods.

Transfer parsing is often a suitable solution when dealing with low-resource languages (McDonald et al., 2011). Projected transfer relies on parallel corpora in which one of the languages does not have labelled training data to learn a parser, but the other does. One commonly employed solution is to use word alignments to project parsed sentences from one side onto the low-resource side of the parallel text, using heuristics (Hwa et al., 2005) or partial annotations (Lacroix et al., 2016). Agić et al. (2016) parse the resource-rich languages in a multi-parallel corpus, proposing a projection method to obtain POS tags and dependency trees for low-resource languages from multiple-language word alignments. The parsing model for the target language can also be obtained in an unsupervised fashion, by optimising a function that combines the likelihood of parallel data and the likelihood of the transferred model on non-annotated data in the low-resource language (Ma and Xia, 2014).

Instead of assuming the availability of parallel corpora, direct transfer approaches capitalize on language similarities. For instance, Lynn et al. (2014) build parser for Irish by first training a delexicalised parser on another language, and then applying it on Irish. They surprisingly found out that Indonesian was the language providing the best parsing results for Irish, even if they do not belong to the same language family, because long-distance dependencies are better represented in Indonesian than in the other languages tested.

Low-resource languages may have some (insufficient) amount of training material available. One can employ bilingual parsing, concatenating training corpora in two languages, to verify if there is an improvement in the results compared to a monolingual parser (Vilares et al., 2015). Direct transfer and bilingual parsing methods are close to the present article, since we also concatenate training corpora. However, in our case, we combine treebanks from many more sources (around 40 languages) and include typological features.

The combination of corpora in multiple languages for parser training is facilitated by the recent advent of multilingual standards and resources, in particular in Universal Dependencies for dependency syntax (Nivre et al., 2016). This initiative enables the annotation of POS, morphology and syntactic dependencies for all languages with the same guidelines and label sets. The availability of such corpora favours the development of cross-lingual methods (Tiedemann, 2015).

Multilingual parsing research is also encouraged by initiatives such as the CoNLL 2017 and 2018 shared tasks, on highly multilingual dependency parsing from raw text (Zeman et al., 2017, 2018).

Delexicalised parsers ignore the word forms and lemmas when analysing a sentence, usually relying on more abstract features such as word classes and POS tags. The use of delexicalised parsers is especially relevant when learning multilingual parsers, since languages generally share only a limited amount of lexical units. The approach proposed by Zeman and Resnik (2008) consists in adapting a parser for a new related language using either parallel corpora or delexicalised parsing. This method can be used to quickly construct a parser if the source and target languages are sufficiently related. McDonald et al. (2011) show that delexicalised parsers can be directly transferred between languages, yielding significantly higher accuracy than unsupervised parsers.

Moreover, typological features such as those present in the WALS provide information about the structure of languages (Dryer and Haspelmath, 2013). These could be useful to guide multilingual parsers, informing them about the model parameters that can be shared among languages with similar characteristics. Naseem et al. (2012) and Zhang and Barzilay (2015) use word-order features available for all their languages, while Ponti et al. (2018) used features they judged relevant in many categories (not only word order). The parameters proposed in the WALS are not the only way to represent properties of languages. Methods based on language embeddings (Östling and Tiedemann, 2017; Bjerva et al., 2019) also constitute interesting language representation.

Täckström et al. (2013) use a multilingual delexicalised transfer method, showing how selective parameter sharing, based on typological features and language family membership, can be incorporated in a discriminative graph-based dependency parser. They select the typological features based on those used by Naseem et al. (2012), removing two features not considered useful.

The work closest to ours experimented with concatenating treebanks to train a multilingual parser (Ammar et al., 2016). The authors use an S-LSTM transition-based parser similar to ours (although we do not include recurrent representations) trained on a set of lexicalised features that include multilingual word embeddings, Brown clusters, and fine-grained POS tags, whereas we only use coarse-grained POS and morphological features in a delexicalised setting. They include a one-hot language-ID vector, a set of six word-order features from the WALS (Naseem et al., 2012), or the whole WALS vectors. We use the two former plus a set of 22 selected features from WALS. They perform experiments on seven high-resourced languages while we report results on a larger set of 40 languages. Although Ammar et al. (2016) showed that, in a lexicalised setting, treebank concatenation could perform on par with monolingual parsers, the origins and limits of these improvements are not clear. We explore directions for assessing the benefits of typological features in a delexicalised parser.

3 Universal Dependencies

A major issue in multilingual parsing is the consistency of annotation across languages, since most corpora are annotated using different guidelines and tagsets. Universal Dependencies (UD) is an initiative whose goal is to create cross-linguistically consistent treebanks, facilitating cross-lingual analyses for language and parsing studies. Currently at version 2.3, 129 treebanks in 76 languages are available.

We use the UD v2.0 release for training and development, and the CoNLL 2017 shared task test sets for evaluation. For training and development, 64 UD treebanks in 45 languages are available. These treebanks vary in size: some are very small (e.g., 529 words for Kazakh), whereas others can be rather large (e.g., 1,842,867 words for Czech). Test corpora contain at least 10,000 words per language and are available for 49 languages.[3]

We learn delexicalised parsers from the UD treebanks using universal parts of speech (UPOS) and morphological features (FEAT) as input, and predicting labelled dependency trees which include language-specific extensions (e.g., *acl:relcl*). Morphological features are present in almost all treebanks, but exhibit high variability. Therefore, we choose to keep only the 16 most frequent features (e.g., Number, Case, VerbForm), which appear in at least 28 languages. Furthermore, morphology is represented as a list of *key=value* pairs, which we split so that each pair is considered separately, yielding a fixed set of 16 morphological features per word.

4 World Atlas of Language Structures

The World Atlas of Language Structures (WALS) is a database of structural (phonological, grammatical and lexical) properties of languages gathered by 55 authors from descriptive materials such

---

[3] 4 languages do not have training sets.

as reference grammars. We have used this resource to associate to every language of UD corpora a set of features describing its properties that are relevant for syntactic parsing.

The WALS describes 2,676 languages with a set of 192 features, organized into 11 feature genus (e.g. Phonology, Word Order). It can therefore be represented as a matrix W of 2,676 rows and 192 columns, in which cell $W(l, f)$ gives the value of feature f for language l, and each row $W(l)$ is the feature vector of a language l. This matrix has been pruned and completed to match our experimental setup. First, we have kept only the rows corresponding to the 49 languages of our test corpora. Conversely, four UD languages do not appear in the WALS and have been left aside: Old Church Slavonic (cu), Gothic (got), Ancient Greek (grc), and Latin (la). As a result, we obtain a reduced version of W containing 45 rows.

We experimented with two language representations obtained from the WALS. The first one, henceforth W_N, is based on the work of Naseem et al. (2012). They selected the six Word Order features available for all their 17 target languages, identified by the codes 81A, 85A, 86A, 87A, 88A, 89A[4]. These features cover phenomena such as verb-object and adjective-noun order, and have been widely discussed in the literature (Täckström et al., 2013; Zhang and Barzilay, 2015; Ammar et al., 2016). The resulting matrix has 45 rows (languages) and 6 columns (features). However, the WALS seen as a matrix is sparse, as some features are unspecified for some languages. Therefore, we chose to keep only languages for which at most half of this vector is unspecified, resulting in the removal of 5 more languages: Galician (gl), Upper Sorbian (hsb), Kazakh (kk), Slovak (sk), and Uyghur (ug). All our experiments are carried out on this set of 40 languages.

The second language representation proposed in this work, henceforth W_{80}, is a relaxed version of W_N. Since the WALS is sparse, we include in W_{80} all features specified for at least 80% of our 40 languages. Furthermore, in addition to features from the Word Order family, we also include features from the Simple Clauses family. This results in a matrix of 40 rows and 22 columns corresponding to 3 features from the Simple Clauses family (101A, 112A, 116A), and 19 from the Word Order family (81A, 82A, 83A, 85A, 86A, 87A, 88A, 89A, 90A, 92A, 94A, 95A, 96A, 97A, 144A, 143A, 143E, 143F, 143G).[5]

	Romance	Germanic	Slavic	Random
W_N	0.33	1.33	0.67	2.41
W_{80}	4.13	4.47	4.19	10.15

Table 1: MID values of typological language genus compared to Random. Random MID is the average of the MID for 50,000 sets of 6 languages randomly selected.

The final matrices W_N and W_{80} obtained after feature selection are not complete: they contain respectively 4 and 35 unspecified values, which were filled automatically. Each matrix W (shorthand for W_N and W_{80}) offers a straightforward way to compare languages l_1 and l_2 using the Hamming distance[6] between their vectors $W(l_1)$ and $W(l_2)$, noted $d(l_1, l_2)$. To fill in the missing values, we have selected, for every language l_1 containing unspecified feature values ('?'), the corresponding value from its closest fully specified language l_2, that is, $l_2 = \underset{l_i \,\mid\, ? \notin W(l_i)}{\arg\min}\ d(l_1, l_i)$.

The W_N and W_{80} matrices only provide partial descriptions of languages, heavily biased towards parsing and ignoring other aspects (e.g., phonology). Nevertheless, it is tempting to compare how they relate languages that belong to the same typological genus. In order to do so, we have concentrated on three genus present in our set of 40 languages: Romance (6 languages), Germanic (6 languages) and Slavic (7 languages), and computed how close the vectors of these languages are. We define the *mean internal distance* (MID) of a language set $L = \{l_1, \ldots, l_n\}$, as the average of the distances of every pair of languages in L:

$$MID(L) = \frac{1}{n^2 - n} \sum_{\substack{(l_i,l_j) \in L\times L \\ i\neq j}} d(l_i, l_j)$$

We have computed the MID of each language genus, and compared it with the MID of randomly chosen sets of 6 languages (number of languages in the Romance and Germanic genus). The results in Table 1 clearly indicate that WALS vectors capture language genus similarities.

Others methods could have been used to assess whether the language descriptions that we have

---

[4] The description of the features of the WALS relevant for this paper can be found in appendix A.

[5] We have also considered the Complex Sentences family, but no feature exceeded the 80% threshold.

[6] The number of dimensions for which their values differ.

extracted from the WALS can measure language proximity. It could be interesting, for example, to reproduce the results of (Rabinovich et al., 2017) on reconstructing phylogenetic trees of language from the WALS features.

5 Parser

The parser used in our experiments is an arc-eager transition-based parser (Nivre, 2008), trained with a dynamic oracle (Goldberg and Nivre, 2012). The prediction of transitions is performed with a multilayer perceptron (MLP), as in Chen and Manning (2014). The MLP consists of an input layer, one hidden layer, and an output layer. Two sets of delexicalised features have been defined for the prediction: BASIC and EXTENDED. BASIC is a standard set composed of 9 POS features, 7 syntactic features, 32 morphological features, and a distance feature (the distance between the head and the dependent).[7] EXTENDED adds to BASIC new features that correspond to the WALS vectors W_N (6 features) and W_{80} (22 features), and/or the language ID of the sentence's language (1 feature). Each feature, including ID, is associated with a zero-initialized learnable embedding of size 3. The input layer of the MLP corresponds to the concatenation of the embeddings of the different features, with dimensions varying from 396 to 465, depending on the configuration (with or without language vectors W_N and W_{80}, or a language identifier ID). The output layer has 263 neurons, corresponding to the number of transitions that the parser can predict. The hidden layer has 1,000 units, the dropout rate used during training is equal to 0.4, the number of epochs is equal to 10, the activation function is a ReLU, the loss function is negative softmax log likelihood, and the learning algorithm is AMSgrad, using default parameters from Dynet (Neubig et al., 2017).[8]

At every step of the parsing process, the parser predicts an action to perform, which may yield the creation of a new dependency between two words of the sentence. The prediction of the actions is based on the values of the features fed to the MLP. In BASIC mode, these features describe different aspects of the head and the dependent, as well as their neighbourhood. For example, if the head is a verb and the dependent is a noun located before the verb, a subject dependency has high probability in languages that prefer subject-verb ordering (SV). In EXTENDED mode, the information of whether the language is SV is made explicit. The MLP has therefore the possibility to combine a *sentential configuration* (e.g., a noun before a verb) with a *language configuration* (e.g., the language is SV) when predicting an action. All languages that share a common feature in W will therefore be able to perform the same prediction for sentential configurations that are specific to this common feature (e.g., the noun preceding the verb and the language being SV).[9]

6 Experimental Settings

Corpora Our experiments were performed on the CoNLL 2017 shared task data (Zeman et al., 2017), on gold tokenisation and ignoring contractions (i.e., ranges). We evaluate our models individually on each of the 40 languages for which we have a $W(l)$ vector (section 4), using the original CoNLL 2017 shared task test sets. The test corpora for each language are simply the concatenation of all test treebanks for that language.

Training and development are performed on multilingual corpora (henceforth TRAIN-ML and DEV-ML) derived from the training and development treebanks of 37 UD languages.[10] The UD training and development corpora have different sizes for different languages, ranging from 529 words for Kazakh (kk) to 1,842,867 for Czech (cs). Thus, simply concatenating all corpora to constitute TRAIN-ML and DEV-ML would overrepresent certain languages and possibly bias the parser towards them. This is why we have decided to balance TRAIN-ML and DEV-ML across languages.

First, all available training and development corpora of the 37 languages have been concatenated. From this large corpus, we build two new intermediate corpora, PRE-TRAIN-ML and PRE-DEV-ML, with each sentence having 90% chances to belong to PRE-TRAIN-ML, and 10% chances

---

[7] Corpora, configuration files and WALS vectors available at: `http://pageperso.lis-lab.fr/~carlos.ramisch/?page=downloads/wals-ud-parse`

[8] Hyperparameters were tuned in preliminary experiments, in conditions similar to $\Sigma\ W_N$ (see Section 6).

[9] Our parser cannot predict non projective trees, systematically generating a wrong parse at test time. The average non projectivity rate of the test corpora is equal to 1%, with a standard deviation of 1% among the 40 languages. We ran some tests with pseudo projective tree transformation (Nivre and Nilsson, 2005), but it had a negligible impact on the results, so we have decided to keep the original projective algorithm.

[10] Three languages among our 40 target languages have no corresponding training nor development data (bxr, kmr, sme).

to belong to PRE-DEV-ML. Second, we build TRAIN-ML (respectively DEV-ML) by randomly selecting sentences from PRE-TRAIN-ML (resp. PRE-DEV-ML) until the number of tokens exceeds 20,000 (resp. 2,000 for DEV-ML) per language. At the end, we shuffle the selected sentences to obtain the final training and development corpora TRAIN-ML and DEV-ML. Using this procedure, the same sentence can appear several times in a corpus. Nonetheless, this method guarantees a balanced representation of every language in TRAIN-ML and DEV-ML.

Metrics The quality of the predicted trees is assessed with a standard measure for dependency parsing: labelled attachment score (LAS).[11], [12] We report LAS per language, as well as MACRO-LAS which is the macro-average of LAS on all languages that have a training set. This measure is therefore independent of the size of the test corpus of each language, and is not biased towards over-represented languages in the test sets.

Training Configurations Our experiments on several ⟨training corpus, language vector⟩ pairs are designated by the following codes:

***L*:** Monolingual corpus. The training corpus of a language l consists of the sentences of l in TRAIN-ML. Thirty-seven BASIC delexicalised parsers have been trained, one per language. This configuration corresponds to the standard one in parsing experiments: training and testing on the same language.

Σ: Multilingual corpus. A BASIC parser is trained on the whole TRAIN-ML corpus, with no indication of the inputs language. The parsing model is delexicalised, so the corpus contains only POS tags (gold), morphological features (gold) and syntactic relations (to be learned).[13]

Σ ID: Multilingual corpus + language ID. An EXTENDED parser is trained on the TRAIN-ML corpus using as extra feature the identifier of the language attached to each word.

Σ W_N, Σ W_{80}: Multilingual corpus + WALS. Two EXTENDED parsers are trained on the TRAIN-ML corpus, with W_N (resp. W_{80}) vectors derived from the WALS attached to each word.

7 Results

The detail of the LAS obtained for every language, as well as the macro-averaged LAS (MACRO) are displayed in Table 2. We comment below the results for L, and compare the results of meaningful pairs of experiments, summarised in Table 3.

L	Σ	Σ ID	Σ W_N	Σ W_{80}	Lang.
65.89	60.59	62.97	63.15	64.38	ar
78.59	74.32	76.43	76.26	77.47	bg S
77.18	72.76	74.11	73.03	76.27	ca R
68.92	68.01	68.91	68.72	69.61	cs S
73.62	67.38	70.25	70.19	70.25	da G
71.07	63.76	66.47	69.18	69.22	de G
77.11	71.26	72.72	73.29	75.84	el
70.05	66.02	69.3	69.91	70.19	en G
71.47	71.98	72.38	72.29	73.22	es R
66.98	63.76	66.89	65.75	67.79	et
63.26	55.76	59.54	60.22	59.39	eu
72.85	66.02	67.23	69.63	70	fa
60.97	56.29	58.86	57.37	59.28	fi
75.74	74.25	75.44	74.79	75.82	fr R
66.55	60.41	63.12	64.68	65.96	ga
70.21	63.03	65.69	66.09	67.45	he
78.91	73.86	75.81	75.77	74.45	hi
71.03	67.49	68.39	70	70.4	hr S
67.08	62.55	66.89	67.19	67.51	hu
68.64	58.38	63.99	62.61	64.57	id
81.44	76.45	78.03	76.97	79.83	it R
78.26	68.22	76.21	74.85	75.56	ja
47.68	37.17	40.03	38.07	39.66	ko
59.89	54.11	58.45	58.23	60.17	lv
62.56	57.21	57.15	58.13	58.59	nl G
74.59	73.19	75.13	73.51	75.93	no G
81.24	74.24	77.29	74.78	79.02	pl S
72	65.74	68.1	68.74	69.86	pt R
70.99	67.72	69.37	70.38	70.61	ro R
74.06	61.35	74.12	68.65	74.45	ru S
67.1	64.12	65.41	64.75	66.36	sl S
72.05	69.97	71.6	70.71	71.88	sv G
46.78	41.01	45.41	43.26	41.62	tr
71.6	69.4	71.96	69.77	72.81	uk S
74.35	69.15	72.07	70.71	70.76	ur
54.4	42.42	51.28	51.94	51.72	vi
59.83	45.46	58.94	53.42	54.87	zh
69.32	63.64	66.92	66.41	67.64	MACRO
-	33.32	31.99	30.37	28.49	bxr
-	40.34	38.14	41.41	44.04	kmr
-	47.34	46.5	47.63	42.38	sme

Table 2: LAS for each language and MACRO LAS, for the five configurations. Languages followed by a S belong to the Slavic genus, G belong to the German genus and R belong to the Romance genus.

***L*:** The results obtained in the L experiment show an important variation of performances for different languages. LAS ranges from 46.78 for

---

[11] For brevity, we omit UAS figures in our experiments, as UAS and LAS are tightly correlated ($r = 0.98$)

[12] Using the CoNLL shared task 2017 evaluation script.

[13] The decision to use gold POS tags and morphological features may seem unrealistic. This article is the first step of a process in which we intend to predict the POS tags and the morphological features in the same fashion.

X	Y	$\overline{X-Y}$	σ	min	max
L	Σ	5.68	3.32	-0.51 es	14.37 zh
$\Sigma\ W_{80}$	Σ	4.00	2.58	0.59 hi	13.10 ru
$\Sigma\ W_{80}$	$\Sigma\ W_N$	1.24	1.45	-1.64 tr	5.80 ru
Σ ID	Σ	3.27	3.00	-0.06 nl	13.48 zh
L	Σ ID	2.41	1.99	-0.91 es	7.65 ko
$\Sigma\ W_{80}$	Σ ID	0.73	1.54	-4.07 zh	3.12 el

Table 3: Differences between X and Y configurations: average ($\overline{X-Y}$), standard deviation (σ), minimum and maximum with corresponding languages.

Turkish to 81.44 for Italian. A detailed investigation for the reasons of such a variability is beyond the scope of this paper. Let us just mention a few hypotheses. Some are language specific, such as the balance between morphological and syntactic marking of linguistic constructions (i.e., morphologically rich languages are probably favoured in our setting, since the morphological analysis is given as input to the parser). Others are genre specific: the corpora for different languages pertain to different genres. Although delexicalisation neutralises some genre biases (some genres can feature a moderate lexical variability which can ease parsing) genres can also influence syntax, through sentence length (longer sentences are generally harder to parse), or the ratio of error-prone constructions, such as ambiguous prepositional phrases and coordination. Finally, annotation quality is heterogeneous across languages, potentially explaining the variability in LAS.

***L* vs Σ:** An expected drop in performances is observed when switching from *L* to Σ. The MACRO LAS loses 5.68 points. The main hypothesis to explain such a drop is the noise introduced when mixing different languages. This noise takes the form of contradictory information seen by the parser during training. For example, the sentential configuration associated to a subject dependency in SV and VS languages are very different, yet the parser is unaware of this distinction and will see contradictory examples. The variation of the LAS drop is different across languages. In the case of Spanish, switching from *L* to Σ even increases LAS (+0.51 points). We do not have a conclusive explanation for this result. The intuitive explanation is that Σ is a (noisy) language which on average is closer to Spanish than it is to Chinese (which performance drops by 14.37 points). This fact itself is the consequence that, on average, languages that compose Σ are closer to Spanish than they are to Chinese.

Σ vs $\Sigma\ W_{80}$: This is our first major result: when adding W_{80} to the parser, the MACRO LAS increases by 4 points when compared to Σ. LAS increases for all languages. There are two interpretations of this result. The optimistic one is that W_{80} helps decreasing the noise introduced by mixing languages in Σ by “explaining” some apparently contradictory information in the data through the use of linguistic features encoded in the WALS. The pessimistic interpretation is that the WALS vectors are merely an arbitrary encoding of the languages. In this case, the parser’s MLP would be associating sentential configurations to specific languages, thus learning different models for different languages. Figuring out what the model is actually learning is not an easy task. We propose in section 8 some clues to answer this question. Moreover, there is not a clear tendency to increase or decrease when using the WALS vector in the case of the 3 languages without training data. More experiments are required to study the performances when the language is not in the training corpus.

$\Sigma\ W_N$ vs $\Sigma\ W_{80}$: When added to Σ, vectors W_N and W_{80} do not have the same impact on the performances. Adding W_{80} to Σ yields an increase of 4 points while adding W_N increases the performance by 2.77 points only. The parser is therefore able to take advantage from a richer description of languages when learning the model. This result indicates that the disappointing parsing results reported by Ammar et al. (2016), who adopted the W_N vector, are probably due to the fact that the features extracted from the WALS were not rich enough to explain differences between languages that are important for a parser.

Σ vs Σ ID: Adding the ID vector to Σ yields an improvement of 3.28 MACRO points. This increase was expected since, in this setting, sentential configurations are associated to a language ID, which helps decreasing the noise in the data.

***L* vs Σ ID:** One could expect that Σ ID would reach the result obtained by *L* since in both configurations the same amount of data is available and languages are unambiguously identified. This is not the case: the performance of Σ ID is 2.41 points behind *L*. The difference in performances is due to the MLP architecture (in particular the size of the hidden layer), which is the same for Σ ID and for each of the *L* models. Each language is described with more parameters in an L model than it is in the Σ ID model.

Σ ID vs Σ W_{80}: This is our second major result: adding W_{80} to Σ yields better results than adding ID to Σ. This result indicates that it is more interesting, in our setting, to describe a language as a vector of typological features, allowing to identify features that are common to several languages, than describing a language by an arbitrary code. As mentioned above, such a conclusion is valid for models of a fixed size only, which is the case here. It could be the case that, when increasing the number of parameters of the models, Σ ID gets better results than Σ W_{80}.

We do not report here a series of experiences combining ID and W_{80}. We observed a slight improvement (MACRO=67.86) when adding ID in the input of the parser. This effect indicates that the information contained in ID and W_{80} vectors are complementary and the parser has the opportunity to rely on both of them. Figuring out exactly how the parser uses this information is a complex issue that we address in the following section.

8 How does the parser use W?

As already conjectured, one hypothesis for explaining the behaviour of the parser in the presence of W is that it uses the additional features to identify a language, not to better generalise on the syntactic phenomena that the features address. Table 4 shows the accuracy of a logistic regression classifier trained to predict the language ID based on either the input features of the parser's MLP, or on the activations after the hidden layer, with Σ, Σ W_N and Σ W_{80}. The table shows that indeed, WALS features, especially W_{80}, greatly improve the capability of the language classifier, suggesting that the parser can use language identity in its predictions. The fact that this information is still available just before the decision layer means that it can be used for predicting parsing actions.

Another interesting analysis consists in comparing the distribution of activations for two languages. In the following, the activations are measured at the hidden layer before the ReLU nonlinearity, and are assumed to follow normal distributions at the neuron level. We compute the Jensen-Shannon Divergence (JSD) between the activations of a given neuron for a pair of languages. Table 5 shows the mean, maximum and minimum neuron-level JSD between cherry-picked language pairs. We selected three language pairs with increasing distance. Dutch and German (nl-de) belong to the same typological genus and have identical W_{80} vectors. Portuguese and French (pt-fr) also belong to the same genus but their vectors differ in six features (e.g. 101A pronominal subject, 143E postverbal negation). On the other extreme, Russian Buriat and Irish (brx-ga) have very different W_{80} vectors, with only two shared values out of 22.

Configuration	Features	Accuracy
Σ	input	0.432
Σ W_N	input	0.678
Σ W_{80}	input	0.954
Σ	hidden	0.436
Σ W_N	hidden	0.682
Σ W_{80}	hidden	0.956

Table 4: Language identification accuracy for a logistic regression classifier trained on the activations after the hidden layer, or at the input. The classifier is trained on the development set, results are reported on the test set.

L1	L2	Model	Mean	Max	Min
nl	de	Σ	0.860	1.027	0.798
		Σ W_{80}	0.854	0.940	0.793
pt	fr	Σ	0.878	1.335	0.794
		Σ W_{80}	0.912	3.550	0.700
bxr	ga	Σ	0.890	1.600	0.782
		Σ W_{80}	1.160	4.888	0.757

Table 5: Neuron-level JSD statistics between activations at the hidden layer of the parser models for selected pairs of languages.

For nl-de, the average difference between the activation distributions in Σ W_{80} (0.854) is lower than in Σ (0.86), suggesting that W_{80} helps leveraging the similarity between those languages, which is also confirmed by an increase in LAS (Table 2). For pt-fr, however, the addition of W_{80} results in an increase in the average distance between the activation distributions (0.912) when compared to Σ (0.878). Analogously, this difference also increases by a larger margin (from 0.89 to 1.16) for the most distant pair bxr-ga. Overall, these observations indicate that W_{80} reinforces parameter sharing between similar languages and increases contrast between dissimilar ones. As an example, Figure 1 shows that the distributions for the neuron with highest JSD are very similar for nl-de while they are different for bxr-ga.

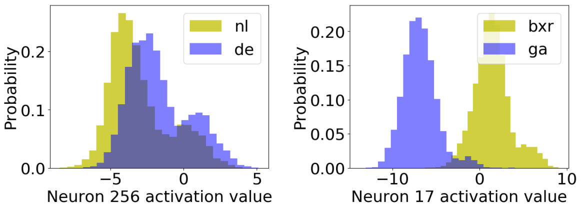


Figure 1: Activation distributions for the neuron with highest JSD on Σ for (nl, de) and (bxr, ga) pairs.

9 Conclusions and Future Work

This paper has studied how high-level typological language descriptions coming from the WALS can guide a multilingual parser to learn cross-language generalisations. Two interpretations of what the parser is doing in the light of such information have been opposed. In the first (optimistic) one, the parser uses the high-level descriptions to cluster coherent observable patterns across languages. In the second (pessimistic) one, the parser uses the high-level descriptions given as input to figure out the identity of the language and uses this ID to trigger parts of the model that are language specific. Our results and parsing model analyses hint that, although it is difficult to draw definitive conclusions, the model indeed uses information in the WALS vectors as language identifiers, but some extra gain is observed, favouring the cross-lingual sharing hypothesis.

As future work, we plan to study the influence of typological features on each dependency type. Whereas a delexicalised parser offers a simple experimental setup, it impacts parsing performance. Thus, we would like to use multilingual word embeddings to make lexical information accessible to the parser, making it more realistic. The results in section 8 suggest that the parser struggles between two behaviours. One way to intervene would be to penalise the parser when it correctly identifies the language, using adversarial learning (Ganin et al., 2016). Our experiments on the three languages with no training corpus are not conclusive on the usefulness of the WALS vector in zero-shot setting, and we plan to make more tests in this setting.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Sgaard. 2016. Multilingual Projection for Parsing Truly Low-Resource Languages. *Transactions of the Association for Computational Linguistics*, 4(0):301–312.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many Languages, One Parser. *arXiv:1602.01595 [cs]*. ArXiv: 1602.01595.

Johannes Bjerva, Robert Östling, Maria Han Veiga, Jörg Tiedemann, and Isabelle Augenstein. 2019. What do language representations really represent? *arXiv preprint arXiv:1901.02646*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

Noam Chomsky. 1995. *The Minimalist Program*. Current studies in linguistics series. MIT Press, Cambridge, MA, USA.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976.

Joseph H. Greenberg. 1963. *Universals of Human Language*. MIT Press, Cambridge, MA, USA.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual Dependency Parsing Based on Distributed Representations. In *ACL (1)*, pages 1234–1244.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California. Association for Computational Linguistics.

Teresa Lynn, Jennifer Foster, Mark Dras, and Lamia Tounsi. 2014. Cross-lingual Transfer Parsing for Low-Resourced Languages: An Irish Case Study.

In *Proceedings of the First Celtic Language Technology Workshop*, pages 41–49, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348, Baltimore, Maryland. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637, Jeju Island, Korea. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106. Association for Computational Linguistics.

Robert Östling and Jrg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649, Valencia, Spain. Association for Computational Linguistics.

Edoardo Maria Ponti, Roi Reichart, Anna Korhonen, and Ivan Vulić. 2018. Isomorphic Transfer of Syntactic Structures in Cross-Lingual NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1531–1542, Melbourne, Australia. Association for Computational Linguistics.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 530–540.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

David Vilares, Carlos Gmez-Rodrguez, and Miguel A. Alonso. 2015. One model, two languages: training bilingual parsers with harmonized treebanks. *arXiv:1507.08449 [cs]*. ArXiv: 1507.08449.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, and Martin Potthast. 2017. CoNLL 2017 shared task: multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Vclava Kettnerov, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missil, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Hctor Martnez Alonso, ar ltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadov, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2018. CoNLL 2018 Shared Task: Multilingual

Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-Language Parser Adaptation between Related Languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*. Association for Computational Linguistics.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867, Lisbon, Portugal. Association for Computational Linguistics.

A Appendix: description of the WALS features used in our work

1. **81A**: Order of Subject, Object and Verb (*SOV; SVO; VSO; VOS; OVS; OSV; No dominant order*)
2. **82A**: Order of Subject and Verb (*SV; VS; No dominant order*)
3. **83A**: Order of Object and Verb (*OV; VO; No dominant order*)
4. **85A**: Order of Adposition and Noun Phrase (*Postpositions; Prepositions; Inpositions; No dominant order; No adpositions*)
5. **86A**: Order of Genitive and Noun (*Genitive-Noun; Noun-Genitive; No dominant order*)
6. **87A**: Order of Adjective and Noun (*Adjective-Noun; Noun-Adjective; No dominant order; Only internally-headed relative clauses*)
7. **88A**: Order of Demonstrative and Noun (*Demonstrative-Noun; Noun-Demonstrative; Demonstrative prefix; Demonstrative suffix; Demonstrative before and after Noun; Mixed*)
8. **89A**: Order of Numeral and Noun (*NumN; NNum; Both orders of numeral and noun with neither order dominant; Numeral only modifies verb*)
9. **90A**: Order of Relative Clause and Noun (*NRel; RelN; Internally-headed relative clause; Correlative relative clause; Adjoined relative clause; Double-headed relative clause; Mixed types of relative clause with none dominant*)
10. **92A**: Position of Polar Question Particles (*Initial; Final; Second position; Other position; In either of two positions; No Question particle*)
11. **95A**: Relationship between the Order of Object and Verb and the Order of Adposition and Noun Phrase (*OV & Postpositions; OV & Prepositions; VO & Postpositions; VO & Prepositions; Other*)
12. **96A**: Relationship between the Order of Object and Verb and the Order of Relative Clause and Noun (*OV & RelN; OV & NRel; VO & RelN; VO & NRel; Other*)
13. **97A**: Relationship between the Order of Object and Verb and the Order of Adjective and Noun (*OV & AdjN; OV & NAdj; VO & AdjN; VO & NAdj; Other*)
14. **101A**: Expression of Pronominal Subjects (*Pronominal subjects are expressed by pronouns in subject position that are normally if not obligatorily present; Pronominal subjects are expressed by affixes on verbs; Pronominal subjects are expressed by clitics with variable host; Pronominal subjects are expressed by subject pronouns that occur in a different syntactic position from full noun phrase subjects; Pronominal subjects are expressed only by pronouns in subject position, but these pronouns are often left out; More than one of the above types with none dominant*)
15. **112A**: Negative Morphemes (*Negative affix; Negative particle; Negative auxiliary verb; Negative word, unclear if verb or particle; Variation between negative word and affix; Double negation*)
16. **116A**: Polar Question (*Question particle; Interrogative verb morphology; Question particle and interrogative verb morphology; Interrogative word order; Absence of declarative morphemes; Interrogative intonation only; No interrogative-declarative distinction*)

17. **143A**: Order of Negative Morpheme and Verb (*NegV; VNeg; [Neg-V]; [V-Neg]; Negative Tone; Type 1 / Type 2; Type 1 / Type 3; Type 1 / Type 4; Type 2 / Type 3; Type 2 / Type 4; Type 3 / Type 4; Type 3 / Negative Infix; Optional Single Negation; Obligatory Double Negation; Optional Double Negation; Optional Triple Negation with Obligatory Double Negation; Optional Triple Negation with Optional Double Negation*)

18. **143E**: Preverbal Negative Morphemes (*Preverbal negative word; Negative prefix; Both preverbal negative word and negative prefix; No preverbal negative morpheme*)

19. **143F**: Postverbal Negative Morphemes (*Postverbal negative word; Negative suffix; Both postverbal negative word and negative suffix; No postverbal negative morpheme*)

20. **143G**: Minor morphological means of signaling negation (*Negative tone; Negative infix; Negative stem change; No negative tone, infix or stem change*)

21. **144A**: Position of Negative Word With Respect to Subject, Object, and Verb (*NegSVO; SNegVO; SVNegO; SVONeg; NegSOV; SNegOV; SONegV; SOVNeg; NegVSO; VSNegO; VSONeg; NegVOS; ONegVS; ONegVS; OSVNeg; More than one position for negative morpheme, with none dominant; Optional single negation; Obligatory double negation; Optional double negation; Morphological negation only (but not double negation); Other language*)

Recommendations for Datasets for Source Code Summarization

Alex LeClair, Collin McMillan
Department of Computer Science and Engineering
University of Notre Dame
{aleclair, cmc}@nd.edu

Abstract

Source Code Summarization is the task of writing short, natural language descriptions of source code. The main use for these descriptions is in software documentation e.g. the one-sentence Java method descriptions in JavaDocs. Code summarization is rapidly becoming a popular research problem, but progress is restrained due to a lack of suitable datasets. In addition, a lack of community standards for creating datasets leads to confusing and unreproducible research results – we observe swings in performance of more than 33% due only to changes in dataset design. In this paper, we make recommendations for these standards from experimental results. We release a dataset based on prior work of over 2.1m pairs of Java methods and one sentence method descriptions from over 28k Java projects. We describe the dataset and point out key differences from natural language data, to guide and support future researchers.

1 Introduction

Source Code Summarization is the task of writing short, natural language descriptions of source code (Eddy et al., 2013). The most common use for these descriptions is in software documentation, such as the summaries of Java methods in JavaDocs (Kramer, 1999). Automatic generation of code summaries is a rapidly-expanding research area at the crossroads of Computational Linguistics and Software Engineering, as a growing tally of new workshops and NSF-sponsored meetings have recognized (Cohen and Devanbu, 2018; Quirk, 2015). The reason, in a nutshell, is that the vast majority of code summarization techniques are adaptations of techniques originally designed to solve NLP problems.

A major barrier to ongoing research is a lack of standardized datasets. In many NLP tasks such as Machine Translation there are large, curated datasets (e.g. Europarl (Koehn, 2018)) used by several research groups. The benefit of these standardized datasets is twofold: First, scientists are able to evaluate new techniques using the same test conditions as older techniques. And second, the datasets tend to conform to community customs of best practice, which avoids errors during evaluation. These benefits are generally not yet available to code summarization researchers; while large, public code repositories do exist, most research projects must parse and process these repositories on their own, leading to significant differences on one project to another. The result is that research progress is slowed as reproducibilty of earlier results is difficult.

Inevitably, differences in dataset creation also occur that can mislead researchers and over or understate the performance of some techniques. For example, a recent source code summarization paper reports achieving 25 BLEU when generating English descriptions of Java methods with an existing technique (Gu et al., 2018), which is 5 points higher than the original paper reports (Iyer et al., 2016). The paper also reports 35+ BLEU for a vanilla seq2seq NMT model, which is 16 points higher than what we are able to replicate. While it is not our intent to single out any one paper, we do wish to call attention to a problem in the research area generally: a lack of standard datasets leads to results that are difficult to interpret and replicate.

In this paper, we propose a set of guidelines for building datasets for source code summarization techniques. We support our guidelines with related literature or experimentation where strong literary consensus is not available. We also compute several metrics related to word usage to guide future researchers who use the dataset. We have made a dataset of over 2.1m Java methods and summaries from over 28k Java projects available via an online appendix (URL in Section 6).

Proceedings of NAACL-HLT 2019, pages 3931–3937
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Related Work

Related work to this paper consists of approaches for source code summarization. As with many research areas, data-driven AI-based approaches have superseded heuristic/template-based techniques, though overall the field is quite new. Work by Haiduc *et al.* (Haiduc et al., 2010a,b) in 2010 coined the term "source code summarization", and several heuristic/template-based techniques followed including work by Sridhara *et al.* (Sridhara et al., 2010, 2011), McBurney *et al.* (McBurney and McMillan, 2016), and Rodeghero *et al.* (Rodeghero et al., 2015).

More recent techniques are data-driven, though the overall size of the field is small. Literature includes work by Hu *et al.* (Hu et al., 2018a,b) and Iyer *et al.* (Iyer et al., 2016). Projects targeting problems similar to code summarization have been published widely, including on commit message generation (Jiang et al., 2017; Loyola et al., 2017), method name generation (Allamanis et al., 2016), pseudocode generation (Oda et al., 2015), and code search (Gu et al., 2018). Nazar *et al.* (Nazar et al., 2016) provide a survey.

Of note is that no standard datasets for code summarization have yet been published. Each of the above papers takes an ad hoc approach, in which the authors download large repositories of code and apply their own preprocessing. There are few standard practices, leading to major differences in the reported results in different papers, as discussed in the previous section. For example, the works by LeClair *et al.* (LeClair and McMillan, 2019) and Hu *et al.* (Hu et al., 2018a) both modify the CODENN model from Iyer *et al.* (Iyer et al., 2016) to work on Java methods and comments. LeClair *et al.* and Hu *et al.* report very disparate results: A BLEU-4 score of 6.3 for CODENN on one dataset, and 25.3 on another, even though both datasets were generated from Java source code repositories.

These disparate results happen for a variety of reasons, such as a difference in data set sizes and tokenization schemes. LeClair *et al.* use a data set of 2.1 million Java method-comment pairs while Hu *et al.* use a total of 69,708. Hu *et al.* also replace out of vocabulary (OOV) tokens in the comments with <UNK> in the training, validation, and testing sets, while LeClair *et al.* remove OOV tokens from the training set only.

3 Dataset Preparations

The dataset we use in this paper is based on the dataset provided by LeClair *et al.* (LeClair and McMillan, 2019) in a pre-release. We used this dataset because it is both the largest and most recent in source code summarization. That dataset has its origins in the Sourcerer project by Lopes *et al.* (Lopes et al., 2010), which includes over 51 million Java methods. LeClair *et al.* provided the dataset after minimal initial processing that filtered for Java methods with JavaDoc comments in English, and removed methods over 100 words long and comments $>$13 and $<$3 words. The result is a dataset of 2.1m Java methods and associated comments. LeClair *et al.* do additional processing, but do not quantify the effects of their decisions – this is a problem because other researchers would not know which of the decisions to follow. We explore the following research questions to help provide guidelines and justifications for our design decisions in creating the dataset.

3.1 Research Questions

Our research objective and contribution in this paper is to quantify the effect of key dataset processing configurations, with the aim to make recommendations on which configurations should be used. We ask the following Research Questions:

RQ$_1$ What is the effect of splitting by method versus splitting by project?

RQ$_2$ What is the effect of removing automatically generated Java methods?

The scope of the dataset in this paper is source code summarization of Java methods – the dataset contains pairs of Java methods and JavaDoc descriptions of those methods. However, we believe these RQs will provide guidance for similar datasets e.g. C/C++ functions and descriptions, or other units of granularity e.g. code snippets instead of methods/functions.

The rationale behind RQ$_1$ is that many papers split the dataset into training, validation, and test sets at the unit of granularity under study. For example, dividing all Java methods in the dataset into 80% in training, 10% in validation, and 10% in testing. However, this results in a situation where it is possible for code from one project to be in both the testing set and the training set. It is possible that similar vocabulary and code patterns

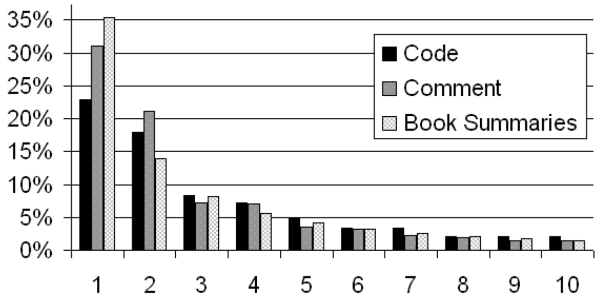


Figure 1: Word count histogram for code, comment, and the book summaries. About 22% of words occur one time across all Java methods, versus 35% in the book summaries.

are used in methods from the same project, and even worse, it is possible that overloaded methods appear in both the training and test sets. However, this possibility is theoretical and a negative effect has never been shown. In contrast, we split by project: randomly divide the Java projects into training/validation/test groups, then place all methods from e.g. test projects into the test set.

The rationale behind RQ_2 is that automatically generated code is common in many Java projects (Shimonaka et al., 2016), and that it is possible that very similar code is generated for projects in the training set and the testing/validation sets. Shimonaka *et al.* (Shimonaka et al., 2016) point out that the typical approach for identifying auto-generated code is a simple case-insensitive text search for the phrase "generated by" in the comments of the Java files. LeClair *et al.* (LeClair and McMillan, 2019) report that this search turns out to be quite aggressive, catching nearly all auto-generated code in the repository. However, as with RQ_1, the effect of this filter is theoretical and has not been measured in practice.

3.2 Methodology

Our methodology for answering RQ_1 is to compare the results of a standard NMT algorithm with the dataset split by project, to the results of the same algorithm on the same dataset, except with the dataset split by function. But because random splits could be "lucky", we created four random datasets split by project, and four split by function, seen in Table 2. We then use an off-the-shelf, standard NMT technique called `attendgru` provided pre-release by LeClair *et al.* (LeClair and McMillan, 2019) and used as a baseline approach in their recent paper. The technique is just an attentional encoder/decoder based on single-layer GRUs, and represents a strong NMT baseline used by many papers. We train `attendgru` with each of the four training sets, find the best-performing model using the validation set associated with that training set (out of 10 maximum epochs), and then obtain test performance for that model. We report the average of the results over the four random splits. Note that we used the same configuration for `attendgru` as LeClair *et al.* report, except that we reduced the output vocabulary to 10k to reduce model size.

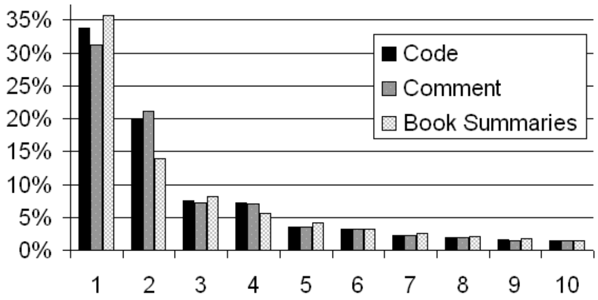


Figure 2: Histogram of word occurrences per document. Approximately 34% of words occur in only one Java method, 20% occur in two methods, etc.

Our process for RQ_2 is similar. We created four random split-by-project sets in which automatically generated code was *not* removed. Then we compared them to the four random split-by-project sets we created for RQ_1 (in which auto-generated code was removed).

3.3 Dataset Characteristics

We make three observations about the dataset that, in our view, are likely to affect how researchers design source code summarization algorithms. First, as depicted in Figure 1, words appear to be used more often in code as compared to natural language – there are fewer words used only one or two times, and in general more used 3+ times. At the same time (Figure 2), the pattern for word occurrences per document appears similar, implying that even though words in code are repeated, they are repeated often in the same method and not across methods. Even though this may suggest that the occurrence of unique words in source code is isolated enough to have little affect on BLEU score, we show in Section 4 that this word overlap causes BLEU score inflation when you split by function. This is important because the typical MT use case assumes that a "dictionary" can be created (e.g., via attention) to map words in a source to words in a target language. An algorithm applied to code summarization needs to tolerate multiple occurrences of the same words. To compare the source code, comments, and natural language datasets we tokenized our data by removing all special characters, lower casing, and for source code – splitting camel case into separate tokens.

SplittingStrategy	Set1	Set2	Set3	Set4
Split by project	17.81	16.73	17.11	17.99
Split by function	20.97	23.74	23.67	23.68
Auto-generated code included	19.11	19.09	18.04	15.66

Table 1: Average BLEU Scores from 15 epochs for each of the four sets.

A related observation is that Java methods tend to be much longer than comments (Figure 3 areas (c) and (d)). Typically, code summarization tools take inspiration from NMT algorithms designed for cases of similar encoder/decoder sequence length. Many algorithms such as recurrent networks are sensitive to sequence length, and may not be optimal off-the-shelf.

A third observation is that the words in methods and comments tend to overlap, but in fact a vast majority of words are different (70% of words in code summary comments do not occur in the code method, see Figure 3 area (b)). This situation makes the code summarization problem quite difficult because the words in the comments represent high level concepts, while the words in the source code represent low level implementation details – a situation known as the "concept assignment problem" (Biggerstaff et al., 1993). A code summarization algorithm cannot only learn a word dictionary as it might in a typical NMT setting, or select summarizing words from the method for a summary as a natural language summarization tool might. A code summarization algorithm must learn to identify concepts from code details, and assign high level terms to those concepts.

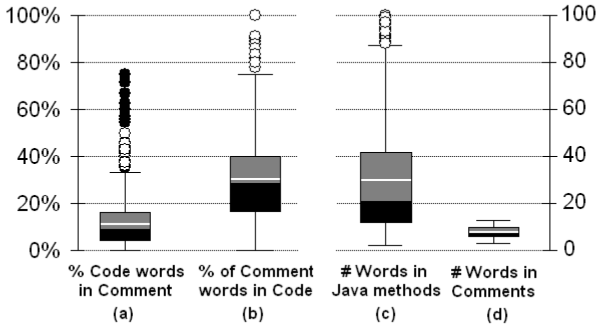


Figure 3: Overlap of words between methods and comments (areas a and b). Over 30% of words in comments, on average also occur in the method it describes. About 11% of words in code, on average, also occur in the comment describing it. Also, word length of methods and comments (areas c and d). Methods average around 30 words, versus 10 for comments.

4 Experimental Results & Conclusion

In this section, we answer our Research Questions and provide supporting evidence and rational.

4.1 RQ$_1$: Splitting Strategy

We observe a large "false" boost in BLEU score when split by function instead of split by project (see Figure 4). We consider this boost false because it involves placing functions from projects in the test set into the training set – an unrealistic scenario. An average of four runs when split by project was 17.41 BLEU, a result relatively consistent across the splits (maximum was 18.28 BLEU, minimum 16.10). In contrast, when split by function, the average BLEU score was 23.02, and increase of nearly one third as seen in Table 1. Our conclusion is that splitting by function is to be avoided during dataset creation for source code summarization. Beyond this narrow answer to the RQ, in general, any leakage of information from test set projects into the training or validation sets ought to be strongly avoided, even if the unit of granularity is smaller than a whole project. We reiterate from Section 1 that this is not a theoretical problem: many papers published using data-driven techniques for code summarization and other research problems split their data at the level of granularity under study.

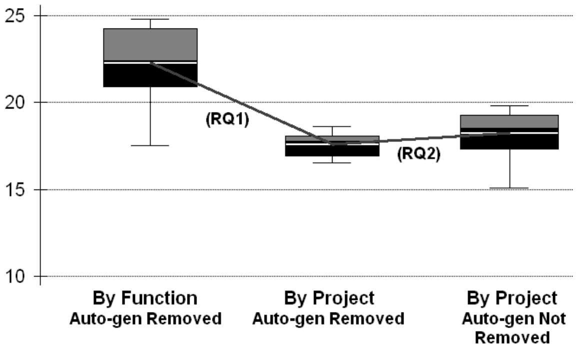


Figure 4: Boxplots of BLEU scores from attendgru for four runs under configurations for RQ$_1$ and RQ$_2$.

	BP Set1	BP Set2	BP Set3	BP Set4	BF All Sets
Training Set	1,935,860	1,950,026	1,942,291	1,933,677	1,943,723
Validation Set	105,693	100,920	104,837	105,997	107,984
Testing Set	107,568	98,175	101,993	109,447	107,984

Table 2: Number of method-comment pairs in the train, validation, test sets used in each random split set when split by project (BP) and by function (BF).

4.2 RQ$_2$: Removing Autogen. Code

We also found a boost in BLEU score when not removing automatically generated code, though the difference was less than observed for RQ$_1$. The baseline performance increased to 18 BLEU when not removing auto-generated code, and it varied much more depending on the split (some projects have much more auto-generated code than others). Our recommendation is that, in general, reasonable precautions should be implemented to remove auto-generated code from the dataset because we do find evidence that auto-generated code can affect the results of experiments.

5 Discussion

This paper provides benefits to researchers in the field of automatic source code summarization in two areas. First, we provide insight into the effects of splitting a Java method and comment dataset by project or by function, and how these different splitting methods effect the task of source code summarization. Second, we provide a dataset of 2.1m pairs of Java methods and one sentence method descriptions in a cleaned and tokenized format (discussed in 6) as well as a training, validation, testing split.

Note however that there may be cases where researchers wish to adapt our recommendations for a specific context. For example, when generating comments in an IDE. The problem of code summarization in an IDE is slightly different than what we have presented, and would benefit from including code-comment pairs from the same project. IDEs have the advantage of access to a programmer's source code and edit history in real time – they do not rely on a repository collected post-hoc. Moreno *et al.* (Moreno et al., 2013) take advantage of this information to generate Java class summaries in an eclipse plugin – their tool uses both the class and project level information from completed projects to generate these summaries, while not using any information from outside sources.

However, even in this case, care must be taken to avoid unrealistic scenarios, such as ensuring that the training set consists only of code older than the code in the test set. For example, consider a programmer at revision 75 of his or her project who requests automatically generated comments from the IDE, then goes on to write a total of 100 revisions for the project. An experiment simulating this situation should only use revisions 1-74 as training data – revisions 76+ are "in the future" from the perspective of the real world situation.

6 Downloadable Dataset

In our online appendix we have made three downloadable sets available. The first is our SQL database, generated using the tool from McMillan *et al.* (McMillan et al., 2011), that contains the file name, method comment, and start/end lines for each method, we call this dataset our "Raw Dataset". We also provide a link to the Sourcerer dataset (Linstead et al., 2009) which is used as a base for the dataset in LeClair *et al.* (LeClair and McMillan, 2019). In addition to the Raw Dataset, we also provide a "Filtered Dataset" that consists of a set of 2.1m method comment pairs. In the Filtered Dataset we removed auto-generated source code files, as well all method's that do not have an associated comment. No preprocessing was applied to the source code and comment strings in the Filtered Dataset. The third downloadable set we supply is the "Tokenized Dataset". In the Tokenized Dataset, we processed the source code and comments from the Filtered Dataset identically to the tokenization scheme described in Section 5 of (LeClair and McMillan, 2019). This set also provides a training, validation, and test set as well as a script to easily reshuffle these sets.

The URL for download is:
`http://leclair.tech/data/funcom`

Acknowledgments

This work is supported in part by the NSF CCF-1452959, CCF-1717607, and CNS-1510329 grants. Any opinions, findings, and conclusions expressed herein are the authors and do not necessarily reflect those of the sponsors

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*, pages 2091–2100.

Ted J Biggerstaff, Bharat G Mitbander, and Dallas Webster. 1993. The concept assignment problem in program understanding. In *Proceedings of the 15th international conference on Software Engineering*, pages 482–498. IEEE Computer Society Press.

William Cohen and Prem Devanbu. 2018. Workshop on nlp for software engineering.

Brian P Eddy, Jeffrey A Robinson, Nicholas A Kraft, and Jeffrey C Carver. 2013. Evaluating source code summarization techniques: Replication and expansion. In *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, pages 13–22. IEEE.

Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2018. Deep code search. In *Proceedings of the 40th International Conference on Software Engineering*, pages 933–944. ACM.

Sonia Haiduc, Jairo Aponte, and Andrian Marcus. 2010a. Supporting program comprehension with source code summarization. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 223–226. ACM.

Sonia Haiduc, Jairo Aponte, Laura Moreno, and Andrian Marcus. 2010b. On the use of automated text summarization techniques for summarizing source code. In *2010 17th Working Conference on Reverse Engineering*, pages 35–44. IEEE.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018a. Deep code comment generation. In *Proceedings of the 26th Conference on Program Comprehension*, pages 200–210. ACM.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018b. Summarizing source code with transferred api knowledge. In *IJCAI*, pages 2269–2275.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2073–2083.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 135–146. IEEE Press.

Philipp Koehn. 2018. European parliament proceedings parallel corpus 1996-2011. `http://www.statmt.org/europarl/`. Accessed October 25, 2018.

Douglas Kramer. 1999. Api documentation from source code comments: a case study of javadoc. In *Proceedings of the 17th annual international conference on Computer documentation*, pages 147–153. ACM.

Alexander LeClair and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. *International Conference on Software Engineering (ICSE) (*under review*)*.

Erik Linstead, Sushil Bajracharya, Trung Ngo, Paul Rigor, Cristina Lopes, and Pierre Baldi. 2009. Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18:300–336. 10.1007/s10618-008-0118-x.

C. Lopes, S. Bajracharya, J. Ossher, and P. Baldi. 2010. UCI source code data sets.

Pablo Loyola, Edison Marrese-Taylor, and Yutaka Matsuo. 2017. A neural architecture for generating natural language descriptions from source code changes. *arXiv preprint arXiv:1704.04856*.

Paul W McBurney and Collin McMillan. 2016. Automatic source code summarization of context for java methods. *IEEE Transactions on Software Engineering*, 42(2):103–119.

Collin McMillan, Mark Grechanik, Denys Poshyvanyk, Qing Xie, and Chen Fu. 2011. Portfolio: finding relevant functions and their usage. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 111–120. ACM.

L. Moreno, A. Marcus, L. Pollock, and K. Vijay-Shanker. 2013. Jsummarizer: An automatic generator of natural language summaries for java classes. In *2013 21st International Conference on Program Comprehension (ICPC)*, pages 230–232.

Najam Nazar, Yan Hu, and He Jiang. 2016. Summarizing software artifacts: A literature review. *Journal of Computer Science and Technology*, 31(5):883–909.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 574–584. IEEE.

Chris Quirk. 2015. Nsf interdisciplinary workshop on statistical nlp and software engineering. `http://www.languageandcode.org/nlse2015/`. Accessed October 25, 2016.

Paige Rodeghero, Cheng Liu, Paul W McBurney, and Collin McMillan. 2015. An eye-tracking study of java programmers and application to source code summarization. *IEEE Transactions on Software Engineering*, 41(11):1038–1054.

Kento Shimonaka, Soichi Sumi, Yoshiki Higo, and Shinji Kusumoto. 2016. Identifying auto-generated code by using machine learning techniques. In *Empirical Software Engineering in Practice (IWESEP), 2016 7th International Workshop on*, pages 18–23. IEEE.

Giriprasad Sridhara, Emily Hill, Divya Muppaneni, Lori Pollock, and K Vijay-Shanker. 2010. Towards automatically generating summary comments for java methods. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 43–52. ACM.

Giriprasad Sridhara, Lori Pollock, and K Vijay-Shanker. 2011. Automatically detecting and describing high level actions within methods. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 101–110. ACM.

Question Answering as an Automatic Evaluation Metric for News Article Summarization

Matan Eyal[1, 2], **Tal Baumel**[1, 3], **Michael Elhadad**[1]
[1]Dept. Computer Science, Ben Gurion University
[2]IBM Research, Israel, [3]Microsoft
{mataney, elhadad}@cs.bgu.ac.il, tabaumel@microsoft.com

Abstract

Recent work in the field of automatic summarization and headline generation focuses on maximizing ROUGE scores for various news datasets. We present an alternative, extrinsic, evaluation metric for this task, ***A****nswering* ***P****erformance for* ***E****valuation of* ***S****ummaries*. APES utilizes recent progress in the field of reading-comprehension to quantify the ability of a summary to answer a set of manually created questions regarding central entities in the source article. We first analyze the strength of this metric by comparing it to known manual evaluation metrics. We then present an end-to-end neural abstractive model that maximizes APES, while increasing ROUGE scores to competitive results.

1 Introduction

The task of automatic text summarization aims to produce a concise version of a source document while preserving its central information. Current summarization models are divided into two approaches, *extractive* and *abstractive*. In extractive summarization, summaries are created by selecting a collection of key sentences from the source document (*e.g.*, Nallapati et al. (2017); Narayan et al. (2018)). Abstractive summarization, on the other hand, aims to rephrase and compress the input text in order to create the summary. Progress in sequence-to-sequence models (Sutskever et al., 2014) has led to recent success in abstractive summarization models. Current models (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017; Celikyilmaz et al., 2018) made various adjustments to sequence-to-sequence models to gain improvements in ROUGE (Lin, 2004) scores.

ROUGE has achieved its status as the most common method for summaries evaluation by showing high correlation to manual evaluation methods, *e.g.*, the Pyramid method (Nenkova et al., 2007). Tasks like TAC AESOP (Owczarzak and Dang, 2011) used ROUGE as a strong baseline and confirmed the correlation of ROUGE with manual evaluation.

See et al. (2017)'s Summary: bolton will offer new contracts to emile heskey, 37, eidur gudjohnsen, 36, and adam bogdan, 27. heskey and gudjohnsen joined on short-term deals in december. eidur gudjohnsen has scored five times in the championship .
APES score: 0.33

Baseline Model Summary (Encoder / Decoder / Attention / Copy / Coverage): bolton will offer new contracts to emile heskey, 37, eidur gudjohnsen, 36, and goalkeeper adam bogdan, 27. heskey and gudjohnsen joined on short-term deals in december, and have helped neil lennon 's side steer clear of relegation. eidur gudjohnsen has scored five times in the championship, as well as once in the cup this season .
APES score: 0.33

Our Model (APES optimization): bolton will offer new contracts to emile heskey, 37, eidur gudjohnsen, 36, and goalkeeper adam bogdan, 27. heskey joined on short-term deals in december, and have helped neil lennon 's side steer clear of relegation. eidur gudjohnsen has scored five times in the championship, as well as once in the cup this season. lennon has also fined midfielders barry bannan and neil danns two weeks wages this week. both players have apologised to lennon .
APES score: 1.00

Questions from the CNN/Daily Mail Dataset:
Q: goalkeeper ____ also rewarded with new contract; **A**: adam bogdan
Q: ____ and neil danns both fined by club after drinking incident; **A**: barry bannan
Q: barry bannan and ____ both fined by club after drinking incident; **A**: neil danns

Figure 1: Example 3083 from the test set.

While it has been shown that ROUGE is correlated to Pyramid, Louis and Nenkova (2013) show that this summary level correlation decreases significantly when only a single reference is given. In contrast to the smaller manually curated DUC datasets used in the past, more recent large-scale summarization and headline generation datasets (*CNN/Daily Mail* (Hermann et al., 2015), Gigaword (Graff et al., 2003), New York Times (Sandhaus, 2008)) provide only a single reference summary for each source document. In this work,

Proceedings of NAACL-HLT 2019, pages 3938–3948
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

we introduce a new automatic evaluation metric more suitable for such single reference news article datasets.

We define APES, ***A****nswering* ***P****erformance for* ***E****valuation of* ***S****ummaries*, a new metric for automatically evaluating summarization systems by querying summaries with a set of questions central to the input document (see Fig. 1).

Reducing the task of summaries evaluation to an extrinsic task such as question answering is intuitively appealing. This reduction, however, is effective only under specific settings: (1) Availability of questions focusing on central information and (2) availability of a reliable question answering (QA) model.

Concerning issue 1, questions focusing on salient entities can be available as part of the dataset: the headline generation dataset most used in recent years, the *CNN/Daily Mail* dataset (Hermann et al., 2015), was constructed by creating questions about entities that appear in the reference summary. Since the target summary contains salient information from the source document, we consider all entities appearing in the target summary as *salient entities*. In other cases, salient questions can be generated in an automated manner, as we discuss below.

Concerning issue 2, we focus on a relatively easy type of questions: given source documents and associated questions, a QA system can be trained over fill-in-the-blank type questions as was shown in Hermann et al. (2015) and Chen et al. (2016). In their work, Chen et al. (2016) achieve 'ceiling performance' for the QA task on the *CNN/Daily Mail* dataset. We empirically assess in our work whether this performance level (accuracy of 72.4 and 75.8 over *CNN* and *Daily Mail* respectively) makes our evaluation scheme feasible and well correlated with manual summary evaluation.

Given the availability of salient questions and automatic QA systems, we propose APES as an evaluation metric for news article datasets, the most popular summarization genre in recent years.

To measure the APES metric of a candidate summary, we run a trained QA system with the summary as input alongside a set of questions associated with the source document. The APES metric for a summarization model is the percentage of questions that were answered correctly over the whole dataset, as depicted in Fig. 2. We leave the task of extending this method to other genres for future work.

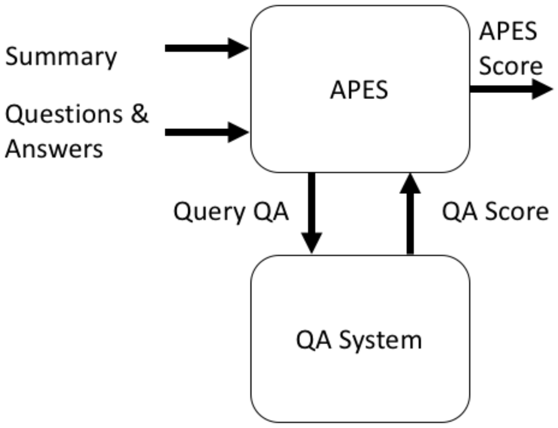


Figure 2: Evaluation flow of APES.

Our contributions in this work are: (1) We first present APES, a new extrinsic summarization evaluation metric; (2) We show APES strength through an analysis of its correlation with Pyramid and Responsiveness manual metrics; (3) we present a new abstractive model which maximizes APES by increasing attention scores of salient entities, while increasing ROUGE to competitive level. We make two software packages available online: (a) An evaluation library which receives the same input as ROUGE and produces both APES and ROUGE scores.[1] (b) Our PyTorch (Paszke et al., 2017) based summarizer that optimizes APES scores together with trained models.[2]

2 Related Work

2.1 Evaluation Methods

Automatic evaluation metrics of summarization methods can be categorized into either *intrinsic* or *extrinsic* metrics. Intrinsic metrics measure a summary's quality by measuring its similarity to a manually produced target gold summary or by inspecting properties of the summary. Examples of such metrics include ROUGE (Lin, 2004), Basic Elements (Hovy et al., 2006) and Pyramid (Nenkova et al., 2007). Alternatively, extrinsic metrics test the ability of a summary to support performing related tasks and compare the performance of humans or systems when completing a task that requires understanding the source document (Steinberger and Ježek, 2012). Such extrinsic tasks may include text categorization, infor-

---

[1] `www.github.com/mataney/APES`

[2] `www.github.com/mataney/APES-optimizer`

mation retrieval, question answering (Jing et al., 1998) or assessing the relevance of a document to a query (Hobson et al., 2007).

ROUGE, or "Recall-Oriented Understudy for Gisting Evaluation" (Lin, 2004), refers to a set of automatic intrinsic metrics for evaluating automatic summaries. ROUGE-N scores a candidate summary by counting the number of N-gram overlaps between the automatic summary and the reference summaries. Other notable metrics from this family are ROUGE-L, where scores are given by the Longest Common Subsequence (LCS) between the suggested and reference documents, and ROUGE-SU4, which uses skip-bigram, a more flexible method for computing the overlap of bigrams.

The Pyramid method (Nenkova et al., 2007) is a manual evaluation metric that analyzes multiple human-made summaries into "Summary Content Units" (SCUs) and assigns importance weights to each SCU. Different summaries are scored by assessing the extent to which they convey SCUs according to their respective weights. Pyramid is most effective when multiple human-made summaries alongside manual intervention to detect SCUs in source and target documents. The Basic Elements method (Hovy et al., 2006), an automated procedure for finding short fragments of content, has been suggested to automate a method related to Pyramid. Like Pyramid, this method requires multiple human-made gold summaries, making this method expensive in time and cost. Responsiveness (Dang, 2005), another manual metric is a measure of overall quality combining both content selection, like Pyramid, and linguistic quality. Both Pyramid and Responsiveness are the standard manual approaches for content evaluation of summaries.

Automated Pyramid evaluation has been attempted in the past (Owczarzak, 2009; Yang et al., 2016; Hirao et al., 2018). This task is complex because it requires (1) identifying SCUs in a text, which requires syntactic parsing and the extraction of key subtrees from the identified units, and (2) the clustering of these extracted textual elements into semantically similar SCUs. These two operations are noisy, and the compounded performance summary evaluation is relying on noisy intermediary representation accordingly suffers.

Other relevant quantities for summaries quality assessment include: readability (or fluency), grammaticality, coherence and structure, focus, referential clarity, and non-redundancy. Although some automatic methods were suggested as summarization evaluation metrics (Vadlapudi and Katragadda, 2010; Tay et al., 2017), these metrics are commonly assessed manually, and, therefore, rarely reported as part of experiments.

Our proposed evaluation method, APES, attempts to capture the capability of a summary to enable readers to answer questions – similar to the manual task initially discussed in Jing et al. (1998) and recently reported in Narayan et al. (2018). Our contribution consists of automating this method and assessing the feasibility of the resulting approximation.

2.2 Neural Methods for Abstractive and Extractive Summarization

The first paper to use an end-to-end neural network for the summarization task was Rush et al. (2015): this work is based on a sequence-to-sequence model (Sutskever et al., 2014) augmented with an attention mechanism (Bahdanau et al., 2014). Nallapati et al. (2016) was the first to tackle the headline generation problem using the *CNN/Daily Mail* dataset (Hermann et al., 2015) adopted for the summarization task.

See et al. (2017) followed the work of Nallapati et al. (2016) and added an additional loss term to reduce repetitions at decoding time. Paulus et al. (2017) introduces intra-attention in order to attend over both the input and previously generated outputs. The authors also present a hybrid learning objective designed to maximize ROUGE scores using Reinforcement Learning.

All the papers mentioned above have been evaluated using ROUGE, and all, except for Rush et al. (2015), used *CNN/Daily Mail* as their main headline generation dataset. Of all the mentioned models we compare our suggested model only to (See et al., 2017), as it is the only paper to publish output summaries.

3 APES

Evaluating a summarization system with APES applies the following method: APES receives a set of news articles summaries, question-and-answer pairs referring to central information from the text and an automatic QA system. Then, APES uses this QA system to determine the total number of questions answered correctly according to the re-

Original Reference Summary:

Arsenal beat Burnley 1-0 in the EPL. a goal from Aaron Ramsey secured all three points. win cuts Chelsea 's EPL lead to four points .

Produces questions:

Q: ____ beat @entity7 1-0 in the @entity4; **A**: Arsenal
Q: @entity0 beat ____ 1-0 in the @entity4; **A**: Burnley
Q: @entity0 beat @entity7 1-0 in the ____; **A**: EPL
Q: a goal from ____ secured all three points; **A**: Aaron Ramsey
Q: win cuts ____'s @entity4 lead to four points; **A**: Chelsea
Q: win cuts @entity19 's ____ lead to four points; **A**: EPL

Figure 3: Example 202 from the *CNN/Daily Mail* test set.

ceived summaries. The evaluation process is depicted in Fig. 2. We use Chen et al. (2016)'s model trained on the *CNN* dataset as our QA system for all our experiments. For a given summarizer and a given dataset, APES reports the average number of questions correctly answered from the summaries produced by the system.

This method is especially relevant for the main headline generation dataset used in recent years, the *CNN/Daily Mail* dataset, as it was initially created for the question answering task by Hermann et al. (2015). It contains 312,085 articles with relevant questions scraped from the two news agencies' websites. The questions were created by removing different entities from the manually produced highlights to create 1,384,887 fill-in-the-blank questions. The dataset was later repurposed by Cheng and Lapata (2016) and Nallapati et al. (2016) to the summarization task by reconstructing the original highlights from the questions. Fig. 3 shows an example for creating questions out of a given summary.

3.1 Using APES as an Evaluation Metric for any News Datasets

When questions are not intrinsically available, one requires to (1) automatically generate relevant questions; (2) use an appropriate automatic QA system.

Similarly to the method used in Hermann et al. (2015), we produce fill-in-the-blank questions in the following way: given a reference summary, we find all possible entities, (*i.e.*, Name, Nationality, Organization, Geopolitical Entity or Facility) using an NER system (Honnibal and Johnson, 2015) and we create fill-in-the-blank type questions where the answers are these entities. We provide code for this procedure and apply it on the *AESOP* datasets in our experiments[3].

For the automatic QA system, we reused in our experiment the same QA system trained on *CNN/Daily Mail* for different News datasets (including AESOP). To enable reproducibility, the trained models used are available online.

4 APES on the TAC2011 AESOP Task

To evaluate if an automatic metric can accurately measure a summarization system performance, we measure its correlation to manual metrics. The TAC 2011 *Automatically Evaluating Summaries of Peers* (*AESOP*) task (Owczarzak and Dang, 2011) has provided a dataset that includes, alongside the source documents and reference summaries, three manual metrics: Pyramid (Nenkova et al., 2007), Overall Responsiveness (Dang, 2005) and Overall Readability. Two sets of documents are provided, we use only the documents from the first set (Generic summarization), as the second set is relevant to the update summarization task.

To evaluate APES on the AESOP dataset, we create the required set of questions as presented in Fig. 3. We used the same QA system (Chen et al., 2016) trained on the *CNN* dataset. This system is a competent QA system for this dataset, as both AESOP and *CNN* consist of news articles. Training a QA model on the AESOP dataset would be optimal, but it is not possible due to the small size of this dataset. Nonetheless, even this incomplete QA system reports valuable results that justify APES value.

While the two datasets are similar, they differ dramatically in the type of topics the articles cover. *CNN/Daily Mail* articles deal with people, or more generally, Named Entities, averaging 6 named entities per summary. In contrast, TAC summaries average 0.87 entities per summary. The TAC dataset is divided into various topics. The first four topics, *Accidents and Natural Disasters*, *Attacks*, *Health and Safety* and *Endangered Resources* average 0.65 named entities per summary, making them incomparable to the typical case in the *CNN/Daily Mail* dataset. The last topic, *Investigations and Trials*, averages 3.35 named entities per summary, making it more similar. We report correlation only on this segment of TAC, which contains 204 documents.

---

[3] `https://github.com/mataney/APES-on-TAC2011`

	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU	APES
Pyramid	0.590	0.468*	0.599	0.563*	**0.608**
Responsiveness	0.540	0.518*	0.537	0.541	**0.576**

Table 1: Pearson Correlation of ROUGE and APES against Pyramid and Responsiveness on summary level. Statistically significant differences are marked with *.

	R-1	R-2	R-L	R-SU	APES
R-1	1.00	0.83	0.92	0.94	0.66
R-2		1.00	0.82	0.90	0.61
R-L			1.00	0.89	0.66
R-SU				1.00	0.67
APES					1.00

Table 2: Correlation matrix of ROUGE and APES.

Model	APES	#Entities	#Salient Entities
See et al. (2017)	38.2	4.90	2.57
Baseline model	39.8	4.99	2.61
Gold Summaries	85.5	6.00	4.90

Table 3: Average number of entities and salient entities.

We follow the work of Louis and Nenkova (2013) and compare input level APES scores with manual Pyramid and Responsiveness scores provided in the AESOP task. Results are in Table 1. In *Input level*, correlation is computed for each summary against its manual score. In contrast, *system level* reports the average score for a summarization system over the entire dataset.

While ROUGE baselines were beaten only by a very small number of suggested metrics in the original AESOP task, we find that APES shows better correlation than the popular R-1, R-2 and R-L, and the strong R-SU. Although showing statistical significance for our hypothesis is difficult because of the small dataset size, we claim APES gives an additional value comparing to ROUGE: ROUGE metrics are highly correlated with each other (around 0.9) as shown in Table 2, indicating that multiple ROUGE metrics provide little additional information. In contrast, APES is not correlated with ROUGE metrics to the same extent (around 0.6). The above suggests that APES offers additional information regarding the text in a manner that ROUGE does not. For this reason, we believe APES complements ROUGE.

Louis and Nenkova (2013) further shows that ROUGE correlation to manual scores tends to drop when reducing the number of reference summaries. While APES is not immune to this, as the number of questions becomes smaller when the number of reference summaries is reduced, it still performs well when reducing the number of references to a single document. In the AESOP dataset, when comparing with respect to each of the 8 assessors separately on Pyramid and Responsiveness, the correlation of APES is highest in 7 out of 16 trials, while that of R1 is highest in 6 trials and RL in 2 trials. In general, the correlation between any of the metrics and single references is extremely noisy, indicating that reliance on evaluations of a single reference, which is standard on large-scale summarization datasets, is far from satisfactory.

We have established that APES achieves equal or improved correlation with manual metrics when compared to ROUGE, and captures a different type of information than ROUGE, by that, APES can complement ROUGE as an automatic evaluation metric. We now turn to develop a model that directly attempts to optimize APES.

5 Model

News articles include a high number of named entities. When analyzing systems performance on APES (Table 3), a system may fail either when it misses to generate a salient entity in the summary, or when it includes the salient entity, but in a context not relevant to corresponding questions. When this happens, the QA system would not be able to identify the entity as an answer to a question referring to the context.

We compared the average number and type of entities in summaries generated by existing automatic summarizers to that in reference summaries. We note that the observed models, while producing state-of-the-art ROUGE scores and a high number of named entities (5 vs. 6 on average), fail to focus on salient entities when generating a summary (about 2.6 salient entities are mentioned on average vs. 4.9 in the reference summaries). Notice that solely increasing the number of entities is damaging: mentioning too many entities causes a decrease in the QA accuracy, as the number of possible answers increases, which would distract the QA system. This has motivated us in suggesting the following model.

5.1 Baseline Model

To experiment with direct optimization of APES, we reconstruct as a starting point a model that encapsulates the key techniques used in recent abstractive summarization models. Our model is based on the OpenNMT project (Klein et al., 2017). All PyTorch (Paszke et al., 2017) code, including entities attention and beam search refinement is available online[4]. We also include generated summaries and trained models in this repository.

Recent work in the field of abstractive summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017) share a common architecture as the foundation for their neural models: an encoder-decoder model (Sutskever et al., 2014) with an attention mechanism (Bahdanau et al., 2014). Nallapati et al. (2016) and See et al. (2017) augment this model with a copy mechanism (Vinyals et al., 2015). This architecture minimizes the following loss function:

$$\begin{aligned} \text{loss}_t &= -\log P(w_t^*) \\ \text{loss} &= \frac{1}{T_y}\sum_{t=0}^{T_y} \text{loss}_t \end{aligned} \tag{1}$$

$loss_t$, is the negative log likelihood of generating the gold target word w_t^* at timestep t where $P(\cdot)$ is the probability distribution over the vocabulary. We refer the reader to See et al. (2017) for a more detailed description of this architecture.

Unlike See et al. (2017), we do not train a specific coverage mechanism to avoid repetitions. Instead, we incorporate Wu et al. (2016)'s refinements of beam search in order to manipulate both the summaries' coverage and their length. In the standard beam search, we search for a sequence Y that maximizes a score function $s(Y, X) = \log(P(Y|X))$. Wu et al. (2016) introduce two additional regularization factors, *coverage penalty* and *length penalty*. These two penalties, with an additional refinement suggested in Gehrmann et al. (2018), yield the following score function:

$$\begin{aligned} s(Y,X) &= \log(P(Y|X))/lp(Y) - cp(X;Y) \\ lp(Y) &= \frac{(5+|Y|)^\alpha}{(5+1)^\alpha} \\ cp(X;Y) &= \beta(-T_X + \sum_{i=1}^{T_X}\max(\sum_{j=1}^{T_Y} a_{i,j}, 1.0)) \end{aligned} \tag{2}$$

where α, β are hyper-parameters that control the length and coverage penalties respectively and $a_{i,j}$ is the attention probability of the j-th target word on the i-th source word.

$cp(X;Y)$, the coverage penalty, is designed to discourage repeated attention to the same source word and favor summaries that cover more of the source document with respect to the attention distribution.

$lp(Y)$, the length normalization, is designed to compare between beam hypotheses of different length accurately. In general, beam search favors shorter outputs as log-probability is added at each step, yielding lower scores for longer sequences. lp compensates for this tendency.

In the following section, we describe how we extend this baseline model in order to maximize the APES metric. The new model learns to incorporate more of the salient entities from the source document in order to optimize its APES metric.

5.2 Entities Attention Layer

As we observed, failure to capture salient entities in summaries is one cause for low APES score. To drive our model towards the identification and mention of salient entities from the source document, we introduce an additional attention layer that learns the important entities of a source document. We hypothesize that these entities are more likely to appear in the target summary, and thus are better candidate answers to one of the salient questions for this document.

We learn for each word in the source document its probability of belonging to a salient entity mention. We adopt the classical soft attention mechanism of Bahdanau et al. (2014): after encoding the source document, we run an additional single alignment model with an empty query and a sigmoid layer instead of the standard softmax layer.

$$\begin{aligned} a_j^e &= \sigma(e_j^e) \\ e_j^e &= v^T \tanh(Uh_j + b) \end{aligned} \tag{3}$$

---

[4] `www.github.com/mataney/APES-optimizer`

Model	APES	ROUGE 1	ROUGE 2	ROUGE L
Source	61.1	-	-	-
Gold-Summaries	85.5	100	100	100
Shuffled Gold-Summaries	30.9	100	7.0	58.3
Lead 3	45.1	40.1	17.3	36.3
Pointer-generator + coverage (See et al., 2017)*	38.2	39.3	16.9	35.7
Baseline model	39.8	39.3	17.3	36.3
Our model	**46.1**	**40.2**	**17.7**	**37.0**
Our model with gold entities positions	46.3	40.4	17.8	37.3

Table 4: APES: Percent of questions answered correctly using by document. *Obtained from the model uploaded to `github.com/abisee/pointer-generator`.

where U, b, v are learnable weight matrices, h_j is the encoder hidden state for the j-th word and $\sigma(\cdot)$ is a logistic sigmoid function. a_j^e reflects the probability of the j-th token of being a salient entity.

The second modification comparing to Bahdanau et al. (2014) is that we replace the softmax function with a sigmoid: while in the standard alignment model, we intend to obtain a normalized probability distribution over all the tokens of the source document, here we would like to get a probability of each token being a salient entity independently of other tokens. In order to drive this attention layer towards salient entities, we define an additional term in the loss function.

$$loss_e = BCE(a^e, s^*) \tag{4}$$

where s^* is a binary vector of source length size, where $s_j^* = 1$ if x_j is a salient entity, and 0 otherwise, and BCE is the binary cross entropy function. This term is added to the standard log-likelihood loss, changing equation (1) to the following composite loss function:

$$\text{loss} = \delta\, loss_e + (1-\delta)\frac{1}{T_y}\sum_{t=0}^{T_y} \text{loss}_t \tag{5}$$

where δ is a hyper-parameter. We join these two terms in the loss function in order to learn the entities attention layer while keeping the summarization ability learned by Eq. (1).

5.3 Entities Attention and Beam Search

After the attention layer has learned the probability of each source token to belong to a salient entity, we pass the predicted alignment to the beam search component at test-time. Using this alignment data, we wish to encourage beam search to favor hypotheses attending salient entities.

Accordingly, we introduce a new term ep to the beam search score function of equation (2):

$$\begin{aligned} s(Y,X) &= \log(P(Y|X))/lp(Y) - cp(X;Y) \\ &\quad - ep(X;Y) \\ ep(X;Y) &= \gamma \sum_{i=1}^{T_X} \max(a_i^e - \sum_{j=1}^{T_Y} a_{i,j}, 0.0) \end{aligned} \tag{6}$$

$ep(X;Y)$ penalizes summaries that do not attend parts of the source document we believe are central.

Fig. 4 compares summaries produced by this model and the baseline model by showing their respective attention distribution and the impact on the decision of which words to include in the summary based on the attention level derived from salient entities.

6 Results

We report our results in Table 4. For each system, we present its APES score alongside its F1 scores for ROUGE-1, ROUGE-2 and ROUGE-L, computed using pyrouge [5].

We first report APES results on full source documents and gold summaries, in order to assess the capabilities of the QA system used for APES. A simple answer extractor could answer 100% of the questions given the gold-summaries. But the QA system is trained over the source documents and learns to generalize and not "just" extract the answer. Answering questions from the full documents is indeed more difficult than from the gold-summaries because the QA system must locate the answer among multiple distractors. While gold-summaries present a very high APES score, the

---

[5] `https://pypi.org/project/pyrouge/`

Source document:

jack wilshere may rub shoulders with the likes of alexis sanchez and mesut ozil on a daily basis but he was left starstruck on thursday evening when he met brazil legend pele . even better for wilshere , the arsenal midfielder was given the opportunity to interview the three-time world cup winner during the launch party of 10ten talent . both wilshere and pele , along with glenn hoddle , are clients and the england international made sure his fans on twitter knew about their meeting by posting several tweets . brazil legend pele -lrb- left -rrb- and arsenal midfielder jack wilshere pose for a photo during launch of 10ten talent . wilshere was given the ‘ honour to interview the legendary pele and asked twitter questions from fans : earlier on thursday , wilshere tweeted : ‘ looking forward to meeting @pele tonight . i ll be asking the best questions you sent . #jackmeetspele . the 23-year-old then followed this up with several tweets about the event , many of which included photos of pele . meanwhile , pele has acknowledged that last year s world cup was a ‘ disaster for brazil but is not surprised how quickly the likes of oscar and ramires have bounced back in the barclays premier league this season : brazil were humiliated by germany in a 7-1 semi-final defeat and the hosts were then thrashed 3-0 by holland in the third-place play-off . pele scored 77 goals in 92 games for brazil and won the world cup three times but the former santos striker still finds last year s capitulation difficult to understand .

Target Summary:

jack wilshere was joined by former **england** manager **glenn hoddle**. the **arsenal** midfielder interviewed **pele** at launch of **10ten talent**. **pele** scored 77 goals in 92 games for **brazil** and won three world cups. the **brazil** legend says the 2014 world cup performance was not expected. the hosts were humiliated 7-1 by **germany** in the semi-finals last summer. **pele** is, however, not surprised by reaction of **oscar** and **ramires** this year.

Baseline Model Prediction:

jack wilshere was given the opportunity to interview the three-time world cup winner. both wilshere and pele are clients and the england international. pele has acknowledged that last year’s world cup was a ‘disaster’

Our Model Prediction:

jack wilshere was given the ‘honour to interview the legendary pele’ and asked twitter questions from fans. pele has acknowledged that last year’s world cup was a ‘disaster’ for brazil but is not surprised how quickly the likes of oscar and ramires have bounced back in the premier league this season. the brazil legend scored 77 goals in 92 games for brazil and won the world cup three times.

Figure 4: Example 4134 from the *CNN/Daily Mail* test set. Colors and underlines in the source reflect differences between baseline and our model attention weights: Red and a single underline reflects words attended by baseline model and not our model, Green and double underline reflects the opposite. Entities in bold in the target summary are answers to the example questions.

score reported for the source documents (61.1%) is a realistic upper bound for APES.

We then present shuffled gold-summaries, where we randomly shuffled the location of each unigram in the gold summary. This score shows that even when all salient entities are in the shuffled text, APES is sensitive to the loss of coherence, readability and meaning. This confirms that APES does not only match the presence of entities. In contrast, ROUGE-1 fails to punish such incoherent sequences. Finally, we report ROUGE and APES for the strong Lead 3 sentences of the source document - a baseline known to beat most existing abstractive methods.

We then present APES and ROUGE scores for abstractive models, See et al. (2017)’s model, our baseline model and our APES-optimized model. Our model achieves significantly higher APES scores (46.1 vs. 39.8) and improves all ROUGE metrics (by about 1 F-point over the baselines). The scores on the validation set are 46.6, 41.2, 18.4, 38.1 for APES, R1, R2, RL respectively.

While our objective is maximizing APES score, our model also increases its corresponding ROUGE scores. Unlike Paulus et al. (2017) where the authors suggested a Reinforcement Learning based model to optimize ROUGE specifically, we optimize for APES and gain better ROUGE score.

We finally report the results obtained by our model when gold salient entities positions are given as oracle inputs instead of the predicted a^e scores. The corresponding score (46.3 vs. 46.1) is only slightly above the score obtained by our model. This indicates that the component of our model predicting entity saliency is good enough to drive summarization.

We carried out an informal error analysis to examine why some summaries perform worse than others with our architecture. We compared summaries that produce perfect APES score (1,630 out of 11,490 total) to the summaries with zero APES score (1,691). We measure the density of salient named entities in the source document: #(salient entity mentions)/#(distinct salient entities). This density in the case of perfect APES summaries is much higher than that for low APES summaries (4.9 vs. 3.6). This observation suggests that we fail to produce higher APES scores when the salient entities aren't marked through sheer repetition.

7 Conclusion

We introduced APES, a new automatic summarization evaluation metric for news articles datasets based on the ability of a summary to answer questions regarding salient information from the text. This approach is useful in domains with source documents of about 1k words that focus on named entities - such as news articles, where named entities are effectively aligned with Pyramid SCUs. In other non-news domains, and longer documents, other methods for generating questions should be designed. We compare APES to manual evaluation metrics on the TAC 2011 AESOP task and confirm its value as a complement to ROUGE.

We introduce a new abstractive model that optimizes APES scores on the *CNN/Daily Mail* dataset by attending salient entities from the input document, which also provides competitive ROUGE scores.

Acknowledgements

This research was supported by the Lynn and William Frankel Centre for Computer Science at Ben-Gurion University.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4:1.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Tsutomu Hirao, Hidetaka Kamigaito, and Masaaki Nagata. 2018. Automatic pyramid evaluation exploiting edu-based extractive reference summaries. In *EMNLP*.

Stacy President Hobson, Bonnie J Dorr, Christof Monz, and Richard Schwartz. 2007. Task-based evaluation of text summarization using relevance prediction. *Information Processing & Management*, 43(6):1482–1499.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of the Fifth Conference on Language Resources and Evaluation (LREC 2006)*, pages 604–611. Citeseer.

Hongyan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. 1998. Summarization evaluation methods: Experiments and analysis. In *AAAI symposium on intelligent summarization*, pages 51–59.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *hiP (yi= 1— hi, si, d)*, 1:1.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

Karolina Owczarzak. 2009. Depeval(summ): Dependency-based evaluation for automatic summaries. In *ACL/IJCNLP*.

Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the tac 2011 summarization track: Guided task and aesop task. In *Proceedings of the Text Analysis Conference (TAC 2011), Gaithersburg, Maryland, USA, November*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Josef Steinberger and Karel Ježek. 2012. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring. *arXiv preprint arXiv:1711.04981*.

Ravikiran Vadlapudi and Rahul Katragadda. 2010. On automated evaluation of readability of summaries: Capturing grammaticality, focus, structure and coherence. In *Proceedings of the NAACL HLT 2010 student research workshop*, pages 7–12. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Qian Yang, Rebecca J. Passonneau, and Gerard de Melo. 2016. Peak: Pyramid evaluation via automated knowledge extraction. In *AAAI*.

A Experiment Settings

For our experiments, we used a bidirectional LSTM encoder with 256-dimensional hidden states for each direction, an LSTM decoder with 512-dimensional hidden states and 128-dimensional embeddings for a 50k shared-vocabulary words. We do not use pretrained word embeddings.

We use the Adagrad (Duchi et al., 2011) optimizer with a starting learning rate of 0.15 and gradient clipping with a maximum gradient norm of 2. At train-time source and target documents are truncated to 400 and 100 tokens respectively. After training our baseline model for 20 epochs, we fine-tune the network with Eq. (5) loss for an additional 5 epochs starting again with 0.15 as initial learning rate. Results reported in this paper correspond to $\lambda = 0.01$.

At test-time, we do not truncate the source documents enabling the network to attend overall input text. We use Eq. (6) as the beam search score function, penalizing using $cp(X;Y)$ every single decoding step and $lp(Y)$ and $ep(X;Y)$ only when all hypotheses are done. We choose α, β, γ values of $0.9, 0.5, 0.5$ respectively for our model. We also used Paulus et al. (2017) suggestion of repetition avoidance by blocking trigrams appearing more than once at inference time.

Running APES evaluation on a generated test set (of size 11,490 summaries) takes about 40 minutes using a single process.

Understanding the Behaviour of Neural Abstractive Summarizers using Contrastive Examples

Krtin Kumar
School of Computer Science
McGill University
krtin.kumar@mail.mcgill.ca

Jackie Chi Kit Cheung
School of Computer Science
McGill University
jcheung@cs.mcgill.ca

Abstract

Neural abstractive summarizers generate summary texts using a language model conditioned on the input source text, and have recently achieved high ROUGE scores on benchmark summarization datasets. We investigate how they achieve this performance with respect to human-written gold-standard abstracts, and whether the systems are able to understand deeper syntactic and semantic structures. We generate a set of contrastive summaries which are perturbed, deficient versions of human-written summaries, and test whether existing neural summarizers score them more highly than the human-written summaries. We analyze their performance on different datasets and find that these systems fail to understand the source text, in a majority of the cases.

1 Introduction

Open-domain abstractive summarization is a long-standing goal of the field of automatic summarization. Compared to extractive techniques, abstraction offers the potential to generate much more useful summaries by simplifying and rephrasing the source text (Knight and Marcu, 2002), and furthermore by aggregating information and performing operations which are not possible with extractive techniques (Genest and Lapalme, 2012; Carenini and Cheung, 2008).

Recently, a number of abstractive summarization systems based on neural sequence-to-sequence architectures have been proposed (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2018; Chen and Bansal, 2018). These systems learn a compressed representation of the source text using an encoder, then generate the output summary using a conditional decoder. Such neural abstractive systems have achieved very good ROUGE scores on different datasets.

Source
A former Iraqi army chief of staff being investigated in denmark for war crimes is believed to be back in **Iraq**, one of his sons said Tuesday.
Contrastive 1
Iraqi general missing **in** denmark believed to be back **from** Iraq.
Contrastive 2
Iraqi general missing from **Iraq** believed to be back in Iraq.

Table 1: Examples of generated contrastive summaries. Bold indicate switched words

Our interest in this paper is to investigate how these abstractive systems achieve such results, and whether they represent progress towards language understanding and generation. ROUGE arguably provides a limited view of the performance of such systems, as they only relate the system summary to a fixed number of gold-standard summaries. We propose a novel method to directly test the abstractive summarizers in terms of how they score potential candidate summaries, viewing them as conditional language models. This allows us to test whether the summarizers favour output summaries with specific desired qualities, such as generating a summary that is semantically consistent and entailed by the source text.

We test how well the neural abstractive summarizers distinguish human-written abstracts from contrastive distractors, which are clearly incorrect summaries that are generated using a rule-based procedure. Table 1 shows contrastive examples which are clearly incorrect[1]. In majority of source texts, we are able to find a contrastive example that scores more highly than the gold-standard summary.

---

[1] For other NLP tasks, others have proposed a similar notion called *adversarial* examples (Jia and Liang, 2017). Since the term 'adversarial' has traditionally implied learning to specifically attack the weakness of a model, which we do not do, we refrain from using the word 'adversarial'.

Proceedings of NAACL-HLT 2019, pages 3949–3954
Minneapolis, Minnesota, June 2 - June 7, 2019.

Our work demonstrates the difficulty of controlling expressive neural abstractive systems to produce correct and fluent output. It also underscores the need to revisit fundamental issues in summarization evaluation for neural abstractive models, so that a comprehensive evaluation scheme that captures all relevant aspects of summary quality can be developed. Our code for generating contrastive summaries is available online.[2]

2 Related Work

Most work in neural abstractive summarization has focused on optimizing ROUGE, whether implicitly by maximum likelihood training or explicitly by reinforcement learning. While this could certainly capture aspects of the content selection problem, we believe that the focus should now shift towards semantic correctness and readability.

Cao et al. (2018) took a step in this direction through their fact-aware neural abstractive summarization system. They use fact descriptions of the source as additional features for the summarizer, and showed improved faithfulness according to human judgments. Multi-task learning is another approach used by Pasunuru et al. (2017) to reduce semantic errors in the generated summaries. They jointly learn summarization and entailment generation tasks, using different encoders but a shared decoder.

A number of automatic evaluation metrics have shown high correlation with human judges (Liu and Liu, 2008; Graham, 2015), but these results are either restricted to extractive systems or were performed with respect to human-generated summaries. Correlation values are significantly reduced when performed on abstractive summarization systems and datasets (Toutanova et al., 2016).

3 Generating Contrastive Summaries

In this section, we describe our method for evaluating summarization systems based on whether they can separate human-written gold summaries from automatically generated contrastive summaries. We define a contrastive summary to be similar to a gold summary, except it contains a perturbation. The perturbation results in either a semantic discrepancy, where facts in source and summary do not corroborate, or a readability issue, where issues with grammar or fluency renders the summary clearly incorrect. Below, we first describe our basic method of introducing these discrepancies, then describe a number of restrictions we apply to ensure that the generated contrastive summaries are of high quality.

Rule	**Switching Criteria**
Noun	NN, NNP or NNS must match. For NNP child DET if present is also switched.
Preposition	For IN tags: parents and their dependencies must match.
Verb	VBP, VBG, VBZ, VBN, VBD or VB must match
Adjective	JJ, JJR or JJS must match

Table 2: Rules for selecting words to switch when generating the contrastive summaries. The tags are as per the Penn Tree-bank (Santorini, 1990).

Perturbation by switching words. Given pairs of source texts and gold summaries, we generate multiple contrastive summaries for each source text by perturbing its associated gold summary. There are many types of possible perturbations, but we focus on two strategies: 1) switching words within a gold summary, and 2) replacing a word in gold summary by a word from the source text. We chose these types of perturbations as they are likely to result in "difficult" contrastive summaries that contain words which are likely to appear in a reasonable summary of the source text, but which are nevertheless incorrect.

In order to select the words to be swapped, we apply four rules, separated by syntactic category, as shown in Table 2, using the dependency parse of the texts (Manning et al., 2014). We switch words either within a gold summary, or from the source text and use a single rule at a time for generating a contrastive summary. For example, if the POS tag *NNS* is matched between a word *'sides'* in the source text and the word *'combatants'* in the gold summary, then the Noun rule would apply, and the words would be switched to obtain the contrastive summary.

Further, for the Noun and Verb categories, the switched words may not match in number or verb conjugation. We use SimpleNLG (Gatt and Reiter, 2009) to convert the word to the appropriate inflectional form of the destination's POS tag.

---

[2] `https://github.com/krtin/ContrastiveSummaries`

Further restrictions. We place a number of restrictions on the words switched, to ensure that the generated summary is contrastive compared to the gold summary. We do not allow switching of the same words. We also do not allow words to be switched if they are separated by any of the following: *'or'*, *'and'* or *','*, as these are likely to be commutative operators.

Furthermore, we only allow switching of words from the source text if the context of the words to be switched sufficiently differ from each other. We compute unigram overlap around a context window of size 2 on each side, and allow a switch when the overlap proportion is less than 0.65. These settings were determined by manual inspection of the generated summaries, and allowed us to reduce the number of examples where generated summary is not contrastive, without significantly reducing the number of generated summaries.

We will describe the results of a human verification study in Section 5, in which we ask human raters to check the quality of our contrastive summaries.

4 Evaluation

We apply our set of contrastive summaries to evaluate a number of neural abstractive summarizers. For each summarizer under evaluation, we assume access to a conditional language model which defines a probability distribution over words conditioned on the source. Formally, such a language model is given by Equation 1:

$$P(y_i|\theta, \mathcal{S}, y_1...y_{i-1}), \qquad (1)$$

here $\mathcal{S}$ is the source, θ are model parameters, $y_i \in V_{sm}$ represents the i^{th} word in the summary, V_{sm} is the vocabulary space of the summary and P is the conditional probability. $\mathcal{S} \subseteq (s_1, ..., s_n)$ where $s_i \in V_{so}$, V_{so} is the vocabulary space of the source and n is the maximum source length. Further, we use Equation 2 as our scoring function,

$$p(y) = \sum_{i=1}^{m} \log P(y_i|\theta, \mathcal{S}, y_1...y_{i-1}), \qquad (2)$$

here m is maximum summary length and y represents a gold or contrastive summary. For a given triple of source, gold (g) and contrastive (c) summary, if $p(g) > p(c)$, then we label the triple *'dodged'*, since the summarizer successfully dodged the generated contrastive summary. If a system is able to dodge all contrastive summaries generated from a source and gold summary tuple, then we label the tuple as *'escaped'*.

5 Experiments

Datasets. We experimented on two datasets, for two abstractive summarization tasks. The first is a short summarization task, where the summary is one sentence long, for which we use the **Gigaword** corpus (Graff et al., 2011; Napoles et al., 2012). We use the scripts provided by Rush et al. (2015) to process the Gigaword corpus, which contains the first sentence of the article and the headline as source and gold summary pairs. The test set contains about 250K source-summary pairs from which we randomly selected 10K pairs and generated 509K contrastive summaries.

The second is a long summarization task, in which the summary consists of multiple sentences. We use the **CNN/Dailymail** corpus, where the highlights of the articles are used as the gold summary (Hermann et al., 2015). We used the scripts from Nallapati et al. (2016) to get the data and use the non-anonymized version like See et al. (2017). We use 11.49K test pairs and were able to generate 563K contrastive summaries.

Models. We analyze and evaluate three state-of-the-art neural abstractive summarization systems: ABS+ (Rush et al., 2015), GTP (See et al., 2017) and FAS (Chen and Bansal, 2018). The **ABS+** system uses an attention-based neural language model as an encoder and a feed-forward neural network for decoding, and is trained on the Gigaword corpus. The **GTP** system is a seq2seq model with attention on the encoder and a pointer-generator mechanism to choose words from the source in the decoder and, is trained on the CNN/Dailymail corpus. **FAS** uses reinforcement learning algorithm to extract the most important sentences from the source text, and then summarizes each sentence using a similar architecture as GTP on the CNN/Dailymail corpus.

These systems have performed well on small and large text summarization tasks, and have open-source implementation available from the authors. We would have liked to test other relevant systems (Pasunuru et al., 2017; Cao et al., 2018), but were unable to obtain their implementations.

Experimental Details. The CNN/Dailymail corpus has a large source length, thus the set of

	Rule	**Dodged (%)**		
		CNN		**Gigaword**
		GTP	**FAS**	**ABS+**
Gold	Noun	98.8	96.5	49.5
	Prep	97.8	96.9	55.4
	Verb	99.0	98.5	48.7
	Adj	97.8	95.6	55.7
Source	Noun	94.3	88.5	47.9
	Prep	92.0	88.2	50.1
	Verb	94.9	91.9	50.2
	Adj	92.6	87.2	49.7

Table 3: Rule-wise Performance, here Source and Gold are based upon rule perturbations in Section 3

Model	**Dataset**	**Dodged**	**Escaped**
GTP	CNN	96.3%	29.8%
FAS	CNN	92.9%	12.2%
ABS+	Gigaword	48.6%	10.8%

Table 4: Performance on *dodged*, *escaped* metrics.

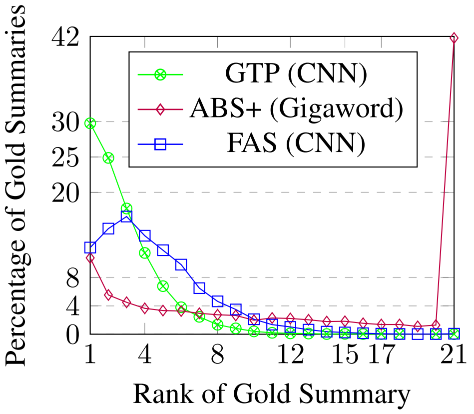


Figure 1: Distribution of Gold Summary Rank

contrastive summaries is very large. To restrict the number of contrastive summaries we randomly select approximately 50 generated summaries, while maintaining the rule-wise distribution. The rule-wise distribution was estimated based upon contrastive summaries, generated from a subset of 100 gold standard summaries from CNN/Dailymail corpus.

In order to correctly evaluate the FAS system, for each extracted sentence we generate all sentences in the gold summary, and pick the set of summaries with the maximum probability.

Human verification. To ensure that we are generating incorrect contrastive summaries, 200 randomly sampled summaries from the Gigaword corpus were evaluated by a human annotator, to verify if a semantic discrepancy or a readability issue was present. We ensured that we sample equally across all the 8 rules, and that we restrict our set of contrastive summaries which the ABS+ system was not able to *'dodge'*. We found that 49.5% had a readability issue while 43.5% had a discrepancy issue, and 93% of the examples had at least one of these issues. This indicates that the vast majority of our contrastive examples are "true negatives"; i.e., a perfect summarization system should score them lower than the gold standard summary.

6 Results

We summarize our results in Table 4, and report rule-wise results in Table 3. We also include examples where the ABS+ system is unsuccessful in dodging the generated contrastive summaries, in Table 5. Since these metrics directly evaluate the posterior distribution of a summarizer, it allows us to explicitly recognize problematic examples for a model. We also look at what percentage of gold summaries lie across different ranks of gold summaries in Figure 1. This gives us an insight into distribution of gold summaries, across different ranks. The rank of a gold summary is 1 plus the number of contrastive summaries that scored higher than it.

Dodged and Escaped. On the CNN/Dailymail dataset, we find both the models were able to dodge most of the contrastive summaries, but a large number of summaries had at least a few contrastive summaries which scored higher than the gold summary, as reflected by the *escaped* metric.

The FAS model performs worse than GTP model, this might be because the abstraction model only observes one sentence, and thus the probability of observing a word outside the source sentence is higher for the contrastive summaries.

Rule-wise Analysis. The GTP and FAS models perform better on rules which switch words within the gold-standard summary. Thus, the decoder LSTM has captured the data distribution very well for words within the summary but is not generalizing for words outside it. This suggest that using words outside the source vocabulary might help in generating harder contrastive examples. The ABS+ model is better in capturing data distributions of prepositions and adjectives. This points towards biases towards particular distributions and can be helpful in further improving these models.

Rank of Gold Summary. As shown in Figure 1, almost all the gold summaries have rank lower than 8 for the GTP model, while a large percentage of gold summaries have rank greater than 20, for the ABS+ model. The maximum rank in both the cases is of the order 500K, which is the size of contrastive summaries (Section 5). We suspect that this might be due to behaviourally extractive nature of GTP model, which allows it to easily distinguish any perturbations in the contrastive summaries.

7 Conclusion

We proposed to analyze existing neural abstractive summarizers by testing how they score contrastive summaries, compared to gold-standard ones. For the majority of the gold-standard summaries, we were able to find contrastive examples which score more highly according to current state-of-the-art systems. These examples can be useful not only in evaluating the performance of these systems, but also for improving these systems in the future.

References

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. nlg-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 33–41. Association for Computational Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL*.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2012. Fully abstractive approach to guided summarization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 354–358. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword, linguistic data consortium.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 128–137.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Feifan Liu and Yang Liu. 2008. Correlation between rouge and human evaluation of extractive meeting summaries. In *Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: Short papers*, pages 201–204. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *International Conference on Learning Representations*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*, page 570.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In (See et al., 2017), pages 1073–1083.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350, Austin, Texas. Association for Computational Linguistics.

A Appendices

Example 1	
Source	The Asian economic contagion that contributed overnight to the year's worst loss on wall street returned home with the opening of trading Wednesday, helping to drag some markets to levels they hadn't seen for months, if not years.
Gold	Asian markets fall after big drop on wall street.
Contrastive	Asian markets fall after big drop on **street wall**.
Problems	The system failed to understand that wall street has real-world significance
Example 2	
Source	Panama's colon duty free area, latin america's biggest re-export zone, ground to a halt monday as business leaders shut their doors to protest a tax hike.
Gold	Business leaders launch strike in Panama's free zone by James.
Contrastive	Business **strike** launch **leaders** in Panama's free zone by James.
Problems	Failed to understand that business leaders belong to a single entity
Example 3	
Source	Iran stood firm on its position towards terrorism and the middle east peace process in talks with an EU mission here . . .
Gold	Iran refuses to budge on terrorism, peace process in talks with EU by Laurent.
Contrastive	Iran refuses to budge on terrorism, peace process **by** talks with EU **in** Laurent.
Problems	Improper usage and understanding of Prepositions
Example 4	
Source	The rear door of a Russian-made cargo plane crammed with Congolese soldiers and their families flew open in midair on Thursday night, 33,000 feet above the jungles of Congo, dropping scores of passengers down a ramp and into the sky, survivors said.
Gold	Passengers fall from plane over Congo; death toll unclear
Contrastive	Passengers fall **over** plane **from** Congo; death toll unclear
Problems	Failed to understand that falling over a plane is improbable in real-world context
Example 5	
Source	The leader of the separatist Georgian black sea region of Abkhazia on Monday rejected Tbilisi's insistence that railway **traffic** via the region would only be restored when refugees displaced by war are allowed to return and their safety is ensured.
Gold	Georgia's breakaway Abkhazia rejects conditions for restoring **rail** traffic.
Contrastive	Georgia's breakaway Abkhazia rejects conditions for restoring **traffic** traffic.
Problems	A simple repetition issue, also pointed out by the authors (Rush et al., 2015)
Example 6	
Source	Incheon, a port city in the republic of Korea (rok), plans to restore its century-old china town, which was destroyed during the Korean war (1950-60), and to build it into a **tourism** attraction by inviting investment from china . . .
Gold	Rok city wants to rebuild china town with Chinese **investment**.
Contrastive	Rok city wants to rebuild china town with Chinese **tourism**.
Problems	The model lacks understanding of the source text

Table 5: Examples of contrastive summaries, that ABS+ system was not able to dodge. Bold indicate switched words

.

Jointly Extracting and Compressing Documents with Summary State Representations

Afonso Mendes[♠] **Shashi Narayan**[◇*] **Sebastião Miranda**[♠]
Zita Marinho[♡♠] **André F. T. Martins**[†♣] **Shay B. Cohen**[◇]

[♠]Priberam Labs, Alameda D. Afonso Henriques, 41, 2°, 1000-123 Lisboa, Portugal
[◇]School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK
[♡] Instituto de Sistemas e Robótica, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[♣]Unbabel Lda, Rua Visconde de Santarém, 67-B, 1000-286 Lisboa, Portugal
`amm@priberam.com`, `shashi.narayan@gmail.com`, `ssm@priberam.com`,
`zam@priberam.com`, `andre.martins@unbabel.com`, `scohen@inf.ed.ac.uk`

Abstract

We present a new neural model for text summarization that first extracts sentences from a document and then compresses them. The proposed model offers a balance that sidesteps the difficulties in abstractive methods while generating more concise summaries than extractive methods. In addition, our model dynamically determines the length of the output summary based on the gold summaries it observes during training, and does not require length constraints typical to extractive summarization. The model achieves state-of-the-art results on the CNN/DailyMail and Newsroom datasets, improving over current extractive and abstractive methods. Human evaluations demonstrate that our model generates concise and informative summaries. We also make available a new dataset of oracle compressive summaries derived automatically from the CNN/DailyMail reference summaries.[1]

1 Introduction

Text summarization is an important NLP problem with a wide range of applications in data-driven industries (e.g., news, health, and defense). Single document summarization—the task of generating a short summary of a document preserving its informative content (Spärck Jones, 2007)—has been a highly studied research topic in recent years (Nallapati et al., 2016b; See et al., 2017; Fan et al., 2018; Pasunuru and Bansal, 2018).

Modern approaches to single document summarization using neural network architectures have primarily focused on two strategies: *extractive* and *abstractive*. The former select a subset of the sentences to assemble a summary (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018a,c). The latter generates sentences that do not appear in the original document (See et al., 2017; Narayan et al., 2018b; Paulus et al., 2018). Both methods suffer from significant drawbacks: extractive systems are wasteful since they cannot trim the original sentences to fit into the summary, and they lack a mechanism to ensure overall coherence. In contrast, abstractive systems require natural language generation and semantic representation, problems that are inherently harder to solve than just extracting sentences from the original document.

(ExConSumm Extractive) • (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

(ExConSumm Compressive) • ~~(CNN)~~ A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen ~~, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war~~.
• Ibrahim al-Rubaish died ~~Monday night~~ in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

Figure 1: Summaries produced by our model. For illustration, the compressive summary shows the removed spans strike-through.

In this paper, we present a novel architecture that attempts to mitigate the problems above via a middle ground, **compressive summarization** (Martins and Smith, 2009). Our model selects a set of sentences from the input document, and

---

[1] Our dataset and code is available at `https://github.com/Priberam/exconsumm`.

[*] Now at Google London.

Proceedings of NAACL-HLT 2019, pages 3955–3966
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

compresses them by removing unnecessary words, while keeping the summaries informative, concise and grammatical. We achieve this by dynamically modeling the generated summary using a Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) to produce **summary state representations**. This state provides crucial information to iteratively increment summaries based on previously extracted information. It also facilitates the generation of variable length summaries as opposed to fixed lengths, in previous extractive systems (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018c; Zhang et al., 2018). Our model can be trained in both extractive (labeling sentences for extraction) or compressive (labeling words for extraction) settings. Figure 1 shows a summary example generated by our model.

Our contributions in this paper are three-fold:

- we present the first end-to-end neural architecture for EXtractive and COmpressive Neural SUMMarization (dubbed EXCONSUMM, see §3),
- we validate this architecture on the CNN/DailyMail and the Newsroom datasets (Hermann et al., 2015; Grusky et al., 2018), showing that our model generates variable-length summaries which correlate well with gold summaries in length and are concise and informative (see §5), and
- we provide a new CNN/DailyMail dataset annotated with automatic compressions for each sentence, and a set of compressed oracle summaries (see §4).

Experimental results show that when evaluated automatically, both the extractive and compressive variants of our model provide state-of-the-art results. Human evaluation further shows that our model is better than previous state-of-the-art systems at generating informative and concise summaries.

2 Related Work

Recent work on neural summarization has mainly focused on sequence-to-sequence (seq2seq) architectures (Sutskever et al., 2014), a formulation particularly suited and initially employed for abstractive summarization (Rush et al., 2015). However, state-of-the-art results have been achieved by RNN-based methods which are extractive. They select sentences based on an LSTM classifier that predicts a binary label for each sentence (Cheng and Lapata, 2016), based on ranking using reinforcement learning (Narayan et al., 2018c), or even by training an extractive latent model (Zhang et al., 2018). Other methods rely on an abstractive approach with strongly conditioned generation on the source document (See et al., 2017). In fact, the best results for abstractive summarization have been achieved with models that are more extractive in nature than abstractive, since most of the words in the summary are copied from the document (Gehrmann et al., 2018).

Due to the lack of training corpora, there is almost no work on neural architectures for compressive summarization. Most compressive summarization work has been applied to smaller datasets (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013). Other non-neural summarization systems apply this idea to select and compress the summary. Dorr et al. (2003) introduced a method to first extract the first sentence of a news article and then use linguistically-motivated heuristics to iteratively trim parts of it. Durrett et al. (2016) also learns a system that selects textual units to include in the summary and compresses them by deleting word spans guided by anaphoric constraints to improve coherence. Recently, Zhang et al. (2018) trained an abstractive sentence compression model using attention-based sequence-to-sequence architecture (Rush et al., 2015) to map a sentence in the document selected by the extractive model to a sentence in the summary. However, as the sentences in the document and in the summary are not aligned for compression, their compression model is significantly inferior to the extractive model.

In this paper, we propose a novel seq2seq architecture for compressive summarization and demonstrate that it avoids the over-extraction of existing extractive approaches (Cheng and Lapata, 2016; Dlikman and Last, 2016; Nallapati et al., 2016a).

Our model builds on recent approaches to neural extractive summarization as a sequence labeling problem, where sentences in the document are labeled to specify whether or not they should be included in the summary (Cheng and Lapata, 2016; Narayan et al., 2018a). These models often condition their labeling decisions on the document representation only. Nallapati et al. (2017) tries to model the summary as the average representation

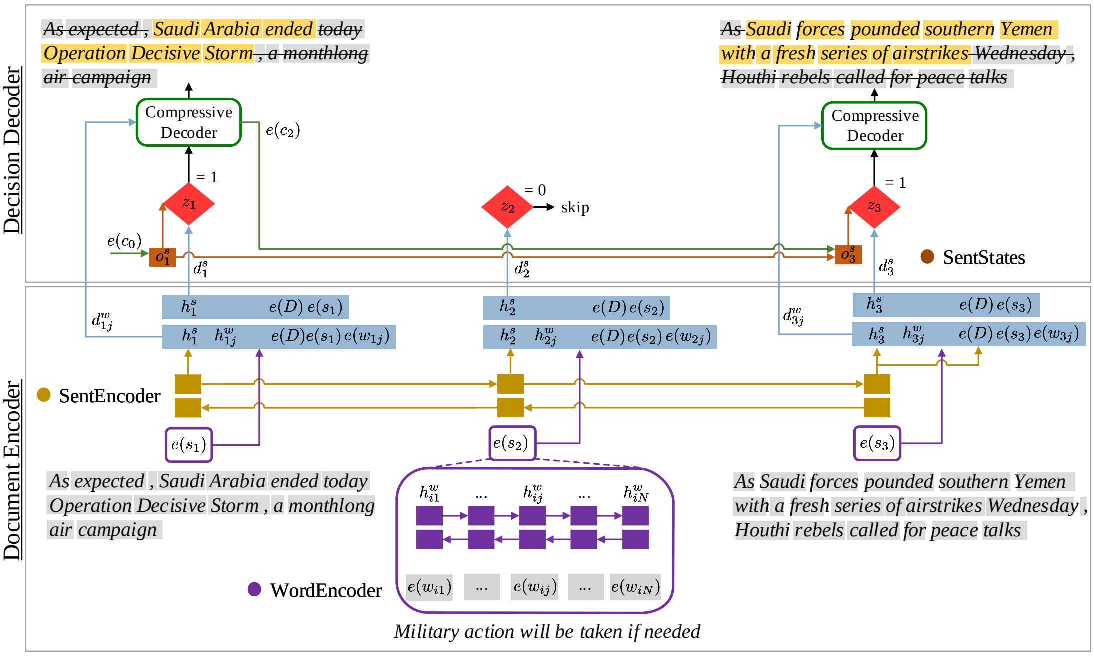


Figure 2: Illustration of our summarization system. The model extracts the most relevant sentences from the document by taking into account the WordEncoder representation of the current sentence $e(s_i)$, the SentEncoder representation of the previous sentence $\boldsymbol{h}_i^s$, the current summary state representation $\boldsymbol{o}_i^s$, and the representation of the document $e(D)$. If a sentence is selected ($z_i = 1$), its representation is fed to SentStates, and we move to the next sentence. Here, sentences s_1 and s_3 were selected. If the model is also compressing, the compressive layer selects words for the final summary (Compressive Decoder). See Figure 3 for details on the decoders.

of the positively labeled sentences. However, as we show later, this strategy is not the most adequate to ensure summary coherence, as it does not take the order of the selected sentences into account. Our approach addresses this problem by maintaining an LSTM cell to dynamically model the generated summary. To the best of our knowledge, our work is the first to use a model that keeps a state of already generated summary to effectively model variable-length summaries in an extractive setting, and the first to learn a compressive summarizer with an end-to end approach.

3 Summarization with Summary State Representation

Our model extracts sentences from a given document and further compresses these sentences by deleting words. More formally, we denote a document $D = (s_1, \ldots, s_M)$ as a sequence of M sentences, and a sentence $s_i = (w_{i1}, \ldots, w_{iN})$ as a sequence of N words. We denote by $\boldsymbol{e}(w_{ij})$, $\boldsymbol{e}(s_i)$ and $\boldsymbol{e}(D)$ the embedding of words, sentences and document in a continuous space. We model document summarization as a sequence labeling problem where the labeler transitions between internal states. Each state is dynamically computed based on the context, and it combines an extractive summarizer followed by a compressive one. First, we encode a document in a multi-level approach, to extract the embeddings of words and sentences ("Document Encoder"). Second, we decode these embeddings using a hierarchical "Decision Decoder." The extractive summarizer labels each sentence s_i with a label $z_i \in \{0, 1\}$ where 1 indicates that the sentence should be included in the final summary and 0 otherwise. An extractive summary is then assembled by selecting all sentences with the label 1. Analogously, the compressive summarizer labels each word w_{ij} with a label $y_{ij} \in \{0, 1\}$, denoting whether the word j in sentence i is included in the summary or not. The final summary is then assembled as the sequence of words w_{ij} for each $z_i = 1$ and $y_{ij} = 1$. See Figures 2 and 3 for an overview of our model. We next describe each of its components in more detail.

3.1 Document Encoder

The document encoder is a two layer biLSTM, one layer encoding each sentence, and the second layer encoding the document. The first layer takes as input the word embeddings $\boldsymbol{e}(w_{ij})$ for each word j in sentence s_i, and outputs the hidden representa-

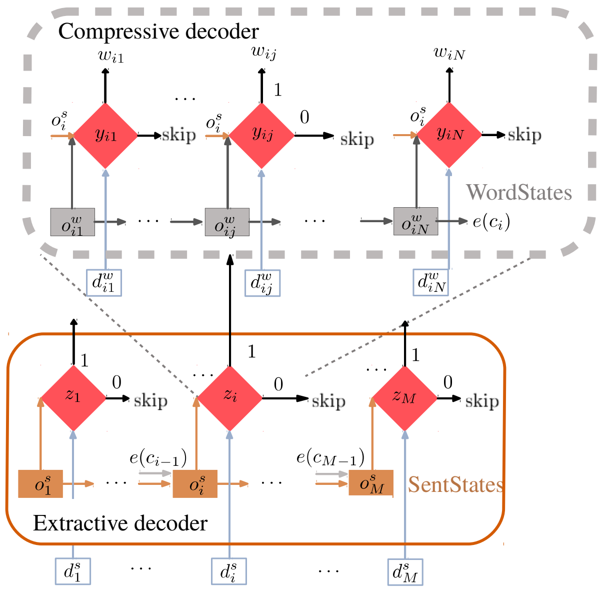


Figure 3: Decision decoder architecture. Decoder contains an extractive level for sentences (orange box) and a compressive level for words (dashed gray box), using an LSTM to model the summary state. Red diamond shapes represent decision variables $z_i = 1$ if $p(z_i \mid \boldsymbol{p}_i) > 0.5$ for selecting the sentence s_i, and $z_i = 0$ if $p(z_i \mid \boldsymbol{p}_i) \leq 0.5$ for skipping this sentence. The same for y_{ij} and $p(y_{ij} \mid \boldsymbol{q}_{ij}) > 0.5$ for deciding over words w_{ij} to keep in the summary.

tion of each word $\boldsymbol{h}_{ij}^w$. The hidden representation consist of the concatenation of a forward $\overrightarrow{\boldsymbol{h}}_{ij}^w$ and a backward $\overleftarrow{\boldsymbol{h}}_{ij}^w$ LSTM (WordEncoder in Figure 2). This layer eventually outputs a representation for each sentence $\boldsymbol{e}(s_i) = [\overrightarrow{\boldsymbol{h}}_{iN}^w, \overleftarrow{\boldsymbol{h}}_{i1}^w]$ that corresponds to the concatenation of the last forward and first backward LSTMs. The second layer encodes information about the document and is also a biLSTM that runs at the sentence-level. This biLSTM takes as input the sentence representation from the previous layer $\boldsymbol{e}(s_i)$ and outputs the hidden representation for each sentence s_i in the document as $\boldsymbol{h}_i^s$ (SentEncoder in Figure 2). We consider the output of the last forward LSTM over M sentences and first backward LSTM to be the final representation of the document $\boldsymbol{e}(D) = [\overrightarrow{\boldsymbol{h}}_M^s, \overleftarrow{\boldsymbol{h}}_1^s]$.

The encoder returns two output vectors, $\boldsymbol{d}_i^s = [\boldsymbol{e}(D), \boldsymbol{e}(s_i), \boldsymbol{h}_i^s]$ associated with each sentence s_i, and $\boldsymbol{d}_{ij}^w = [\boldsymbol{e}(D), \boldsymbol{e}(s_i), \boldsymbol{e}(w_{ij}), \boldsymbol{h}_i^s, \boldsymbol{h}_{ij}^w]$ for each word j at the specific state of the encoder i.

3.2 Decision Decoder

Given that our model operates both at the sentence-level and at the word-level, the decision decoder maintains two state LSTMs denoted by SentStates and WordStates as in Figure 3. For the sentence-level decoder sentences are selected and the state of the summary gets updated by SentStates. For the word-level, all compressed word representations in a sentence are pushed to the word-level layer. In the compressive decoder, words that get selected are pushed onto the WordStates, and once the decoder has reached the end of the sentence, it pushes the output representation of the last state onto the sentence-level layer for the next sentence.

Extractive Decoder The extractive decoder selects the sentences that should go to the summary. For each sentence s_i at time step i, the decoder takes a decision based on the encoder representation $\boldsymbol{d}_i^s$ and the state of the summary $\boldsymbol{o}_i^s$, computed as follows:

$$\boldsymbol{o}_i^s = \text{SentStates}(\{\boldsymbol{e}(c_k)\}_{k<i,z_k=1}).$$

where the $\boldsymbol{o}_i^s$ is modeled by an LSTM taking as input the already selected and compressed sentences comprising the summary so far $\{\boldsymbol{e}(c_k)\}_{k<i,z_k=1}$. This way, at each point in time, we have a representation of the summary given by the SentStates LSTM that encodes the state of summary generated so far, based on the past sentences already processed by the compressive decoder $\boldsymbol{e}(c_{i-1})$ (in WordStates).[2] The summary representation at step i ($\boldsymbol{o}_i^s$) is then used to determine whether to keep or not the current sentence in the summary ($z_i = 1$ or 0 respectively). The summarizer state subsumes information about the document, sentence and summary as:

$$\boldsymbol{p}_i = \tanh(W_E[\boldsymbol{d}_i^s; \boldsymbol{o}_i^s] + \boldsymbol{b}^s),$$

where W_E is a model parameter, $\boldsymbol{o}_i^s$ is the dynamic LSTM state, and $\boldsymbol{b}^s$ is a bias term.

This modeling decision is crucial in order to generate variable length summaries. It captures information about the sentences or words already present in the summary, helping in better understanding the "true" length of the summary given the document.

Finally, the summarizer state $\boldsymbol{p}_i$ is used to compute the probability of the action at time i as:

$$p(z_i \mid \boldsymbol{p}_i) = \frac{\exp\left(W_{z_i}\boldsymbol{p}_i + \boldsymbol{x}_{z_i}\right)}{\sum_{z' \in \{0,1\}} \exp\left(W_{z'}\boldsymbol{p}_i + \boldsymbol{x}_{z'}\right)},$$

---

[2] When using only the extractive model the summary state $\boldsymbol{o}_i^s$ is generated from an LSTM whose inputs correspond to the sentence encoded embeddings $\{\boldsymbol{e}(s_k)\}_{k<i,z_k=1}$ instead of the previously generated compressed representations $\{\boldsymbol{e}(c_k)\}_{k<i,z_k=1}$.

where W_z is a model parameter and $\boldsymbol{x}_z$ is a bias term for the summarizer action z.

We minimize the negative log-likelihood of the observed labels at training time (Dimitroff et al., 2013), where λ_0^s and λ_1^s represent the distribution of each class for the given sentences:[3]

$$L(\theta^s) = -\sum_{c\in\{0,1\}} \frac{\lambda_c^s}{\sum_{i=1}^{M} \mathbb{1}_{z_i=c}} \sum_{i,z_i=0} \log p(z_i|\boldsymbol{p}_i),$$

where $\mathbb{1}_{z_i=c}$ is the indicator function of class c and θ^s represents all the training parameters of the sentence encode/decoder. At test time, the model emits probability $p(z_i \mid \boldsymbol{p}_i)$, which is used as the soft prediction sequentially extracting the sentence i. We admit sentences when $p(z_i = 1 \mid \boldsymbol{p}_i) > 0.5$.

Compressive Decoder Our compressive decoder shares its architecture with the extractive decoder. The compressive layer is triggered every time a sentence is selected in the summary and is responsible for selecting the words within each selected sentence. In practice, WordStates LSTM (see Figure 3) is applied hierarchically after the sentence-level decoder, using as input the collected word embeddings so far:

$$\boldsymbol{o}_{ij}^w = \text{WordStates}(\{\boldsymbol{e}(w_{ik})\}_{k\leq j, y_{ik}=1}).$$

After making the selection decision for all words pertaining to a sentence, the final state of the WordStates, $\boldsymbol{e}(c_i) = \boldsymbol{o}_{iN}^w$ is fed back to SentStates of the extractive level decoder for the consecutive sentence, as depicted in Figure 3.

The word-level summarizer state representation depends on the encoding of words, document and sentence $\boldsymbol{d}_{ij}^w$, on the dynamic LSTM encoding for the summary based on the selected words (WordStates) $\boldsymbol{o}_{ij}^w$ and sentences (SentStates) $\boldsymbol{o}_i^s$:

$$\boldsymbol{q}_{ij} = \tanh(W_C[\boldsymbol{d}_{ij}^w; \boldsymbol{o}_i^s; \boldsymbol{o}_{ij}^w] + \boldsymbol{b}^w),$$

where W_C is a model parameter and $\boldsymbol{b}^w$ is a bias term. Each action at time step j is computed by

$$p(y_{ij} \mid \boldsymbol{q}_{ij}) = \frac{\exp\left(W_{y_{ij}}\boldsymbol{q}_{ij} + \boldsymbol{x}_{y_{ij}}\right)}{\sum_{y'\in\{0,1\}} \exp\left(W_{y'}\boldsymbol{q}_{ij} + \boldsymbol{x}_{y'}\right)},$$

with parameter $W_{y_{ij}}$ and bias $\boldsymbol{x}_{y_{ij}}$. The final loss for the compressive layer is

$$L(\theta^w) = \sum_{i=1}^{M} z_i \phi(i \mid \theta^w),$$

where θ^w represents the set of all the training parameters of the word-level encoder/decoder, $\phi(i)$ is the compressive layer loss over N words:

$$\phi(i \mid \theta^w) = -\sum_{c\in\{0,1\}} \frac{\lambda_c^w}{\sum_{i=1}^{M} \mathbb{1}_{y_{ij}=c}} \sum_{i,z_i=0} \log p(y_{ij}|\boldsymbol{q}_{ij}).$$

The total final loss is then given by the sum of the extractive and compressive counterparts, $L(\theta) = L(\theta^s) + L(\theta^w)$.

4 Experimental Setup

We mainly used the CNN/DailyMail corpus (Hermann et al., 2015) to evaluate our models. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 documents for CNN and 196,961/12,148/10,397 for DailyMail). To evaluate the flexibility of our model, we also evaluated our models on the Newsroom dataset (Grusky et al., 2018), which includes articles form a diverse collection of sources (38 publishers) with different summary style subsets: extractive (Ext.), mixed (Mixed) and abstractive (Abs.). We used the standard splits of Grusky et al. (2018) for training, validation, and testing (331,778/36,332/36,122 documents for Ext., 328,634/35,879/36,006 for Mixed and 332,554/36,380/36,522 for Abs.). We did not anonymize entities or lower case tokens.

4.1 Estimating Oracles

Datasets for training extractive summarization systems do not naturally contain sentence/word-level labels. Instead, they are typically accompanied by abstractive summaries from which extraction labels are extrapolated. We create extractive and compressive summaries prior to training using two types of *oracles*.

We used an *extractive oracle* to identify the set of sentences which collectively gives the highest ROUGE (Lin and Hovy, 2003) with respect to the gold summary (Narayan et al., 2018c).

To build a *compressive oracle*, we trained a supervised sentence labeling classifier, adapted from

---

[3] If $M - \sum_{i=1}^{M} z_i$=0 or $\sum_{i=1}^{M} z_i$=0, we simply consider the whole term to be zero. Here M represents the number of sentences in the document.

Oracle	R1	R2	RL
Extractive Oracle	54.67	30.37	50.81
Compressive Oracle	**57.12**	**32.59**	**53.27**

Table 1: Oracle scores obtained for the CNN and DailyMail testsets. We report ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) F1 scores.

the Transition-Based Chunking Model (Lample et al., 2016), to annotate spans in every sentence that can be dropped in the final summary. We used the publicly released set of 10,000 sentence-compression pairs from the Google sentence compression dataset (Filippova and Altun, 2013; Filippova et al., 2015) for training. After tagging all sentences in the CNN and DailyMail corpora using this compression model, we generated oracle compressive summaries based on the best average of ROUGE-1 (R1) and ROUGE-2 (R2) F_1 scores from the combination of all possible sentences and all removals of the marked compression chunks.

To verify the adequacy of our proposed oracles, we show in Table 1 a comparison of their scores. Our compressive oracle achieves much better scores than the extractive oracle, because of its capability to make summaries concise. Moreover, the linguistic quality of these oracles was preserved due to the tagging of the entire span by the sentence compressor trained on the sentence compression dataset.[4] We believe that our dataset with oracle compression labels will be of significant interest to the sentence compression and summarization community.

4.2 Training Parameters

The parameters for the loss at the sentence-level were λ_0^s=2 and λ_1^s=1 and at the word-level, λ_0^w=1 and λ_1^w=0.5. We used LSTMs with d = 512 for all hidden layers. We performed mini-batch negative log-likelihood training with a batch size of 2 documents for 5 training epochs.We observed the convergence of the model between the 2nd and the 3rd epochs. It took around 12 hrs on a single GTX 1080 GPU to train. We evaluated our model on the validation set after every 5,000 batches. We trained with Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001. Our system was implemented using DyNet (Neubig et al., 2017).

4.3 Model Evaluation

We evaluated summarization quality using F_1 ROUGE (Lin and Hovy, 2003). We report results in terms of unigram and bigram overlap (R1) and (R2) as a means of assessing informativeness, and the longest common subsequence (RL) as a means of assessing fluency.[5] In addition to ROUGE, which can be misleading when used as the only means to assess summaries (Schluter, 2017), we also conducted a question-answering based human evaluation to assess the informativeness of our summaries in their ability to preserve key information from the document (Narayan et al., 2018c).[6] First, questions are written using the gold summary, we then examined how many questions participants were able to answer by reading system summaries alone, without access to the article.[7] Figure 5 shows a set of candidate summaries along with questions used for this evaluation.

4.4 Model and Baselines

We evaluated our model EXCONSUMM in two settings: Extractive (selects sentences to assemble the summary) and Compressive (selects sentences and compresses them by removing unnecessary spans of words). We compared our models against a baseline (LEAD) that selects the first m leading sentences from each document,[8] three neural extractive models, and various abstractive models. For the extractive models, we used SUMMARUNNER (Nallapati et al., 2017), since it shares some similarity to our model, REFRESH (Narayan et al., 2018c) trained with reinforcement learning and LATENT (Zhang et al., 2018) a neural architecture that makes use of latent variable to avoid creating oracle summaries. We further compare against LATENT+COMPRESS (Zhang et al., 2018), an extension of the LATENT model that learns to map extracted sentences to final summaries using an attention-based seq2seq model (Rush et al., 2015). All models, unlike ours, extract a fixed number of sentences to assemble their summaries. For abstractive models, we compare against the state-of-the art models of POINTER+COVERAGE (See et al., 2017), ML+RL (Paulus et al., 2018), and Tan et al. (2017) among others.

---

[4] We show examples of both oracles in Appendix §A.1.

[5] We used `pyrouge` to compute the ROUGE scores. The parameters we used were "-a -c 95 -m -n 4 -w 1.2."

[6] We used the CNN/DailyMail QA test set of Narayan et al. (2018c) for evaluation. It includes 20 documents with a total of 71 manually written question-answer pairs.

[7] See Appendix §A.2 for more details.

[8] We follow Narayan et al. (2018c) and set m = 3 for CNN and 4 for DailyMail. We follow Grusky et al. (2018) and set m = 2 for Newsroom.

Models	CNN			DailyMail			Newsroom Ext.			Newsroom Mixed			Newsroom Abs.		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
LEAD	29.1	11.1	25.9	40.7	18.3	37.2	53.1	49.0	52.4	—	—	—	13.7	2.4	11.2
REFRESH	30.0	11.7	26.9	41.0	18.8	37.7	—	—	—	—	—	—	—	—	—
EXCONSUMM Extractive	**32.5**	12.6	28.5	**42.8**	**19.3**	**38.9**	**69.4**	**64.3**	**68.3**	**31.9**	**16.3**	26.9	**17.2**	**3.1**	13.6
EXCONSUMM Compressive	**32.5**	**12.7**	**29.2**	41.7	18.5	38.4	68.4	62.9	67.3	31.7	16.1	**27.0**	17.1	**3.1**	**14.1**
Pointer+Coverage $^\diamond$	—	—	—	—	—	—	39.1	28.0	36.2	25.5	11.0	21.1	14.7	2.3	11.4
Tan et al. (2017)*	30.3	9.8	20.0	—	—	—	—	—	—	—	—	—	—	—	—

Table 2: Results on the CNN, DailyMail and Newsroom test sets. We report ROUGE R1, R2 and RL F_1 scores. Extractive systems are in the first block, compressive in the second and abstractive in the third. We use — whenever results are not available. Models marked with * are not directly comparable to ours as they are based on an anonymized version of the dataset. The model marked with $^\diamond$ show here the results for the best configuration of See et al. (2017), referred to as Pointer-N in Grusky et al. (2018), which is trained on the whole Newsroom dataset.

Models	CNN+DailyMail		
	R1	R2	RL
LEAD	39.6	17.7	36.2
SUMMARUNNER*	39.6	16.2	35.3
REFRESH	40.0	18.2	36.6
LATENT	41.1	**18.8**	37.4
EXCONSUMM Extractive	**41.7**	18.6	37.8
LATENT+COMPRESS	36.7	15.4	34.3
EXCONSUMM Compressive	40.9	18.0	37.4
Pointer+Coverage	39.5	17.3	36.4
ML + RL*	39.9	15.8	36.9
Tan et al. (2017)*	38.1	13.9	34.0
Li et al. (2018)	39.0	17.1	35.7
Chen and Bansal (2018)	40.4	18.0	37.1
Hsu et al. (2018)	40.7	18.0	37.1
Pasunuru and Bansal (2018)	40.9	17.8	**38.5**
Gehrmann et al. (2018)	41.2	18.7	38.3

Table 3: Results for combined CNN/DailyMail test set.

5 Results

5.1 Automatic Evaluation

Table 2 and 3 show results for the evaluations on the CNN/DailyMail and Newsroom test sets.

Comparison with Extractive Systems. EXCONSUMM Compressive performs best on the CNN dataset and EXCONSUMM Extractive on the DailyMail dataset, probably due to the fact that the CNN dataset is less biased towards extractive methods than the DailyMail dataset (Narayan et al., 2018b). We report similar results on the Newsroom dataset. EXCONSUMM Compressive tends to perform better for mixed (Mixed) and abstractive (Abs.) subsets, while EXCONSUMM Extractive performs better for the extractive (Ext.) subset. Our experiments demonstrate that our compressive model tends to perform better on the dataset which promotes abstractive summaries.

We find that EXCONSUMM Extractive consistently performs better on all metrics when compared to any of the other extractive models, except for the single case where it is narrowly behind LATENT on R2 (18.6 vs 18.8) for the CNN/DailyMail combined test set. It even outperforms REFRESH, which is trained with reinforcement learning. We hypothesize that its superior performance stems from the ability to generate variable length summaries. REFRESH or LATENT, on the other hand, always produces a fixed length summary.

Comparison with Compressive System. EXCONSUMM Compressive reports superior performance compared to LATENT+COMPRESS (+4.2 for R1, +2.6 for R2 and +3.1 for RL). Our results demonstrate that our compressive system is more suitable for document summarization. It first selects sentences and then compresses them by removing irrelevant spans of words. It makes use of an advance oracle sentence compressor trained on a dedicated sentence compression dataset (Sec. 4.1). In contrast, LATENT+COMPRESS naively trains a sequence-to-sequence compressor to map a sentence in the document to a sentence in the summary.

Comparison with Abstractive Systems. Both EXCONSUMM Extractive and Compressive outperform most of the abstractive systems including Pointer+Coverage (See et al., 2017). When comparing with more recent methods (Pasunuru and Bansal, 2018; Gehrmann et al., 2018), our model has comparable performance.

Summary Versatility. We evaluate the ability of our model to generate variable length summaries. Table 4 show the Pearson correlation coefficient between the lengths of the human generated summaries against each unbounded model. Our compressive approach obtains the best results, with a Pearson correlation coefficient of 0.72 ($p < 0.001$).

Figure 4 also shows the distribution of words

	Bounded					Unbounded					
Models	Human QA		ROUGE			Human QA		ROUGE			Pearson
	score	rank	R1	R2	RL	score	rank	R1	R2	RL	r
LEAD	25.50	4rd	30.9	11.9	29.1	36.33	5th	31.6	13.5	29.3	0.40
Refresh	20.88	6th	37.4	17.3	34.8	**66.34**	1st	**43.8**	**25.8**	**41.6**	0.60
Latent	38.45	2nd	**38.9**	19.6	36.4	53.38	4th	40.7	22.0	38.1	-0.02
ExConSumm Extractive	36.34	3rd	38.4	18.5	35.9	54.93	3rd	40.8	21.0	38.2	0.68
ExConSumm Compressive	**39.44**	1ST	38.8	19.0	**37.0**	57.32	2nd	41.4	22.6	39.1	**0.72**
Pointer+Coverage	24.51	5th	38.4	**19.7**	36.7	28.73	6th	40.2	21.4	38.0	0.30

Table 4: QA evaluations: limited length (Bounded) and full length (Unbounded) summaries. We also show ROUGE scores for the summaries being evaluated. We report the Pearson correlation coefficient between the human and predicted summary lengths

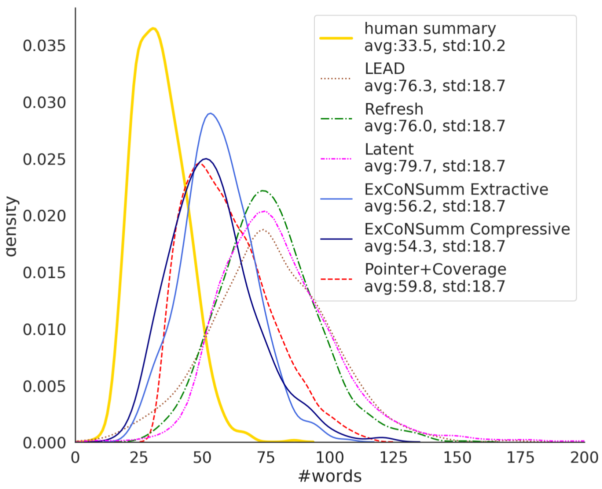


Figure 4: Word distribution in comparison with the human summaries for CNN dataset. Density curves show the length distributions of human authored and system produced summaries.

per summary for the models where predictions were available. Interestingly, both ExConSumm Extractive and Compressive follow the human distribution much better than other extractive systems (Lead, Refresh and Latent), since they are able to generate variable-length summaries depending on the input text. Our compressive model generates a word distribution much closer to the abstractive Pointer+Coverage model but achieves better compression ratio; the summaries generated by Pointer+Coverage contain 59.8 words, while those generated by ExConSumm Compressive have 54.3 words on average.

5.2 QA Evaluation

Table 4 shows results from our question answering based human evaluation. We elicited human judgements in two settings: the "Unbounded", where participants were shown the full system produced summaries; and the "Bounded", where participants were shown summaries that were limited to the same size as the gold summaries.

For the "Unbounded" setting, the output summaries produced by Refresh were able to answer most of the questions correctly, our Compressive and Extractive systems were placed at the 2nd and 3rd places respectively.[9]

We observed that our systems were able to produce more concise summaries than those produced by Refresh (avg. length in words: 76.0 for Refresh, 56.2 for ExConSumm Extractive and 54.3 for ExConSumm Compressive; see Figure 4). Refresh is prone to generating verbose summaries, consequently it has an advantage of accumulating more information. In the "Bounded" setting, we aim to reduce this unfair advantage. Scores are overall lower since the summary sizes are truncated to gold size. The ExConSumm Compressive summaries rank first and can answer 39.44% of questions correctly. ExConSumm Extractive retains its 3rd place answering 36.34% of questions correctly.[10] These results demonstrate that our models generate concise and informative summaries that correlate well with the human summary lengths.[11]

5.3 Summary State Representation

Next, we performed an ablation study to investigate the importance of the summary state representation $\boldsymbol{o}_i^s$ w.r.t. the quality of the overall sum-

---

[9] We carried out pairwise comparisons between all models to assess whether system differences are statistically significant. We found that there is no statistically significant difference between Refresh and ExConSumm Compressive. We use a one-way ANOVA with posthoc Tukey HSD tests with $p < 0.01$. The differences among Latent and both variants of ExConSumm, and between LEAD and Pointer+Coverage are also statistically insignificant. All other differences are statistically significant.

[10] The differences among both variants of ExConSumm and Latent, and among LEAD, Refresh and Pointer+Coverage are statistically insignificant. All other differences are statistically significant. We use a one-way ANOVA with posthoc Tukey HSD tests with $p < 0.01$.

[11] App. §A.2 shows more examples of our summaries.

LEAD
- (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
- Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."
- The Al-Malahem Media obituary characterized al-Rubaish as a religious scholar and combat commander.

Refresh
- (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
- Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."
- Al-Rubaish was once held by the U.S. government at its detention facility in Guantanamo Bay, Cuba.

Latent
- (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
- Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike." The Al-Malahem Media obituary characterized al-Rubaish as a religious scholar and combat commander.
- A Yemeni Defense Ministry official and two Yemeni national security officials not authorized to speak on record confirmed that al-Rubaish had been killed, but could not specify how he died.

ExConSumm Extractive
- (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
- Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

ExConSumm Compressive
- A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen. • Ibrahim al-Rubaish died in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

Pointer+Coverage
- Ibrahim al-Rubaish was among a number of detainees who sued the administration of then-president George W. Bush to challenge the legality of their confinement in Gitmo. • al-Rubaish was once held by the U.S. government at its detention facility in Guantanamo bay, Cuba.

GOLD
- AQAP says a "crusader airstrike" killed Ibrahim al-Rubaish
- Al-Rubaish was once detained by the United States in Guantanamo

---

Question-Answer Pairs
- Who said that an airstrike killed Ibrahim al-Rubaish? (AQAP) • What was the airstrike called? (crusader airstrike) • Where was Ibrahim al-Rubaish once detained? (Guantanamo)

Figure 5: Example output summaries on the CNN/DailyMail dataset, gold standard summary, and corresponding questions. The questions are manually written using the GOLD summary. The same ExConSumm summaries are shown in Figure 1, but the strike-through spans are now removed.

mary. We tested against a State averaging variant, where we replace $\boldsymbol{o}_i^s$ by a weighted average, analogous to Nallapati et al. (2017), $\boldsymbol{o}_i^{avg\ s} = \sum_{i=1}^{j-1} \boldsymbol{e}(s_i)p(z_i \mid \boldsymbol{p}_i^{avg})$, where $\boldsymbol{p}_i^{avg}$ has the same form as $\boldsymbol{p}_i$ but depends recursively on the previous summary state $\boldsymbol{o}_{i-1}^{avg\ s}$. Table 5 shows that using an LSTM state $\boldsymbol{o}_i^s$ to model the current sentences in the summary is very important. The other ablation study shows how learning to extract and compress in a disjoint approach (ExConSumm Ext+Comp oracle) performs against a joint learning approach (ExConSumm Compressive). We compared summaries generated from our best extractive model and compressed them with a compressive oracle. Our joint learning model achieves the best performance in all metrics compared with the other ablations, suggesting that joint learning and using a summary state representation is beneficial for summarization.

State	ROUGE R1	ROUGE R2	ROUGE RL
ExConSumm Extractive	**32.5**	**12.6**	**28.5**
State averaging	30.0	12.3	26.9
ExConSumm Compressive	**32.5**	**12.7**	**29.2**
ExConSumm Ext+Comp oracle	25.5	9.3	23.7

Table 5: Summary state ablation for the CNN dataset.

6 Conclusions

We developed ExConSumm, a novel summarization model to generate variable length extractive and compressive summaries. Experimental results show that the ability of our model to learn a dynamic representation of the summary produces summaries that are informative, concise, and correlate well with human generated summary lengths. Our model outperforms state-of-the-art extractive and most of abstractive systems on the CNN and DailyMail datasets, when evaluated automatically, and through human evaluation for the bounded scenario. We further obtain state-of-the-art results on Newsroom, a more abstractive summary dataset.

Acknowledgments

This work is supported by the EU H2020 SUMMA project (grant agreement N$^{\underline{o}}$ 688139), by Lisbon Regional Operational Programme (Lisboa 2020), under the Portugal 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within project INSIGHT (N$^{\underline{o}}$ 033869), by the European Research Council (ERC StG DeepSPIN 758969), and by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal).

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 196–206.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research (JAIR)*, 31:399–429.

Georgi Dimitroff, Laura Tolosi, Borislav Popov, and Georgi Georgiev. 2013. Weighted maximum likelihood loss as a convenient shortcut to optimizing the f-measure of maximum entropy classifiers. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 207–214, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Alexander Dlikman and Mark Last. 2016. Using machine learning methods and linguistic features in single-document extractive summarization. In *DMNLP@PKDD/ECML*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC 2003, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1998–2008, Berlin, Germany.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368. Association for Computational Linguistics.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *CoRR*, abs/1805.06266.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270. Association for Computational Linguistics.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 55–60.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *North American Chapter of the Association for Computational Linguistics: Workshop on Integer Linear Programming for NLP*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 297–304.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: a recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016a. Classify or select: Neural architectures for extractive document summarization. *CoRR*, abs/1611.04244.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B. Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*, pages 2020–2030, Melbourne, Australia.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018c. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1747–1759, New Orleans, Louisiana.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*, pages 41–45, Valencia, Spain.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449–1481.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, page 9.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.

News Article Teaser Tweets and How to Generate Them

Sanjeev Kumar Karn[1,2], **Mark Buckley**[2], **Ulli Waltinger**[2] **and Hinrich Schütze**[1]

[1]Center for Information and Language Processing (CIS), LMU Munich
[2]Machine Intelligence, Siemens CT, Munich, Germany
[1]`skarn@cis.lmu.de`
[2]`{sanjeev.kumar_karn,mark.buckley,ulli.waltinger}@siemens.com`

Abstract

In this work, we define the task of teaser generation and provide an evaluation benchmark and baseline systems for the process of generating teasers. A teaser is a short reading suggestion for an article that is illustrative and includes curiosity-arousing elements to entice potential readers to read particular news items. Teasers are one of the main vehicles for transmitting news to social media users. We compile a novel dataset of teasers by systematically accumulating tweets and selecting those that conform to the teaser definition. We have compared a number of neural abstractive architectures on the task of teaser generation and the overall best performing system is See et al. (2017)'s seq2seq with pointer network.

1 Introduction

A considerable number of people get their news in some digital format.[1] The trend has made many publishers and editors shift their focus to the web and experiment with new techniques to lure an Internet-savvy generation of readers to read their news stories. Therefore, there has been a noticeable increase in the sharing of short illustrative pieces of texts about the news on social media.

We define a ShortcutText as a short text (about 15 words or less) describing and pointing to a news article and whose purpose is to invite the recipient to read the article. A headline is a ShortcutText that optimizes the relevance of the story to its reader by including interesting and high news value content from the article (Dor, 2003). Clickbait is a pejorative term for web content whose main goal is to make a user click an adjoining link by exploiting the information gap. According to the definition, a principal part of the headline is an extract of the article, thereby creating an impression of the upcoming story. However, click-bait, a ShortcutText, contains mostly elements that create anticipation, thereby making a reader click on the link; however, the reader comes to regret their decision when the story does not match the click-bait's impression (Blom and Hansen, 2015). Thus, click-bait provides a false impression (non-bona fide) and contains insufficient information (highly abstractive).

	bona-fide	teasing	abstractive
headline	yes	no	low
clickbait	no	yes	high
teaser	yes	yes	high

Table 1: The table shows three categories of ShortcutTexts and their properties

We introduce the new concept of *teaser* and define it as a ShortcutText devised by fusing curiosity-arousing elements with interesting facts from the article in a manner that concurrently creates a valid impression of an upcoming story and a sense of incompleteness, which motivates the audience to read the article. A teaser is one of the main vehicles for transmitting news on social media. Table 2 shows some teasers from a popular newswire *The Wall Street Journal*.

We also introduce properties such as teasing, abstractive, and bona-fide, which not only differentiate teasers from other ShortcutTexts but also help in compiling a dataset for the study. Teasing indicates whether curiosity-arousing elements are included in the ShortcutText. Abstractive indicates whether a fair proportion of the ShortcutText is distilled out of the news article. Bona-fide answers whether the news story matches the impression created by the ShortcutText. Table 1 lists the common forms of the ShortcutTexts along with the presence or absence of the properties mentioned

---

[1] http://www.journalism.org/2008/07/21/the-influence-of-the-web/

Proceedings of NAACL-HLT 2019, pages 3967–3977
Minneapolis, Minnesota, June 2 - June 7, 2019.

Article	Global trade is in trouble, and investors dont seem to care. One of the ironies of the election of a fierce nationalist in the U.S. . . .
Headline	Steel Yourself for Trumps Anti-Trade Moves
Teaser	Investors don't seem worried about a trade war. Could tariffs by Trump start one?
Article	The U.S. Supreme Court on Monday partially revived President Donald Trumps executive order suspending travel from six countries . . .
Headline	High Court Says Travel Ban Not For Those With 'Bona Fide' Relationships
Teaser	In a 'bona fide' relationship? You can visit the U.S.
Article	Gan Liping pumped her bike across a busy street, racing to beat a crossing light before it turned red. She didnt make it. . . .
Headline	China's All-Seeing Surveillance State Is Reading Its Citizens' Faces
Teaser	China is monitoring its citizens very closely. Just ask jaywalkers.

Table 2: The table contains tuples of news articles and their ShortcutTexts: headline and teaser. These tuples are from a popular newswire, *The Wall Street Journal*.

above.

In this study, we focus on teasers shared on Twitter[2], a social media platform whose role as a news conduit is rapidly increasing. An indicative tweet is a Twitter post containing a link to an external web page that is primarily composed of text. The presence of the URL in an indicative tweet signals that it functions to help users decide whether to read the article, and the short length confirms it as a ShortcutText like a headline or teaser. Lloret and Palomar (2013) made an early attempt at generating indicative tweets using off-the-shelf extractive summarization models, and graded the generated texts as informative but uninteresting. Additionally, Sidhaye and Cheung (2015)'s analysis showed extractive summarization as an inappropriate method for generating such tweets as the overlaps between the tweets and the corresponding articles often are low. Our study shows that teasers, bona fide indicative tweets, do exhibit significant, though not complete, overlaps, and, therefore, are not appropriate for extractive but certainly for abstractive summarization.

Our contributions:

1) To the best of our knowledge, this is the first attempt to compare different types of ShortcutTexts associated with a news article. Furthermore, we introduce a novel concept of a teaser, an amalgamation of article content and curiosity-arousing elements, used for broadcasting news on social media by a news publisher.

2) We compiled a novel dataset to address the task of teaser generation. The dataset is a collection of news articles, ShortcutTexts (both teasers and headlines), and story-highlights. Unlike ShortcutText, a story-highlight is brief and includes self-contained sentences (about 25-40 words) that allow the recipient to gather information on news stories quickly. As all corpora based on news articles include only one of these short texts, our dataset provides the NLP community with a unique opportunity for a joint study of the generation of many short texts.

3) We propose techniques like unigram overlap and domain relevance score to establish abstractivity and teasingness in the teasers. We also apply these techniques to headlines and compare the results with teasers. The comparison shows teasers are more abstractive than headlines.

4) High abstractivity makes teaser generation a tougher task; however, we show seq2seq methods trained on such a corpus are quite effective. A comparison of different seq2seq methods for teaser generation shows a seq2seq combining two levels of vocabularies, source and corpus, is better than one using only the corpus level. Therefore, we set a strong baseline on the teaser generation task with a seq2seq model of See et al. (2017).

The remaining paper is structured as follows. In Section 2, we provide a detailed description of the data collection and analyses. In Section 3, we describe and discuss the experiments. In Section 4, we describe a user study of model-generated teasers. In Section 5, we discuss the related works. Section 6 concludes the study.

2 Teaser Dataset

Several linguistic patterns invoke curiosity, e.g., provocative questions and extremes for comparison. A retrieval of teasers from a social media platform using such patterns requires the formulation of a large number of complex rules as these patterns often involve many marker words and correspondingly many grammar rules. A computationally easy approach is to compile circulations from bona-fide agents involved in luring business on such media, and then filtering out those that don't comply with defined characteristics of a teaser; see Table 1. We followed the latter approach and chose Twitter to conduct our study.

2.1 Collection

We identified the official Twitter accounts of English-language news publications that had

---

[2] https://twitter.com/

tweeted a substantial number of times before the collection began; this removes a potential source of noise, namely indicative tweets by third-party accounts referencing the articles via their URL. See supplementary A.1 for the list of Twitter accounts. We downloaded each new tweet from the accounts via Twitter's live streaming API. We limited the collection to indicative tweets and extracted the article text and associated metadata from the webpage using a general-purpose HTML parser for news websites.[3] Overall, we collected approximately 1.4 million data items.

2.2 Analysis

We propose methods that evaluate teasingness and abstractivity in the teasers and verify them through analyses. We then combine those methods and devise a teaser recognition algorithm. Analyses are performed on lowercase, and stopwords-pruned texts.

2.2.1 Extractivity

For a given pair of strings, one is an extract of another if it is a substring of it. Teasers are abstractive, which we confirm by making sure that the ShortcutText is not an extract of article sentences. Additionally, a teaser of an article is designed differently than the headline; therefore, they must be independent of one other, i.e., non-extractive.

2.2.2 Abstractivity

Abstractivity, a principle characteristic of the teaser, implies that the teaser should exhibit content overlap with its source, but not a full overlap.

We rely on Sidhaye and Cheung (2015)'s method of computing the percentage match between two stemmed texts for grading abstractivity. We obtain unigrams of the first, X_1, and second text, X_2, using function $uni(X)$ and compute the percentage match using Eq. 1:

$$perc\_match(X_1, X_2) = \frac{|uni(X_1) \cap uni(X_2)|}{|uni(X_1)|} \tag{1}$$

Given a ShortcutText and article, initially, a sequence of texts is obtained by sliding a window of size p on the article sentences. Then, $perc\_match$ scores between the ShortcutText and sequence of texts are computed. A text with the highest score is selected as the prominent section for the ShortcutText in the article.

Article	Diabetes medication, such as insulin, lowers blood sugar levels and . . . . But experts have revealed a natural treatment for diabetes could be lurking in the garden. . . . Fig leaves have properties that can help diabetes . . . . An additional remedy . . .
headline	Diabetes treatment: Natural remedy made from fig leaves revealed
Teaser	**Would** you **Adam** and **Eve** it? Natural treatment for DIABETES could be **growing** in your garden

Table 3: ShortcutTexts and their non-overlaps (bold).

A full-overlap, i.e., $perc\_match$ of 1 is likely to be a case where the ShortcutText disseminates information of its prominent section. However, a non-overlap is very likely to be click-bait or noise. Thus, we filter out instances where the match score between a ShortcutText, potential teaser, and its prominent section is above 80% or below 20%. The intuition for the filtering is that the teasing words are likely to be absent from the prominent section, and an absence of a minimum of 2-3 words (often 20%) is the easiest way to ascertain this fact. Table 3 shows an example. Analogously, a presence of a minimum of 2-3 words from the source asserts that it is not click-bait or noise.

We use the sliding window size, p, of 5,[4] and filter the data instances where the $perc\_match$ between the tweet and prominent section is lower than 0.2 or greater than 0.8.

2.2.3 Teasingness

Apart from abstractivity, teasers include words and phrases that tease and are are embedded by authors who often draw on their vast knowledge of style and vocabulary to devise teasers. A commonly recognizable pattern among them is the inclusion of unusual and interesting words in a given context, e.g., words like *Adam* and *Eve* in the example of Table 3.

The Pareto principle or the law of the vital few, states that the 2,000 of the most frequently used words in a domain cover about 80% of the usual conversation texts (Nation, 2001; Newman, 2005). At first glance, filtering those abstractive ShortcutTexts that constitute only frequent words should intuitively prune uninteresting ones and save ones that are similar to the example in Table 3. However, a closer look at the pruned ShortcutTexts shows several interesting teasers with substrings comprised of out-of-place frequent-words, e.g., *Las Vegas gunman Stephen bought nearly %%*

---

[3] https://github.com/codelucas/newspaper/

[4] Most of the prominent information is supposedly within a few leading sentences in the news articles due to the inverted pyramid news writing style.

guns legally. But none of the purchases set off any red flags, with an interesting sentence fragment containing the phrase *red flags*. This suggests that the methodology that uses plain frequency of words is not sufficient for determining interesting information.

$$\begin{aligned} tf_{domain}(w,d) &= \frac{|\text{term } w \text{ in domain } d|}{|\text{terms in domain } d|} \\ idf_{domain}(w) &= log\, \frac{|\text{domains}|}{|\text{domains containing } w|} \\ dr(w,d) &= tf_{domain}(w,d) \times idf_{domain}(w) \end{aligned} \tag{2}$$

Thus, we look at unusualness at a level lower than the corpus. We rely on domain relevance (dr) (Schulder and Hovy, 2014), an adapted TF-IDF (term frequency inverse document frequency) metric that measures the impact of a word in a domain and, therefore, identifies unusual words in a specific domain, and is computed using Eq. 2.

A word is assigned a very low dr score if the word is either non-frequent in the domain and too frequent among other domains (unusualness) or non-frequent in all domains (rare); see Table 4. As a very low dr score corresponds to unusualness, a presence of very low dr values among the non-overlapping words of the ShortcutText suggest a high likelihood of it being a teaser, and therefore, we compile them as teasers. However, the filtering requires a threshold dr value that defines anything lower than it as a very low dr. Also, computing dr requires domain information of the text.

	$freq$_Out	¬ $freq$_Out
$freq$_IN	low	high
¬ $freq$_IN	very low	very low

Table 4: The table shows dr score range. IN and Out refer to in-domain and out-of-domain respectively.

Would	**Adam**	**Eve**	Natural	treatment
0.104	**0.0027**	**0.0025**	0.025	0.016
DIABETES	could	**growing**	garden	
0.005	0.105	0.022	0.01	

Table 5: Teaser words and their dr values. Non-overlaps are in bold blue.

Obtaining Domains

We make use of articles and their keywords to determine domains. Keywords are meta-information available for a subset of corpus instances. We rely on Doc2vec (Le and Mikolov, 2014) for obtaining the representations for the articles and cluster these representations by K-Means clustering (Hartigan and Wong, 1979).

We rely on elbow criterion and uniformity among keywords in the clusters to determine the number of clusters. The uniformity is validated by manual inspection of 100 most-frequent keywords. Clustering the corpus into eight domains resulted in the final abrupt decrease of the Sum of Squared Error (SSE) as well as uniformly distributed keyword sets. See Table 6 for domain-wise keywords and other statistics.

Selecting a Threshold

We use the domain information and compute dr values of potential teaser texts in the corpus. Table 5 shows nonstop words and dr scores for Table 3 example. Evidently, unusual words have very low dr scores (bold values).

To determine an appropriate threshold, we design an unsupervised methodology based on the Pareto principle. The cue remains the same, i.e., a right threshold will filter only the teasers, and the non-overlapping words in them are less likely to be frequent words.

Thus, we define a range of possible threshold values, and for each value, we compile a corpus of teasers where a non-overlapping word has dr below it. Meanwhile, we also compile sets of most-frequent words that cover 80% of the total word occurrences in all 8 domains (sizes ≈ 2000). Then, we determine the ratio of the teasers that have their non-overlapping words completely overlapping the frequent word sets. Finally, we select a value which has the least overlap as the threshold; see Figure 1. We chose 0.005 as it is the boundary below which there is no overlap. We apply this value to abstractive ShortcutTexts and obtain a teaser corpus.

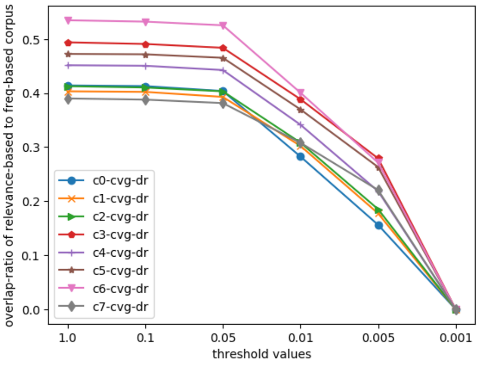


Figure 1: Overlap-ratio of frequency-based and threshold-based filtered teasers for domains (c#).

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7
Keywords	politics, UK, world, Brexit, Europe, UK Politics, Theresa May	Trump, United States, election, immigration, White House	culture, travel, home and garden, food and beverage	entertainment, celebrities, movies, concerts, Netflix	Europe, Russia, North Korea, Diplomacy, Conflicts	shooting, police, murder, killed, dead, fire, suspect, crash	Corporate, business, Company, automotive, Equities	Sport, Football, Premier League, NFL, Dallas Cowboys, Rugby
Avg. size (words)	763	842	526	838	886	651	791	741

Table 6: The table shows clusters of domains and corresponding frequent-keywords and average article size (words) in them.

2.3 Teaser Recognition Algorithm

We combine the above three methodologies and devise a teaser recognition algorithm; see Algorithm. 1.

We use notations like uppercase bold for a matrix, lowercase italic for a variable and uppercase italic for an array. A data instance in the corpus has an article A, headline H, tweet T, and domain d. An article, A, has a sequence of sentences, $S = \langle S_1, \ldots, S_{|A|} \rangle$, and each sentence, S_i, has a sequence of words, $\langle w_1, \ldots, w_{|S_i|} \rangle$. WINDOW takes a sequence of sentences, S, and returns a sequence of texts, Z, of size $\frac{|S|-p}{q} + 1$, where p and q are window size and sliding step respectively. The domain-wise dr values for words in the vocabulary, U, is stacked into a matrix, $\mathbf{D}$. IS_TEASER takes $\mathbf{D}$ and items of a data instance, and determines whether its tweet, T, is a teaser.

Algorithm 1 Teaser Recognition

```
1:  procedure IS_TEASER($A, H, T, d, \mathbf{D}$)
2:      if ($T$ in $H$) Or ($H$ in $T$) then
3:          return False                              ▷ sub-string match
4:      for $S$ in $A$ do
5:          if ($T$ in $S$) Or ($S$ in $T$) then
6:              return False                          ▷ sub-string match
7:      $Z \leftarrow$ WINDOW($A, p = 5, q = 1$)
8:      $V \leftarrow$ Array()
9:      for $i$ = 1 to $|Z|$ do
10:         $V$.ADD($perc\_match(Z[i], T)$)           ▷ see Eq. 1
11:     if $max(V) > 0.8$ Or $max(V) < 0.2$ then:
12:         return False                              ▷ abstractivity
13:     $\hat{Y} \leftarrow Z[max(V)]$                ▷ prominent section
14:     $T' \leftarrow \hat{Y} \backslash T$          ▷ non-overlap
15:     $L \leftarrow \mathbf{D}[d;T']$               ▷ indexing
16:     if any ($L < 0.005$) then
17:         return True                               ▷ teasingness
18:     return False
```

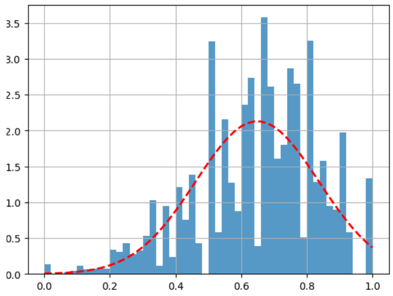
Overall, in Algorithm. 1, steps 2 to 6 checks Extractivity, steps 7 to 12 checks Abstractivity, and steps 13 to 17 checks Teasingness. Table 7 shows the percentage distribution of the total data points that are pruned by each of those analyses. Finally, we compile the remaining 23% data points, i.e., 330k as a teaser corpus.

Analysis		% pruned
Extractivity	wrt headline	37%
	wrt article	5%
Abstractivity		22%
Teasingness		13%

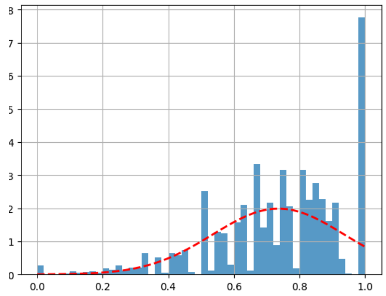
Table 7: The table shows different analyses performed in Algorithm. 1 and the corresponding approximate percentage of data points pruned using them. "wrt" = with respect to.

2.4 Comparing ShortcutTexts

The two ShortcutTexts, headline and teaser, have distinct conveyance mediums and therefore are designed differently, e.g., mean lengths of 10 and 14 respectively. However, abstractivity is also presumed for the headline. Therefore, we conduct additional overlap-based studies to understand the



(a) teasers ($t1$) and articles ($t2$).



(b) headlines ($t1$) and articles ($t2$)

Figure 2: Histogram of unigram overlaps obtained using Eq. 1. The histograms are normalized, i.e., the area under the curve ($\sum$bin-height×bin-width) sum up to one.

differences in the abstractive property between them. We compute and plot the distribution of the overlaps between teasers (T_1) and articles (T_2), and one between headlines (T_1) and articles (T_2); see Figure 2a and Figure 2b for respective plot. Clearly, compared to the teaser, headline distribution is left-skewed (mean 74% and std 20%), and thereby implies that headlines have a lesser abstractive value than teasers.

Further, a review of a few instances of headline-article instances with lesser than 60% overlap reveals cases of noisy headlines or HTML-parse failures; therefore, in a typical scenario a headline with a size of 10 words takes nearly all of its content (≈80%) from the source while a teaser of size 14 has sufficient non-extractive contents (≈32%). See Table 3 for an example.

3 Experiments

3.1 Models

We experiment with two state-of-the-art neural abstractive summarization techniques, attentive seq2seq (Bahdanau et al., 2014) and pointer seq2seq (See et al., 2017), for teaser generation. Attentive seq2seq learns to generate a target with words from a fixed vocabulary, while pointer seq2seq uses a flexible vocabulary, which is augmented with words from the source delivered through the pointer network. We refer to the individual papers for further details.

Evaluation Metrics: Studies on text-summarization evaluate their system using Rouge; therefore, we report Rouge-1 (unigram), Rouge-2 (bigram), and Rouge-L (longest-common substring) as the quantitative evaluation of models on our corpus.

Parameters: We initialized all weights, including word embeddings, with a random uniform distribution with mean 0 and standard deviation 0.1. The embedding vectors are of dimension 100. All hidden states of encoder and decoder in the seq2seq models are set to dimension 200. We pad short sequences with a special symbol $\langle PAD \rangle$. We use Adam with initial learning rate .0007 and batch size 32 for training. Texts are lowercased and numbers are replaced by the special symbol %. The token length for the source is limited to 100 and target sequence to 25. The teaser baseline experiments and headline generation use vocabulary size of 20000.

	Rouge-1	Rouge-2	Rouge-L
ABS	29.55	11.32	26.42
ABS+	29.76	11.88	26.96
RAS-Elman	33.78	**15.97**	31.15
Nallapati et al.	32.67	15.59	30.64
seq2seq	31.21	12.96	28.87
seq2seq_point	**34.81**	15.59	**32.05**

Table 8: Rouge scores on the standard task of Headline Generation (Gigaword). seq2seq and seq2seq_point are reimplementations of Bahdanau et al. (2014) and See et al. (2017) respectively.

3.2 Baseline Setting

As we reimplemented (Bahdanau et al., 2014) and (See et al., 2017) models, we initially evaluate them on a standard task of headline generation.[5] We use popular headline generation corpus, Gigaword (Napoles et al., 2012), with 3.8M training examples. We fetched the test set from Rush et al. (2015) and report the results on it. The results are compared with the state-of-the-art headline generation methods like Nallapati et al. (Nallapati et al., 2016), ABS (Rush et al., 2015), ABS+ (Rush et al., 2015), and RAS-Elman (Chopra et al., 2016). Since our aim for this experiment is to demonstrate the strength of the models, we limit the model parameters to the extent that we produce comparable results in less computation time. Table 8 compares performances of seq2seq and seq2seq_pointer models with other state-of-the-art methods. The results indicate that the implementations have performance competitive with other state-of-the-art methods.

	Validation		
	Rouge-1	Rouge-2	Rouge-L
seq2seq	15.77	03.52	13.53
seq2seq_point	**21.57**	**07.03**	**18.64**
	Test		
seq2seq	15.26	03.38	13.15
seq2seq_point	**21.05**	**07.11**	**18.49**

Table 9: Rouge F1 scores for seq2seq model and seq2seq_point models on the teaser task.

These models are then trained and evaluated on the teaser corpus obtained using Algorithm. 1 that initially has 330k instances. We then sample 255k instances that have all associated short texts in them. The sampled corpus is split into

---

[5] codes for collection, analyses and experiments: https://github.com/sanjeevkrn/teaser_collect.git and https://github.com/sanjeevkrn/teaser_generate.git

non-overlapping 250k, 2k and 2k sets for training, validation, and testing, respectively. The split is constructed such that training, validation and test sets have equal representation of all eight domains. Any instances that describe events that were also described in training are removed from validation and test sets; thus, instances encountered in validation / test are quite distinct from instances encountered in training. Models were selected based on their performance on the validation set. Table 9 shows the performance comparison. Clearly, seq2seq_point performs better than seq2seq due to the boost in the recall gained by copying source words through the pointer network.

	Rouge-1	Rouge-2	Rouge-L
	Teaser		
seq2seq	15.26	03.38	13.15
seq2seq_point	**21.05**	**07.11**	**18.49**
	Headline		
seq2seq	18.52	05.34	16.74
seq2seq_point	**23.83**	**08.73**	**21.68**
	Highlights		
seq2seq	31.18	17.57	27.30
seq2seq_point	**35.92**	**22.44**	**31.53**

Table 10: Rouge F1 scores for seq2seq model and seq2seq_point models on the teaser, headline and highlights generation task.

Additionally, models are also trained and evaluated on the other short texts that are available in the novel corpus: headlines (also a ShortcutText) and story-highlights. All the model parameters remain the same except the generation size, which depends on the short text average size, e.g., 35 for highlights. Table 10 compares the performance on the test data. Clearly, seq2seq_point performs better than seq2seq for all the types of short texts. Additionally, the change in the rouge scores with the change of dataset, i.e., Teaser<Headline<Highlights, also corresponds to the level of distillation of source information in them.

Table 11 shows an example of a data instance in the corpus and seq2seq_point model generations. Among generations, only headline and teaser have non-overlapping words; however, the headline non-overlap, *says*, is a frequent word with a high dr (0.11) while the teaser non-overlap, *catch*, is a domain-wise non-frequent one, and therefore, has a very low dr (0.006).

Further, the teaser is the most detached from the core news information among the three generations, while still being relevant. The generated highlight is extractive, and this is a reason for relatively high Rouge scores for highlights (see Table 10). Rouge is an overlap-based measure and, therefore, is inclined towards extractive datasets.

Article	Millions of disease carrying mosquitoes are to be freed in a well-meaning bid . . .. The lab-grown versions are infected with a disease which prevents natural mosquito . . . . But some activists fear the disease could transfer to humans ultimately making all human males sterile . . . . Despite claims it is safe for humans , there are also some concerns . . . rendering humans unable to breed . . .
	Ground-Truth
Headline	**SHOCK** CLAIM: Lab created super-mosquitos released into wild could 'make all men infertile'
Highlight	A **NEW** lab-designed mosquito being released into the wild could **end** the human race by making men sterile, it was claimed **today**.
Teaser	**PLAYING GOD**? Millions of lab-grown diseased mosquitoes to be released into **wild**
	Generated
Headline	millions of mosquitoes could be freed , study **says**
Highlight	millions of disease carrying mosquitoes are to be freed in a well-meaning bid to decimate natural populations of the malaria-spreading insects .
Teaser	activists fear the disease could be freed - but there 's a **catch**

Table 11: seq2seq_pointer generated examples. Non-overlapping words are in bold blue. More examples in supplementary A.2.

3.3 Impact of very low dr

We performed additional experiments to study the impact that can be generated using the domain relevance (dr). All the settings are kept intact as in Section 3.2 except the training corpus; this is changed by increasing the proportion of very low dr (<0.005) terms in the teasers. New models are trained using equal size training instances sampled out of the revised corpora.

A bucketing of very low dr percentages into [0%, 25%), [25%, 35%), [35%, 45%), [45%, 55%) and [55%, 100%) divides the corpus into approximately equal sizes. Also, the mean and standard deviation of teaser-article overlap ratio is nearly equal in all the buckets, i.e., 0.559±0.148, 0.559±0.146, 0.564±0.146, 0.566±0.142, 0.566±0.146, respectively. Thus, the range of buckets corresponds to a range in the percentage of uncommon words. We evaluate the precision and recall of the models. Recall ($|overlap|/|ground\text{-}truth|$) estimates the model capacity in recovering the ground-truth content, while precision ($|overlap|/|generation|$) estimates the relevancy in the generation.

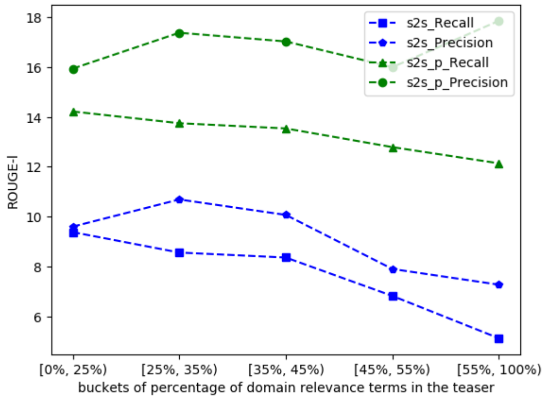


Figure 3: Variation in ROUGE-l on increasing the proportion of domain-relevant terms in the teasers. Models trained on 40k sampled instances.

As shown in Figure 3, the test recall for both models decreases with the increase in uncommon words in their training. An increase in the proportion of uncommon words makes the models also generate uncommon words, which are not likely to match the ground-truth, thereby reducing the recall. However, in extreme cases, i.e., [45%, 100%), not only training teasers get slightly shorter but also a relatively large proportion of out-of-vocabulary (UNK) is introduced in them, and thereby in the generations. The UNK appears for novel informative words, which are rare words with a very low *dr* as well (see Table 4). Unlike seq2seq, seq2seq_pointer recovers those from the source using pointer network and thus doesn't suffer an abrupt drop in the scores.

Further, the precision scores in extreme cases have a slightly different trend than recall scores, and this is due to shorter generations, which supports precision, but is irrelevant for recall.

4 Human Evaluation

The quantitative evaluations show that state-of-the-art models perform moderately on the novel task. This is mostly due to deficiencies of Rouge, which fails to reward heterogeneous contents. We took a closer look at some of the generated examples, see Table 12, and observed frequent cases where the generation suffered from the typical seq2seq issues, e.g., repetition of words; however, there are also cases where generation is more distinctive than ground-truth and is well formed too. Thus, we carried out a small user study to understand the quality of the generated teasers; however, we only selected non-repeating and non-UNK generations to anonymize the source. The participants in the user study are undergraduate or graduate students with some computer science background and familiarity with social media platforms. Additionally, all the participants have used or have been using twitter.

- pres . trump lashed out on twitter at the hosts of “ msnbcs morning ”
- migration agency says more than %% people drowned and presumed dead in myanmar to bangladesh
- computer glitch led to google to be dramatically undervalued this morning
- alt-right activist jason kessler says he was swarmed by a group of charlottesville
- of identical triplets who beat the incredible odds of %%% million to survive
- singer and guitar player who declined to appear on britain ’s got talent

Table 12: The table shows seq2seq_point generated teasers used in the survey-based study. More examples in supplementary A.2.

	On Twitter		Stimulating	
	Mean	Std	Mean	Std
ground-truth	0.660	0.064	0.621	0.079
seq2seq_point teaser	0.588	0.078	0.559	0.089
baseline	0.476	0.127	0.501	0.111

Table 13: The table shows the mean values and standard deviations of the likelihood of being social-media text and stimulating for users to read. Baseline = lead sentences

We assembled a set of texts by randomly sampling 40 seq2seq_point teasers, 40 ground-truth teasers, and 40 lead sentences (baseline), and also established equal representation of the domains. We then assigned 72 sentences (3 per domain per category) to ten participants and asked them to rate texts for two questions: 1) How likely is it that the text is shared on Twitter for a news story by a news organization? and 2) How likely is it that the text makes a reader want to read the story? The first question helps us recognize the participant's understanding of the teasers, as an informed reader will rate a ground-truth significantly higher than the baseline, and 8 of them recognized it correctly, and their ratings are selected for the evaluation. The second question provides a cue as to the model capacity in generating teasing texts by learning interesting aspects present in the teaser corpus.

The annotators rated samples on a scale of 1 to

5; however, we normalized the ratings to avoid the influence of annotators having different rating personalities. The results, summarized in Table 13, show that the human written teasers are most likely to be recognized as social media texts due to their style, which is distinct from the lead sentence; the model trained on such teasers closely follows it. Similarly, human written teasers are good at stimulating readers to read a story compared to the lead sentence and the generated teasers.

5 Related Work

There are two kinds of summarization: abstractive and extractive. In abstractive summarization, the model utilizes a corpus-level vocabulary and generates novel sentences as the summary, while extractive models extract or rearrange the source words as the summary. Abstractive models based on neural sequence-to-sequence (seq2seq) (Rush et al., 2015) proved to generate summaries with higher Rouge scores than the feature-based abstractive models. The integration of attention into seq2seq (Bahdanau et al., 2014) led to further advancement of abstractive summarization (Nallapati et al., 2016; Chopra et al., 2016; See et al., 2017).

There are studies utilizing cross-media correlation like coupling newswire with microblogs; however, most of them involve improving tasks on newswire by utilizing complementary information from microblogs, e.g., improving news article summarization using tweets (Gao et al., 2012; Wei and Gao, 2014), generating event summaries through comments (Wang et al., 2015), etc. There is very limited work on using newswire and generating microblogs, e.g., article tweet generation (Lloret and Palomar, 2013) and indicative tweet generation (Sidhaye and Cheung, 2015). Lloret and Palomar (2013) observed that off-the-shelf extractive models produce summaries that have high quantitative scores, but that are not interesting enough. Similarly, Sidhaye and Cheung (2015)'s analysis of indicative tweets shows the narrow overlap between such tweets and their source limits the application of an extractive method for generating them. Our controlled compilation of such tweets shows a mean percentage match of 68.3% (std: 16%) with its source. These analyses strongly suggest that indicative tweets are not regular information-disseminating short texts. Also, the mixed nature of such texts suggests an abstractive, rather than extractive study.

Most abstractive summarization systems use a popular dataset, CNN/DailyMail(Napoles et al., 2012), that includes news articles and story highlights to train and test their performance. However, story highlights are brief and self-contained sentences (about 25-40 words) that allow the recipient to quickly gather information on news stories; it is largely extractive (Woodsend and Lapata, 2010). Our novel corpus includes not only extractive short texts (i.e., story-highlights) and nearly extractive (i.e., headlines), but also very abstractive teasers, and therefore is a challenging and more appropriate dataset to measure abstractive systems.

6 Conclusion

We defined a novel concept of a teaser, a Shortcut-Text amalgamating interesting facts from the news article and teasing elements. We compiled a novel dataset that includes all of the short texts that are associated with news articles. We identified properties like abstractive, teasing, and bona-fide that assist in comparing a teaser with the other forms of short texts. We illustrated techniques to control these properties in teasers and verified their impact through experiments. An overlap-based comparative study of headlines and teasers shows teasers as abstractive while headlines as nearly extractive. Thus, we performed neural abstractive summarization studies on teasers and set a strong benchmark on the novel task of teaser generation.

Acknowledgments

We thank Siemens CT members and the anonymous reviewers for valuable feedback. This research was supported by Bundeswirtschaftsministerium (`bmwi.de`), grant 01MD15010A (Smart Data Web).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jonas Nygaard Blom and Kenneth Reinecke Hansen. 2015. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, 76:87 – 100.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 93–98. The Association for Computational Linguistics.

Daniel Dor. 2003. On newspaper headlines as relevance optimizers. *Journal of Pragmatics*, 35(5):695 – 721.

Wei Gao, Peng Li, and Kareem Darwish. 2012. Joint topic modeling for event summarization across news and social media streams. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1173–1182. ACM.

John A Hartigan and Manchek A Wong. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1188–1196, Bejing, China. PMLR.

Elena Lloret and Manuel Palomar. 2013. Towards automatic tweet generation: A comparative study from the text summarization perspective in the journalism genre. *Expert Syst. Appl.*, 40(16):6624–6630.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

I. S. P. Nation. 2001. *Learning Vocabulary in Another Language*. Cambridge Applied Linguistics. Cambridge University Press.

M. E. J. Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46:323–351.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389. The Association for Computational Linguistics.

Marc Schulder and Eduard Hovy. 2014. Metaphor detection through term relevance. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 18–26, Baltimore, MD. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Priya Sidhaye and Jackie Chi Kit Cheung. 2015. Indicative tweet generation: An extractive summarization problem? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 138–147, Lisbon, Portugal. Association for Computational Linguistics.

Lu Wang, Claire Cardie, and Galen Marchetti. 2015. Socially-informed timeline generation for complex events. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1055–1065. Association for Computational Linguistics.

Zhongyu Wei and Wei Gao. 2014. Utilizing microblogs for automatic news highlights extraction. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 872–883. ACL.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 565–574, Uppsala, Sweden. Association for Computational Linguistics.

A Supplementary

A.1 List of Twitter accounts

The following is the list of Twitter accounts from which data was collected.

Account ID	Account name
759251	CNN
807095	nytimes
35773039	theatlantic
14677919	newyorker
14511951	HuffingtonPost
1367531	FoxNews
28785486	ABC
14173315	NBCNews
2467791	washingtonpost
14293310	TIME

2884771	Newsweek
15754281	USATODAY
16273831	VOANews
3108351	WSJ
14192680	NOLAnews
15012486	CBSNews
12811952	Suntimes
14304462	TB_Times
8940342	HoustonChron
16664681	latimes
14221917	phillydotcom
14179819	njdotcom
15679641	dallasnews
4170491	ajc
6577642	usnews
1652541	reuters
9763482	nydailynews
17469289	nypost
12811952	suntimes
7313362	chicagotribune
8861182	newsday
17820493	ocregister
11877492	starledger
14267944	clevelanddotcom
14495726	phillyinquirer
17348525	startribune
87818409	guardian
15084853	IrishTimes
34655603	thesun
15438913	mailonline
111556423	dailymailuk
380285402	dailymail
5988062	theeconomist
17680050	thescotsman
16973333	independent
17895820	daily_express
4970411	ajenglish

Table 14: To access the Twitter page of an Account name X, use URL `https://twitter.com/X` and to access the Twitter page of an Account Id X, use URL `https://twitter.com/intent/user?user_id=X`. The script to download tweets from the above accounts is available in `https://github.com/sanjeevkrn/teaser_collect.git`.

A.2 Results

The following table shows examples that include input news articles and short text outputs (headline, highlight, and Teaser) both ground-truths and model generations.

Article	Sir Robert Fellowes , the Queen 's private secretary , was on the verge of making the extraordinary request . . .. But he was persuaded to back off by fellow courtiers and the party went ahead as planned putting Camilla . . . . a visible and acknowledged part of the Prince 's life . A new book , The Duchess : The Untold Story by Penny Junor , makes sensational claims about Prince Charles . . . . furious about the birthday party even though by this stage she was fairly relaxed about Charles 's relationship . . .
	Ground-Truth
Headline	Princess Diana latest: Queen aide planned to end Charles affair with Camilla amid rage
Highlight	PRINCE Charles faced being told by the Queen to end his relationship with Camilla after Princess Diana erupted with fury over a lavish party for the Duchess-to-be , it is claimed .
Teaser	**Royal intervention** threatened Charles and Camilla's affair after Diana's **fury** at **posh** bash
	Generated
Headline	duchess of the queen 's private secretary robert fellowes
Highlight	sir robert fellowes , the queen 's private secretary , was on the verge of making the extraordinary request of her majesty .
Teaser	penny junor **reveals** why she was on the **brink** of making the queen 's life
Article	The top Democrat on the Senate Finance Committee asked the Trump administration on Thursday to turn over the names of visitors . . .. That investigation found that members of the golf clubs Trump visited most often as president . . . . Membership lists at Trump's private clubs are secret. USA TODAY found the names of about 4,500 members by reviewing . . . . In a letter to the Department of Homeland Security's Acting Secretary Elaine Duke, Wyden said USA TODAY's examination . . .
	Ground-Truth
Headline	Senator seeks visitor logs, golf partners
Highlight	An investigation by USA TODAY prompts a senior Democratic senator to seek visitor logs at Trump clubs and names of his golfing partners .
Teaser	**Citing** USA TODAY's investigation, a top **Sen.** Democrat **seeks** visitor **logs** to Trump's golf courses & golfing partners.
	Generated
Headline	trump [UNK] to turn over trump 's private clubs
Highlight	the top democrat on the senate finance committee asked the trump administration .
Teaser	the top democrat on trump 's golf club : “ it ' s a **lot** of **money** , but it ' s not **going** to

Table 15: The table shows input articles, ground-truth short texts, and seq2seq_pointer generated short texts. Non-overlapping words between short texts and Articles are in bold blue.

Cross-referencing using Fine-grained Topic Modeling

Jeffrey Lund, Piper Armstrong, Wilson Fearn, Stephen Cowley, Emily Hales, Kevin Seppi
Computer Science Department
Brigham Young University
{jefflund, piper.armstrong, wfearn,
scowley4, emilyhales, kseppi}@byu.edu

Abstract

Cross-referencing, which links passages of text to other related passages, can be a valuable study aid for facilitating comprehension of a text. However, cross-referencing requires first, a comprehensive thematic knowledge of the entire corpus, and second, a focused search through the corpus specifically to find such useful connections. Due to this, cross-reference resources are prohibitively expensive and exist only for the most well-studied texts (e.g. religious texts). We develop a topic-based system for automatically producing candidate cross-references which can be easily verified by human annotators. Our system utilizes fine-grained topic modeling with thousands of highly nuanced and specific topics to identify verse pairs which are topically related. We demonstrate that our system can be cost effective compared to having annotators acquire the expertise necessary to produce cross-reference resources unaided.

1 Introduction

Cross-references are references within or between bodies of text that can help elaborate upon or clarify that text. They can be a useful tool for deep understanding and can also be used to analyze the relational structure of a text. In contrast to a word concordance which simply shows passages which share a common keyword, cross-references often include links which do not necessarily share the same keywords, but are still related topically. The existence of a thorough and complete cross-reference resource can facilitate better scholarship of a text and help readers to quickly find clarifying information or see repeated themes throughout a text.

Compared to language tasks such as part-of-speech tagging, producing cross-reference annotations is much more labor intensive. To produce cross-references, annotators must become intimately familiar with a text in order to note that a particular passage is related to another passage they happen to recall. This level of familiarity and expertise with a particular text typically requires the annotator to spend a great deal of time studying and reading. Possibly for this reason, expansive cross-references have only been produced for the most well-studied texts, such as the Bible. Other texts, such as academic textbooks, may include indices or other similar references, but these tend to be sparse, focusing on a small number of keywords rather than linking each individual passage with other relevant passages.

The process of creating such a resource can be expensive and time consuming. For example, the Bible[1] published by The Church of Jesus Christ of Latter-day Saints includes numerous cross-references and topic-based categories. These cross-references took hundreds of volunteers thousands of hours over seven years to produce (Anderson, 1979). This process involved collecting more than 19,900 manually curated entries from volunteers, and then editing and refining those references with a small committee of experts down to a final cross-reference database containing 12,475 entries.

Cross-referencing grows quadratically with the size of the data because for each of n passages, there are $n-1$ possibly related passages, yielding $\mathcal{O}(n^2)$ potential pairs. This differs from tasks such as part-of-speech tagging where annotators can tag individual sentences in isolation ($\mathcal{O}(n)$). We can, however, evaluate pairs in isolation. Therefore, our approach is to produce a system which utilizes fine-grained topic modeling in order to dramatically lower the cost of producing a cross-reference resource for new texts. We do not expect that such

---

[1] `https://www.lds.org/scriptures/bible`

Proceedings of NAACL-HLT 2019, pages 3978–3987
Minneapolis, Minnesota, June 2 - June 7, 2019.

a system will produce only–or even primarily–valid cross-references, but we hope that the system could be accurate enough to allow annotators to simply review the proposed cross-references and reduce the search cost.

2 Methodology

In this section, we describe our experimental setup and how we approach the problem of automatic cross-reference generation.

2.1 Cross-reference Datasets

While exact statistics are impossible to obtain, the number of printed copies of the Bible is estimated to be more than 5 billion (Guiness). It is one of the most well-studied texts in existence, and one of the few texts in the world with extensive cross-reference data. We utilize the English Standard Version of the Holy Bible (Crossway, 2001) in order to validate our method. We use this specific translation of the Bible because it is used on `openbible.info`. While our work focuses on this specific religious text out of necessity, it can also be applied to other texts, including literary classics and collections of historical documents.

To aid any readers who are unfamiliar with this religious text, we note that the term 'verse' refers to a short division of a chapter. The entirety of the text is divided up into books. Typically, individual verses are referenced by the name of the book, the chapter number, and the verse number, e.g., Isaiah 25:8. For convenience, each referenced verse in electronic versions of this paper is a hyperlink to `openbible.info`, showing the verse along with associated cross-references. For the sake of narrowing our focus, our efforts in cross-referencing the Bible will focus on finding topically related verses, though other work could potentially link larger passages, such as entire chapters or passages spanning multiple verses.

As a ground truth for cross-references, we utilize two sources. The first is the "Treasury of Scripture Knowledge, Enhanced" (Morton, 2010) (an extended version of the original "Treasury of Scripture Knowledge" (Torry, 2002)), which includes 670,796 cross-references between the 31,085 verses of the Bible. To our knowledge, this is the most exhaustive resource of human curated cross-references for the Bible to date. We will denote this cross-reference dataset as *TSKE*.

The second source of ground truth cross-references is a dataset from `openbible.info`. This dataset was seeded with various public domain cross-reference data, including the Treasury of Scripture Knowledge. As shown in Figure 1, users search for a verse they are interested in and can then vote on whether they found a particular cross-reference to be helpful or not. With each helpful or not helpful vote counting as $+1$ and -1 respectively, the dataset includes the net result of the votes for each included cross-reference.

Thus we can filter the dataset of cross-references based on how helpful each verse was rated to be. Counting only those cross-references which have a non-negative vote total, this dataset contains 344,441 cross-references. In figures, we denote this subset of the `openbible.net` cross-reference dataset as *OpenBible+0*. We also use the subset of cross-references which received a net total of at least 5 helpful votes. This subset, denoted as *OpenBible+5*, has 50,098 cross-references.

We do note however that the voting data has some skewness in the number of votes for each cross-reference. The overwhelming majority of cross-references received fewer than five total votes for or against the reference. A small number of verses, including both popular verses as well as verses which happen to come from the very beginning of the Bible, have received hundreds of votes.

2.2 Baselines for Automated Cross-reference Generation

These cross-reference datasets were produced at a tremendous cost in time and human effort. To the best of our knowledge, efforts at automating this process are limited and have not received much attention in computer science literature. That said, a reasonable baseline for automated efforts is a simple word-based concordance, which lists words along with references to where the words occur in the text[2].

Using the TSKE as the ground truth for cross-references, this simple baseline will recover roughly 65% of the cross-references. For example, as shown in Figure 1 and assuming that stemming is performed, the verse Isaiah 25:8 would be properly linked to 1 Corinthians 15:54 due to the two verses sharing the terms 'death' and 'swallow'. On the other hand, verses such as Hosea 13:14 or 1 Corinthians 15:55 which reference the

---

[2] For an example of such a concordance for the King James Version of the Bible, see Strong's Concordance (Strong, 1890).

‘sting of death’ should not be linked to verses such as Revelation 9:10, which references the sting of a scorpion. For this reason, roughly 99% of cross-references found using word-based concordance are spurious according to the TSKE, making this baseline less useful as a cross-referencing resource.[3] We refer to this baseline as *word match*.

As a slightly stronger baseline, we also consider a topical concordance in which verses assigned to the same topic by some topic model are considered to be linked. We refer to this baseline as *topic match*. For example, suppose that a topic model includes a topic which gives high probability to terms such as ‘death’, ‘swallow’, ‘victory’ and ‘sting’. Assuming that such a model would assign the previously mentioned verse to this topic, then verses such as Isaiah 25:8, Hosea 13:14, and 1 Corinthians 14:54 would be linked, but Revelation 9:10 which uses the term ‘sting’ in a different context (i.e., the sting of a scorpion) would not be linked.

We can further increase the precision of this baseline by only linking references which share a topic and a word, although this does come at the cost of recall. We refer to this final baseline method as *topic-word match*.

2.3 Topic-based Cross-referencing

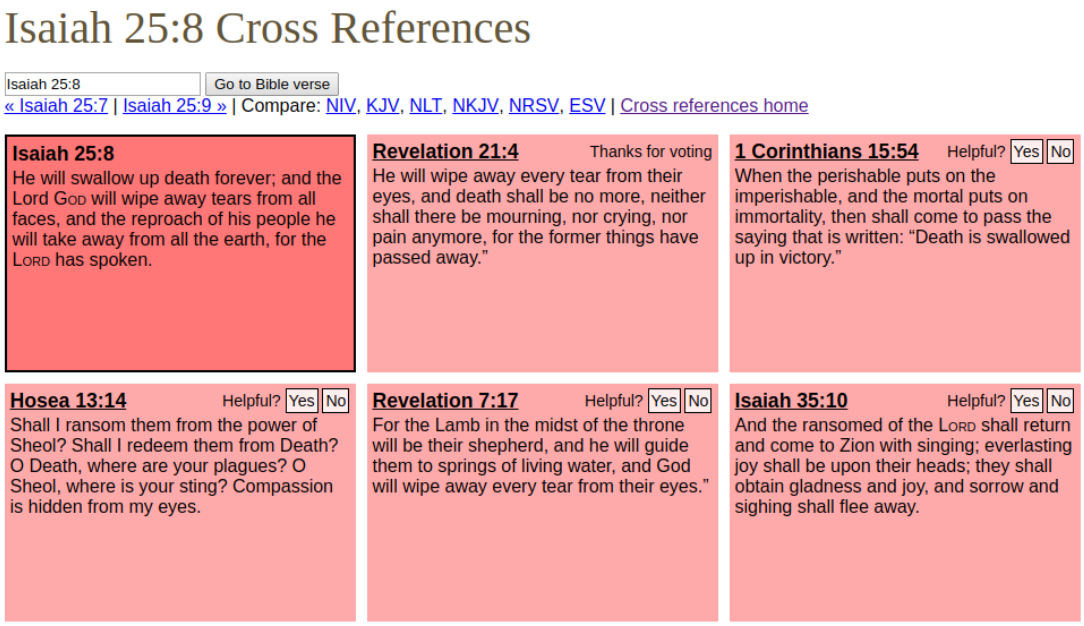
We now describe our approach to topic-based cross-referencing. The baselines built upon word or topic concordances simply propose any cross-reference for which a word or topic matches another verse, meaning that we cannot set a threshold on the quality of the proposed cross-references. Instead, we propose comparing document-topic distributions as K-dimensional vectors, where K is the number of topics, using standard vector distance metrics to compare verses. This idea has been used before (Towne et al., 2016), although not for the task of producing cross-references. By using a vector distance metric to compare the topical similarity of verse pairs, we can set a threshold on the number of proposed cross-references and propose only the most topically related verse pairs as cross-references. We experiment with four distance metrics: cosine distance, Euclidean distance, cityblock (or Manhattan) distance, and Chebyshev distance. However, given that previous work comparing document-topic vectors from LDA seem to default to cosine similarity (Towne et al., 2016; Chuang et al., 2012), we anticipate that cosine distance will be the best metric for selecting cross-references.

2.4 Model Selection

We claim that a fine-grained topic model, i.e., a topic model with a large number of highly nuanced topics, will be able to provide more value

---

[3] For this reason, published biblical word concordances typically only give a manually curated subset of significant vocabulary terms.



Isaiah 25:8 Cross References

Isaiah 25:8
He will swallow up death forever; and the Lord God will wipe away tears from all faces, and the reproach of his people he will take away from all the earth, for the Lord has spoken.

Revelation 21:4
He will wipe away every tear from their eyes, and death shall be no more, neither shall there be mourning, nor crying, nor pain anymore, for the former things have passed away.”

1 Corinthians 15:54
When the perishable puts on the imperishable, and the mortal puts on immortality, then shall come to pass the saying that is written: “Death is swallowed up in victory.”

Hosea 13:14
Shall I ransom them from the power of Sheol? Shall I redeem them from Death? O Death, where are your plagues? O Sheol, where is your sting? Compassion is hidden from my eyes.

Revelation 7:17
For the Lamb in the midst of the throne will be their shepherd, and he will guide them to springs of living water, and God will wipe away every tear from their eyes.”

Isaiah 35:10
And the ransomed of the Lord shall return and come to Zion with singing; everlasting joy shall be upon their heads; they shall obtain gladness and joy, and sorrow and sighing shall flee away.

Figure 1: Voting interface for cross-reference data from `openbible.info`.

for tasks like cross-referencing than traditional coarse-grained topic models. In order to validate this claim, we will compare our fine-grained models with topics from a traditional Latent Dirichlet Allocation model with 100 topics. We refer to this baseline model as *coarse*.

Traditional probabilistic topic models such as Latent Dirichlet Allocation are not able to utilize large numbers of topics (Wallach et al., 2009). However, we successfully train anchor-based topic models with thousands of topics. Consequently, for our fine-grained models, we will employ the Anchor Word algorithm (Arora et al., 2013). Anchor-based topic models view topic modeling as non-negative matrix factorization. This class of topic models attempts to decompose a document-word matrix into two matrices, including a topic-word matrix which gives the conditional probabilities of a particular word given a topic. Ordinarily, this factorization is NP-Hard (Arora et al., 2012). However, given a set of anchor words, or words which uniquely identify a topic, the computation requires only $\mathcal{O}(KV^2 + K^2VI)$ where K is the number of topics, V is the size of the vocabulary, and I (typically around 100) is the average number of iterations (Arora et al., 2013).

We train our anchor-based model using 3,000 topics. We choose this number based on the number of documents we expect each topic to explain: there are roughly 30,000 verses and, according to OpenBible+0, a median of 10 cross-references per verse, so we want each topic to be responsible for roughly 10 documents.

By default, the anchors for the 3,000 topics are produced using a modified form of the Gram-Schmidt process (Arora et al., 2013). This process views each word as a vector in high-dimensional space and attempts to pick anchor words which maximally span that space. For more details, see Arora et al. (2013). In our results and figures, we refer to this model with the default anchor selection method as *Gram-Schmidt*.

This does present us with some difficulty with anchor words as this process tends to select the most extreme and esoteric anchors possible (Lee and Mimno, 2014), which can lead to less useful topics as we increase their number. Lund et al. (2017) introduced a method of using multiple words to form a single anchor. This method, called tandem anchoring, was originally formulated as a way to extend the anchor algorithm to allow for interactive topic modeling.

Instead of utilizing human interaction to seed the topic anchors, we will seed the tandem anchors using the terms from randomly selected verses. For example, suppose we randomly select Isaiah 25:8 as a verse from which to form an anchor. As shown in Figure 1, this verse includes terms such as 'swallow', 'death', and 'tears'. Each of these terms is represented as a point in high-dimensional space. To produce a single anchor from these terms, we average the words using the element-wise harmonic mean[4]. While this new point may not correspond to any particular word, it does capture the joint occurrence pattern of the words which form the anchor. We repeat this process 3,000 times to produce an anchor-based topic model with tandem anchors. While this exact methodology of seeding topic anchors using randomly selected verses is novel, we note the similarity to the method used to seed topics in Rephil, a web scale topic model used by Google (Murphy, 2012). In figures, we refer to this model with tandem anchors as *tandem*.

For each of these models, we must take the topic-word distributions from the topic model and produce document specific topic assignments. We utilize mean field variational inference in order to assign the individual verses to topics, similar to Nguyen et al. (2015).

3 Results

In this section we present the results of our experiments with topic-based cross-referencing. As discussed in Section 2, we experiment with three different cross-reference datasets: TSKE, OpenBible+0, and OpenBible+5. We utilize three different topic models: coarse, Gram-Schmidt, and tandem. We seek to demonstrate that our topic-based cross-referencing system can effectively utilize fine-grained topic modeling to produce candidate cross-references which can be annotated by humans in a cost effective manner.

3.1 Metric Comparisons

We first explore the various metrics for selecting cross-references discussed in Section 2.3. With each proposed distance metric, we are able to set a threshold and determine which Bible verse pairs to keep as candidate cross-references and which

---

[4] See Lund et al. (2017) for details on why the harmonic mean is useful for forming tandem anchors

to discard. Figure 2 and Figure 3 summarize our results with respect to metrics.

Figure 2 gives a receiver operator characteristic (or ROC) curve, which compares the true positive rate (or recall) against the false positive rate (or fall-out). We show the curve for the various metrics using the TSKE as our ground truth. We show these curves on each of the three different models discussed in Section 2.4.

Overall, cosine distance is the best method for selecting cross-references, as it gives the largest area under the ROC curve. The major exception to this is with the traditional coarse-grained topic model for which Euclidean distance performs the best. Considering that cosine distance has frequently been used in conjunction with topics from LDA (Towne et al., 2016; Chuang et al., 2012), this result is somewhat surprising.

Also of interest is the fact that the word-match baseline does reasonably well with respect to the true positive rate, at least if a false positive rate of roughly 0.196 is acceptable. Note that this corresponds to 188,974,806 false positives in the TSKE dataset, so while the raw number of true positives may be impressive, this baseline is not likely to be useful in practice.

While the ROC plot is undoubtedly more popular than the precision-recall plot, in cases where the data is imbalanced, the precision-recall plot can be much more informative. This is mainly because of the use of false-positives in the ROC curve, which can present an overly optimistic picture of the classifier performance (Saito and Rehmsmeier, 2015). Consequently, in Figure 3, we also compare each of the proposed metrics using a precision-recall curve (or PRC).

The PRC plot reinforces the claim that cosine distance is the best distance metric to threshold cross-references since for any reasonable level of precision, cosine distance yields the best results. Once again, with the exception of Euclidean distance performing better with the coarse-grained topic model. However, we note that the precision using coarse-grained topics is much lower than using fine-grained topics with cosine distance. Even with Gram-Schmidt based fine-grained topics, where Euclidean distance eventually wins out against cosine distance, the high amount of false

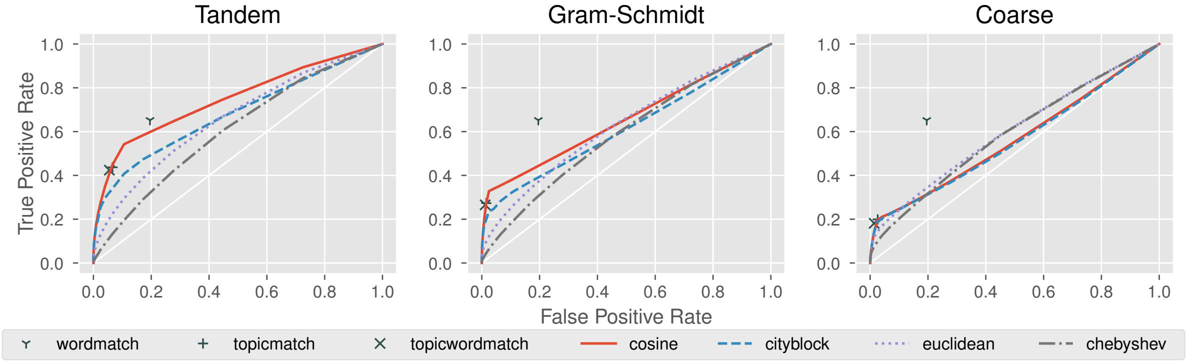


Figure 2: Cross-reference ROC curves for different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground truth.

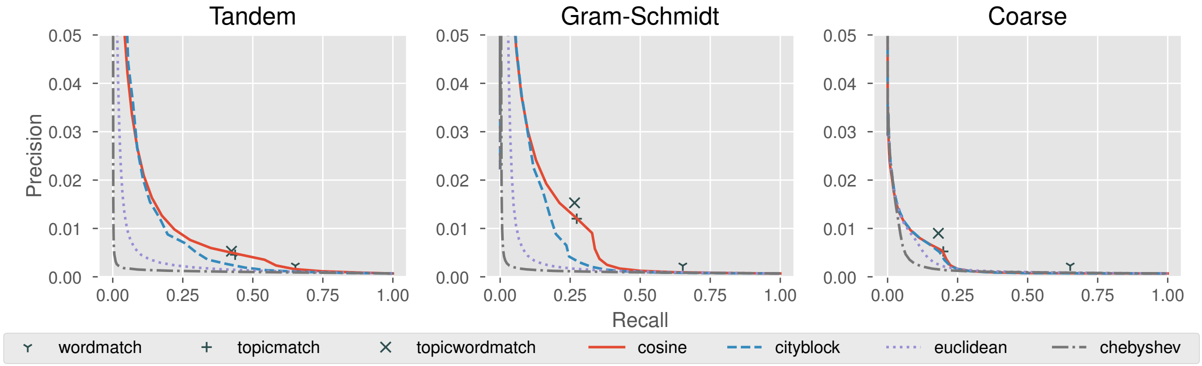


Figure 3: Cross-reference PRC plots with different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground truth.

positives means that cosine distance is the most useful metric for this task. The PRC plot also illustrates why the matching baselines are not practically useful—they do have decent true positive rates, but the precision with these baselines is extremely low.

3.2 Topic Model Comparison

We now explore the various topic models discussed in Section 2.4. Figure 4 and Figure 5 summarize these results. Based on Section 3.1, each reported result in this section uses cosine distance to determine verse pairs which should be considered as candidate cross-references. We compare the results of the three topic models on the three datasets discussed in Section 2.1.

The overall trend on each of the three datasets is simply that we lower the true positive rate and precision as we use more selective datasets. Considering that OpenBible+5 is a subset of OpenBible+0, and that OpenBible+0 is nearly a subset of TSKE[5], this result is not surprising. In the final cost analysis (see Section 3.3), the selectivity of the actual annotators must be taken into account when attempting to predict the true positive rate or the precision of a system.

As shown in Figure 4, regardless of the anchor selection strategy, both fine-grained topic models outperform the traditional coarse-grained model. Our model using tandem anchors built from randomly selected verses performs the best for nearly all levels of false positive rate. However, the Gram-Schmidt based anchors produce better true positive rates for very low false positive rates.

This trend is better illustrated with the PRC plots in Figure 5. While for higher values of recall the tandem anchor selection strategy does win out, it is only after precision significantly drops that tandem anchors produce superior predictions to Gram-Schmidt anchors.

---

[5] OpenBible+0 has 533 references seeded from other public domain sources, which are not included in TSKE.

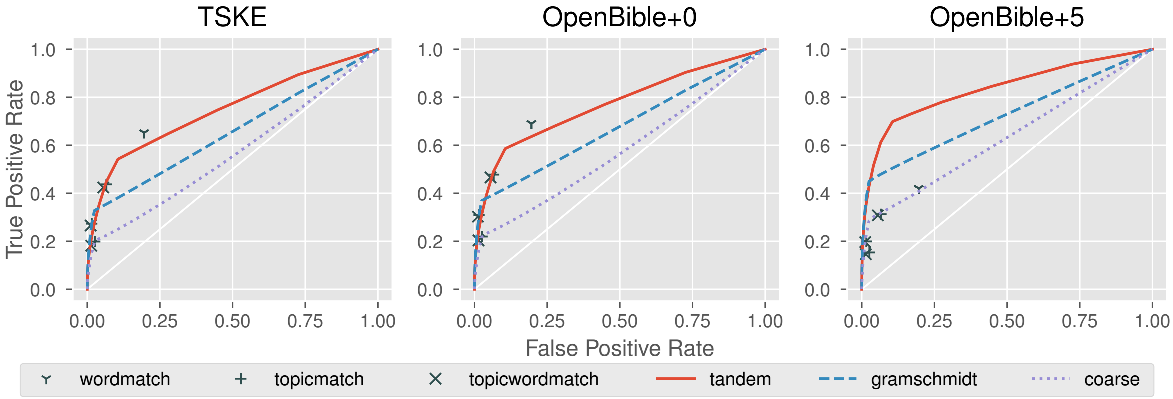


Figure 4: Cross-reference ROC curves with different models for cross-reference selection with three different datasets.

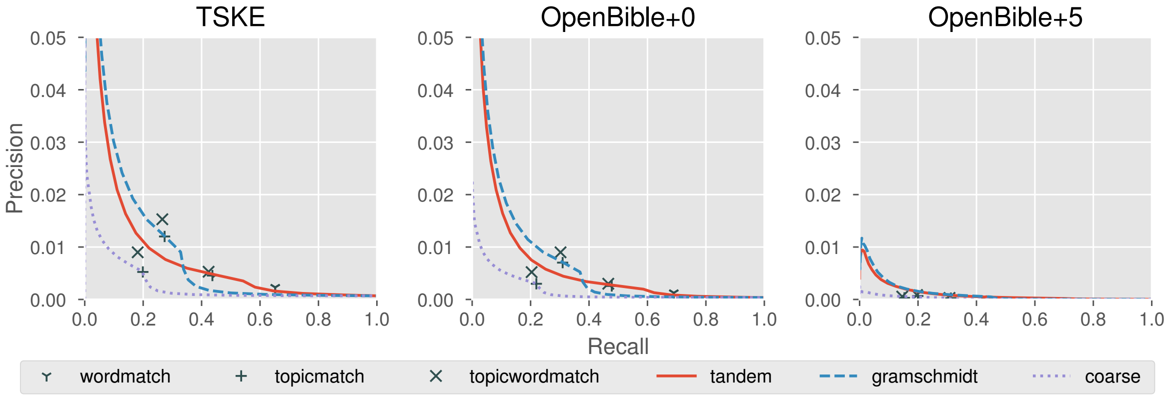


Figure 5: Cross-reference PRC plots with different models for cross-reference selection with three different datasets.

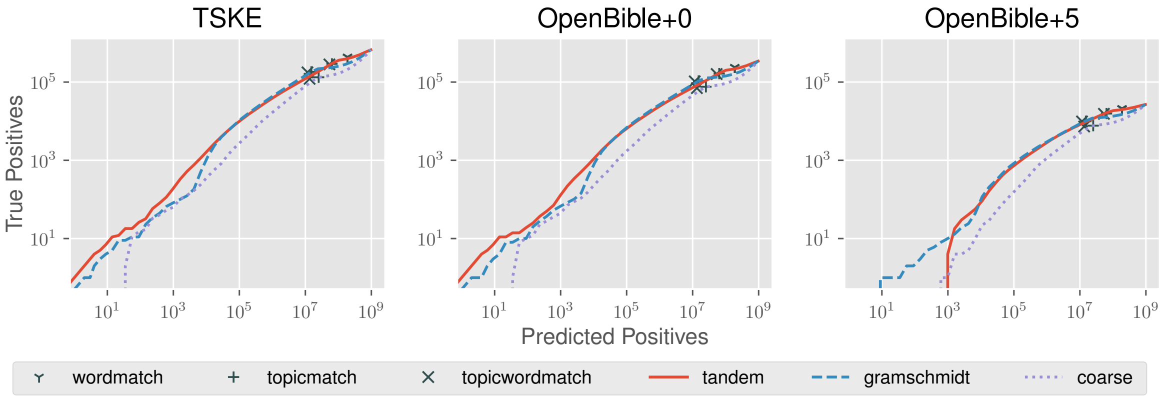


Figure 6: Cross-reference cost curves with different models for cross-reference selection with three different datasets. Note the log-log scale. The x-axis denotes the number of cross-references produced by our method, while the y-axis indicates how many of those cross-references are valid according to the human-provided ground truth.

3.3 Cost Analysis

While the precision-recall curves in Figure 5 may suggest that Gram-Schmidt based topic models produce superior topics for cross-referencing, we suggest that this analysis may be missing a key point in real world analysis. As an alternative to both PRC and ROC curves, we suggest that this task might be best served with an analysis of cost per true positive.

We envision that our system would be used to produce a set of candidate cross-references which would then be curated using human annotators. These annotators would be tasked with evaluating each potential reference and determining whether or not each cross-reference is valid. Critically, the annotator would only be required to evaluate individual cross-references, not the entire text.

As a working example of the cost of such an annotation process, suppose we use a popular crowdsourcing service (e.g., Amazon Mechanical Turk) to produce human annotations. We might reasonably expect to pay something around $0.01 USD per annotation. We would likely require some form of quality control in the form of redundant annotations, so we might end up paying $0.05 USD per annotated cross-reference candidate. Of course, the exact cost per annotated cross-reference will vary depending on the service and difficulty of the specific text being cross-referenced. However, we will use these estimates for the purpose of illustration.

Suppose as part of this working example, we are interested in producing a resource with 12,000 valid cross-reference annotations (roughly matching the size of the previously mentioned LDS edition of the Bible (Anderson, 1979)). Consulting Figure 6 we can then determine how many candidate cross-references we would need to produce for human annotation in order to create the final curated cross-reference resource. For example, using the TSKE as our ground truth, we would need approximately 150,000 predicted positives in order to find 12,000 true positives. At $0.05 USD per annotation, this would cost about $7,500 USD. Supposing that our annotators were more selective, we could use the OpenBible+0 as the ground truth, which would roughly double the cost. With OpenBible+5, which is considerably more selective, this cost rises to approximately $1,000,000 USD. In contrast, with traditional coarse-grained topic modeling, this cost is anywhere from $40,000 USD using the TSKE as the ground truth, to $17,500,000 USD using OpenBible+5 as the ground truth.

While these costs may seem prohibitive, consider that the alternative is to have experts understand the entire text to the degree that they can read one passage and recall other relevant passages they have previously read. In the case of religious texts, this is often possible since adherents study those texts as part of their daily routine. For example, in the case of the LDS edition, it took a committee of experts seven years of work to produce their cross-reference resource, even with the aid of hundreds of volunteers. However, without those experts and volunteers, the cost would have been even greater. In the naive case where every possible reference is manually checked, the cost skyrockets to around $48,000,000 USD.

Passage 1	**Passage 2**
“I have an excessive regard for Miss Jane Bennet, she is really a very sweet girl, and I wish with all my heart she were well settled. But with such a father and mother, and such low connections, I am afraid there is no chance of it.”	“I wish you may not get into a scrape, Harriet, whenever he does marry;–I mean, as to being acquainted with his wife–for though his sisters, from a superior education, are not to be altogether objected to, it does not follow that he might marry any body at all fit for you to notice. The misfortune of your birth ought to make you particularly careful as to your associates. There can be no doubt of your being a gentleman’s daughter, and you must support your claim to that station by every thing within your own power, or there will be plenty of people who would take pleasure in degrading you.”

Table 1: Passage 1 is taken from the eighth chapter of *Pride and Prejudice*, and Passage 2 is taken from the fourth chapter of *Emma*. These two passages are a valid cross-reference because they both discuss social standing and family connections in the context of marriage. Their connection was found even with their lack of shared words.

4 Discussion

Without extensive cross-referencing resources for more secular datasets, it is difficult to empirically prove the usefulness of our system generally without an extremely costly user study. That said, we make a small attempt by manually examining cross-references generated from the complete works of Jane Austen and Plato. Based on our cost analysis in Section 3.3, and since we will be examining only a small number of cross-references, we utilize tandem anchors to generate topics. With each dataset, we examine the first 300 cross-references produced by our system.

We also examine what our model got wrong in proposing Bible cross-references.

4.1 Jane Austen

Of the first 300 cross-references, we find that 39 of them are valid, linking passages from all six works by Jane Austen. As with our experiments with the Bible, this level of precision is sufficient that we believe that we could dramatically lower the cost of producing a full cross-referencing resource for this text.

We note that 109 of these are cross-references linking a paragraph in the eighth chapter of *Pride and Prejudice* to other passages in our corpus. Of the references involving this one paragraph, 22 were valid. An example of such a cross-reference is shown in Table 1. While marriage in general is a common theme in the works of Jane Austen, this particular paragraph more specifically discusses the role of social status and family connection as it relates to choosing a marriage partner. We note that the connection between the passages in Table 1 is thematic; they share no significant words in common, demonstrating the capability of the system to detect nuanced topics and themes.

4.2 Plato

Of the first 300 cross-references generated from the works of Plato, we found 119 that were valid. Many of the cross references were between distinct works, and included discussions about the nature of justice, arguments about the composition of things, the nature and role that certain things play, and discussions of appropriate legislation.

It is important to note that the model found significantly more cross references in the works of Plato than those by Jane Austen. This is likely due to the nature of the writing. We find in the works of Plato that ideas themselves are discussed directly, similar to the bible, and thus we would expect it to be easier for a model to find words and phrases that link to a specific topic. An example of this is that words like “virtue,” “courage,” “mean,” and “cowardice” would likely identify a topic about virtue that comes up in the works of Plato. However, in Jane Austen we find ideas discussed implicitly through interactions of the characters and commentary by the author. The meaning, while present, is found by “reading between the lines.” An example of this is that we might find marriage discussed in the works of Jane Austen, but more often through characters discussing their feelings after getting married. Here, words that we might see could be words such as “happy,” “elated,” “love,” “efficient,” etc. However, these words could also correlate equally well with other topics, and thus it would be harder for our model to discern. As further evidence of this we point out that the cross reference we used above from Jane Austen is an explicit discussion of marriage. It is likely that an implicit discussion of marriage would be harder for our model to find. We also point out that in such cases it is a non-trivial task for humans to come to a specific consensus about what a given passage could mean or relate to.

That said, given the relevant and influential nature that the works of Plato still hold even today, we can see that these cross references are highly useful in that they facilitate study and understanding of his works that a study of each individual work separately might miss.

4.3 Error Analysis

We examine the errors of running tandem anchors using cosine distance on the Bible. There are two types of errors to examine: candidate cross-references proposed early in the process that are not valid cross-references and valid cross-references that are not proposed until the end of the process (as determined by the Treasury of Scripture Knowledge).

Early invalid candidate cross-references all exhibit the same characteristic; the documents are exactly or substantively the same (e.g. *Deuteronomy 2:17 the Lord said to me,* and *Deuteronomy 2:2 Then the Lord said to me*). Indeed, a human given only those two documents would also mark them as related, and many valid cross-references exhibit this same characteristic (e.g. Psalms 107:6 and Psalms 107:28 are exactly the same and are a valid cross-reference).

Cross-references are partially so difficult because what constitutes a valid cross-reference is at least partially determined by what the community surrounding the text views as significant, and so two documents that are identical may or may not be a valid cross-reference depending on the view of that community. This particular issue would make any end-to-end automated solution to cross-referencing particularly difficult.

We also examined the last one hundred valid cross-references proposed. We consider these errors because in a system where a predetermined number of candidate cross-references are considered, these candidate cross-references would most likely never be considered. For 28 of the last one hundred valid cross-references, it is unclear why they are considered valid cross references (e.g., *Ezekiel 16:10 I clothed you also with embroidered cloth and shod you with fine leather. I wrapped you in fine linen and covered you with silk.* and *Deuteronomy 8:11 "Take care lest you forget the Lord your God by not keeping his commandments and his rules and his statutes, which I command you today,*).

Many of the other 72 valid cross-references are also difficult in some way. Many of the connections involve some use of metaphor (25) or are linked by a single key word, such as a name, but are otherwise topically dissimilar (34). The other 13 cross-references are all documents describing the construction of the tabernacle, and we have enough extra context to recognize this, however it isn't surprising that the model doesn't find this connection.

It is useful to note that of the final 600,000 candidate cross-references, only one hundred of them were valid cross-references.

5 Conclusion

We have produced a system using fine-grained topic modeling which is able to propose candidate cross-references which can be verified by non-expert human annotators for the purpose of creating a cross-reference resource at a fraction of the cost of current manual techniques. Our method, which utilizes tandem anchors to produce large numbers of highly nuanced topics coupled with an effective assignment strategy, is able to produce document-topic vectors which are comparable using cosine distance.

Our results also demonstrate that this system would not be as cost effective with traditional coarse-grained topic modeling. While we can find sets of topically related documents using coarse-grained topics, for the task of finding the most closely related documents we require a system which is more specific. We suggest that this success serves as motivation for exploration of fine-grained topic modeling for other topic-based use cases which require nuance and precision.

Acknowledgements

This work was supported by the NSF Grant IIS-1409739

References

Lavina Fielding Anderson. 1979. Church publishes first LDS edition of the bible. *Ensign*, 10.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning topic models–going beyond svd. In *Fifty-Third IEEE Annual Symposium on Foundations of Computer Science*.

Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. 2012. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 443–452. ACM.

Crossway. 2001. *The Holy Bible: English Standard Version*. Crossway Bibles.

Guiness. 2018. Best selling book of non-fiction. Http://www.guinnessworldrecords.com/world-records/best-selling-book-of-non-fiction.

Moontae Lee and David Mimno. 2014. Low-dimensional embeddings for interpretable anchor-based topic inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jeffrey Lund, Connor Cook, Kevin Seppi, and Jordan Boyd-Graber. 2017. Tandem anchoring: A multi-word anchor approach for interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Timothy Morton. 2010. *Treasury of Scripture Knowledge, Enhanced*. BibleAnalyzer.com.

Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*, pages 928–929. MIT Press.

Thang Nguyen, Jordan Boyd-Graber, Jeffrey Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.

James Strong. 1890. *The Exhaustive Concordance of the Bible*. Jennings & Graham.

R.A. Torry. 2002. *Treasury of Scripture Knowledge*. Hendrickson Publishers.

W Ben Towne, Carolyn P Rosé, and James D Herbsleb. 2016. Measuring similarity similarly: Lda and human perception. *ACM Transactions on Intelligent Systems and Technology*, 8(1).

Hanna M Wallach, David M Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In *NIPS*, volume 22, pages 1973–1981.

Conversation Initiation by Diverse News Contents Introduction

Satoshi Akasaki[*]
The University of Tokyo
akasaki@tkl.iis.u-tokyo.ac.jp

Nobuhiro Kaji
Yahoo Japan Corporation
nkaji@yahoo-corp.jp

Abstract

In our everyday chit-chat, there is a conversation initiator, who proactively casts an initial utterance to start chatting. However, most existing conversation systems cannot play this role. Previous studies on conversation systems assume that the user always initiates conversation, and have placed emphasis on how to respond to the given user's utterance. As a result, existing conversation systems become passive. Namely they continue waiting until being spoken to by the users. In this paper, we consider the system as a conversation initiator and propose a novel task of generating the initial utterance in open-domain non-task-oriented conversation. Here, in order not to make users bored, it is necessary to generate diverse utterances to initiate conversation without relying on boilerplate utterances like greetings. To this end, we propose to generate initial utterance by summarizing and chatting about news articles, which provide fresh and various contents everyday. To address the lack of the training data for this task, we constructed a novel large-scale dataset through crowd-sourcing. We also analyzed the dataset in detail to examine how humans initiate conversations (the dataset will be released to facilitate future research activities). We present several approaches to conversation initiation including information retrieval based and generation based models. Experimental results showed that the proposed models trained on our dataset performed reasonably well and outperformed baselines that utilize automatically collected training data in both automatic and manual evaluation.

1 Introduction

Conversation[1] systems are becoming increasingly important as a means to facilitate human-computer communication. However, most of the studies on conversation systems have been based on the assumption that a human always initiates conversation. As a result, the systems are designed to be passive (Yan, 2018), meaning that they keep waiting until they are spoken to by the human and will never speak to the human proactively. For example, popular encoder-decoder models (Sutskever et al., 2014; Vinyals and Le, 2015) are designed to respond to input utterances provided by humans, and it is difficult for them to proactively initiate the conversation. Although some systems are able to initiate conversations, they basically adopt template-based generation methods and thus lack diversity.

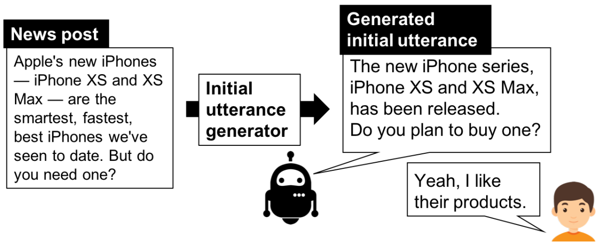


Figure 1: Conversation initiation task. The system in this example is given a news post about "iPhone" and generates an initial utterance for chatting about it.

This paper investigates generating the very first utterance in a conversation. We feel strongly that conversation systems should not always be passive; sometimes, they have to proactively initiate the conversation to enable more natural conversation. In addition, it is crucial to be able to initiate conversation in various ways in actual applications, since systems that initiate a conversation by always saying "*Let's talk about something*" or "*Hello*" are inherently boring.

We propose a task setting in which the system initiates a conversation by talking about a news topic. In this task, the system is provided with

---

[*] This work was done during research internship at Yahoo Japan Corporation.

[1] "Conversation" in this paper refers to open-domain non-task-oriented conversations and chit-chat.

Proceedings of NAACL-HLT 2019, pages 3988–3998
Minneapolis, Minnesota, June 2 - June 7, 2019.

a news post to talk about and uses it to generate the initial utterance of the conversation (Fig. 1). This task is referred to as *conversation initiation* in this paper. We have two primary reasons for using news posts. First, sharing and exchanging opinions about the latest news with friends is common in our daily conversations (Purcell et al., 2010) (*e.g,* asking something like *"What do you think about today's news on Trump?"*). Second, and more importantly, this task setting allows us to proactively generate diverse utterances to initiate conversations by simply using the latest news posts, which include a wide variety of content published daily.

We created a large-scale dataset for training and evaluating conversation initiation models through a crowd-sourcing service. The crowd-sourcing workers were presented with news posts collected from Twitter and asked to create utterances to initiate a conversation about the post. The resulting dataset will be released to facilitate future studies at the time of publication.

We developed several neural models, including retrieval-based and generation-based ones, to empirically compare their performances. We also compared the proposed models against baselines that utilize automatically constructed training dataset to investigate the effectiveness of our dataset. Both automatic and manual evaluation were used to assess not only the quality but also the diversity of the generated initial utterances. The results indicate that the proposed models successfully generated initial utterances for the given news posts, and significantly outperformed the baseline models.

Our contributions are the following:

- We investigate the task of conversation initiation, which has been largely overlooked in previous studies.
- We construct and release a large-scale dataset for conversation initiation.
- We develop several neural models and empirically compare their effectiveness on our dataset.

2 Related work

2.1 Non-task-oriented Conversation System

There are many existing studies on non-task-oriented conversation systems. Research started with rule-based methods (Weizenbaum, 1966; Wallace, 2009) and gradually shifted to statistical approaches (Ritter et al., 2011; Vinyals and Le, 2015), and many follow-up studies have since been undertaken to improve the quality of the generated responses (Hasegawa et al., 2013; Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016b; Serban et al., 2017).

However, the task of conversation initiation has been largely absent in these studies.

2.2 Data Grounded Conversation

There have also been efforts to develop systems that can chat with users about specific documents such as Wikipedia articles (Zhou et al., 2018) or reviews (Moghe et al., 2018). However, these studies did not investigate how to initiate such conversations, and as a result, their models assume that the initial utterance is always given by users. Also, their datasets are designed to be used to train models of multi-turn conversations about the given documents, rather than models of conversation initiation. For example, Moghe et al. (2018) utilized fixed templates to initiate conversations, and there are only a few (around 4k) utterances that can be used to train the model of conversation initiation in Zhou's dataset (2018).

In contrast, we focus on the conversation initiation task, which those studies have largely overlooked, and develop a large-scale dataset that includes 109,460 utterances for this task (see Section 3). Therefore, our work can be considered complementary to the previous studies.

In an approach that uses images rather than documents, (Mostafazadeh et al., 2016) proposed a method of generating questions about an image to initiate conversation. Although, like us, they explored initiating conversation, they focused only on generating questions. In contrast, we investigate generating other types of initial utterances than questions. Also, they investigated a task setting in which users can see the images along with the conversation, while we do not present the news posts to users. This difference makes our generation task a bit more complicated (see Section 3).

2.3 Proactive Conversation System

Some studies have attempted to make conversation systems more proactive rather than passively waiting for utterances from a user. (Li et al., 2016c) proposed a system that detects a stalemate in the conversation and then proactively casts a specific response for breaking the stalemate. They use the history of the user's utterances to select response candidates. (Yan et al., 2017; Yan and Zhao, 2018) proposed a method of proactively suggesting the user's next utterance. Although these methods have been successfully used in proactive conversation systems, the conversation initiation has not been investigated.

2.4 Diverse Response Generation

A well-known problem of encoder-decoder-based conversational models is that they tend to generate generic responses such as "*I don't know*" (Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016). Such responses understandably bore users, so there has been much research focus on generating more diverse responses (Li et al., 2016a; Xu et al., 2018; Baheti et al., 2018).

We explore the problem of generating diverse initial utterances from a different perspective than other studies. In our problem setting, it is not obvious how to go beyond simple template-based systems, which cannot generate diverse utterances. We address this problem by generating initial utterances based on news posts, which feature various content and are updated every day.

This study is complementary to previous attempts at diversification. Our method exploits existing neural conversation models, which tend to generate generic responses, as a component. The previous diversification methods can be used to improve the initial utterances in our method.

2.5 Other Related Work

Question Answering (QA) tasks have long been studied in the research community (Rajpurkar et al., 2016, 2018). In recent years, conversational variants of this task such as visual QA (Antol et al., 2015; Das et al., 2017) and conversational QA (Reddy et al., 2018) have been proposed. All of these tasks differ from our conversation initiation task since they focus on how to respond to questions.

(Yoshino and Kawahara, 2014) proposed an information navigation system that presents users with the contents of news articles through conversation. Although this setting is similar to ours, their system always opens conversation by just presenting the news headline. Our study investigates initiating conversation in a more chatty way, and should contribute to making the systems more conversational and attractive.

(Qin et al., 2018) proposed the task of generating comments about given news articles. Although this task is similar to ours, it is not designed to converse with users. Our task focuses on conversation and tries to generate initial utterances using news articles (posts).

3 Conversation Initiation Dataset

In this section, we explain how we constructed the dataset for the task of conversation initiation. We then analyze the constructed dataset to provide insights into its effectiveness.

3.1 Data Construction

We first collected 104,960 Japanese news posts from the Twitter account @YahooNewsTopics,[2] which delivers the latest news in the world every day. The data were collected between December 31, 2013 and October 31, 2017. Some example posts collected from this account are listed in the third column of Table 1.[3]

We investigate the task setting in which the system opens a conversation about a given news post. Here, we presume the post is not presented to the user during the conversation. Although letting users see the news posts would be possible, such a setting is not investigated here because our focus is a situation where users converse with the system only by voice. Such situations are growing more popular in recent years with the rise of voice-controlled conversation systems such as intelligent assistants (*e.g.,* Siri, Alexa, and Cortana) (Jiang et al., 2015; Sano et al., 2016; Akasaki and Kaji, 2017) and smart speakers (*e.g.,* Amazon Echo and Google Home).

Therefore, in our task setting, since the user does not always know about the news, it is preferred to first introduce the news summary so as to share the background knowledge before starting the conversation (see Fig. 1). In this sense, our task can be understood as a combination of summarization and chit-chat. Interestingly, the summarization subtask goes beyond the ordinary one in that we not only compress the content but also generate the text in a chit-chat-like style.

---

[2] https://twitter.com/yahoonewstopics

[3] Original news posts were written in Japanese. We have translated them for clarity.

dialogue acts	# examples	news posts (translated)	initial utterances (translated)
IMPRESSION	7,929	Yu Abiru announces her marriage on a broadcast. Her affiliation office and partner also commented.	It seems that Yu Abiru announced her marriage on a broadcast. *Congrats!*
		Major beer companies will increase beer production by about 10% this summer compared to the same period last year. Managerial resources have been shifted from other products due to a hot summer and tax cuts.	I heard that major beer companies are planning to increase beer production by about 10% this summer compared to last summer. *It makes me want to drink a cold beer on a hot day.*
URGING	273	The Korean Defense Department revealed that the North Korean army launched several ”short-range projectiles” toward the Sea of Japan on the morning of the 3rd.	North Korean forces launched missiles toward the Sea of Japan. *Let's evacuate quickly!*
		Severe damage to the mind and body due to abuse cannot be healed easily. We followed the cases of women who suffered abuse as children.	Wounds by abuse are stored deeply in the body and mind. *Let's do something to help if child abuse is happening around you.*
QUESTION	1,028	Players of the national men's handball team smoked in a non-smoking area while staying at Ajinomoto national training center. They received an indefinite ban from JOC.	The national men's handball team players smoked in a non-smoking area and were expelled, I heard. *Have you ever seen a handball game?*
		An infant was caught between a bed guard and a mattress and subsequently died. In the US, 13 children were killed in the past 11 years due to such accidents. Japan Pediatric Society has called attention to this.	There seems to have been an accident caused by an infant's bed guard. *Do you think that bed guards are necessary?*

Table 1: Distribution over sampled dialogue acts and example initial utterances. *Italics* are chit-chat parts.

	# word	# sent.	vocab. size
News post	33.85	1.96	54,830
Initial utterance	31.50	2.03	49,211
*Summary part	22.27	1.00	45,850
*Chit-chat part	9.23	1.03	19,520

Table 2: Statistics of the dataset. First and second columns show the average numbers per utterance.

To construct the dataset, we had cloud workers create the initial utterance of a conversation on the basis of a given news post. We instructed workers to not only chat about the news post but also to provide its brief summary. The workers were asked to use colloquial expressions because users feel strange when spoken to in literary expressions. We obtained a total of 104,960 pairs of news post and initial utterance[4]. Note that we created only the initial utterances (same as (Mostafazadeh et al., 2016)) because our focus is how to initiate conversation[5].

3.2 Data Analysis

Here we discuss our investigation of the 104,960 initial utterances. Some examples of the utterances are listed in Table 1. Most initial utterances first summarize the contents of the news post and then begin to chat about it, as we instructed. For subsequent analysis and model designing, we divided each initial utterance into sentences and then designated the one with the smallest edit distance from the input news post as “summary part” and the rest as “chit-chat parts”. The rationale behind the use of this heuristic is that the summary part shares more words with the original news post than the chit-chat part and consists of just one sentence in most cases. The statistics of the dataset are shown in Table 2.

For the summary part, as seen in Tables 1 and 2, original news posts are compressed by 32.29% on average and are converted into a colloquial style. This indicates that the recruited cloud workers properly extracted the important contents from the input news posts and used them for making the summary part.

Compared with the summary part, the number of words and vocabulary size for the chit-chat part are relatively small (Table 2). This is a natural phenomenon since the summary part uses more con-

---

[4] Some news posts (typically emergency news such as earthquake) were posted more than once, and as the consequence the dataset includes 102,844 unique news posts. In the experiment, we took care so that the training and test datasets do not include the same news posts.

[5] Of course it is necessary to continue successive conversation in an actual application, but we here leave this setting as a future work.

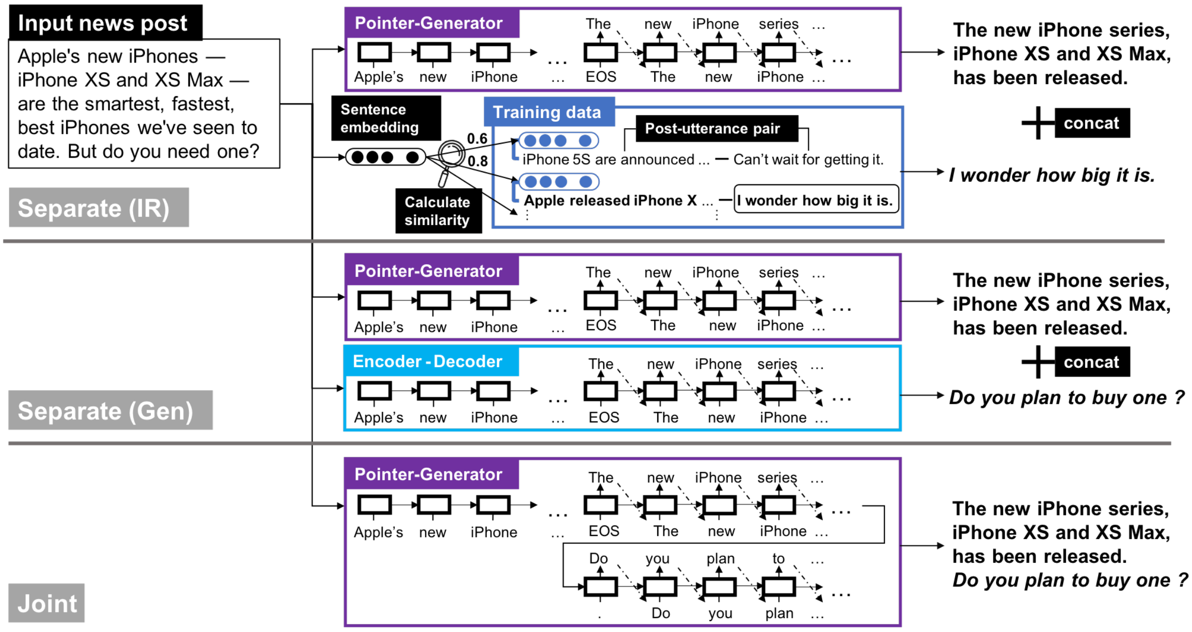


Figure 2: Overview of initial utterance generation by our proposed approaches. *Italics* are chit-chat parts.

tent words for summarization than the chit-chat part. To clarify how workers created these chit-chats, we randomly sampled 10,000 utterances and manually classified them according to their dialogue acts, as shown in Table 1. We found that the majority (92% = (7929 + 273 + 1082) / 10000) are classified into three dialogue acts (IMPRESSION, URGING, and QUESTION). The remaining 8% miscellaneous utterances that do not belong to any of the three dialog acts.

Most of the labeled initial utterances are the impressions and opinions of cloud workers about news posts (see the IMPRESSION act). Some of them are boilerplates (*e.g., "Congrats"*) while others show tremendous diversity (*e.g., "It makes me want to drink a cold beer on a hot day"*). It is interesting that some workers make an urging (*e.g., "Let's evacuate quickly"*) or ask a question (*e.g., "Have you ever seen a handball game?"*). These acts that attempt to solicit the user's response are important elements for conversation initiation.

4 Generation Method

As described in Section 2, most of the initial utterances in the dataset can be divided into a summary part and a chit-chat part. Because it is possible to generate these two parts by two separate models or by a single joint model, we investigate both approaches and compare their performance in an experiment. The overview of our proposed approaches is given in Fig. 2.

4.1 Separate Approach

The separate approach utilizes two different models to generate the summary part and the chit-chat part, respectively.

The summary part is generated by the pointer-generator model, which allows both copying words by pointing to the input sentence and generating words from a fixed vocabulary (See et al., 2017). This model is suitable for generating the summary part because it can appropriately select the contents of the input sentence while compressing them to a proper length.

To generate the chit-chat part, both generation-based and information retrieval (IR)-based methods are investigated. We use a common encoder-decoder model (Vinyals and Le, 2015) as the generation-based method (see **Separate (Gen)** in Fig. 2). Since this model tends to generate generic sentences that lack diversity (Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016), we also adopt the MMI-antiLM method proposed by (Li et al., 2016a) to promote diversity. This method uses the following score function, instead of the commonly used log-likelihood, when decoding:

$$\log P(T|S) - \lambda \log U(T), \qquad (1)$$

where T is an initial utterance and S is a news post. $P(T|S)$ is the conditional likelihood of T

given S, and U is a language model. In decoding, output candidates are generated using beam search and are then reranked by Eq. 1. This model penalizes generic sentences by $U(T)$.

As the IR-based method, we utilize the embedding of an input news post to retrieve the closest news posts in the training data using cosine distance, and then extract the corresponding chit-chat part (Ritter et al., 2011) (see **Separate (IR)** in Fig. 2). We adopt Smooth Inverse Frequency (SIF)-based embedding (Arora et al., 2017) for inducing news post embeddings. This method first calculates a weighted average of word embeddings in a news post s as:

$$v_s = \frac{1}{|s|} \sum_{w \in s} \frac{a}{a + P(w)} v_w, \qquad (2)$$

where a is a hyperparameter and $P(w)$ is the unigram probability calculated from the training data. Then, it reduces the influence of the first principal component by using the first singular vector u of the word vector matrix:

$$v_s = v_s - uu^{\mathrm{T}} v_s, \qquad (3)$$

This method has demonstrated a competitive performance across various tasks (Arora et al., 2017).

4.2 Joint Approach

We concatenate the summary part and the chit-chat part of the training data and train only one pointer-generator model, as mentioned in Section 4.1 (see **Joint** in Fig. 2).

Unlike the separate approach, this method can be considered multi-task learning of the summary and the chit-chat part generation. Thus, we expect it can generate the initial utterance precisely by considering the coherence between the summary and the chit-chat part. We examine the effectiveness of this approach through experiments in the following section.

5 Experiments

We empirically evaluate the performance of the proposed methods on the constructed dataset.

5.1 Models

In addition to the proposed methods, we implemented baselines that do not use labor-intensive labeled data, since carefully preparing the dataset is one of our contributions. These baselines generate summary and chit-chat parts separately in the following way and concatenate them as output.

RNN type	Bi-LSTM
Layers	2
Hidden layer dim.	512
Embedding dim.	256
Dropout rate	0.2
Parameter init.	(-0.08, 0.08) (uniform)
Vocabulary size	50,000
Batch size	64
Epochs	30
Max. grad. norm.	1
Optimization	Adam
Learning rate	0.001
Beam size	5
λ (MMI)	0.3
γ (MMI)	0.15

Table 3: Hyperparameter settings for training encoder-decoder models.

We gathered tweets (news posts) of major news accounts from Twitter and their corresponding replies (regarded as chit-chats). Those tweet-reply pairs can be used as pseudo training data to generate the chit-chat part. Since we cannot automatically acquire training data for generating the summary part, we output the first sentence of the input news post as the summary part.

Overall, the following proposed and baseline methods were implemented for comparison:

Baseline Generate the summary part and the chit-chat part by separate models using the pseudo-training data collected from Twitter. There are three variants of this method for generating the chit-chat part. **Baseline (IR)** and **Baseline (Gen)** use the IR-based method and the generation-based method, respectively. **Baseline (Gen+MMI)** uses MMI-antiLM (Li et al., 2016a) for decoding.

Separate Generate the summary part and the chit-chat part separately using the approach described in Section 4.1 and the dataset described in Section 3.1. There are also three variants of this method, same as the baselines (**Separate (IR)**, **Separate (Gen),** and **Separate (Gen+MMI),** respectively).

Joint Generate the summary part and the chit-chat part jointly using the approach described

Model	R-1	R-2	R-L	D-1	D-2	D-S
Baseline	**70.2**	**59.1**	**67.5**	**17.7**	**60.1**	**99.8**
Separate	66.5	50.6	63.8	15.7	52.4	**99.8**
Joint	68.8	54.1	66.3	15.2	51.8	**99.8**

Table 4: Results of summary part generation.

Model	BLEU	D-1	D-2	D-S
Baseline (IR)	0.2	**21.8**	**65.0**	**90.3**
Baseline (Gen)	0.2	1.2	3.4	2.8
Baseline (Gen+MMI)	0.2	1.1	3.1	3.7
Separate (IR)	3.5	12.6	34.9	65.2
Separate (Gen)	6.3	1.5	2.9	3.2
Separate (Gen+MMI)	**9.6**	2.2	5.6	13.4
Joint	6.4	6.7	15.8	28.5

Table 5: Results of chit-chat part generation.

in Section 4.2 and the dataset described in Section 3.1.

5.2 Experimental Settings

We divided the 104,960 items of data (news post and initial utterance pairs) into 90,000, 10,000, and 4,960 for training data, development data, and test data, respectively. Input news posts that appear in the training data were removed from the test data. Consequently, 4,776 data were used as the final test data.

To train the baseline model, we collected 277,813 tweets and their corresponding replies from six major Japanese news accounts[6] on Twitter. We then divided those pairs into 260,000 and 17,813 for training data and development data for the baselines.

We performed tokenization using a Japanese morphological analyzer, MeCab,[7] with IPAdic dictionary,[8] and then removed usernames, URLs, and hashtags. We used OpenNMT-py (Klein et al., 2017)[9] for building the models described in Section 4. Their hyperparameter settings are given in Table 3. We used GloVe (Pennington et al., 2014)[10] to learn 300-dimensional word embeddings. We trained word embedding using a Japanese Wikipedia dump released on February 22nd, 2018. These embeddings were used for acquiring news post embeddings, as described in Section 4.1.

5.3 Automatic Evaluation

As discussed in Section 3.2, since the initial utterance can be divided into separate parts that have different properties, we evaluated each part separately to examine the generated initial utterances.

We automatically divided the generated sentences and reference sentences into summary parts and chit-chat parts, as explained in Section 3.2. We used ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) for evaluating the summary part (denoted as **R-1**, **R-2**, and **R-L**, respectively) and BLEU (Papineni et al., 2002) for evaluating the chit-chat part. We use different metrics for each part because ROUGE is often used for summarization tasks while BLEU is used for conversational tasks. Since these automatic metrics are insufficient for evaluation (Novikova et al., 2017), we also perform a manual evaluation in Section 5.4.

To evaluate diversity, we calculate the proportion of distinct unigrams, bigrams, and sentences (**D-1**, **D-2**, and **D-S**, respectively) in the generated initial utterances (Li et al., 2016a).

Table 4 lists the results of the summary part. The baseline method that outputs the first sentence of an input news post achieved higher ROUGE scores than the proposed methods. This does not necessarily mean that the proposed methods are poor because even the SOTA summarization system exceeds such a baseline by only a small margin (See et al., 2017). Also, our task has a requirement to convert sentences into colloquial expressions, and the ROUGE metric cannot capture such a subtle difference. We perform a deeper investigation into the quality of the generated initial utterance in the next section. Regarding the diversity, almost all of the generated initial utterances are distinct, as shown in Table 4.

Table 5 shows the result of the chit-chat part. The proposed methods outperformed the baselines in terms of BLEU score. Although the baselines use two times as much training data as the proposed methods, their scores were quite low. This demonstrates the quality of our dataset. The score of **Separate (IR)** was relatively low among the proposed methods, presumably because the chit-chat parts retrieved from the training data do not always match the content of the input news post.

---

[6] @YahooNewsTopics, @livedoornews, @asahi, @mainichi, @mainichi_jp, @nhk_news

[7] `http://taku910.github.io/mecab/`

[8] `https://ja.osdn.net/projects/ipadic/`

[9] `https://github.com/OpenNMT/OpenNMT-py`

[10] `https://nlp.stanford.edu/projects/glove/`

Model	Naturalness	Coherency
Human	3.31	3.39
Baseline (Gen+MMI)	2.26	2.24
Separate (IR)	2.32	2.21
Separate (Gen+MMI)	2.96	2.96
Joint	**3.07**	**3.06**

Table 6: Results of evaluating **Naturalness** and **Coherency** of the generated utterances by the manual evaluation (higer is better).

Model	Dullness
Human	2.18
Boilerplate	3.06
Separate (IR)	2.47
Separate (Gen+MMI)	2.39
Joint	**2.36**

Table 7: Results of evaluating **Dullness** of the generated utterances by the manual evaluation (lower is better). Boilerplate uses manually created boilerplate utterances.

We also see that all the BLEU scores of the models are quite lower than ROUGE scores in Table 4. In general, both summarization and chat generation tasks often use automatic evaluation metrics to evaluate generated sentences, their scores tend to be much lower in the chat generation task. This is because the answer sentences (utterances) of the chat generation task have more diverse candidates than other generation tasks such as machine translation and summarization (Li et al., 2016a,b; Baheti et al., 2018). We also examine the diversity of the chit-chat part in Table 5. Although the diversity of the IR-based methods was high, their BLEU scores deteriorated considerably. Among the generation-based methods, although **Separate (Gen+MMI)** achieved the highest BLEU score, it lacked diversity. In contrast, **Joint** achieved a reasonable BLEU score while maintaining diversity to some extent.

5.4 Manual Evaluation

Although diversity of utterances can be quantified automatically, ROUGE and BLEU scores do not always follow human intuition (Novikova et al., 2017; Lowe et al., 2017). Therefore, we evaluate the generated initial utterances manually. We picked the three proposed models with good performance in the automatic evaluation along with one baseline for this manual evaluation. 300 posts were sampled as the input news posts, and the outputs of the four methods were manually evaluated from two perspectives: 1) **Naturalness**: *Does the utterance naturally initiate conversation?* and 2) **Coherency**: *Is the content of the utterance coherent with the given news post?* We recruited crowd workers to score each utterance on a 4-point scale (Agree, Slightly Agree, Slightly Disagree, Disagree).

Table 6 show the results of the manual evaluation for **Naturalness** and **Coherency** of the generated initial utterances. The proposed methods excluding **Separate (IR)** outperformed **Baseline (Gen+MMI)** in both perspectives and achieved reasonable scores compared to human upperbound. The scores of **Separate (IR)** are quite low because the retrieval result does not follow the input news post in many cases. This reveals that although those sentences have high diversity, their quality is poor as initial utterances. Although **Baseline (Gen+MMI)** achieved high ROUGE scores in Table 4, its style is not colloquial. Thus, workers felt odd and lowered their scores. In conclusion, it is better to use the generation-based methods for conversation initiation.

We also evaluated **Dullness**: *Is the given utterance dull or boring?* We used 15 manually created boilerplate utterances (*e.g.,* Hello., How are you?, Let's talk with me.) rather than **Baseline (Gen+MMI)** to confirm the effectiveness of utilizing news contents as the initial utterances. Table 7 show the results of the manual evaluation for **Dullness** of the generated initial utterances. We see that compared to our proposed methods, the score of the boilerplate baseline is quite high. This indicates that using boilerplate utterances for conversation initiation often bores users and possibly leads to early abandonment of the conversation.

To determine the statistical significance of our results, we performed Wilcoxon signed-rank tests with Bonferroni correction (Wilcoxon, 1945). In Table 6, for all combinations except **Baseline (Gen+MMI)** vs. **Separate (IR)** and **Separate (Gen+MMI)** vs. **Joint**, there were significant differences (p-value < 0.005 (corrected)) in both perspectives. Similarly, in Table 7, there were statistically significant differences for all combinations except **Separate (IR)** vs. **Separate (Gen+MMI)** and **Separate (Gen+MMI)** vs. **Joint**.

5.5 Examples

Finally, we investigated the initial utterances generated by **Separate (Gen+MMI)** and **Joint**. Ex-

news posts	initial utterances
A parade for the Rio Olympics and Paralympic medalists will be held in October. Approximately 500,000 people gathered at the time of the London Olympics.	**Separate (Gen+MMI):** I heard that a parade for the Rio Olympics and Paralympic medalists will be held in October. *That's amazing.*
	Joint: I heard that a parade for the Rio Olympics and Paralympic medalists will be held in October. *I would like to see what parade it is.*
A Chinese captain who was poaching a coral in the offshore of Kagoshima was arrested. The number of poaching boats has sharply declined in Ogasawara.	**Separate (Gen+MMI):** I heard that the coral was arrested in Kagoshima prefecture offshore because of poaching. *Get it together.*
	Joint: I heard that a Chinese captain who poached a coral in the offshore of Kagoshima was arrested. *Do not do poaching!*
On a suicide bombing that happened at a concert in England, the homeless action around the scene attracted praise. The UK raised the terrorist threat level to the "highest."	**Separate (Gen+MMI):** I heard that the homeless action around the scene of a suicide bombing that happened in England was praised. *That's scary.*
	Joint: I heard that the homeless action on a suicide bombing in England was praised. *Do you take measures against terrorism?*
The Meteorological Agency announced that Typhoon 11 will approach Tohoku in the early morning of 21st. Typhoon 9 is expected to approach Tokai on 21st and Typhoon 10 will approach Tokai or Kanto.	**Separate (Gen+MMI):** It seems that Typhoon 11 will approach Tohoku on 21st. *Let's watch out!*
	Joint: I heard that The Meteorological Agency announced that Typhoon 11 will approach Tohoku in the early morning of 21st. *We should pay attention to the future movement.*

Table 8: Examples of generated initial utterances. *Italics* are chit-chat parts.

amples of these utterances are provided in Table 8.

We found that **Separate (Gen+MMI)** tended to generate generic utterances (*e.g., "That's amazing", "Get it together"*) as the chit-chat part that fit any context, even though it uses a diversity-promoting function when decoding. In contrast, **Joint** could generate more diverse chit-chat parts by utilizing contents words such as *"parade"* and *"poaching"*. One possible reason for this phenomena is that the generated summary part acts like an additional condition of $P(T|S)$ at the time of decoding the chit-chat part. This does not happen with **Separate (Gen+MMI)**, which simply concatenates the outputs of separate models.

Interestingly, we found that there are some utterances giving a *question* (third example of **Joint** in Table 8) or making an *urging* (fourth example of **Separate (Gen+MMI)** in Table 8). Controlling utterances of the model by such dialogue acts (Wen et al., 2015; Zhao et al., 2017) can make the conversation initiation more diverse and attractive. We leave them as the future work at this time.

We should note that although it is a problem common to all the generation-models, there is a possibility of transmitting false news contents (as in the second example of **Separate (Gen+MMI)** in Table 8) or ethically inappropriate contents to the users. Therefore, when adopting our method into an actual conversation application, we have to pay close attention to this problem.

6 Conclusion

In this paper, we proposed the new task of conversation initiation. To generate diverse initial utterances that can improve user engagement, we utilized news articles that provide fresh and varied information every day and constructed a large-scale dataset using crowd workers. To perform the conversation initiation, we designed separate and joint approaches including both IR-based and generation-based methods. Empirical experiments showed that the proposed methods outperformed the baselines in both automatic and manual evaluation, and can generate diverse initial utterances that template-based methods cannot make. These results demonstrate the quality of our constructed dataset, that will be released for future studies[11].

As a natural next step, we plan to develop a more sophisticated conversation model, which can not only generate initial utterances but also continue the conversation for the given news contents (Yoshino and Kawahara, 2014). In that case, depending on the user's interest, the model needs to determine whether to do a usual chat or talk about the news contents. We also plan to improve the proposed method so that it can generate even better initial utterances. Since our task has two elements, summarization and chit-chat, the focus of our future work will be a more sophisticated multi-task model that considers these relations.

Acknowledgments

We thank Manabu Sassano for fruitful discussions and comments. We also thank the anonymous reviewers.

---

[11] https://research-lab.yahoo.co.jp/en/software/

References

Satoshi Akasaki and Nobuhiro Kaji. 2017. Chat detection in an intelligent assistant: Combining task-oriented and non-task-oriented spoken dialogue systems. In *Proceedings of ACL*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of ICCV*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceesdings of ICLR*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of EMNLP*, pages 3970–3980.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Jose MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of CVPR*, pages 1080–1089. IEEE.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *Proceedings of ACL*, pages 964–972.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, Umut Ozertem, Imed Zitouni, Ranjitha Gurunath Kulkarni, and Omar Zia Khan. 2015. Automatic online evaluation of intelligent assistants. In *Proceedings of WWW*, pages 506–516.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *In Proceedings of ACL*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of ACL*, pages 994–1003.

Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016c. Stalematebreaker: a proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of IJCAI*, pages 2845–2851.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of ACL Workshop*, pages 74–81.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of ACL*, volume 1, pages 1116–1126.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of EMNLP*, pages 2322–2332.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of ACL*, pages 1802–1813.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of EMNLP*, pages 2241–2252.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *In Proceedings of EMNLP*, pages 1532–1543.

Kristen Purcell, Lee Rainie, Amy Mitchell, Tom Rosenstiel, and Kenny Olmstead. 2010. Understanding the participatory news consumer. In *Pew Internet and American Life Project*.

Lianhui Qin, Lemao Liu, Wei Bi, Yan Wang, Xiaojiang Liu, Zhiting Hu, Hai Zhao, and Shuming Shi. 2018. Automatic article commenting: the task and dataset. In *Proceedings of ACL*, pages 151–156.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of ACL*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP*, pages 583–593.

Shumpei Sano, Nobuhiro Kaji, and Manabu Sassano. 2016. Prediction of prospective user engagement with intelligent assistants. In *Proceedings of ACL*, pages 1203–1212.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*, pages 1073–1083.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI*, pages 3295–3301.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Workshop*.

Richard S. Wallace. 2009. *The Anatomy of A.L.I.C.E.*, pages 181–210. Springer.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*, pages 1711–1721.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83.

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas, and Verena Rieser. 2018. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *Proceedings of EMNLP*, pages 3981–3991.

Rui Yan. 2018. "chitty-chitty-chat bot": Deep learning for conversational ai. In *Proceedings of IJCAI*, pages 5520–5526.

Rui Yan and Dongyan Zhao. 2018. Smarter response with proactive suggestion: A new generative neural conversation paradigm. In *Proceedings of IJCAI*, pages 4525–4531.

Rui Yan, Dongyan Zhao, et al. 2017. Joint learning of response ranking and next utterance suggestion in human-computer conversation system. In *Proceedings of SIGIR*, pages 685–694. ACM.

Koichiro Yoshino and Tatsuya Kawahara. 2014. Information navigation system based on pomdp that tracks user focus. In *Proceedings of SIGDIAL*, pages 32–40.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of ACL*, volume 1, pages 654–664.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A dataset for document grounded conversations. In *Proceedings of EMNLP*, pages 708–713.

Positional Encoding to Control Output Sequence Length

Sho Takase   **Naoaki Okazaki**
Tokyo Institute of Technology
`{sho.takase@nlp.c, okazaki@c}.titech.ac.jp`

Abstract

Neural encoder-decoder models have been successful in natural language generation tasks. However, real applications of abstractive summarization must consider additional constraint that a generated summary should not exceed a desired length. In this paper, we propose a simple but effective extension of a sinusoidal positional encoding (Vaswani et al., 2017) to enable neural encoder-decoder model to preserves the length constraint. Unlike in previous studies where that learn embeddings representing each length, the proposed method can generate a text of any length even if the target length is not present in training data. The experimental results show that the proposed method can not only control the generation length but also improve the ROUGE scores.

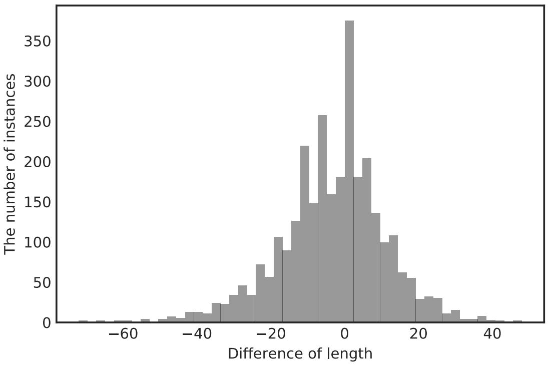


Figure 1: Difference in number of characters between correct headlines and outputs of a widely used LSTM encoder-decoder (Luong et al., 2015) which is trained on sentence-headline pairs created by Rush et al. (2015) from the annotated English Gigaword corpus. The difference was investigated for 3,000 sentence-headline pairs randomly sampled from the test splits.

1 Introduction

Neural encoder-decoder models have been successfully applied to various natural language generation tasks including machine translation (Sutskever et al., 2014), summarization (Rush et al., 2015), and caption generation (Vinyals et al., 2015). Still, it is necessary to control the output length for abstractive summarization, which generates a summary for a given text while satisfying a space constraint. In fact, Figure 1 shows a large variance in output sequences produced by a widely used encoder-decoder model (Luong et al., 2015), which has no mechanism for controlling the length of the output sequences.

Fan et al. (2018) trained embeddings that correspond to each output length to control the output sequence length. Since the embeddings for different lengths are independent, it is hard to generate a sequence of the length that is infrequent in training data. Thus, a method that can model any lengths continuously is required.

Kikuchi et al. (2016) proposed two learning based methods for an LSTM encoder-decoder: LenEmb and LenInit. LenEmb inputs an embedding representing the remaining length in each decoding step. Since this approach also prepares embeddings for each length independently, it suffers from the same problem as that in Fan et al. (2018).

On the other hand, LenInit can handle arbitrary lengths because it combines the scalar value of a desired length with a trainable embedding. LenInit initializes the LSTM cell of the decoder with the embedding depending on the scalar value of the desired length. Liu et al. (2018) incorporated such scalar values into the initial state of the decoder in a CNN encoder-decoder. These approaches deal with any length but it is reasonable to incorporate the distance to the desired terminal position into each decoding step such as in LenEmb.

In this study, we focused on Transformer (Vaswani et al., 2017), which recently achieved the state-of-the-art score on the machine

Proceedings of NAACL-HLT 2019, pages 3999–4004
Minneapolis, Minnesota, June 2 - June 7, 2019.

translation task. We extend the sinusoidal positional encoding, which represents a position of each token in Transformer (Vaswani et al., 2017), to represent a distance from a terminal position on the decoder side. In this way, the proposed method considers the remaining length explicitly at each decoding step. Moreover, the proposed method can handle any desired length regardless of its appearance in a training corpus because it uses the same continuous space for any length.

We conduct experiments on the headline generation task. The experimental results show that our proposed method is able to not only control the output length but also improve the ROUGE scores from the baselines. Our code and constructed test data are publicly available at: https://github.com/takase/control-length.

2 Positional Encoding

Transformer (Vaswani et al., 2017) uses a sinusoidal positional encoding to represent the position of an input. Transformer feeds the sum of the positional encoding and token embedding to the input layer of its encoder and decoder. Let pos be the position and d be the embedding size. Then, the i-th dimension of the sinusoidal positional encoding $PE_{(pos,i)}$ is as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \quad (1)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right). \quad (2)$$

In short, each dimension of the positional encoding corresponds to a sinusoid whose period is $10000^{2i/d} \times 2\pi$. Since this function returns an identical value at the same position pos, the above positional encoding can be interpreted as representing the absolute position of each input token.

In this paper, we extend Equations (1) and (2) to depend on the given output length and the distance from the terminal position. We propose two extensions: length-difference positional encoding ($LDPE$) and length-ratio positional encoding ($LRPE$). Then we replace Equations (1) and (2) with (3) and (4) (or (5) and (6)) on the decoder side to control the output sequence length. We define $LDPE$ and $LRPE$ as follows:

$$LDPE_{(pos,len,2i)} = \sin\left(\frac{len - pos}{10000^{\frac{2i}{d}}}\right), \quad (3)$$

$$LDPE_{(pos,len,2i+1)} = \cos\left(\frac{len - pos}{10000^{\frac{2i}{d}}}\right), \quad (4)$$

$$LRPE_{(pos,len,2i)} = \sin\left(\frac{pos}{len^{\frac{2i}{d}}}\right), \quad (5)$$

$$LRPE_{(pos,len,2i+1)} = \cos\left(\frac{pos}{len^{\frac{2i}{d}}}\right), \quad (6)$$

where len presents the given length constraint. $LDPE$ returns an identical value at the position where the remaining length to the terminal position is the same. $LRPE$ returns a similar value at the positions where the ratio of the remaining length to the terminal position is similar. Let us consider the d-th dimension as the simplest example. Since we obtain $\sin(pos/len)$ (or $\cos(pos/len)$) at this dimension, the equations yield the same value when the remaining length ratio is the same, e.g., $pos = 5$, $len = 10$ and $pos = 10$, $len = 20$.

We add $LDPE$ (or $LRPE$) to the input layer of Transformer in the same manner as in Vaswani et al. (2017). In the training step, we assign the length of the correct output to len. In the test phase, we control the output length by assigning the desired length to len.

3 Experiments

3.1 Datasets

We conduct experiments on the headline generation task on Japanese and English datasets. The purpose of the experiments is to evaluate the ability of the proposed method to generate a summary of good quality within a specified length. We used JAMUL corpus as the Japanese test set (Hitomi et al., 2019). This test set contains three kinds of headlines for 1,181[1] news articles written by professional editors under the different upper bounds of headline lengths. The upper bounds are 10, 13, and 26 characters ($len = 10, 13, 26$). This test set is suitable for simulating the real process of news production because it is constructed by a Japanese media company.

In contrast, we have no English test sets that contain headlines of multiple lengths. Thus, we randomly extracted 3,000 sentence-headline

---

[1] We obtained this test set by applying the pre-processing script at https://github.com/asahi-research/Gingo to the original JAMUL corpus.

pairs that satisfy a length constraint from the test set constructed from annotated English Gigaword (Napoles et al., 2012) by pre-processing scripts of Rush et al. (2015)[2]. We set three configurations for the number of characters as the length constraint: 0 to 30 characters ($len = 30$), 30 to 50 characters ($len = 50$), and 50 to 75 characters ($len = 75$). Moreover, we also evaluate the proposed method on the DUC-2004 task 1 (Over et al., 2007) for comparison with published scores in previous studies.

Unfortunately, we have no large supervision data with multiple headlines of different lengths associated with each news article in both languages. Thus, we trained the proposed method on pairs with a one-to-one correspondences between the source articles and headlines. In the training step, we regarded the length of the target headline as the desired length len. For Japanese, we used the JNC corpus, which contains a pair of the lead three sentences of a news article and its headline (Hitomi et al., 2019). The training set contains about 1.6M pairs[3]. For English, we used sentence-headline pairs extracted from the annotated English Gigaword with the same pre-processing script used in the construction of the test set. The training set contains about 3.8M pairs.

In this paper, we used a character-level decoder to control the number of characters. On the encoder side, we used subword units to construct the vocabulary (Sennrich et al., 2016; Kudo, 2018). We set the hyper-parameter to fit the vocabulary size to about 8k for Japanese and 16k for English.

3.2 Baselines

We implemented two methods proposed by previous studies to control the output length and handle arbitrary lengths. We employed them and Transformer as baselines.

LenInit Kikuchi et al. (2016) proposed LenInit, which controls the output length by initializing the LSTM cell m of the decoder as follows:

$$m = len \times b, \tag{7}$$

where b is a trainable vector. We incorporated this method with a widely used LSTM encoder-decoder model (Luong et al., 2015)[4]. For a fair comparison, we set the same hyper-parameters as in Takase et al. (2018) because they indicated that the LSTM encoder-decoder model trained with the hyper-parameters achieved a similar performance to the state-of-the-art on the headline generation.

Length Control (LC) Liu et al. (2018) proposed a length control method that multiplies the desired length by input token embeddings. We trained the model with their hyper-parameters.

Transformer Our proposed method is based on Transformer (Vaswani et al., 2017)[5]. We trained Transformer with the equal hyper-parameters as in the base model in Vaswani et al. (2017).

3.3 Results

Table 1 shows the recall-oriented ROUGE-1 (R-1), 2 (R-2), and L (R-L) scores of each method on the Japanese test set[6]. This table indicates that Transformer with the proposed method (Transformer+$LDPE$ and Transformer+$LRPE$) outperformed the baselines for all given constraints ($len = 10, 13, 26$). Transformer+$LRPE$ performed slightly better than Transformer+$LDPE$. Moreover, we improved the performance by incorporating the standard sinusoidal positional encoding (+PE) on $len = 10$ and 26. The results imply that the absolute position also helps to generate better headlines while controlling the output length.

Table 2 shows the recall-oriented ROUGE scores on the English Gigaword test set. This table indicates that $LDPE$ and $LRPE$ significantly improved the performance on $len = 75$. Moreover, the absolute position (PE) also improved the performance in this test set. In particular, PE was very effective in the setting of very short headlines ($len = 30$). However, the proposed method slightly lowered ROUGE-2 scores from the bare Transformer on $len = 30, 50$. We infer that the bare Transformer can generate headlines whose lengths are close to 30 and 50 because the majority of the training set consists of headlines whose lengths are less than or equal to 50. However, most of the generated headlines breached the length constraints, as explained in Section 3.4.

To investigate whether the proposed method can generate good headlines for unseen lengths, we excluded headlines whose lengths are equal to the

---

[2] https://github.com/facebookarchive/NAMAS

[3] We obtained this training set by applying the pre-processing script at https://github.com/asahi-research/Gingo.

[4] We used an implementation at https://github.com/mlpnlp/mlpnlp-nmt.

[5] We used an implementation at https://github.com/pytorch/fairseq.

[6] To calculate ROUGE scores on the Japanese dataset, we used https://github.com/asahi-research/Gingo.

	$len = 10$			$len = 13$			$len = 26$		
Model	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Baselines									
LenInit	38.08	17.72	36.84	41.83	19.53	39.22	47.07	22.02	38.36
LC	35.88	15.73	34.80	40.28	18.86	38.16	42.62	19.38	35.61
Transformer	34.63	15.48	33.02	43.94	21.35	40.77	46.43	23.03	38.10
Proposed method									
Transformer+$LDPE$	42.84	21.07	41.31	46.51	22.83	43.76	50.89	24.18	40.82
+PE	42.85	20.67	41.47	46.72	22.70	43.75	**51.32**	**25.15**	**41.48**
Transformer+$LRPE$	42.70	21.62	41.35	**47.05**	**23.70**	**44.13**	50.68	24.70	41.23
+PE	**43.36**	**21.63**	**41.93**	46.39	23.09	43.49	51.21	25.03	41.43
Proposed method trained on the dataset without headlines consisting of target lengths									
Transformer+$LDPE$	41.91	20.01	40.69	45.88	22.61	43.16	50.90	24.37	40.48
+PE	42.33	20.46	40.88	44.78	22.33	42.27	50.87	24.54	40.89
Transformer+$LRPE$	41.91	20.10	40.52	46.01	22.87	43.47	50.33	24.37	41.00
+PE	42.59	20.76	41.16	46.52	23.65	43.81	50.73	24.64	41.01

Table 1: Recall-oriented ROUGE scores for each length on Japanese test set. This test set contains three kinds of headlines, i.e., $len = 10, 13, 26$, tied to a single article.

	$len = 30$			$len = 50$			$len = 75$		
Model	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Baselines									
LenInit	44.58	25.90	43.34	48.42	25.47	45.56	50.78	25.74	46.42
LC	45.17	26.73	44.09	46.56	24.55	44.10	48.67	24.83	44.98
Transformer	47.48	**29.77**	46.17	50.02	**28.04**	47.29	47.31	24.83	43.75
Proposed method									
Transformer+$LDPE$	47.26	26.98	45.77	50.21	26.13	47.15	53.99	27.78	49.24
+PE	48.13	27.18	46.43	50.29	25.97	47.17	53.65	27.65	49.06
Transformer+$LRPE$	48.79	28.77	47.17	50.09	26.08	46.91	53.91	27.82	49.15
+PE	**49.23**	29.26	**47.68**	50.41	26.37	47.39	**54.21**	**27.84**	**49.38**
Proposed method trained on the dataset without headlines consisting of the target lengths									
Transformer+$LDPE$	47.35	26.76	45.70	50.46	25.96	47.30	53.69	27.61	49.04
+PE	47.44	27.42	45.99	50.67	26.07	47.57	53.76	27.53	49.03
Transformer+$LRPE$	48.54	28.89	47.06	50.65	26.19	47.34	53.94	27.88	49.11
+PE	49.08	29.09	47.58	**50.78**	26.64	**47.60**	53.77	27.68	48.93

Table 2: Recall-oriented ROUGE scores for each length on test data extracted from annotated English Gigaword.

Model	R-1	R-2	R-L
Baselines			
LenInit	29.78	11.05	26.49
LC	28.68	10.79	25.72
Transformer	26.15	9.14	23.19
Proposed method			
Transformer+$LDPE$	30.95	10.53	26.79
+PE	31.00	10.78	27.02
+Re-ranking	31.65	11.25	27.46
Transformer+$LRPE$	30.74	10.83	26.69
+PE	31.10	11.05	27.25
+Re-ranking	**32.29**	**11.49**	**28.03**
Previous studies for controlling output length			
Kikuchi et al. (2016)	26.73	8.39	23.88
Fan et al. (2018)	30.00	10.27	26.43
Other previous studies			
Rush et al. (2015)	28.18	8.49	23.81
Suzuki and Nagata (2017)	32.28	10.54	27.80
Zhou et al. (2017)	29.21	9.56	25.51
Li et al. (2017)	31.79	10.75	27.48
Li et al. (2018)	29.33	10.24	25.24

Table 3: Recall-oriented ROUGE scores in DUC-2004.

desired length (len) from the training data. The lower parts of Table 1 and 2 show ROUGE scores of the proposed method trained on the modified training data. These parts show that the proposed method achieved comparable scores to ones trained on whole training dataset. These results indicate that the proposed method can generate high-quality headlines even if the length does not appear in the training data.

Table 3 shows the recall-oriented ROUGE scores on the DUC-2004 test set. Following the evaluation protocol (Over et al., 2007), we truncated characters over 75 bytes. The table indicates that $LDPE$ and $LRPE$ significantly improved the performance compared to the bare Transformer, and achieved better performance than the baselines except for R-2 of LenInit. This table also shows the scores reported in the previous studies. The proposed method outperformed the previous methods that control the output length and achieved the competitive score to the state-of-the-art scores.

Since the proposed method consists of a character-based decoder, it sometimes generated

	Variance					
	Japanese dataset			English Gigaword		
Model	$len = 10$	$len = 13$	$len = 26$	$len = 30$	$len = 50$	$len = 75$
Baselines						
LenInit	0.047	0.144	0.058	0.114	0.112	0.091
LC	0.021	0.028	0.040	0.445	0.521	0.871
Transformer	181.261	115.431	38.169	193.119	138.566	620.887
Proposed method						
Transformer+$LDPE$	**0.000**	**0.000**	**0.000**	**0.015**	0.012	0.013
+PE	0.003	0.001	0.001	0.016	**0.009**	**0.007**
Transformer+$LRPE$	0.121	0.210	0.047	0.082	0.071	0.187
+PE	0.119	0.144	0.058	0.142	0.110	0.173
Proposed method trained on the dataset without headlines consisting of the target lengths						
Transformer+$LDPE$	**0.000**	0.002	**0.000**	0.018	**0.009**	0.009
+PE	0.021	0.001	0.003	0.021	0.013	0.010
Transformer+$LRPE$	0.191	0.362	0.043	0.120	0.058	0.133
+PE	0.183	0.406	0.052	0.138	0.081	0.154

Table 4: Variances of generated headlines.

words unrelated to a source sentence. Thus, we applied a simple re-ranking to each n-best headlines generated by the proposed method ($n = 20$ in this experiment) based on the contained words. Our re-ranking strategy selects a headline that contains source-side words the most. Table 3 shows that Transformer+$LRPE$+PE with this re-ranking (+Re-ranking) achieved better scores than the state-of-the-art (Suzuki and Nagata, 2017).

3.4 Analysis of Output Length

Following Liu et al. (2018), we used the variance of the generated summary lengths against the desired lengths as an indicator of the preciseness of the output lengths. We calculated variance (var) for n generated summaries as follows[7]:

$$var = \frac{1}{n}\sum_{i=1}^{n} |l_i - len|^2, \qquad (8)$$

where len is the desired length and l_i is the length of the generated summary.

Table 4 shows the values of Equation (8) computed for each method and the desired lengths. This table indicates that $LDPE$ could control the length of headlines precisely. In particular, $LDPE$ could generate headlines with the identical length to the desired one in comparison with LenInit and LC. $LRPE$ also generated headlines with a precise length but its variance is larger than those of previous studies in very short lengths, i.e., $len = 10$ and 13 in Japanese. However, we consider $LRPE$ is enough for real applications because the averaged difference between its output and the desired length is small, e.g., 0.1 for $len = 10$.

The lower part of Table 4 shows the variances of the proposed method trained on the modified training data that does not contain headlines whose lengths are equal to the desired length, similar to the lower parts of Table 1 and 2. The variances for this part are comparable to the ones obtained when we trained the proposed method with whole training dataset. This fact indicates that the proposed method can generate an output that satisfies the constraint of the desired length even if the training data does not contain instances of such a length.

4 Conclusion

In this paper, we proposed length-dependent positional encodings, $LDPE$ and $LRPE$, that can control the output sequence length in Transformer. The experimental results demonstrate that the proposed method can generate a headline with the desired length even if the desired length is not present in the training data. Moreover, the proposed method significantly improved the quality of headlines on the Japanese headline generation task while preserving the given length constraint. For English, the proposed method also generated headlines with the desired length precisely and achieved the top ROUGE scores on the DUC-2004 test set.

Acknowledgments

The research results have been achieved by "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation", the Commissioned Research of National Institute of Information and Communications Technology (NICT), Japan.

---

[7] Liu et al. (2018) multiplies Equation (8) by 0.001.

References

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation (WMT 2018)*, pages 45–54.

Yuta Hitomi, Yuya Taguchi, Hideaki Tamori, Ko Kikuta, Jiro Nishitoba, Naoaki Okazaki, Inui Kentaro, and Manabu Okumura. 2019. A large-scale multi-length headline corpus for improving length-constrained headline generation model evaluation. *CoRR*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1328–1338.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 66–75.

Haoran Li, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 1430–1441.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2091–2100.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4110–4119.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1412–1421.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 291–297.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4599–4609.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, pages 3156–3164.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104.

The Lower The Simpler: Simplifying Hierarchical Recurrent Models

Chao Wang and Hui Jiang
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
$\{chwang, hj\}$@eecs.yorku.ca

Abstract

To improve the training efficiency of hierarchical recurrent models without compromising their performance, we propose a strategy named as "the lower the simpler", which is to simplify the baseline models by making the lower layers simpler than the upper layers. We carry out this strategy to simplify two typical hierarchical recurrent models, namely Hierarchical Recurrent Encoder-Decoder (HRED) and R-NET, whose basic building block is GRU. Specifically, we propose Scalar Gated Unit (SGU), which is a simplified variant of GRU, and use it to replace the GRUs at the middle layers of HRED and R-NET. Besides, we also use Fixed-size Ordinally-Forgetting Encoding (FOFE), which is an efficient encoding method without any trainable parameter, to replace the GRUs at the bottom layers of HRED and R-NET. The experimental results show that the simplified HRED and the simplified R-NET contain significantly less trainable parameters, consume significantly less training time, and achieve slightly better performance than their baseline models.

1 Introduction

With the advance of various deep learning frameworks, neural network based models proposed for natural language understanding tasks are becoming increasingly complicated. To the best of our knowledge, a considerable part of these complicated models are both hierarchical and recurrent. For example, Hierarchical Recurrent Encoder-Decoder (HRED) (Sordoni et al., 2015; Serban et al., 2016), which is a conversational model, is constructed by stacking three layers of GRUs (Cho et al., 2014). Besides, several well-known Machine Reading Comprehension (MRC) models, such as R-NET (Wang et al., 2017) and Fusion-Net (Huang et al., 2017), are mainly composed of multiple layers of bidirectional GRUs (BiGRUs) or bidirectional LSTMs (BiLSTMs) (Hochreiter and Schmidhuber, 1997). The above hierarchical recurrent models have achieved excellent performance, but training them usually consumes a lot of time and memory, that is because their computational graphs contain a large amount of operators and trainable parameters, which makes their training computationally expensive.

According to Williams and Zipser (1995), in the training of recurrent neural networks, it is the backward propagation rather than the forward propagation that consumes the majority of the computational resources. Besides, considering the chain rule in the backward propagation, the complexity of computing gradients for a hierarchical recurrent model increases exponentially from the top layer of the model down to the bottom layer. Therefore, to improve the training efficiency of hierarchical recurrent models, our strategy is to **simplify the baseline models by making the lower layers simpler than the upper layers**, which we name as "**the lower the simpler**". Here "simpler" means containing less operators and trainable parameters. This strategy is guaranteed to work, since it can accelerate the computation of gradients, which is the substance of the backward propagation. However, there is still a big concern: once the baseline models are simplified, will their performance be compromised?

To address this concern, we carry out our proposed strategy to simplify two typical hierarchical recurrent models, namely HRED and R-NET, whose basic building block is GRU. Specifically, we propose Scalar Gated Unit (SGU), which is a simplified variant of GRU, and use it to replace the GRUs at the middle layers of HRED and R-NET. Besides, we also use Fixed-size Ordinally-Forgetting Encoding (FOFE) (Zhang et al., 2015), which is an efficient encoding method without any trainable parameter, to replace the GRUs at the bottom

Proceedings of NAACL-HLT 2019, pages 4005–4009
Minneapolis, Minnesota, June 2 - June 7, 2019.

layers of HRED and R-NET. In the experiments, we separately compare the simplified HRED and the simplified R-NET with their baseline models in terms of both the training efficiency and the performance. The experimental results show that the simplified models contain significantly less trainable parameters, consume significantly less training time, and achieve slightly better performance than their baseline models.

2 Baseline Models

2.1 Hierarchical Recurrent Encoder-Decoder

Hierarchical Recurrent Encoder-Decoder (HRED) is a conversational model for building end-to-end dialogue systems. Since a dialogue is a sequence of sentences, where each sentence is a sequence of words, HRED models this hierarchy with a hierarchical recurrent structure. Specifically, HRED consists of three layers of GRUs, which from bottom to top separately serve as the sentence-level encoder, the dialogue-level encoder, and the decoder. The sentence-level encoder GRU iteratively takes the embeddings of the words in a sentence to update its hidden state, thus its final hidden state is a representation of the sentence. The dialogue-level encoder GRU iteratively takes the representations of the sentences in a dialogue to update its hidden state, thus its hidden state at each time-step is a representation of the current dialogue. The decoder GRU takes the current dialogue representation to initialize its hidden state so as to generate a response sentence word by word.

2.2 R-NET

R-NET is an end-to-end MRC model that predicts an answer span for each given passage-question pair. Specifically, R-NET consists of five layers, which from bottom to top are separately the embedding layer, the encoding layer, the matching layer, the self-matching layer, and the output layer. The embedding layer maps the words to the word-level embeddings and the character-level embeddings. The character-level embeddings are generated by processing the character embeddings of the words with a BiGRU and concatenating the forward GRU final hidden states and the backward GRU final hidden states. The encoding layer processes the concatenation of the word-level embeddings and the character-level embeddings with another BiGRU and concatenates the forward GRU outputs and the backward GRU outputs so as to generate the context representations. The matching layer uses a gated attention-based BiGRU to fuse the context representations of the question into those of the passage so as to generate the question-aware passage representations. The self-matching layer uses another gated attention-based BiGRU to fuse the question-aware passage representations into themselves so as to generate the final passage representations. On this basis, the output layer uses a pointer network (Vinyals et al., 2015) to generate an answer span.

3 Model Simplification

3.1 Scalar Gated Unit

Just like LSTM, GRU is a recurrent structure that leverages gating mechanisms to capture long-term dependencies in sequential data:

$$\begin{aligned}
&\text{Update Gate: } z_t = \sigma(W_z[h_{t-1}, x_t]) \\
&\text{Reset Gate: } r_t = \sigma(W_r[h_{t-1}, x_t]) \\
&\text{New Memory: } \hat{h}_t = tanh(W_h[r_t \odot h_{t-1}, x_t]) \\
&\text{Hidden State: } h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t
\end{aligned}$$

Researchers have proposed several simplified variants of GRU. For example, Zhou et al. (2016) proposed Minimal Gated Unit (MGU), which combines the update gate and the reset gate into a single forget gate. Compared with GRU, MGU contains less trainable parameters, consumes less training time, and achieves similar performance. However, in this paper, to better carry out our proposed "the lower the simpler" strategy, we propose Scalar Gated Unit (SGU), which is an even more simplified variant of GRU:

$$\begin{aligned}
&\text{Scalar Update Gate: } z_t = \sigma(w_z[h_{t-1}, x_t]) \\
&\text{Scalar Reset Gate: } r_t = \sigma(w_r[h_{t-1}, x_t]) \\
&\text{New Memory: } \hat{h}_t = tanh(W_h[r_t * h_{t-1}, x_t]) \\
&\text{Hidden State: } h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t
\end{aligned}$$

By comparing the formulation of SGU with that of GRU, it is easy to see that both the update gate z_t and the reset gate r_t change from the vectors in GRU to the scalars in SGU. Accordingly, the weights for generating the gates change from the matrices W_z and W_r in GRU to the vectors w_z and w_r in SGU. Besides, the gating operator also changes from the element-wise multiplication $\odot$ in GRU to the scalar multiplication $*$ in SGU. Therefore SGU is guaranteed to be the simplest among all the variants of GRU.

3.2 Fixed-size Ordinally-Forgetting Encoding

Fixed-size Ordinally-Forgetting Encoding (FOFE) is an encoding method that uses the following recurrent structure to map a varied-length word sequence to a fixed-size representation:

$$h_t = \begin{cases} 0, & \text{if } t = 0 \\ \alpha * h_{t-1} + x_t, & \text{otherwise} \end{cases}$$

where h_t is the hidden state at time step t, x_t is the embedding of the t-th word, and α $(0 < \alpha < 1)$ is the forgetting factor that decays the previous hidden state. Given a word sequence of length N, the final hidden state h_N is a fixed-size representation of the word sequence. Although formulated as a recurrent structure, FOFE can actually be implemented with an efficient matrix multiplication. Besides, the forgetting factor α is designed as a hyper-parameter so that FOFE contains no trainable parameter. Therefore FOFE is guaranteed to be the simplest among all the recurrent structures. As for the performance, according to Zhang et al. (2015), FOFE based language models outperform their LSTM based competitors.

3.3 Simplified Models

According to the above descriptions, SGU is simpler than GRU, and FOFE is simpler than SGU. Therefore, now we can carry out our proposed "the lower the simpler" strategy by using SGUs and FOFEs to replace certain GRUs in HRED and R-NET. For HRED, we keep the decoder GRU at the top layer unchanged, use a SGU to replace the dialogue-level encoder GRU at the middel layer, and use a FOFE to replace the sentence-level encoder GRU at the bottom layer. For R-NET, we keep the output layer, the self-matching layer, and the matching layer unchanged, use a bidirectional SGU (BiSGU) to replace the BiGRU that generates context representations at the encoding layer, and use a bidirectional FOFE (BiFOFE, i.e., running FOFE both forward and backward) to replace the BiGRU that generates character-level embeddings at the embedding layer. After conducting the above replacements, we finally obtain a simplified HRED and a simplified R-NET.

4 Experiments

4.1 Experimental Datasets

Dialogue Datasets. We compare the simplified HRED with the baseline HRED on two dialogue datasets, namely MovieTriples (Serban et al., 2016) and Ubuntu (Lowe et al., 2017). MovieTriples contains over $240,000$ dialogues collected from various movie scripts, with each dialogue consisting of three sentences. Ubuntu contains over $490,000$ dialogues collected from the Ubuntu chat-logs, with each dialogue consisting of seven sentences on average. Both MovieTriples and Ubuntu have been randomly partitioned into three parts: a training set (80%), a development set (10%), and a test set (10%).

MRC Dataset. We compare the simplified R-NET with the baseline R-NET on an MRC dataset, namely SQuAD (Rajpurkar et al., 2016). SQuAD contains over $100,000$ passage-question pairs with human-generated answer spans, where the passages are collected from Wikipedia, and the answer to each question is guaranteed to be a fragment in the corresponding passage. Besides, SQuAD has also been randomly partitioned into three parts: a training set (80%), a development set (10%), and a test set (10%). Both the training set and the development set are publicly available, but the test set is confidential.

4.2 Implementation Details

HRED. We implement both the simplified HRED and the baseline HRED with TensorFlow (Abadi et al., 2016). For the word embeddings, we set their size to 200, 400, and 600 on MovieTriples and 600 on Ubuntu, initialize them randomly, and update them during the training. For the forgetting factor α of FOFE, we set it to 0.9 on both MovieTriples and Ubuntu. For the hidden state size of the sentence-level encoder GRU, we set it to 200, 400, and 600 on MovieTriples and 600 on Ubuntu. For the hidden state size of the dialogue-level encoder GRU and SGU, we set it to 1200 on both MovieTriples and Ubuntu. For the hidden state size of the decoder GRU, we set it to 200, 400, and 600 on MovieTriples and 600 on Ubuntu. For model optimization, we apply the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.0001 and a mini-batch size of 32. For performance evaluation, we use both perplexity and error rate as evaluation metrics.

R-NET. We implement both the simplified R-NET and the baseline R-NET with TensorFlow. For the word-level embeddings, we initialize them with the 300-dimensional pre-trained GloVe (Pennington et al., 2014) vectors, and fix them during the

Model	Word Embedding	Hidden States (bottom-up)	Trainable Parameters	Training Time (secs * epochs)	Performance (ppl, err rate)
Baseline HRED	200	200-1200-200	10,777,003	4,100 * 33	35.72, 66.62%
	400	400-1200-400	18,740,403	4,660 * 29	34.35, 66.13%
	600	600-1200-600	28,223,803	5,700 * 29	34.11, 65.95%
Simplified HRED	200	200-1200-200	6,456,605	2,030 * 35	35.14, 66.46%
	400	400-1200-400	12,019,605	2,210 * 30	34.01, 66.05%
	600	600-1200-600	18,142,605	2,590 * 29	33.79, 65.89%

Table 1: Comparing the simplified HRED with the baseline HRED on MovieTriples.

Model	Word Embedding	Hidden States (bottom-up)	Trainable Parameters	Training Time (secs * epochs)	Performance (ppl, err rate)
Baseline HRED	600	600-1200-600	40,231,401	51,770 * 33	46.29, 68.76%
Simplified HRED	600	600-1200-600	30,150,203	21,690 * 33	45.55, 68.55%

Table 2: Comparing the simplified HRED with the baseline HRED on Ubuntu.

Model	Trainable Parameters	Training Time (secs * epochs)	Dev Performance (EM / F1)
Baseline R-NET	2,307,991	2454 * 41	71.1 / 79.5
Simplified R-NET	2,007,435	2085 * 38	71.2 / 79.7

Table 3: Comparing the simplified HRED with the baseline HRED on SQuAD.

training. For the character embeddings, we initialize them with the same pre-trained GloVe vectors, and update them during the training. For the forgetting factor α of FOFE, we set it to 0.7. For the hidden state size of both the BiGRUs and the BiSGU, we set it to 128. For model optimization, we apply the Adam optimizer with a learning rate of 0.0005 and a mini-batch size of 32. For performance evaluation, we use both Exact Match (EM) and F1 score as evaluation metrics, which are calculated on the development set.

4.3 Experimental Results

For model comparison in the training efficiency, we use the same hardware (i.e., Intel Core i7-6700 CPU and NVIDIA GeForce GTX 1070 GPU) to train both the baseline models and the simplified models. The experimental results show that our proposed "the lower the simpler" strategy improves the training efficiency of both HRED and R-NET without compromising their performance. On the one hand, as shown in Table 1 and Table 2, the simplified HRED contains 25%–35% less trainable parameters, consumes over 50% less training time, and achieves slightly better performance than the baseline HRED. Besides, Table 1 also shows that appropriately scaling up the model brings better performance but consumes more resource, which implies that the simplified HRED will perform better than the baseline HRED when time or memory is limited. On the other hand, as shown in Table 3, the simplified R-NET contains 13% less trainable parameters, consumes 21% less training time, and achieves slightly better performance than the baseline R-NET.

5 Conclusion

In this paper, we propose a strategy named as "the lower the simpler", which is aimed at improving the training efficiency of hierarchical recurrent models without compromising their performance. This strategy has been verified on two typical hierarchical recurrent models, namely HRED and R-NET, where we replace their middle layers and bottom layers with two simpler recurrent structures. The significance of this paper lies in that it reveals a methodology for avoiding unnecessary complexity in training hierarchical recurrent models, which we believe is applicable to many other hierarchical recurrent models.

Acknowledgments

This work is partially supported by a research donation from iFLYTEK Co., Ltd., Hefei, China, and a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Ronald J Williams and David Zipser. 1995. Gradient-based learning algorithms for recurrent. *Backpropagation: Theory, architectures, and applications*, 433.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 495–500.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. 2016. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234.

Using Natural Language Relations between Answer Choices for Machine Comprehension

Rajkumar Pujari
Department of Computer Science
Purdue University
rpujari@purdue.edu

Dan Goldwasser
Department of Computer Science
Purdue University
dgoldwas@purdue.edu

Abstract

When evaluating an answer choice for Reading Comprehension task, other answer choices available for the question and the answers of related questions about the same paragraph often provide valuable information. In this paper, we propose a method to leverage the natural language relations between the answer choices, such as entailment and contradiction, to improve the performance of machine comprehension. We use a stand-alone question answering (QA) system to perform QA task and a Natural Language Inference (NLI) system to identify the relations between the choice pairs. Then we perform inference using an Integer Linear Programming (ILP)-based relational framework to re-evaluate the decisions made by the standalone QA system in light of the relations identified by the NLI system. We also propose a multitask learning model that learns both the tasks jointly.

1 Introduction

Given an input text and a set of related questions with multiple answer choices, the reading comprehension (RC) task evaluates the correctness of each answer choice. Current approaches to the RC task quantify the relationship between each question and answer choice independently and pick the highest scoring option. In this paper, we follow the observation that when humans approach such RC tasks, they tend to take a holistic view ensuring that their answers are consistent across the given questions and answer choices. In this work we attempt to model these pragmatic inferences, by leveraging the *entailment* and *contradiction* relations between the answer choices to improve machine comprehension. To help clarify these concepts, consider the following examples:

How can the military benefit from the existence of the CIA?
c_1: They can use them
c_2: These agencies are keenly attentive to the military's strategic and tactical requirements (✗)
c_3: The CIA knows what intelligence the military requires and has the resources to obtain that intelligence (✓)

The above example contains multiple correct answer choices, some are easier to capture than others. For example, identifying that c_3 is true might be easier than c_2 based on its alignment with the input text. However, capturing that c_3 entails c_2 allows us to predict c_2 correctly as well. Classification of the answer in red (marked ✗) could be corrected using the blue (marked ✓) answer choice.

Q1: *When were the eggs added to the pan to make the omelette?*
c_1^1: When they turned on the stove
c_2^1: When the pan was the right temperature (✓)
Q2: *Why did they use stove to cook omelette?*
c_1^2: They didn't use the stove but a microwave
c_2^2: Because they needed to heat up the pan (✗)

Similarly, answering Q1 correctly helps in answering Q2. Our goal is to leverage such inferences for machine comprehension.

Our approach contains three steps. First, we use a stand-alone QA system to classify the answer choices as true/false. Then, we classify the relation between each pair of choices for a given question as *entailment*, *contradiction* or *neutral*. Finally, we re-evaluate the labels assigned to choices using an Integer Linear Programming based inference procedure. We discuss different training protocols and representation choices for the combined decision problem. An overview is in figure 1.

We empirically evaluate on two recent datasets, MultiRC (Khashabi et al., 2018) and SemEval-

Proceedings of NAACL-HLT 2019, pages 4010–4015
Minneapolis, Minnesota, June 2 - June 7, 2019.

2018 task-11 (Ostermann et al., 2018) and show that it improves machine comprehension in both.

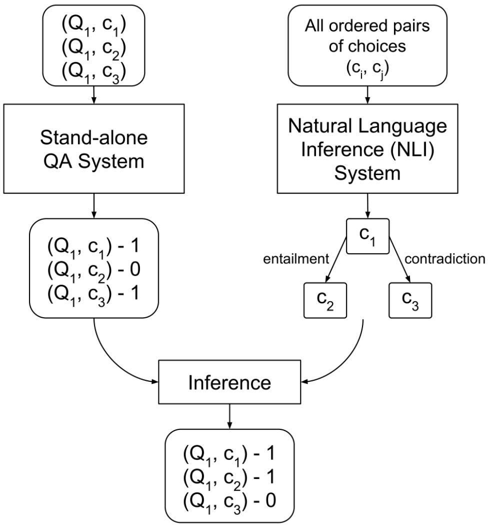


Figure 1: Proposed Approach

2 Related Work

Recently, several QA datasets have been proposed to test machine comprehension (Richardson, 2013; Weston et al., 2015; Rajpurkar et al., 2016; Trischler et al., 2016a; Nguyen et al., 2016). Yatskar (2018) showed that a high performance on these datasets could be achieved without necessarily achieving the capability of making commonsense inferences. Trischler et al. (2016b), Kumar et al. (2016), Liu and Perez (2017), Min et al. (2018) and Xiong et al. (2016) proposed successful models on those datasets. To address this issue, new QA datasets which require commonsense reasoning have been proposed (Khashabi et al., 2018; Ostermann et al., 2018; Mihaylov et al., 2018). Using common sense inferences in Machine Comprehension is a far from solved problem. There have been several attempts in literature to use inferences to answer questions. Most of the previous works either attempt to infer the answer from the given text (Sachan and Xing, 2016; Sun et al., 2018) or an external commonsense knowledge base (Das et al., 2017; Mihaylov and Frank, 2018; Bauer et al., 2018; Weissenborn et al., 2017).

While neural models can capture some dependencies between choices through shared representations, to the best of our knowledge, inferences capturing the dependencies between answer choices or different questions have been not explicitly modeled.

3 Model

Formally, the task of machine comprehension can be defined as: given text $\mathcal{P}$ and a set of n related questions $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ each having m choices $\mathcal{C} = \{c_1^i, c_2^i, \ldots, c_m^i\} \forall q_i \in \mathcal{Q}$, the task is to assign true/false value for each choice c_j^i.

3.1 Model Architecture

Our model consists of three separate systems, one for each step, namely, the stand-alone question answering (QA) system, the Natural Language Inference (NLI) system and the inference framework connecting the two. First, we assign a true/false label to each question-choice pair using the stand-alone QA system along with an associated confidence score s_1. Consequently, we identify the natural language relation (entailment, contradiction or neutral) between each ordered pair of choices for a given question, along with an associated confidence score s_2. Then, we use a relational framework to perform inference using the information obtained from the stand-alone QA and the NLI systems. Each of the components is described in detail in the following sub-sections.

We further propose a joint model whose parameters are trained jointly on both the tasks. The joint model uses the answer choice representation generated by the stand-alone QA system as input to the NLI detection system. The architecture of our joint model is shown in figure 2.

3.1.1 Stand-alone QA system

We use the TriAN-single model proposed by Wang et al. (2018) for SemEval-2018 task-11 as our stand-alone QA system. We use the implementation[1] provided by Wang et al. (2018) for our experiments. The system is a tri-attention model that takes passage-question-choice triplet as input and produces the probability of the choice being true as its output.

3.2 NLI System

Our NLI system is inspired from decomposable-attention model proposed by Parikh et al. (2016). We modified the architecture proposed in Parikh et al. (2016) to accommodate the question-choice pairs as opposed to sentence pairs in the original model. We added an additional sequence-attention layer for the question-choice pairs to allow for the

---

[1] `https://github.com/intfloat/commonsense-rc`

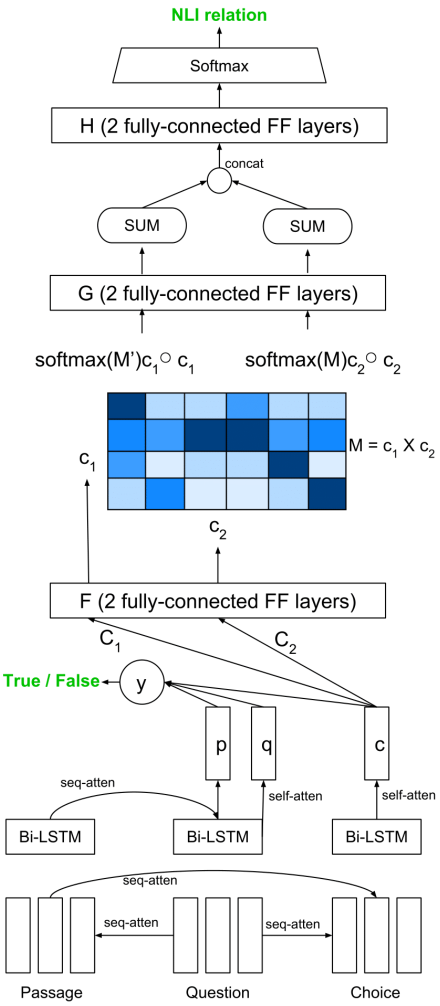


Figure 2: Architecture of the Joint Model

representation of both the answer choice and the question. Sequence-attention is defined in Wang et al. (2018) as:

$$Att_{seq}(\mathbf{u}, \{\mathbf{v}_i\}_{i=1}^{n}) = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \tag{1}$$

$$\alpha_i = softmax_i(f(\mathbf{W}_1\mathbf{u})^T f(\mathbf{W}_1\mathbf{v}_i))$$

where $\mathbf{u}$ and $\mathbf{v}_i$ are word embeddings, $\mathbf{W}_1$ is the associated weight parameter and f is non-linearity. Self-attention is Att_{seq} of a vector onto itself.

The embedding of each word in the answer choice is attended to by the sequence of question word embeddings. We use pre-trained GloVe (Pennington et al., 2014) embeddings to represent the words. The question-attended choices are then passed through the decomposable-attention layer proposed in Parikh et al. (2016).

3.2.1 Inference using DRAIL

We use Deep Relational Learning (DRaiL) framework proposed by Zhang et al. (2016) to perform the final inference. The framework allows for declaration of predicate logic rules to perform relational inference. The rules are scored by the confidence scores obtained from the stand-alone QA and the NLI systems. DRaiL uses an Integer Linear Programming (ILP) based inference procedure to output binary prediction for each of the choices. We use the following constraints for our inference:

1. c_i is true & c_i entails $c_j \implies c_j$ is true.
2. c_i is true & c_i contradicts $c_j \implies c_j$ is false.

On the MultiRC dataset, we use the dependencies between the answer choices for a given question. On SemEval dataset, we use the dependencies between different questions about the same paragraph.

3.3 Joint Model

The design of our joint model is motivated by the two objectives: 1) to obtain a better representation for the question-choice pair for NLI detection and 2) to leverage the benefit of multitask learning. Hence, in the joint model, choice representation from stand-alone QA system is input to the decomposable-attention layer of the NLI system.

The joint model takes two triplets (p, q_i, c_i) and (p, q_j, c_j) as input. It outputs a `true/false` for each choice and an NLI relation (entailment, contradiction or neutral) between the choices. The representations for passage, question and choice are obtained using Bi-LSTMs. The hidden states of the Bi-LSTM are concatenated to generate the representation. This part of the model is similar to TriAN model proposed in Wang et al. (2018). The choice representations of c_i and c_j are passed as input to the decomposable attention layer proposed in Parikh et al. (2016). The architecture of the joint model is shown in figure 2.

3.4 Training

We train the stand-alone QA system using the MultiRC and SemEval datasets for respective experiments. We experiment with 2 different training settings for the NLI system. In the first setting, we use SNLI dataset (Bowman et al., 2015) to train the NLI system. The sequence-attention layer is left untrained during this phase. Hence, we only use the answer choice and do not consider the question for NLI detection.

Self-Training: Subsequently, to help the system adapt to our settings, we devise a self-training protocol over the RC datasets to train the NLI system. Self-training examples for the NLI system were obtained using the following procedure: if the SNLI-trained NLI model predicted entailment and the gold labels of the ordered choice pair were `true-true`, then the choice pair is labeled as *entailment*. Similarly, if the SNLI-trained NLI model predicted contradiction and the gold labels of the ordered choice pair were `true-false`, then the choice pair is labeled as *contradiction*. This is noisy labelling as the labels do not directly indicate the presence of NLI relations between the choices. The NLI model was additionally trained using this data.

Model	**Entailment**	**Contradiction**	**Overall**
NLI_{SNLI}	40.80	74.25	55.11
$NLI_{MultiRC}$	57.30	69.22	66.31

Table 1: Accuracy of entailment and contradiction detection on the development set of self-training data for NLI model trained on SNLI data (NLI_{SNLI}) vs training set of self-training data ($NLI_{MultiRC}$)

To train the joint model we use ordered choice pairs, labeled as *entailment* if the gold labels are `true-true` and labeled as *contradiction* if the gold labels are `true-false`. This data was also used to test the effectiveness of the self-training procedure. The results on the development set of MultiRC dataset are in table 1.

The NLI model trained on SNLI dataset achieves 55.11% accuracy. Training the NLI model on the data from MultiRC data increases the overall accuracy to 66.31%. Further discussion about self-training is provided in section 5.

4 Experiments

We perform experiments in four phases. In the first phase, we evaluate the stand-alone QA system. In the second phase, we train the NLI system on SNLI data and evaluate the approach shown in figure 1. In the third phase, we train the NLI system using the self-training data. In the fourth phase, we evaluate the proposed joint model. We evaluate all models on MultiRC dataset. The results are shown in table 2. We evaluate the joint model on SemEval dataset, shown in table 3.

4.1 Datasets

We use two datasets for our experiments, MultiRC dataset[2] and the SemEval 2018 task 11 dataset[3]. MultiRC dataset consisted of a training and development set with a hidden test set. We split the given training set into training and development sets and use the given development set as test set.

Each question in the MultiRC dataset has approximately 5 choices on average. Multiple of them may be `true` for a given question. The training split of MultiRC consisted of 433 paragraphs and $4,853$ questions with $25,818$ answer choices. The development split has 23 paragraphs and 275 questions with $1,410$ answer choices. Test set has 83 paragraphs and 953 questions with $4,848$ answer choices.

SemEval dataset has 2 choices for each question, exactly one of them is `true`. The training set consists of $1,470$ paragraphs with $9,731$ questions. The development set has 219 paragraphs with $1,411$ questions. And the test set has 430 paragraphs with $2,797$ questions.

4.2 Evaluation Metrics

For MultiRC dataset, we use two metrics for evaluating our approach, namely $EM0$ and $EM1$. $EM0$ refers to the percentage of questions for which all the choices have been correctly classified. $EM1$ is the the percentage of questions for which at most one choice is wrongly classified. For the SemEval dataset, we use *accuracy* metric.

4.3 Results

Results of our experiments are summarized in tables 2 & 3. $EM0$ on MC task improves from 18.15% to 19.41% when we use the NLI model trained over SNLI data and it further improves to 21.62% when we use MultiRC self-training data. Joint model achieves 20.36% on $EM0$ but achieves the highest $EM1$ of 57.08%. Human $EM0$ is 56.56%.

Method	**EM0**	**EM1**
Stand-alone QA	18.15	52.99
QA + NLI_{SNLI}	19.41	56.13
QA + $NLI_{MultiRC}$	21.62	55.72
Joint Model	20.36	57.08
Human	56.56	83.84

Table 2: Summary of the results on MultiRC dataset. $EM0$ is the percentage of questions for which all the choices are correct. $EM1$ is the the percentage of questions for which at most one choice is wrong.

Results of SemEval experiments are summarized in table 3. TriAN-single results are as re-

---

[2] `http://cogcomp.org/multirc/`

[3] `https://competitions.codalab.org/competitions/17184`

ported in (Wang et al., 2018). The results we obtained using their implementation are stand-alone QA results. With the same setting, joint model got 85.4% on dev set and 82.1% on test set. The difference in performance of the models in tables 2 and 3 is statistically significant according to McNemar's chi-squared test.

Model	Dev	Test
TriAN-single (Wang et al., 2018)	83.84%	81.94%
Stand-alone QA	83.20%	80.80%
Joint Model	85.40%	82.10%

Table 3: Accuracy of various models on SemEval'18 task-11 dataset

5 Discussion

We have shown that capturing the relationship between various answer choices or subsequent questions helps in answering questions better. Our experimental results, shown in tables 2 & 3, are only a first step towards leveraging this relationship to help construct better machine reading systems. We suggest two possible extensions to our model, that would help realize the potential of these relations.

1. Improving the performance of entailment and contradiction detection.
2. Using the information given in the text to identify the relations between choices better.

As shown in table 1, identification of entailment/contradiction is far from perfect. Entailment detection is particularly worse because often the system returns *entailment* when there is a high lexical overlap. Moreover, the presence of a strong negation word (*not*) causes the NLI system to predict *contradiction* even for *entailment* and *neutral* cases. This issue impedes the performance of our model on SemEval'18 dataset as roughly 40% of the questions have *yes/no* answers. Naik et al. (2018) show that this is a common issue with state-of-the-art NLI detection models.

Self-training (table 1) results suggest that there are other types of relationships present among answer choice pairs that do not come under the strict definitions of *entailment* or *contradiction*. Upon investigating, we found that although some answer hypotheses do not directly have an inference relation between them, they might be related in context of the given text. For example, consider the sentence, '*I snack when I shop*' and the answer choices: c_1: '*She went shopping this extended weekend*' and c_2: '*She ate a lot of junk food recently*'. Although the sentences don't have an explicit relationship when considered in isolation, the text suggests that c_1 might entail c_2. Capturing these kinds of relationships could potentially improve MC further.

6 Conclusion

In this paper we take a first step towards modeling an accumulative knowledge state for machine comprehension, ensuring consistency between the model's answers. We show that by adapting NLI to the MC task using self-training, performance over multiple tasks improves.

In the future, we intend to generalize our model to other relationships beyond strict entailment and contradiction relations.

Acknowledgements

We would like to thank the reviewers for their insightful comments. This work was partially supported by the NSF through grant NSF-1814105.

References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. *CoRR*, abs/1704.08384.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1–10.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. *CoRR*, abs/1805.08092.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Semeval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Matthew Richardson. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 486–492.

Yawei Sun, Gong Cheng, and Yuzhong Qu. 2018. Reading comprehension with graph-based temporal-casual reasoning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 806–817. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016a. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016b. A parallel-hierarchical model for machine comprehension on sparse data. *CoRR*, abs/1603.08884.

Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Liu Jingming. 2018. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. *CoRR*, abs/1803.00191.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017. Reading twice for natural language understanding. *CoRR*, abs/1706.02596.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Mark Yatskar. 2018. A qualitative comparison of coqa, squad 2.0 and quac. *arXiv preprint arXiv:1809.10735*.

Xiao Zhang, Maria Leonor Pacheco, Chang Li, and Dan Goldwasser. 2016. Introducing DRAIL - a step towards declarative deep relational learning. In *Proceedings of the Workshop on Structured Prediction for NLP@EMNLP 2016, Austin, TX, USA, November 5, 2016*, pages 54–62.

Saliency Learning: Teaching the Model Where to Pay Attention

Reza Ghaeini, Xiaoli Z. Fern, Hamed Shahbazi, Prasad Tadepalli
School of Electrical Engineering and Computer Science, Oregon State University
1148 Kelley Engineering Center, Corvallis, OR 97331-5501, USA
`{ghaeinim, xfern, shahbazh, tadepall}@eecs.oregonstate.edu`

Abstract

Deep learning has emerged as a compelling solution to many NLP tasks with remarkable performances. However, due to their opacity, such models are hard to interpret and trust. Recent work on explaining deep models has introduced approaches to provide insights toward the model's behaviour and predictions, which are helpful for assessing the reliability of the model's predictions. However, such methods do not improve the model's reliability. In this paper, we aim to teach the model to make the right prediction for the right reason by providing explanation training and ensuring the alignment of the model's explanation with the ground truth explanation. Our experimental results on multiple tasks and datasets demonstrate the effectiveness of the proposed method, which produces more reliable predictions while delivering better results compared to traditionally trained models.

1 Introduction

It is unfortunate that our data is often plagued by meaningless or even harmful statistical biases. When we train a model on such data, it is possible that the classifier focuses on irrelevant biases to achieve high performance on the biased data. Recent studies demonstrate that deep learning models noticeably suffer from this issue (Agrawal et al., 2016; Wadhwa et al., 2018; Gururangan et al., 2018). Due to the black-box nature of deep models and the high dimensionality of their inherent representations, it is difficult to interpret and trust their behaviour and predictions. Recent work on explanation and interpretation has introduced a few approaches (Simonyan et al., 2013; Ribeiro et al., 2016; Lei et al., 2016; Li et al., 2016, 2017; Ghaeini et al., 2018b; Ribeiro et al., 2018) for explanation. Such methods provide insights toward the model's behaviour, which is helpful for detecting biases in our models. However, they do not correct them. Here, we investigate how to incorporate explanations into the learning process to ensure that our model not only makes correct predictions but also makes them for the right reason.

Specifically, we propose to train a deep model using both ground truth labels and additional annotations suggesting the desired explanation. The learning is achieved via a novel method called *saliency learning*, which regulates the model's behaviour using saliency to ensure that the most critical factors impacting the model's prediction are aligned with the desired explanation.

Our work is closely related to Ross et al. (2017), which also uses the gradient/saliency information to regularize model's behaviour. However, we differ in the following points: 1) Ross et al. (2017) is limited to regularizing model with gradient of the model's input. In contrast, we extend this concept to the intermediate layers of deep models, which is demonstrated to be beneficial based on the experimental results; 2) Ross et al. (2017) considers annotation at the dimension level, which is not appropriate for NLP tasks since the individual dimensions of the word embeddings are not interpretable; 3) most importantly, Ross et al. (2017) learns from annotations of *irrelevant* parts of the data, whereas we focus on positive annotations identifying parts of the data that contributes positive evidence toward a specific class. In textual data, it is often unrealistic to annotate a word (even a stop word) to be completely irrelevant. On the other hand, it can be reasonably easy to identify group of words that are positively linked to a class.

We make the following contributions: 1) we propose a new method for teaching the model where to pay attention; 2) we evaluate our method on multiple tasks and datasets and demonstrate that our method achieves more reliable predictions while delivering better results than traditionally trained models; 3) we verify the sensitivity of our

Proceedings of NAACL-HLT 2019, pages 4016–4025
Minneapolis, Minnesota, June 2 - June 7, 2019.

saliency-trained model to perturbations introduced on part of the data that contributes to the explanation.

2 Saliency-based Explanation Learning

Our goal is to teach the model where to pay attention in order to avoid focusing on meaningless statistical biases in the data. In this work, we focus on positive explanations. In other words, we expect the explanation to highlight information that contributes positively towards the label. For example, if a piece of text contains the mention of a particular event, then the explanation will highlight parts of the text indicating the event, not non-existence of some other events. This choice is because positive evidence is more natural for human to specify.

Formally, each training example is a tuple (X, y, Z), where $X = [X_1, X_2, \dots, X_n]$ is the input text (length n), y is the ground-truth label, and $Z \in \{0, 1\}^n$ is the ground-truth explanation as a binary mask indicating whether each word contributes positive evidence toward the label y.

Recent studies have shown that the model's predictions can be explained by examining the saliency of the inputs (Simonyan et al., 2013; Hechtlinger, 2016; Ross et al., 2017; Li et al., 2016) as well as other internal elements of the model (Ghaeini et al., 2018b). Given an example, for which the model makes a prediction, the saliency of a particular element is computed as the derivative of the model's prediction with respect to that element. Saliency provides clues as to where the model is drawing strong evidence to support its prediction. As such, if we constrain the saliency to be aligned with the desired explanation during learning, our model will be coerced to pay attention to the right evidence.

In computing saliency, we are dealing with high-dimensional data. For example, each word is represented by an embedding of d dimensions. To aggregate the contribution of all dimensions, we consider sum of the gradients of all dimensions as the overall vector/embedding contribution. For the i-th word, if $Z[i] = 1$, then its vector should have a positive gradient/contribution, otherwise the model would be penalized. To accomplish this, we incorporate a saliency regularization term to the model cost function using hinge loss. Equation 1 describes our cost function evaluated on a single example (X, y, Z).

$$\begin{aligned}\mathcal{C}(\theta, X, y, Z) &= \mathcal{L}(\theta, X, y) \\ &+ \lambda \sum_{i=1}^{n} \max\left(0, -Z_i \sum_{j=1}^{d} \frac{\partial f_\theta(X, y)}{\partial X_{i,j}}\right)\end{aligned} \tag{1}$$

where $\mathcal{L}$ is a traditional model cost function (e.g. cross-entropy), λ is a hyper parameter, f specifies the model with parameter θ, and $\frac{\partial f}{\partial X_{i,j}}$ represents the saliency of the j-th dimension of word X_i. The new term in the $\mathcal{C}$ penalizes negative gradient for the marked words in Z (contributory words).

Since $\mathcal{C}$ is differentiable respect to θ, it can be optimized using existing gradient-based optimization methods. It is important to note that while Equation 1 only regularizes the saliency of the input layer, the same principle can be applied to the intermediate layers of the model (Ghaeini et al., 2018b) by considering the intermediate layer as the input for the later layers.

Note that if $Z = 0$ then $\mathcal{C} = \mathcal{L}$. So, in case of lacking proper annotations for a specific sample or sequence, we can simply use 0 as its annotation. This property enables our method to be easily used in semi-supervised or active learning settings.

3 Tasks and Datasets

To teach the model where to pay attention, we need ground-truth explanation annotation Z, which is difficult to come by. As a proof of concept, we modify two well known real tasks (Event Extraction and Cloze-Style Question Answering) to simulate approximate annotations for explanation. Details of the main tasks and datasets could be found in section B of the Appendix. We describe the modified tasks as follows:

1) **Event Extraction:** Given a sentence, the goal is to determine whether the sentence contains an event. Note that event extraction benchmarks contain the annotation of event triggers, which we use to build the annotation Z. In particular, the Z value of every word is annotated to be zero unless it belongs to an event trigger. For this task, we consider two well known event extraction datasets, namely ACE 2005 and Rich ERE 2015.

2) **Cloze-Style Question Answering:** Given a sentence and a query with a blank, the goal is to determine whether the sentence contains the correct replacement for the blank. Here, annotation of each word is zero unless it belongs to the gold

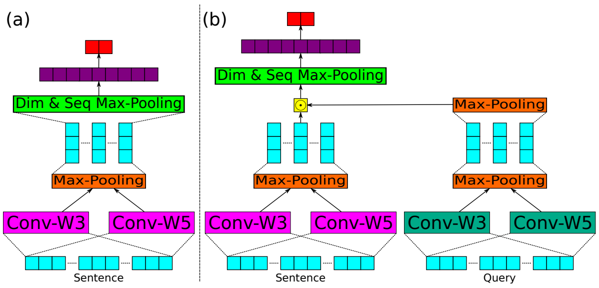


Figure 1: A high-level view of the models used for event extraction (a) and question answering (b).

Dataset	S.^{a}	P.^{b}	R.^{c}	F1	Acc.^{d}
ACE	No	66.0	**77.5**	71.3	74.4
	Yes	**70.1**	76.1	**73.0**	**76.9**
ERE	No	85.0	86.6	85.8	83.1
	Yes	**85.8**	**87.3**	**86.6**	**84.0**
CBT-NE	No	55.6	**76.3**	64.3	75.5
	Yes	**57.2**	74.5	**64.7**	**76.5**
CBT-CN	No	47.4	**39.0**	42.8	77.3
	Yes	**48.3**	38.9	**43.1**	**77.7**

^{a}Saliency Learning. ^{b}Precision.
^{c}Recall. ^{d}Accuracy

Table 1: Performance of trained models on multiple datasets using traditional method and saliency learning.

replacement. For this task, we use two well known cloze-style question answering datasets: Children Book Test Named Entity (CBT-NE) and Common Noun (CBT-CN) (Hill et al., 2015).

Here, we only consider the simple binary tasks as a first attempt to examine the effectiveness of our method. However, our method is not restricted to binary tasks. In multi-class problems, each class can be treated as the positive class of the binary classification. In such a setting, each class would have its own explanation and annotation Z.

Note that for both tasks if an example is negative, its explanation annotation will be all zero. In other words, for negative examples we have $\mathcal{C} = \mathcal{L}$.

4 Model

We use simple CNN based models to avoid complexity. Figure 1 illustrates the models used in this paper. Both models have a similar structure. The main difference is that QA has two inputs (sentence and query). We first describe the event extraction model followed by the QA model.

Figure 1 (a) shows the event extraction model. Given a sentence $W = [w_1, \ldots, w_n]$ where $w_i \in \mathbb{R}^d$, we first pass the embeddings to two CNNs with feature size of d and window size of 3 and 5. Next we apply max-pooling to both CNN outputs. It will give us the representation $I \in \mathbb{R}^{n \times d}$, which we refer to as the *intermediate representation*. Then, we apply sequence-wise and dimension-wise max-poolings to I to capture $D_{seq} \in \mathbb{R}^d$ and $D_{dim} \in \mathbb{R}^n$ respectively. D_{dim} will be referred as *decision representation*. Finally we pass the concatenation of D_{seq} and D_{dim} to a feed-forward layer for prediction.

Figure 1 (b) depicts the QA model. The main difference is having *query* as an extra input. To process the query, we use a similar structure to the main model. After CNNs and max-pooling we end up with $Q \in \mathbb{R}^{m \times d}$ where m is the length of query. To obtain a sequence independent vector, we apply another max-pooling to Q resulting in a query representation $q \in \mathbb{R}^d$. We follow a similar approach to in event extraction for the given sentence. The only difference is that we apply a dot product between the *intermediate representations* and query representation ($I_i = I_i \odot q$).

As mentioned previously, we can apply saliency regularization to different levels of the model. In this paper, we apply saliency regularization on the following three levels: 1) Word embeddings (W). 2) Intermediate representation (I). 3) Decision representation (D_{dim}). Note that the aforementioned levels share the same annotation for training. For training details please refer to Section C of the Appendix.

5 Experiments and Analysis

5.1 Performance

Table 1 shows the performance of the trained models on ACE, ERE, CBT-NE, and CBT-CN datasets using the aforementioned models with and without saliency learning. The results indicate that using saliency learning yields better accuracy and F1 measure on all four datasets. It is interesting to note that saliency learning consistently helps the models to achieve noticeably higher precision without hurting the F1 measure and accuracy. This observation suggests that saliency learning is effective in providing proper guidance for more accurate predictions – Note that here we only have guidance for positive prediction. To verify the statistical significance of the observed performance improvement over traditionally trained

Dataset	S.	W.^{a}	I.^{b}	D.^{c}
ACE	No	61.60	66.05	63.27
	Yes	**99.26**	**77.92**	**65.49**
ERE	No	51.62	56.71	44.37
	Yes	**99.77**	**77.45**	**51.78**
CBT-NE	No	52.32	65.38	68.81
	Yes	**98.17**	**98.34**	**95.56**
CBT-CN	No	47.78	53.68	45.15
	Yes	**99.13**	**98.94**	**97.06**

[a] Word Level Saliency Accuracy.
[b] Intermediate Level Saliency Accuracy.
[c] Decision Level Saliency Accuracy.

Table 2: Saliency accuracy of different layer of our models trained on ACE, ERE, CBT-NE, CBT-CN.

models without saliency learning, we conducted the one-sided McNemar's test. The obtained p-values are 0.03, 0.03, 0.0001, and 0.04 for ACE, ERE, CBT-NE, and CBT-CN respectively, indicating that the performance gain by saliency learning is statistically significant.

5.2 Saliency Accuracy

In this section, we examine how well does the saliency of the trained model aligns with the annotation. To this end, we define a metric called *saliency accuracy* (s_{acc}), which measures what percentage of all positive positions of annotation Z indeed obtain a positive gradient. Formally, $s_{acc} = 100\frac{\sum_i \delta(Z_i G_i > 0)}{\sum_i Z_i}$ where G_i is the gradient of element i and δ is the indicator function.

Table 2 shows the saliency accuracy at different layers of the trained model with and without saliency learning. According to Table 2, our method achieves a much higher saliency accuracy for all datasets indicating that the learning was indeed effective in aligning the model saliency with the annotation. In other words, important words will have positive contributions in the saliency-trained model, and as such, it learns to focus on the right part(s) of the data. This claim can also be verified by visualizing the saliency, which is provided in the next section.

5.3 Saliency Visualization

Here, we visualize the saliency of three positive samples from the ACE dataset for both the traditionally trained (Baseline Model) and the saliency-trained model (saliency-trained Model). Table 3 shows the top 6 salient words (words with highest saliency/gradient) of three positive samples along with their contributory words (annotation Z), the baseline model prediction (P_B), and the saliency-trained model prediction (P_S). Darker red color indicates more salient words. According to Table 3, both models correctly predict 1 and the saliency-trained model successfully pays attention to the expected meaningful words while the baseline model pays attention to mostly irrelevant ones. More analyses are provided in section D of the Appendix.

5.4 Verification

Up to this point, we show that using saliency learning yields noticeably better precision, F1 measure, accuracy, and saliency accuracy. Here, we aim to verify our claim that saliency learning coerces the model to pay more attention to the critical parts. The annotation Z describes the influential words toward the positive labels. Our hypothesis is that *removing such words would cause more impact on the saliency-trained models* since by training, they should be more sensitive to these words. We measure the impact as the percentage change of the model's true positive rate. This measure is chosen because negative examples do not have any annotated contributory words, and hence we are particularly interested in how removing contributory words of positive examples would impact the model's true positive rate (TPR).

Table 4 shows the outcome of the aforementioned experiment, where the last column lists the TPR reduction rates. From the table, we see a consistently higher rate of TPR reduction for saliency-trained models compared to traditionally trained models, suggesting that the saliency-trained models are more sensitive to the perturbation of the contributory word(s) and confirming our hypothesis.

It is worth noting that we observe less substantial change to the true positive rate for the event task. This is likely due to the fact that we are using trigger words as simulated explanations. While trigger words are clearly related to events, there are often other words in the sentence relating to events but not annotated as trigger words.

6 Conclusion

In this paper, we proposed *saliency learning*, a novel approach for teaching a model where to pay attention. We demonstrated the effectiveness

id	Baseline Model	Saliency-trained Model	Z	P_B	P_S
1	The judge at Hassan's extradition hearing said that he found the French handwriting report very problematic, very confusing, and with suspect conclusions.	The judge at Hassan 's extradition hearing said that he found the French handwriting report very problematic, very confusing, and with suspect conclusions.	extradition hearing said	1	1
2	Solana said the EU would help in the humanitarian crisis expected to follow an attack on Iraq.	Solana said the EU would help in the humanitarian crisis expected to follow an attack on Iraq .	attack	1	1
3	The trial will start on March 13 , the court said .	The trial will start on March 13 , the court said.	trial	1	1

Table 3: Top 6 salient words visualization of data samples from ACE for the baseline and the saliency-trained models.

Dataset	S.	TPR$^a_0$	TPR$^b_1$	ΔTPR^c
ACE	No	77.5	52.2	32.6
	Yes	76.1	45.0	**40.9**
ERE	No	86.6	73.2	15.4
	Yes	87.3	70.6	**19.1**
CBT-NE	No	76.3	30.2	60.4
	Yes	74.5	28.5	**61.8**
CBT-CN	No	39.0	16.6	57.4
	Yes	38.9	15.4	**60.4**

[a] True Positive Rate (before removal).
[b] TPR after removing the critical word(s).
[c] TPR change rate.

Table 4: True positive rate and true positive rate change of the trained models before and after removing the contributory word(s).

of our method on multiple tasks and datasets using simulated explanations. The results show that saliency learning enables us to obtain better precision, F1 measure and accuracy on these tasks and datasets. Further, it produces models whose saliency is more properly aligned with the desired explanation. In other words, *saliency learning* gives us more reliable predictions while delivering better performance than traditionally trained models. Finally, our verification experiments illustrate that the saliency-trained models show higher sensitivity to the removal of contributory words in a positive example. For future work, we will extend our study to examine saliency learning on NLP tasks in an active learning setting where real explanations are requested and provided by a human.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract N66001-17-2-4030.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1955–1960.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 167–176.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 593–602.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846.

Reza Ghaeini, Xiaoli Z. Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Reza Ghaeini, Xiaoli Z. Fern, Hamed Shahbazi, and Prasad Tadepalli. 2018a. Dependent gated reading for cloze-style question answering. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3330–3345.

Reza Ghaeini, Xiaoli Z. Fern, and Prasad Tadepalli. 2018b. Interpreting recurrent and attention-based neural models: a case study on natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4952–4957.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 107–112.

Yotam Hechtlinger. 2016. Interpretation of prediction models using the input gradient. *CoRR*, abs/1611.07634.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 107–117.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 681–691.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220.

John Walker Orr, Prasad Tadepalli, and Xiaoli Z. Fern. 2018. Event detection with neural networks: A rigorous empirical evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 999–1004.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1527–1535.

Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2662–2670.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 128–137.

Soumya Wadhwa, Varsha Embar, Matthias Grabmair, and Eric Nyberg. 2018. Towards inference-oriented reading comprehension: Parallelqa. *CoRR*, abs/1805.03830.

A Background: Saliency

The concept of saliency was first introduced in vision for visualizing the spatial support on an image for particular object class (Simonyan et al., 2013). Considering a deep model prediction as a differentiable model f parameterized by θ with input $X \in \mathbb{R}^{n \times d}$. Such a model could be described using the Taylor series as follow:

$$f(x) = f(a)+f^{'}(a)(x-a)+\frac{f^{''}(a)}{2!}(x-a)^2+\dots \tag{2}$$

By approximating that a deep model is a linear function, we could use the first order Taylor expansion.

$$f(x) \approx f^{'}(a)x + b \tag{3}$$

According to Equation 3, the first derivative of the model's prediction respect to its input ($f^{'}(a)$ or $\frac{\partial f}{\partial x}|_{x=a}$) describes the model's behaviour near the input. To make it more clear, bigger derivative/gradient indicates more impact and contribution toward the model's prediction. Consequently, the large-magnitude derivative values determine elements of input that would greatly affect $f(x)$ if changed.

B Task and Dataset

Here, we first describe the main and real Event Extraction and Close-Style Question Answering tasks (before our modification). Next, we provide data statistics of the modified version of ACE, ERE, CBT-NE, and CBT-CN datasets in Table 5.

- **Event Extraction:** Given a set of ontologized event types (e.g. Movement, Transaction, Conflict, etc.), the goal of event extraction is to identify the mentions of different events along with their types from natural texts (Chen et al., 2015; Ghaeini et al., 2016; Orr et al., 2018).

- **Cloze-Style Question Answering:** Documents in CBT consist of 20 contiguous sentences from the body of a popular children book and queries are formed by replacing a token from the 21^{st} sentence with a blank. Given a document, a query, and a set of candidates, the goal is to find the correct replacement for blank in the query among the given candidates. To avoid having too many negative examples in our modified datasets, we only consider sentences that contain at least one candidate. To be more clear, each sample from the CBT dataset is split to at most 20 samples – each sentence of the main sample as long as it contains one of the candidates (Trischler et al., 2016; Kadlec et al., 2016; Cui et al., 2017; Dhingra et al., 2017; Ghaeini et al., 2018a).

Dataset	Sample Count			
	Train		Test	
	P.[a]	N.[b]	P.	N.
ACE	3.2K	15K	293	421
ERE	3.1K	4K	2.7K	1.91K
CBT-NE	359K	1.82M	8.8K	41.1K
CBT-CN	256K	2.16M	5.5K	44.4K

[a] Positive Sample Count

[b] Negative Sample Count

Table 5: Dataset statistics of the modified tasks and datasets.

C Training

All hyper-parameters are tuned based on the development set. We use pre-trained $300-D$ Glove $840B$ vectors (Pennington et al., 2014) to initialize our word embedding vectors. All hidden states and feature sizes are 300 dimensions ($d = 300$). The weights are learned by minimizing the cost function on the training data via Adam optimizer. The initial learning rate is 0.0001 and $\lambda = 0.5, 0.7, 0.4$, and 0.35 for ACE, ERE, CBT-NE, and CBT-CN respectively. To avoid overfitting, we use dropout with a rate of 0.5 for regularization, which is applied to all feedforward connections. During training, the word embeddings are updated to learn effective representations for each task and dataset. We use a fairly small batch size of 32 to provide more exploration power to the model. Finally, Equation 4 indicates the the cost function that is used for the training where W, I, and D_{dim} are the *word embeddings*, *Intermediate representation*, and *Decision representa-*

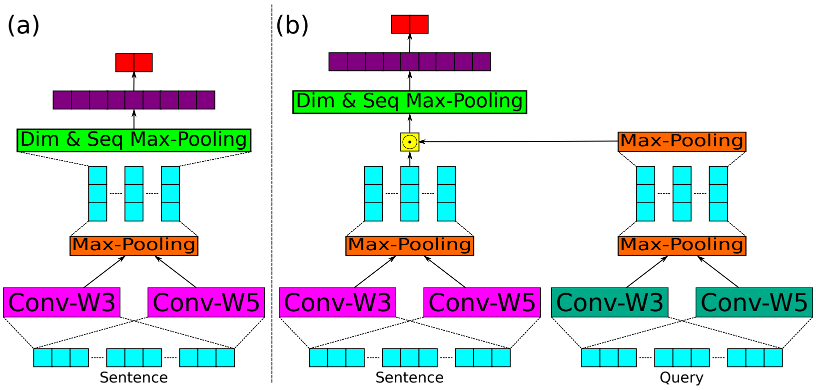


Figure 2: A high-level view of the models used for event extraction(a) and question answering (b).

tion respectively.

$$\begin{aligned} \mathcal{C}(\theta, X, y, Z) &= \mathcal{L}(\theta, X, y) \\ &+ \lambda \sum_{i=1}^{n} \max\left(0, -Z_i \sum_{j=1}^{d} \frac{\partial f_W(W, y)}{\partial W_{i,j}}\right) \\ &+ \lambda \sum_{i=1}^{n} \max\left(0, -Z_i \sum_{j=1}^{d} \frac{\partial f_I(I, y)}{\partial I_{i,j}}\right) \\ &+ \lambda \sum_{i=1}^{n} \max\left(0, -Z_i \frac{\partial f_{D_{dim}}(D_{dim}, y)}{\partial D_{dim,i}}\right) \end{aligned} \tag{4}$$

D Saliency Visualization

In this section, we empirically analyze the traditionally trained (Baseline Model) and the saliency-trained model (saliency-trained Model) behaviour by observing the saliency of 23 positive samples from ACE and ERE datasets. Tables 6 and 7 show the top 6 salient words (words with highest saliency/gradient) of positive samples from ACE or ERE dataset along with their contributory word(s) (Z), the baseline model prediction (P_B), and the saliency-trained model prediction (P_S). Darker red color indicates more salient words. Our observations could be divided into six categories as follow:

- Samples 1-7: Both models correctly predict 1 for these samples. The saliency-trained model successfully pays attention to the expected meaningful words while the baseline model pays attention to mostly irrelevant ones.

- Samples 8-11: Both models correctly predict 1 and pays attention to the contributory words. Yet, we observe lower saliency for important words and higher saliency for irrelevant ones.

- Samples 12-14: Here, the baseline model fails to pay attention to the contributory words and predicts 0 while the saliency-trained model one successfully pays attention to them and predicts 1.

- Samples 15-18: Although the models have high saliency for the contributory words, still they could not correctly disambiguate these samples. This observation suggests that having high saliency for important words does not guarantee positive prediction. High saliency for these words indicate their positive contribution toward the positive prediction but still, the model might consider higher probability for negative prediction.

- Samples 19-21: Here, only the baseline model could correctly predict 1. However, the baseline model does not pay attention to the contributory words. In other words, the explanation does not support the prediction (unreliable).

- Samples 22-23: Not always the saliency-trained model could pay proper attention to the contributory words. In these examples, the baseline model has high saliency for contributory words. It is worth noting that when the saliency-trained model does not have high saliency for contributory words, it does not predict 1. Such observation could suggest that the saliency-trained model predictions are more reliable. The aforementioned claim is also verified by consistently obtaining noticeably higher precision for all datasets and tasks (Section 5.1 and Table 1 in the paper).

id	Baseline Model	Saliency-trained Model	Z	P_B	P_S
1	The judge at Hassan's extradition hearing said that he found the French handwriting report very problematic, very confusing, and with suspect conclusions.	The judge at Hassan 's extradition hearing said that he found the French handwriting report very problematic, very confusing, and with suspect conclusions.	extradition hearing said	1	1
2	Solana said the EU would help in the humanitarian crisis expected to follow an attack on Iraq.	Solana said the EU would help in the humanitarian crisis expected to follow an attack on Iraq .	attack	1	1
3	The trial will start on March 13 , the court said .	The trial will start on March 13 , the court said.	trial	1	1
4	India 's has been reeling under a heatwave since mid-May which has killed 1,403 people.	India 's has been reeling under a heatwave since mid-May which has killed 1,403 people .	killed	1	1
5	Retired General Electric Co. Chairman Jack Welch is seeking work-related documents of his estranged wife in his high-stakes divorce case .	Retired General Electric Co. Chairman Jack Welch is seeking work-related documents of his estranged wife in his high-stakes divorce case .	Retired divorce	1	1
6	The following year, he was acquitted in the Guatemala case, but the U.S. continued to push for his prosecution.	The following year, he was acquitted in the Guatemala case , but the U.S. continued to push for his prosecution .	acquitted case	1	1
7	In 2011, a Spanish National Court judge issued arrest warrants for 20 men , including Montano,suspected of participating in the slaying of the priests.	In 2011, a Spanish National Court judge issued arrest warrants for 20 men, including Montano,suspected of participating in the slaying of the priests.	issued slaying arrest	1	1
8	Slobodan Milosevic's wife will go on trial next week on charges of mismanaging state property during the former president's rule, a court said Thursday.	Slobodan Milosevic's wife will go on trial next week on charges of mismanaging state property during the former president 's rule, a court said Thursday .	trial charges former	1	1
9	Iraqis mostly fought back with small arms, pistols, machine guns and rocket-propelled grenades .	Iraqis mostly fought back with small arms, pistols, machine guns and rocket-propelled grenades.	fought	1	1
10	But the Saint Petersburg summit ended without any formal declaration on Iraq .	But the Saint Petersburg summit ended without any formal declaration on Iraq .	summit	1	1

Table 6: Top 6 salient words visualization of samples from ACE and ERE for the baseline and the saliency-trained models.

id	Baseline Model	Saliency-trained Model	Z	P_B	P_S
11	He will then stay on for a regional summit before heading to Saint Petersburg for celebrations marking the 300th anniversary of the city's founding.	He will then stay on for a regional summit before heading to Saint Petersburg for celebrations marking the 300th anniversary of the city's founding.	heading summit	1	1
12	From greatest moment of his life to divorce in 3 years or less.	From greatest moment of his life to divorce in 3 years or less.	divorce	0	1
13	The state s execution record has often been criticized.	The state s execution record has often been criticized.	execution	0	1
14	The student, who was 18 at the time of the alleged sexual relationship, testified under a pseudonym.	The student, who was 18 at the time of the alleged sexual relationship, testified under a pseudonym.	testified	0	1
15	U.S. aircraft bombed Iraqi tanks holding bridges close to the city.	U.S. aircraft bombed Iraqi tanks holding bridges close to the city.	bombed	0	0
16	However, no blasphemy convict has ever been executed in the country.	However, no blasphemy convict has ever been executed in the country.	executed	0	0
17	Gul 's resignation had been long expected.	Gul 's resignation had been long expected.	resignation	0	0
18	aside from purchasing alcohol, what rights don't 18 year olds have?	aside from purchasing alcohol, what rights don't 18 year olds have?	purchasing	0	0
19	He also ordered him to have no contact with Shannon Molden.	He also ordered him to have no contact with Shannon Molden.	ordered contact	1	0
20	This means your account is once again active and operational, Riao wrote Colombia Reports.	This means your account is once again active and operational, Riao wrote Colombia Reports.	wrote	1	0
21	I am a Christian as is my ex husband yet we are divorced.	I am a Christian as is my ex husband yet we are divorced.	divorced ex	1	0
22	Taylor acknowledged in his testimony that he ran up toward the pulpit with a large group and followed the men outside.	Taylor acknowledged in his testimony that he ran up toward the pulpit with a large group and followed the men outside.	testimony followed ran	1	0
23	The note admonished Jasper Molden, and his then-fiance, Shannon Molden.	The note admonished Jasper Molden, and his then-fiance, Shannon Molden.	note	0	0

Table 7: Top 6 salient words visualization of samples from ACE and ERE for the baseline and the saliency-trained models.

Understanding Dataset Design Choices for Multi-hop Reasoning

Jifan Chen and **Greg Durrett**
The University of Texas at Austin
`{jfchen,gdurrett}@cs.utexas.edu`

Abstract

Learning multi-hop reasoning has been a key challenge for reading comprehension models, leading to the design of datasets that explicitly focus on it. Ideally, a model should not be able to perform well on a multi-hop question answering task without doing multi-hop reasoning. In this paper, we investigate two recently proposed datasets, WikiHop (Welbl et al., 2018) and HotpotQA (Yang et al., 2018). First, we explore sentence-factored models for these tasks; by design, these models cannot do multi-hop reasoning, but they are still able to solve a large number of examples in both datasets. Furthermore, we find spurious correlations in the unmasked version of WikiHop, which make it easy to achieve high performance considering only the questions and answers. Finally, we investigate one key difference between these datasets, namely span-based vs. multiple-choice formulations of the QA task. Multiple-choice versions of both datasets can be easily gamed, and two models we examine only marginally exceed a baseline in this setting. Overall, while these datasets are useful testbeds, high-performing models may not be learning as much multi-hop reasoning as previously thought.

1 Introduction

Question answering from text (Richardson et al., 2013; Hill et al., 2015; Hermann et al., 2015; Rajpurkar et al., 2016) is a key challenge problem for NLP that tests whether models can extract information based on a query. However, even sophisticated models that perform well on QA benchmarks (Seo et al., 2017; Shen et al., 2017; Yu et al., 2018) may only be doing shallow pattern matching of the question against the supporting passage (Weissenborn et al., 2017). More recent work (Kumar et al., 2016; Joshi et al., 2017; Welbl et al., 2018) has emphasized gathering information from different parts of a passage to answer the question, leading to a number of models designed to do *multi-hop reasoning*. Two recent large-scale datasets have been specifically designed to test multi-hop reasoning: WikiHop (Welbl et al., 2018) and HotpotQA (Yang et al., 2018).

In this paper, we seek to answer two main questions. First, although the two datasets are explicitly constructed for multi-hop reasoning, do models really need to do multi-hop reasoning to do well on them? Recent work has shown that large-scale QA datasets often do not exhibit their advertised properties (Chen et al., 2016; Kaushik and Lipton, 2018). We devise a test setting to see whether multi-hop reasoning is necessary: can a model which treats each sentence independently select the sentence containing the answer? This provides a rough estimate of the fraction of questions solvable by a non-multi-hop system. Our results show that more than half of the questions in WikiHop and HotpotQA do not require multi-hop reasoning to solve. Surprisingly, we find that a simple baseline which ignores the passage and only uses the question and answer can achieve strong results on WikiHop and a modified version of HotpotQA, further confirming this view.

Second, we study the nature of the supervision on the two datasets. One critical difference is that HotpotQA is span-based (the answer is a span of the passage) while WikiHop is multiple-choice. How does this difference affect learning and evaluation of multi-hop reasoning systems? We show that a multiple-choice version of HotpotQA is vulnerable to the same baseline that performs well on WikiHop, showing that this distinction may be important from an evaluation standpoint. Furthermore, we show that a state-of-the-art model, BiDAF++, trained on span-based HotpotQA and adapted to the multiple-choice setting outperforms the same model trained natively on

Proceedings of NAACL-HLT 2019, pages 4026–4032
Minneapolis, Minnesota, June 2 - June 7, 2019.

the multiple-choice setting. However, even in the span-based setting, the high performance of the sentence-factored models raises questions about whether multi-hop reasoning is being learned.

Our conclusions are as follows: (1) Many examples in both WikiHop and HotpotQA do not require multi-hop reasoning to solve, as the sentence-factored model can find the answers. (2) On WikiHop and a multiple-choice version of HotpotQA, a no context baseline does very well. (3) Span-based supervision provides a harder testbed than multiple choice by having more answers to choose from, but given the strong performance of the sentence-factored models, it is unclear whether any of the proposed models are doing a good job at multi-hop reasoning in any setting.

2 Datasets

WikiHop Welbl et al. (2018) introduced this English dataset specially designed for text understanding across multiple documents. The dataset consists of 40k+ questions, answers, and passages, where each passage consists of several documents collected from Wikipedia. Questions are posed as a query of a relation r followed by a head entity h, with the task being to find the tail entity t from a set of entity candidates E. Annotators followed links between documents and were required to use multiple documents to get the answer.

HotpotQA Yang et al. (2018) proposed a new dataset with 113k English Wikipedia-based question-answer pairs. The questions are diverse, falling into several categories, but all require finding and reasoning over multiple supporting documents to answer. Models should choose answers by selecting variable-length spans from these documents. Sentences relevant to finding the answer are annotated in the dataset as "supporting facts" so models can use these at training time as well.

3 Probing Multi-hop Datasets

In this section, we seek to answer whether multi-hop reasoning is really needed to solve these two multi-hop datasets.

3.1 Sentence-Factored Model Test

If a question requires a multi-hop model, then we should not be able to figure out the answer by only looking at the question and each sentence separately. Based on this idea, we propose a sentence-factored modeling setting, where a model must predict which sentence contains the answer but must score each sentence independently, i.e., without using information from other sentences in this process. Identifying the presence of the answer is generally easier than predicting the answer directly, particularly if a sentence is complicated, and is still sufficient to provide a bound on how strongly multi-hop reasoning is required. Figure 1 shows a typical example from these datasets, where identifying the answer (*Delhi*) requires bridging to an entity not mentioned in the question.

Method	Random	Factored	Factored BiDAF
WikiHop	6.5	60.9	66.1
HotpotQA	5.4	45.4	57.2
SQuAD	22.1	70.0	88.0

Table 1: The accuracy of our proposed sentence-factored models on identifying answer location in the development sets of WikiHop, HotpotQA and SQuAD. *Random*: we randomly pick a sentence in the passage to see whether it contains the answer. *Factored* and *Factored BiDAF* refer to the models of Section 3.1. As expected, these models perform better on SQuAD than the other two datasets, but the model can nevertheless find many answers in WikiHop especially.

Simple Factored Model We encode each passage sentence s_i and the question q into a contextual representation h_{s_i} and h_q using a bi-directional GRU (Chung et al., 2014). Then, $S_i = h_{s_i}^\top W h_q$; that is, compute a bilinear product of these representations with trainable weights W to get the score of the ith sentence. Finally, let $p_i = \text{softmax}_i(S_i)$; softmax over the sentences to get a probability distribution. We maximize the marginal log probability of picking a sentence containing the correct answer: $\log(\sum_{i:s_i \in \mathbf{s}^*} p_i)$, where $\mathbf{s}^*$ is the set of sentences containing the answer. During evaluation, we pick the sentence s with the highest score and treat it as correct if it contains the answer.

Factored BiDAF We encode the question and each sentence *separately* using bi-GRUs. Then, we generate the question-aware token representation for each token of sentence by using a co-attention layer (Seo et al., 2017). Finally, we max-pool over each sentence to get the sentence representation and feed those to a FFNN to compute the sentence score. Training and inference are the same as for the simple model.

We run this test on both datasets as well as

Question: **The Oberoi family** is part of a hotel company that has a head office in what city?

……

The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through **The Oberoi Group**.

……

The Oberoi Group is a hotel company with its head office in **Delhi**.

Figure 1: An example from the HotpotQA dev set. Here, a model should have to form a reasoning chain *Oberoi family* → *Oberoi Group* → *Delhi* to arrive at the answer. However, the sentence containing *Delhi* has a substantial lexical overlap with the question, so strong QA systems can answer it directly.

SQuAD (Rajpurkar et al., 2016), where multi-hop reasoning is only needed in a few questions. Results in Table 1 indicate that although intentionally created for multi-hop reasoning, for more than half of questions in WikiHop and HotpotQA, we can figure out where the answer is without doing multi-hop reasoning. This result is initially surprising, but one reason it may be possible is suggested by the example from HotpotQA shown in Figure 1. We can see that the model could easily figure out the answer sentence without looking at the bridging entities using lexical cues alone. This observation is also in accordance with the work of Jansen (2018), which demonstrates that high performance for a simple baseline can be achieved in cases when passages have increasing lexical overlap with the question.

We note that this method potentially overestimates performance of a non-multi-hop model on HotpotQA, since there are some examples where many plausible answers are in the same sentence and require other context to resolve. However, these still form a minority in the dataset (see Table 3 of Yang et al. (2018)).

3.2 No Context Baseline

The results of the previous section show that a model can identify correspondences between questions and answer sentences. One other pair of correlations we can study is suggested in the work of Kaushik and Lipton (2018), namely examining question-answer correlations independent of the passage. We construct a “no context” baseline to verify whether it is possible to pick the correct answer without consulting the passage. In a similar fashion to the factored model, we encode the query q and each answer candidate c_i using a bi-GRU and once again compute a bilinear product between them to get the scores over candidates, making no reference to the document.

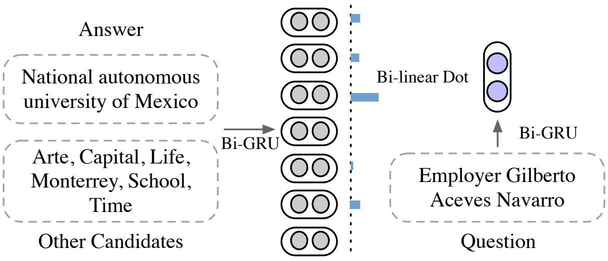


Figure 2: An example of question and candidates from WikiHop. Here we can see that among the candidates, only *National autonomous university of Mexico* is an organization which could be Navarro’s employer; the model may pick up on this entity typing.

NoContext	Coref-GRU	MHQA-GRN	Entity-GCN
59.70	56.00	62.80	64.80

Table 2: The results of our no-context baseline compared with Coref-GRU (Dhingra et al., 2018), MHQA-GRN (Song et al., 2018), and Entity-GCN (De Cao et al., 2018) on the WikiHop dev set.

Results of this model on the multiple-choice WikiHop dataset are shown in Table 2. Surprisingly, the no-context baseline achieves high performance, comparable to some recently-published systems, showing that WikiHop is actually possible to solve reasonably well without using the document at all. One possible reason for this is that this model can filter possible answers based on expected answer type (Sugawara et al., 2018), as shown in the example of Figure 2, or perhaps capture other correlations between training and test. This model substantially outperforms the unlearned baseline reported in the WikiHop paper (Welbl et al., 2018) (38.8%) as well as the BiDAF (Seo et al., 2017) results reported there (42.9%).

4 Span-based vs. Multiple-choice

The no context model indicates that having multiple-choice questions may provide an avenue for a dataset to be gamed. In order to investigate the difference in multiple-choice vs. span supervision while controlling for other aspects of dataset difficulty, we first recast each dataset in the other’s framework, then investigate the performance of two models each of these settings.

To modify Hotpot to be multiple-choice, we

Dataset	HotpotQA-MC	WikiHop-MC
Metric	Accuracy	Accuracy
NoContext	68.01	59.70
MC-BiDAF++	70.01	61.32
MC-MemNet	68.75	61.80
Span2MC-BiDAF++	76.01	59.85

Table 3: The performance of different models on the dev sets of WikiHop and HotpotQA. MC denotes using both the multiple-choice dataset and model. Span2MC means we train the model with span-based supervision and evaluate the model on a multiple choice setting. Our models only mildly outperform the no-context baseline in all settings.

randomly select 9 entities in all of the documents as distractors, and add the answer to make a 10-choice candidates set. To modify WikiHop to be span-based, we concatenate all documents and treat the first appearance of the answer mention as the gold span for training. Any answer occurrence is treated as correct for evaluation.

4.1 Systems to Compare

MemNet Memory networks (Weston et al., 2015) define a generic model class which can gather information from different parts of the passage. Kumar et al. (2016) and Miller et al. (2016) have demonstrated its effectiveness in certain multi-hop settings. These models process a document over several timesteps. On the ith step, the model takes a question representation q_i, attends to the context representation $\mathbf{p}$, gets an attention distribution α_i, computes a new memory cell value $m_i = \sum \alpha_i p_i$, then forms an updated $q_{i+1} = f(m_i, q_i)$. The final memory cell m_T is used to compute a score $s_i = g(m_T, c_j)$ with the jth candidate representation c_j. We modify this architecture slightly using a standard hierarchical attention module (Li et al., 2015).

We can also modify this architecture to predict an answer span – we use the memory cell m_T of the last step, and do a bi-linear product with the context representation $\mathbf{p}$ to compute a distribution over start points $P_{start} = \text{softmax}(\mathbf{p}W_{start}m_T)$ and end points distribution $P_{end} = \text{softmax}(\mathbf{p}W_{end}m_T)$ of the answer span, where W_{start} and W_{end} are two parameter matrix to be learned. We call this Span-MemNet.

BiDAF++ Recently proposed by Clark and Gardner (2018), this is a high-performing model on SQuAD. It combines the bi-directional attention flow (Seo et al., 2017) and self-attention mechanisms. We use the implementation described in Yang et al. (2018).

Dataset	HotpotQA-Span		WikiHop-Span	
Metric	EM	F1	EM	F1
BiDAF++ (Yang+ 18)	42.79	56.19	–	–
Span-BiDAF++	42.45	56.46	24.23	46.13
Span-MemNet	18.75	26.11	13.54	19.23

Table 4: The performance of different models on the dev sets of WikiHop and HotpotQA. Span denotes using both span-based dataset and model. BiDAF++ denotes the performance reported in HotpotQA (Yang et al., 2018).

We can modify this model for the multiple-choice setting as well. Specifically, we use the start P_{start} and end P_{end} distribution to do a weighted sum over the context $\mathbf{p}$ to get a summarized representation $D_{start} = \sum P_{start_i} p_i$, $D_{end} = \sum P_{end_i} p_i$ of the context. Then we concatenate them to do a bilinear dot product with each candidate representation to get the answer score as we described for MemNet. We call this model MC-BiDAF++.

4.2 Results

Table 3 and Table 4 show our results in the two settings. As a baseline on multiple-choice HotpotQA, we also test the no-context baseline, which achieves an accuracy of 68.01%, around 10% absolute higher than on WikiHop. Our candidates were randomly chosen, so this setting may not be quite as challenging as a well-constructed multiple-choice dataset. From Table 3 and Table 4 we draw the following conclusions.

When trained and tested on multiple-choice datasets, our models do not learn multi-hop reasoning. Comparing MC-BiDAF++ and MC-MemNet on the multiple-choice setting of both datasets as shown in Table 3, the models appear to have similar capabilities to learn multi-hop reasoning. However, looking at the no-context baseline for comparison, we find that it is only around 2% lower than the two relatively more complex models. This indicates that much of the performance is achieved by "cheating" through the correlation between the candidates and question/context. Surprisingly, this is true even for HotpotQA, which seems stronger based on the analysis in Table 1.

Span-based data is less "hackable", but models still may not be doing multi-hop reasoning. We then compare the results of Span-BiDAF++

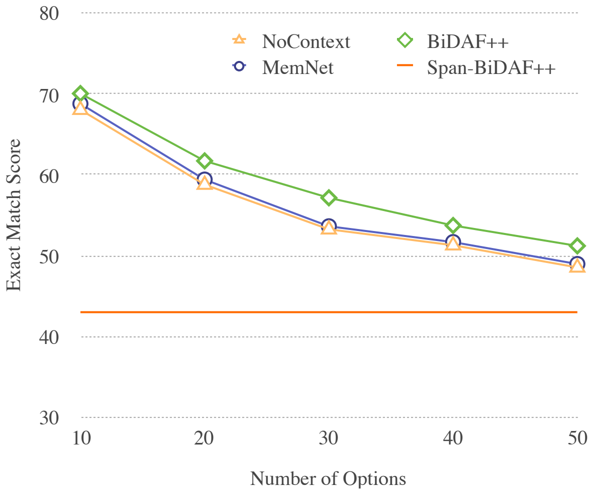


Figure 3: Performance of different options on HotpotQA-MC. Adding more options does not strengthen the model's ability of learning multi-hop reasoning.

and Span-MemNet on the span-based settings of both datasets, which are substantially different from the multiple-choice setting as shown in Table 4. BiDAF++ substantially outperforms the MemNet on both datasets, indicating that BiDAF++ is a stronger model for multi-hop reasoning, despite being less explicitly designed for this task. However, this model still underperforms the Factored BiDAF model, indicating that it could just be doing strong single-sentence reasoning.

Adding more options does not qualitatively change the multiple choice setting. The span-based model requires dealing with a much larger output space than the multiple-choice setting. To test the effects of this, we conduct another experiment by making more spurious options on HotpotQA-MC using the method described in Section 4. The results are shown in Figure 3. As we increase the number of options, we can see that the performance of all models drops. However, even with more options, the no-context baseline can still achieve comparable performance to the other two more complex models, which indicates that these models still aren't learning multi-hop reasoning in such a strengthened setting.

Span-based training data is more powerful. To further understand the two different supervision signals, we conduct another experiment where we train using span-based supervision and evaluate on the multiple-choice setting. Specifically, during evaluation, we select all document spans that map onto some answer candidate, then max over the scores of all spans to pick the predicted answer candidate. The multiple choice options therefore filter the span model's predictions.

From the results in Table 3, we can see that Span2MC-BiDAF++ achieves higher performance compared to MC-BiDAF++ on HotpotQA and nearly comparable performance on WikiHop even with random span selection during training. This shows that with the span-based supervision, the model can learn at least the same thing as the multiple-choice and avoid "cheating" through learning question-candidate correspondences.

5 Discussion and Conclusion

There exist several other multi-hop reasoning datasets including WorldTree (Jansen et al., 2018), OpenBookQA (Mihaylov et al., 2018), and MultiRC (Khashabi et al., 2018). These datasets are more complex to analyze since the answers may not appear directly in the passage and may simply be entailed by passage content. We leave a detailed investigation of these for future work.

For researchers working on the problem of multi-hop reasoning, we think the following points should be considered: (1) Prefer models using span-based supervision to avoid "cheating" by using the extra candidate information. (2) If using multiple-choice supervision, check the no-context baseline to see whether there are strong correlations between question and candidates. (3) When constructing a multi-hop oriented dataset, it would be best to do an adversarial test using a sentence-factored model to see whether multi-hop reasoning is really needed. Both HotpotQA and WikiHop contain good examples for evaluating multi-hop reasoning, but this evaluation is clouded by the presence of easily-solvable examples, which can confuse the learning process as well.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, NSF Grant SHF-1762299, a Bloomberg Data Science Grant, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. Thanks as well to the anonymous reviewers for their helpful comments.

References

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. *ACL*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Deep Learning workshop at NeurIPS 2014*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. *ACL*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *EMNLP*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural Models for Reasoning over Multiple Mentions using Coreference. *NAACL*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv preprint arXiv:1511.02301*.

Peter Jansen. 2018. Multi-hop Inference for Sentence-level TextGraphs: How Challenging is Meaningfully Combining Information for Science Question Answering? *TextGraphs*.

Peter A Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton T Morrison. 2018. Worldtree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. *LREC*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *ACL*.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. *EMNLP*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 252–262.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. *EMNLP*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *EMNLP*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *EMNLP*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *EMNLP*, volume 3, page 4.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks. *arXiv preprint arXiv:1809.02040*.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What Makes Reading Comprehension Questions Easier? *EMNLP*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making Neural QA as Simple as Possible but not Simpler. *CoNLL*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *EMNLP*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *ICLR*.

Neural Grammatical Error Correction with Finite State Transducers

Felix Stahlberg[†] and **Christopher Bryant**[‡] and **Bill Byrne**[†]
[†]Department of Engineering
[‡] Department of Computer Science and Technology
University of Cambridge
United Kingdom
`{fs439, cjb255, wjb31}@cam.ac.uk`

Abstract

Grammatical error correction (GEC) is one of the areas in natural language processing in which purely neural models have not yet superseded more traditional symbolic models. Hybrid systems combining phrase-based statistical machine translation (SMT) and neural sequence models are currently among the most effective approaches to GEC. However, both SMT and neural sequence-to-sequence models require large amounts of annotated data. Language model based GEC (LM-GEC) is a promising alternative which does not rely on annotated training data. We show how to improve LM-GEC by applying modelling techniques based on finite state transducers. We report further gains by rescoring with neural language models. We show that our methods developed for LM-GEC can also be used with SMT systems if annotated training data is available. Our best system outperforms the best published result on the CoNLL-2014 test set, and achieves far better relative improvements over the SMT baselines than previous hybrid systems.

1 Introduction

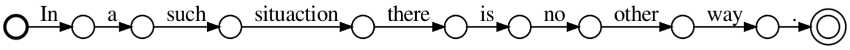
Grammatical error correction (GEC) is the task of automatically correcting all types of errors in text; e.g. [*In a such situaction* → *In such a situation*]. Using neural models for GEC is becoming increasingly popular (Xie et al., 2016; Yuan and Briscoe, 2016; Ji et al., 2017; Sakaguchi et al., 2017; Schmaltz et al., 2017; Chollampatt and Ng, 2018; Ge et al., 2018a,b), possibly combined with phrase-based SMT (Chollampatt et al., 2016; Chollampatt and Ng, 2017; Grundkiewicz and Junczys-Dowmunt, 2018). A potential challenge for purely neural GEC models is their vast output space since they assign non-zero probability mass to any sequence. GEC is – compared to machine translation – a highly constrained problem as corrections tend to be very local, and lexical choices are usually limited. Finite state transducers (FSTs) are an efficient way to represent large structured search spaces. In this paper, we propose to construct a hypothesis space using standard FST operations like composition, and then constrain the output of a neural GEC system to that space. We study two different scenarios: In the first scenario, we do not have access to annotated training data, and only use a small development set for tuning. In this scenario, we construct the hypothesis space using word-level context-independent confusion sets (Bryant and Briscoe, 2018) based on spell checkers and morphology databases, and rescore it with count-based and neural language models (NLMs). In the second scenario, we assume to have enough training data available to train SMT and neural machine translation (NMT) systems. In this case, we make additional use of the SMT lattice and rescore with an NLM-NMT ensemble. Our contributions are:

- We present an FST-based adaptation of the work of Bryant and Briscoe (2018) which allows exact inference, and does not require annotated training data. We report large gains from rescoring with a neural language model.
- Our technique beats the best published result with comparable amounts of training data on the CoNLL-2014 (Ng et al., 2014) test set when applied to SMT lattices. Our combination strategy yields larger gains over the SMT baselines than simpler rescoring or pipelining used in prior work on hybrid systems (Grundkiewicz and Junczys-Dowmunt, 2018).

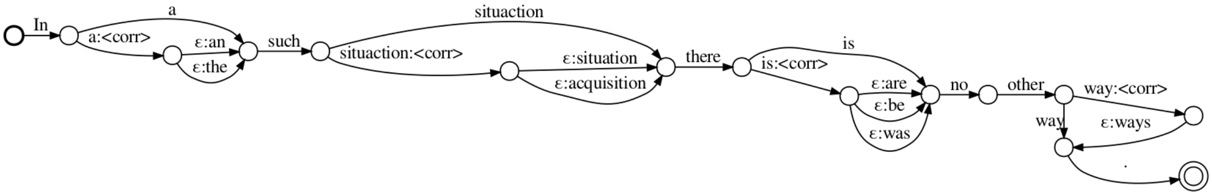
2 Constructing the Hypothesis Space

Constructing the set of hypotheses The core idea of our approach is to first construct a

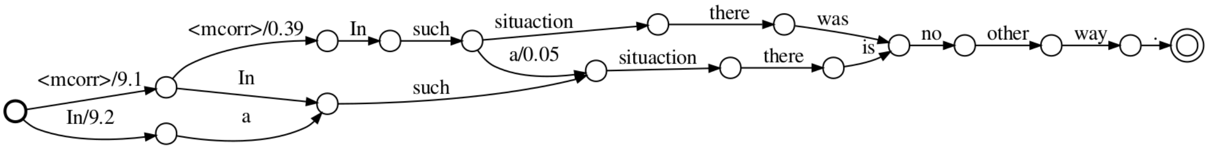
Proceedings of NAACL-HLT 2019, pages 4033–4039
Minneapolis, Minnesota, June 2 - June 7, 2019.



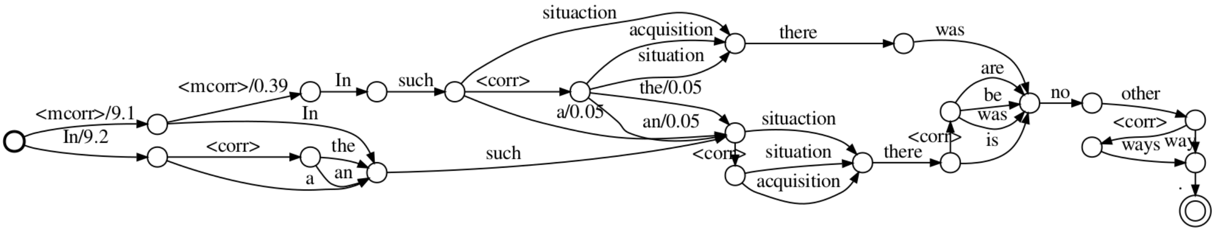
(a) The input lattice I without SMT (no annotated training data).



(b) The base lattice B without SMT.



(c) The input lattice I with SMT.



(d) The base lattice B with SMT.

Figure 1: Building the hypothesis space for the input sentence “In a such situaction there is no other way .”.

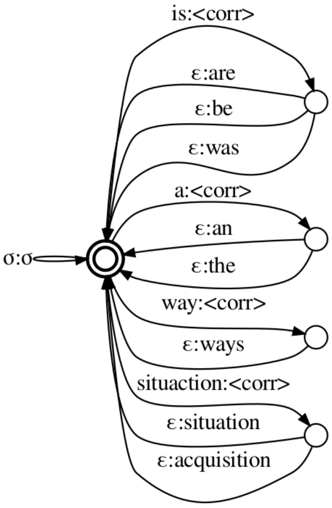


Figure 2: The edit transducer E. The σ-label can match any input symbol.

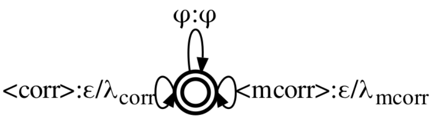


Figure 3: The penalization transducer P. The ϕ-label can match any input except <corr> and <mcorr>.

(weighted) hypothesis space H which is large enough to be likely to contain good corrections, but constrained enough to embrace the highly structured nature of GEC. Then, we use H to constrain a neural beam decoder. We make extensive use of the FST operations available in OpenFST (Allauzen et al., 2007) like composition (denoted with the $\circ$-operator) and projection (denoted with $\Pi_{\text{input}}(\cdot)$ and $\Pi_{\text{output}}(\cdot)$) to build H. The process starts with an input lattice I. In our experiments without annotated training data, I is an FST which simply maps the input sentence to itself as shown in Fig. 1(a). If we do have access to enough annotated data, we train an SMT system on it and derive I from the SMT n-best list.[1] For each hypothesis $\mathbf{y}$ we compute the Levenshtein distance $\text{lev}(\mathbf{x}, \mathbf{y})$ to the source sentence $\mathbf{x}$. We construct a string $\mathbf{z}$ by prepending $\text{lev}(\mathbf{x}, \mathbf{y})$ many `<mcorr>` tokens to $\mathbf{y}$, and construct I such that:

$$\mathbf{z} = (\texttt{<mcorr>})^{\text{lev}(\mathbf{x},\mathbf{y})} \cdot \mathbf{y} \qquad (1)$$

$$[[I]](\mathbf{z}) = -\lambda_{\text{SMT}} \text{SMT}(\mathbf{y}|\mathbf{x}). \qquad (2)$$

---

[1] In the rare cases in which the n-best list did not contain the source sentence $\mathbf{x}$ we added it in a postprocessing step.

We adapt the notation of Mohri (2003) and denote the cost I assigns to mapping a string $\mathbf{z}$ to itself as $[[I]](\mathbf{z})$, and set $[[I]](\mathbf{z}) = \infty$ if I does not accept $\mathbf{z}$. $\text{SMT}(\mathbf{y}|\mathbf{x})$ is the SMT score. In other words, I represents the weighted SMT n-best list after adding $\text{lev}(\mathbf{x}, \mathbf{y})$ many `<mcorr>` tokens to each hypothesis as illustrated in Fig. 1(c). We scale SMT scores by a factor λ_{SMT} for tuning.

Bryant and Briscoe (2018) addressed substitution errors such as non-words, morphology-, article-, and preposition-errors by creating confusion sets $C(x_i)$ that contain possible (context-independent) 1:1 corrections for each input word x_i. Specifically, they relied on CyHunspell for spell checking (Rodriguez and Seal, 2014), the AGID morphology database for morphology errors (Atkinson, 2011), and manually defined confusion sets for determiner and preposition errors, hence avoiding the need for annotated training data. We use the same confusion sets as Bryant and Briscoe (2018) to augment our hypothesis space via the edit flower transducer E shown in Fig. 2. E can map any sequence to itself via its σ-self-loop. Additionally, it allows the mapping $x_i \rightarrow$ `<corr>` $\cdot\, y$ for each $y \in C(x_i)$. For example, for the misspelled word $x_i =$ 'situaction' and the confusion set $C(\text{'situaction'}) = \{\text{'situation'}, \text{'acquisition'}\}$, E allows mapping 'situaction' to '`<corr>` situation' and '`<corr>` acquisition', and to itself via the σ-self-loop. The additional `<corr>` token will help us to keep track of the edits. We obtain our *base lattice* B which defines the set of possible hypotheses by composition and projection:

$$B := \Pi_{\text{output}}(I \circ E). \tag{3}$$

Fig. 1(d) shows B for our running example.

Scoring the hypothesis space We apply multiple scoring strategies to the hypotheses in B. First, we penalize `<mcorr>` and `<corr>` tokens with two further parameters, λ_{mcorr} and λ_{corr}, by composing B with the penalization transducer P shown in Fig. 3.[2] The λ_{mcorr} and λ_{corr} parameters control the trade-off between the number and quality of the proposed corrections since high values bias towards fewer corrections.

To incorporate word-level language model scores we train a 5-gram count-based LM with KenLM (Heafield, 2011) on the One Billion Word Benchmark dataset (Chelba et al., 2014), and convert it to an FST L using the OpenGrm NGram Library (Roark et al., 2012). For tuning purposes we scale weights in L with λ_{KenLM}:

$$[[L]](\mathbf{y}) = -\lambda_{\text{KenLM}} \log P_{\text{KenLM}}(\mathbf{y}). \tag{4}$$

Our combined word-level scores can be expressed with the following transducer:

$$H_{\text{word}} = B \circ P \circ L. \tag{5}$$

Since we operate in the tropical semiring, path scores in H_{word} are linear combinations of correction penalties, LM scores, and, if applicable, SMT scores, weighted with the λ-parameters. Note that exact inference in H_{word} is possible using FST shortest path search. This is an improvement over the work of Bryant and Briscoe (2018) who selected correction options greedily. Our ultimate goal, however, is to rescore H_{word} with neural models such as an NLM and – if annotated training data is available – an NMT model. Since our neural models use subword units (Sennrich et al., 2016, BPEs), we compose H_{word} with a transducer T which maps word sequences to BPE sequences. Our final transducer H_{BPE} which we use to constrain the neural beam decoder can be written as:

$$\begin{aligned} H_{\text{BPE}} &= \Pi_{\text{output}}(H_{\text{word}} \circ T) \\ &= \Pi_{\text{output}}(I \circ E \circ P \circ L \circ T). \end{aligned} \tag{6}$$

To help downstream beam decoding we apply ϵ-removal, determinization, minimization, and weight pushing (Mohri, 1997; Mohri and Riley, 2001) to H_{BPE}. We search for the best hypothesis $\mathbf{y}^*_{\text{BPE}}$ with beam search using a combined score of word-level symbolic models (represented by H_{BPE}) and subword unit based neural models:

$$\begin{aligned} \mathbf{y}^*_{\text{BPE}} = \arg\max_{\mathbf{y}_{\text{BPE}}} \Big(&-[[H_{\text{BPE}}]](\mathbf{y}_{\text{BPE}}) \\ &+ \lambda_{\text{NLM}} \log P_{\text{NLM}}(\mathbf{y}_{\text{BPE}}) \\ &+ \lambda_{\text{NMT}} \log P_{\text{NMT}}(\mathbf{y}_{\text{BPE}}|\mathbf{x}_{\text{BPE}}) \Big) \end{aligned} \tag{7}$$

The final decoding pass can be seen as an ensemble of a neural LM and an NMT model which is constrained and scored at each time step by the set of possible tokens in H_{BPE}.

---

[2] Rather than using `<mcorr>` and `<corr>` tokens and the transducer P we could directly incorporate the costs in the transducers I and E, respectively. We chose to use explicit correction tokens for clarity.

	Uses E	5-gram FST-LM	NLM (BPE)	CoNLL-2014 P	R	M2	GLEU	JFLEG Test P	R	M2	GLEU
1	Best published (B&B, 2018)			40.56	20.81	34.09	59.35	76.23	28.48	57.08	48.75
2	✓	✓		40.62	20.72	34.08	64.03	81.08	28.69	59.38	48.95
3	✓	✓	✓	54.43	25.21	44.19	66.75	79.88	32.99	62.20	50.93
4	✓	✓	✓	53.64	26.34	44.43	66.89	70.24	38.94	60.51	52.61

Table 1: Results without using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the source sentence as in Fig. 1(a).

	Uses E	5-gram FST-LM	NMT (BPE)	NLM (BPE)	CoNLL-2014 P	R	M2	GLEU	JFLEG Test P	R	M2	GLEU
1	Best published (G&J-D, 2018)				66.77	34.49	56.25	n/a	n/a	n/a	n/a	61.50
2	Unconstrained single NMT				54.98	22.20	42.45	67.19	67.49	38.47	58.64	50.71
3					60.95	26.21	48.18	68.30	66.64	40.68	59.09	50.86
4	✓	✓			57.58	32.39	49.83	68.82	71.60	42.45	62.95	53.20
5			✓	✓	65.26	33.03	54.61	69.92	76.35	40.55	64.89	51.75
6	✓		✓	✓	64.55	37.33	56.33	70.30	78.85	47.72	69.75	55.39
7	✓		✓(4x)	✓	66.71	38.97	58.40	70.60	82.15	47.82	71.84	55.60
8	✓		✓(4x)	✓	66.96	38.62	58.39	70.60	74.19	56.41	69.79	58.63

Table 2: Results with using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the Moses 1000-best list as in Fig. 1(c). Row 3 is the SMT baseline.

We have introduced three λ-parameters λ_{corr}, λ_{KenLM}, and λ_{NLM}, and three additional parameters λ_{SMT}, λ_{mcorr}, and λ_{NMT} if we make use of annotated training data. We also use a word insertion penalty λ_{wc} for our SMT-based experiments. We tune all these parameters on the development sets using Powell search (Powell, 1964).[3]

3 Experiments

Experimental setup In our experiments with annotated training data we use the SMT system of Junczys-Dowmunt and Grundkiewicz (2016)[4] to create 1000-best lists from which we derive the input lattices I. All our LMs are trained on the One Billion Word Benchmark dataset (Chelba et al., 2014). Our neural LM is a Transformer decoder architecture in the `transformer_base` configuration trained with Tensor2Tensor (Vaswani et al., 2018). Our NMT model is a Transformer model (`transformer_base`) trained on the concatenation of the NUCLE corpus (Dahlmeier et al., 2013) and the Lang-8 Corpus of Learner English v1.0 (Mizumoto et al., 2012). We only keep sentences with at least one correction (659K sentences in total). Both NMT and NLM models use byte pair encoding (Sennrich et al., 2016, BPE) with 32K merge operations. We delay SGD updates by 2 on four physical GPUs as suggested by Saunders et al. (2018). We decode with beam size 12 using the SGNMT decoder (Stahlberg et al., 2017). We evaluate on CoNLL-2014 (Ng et al., 2014) and JFLEG-Test (Napoles et al., 2017), using CoNLL-2013 (Ng et al., 2013) and JFLEG-Dev as development sets. Our evaluation metrics are GLEU (Napoles et al., 2015) and M2 (Dahlmeier and Ng, 2012). We generated M2 files using ERRANT (Bryant et al., 2017) for JFLEG and Tab. 1 to be comparable to Bryant and Briscoe (2018), but used the official M2 files in Tab. 2 to be comparable to Grundkiewicz and Junczys-Dowmunt (2018).

Results Our LM-based GEC results without using annotated training data are summarized in Tab. 1. Even when we use the same resources (same LM and same confusion sets) as Bryant and Briscoe (2018), we see gains on JFLEG (rows 1 vs. 2), probably because we avoid search errors in our FST-based scheme. Adding an NLM yields significant gains across the board. Tab. 2 shows that adding confusion sets to SMT lattices is effective even without neural models (rows 3 vs. 4). Rescoring with neural models also benefits from the confusion sets (rows 5 vs. 6). With our ensemble systems (rows 7 and 8) we are able to outperform prior work[5] (row 1) on CoNLL-2014 and

---

[3] Similarly to Bryant and Briscoe (2018), even in our experiments without annotated *training* data, we do need a very small amount of annotated sentences for tuning.

[4] `https://github.com/grammatical/baselines-emnlp2016`

[5] We compare our systems to the work of Grundkiewicz and Junczys-Dowmunt (2018) as they used similar training data. We note, however, that Ge et al. (2018b) reported even better results with much more (non-public) training data. Comparing (Ge et al., 2018a) and (Ge et al., 2018b) suggests that most of their gains come from the larger training set.

	G&J-D (2018)		This work	
	CoNLL (M2)	JFLEG (GLEU)	CoNLL (M2)	JFLEG (GLEU)
SMT	50.27	55.79	48.18	50.86
Hybrid	56.25	61.50	58.40	58.63
Rel. gain	**11.90%**	**10.23%**	**21.21%**	**15.28%**

Table 3: Improvements over SMT baselines.

come within 3 GLEU on JFLEG. Since the baseline SMT systems of Grundkiewicz and Junczys-Dowmunt (2018) were better than the ones we used, we achieve even higher relative gains over the respective SMT baselines (Tab. 3).

Error type analysis We also carried out a more detailed error type analysis of the best CoNLL-2014 M2 system with/without training data using ERRANT (Tab. 4). Specifically, this table shows that while the trained system was consistently better than the untrained system, the degree of the improvement differs significantly depending on the error type. In particular, since the untrained system was only designed to handle Replacement word errors, much of the improvement in the trained system comes from the ability to correct Missing and Unnecessary word errors. The trained system nevertheless still improves upon the untrained system in terms of replacement errors by 10 $F_{0.5}$ (45.53 vs. 55.63).

In terms of more specific error types, the trained system was also able to capture a wider variety of error types, including content word errors (adjectives, adverbs, nouns and verbs) and other categories such as pronouns and punctuation. Since the untrained system only targets spelling, orthographic and morphological errors however, it is interesting to note that the difference in scores between these categories tends to be smaller than others; e.g. noun number (53.43 vs 64.96), orthography (62.77 vs 74.07), spelling (67.91 vs 75.21) and subject-verb agreement (66.67 vs 68.39). This suggests that an untrained system is already able to capture the majority of these error types.

Oracle experiments Our FST-based composition cascade is designed to enrich the search space to allow the neural models to find better hypotheses. Tab. 5 reports the oracle sentence error rate for different configurations, i.e. the fraction of reference sentences in the test set which are not in the FSTs. Expanding the SMT lattice significantly reduces the oracle error rate from 55.63% to 48.17%.

	ERRANT $F_{0.5}$	
Type	**No train**	**Train**
Missing	-	51.96
Replacement	45.53	55.63
Unnecessary	-	50.38
ADJ	-	27.03
ADV	-	29.80
DET	19.17	55.01
MORPH	33.20	64.81
NOUN	4.31	34.88
NOUN:NUM	53.43	64.96
NOUN:POSS	-	13.51
ORTH	62.77	74.07
OTHER	2.45	18.39
PREP	34.39	56.58
PRON	-	40.91
PUNCT	-	46.08
SPELL	67.91	75.21
VERB	-	37.94
VERB:FORM	48.03	63.33
VERB:SVA	66.67	68.39
VERB:TENSE	35.39	47.90

Table 4: A selection of ERRANT $F_{0.5}$ error type scores comparing the best CoNLL-2014 system with and without training data. A dash means the system did not attempt to correct the error type.

Hypothesis space	**Error rate**
Expanded input sentence (Tab. 1)	61.28%
SMT lattice (Tab. 2, rows 3, 5)	55.64%
Expanded SMT lattice (Tab. 2, rows 4, 6-8)	48.17%

Table 5: Oracle error rates for different hypothesis spaces using the first annotator in CoNLL-2014.

4 Conclusion

We demonstrated that our FST-based approach to GEC outperforms prior work on LM-based GEC significantly, especially when combined with a neural LM. We also applied our approach to SMT lattices and reported much better relative gains over the SMT baselines than previous work on hybrid systems. Our results suggest that FSTs provide a powerful and effective framework for constraining neural GEC systems.

Acknowledgements

This paper reports on research supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1) and Cambridge Assessment, University of Cambridge.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

Kevin Atkinson. 2011. Automatically generated inflection database (AGID). `http://wordlist.aspell.net/other/`. [Online; accessed 24-December-2018].

Christopher Bryant and Ted Briscoe. 2018. Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH-2014)*, pages 2635–2639.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2768–2774. AAAI Press.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556. Association for Computational Linguistics.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872. The COLING 2012 Organizing Committee.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Mehryar Mohri. 2003. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982.

Mehryar Mohri and Michael Riley. 2001. A weight pushing algorithm for large vocabulary speech recognition. In *Seventh European Conference on Speech Communication and Technology*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12. Association for Computational Linguistics.

Michael JD Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66. Association for Computational Linguistics.

Tim Rodriguez and Matthew Seal. 2014. Cy-Hunspell. `https://github.com/MSeal/cython_hunspell`. [Online; accessed 24-December-2018].

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372. Asian Federation of Natural Language Processing.

Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018. Multi-representation ensembles and delayed SGD updates improve syntax-based NMT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 319–325. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Felix Stahlberg, Eva Hasler, Danielle Saunders, and Bill Byrne. 2017. SGNMT – A flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 25–30. Association for Computational Linguistics.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199. Association for Machine Translation in the Americas.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386. Association for Computational Linguistics.

Convolutional Self-Attention Networks

Baosong Yang[†] **Longyue Wang**[‡] **Derek F. Wong**[†] **Lidia S. Chao**[†] **Zhaopeng Tu**[‡*]
[†]NLP^{2}CT Lab, Department of Computer and Information Science, University of Macau
[‡]Tencent AI Lab
[†]`nlp2ct.baosong@gmail.com`, `{derekfw,lidiasc}@umac.mo`
[‡]`{vinnylywang,zptu}@tencent.com`

Abstract

Self-attention networks (SANs) have drawn increasing interest due to their high parallelization in computation and flexibility in modeling dependencies. SANs can be further enhanced with multi-head attention by allowing the model to attend to information from different representation subspaces. In this work, we propose novel *convolutional self-attention networks*, which offer SANs the abilities to 1) strengthen dependencies among neighboring elements, and 2) model the interaction between features extracted by multiple attention heads. Experimental results of machine translation on different language pairs and model settings show that our approach outperforms both the strong Transformer baseline and other existing models on enhancing the locality of SANs. Comparing with prior studies, the proposed model is parameter free in terms of introducing no more parameters.

1 Introduction

Self-attention networks (SANs) (Parikh et al., 2016; Lin et al., 2017) have shown promising empirical results in various natural language processing (NLP) tasks, such as machine translation (Vaswani et al., 2017), natural language inference (Shen et al., 2018a), and acoustic modeling (Sperber et al., 2018). One appealing strength of SANs lies in their ability to capture dependencies regardless of distance by explicitly attending to all the elements. In addition, the performance of SANs can be improved by multi-head attention (Vaswani et al., 2017), which projects the input sequence into multiple subspaces and applies attention to the representation in each subspace.

Despite their success, SANs have two major limitations. First, the model fully take into account all the elements, which disperses the attention distribution and thus overlooks the relation of neighboring elements and phrasal patterns (Yang et al., 2018; Wu et al., 2018; Guo et al., 2019). Second, multi-head attention extracts distinct linguistic properties from each subspace in a parallel fashion (Raganato and Tiedemann, 2018), which fails to exploit useful interactions across different heads. Recent work shows that better features can be learned if different sets of representations are present at feature learning time (Ngiam et al., 2011; Lin et al., 2014).

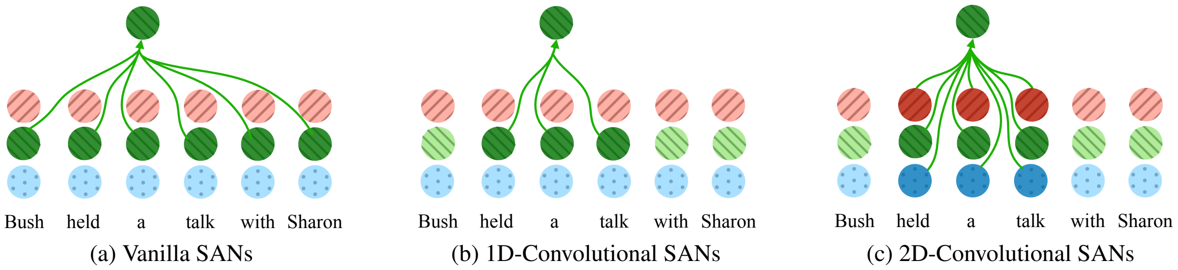
To this end, we propose novel *convolutional self-attention networks* (CSANs), which model locality for self-attention model and interactions between features learned by different attention heads in an unified framework. Specifically, in order to pay more attention to a local part of the input sequence, we restrict the attention scope to a window of neighboring elements. The localness is therefore enhanced via a parameter-free 1-dimensional convolution. Moreover, we extend the convolution to a 2-dimensional area with the axis of attention head. Thus, the proposed model allows each head to interact local features with its adjacent subspaces at attention time. We expect that the interaction across different subspaces can further improve the performance of SANs.

We evaluate the effectiveness of the proposed model on three widely-used translation tasks: WMT14 English-to-German, WMT17 Chinese-to-English, and WAT17 Japanese-to-English. Experimental results demonstrate that our approach consistently improves performance over the strong TRANSFORMER model (Vaswani et al., 2017) across language pairs. Comparing with previous work on modeling locality for SANs (e.g. Shaw et al., 2018; Yang et al., 2018; Sperber et al., 2018), our model boosts performance on both translation quality and training efficiency.

---

[*] Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Baosong Yang was interning at Tencent AI Lab.

Proceedings of NAACL-HLT 2019, pages 4040–4045
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) Vanilla SANs (b) 1D-Convolutional SANs (c) 2D-Convolutional SANs

Figure 1: Illustration of (a) vanilla SANs; (b) 1-dimensional convolution with the window size being 3; and (c) 2-dimensional convolution with the area being 3 × 3. Different colors and patterns represent different subspaces modeled by multi-head attention, and transparent colors denote masked tokens that are invisible to SANs.

2 Multi-Head Self-Attention Networks

SANs produce representations by applying attention to each pair of tokens from the input sequence, regardless of their distance. Vaswani et al. (2017) found it is beneficial to capture different contextual features with multiple individual attention functions. Given an input sequence $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_I\} \in \mathbb{R}^{I \times d}$, the model first transforms it into queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{I \times d} \quad (1)$$

where $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\} \in \mathbb{R}^{d \times d}$ are trainable parameters and d indicates the hidden size. The three types of representations are split into H different subspaces, e.g., $[\mathbf{Q}^1, \dots, \mathbf{Q}^H] = \mathbf{Q}$ with $\mathbf{Q}^h \in \mathbb{R}^{I \times \frac{d}{H}}$. In each subspace h, the element $\mathbf{o}_i^h$ in the output sequence $\mathbf{O}^h = \{\mathbf{o}_1^h, \dots, \mathbf{o}_I^h\}$ is calculated by

$$\mathbf{o}_i^h = \textsc{Att}(\mathbf{q}_i^h, \mathbf{K}^h)\mathbf{V}^h \qquad \in \mathbb{R}^{\frac{d}{H}} \quad (2)$$

where $\textsc{Att}(\cdot)$ is an attention model (Bahdanau et al., 2015; Vaswani et al., 2017) that retrieves the keys $\mathbf{K}^h$ with the query $\mathbf{q}_i^h$. The final output representation $\mathbf{O}$ is the concatenation of outputs generated by multiple attention models:

$$\mathbf{O} = [\mathbf{O}^1, \dots, \mathbf{O}^H] \qquad \in \mathbb{R}^{I \times d} \quad (3)$$

3 Approach

As shown in Figure 1(a), the vanilla SANs use the query $\mathbf{q}_i^h$ to compute a categorical distribution over all elements from $\mathbf{K}^h$ (Equation 2). It may inherit the attention to neighboring information (Yu et al., 2018; Yang et al., 2018; Guo et al., 2019). In this work, we propose to model locality for SANs by restricting the model to attend to a local region via convolution operations (1D-CSANs, Figure 1(b)). Accordingly, it provides distance-aware information (e.g. phrasal patterns), which is complementary to the distance-agnostic dependencies modeled by the standard SANs (Section 3.1).

Moreover, the calculation of output $\mathbf{o}^h$ are restricted to the a single individual subspace, overlooking the richness of contexts and the dependencies among groups of features, which have proven beneficial to the feature learning (Ngiam et al., 2011; Wu and He, 2018). We thus propose to convolute the items in adjacent heads (2D-CSANs, Figure 1(c)). The proposed model is expected to improve performance through interacting linguistic properties across heads (Section 3.2).

3.1 Locality Modeling via 1D Convolution

For each query $\mathbf{q}_i^h$, we restrict its attention region (e.g., $\mathbf{K}^h = \{\mathbf{k}_1^h, \dots, \mathbf{k}_i^h, \dots, \mathbf{k}_I^h\}$) to a local scope with a fixed size $M + 1$ ($M \leq I$) centered at the position i:

$$\widehat{\mathbf{K}}^h = \{\mathbf{k}_{i-\frac{M}{2}}^h, \dots, \mathbf{k}_i^h, \dots, \mathbf{k}_{i+\frac{M}{2}}^h\} \quad (4)$$

$$\widehat{\mathbf{V}}^h = \{\mathbf{v}_{i-\frac{M}{2}}^h, \dots, \mathbf{v}_i^h, \dots, \mathbf{v}_{i+\frac{M}{2}}^h\} \quad (5)$$

Accordingly, the calculation of corresponding output in Equation (2) is modified as:

$$\mathbf{o}_i^h = \textsc{Att}(\mathbf{q}_i^h, \widehat{\mathbf{K}}^h)\widehat{\mathbf{V}}^h \quad (6)$$

As seen, SANs are only allowed to attend to the neighboring tokens (e.g., $\widehat{\mathbf{K}}^h$, $\widehat{\mathbf{V}}^h$), instead of all the tokens in the sequence (e.g., $\mathbf{K}^h$, $\mathbf{V}^h$).

The SAN-based models are generally implemented as multiple layers, in which higher layers tend to learn semantic information while lower layers capture surface and lexical information (Peters et al., 2018; Raganato and Tiedemann, 2018). Therefore, we merely apply locality modeling to the lower layers, which same to the configuration in Yu et al. (2018) and Yang et al. (2018). In this way, the representations are learned in a hierarchical fashion (Yang et al., 2017). That is, the distance-aware and local information extracted by the lower SAN layers, is expected to complement distance-agnostic and global information captured by the higher SAN layers.

3.2 Attention Interaction via 2D Convolution

Mutli-head mechanism allows different heads to capture distinct linguistic properties (Raganato and Tiedemann, 2018; Li et al., 2018), especially in diverse local contexts (Yang et al., 2018). We hypothesis that exploiting local properties across heads can further improve the performance of SANs. To this end, we expand the 1-dimensional window to a 2-dimensional area with the new dimension being the index of attention head. Suppose that the area size is $(N+1) \times (M+1)$ $(N \leq H)$, the keys and values in the area are:

$$\widetilde{\mathbf{K}}^h = \bigcup[\widehat{\mathbf{K}}^{h-\frac{N}{2}}, \dots, \widehat{\mathbf{K}}^h, \dots, \widehat{\mathbf{K}}^{h+\frac{N}{2}}] \quad (7)$$

$$\widetilde{\mathbf{V}}^h = \bigcup[\widehat{\mathbf{V}}^{h-\frac{N}{2}}, \dots, \widehat{\mathbf{V}}^h, \dots, \widehat{\mathbf{V}}^{h+\frac{N}{2}}] \quad (8)$$

where $\widehat{\mathbf{K}}^h, \widehat{\mathbf{V}}^h$ are elements in the h-th subspace, which are calculated by Equations 4 and 5 respectively. The union operation $\bigcup$ means combining the keys and values in different subspaces. The corresponding output is calculated as:

$$\mathbf{o}_i^h = \textsc{Att}(\mathbf{q}_i^h, \widetilde{\mathbf{K}}^h)\widetilde{\mathbf{V}}^h \quad (9)$$

The 2D convolution allows SANs to build relevance between elements across adjacent heads, thus flexibly extract local features from different subspaces rather than merely from an unique head.

The vanilla SAN models linearly aggregate features from different heads, and this procedure limits the extent of abstraction (Fukui et al., 2016; Li et al., 2019). Multiple sets of representations presented at feature learning time can further improve the expressivity of the learned features (Ngiam et al., 2011; Wu and He, 2018).

4 Related Work

Self-Attention Networks Recent studies have shown that SANs can be further improved by capturing complementary information. For example, Hao et al. (2019) complemented SANs with recurrence modeling, while Yang et al. (2019) modeled contextual information for SANs.

Concerning modeling locality for SANs, Yu et al. (2018) injected several CNN layers (Kim, 2014) to fuse local information, the output of which is fed to the subsequent SAN layer. Several researches proposed to revise the attention distribution with a parametric localness bias, and succeed on machine translation (Yang et al., 2018) and natural language inference (Guo et al., 2019). While both models introduce additional parameters, our approach is a more lightweight solution without introducing any new parameters. Closely related to this work, Shen et al. (2018a) applied a positional mask to encode temporal order, which only allows SANs to attend to the previous or following tokens in the sequence. In contrast, we employ a positional mask (i.e. the tokens outside the local window is masked as 0) to encode the distance-aware local information.

In the context of distance-aware SANs, Shaw et al. (2018) introduced relative position encoding to consider the relative distances between sequence elements. While they modeled locality from position embedding, we improve locality modeling from revising attention scope. To make a fair comparison, we re-implemented the above approaches under a same framework. Empirical results on machine translation tasks show the superiority of our approach in both translation quality and training efficiency.

Multi-Head Attention Multi-head attention mechanism (Vaswani et al., 2017) employs different attention heads to capture distinct features (Raganato and Tiedemann, 2018). Along this direction, Shen et al. (2018a) explicitly used multiple attention heads to model different dependencies of the same word pair, and Strubell et al. (2018) employed different attention heads to capture different linguistic features. Li et al. (2018) introduced disagreement regularizations to encourage the diversity among attention heads. Inspired by recent successes on fusing information across layers (Dou et al., 2018, 2019), Li et al. (2019) proposed to aggregate information captured by different attention heads. Based on these findings, we model interactions among attention heads to exploit the richness of local properties distributed in different heads.

5 Experiments

We conducted experiments with the Transformer model (Vaswani et al., 2017) on English⇒German (En⇒De), Chinese⇒English (Zh⇒En) and Japanese⇒English (Ja⇒En) translation tasks.

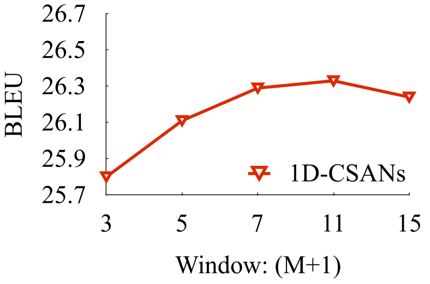
For the En⇒De and Zh⇒En tasks, the models were trained on widely-used WMT14 and WMT17 corpora, consisting of around 4.5 and 20.62 million sentence pairs, respectively. Concerning Ja⇒En, we used the first two sections of WAT17 corpus as the training data, which consists of 2M sentence pairs. To reduce the vocabulary size, all the data were tokenized and segmented into subword symbols using byte-pair encoding (Sennrich et al., 2016) with 32K merge operations. Following Shaw et al. (2018), we incorporated the proposed model into the encoder, which is a stack of 6 SAN layers. Prior studies revealed that modeling locality in lower layers can achieve better performance (Shen et al., 2018b; Yu et al., 2018; Yang et al., 2018), we applied our approach to the lowest three layers of the encoder. About configurations of NMT models, we used the *Base* and *Big* settings same as Vaswani et al. (2017), and all models were trained on 8 NVIDIA P40 GPUs with a batch of 4096 tokens.

5.1 Effects of Window/Area Size

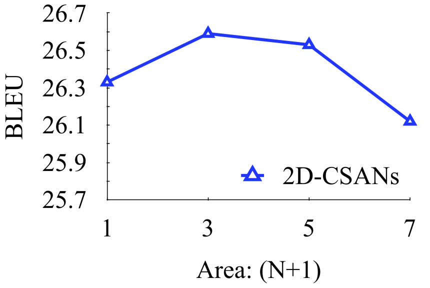
We first investigated the effects of window size (1D-CSANs) and area size (2D-CSANs) on En⇒De validation set, as plotted in Figure 2. For 1D-CSANs, the local size with 11 is superior to other settings. This is consistent with Luong et al. (2015) who found that 10 is the best window size in their local attention experiments. Then, we fixed the number of neighboring tokens being 11 and varied the number of heads. As seen, by considering the features across heads (i.e. > 1), 2D-CSANs further improve the translation quality. However, when the number of heads in attention goes up, the translation quality inversely drops. One possible reason is that the model still has the flexibility of learning a different distribution for each head with few interactions, while a large amount of interactions assumes more heads make “similar contributions” (Wu and He, 2018).

5.2 Accuracy of Phrase Translation

One intuition of our approach is to capture useful phrasal patterns via modeling locality. To evaluate the accuracy of phrase translations, we calculate the improvement of the proposed approaches over multiple granularities of n-grams, as shown in Figure 3. Both the two model variations consistently outperform the baseline on larger granularities, indicating that modeling locality can raise the ability of self-attention model on capturing the phrasal information. Furthermore, the dependencies among heads can be complementary to the localness modeling, which reveals the necessity of the interaction of features in different subspaces.



(a) 1D-CSANs



(b) 2D-CSANs

Figure 2: Effects of (a) window size on 1D-CSANs, and (b) attended head numbers on 2D-CSANs. For 2D-CSANs, the window size dimension is fixed to be 11.

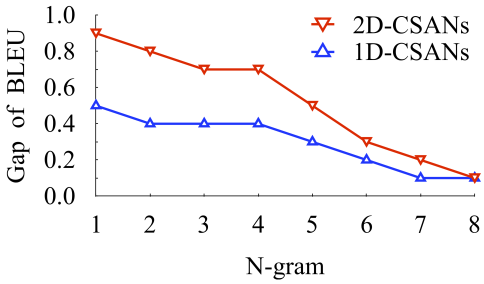


Figure 3: Performance improvement on different n-grams. “Gap of BLEU” denotes the improvement achieved by the proposed models over the baseline.

5.3 Comparison to Related Work

We re-implemented and compared several exiting works (Section 4) upon the same framework. Table 1 lists the results on the En⇒De translation task. As seen, all the models improve translation quality, reconfirming the necessity of modeling locality and distance information. Besides, our models outperform all the existing works, indicating the superiority of the proposed approaches. In particular, CSANs achieve better performance than

Model	Parameter	Speed	BLEU	△
Transformer-Base (Vaswani et al., 2017)	88.0M	1.28	27.31	-
+ Bi_Direct (Shen et al., 2018a)	+0.0M	-0.00	27.58	+0.27
+ Rel_Pos (Shaw et al., 2018)	+0.1M	-0.11	27.63	+0.32
+ Neighbor (Sperber et al., 2018)	+0.4M	-0.06	27.60	+0.29
+ Local_Hard (Luong et al., 2015)	+0.4M	-0.06	27.73	+0.42
+ Local_Soft (Yang et al., 2018)	+0.8M	-0.09	27.81	+0.50
+ Block (Shen et al., 2018b)	+6.0M	-0.33	27.59	+0.28
+ CNNs (Yu et al., 2018)	+42.6M	-0.54	27.70	+0.39
+ 1D-CSANs	+0.0M	-0.00	**27.86**	+0.55
+ 2D-CSANs	+0.0M	-0.06	**28.18**	+0.87

Table 1: Comparing with the existing approaches on WMT14 En⇒De translation task. For a fair comparison, we re-implemented the existing locality approaches under the same framework. “Parameter” denotes the number of model parameters (M = million) and “Speed” denotes the training speed (steps/second). “△” column denotes performance improvements over the Transformer baseline.

Model	WMT14 En⇒De		WMT17 Zh⇒En		WAT17 Ja⇒En	
	Speed	BLEU	Speed	BLEU	Speed	BLEU
Transformer-Base	1.28	27.31	1.21	24.13	1.33	28.10
+ CSANs	1.22	$28.18^{\Uparrow}$	1.16	$24.80^{\Uparrow}$	1.28	$28.50^{\uparrow}$
Transformer-Big	0.61	28.58	0.58	24.56	0.65	28.41
+ CSANs	0.50	28.74	0.48	$25.01^{\uparrow}$	0.55	$28.73^{\uparrow}$

Table 2: Experimental results on WMT14 En⇒De, WMT17 Zh⇒En and WAT17 Ja⇒En test sets. “Speed” denotes the training speed (steps/second). “↑ / ⇑” indicates statistically significant difference from the vanilla self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling (Koehn, 2004).

CNNs, revealing that extracting local features with dynamic weights is superior to assigning fixed parameters. Moreover, while most of the existing approaches (except for Shen et al. (2018a)) introduce new parameters, our methods are parameter-free and thus only marginally affect training efficiency.

5.4 Universality of The Proposed Model

To validate the universality of our approach on MT tasks, we evaluated the proposed approach on different language pairs and model settings. Table 2 lists the results on En⇒De, Zh⇒En and Ja⇒En translation tasks. As seen, our model consistently improves translation quality across language pairs, which demonstrates the effectiveness and universality of the proposed approach. It is encouraging to see that CSANs with *base* setting yields comparable performance with Transformer-Big.

6 Conclusion

In this paper, we propose a parameter-free convolutional self-attention model to enhance the feature extraction of neighboring elements across multiple heads. Empirical results of machine translation task on a variety of language pairs demonstrate the effectiveness and universality of the proposed methods. The extensive analyses suggest that: 1) modeling locality is beneficial to SANs; 2) interacting features across multiple heads at attention time can further improve the performance; and 3) to some extent, the dynamic weights are superior to their fixed counterpart (i.e. CSANs vs. CNNs) on local feature extraction. Moreover, it is interesting to validate the proposed model in other sequence modeling tasks.

Acknowledgments

The work was partly supported by the National Natural Science Foundation of China (Grant No. 61672555), the Joint Project of Macao Science and Technology Development Fund and National Natural Science Foundation of China (Grant No. 045/2017/AFJ) and the Multiyear Research Grant from the University of Macau (Grant No. MYRG2017-00087-FST). We thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting Deep Representations for Neural Machine Translation. In *EMNLP*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic Layer Aggregation for Neural Machine Translation. In *AAAI*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *EMNLP*.

Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian Transformer: A Lightweight Approach for Natural Language Inference. In *AAAI*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling Recurrence for Transformer. In *NAACL*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *EMNLP*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R. Lyu, and Zhaopeng Tu. 2019. Information Aggregation for Multi-Head Attention with Routing-by-Agreement. In *NAACL*.

Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in Network. In *ICLR*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-Aattentive Sentence Embedding. In *ICLR*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal Deep Learning. In *ICML*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

Alessandro Raganato and Jörg Tiedemann. 2018. An Analysis of Encoder Representations in Transformer-Based Machine Translation. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. In *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. In *ICLR*.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. 2018. Self-Attentional Acoustic Models. *Interspeech*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS*.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level Self-Attention Networks for Universal Sentence Encoding. In *EMNLP*.

Yuxin Wu and Kaiming He. 2018. Group Normalization. *arXiv:1803.08494*.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019. Context-Aware Self-Attention Networks. In *AAAI*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling Localness for Self-Attention Networks. In *EMNLP*.

Baosong Yang, Derek F Wong, Tong Xiao, Lidia S Chao, and Jingbo Zhu. 2017. Towards Bidirectional Hierarchical Representations for Attention-based Neural Machine Translation. In *EMNLP*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining Local Convolution with Global Self-attention for Reading Comprehension. In *ICLR*.

Rethinking Complex Neural Network Architectures for Document Classification

Ashutosh Adhikari[*] **Achyudh Ram**[*] **Raphael Tang,** and **Jimmy Lin**
David R. Cheriton School of Computer Science
University of Waterloo
{adadhika, arkeshav, r33tang, jimmylin}@uwaterloo.ca

Abstract

Neural network models for many NLP tasks have grown increasingly complex in recent years, making training and deployment more difficult. A number of recent papers have questioned the necessity of such architectures and found that well-executed, simpler models are quite effective. We show that this is also the case for document classification: in a large-scale reproducibility study of several recent neural models, we find that a simple BiLSTM architecture with appropriate regularization yields accuracy and F_1 that are either competitive or exceed the state of the art on four standard benchmark datasets. Surprisingly, our simple model is able to achieve these results *without* attention mechanisms. While these regularization techniques, borrowed from language modeling, are not novel, to our knowledge we are the first to apply them in this context. Our work provides an open-source platform and the foundation for future work in document classification.

1 Introduction

Recent developments in neural architectures for a wide range of NLP tasks can be characterized as a drive towards increasingly complex network components and modeling techniques. Worryingly, these new models are accompanied by smaller and smaller improvements in effectiveness on standard benchmark datasets, which leads us to wonder if observed improvements are "real". There is, however, ample evidence to the contrary. To provide a few examples: Melis et al. (2018) report that standard LSTM architectures outperform more recent models when properly tuned. Vaswani et al. (2017) show that sequence transduction using encoder–decoder networks with attention mechanisms work just as well with the attention module only, making most of the complex neural machinery unnecessary. Mohammed et al. (2018) show that simple RNN- and CNN-based models yield accuracies rivaling far more complex architectures in simple question answering over knowledge graphs.

Perhaps most damning are the indictments of Sculley et al. (2018), who lament the lack of empirical rigor in our field and cite even more examples where improvements can be attributed to far more mundane reasons (e.g., hyperparameter tuning) or are simply noise. Lipton and Steinhardt (2018) concur with these sentiments, adding that authors often use fancy mathematics to obfuscate or to impress (reviewers) rather than to clarify. Complex architectures are more difficult to train, more sensitive to hyperparameters, and brittle with respect to domains with different data characteristics—thus both exacerbating the "crisis of reproducibility" and making it difficult for practitioners to deploy networks that tackle real-world problems in production environments.

Like the papers cited above, we question the need for overly complex neural architectures, focusing on the problem of document classification. Starting with a large-scale reproducibility study of several recent neural models, we find that a simple bi-directional LSTM (BiLSTM) architecture with appropriate regularization yields accuracy and F_1 that are either competitive or exceed the state of the art on four standard benchmark datasets. As the closest comparison point, we find no benefit to the hierarchical modeling proposed by Yang et al. (2016) and we are able to achieve good classification results *without* attention mechanisms. While these regularization techniques, borrowed from language modeling, are not novel, we are to our knowledge the first to apply them in this context. Our work provides an open-source platform and the foundation for future work in document classification.

---

[*] Equal contribution.

Proceedings of NAACL-HLT 2019, pages 4046–4051
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Background and Related Work

2.1 Document Classification

Over the last few years, deep neural networks have achieved the state of the art in document classification. One popular model, hierarchical attention network (HAN), uses word- and sentence-level attention in classifying documents (Yang et al., 2016). Although this model nicely captures the intuition that modeling word sequences in sentences should be handled separately from sentence-level discourse modeling, one wonders if such complex architectures are really necessary, especially given the size of training data available today.

An important variant of document classification is the multi-label, multi-class case. Liu et al. (2017) develop XML-CNNs for multi-label text classification, basing the architecture on KimCNN (Kim, 2014) with increased filter sizes and an additional fully-connected layer. They also incorporate dynamic adaptive max-pooling (Chen et al., 2015) instead of the vanilla max-pooling over time in KimCNN. The paper compares with CNN-based approaches for the multi-label task, but only reports precision and disregards recall. Yang et al. (2018) instead adopts encoder–decoder sequence generation models (SGMs) for generating multiple labels for each document. Similar to our critique of HAN, we opine against the high complexity of these multi-label approaches.

2.2 Regularizing RNNs

There have been attempts to extend dropout (Srivastava et al., 2014) from feedforward neural networks to recurrent ones. Unfortunately, direct application of dropout on the hidden units of an RNN empirically harms its ability to retain long-term information (Zaremba et al., 2014). Recently, however, Merity et al. (2018) successfully apply dropout-like techniques to regularize RNNs for language modeling, achieving competitive word-level perplexity on multiple datasets. Inspired by this development, we adopt two of their regularization techniques, embedding dropout and weight-dropped LSTMs, to our task of document classification.

Weight-dropped LSTM. LSTMs comprise eight total input–hidden and hidden–hidden weight matrices; in weight dropping, Merity et al. (2018) regularize the four hidden–hidden matrices with DropConnect (Wan et al., 2013). The operation is applied only once per sequence, using the same dropout mask across multiple timesteps. Conveniently, this allows practitioners to use fast, out-of-the-box LSTM implementations without affecting the RNN formulation or training performance.

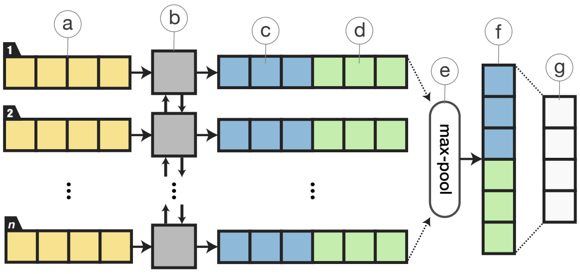


Figure 1: Illustration of the model architecture, where the labels are the following: (a) input word embeddings (b) BiLSTM (c, d) concatenated forward $h^f_{1:n}$ and backward $h^b_{1:n}$ hidden features (e) max-pooling over time (f) document feature vector (g) softmax or sigmoid output.

Embedding Dropout. Introduced in Gal and Ghahramani (2016) and successfully employed for neural language modeling (Merity et al., 2018), embedding dropout performs dropout on entire word embeddings, effectively removing some of the words at each training iteration. As a result, the technique conditions the model to be robust against missing input; for document classification, this discourages the model from relying on a small set of input words for prediction.

3 BiLSTM Model

We design our model to be minimalistic: First, we feed the word embeddings $\mathbf{w}_{1:n}$ of a document to a single-layer BiLSTM, extracting concatenated forward and backward word-level context vectors $\mathbf{h}_{1:n} = \mathbf{h}^f_{1:n} \oplus \mathbf{h}^b_{1:n}$. Subsequently, we max-pool $\mathbf{h}_{1:n}$ across time to yield document vector $\mathbf{d}$—see Figure 1, labels a–f. Finally, we feed $\mathbf{d}$ to a sigmoid or a softmax layer over the labels, depending on if the task type is multi-label or single-label classification (label g).

Contrary to prior art, our approach refrains from attention, hierarchical structure, and sequence generation, each of which increases model complexity. For one, hierarchical structure requires sentence-level tokenization and multiple RNNs. For another, sequence generation uses an encoder–decoder architecture, reducing computational parallelism. All three methods add depth to the model; our approach instead uses a single-layer BiLSTM with trivial max-pooling and concatenation operations, which makes for both simple implementation and resource-efficient inference.

4 Experimental Setup

We conduct a large-scale reproducibility study involving HAN, XML-CNN, KimCNN, and SGM. These are compared to our proposed model, referred to as LSTM_{reg}, as well as an ablated variant without regularization, denoted LSTM_{base}. The implementation of our model as well as from-scratch reimplementations of all the comparison models (except for SGM) are provided in our toolkit called Hedwig, which we make publicly available to serve as the foundation for future work.[1] In addition, we compare the neural approaches to logistic regression (LR) and support vector machines (SVMs). The LR model is trained using a one-vs-rest multi-label objective, while the SVM is trained with a linear kernel. Both of these methods use word-level tf–idf vectors of the documents as features.

All of our experiments are performed on Nvidia GTX 1080 and RTX 2080 Ti GPUs, with PyTorch 0.4.1 as the backend framework. We use Scikit-learn 0.19.2 for computing the tf–idf vectors and implementing LR and SVMs.

4.1 Datasets

We evaluate our models on the following four datasets: Reuters-21578, arXiv Abstract Paper dataset (AAPD), IMDB, and Yelp 2014. Reuters and AAPD are multi-label datasets, whereas IMDB and Yelp are single-label ones. For IMDB and Yelp, we use random sampling to split the dataset such that 80% is used for training, 10% for validation, and 10% for test. We use the standard ModApte splits (Apté et al., 1994) for the Reuters dataset, and author-defined splits for AAPD (Yang et al., 2018). We summarize the statistics of these datasets in Table 1.

Unfortunately, there is little consensus within the natural language processing community for choosing the splits of IMDB and Yelp 2014. Furthermore, they are often unreported in modeling papers, hence preventing direct comparison with past results. We are not able to find the exact splits Yang et al. (2016) use; for consistency, we use the same proportion the authors report, but of course this yields different samples in each split.

Dataset	C	N	W	S
Reuters	90	10,789	144.3	6.6
AAPD	54	55,840	167.3	1.0
IMDB	10	135,669	393.8	14.4
Yelp 2014	5	1,125,386	148.8	9.1

Table 1: Summary of the datasets. C denotes the number of classes in the dataset, N the number of samples, and W and S the average number of words and sentences per document, respectively.

For the multi-label datasets, we report the well-known micro-averaged F_1 score, which is the class-weighted harmonic mean between recall and precision. For the single-label datasets, we compare the models using accuracy.

4.2 Training and Hyperparameters

To ensure a fair comparison, we tune the hyperparameters for all baseline models. For HAN, we use a batch size of 32 across all the datasets, with a learning rate of 0.01 for Reuters and 0.001 for the rest. To train XML-CNN, we select a dynamic pooling window length of eight, a learning rate of 0.001, and 128 output channels, with batch sizes of 32 and 64 for single-label and multi-label datasets, respectively. For KimCNN, we use a batch size of 64 with a learning rate of 0.01. For training SGM on Reuters, we use the source code provided by the authors[2] and follow the same hyperparameters in their paper (Yang et al., 2018). For the LR and SVM models, we use the default set of hyperparameters in Scikit-learn.

For LSTM_{reg} and LSTM_{base}, we use the Adam optimizer with a learning rate of 0.01 on Reuters and 0.001 on the rest of the datasets, using batch sizes of 32 and 64 for multi-label and single-label tasks, respectively. For LSTM_{reg}, we also apply temporal averaging (TA): as shown in Kingma and Ba (2014), TA reduces both generalization error and stochastic noise in recent parameter estimates from stochastic approximation. We set the default TA exponential smoothing coefficient of β_{EMA} to 0.99. We choose 512 hidden units for the Bi-LSTM models, whose max-pooled output is regularized using a dropout rate of 0.5. We also regularize the input–hidden and hidden–hidden Bi-LSTM connections using embedding dropout and weight dropping, respectively, with dropout rates of 0.1 and 0.2.

---

[1] `http://hedwig.ca`

[2] `https://github.com/lancopku/SGM`

#	Model	Reuters		AAPD		IMDB		Yelp '14	
		Val. F_1	Test F_1	Val. F_1	Test F_1	Val. Acc.	Test Acc.	Val. Acc.	Test Acc.
1	LR	77.0	74.8	67.1	64.9	43.1	43.4	61.1	60.9
2	SVM	89.1	86.1	71.1	69.1	42.5	42.4	59.7	59.6
3	KimCNN Repl.	83.5 ±0.4	80.8 ±0.3	54.5 ±1.4	51.4 ±1.3	42.9 ±0.3	42.7 ±0.4	66.5 ±0.1	66.1 ±0.6
4	KimCNN Orig.	–	–	–	–	–	37.6††	–	61.0††
5	XML-CNN Repl.	88.8 ±0.5	86.2 ±0.3	70.2 ±0.7	68.7 ±0.4	–	–	–	–
6	HAN Repl.	87.6 ±0.5	85.2 ±0.6	70.2 ±0.2	68.0 ±0.6	51.8 ±0.3	51.2 ±0.3	68.2 ±0.1	67.9 ±0.1
7	HAN Orig.	–	–	–	–	–	49.4‡	–	**70.5**‡
8	SGM Orig.	82.5 ±0.4	78.8 ±0.9	–	**71.0**†	–	–	–	–
9	LSTM$_{base}$	87.6 ±0.2	84.9 ±0.3	72.1 ±0.4	69.6 ±0.4	52.5 ±0.2	52.1 ±0.3	68.6 ±0.1	68.4 ±0.1
10	LSTM$_{reg}$	**89.1** ±0.8	**87.0** ±0.5	**73.1** ±0.4	70.5 ±0.5	**53.4** ±0.2	**52.8** ±0.3	**69.0** ±0.1	68.7 ±0.1

Table 2: Results for each model on the validation and test sets; best values are bolded in blue. *Repl.* reports mean ± SD of five runs from our reimplementations; *Orig.* refers to point estimates from †Yang et al. (2018), ‡Yang et al. (2016), and ††Tang et al. (2015).

For our optimization objective, we use cross-entropy and binary cross-entropy loss for single-label and multi-label tasks, respectively. On all datasets and models, we use 300-dimensional word vectors (Mikolov et al., 2013) pre-trained on Google News. We train all neural models for 30 epochs with five random seeds, reporting the mean validation set scores and their corresponding test set results.

Toward Robust Baselines. Recently, reproducibility is becoming a growing concern for the NLP community (Crane, 2018). Indeed, very few of the papers that we consider in this study report validation set results, let alone run on multiple seeds. In order to address these issues, we report scores on both validation and test sets for our reimplementations; doing so is good practice, since it reinforces the validity of the experimental results and claims. We also provide the standard deviation of the scores across different seeds to demonstrate the stability of our results. This is in line with previous papers (Zhang and Wallace, 2017; Reimers and Gurevych, 2017; Crane, 2018) that emphasize reporting variance for robustness against potentially spurious conclusions.

5 Results and Discussion

We report the mean and standard deviation (SD) of the F_1 scores and accuracy for all five runs in Table 2. For HAN and KimCNN, we include results from the original papers to validate our reimplementation. We fail to replicate the reported results of SGM on AAPD using the authors' codebase and data splits.[3] As a result, we simply copy the value reported in Yang et al. (2018) in Table 2, row 8, which represents their maximum F_1 score. To verify the correctness of our HAN and KimCNN reimplementations, we compare the differences in F_1 and accuracy on the appropriate datasets. We attribute the small differences to using different dataset splits (see Section 4.1) and reporting mean values.

Baseline Comparison. We see that our simple LSTM$_{reg}$ model achieves state of the art on Reuters and IMDB (see Table 2, rows 9 and 10), establishing mean scores of 87.0 and 52.8 for F_1 score and accuracy on the test sets of Reuters and IMDB, respectively. This highlights the efficacy of proper regularization and optimization techniques for the task. We observe that LSTM$_{reg}$ consistently improves upon the performance of LSTM$_{base}$ across all of the tasks—see rows 9 and 10, where, on average, regularization yields increases of 1.5 and 0.5 points for F_1 score and accuracy, respectively.

A few of our LSTM$_{reg}$ runs attain state-of-the-art test F_1 scores on AAPD. However, in the interest of robustness, we report the mean value, as mentioned in Section 4.2. We also find the accuracy of LSTM$_{reg}$ and our reimplemented version of HAN on Yelp 2014 to be almost two points lower than the copied result of HAN (rows 6, 7, and 10) from Yang et al. (2016). On the other hand, both of the models surpass the original result by nearly two points for the IMDB dataset. We cannot rule out that these disparities are caused

---

[3] The authors did not answer our e-mails seeking assistance.

by the absence of any widely-accepted splits for evaluation on Yelp 2014 and IMDB (as opposed to model or implementation differences).

Interestingly, the non-neural LR and SVM baselines perform remarkably well. On Reuters, for example, the SVM beats many neural baselines, including our non-regularized LSTM_{base} (rows 2–9). On AAPD, the SVM either ties or beats the other models, losing only to SGM (rows 2–8). Compared to the SVM, the LR baseline appears better suited for the single-label datasets IMDB and Yelp 2014, where it achieves better accuracy than the SVM does.

6 Conclusions and Future Work

In this paper, we question the complexity of existing neural network architectures for document classification. To demonstrate the effectiveness of proper regularization and optimization, we apply embedding dropout, weight dropping, and temporal averaging when training a simple BiLSTM model, establishing either competitive or state-of-the-art results on multiple datasets.

One potential extension of this work is to conduct a comprehensive ablation study to determine the relative contribution of each of the regularization and optimization techniques. Furthermore, it would be interesting to compare these techniques to the recent line of research in deep language representation models, such as Embeddings from Language Models (ELMo; Peters et al., 2018) and pre-trained transformers (Devlin et al., 2018; Radford, 2018). Finally, the examined regularization and optimization methods deserve exploration in other NLP tasks as well.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We thank Nabiha Asghar for providing us with the Yelp 2014 dataset. We also thank the anonymous reviewers for their valuable comments.

References

Chidanand Apté, Fred Damerau, and Sholom M Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

Matt Crane. 2018. Questionable answers in question answering research: reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Zachary C. Lipton and Jacob Steinhardt. 2018. Troubling trends in machine learning scholarship. *arXiv:1807.03341v2*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124.

Gbor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. 2018. Winner's curse? On pace, progress, and empirical rigor. In *Proceedings of the 6th International Conference on Learning Representations, Workshop Track (ICLR 2018)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv:1409.2329*.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 253–263.

Pre-trained Language Model Representations for Language Generation

Sergey Edunov[*], **Alexei Baevski**[*], **Michael Auli**
Facebook AI Research
Menlo Park, CA
`{edunov,abaevski,michaelauli}@fb.com`

Abstract

Pre-trained language model representations have been successful in a wide range of language understanding tasks. In this paper, we examine different strategies to integrate pre-trained representations into sequence to sequence models and apply it to neural machine translation and abstractive summarization. We find that pre-trained representations are most effective when added to the encoder network which slows inference by only 14%. Our experiments in machine translation show gains of up to 5.3 BLEU in a simulated resource-poor setup. While returns diminish with more labeled data, we still observe improvements when millions of sentence-pairs are available. Finally, on abstractive summarization we achieve a new state of the art on the full text version of CNN-DailyMail. [1]

1 Introduction

Pre-training of language models has been shown to provide large improvements for a range of language understanding tasks (Peters et al., 2018; Radford et al., 2018; Phang et al., 2018; Devlin et al., 2018). The key idea is to train a large generative model on vast corpora and use the resulting representations on tasks for which only limited amounts of labeled data is available. Pre-training of sequence to sequence models has been previously investigated for text classification (Dai and Le, 2015) but not for text generation. In neural machine translation, there has been work on transferring representations from high-resource language pairs to low-resource settings (Zoph et al., 2016).

In this paper, we apply pre-trained representations from language models to language generation tasks that can be modeled by sequence to sequence architectures. Previous work on integrating language models with sequence to sequence models focused on the decoder network and added language model representations right before the output of the decoder (Gulcehre et al., 2015). We extend their study by investigating several other strategies such as inputting ELMo-style representations (Peters et al., 2018) or fine-tuning the language model (§2).

Our experiments rely on strong transformer-based language models trained on up to six billion tokens (§3). We present a detailed study of various strategies in different simulated labeled training data scenarios and observe the largest improvements in low-resource settings but gains of over 1 BLEU are still possible when five million sentence-pairs are available. The most successful strategy to integrate pre-trained representations is as input to the encoder network (§4).

2 Strategies to add representations

We consider augmenting a standard sequence to sequence model with pre-trained representations following an ELMo-style regime (§2.1) as well as by fine-tuning the language model (§2.2).

2.1 ELMo augmentation

The ELMo approach of Peters et al. (2018) forms contextualized word embeddings based on language model representations without adjusting the actual language model parameters. Specifically, the ELMo module contains a set of parameters $\lambda_1 \dots \lambda_L, \gamma$ to form a linear combination of the L layers of the language model: ELMo = $\gamma \sum_{i=0}^{L} \frac{1}{Z} \exp(\lambda_i) \mathbf{h}^k$ where γ is a learned scalar, Z is a constant to normalize the $\exp(\lambda_i)$ to sum to one and $\mathbf{h}^k$ is the output of the k-th language model layer; the module also considers the input word embeddings of the language model. We also

---

[*] Equal contribution.

[1] Code and pre-trained models are available at `https://github.com/pytorch/fairseq/tree/bi_trans_lm/examples/pretraining`

Proceedings of NAACL-HLT 2019, pages 4052–4059
Minneapolis, Minnesota, June 2 - June 7, 2019.

apply layer normalization (Ba et al., 2016) to each $\mathbf{h}^k$ before computing ELMo vectors.

We experiment with an ELMo module to input contextualized embeddings either to the encoder (SRC-ELMO) or the decoder (TGT-ELMO). This provides word representations specific to the current input sentence and these representations have been trained on much more data than is available for the text generation task.

2.2 Fine-tuning approach

Fine-tuning the pre-trained representations adjusts the language model parameters by the learning signal of the end-task (Radford et al., 2018; Devlin et al., 2018). We replace learned input word embeddings in the encoder network with the output of the language model (SRC-FT). Specifically, we use the language model representation of the layer before the softmax and feed it to the encoder. We also add dropout to the language model output. Tuning separate learning rates for the language model and the sequence to sequence model may lead to better performance but we leave this to future work. However, we do tune the number of encoder blocks N as we found this important to obtain good accuracy for this setting. We apply the same strategy to the decoder: we input language model representations to the decoder network and fine-tune the language model when training the sequence to sequence model (TGT-FT).

3 Experimental setup

3.1 Datasets

Pre-training. We train language models on two languages: One model is estimated on the German newscrawl distributed by WMT'18 comprising 260M sentences or 6B tokens. Another model is trained on the English newscrawl data comprising 193M sentences or 5B tokens. We learn a joint Byte-Pair-Encoding (BPE; Sennrich et al., 2016) vocabulary of 37K types on the German and English newscrawl and train the language models with this vocabulary.

Machine translation. We consider two benchmarks: Most experiments are run on the WMT'18 English-German (en-de) news translation task and we validate our findings on the WMT'18 English-Turkish (en-tr) news task. For WMT'18 English-German, the training corpus consists of all available bitext excluding the ParaCrawl corpus and we remove sentences longer than 250 tokens as well as sentence-pairs with a source/target length ratio exceeding 1.5. This results in 5.18M sentence pairs. We tokenize all data with the Moses tokenizer (Koehn et al., 2007) and apply the BPE vocabulary learned on the monolingual corpora.

For WMT'18 English-Turkish, we use all of the available bitext comprising 208K sentence-pairs without any filtering. We develop on newstest2017 and test on newstest2018. For en-tr we only experiment with adding representations to the encoder and therefore apply the language model vocabulary to the source side. For the target vocabulary we learn a BPE code with 32K merge operations on the Turkish side of the bitext. Both datasets are evaluated in terms of case-sensitive de-tokenized BLEU (Papineni et al., 2002; Post, 2018).[2]

Summarization. We consider the CNN-DailyMail abstractive document summarization task comprising over 280K news articles paired with multi-sentence summaries. CNN-DailyMail is a widely used dataset for abstractive text summarization. Following (See et al., 2017), we report results on the non-anonymized version of CNN-DailyMail rather than the entity-anonymized version (Hermann et al., 2015; Nallapati et al., 2016) because the language model was trained on full text. Articles are truncated to 400 tokens (See et al., 2017) and we use a BPE vocabulary of 32K types (Fan et al., 2017). We evaluate in terms of F1-Rouge, that is Rouge-1, Rouge-2 and Rouge-L (Lin, 2004).[3]

3.2 Language model pre-training

We consider two types of architectures: a bi-directional language model to augment the sequence to sequence encoder and a uni-directional model to augment the decoder. Both use self-attention (Vaswani et al., 2017) and the uni-directional model contains $N = 12$ transformer blocks, followed by a word classifier to predict the next word on the right. The bi-directional model solves a cloze-style token prediction task at training time (Baevski et al., 2019). The model consists of two towers, the *forward* tower operates left-to-right and the tower operating right-to-left as *backward* tower; each tower contains $N = 12$ trans-

---

[2] sacreBLEU signatures: BLEU+case.mixed+lang.en-{de,tr}+numrefs.1+smooth.exp+test.wmt18+tok.13a +version.1.2.1

[3] We use the following parameters for `ROUGE-1.5.5.pl`: -m -a -n 2

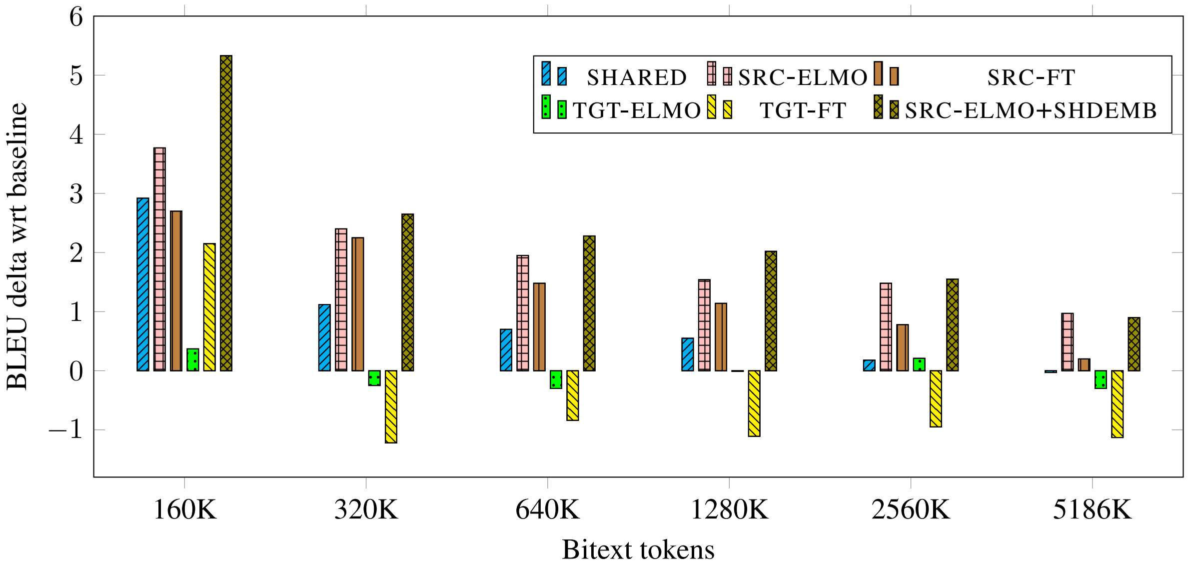


Figure 1: BLEU difference to a bitext-only baseline when adding pre-trained language model representations to a neural machine translation model in different simulated bitext settings. Results are based on averaging newstest2012-2017 of WMT English-German translation.

former blocks. The forward and backward representations are combined via a self-attention module and the output of this module is used to predict the token at position i. The model has access to the entire input surrounding the current target token. Models use the standard settings for the Big Transformer (Vaswani et al., 2017). The bi-directional model contains 353M parameters and the uni-directional model 190M parameters. Both models were trained for 1M steps using Nesterov's accelerated gradient (Sutskever et al., 2013) with momentum 0.99 following Baevski and Auli (2018). The learning rate is linearly warmed up from 10^{-7} to 1 for 16K steps and then annealed using a cosine learning rate schedule with a single phase to 0.0001 (Loshchilov and Hutter, 2016). We train on 32 Nvidia V100 SXM2 GPUs and use the NCCL2 library as well as the torch distributed package for inter-GPU communication. Training relies on 16-bit floating point operations (Ott et al., 2018) and it took six days for the bi-directional model and four days for the uni-directional model.

3.3 Sequence to sequence model

We use the transformer implementation of the fairseq toolkit (Ott et al., 2019). The WMT en-de and en-tr experiments are based on the Big Transformer sequence to sequence architecture with 6 blocks in the encoder and decoder. For abstractive summarization we use a base transformer model (Vaswani et al., 2017). We tune dropout values of between 0.1 and 0.4 on the validation set. Models are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e - 8$ and we use the same learning rate schedule as Vaswani et al. (2017); we perform 10K-200K depending on bitext size. All models use label smoothing with a uniform prior distribution over the vocabulary $\epsilon = 0.1$ (Szegedy et al., 2015; Pereyra et al., 2017). We run experiments on 8 GPUs and generate translations with a beam of size 5.

4 Results

4.1 Machine translation

We first present a comparison of the various strategies in different simulated parallel corpus size settings. For each experiment, we tune the dropout applied to the language model representations, and we reduce the number of optimizer steps for smaller bitext setups as models converge faster; all other hyper-parameters are equal between setups. Our baseline is a Big Transformer model and we also consider a variant where we share token embeddings between the encoder and decoder (SHARED; Inan et al., 2016; Press & Wolf, 2016).

Figure 1 shows results averaged over six test sets relative to the baseline which does not share source and target embeddings (Appendix A shows a detailed breakdown). SHARED performs very well with little labeled data but the gains erode to practically zero in large bitext settings. Pre-trained

	160K	640K	5186K
baseline	21.4	33.1	40.1
SRC-ELMO	26.6	35.6	41.8
SRC-FT	24.3	34.9	40.8
TGT-ELMO	21.3	31.9	40.5
TGT-FT	24.2	31.4	38.8
SRC-ELMO+SHDEMB	29.0	36.2	41.8

Table 1: BLEU on newstest2018 of WMT English-German in three simulated bitext size scenarios.

	news2017	news2018
baseline	9.8	9.5
SRC-ELMO	12.0	11.3
SRC-ELMO+SHDEMB	12.9	11.8

Table 2: WMT English-Turkish translation results in terms of BLEU on newstest2017 (valid) and newstest2018 (test) with ELMo inputs to the encoder.

language model representations are most effective in low bitext setups. The best performing strategy is ELMo embeddings input to the encoder (SRC-ELMO). This improves the baseline by 3.8 BLEU in the 160K bitext setting and it still improves the 5.2M setting by over 1 BLEU.

We further improve SRC-ELMO by sharing learned word representations in the decoder by tying input and output embeddings (SRC-ELMO+SHDEMB). This configuration performs even better than SRC-ELMO with a gain of 5.3 BLEU in the 160K setup. Sharing decoder embeddings is equally applicable to SRC-FT. Language model representations are much less effective in the decoder: TGT-FT improves the 160K bitext setup but yields no improvements thereafter and TGT-ELMO performs even worse. We conjecture that pre-trained representations give much easier wins in the encoder. Table 1 shows additional results on newstest2018.

Pre-trained representations mostly impacts the training time of the sequence to sequence model (see Appendix B): SRC-ELMO slows throughput during training by about 5.3x and SRC-FT is even slower because of the need to backpropagate through the LM for fine-tuning (9.2x). However, inference is only 12-14% slower than the baseline when adding pre-trained embeddings to the encoder (SRC-ELMO, SRC-FT). This is because the LM computation can be paralelized for all input tokens. Inference is much slower when adding representations to the decoder because the LM needs to be invoked repeatedly. Our current implementation does not cache LM operations for the previous state and can be made much faster.

	ROUGE		
	1	2	L
Lead-3	40.34	17.70	36.57
See et al. (2017)	39.53	17.28	36.38
Gehrmann et al. (2018)	41.22	18.68	38.34
baseline	40.07	17.61	36.78
SRC-ELMO+SHDEMB	41.56	18.94	38.47

Table 3: Abstractive summarization results on CNN-DailyMail. ELMo inputs achieve a new state of the art.

The baseline uses a BPE vocabulary estimated on the language model corpora (§3). Appendix A shows that this vocabulary actually leads to sligtly better performance than a joint BPE code learned on the bitext as is usual.

Next, we validate our findings on the WMT'18 English-Turkish task for which the bitext is truly limited (208K sentence-pairs). We use the language model vocab for the the English side of the bitext and a BPE vocabulary learned on the Turkish side. Table 2 shows that ELMo embeddings for the encoder improve English-Turkish translation.

4.2 Abstractive summarization

Following See et al. (2017), we experiment on the non-anonymized version of CNN-DailyMail. When generating summaries, we follow standard practice of tuning the maximum output length and disallow repeating the same trigram (Paulus et al., 2017; Fan et al., 2017). For this task we train language model representations on the combination of newscrawl and the CNN-DailyMail training data. Table 3 shows that pre-trained embeddings can significantly improve on top of a strong baseline transformer. We also compare to Gehrmann et al. (2018) who use a task-specific architecture compared to our generic sequence to sequence baseline. Pre-trained representations are complementary to their method.

5 Conclusion

We presented an analysis of different strategies to add pre-trained language model representations to sequence to sequence models for neural machine translation and abstractive document summarization. Adding pre-trained representations is very effective for the encoder network and while returns diminish when more labeled data becomes available, we still observe improvements when millions of examples are available. In future research we will investigate ways to improve the decoder with pre-trained representations.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv*, abs/1607.06450.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv*, abs/1809.10853.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *arXiv*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. *arXiv*, abs/1511.01432.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv*, abs/1711.05217.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv*, abs/1808.10792.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv*, abs/1503.03535.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. of NIPS*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv*, abs/1611.01462.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL Demo Session*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out*.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *arXiv*, abs/1608.03983.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proc. of CONLL*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. of WMT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv*, abs/1705.04304.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *International Conference on Learning Representations (ICLR) Workshop*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of ACL*.

Jason Phang, Thibault Fevry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv*, abs/1811.01088.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv*, abs/1804.08771.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proc. of EACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proc. of ICML*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv preprint arXiv:1512.00567*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proc. of NIPS*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proc. of EMNLP*.

A Detailed WMT English-German Results

bitext	method	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**Avg**
160K	baseline	13.2	15.7	13.5	15.7	18.6	14.8	21.4	16.1
	SHARED	15.3	18.2	16.7	19.0	21.6	18.2	24.9	19.1
	SHARED+bitext-BPE	15.1	17.9	16.2	18.9	22.0	18.0	25.2	19.0
	SRC-ELMO	16.0	19.4	17.1	19.9	23.0	18.7	26.6	20.1
	SRC-FT	15.3	18.5	16.6	18.9	20.8	17.6	24.3	18.9
	TGT-ELMO	13.3	16.4	14.1	16.2	18.8	14.9	21.3	16.4
	TGT-FT	14.7	17.2	15.8	18.4	21.4	16.9	24.2	18.4
	SRC-ELMO+SHDEMB	17.4	20.8	18.6	21.5	24.9	20.3	29.0	**21.8**
320K	baseline	17.2	20.4	18.1	21.2	25.0	19.6	28.9	21.5
	SHARED	18.1	21.1	19.1	22.4	26.3	21.2	30.6	22.7
	SHARED+bitext-BPE	17.6	20.6	19.1	22.3	26.1	20.8	29.9	22.3
	src-elmo	18.8	22.3	21.1	24.0	27.5	22.2	32.5	24.1
	SRC-FT	19.0	22.5	20.9	23.5	26.9	22.2	32.1	23.9
	TGT-ELMO	16.7	20.7	18.2	20.9	24.1	19.4	28.0	21.1
	TGT-FT	16.1	19.4	17.1	20.0	23.1	18.5	26.3	20.1
	SRC-ELMO+SHDEMB	19.5	22.9	21.2	24.0	27.4	22.4	32.3	**24.2**
640K	baseline	19.2	22.9	21.2	24.5	27.9	22.4	33.1	24.5
	SHARED	19.9	23.4	22.1	25.1	28.8	23.0	34.1	25.2
	SHARED+bitext-BPE	19.4	22.8	21.7	24.9	28.4	22.9	33.6	24.8
	src-elmo	21.0	24.3	23.4	26.5	30.0	24.6	35.6	26.5
	SRC-FT	20.5	24.0	22.9	26.1	29.1	24.4	34.9	26.0
	TGT-ELMO	18.9	22.6	20.8	24.2	27.5	22.3	31.9	24.0
	TGT-FT	18.2	21.8	20.6	23.7	27.0	21.8	31.4	23.5
	SRC-ELMO+SHDEMB	21.2	25.1	23.9	26.7	30.2	24.7	36.2	**26.9**
1280K	baseline	20.9	24.6	23.6	26.5	30.5	24.7	36.2	26.7
	SHARED	21.1	24.6	24.6	27.6	31.0	25.2	37.3	27.3
	SHARED+bitext-BPE	20.5	24.0	23.9	26.2	30.6	24.8	36.2	26.6
	src-elmo	22.1	25.7	25.7	28.5	31.7	26.3	38.2	28.3
	SRC-FT	21.3	25.2	25.3	28.5	31.1	26.2	37.4	27.9
	TGT-ELMO	20.9	24.4	23.6	26.6	30.3	24.9	36.1	26.7
	TGT-FT	20.1	23.7	22.4	25.2	29.1	23.6	34.4	25.5
	SRC-ELMO+SHDEMB	22.3	26.0	26.3	28.9	32.6	26.8	38.6	**28.8**
2560K	baseline	21.7	25.6	25.4	28.2	32.3	26.2	39.1	28.4
	SHARED	22.2	25.9	25.7	28.3	32.1	26.3	38.9	28.5
	SHARED+bitext-BPE	21.8	25.5	25.5	27.9	32.1	26.0	38.6	28.2
	src-elmo	22.9	27.0	27.0	30.0	33.4	28.0	40.0	**29.8**
	SRC-FT	22.2	26.4	26.3	29.5	32.4	27.3	39.3	29.1
	TGT-ELMO	21.8	25.7	25.8	28.5	32.3	26.6	39.3	28.6
	TGT-FT	21.5	25.3	24.5	27.0	30.2	25.2	36.8	27.2
	SRC-ELMO+SHDEMB	23.1	27.2	27.1	29.7	33.7	27.9	40.0	**29.8**
5186K	baseline	23.1	26.8	27.7	30.1	33.6	27.9	40.1	29.9
	SHARED	22.6	26.6	27.7	30.5	33.4	28.2	40.2	29.9
	SHARED+bitext-BPE	22.5	26.0	27.0	29.7	33.4	27.7	40.6	29.6
	src-elmo	23.7	27.8	28.7	31.1	34.5	29.2	41.8	**31.0**
	SRC-FT	23.1	27.0	27.8	30.5	33.7	28.3	40.8	30.2
	TGT-ELMO	22.9	26.6	26.9	29.5	33.8	27.7	40.5	29.7
	TGT-FT	22.3	26.1	26.1	28.9	32.5	26.5	38.8	28.7
	SRC-ELMO+SHDEMB	23.4	28.0	28.8	31.2	34.5	28.7	41.8	30.9

Table 4: BLEU on newstest2012 to newstest2018 of WMT English-German translation in varioius simulated bitext size scenarios (cf. Figure 1).

B Training and inference speed

	train (tok/sec)	inference (tok/sec)
SHARED	528,802	2,334
SRC-ELMO	100,636	2,011
SRC-FT	57,753	2,080
TGT-ELMO	142,525	259
TGT-FT	95,313	299

Table 5: Training and inference speed of models trained on WMT English-German. Training speed based on 32 V100 GPUs. Inference speed measured on a single V100 and by batching up to 12K source or target tokens.

Pragmatically Informative Text Generation

Sheng Shen† **Daniel Fried**† **Jacob Andreas**‡ **Dan Klein**†
†Computer Science Division, UC Berkeley
‡Computer Science and Artificial Intelligence Laboratory, MIT
`{sheng.s,dfried,klein}@berkeley.edu, jda@mit.edu`

Abstract

We improve the informativeness of models for conditional text generation using techniques from computational pragmatics. These techniques formulate language production as a game between speakers and listeners, in which a speaker should generate output text that a listener can use to correctly identify the original input that the text describes. While such approaches are widely used in cognitive science and grounded language learning, they have received less attention for more standard language generation tasks. We consider two pragmatic modeling methods for text generation: one where pragmatics is imposed by information preservation, and another where pragmatics is imposed by explicit modeling of distractors. We find that these methods improve the performance of strong existing systems for abstractive summarization and generation from structured meaning representations.

1 Introduction

Computational approaches to pragmatics cast language generation and interpretation as game-theoretic or Bayesian inference procedures (Golland et al., 2010; Frank and Goodman, 2012). While such approaches are capable of modeling a variety of pragmatic phenomena, their main application in natural language processing has been to improve the *informativeness* of generated text in grounded language learning problems (Monroe et al., 2018). In this paper, we show that pragmatic reasoning can be similarly used to improve performance in more traditional language generation tasks like generation from structured meaning representations (Figure 1) and summarization.

Our work builds on a line of learned Rational Speech Acts (RSA) models (Monroe and Potts, 2015; Andreas and Klein, 2016), in which generated strings are selected to optimize the behavior of an embedded listener model. The canonical presentation of the RSA framework (Frank and Goodman, 2012) is grounded in reference resolution: models of speakers attempt to describe referents in the presence of distractors, and models of listeners attempt to resolve descriptors to referents. Recent work has extended these models to more complex groundings, including images (Mao et al., 2015) and trajectories (Fried et al., 2018). The techniques used in these settings are similar, and the primary intuition of the RSA framework is preserved: from the speaker's perspective, a good description is one that picks out, as discriminatively as possible, the content the speaker intends for the listener to identify.

Input meaning representation (i):
Name[Fitzbillies], EatType[coffee shop], Food[English], PriceRange[cheap], CustomerRating[5 out of 5], Area[Riverside], FamilyFriendly[yes]

Human-written
A cheap coffee shop in riverside with a 5 out of 5 customer rating is Fitzbillies. Fitzbillies is family friendly and serves English food.

Base sequence-to-sequence model (S_0)
Fitzbillies is a family friendly coffee shop located near the river.

Distractor-based pragmatic system (S_1^D)
Fitzbillies is a family friendly coffee shop that serves English food. It is located in riverside area. It has a customer rating of 5 out of 5 and is cheap.

Reconstructor-based pragmatic system (S_1^R)
Fitzbillies is a family friendly coffee shop that serves cheap English food in the riverside area. It has a customer rating of 5 out of 5.

Figure 1: Example outputs of our systems on the E2E generation task. While a base sequence-to-sequence model (S_0, Sec. 2) fails to describe all attributes in the input meaning representation, both of our pragmatic systems (S_1^R, Sec. 3.1 and S_1^D, Sec. 3.2) and the human-written reference do.

Outside of grounding, cognitive modeling (Frank et al., 2009), and targeted analysis of linguistic phenomena (Orita et al., 2015), rational speech acts models have seen limited application in the natural language processing literature. In this work we show that they can be extended

Proceedings of NAACL-HLT 2019, pages 4060–4067
Minneapolis, Minnesota, June 2 - June 7, 2019.

to a distinct class of language generation problems that use as referents structured descriptions of lingustic content, or other natural language texts. In accordance with the maxim of quantity (Grice, 1970) or the Q-principle (Horn, 1984), pragmatic approaches naturally correct underinformativeness problems observed in state-of-the-art language generation systems (S_0 in Figure 1).

We present experiments on two language generation tasks: generation from meaning representations (Novikova et al., 2017) and summarization. For each task, we evaluate two models of pragmatics: the reconstructor-based model of Fried et al. (2018) and the distractor-based model of Cohn-Gordon et al. (2018). Both models improve performance on both tasks, increasing ROUGE scores by 0.2–0.5 points on the CNN/Daily Mail abstractive summarization dataset and BLEU scores by 2 points on the End-to-End (E2E) generation dataset, obtaining new state-of-the-art results.

2 Tasks

We formulate a conditional generation task as taking an input i from a space of possible inputs $\mathcal{I}$ (e.g., input sentences for abstractive summarization; meaning representations for structured generation) and producing an output o as a sequence of tokens $(o_1, \ldots, o_T)$. We build our pragmatic approaches on top of learned *base speaker* models S_0, which produce a probability distribution $S_0(o \mid i)$ over output text for a given input. We focus on two conditional generation tasks where the information in the input context should largely be preserved in the output text, and apply the pragmatic procedures outlined in Sec. 3 to each task. For these S_0 models we use systems from past work that are strong, but may still be underinformative relative to human reference outputs (e.g., Figure 1).

Meaning Representations Our first task is generation from structured meaning representations (MRs) containing attribute-value pairs (Novikova et al., 2017). An example is shown in Figure 1, where systems must generate a description of the restaurant with the specified attributes. We apply pragmatics to encourage output strings from which the input MR can be identified. For our S_0 model, we use a publicly-released neural generation system (Puzikov and Gurevych, 2018) that achieves comparable performance to the best published results in Dušek et al. (2018).

Abstractive Summarization Our second task is multi-sentence document summarization. There is a vast amount of past work on summarization (Nenkova and McKeown, 2011); recent neural models have used large datasets (e.g., Hermann et al. (2015)) to train models in both the extractive (Cheng and Lapata, 2016; Nallapati et al., 2017) and abstractive (Rush et al., 2015; See et al., 2017) settings. Among these works, we build on the recent abstractive neural summarization system of Chen and Bansal (2018). First, this system uses a sentence-level extractive model RNN-EXT to identify a sequence of salient sentences $i^{(1)}, \ldots i^{(P)}$ in each source document. Second, the system uses an abstractive model ABS to rewrite each $i^{(p)}$ into an output $o^{(p)}$, which are then concatenated to produce the final summary. We rely on the fixed RNN-EXT model to extract sentences as inputs in our pragmatic procedure, using ABS as our S_0 model and applying pragmatics to the $i^{(p)} \rightarrow o^{(p)}$ abstractive step.

3 Pragmatic Models

To produce informative outputs, we consider pragmatic methods that extend the *base speaker* models, S_0, using *listener* models, L, which produce a distribution $L(i \mid o)$ over possible inputs given an output. Listener models are used to derive *pragmatic speakers*, $S_1(o \mid i)$, which produce output that has a high probability of making a listener model L identify the correct input. There are a large space of possible choices for designing L and deriving S_1; we follow two lines of past work which we categorize as *reconstructor-based* and *distractor-based*. We tailor each of these pragmatic methods to both our two tasks by developing reconstructor models and methods of choosing distractors.

3.1 Reconstructor-Based Pragmatics

Pragmatic approaches in this category (Dušek and Jurčíček, 2016; Fried et al., 2018) rely on a *reconstructor listener* model L^R defined independently of the speaker. This listener model produces a distribution $L^R(i \mid o)$ over all possible input contexts $i \in \mathcal{I}$, given an output description o. We use sequence-to-sequence or structured classification models for L^R (described below), and train these models on the same data used to supervise the S_0 models.

The listener model and the base speaker model together define a *pragmatic speaker*, with output score given by:

$$S_1^R(o \mid i) = L^R(i \mid o)^\lambda \cdot S_0(o \mid i)^{1-\lambda} \quad (1)$$

where λ is a *rationality parameter* that controls how much the model optimizes for discriminative outputs (see Monroe et al. (2017) and Fried et al. (2018) for a discussion). We select an output text sequence o for a given input i by choosing the highest scoring output under Eq. 1 from a set of candidates obtained by beam search in $S_0(\cdot \mid i)$.

Meaning Representations We construct L^R for the meaning representation generation task as a multi-task, multi-class classifier, defining a distribution over possible values for each attribute. Each MR attribute has its own prediction layer and attention-based aggregation layer, which conditions on a basic encoding of o shared across all attributes. See Appendix A.1 for architecture details. We then define $L^R(i \mid o)$ as the joint probability of predicting all input MR attributes in i from o.

Summarization To construct L^R for summarization, we train an ABS model (of the type we use for S_0, Chen and Bansal (2018)) but in reverse, i.e., taking as input a sentence in the summary and producing a sentence in the source document. We train L^R on the same heuristically-extracted and aligned source document sentences used to train S_0 (Chen and Bansal, 2018).

3.2 Distractor-Based Pragmatics

Pragmatic approaches in this category (Frank and Goodman, 2012; Andreas and Klein, 2016; Vedantam et al., 2017; Cohn-Gordon et al., 2018) derive pragmatic behavior by producing outputs that distinguish the input i from an alternate *distractor* input (or inputs). We construct a distractor $\widetilde{\imath}$ for a given input i in a task-dependent way.[1]

We follow the approach of Cohn-Gordon et al. (2018), outlined briefly here. The base speakers we build on produce outputs incrementally, where the probability of o_t, the word output at time t, is conditioned on the input and the previously generated words: $S_0(o_t \mid i, o_{<t})$. Since the output is generated incrementally and there is no separate listener model that needs to condition on entire output decisions, the distractor-based approach is able to make pragmatic decisions at each word rather than choosing between entire output candidates (as in the reconstructor approaches).

The listener L^D and pragmatic speaker S_1^D are derived from the base speaker S_0 and a belief distribution $p_t(\cdot)$ maintained at each timestep t over the possible inputs $\mathcal{I}^D$:

$$L^D(i \mid o_{<t}) \propto S_0(o_{<t} \mid i) \cdot p_{t-1}(i) \quad (2)$$

$$S_1^D(o_t \mid i, o_{<t}) \propto L^D(i \mid o_{<t})^\alpha \cdot S_0(o_t \mid i, o_{<t}) \quad (3)$$

$$p_t(i) \propto S_0(o_t \mid i, o_{<t}) \cdot L^D(i \mid o_{<t}) \quad (4)$$

where α is again a rationality parameter, and the initial belief distribution $p_0(\cdot)$ is uniform, i.e., $p_0(i) = p_0(\widetilde{\imath}) = 0.5$. Eqs. 2 and 4 are normalized over the true input i and distractor $\widetilde{\imath}$; Eq. 3 is normalized over the output vocabulary. We construct an output text sequence for the pragmatic speaker S_1^D incrementally using beam search to approximately maximize Eq. 3.

Meaning Representations A distractor MR is automatically constructed for each input to be the most distinctive possible against the input. We construct this distractor by masking each present input attribute and replacing the value of each non-present attribute with the value that is most frequent for that attribute in the training data. For example, for the input MR in Figure 1, the distractor is NEAR[BURGER KING].

Summarization For each extracted input sentence $i^{(p)}$, we use the previous extracted sentence $i^{(p-1)}$ from the same document as the distractor input $\widetilde{\imath}$ (for the first sentence we do not use a distractor). This is intended to encourage outputs $o^{(p)}$ to contain distinctive information against other summaries produced within the same document.

4 Experiments

For each of our two conditional generation tasks we evaluate on a standard benchmark dataset, following past work by using automatic evaluation against human-produced reference text. We choose hyperparameters for our models (beam size, and parameters α and λ) to maximize task metrics on each dataset's development set; see Appendix A.2 for the settings used.[2]

---

[1] In tasks such as contrastive captioning or referring expression generation, these distractors are given; for the conditional generation task, we will show that pragmatic behavior can be obtained by constructing or selecting a single distractor that contrasts with the input i.

[2] Our code is publicly available at `https://github.com/sIncerass/prag_generation`.

System	BLEU	NIST	METEOR	R-L	CIDEr
T-Gen	65.93	8.61	44.83	68.50	2.23
Best Prev.	66.19†	8.61†	**45.29**‡	**70.83**$^\diamond$	2.27$^\bullet$
S_0	66.52	8.55	44.45	69.34	2.23
S_0 ×2	65.93	8.31	43.52	69.58	2.12
S_1^R	**68.60**	**8.73**	**45.25**	**70.82**	**2.37**
S_1^D	67.76	8.72	44.59	69.41	2.27

Table 1: Test results for the E2E generation task, in comparison to the T-Gen baseline (Dušek and Jurčíček, 2016) and the best results from the E2E challenge, reported by Dušek et al. (2018): †Juraska et al. (2018), ‡Puzikov and Gurevych (2018), $^\diamond$Zhang et al. (2018), and $^\bullet$Gong (2018). We bold our highest performing model on each metric, as well as previous work if it outperforms all of our models.

4.1 Meaning Representations

We evaluate on the E2E task of generation from meaning representations containing restaurant attributes (Novikova et al., 2017). We report the task's five automatic metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin, 2004) and CIDEr (Vedantam et al., 2015).

Table 1 compares the performance of our base S_0 and pragmatic models to the baseline T-Gen system (Dušek and Jurčíček, 2016) and the *best previous* result from the 20 primary systems evaluated in the E2E challenge (Dušek et al., 2018). The systems obtaining these results encompass a range of approaches: a template system (Puzikov and Gurevych, 2018), a neural model (Zhang et al., 2018), models trained with reinforcement learning (Gong, 2018), and systems using ensembling and reranking (Juraska et al., 2018). To ensure that the benefit of the reconstructor-based pragmatic approach, which uses two models, is not due solely to a model combination effect, we also compare to an ensemble of two base models (S_0 ×2). This ensemble uses a weighted combination of scores of two independently-trained S_0 models, following Eq. 1 (with weights tuned on the development data).

Both of our pragmatic systems improve over the strong baseline S_0 system on all five metrics, with the largest improvements (2.1 BLEU, 0.2 NIST, 0.8 METEOR, 1.5 ROUGE-L, and 0.1 CIDEr) from the S_1^R model. This S_1^R model outperforms the previous best results obtained by any system in the E2E challenge on BLEU, NIST, and CIDEr, with comparable performance on METEOR and ROUGE-L.

System	R-1	R-2	R-L	METEOR
Extractive				
Lead-3	40.34	17.70	36.57	**22.21**
Inputs	38.93	18.23	35.90	**24.66**
Abstractive				
Best Previous	**41.69**†	**19.47**†	**39.08**‡	21.00$^\diamond$
S_0	40.88	17.80	38.54	20.38
S_0 ×2	40.76	17.88	38.46	19.88
S_1^R	41.23	18.07	38.76	20.57
S_1^D	**41.39**	**18.30**	**38.78**	**21.70**

Table 2: Test results for the non-anonymized CNN/Daily Mail summarization task. We compare to extractive baselines, and the best previous abstractive results of †Celikyilmaz et al. (2018), ‡Paulus et al. (2018) and $^\diamond$Chen and Bansal (2018). We bold our highest performing model on each metric, as well as previous work if it outperforms all of our models.

4.2 Abstractive Summarization

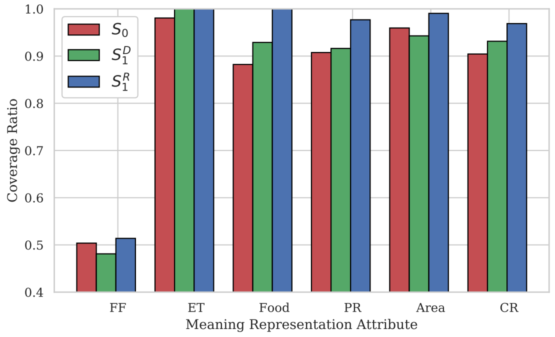
We evaluate on the CNN/Daily Mail summarization dataset (Hermann et al., 2015; Nallapati et al., 2016), using See et al.'s (2017) non-anonymized preprocessing. As in previous work (Chen and Bansal, 2018), we evaluate using ROUGE and METEOR.

Table 2 compares our pragmatic systems to the base S_0 model (with scores taken from Chen and Bansal (2018); we obtained comparable performance in our reproduction[3]), an ensemble of two of these base models, and the *best previous* abstractive summarization result for each metric on this dataset (Celikyilmaz et al., 2018; Paulus et al., 2018; Chen and Bansal, 2018). We also report two extractive baselines: *Lead-3*, which uses the first three sentences of the document as the summary (See et al., 2017), and *Inputs*, the concatenation of the extracted sentences used as inputs to our models (i.e., $i^{(1)}, \dots, i^{(P)}$).

The pragmatic methods obtain improvements of 0.2–0.5 in ROUGE scores and 0.2–1.8 METEOR over the base S_0 model, with the distractor-based approach S_1^D outperforming the reconstructor-based approach S_1^R. S_1^D is strong across all metrics, obtaining results competitive to the best previous abstractive systems.

---

[3] We use retrained versions of Chen and Bansal (2018)'s sentence extractor and abstractive S_0 models in all our experiments, as well as their n-gram reranking-based inference procedure, replacing scores from the base model S_0 with scores from S_1^R or S_1^D in the respective pragmatic procedures.



(a) Coverage ratios by attribute type for the base model S_0 and pragmatic models S_1^R and S_1^D. The pragmatic models typically improve coverage ratios across attribute types when compared to the base model.

		Coverage Ratio for Attribute					
		FF	ET	Food	PR	Area	CR
	S_0	0.50	0.98	0.88	0.91	0.96	0.90
Distractor Attr.	S_1^D-FF	0.57	1.00	0.92	0.90	0.95	0.95
	S_1^D-ET	0.47	1.00	0.96	0.92	0.96	0.95
	S_1^D-Food	0.45	1.00	1.00	0.93	0.95	0.94
	S_1^D-PR	0.51	1.00	0.90	0.98	0.93	0.92
	S_1^D-Area	0.47	1.00	0.93	0.91	0.98	0.93
	S_1^D-CR	0.45	1.00	0.91	0.90	0.91	0.95

(b) Coverage ratios by attribute type (columns) for the base model S_0, and for the pragmatic system S_1^D when constructing the distractor by masking the specified attribute (rows). Cell colors are the degree the coverage ratio increases (green) or decreases (red) relative to S_0.

Figure 2: Coverage ratios for the E2E task by attribute type, estimating how frequently the values for each attribute from the input meaning representations are mentioned in the output text.

5 Analysis

The base speaker S_0 model is often underinformative, e.g., for the E2E task failing to mention certain attributes of a MR, even though almost all the training examples incorporate all of them. To better understand the performance improvements from the pragmatic models for E2E, we compute a *coverage ratio* as a proxy measure of how well content in the input is preserved in the generated outputs. The coverage ratio for each attribute is the fraction of times there is an exact match between the text in the generated output and the attribute's value in the source MR (for instances where the attribute is specified).[4]

Figure 2(a) shows coverage ratio by attribute category for all models. The S_1^R model increases the coverage ratio when compared to S_0 across all attributes, showing that using the reconstruction model score to select outputs does lead to an increase in mentions for each attribute. Coverage ratios increase for S_1^D as well in four out of six categories, but the increase is typically less than that produced by S_1^R.

While S_1^D optimizes less explicitly for attribute mentions than S_1^R, it still provides a potential method to control generated outputs by choosing alternate distractors. Figure 2(b) shows coverage ratios for S_1^D when masking only a single attribute in the distractor. The highest coverage ratio for each attribute is usually obtained when masking that attribute in the distractor MR (entries on the main diagonal, underlined), in particular for FAMILYFRIENDLY (FF), FOOD, PRICERANGE (PR), and AREA. However, masking a single attribute sometimes results in decreasing the coverage ratio, and we also observe substantial increases from masking other attributes: e.g., masking either FAMILYFRIENDLY or CUSTOMERRATING (CR) produces an equal increase in coverage ratio for the CUSTOMERRATING attribute. This may reflect underlying correlations in the training data, as these two attributes have a small number of possible values (3 and 7, respectively).

6 Conclusion

Our results show that S_0 models from previous work, while strong, still imperfectly capture the behavior that people exhibit when generating text; and an explicit pragmatic modeling procedure can improve results. Both pragmatic methods evaluated in this paper encourage prediction of outputs that can be used to identify their inputs, either by reconstructing inputs in their entirety or distinguishing true inputs from distractors, so it is perhaps unsurprising that both methods produce similar improvements in performance. Future work might allow finer-grained modeling of the tradeoff between *under*- and *over*-informativity within the sequence generation pipeline (e.g., with a learned communication cost model) or explore applications of pragmatics for content selection earlier in the generation pipeline.

Acknowledgments

Thanks to Reuben Cohn-Gordon for many helpful discussions and suggestions. This work was supported by DARPA through the XAI program. DF is supported by a Tencent AI Lab Fellowship.

---

[4] Note that this measure roughly provides a lower bound on the model's actual informativeness for each attribute, since the measure does not assign credit for paraphrases.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Reuben Cohn-Gordon, Noah Goodman, and Chris Potts. 2018. Pragmatically informative image captioning with character-level reference. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram cooccurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Ondrej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *ACL*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the International Conference on Natural Language Generation*.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Michael C Frank, Noah D Goodman, Peter Lai, and Joshua B Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the 31st annual conference of the cognitive science society*, pages 1228–1233.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing*, pages 410–419. Association for Computational Linguistics.

Heng Gong. 2018. Technical report for E2E NLG challenge. In *E2E NLG Challenge System Descriptions*.

Herbert P Grice. 1970. Logic and conversation.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Laurence Horn. 1984. Toward a new taxonomy for pragmatic inference: Q-based and r-based implicature. *Meaning, form, and use in context: Linguistic applications*, 11:42.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 152–162, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *arXiv preprint arXiv:1511.02283*.

Will Monroe, Robert XD Hawkins, Noah D Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*.

Will Monroe, Jennifer Hu, Andrew Jong, and Christopher Potts. 2018. Generating bilingual pragmatic color references. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Will Monroe and Christopher Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*, Amsterdam. ILLC.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Meeting of the Association for the Advancement of Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue*.

Naho Orita, Eliana Vornov, Naomi Feldman, and Hal Daumé III. 2015. Why discourse affects speakers' choice of referring expressions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1639–1649.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Proceedings of the International Conference on Learning Representations*, volume abs/1705.04304.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2E NLG challenge: Neural models vs. templates. In *Proceedings of the International Conference on Natural Language Generation*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Biao Zhang, Jing Yang, Qian Lin, and Jinsong Su. 2018. Attention regularized sequence-to-sequence learning for E2E NLG challenge. In *E2E NLG Challenge System Descriptions*.

A Supplemental Material

A.1 Reconstructor Model Details

For the reconstructor-based speaker in the E2E task, we first follow the same data preprocessing steps as Puzikov and Gurevych (2018), which includes a delexicalization module that deals with sparsely occurring MR attributes (NAME, NEAR) by mapping such values to placeholder tokens.

MRs have only a few possible values for most attributes: six out of eight attributes have fewer than seven unique values, and the remaining two attributes (NAME, NEAR) are handled by our S_0 and S_1^D using delexicalized placeholders, following Puzikov and Gurevych (2018). In this way, the reconstructor only needs to predict the presence of these two attributes with a boolean variable, and other attributes with the corresponding categorical variable. We use a one layer bi-directional GRU (Cho et al., 2014) for the shared sentence encoder. We concatenate the latent vectors from both directions to construct a bi-directional encoded vector h_i for every single word vector d_i as:

$$\overrightarrow{h_i} = \overrightarrow{GRU}(d_i, h_{i-1}), \overleftarrow{h_i} = \overleftarrow{GRU}(d_i, h_{i+1})$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}], i \in [1, L]$$

Since not all words contribute equally to predicting each MR attribute, we thus use an attention mechanism (Bahdanau et al., 2014) to determine the importance of every single word. The aggregated sentence vector for task k is calculated by

$$a_i^{(k)} = \frac{exp(W_a^{(k)} h_i)}{\sum_{j=1}^{L} exp(W_a^{(k)} h_j)}, v^{(k)} = \sum_{i=1}^{L} a_i^{(k)} h_i,$$

The task-specific sentence representation is then used as input to k layers with softmax outputs, returning a probability vector $Y^{(k)}$ for each of the k MR attributes.

A.2 Hyperparameters

For structured generation, we use beam size 10, $\lambda = 0.4$, and $\alpha = 0.2$, tuned to maximize the normalized average of all five metrics on the development set.

For abstractive summarization, we use beam size 20, $\lambda = 0.9$, and $\alpha = 1.0$, tuned to maximize ROUGE-L on the development set.

Stochastic Wasserstein Autoencoder for Probabilistic Sentence Generation

Hareesh Bahuleyan,[†] **Lili Mou,**[†] **Hao Zhou,**[‡] **Olga Vechtomova**[†]
[†]University of Waterloo [‡]ByteDance AI Lab
`hpallika@uwaterloo.ca` `doublepower.mou@gmail.com`
`zhouhao.nlp@bytedance.com` `ovechtomova@uwaterloo.ca`

Abstract

The variational autoencoder (VAE) imposes a probabilistic distribution (typically Gaussian) on the latent space and penalizes the Kullback–Leibler (KL) divergence between the posterior and prior. In NLP, VAEs are extremely difficult to train due to the problem of KL collapsing to zero. One has to implement various heuristics such as KL weight annealing and word dropout in a carefully engineered manner to successfully train a VAE for text. In this paper, we propose to use the Wasserstein autoencoder (WAE) for probabilistic sentence generation, where the encoder could be either stochastic or deterministic. We show theoretically and empirically that, in the original WAE, the stochastically encoded Gaussian distribution tends to become a Dirac-delta function, and we propose a variant of WAE that encourages the stochasticity of the encoder. Experimental results show that the latent space learned by WAE exhibits properties of continuity and smoothness as in VAEs, while simultaneously achieving much higher BLEU scores for sentence reconstruction.[1]

1 Introduction

Natural language sentence generation in the deep learning regime typically uses a recurrent neural network (RNN) to predict the most probable next word given previous words (Mikolov et al., 2010). Such RNN architecture can be further conditioned on some source information, for example, an input sentence, resulting in a sequence-to-sequence (Seq2Seq) model.

Traditionally, sentence generation is accomplished in a *deterministic* fashion, i.e., the model uses a deterministic neural network to encode an input sentence to some hidden representations, from which it then decodes an output sentence using another deterministic neural network.

Bowman et al. (2016) propose to use the variational autoencoder (VAE, Kingma and Welling, 2014) to map an input sentence to a probabilistic continuous latent space. VAE makes it possible to generate sentences from a distribution, which is desired in various applications. For example, in an open-domain dialog system, the information of an utterance and its response is not necessarily a one-to-one mapping, and multiple plausible responses could be suitable for a given input. Probabilistic sentence generation makes the dialog system more diversified and more meaningful (Serban et al., 2017; Bahuleyan et al., 2018). Besides, probabilistic modeling of the hidden representations serves as a way of posterior regularization (Zhang et al., 2016), facilitating interpolation (Bowman et al., 2016) and manipulation of the latent representation (Hu et al., 2017).

However, training VAEs in NLP is more difficult than the image domain (Kingma and Welling, 2014). The VAE training involves a reconstruction loss and a Kullback–Leibler (KL) divergence between the posterior and prior of the latent space. In NLP, the KL term tends to vanish to zero during training, leading to an ineffective latent space. Previous work has proposed various engineering tricks to alleviate this problem, including KL annealing and word dropout (Bowman et al., 2016).

In this paper, we address the difficulty of training VAE sentence generators by using a Wasserstein autoencoder (WAE, Tolstikhin et al., 2018). WAE modifies VAE in that it requires the integration of the posterior to be close to its prior, where the closeness is measured with empirical samples drawn from the distributions. In this way, the encoder could be either stochastic or deterministic, but the model still retains probabilistic properties.

---

[1] Our code is availabe at `https://github.com/HareeshBahuleyan/probabilistic_nlg`
A preliminary version of this paper was preprinted at `https://arxiv.org/abs/1806.08462`

Proceedings of NAACL-HLT 2019, pages 4068–4076
Minneapolis, Minnesota, June 2 - June 7, 2019.

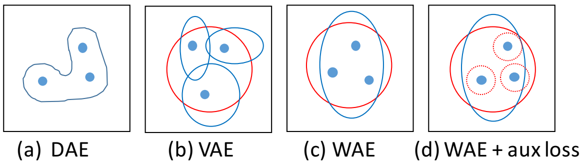


Figure 1: The latent space of the deterministic autoencoder (DAE), variational autoencoder (VAE), Wasserstein autoencoder (WAE), as well as WAE with by our KL penalty. Blue circles: Posterior or aggregated posterior distributions of data in the latent space. Red circles: Regularizations of the posterior.

Moreover, we show both theoretically and empirically that the stochastic Gaussian encoder in the original form tends to be a Dirac-delta function. We thus propose a WAE variant that encourages the encoder's stochasticity by penalizing an auxiliary KL term.

Experiments show that the sentences generated by WAE exhibit properties of continuity and smoothness as in VAE, while achieving a much higher reconstruction performance. Our proposed variant further encourages the stochasticity of the encoder. More importantly, WAE is robust to hyperparameters and much easier to train, without the need for KL annealing or word dropout as in VAE. In a dialog system, we demonstrate that WAEs are capable of generating better quality and more diverse sentences than VAE.

2 Probabilistic Sentence Generation

Base Model: Deterministic Autoencoder (DAE). DAE encodes an input sentence with a recurrent neural network (RNN) and then decodes the same sentence through another RNN.

For the encoder, the hidden state of the last word is represented as the latent space of the input sentence $\mathbf{x}$. The latent representation is denoted as $\boldsymbol{z}$. We feed $\boldsymbol{z}$ to the decoder RNN, which predicts one word at a time using a softmax layer, given by $p(\mathrm{x}_t|\boldsymbol{z}, \mathbf{x}_{<t})$.The training objective for DAE is the sequence-aggregated cross-entropy loss, given by

$$J = -\sum_{n=1}^{N}\sum_{t=1}^{|\mathbf{x}^{(n)}|}\log p(\mathrm{x}_t^{(n)}|\boldsymbol{z}^{(n)}, \mathbf{x}_{<t}^{(n)}) \qquad (1)$$

where superscript (n) indicates the nth data point among $1, \cdots, N$.

In DAE, the latent space is encoded and then decoded in a deterministic way, i.e., there is no probabilistic modeling of the hidden space. The hidden representations of data may be located on an arbitrary manifold (Figure 1a), which is not suitable for probabilistic generation.

Variational Autoencoder (VAE). VAE extends DAE by imposing a prior distribution $p(\boldsymbol{z})$ on the latent variable $\boldsymbol{z}$, which is typically set to the standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (Kingma and Welling, 2014). Given an input sentence $\mathbf{x}$, we would like to model the posterior of $\boldsymbol{z}$ by another normal distribution, $q(\boldsymbol{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \operatorname{diag}\boldsymbol{\sigma}^2_{\text{post}})$, where $\boldsymbol{\mu}_{\text{post}}$ and $\boldsymbol{\sigma}^2_{\text{post}}$ are the outputs of the encoder.

In the training of VAE, $\boldsymbol{z}$ is sampled from $q(\boldsymbol{z}|\mathbf{x})$, and the training objective is to maximize a variational lower bound of the likelihood of data. This is equivalent to minimizing the (expected) reconstruction loss similar to (1), while being regularized by the KL divergence between $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$, given by

$$J = \sum_{n=1}^{N}\Big[-\underset{\boldsymbol{z}^{(n)}\sim q}{\mathbb{E}}\sum_{t=1}^{|\mathbf{x}^{(n)}|}\log p(\mathrm{x}_t^{(n)}|\boldsymbol{z}^{(n)}, \mathbf{x}_{<t}^{(n)}) + \lambda_{\text{VAE}}\cdot \mathrm{KL}(q(\boldsymbol{z}^{(n)}|\mathbf{x}^{(n)})\|p(\boldsymbol{z}))\Big] \qquad (2)$$

where in the expectation $\boldsymbol{z}^{(n)}$ is sampled from $q(\boldsymbol{z}|\mathbf{x}^{(n)})$ and λ_{VAE} is a hyperparameter balancing the two terms.

Since VAE penalizes the divergence of $\boldsymbol{z}$'s posterior from its prior, it serves as a way of posterior regularization, making it possible to generate sentences from the continuous latent space.

However, the two objectives in (2) are contradictory to each other, as argued by Tolstikhin et al. (2018). VAE pushes the posterior of $\boldsymbol{z}$, given any input $\mathbf{x}^{(n)}$, to be close to its prior, i.e., every blue ellipse in Figure 1b should be close to the red one. This makes perfect reconstruction impossible.

Further, VAE is difficult to train in NLP due to the problem of KL collapse, where the KL term tends to be zero, meaning that the encoder captures no information and the decoder learns an unconditioned language model. This phenomenon is observed in variational auto-regressive decoders using RNN. To alleviate this problem, existing tricks include KL annealing and word dropout (Bowman et al., 2016), but both require extensive engineering.

Wasserstein Autoencoder (WAE). An alternative way of posterior regularization is to impose a constraint that the *aggregated posterior* of $\boldsymbol{z}$ should be the same as its prior (Tolstikhin et al., 2018), i.e., $q(\boldsymbol{z}) \overset{\text{def}}{=} \sum_{\mathbf{x}} q(\boldsymbol{z}|\mathbf{x})p_{\mathcal{D}}(\mathbf{x}) \overset{\text{set}}{=} p(\boldsymbol{z})$, where $p_{\mathcal{D}}$ is the data distribution. This is also demonstrated in Figure 1c. By contrast, VAE requires that $q(\boldsymbol{z}|\mathbf{x})$ should be close to $p(\boldsymbol{z})$ for every input sentence $\mathbf{x}$.

For computational purposes, Tolstikhin et al. (2018) relax the above constraint by penalizing the Wasserstein distance between $q(\boldsymbol{z})$ and $p(\boldsymbol{z})$. In particular, it is computed by the Maximum Mean Discrepancy (MMD), defined as

$$\mathrm{MMD} = \left\| \int k(\boldsymbol{z},\cdot)\,\mathrm{d}P(\boldsymbol{z}) - \int k(\boldsymbol{z},\cdot)\,\mathrm{d}Q(\boldsymbol{z}) \right\|_{\mathcal{H}_k}$$

where $P(\boldsymbol{z})$ and $Q(\boldsymbol{z})$ are cumulative density functions. $\mathcal{H}_k$ refers to the reproducing kernel Hilbert space defined by the kernel k, which is often chosen as the inverse multiquadratic kernel $k(x,y) = \frac{C}{C+\|x-y\|_2^2}$ for high-dimensional Gaussians.

One advantage of the Wasserstein distance is that it can be estimated by empirical samples as

$$\widehat{\mathrm{MMD}} = \frac{1}{N(N-1)}\sum_{n\neq m} k(\boldsymbol{z}^{(n)},\boldsymbol{z}^{(m)}) + \frac{1}{N(N-1)}\sum_{n\neq m} k(\widetilde{\boldsymbol{z}}^{(n)},\widetilde{\boldsymbol{z}}^{(m)}) - \frac{1}{N^2}\sum_{n,m} k(\boldsymbol{z}^{(n)},\widetilde{\boldsymbol{z}}^{(m)}) \tag{3}$$

where $\widetilde{\boldsymbol{z}}^{(n)}$ is a sample from the prior $p(\boldsymbol{z})$, and $\boldsymbol{z}^{(n)}$ is a sample from the aggregated posterior $q(\boldsymbol{z})$, which is obtained by sampling $\mathbf{x}^{(n)}$ from the data distribution and then sampling $\boldsymbol{z}^{(n)}$ from $q(\boldsymbol{z}|\mathbf{x}^{(n)})$. In summary, the training objective of WAE is

$$J_{\mathrm{WAE}} = -\sum_{n=1}^{N}\sum_{t=1}^{|\mathbf{x}^{(n)}|}\log p(\mathrm{x}_t^{(n)}|\boldsymbol{z}^{(n)},\mathbf{x}_{<t}^{(n)}) + \lambda_{\mathrm{WAE}}\cdot\widehat{\mathrm{MMD}} \tag{4}$$

where λ_{WAE} balances the MMD penalty and the reconstruction loss.

Alternatively, the dual form (adversarial loss) can also be used for WAE (Zhao et al., 2018). In our preliminary experiments, we found MMD similar to but slightly better than the adversarial loss. The difference between our work and Zhao et al. (2018)—who extend the original WAE to sequence generation—is that we address the KL annealing problem of VAE and further analyze the stochasticity of WAE from a theoretical perspective, as follows.

WAE with Auxiliary Loss. In WAE, the aggregated posterior $q(\boldsymbol{z})$ involves an integration of data distribution, which allows using a deterministic function to encode $\boldsymbol{z}$ as $\boldsymbol{z} = \boldsymbol{f}_{\text{encode}}(\mathbf{x})$ as suggested by Tolstikhin et al. (2018). This would largely alleviate the training difficulties as in VAE, because backpropagating gradient into the encoder no longer involves a stochastic layer.

The stochasticity of the encoder, however, is still a desired property in some applications, for example, generating diverse responses in a dialog system. We show both theoretically and empirically that a *dangling* Gaussian stochastic encoder could possibly degrade to a deterministic one.

Theorem 1. *Suppose we have a Gaussian family $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}\boldsymbol{\sigma}^2)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are parameters. The covariance is diagonal, meaning that the variables are independent. If the gradient of $\boldsymbol{\sigma}$ completely comes from sample gradient and $\boldsymbol{\sigma}$ is small at the beginning of training, then the Gaussian converges to a Dirac delta function with stochastic gradient descent, i.e., $\boldsymbol{\sigma} \to \mathbf{0}$. (See Appendix A for the proof.)* ☐

To alleviate this problem, we propose a simple heuristic that encourages the stochasticity of the encoder. In particular, we penalize, for every data point, a KL term between the predicted posterior $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \operatorname{diag}\boldsymbol{\sigma}_{\text{post}}^2)$ and a Gaussian with covariance $\mathbf{I}$ centered at the predicted mean, i.e., $\mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \mathbf{I})$. This is shown in Figure 1d, where each posterior is encouraged to stretch with covariance $\mathbf{I}$. Formally, the loss is

$$J = J_{\text{rec}} + \lambda_{\text{WAE}}\cdot\widehat{\mathrm{MMD}} + \lambda_{\text{KL}}\sum\nolimits_n \mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)}, \operatorname{diag}(\boldsymbol{\sigma}_{\text{post}}^{(n)})^2)\|\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)}, \mathbf{I})\right) \tag{5}$$

While our approach appears heuristic, the next theorem shows its theoretical justification.

Theorem 2. *Objective (5) is a relaxed optimization of the WAE loss (4) with a constraint on $\boldsymbol{\sigma}_{\text{post}}$. (See Appendix B for the proof.)* ☐

We will show empirically that such auxiliary loss enables us to generate smoother and more diverse sentences in WAE. It, however, does not suffer from KL collapse as in VAEs. The auxiliary KL loss that we define for stochastic WAE is computed against a target distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)}, \mathbf{I})$ for each data sample $\mathbf{x}^{(n)}$. Here, the predicted posterior mean itself is used in the target distribution.

	BLEU$^{\uparrow}$	**PPL**$^{\downarrow}$	**UniKL**$^{\downarrow}$	**Entropy**	**AvgLen**
Corpus	-	-	-	$\rightarrow 5.65$	$\rightarrow 9.6$
DAE	**86.35**	146.2	0.178	6.23	11.0
VAE (KL-annealed)	43.18	79.4	0.081	5.04	8.8
WAE-D $\lambda_{\text{WAE}}=3$	86.03	113.8	0.071	5.59	10.0
WAE-D $\lambda_{\text{WAE}}=10$	84.29	104.9	0.073	5.57	9.9
WAE-S $\lambda_{\text{KL}}=0.0$	75.66	115.2	0.069	**5.61**	9.9
WAE-S $\lambda_{\text{KL}}=0.01$	82.01	84.9	**0.058**	5.26	**9.4**
WAE-S $\lambda_{\text{KL}}=0.1$	47.63	**62.5**	0.150	4.65	8.7

Table 1: Results of SNLI-style sentence generation, where WAE is compared with DAE and VAE. **D** and **S** refer to the deterministic and stochastic encoders, respectively. $^{\uparrow/\downarrow}$The larger/lower, the better. For **Entropy** and **AvgLen**, the closer to corpus statistics, the better (indicated by the $\rightarrow$ arrow).

As a result, this KL term does not force the model to learn the same posterior for all data samples (as in VAE), and thus, the decoder does not degrade to an unconditioned language model.

3 Experiments

We evaluate WAE in sentence generation on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) as well as dialog response generation. All models use single-layer RNN with long short term memory (LSTM) units for both the encoder and decoder. Appendix C details our experimental settings.

VAE training. VAE is notoriously difficult to train in the RNN setting. While different researchers have their own practice of training VAE, we follow our previous experience (Bahuleyan et al., 2018) and adopt the following tricks to stabilize the training: (1) λ_{VAE} was annealed in a $\mathrm{sigmoid}$ manner. We monitored the value of $\lambda \cdot \text{KL}$ and stop annealing once it reached its peak value, known as *peaking annealing*. (2) For word dropout, we started with no dropout, and gradually increased the dropout rate by 0.05 every epoch until it reached a value of 0.5. The effect of KL annealing is further analyzed in Appendix D.

3.1 SNLI Generation

The SNLI sentences are written by crowdsourcing human workers in an image captioning task. It is a massive corpus but with comparatively simple sentences (examples shown in Table 4). This task could be thought of as domain-specific sentence generation, analogous to hand written digit generation in computer vision.

In Table 1, we compare all methods in two aspects. (1) We evaluate by **BLEU** (Papineni et al., 2002) how an autoencoder preserves input information in a reconstruction task. (2) We also evaluate the quality of probabilistic sentence generation from the latent space. Although there is no probabilistic modeling of the latent space in DAE, we nevertheless draw samples from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, which could serve as a non-informative prior. Perplexity (**PPL**) evaluates how fluent the generated sentences are. This is given by a third-party n-gram language model trained on the Wikipedia dataset. The unigram-KL (**UniKL**) evaluates if the word distribution of the generated sentences is close to that of the training corpus. Other surface metrics (entropy of the word distribution and average sentence length) also measure the similarity of the latent space generated sentence set to that of the corpus.

We see that DAE achieves the best BLEU score, which is not surprising because DAE directly optimizes the maximum likelihood of data as a surrogate of word prediction accuracy. Consequently, DAE performs poorly for probabilistic sentence generation as indicated by the other metrics.

VAE and WAE have additional penalties that depart from the goal of reconstruction. However, we see that WAEs, when trained with appropriate hyperparameters ($\lambda_{\text{WAE}}, \lambda_{\text{KL}}$), achieve close performance to DAE, outperforming VAE by 40 BLEU points. This is because VAE encodes each input's posterior to be close to the prior, from which it is impossible to perfectly reconstruct the data.

Comparing the deterministic and stochastic encoders in WAE, we observe the same trade-off between reconstruction and sampling. However, our proposed stochastic encoder, with $\lambda_{\text{KL}} = 0.1$ for WAE, consistently outperforms VAE in the contradictory metrics BLEU and PPL. The hyperparameters $\lambda_{\text{WAE}} = 10.0$ and $\lambda_{\text{KL}} = 0.01$ appear to have the best balance between reconstruction, sentence fluency, as well as similarity to the original corpus.

Moreover, all our WAEs are trained without annealing or word dropout. It is significantly simpler than training a VAE, whose KL annealing typically involves a number of engineering tricks, such as the time step when KL is included, the slope of annealing, and the stopping criterion for annealing.

	BLEU-2	BLEU-4	Entropy	Dist-1	Dist-2
Test Set	-	-	6.15	0.077	0.414
DED	3.96	0.85	5.55	0.044	0.275
VED	3.26	0.59	5.45	0.053	0.204
WED-D	**4.05**	**0.98**	5.53	0.042	0.272
WED-S	3.72	0.69	**5.59**	**0.066**	**0.309**

Table 2: Results on dialog generation, where VED/WED hyperparameters for each model were chosen by Table 1.

3.2 Dialog Generation

We extend WAE to an encoder-decoder framework (denoted by WED) and evaluate it on the DailyDialog corpus (Li et al., 2017).[2] We follow Bahuleyan et al. (2018), using the encoder to capture an utterance and the decoder to generate a reply.

Table 2 shows that WED with a deterministic encoder (WED-D) is better than the variational encoder-decoder (VED) in BLEU scores, but the generated sentences lack variety, which is measured by output entropy and the percentage of distinct unigrams and bigrams (Dist-1/Dist-2, Li et al., 2016), evaluated on the generated test set responses.

We then applied our stochastic encoder for WED and see that, equipped with our KL-penalized stochastic encoder, WED-S outperforms DED, VED, and WED-D in all diversity measures. WED-S also outperforms VED in generation quality, consistent with the results in Table 1.

4 Conclusion

In this paper, we address the difficulty of training VAE by using a Wasserstein autoencoder (WAE) for probabilistic sentence generation. WAE implementation can be carried out with either a deterministic encoder or a stochastic one. The deterministic version achieves high reconstruction performance, but lacks diversity for generation. The stochastic encoder in the original form may collapse to a Dirac delta function, shown by both a theorem and empirical results. We thus propose to encourage stochasticity by penalizing a heuristic KL loss for WAE, which turns out to be a relaxed optimization of the Wasserstein distance with a constraint on the posterior family.

We evaluated our model on both SNLI sentence generation and dialog systems. We see that WAE achieves high reconstruction performance as DAE, while retaining the probabilistic property as VAE. Our KL-penalty further improves the stochasticity of WAE, as we achieve the highest performance in all diversity measures.

Acknowledgments

We would like to acknowledge Yiping Song and Zhiliang Tian for their independent investigation on the DailyDialog corpus. We also thank Yanran Li, one of the authors who released DailyDialog, for discussion on this issue. This work was supported in part by the NSERC grant RGPIN-261439-2013 and an Amazon Research Award.

References

Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2018. Variational attention for sequence-to-sequence models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1672–1682.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2017. Understanding disentangling in beta-VAE. In *Proceedings of the Workshop on Learning Disentangled Representations: From Perception to Control*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the*

---

[2] In our pilot experiment, we obtained a BLEU-4 score of 6 by training a pure Seq2Seq model with LSTM units for 200 epochs, whereas Li et al. (2017) report 0.009 BLEU-4 and Luo et al. (2018) report 2.84 BLEU-4. Due to our unreasonably high performance, we investigated this in depth and found that the training and test sets of the DailyDialog corpus have overlaps. For the results reported in our paper, we have removed duplicate data in the test set, which is also available on our website (Footnote 1). To the best of our knowledge, we are the first to figure out the problem, which, unfortunately, makes comparison with previous work impossible.

34th International Conference on Machine Learning, pages 1587–1596.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of International Conference on Learning Representations*, pages 10–21.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: a manually labelled multi-turn dialogue dataset. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pages 986–995.

Liangchen Luo, Jingjing Xu, Junyang Lin, Qi Zeng, and Xu Sun. 2018. An auto-encoder matching model for learning utterance-level semantic dependency in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 702–707.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, pages 1045–1048.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3295–3301.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scholkopf. 2018. Wasserstein auto-encoders. In *Proceedings of International Conference on Learning Representations*.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the International Conference on Machine Learning*, pages 3881–3890.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5902–5911.

A Proof of Theorem 1

Theorem 1. *Suppose we have a Gaussian family* $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag} \boldsymbol{\sigma}^2)$*, where* $\boldsymbol{\mu}$ *and* $\boldsymbol{\sigma}$ *are parameters. The covariance is diagonal, meaning that the variables are independent. If the gradient of* $\boldsymbol{\sigma}$ *completely comes from sample gradient and* $\boldsymbol{\sigma}$ *is small at the beginning of training, then the Gaussian converges to a Dirac delta function with stochastic gradient descent, i.e.,* $\boldsymbol{\sigma} \to \mathbf{0}$.

Proof. For the predicted posterior $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag} \boldsymbol{\sigma}^2)$ where all dimensions are independent, we consider a certain dimension, where the sample is $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

We denote the gradient of J wrt to z_i at $z_i = \mu_i$ by $g_i \triangleq \frac{\partial J}{\partial z_i}\big|_{z_i=\mu_i}$. At a particular sample $z_i^{(j)}$ around μ_i, the gradient $g_i^{(j)}$ is

$$
\begin{aligned}
g_i^{(j)} &\triangleq \left.\frac{\partial J}{\partial z_i}\right|_{z_i=z_i^{(j)}} && (6)\\
&\approx g_i + \frac{\partial^2 J}{\partial \mu_i^2}(z_i^{(j)} - \mu_i) && (7)\\
&\triangleq g_i + k(z_i^{(j)} - \mu_i) && (8)\\
&= g_i + k(\mu_i + \epsilon^{(j)}\sigma_i - \mu_i) && (9)\\
&= g_i + k\sigma_i\epsilon^{(j)} && (10)
\end{aligned}
$$

where (7) is due to Taylor series approximation, if we assume σ_i^2 is small and thus $z_i^{(j)}$ is near μ_i. k denotes $\frac{\partial^2 J}{\partial \mu_i^2}$.

We compute the expected gradient wrt to σ_i for $\epsilon \sim \mathcal{N}(0, 1)$. The assumption of this theorem is that the gradient of μ_i and σ_i completely comes from the sample z_i. By the chain rule, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\epsilon^{(j)}\sim\mathcal{N}(0,1)}\left[\frac{\partial J}{\partial \sigma_i}\right] && (11)\\
=&\mathop{\mathbb{E}}_{\epsilon^{(j)}\sim\mathcal{N}(0,1)}\left[\frac{\partial J}{\partial z_i^{(j)}}\cdot\frac{\partial z_i^{(j)}}{\partial \sigma_i}\right] && (12)\\
=&\mathop{\mathbb{E}}_{\epsilon^{(j)}\sim\mathcal{N}(0,1)}\left[(g_i+k\sigma_i\epsilon^{(j)})\cdot\epsilon^{(j)}\right] && (13)\\
=&\mathop{\mathbb{E}}_{\epsilon^{(j)}\sim\mathcal{N}(0,1)}\left[g_i\epsilon_i^{(j)}+k\sigma_i(\epsilon^{(j)})^2\right] && (14)\\
=&k\sigma_i && (15)
\end{aligned}
$$

Notice that $k > 0$ if we are near a local optimum (locally convex).

In other words, the expected gradient of σ_i is proportional to σ_i. According to stochastic gradient descent (SGD), σ_i will converge to zero. ☐

The theorem assumes $\boldsymbol{\sigma}^2$ is small, compared with how J changes in the latent space. In practice, the encoded vectors of different samples may vary a lot, whereas if we sample different vectors from a certain predicted multi-variate Gaussian, we would generally obtain the same sentence. Therefore, J is kind of smooth in the latent space. The phenomenon can also be verified empirically by plotting the histogram of σ in WAE with a stochastic Gaussian encoder (Figure 2). We see that if the KL coefficient λ_{KL} is 0, meaning that the gradient of σ comes only from the samples, then most σ's collapse to 0.

Notice, however, that the theorem does not suggest a stochastic WAE and a deterministic WAE will yield exactly the same result, as their trajectories may be different.

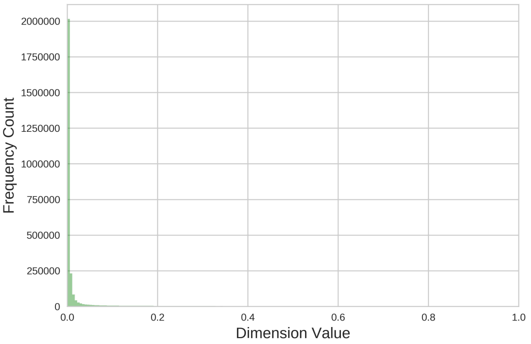
B Proof of Theorem 2

Theorem 2. *Objective (5) is a relaxed optimization of the WAE loss (4) with a constraint on* $\boldsymbol{\sigma}_{\text{post}}$.

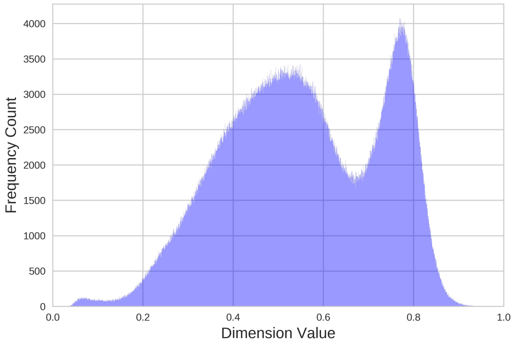
Proof. Objective (5) optimizes

$$
\begin{aligned}
J = &J_{\text{rec}} + \lambda_{\text{WAE}}\cdot\widehat{\text{MMD}}\\
&+\lambda_{\text{KL}}\sum\nolimits_n \text{KL}\left(\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)},\text{diag}(\boldsymbol{\sigma}_{\text{post}}^{(n)})^2)\|\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)},\mathbf{I})\right)
\end{aligned}
$$

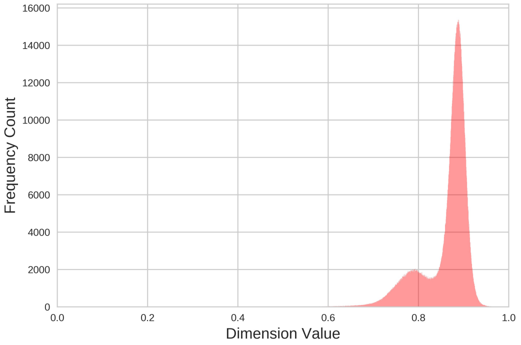
The first two terms are the WAE loss, whereas the last penalty relaxes the following optimization problem



(a) $\lambda_{\text{KL}} = 0$



(b) $\lambda_{\text{KL}} = 0.01$



(c) $\lambda_{\text{KL}} = 0.1$

Figure 2: The histograms of σ in the posterior of the WAE in the SNLI experiment. In the plot, there are 200 buckets in the range of $(0, 1)$.

$$
\begin{aligned}
&\text{minimize}\ \ J_{\text{rec}} + \lambda_{\text{WAE}}\cdot\widehat{\text{MMD}}\\
&\text{subject to}\\
&\sum\nolimits_n \text{KL}\left(\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)},\text{diag}(\boldsymbol{\sigma}_{\text{post}}^{(n)2}))\|\mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(n)},\mathbf{I})\right) < C
\end{aligned}
\tag{16}
$$

for some constant C.

As known, the KL divergence between two

SNLI Experiment

LSTM Hidden Dimension	100d, single layer
Word Embeddings	300d, pretrained on SNLI Corpus
Latent Dimension	100d
Epochs	20
Learning Rate	Fixed rate of 0.001
Batch Size	128
Max Sequence Length	20
Vocab Size	30000

Dialog Experiment

LSTM Hidden Dimension	500d, single layer
Word Embeddings	300d, pretrained on DialyDialog Corpus
Latent Dimension	300d
Epochs	200
Learning Rate	Initial rate of 0.001, multiplicative decay of 0.98 until a minimum of 0.00001
Batch Size	128
Max Sequence Length	20
Vocab Size	20000

Table 3: Experimental settings.

(univariant) Gaussian distributions is

$$\begin{aligned} &\mathrm{KL}(\mathcal{N}(\mu_1,\sigma_1)\|(\mathcal{N}(\mu_2,\sigma_2))) \\ &= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned} \tag{17}$$

The constraint in (16) is equivalent to

$$\sum_n\sum_i\left[-\log\sigma_i^{(n)} + \frac{1}{2}(\sigma_i^{(n)})^2\right] < C \tag{18}$$

In other words, our seemingly heuristic KL penalty optimizes the Wasserstein loss, while restricting the posterior family. □

C Implementation Details

All models were trained with the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In all our experiments, we feed the sampled latent vector $\boldsymbol{z}$ to each time step of the decoder. Task-specific settings are listed in Table 3.

D VAE Training Difficulties

It is a common practice that training VAEs involves KL annealing and word dropout, which further consists of hacks for tuning hyperparameters. We conducted an experiment of training VAE without KL annealing. In Figure 3, we present the KL loss (weighted by λ_{VAE}) during the training process for different values of λ_{VAE}. The KL loss is believed to be an important diagnostic measure to indicate if the latent space is "variational" (Yang et al., 2017; Higgins et al., 2017; Burgess et al., 2017). We see that if the penalty is too large, KL simply collapses to zero ignoring the entire input, in the case of which, the decoder becomes an unconditioned language model. On the other hand, if the KL penalty is too small, the model tends to become more *deterministic* and the KL term does not play a role in the training. This is expected since in the limit of λ_{VAE} to 0, the model would probably ignore the KL term and becomes a deterministic autoencoder (shown also by Theorem 1).

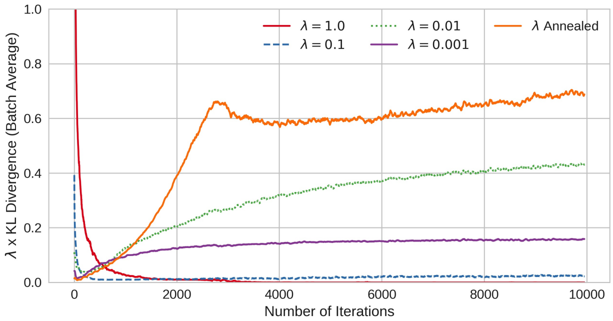


Figure 3: Learning curves of the KL term in the VAE loss function ($\lambda \cdot \mathrm{KL}$) for different values of λ, and the variant where λ is annealed.

The VAE with collapsed KL does not exhibit interesting properties such as random sampling for probabilistic sequence generation (Bowman et al., 2016). As seen in Table 4, the generated sentences by VAE without annealing are very close to each other. This is because VAE's encoder does not capture useful information in the latent space, which is simply ignored during the decoding phase. By sampling the latent space, we do not obtain varying sentences. The empirical evidence verifies our intuition.

A recent study (Xu and Durrett, 2018) propose to get rid of KL annealing by using the von Mises–Fisher (vMF) family of posterior and prior. In particular, they set the prior to the uniform distribution on a unit hypersphere, whereas the posterior family is normal distribution on the surface of the same sphere. They fix the standard deviation (parametrized by κ) of the posterior, so that their KL is a constant and annealing is not required. This, unfortunately, loses the privilege of learning uncertainty in the probabilistic modeling. Examples in Table 4 show that, while we have reproduced the reconstruction negative log-likelihood with vMF-VAE (the metric used in their paper), the generated sentences are of poor quality. As also suggested by Xu and Durrett (2018), if the posterior uncertainty in vMF is made learnable, it re-introduces the KL collapse problem, in which

Training Samples
a mother and her child are outdoors.
the people are opening presents.
the girls are looking toward the water.
a small boy walks down a wooden path in the woods.
a person in a green jacket it surfing while holding on to a line.
DAE
two families walking in a towel down alaska sands a cot .
a blade is rolling its nose furiously paper .
a woman in blue shirts is passing by a some beach
transporting his child are wearing overalls .
a guys are blowing on professional thinks the horse .
VAE without Annealing
a man is playing a guitar .
a man is playing with a dog .
a man is playing with a dog .
a man is playing a guitar .
a man is playing with a dog .
VAE with Annealing
the band is sitting on the main street .
couple dance on stage in a crowded room .
two people run alone in an empty field .
the group of people have gathered in a picture .
a cruise ship is docking a boat ship .
VAE vMF (κ fixed)
a car is a and and a blue shirt top is .
two children are playing on the group in are the the . the
a child and a adult and
the young is playing for a picture a are playing to
a little is playing a background . .
WAE-D ($\lambda_{\text{WAE}} = 10$)
the lone man is working .
the group of men is using ice at the sunset .
a family is outside in the background .
two women are standing on a busy street outside a fair
a tourists is having fun on a sunny day
WAE-S ($\lambda_{\text{WAE}} = 10$, $\lambda_{\text{KL}} = 0.01$)
an asian man is dancing in a highland house .
a person wearing a purple snowsuit jumps over the tree .
the vocalist is at the music and dancing with a microphone .
a young man is dressed in a white shirt cleaning clothes .
three children lie together and a woman falls in a plane .

Table 4: Sentences generated by randomly sampling from the prior for different models.

case, the KL annealing is still needed.

By contrast, WAEs for sequence-to-sequence models are trained without any additional optimization strategies such as annealing. Even in our stochastic encoder, the KL penalty does not make WAE an unconditioned language model, because it does not force the encoded posterior to be the same for different input sentences.

E Qualitative Samples

Table 4 shows sentences generated by randomly sampling points in the latent space for different models, along with sample sentences from the training set. They provide a qualitative understanding of each model's performance.

Benchmarking Hierarchical Script Knowledge

Yonatan Bisk[1] **Jan Buys**[1] **Karl Pichotta**[*] **Yejin Choi**[1,2]
[1]Paul G. Allen School of Computer Science and Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
{ybisk, jbuys}@cs.washington.edu

Abstract

Understanding procedural language requires reasoning about both hierarchical and temporal relations between events. For example, "boiling pasta" is a *sub-event* of "making a pasta dish", typically happens *before* "draining pasta," and *requires* the use of omitted tools (e.g. a strainer, sink...). While people are able to choose when and how to use abstract versus concrete instructions, the NLP community lacks corpora and tasks for evaluating if our models can do the same. In this paper, we introduce **KIDSCOOK**, a parallel script corpus, as well as a cloze task which matches video captions with missing procedural details. Experimental results show that state-of-the-art models struggle at this task, which requires inducing functional commonsense knowledge not explicitly stated in text.

1 Introduction

The level of detail used in natural language communication varies: descriptive or instructive text for experts may elide over details the reader can seamlessly infer, while text for more novice audiences may be more verbose. A given document typically adheres to a single level of verbosity suited to its presumed audience (Grice, 1975), so learning correspondences between abstract and detailed descriptions of similar concepts from text is a challenging problem.

Commonsense knowledge of how complex events decompose into stereotypical sequences of simpler events is a necessary component of a system that can automatically understand and reason about different types of discourse. Hierarchical correspondences between abstract and detailed representations of concepts and events were an important aspect of the original formulation of scripts for natural language understanding (Schank and Abelson, 1977; DeJong, 1981) but required hand-written data structures encoding world knowledge. However, the automatic induction of such commonsense knowledge from open-domain noisy text corpora remains an open problem (Chambers, 2013; Weber et al., 2018; Zellers et al., 2018). As a step towards solving this problem we consider textual descriptions of actions in a cooking domain.

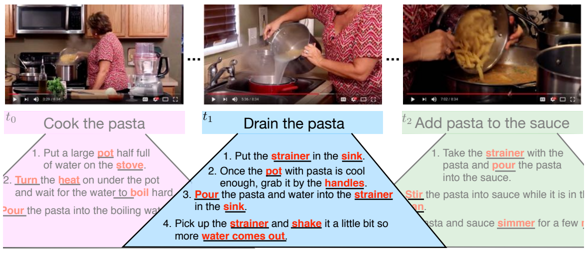


Figure 1: An example **KIDSCOOK** sequence with multiple types of hierarchy and abstraction: the example contains sequences of complex instructions, given both as sentences and sequences of simpler instructions.

We introduce a dataset, **KIDSCOOK**, targeted at exploring the automatic acquisition of correspondences between abstract and concrete descriptions of actions. The dataset consists of higher-level single-sentence imperative descriptions paired with lower-level descriptions with elided details included. Descriptions come from real grounded actions, built on top of the YouCookII video caption dataset (Zhou et al., 2017).

Figure 1 gives an example annotation from the dataset: the phrase "drain the pasta," presented to an annotator with its corresponding video clip, was annotated as corresponding to four constituent steps appropriate as instruction for a child. The constituent steps are "simpler" in the sense that they correspond to more atomic actions, but not necessarily in their linguistic complexity. We identify over 1,500 procedures and tools which **KIDSCOOK** makes explicit but are assumed as

---

[*] Author now at Google. Work done while unaffiliated.

Proceedings of NAACL-HLT 2019, pages 4077–4085
Minneapolis, Minnesota, June 2 - June 7, 2019.

commonsense world knowledge by YouCookII.

The **KidsCook** dataset allows us to learn mappings between abstract and concrete descriptions via sequence-to-sequence prediction. We apply several standard neural sequence-to-sequence models; however, since these models do not expose explicit, interpretable correspondences between abstract and concrete descriptions, we also propose the application of neural transduction models which capture correspondences with latent hard alignment variables. We define a cloze-style evaluation to complement our dataset, in which models must predict the values of held-out tokens which target knowledge of tool usage, temporal ordering, and kitchen commonsense. We find that our neural transduction models are able to match the predictive power of traditional neural sequence models while providing interpretable alignments between abstract and concrete subsequences useful for our primary goal of analysis of implicit hierarchical script knowledge.

2 Data & Task

Our approach situates script learning as a case of grounding. For simplicity of exposition, let us assume there are three levels of abstraction to grounding: *abstract* → *concrete* → *motor control*. Most prior work in grounding treats language monolithically[1] and ignores the issue of audience. In practice, this means the task formulation or exposed API may implicitly bias the language to be more concrete. By viewing the task as purely linguistic, we have no API or robot that constrains our language; instead, we define our audience as children. By eliciting child-directed instructions, we collect concrete language capturing otherwise implicit world knowledge that a child would not know. Because annotators assume a smart and capable but uninformed listener, we posit this language corresponds closely to the most "concrete" form in which language naturally occurs.

2.1 Data Collection

We construct a task on Amazon's Mechanical Turk, where workers are asked to explain a video action caption to a child.[2] Every instruction is paired with the original YouTube video and YouCook caption so the annotator could see how the action was performed, rather than hallucinating additional details. All captions received three simplifications. The instructions ask users to focus on missing information and allow them up to five steps. Finally, we explicitly asked annotators to simplify complex actions (e.g. *dice*) that can be defined by a series of more basic actions (e.g. *cut*).

	Seqs	Tokens	Vocab	Avg Len YC	 **KC**
Train	8,125	307,573	3,573	10.0	37.9
Valid	1,014	36,830	1,479	8.8	36.3
Test	1,020	37,156	1,489	8.8	36.4

Table 1: **KidsCook** corpus statistics

Our **KidsCook** corpus statistics are shown in Table 1. In total we collected over 10K action sequences (~400K tokens). The average caption is approximately 4x longer than a YouCook caption. Most importantly 1,536 lemmas and 2,316 lexical types from **KidsCook**'s vocabulary do not appear in any of the original captions. This indicates that there are over 1,500 new concepts, tools, and procedures that were assumed by YouCookII but are now explicit in **KidsCook**.

2.2 Cloze Task

To investigate what new knowledge is being introduced and whether a model has captured it, we construct a cloze-style slot-filling task (Chambers, 2017; Hermann et al., 2015). We drop key content words from the concrete realization of an abstract instruction and ask the model to predict them. Several examples from the validation set are shown in Table 2. Correctly predicting the missing words requires knowledge of the manner of executing a task and the tools required.

To choose candidate words to drop, we only allow words that occur primarily in the concrete instructions. Additionally, we do not drop stop words, numbers, or words occurring fewer than five times. We do, however, drop units of measure (*cup*, *minute*, etc.). This ensures we create blanks whose answers are previously omitted concrete details. Relatedly, under this filter the answer to a blank is very rarely an ingredient, as our goal is not to memorize recipes, but to infer the tool knowledge necessary to execute them. In total, we whitelist ∼1,000 words that can be dropped to create blanks. We prefer longer blanks when available to give preference to compound nouns (e.g. *wire whisk*). Finally, we do not drop any words

---

[1] Notable exceptions include the hierarchical instructions of (Regneri et al., 2013) and (Bisk et al., 2016).

[2] Pay was calculated based on $15/hr and assuming workers took 1.5x as long to complete a task as the experimenters.

ABS	chop garlic into small pieces .
CON	put garlic on **cutting board**. **press** on back of **knife** with **hand**, **cutting** into small pieces.
ABS	add some parmesan cheese into the bowl and mix them well.
CON	use a **grater** to **grate** some parmesan cheese into the bowl. use a **wire whisk** to **stir** the cheese in.
ABS	add the tofu to the wok.
CON	drain the **water** from the tofu using a **strainer**. add the tofu into the **pan**. use a **spoon** to **stir** the tofu in the mixture.

Table 2: Example abstract/concrete pairs with blanks (**red**) where predictions and surprisal are computed.

from the concrete sentence if they occur in the abstract description. This restriction eliminates any benefits that might have been achieved via models with copy mechanisms. Examples that do not meet our criteria are removed from the corpus.

3 Models

We investigate the utility of sequence-to-sequence models with attention (Bahdanau et al., 2015) to generate concrete realizations of abstract task descriptions. We hypothesize that models that learn explicit alignments are particularly amenable to interpretable analysis on the task. Therefore, in addition to using the global attention model of (Luong et al., 2015), we adapt the transducer model proposed by Yu et al. (2016), which uses learned latent discrete variables to model phrase-to-phrase alignments. In contrast to many standard neural models, this approach enables us to incorporate prior knowledge about the alignment structure, and to extract interpretable alignments between task phrases. Closely related architectures have been proposed for segmental sequence modeling (Wang et al., 2017) and phrase-based neural machine translation (Huang et al., 2018).

We train the transducer models using Viterbi EM (after doing marginal likelihood training for the initial iterations), as we found it gave higher predictive accuracy than marginal likelihood training only. Following Yu et al. (2016) we experiment with both a fixed alignment transition probability model and a transition model with a neural parameterization. Cloze task prediction is performed greedily.[3] At each slot the Viterbi alignment of the prefix of the sequence up to that slot is computed. See appendix 7 for model details.[4] We also evaluate the performance of a language modelling baseline and a seq2seq model without attention (Sutskever et al., 2014), to compare the effect of not modeling alignment at all.

We expect all the models to implicitly capture aspects of world knowledge. However, the discrete latent variable models provide Viterbi alignments over the training data, from which we can compile a look-up table with the extracted knowledge. In neural attention models, this knowledge is only weakly recoverable: extracting information requires hand tuning attention thresholds and there is no direct way to extract contiguous alignments for multi-word phrases.

4 Results

4.1 Evaluation Metrics

During generation, we provide the model with the number of words in each blank to be predicted. We consider two setups for evaluating examples with multiple blanks, both assuming that predictions are made left-to-right: Oracle, where the gold prediction of each blank is fed into the model to condition on for future predictions, and Greedy, where the model prediction is used for future predictions. We compute the proportion of exact word matches over each blank and the precision of the top $k = 5$ predictions for both setups. Additionally we compute the average surprisal of the gold prediction (conditioning on oracle predictions). The *surprisal* of a word (Attneave, 1959; Hale, 2001) is its negative log probability under the model: $-log(P(w_i|w_{1:i-1}))$. The higher the probability of the ground truth, the lower the model's "surprise" at seeing it in that context.

Finally, as a quantitative proxy for interpretability, we report the length of the transducer models' average Viterbi alignment span: our goal is a model which balances low average alignment lengths and high matching or ranking scores.

4.2 Cloze Task Results

We report results on the prediction task in Table 4. First we consider models trained only on our dataset: All the models that incorporate a notion of alignment do substantially better than those who

---

[3] During preliminary experiments beam search did not improve performance.

[4] All code and data is available at `https://github.com/janmbuys/ScriptTransduction`.

abstract → concrete		concrete → abstract	
parmesan	sprinkle grated, grate, hold a grater, ...	whisk	eggs, mayonnaise, milk, combine, pour, stir, ...
macaroni	stove on high heat, large pot, bowl, ...	spatula	colors, thickens, coated, simmer, grill, ...
egg	place the boiled, gently crack the, crack, ...	tongs	shrimp, bratwurst, turn, bun, marinate, ...
sauce	stir hot, pour gravy, lower setting, find a spoon, ...	cutting board	onions, bell pepper, meat, bok choy, ...
oil	spray cooking, splashing, slowly pour, ...	preheat	oven, broil, medium, degrees, ...

Table 3: Example Viterbi Alignments. For concrete to abstract, we match any phrase containing the word(s).

	Oracle		Greedy		
Model	Match	Top-5	Match	Top-5	Surp
Language Model	21.59	52.32	21.72	43.33	3.970
Seq2seq	23.57	53.38	23.44	45.76	3.755
+Att	24.52	53.98	24.57	47.34	3.780
Transducer	*24.72*	*55.09*	*24.91*	*47.80*	4.780
+ParamTran	23.81	53.98	24.00	46.19	*3.547*
OpenAI GPT	23.19	42.43	15.55	32.86	4.781
+ fine-tuning	**38.01**	**63.69**	**31.05**	**57.05**	**3.151**

Table 4: Results on the Cloze prediction task (Match = Exact Match, Top-5 = Precision of Top 5 predictions, Surp = Surprisal). Transducer results are reported for models with unparameterized and parameterized (+ParamTran) alignment transition models. The **best** and *second best* results are emphasized.

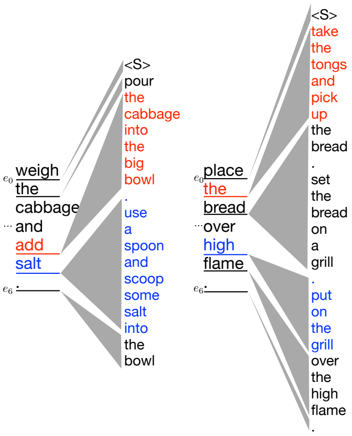


Figure 2: Example Viterbi alignments

do not. We see that our transducer model with fixed alignment transition probabilities performs best in terms of predictive accuracy (exact match and top-5 precision), while the seqseq model with attention is the next best in most comparisons. The model with parameterized transitions has the lowest surprisal though, as it is more confident about the alignment predictions it is making.

Using average alignment length to quantify whether the phrase alignments exhibit desirable structure, we see that the alignments found by the unparameterized transition model (average length 6.18) are significantly shorter than those of the parameterized model (average length 16.61). Investigation shows that the paramaterized model mostly learns degenerate alignments, aligning most of the concrete sequence to either the start or end of the abstract sentence. In contrast, qualitative analysis of the unparameterized transition model show that its alignments learn desirable correspondences (see Figure 2). Therefore among our proposed models (trained on in-domain data only) the transducer with unparameterized transitions satisfies our desiderata of displaying both good predictive power for word generation, and learning interpretable alignments.

Given the recent success of massively pretrained language models (Peters et al., 2018), we are interested if these approaches transfer to our cloze task. We evaluate the OpenAI GPT transformer language model (Radford et al., 2018) with and without fine-tuning.Without fine-tuning this model does slightly worse than our best domain-specific model. With fine-tuning, its accuracy is substantially higher, but it still suffers from the same fundamental limitations as our other models (see Table 5). The transformer (Vaswani et al., 2017) attention is multi-headed and multi-layered which prohibits direct interpretability.

5 Qualitative Analysis

We visualize alignments of our transduction model over two partial sequences in Fig. 2. This shows which hidden vector of the abstract sentence aligned to every region of the concrete sequence. Specifically, we see how tools like *the big bowl*, *spoon*, and *tongs* are introduced to facilitate the actions. There are also implications, e.g. that *high* indicates *grill*. For further analysis we extract alignments over the training corpus, linking each decoded phrase with the word from the encoding it used during generation. We then aggregate these tuples into a table which we can filter (based on our whitelist) and sort (with PMI). This process is imprecise as it discards the context in which the alignment occurs, but it nonetheless extracts many

`Abs`	shape each dough ball into a circle and add tomato sauce .
`Pred`	flatten out your dough into a flat circle using your **hands**. take a **knife** to add tomato sauce to the center of your dough. use the back side of the **knife** to **cut** the sauce out. make **sure** you keep the sauce about **an inch** from the edges.
`Gold`	flatten out your dough into a flat circle using your **hands**. take a **spoon** to add tomato sauce to the center of your dough. use the back side of the **spoon** to **spread** the sauce out. make **sure** you keep the sauce about **an inch** from the edges.
`Abs`	place the kale cucumber bell peppers carrots and radishes on the wrapper .
`Pred`	put the **cut** on a **cutting** . put a **cutting** amount of kale on the **cutting** . add a **cut** amount of cucumber ...
`Gold`	put the **wrap** on a **plate** . put a **small** amount of kale on the **wrap** . add a **small** amount of cucumber ...
`Abs`	wrap the pizza .
`Pred`	find a **large piece** to put the pizza om . place the pizza in the center for it not to **stick** around . grab the plastic wrap and start **wrapping** the **entire thing** and pizza . wrap all around until **completely covered** on all corners . put in freezer on a **cold water** and freeze overnight
`Gold`	find a **hard surface** to put the pizza om . place the pizza in the center for it not to **slide** around . grab the plastic wrap and start **wrapping** the **hard surface** and pizza . wrap all around until **fully covered** on all corners . put in freezer on a **flat surface** and freeze overnight

Table 5: Above is the output of OpenAI GPT when forced to greedily decode answers to blanks in validation.

of the phenomena we would hope to see (Table 3).

The left-hand side of the table shows words from the abstract YouCook annotations and corresponding phrases in the concrete annotation. For the righthand side we searched for common concrete terms that may be preceded or followed by other terms, and present the abstract terms they were most often generated by.

Finally, Table 5 shows three randomly chosen examples (from the validation set) of greedy decodings for slot filling with GPT fine-tuned on our dataset. These examples demonstrate that, first, there are cases where GPT is successful or produces a semantically valid answer (e.g. *fully* vs *completely*). Second, as is common with greedy decoding, the model can get stuck in a loop (e.g. *cut, cutting, cutting, ...*). Finally, note there are nonsensical cases where the model appears to have discarded the abstract context (e.g. *knife to add tomato sauce* or *freezer on a cold water*).

6 Related Work

Many script learning systems are based on event co-occurrence and language modeling in large text corpora, and can infer implicit events without creating explicit situation-specific frame structures (Chambers and Jurafsky, 2008; Rudinger et al., 2015; Pichotta and Mooney, 2016). Other systems induce situation-specific frames from text (Cheung et al., 2013; Balasubramanian et al., 2013). However, these methods do not explicitly target the commonsense correspondence between differing levels of detail of complex events.

Most relevant to this paper is the pioneering work of Regneri et al. (2013) as extended by Senina et al. (2014) and Rohrbach et al. (2014). These papers present the TACOS corpus, consisting of natural language descriptions of activities in videos paired with low-level activity labels. Senina et al. (2014) collect an additional level of multi-sentence annotations on the corpus, which allowing for video caption generation at multiple levels of detail. Rohrbach et al. (2014) describe a similar corpus of natural descriptions of composite actions, useful for activity recognition in video. These corpora differ in a number of important ways from **KIDSCOOK**; in particular, the language has somewhat limited complexity and "naturalness" when describing complex scenarios, a phenomenon also observed in the robotics literature (Scalise et al., 2018). Our data collection process avoids more formulaic language by eliciting "child-directed" descriptions.

7 Conclusion

We introduce a new hierarchical script learning dataset and cloze task in which models must learn commonsense world knowledge about tools, procedures and even basic physics to perform well. Our aim is to begin a conversation about abstraction in language, how it is modeled, and what is implicitly hidden. Our abstract and concrete instructions are grounded in the same videos yet differ dramatically due to their assumed audiences. We show that a neural transduction model produces interpretable alignments for analyzing these otherwise latent correlations and phenomena.

Acknowledgements

This work was supported in part by NSF (IIS-1524371 & 1703166) and through DARPA's CwC program through ARO (W911NF-15-1-0543).

References

Fred Attneave. 1959. *Applications of information theory to psychology: A summary of basic concepts, methods, and results.* Henry Holt.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*.

Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI 2016 Workshop on Symbiotic Cognitive Systems*, pages 729–732, Phoenix, AZ.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *CL*, 19(2):263–311.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*.

Nathanael Chambers. 2017. Behind the scenes of an evolving event cloze test. In *LSDSem*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *NAACL*.

Gerald F. DeJong. 1981. Generalizations based on explanations. In *IJCAI*.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society.*, pages 1–38.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Workshop on Structured Prediction for NLP*.

H. Paul Grice. 1975. Logic and conversation. *Syntax and Semantics 3: Speech Acts*, pages 41–58.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *ICLR*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI*.

Alex Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Mechaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, 1.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *GCPR*.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015. Learning to predict script events from domain-specific text. In **Sem*.

Rosario Scalise, Yonatan Bisk, Maxwell Forbes, Daqing Yi, Yejin Choi, and Siddhartha Srinivasa. 2018. Balancing shared autonomy with human-robot communication. *arXiv preprint arXiv:1805.07719*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. LEA.

Anna Senina, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. *arXiv preprint arXiv:1403.6173*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *ICML*.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018. Hierarchical quantized representations for script generation. In *EMNLP*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017. The neural noisy channel. In *ICLR*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2017. Towards automatic learning of procedures from web instructional videos. *arXiv preprint arXiv:1703.09788*.

A Transducer Model

We briefly describe the model of Yu, Buys, and Blunsom (2016) and our minor modifications thereto.

A.1 Alignment with Latent Variables

We model the conditional probability of a concrete sequence y given abstract sequence x through a latent alignment variable a between x and y, which is a sequence of variables a_j, with $a_j = i$ signifying that y_j is aligned to x_i. The marginal probability of y given x is

$$p(y|x) = \sum_a p(y, a|x). \tag{1}$$

In the following, we use m to denote the length of x and n to denote the length of y. The model formulation restricts alignments to be monotonic, i.e. $a_{j+1} \geq a_j$ for all j.

The model factorizes over timesteps into alignment and word prediction probabilities, such that the word prediction at each timestep is informed by its alignment:

$$\begin{aligned} p(y, a|x) = \prod_j & p(a_j | a_{j-1}, x_{1:a_{j-1}}, y_{1:j-1}) \\ & \times p(y_j | a_j, x_{1:a_j}, y_{1:j-1}) \end{aligned} \tag{2}$$

The abstract and concrete sequences are both encoded with LSTM Recurrent Neural Networks (Hochreiter and Schmidhuber, 1997). In contrast to standard attention-based models, the aligned encoder representation is not fed into the decoder RNN state, but only used to make next word predictions. Due to the small size of the training data, words in both sequences are embedded using fixed GloVe embeddings (Pennington et al., 2014). The word emission probability is then defined as

$$p(y_j | a_j, x_{1:a_j}, y_{1:j-1}) = \text{softmax}(\text{MLP}(e_{a_j}, d_j)) \tag{3}$$

with e the encoder hidden states and d the decoder hidden states.

The alignment probability factorizes into *shift* and *emit* probabilities, where a *shift* action increments the alignment to the next word in the input sequence, and an *emit* action generates the next output word. We refer to these as *transition* probabilities. This formulation enables us to restrict the hard alignment to be monotonic.

We consider two parameterizations of this distribution. In the first, the probabilities are parameterized by the neural network, using the encoder and decoder hidden state in a similar manner to how the word emission probability was computed. The alignment probability at a given timestep is therefore parameterized as

$$\begin{aligned} p(a_j | a_{j-1}, x_{1:a_{j-1}}, y_{1:j-1}) &= p(\text{emit} | a_j, x_{1:a_j}, y_{1:j-1}) \\ & \times \prod_{i=a_{j-1}}^{a_j - 1} p(\text{shift} | i, x_{1:i}, y_{1:j-1}), \end{aligned} \tag{4}$$

where

$$p(\text{shift} | i, x_{1:i}, y_{1:j-1}) = \sigma(MLP(e_i, d_j)), \tag{5}$$

$$p(\text{emit} | i, x_{1:i}, y_{1:j-1}) = 1 - p(\text{shift} | i, x_{1:i}, y_{1:j-1}). \tag{6}$$

We also consider using the simpler, fixed alignment parameterization in Yu, Buys, and Blunsom (2016), where the transition probability is conditioned only on sequence length, not on x or y, and can therefore be estimated using the ratio between input and output sentence lengths. The alignment probabilities are not updated during training, and consequently the posterior distribution over the alignments is biased towards this prior, favoring alignments close to the diagonal.

The parameterized alignment model contains as special cases two degenerate solutions: (1) an unconditional language model and (2) a seq2seq model. These occur if the model performs all emits before shifting or all shifts before emitting, respectively. To prevent the creation of a language model we force the last output word to be aligned to the last word in the abstract sequence, similar to Yu et al. (2017). However, the parameterized transition model could still in practice revert to a pure sequence-to-sequence model.

A.2 Marginalization

Next we briefly describe the dynamic program used to marginalize over alignments during training and to find the most likely alignments of a given alignment during inference; we refer the reader to Yu, Buys, and Blunsom (2016) for a more thorough treatment.

The forward variable $\alpha_i(j)$ representing $p(y_{1:j}, a_j = i | x_{1:i})$ is recursively as

$$\begin{aligned} \alpha_i(j) &= p(y_j | i, x_{1:i}, y_{1:j-1}) \\ & \times \sum_{k=1}^{i} \alpha_k(j-1) p(a_j = i | k, x_{1:k}, y_{1:j-1}). \end{aligned} \tag{7}$$

The marginal likelihood objective is to train the model to optimize $\alpha_m(n) = p(y_{1:n}, a_n =$

$m|x_{1:m})$. The gradients are computed with automatic differentiation; as this is is equivalent to using the forward-backward algorithm to estimate the gradients (Eisner, 2016), only the forward algorithm has to be implemented.

To make the implementation GPU-efficient, we vectorize the computation of α. The computation iterates through decoding steps, each of which can be generated from an alignment to any of the encoder tokens. We can efficiently construct a transition matrix T, corresponding to all possible encoder states performing all possible shifts, and emission matrix E_j which is a gather by word index j.

To compute the forward probabilities at each timestep, the current forward probabilities are first multiplied by all possible transitions. A sum in logspace collapses all paths, and the emission (word generation) probabilities are multiplied to obtain the new forward probabilities. When fixed transition probabilities are used, T is precomputed.

A.3 Viterbi EM Training

Latent variable models can be trained either through directly optimizing the likelihood objective through gradient descent (as described above), or with the Expectation Maximization (EM) algorithm (Dempster et al., 1977), which alternates between calculating expectations over the values of the latent variables given the current parameters, and maximizing the expected complete data log likelihood given those expectations. We consider training our alignment model with Viterbi EM (Brown et al., 1993), also known as “hard” EM, where at each iteration the most likely assignment of the hidden variables (alignments) are found and the parameters are updated to optimize the log likelihood given those alignments. Viterbi EM has been shown to give superior performance to standard EM on unsupervised parsing (Spitkovsky et al., 2010), due to better convergence properties in practice by making the distribution more peaked.

We perform batched Viterbi EM training by computing the Viterbi alignments for a batch, and then performing a gradient step based on treating those alignments as observations.

We follow a two-stage training procedure: we first directly optimize the marginal likelihood with batched SGD to find a reasonable initial distribution over alignments, before switching to Viterbi EM training. Such a strategy has been shown to reduce the chance that the model will get stuck in local optima (Spitkovsky et al., 2011).

A.4 Inference

We apply the trained models to multiple inference problems to evaluate how well they are capturing script knowledge. The first is finding the most likely alignment given a pair of abstract and concrete sequences. We use the standard Viterbi algorithm, in which we replace the sum in equation (7) with $\max$, and keep track of the index corresponding to each value of α during the forward computation. The most likely alignment can then be traced back from $a_n = m$.

The second inference problem is slot-filling, for application to the cloze task. Given an abstract sentence and a partially-filled concrete sequence, we want to use the model to predict words to fill the given blanks. To make the prediction, we sample 5 candidate sequences by predicting words for each slot, in left-to-right order, and then choosing the sequence with the highest overall probability. Words are predicted by sampling with temperature 0.1, in order to peak the distribution while still allowing some diversity in the samples. The motivation for selecting the final output from multiple samples is that the original samples are biased, as they are only conditioned on the left context.

At the start of the prediction for each slot, the Viterbi alignment of the prefix of the sequence up to the start of that slot is re-predicted, independent of previous alignment predictions. Consequently alignment decisions can be revised, and the slot alignments are no longer constrained to be monotonic, which makes the slot prediction model more flexible. For the parameterized transition model, the slot alignment is predicted greedily by incrementing the last predicted alignment while the shift probability is greater than 0.5. The fixed transition model assumes that the alignment of the word preceding the slot is shared across the slot.

A Large-Scale Study of the Effects of Word Frequency and Predictability in Naturalistic Reading

Cory Shain
Department of Linguistics
The Ohio State University
`shain.3@osu.edu`

Abstract

A number of psycholinguistic studies have factorially manipulated words' contextual predictabilities and corpus frequencies and shown separable effects of each on measures of human sentence processing, a pattern which has been used to support distinct mechanisms underlying prediction on the one hand and lexical retrieval on the other. This paper examines the generalizability of this finding to more realistic conditions of sentence processing by studying effects of frequency and predictability in three large-scale naturalistic reading corpora. Results show significant effects of word frequency and predictability in isolation but no effect of frequency over and above predictability, and thus do not provide evidence of distinct mechanisms. The non-replication of separable effects in a naturalistic setting raises doubts about the existence of such a distinction in everyday sentence comprehension. Instead, these results are consistent with previous claims that apparent effects of frequency are underlyingly effects of predictability.

1 Introduction

Are there distinct effects of a word's frequency versus predictability in human sentence comprehension? Recent evidence implicates prediction as a major organizing principle in cognition (Bubic et al., 2010; Singer et al., 2018; Keller and Mrsic-Flogel, 2018), and psycholinguists have long studied the role of prediction in human sentence processing and its relation to other comprehension mechanisms (Marslen-Wilson, 1975; Kutas and Hillyard, 1984; MacDonald et al., 1994; Tanenhaus et al., 1995; Hale, 2001; Norris, 2006; Levy, 2008; Frank and Bod, 2011). Some prominent theories of word recognition claim that ease of lexical access is modulated by the strength of a word's representation in memory, independently of contextual factors that guide prediction (Seidenberg and McClelland, 1989; Coltheart et al., 2001; Harm and Seidenberg, 2004). Other theories hold that apparent effects of frequency are underlyingly effects of predictability (Norris, 2006; Levy, 2008; Rasmussen and Schuler, 2018).

A number of studies using constructed stimuli that factorially manipulate word frequency and predictability have found separable additive effects of each, suggesting distinct influences on lexical processing (see Staub, 2015 for a review). This paper examines the generalizability of these findings to typical sentence comprehension by searching for separable effects of frequency and *n*-gram predictability using deconvolutional time series regression (DTSR) models (Shain and Schuler, 2018) fitted to three large naturalistic reading corpora: Natural Stories (Futrell et al., 2018), Dundee (Kennedy et al., 2003), and UCL (Frank et al., 2013). While results show evidence of both frequency and predictability effects in isolation, they show no effect of frequency over predictability and thus do not support the existence of separable effects. They are instead consistent with either (1) an account of apparent frequency effects as epiphenomena of predictive processing (Norris, 2006; Levy, 2008) or (2) a more circumscribed role for frequency effects in naturalistic reading than constructed experiments suggest.

2 Background and Related Work

2.1 Frequency and Predictability in Human Sentence Processing

It has long been recognized that low-frequency words are harder to process (Inhoff and Rayner, 1986). For example, in a neutral context, the more frequent *bottle* should on average be processed more quickly than the less frequent *kettle*:

Proceedings of NAACL-HLT 2019, pages 4086–4094
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

(1) a. I have a **bottle**.
b. I have a **kettle**.

However, context can dramatically alter these patterns by changing words' predictability (Ehrlich and Rayner, 1981):

(2) a. the pot calling the **bottle** black
b. the pot calling the **kettle** black

Some models of word recognition (Seidenberg and McClelland, 1989; Coltheart et al., 2001; Harm and Seidenberg, 2004) posit a context-independent lexical retrieval mechanism, distinct from any mechanisms for predictive coding, with processing cost proportional to the strength of a word's representation in memory (a function of lexical frequency). Such a view predicts separable effects of frequency and predictability in human language comprehension. Other models (Hale, 2001; Norris, 2006; Levy, 2008; Rasmussen and Schuler, 2018) posit no such context-independent retrieval mechanism, and instead propose a unified comprehension mechanism that incrementally reallocates resources between possible interpretations of the unfolding sentence, with processing cost proportional to the amount of information (resource reallocation) contributed by each new word. Such a view predicts no separable effects of frequency and predictability because lexical frequencies are subsumed into the incremental probability model.

Consistently with the first hypothesis, previous studies have shown separable additive effects of frequency and predictability by factorially manipulating corpus frequency and cloze predictability (Rayner et al., 2004; Ashby et al., 2005; Gollan et al., 2011; Staub and Benatar, 2013, see Staub, 2015 for a review). However, cloze estimates poorly distinguish degrees of low contextual probability (Smith and Levy, 2013), and constructed stimuli, while affording direct control over linguistic variables, may fail to reflect the typical distributional characteristics of the language, lack context, and/or inadvertently trigger suspension of the usual processes of pragmatic inference due to the absence of an overarching discourse (Demberg and Keller, 2008; Hasson and Honey, 2012; Shain et al., 2018). It is therefore not yet clear whether frequency and predictability effects can be separated in a more realistic setting.

2.2 The Naturalistic Experimental Paradigm

Concerns about the ecological validity of constructed stimuli can be addressed by the use of naturalistic stimuli (e.g. stories, newspaper articles, persuasive pieces, etc.). Naturalistic experiments are therefore an important complement to constructed experiments in the study of cognitive processes (Hasson and Honey, 2012).

However, naturalistic experiments introduce their own challenges. Without the ability to factorially manipulate frequency and predictability, naturalistic studies must confront the natural collinearity between these two variables in ordinary language (Demberg and Keller, 2008). Furthermore, because naturalistic stimuli do not define a critical region of the stimulus, responses are generally modeled word-by-word (Demberg and Keller, 2008; Frank and Bod, 2011; Smith and Levy, 2013; van Schijndel and Schuler, 2015). It is standard psycholinguistic practice to do so through ablative likelihood ratio testing (LRT) of linear mixed effects regression (LMER) models (Bates et al., 2015) fitted to the dependent variable of interest (e.g. fixation duration) (Demberg and Keller, 2008; Frank and Bod, 2011; van Schijndel and Schuler, 2015; Shain et al., 2016). However, this approach has important disadvantages. First, naturalistic data constitute time series that may violate the independence assumptions of linear regression and therefore confound model interpretation and hypothesis testing (Baayen et al., 2017, 2018; Shain and Schuler, 2018). One major such confound is *temporal diffusion* (i.e. a lingering response to stimuli), which can be brought under statistical control through deconvolutional time series regression (DTSR) models that directly estimate temporal structure in the relationships between predictors and response (Shain and Schuler, 2018). Second, LRT implicitly evaluates on in-sample data, making it challenging to diagnose overfitting and to assess external validity (Vasishth et al., 2018). This can be addressed through out-of-sample non-parametric tests, such as the paired permutation test widely used in machine learning (Demšar, 2006).

3 Experimental Setup

This paper seeks to complement constructed stimulus experiments by searching for separable effects of frequency and predictability during naturalistic reading, using methods designed to ad-

	Effect estimate (log-ms)							
Corpus	SentPos	Trial	Rate	WordLen	SaccLen	PrevFix	Unigram	5-gram
Natural Stories	0.0098	-0.0216	-0.3069	—	—	0.0158	-0.0018	0.0174
Dundee	-0.0085	-0.0052	-0.0277	0.0068	-0.0021	-0.0178	-0.0067	0.0117
UCL	0.0524		-0.1330	0.0023	0.0221	0.0778	0.0005	0.0184

Table 1: Effect estimates in log-ms by corpus, computed as the IRF integral over the longest time offset seen in training. Following psycholinguistic convention, unigrams and 5-grams have opposite sign (log prob vs. surprisal). In UCL, *sentence position* and *trial* are identical (sentences were shuffled).

Corpus	ρ
Natural Stories	-0.78
Dundee	-0.73
UCL	-0.74

Table 2: Pearson's correlation between *5-gram surprisal* and *unigram log probability* by corpus.

dress the challenges of Section 2.2. The problem of temporal diffusion is addressed by using DTSR models rather than LMER (see Appendix A for implementation details). The problem of external validity is addressed by using held-out paired permutation testing rather than LRT, thus basing the hypothesis test directly on generalization error. The possibility that cloze probabilities are poor estimates of predictability for low-frequency words is addressed by operationalizing predictability as 5-gram surprisal generated by a large-vocabulary statistical language model. The natural collinearity of frequency and predictability is addressed through the use of large-scale data that should permit subtle differentiation of collinear effects. Taken together, the corpora examined in this study contain over one million fixations generated by 243 human subjects. Although there is a large-magnitude correlation between *unigram log probability* (frequency) and *5-gram surprisal* (predictability) in these corpora, as shown in Table 2, synthetic experiments show that DTSR can faithfully identify models from much smaller data than that used here, even when all predictors are correlated at the 0.75 level (Shain, 2018). Given the size of the data, failure to distinguish effects of frequency and predictability would raise doubts about the existence of such a separation in naturalistic reading.

3.1 Statistical Procedure

DTSR models are fitted separately to each of the Natural Stories (Futrell et al., 2018), Dundee (Kennedy et al., 2003), and UCL (Frank et al., 2013) corpora.[1] Following previous investigations of this question (Rayner et al., 2004; Ashby et al., 2005; Gollan et al., 2011, *inter alia*), frequency is estimated from corpus statistics — in this case, KenLM (Heafield et al., 2013) unigram models trained on the Gigaword 3 corpus (Graff and Cieri, 2003). Unlike previous studies using close estimates of predictability (Rayner et al., 2004; Ashby et al., 2005; Gollan et al., 2011, *inter alia*), predictability is statistically estimated, again using KenLM models (5-gram) trained on Gigaword 3. This is both because (1) cloze norming all words contained in thousands of naturalistic sentences is prohibitive and (2) statistical language models trained on large data can more reliably differentiate low probability continuations (Smith and Levy, 2013). Following recent work on prediction effects in naturalistic sentence comprehension (Demberg and Keller, 2008; Frank and Bod, 2011; Smith and Levy, 2013), predictability estimates are encoded as *surprisal* by negating the 5-gram log probabilities.

The models assume ShiftedGamma impulse response functions (Shain and Schuler, 2018, see Appendix A) for each of these variables, as well as for the nuisance variables *word length*, *saccade length* and an indicator variable for whether the previous word was fixated.[2] To capture trends in the response at different timescales, the mod-

---

[1] Natural Stories is a self-paced reading corpus containing 848,768 word fixations from 181 subjects reading narrative and informational texts. Dundee is an eye-tracking corpus containing 260,065 word fixations from 10 subjects reading newspaper editorials. UCL is an eye-tracking corpus containing 53,070 fixations from 42 subjects reading sentences taken from novels by amateur authors.

Although the sentences in UCL were randomized and presented in isolation — and therefore subject to some of the concerns about constructed stimuli raised in Section 2 — they are included here because the stimuli are naturally occurring rather than constructed for a particular experimental purpose. The UCL results replicate the overall pattern of significance (Table 3), and excluding them has no impact on the overall results.

[2] The variables *saccade length* and *previous was fixated* are only used for eye-tracking since they are not relevant to self-paced reading.

		Corpus		
Comparison	Pooled	Natural Stories	Dundee	UCL
5-gram only vs. baseline	**0.0001*****	**0.0001*****	**0.0001*****	**0.0001*****
Unigram only vs. baseline	**0.0001*****	**0.0001*****	**0.0001*****	**0.0001*****
5-gram + Unigram vs. Unigram-only	**0.0001*****	**0.0001*****	0.0626	**0.0006*****
5-gram + Unigram vs. 5-gram-only	0.1515	0.1831	0.0105	0.1491

Table 3: Held-out paired permutation testing results, both pooled (left) and by corpus (right).

els also include linear effects for the word's index in the sentence (*sentence position*) and document (*trial*). Following Shain and Schuler (2018), in addition to the intercept, the models contain a convolved intercept (*rate*) designed to capture effects of stimulus timing. The response used in all corpora is log fixation duration (go-past for eye-tracking).[3] Outlier filtering is performed in each corpus following the procedures described in Shain and Schuler (2018).

Approximately half the data in each corpus is used for training, with the remaining half reserved for held-out evaluation. Models include by-subject random intercepts as well as by-subject random slopes and impulse response parameters for each predictor.[4] Held-out hypothesis testing uses a "diamond" ablative structure first ablating fixed effects for *5-gram surprisal* and *unigram log probability* individually and then ablating both. All random effects are retained in all models. Comparisons use paired permutation tests of the by-item losses on the evaluation set, pooling across all corpora.[5] Note that the non-parametric permutation test permits this pooling procedure to unify the models from all three corpora into a single test, since (unlike LRT) permutation testing supports out-of-sample comparison. Data processing was performed using the ModelBlocks toolchain (van Schijndel and Schuler, 2013), available at `https://github.com/modelblocks/modelblocks-release`. Model fitting was performed using the DTSR software library (Shain and Schuler, 2018), available at `https://github.com/coryshain/dtsr`. See the citations above for data access instructions.

4 Results

Effect estimates[6] from the full models are presented in Table 1 and pooled statistical comparisons are presented in the *Pooled* column of Table 3. If predictability and frequency effects are additive, all four comparisons in Table 3 should be significant. As shown, this is not the case. There is evidence that both frequency (*unigram log probability*) and predictability (*5-gram surprisal*) in isolation reliably index processing difficulty, as shown by the significance of both effects over the baseline. However, when the effects are compared to each other, predictability explains significantly more variance than frequency but not vice versa.

This general pattern of results further obtains for each corpus individually, as shown by the *Corpus* column breakdown in Table 3. One minor exception is that neither predictability nor frequency improves significantly over the other in Dundee.[7] The Dundee results are nevertheless consistent with an interpretation in which frequency and predictability do not index distinct processing phenomena and inconsistent with an interpretation in which they do. These results thus provide no evidence of separable frequency and predictability effects, whether the corpora are considered together or individually.

5 Discussion

As described in Section 4, results show no evidence of separable effects of frequency and predictability in naturalistic reading. One possible explanation for this outcome is that 5-gram surprisal tracks human prediction effort better than cloze probabilities, in part because cloze probabilities are less reliable for infrequent words. Although countervailing evidence exists in the literature (e.g. Smith and Levy, 2011 found effects of cloze but not *n*-gram probabilities in human read-

---

[3] The overall pattern of significance does not change when first-pass durations are used.

[4] By-word random intercepts are not included because of their potential to subsume frequency effects.

[5] To correct for different error variances, errors are rescaled by the joint standard deviation of the errors from the full and ablated models by corpus.

[6] The estimated impulse response functions that underlie these effect sizes are plotted in Appendix B.

[7] The *p*-value of 0.0105 observed for frequency over predictability does not achieve significance at the 0.05 level under 6-way Bonferroni correction (2 variables × 3 corpora).

ing times), in general this evidence is based on weak statistical competitors to cloze (e.g. Smith and Levy, 2011 used tri-grams). By contrast, recent trends in cognitive modeling point toward a correlation between the linguistic and psycholinguistic performance of language models, such that more powerful models with lower perplexity also tend to correlate more strongly with measures of cognitive effort (Goodkind and Bicknell, 2018; van Schijndel and Linzen, 2018). This suggests that apparent frequency effects may arise in part from poor estimates of predictability. Note that by using 5-gram surprisal rather than more powerful neural language models (Jozefowicz et al., 2016), the analysis described in this paper is conservative in its attribution of variance to predictability. The failure of frequency is thus all the more compelling, since replacing 5-gram surprisal with surprisals obtained from more powerful language models would be unlikely to increase the explanatory power of frequency.

Another potential explanation for the lack of separable effects of frequency and predictability is the use of naturalistic rather than constructed stimuli. Neuroscientific evidence shows that domain-general executive control regions activate during the processing of some artificially constructed language stimuli (Kaan and Swaab, 2002; Kuperberg et al., 2003; Novick et al., 2005; January et al., 2009) but fail to activate during the processing of naturalistic stimuli (Blank and Fedorenko, 2017). Such results have led some to argue that artificially constructed experimental stimuli may increase general cognitive load by coercing comprehension into problem solving, thereby engaging mechanisms that play little role in everyday sentence processing (Campbell and Tyler, 2018, Wehbe et al., in prep; Diashek et al., in prep). It is possible that the language comprehension mechanisms that implement linguistic prediction (Shain et al., under review) are relatively less engaged while domain general executive control mechanisms are relatively more engaged during the processing of constructed stimuli presented without context, perhaps suppressing the influence of preceding words on participants' reading behavior. Further investigation is needed in order to explore this hypothesis.

In any case, it is a statistical truism that negative results do not motivate acceptance of the null hypothesis. Thus, it is possible that frequency effects exist in naturalistic reading but are too small to be detected here. Nevertheless, the failure to find frequency effects in large naturalistic data indicates that any such effects are greatly attenuated in the processing of naturalistic texts in comparison to the processing of constructed stimuli, which circumscribes the importance that any such effects might have in driving comprehension effort during typical reading.

6 Conclusion

This paper explored whether effects of word frequency and predictability are distinguishable in naturalistic sentence processing. Despite the size of the combined dataset, results showed no evidence of separable effects in naturalistic reading, contrary to previous findings of separable effects in studies using constructed stimuli. This investigation thus shows no evidence of a distinct, context-independent lexical retrieval mechanism modulated by strength of memory representation (Seidenberg and McClelland, 1989; Coltheart et al., 2001; Harm and Seidenberg, 2004), and instead favors a view in which sentence processing effort is driven by a mechanism that incrementally reallocates resources between competing interpretations, subsuming any effects of raw lexical frequency (Norris, 2006; Levy, 2008; Rasmussen and Schuler, 2018). The discrepancy between constructed and naturalistic experimental settings presents a puzzle for our understanding of the mental processes that underlie human language comprehension, and is perhaps linked to recent evidence that artificially constructed linguistic stimuli can spuriously engage non-linguistic executive mechanisms by increasing general cognitive load as compared to naturalistic settings (Blank and Fedorenko, 2017; Campbell and Tyler, 2018). Further investigation into the precise sources of the discrepancy may shed new light on the interplay between prediction and memory in human sentence processing.

Acknowledgements

The author would like to thank the anonymous reviewers for their helpful comments. This work was supported by National Science Foundation grants #1551313 and #1816891. All views expressed are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

Jane Ashby, Keith Rayner, and Charles Clifton. 2005. Eye movements of highly skilled and average readers: Differential effects of frequency and predictability. *The Quarterly Journal of Experimental Psychology Section A*, 58(6):1065–1086.

Harald Baayen, Shravan Vasishth, Reinhold Kliegl, and Douglas Bates. 2017. The cave of shadows: Addressing the human factor with generalized additive mixed models. *Journal of Memory and Language*, 94(Supplement C):206–234.

R Harald Baayen, Jacolien van Rij, Cecile de Cat, and Simon Wood. 2018. Autocorrelated errors in experimental data in the language sciences: Some solutions offered by Generalized Additive Mixed Models. In Dirk Speelman, Kris Heylen, and Dirk Geeraerts, editors, *Mixed Effects Regression Models in Linguistics*. Springer, Berlin.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48.

Idan Blank and Evelina Fedorenko. 2017. Domain-general brain regions do not track linguistic input as closely as language-selective regions. *Journal of Neuroscience*, pages 3616–3642.

Andreja Bubic, D Yves Von Cramon, and Ricarda I Schubotz. 2010. Prediction, cognition and the brain. *Frontiers in human neuroscience*, 4:25.

Karen L Campbell and Lorraine K Tyler. 2018. Language-related domain-specific and domain-general systems in the human brain. *Current opinion in behavioral sciences*, 21:132–137.

Max Coltheart, Kathleen Rastle, Conrad Perry, Robyn Langdon, and Johannes Ziegler. 2001. DRC: a dual route cascaded model of visual word recognition and reading aloud. *Psychological review*, 108(1):204.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30.

Timothy Dozat. 2016. Incorporating Nesterov momentum into Adam. In *ICLR Workshop*.

Susan F Ehrlich and Keith Rayner. 1981. Contextual effects on word perception and eye movements during reading. *Journal of verbal learning and verbal behavior*, 20(6):641–655.

Stefan L Frank and Rens Bod. 2011. Insensitivity of the Human Sentence-Processing System to Hierarchical Structure. *Psychological Science*, 22(6):829–834.

Stefan L Frank, Irene Fernandez Monsalve, Robin L Thompson, and Gabriella Vigliocco. 2013. Reading time data for evaluating broad-coverage models of English sentence processing. *Behavior Research Methods*, 45(4):1182–1190.

Richard Futrell, Edward Gibson, Harry J . Tily, Idan Blank, Anastasia Vishnevetsky, Steven Piantadosi, and Evelina Fedorenko. 2018. The Natural Stories Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Tamar H Gollan, Timothy J Slattery, Diane Goldenberg, Eva Van Assche, Wouter Duyck, and Keith Rayner. 2011. Frequency drives lexical access in reading but not in speaking: The frequency-lag hypothesis. *Journal of Experimental Psychology: General*, 140(2):186.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18.

David Graff and Christopher Cieri. 2003. English Gigaword LDC2003T05.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8.

Michael W Harm and Mark S Seidenberg. 2004. Computing the meanings of words in reading: cooperative division of labor between visual and phonological processes. *Psychological review*, 111(3):662.

Uri Hasson and Christopher J Honey. 2012. Future trends in Neuroimaging: Neural processes as expressed within real-life contexts. *NeuroImage*, 62(2):1272–1278.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Albrecht Werner Inhoff and Keith Rayner. 1986. Parafoveal word processing during eye fixations in reading: Effects of word frequency. *Perception \& psychophysics*, 40(6):431–439.

David January, John C Trueswell, and Sharon L Thompson-Schill. 2009. Co-localization of Stroop and syntactic ambiguity resolution in Broca's area: Implications for the neural basis of sentence processing. *Journal of Cognitive Neuroscience*, 21(12):2434–2444.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Edith Kaan and Tamara Y Swaab. 2002. The brain circuitry of syntactic comprehension. *Trends in cognitive sciences*, 6(8):350–356.

Georg B Keller and Thomas D Mrsic-Flogel. 2018. Predictive Processing: A Canonical Cortical Computation. *Neuron*, 100(2):424–435.

Alan Kennedy, James Pynte, and Robin Hill. 2003. The Dundee corpus. In *Proceedings of the 12th European conference on eye movement*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6.

Gina R Kuperberg, Phillip J Holcomb, Tatiana Sitnikova, Douglas Greve, Anders M Dale, and David Caplan. 2003. Distinct patterns of neural modulation during the processing of conceptual and syntactic anomalies. *Journal of Cognitive Neuroscience*, 15(2):272–293.

M Kutas and S A Hillyard. 1984. Brain potentials during reading reflect word expectancy and semantic association. *Nature*, 307(5947):161–163.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Maryellen C MacDonald, Neal J Pearlmutter, and Mark S Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological Review*, 101(4):676–703.

William D Marslen-Wilson. 1975. Sentence Perception as an Interactive Parallel Process. *Science*, 189(4198):226–228.

Yurii E Nesterov. 1983. A method for solving the convex programming problem with convergence rate O(1/k^2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547.

Dennis Norris. 2006. The Bayesian Reader: Explaining word recognition as an optimal Bayesian decision process. *Psychological review*, 113(2):327.

Jared M Novick, John C Trueswell, and Sharon L Thompson-Schill. 2005. Cognitive control and parsing: Reexamining the role of Brocas area in sentence comprehension. *Cognitive, Affective, \& Behavioral Neuroscience*, 5(3):263–281.

Nathan E Rasmussen and William Schuler. 2018. Left-Corner Parsing With Distributed Associative Memory Produces Surprisal and Locality Effects. *Cognitive science*, 42:1009–1042.

Keith Rayner, Jane Ashby, Alexander Pollatsek, and Erik D Reichle. 2004. The effects of frequency and predictability on eye fixations in reading: Implications for the EZ Reader model. *Journal of Experimental Psychology: Human Perception and Performance*, 30(4):720.

Marten van Schijndel and Tal Linzen. 2018. A Neural Model of Adaptation in Reading. In *EMNLP 2018*, pages 4704–4710.

Marten van Schijndel and William Schuler. 2013. An Analysis of Frequency- and Memory-Based Processing Costs. In *Proceedings of Human Language Technologies: The 2013 Annual Conference of the North American Chapter of the ACL*.

Marten van Schijndel and William Schuler. 2015. Hierarchic syntax improves reading time prediction. In *Proceedings of NAACL-HLT 2015*. Association for Computational Linguistics.

Mark S Seidenberg and James L McClelland. 1989. A distributed, developmental model of word recognition and naming. *Psychological review*, 96(4):523.

Cory Shain. 2018. Deconvolutional time series regression: A technique for modeling temporally diffuse effects. Technical report, The Ohio State University, Columbus.

Cory Shain, Marten van Schijndel, Richard Futrell, Edward Gibson, and William Schuler. 2016. Memory access during incremental sentence processing causes reading time latency. In *Proceedings of the Computational Linguistics for Linguistic Complexity Workshop*, pages 49–58. Association for Computational Linguistics.

Cory Shain, Marten van Schijndel, and William Schuler. 2018. Deep syntactic annotations for broad-coverage psycholinguistic modeling. In *Workshop on Linguistic and Neuro-Cognitive Resources (LREC 2018)*.

Cory Shain and William Schuler. 2018. Deconvolutional time series regression: A technique for modeling temporally diffuse effects. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yosef Singer, Yayoi Teramoto, Ben D B Willmore, Jan W H Schnupp, Andrew J King, and Nicol S Harper. 2018. Sensory cortex is optimized for prediction of future input. *eLife*, 7:e31557.

Nathaniel J Smith and Roger Levy. 2011. Cloze but no cigar: The complex relationship between cloze, corpus, and subjective probabilities in language processing. In *Proceedings of the 33rd CogSci Conference*.

Nathaniel J Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128:302–319.

Adrian Staub. 2015. The effect of lexical predictability on eye movements in reading: Critical review and theoretical interpretation. *Language and Linguistics Compass*, 9(8):311–327.

Adrian Staub and Ashley Benatar. 2013. Individual differences in fixation duration distributions in reading. *Psychonomic Bulletin & Review*, 20(6):1304–1311.

Michael K Tanenhaus, Michael J Spivey-Knowlton, Kathleen M Eberhard, and Julie C E Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Shravan Vasishth, Daniela Mertzen, Lena A Jäger, and Andrew Gelman. 2018. The statistical significance filter leads to overoptimistic expectations of replicability. *Journal of Memory and Language*, 103:151–175.

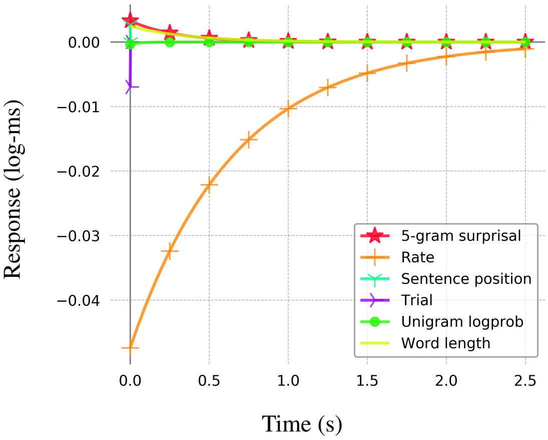


Figure 1: Natural Stories IRFs

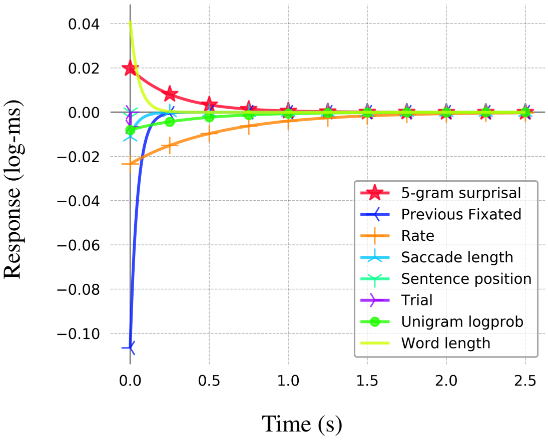


Figure 2: Dundee IRFs

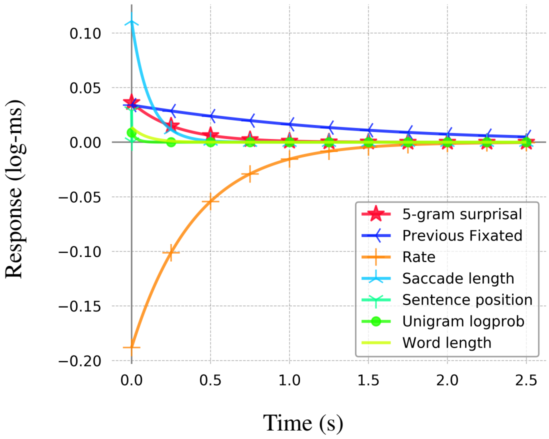


Figure 3: UCL IRFs

A DTSR Implementation

The deconvolutional time series regression (DTSR) models used in this paper were fitted using the code repository released by Shain and Schuler (2018), available at: `https://github.com/coryshain/dtsr`. Models used variational inference to fit the means and variances of independent normal posterior distributions over all model parameters assuming an improper uniform prior. Convolved predictors used the three-parameter ShiftedGamma impulse response function (IRF) kernel:

$$f(x; \alpha, \beta, \delta) = \frac{\beta^{\alpha}(x-\delta)^{\alpha-1}e^{-\beta(x-\delta)}}{\Gamma(\alpha)} \tag{1}$$

Posterior means for the IRF parameters were initialized at $\alpha = 2$, $\beta = 5$, and $\delta = -0.2$, which defines a decreasing IRF with peak centered at $t = 0$ that decays to near-zero within about 1s. Models were fitted using the Adam optimizer (Kingma and Ba, 2014) with Nesterov momentum (Nesterov, 1983; Dozat, 2016), a constant learning rate of 0.01, and minibatches of size 1024. For computational efficiency, histories were truncated at 128 timesteps. Prediction from the network used an exponential moving average of parameter iterates with a decay rate of 0.999, and models were evaluated using *maximum a posteriori* estimates obtained by setting all parameters to their posterior means.[8] Convergence was visually diagnosed.

---

[8] Since all parameters have independent normal distributions in the variational posterior, the law of large numbers guarantees that samples from the posterior converge in probability to the posterior mean.

B Impulse response shapes

For reference, estimated impulse response shapes by corpus are plotted in Figures 1–3. Plotted curves describe the estimated change in the response t seconds after having observed a unit impulse of each predictor. For example, in the Dundee estimates, observing a word with one standard deviation of *5-gram surprisal* (red curve) is expected to increase reading time by about 0.04 log-ms instantaneously, and by about 0.01 log-ms at a subsequent word observed 0.5s later. Positive IRFs (curves above 0) mean that predictors are estimated to increase reading time (and, by assumption, comprehension difficulty), and negative IRFs (curves below 0) mean that predictors are estimated to decrease reading time. For more detailed psycholinguistic interpretation of IRF estimates like these, see Shain and Schuler (2018).

Augmenting word2vec with Latent Dirichlet Allocation within a Clinical Application

Akshay Budhkar
University of Toronto;
Vector Institute for Artificial Intelligence
abudhkar@cs.toronto.edu

Frank Rudzicz
University of Toronto;
St Michael's Hospital;
Surgical Safety Technologies Inc.;
Vector Institute for Artificial Intelligence
frank@cs.toronto.edu

Abstract

This paper presents three hybrid models that directly combine latent Dirichlet allocation and word embedding for distinguishing between speakers with and without Alzheimer's disease from transcripts of picture descriptions. Two of our models get F-scores over the current state-of-the-art using automatic methods on the DementiaBank dataset.

1 Introduction

Word embedding projects words into a lower-dimensional latent space that captures semantic and morphological information. Separately but related, the task of topic modelling also discovers latent semantic structures or *topics* in a corpus. Latent Dirichlet allocation (LDA) uses bag-of-words statistics to infer topics in an unsupervised manner. LDA considers each document to be a probability distribution over hidden topics, and each topic is a probability distribution over all words in the vocabulary, both with Dirichlet priors.

The inferred probabilities over learned latent topics of a given document (i.e., *topic vectors*) can be used along with a discriminative classifier, as in the work by Luo and Li (2014), but other approaches such as TF-IDF (Lan et al., 2005) easily outperform this model, like in the case of the *Reuters-21578* corpus (Lewis et al., 1987). Here, we hypothesize that creating a hybrid of LDA and word2vec (Mikolov et al., 2013b) models will produce discriminative features. We introduce three new variants of hybrid LDA-word2vec models, and investigate the effect of dropping the first component after principal component analysis (PCA). These models can be thought of as extending the conglomeration of topical embedding models. We incorporate topical information into our word2vec models by using the final state of the topic-word distribution in the LDA model during training.

1.1 Motivation and Related Work

Alzheimer's disease (AD) is a neurodegenerative disease that affects approximately 5.5 million Americans with annual costs of care up to $259B in the US, in 2017, alone (Alzheimer's Association et al., 2017). The existing state-of-the-art methods for detecting AD from speech used extensive feature engineering, some of which involved experienced clinicians. Fraser et al. (2016) investigated multiple linguistic and acoustic characteristics and obtained accuracies up to 81% with aggressive feature selection. Yancheva and Rudzicz (2016) use vector-space topic models, and achieved F-scores up to 74%. It is generally expensive to get sufficient labeled data for arbitrary pathological conditions.

In our experiments, we train our hybrid models on a normative dataset and apply them for classification on a clinical dataset. The goal of this project is to i) effectively augment word2vec with LDA for classification, and ii) to improve the accuracy of dementia detection using automatic methods.

2 Datasets

The **Wisconsin Longitudinal Study** (WLS) is a normative dataset where residents of Wisconsin perform the Cookie Theft picture description task (Goodglass and Barresi, 2000). The audio excerpts from the 2011 survey *(N = 1,366)* were converted to text using the Kaldi open source automatic speech recognition (ASR) engine (Povey et al., 2011), specifically using a bi-directional LSTM trained to the Fisher data set (Cieri et al., 2004).

DementiaBank (DB) is part of the TalkBank project (MacWhinney et al., 2011). Each participant was assigned to either the 'Dementia' group ($N = 167$) or the 'Control' group ($N = 97$). We use 240 samples from those in the 'Dementia'

Proceedings of NAACL-HLT 2019, pages 4095–4099
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Total
CT	55	56	40	40	50	241
AD	56	54	70	70	60	310

Table 1: DB test-data distribution

group, and 233 from those in the ‘Control’ group. Each speech sample was recorded and manually transcribed at the word level following the CHAT protocol (MacWhinney, 1992). We use a 5−fold group cross-validation (CV) to split this dataset while ensuring that a particular participant does not occur in both the train and test splits. Table 1 presents the distribution of Control and Dementia groups in the test split for each fold.

WLS is used to train our LDA, word2vec and hybrid models that are then used to generate feature vectors on the DB dataset. The feature vectors on the train set are used to train a discriminative classifier (e.g., SVM), that is then used to do the AD/CT binary classification on the feature vectors of the test set. During training we filter out spaCy’s (Honnibal and Montani, 2017) list of stop words from our datasets. For our LDA models trained on ASR transcripts, we remove the *[UNK]* and *[NOISE]* tokens generated by Kaldi, as well as the *um* and *uh* tokens, as this improved downstream model performance.

3 Methods

3.1 Baselines

Once an **LDA** model is trained, it can be used to infer the topic distribution on a given document. We set the number of topics empirically to *K=5* and *K=25*.

We use a pre-trained **word2vec** model trained on the Google News Dataset (Mikolov et al., 2013a). We also train our own word vectors with 300 dimensions and window size of 2 to be consistent with the pre-trained variant. We represent a document by averaging the word embeddings for all the words in that document.

Third, **TF-IDF** is a common numerical statistic in information retrieval that measures the number of times a word occurs in a document, and through the entire corpus. We use a TF-IDF vector representation for each transcript for the top *1,000* words after preprocessing, learned on the train set.

Finally, since the goal of this paper is to create a hybrid of LDA and word2vec models, one of the simpler hybrid models – i.e., **concatenating** LDA probabilities with average word2vec representations – is the fourth baseline model.

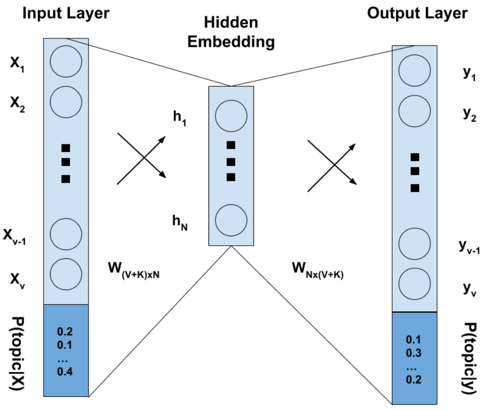


Figure 1: Neural representation of topical word2vec

3.2 Topic Vectors

We represent a topic vector as the *weighted* combination of the word2vec vectors of the words in the vocabulary. This represents every inferred *topic* as a real-valued vector, with the same dimensions as the word embedding. A topic vector for a given topic is defined as:

$$topic\_vector_D = \frac{\sum_{i=1}^{V} p_i W_i}{V} \tag{1}$$

where V is the vocabulary size of our corpus, p_i is the probability that a given word appears in the topic, from LDA, and W_i is the word2vec embedding of that word.

A document vector is given by:

$$avg\_topic\_vector_D = \frac{\sum_{i=1}^{K} p_i T_i}{K} \tag{2}$$

where T_i is the topic vector defined in Equation 1, K is the number of topics of the LDA model, and p_i is the inferred probability that a given document contains topic i.

3.3 Topical Embedding

To generate topical embeddings, we use the $P(word\,|\,topic)$ from LDA training as the ground truth of how words and topics are related to each other. We normalize that distribution, so that $\sum_{topics} P(topic\,|\,word) = 1$. This gives a topical representation for every word in the vocabulary.

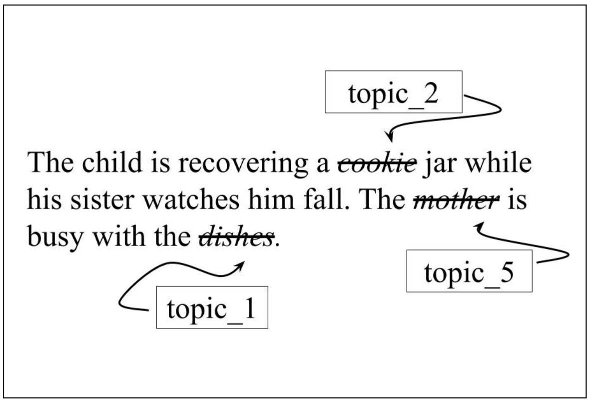


Figure 2: Example topic induction in the WLS corpus.

We concatenate this representation to the one-hot encoding of a given word to train a skip-gram word2vec model. Figure 1 shows a single pass of the word2vec training with this added information. There, X and Y are the concatenated representations of the input-output words determined by a context window, and h is an N-dimensional hidden layer. All the words and the topics are mapped to an N-dimensional embedding during inference. Our algorithm also skips the softmax layer at the output of a standard word2vec model, as our vectors are now a combination of one-hot encoding and dense probability matrices.

To get document representations, we use the average these modified word2vec embeddings. We also propose a new way of representing documents as seen in Figure 3 where we concatenate the average word2vec with the word2vec representation of the most prevalent topic in the document following LDA inference.

3.4 Topic-induced word2vec

Our final model involves inducing topics into the corpus itself. We represent every topic with the string *topic_i* where i is its topic number; e.g., topic 1 is *topic_1*, and topic 25 is *topic_25*. We also create a *sunk* topic character (analogous to *UNK* in vocabulary space) and set it to *topic_(K+1)*, where K is the number of topics in the LDA model.

We normalize $P(word\,|\,topic)$ to get $P(topic\,|\,word)$ (Section 3.3). With a probability of 0.5, set empirically, we replace a given word with the topic string for $\max(P(topic\,|\,word))$, provided the max value is ≥ 0.2. If this max value is < 0.2, the word is replaced with the sunk topic for that model.

Figure 2 shows an example of topic induction on a snapshot of an ASR transcript of WLS. This process is repeated $N = 10$ times and this augmented corpus is now run through a standard skip-gram word2vec model with dimensions set to 400 to accommodate the bigger corpus. The intuition behind this approach is that it allows words to learn how they occur around *topics* in a corpus and vice versa. Document representations follow the same format as in Section 3.3 and in Figure 3.

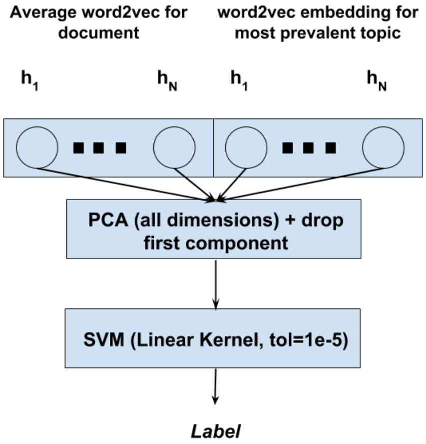


Figure 3: Setup for classification using hybrid models. The PCA step exists for models applying work described in Section 3.4.1.

3.4.1 PCA Update

Inspired by the work of Arora et al. (2016), we transform the features of our models with PCA, drop the first component, and input the result to the classifier, as it improves accuracy empirically.

Apart from the ablation study, all experiments use an SVM classifier with a linear kernel and tolerance set to 10^{-5}.

4 Results

4.1 DB Classification

We report the average of the F1 micro and F1 macro scores for the 5-folds for all baseline and proposed models. These results are presented in two parts in Tables 2 and 3.

The TF-IDF model sets a very strong baseline with an accuracy of 74.95%, which is already better than the automatic models of Yancheva and Rudzicz (2016) on the same data.

The 25-topic topical embedding model outperforms the TF-IDF baseline and gives accuracies of 75.32% when using the average word2vec approach. All topic-induced models beat the topical

	LDA		**Pre-trained word2vec**		**Trained word2vec**		**TF-IDF**	**Concatenation**	**Topic Vectors**
	5 Topics	25 Topics		PCA Update		PCA Update			
F1 micro	55.70%	62.78%	66.97%	67.34%	71.50%	72.60%	**74.95%**	**74.22%**	56.27%
F1 macro	54.44%	62.46%	64.78%	65.10%	71.33%	72.25%	**74.49%**	**73.90%**	35.90%

Table 2: DB Classification results (Average 5-Fold F-scores): Part 1

	Topical word2vec	**Topical word2vec + topic**	**Topical word2vec**	**Topical word2vec + topic**	**Topic-Induced word2vec**	**Topic-Induced word2vec + topic**	**Topic-Induced word2vec**	**Topic-Induced word2vec + topic**	**Topic-Induced word2vec**	**Topic-Induced word2vec + topic**	**Topic-Induced word2vec**	**Topic-Induced word2vec + topic**
	25 topics		25 topics and PCA		5 topics		25 topics		5 topics and PCA		25 topics and PCA	
F1 micro	75.32%	75.32%	73.69%	71.14%	75.32%	75.68%	**77.50%**	76.40%	**77.10%**	74.59%	**76.77%**	75.31%
F1 macro	74.97%	75.01%	73.32%	70.70%	74.98%	75.36%	**77.19%**	76.09%	**76.86%**	72.27%	**76.48%**	75%

Table 3: DB Classification results (Average 5-Fold F-scores): Part 2

embedding model, with the 25-topics variant giving an average accuracy of 77.5%.

To check if accuracies are statistically significant, we calculate our test statistic (Z) as follows:

$$Z = \frac{p_1 - p_2}{\sqrt{2\bar{p}(1-\bar{p})/n}} \tag{3}$$

where (p_1, p_2) are the proportions of samples correctly classified by the two classifiers respectively, n is the number of samples (which in our case is 551) and $\bar{p} = \frac{p_1+p_2}{2}$.

Augmenting word2vec models with topic information significantly improves accuracy in the topic-induced word2vec model ($p = 0.0227$) when compared to the vanilla-trained word2vec model. This change is not significant in the topical embedding model ($p = 0.152$).

4.2 Evaluation of Different Classifiers

Using the the 25-topic topic-induced word2vec, we consider other discriminative classifiers. As seen in Table 4, the linear SVM model gives the best accuracy of 77.5%, though all other models perform similarly, with accuracies upwards of 70%. There is no statistically significant difference between using an SVM vs. a LR ($p = 0.569$) or a gradient boosting classifier ($p = 0.094$).

Discriminative Classifier	**F1 micro**	**F1 macro**
SVM w/ linear kernel	**77.50%**	**77.19%**
Logistic Regression (LR)	76.05%	75.51%
Random Forest	71.13%	69.97%
Gradient Boosting Classifier	73.14%	72.39%

Table 4: DB: Discriminative Classifiers on Topic-induced LDA-25 model

5 Discussions

Although the topic distributions of the LDA models were not distinctive enough in themselves, they capture subtle differences between the AD and CT patients *missed* by the vanilla word2vec models. Simple concatenation of this distribution to the document increases the accuracy by 2.72% ($p = 0.31$).

Topic vectors on their own do not provide much generative potential for this clinical data set, as representing a document as a single point in space, after going through two layers of contraction, removes information relevant to classification.

Our novel topic-induced model performs the best among our proposed models, with an accuracy of 77.5% on a 5-fold split of the DB dataset. To put this in perspective, Yancheva and Rudzicz (2016)'s automatic vector-space topic models achieved 74% on the same data set, albeit with a slightly different setup. Applying PCA to the features does not have a significant trend.

6 Limitations and Future Work

Both of our proposed topic-induced and topical embedding models could benefit from using corpus-level word probability priors during normalization, and we intend on experimenting with those in the future. Work needs to be done to directly compare the performance of our models to the topical models proposed by (Liu et al., 2015), given that both kinds of models fall in the same universe. Finally, while we get promising results on a clinical application, the generalizability of these methods needs to be studied on other text classification tasks.

7 Conclusions

In this paper, we show the utility of augmenting word2vec with LDA-induced topics. We present three models, two of which outperform vanilla word2vec and LDA models for a clinical binary text classification task. Going forward, we will test this model on other tasks, diagnostic and otherwise, to see its generalizability. This can provide a starting point for clinical classification problems where labeled data may be scarce.

Acknowledgements

The Wisconsin Longitudinal Study is sponsored by the National Institute on Aging (grant numbers R01AG009775, R01AG033285, and R01AG041868), and was conducted by the University of Wisconsin. Rudzicz is an Inaugural CIFAR Chair in artificial intelligence.

References

Alzheimer's Association et al. 2017. 2017 Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 13(4):325–373.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Christopher Cieri, David Miller, and Kevin Walker. 2004. Fisher English training speech parts 1 and 2. *Philadelphia: Linguistic Data Consortium.*

Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. 2016. Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

Harold Goodglass and Barbara Barresi. 2000. *Boston diagnostic aphasia examination: Short form record booklet*. Lippincott Williams & Wilkins.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. 2005. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1032–1033. ACM.

David Lewis et al. 1987. Reuters-21578. *Test Collections*, 1.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Association for the Advancement of Artificial Intelligence*, pages 2418–2424.

Le Luo and Li Li. 2014. Defining and evaluating classification algorithm for high-dimensional data based on latent topics. *PloS one*, 9(1):e82119.

Brian MacWhinney. 1992. The CHILDES project: Tools for analyzing talk. *Child Language Teaching and Therapy*, 8(2):217–218.

Brian MacWhinney, Davida Fromm, Margaret Forbes, and Audrey Holland. 2011. AphasiaBank: Methods for studying discourse. *Aphasiology*, 25(11):1286–1307.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Maria Yancheva and Frank Rudzicz. 2016. Vector-space topic models for detecting Alzheimer's disease. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2337–2346.

On the Idiosyncrasies of the Mandarin Chinese Classifier System

Shijia Liu[ɘ] and **Hongyuan Mei**[ɘ] and **Adina Williams**[?] and **Ryan Cotterell**[ɘ, ﬁ]
[ɘ]Department of Computer Science, Johns Hopkins University
[?]Facebook Artificial Intelligence Research
[ﬁ]Department of Computer Science and Technology, University of Cambridge
`{sliu126,hmei}@jhu.edu`, `adinawilliams@fb.com`, `rdc42@cam.ac.uk`

Abstract

While idiosyncrasies of the Chinese classifier system have been a richly studied topic among linguists (Adams and Conklin, 1973; Erbaugh, 1986; Lakoff, 1986), not much work has been done to quantify them with statistical methods. In this paper, we introduce an information-theoretic approach to measuring idiosyncrasy; we examine how much the uncertainty in Mandarin Chinese classifiers can be reduced by knowing semantic information about the nouns that the classifiers modify. Using the empirical distribution of classifiers from the parsed Chinese Gigaword corpus (Graff et al., 2005), we compute the mutual information (in bits) between the distribution over classifiers and distributions over other linguistic quantities. We investigate whether semantic classes of nouns and adjectives differ in how much they reduce uncertainty in classifier choice, and find that it is not fully idiosyncratic; while there are no obvious trends for the majority of semantic classes, shape nouns reduce uncertainty in classifier choice the most.

1 Introduction

Many of the world's languages make use of **numeral classifiers** (Aikhenvald, 2000). While theoretical debate still rages on the function of numeral classifiers (Krifka, 1995; Ahrens and Huang, 1996; Cheng et al., 1998; Chierchia, 1998; Li, 2000; Nisbett, 2004; Bale and Coon, 2014), it is generally accepted that they need to be present for nouns to be modified by numerals, quantifiers, demonstratives, or other qualifiers (Li and Thompson, 1981, 104). In Mandarin Chinese, for instance, the phrase *one person* translates as 一个人 (*yī gè rén*); the classifier 个 (*gè*) has no clear translation in English, yet, nevertheless, it is necessary to place it between the numeral 一 (*yī*) and the word for person 人 (*rén*).

There are hundreds of numeral classifiers in the Mandarin lexicon (Po-Ching and Rimmington, 2015, Table 1 gives some canonical examples), and classifier choices are often argued to be based on inherent, possibly universal, semantic properties associated with the noun, such as shape (Kuo and Sera, 2009; Zhang and Jiang, 2016). Indeed, in a summary article, Tai (1994) writes: "Chinese classifier systems are cognitively based, rather than arbitrary systems of classification." If classifier choice were solely based on conceptual features of a given noun, then we might expect it to be nearly determinate—like gender-marking on nouns in German, Slavic, or Romance languages (Erbaugh, 1986, 400)—and perhaps even fixed for all of a given noun's synonyms.

Classifier	**Pīnyīn**	**Semantic Class**
个	*gè*	objects, general-purpose
件	*jiàn*	matters
头	*tóu*	domesticated animals
只	*zhī*	general animals
张	*zhāng*	flat objects
条	*tiáo*	long, narrow objects
项	*xiàng*	items, projects
道	*dào*	orders, roads, projections
匹	*pǐ*	horses, cloth
顿	*dùn*	meals

Table 1: Examples of Mandarin classifiers. Classifiers' simplified Mandarin Chinese characters (1st column), *pīnyīn* pronunciations (2nd column), and commonly modified noun types (3rd column) are listed.

However, selecting which classifers go with nouns in practice is an idiosyncratic process, often with several grammatical options (see Table 2 for two examples). Moreover, knowing what a noun means doesn't always mean you can guess the classifier. For example, most nouns referring to animals, such as 驴 (*lǘ*, *donkey*) or 羊 (*yáng*, *goat*), use the classifier 只 (*zhī*). However, horses

Proceedings of NAACL-HLT 2019, pages 4100–4106
Minneapolis, Minnesota, June 2 - June 7, 2019.

Classifier	$p(C \mid N = \text{人士})$	$p(C \mid N = \text{工程})$
位 (*wèi*)	**0.4838**	0.0058
名 (*míng*)	**0.3586**	0.0088
个 (*gè*)	0.0205	0.0486
批 (*pī*)	0.0128	0.0060
项 (*xiàng*)	0.0063	**0.4077**
期 (*qī*)	0.0002	**0.2570**
Everything else	0.1178	0.2661

Table 2: Empirical distribution of selected classifiers over two nouns: 人士 (*rén shì, people*) and 工程 (*gōng chéng, project*). Most of the probability mass is allocated to only a few classifiers for both nouns (bolded).

cannot use 只 (*zhī*), despite being semantically similar to goats and donkeys, and instead must appear with 匹 (*pǐ*). Conversely, knowing which particular subset of noun meaning is reflected in the classifer also doesn't mean that you can use that classifier with any noun that seems to have the right semantics. For example, classifier 条 (*tiáo*) can be used with nouns referring to long and narrow objects, like rivers, snakes, fish, pants, and certain breeds of dogs—but never cats, regardless of how long and narrow they might be! In general, classifiers carve up the semantic space in a very idiosyncratic manner that is neither fully arbitrary, nor fully predictable from semantic features.

Given this, we can ask: precisely *how* idiosyncratic is the Mandarin Chinese classifier system? For a given noun, how predictable is the set of classifiers that can be grammatically employed? For instance, had we not known that the Mandarin word for horse 马 (*mǎ*) predominantly takes the classifier 匹 (*pǐ*), how likely would we have been to guess it over the much more common animal classifier 只 (*zhī*)? Is it more important to know that a noun is 马 (*mǎ*) or simply that the noun is an animal noun? We address these questions by computing how much the uncertainty in the distribution over classifiers can be reduced by knowing information about nouns and noun semantics. We quantify this notion of classifier idiosyncrasy by calculating the mutual information between classifiers and nouns, and also between classifiers and several sets that are relevant to noun meaning (i.e., categories of noun senses, sets of noun synonyms, adjectives, and categories of adjective senses). Our results yield concrete, quantitative measures of idiosyncrasy in bits, that can supplement existing hand-annotated, intuition-based approaches that organize Mandarin classifiers into an ontology.

Why investigate the idiosyncrasy of the Mandarin Chinese classifier system? How idiosyncratic or predictable natural language is has captivated researchers since Shannon (1951) originally proposed the question in the context of printed English text. Indeed, looking at predictability directly relates to the complexity of language—a fundamental question in linguistics (Newmeyer and Preston, 2014; Dingemanse et al., 2015)—which has also been claimed to have consequences learnability and processing. For example, how hard it is for a learner to master irregularity, say, in the English past tense (Rumelhart and McClelland, 1987; Pinker and Prince, 1994; Pinker and Ullman, 2002; Kirov and Cotterell, 2018) might be affected by predictability, and highly predictable noun-adjacent words, such as gender affixes in German and prenominal adjectives in English, are also shown to confer online processing advantages (Dye et al., 2016, 2017, 2018). Within the Chinese classifier system itself, the very common, general-purpose classifier 个 (*gè*) is acquired by children earlier than rarer, more semantically rich ones (Hu, 1993). General classifiers are also found to occur more often in corpora with nouns that are less predictable in context (i.e., nouns with high *surprisal*; Hale 2001) (Zhan and Levy, 2018), providing initial evidence that predictability likely plays a role in classifier-noun pairing more generally. Furthermore, providing classifiers improves participants' recall of nouns in laboratory experiments (Zhang and Schmitt, 1998; Gao and Malt, 2009) (but see Huang and Chen 2014)—but, it isn't known whether classifiers do so by modulating noun predictability.

2 Quantifying Classifier Idiosyncrasy

We take an information-theoretic approach to statistically quantify the idiosyncrasy of the Mandarin Chinese classifier system, and measure the uncertainty (entropy) reduction—or **mutual information** (MI) (Cover and Thomas, 2012)—between classifiers and other linguistic quantities, like nouns or adjectives. Intuitively, MI lets us directly measure classifier "idiosyncrasy", because it tells us how much information (in bits) about a classifier we can get by observing another linguistic quantity. If classifiers were completely independent from other quantities, knowing them would give us no information about classifier choice.

Notation. Let C be a classifier-valued random variable with range $\mathcal{C}$, the set of Mandarin Chinese classifiers. Let X be a second random variable, which models a second linguistic quantity, with range $\mathcal{X}$. Mutual information (MI) is defined as

$$I(C;X) \equiv H(C) - H(C \mid X) \tag{1a}$$

$$= \sum_{c \in \mathcal{C}, x \in \mathcal{X}} p(c,x) \log \frac{p(c,x)}{p(c)p(x)} \tag{1b}$$

Let $\mathcal{N}$ and $\mathcal{A}$ denote the sets of nouns and adjectives, respectively, with N and A be noun- and adjective-valued random variables, respectively. Let $\mathcal{N}_i$ and $\mathcal{A}_i$ denote the sets of nouns and adjectives in i^{th} *SemCor supersense category* for nouns (Tsvetkov et al., 2015) and adjectives (Tsvetkov et al., 2014), respectively, with their random variables being N_i and A_i, respectively. Let $\mathcal{S}$ be the set of all English WordNet (Miller, 1998) senses of nouns, with S be the WordNet sense-valued random variable. Given the formula above and any choice for $X \in \{N, A, N_i, A_i, S\}$, we can calculate the mutual information between classifiers and other relevant linguistic quantities.

2.1 MI between Classifiers and Nouns

Mutual information between classifiers and nouns ($I(C;N)$) shows how much uncertainty (i.e., entropy) in classifiers can be reduced once we know the noun, and vice versa. Because only a few classifiers are suitable to modify a given noun (again, see Table 2) and the entropy of classifiers for a given noun is predicted to be close to zero, MI between classifiers and nouns is expected to be high.

2.2 MI Between Classifiers and Adjectives

The distribution over adjectives that modify a noun in a large corpus give us a language-internal peek into a word's lexical semantics. Moreover, adjectives have been found to increase the predictability of nouns (Dye et al., 2018), so we ask whether they might affect classifier predictability too. We compute MI between classifiers and adjectives ($I(C;A)$) that modify the same nouns to investigate their relationship. If both adjectives and classifiers track comparable portions of the noun's semantics, we expect $I(C;A)$ to be significantly greater than zero, which implies mutual dependence between classifier C and adjective A.

2.3 MI between Classifiers and Noun Supersenses

To uncover which nouns are able to reduce the uncertainty in classifiers more, we divide them into 26 *SemCor supersense categories* (Tsvetkov et al., 2015), and then compute $I(C;N_i)$ ($i \in \{1, 2, ..., 26\}$) for each supersense category. The supersense categories (e.g., *animal, plant, person, artifact*, etc.) provide a semantic classification system for English nouns. Since there are no known supersense categories for Mandarin, we need to translate Chinese nouns into English to perform our analysis. We use *SemCor supersense categories* instead of WordNet hypernyms because different "basic levels" for each noun make it difficult to determine the "correct" category for each noun.

2.4 MI between Classifiers and Adjective Supersenses

We translated and divided the adjectives into 12 *supersense categories* (Tsvetkov et al., 2014), and compute mutual information $I(C;A_i)$ ($i \in \{1, 2, ..., 12\}$) for each category separately to determine which categories have more mutual dependence on classifiers. Adjective supersenses are defined as categories describing certain properties of nouns. For example, adjectives in MIND category describe intelligence and awareness, while those in the PERCEPTION category focus on, e.g., color, brightness, and taste. Examining the distribution over adjectives is a language-specific measure of *noun* meaning, albeit an imperfect one, because only certain adjectives modify any given noun.

2.5 MI between Classifiers and Noun Synsets

We also compute the mutual information $I(C;S)$ between classifiers and nouns' WordNet (Miller, 1998) synonym sets (**synsets**), assuming that each synset is independent. For nouns with multiple synsets, we assume that all synsets are equally probable for simplicity. If classifiers are fully semantically determined, then knowing a noun's synsets should enable one to know the appropriate classifier(s), resulting in high MI. If classifiers are largely idiosyncratic, then noun synsets should have lower MI with classifiers. We do not use WordNet to attempt to capture word polysemy here.

3 Data and Experiments

Data Provenance. We apply an existing neural Mandarin word segmenter (Cai et al., 2017) to the Chinese Gigaword corpus (Graff et al., 2005), and then feed the segmented corpus to a neural dependency parser, using Google's pretrained Parsey Uni-

$H(C)$	$H(C \mid N)$	$I(C;N)$	$H(C \mid S)$	$I(C;S)$	$H(C \mid A)$	$I(C;A)$
5.61	0.66	4.95	4.14	1.47	3.53	2.08

Table 3: Mutual information between classifiers and nouns $I(C;N)$, noun senses $I(C;S)$, and adjectives $I(C;A)$, is compared to their entropies.

versal model on Mandarin.[1] The model is trained on Universal Dependencies datasets v1.3.[2] We extract classifier-noun pairs and adjective-classifier-noun triples from sentences, where the adjective and the classifier modify the same noun—this is easily determined from the parse. We also record the tuple counts, and use them to compute an empirical distribution over classifiers that modify nouns, and noun-adjective pairs, respectively.

Data Preprocessing. Since no annotated supersense list exists for Mandarin, we first use CC-CEDICT[3] as a Mandarin Chinese-to-English dictionary to translate nouns and adjectives into English. Acknowledging that translating might introduce noise, we subsequently categorize our words into different senses using the SemCor supersense data for nouns (Miller et al., 1993; Tsvetkov et al., 2015), and adjectives (Tsvetkov et al., 2014). After that, we calculate the mutual information under each noun, and adjective supersense.

Modeling Assumptions. As this contribution is the first to investigate classifier predictability, we make several simplifying assumptions. Extracting distributions over classifiers from a large corpus, as we do, *ignores* sentential context, which means we ignore the fact that some nouns (i.e., relational nouns, like *māmā*, *Mom*) are more likely to be found in verb frames or other constructions where classifiers are not needed. We also ignore singular-plural, which might affect classifier choice, and the mass-count distinction (Cheng et al., 1998; Bale and Coon, 2014), to the extent that it is not encoded in noun superset categories (e.g., *substance* includes mass nouns).

We also assume that every classifier-noun or classifier-adjective pairing we extract is equally acceptable to native speakers. However, it's possible that native speakers differ in either their knowledge of classifier-noun distributions or confidence in particular combinations. Whether and how such human knowledge interacts with our calculations would be an interesting future avenue.

4 Results and Analysis

4.1 MI between Classifiers and Nouns, Noun Synsets, and Adjectives

Table 3 shows MI between classifiers and other linguistic quantities. As we can see, $I(C;N) > I(C;A) > I(C;S)$. As expected, knowing the noun greatly reduces classifier uncertainty; the noun and classifier have high MI (4.95 bits). Classifier MI with noun synsets (1.47 bits) is not comparable to with nouns (4.95 bits), suggesting that knowing a synset does not greatly reduce classifier uncertainty, leaving $>{}^{3}\!/_{5}$ of the entropy unaccounted for. We also see that adjectives (2.08 bits) reduce the uncertainty in classifiers more than noun synsets (1.47 bits), but less than nouns (4.95 bits).

4.2 MI between Classifiers and Noun Supersenses

Noun supersense results are in Figure 1. Natural categories are helpful, but are far from completely predictive of the classifier distribution: knowing that a noun is a *plant* helps, but cannot account for about ${}^{1}\!/_{3}$ of the original entropy for the distribution over classifiers, and knowing that a noun is a *location* leaves $>{}^{1}\!/_{2}$ unexplained. The three supersenses with highest $I(C;N)$ are *body*, *artifact*, and *shape*. Of particular interest is the *shape* category. Knowing that a noun refers to a shape (e.g., 角度; *jiǎo dù*, *angle*), makes the choice of classifier relatively predictable. This result sheds new light on psychological findings that Mandarin speakers are more likely to classify words as similar based on shape than English speakers (Kuo and Sera, 2009), by uncovering a possible role for information structure in shape-based choice. It also accords with Chien et al. (2003) and Li et al. (2010, 216) that show that children as young as three know classifiers often delineate categories of objects with similar shapes.

---

[1] `https://github.com/tensorflow/models/blob/master/research/syntaxnet/g3doc/universal.md`

[2] `https://universaldependencies.org/`

[3] `www.mdbg.net/chinese/dictionary`

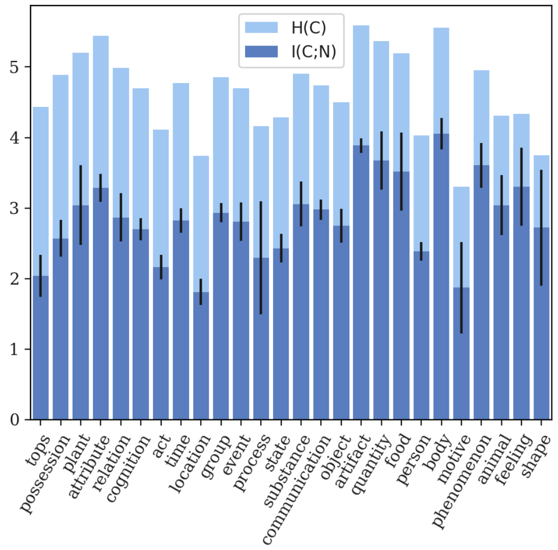


Figure 1: Mutual information between classifiers and nouns (dark blue), and classifier entropy (light & dark blue) plotted with $H(C \mid N) = H(C) - I(C;N)$ (light blue) decreasing from left. Error bars denote bootstrapped 95% confidence interval of $I(C;N)$.

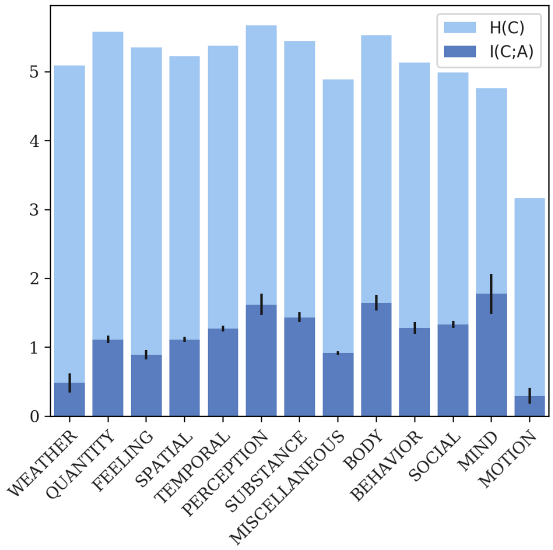


Figure 2: Mutual information between classifiers and adjectives (dark blue), and classifier entropy (light & dark blue) plotted with $H(C \mid A) = H(C) - I(C;A)$ (light blue) decreasing from left. Error bars denote bootstrapped 95% confidence interval of $I(C;A)$.

4.3 MI between Classifiers and Adjectives Supersenses

Adjective supersense results are in Figure 2. Interestingly, the top three senses that have the highest mutual information between their adjectives and classifiers—MIND, BODY (constitution, appearance) and PERCEPTION—are all involved with people's subjective views. With respect to Kuo and Sera (2009), adjectives from the SPATIAL sense pick out shape nouns in our results. Although it does not make it into the top three, MI for the SPATIAL sense is still significant.

5 Conclusion

While classifier choice is known to be idiosyncratic, no extant study has precisely quantified this. To do so, we measure the mutual information between classifiers and other linguistic quantities, and find that classifiers are highly mutually dependent on nouns, but are less mutually dependent on adjectives and noun synsets. Furthermore, knowing which noun or adjective supersense a word comes from helps, often significantly, but still leaves much of the original entropy in the classifier distribution unexplained, providing quantitative support for the notion that classifier choice is largely idiosyncratic. Although the amount of mutual dependence is highly variable across the semantic classes we investigate, we find that knowing a noun refers to a shape reduces uncertainty in classifier choice more than knowing it falls into any other semantic class, arguing for a role for information structure in Mandarin speakers' reliance on shape (Kuo and Sera, 2009). This result might have implications for second language pedagogy, adducing additional, tentative evidence in favor of collocational approaches to teaching classifiers (Zhang and Lu, 2013) that encourages memorizing classifiers and nouns together. Investigating classifiers might also provide cognitive scientific insights into conceptual categorization (Lakoff, 1986)—often considered crucial for language use (Ungerer, 1996; Taylor, 2002; Croft and Cruse, 2004). Studies like this one opens up avenues for comparisons with other phenomena long argued to be idiosyncratic, such as grammatical gender, or declension class.

Acknowledgments

This research benefited from generous financial support from a Bloomberg Data Science Fellowship to Hongyuan Mei and a Facebook Fellowship to Ryan Cotterell. We thank Meilin Zhan, Roger Levy, Géraldine Walther, Arya McCarthy, Ekaterina Vylomova, Sebastian Mielke, Katharina Kann, Jason Eisner, Jacob Eisenstein, and anonymous reviewers (NAACL 2019) for their comments.

References

Karen L. Adams and Nancy F. Conklin. 1973. Toward a theory of natural classification. In *Paper from the Ninth Regional Meeting of the Chicago Linguistic Society*, pages 1–10.

Kathleen Ahrens and Chu-Ren Huang. 1996. Classifiers and semantic type coercion: Motivating a new classification of classifiers. In *Proceedings of the 11th Pacific Asia Conference on Language, Information and Computation*, pages 1–10.

Alexandra Y. Aikhenvald. 2000. *Classifiers: a Typology of Noun Categorization Devices*. Oxford University Press, Oxford.

Alan Bale and Jessica Coon. 2014. Classifiers are for numerals, not for nouns: Consequences for the mass/count distinction. *Linguistic Inquiry*, 45(4):695–707.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 608–615, Vancouver, Canada.

Lisa L.S. Cheng, Rint Sybesma, et al. 1998. Yi-wan tang, yi-ge tang: Classifiers and massifiers. *Tsing Hua journal of Chinese studies*, 28(3):385–412.

Yu-Chin Chien, Barbara Lust, and Chi-Pang Chiang. 2003. Chinese children's comprehension of count-classifiers and mass-classifiers. *Journal of East Asian Linguistics*, 12(2):91–120.

Gennaro Chierchia. 1998. Reference to kinds across language. *Natural Language Semantics*, 6(4):339–405.

Thomas M. Cover and Joy A. Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.

William Croft and D. Alan Cruse. 2004. *Cognitive linguistics*. Cambridge University Press.

Mark Dingemanse, Damián E. Blasi, Gary Lupyan, Morten H. Christiansen, and Padraic Monaghan. 2015. Arbitrariness, iconicity, and systematicity in language. *Trends in Cognitive Sciences*, 19(10):603–615.

Melody Dye, Petar Milin, Richard Futrell, and Michael Ramscar. 2016. A functional theory of gender paradigms. *Morphological paradigms and functions. Leiden: Brill.*

Melody Dye, Petar Milin, Richard Futrell, and Michael Ramscar. 2017. Cute little puppies and nice cold beers: An information theoretic analysis of prenominal adjectives. In *39th Annual Meeting of the Cognitive Science Society, London, UK.*

Melody Dye, Petar Milin, Richard Futrell, and Michael Ramscar. 2018. Alternative solutions to a language design problem: The role of adjectives and gender marking in efficient communication. *Topics in cognitive science*, 10(1):209–224.

Mary S. Erbaugh. 1986. Taking stock: The development of Chinese noun classifiers historically and in young children. *Noun Classes and Categorization*, pages 399–436.

Ming Y. Gao and Barbara C. Malt. 2009. Mental representation and cognitive consequences of Chinese individual classifiers. *Language and Cognitive Processes*, 24(7-8):1124–1179.

David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2005. *Chinese Gigaword Second Edition LDC2005T14. Web Download.* Philadelphia: Linguistic Data Consortium.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Qian Hu. 1993. *The Acquisition of Chinese Classifiers by Young Mandarin-Speaking Children*. Ph.D. thesis, Boston University.

Shuping Huang and Jenn-Yeu Chen. 2014. The effects of numeral classifiers and taxonomic categories on Chinese and English speakers' recall of nouns. *Journal of East Asian Linguistics*, 23(1):27–42.

Christo Kirov and Ryan Cotterell. 2018. Recurrent neural networks in linguistic theory: Revisiting pinker and prince (1988) and the past tense debate. *arXiv preprint arXiv:1807.04783*.

Manfred Krifka. 1995. *Common nouns: A contrastive analysis of English and Chinese*, page 398–411. Chicago: University of Chicago Press.

Jenny Y. Kuo and Maria D. Sera. 2009. Classifier effects on human categorization: the role of shape classifiers in Mandarin Chinese. *Journal of East Asian Linguistics*, 18:1–19.

George Lakoff. 1986. Classifiers as a reflection of mind. *Noun classes and categorization*, 7:13–51.

Charles N. Li and Sandra A Thompson. 1981. *A grammar of spoken Chinese: A functional reference grammar*. Berkeley: University of California Press.

Peggy Li, Becky Huang, and Yaling Hsiao. 2010. Learning that classifiers count: Mandarin-speaking children's acquisition of sortal and mensural classifiers. *Journal of East Asian Linguistics*, 19(3):207–230.

Wendan Li. 2000. The pragmatic function of numeral-classifiers in Mandarin Chinese. *Journal of Pragmatics*, 32(8):1113–1133.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT Press.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics.

Frederick J. Newmeyer and Laurel B. Preston. 2014. *Measuring grammatical complexity*. Oxford University Press.

Richard Nisbett. 2004. *The geography of thought: How Asians and Westerners think differently . . . and why*. Simon and Schuster.

Steven Pinker and Alan Prince. 1994. Regular and irregular morphology and the psychological status of rules of grammar. *The Reality of Linguistic Rules*, 321:51.

Steven Pinker and Michael T. Ullman. 2002. The past and future of the past tense. *Trends in cognitive sciences*, 6(11):456–463.

Yip Po-Ching and Don Rimmington. 2015. *Chinese: A comprehensive grammar*. Routledge.

David E. Rumelhart and James L. McClelland. 1987. Learning the past tenses of english verbs: Implicit rules or parallel distributed processing. *Mechanisms of language acquisition*, pages 195–248.

Claude E. Shannon. 1951. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64.

James H. Y. Tai. 1994. Chinese classifier systems and human categorization. *In honor of William S.-Y. Wang: Interdisciplinary studies on language and language change*, pages 479–494.

John R. Taylor. 2002. Cognitive grammar.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*.

Yulia Tsvetkov, Nathan Schneider, Dirk Hovy, Archna Bhatia, Manaal Faruqui, and Chris Dyer. 2014. Augmenting English adjective senses with supersenses. In *Proceedings of LREC*.

Schmid H. J. Ungerer, Friedrich. 1996. *An introduction to cognitive linguistics*. Essex: Longman.

Meilin Zhan and Roger Levy. 2018. Comparing theories of speaker choice using a model of classifier production in Mandarin Chinese. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1997–2005.

Jie Zhang and Xiaofei Lu. 2013. Variability in chinese as a foreign language learners' development of the chinese numeral classifier system. *The Modern Language Journal*, 97(S1):46–60.

Liulin Zhang and Song Jiang. 2016. A cognitive linguistics approach to Chinese classifier teaching: An experimental study. *Journal of Language Teaching and Research*, 7(3):467–475.

Shi Zhang and Bernd Schmitt. 1998. Language-dependent classification: The mental representation of classifiers in cognition, memory, and ad evaluations. *Journal of Experimental Psychology: Applied*, 4(4):375.

Joint Learning of Pre-Trained and Random Units for Domain Adaptation in Part-of-Speech Tagging

Sara Meftah[*], **Youssef Tamaazousti**[∓], **Nasredine Semmar**[*], **Hassane Essafi**[*], **Fatiha Sadat**[+]
[*]CEA, LIST, LASTI, France
[∓]MIT, CSAIL, USA
[+]UQÀM, Montréal, Canada
{firstname.lastname}@cea.fr, ytamaaz@mit.edu, sadat.fatiha@uqam.ca

Abstract

Fine-tuning neural networks is widely used to transfer valuable knowledge from high-resource to low-resource domains. In a standard fine-tuning scheme, source and target problems are trained using the same architecture. Although capable of adapting to new domains, pre-trained units struggle with learning uncommon target-specific patterns. In this paper, we propose to augment the target-network with normalised, weighted and randomly initialised units that beget a better adaptation while maintaining the valuable source knowledge. Our experiments on POS tagging of social media texts (Tweets domain) demonstrate that our method achieves state-of-the-art performances on 3 commonly used datasets.

1 Introduction

POS tagging is a sequence labelling problem, that consists on assigning to each sentence' word, its disambiguated POS tag (*e.g.*, Pronoun, Noun) in the phrasal context in which the word is used. Such information is useful for higher-level applications, such as machine-translation (Niehues and Cho, 2017) or cross-lingual information retrieval (Semmar et al., 2006, 2008).

One of the best approaches for POS tagging of social media text (Meftah et al., 2018a), is transfer-learning, which relies on a neural-network learned on a *source-dataset* with sufficient annotated data, then further adapted to the problem of interest (*target-dataset*). While this approach is known to be very effective (Zennaki et al., 2019), because it takes benefit from pre-trained neurons, it has one main drawback by design. Indeed, it has been shown in computer-vision (Zhou et al., 2018a) that, when fine-tuning on scenes a model pre-trained on objects, it is the neuron firing on the *white dog* object that became highly sensitive to the *white waterfall* scene. Simply said, pre-trained neurons are *biased* by what they have learned in the source-dataset. This is also the case on NLP (see experiments). Consequently, pre-trained units struggle with learning patterns specific to the target-dataset (*e.g.*, "wanna" or "gonna" in the Tweets domain). This last is non-desirable, since it has been shown recently (Zhou et al., 2018b) that such specific units are important for performance. To overcome this drawback, one can propose to take benefit from randomly initialised units, that are by design non-biased. However, it is common to face small target-datasets that contain too few data to learn such neurons from scratch. Hence, in such setting, it is hard to learn random units that fire on specific patterns and generalise well.

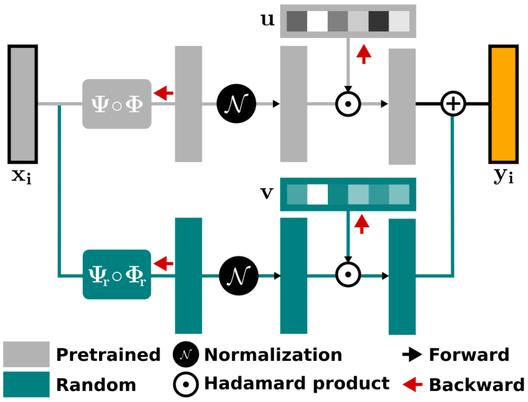


Figure 1: Given a word representation x_i, a BiLSTM (Φ) models the sequence, and a FC layer (Ψ) performs classification. In standard fine-tuning, the units are pre-trained on a large source-dataset then adapted to the target one. In this work, we propose to add randomly initialised units (green branch) and jointly adapt them with pre-trained ones (gray branch). An element-wise sum is further applied to merge the two branches. Before merging, we balance the different behaviours of pre-trained and random units, using an independent normalisation ($\mathcal{N}$). Finally we let the network learn which of pre-trained or random neurons are more suited for every class, by adding learnable weighting vectors (u and v initialised with 1-values) on the FC layers.

In this article, we propose a hybrid method that takes benefit from both worlds, without their drawbacks. It consists in augmenting the source-

Proceedings of NAACL-HLT 2019, pages 4107–4112
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

network (set of pre-trained units) with randomly initialised units and jointly learn them. We call our method **PretRand** (**Pret**rained and **Rand**om units) and illustrate it in Fig. 1. The main difficulty is forcing the network to consider random units, because they have different behaviours than pre-trained ones. Indeed, while these last strongly fire discriminatively on many words, these first do not fire on any word at the initial stage of fine-tuning. Therefore, random units do *not* significantly contribute to the computation of gradients and are thus slowly updated. To overcome this problem, we proposed to independently normalise pre-trained and random layers. This last balances their range of activations and thus forces the network to consider them, both. Last but not least, we do not know which of pre-trained and random units are the best for every class-predictor, thus we propose to learn weighting vectors on top of each branch.

Evaluation was carried on 3 POS tagging Tweets datasets in a transfer-learning setting. Our method outperforms SOTA methods and significantly surpasses fairly comparable baselines.

2 Proposed Method: PretRand

2.1 Base Model

Given an input sentence $S = [w_1, \ldots, w_n]$ of n successive tokens w_i, the goal of a POS tagger is to predict the POS-tag $c_i \in \mathcal{C}$ of every w_i, with $\mathcal{C} \in \mathbb{R}^C$ being the tag-set. Hence, for our base model, we used a common sequence labelling model which first, computes for each token w_i, a word-level embedding (denoted $\boldsymbol{\Upsilon}_\mathbf{w}$) and character-level embedding using biLSTM encoder ($\boldsymbol{\Upsilon}_\mathbf{c}$), and concatenates them to get a final representation x_i. Second, it feeds the later representation into a biLSTM features extractor (denoted $\boldsymbol{\Phi}$) that outputs a hidden representation, that is itself fed into a fully-connected (FC) layer (denoted $\boldsymbol{\Psi}$) for classification. Formally, given w_i, the logits are obtained using: $\hat{y}_{w_i} = \boldsymbol{\Psi} \circ \boldsymbol{\Phi} \circ \boldsymbol{\Upsilon}(w_i)$, with $\boldsymbol{\Upsilon}$ being the concatenation of the output of $\boldsymbol{\Upsilon}_\mathbf{c}$ and $\boldsymbol{\Upsilon}_\mathbf{w}$ for w_i. In a standard fine-tuning scheme (Meftah et al., 2018b), $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Phi}$ are pre-trained on the *source-task* and $\boldsymbol{\Psi}$ is randomly initialised. Then, the three modules are further jointly trained on the *target-task* by minimising a Softmax Cross-Entropy (SCE) loss using the SGD algorithm.

2.2 Adding Random Branch

As mentioned in the introduction, pre-trained neurons are biased by design, thus limited. This motivated our proposal to augment the pre-trained branch with additional random units (as illustrated in Fig. 1). To do so, theoretically one can add the new units in any layer of the base model. However in practice, we have to make a trade-off between performances and the number of parameters (model complexity). Thus, given that deep layers are more task-specific than shallow ones (Peters et al., 2018; Mou et al., 2016), and that word embeddings (shallow layers) contain the majority of parameters, we choose to expand only the top layers. With this choice, we desirably increase the complexity of the model only by $1.02\times$ compared to the base one. In terms of the layers expanded, we specifically add k units to $\boldsymbol{\Phi}$ resulting in an extra biLSTM layer: $\boldsymbol{\Phi}_\mathbf{r}$ (r for rand); and C units in $\boldsymbol{\Psi}$ resulting in an extra FC layer: $\boldsymbol{\Psi}_\mathbf{r}$. Hence, for every w_i, the additional random branch predicts class-probabilities following: $\hat{y}^r_{w_i} = \boldsymbol{\Psi}_\mathbf{r} \circ \boldsymbol{\Phi}_\mathbf{r}(x_i)$ (with $x_i = \boldsymbol{\Upsilon}(w_i)$). Note that, having two FC layers obviously outputs two predictions per class (one from the pre-trained FC $\hat{y}^p_{w_i}$ and one from the random $\hat{y}^r_{w_i}$), that thus need to be merged. Hence, to get the final predictions, we simply apply an element-wise sum between the output of both branches: $\hat{y}_{w_i} = \hat{y}^p_{w_i} \oplus \hat{y}^r_{w_i}$. As in the classical scheme, SCE is minimised but here, both branches are trained jointly.

2.3 Independent Normalisation

Nevertheless, while at the initial stage of fine-tuning, the pre-trained units are strongly firing on many words, the random ones are firing very weakly. As stated in some computer-vision works (Liu et al., 2015; Tamaazousti et al., 2018), the later setting causes an absorption of the weights, outputs and thus gradients of the random units by the pre-trained ones, which thus makes them useless at the end. We encountered the same problem with textual data on the POS-tagging problem. Indeed, as illustrated in the left plot of Fig.2, at the end of training, the distribution of the random units' weights is still absorbed (closer to zero) by that of the pre-trained ones.

To prompt the two classifiers to work cooperatively, we normalise (using an ℓ_p-norm) both of them independently before merging them. Formally, we apply $\boldsymbol{\mathcal{N}}_p(x) = \frac{x}{||x||_p}$ on $\hat{y}^p_{w_i}$ and $\hat{y}^r_{w_i}$. The normalisation is desirably solving the weights absorption problem since at the end of the training, the distributions of the pre-trained and random weights become very similar (right of Fig. 2).

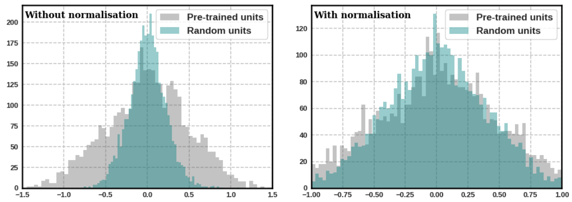


Figure 2: Distributions of learned weight-values for the randomly initialised (green) and pre-trained (gray) fully-connected layers after joint training. Left: without normalisation, right: with normalisation.

Furthermore, we have observed that despite the normalisation, the performances of the pre-trained classifiers were still much better than the randomly initialised ones. Thus, to make them more competitive, we propose to start with optimising only the randomly initialised units while freezing the pre-trained ones, then, launch the joint training. This is called random++ in the following.

2.4 Learnable Weighting Vectors

Back to the extra predictor (FC layer of random branch), it is important to note that both branches are equally important for making a decision for every class, *i.e.,* no weight is applied on the dimensions of $\hat{y}^p_{w_i}$ and $\hat{y}^r_{w_i}$. However, this latter is sub-optimal since we, a priori, do not know which kind of units (random or pre-trained) is better for making a decision. Consequently, we propose to weight the contribution of the predictions for each class. For this end, instead of simply performing an element-wise sum between the random and pre-trained predictions, we first *weight* each of them with learnable weighting vectors, then compute a Hadamard product with their associated normalised predictions; the learnable vectors $u \in \mathbb{R}^C$ and $v \in \mathbb{R}^C$, respectively corresponding to the pre-trained and random branch, are initialised with 1-values and are learned by SGD. Formally, the final predictions are computed following: $\hat{y}_{w_i} = u \odot \boldsymbol{\mathcal{N}}_p(\hat{y}^p_{w_i}) \oplus v \odot \boldsymbol{\mathcal{N}}_p(\hat{y}^r_{w_i})$.

3 Experiments

3.1 Implementation Details

In the word-level embeddings, tokens are lower-cased while the character-level component still retains access to the capitalisation information. We set the character embedding dimension at 50, the dimension of hidden states of the character-level biLSTM at 100 and used 300-dimensional word-level embeddings. The latter were pre-loaded from publicly available Glove pre-trained vectors on 42 billions words from a web crawling and containing 1.9M words (Pennington et al., 2014). Note that, these embeddings are also updated during fine-tuning. For biLSTM (token-level feature extractor), we set the number of units of the pre-trained branch to 200 and experimented our approach with k added random-units, with $k \in \{50, 100, 150, 200\}$. For the normalisation, we used ℓ_2-norm. Finally, in all experiments, training was performed using SGD with momentum and mini-batches of 8 sentences. Evidently, all the hyperparameters have been cross-validated.

Corpus	TPoS	Ark	TweeBank
Train	10,652	26,594	24,753
Dev	2,242	n/a	11,742
Test	2,291	7,707	19,112

Table 1: Number of tokens in every used dataset.

3.2 Datasets

For the source-dataset, we used the Wall Street Journal (**WSJ**) part of Penn-Tree-Bank (PTB), a large English dataset containing 1.2M+ tokens from the newswire domain annotated with the PTB tag-set. Regarding the target-datasets, we used three Tweets datasets: **TPoS** (Ritter et al., 2011), annotated with 40 tags ; **ARK** (Owoputi et al., 2013) containing 25 coarse tags; and the recent **TweeBank** (Liu et al., 2018) containing 17 tags (PTB universal tag-set). The number of tokens in the datasets are given in Table 1.

3.3 Comparison Methods

To assess the POS tagging performances of our PretRand model, we compared it to 5 baselines: **Random-200** and **Random-400**: randomly initialised neural model with 200 and 400 biLSTM's units; **Fine-tuning**: pre-trained neural model, fine-tuned with the standard scheme; **Ensemble (2 rand)**: averaging the predictions of two base models randomly initialised and learned independently (with different random initialisation) on Tweets datasets; and **Ensemble (1 pret + 1 rand)**: same as the previous but with one pre-trained on WSJ and the other randomly initialised.

We also compared it to the 3 best SOTA methods: Derczynski et al. (2013) (GATE) is a model based on HMMs with a set of normalisation rules, external dictionaries and lexical features. They experiment it on TPoS, with WSJ and 32K tokens from the NPS IRC corpus. They also used 1.5M additional training tokens annotated by vote-constrained bootstrapping (GATE-bootstrap). Owoputi et al. (2013) proposed a model based on first-order Maximum Entropy

Method	#params	TPoS		ArK	TweeBank		Avg
		Dev	*Test*	*Test*	*Dev*	*Test*	
GATE (Derczynski et al., 2013)	n/a	89.37	88.69	n/a	n/a	n/a	n/a
GATE-bootstrap (Derczynski et al., 2013)	n/a	n/a	90.54	n/a	n/a	n/a	n/a
ARK (Owoputi et al., 2013)	n/a	n/a	90.40	93.2	n/a	94.6	n/a
TPANN (Gui et al., 2017)	n/a	91.08	90.92	92.8	n/a	n/a	n/a
Random-200	1×	88.32	87.76	90.67	91.20	91.56	89.90
Random-400	1.03×	89.01	88.89	90.99	91.38	91.63	90.38
Standard fine-tuning	1×	90.96	90.7	91.72	92.59	92.99	91.79
Ensemble Model (2 rand)	2×	89.20	88.8	91.36	91.73	92.05	90.62
Ensemble Model (1 pret + 1 rand)	2×	89.77	88.61	91.41	92.57	92.85	91.04
PretRand (Ours)	1.02×	**91.56**	**91.46**	**93.77**	**94.51**	**94.95**	**93.24**

Table 2: Comparison of our method to state-of-the-art (top) and baselines (bottom) in terms of token-level accuracy (in %) on 3 Tweets datasets. Note that, baselines are more fairly comparable to our method. In the second and last columns, we respectively highlighted the number of parameters and the average performance on the 3 datasets.

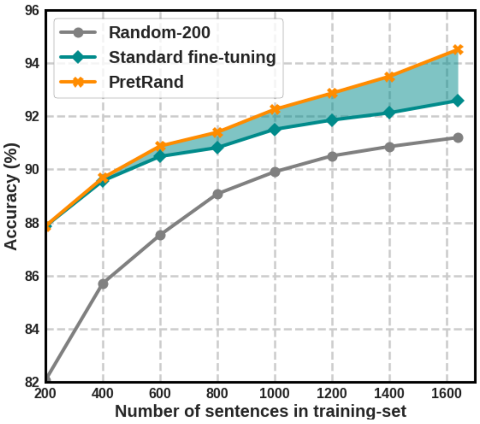


Figure 3: Performances (on dev-set of TweeBank) according different training-set sizes for the target-dataset. Transparent green highlights the difference between our PretRand and standard fine-tuning.

Method	TPoS	ArK	TweeBank	Avg
PretRand	**91.46**	**93.77**	**94.95**	**93.39**
-learnVect	91.25	93.46	94.59	93.10
-random^{++}	90.97	93.11	94.13	92.73
-l2 norm	90.76	92.11	93.38	92.08

Table 3: Ablation study. Token level accuracy (in %) when progressively ablating PretRand components.

Markov Model (MEMM) with greedy decoding and using brown clustering and careful hand-engineered features. Recently, Gui et al. (2017) proposed TPANN that uses adversarial training to leverage huge amounts of unlabelled Tweets.

3.4 Results

From the results given in Table 2, one can first see that our approach outperforms the SOTA and baseline methods on all the datasets. More interestingly, PretRand significantly outperforms the popular fine-tuning baseline by +1.4% absolute point on average and is better on all classes (see per-class improvement on Fig. 4). PretRand also outperforms the challenging Ensemble Model by a large margin (+2.2%), while using much less parameters. This clearly highlights the difference of our method with ensemble methods and the importance of having a shared word representation as well as our normalisation and weighting learnable vectors during training. A key asset of PretRand, is that it uses only 0.02% more parameters compared to the fine-tuning baseline.

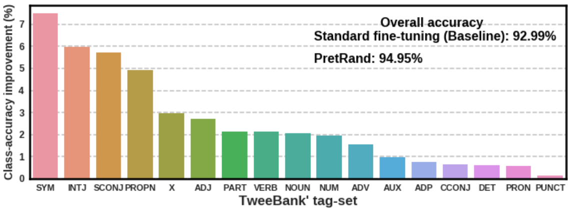


Figure 4: Sorted class-accuracy improvement (%) on TweeBank of PretRand compared to fine-tuning.

An interesting experiment is to evaluate the gain of performance of PretRand compared to fine-tuning, according different target-datasets' sizes. From the results in Fig. 3, PretRand has desirably a bigger gain with bigger *target-task* datasets, which clearly means that the more target training-data, the more interesting our method will be.

To assess the contribution of different components of PretRand, we performed an ablation study. Specifically, we successively ablated the main components of PretRand, namely, the learnable vectors (learnVect), the longer training for random units (random++) and the normalisation (ℓ_2-norm). From the results in Table 3, we can observe that the performances are only marginally better than standard fine-tuning when ablating the three components from PretRand. More importantly, adding each of them successively, makes the performances significantly better, which highlights the importance of every component.

4 Analysis

Bias when fine-tuning pre-trained units

Here our goal is to highlight that as in (Zhou et al., 2018a), pre-trained units can be biased in

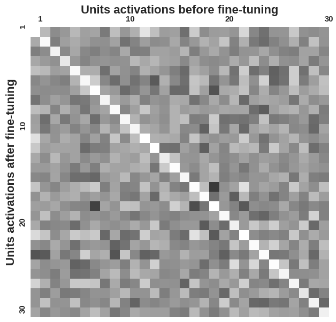
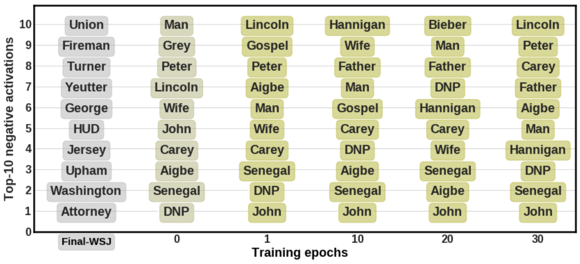


Figure 5: Correlation between units' activations before fine-tuning (columns) and after fine-tuning (rows).

the standard fine-tuning scheme. To do so, we follow (Tamaazousti et al., 2017) and analyse the units of **Φ** (biLSTM layer) *before* and *after* fine-tuning. Specifically, we compute the Pearson's correlation between all the units of the layer before and after fine-tuning. Here, a unit is represented by the random variable being the concatenation of its output activations from all the validation samples of the TweeBank dataset. From the resulting correlation matrix illustrated in Fig. 5, one can clearly observe the white diagonal, highlighting the fact that, every unit after fine-tuning is more correlated with itself before fine-tuning than with any other unit. This clearly confirms our initial motivation that pre-trained units are highly biased to what they have learned in the source-dataset, making them limited to learn patterns specific to the target-dataset.

Additionally, we visualise in Fig. 6 a concrete example of a biased neuron when transferring from newswire to Tweets domain. Specifically, we show the top-10 words activating unit-169 of **Φ** (from the standard fine-tuning baseline), *before* fine-tuning (at this stage, the model is trained on the source-dataset WSJ) and *during fine-tuning* on the TweeBank dataset. We can observe that this unit is highly sensitive to proper nouns (*e.g.*, *George* and *Washington*) before fine-tuning, and to words with capitalised first-letter whether the word is a proper noun or not (*e.g.*, *Man* and *Father*) during fine-tuning on TweeBank dataset. Indeed, we found that most of tokens with upper-cased first letter are mistakenly predicted as proper nouns (*PROPN*) in the standard fine-tuning scheme. In fact, in standard English, inside sentences, only proper nouns start with upper-cased letter thus fine-tuning the pre-trained model fails to slough this pattern which is not always respected in Tweets.



	Final-WSJ	0	1	10	20	30
10	Union	Man	Lincoln	Hannigan	Bieber	Lincoln
9	Fireman	Grey	Gospel	Wife	Man	Peter
8	Turner	Peter	Peter	Father	Father	Carey
7	Yeutter	Lincoln	Aigbe	Man	DNP	Father
6	George	Wife	Man	Gospel	Hannigan	Aigbe
5	HUD	John	Wife	Carey	Carey	Man
4	Jersey	Carey	Carey	DNP	Wife	Hannigan
3	Upham	Aigbe	Senegal	Aigbe	Senegal	DNP
2	Washington	Senegal	DNP	Senegal	Aigbe	Senegal
1	Attorney	DNP	John	John	John	John

Figure 6: Top-10 words activating unit-169 of standard fine-tuning scheme, before fine-tuning (Final-WSJ) and during fine-tuning on TweeBank.

	0	1	10	20	30	40
10	Admin	n't	do	na	n't	n't
9	deeply	n't	n't	n't	talk	ta
8	Allah	n't	ta	na	n't	na
7	:')	do	n't	n't	n't	n't
6	empower	n't	n't	n't	n't	n't
5	blessed	n't	start	n't	say	n't
4	blessed	n't	WIN	n't	do	n't
3	blessed	n't	n't	n't	talk	n't
2	bless	listen	listen	do	na	say
1	blessed	n't	n't	na	na	do

Figure 7: Top-10 words activating unit-99 of the random branch of PretRand, before and during training.

Unique units emerge in random branch

Finally, we highlight the ability of randomly initialised units to learn patterns specific to the target-dataset and not learned by the pre-trained ones because of their bias problem. To do so, we visualise unique units – *i.e.*, random units having a correlation lower than 0.4 with pre-trained ones – emerging in the random branch. While only one shown in Fig. 7, many unique units have been learned by the random branch of our PretRand model: 37.5% of the 200 random units have correlation lower than 0.4 with the pre-trained ones. Regarding unit-99, it is highly discriminative to tokens *"na"*, *"ta"* and *"n't"*. Indeed, in TweeBank, words like *"gonna"* (going to) are tokenized into two tokens: *"gon"* and *"na"*, with the later annotated as a *particle* and the former as a *verb*. Importantly, not even one unit from the standard fine-tuning scheme has been found firing on the same important and target-dataset specific pattern.

5 Conclusion

In this paper, we introduced a method to improve fine-tuning using 3 main ideas: adding random units and jointly learn them with pre-trained ones; normalising the activations of both to balance their different behaviours; applying learnable weights on both predictors to let the network learn which of random or pre-trained one is better for every class. We have demonstrated its effectiveness on domain adaptation from newswire domain to three commonly used Tweets-datasets for POS tagging.

References

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP 2013*, pages 198–206.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *EMNLP*, pages 2411–2420.

Wei Liu, Andrew Rabinovich, and Alexander C Berg. 2015. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

Yijia Liu, Yi Zhu, Wanxiang Che, Bing Qin, Nathan Schneider, and Noah A Smith. 2018. Parsing tweets into universal dependencies. In *NAACL*, volume 1, pages 965–975.

Sara Meftah, Nasredine Semmar, and Fatiha Sadat. 2018a. A neural network model for part-of-speech tagging of social media texts. In *LREC*.

Sara Meftah, Nasredine Semmar, Fatiha Sadat, and Stephan Raaijmakers. 2018b. Using neural transfer learning for morpho-syntactic tagging of south-slavic languages tweets. In *VarDial 2018*, pages 235–243.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *EMNLP*, pages 479–489.

Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *Proceedings of the Second Conference on Machine Translation*, pages 80–89.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL*, pages 380–390.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, volume 1, pages 2227–2237.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*, pages 1524–1534. Association for Computational Linguistics.

Nasredine Semmar, Laib Meriama, and Christian Fluhr. 2006. A deep linguistic analysis for cross-language information retrieval. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

Nasredine Semmar, Laib Meriama, and Christian Fluhr. 2008. Evaluating a natural language processing approach in arabic information retrieval. In *ELRA Workshop on Evaluation Looking into the Future of Evaluation: When automatic metrics meet task-based and performance-based approaches*, page 24.

Youssef Tamaazousti, Hervé Le Borgne, and Céline Hudelot. 2017. Mucale-net: Multi categorical-level networks to generate more discriminating features. In *CVPR*.

Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed El Amine Seddik, and Mohamed Tamaazousti. 2018. Learning more universal representations for transfer-learning. *arXiv:1712.09708*.

O Zennaki, N Semmar, and Laurent Besacier. 2019. A neural approach for inducing multilingual resources and natural language processing tools for low-resource languages. *Natural Language Engineering*, 25(1):43–67.

Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. 2018a. Interpreting deep visual representations via network dissection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018b. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*.

Show Some Love to Your n-grams: A Bit of Progress and Stronger n-gram Language Modeling Baselines

Ehsan Shareghi, Daniela Gerz, Ivan Vulić, Anna Korhonen
Language Technology Lab, DTAL, University of Cambridge
`{es776, dsg40, iv250, alk23}@cam.ac.uk`

Abstract

In recent years neural language models (LMs) have set state-of-the-art performance for several benchmarking datasets. While the reasons for their success and their computational demand are well-documented, a comparison between neural models and more recent developments in n-gram models is neglected. In this paper, we examine the recent progress in n-gram literature, running experiments on 50 languages covering all morphological language families. Experimental results illustrate that a simple extension of Modified Kneser-Ney outperforms an LSTM language model on 42 languages while a word-level Bayesian n-gram LM (Shareghi et al., 2017) outperforms the character-aware neural model (Kim et al., 2016) on average across all languages, and its extension which explicitly injects linguistic knowledge (Gerz et al., 2018a) on 8 languages. Further experiments on larger Europarl datasets for 3 languages indicate that neural architectures are able to outperform computationally much cheaper n-gram models: n-gram training is up to $15,000\times$ quicker. Our experiments illustrate that standalone n-gram models lend themselves as natural choices for resource-lean or morphologically rich languages, while the recent progress has significantly improved their accuracy.

1 Introduction

Statistical language models (LMs) are the pivot for several natural language processing tasks where a model trained on a text corpus is required to assign a probability to a given sequence $w_1 w_2 ... w_N$ (denoted by w_1^N). This probability indicates how likely is for w_1^N to belong to the corpus and is decomposed into conditional probabilities of words given their preceding contexts as $P(w_1^N) = \prod_{i=1}^{N} P(w_i|w_1^{i-1})$.

In n-gram LMs the unbounded conditional probabilities $P(w_i|w_1^{i-1})$ are approximated by imposing a finite-order Markov assumption, $P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1})$. Several smoothing techniques address the statistical sparsity issue for computing the conditional probabilities (Kneser and Ney, 1995; Chen and Goodman, 1999; Teh, 2006; Shareghi et al., 2016a), while others avoided the above approximation with unbounded hierarchical nonparametric Bayesian frameworks (Wood et al., 2011; Shareghi et al., 2017).

Alternatively, neural LMs compute $P(w_i|w_1^{i-1})$ via recurrent neural units which, in theory, are capable of encoding an unbounded context w_1^{i-1}. In recent years, neural LMs have become the prominent class of language modeling and have established state-of-the-art results on almost all sufficiently large benchmarks (Melis et al., 2018; Yang et al., 2018). While outperforming n-grams in terms of predictive accuracy, the computational shortcomings of neural LMs are well-documented: Training neural LMs is computationally expensive to the point that running experiments on large data ($\geq$ a few GiBs) is beyond the reach of academic research to this date (Chen et al., 2016; Patwary et al., 2018; Puri et al., 2018).[1] Similarly, querying is slower for neural LMs due to the required matrix-based operations, whereas most of the widely used n-gram LM toolkits rely on a few hash lookups and much cheaper scalar-based operations (Liu et al., 2018; Tang and Lin, 2018).

Nonetheless, it has been shown that the best predictive performance is still achieved by combining the two models via a basic interpolation or a mixture model (Jozefowicz et al., 2016; Neubig and Dyer, 2016): this indicates that the progress in n-gram LM should eventually be reflected in improving the

---

[1] For instance, n-gram LMs could be trained on 32GiB of data on a single CPU with ~32GiB of RAM in half a day (Shareghi et al., 2016b). A ballpark estimate for neural LMs, based on Puri et al. (2018), requires 26 Tesla V100 16GB GPUs to finish within the same amount of time while its financial cost is at least 100× higher.

Proceedings of NAACL-HLT 2019, pages 4113–4118
Minneapolis, Minnesota, June 2 - June 7, 2019.

state-of-the-art performance. Inspired by this, in this paper we shed new light on the most notable recent progress in n-gram statistical LMs which improves their predictive accuracy.

We demonstrate that under a recent massively multilingual experimental setup of Gerz et al. (2018a), more recent extensions of Kneser-Ney family of n-gram LMs (Shareghi et al., 2016a, 2017) are highly competitive with neural LMs. More specifically, we experiment on 50 languages from different morphological families, and illustrate that a word-level Bayesian n-gram LM (Shareghi et al., 2017) outperforms the character-level informed neural counterpart (Kim et al., 2016) on average, and its linguistically informed variant (Gerz et al., 2018a) on 8 languages. On larger Europarl datasets we find that n-gram models cannot reach the performance peaks of computationally much more expensive neural models, but a 2× decrease in perplexity comes at the cost of 15,000× longer training time. Our work reveals that recent n-gram LMs should be used as strong baselines, especially in resource-lean LM data and for morphologically rich languages.

Additionally, n-gram LMs offer a stringent way of dealing with Out-of-Vocabulary (OOVs) and rare words in a *full vocabulary* setting without relying on any pruning (Heafield, 2013). However, in neural LMs this remains an open question (Kawakami et al., 2017; Kim et al., 2016; Cotterell et al., 2018), while a common practice is pruning the training corpus and imposing *closed vocabulary* assumption (Mikolov et al., 2010) where rare words at training and unseen words at test are treated as an UNK token. We provide the mathematical underpinnings of n-gram models and highlight how this popular treatment works in favor of neural LMs (in comparative studies), and enforces n-gram LMs to perform much worse than their full potential.

2 n-gram Language Models: Smoothing

We now provide an overview of established smoothing techniques for n-gram LMs and their recent extensions. Smoothing is typically achieved by interpolation, where the probability of seeing a word w_i after a context w_{i-n+1}^{i-1} $P(w_i|w_{i-n+1}^{i-1}, \Theta)$ is smoothed by its probability after a shorter context, w_{i-n+2}^{i-1}, and follows the following general form:

$$\beta(w_i|w_{i-n+1}^{i-1},\Theta)+\gamma(w_{i-n+1}^{i-1},\Theta)P(w_i|w_{i-n+2}^{i-1},\Theta). \tag{1}$$

	$\beta(w_i \mid w_{i-n+1}^{i-1}, \Theta)$	Θ
KN	$\frac{\left[c(w_{i-n+1}^{i})-D_n\right]^+}{c(w_{i-n+1}^{i-1})}$	D_n
MKN	$\frac{\left[c(w_{i-n+1}^{i})-D_n^{c(w_{i-n+1}^{i})}\right]^+}{c(w_{i-n+1}^{i-1})}$	$D_n^{i\in\{1,2,3+\}}$
GKN	$\frac{\left[c(w_{i-n+1}^{i})-D_n^{c(w_{i-n+1}^{i})}\right]^+}{c(w_{i-n+1}^{i-1})}$	$D_n^{i\in\{1,...,10+\}}$
BKN	$\frac{\left[c(w_{i-n+1}^{i})-D_{w_{i-n+1}}t_{w_{i-n+1}}^{w_i}\right]^+}{c(w_{i-n+1}^{i-1})+\theta_{w_{i-n+1}}}$	$D_{w_{i-n+1}}, \theta_{w_{i-n+1}}, t_{w_{i-n+1}}^{w_i}$

Table 1: Top-level interpolation for n-gram LMs smoothings and its parameters, and $[x]^+ \overset{\text{def}}{=} \max\{x, 0\}$.

The term $\beta(\cdot)$ represents the existing mass for the n-gram (e.g. via maximum likelihood estimation), γ is the weight used for redistributing the preserved mass (e.g. via discounting), and Θ are the parameters of the smoothing technique. The recursion stops at the unigram level where the conditioning context is empty. The recursion at lower levels relies on different quantities (e.g. pseudo-counts) but for brevity we focus on the top level of recursion and only the first term, $\beta(w_i|w_{i-n+1}^{i-1}, \Theta)$, which suffices to highlight the key differences between smoothing techniques (see Table 1).

Kneser-Ney (KN). The key parameters of KN (Kneser and Ney, 1995) are the k-gram specific discounts D_k which control the amount of preserved and redistributed mass at kth level of recursion. While learning the discounts (on held-out data) is a possibility, the following estimation is shown to work well in practice:

$$D_k = 1 - 2\frac{n_2(k)}{n_1(k)}.\frac{n_1(k)}{n_1(k) + 2n_2(k)}. \tag{2}$$

It captures the characteristics of different k-gram orders by looking at the number of unique k-grams which occurred once, $n_1(k)$, or twice, $n_2(k)$ in the training data, defined as follows:

$$n_j(k) = \begin{cases} |\{\alpha \text{ s.t. } |\alpha|=k, c(\alpha)=j\}|, & \text{if } k=n \\ \left|\left\{\alpha \text{ s.t. } |\alpha|=k, N_{1+}(\bullet\alpha)=j\right\}\right|, & \text{if } k<n \end{cases} \tag{3}$$

where $N_{1+}(\bullet\alpha) = |\{w : c(w\alpha) > 0\}|$, and is referred to as a form of pseudo-count, and $c(\alpha)$ denotes the frequency of sequence α. KN considers one discount value at each level of recursion and the discounts are bounded, $0 \leq D_k < 1$.

Modified Kneser-Ney (MKN). Similarly, Modified Kneser-Ney (Chen and Goodman, 1999) is defined with modifications applied to the discounting mechanism in order to make them sensitive to the *existing* mass j. The discounts are estimated as:

$$D_k^j = \begin{cases} 0, & \text{if } j = 0 \\ j-(j+1)\frac{n_{j+1}(k)}{n_j(k)} \frac{n_1(k)}{n_1(k)+2n_2(k)}, & \text{if } j < \mathcal{B} \\ \mathcal{B}-(\mathcal{B}+1)\frac{n_4(k)}{n_3(k)} \cdot \frac{n_1(k)}{n_1(k)+2n_2(k)}, & \text{if } j \geq \mathcal{B} \end{cases} \quad (4)$$

where $n_j(k)$ is defined in Eq. (3), and $\mathcal{B} = 3$. This leads to three discount parameters $\{D_k^1, D_k^2, D_k^{3+}\}$ for each recursion level, and allows for discount values to be as large as $c(w_{i-n+1}^i)$. MKN is widely accepted as the state-of-the-art n-gram LM and is very frequently confused with KN in the literature.

Generalized Modified Kneser-Ney (GKN). In conditions where statistical sparsity is more severe a more refined approach to model the distributions is necessary. Motivated by this, Shareghi et al. (2016a) provided the mathematical proof (based on leave-one-out log-likelihood) of the discount bounds used in MKN and proposed a natural extension of its discount binning to $\mathcal{B} = 10$. This was shown to be effective for further perplexity reduction in out-of-domain setting where OOV ratio is naturally high.

Bayesian Kneser-Ney (BKN). The Bayesian generalization of KN is the Hierarchical Pitman-Yor Process (HPYP) language model (Teh, 2006). This can be interpreted as a richer parameterization of KN and MKN, where the additional parameters (introduced shortly) allow the model to have more flexibility in capturing the desired distribution.

The Pitman-Yor Process (Pitman et al., 1997), $\text{PYP}(D, \theta, H)$, is a distribution defined over probability vectors (each draw from a PYP is a multinomial distribution) and has three parameters: a base probability vector H which is the expected value of a draw from a PYP, the concentration parameter $-D < \theta$ which controls the variation of draws from a PYP around H, and the discount parameter $0 \leq D < 1$ which allows the drawn vectors to capture the power-law behavior.

In the LM context, given a sequence of words w_{i-n+1}, a draw from a PYP is a multinomial distribution over the words following this sequence, denoted by $G_{w_{i-n+1}}$. This distribution can be captured by a vector of two counts $\{t_{w_{i-n+1}}^{w_i}, n_{w_{i-n+1}}^{w_i}\}^{w_i \in \sigma}$ which defines a partitioning arrangement of $G_{w_{i-n+1}}$, while different partitioning correspond to different multinomial draws from PYP. Here, $n_{i-n+1}^{w_i}$ is the total number of evidence [2] for word w_i after the context w_{i-n+1}, $t_{i-n+1}^{w_i}$ is the total number of partitions dedicated to w_i constrained by $0 \leq t_{i-n+1}^{w_i} \leq n_{i-n+1}^{w_i}$, and σ is the vocabulary.

The HPYP ties PYP distributions through their base and offers the statistical mean to smooth over infinitely long contexts (Wood et al., 2011). For instance, $\text{PYP}(D_u, \theta_u, \text{PYP}(D_{\pi(u)}, \theta_{\pi(u)}, H))$ is a two-level HPYP where a draw from a *child* distribution u, is smoothed by a draw from its *parent* $\pi(u)$. Here, $\pi(u)$ is u with its most earliest word dropped (e.g., $u = abc$, $\pi(u) = bc$).

KN can be seen as a special case for BKN, when the concentration is 0, and $\{t_{w_{i-n+1}}^{w_i}{=}1\}^{w_i \in \sigma}$. BKN can be considered as a richer parameterization of MKN: The product $D_{w_{i-n+1}} t_{w_{i-n+1}}^{w_i}$ (see Table 1) allows ≥ 1 discounting, simulating the discount range of MKN, while an additional parameter $\theta_{w_{i-n+1}}$ permits further adjustments of the distribution. BKN is shown to outperform MKN (Shareghi et al., 2017) but relies on expensive parameter sampling.

Out-of-Vocabulary (OOV). To complete the definitions we now explain how unseen words or contexts are handled during the computation without resorting to pruning or closed vocabulary setting. This treatment is the same for both non-Bayesian and Bayesian methods described in this paper. Let us consider the generic interpolation form of Eq. (1) and the maximum likelihood estimation of the $\beta(.)$ term (hence Θ is dropped) in the top level of the interpolation, $\beta(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i)}{c(w_{i-n+1}^{i-1})}$. Regardless of the level of interpolation and the paradigm used for computing the $\beta(.)$ term, $\beta(.)$ and $\gamma(.)$ always share the same denominator (normalizing factor).

An unseen word can appear as the target word w_i, which results in $\beta(.) = 0$ as $c(w_{i-n+1}^i) = 0$. It can also appear as a part of the prediction context w_{i-n+1}^{i-1} in which case both the $\beta(.)$ and $\gamma(.)$ terms will be undefined (and ignored) as the denominator $c(w_{i-n+1}^{i-1}) = 0$. This procedure is applied to all levels of interpolation without loss of generality, and as can be seen it only relies on the basic mathematical property of the involved computations rather than any other presumptions about data preprocessing or vocabulary. [3]

---

[2] This is equal to $c(w_{i-n+1}^i)$ at the top level of recursion, hence not mentioned as a part of Θ in Table 1.

[3] See Shareghi (2017) for a comprehensive explanation of the models covered in this section.

		n-gram LMs pplx				neural LMs pplx		
Lang	OOV %	KN	MKN	GKN	BKN	LSTM	CNN	AP
♣ am	15.2	1289	1252	1101	941†	1535	981	817
♣ ar	11.1	2241	2156	1871	1746	2587	1659	1604
♡ bg	9.9	636	610	524	470	651	415	409
♡ ca	5.2	369‡	358‡	307	280	318	241	238
♡ cs	11	1724	1658	1433	1331	2200	1252	1131
♡ da	9.6	690	668	582	526	710	466	442
♡ de	10.6	964‡	930‡	814	721	903	602	551
♡ el	8.2	627‡	607‡	526	499	538	405	389
♡ en	6.1	553‡	533‡	454	391	494	371	349
♡ es	5.9	430‡	415‡	356	319	366	275	270
♠ et	12.9	1655	1609	1422†	1223†	2564	1478	1388
♠ eu	9.1	571‡	560‡	496	456	533	347	309
♡ fa	4.7	362‡	355‡	308‡	283‡	263	208	205
♠ fi	17.6	2709	2611	2318	2143†	4263	2236	1858
♡ fr	6.1	361‡	350‡	298‡	268	294	231	220
♣ he	8.9	1870	1797	1531	1373§	2189	1519	1375
♡ hi	7.1	488‡	473‡	408	378	426	326	299
♡ hr	10.6	1345	1294	1124	972†	1665	1014	906
♠ hu	12	1188	1151	1011	877†	1595	929	819
◇ id	6	469‡	454‡	388‡	358	359	286	263
♡ it	6	582‡	567‡	489	465	493	349	350
♠ ja	4.7	174‡	169‡	142	129†	156	136	125
♠ jv	12.2	1462‡	1387	1217	1138†	1443	1158	1003
♠ ka	11.7	1422	1370	1198	1115	1827	1097	939
◇ km	4.8	608	586	501§	451§	637	522	535
♠ kn	17.7	2378	2315	2051§	1914§	5310	2558	2265
♠ ko	19.3	5332	5146	4492†	4019†	10063	4778	3821
♡ lt	11	1187	1155	1001	891	1415	854	827
♡ lv	12.7	1518	1452	1255	1144	1967	1129	969
♠mng	10.8	1441	1392	1215	1055§	1716	1165	1091
◇ ms	7.2	807‡	776‡	659	572	725	525	513
◇ my	2	216‡	209	178§	168§	212	182	180
◇ nan	3.8	63‡	61‡	52‡	45‡	43	39	38
♡ nl	5.6	407‡	397‡	347‡	305	340	267	248
♡ no	9	551‡	534‡	465	435	513	379	346
♡ pl	12	1810	1741	1514	1363†	2641	1491	1328
♡ pt	5	351‡	342‡	290‡	261	272	214	202
♡ ro	7	395‡	384‡	330	290	359	256	247
♡ ru	10.1	1160	1128	977	906	1309	812	715
♡ sk	13.1	1633	1560	1352	1234†	2062	1275	1151
♡ sl	10.9	1160	1114	969	856	1308	776	733
♡ sr	9.3	812	790	683	637	961	582	547
♡ sv	10	873‡	843‡	734	634	832	583	543
♠ ta	18.2	3469†	3342†	2920†	2635§	6234	3496	2768
◇ th	3.3	238	233	199§	181§	241	206	199
◇ tl	7.7	393‡	379‡	321‡	293	298	219	211
♠ tr	12.3	1784	1724	1497	1416	2267	1350	1290
♡ uk	12.8	1707	1639	1418	1338	1893	1283	1091
◇ vi	2.7	202‡	197‡	170	145§	190	158	165
◇ zh	3.8	1110‡	1064‡	899‡	777†	826	797	762
◇ avg	4.6	456‡	440‡	374	332	392	326	318
♡ avg	8.8	873	842	729	661	969	618	566
♠ avg	13.2	1965	1898	1665†	1510†	3164	1727	1473
♣ avg	11.7	1800	1735	1501	1354†	2104	1386	1265
+ avg	9.3	1116	1077	936	847†	1460	878	781

Table 2: Data Statistics and Perplexity scores. OOV denotes the percentage of unseen words at test time. For detailed data stats see (Gerz et al., 2018a). Suit symbols denote morphological types: ◇ Isolating, ♡ Fusional, ♠ Agglutinative, ♣ Introflexive. Color coded shapes denote comparative performance: ‡ is outperformed by LSTM, † outperforms CNN, § outperforms AP.

3 Experiments and Results

As our main large-scale experiment we use a typologically diverse set of 50 languages. These LM datasets cover many languages which are challenging in terms of data size, as well as the type-token ratio. In a less challenging setup, we experiment on 3 languages with larger training data from the Europarl (Koehn, 2005) to compare the two classes of LMs based on perplexity reduction and training time. For full data statistics, actual sampling of languages and data curation see Gerz et al. (2018a,b).

The common practice of setting a frequency threshold and mapping training data unigrams ($(k = 1)$-gram) to an UNK token degrades the performance of n-gram LMs by discarding a range of discount parameters: e.g., using threshold (< 3) results in $n_1(1), n_2(1) = 0$, both included in Eq. (2) and Eq. (4), increases the average perplexity score of 5-gram KN in our experiments by 11%. Motivated by its significance, we base our comparison on reported results by Gerz et al. (2018a): they deal with the task in the full vocabulary setting (Adams et al., 2017) with word-level predictions, and follow a relatively comparable treatment of unseen words with both n-gram and neural LM families (although not identical) at test time without enforcing any threshold over the training data.

The benchmarked neural LMs include three models with word-level predictions: a standard LSTM (Zaremba et al., 2014), a Char-CNN-LSTM (denoted as CNN) (Kim et al., 2016) which incorporates character-level information in the input, and the Attract-Preserve model (denoted as AP) (Gerz et al., 2018a) which injects further subword-level information. All benchmarked n-gram LMs are 5-grams, with the exception of BKN which is an ∞-gram model trained via 5 samples[4] following the recipe of Shareghi et al. (2017). GKN results are based on $\mathcal{B} = 5$, tuned on a development set.

Results and Discussion. The main results on the 50-languages benchmark are summarized in Table 2. The results for the more recent n-gram LMs indicate that these n-gram models are highly competitive with neural LMs in this challenging resource-lean setup.[5] For instance, for 26/50 languages all n-gram models outperform a regular LSTM, while GKN and BKN extend the lead to 42/50

---

[4] Marginal improvements achieved with more sampling.

[5] The size of each dataset is ≈ 40K sentences, which is at the level of the standard Penn Treebank dataset often used for LM evaluation in English (Marcus et al., 1993).

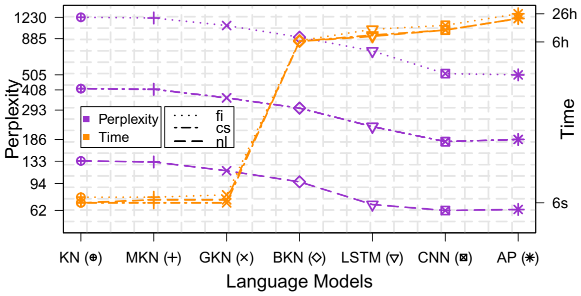


Figure 1: Time-Perplexity for *n*-gram and neural LMs across 3 languages fi, cs, nl. Timings done on a single core of AMD Ryzen 1900X for *n*-grams, and on a Nvidia TITAN X Pascal GPU for neural models.

and 48/50 languages, respectively. For certain morphologically rich languages (e.g., Tamil, Mongolian, Hebrew), the *n*-gram LMs are able to outscore even character-aware neural LMs. On average we observe *n*-grams succeed, especially the BKN model, for introflexive and agglutinative languages which are known to have productive morphological systems: as reported in Table 2, they have the higher OOV ratio compared to the other language families. Overall, the best performing *n*-gram model, BKN, outperforms both the LSTM (42% reduction in perplexity) and CNN models (3% reduction in perplexity), while falling behind AP by 8%.

These results highlight that *n*-gram models can serve as strong baselines for such morphologically rich languages with high OOV rates and type-to-token ratios, which are especially problematic in scarce data setups. Additionally, they suggest that more sophisticated *n*-gram variants such as GKN or BKN should be used to provide adequate comparison points with *n*-gram LMs than the commonly used KN or MKN.

As expected, experiments on 10× larger Europarl datasets for 3 languages show that neural models outperform *n*-gram models in less challenging data-intensive scenarios. However, training on large datasets comes at the expense of training efficiency for neural models: e.g., according to Figure 1 training non-Bayesian *n*-grams is around 15,000× quicker than training neural models. We leave a full-fledged investigation on the relation of training corpus size and efficacy of *n*-gram vs. neural LM training for future work. In addition, motivated by these preliminary insights, we advocate investing further efforts in future work into coupling the ideas behind *n*-gram and neural LMs towards improved language modeling.

4 Conclusion

We provided an overview of previous work and very recent progress in *n*-gram LMs. The recent developments, when tested on a challenging set of 50 languages, demonstrated superior or highly competitive performance compared with neural LMs, while being substantially cheaper to train. We also shed light on a common issue in the experimental setups, concerning OOV or rare words handling, when comparing *n*-grams and neural LMs.

While being non-trivial, investigating any correlation between cheap-to-compute heuristics (e.g., basic data statistics) and the choice of the most suitable model for a given dataset is worth exploring. Also, motivated by our findings, we will work on utilizing continuous space representations as side information in sampling the parameters of BKN, i.e. similar to Zhao et al. (2018), which potentially can reduce the gap between BKN and neural models.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909). The authors would like to thank the anonymous reviewers for their helpful suggestions. The first author would like to also thank the members of the Language Technology Lab for their comments on the presentation of this work.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *EACL*, pages 937–947.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *ACL*, pages 1975–1985.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *NAACL*, pages 536–541.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018a. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *TACL*, 6:451–465.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018b. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *EMNLP*, pages 316–327.

Kenneth Heafield. 2013. *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *ACL*, pages 1492–1502.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*, pages 181–184.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Xuan Liu, Di Cao, and Kai Yu. 2018. Binarized LSTM language model. In *NAACL*, pages 2113–2121.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Gabor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *ICLR*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Graham Neubig and Chris Dyer. 2016. Generalizing and hybridizing count-based and neural language models. In *EMNLP*, pages 1163–1172.

Mostofa Patwary, Milind Chabbi, Heewoo Jun, Jiaji Huang, Gregory Diamos, and Kenneth Church. 2018. Language modeling at scale. *arXiv preprint arXiv:1810.10045*.

Jim Pitman, Marc Yor, et al. 1997. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900.

Raul Puri, Robert Kirby, Nikolai Yakovenko, and Bryan Catanzaro. 2018. Large scale language modeling: Converging on 40GB of text in four hours. *arXiv preprint arXiv:1808.01371*.

Ehsan Shareghi. 2017. *Scalable Non-Markovian Sequential Modelling for Natural Language Processing*. Ph.D. thesis, Monash University.

Ehsan Shareghi, Trevor Cohn, and Gholamreza Haffari. 2016a. Richer interpolative smoothing based on modified Kneser-Ney language modeling. In *EMNLP*, pages 944–949.

Ehsan Shareghi, Gholamreza Haffari, and Trevor Cohn. 2017. Compressed nonparametric language modelling. In *IJCAI*, pages 2701–2707.

Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2016b. Fast, small and exact: Infinite-order language modelling with compressed suffix trees. *TACL*, pages 477–490.

Raphael Tang and Jimmy Lin. 2018. Progress and tradeoffs in neural language models. *arXiv preprint arXiv:1811.00942*.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical report, NUS School of Computing.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The sequence memoizer. *Communications of the ACM*, 54(2):91–98.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

He Zhao, Lan Du, Wray L. Buntine, and Mingyuan Zhou. 2018. Inter and intra topic structure learning with word embeddings. In *ICML*, pages 5887–5896.

Data Augmentation for Context-Sensitive Neural Lemmatization Using Inflection Tables and Raw Text

Toms Bergmanis
School of Informatics
University of Edinburgh
`T.Bergmanis@sms.ed.ac.uk`

Sharon Goldwater
School of Informatics
University of Edinburgh
`sgwater@inf.ed.ac.uk`

Abstract

Lemmatization aims to reduce the sparse data problem by relating the inflected forms of a word to its dictionary form. Using context can help, both for unseen and ambiguous words. Yet most context-sensitive approaches require full lemma-annotated sentences for training, which may be scarce or unavailable in low-resource languages. In addition (as shown here), in a low-resource setting, a lemmatizer can learn more from n labeled examples of distinct words (types) than from n (contiguous) labeled tokens, since the latter contain far fewer distinct types. To combine the efficiency of type-based learning with the benefits of context, we propose a way to train a context-sensitive lemmatizer with little or no labeled corpus data, using inflection tables from the UniMorph project and raw text examples from Wikipedia that provide sentence contexts for the unambiguous UniMorph examples. Despite these being unambiguous examples, the model successfully generalizes from them, leading to improved results (both overall, and especially on unseen words) in comparison to a baseline that does not use context.

1 Introduction

Many lemmatizers work on isolated wordforms (Wicentowski, 2002; Dreyer et al., 2008; Rastogi et al., 2016; Makarov and Clematide, 2018b,a). Lemmatizing in context can improve accuracy on ambiguous and unseen words (Bergmanis and Goldwater, 2018), but most systems for context-sensitive lemmatization must train on complete sentences labeled with POS and/or morphological tags as well as lemmas, and have only been tested with 20k-300k training tokens (Chrupała et al., 2008; Müller et al., 2015; Chakrabarty et al., 2017).[1]

Intuitively, though, sentence-annotated data is inefficient for training a lemmatizer, especially in low-resource settings. Training on (say) 1000 word types will provide far more information about a language's morphology than training on 1000 contiguous tokens, where fewer types are represented. As noted above, sentence data can help with ambiguous and unseen words, but we show here that when data is scarce, this effect is small relative to the benefit of seeing more word types.[2] Motivated by this result, we propose a training data augmentation method that combines the efficiency of type-based learning and the expressive power of a context-sensitive model.[3] We use *Lematus* (Bergmanis and Goldwater, 2018), a state-of-the-art lemmatizer that learns from lemma-annotated words in their N-character contexts. No predictions about surrounding words are used, so fully annotated training sentences are not needed. We exploit this fact by combining two sources of training data: 1k lemma-annotated types (with contexts) from the Universal Dependency Treebank (UDT) v2.2[4] (Nivre et al., 2017), plus examples obtained by finding *unambiguous* word-lemma pairs in inflection tables from the Universal Morphology (UM) project[5] and collecting sentence contexts for them from Wikipedia. Although these examples are noisy and biased, we show that they improve lemmatization accuracy in experiments on 10 languages, and that the use of context helps, both overall and especially on unseen words.

2 Method

Lematus (Bergmanis and Goldwater, 2018) is a neural sequence-to-sequence model with attention

---

[1] The smallest of these corpora contains 20k tokens of Bengali annotated only with lemmas, which Chakrabarty et al. (2017) reported took around two person months to create.

[2] Garrette et al. (2013) found the same for POS tagging.

[3] Code and data: `https://bitbucket.org/tomsbergmanis/data_augumentation_um_wiki`

[4] `http://hdl.handle.net/11234/1-2837`

[5] `http://unimorph.org`

Proceedings of NAACL-HLT 2019, pages 4119–4128
Minneapolis, Minnesota, June 2 - June 7, 2019.

inspired by the re-inflection model of Kann and Schütze (2016), which won the 2016 SIGMORPHON shared task (Cotterell et al., 2016). It is built using the Nematus machine translation toolkit,[6] which uses the architecture of Sennrich et al. (2017): a 2-layer bidirectional GRU encoder and a 2-layer decoder with a conditional GRU (Sennrich et al., 2017) in the first layer and a GRU in the second layer.

Lematus takes as input a character sequence representing the wordform in its N-character context, and outputs the characters of the lemma. Special input symbols are used to represent the left and right boundary of the target wordform (`<lc>`, `<rc>`) and other word boundaries (`<s>`). For example, if $N = 15$, the system trained on Latvian would be expected to produce the characters of the lemma *ceļš* (meaning *road*) given input such as:

```
s a k a <s> p a š v a l d ī b u
<lc> c e ļ u <rc>
u n <s> i e l u <s> r e ģ i s t r
```

When $N = 0$ (**Lematus 0-ch**), no context is used, making Lematus 0-ch comparable to other systems that do not model context (Dreyer et al., 2008; Rastogi et al., 2016; Makarov and Clematide, 2018b,a). In our experiments we use both Lematus 0-ch and **Lematus 20-ch** (20 characters of context), which was the best-performing system reported by Bergmanis and Goldwater (2018).

2.1 Data Augmentation

Our data augmentation method uses UM inflection tables and creates additional training examples by finding Wikipedia sentences that use the inflected wordforms in context, pairing them with their lemma as shown in the inflection table. However, we cannot use all the words in the tables because some of them are ambiguous: for example, Figure 1 shows that the form `ceļi` could be lemmatized either as `ceļš` or `celis`. Since we don't know which would be correct for any particular Wikipedia example, we only collect examples for forms which are unambiguous according to the UM tables. However, this method is only as good as the coverage of the UM tables. For example, if UM doesn't include a table for the Latvian verb *celt*, then the underlined forms in Table 1 would be incorrectly labeled as unambiguous.

	noun: `ceļš`		**noun: `celis`**	
	SG	**PL**	**SG**	**PL**
NOM	ceļš	~~ceļi~~	celis	~~ceļi~~
GEN	~~ceļa~~	~~ceļu~~	~~ceļa~~	~~ceļu~~
DAT	ceļam	~~ceļiem~~	celim	~~ceļiem~~
ACC	ceļu	~~ceļus~~	celi	~~ceļus~~
INS	ceļu	~~ceļiem~~	celi	~~ceļiem~~
LOC	ceļā	~~ceļos~~	celī	~~ceļos~~
VOC	ceļ	~~ceļi~~	celi	~~ceļi~~

Table 1: Example UM inflection tables for Latvian nouns *ceļš* (*road*) and *celis* (*knee*). The ~~crossed out forms~~ are examples of evidently ambiguous forms that are not used for data augmentation because of being shared by the two lemmas. The underlined forms appear unambiguous in this toy example but actually conflict with inflections of the verb *celt* (*to lift*).

There are several other issues with this method that could potentially limit its usefulness. First, the UM tables only include verbs, nouns and adjectives, whereas we test the system on UDT data, which includes all parts of speech. Second, by excluding ambiguous forms, we may be restricting the added examples to a non-representative subset of the potential inflections, or the system may simply ignore the context because it isn't needed for these examples. Finally, there are some annotation differences between UM and UDT.[7] Despite all of these issues, however, we show below that the added examples and their contexts do actually help.

3 Experimental Setup

Baselines and Training Parameters We use four baselines: (1) **Lemming**[8] (Müller et al., 2015) is a context-sensitive system that uses log-linear models to jointly tag and lemmatize the data, and is trained on sentences annotated with both lemmas and POS tags. (2) The hard monotonic attention model (**HMAM**)[9] (Makarov and Clematide, 2018b) is a neural sequence-to-sequence model with a hard attention mechanism that advances through the sequence monotonically. It is trained on word-lemma pairs (without context)

---

[6] Code for Nematus: `https://github.com/EdinburghNLP/nematus`, Code for Lematus: `https://bitbucket.org/tomsbergmanis/lematus.git`

[7] Recent efforts to unify the two resources have mostly focused on validating dataset schema (McCarthy et al., 2018), leaving conflicts in word lemmas unresolved. We estimated (by counting types that are unambiguous in each dataset but have different lemmas across them) that annotation inconsistencies affect up to 1% of types in the languages we used.

[8] `http://cistern.cis.lmu.de/lemming`

[9] `https://github.com/ZurichNLP/coling2018-neural-transition-based-morphology`

with character-level alignments learned in a preprocessing step using an alignment model, and it has proved to be competitive in low resource scenarios. (3) Our naive **Baseline** outputs the most frequent lemma (or one lemma at random from the options that are equally frequent) for words observed in training. For unseen words it outputs the wordform itself. (4) We also try a baseline data augmentation approach (**AE Aug Baseline**) inspired by Bergmanis et al. (2017) and Kann and Schütze (2017), who showed that adding training examples where the network simply learns to auto-encode corpus words can improve morphological inflection results in low-resource settings. The AE Aug Baseline is a variant of Lematus 0-ch which augments the UDT lemmatization examples by auto-encoding the inflected forms of the UM examples (i.e., it just treats them as corpus words). Comparing AE Aug Baseline to Lematus 0-ch augmented with UM lemma-inflection examples tells us whether using the UM lemma information helps more than simply auto-encoding more inflected examples.

To train the models we use the default settings for Lemming and the suggested lemmatization parameters for HMAM. We mainly follow the hyperparameters used by Bergmanis and Goldwater (2018) for Lematus; details are in Appendix B.

Languages and Training Data We conduct preliminary experiments on five development languages: Estonian, Finnish, Latvian, Polish, and Russian. In our final experiments we also add Bulgarian, Czech, Romanian, Swedish and Turkish. We vary the amount and type of training data (types vs. tokens, UDT only, UM only, or UDT plus up to 10k UM examples), as described in Section 4.

To obtain N UM-based training examples, we select the first N unambiguous UM types (with their sentence contexts) from shuffled Wikipedia sentences. For experiments with $j > 1$ examples per type, we first find all UM types with at least j sentence contexts in Wikipedia and then choose the N distinct types and their j contexts uniformly at random.

Evaluation To evaluate models' ability to lemmatize wordforms in their sentence context we follow Bergmanis and Goldwater (2018) and use the full UDT development and test sets. Unlike Bergmanis and Goldwater (2018) who reported token level lemmatization exact match accuracy, we report *type-level* micro averaged lemmatization exact match accuracy. This measure better reflects improvements on unseen words, which tend to be rare but are more important (since a most-frequent-lemma baseline does very well on seen words, as shown by Bergmanis and Goldwater (2018)).

		Ambig.	**Unseen**	**All**
Tokens	*Baseline*	41.0	26.6	31.0
	Lemming	38.2	48.3	50.6
	HMAM	41.4	50.2	52.1
	Lematus 0-ch	39.9	43.7	46.8
	Lematus 20-ch	38.4	42.8	45.8
Types	*Baseline*	45.0	26.6	32.4
	Lemming	N/A	N/A	N/A
	HMAM	41.8	53.7	56.3
	Lematus 0-ch	42.5	53.7	55.1
	Lematus 20-ch	43.1	51.7	54.9

Table 2: Average type level lemmatization exact match accuracy on five development languages in type and token based training data scenarios. Colour-scale is computed over the whole *Ambig.* column and over all but *Baseline* rows for the other columns.

We separately report performance on unseen and ambiguous tokens. For a fair comparison across scenarios with different training sets, we count as unseen only words that are not ambiguous and are absent from *all* training sets/scenarios introduced in Section 4. Due to the small training sets, between 70-90% of dev set types are classed as unseen in each language. We define a type as ambiguous if the empirical entropy over its lemmas is greater than 0.1 in the full original UDT training splits.[10] According to this measure, only 1.2-5.3% of dev set types are classed as ambiguous in each language.

Significance Testing All systems are trained and tested on ten languages. To test for statistically significant differences between the results of two systems we use a Monte Carlo method: for each set of results (i.e. a set of 10 numerical values) we generate 10000 random samples, where each sample swaps the results of the two systems for each language with a probability of 0.5. We then obtain a p-value as the proportion of samples for which the difference on average was at least as large as the difference observed in our experiments.

4 Experiments, Results, and Discussion

Types vs. Tokens and Context in Very Low Resource Settings We compare training on the first

---

[10] This measure, *adjusted ambiguity*, was defined by Kirefu (2018), who noticed that many frequent wordforms appear to have multiple lemmas due to annotation errors. The adjusted ambiguity filters out these cases.

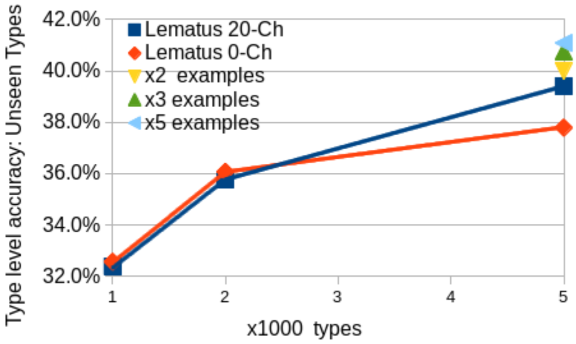


Figure 1: Average type level lemmatization exact match accuracy on unseen words of five development languages. X-axis: thousands of types in training data.

1k tokens vs. first 1k distinct types of the UDT training sets. Table 2 shows that if only 1k examples are available, using types is clearly better for all systems. Although Lematus does relatively poorly on the token data, it benefits the most from switching to types, putting it on par with HMAM and suggesting is it likely to benefit more from additional type data. Lemming requires token-based data, but does worse than HMAM (a context-free method) in the token-based setting, and we also see no benefit from context in comparing Lematus 20-ch vs Lematus 0-ch. So overall, in this very low-resource scenario with no data augmentation, context does not appear to help.

Using UM + Wikipedia Only We now try training only on UM + Wikipedia examples, rather than examples from UDT. We use 1k, 2k or 5k unambiguous types from UM with a single example context from Wikipedia for each. With 5k types we also try adding more example contexts (2, 3, or 5 examples for each type).

Figure 1 presents the results (for unseen words only). As with the UDT experiments, there is little difference between Lematus 20-ch and Lematus 0-ch in the smallest data setting. However, when the number of training types increases to 5k, the benefits of context begin to show, with Lematus 20-ch yielding a 1.6% statistically significant ($p < 0.001$) improvement over Lematus 0-ch. The results for increasing the number of examples per type are numerically higher than the one-example case, but the differences are not statistically significant.

It is worth noting that the accuracy even with 5k UM types is considerably lower than the accuracy of the model trained on only 1k UDT types (see Table 2). We believe this discrepancy is due to the issues of biased/incomplete data noted above.

	DEVELOPMENT			TEST
	Ambig.	*Unseen*	*All*	*All*
	1k UDT types (No augmentation)			
Baseline	**49.1**	30.8	36.7	-
HMAM	46.3	**58.9**†‡	**61.5**†‡	**62.6**†‡
Lematus 0-ch	46.5	55.0	58.5	59.1‡
Lematus 20-ch	45.0	54.3	57.7	57.7
	1k UDT types + 1k UM types			
Baseline	45.9	30.8	38.4	-
AE Aug Baseline	45.6	57.5	60.4	60.8
HMAM	45.9	60.2	64.2	64.3
Lematus 0-ch	46.6	59.0	63.4	63.6
Lematus 20-ch	**49.8***	**61.7***†	**65.5***†	**65.3**†
	1k UDT types + 5k UM types			
Baseline	**55.4***†‡	30.7	41.7	-
AE Aug Baseline	46.0	58.8	61.3	61.6
HMAM	46.7	60.8	65.7	65.7
Lematus 0-ch	46.2	61.5	66.1	66.4
Lematus 20-ch	48.6	**65.4***†	**69.2***†	**69.5** *†
	1k UDT types + 10k UM types			
Baseline	**54.9***†	31.2	43.5	-
AE Aug Baseline	46.3	58.6	61.2	61.7
HMAM	45.4	60.8	65.5	65.3
Lematus 0-ch	45.5	62.1	66.4	66.4
Lematus 20-ch	49.5*	**66.7***†	**70.6***†	**70.9***†

Table 3: Average lemmatization accuracy for all 10 languages, trained on 1k UDT types (No aug.), or 1k UDT plus 1k, 5k, or 10k UM types with contexts from Wikipedia. The numerically highest scores in each data setting are bold; *, †, and ‡ indicate statistically significant improvements over HMAM (Makarov and Clematide, 2018b), Lematus 0-ch and 20-ch, respectively (all $p < 0.05$; see text for details). Colour-scale is computed over the whole *Ambig.* column and over all but *Baseline* rows for the other columns.

For example, we analyzed the Latvian data and found that the available tables for nouns, verbs, and adjectives give rise to 78 paradigm slots. The 17 POS tags in UDT give rise to about 10 times as many paradigm slots, although only 448 are present in the unseen words of the dev set. Of these, 197 are represented amongst the 1k UDT training types, whereas only 25 are included in the 1k UM training types. As a result, about 72% of the unseen types of dev set have no representative of their paradigm slot in 1k types of UM, whereas this figure is only 17% for the 1k types of UDT.

Data Augmentation Although UM + Wikipedia examples alone are not sufficient to train a good lemmatizer, they might improve a low-resource baseline trained on UDT data. To see, we augmented the 1k UDT types with 1k, 5k or 10k UM

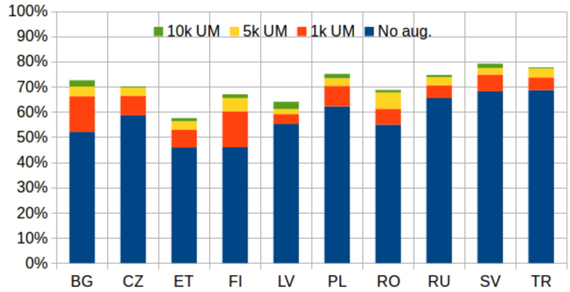


Figure 2: Lematus 20-ch lemmatization accuracy for each language on all types in the dev sets.

types with contexts from Wikipedia.

Table 3 summarizes the results, showing that despite the lower quality of the UM + Wikipedia examples, using them improves results of all systems, and more so with more examples. Improvements are especially strong for unseen types, which constitute more than 70% of types in the dev set. Furthermore, the benefit of the additional UM examples is above and beyond the effect of auto-encoding (AE Aug Baseline) for all systems in all data scenarios.

Considering the two context-free models, HMAM does better on the un-augmented 1k UDT data, but (as predicted by our results above) it benefits less from data augmentation than does Lematus 0-ch, so with added data they are statistically equivalent ($p = 0.07$ on the test set with 10k UM).

More importantly, Lematus 20-ch begins to outperform the context-free models with as few as 1k UM + Wikipedia examples, and the difference increases with more examples, eventually reaching over 4% better on the test set than the next best model (Lematus 0-ch) when 10k UM + Wikipedia examples are used ($p < 0.001$) This indicates that the system can learn useful contextual cues even from unambiguous training examples.

Finally, Figure 2 gives a breakdown of Lematus 20-ch dev set accuracy for individual languages, showing that data augmentation helps consistently, although results suggest diminishing returns.

Data Augmentation in Medium Resource Setting To examine the extent to which augmented data can help in the medium resource setting of 10k continuous tokens of UDT used in previous work, we follow Bergmanis and Goldwater (2018) and train Lematus 20-ch models for all ten languages using the first 10k tokens of UDT and compare them with models trained on 10k tokens of UDT augmented with 10k UM types. To provide a better comparison of our results, we report both the type and the token level development set accuracy. First of all, Table 4 shows that training on 10k continuous tokens of UDT yields a token level accuracy that is about 8% higher than when using the 1k types of UDT augmented with 10k UM types—the best-performing data augmentation systems (see Table 3). Again, we believe this performance gap is due to the issues with the biased/incomplete data noted above. For example, we analyzed errors that were unique to the model trained on the Latvian augmented data and found that 41% of the errors were due to wrongly lemmatized words other than nouns, verbs, and adjectives—the three POSs with available inflection tables in UM. For instance, improperly lemmatized pronouns amounted to 14% of the errors on the Latvian dev set. Table 4 also shows that UM examples with Wikipedia contexts benefit lemmatization not only in the low but also the medium resource setting, yielding statistically significant type and token level accuracy gains over models trained on 10k UDT continuous tokens alone (for both Unseen and All $p < 0.001$).

Type accuracy:	**Ambig.**	**Unseen**	**All**
1k UDT+10k UM	49.5	66.7	70.6
10k UDT tok.	59.6	71.4	76.6
10k UDT tok.+10k UM	**60.8**	**75.1**	**80.1**
Token accuracy:	**Ambig.**	**Uns.**	**All**
1k UDT+10k UM	55.5	66.5	77.0
10k UDT tok.	**72.4**	72.5	85.3
10k UDT tok.+10k UM	72.3	**75.3**	**87.3**

Table 4: Lematus 20-ch average lemmatization type and token accuracy for all 10 languages, trained on 1k UDT types, 1k UDT augmented with 10k UM types, 10k UDT continuous tokens, or 10k UDT continuous tokens augmented with 10k UM types. Unless specified otherwise data consists of distinct types.

5 Conclusion

We proposed a training data augmentation method that combines the efficiency of type-based learning and the expressive power of a context-sensitive lemmatization model. The proposed method uses Wikipedia sentences to provide contextualized examples for unambiguous inflection-lemma pairs from UniMorph tables. These examples are noisy and biased, but nevertheless they improve lemmatization accuracy on all ten languages both in low (1k) and medium (10k) resource settings. In particular, we showed that context is helpful, both overall and especially on unseen words—the first work we know of to demonstrate improvements from context in a very low-resource setting.

References

Joakim Nivre et al. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Toms Bergmanis and Sharon Goldwater. 2018. Context Sensitive Neural Lemmatization with Lematus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training Data Augmentation for Low-Resource Morphological Inflection. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1491, Vancouver, Canada. Association for Computational Linguistics.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning Morphology with Morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared taskmorphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-Variable Modeling of String Transductions with Finite-State Methods. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1080–1089. Association for Computational Linguistics.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of postaggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 583–592.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of ACL*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017. Unlabeled Data for Morphological Generation With Character-Based Sequence-to-Sequence Models. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 76–81.

Faheem Kirefu. 2018. *Exploring Context Representations for Neural Lemmatisation*. Master's thesis, University of Edinburgh.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Peter Makarov and Simon Clematide. 2018a. Imitation Learning for Neural Morphological String Transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2877–2882.

Peter Makarov and Simon Clematide. 2018b. Neural Transition-based String Transduction for Limited-Resource Setting in Morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93.

Arya D McCarthy, Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2018. Marrying Universal Dependencies and Universal Morphology. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 91–101.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint Lemmatization and Morphological Tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal. Association for Computational Linguistics.

Lutz Prechelt. 1998. Early Stopping-but When? *Neural Networks: Tricks of the trade*, pages 553–553.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting Finite-State Transductions With Neural Context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde.

2017. Nematus: a toolkit for neural machine translation. *CoRR*, abs/1703.04357.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

A Lematus Training

Lematus is implemented using the *Nematus* machine translation toolkit[11]. We use default training parameters of Lematus as specified by Bergmanis and Goldwater (2018) except for early stopping with patience (Prechelt, 1998) which we increase to 20. Similar to Bergmanis and Goldwater (2018) we use the first epochs as a burn-in period, after which we validate the current model by its lemmatization exact match accuracy on the *first 3k instances* of development set and save this model if it performs better than the previous best model. We choose a burn-in period of 20 and validation interval of 5 epochs for models that we train on datasets up to 2k instances and a burn-in period of 10 and validation interval of 2 epochs for others. As we work with considerably smaller datasets than Bergmanis and Goldwater (2018) we reduce the effective model size and increase the rate of convergence by tying the input embeddings of the encoder, the decoder and the softmax output embeddings (Press and Wolf, 2017).

B Data Preparation

Wikipedia database dumps contain XML structured articles that are formatted using the *wikitext* markup language. To obtain wordforms in their sentence context we 1) use *WikiExtractor*[12] to extract plain text from Wikipedia database dumps, followed by scripts from *Moses* statistical machine translation system[13] (Koehn et al., 2007) to 2) split text into sentences (*split-sentences.perl*), and 3) extract separate tokens (*tokenizer.perl*). Finally, we shuffle the extracted sentences to encourage homogeneous type distribution across the entire text.

---

[11] `https://github.com/EdinburghNLP/nematus`

[12] `https://github.com/attardi/wikiextractor`

[13] `https://github.com/moses-smt/mosesdecoder`

C Result Breakdown by Language

	Type accuracy:	**Ambig.**	**Unseen**	**All**
Bulgarian	*Baseline*	**63.5**	39.3	45.0
	AE Aug Baseline	-	-	-
	HMAM	50.7	**61.0**	**63.5**
	Lematus 0-ch	45.9	51.3	55.7
	Lematus 20-ch	41.6	47.2	52.1
Czech	*Baseline*	38.1	31.2	33.0
	AE Aug Baseline	-	-	-
	HMAM	**45.2**	**66.8**	**66.7**
	Lematus 0-ch	40.7	59.9	60.1
	Lematus 20-ch	40.1	58.3	58.6
Estonian	*Baseline*	**51.0**	24.1	32.0
	AE Aug Baseline	-	-	-
	HMAM	39.9	41.2	46.2
	Lematus 0-ch	38.0	**42.8**	**47.6**
	Lematus 20-ch	47.8	39.9	45.9
Finnish	*Baseline*	46.4	21.3	26.1
	AE Aug Baseline	-	-	-
	HMAM	**44.7**	**48.0**	**50.4**
	Lematus 0-ch	44.4	41.5	44.9
	Lematus 20-ch	44.6	43.0	46.0
Latvian	*Baseline*	42.4	25.6	31.6
	AE Aug Baseline	-	-	-
	HMAM	44.0	**52.6**	**55.6**
	Lematus 0-ch	**47.1**	51.8	55.2
	Lematus 20-ch	43.1	52.1	55.2
Polish	*Baseline*	**42.9**	26.6	33.3
	AE Aug Baseline	-	-	-
	HMAM	41.2	**60.5**	62.4
	Lematus 0-ch	40.9	60.4	**62.6**
	Lematus 20-ch	35.5	59.7	62.2
Romanian	*Baseline*	27.6	34.9	40.0
	AE Aug Baseline	-	-	-
	HMAM	38.8	**55.1**	**57.9**
	Lematus 0-ch	**44.6**	50.2	54.5
	Lematus 20-ch	40.7	50.9	54.9
Russian	*Baseline*	43.0	34.9	39.0
	AE Aug Baseline	-	-	-
	HMAM	39.3	**66.4**	**67.0**
	Lematus 0-ch	42.3	63.4	65.4
	Lematus 20-ch	**44.6**	63.7	65.5
Swedish	*Baseline*	77.8	42.8	52.7
	AE Aug Baseline	-	-	-
	HMAM	58.5	**67.7**	**72.6**
	Lematus 0-ch	**59.5**	64.1	70.1
	Lematus 20-ch	54.0	62.6	68.1
Turkish	*Baseline*	58.8	26.6	33.6
	AE Aug Baseline	-	-	-
	HMAM	60.2	**69.6**	**72.3**
	Lematus 0-ch	**61.8**	64.7	68.4
	Lematus 20-ch	58.2	65.4	68.6

Table 5: Individual type level lemmatization accuracy for all 10 languages on development set, trained on **1k UDT types (no augmentation)** with contexts from Wikipedia. The numerically highest scores for each language are bold. For the summary of results see Table 3.

	Type accuracy:	**Ambig.**	**Unseen**	**All**
Bulgarian	*Baseline*	**64.3**	39.3	47.2
	AE Aug Baseline	49.2	60.3	63.3
	HMAM	41.4	**63.8**	**67.4**
	Lematus 0-ch	49.2	59.2	64.2
	Lematus 20-ch	53.3	61.7	66.2
Czech	*Baseline*	40.3	31.2	34.2
	AE Aug Baseline	42.5	63.6	63.6
	HMAM	42.5	**64.9**	**66.6**
	Lematus 0-ch	38.9	58.7	60.9
	Lematus 20-ch	**53.3**	61.7	66.2
Estonian	*Baseline*	**58.1**	24.1	34.4
	AE Aug Baseline	41.4	42.6	47.4
	HMAM	47.9	43.8	51.4
	Lematus 0-ch	48.1	45.2	52.5
	Lematus 20-ch	45.4	**46.3**	**52.9**
Finnish	*Baseline*	46.3	21.3	27.4
	AE Aug Baseline	44.3	43.3	45.5
	HMAM	45.6	55.7	59.3
	Lematus 0-ch	47.1	55.2	59.3
	Lematus 20-ch	**49.2**	**56.6**	**60.3**
Latvian	*Baseline*	45.4	25.6	33.7
	AE Aug Baseline	38.8	52.2	54.9
	HMAM	42.7	52.7	56.9
	Lematus 0-ch	45.2	51.8	56.4
	Lematus 20-ch	**48.6**	**56.3**	**59.2**
Polish	*Baseline*	**46.3**	26.6	35.4
	AE Aug Baseline	37.5	62.7	64.8
	HMAM	37.4	62.0	66.4
	Lematus 0-ch	45.3	62.3	67.1
	Lematus 20-ch	38.1	**66.9**	**70.3**
Romanian	*Baseline*	37.7	34.9	43.3
	AE Aug Baseline	42.6	53.2	57.0
	HMAM	48.3	**57.3**	62.7
	Lematus 0-ch	**51.1**	57.0	**62.8**
	Lematus 20-ch	49.0	55.9	61.2
Russian	*Baseline*	44.4	34.7	40.6
	AE Aug Baseline	42.6	65.7	67.1
	HMAM	43.2	66.1	68.3
	Lematus 0-ch	38.7	64.6	67.3
	Lematus 20-ch	**50.3**	**67.6**	**70.4**
Swedish	*Baseline*	**78.4**	42.8	54.2
	AE Aug Baseline	58.4	64.8	70.3
	HMAM	53.5	68.9	73.8
	Lematus 0-ch	48.8	**70.9**	**75.4**
	Lematus 20-ch	56.4	69.7	74.8
Turkish	*Baseline*	59.9	26.6	35.5
	AE Aug Baseline	58.6	66.9	69.9
	HMAM	56.1	67.0	69.4
	Lematus 0-ch	54.1	65.2	67.8
	Lematus 20-ch	**62.9**	**70.6**	**73.7**

Table 6: Individual type level lemmatization accuracy for all 10 languages on development set, trained on **1k UDT types plus 1k UM types** with contexts from Wikipedia. The numerically highest scores for each language are bold. For the summary of results see Table 3.

	Type accuracy:	**Ambig.**	**Unseen**	**All**
Bulgarian	*Baseline*	**67.2%**	39.3%	50.0%
	AE Aug Baseline	47.9%	62.6%	65.0%
	HMAM	44.3%	**68.2%**	**72.1%**
	Lematus 0-ch	43.1%	67.0%	71.1%
	Lematus 20-ch	50.4%	65.9%	70.0%
Czech	*Baseline*	43.0%	31.2%	36.8%
	AE Aug Baseline	43.2%	66.9%	66.6%
	HMAM	41.0%	61.9%	64.7%
	Lematus 0-ch	39.5%	61.6%	64.4%
	Lematus 20-ch	**42.6%**	**68.4%**	**69.7%**
Estonian	*Baseline*	62.9%	24.1%	37.1%
	AE Aug Baseline	43.1%	40.3%	45.3%
	HMAM	48.0%	44.9%	53.3%
	Lematus 0-ch	51.3%	45.2%	53.5%
	Lematus 20-ch	**48.6%**	**49.7%**	**56.3%**
Finnish	*Baseline*	49.4%	21.3%	30.3%
	AE Aug Baseline	42.5%	44.9%	47.6%
	HMAM	44.0%	58.4%	62.5%
	Lematus 0-ch	45.9%	60.8%	64.7%
	Lematus 20-ch	**52.5%**	**61.9%**	**65.5%**
Latvian	*Baseline*	**45.6%**	25.6%	35.9%
	AE Aug Baseline	39.6%	53.4%	55.3%
	HMAM	45.2%	52.3%	57.6%
	Lematus 0-ch	43.8%	54.5%	59.1%
	Lematus 20-ch	44.7%	**57.6%**	**61.1%**
Polish	*Baseline*	**50.4%**	26.6%	39.2%
	AE Aug Baseline	38.8%	64.1%	66.2%
	HMAM	41.6%	62.3%	68.4%
	Lematus 0-ch	43.3%	65.2%	70.7%
	Lematus 20-ch	40.3%	**69.7%**	**73.4%**
Romanian	*Baseline*	44.3%	34.9%	47.9%
	AE Aug Baseline	41.3%	54.9%	58.4%
	HMAM	50.2%	58.4%	65.6%
	Lematus 0-ch	51.4%	60.8%	67.2%
	Lematus 20-ch	**47.9%**	**62.6%**	**67.7%**
Russian	*Baseline*	48.5%	34.7%	44.4%
	AE Aug Baseline	42.1%	65.5%	66.5%
	HMAM	46.4%	65.5%	69.7%
	Lematus 0-ch	40.5%	64.4%	68.5%
	Lematus 20-ch	**42.7%**	**71.1%**	**73.8%**
Swedish	*Baseline*	**80.6%**	42.8%	58.0%
	AE Aug Baseline	58.7%	67.3%	71.4%
	HMAM	51.9%	**72.6%**	**77.7%**
	Lematus 0-ch	49.1%	71.4%	76.2%
	Lematus 20-ch	49.5%	72.3%	77.4%
Turkish	*Baseline*	61.8%	26.6%	37.9%
	AE Aug Baseline	62.8%	68.5%	71.2%
	HMAM	54.2%	63.6%	65.7%
	Lematus 0-ch	53.6%	63.9%	65.5%
	Lematus 20-ch	**67.1%**	**74.6%**	**77.2%**

Table 7: Individual type level lemmatization accuracy for all 10 languages on development set, trained on **1k UDT types plus 5k UM types** with contexts from Wikipedia. The numerically highest scores for each language are bold. For the summary of results see Table 3.

	Type accuracy:	**Ambig.**	**Unseen**	**All**
Bulgarian	*Baseline*	**66.2**	39.7	51.2
	AE Aug Baseline	48.1	62.8	65.4
	HMAM	42.7	**70.6**	**74.2**
	Lematus 0-ch	44.8	67.4	71.4
	Lematus 20-ch	50.6	68.4	72.5
Czech	*Baseline*	43.2	31.5	38.2
	AE Aug Baseline	44.9	68.0	68.1
	HMAM	41.1	61.7	64.7
	Lematus 0-ch	38.1	61.9	64.6
	Lematus 20-ch	**42.7**	**68.7**	**70.1**
Estonian	*Baseline*	**62.0**	24.3	37.8
	AE Aug Baseline	45.8	41.0	45.8
	HMAM	48.9	45.4	53.7
	Lematus 0-ch	46.6	45.7	53.8
	Lematus 20-ch	44.9	**51.3**	**57.5**
Finnish	*Baseline*	49.3	21.9	32.7
	AE Aug Baseline	41.6	46.0	48.5
	HMAM	45.0	59.2	62.7
	Lematus 0-ch	43.2	62.8	66.2
	Lematus 20-ch	**49.4**	**63.8**	**67.0**
Latvian	*Baseline*	46.7	25.8	36.9
	AE Aug Baseline	41.6	52.0	54.6
	HMAM	44.6	53.8	59.0
	Lematus 0-ch	42.6	55.3	59.7
	Lematus 20-ch	**47.7**	**60.6**	**64.0**
Polish	*Baseline*	**48.7**	27.0	42.1
	AE Aug Baseline	36.8	64.3	65.4
	HMAM	44.0	60.9	66.4
	Lematus 0-ch	46.2	67.2	72.4
	Lematus 20-ch	42.0	**71.2**	**75.1**
Romanian	*Baseline*	43.7	35.5	49.6
	AE Aug Baseline	41.0	54.3	57.2
	HMAM	45.6	56.8	63.7
	Lematus 0-ch	**50.3**	61.7	67.8
	Lematus 20-ch	49.5	**63.4**	**68.7**
Russian	*Baseline*	**50.2**	35.4	47.1
	AE Aug Baseline	46.8	65.6	67.0
	HMAM	39.1	64.6	68.2
	Lematus 0-ch	38.7	64.6	67.3
	Lematus 20-ch	47.3	**71.2**	**74.7**
Swedish	*Baseline*	**77.3**	43.0	59.9
	AE Aug Baseline	58.8	66.9	71.7
	HMAM	47.3	73.0	77.7
	Lematus 0-ch	55.5	74.1	78.6
	Lematus 20-ch	55.6	**75.1**	**79.1**
Turkish	*Baseline*	62.4	27.1	39.5
	AE Aug Baseline	57.9	67.7	70.3
	HMAM	55.9	62.4	64.8
	Lematus 0-ch	49.1	60.7	62.5
	Lematus 20-ch	**65.8**	**73.6**	**76.8**

Table 8: Individual type level lemmatization accuracy for all 10 languages on development set, trained on **1k UDT types plus 10k UM types** with contexts from Wikipedia. The numerically highest scores for each language are bold. For the summary of results see Table 3.

		Type level accuracy:			**Token level accuracy:**		
	Training data	**Ambig.**	**Unseen**	**All**	**Ambig.**	**Unseen**	**All**
Bulgarian	10k UDT tok.	**62.3**	75.7	80.1	72.3	75.7	89.5
	10k UDT tok. + 10k UM types	62.2	**78.7**	**83.6**	**73.3**	**78.1**	**91.0**
Czech	10k UDT tok.	49.7	76.4	77.8	**80.7**	77.7	88.3
	10k UDT tok. + 10k UM types	**52.4**	**78.3**	**80.4**	80.0	**80.0**	**89.6**
Estonian	10k UDT tok.	65.3	54.0	64.5	80.1	54.3	76.8
	10k UDT tok. + 10k UM types	**65.9**	**63.4**	**72.6**	**81.5**	**64.2**	**82.4**
Finnish	10k UDT tok.	**60.7**	60.1	66.5	**73.8**	62.4	78.2
	10k UDT tok. + 10k UM types	57.8	**63.7**	**69.4**	70.3	**66.0**	**79.8**
Latvian	10k UDT tok.	57.5	70.9	75.6	69.2	70.5	82.6
	10k UDT tok. + 10k UM types	**58.9**	**73.6**	**77.8**	**70.2**	**73.8**	**84.4**
Polish	10k UDT tok.	**59.8**	78.7	83.6	**76.5**	78.8	89.5
	10k UDT tok. + 10k UM types	57.4	**81.2**	**86.1**	71.3	**81.4**	**90.9**
Romanian	10k UDT tok.	51.7	61.1	66.6	54.1	60.6	79.1
	10k UDT tok. + 10k UM types	**57.1**	**68.2**	**74.2**	**60.7**	**68.2**	**83.9**
Russian	10k UDT tok.	**64.4**	80.5	83.5	**65.9**	80.8	88.5
	10k UDT tok. + 10k UM types	61.1	**82.6**	**85.9**	59.9	**82.7**	**89.8**
Swedish	10k UDT tok.	63.2	74.9	80.9	78.5	73.6	89.6
	10k UDT tok. + 10k UM types	**65.1**	**78.4**	**83.7**	**79.0**	**75.9**	**90.4**
Turkish	10k UDT tok.	64.2	82.1	87.1	73.1	81.8	91.2
	10k UDT tok. + 10k UM types	69.9	**82.9**	**87.3**	**76.9**	**82.7**	**91.5**

Table 9: Individual type and token level lemmatization accuracy for all 10 languages on development set for Lematus 20-ch models trained on **10k UDT tokens** and **10k UDT tokens plus 10k UM types** with contexts from Wikipedia. The numerically highest scores for each language are bold. For the summary of results see Table 4.

A Structural Probe for Finding Syntax in Word Representations

John Hewitt
Stanford University
`johnhew@stanford.edu`

Christopher D. Manning
Stanford University
`manning@stanford.edu`

Abstract

Recent work has improved our ability to detect linguistic knowledge in word representations. However, current methods for detecting syntactic knowledge do not test whether syntax trees are represented in their entirety. In this work, we propose a *structural probe*, which evaluates whether syntax trees are embedded in a linear transformation of a neural network's word representation space. The probe identifies a linear transformation under which squared L2 distance encodes the distance between words in the parse tree, and one in which squared L2 norm encodes depth in the parse tree. Using our probe, we show that such transformations exist for both ELMo and BERT but not in baselines, providing evidence that entire syntax trees are embedded implicitly in deep models' vector geometry.

1 Introduction

As pretrained deep models that build contextualized representations of language continue to provide gains on NLP benchmarks, understanding what they learn is increasingly important. To this end, probing methods are designed to evaluate the extent to which representations of language encode particular knowledge of interest, like part-of-speech (Belinkov et al., 2017), morphology (Peters et al., 2018a), or sentence length (Adi et al., 2017). Such methods work by specifying a *probe* (Conneau et al., 2018; Hupkes et al., 2018), a supervised model for finding information in a representation.

Of particular interest, both for linguistics and for building better models, is whether deep models' representations encode syntax (Linzen, 2018). Despite recent work (Kuncoro et al., 2018; Peters et al., 2018b; Tenney et al., 2019), open questions remain as to whether deep contextual models encode entire parse trees in their word representations.

In this work, we propose a *structural probe*, a simple model which tests whether syntax trees are consistently embedded in a linear transformation of a neural network's word representation space. Tree structure is embedded if the transformed space has the property that squared L2 distance between two words' vectors corresponds to the number of edges between the words in the parse tree. To reconstruct edge directions, we hypothesize a linear transformation under which the squared L2 norm corresponds to the depth of the word in the parse tree. Our probe uses supervision to find the transformations under which these properties are best approximated for each model. If such transformations exist, they define inner products on the original space under which squared distances and norms encode syntax trees – even though the models being probed were never given trees as input or supervised to reconstruct them. This is a structural property of the word representation space, akin to vector offsets encoding word analogies (Mikolov et al., 2013). Using our probe, we conduct a targeted case study, showing that ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) representations embed parse trees with high consistency in contrast to baselines, and in a low-rank space.[1]

In summary, we contribute a simple structural probe for finding syntax in word representations (§2), and experiments providing insights into and examples of how a low-rank transformation recovers parse trees from ELMo and BERT representations (§3,4). Finally, we discuss our probe and limitations in the context of recent work (§5).

2 Methods

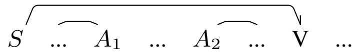
Our goal is to design a simple method for testing whether a neural network embeds each sentence's

---

[1] We release our code at `https://github.com/john-hewitt/structural-probes`.

Proceedings of NAACL-HLT 2019, pages 4129–4138
Minneapolis, Minnesota, June 2 - June 7, 2019.

dependency parse tree in its contextual word representations – a structural hypothesis. Under a reasonable definition, to embed a graph is to learn a vector representation of each node such that geometry in the vector space—distances and norms—approximates geometry in the graph (Hamilton et al., 2017). Intuitively, why do parse tree distances and depths matter to syntax? The distance metric—the path length between each pair of words—recovers the tree T simply by identifying that nodes u, v with distance $d_T(u, v) = 1$ are neighbors. The node with greater norm—depth in the tree—is the child. Beyond this identity, the distance metric explains hierarchical behavior. For example, the ability to perform the classic hierarchy test of subject-verb number agreeement (Linzen et al., 2016) in the presence of "attractors" can be explained as the verb (V) being closer in the tree to its subject (S) than to any of the attractor nouns:



Intuitively, if a neural network embeds parse trees, it likely will not use its entire representation space to do so, since it needs to encode many kinds of information. Our probe learns a linear transformation of a word representation space such that the transformed space embeds parse trees across all sentences. This can be interpreted as finding the part of the representation space that is used to encode syntax; equivalently, it is finding the distance on the original space that best fits the tree metrics.

2.1 The structural probe

In this section we provide a description of our proposed structural probe, first discussing the distance formulation. Let $\mathcal{M}$ be a model that takes in a sequence of n words $w^\ell_{1:n}$ and produces a sequence of vector representations $\mathbf{h}^\ell_{1:n}$, where ℓ identifies the sentence. Starting with the dot product, recall that we can define a family of inner products, $\mathbf{h}^T A \mathbf{h}$, parameterized by any positive semi-definite, symmetric matrix $A \in \mathbb{S}^{m\times m}_{+}$. Equivalently, we can view this as specifying a linear transformation $B \in \mathbb{R}^{k\times m}$, such that $A = B^T B$. The inner product is then $(B\mathbf{h})^T(B\mathbf{h})$, the norm of $\mathbf{h}$ once transformed by B. Every inner product corresponds to a distance metric. Thus, our family of squared distances is defined as:

$$d_B(\mathbf{h}^\ell_i, \mathbf{h}^\ell_j)^2 = \left(B(\mathbf{h}^\ell_i - \mathbf{h}^\ell_j)\right)^T \left(B(\mathbf{h}^\ell_i - \mathbf{h}^\ell_j)\right) \tag{1}$$

where i, j index the word in the sentence.[2] The parameters of our probe are exactly the matrix B, which we train to recreate the tree distance between all pairs of words (w^ℓ_i, w^ℓ_j) in all sentences T^ℓ in the training set of a parsed corpus. Specifically, we approximate through gradient descent:

$$\min_B \sum_\ell \frac{1}{|s^\ell|^2} \sum_{i,j} \left| d_{T^\ell}(w^\ell_i, w^\ell_j) - d_B(\mathbf{h}^\ell_i, \mathbf{h}^\ell_j)^2 \right|$$

where $|s^\ell|$ is the length of the sentence; we normalize by the square since each sentence has $|s^\ell|^2$ word pairs.

2.2 Properties of the structural probe

Because our structural probe defines a valid distance metric, we get a few nice properties for free. The simplest is that distances are guaranteed nonnegative and symmetric, which fits our probing task. Perhaps most importantly, the probe tests the concrete claim that there exists an inner product on the representation space whose squared distance—a global property of the space—encodes syntax tree distance. This means that the model not only encodes which word is governed by which other word, but each word's proximity to every other word in the syntax tree.[3] This is a claim about the structure of the representation space, akin to the claim that analogies are encoded as vector-offsets in uncontextualized word embeddings (Mikolov et al., 2013). One benefit of this is the ability to query the nature of this structure: for example, the dimensionality of the transformed space (§ 4.1).

2.3 Tree depth structural probes

The second tree property we consider is the *parse depth* $\|w_i\|$ of a word w_i, defined as the number of edges in the parse tree between w_i and the root of the tree. This property is naturally represented as a norm – it imposes a total order on the words in the sentence. We wish to probe to see if there exists a squared norm on the word representation

---

[2] As noted in Eqn 1, in practice, we find that approximating the parse tree distance and norms with the *squared* vector distances and norms consistently performs better. Because a distance metric and its square encode exactly the same parse trees, we use the squared distance throughout this paper. Also strictly, since A is not positive definite, the inner product is indefinite, and the distance a pseudometric. Further discussion can be found in our appendix.

[3] Probing for distance instead of headedness also helps avoid somewhat arbitrary decisions regarding PP headedness, the DP hypothesis, and auxiliaries, letting the representation "disagree" on these while still encoding roughly the same global structure. See Section 5 for more discussion.

Method	Distance UUAS	Distance DSpr.	Depth Root%	Depth NSpr.
Linear	48.9	0.58	2.9	0.27
ELMo0	26.8	0.44	54.3	0.56
Decay0	51.7	0.61	54.3	0.56
Proj0	59.8	0.73	64.4	0.75
ELMo1	77.0	0.83	86.5	0.87
BERTbase7	79.8	0.85	88.0	0.87
BERTlarge15	**82.5**	0.86	89.4	0.88
BERTlarge16	81.7	**0.87**	**90.1**	**0.89**

Table 1: Results of structural probes on the PTB WSJ test set; baselines in the top half, models hypothesized to encode syntax in the bottom half. For the distance probes, we show the Undirected Unlabeled Attachment Score (UUAS) as well as the average Spearman correlation of true to predicted distances, DSpr. For the norm probes, we show the root prediction accuracy and the average Spearman correlation of true to predicted norms, NSpr.

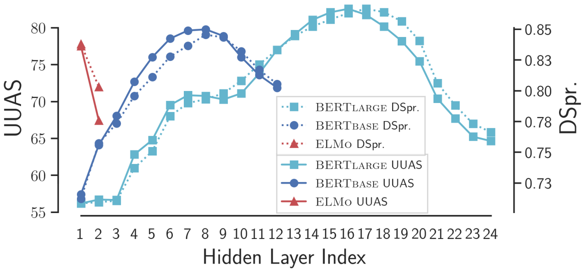


Figure 1: Parse distance UUAS and distance Spearman correlation across the BERT and ELMo model layers.

space that encodes this tree norm. We replace the vector distance function $d_B(\mathbf{h}_i, \mathbf{h}_j)$ with the squared vector norm $\|\mathbf{h}_i\|_B^2$, replacing Equation 1 with $\|\mathbf{h}_i\|_A = (B\mathbf{h}_i)^T(B\mathbf{h}_i)$ and training B to recreate $\|w_i\|$. Like the distance probe, this norm formulation makes a concrete claim about the structure of the vector space.

3 Experiments

Using our probe, we evaluate whether representations from ELMo and BERT, two popular English models pre-trained on language modeling-like objectives, embed parse trees according to our structural hypothesis. Unless otherwise specified, we permit the linear transformation B to be potentially full-rank (i.e., B is square.) Later, we explore what rank of transformation is actually necessary for encoding syntax (§ 4.1).

Representation models We use the 5.5B-word pre-trained ELMo weights for all ELMo representations, and both BERT-base (cased) and BERT-large (cased). The representations we evaluate are denoted ELMoK, BERTbaseK, BERTlargeK, where K indexes the hidden layer of the corresponding model. All ELMo and BERT-large layers are dimensionality 1024; BERT-base layers are dimensionality 768.

Data We probe models for their ability to capture the Stanford Dependencies formalism (de Marneffe et al., 2006), claiming that capturing most aspects of the formalism implies an understanding of English syntactic structure. To this end, we obtain fixed word representations for sentences of the parsing train/dev/test splits of the Penn Treebank (Marcus et al., 1993), with no pre-processing.[4]

Baselines Our baselines should encode features useful for training a parser, but not be capable of parsing themselves, to provide points of comparison against ELMo and BERT. They are as follows:

Linear : The tree resulting from the assumption that English parse trees form a left-to-right chain. A model that encodes the positions of words should be able to meet this baseline.

ELMo0 : Strong character-level word embeddings with no contextual information. As these representations lack even position information, we should be completely unable to find syntax trees embedded.

Decay0 : Assigns each word a weighted average of all ELMo0 embeddings in the sentence. The weight assigned to each word decays exponentially as $\frac{1}{2^d}$, where d is the linear distance between the words.

Proj0 : Contextualizes the ELMo0 embeddings with a randomly initialized BiLSTM layer of dimensionality identical to ELMo (1024), a surprisingly strong baseline for contextualization (Conneau et al., 2018).

3.1 Tree distance evaluation metrics

We evaluate models on how well the predicted distances between all pairs of words reconstruct gold parse trees and correlate with the parse trees' distance metrics. To evaluate tree reconstruction, we take each test sentence's predicted parse tree distances and compute the minimum spanning tree. We evaluate the predicted tree on undirected

---

[4] Since BERT constructs subword representations, we align subword vectors with gold Penn Treebank tokens, and assign each token the average of its subword representation. This thus represents a lower-bound on BERT's performance.

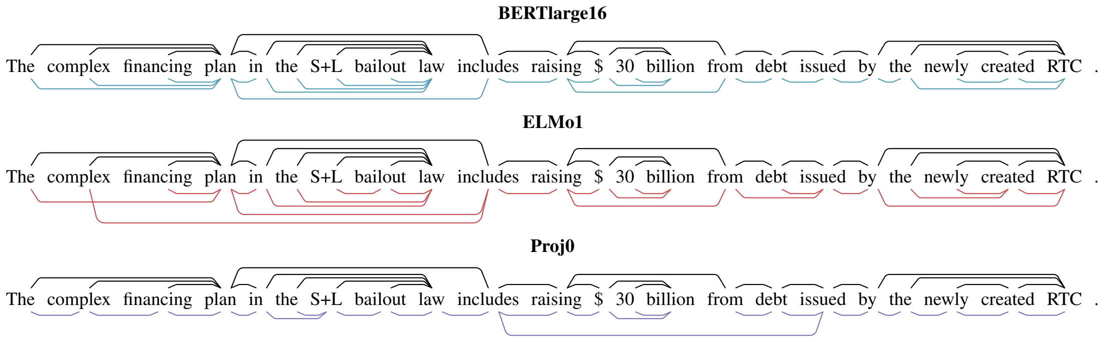


Figure 2: Minimum spanning trees resultant from predicted squared distances on BERTLARGE16 and ELMO1 compared to the best baseline, PROJ0. Black edges are the gold parse, above each sentence; blue are BERTLARGE16, red are ELMO1, and purple are PROJ0.

attachment score (UUAS)—the percent of undirected edges placed correctly—against the gold tree. For distance correlation, we compute the Spearman correlation between true and predicted distances for each word in each sentence. We average these correlations between all sentences of a fixed length, and report the macro average across sentence lengths 5–50 as the "distance Spearman (DSpr.)" metric.[5]

3.2 Tree depth evaluation metrics

We evaluate models on their ability to recreate the order of words specified by their depth in the parse tree. We report the Spearman correlation betwen the true depth ordering and the predicted ordering, averaging first between sentences of the same length, and then across sentence lengths 5–50, as the "norm Spearman (NSpr.)". We also evaluate models' ability to identify the root of the sentence as the least deep, as the "root%".[6]

4 Results

We report the results of parse distance probes and parse depth probes in Table 1. We first confirm that our probe can't simply "learn to parse" on top of any informative representation, unlike parser-based probes (Peters et al., 2018b). In particular, ELMO0 and DECAY0 fail to substantially outperform a right-branching-tree oracle that encodes the linear sequence of words. PROJ0, which has all of the representational capacity of ELMO1 but none of the training, performs the best among the baselines. Upon inspection, we found that our probe on PROJ0 improves over the linear hypothesis with mostly simple deviations from linearity, as visualized in Figure 2.

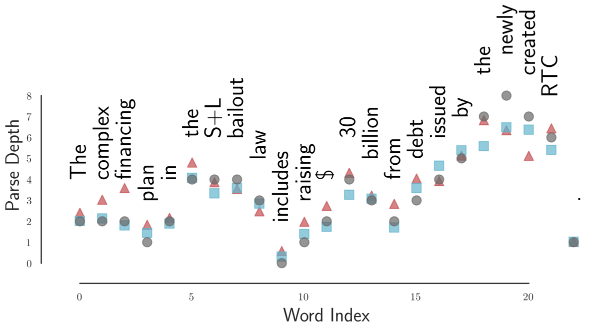


Figure 3: Parse tree depth according to the gold tree (black, circle) and the norm probes (squared) on ELMO1 (red, triangle) and BERTLARGE16 (blue, square).

We find surprisingly robust syntax embedded in each of ELMo and BERT according to our probes. Figure 2 shows the surprising extent to which a minimum spanning tree on predicted distances recovers the dependency parse structure in both ELMo and BERT. As we note however, the distance metric itself is a global notion; all pairs of words are trained to know their distance – not just which word is their head; Figure 4 demonstrates the rich structure of the true parse distance metric recovered by the predicted distances. Figure 3 demonstrates the surprising extent to which the depth in the tree is encoded by vector norm after the probe transformation. Between models, we find consistently that BERTLARGE performs better than BERTBASE, which performs better than ELMO.[7] We also find, as in Peters et al. (2018b), a clear difference in syntactic information between layers; Figure 1 reports the performance

---

[5] The 5–50 range is chosen to avoid simple short sentences as well as sentences so long as to be rare in the test data.

[6] In UUAS and "root%" evaluations, we ignore all punctuation tokens, as is standard.

[7] It is worthwhile to note that our hypotheses were developed while analyzing LSTM models like ELMo, and applied without modification on the self-attention based BERT models.

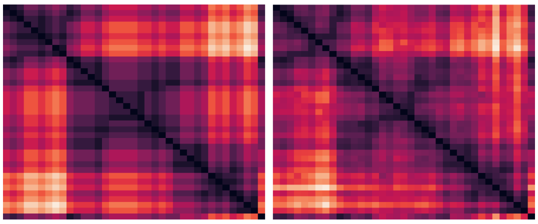


Figure 4: (left) Matrix representing gold tree distances between all pairs of words in a sentence, whose linear order runs top-to-bottom and left-to-right. Darker colors indicate close words, lighter indicate far. (right) The same distances as embedded by BERTLARGE16 (squared). More detailed graphs available in the Appendix.

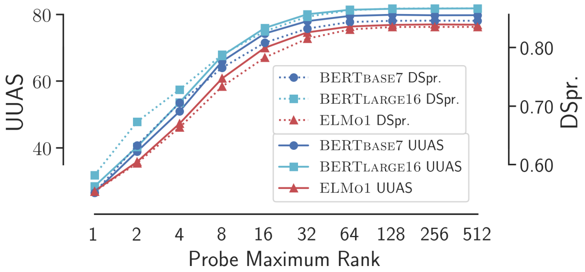


Figure 5: Parse distance tree reconstruction accuracy when the linear transformation is constrained to varying maximum dimensionality.

of probes trained on each layer of each system.

4.1 Analysis of linear transformation rank

With the result that there exists syntax-encoding vector structure in both ELMo and BERT, it is natural to ask how compactly syntactic information is encoded in the vector space. We find that in both models, the effective rank of linear transformation required is surprisingly low. We train structural probes of varying k, that is, specifying a matrix $B \in \mathbb{R}^{k \times m}$ such that the transformed vector $B\mathbf{h}$ is in $\mathbb{R}^k$. As shown in Figure 5, increasing k beyond 64 or 128 leads to no further gains in parsing accuracy. Intuitively, larger k means a more expressive probing model, and a larger fraction of the representational capacity of the model being devoted to syntax. We also note with curiosity that the three models we consider all seem to require transformations of approximately the same rank; we leave exploration of this to exciting future work.

5 Discussion & Conclusion

Recent work has analyzed model behavior to determine if a model understands hierarchy and other linguistic phenomena (Linzen, 2018; Gulordava et al., 2018; Kuncoro et al., 2018; Linzen and Leonard, 2018; van Schijndel and Linzen, 2018; Tang et al., 2018; Futrell et al., 2018). Our work extends the literature on linguistic probes, found at least in (Peters et al., 2018b; Belinkov et al., 2017; Blevins et al., 2018; Hupkes et al., 2018). Conneau et al. (2018) present a task similar to our parse depth prediction, where a sentence representation vector is asked to classify the maximum parse depth ever achieved in the sentence. Tenney et al. (2019) evaluates a complementary task to ours, training probes to learn the *labels* on structures when the gold structures themselves are given. Peters et al. (2018b) evaluates the extent to which constituency trees can be extracted from hidden states, but uses a probe of considerable complexity, making less concrete hypotheses about how the information is encoded.

Probing tasks and limitations Our reviewers rightfully noted that one might just probe for headedness, as in a bilinear graph-based dependency parser. More broadly, a deep neural network probe of some kind is almost certain to achieve higher parsing accuracies than our method. Our task and probe construction are designed not to test for some notion of syntactic knowledge broadly construed, but instead for an extremely strict notion where all pairs of words know their syntactic distance, and this information is a global structural property of the vector space. However, this study is limited to testing that hypothesis, and we foresee future probing tasks which make other tradeoffs between probe complexity, probe task, and hypotheses tested.

In summary, through our structural probes we demonstrate that the structure of syntax trees emerges through properly defined distances and norms on two deep models' word representation spaces. Beyond this actionable insight, we suggest our probe may be useful for testing the existence of different types of graph structures on any neural representation of language, an exciting avenue for future work.

6 Acknowledgements

We would like to acknowledge Urvashi Khandelwal and Tatsunori B. Hashimoto for formative advice in early stages, Abigail See, Kevin Clark, Siva Reddy, Drew A. Hudson, and Roma Patel for helpful comments on drafts, and Percy Liang, for guidance on rank experiments. We would also like to thank the reviewers, whose helpful comments led to increased clarity and extra experiments. This research was supported by a gift from Tencent.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. RNNs as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1195–1205.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Tal Linzen. 2018. What can linguistics and deep learning contribute to each other? *arXiv preprint arXiv:1809.04179*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Tal Linzen and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 692–697. Cognitive Science Society, Austin, TX.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Marten van Schijndel and Tal Linzen. 2018. Modeling garden path effects without explicit hierarchical syntax. In Tim Rogers, Marina Rau, Jerry Zhu, and Chuck Kalish, editors, *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2600–2605. Cognitive Science Society, Austin, TX.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

A Appendix: Implementation Details

A.1 Squared L2 distance vs. L2 distance

In Section 2.2, we note that while our distance probe specifies a distance metric, we recreate it with a squared vector distance; likewise, while our norm probe specifies a norm, we recreate it with a squared vector norm. We found this to be important for recreating the exact parse tree distances and norms. This does mean that in order to recreate the exact scalar values of the parse tree structures, we need to use the squared vector quantities. This may be problematic, since for example squared distance doesn't obey the triangle inequality, whereas a valid distance metric does.

However, we note that in terms of the graph structures encoded, distance and squared distance are identical. After training with the squared vector distance, we can square-root the predicted quantities to achieve a distance metric. The relative ordering between all pairs of words will be unchanged; the same tree is encoded either way, and none of our quantitative metrics will change; however, the exact scalar distances will differ from the true tree distances.

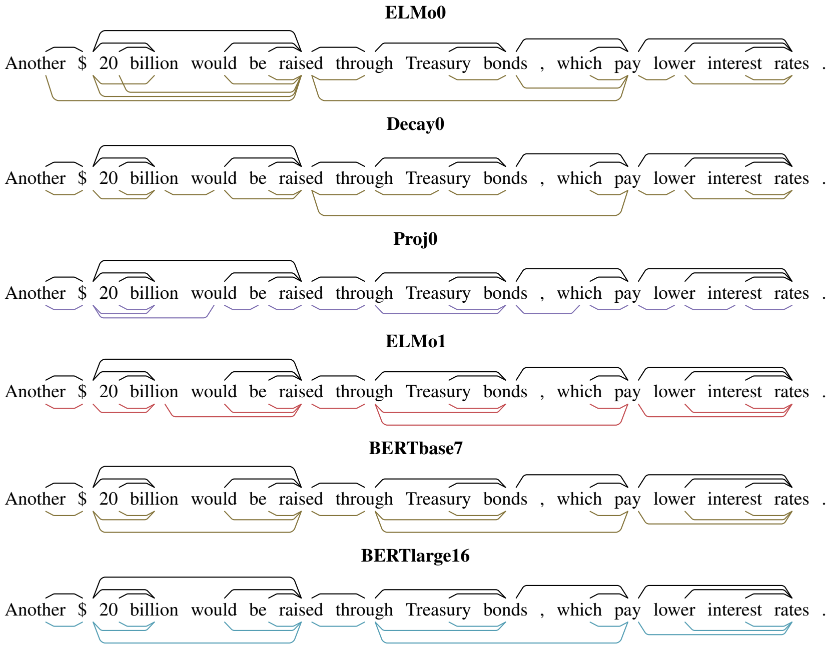
This raises a question for future work as to why squared distance works better than distance, and beyond that, what function of the L2 distance (or perhaps, what L^p distance) would best encode tree distances. It is possibly related to the gradients of the loss with respect to the function of the distance, as well as how amenable the function is to matching the exact scalar values of the tree distances.

A.2 Probe training details

All probes are trained to minimize L1 loss of the predicted squared distance or squared norm w.r.t. the true distance or norm. Optimization is performed using the Adam optimizer (Kingma and Ba, 2014) initialized at learning rate 0.001, with $\beta_1 = .9, \beta_2 = .999, \epsilon = 10^{-8}$. Probes are trained to convergence, up to 40 epochs, with a batch size of 20. For depth probes, loss is summed over all predictions in a sentence, normalized by the length of the sentence, and then summed over all sentences in a batch before a gradient step is taken. For distance probes, normalization is performed by the square of the length of the sentence. At each epoch, dev loss is computed; if the dev loss does not achieve a new minimum, the optimizer is reset (no momentum terms are kept) with an initial learning rate multiplied by 0.1. All models were implemented in both DyNet (Neubig et al., 2017), and in PyTorch (Paszke et al., 2017).

B Appendix: Extra examples

In this section we provide additional examples of model behavior, including baseline model behavior, across parse distance prediction and parse depth prediction. In Figure 6 and Figure 7, we present a single sentence with dependency trees as extracted from many of our models and baselines. In Figure 8, we present tree depth predictions on a complex sentence from ELMO1, BERTLARGE16, and our baseline PROJ0. Finally, in Figure 9, we present gold parse distances and predicted squared parse distances between all pairs of words in large, high-resolution format.


ELMo0

Decay0

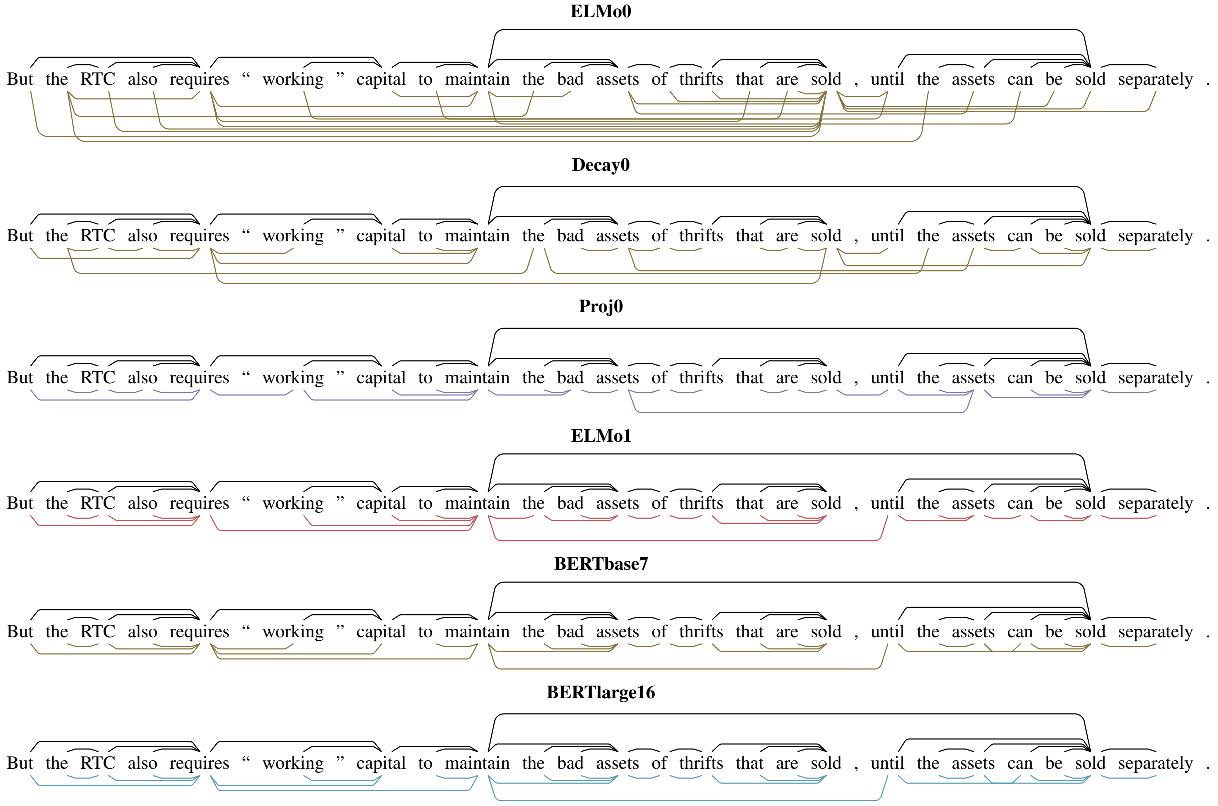
Proj0

ELMo1

BERTbase7

BERTlarge16

Figure 6: A relatively simple sentence, and the minimum spanning trees extracted by various models.


ELMo0

Decay0

Proj0

ELMo1

BERTbase7

BERTlarge16

Figure 7: A complex sentence, and the minimum spanning trees extracted by various models.

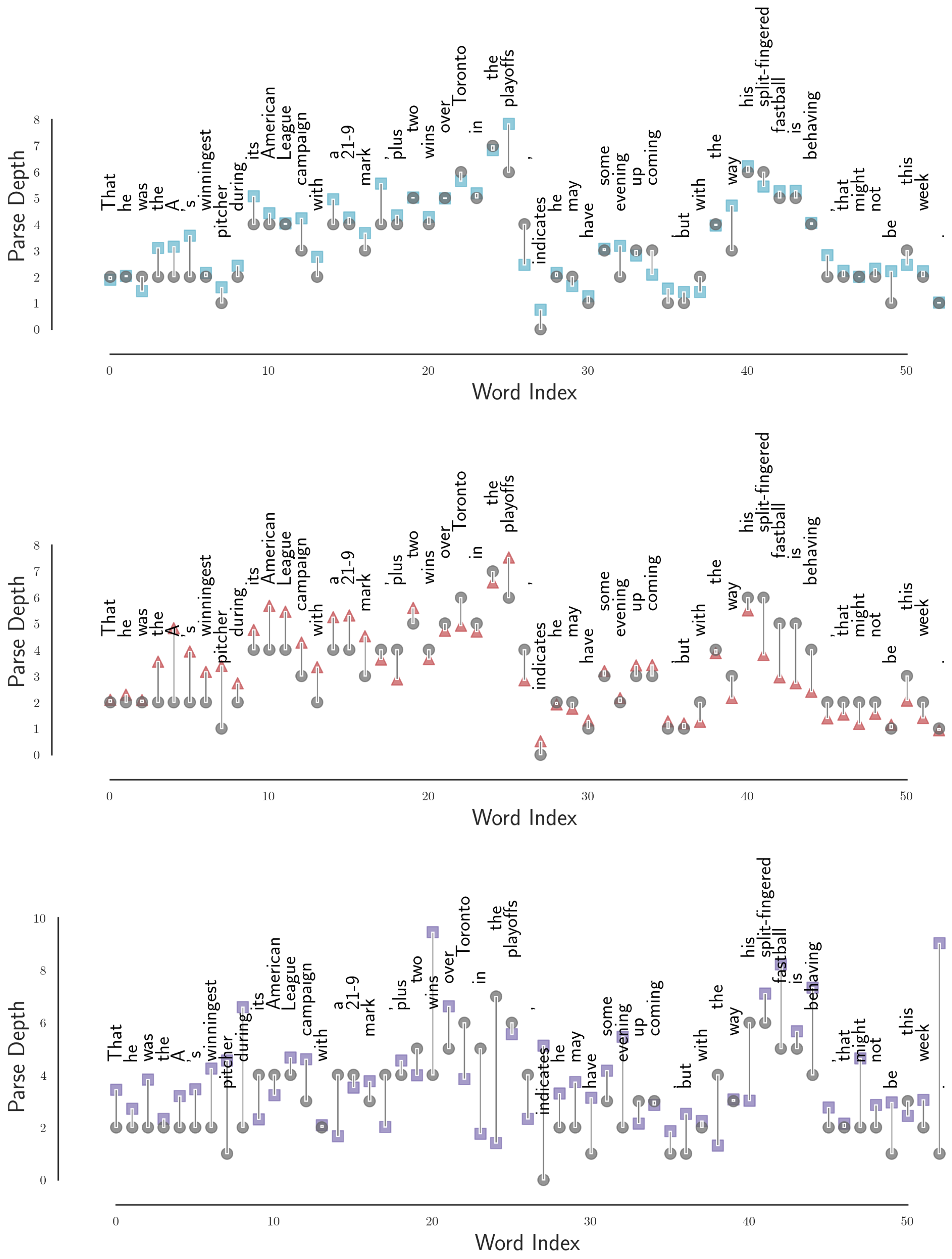


Figure 8: A long sentence with gold dependency parse depths (grey) and dependency parse depths (squared) as extracted by BERTLARGE16 (blue, top), ELMO1 (red, middle), and the baseline PROJ0 (purple, bottom). Note the non-standard subject, “that he was the A’s winningest pitcher”.

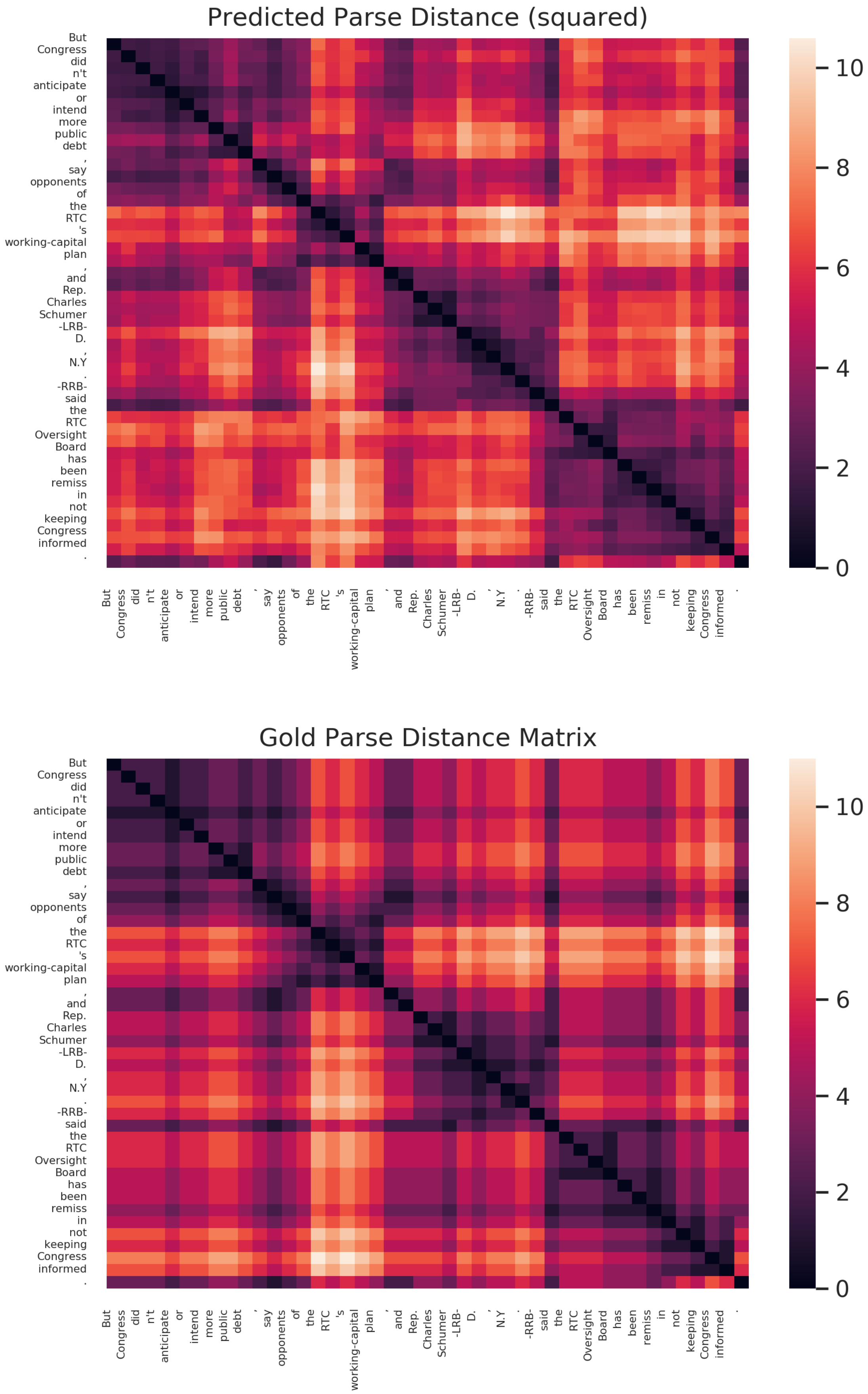


Figure 9: The distance graphs defined by the gold parse distances on a sentence (below) and as extracted from BERTLARGE16 (above, squared).

CNM: An Interpretable Complex-valued Network for Matching

Qiuchi Li [*]
University of Padua
Padua, Italy
qiuchili@dei.unipd.it

Benyou Wang [*]
University of Padua
Padua, Italy
wang@dei.unipd.it

Massimo Melucci [†]
University of Padua
Padua, Italy
melo@dei.unipd.it

Abstract

This paper seeks to model human language by the mathematical framework of quantum physics. With the well-designed mathematical formulations in quantum physics, this framework unifies different linguistic units in a single complex-valued vector space, e.g. words as particles in quantum states and sentences as mixed systems. A complex-valued network is built to implement this framework for semantic matching. With well-constrained complex-valued components, the network admits interpretations to explicit physical meanings. The proposed complex-valued network for matching (CNM)[1] achieves comparable performances to strong CNN and RNN baselines on two benchmarking question answering (QA) datasets.

1 Introduction

There is a growing concern on the interpretability of neural networks. Along with the increasing power of neural networks comes the challenge of interpreting the numerical representation of network components into human-understandable language. Lipton (2018) points out two important factors for a model to be interpretable, namely *post-hoc interpretability* and *transparency*. The former refers to explanations of why a model works after it is executed, while the latter concerns self-explainability of components through some mechanisms in the designing phase of the model.

We seek inspirations from quantum physics to build transparent and post-hoc interpretable networks for modeling human language. The emerging research field of cognition suggests that there exist quantum-like phenomena in human cognition (Aerts and Sozzo, 2014), especially language understanding (Bruza et al., 2008). Intuitively, a sentence can be treated as a physical system with multiple words (like particles), and these words are usually polysemous (superposed) and correlated (entangled) with each other. Motivated by these existing works, we aim to investigate the following Research Question (RQ).

RQ1*: Is it possible to model human language with the mathematical framework of quantum physics?*

Towards this question, we build a novel quantum-theoretic framework for modeling language, in an attempt to capture the quantumness in the cognitive aspect of human language. The framework models different linguistic units as quantum states with the adoption of quantum probability (QP), which is the mathematical framework of quantum physics that models uncertainly on a uniform *Semantic Hilbert Space* (SHS).

Complex values are crucial in the mathematical framework of characterizing quantum physics. In order to preserve physical properties, the linguistic units have to be represented as complex vectors or matrices. This naturally gives rise to another research question:

RQ2*: Can we benefit from the complex-valued representation of human language in a real natural language processing (NLP) scenario?*

To this end, we formulate a linguistic unit as a complex-valued vector, and link its length and direction to different physical meanings: the length represents the relative weight of the word while the direction is viewed as a superposition state. The superposition state is further represented in an amplitude-phase manner, with amplitudes corresponding to the lexical meaning and phases implicitly reflecting the higher-level semantic aspects such as polarity, ambiguity or emotion.

In order to evaluate the above framework, we implement it as a complex-valued network (CNM) for semantic matching. The network is applied to the question answering task, which is the most

---

[*] Equal Contribution

[†] Corresponding Author

[1] https://github.com/wabyking/qnn.git

Proceedings of NAACL-HLT 2019, pages 4139–4148
Minneapolis, Minnesota, June 2 - June 7, 2019.

typical matching task that aims at selecting the best answer for a question from a pool of candidates. In order to facilitate local matching with n-grams of a sentence pair, we design a local matching scheme in CNM. Most of State-of-the-art QA models are mainly based on Convolution Neural Network (CNN), Recurrent Neural Network (RNN) and many variants thereof (Wang and Nyberg, 2015; Yang et al., 2016; Hu et al., 2014; Tan et al., 2015). However, with opaque structures of convolutional kernels and recurrent cells, these models are hard to understand for humans. We argue that our model is advantageous in terms of interpretability.

Our proposed CNM is transparent in that it is designed in alignment with quantum physics. Experiments on benchmarking QA datasets show that CNM has comparable performance to strong CNN and RNN baselines, whilst admitting post-hoc interpretations to human-understandable language. We therefore answer RQ1 by claiming that it is possible to model human language with the proposed quantum-theoretical framework in this paper. Furthermore, an ablation study shows that the complex-valued word embedding performs better than its real counterpart, which allows us to answer RQ2 by claiming that we benefit from the complex-valued representation of natural language on the QA task.

2 Background

Here we briefly introduce quantum probability and discuss a relevant work on quantum-inspired framework for QA.

2.1 Quantum Probability

Quantum probability provides a sound explanation for the phenomena and concepts of quantum mechanics, by formulating events as subspaces in a vector space with projective geometry.

2.1.1 Quantum Superposition

Quantum Superposition is one of the fundamental concepts in Quantum Physics, which describes the uncertainty of a single particle. In the micro world, a particle like a photon can be in multiple mutual-exclusive basis states simultaneously with a probability distribution. In a two-dimensional example, two basis vectors are denoted as $|0\rangle$ and $|1\rangle$[2]. *Superposition* is implemented to model a general state which is a linear combination of basis vectors with complex-valued weights such that

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \qquad (1)$$

where α_0 and α_1 are complex scalars satisfying $0 \leq |\alpha_0|^2 \leq 1, 0 \leq |\alpha_1|^2 \leq 1$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. It follows that $|\phi\rangle$ is defined over the complex field. When α_0 and α_1 are non-zero values, the state $|\phi\rangle$ is said to be a superposition of the states $|0\rangle$ and $|1\rangle$, and the scalars α_0 and α_1 denote the probability amplitudes of the superposition.

2.1.2 Measurement

The uncertainty of an ensemble system with multiple particles is encapsulated as a mixed state, represented by a positive semi-definite matrix with unitary trace called *density matrix*: $\rho = \sum_i^m |\phi_i\rangle \langle\phi_i|$, where $\{|\phi_i\rangle\}_{i=0}^m$ are pure states like Eq. 1. In order to infer the probabilistic properties of ρ in the state space, Gleason's theorem (Gleason, 1957; Hughes, 1992) is used to calculate probability to observe x through projection measurements $|x\rangle \langle x|$ that is a rank-one projector denoted as a outer product of $|x\rangle$.

$$p_x(\rho) = \langle x| \rho |x\rangle = tr(\rho |x\rangle \langle x|) \qquad (2)$$

The measured probability $p_x(\rho)$ is a non-negative real-valued scalar, since both ρ and $|x\rangle \langle x|$ are Hermitian. The unitary trace property guarantees $\sum_{x \in X} p_x(\rho) = 1$ for X being a set of orthogonal basis states.

2.2 Neural Network based Quantum-like Language Model (NNQLM)

Based on the density matrices representation for documents in information retrieval (Van Rijsbergen, 2004; Sordoni et al., 2013), Zhang et al. (2018a) built a neural network with density matrix for question answering. This Neural Network based Quantum Language Model (NNQLM) embeds a word as a unit vector and a sentence as a real-valued density matrix. The distance between a pair of density matrices is obtained by extracting features of their matrix multiplication in two ways: NNQLM-I directly takes the trace of the resulting matrix, while NNQLM-II applies convolutional structures on top of the matrix to determine whether the pair of sentences match or not.

NNQLM is limited in that it does not make proper use of the full potential of probabilistic

---

[2] We here adopt the widely used Dirac notations in quantum probability, in which a *unit* vector $\vec{\mu}$ and its transpose $\vec{\mu}^T$ are denoted as a ket $|u\rangle$ and a bra $\langle u|$ respectively.

property of a density matrices.By treating density matrices as ordinary real vectors (NNQLM-I) or matrices (NNQLM-II), the full potential with complex-valued formulations is largely ignored. Meanwhile, adding convolutional layers on top of a density matrix is more of an empirical workaround than an implementation of a theoretical framework.

In contrast, a complex-valued matching network is built on top of a quantum-theoretical framework for natural language. In particular, an indirect way to measure the distance between two density matrices through trainable measurement operations, which makes advantage of the probabilistic properties of density matrices and also provides flexible matching score driven by training data.

3 Semantic Hilbert Space

Here we introduce the Semantic Hilbert Space $\mathcal{H}$ defined on a complex vector space $\mathcal{C}^n$, and three different linguistic units, namely sememes, words and word combinations on the space. The concept of semantic measurement is introduced at last.

Sememes. We assume $\mathcal{H}$ is spanned by the set of orthogonal basis states $\{|e_j\rangle\}_{j=1}^n$ for *sememes*, which are the minimum semantic units of word meanings in language universals (Goddard and Wierzbicka, 1994). The unit state $|e_j\rangle$ can be seen as a one-hot vector, i.e., the j-th element in $|e_j\rangle$ is one while other elements are zero, in order to obtain a set of orthogonal unit states. Semantic units with larger granularities are based on the set of sememe basis.

Words. Words are composed of sememes in superposition. Each word w is a superposition over all sememes $\{|e_j\rangle\}_{j=1}^n$, or equivalently a unit-length vector on $\mathcal{H}$:

$$|w\rangle = \sum_{j=1}^{n} r_j e^{i\phi_j} |e_j\rangle, \tag{3}$$

i is the imaginary number with $i^2 = -1$. In the above expression, $\{r_j\}_{j=1}^n$ are non-negative real-valued amplitudes satisfying $\sum_{j=1}^n {r_j}^2$ =1 and $\phi_j \in [-\pi, \pi]$ are the corresponding complex phases. In comparison to Eq. 1, $\{r_j e^{i\phi_j}\}_{j=0}^n$ are the polar form representation of the complex-valued scalars $\{\alpha_j\}_{j=0}^1$.

Word Combinations. We view a combination of words (e.g. phrase, n-gram, sentence or document) as a mixed system composed of individual words, and its representation is computed as follows:

$$\rho = \sum_{j}^{m} \frac{1}{m} |w_j\rangle \langle w_j|, \tag{4}$$

where m is the number of words and $|w_j\rangle$ is word superposition state in Eq. 3, allowing multiple occurrences. Eq. 4 produces a density matrix ρ for semantic composition of words. It also describes a non-classical distribution over the set of sememes: the complex-valued off-diagonal elements describes the correlations between sememes, while the diagonal entries (guaranteed to be real by its original property) correspond to a standard probability distribution. The off-diagonal elements provide our framework some potentials to model the possible interactions between the basic sememe basis, which was usually considered mutually independent with each other.

Semantic Measurements. The high-level features of a sequence of words are extracted through measurements on its mixed state. Given a density matrix ρ of a mixed state, a rank-one projector P, which is the outer product of a unit complex vector, i.e. $P = |x\rangle \langle x|$, is applied as a measurement projector. It is worth mentioning that $|x\rangle$ could be any pure state in this Hilbert space (not only limited to a specific word w). The measured probability is computed by Gleason‘s Theorem in Eq. 2.

4 Complex-valued Network for Matching

We implemented an end-to-end network for matching on the Semantic Hilbert Space. Fig. 1 shows the overall structure of the proposed Complex-valued Network for Matching (CNM). Each component of the network is further discussed in this section.

4.1 Complex-valued Embedding

On the Semantic Hilbert Space, each word w is embedded as a complex-valued vector $\vec{w}$. Here we link its length and direction to different physical meanings: the length of a vector represents the relative weight of the word while the vector direction is viewed as a superposition state. Each word w adopts a normalization into a superposition state $|w\rangle$ and a word-dependent weight $\pi(w)$:

$$|w\rangle = \frac{\vec{w}}{||\vec{w}||}, \;\; \pi(w) = ||\vec{w}||, \tag{5}$$

where $||\vec{w}||$ denotes the 2-norm length of $\vec{w}$. $\pi(w)$ is used to compute the relative weight of a word in

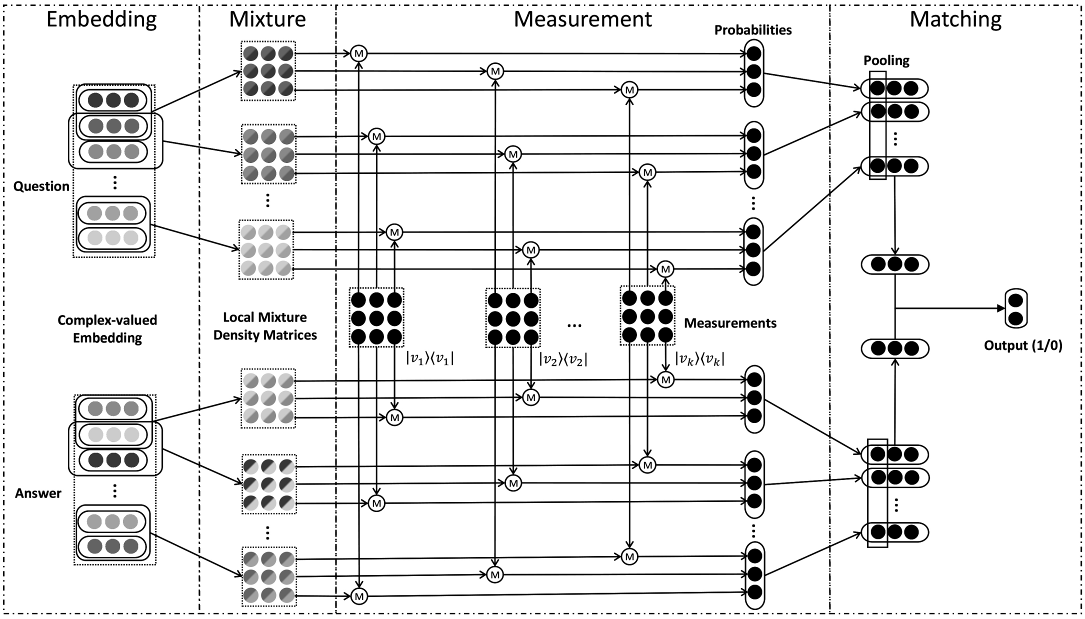


Figure 1: Architecture of Complex-valued Network for Matching. Ⓜ refers to the measurement operation in Eq. 2.

a local context window, which we will elaborate in Section 4.2.

4.2 Sentence Modeling with Local Mixture Scheme

A sentence is modeled as a combination of individual words in it. NNQLM (Zhang et al., 2018a) models a sentence as a *global mixture* of all words, which implicitly assumes a global interaction among all sentence words. This seems to be unreasonable in practice, especially for a long text segment such as a paragraph or a document, where the interaction between the first word and the last word is often negligible. Therefore, we address this limitation by proposing a *local mixture* of words, which tends to capture the semantic relations between neighboring words and undermine the long-range word dependencies. As is shown in Fig. 2, a sliding window is applied and a density matrix is constructed for a local window of length l (e.g. 3). Therefore, a sentence is composed of a sequence of density matrices for l-grams.

The representation of a local l-gram window is obtained by an improved approach over Eq. 4. In Eq. 4, each word is assigned with the same weight, which does not hold from an empirical point of view. In this study, we take the $L2$-norm of the word vector as the relative weight in a local context window for a specific word, which could be updated during training. To some extent, $L2$-norm is a measure of semantic richness of a word, i.e. the longer the vector the richer the meaning. The

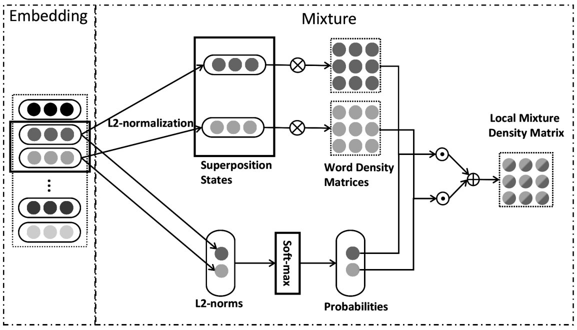


Figure 2: Architecture of local mixture component. A sliding window in the black color is applied to the sentence, generating a local mixture density matrix for each local window of length l. ⊙ means that a matrix multiplies a number with each element. ⊗ denotes the outer product of a vector with itself.

density matrix of an l-gram is computed as follows:

$$\rho = \sum_{i}^{l} p(w_i) \, |w_i\rangle \, \langle w_i|, \tag{6}$$

where the relative importance of each word $p(w_i)$ in an l-gram is the soft-max normalized word-dependent weight: $p(w_i) = \frac{e^{\pi(w_i)}}{\sum_j^l e^{\pi(w_j)}}$, where $\pi(w_i)$ is the word-dependent weight. By converting word-dependent weights to a probability distribution, a legal density matrix is produced, because $\sum_i^l p(w_i) = 1$ gives $tr(\rho) = 1$. Moreover, the weight of a word also depends on its neighboring words in a local context.

4.3 Matching of Question and Answer

In quantum information, there have been works trying to estimate a quantum state from the results of a series of measurements (Řeháček et al., 2001; Lvovsky, 2004). Inspired by these works, we introduce trainable measurements to extract density matrix features and match a pair of sentences.

Suppose a pair of sentences with length L are represented as two sets of density matrices $\{\rho_{1j}\}_{j=1}^{L}$ and $\{\rho_{2j}\}_{j=1}^{L}$ respectively. The same set of K semantic measurement operators $\{|v_k\rangle\}_{k=1}^{K}$ are applied to both sets, producing a pair of k-by-L probability matrix p^1 and p^2, where $p_{jk}^1 = \langle v_k| \rho_{1j} |v_k\rangle$ and $p_{jk}^2 = \langle v_k| \rho_{2j} |v_k\rangle$ for $k \in \{1, ..., K\}$ and $j \in \{1, ..., L\}$. A classical vector-based distances between p^1 and p^2 can be computed as the matching score of the sentence pair. By involving a set of semantic measurements, the properties of density matrix are taken into consideration in computing the density matrix distance.

We believe that this way of computing density matrix distance is both theoretically sound and applicable in practice. The trace inner product of density matrices (Zhang et al., 2018a) breaks the basic axioms of metric, namely the non-negativity, identity of indiscernibles and triangle inequality. The CNN-based feature extraction (Zhang et al., 2018a) for density matrix multiplication loses the property of density matrix as a probability distribution. Nielsen and Chuang (2010) introduced three measures namely trace distance, fidelity, and VN-divergence. However, it is computationally costly to compute these metrics and propagate the loss in an end-to-end training framework.

We set the measurements to be trainable so that the matching of question and answering can be integrated into the whole neural network, and identify the discriminative semantic measurements in a data-driven manner. From the perspective of linear discriminant analysis (LDA) (Fisher, 1936), this approach is intended to find a group of finite discriminative projection directions for a better division of different classes, but in a more sound framework inspired by quantum probability with complex-valued values. From an empirical point of view, the data-driven measurements make it flexible to match two sentences.

Dataset	train	dev	test
TREC QA	1229/53417	65/117	68/1442
WikiQA	873/8627	126/130	633/2351

Table 1: Dataset Statistics. For each cell, the values denote the number of questions and question-answer pairs respectively.

5 Experiments

5.1 Datasets and Evaluation Metrics

The experiments were conducted on two benchmarking question answering datasets for question answering (QA), namely TREC QA (Voorhees and Tice, 2000) and WikiQA (Yang et al., 2015). TREC QA is a standard QA dataset in the Text REtrieval Conference (TREC). WikiQA is released by Microsoft Research on open domain question answering. On both datasets, the task is to select the most appropriate answer from the candidate answers for a question, which requires a ranking of candidate answers. After removing the questions with no correct answers, the statistics of the cleaned datasets are given in the Tab. 1. Two common rank-based metrics, namely mean average precision (MAP) and mean reciprocal rank (MRR), are used to measure the performance of models.

5.2 Experiment Details

5.2.1 Baselines

We conduct a comprehensive comparison across a wide range of models. On TREC QA the experimented models include Bigram-CNN (Yu et al., 2014), three-layered Long Short-term Memory (LSTM) in combination with BM25 (LSTM-3L-BM25) (Wang and Nyberg, 2015), attention-based neural matching model (aNMM) (Yang et al., 2016), Multi-perspective CNN (MP-CNN) (He et al., 2015), CNTN (Qiu and Huang, 2015), attention-based LSTM+CNN model (LSTM-CNN-attn) (Tang et al., 2015) and pairwise word interaction modeling (PWIM) (He and Lin, 2016). On WikiQA dataset, we involve the following models into comparison: Bigram-CNN (Yu et al., 2014), CNN with word count information (CNN-Cnt) (Yang et al., 2015), QA-BILSTM (Santos et al., 2016), BILSTM with attentive pooling (AP-BILSTM) (Santos et al., 2016), and LSTM with attention (LSTM-attn) (Miao et al., 2015). On both datasets, we report the results of quantum language model (Sordoni et al., 2013) and two models NNQLM-I, NNQLM-II by (Zhang et al., 2018a) for comparison.

5.2.2 Parameter Settings

The parameters in the network are $\Theta = \{R, \Phi, \{|v_i\rangle\}_{i=1}^{k}\}$, in which R and Φ denote the lookup tables for amplitudes and complex phases of each word, and $\{|v_i\rangle\}_{i=1}^{k}$ denotes the set of semantic measurements. We use 50-dimension complex word embedding. The amplitudes are initialized with 50-dimension Glove vectors (Pennington et al., 2014) and L2-norm regularized during training. The phases are randomly initialized under a normal distribution of $[-\pi, \pi]$. The semantic measurements $\{|v_i\rangle\}_{i=1}^{k}\}$ are initialized with orthogonal real-valued one-hot vectors, and each measurement is constrained to be of unit length during training. We perform max pooling over the sentence dimension on the measurement probability matrices, resulting in a k-dim vector for both a question and an answer. We concatenate the vectors for $l = 1, 2, 3, 4$ for questions and answers, and the larger size of windows are also tried. We will use a longer sliding window in datasets with longer sentences. The cosine similarity is used as the distance metric of measured probabilities. We use triplet hinge loss and set the margin $\alpha = 0.1$. A dropout layer is built over the embedding layer and measurement probabilities with a dropout rate of 0.9.

A grid search is conducted over the parameter pools to explore the best parameters. The parameters under exploration include $\{0.01, 0.05, 0.1\}$ for the learning rate, $\{1e-5, 1e-6, 1e-7, 1e-8\}$ for the L2-normalization of complex word embeddings, $\{8, 16, 32\}$ for batch size, and $\{50, 100, 300, 500\}$ for the number of semantic measurements.

5.2.3 Parameter Scale

The proposed CNM has a limited scale of parameters. Apart from the complex word embeddings which are $|V| \times 2n$ by size, the only set of parameters are $\{|v_i\rangle\}_{i=1}^{k}$ which is $k \times 2n$, with $|V|, k, n$ being the vocabulary size, number of semantic measurements and the embedding dimension, respectively. In comparison, a single-layered CNN has at least $l \times k \times n$ additional parameters with l being the filter width, while a single-layered LSTM is $4 \times k \times (k + n)$ by the minimum parameter scale. Although we use both amplitude part and phase part for word embedding, lower dimensions of embedding are adopted, namely 50, with the comparable performance.

Model	MAP	MRR
Bigram-CNN	0.5476	0.6437
LSTM-3L-BM25	0.7134	0.7913
LSTM-CNN-attn	0.7279	0.8322
aNMM	0.7495	0.8109
MP-CNN	**0.7770**	0.8360
CNTN	0.7278	0.7831
PWIM	0.7588	0.8219
QLM	0.6780	0.7260
NNQLM-I	0.6791	0.7529
NNQLM-II	0.7589	0.8254
CNM	0.7701	**0.8591**
Over NNQLM-II	1.48% ↑	4.08% ↑

Table 2: Experiment Results on TREC QA Dataset. The best performed values are in bold.

Model	MAP	MRR
Bigram-CNN	0.6190	0.6281
QA-BILSTM	0.6557	0.6695
AP-BILSTM	0.6705	0.6842
LSTM-attn	0.6639	0.6828
CNN-Cnt	0.6520	0.6652
QLM	0.5120	0.5150
NNQLM-I	0.5462	0.5574
NNQLM-II	0.6496	0.6594
CNM	**0.6748**	**0.6864**
Over NNQLM-II	3.88% ↑	4.09% ↑

Table 3: Experiment Results on WikiQA Dataset.The best performed values for each dataset are in bold.

Therefore, our network scales better than the advanced models on the CNN or LSTM basis.

5.3 Experiment Results

Tab. 2 and 3 show the experiment results on TREC QA and WikiQA respectively, where bold values are the best performances out of all models. Our model achieves 3 best performances out of the 4 metrics on TREC QA and WikiQA, and performs slightly worse than the best-performed models on the remaining metric. This illustrates the effectiveness of our proposed model from a general perspective.

Specifically, CNM outperforms most CNN and LSTM-based models, which have more complicated structures and a relatively larger parameters scale. Also, CNM performs better than existing quantum-inspired QA models, QLM and NNQLM on both datasets, which means that the quantum theoretical framework gives rise to better performs model. Moreover, a significant improvement over NNQLM-1 is observed on these two datasets, supporting our claim that the trace inner product is not an effective distance metric of two density matrices.

Setting	MAP	MRR
FastText-MaxPool	0.6659 (0.1042↓)	0.7152 (0.1439↓)
CNM-Real	0.7112 (0.0589↓)	0.7922 (0.0659↓)
CNM-Global-Mixture	0.6968 (0.0733↓)	0.7829 (0.0762↓)
CNM-trace-inner-product	0.6952 (0.0749↓)	0.7688 (0.0903↓)
CNM	0.7701	0.8591

Table 4: Ablation Test. The values in parenthesis are the performance differences between the model and CNM.

5.4 Ablation Test

An ablation test is conducted to examine the influence of each component on our proposed CNM. The following models are implemented in the ablation test. *FastText-MaxPool* adopt max pooling over word-embedding, just like FastText (Joulin et al., 2016). *CNM-Real* replaces word embeddings and measurements with their real counterparts. *CNM-Global-Mixture* adopts a global mixture of the whole sentence, in which a sentence is represented as a single density matrix, leading to a probability vector for the measurement result. *CNM-trace-inner-product* replaces the trainable measurements with trace inner product like NNQLM.

For the real-valued models, we replace the embedding with double size of dimension, in order to eliminate the impact of the parameter scale on the performance. Due to limited space, we only report the ablation test result on TREC QA, and WikiQA has similar trends. The test results in Tab. 4 demonstrate that each component plays a crucial role in the CNM model. In particular, the comparison with CNM-Real and FastText-MaxPool shows the effectiveness of introducing complex-valued components, the increase in performance over CNM-Global-Mixture reveals the superiority of local mixture, and the comparison with CNM-trace-inner-product confirms the usefulness of trainable measurements.

6 Discussions

This section aims to investigate the proposed research questions mentioned in Sec 1. For RQ1, we explain the physical meaning of each component in term of the transparency (Sec. 6.1), and design some case studies for the post-hoc interpretability (Sec. 6.2). For RQ2, we argue that the complex-valued representation can model different aspects of semantics and naturally address the non-linear semantic compositionality, as discussed in Sec. 6.3.

Components	DNN	CNM
Sememe	-	basis one-hot vector / **basis state** $\{e \vert e \in \mathcal{R}^n, \vert\vert e\vert\vert_2 = 1\}$ complete &orthogonal
Word	real vector $(-\infty, \infty)$	unit complex vector / **superposition state** $\{w \vert w \in \mathcal{C}^n, \vert\vert w\vert\vert_2 = 1\}$
N-gram/ Word combinations	real vector $(-\infty, \infty)$	density matrix / **mixed system** $\{\rho \vert \rho = \rho^*, tr(\rho) = 1$
Abstraction	CNN/RNN $(-\infty, \infty)$	projector / **measurement** $\{vv^T \vert v \in \mathcal{C}^n, \vert\vert v\vert\vert_2 = 1\}$
Sentence representation	real vector $(-\infty, \infty)$	real value/ **measured probability** $(0, 1)$

Table 5: Physical meanings and constraints.

	Selected words
Important	studio, president, women, philosophy scandinavian, washingtonian, berliner, championship defiance, reporting, adjusted, jarred
Unimportant	71.2, 5.5, 4m, 296036, 3.5 may, be, all, born movements, economists, revenues, computers

Table 6: Selected learned important words in TREC QA. All words are converted to lower cases.

6.1 Transparency

CNM aims to unify many semantic units with different granularity e.g. sememes, words, phrases (or N-gram) and document in a single complex-valued vector space, as shown in Tab. 5. In particular, we formulate atomic sememes as a group of complete orthogonal basis states and words as superposition states over them. A linguistic unit with larger-granularity e.g. a word phrase or a sentence is represented as a mixed system over the words (with a density matrix, i.e. a positive semi-definite matrix with unit trace).

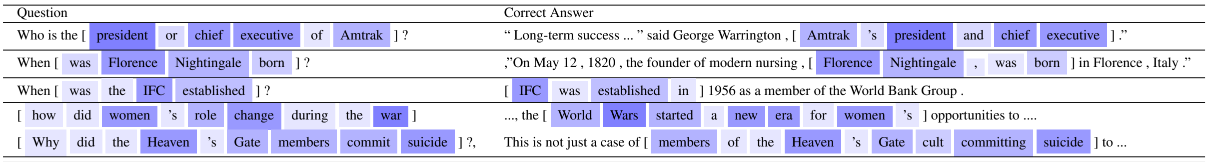
More importantly, trainable projection measurements are used to extract high-level representation for a word phrase or a sentence. Each measurement is also directly embedded in this unified Hilbert space, as a specific unit state (like words), thus making it easily understood by the neighbor words near this specific state. The corresponding trainable components in state-of-art neural network architectures, namely, kernels in CNN and cells in RNN, are represented as arbitrary real-valued without any constraints, lead to difficulty to be understood.

6.2 Post-hoc Interpretability

The post-hoc Interpretability is shown in three groups of case studies, namely word weight scheme, matching pattern, and discriminative semantic measurements.

6.2.1 Word Weighting Scheme

Tab. 6 shows the words selected from the top-50 most important words as well as top-50 unimpor-



Question	Correct Answer
Who is the [president or chief executive of Amtrak] ?	“ Long-term success ... ” said George Warrington , [Amtrak ’s president and chief executive] .”
When [was Florence Nightingale born] ?	,”On May 12 , 1820 , the founder of modern nursing , [Florence Nightingale , was born] in Florence , Italy .”
When [was the IFC established] ?	[IFC was established in] 1956 as a member of the World Bank Group .
[how did women ’s role change during the war]	..., the [World Wars started a new era for women ’s] opportunities to ....
[Why did the Heaven ’s Gate members commit suicide] ?,	This is not just a case of [members of the Heaven ’s Gate cult committing suicide] to ...

Table 7: The matching patterns for specific sentence pairs in TREC QA. The darker the color, the bigger the word weight is. [and] denotes the possible border of the current sliding windows.

tant ones. The importance of words is based on the L2-norm of its learned amplitude embedding according to Eq. 5. It is consistent with the intuition that, the important words are more about specific topics or discriminative nouns, while the unimportant words include meaningless numbers or super-high frequency words. Note that some special form (e.g. plural form in the last row) of words are also identified as unimportant words, since we commonly did not stem the words.

6.2.2 Matching Pattern

Tab. 7 shows the match schema with local sliding windows. In a local context window, we visualize the relative weights (i.e. the weights after normalized by *softmax*) for each word with darkness degrees. The table illustrates that our model is capable of identifying true matched local windows of a sentence pair. Even some words are replaced with similar forms (e.g. commit and committing in the last case) or meanings (e.g. change and new in the fourth case), it could be robust to get a relatively high matching score. From an empirical point of view, our model outperforms other models in situations where specific matching pattern are crucial to the sentence meaning, such as when two sentences share some unordered bag-of-word combinations. To some extent, it is robust up to replacement of words with similar ones in the Semantic Hilbert Space.

6.2.3 Discriminative Semantic Measurements

The semantic measurements are performed through rank-one projectors $\{|x\rangle \langle x|\}$. From a classical point of view, each projector is associated with a superposition of fundamental sememes, which is not necessarily linked to a particular word. Since the similarity metric in the Semantic Hilbert Space can be used to indicate semantic relatedness, we rely on the nearby words of the learned measurement projectors to understand what they may refer to.

Essentially, we identified the 10 most similar words to a measurement based on the cosine similarity metric. Tab. 8 shows part of the most similar words of 5 measurements, which are randomly chosen from the total number of k=10 trainable measurements for the TREC QA dataset. It can be seen that the first three selected measurements were about positions, movement verbs, and people’s names, while the rest were about the topic of history and rebellion respectively. Even though a clear explanation of the measurements is not available, we are still able to roughly understand the meaning of the measurements in the proposed data-driven approach.

	Selected neighborhood words for a measurement vector
1	andes, nagoya, inter-american, low-caste
2	cools, injection, boiling,adrift
3	andrews, paul, manson, bair
4	historically, 19th-century, genetic, hatchback
5	missile, exile, rebellion, darkness

Table 8: Selected learned measurements for TREC QA. They were selected according to nearest words for a measurement vector in Semantic Hilbert Space.

6.3 Complex-valued Representation

In CNM, each word is naturally embedded as a complex vector, composed of a complex phase part, a unit amplitude part, and a scalar-valued length. We argue that the amplitude part (i.e. the squared root of a probabilistic weight), corresponds to the classical word embedding with the lexical meaning, while the phase part implicitly reflects the higher-level semantic aspect e.g. polarity, ambiguity or emotion. The scalar-valued length is considered as the relative weight in a mixed system. The ablation study in Sec. 5.4 confirms that the complex-valued word embedding performs better than the real word embedding, which indicates that we benefit from the complex-valued embedding on the QA task.

From a mathematical point of view, complex-valued word embedding and other complex-valued components forms a new Hilbert vector space for modelling language, with a new definitions of addition and multiplication, as well as a new inner product operation. For instance, addition in the word meaning combination is defined as

$$\begin{aligned} z = & z_1 + z_2 = r_1 e^{i\theta_1} + r_2 e^{i\theta_2} \\ = & \sqrt{r_1^2 + r_2^2 + 2r_1 r_2 \cos(\theta_2 - \theta_1)} \\ & \times e^{i \arctan\left(\frac{r_1 \sin(\theta_1) + r_2 \sin(\theta_2)}{r_1 \cos(\theta_1) + r_2 \cos(\theta_2)}\right)} \end{aligned} \quad (7)$$

where z_1 and z_2 are the values for the corresponding element for two different word vectors $|w_1\rangle$ and $|w_2\rangle$ respectively. Both the amplitudes and complex phases of z are added with a nonlinear combination of phases and amplitudes of z_1 and z_2. A classical linear addition gives $\hat{z} = r_1 + r_2$, which can be viewed as a degenerating case of the complex-valued addition with the phase information being removed ($\theta_1 = \theta_2 = 0$ in the example).

7 Conclusions and Future Work

Towards the interpretable matching issue, we propose two research questions to investigate the possibility of language modelling with quantum mathematical framework. To this end, we design a new framework to model all the linguistic units in a unified Hilbert space with well-defined mathematical constraints and explicit physical meaning. We implement the above framework with neural network and then demonstrate its effectiveness in question answering (QA) task. Due to the well-designed components, our model is advantageous with its interpretability in term of transparency and post-hoc interpretability, and also shows its potential to use complex-valued components in NLP.

Despite the effectiveness of the current network, we would like to further explore the phase part in complex-valued word embedding to directly link to concrete semantics such as word sentiment or word position. Another possible direction is to borrow other quantum concepts to capture the interaction and non-interaction between word semantics, such as the *Fock Space* (Sozzo, 2014) which considers both interacting and non-interacting entities in different Hilbert Spaces. Furthermore, a deeper and robust quantum-inspired neural architecture in a higher-dimension Hilbert space like (Zhang et al., 2018b) is also worth to be investigated for achieving stronger performances with better explanatory power.

Acknowledgments

We thank Sagar Uprety, Dawei Song, and Prayag Tiwari for helpful discussions. Peng Zhang and Peter Bruza gave us constructive comments to improve the paper. The GPU computing resources are partly supported by Beijing Ultrapower Software Co., Ltd and Jianquan Li. Anne Elize R. V. Lima helped us redraw the figures using her talent.

The three authors are supported by the Quantum Access and Retrieval Theory (QUARTZ) project, which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 721321.

References

Diederik Aerts and Sandro Sozzo. 2014. Quantum Entanglement in Concept Combinations. *International Journal of Theoretical Physics*, 53(10):3587–3603.

Peter D Bruza, Kirsty Kitto, Douglas McEvoy, and Cathy McEvoy. 2008. Entangling words and meaning. pages QI, 118–124. College Publications.

Ronald A Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Andrew M Gleason. 1957. Measures on the closed subspaces of a hilbert space. *J. of Math. and Mech.*, pages 885–893.

Cliff Goddard and Anna Wierzbicka. 1994. *Semantic and lexical universals: Theory and empirical findings*. John Benjamins Publishing.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *EMNLP*, pages 1576–1586. ACL.

Hua He and Jimmy Lin. 2016. Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In *NAACL*, pages 937–948. ACL.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc.

Richard IG Hughes. 1992. *The structure and interpretation of quantum mechanics*. Harvard university press.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Zachary C. Lipton. 2018. The Mythos of Model Interpretability. *Queue*, 16(3):30:31–30:57.

A I Lvovsky. 2004. Iterative maximum-likelihood reconstruction in quantum homodyne tomography. *Journal of Optics B: Quantum and Semiclassical Optics*, 6(6):S556.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural Variational Inference for Text Processing. *arXiv:1511.06038*.

Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum computation and quantum information*. Cambridge University Press, Cambridge ; New York.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*, pages 1305–1311.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Alessandro Sordoni, Jian-Yun Nie, and Yoshua Bengio. 2013. Modeling term dependencies with quantum language models for ir. In *SIGIR*, pages 653–662. ACM.

Sandro Sozzo. 2014. A quantum probability explanation in Fock space for borderline contradictions. *Journal of Mathematical Psychology*, 58:1–12.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for Non-factoid Answer Selection. ArXiv: 1511.04108.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *EMNLP*, pages 1422–1432, Lisbon, Portugal.

Cornelis Joost Van Rijsbergen. 2004. *The geometry of information retrieval*. Cambridge University Press.

J. Řeháček, Z. Hradil, and M. Ježek. 2001. Iterative algorithm for reconstruction of entangled states. *Phys. Rev. A*, 63:040303.

Ellen M Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. *SIGIR*, pages 200–207.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *ACL*, pages 707–712.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In *CIKM*, pages 287–296. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP*, pages 2013–2018. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. ArXiv: 1412.1632.

Peng Zhang, Jiabin Niu, Zhan Su, Benyou Wang, Liqun Ma, and Dawei Song. 2018a. End-to-End Quantum-like Language Models with Application to Question Answering. *AAAI.*, pages 5666–5673.

Peng Zhang, Zhan Su, Lipeng Zhang, Benyou Wang, and Dawei Song. 2018b. A quantum many-body wave function inspired language modeling approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1303–1312. ACM.

COMMONSENSEQA: A Question Answering Challenge Targeting Commonsense Knowledge

Alon Talmor^{*,1,2} **Jonathan Herzig**^{*,1} **Nicholas Lourie**^{2} **Jonathan Berant**^{1,2}

^{1}School of Computer Science, Tel-Aviv University
^{2}Allen Institute for Artificial Intelligence
{alontalmor@mail,jonathan.herzig@cs,joberant@cs}.tau.ac.il,
nicholasl@allenai.org

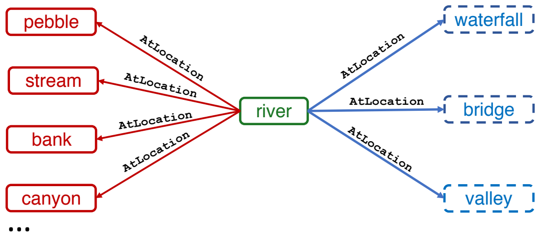
Abstract

When answering a question, people often draw upon their rich world knowledge in addition to the particular context. Recent work has focused primarily on answering questions given some relevant document or context, and required very little general background. To investigate question answering with prior knowledge, we present COMMONSENSEQA: a challenging new dataset for commonsense question answering. To capture common sense beyond associations, we extract from CONCEPTNET (Speer et al., 2017) multiple target concepts that have the same semantic relation to a single source concept. Crowd-workers are asked to author multiple-choice questions that mention the source concept and discriminate in turn between each of the target concepts. This encourages workers to create questions with complex semantics that often require prior knowledge. We create 12,247 questions through this procedure and demonstrate the difficulty of our task with a large number of strong baselines. Our best baseline is based on BERT-large (Devlin et al., 2018) and obtains 56% accuracy, well below human performance, which is 89%.

1 Introduction

When humans answer questions, they capitalize on their common sense and background knowledge about spatial relations, causes and effects, scientific facts and social conventions. For instance, given the question *"Where was Simon when he heard the lawn mower?"*, one can infer that the lawn mower is close to Simon, and that it is probably outdoors and situated at street level. This type of knowledge seems trivial for humans, but is still out of the reach of current natural language understanding (NLU) systems.

a) Sample ConceptNet for specific subgraphs



b) Crowd source corresponding natural language questions and two additional distractors

*Where on a **river** can you hold a cup upright to catch water on a sunny day?*
✓ waterfall, ✗ bridge, ✗ valley, ✗ *pebble*, ✗ **mountain**

*Where can I stand on a **river** to see water falling without getting wet?*
✗ waterfall, ✓ bridge, ✗ valley, ✗ *stream*, ✗ **bottom**

*I'm crossing the **river**, my feet are wet but my body is dry, where am I?*
✗ waterfall, ✗ bridge, ✓ valley, ✗ *bank*, ✗ **island**

Figure 1: (a) A source concept ('river') and three target concepts (dashed) are sampled from CONCEPTNET (b) Crowd-workers generate three questions, each having one of the target concepts for its answer (✓), while the other two targets are not (✗). Then, for each question, workers choose an additional distractor from CONCEPTNET (in italics), and author one themselves (in bold).

Work on Question Answering (QA) has mostly focused on answering factoid questions, where the answer can be found in a given context with little need for commonsense knowledge (Hermann et al., 2015; Rajpurkar et al., 2016; Nguyen et al., 2016; Joshi et al., 2017). Small benchmarks such as the Winograd Scheme Challenge (Levesque, 2011) and COPA (Roemmele et al., 2011), targeted common sense more directly, but have been difficult to collect at scale.

Recently, efforts have been invested in developing large-scale datasets for commonsense reasoning. In SWAG (Zellers et al., 2018b), given a textual description of an event, a probable subsequent event needs to be inferred. However, it

---

[*] The authors contributed equally

Proceedings of NAACL-HLT 2019, pages 4149–4158
Minneapolis, Minnesota, June 2 - June 7, 2019.

has been quickly realized that models trained on large amounts of unlabeled data (Devlin et al., 2018) capture well this type of information and performance on SWAG is already at human level. VCR (Zellers et al., 2018a) is another very recent attempt that focuses on the visual aspects of common sense. Such new attempts highlight the breadth of commonsense phenomena, and make it evident that research on common sense has only scratched the surface. Thus, there is need for datasets and models that will further our understanding of what is captured by current NLU models, and what are the main lacunae.

In this work, we present COMMONSENSEQA, a new dataset focusing on commonsense question answering, based on knowledge encoded in CONCEPTNET (Speer et al., 2017). We propose a method for generating commonsense questions at scale by asking crowd workers to author questions that describe the relation between concepts from CONCEPTNET (Figure 1). A crowd worker observes a source concept (*'River'* in Figure 1) and three target concepts (*'Waterfall'*, *'Bridge'*, *'Valley'*) that are all related by the same CONCEPTNET relation (`AtLocation`). The worker then authors three questions, one per target concept, such that only that particular target concept is the answer, while the other two distractor concepts are not. This primes the workers to add commonsense knowledge to the question, that separates the target concept from the distractors. Finally, for each question, the worker chooses one additional distractor from CONCEPTNET, and authors another distractor manually. Thus, in total, five candidate answers accompany each question.

Because questions are generated freely by workers, they often require background knowledge that is trivial to humans but is seldom explicitly reported on the web due to reporting bias (Gordon and Van Durme, 2013). Thus, questions in COMMONSENSEQA have a different nature compared to prior QA benchmarks, where questions are authored given an input text.

Using our method, we collected 12,247 commonsense questions. We present an analysis that illustrates the uniqueness of the gathered questions compared to prior work, and the types of commonsense skills that are required for tackling it. We extensively evaluate models on COMMONSENSEQA, experimenting with pre-trained models, fine-tuned models, and reading comprehension (RC) models that utilize web snippets extracted from Google search on top of the question itself. We find that fine-tuning BERT-LARGE (Devlin et al., 2018) on COMMONSENSEQA obtains the best performance, reaching an accuracy of 55.9%. This is substantially lower than human performance, which is 88.9%.

To summarize, our contributions are:

1. A new QA dataset centered around common sense, containing 12,247 examples.
2. A new method for generating commonsense questions at scale from CONCEPTNET.
3. An empirical evaluation of state-of-the-art NLU models on COMMONSENSEQA, showing that humans substantially outperform current models.

The dataset can be downloaded from `www.tau-nlp.org/commonsenseqa`. The code for all our baselines is available at `github.com/jonathanherzig/commonsenseqa`.

2 Related Work

Machine common sense, or the knowledge of and ability to reason about an open ended world, has long been acknowledged as a critical component for natural language understanding. Early work sought programs that could reason about an environment in natural language (McCarthy, 1959), or leverage a world-model for deeper language understanding (Winograd, 1972). Many commonsense representations and inference procedures have been explored (McCarthy and Hayes, 1969; Kowalski and Sergot, 1986) and large-scale commonsense knowledge-bases have been developed (Lenat, 1995; Speer et al., 2017). However, evaluating the degree of common sense possessed by a machine remains difficult.

One important benchmark, the Winograd Schema Challenge (Levesque, 2011), asks models to correctly solve paired instances of coreference resolution. While the Winograd Schema Challenge remains a tough dataset, the difficulty of generating examples has led to only a small available collection of 150 examples. The Choice of Plausible Alternatives (COPA) is a similarly important but small dataset consisting of 500 development and 500 test questions (Roemmele et al., 2011). Each question asks which of two alternatives best reflects a cause or effect relation to the premise. For both datasets, scalability is an issue when evaluating modern modeling approaches.

With the recent adoption of crowdsourcing, several larger datasets have emerged, focusing on predicting relations between situations or events in natural language. JHU Ordinal Commonsense Inference requests a label from 1-5 for the plausibility that one situation entails another (Zhang et al., 2017). The Story Cloze Test (also referred to as ROC Stories) pits ground-truth endings to stories against implausible false ones (Mostafazadeh et al., 2016). Interpolating these approaches, Situations with Adversarial Generations (SWAG), asks models to choose the correct description of what happens next after an initial event (Zellers et al., 2018b). LM-based techniques achieve very high performance on the Story Cloze Test and SWAG by fine-tuning a pre-trained LM on the target task (Radford et al., 2018; Devlin et al., 2018).

Investigations of commonsense datasets, and of natural language datasets more generally, have revealed the difficulty in creating benchmarks that measure the understanding of a program rather than its ability to take advantage of distributional biases, and to model the annotation process (Gururangan et al., 2018; Poliak et al., 2018). Annotation artifacts in the Story Cloze Test, for example, allow models to achieve high performance while only looking at the proposed endings and ignoring the stories (Schwartz et al., 2017; Cai et al., 2017). Thus, the development of benchmarks for common sense remains a difficult challenge.

Researchers have also investigated question answering that utilizes common sense. Science questions often require common sense, and have recently received attention (Clark et al., 2018; Mihaylov et al., 2018; Ostermann et al., 2018); however, they also need specialized scientific knowledge. In contrast to these efforts, our work studies common sense without requiring additional information. SQUABU created a small hand-curated test of common sense and science questions (Davis, 2016), which are difficult for current techniques to solve. In this work, we create similarly well-crafted questions but at a larger scale.

3 Dataset Generation

Our goal is to develop a method for generating questions that can be easily answered by humans without context, and require commonsense knowledge. We generate multiple-choice questions in a process that comprises the following steps:

1. We extract subgraphs from ConceptNet, each with one source concept and three target concepts.
2. We ask crowdsourcing workers to author three questions per subgraph (one per target concept), to add two additional distractors per question, and to verify questions' quality.
3. We add textual context to each question by querying a search engine and retrieving web snippets.

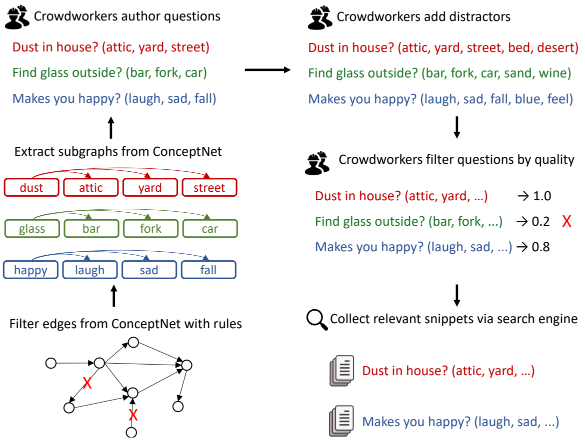


Figure 2: CommonsenseQA generation process. The input is ConceptNet knowledge base, and the output is a set of multiple-choice questions with corresponding relevant context (snippets).

The entire data generation process is summarized in Figure 2. We now elaborate on each of the steps:

Extraction from ConceptNet ConceptNet is a graph knowledge-base $G \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$, where the nodes $\mathcal{C}$ represent natural language concepts, and edges $\mathcal{R}$ represent commonsense relations. Triplets (c_1, r, c_2) carry commonsense knowledge such as '(*gambler*, `CapableOf`, *lose money*)'. ConceptNet contains 32 million triplets. To select a subset of triplets for crowdsourcing we take the following steps:

1. We filter triplets with general relations (e.g., `RelatedTo`) or relations that are already well-explored in NLP (e.g., `IsA`). In total we use 22 relations.
2. We filter triplets where one of the concepts is more than four words or not in English.
3. We filter triplets where the edit distance between c_1 and c_2 is too low.

This results in a set of 236,208 triplets (q, r, a), where we call the first concept the *question concept* and the second concept the *answer concept*.

We aim to generate questions that contain the question concept and where the answer is the answer concept. To create multiple-choice questions we need to choose *distractors* for each question. Sampling distractors at random from CONCEPTNET is a bad solution, as such distractors are easy to eliminate using simple surface clues.

To remedy this, we propose to create *question sets*: for each question concept q and relation r we group three different triplets $\{(q, r, a_1), (q, r, a_2), (q, r, a_3)\}$ (see Figure 1). This generates three answer concepts that are semantically similar and have a similar relation to the question concept q. This primes crowd workers to formulate questions that require background knowledge about the concepts in order to answer the question.

The above procedure generates approximately 130,000 triplets (43,000 question sets), for which we can potentially generate questions.

Crowdsourcing questions We used Amazon Mechanical Turk (AMT) workers to generate and validate commonsense questions.

AMT workers saw, for every question set, the question concept and three answer concepts. They were asked to formulate three questions, where all questions contain the question concept. Each question should have as an answer one of the answer concepts, but not the other two. To discourage workers from providing simple surface clues for the answer, they were instructed to avoid using words that have a strong relation to the answer concept, for example, not to use the word *'open'* when the answer is *'door'*.

Formulating questions for our task is non-trivial. Thus, we only accept annotators for which at least 75% of the questions they formulate pass the verification process described below.

Adding additional distractors To make the task more difficult, we ask crowd-workers to add two additional incorrect answers to each formulated question. One distractor is selected from a set of answer concepts with the same relation to the question concept in CONCEPTNET (Figure 1, in red). The second distractor is formulated manually by the workers themselves (Figure 1, in purple). Workers were encouraged to formulate a distractor that would seem plausible or related to the question but easy for humans to dismiss as incorrect. In total, each formulated question is accompanied with five candidate answers, including one correct answer and four distractors.

Measurement	Value
# CONCEPTNET distinct question nodes	2,254
# CONCEPTNET distinct answer nodes	12,094
# CONCEPTNET distinct nodes	12,107
# CONCEPTNET distinct relation lables	22
average question length (tokens)	13.41
long questions (more than 20 tokens)	10.3%
average answer length (tokens)	1.5
# answers with more than 1 token	44%
# of distinct words in questions	14,754
# of distinct words in answers	4,911

Table 1: Key statistics for COMMONSENSEQA

Verifying questions quality We train a disjoint group of workers to verify the generated questions. Verifiers annotate a question as unanswerable, or choose the right answer. Each question is verified by 2 workers, and only questions verified by at least one worker that answered correctly are used. This processes filters out 15% of the questions.

Adding textual context To examine whether web text is useful for answering commonsense questions, we add textual information to each question in the following way: We issue a web query to Google search for every question and candidate answer, concatenating the answer to the question, e.g., *'What does a parent tell their child to do after they've played with a lot of toys? + "clean room"'*. We take the first 100 result snippets for each of the five answer candidates, yielding a context of 500 snippets per question. Using this context, we can investigate the performance of reading comprehension (RC) models on COMMONSENSEQA.

Overall, we generated 12,247 final examples, from a total of 16,242 that were formulated. The total cost per question is $0.33. Table 1 describes the key statistics of COMMONSENSEQA.

4 Dataset Analysis

CONCEPTNET concepts and relations COMMONSENSEQA builds on CONCEPTNET, which contains *concepts* such as `dog`, `house`, or `row boat`, connected by *relations* such as `Causes`, `CapableOf`, or `Antonym`. The top-5 question concepts in COMMONSENSEQA are *'Person'* (3.1%), *'People'* (2.0%), *'Human'* (0.7%), *'Water'* (0.5%) and *'Cat'* (0.5%). In addition, we present the main relations along with the percentage of questions generated from them in Table 2. It's

Relation	**Formulated question example**	%
`AtLocation`	*Where would I not want a fox?* **A.** hen house, **B.** england, **C.** mountains, **D.** ...	47.3
`Causes`	*What is the hopeful result of going to see a play?* **A.** being entertained, **B.** meet, **C.** sit, **D.** ...	17.3
`CapableOf`	*Why would a person put flowers in a room with dirty gym socks?* **A.** smell good, **B.** many colors, **C.** continue to grow , **D.** ...	9.4
`Antonym`	*Someone who had a very bad flight might be given a trip in this to make up for it?* **A.** first class, **B.** reputable, **C.** propitious , **D.** ...	8.5
`HasSubevent`	*How does a person begin to attract another person for reproducing?* **A.** kiss, **B.** genetic mutation, **C.** have sex , **D.** ...	3.6
`HasPrerequisite`	*If I am tilting a drink toward my face, what should I do before the liquid spills over?* **A.** open mouth, **B.** eat first, **C.** use glass , **D.** ...	3.3
`CausesDesire`	*What do parents encourage kids to do when they experience boredom?* **A.** read book, **B.** sleep, **C.** travel , **D.** ...	2.1
`Desires`	*What do all humans want to experience in their own home?* **A.** feel comfortable, **B.** work hard, **C.** fall in love , **D.** ...	1.7
`PartOf`	*What would someone wear to protect themselves from a cannon?* **A.** body armor, **B.** tank, **C.** hat , **D.** ...	1.6
`HasProperty`	*What is a reason to pay your television bill?* **A.** legal, **B.** obsolete, **C.** entertaining , **D.** ...	1.2

Table 2: Top CONCEPTNET relations in COMMONSENSEQA, along with their frequency in the data and an example question. The first answer (**A**) is the correct answer

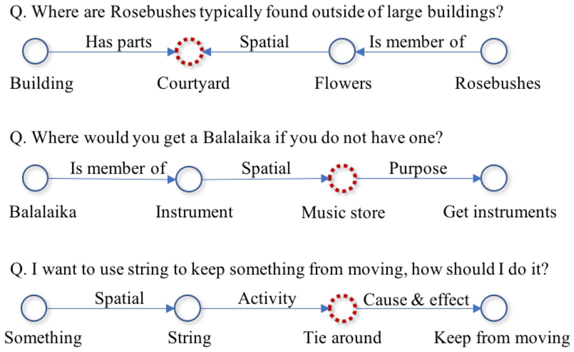


Figure 3: Examples of manually-annotated questions, with the required skills needed to arrive at the answers (red circles). Skills are labeled edges, and concepts are nodes.

Category	Definition	%
Spatial	Concept A appears near Concept B	41
Cause & Effect	Concept A causes Concept B	23
Has parts	Concept A contains Concept B as one of its parts	23
Is member of	Concept A belongs to the larger class of Concept B	17
Purpose	Concept A is the purpose of Concept B	18
Social	It is a social convention that Concept A correlates with Concept B	15
Activity	Concept A is an activity performed in the context of Concept B	8
Definition	Concept A is a definition of Concept B	6
Preconditions	Concept A must hold true in order for Concept B to take place	3

Table 3: Skills and their frequency in the sampled data. As each example can be annotated with multiple skills, the total frequency does not sum to 100%.

worth noting that since question formulators were not shown the CONCEPTNET relation, they often asked questions that probe other relationships between the concepts. For example, the question *"What do **audiences** clap for?"* was generated from the `AtLocation` relation, but focuses on social conventions instead.

Question formulation Question formulators were instructed to create questions with high language variation. 122 formulators contributed to question generation. However, 10 workers formulated more than 85% of the questions.

We analyzed the distribution of first and second words in the formulated questions along with example questions. Figure 4 presents the breakdown. Interestingly, only 44% of the first words are WH-words. In about 5% of the questions, formulators used first names to create a context story, and in 7% they used the word *"if"* to present a hypothetical question. This suggests high variability in the question language.

Commonsense Skills To analyze the types of commonsense knowledge needed to correctly answer questions in COMMONSENSEQA, we randomly sampled 100 examples from the development set and performed the following analysis.

For each question, we explicitly annotated the types of commonsense skills that a human uses to answer the question. We allow multiple commonsense skills per questions, with an average of 1.75 skills per question. Figure 3 provides three example annotations. Each annotation contains a node for the answer concept, and other nodes for concepts that appear in the question or latent concepts. Labeled edges describe the commonsense skill that relates the two nodes. We defined commonsense skills based on the analysis of LoBue and Yates (2011), with slight modifications to accommodate the phenomena in our data. Table 3 presents the skill categories we used, their definition and their frequency in the analyzed examples.

5 Baseline Models

Our goal is to collect a dataset of commonsense questions that are easy for humans, but hard for current NLU models. To evaluate this, we experiment with multiple baselines. Table 4 summarizes the various baseline types and characterizes them based on (a) whether training is done on COMMONSENSEQA or the model is fully pre-trained,

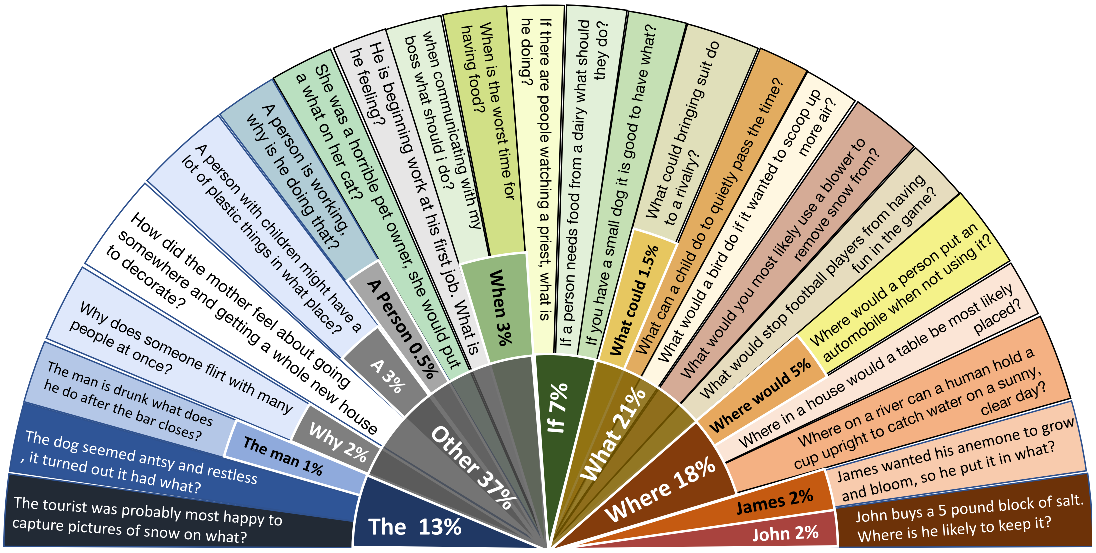


Figure 4: Distribution of the first and second words in questions. The inner part displays words and their frequency and the outer part provides example questions.

Model	Training	Context
VECSIM	✗	✗
LM1B	✗	✗
QABILINEAR	✓	✗
QACOMPARE	✓	✗
ESIM	✓	✗
GPT	✓	✗
BERT	✓	✗
BIDAF++	✓	✓

Table 4: Baseline models along with their characteristics. *Training* states whether the model was trained on COMMONSENSEQA, or was only trained a different dataset. *Context* states whether the model uses extra context as input.

and (b) whether context (web snippets) is used. We now elaborate on the different baselines.

a VECSIM A model that chooses the answer with highest cosine similarity to the question, where the question and answers are represented by an average of pre-trained word embeddings.

b LM1B Inspired by Trinh and Le (2018), we employ a large language model (LM) from Jozefowicz et al. (2016), which was pre-trained on the One Billion Words Benchmark (Chelba et al., 2013). We use this model in two variations. In the first (LM1B-CONCAT), we simply concatenate each answer to the question. In the second (LM1B-REP), we first cluster questions according to their first two words. Then, we recognize five high-frequency prefixes that cover 35% of the development set (e.g., *"what is"*). We rephrase questions that fit into one of these prefixes as a declarative sentence that contains the answer. E.g., we rephrase *"What is usually next to a door?"* and the candidate answer *"wall"* to *"Wall is usually next to a door"*. For questions that do not start with the above prefixes, we concatenate the answer as in LM1B-CONCAT. In both variations we return the answer with highest LM probability.

c QABILINEAR This model, propsed by Yu et al. (2014) for QA, scores an answer a_i with a bilinear model: $qWa_i^\top$, where the question q and answers a_i are the average pre-trained word embeddings and W is a learned parameter matrix. A softmax layer over the candidate answers is used to train the model with cross-entropy loss.

d QACOMPARE This model is similar to an NLI model from Liu et al. (2016). The model represents the interaction between the question q and a candidate answer a_i as: $h = \text{relu}([q; a_i; q \odot a_i; q - a_i]W_1 + b_1)$, where ';' denotes concatenation and $\odot$ is element-wise product. Then, the model predicts an answer score using a feed forward layer: $hW_2 + b_2$. Average pre-trained embeddings and softmax are used to train the model.

e ESIM We use ESIM, a strong NLI model (Chen et al., 2016). Similar to Zellers et al. (2018b), we change the output layer size to the number of candidate answers, and apply softmax to train with cross-entropy loss.

f BIDAF++ A state-of-the-art RC model, that uses the retrieved Google web snippets (Section 3) as context. We augment BIDAF (Seo et al., 2016) with a self-attention layer and ELMo representations (Peters et al., 2018; Huang et al., 2018). To adapt to the multiple-choice setting, we choose the answer with highest model probability.

g GENERATIVE PRE-TRAINED TRANSFORMER (GPT) Radford et al. (2018) proposed a method for adapting pre-trained LMs to perform a wide range of tasks. We applied their model to COMMONSENSEQA by encoding each question and its candidate answers as a series of delimiter-separated sequences. For example, the question *"If you needed a lamp to do your work, where would you put it?"*, and the candidate answer *"bedroom"* would become "`[start]` *If ... ?* `[sep]` *bedroom* `[end]`". The hidden representations over each `[end]` token are converted to logits by a linear transformation and passed through a softmax to produce final probabilities for the answers. We used the same pre-trained LM and hyper-parameters for fine-tuning as Radford et al. (2018) on ROC Stories, except with a batch size of 10.

h BERT Similarly to the GPT, BERT fine-tunes a language model and currently holds state-of-the-art across a broad range of tasks (Devlin et al., 2018). BERT uses a masked language modeling objective, which predicts missing words masked from unlabeled text. To apply BERT to COMMONSENSEQA, we linearize each question-answer pair into a delimiter-separated sequence (i.e., "`[CLS]` *If ... ?* `[SEP]` *bedroom* `[SEP]`") then fine-tune the pre-trained weights from uncased BERT-LARGE.[1] Similarly to the GPT, the hidden representations over each `[CLS]` token are run through a softmax layer to create the predictions. We used the same hyper-parameters as Devlin et al. (2018) for SWAG.

6 Experiments

Experimental Setup We split the data into a training/development/test set with an 80/10/10 split. We perform two types of splits: (a) *random split* – where questions are split uniformly at random, and (b) *question concept split* – where each of the three sets have disjoint question concepts. We empirically find (see below) that a random split is harder for models that learn from COMMONSENSEQA, because the same question concept appears in the training set and development/test set with different answer concepts, and networks that memorize might fail in such a scenario. Since the random split is harder, we consider it the primary split of COMMONSENSEQA.

We evaluate all models on the test set using accuracy (proportion of examples for which prediction is correct), and tune hyper-parameters for all trained models on the development set. To understand the difficulty of the task, we add a SANITY mode, where we replace the hard distractors (that share a relation with the question concept and one formulated by a worker) with random CONCEPTNET distractors. We expect a reasonable baseline to perform much better in this mode.

For pre-trained word embeddings we consider 300d GloVe embeddings (Pennington et al., 2014) and 300d Numberbatch CONCEPTNET node embeddings (Speer et al., 2017), which are kept fixed at training time. We also combine ESIM with 1024d ELMo contextual representations, which are also fixed during training.

Human Evaluation To test human accuracy, we created a separate task for which we did not use a qualification test, nor used AMT master workers. We sampled 100 random questions and for each question gathered answers from five workers that were not involved in question generation. Humans obtain 88.9% accuracy, taking a majority vote for each question.

Results Table 5 presents test set results for all models and setups.

The best baselines are BERT-LARGE and GPT with an accuracy of 55.9% and 45.5%, respectively, on the random split (63.6% and 55.5%, respectively, on the question concept split). This is well below human accuracy, demonstrating that the benchmark is much easier for humans. Nevertheless, this result is much higher than random (20%), showing the ability of language models to store large amounts of information related to commonsense knowledge.

The top part of Table 5 describes untrained models. We observe that performance is higher than random, but still quite low. The middle part describes models that were trained on COMMONSENSEQA, where BERT-LARGE obtains best performance, as mentioned above. ESIM models follow BERT-LARGE and GPT, and obtain much lower performance. We note that ELMo representations did not improve performance compared to GloVe embeddings, possibly because we were un-

---

[1] The original weights and code released by Google may be found here: https://github.com/google-research/bert

	Random split		Question concept split	
Model	Accuracy	SANITY	Accuracy	SANITY
VecSim+Numberbatch	29.1	54.0	30.3	54.9
LM1B-rep	26.1	39.6	26.0	39.1
LM1B-concat	25.3	37.4	25.3	35.2
VecSim+GloVe	22.3	26.8	20.8	27.1
BERT-large	**55.9**	**92.3**	**63.6**	**93.2**
GPT	45.5	87.2	55.5	88.9
ESIM+ELMo	34.1	76.9	37.9	77.8
ESIM+GloVe	32.8	79.1	40.4	78.2
QABilinear+GloVe	31.5	74.8	34.2	71.8
ESIM+Numberbatch	30.1	74.6	31.2	75.1
QABilinear+Numberbatch	28.8	73.3	32.0	71.6
QACompare+GloVe	25.7	69.2	34.1	71.3
QACompare+Numberbatch	20.4	60.6	25.2	66.8
BiDAF++	32.0	71.0	38.4	72.0
Human	**88.9**			

Table 5: Test set accuracy for all models.

Category	**Formulated question example**	**Correct answer**	**Distractor**	**Accuracy**	**%**
Surface clues	*If someone laughs after surprising them they have a good sense of what?* *How might a automobile get off a freeway?*	**humor** **exit ramp**	laughter driveway	77.7	35%
Negation / Antonym	*Where would you store a pillow case that is not in use?* *Where might the stapler be if I cannot find it?*	drawer desk drawer	**bedroom** **desktop**	42.8	7%
Factoid knowledge	*How many hours are in a day?* *What geographic area is a lizard likely to be?*	twenty four west texas	**week** **ball stopped**	38.4	13%
Bad granularity	*Where is a well used toy car likely to be found?* *Where may you be if you're buying pork chops at a corner shop?*	child's room iowa	**own home** **town**	35.4	31%
Conjunction	*What can you use to store a book while traveling?* *On a hot day what can you do to enjoy something cool and sweet?*	suitcase eat ice cream	**library of congress** **fresh cake**	23.8	23%

Table 6: BERT-large baseline analysis. For each category we provide two examples, the correct answer, one distractor, model accuracy and frequency in the dataset. The predicted answer is in bold.

able to improve performance by back-propagating into the representations themselves (as we do in BERT-large and GPT). The bottom part shows results for BiDAF++ that uses web snippets as context. We observe that using snippets does not lead to high performance, hinting that they do not carry a lot of useful information.

Performance on the random split is five points lower than the question concept split on average across all trained models. We hypothesize that this is because having questions in the development/test set that share a question concept with the training set, but have a different answer, creates difficulty for networks that memorize the relation between a question concept and an answer.

Lastly, all SANITY models that were trained on CommonsenseQA achieve very high performance (92% for BERT-large), showing that selecting difficult distractors is crucial.

Baseline analysis To understand the performance of BERT-large, we analyzed 100 examples from the development set (Table 6). We labeled examples with categories (possibly more than one per example) and then computed the average accuracy of the model for each category.

We found that the model does well (77.7% accuracy) on examples where surface clues hint to the correct answer. Examples that involve negation or understanding antonyms have lower accuracy (42.8%), similarly to examples that require factoid knowledge (38.4%). Accuracy is particularly low in questions where the correct answer has finer granularity compared to one of the distractors (35.4%), and in cases where the correct answer needs to meet a conjunction of conditions, and the distractor meets only one of them (23.8%).

Learning Curves To extrapolate how current models might perform with more data, we evaluated BERT-large on the development set, training with varying amounts of data. The resulting learning curves are plotted in figure 5. For each training set size, hyper-parameters were identical to section 5, except the number of epochs was varied to keep the number of mini-batches during training constant. To deal with learning instabilities, each data point is the best of 3 runs. We observe that the accuracy of BERT-large is expected to be roughly 75% assuming 100k examples, still sub-

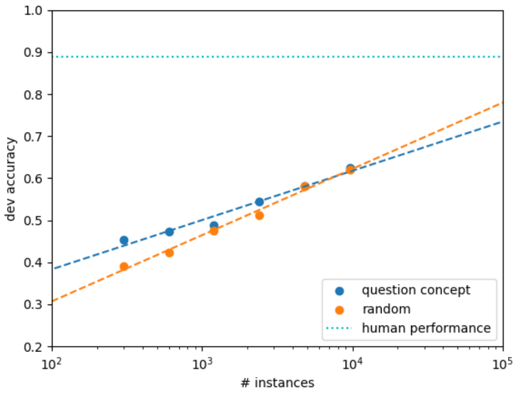


Figure 5: Development accuracy for BERT-LARGE trained with varying amounts of data.

stantially lower than human performance.

7 Conclusion

We present COMMONSENSEQA, a new QA dataset that contains 12,247 examples and aims to test commonsense knowledge. We describe a process for generating difficult questions at scale using CONCEPTNET, perform a detailed analysis of the dataset, which elucidates the unique properties of our dataset, and extensively evaluate on a strong suite of baselines. We find that the best model is a pre-trained LM tuned for our task and obtains 55.9% accuracy, dozens of points lower than human accuracy. We hope that this dataset facilitates future work in incorporating commonsense knowledge into NLU systems.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work was completed in partial fulfillment for the PhD degree of Jonathan Herzig, which was also supported by a Google PhD fellowship. This research was partially supported by The Israel Science Foundation grant 942/16, The Blavatnik Computer Science Research Fund and The Yandex Initiative for Machine Learning.

References

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In *ACL*.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Ernest Davis. 2016. How to write science questions that are easy for people and hard for computers. *AI magazine*, 37(1):13–22.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 25–30, New York, NY, USA. ACM.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. 2018. Flowqa: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683*.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

R Kowalski and M Sergot. 1986. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95.

Douglas B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38:32–38.

Hector J. Levesque. 2011. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 329–334. Association for Computational Linguistics.

J. McCarthy. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*.

John McCarthy and Patrick J. Hayes. 1969. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press. Reprinted in McC90.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering.

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *North American Association for Computational Linguistics (NAACL)*.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop on Cognitive Computing at NIPS*.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge. *CoRR*, abs/1803.05223.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proc. of *SEM*.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. 2018. Improving language understanding by generative pre-training. *Technical Report, OpenAI*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Roemmele, C. Bejan, and A. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. In *CoNLL*.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

O. Tange. 2011. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018a. From recognition to cognition: Visual commonsense reasoning. *arXiv preprint arXiv:1811.10830*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018b. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *TACL*, 5:379–395.

Probing the Need for Visual Context in Multimodal Machine Translation

Ozan Caglayan
LIUM, Le Mans University
ozan.caglayan@univ-lemans.fr

Pranava Madhyastha
Imperial College London
pranava@imperial.ac.uk

Lucia Specia
Imperial College London
l.specia@imperial.ac.uk

Loïc Barrault
LIUM, Le Mans University
loic.barrault@univ-lemans.fr

Abstract

Current work on multimodal machine translation (MMT) has suggested that the visual modality is either unnecessary or only marginally beneficial. We posit that this is a consequence of the very simple, short and repetitive sentences used in the only available dataset for the task (Multi30K), rendering the source text sufficient as context. In the general case, however, we believe that it is possible to combine visual and textual information in order to ground translations. In this paper we probe the contribution of the visual modality to state-of-the-art MMT models by conducting a systematic analysis where we partially deprive the models from source-side textual context. Our results show that under limited textual context, models are capable of leveraging the visual input to generate better translations. This contradicts the current belief that MMT models disregard the visual modality because of either the quality of the image features or the way they are integrated into the model.

1 Introduction

Multimodal Machine Translation (MMT) aims at designing better translation systems which take into account auxiliary inputs such as images. Initially organized as a shared task within the First Conference on Machine Translation (WMT16) (Specia et al., 2016), MMT has so far been studied using the Multi30K dataset (Elliott et al., 2016), a multilingual extension of Flickr30K (Young et al., 2014) with translations of the English image descriptions into German, French and Czech (Elliott et al., 2017; Barrault et al., 2018).

The three editions of the shared task have seen many exciting approaches that can be broadly categorized as follows: (i) multimodal attention using convolutional features (Caglayan et al., 2016; Calixto et al., 2016; Libovický and Helcl, 2017; Helcl et al., 2018) (ii) cross-modal interactions with spatially-unaware global features (Calixto and Liu, 2017; Ma et al., 2017; Caglayan et al., 2017a; Madhyastha et al., 2017) and (iii) the integration of regional features from object detection networks (Huang et al., 2016; Grönroos et al., 2018). Nevertheless, the conclusion about the contribution of the visual modality is still unclear: Grönroos et al. (2018) consider their multimodal gains "modest" and attribute the largest gain to the usage of external parallel corpora. Lala et al. (2018) observe that their multimodal word-sense disambiguation approach is not significantly different than the monomodal counterpart. The organizers of the latest edition of the shared task concluded that the multimodal integration schemes explored so far resulted in marginal changes in terms of automatic metrics and human evaluation (Barrault et al., 2018). In a similar vein, Elliott (2018) demonstrated that MMT models can translate without significant performance losses even in the presence of features from unrelated images.

These empirical findings seem to indicate that images are ignored by the models and hint at the fact that this is due to representation or modeling limitations. We conjecture that the most plausible reason for the linguistic dominance is that – at least in Multi30K – the source text is sufficient to perform the translation, eventually preventing the visual information from intervening in the learning process. To investigate this hypothesis, we introduce several input degradation regimes (Section 2) and revisit state-of-the-art MMT models (Section 3) to assess their behavior under degraded regimes. We further probe the visual sensitivity by deliberately feeding features from unrelated images. Our results (Section 4) show that MMT models successfully exploit the visual modality when the linguistic context is scarce, but indeed tend to be less sensitive to this modality when exposed to complete sentences.

Proceedings of NAACL-HLT 2019, pages 4159–4170
Minneapolis, Minnesota, June 2 - June 7, 2019.

2 Input Degradation

In this section we propose several degradations to the input language modality to simulate conditions where sentences may miss crucial information. We denote a set of translation pairs by $\mathcal{D}$ and indicate degraded variants with subscripts. Both the training and the test sets are degraded.

Color Deprivation. We consistently replace *source* words that refer to colors with a special token `[v]` ($\mathcal{D}_{\text{C}}$ in Table 1). Our hypothesis is that a monomodal system will have to rely on source-side contextual information and biases, while a multimodal architecture could potentially capitalize on color information extracted by exploiting the image and thus obtain better performance. This affects 3.3% and 3.1% of the words in the training and the test set, respectively.

Entity Masking. The Flickr30K dataset, from which Multi30K is derived, has also been extended with coreference chains to tag mentions of *visually depictable* entities in image descriptions (Plummer et al., 2015). We use these to mask out the head nouns in the *source* sentences ($\mathcal{D}_{\text{N}}$ in Table 1). This affects 26.2% of the words in both the training and the test set. We hypothesize that a multimodal system should heavily rely on the images to infer the missing parts.

Progressive Masking. A *progressively* degraded variant $\mathcal{D}_k$ replaces all but the first k tokens of *source* sentences with `[v]`. Unlike the color deprivation and entity masking, masking out suffixes does not guarantee systematic removal of visual context, but rather simulates an increasingly low-resource scenario. Overall, we form 16 degraded variants $\mathcal{D}_k$ (Table 1) where $k \in \{0, 2, \ldots, 30\}$. We stop at $\mathcal{D}_{30}$ since 99.8% of the sentences in Multi30K are shorter than 30 words with an average sentence length of 12 words. $\mathcal{D}_0$ – where the only remaining information is the source sentence length – is an interesting case from two perspectives: a neural machine translation (NMT) model trained on it resembles a target language model, while an MMT model becomes an image captioner with access to "expected length information".

Visual Sensitivity. Inspired by Elliott (2018), we experiment with *incongruent decoding* in order to understand how sensitive the multimodal systems are to the visual modality. This is achieved by explicitly violating the test-time semantic congruence across modalities. Specifically, we feed the visual features in *reverse* sample order to break image-sentence alignments. Consequently, a model capable of integrating the visual modality would likely deteriorate in terms of metrics.

$\mathcal{D}$	**a**	**lady**	**in**	**a**	**blue**	**dress**	**singing**
$\mathcal{D}_{\text{C}}$	a	lady	in	a	`[v]`	dress	singing
$\mathcal{D}_{\text{N}}$	a	`[v]`	in	a	blue	`[v]`	singing
$\mathcal{D}_4$	a	lady	in	a	`[v]`	`[v]`	`[v]`
$\mathcal{D}_2$	a	lady	`[v]`	`[v]`	`[v]`	`[v]`	`[v]`
$\mathcal{D}_0$	`[v]`	`[v]`	`[v]`	`[v]`	`[v]`	`[v]`	`[v]`

Table 1: An example of the proposed input degradation schemes: $\mathcal{D}$ is the original sentence.

3 Experimental Setup

Dataset. We conduct experiments on the English→French part of Multi30K. The models are trained on the concatenation of the *train* and *val* sets (30K sentences) whereas *test2016 (dev)* and *test2017 (test)* are used for early-stopping and model evaluation, respectively. For *entity masking*, we revert to the default Flickr30K splits and perform the model evaluation on *test2016*, since *test2017* is not annotated for entities. We use word-level vocabularies of 9,951 English and 11,216 French words. We use Moses (Koehn et al., 2007) scripts to lowercase, normalize and tokenize the sentences with hyphen splitting. The hyphens are stitched back prior to evaluation.

Visual Features. We use a ResNet-50 CNN (He et al., 2016) trained on ImageNet (Deng et al., 2009) as image encoder. Prior to feature extraction, we center and standardize the images using ImageNet statistics, resize the shortest edge to 256 pixels and take a center crop of size 256x256. We extract spatial features of size 2048x8x8 from the final convolutional layer and apply L_2 normalization along the depth dimension (Caglayan et al., 2018). For the non-attentive model, we use the 2048-dimensional global average pooled version (pool5) of the above convolutional features.

Models. Our baseline NMT is an attentive model (Bahdanau et al., 2014) with a 2-layer bidirectional GRU encoder (Cho et al., 2014) and a 2-layer conditional GRU decoder (Sennrich et al., 2017). The second layer of the decoder receives the output of the attention layer as input.

	$\mathcal{D}$	$\mathcal{D}_{\mathrm{C}}$
NMT	70.6 ± 0.5	68.4 ± 0.1
INIT	70.7 ± 0.2	$\mathbf{68.9} \pm 0.1$
HIER	70.9 ± 0.3	$\mathbf{69.0} \pm 0.3$
DIRECT	70.9 ± 0.2	$\mathbf{68.8} \pm 0.3$

Table 2: Baseline and color-deprivation METEOR scores: bold systems are significantly different from the NMT system within the same column (p-value ≤ 0.03).

For the MMT model, we explore the basic multimodal attention (DIRECT) (Caglayan et al., 2016) and its hierarchical (HIER) extension (Libovický and Helcl, 2017). The former linearly projects the concatenation of textual and visual context vectors to obtain the multimodal context vector, while the latter replaces the concatenation with another attention layer. Finally, we also experiment with encoder-decoder initialization (INIT) (Calixto and Liu, 2017; Caglayan et al., 2017a) where we initialize both the encoder and the decoder using a non-linear transformation of the pool5 features.

Hyperparameters. The encoder and decoder GRUs have 400 hidden units and are initialized with 0 except the multimodal INIT system. All embeddings are 200-dimensional and the decoder embeddings are tied (Press and Wolf, 2016). A dropout of 0.4 and 0.5 is applied on source embeddings and encoder/decoder outputs, respectively (Srivastava et al., 2014). The weights are decayed with a factor of $1e{-}5$. We use ADAM (Kingma and Ba, 2014) with a learning rate of $4e{-}4$ and mini-batches of 64 samples. The gradients are clipped if the total norm exceeds 1 (Pascanu et al., 2013). The training is early-stopped if *dev* set METEOR (Denkowski and Lavie, 2014) does not improve for ten epochs. All experiments are conducted with *nmtpytorch*[1] (Caglayan et al., 2017b).

4 Results

We train all systems three times each with different random initialization in order to perform significance testing with *multeval* (Clark et al., 2011). Throughout the section, we always report the mean over three runs (and the standard deviation) of the considered metrics. We decode the translations with a beam size of 12.

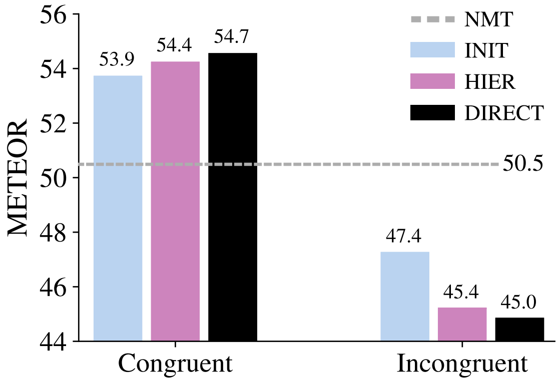


Figure 1: Entity masking: all masked MMT models are significantly better than the masked NMT (dashed). Incongruent decoding severely worsens all systems. The vanilla NMT baseline is 75.9[2].

We first present *test2017* METEOR scores for the baseline NMT and MMT systems, when trained on the full dataset $\mathcal{D}$ (Table 2). The first column indicates that, although MMT models perform slightly better on average, they are not significantly better than the baseline NMT. We now introduce and discuss the results obtained under the proposed degradation schemes. Please refer to Table 5 and the appendix for qualitative examples.

4.1 Color Deprivation

Unlike the inconclusive results for $\mathcal{D}$, we observe that all MMT models are significantly better than NMT when color deprivation is applied ($\mathcal{D}_{\mathrm{C}}$ in Table 2). If we further focus on the subset of the test set subjected to color deprivation (247 sentences), the gain increases to 1.6 METEOR for HIER. For the latter subset, we also computed the average color accuracy per sentence and found that the attentive models are 12% better than the NMT (32.5→44.5) whereas the INIT model only brings 4% (32.5→36.5) improvement. This shows that more complex MMT models are better at integrating visual information to perform better.

4.2 Entity Masking

The gains are much more prominent with entity masking, where the degradation occurs at a larger scale: Attentive MMT models show up to 4.2 METEOR improvement over NMT (Figure 1). We observed a large performance drop with *incongruent decoding*, suggesting that the visual modality is

---

[1] `github.com/lium-lst/nmtpytorch`

[2] Since entity masking uses Flickr30K splits (Section 3) rather than our splits, the scores are not comparable to those from other experiments in this paper.

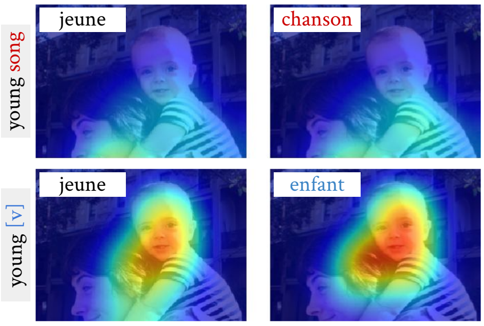


Figure 2: Baseline MMT (top) translates the misspelled "son" while the masked MMT (bottom) correctly produces "enfant" (child) by focusing on the image.

	+ Gain (↓ Incongruence Drop)		
	INIT	HIER	DIRECT
Czech	+1.4 (↓ 2.9)	+1.7 (↓ 3.5)	+1.7 (↓ 4.1)
German	+2.1 (↓ 4.7)	+2.5 (↓ 5.9)	+2.7 (↓ 6.5)
French	+3.4 (↓ 6.5)	+3.9 (↓ 9.0)	+4.2 (↓ 9.7)

Table 3: *Entity masking* results across three languages: all MMT models perform significantly better than their NMT counterparts (p-value ≤ 0.01). The incongruence drop applies on top of the MMT score.

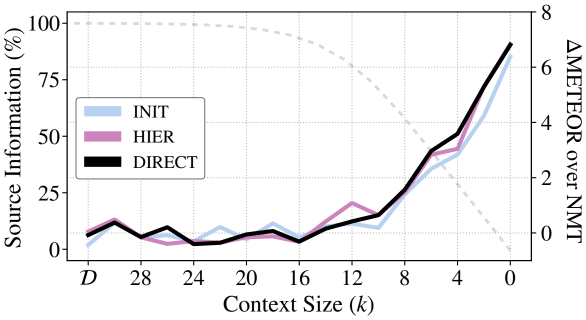


Figure 3: Multimodal gain in absolute METEOR for *progressive masking*: the dashed gray curve indicates the percentage of non-masked words in the training set.

	$\mathcal{D}_4$	$\mathcal{D}_6$	$\mathcal{D}_{12}$	$\mathcal{D}_{20}$	$\mathcal{D}$
DIRECT	**32.3**	**42.2**	**64.5**	**70.1**	**70.9**
Incongruent Dec.	↓ 6.4	↓ 5.5	↓ 1.4	↓ 0.7	↓ 0.7
Blinding	↓ 3.9	↓ 2.9	↓ 0.4	↓ 0.5	↓ 0.3
NMT	↓ 3.7	↓ 2.6	↓ 0.6	↓ 0.2	↓ 0.3

Table 4: The impact of incongruent decoding for *progressive masking*: all METEOR differences are against the DIRECT model. The blinded systems are both trained and decoded using incongruent features.

now much more important than previously demonstrated (Elliott, 2018). A comparison of attention maps produced by the baseline and masked MMT models reveals that the attention weights are more consistent in the latter. An interesting example is given in Figure 2 where the masked MMT model attends to the correct region of the image and successfully translates a dropped word that was otherwise a spelling mistake ("son"→"song").

Czech and German. In order to understand whether the above observations are also consistent across different languages, we extend the *entity masking* experiments to German and Czech parts of Multi30K. Table 3 shows the gain of each MMT system with respect to the NMT model and the subsequent drop caused by incongruent decoding[3]. First, we see that the multimodal benefits clearly hold for German and Czech, although the gains are lower than for French[4]. Second, when we compute the average drop from using incongruent images across all languages, we see how conservative the INIT system is (↓ 4.7) compared to HIER (↓ 6.1) and DIRECT (↓ 6.8). This raises a follow-up question as to whether the hidden state initialization eventually loses its impact throughout the recurrence where, as a consequence, the only modality processed is the text.

4.3 Progressive Masking

Finally, we discuss the results of the progressive masking experiments for French. Figure 3 clearly shows that as the sentences are progressively degraded, all MMT systems are able to leverage the visual modality. When the multimodal task becomes image captioning at $k{=}0$, MMT models improve over the language-model counterpart by $\sim$7 METEOR. Further qualitative examples show that the systems perform surprisingly well by producing visually plausible sentences (see Table 5 and the Appendix).

To get a sense of the visual sensitivity, we pick the DIRECT models trained on four degraded variants and perform *incongruent decoding*. We notice that as the amount of linguistic information increases, the gap narrows down: the MMT system gradually becomes less perplexed by the incongruence or, put in other words, less sensitive to the visual modality (Table 4).

---

[3] For example, the INIT system for French (Figure 1) surpasses the baseline (50.5) by reaching 53.9 (+3.4), which ends up at 47.4 (↓ 6.5) after incongruent decoding.

[4] This is probably due to the morphological richness of DE and CS which is suboptimally handled by word-level MT.

	`SRC`:	an older woman in `[v] [v] [v] [v] [v] [v] [v] [v] [v] [v] [v]`
	`NMT`:	une femme âgée avec un t-shirt blanc et des lunettes de soleil est assise sur un banc
		(an older woman with a white t-shirt and sunglasses is sitting on a bank)
	`MMT`:	une femme âgée en **maillot de bain rose** est assise sur un **rocher au bord de l'eau**
		(an older woman with a pink swimsuit is sitting on a rock at the seaside)
	`REF`:	une femme âgée **en bikini** bronze sur **un rocher au bord de l'océan**
		(an older woman in bikini is tanning on a rock at the edge of the ocean)
	`SRC`:	a young `[v]` in `[v]` holding a tennis `[v]`
	`NMT`:	un jeune garçon en bleu tenant une raquette de tennis
		(a young boy in blue holding a tennis racket)
	`MMT`:	**une** jeune **femme** en **blanc** tenant une raquette de tennis
	`REF`:	**une** jeune **femme** en **blanc** tenant une raquette de tennis
		(a young girl in white holding a tennis racket)
	`SRC`:	little girl covering her face with a `[v]` towel
	`NMT`:	une petite fille couvrant son visage avec une serviette blanche
		(a little girl covering her face with a white towel)
	`MMT`:	une petite fille couvrant son visage avec une serviette **bleue**
	`REF`:	une petite fille couvrant son visage avec une serviette **bleue**
		(a little girl covering her face with a blue towel)

Table 5: Qualitative examples from progressive masking, entity masking and color deprivation, respectively. Underlined and bold words highlight the bad and good lexical choices. MMT is an attentive system.

We then conduct a contrastive "blinding" experiment where the DIRECT models are not only fed with incongruent features at decoding time but also trained with them from scratch. The results suggest that the blinded models learn to ignore the visual modality. In fact, their performance is equivalent to NMT models.

5 Discussion and Conclusions

We presented an in-depth study on the potential contribution of images for multimodal machine translation. Specifically, we analysed the behavior of state-of-the-art MMT models under several degradation schemes in the Multi30K dataset, in order to reveal and understand the impact of textual predominance. Our results show that the models explored are able to integrate the visual modality if the available modalities are complementary rather than redundant. In the latter case, the primary modality (text) sufficient to accomplish the task. This dominance effect corroborates the seminal work of Colavita (1974) in Psychophysics where it has been demonstrated that visual stimuli dominate over the auditory stimuli when humans are asked to perform a simple audiovisual discrimination task. Our investigation using source degradation also suggests that visual grounding can increase the robustness of machine translation systems by mitigating input noise such as errors in the source text. In the future, we would like to devise models that can learn when and how to integrate multiple modalities by taking care of the complementary and redundant aspects of them in an intelligent way.

Acknowledgments

This work is a follow-up on the research efforts conducted within the "Grounded sequence-to-sequence transduction" team of the JSALT 2018 Workshop. We would like to thank Jindřich Libovický for contributing the hierarchical attention to *nmtpytorch* during the workshop. We also thank the reviewers for their valuable comments.

Ozan Caglayan and Loïc Barrault received funding from the French National Research Agency (ANR) through the CHIST-ERA M2CR project under the contract ANR-15-CHR2-0006-01. Lucia Specia and Pranava Madhyastha received funding from the MultiMT (H2020 ERC Starting Grant No. 678017) and MMVC (Newton Fund Institutional Links Grant, ID 352343575) projects.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computing Research Repository*, arXiv:1409.0473. Version 7.

Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. 2018. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 308–327, Belgium, Brussels. Association for Computational Linguistics.

Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. 2017a. LIUM-CVC submissions for WMT17 multimodal translation task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 432–439, Copenhagen, Denmark. Association for Computational Linguistics.

Ozan Caglayan, Adrien Bardet, Fethi Bougares, Loïc Barrault, Kai Wang, Marc Masana, Luis Herranz, and Joost van de Weijer. 2018. LIUM-CVC submissions for WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 603–608, Belgium, Brussels. Association for Computational Linguistics.

Ozan Caglayan, Loïc Barrault, and Fethi Bougares. 2016. Multimodal attention for neural machine translation. *Computing Research Repository*, arXiv:1609.03976.

Ozan Caglayan, Mercedes García-Martínez, Adrien Bardet, Walid Aransa, Fethi Bougares, and Loïc Barrault. 2017b. NMTPY: A flexible toolkit for advanced neural machine translation systems. *Prague Bull. Math. Linguistics*, 109:15–28.

Iacer Calixto, Desmond Elliott, and Stella Frank. 2016. DCU-UvA multimodal MT system report. In *Proceedings of the First Conference on Machine Translation*, pages 634–638, Berlin, Germany. Association for Computational Linguistics.

Iacer Calixto and Qun Liu. 2017. Incorporating global visual features into attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 992–1003, Copenhagen, Denmark. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Francis B. Colavita. 1974. Human sensory dominance. *Perception & Psychophysics*, 16 (2):409–412.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics.

Desmond Elliott. 2018. Adversarial evaluation of multimodal machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2978, Brussels, Belgium. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 215–233, Copenhagen, Denmark. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany. Association for Computational Linguistics.

Stig-Arne Grönroos, Benoit Huet, Mikko Kurimo, Jorma Laaksonen, Bernard Merialdo, Phu Pham, Mats Sjöberg, Umut Sulubacak, Jörg Tiedemann, Raphael Troncy, and Raúl Vázquez. 2018. The MeMAD submission to the WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation*, pages 609–617, Belgium, Brussels. Association for Computational Linguistics.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Jindřich Helcl, Jindřich Libovický, and Dusan Varis. 2018. CUNI system for the WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 622–629, Belgium, Brussels. Association for Computational Linguistics.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 639–645, Berlin, Germany. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository*, arXiv:1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chiraag Lala, Pranava Swaroop Madhyastha, Carolina Scarton, and Lucia Specia. 2018. Sheffield submissions for WMT18 multimodal translation shared task. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 630–637, Belgium, Brussels. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–202, Vancouver, Canada. Association for Computational Linguistics.

Mingbo Ma, Dapeng Li, Kai Zhao, and Liang Huang. 2017. OSU multimodal machine translation system report. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 465–469, Copenhagen, Denmark. Association for Computational Linguistics.

Pranava Swaroop Madhyastha, Josiah Wang, and Lucia Specia. 2017. Sheffield MultiMT: Using object posterior predictions for multimodal machine translation. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 470–476, Copenhagen, Denmark. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *Computing Research Repository*, arXiv:1608.05859.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68. Association for Computational Linguistics.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation*, pages 543–553, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

A Qualitative Examples

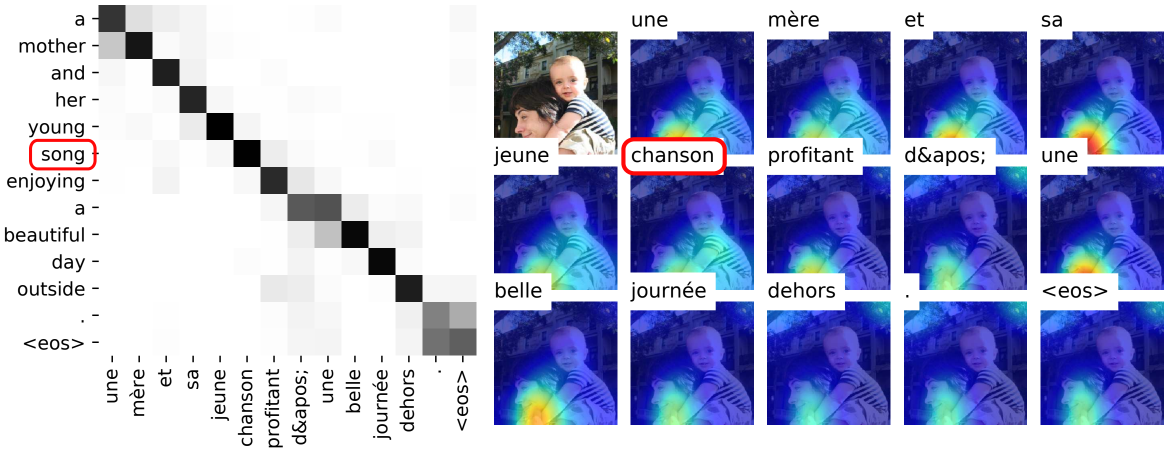
In this appendix, we provide further translation examples for color deprivation (Table 6), entity masking (Table 7) and progressive masking (Table 8). Specifically for the entity masking experiments, we also give further examples to showcase the behavior of the visual attention in Figure 4 and Figure 5.

	`SRC:` a girl in `[v]` is sitting on a bench `NMT:` pink `Init:` pink `Hier:` **black** `Direct:` **black**
	`SRC:` a man dressed in `[v]` talking to a girl `NMT:` black `Init:` black `Hier:` **white** `Direct:` **white**
	`SRC:` a `[v]` dog sits under a `[v]` umbrella `NMT:` brown / **blue** `Init:` **black / blue** `Hier:` **black / blue** `Direct:` **black / blue**
	`SRC:` a woman in a `[v]` top is dancing as a woman and boy in a `[v]` shirt watch `NMT:` blue / blue `Init:` blue / blue `Hier:` **red / red** `Direct:` **red / red**
	`SRC:` three female dancers in `[v]` dresses are performing a dance routine `NMT:` white `Init:` white `Hier:` white `Direct:` **blue**

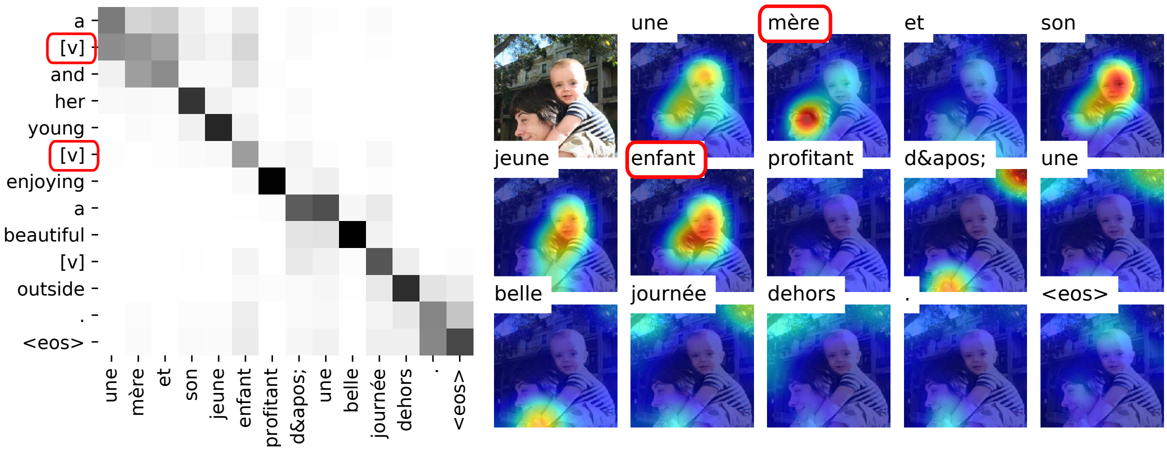
Table 6: Color deprivation examples from the English→French models: bold indicates correctly predicted cases. The colors generated by the models are shown in English for the sake of clarity.

	`SRC`: a `[v]` in a red `[v]` plays in the `[v]` `NMT`: un garçon en t-shirt rouge joue dans la neige *(a boy in a red t-shirt plays in the snow)* `MMT`: un garçon en **maillot de bain** rouge joue dans **l'eau** `REF`: un garçon en **maillot de bain** rouge joue dans **l'eau** *(a boy in a red swimsuit plays in the water)*
	`SRC`: a `[v]` drinks `[v]` outside on the `[v]` `NMT`: un homme boit du vin dehors sur le trottoir *(a man drinks wine outside on the sidewalk)* `MMT`: un **chien** boit de **l'eau** dehors sur **l'herbe** `REF`: un **chien** boit de **l'eau** dehors sur **l'herbe** *(a dog drinks water outside on the grass)*
	`SRC`: two `[v]` are driving on a `[v]` `NMT`: deux hommes font du vélo sur une route *(two men riding bicycles on a road)* `MMT`: deux **voitures roulent sur une piste** *(two cars driving on a track/circuit)* `REF`: deux **voitures roulent sur** un circuit
	`SRC`: a `[v]` turns on the `[v]` to pursue a flying `[v]` `NMT`: un homme tourne sur la plage pour attraper un frisbee volant *(a man turns on the beach to catch a flying frisbee)* `MMT`: un **chien** tourne sur **l'herbe** pour attraper un frisbee volant *(a dog turns on the grass to catch a flying frisbee)* `REF`: un **chien** tourne sur **l'herbe** pour poursuivre une balle en l'air *(a dog turns on the grass to chase a ball in the air)*
	`SRC`: a `[v]` jumping `[v]` on a `[v]` near a parking `[v]` `NMT`: un homme sautant à cheval sur une plage près d'un parking *(a man jumping on a beach near a parking lot)* `MMT`: une **fille** sautant à la **corde** sur un **trottoir** près d'un parking `REF`: une **fille** sautant à la **corde** sur un **trottoir** près d'un parking *(a girl jumping rope on a sidewalk near a parking lot)*

Table 7: Entity masking examples from the English→French models: underlined and bold words highlight bad and good lexical choices, respectively. English translations are provided in parentheses. MMT is an attentive model.

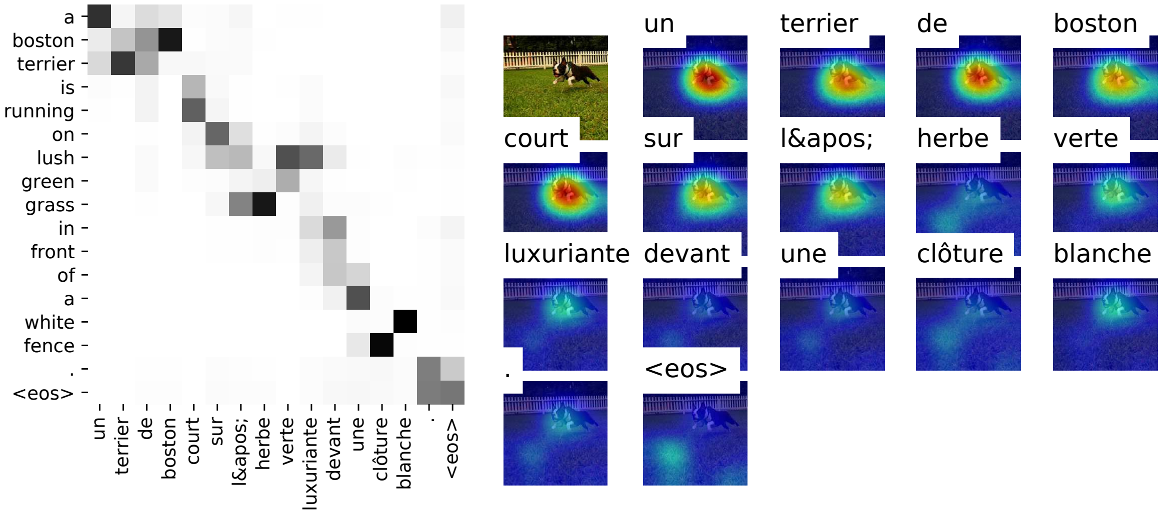


(a) Baseline (non-masked) MMT

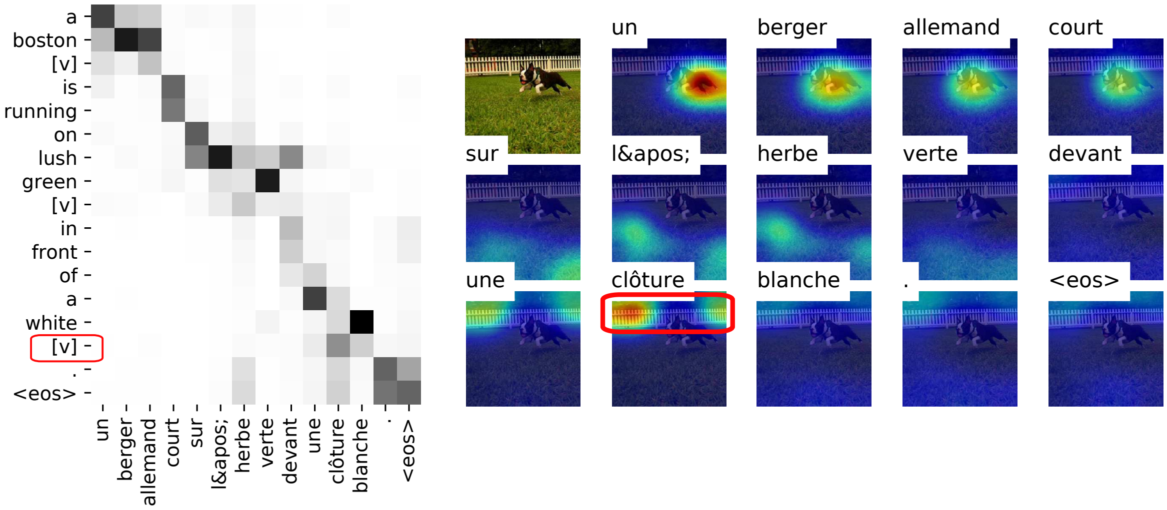


(b) Entity-masked MMT

Figure 4: Attention example from entity masking experiments: (a) Baseline MMT translates the misspelled "son" (song → chanson) while (b) the masked MMT achieves a correct translation (`[v]` → enfant) by exploiting the visual modality.



(a) Baseline (non-masked) MMT



(b) Entity-masked MMT

Figure 5: Attention example from entity masking experiments where *terrier, grass* and *fence* are dropped from the source sentence: (a) Baseline MMT is not able to shift attention from the salient *dog* to the *grass* and *fence*, (b) the attention produced by the masked MMT first shifts to the background area while translating “on lush green `[v]`” then focuses on the *fence*.

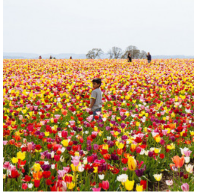
	`SRC`: a child `[v] [v] [v] [v] [v] [v]` `NMT`: un enfant avec des lunettes de soleil en train de jouer au tennis *(a child with sunglasses playing tennis)* `MMT`: un enfant **est debout dans un champ de fleurs** *(a child is standing in field of flowers)* `REF`: un enfant **dans un champ de tulipes** *(a child in a field of tulips)*
	`SRC`: a jockey riding his `[v] [v]` `NMT`: un jockey sur son vélo *(a jockey on his bike)* `MMT`: un jockey sur son **cheval** `REF`: un jockey sur son **cheval** *(a jockey on his horse)*
	`SRC`: girls are playing a `[v] [v] [v]` `NMT`: des filles jouent à un jeu de cartes *(girls are playing a card game)* `MMT`: des filles jouent un **match de football** `REF`: des filles jouent un **match de football** *(girls are playing a football match)*
	`SRC`: trees are in front `[v] [v] [v] [v] [v]` `NMT`: des vélos sont devant un bâtiment en plein air *(bicycles are in front of an outdoor building)* `MMT`: des **arbres** sont devant la **montagne** *(trees are in front of the mountain)* `REF`: des **arbres** sont devant une grande **montagne** *(trees are in front of a big mountain)*
	`SRC`: a fishing net on the deck of a `[v] [v]` `NMT`: un filet de pêche sur la terrasse d'un bâtiment *(a fishing net on the terrace of a building)* `MMT`: un filet de pêche sur le **pont d'un bateau** *(a fishing net on the deck of a boat)* `REF`: un filet de pêche sur le **pont d'un bateau** rouge *(a fishing net on the deck of a red boat)*
	`SRC`: girls wave purple flags `[v] [v] [v] [v] [v] [v] [v]` `NMT`: des filles en t-shirts violets sont assises sur des chaises dans une salle de classe *(girls in purple t-shirts are sitting on chairs in a classroom)* `MMT`: des filles en costumes violets **dansent dans une rue en ville** *(girls in purple costumes dance on a city street)* `REF`: des filles agitent des drapeaux violets tandis qu'elles défilent dans la rue *(girls wave purple flags as they parade down the street)*

Table 8: English→French progressive masking examples: underlined and bold words highlight bad and good lexical choices, respectively. English translations are provided in parentheses. MMT is an attentive model.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
`{jacobdevlin,mingweichang,kentonl,kristout}@google.com`

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pretrained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

Proceedings of NAACL-HLT 2019, pages 4171–4186
Minneapolis, Minnesota, June 2 - June 7, 2019.

word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer. In addition to the masked language model, we also use a "next sentence prediction" task that jointly pre-trains text-pair representations. The contributions of our paper are as follows:

- We demonstrate the importance of bidirectional pre-training for language representations. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pre-trained deep bidirectional representations. This is also in contrast to Peters et al. (2018a), which uses a shallow concatenation of independently trained left-to-right and right-to-left LMs.
- We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level *and* token-level tasks, outperforming many task-specific architectures.
- BERT advances the state of the art for eleven NLP tasks. The code and pre-trained models are available at `https://github.com/google-research/bert`.

2 Related Work

There is a long history of pre-training general language representations, and we briefly review the most widely-used approaches in this section.

2.1 Unsupervised Feature-based Approaches

Learning widely applicable representations of words has been an active area of research for decades, including non-neural (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) and neural (Mikolov et al., 2013; Pennington et al., 2014) methods. Pre-trained word embeddings are an integral part of modern NLP systems, offering significant improvements over embeddings learned from scratch (Turian et al., 2010). To pre-train word embedding vectors, left-to-right language modeling objectives have been used (Mnih and Hinton, 2009), as well as objectives to discriminate correct from incorrect words in left and right context (Mikolov et al., 2013).

These approaches have been generalized to coarser granularities, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014). To train sentence representations, prior work has used objectives to rank candidate next sentences (Jernite et al., 2017; Logeswaran and Lee, 2018), left-to-right generation of next sentence words given a representation of the previous sentence (Kiros et al., 2015), or denoising auto-encoder derived objectives (Hill et al., 2016).

ELMo and its predecessor (Peters et al., 2017, 2018a) generalize traditional word embedding research along a different dimension. They extract *context-sensitive* features from a left-to-right and a right-to-left language model. The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations. When integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP benchmarks (Peters et al., 2018a) including question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), and named entity recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) proposed learning contextual representations through a task to predict a single word from both left and right context using LSTMs. Similar to ELMo, their model is feature-based and not deeply bidirectional. Fedus et al. (2018) shows that the cloze task can be used to improve the robustness of text generation models.

2.2 Unsupervised Fine-tuning Approaches

As with the feature-based approaches, the first works in this direction only pre-trained word embedding parameters from unlabeled text (Collobert and Weston, 2008).

More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). The advantage of these approaches is that few parameters need to be learned from scratch. At least partly due to this advantage, OpenAI GPT (Radford et al., 2018) achieved previously state-of-the-art results on many sentence-level tasks from the GLUE benchmark (Wang et al., 2018a). Left-to-right language model-

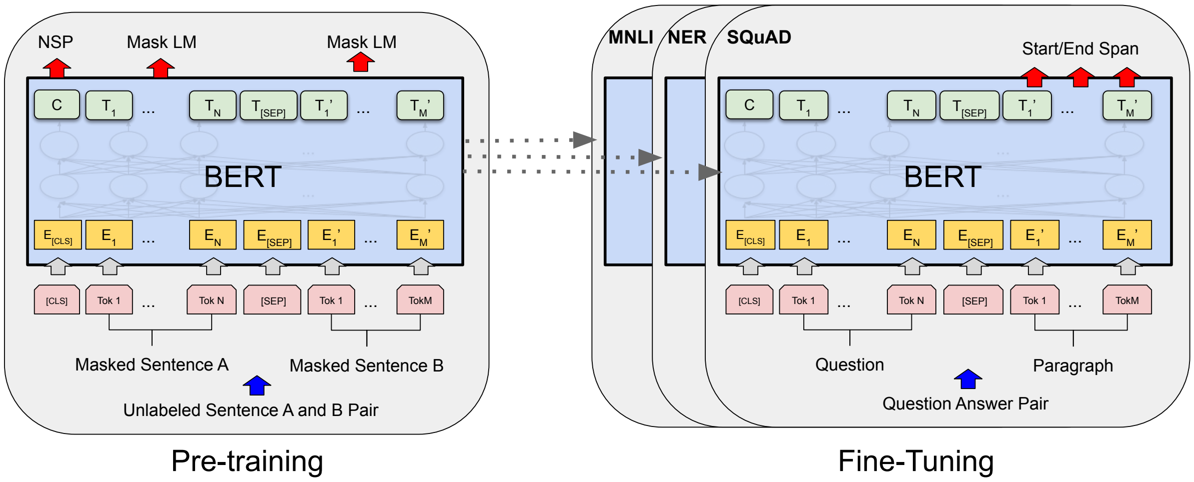


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

ing and auto-encoder objectives have been used for pre-training such models (Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015).

2.3 Transfer Learning from Supervised Data

There has also been work showing effective transfer from supervised tasks with large datasets, such as natural language inference (Conneau et al., 2017) and machine translation (McCann et al., 2017). Computer vision research has also demonstrated the importance of transfer learning from large pre-trained models, where an effective recipe is to fine-tune models pre-trained with ImageNet (Deng et al., 2009; Yosinski et al., 2014).

3 BERT

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: *pre-training* and *fine-tuning*. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section.

A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

Model Architecture BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017) and released in the `tensor2tensor` library.[1] Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani et al. (2017) as well as excellent guides such as "The Annotated Transformer."[2]

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.[3] We primarily report results on two model sizes: **BERT$_{\text{BASE}}$** (L=12, H=768, A=12, Total Parameters=110M) and **BERT$_{\text{LARGE}}$** (L=24, H=1024, A=16, Total Parameters=340M).

BERT$_{\text{BASE}}$ was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.[4]

---

[1] https://github.com/tensorflow/tensor2tensor

[2] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[3] In all cases we set the feed-forward/filter size to be $4H$, i.e., 3072 for the $H = 768$ and 4096 for the $H = 1024$.

[4] We note that in the literature the bidirectional Trans-

Input/Output Representations To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together.

We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. The first token of every sequence is always a special classification token (`[CLS]`). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token (`[SEP]`). Second, we add a learned embedding to every token indicating whether it belongs to sentence `A` or sentence `B`. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special `[CLS]` token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$.

For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. A visualization of this construction can be seen in Figure 2.

3.1 Pre-training BERT

Unlike Peters et al. (2018a) and Radford et al. (2018), we do not use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, we pre-train BERT using two unsupervised tasks, described in this section. This step is presented in the left part of Figure 1.

Task #1: Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right *or* right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context.

In order to train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a "masked LM" (MLM), although it is often referred to as a *Cloze* task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input.

Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the `[MASK]` token does not appear during fine-tuning. To mitigate this, we do not always replace "masked" words with the actual `[MASK]` token. The training data generator chooses 15% of the token positions at random for prediction. If the i-th token is chosen, we replace the i-th token with (1) the `[MASK]` token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time. Then, T_i will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix C.2.

Task #2: Next Sentence Prediction (NSP) Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the *relationship* between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train for a binarized *next sentence prediction* task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences `A` and `B` for each pre-training example, 50% of the time `B` is the actual next sentence that follows `A` (labeled as `IsNext`), and 50% of the time it is a random sentence from the corpus (labeled as `NotNext`). As we show in Figure 1, C is used for next sentence prediction (NSP).[5] Despite its simplicity, we demonstrate in Section 5.1 that pre-training towards this task is very beneficial to both QA and NLI. [6]

---

former is often referred to as a "Transformer encoder" while the left-context-only version is referred to as a "Transformer decoder" since it can be used for text generation.

[5] The final model achieves 97%-98% accuracy on NSP.

[6] The vector C is not a meaningful sentence representation without fine-tuning, since it was trained with NSP.

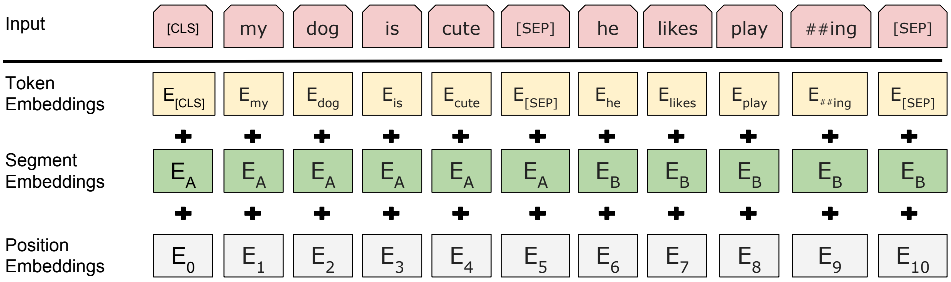


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

The NSP task is closely related to representation-learning objectives used in Jernite et al. (2017) and Logeswaran and Lee (2018). However, in prior work, only sentence embeddings are transferred to down-stream tasks, where BERT transfers all parameters to initialize end-task model parameters.

Pre-training data The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion Word Benchmark (Chelba et al., 2013) in order to extract long contiguous sequences.

3.2 Fine-tuning BERT

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks—whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs. For applications involving text pairs, a common pattern is to independently encode text pairs before applying bidirectional cross attention, such as Parikh et al. (2016); Seo et al. (2017). BERT instead uses the self-attention mechanism to unify these two stages, as encoding a concatenated text pair with self-attention effectively includes *bidirectional* cross attention between two sentences.

For each task, we simply plug in the task-specific inputs and outputs into BERT and fine-tune all the parameters end-to-end. At the input, sentence `A` and sentence `B` from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text-$\varnothing$ pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for token-level tasks, such as sequence tagging or question answering, and the `[CLS]` representation is fed into an output layer for classification, such as entailment or sentiment analysis.

Compared to pre-training, fine-tuning is relatively inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model.[7] We describe the task-specific details in the corresponding subsections of Section 4. More details can be found in Appendix A.5.

4 Experiments

In this section, we present BERT fine-tuning results on 11 NLP tasks.

4.1 GLUE

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018a) is a collection of diverse natural language understanding tasks. Detailed descriptions of GLUE datasets are included in Appendix B.1.

To fine-tune on GLUE, we represent the input sequence (for single sentence or sentence pairs) as described in Section 3, and use the final hidden vector $C \in \mathbb{R}^H$ corresponding to the first input token (`[CLS]`) as the aggregate representation. The only new parameters introduced during fine-tuning are classification layer weights $W \in \mathbb{R}^{K \times H}$, where K is the number of labels. We compute a standard classification loss with C and W, i.e., $\log(\mathrm{softmax}(CW^T))$.

---

[7] For example, the BERT SQuAD model can be trained in around 30 minutes on a single Cloud TPU to achieve a Dev F1 score of 91.0%.

[8] See (10) in `https://gluebenchmark.com/faq`.

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

We use a batch size of 32 and fine-tune for 3 epochs over the data for all GLUE tasks. For each task, we selected the best fine-tuning learning rate (among 5e-5, 4e-5, 3e-5, and 2e-5) on the Dev set. Additionally, for BERT$_{\text{LARGE}}$ we found that fine-tuning was sometimes unstable on small datasets, so we ran several random restarts and selected the best model on the Dev set. With random restarts, we use the same pre-trained checkpoint but perform different fine-tuning data shuffling and classifier layer initialization.[9]

Results are presented in Table 1. Both BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$ outperform all systems on all tasks by a substantial margin, obtaining 4.5% and 7.0% respective average accuracy improvement over the prior state of the art. Note that BERT$_{\text{BASE}}$ and OpenAI GPT are nearly identical in terms of model architecture apart from the attention masking. For the largest and most widely reported GLUE task, MNLI, BERT obtains a 4.6% absolute accuracy improvement. On the official GLUE leaderboard[10], BERT$_{\text{LARGE}}$ obtains a score of 80.5, compared to OpenAI GPT, which obtains 72.8 as of the date of writing.

We find that BERT$_{\text{LARGE}}$ significantly outperforms BERT$_{\text{BASE}}$ across all tasks, especially those with very little training data. The effect of model size is explored more thoroughly in Section 5.2.

4.2 SQuAD v1.1

The Stanford Question Answering Dataset (SQuAD v1.1) is a collection of 100k crowd-sourced question/answer pairs (Rajpurkar et al., 2016). Given a question and a passage from Wikipedia containing the answer, the task is to predict the answer text span in the passage.

As shown in Figure 1, in the question answering task, we represent the input question and passage as a single packed sequence, with the question using the `A` embedding and the passage using the `B` embedding. We only introduce a start vector $S \in \mathbb{R}^H$ and an end vector $E \in \mathbb{R}^H$ during fine-tuning. The probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph: $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$. The analogous formula is used for the end of the answer span. The score of a candidate span from position i to position j is defined as $S{\cdot}T_i + E{\cdot}T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction. The training objective is the sum of the log-likelihoods of the correct start and end positions. We fine-tune for 3 epochs with a learning rate of 5e-5 and a batch size of 32.

Table 2 shows top leaderboard entries as well as results from top published systems (Seo et al., 2017; Clark and Gardner, 2018; Peters et al., 2018a; Hu et al., 2018). The top results from the SQuAD leaderboard do not have up-to-date public system descriptions available,[11] and are allowed to use any public data when training their systems. We therefore use modest data augmentation in our system by first fine-tuning on TriviaQA (Joshi et al., 2017) befor fine-tuning on SQuAD.

Our best performing system outperforms the top leaderboard system by +1.5 F1 in ensembling and +1.3 F1 as a single system. In fact, our single BERT model outperforms the top ensemble system in terms of F1 score. Without TriviaQA fine-

---

[9] The GLUE data set distribution does not include the Test labels, and we only made a single GLUE evaluation server submission for each of BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$.

[10] https://gluebenchmark.com/leaderboard

[11] QANet is described in Yu et al. (2018), but the system has improved substantially after publication.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
BERT$_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

tuning data, we only lose 0.1-0.4 F1, still outperforming all existing systems by a wide margin.[12]

4.3 SQuAD v2.0

The SQuAD 2.0 task extends the SQuAD 1.1 problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic.

We use a simple approach to extend the SQuAD v1.1 BERT model for this task. We treat questions that do not have an answer as having an answer span with start and end at the `[CLS]` token. The probability space for the start and end answer span positions is extended to include the position of the `[CLS]` token. For prediction, we compare the score of the no-answer span: $s_{\mathtt{null}} = S{\cdot}C + E{\cdot}C$ to the score of the best non-null span $\hat{s_{i,j}}$ = $\mathtt{max}_{j \geq i} S{\cdot}T_i + E{\cdot}T_j$. We predict a non-null answer when $\hat{s_{i,j}} > s_{\mathtt{null}} + \tau$, where the threshold τ is selected on the dev set to maximize F1. We did not use TriviaQA data for this model. We fine-tuned for 2 epochs with a learning rate of 5e-5 and a batch size of 48.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT$_{\text{BASE}}$	81.6	-
BERT$_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

The results compared to prior leaderboard entries and top published work (Sun et al., 2018; Wang et al., 2018b) are shown in Table 3, excluding systems that use BERT as one of their components. We observe a +5.1 F1 improvement over the previous best system.

4.4 SWAG

The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair completion examples that evaluate grounded commonsense inference (Zellers et al., 2018). Given a sentence, the task is to choose the most plausible continuation among four choices.

When fine-tuning on the SWAG dataset, we construct four input sequences, each containing the concatenation of the given sentence (sentence `A`) and a possible continuation (sentence `B`). The only task-specific parameters introduced is a vector whose dot product with the `[CLS]` token representation C denotes a score for each choice which is normalized with a softmax layer.

We fine-tune the model for 3 epochs with a learning rate of 2e-5 and a batch size of 16. Results are presented in Table 4. BERT$_{\text{LARGE}}$ outperforms the authors' baseline ESIM+ELMo system by +27.1% and OpenAI GPT by 8.3%.

5 Ablation Studies

In this section, we perform ablation experiments over a number of facets of BERT in order to better understand their relative importance. Additional

---

[12] The TriviaQA data we used consists of paragraphs from TriviaQA-Wiki formed of the first 400 tokens in documents, that contain at least one of the provided possible answers.

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

ablation studies can be found in Appendix C.

5.1 Effect of Pre-training Tasks

We demonstrate the importance of the deep bidirectionality of BERT by evaluating two pre-training objectives using exactly the same pre-training data, fine-tuning scheme, and hyperparameters as $\text{BERT}_{\text{BASE}}$:

No NSP: A bidirectional model which is trained using the "masked LM" (MLM) but without the "next sentence prediction" (NSP) task.

LTR & No NSP: A left-context-only model which is trained using a standard Left-to-Right (LTR) LM, rather than an MLM. The left-only constraint was also applied at fine-tuning, because removing it introduced a pre-train/fine-tune mismatch that degraded downstream performance. Additionally, this model was pre-trained without the NSP task. This is directly comparable to OpenAI GPT, but using our larger training dataset, our input representation, and our fine-tuning scheme.

We first examine the impact brought by the NSP task. In Table 5, we show that removing NSP hurts performance significantly on QNLI, MNLI, and SQuAD 1.1. Next, we evaluate the impact of training bidirectional representations by comparing "No NSP" to "LTR & No NSP". The LTR model performs worse than the MLM model on all tasks, with large drops on MRPC and SQuAD.

For SQuAD it is intuitively clear that a LTR model will perform poorly at token predictions, since the token-level hidden states have no right-side context. In order to make a good faith attempt at strengthening the LTR system, we added a randomly initialized BiLSTM on top. This does significantly improve results on SQuAD, but the results are still far worse than those of the pre-trained bidirectional models. The BiLSTM hurts performance on the GLUE tasks.

We recognize that it would also be possible to train separate LTR and RTL models and represent each token as the concatenation of the two models, as ELMo does. However: (a) this is twice as expensive as a single bidirectional model; (b) this is non-intuitive for tasks like QA, since the RTL model would not be able to condition the answer on the question; (c) this it is strictly less powerful than a deep bidirectional model, since it can use both left and right context at every layer.

5.2 Effect of Model Size

In this section, we explore the effect of model size on fine-tuning task accuracy. We trained a number of BERT models with a differing number of layers, hidden units, and attention heads, while otherwise using the same hyperparameters and training procedure as described previously.

Results on selected GLUE tasks are shown in Table 6. In this table, we report the average Dev Set accuracy from 5 random restarts of fine-tuning. We can see that larger models lead to a strict accuracy improvement across all four datasets, even for MRPC which only has 3,600 labeled training examples, and is substantially different from the pre-training tasks. It is also perhaps surprising that we are able to achieve such significant improvements on top of models which are already quite large relative to the existing literature. For example, the largest Transformer explored in Vaswani et al. (2017) is (L=6, H=1024, A=16) with 100M parameters for the encoder, and the largest Transformer we have found in the literature is (L=64, H=512, A=2) with 235M parameters (Al-Rfou et al., 2018). By contrast, $\text{BERT}_{\text{BASE}}$ contains 110M parameters and $\text{BERT}_{\text{LARGE}}$ contains 340M parameters.

It has long been known that increasing the model size will lead to continual improvements on large-scale tasks such as machine translation and language modeling, which is demonstrated by the LM perplexity of held-out training data shown in Table 6. However, we believe that this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented mixed results on the downstream task impact of increasing the pre-trained bi-LM size from two to four layers and Melamud et al. (2016) mentioned in passing that increasing hidden dimension size from 200 to 600 helped, but increasing further to 1,000 did not bring further improvements. Both of these prior works used a feature-based approach — we hypothesize that when the model is fine-tuned directly on the downstream tasks and uses only a very small number of randomly initialized additional parameters, the task-specific models can benefit from the larger, more expressive pre-trained representations even when downstream task data is very small.

5.3 Feature-based Approach with BERT

All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the feature-based approach, where fixed features are extracted from the pre-trained model, has certain advantages. First, not all tasks can be easily represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added. Second, there are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation.

In this section, we compare the two approaches by applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task (Tjong Kim Sang and De Meulder, 2003). In the input to BERT, we use a case-preserving WordPiece model, and we include the maximal document context provided by the data. Following standard practice, we formulate this as a tagging task but do not use a CRF layer in the output. We use the representation of the first sub-token as the input to the token-level classifier over the NER label set.

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
BERT$_{\text{LARGE}}$	96.6	92.8
BERT$_{\text{BASE}}$	96.4	92.4
Feature-based approach (BERT$_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

To ablate the fine-tuning approach, we apply the feature-based approach by extracting the activations from one or more layers *without* fine-tuning any parameters of BERT. These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer.

Results are presented in Table 7. BERT$_{\text{LARGE}}$ performs competitively with state-of-the-art methods. The best performing method concatenates the token representations from the top four hidden layers of the pre-trained Transformer, which is only 0.3 F1 behind fine-tuning the entire model. This demonstrates that BERT is effective for both fine-tuning and feature-based approaches.

6 Conclusion

Recent empirical improvements due to transfer learning with language models have demonstrated that rich, unsupervised pre-training is an integral part of many language understanding systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep *bidirectional* architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*. Association for Computational Linguistics.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR*, abs/1705.00557.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *NAACL*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*.

Wilson L Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Wei Wang, Ming Yan, and Chen Wu. 2018b. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Appendix for "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

We organize the appendix into three sections:

- Additional implementation details for BERT are presented in Appendix A;
- Additional details for our experiments are presented in Appendix B; and
- Additional ablation studies are presented in Appendix C.

  We present additional ablation studies for BERT including:
  - Effect of Number of Training Steps; and
  - Ablation for Different Masking Procedures.

A Additional Details for BERT

A.1 Illustration of the Pre-training Tasks

We provide examples of the pre-training tasks in the following.

Masked LM and the Masking Procedure Assuming the unlabeled sentence is `my dog is hairy`, and during the random masking procedure we chose the 4-th token (which corresponding to `hairy`), our masking procedure can be further illustrated by

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict or which have been replaced by random words, so it is forced to keep a distributional contextual representation of *every* input token. Additionally, because random replacement only occurs for 1.5% of all tokens (i.e., 10% of 15%), this does not seem to harm the model's language understanding capability. In Section C.2, we evaluate the impact this procedure.

Compared to standard langauge model training, the masked LM only make predictions on 15% of tokens in each batch, which suggests that more pre-training steps may be required for the model

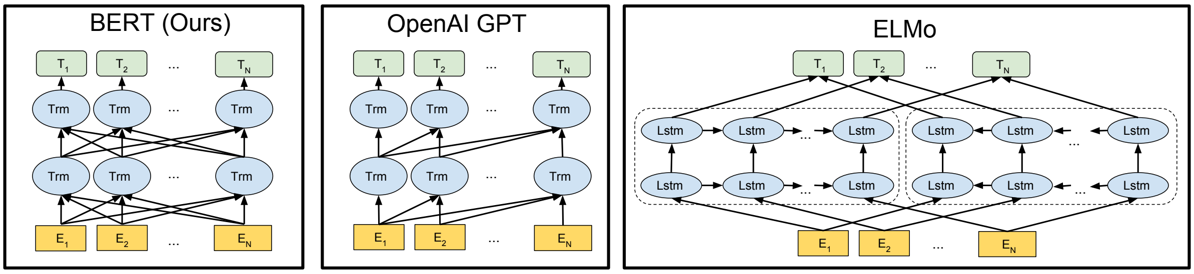


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

to converge. In Section C.1 we demonstrate that MLM does converge marginally slower than a left-to-right model (which predicts every token), but the empirical improvements of the MLM model far outweigh the increased training cost.

Next Sentence Prediction The next sentence prediction task can be illustrated in the following examples.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

A.2 Pre-training Procedure

To generate each training input sequence, we sample two spans of text from the corpus, which we refer to as "sentences" even though they are typically much longer than single sentences (but can be shorter also). The first sentence receives the `A` embedding and the second receives the `B` embedding. 50% of the time `B` is the actual next sentence that follows `A` and 50% of the time it is a random sentence, which is done for the "next sentence prediction" task. They are sampled such that the combined length is $\leq$ 512 tokens. The LM masking is applied after WordPiece tokenization with a uniform masking rate of 15%, and no special consideration given to partial word pieces.

We train with batch size of 256 sequences (256 sequences * 512 tokens = 128,000 tokens/batch) for 1,000,000 steps, which is approximately 40 epochs over the 3.3 billion word corpus. We use Adam with learning rate of 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps, and linear decay of the learning rate. We use a dropout probability of 0.1 on all layers. We use a `gelu` activation (Hendrycks and Gimpel, 2016) rather than the standard `relu`, following OpenAI GPT. The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood.

Training of $\mathrm{BERT}_{\mathrm{BASE}}$ was performed on 4 Cloud TPUs in Pod configuration (16 TPU chips total).[13] Training of $\mathrm{BERT}_{\mathrm{LARGE}}$ was performed on 16 Cloud TPUs (64 TPU chips total). Each pretraining took 4 days to complete.

Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. To speed up pretraing in our experiments, we pre-train the model with sequence length of 128 for 90% of the steps. Then, we train the rest 10% of the steps of sequence of 512 to learn the positional embeddings.

A.3 Fine-tuning Procedure

For fine-tuning, most model hyperparameters are the same as in pre-training, with the exception of the batch size, learning rate, and number of training epochs. The dropout probability was always kept at 0.1. The optimal hyperparameter values are task-specific, but we found the following range of possible values to work well across all tasks:

- **Batch size**: 16, 32

---

[13] https://cloudplatform.googleblog.com/2018/06/Cloud-TPU-now-offers-preemptible-pricing-and-global-availability.html

- **Learning rate (Adam)**: 5e-5, 3e-5, 2e-5
- **Number of epochs**: 2, 3, 4

We also observed that large data sets (e.g., 100k+ labeled training examples) were far less sensitive to hyperparameter choice than small data sets. Fine-tuning is typically very fast, so it is reasonable to simply run an exhaustive search over the above parameters and choose the model that performs best on the development set.

A.4 Comparison of BERT, ELMo ,and OpenAI GPT

Here we studies the differences in recent popular representation learning models including ELMo, OpenAI GPT and BERT. The comparisons between the model architectures are shown visually in Figure 3. Note that in addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

The most comparable existing pre-training method to BERT is OpenAI GPT, which trains a left-to-right Transformer LM on a large text corpus. In fact, many of the design decisions in BERT were intentionally made to make it as close to GPT as possible so that the two methods could be minimally compared. The core argument of this work is that the bi-directionality and the two pre-training tasks presented in Section 3.1 account for the majority of the empirical improvements, but we do note that there are several other differences between how BERT and GPT were trained:

- GPT is trained on the BooksCorpus (800M words); BERT is trained on the BooksCorpus (800M words) and Wikipedia (2,500M words).
- GPT uses a sentence separator (`[SEP]`) and classifier token (`[CLS]`) which are only introduced at fine-tuning time; BERT learns `[SEP]`, `[CLS]` and sentence `A`/`B` embeddings during pre-training.
- GPT was trained for 1M steps with a batch size of 32,000 words; BERT was trained for 1M steps with a batch size of 128,000 words.
- GPT used the same learning rate of 5e-5 for all fine-tuning experiments; BERT chooses a task-specific fine-tuning learning rate which performs the best on the development set.

To isolate the effect of these differences, we perform ablation experiments in Section 5.1 which demonstrate that the majority of the improvements are in fact coming from the two pre-training tasks and the bidirectionality they enable.

A.5 Illustrations of Fine-tuning on Different Tasks

The illustration of fine-tuning BERT on different tasks can be seen in Figure 4. Our task-specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, `[CLS]` is the special symbol for classification output, and `[SEP]` is the special symbol to separate non-consecutive token sequences.

B Detailed Experimental Setup

B.1 Detailed Descriptions for the GLUE Benchmark Experiments.

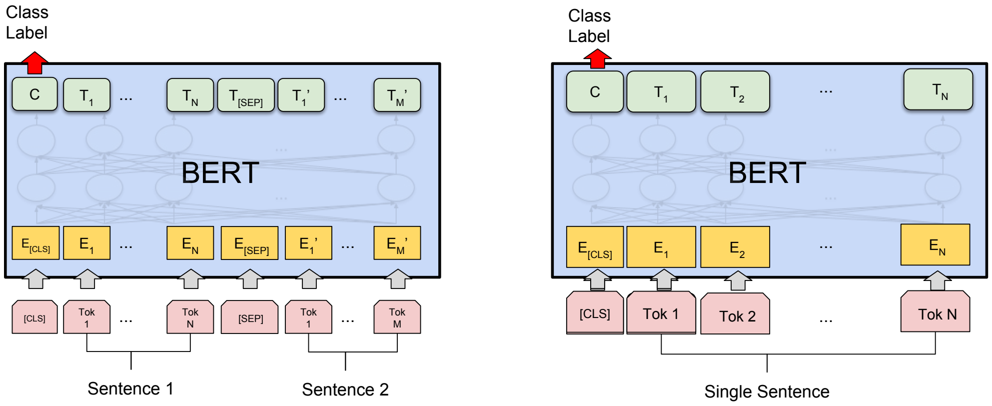
The GLUE benchmark includes the following datasets, the descriptions of which were originally summarized in Wang et al. (2018a):

MNLI Multi-Genre Natural Language Inference is a large-scale, crowdsourced entailment classification task (Williams et al., 2018). Given a pair of sentences, the goal is to predict whether the second sentence is an *entailment*, *contradiction*, or *neutral* with respect to the first one.

QQP Quora Question Pairs is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent (Chen et al., 2018).

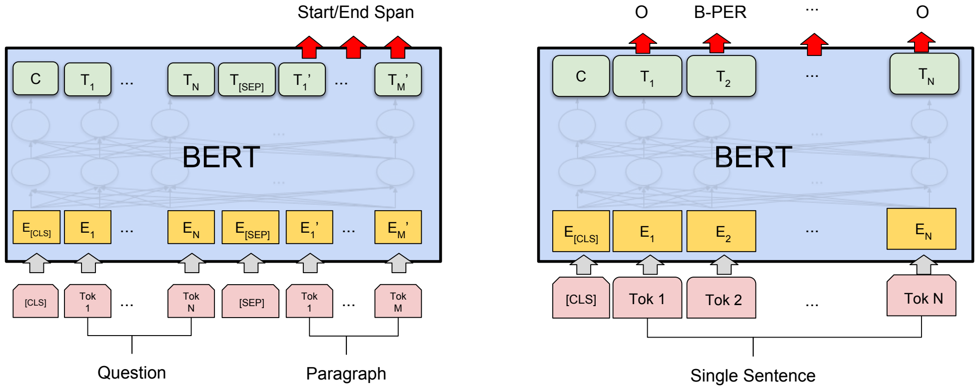
QNLI Question Natural Language Inference is a version of the Stanford Question Answering Dataset (Rajpurkar et al., 2016) which has been converted to a binary classification task (Wang et al., 2018a). The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer.

SST-2 The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

with human annotations of their sentiment (Socher et al., 2013).

CoLA The Corpus of Linguistic Acceptability is a binary single-sentence classification task, where the goal is to predict whether an English sentence is linguistically “acceptable” or not (Warstadt et al., 2018).

STS-B The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines and other sources (Cer et al., 2017). They were annotated with a score from 1 to 5 denoting how similar the two sentences are in terms of semantic meaning.

MRPC Microsoft Research Paraphrase Corpus consists of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent (Dolan and Brockett, 2005).

RTE Recognizing Textual Entailment is a binary entailment task similar to MNLI, but with much less training data (Bentivogli et al., 2009).[14]

WNLI Winograd NLI is a small natural language inference dataset (Levesque et al., 2011). The GLUE webpage notes that there are issues with the construction of this dataset, [15] and every trained system that’s been submitted to GLUE has performed worse than the 65.1 baseline accuracy of predicting the majority class. We therefore exclude this set to be fair to OpenAI GPT. For our GLUE submission, we always predicted the majority class.

---

[14] Note that we only report single-task fine-tuning results in this paper. A multitask fine-tuning approach could potentially push the performance even further. For example, we did observe substantial improvements on RTE from multitask training with MNLI.

[15] `https://gluebenchmark.com/faq`

C Additional Ablation Studies

C.1 Effect of Number of Training Steps

Figure 5 presents MNLI Dev accuracy after fine-tuning from a checkpoint that has been pre-trained for k steps. This allows us to answer the following questions:

1. Question: Does BERT really need such a large amount of pre-training (128,000 words/batch * 1,000,000 steps) to achieve high fine-tuning accuracy?
   Answer: Yes, BERT$_{\text{BASE}}$ achieves almost 1.0% additional accuracy on MNLI when trained on 1M steps compared to 500k steps.

2. Question: Does MLM pre-training converge slower than LTR pre-training, since only 15% of words are predicted in each batch rather than every word?
   Answer: The MLM model does converge slightly slower than the LTR model. However, in terms of absolute accuracy the MLM model begins to outperform the LTR model almost immediately.

C.2 Ablation for Different Masking Procedures

In Section 3.1, we mention that BERT uses a mixed strategy for masking the target tokens when pre-training with the masked language model (MLM) objective. The following is an ablation study to evaluate the effect of different masking strategies.

Note that the purpose of the masking strategies is to reduce the mismatch between pre-training and fine-tuning, as the `[MASK]` symbol never appears during the fine-tuning stage. We report the Dev results for both MNLI and NER. For NER, we report both fine-tuning and feature-based approaches, as we expect the mismatch will be amplified for the feature-based approach as the model will not have the chance to adjust the representations.

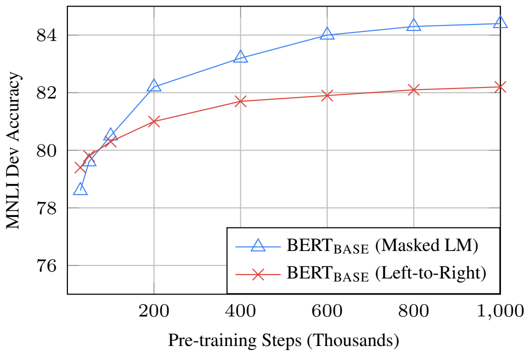


Figure 5: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k.

Masking Rates			Dev Set Results		
Mask	Same	Rnd	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 8: Ablation over different masking strategies.

The results are presented in Table 8. In the table, Mask means that we replace the target token with the `[MASK]` symbol for MLM; Same means that we keep the target token as is; Rnd means that we replace the target token with another random token.

The numbers in the left part of the table represent the probabilities of the specific strategies used during MLM pre-training (BERT uses 80%, 10%, 10%). The right part of the paper represents the Dev set results. For the feature-based approach, we concatenate the last 4 layers of BERT as the features, which was shown to be the best approach in Section 5.3.

From the table it can be seen that fine-tuning is surprisingly robust to different masking strategies. However, as expected, using only the Mask strategy was problematic when applying the feature-based approach to NER. Interestingly, using only the Rnd strategy performs much worse than our strategy as well.

What's in a Name?
Reducing Bias in Bios without Access to Protected Attributes

Alexey Romanov[1], **Maria De-Arteaga**[2], **Hanna Wallach**[3],
Jennifer Chayes[3], **Christian Borgs**[3], **Alexandra Chouldechova**[2],
Sahin Geyik[4], **Krishnaram Kenthapadi**[4], **Anna Rumshisky**[1], **Adam Tauman Kalai**[3]
[1]University of Massachusetts Lowell
`{aromanov,arum}@cs.uml.edu`
[2]Carnegie Mellon University
`mdeartea@andrew.cmu.edu,achould@cmu.edu`
[3]Microsoft Research
`{wallach,jchayes,Christian.Borgs,Adam.Kalai}@microsoft.com`
[4]LinkedIn
`{sgeyik,kkenthapadi}@linkedin.com`

Abstract

There is a growing body of work that proposes methods for mitigating bias in machine learning systems. These methods typically rely on access to protected attributes such as race, gender, or age. However, this raises two significant challenges: (1) protected attributes may not be available or it may not be legal to use them, and (2) it is often desirable to simultaneously consider multiple protected attributes, as well as their intersections. In the context of mitigating bias in occupation classification, we propose a method for discouraging correlation between the predicted probability of an individual's true occupation and a word embedding of their name. This method leverages the societal biases that are encoded in word embeddings, eliminating the need for access to protected attributes. Crucially, it only requires access to individuals' names at training time and not at deployment time. We evaluate two variations of our proposed method using a large-scale dataset of online biographies. We find that both variations simultaneously reduce race and gender biases, with almost no reduction in the classifier's overall true positive rate.

1 Introduction

In recent years, the performance of machine learning systems has improved substantially, leading to the widespread use of machine learning in many domains, including high-stakes domains such as healthcare, employment, and criminal justice (Chalfin et al., 2016; Miotto et al., 2017; Chouldechova, 2017). This increased prevalence has led many people to ask the question, "accurate, but for whom?" (Chouldechova and G'Sell, 2017).

When the performance of a machine learning system differs substantially for different groups of people, a number of concerns arise (Barocas and Selbst, 2016; Kim, 2016). First and foremost, there is a risk that the deployment of such a method may harm already marginalized groups and widen existing inequalities. Recent work highlights this concern in the context of online recruiting and automated hiring (De-Arteaga et al., 2019). When predicting an individual's occupation from their online biography, the authors show that if occupation-specific gender gaps in true positive rates are correlated with existing gender imbalances in those occupations, then those imbalances will be compounded over time—a phenomenon sometimes referred to as the "leaky pipeline." Second, the correlations that lead to performance differences between groups are often irrelevant. For example, while an occupation classifier should predict a higher probability of software engineer if an individual's biography mentions coding experience, there is no good reason for it to predict a lower probability of software engineer if the biography also mentions softball.

Prompted by such concerns about bias in machine learning systems, there is a growing body of work on fairness in machine learning.

"What's in a name? That which we call a rose by any other name would smell as sweet." – *William Shakespeare, Romeo and Juliet.*

Proceedings of NAACL-HLT 2019, pages 4187–4195
Minneapolis, Minnesota, June 2 - June 7, 2019.

Some of the foundational papers in this area highlighted the limitations of trying to mitigate bias using methods that are "unaware" of protected attributes such as race, gender, or age (e.g., Dwork et al., 2012). As a result, subsequent work has primarily focused on introducing fairness constraints, defined in terms of protected attributes, that reduce incentives to rely on undesirable correlations (e.g., Hardt et al., 2016; Zhang et al., 2018). This approach is particularly useful if similar performance can be achieved by slightly different means—i.e., fairness constraints may aid in model selection if there are many near-optima.

In practice, though, any approach that relies on protected attributes may stand at odds with anti-discrimination law, which limits the use of protected attributes in domains such as employment and education, even for the purpose of mitigating bias. And, in other domains, protected attributes are often not available (Holstein et al., 2019). Moreover, even when they are, it is usually desirable to simultaneously consider multiple protected attributes, as well as their intersections. For example, Buolamwini (2017) showed that commercial gender classifiers have higher error rates for women with darker skin tones than for either women or people with darker skin tones overall.

We propose a method for reducing bias in machine learning classifiers without relying on protected attributes. In the context of occupation classification, this method discourages a classifier from learning a correlation between the predicted probability of an individual's occupation and a word embedding of their name. Intuitively, the probability of an individual's occupation should not depend on their name—nor on any protected attributes that may be inferred from it. We present two variations of our method—i.e., two loss functions that enforce this constraint—and show that they simultaneously reduce both race and gender biases with little reduction in classifier accuracy. Although we are motivated by the need to mitigate bias in online recruiting and automated hiring, our method can be applied in any domain where individuals' names are available at training time.

Instead of relying on protected attributes, our method leverages the societal biases that are encoded in word embeddings (Bolukbasi et al., 2016; Caliskan et al., 2017). In particular, we build on the work of Swinger et al. (2019), which showed that word embeddings of names typically reflect the societal biases that are associated with those names, including race, gender, and age biases, as well encoding information about other factors that influence naming practices such as nationality and religion. By using word embeddings of names as a tool for mitigating bias, our method is conceptually simple and empirically powerful. Much like the "proxy fairness" approach of Gupta et al. (2018), it is applicable when protected attributes are not available; however, it additionally eliminates the need to specify which biases are to be mitigated, and allows simultaneous mitigation of multiple biases, including those that relate to group intersections. Moreover, our method only requires access to proxy information (i.e., names) at training time and not at deployment time, which avoids disparate treatment concerns and extends fairness gains to individuals with ambiguous names. For example, a method that explicitly or implicitly infers protected attributes from names at deployment time may fail to correctly infer that an individual named Alex is female and, in turn, fail to mitigate gender bias for her. Methodologically, our work is also similar to that of Zafar et al. (2017), which promotes fairness by requiring that the covariance between a protected attribute and a data point's distance from a classifier's decision boundary is smaller than some constant. However, unlike our method, it requires access to protected attributes, and does not facilitate simultaneous mitigation of multiple biases.

We present our method in Section 2. In section 3, we describe our evaluation, followed by results in Section 4 and conclusions in Section 5.

2 Method

Our method discourages an occupation classifier from learning a correlation between the predicted probability of an individual's occupation and a word embedding of their name. In this section, we present two variations of our method—i.e., two penalties that can be added to an arbitrary loss function and used when training any classifier.

We assume that each data point corresponds to an individual, with a label indicating that individual's occupation. We also assume access to the names of the individuals represented in the training set. The first variation, which we call Cluster Constrained Loss (CluCL), uses k-means to cluster word embeddings of the names in the training set. Then, for each pair of clusters, it minimizes between-cluster disparities in the predicted probabilities of the true labels for the data points that correspond to the names in the clusters. In contrast, the second variation minimizes the covariance between the predicted probability of an individual's occupation and a word embedding of their name. Because this variation minimizes the covariance directly, we call it Covariance Constrained Loss (CoCL). The most salient difference between these variations is that CluCL only minimizes disparities between the latent groups captured by the clusters. For example, if the clusters correspond only to gender, then CluCL is only capable of mitigating gender bias. However, given a sufficiently large number of clusters, CluCL is able to simultaneously mitigate multiple biases, including those that relate to group intersections. For both variations, individual's names are not used as input to the classifier itself; they appear only in the loss function used when training the classifier. The resulting trained classifier can therefore be deployed without access to individuals' names.

2.1 Formulation

We define $x_i = \{x_i^1, \ldots, x_i^M\}$ to be a data point, y_i to be its corresponding (true) label, and n_i^f and n_i^l to be the first and last name of the corresponding individual. The classification task is then to (correctly) predict the label for each data point:

$$p_i = H(x_i) \tag{1}$$

$$\hat{y}_i = \operatorname*{arg\,max}_{1 \leq j \leq |C|} p_i[j], \tag{2}$$

where $H(\cdot)$ is the classifier, C is the set of possible classes, $p_i \in \mathbb{R}^{|C|}$ is the output of the classifier for data point x_i—e.g., $p_i[j]$ is the predicted probability of x_i belonging to class j—and $\hat{y}_i$ is the predicted label for x_i. We define p_i^y to be the predicted probability of y_i—i.e., the true label for x_i.

The conventional way to train such a classifier is to minimize some loss function $\mathcal{L}$, such as the cross-entropy loss function. Our method simply adds an additional penalty to this loss function:

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda \cdot \mathcal{L}_{\text{CL}}, \tag{3}$$

where $\mathcal{L}_{\text{CL}}$ is either $\mathcal{L}_{\text{CluCL}}$ or $\mathcal{L}_{\text{CoCL}}$ (defined in Sections 2.2 and 2.3, respectively), and λ is a hyperparameter that determines the strength of the penalty. This loss function is only used during training, and plays no role in the resulting trained classifier. Moreover, it can be used in any standard setup for training a classifier—e.g., training a deep neural network using mini-batches and the Adam optimization algorithm (Kingma and Ba, 2014).

2.2 Cluster Constrained Loss

This variation represents each first name n_i^f and last name n_i^l as a pair of low-dimensional vectors using a set of pretrained word embeddings E. These are then combined to form a single vector:

$$n_i^e = \frac{1}{2}\left(E[n_i^f] + E[n_i^l]\right). \tag{4}$$

Using k-means (Arthur and Vassilvitskii, 2007), CluCL then clusters the resulting embeddings into k clusters, yielding a cluster assignment k_i for each name (and corresponding data point). Next, for each class $c \in C$, CluCL computes the following average pairwise difference between clusters:

$$l_c = \frac{1}{k(k-1)} \times \sum_{u,v=1}^{k} \left(\frac{1}{N_{c,u}} \sum_{\substack{i:y_i=c,\\ k_i=u}} p_i^y - \frac{1}{N_{c,v}} \sum_{\substack{i:y_i=c,\\ k_i=v}} p_i^y \right)^2, \tag{5}$$

where u and v are clusters and $N_{c,u}$ is the number of data points in cluster u for which $y_i = c$. CluCL considers each class individually because different classes will likely have different numbers of training data points and different disparities. Finally, CluCL computes the average of $l_1, \ldots l_{|C|}$ to yield

$$\mathcal{L}_{\text{CluCL}} = \frac{1}{|C|} \sum_{c \in C} l_c. \tag{6}$$

2.3 Covariance Constrained Loss

This variation minimizes the covariance between the predicted probability of a data point's label and the corresponding individual's name. Like CluCL, CoCL represents each name as a single vector n_i^e and considers each class individually:

$$l_c = \mathbb{E}_{i:y_i=c}\left[\left(p_i^y - \mu_p^c\right) \cdot \left(n_i^e - \mu_n^c\right)\right], \tag{7}$$

where $\mu_p^c = \mathbb{E}_{i:y_i=c}[p_i^y]$ and $\mu_n^c = \mathbb{E}_{i:y_i=c}[n_i^e]$. Finally, CoCL computes the following average:

$$\mathcal{L}_{\text{CoCL}} = \frac{1}{|C|} \sum_{c \in C} \|l_c\|,$$

where $\|\cdot\|$ is the ℓ_2 norm.

3 Evaluation

One of our method's strengths is its ability to simultaneously mitigate multiple biases without access to protected attributes; however, this strength also poses a challenge for evaluation. We are unable to quantify this ability without access to these attributes. To facilitate evaluation, we focus on race and gender biases only because race and gender attributes are more readily available than attributes corresponding to other biases. We further conceptualize both race and gender to be binary ("white/non-white" and "male/female") but note that these conceptualizations are unrealistic, reductive simplifications that fail to capture many aspects of race and gender, and erase anyone who does not fit within their assumptions. We emphasize that we use race and gender attributes only for evaluation—they do not play a role in our method.

3.1 Datasets

We use two datasets to evaluate our method: the adult income dataset from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017), where the task is to predict whether an individual earns more than \$50k per year (i.e., whether their occupation is "high status"), and a dataset of online biographies (De-Arteaga et al., 2019), where the task is to predict an individual's occupation from the text of their online biography.

Each data point in the *Adult* dataset consists of a set of binary, categorical, and continuous attributes, including race and gender. We preprocess these attributes to more easily allow us to understand the classifier's decisions. Specifically, we normalize continuous attributes to be in the range $[0, 1]$ and we convert categorical attributes into binary indicator variables. Because the data points do not have names associated with them, we generate synthetic first names using the race and gender attributes. First, we use the dataset of Tzioumis (2018) to identify "white" and "non-white" names. For each name, if the proportion of "white" people with that name is higher than 0.5, we deem the name to be "white;" otherwise, we deem it to be "non-white."[1] Next, we use Social Security Administration data about baby names (2018) to identify "male" and "female" names. For each name, if the proportion of boys with that name is higher than 0.5, we deem the name to be "male;" otherwise, we deem it to be "female."[2] We then take the intersection of these two sets of names to yield a single set of names that is partitioned into four non-overlapping categories by (binary) race and gender. Finally, we generate a synthetic first name for each data point by sampling a name from the relevant category.

Each data point in the *Bios* dataset consists of the text of an individual's biography, written in the third person. We represent each biography as a vector of length V, where V is the size of the vocabulary. Each element corresponds to a single word type and is equal to 1 if the biography contains that type (and 0 otherwise). We limit the size of the vocabulary by discarding the 10% most common word types, as well as any word types that occur fewer than twenty times. Unlike the *Adult* dataset, each data point has a name associated with it. And, because biographies are typically written in the third person and because pronouns are gendered in English, we can extract (likely) self-identified gender. We infer race for each data point by sampling from a Bernoulli distribution with probability equal to the average of the probability that an individual with that first name is "white" (from the dataset of Tzioumis (2018), using a threshold of 0.5, as described above) and the probability that an individual with that last name is "white" (from the dataset of Comenetz (2016), also using a threshold of 0.5).[3] Finally, like De-Arteaga et al. (2019), we consider two versions of the *Bios* dataset: one where first names and pronouns are available to the classifier and one where they are "scrubbed."

Throughout our evaluation, we use the fastText word embeddings, pretrained on Common Crawl data (Bojanowski et al., 2016), to represent names.

3.2 Classifier and Loss Function

Our method can be used with any classifier, including deep neural networks such as recurrent neural networks and convolutional neural networks. However, because the focus of this paper is mitigating bias, not maximizing classifier

---

[1] For 90% of the names, the proportion of "white" people with that name is greater than 0.7 or less than 0.3, so there is a clear distinction between "white" and "non-white" names.

[2] For 98% of the names, the proportion of boys with that name is greater than 0.7 or less than 0.3, so there is an even clearer distinction between "male" and "female" names.

[3] We note that, in general, an individual's race or gender should be directly reported by the individual in question; inferring race or gender can be both inaccurate and reductive.

accuracy, we use a single-layer neural network:

$$h_i = W_h \cdot x_i + b_h$$
$$p_i = \text{softmax}(h_i)$$

where $W_h \in \mathbb{R}^{|C| \times M}$ and $b_h \in \mathbb{R}^{|C|}$ are the weights. This structure allows us to examine individual elements of the matrix W_h in order to understand the classifier's decisions for any dataset.

Both the *Adult* dataset and the *Bios* dataset have a strong class imbalance. We therefore use a weighted cross-entropy loss as $\mathcal{L}$, with weights set to the values proposed by King and Zeng (2001).

3.3 Quantifying Bias

To quantify race bias and gender bias, we follow the approach proposed by De-Arteaga et al. (2019) and compute the true positive rate (TPR) race gap and the TPR gender gap—i.e., the differences in the TPRs between races and between genders, respectively—for each occupation. The TPR race gap for occupation c is defined as follows:

$$\text{TPR}_{r,c} = P\left[\hat{Y} = c \,|\, R = r, Y = c\right] \quad (8)$$
$$\text{Gap}_{r,c} = \text{TPR}_{r,c} - \text{TPR}_{\sim r,c}, \quad (9)$$

where r and $\sim r$ are binary races, $\hat{Y}$ and Y are random variables representing the predicted and true occupations for an individual, and R is a random variable representing that individual's race. Similarly, the TPR gender gap for occupation c is

$$\text{TPR}_{g,c} = P\left[\hat{Y} = c \,|\, G = g, Y = c\right] \quad (10)$$
$$\text{Gap}_{g,c} = \text{TPR}_{g,c} - \text{TPR}_{\sim g,c}, \quad (11)$$

where g and $\sim g$ are binary genders and G is a random variable representing an individual's gender.

To obtain a single score that quantifies race bias, thus facilitating comparisons, we calculate the root mean square of the per-occupation TPR race gaps:

$$\text{Gap}_r^{\text{RMS}} = \sqrt{\frac{1}{|C|} \sum_{c \in C} \text{Gap}_{r,c}^2}. \quad (12)$$

We obtain a single score that quantifies gender bias similarly. The motivation for using the root mean square instead of an average is that larger values have a larger effect and we are more interested in mitigating larger biases. Finally, to facilitate worst-case analyses, we calculate the maximum TPR race gap and the maximum TPR gender gap.

We again emphasize that race and gender attributes are used only for evaluating our method.

4 Results

We first demonstrate that word embeddings of names encode information about race and gender. We then present the main results of our evaluation, before examining individual elements of the matrix W_h in order to better understand our method.

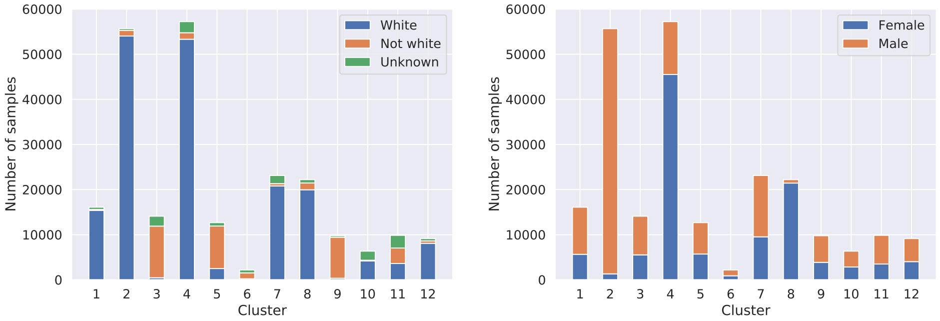
4.1 Word Embeddings of Names as Proxies

We cluster the names associated with the data points in the *Bios* dataset, represented as word embeddings, to verify that such embeddings indeed capture information about race and gender. We perform k-means clustering (using the k-means++ algorithm) with $k = 12$ clusters, and then plot the number of data points in each cluster that correspond to each (inferred) race and gender. Figures 1a and 1b depict these numbers, respectively.

Clusters 1, 2, 4, 7, 8, and 12 contain mostly "white" names, while clusters 3, 5, and 9 contain mostly "non-white names." Similarly, clusters 4 and 8 contain mostly "female" names, while cluster 2 contains mostly "male" names. The other clusters are more balanced by race and gender. Manual inspection of the clusters reveals that cluster 9 contains mostly Asian names, while cluster 8 indeed contains mostly "female" names. The names in cluster 2 are mostly "white" and "male," while the names in cluster 4 are mostly "white" and "female." This suggests that the clusters are capturing at least some intersections. Together these results demonstrate that word embeddings of names do indeed encode at least some information about race and gender, even when first and last names are combined into a single embedding vector. For a longer discussion of the societal biases reflected in word embeddings of names, we recommend the work of Swinger et al. (2019).

4.2 *Adult* Dataset

The results of our evaluation using the *Adult* dataset are shown in Table 1. The task is to predict whether an individual earns more than $50k per year (i.e., whether their occupation is "high status"). Because the dataset has a strong class imbalance, we report the balanced TPR—i.e., we compute the per-class TPR and then average over the classes. We experiment with different values of the hyperparameter λ. When $\lambda = 0$, the method is equivalent to using the conventional weighted cross-entropy loss function. Larger values of λ increase the strength of the penalty, but may lead to



(a) Race.

(b) Gender.

Figure 1: Number of data points (from the *Bios* dataset) in each cluster that correspond to each race and gender.

Method	λ	Balanced TPR	$\mathrm{Gap}_g^{\mathrm{RMS}}$	$\mathrm{Gap}_r^{\mathrm{RMS}}$	$\mathrm{Gap}_g^{\max}$	$\mathrm{Gap}_r^{\max}$
None	0	**0.795**	0.299	0.120	0.303	0.148
CluCL	1	0.788	0.278	0.121	0.297	0.145
CluCL	2	0.793	0.259	0.085	0.282	0.114
CoCL	1	0.794	0.215	0.091	0.251	0.119
CoCL	2	0.790	**0.163**	**0.080**	**0.201**	**0.109**

Table 1: Results for the *Adult* dataset. Balanced TPR (i.e., per-occupation TPR, averaged over occupations), gender bias quantified as $\mathrm{Gap}_g^{\mathrm{RMS}}$, race bias quantified as $\mathrm{Gap}_r^{\mathrm{RMS}}$, maximum TPR gender gap, and maximum TPR race gap for different values of hyperparameter λ. Results are averaged over four runs with different random seeds.

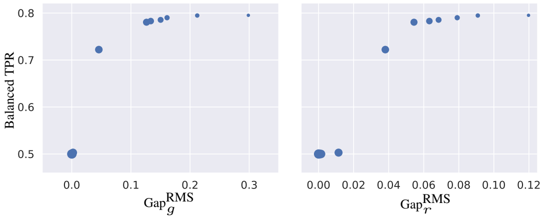


Figure 2: Gender bias quantified as $\mathrm{Gap}_g^{\mathrm{RMS}}$ (left) and race bias quantified as $\mathrm{Gap}_r^{\mathrm{RMS}}$ (right) versus balanced TPR for the CoCL variation of our method with different values of hyperparameter λ (a larger dot means a larger value of λ) for the *Adult* dataset. Results are averaged over four runs with different random seeds.

a less accurate classifier. Using $\lambda = 0$ leads to significant gender bias: the maximum TPR gender gap is 0.303. This means that the TPR is 30% higher for men than for women. We emphasize that this does *not* mean that the classifier is more likely to predict that a man earns more than \$50k per year, but means that the classifier is more likely to *correctly* predict that a man earns more than \$50k per year. Both variations of our method significantly reduce race and gender biases. With CluCL, the root mean square TPR race gap is reduced from 0.12 to 0.085, while the root mean square TPR gender gap is reduced from 0.299 to 0.25. These reductions in bias result in less than one percent decrease in the balanced TPR (79.5% is decreased to 79.3%). With CoCL, the race and gender biases are further reduced: the root mean square TPR race gap is reduced to 0.08, while the root mean square TPR gender gap is reduced to 0.163, with 0.5% decrease in the balanced TPR.

We emphasize that although our proposed method significantly reduces race and gender biases, neither variation can completely eliminate them. In order to understand how different values of hyperparameter λ influence the reduction in race and gender biases, we perform additional experiments using CoCL where we vary λ from 0 to 10. Figure 2 depicts these results. Larger values of λ indeed reduce race and gender biases; however, to achieve a root mean square TPR gender gap of zero means reducing the balanced TPR to 50%, which is unacceptably low. That said, there are a wide range of values of λ that significantly reduce race and gender biases, while maintaining an acceptable balanced TPR. For example, $\lambda = 6$ results in a root mean square TPR race gap of 0.038 and a root mean square TPR gender gap of 0.046, with only a 7.3% decrease in the balanced TPR.

Method	λ	Balanced TPR	$\text{Gap}_g^{\text{RMS}}$	$\text{Gap}_r^{\text{RMS}}$	$\text{Gap}_g^{\max}$	$\text{Gap}_r^{\max}$
None	0	**0.788**	0.173	0.051	0.511	0.121
CluCL	1	0.784	0.168	0.048	0.494	0.120
CluCL	2	0.781	**0.165**	**0.047**	**0.486**	0.114
CoCL	1	0.785	0.168	0.048	0.507	**0.109**
CoCL	2	0.779	0.169	0.048	0.512	0.116

Table 2: Results for the original *Bios* dataset. Balanced TPR (i.e., per-occupation TPR, averaged over occupations), gender bias quantified as $\text{Gap}_g^{\text{RMS}}$, race bias quantified as $\text{Gap}_r^{\text{RMS}}$, maximum TPR gender gap, and maximum TPR race gap for different values of hyperparameter λ. Results are averaged over four runs with different random seeds.

Method	λ	Balanced TPR	$\text{Gap}_g^{\text{RMS}}$	$\text{Gap}_r^{\text{RMS}}$	$\text{Gap}_g^{\max}$	$\text{Gap}_r^{\max}$
None	0	**0.785**	0.111	0.049	0.385	0.123
CluCL	1	0.782	**0.107**	0.048	**0.383**	0.112
CluCL	2	0.778	0.112	0.046	0.395	**0.107**
CoCL	1	0.780	0.109	0.047	0.388	0.117
CoCL	2	0.775	0.108	**0.046**	0.387	0.109

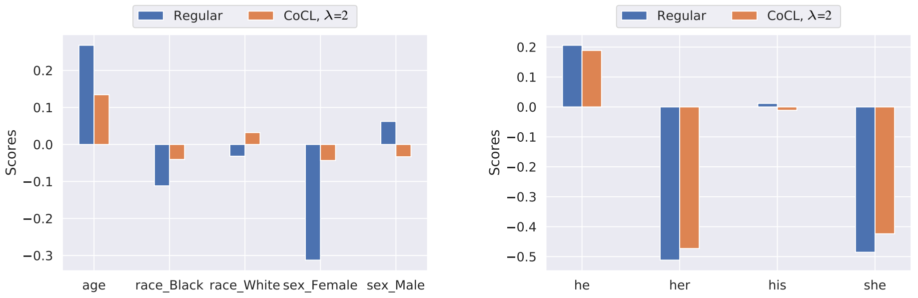
Table 3: Results for the "scrubbed" *Bios* dataset. Balanced TPR (i.e., per-occupation TPR, averaged over occupations), gender bias quantified as $\text{Gap}_g^{\text{RMS}}$, race bias quantified as $\text{Gap}_r^{\text{RMS}}$, maximum TPR gender gap, and maximum TPR race gap for different values of hyperparameter λ. Again, results are averaged over four runs.

4.3 *Bios* Dataset

The results of our evaluation using the original and "scrubbed" (i.e., names and pronouns are "scrubbed") versions of the *Bios* dataset are shown in Tables 2 and 3, respectively. The task is to predict an individual's occupation from the text of their online biography. Because the dataset has a strong class imbalance, we again report the balanced TPR. CluCL and CoCL reduce race and gender biases for both versions of the dataset. For the original version, CluCL reduces the root mean square TPR gender gap from 0.173 to 0.165 and the maximum TPR gender gap by 2.5%. Race bias is also reduced, though to a lesser extent. These reductions reduce the balanced TPR by 0.7%. For the "scrubbed" version, the reductions in race and gender biases are even smaller, likely because most of the information about race and gender has been removed by "scrubbing" names and pronouns. We hypothesize that these smaller reductions in race and gender biases, compared to the *Adult* dataset, are because the *Adult* dataset has fewer attributes and classes than the *Bios* dataset, and contains explicit race and gender information, making the task of reducing biases much simpler. We also note that each biography in the *Bios* dataset is represented as a vector of length V, where V is over 11,000. This means that the corresponding classifier has a very large number of weights, and there is a strong overfitting effect. Because this overfitting effect increases with λ, we suspect it explains why CluCL has a larger root mean square TPR gender gap when $\lambda = 2$ than when $\lambda = 1$. Indeed, the root mean square TPR gender gap for the training set is 0.05 when $\lambda = 2$. Using dropout and ℓ_2 weight regularization lessened this effect, but did not eliminate it entirely.

4.4 Understanding the Method

Our method mitigates bias by making training-time adjustments to the classifier's weights that minimize the correlation between the predicted probability of an individual's occupation and a word embedding of their name. Because of our choice of classifier (a single-layer neural network, as described in Section 3.2), we can examine individual elements of the matrix W_h to understand the effect of our method on the classifier's decisions. Figure 3a depicts the values of several weights for the conventional weighted cross-entropy loss function (i.e., $\lambda = 0$) and for CoCL with $\lambda = 2$ for the *Adult* dataset. When $\lambda = 0$, the attributes "sex_Female" and "sex_Male" have large negative and positive weights, respectively. This means that the classifier is more likely to predict that a man earns more than \$50k per year. With CoCL, these weights are much closer to zero. Similarly, the weights for the race attributes are also closer to zero. We note that the weight for



(a) *Adult* dataset.

(b) *Bios* dataset, occupation "surgeon."

Figure 3: Classifier weight values for several attributes for the conventional weighted cross-entropy loss function (i.e., $\lambda = 0$) and for CoCL with $\lambda = 2$. Results are averaged over four runs with different random seeds.

the attribute "age" is also reduced, suggesting that CoCL may have mitigated some form of age bias.

Figure 3b depicts the values of several weights specific to the occupation "surgeon" for the conventional weighted cross-entropy loss function (i.e., $\lambda = 0$) and for CoCL with $\lambda = 2$ for the original version of the *Bios* dataset. When $\lambda = 0$, the attributes "she" and "her" have large negative weights, while the attribute "he" has a positive weight. This means that the classifier is less likely to predict that a biography that contains the words "she" or "her" belongs to a surgeon. With CoCL, these magnitudes of these weights are reduced, though these reductions are not as significant as the reductions shown for the *Adult* dataset.

5 Conclusion

In this paper, we propose a method for reducing bias in machine learning classifiers without relying on protected attributes. In contrast to previous work, our method eliminates the need to specify which biases are to be mitigated, and allows simultaneous mitigation of multiple biases, including those that relate to group intersections. Our method leverages the societal biases that are encoded in word embeddings of names. Specifically, it discourages an occupation classifier from learning a correlation between the predicted probability of an individual's occupation and a word embedding of their name. We present two variations of our method, and evaluate them using a large-scale dataset of online biographies. We find that both variations simultaneously reduce race and gender biases, with almost no reduction in the classifier's overall true positive rate. Our method is conceptually simple and empirically powerful, and can be used with any classifier, including deep neural networks. Finally, although we focus on English, we expect our method will work well for other languages, but leave this direction for future work.

References

Social Security Administration. 2018. Baby names from social security card applications - national level data. Accessed 5 July 2018.

David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

Solon Barocas and Andrew D Selbst. 2016. Big data's disparate impact. *Cal. L. Rev.*, 104:671.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Joy Adowaa Buolamwini. 2017. *Gender shades: intersectional phenotypic and demographic evaluation of face datasets and gender classifiers*. Ph.D. thesis, Massachusetts Institute of Technology.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Aaron Chalfin, Oren Danieli, Andrew Hillis, Zubin Jelveh, Michael Luca, Jens Ludwig, and Sendhil Mullainathan. 2016. Productivity and selection of human capital with machine learning. *American Economic Review*, 106(5):124–27.

Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163.

Alexandra Chouldechova and Max G'Sell. 2017. Fairer and more accurate, but for whom? *arXiv preprint arXiv:1707.00046*.

Joshua Comenetz. 2016. Frequently occurring surnames in the 2010 census. *United States Census Bureau*.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 120–128. ACM.

Dua Dheeru and Efi Karra Taniskidou. 2017. UCI machine learning repository.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 214–226, New York, NY, USA. ACM.

Maya Gupta, Andrew Cotter, Mahdi Milani Fard, and Serena Wang. 2018. Proxy fairness. *arXiv preprint arXiv:1806.11212*.

Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323.

K. Holstein, J. Wortman Vaughan, H. Daumé III, M. Dudík, and H. Wallach. 2019. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*.

Pauline T Kim. 2016. Data-driven discrimination at work. *Wm. & Mary L. Rev.*, 58:857.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political Analysis*, 9:137–163.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. 2017. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246.

Nathaniel Swinger, Maria De-Arteaga, IV Heffernan, Neil Thomas, Mark DM Leiserson, and Adam Tauman Kalai. 2019. What are the biases in my word embedding? *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*.

Konstantinos Tzioumis. 2018. Demographic aspects of first names. *Scientific data*, 5:180025.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340. ACM.

Author Index